

mlpack
3.1.0

Generated by Doxygen 1.8.13

Contents

1	mlpack Documentation	1
1.1	Introduction	1
1.2	How To Use This Documentation	1
1.3	Tutorials	1
1.4	Final Remarks	2
2	mlpack automatic bindings to other languages	3
2.1	Overview	3
2.2	Introduction	4
2.3	Writing code that can be turned into a binding	5
2.4	How to write mlpack bindings	7
2.4.1	Documenting a program with PROGRAM_INFO()	8
2.4.2	Defining parameters for a program	10
2.4.3	Using CLI in an mlpackMain() function	13
2.4.4	More documentation on using CLI	14
2.5	Structure of CLI module and associated macros	14
2.6	Command-line program bindings	16
2.6.1	mlpackMain() definition	16
2.6.2	Matrix and model parameter handling	17
2.6.3	Parsing the command line	17
2.7	Python bindings	18
2.7.1	Passing matrices to/from Python	18
2.7.2	Passing model parameter to/from Python	18
2.7.3	CMake generation of setup.py	19
2.7.4	Generation of .pyx files	19
2.7.5	Building the .pyx files	19
2.7.6	Testing the Python bindings	19
2.8	Adding new binding types	20

3	Building mlpack From Source	21
3.1	Introduction	21
3.2	Simple Linux build instructions	22
3.3	Creating Build Directory	22
3.4	Dependencies of mlpack	22
3.5	Configuring CMake	23
3.6	Building mlpack	24
3.7	Installing mlpack	25
3.8	Using mlpack without installing	25
4	Building mlpack From Source on Windows	27
4.1	Introduction	27
4.2	Build Environment	27
4.3	Pre-requisites	28
4.4	Windows build instructions	28
4.5	Dependencies	28
4.6	Building mlpack	29
4.7	Appendix	30
4.8	Additional Information	30
5	mlpack command-line quickstart guide	31
5.1	Introduction	31
5.2	Installing mlpack	31
5.3	Simple mlpack quickstart example	32
5.4	What else does mlpack implement?	32
5.5	Using mlpack for movie recommendations	33
5.6	Next steps with mlpack	34

6	Cross-Validation	35
6.1	Introduction	35
6.2	Simple cross-validation examples	36
6.2.1	10-fold cross-validation on softmax regression	36
6.2.2	10-fold cross-validation on weighted decision trees	37
6.2.3	10-fold cross-validation with categorical decision trees	37
6.2.4	Simple cross-validation for linear regression	38
6.3	Performance measures	38
6.4	The KFoldCV and SimpleCV classes	39
6.4.1	The KFoldCV and SimpleCV constructors	39
6.4.2	The Evaluate() method	40
6.5	Further references	41
7	File formats and loading data in mlpack	43
7.1	Introduction	43
7.2	Simple examples to load data in C++	43
7.3	Supported dataset types	44
7.4	Loading simple matrices in C++	45
7.5	Dealing with sparse matrices	45
7.6	Categorical features and command line programs	46
7.7	Categorical features and C++	46
7.8	Loading and saving models	47
7.9	Loading and saving models in C++	48
7.10	Final notes	48
8	Hyper-Parameter Tuning	49
8.1	Introduction	49
8.2	Basic Usage	49
8.3	Fixed Arguments	50
8.4	Gradient-Based Optimization	51
8.5	The HyperParameterTuner class	51
8.6	Further documentation	52

9	Writing an mlpack binding	53
9.1	Introduction	53
9.2	Simple Logging Example	53
9.3	Simple CLI Example	55
10	Matrices in mlpack	57
10.1	Introduction	57
10.2	Column-major Matrices	57
10.3	Loading Matrices	58
11	mlpack in Python quickstart guide	59
11.1	Introduction	59
11.2	Installing mlpack	59
11.3	Simple mlpack quickstart example	60
11.4	What else does mlpack implement?	60
11.5	Using mlpack for movie recommendations	61
11.6	Next steps with mlpack	61
12	Simple Sample mlpack Programs	63
12.1	Introduction	63
12.2	Covariance Computation	63
12.3	Nearest Neighbor	64
12.4	Other examples	64
13	Sample C++ ML App for Windows	65
13.1	Introduction	65
13.2	Creating the VS project	65
13.3	Project Configuration	65
13.4	The app goal	66
13.5	Headers and namespaces	66
13.6	Loading the dataset	67
13.7	Training	67
13.8	Cross-Validating	68
13.9	Saving the model	68
13.10	Loading the model	68
13.11	Classifying a new sample	68
13.12	Final thoughts	68

14 mlpack Timers	69
14.1 Introduction	69
14.2 Timer API	69
14.3 Timer Example	70
15 mlpack version information	71
15.1 mlpack versions in code	71
15.2 mlpack executable versions	71
16 Alternating Matrix Factorization tutorial	73
16.1 Introduction	73
16.2 Table of Contents	73
16.3 The 'AMF' class	74
16.3.1 Using different termination policies	74
16.3.2 Using different initialization policies	75
16.3.3 Using different update rules	75
16.3.4 Using Non-Negative Matrix Factorization with AMF	76
16.3.5 Using Singular Value Decomposition with AMF	76
16.4 Further documentation	76
17 Neural Network tutorial	77
17.1 Introduction	77
17.2 Table of Contents	77
17.3 Model API	78
17.4 Layer API	81
17.5 Model Setup & Training	83
17.6 Saving & Loading	84
17.7 Extracting Parameters	86
17.8 Further documentation	86

18 Approximate furthest neighbor search (mlpack_approx_kfn) tutorial	87
18.1 Introduction	87
18.2 Table of Contents	88
18.3 Which algorithm should be used?	89
18.4 Command-line 'mlpack_approx_kfn' and 'mlpack_kfn'	90
18.4.1 Calculate 5 furthest neighbors with default options	90
18.4.2 Specifying algorithm parameters for DrusillaSelect	91
18.4.3 Using QDAFN instead of DrusillaSelect	91
18.4.4 Printing results quality with exact distances	92
18.4.5 Using cached exact distances for quality results	93
18.4.6 Using tree-based approximation with mlpack_kfn	93
18.4.7 Different algorithms with 'mlpack_kfn'	94
18.4.8 Saving a model for later use	95
18.4.9 Final command-line program notes	96
18.5 DrusillaSelect C++ class	96
18.5.1 Approximate furthest neighbors with defaults	96
18.5.2 Custom numbers of tables and projections	97
18.5.3 Accessing the candidate set	97
18.5.4 Retraining on a new reference set	98
18.5.5 Running on sparse data	98
18.6 QDAFN C++ class	98
18.6.1 Approximate furthest neighbors with defaults	99
18.6.2 Custom numbers of tables and projections	99
18.6.3 Accessing the candidate set	99
18.6.4 Retraining on a new reference set	100
18.6.5 Running on sparse data	100
18.7 KFN C++ class	100
18.7.1 Simple furthest neighbors example	101
18.7.2 Retraining on a new reference set	101
18.7.3 Searching in single-tree mode	101
18.7.4 Searching in brute-force mode	102
18.8 Further documentation	102

19 Collaborative filtering tutorial	103
19.1 Introduction	103
19.2 Table of Contents	104
19.3 The 'mlpack_cf' program	104
19.3.1 Input format for mlpack_cf	105
19.3.2 mlpack_cf with default parameters	105
19.3.3 Saving mlpack_cf models	106
19.3.4 Loading mlpack_cf models	106
19.3.5 Specifying rank of mlpack_cf decomposition	106
19.3.6 mlpack_cf with single-user recommendation	106
19.3.7 mlpack_cf with non-default factorizer	107
19.3.8 mlpack_cf with non-default neighborhood size	107
19.4 The 'CF' class	107
19.4.1 CF with default parameters	108
19.4.2 CF with other factorizers	108
19.4.3 Predicting individual user/item ratings	109
19.4.4 Other operations with the W and H matrices	109
19.5 Template parameters for the 'CF' class	109
19.6 Further documentation	110
20 Density Estimation Tree (DET) tutorial	111
20.1 Introduction	111
20.2 Table of Contents	111
20.3 Command-Line mlpack_det	112
20.3.1 Plain-vanilla density estimation	113
20.3.2 Estimation on a test set	113
20.3.3 Computing the variable importance	114
20.3.4 Saving trained DETs	114
20.3.5 Loading trained DETs	114
20.4 The 'DTree' class	114
20.4.1 Public Functions	115
20.5 'namespace mlpack::det'	115
20.5.1 Utility Functions	116
20.6 Further Documentation	116

21 EMST Tutorial	117
21.1 Introduction	117
21.2 Table of Contents	118
21.3 Command-Line 'EMST'	118
21.4 The 'DualTreeBoruvka' class	119
21.5 Further documentation	119
22 Fast max-kernel search tutorial (fastmks)	121
22.1 Introduction	121
22.2 Table of Contents	122
22.3 Command-line FastMKS (mlpack_fastmks)	122
22.3.1 FastMKS with a linear kernel on one dataset	123
22.3.2 FastMKS on a reference and query dataset	124
22.3.3 FastMKS with a different kernel	124
22.3.4 Using single-tree search or naive search	124
22.3.5 Parameters for alternate kernels	124
22.3.6 Saving a FastMKS model/tree	125
22.3.7 Loading a FastMKS model for further searches	125
22.4 The 'FastMKS' class	125
22.4.1 FastMKS on one dataset	126
22.4.2 FastMKS with a query and reference dataset	126
22.4.3 FastMKS with an initialized kernel	126
22.4.4 FastMKS with an already-created tree	127
22.5 Writing a custom kernel for FastMKS	128
22.6 Using other tree types for FastMKS	128
22.7 Running FastMKS on objects	128
22.8 Further documentation	129

23 K-Means tutorial (kmeans)	131
23.1 Introduction	131
23.2 Table of Contents	132
23.3 Command-Line 'kmeans'	132
23.3.1 Simple k-means clustering	133
23.3.2 Saving the resulting centroids	133
23.3.3 Allowing empty clusters	133
23.3.4 Allowing empty clusters	133
23.3.5 Limiting the maximum number of iterations	133
23.3.6 Using Bradley-Fayyad "refined start"	134
23.3.7 Using different k-means algorithms	134
23.4 The 'KMeans' class	135
23.4.1 Running k-means and getting cluster assignments	135
23.4.2 Running k-means and getting centroids of clusters	135
23.4.3 Limiting the maximum number of iterations	136
23.4.4 Setting initial cluster assignments	136
23.4.5 Setting initial cluster centroids	137
23.4.6 Running sparse k-means	138
23.5 Template parameters for the 'KMeans' class	138
23.5.1 Changing the distance metric used for k-means	139
23.5.2 Changing the initial partitioning strategy used for k-means	139
23.5.3 Changing the action taken when an empty cluster is encountered	140
23.5.4 The LloydStepType template parameter	141
23.6 Further documentation	141

24 Linear/ridge regression tutorial (mlpack_linear_regression)	143
24.1 Introduction	143
24.2 Table of Contents	144
24.3 Command-Line 'mlpack_linear_regression'	144
24.3.1 One file, generating the function coefficients	145
24.3.2 Compute model and predict at the same time	145
24.3.3 Prediction using a precomputed model	146
24.3.4 Using ridge regression	147
24.4 The 'LinearRegression' class	148
24.4.1 Generating a model	148
24.4.2 Setting a model	148
24.4.3 Load a model from a file	148
24.4.4 Prediction	149
24.4.5 Setting lambda for ridge regression	149
24.5 Further documentation	149
25 NeighborSearch tutorial (k-nearest-neighbors)	151
25.1 Introduction	151
25.2 Table of Contents	151
25.3 Command-Line 'mlpack_knn'	152
25.3.1 One dataset, 5 nearest neighbors	152
25.3.2 Query and reference dataset, 10 nearest neighbors	153
25.3.3 One dataset, 3 nearest neighbors, leaf size of 15 points	154
25.4 The 'KNN' class	154
25.4.1 5 nearest neighbors on a single dataset	155
25.4.2 10 nearest neighbors on a query and reference dataset	155
25.4.3 Naive (exhaustive) search for 6 nearest neighbors on one dataset	155
25.5 The extensible 'NeighborSearch' class	156
25.5.1 SortPolicy policy class	156
25.5.2 MetricType policy class	156
25.5.3 MatType policy class	157
25.5.4 TreeType policy class	157
25.5.5 TraverserType policy class	157
25.6 Further documentation	157

26 RangeSearch tutorial (mlpack_range_search)	159
26.1 Introduction	159
26.2 Table of Contents	159
26.3 The 'mlpack_range_search' command-line executable	160
26.3.1 One dataset, points with distance ≤ 0.01	160
26.3.2 Query and reference dataset, range [1.0, 1.5]	161
26.3.3 One dataset, range [0.7 0.8], leaf size of 15 points	162
26.4 The 'RangeSearch' class	162
26.4.1 Distance less than 2.0 on a single dataset	163
26.4.2 Range [3.0, 4.0] on a query and reference dataset	163
26.4.3 Naive (exhaustive) search for distance greater than 5.0 on one dataset	163
26.5 The extensible 'RangeSearch' class	164
26.5.1 MetricType policy class	164
26.5.2 MatType policy class	164
26.5.3 TreeType policy class	164
26.6 Further documentation	165
27 Tutorials	167
27.1 Quickstart Tutorials	167
27.2 Introductory Tutorials	167
27.3 Method-specific Tutorials	168
27.4 Advanced Tutorials	168
27.5 Policy Class Documentation	168
28 The ElemType policy in mlpack	169
28.1 Overview	169
28.2 note for developers	169

29 The FunctionType policy in mlpack	171
29.1 Overview	171
29.2 Interface requirements	171
30 The KernelType policy in mlpack	173
30.1 Table of Contents	173
30.2 Introduction to the KernelType policy	173
30.3 The KernelTraits trait class	175
30.4 List of kernels and classes that use a \c KernelType	175
31 The MetricType policy in mlpack	177
32 The TreeType policy in mlpack	179
32.1 Introduction	179
32.2 What is a tree?	180
32.3 Template parameters required by the TreeType policy	181
32.4 The TreeType API	182
32.5 Rigorous API documentation	184
32.5.1 Template parameters	185
32.5.2 Constructors and destructors	186
32.5.3 Basic tree functionality	187
32.5.4 Complex tree functionality and bounds	188
32.5.5 Serialization	190
32.6 The TreeTraits trait class	190
32.7 A list of trees in mlpack and more information	191
33 Bug List	193
34 Namespace Index	197
34.1 Namespace List	197

35 Hierarchical Index	199
35.1 Class Hierarchy	199
36 Class Index	213
36.1 Class List	213
37 File Index	237
37.1 File List	237
38 Namespace Documentation	251
38.1 boost Namespace Reference	251
38.1.1 Detailed Description	251
38.2 boost::serialization Namespace Reference	251
38.3 ens Namespace Reference	252
38.4 mlpack Namespace Reference	252
38.4.1 Detailed Description	254
38.4.2 Function Documentation	254
38.4.2.1 CheckMatrices() [1/3]	255
38.4.2.2 CheckMatrices() [2/3]	255
38.4.2.3 CheckMatrices() [3/3]	255
38.4.2.4 SerializeObject()	255
38.4.2.5 SerializeObjectAll()	256
38.4.2.6 SerializePointerObject()	256
38.4.2.7 SerializePointerObjectAll()	256
38.4.2.8 TestAllArmadilloSerialization() [1/2]	256
38.4.2.9 TestAllArmadilloSerialization() [2/2]	257
38.4.2.10 TestArmadilloSerialization() [1/2]	257
38.4.2.11 TestArmadilloSerialization() [2/2]	257
38.5 mlpack::adaboost Namespace Reference	257
38.6 mlpack::amf Namespace Reference	257

38.6.1	Detailed Description	259
38.6.2	Typedef Documentation	259
38.6.2.1	NMFALSFactorizer	259
38.6.2.2	SVDBatchFactorizer	260
38.6.2.3	SVDCompleteIncrementalFactorizer	260
38.6.2.4	SVDIncompleteIncrementalFactorizer	260
38.6.3	Function Documentation	261
38.6.3.1	SVDBatchLearning::HUpdate< arma::sp_mat >()	261
38.6.3.2	SVDBatchLearning::WUpdate< arma::sp_mat >()	261
38.6.3.3	SVDIncompleteIncrementalLearning::HUpdate< arma::sp_mat >()	261
38.6.3.4	SVDIncompleteIncrementalLearning::WUpdate< arma::sp_mat >()	261
38.7	mlpack::ann Namespace Reference	262
38.7.1	Detailed Description	268
38.7.2	Typedef Documentation	268
38.7.2.1	CustomLayer	269
38.7.2.2	Embedding	269
38.7.2.3	GlorotInitialization	269
38.7.2.4	HardSigmoidLayer	269
38.7.2.5	IdentityLayer	269
38.7.2.6	LayerTypes	270
38.7.2.7	ReLULayer	270
38.7.2.8	Residual	270
38.7.2.9	SELU	271
38.7.2.10	SigmoidLayer	271
38.7.2.11	SoftPlusLayer	271
38.7.2.12	TanHLayer	271
38.7.2.13	XavierInitialization	271
38.7.3	Function Documentation	272

38.7.3.1	HAS_ANY_METHOD_FORM()	272
38.7.3.2	HAS_MEM_FUNC() [1/13]	272
38.7.3.3	HAS_MEM_FUNC() [2/13]	272
38.7.3.4	HAS_MEM_FUNC() [3/13]	272
38.7.3.5	HAS_MEM_FUNC() [4/13]	272
38.7.3.6	HAS_MEM_FUNC() [5/13]	273
38.7.3.7	HAS_MEM_FUNC() [6/13]	273
38.7.3.8	HAS_MEM_FUNC() [7/13]	273
38.7.3.9	HAS_MEM_FUNC() [8/13]	273
38.7.3.10	HAS_MEM_FUNC() [9/13]	273
38.7.3.11	HAS_MEM_FUNC() [10/13]	273
38.7.3.12	HAS_MEM_FUNC() [11/13]	274
38.7.3.13	HAS_MEM_FUNC() [12/13]	274
38.7.3.14	HAS_MEM_FUNC() [13/13]	274
38.8	mlpack::ann::augmented Namespace Reference	274
38.9	mlpack::ann::augmented::scorers Namespace Reference	274
38.9.1	Function Documentation	274
38.9.1.1	SequencePrecision()	275
38.10	mlpack::ann::augmented::tasks Namespace Reference	276
38.11	mlpack::bindings Namespace Reference	276
38.12	mlpack::bindings::cli Namespace Reference	276
38.12.1	Function Documentation	284
38.12.1.1	AddToPO() [1/4]	284
38.12.1.2	AddToPO() [2/4]	284
38.12.1.3	AddToPO() [3/4]	285
38.12.1.4	AddToPO() [4/4]	285
38.12.1.5	DefaultParam()	286
38.12.1.6	DefaultParamImpl() [1/5]	286

38.12.1.7 DefaultParamImpl() [2/5]	286
38.12.1.8 DefaultParamImpl() [3/5]	287
38.12.1.9 DefaultParamImpl() [4/5]	287
38.12.1.10 DefaultParamImpl() [5/5]	287
38.12.1.11 DeleteAllocatedMemory()	287
38.12.1.12 DeleteAllocatedMemoryImpl() [1/3]	288
38.12.1.13 DeleteAllocatedMemoryImpl() [2/3]	288
38.12.1.14 DeleteAllocatedMemoryImpl() [3/3]	288
38.12.1.15 EndProgram()	288
38.12.1.16 GetAllocatedMemory() [1/4]	289
38.12.1.17 GetAllocatedMemory() [2/4]	289
38.12.1.18 GetAllocatedMemory() [3/4]	289
38.12.1.19 GetAllocatedMemory() [4/4]	289
38.12.1.20 GetBindingName()	290
38.12.1.21 GetParam() [1/5]	290
38.12.1.22 GetParam() [2/5]	290
38.12.1.23 GetParam() [3/5]	291
38.12.1.24 GetParam() [4/5]	291
38.12.1.25 GetParam() [5/5]	291
38.12.1.26 GetPrintableParam() [1/5]	292
38.12.1.27 GetPrintableParam() [2/5]	292
38.12.1.28 GetPrintableParam() [3/5]	292
38.12.1.29 GetPrintableParam() [4/5]	293
38.12.1.30 GetPrintableParam() [5/5]	293
38.12.1.31 GetPrintableParamName() [1/5]	293
38.12.1.32 GetPrintableParamName() [2/5]	293
38.12.1.33 GetPrintableParamName() [3/5]	294
38.12.1.34 GetPrintableParamName() [4/5]	294

38.12.1.35GetPrintableParamName()	[5/5]	294
38.12.1.36GetPrintableParamValue()	[1/5]	294
38.12.1.37GetPrintableParamValue()	[2/5]	295
38.12.1.38GetPrintableParamValue()	[3/5]	295
38.12.1.39GetPrintableParamValue()	[4/5]	295
38.12.1.40GetPrintableParamValue()	[5/5]	295
38.12.1.41GetPrintableType()	[1/6]	296
38.12.1.42GetPrintableType()	[2/6]	296
38.12.1.43GetPrintableType()	[3/6]	296
38.12.1.44GetPrintableType()	[4/6]	296
38.12.1.45GetPrintableType()	[5/6]	297
38.12.1.46GetPrintableType()	[6/6]	297
38.12.1.47GetRawParam()	[1/4]	297
38.12.1.48GetRawParam()	[2/4]	297
38.12.1.49GetRawParam()	[3/4]	298
38.12.1.50GetRawParam()	[4/4]	298
38.12.1.51IgnoreCheck()		298
38.12.1.52MapParameterName()	[1/3]	299
38.12.1.53MapParameterName()	[2/3]	299
38.12.1.54MapParameterName()	[3/3]	299
38.12.1.55OutputParam()		300
38.12.1.56OutputParamImpl()	[1/5]	300
38.12.1.57OutputParamImpl()	[2/5]	300
38.12.1.58OutputParamImpl()	[3/5]	301
38.12.1.59OutputParamImpl()	[4/5]	301
38.12.1.60OutputParamImpl()	[5/5]	301
38.12.1.61PARAM_FLAG()	[1/3]	301
38.12.1.62PARAM_FLAG()	[2/3]	301

38.12.1.63PARAM_FLAG() [3/3]	302
38.12.1.64PARAM_STRING_IN()	302
38.12.1.65ParamString()	302
38.12.1.66ParseCommandLine()	302
38.12.1.67PrintDataset()	303
38.12.1.68PrintDefault()	303
38.12.1.69PrintHelp()	303
38.12.1.70PrintImport()	303
38.12.1.71PrintModel()	304
38.12.1.72PrintOutputOptionInfo()	304
38.12.1.73PrintType()	304
38.12.1.74PrintTypeDoc() [1/6]	304
38.12.1.75PrintTypeDoc() [2/6]	304
38.12.1.76PrintTypeDoc() [3/6]	305
38.12.1.77PrintTypeDoc() [4/6]	305
38.12.1.78PrintTypeDoc() [5/6]	305
38.12.1.79PrintTypeDoc() [6/6]	305
38.12.1.80PrintTypeDocs()	306
38.12.1.81PrintValue()	306
38.12.1.82ProcessOptions() [1/2]	306
38.12.1.83ProcessOptions() [2/2]	306
38.12.1.84ProgramCall() [1/2]	306
38.12.1.85ProgramCall() [2/2]	307
38.12.1.86SetParam() [1/5]	307
38.12.1.87SetParam() [2/5]	307
38.12.1.88SetParam() [3/5]	308
38.12.1.89SetParam() [4/5]	308
38.12.1.90SetParam() [5/5]	308

38.12.1.91StringTypeParam()	309
38.12.1.92StringTypeParam< bool >()	309
38.12.1.93StringTypeParam< double >()	309
38.12.1.94StringTypeParam< int >()	310
38.12.1.95StringTypeParam< std::string >()	310
38.12.1.96StringTypeParam< std::tuple< mlpack::data::DatasetInfo, arma::mat > >()	310
38.12.1.97StringTypeParamImpl() [1/3]	310
38.12.1.98StringTypeParamImpl() [2/3]	311
38.12.1.99StringTypeParamImpl() [3/3]	311
38.13mlpack::bindings::markdown Namespace Reference	311
38.13.1 Function Documentation	314
38.13.1.1 DefaultParam()	314
38.13.1.2 GetBindingName() [1/2]	314
38.13.1.3 GetBindingName() [2/2]	315
38.13.1.4 GetParam()	315
38.13.1.5 GetPrintableParam() [1/6]	315
38.13.1.6 GetPrintableParam() [2/6]	316
38.13.1.7 GetPrintableParam() [3/6]	316
38.13.1.8 GetPrintableParam() [4/6]	316
38.13.1.9 GetPrintableParam() [5/6]	317
38.13.1.10GetPrintableParam() [6/6]	317
38.13.1.11GetPrintableParamName() [1/5]	317
38.13.1.12GetPrintableParamName() [2/5]	318
38.13.1.13GetPrintableParamName() [3/5]	318
38.13.1.14GetPrintableParamName() [4/5]	318
38.13.1.15GetPrintableParamName() [5/5]	318
38.13.1.16GetPrintableParamValue() [1/5]	319
38.13.1.17GetPrintableParamValue() [2/5]	319

38.13.1.18	GetPrintableParamValue()	[3/5]	319
38.13.1.19	GetPrintableParamValue()	[4/5]	319
38.13.1.20	GetPrintableParamValue()	[5/5]	320
38.13.1.21	GetPrintableType()	[1/2]	320
38.13.1.22	GetPrintableType()	[2/2]	320
38.13.1.23	IgnoreCheck()		321
38.13.1.24	IsSerializable()	[1/4]	321
38.13.1.25	IsSerializable()	[2/4]	321
38.13.1.26	IsSerializable()	[3/4]	321
38.13.1.27	IsSerializable()	[4/4]	322
38.13.1.28	ParamString()		322
38.13.1.29	ParamType()		322
38.13.1.30	PrintDataset()		322
38.13.1.31	PrintDefault()		322
38.13.1.32	PrintImport()		323
38.13.1.33	PrintLanguage()		323
38.13.1.34	PrintModel()		323
38.13.1.35	PrintOutputOptionInfo()		323
38.13.1.36	PrintTypeDoc()		323
38.13.1.37	PrintTypeDocs()		324
38.13.1.38	PrintValue()		324
38.13.1.39	ProgramCall()	[1/2]	324
38.13.1.40	ProgramCall()	[2/2]	324
38.14	mplpack::bindings::python Namespace Reference		324
38.14.1	Function Documentation		331
38.14.1.1	DefaultParam()		331
38.14.1.2	DefaultParamImpl()	[1/5]	332
38.14.1.3	DefaultParamImpl()	[2/5]	332

38.14.1.4 DefaultParamImpl() [3/5]	332
38.14.1.5 DefaultParamImpl() [4/5]	332
38.14.1.6 DefaultParamImpl() [5/5]	333
38.14.1.7 GetArmaType()	333
38.14.1.8 GetBindingName()	333
38.14.1.9 GetCythonType() [1/4]	333
38.14.1.10 GetCythonType() [2/4]	333
38.14.1.11 GetCythonType() [3/4]	334
38.14.1.12 GetCythonType() [4/4]	334
38.14.1.13 GetCythonType< bool >()	334
38.14.1.14 GetCythonType< double >()	334
38.14.1.15 GetCythonType< int >()	335
38.14.1.16 GetCythonType< size_t >()	335
38.14.1.17 GetCythonType< std::string >()	335
38.14.1.18 GetNumpyType()	335
38.14.1.19 GetNumpyType< double >()	336
38.14.1.20 GetNumpyType< size_t >()	336
38.14.1.21 GetNumpyTypeChar()	336
38.14.1.22 GetNumpyTypeChar< arma::Col< size_t > >()	336
38.14.1.23 GetNumpyTypeChar< arma::mat >()	336
38.14.1.24 GetNumpyTypeChar< arma::Mat< size_t > >()	336
38.14.1.25 GetNumpyTypeChar< arma::Row< size_t > >()	337
38.14.1.26 GetNumpyTypeChar< arma::rowvec >()	337
38.14.1.27 GetNumpyTypeChar< arma::vec >()	337
38.14.1.28 GetParam()	337
38.14.1.29 GetPrintableParam() [1/6]	338
38.14.1.30 GetPrintableParam() [2/6]	338
38.14.1.31 GetPrintableParam() [3/6]	338

38.14.1.32GetPrintableParam() [4/6]	339
38.14.1.33GetPrintableParam() [5/6]	339
38.14.1.34GetPrintableParam() [6/6]	339
38.14.1.35GetPrintableType() [1/6]	340
38.14.1.36GetPrintableType() [2/6]	340
38.14.1.37GetPrintableType() [3/6]	340
38.14.1.38GetPrintableType() [4/6]	340
38.14.1.39GetPrintableType() [5/6]	341
38.14.1.40GetPrintableType() [6/6]	341
38.14.1.41GetPrintableType< bool >()	341
38.14.1.42GetPrintableType< double >()	341
38.14.1.43GetPrintableType< int >()	342
38.14.1.44GetPrintableType< size_t >()	342
38.14.1.45GetPrintableType< std::string >()	342
38.14.1.46IgnoreCheck() [1/3]	342
38.14.1.47IgnoreCheck() [2/3]	343
38.14.1.48IgnoreCheck() [3/3]	343
38.14.1.49ImportDecl() [1/4]	343
38.14.1.50ImportDecl() [2/4]	343
38.14.1.51ImportDecl() [3/4]	344
38.14.1.52ImportDecl() [4/4]	344
38.14.1.53ParamString()	344
38.14.1.54PrintClassDefn() [1/4]	344
38.14.1.55PrintClassDefn() [2/4]	345
38.14.1.56PrintClassDefn() [3/4]	345
38.14.1.57PrintClassDefn() [4/4]	345
38.14.1.58PrintDataset()	346
38.14.1.59PrintDefault()	346

38.14.1.60PrintDefn()	346
38.14.1.61PrintDoc()	346
38.14.1.62PrintImport()	347
38.14.1.63PrintInputOptions() [1/2]	347
38.14.1.64PrintInputOptions() [2/2]	347
38.14.1.65PrintInputProcessing() [1/6]	348
38.14.1.66PrintInputProcessing() [2/6]	348
38.14.1.67PrintInputProcessing() [3/6]	349
38.14.1.68PrintInputProcessing() [4/6]	349
38.14.1.69PrintInputProcessing() [5/6]	350
38.14.1.70PrintInputProcessing() [6/6]	350
38.14.1.71PrintModel()	351
38.14.1.72PrintOutputOptionInfo()	351
38.14.1.73PrintOutputOptions() [1/2]	351
38.14.1.74PrintOutputOptions() [2/2]	351
38.14.1.75PrintOutputProcessing() [1/5]	351
38.14.1.76PrintOutputProcessing() [2/5]	352
38.14.1.77PrintOutputProcessing() [3/5]	352
38.14.1.78PrintOutputProcessing() [4/5]	353
38.14.1.79PrintOutputProcessing() [5/5]	353
38.14.1.80PrintPYX()	354
38.14.1.81PrintTypeDoc() [1/6]	354
38.14.1.82PrintTypeDoc() [2/6]	354
38.14.1.83PrintTypeDoc() [3/6]	355
38.14.1.84PrintTypeDoc() [4/6]	355
38.14.1.85PrintTypeDoc() [5/6]	355
38.14.1.86PrintTypeDoc() [6/6]	355
38.14.1.87PrintValue() [1/2]	356

38.14.1.88	PrintValue() [2/2]	356
38.14.1.89	ProgramCall() [1/2]	356
38.14.1.90	ProgramCall() [2/2]	356
38.14.1.91	SerializeIn()	356
38.14.1.92	SerializeOut()	357
38.14.1.93	StripType()	357
38.14.2	Variable Documentation	357
38.14.2.1	programName	357
38.15	mlpack::bindings::tests Namespace Reference	357
38.15.1	Function Documentation	359
38.15.1.1	CleanMemory()	359
38.15.1.2	DeleteAllocatedMemory()	359
38.15.1.3	DeleteAllocatedMemoryImpl() [1/3]	359
38.15.1.4	DeleteAllocatedMemoryImpl() [2/3]	360
38.15.1.5	DeleteAllocatedMemoryImpl() [3/3]	360
38.15.1.6	GetAllocatedMemory() [1/4]	360
38.15.1.7	GetAllocatedMemory() [2/4]	360
38.15.1.8	GetAllocatedMemory() [3/4]	361
38.15.1.9	GetAllocatedMemory() [4/4]	361
38.15.1.10	GetParam() [1/2]	361
38.15.1.11	GetParam() [2/2]	361
38.15.1.12	GetPrintableParam() [1/6]	362
38.15.1.13	GetPrintableParam() [2/6]	362
38.15.1.14	GetPrintableParam() [3/6]	363
38.15.1.15	GetPrintableParam() [4/6]	363
38.15.1.16	GetPrintableParam() [5/6]	363
38.15.1.17	GetPrintableParam() [6/6]	364
38.15.1.18	IgnoreCheck()	364

38.15.2 Variable Documentation	364
38.15.2.1 programName	364
38.16mlpack::bound Namespace Reference	364
38.17mlpack::bound::addr Namespace Reference	365
38.17.1 Function Documentation	365
38.17.1.1 AddressToPoint()	366
38.17.1.2 CompareAddresses()	367
38.17.1.3 Contains()	367
38.17.1.4 PointToAddress()	367
38.18mlpack::bound::meta Namespace Reference	368
38.18.1 Detailed Description	368
38.19mlpack::cf Namespace Reference	368
38.19.1 Detailed Description	370
38.19.2 Typedef Documentation	370
38.19.2.1 ArmaSVDFactorizer	370
38.19.2.2 EuclideanSearch	370
38.20mlpack::cv Namespace Reference	371
38.20.1 Enumeration Type Documentation	372
38.20.1.1 AverageStrategy	372
38.20.2 Function Documentation	372
38.20.2.1 AssertSizes()	372
38.21mlpack::data Namespace Reference	373
38.21.1 Detailed Description	375
38.21.2 Typedef Documentation	376
38.21.2.1 DatasetInfo	376
38.21.3 Enumeration Type Documentation	376
38.21.3.1 Datatype	376
38.21.3.2 format	376

38.21.4 Function Documentation	377
38.21.4.1 Binarize() [1/2]	377
38.21.4.2 Binarize() [2/2]	377
38.21.4.3 ConfusionMatrix()	378
38.21.4.4 Extension()	379
38.21.4.5 HAS_EXACT_METHOD_FORM()	379
38.21.4.6 IsNaNInf()	379
38.21.4.7 Load() [1/5]	380
38.21.4.8 Load() [2/5]	381
38.21.4.9 Load() [3/5]	381
38.21.4.10 Load() [4/5]	382
38.21.4.11 Load() [5/5]	383
38.21.4.12 LoadARFF() [1/2]	384
38.21.4.13 LoadARFF() [2/2]	384
38.21.4.14 NormalizeLabels()	385
38.21.4.15 OneHotEncoding()	385
38.21.4.16 RevertLabels()	385
38.21.4.17 Save() [1/2]	386
38.21.4.18 Save() [2/2]	387
38.21.4.19 Split() [1/4]	387
38.21.4.20 Split() [2/4]	388
38.21.4.21 Split() [3/4]	389
38.21.4.22 Split() [4/4]	389
38.22 mpack::dbscan Namespace Reference	390
38.23 mpack::decision_stump Namespace Reference	390
38.24 mpack::det Namespace Reference	390
38.24.1 Detailed Description	391
38.24.2 Function Documentation	391

38.24.2.1 PrintLeafMembership()	391
38.24.2.2 PrintVariableImportance()	392
38.24.2.3 Trainer()	392
38.25mlpack::distribution Namespace Reference	392
38.25.1 Detailed Description	393
38.26mlpack::emst Namespace Reference	393
38.26.1 Detailed Description	393
38.27mlpack::fastmks Namespace Reference	394
38.27.1 Detailed Description	394
38.28mlpack::gmm Namespace Reference	394
38.28.1 Detailed Description	395
38.29mlpack::hmm Namespace Reference	395
38.29.1 Detailed Description	396
38.29.2 Enumeration Type Documentation	396
38.29.2.1 HMMType [1/2]	396
38.29.2.2 HMMType [2/2]	396
38.29.3 Function Documentation	397
38.29.3.1 LoadHMMAndPerformAction()	397
38.29.3.2 SaveHMM()	397
38.30mlpack::hpt Namespace Reference	397
38.30.1 Typedef Documentation	398
38.30.1.1 TupleOfHyperParameters	398
38.30.2 Function Documentation	398
38.30.2.1 Fixed()	399
38.31mlpack::kde Namespace Reference	399
38.31.1 Detailed Description	400
38.31.2 Typedef Documentation	400
38.31.2.1 KDEType	400

38.31.3 Enumeration Type Documentation	400
38.31.3.1 KDEMode	400
38.32mlpack::kernel Namespace Reference	401
38.32.1 Detailed Description	402
38.33mlpack::kmeans Namespace Reference	402
38.33.1 Detailed Description	403
38.33.2 Typedef Documentation	404
38.33.2.1 CoverTreeDualTreeKMeans	404
38.33.2.2 DefaultDualTreeKMeans	404
38.33.3 Function Documentation	404
38.33.3.1 HideChild() [1/2]	404
38.33.3.2 HideChild() [2/2]	405
38.33.3.3 RestoreChildren() [1/2]	405
38.33.3.4 RestoreChildren() [2/2]	405
38.34mlpack::kpca Namespace Reference	405
38.35mlpack::lcc Namespace Reference	406
38.36mlpack::lmnn Namespace Reference	406
38.36.1 Detailed Description	406
38.37mlpack::math Namespace Reference	406
38.37.1 Detailed Description	409
38.37.2 Typedef Documentation	409
38.37.2.1 Range	410
38.37.3 Function Documentation	410
38.37.3.1 AccuLog()	410
38.37.3.2 Center()	410
38.37.3.3 ClampNonNegative()	411
38.37.3.4 ClampNonPositive()	411
38.37.3.5 ClampRange()	412

38.37.3.6 ClearAlias() [1/2]	412
38.37.3.7 ClearAlias() [2/2]	412
38.37.3.8 ColumnCovariance() [1/2]	413
38.37.3.9 ColumnCovariance() [2/2]	413
38.37.3.10 FixedRandomSeed()	413
38.37.3.11 LogAdd()	413
38.37.3.12 MakeAlias() [1/7]	414
38.37.3.13 MakeAlias() [2/7]	414
38.37.3.14 MakeAlias() [3/7]	414
38.37.3.15 MakeAlias() [4/7]	415
38.37.3.16 MakeAlias() [5/7]	415
38.37.3.17 MakeAlias() [6/7]	415
38.37.3.18 MakeAlias() [7/7]	415
38.37.3.19 ObtainDistinctSamples()	416
38.37.3.20 Orthogonalize() [1/2]	416
38.37.3.21 Orthogonalize() [2/2]	416
38.37.3.22 RandBernoulli()	417
38.37.3.23 RandInt() [1/2]	417
38.37.3.24 RandInt() [2/2]	417
38.37.3.25 RandNormal() [1/2]	418
38.37.3.26 RandNormal() [2/2]	418
38.37.3.27 Random() [1/2]	418
38.37.3.28 Random() [2/2]	419
38.37.3.29 RandomBasis()	419
38.37.3.30 RandomSeed()	419
38.37.3.31 RandVector()	420
38.37.3.32 RemoveRows()	420
38.37.3.33 ShuffleData() [1/5]	420

38.37.3.34	ShuffleData() [2 / 5]	421
38.37.3.35	ShuffleData() [3 / 5]	421
38.37.3.36	ShuffleData() [4 / 5]	422
38.37.3.37	ShuffleData() [5 / 5]	422
38.37.3.38	Sign()	422
38.37.3.39	Smat()	423
38.37.3.40	Svec() [1 / 2]	423
38.37.3.41	Svec() [2 / 2]	424
38.37.3.42	SvecIndex()	424
38.37.3.43	SymKronId()	424
38.37.3.44	VectorPower()	424
38.37.3.45	WhitenUsingEig()	425
38.37.3.46	WhitenUsingSVD()	425
38.37.4	Variable Documentation	425
38.37.4.1	randGen	425
38.37.4.2	randNormalDist	425
38.37.4.3	randUniformDist	426
38.38	mlpack::matrix_completion Namespace Reference	426
38.39	mlpack::meanshift Namespace Reference	426
38.39.1	Detailed Description	426
38.40	mlpack::metric Namespace Reference	426
38.40.1	Typedef Documentation	427
38.40.1.1	ChebyshevDistance	427
38.40.1.2	EuclideanDistance	427
38.40.1.3	ManhattanDistance	427
38.40.1.4	SquaredEuclideanDistance	428
38.41	mlpack::naive_bayes Namespace Reference	428
38.41.1	Detailed Description	428

38.42mlpack::nca Namespace Reference	428
38.42.1 Detailed Description	428
38.43mlpack::neighbor Namespace Reference	429
38.43.1 Typedef Documentation	431
38.43.1.1 DefeatistKNN	431
38.43.1.2 FurthestNeighborSort	432
38.43.1.3 KFN	432
38.43.1.4 KNN	432
38.43.1.5 KRAFN	432
38.43.1.6 KRANN	433
38.43.1.7 NearestNeighborSort	433
38.43.1.8 NSType	433
38.43.1.9 RAType	433
38.43.1.10SpillKNN	434
38.43.2 Enumeration Type Documentation	434
38.43.2.1 NeighborSearchMode	434
38.43.3 Function Documentation	434
38.43.3.1 Unmap() [1/2]	434
38.43.3.2 Unmap() [2/2]	435
38.44mlpack::nn Namespace Reference	435
38.44.1 Function Documentation	436
38.44.1.1 MaximalInputs()	436
38.44.1.2 NormalizeColByMax()	437
38.45mlpack::pca Namespace Reference	437
38.46mlpack::perceptron Namespace Reference	438
38.47mlpack::radical Namespace Reference	438
38.47.1 Function Documentation	438
38.47.1.1 WhitenFeatureMajorMatrix()	438

38.48mlpack::range Namespace Reference	439
38.48.1 Detailed Description	439
38.48.2 Typedef Documentation	439
38.48.2.1 RSType	440
38.49mlpack::regression Namespace Reference	440
38.49.1 Detailed Description	440
38.50mlpack::rl Namespace Reference	440
38.50.1 Typedef Documentation	441
38.50.1.1 Acrobat	442
38.50.1.2 NStepQLearning	442
38.50.1.3 OneStepQLearning	442
38.50.1.4 OneStepSarsa	443
38.51mlpack::sfinae Namespace Reference	443
38.52mlpack::sparse_coding Namespace Reference	443
38.53mlpack::svd Namespace Reference	444
38.54mlpack::svm Namespace Reference	444
38.55mlpack::tree Namespace Reference	444
38.55.1 Detailed Description	450
38.55.2 Typedef Documentation	450
38.55.2.1 AxisOrthogonalHyperplane	451
38.55.2.2 BallTree	451
38.55.2.3 BinaryDoubleNumericSplit	451
38.55.2.4 CosineNodeQueue	452
38.55.2.5 DecisionStump	452
38.55.2.6 DiscreteHilbertRTreeAuxiliaryInformation	452
38.55.2.7 HilbertRTree	453
38.55.2.8 HoeffdingDoubleNumericSplit	453
38.55.2.9 HoeffdingTreeType	453

38.55.2.10Hyperplane	453
38.55.2.11KDTree	454
38.55.2.12MaxRPTree	454
38.55.2.13MeanSplitBallTree	455
38.55.2.14MeanSplitKDTree	456
38.55.2.15MeanSPTree	456
38.55.2.16NonOrtMeanSPTree	457
38.55.2.17NonOrtSPTree	457
38.55.2.18RPlusPlusTree	458
38.55.2.19RPlusTree	458
38.55.2.20RPTree	459
38.55.2.21RStarTree	459
38.55.2.22RTree	460
38.55.2.23SPTree	460
38.55.2.24StandardCoverTree	461
38.55.2.25UBTree	461
38.55.2.26VPTree	462
38.55.2.27VPTreeSplit	462
38.55.2.28XTree	463
38.55.3 Function Documentation	463
38.55.3.1 Bootstrap()	463
38.55.3.2 EnumerateTree()	463
38.55.4 Variable Documentation	464
38.55.4.1 MAX_OVERLAP	464
38.56mlpack::tree::enumerate Namespace Reference	464
38.56.1 Function Documentation	464
38.56.1.1 EnumerateTreeImpl()	464
38.57mlpack::tree::split Namespace Reference	465

38.57.1 Function Documentation	465
38.57.1.1 PerformSplit() [1/2]	465
38.57.1.2 PerformSplit() [2/2]	465
38.58 mpack::util Namespace Reference	466
38.58.1 Function Documentation	468
38.58.1.1 DisableBacktrace()	468
38.58.1.2 DisableVerbose()	468
38.58.1.3 EnableTimers()	468
38.58.1.4 EnableVerbose()	469
38.58.1.5 GetParamPtr()	469
38.58.1.6 GetParamWithInfo()	469
38.58.1.7 GetVersion()	469
38.58.1.8 HyphenateString()	469
38.58.1.9 ReportIgnoredParam() [1/2]	470
38.58.1.10 ReportIgnoredParam() [2/2]	470
38.58.1.11 RequireAtLeastOnePassed()	471
38.58.1.12 RequireNoneOrAllPassed()	471
38.58.1.13 RequireOnlyOnePassed()	472
38.58.1.14 RequireParamInSet()	473
38.58.1.15 RequireParamValue()	473
38.58.1.16 ResetTimers()	474
38.58.1.17 SetInputParam()	474
38.58.1.18 SetParam()	475
38.58.1.19 SetParamPtr()	475
38.58.1.20 SetParamWithInfo()	476
38.59 std Namespace Reference	476
38.59.1 Typedef Documentation	476
38.59.1.1 enable_if_t	476

39 Class Documentation	477
39.1 version< mlpack::adaboost::AdaBoost< WeakLearnerType, MatType > > Struct Template Reference	477
39.1.1 Detailed Description	477
39.1.2 Member Function Documentation	477
39.1.2.1 BOOST_STATIC_CONSTANT()	477
39.2 version< mlpack::ann::BRNN< OutputLayerType, MergeLayerType, MergeOutputType, InitializationRuleType, CustomLayer... > > Struct Template Reference	478
39.2.1 Detailed Description	478
39.2.2 Member Function Documentation	478
39.2.2.1 BOOST_STATIC_CONSTANT()	478
39.3 version< mlpack::ann::FFN< OutputLayerType, InitializationRuleType, CustomLayer... > > Struct Template Reference	478
39.3.1 Detailed Description	478
39.3.2 Member Function Documentation	479
39.3.2.1 BOOST_STATIC_CONSTANT()	479
39.4 version< mlpack::ann::RNN< OutputLayerType, InitializationRuleType, CustomLayer... > > Struct Template Reference	479
39.4.1 Detailed Description	479
39.4.2 Member Function Documentation	479
39.4.2.1 BOOST_STATIC_CONSTANT()	479
39.5 InitHMMModel Struct Reference	480
39.5.1 Detailed Description	480
39.5.2 Member Function Documentation	480
39.5.2.1 Apply()	480
39.5.2.2 Create() [1/4]	481
39.5.2.3 Create() [2/4]	481
39.5.2.4 Create() [3/4]	481
39.5.2.5 Create() [4/4]	481
39.5.2.6 RandomInitialize() [1/4]	482

39.5.2.7 RandomInitialize() [2/4]	482
39.5.2.8 RandomInitialize() [3/4]	482
39.5.2.9 RandomInitialize() [4/4]	482
39.6 IsVector< VecType > Struct Template Reference	482
39.6.1 Detailed Description	483
39.6.2 Member Data Documentation	483
39.6.2.1 value	483
39.7 IsVector< arma::Col< eT > > Struct Template Reference	483
39.7.1 Detailed Description	484
39.7.2 Member Data Documentation	484
39.7.2.1 value	484
39.8 IsVector< arma::Row< eT > > Struct Template Reference	484
39.8.1 Detailed Description	484
39.8.2 Member Data Documentation	484
39.8.2.1 value	485
39.9 IsVector< arma::SpCol< eT > > Struct Template Reference	485
39.9.1 Detailed Description	485
39.9.2 Member Data Documentation	485
39.9.2.1 value	485
39.10 IsVector< arma::SpRow< eT > > Struct Template Reference	485
39.10.1 Detailed Description	486
39.10.2 Member Data Documentation	486
39.10.2.1 value	486
39.11 IsVector< arma::SpSubview< eT > > Struct Template Reference	486
39.11.1 Detailed Description	486
39.11.2 Member Data Documentation	486
39.11.2.1 value	487
39.12 IsVector< arma::subview_col< eT > > Struct Template Reference	487

39.12.1 Detailed Description	487
39.12.2 Member Data Documentation	487
39.12.2.1 value	487
39.13IsVector< arma::subview_row< eT > > Struct Template Reference	487
39.13.1 Detailed Description	488
39.13.2 Member Data Documentation	488
39.13.2.1 value	488
39.14AdaBoost< WeakLearnerType, MatType > Class Template Reference	488
39.14.1 Detailed Description	489
39.14.2 Constructor & Destructor Documentation	490
39.14.2.1 AdaBoost() [1/2]	490
39.14.2.2 AdaBoost() [2/2]	491
39.14.3 Member Function Documentation	491
39.14.3.1 Alpha() [1/2]	491
39.14.3.2 Alpha() [2/2]	491
39.14.3.3 Classify()	491
39.14.3.4 NumClasses()	492
39.14.3.5 serialize()	492
39.14.3.6 Tolerance() [1/2]	492
39.14.3.7 Tolerance() [2/2]	492
39.14.3.8 Train()	493
39.14.3.9 WeakLearner() [1/2]	493
39.14.3.10WeakLearner() [2/2]	493
39.14.3.11WeakLearners()	494
39.15AdaBoostModel Class Reference	494
39.15.1 Detailed Description	495
39.15.2 Member Enumeration Documentation	495
39.15.2.1 WeakLearnerTypes	495

39.15.3 Constructor & Destructor Documentation	495
39.15.3.1 AdaBoostModel() [1/4]	495
39.15.3.2 AdaBoostModel() [2/4]	496
39.15.3.3 AdaBoostModel() [3/4]	496
39.15.3.4 AdaBoostModel() [4/4]	496
39.15.3.5 ~AdaBoostModel()	496
39.15.4 Member Function Documentation	496
39.15.4.1 Classify()	496
39.15.4.2 Dimensionality() [1/2]	497
39.15.4.3 Dimensionality() [2/2]	497
39.15.4.4 Mappings() [1/2]	497
39.15.4.5 Mappings() [2/2]	497
39.15.4.6 operator=()	497
39.15.4.7 serialize()	498
39.15.4.8 Train()	498
39.15.4.9 WeakLearnerType() [1/2]	498
39.15.4.10WeakLearnerType() [2/2]	498
39.16AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType > Class Template Reference	499
39.16.1 Detailed Description	499
39.16.2 Constructor & Destructor Documentation	500
39.16.2.1 AMF()	500
39.16.3 Member Function Documentation	500
39.16.3.1 Apply()	501
39.16.3.2 InitializeRule() [1/2]	501
39.16.3.3 InitializeRule() [2/2]	501
39.16.3.4 TerminationPolicy() [1/2]	501
39.16.3.5 TerminationPolicy() [2/2]	502
39.16.3.6 Update() [1/2]	502

39.16.3.7 Update() [2/2]	502
39.17AverageInitialization Class Reference	502
39.17.1 Detailed Description	503
39.17.2 Constructor & Destructor Documentation	503
39.17.2.1 AverageInitialization()	503
39.17.3 Member Function Documentation	503
39.17.3.1 Initialize()	503
39.17.3.2 serialize()	504
39.18CompleteIncrementalTermination< TerminationPolicy > Class Template Reference	504
39.18.1 Detailed Description	505
39.18.2 Constructor & Destructor Documentation	505
39.18.2.1 CompleteIncrementalTermination()	505
39.18.3 Member Function Documentation	505
39.18.3.1 Index()	505
39.18.3.2 Initialize() [1/2]	506
39.18.3.3 Initialize() [2/2]	506
39.18.3.4 IsConverged()	506
39.18.3.5 Iteration()	507
39.18.3.6 MaxIterations() [1/2]	507
39.18.3.7 MaxIterations() [2/2]	507
39.18.3.8 TPolicy() [1/2]	507
39.18.3.9 TPolicy() [2/2]	508
39.19GivenInitialization Class Reference	508
39.19.1 Detailed Description	508
39.19.2 Constructor & Destructor Documentation	508
39.19.2.1 GivenInitialization() [1/3]	509
39.19.2.2 GivenInitialization() [2/3]	509
39.19.2.3 GivenInitialization() [3/3]	509

39.19.3 Member Function Documentation	509
39.19.3.1 Initialize()	509
39.19.3.2 serialize()	510
39.20IncompleteIncrementalTermination< TerminationPolicy > Class Template Reference	510
39.20.1 Detailed Description	511
39.20.2 Constructor & Destructor Documentation	511
39.20.2.1 IncompleteIncrementalTermination()	511
39.20.3 Member Function Documentation	511
39.20.3.1 Index()	511
39.20.3.2 Initialize()	512
39.20.3.3 IsConverged()	512
39.20.3.4 Iteration()	512
39.20.3.5 MaxIterations() [1/2]	512
39.20.3.6 MaxIterations() [2/2]	513
39.20.3.7 TPolicy() [1/2]	513
39.20.3.8 TPolicy() [2/2]	513
39.21MaxIterationTermination Class Reference	513
39.21.1 Detailed Description	514
39.21.2 Constructor & Destructor Documentation	514
39.21.2.1 MaxIterationTermination()	514
39.21.3 Member Function Documentation	515
39.21.3.1 Index()	515
39.21.3.2 Initialize()	515
39.21.3.3 IsConverged()	515
39.21.3.4 Iteration() [1/2]	515
39.21.3.5 Iteration() [2/2]	516
39.21.3.6 MaxIterations() [1/2]	516
39.21.3.7 MaxIterations() [2/2]	516

39.22NMFALSUpdate Class Reference	516
39.22.1 Detailed Description	517
39.22.2 Constructor & Destructor Documentation	517
39.22.2.1 NMFALSUpdate()	517
39.22.3 Member Function Documentation	517
39.22.3.1 HUpdate()	518
39.22.3.2 Initialize()	518
39.22.3.3 serialize()	518
39.22.3.4 WUpdate()	519
39.23NFMultiplicativeDistanceUpdate Class Reference	519
39.23.1 Detailed Description	520
39.23.2 Constructor & Destructor Documentation	520
39.23.2.1 NFMultiplicativeDistanceUpdate()	520
39.23.3 Member Function Documentation	520
39.23.3.1 HUpdate()	521
39.23.3.2 Initialize()	522
39.23.3.3 serialize()	522
39.23.3.4 WUpdate()	522
39.24NFMultiplicativeDivergenceUpdate Class Reference	523
39.24.1 Detailed Description	524
39.24.2 Constructor & Destructor Documentation	524
39.24.2.1 NFMultiplicativeDivergenceUpdate()	524
39.24.3 Member Function Documentation	524
39.24.3.1 HUpdate()	524
39.24.3.2 Initialize()	525
39.24.3.3 serialize()	525
39.24.3.4 WUpdate()	525
39.25RandomAcolInitialization< columnsToAverage > Class Template Reference	526

39.25.1 Detailed Description	526
39.25.2 Constructor & Destructor Documentation	527
39.25.2.1 RandomAcollInitialization()	527
39.25.3 Member Function Documentation	527
39.25.3.1 Initialize()	527
39.25.3.2 serialize()	527
39.26 RandomInitialization Class Reference	528
39.26.1 Detailed Description	528
39.26.2 Constructor & Destructor Documentation	528
39.26.2.1 RandomInitialization()	528
39.26.3 Member Function Documentation	528
39.26.3.1 Initialize()	528
39.26.3.2 serialize()	529
39.27 SimpleResidueTermination Class Reference	529
39.27.1 Detailed Description	530
39.27.2 Constructor & Destructor Documentation	530
39.27.2.1 SimpleResidueTermination()	530
39.27.3 Member Function Documentation	531
39.27.3.1 Index()	531
39.27.3.2 Initialize()	531
39.27.3.3 IsConverged()	531
39.27.3.4 Iteration()	532
39.27.3.5 MaxIterations() ^[1/2]	532
39.27.3.6 MaxIterations() ^[2/2]	532
39.27.3.7 MinResidue() ^[1/2]	533
39.27.3.8 MinResidue() ^[2/2]	533
39.27.4 Member Data Documentation	533
39.27.4.1 iteration	533

39.27.4.2 maxIterations	533
39.27.4.3 minResidue	534
39.27.4.4 nm	534
39.27.4.5 normOld	534
39.27.4.6 residue	534
39.28SimpleToleranceTermination< MatType > Class Template Reference	535
39.28.1 Detailed Description	535
39.28.2 Constructor & Destructor Documentation	535
39.28.2.1 SimpleToleranceTermination()	536
39.28.3 Member Function Documentation	536
39.28.3.1 Index()	536
39.28.3.2 Initialize()	536
39.28.3.3 IsConverged()	536
39.28.3.4 Iteration()	538
39.28.3.5 MaxIterations() [1/2]	538
39.28.3.6 MaxIterations() [2/2]	538
39.28.3.7 Tolerance() [1/2]	538
39.28.3.8 Tolerance() [2/2]	539
39.29SVDBatchLearning Class Reference	539
39.29.1 Detailed Description	539
39.29.2 Constructor & Destructor Documentation	540
39.29.2.1 SVDBatchLearning()	540
39.29.3 Member Function Documentation	540
39.29.3.1 HUpdate()	540
39.29.3.2 Initialize()	541
39.29.3.3 serialize()	541
39.29.3.4 WUpdate()	541
39.30SVDCompleteIncrementalLearning< MatType > Class Template Reference	542

39.30.1 Detailed Description	542
39.30.2 Constructor & Destructor Documentation	543
39.30.2.1 SVDCompleteIncrementalLearning()	543
39.30.3 Member Function Documentation	543
39.30.3.1 HUpdate()	543
39.30.3.2 Initialize()	544
39.30.3.3 WUpdate()	544
39.31 SVDCompleteIncrementalLearning< arma::sp_mat > Class Template Reference	544
39.31.1 Detailed Description	545
39.31.2 Constructor & Destructor Documentation	545
39.31.2.1 SVDCompleteIncrementalLearning()	545
39.31.2.2 ~SVDCompleteIncrementalLearning()	545
39.31.3 Member Function Documentation	545
39.31.3.1 HUpdate()	546
39.31.3.2 Initialize()	546
39.31.3.3 WUpdate()	546
39.32 SVDIncompleteIncrementalLearning Class Reference	547
39.32.1 Detailed Description	547
39.32.2 Constructor & Destructor Documentation	548
39.32.2.1 SVDIncompleteIncrementalLearning()	548
39.32.3 Member Function Documentation	548
39.32.3.1 HUpdate()	548
39.32.3.2 Initialize()	549
39.32.3.3 WUpdate()	549
39.33 ValidationRMSETermination< MatType > Class Template Reference	549
39.33.1 Detailed Description	550
39.33.2 Constructor & Destructor Documentation	551
39.33.2.1 ValidationRMSETermination()	551

39.33.3 Member Function Documentation	551
39.33.3.1 Index()	551
39.33.3.2 Initialize()	551
39.33.3.3 IsConverged()	552
39.33.3.4 Iteration()	552
39.33.3.5 MaxIterations() [1/2]	552
39.33.3.6 MaxIterations() [2/2]	553
39.33.3.7 NumTestPoints()	553
39.33.3.8 Tolerance() [1/2]	553
39.33.3.9 Tolerance() [2/2]	553
39.34Add< InputDataType, OutputDataType > Class Template Reference	553
39.34.1 Detailed Description	554
39.34.2 Constructor & Destructor Documentation	555
39.34.2.1 Add()	555
39.34.3 Member Function Documentation	555
39.34.3.1 Backward()	555
39.34.3.2 Delta() [1/2]	555
39.34.3.3 Delta() [2/2]	556
39.34.3.4 Forward()	556
39.34.3.5 Gradient() [1/3]	556
39.34.3.6 Gradient() [2/3]	557
39.34.3.7 Gradient() [3/3]	557
39.34.3.8 OutputParameter() [1/2]	557
39.34.3.9 OutputParameter() [2/2]	557
39.34.3.10Parameters() [1/2]	558
39.34.3.11Parameters() [2/2]	558
39.34.3.12Serialize()	558
39.35AddMerge< InputDataType, OutputDataType, CustomLayers > Class Template Reference	558

39.35.1 Detailed Description	560
39.35.2 Constructor & Destructor Documentation	560
39.35.2.1 AddMerge()	560
39.35.2.2 ~AddMerge()	560
39.35.3 Member Function Documentation	561
39.35.3.1 Add() [1/2]	561
39.35.3.2 Add() [2/2]	561
39.35.3.3 Backward() [1/2]	561
39.35.3.4 Backward() [2/2]	561
39.35.3.5 Delta() [1/2]	562
39.35.3.6 Delta() [2/2]	562
39.35.3.7 Forward()	562
39.35.3.8 Gradient() [1/2]	563
39.35.3.9 Gradient() [2/2]	563
39.35.3.10 InputParameter() [1/2]	563
39.35.3.11 InputParameter() [2/2]	563
39.35.3.12 Model()	564
39.35.3.13 OutputParameter() [1/2]	564
39.35.3.14 OutputParameter() [2/2]	564
39.35.3.15 Parameters() [1/2]	564
39.35.3.16 Parameters() [2/2]	564
39.35.3.17 Run() [1/2]	565
39.35.3.18 Run() [2/2]	565
39.35.3.19 Serialize()	565
39.36 AddVisitor< CustomLayers > Class Template Reference	565
39.36.1 Detailed Description	566
39.36.2 Constructor & Destructor Documentation	566
39.36.2.1 AddVisitor()	566

39.36.3 Member Function Documentation	566
39.36.3.1 operator>()	566
39.37 AlphaDropout< InputDataType, OutputDataType > Class Template Reference	567
39.37.1 Detailed Description	568
39.37.2 Constructor & Destructor Documentation	568
39.37.2.1 AlphaDropout()	568
39.37.3 Member Function Documentation	569
39.37.3.1 A()	569
39.37.3.2 AlphaDash()	569
39.37.3.3 B()	569
39.37.3.4 Backward()	569
39.37.3.5 Delta() [1/2]	570
39.37.3.6 Delta() [2/2]	570
39.37.3.7 Deterministic() [1/2]	570
39.37.3.8 Deterministic() [2/2]	570
39.37.3.9 Forward()	570
39.37.3.10 Mask()	571
39.37.3.11 OutputParameter() [1/2]	571
39.37.3.12 OutputParameter() [2/2]	571
39.37.3.13 Ratio() [1/2]	571
39.37.3.14 Ratio() [2/2]	572
39.37.3.15 serialize()	572
39.38 AtrousConvolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, InputDataType, OutputDataType > Class Template Reference	572
39.38.1 Detailed Description	574
39.38.2 Constructor & Destructor Documentation	574
39.38.2.1 AtrousConvolution() [1/2]	574
39.38.2.2 AtrousConvolution() [2/2]	575

39.38.3 Member Function Documentation	575
39.38.3.1 Backward()	575
39.38.3.2 Delta() [1/2]	576
39.38.3.3 Delta() [2/2]	576
39.38.3.4 Forward()	576
39.38.3.5 Gradient() [1/3]	577
39.38.3.6 Gradient() [2/3]	577
39.38.3.7 Gradient() [3/3]	577
39.38.3.8 InputHeight() [1/2]	577
39.38.3.9 InputHeight() [2/2]	577
39.38.3.10 InputWidth() [1/2]	578
39.38.3.11 InputWidth() [2/2]	578
39.38.3.12 OutputHeight() [1/2]	578
39.38.3.13 OutputHeight() [2/2]	578
39.38.3.14 OutputParameter() [1/2]	579
39.38.3.15 OutputParameter() [2/2]	579
39.38.3.16 OutputWidth() [1/2]	579
39.38.3.17 OutputWidth() [2/2]	579
39.38.3.18 Parameters() [1/2]	579
39.38.3.19 Parameters() [2/2]	580
39.38.3.20 Reset()	580
39.38.3.21 Serialize()	580
39.39 AddTask Class Reference	580
39.39.1 Detailed Description	581
39.39.2 Constructor & Destructor Documentation	581
39.39.2.1 AddTask()	581
39.39.3 Member Function Documentation	581
39.39.3.1 Generate() [1/2]	581

39.39.3.2 Generate() [2/2]	582
39.40 CopyTask Class Reference	582
39.40.1 Detailed Description	583
39.40.2 Constructor & Destructor Documentation	583
39.40.2.1 CopyTask()	583
39.40.3 Member Function Documentation	584
39.40.3.1 Generate() [1/2]	584
39.40.3.2 Generate() [2/2]	584
39.41 SortTask Class Reference	584
39.41.1 Detailed Description	585
39.41.2 Constructor & Destructor Documentation	585
39.41.2.1 SortTask()	585
39.41.3 Member Function Documentation	586
39.41.3.1 Generate() [1/2]	586
39.41.3.2 Generate() [2/2]	586
39.42 BackwardVisitor Class Reference	587
39.42.1 Detailed Description	587
39.42.2 Constructor & Destructor Documentation	587
39.42.2.1 BackwardVisitor() [1/2]	588
39.42.2.2 BackwardVisitor() [2/2]	588
39.42.3 Member Function Documentation	588
39.42.3.1 operator>()	588
39.43 BaseLayer< ActivationFunction, InputDataType, OutputDataType > Class Template Reference	588
39.43.1 Detailed Description	589
39.43.2 Constructor & Destructor Documentation	590
39.43.2.1 BaseLayer()	590
39.43.3 Member Function Documentation	590
39.43.3.1 Backward()	590

39.43.3.2 Delta() [1/2]	590
39.43.3.3 Delta() [2/2]	591
39.43.3.4 Forward()	591
39.43.3.5 OutputParameter() [1/2]	591
39.43.3.6 OutputParameter() [2/2]	592
39.43.3.7 serialize()	592
39.44BatchNorm< InputDataType, OutputDataType > Class Template Reference	592
39.44.1 Detailed Description	593
39.44.2 Constructor & Destructor Documentation	594
39.44.2.1 BatchNorm() [1/2]	594
39.44.2.2 BatchNorm() [2/2]	594
39.44.3 Member Function Documentation	594
39.44.3.1 Backward()	594
39.44.3.2 Delta() [1/2]	596
39.44.3.3 Delta() [2/2]	596
39.44.3.4 Deterministic() [1/2]	596
39.44.3.5 Deterministic() [2/2]	596
39.44.3.6 Forward()	596
39.44.3.7 Gradient() [1/3]	597
39.44.3.8 Gradient() [2/3]	597
39.44.3.9 Gradient() [3/3]	597
39.44.3.10OutputParameter() [1/2]	598
39.44.3.11OutputParameter() [2/2]	598
39.44.3.12Parameters() [1/2]	598
39.44.3.13Parameters() [2/2]	598
39.44.3.14Reset()	598
39.44.3.15serialize()	599
39.44.3.16TrainingMean()	599

39.44.3.17	TrainingVariance()	599
39.45	BernoulliDistribution< DataType > Class Template Reference	599
39.45.1	Detailed Description	600
39.45.2	Constructor & Destructor Documentation	600
39.45.2.1	BernoulliDistribution() [1/2]	601
39.45.2.2	BernoulliDistribution() [2/2]	601
39.45.3	Member Function Documentation	601
39.45.3.1	Logits() [1/2]	601
39.45.3.2	Logits() [2/2]	602
39.45.3.3	LogProbability()	602
39.45.3.4	LogProbBackward()	602
39.45.3.5	Probability() [1/3]	602
39.45.3.6	Probability() [2/3]	604
39.45.3.7	Probability() [3/3]	604
39.45.3.8	Sample()	604
39.45.3.9	serialize()	605
39.46	BilinearInterpolation< InputDataType, OutputDataType > Class Template Reference	605
39.46.1	Detailed Description	606
39.46.2	Constructor & Destructor Documentation	606
39.46.2.1	BilinearInterpolation() [1/2]	606
39.46.2.2	BilinearInterpolation() [2/2]	606
39.46.3	Member Function Documentation	607
39.46.3.1	Backward()	607
39.46.3.2	Delta() [1/2]	607
39.46.3.3	Delta() [2/2]	607
39.46.3.4	Forward()	608
39.46.3.5	OutputParameter() [1/2]	609
39.46.3.6	OutputParameter() [2/2]	609

39.46.3.7 serialize()	609
39.47 BinaryRBM Class Reference	609
39.47.1 Detailed Description	610
39.48 BRNN< OutputLayerType, MergeLayerType, MergeOutputType, InitializationRuleType, CustomLayers > Class Template Reference	610
39.48.1 Detailed Description	611
39.48.2 Member Typedef Documentation	612
39.48.2.1 NetworkType	612
39.48.3 Constructor & Destructor Documentation	612
39.48.3.1 BRNN()	612
39.48.4 Member Function Documentation	613
39.48.4.1 Add() [1/2]	613
39.48.4.2 Add() [2/2]	613
39.48.4.3 Evaluate() [1/2]	613
39.48.4.4 Evaluate() [2/2]	613
39.48.4.5 EvaluateWithGradient()	614
39.48.4.6 Gradient()	614
39.48.4.7 NumFunctions()	616
39.48.4.8 Parameters() [1/2]	616
39.48.4.9 Parameters() [2/2]	616
39.48.4.10 Predict()	617
39.48.4.11 Predictors() [1/2]	617
39.48.4.12 Predictors() [2/2]	617
39.48.4.13 Reset()	618
39.48.4.14 ResetParameters()	618
39.48.4.15 Responses() [1/2]	618
39.48.4.16 Responses() [2/2]	618
39.48.4.17 Rho() [1/2]	619

39.48.4.18	Rho() [2/2]	619
39.48.4.19	Serialize()	619
39.48.4.20	Shuffle()	619
39.48.4.21	Train() [1/2]	620
39.48.4.22	Train() [2/2]	620
39.49	Concat< InputDataType, OutputDataType, CustomLayers > Class Template Reference	621
39.49.1	Detailed Description	623
39.49.2	Constructor & Destructor Documentation	623
39.49.2.1	Concat() [1/2]	623
39.49.2.2	Concat() [2/2]	623
39.49.2.3	~Concat()	625
39.49.3	Member Function Documentation	625
39.49.3.1	Add() [1/3]	625
39.49.3.2	Add() [2/3]	625
39.49.3.3	Add() [3/3]	625
39.49.3.4	Backward() [1/2]	626
39.49.3.5	Backward() [2/2]	626
39.49.3.6	Delta() [1/2]	626
39.49.3.7	Delta() [2/2]	627
39.49.3.8	Forward()	627
39.49.3.9	Gradient() [1/4]	627
39.49.3.10	Gradient() [2/4]	627
39.49.3.11	Gradient() [3/4]	628
39.49.3.12	Gradient() [4/4]	628
39.49.3.13	InputParameter() [1/2]	628
39.49.3.14	InputParameter() [2/2]	628
39.49.3.15	Model()	628
39.49.3.16	OutputParameter() [1/2]	629

39.49.3.17	OutputParameter() [2/2]	629
39.49.3.18	Parameters() [1/2]	629
39.49.3.19	Parameters() [2/2]	629
39.49.3.20	Run() [1/2]	629
39.49.3.21	Run() [2/2]	630
39.49.3.22	Serialize()	630
39.50	Concatenate< InputDataType, OutputDataType > Class Template Reference	630
39.50.1	Detailed Description	631
39.50.2	Constructor & Destructor Documentation	631
39.50.2.1	Concatenate()	631
39.50.3	Member Function Documentation	631
39.50.3.1	Backward()	632
39.50.3.2	Concat() [1/2]	632
39.50.3.3	Concat() [2/2]	632
39.50.3.4	Delta() [1/2]	632
39.50.3.5	Delta() [2/2]	633
39.50.3.6	Forward()	633
39.50.3.7	OutputParameter() [1/2]	633
39.50.3.8	OutputParameter() [2/2]	633
39.50.3.9	Parameters() [1/2]	634
39.50.3.10	Parameters() [2/2]	634
39.50.3.11	Serialize()	634
39.51	ConcatPerformance< OutputLayerType, InputDataType, OutputDataType > Class Template Reference	634
39.51.1	Detailed Description	635
39.51.2	Constructor & Destructor Documentation	635
39.51.2.1	ConcatPerformance()	636
39.51.3	Member Function Documentation	636
39.51.3.1	Backward()	636

39.51.3.2 Delta() [1/2]	636
39.51.3.3 Delta() [2/2]	637
39.51.3.4 Forward()	637
39.51.3.5 OutputParameter() [1/2]	637
39.51.3.6 OutputParameter() [2/2]	637
39.51.3.7 serialize()	637
39.52Constant< InputDataType, OutputDataType > Class Template Reference	638
39.52.1 Detailed Description	638
39.52.2 Constructor & Destructor Documentation	638
39.52.2.1 Constant()	639
39.52.3 Member Function Documentation	639
39.52.3.1 Backward()	639
39.52.3.2 Delta() [1/2]	639
39.52.3.3 Delta() [2/2]	640
39.52.3.4 Forward()	640
39.52.3.5 OutputParameter() [1/2]	640
39.52.3.6 OutputParameter() [2/2]	640
39.52.3.7 serialize()	641
39.53ConstInitialization Class Reference	641
39.53.1 Detailed Description	641
39.53.2 Constructor & Destructor Documentation	641
39.53.2.1 ConstInitialization()	642
39.53.3 Member Function Documentation	642
39.53.3.1 Initialize() [1/2]	642
39.53.3.2 Initialize() [2/2]	642
39.53.3.3 InitValue()	643
39.53.3.4 initValue()	643

39.54 Convolution < ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, Input↔ DataType, OutputDataType > Class Template Reference	643
39.54.1 Detailed Description	645
39.54.2 Constructor & Destructor Documentation	645
39.54.2.1 Convolution() [1/2]	645
39.54.2.2 Convolution() [2/2]	645
39.54.3 Member Function Documentation	646
39.54.3.1 Backward()	646
39.54.3.2 Delta() [1/2]	646
39.54.3.3 Delta() [2/2]	647
39.54.3.4 Forward()	647
39.54.3.5 Gradient() [1/3]	647
39.54.3.6 Gradient() [2/3]	647
39.54.3.7 Gradient() [3/3]	648
39.54.3.8 InputHeight() [1/2]	648
39.54.3.9 InputHeight() [2/2]	648
39.54.3.10 InputParameter() [1/2]	648
39.54.3.11 InputParameter() [2/2]	648
39.54.3.12 InputWidth() [1/2]	649
39.54.3.13 InputWidth() [2/2]	649
39.54.3.14 OutputHeight() [1/2]	649
39.54.3.15 OutputHeight() [2/2]	649
39.54.3.16 OutputParameter() [1/2]	650
39.54.3.17 OutputParameter() [2/2]	650
39.54.3.18 OutputWidth() [1/2]	650
39.54.3.19 OutputWidth() [2/2]	650
39.54.3.20 Parameters() [1/2]	650
39.54.3.21 Parameters() [2/2]	651

39.54.3.22	Reset()	651
39.54.3.23	Serialize()	651
39.55	CopyVisitor< CustomLayers > Class Template Reference	651
39.55.1	Detailed Description	652
39.55.2	Member Function Documentation	652
39.55.2.1	operator>()	652
39.56	CReLU< InputDataType, OutputDataType > Class Template Reference	652
39.56.1	Detailed Description	653
39.56.2	Constructor & Destructor Documentation	654
39.56.2.1	CReLU()	654
39.56.3	Member Function Documentation	654
39.56.3.1	Backward()	654
39.56.3.2	Delta() [1/2]	654
39.56.3.3	Delta() [2/2]	655
39.56.3.4	Forward()	655
39.56.3.5	OutputParameter() [1/2]	655
39.56.3.6	OutputParameter() [2/2]	655
39.56.3.7	serialize()	656
39.57	CrossEntropyError< InputDataType, OutputDataType > Class Template Reference	656
39.57.1	Detailed Description	656
39.57.2	Constructor & Destructor Documentation	657
39.57.2.1	CrossEntropyError()	657
39.57.3	Member Function Documentation	657
39.57.3.1	Backward()	657
39.57.3.2	Eps() [1/2]	658
39.57.3.3	Eps() [2/2]	658
39.57.3.4	Forward()	658
39.57.3.5	OutputParameter() [1/2]	658

39.57.3.6 OutputParameter() [2/2]	659
39.57.3.7 serialize()	659
39.58DeleteVisitor Class Reference	659
39.58.1 Detailed Description	660
39.58.2 Member Function Documentation	660
39.58.2.1 operator>()	660
39.59DeltaVisitor Class Reference	660
39.59.1 Detailed Description	661
39.59.2 Member Function Documentation	661
39.59.2.1 operator>()	661
39.60DeterministicSetVisitor Class Reference	661
39.60.1 Detailed Description	662
39.60.2 Constructor & Destructor Documentation	662
39.60.2.1 DeterministicSetVisitor()	662
39.60.3 Member Function Documentation	662
39.60.3.1 operator>()	662
39.61DiceLoss< InputDataType, OutputDataType > Class Template Reference	662
39.61.1 Detailed Description	663
39.61.2 Constructor & Destructor Documentation	663
39.61.2.1 DiceLoss()	664
39.61.3 Member Function Documentation	664
39.61.3.1 Backward()	664
39.61.3.2 Forward()	664
39.61.3.3 OutputParameter() [1/2]	665
39.61.3.4 OutputParameter() [2/2]	665
39.61.3.5 serialize()	665
39.61.3.6 Smooth() [1/2]	665
39.61.3.7 Smooth() [2/2]	666

39.62DropConnect< InputDataType, OutputDataType > Class Template Reference	666
39.62.1 Detailed Description	667
39.62.2 Constructor & Destructor Documentation	668
39.62.2.1 DropConnect() [1/2]	668
39.62.2.2 DropConnect() [2/2]	668
39.62.2.3 ~DropConnect()	668
39.62.3 Member Function Documentation	668
39.62.3.1 Backward()	669
39.62.3.2 Delta() [1/2]	669
39.62.3.3 Delta() [2/2]	669
39.62.3.4 Deterministic() [1/2]	669
39.62.3.5 Deterministic() [2/2]	670
39.62.3.6 Forward()	670
39.62.3.7 Gradient() [1/3]	670
39.62.3.8 Gradient() [2/3]	670
39.62.3.9 Gradient() [3/3]	671
39.62.3.10Model()	671
39.62.3.11OutputParameter() [1/2]	671
39.62.3.12OutputParameter() [2/2]	671
39.62.3.13Parameters() [1/2]	672
39.62.3.14Parameters() [2/2]	672
39.62.3.15Ratio() [1/2]	672
39.62.3.16Ratio() [2/2]	672
39.62.3.17Serialize()	673
39.63Dropout< InputDataType, OutputDataType > Class Template Reference	673
39.63.1 Detailed Description	674
39.63.2 Constructor & Destructor Documentation	674
39.63.2.1 Dropout()	674

39.63.3 Member Function Documentation	675
39.63.3.1 Backward()	675
39.63.3.2 Delta() [1/2]	675
39.63.3.3 Delta() [2/2]	675
39.63.3.4 Deterministic() [1/2]	675
39.63.3.5 Deterministic() [2/2]	676
39.63.3.6 Forward()	676
39.63.3.7 OutputParameter() [1/2]	676
39.63.3.8 OutputParameter() [2/2]	676
39.63.3.9 Ratio() [1/2]	677
39.63.3.10 Ratio() [2/2]	677
39.63.3.11 serialize()	677
39.64 EarthMoverDistance< InputDataType, OutputDataType > Class Template Reference	677
39.64.1 Detailed Description	678
39.64.2 Constructor & Destructor Documentation	678
39.64.2.1 EarthMoverDistance()	678
39.64.3 Member Function Documentation	679
39.64.3.1 Backward()	679
39.64.3.2 Forward()	679
39.64.3.3 OutputParameter() [1/2]	679
39.64.3.4 OutputParameter() [2/2]	680
39.64.3.5 serialize()	680
39.65 ELU< InputDataType, OutputDataType > Class Template Reference	680
39.65.1 Detailed Description	681
39.65.2 Constructor & Destructor Documentation	682
39.65.2.1 ELU() [1/2]	682
39.65.2.2 ELU() [2/2]	682
39.65.3 Member Function Documentation	683

39.65.3.1 Alpha() [1/2]	683
39.65.3.2 Alpha() [2/2]	683
39.65.3.3 Backward()	683
39.65.3.4 Delta() [1/2]	684
39.65.3.5 Delta() [2/2]	684
39.65.3.6 Forward()	684
39.65.3.7 Lambda()	684
39.65.3.8 OutputParameter() [1/2]	685
39.65.3.9 OutputParameter() [2/2]	685
39.65.3.10Serialize()	685
39.66FastLSTM< InputDataType, OutputDataType > Class Template Reference	685
39.66.1 Detailed Description	686
39.66.2 Member Typedef Documentation	687
39.66.2.1 ElemType	687
39.66.2.2 InputElemType	688
39.66.3 Constructor & Destructor Documentation	688
39.66.3.1 FastLSTM() [1/2]	688
39.66.3.2 FastLSTM() [2/2]	688
39.66.4 Member Function Documentation	688
39.66.4.1 Backward()	688
39.66.4.2 Delta() [1/2]	689
39.66.4.3 Delta() [2/2]	689
39.66.4.4 Forward()	689
39.66.4.5 Gradient() [1/3]	690
39.66.4.6 Gradient() [2/3]	690
39.66.4.7 Gradient() [3/3]	690
39.66.4.8 OutputParameter() [1/2]	690
39.66.4.9 OutputParameter() [2/2]	691

39.66.4.10Parameters() [1/2]	691
39.66.4.11Parameters() [2/2]	691
39.66.4.12Reset()	691
39.66.4.13ResetCell()	691
39.66.4.14Rho() [1/2]	692
39.66.4.15Rho() [2/2]	692
39.66.4.16Serialize()	692
39.67FFN< OutputLayerType, InitializationRuleType, CustomLayers > Class Template Reference	692
39.67.1 Detailed Description	694
39.67.2 Member Typedef Documentation	694
39.67.2.1 NetworkType	695
39.67.3 Constructor & Destructor Documentation	695
39.67.3.1 FFN() [1/3]	695
39.67.3.2 FFN() [2/3]	695
39.67.3.3 FFN() [3/3]	695
39.67.3.4 ~FFN()	696
39.67.4 Member Function Documentation	696
39.67.4.1 Add() [1/2]	696
39.67.4.2 Add() [2/2]	696
39.67.4.3 Backward()	696
39.67.4.4 Evaluate() [1/4]	697
39.67.4.5 Evaluate() [2/4]	697
39.67.4.6 Evaluate() [3/4]	697
39.67.4.7 Evaluate() [4/4]	698
39.67.4.8 EvaluateWithGradient() [1/2]	698
39.67.4.9 EvaluateWithGradient() [2/2]	699
39.67.4.10Forward() [1/2]	699
39.67.4.11Forward() [2/2]	700

39.67.4.12	Gradient()	700
39.67.4.13	NumFunctions()	701
39.67.4.14	operator=()	701
39.67.4.15	Parameters() [1/2]	701
39.67.4.16	Parameters() [2/2]	701
39.67.4.17	Predict()	701
39.67.4.18	Predictors() [1/2]	702
39.67.4.19	Predictors() [2/2]	702
39.67.4.20	ResetParameters()	702
39.67.4.21	Responses() [1/2]	703
39.67.4.22	Responses() [2/2]	703
39.67.4.23	Serialize()	703
39.67.4.24	Shuffle()	703
39.67.4.25	Train() [1/2]	703
39.67.4.26	Train() [2/2]	704
39.68	FFTConvolution< BorderMode, padLastDim > Class Template Reference	705
39.68.1	Detailed Description	705
39.68.2	Member Function Documentation	706
39.68.2.1	Convolution() [1/5]	706
39.68.2.2	Convolution() [2/5]	706
39.68.2.3	Convolution() [3/5]	706
39.68.2.4	Convolution() [4/5]	707
39.68.2.5	Convolution() [5/5]	707
39.69	FlexibleReLU< InputDataType, OutputDataType > Class Template Reference	707
39.69.1	Detailed Description	708
39.69.2	Constructor & Destructor Documentation	709
39.69.2.1	FlexibleReLU()	709
39.69.3	Member Function Documentation	709

39.69.3.1 Alpha() [1/2]	709
39.69.3.2 Alpha() [2/2]	710
39.69.3.3 Backward()	710
39.69.3.4 Delta() [1/2]	710
39.69.3.5 Delta() [2/2]	710
39.69.3.6 Forward()	711
39.69.3.7 Gradient() [1/3]	711
39.69.3.8 Gradient() [2/3]	711
39.69.3.9 Gradient() [3/3]	712
39.69.3.10OutputParameter() [1/2]	712
39.69.3.11OutputParameter() [2/2]	712
39.69.3.12Parameters() [1/2]	712
39.69.3.13Parameters() [2/2]	712
39.69.3.14Reset()	713
39.69.3.15serialize()	713
39.70ForwardVisitor Class Reference	713
39.70.1 Detailed Description	714
39.70.2 Constructor & Destructor Documentation	714
39.70.2.1 ForwardVisitor()	714
39.70.3 Member Function Documentation	714
39.70.3.1 operator>()()	714
39.71FullConvolution Class Reference	714
39.71.1 Detailed Description	714
39.72GaussianInitialization Class Reference	715
39.72.1 Detailed Description	715
39.72.2 Constructor & Destructor Documentation	715
39.72.2.1 GaussianInitialization()	715
39.72.3 Member Function Documentation	715

39.72.3.1 Initialize() [1/2]	716
39.72.3.2 Initialize() [2/2]	716
39.73 Glimpse< InputDataType, OutputDataType > Class Template Reference	717
39.73.1 Detailed Description	718
39.73.2 Constructor & Destructor Documentation	718
39.73.2.1 Glimpse()	718
39.73.3 Member Function Documentation	718
39.73.3.1 Backward()	719
39.73.3.2 Delta() [1/2]	719
39.73.3.3 Delta() [2/2]	719
39.73.3.4 Deterministic() [1/2]	719
39.73.3.5 Deterministic() [2/2]	720
39.73.3.6 Forward()	720
39.73.3.7 InputHeight() [1/2]	720
39.73.3.8 InputHeight() [2/2]	720
39.73.3.9 InputWidth() [1/2]	721
39.73.3.10 InputWidth() [2/2]	721
39.73.3.11 Location()	721
39.73.3.12 OutputHeight() [1/2]	721
39.73.3.13 OutputHeight() [2/2]	721
39.73.3.14 OutputParameter() [1/2]	722
39.73.3.15 OutputParameter() [2/2]	722
39.73.3.16 OutputWidth() [1/2]	722
39.73.3.17 OutputWidth() [2/2]	722
39.73.3.18 Serialize()	722
39.74 GlorotInitializationType< Uniform > Class Template Reference	723
39.74.1 Detailed Description	723
39.74.2 Constructor & Destructor Documentation	724

39.74.2.1 GlorotInitializationType()	724
39.74.3 Member Function Documentation	724
39.74.3.1 Initialize() [1/4]	724
39.74.3.2 Initialize() [2/4]	724
39.74.3.3 Initialize() [3/4]	725
39.74.3.4 Initialize() [4/4]	725
39.75 GradientSetVisitor Class Reference	725
39.75.1 Detailed Description	726
39.75.2 Constructor & Destructor Documentation	726
39.75.2.1 GradientSetVisitor()	726
39.75.3 Member Function Documentation	726
39.75.3.1 operator>()	726
39.76 GradientUpdateVisitor Class Reference	727
39.76.1 Detailed Description	727
39.76.2 Constructor & Destructor Documentation	727
39.76.2.1 GradientUpdateVisitor()	727
39.76.3 Member Function Documentation	728
39.76.3.1 operator>()	728
39.77 GradientVisitor Class Reference	728
39.77.1 Detailed Description	728
39.77.2 Constructor & Destructor Documentation	729
39.77.2.1 GradientVisitor() [1/2]	729
39.77.2.2 GradientVisitor() [2/2]	729
39.77.3 Member Function Documentation	729
39.77.3.1 operator>()	729
39.78 GradientZeroVisitor Class Reference	730
39.78.1 Detailed Description	730
39.78.2 Constructor & Destructor Documentation	730

39.78.2.1 GradientZeroVisitor()	730
39.78.3 Member Function Documentation	730
39.78.3.1 operator()	731
39.79GRU< InputDataType, OutputDataType > Class Template Reference	731
39.79.1 Detailed Description	732
39.79.2 Constructor & Destructor Documentation	732
39.79.2.1 GRU() [1/2]	732
39.79.2.2 GRU() [2/2]	733
39.79.2.3 ~GRU()	733
39.79.3 Member Function Documentation	733
39.79.3.1 Backward()	733
39.79.3.2 Delta() [1/2]	734
39.79.3.3 Delta() [2/2]	734
39.79.3.4 Deterministic() [1/2]	734
39.79.3.5 Deterministic() [2/2]	734
39.79.3.6 Forward()	734
39.79.3.7 Gradient() [1/3]	735
39.79.3.8 Gradient() [2/3]	735
39.79.3.9 Gradient() [3/3]	735
39.79.3.10Model()	735
39.79.3.11OutputParameter() [1/2]	736
39.79.3.12OutputParameter() [2/2]	736
39.79.3.13Parameters() [1/2]	736
39.79.3.14Parameters() [2/2]	736
39.79.3.15ResetCell()	736
39.79.3.16Rho() [1/2]	737
39.79.3.17Rho() [2/2]	737
39.79.3.18Serialize()	737

39.80HardSigmoidFunction Class Reference	737
39.80.1 Detailed Description	738
39.80.2 Member Function Documentation	738
39.80.2.1 Deriv() [1/2]	738
39.80.2.2 Deriv() [2/2]	738
39.80.2.3 Fn() [1/2]	739
39.80.2.4 Fn() [2/2]	739
39.81HardTanH< InputDataType, OutputDataType > Class Template Reference	740
39.81.1 Detailed Description	741
39.81.2 Constructor & Destructor Documentation	741
39.81.2.1 HardTanH()	741
39.81.3 Member Function Documentation	742
39.81.3.1 Backward()	742
39.81.3.2 Delta() [1/2]	742
39.81.3.3 Delta() [2/2]	742
39.81.3.4 Forward()	742
39.81.3.5 MaxValue() [1/2]	743
39.81.3.6 MaxValue() [2/2]	743
39.81.3.7 MinValue() [1/2]	743
39.81.3.8 MinValue() [2/2]	743
39.81.3.9 OutputParameter() [1/2]	744
39.81.3.10OutputParameter() [2/2]	744
39.81.3.11serialize()	744
39.82HelInitialization Class Reference	744
39.82.1 Detailed Description	745
39.82.2 Constructor & Destructor Documentation	745
39.82.2.1 HelInitialization()	745
39.82.3 Member Function Documentation	745

39.82.3.1 Initialize() [1/2]	745
39.82.3.2 Initialize() [2/2]	746
39.83IdentityFunction Class Reference	746
39.83.1 Detailed Description	747
39.83.2 Member Function Documentation	747
39.83.2.1 Deriv() [1/3]	747
39.83.2.2 Deriv() [2/3]	748
39.83.2.3 Deriv() [3/3]	748
39.83.2.4 Fn() [1/2]	748
39.83.2.5 Fn() [2/2]	750
39.84InitTraits< InitRuleType > Class Template Reference	750
39.84.1 Detailed Description	751
39.84.2 Member Data Documentation	751
39.84.2.1 UseLayer	751
39.85InitTraits< KathirvalavakumarSubavathiInitialization > Class Template Reference	751
39.85.1 Detailed Description	751
39.85.2 Member Data Documentation	752
39.85.2.1 UseLayer	752
39.86InitTraits< NguyenWidrowInitialization > Class Template Reference	752
39.86.1 Detailed Description	752
39.86.2 Member Data Documentation	752
39.86.2.1 UseLayer	753
39.87Join< InputDataType, OutputDataType > Class Template Reference	753
39.87.1 Detailed Description	753
39.87.2 Constructor & Destructor Documentation	754
39.87.2.1 Join()	754
39.87.3 Member Function Documentation	754
39.87.3.1 Backward()	754

39.87.3.2 Delta() [1/2]	755
39.87.3.3 Delta() [2/2]	755
39.87.3.4 Forward()	755
39.87.3.5 OutputParameter() [1/2]	755
39.87.3.6 OutputParameter() [2/2]	756
39.87.3.7 serialize()	756
39.88KathirvalavakumarSubavathilInitialization Class Reference	756
39.88.1 Detailed Description	757
39.88.2 Constructor & Destructor Documentation	757
39.88.2.1 KathirvalavakumarSubavathilInitialization()	757
39.88.3 Member Function Documentation	757
39.88.3.1 Initialize() [1/2]	758
39.88.3.2 Initialize() [2/2]	758
39.89KLDivergence< InputDataType, OutputDataType > Class Template Reference	759
39.89.1 Detailed Description	759
39.89.2 Constructor & Destructor Documentation	760
39.89.2.1 KLDivergence()	760
39.89.3 Member Function Documentation	760
39.89.3.1 Backward()	760
39.89.3.2 Forward()	760
39.89.3.3 OutputParameter() [1/2]	761
39.89.3.4 OutputParameter() [2/2]	761
39.89.3.5 serialize()	761
39.89.3.6 TakeMean() [1/2]	761
39.89.3.7 TakeMean() [2/2]	762
39.90LayerNorm< InputDataType, OutputDataType > Class Template Reference	762
39.90.1 Detailed Description	763
39.90.2 Constructor & Destructor Documentation	764

39.90.2.1 LayerNorm() [1/2]	764
39.90.2.2 LayerNorm() [2/2]	764
39.90.3 Member Function Documentation	764
39.90.3.1 Backward()	764
39.90.3.2 Delta() [1/2]	765
39.90.3.3 Delta() [2/2]	765
39.90.3.4 Forward()	765
39.90.3.5 Gradient() [1/3]	765
39.90.3.6 Gradient() [2/3]	767
39.90.3.7 Gradient() [3/3]	767
39.90.3.8 Mean()	767
39.90.3.9 OutputParameter() [1/2]	767
39.90.3.10OutputParameter() [2/2]	768
39.90.3.11Parameters() [1/2]	768
39.90.3.12Parameters() [2/2]	768
39.90.3.13Reset()	768
39.90.3.14Serialize()	768
39.90.3.15Variance()	769
39.91 LayerTraits< LayerType > Class Template Reference	769
39.91.1 Detailed Description	769
39.91.2 Member Data Documentation	769
39.91.2.1 IsBiasLayer	770
39.91.2.2 IsBinary	770
39.91.2.3 IsConnection	770
39.91.2.4 IsLSTMLayer	770
39.91.2.5 IsOutputLayer	770
39.92 LeakyReLU< InputDataType, OutputDataType > Class Template Reference	771
39.92.1 Detailed Description	771

39.92.2 Constructor & Destructor Documentation	772
39.92.2.1 LeakyReLU()	772
39.92.3 Member Function Documentation	772
39.92.3.1 Alpha() [1/2]	772
39.92.3.2 Alpha() [2/2]	772
39.92.3.3 Backward()	773
39.92.3.4 Delta() [1/2]	773
39.92.3.5 Delta() [2/2]	773
39.92.3.6 Forward()	773
39.92.3.7 OutputParameter() [1/2]	774
39.92.3.8 OutputParameter() [2/2]	774
39.92.3.9 serialize()	774
39.93LecunNormalInitialization Class Reference	774
39.93.1 Detailed Description	775
39.93.2 Constructor & Destructor Documentation	775
39.93.2.1 LecunNormalInitialization()	775
39.93.3 Member Function Documentation	775
39.93.3.1 Initialize() [1/2]	775
39.93.3.2 Initialize() [2/2]	776
39.94Linear< InputDataType, OutputDataType > Class Template Reference	776
39.94.1 Detailed Description	777
39.94.2 Constructor & Destructor Documentation	778
39.94.2.1 Linear() [1/2]	778
39.94.2.2 Linear() [2/2]	778
39.94.3 Member Function Documentation	778
39.94.3.1 Backward()	778
39.94.3.2 Delta() [1/2]	779
39.94.3.3 Delta() [2/2]	779

39.94.3.4 Forward()	779
39.94.3.5 Gradient() [1/3]	780
39.94.3.6 Gradient() [2/3]	780
39.94.3.7 Gradient() [3/3]	780
39.94.3.8 InputParameter() [1/2]	780
39.94.3.9 InputParameter() [2/2]	781
39.94.3.10 OutputParameter() [1/2]	781
39.94.3.11 OutputParameter() [2/2]	781
39.94.3.12 Parameters() [1/2]	781
39.94.3.13 Parameters() [2/2]	781
39.94.3.14 Reset()	782
39.94.3.15 Serialize()	782
39.95 LinearNoBias< InputDataType, OutputDataType > Class Template Reference	782
39.95.1 Detailed Description	783
39.95.2 Constructor & Destructor Documentation	783
39.95.2.1 LinearNoBias() [1/2]	783
39.95.2.2 LinearNoBias() [2/2]	784
39.95.3 Member Function Documentation	784
39.95.3.1 Backward()	784
39.95.3.2 Delta() [1/2]	784
39.95.3.3 Delta() [2/2]	785
39.95.3.4 Forward()	785
39.95.3.5 Gradient() [1/3]	785
39.95.3.6 Gradient() [2/3]	785
39.95.3.7 Gradient() [3/3]	786
39.95.3.8 InputParameter() [1/2]	786
39.95.3.9 InputParameter() [2/2]	786
39.95.3.10 OutputParameter() [1/2]	786

39.95.3.11	OutputParameter() [2/2]	786
39.95.3.12	Parameters() [1/2]	787
39.95.3.13	Parameters() [2/2]	787
39.95.3.14	Reset()	787
39.95.3.15	Serialize()	787
39.96	LoadOutputParameterVisitor Class Reference	788
39.96.1	Detailed Description	788
39.96.2	Constructor & Destructor Documentation	788
39.96.2.1	LoadOutputParameterVisitor()	788
39.96.3	Member Function Documentation	788
39.96.3.1	operator>()	789
39.97	LogisticFunction Class Reference	789
39.97.1	Detailed Description	789
39.97.2	Member Function Documentation	790
39.97.2.1	Deriv() [1/2]	790
39.97.2.2	Deriv() [2/2]	790
39.97.2.3	Fn() [1/2]	790
39.97.2.4	Fn() [2/2]	791
39.97.2.5	Inv() [1/2]	791
39.97.2.6	Inv() [2/2]	792
39.98	LogSoftMax< InputDataType, OutputDataType > Class Template Reference	792
39.98.1	Detailed Description	793
39.98.2	Constructor & Destructor Documentation	793
39.98.2.1	LogSoftMax()	793
39.98.3	Member Function Documentation	793
39.98.3.1	Backward()	793
39.98.3.2	Delta() [1/2]	794
39.98.3.3	Delta() [2/2]	794

39.98.3.4 Forward()	794
39.98.3.5 OutputParameter() [1/2]	795
39.98.3.6 OutputParameter() [2/2]	795
39.98.3.7 serialize()	795
39.99Lookup< InputDataType, OutputDataType > Class Template Reference	795
39.99.1 Detailed Description	796
39.99.2 Constructor & Destructor Documentation	797
39.99.2.1 Lookup()	797
39.99.3 Member Function Documentation	797
39.99.3.1 Backward()	797
39.99.3.2 Delta() [1/2]	798
39.99.3.3 Delta() [2/2]	798
39.99.3.4 Forward()	798
39.99.3.5 Gradient() [1/3]	798
39.99.3.6 Gradient() [2/3]	799
39.99.3.7 Gradient() [3/3]	799
39.99.3.8 OutputParameter() [1/2]	799
39.99.3.9 OutputParameter() [2/2]	799
39.99.3.10Parameters() [1/2]	799
39.99.3.11Parameters() [2/2]	800
39.99.3.12serialize()	800
39.100LossVisitor Class Reference	800
39.100.1Detailed Description	801
39.100.2Member Function Documentation	801
39.100.2.1operator()()	801
39.101LSTM< InputDataType, OutputDataType > Class Template Reference	801
39.101.1Detailed Description	802
39.101.2Constructor & Destructor Documentation	803

39.101.2.1	LSTM() [1/2]	803
39.101.2.2	LSTM() [2/2]	803
39.101.3	Member Function Documentation	803
39.101.3.1	Backward()	804
39.101.3.2	Delta() [1/2]	804
39.101.3.3	Delta() [2/2]	804
39.101.3.4	Forward() [1/2]	804
39.101.3.5	Forward() [2/2]	805
39.101.3.6	Gradient() [1/3]	805
39.101.3.7	Gradient() [2/3]	805
39.101.3.8	Gradient() [3/3]	806
39.101.3.9	OutputParameter() [1/2]	806
39.101.3.10	OutputParameter() [2/2]	806
39.101.3.11	Parameters() [1/2]	806
39.101.3.12	Parameters() [2/2]	806
39.101.3.13	Reset()	807
39.101.3.14	ResetCell()	807
39.101.3.15	rho() [1/2]	807
39.101.3.16	rho() [2/2]	807
39.101.3.17	Serialize()	807
39.102	MaxPooling< InputDataType, OutputDataType > Class Template Reference	808
39.102.1	Detailed Description	809
39.102.2	Constructor & Destructor Documentation	809
39.102.2.1	MaxPooling() [1/2]	809
39.102.2.2	MaxPooling() [2/2]	809
39.102.3	Member Function Documentation	810
39.102.3.1	Backward()	810
39.102.3.2	Delta() [1/2]	810

39.102.3.3	Delta() [2/2]	810
39.102.3.4	Deterministic() [1/2]	811
39.102.3.5	Deterministic() [2/2]	811
39.102.3.6	Forward()	811
39.102.3.7	InputHeight() [1/2]	811
39.102.3.8	InputHeight() [2/2]	812
39.102.3.9	InputWidth() [1/2]	812
39.102.3.10	InputWidth() [2/2]	812
39.102.3.10	OutputHeight() [1/2]	812
39.102.3.10	OutputHeight() [2/2]	812
39.102.3.10	OutputParameter() [1/2]	813
39.102.3.10	OutputParameter() [2/2]	813
39.102.3.10	OutputWidth() [1/2]	813
39.102.3.10	OutputWidth() [2/2]	813
39.102.3.15	Serialize()	813
39.103	MaxPoolingRule Class Reference	814
39.103.1	Detailed Description	814
39.103.2	Member Function Documentation	814
39.103.2.1	Pooling()	814
39.104	MeanPooling< InputDataType, OutputDataType > Class Template Reference	814
39.104.1	Detailed Description	816
39.104.2	Constructor & Destructor Documentation	817
39.104.2.1	MeanPooling() [1/2]	817
39.104.2.2	MeanPooling() [2/2]	817
39.104.3	Member Function Documentation	817
39.104.3.1	Backward()	818
39.104.3.2	Delta() [1/2]	818
39.104.3.3	Delta() [2/2]	818

39.104.3.4	Deterministic() [1/2]	818
39.104.3.5	Deterministic() [2/2]	819
39.104.3.6	Forward()	819
39.104.3.7	InputHeight() [1/2]	819
39.104.3.8	InputHeight() [2/2]	819
39.104.3.9	InputWidth() [1/2]	820
39.104.3.10	InputWidth() [2/2]	820
39.104.3.10	OutputHeight() [1/2]	820
39.104.3.10	OutputHeight() [2/2]	820
39.104.3.10	OutputParameter() [1/2]	820
39.104.3.10	OutputParameter() [2/2]	821
39.104.3.10	OutputWidth() [1/2]	821
39.104.3.10	OutputWidth() [2/2]	821
39.104.3.15	Serialize()	821
39.105	MeanPoolingRule Class Reference	822
39.105.1	Detailed Description	822
39.105.2	Member Function Documentation	822
39.105.2.1	Pooling()	822
39.105.2.2	Unpooling()	822
39.106	MeanSquaredError< InputDataType, OutputDataType > Class Template Reference	823
39.106.1	Detailed Description	823
39.106.2	Constructor & Destructor Documentation	823
39.106.2.1	MeanSquaredError()	824
39.106.3	Member Function Documentation	824
39.106.3.1	Backward()	824
39.106.3.2	Forward()	824
39.106.3.3	OutputParameter() [1/2]	825
39.106.3.4	OutputParameter() [2/2]	825

39.106.3.5	serialize()	825
39.107	MultiplyConstant< InputDataType, OutputDataType > Class Template Reference	825
39.107.1	Detailed Description	826
39.107.2	Constructor & Destructor Documentation	826
39.107.2.1	MultiplyConstant()	827
39.107.3	Member Function Documentation	827
39.107.3.1	Backward()	827
39.107.3.2	Delta() [1/2]	827
39.107.3.3	Delta() [2/2]	827
39.107.3.4	Forward()	828
39.107.3.5	OutputParameter() [1/2]	828
39.107.3.6	OutputParameter() [2/2]	828
39.107.3.7	serialize()	828
39.108	MultiplyMerge< InputDataType, OutputDataType, CustomLayers > Class Template Reference	829
39.108.1	Detailed Description	830
39.108.2	Constructor & Destructor Documentation	830
39.108.2.1	MultiplyMerge()	830
39.108.2.2	~MultiplyMerge()	830
39.108.3	Member Function Documentation	831
39.108.3.1	Add() [1/2]	831
39.108.3.2	Add() [2/2]	831
39.108.3.3	Backward()	831
39.108.3.4	Delta() [1/2]	831
39.108.3.5	Delta() [2/2]	832
39.108.3.6	Forward()	832
39.108.3.7	Gradient() [1/3]	832
39.108.3.8	Gradient() [2/3]	832
39.108.3.9	Gradient() [3/3]	833

39.108.3.10	Model()	833
39.108.3.10	OutputParameter()	[1/2]	833
39.108.3.10	OutputParameter()	[2/2]	833
39.108.3.10	Parameters()	[1/2]	833
39.108.3.10	Parameters()	[2/2]	834
39.108.3.10	Serialize()	834
39.109	NaiveConvolution< BorderMode > Class Template Reference	834
39.109.1	Detailed Description	835
39.109.2	Member Function Documentation	835
39.109.2.1	Convolution()	[1/5]	835
39.109.2.2	Convolution()	[2/5]	836
39.109.2.3	Convolution()	[3/5]	836
39.109.2.4	Convolution()	[4/5]	836
39.109.2.5	Convolution()	[5/5]	837
39.110	NegativeLogLikelihood< InputDataType, OutputDataType > Class Template Reference	837
39.110.1	Detailed Description	838
39.110.2	Constructor & Destructor Documentation	838
39.110.2.1	NegativeLogLikelihood()	838
39.110.3	Member Function Documentation	838
39.110.3.1	Backward()	838
39.110.3.2	Delta()	[1/2]	839
39.110.3.3	Delta()	[2/2]	839
39.110.3.4	Forward()	839
39.110.3.5	InputParameter()	[1/2]	840
39.110.3.6	InputParameter()	[2/2]	840
39.110.3.7	OutputParameter()	[1/2]	840
39.110.3.8	OutputParameter()	[2/2]	840
39.110.3.9	Serialize()	840

39.11	NetworkInitialization < InitializationRuleType, CustomLayers > Class Template Reference	841
39.111.	Detailed Description	841
39.111.	Constructor & Destructor Documentation	841
39.111.2.	NetworkInitialization()	841
39.111.	Member Function Documentation	841
39.111.3.	Initialize()	842
39.112	NguyenWidrowInitialization Class Reference	842
39.112.	Detailed Description	842
39.112.	Constructor & Destructor Documentation	843
39.112.2.	NguyenWidrowInitialization()	843
39.112.	Member Function Documentation	843
39.112.3.	Initialize() [1/2]	843
39.112.3.	Initialize() [2/2]	844
39.113	OivsInitialization < ActivationFunction > Class Template Reference	844
39.113.	Detailed Description	845
39.113.	Constructor & Destructor Documentation	845
39.113.2.	OivsInitialization()	845
39.113.	Member Function Documentation	846
39.113.3.	Initialize() [1/2]	846
39.113.3.	Initialize() [2/2]	846
39.114	OrthogonalInitialization Class Reference	848
39.114.	Detailed Description	848
39.114.	Constructor & Destructor Documentation	848
39.114.2.	OrthogonalInitialization()	848
39.114.	Member Function Documentation	849
39.114.3.	Initialize() [1/2]	849
39.114.3.	Initialize() [2/2]	849
39.115	OutputHeightVisitor Class Reference	850

39.115.1	Detailed Description	850
39.115.2	Member Function Documentation	850
39.115.2.1	operator>()	851
39.116	OutputParameterVisitor Class Reference	851
39.116.1	Detailed Description	851
39.116.2	Member Function Documentation	851
39.116.2.1	operator>()	852
39.117	OutputWidthVisitor Class Reference	852
39.117.1	Detailed Description	852
39.117.2	Member Function Documentation	852
39.117.2.1	operator>()	853
39.118	ParametersSetVisitor Class Reference	853
39.118.1	Detailed Description	853
39.118.2	Constructor & Destructor Documentation	853
39.118.2.1	ParametersSetVisitor()	854
39.118.3	Member Function Documentation	854
39.118.3.1	operator>()	854
39.119	ParametersVisitor Class Reference	854
39.119.1	Detailed Description	855
39.119.2	Constructor & Destructor Documentation	855
39.119.2.1	ParametersVisitor()	855
39.119.3	Member Function Documentation	855
39.119.3.1	operator>()	855
39.120	ReLU< InputDataType, OutputDataType > Class Template Reference	855
39.120.1	Detailed Description	856
39.120.2	Constructor & Destructor Documentation	857
39.120.2.1	ReLU()	857
39.120.3	Member Function Documentation	857

39.120.3.1Alpha() [1/2]	857
39.120.3.2Alpha() [2/2]	858
39.120.3.3Backward()	858
39.120.3.4Delta() [1/2]	858
39.120.3.5Delta() [2/2]	858
39.120.3.6Forward()	859
39.120.3.7Gradient() [1/3]	859
39.120.3.8Gradient() [2/3]	859
39.120.3.9Gradient() [3/3]	860
39.120.3.10OutputParameter() [1/2]	860
39.120.3.10OutputParameter() [2/2]	860
39.120.3.12Parameters() [1/2]	860
39.120.3.12Parameters() [2/2]	860
39.120.3.14Reset()	861
39.120.3.15Serialize()	861
39.121RandomInitialization Class Reference	861
39.121.1Detailed Description	861
39.121.2Constructor & Destructor Documentation	862
39.121.2.1RandomInitialization() [1/2]	862
39.121.2.2RandomInitialization() [2/2]	863
39.121.3Member Function Documentation	863
39.121.3.1Initialize() [1/2]	863
39.121.3.2Initialize() [2/2]	864
39.122BM< InitializationRuleType, DataType, PolicyType > Class Template Reference	864
39.122.1Detailed Description	867
39.122.2Member Typedef Documentation	868
39.122.2.1ElemType	868
39.122.2.2NetworkType	868

39.122.3	Constructor & Destructor Documentation	868
39.122.3.1	RBM()	868
39.122.4	Member Function Documentation	869
39.122.4.1	Evaluate()	869
39.122.4.2	FreeEnergy() [1/2]	869
39.122.4.3	FreeEnergy() [2/2]	870
39.122.4.4	Gibbs()	870
39.122.4.5	Gradient()	870
39.122.4.6	HiddenBias() [1/2]	871
39.122.4.7	HiddenBias() [2/2]	871
39.122.4.8	HiddenMean() [1/2]	871
39.122.4.9	HiddenMean() [2/2]	872
39.122.4.10	HiddenSize()	872
39.122.4.11	NumFunctions()	872
39.122.4.12	NumSteps()	872
39.122.4.13	Parameters() [1/2]	873
39.122.4.14	Parameters() [2/2]	873
39.122.4.15	Phase() [1/2]	873
39.122.4.16	Phase() [2/2]	873
39.122.4.17	PoolSize()	874
39.122.4.18	Reset() [1/2]	874
39.122.4.19	Reset() [2/2]	874
39.122.4.20	SampleHidden() [1/2]	874
39.122.4.21	SampleHidden() [2/2]	874
39.122.4.22	SampleSlab()	875
39.122.4.23	SampleSpike()	875
39.122.4.24	SampleVisible() [1/2]	876
39.122.4.25	SampleVisible() [2/2]	876

39.122.4.28	Serialize()	876
39.122.4.29	Shuffle()	876
39.122.4.30	SlabMean()	877
39.122.4.31	SlabPenalty()	877
39.122.4.32	SpikeBias() [1/2]	877
39.122.4.33	SpikeBias() [2/2]	877
39.122.4.34	SpikeMean()	877
39.122.4.35	Train()	878
39.122.4.36	VisibleBias() [1/2]	878
39.122.4.37	VisibleBias() [2/2]	878
39.122.4.38	VisibleMean() [1/2]	879
39.122.4.39	VisibleMean() [2/2]	879
39.122.4.40	VisiblePenalty() [1/2]	879
39.122.4.41	VisiblePenalty() [2/2]	879
39.122.4.42	VisibleSize()	880
39.122.4.43	Weight() [1/2]	880
39.122.4.44	Weight() [2/2]	880
39.123	ReconstructionLoss< InputDataType, OutputDataType, DistType > Class Template Reference	880
39.123.1	Detailed Description	881
39.123.2	Constructor & Destructor Documentation	881
39.123.2.1	ReconstructionLoss()	881
39.123.3	Member Function Documentation	882
39.123.3.1	Backward()	882
39.123.3.2	Forward()	882
39.123.3.3	OutputParameter() [1/2]	882
39.123.3.4	OutputParameter() [2/2]	883
39.123.3.5	Serialize()	883
39.124	RectifierFunction Class Reference	883

39.124.1	Detailed Description	884
39.124.2	Member Function Documentation	884
39.124.2.1	Deriv() [1/2]	884
39.124.2.2	Deriv() [2/2]	884
39.124.2.3	Fn() [1/3]	885
39.124.2.4	Fn() [2/3]	885
39.124.2.5	Fn() [3/3]	886
39.125	Recurrent< InputDataType, OutputDataType, CustomLayers > Class Template Reference	886
39.125.1	Detailed Description	887
39.125.2	Constructor & Destructor Documentation	887
39.125.2.1	Recurrent() [1/3]	888
39.125.2.2	~Recurrent()	888
39.125.2.3	Recurrent() [2/3]	888
39.125.2.4	Recurrent() [3/3]	888
39.125.3	Member Function Documentation	889
39.125.3.1	Backward()	889
39.125.3.2	Delta() [1/2]	889
39.125.3.3	Delta() [2/2]	889
39.125.3.4	Deterministic() [1/2]	890
39.125.3.5	Deterministic() [2/2]	890
39.125.3.6	Forward()	890
39.125.3.7	Gradient() [1/3]	890
39.125.3.8	Gradient() [2/3]	891
39.125.3.9	Gradient() [3/3]	891
39.125.3.10	Model()	891
39.125.3.11	OutputParameter() [1/2]	891
39.125.3.12	OutputParameter() [2/2]	891
39.125.3.13	Parameters() [1/2]	892

39.125.3.1	Parameters() [2 / 2]	892
39.125.3.1	Serialize()	892
39.126	RecurrentAttention < InputDataType, OutputDataType > Class Template Reference	892
39.126.1	Detailed Description	894
39.126.2	Constructor & Destructor Documentation	894
39.126.2.1	RecurrentAttention() [1 / 2]	894
39.126.2.2	RecurrentAttention() [2 / 2]	894
39.126.3	Member Function Documentation	895
39.126.3.1	Backward()	895
39.126.3.2	Delta() [1 / 2]	895
39.126.3.3	Delta() [2 / 2]	895
39.126.3.4	Deterministic() [1 / 2]	896
39.126.3.5	Deterministic() [2 / 2]	896
39.126.3.6	Forward()	896
39.126.3.7	Gradient() [1 / 3]	896
39.126.3.8	Gradient() [2 / 3]	897
39.126.3.9	Gradient() [3 / 3]	897
39.126.3.10	Model()	897
39.126.3.10	OutputParameter() [1 / 2]	897
39.126.3.10	OutputParameter() [2 / 2]	897
39.126.3.11	Parameters() [1 / 2]	898
39.126.3.11	Parameters() [2 / 2]	898
39.126.3.12	Serialize()	898
39.127	ReinforceNormal < InputDataType, OutputDataType > Class Template Reference	898
39.127.1	Detailed Description	899
39.127.2	Constructor & Destructor Documentation	900
39.127.2.1	ReinforceNormal()	900
39.127.3	Member Function Documentation	900

39.127.3.1	Backward()	900
39.127.3.2	Delta() [1/2]	900
39.127.3.3	Delta() [2/2]	901
39.127.3.4	Deterministic() [1/2]	901
39.127.3.5	Deterministic() [2/2]	901
39.127.3.6	Forward()	901
39.127.3.7	OutputParameter() [1/2]	902
39.127.3.8	OutputParameter() [2/2]	902
39.127.3.9	Reward() [1/2]	902
39.127.3.10	Reward() [2/2]	902
39.127.3.11	Serialize()	903
39.128	Reparametrization< InputDataType, OutputDataType > Class Template Reference	903
39.128.1	Detailed Description	904
39.128.2	Constructor & Destructor Documentation	904
39.128.2.1	Reparametrization() [1/2]	904
39.128.2.2	Reparametrization() [2/2]	904
39.128.3	Member Function Documentation	905
39.128.3.1	Backward()	905
39.128.3.2	Delta() [1/2]	905
39.128.3.3	Delta() [2/2]	905
39.128.3.4	Forward()	906
39.128.3.5	Loss()	906
39.128.3.6	OutputParameter() [1/2]	906
39.128.3.7	OutputParameter() [2/2]	906
39.128.3.8	OutputSize() [1/2]	907
39.128.3.9	OutputSize() [2/2]	907
39.128.3.10	Serialize()	907
39.129	ResetCellVisitor Class Reference	907

39.129.1	Detailed Description	908
39.129.2	Constructor & Destructor Documentation	908
39.129.2.1	ResetCellVisitor()	908
39.129.3	Member Function Documentation	908
39.129.3.1	operator>()	908
39.130	ResetVisitor Class Reference	909
39.130.1	Detailed Description	909
39.130.2	Member Function Documentation	909
39.130.2.1	operator>()	909
39.131	RewardSetVisitor Class Reference	910
39.131.1	Detailed Description	910
39.131.2	Constructor & Destructor Documentation	910
39.131.2.1	RewardSetVisitor()	910
39.131.3	Member Function Documentation	910
39.131.3.1	operator>()	911
39.132	RNN< OutputLayerType, InitializationRuleType, CustomLayers > Class Template Reference	911
39.132.1	Detailed Description	913
39.132.2	Member Typedef Documentation	913
39.132.2.1	NetworkType	913
39.132.3	Constructor & Destructor Documentation	913
39.132.3.1	RNN()	914
39.132.3.2	~RNN()	914
39.132.4	Member Function Documentation	914
39.132.4.1	Add() [1/2]	914
39.132.4.2	Add() [2/2]	915
39.132.4.3	Evaluate() [1/2]	915
39.132.4.4	Evaluate() [2/2]	915
39.132.4.5	EvaluateWithGradient()	916

39.132.4.6	Gradient()	916
39.132.4.7	NumFunctions()	917
39.132.4.8	Parameters() [1/2]	917
39.132.4.9	Parameters() [2/2]	917
39.132.4.10	Predict()	917
39.132.4.11	Predictors() [1/2]	918
39.132.4.12	Predictors() [2/2]	918
39.132.4.13	Reset()	918
39.132.4.14	ResetParameters()	918
39.132.4.15	Responses() [1/2]	919
39.132.4.16	Responses() [2/2]	919
39.132.4.17	Rho() [1/2]	919
39.132.4.18	Rho() [2/2]	919
39.132.4.19	Serialize()	919
39.132.4.20	Shuffle()	920
39.132.4.21	Train() [1/2]	920
39.132.4.22	Train() [2/2]	921
39.133	RunSetVisitor Class Reference	921
39.133.1	Detailed Description	922
39.133.2	Constructor & Destructor Documentation	922
39.133.2.1	RunSetVisitor()	922
39.133.3	Member Function Documentation	922
39.133.3.1	operator>()()	923
39.134	SaveOutputParameterVisitor Class Reference	923
39.134.1	Detailed Description	923
39.134.2	Constructor & Destructor Documentation	923
39.134.2.1	SaveOutputParameterVisitor()	924
39.134.3	Member Function Documentation	924

39.134.3.1operator>()	924
39.135.1Select< InputDataType, OutputDataType > Class Template Reference	924
39.135.1Detailed Description	925
39.135.2Constructor & Destructor Documentation	925
39.135.2.1Select()	925
39.135.3Member Function Documentation	925
39.135.3.1Backward()	925
39.135.3.2Delta() [1/2]	926
39.135.3.3Delta() [2/2]	926
39.135.3.4Forward()	926
39.135.3.5OutputParameter() [1/2]	927
39.135.3.6OutputParameter() [2/2]	927
39.135.3.7serialize()	927
39.136.1Sequential< InputDataType, OutputDataType, Residual, CustomLayers > Class Template Reference	927
39.136.1Detailed Description	929
39.136.2Constructor & Destructor Documentation	929
39.136.2.1Sequential()	929
39.136.2.2~Sequential()	930
39.136.3Member Function Documentation	930
39.136.3.1Add() [1/2]	930
39.136.3.2Add() [2/2]	930
39.136.3.3Backward()	930
39.136.3.4DeleteModules()	931
39.136.3.5Delta() [1/2]	931
39.136.3.6Delta() [2/2]	931
39.136.3.7Forward()	931
39.136.3.8Gradient() [1/3]	932
39.136.3.9Gradient() [2/3]	932

39.136.3.10	Gradient() [3/3]	932
39.136.3.11	InputParameter() [1/2]	932
39.136.3.12	InputParameter() [2/2]	933
39.136.3.13	Model()	933
39.136.3.14	OutputParameter() [1/2]	933
39.136.3.15	OutputParameter() [2/2]	933
39.136.3.16	Parameters() [1/2]	933
39.136.3.17	Parameters() [2/2]	934
39.136.3.18	Serialize()	934
39.137	SetInputHeightVisitor Class Reference	934
39.137.1	Detailed Description	935
39.137.2	Constructor & Destructor Documentation	935
39.137.2.1	SetInputHeightVisitor()	935
39.137.3	Member Function Documentation	935
39.137.3.1	operator>()	935
39.138	SetInputWidthVisitor Class Reference	935
39.138.1	Detailed Description	936
39.138.2	Constructor & Destructor Documentation	936
39.138.2.1	SetInputWidthVisitor()	936
39.138.3	Member Function Documentation	936
39.138.3.1	operator>()	936
39.139	SigmoidCrossEntropyError< InputDataType, OutputDataType > Class Template Reference	937
39.139.1	Detailed Description	937
39.139.2	Constructor & Destructor Documentation	938
39.139.2.1	SigmoidCrossEntropyError()	938
39.139.3	Member Function Documentation	938
39.139.3.1	Backward()	938
39.139.3.2	Forward()	938

39.139.3.3	<code>OutputParameter()</code> [1/2]	939
39.139.3.4	<code>OutputParameter()</code> [2/2]	939
39.139.3.5	<code>Serialize()</code>	939
39.140	<code>SoftplusFunction</code> Class Reference	940
39.140.1	Detailed Description	940
39.140.2	Member Function Documentation	940
39.140.2.1	<code>Deriv()</code> [1/2]	940
39.140.2.2	<code>Deriv()</code> [2/2]	941
39.140.2.3	<code>Fn()</code> [1/2]	941
39.140.2.4	<code>Fn()</code> [2/2]	942
39.140.2.5	<code>Inv()</code> [1/2]	942
39.140.2.6	<code>Inv()</code> [2/2]	943
39.141	<code>SoftsignFunction</code> Class Reference	943
39.141.1	Detailed Description	944
39.141.2	Member Function Documentation	944
39.141.2.1	<code>Deriv()</code> [1/2]	944
39.141.2.2	<code>Deriv()</code> [2/2]	944
39.141.2.3	<code>Fn()</code> [1/2]	945
39.141.2.4	<code>Fn()</code> [2/2]	945
39.141.2.5	<code>Inv()</code> [1/2]	946
39.141.2.6	<code>Inv()</code> [2/2]	946
39.142	<code>SpikeSlabRBM</code> Class Reference	947
39.142.1	Detailed Description	947
39.143	<code>Subview< InputDataType, OutputDataType ></code> Class Template Reference	947
39.143.1	Detailed Description	948
39.143.2	Constructor & Destructor Documentation	948
39.143.2.1	<code>Subview()</code>	948
39.143.3	Member Function Documentation	948

39.143.3.1	Backward()	949
39.143.3.2	Delta() [1/2]	949
39.143.3.3	Delta() [2/2]	949
39.143.3.4	Forward()	949
39.143.3.5	OutputParameter() [1/2]	950
39.143.3.6	OutputParameter() [2/2]	950
39.143.3.7	Serialize()	950
39.144	VDConvolution< BorderMode > Class Template Reference	950
39.144.1	Detailed Description	951
39.144.2	Member Function Documentation	951
39.144.2.1	Convolution() [1/4]	951
39.144.2.2	Convolution() [2/4]	952
39.144.2.3	Convolution() [3/4]	952
39.144.2.4	Convolution() [4/4]	952
39.145	WishFunction Class Reference	953
39.145.1	Detailed Description	953
39.145.2	Member Function Documentation	953
39.145.2.1	Deriv() [1/2]	953
39.145.2.2	Deriv() [2/2]	954
39.145.2.3	Fn() [1/3]	954
39.145.2.4	Fn() [2/3]	955
39.145.2.5	Fn() [3/3]	955
39.146	TanhFunction Class Reference	955
39.146.1	Detailed Description	956
39.146.2	Member Function Documentation	956
39.146.2.1	Deriv() [1/2]	956
39.146.2.2	Deriv() [2/2]	957
39.146.2.3	Fn() [1/2]	957

39.146.2.4Fn() [2/2]	958
39.146.2.5Inv() [1/2]	958
39.146.2.6Inv() [2/2]	958
39.147.TransposedConvolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, InputDataType, OutputDataType > Class Template Reference	959
39.147.1Detailed Description	960
39.147.2Constructor & Destructor Documentation	961
39.147.2.1TransposedConvolution() [1/2]	961
39.147.2.2TransposedConvolution() [2/2]	961
39.147.3Member Function Documentation	962
39.147.3.1Backward()	962
39.147.3.2Delta() [1/2]	962
39.147.3.3Delta() [2/2]	962
39.147.3.4Forward()	962
39.147.3.5Gradient() [1/3]	963
39.147.3.6Gradient() [2/3]	963
39.147.3.7Gradient() [3/3]	963
39.147.3.8InputHeight() [1/2]	963
39.147.3.9InputHeight() [2/2]	964
39.147.3.10InputParameter() [1/2]	964
39.147.3.11InputParameter() [2/2]	964
39.147.3.12InputWidth() [1/2]	964
39.147.3.13InputWidth() [2/2]	964
39.147.3.14OutputHeight() [1/2]	965
39.147.3.15OutputHeight() [2/2]	965
39.147.3.16OutputParameter() [1/2]	965
39.147.3.17OutputParameter() [2/2]	965
39.147.3.18OutputWidth() [1/2]	966

39.147.3.1	OutputWidth() [2/2]	966
39.147.3.2	Parameters() [1/2]	966
39.147.3.2	Parameters() [2/2]	966
39.147.3.2	Reset()	966
39.147.3.2	Serialize()	967
39.148	ValidConvolution Class Reference	967
39.148.1	Detailed Description	967
39.149	VRClassReward< InputDataType, OutputDataType > Class Template Reference	967
39.149.1	Detailed Description	968
39.149.2	Constructor & Destructor Documentation	968
39.149.2.1	VRClassReward()	968
39.149.3	Member Function Documentation	969
39.149.3.1	Add() [1/2]	969
39.149.3.2	Add() [2/2]	969
39.149.3.3	Backward()	969
39.149.3.4	Delta() [1/2]	970
39.149.3.5	Delta() [2/2]	970
39.149.3.6	Deterministic() [1/2]	970
39.149.3.7	Deterministic() [2/2]	970
39.149.3.8	Forward()	970
39.149.3.9	OutputParameter() [1/2]	971
39.149.3.10	OutputParameter() [2/2]	971
39.149.3.1	Serialize()	971
39.150	WeightSetVisitor Class Reference	972
39.150.1	Detailed Description	972
39.150.2	Constructor & Destructor Documentation	972
39.150.2.1	WeightSetVisitor()	972
39.150.3	Member Function Documentation	973

39.150.3.1operator()()	. 973
39.151WeightSizeVisitor Class Reference	. 973
39.151.1Detailed Description	. 973
39.151.2Member Function Documentation	. 974
39.151.2.1operator()()	. 974
39.152Backtrace Class Reference	. 974
39.152.1Detailed Description	. 974
39.152.2Constructor & Destructor Documentation	. 975
39.152.2.1Backtrace()	. 975
39.152.3Member Function Documentation	. 975
39.152.3.1ToString()	. 975
39.153CLIOption< N > Class Template Reference	. 975
39.153.1Detailed Description	. 976
39.153.2Constructor & Destructor Documentation	. 976
39.153.2.1CLIOption()	. 976
39.154ParameterType< T > Struct Template Reference	. 977
39.154.1Detailed Description	. 977
39.154.2Member Typedef Documentation	. 977
39.154.2.1type	. 977
39.155ParameterType< arma::Col< eT > > Struct Template Reference	. 978
39.155.1Detailed Description	. 978
39.155.2Member Typedef Documentation	. 978
39.155.2.1type	. 978
39.156ParameterType< arma::Mat< eT > > Struct Template Reference	. 978
39.156.1Detailed Description	. 979
39.156.2Member Typedef Documentation	. 979
39.156.2.1type	. 979
39.157ParameterType< arma::Row< eT > > Struct Template Reference	. 979

39.157.1	Detailed Description	979
39.157.2	Member Typedef Documentation	980
39.157.2.1	type	980
39.158	ParameterType< std::tuple< mlpack::data::DatasetMapper< PolicyType, std::string >, arma::Mat< eT > > > Struct Template Reference	980
39.158.1	Detailed Description	980
39.158.2	Member Typedef Documentation	980
39.158.2.1	type	981
39.159	ParameterTypeDeducer< HasSerialize, T > Struct Template Reference	981
39.159.1	Detailed Description	981
39.159.2	Member Typedef Documentation	981
39.159.2.1	type	981
39.160	ParameterTypeDeducer< true, T > Struct Template Reference	981
39.160.1	Detailed Description	982
39.160.2	Member Typedef Documentation	982
39.160.2.1	type	982
39.161	ProgramDoc Class Reference	982
39.161.1	Detailed Description	983
39.161.2	Constructor & Destructor Documentation	983
39.161.2.1	ProgramDoc()	983
39.161.3	Member Data Documentation	983
39.161.3.1	documentation	983
39.161.3.2	programName	984
39.162	BindingInfo Class Reference	984
39.162.1	Detailed Description	984
39.162.2	Member Function Documentation	984
39.162.2.1	GetProgramDoc()	985
39.162.2.2	Language()	985

39.162.2.3RegisterProgramDoc()	985
39.163IMDOption< T > Class Template Reference	985
39.163.1Detailed Description	986
39.163.2Constructor & Destructor Documentation	986
39.163.2.1MDOption()	986
39.164ProgramDocWrapper Class Reference	986
39.164.1Detailed Description	987
39.164.2Constructor & Destructor Documentation	987
39.164.2.1ProgramDocWrapper()	987
39.165PyOption< T > Class Template Reference	987
39.165.1Detailed Description	987
39.165.2Constructor & Destructor Documentation	988
39.165.2.1PyOption()	988
39.166ProgramDoc Class Reference	988
39.166.1Detailed Description	989
39.166.2Constructor & Destructor Documentation	989
39.166.2.1ProgramDoc()	989
39.166.3Member Data Documentation	989
39.166.3.1documentation	989
39.166.3.2programName	990
39.167TestOption< N > Class Template Reference	990
39.167.1Detailed Description	990
39.167.2Constructor & Destructor Documentation	990
39.167.2.1TestOption()	991
39.168BallBound< MetricType, VecType > Class Template Reference	991
39.168.1Detailed Description	993
39.168.2Member Typedef Documentation	993
39.168.2.1ElemType	994

39.168.2.2Vec	994
39.168.3 Constructor & Destructor Documentation	994
39.168.3.1BallBound() [1/5]	994
39.168.3.2BallBound() [2/5]	994
39.168.3.3BallBound() [3/5]	995
39.168.3.4BallBound() [4/5]	995
39.168.3.5BallBound() [5/5]	995
39.168.3.6~BallBound()	995
39.168.4 Member Function Documentation	995
39.168.4.1Center() [1/3]	996
39.168.4.2Center() [2/3]	996
39.168.4.3Center() [3/3]	996
39.168.4.4Contains()	996
39.168.4.5Diameter()	997
39.168.4.6Dim()	997
39.168.4.7MaxDistance() [1/2]	997
39.168.4.8MaxDistance() [2/2]	998
39.168.4.9Metric() [1/2]	998
39.168.4.10Metric() [2/2]	998
39.168.4.11MinDistance() [1/2]	998
39.168.4.12MinDistance() [2/2]	999
39.168.4.13MinWidth()	999
39.168.4.14operator=()	999
39.168.4.15operator[]()	999
39.168.4.16operator" =() [1/2]	1000
39.168.4.17operator" =() [2/2]	1000
39.168.4.18Radius() [1/2]	1000
39.168.4.19Radius() [2/2]	1000

39.168.4.20	RangeDistance() [1/2]	1001
39.168.4.21	RangeDistance() [2/2]	1001
39.168.4.22	Serialize()	1001
39.169	BoundTraits< BoundType > Struct Template Reference	1002
39.169.1	Detailed Description	1002
39.169.2	Member Data Documentation	1002
39.169.2.1	HasTightBounds	1002
39.170	BoundTraits< BallBound< MetricType, VecType > > Struct Template Reference	1003
39.170.1	Detailed Description	1003
39.170.2	Member Data Documentation	1003
39.170.2.1	HasTightBounds	1003
39.171	BoundTraits< CellBound< MetricType, ElemType > > Struct Template Reference	1003
39.171.1	Detailed Description	1004
39.171.2	Member Data Documentation	1004
39.171.2.1	HasTightBounds	1004
39.172	BoundTraits< HollowBallBound< MetricType, ElemType > > Struct Template Reference	1004
39.172.1	Detailed Description	1004
39.172.2	Member Data Documentation	1005
39.172.2.1	HasTightBounds	1005
39.173	BoundTraits< HRectBound< MetricType, ElemType > > Struct Template Reference	1005
39.173.1	Detailed Description	1005
39.173.2	Member Data Documentation	1005
39.173.2.1	HasTightBounds	1005
39.174	CellBound< MetricType, ElemType > Class Template Reference	1006
39.174.1	Detailed Description	1006
39.175	HollowBallBound< TMetricType, ElemType > Class Template Reference	1006
39.175.1	Detailed Description	1008
39.175.2	Member Typedef Documentation	1009

39.175.2.1MetricType	1009
39.175.3Constructor & Destructor Documentation	1009
39.175.3.1HollowBallBound() [1/5]	1009
39.175.3.2HollowBallBound() [2/5]	1009
39.175.3.3HollowBallBound() [3/5]	1009
39.175.3.4HollowBallBound() [4/5]	1010
39.175.3.5HollowBallBound() [5/5]	1010
39.175.3.6~HollowBallBound()	1010
39.175.4Member Function Documentation	1010
39.175.4.1Center() [1/3]	1011
39.175.4.2Center() [2/3]	1011
39.175.4.3Center() [3/3]	1011
39.175.4.4Contains() [1/2]	1011
39.175.4.5Contains() [2/2]	1012
39.175.4.6Diameter()	1012
39.175.4.7Dim()	1012
39.175.4.8HollowCenter() [1/2]	1013
39.175.4.9HollowCenter() [2/2]	1013
39.175.4.10InnerRadius() [1/2]	1013
39.175.4.11InnerRadius() [2/2]	1013
39.175.4.12MaxDistance() [1/2]	1013
39.175.4.13MaxDistance() [2/2]	1014
39.175.4.14Metric() [1/2]	1014
39.175.4.15Metric() [2/2]	1014
39.175.4.16MinDistance() [1/2]	1014
39.175.4.17MinDistance() [2/2]	1015
39.175.4.18MinWidth()	1015
39.175.4.19operator=()	1015

39.175.4.20	operator[]()	1016
39.175.4.21	operator" =() [1/2]	1016
39.175.4.22	operator" =() [2/2]	1016
39.175.4.23	OuterRadius() [1/2]	1017
39.175.4.24	OuterRadius() [2/2]	1017
39.175.4.25	RangeDistance() [1/2]	1017
39.175.4.26	RangeDistance() [2/2]	1017
39.175.4.27	Serialize()	1018
39.176.1	RectBound< MetricType, ElemType > Class Template Reference	1018
39.176.1	Detailed Description	1020
39.176.2	Constructor & Destructor Documentation	1020
39.176.2.1	HRectBound() [1/4]	1021
39.176.2.2	HRectBound() [2/4]	1021
39.176.2.3	HRectBound() [3/4]	1021
39.176.2.4	HRectBound() [4/4]	1021
39.176.2.5	~HRectBound()	1021
39.176.3	Member Function Documentation	1022
39.176.3.1	Center()	1022
39.176.3.2	Clear()	1022
39.176.3.3	Contains() [1/2]	1022
39.176.3.4	Contains() [2/2]	1022
39.176.3.5	Diameter()	1023
39.176.3.6	Dim()	1023
39.176.3.7	MaxDistance() [1/2]	1023
39.176.3.8	MaxDistance() [2/2]	1024
39.176.3.9	Metric() [1/2]	1024
39.176.3.10	Metric() [2/2]	1024
39.176.3.11	MinDistance() [1/2]	1024

39.176.3.12	MinDistance() [2/2]	1025
39.176.3.13	MinWidth() [1/2]	1025
39.176.3.14	MinWidth() [2/2]	1025
39.176.3.15	Operator &()	1025
39.176.3.16	Operator &=()	1026
39.176.3.17	Operator=()	1026
39.176.3.18	Operator[]() [1/2]	1026
39.176.3.19	Operator[]() [2/2]	1026
39.176.3.20	Operator" =() [1/2]	1026
39.176.3.21	Operator" =() [2/2]	1027
39.176.3.22	Overlap()	1027
39.176.3.23	RangeDistance() [1/2]	1027
39.176.3.24	RangeDistance() [2/2]	1028
39.176.3.25	Serialize()	1028
39.176.3.26	Volume()	1028
39.177	LMetric< MetricType > Struct Template Reference	1028
39.177.1	Detailed Description	1029
39.177.2	Member Data Documentation	1029
39.177.2.1	Value	1029
39.178	LMetric< metric::LMetric< Power, TakeRoot > > Struct Template Reference	1029
39.178.1	Detailed Description	1029
39.178.2	Member Data Documentation	1030
39.178.2.1	Value	1030
39.179	AverageInterpolation Class Reference	1030
39.179.1	Detailed Description	1030
39.179.2	Constructor & Destructor Documentation	1031
39.179.2.1	AverageInterpolation() [1/2]	1031
39.179.2.2	AverageInterpolation() [2/2]	1031

39.179.3	Member Function Documentation	1031
39.179.3.1	GetWeights()	1031
39.180	BatchSVDPolicy Class Reference	1032
39.180.1	Detailed Description	1032
39.180.2	Member Function Documentation	1033
39.180.2.1	Apply()	1033
39.180.2.2	GetNeighborhood()	1033
39.180.2.3	GetRating()	1034
39.180.2.4	GetRatingOfUser()	1034
39.180.2.5	H()	1035
39.180.2.6	Serialize()	1035
39.180.2.7	W()	1035
39.181	BiasSVDPolicy Class Reference	1035
39.181.1	Detailed Description	1036
39.181.2	Constructor & Destructor Documentation	1037
39.181.2.1	BiasSVDPolicy()	1037
39.181.3	Member Function Documentation	1037
39.181.3.1	Alpha() [1/2]	1037
39.181.3.2	Alpha() [2/2]	1038
39.181.3.3	Apply()	1038
39.181.3.4	GetNeighborhood()	1038
39.181.3.5	GetRating()	1039
39.181.3.6	GetRatingOfUser()	1039
39.181.3.7	H()	1040
39.181.3.8	Lambda() [1/2]	1040
39.181.3.9	Lambda() [2/2]	1040
39.181.3.10	MaxIterations() [1/2]	1040
39.181.3.11	MaxIterations() [2/2]	1041

39.181.3.12()	1041
39.181.3.13()	1041
39.181.3.14Serialize()	1041
39.181.3.15()	1042
39.182CFModel Class Reference	1042
39.182.1Detailed Description	1043
39.182.2Constructor & Destructor Documentation	1043
39.182.2.1CFModel()	1043
39.182.2.2~CFModel()	1043
39.182.3Member Function Documentation	1043
39.182.3.1CFPtr()	1043
39.182.3.2GetRecommendations() [1/2]	1044
39.182.3.3GetRecommendations() [2/2]	1044
39.182.3.4Predict()	1044
39.182.3.5Serialize()	1044
39.182.3.6Train()	1045
39.183CFType< DecompositionPolicy, NormalizationType > Class Template Reference	1045
39.183.1Detailed Description	1046
39.183.2Constructor & Destructor Documentation	1047
39.183.2.1CFType() [1/2]	1047
39.183.2.2CFType() [2/2]	1047
39.183.3Member Function Documentation	1048
39.183.3.1CleanData()	1048
39.183.3.2CleanedData()	1048
39.183.3.3Decomposition()	1048
39.183.3.4GetRecommendations() [1/2]	1048
39.183.3.5GetRecommendations() [2/2]	1049
39.183.3.6Normalization()	1049

39.183.3.7	NumUsersForSimilarity() [1/2]	1050
39.183.3.8	NumUsersForSimilarity() [2/2]	1050
39.183.3.9	Predict() [1/2]	1050
39.183.3.10	Predict() [2/2]	1051
39.183.3.11	Rank() [1/2]	1051
39.183.3.12	Rank() [2/2]	1051
39.183.3.13	Serialize()	1052
39.183.3.14	Train() [1/2]	1052
39.183.3.15	Train() [2/2]	1052
39.184	CombinedNormalization< NormalizationTypes > Class Template Reference	1053
39.184.1	Detailed Description	1054
39.184.2	Member Typedef Documentation	1054
39.184.2.1	TupleType	1054
39.184.3	Constructor & Destructor Documentation	1054
39.184.3.1	CombinedNormalization()	1054
39.184.4	Member Function Documentation	1054
39.184.4.1	Denormalize() [1/2]	1055
39.184.4.2	Denormalize() [2/2]	1055
39.184.4.3	Normalizations()	1055
39.184.4.4	Normalize()	1056
39.184.4.5	Serialize()	1056
39.185	CosineSearch Class Reference	1056
39.185.1	Detailed Description	1057
39.185.2	Constructor & Destructor Documentation	1057
39.185.2.1	CosineSearch()	1057
39.185.3	Member Function Documentation	1058
39.185.3.1	Search()	1058
39.186	DeleteVisitor Class Reference	1058

39.186.1	Detailed Description	1059
39.186.2	Member Function Documentation	1059
39.186.2.1	operator>()	1059
39.187	DummyClass Class Reference	1059
39.187.1	Detailed Description	1060
39.188	GetValueVisitor Class Reference	1060
39.188.1	Detailed Description	1060
39.188.2	Member Function Documentation	1060
39.188.2.1	operator>()	1061
39.189	ItemMeanNormalization Class Reference	1061
39.189.1	Detailed Description	1061
39.189.2	Constructor & Destructor Documentation	1062
39.189.2.1	ItemMeanNormalization()	1062
39.189.3	Member Function Documentation	1062
39.189.3.1	Denormalize() [1/2]	1062
39.189.3.2	Denormalize() [2/2]	1062
39.189.3.3	Mean()	1063
39.189.3.4	Normalize() [1/2]	1063
39.189.3.5	Normalize() [2/2]	1063
39.189.3.6	Serialize()	1064
39.190	LMetricSearch< TPower > Class Template Reference	1064
39.190.1	Detailed Description	1064
39.190.2	Member Typedef Documentation	1065
39.190.2.1	NeighborSearchType	1065
39.190.3	Constructor & Destructor Documentation	1065
39.190.3.1	LMetricSearch()	1065
39.190.4	Member Function Documentation	1065
39.190.4.1	Search()	1065

39.19	NMF Policy Class Reference	1066
39.191.	D etailed Description	1067
39.191.	M ember Function Documentation	1067
39.191.2.	A pply()	1067
39.191.2.2	G etNeighborhood()	1068
39.191.2.3	G etRating()	1068
39.191.2.4	G etRatingOfUser()	1068
39.191.2.5	H ()	1069
39.191.2.6	S erialize()	1069
39.191.2.7	W ()	1069
39.192	No Normalization Class Reference	1070
39.192.	D etailed Description	1070
39.192.	C onstructor & Destructor Documentation	1070
39.192.2.	N oNormalization()	1070
39.192.	M ember Function Documentation	1070
39.192.3.1	D enormalize() [1/2]	1070
39.192.3.2	D enormalize() [2/2]	1071
39.192.3.3	N ormalize()	1071
39.192.3.4	S erialize()	1071
39.193	O verallMeanNormalization Class Reference	1072
39.193.	D etailed Description	1072
39.193.	C onstructor & Destructor Documentation	1073
39.193.2.	O verallMeanNormalization()	1073
39.193.	M ember Function Documentation	1073
39.193.3.1	D enormalize() [1/2]	1073
39.193.3.2	D enormalize() [2/2]	1073
39.193.3.3	M ean()	1074
39.193.3.4	N ormalize() [1/2]	1074

39.193.3.5	Normalize() [2/2]	1074
39.193.3.6	Serialize()	1075
39.194	PearsonSearch Class Reference	1075
39.194.1	Detailed Description	1075
39.194.2	Constructor & Destructor Documentation	1076
39.194.2.1	PearsonSearch()	1076
39.194.3	Member Function Documentation	1076
39.194.3.1	Search()	1076
39.195	PredictVisitor< NeighborSearchPolicy, InterpolationPolicy > Class Template Reference	1077
39.195.1	Detailed Description	1077
39.195.2	Constructor & Destructor Documentation	1078
39.195.2.1	PredictVisitor()	1078
39.195.3	Member Function Documentation	1078
39.195.3.1	operator>()	1078
39.196	RandomizedSVDPolicy Class Reference	1078
39.196.1	Detailed Description	1079
39.196.2	Constructor & Destructor Documentation	1080
39.196.2.1	RandomizedSVDPolicy()	1080
39.196.3	Member Function Documentation	1080
39.196.3.1	Apply()	1080
39.196.3.2	GetNeighborhood()	1081
39.196.3.3	GetRating()	1081
39.196.3.4	GetRatingOfUser()	1082
39.196.3.5	H()	1082
39.196.3.6	IteratedPower() [1/2]	1082
39.196.3.7	IteratedPower() [2/2]	1082
39.196.3.8	MaxIterations() [1/2]	1083
39.196.3.9	MaxIterations() [2/2]	1083

39.196.3.1	Serialize()	1083
39.196.3.1	W()	1083
39.197	RecommendationVisitor< NeighborSearchPolicy, InterpolationPolicy > Class Template Reference . .	1084
39.197.1	Detailed Description	1084
39.197.2	Constructor & Destructor Documentation	1084
39.197.2.1	RecommendationVisitor()	1085
39.197.3	Member Function Documentation	1085
39.197.3.1	operator>()	1085
39.198	RegressionInterpolation Class Reference	1085
39.198.1	Detailed Description	1086
39.198.2	Constructor & Destructor Documentation	1086
39.198.2.1	RegressionInterpolation() [1/2]	1086
39.198.2.2	RegressionInterpolation() [2/2]	1086
39.198.3	Member Function Documentation	1087
39.198.3.1	GetWeights()	1087
39.199	RegSVDPolicy Class Reference	1088
39.199.1	Detailed Description	1088
39.199.2	Constructor & Destructor Documentation	1089
39.199.2.1	RegSVDPolicy()	1089
39.199.3	Member Function Documentation	1089
39.199.3.1	Apply()	1089
39.199.3.2	GetNeighborhood()	1090
39.199.3.3	GetRating()	1090
39.199.3.4	GetRatingOfUser()	1091
39.199.3.5	H()	1091
39.199.3.6	MaxIterations() [1/2]	1091
39.199.3.7	MaxIterations() [2/2]	1091
39.199.3.8	Serialize()	1092

39.199.3.9W()	1092
39.200.1SimilarityInterpolation Class Reference	1092
39.200.1Detailed Description	1093
39.200.2Constructor & Destructor Documentation	1093
39.200.2.1SimilarityInterpolation() [1/2]	1093
39.200.2.2SimilarityInterpolation() [2/2]	1093
39.200.3Member Function Documentation	1093
39.200.3.1GetWeights()	1094
39.201.1SVDCompletePolicy Class Reference	1094
39.201.1Detailed Description	1095
39.201.2Member Function Documentation	1095
39.201.2.1Apply()	1095
39.201.2.2GetNeighborhood()	1096
39.201.2.3GetRating()	1096
39.201.2.4GetRatingOfUser()	1097
39.201.2.5H()	1097
39.201.2.6Serialize()	1097
39.201.2.7W()	1098
39.202.1SVDIncompletePolicy Class Reference	1098
39.202.1Detailed Description	1099
39.202.2Member Function Documentation	1099
39.202.2.1Apply()	1099
39.202.2.2GetNeighborhood()	1100
39.202.2.3GetRating()	1100
39.202.2.4GetRatingOfUser()	1100
39.202.2.5H()	1101
39.202.2.6Serialize()	1101
39.202.2.7W()	1101

39.203	SVDPlusPlusPolicy Class Reference	1102
39.203.1	Detailed Description	1103
39.203.2	Constructor & Destructor Documentation	1103
39.203.2.1	SVDPlusPlusPolicy()	1103
39.203.3	Member Function Documentation	1103
39.203.3.1	Alpha() [1/2]	1104
39.203.3.2	Alpha() [2/2]	1104
39.203.3.3	Apply()	1104
39.203.3.4	GetNeighborhood()	1105
39.203.3.5	GetRating()	1105
39.203.3.6	GetRatingOfUser()	1105
39.203.3.7	H()	1106
39.203.3.8	ImplicitData()	1106
39.203.3.9	Lambda() [1/2]	1106
39.203.3.10	Lambda() [2/2]	1106
39.203.3.11	MaxIterations() [1/2]	1107
39.203.3.12	MaxIterations() [2/2]	1107
39.203.3.13	B()	1107
39.203.3.14	Q()	1107
39.203.3.15	Serialize()	1107
39.203.3.16	W()	1108
39.203.3.17	Y()	1108
39.204	SVDWrapper< Factorizer > Class Template Reference	1108
39.204.1	Detailed Description	1108
39.204.2	Constructor & Destructor Documentation	1109
39.204.2.1	SVDWrapper()	1109
39.204.3	Member Function Documentation	1109
39.204.3.1	Apply() [1/2]	1109

39.204.3.2	Apply() [2/2]	1109
39.205	UserMeanNormalization Class Reference	1111
39.205.1	Detailed Description	1111
39.205.2	Constructor & Destructor Documentation	1112
39.205.2.1	UserMeanNormalization()	1112
39.205.3	Member Function Documentation	1112
39.205.3.1	Denormalize() [1/2]	1112
39.205.3.2	Denormalize() [2/2]	1112
39.205.3.3	Mean()	1113
39.205.3.4	Normalize() [1/2]	1113
39.205.3.5	Normalize() [2/2]	1113
39.205.3.6	Serialize()	1114
39.206	ScoreNormalization Class Reference	1114
39.206.1	Detailed Description	1114
39.206.2	Constructor & Destructor Documentation	1115
39.206.2.1	ScoreNormalization()	1115
39.206.3	Member Function Documentation	1115
39.206.3.1	Denormalize() [1/2]	1115
39.206.3.2	Denormalize() [2/2]	1115
39.206.3.3	Mean()	1116
39.206.3.4	Normalize() [1/2]	1116
39.206.3.5	Normalize() [2/2]	1116
39.206.3.6	Serialize()	1117
39.206.3.7	Stddev()	1117
39.207	CLI Class Reference	1117
39.207.1	Detailed Description	1119
39.207.2	Adding parameters to a program	1119
39.207.3	Documenting the program itself	1120

39.207.4	Parsing the command line with CLI	1120
39.207.5	Getting parameters with CLI	1120
39.207.6	Member Typedef Documentation	1121
39.207.6.1	FunctionMapType	1121
39.207.7	Member Function Documentation	1121
39.207.7.1	Add()	1121
39.207.7.2	Aliases()	1122
39.207.7.3	ClearSettings()	1122
39.207.7.4	GetParam()	1122
39.207.7.5	GetPrintableParam()	1122
39.207.7.6	GetRawParam()	1123
39.207.7.7	GetSingleton()	1123
39.207.7.8	HasParam()	1124
39.207.7.9	Parameters()	1124
39.207.7.10	ProgramName()	1124
39.207.7.11	RegisterProgramDoc()	1124
39.207.7.12	RestoreSettings()	1125
39.207.7.13	SetPassed()	1125
39.207.7.14	StoreSettings()	1125
39.207.8	Member Data Documentation	1126
39.207.8.1	didParse	1126
39.207.8.2	doc	1126
39.207.8.3	functionMap	1126
39.207.8.4	programName	1127
39.207.8.5	timer	1127
39.208	Accuracy Class Reference	1127
39.208.1	Detailed Description	1128
39.208.2	Member Function Documentation	1128

39.208.2.1Evaluate()	1128
39.208.3Member Data Documentation	1128
39.208.3.1NeedsMinimization	1128
39.209CVBase< MLAlgorithm, MatType, PredictionsType, WeightsType > Class Template Reference	1129
39.209.1Detailed Description	1129
39.209.2Member Typedef Documentation	1130
39.209.2.1MIE	1130
39.209.3Constructor & Destructor Documentation	1130
39.209.3.1CVBase() [1/3]	1130
39.209.3.2CVBase() [2/3]	1130
39.209.3.3CVBase() [3/3]	1131
39.209.4Member Function Documentation	1131
39.209.4.1AssertDataConsistency()	1131
39.209.4.2AssertWeightsConsistency()	1131
39.209.4.3Train() [1/2]	1132
39.209.4.4Train() [2/2]	1132
39.210F1< AS, PositiveClass > Class Template Reference	1132
39.210.1Detailed Description	1133
39.210.2Member Function Documentation	1133
39.210.2.1Evaluate()	1133
39.210.3Member Data Documentation	1134
39.210.3.1NeedsMinimization	1134
39.211KFoldCV< MLAlgorithm, Metric, MatType, PredictionsType, WeightsType > Class Template Reference	1134
39.211.1Detailed Description	1135
39.211.2Constructor & Destructor Documentation	1136
39.211.2.1KFoldCV() [1/6]	1136
39.211.2.2KFoldCV() [2/6]	1136
39.211.2.3KFoldCV() [3/6]	1137

39.211.2.4	KFoldCV() [4/6]	1137
39.211.2.5	KFoldCV() [5/6]	1138
39.211.2.6	KFoldCV() [6/6]	1138
39.211.3	Member Function Documentation	1139
39.211.3.1	Evaluate()	1139
39.211.3.2	Model()	1139
39.211.3.3	Shuffle() [1/2]	1139
39.211.3.4	Shuffle() [2/2]	1139
39.212	MetaInfoExtractor< MLAlgorithm, MT, PT, WT > Class Template Reference	1140
39.212.1	Detailed Description	1140
39.212.2	Member Typedef Documentation	1141
39.212.2.1	PredictionsType	1141
39.212.2.2	WeightsType	1141
39.212.3	Member Data Documentation	1141
39.212.3.1	IsSupported	1142
39.212.3.2	SupportsWeights	1142
39.212.3.3	TakesDatasetInfo	1142
39.212.3.4	TakesNumClasses	1142
39.213	MSE Class Reference	1143
39.213.1	Detailed Description	1143
39.213.2	Member Function Documentation	1143
39.213.2.1	Evaluate()	1143
39.213.3	Member Data Documentation	1144
39.213.3.1	NeedsMinimization	1144
39.214	NotFoundMethodForm Struct Reference	1144
39.214.1	Detailed Description	1144
39.214.2	Member Typedef Documentation	1144
39.214.2.1	PredictionsType	1144

39.214.2.2WeightsType	1145
39.215Precision< AS, PositiveClass > Class Template Reference	1145
39.215.1Detailed Description	1145
39.215.2Member Function Documentation	1146
39.215.2.1Evaluate()	1146
39.215.3Member Data Documentation	1146
39.215.3.1NeedsMinimization	1146
39.216Recall< AS, PositiveClass > Class Template Reference	1147
39.216.1Detailed Description	1147
39.216.2Member Function Documentation	1148
39.216.2.1Evaluate()	1148
39.216.3Member Data Documentation	1148
39.216.3.1NeedsMinimization	1148
39.217SelectMethodForm< MLAlgorithm, HMFs > Struct Template Reference	1148
39.217.1Detailed Description	1149
39.218SelectMethodForm< MLAlgorithm > Struct Template Reference	1149
39.218.1Detailed Description	1149
39.219SelectMethodForm< MLAlgorithm >::From< Forms > Struct Template Reference	1149
39.219.1Detailed Description	1150
39.219.2Member Typedef Documentation	1150
39.219.2.1Type	1150
39.220SelectMethodForm< MLAlgorithm, HasMethodForm, HMFs... > Struct Template Reference	1150
39.220.1Detailed Description	1150
39.221SelectMethodForm< MLAlgorithm, HasMethodForm, HMFs... >::From< Forms > Class Template Reference	1151
39.221.1Detailed Description	1151
39.221.2Member Typedef Documentation	1151
39.221.2.1Type	1151

39.22	S impleCV< MLAlgorithm, Metric, MatType, PredictionsType, WeightsType > Class Template Reference	1151
39.222.	1 Detailed Description	1152
39.222.	2 Constructor & Destructor Documentation	1153
39.222.2.	1 SimpleCV() [1/6]	1153
39.222.2.	2 SimpleCV() [2/6]	1154
39.222.2.	3 SimpleCV() [3/6]	1154
39.222.2.	4 SimpleCV() [4/6]	1155
39.222.2.	5 SimpleCV() [5/6]	1155
39.222.2.	6 SimpleCV() [6/6]	1156
39.222.	3 Member Function Documentation	1157
39.222.3.	1 Evaluate()	1157
39.222.3.	2 Model()	1157
39.22	T rainForm< MatType, PredictionsType, WeightsType, DatasetInfo, NumClasses > Struct Template Reference	1157
39.223.	1 Detailed Description	1157
39.22	4 rainForm< MT, PT, void, false, false > Struct Template Reference	1158
39.224.	1 Detailed Description	1158
39.22	5 rainForm< MT, PT, void, false, true > Struct Template Reference	1159
39.225.	1 Detailed Description	1159
39.22	6 rainForm< MT, PT, void, true, false > Struct Template Reference	1160
39.226.	1 Detailed Description	1160
39.22	7 rainForm< MT, PT, void, true, true > Struct Template Reference	1160
39.227.	1 Detailed Description	1161
39.22	8 rainForm< MT, PT, WT, false, false > Struct Template Reference	1161
39.228.	1 Detailed Description	1161
39.22	9 rainForm< MT, PT, WT, false, true > Struct Template Reference	1162
39.229.	1 Detailed Description	1162
39.23	0 rainForm< MT, PT, WT, true, false > Struct Template Reference	1162

39.230.	Detailed Description	1163
39.231	TrainForm< MT, PT, WT, true, true > Struct Template Reference	1163
39.231.	Detailed Description	1163
39.232	TrainFormBase4< PT, WT, T1, T2 > Struct Template Reference	1164
39.232.	Detailed Description	1164
39.232.1	Member Typedef Documentation	1164
39.232.2.1	PredictionsType	1165
39.232.2.2	Type	1165
39.232.2.3	WeightsType	1165
39.232.3	Member Data Documentation	1165
39.232.3.1	MinNumberOfAdditionalArgs	1165
39.233	TrainFormBase5< PT, WT, T1, T2, T3 > Struct Template Reference	1165
39.233.	Detailed Description	1166
39.233.1	Member Typedef Documentation	1166
39.233.2.1	PredictionsType	1166
39.233.2.2	Type	1166
39.233.2.3	WeightsType	1166
39.233.3	Member Data Documentation	1167
39.233.3.1	MinNumberOfAdditionalArgs	1167
39.234	TrainFormBase6< PT, WT, T1, T2, T3, T4 > Struct Template Reference	1167
39.234.	Detailed Description	1167
39.234.1	Member Typedef Documentation	1168
39.234.2.1	PredictionsType	1168
39.234.2.2	Type	1168
39.234.2.3	WeightsType	1168
39.234.3	Member Data Documentation	1168
39.234.3.1	MinNumberOfAdditionalArgs	1168
39.235	TrainFormBase7< PT, WT, T1, T2, T3, T4, T5 > Struct Template Reference	1169

39.235.1Detailed Description	1169
39.235.2Member Typedef Documentation	1169
39.235.2.1PredictionsType	1169
39.235.2.2Type	1170
39.235.2.3WeightsType	1170
39.235.3Member Data Documentation	1170
39.235.3.1MinNumberOfAdditionalArgs	1170
39.236CustomImputation< T > Class Template Reference	1170
39.236.1Detailed Description	1170
39.236.2Constructor & Destructor Documentation	1171
39.236.2.1CustomImputation()	1171
39.236.3Member Function Documentation	1171
39.236.3.1Impute()	1171
39.237DatasetMapper< PolicyType, InputType > Class Template Reference	1172
39.237.1Detailed Description	1173
39.237.2Constructor & Destructor Documentation	1173
39.237.2.1DatasetMapper() [1/2]	1174
39.237.2.2DatasetMapper() [2/2]	1174
39.237.3Member Function Documentation	1174
39.237.3.1Dimensionality()	1174
39.237.3.2MapFirstPass()	1174
39.237.3.3MapString()	1175
39.237.3.4NumMappings()	1175
39.237.3.5NumUnmappings()	1175
39.237.3.6Policy() [1/3]	1176
39.237.3.7Policy() [2/3]	1176
39.237.3.8Policy() [3/3]	1176
39.237.3.9Serialize()	1176

39.237.3.1	SetDimensionality()	1176
39.237.3.1	Type() [1/2]	1177
39.237.3.1	Type() [2/2]	1177
39.237.3.1	InmapString()	1177
39.237.3.1	InmapValue()	1178
39.238	Serialize< T > Struct Template Reference	1178
39.238.1	Detailed Description	1179
39.238.2	Member Typedef Documentation	1179
39.238.2.1	no	1179
39.238.2.2	yes	1179
39.238.3	Member Function Documentation	1179
39.238.3.1	chk() [1/2]	1179
39.238.3.2	chk() [2/2]	1180
39.238.4	Member Data Documentation	1180
39.238.4.1	value	1180
39.239	Serialize< T >::check< U, V, W > Struct Template Reference	1180
39.239.1	Detailed Description	1180
39.240	SerializeFunction< T > Struct Template Reference	1180
39.240.1	Detailed Description	1181
39.240.2	Member Typedef Documentation	1181
39.240.2.1	NonStaticSerialize	1181
39.240.2.2	StaticSerialize	1181
39.240.3	Member Data Documentation	1181
39.240.3.1	value	1181
39.241	Imputer< T, MapperType, StrategyType > Class Template Reference	1182
39.241.1	Detailed Description	1182
39.241.2	Constructor & Destructor Documentation	1182
39.241.2.1	Imputer() [1/2]	1183

39.241.2.2Imputer() [2/2]	1183
39.241.3Member Function Documentation	1183
39.241.3.1Impute()	1183
39.241.3.2Mapper() [1/2]	1183
39.241.3.3Mapper() [2/2]	1184
39.241.3.4Strategy() [1/2]	1184
39.241.3.5Strategy() [2/2]	1184
39.242IncrementPolicy Class Reference	1184
39.242.1Detailed Description	1185
39.242.2Member Typedef Documentation	1185
39.242.2.1MappedType	1185
39.242.3Constructor & Destructor Documentation	1185
39.242.3.1IncrementPolicy()	1186
39.242.4Member Function Documentation	1186
39.242.4.1MapFirstPass()	1186
39.242.4.2MapString()	1186
39.242.5Member Data Documentation	1187
39.242.5.1NeedsFirstPass	1187
39.243BitwiseDeletion< T > Class Template Reference	1187
39.243.1Detailed Description	1187
39.243.2Member Function Documentation	1188
39.243.2.1Impute()	1188
39.244LoadCSV Class Reference	1188
39.244.1Detailed Description	1189
39.244.2Constructor & Destructor Documentation	1189
39.244.2.1LoadCSV()	1189
39.244.3Member Function Documentation	1189
39.244.3.1GetMatrixSize()	1189

39.244.3.2	GetTransposeMatrixSize()	1190
39.244.3.3	Load()	1190
39.245	MeanImputation< T > Class Template Reference	1191
39.245.1	Detailed Description	1191
39.245.2	Member Function Documentation	1191
39.245.2.1	Impute()	1192
39.246	MedianImputation< T > Class Template Reference	1192
39.246.1	Detailed Description	1192
39.246.2	Member Function Documentation	1193
39.246.2.1	Impute()	1193
39.247	MissingPolicy Class Reference	1193
39.247.1	Detailed Description	1194
39.247.2	Member Typedef Documentation	1194
39.247.2.1	MappedType	1194
39.247.3	Constructor & Destructor Documentation	1194
39.247.3.1	MissingPolicy() [1/2]	1195
39.247.3.2	MissingPolicy() [2/2]	1195
39.247.4	Member Function Documentation	1195
39.247.4.1	MapFirstPass()	1195
39.247.4.2	MapString()	1196
39.247.5	Member Data Documentation	1197
39.247.5.1	NeedsFirstPass	1197
39.248	DBSCAN< RangeSearchType, PointSelectionPolicy > Class Template Reference	1197
39.248.1	Detailed Description	1198
39.248.2	Constructor & Destructor Documentation	1198
39.248.2.1	DBSCAN()	1198
39.248.3	Member Function Documentation	1199
39.248.3.1	Cluster() [1/3]	1199

39.248.3.2	Cluster() [2/3]	1199
39.248.3.3	Cluster() [3/3]	1200
39.249	OrderedPointSelection Class Reference	1200
39.249.1	Detailed Description	1201
39.249.2	Member Function Documentation	1201
39.249.2.1	Select()	1201
39.250	RandomPointSelection Class Reference	1201
39.250.1	Detailed Description	1201
39.250.2	Member Function Documentation	1202
39.250.2.1	Select()	1202
39.251	DecisionStump< MatType > Class Template Reference	1202
39.251.1	Detailed Description	1203
39.251.2	Constructor & Destructor Documentation	1203
39.251.2.1	DecisionStump() [1/3]	1203
39.251.2.2	DecisionStump() [2/3]	1204
39.251.2.3	DecisionStump() [3/3]	1204
39.251.3	Member Function Documentation	1205
39.251.3.1	BinLabels() [1/2]	1205
39.251.3.2	BinLabels() [2/2]	1205
39.251.3.3	Classify()	1205
39.251.3.4	Serialize()	1205
39.251.3.5	Split() [1/2]	1206
39.251.3.6	Split() [2/2]	1206
39.251.3.7	SplitDimension() [1/2]	1206
39.251.3.8	SplitDimension() [2/2]	1206
39.251.3.9	Train() [1/2]	1206
39.251.3.10	Train() [2/2]	1207
39.252	Tree< MatType, TagType > Class Template Reference	1207

39.252.1 Detailed Description	1210
39.252.2 Member Typedef Documentation	1210
39.252.2.1 ElemType	1210
39.252.2.2 StatType	1210
39.252.2.3 VecType	1211
39.252.3 Constructor & Destructor Documentation	1211
39.252.3.1 DTree() [1/7]	1211
39.252.3.2 DTree() [2/7]	1211
39.252.3.3 DTree() [3/7]	1211
39.252.3.4 DTree() [4/7]	1212
39.252.3.5 DTree() [5/7]	1212
39.252.3.6 DTree() [6/7]	1212
39.252.3.7 DTree() [7/7]	1213
39.252.3.8 ~DTree()	1213
39.252.4 Member Function Documentation	1213
39.252.4.1 AlphaUpper()	1214
39.252.4.2 BucketTag()	1214
39.252.4.3 Child()	1214
39.252.4.4 ChildPtr()	1214
39.252.4.5 ComputeValue()	1215
39.252.4.6 ComputeVariableImportance()	1215
39.252.4.7 End()	1215
39.252.4.8 FindBucket()	1215
39.252.4.9 Grow()	1216
39.252.4.10 Left()	1216
39.252.4.11 logNegativeError()	1216
39.252.4.12 logNegError()	1217
39.252.4.13 logVolume()	1217

39.252.4.1	MaxVals()	1217
39.252.4.1	MinVals()	1217
39.252.4.1	NumChildren()	1218
39.252.4.1	Operator=() [1/2]	1218
39.252.4.1	Operator=() [2/2]	1218
39.252.4.1	PruneAndUpdate()	1218
39.252.4.2	Ratio()	1219
39.252.4.2	Right()	1219
39.252.4.2	Root()	1219
39.252.4.2	Serialize()	1220
39.252.4.2	SplitDim()	1220
39.252.4.2	SplitValue()	1220
39.252.4.2	Start()	1220
39.252.4.2	SubtreeLeaves()	1220
39.252.4.2	SubtreeLeavesLogNegError()	1221
39.252.4.2	TagTree()	1221
39.252.4.3	WithinRange()	1221
39.253	PathCacher Class Reference	1221
39.253.1	Detailed Description	1222
39.253.2	Member Typedef Documentation	1223
39.253.2.1	PathCacheType	1223
39.253.2.2	PathType	1223
39.253.3	Member Enumeration Documentation	1223
39.253.3.1	PathFormat	1223
39.253.4	Constructor & Destructor Documentation	1223
39.253.4.1	PathCacher()	1224
39.253.5	Member Function Documentation	1224
39.253.5.1	BuildString()	1224

39.253.5.2	Enter()	1224
39.253.5.3	Leave()	1224
39.253.5.4	NumNodes()	1225
39.253.5.5	ParentOf()	1225
39.253.5.6	PathFor()	1225
39.253.6	Member Data Documentation	1225
39.253.6.1	format	1225
39.253.6.2	path	1225
39.253.6.3	pathCache	1226
39.254	DiagonalGaussianDistribution Class Reference	1226
39.254.1	Detailed Description	1227
39.254.2	Constructor & Destructor Documentation	1227
39.254.2.1	DiagonalGaussianDistribution() [1/3]	1227
39.254.2.2	DiagonalGaussianDistribution() [2/3]	1227
39.254.2.3	DiagonalGaussianDistribution() [3/3]	1228
39.254.3	Member Function Documentation	1228
39.254.3.1	Covariance() [1/3]	1228
39.254.3.2	Covariance() [2/3]	1228
39.254.3.3	Covariance() [3/3]	1228
39.254.3.4	Dimensionality()	1229
39.254.3.5	LogProbability() [1/2]	1229
39.254.3.6	LogProbability() [2/2]	1229
39.254.3.7	Mean() [1/2]	1229
39.254.3.8	Mean() [2/2]	1230
39.254.3.9	Probability() [1/2]	1230
39.254.3.10	Probability() [2/2]	1230
39.254.3.11	Random()	1230
39.254.3.12	Serialize()	1231

39.254.3.1	<code>Train()</code> [1/2]	1231
39.254.3.1	<code>Train()</code> [2/2]	1231
39.255	<code>DiscreteDistribution</code> Class Reference	1232
39.255.1	Detailed Description	1233
39.255.2	Constructor & Destructor Documentation	1233
39.255.2.1	<code>DiscreteDistribution()</code> [1/4]	1233
39.255.2.2	<code>DiscreteDistribution()</code> [2/4]	1233
39.255.2.3	<code>DiscreteDistribution()</code> [3/4]	1234
39.255.2.4	<code>DiscreteDistribution()</code> [4/4]	1234
39.255.3	Member Function Documentation	1234
39.255.3.1	<code>Dimensionality()</code>	1235
39.255.3.2	<code>LogProbability()</code> [1/2]	1235
39.255.3.3	<code>LogProbability()</code> [2/2]	1235
39.255.3.4	<code>Probabilities()</code> [1/2]	1236
39.255.3.5	<code>Probabilities()</code> [2/2]	1236
39.255.3.6	<code>Probability()</code> [1/2]	1236
39.255.3.7	<code>Probability()</code> [2/2]	1237
39.255.3.8	<code>Random()</code>	1237
39.255.3.9	<code>Serialize()</code>	1237
39.255.3.10	<code>Train()</code> [1/2]	1238
39.255.3.11	<code>Train()</code> [2/2]	1238
39.256	<code>GammaDistribution</code> Class Reference	1238
39.256.1	Detailed Description	1239
39.256.2	Constructor & Destructor Documentation	1240
39.256.2.1	<code>GammaDistribution()</code> [1/3]	1240
39.256.2.2	<code>GammaDistribution()</code> [2/3]	1240
39.256.2.3	<code>GammaDistribution()</code> [3/3]	1240
39.256.2.4	<code>~GammaDistribution()</code>	1241

39.256.3	Member Function Documentation	1241
39.256.3.1	Alpha() [1/2]	1241
39.256.3.2	Alpha() [2/2]	1241
39.256.3.3	Beta() [1/2]	1242
39.256.3.4	Beta() [2/2]	1242
39.256.3.5	Dimensionality()	1242
39.256.3.6	LogProbability() [1/2]	1242
39.256.3.7	LogProbability() [2/2]	1243
39.256.3.8	Probability() [1/2]	1243
39.256.3.9	Probability() [2/2]	1244
39.256.3.10	Random()	1244
39.256.3.11	Train() [1/3]	1244
39.256.3.12	Train() [2/3]	1245
39.256.3.13	Train() [3/3]	1245
39.257	GaussianDistribution Class Reference	1246
39.257.1	Detailed Description	1246
39.257.2	Constructor & Destructor Documentation	1247
39.257.2.1	GaussianDistribution() [1/3]	1247
39.257.2.2	GaussianDistribution() [2/3]	1247
39.257.2.3	GaussianDistribution() [3/3]	1247
39.257.3	Member Function Documentation	1247
39.257.3.1	Covariance() [1/3]	1248
39.257.3.2	Covariance() [2/3]	1248
39.257.3.3	Covariance() [3/3]	1248
39.257.3.4	Dimensionality()	1248
39.257.3.5	LogProbability() [1/2]	1248
39.257.3.6	LogProbability() [2/2]	1248
39.257.3.7	Mean() [1/2]	1249

39.257.3.8	Mean() [2/2]	1249
39.257.3.9	Probability() [1/2]	1249
39.257.3.10	Probability() [2/2]	1249
39.257.3.11	Random()	1250
39.257.3.12	Serialize()	1250
39.257.3.13	Train() [1/2]	1250
39.257.3.14	Train() [2/2]	1251
39.258	LaplaceDistribution Class Reference	1251
39.258.1	Detailed Description	1252
39.258.2	Constructor & Destructor Documentation	1252
39.258.2.1	LaplaceDistribution() [1/3]	1253
39.258.2.2	LaplaceDistribution() [2/3]	1253
39.258.2.3	LaplaceDistribution() [3/3]	1253
39.258.3	Member Function Documentation	1254
39.258.3.1	Dimensionality()	1254
39.258.3.2	Estimate() [1/2]	1254
39.258.3.3	Estimate() [2/2]	1254
39.258.3.4	LogProbability() [1/2]	1254
39.258.3.5	LogProbability() [2/2]	1255
39.258.3.6	Mean() [1/2]	1255
39.258.3.7	Mean() [2/2]	1255
39.258.3.8	Probability() [1/2]	1255
39.258.3.9	Probability() [2/2]	1256
39.258.3.10	Random()	1256
39.258.3.11	Scale() [1/2]	1257
39.258.3.12	Scale() [2/2]	1257
39.258.3.13	Serialize()	1257
39.259	RegressionDistribution Class Reference	1257

39.259.1	Detailed Description	1258
39.259.2	Constructor & Destructor Documentation	1259
39.259.2.1	RegressionDistribution() [1/3]	1259
39.259.2.2	RegressionDistribution() [2/3]	1259
39.259.2.3	RegressionDistribution() [3/3]	1259
39.259.3	Member Function Documentation	1260
39.259.3.1	Dimensionality()	1260
39.259.3.2	Err() [1/2]	1260
39.259.3.3	Err() [2/2]	1260
39.259.3.4	LogProbability()	1260
39.259.3.5	Parameters()	1261
39.259.3.6	Predict() [1/2]	1261
39.259.3.7	Predict() [2/2]	1261
39.259.3.8	Probability()	1262
39.259.3.9	Rf() [1/2]	1262
39.259.3.10	Rf() [2/2]	1262
39.259.3.11	Serialize()	1263
39.259.3.12	Train() [1/3]	1263
39.259.3.13	Train() [2/3]	1263
39.259.3.14	Train() [3/3]	1263
39.260	DTBRules< MetricType, TreeType > Class Template Reference	1264
39.260.1	Detailed Description	1265
39.260.2	Member Typedef Documentation	1265
39.260.2.1	TraverseAllInfoType	1265
39.260.3	Constructor & Destructor Documentation	1265
39.260.3.1	DTBRules()	1265
39.260.4	Member Function Documentation	1265
39.260.4.1	BaseCase()	1265

39.260.4.2BaseCases() [1/2]	1266
39.260.4.3BaseCases() [2/2]	1266
39.260.4.4Rescore() [1/2]	1266
39.260.4.5Rescore() [2/2]	1266
39.260.4.6Score() [1/2]	1267
39.260.4.7Score() [2/2]	1267
39.260.4.8Scores() [1/2]	1268
39.260.4.9Scores() [2/2]	1268
39.260.4.10TraversallInfo() [1/2]	1268
39.260.4.11TraversallInfo() [2/2]	1268
39.261DTBStat Class Reference	1269
39.261.1Detailed Description	1269
39.261.2Constructor & Destructor Documentation	1269
39.261.2.1DTBStat() [1/2]	1270
39.261.2.2DTBStat() [2/2]	1270
39.261.3Member Function Documentation	1270
39.261.3.1Bound() [1/2]	1270
39.261.3.2Bound() [2/2]	1271
39.261.3.3ComponentMembership() [1/2]	1271
39.261.3.4ComponentMembership() [2/2]	1271
39.261.3.5MaxNeighborDistance() [1/2]	1271
39.261.3.6MaxNeighborDistance() [2/2]	1271
39.261.3.7MinNeighborDistance() [1/2]	1272
39.261.3.8MinNeighborDistance() [2/2]	1272
39.262DualTreeBoruvka< MetricType, MatType, TreeType > Class Template Reference	1272
39.262.1Detailed Description	1273
39.262.2Member Typedef Documentation	1273
39.262.2.1Tree	1273

39.262.3	Constructor & Destructor Documentation	1274
39.262.3.1	DualTreeBoruvka() [1/2]	1274
39.262.3.2	DualTreeBoruvka() [2/2]	1274
39.262.3.3	~DualTreeBoruvka()	1275
39.262.4	Member Function Documentation	1275
39.262.4.1	ComputeMST()	1275
39.263	EdgePair Class Reference	1275
39.263.1	Detailed Description	1276
39.263.2	Constructor & Destructor Documentation	1276
39.263.2.1	EdgePair()	1276
39.263.3	Member Function Documentation	1276
39.263.3.1	Distance() [1/2]	1277
39.263.3.2	Distance() [2/2]	1277
39.263.3.3	Greater() [1/2]	1277
39.263.3.4	Greater() [2/2]	1277
39.263.3.5	Lesser() [1/2]	1277
39.263.3.6	Lesser() [2/2]	1278
39.264	UnionFind Class Reference	1278
39.264.1	Detailed Description	1278
39.264.2	Constructor & Destructor Documentation	1278
39.264.2.1	UnionFind()	1279
39.264.2.2	~UnionFind()	1279
39.264.3	Member Function Documentation	1279
39.264.3.1	Find()	1279
39.264.3.2	Union()	1280
39.265	FastMKS< KernelType, MatType, TreeType > Class Template Reference	1280
39.265.1	Detailed Description	1282
39.265.2	Member Typedef Documentation	1283

39.265.2.1Tree	1283
39.265.3Constructor & Destructor Documentation	1283
39.265.3.1FastMKS() [1/8]	1283
39.265.3.2FastMKS() [2/8]	1283
39.265.3.3FastMKS() [3/8]	1284
39.265.3.4FastMKS() [4/8]	1284
39.265.3.5FastMKS() [5/8]	1285
39.265.3.6FastMKS() [6/8]	1285
39.265.3.7FastMKS() [7/8]	1286
39.265.3.8FastMKS() [8/8]	1286
39.265.3.9~FastMKS()	1286
39.265.4Member Function Documentation	1286
39.265.4.1Metric() [1/2]	1286
39.265.4.2Metric() [2/2]	1286
39.265.4.3Naive() [1/2]	1287
39.265.4.4Naive() [2/2]	1287
39.265.4.5operator=()	1287
39.265.4.6Search() [1/3]	1287
39.265.4.7Search() [2/3]	1288
39.265.4.8Search() [3/3]	1288
39.265.4.9Serialize()	1289
39.265.4.10SingleMode() [1/2]	1289
39.265.4.11SingleMode() [2/2]	1289
39.265.4.12Train() [1/5]	1289
39.265.4.13Train() [2/5]	1290
39.265.4.14Train() [3/5]	1290
39.265.4.15Train() [4/5]	1290
39.265.4.16Train() [5/5]	1291

39.266.1	FastMKModel Class Reference	1291
39.266.1.1	Detailed Description	1292
39.266.1.2	Member Enumeration Documentation	1292
39.266.1.2.1	KernelTypes	1293
39.266.1.3	Constructor & Destructor Documentation	1294
39.266.1.3.1	FastMKModel() [1/3]	1294
39.266.1.3.2	FastMKModel() [2/3]	1294
39.266.1.3.3	FastMKModel() [3/3]	1294
39.266.1.3.4	~FastMKModel()	1295
39.266.1.4	Member Function Documentation	1295
39.266.1.4.1	BuildModel()	1295
39.266.1.4.2	KernelType() [1/2]	1295
39.266.1.4.3	KernelType() [2/2]	1295
39.266.1.4.4	Naive() [1/2]	1296
39.266.1.4.5	Naive() [2/2]	1296
39.266.1.4.6	operator=()	1296
39.266.1.4.7	Search() [1/2]	1296
39.266.1.4.8	Search() [2/2]	1297
39.266.1.4.9	Serialize()	1297
39.266.1.4.10	SingleMode() [1/2]	1297
39.266.1.4.11	SingleMode() [2/2]	1297
39.267	FastMKSRules< KernelType, TreeType > Class Template Reference	1298
39.267.1	Detailed Description	1298
39.267.2	Member Typedef Documentation	1299
39.267.2.1	TraversallInfoType	1299
39.267.3	Constructor & Destructor Documentation	1299
39.267.3.1	FastMKSRules()	1299
39.267.4	Member Function Documentation	1300

39.267.4.1BaseCase()	1300
39.267.4.2BaseCases() [1/2]	1300
39.267.4.3BaseCases() [2/2]	1300
39.267.4.4GetResults()	1300
39.267.4.5Rescore() [1/2]	1301
39.267.4.6Rescore() [2/2]	1301
39.267.4.7Score() [1/2]	1301
39.267.4.8Score() [2/2]	1302
39.267.4.9Scores() [1/2]	1302
39.267.4.10Scores() [2/2]	1302
39.267.4.11TraversallInfo() [1/2]	1303
39.267.4.12TraversallInfo() [2/2]	1303
39.268FastMKStat Class Reference	1303
39.268.1Detailed Description	1304
39.268.2Constructor & Destructor Documentation	1304
39.268.2.1FastMKStat() [1/2]	1304
39.268.2.2FastMKStat() [2/2]	1304
39.268.3Member Function Documentation	1304
39.268.3.1Bound() [1/2]	1305
39.268.3.2Bound() [2/2]	1305
39.268.3.3LastKernel() [1/2]	1305
39.268.3.4LastKernel() [2/2]	1305
39.268.3.5LastKernelNode() [1/2]	1305
39.268.3.6LastKernelNode() [2/2]	1306
39.268.3.7SelfKernel() [1/2]	1306
39.268.3.8SelfKernel() [2/2]	1306
39.268.3.9Serialize()	1306
39.269DiagonalConstraint Class Reference	1306

39.269.1	Detailed Description	1307
39.269.2	Member Function Documentation	1307
39.269.2.1	ApplyConstraint() [1/2]	1307
39.269.2.2	ApplyConstraint() [2/2]	1307
39.269.2.3	Serialize()	1308
39.270	DiagonalGMM Class Reference	1308
39.270.1	Detailed Description	1309
39.270.2	Constructor & Destructor Documentation	1310
39.270.2.1	DiagonalGMM() [1/4]	1310
39.270.2.2	DiagonalGMM() [2/4]	1310
39.270.2.3	DiagonalGMM() [3/4]	1311
39.270.2.4	DiagonalGMM() [4/4]	1311
39.270.3	Member Function Documentation	1311
39.270.3.1	Classify()	1311
39.270.3.2	Component() [1/2]	1312
39.270.3.3	Component() [2/2]	1312
39.270.3.4	Dimensionality()	1312
39.270.3.5	Gaussians()	1313
39.270.3.6	LogProbability() [1/2]	1313
39.270.3.7	LogProbability() [2/2]	1313
39.270.3.8	operator=()	1313
39.270.3.9	Probability() [1/2]	1314
39.270.3.10	Probability() [2/2]	1314
39.270.3.11	Random()	1314
39.270.3.12	Serialize()	1315
39.270.3.13	Train() [1/2]	1315
39.270.3.14	Train() [2/2]	1315
39.270.3.15	Weights() [1/2]	1316

39.270.3.1	Weights() [2/2]	1316
39.271	EigenvalueRatioConstraint Class Reference	1317
39.271.1	Detailed Description	1317
39.271.2	Constructor & Destructor Documentation	1317
39.271.2.1	EigenvalueRatioConstraint()	1317
39.271.3	Member Function Documentation	1318
39.271.3.1	ApplyConstraint() [1/2]	1318
39.271.3.2	ApplyConstraint() [2/2]	1318
39.271.3.3	Serialize()	1318
39.272	EMFit< InitialClusteringType, CovarianceConstraintPolicy, Distribution > Class Template Reference	1318
39.272.1	Detailed Description	1319
39.272.2	Constructor & Destructor Documentation	1320
39.272.2.1	EMFit()	1320
39.272.3	Member Function Documentation	1320
39.272.3.1	Clusterer() [1/2]	1320
39.272.3.2	Clusterer() [2/2]	1321
39.272.3.3	Constraint() [1/2]	1321
39.272.3.4	Constraint() [2/2]	1321
39.272.3.5	Estimate() [1/2]	1321
39.272.3.6	Estimate() [2/2]	1322
39.272.3.7	MaxIterations() [1/2]	1322
39.272.3.8	MaxIterations() [2/2]	1322
39.272.3.9	Serialize()	1323
39.272.3.10	Tolerance() [1/2]	1323
39.272.3.11	Tolerance() [2/2]	1323
39.273	GMM Class Reference	1323
39.273.1	Detailed Description	1325
39.273.2	Constructor & Destructor Documentation	1325

39.273.2.1	GMM() [1/4]	1326
39.273.2.2	GMM() [2/4]	1326
39.273.2.3	GMM() [3/4]	1326
39.273.2.4	GMM() [4/4]	1327
39.273.3	Member Function Documentation	1327
39.273.3.1	Classify()	1327
39.273.3.2	Component() [1/2]	1327
39.273.3.3	Component() [2/2]	1328
39.273.3.4	Dimensionality()	1328
39.273.3.5	Gaussians()	1328
39.273.3.6	LogProbability() [1/2]	1328
39.273.3.7	LogProbability() [2/2]	1329
39.273.3.8	operator=()	1329
39.273.3.9	Probability() [1/2]	1329
39.273.3.10	Probability() [2/2]	1330
39.273.3.11	Random()	1330
39.273.3.12	Serialize()	1330
39.273.3.13	Train() [1/2]	1331
39.273.3.14	Train() [2/2]	1331
39.273.3.15	Weights() [1/2]	1332
39.273.3.16	Weights() [2/2]	1332
39.274	NoConstraint Class Reference	1332
39.274.1	Detailed Description	1333
39.274.2	Member Function Documentation	1333
39.274.2.1	ApplyConstraint()	1333
39.274.2.2	Serialize()	1333
39.275	PositiveDefiniteConstraint Class Reference	1334
39.275.1	Detailed Description	1334

39.275.2	Member Function Documentation	1334
39.275.2.1	ApplyConstraint() [1/2]	1334
39.275.2.2	ApplyConstraint() [2/2]	1335
39.275.2.3	Serialize()	1335
39.276	HMM< Distribution > Class Template Reference	1335
39.276.1	Detailed Description	1337
39.276.2	Constructor & Destructor Documentation	1338
39.276.2.1	HMM() [1/2]	1338
39.276.2.2	HMM() [2/2]	1339
39.276.3	Member Function Documentation	1339
39.276.3.1	Backward()	1339
39.276.3.2	Dimensionality() [1/2]	1341
39.276.3.3	Dimensionality() [2/2]	1341
39.276.3.4	Emission() [1/2]	1341
39.276.3.5	Emission() [2/2]	1341
39.276.3.6	Estimate() [1/2]	1342
39.276.3.7	Estimate() [2/2]	1342
39.276.3.8	Filter()	1343
39.276.3.9	Forward()	1343
39.276.3.10	Generate()	1344
39.276.3.11	Initial() [1/2]	1344
39.276.3.12	Initial() [2/2]	1344
39.276.3.13	LogEstimate()	1345
39.276.3.14	LogLikelihood()	1346
39.276.3.15	Predict()	1346
39.276.3.16	Serialize()	1347
39.276.3.17	Smooth()	1347
39.276.3.18	Tolerance() [1/2]	1347

39.276.3.1Tolerance() [2/2]	1348
39.276.3.2Train() [1/2]	1348
39.276.3.2Train() [2/2]	1348
39.276.3.2Transition() [1/2]	1349
39.276.3.2Transition() [2/2]	1349
39.276.4Member Data Documentation	1349
39.276.4.1emission	1350
39.276.4.2transition	1350
39.277HMMModel Class Reference	1350
39.277.1Detailed Description	1351
39.277.2Constructor & Destructor Documentation	1351
39.277.2.1HMMModel() [1/3]	1351
39.277.2.2HMMModel() [2/3]	1351
39.277.2.3HMMModel() [3/3]	1351
39.277.2.4HMMModel()	1352
39.277.3Member Function Documentation	1352
39.277.3.1DiagGMMHMM()	1352
39.277.3.2DiscreteHMM()	1352
39.277.3.3GaussianHMM()	1352
39.277.3.4GMMHMM()	1353
39.277.3.5operator=()	1353
39.277.3.6PerformAction()	1353
39.277.3.7serialize()	1353
39.277.3.8Type()	1354
39.278HMMRegression Class Reference	1354
39.278.1Detailed Description	1355
39.278.2Constructor & Destructor Documentation	1356
39.278.2.1HMMRegression() [1/2]	1356

39.278.2.2HMMRegression() [2/2]	1356
39.278.3Member Function Documentation	1357
39.278.3.1Estimate() [1/2]	1357
39.278.3.2Estimate() [2/2]	1358
39.278.3.3Filter()	1358
39.278.3.4LogLikelihood()	1359
39.278.3.5Predict()	1359
39.278.3.6Smooth()	1360
39.278.3.7Train() [1/2]	1360
39.278.3.8Train() [2/2]	1361
39.279CVFunction< CVType, MLAlgorithm, TotalArgs, BoundArgs > Class Template Reference	1361
39.279.1Detailed Description	1362
39.279.2Constructor & Destructor Documentation	1362
39.279.2.1CVFunction()	1362
39.279.3Member Function Documentation	1364
39.279.3.1BestModel()	1364
39.279.3.2Evaluate()	1364
39.279.3.3Gradient()	1365
39.280DeduceHyperParameterTypes< Args > Struct Template Reference	1365
39.280.1Detailed Description	1365
39.281DeduceHyperParameterTypes< Args >::ResultHolder< HPTypes > Struct Template Reference	1366
39.281.1Detailed Description	1366
39.281.2Member Typedef Documentation	1366
39.281.2.1TupleType	1366
39.282DeduceHyperParameterTypes< PreFixedArg< T >, Args... > Struct Template Reference	1366
39.282.1Detailed Description	1367
39.282.2Member Typedef Documentation	1367
39.282.2.1TupleType	1367

39.282	DeduceHyperParameterTypes< PreFixedArg< T >, Args... >::ResultHolder< HPTypes > Struct Template Reference	1367
39.283.1	Detailed Description	1367
39.283.2	Member Typedef Documentation	1368
39.283.2.1	TupleType	1368
39.284	DeduceHyperParameterTypes< T, Args... > Struct Template Reference	1368
39.284.1	Detailed Description	1368
39.284.2	Member Typedef Documentation	1369
39.284.2.1	TupleType	1369
39.285	DeduceHyperParameterTypes< T, Args... >::IsCollectionType< Type > Struct Template Reference	1369
39.285.1	Detailed Description	1369
39.285.2	Member Typedef Documentation	1370
39.285.2.1	No	1370
39.285.2.2	Yes	1370
39.285.3	Member Function Documentation	1370
39.285.3.1	Check() [1/2]	1370
39.285.3.2	Check() [2/2]	1370
39.285.4	Member Data Documentation	1370
39.285.4.1	value	1371
39.286	DeduceHyperParameterTypes< T, Args... >::ResultHolder< HPTypes > Struct Template Reference	1371
39.286.1	Detailed Description	1371
39.286.2	Member Typedef Documentation	1371
39.286.2.1	TupleType	1371
39.287	DeduceHyperParameterTypes< T, Args... >::ResultHPTType< ArgumentType, IsArithmetic > Struct Template Reference	1372
39.287.1	Detailed Description	1372
39.288	DeduceHyperParameterTypes< T, Args... >::ResultHPTType< ArithmeticType, true > Struct Template Reference	1372
39.288.1	Detailed Description	1372

39.288.1	Member Typedef Documentation	1372
39.288.2.1	Type	1373
39.289	DeduceHyperParameterTypes< T, Args... >::ResultHPType< CollectionType, false > Struct Template Reference	1373
39.289.1	Detailed Description	1373
39.289.2	Member Typedef Documentation	1373
39.289.2.1	Type	1373
39.290	FixedArg< T, I > Struct Template Reference	1373
39.290.1	Detailed Description	1374
39.290.2	Member Data Documentation	1374
39.290.2.1	index	1374
39.290.2.2	value	1375
39.291	HyperParameterTuner< MLAlgorithm, Metric, CV, OptimizerType, MatType, PredictionsType, WeightsType > Class Template Reference	1375
39.291.1	Detailed Description	1376
39.291.2	Constructor & Destructor Documentation	1377
39.291.2.1	HyperParameterTuner()	1377
39.291.3	Member Function Documentation	1377
39.291.3.1	BestModel() [1/2]	1377
39.291.3.2	BestModel() [2/2]	1377
39.291.3.3	BestObjective()	1378
39.291.3.4	MinDelta() [1/2]	1378
39.291.3.5	MinDelta() [2/2]	1378
39.291.3.6	Optimize()	1379
39.291.3.7	Optimizer()	1379
39.291.3.8	RelativeDelta() [1/2]	1379
39.291.3.9	RelativeDelta() [2/2]	1380
39.292	PreFixedArg< T > Class Template Reference	1380
39.292.1	Detailed Description	1380

39.292.2	Member Data Documentation	1380
39.292.2.1	value	1381
39.293	PreFixedArg< T > Struct Template Reference	1381
39.293.1	Detailed Description	1381
39.293.2	Member Typedef Documentation	1381
39.293.2.1	Type	1382
39.293.3	Member Data Documentation	1382
39.293.3.1	value	1382
39.294	PreFixedArg< T & > Struct Template Reference	1382
39.294.1	Detailed Description	1382
39.294.2	Member Typedef Documentation	1383
39.294.2.1	Type	1383
39.294.3	Member Data Documentation	1383
39.294.3.1	value	1383
39.295	DeleteVisitor Class Reference	1383
39.295.1	Detailed Description	1384
39.295.2	Member Function Documentation	1384
39.295.2.1	operator>()	1384
39.296	DualBiKDE Class Reference	1384
39.296.1	Detailed Description	1385
39.296.2	Member Typedef Documentation	1385
39.296.2.1	KDETypeT	1385
39.296.3	Constructor & Destructor Documentation	1385
39.296.3.1	DualBiKDE()	1385
39.296.4	Member Function Documentation	1385
39.296.4.1	operator>()	1386
39.297	DualMonoKDE Class Reference	1386
39.297.1	Detailed Description	1386

39.297.2	Member Typedef Documentation	1387
39.297.2.1	KDETypeT	1387
39.297.3	Constructor & Destructor Documentation	1387
39.297.3.1	DualMonoKDE()	1387
39.297.4	Member Function Documentation	1387
39.297.4.1	operator>()	1387
39.298	KDE< KernelType, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTreeTraversalType > Class Template Reference	1387
39.298.1	Detailed Description	1389
39.298.2	Member Typedef Documentation	1389
39.298.2.1	Tree	1389
39.298.3	Constructor & Destructor Documentation	1390
39.298.3.1	KDE() [1/3]	1390
39.298.3.2	KDE() [2/3]	1390
39.298.3.3	KDE() [3/3]	1390
39.298.3.4	~KDE()	1391
39.298.4	Member Function Documentation	1391
39.298.4.1	AbsoluteError() [1/2]	1391
39.298.4.2	AbsoluteError() [2/2]	1391
39.298.4.3	Evaluate() [1/3]	1391
39.298.4.4	Evaluate() [2/3]	1392
39.298.4.5	Evaluate() [3/3]	1392
39.298.4.6	IsTrained()	1393
39.298.4.7	Kernel() [1/2]	1393
39.298.4.8	Kernel() [2/2]	1393
39.298.4.9	Metric() [1/2]	1393
39.298.4.10	Metric() [2/2]	1394
39.298.4.11	Mode() [1/2]	1394

39.298.4.12	<code>Model()</code> [2/2]	1394
39.298.4.13	<code>operator=()</code>	1394
39.298.4.14	<code>OwnsReferenceTree()</code>	1395
39.298.4.15	<code>ReferenceTree()</code>	1395
39.298.4.16	<code>RelativeError()</code> [1/2]	1395
39.298.4.17	<code>RelativeError()</code> [2/2]	1395
39.298.4.18	<code>Serialize()</code>	1396
39.298.4.19	<code>Train()</code> [1/2]	1396
39.298.4.20	<code>Train()</code> [2/2]	1396
39.298	KDEModel Class Reference	1397
39.299.1	Detailed Description	1398
39.299.2	Member Enumeration Documentation	1398
39.299.2.1	KernelTypes	1398
39.299.2.2	TreeTypes	1399
39.299.3	Constructor & Destructor Documentation	1399
39.299.3.1	<code>KDEModel()</code> [1/3]	1399
39.299.3.2	<code>KDEModel()</code> [2/3]	1400
39.299.3.3	<code>KDEModel()</code> [3/3]	1400
39.299.3.4	<code>~KDEModel()</code>	1400
39.299.4	Member Function Documentation	1400
39.299.4.1	<code>AbsoluteError()</code> [1/2]	1400
39.299.4.2	<code>AbsoluteError()</code> [2/2]	1400
39.299.4.3	<code>Bandwidth()</code> [1/2]	1401
39.299.4.4	<code>Bandwidth()</code> [2/2]	1401
39.299.4.5	<code>BuildModel()</code>	1401
39.299.4.6	<code>Evaluate()</code> [1/2]	1401
39.299.4.7	<code>Evaluate()</code> [2/2]	1402
39.299.4.8	<code>KernelType()</code> [1/2]	1402

39.299.4.9	K KernelType() [2/2]	1402
39.299.4.10	M Mode() [1/2]	1403
39.299.4.11	M Mode() [2/2]	1403
39.299.4.12	O perator=()	1403
39.299.4.13	R RelativeError() [1/2]	1403
39.299.4.14	R RelativeError() [2/2]	1403
39.299.4.15	S erialize()	1404
39.299.4.16	T reeType() [1/2]	1404
39.299.4.17	T reeType() [2/2]	1404
39.300	K DERules< MetricType, KernelType, TreeType > Class Template Reference	1404
39.300.1	D etailed Description	1405
39.300.2	M ember Typedef Documentation	1405
39.300.2.1	T raversalInfoType	1405
39.300.3	C onstructor & Destructor Documentation	1406
39.300.3.1	K DERules()	1406
39.300.4	M ember Function Documentation	1406
39.300.4.1	B aseCase()	1406
39.300.4.2	B aseCases()	1407
39.300.4.3	R escore() [1/2]	1407
39.300.4.4	R escore() [2/2]	1407
39.300.4.5	S core() [1/2]	1407
39.300.4.6	S core() [2/2]	1407
39.300.4.7	S cores()	1408
39.300.4.8	T raversalInfo() [1/2]	1408
39.300.4.9	T raversalInfo() [2/2]	1408
39.301	K DEStat Class Reference	1408
39.301.1	D etailed Description	1409
39.301.2	C onstructor & Destructor Documentation	1409

39.301.2.1	KDEStat() [1/2]	1409
39.301.2.2	KDEStat() [2/2]	1409
39.301.3	Member Function Documentation	1409
39.301.3.1	Centroid()	1409
39.301.3.2	Serialize()	1410
39.301.3.3	SetCentroid()	1410
39.301.3.4	ValidCentroid()	1410
39.302	KernelNormalizer Class Reference	1410
39.302.1	Detailed Description	1411
39.302.2	Member Function Documentation	1411
39.302.2.1	ApplyNormalizer() [1/2]	1411
39.302.2.2	ApplyNormalizer() [2/2]	1411
39.303	ModeVisitor Class Reference	1412
39.303.1	Detailed Description	1412
39.303.2	Member Function Documentation	1412
39.303.2.1	operator>()	1412
39.304	TrainVisitor Class Reference	1413
39.304.1	Detailed Description	1413
39.304.2	Constructor & Destructor Documentation	1413
39.304.2.1	TrainVisitor()	1413
39.304.3	Member Function Documentation	1413
39.304.3.1	operator>()	1414
39.305	CauchyKernel Class Reference	1414
39.305.1	Detailed Description	1414
39.305.2	Constructor & Destructor Documentation	1415
39.305.2.1	CauchyKernel()	1415
39.305.3	Member Function Documentation	1415
39.305.3.1	Evaluate()	1415

39.305.3.2	Serialize()	1416
39.306	CosineDistance Class Reference	1416
39.306.1	Detailed Description	1416
39.306.2	Member Function Documentation	1416
39.306.2.1	Evaluate()	1416
39.306.2.2	Serialize()	1417
39.307	EpanechnikovKernel Class Reference	1417
39.307.1	Detailed Description	1418
39.307.2	Constructor & Destructor Documentation	1418
39.307.2.1	EpanechnikovKernel()	1418
39.307.3	Member Function Documentation	1418
39.307.3.1	ConvolutionIntegral()	1418
39.307.3.2	Evaluate() [1/2]	1419
39.307.3.3	Evaluate() [2/2]	1419
39.307.3.4	Gradient()	1420
39.307.3.5	GradientForSquaredDistance()	1420
39.307.3.6	Normalizer()	1420
39.307.3.7	serialize()	1420
39.308	ExampleKernel Class Reference	1421
39.308.1	Detailed Description	1421
39.308.2	Constructor & Destructor Documentation	1422
39.308.2.1	ExampleKernel()	1422
39.308.3	Member Function Documentation	1422
39.308.3.1	ConvolutionIntegral()	1422
39.308.3.2	Evaluate()	1423
39.308.3.3	Normalizer()	1423
39.308.3.4	serialize()	1424
39.309	GaussianKernel Class Reference	1424

39.309.1	Detailed Description	1425
39.309.2	Constructor & Destructor Documentation	1425
39.309.2.1	GaussianKernel() [1/2]	1425
39.309.2.2	GaussianKernel() [2/2]	1425
39.309.3	Member Function Documentation	1426
39.309.3.1	Bandwidth() [1/2]	1426
39.309.3.2	Bandwidth() [2/2]	1426
39.309.3.3	ConvolutionIntegral()	1426
39.309.3.4	Evaluate() [1/2]	1427
39.309.3.5	Evaluate() [2/2]	1427
39.309.3.6	Gamma()	1428
39.309.3.7	Gradient()	1428
39.309.3.8	GradientForSquaredDistance()	1428
39.309.3.9	Normalizer()	1429
39.309.3.10	Serialize()	1429
39.310	HyperbolicTangentKernel Class Reference	1430
39.310.1	Detailed Description	1430
39.310.2	Constructor & Destructor Documentation	1430
39.310.2.1	HyperbolicTangentKernel() [1/2]	1431
39.310.2.2	HyperbolicTangentKernel() [2/2]	1431
39.310.3	Member Function Documentation	1431
39.310.3.1	Evaluate()	1431
39.310.3.2	Offset() [1/2]	1432
39.310.3.3	Offset() [2/2]	1432
39.310.3.4	Scale() [1/2]	1432
39.310.3.5	Scale() [2/2]	1433
39.310.3.6	Serialize()	1433
39.311	KernelTraits< KernelType > Class Template Reference	1433

39.311.1	Detailed Description	1433
39.311.2	Member Data Documentation	1434
39.311.2.1	IsNormalized	1434
39.311.2.2	UsesSquaredDistance	1434
39.312	KernelTraits< CauchyKernel > Class Template Reference	1434
39.312.1	Detailed Description	1434
39.312.2	Member Data Documentation	1435
39.312.2.1	IsNormalized	1435
39.313	KernelTraits< CosineDistance > Class Template Reference	1435
39.313.1	Detailed Description	1435
39.313.2	Member Data Documentation	1435
39.313.2.1	IsNormalized	1436
39.313.2.2	UsesSquaredDistance	1436
39.314	KernelTraits< EpanechnikovKernel > Class Template Reference	1436
39.314.1	Detailed Description	1436
39.314.2	Member Data Documentation	1437
39.314.2.1	IsNormalized	1437
39.314.2.2	UsesSquaredDistance	1437
39.315	KernelTraits< GaussianKernel > Class Template Reference	1437
39.315.1	Detailed Description	1437
39.315.2	Member Data Documentation	1438
39.315.2.1	IsNormalized	1438
39.315.2.2	UsesSquaredDistance	1438
39.316	KernelTraits< LaplacianKernel > Class Template Reference	1438
39.316.1	Detailed Description	1438
39.316.2	Member Data Documentation	1439
39.316.2.1	IsNormalized	1439
39.316.2.2	UsesSquaredDistance	1439

39.317	KernelTraits < SphericalKernel > Class Template Reference	1439
39.317.1	Detailed Description	1439
39.317.2	Member Data Documentation	1440
39.317.2.1	IsNormalized	1440
39.317.2.2	UsesSquaredDistance	1440
39.318	KernelTraits < TriangularKernel > Class Template Reference	1440
39.318.1	Detailed Description	1440
39.318.2	Member Data Documentation	1441
39.318.2.1	IsNormalized	1441
39.318.2.2	UsesSquaredDistance	1441
39.319	MeansSelection < ClusteringType, maxIterations > Class Template Reference	1441
39.319.1	Detailed Description	1441
39.319.2	Member Function Documentation	1442
39.319.2.1	Select()	1442
39.320	LaplacianKernel Class Reference	1442
39.320.1	Detailed Description	1443
39.320.2	Constructor & Destructor Documentation	1443
39.320.2.1	LaplacianKernel() [1/2]	1443
39.320.2.2	LaplacianKernel() [2/2]	1443
39.320.3	Member Function Documentation	1444
39.320.3.1	Bandwidth() [1/2]	1444
39.320.3.2	Bandwidth() [2/2]	1444
39.320.3.3	Evaluate() [1/2]	1444
39.320.3.4	Evaluate() [2/2]	1445
39.320.3.5	Gradient()	1445
39.320.3.6	Serialize()	1446
39.321	LinearKernel Class Reference	1446
39.321.1	Detailed Description	1447

39.321.2	Constructor & Destructor Documentation	1447
39.321.2.1	LinearKernel()	1447
39.321.3	Member Function Documentation	1447
39.321.3.1	Evaluate()	1447
39.321.3.2	Serialize()	1448
39.322	NystroemMethod< KernelType, PointSelectionPolicy > Class Template Reference	1448
39.322.1	Detailed Description	1448
39.322.2	Constructor & Destructor Documentation	1449
39.322.2.1	NystroemMethod()	1449
39.322.3	Member Function Documentation	1449
39.322.3.1	Apply()	1449
39.322.3.2	GetKernelMatrix() [1/2]	1449
39.322.3.3	GetKernelMatrix() [2/2]	1450
39.323	OrderedSelection Class Reference	1450
39.323.1	Detailed Description	1450
39.323.2	Member Function Documentation	1451
39.323.2.1	Select()	1451
39.324	PolynomialKernel Class Reference	1451
39.324.1	Detailed Description	1452
39.324.2	Constructor & Destructor Documentation	1452
39.324.2.1	PolynomialKernel()	1452
39.324.3	Member Function Documentation	1452
39.324.3.1	Degree() [1/2]	1452
39.324.3.2	Degree() [2/2]	1453
39.324.3.3	Evaluate()	1453
39.324.3.4	Offset() [1/2]	1453
39.324.3.5	Offset() [2/2]	1454
39.324.3.6	Serialize()	1454

39.325	PSpectrumStringKernel Class Reference	1454
39.325.1	Detailed Description	1455
39.325.2	Constructor & Destructor Documentation	1455
39.325.2.1	PSpectrumStringKernel()	1455
39.325.3	Member Function Documentation	1456
39.325.3.1	Counts() [1/2]	1456
39.325.3.2	Counts() [2/2]	1456
39.325.3.3	Evaluate()	1456
39.325.3.4	P() [1/2]	1456
39.325.3.5	P() [2/2]	1457
39.326	RandomSelection Class Reference	1457
39.326.1	Detailed Description	1457
39.326.2	Member Function Documentation	1457
39.326.2.1	Select()	1457
39.327	SphericalKernel Class Reference	1458
39.327.1	Detailed Description	1458
39.327.2	Constructor & Destructor Documentation	1459
39.327.2.1	SphericalKernel()	1459
39.327.3	Member Function Documentation	1459
39.327.3.1	ConvolutionIntegral()	1459
39.327.3.2	Evaluate() [1/2]	1460
39.327.3.3	Evaluate() [2/2]	1460
39.327.3.4	Gradient()	1460
39.327.3.5	Normalizer()	1461
39.327.3.6	Serialize()	1461
39.328	TriangularKernel Class Reference	1461
39.328.1	Detailed Description	1462
39.328.2	Constructor & Destructor Documentation	1462

39.328.2.1TriangularKernel()	1462
39.328.3Member Function Documentation	1462
39.328.3.1Bandwidth() [1/2]	1462
39.328.3.2Bandwidth() [2/2]	1463
39.328.3.3Evaluate() [1/2]	1463
39.328.3.4Evaluate() [2/2]	1463
39.328.3.5Gradient()	1464
39.328.3.6Serialize()	1464
39.329AllowEmptyClusters Class Reference	1464
39.329.1Detailed Description	1465
39.329.2Constructor & Destructor Documentation	1465
39.329.2.1AllowEmptyClusters()	1465
39.329.3Member Function Documentation	1465
39.329.3.1EmptyCluster()	1465
39.329.3.2Serialize()	1466
39.330DualTreeKMeans< MetricType, MatType, TreeType > Class Template Reference	1466
39.330.1Detailed Description	1467
39.330.2Member Typedef Documentation	1467
39.330.2.1INNSTreeType	1467
39.330.2.2Tree	1468
39.330.3Constructor & Destructor Documentation	1468
39.330.3.1DualTreeKMeans()	1468
39.330.3.2~DualTreeKMeans()	1468
39.330.4Member Function Documentation	1468
39.330.4.1DistanceCalculations() [1/2]	1468
39.330.4.2DistanceCalculations() [2/2]	1469
39.330.4.3Iterate()	1469
39.331DualTreeKMeansRules< MetricType, TreeType > Class Template Reference	1469

39.331.1Detailed Description	1470
39.331.2Member Typedef Documentation	1470
39.331.2.1TraversallInfoType	1470
39.331.3Constructor & Destructor Documentation	1470
39.331.3.1DualTreeKMeansRules()	1470
39.331.4Member Function Documentation	1471
39.331.4.1BaseCase()	1471
39.331.4.2BaseCases() [1/2]	1471
39.331.4.3BaseCases() [2/2]	1471
39.331.4.4Rescore() [1/2]	1471
39.331.4.5Rescore() [2/2]	1471
39.331.4.6Score() [1/2]	1472
39.331.4.7Score() [2/2]	1472
39.331.4.8Scores() [1/2]	1472
39.331.4.9Scores() [2/2]	1472
39.331.4.10TraversallInfo() [1/2]	1472
39.331.4.11TraversallInfo() [2/2]	1472
39.332DualTreeKMeansStatistic Class Reference	1473
39.332.1Detailed Description	1474
39.332.2Constructor & Destructor Documentation	1474
39.332.2.1DualTreeKMeansStatistic() [1/2]	1474
39.332.2.2DualTreeKMeansStatistic() [2/2]	1474
39.332.3Member Function Documentation	1474
39.332.3.1Centroid() [1/2]	1474
39.332.3.2Centroid() [2/2]	1474
39.332.3.3LowerBound() [1/2]	1475
39.332.3.4LowerBound() [2/2]	1475
39.332.3.5NumTrueChildren()	1475

39.332.3.6Owner() [1/2]	1475
39.332.3.7Owner() [2/2]	1475
39.332.3.8Pruned() [1/2]	1475
39.332.3.9Pruned() [2/2]	1476
39.332.3.10StaticLowerBoundMovement() [1/2]	1476
39.332.3.11StaticLowerBoundMovement() [2/2]	1476
39.332.3.12StaticPruned() [1/2]	1476
39.332.3.13StaticPruned() [2/2]	1476
39.332.3.14StaticUpperBoundMovement() [1/2]	1476
39.332.3.15StaticUpperBoundMovement() [2/2]	1477
39.332.3.16TrueChild() [1/2]	1477
39.332.3.17TrueChild() [2/2]	1477
39.332.3.18TrueParent() [1/2]	1477
39.332.3.19TrueParent() [2/2]	1477
39.332.3.20UpperBound() [1/2]	1478
39.332.3.21UpperBound() [2/2]	1478
39.333ElkanKMeans< MetricType, MatType > Class Template Reference	1478
39.333.1Detailed Description	1478
39.333.2Constructor & Destructor Documentation	1478
39.333.2.1ElkanKMeans()	1479
39.333.3Member Function Documentation	1479
39.333.3.1DistanceCalculations()	1479
39.333.3.2Iterate()	1479
39.334HamerlyKMeans< MetricType, MatType > Class Template Reference	1479
39.334.1Detailed Description	1480
39.334.2Constructor & Destructor Documentation	1480
39.334.2.1HamerlyKMeans()	1480
39.334.3Member Function Documentation	1480

39.334.3.1DistanceCalculations()	1480
39.334.3.2Iterate()	1480
39.335KillEmptyClusters Class Reference	1481
39.335.1Detailed Description	1481
39.335.2Constructor & Destructor Documentation	1481
39.335.2.1KillEmptyClusters()	1482
39.335.3Member Function Documentation	1482
39.335.3.1EmptyCluster()	1482
39.335.3.2Serialize()	1483
39.336Means< MetricType, InitialPartitionPolicy, EmptyClusterPolicy, LloydStepType, MatType > Class Template Reference	1483
39.336.1Detailed Description	1484
39.336.2Constructor & Destructor Documentation	1485
39.336.2.1KMeans()	1485
39.336.3Member Function Documentation	1485
39.336.3.1Cluster() [1/3]	1485
39.336.3.2Cluster() [2/3]	1486
39.336.3.3Cluster() [3/3]	1486
39.336.3.4EmptyClusterAction() [1/2]	1487
39.336.3.5EmptyClusterAction() [2/2]	1487
39.336.3.6MaxIterations() [1/2]	1488
39.336.3.7MaxIterations() [2/2]	1488
39.336.3.8Metric() [1/2]	1488
39.336.3.9Metric() [2/2]	1488
39.336.3.10Partitioner() [1/2]	1488
39.336.3.11Partitioner() [2/2]	1489
39.336.3.12Serialize()	1489
39.337MaxVarianceNewCluster Class Reference	1489

39.337.1	Detailed Description	1490
39.337.2	Constructor & Destructor Documentation	1490
39.337.2.1	MaxVarianceNewCluster()	1490
39.337.3	Member Function Documentation	1490
39.337.3.1	EmptyCluster()	1490
39.337.3.2	Serialize()	1491
39.338	NaiveKMeans< MetricType, MatType > Class Template Reference	1491
39.338.1	Detailed Description	1491
39.338.2	Constructor & Destructor Documentation	1492
39.338.2.1	NaiveKMeans()	1492
39.338.3	Member Function Documentation	1492
39.338.3.1	DistanceCalculations()	1492
39.338.3.2	Iterate()	1492
39.339	PellegMooreKMeans< MetricType, MatType > Class Template Reference	1493
39.339.1	Detailed Description	1493
39.339.2	Member Typedef Documentation	1494
39.339.2.1	TreeType	1494
39.339.3	Constructor & Destructor Documentation	1494
39.339.3.1	PellegMooreKMeans()	1494
39.339.3.2	~PellegMooreKMeans()	1494
39.339.4	Member Function Documentation	1494
39.339.4.1	DistanceCalculations() [1/2]	1494
39.339.4.2	DistanceCalculations() [2/2]	1495
39.339.4.3	Iterate()	1495
39.340	PellegMooreKMeansRules< MetricType, TreeType > Class Template Reference	1495
39.340.1	Detailed Description	1496
39.340.2	Constructor & Destructor Documentation	1496
39.340.2.1	PellegMooreKMeansRules()	1496

39.340.3	Member Function Documentation	1496
39.340.3.1	BaseCase()	1497
39.340.3.2	DistanceCalculations() [1/2]	1497
39.340.3.3	DistanceCalculations() [2/2]	1497
39.340.3.4	Rescore()	1497
39.340.3.5	Score()	1498
39.341	PellegMooreKMeansStatistic Class Reference	1498
39.341.1	Detailed Description	1499
39.341.2	Constructor & Destructor Documentation	1499
39.341.2.1	PellegMooreKMeansStatistic() [1/2]	1499
39.341.2.2	PellegMooreKMeansStatistic() [2/2]	1499
39.341.3	Member Function Documentation	1499
39.341.3.1	Blacklist() [1/2]	1500
39.341.3.2	Blacklist() [2/2]	1500
39.341.3.3	Centroid() [1/2]	1500
39.341.3.4	Centroid() [2/2]	1500
39.342	RandomPartition Class Reference	1500
39.342.1	Detailed Description	1501
39.342.2	Constructor & Destructor Documentation	1501
39.342.2.1	RandomPartition()	1501
39.342.3	Member Function Documentation	1501
39.342.3.1	Cluster()	1501
39.342.3.2	Serialize()	1502
39.343	RefinedStart Class Reference	1502
39.343.1	Detailed Description	1503
39.343.2	Constructor & Destructor Documentation	1503
39.343.2.1	RefinedStart()	1503
39.343.3	Member Function Documentation	1503

39.343.3.1	Cluster() [1/2]	1504
39.343.3.2	Cluster() [2/2]	1504
39.343.3.3	Percentage() [1/2]	1505
39.343.3.4	Percentage() [2/2]	1505
39.343.3.5	Samplings() [1/2]	1505
39.343.3.6	Samplings() [2/2]	1505
39.343.3.7	Serialize()	1505
39.344	SampleInitialization Class Reference	1506
39.344.1	Detailed Description	1506
39.344.2	Constructor & Destructor Documentation	1506
39.344.2.1	SampleInitialization()	1506
39.344.3	Member Function Documentation	1506
39.344.3.1	Cluster()	1506
39.345	KernelPCA< KernelType, KernelRule > Class Template Reference	1507
39.345.1	Detailed Description	1508
39.345.2	Constructor & Destructor Documentation	1508
39.345.2.1	KernelPCA()	1508
39.345.3	Member Function Documentation	1508
39.345.3.1	Apply() [1/4]	1508
39.345.3.2	Apply() [2/4]	1509
39.345.3.3	Apply() [3/4]	1509
39.345.3.4	Apply() [4/4]	1510
39.345.3.5	CenterTransformedData() [1/2]	1510
39.345.3.6	CenterTransformedData() [2/2]	1510
39.345.3.7	Kernel() [1/2]	1511
39.345.3.8	Kernel() [2/2]	1511
39.346	NaiveKernelRule< KernelType > Class Template Reference	1511
39.346.1	Detailed Description	1511

39.346.2	Member Function Documentation	1511
39.346.2.1	ApplyKernelMatrix()	1511
39.347	NystroemKernelRule< KernelType, PointSelectionPolicy > Class Template Reference	1512
39.347.1	Detailed Description	1512
39.347.2	Member Function Documentation	1512
39.347.2.1	ApplyKernelMatrix()	1512
39.348	LocalCoordinateCoding Class Reference	1513
39.348.1	Detailed Description	1514
39.348.2	Constructor & Destructor Documentation	1515
39.348.2.1	LocalCoordinateCoding() [1/2]	1515
39.348.2.2	LocalCoordinateCoding() [2/2]	1515
39.348.3	Member Function Documentation	1516
39.348.3.1	Atoms() [1/2]	1516
39.348.3.2	Atoms() [2/2]	1516
39.348.3.3	Dictionary() [1/2]	1516
39.348.3.4	Dictionary() [2/2]	1517
39.348.3.5	Encode()	1517
39.348.3.6	Lambda() [1/2]	1517
39.348.3.7	Lambda() [2/2]	1517
39.348.3.8	MaxIterations() [1/2]	1518
39.348.3.9	MaxIterations() [2/2]	1518
39.348.3.10	Objective()	1518
39.348.3.11	OptimizeDictionary()	1518
39.348.3.12	Serialize()	1519
39.348.3.13	Tolerance() [1/2]	1519
39.348.3.14	Tolerance() [2/2]	1519
39.348.3.15	Train()	1519
39.349	Constraints< MetricType > Class Template Reference	1520

39.349.1	Detailed Description	1521
39.349.2	Member Typedef Documentation	1521
39.349.2.1	KNN	1521
39.349.3	Constructor & Destructor Documentation	1521
39.349.3.1	Constraints()	1521
39.349.4	Member Function Documentation	1522
39.349.4.1	Impostors() [1/5]	1522
39.349.4.2	Impostors() [2/5]	1522
39.349.4.3	Impostors() [3/5]	1523
39.349.4.4	Impostors() [4/5]	1523
39.349.4.5	Impostors() [5/5]	1524
39.349.4.6	K() [1/2]	1524
39.349.4.7	K() [2/2]	1524
39.349.4.8	PreCalulated() [1/2]	1525
39.349.4.9	PreCalulated() [2/2]	1525
39.349.4.10	TargetNeighbors() [1/2]	1525
39.349.4.11	TargetNeighbors() [2/2]	1525
39.349.4.12	Triplets()	1526
39.350	LMNN< MetricType, OptimizerType > Class Template Reference	1526
39.350.1	Detailed Description	1527
39.350.2	Constructor & Destructor Documentation	1528
39.350.2.1	LMNN()	1528
39.350.3	Member Function Documentation	1528
39.350.3.1	Dataset()	1528
39.350.3.2	K() [1/2]	1529
39.350.3.3	K() [2/2]	1529
39.350.3.4	Labels()	1529
39.350.3.5	LearnDistance()	1529

39.350.3.6Optimizer() [1/2]	1530
39.350.3.7Optimizer() [2/2]	1530
39.350.3.8Range() [1/2]	1530
39.350.3.9Range() [2/2]	1530
39.350.3.10Regularization() [1/2]	1530
39.350.3.11Regularization() [2/2]	1531
39.351LMNNFunction< MetricType > Class Template Reference	1531
39.351.1Detailed Description	1532
39.351.2Constructor & Destructor Documentation	1532
39.351.2.1LMNNFunction()	1532
39.351.3Member Function Documentation	1533
39.351.3.1Dataset()	1533
39.351.3.2Evaluate() [1/2]	1533
39.351.3.3Evaluate() [2/2]	1533
39.351.3.4EvaluateWithGradient() [1/2]	1534
39.351.3.5EvaluateWithGradient() [2/2]	1534
39.351.3.6GetInitialPoint()	1535
39.351.3.7Gradient() [1/2]	1535
39.351.3.8Gradient() [2/2]	1536
39.351.3.9K() [1/2]	1536
39.351.3.10K() [2/2]	1537
39.351.3.11NumFunctions()	1537
39.351.3.12Range() [1/2]	1537
39.351.3.13Range() [2/2]	1537
39.351.3.14Regularization() [1/2]	1537
39.351.3.15Regularization() [2/2]	1538
39.351.3.16Shuffle()	1538
39.352Log Class Reference	1538

39.352.1 Detailed Description	1539
39.352.2 Member Function Documentation	1539
39.352.2.1 Assert()	1539
39.352.3 Member Data Documentation	1540
39.352.3.1 cout	1540
39.352.3.2 Debug	1540
39.352.3.3 Fatal	1540
39.352.3.4 Info	1541
39.352.3.5 Warn	1541
39.353 ColumnsToBlocks Class Reference	1541
39.353.1 Detailed Description	1543
39.353.2 Constructor & Destructor Documentation	1544
39.353.2.1 ColumnsToBlocks()	1544
39.353.3 Member Function Documentation	1544
39.353.3.1 BlockHeight() [1/2]	1544
39.353.3.2 BlockHeight() [2/2]	1545
39.353.3.3 BlockWidth() [1/2]	1545
39.353.3.4 BlockWidth() [2/2]	1545
39.353.3.5 BufSize() [1/2]	1545
39.353.3.6 BufSize() [2/2]	1546
39.353.3.7 BufValue() [1/2]	1546
39.353.3.8 BufValue() [2/2]	1546
39.353.3.9 Cols() [1/2]	1546
39.353.3.10 Cols() [2/2]	1546
39.353.3.11 MaxRange() [1/2]	1547
39.353.3.12 MaxRange() [2/2]	1547
39.353.3.13 MinRange() [1/2]	1547
39.353.3.14 MinRange() [2/2]	1547

39.353.3.1	Rows() [1/2]	1548
39.353.3.1	Rows() [2/2]	1548
39.353.3.1	Scale() [1/2]	1548
39.353.3.1	Scale() [2/2]	1548
39.353.3.1	Transform()	1548
39.354	RangeType < T > Class Template Reference	1549
39.354.1	Detailed Description	1550
39.354.2	Constructor & Destructor Documentation	1551
39.354.2.1	RangeType() [1/3]	1551
39.354.2.2	RangeType() [2/3]	1551
39.354.2.3	RangeType() [3/3]	1551
39.354.3	Member Function Documentation	1551
39.354.3.1	Contains() [1/2]	1552
39.354.3.2	Contains() [2/2]	1552
39.354.3.3	Hi() [1/2]	1552
39.354.3.4	Hi() [2/2]	1553
39.354.3.5	Lo() [1/2]	1553
39.354.3.6	Lo() [2/2]	1553
39.354.3.7	Mid()	1553
39.354.3.8	operator &()	1553
39.354.3.9	operator &=()	1554
39.354.3.10	operator"!=()	1554
39.354.3.11	operator*()	1554
39.354.3.12	operator*=()	1555
39.354.3.13	operator<()	1555
39.354.3.14	operator==()	1555
39.354.3.15	operator>()	1556
39.354.3.16	operator" ()	1556

39.354.3.1operator" =()	1556
39.354.3.1Serialize()	1557
39.354.3.1Width()	1557
39.354.4Friends And Related Function Documentation	1557
39.354.4.1operator*	1557
39.355MatrixCompletion Class Reference	1558
39.355.1Detailed Description	1558
39.355.2Constructor & Destructor Documentation	1559
39.355.2.1MatrixCompletion() [1/3]	1559
39.355.2.2MatrixCompletion() [2/3]	1559
39.355.2.3MatrixCompletion() [3/3]	1560
39.355.3Member Function Documentation	1560
39.355.3.1Recover()	1560
39.355.3.2Sdp() [1/2]	1560
39.355.3.3Sdp() [2/2]	1561
39.356MeanShift< UseKernel, KernelType, MatType > Class Template Reference	1561
39.356.1Detailed Description	1562
39.356.2Constructor & Destructor Documentation	1562
39.356.2.1MeanShift()	1562
39.356.3Member Function Documentation	1563
39.356.3.1Cluster()	1563
39.356.3.2EstimateRadius()	1563
39.356.3.3Kernel() [1/2]	1564
39.356.3.4Kernel() [2/2]	1564
39.356.3.5MaxIterations() [1/2]	1564
39.356.3.6MaxIterations() [2/2]	1564
39.356.3.7Radius() [1/2]	1564
39.356.3.8Radius() [2/2]	1565

39.357	IP Metric< KernelType > Class Template Reference	1565
39.357.1	Detailed Description	1566
39.357.2	Constructor & Destructor Documentation	1566
39.357.2.1	IP Metric() [1/3]	1566
39.357.2.2	IP Metric() [2/3]	1567
39.357.2.3	IP Metric()	1567
39.357.2.4	IP Metric() [3/3]	1567
39.357.3	Member Function Documentation	1567
39.357.3.1	Evaluate()	1567
39.357.3.2	Kernel () [1/2]	1568
39.357.3.3	Kernel () [2/2]	1568
39.357.3.4	operator=()	1568
39.357.3.5	Serialize()	1568
39.358	IP Metric< TPower, TTakeRoot > Class Template Reference	1569
39.358.1	Detailed Description	1570
39.358.2	Constructor & Destructor Documentation	1570
39.358.2.1	IP Metric()	1571
39.358.3	Member Function Documentation	1571
39.358.3.1	Evaluate()	1571
39.358.3.2	Serialize()	1571
39.358.4	Member Data Documentation	1572
39.358.4.1	Power	1572
39.358.4.2	TakeRoot	1572
39.359	IP MahalanobisDistance< TakeRoot > Class Template Reference	1572
39.359.1	Detailed Description	1573
39.359.2	Constructor & Destructor Documentation	1573
39.359.2.1	IP MahalanobisDistance() [1/3]	1573
39.359.2.2	IP MahalanobisDistance() [2/3]	1574

39.359.2.3MahalanobisDistance() [3/3]	1574
39.359.3Member Function Documentation	1574
39.359.3.1Covariance() [1/2]	1575
39.359.3.2Covariance() [2/2]	1575
39.359.3.3Evaluate()	1575
39.359.3.4Serialize()	1576
39.360NaiveBayesClassifier< ModelMatType > Class Template Reference	1576
39.360.1Detailed Description	1577
39.360.2Member Typedef Documentation	1578
39.360.2.1ElemType	1578
39.360.3Constructor & Destructor Documentation	1578
39.360.3.1NaiveBayesClassifier() [1/2]	1578
39.360.3.2NaiveBayesClassifier() [2/2]	1579
39.360.4Member Function Documentation	1579
39.360.4.1Classify() [1/4]	1579
39.360.4.2Classify() [2/4]	1579
39.360.4.3Classify() [3/4]	1580
39.360.4.4Classify() [4/4]	1580
39.360.4.5Means() [1/2]	1581
39.360.4.6Means() [2/2]	1581
39.360.4.7Probabilities() [1/2]	1581
39.360.4.8Probabilities() [2/2]	1582
39.360.4.9Serialize()	1582
39.360.4.10Train() [1/2]	1582
39.360.4.11Train() [2/2]	1583
39.360.4.12Variances() [1/2]	1583
39.360.4.13Variances() [2/2]	1583
39.360NCA< MetricType, OptimizerType > Class Template Reference	1583

39.361.1	Detailed Description	1584
39.361.2	Constructor & Destructor Documentation	1584
39.361.2.1	NCA()	1584
39.361.3	Member Function Documentation	1585
39.361.3.1	Dataset()	1585
39.361.3.2	Labels()	1585
39.361.3.3	LearnDistance()	1585
39.361.3.4	Optimizer() [1/2]	1586
39.361.3.5	Optimizer() [2/2]	1586
39.362	SoftmaxErrorFunction< MetricType > Class Template Reference	1586
39.362.1	Detailed Description	1587
39.362.2	Constructor & Destructor Documentation	1587
39.362.2.1	SoftmaxErrorFunction()	1587
39.362.3	Member Function Documentation	1588
39.362.3.1	Evaluate() [1/2]	1588
39.362.3.2	Evaluate() [2/2]	1588
39.362.3.3	GetInitialPoint()	1588
39.362.3.4	Gradient() [1/2]	1589
39.362.3.5	Gradient() [2/2]	1589
39.362.3.6	NumFunctions()	1590
39.362.3.7	Shuffle()	1590
39.363	AlphaVisitor Class Reference	1590
39.363.1	Detailed Description	1591
39.363.2	Member Function Documentation	1591
39.363.2.1	operator()()	1591
39.364	BiSearchVisitor< SortPolicy > Class Template Reference	1591
39.364.1	Detailed Description	1592
39.364.2	Member Typedef Documentation	1592

39.364.2.1	INSTypeT	1593
39.364.2.2	RATypeT	1593
39.364.3	Constructor & Destructor Documentation	1593
39.364.3.1	BiSearchVisitor() [1/2]	1593
39.364.3.2	BiSearchVisitor() [2/2]	1593
39.364.4	Member Function Documentation	1594
39.364.4.1	operator()() [1/8]	1594
39.364.4.2	operator()() [2/8]	1594
39.364.4.3	operator()() [3/8]	1594
39.364.4.4	operator()() [4/8]	1594
39.364.4.5	operator()() [5/8]	1594
39.364.4.6	operator()() [6/8]	1595
39.364.4.7	operator()() [7/8]	1595
39.364.4.8	operator()() [8/8]	1595
39.365	DeleteVisitor Class Reference	1595
39.365.1	Detailed Description	1596
39.365.2	Member Function Documentation	1596
39.365.2.1	operator()() [1/2]	1596
39.365.2.2	operator()() [2/2]	1596
39.366	DrusillaSelect< MatType > Class Template Reference	1597
39.366.1	Detailed Description	1597
39.366.2	Constructor & Destructor Documentation	1597
39.366.2.1	DrusillaSelect() [1/2]	1597
39.366.2.2	DrusillaSelect() [2/2]	1598
39.366.3	Member Function Documentation	1598
39.366.3.1	CandidateIndices() [1/2]	1598
39.366.3.2	CandidateIndices() [2/2]	1598
39.366.3.3	CandidateSet() [1/2]	1599

39.366.3.4	CandidateSet() [2/2]	1599
39.366.3.5	Search()	1599
39.366.3.6	Serialize()	1599
39.366.3.7	Train()	1600
39.367	EpsilonVisitor Class Reference	1600
39.367.1	Detailed Description	1601
39.367.2	Member Function Documentation	1601
39.367.2.1	operator()()	1601
39.368	FirstLeafExactVisitor Class Reference	1601
39.368.1	Detailed Description	1602
39.368.2	Member Function Documentation	1602
39.368.2.1	operator()()	1602
39.369	FurthestNS Class Reference	1602
39.369.1	Detailed Description	1603
39.369.2	Member Function Documentation	1603
39.369.2.1	BestDistance()	1604
39.369.2.2	BestNodeToNodeDistance() [1/3]	1604
39.369.2.3	BestNodeToNodeDistance() [2/3]	1604
39.369.2.4	BestNodeToNodeDistance() [3/3]	1604
39.369.2.5	BestPointToNodeDistance() [1/2]	1605
39.369.2.6	BestPointToNodeDistance() [2/2]	1605
39.369.2.7	CombineBest()	1605
39.369.2.8	CombineWorst()	1606
39.369.2.9	ConvertToDistance()	1606
39.369.2.10	ConvertToScore()	1606
39.369.2.11	GetBestChild() [1/2]	1607
39.369.2.12	GetBestChild() [2/2]	1607
39.369.2.13	Better()	1607

39.369.2.1 Relax()	1608
39.369.2.1 WorstDistance()	1608
39.370 LSHSearch < SortPolicy > Class Template Reference	1609
39.370.1Detailed Description	1610
39.370.2Constructor & Destructor Documentation	1610
39.370.2.1LSHSearch() [1/5]	1610
39.370.2.2LSHSearch() [2/5]	1611
39.370.2.3LSHSearch() [3/5]	1612
39.370.2.4LSHSearch() [4/5]	1612
39.370.2.5LSHSearch() [5/5]	1612
39.370.3Member Function Documentation	1612
39.370.3.1BucketSize()	1612
39.370.3.2ComputeRecall()	1613
39.370.3.3DistanceEvaluations() [1/2]	1613
39.370.3.4DistanceEvaluations() [2/2]	1613
39.370.3.5NumProjections()	1614
39.370.3.6Offsets()	1614
39.370.3.7operator=() [1/2]	1614
39.370.3.8operator=() [2/2]	1614
39.370.3.9Projections() [1/2]	1615
39.370.3.10Projections() [2/2]	1615
39.370.3.11ReferenceSet()	1615
39.370.3.12Search() [1/2]	1615
39.370.3.13Search() [2/2]	1616
39.370.3.14SecondHashTable()	1616
39.370.3.15SecondHashWeights()	1617
39.370.3.16Serialize()	1617
39.370.3.17Train()	1617

39.371	MonoSearchVisitor Class Reference	1618
39.371.1	Detailed Description	1619
39.371.2	Constructor & Destructor Documentation	1619
39.371.2.1	MonoSearchVisitor() [1/2]	1619
39.371.2.2	MonoSearchVisitor() [2/2]	1619
39.371.3	Member Function Documentation	1619
39.371.3.1	operator()() [1/2]	1620
39.371.3.2	operator()() [2/2]	1620
39.372	NaiveVisitor Class Reference	1620
39.372.1	Detailed Description	1621
39.372.2	Member Function Documentation	1621
39.372.2.1	operator()()	1621
39.373	NearestNS Class Reference	1621
39.373.1	Detailed Description	1622
39.373.2	Member Function Documentation	1622
39.373.2.1	BestDistance()	1623
39.373.2.2	BestNodeToNodeDistance() [1/3]	1623
39.373.2.3	BestNodeToNodeDistance() [2/3]	1623
39.373.2.4	BestNodeToNodeDistance() [3/3]	1623
39.373.2.5	BestPointToNodeDistance() [1/2]	1624
39.373.2.6	BestPointToNodeDistance() [2/2]	1624
39.373.2.7	CombineBest()	1624
39.373.2.8	CombineWorst()	1625
39.373.2.9	ConvertToDistance()	1625
39.373.2.10	ConvertToScore()	1625
39.373.2.11	GetBestChild() [1/2]	1625
39.373.2.12	GetBestChild() [2/2]	1626
39.373.2.13	Better()	1626

39.373.2.1 Relax ()	1626
39.373.2.1 W orstDistance()	1627
39.374NeighborSearch< SortPolicy, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTree↵ TraversalType > Class Template Reference	1627
39.374.1Detailed Description	1630
39.374.2Member Typedef Documentation	1630
39.374.2.1Tree	1630
39.374.3Constructor & Destructor Documentation	1630
39.374.3.1NeighborSearch() [1/5]	1631
39.374.3.2NeighborSearch() [2/5]	1631
39.374.3.3NeighborSearch() [3/5]	1632
39.374.3.4NeighborSearch() [4/5]	1632
39.374.3.5NeighborSearch() [5/5]	1632
39.374.3.6~NeighborSearch()	1633
39.374.4Member Function Documentation	1633
39.374.4.1BaseCases()	1633
39.374.4.2EffectiveError()	1633
39.374.4.3Epsilon() [1/2]	1634
39.374.4.4Epsilon() [2/2]	1634
39.374.4.5operator=() [1/2]	1634
39.374.4.6operator=() [2/2]	1635
39.374.4.7Recall()	1635
39.374.4.8ReferenceSet()	1635
39.374.4.9ReferenceTree() [1/2]	1636
39.374.4.10ReferenceTree() [2/2]	1636
39.374.4.11Scores()	1636
39.374.4.12Search() [1/3]	1636
39.374.4.13Search() [2/3]	1637

39.374.4.1 Search() [3/3]	1637
39.374.4.1 SearchMode() [1/2]	1638
39.374.4.1 SearchMode() [2/2]	1638
39.374.4.1 Serialize()	1638
39.374.4.1 Train() [1/2]	1639
39.374.4.1 Train() [2/2]	1640
39.375.1 NeighborSearchRules < SortPolicy, MetricType, TreeType > Class Template Reference	1640
39.375.1 Detailed Description	1642
39.375.2 Member Typedef Documentation	1643
39.375.2.1 Candidate	1643
39.375.2.2 CandidateList	1643
39.375.2.3 TraversallInfoType	1643
39.375.3 Constructor & Destructor Documentation	1643
39.375.3.1 NeighborSearchRules()	1643
39.375.4 Member Function Documentation	1644
39.375.4.1 BaseCase()	1644
39.375.4.2 BaseCases() [1/2]	1644
39.375.4.3 BaseCases() [2/2]	1645
39.375.4.4 CalculateBound()	1645
39.375.4.5 GetBestChild() [1/2]	1645
39.375.4.6 GetBestChild() [2/2]	1645
39.375.4.7 GetResults()	1646
39.375.4.8 InsertNeighbor()	1646
39.375.4.9 Rescore() [1/2]	1646
39.375.4.10 Rescore() [2/2]	1647
39.375.4.11 Score() [1/2]	1647
39.375.4.12 Score() [2/2]	1648
39.375.4.13 Scores() [1/2]	1648

39.375.4.1	39.375.4.1 <code>scores()</code> [2/2]	1648
39.375.4.1	39.375.4.1 <code>TraversalInfo()</code> [1/2]	1649
39.375.4.1	39.375.4.1 <code>TraversalInfo()</code> [2/2]	1649
39.375.5	39.375.5 Member Data Documentation	1649
39.375.5.1	39.375.5.1 <code>baseCases</code>	1649
39.375.5.2	39.375.5.2 <code>candidates</code>	1649
39.375.5.3	39.375.5.3 <code>epsilon</code>	1650
39.375.5.4	39.375.5.4 <code>k</code>	1650
39.375.5.5	39.375.5.5 <code>lastBaseCase</code>	1650
39.375.5.6	39.375.5.6 <code>lastQueryIndex</code>	1650
39.375.5.7	39.375.5.7 <code>lastReferenceIndex</code>	1650
39.375.5.8	39.375.5.8 <code>metric</code>	1651
39.375.5.9	39.375.5.9 <code>querySet</code>	1651
39.375.5.10	39.375.5.10 <code>referenceSet</code>	1651
39.375.5.11	39.375.5.11 <code>sameSet</code>	1651
39.375.5.12	39.375.5.12 <code>scores</code>	1651
39.375.5.13	39.375.5.13 <code>TraversalInfo</code>	1652
39.376	39.376 <code>NeighborSearchRules< SortPolicy, MetricType, TreeType >::CandidateCmp</code> Struct Reference	1652
39.376.1	39.376.1 Detailed Description	1652
39.376.2	39.376.2 Member Function Documentation	1652
39.376.2.1	39.376.2.1 <code>operator>()()</code>	1652
39.377	39.377 <code>NeighborSearchStat< SortPolicy ></code> Class Template Reference	1653
39.377.1	39.377.1 Detailed Description	1654
39.377.2	39.377.2 Constructor & Destructor Documentation	1654
39.377.2.1	39.377.2.1 <code>NeighborSearchStat()</code> [1/2]	1654
39.377.2.2	39.377.2.2 <code>NeighborSearchStat()</code> [2/2]	1654
39.377.3	39.377.3 Member Function Documentation	1655
39.377.3.1	39.377.3.1 <code>AuxBound()</code> [1/2]	1655

39.377.3.2	AuxBound() [2/2]	1655
39.377.3.3	FirstBound() [1/2]	1655
39.377.3.4	FirstBound() [2/2]	1655
39.377.3.5	LastDistance() [1/2]	1656
39.377.3.6	LastDistance() [2/2]	1656
39.377.3.7	Reset()	1656
39.377.3.8	SecondBound() [1/2]	1656
39.377.3.9	SecondBound() [2/2]	1656
39.377.3.10	Serialize()	1657
39.378	NSModel< SortPolicy > Class Template Reference	1657
39.378.1	Detailed Description	1659
39.378.2	Member Enumeration Documentation	1659
39.378.2.1	TreeTypes	1659
39.378.3	Constructor & Destructor Documentation	1660
39.378.3.1	NSModel() [1/3]	1660
39.378.3.2	NSModel() [2/3]	1660
39.378.3.3	NSModel() [3/3]	1660
39.378.3.4	~NSModel()	1661
39.378.4	Member Function Documentation	1661
39.378.4.1	BuildModel()	1661
39.378.4.2	Dataset()	1661
39.378.4.3	Epsilon() [1/2]	1661
39.378.4.4	Epsilon() [2/2]	1662
39.378.4.5	LeafSize() [1/2]	1662
39.378.4.6	LeafSize() [2/2]	1662
39.378.4.7	operator=() [1/2]	1662
39.378.4.8	operator=() [2/2]	1662
39.378.4.9	RandomBasis() [1/2]	1663

39.378.4.1 0 RandomBasis()	[2/2]	1663
39.378.4.1 1 Rho()	[1/2]	1663
39.378.4.1 2 Rho()	[2/2]	1663
39.378.4.1 3 Search()	[1/2]	1664
39.378.4.1 3 Search()	[2/2]	1664
39.378.4.1 5 SearchMode()	[1/2]	1664
39.378.4.1 6 SearchMode()	[2/2]	1664
39.378.4.1 7 Serialize()		1664
39.378.4.1 8 Tau()	[1/2]	1665
39.378.4.1 8 Tau()	[2/2]	1665
39.378.4.2 0 TreeName()		1665
39.378.4.2 1 TreeType()	[1/2]	1665
39.378.4.2 2 TreeType()	[2/2]	1665
39.379.0 0 DAFN< MatType > Class Template Reference		1666
39.379.1 0 Detailed Description		1666
39.379.2 0 Constructor & Destructor Documentation		1666
39.379.2.1 0 DAFN()	[1/2]	1666
39.379.2.2 0 DAFN()	[2/2]	1667
39.379.3 0 Member Function Documentation		1667
39.379.3.1 0 CandidateSet()	[1/2]	1667
39.379.3.2 0 CandidateSet()	[2/2]	1667
39.379.3.3 0 NumProjections()		1668
39.379.3.4 0 Search()		1668
39.379.3.5 0 Serialize()		1668
39.379.3.6 0 Train()		1668
39.380.0 0 RAModel< SortPolicy > Class Template Reference		1669
39.380.1 0 Detailed Description		1671
39.380.2 0 Member Enumeration Documentation		1671

39.380.2.1	TreeTypes	1671
39.380.3	Constructor & Destructor Documentation	1672
39.380.3.1	IRAModel() [1/3]	1672
39.380.3.2	IRAModel() [2/3]	1672
39.380.3.3	IRAModel() [3/3]	1672
39.380.3.4	~IRAModel()	1672
39.380.4	Member Function Documentation	1673
39.380.4.1	Alpha() [1/2]	1673
39.380.4.2	Alpha() [2/2]	1673
39.380.4.3	BuildModel()	1673
39.380.4.4	Dataset()	1673
39.380.4.5	FirstLeafExact() [1/2]	1673
39.380.4.6	FirstLeafExact() [2/2]	1674
39.380.4.7	LeafSize() [1/2]	1674
39.380.4.8	LeafSize() [2/2]	1674
39.380.4.9	Naive() [1/2]	1674
39.380.4.10	Naive() [2/2]	1674
39.380.4.11	operator=() [1/2]	1674
39.380.4.12	operator=() [2/2]	1675
39.380.4.13	RandomBasis() [1/2]	1675
39.380.4.14	RandomBasis() [2/2]	1675
39.380.4.15	SampleAtLeaves() [1/2]	1675
39.380.4.16	SampleAtLeaves() [2/2]	1676
39.380.4.17	Search() [1/2]	1676
39.380.4.18	Search() [2/2]	1676
39.380.4.19	Serialize()	1676
39.380.4.20	SingleMode() [1/2]	1676
39.380.4.21	SingleMode() [2/2]	1677

39.380.4.2SingleSampleLimit() [1/2]	1677
39.380.4.2SingleSampleLimit() [2/2]	1677
39.380.4.2Tau() [1/2]	1677
39.380.4.2Tau() [2/2]	1677
39.380.4.2TreeName()	1677
39.380.4.2TreeType() [1/2]	1678
39.380.4.2TreeType() [2/2]	1678
39.381RAQueryStat< SortPolicy > Class Template Reference	1678
39.381.1Detailed Description	1679
39.381.2Constructor & Destructor Documentation	1679
39.381.2.1RAQueryStat() [1/2]	1679
39.381.2.2RAQueryStat() [2/2]	1679
39.381.3Member Function Documentation	1679
39.381.3.1Bound() [1/2]	1680
39.381.3.2Bound() [2/2]	1680
39.381.3.3NumSamplesMade() [1/2]	1680
39.381.3.4NumSamplesMade() [2/2]	1680
39.381.3.5Serialize()	1680
39.382RASearch< SortPolicy, MetricType, MatType, TreeType > Class Template Reference	1681
39.382.1Detailed Description	1682
39.382.2Member Typedef Documentation	1683
39.382.2.1Tree	1683
39.382.3Constructor & Destructor Documentation	1683
39.382.3.1RASearch() [1/3]	1683
39.382.3.2RASearch() [2/3]	1684
39.382.3.3RASearch() [3/3]	1685
39.382.3.4~RASearch()	1685
39.382.4Member Function Documentation	1686

39.382.4.1Alpha() [1/2]	1686
39.382.4.2Alpha() [2/2]	1686
39.382.4.3FirstLeafExact() [1/2]	1686
39.382.4.4FirstLeafExact() [2/2]	1686
39.382.4.5Naive() [1/2]	1687
39.382.4.6Naive() [2/2]	1687
39.382.4.7ReferenceSet()	1687
39.382.4.8ResetQueryTree()	1687
39.382.4.9SampleAtLeaves() [1/2]	1688
39.382.4.10SampleAtLeaves() [2/2]	1688
39.382.4.11Search() [1/3]	1688
39.382.4.12Search() [2/3]	1689
39.382.4.13Search() [3/3]	1689
39.382.4.14Serialize()	1690
39.382.4.15SingleMode() [1/2]	1690
39.382.4.16SingleMode() [2/2]	1690
39.382.4.17SingleSampleLimit() [1/2]	1690
39.382.4.18SingleSampleLimit() [2/2]	1691
39.382.4.19Tau() [1/2]	1691
39.382.4.20Tau() [2/2]	1691
39.382.4.21Train() [1/2]	1691
39.382.4.22Train() [2/2]	1692
39.383.1IRASearchRules< SortPolicy, MetricType, TreeType > Class Template Reference	1692
39.383.2Detailed Description	1693
39.383.3Member Typedef Documentation	1693
39.383.4TraversallInfoType	1693
39.383.5Constructor & Destructor Documentation	1693
39.383.6IRASearchRules()	1693

39.383.4	Member Function Documentation	1694
39.383.4.1	BaseCase()	1694
39.383.4.2	GetResults()	1694
39.383.4.3	NumDistComputations()	1695
39.383.4.4	NumEffectiveSamples()	1695
39.383.4.5	Rescore() [1/2]	1695
39.383.4.6	Rescore() [2/2]	1696
39.383.4.7	Score() [1/4]	1696
39.383.4.8	Score() [2/4]	1697
39.383.4.9	Score() [3/4]	1697
39.383.4.10	Score() [4/4]	1698
39.383.4.11	TraversalInfo() [1/2]	1698
39.383.4.12	TraversalInfo() [2/2]	1699
39.384	RAUtil Class Reference	1699
39.384.1	Detailed Description	1699
39.384.2	Member Function Documentation	1699
39.384.2.1	MinimumSamplesReqd()	1699
39.384.2.2	ObtainDistinctSamples()	1700
39.384.2.3	SuccessProbability()	1700
39.385	ReferenceSetVisitor Class Reference	1701
39.385.1	Detailed Description	1701
39.385.2	Member Function Documentation	1701
39.385.2.1	operator()() [1/2]	1702
39.385.2.2	operator()() [2/2]	1702
39.386	SampleAtLeavesVisitor Class Reference	1702
39.386.1	Detailed Description	1703
39.386.2	Member Function Documentation	1703
39.386.2.1	operator()()	1703

39.385	SearchModeVisitor Class Reference	1703
39.387.1	Detailed Description	1704
39.387.2	Member Function Documentation	1704
39.387.2.1	operator>()	1704
39.388	SingleModeVisitor Class Reference	1704
39.388.1	Detailed Description	1705
39.388.2	Member Function Documentation	1705
39.388.2.1	operator>()	1705
39.389	SingleSampleLimitVisitor Class Reference	1705
39.389.1	Detailed Description	1706
39.389.2	Member Function Documentation	1706
39.389.2.1	operator>()	1706
39.390	TauVisitor Class Reference	1706
39.390.1	Detailed Description	1707
39.390.2	Member Function Documentation	1707
39.390.2.1	operator>()	1707
39.391	TrainVisitor< SortPolicy > Class Template Reference	1707
39.391.1	Detailed Description	1708
39.391.2	Member Typedef Documentation	1708
39.391.2.1	INSTypeT	1709
39.391.2.2	RATypeT	1709
39.391.3	Constructor & Destructor Documentation	1709
39.391.3.1	TrainVisitor() [1/2]	1709
39.391.3.2	TrainVisitor() [2/2]	1709
39.391.4	Member Function Documentation	1709
39.391.4.1	operator>() [1/8]	1710
39.391.4.2	operator>() [2/8]	1710
39.391.4.3	operator>() [3/8]	1710

39.391.4.4operator() [4/8]	1710
39.391.4.5operator() [5/8]	1710
39.391.4.6operator() [6/8]	1711
39.391.4.7operator() [7/8]	1711
39.391.4.8operator() [8/8]	1711
39.392SparseAutoencoder Class Reference	1711
39.392.1Detailed Description	1712
39.392.2Constructor & Destructor Documentation	1713
39.392.2.1SparseAutoencoder()	1713
39.392.3Member Function Documentation	1714
39.392.3.1Beta() [1/2]	1714
39.392.3.2Beta() [2/2]	1714
39.392.3.3GetNewFeatures()	1714
39.392.3.4HiddenSize() [1/2]	1715
39.392.3.5HiddenSize() [2/2]	1715
39.392.3.6Lambda() [1/2]	1715
39.392.3.7Lambda() [2/2]	1715
39.392.3.8Rho() [1/2]	1716
39.392.3.9Rho() [2/2]	1716
39.392.3.10Sigmoid()	1716
39.392.3.11VisibleSize() [1/2]	1716
39.392.3.12VisibleSize() [2/2]	1717
39.393SparseAutoencoderFunction Class Reference	1717
39.393.1Detailed Description	1718
39.393.2Constructor & Destructor Documentation	1718
39.393.2.1SparseAutoencoderFunction()	1718
39.393.3Member Function Documentation	1718
39.393.3.1Beta() [1/2]	1718

39.393.3.2Beta() [2/2]	1719
39.393.3.3Evaluate()	1719
39.393.3.4GetInitialPoint()	1719
39.393.3.5Gradient()	1719
39.393.3.6HiddenSize() [1/2]	1720
39.393.3.7HiddenSize() [2/2]	1720
39.393.3.8InitializeWeights()	1720
39.393.3.9Lambda() [1/2]	1720
39.393.3.10Lambda() [2/2]	1721
39.393.3.11Rho() [1/2]	1721
39.393.3.12Rho() [2/2]	1721
39.393.3.13Sigmoid()	1721
39.393.3.14VisibleSize() [1/2]	1722
39.393.3.15VisibleSize() [2/2]	1722
39.394.ExactSVDPolicy Class Reference	1722
39.394.1Detailed Description	1722
39.394.2Member Function Documentation	1722
39.394.2.1Apply()	1722
39.395.PCA< DecompositionPolicy > Class Template Reference	1723
39.395.1Detailed Description	1724
39.395.2Constructor & Destructor Documentation	1724
39.395.2.1PCA()	1724
39.395.3Member Function Documentation	1724
39.395.3.1Apply() [1/5]	1724
39.395.3.2Apply() [2/5]	1725
39.395.3.3Apply() [3/5]	1725
39.395.3.4Apply() [4/5]	1726
39.395.3.5Apply() [5/5]	1726

39.395.3.6	ScaleData() [1/2]	1727
39.395.3.7	ScaleData() [2/2]	1727
39.396	QUICSVDPolicy Class Reference	1727
39.396.1	Detailed Description	1728
39.396.2	Constructor & Destructor Documentation	1728
39.396.2.1	QUICSVDPolicy()	1728
39.396.3	Member Function Documentation	1728
39.396.3.1	Apply()	1728
39.396.3.2	Delta() [1/2]	1729
39.396.3.3	Delta() [2/2]	1729
39.396.3.4	Epsilon() [1/2]	1729
39.396.3.5	Epsilon() [2/2]	1730
39.397	RandomizedBlockKrylovSVDPolicy Class Reference	1730
39.397.1	Detailed Description	1730
39.397.2	Constructor & Destructor Documentation	1730
39.397.2.1	RandomizedBlockKrylovSVDPolicy()	1730
39.397.3	Member Function Documentation	1731
39.397.3.1	Apply()	1731
39.397.3.2	BlockSize() [1/2]	1731
39.397.3.3	BlockSize() [2/2]	1732
39.397.3.4	MaxIterations() [1/2]	1732
39.397.3.5	MaxIterations() [2/2]	1732
39.398	RandomizedSVDPolicy Class Reference	1732
39.398.1	Detailed Description	1733
39.398.2	Constructor & Destructor Documentation	1733
39.398.2.1	RandomizedSVDPolicy()	1733
39.398.3	Member Function Documentation	1733
39.398.3.1	Apply()	1733

39.398.3.2	IteratedPower() [1/2]	1734
39.398.3.3	IteratedPower() [2/2]	1734
39.398.3.4	MaxIterations() [1/2]	1734
39.398.3.5	MaxIterations() [2/2]	1735
39.399	Perceptron< LearnPolicy, WeightInitializationPolicy, MatType > Class Template Reference	1735
39.399.1	Detailed Description	1736
39.399.2	Constructor & Destructor Documentation	1736
39.399.2.1	Perceptron() [1/3]	1736
39.399.2.2	Perceptron() [2/3]	1736
39.399.2.3	Perceptron() [3/3]	1737
39.399.3	Member Function Documentation	1737
39.399.3.1	Biases() [1/2]	1737
39.399.3.2	Biases() [2/2]	1738
39.399.3.3	Classify()	1738
39.399.3.4	MaxIterations() [1/2]	1738
39.399.3.5	MaxIterations() [2/2]	1739
39.399.3.6	NumClasses()	1739
39.399.3.7	serialize()	1739
39.399.3.8	Train()	1739
39.399.3.9	Weights() [1/2]	1740
39.399.3.10	Weights() [2/2]	1740
39.400	RandomInitialization Class Reference	1740
39.400.1	Detailed Description	1741
39.400.2	Constructor & Destructor Documentation	1741
39.400.2.1	RandomInitialization()	1741
39.400.3	Member Function Documentation	1741
39.400.3.1	Initialize()	1741
39.401	SimpleWeightUpdate Class Reference	1741

39.401.1	Detailed Description	1742
39.401.2	Member Function Documentation	1742
39.401.2.1	UpdateWeights()	1742
39.402	ZeroInitialization Class Reference	1742
39.402.1	Detailed Description	1743
39.402.2	Constructor & Destructor Documentation	1743
39.402.2.1	ZeroInitialization()	1743
39.402.3	Member Function Documentation	1743
39.402.3.1	Initialize()	1743
39.403	Radical Class Reference	1744
39.403.1	Detailed Description	1744
39.403.2	Constructor & Destructor Documentation	1745
39.403.2.1	Radical()	1745
39.403.3	Member Function Documentation	1745
39.403.3.1	Angles() [1/2]	1745
39.403.3.2	Angles() [2/2]	1745
39.403.3.3	CopyAndPerturb()	1746
39.403.3.4	DoRadical()	1746
39.403.3.5	DoRadical2D()	1746
39.403.3.6	NoiseStdDev() [1/2]	1746
39.403.3.7	NoiseStdDev() [2/2]	1747
39.403.3.8	Replicates() [1/2]	1747
39.403.3.9	Replicates() [2/2]	1747
39.403.3.10	Sweeps() [1/2]	1747
39.403.3.11	Sweeps() [2/2]	1747
39.403.3.12	Basic() [1/2]	1747
39.403.3.13	Basic() [2/2]	1747
39.404	BiSearchVisitor Class Reference	1748
39.404.1	Detailed Description	1749

39.404.2	Member Typedef Documentation	1749
39.404.2.1	RSTypeT	1749
39.404.3	Constructor & Destructor Documentation	1749
39.404.3.1	BiSearchVisitor()	1749
39.404.4	Member Function Documentation	1749
39.404.4.1	operator() [1/4]	1750
39.404.4.2	operator() [2/4]	1750
39.404.4.3	operator() [3/4]	1750
39.404.4.4	operator() [4/4]	1750
39.405	DeleteVisitor Class Reference	1751
39.405.1	Detailed Description	1751
39.405.2	Member Function Documentation	1751
39.405.2.1	operator()	1751
39.406	MonoSearchVisitor Class Reference	1752
39.406.1	Detailed Description	1752
39.406.2	Constructor & Destructor Documentation	1752
39.406.2.1	MonoSearchVisitor()	1752
39.406.3	Member Function Documentation	1753
39.406.3.1	operator()	1753
39.407	NaiveVisitor Class Reference	1753
39.407.1	Detailed Description	1753
39.407.2	Member Function Documentation	1754
39.407.2.1	operator()	1754
39.408	RangeSearch< MetricType, MatType, TreeType > Class Template Reference	1754
39.408.1	Detailed Description	1755
39.408.2	Member Typedef Documentation	1756
39.408.2.1	Tree	1756
39.408.3	Constructor & Destructor Documentation	1756

39.408.3.1RangeSearch() [1/5]	1756
39.408.3.2RangeSearch() [2/5]	1757
39.408.3.3RangeSearch() [3/5]	1757
39.408.3.4RangeSearch() [4/5]	1758
39.408.3.5RangeSearch() [5/5]	1758
39.408.3.6~RangeSearch()	1758
39.408.4Member Function Documentation	1758
39.408.4.1BaseCases()	1759
39.408.4.2Naive() [1/2]	1759
39.408.4.3Naive() [2/2]	1759
39.408.4.4operator=()	1759
39.408.4.5ReferenceSet()	1760
39.408.4.6ReferenceTree()	1760
39.408.4.7Scores()	1760
39.408.4.8Search() [1/3]	1760
39.408.4.9Search() [2/3]	1761
39.408.4.10Search() [3/3]	1762
39.408.4.11Serialize()	1762
39.408.4.12SingleMode() [1/2]	1763
39.408.4.13SingleMode() [2/2]	1763
39.408.4.14Train() [1/2]	1763
39.408.4.15Train() [2/2]	1763
39.409RangeSearchRules< MetricType, TreeType > Class Template Reference	1764
39.409.1Detailed Description	1764
39.409.2Member Typedef Documentation	1765
39.409.2.1TraversallInfoType	1765
39.409.3Constructor & Destructor Documentation	1765
39.409.3.1RangeSearchRules()	1765

39.409.4	Member Function Documentation	1766
39.409.4.1	BaseCase()	1766
39.409.4.2	BaseCases()	1766
39.409.4.3	Rescore() [1/2]	1766
39.409.4.4	Rescore() [2/2]	1767
39.409.4.5	Score() [1/2]	1767
39.409.4.6	Score() [2/2]	1767
39.409.4.7	Scores()	1768
39.409.4.8	TraversallInfo() [1/2]	1768
39.409.4.9	TraversallInfo() [2/2]	1768
39.410	RangeSearchStat Class Reference	1768
39.410.1	Detailed Description	1769
39.410.2	Constructor & Destructor Documentation	1769
39.410.2.1	RangeSearchStat() [1/2]	1769
39.410.2.2	RangeSearchStat() [2/2]	1769
39.410.3	Member Function Documentation	1770
39.410.3.1	LastDistance() [1/2]	1770
39.410.3.2	LastDistance() [2/2]	1770
39.410.3.3	Serialize()	1770
39.411	ReferenceSetVisitor Class Reference	1771
39.411.1	Detailed Description	1771
39.411.2	Member Function Documentation	1771
39.411.2.1	operator>()	1771
39.412	RSModel Class Reference	1772
39.412.1	Detailed Description	1773
39.412.2	Member Enumeration Documentation	1773
39.412.2.1	TreeTypes	1773
39.412.3	Constructor & Destructor Documentation	1774

39.412.3.1	RSModel() [1/3]	1774
39.412.3.2	RSModel() [2/3]	1774
39.412.3.3	RSModel() [3/3]	1774
39.412.3.4	~RSModel()	1775
39.412.4	Member Function Documentation	1775
39.412.4.1	BuildModel()	1775
39.412.4.2	Dataset()	1775
39.412.4.3	LeafSize() [1/2]	1776
39.412.4.4	LeafSize() [2/2]	1776
39.412.4.5	Naive() [1/2]	1776
39.412.4.6	Naive() [2/2]	1776
39.412.4.7	operator=()	1776
39.412.4.8	RandomBasis() [1/2]	1777
39.412.4.9	RandomBasis() [2/2]	1777
39.412.4.10	Search() [1/2]	1777
39.412.4.11	Search() [2/2]	1778
39.412.4.12	Serialize()	1778
39.412.4.13	SingleMode() [1/2]	1778
39.412.4.14	SingleMode() [2/2]	1778
39.412.4.15	TreeType() [1/2]	1779
39.412.4.16	TreeType() [2/2]	1779
39.413	SingleModeVisitor Class Reference	1779
39.413.1	Detailed Description	1780
39.413.2	Member Function Documentation	1780
39.413.2.1	operator()()	1780
39.414	TrainVisitor Class Reference	1780
39.414.1	Detailed Description	1781
39.414.2	Member Typedef Documentation	1781

39.414.2.1RSTypeT	1781
39.414.3Constructor & Destructor Documentation	1781
39.414.3.1TrainVisitor()	1781
39.414.4Member Function Documentation	1782
39.414.4.1operator>() [1/4]	1782
39.414.4.2operator>() [2/4]	1782
39.414.4.3operator>() [3/4]	1782
39.414.4.4operator>() [4/4]	1782
39.415LARS Class Reference	1783
39.415.1Detailed Description	1784
39.415.2Constructor & Destructor Documentation	1784
39.415.2.1LARS() [1/4]	1784
39.415.2.2LARS() [2/4]	1785
39.415.2.3LARS() [3/4]	1785
39.415.2.4LARS() [4/4]	1786
39.415.3Member Function Documentation	1786
39.415.3.1ActiveSet()	1787
39.415.3.2Beta()	1787
39.415.3.3BetaPath()	1787
39.415.3.4LambdaPath()	1787
39.415.3.5MatUtriCholFactor()	1787
39.415.3.6Predict()	1787
39.415.3.7serialize()	1788
39.415.3.8Train() [1/2]	1788
39.415.3.9Train() [2/2]	1789
39.416LinearRegression Class Reference	1789
39.416.1Detailed Description	1790
39.416.2Constructor & Destructor Documentation	1790

39.416.2.1	LinearRegression() [1/3]	1790
39.416.2.2	LinearRegression() [2/3]	1791
39.416.2.3	LinearRegression() [3/3]	1791
39.416.3	Member Function Documentation	1791
39.416.3.1	ComputeError()	1792
39.416.3.2	Intercept()	1792
39.416.3.3	Lambda() [1/2]	1792
39.416.3.4	Lambda() [2/2]	1793
39.416.3.5	Parameters() [1/2]	1793
39.416.3.6	Parameters() [2/2]	1793
39.416.3.7	Predict()	1793
39.416.3.8	Serialize()	1794
39.416.3.9	Train() [1/2]	1794
39.416.3.10	Train() [2/2]	1794
39.417	LogisticRegression< MatType > Class Template Reference	1795
39.417.1	Detailed Description	1796
39.417.2	Constructor & Destructor Documentation	1797
39.417.2.1	LogisticRegression() [1/4]	1797
39.417.2.2	LogisticRegression() [2/4]	1797
39.417.2.3	LogisticRegression() [3/4]	1798
39.417.2.4	LogisticRegression() [4/4]	1798
39.417.3	Member Function Documentation	1798
39.417.3.1	Classify() [1/3]	1799
39.417.3.2	Classify() [2/3]	1799
39.417.3.3	Classify() [3/3]	1800
39.417.3.4	ComputeAccuracy()	1800
39.417.3.5	ComputeError()	1800
39.417.3.6	Lambda() [1/2]	1801

39.417.3.7	Lambda() [2/2]	1801
39.417.3.8	Parameters() [1/2]	1801
39.417.3.9	Parameters() [2/2]	1802
39.417.3.10	Serialize()	1802
39.417.3.11	Train() [1/2]	1802
39.417.3.12	Train() [2/2]	1803
39.418	LogisticRegressionFunction< MatType > Class Template Reference	1803
39.418.1	Detailed Description	1805
39.418.2	Constructor & Destructor Documentation	1805
39.418.2.1	LogisticRegressionFunction() [1/2]	1805
39.418.2.2	LogisticRegressionFunction() [2/2]	1805
39.418.3	Member Function Documentation	1806
39.418.3.1	Evaluate() [1/2]	1806
39.418.3.2	Evaluate() [2/2]	1806
39.418.3.3	EvaluateWithGradient() [1/2]	1807
39.418.3.4	EvaluateWithGradient() [2/2]	1807
39.418.3.5	GetInitialPoint()	1807
39.418.3.6	Gradient() [1/2]	1807
39.418.3.7	Gradient() [2/2]	1808
39.418.3.8	InitialPoint() [1/2]	1808
39.418.3.9	InitialPoint() [2/2]	1808
39.418.3.10	Lambda() [1/2]	1809
39.418.3.11	Lambda() [2/2]	1809
39.418.3.12	NumFeatures()	1809
39.418.3.13	NumFunctions()	1809
39.418.3.14	PartialGradient()	1809
39.418.3.15	Predictors()	1810
39.418.3.16	Responses()	1810

39.418.3.1	Shuffle()	1810
39.419	SoftmaxRegression Class Reference	1811
39.419.1	Detailed Description	1812
39.419.2	Constructor & Destructor Documentation	1812
39.419.2.1	SoftmaxRegression() [1/2]	1812
39.419.2.2	SoftmaxRegression() [2/2]	1813
39.419.3	Member Function Documentation	1813
39.419.3.1	Classify() [1/4]	1813
39.419.3.2	Classify() [2/4]	1814
39.419.3.3	Classify() [3/4]	1814
39.419.3.4	Classify() [4/4]	1815
39.419.3.5	ComputeAccuracy()	1815
39.419.3.6	FeatureSize()	1815
39.419.3.7	FitIntercept()	1816
39.419.3.8	Lambda() [1/2]	1816
39.419.3.9	Lambda() [2/2]	1816
39.419.3.10	NumClasses() [1/2]	1816
39.419.3.11	NumClasses() [2/2]	1817
39.419.3.12	Parameters() [1/2]	1817
39.419.3.13	Parameters() [2/2]	1817
39.419.3.14	Serialize()	1817
39.419.3.15	Train()	1817
39.420	SoftmaxRegressionFunction Class Reference	1818
39.420.1	Detailed Description	1819
39.420.2	Constructor & Destructor Documentation	1819
39.420.2.1	SoftmaxRegressionFunction()	1819
39.420.3	Member Function Documentation	1820
39.420.3.1	Evaluate() [1/2]	1820

39.420.3.2	Evaluate() [2/2]	1820
39.420.3.3	FitIntercept()	1821
39.420.3.4	GetGroundTruthMatrix()	1821
39.420.3.5	GetInitialPoint()	1821
39.420.3.6	GetProbabilitiesMatrix()	1821
39.420.3.7	Gradient() [1/2]	1822
39.420.3.8	Gradient() [2/2]	1822
39.420.3.9	InitializeWeights() [1/3]	1823
39.420.3.10	InitializeWeights() [2/3]	1823
39.420.3.11	InitializeWeights() [3/3]	1823
39.420.3.12	Lambda() [1/2]	1824
39.420.3.13	Lambda() [2/2]	1824
39.420.3.14	NumClasses()	1824
39.420.3.15	NumFeatures()	1824
39.420.3.16	PartialGradient()	1824
39.420.3.17	Shuffle()	1825
39.421	Acrobot Class Reference	1825
39.421.1	Detailed Description	1826
39.421.2	Member Enumeration Documentation	1826
39.421.2.1	Action	1826
39.421.3	Constructor & Destructor Documentation	1827
39.421.3.1	Acrobot()	1827
39.421.4	Member Function Documentation	1827
39.421.4.1	Dsdt()	1828
39.421.4.2	InitialSample()	1828
39.421.4.3	IsTerminal()	1828
39.421.4.4	Rk4()	1829
39.421.4.5	Sample() [1/2]	1829

39.421.4.6	Sample() [2/2]	1830
39.421.4.7	Torque()	1830
39.421.4.8	Wrap()	1831
39.422	Acrobot::State Class Reference	1831
39.422.1	Detailed Description	1832
39.422.2	Constructor & Destructor Documentation	1832
39.422.2.1	State() [1/2]	1832
39.422.2.2	State() [2/2]	1832
39.422.3	Member Function Documentation	1833
39.422.3.1	AngularVelocity1() [1/2]	1833
39.422.3.2	AngularVelocity1() [2/2]	1833
39.422.3.3	AngularVelocity2() [1/2]	1833
39.422.3.4	AngularVelocity2() [2/2]	1834
39.422.3.5	Data()	1834
39.422.3.6	Encode()	1834
39.422.3.7	Theta1() [1/2]	1834
39.422.3.8	Theta1() [2/2]	1834
39.422.3.9	Theta2() [1/2]	1835
39.422.3.10	Theta2() [2/2]	1835
39.422.4	Member Data Documentation	1835
39.422.4.1	dimension	1835
39.423	AggregatedPolicy< PolicyType > Class Template Reference	1835
39.423.1	Detailed Description	1835
39.423.2	Member Typedef Documentation	1836
39.423.2.1	ActionType	1836
39.423.3	Constructor & Destructor Documentation	1836
39.423.3.1	AggregatedPolicy()	1836
39.423.4	Member Function Documentation	1836

39.423.4.1	Anneal()	1837
39.423.4.2	Sample()	1837
39.424	AsyncLearning< WorkerType, EnvironmentType, NetworkType, UpdaterType, PolicyType > Class Template Reference	1837
39.424.1	Detailed Description	1838
39.424.2	Constructor & Destructor Documentation	1839
39.424.2.1	AsyncLearning()	1839
39.424.3	Member Function Documentation	1839
39.424.3.1	Config() [1/2]	1839
39.424.3.2	Config() [2/2]	1840
39.424.3.3	Environment() [1/2]	1840
39.424.3.4	Environment() [2/2]	1840
39.424.3.5	Network() [1/2]	1840
39.424.3.6	Network() [2/2]	1840
39.424.3.7	Policy() [1/2]	1841
39.424.3.8	Policy() [2/2]	1841
39.424.3.9	Train()	1841
39.424.3.10	Updater() [1/2]	1841
39.424.3.11	Updater() [2/2]	1842
39.425	CartPole Class Reference	1842
39.425.1	Detailed Description	1843
39.425.2	Member Enumeration Documentation	1843
39.425.2.1	Action	1843
39.425.3	Constructor & Destructor Documentation	1843
39.425.3.1	CartPole()	1843
39.425.4	Member Function Documentation	1844
39.425.4.1	InitialSample()	1844
39.425.4.2	IsTerminal()	1844

39.425.4.3	Sample() [1/2]	1845
39.425.4.4	Sample() [2/2]	1845
39.426	CartPole::State Class Reference	1846
39.426.1	Detailed Description	1847
39.426.2	Constructor & Destructor Documentation	1847
39.426.2.1	State() [1/2]	1847
39.426.2.2	State() [2/2]	1847
39.426.3	Member Function Documentation	1848
39.426.3.1	Angle() [1/2]	1848
39.426.3.2	Angle() [2/2]	1848
39.426.3.3	AngularVelocity() [1/2]	1848
39.426.3.4	AngularVelocity() [2/2]	1848
39.426.3.5	Data()	1849
39.426.3.6	Encode()	1849
39.426.3.7	Position() [1/2]	1849
39.426.3.8	Position() [2/2]	1849
39.426.3.9	Velocity() [1/2]	1850
39.426.3.10	Velocity() [2/2]	1850
39.426.4	Member Data Documentation	1850
39.426.4.1	dimension	1850
39.427	ContinuousMountainCar Class Reference	1850
39.427.1	Detailed Description	1851
39.427.2	Constructor & Destructor Documentation	1851
39.427.2.1	ContinuousMountainCar()	1851
39.427.3	Member Function Documentation	1852
39.427.3.1	InitialSample()	1852
39.427.3.2	IsTerminal()	1852
39.427.3.3	Sample() [1/2]	1853

39.427.3.4Sample() [2/2]	1853
39.428ContinuousMountainCar::Action Struct Reference	1854
39.428.1Detailed Description	1854
39.428.2Member Data Documentation	1854
39.428.2.1action	1855
39.428.2.2size	1855
39.429ContinuousMountainCar::State Class Reference	1855
39.429.1Detailed Description	1856
39.429.2Constructor & Destructor Documentation	1856
39.429.2.1State() [1/2]	1856
39.429.2.2State() [2/2]	1856
39.429.3Member Function Documentation	1856
39.429.3.1Data()	1857
39.429.3.2Encode()	1857
39.429.3.3Position() [1/2]	1857
39.429.3.4Position() [2/2]	1857
39.429.3.5Velocity() [1/2]	1858
39.429.3.6Velocity() [2/2]	1858
39.429.4Member Data Documentation	1858
39.429.4.1dimension	1858
39.430GreedyPolicy< EnvironmentType > Class Template Reference	1858
39.430.1Detailed Description	1859
39.430.2Member Typedef Documentation	1859
39.430.2.1ActionType	1859
39.430.3Constructor & Destructor Documentation	1859
39.430.3.1GreedyPolicy()	1860
39.430.4Member Function Documentation	1860
39.430.4.1Anneal()	1860

39.430.4.2Epsilon()	1860
39.430.4.3Sample()	1860
39.43MountainCar Class Reference	1861
39.431.1Detailed Description	1862
39.431.2Member Enumeration Documentation	1862
39.431.2.1Action	1862
39.431.3Constructor & Destructor Documentation	1862
39.431.3.1MountainCar()	1862
39.431.4Member Function Documentation	1863
39.431.4.1InitialSample()	1863
39.431.4.2sTerminal()	1863
39.431.4.3Sample() [1/2]	1864
39.431.4.4Sample() [2/2]	1864
39.432MountainCar::State Class Reference	1865
39.432.1Detailed Description	1866
39.432.2Constructor & Destructor Documentation	1866
39.432.2.1State() [1/2]	1866
39.432.2.2State() [2/2]	1866
39.432.3Member Function Documentation	1866
39.432.3.1Data()	1866
39.432.3.2Encode()	1867
39.432.3.3Position() [1/2]	1867
39.432.3.4Position() [2/2]	1867
39.432.3.5Velocity() [1/2]	1867
39.432.3.6Velocity() [2/2]	1868
39.432.4Member Data Documentation	1868
39.432.4.1dimension	1868

39.432	StepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType > Class Template Reference	1868
39.433.1	Detailed Description	1869
39.433.2	Member Typedef Documentation	1869
39.433.2.1	ActionType	1869
39.433.2.2	StateType	1869
39.433.2.3	TransitionType	1869
39.433.3	Constructor & Destructor Documentation	1870
39.433.3.1	INStepQLearningWorker()	1870
39.433.4	Member Function Documentation	1870
39.433.4.1	Initialize()	1870
39.433.4.2	Step()	1871
39.434	OneStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType > Class Template Reference	1871
39.434.1	Detailed Description	1872
39.434.2	Member Typedef Documentation	1872
39.434.2.1	ActionType	1872
39.434.2.2	StateType	1873
39.434.2.3	TransitionType	1873
39.434.3	Constructor & Destructor Documentation	1873
39.434.3.1	OneStepQLearningWorker()	1873
39.434.4	Member Function Documentation	1873
39.434.4.1	Initialize()	1874
39.434.4.2	Step()	1874
39.435	OneStepSarsaWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType > Class Template Reference	1875
39.435.1	Detailed Description	1875
39.435.2	Member Typedef Documentation	1875
39.435.2.1	ActionType	1876

39.435.2.2	StateType	1876
39.435.2.3	TransitionType	1876
39.435.3	Constructor & Destructor Documentation	1876
39.435.3.1	OneStepSarsaWorker()	1876
39.435.4	Member Function Documentation	1877
39.435.4.1	Initialize()	1877
39.435.4.2	Step()	1877
39.436	Pendulum Class Reference	1878
39.436.1	Detailed Description	1878
39.436.2	Constructor & Destructor Documentation	1878
39.436.2.1	Pendulum()	1878
39.436.3	Member Function Documentation	1879
39.436.3.1	AngleNormalize()	1879
39.436.3.2	InitialSample()	1879
39.436.3.3	Sample() [1/2]	1880
39.436.3.4	Sample() [2/2]	1880
39.437	Pendulum::Action Struct Reference	1881
39.437.1	Detailed Description	1881
39.437.2	Member Data Documentation	1881
39.437.2.1	faction	1881
39.437.2.2	size	1882
39.438	Pendulum::State Class Reference	1882
39.438.1	Detailed Description	1882
39.438.2	Constructor & Destructor Documentation	1883
39.438.2.1	State() [1/2]	1883
39.438.2.2	State() [2/2]	1883
39.438.3	Member Function Documentation	1883
39.438.3.1	AngularVelocity() [1/2]	1883

39.438.3.2AngularVelocity() [2/2]	1884
39.438.3.3Data()	1884
39.438.3.4Encode()	1884
39.438.3.5Theta() [1/2]	1884
39.438.3.6Theta() [2/2]	1884
39.438.4Member Data Documentation	1885
39.438.4.1dimension	1885
39.439QLearning< EnvironmentType, NetworkType, UpdaterType, PolicyType, ReplayType > Class Template Reference	1885
39.439.1Detailed Description	1886
39.439.2Member Typedef Documentation	1886
39.439.2.1ActionType	1887
39.439.2.2StateType	1887
39.439.3Constructor & Destructor Documentation	1887
39.439.3.1QLearning()	1887
39.439.4Member Function Documentation	1888
39.439.4.1Deterministic() [1/2]	1888
39.439.4.2Deterministic() [2/2]	1888
39.439.4.3Environment() [1/2]	1888
39.439.4.4Environment() [2/2]	1888
39.439.4.5Episode()	1889
39.439.4.6Network() [1/2]	1889
39.439.4.7Network() [2/2]	1889
39.439.4.8State() [1/2]	1889
39.439.4.9State() [2/2]	1890
39.439.4.10Step()	1890
39.439.4.11TotalSteps()	1890
39.440RandomReplay< EnvironmentType > Class Template Reference	1890

39.440.1	Detailed Description	1891
39.440.2	Member Typedef Documentation	1891
39.440.2.1	ActionType	1891
39.440.2.2	StateType	1892
39.440.3	Constructor & Destructor Documentation	1892
39.440.3.1	RandomReplay()	1892
39.440.4	Member Function Documentation	1892
39.440.4.1	Sample()	1892
39.440.4.2	Size()	1893
39.440.4.3	Store()	1893
39.44	RewardClipping< EnvironmentType > Class Template Reference	1894
39.441.1	Detailed Description	1895
39.441.2	Member Typedef Documentation	1896
39.441.2.1	Action	1896
39.441.2.2	State	1896
39.441.3	Constructor & Destructor Documentation	1896
39.441.3.1	RewardClipping()	1896
39.441.4	Member Function Documentation	1897
39.441.4.1	Environment() [1/2]	1897
39.441.4.2	Environment() [2/2]	1897
39.441.4.3	InitialSample()	1897
39.441.4.4	IsTerminal()	1897
39.441.4.5	MaxReward() [1/2]	1898
39.441.4.6	MaxReward() [2/2]	1898
39.441.4.7	MinReward() [1/2]	1898
39.441.4.8	MinReward() [2/2]	1899
39.441.4.9	Sample() [1/2]	1899
39.441.4.10	Sample() [2/2]	1900

39.442	TrainingConfig Class Reference	1901
39.442.1	Detailed Description	1901
39.442.2	Constructor & Destructor Documentation	1902
39.442.2.1	TrainingConfig() [1/2]	1902
39.442.2.2	TrainingConfig() [2/2]	1902
39.442.3	Member Function Documentation	1902
39.442.3.1	Discount() [1/2]	1902
39.442.3.2	Discount() [2/2]	1903
39.442.3.3	DoubleQLearning() [1/2]	1903
39.442.3.4	DoubleQLearning() [2/2]	1903
39.442.3.5	ExplorationSteps() [1/2]	1903
39.442.3.6	ExplorationSteps() [2/2]	1903
39.442.3.7	GradientLimit() [1/2]	1904
39.442.3.8	GradientLimit() [2/2]	1904
39.442.3.9	NumWorkers() [1/2]	1904
39.442.3.10	NumWorkers() [2/2]	1904
39.442.3.11	StepLimit() [1/2]	1905
39.442.3.12	StepLimit() [2/2]	1905
39.442.3.13	StepSize() [1/2]	1905
39.442.3.14	StepSize() [2/2]	1905
39.442.3.15	TargetNetworkSyncInterval() [1/2]	1906
39.442.3.16	TargetNetworkSyncInterval() [2/2]	1906
39.442.3.17	UpdateInterval() [1/2]	1906
39.442.3.18	UpdateInterval() [2/2]	1906
39.443	MethodFormDetector< Class, MethodForm, AdditionalArgsCount > Struct Template Reference	1907
39.443.1	Detailed Description	1907
39.444	MethodFormDetector< Class, MethodForm, 0 > Struct Template Reference	1907
39.444.1	Detailed Description	1907

39.444.2	Member Function Documentation	1907
39.444.2.1	operator>()	1907
39.445	MethodFormDetector< Class, MethodForm, 1 > Struct Template Reference	1908
39.445.1	Detailed Description	1908
39.445.2	Member Function Documentation	1908
39.445.2.1	operator>()	1908
39.446	MethodFormDetector< Class, MethodForm, 2 > Struct Template Reference	1908
39.446.1	Detailed Description	1908
39.446.2	Member Function Documentation	1909
39.446.2.1	operator>()	1909
39.447	MethodFormDetector< Class, MethodForm, 3 > Struct Template Reference	1909
39.447.1	Detailed Description	1909
39.447.2	Member Function Documentation	1909
39.447.2.1	operator>()	1909
39.448	MethodFormDetector< Class, MethodForm, 4 > Struct Template Reference	1910
39.448.1	Detailed Description	1910
39.448.2	Member Function Documentation	1910
39.448.2.1	operator>()	1910
39.449	MethodFormDetector< Class, MethodForm, 5 > Struct Template Reference	1910
39.449.1	Detailed Description	1910
39.449.2	Member Function Documentation	1911
39.449.2.1	operator>()	1911
39.450	MethodFormDetector< Class, MethodForm, 6 > Struct Template Reference	1911
39.450.1	Detailed Description	1911
39.450.2	Member Function Documentation	1911
39.450.2.1	operator>()	1911
39.451	MethodFormDetector< Class, MethodForm, 7 > Struct Template Reference	1912
39.451.1	Detailed Description	1912

39.451.2	Member Function Documentation	1912
39.451.2.1	operator>()	1912
39.452	sigCheck< U, U > Struct Template Reference	1912
39.452.1	Detailed Description	1913
39.453	DataDependentRandomInitializer Class Reference	1913
39.453.1	Detailed Description	1913
39.453.2	Member Function Documentation	1913
39.453.2.1	Initialize()	1913
39.454	NothingInitializer Class Reference	1914
39.454.1	Detailed Description	1914
39.454.2	Member Function Documentation	1914
39.454.2.1	Initialize()	1915
39.455	RandomInitializer Class Reference	1915
39.455.1	Detailed Description	1915
39.455.2	Member Function Documentation	1915
39.455.2.1	Initialize()	1915
39.456	SparseCoding Class Reference	1916
39.456.1	Detailed Description	1917
39.456.2	Constructor & Destructor Documentation	1918
39.456.2.1	SparseCoding() [1/2]	1919
39.456.2.2	SparseCoding() [2/2]	1919
39.456.3	Member Function Documentation	1920
39.456.3.1	Atoms() [1/2]	1920
39.456.3.2	Atoms() [2/2]	1920
39.456.3.3	Dictionary() [1/2]	1920
39.456.3.4	Dictionary() [2/2]	1921
39.456.3.5	Encode()	1921
39.456.3.6	Lambda1() [1/2]	1921

39.456.3.7	<code>Lambda1()</code> [2/2]	1921
39.456.3.8	<code>Lambda2()</code> [1/2]	1922
39.456.3.9	<code>Lambda2()</code> [2/2]	1922
39.456.3.10	<code>MaxIterations()</code> [1/2]	1922
39.456.3.11	<code>MaxIterations()</code> [2/2]	1922
39.456.3.12	<code>NewtonTolerance()</code> [1/2]	1922
39.456.3.13	<code>NewtonTolerance()</code> [2/2]	1923
39.456.3.14	<code>Objective()</code>	1923
39.456.3.15	<code>ObjTolerance()</code> [1/2]	1923
39.456.3.16	<code>ObjTolerance()</code> [2/2]	1923
39.456.3.17	<code>OptimizeDictionary()</code>	1923
39.456.3.18	<code>ProjectDictionary()</code>	1924
39.456.3.19	<code>Serialize()</code>	1924
39.456.3.20	<code>Train()</code>	1924
39.457	<code>BiasSVD< OptimizerType > Class Template Reference</code>	1925
39.457.1	Detailed Description	1925
39.457.2	Constructor & Destructor Documentation	1925
39.457.2.1	<code>BiasSVD()</code>	1925
39.457.3	Member Function Documentation	1926
39.457.3.1	<code>Apply()</code>	1926
39.458	<code>BiasSVDFunction< MatType > Class Template Reference</code>	1926
39.458.1	Detailed Description	1927
39.458.2	Constructor & Destructor Documentation	1928
39.458.2.1	<code>BiasSVDFunction()</code>	1928
39.458.3	Member Function Documentation	1928
39.458.3.1	<code>Dataset()</code>	1928
39.458.3.2	<code>Evaluate()</code> [1/2]	1928
39.458.3.3	<code>Evaluate()</code> [2/2]	1929

39.458.3.4	GetInitialPoint()	1929
39.458.3.5	Gradient() [1/2]	1929
39.458.3.6	Gradient() [2/2]	1930
39.458.3.7	Lambda()	1930
39.458.3.8	NumFunctions()	1931
39.458.3.9	NumItems()	1931
39.458.3.10	NumUsers()	1931
39.458.3.11	Rank()	1931
39.458.3.12	Shuffle()	1931
39.459	QUIC_SVD Class Reference	1932
39.459.1	Detailed Description	1932
39.459.2	Constructor & Destructor Documentation	1932
39.459.2.1	QUIC_SVD()	1933
39.459.3	Member Function Documentation	1933
39.459.3.1	ExtractSVD()	1933
39.460	RandomizedBlockKrylovSVD Class Reference	1934
39.460.1	Detailed Description	1934
39.460.2	Constructor & Destructor Documentation	1935
39.460.2.1	RandomizedBlockKrylovSVD() [1/2]	1935
39.460.2.2	RandomizedBlockKrylovSVD() [2/2]	1935
39.460.3	Member Function Documentation	1936
39.460.3.1	Apply()	1936
39.460.3.2	BlockSize() [1/2]	1936
39.460.3.3	BlockSize() [2/2]	1937
39.460.3.4	MaxIterations() [1/2]	1937
39.460.3.5	MaxIterations() [2/2]	1937
39.461	RandomizedSVD Class Reference	1937
39.461.1	Detailed Description	1938

39.461.2	Constructor & Destructor Documentation	1939
39.461.2.1	RandomizedSVD() [1/2]	1939
39.461.2.2	RandomizedSVD() [2/2]	1940
39.461.3	Member Function Documentation	1940
39.461.3.1	Apply() [1/3]	1940
39.461.3.2	Apply() [2/3]	1940
39.461.3.3	Apply() [3/3]	1941
39.461.3.4	Epsilon() [1/2]	1941
39.461.3.5	Epsilon() [2/2]	1942
39.461.3.6	IteratedPower() [1/2]	1942
39.461.3.7	IteratedPower() [2/2]	1942
39.461.3.8	MaxIterations() [1/2]	1942
39.461.3.9	MaxIterations() [2/2]	1943
39.462	RegularizedSVD< OptimizerType > Class Template Reference	1943
39.462.1	Detailed Description	1943
39.462.2	Constructor & Destructor Documentation	1944
39.462.2.1	RegularizedSVD()	1944
39.462.3	Member Function Documentation	1944
39.462.3.1	Apply()	1944
39.463	RegularizedSVDFunction< MatType > Class Template Reference	1945
39.463.1	Detailed Description	1945
39.463.2	Constructor & Destructor Documentation	1946
39.463.2.1	RegularizedSVDFunction()	1946
39.463.3	Member Function Documentation	1946
39.463.3.1	Dataset()	1946
39.463.3.2	Evaluate() [1/2]	1947
39.463.3.3	Evaluate() [2/2]	1948
39.463.3.4	GetInitialPoint()	1948

39.463.3.5	Gradient() [1/2]	1948
39.463.3.6	Gradient() [2/2]	1949
39.463.3.7	Lambda()	1949
39.463.3.8	NumFunctions()	1950
39.463.3.9	NumItems()	1950
39.463.3.10	NumUsers()	1950
39.463.3.11	Rank()	1950
39.463.3.12	Shuffle()	1950
39.464	SVDPlusPlus< OptimizerType > Class Template Reference	1951
39.464.1	Detailed Description	1951
39.464.2	Constructor & Destructor Documentation	1952
39.464.2.1	SVDPlusPlus()	1952
39.464.3	Member Function Documentation	1952
39.464.3.1	Apply() [1/2]	1952
39.464.3.2	Apply() [2/2]	1953
39.464.3.3	CleanData()	1953
39.465	SVDPlusPlusFunction< MatType > Class Template Reference	1954
39.465.1	Detailed Description	1955
39.465.2	Constructor & Destructor Documentation	1955
39.465.2.1	SVDPlusPlusFunction()	1955
39.465.3	Member Function Documentation	1955
39.465.3.1	Dataset()	1956
39.465.3.2	Evaluate() [1/2]	1956
39.465.3.3	Evaluate() [2/2]	1956
39.465.3.4	GetInitialPoint()	1956
39.465.3.5	Gradient() [1/2]	1957
39.465.3.6	Gradient() [2/2]	1957
39.465.3.7	ImplicitDataset()	1958

39.465.3.8	<code>Lambda()</code>	1958
39.465.3.9	<code>NumFunctions()</code>	1958
39.465.3.10	<code>NumItems()</code>	1958
39.465.3.11	<code>NumUsers()</code>	1958
39.465.3.12	<code>Rank()</code>	1959
39.465.3.13	<code>Shuffle()</code>	1959
39.466	<code>LinearSVM< MatType > Class Template Reference</code>	1959
39.466.1	Detailed Description	1960
39.466.2	Constructor & Destructor Documentation	1961
39.466.2.1	<code>LinearSVM()</code> [1/2]	1961
39.466.2.2	<code>LinearSVM()</code> [2/2]	1962
39.466.3	Member Function Documentation	1962
39.466.3.1	<code>Classify()</code> [1/4]	1962
39.466.3.2	<code>Classify()</code> [2/4]	1963
39.466.3.3	<code>Classify()</code> [3/4]	1963
39.466.3.4	<code>Classify()</code> [4/4]	1964
39.466.3.5	<code>ComputeAccuracy()</code>	1964
39.466.3.6	<code>FeatureSize()</code>	1964
39.466.3.7	<code>Lambda()</code> [1/2]	1965
39.466.3.8	<code>Lambda()</code> [2/2]	1965
39.466.3.9	<code>NumClasses()</code> [1/2]	1965
39.466.3.10	<code>NumClasses()</code> [2/2]	1965
39.466.3.11	<code>Parameters()</code> [1/2]	1965
39.466.3.12	<code>Parameters()</code> [2/2]	1966
39.466.3.13	<code>Serialize()</code>	1966
39.466.3.14	<code>Train()</code>	1966
39.467	<code>LinearSVMFunction< MatType > Class Template Reference</code>	1967
39.467.1	Detailed Description	1968

39.467.2	Constructor & Destructor Documentation	1968
39.467.2.1	LinearSVMFunction()	1968
39.467.3	Member Function Documentation	1969
39.467.3.1	Dataset() [1/2]	1969
39.467.3.2	Dataset() [2/2]	1969
39.467.3.3	Evaluate() [1/2]	1969
39.467.3.4	Evaluate() [2/2]	1970
39.467.3.5	EvaluateWithGradient() [1/2]	1970
39.467.3.6	EvaluateWithGradient() [2/2]	1971
39.467.3.7	FitIntercept()	1971
39.467.3.8	GetGroundTruthMatrix()	1972
39.467.3.9	Gradient() [1/2]	1972
39.467.3.10	Gradient() [2/2]	1972
39.467.3.11	InitializeWeights()	1973
39.467.3.12	InitialPoint() [1/2]	1973
39.467.3.13	InitialPoint() [2/2]	1974
39.467.3.14	Lambda() [1/2]	1974
39.467.3.15	Lambda() [2/2]	1974
39.467.3.16	NumFunctions()	1974
39.467.3.17	Shuffle()	1974
39.468	Timer Class Reference	1975
39.468.1	Detailed Description	1975
39.468.2	Member Function Documentation	1975
39.468.2.1	DisableTiming()	1975
39.468.2.2	EnableTiming()	1976
39.468.2.3	Get()	1976
39.468.2.4	ResetAll()	1976
39.468.2.5	Start()	1976

39.468.2.6	Stop()	1977
39.469	Timers Class Reference	1977
39.469.1	Detailed Description	1978
39.469.2	Constructor & Destructor Documentation	1978
39.469.2.1	Timers()	1978
39.469.3	Member Function Documentation	1978
39.469.3.1	Enabled() [1/2]	1978
39.469.3.2	Enabled() [2/2]	1979
39.469.3.3	GetAllTimers()	1979
39.469.3.4	GetState()	1979
39.469.3.5	GetTimer()	1979
39.469.3.6	PrintTimer()	1980
39.469.3.7	Reset()	1980
39.469.3.8	StartTimer()	1980
39.469.3.9	StopAllTimers()	1981
39.469.3.10	StopTimer()	1981
39.470	AllCategoricalSplit< FitnessFunction > Class Template Reference	1981
39.470.1	Detailed Description	1982
39.470.2	Member Function Documentation	1982
39.470.2.1	CalculateDirection()	1982
39.470.2.2	NumChildren()	1983
39.470.2.3	SplitIfBetter()	1983
39.471	AllCategoricalSplit< FitnessFunction >::AuxiliarySplitInfo< ElemType > Class Template Reference	1984
39.471.1	Detailed Description	1984
39.472	AllDimensionSelect Class Reference	1984
39.472.1	Detailed Description	1985
39.472.2	Constructor & Destructor Documentation	1985
39.472.2.1	AllDimensionSelect()	1985

39.472.3	Member Function Documentation	1985
39.472.3.1	Begin()	1985
39.472.3.2	Dimensions() [1/2]	1985
39.472.3.3	Dimensions() [2/2]	1986
39.472.3.4	End()	1986
39.472.3.5	Next()	1986
39.473	AxisParallelProjVector Class Reference	1986
39.473.1	Detailed Description	1987
39.473.2	Constructor & Destructor Documentation	1987
39.473.2.1	AxisParallelProjVector()	1987
39.473.3	Member Function Documentation	1988
39.473.3.1	Project() [1/3]	1988
39.473.3.2	Project() [2/3]	1988
39.473.3.3	Project() [3/3]	1988
39.473.3.4	Serialize()	1989
39.474	BestBinaryNumericSplit< FitnessFunction > Class Template Reference	1989
39.474.1	Detailed Description	1990
39.474.2	Member Function Documentation	1990
39.474.2.1	CalculateDirection()	1990
39.474.2.2	NumChildren()	1990
39.474.2.3	SplitIfBetter()	1991
39.475	BestBinaryNumericSplit< FitnessFunction >::AuxiliarySplitInfo< ElemType > Class Template Reference	1991
39.475.1	Detailed Description	1991
39.476	BinaryNumericSplit< FitnessFunction, ObservationType > Class Template Reference	1992
39.476.1	Detailed Description	1993
39.476.2	Member Typedef Documentation	1993
39.476.2.1	SplitInfo	1993
39.476.3	Constructor & Destructor Documentation	1993

39.476.3.1	BinaryNumericSplit() [1/2]	1993
39.476.3.2	BinaryNumericSplit() [2/2]	1994
39.476.4	Member Function Documentation	1994
39.476.4.1	EvaluateFitnessFunction()	1994
39.476.4.2	MajorityClass()	1994
39.476.4.3	MajorityProbability()	1995
39.476.4.4	NumChildren()	1995
39.476.4.5	Serialize()	1995
39.476.4.6	Split()	1995
39.476.4.7	Train()	1996
39.476	BinaryNumericSplitInfo< ObservationType > Class Template Reference	1996
39.477.1	Detailed Description	1996
39.477.2	Constructor & Destructor Documentation	1997
39.477.2.1	BinaryNumericSplitInfo() [1/2]	1997
39.477.2.2	BinaryNumericSplitInfo() [2/2]	1997
39.477.3	Member Function Documentation	1997
39.477.3.1	CalculateDirection()	1997
39.477.3.2	Serialize()	1997
39.478	BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > Class Template Reference	1998
39.478.1	Detailed Description	2001
39.478.2	Member Typedef Documentation	2002
39.478.2.1	ElemType	2002
39.478.2.2	Mat	2002
39.478.2.3	Split	2002
39.478.3	Constructor & Destructor Documentation	2002
39.478.3.1	BinarySpaceTree() [1/13]	2002
39.478.3.2	BinarySpaceTree() [2/13]	2003

39.478.3.3	BinarySpaceTree()	[3/13]	2003
39.478.3.4	BinarySpaceTree()	[4/13]	2004
39.478.3.5	BinarySpaceTree()	[5/13]	2004
39.478.3.6	BinarySpaceTree()	[6/13]	2004
39.478.3.7	BinarySpaceTree()	[7/13]	2005
39.478.3.8	BinarySpaceTree()	[8/13]	2005
39.478.3.9	BinarySpaceTree()	[9/13]	2006
39.478.3.10	BinarySpaceTree()	[10/13]	2007
39.478.3.11	BinarySpaceTree()	[11/13]	2007
39.478.3.12	BinarySpaceTree()	[12/13]	2007
39.478.3.13	BinarySpaceTree()		2007
39.478.3.14	BinarySpaceTree()	[13/13]	2008
39.478.4	Member Function Documentation		2008
39.478.4.1	Begin()	[1/2]	2008
39.478.4.2	Begin()	[2/2]	2008
39.478.4.3	Bound()	[1/2]	2009
39.478.4.4	Bound()	[2/2]	2009
39.478.4.5	Center()		2009
39.478.4.6	Child()		2009
39.478.4.7	ChildPtr()		2010
39.478.4.8	Count()	[1/2]	2010
39.478.4.9	Count()	[2/2]	2010
39.478.4.10	Dataset()	[1/2]	2011
39.478.4.11	Dataset()	[2/2]	2011
39.478.4.12	Descendant()		2011
39.478.4.13	FurthestDescendantDistance()		2011
39.478.4.14	FurthestPointDistance()		2012
39.478.4.15	GetFurthestChild()	[1/2]	2012

39.478.4.16	GetFurthestChild() [2/2]	2012
39.478.4.17	GetNearestChild() [1/2]	2012
39.478.4.18	GetNearestChild() [2/2]	2013
39.478.4.19	Leaf()	2013
39.478.4.20	Left() [1/2]	2013
39.478.4.21	Left() [2/2]	2013
39.478.4.22	MaxDistance() [1/2]	2013
39.478.4.23	MaxDistance() [2/2]	2014
39.478.4.24	Metric()	2014
39.478.4.25	MinDistance() [1/2]	2014
39.478.4.26	MinDistance() [2/2]	2014
39.478.4.27	MinimumBoundDistance()	2015
39.478.4.28	NumChildren()	2015
39.478.4.29	NumDescendants()	2015
39.478.4.30	NumPoints()	2015
39.478.4.31	Parent() [1/2]	2016
39.478.4.32	Parent() [2/2]	2016
39.478.4.33	ParentDistance() [1/2]	2016
39.478.4.34	ParentDistance() [2/2]	2016
39.478.4.35	Point()	2016
39.478.4.36	RangeDistance() [1/2]	2017
39.478.4.37	RangeDistance() [2/2]	2017
39.478.4.38	Right() [1/2]	2017
39.478.4.39	Right() [2/2]	2018
39.478.4.40	Serialize()	2018
39.478.4.41	Stat() [1/2]	2018
39.478.4.42	Stat() [2/2]	2018

39.479	BinarySpaceTree < MetricType, StatisticType, MatType, BoundType, SplitType >::BreadthFirstDualTreeTraverser< RuleType > Class Template Reference	2019
39.479.1	Detailed Description	2019
39.479.2	Member Typedef Documentation	2019
39.479.2.1	QueueFrameType	2020
39.479.3	Constructor & Destructor Documentation	2020
39.479.3.1	BreadthFirstDualTreeTraverser()	2020
39.479.4	Member Function Documentation	2020
39.479.4.1	NumBaseCases() [1/2]	2020
39.479.4.2	NumBaseCases() [2/2]	2020
39.479.4.3	NumPrunes() [1/2]	2021
39.479.4.4	NumPrunes() [2/2]	2021
39.479.4.5	NumScores() [1/2]	2021
39.479.4.6	NumScores() [2/2]	2021
39.479.4.7	NumVisited() [1/2]	2021
39.479.4.8	NumVisited() [2/2]	2022
39.479.4.9	Traverse() [1/2]	2022
39.479.4.10	Traverse() [2/2]	2022
39.480	BinarySpaceTree < MetricType, StatisticType, MatType, BoundType, SplitType >::DualTreeTraverser< RuleType > Class Template Reference	2022
39.480.1	Detailed Description	2023
39.480.2	Constructor & Destructor Documentation	2023
39.480.2.1	DualTreeTraverser()	2023
39.480.3	Member Function Documentation	2024
39.480.3.1	NumBaseCases() [1/2]	2024
39.480.3.2	NumBaseCases() [2/2]	2024
39.480.3.3	NumPrunes() [1/2]	2024
39.480.3.4	NumPrunes() [2/2]	2024
39.480.3.5	NumScores() [1/2]	2025

39.480.3.6NumScores() [2/2]	2025
39.480.3.7NumVisited() [1/2]	2025
39.480.3.8NumVisited() [2/2]	2025
39.480.3.9Traverse()	2025
39.481.BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::SingleTreeTraverser< RuleType > Class Template Reference	2026
39.481.1Detailed Description	2026
39.481.2Constructor & Destructor Documentation	2026
39.481.2.1SingleTreeTraverser()	2027
39.481.3Member Function Documentation	2027
39.481.3.1NumPrunes() [1/2]	2027
39.481.3.2NumPrunes() [2/2]	2027
39.481.3.3Traverse()	2027
39.482.CategoricalSplitInfo Class Reference	2028
39.482.1Detailed Description	2028
39.482.2Constructor & Destructor Documentation	2028
39.482.2.1CategoricalSplitInfo()	2028
39.482.3Member Function Documentation	2028
39.482.3.1CalculateDirection()	2029
39.482.3.2Serialize()	2029
39.483.CompareCosineNode Class Reference	2029
39.483.1Detailed Description	2029
39.483.2Member Function Documentation	2029
39.483.2.1operator>()	2030
39.484.CosineTree Class Reference	2030
39.484.1Detailed Description	2031
39.484.2Constructor & Destructor Documentation	2031
39.484.2.1CosineTree() [1/3]	2032

39.484.2.2	CosineTree() [2/3]	2032
39.484.2.3	CosineTree() [3/3]	2032
39.484.2.4	~CosineTree()	2033
39.484.3	Member Function Documentation	2033
39.484.3.1	BasisVector() [1/2]	2033
39.484.3.2	BasisVector() [2/2]	2033
39.484.3.3	BinarySearch()	2033
39.484.3.4	CalculateCentroid()	2034
39.484.3.5	CalculateCosines()	2034
39.484.3.6	Centroid()	2034
39.484.3.7	ColumnSampleLS()	2035
39.484.3.8	ColumnSamplesLS()	2035
39.484.3.9	ConstructBasis()	2035
39.484.3.10	CosineNodeSplit()	2035
39.484.3.11	FrobNormSquared()	2036
39.484.3.12	GetDataset()	2036
39.484.3.13	GetFinalBasis()	2036
39.484.3.14	Error() [1/2]	2036
39.484.3.15	Error() [2/2]	2037
39.484.3.16	Left() [1/2]	2037
39.484.3.17	Left() [2/2]	2037
39.484.3.18	ModifiedGramSchmidt()	2037
39.484.3.19	MonteCarloError()	2038
39.484.3.20	NumColumns()	2038
39.484.3.21	Parent() [1/2]	2038
39.484.3.22	Parent() [2/2]	2039
39.484.3.23	Right() [1/2]	2039
39.484.3.24	Right() [2/2]	2039

39.484.3.2	SplitPointIndex()	2039
39.484.3.2	VectorIndices()	2039
39.485	CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > Class Template Reference	2040
39.485.1	Detailed Description	2043
39.485.2	Member Typedef Documentation	2044
39.485.2.1	BreadthFirstDualTreeTraverser	2044
39.485.2.2	ElemType	2045
39.485.2.3	Mat	2045
39.485.3	Constructor & Destructor Documentation	2045
39.485.3.1	CoverTree() [1/10]	2045
39.485.3.2	CoverTree() [2/10]	2045
39.485.3.3	CoverTree() [3/10]	2046
39.485.3.4	CoverTree() [4/10]	2046
39.485.3.5	CoverTree() [5/10]	2047
39.485.3.6	CoverTree() [6/10]	2048
39.485.3.7	CoverTree() [7/10]	2048
39.485.3.8	CoverTree() [8/10]	2049
39.485.3.9	CoverTree() [9/10]	2049
39.485.3.10	CoverTree()	2049
39.485.3.10	CoverTree() [10/10]	2049
39.485.4	Member Function Documentation	2049
39.485.4.1	Base() [1/2]	2050
39.485.4.2	Base() [2/2]	2050
39.485.4.3	Center()	2050
39.485.4.4	Child() [1/2]	2050
39.485.4.5	Child() [2/2]	2051
39.485.4.6	ChildPtr()	2051
39.485.4.7	Children() [1/2]	2051

39.485.4.8Children() [2/2]	2051
39.485.4.9Dataset()	2052
39.485.4.10Descendant()	2052
39.485.4.11DistanceComps() [1/2]	2052
39.485.4.12DistanceComps() [2/2]	2052
39.485.4.13FurthestDescendantDistance() [1/2]	2052
39.485.4.14FurthestDescendantDistance() [2/2]	2053
39.485.4.15FurthestPointDistance()	2053
39.485.4.16GetFurthestChild() [1/2]	2053
39.485.4.17GetFurthestChild() [2/2]	2053
39.485.4.18GetNearestChild() [1/2]	2054
39.485.4.19GetNearestChild() [2/2]	2054
39.485.4.20Leaf()	2054
39.485.4.21MaxDistance() [1/4]	2054
39.485.4.22MaxDistance() [2/4]	2055
39.485.4.23MaxDistance() [3/4]	2055
39.485.4.24MaxDistance() [4/4]	2055
39.485.4.25Metric()	2055
39.485.4.26MinDistance() [1/4]	2056
39.485.4.27MinDistance() [2/4]	2056
39.485.4.28MinDistance() [3/4]	2056
39.485.4.29MinDistance() [4/4]	2056
39.485.4.30MinimumBoundDistance()	2056
39.485.4.31NumChildren()	2057
39.485.4.32NumDescendants()	2057
39.485.4.33NumPoints()	2057
39.485.4.34Parent() [1/2]	2057
39.485.4.35Parent() [2/2]	2057

39.485.4.36	ParentDistance() [1/2]	2058
39.485.4.37	ParentDistance() [2/2]	2058
39.485.4.38	Point() [1/2]	2058
39.485.4.39	Point() [2/2]	2058
39.485.4.40	RangeDistance() [1/4]	2058
39.485.4.41	RangeDistance() [2/4]	2059
39.485.4.42	RangeDistance() [3/4]	2059
39.485.4.43	RangeDistance() [4/4]	2059
39.485.4.44	Scale() [1/2]	2059
39.485.4.45	Scale() [2/2]	2059
39.485.4.46	Serialize()	2060
39.485.4.47	Stat() [1/2]	2060
39.485.4.48	Stat() [2/2]	2060
39.486	CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::DualTreeTraverser< RuleType > Class Template Reference	2060
39.486.1	Detailed Description	2061
39.486.2	Constructor & Destructor Documentation	2061
39.486.2.1	DualTreeTraverser()	2061
39.486.3	Member Function Documentation	2061
39.486.3.1	NumBaseCases()	2062
39.486.3.2	NumPrunes() [1/2]	2062
39.486.3.3	NumPrunes() [2/2]	2062
39.486.3.4	NumScores()	2062
39.486.3.5	NumVisited()	2062
39.486.3.6	Traverse()	2062
39.487	CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::SingleTreeTraverser< RuleType > Class Template Reference	2063
39.487.1	Detailed Description	2063
39.487.2	Constructor & Destructor Documentation	2063

39.487.2.1	SingleTreeTraverser()	2064
39.487.3	Member Function Documentation	2064
39.487.3.1	NumPrunes() [1/2]	2064
39.487.3.2	NumPrunes() [2/2]	2064
39.487.3.3	Traverse()	2064
39.488	DecisionTree< FitnessFunction, NumericSplitType, CategoricalSplitType, DimensionSelectionType, ElemType, NoRecursion > Class Template Reference	2065
39.488.1	Detailed Description	2067
39.488.2	Member Typedef Documentation	2067
39.488.2.1	CategoricalSplit	2068
39.488.2.2	DimensionSelection	2068
39.488.2.3	NumericSplit	2068
39.488.3	Constructor & Destructor Documentation	2068
39.488.3.1	DecisionTree() [1/7]	2068
39.488.3.2	DecisionTree() [2/7]	2069
39.488.3.3	DecisionTree() [3/7]	2069
39.488.3.4	DecisionTree() [4/7]	2070
39.488.3.5	DecisionTree() [5/7]	2071
39.488.3.6	DecisionTree() [6/7]	2071
39.488.3.7	DecisionTree() [7/7]	2072
39.488.3.8	~DecisionTree()	2072
39.488.4	Member Function Documentation	2072
39.488.4.1	CalculateDirection()	2072
39.488.4.2	Child() [1/2]	2073
39.488.4.3	Child() [2/2]	2073
39.488.4.4	Classify() [1/4]	2073
39.488.4.5	Classify() [2/4]	2073
39.488.4.6	Classify() [3/4]	2074

39.488.4.7Classify() [4 / 4]	2074
39.488.4.8NumChildren()	2075
39.488.4.9NumClasses()	2075
39.488.4.10operator=() [1 / 2]	2075
39.488.4.11operator=() [2 / 2]	2075
39.488.4.12Serialize()	2076
39.488.4.13SplitDimension()	2076
39.488.4.14Train() [1 / 4]	2076
39.488.4.15Train() [2 / 4]	2077
39.488.4.16Train() [3 / 4]	2078
39.488.4.17Train() [4 / 4]	2078
39.489DiscreteHilbertValue< TreeElemType > Class Template Reference	2079
39.489.1Detailed Description	2079
39.490EmptyStatistic Class Reference	2080
39.490.1Detailed Description	2080
39.490.2Constructor & Destructor Documentation	2080
39.490.2.1EmptyStatistic() [1 / 2]	2080
39.490.2.2~EmptyStatistic()	2081
39.490.2.3EmptyStatistic() [2 / 2]	2081
39.490.3Member Function Documentation	2081
39.490.3.1serialize()	2081
39.491ExampleTree< MetricType, StatisticType, MatType > Class Template Reference	2081
39.491.1Detailed Description	2083
39.491.2Constructor & Destructor Documentation	2083
39.491.2.1ExampleTree()	2083
39.491.3Member Function Documentation	2084
39.491.3.1Centroid()	2084
39.491.3.2Child() [1 / 2]	2084

39.491.3.3	Child() [2/2]	2084
39.491.3.4	Descendant()	2085
39.491.3.5	FurthestDescendantDistance()	2085
39.491.3.6	MaxDistance() [1/2]	2085
39.491.3.7	MaxDistance() [2/2]	2085
39.491.3.8	Metric() [1/2]	2086
39.491.3.9	Metric() [2/2]	2086
39.491.3.10	MinDistance() [1/2]	2086
39.491.3.11	MinDistance() [2/2]	2086
39.491.3.12	NumChildren()	2087
39.491.3.13	NumDescendants()	2087
39.491.3.14	NumPoints()	2087
39.491.3.15	Parent()	2087
39.491.3.16	ParentDistance()	2088
39.491.3.17	Point()	2088
39.491.3.18	RangeDistance() [1/2]	2088
39.491.3.19	RangeDistance() [2/2]	2088
39.491.3.20	Stat() [1/2]	2089
39.491.3.21	Stat() [2/2]	2089
39.492	FirstPointsRoot Class Reference	2089
39.492.1	Detailed Description	2090
39.492.2	Member Function Documentation	2090
39.492.2.1	ChooseRoot()	2090
39.493	IniGain Class Reference	2090
39.493.1	Detailed Description	2091
39.493.2	Member Function Documentation	2091
39.493.2.1	Evaluate()	2091
39.493.2.2	EvaluatePtr()	2091

39.493.2.3	Range()	2092
39.494	GreedySingleTreeTraverser< TreeType, RuleType > Class Template Reference	2092
39.494.1	Detailed Description	2092
39.494.2	Member Function Documentation	2092
39.494.2.1	Evaluate()	2092
39.494.2.2	Range()	2093
39.495	GreedySingleTreeTraverser< TreeType, RuleType > Class Template Reference	2093
39.495.1	Detailed Description	2093
39.495.2	Constructor & Destructor Documentation	2093
39.495.2.1	GreedySingleTreeTraverser()	2094
39.495.3	Member Function Documentation	2094
39.495.3.1	MinBaseCases() [1/2]	2094
39.495.3.2	MinBaseCases() [2/2]	2094
39.495.3.3	NumPrunes()	2094
39.495.3.4	Traverse()	2094
39.496	HilbertRTreeAuxiliaryInformation< TreeType, HilbertValueType > Class Template Reference	2095
39.496.1	Detailed Description	2096
39.496.2	Member Typedef Documentation	2096
39.496.2.1	ElemType	2096
39.496.3	Constructor & Destructor Documentation	2096
39.496.3.1	HilbertRTreeAuxiliaryInformation() [1/4]	2096
39.496.3.2	HilbertRTreeAuxiliaryInformation() [2/4]	2096
39.496.3.3	HilbertRTreeAuxiliaryInformation() [3/4]	2097
39.496.3.4	HilbertRTreeAuxiliaryInformation() [4/4]	2097
39.496.4	Member Function Documentation	2097
39.496.4.1	Children()	2097
39.496.4.2	HandleNodeInsertion()	2098
39.496.4.3	HandleNodeRemoval()	2098

39.496.4.4	HandlePointDeletion()	2098
39.496.4.5	HandlePointInsertion()	2099
39.496.4.6	HilbertValue() [1/2]	2099
39.496.4.7	HilbertValue() [2/2]	2099
39.496.4.8	NullifyData()	2100
39.496.4.9	operator=()	2100
39.496.4.10	Serialize()	2100
39.496.4.11	UpdateAuxiliaryInfo()	2100
39.497	HilbertRTreeDescentHeuristic Class Reference	2101
39.497.1	Detailed Description	2101
39.497.2	Member Function Documentation	2101
39.497.2.1	ChooseDescentNode() [1/2]	2101
39.497.2.2	ChooseDescentNode() [2/2]	2102
39.498	HilbertRTreeSplit< splitOrder > Class Template Reference	2102
39.498.1	Detailed Description	2103
39.498.2	Member Function Documentation	2103
39.498.2.1	SplitLeafNode()	2103
39.498.2.2	SplitNonLeafNode()	2103
39.499	HoeffdingCategoricalSplit< FitnessFunction > Class Template Reference	2104
39.499.1	Detailed Description	2105
39.499.2	Member Typedef Documentation	2105
39.499.2.1	SplitInfo	2105
39.499.3	Constructor & Destructor Documentation	2105
39.499.3.1	HoeffdingCategoricalSplit() [1/2]	2105
39.499.3.2	HoeffdingCategoricalSplit() [2/2]	2106
39.499.4	Member Function Documentation	2106
39.499.4.1	EvaluateFitnessFunction()	2106
39.499.4.2	MajorityClass()	2106

39.499.4.3	MajorityProbability()	2107
39.499.4.4	NumChildren()	2107
39.499.4.5	Serialize()	2107
39.499.4.6	Split()	2107
39.499.4.7	Train()	2108
39.500	HoeffdingNumericSplit< FitnessFunction, ObservationType > Class Template Reference	2108
39.500.1	Detailed Description	2109
39.500.2	Member Typedef Documentation	2110
39.500.2.1	SplitInfo	2110
39.500.3	Constructor & Destructor Documentation	2110
39.500.3.1	HoeffdingNumericSplit() [1/2]	2110
39.500.3.2	HoeffdingNumericSplit() [2/2]	2111
39.500.4	Member Function Documentation	2111
39.500.4.1	Bins()	2111
39.500.4.2	EvaluateFitnessFunction()	2111
39.500.4.3	MajorityClass()	2111
39.500.4.4	MajorityProbability()	2112
39.500.4.5	NumChildren()	2112
39.500.4.6	Serialize()	2112
39.500.4.7	Split()	2113
39.500.4.8	Train()	2113
39.501	HoeffdingTree< FitnessFunction, NumericSplitType, CategoricalSplitType > Class Template Reference	2113
39.501.1	Detailed Description	2115
39.501.2	Member Typedef Documentation	2116
39.501.2.1	CategoricalSplit	2116
39.501.2.2	NumericSplit	2116
39.501.3	Constructor & Destructor Documentation	2116
39.501.3.1	HoeffdingTree() [1/4]	2117

39.501.3.2	HoeffdingTree() [2/4]	2117
39.501.3.3	HoeffdingTree() [3/4]	2118
39.501.3.4	HoeffdingTree() [4/4]	2118
39.501.3.5	~HoeffdingTree()	2119
39.501.4	Member Function Documentation	2119
39.501.4.1	CalculateDirection()	2119
39.501.4.2	CheckInterval() [1/2]	2119
39.501.4.3	CheckInterval() [2/2]	2120
39.501.4.4	Child() [1/2]	2120
39.501.4.5	Child() [2/2]	2120
39.501.4.6	Classify() [1/4]	2120
39.501.4.7	Classify() [2/4]	2121
39.501.4.8	Classify() [3/4]	2121
39.501.4.9	Classify() [4/4]	2122
39.501.4.10	CreateChildren()	2122
39.501.4.11	MajorityClass() [1/2]	2122
39.501.4.12	MajorityClass() [2/2]	2122
39.501.4.13	MajorityProbability() [1/2]	2123
39.501.4.14	MajorityProbability() [2/2]	2123
39.501.4.15	MaxSamples() [1/2]	2123
39.501.4.16	MaxSamples() [2/2]	2123
39.501.4.17	MinSamples() [1/2]	2123
39.501.4.18	MinSamples() [2/2]	2124
39.501.4.19	NumChildren()	2124
39.501.4.20	NumDescendants()	2124
39.501.4.21	Serialize()	2124
39.501.4.22	SplitCheck()	2124
39.501.4.23	SplitDimension()	2125

39.501.4.2 3 SuccessProbability() [1/2]	2125
39.501.4.2 5 SuccessProbability() [2/2]	2125
39.501.4.2 7 Train() [1/3]	2125
39.501.4.2 7 Train() [2/3]	2126
39.501.4.2 8 Train() [3/3]	2126
39.502.HoeffdingTreeModel Class Reference	2126
39.502.1Detailed Description	2127
39.502.2Member Typedef Documentation	2128
39.502.2.1GiniBinaryTreeType	2128
39.502.2.2GiniHoeffdingTreeType	2128
39.502.2.3InfoBinaryTreeType	2128
39.502.2.4InfoHoeffdingTreeType	2128
39.502.3Member Enumeration Documentation	2128
39.502.3.1TreeType	2128
39.502.4Constructor & Destructor Documentation	2129
39.502.4.1HoeffdingTreeModel() [1/3]	2129
39.502.4.2HoeffdingTreeModel() [2/3]	2129
39.502.4.3HoeffdingTreeModel() [3/3]	2130
39.502.4.4~HoeffdingTreeModel()	2130
39.502.5Member Function Documentation	2130
39.502.5.1BuildModel()	2130
39.502.5.2Classify() [1/2]	2131
39.502.5.3Classify() [2/2]	2131
39.502.5.4NumNodes()	2132
39.502.5.5operator=() [1/2]	2132
39.502.5.6operator=() [2/2]	2132
39.502.5.7serialize()	2132
39.502.5.8Train()	2133

39.503.1	HyperplaneBase < BoundT, ProjVectorT > Class Template Reference	2133
39.503.1	Detailed Description	2134
39.503.2	Member Typedef Documentation	2134
39.503.2.1	BoundType	2135
39.503.2.2	ProjVectorType	2135
39.503.3	Constructor & Destructor Documentation	2135
39.503.3.1	HyperplaneBase() [1/2]	2135
39.503.3.2	HyperplaneBase() [2/2]	2135
39.503.4	Member Function Documentation	2136
39.503.4.1	Left() [1/2]	2136
39.503.4.2	Left() [2/2]	2136
39.503.4.3	Project()	2136
39.503.4.4	Right() [1/2]	2137
39.503.4.5	Right() [2/2]	2137
39.503.4.6	Serialize()	2138
39.504	InformationGain Class Reference	2138
39.504.1	Detailed Description	2138
39.504.2	Member Function Documentation	2138
39.504.2.1	Evaluate() [1/2]	2138
39.504.2.2	Evaluate() [2/2]	2139
39.504.2.3	EvaluatePtr()	2139
39.504.2.4	Range() [1/2]	2139
39.504.2.5	Range() [2/2]	2140
39.505	SpillTree< TreeType > Struct Template Reference	2140
39.505.1	Detailed Description	2140
39.505.2	Member Data Documentation	2140
39.505.2.1	value	2140

39.506	SpillTree< tree::SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > > Struct Template Reference	2141
39.506.1	Detailed Description	2141
39.506.2	Member Data Documentation	2141
39.506.2.1	value	2141
39.507	MeanSpaceSplit< MetricType, MatType > Class Template Reference	2141
39.507.1	Detailed Description	2142
39.507.2	Member Function Documentation	2142
39.507.2.1	SplitSpace()	2142
39.508	MeanSplit< BoundType, MatType > Class Template Reference	2142
39.508.1	Detailed Description	2143
39.508.2	Member Function Documentation	2143
39.508.2.1	AssignToLeftNode()	2143
39.508.2.2	PerformSplit() [1/2]	2144
39.508.2.3	PerformSplit() [2/2]	2144
39.508.2.4	SplitNode()	2145
39.509	MeanSplit< BoundType, MatType >::SplitInfo Struct Reference	2145
39.509.1	Detailed Description	2146
39.509.2	Member Data Documentation	2146
39.509.2.1	splitDimension	2146
39.509.2.2	splitVal	2146
39.510	MidpointSpaceSplit< MetricType, MatType > Class Template Reference	2146
39.510.1	Detailed Description	2147
39.510.2	Member Function Documentation	2147
39.510.2.1	SplitSpace()	2147
39.511	MidpointSplit< BoundType, MatType > Class Template Reference	2147
39.511.1	Detailed Description	2148
39.511.2	Member Function Documentation	2148

39.511.2.1	AssignToLeftNode()	2148
39.511.2.2	PerformSplit() [1/2]	2149
39.511.2.3	PerformSplit() [2/2]	2149
39.511.2.4	SplitNode()	2150
39.512	MidpointSplit< BoundType, MatType >::SplitInfo Struct Reference	2150
39.512.1	Detailed Description	2151
39.512.2	Member Data Documentation	2151
39.512.2.1	splitDimension	2151
39.512.2.2	splitVal	2151
39.513	MinimalCoverageSweep< SplitPolicy > Class Template Reference	2151
39.513.1	Detailed Description	2152
39.513.2	Member Function Documentation	2152
39.513.2.1	CheckLeafSweep()	2153
39.513.2.2	CheckNonLeafSweep()	2153
39.513.2.3	SweepLeafNode()	2153
39.513.2.4	SweepNonLeafNode()	2154
39.514	MinimalCoverageSweep< SplitPolicy >::SweepCost< TreeType > Struct Template Reference	2154
39.514.1	Detailed Description	2154
39.514.2	Member Typedef Documentation	2155
39.514.2.1	type	2155
39.515	MinimalSplitsNumberSweep< SplitPolicy > Class Template Reference	2155
39.515.1	Detailed Description	2155
39.515.2	Member Function Documentation	2156
39.515.2.1	SweepLeafNode()	2156
39.515.2.2	SweepNonLeafNode()	2156
39.516	MinimalSplitsNumberSweep< SplitPolicy >::SweepCost< typename > Struct Template Reference	2157
39.516.1	Detailed Description	2157
39.516.2	Member Typedef Documentation	2157

39.516.2.1	type	2157
39.517	MultipleRandomDimensionSelect Class Reference	2158
39.517.1	Detailed Description	2158
39.517.2	Constructor & Destructor Documentation	2158
39.517.2.1	MultipleRandomDimensionSelect()	2158
39.517.3	Member Function Documentation	2159
39.517.3.1	Begin()	2159
39.517.3.2	Dimensions() [1/2]	2159
39.517.3.3	Dimensions() [2/2]	2159
39.517.3.4	End()	2159
39.517.3.5	Next()	2160
39.518	NoAuxiliaryInformation< TreeType > Class Template Reference	2160
39.518.1	Detailed Description	2161
39.518.2	Constructor & Destructor Documentation	2161
39.518.2.1	NoAuxiliaryInformation() [1/4]	2161
39.518.2.2	NoAuxiliaryInformation() [2/4]	2161
39.518.2.3	NoAuxiliaryInformation() [3/4]	2161
39.518.2.4	NoAuxiliaryInformation() [4/4]	2162
39.518.3	Member Function Documentation	2162
39.518.3.1	HandleNodeInsertion()	2162
39.518.3.2	HandleNodeRemoval()	2162
39.518.3.3	HandlePointDeletion()	2163
39.518.3.4	HandlePointInsertion()	2163
39.518.3.5	NullifyData()	2164
39.518.3.6	operator=()	2164
39.518.3.7	Serialize()	2164
39.518.3.8	SplitAuxiliaryInfo()	2164
39.518.3.9	UpdateAuxiliaryInfo()	2165

39.519.1	NumericSplitInfo< ObservationType > Class Template Reference	2165
39.519.1	Detailed Description	2166
39.519.2	Constructor & Destructor Documentation	2166
39.519.2.1	NumericSplitInfo() [1/2]	2166
39.519.2.2	NumericSplitInfo() [2/2]	2166
39.519.3	Member Function Documentation	2166
39.519.3.1	CalculateDirection()	2166
39.519.3.2	Serialize()	2167
39.520.1	Octree< MetricType, StatisticType, MatType > Class Template Reference	2167
39.520.1	Detailed Description	2170
39.520.2	Member Typedef Documentation	2170
39.520.2.1	ElemType	2170
39.520.2.2	Mat	2170
39.520.3	Constructor & Destructor Documentation	2170
39.520.3.1	Octree() [1/12]	2170
39.520.3.2	Octree() [2/12]	2171
39.520.3.3	Octree() [3/12]	2171
39.520.3.4	Octree() [4/12]	2172
39.520.3.5	Octree() [5/12]	2172
39.520.3.6	Octree() [6/12]	2172
39.520.3.7	Octree() [7/12]	2173
39.520.3.8	Octree() [8/12]	2173
39.520.3.9	Octree() [9/12]	2174
39.520.3.10	Octree() [10/12]	2174
39.520.3.11	Octree() [11/12]	2175
39.520.3.12	Octree()	2175
39.520.3.13	Octree() [12/12]	2175
39.520.4	Member Function Documentation	2176

39.520.4.1Bound() [1/2]	2176
39.520.4.2Bound() [2/2]	2176
39.520.4.3Center()	2176
39.520.4.4Child() [1/2]	2176
39.520.4.5Child() [2/2]	2177
39.520.4.6ChildPtr()	2177
39.520.4.7Dataset()	2177
39.520.4.8Descendant()	2177
39.520.4.9FurthestDescendantDistance()	2178
39.520.4.10FurthestPointDistance()	2178
39.520.4.11GetFurthestChild() [1/2]	2178
39.520.4.12GetFurthestChild() [2/2]	2178
39.520.4.13GetNearestChild() [1/2]	2179
39.520.4.14GetNearestChild() [2/2]	2179
39.520.4.15Leaf()	2179
39.520.4.16MaxDistance() [1/2]	2179
39.520.4.17MaxDistance() [2/2]	2180
39.520.4.18Metric()	2180
39.520.4.19MinDistance() [1/2]	2180
39.520.4.20MinDistance() [2/2]	2180
39.520.4.21MinimumBoundDistance()	2181
39.520.4.22NumChildren()	2181
39.520.4.23NumDescendants()	2181
39.520.4.24NumPoints()	2181
39.520.4.25Parent() [1/2]	2181
39.520.4.26Parent() [2/2]	2182
39.520.4.27ParentDistance() [1/2]	2182
39.520.4.28ParentDistance() [2/2]	2182

39.520.4.29	Point()	2182
39.520.4.30	RangeDistance() [1/2]	2183
39.520.4.31	RangeDistance() [2/2]	2183
39.520.4.32	Serialize()	2183
39.520.4.33	Stat() [1/2]	2183
39.520.4.34	Stat() [2/2]	2184
39.520	Octree< MetricType, StatisticType, MatType >::DualTreeTraverser< MetricType, StatisticType, MatType > Class Template Reference	2184
39.521.1	Detailed Description	2185
39.521.2	Constructor & Destructor Documentation	2185
39.521.2.1	DualTreeTraverser()	2185
39.521.3	Member Function Documentation	2185
39.521.3.1	NumBaseCases() [1/2]	2185
39.521.3.2	NumBaseCases() [2/2]	2185
39.521.3.3	NumPrunes() [1/2]	2186
39.521.3.4	NumPrunes() [2/2]	2186
39.521.3.5	NumScores() [1/2]	2186
39.521.3.6	NumScores() [2/2]	2186
39.521.3.7	NumVisited()	2186
39.521.3.8	NumVistied()	2187
39.521.3.9	Traverse()	2187
39.522	Octree< MetricType, StatisticType, MatType >::SingleTreeTraverser< RuleType > Class Template Ref- erence	2187
39.522.1	Detailed Description	2188
39.522.2	Constructor & Destructor Documentation	2188
39.522.2.1	SingleTreeTraverser()	2188
39.522.3	Member Function Documentation	2188
39.522.3.1	NumPrunes() [1/2]	2188
39.522.3.2	NumPrunes() [2/2]	2188

39.522.3.3	Traverse()	2188
39.523	Octree< MetricType, StatisticType, MatType >::SplitType::SplitInfo Struct Reference	2189
39.523.1	Detailed Description	2189
39.523.2	Constructor & Destructor Documentation	2189
39.523.2.1	SplitInfo()	2189
39.523.3	Member Data Documentation	2190
39.523.3.1	center	2190
39.523.3.2	id	2190
39.524	ProjVector Class Reference	2190
39.524.1	Detailed Description	2191
39.524.2	Constructor & Destructor Documentation	2191
39.524.2.1	ProjVector() [1/2]	2191
39.524.2.2	ProjVector() [2/2]	2191
39.524.3	Member Function Documentation	2191
39.524.3.1	Project() [1/2]	2191
39.524.3.2	Project() [2/2]	2192
39.524.3.3	Serialize()	2192
39.525	QueueFrame< TreeType, TraversalInfoType > Struct Template Reference	2193
39.525.1	Detailed Description	2193
39.525.2	Member Data Documentation	2193
39.525.2.1	queryDepth	2193
39.525.2.2	queryNode	2193
39.525.2.3	referenceNode	2193
39.525.2.4	score	2194
39.525.2.5	traversalInfo	2194
39.526	RandomDimensionSelect Class Reference	2194
39.526.1	Detailed Description	2194
39.526.2	Constructor & Destructor Documentation	2195

39.526.2.1RandomDimensionSelect()	2195
39.526.3Member Function Documentation	2195
39.526.3.1Begin()	2195
39.526.3.2Dimensions() [1/2]	2195
39.526.3.3Dimensions() [2/2]	2195
39.526.3.4End()	2196
39.526.3.5Next()	2196
39.527RandomForest< FitnessFunction, DimensionSelectionType, NumericSplitType, CategoricalSplitType, ElemType > Class Template Reference	2196
39.527.1Detailed Description	2198
39.527.2Member Typedef Documentation	2198
39.527.2.1DecisionTreeType	2198
39.527.3Constructor & Destructor Documentation	2199
39.527.3.1RandomForest() [1/5]	2199
39.527.3.2RandomForest() [2/5]	2199
39.527.3.3RandomForest() [3/5]	2200
39.527.3.4RandomForest() [4/5]	2200
39.527.3.5RandomForest() [5/5]	2201
39.527.4Member Function Documentation	2201
39.527.4.1Classify() [1/4]	2202
39.527.4.2Classify() [2/4]	2202
39.527.4.3Classify() [3/4]	2202
39.527.4.4Classify() [4/4]	2203
39.527.4.5NumTrees()	2203
39.527.4.6Serialize()	2203
39.527.4.7Train() [1/4]	2204
39.527.4.8Train() [2/4]	2204
39.527.4.9Train() [3/4]	2205

39.527.4.1	Train() [4 / 4]	2206
39.527.4.1	Tree() [1 / 2]	2206
39.527.4.1	Tree() [2 / 2]	2207
39.528	RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType > Class Template Reference	2207
39.528.1	Detailed Description	2211
39.528.2	Member Typedef Documentation	2212
39.528.2.1	AuxiliaryInformation	2212
39.528.2.2	ElemType	2212
39.528.2.3	Mat	2212
39.528.3	Constructor & Destructor Documentation	2213
39.528.3.1	RectangleTree() [1 / 7]	2213
39.528.3.2	RectangleTree() [2 / 7]	2213
39.528.3.3	RectangleTree() [3 / 7]	2214
39.528.3.4	RectangleTree() [4 / 7]	2214
39.528.3.5	RectangleTree() [5 / 7]	2215
39.528.3.6	RectangleTree() [6 / 7]	2215
39.528.3.7	~RectangleTree()	2215
39.528.3.8	RectangleTree() [7 / 7]	2215
39.528.4	Member Function Documentation	2216
39.528.4.1	AuxiliaryInfo() [1 / 2]	2216
39.528.4.2	AuxiliaryInfo() [2 / 2]	2216
39.528.4.3	Begin() [1 / 2]	2216
39.528.4.4	Begin() [2 / 2]	2216
39.528.4.5	Bound() [1 / 2]	2217
39.528.4.6	Bound() [2 / 2]	2217
39.528.4.7	Center()	2217
39.528.4.8	Child() [1 / 2]	2217

39.528.4.9Child() [2/2]	2218
39.528.4.10CondenseTree()	2218
39.528.4.11Count() [1/2]	2218
39.528.4.12Count() [2/2]	2219
39.528.4.13Dataset() [1/2]	2219
39.528.4.14Dataset() [2/2]	2219
39.528.4.15DeletePoint() [1/2]	2219
39.528.4.16DeletePoint() [2/2]	2220
39.528.4.17Descendant()	2220
39.528.4.18ExactClone()	2220
39.528.4.19FindByBeginCount() [1/2]	2220
39.528.4.20FindByBeginCount() [2/2]	2221
39.528.4.21FurthestDescendantDistance()	2221
39.528.4.22FurthestPointDistance()	2222
39.528.4.23GetFurthestChild() [1/2]	2222
39.528.4.24GetFurthestChild() [2/2]	2222
39.528.4.25GetNearestChild() [1/2]	2222
39.528.4.26GetNearestChild() [2/2]	2223
39.528.4.27InsertNode()	2223
39.528.4.28InsertPoint() [1/2]	2223
39.528.4.29InsertPoint() [2/2]	2224
39.528.4.30Leaf()	2224
39.528.4.31MaxDistance() [1/2]	2224
39.528.4.32MaxDistance() [2/2]	2224
39.528.4.33MaxLeafSize() [1/2]	2225
39.528.4.34MaxLeafSize() [2/2]	2225
39.528.4.35MaxNumChildren() [1/2]	2225
39.528.4.36MaxNumChildren() [2/2]	2225

39.528.4.37	Metric()	2225
39.528.4.38	MinDistance() [1/2]	2226
39.528.4.39	MinDistance() [2/2]	2226
39.528.4.40	MinimumBoundDistance()	2226
39.528.4.41	MinLeafSize() [1/2]	2226
39.528.4.42	MinLeafSize() [2/2]	2227
39.528.4.43	MinNumChildren() [1/2]	2227
39.528.4.44	MinNumChildren() [2/2]	2227
39.528.4.45	MullifyData()	2227
39.528.4.46	NumChildren() [1/2]	2227
39.528.4.47	NumChildren() [2/2]	2228
39.528.4.48	NumDescendants()	2228
39.528.4.49	NumPoints()	2228
39.528.4.50	Parent() [1/2]	2228
39.528.4.51	Parent() [2/2]	2229
39.528.4.52	ParentDistance() [1/2]	2229
39.528.4.53	ParentDistance() [2/2]	2229
39.528.4.54	Point() [1/2]	2229
39.528.4.55	Point() [2/2]	2230
39.528.4.56	RangeDistance() [1/2]	2230
39.528.4.57	RangeDistance() [2/2]	2230
39.528.4.58	RemoveNode()	2231
39.528.4.59	Serialize()	2231
39.528.4.60	ShrinkBoundForBound()	2231
39.528.4.61	ShrinkBoundForPoint()	2231
39.528.4.62	SoftDelete()	2232
39.528.4.63	Stat() [1/2]	2232
39.528.4.64	Stat() [2/2]	2232

39.528.4.5FreeDepth()	2232
39.528.4.6FreeSize()	2233
39.528.5Member Data Documentation	2233
39.528.5.1AuxiliaryInformation	2233
39.528.5.2DescentType	2233
39.528.5.3SplitType	2233
39.529RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::DualTreeTraverser< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType > Class Template Reference	2234
39.529.1Detailed Description	2234
39.529.2Constructor & Destructor Documentation	2235
39.529.2.1DualTreeTraverser()	2235
39.529.3Member Function Documentation	2235
39.529.3.1NumBaseCases() [1/2]	2235
39.529.3.2NumBaseCases() [2/2]	2235
39.529.3.3NumPrunes() [1/2]	2235
39.529.3.4NumPrunes() [2/2]	2236
39.529.3.5NumScores() [1/2]	2236
39.529.3.6NumScores() [2/2]	2236
39.529.3.7NumVisited() [1/2]	2236
39.529.3.8NumVisited() [2/2]	2236
39.529.3.9Traverse()	2236
39.530RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::SingleTreeTraverser< RuleType > Class Template Reference	2237
39.530.1Detailed Description	2237
39.530.2Constructor & Destructor Documentation	2238
39.530.2.1SingleTreeTraverser()	2238
39.530.3Member Function Documentation	2238
39.530.3.1NumPrunes() [1/2]	2238

39.530.3.2NumPrunes() [2/2]	2238
39.530.3.3Traverse()	2238
39.531RPlusPlusTreeAuxiliaryInformation< TreeType > Class Template Reference	2239
39.531.1Detailed Description	2240
39.531.2Member Typedef Documentation	2240
39.531.2.1BoundType	2240
39.531.2.2ElemType	2240
39.531.3Constructor & Destructor Documentation	2240
39.531.3.1RPlusPlusTreeAuxiliaryInformation() [1/4]	2241
39.531.3.2RPlusPlusTreeAuxiliaryInformation() [2/4]	2241
39.531.3.3RPlusPlusTreeAuxiliaryInformation() [3/4]	2242
39.531.3.4RPlusPlusTreeAuxiliaryInformation() [4/4]	2242
39.531.4Member Function Documentation	2242
39.531.4.1HandleNodeInsertion()	2242
39.531.4.2HandleNodeRemoval()	2243
39.531.4.3HandlePointDeletion()	2243
39.531.4.4HandlePointInsertion()	2244
39.531.4.5NullifyData()	2244
39.531.4.6OuterBound() [1/2]	2244
39.531.4.7OuterBound() [2/2]	2245
39.531.4.8Serialize()	2245
39.531.4.9SplitAuxiliaryInfo()	2245
39.531.4.10UpdateAuxiliaryInfo()	2246
39.532RPlusPlusTreeDescentHeuristic Class Reference	2246
39.532.1Detailed Description	2246
39.532.2Member Function Documentation	2246
39.532.2.1ChooseDescentNode() [1/2]	2246
39.532.2.2ChooseDescentNode() [2/2]	2247

39.533	PlusPlusTreeSplitPolicy Class Reference	2247
39.533.1	Detailed Description	2248
39.533.2	Member Function Documentation	2248
39.533.2.1	Bound()	2248
39.533.2.2	GetSplitPolicy()	2248
39.533.3	Member Data Documentation	2249
39.533.3.1	AssignToFirstTree	2249
39.533.3.2	AssignToSecondTree	2249
39.533.3.3	SplitRequired	2250
39.534	PlusTreeDescentHeuristic Class Reference	2250
39.534.1	Detailed Description	2250
39.534.2	Member Function Documentation	2250
39.534.2.1	ChooseDescentNode() [1/2]	2250
39.534.2.2	ChooseDescentNode() [2/2]	2251
39.535	PlusTreeSplit< SplitPolicyType, SweepType > Class Template Reference	2251
39.535.1	Detailed Description	2252
39.535.2	Member Typedef Documentation	2252
39.535.2.1	SplitPolicy	2252
39.535.3	Member Function Documentation	2252
39.535.3.1	SplitLeafNode()	2252
39.535.3.2	SplitNonLeafNode()	2253
39.536	PlusTreeSplitPolicy Class Reference	2253
39.536.1	Detailed Description	2254
39.536.2	Member Function Documentation	2254
39.536.2.1	Bound()	2254
39.536.2.2	GetSplitPolicy()	2254
39.536.3	Member Data Documentation	2255
39.536.3.1	AssignToFirstTree	2255

39.536.3.2	AssignToSecondTree	2255
39.536.3.3	SplitRequired	2255
39.537	RPTreeMaxSplit< BoundType, MatType > Class Template Reference	2255
39.537.1	Detailed Description	2256
39.537.2	Member Typedef Documentation	2256
39.537.2.1	ElemType	2257
39.537.3	Member Function Documentation	2257
39.537.3.1	AssignToLeftNode()	2257
39.537.3.2	PerformSplit() [1/2]	2257
39.537.3.3	PerformSplit() [2/2]	2258
39.537.3.4	SplitNode()	2258
39.538	RPTreeMaxSplit< BoundType, MatType >::SplitInfo Struct Reference	2260
39.538.1	Detailed Description	2260
39.538.2	Member Data Documentation	2260
39.538.2.1	direction	2260
39.538.2.2	splitVal	2261
39.539	RPTreeMeanSplit< BoundType, MatType > Class Template Reference	2261
39.539.1	Detailed Description	2262
39.539.2	Member Typedef Documentation	2262
39.539.2.1	ElemType	2262
39.539.3	Member Function Documentation	2262
39.539.3.1	AssignToLeftNode()	2262
39.539.3.2	PerformSplit() [1/2]	2263
39.539.3.3	PerformSplit() [2/2]	2263
39.539.3.4	SplitNode()	2264
39.540	RPTreeMeanSplit< BoundType, MatType >::SplitInfo Struct Reference	2264
39.540.1	Detailed Description	2265
39.540.2	Member Data Documentation	2265

39.540.2.1direction	2265
39.540.2.2mean	2265
39.540.2.3meanSplit	2266
39.540.2.4splitVal	2266
39.541RStarTreeDescentHeuristic Class Reference	2266
39.541.1Detailed Description	2266
39.541.2Member Function Documentation	2267
39.541.2.1ChooseDescentNode() [1/2]	2267
39.541.2.2ChooseDescentNode() [2/2]	2267
39.542RStarTreeSplit Class Reference	2267
39.542.1Detailed Description	2268
39.542.2Member Function Documentation	2268
39.542.2.1PickLeafSplit()	2268
39.542.2.2ReinsertPoints()	2268
39.542.2.3SplitLeafNode()	2269
39.542.2.4SplitNonLeafNode()	2269
39.543RTreeDescentHeuristic Class Reference	2269
39.543.1Detailed Description	2269
39.543.2Member Function Documentation	2270
39.543.2.1ChooseDescentNode() [1/2]	2270
39.543.2.2ChooseDescentNode() [2/2]	2270
39.544RTreeSplit Class Reference	2271
39.544.1Detailed Description	2271
39.544.2Member Function Documentation	2271
39.544.2.1SplitLeafNode()	2271
39.544.2.2SplitNonLeafNode()	2272
39.545SpaceSplit< MetricType, MatType > Class Template Reference	2272
39.545.1Detailed Description	2272

39.545.2	Member Function Documentation	2272
39.545.2.1	GetProjVector() [1/2]	2272
39.545.2.2	GetProjVector() [2/2]	2273
39.546	SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > Class Template Reference	2274
39.546.1	Detailed Description	2277
39.546.2	Member Typedef Documentation	2278
39.546.2.1	BoundType	2278
39.546.2.2	DefeatistDualTreeTraverser	2278
39.546.2.3	DefeatistSingleTreeTraverser	2278
39.546.2.4	DualTreeTraverser	2279
39.546.2.5	ElemType	2279
39.546.2.6	Mat	2279
39.546.2.7	SingleTreeTraverser	2279
39.546.3	Constructor & Destructor Documentation	2279
39.546.3.1	SpillTree() [1/7]	2279
39.546.3.2	SpillTree() [2/7]	2280
39.546.3.3	SpillTree() [3/7]	2280
39.546.3.4	SpillTree() [4/7]	2281
39.546.3.5	SpillTree() [5/7]	2281
39.546.3.6	SpillTree() [6/7]	2281
39.546.3.7	~SpillTree()	2283
39.546.3.8	SpillTree() [7/7]	2283
39.546.4	Member Function Documentation	2283
39.546.4.1	Bound() [1/2]	2283
39.546.4.2	Bound() [2/2]	2284
39.546.4.3	Center()	2284
39.546.4.4	Child()	2284
39.546.4.5	ChildPtr()	2284

39.546.4.6Dataset()	2285
39.546.4.7Descendant()	2285
39.546.4.8FurthestDescendantDistance()	2285
39.546.4.9FurthestPointDistance()	2286
39.546.4.10GetFurthestChild() [1/2]	2286
39.546.4.10GetFurthestChild() [2/2]	2286
39.546.4.10GetNearestChild() [1/2]	2286
39.546.4.10GetNearestChild() [2/2]	2287
39.546.4.11HasSelfChildren()	2287
39.546.4.11Hyperplane()	2287
39.546.4.11Leaf()	2287
39.546.4.11Left() [1/2]	2287
39.546.4.11Left() [2/2]	2288
39.546.4.11MaxDistance() [1/2]	2288
39.546.4.11MaxDistance() [2/2]	2288
39.546.4.11Metric()	2288
39.546.4.11MinDistance() [1/2]	2289
39.546.4.11MinDistance() [2/2]	2289
39.546.4.11MinimumBoundDistance()	2289
39.546.4.11SumChildren()	2289
39.546.4.11SumDescendants()	2290
39.546.4.11SumPoints()	2290
39.546.4.11Overlap()	2290
39.546.4.11Parent() [1/2]	2290
39.546.4.11Parent() [2/2]	2291
39.546.4.11ParentDistance() [1/2]	2291
39.546.4.11ParentDistance() [2/2]	2291
39.546.4.11Point()	2291

39.546.4.36	RangeDistance() [1/2]	2292
39.546.4.36	RangeDistance() [2/2]	2292
39.546.4.36	Right() [1/2]	2292
39.546.4.36	Right() [2/2]	2292
39.546.4.36	Serialize()	2293
39.546.4.36	Stat() [1/2]	2293
39.546.4.36	Stat() [2/2]	2293
39.547	SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::SpillDualTreeTraverser< MetricType, StatisticType, MatType, HyperplaneType, SplitType > Class Template Reference	2293
39.547.1	Detailed Description	2294
39.547.2	Constructor & Destructor Documentation	2294
39.547.2.1	SpillDualTreeTraverser()	2295
39.547.3	Member Function Documentation	2295
39.547.3.1	NumBaseCases() [1/2]	2295
39.547.3.2	NumBaseCases() [2/2]	2295
39.547.3.3	NumPrunes() [1/2]	2295
39.547.3.4	NumPrunes() [2/2]	2296
39.547.3.5	NumScores() [1/2]	2296
39.547.3.6	NumScores() [2/2]	2296
39.547.3.7	NumVisited() [1/2]	2296
39.547.3.8	NumVisited() [2/2]	2296
39.547.3.9	Traverse()	2296
39.548	SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::SpillSingleTreeTraverser< MetricType, StatisticType, MatType, HyperplaneType, SplitType > Class Template Reference	2297
39.548.1	Detailed Description	2297
39.548.2	Constructor & Destructor Documentation	2298
39.548.2.1	SpillSingleTreeTraverser()	2298
39.548.3	Member Function Documentation	2298
39.548.3.1	NumPrunes() [1/2]	2298

39.548.3.2	NumPrunes() [2/2]	2298
39.548.3.3	Traverse()	2298
39.549	TraversallInfo< TreeType > Class Template Reference	2299
39.549.1	Detailed Description	2300
39.549.2	Constructor & Destructor Documentation	2300
39.549.2.1	TraversallInfo()	2300
39.549.3	Member Function Documentation	2300
39.549.3.1	LastBaseCase() [1/2]	2301
39.549.3.2	LastBaseCase() [2/2]	2301
39.549.3.3	LastQueryNode() [1/2]	2301
39.549.3.4	LastQueryNode() [2/2]	2301
39.549.3.5	LastReferenceNode() [1/2]	2301
39.549.3.6	LastReferenceNode() [2/2]	2302
39.549.3.7	LastScore() [1/2]	2302
39.549.3.8	LastScore() [2/2]	2302
39.550	TreeTraits< TreeType > Class Template Reference	2302
39.550.1	Detailed Description	2303
39.550.2	Member Data Documentation	2303
39.550.2.1	BinaryTree	2304
39.550.2.2	FirstPointIsCentroid	2304
39.550.2.3	HasDuplicatedPoints	2304
39.550.2.4	HasOverlappingChildren	2304
39.550.2.5	HasSelfChildren	2304
39.550.2.6	RearrangesDataset	2305
39.550.2.7	UniqueNumDescendants	2305
39.551	TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::BallBound, SplitType > > Class Template Reference	2305
39.551.1	Detailed Description	2305

39.551.1	Member Data Documentation	2306
39.551.2.1	BinaryTree	2306
39.551.2.2	FirstPointIsCentroid	2306
39.551.2.3	HasDuplicatedPoints	2306
39.551.2.4	HasOverlappingChildren	2306
39.551.2.5	HasSelfChildren	2306
39.551.2.6	RearrangesDataset	2307
39.551.2.7	UniqueNumDescendants	2307
39.552	TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::CellBound, SplitType > > Class Template Reference	2307
39.552.1	Detailed Description	2307
39.552.2	Member Data Documentation	2308
39.552.2.1	BinaryTree	2308
39.552.2.2	FirstPointIsCentroid	2308
39.552.2.3	HasDuplicatedPoints	2308
39.552.2.4	HasOverlappingChildren	2308
39.552.2.5	HasSelfChildren	2308
39.552.2.6	RearrangesDataset	2309
39.552.2.7	UniqueNumDescendants	2309
39.553	TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::HollowBallBound, SplitType > > Class Template Reference	2309
39.553.1	Detailed Description	2309
39.553.2	Member Data Documentation	2310
39.553.2.1	BinaryTree	2310
39.553.2.2	FirstPointIsCentroid	2310
39.553.2.3	HasDuplicatedPoints	2310
39.553.2.4	HasOverlappingChildren	2310
39.553.2.5	HasSelfChildren	2310
39.553.2.6	RearrangesDataset	2311

39.553.2.7UniqueNumDescendants	2311
39.554TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, RPTreeMaxSplit > > Class Template Reference	2311
39.554.1Detailed Description	2312
39.554.2Member Data Documentation	2312
39.554.2.1BinaryTree	2312
39.554.2.2FirstPointIsCentroid	2312
39.554.2.3HasDuplicatedPoints	2312
39.554.2.4HasOverlappingChildren	2313
39.554.2.5HasSelfChildren	2313
39.554.2.6RearrangesDataset	2313
39.554.2.7UniqueNumDescendants	2313
39.555TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, RPTreeMeanSplit > > Class Template Reference	2313
39.555.1Detailed Description	2314
39.555.2Member Data Documentation	2314
39.555.2.1BinaryTree	2314
39.555.2.2FirstPointIsCentroid	2315
39.555.2.3HasDuplicatedPoints	2315
39.555.2.4HasOverlappingChildren	2315
39.555.2.5HasSelfChildren	2315
39.555.2.6RearrangesDataset	2315
39.555.2.7UniqueNumDescendants	2316
39.556TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > > Class Template Reference	2316
39.556.1Detailed Description	2316
39.556.2Member Data Documentation	2317
39.556.2.1BinaryTree	2317
39.556.2.2FirstPointIsCentroid	2317

39.556.2.3	HasDuplicatedPoints	2317
39.556.2.4	HasOverlappingChildren	2317
39.556.2.5	HasSelfChildren	2318
39.556.2.6	RearrangesDataset	2318
39.556.2.7	UniqueNumDescendants	2318
39.557	TreeTraits< CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > > Class Template Reference	2318
39.557.1	Detailed Description	2319
39.557.2	Member Data Documentation	2319
39.557.2.1	BinaryTree	2319
39.557.2.2	FirstPointIsCentroid	2320
39.557.2.3	HasDuplicatedPoints	2320
39.557.2.4	HasOverlappingChildren	2320
39.557.2.5	HasSelfChildren	2320
39.557.2.6	RearrangesDataset	2320
39.557.2.7	UniqueNumDescendants	2321
39.558	TreeTraits< Octree< MetricType, StatisticType, MatType > > Class Template Reference	2321
39.558.1	Detailed Description	2321
39.558.2	Member Data Documentation	2322
39.558.2.1	BinaryTree	2322
39.558.2.2	FirstPointIsCentroid	2322
39.558.2.3	HasDuplicatedPoints	2322
39.558.2.4	HasOverlappingChildren	2322
39.558.2.5	HasSelfChildren	2323
39.558.2.6	RearrangesDataset	2323
39.558.2.7	UniqueNumDescendants	2323
39.559	TreeTraits< RectangleTree< MetricType, StatisticType, MatType, RPlusTreeSplit< SplitPolicyType, SweepType >, DescentType, AuxiliaryInformationType > > Class Template Reference	2323
39.559.1	Detailed Description	2324

39.559.1	Member Data Documentation	2324
39.559.2.1	BinaryTree	2324
39.559.2.2	FirstPointIsCentroid	2325
39.559.2.3	HasDuplicatedPoints	2325
39.559.2.4	HasOverlappingChildren	2325
39.559.2.5	HasSelfChildren	2325
39.559.2.6	RearrangesDataset	2325
39.559.2.7	UniqueNumDescendants	2326
39.560	TreeTraits< RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, Auxiliary← InformationType > > Class Template Reference	2326
39.560.1	Detailed Description	2326
39.560.2	Member Data Documentation	2327
39.560.2.1	BinaryTree	2327
39.560.2.2	FirstPointIsCentroid	2327
39.560.2.3	HasDuplicatedPoints	2327
39.560.2.4	HasOverlappingChildren	2327
39.560.2.5	HasSelfChildren	2328
39.560.2.6	RearrangesDataset	2328
39.560.2.7	UniqueNumDescendants	2328
39.561	TreeTraits< SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > > Class Tem- plate Reference	2328
39.561.1	Detailed Description	2329
39.561.2	Member Data Documentation	2329
39.561.2.1	BinaryTree	2329
39.561.2.2	FirstPointIsCentroid	2330
39.561.2.3	HasOverlappingChildren	2330
39.561.2.4	HasSelfChildren	2330
39.561.2.5	RearrangesDataset	2330
39.561.2.6	UniqueNumDescendants	2331

39.562	B TreeSplit< BoundType, MatType > Class Template Reference	2331
39.562.1	Detailed Description	2331
39.563	V antagePointSplit< BoundType, MatType, MaxNumSamples > Class Template Reference	2331
39.563.1	Detailed Description	2332
39.563.2	Member Typedef Documentation	2332
39.563.2.1	ElemType	2332
39.563.2.2	MetricType	2333
39.563.3	Member Function Documentation	2333
39.563.3.1	AssignToLeftNode()	2333
39.563.3.2	PerformSplit() [1/2]	2333
39.563.3.3	PerformSplit() [2/2]	2334
39.563.3.4	SplitNode()	2334
39.564	V antagePointSplit< BoundType, MatType, MaxNumSamples >::SplitInfo Struct Reference	2336
39.564.1	Detailed Description	2336
39.564.2	Constructor & Destructor Documentation	2337
39.564.2.1	SplitInfo() [1/2]	2337
39.564.2.2	SplitInfo() [2/2]	2337
39.564.3	Member Data Documentation	2337
39.564.3.1	metric	2337
39.564.3.2	mu	2337
39.564.3.3	avantagePoint	2338
39.565	X TreeAuxiliaryInformation< TreeType > Class Template Reference	2338
39.565.1	Detailed Description	2339
39.565.2	Member Typedef Documentation	2339
39.565.2.1	SplitHistoryStruct	2339
39.565.3	Constructor & Destructor Documentation	2340
39.565.3.1	XTreeAuxiliaryInformation() [1/4]	2340
39.565.3.2	XTreeAuxiliaryInformation() [2/4]	2340

39.565.3.3XTreeAuxiliaryInformation() [3/4]	2340
39.565.3.4XTreeAuxiliaryInformation() [4/4]	2341
39.565.4Member Function Documentation	2341
39.565.4.1HandleNodeInsertion()	2341
39.565.4.2HandleNodeRemoval()	2342
39.565.4.3HandlePointDeletion()	2342
39.565.4.4HandlePointInsertion()	2342
39.565.4.5NormalNodeMaxNumChildren() [1/2]	2343
39.565.4.6NormalNodeMaxNumChildren() [2/2]	2343
39.565.4.7NullifyData()	2343
39.565.4.8operator=()	2343
39.565.4.9Serialize()	2344
39.565.4.10SplitHistory() [1/2]	2344
39.565.4.11SplitHistory() [2/2]	2344
39.565.4.12UpdateAuxiliaryInfo()	2344
39.566XTreeAuxiliaryInformation< TreeType >::SplitHistoryStruct Struct Reference	2345
39.566.1Detailed Description	2345
39.566.2Constructor & Destructor Documentation	2345
39.566.2.1SplitHistoryStruct() [1/3]	2346
39.566.2.2SplitHistoryStruct() [2/3]	2346
39.566.2.3SplitHistoryStruct() [3/3]	2346
39.566.3Member Function Documentation	2346
39.566.3.1operator=()	2346
39.566.3.2Serialize()	2347
39.566.4Member Data Documentation	2347
39.566.4.1history	2347
39.566.4.2lastDimension	2347
39.567XTreeSplit Class Reference	2347

39.567.1	Detailed Description	2348
39.567.2	Member Function Documentation	2348
39.567.2.1	SplitLeafNode()	2348
39.567.2.2	SplitNonLeafNode()	2348
39.568	StdVector< T > Struct Template Reference	2349
39.568.1	Detailed Description	2349
39.568.2	Member Data Documentation	2349
39.568.2.1	value	2349
39.569	StdVector< std::vector< T, A > > Struct Template Reference	2349
39.569.1	Detailed Description	2350
39.569.2	Member Data Documentation	2350
39.569.2.1	value	2350
39.570	NullOutStream Class Reference	2350
39.570.1	Detailed Description	2351
39.570.2	Constructor & Destructor Documentation	2351
39.570.2.1	NullOutStream() [1/2]	2351
39.570.2.2	NullOutStream() [2/2]	2352
39.570.3	Member Function Documentation	2352
39.570.3.1	operator<<() [1/18]	2352
39.570.3.2	operator<<() [2/18]	2352
39.570.3.3	operator<<() [3/18]	2352
39.570.3.4	operator<<() [4/18]	2353
39.570.3.5	operator<<() [5/18]	2353
39.570.3.6	operator<<() [6/18]	2353
39.570.3.7	operator<<() [7/18]	2353
39.570.3.8	operator<<() [8/18]	2354
39.570.3.9	operator<<() [9/18]	2354
39.570.3.10	operator<<() [10/18]	2354

39.570.3.10operator<<() [11/18]	2354
39.570.3.10operator<<() [12/18]	2355
39.570.3.10operator<<() [13/18]	2355
39.570.3.10operator<<() [14/18]	2355
39.570.3.10operator<<() [15/18]	2355
39.570.3.10operator<<() [16/18]	2356
39.570.3.10operator<<() [17/18]	2356
39.570.3.10operator<<() [18/18]	2356
39.57ParamData Struct Reference	2356
39.571.Detailed Description	2357
39.571.1Member Data Documentation	2357
39.571.2.1alias	2358
39.571.2.2appType	2358
39.571.2.3desc	2358
39.571.2.4input	2358
39.571.2.5loaded	2359
39.571.2.6name	2359
39.571.2.7noTranspose	2359
39.571.2.8persistent	2360
39.571.2.9required	2360
39.571.2.10name	2360
39.571.2.11value	2361
39.571.2.12asPassed	2361
39.572PrefixedOutStream Class Reference	2361
39.572.1Detailed Description	2363
39.572.2Constructor & Destructor Documentation	2363
39.572.2.1PrefixedOutStream()	2363
39.572.3Member Function Documentation	2364

39.572.3.1operator<<() [1/18]	2364
39.572.3.2operator<<() [2/18]	2364
39.572.3.3operator<<() [3/18]	2364
39.572.3.4operator<<() [4/18]	2365
39.572.3.5operator<<() [5/18]	2365
39.572.3.6operator<<() [6/18]	2365
39.572.3.7operator<<() [7/18]	2365
39.572.3.8operator<<() [8/18]	2365
39.572.3.9operator<<() [9/18]	2366
39.572.3.10operator<<() [10/18]	2366
39.572.3.11operator<<() [11/18]	2366
39.572.3.12operator<<() [12/18]	2366
39.572.3.13operator<<() [13/18]	2366
39.572.3.14operator<<() [14/18]	2367
39.572.3.15operator<<() [15/18]	2367
39.572.3.16operator<<() [16/18]	2367
39.572.3.17operator<<() [17/18]	2367
39.572.3.18operator<<() [18/18]	2367
39.572.4Member Data Documentation	2367
39.572.4.1backtrace	2368
39.572.4.2destination	2368
39.572.4.3ignoreInput	2368
39.573ProgramDoc Class Reference	2368
39.573.1Detailed Description	2369
39.573.2Constructor & Destructor Documentation	2369
39.573.2.1ProgramDoc() [1/2]	2369
39.573.2.2ProgramDoc() [2/2]	2370
39.573.3Member Data Documentation	2370
39.573.3.1documentation	2370
39.573.3.2programName	2370
39.573.3.3seeAlso	2370
39.573.3.4shortDocumentation	2371
39.574TrainHMMModel Struct Reference	2371
39.574.1Detailed Description	2371
39.574.2Member Function Documentation	2371
39.574.2.1Apply()	2371

40 File Documentation	2373
40.1 /home/barak/src/git/debian-src/mlpack/doc/guide/bindings.hpp File Reference	2373
40.2 /home/barak/src/git/debian-src/mlpack/doc/guide/build.hpp File Reference	2373
40.3 /home/barak/src/git/debian-src/mlpack/doc/guide/build_windows.hpp File Reference	2373
40.3.1 Detailed Description	2373
40.4 /home/barak/src/git/debian-src/mlpack/doc/guide/cli_quickstart.hpp File Reference	2373
40.4.1 Detailed Description	2373
40.5 /home/barak/src/git/debian-src/mlpack/doc/guide/cv.hpp File Reference	2373
40.6 /home/barak/src/git/debian-src/mlpack/doc/guide/formats.hpp File Reference	2374
40.6.1 Detailed Description	2374
40.7 /home/barak/src/git/debian-src/mlpack/doc/guide/iodoc.hpp File Reference	2374
40.8 /home/barak/src/git/debian-src/mlpack/doc/guide/matrices.hpp File Reference	2374
40.9 /home/barak/src/git/debian-src/mlpack/doc/guide/python_quickstart.hpp File Reference	2374
40.9.1 Detailed Description	2374
40.10/home/barak/src/git/debian-src/mlpack/doc/guide/sample.hpp File Reference	2374
40.11/home/barak/src/git/debian-src/mlpack/doc/guide/sample_ml_app.hpp File Reference	2374
40.11.1 Detailed Description	2374
40.12/home/barak/src/git/debian-src/mlpack/doc/guide/timer.hpp File Reference	2375
40.13/home/barak/src/git/debian-src/mlpack/doc/policies/elemtype.hpp File Reference	2375
40.14/home/barak/src/git/debian-src/mlpack/doc/policies/functiontype.hpp File Reference	2375
40.15/home/barak/src/git/debian-src/mlpack/doc/policies/kernels.hpp File Reference	2375
40.16/home/barak/src/git/debian-src/mlpack/doc/policies/metrics.hpp File Reference	2375
40.17/home/barak/src/git/debian-src/mlpack/doc/policies/trees.hpp File Reference	2375
40.18/home/barak/src/git/debian-src/mlpack/doc/tutorials/amf/amf.txt File Reference	2375
40.18.1 Detailed Description	2375
40.19/home/barak/src/git/debian-src/mlpack/doc/tutorials/ann/ann.txt File Reference	2375
40.19.1 Detailed Description	2375
40.20/home/barak/src/git/debian-src/mlpack/doc/tutorials/approx_kfn/approx_kfn.txt File Reference	2376

40.20.1 Detailed Description	2376
40.21/home/barak/src/git/debian-src/mlpack/doc/tutorials/cf/cf.txt File Reference	2376
40.21.1 Detailed Description	2376
40.22/home/barak/src/git/debian-src/mlpack/doc/tutorials/det/det.txt File Reference	2376
40.22.1 Detailed Description	2377
40.22.2 Function Documentation	2377
40.22.2.1 \$V()	2377
40.22.2.2 alpha()	2377
40.22.3 Variable Documentation	2377
40.22.3.1 estimation	2377
40.22.3.2 now	2378
40.22.3.3 regularization	2378
40.22.3.4 Thus	2378
40.23/home/barak/src/git/debian-src/mlpack/doc/tutorials/emst/emst.txt File Reference	2378
40.23.1 Detailed Description	2378
40.24/home/barak/src/git/debian-src/mlpack/doc/tutorials/fastmks/fastmks.txt File Reference	2378
40.24.1 Detailed Description	2379
40.25/home/barak/src/git/debian-src/mlpack/doc/tutorials/kmeans/kmeans.txt File Reference	2379
40.25.1 Detailed Description	2379
40.26/home/barak/src/git/debian-src/mlpack/doc/tutorials/linear_regression/linear_regression.txt File Reference	2379
40.26.1 Detailed Description	2379
40.27/home/barak/src/git/debian-src/mlpack/doc/tutorials/neighbor_search/neighbor_search.txt File Reference	2379
40.27.1 Detailed Description	2380
40.28/home/barak/src/git/debian-src/mlpack/doc/tutorials/range_search/range_search.txt File Reference . .	2380
40.28.1 Detailed Description	2380
40.29/home/barak/src/git/debian-src/mlpack/doc/tutorials/tutorials.txt File Reference	2380
40.29.1 Detailed Description	2380
40.30/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/add_to_po.hpp File Reference	2381

40.30.1 Detailed Description	2382
40.31/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/cli_option.hpp File Reference	2382
40.31.1 Macro Definition Documentation	2383
40.31.1.1 BASH_CLEAR	2383
40.31.1.2 BASH_RED	2383
40.32/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/CMakeLists.txt File Reference	2383
40.32.1 Function Documentation	2384
40.32.1.1 set() [1/2]	2384
40.32.1.2 set() [2/2]	2384
40.33/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/CMakeLists.txt File Reference	2384
40.33.1 Function Documentation	2384
40.33.1.1 add_subdirectory()	2385
40.33.1.2 set() [1/2]	2385
40.33.1.3 set() [2/2]	2385
40.34/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/CMakeLists.txt File Reference	2385
40.34.1 Function Documentation	2385
40.34.1.1 macro() [1/2]	2386
40.34.1.2 macro() [2/2]	2386
40.34.1.3 set()	2386
40.35/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/CMakeLists.txt File Reference	2386
40.35.1 Function Documentation	2387
40.35.1.1 else()	2387
40.35.1.2 endif()	2387
40.35.1.3 find_python_module()	2387
40.35.1.4 macro() [1/2]	2387
40.35.1.5 macro() [2/2]	2387
40.35.1.6 set()	2388
40.36/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/tests/CMakeLists.txt File Reference	2388

40.37/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/CMakeLists.txt File Reference	2388
40.37.1 Function Documentation	2388
40.37.1.1 set() [1/2]	2388
40.37.1.2 set() [2/2]	2388
40.38/home/barak/src/git/debian-src/mlpack/src/mlpack/CMakeLists.txt File Reference	2389
40.38.1 Function Documentation	2389
40.38.1.1 include_directories()	2389
40.39/home/barak/src/git/debian-src/mlpack/src/mlpack/core/CMakeLists.txt File Reference	2389
40.39.1 Function Documentation	2389
40.39.1.1 add_subdirectory()	2389
40.39.1.2 set()	2390
40.40/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/CMakeLists.txt File Reference	2390
40.40.1 Function Documentation	2390
40.40.1.1 add_subdirectory()	2390
40.40.1.2 set()	2390
40.41/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/CMakeLists.txt File Reference	2390
40.41.1 Function Documentation	2391
40.41.1.1 set() [1/2]	2391
40.41.1.2 set() [2/2]	2391
40.42/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/CMakeLists.txt File Reference	2391
40.42.1 Function Documentation	2391
40.42.1.1 set() [1/2]	2392
40.42.1.2 set() [2/2]	2392
40.43/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/CMakeLists.txt File Reference	2392
40.43.1 Function Documentation	2392
40.43.1.1 set() [1/2]	2392
40.43.1.2 set() [2/2]	2393

40.44/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/map_policies/CMakeLists.txt File Reference	2393
40.44.1 Function Documentation	2393
40.44.1.1 set() [1/2]	2393
40.44.1.2 set() [2/2]	2393
40.45/home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/CMakeLists.txt File Reference	2393
40.45.1 Function Documentation	2394
40.45.1.1 set() [1/2]	2394
40.45.1.2 set() [2/2]	2394
40.46/home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/CMakeLists.txt File Reference	2394
40.46.1 Function Documentation	2394
40.46.1.1 set() [1/2]	2394
40.46.1.2 set() [2/2]	2395
40.47/home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/CMakeLists.txt File Reference	2395
40.47.1 Function Documentation	2395
40.47.1.1 set() [1/2]	2395
40.47.1.2 set() [2/2]	2395
40.48/home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/CMakeLists.txt File Reference	2396
40.48.1 Function Documentation	2396
40.48.1.1 set() [1/2]	2396
40.48.1.2 set() [2/2]	2396
40.49/home/barak/src/git/debian-src/mlpack/src/mlpack/core/metrics/CMakeLists.txt File Reference	2396
40.49.1 Function Documentation	2397
40.49.1.1 set() [1/2]	2397
40.49.1.2 set() [2/2]	2397
40.50/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/CMakeLists.txt File Reference	2397
40.50.1 Function Documentation	2398
40.50.1.1 set() [1/2]	2398
40.50.1.2 set() [2/2]	2399

40.51/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/CMakeLists.txt File Reference	2399
40.51.1 Function Documentation	2400
40.51.1.1 set() [1/2]	2400
40.51.1.2 set() [2/2]	2400
40.52/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/adaboost/CMakeLists.txt File Reference . .	2400
40.52.1 Function Documentation	2400
40.52.1.1 set() [1/2]	2400
40.52.1.2 set() [2/2]	2401
40.53/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/CMakeLists.txt File Reference	2401
40.53.1 Function Documentation	2401
40.53.1.1 set() [1/2]	2401
40.53.1.2 set() [2/2]	2401
40.54/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/init_rules/CMakeLists.txt File Reference	2402
40.54.1 Function Documentation	2402
40.54.1.1 set() [1/2]	2402
40.54.1.2 set() [2/2]	2402
40.55/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/CMakeLists.txt File Reference	2402
40.55.1 Function Documentation	2403
40.55.1.1 set() [1/2]	2403
40.55.1.2 set() [2/2]	2403
40.56/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/CMakeLists.txt File Refer- ence	2403
40.56.1 Function Documentation	2403
40.56.1.1 set() [1/2]	2403
40.56.1.2 set() [2/2]	2404
40.57/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/CMakeLists.txt File Reference	2404
40.57.1 Function Documentation	2404

40.57.1.1 set() [1/2]	2404
40.57.1.2 set() [2/2]	2404
40.58/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented/CMakeLists.txt File Reference	2404
40.58.1 Function Documentation	2405
40.58.1.1 set()	2405
40.59/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented/tasks/CMakeLists.txt File Reference	2405
40.59.1 Function Documentation	2405
40.59.1.1 set() [1/2]	2405
40.59.1.2 set() [2/2]	2405
40.60/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/CMakeLists.txt File Reference	2406
40.60.1 Function Documentation	2406
40.60.1.1 set() [1/2]	2406
40.60.1.2 set() [2/2]	2406
40.61/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/CMakeLists.txt File Reference	2406
40.61.1 Function Documentation	2407
40.61.1.1 set() [1/2]	2407
40.61.1.2 set() [2/2]	2407
40.62/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/dists/CMakeLists.txt File Reference	2407
40.62.1 Function Documentation	2407
40.62.1.1 set() [1/2]	2407
40.62.1.2 set() [2/2]	2408
40.63/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/CMakeLists.txt File Reference	2408
40.63.1 Function Documentation	2408
40.63.1.1 set() [1/2]	2408
40.63.1.2 set() [2/2]	2408
40.64/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/CMakeLists.txt File Reference	2409
40.64.1 Function Documentation	2409

40.64.1.1 set() [1/2]	2409
40.64.1.2 set() [2/2]	2410
40.65/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/CMakeLists.txt File Reference	2410
40.65.1 Function Documentation	2410
40.65.1.1 set() [1/2]	2410
40.65.1.2 set() [2/2]	2410
40.66/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/rbm/CMakeLists.txt File Reference	2411
40.66.1 Function Documentation	2411
40.66.1.1 set() [1/2]	2411
40.66.1.2 set() [2/2]	2411
40.67/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/CMakeLists.txt File Reference	2411
40.67.1 Function Documentation	2412
40.67.1.1 set() [1/2]	2412
40.67.1.2 set() [2/2]	2412
40.68/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/approx_kfn/CMakeLists.txt File Reference	2412
40.68.1 Function Documentation	2413
40.68.1.1 set() [1/2]	2413
40.68.1.2 set() [2/2]	2413
40.69/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/bias_svd/CMakeLists.txt File Reference	2413
40.69.1 Function Documentation	2413
40.69.1.1 set() [1/2]	2413
40.69.1.2 set() [2/2]	2414
40.70/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/block_krylov_svd/CMakeLists.txt File Reference	2414
40.70.1 Function Documentation	2414
40.70.1.1 set() [1/2]	2414
40.70.1.2 set() [2/2]	2414
40.71/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/CMakeLists.txt File Reference	2414

40.71.1 Function Documentation	2415
40.71.1.1 set() [1/2]	2415
40.71.1.2 set() [2/2]	2415
40.72/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/CMakeLists.txt File Reference	2415
40.72.1 Function Documentation	2415
40.72.1.1 set() [1/2]	2415
40.72.1.2 set() [2/2]	2416
40.73/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/interpolation_policies/CMakeLists.txt File Reference	2416
40.73.1 Function Documentation	2416
40.73.1.1 set() [1/2]	2416
40.73.1.2 set() [2/2]	2416
40.74/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/neighbor_search_policies/CMakeLists.txt File Reference	2416
40.74.1 Function Documentation	2417
40.74.1.1 set() [1/2]	2417
40.74.1.2 set() [2/2]	2417
40.75/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/CMakeLists.txt File Refer- ence	2417
40.75.1 Function Documentation	2417
40.75.1.1 set() [1/2]	2417
40.75.1.2 set() [2/2]	2418
40.76/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/CMakeLists.txt File Reference	2418
40.76.1 Function Documentation	2418
40.76.1.1 add_subdirectory()	2418
40.76.1.2 set()	2418
40.77/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/dbscan/CMakeLists.txt File Reference . . .	2419
40.77.1 Function Documentation	2419
40.77.1.1 set() [1/2]	2419

40.77.1.2 set() [2/2]	2419
40.78/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_stump/CMakeLists.txt File Reference	2419
40.78.1 Function Documentation	2420
40.78.1.1 set() [1/2]	2420
40.78.1.2 set() [2/2]	2420
40.79/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/CMakeLists.txt File Reference	2420
40.79.1 Function Documentation	2420
40.79.1.1 set() [1/2]	2421
40.79.1.2 set() [2/2]	2421
40.80/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/det/CMakeLists.txt File Reference	2421
40.80.1 Function Documentation	2421
40.80.1.1 set() [1/2]	2421
40.80.1.2 set() [2/2]	2422
40.81/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/CMakeLists.txt File Reference	2422
40.81.1 Function Documentation	2422
40.81.1.1 set() [1/2]	2422
40.81.1.2 set() [2/2]	2422
40.82/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/fastmks/CMakeLists.txt File Reference . . .	2423
40.82.1 Function Documentation	2423
40.82.1.1 set() [1/2]	2423
40.82.1.2 set() [2/2]	2423
40.83/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/CMakeLists.txt File Reference	2423
40.83.1 Function Documentation	2424
40.83.1.1 set() [1/2]	2424
40.83.1.2 set() [2/2]	2424
40.84/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hmm/CMakeLists.txt File Reference	2424
40.84.1 Function Documentation	2424
40.84.1.1 set() [1/2]	2424

40.84.1.2 set() [2/2]	2425
40.85/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/CMakeLists.txt File Reference	2425
40.85.1 Function Documentation	2425
40.85.1.1 set() [1/2]	2425
40.85.1.2 set() [2/2]	2425
40.86/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/CMakeLists.txt File Reference	2426
40.86.1 Function Documentation	2426
40.86.1.1 set() [1/2]	2426
40.86.1.2 set() [2/2]	2426
40.87/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kernel_pca/CMakeLists.txt File Reference	2426
40.87.1 Function Documentation	2427
40.87.1.1 set() [1/2]	2427
40.87.1.2 set() [2/2]	2427
40.88/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kernel_pca/kernel_rules/CMakeLists.txt File Reference	2427
40.88.1 Function Documentation	2427
40.88.1.1 set() [1/2]	2427
40.88.1.2 set() [2/2]	2428
40.89/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/CMakeLists.txt File Reference	2428
40.89.1 Function Documentation	2428
40.89.1.1 set() [1/2]	2428
40.89.1.2 set() [2/2]	2428
40.90/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lars/CMakeLists.txt File Reference	2429
40.90.1 Function Documentation	2429
40.90.1.1 set() [1/2]	2429
40.90.1.2 set() [2/2]	2429
40.91/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear_regression/CMakeLists.txt File Reference	2429
40.91.1 Function Documentation	2429

40.91.1.1 set() [1/2]	2430
40.91.1.2 set() [2/2]	2430
40.92/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear_svm/CMakeLists.txt File Reference	2430
40.92.1 Function Documentation	2430
40.92.1.1 set() [1/2]	2430
40.92.1.2 set() [2/2]	2431
40.93/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lmnn/CMakeLists.txt File Reference	2431
40.93.1 Function Documentation	2431
40.93.1.1 set() [1/2]	2431
40.93.1.2 set() [2/2]	2431
40.94/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/local_coordinate_coding/CMakeLists.txt File Reference	2432
40.94.1 Function Documentation	2432
40.94.1.1 set() [1/2]	2432
40.94.1.2 set() [2/2]	2432
40.95/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/logistic_regression/CMakeLists.txt File Reference	2432
40.95.1 Function Documentation	2433
40.95.1.1 set() [1/2]	2433
40.95.1.2 set() [2/2]	2433
40.96/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lsh/CMakeLists.txt File Reference	2433
40.96.1 Function Documentation	2433
40.96.1.1 set() [1/2]	2433
40.96.1.2 set() [2/2]	2434
40.97/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/matrix_completion/CMakeLists.txt File Reference	2434
40.97.1 Function Documentation	2434
40.97.1.1 set() [1/2]	2434
40.97.1.2 set() [2/2]	2434

40.98/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/mean_shift/CMakeLists.txt File Reference .	2434
40.98.1 Function Documentation	2435
40.98.1.1 set() [1/2]	2435
40.98.1.2 set() [2/2]	2435
40.99/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/naive_bayes/CMakeLists.txt File Reference	2435
40.99.1 Function Documentation	2435
40.99.1.1 set() [1/2]	2435
40.99.1.2 set() [2/2]	2436
40.100/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nca/CMakeLists.txt File Reference	2436
40.100.1 Function Documentation	2436
40.100.1.1 set() [1/2]	2436
40.100.1.2 set() [2/2]	2436
40.101/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/CMakeLists.txt File Reference	2437
40.101.1 Function Documentation	2437
40.101.1.1 set() [1/2]	2437
40.101.1.2 set() [2/2]	2437
40.102/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nmf/CMakeLists.txt File Reference	2437
40.102.1 Function Documentation	2438
40.102.1.1 add_cli_executable()	2438
40.103/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nystroem_method/CMakeLists.txt File Reference	2438
40.103.1 Function Documentation	2438
40.103.1.1 set() [1/2]	2438
40.103.1.2 set() [2/2]	2438
40.104/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/CMakeLists.txt File Reference	2439
40.104.1 Function Documentation	2439
40.104.1.1 set() [1/2]	2439
40.104.1.2 set() [2/2]	2439

40.105	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/decomposition_policies/CMakeLists.txt	
	File Reference	2439
40.105	Function Documentation	2440
40.105.1	1.set() [1/2]	2440
40.105.1	2.set() [2/2]	2440
40.106	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/CMakeLists.txt	
	File Reference	2440
40.106	Function Documentation	2440
40.106.1	1.set() [1/2]	2440
40.106.1	2.set() [2/2]	2441
40.107	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/initialization_methods/CMakeLists.txt	
	File Reference	2441
40.107	Function Documentation	2441
40.107.1	1.set() [1/2]	2441
40.107.1	2.set() [2/2]	2441
40.108	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/learning_policies/CMakeLists.txt	
	File Reference	2441
40.108	Function Documentation	2442
40.108.1	1.set() [1/2]	2442
40.108.1	2.set() [2/2]	2442
40.109	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/preprocess/CMakeLists.txt	
	File Reference	2442
40.109	Function Documentation	2442
40.109.1	1.set() [1/2]	2442
40.109.1	2.set() [2/2]	2443
40.110	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/quic_svd/CMakeLists.txt	
	File Reference	2443
40.110	Function Documentation	2443
40.110.1	1.set() [1/2]	2443
40.110.1	2.set() [2/2]	2443
40.111	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/radical/CMakeLists.txt	
	File Reference	2443
40.111	Function Documentation	2444

40.111.1.1set() [1/2]	2444
40.111.1.2set() [2/2]	2444
40.112home/barak/src/git/debian-src/mlpack/src/mlpack/methods/random_forest/CMakeLists.txt File Reference	2444
40.112. Function Documentation	2444
40.112.1.1set() [1/2]	2444
40.112.1.2set() [2/2]	2445
40.113home/barak/src/git/debian-src/mlpack/src/mlpack/methods/randomized_svd/CMakeLists.txt File Reference	2445
40.113. Function Documentation	2445
40.113.1.1set() [1/2]	2445
40.113.1.2set() [2/2]	2445
40.114home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/CMakeLists.txt File Reference	2446
40.114. Function Documentation	2446
40.114.1.1set() [1/2]	2446
40.114.1.2set() [2/2]	2446
40.115home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/CMakeLists.txt File Reference	2446
40.115. Function Documentation	2447
40.115.1.1set() [1/2]	2447
40.115.1.2set() [2/2]	2447
40.116home/barak/src/git/debian-src/mlpack/src/mlpack/methods/regularized_svd/CMakeLists.txt File Reference	2447
40.116. Function Documentation	2447
40.116.1.1set() [1/2]	2447
40.116.1.2set() [2/2]	2448
40.117home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/CMakeLists.txt File Reference	2448
40.117. Function Documentation	2448
40.117.1.1set() [1/2]	2448
40.117.1.2set() [2/2]	2448

40.118	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/CMakeLists.txt File Reference	2449
40.118	Function Documentation	2449
40.118.1	1.set() [1/2]	2449
40.118.1	2.set() [2/2]	2449
40.119	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/policy/CMakeLists.txt File Reference	2449
40.119	Function Documentation	2449
40.119.1	1.set() [1/2]	2450
40.119.1	2.set() [2/2]	2450
40.120	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/replay/CMakeLists.txt File Reference	2450
40.120	Function Documentation	2450
40.120.1	1.set() [1/2]	2450
40.120.1	2.set() [2/2]	2450
40.121	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/CMakeLists.txt File Reference	2451
40.121	Function Documentation	2451
40.121.1	1.set() [1/2]	2451
40.121.1	2.set() [2/2]	2451
40.122	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/softmax_regression/CMakeLists.txt File Reference	2451
40.122	Function Documentation	2452
40.122.1	1.set() [1/2]	2452
40.122.1	2.set() [2/2]	2452
40.123	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_autoencoder/CMakeLists.txt File Reference	2452
40.123	Function Documentation	2452
40.123.1	1.set() [1/2]	2452
40.123.1	2.set() [2/2]	2453
40.124	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_coding/CMakeLists.txt File Reference	2453

40.124. Function Documentation	2453
40.124.1.1.set() [1/2]	2453
40.124.1.2.set() [2/2]	2453
40.125. home/barak/src/git/debian-src/mlpack/src/mlpack/methods/svdplusplus/CMakeLists.txt File Reference	2454
40.125. Function Documentation	2454
40.125.1.1.set() [1/2]	2454
40.125.1.2.set() [2/2]	2454
40.126. home/barak/src/git/debian-src/mlpack/src/mlpack/tests/CMakeLists.txt File Reference	2454
40.126. Function Documentation	2455
40.126.1.1.add_executable()	2455
40.127. home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/default_param.hpp File Reference	2456
40.128. home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/default_param.hpp File Reference	2458
40.129. home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/default_param.hpp File Reference .	2459
40.130. home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/delete_allocated_memory.hpp File Reference	2460
40.131. home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/delete_allocated_memory.hpp File Reference	2461
40.132. home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/end_program.hpp File Reference	2462
40.132.1. Detailed Description	2463
40.133. home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get_allocated_memory.hpp File Reference	2463
40.134. home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/get_allocated_memory.hpp File Reference	2464
40.135. home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get_param.hpp File Reference	2465
40.136. home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/get_param.hpp File Reference .	2467
40.137. home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/get_param.hpp File Reference . . .	2468
40.138. home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/get_param.hpp File Reference	2469
40.139. home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get_printable_param.hpp File Reference	2470
40.140. home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/get_printable_param.hpp File Reference	2471

40.141	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/get_printable_param.hpp File Reference	2472
40.142	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/get_printable_param.hpp File Reference	2474
40.143	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get_printable_param_name.hpp File Reference	2475
40.144	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/get_printable_param_name.hpp File Reference	2477
40.145	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get_printable_param_value.hpp File Reference	2478
40.146	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/get_printable_param_value.hpp File Reference	2479
40.147	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get_printable_type.hpp File Reference	2481
40.148	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/get_printable_type.hpp File Reference	2482
40.149	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/get_printable_type.hpp File Reference	2483
40.150	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get_raw_param.hpp File Reference	2485
40.150.	Detailed Description	2487
40.151	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/map_parameter_name.hpp File Reference	2487
40.151.	Detailed Description	2488
40.152	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/output_param.hpp File Reference	2488
40.152.	Detailed Description	2490
40.153	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/parameter_type.hpp File Reference	2490
40.153.	Detailed Description	2491
40.154	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/parse_command_line.hpp File Reference	2491
40.154.	Detailed Description	2492
40.155	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/print_doc_functions.hpp File Reference	2492
40.156	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/print_doc_functions.hpp File Reference	2494
40.157	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/print_doc_functions.hpp File Reference	2495
40.158	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/print_help.hpp File Reference	2496
40.158.	Detailed Description	2497

40.159	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/print_type_doc.hpp File Reference . . .	2497
40.160	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/print_type_doc.hpp File Reference	2499
40.161	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/print_type_doc.hpp File Reference .	2500
40.162	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/set_param.hpp File Reference	2501
40.163	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/string_type_param.hpp File Reference .	2503
40.163.1	Detailed Description	2504
40.164	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/binding_info.hpp File Reference	2504
40.164.1	Detailed Description	2505
40.165	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/get_binding_name.hpp File Reference	2505
40.166	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/is_serializable.hpp File Reference	2506
40.166.1	Detailed Description	2507
40.167	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/md_option.hpp File Reference .	2507
40.167.1	Detailed Description	2508
40.168	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/print_docs.hpp File Reference .	2508
40.168.1	Detailed Description	2508
40.168.2	Function Documentation	2509
40.168.2.1	PrintDocs()	2509
40.168.2.2	PrintHeaders()	2509
40.169	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/program_doc_wrapper.hpp File Reference	2509
40.169.1	Detailed Description	2510
40.170	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/get_arma_type.hpp File Reference .	2510
40.170.1	Detailed Description	2511
40.171	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/get_cython_type.hpp File Reference	2512
40.171.1	Detailed Description	2513
40.172	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/get_numpy_type.hpp File Reference	2513
40.172.1	Detailed Description	2514

40.173	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/get_numpy_type_char.hpp File Reference	2515
40.173.1	Detailed Description	2516
40.174	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/import_decl.hpp File Reference	2516
40.174.1	Detailed Description	2518
40.175	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/mlpack/arma_util.hpp File Reference	2518
40.175.1	Detailed Description	2518
40.175.2	Function Documentation	2519
40.175.2.1	GetMemory()	2519
40.175.2.2	GetMemState()	2519
40.175.2.3	SetMemState()	2519
40.176	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/mlpack/cli_util.hpp File Reference	2519
40.176.1	Detailed Description	2520
40.177	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/mlpack/serialization.hpp File Reference	2521
40.178	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/serialization.hpp File Reference	2521
40.179	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/print_class_defn.hpp File Reference	2522
40.179.1	Detailed Description	2524
40.180	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/print_defn.hpp File Reference	2524
40.180.1	Detailed Description	2525
40.181	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/print_doc.hpp File Reference	2525
40.181.1	Detailed Description	2526
40.182	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/print_input_processing.hpp File Reference	2526
40.182.1	Detailed Description	2528
40.183	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/print_output_processing.hpp File Reference	2528
40.183.1	Detailed Description	2529
40.184	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/print_pyx.hpp File Reference	2530
40.185	home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/py_option.hpp File Reference	2530

40.185. Detailed Description	2531
40.186. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/strip_type.hpp</code> File Reference . . .	2531
40.186. Detailed Description	2532
40.187. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/clean_memory.hpp</code> File Reference . .	2532
40.187. Detailed Description	2533
40.188. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/ignore_check.hpp</code> File Reference . . .	2533
40.188. Detailed Description	2533
40.189. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/test_option.hpp</code> File Reference . . .	2534
40.189. Detailed Description	2534
40.190. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/core.hpp</code> File Reference	2535
40.190. Detailed Description	2535
40.191. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/cv_base.hpp</code> File Reference	2535
40.191. Detailed Description	2536
40.192. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/k_fold_cv.hpp</code> File Reference	2536
40.192. Detailed Description	2537
40.193. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/meta_info_extractor.hpp</code> File Reference . . .	2537
40.193. Detailed Description	2538
40.194. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/accuracy.hpp</code> File Reference	2539
40.194. Detailed Description	2539
40.195. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/average_strategy.hpp</code> File Reference	2539
40.195. Detailed Description	2540
40.196. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/f1.hpp</code> File Reference	2540
40.196. Detailed Description	2541
40.197. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/facilities.hpp</code> File Reference	2541
40.197. Detailed Description	2542
40.198. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/mse.hpp</code> File Reference	2542
40.198. Detailed Description	2542
40.199. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/precision.hpp</code> File Reference	2543

40.199. Detailed Description	2543
40.200home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/recall.hpp File Reference	2543
40.200. Detailed Description	2544
40.201home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/simple_cv.hpp File Reference	2544
40.201. Detailed Description	2545
40.202home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/binarize.hpp File Reference	2545
40.202. Detailed Description	2546
40.203home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/confusion_matrix.hpp File Reference	2546
40.204home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/dataset_mapper.hpp File Reference	2547
40.204. Detailed Description	2547
40.205home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/extension.hpp File Reference	2548
40.205. Detailed Description	2548
40.206home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/format.hpp File Reference	2549
40.207home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/has_serialize.hpp File Reference	2549
40.207. Detailed Description	2550
40.208home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/custom_imputation.hpp File Reference	2550
40.208. Detailed Description	2551
40.209home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/listwise_deletion.hpp File Reference	2551
40.209. Detailed Description	2552
40.210home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/mean_imputation.hpp File Reference	2552
40.210. Detailed Description	2552
40.211home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/median_imputation.hpp File Reference	2553
40.211. Detailed Description	2553
40.212home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputer.hpp File Reference	2553
40.212. Detailed Description	2554
40.213home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/is_naninf.hpp File Reference	2554

40.213. Detailed Description	2555
40.214. home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/load.hpp File Reference	2555
40.214. Detailed Description	2556
40.215. home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/load_arff.hpp File Reference	2556
40.215. Detailed Description	2557
40.216. home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/load_csv.hpp File Reference	2557
40.216. Detailed Description	2558
40.217. home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/map_policies/datatype.hpp File Reference	2558
40.218. home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/map_policies/increment_policy.hpp File Reference	2559
40.218. Detailed Description	2559
40.219. home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/map_policies/missing_policy.hpp File Reference	2560
40.219. Detailed Description	2561
40.220. home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/normalize_labels.hpp File Reference	2561
40.220. Detailed Description	2562
40.221. home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/one_hot_encoding.hpp File Reference	2562
40.221. Detailed Description	2563
40.222. home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/save.hpp File Reference	2563
40.222. Detailed Description	2564
40.223. home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/serialization_template_version.hpp File Reference	2564
40.223. Detailed Description	2564
40.223.2. Macro Definition Documentation	2565
40.223.2.1. BOOST_TEMPLATE_CLASS_VERSION	2565
40.224. home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/split_data.hpp File Reference	2565
40.224. Detailed Description	2566
40.225. home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/diagonal_gaussian_distribution.hpp File Reference	2566
40.225. Detailed Description	2567

40.226	home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/discrete_distribution.hpp File Reference	2567
40.226	Detailed Description	2568
40.227	home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/gamma_distribution.hpp File Reference	2568
40.227	Detailed Description	2569
40.228	home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/gaussian_distribution.hpp File Reference	2569
40.228	Detailed Description	2570
40.229	home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/laplace_distribution.hpp File Reference	2570
40.230	home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/regression_distribution.hpp File Reference	2571
40.230	Detailed Description	2572
40.231	home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/cv_function.hpp File Reference	2572
40.231	Detailed Description	2572
40.232	home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/deduce_hp_types.hpp File Reference	2573
40.232	Detailed Description	2574
40.233	home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/fixed.hpp File Reference	2574
40.233	Detailed Description	2576
40.234	home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/hpt.hpp File Reference	2576
40.235	home/barak/src/git/debian-src/mlpack/doc/guide/hpt.hpp File Reference	2577
40.236	home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/cauchy_kernel.hpp File Reference	2577
40.236	Detailed Description	2578
40.237	home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/cosine_distance.hpp File Reference	2578
40.237	Detailed Description	2579
40.238	home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/epanechnikov_kernel.hpp File Reference	2579
40.238	Detailed Description	2580
40.239	home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/example_kernel.hpp File Reference	2580
40.239	Detailed Description	2580
40.240	home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/gaussian_kernel.hpp File Reference	2581
40.240	Detailed Description	2581

40.241	home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/hyperbolic_tangent_kernel.hpp File Reference	2582
40.241.1	Detailed Description	2582
40.242	home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/kernel_traits.hpp File Reference	2583
40.242.1	Detailed Description	2583
40.243	home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/laplacian_kernel.hpp File Reference	2583
40.243.1	Detailed Description	2584
40.244	home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/linear_kernel.hpp File Reference	2584
40.244.1	Detailed Description	2585
40.245	home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/polynomial_kernel.hpp File Reference	2585
40.245.1	Detailed Description	2586
40.246	home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/pspectrum_string_kernel.hpp File Reference	2586
40.246.1	Detailed Description	2587
40.247	home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/spherical_kernel.hpp File Reference	2587
40.247.1	Detailed Description	2588
40.248	home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/triangular_kernel.hpp File Reference	2588
40.248.1	Detailed Description	2589
40.249	home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/ccov.hpp File Reference	2589
40.249.1	Detailed Description	2590
40.250	home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/clamp.hpp File Reference	2590
40.250.1	Detailed Description	2591
40.251	home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/columns_to_blocks.hpp File Reference	2591
40.251.1	Detailed Description	2592
40.252	home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/lin_alg.hpp File Reference	2592
40.252.1	Detailed Description	2593
40.253	home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/log_add.hpp File Reference	2594
40.253.1	Detailed Description	2594
40.254	home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/make_alias.hpp File Reference	2594

40.254. Detailed Description	2595
40.255. home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/random.hpp File Reference	2596
40.255. Detailed Description	2597
40.256. home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/random_basis.hpp File Reference	2597
40.256. Detailed Description	2598
40.257. home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/range.hpp File Reference	2598
40.257. Detailed Description	2599
40.258. home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/round.hpp File Reference	2599
40.258. Detailed Description	2599
40.259. home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/shuffle_data.hpp File Reference	2600
40.259. Detailed Description	2601
40.260. home/barak/src/git/debian-src/mlpack/src/mlpack/core/metrics/lp_metric.hpp File Reference	2601
40.260. Detailed Description	2602
40.261. home/barak/src/git/debian-src/mlpack/src/mlpack/core/metrics/lmetric.hpp File Reference	2603
40.261. Detailed Description	2604
40.262. home/barak/src/git/debian-src/mlpack/src/mlpack/core/metrics/mahalanobis_distance.hpp File Reference	2604
40.263. home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/address.hpp File Reference	2604
40.263. Detailed Description	2605
40.264. home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ballbound.hpp File Reference	2606
40.264. Detailed Description	2606
40.265. home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/binary_space_tree.hpp File Reference	2607
40.266. home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree.hpp File Reference	2608
40.267. home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/breadth_first_dual_↵ tree_traverser.hpp File Reference	2608
40.267. Detailed Description	2609
40.268. home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/dual_tree_traverser.hpp File Reference	2610
40.269. home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/dual_tree_traverser.hpp File Reference	2611

40.270	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree/dual_tree_traverser.hpp File Reference	2611
40.271	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/dual_tree_traverser.hpp File Reference	2612
40.272	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/mean_split.hpp File Reference	2613
40.272.1	Detailed Description	2614
40.273	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/midpoint_split.hpp File Reference	2615
40.273.1	Detailed Description	2615
40.274	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/rp_tree_max_split.hpp File Reference	2616
40.274.1	Detailed Description	2616
40.275	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/rp_tree_mean_split.hpp File Reference	2617
40.275.1	Detailed Description	2617
40.276	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/single_tree_traverser.hpp File Reference	2618
40.277	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/single_tree_traverser.hpp File Reference	2619
40.278	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree/single_tree_traverser.hpp File Reference	2620
40.279	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/single_tree_traverser.hpp File Reference	2621
40.280	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/traits.hpp File Reference	2622
40.281	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/traits.hpp File Reference	2623
40.282	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree/traits.hpp File Reference	2624
40.283	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/traits.hpp File Reference . .	2625
40.284	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/traits.hpp File Reference	2626
40.285	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/typedef.hpp File Reference	2627
40.286	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/typedef.hpp File Reference . .	2629
40.287	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/typedef.hpp File Reference	2630
40.288	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/typedef.hpp File Reference . . .	2631

40.289	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/typedef.hpp File Reference	2633
40.290	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/typedef.hpp File Reference	2633
40.291	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ub_tree_split.hpp File Reference	2634
40.291.1	Detailed Description	2635
40.292	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/vantage_point_split.hpp File Reference	2636
40.292.1	Detailed Description	2636
40.293	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/bound_traits.hpp File Reference	2637
40.293.1	Detailed Description	2637
40.294	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/bounds.hpp File Reference	2638
40.294.1	Detailed Description	2638
40.295	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cellbound.hpp File Reference	2638
40.295.1	Detailed Description	2639
40.296	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cosine_tree/cosine_tree.hpp File Reference	2640
40.296.1	Detailed Description	2641
40.297	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/cover_tree.hpp File Reference	2641
40.298	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree.hpp File Reference	2642
40.299	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/first_point_is_root.hpp File Reference	2643
40.299.1	Detailed Description	2644
40.300	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/enumerate_tree.hpp File Reference	2644
40.300.1	Detailed Description	2644
40.301	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/example_tree.hpp File Reference	2645
40.301.1	Detailed Description	2645
40.302	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/greedy_single_tree_traverser.hpp File Reference	2645
40.303	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/hollow_ball_bound.hpp File Reference	2646
40.303.1	Detailed Description	2647
40.304	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/hrectbound.hpp File Reference	2647

40.304. Detailed Description	2648
40.305home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree/octree.hpp File Reference	2648
40.306home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree.hpp File Reference	2650
40.307home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/perform_split.hpp File Reference	2650
40.307. Detailed Description	2651
40.308home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/discrete_hilbert_value.hpp File Reference	2651
40.308. Detailed Description	2652
40.309home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/hilbert_r_tree_auxiliary_↔ information.hpp File Reference	2653
40.309. Detailed Description	2654
40.310home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/hilbert_r_tree_descent_↔ heuristic.hpp File Reference	2654
40.310. Detailed Description	2655
40.311home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/hilbert_r_tree_split.hpp File Reference	2655
40.311. Detailed Description	2656
40.312home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/minimal_coverage_↔ sweep.hpp File Reference	2656
40.312. Detailed Description	2657
40.313home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/minimal_splits_number_↔ sweep.hpp File Reference	2658
40.313. Detailed Description	2659
40.314home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/no_auxiliary_information.hpp File Reference	2659
40.314. Detailed Description	2660
40.315home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/r_plus_plus_tree_auxiliary_↔ _information.hpp File Reference	2660
40.315. Detailed Description	2661
40.316home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/r_plus_plus_tree_descent_↔ _heuristic.hpp File Reference	2662
40.316. Detailed Description	2662

40.317	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/r_plus_plus_tree_split_↔ policy.hpp File Reference	2663
40.317	Detailed Description	2663
40.318	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/r_plus_tree_descent_↔ heuristic.hpp File Reference	2664
40.318	Detailed Description	2664
40.319	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/r_plus_tree_split.hpp File Reference	2665
40.319	Detailed Description	2665
40.320	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/r_plus_tree_split_policy.hpp File Reference	2666
40.320	Detailed Description	2666
40.321	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/r_star_tree_descent_↔ heuristic.hpp File Reference	2667
40.321	Detailed Description	2667
40.322	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/r_star_tree_split.hpp File Reference	2668
40.323	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/r_tree_descent_heuristic.hpp File Reference	2669
40.323	Detailed Description	2670
40.324	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/r_tree_split.hpp File Reference	2670
40.324	Detailed Description	2671
40.325	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/rectangle_tree.hpp File Ref- erence	2671
40.326	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree.hpp File Reference	2672
40.327	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/x_tree_auxiliary_information.hpp File Reference	2673
40.328	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/x_tree_split.hpp File Refer- ence	2673
40.329	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/hyperplane.hpp File Reference	2674
40.329	Detailed Description	2676
40.330	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/mean_space_split.hpp File Ref- erence	2676

40.330.	Detailed Description	2677
40.331	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/midpoint_space_split.hpp File Reference	2678
40.332	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/projection_vector.hpp File Reference	2679
40.332.	Detailed Description	2680
40.333	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/space_split.hpp File Reference	2680
40.333.	Detailed Description	2681
40.334	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/is_spill_tree.hpp File Reference	2681
40.334.	Detailed Description	2682
40.335	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/spill_dual_tree_traverser.hpp File Reference	2683
40.335.	Detailed Description	2684
40.336	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/spill_single_tree_traverser.hpp File Reference	2684
40.336.	Detailed Description	2685
40.337	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/spill_tree.hpp File Reference	2686
40.338	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree.hpp File Reference	2687
40.339	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/statistic.hpp File Reference	2688
40.339.	Detailed Description	2688
40.340	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/traversal_info.hpp File Reference	2689
40.340.	Detailed Description	2689
40.341	home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/tree_traits.hpp File Reference	2690
40.341.	Detailed Description	2690
40.342	home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/arma_config.hpp File Reference	2690
40.342.	Detailed Description	2691
40.342.1	Macro Definition Documentation	2691
40.342.2.1	MLPACK_ARMA_NO64BIT_WORD	2691
40.342.2.2	MLPACK_ARMA_USE_OPENMP	2691
40.343	home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/arma_config_check.hpp File Reference	2692

40.343. Detailed Description	2692
40.344. home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/arma_traits.hpp File Reference	2692
40.344. Detailed Description	2693
40.345. home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/backtrace.hpp File Reference	2693
40.345. Detailed Description	2694
40.346. home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/cli.hpp File Reference	2694
40.346. Detailed Description	2695
40.347. home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/deprecated.hpp File Reference	2695
40.347. Detailed Description	2695
40.347. Macro Definition Documentation	2695
40.347.2. mlpack_deprecated	2696
40.348. home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/gitversion.hpp File Reference	2696
40.348. Variable Documentation	2696
40.348.1. git	2696
40.349. home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/hyphenate_string.hpp File Reference	2696
40.349. Detailed Description	2697
40.350. home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/is_std_vector.hpp File Reference	2697
40.350. Detailed Description	2698
40.351. home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/log.hpp File Reference	2698
40.351. Detailed Description	2699
40.352. home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/mlpack_main.hpp File Reference	2699
40.352. Macro Definition Documentation	2700
40.352.1. BINDING_TYPE	2700
40.352.1.2 BINDING_TYPE_CLI	2700
40.352.1.3 BINDING_TYPE_MARKDOWN	2701
40.352.1.4 BINDING_TYPE_PYX	2701
40.352.1.5 BINDING_TYPE_TEST	2701
40.352.1.6 BINDING_TYPE_UNKNOWN	2701

40.353	home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/nullostream.hpp File Reference	2702
40.353.1	Detailed Description	2702
40.354	home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/param.hpp File Reference	2703
40.354.1	Detailed Description	2705
40.354.2	Macro Definition Documentation	2705
40.354.2.1	PARAM_COL	2705
40.354.2.2	PARAM_COL_IN	2706
40.354.2.3	PARAM_COL_IN_REQ	2706
40.354.2.4	PARAM_COL_OUT	2707
40.354.2.5	PARAM_DOUBLE_IN	2708
40.354.2.6	PARAM_DOUBLE_IN_REQ	2708
40.354.2.7	PARAM_DOUBLE_OUT	2709
40.354.2.8	PARAM_FLAG	2710
40.354.2.9	PARAM_IN	2710
40.354.2.10	PARAM_INT_IN	2711
40.354.2.11	PARAM_INT_IN_REQ	2712
40.354.2.12	PARAM_INT_OUT	2712
40.354.2.13	PARAM_MATRIX	2714
40.354.2.14	PARAM_MATRIX_AND_INFO_IN	2714
40.354.2.15	PARAM_MATRIX_IN	2715
40.354.2.16	PARAM_MATRIX_IN_REQ	2715
40.354.2.17	PARAM_MATRIX_OUT	2716
40.354.2.18	PARAM_MODEL	2717
40.354.2.19	PARAM_MODEL_IN	2717
40.354.2.20	PARAM_MODEL_IN_REQ	2718
40.354.2.21	PARAM_MODEL_OUT	2719
40.354.2.22	PARAM_OUT	2719
40.354.2.23	PARAM_ROW	2720

40.354.2.24	PARAM_ROW_IN	2720
40.354.2.25	PARAM_ROW_OUT	2721
40.354.2.26	PARAM_STRING_IN	2721
40.354.2.27	PARAM_STRING_IN_REQ	2722
40.354.2.28	PARAM_STRING_OUT	2723
40.354.2.29	PARAM_TMATRIX_IN	2723
40.354.2.30	PARAM_TMATRIX_IN_REQ	2724
40.354.2.31	PARAM_TMATRIX_OUT	2725
40.354.2.32	PARAM_UCOL	2725
40.354.2.33	PARAM_UCOL_IN	2726
40.354.2.34	PARAM_UCOL_OUT	2727
40.354.2.35	PARAM_UMATRIX	2727
40.354.2.36	PARAM_UMATRIX_IN	2728
40.354.2.37	PARAM_UMATRIX_IN_REQ	2728
40.354.2.38	PARAM_UMATRIX_OUT	2729
40.354.2.39	PARAM_UROW	2730
40.354.2.40	PARAM_UROW_IN	2730
40.354.2.41	PARAM_UROW_OUT	2731
40.354.2.42	PARAM_VECTOR_IN	2732
40.354.2.43	PARAM_VECTOR_IN_REQ	2732
40.354.2.44	PARAM_VECTOR_OUT	2733
40.354.2.45	PROGRAM_INFO	2734
40.354.2.46	SEE_ALSO	2734
40.354.2.47	TUPLE_TYPE	2735
40.355	home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/param_checks.hpp File Reference	2736
40.355.1	Detailed Description	2737
40.356	home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/param_data.hpp File Reference	2737
40.356.1	Detailed Description	2738

40.356.2	Macro Definition Documentation	2738
40.356.2.1	TYPENAME	2739
40.357	home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/prefixedostream.hpp File Reference . . .	2739
40.357.1	Detailed Description	2740
40.358	home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/program_doc.hpp File Reference	2740
40.358.1	Detailed Description	2740
40.359	home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/sfinae_utility.hpp File Reference	2741
40.359.1	Detailed Description	2742
40.359.2	Macro Definition Documentation	2742
40.359.2.1	HAS_ANY_METHOD_FORM	2742
40.359.2.2	HAS_EXACT_METHOD_FORM	2743
40.359.2.3	HAS_MEM_FUNC	2744
40.359.2.4	HAS_METHOD_FORM	2744
40.359.2.5	HAS_METHOD_FORM_BASE	2745
40.359.2.6	SINGLE_ARG	2745
40.360	home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/timers.hpp File Reference	2745
40.360.1	Detailed Description	2746
40.361	home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/version.hpp File Reference	2746
40.361.1	Macro Definition Documentation	2747
40.361.1.1	MLPACK_VERSION_MAJOR	2747
40.361.1.2	MLPACK_VERSION_MINOR	2747
40.361.1.3	MLPACK_VERSION_PATCH	2748
40.362	home/barak/src/git/debian-src/mlpack/doc/guide/version.hpp File Reference	2748
40.363	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/adaboost/adaboost.hpp File Reference . .	2748
40.363.1	Detailed Description	2749
40.364	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/adaboost/adaboost_model.hpp File Reference	2749
40.364.1	Detailed Description	2750
40.365	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/amf.hpp File Reference	2750

40.365. Detailed Description	2751
40.366. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/init_rules/average_init.hpp</code> File Reference	2752
40.367. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/init_rules/given_init.hpp</code> File Reference	2752
40.368. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/init_rules/random_acol_init.hpp</code> File Reference	2753
40.368. Detailed Description	2754
40.369. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/init_rules/random_init.hpp</code> File Reference	2754
40.370. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/random_init.hpp</code> File Reference	2755
40.371. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/initialization_methods/random_↵_init.hpp</code> File Reference	2756
40.372. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/complete_↵_incremental_termination.hpp</code> File Reference	2757
40.372. Detailed Description	2757
40.373. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/incomplete_↵_incremental_termination.hpp</code> File Reference	2757
40.373. Detailed Description	2758
40.374. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/max_iteration_↵_termination.hpp</code> File Reference	2758
40.374. Detailed Description	2759
40.375. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/simple_residue_↵_termination.hpp</code> File Reference	2759
40.375. Detailed Description	2760
40.376. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/simple_tolerance_↵_termination.hpp</code> File Reference	2760
40.376. Detailed Description	2761
40.377. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/validation_rmse_↵_termination.hpp</code> File Reference	2762
40.378. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/nmf_als.hpp</code> File Reference	2762
40.378. Detailed Description	2763
40.379. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/nmf_mult_dist.hpp</code> File Reference	2764
40.379. Detailed Description	2765

40.380	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/nmf_mult_div.hpp	File Reference	2765
40.381	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/svd_batch_learning.hpp	File Reference	2766
40.381	Detailed Description		2767
40.382	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/svd_complete_incremental_↵_learning.hpp	File Reference	2767
40.382	Detailed Description		2768
40.383	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/svd_incomplete_↵_incremental_learning.hpp	File Reference	2769
40.383	Detailed Description		2770
40.384	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/hard_sigmoid_↵_function.hpp	File Reference	2770
40.384	Detailed Description		2771
40.385	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/identity_function.hpp	File Reference	2771
40.385	Detailed Description		2772
40.386	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/logistic_function.hpp	File Reference	2772
40.386	Detailed Description		2773
40.387	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/rectifier_function.hpp	File Reference	2773
40.387	Detailed Description		2774
40.388	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/softplus_function.hpp	File Reference	2774
40.388	Detailed Description		2775
40.389	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/softsign_function.hpp	File Reference	2775
40.389	Detailed Description		2776
40.390	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/swish_function.hpp	File Reference	2776
40.390	Detailed Description		2777
40.391	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/tanh_function.hpp	File Reference	2777

40.391. Detailed Description	2778
40.392 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented/tasks/add.hpp File Reference	2778
40.393 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/add.hpp File Reference	2779
40.394 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented/tasks/copy.hpp File Reference	2779
40.394. Detailed Description	2780
40.395 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented/tasks/score.hpp File Reference	2780
40.395. Detailed Description	2781
40.396 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented/tasks/sort.hpp File Reference	2781
40.396. Detailed Description	2782
40.397 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/brnn.hpp File Reference	2782
40.397. Detailed Description	2783
40.398 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/border_modes.hpp File Reference	2783
40.398. Detailed Description	2783
40.399 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/fft_convolution.hpp File Reference	2784
40.399. Detailed Description	2784
40.400 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/naive_convolution.hpp File Reference	2785
40.400. Detailed Description	2785
40.401 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/svd_convolution.hpp File Reference	2786
40.401. Detailed Description	2786
40.402 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/dists/bernoulli_distribution.hpp File Reference	2787
40.402. Detailed Description	2788
40.403 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/ffn.hpp File Reference	2788
40.403. Detailed Description	2788
40.404 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/const_init.hpp File Reference	2789
40.404. Detailed Description	2789

40.405	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/gaussian_init.hpp File Reference	2790
40.405.1	Detailed Description	2791
40.406	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/glorot_init.hpp File Reference	2791
40.406.1	Detailed Description	2792
40.407	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/he_init.hpp File Reference	2792
40.407.1	Detailed Description	2792
40.408	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/init_rules_traits.hpp File Reference	2793
40.408.1	Detailed Description	2793
40.409	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/kathirvalavakumar_subavathi_init.hpp File Reference	2794
40.409.1	Detailed Description	2794
40.410	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/lecun_normal_init.hpp File Reference	2795
40.410.1	Detailed Description	2795
40.411	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/network_init.hpp File Reference	2796
40.411.1	Detailed Description	2796
40.412	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/nguyen_widrow_init.hpp File Reference	2797
40.412.1	Detailed Description	2797
40.413	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/oivs_init.hpp File Reference	2798
40.413.1	Detailed Description	2798
40.414	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/orthogonal_init.hpp File Reference	2799
40.414.1	Detailed Description	2799
40.415	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/add_merge.hpp File Reference	2800
40.415.1	Detailed Description	2801
40.416	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/alpha_dropout.hpp File Reference	2801
40.416.1	Detailed Description	2802

40.417	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/atrous_convolution.hpp File Reference	2802
40.417	Detailed Description	2803
40.418	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/base_layer.hpp File Reference	2803
40.418	Detailed Description	2804
40.419	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/batch_norm.hpp File Reference	2805
40.419	Detailed Description	2805
40.420	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/bilinear_interpolation.hpp File Reference	2806
40.420	Detailed Description	2806
40.421	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/c_relu.hpp File Reference	2807
40.422	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/concat.hpp File Reference	2807
40.422	Detailed Description	2808
40.423	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/concat_performance.hpp File Reference	2809
40.423	Detailed Description	2809
40.424	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/concatenate.hpp File Reference	2810
40.424	Detailed Description	2810
40.425	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/constant.hpp File Reference	2811
40.425	Detailed Description	2811
40.426	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/convolution.hpp File Reference	2812
40.426	Detailed Description	2812
40.427	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/dropconnect.hpp File Reference	2813
40.427	Detailed Description	2813
40.428	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/dropout.hpp File Reference	2814
40.428	Detailed Description	2814
40.429	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/elu.hpp File Reference	2815
40.429	Detailed Description	2815
40.430	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/fast_lstm.hpp File Reference	2816

40.430. Detailed Description	2816
40.431home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/flexible_relu.hpp File Reference	2817
40.431. Detailed Description	2817
40.432home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/glimpse.hpp File Reference	2818
40.432. Detailed Description	2818
40.433home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/gru.hpp File Reference	2819
40.433. Detailed Description	2820
40.434home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/hard_tanh.hpp File Reference	2820
40.434. Detailed Description	2821
40.435home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/join.hpp File Reference	2821
40.435. Detailed Description	2822
40.436home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/layer.hpp File Reference	2822
40.436. Detailed Description	2823
40.437home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/layer_norm.hpp File Reference	2823
40.437. Detailed Description	2824
40.438home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/layer_traits.hpp File Reference	2824
40.438. Detailed Description	2825
40.439home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/layer_types.hpp File Reference	2826
40.439. Detailed Description	2828
40.440home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/leaky_relu.hpp File Reference	2828
40.440. Detailed Description	2829
40.441home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear.hpp File Reference	2829
40.441. Detailed Description	2830
40.442home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear_no_bias.hpp File Reference	2830
40.443home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/log_softmax.hpp File Reference	2831
40.443. Detailed Description	2832
40.444home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/lookup.hpp File Reference	2832
40.444. Detailed Description	2833

40.445	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/lstm.hpp File Reference	2833
40.445	Detailed Description	2834
40.446	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/max_pooling.hpp File Reference	2834
40.446	Detailed Description	2835
40.447	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/mean_pooling.hpp File Reference	2835
40.447	Detailed Description	2836
40.448	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/multiply_constant.hpp File Reference	2836
40.448	Detailed Description	2837
40.449	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/multiply_merge.hpp File Reference	2837
40.449	Detailed Description	2838
40.450	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/parametric_relu.hpp File Reference	2838
40.450	Detailed Description	2839
40.451	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/recurrent.hpp File Reference	2839
40.451	Detailed Description	2840
40.452	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/recurrent_attention.hpp File Reference	2840
40.452	Detailed Description	2841
40.453	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/reinforce_normal.hpp File Reference	2841
40.453	Detailed Description	2842
40.454	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/reparametrization.hpp File Reference	2842
40.454	Detailed Description	2843
40.455	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/select.hpp File Reference	2843
40.455	Detailed Description	2844
40.456	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/sequential.hpp File Reference	2844
40.456	Detailed Description	2845
40.457	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/subview.hpp File Reference	2846
40.457	Detailed Description	2846

40.458	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/transposed_convolution.hpp	File Reference	2847
40.458	Detailed Description		2847
40.459	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/vr_class_reward.hpp	File Reference	2848
40.459	Detailed Description		2848
40.460	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/cross_entropy_error.hpp	File Reference	2849
40.460	Detailed Description		2849
40.461	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/dice_loss.hpp	File Reference	2849
40.461	Detailed Description		2850
40.462	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/earth_mover_distance.hpp	File Reference	2850
40.462	Detailed Description		2851
40.463	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/kl_divergence.hpp	File Reference	2851
40.463	Detailed Description		2852
40.464	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/mean_squared_error.hpp	File Reference	2852
40.464	Detailed Description		2852
40.465	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/negative_log_likelihood.hpp	File Reference	2853
40.465	Detailed Description		2853
40.466	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/reconstruction_loss.hpp	File Reference	2854
40.466	Detailed Description		2854
40.467	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/softmax_cross_entropy_error.hpp	File Reference	2854
40.467	Detailed Description		2855
40.468	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/rbm/rbm.hpp	File Reference	2855
40.468	Detailed Description		2856
40.469	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/rbm/rbm_policies.hpp	File Reference	2856

40.469. Detailed Description	2857
40.470. home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/rnn.hpp File Reference	2857
40.470. Detailed Description	2858
40.471. home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/add_visitor.hpp File Reference .	2858
40.471. Detailed Description	2858
40.472. home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/backward_visitor.hpp File Reference	2859
40.472. Detailed Description	2859
40.473. home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/copy_visitor.hpp File Reference	2860
40.473. Detailed Description	2860
40.474. home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/delete_visitor.hpp File Reference	2861
40.474. Detailed Description	2861
40.475. home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/delta_visitor.hpp File Reference	2862
40.475. Detailed Description	2863
40.476. home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/deterministic_set_visitor.hpp File Reference	2863
40.476. Detailed Description	2864
40.477. home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/forward_visitor.hpp File Reference	2864
40.477. Detailed Description	2864
40.478. home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/gradient_set_visitor.hpp File Reference	2865
40.478. Detailed Description	2865
40.479. home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/gradient_update_visitor.hpp File Reference	2866
40.479. Detailed Description	2866
40.480. home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/gradient_visitor.hpp File Reference	2867
40.480. Detailed Description	2867
40.481. home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/gradient_zero_visitor.hpp File Reference	2867
40.481. Detailed Description	2868

40.482	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/load_output_parameter_↔ visitor.hpp File Reference	2868
40.482.1	Detailed Description	2869
40.483	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/loss_visitor.hpp File Reference .	2869
40.483.1	Detailed Description	2870
40.484	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/output_height_visitor.hpp File Reference	2871
40.484.1	Detailed Description	2872
40.485	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/output_parameter_visitor.hpp File Reference	2872
40.485.1	Detailed Description	2873
40.486	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/output_width_visitor.hpp File Reference	2873
40.486.1	Detailed Description	2874
40.487	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/parameters_set_visitor.hpp File Reference	2875
40.487.1	Detailed Description	2875
40.488	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/parameters_visitor.hpp File Ref- erence	2875
40.488.1	Detailed Description	2876
40.489	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/reset_cell_visitor.hpp File Refer- ence	2876
40.489.1	Detailed Description	2877
40.490	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/reset_visitor.hpp File Reference	2877
40.490.1	Detailed Description	2878
40.491	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/reward_set_visitor.hpp File Ref- erence	2879
40.491.1	Detailed Description	2879
40.492	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/run_set_visitor.hpp File Reference	2880
40.492.1	Detailed Description	2880
40.493	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/save_output_parameter_↔ visitor.hpp File Reference	2881

40.493. Detailed Description	2881
40.494. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/set_input_height_visitor.hpp</code> File Reference	2882
40.494. Detailed Description	2882
40.495. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/set_input_width_visitor.hpp</code> File Reference	2883
40.495. Detailed Description	2883
40.496. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/weight_set_visitor.hpp</code> File Reference	2884
40.496. Detailed Description	2885
40.497. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/weight_size_visitor.hpp</code> File Reference	2885
40.497. Detailed Description	2886
40.498. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/approx_kfn/drusilla_select.hpp</code> File Reference	2887
40.498. Detailed Description	2887
40.499. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/approx_kfn/qdafn.hpp</code> File Reference	2888
40.499. Detailed Description	2888
40.500. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/bias_svd/bias_svd.hpp</code> File Reference	2889
40.500. Detailed Description	2890
40.501. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/bias_svd/bias_svd_function.hpp</code> File Reference	2890
40.501. Detailed Description	2892
40.502. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/block_krylov_svd/randomized_block_krylov_svd.hpp</code> File Reference	2892
40.502. Detailed Description	2893
40.503. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/cf.hpp</code> File Reference	2893
40.503. Detailed Description	2894
40.504. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/cf_model.hpp</code> File Reference	2894
40.504. Detailed Description	2895
40.505. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/batch_svd_method.hpp</code> File Reference	2895

40.505. Detailed Description	2896
40.506 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/bias_svd_↔ method.hpp File Reference	2896
40.506. Detailed Description	2897
40.507 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/nmf_method.hpp File Reference	2898
40.507. Detailed Description	2899
40.508 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/randomized_↔ svd_method.hpp File Reference	2899
40.509 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/decomposition_policies/randomized_↔ svd_method.hpp File Reference	2900
40.510 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/regularized_↔ svd_method.hpp File Reference	2900
40.510. Detailed Description	2901
40.511 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/svd_complete_↔ _method.hpp File Reference	2902
40.512 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/svd_incomplete_↔ _method.hpp File Reference	2903
40.513 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/svdplusplus_↔ method.hpp File Reference	2904
40.513. Detailed Description	2905
40.514 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/interpolation_policies/average_interpolation.hpp File Reference	2905
40.514. Detailed Description	2906
40.515 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/interpolation_policies/regression_↔ interpolation.hpp File Reference	2906
40.515. Detailed Description	2907
40.516 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/interpolation_policies/similarity_↔ interpolation.hpp File Reference	2907
40.516. Detailed Description	2907
40.517 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/neighbor_search_policies/cosine_↔ search.hpp File Reference	2908
40.517. Detailed Description	2908

40.518	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/neighbor_search_policies/lmetric_↵ search.hpp File Reference	2909
40.518	Detailed Description	2910
40.519	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/neighbor_search_policies/pearson_↵ search.hpp File Reference	2910
40.519	Detailed Description	2911
40.520	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/combined_normalization.hpp File Reference	2911
40.520	Detailed Description	2911
40.521	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/item_mean_normalization.hpp File Reference	2912
40.521	Detailed Description	2912
40.522	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/no_normalization.hpp File Reference	2913
40.522	Detailed Description	2914
40.523	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/overall_mean_normalization.hpp File Reference	2914
40.523	Detailed Description	2914
40.524	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/user_mean_normalization.hpp File Reference	2915
40.524	Detailed Description	2915
40.525	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/z_score_normalization.hpp File Reference	2915
40.525	Detailed Description	2916
40.526	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/svd_wrapper.hpp File Reference	2916
40.526	Detailed Description	2917
40.527	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/dbscan/dbscan.hpp File Reference	2917
40.527	Detailed Description	2918
40.528	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/dbscan/ordered_point_selection.hpp File Reference	2918
40.528	Detailed Description	2919
40.529	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/dbscan/random_point_selection.hpp File Reference	2919

40.529.	Detailed Description	2920
40.530.	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_stump/decision_stump.hpp File Reference	2920
40.530.	Detailed Description	2921
40.531.	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/all_categorical_split.hpp File Reference	2921
40.531.	Detailed Description	2922
40.532.	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/all_dimension_select.hpp File Reference	2923
40.532.	Detailed Description	2924
40.533.	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/best_binary_numeric_split.hpp File Reference	2924
40.533.	Detailed Description	2925
40.534.	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/decision_tree.hpp File Reference	2925
40.534.	Detailed Description	2926
40.535.	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/gini_gain.hpp File Reference	2927
40.535.	Detailed Description	2928
40.536.	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/information_gain.hpp File Reference	2928
40.537.	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/information_gain.hpp File Reference	2929
40.538.	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/multiple_random_dimension_select.hpp File Reference	2930
40.538.	Detailed Description	2930
40.539.	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/random_dimension_select.hpp File Reference	2931
40.539.	Detailed Description	2931
40.540.	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/det/dt_utils.hpp File Reference	2931
40.540.	Detailed Description	2932
40.541.	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/det/dtree.hpp File Reference	2932
40.541.	Detailed Description	2933

40.542	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/dtb.hpp File Reference	2934
40.542.	Detailed Description	2934
40.543	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/dtb_rules.hpp File Reference	2935
40.544	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/dtb_stat.hpp File Reference	2936
40.545	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/edge_pair.hpp File Reference	2937
40.545.	Detailed Description	2937
40.546	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/union_find.hpp File Reference	2938
40.546.	Detailed Description	2939
40.547	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/fastmks/fastmks.hpp File Reference	2939
40.547.	Detailed Description	2940
40.548	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/fastmks/fastmks_model.hpp File Reference	2940
40.548.	Detailed Description	2941
40.549	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/fastmks/fastmks_rules.hpp File Reference	2941
40.549.	Detailed Description	2942
40.550	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/fastmks/fastmks_stat.hpp File Reference	2942
40.550.	Detailed Description	2943
40.551	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/diagonal_constraint.hpp File Reference	2943
40.551.	Detailed Description	2944
40.552	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/diagonal_gmm.hpp File Reference	2945
40.552.	Detailed Description	2946
40.553	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/eigenvalue_ratio_constraint.hpp File Reference	2946
40.553.	Detailed Description	2946
40.554	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/em_fit.hpp File Reference	2947
40.554.	Detailed Description	2948
40.555	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/gmm.hpp File Reference	2948
40.555.	Detailed Description	2949
40.556	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/no_constraint.hpp File Reference	2949

40.556. Detailed Description	2950
40.557home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/positive_definite_constraint.hpp File Reference	2950
40.557. Detailed Description	2951
40.558home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hmm/hmm.hpp File Reference	2951
40.558. Detailed Description	2952
40.559home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hmm/hmm_model.hpp File Reference	2952
40.559. Detailed Description	2953
40.559.2 Function Documentation	2953
40.559.2.1BOOST_CLASS_VERSION()	2953
40.560home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hmm/hmm_regression.hpp File Reference	2954
40.560. Detailed Description	2954
40.561home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hmm/hmm_util.hpp File Reference	2954
40.561. Detailed Description	2955
40.562home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/binary_numeric_split.hpp File Reference	2956
40.562. Detailed Description	2957
40.563home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/binary_numeric_split_↔ info.hpp File Reference	2957
40.563. Detailed Description	2958
40.564home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/categorical_split_info.hpp File Reference	2958
40.564. Detailed Description	2960
40.565home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/gini_impurity.hpp File Reference	2960
40.565. Detailed Description	2961
40.566home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/hoeffding_categorical_↔ split.hpp File Reference	2961
40.566. Detailed Description	2962
40.567home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/hoeffding_numeric_↔ split.hpp File Reference	2963

40.567. Detailed Description	2964
40.568 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/hoeffding_tree.hpp File Reference	2964
40.569 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/hoeffding_tree_model.hpp File Reference	2965
40.569. Detailed Description	2965
40.570 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/numeric_split_info.hpp File Reference	2966
40.570. Detailed Description	2967
40.571 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/kde.hpp File Reference	2967
40.571. Detailed Description	2968
40.572 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/kde_model.hpp File Reference	2968
40.572. Detailed Description	2969
40.573 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/kde_rules.hpp File Reference	2970
40.573. Detailed Description	2970
40.574 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/kde_stat.hpp File Reference	2971
40.574. Detailed Description	2972
40.575 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kernel_pca/kernel_pca.hpp File Reference	2972
40.575. Detailed Description	2972
40.576 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kernel_pca/kernel_rules/naive_method.hpp File Reference	2973
40.576. Detailed Description	2973
40.577 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kernel_pca/kernel_rules/nystroem_method.hpp File Reference	2974
40.577. Detailed Description	2974
40.578 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nystroem_method/nystroem_method.hpp File Reference	2974
40.579 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/allow_empty_clusters.hpp File Reference	2975
40.579. Detailed Description	2976
40.580 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/dual_tree_kmeans.hpp File Reference	2976

40.581	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/dual_tree_kmeans_rules.hpp	File Reference	2977
40.582	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/dual_tree_kmeans_statistic.hpp	File Reference	2978
40.583	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/elkan_kmeans.hpp	File Reference	2979
40.583.1	Detailed Description		2979
40.584	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/hamerly_kmeans.hpp	File Reference	2980
40.584.1	Detailed Description		2980
40.585	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/kill_empty_clusters.hpp	File Reference	2981
40.586	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/kmeans.hpp	File Reference	2981
40.586.1	Detailed Description		2982
40.587	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/max_variance_new_cluster.hpp	File Reference	2983
40.587.1	Detailed Description		2984
40.588	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/naive_kmeans.hpp	File Reference	2984
40.588.1	Detailed Description		2985
40.589	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/pelleg_moore_kmeans.hpp	File Reference	2985
40.589.1	Detailed Description		2986
40.590	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/pelleg_moore_kmeans_rules.hpp	File Reference	2986
40.591	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/pelleg_moore_kmeans_statistic.hpp	File Reference	2987
40.591.1	Detailed Description		2988
40.592	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/random_partition.hpp	File Reference	2988
40.592.1	Detailed Description		2988
40.593	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/refined_start.hpp	File Reference	2989
40.593.1	Detailed Description		2989
40.594	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/sample_initialization.hpp	File Reference	2990
40.594.1	Detailed Description		2991

40.595	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lars/lars.hpp File Reference	2991
40.595.1	Detailed Description	2992
40.596	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear_regression/linear_regression.hpp File Reference	2992
40.596.1	Detailed Description	2993
40.597	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear_svm/linear_svm.hpp File Reference	2994
40.597.1	Detailed Description	2994
40.598	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear_svm/linear_svm_function.hpp File Reference	2995
40.598.1	Detailed Description	2995
40.599	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lmnn/constraints.hpp File Reference	2996
40.599.1	Detailed Description	2997
40.600	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lmnn/lmnn.hpp File Reference	2997
40.600.1	Detailed Description	2997
40.601	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lmnn/lmnn_function.hpp File Reference	2998
40.601.1	Detailed Description	2999
40.602	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/local_coordinate_coding/lcc.hpp File Reference	2999
40.602.1	Detailed Description	2999
40.603	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/logistic_regression/logistic_regression.hpp File Reference	3000
40.603.1	Detailed Description	3001
40.604	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/logistic_regression/logistic_regression_function.hpp File Reference	3001
40.604.1	Detailed Description	3002
40.605	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lsh/lsh_search.hpp File Reference	3002
40.605.1	Detailed Description	3003
40.605.2	Function Documentation	3003
40.605.2.1	BOOST_TEMPLATE_CLASS_VERSION()	3003
40.606	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/matrix_completion/matrix_completion.hpp File Reference	3004

40.606. Detailed Description	3004
40.607home/barak/src/git/debian-src/mlpack/src/mlpack/methods/mean_shift/mean_shift.hpp File Reference	3005
40.607. Detailed Description	3005
40.608home/barak/src/git/debian-src/mlpack/src/mlpack/methods/naive_bayes/naive_bayes_classifier.hpp File Reference	3005
40.608. Detailed Description	3006
40.609home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nca/nca.hpp File Reference	3006
40.609. Detailed Description	3007
40.610home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nca/nca_softmax_error_function.hpp File Reference	3007
40.610. Detailed Description	3008
40.611home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/neighbor_search.hpp File Reference	3008
40.611. Detailed Description	3009
40.612home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/neighbor_search_rules.hpp File Reference	3010
40.612. Detailed Description	3011
40.613home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/neighbor_search_stat.hpp File Reference	3011
40.614home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ns_model.hpp File Reference	3012
40.614. Detailed Description	3013
40.614.2Function Documentation	3013
40.614.2.1BOOST_TEMPLATE_CLASS_VERSION()	3013
40.615home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/sort_policies/furthest_neighbor_sort.hpp File Reference	3014
40.615. Detailed Description	3015
40.616home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/sort_policies/nearest_neighbor_sort.hpp File Reference	3015
40.616. Detailed Description	3016
40.617home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/unmap.hpp File Reference	3016
40.617. Detailed Description	3017

40.618	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nystroem_method/kmeans_selection.hpp File Reference	3017
40.618	Detailed Description	3018
40.619	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nystroem_method/ordered_selection.hpp File Reference	3019
40.619	Detailed Description	3019
40.620	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nystroem_method/random_selection.hpp File Reference	3019
40.620	Detailed Description	3020
40.621	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/decomposition_policies/exact_svd_↔ method.hpp File Reference	3020
40.621	Detailed Description	3021
40.622	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/decomposition_policies/quic_svd_↔ method.hpp File Reference	3021
40.622	Detailed Description	3022
40.623	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/decomposition_policies/randomized_↔ block_krylov_method.hpp File Reference	3022
40.623	Detailed Description	3023
40.624	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/pca.hpp File Reference	3023
40.624	Detailed Description	3023
40.625	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/initialization_methods/zero_↔ init.hpp File Reference	3024
40.625	Detailed Description	3025
40.626	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/learning_policies/simple_↔ weight_update.hpp File Reference	3025
40.626	Detailed Description	3026
40.627	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/perceptron.hpp File Reference .	3026
40.627	Detailed Description	3027
40.628	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/quic_svd/quic_svd.hpp File Reference . . .	3027
40.628	Detailed Description	3028
40.629	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/radical/radical.hpp File Reference	3028
40.629	Detailed Description	3029

40.630	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/random_forest/bootstrap.hpp File Reference	3029
40.630.1	Detailed Description	3030
40.631	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/random_forest/random_forest.hpp File Reference	3030
40.631.1	Detailed Description	3031
40.632	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/randomized_svd/randomized_svd.hpp File Reference	3031
40.632.1	Detailed Description	3032
40.633	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/range_search.hpp File Reference	3032
40.633.1	Detailed Description	3033
40.634	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/range_search_rules.hpp File Reference	3034
40.634.1	Detailed Description	3034
40.635	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/range_search_stat.hpp File Reference	3035
40.635.1	Detailed Description	3036
40.636	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/rs_model.hpp File Reference	3036
40.636.1	Detailed Description	3037
40.637	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ra_model.hpp File Reference	3038
40.637.1	Detailed Description	3039
40.638	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ra_query_stat.hpp File Reference	3039
40.638.1	Detailed Description	3040
40.639	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ra_search.hpp File Reference	3041
40.639.1	Detailed Description	3042
40.640	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ra_search_rules.hpp File Reference	3042
40.640.1	Detailed Description	3043
40.641	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ra_typedef.hpp File Reference	3043
40.641.1	Detailed Description	3044
40.642	home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ra_util.hpp File Reference	3045

40.642. Detailed Description	3046
40.643 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/regularized_svd/regularized_svd.hpp File Reference	3046
40.643. Detailed Description	3047
40.644 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/regularized_svd/regularized_svd_function.hpp File Reference	3047
40.644. Detailed Description	3049
40.645 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/async_learning.hpp File Reference	3049
40.645. Detailed Description	3050
40.646 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/acrobot.hpp File Reference	3050
40.646. Detailed Description	3051
40.647 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/cart_←_pole.hpp File Reference	3051
40.647. Detailed Description	3052
40.648 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/continuous_←_mountain_car.hpp File Reference	3052
40.648. Detailed Description	3053
40.649 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/mountain_←_car.hpp File Reference	3053
40.649. Detailed Description	3054
40.650 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/pendulum.hpp File Reference	3054
40.650. Detailed Description	3055
40.651 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/reward_←_clipping.hpp File Reference	3055
40.651. Detailed Description	3055
40.652 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/policy/aggregated_←_policy.hpp File Reference	3056
40.652. Detailed Description	3056
40.653 home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/policy/greedy_←_policy.hpp File Reference	3056

40.653. Detailed Description	3057
40.654. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/q_learning.hpp</code> File Reference	3057
40.654. Detailed Description	3058
40.655. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/replay/random_replay.hpp</code> File Reference	3058
40.655. Detailed Description	3059
40.656. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/training_config.hpp</code> File Reference	3059
40.656. Detailed Description	3060
40.657. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/n_step_q_learning_worker.hpp</code> File Reference	3060
40.657. Detailed Description	3061
40.658. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/one_step_q_learning_worker.hpp</code> File Reference	3061
40.658. Detailed Description	3062
40.659. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/one_step_sarsa_worker.hpp</code> File Reference	3063
40.659. Detailed Description	3064
40.660. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/softmax_regression/softmax_regression.hpp</code> File Reference	3064
40.660. Detailed Description	3064
40.661. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/softmax_regression/softmax_regression_function.hpp</code> File Reference	3065
40.661. Detailed Description	3065
40.662. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_autoencoder/maximal_inputs.hpp</code> File Reference	3066
40.662. Detailed Description	3066
40.663. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_autoencoder/sparse_autoencoder.hpp</code> File Reference	3066
40.663. Detailed Description	3067
40.664. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_autoencoder/sparse_autoencoder_function.hpp</code> File Reference	3067

40.664. Detailed Description	3068
40.665. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_coding/data_dependent_random_initializer.hpp</code> File Reference	3068
40.665. Detailed Description	3069
40.666. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_coding/nothing_initializer.hpp</code> File Reference	3069
40.666. Detailed Description	3070
40.667. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_coding/random_initializer.hpp</code> File Reference	3070
40.667. Detailed Description	3071
40.668. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_coding/sparse_coding.hpp</code> File Reference	3071
40.668. Detailed Description	3072
40.669. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/svdplusplus/svdplusplus.hpp</code> File Reference	3072
40.669. Detailed Description	3073
40.670. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/methods/svdplusplus/svdplusplus_function.hpp</code> File Reference	3073
40.670. Detailed Description	3075
40.671. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/prereqs.hpp</code> File Reference	3075
40.671. Detailed Description	3076
40.671.2. Macro Definition Documentation	3076
40.671.2.1. <code>_USE_MATH_DEFINES</code>	3076
40.671.2.2. <code>BOOST_MPL_CFG_NO_PREPROCESSED_HEADERS</code>	3076
40.671.2.3. <code>BOOST_MPL_LIMIT_LIST_SIZE</code>	3076
40.671.2.4. <code>BOOST_PFTO</code>	3076
40.671.2.5. <code>force_inline</code>	3077
40.671.2.6. <code>M_PI</code>	3077
40.671.2.7. <code>omp_size_t</code>	3077
40.672. <code>home/barak/src/git/debian-src/mlpack/src/mlpack/tests/ann_test_tools.hpp</code> File Reference	3077
40.672. Detailed Description	3078

40.672.2	Function Documentation	3078
40.672.2.1	CheckGradient()	3078
40.672.2.2	JacobianPerformanceTest()	3078
40.672.2.3	JacobianTest()	3078
40.672.2.4	ResetFunction() [1/2]	3079
40.672.2.5	ResetFunction() [2/2]	3079
40.673	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/custom_layer.hpp File Reference	3079
40.673.1	Detailed Description	3080
40.674	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/braziltourism_labels.txt File Reference . .	3080
40.675	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/corrupt-observations-1.txt File Reference .	3080
40.676	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/corrupt-observations-2.txt File Reference .	3080
40.677	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/data_3d_ind.txt File Reference	3080
40.678	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/data_3d_mixed.txt File Reference	3080
40.679	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/iris_labels.txt File Reference	3080
40.680	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/labels.txt File Reference	3080
40.681	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/observations.txt File Reference	3080
40.682	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/r10.txt File Reference	3081
40.683	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/test_labels_nonlinsep.txt File Reference .	3081
40.684	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/test_nonlinsep.txt File Reference	3081
40.685	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/train_labels_nonlinsep.txt File Reference .	3081
40.686	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/train_nonlinsep.txt File Reference	3081
40.687	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/vc2_labels.txt File Reference	3081
40.688	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/vc2_test_labels.txt File Reference	3081
40.689	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/main_tests/hmm_test_utils.hpp File Reference	3081
40.689.1	Detailed Description	3082
40.690	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/main_tests/range_search_utils.hpp File Reference	3082
40.690.1	Detailed Description	3082
40.690.2	Function Documentation	3083

40.690.2.1	CheckMatrices() [1/2]	3083
40.690.2.2	CheckMatrices() [2/2]	3083
40.690.2.3	ModelToString()	3083
40.690.2.4	ReadData()	3084
40.691	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/main_tests/test_helper.hpp File Reference	3084
40.691.1	Detailed Description	3085
40.692	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/mock_categorical_data.hpp File Reference	3085
40.692.1	Detailed Description	3085
40.692.2	Function Documentation	3085
40.692.2.1	MockCategoricalData()	3086
40.693	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/test_function_tools.hpp File Reference	3086
40.693.1	Detailed Description	3086
40.693.2	Function Documentation	3086
40.693.2.1	LogisticRegressionTestData()	3087
40.694	home/barak/src/git/debian-src/mlpack/src/mlpack/tests/test_tools.hpp File Reference	3087
40.694.1	Detailed Description	3088
40.694.2	Macro Definition Documentation	3088
40.694.2.1	REQUIRE_RELATIVE_ERR	3088
40.694.3	Function Documentation	3088
40.694.3.1	CheckMatrices() [1/3]	3089
40.694.3.2	CheckMatrices() [2/3]	3089
40.694.3.3	CheckMatrices() [3/3]	3089
40.694.3.4	CheckMatricesNotEqual() [1/3]	3089
40.694.3.5	CheckMatricesNotEqual() [2/3]	3090
40.694.3.6	CheckMatricesNotEqual() [3/3]	3090
40.694.3.7	FilterFileName()	3090

Chapter 1

mlpack Documentation

1.1 Introduction

mlpack is an intuitive, fast, and flexible C++ machine learning library with bindings to other languages. It is meant to be a machine learning analog to LAPACK, and aims to implement a wide array of machine learning methods and function as a "swiss army knife" for machine learning researchers. The mlpack website can be found at <https://mlpack.org>.

1.2 How To Use This Documentation

This documentation is API documentation similar to Javadoc. It isn't necessarily a tutorial, but it does provide detailed documentation on every namespace, method, and class.

Each mlpack namespace generally refers to one machine learning method, so browsing the list of namespaces provides some insight as to the breadth of the methods contained in the library.

To generate this documentation in your own local copy of mlpack, you can use the 'doc' CMake target, which is available if CMake has found Doxygen, from the build directory:

```
$ make doc
```

1.3 Tutorials

A few short tutorials on how to use mlpack are given below.

- **Building mlpack From Source** (p. 21)
- **Building mlpack From Source on Windows** (p. 27)
- **Matrices in mlpack** (p. 57)
- **Writing an mlpack binding** (p. 53)

- **mlpack Timers** (p. 69)
- **Simple Sample mlpack Programs** (p. 63)
- **Sample C++ ML App for Windows** (p. 65)
- **Cross-Validation** (p. 35)
- **Hyper-Parameter Tuning** (p. 49)
- **mlpack version information** (p. 71)

1.4 Final Remarks

For the list of contributors to mlpack, see <https://www.mlpack.org/community.html>. This library would not be possible without everyone's hard work and contributions!

Chapter 2

mlpack automatic bindings to other languages

2.1 Overview

mlpack has a system to automatically generate bindings to other languages, such as Python and command-line programs, and it is extensible to other languages with some amount of ease. The maintenance burden of this system is low, and it is designed in such a way that the bindings produced are always up to date across languages and up to date with the mlpack library itself.

This document describes the full functioning of the system, and is a good place to start for someone who wishes to understand the system so that they can contribute a new binding language, or someone who wants to understand so they can adapt the system for use in their own project, or someone who is simply curious enough to see how the sausage is made.

The document is split into several sections:

- **Introduction** (p. 4)
- **Writing code that can be turned into a binding** (p. 5)
- **How to write mlpack bindings** (p. 7)
 - **Documenting a program with PROGRAM_INFO()** (p. 8)
 - **Defining parameters for a program** (p. 10)
 - **Using CLI in an mlpackMain() function** (p. 13)
 - **More documentation on using CLI** (p. 14)
- **Structure of CLI module and associated macros** (p. 14)
- **Command-line program bindings** (p. 16)
 - **mlpackMain() definition** (p. 16)
 - **Matrix and model parameter handling** (p. 17)
 - **Parsing the command line** (p. 17)
- **Python bindings** (p. 18)
 - **Passing matrices to/from Python** (p. 18)
 - **Passing model parameter to/from Python** (p. 18)
 - **CMake generation of setup.py** (p. 19)
 - **Building the .pyx files** (p. 19)
 - **Testing the Python bindings** (p. 19)
- **Adding new binding types** (p. 20)

2.2 Introduction

C++ is not the most popular language on the planet, and it (unfortunately) can scare many away with its ultra-verbose error messages, confusing template rules, and complex metaprogramming techniques. Most practitioners of machine learning tend to avoid writing native C++ and instead prefer other languages—probably most notably Python.

In the case of Python, many projects will use tools like SWIG (<http://www.swig.org/>) to automatically generate bindings, or they might hand-write Cython. The same types of strategies may be used for other languages; hand-written MEX files may be used for MATLAB, hand-written RCpp bindings might be used for R bindings, and so forth.

However, these approaches have a fundamental flaw: the hand-written bindings must be maintained, and risk going out of date as the rest of the library changes or new functionality is added. This incurs a maintenance burden: each major change to the library means that someone must update the bindings and test that they are still working. mlpack is not prepared to handle this maintenance workload; therefore an alternate solution is needed.

At the time of the design of this system, mlpack shipped headers for a C++ library as well as many (~40) hand-written command-line programs that used the `mlpack::CLI` (p. 1117) object to manage command-line arguments. These programs all had similar structure, and could be logically split into three sections:

- parse the input options supplied by the user
- run the machine learning algorithm
- prepare the output to return to the user

The user might interface with this command-line program like the following:

```
$ mlpack_knn -r reference.csv -q query.csv -k 3 -d d.csv -n n.csv
```

That is, they would pass a number of input options—some were numeric values (like `-k 3`); some were filenames (like `-r reference.csv`); and a few other types also. Therefore, the first stage of the program—parsing input options—would be handled by reading the command line and loading any input matrices. Preparing the output, which usually consists of data matrices (i.e. `-d d.csv`) involves saving the matrix returned by the algorithm to the user's desired file.

Ideally, any binding to any language would have this same structure, and the actual "run the machine learning algorithm" code could be identical. For MATLAB, for instance, we would not need to read the file `reference.csv` but instead the user would simply pass their data matrix as an argument. So each input and output parameter would need to be handled differently, but the algorithm could be run identically across all bindings.

Therefore, design of an automatically-generated binding system would simply involve generating the boilerplate code necessary to parse input options for a given language, and to return output options to a user.

2.3 Writing code that can be turned into a binding

This section details what a binding file might actually look like. It is good to have this API in mind when reading the following sections.

Each mlpack binding is typically contained in the `src/mlpack/methods/` folder corresponding to a given machine learning algorithm, with the suffix `_main.cpp`; so an example is `src/mlpack/methods/pca/pca_main.cpp`.

These files have roughly two parts:

- definition of the input and output parameters with `PARAM` macros
- implementation of `mlpackMain()`, which is the actual machine learning code

Here is a simple example file:

```
// This is a stripped version of mean_shift_main.cpp.
#include <mlpack/prereqs.hpp>
#include <mlpack/core/util/cli.hpp>
#include <mlpack/core/util/mlpack_main.hpp>

#include <mlpack/core/kernels/gaussian_kernel.hpp>
#include "mean_shift.hpp"

using namespace mlpack;
using namespace mlpack::meanshift;
using namespace mlpack::kernel;
using namespace std;

// Define the help text for the program. The PRINT_PARAM_STRING() and
// PRINT_DATASET() macros are used to print the name of the parameter as seen in
// the binding type that is being used, and the PRINT_CALL() macro generates a
// sample invocation of the program in the language of the binding type that is
// being used. Note that the macros must have + on either side of them. We
// provide some extra references with the "SEE_ALSO()" macro, which is used to
// generate documentation for the website.
PROGRAM_INFO("Mean Shift Clustering",
  // Short description.
  "A fast implementation of mean-shift clustering using dual-tree range "
  "search. Given a dataset, this uses the mean shift algorithm to produce "
  "and return a clustering of the data.",
  // Long description.
  "This program performs mean shift clustering on the given dataset, storing "
  "the learned cluster assignments either as a column of labels in the input "
  "dataset or separately."
  "\n\n"
  "The input dataset should be specified with the " +
  PRINT_PARAM_STRING("input") + " parameter, and the radius used for search "
  "can be specified with the " + PRINT_PARAM_STRING("radius") + " "
  "parameter. The maximum number of iterations before algorithm termination "
  "is controlled with the " + PRINT_PARAM_STRING("max_iterations") + " "
  "parameter."
  "\n\n"
  "The output labels may be saved with the " + PRINT_PARAM_STRING("output") +
  " output parameter and the centroids of each cluster may be saved with the "
  " " + PRINT_PARAM_STRING("centroid") + " output parameter."
  "\n\n"
  "For example, to run mean shift clustering on the dataset " +
  PRINT_DATASET("data") + " and store the centroids to " +
  PRINT_DATASET("centroids") + ", the following command may be used: "
  "\n\n" +
  PRINT_CALL("mean_shift", "input", "data", "centroid", "centroids"),
  SEE_ALSO("@kmeans", "#kmeans"),
  SEE_ALSO("@dbscan", "#dbscan"),
  SEE_ALSO("Mean shift on Wikipedia",
    "https://en.wikipedia.org/wiki/Mean_shift"),
  SEE_ALSO("Mean Shift, Mode Seeking, and Clustering (pdf)",
    "http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.510.1222")
);
```

```

    "&rep=repl&type=pdf"),
    SEE_ALSO("mlpack::mean_shift::MeanShift C++ class documentation",
    "@doxygen/classmlpack_1_1meanshift_1_1MeanShift.html"));

// Define parameters for the executable.

// Required option: the user must give us a matrix.
PARAM_MATRIX_IN_REQ("input", "Input dataset to perform clustering on.", "i");

// Output options: the user can save the output matrix of labels and/or the
// centroids.
PARAM_UCOL_OUT("output", "Matrix to write output labels to.", "o");
PARAM_MATRIX_OUT("centroid", "If specified, the centroids of each cluster will "
    "be written to the given matrix.", "C");

// Mean shift configuration options.
PARAM_INT_IN("max_iterations", "Maximum number of iterations before mean shift "
    "terminates.", "m", 1000);
PARAM_DOUBLE_IN("radius", "If the distance between two centroids is less than "
    "the given radius, one will be removed. A radius of 0 or less means an "
    "estimate will be calculated and used for the radius.", "r", 0);

void mlpackMain()
{
    // Process the parameters that the user passed.
    const double radius = CLI::GetParam<double>("radius");
    const int maxIterations = CLI::GetParam<int>("max_iterations");

    if (maxIterations < 0)
    {
        Log::Fatal << "Invalid value for maximum iterations (" << maxIterations <<
            ")! Must be greater than or equal to 0." << endl;
    }

    // Warn, if the user did not specify that they wanted any output.
    if (!CLI::HasParam("output") && !CLI::HasParam("centroid"))
    {
        Log::Warn << "--output_file, --in_place, and --centroid_file are not set; "
            << "no results will be saved." << endl;
    }

    arma::mat dataset = std::move(CLI::GetParam<arma::mat>("input"));
    arma::mat centroids;
    arma::Col<size_t> assignments;

    // Prepare and run the actual algorithm.
    MeanShift<> meanShift(radius, maxIterations);

    Timer::Start("clustering");
    Log::Info << "Performing mean shift clustering..." << endl;
    meanShift.Cluster(dataset, assignments, centroids);
    Timer::Stop("clustering");

    Log::Info << "Found " << centroids.n_cols << " centroids." << endl;
    if (radius <= 0.0)
        Log::Info << "Estimated radius was " << meanShift.Radius() << ".\n";

    // Should we give the user the output matrix?
    if (CLI::HasParam("output"))
        CLI::GetParam<arma::Col<size_t>>("output") = std::move(assignments);

    // Should we give the user the centroid matrix?
    if (CLI::HasParam("centroid"))
        CLI::GetParam<arma::mat>("centroid") = std::move(centroids);
}

```

We can see that we have defined the basic program information in the **PROGRAM_INFO()** (p.2733) macro. This is, for instance, what is displayed to describe the binding if the user passed the `--help` option for a command-line program.

Then, we define five parameters, three input and two output, that define the data and options that the mean shift clustering will function on. These parameters are defined with the `PARAM` macros, of which there are many. The names of these macros specify the type, whether the parameter is required, and whether the parameter is input or output. Some examples:

- `PARAM_STRING_IN()` (p. 2721) – a string-type input parameter
- `PARAM_MATRIX_OUT()` (p. 2716) – a matrix-type output parameter
- `PARAM_DOUBLE_IN_REQ()` (p. 2708) – a required double-type input parameter
- `PARAM_UMATRIX_IN()` (p. 2727) – an unsigned matrix-type input parameter
- `PARAM_MODEL_IN()` (p. 2717) – a serializable model-type input parameter

Note that each of these macros may have slightly different syntax. See the links above for further documentation.

In order to write a new binding, then, you simply must write a `PROGRAM_INFO()` (p. 2733) definition of the program with some documentation, define the input and output parameters as `PARAM` macros, and then write an `mlpackMain()` function that actually performs the functionality of the binding. Inside of `mlpackMain()`:

- All input parameters are accessible through `CLI::GetParam<type>("name")`.
- All output parameters should be set by the end of the function with the `CLI::GetParam<type>("name")` method.

Then, assuming that your program is saved in the file `program_name_main.cpp`, generating bindings for other languages is a simple addition to the `CMakeLists.txt` file:

```
add_cli_executable(program_name)
add_python_binding(program_name)
add_markdown_docs(program_name "cli;python" "category")
```

In this example, `add_markdown_docs()` will generate documentation that is typically used to build the website. The "category" parameter should be one of the categories in `src/mlpack/bindings/markdown/MarkdownCategories.cmake`.

2.4 How to write mlpack bindings

This section describes the general structure of the `CLI` code and how one might write a new binding for mlpack. After reading this section it should be relatively clear how one could use the `CLI` functionality along with CMake to add a binding for a new mlpack machine learning method. If it is not clear, then the examples in the following sections should clarify.

2.4.1 Documenting a program with `PROGRAM_INFO()`

Any mlpack program should be documented with the `PROGRAM_INFO()` (p. 2733) macro, which is available from the `<mlpack/core/util/mlpack_main.hpp` (p. 2699) header. The macro is of the form

```
PROGRAM_INFO("program name", "short documentation", "long documentation",
    SEE_ALSO("link", "description"), ...)
```

The short documentation should be two sentences indicating what the program implements and does, and a quick overview of how it can be used and what it should be used for. When writing new short documentation, it is a good idea to take a look at the existing documentation to get an idea of the general format.

For the "see also" section, you can specify as many `SEE_ALSO()` (p. 2734) calls as you see fit. These are links used at the "see also" section of the website documentation for each binding, and it's very important that relevant links are provided (also to other bindings). See the `SEE_ALSO()` (p. 2734) documentation for more details.

Although it is possible to provide very short documentation, it is certainly better to provide a long description including

- what the program does
- a basic overview of what input and output parameters the program has
- at least one example invocation

Examples are very important, and are probably what most users are going to immediately search for, instead of taking a long time to read and carefully consider all of the written documentation.

However, it is difficult to write language-agnostic documentation. For instance, in a command-line program, an output parameter `--output_file` would be specified on the command line as an input parameter, but in Python, the output parameter `'output'` would actually simply be returned from the call to the Python function. Therefore, we must be careful how our documentation refers to input and output parameters. The following general guidelines can help:

- Always refer to output parameters as "output parameters", which is a fairly close term that can be interpreted to mean both "return values" for languages like Python and MATLAB and also "arguments given on the command line" for command line programs.
- Use the provided `PRINT_PARAM_STRING()` macro to print the names of parameters. For instance, `PRINT_PARAM_STRING("shuffle")` will print `'--shuffle'` for a command line program and `'shuffle'` for a Python binding. The `PRINT_PARAM_STRING()` macro also takes into account the type of the parameter.
- Use the provided `PRINT_DATASET()` and `PRINT_MODEL()` macro to introduce example datasets or models, which can be useful when introducing an example usage of the program. So you could write `"to @c run @c with @c a @c dataset @c " + PRINT_DATASET("data") + "..."`.
- Use the provided `PRINT_CALL()` macro to print example invocations of the program. The first argument is the name of the program, and then the following arguments should be the name of a parameter followed by the value of that parameter.
- Never mention files in the documentation—files are only relevant to command-line programs. Similarly, avoid mentioning anything language-specific.

- Remember that some languages give output through return values and some give output using other input parameters. So the right verbiage to use is, e.g., 'the results may be saved using the `PRINT_PARAM_STRING("output")` parameter', and **not** 'the results are returned through the `PRINT_PARAM_STRING("output")` parameter'.

Each of these macros (`PRINT_PARAM_STRING()`, `PRINT_DATASET()`, `PRINT_MODEL()`, and `PRINT_CALL()`) provides different output depending on the language. Below are some example of documentation strings and their outputs for different languages. Note that the output might not be *exactly* as written or formatted here, but the general gist should be the same.

Input C++ (snippet):

```
"The parameter " + PRINT_PARAM_STRING("shuffle") + ", if set, will shuffle "
"the data before learning."
```

Command-line program output (snippet):

```
The parameter '--shuffle', if set, will shuffle the data before learning.
```

Python binding output (snippet):

```
The parameter 'shuffle', if set, will shuffle the data before learning.
```

Input C++ (snippet):

```
"The output matrix can be saved with the " + PRINT_PARAM_STRING("output") +
" output parameter."
```

Command-line program output (snippet):

```
The output matrix can be saved with the '--output_file' output parameter.
```

Python binding output (snippet):

```
The output matrix can be saved with the 'output' output parameter.
```

Input C++ (snippet):

```
"For example, to train a model on the dataset " + PRINT_DATASET("x") + " and "
"save the output model to " + PRINT_MODEL("model") + ", the following command"
" can be used:"
"\n\n" +
PRINT_CALL("program", "input", "x", "output_model", "model")
```

Command-line program output (snippet):

```
For example, to train a model on the dataset 'x.csv' and save the output model
to 'model.bin', the following command can be used:
```

```
$ program --input_file x.csv --output_model_file model.bin
```

Python binding output (snippet):

```
For example, to train a model on the dataset 'x' and save the output model to
'model', the following command can be used:
```

```
>>> output = program(input=x)
>>> model = output['output_model']
```

Input C++ (full program, 'random_numbers_main.cpp'):

```
PROGRAM_INFO("Random Numbers", "This program generates random numbers with a "
"variety of nonsensical techniques and example parameters. The input "
"dataset, which will be ignored, can be specified with the " +
PRINT_PARAM_STRING("input") + " parameter. If you would like to subtract "
" values from each number, specify the " +
PRINT_PARAM_STRING("subtract") + " parameter. The number of random "
"numbers to generate is specified with the " +
PRINT_PARAM_STRING("num_values") + " parameter.")
```

```

"\n\n"
"The output random numbers can be saved with the " +
PRINT_PARAM_STRING("output") + " output parameter. In addition, a "
"randomly generated linear regression model can be saved with the " +
PRINT_PARAM_STRING("output_model") + " output parameter."
"\n\n"
"For example, to generate 100 random numbers with 3 subtracted from them "
"and save the output to " + PRINT_DATASET("rand") + " and the random "
"model to " + PRINT_MODEL("rand_lr") + ", use the following "
"command:"
"\n\n" +
PRINT_CALL("random_numbers", "num_values", 100, "subtract", 3, "output",
"rand", "output_model", "rand_lr"));

```

Command line output:

Random Numbers

This program generates random numbers with a variety of nonsensical techniques and example parameters. The input dataset, which will be ignored, can be specified with the '--input_file' parameter. If you would like to subtract values from each number, specify the '--subtract' parameter. The number of random numbers to generate is specified with the '--num_values' parameter.

The output random numbers can be saved with the '--output_file' output parameter. In addition, a randomly generated linear regression model can be saved with the '--output_model_file' output parameter.

For example, to generate 100 random numbers with 3 subtracted from them and save the output to 'rand.csv' and the random model to 'rand_lr.bin', use the following command:

```
$ random_numbers --num_values 100 --subtract 3 --output_file rand.csv
--output_model_file rand_lr.bin
```

Python binding output:

Random Numbers

This program generates random numbers with a variety of nonsensical techniques and example parameters. The input dataset, which will be ignored, can be specified with the 'input' parameter. If you would like to subtract values from each number, specify the 'subtract' parameter. The number of random numbers to generate is specified with the 'num_values' parameter.

The output random numbers can be saved with the 'output' output parameter. In addition, a randomly generated linear regression model can be saved with the 'output_model' output parameter.

For example, to generate 100 random numbers with 3 subtracted from them and save the output to 'rand' and the random model to 'rand_lr', use the following command:

```
>>> output = random_numbers(num_values=100, subtract=3)
>>> rand = output['output']
>>> rand_lr = output['output_model']
```

2.4.2 Defining parameters for a program

There exist several macros that can be used after a **PROGRAM_INFO()** (p.2733) definition to define the parameters that can be specified for a given mlpack program. These macros all have the same general definition: the name of the macro specifies the type of the parameter, whether or not the parameter is required, and whether the parameter is an input or output parameter. Then as arguments to the macro, the name, description, and sometimes the single-character alias and the default value of the parameter.

To give a flavor of how these definitions look, the definition

```
PARAM_STRING_IN("algorithm", "The algorithm to use: 'svd' or 'blah'.", "a");
```

will define a string input parameter `algorithm` (referenced as `--algorithm` from the command-line or `'algorithm'` from Python) with the description `The algorithm to use: 'svd' or 'blah'`. The single-character alias `-a` can be used from a command-line program (but means nothing in Python).

There are numerous different macros that can be used:

- **PARAM_FLAG()** (p. 2710) - boolean flag parameter
- **PARAM_INT_IN()** (p. 2711) - integer input parameter
- **PARAM_INT_OUT()** (p. 2712) - integer output parameter
- **PARAM_DOUBLE_IN()** (p. 2708) - double input parameter
- **PARAM_DOUBLE_OUT()** (p. 2709) - double output parameter
- **PARAM_STRING_IN()** (p. 2721) - string input parameter
- **PARAM_STRING_OUT()** (p. 2723) - string output parameter
- **PARAM_MATRIX_IN()** (p. 2714) - double-valued matrix (`arma::mat`) input parameter
- **PARAM_MATRIX_OUT()** (p. 2716) - double-valued matrix (`arma::mat`) output parameter
- **PARAM_UMATRIX_IN()** (p. 2727) - `size_t`-valued matrix (`arma::Mat<size_t>`) input parameter
- **PARAM_UMATRIX_OUT()** (p. 2729) - `size_t`-valued matrix (`arma::Mat<size_t>`) output parameter
- **PARAM_TMATRIX_IN()** (p. 2723) - transposed double-valued matrix (`arma::mat`) input parameter
- **PARAM_TMATRIX_OUT()** (p. 2725) - transposed double-valued matrix (`arma::mat`) output parameter
- **PARAM_MATRIX_AND_INFO_IN()** (p. 2714) - matrix with categoricals input parameter (`std::tuple<data::DatasetInfo, arma::mat>`)
- **PARAM_COL_IN()** (p. 2705) - double-valued column vector (`arma::vec`) input parameter
- **PARAM_COL_OUT()** (p. 2707) - double-valued column vector (`arma::vec`) output parameter
- **PARAM_UCOL_IN()** (p. 2726) - `size_t`-valued column vector (`arma::Col<size_t>`) input parameter
- **PARAM_UCOL_OUT()** (p. 2726) - `size_t`-valued column vector (`arma::Col<size_t>`) output parameter
- **PARAM_ROW_IN()** (p. 2720) - double-valued row vector (`arma::rowvec`) input parameter
- **PARAM_ROW_OUT()** (p. 2721) - double-valued row vector (`arma::rowvec`) output parameter
- **PARAM_VECTOR_IN()** (p. 2731) - `std::vector` input parameter
- **PARAM_VECTOR_OUT()** (p. 2733) - `std::vector` output parameter
- **PARAM_MODEL_IN()** (p. 2717) - serializable model input parameter
- **PARAM_MODEL_OUT()** (p. 2718) - serializable model output parameter

And for input parameters, the parameter may also be required:

- **PARAM_INT_IN_REQ()** (p. 2712)
- **PARAM_DOUBLE_IN_REQ()** (p. 2708)

- `PARAM_STRING_IN_REQ()` (p. 2722)
- `PARAM_MATRIX_IN_REQ()` (p. 2715)
- `PARAM_UMATRIX_IN_REQ()` (p. 2728)
- `PARAM_TMATRIX_IN_REQ()` (p. 2724)
- `PARAM_VECTOR_IN_REQ()` (p. 2732)
- `PARAM_MODEL_IN_REQ()` (p. 2718)

Click the links for each macro to read further documentation. Note also that each possible combination of `IN`, `OUT`, and `REQ` is not available—output options cannot be required, and some combinations simply have not been added because they have not been needed.

The `PARAM_MODEL_IN()` (p. 2717) and `PARAM_MODEL_OUT()` (p. 2718) macros are used to serialize mlpack models. These could be used, for instance, to allow the user to save a trained model (like a linear regression model) or load an input model. The first parameter to the `PARAM_MODEL_IN()` (p. 2717) or `PARAM_MODEL_OUT()` (p. 2718) macro should be the C++ type of the model to be serialized; this type **must** have a function template `<typename Archive> void Serialize(Archive&, const unsigned int)` (i.e. the type must be serializable via mlpack's `boost::serialization` (p. 251) shim). For example, to allow a user to specify an input model of type `LinearRegression`, the follow definition could be used:

```
PARAM_MODEL_IN(LinearRegression, "input_model", "The input model to be used.",
               "i");
```

Then, the user will be able to specify their model from the command-line as `--input_model_file` and from Python using the `input_model` option to the generated binding.

From the command line, `matrix-type` and `model-type` options (both input and output) are loaded from or saved to the specified file. This means that `_file` is appended to the name of the parameter; so if the parameter name is `data` and it is of a matrix or model type, then the name that the user will specify on the command line will be `--data_file`. This displayed parameter name change **only** occurs with matrix and model type parameters for command-line programs.

The `PARAM_MATRIX_AND_INFO()` macro defines a categorical matrix parameter (more specifically, a matrix type that can support categorical columns). From the C++ program side, this means that the parameter type is `std::tuple<data::DatasetInfo, arma::mat>`. From the user side, for a command-line program, this means that the user will pass the filename of a dataset that can have categorical features, such as an ARFF dataset. For a Python program, the user may pass a Pandas matrix with categorical columns. When the program is run, the input that the user gives will be processed and the `data::DatasetInfo` object will be filled with the dimension types and the `arma::mat` object will be filled with the data itself.

To give some examples, the parameter definitions from the example "random_numbers" program in the previous section are shown below.

```
PARAM_MATRIX_IN("input", "The input matrix that will be ignored.", "i");
PARAM_DOUBLE_IN("subtract", "The value to subtract from each parameter.", "s",
                0.0); // Default value of 0.0.
PARAM_INT_IN("num_samples", "The number of samples to generate.", "n", 100);

PARAM_MATRIX_OUT("output", "The output matrix of random samples.", "o");
PARAM_MODEL_OUT(LinearRegression, "output_model", "The randomly generated "
               "linear regression output model.", "M");
```

Note that even the parameter documentation strings must be a little be agnostic to the binding type, because the command-line interface is so different than the Python interface to the user.

2.4.3 Using CLI in an mlpackMain() function

mlpack's CLI module provides a unified abstract interface for getting input from and providing output to users without needing to consider the language (command-line, Python, MATLAB, etc.) that the user is running the program from. This means that after the `PROGRAM_INFO()` (p. 2733) macro and the `PARAM_*`() macros have been defined, a language-agnostic `mlpackMain()` function can be written. This function then can perform the actual computation that the entire program is meant to.

Inside of an `mlpackMain()` function, the `mlpack::CLI` (p. 1117) module can be used to access input parameters and set output parameters. There are two main functions for this, plus a utility printing function:

- `CLI::GetParam<T>()` (p. 290) - get a reference to a parameter
- `CLI::HasParam()` - returns true if the user specified the parameter
- `CLI::GetPrintableParam<T>()` (p. 292) - returns a string representing the value of the parameter

So, to print "hello" if the user specified the `print_hello` parameter, the following code could be used:

```
using namespace mlpack;

if (CLI::HasParam("print_hello"))
  std::cout << "Hello!" << std::endl;
else
  std::cout << "No greetings for you!" << std::endl;
```

To access a string that a user passed in to the `string` parameter, the following code could be used:

```
using namespace mlpack;

const std::string& str = CLI::GetParam<std::string>("string");
```

Matrix types are accessed in the same way:

```
using namespace mlpack;

arma::mat& matrix = CLI::GetParam<arma::mat>("matrix");
```

Similarly, model types can be accessed. If a `LinearRegression` model was specified by the user as the parameter `model`, the following code can access the model:

```
using namespace mlpack;

LinearRegression& lr = CLI::GetParam<LinearRegression>("model");
```

Matrices with categoricals are a little trickier to access since the C++ parameter type is `std::tuple<data::DatasetInfo, arma::mat>`. The example below creates references to both the `DatasetInfo` and `matrix` objects, assuming the user has passed a matrix with categoricals as the `matrix` parameter.

```
using namespace mlpack;

typename std::tuple<data::DatasetInfo, arma::mat> TupleType;
data::DatasetInfo& di = std::get<0>(CLI::GetParam<TupleType>("matrix"));
arma::mat& matrix = std::get<1>(CLI::GetParam<TupleType>("matrix"));
```

These two functions can be used to write an entire program. The third function, `GetPrintableParam()` (p. 292), can be used to help provide useful output in a program. Typically, this function should be used if you want to provide some kind of error message about a matrix or model parameter, but want to avoid printing the matrix itself. For instance, printing a matrix parameter with `GetPrintableParam()` (p. 292) will print the filename for a command-line binding or the size of a matrix for a Python binding. `GetPrintableParam()` (p. 292) for a model parameter will print the filename for the model for a command-line binding or a simple string representing the type of the model for a Python binding.

Putting all of these ideas together, here is the `mlpackMain()` function that could be created for the "random_↵ numbers" program from earlier sections.

```
#include <mlpack/core/util/mlpack_main.hpp>

// The PROGRAM_INFO() and PARAM_*() definitions should go here:
// ...

using namespace mlpack;

void mlpackMain()
{
    // If the user passed an input matrix, tell them that we'll be ignoring it.
    if (CLI::HasParam("input"))
    {
        // Print the filename the user passed, if a command-line binding, or the
        // size of the matrix passed, if a Python binding.
        Log::Warn << "The input matrix "
            << CLI::GetPrintableParam<arma::mat>("input") << " is ignored!"
            << std::endl;
    }

    // Get the number of samples and also the value we should subtract.
    const size_t numSamples = (size_t) CLI::GetParam<int>("num_samples");
    const double subtractValue = CLI::GetParam<double>("subtract");

    // Create the random matrix (1-dimensional).
    arma::mat output(1, numSamples, arma::fill::randu);
    output -= subtractValue;

    // Save the output matrix if the user wants.
    if (CLI::HasParam("output"))
        CLI::GetParam<arma::mat>("output") = std::move(output); // Avoid copy.

    // Did the user request a random linear regression model?
    if (CLI::HasParam("output_model"))
    {
        LinearRegression lr;
        lr.Parameters().randu(10); // 10-dimensional (arbitrary).
        lr.Lambda() = 0.0;
        lr.Intercept() = false; // No intercept term.

        CLI::GetParam<LinearRegression>("output_model") = std::move(lr);
    }
}
```

2.4.4 More documentation on using CLI

More documentation for the CLI module can either be found on the `mlpack::CLI` (p. 1117) documentation page, or by reading the existing mlpack bindings. These can be found in the `src/mlpack/methods/` folders, by finding the `_main.cpp` files. For instance, `src/mlpack/methods/neighbor_search/knn_main.cpp` is the k-nearest-neighbor search program definition.

2.5 Structure of CLI module and associated macros

This section describes the internal functionality of the CLI module and the associated macros. If you are only interested in writing mlpack programs, this section is probably not worth reading.

There are four main components involved with mlpack bindings:

- the CLI module, a singleton class that stores parameter information
- the `mlpackMain()` function that defines the functionality of the binding
- the **PROGRAM_INFO()** (p. 2733) macro that defines the binding name and documentation
- the `PARAM_*`() macros that define parameters for the binding

The `mlpack::CLI` (p. 1117) module is a singleton class that stores, at runtime, the binding name, the documentation, and the parameter information and values. In order to do this, each parameter and the program documentation must make themselves known to the CLI singleton. This is accomplished by having the **PROGRAM_INFO()** (p. 2733) and `PARAM_*`() macros declare global variables that, in their constructors, register themselves with the CLI singleton.

The **PROGRAM_INFO()** (p. 2733) macro declares an object of type `mlpack::util::ProgramDoc` (p. 2368). The `ProgramDoc` class constructor calls `CLI::RegisterProgramDoc()` in order to register the given program name and documentation.

The `PARAM_*`() macros declare an object that will, in its constructor, call `CLI::Add()` to register that parameter with the CLI singleton. The specific type of that object will depend on the binding type being used.

The `CLI::Add()` function takes an `mlpack::util::ParamData` (p. 2356) object as its input. This `ParamData` object has a number of fields that must be set to properly describe the parameter. Each of the fields is documented and probably self-explanatory, but three fields deserve further explanation:

- the `std::string tname` member is used to encode the true type of the parameter—which is not known by the CLI singleton at runtime. This should be set to **TYPENAME (T)** (p. 2738) where `T` is the type of the parameter.
- the `boost::any value` member is used to hold the actual value of the parameter. Typically this will simply be the parameter held by a `boost::any` object, but for some types it may be more complex. For instance, for a command-line matrix option, the `value` parameter will actually hold a tuple containing both the filename and the matrix itself.
- the `std::string cppType` should be a string containing the type as seen in C++ code. Typically this can be encoded by stringifying a `PARAM_*`() macro argument.

Thus, the global object defined by the `PARAM_*`() macro must turn its arguments into a fully specified `ParamData` object and then call `CLI::Add()` with it.

With different binding types, different behavior is often required for the **GetParam<T>()** (p. 290), `HasParam()`, and **GetPrintableParam<T>()** (p. 292) functions. In order to handle this, the CLI singleton also holds a function pointer map, so that a given type of option can call specific functionality for a certain task. This function map is accessible as `CLI::functionMap` and is not meant to be used by users, but instead by people writing binding types.

Each function in the map must have signature

```
void MapFunction(const util::ParamData& d,
                 const void* input,
                 void* output);
```

The use of void pointers allows any type to be specified as input or output to the function without changing the signature for the map. The CLI function map is of type

```
std::map<std::string, std::map<std::string,
    void (*) (const util::ParamData&, const void*, void*)>>
```

and the first map key is the typename (`tname`) of the parameter, and the second map key is the string name of the function. For instance, calling

```
const util::ParamData& d = CLI::Parameters()["param"];
CLI::GetSingleton().functionMap[d.tname]["GetParam"](d, input, output);
```

will call the **GetParam()** (p. 290) function for the type of the "param" parameter. Examples are probably easiest to understand how this functionality works; see the **CLI::GetParam<T>()** (p. 290) source to see how this might be used.

The CLI singleton expects the following functions to be defined in the function map for each type:

- **GetParam** – return a pointer to the parameter in output.
- **GetPrintableParam** – return a pointer to a string description of the parameter in output.

If these functions are properly defined, then the CLI module will work correctly. Other functions may also be defined; these may be used by other parts of the binding infrastructure for different languages.

2.6 Command-line program bindings

This section describes the internal functionality of the command-line program binding generator. If you are only interested in writing mlpack programs, this section probably is not worth reading. This section is worth reading only if you want to know the specifics of how the `mlpackMain()` function and macros get turned into a fully working command-line program.

The code for the command-line bindings is found in `src/mlpack/bindings/cli`.

2.6.1 mlpackMain() definition

Any command-line program must be compiled with the `BINDING_TYPE` macro set to the value `BINDING_TYPE_CLI`. This is handled by the CMake macro **add_cli_executable()** (p. 2438).

When `BINDING_TYPE` is set to `BINDING_TYPE_CLI`, the following is set in `src/mlpack/core/util/mlpack_main.hpp` (p. 2699), which must be included by every mlpack binding:

- The options defined by `PARAM_*` macros are of type **mlpack::bindings::cli::CLIOption** (p. 975).
- The parameter and value printing macros for **PROGRAM_INFO()** (p. 2733) are set: The `PRINT_PARAMETER_STRING()` macro is defined as **mlpack::bindings::cli::ParamString()** (p. 302). The `PRINT_DATASET()` macro is defined as **mlpack::bindings::cli::PrintDataset()** (p. 302). The `PRINT_MODEL()` macro is defined as **mlpack::bindings::cli::PrintModel()** (p. 303). The `PRINT_CALL()` macro is defined as **mlpack::bindings::cli::ProgramCall()** (p. 306).
- The function `int main()` is defined as:


```
int main(int argc, char** argv)
{
    // Parse the command-line options; put them into CLI.
    mlpack::bindings::cli::ParseCommandLine(argc, argv);

    mlpackMain();

    // Print output options, print verbose information, save model parameters,
    // clean up, and so forth.
    mlpack::bindings::cli::EndProgram();
}
```

Thus any mlpack command-line binding first processes the command-line arguments with `mlpack::bindings::cli::ParseCommandLine()` (p. 302), then runs the binding with `mlpackMain()`, then cleans up with `mlpack::bindings::cli::EndProgram()` (p. 288).

The `ParseCommandLine()` (p. 302) function reads the input parameters and sets the values in CLI. For matrix-type and model-type parameters, this reads the filenames from the command-line, but does not load the matrix or model. Instead the matrix or model is loaded the first time it is accessed with `GetParam<T>()` (p. 290).

The `--help` parameter is handled by the `mlpack::bindings::cli::PrintHelp()` (p. 303) function.

At the end of program execution, the `mlpack::bindings::cli::EndProgram()` (p. 288) function is called. This writes any output matrix or model parameters to disk, and prints the program parameters and timers if `--verbose` was given.

2.6.2 Matrix and model parameter handling

For command line bindings, the matrix, model, and matrix with categorical type parameters all require special handling, since it is not possible to pass a matrix of any reasonable size or a model on the command line directly. Therefore for a matrix or model parameter, the user specifies the file containing that matrix or model parameter. If the parameter is an input parameter, then the file is loaded when `GetParam<T>()` (p. 290) is called. If the parameter is an output parameter, then the matrix or model is saved to the file when `EndProgram()` (p. 288) is called.

The actual implementation of this is that the `boost::any` value member of the `ParamData` struct does not hold the model or the matrix, but instead a `std::tuple` containing both the matrix or the model, and the filename associated with that matrix or model.

This means that functions like `GetParam<T>()` (p. 290) and `GetPrintableParam<T>()` (p. 292) (and all of the other associated functions in the CLI function map) must have special handling for matrix or model types. See those implementations for more details—the special handling is enforced via SFINAE.

2.6.3 Parsing the command line

The `ParseCommandLine()` (p. 302) function uses `boost::program_options` to read the values from the command line into the `ParamData` structs held by the CLI singleton.

In order to set up `boost::program_options`—and to keep its headers from needing to be included by the rest of the library—the code loops over each parameter known by the CLI singleton and calls the "AddToPO" function from the function map. This in turn calls the necessary functions to register a given parameter with `boost::program_options`, and once all parameters have been registered, the facilities provided by `boost::program_options` are used to parse the command line input properly.

2.7 Python bindings

This section describes the internal functionality of the mlpack Python binding generator. If you are only interested in writing new bindings or building the bindings, this section is probably not worth reading. But if you are interested in the internal working of the Python binding generator, then this section is for you.

The Python bindings are significantly more complex than the command line bindings because we cannot just compile directly to a finished product. Instead we need a multi-stage compilation:

- We must generate a `setup.py` file that can be used to compile the bindings.
- We must generate the `.pyx` (Cython) bindings for each program.
- Then we must build each `.pyx` into a `.so` that is loadable from Python.
- We must also test the Python bindings.

This is done with a combination of C++ code to generate the `.pyx` bindings, CMake to run the actual compilation and generate the `setup.py` file, some utility Python functions, and tests written in both Python and C++. This code is primarily contained in `src/mlpack/bindings/python/`.

2.7.1 Passing matrices to/from Python

The standard Python matrix library is numpy, so mlpack bindings should accept numpy matrices as input. Fortunately, numpy Cython bindings already exist, which make it easy to convert from a numpy object to an Armadillo object without copying any data. This code can be found in `src/mlpack/bindings/python/mlpack/arma_numpy.pyx`, and is used by the Python `GetParam<T>()` (p.290) functionality.

mlpack also supports categorical matrices; in Python, the typical way of representing matrices with categorical features is with Pandas. Therefore, mlpack also accepts Pandas matrices, and if any of the Pandas matrix dimensions are categorical, these are properly encoded. The function `to_matrix_with_info()` from `mlpack/bindings/python/mlpack/matrix_utils.py` is used to perform this conversion.

2.7.2 Passing model parameter to/from Python

We use (or abuse) Cython functionality in order to give the user a model object that they can use in their Python code. However, we do not want to (or have the infrastructure to) write bindings for every method that a serializable model class might support; therefore, we only desire to return a memory pointer to the model to the user.

In this way, a user that receives a model from an output parameter can then reuse the model as an input parameter to another binding (or the same binding).

To return a function pointer we have to define a Cython class in the following way (this example is taken from the perceptron binding):

```
cdef extern from "</home/ryan/src/mlpack-rc/src/mlpack/methods/perceptron/perceptron_main.cpp>" nogil:
    cdef int mlpackMain() nogil except +RuntimeError

    cdef cppclass PerceptronModel:
        PerceptronModel() nogil

    cdef class PerceptronModelType:
        cdef PerceptronModel* modelptr

        def __cinit__(self):
            self.modelptr = new PerceptronModel()

        def __dealloc__(self):
            del self.modelptr
```

This class definition is automatically generated when the `.pyx` file is automatically generated.

2.7.3 CMake generation of setup.py

A boilerplate setup.py file can be found in `src/mlpack/bindings/python/setup.py.in`. This will be configured by CMake to produce the final `setup.py` file, but in order to do this, a list of the .pyx files to be compiled must be gathered.

Therefore, the `add_python_binding()` macro is defined in `src/mlpack/bindings/python/CMakeLists.txt` (p. 2386). This adds the given binding to the `MLPACK_PYXS` variable, which is then inserted into `setup.py` as part of the `configure_file()` step in `src/mlpack/CMakeLists.txt` (p. 2389).

2.7.4 Generation of .pyx files

A binding named `program` is built into a program called `generate_pyx_program` (this a CMake target, so you can build these individually if you like). The file `src/mlpack/bindings/python/generate_pyx.cpp.in` is configured by CMake to set the name of the program and the `*_main.cpp` file to include correctly, then the `mlpack::bindings::python::PrintPYX()` (p. 354) function is called by the program. The `PrintPYX()` (p. 354) function uses the parameters that have been set in the CLI singleton by the `PROGRAM_INFO()` (p. 2733) and `PARAM_*`() macros in order to actually print a fully-working .pyx file that can be compiled. The file has several sections:

- Python imports (numpy/pandas/cython/etc.)
- Cython imports of C++ utility functions and Armadillo functionality
- Cython imports of any necessary serializable model types
- Definitions of classes for serializable model types
- The binding function definition
- Documentation: input and output parameters
- The call to `mlpackMain()`
- Handling of output functionality
- Return of output parameters

Any output parameters for Python bindings are returned in a dict containing named elements.

2.7.5 Building the .pyx files

After building the `generate_pyx_program` target, the `build_pyx_program` target is built as a dependency of the `python` target. This simply takes the generated .pyx file and uses Python `setuptools` to compile this to a Python binding.

2.7.6 Testing the Python bindings

We cannot do our tests only from the Boost Unit Test Framework in C++ because we need to see that we are able to load parameters properly from Python and return output correctly.

The tests are in `src/mlpack/bindings/python/tests/` and test both the actual bindings and also the auxiliary Python code included in `src/mlpack/bindings/python/mlpack/`.

2.8 Adding new binding types

Adding a new binding type to mlpack is fairly straightforward once the general structure of the CLI singleton and the function map that CLI uses is understood. For each different language that bindings are desired for, the route to a solution will be particularly different—so it is hard to provide any general guidance for how to make new bindings that will be applicable to each language.

In general, the first thing to handle will be how matrices are passed back and forth between the target language. Typically this might mean getting the memory address of an input matrix and wrapping an `arma::mat` object around that memory address. This can be handled in the `GetParam()` (p. 290) function that is part of the CLI singleton function map; see `get_param.hpp` for both the CLI and Python bindings for an example (in `src/mlpack/bindings/cli/` and `src/mlpack/bindings/python/`).

Serialization of models is also a tricky consideration; in some languages you will be able to pass a pointer to the model itself. This is generally best—users should not expect to be able to manipulate the model in the target language, but they should expect that they can pass a model back and forth without paying a runtime penalty. So, for example, serializing a model using a `boost::text_oarchive` and then returning the string that represents the model is not acceptable, because that string can be extremely large and the time it takes to decode the model can be very large.

The strategy of generating a binding definition for the target language, like what is done with Python, can be a useful strategy that should be considered. If this is the route that is desired, a large amount of CMake boilerplate may be necessary. The Python CMake configuration can be referred to as an example, but probably a large amount of adaptation to other languages will be necessary.

Lastly, when adding a new language, be sure to make sure it works with the Markdown documentation generator. In order to make this happen, you will need to modify all of the `add_markdown_docs()` calls in the different CMake `Lists.txt` files to contain the name of the language you have written a binding for. You will also need to modify every function in `src/mlpack/bindings/markdown/print_doc_functions_impl.hpp` to correctly call out to the corresponding function for the language that you have written bindings for.

Chapter 3

Building mlpack From Source

3.1 Introduction

This document discusses how to build mlpack from source. However, mlpack is in the repositories of many Linux distributions and so it may be easier to use the package manager for your system. For example, on Ubuntu, you can install mlpack with the following command:

```
$ sudo apt-get install libmlpack-dev
```

Note

Older Ubuntu versions may not have the most recent version of mlpack available—for instance, at the time of this writing, Ubuntu 16.04 only has mlpack 2.0.1 available. Options include upgrading Ubuntu to a newer release, finding a PPA or other non-official sources, or installing with a manual build (below).

If mlpack is not available in your system's package manager, then you can follow this document for how to compile and install mlpack from source.

mlpack uses CMake as a build system and allows several flexible build configuration options. One can consult any of numerous CMake tutorials for further documentation, but this tutorial should be enough to get mlpack built and installed on most Linux and UNIX-like systems (including OS X). If you want to build mlpack on Windows, see **Building mlpack From Source on Windows** (p. 27) (alternatively, you can read Keon's excellent tutorial which is based on older versions).

You can download the latest mlpack release from here: `mlpack-3.1.1`

3.2 Simple Linux build instructions

Assuming all dependencies are installed in the system, you can run the commands below directly to build and install mlpack.

```
$ wget https://www.mlpack.org/files/mlpack-3.1.1.tar.gz
$ tar -xvzpf mlpack-3.1.1.tar.gz
$ mkdir mlpack-3.1.1/build && cd mlpack-3.1.1/build
$ cmake ../
$ make -j4 # The -j is the number of cores you want to use for a build.
$ sudo make install
```

If the `cmake ..` command fails, you are probably missing a dependency, so check the output and install any necessary libraries. (See **Dependencies of mlpack** (p. 22).)

On many Linux systems, mlpack will install by default to `/usr/local/lib` and you may need to set the `LD_LIBRARY_PATH` environment variable:

```
export LD_LIBRARY_PATH=/usr/local/lib
```

The instructions above are the simplest way to get, build, and install mlpack. The sections below discuss each of those steps in further detail and show how to configure mlpack.

3.3 Creating Build Directory

First we should unpack the mlpack source and create a build directory.

```
$ tar -xvzpf mlpack-3.1.1.tar.gz
$ cd mlpack-3.1.1
$ mkdir build
```

The directory can have any name, not just 'build', but 'build' is sufficient.

3.4 Dependencies of mlpack

mlpack depends on the following libraries, which need to be installed on the system and have headers present:

- Armadillo $\geq 6.500.0$ (with LAPACK support)
- Boost (math_c99, program_options, serialization, unit_test_framework, heap, spirit) ≥ 1.49

For Python bindings, the following packages are required:

- setuptools
- cython ≥ 0.24

- numpy
- pandas \geq 0.15.0
- pytest-runner

In Ubuntu and Debian, you can get all of these dependencies through apt:

```
# apt-get install libboost-math-dev libboost-program-options-dev
libboost-test-dev libboost-serialization-dev libarmadillo-dev binutils-dev
python-pandas python-numpy cython python-setuptools
```

On Fedora, Red Hat, or CentOS, these same dependencies can be obtained via dnf:

```
# dnf install boost-devel boost-test boost-program-options boost-math
armadillo-devel binutils-devel python2-Cython python2-setuptools
python2-numpy python2-pandas
```

(It's also possible to use python3 packages from the package manager—mlpack will work with either.)

3.5 Configuring CMake

Running CMake is the equivalent to running `./configure` with autotools. If you run CMake with no options, it will configure the project to build without debugging or profiling information (for speed).

```
$ cd build
$ cmake ../
```

You can manually specify options to compile with debugging information and profiling information (useful if you are developing mlpack):

```
$ cd build
$ cmake -D DEBUG=ON -D PROFILE=ON ../
```

The full list of options mlpack allows:

- `DEBUG=(ON/OFF)`: compile with debugging symbols (default OFF)
- `PROFILE=(ON/OFF)`: compile with profiling symbols (default OFF)
- `ARMA_EXTRA_DEBUG=(ON/OFF)`: compile with extra Armadillo debugging symbols (default OFF)
- `BUILD_TESTS=(ON/OFF)`: compile the `mlpack_test` program (default ON)
- `BUILD_CLI_EXECUTABLES=(ON/OFF)`: compile the mlpack command-line executables (i.e. `mlpack_knn`, `mlpack_kfn`, `mlpack_logistic_regression`, etc.) (default ON)
- `BUILD_PYTHON_BINDINGS=(ON/OFF)`: compile the bindings for Python, if the necessary Python libraries are available (default ON except on Windows)

- `MATLAB_BINDINGS=(ON/OFF)`: Compile MATLAB bindings if MATLAB is found (default OFF)
- `BUILD_SHARED_LIBS=(ON/OFF)`: compile shared libraries as opposed to static libraries (default ON)
- `TEST_VERBOSE=(ON/OFF)`: run test cases in `mlpack_test` with verbose output (default OFF)
- `DOWNLOAD_ENSMALLEN=(ON/OFF)`: If `ensmallen` is not found, download it (default ON)
- `BUILD_WITH_COVERAGE=(ON/OFF)`: Build with support for code coverage tools (gcc only) (default OFF)
- `BUILD_MARKDOWN_BINDINGS=(ON/OFF)`: Build Markdown bindings for website documentation (default OFF)
- `MATHJAX=(ON/OFF)`: use MathJax for generated Doxygen documentation (default OFF)
- `FORCE_CXX11=(ON/OFF)`: assume that the compiler supports C++11 instead of checking; be sure to specify any necessary flag to enable C++11 as part of `CXXFLAGS` (default OFF)
- `USE_OPENMP=(ON/OFF)`: if ON, then use OpenMP if the compiler supports it; if OFF, OpenMP support is manually disabled (default ON)

Each option can be specified to CMake with the '-D' flag. Other tools can also be used to configure CMake, but those are not documented here.

In addition, the following directories may be specified, to find include files and libraries. These also use the '-D' flag.

- `ARMADILLO_INCLUDE_DIR=(/path/to/armadillo/include/)`: path to Armadillo headers
- `ARMADILLO_LIBRARY=(/path/to/armadillo/libarmadillo.so)`: location of Armadillo library
- `BOOST_ROOT=(/path/to/boost/)`: path to root of boost installation
- `ENSMALLEN_INCLUDE_DIR=(/path/to/ensmallen/include)`: path to include directory for `ensmallen`
- `MATHJAX_ROOT=(/path/to/mathjax)`: path to root of MathJax installation

3.6 Building mlpack

Once CMake is configured, building the library is as simple as typing 'make'. This will build all library components as well as 'mlpack_test'.

```
$ make
Scanning dependencies of target mlpack
[ 1%] Building CXX object
src/mlpack/CMakeFiles/mlpack.dir/core/optimizers/aug_lagrangian/aug_lagrangian_test_functions.cpp.o
<...>
```

It's often useful to specify `-jN` to the `make` command, which will build on `N` processor cores. That can accelerate the build significantly.

You can specify individual components which you want to build, if you do not want to build everything in the library:

```
$ make mlpack_pca mlpack_knn mlpack_kfn
```


One particular component of interest is `mlpack_test`, which runs the mlpack test suite. You can build this component with

```
$ make mlpack_test
```

and then run all of the tests, or an individual test suite:

```
$ bin/mlpack_test
$ bin/mlpack_test -t KNNTest
```

If the build fails and you cannot figure out why, register an account on Github and submit an issue and the mlpack developers will quickly help you figure it out:

<https://mlpack.org/>

<https://github.com/mlpack/mlpack>

Alternately, mlpack help can be found in IRC at `#mlpack` on `irc.freenode.net`.

3.7 Installing mlpack

If you wish to install mlpack to the system, make sure you have root privileges (or write permissions to those two directories), and simply type

```
# make install
```

You can now run the executables by name; you can link against mlpack with `-lmlpack`, and the mlpack headers are found in `/usr/include` or `/usr/local/include` (depending on the system and CMake configuration). If Python bindings were installed, they should be available when you start Python.

3.8 Using mlpack without installing

If you would prefer to use mlpack after building but without installing it to the system, this is possible. All of the command-line programs in the `build/bin/` directory will run directly with no modification.

For running the Python bindings from the build directory, the situation is a little bit different. You will need to set the following environment variables:

```
export LD_LIBRARY_PATH=/path/to/mlpack/build/lib:${LD_LIBRARY_PATH}
export PYTHONPATH=/path/to/mlpack/build/src/mlpack/bindings/python:${PYTHONPATH}
```

(Be sure to substitute the correct path to your build directory for `/path/to/mlpack/build/`.)

Once those environment variables are set, you should be able to start a Python interpreter and `import mlpack`, then use the Python bindings.

Chapter 4

Building mlpack From Source on Windows

4.1 Introduction

This tutorial will show you how to build mlpack for Windows from source, so you can later create your own C++ applications. Before you try building mlpack, you may want to install mlpack using vcpkg for Windows. If you don't want to install using vcpkg, skip this section and continue with the build tutorial.

- Install Git (<https://git-scm.com/downloads> and execute setup)
- Install CMake (<https://cmake.org/> and execute setup)
- Install vcpkg (<https://github.com/Microsoft/vcpkg> and execute setup)
- To install the mlpack library only:

```
PS> .\vcpkg install mlpack:x64-windows
```

- To install mlpack and its console programs:

```
PS> .\vcpkg install mlpack[tools]:x64-windows
```

After installing, in Visual Studio, you can create a new project (or open an existing one). The library is immediately ready to be included (via preprocessor directives) and used in your project without additional configuration.

4.2 Build Environment

This tutorial has been designed and tested using:

- Windows 10
- Visual Studio 2017 (toolset v141)
- mlpack
- OpenBLAS.0.2.14.1
- boost_1_66_0-msvc-14.1-64
- armadillo-8.500.1
- and x64 configuration

The directories and paths used in this tutorial are just for reference purposes.

4.3 Pre-requisites

- Install CMake for Windows (win64-x64 version from <https://cmake.org/download/>) and make sure you can use it from the Command Prompt (may need to add to the PATH)
- Download the latest mlpack release from here: [mlpack website](#)

4.4 Windows build instructions

- Unzip mlpack to "C:\mlpack\mlpack"
- Open Visual Studio and select: File > New > Project from Existing Code
 - Type of project: Visual C++
 - Project location: "C:\mlpack\mlpack"
 - Project name: mlpack
 - Finish
- We will use this Visual Studio project to get the OpenBLAS dependency in the next section

4.5 Dependencies

OpenBLAS Dependency

- Open the NuGet packages manager (Tools > NuGet Package Manager > Manage NuGet Packages for Solution...)
- Click on the "Browse" tab and search for "openblas"
- Click on OpenBlas and check the mlpack project, then click Install
- Once it has finished installing, close Visual Studio

Boost Dependency

You can either get Boost via NuGet or you can download the prebuilt Windows binaries separately. This tutorial follows the second approach for simplicity.

- Download the "Prebuilt Windows binaries" of the Boost library ("boost_1_66_0-msvc-14.1-64") from [Sourceforge](#)

Note

Make sure you download the MSVC version that matches your Visual Studio

- Install or unzip to "C:\boost"

Armadillo Dependency

- Download the newest version of Armadillo from [Sourceforge](#)
- Unzip to "C:\mlpack\armadillo"
- Create a "build" directory into "C:\mlpack\armadillo\"
- Open the Command Prompt and navigate to "C:\mlpack\armadillo\build"
- Run cmake:

```
cmake -G "Visual Studio 15 2017 Win64" -DBLAS_LIBRARY:FILEPATH="
C:/mlpack/mlpack/packages/OpenBLAS.0.2.14.1/lib/native/lib/x64/libopenblas.dll.a" -DLAPACK_LIBRARY:FILEPATH="
C:/mlpack/mlpack/packages/OpenBLAS.0.2.14.1/lib/native/lib/x64/libopenblas.dll.a" ..
```

Note

If you are using different directory paths, a different configuration (e.g. Release) or a different VS version, update the cmake command accordingly.

- Once it has successfully finished, open "C:\mlpack\armadillo\build\armadillo.sln"
- Build > Build Solution
- Once it has successfully finished, close Visual Studio

4.6 Building mlpack

- Create a "build" directory into "C:\mlpack\mlpack"
- You can generate the project using either cmake via command line or GUI. If you prefer to use GUI, refer to the **appendix** (p. 30)
- To use the CMake command line prompt, open the Command Prompt and navigate to "C:\mlpack\mlpack\build"
- Run cmake:

```
cmake -G "Visual Studio 15 2017 Win64" -DBLAS_LIBRARY:FILEPATH="
C:/mlpack/mlpack/packages/OpenBLAS.0.2.14.1/lib/native/lib/x64/libopenblas.dll.a" -DLAPACK_LIBRARY:FILEPATH="
C:/mlpack/mlpack/packages/OpenBLAS.0.2.14.1/lib/native/lib/x64/libopenblas.dll.a" -DARMADILLO_INCLUDE_DIR="C:/mlpack/armadillo"
-DARMADILLO_LIBRARY:FILEPATH="C:/mlpack/armadillo/build/Debug/armadillo.lib" -DBOOST_INCLUDEDIR:PATH="C:/boost/" -
DBOOST_LIBRARYDIR:PATH="C:/boost/lib64-msvc-14.1" -DDEBUG=OFF -DPROFILE=OFF ..
```

Note

cmake will attempt to automatically download the ensmallen dependency. If for some reason cmake can't download the dependency, you will need to manually download ensmallen from <http://ensmallen.org/> and extract it to "C:\mlpack\mlpack\deps". Then, specify the path to ensmallen using the flag: `-DENSMALLEN_INCLUDE_↵ DIR=C:/mlpack/mlpack/deps/ensmallen/include`

- Once CMake configuration has successfully finished, open "C:\mlpack\mlpack\build\mlpack.sln"
- Build > Build Solution (this may be by default in Debug mode)
- Once it has successfully finished, you will find the library files you need in: "C:\mlpack\mlpack\build\Debug" (or "C:\mlpack\mlpack\build\Release" if you changed to Release mode)

You are ready to create your first application, take a look at the **Sample C++ ML App** (p. 65)

4.7 Appendix

If you prefer to use cmake GUI, follow these instructions:

- To use the CMake GUI, open "CMake".
 - For "Where is the source code:" set `C:\mlpack\mlpack\`
 - For "Where to build the binaries:" set `C:\mlpack\mlpack\build`
 - Click `Configure`
 - If there is an error and Armadillo is not found, try "Add Entry" with the following variables and reconfigure:
 - * Name: `ARMADILLO_INCLUDE_DIR`; type `PATH`; value `C:/mlpack/armadillo/include/`
 - * Name: `ARMADILLO_LIBRARY`; type `FILEPATH`; value `C:/mlpack/armadillo/build/Debug/armadillo.lib`
 - * Name: `BLAS_LIBRARY`; type `FILEPATH`; value `C:/mlpack/mlpack/packages/OpenBLAS.0.2.14.1/lib/native/lib/x64/libopenblas.dll.a`
 - * Name: `LAPACK_LIBRARY`; type `FILEPATH`; value `C:/mlpack/mlpack/packages/OpenBLAS.0.2.14.1/lib/native/lib/x64/libopenblas.dll.a`
 - If there is an error and Boost is not found, try "Add Entry" with the following variables and reconfigure:
 - * Name: `BOOST_INCLUDEDIR`; type `PATH`; value `C:/boost/`
 - * Name: `BOOST_LIBRARYDIR`; type `PATH`; value `C:/boost/lib64-msvc-14.1`
 - If Boost is still not found, try adding the following variables and reconfigure:
 - * Name: `Boost_INCLUDE_DIR`; type `PATH`; value `C:/boost/`
 - * Name: `Boost_PROGRAM_OPTIONS_LIBRARY_DEBUG`; type `FILEPATH`; value should be `C:/boost/lib64-msvc-14.1/boost_program_options-vc141-mt-gd-x64-1_66.lib`
 - * Name: `Boost_PROGRAM_OPTIONS_LIBRARY_RELEASE`; type `FILEPATH`; value should be `C:/boost/lib64-msvc-14.1/boost_program_options-vc141-mt-x64-1_66.lib`
 - * Name: `Boost_SERIALIZATION_LIBRARY_DEBUG`; type `FILEPATH`; value should be `C:/boost/lib64-msvc-14.1/boost_serialization-vc141-mt-gd-x64-1_66.lib`
 - * Name: `Boost_SERIALIZATION_LIBRARY_RELEASE`; type `FILEPATH`; value should be `C:/boost/lib64-msvc-14.1/boost_program_options-vc141-mt-x64-1_66.lib`
 - * Name: `Boost_UNIT_TEST_FRAMEWORK_LIBRARY_DEBUG`; type `FILEPATH`; value should be `C:/boost/lib64-msvc-14.1/boost_unit_test_framework-vc141-mt-gd-x64-1_66.lib`
 - * Name: `Boost_UNIT_TEST_FRAMEWORK_LIBRARY_RELEASE`; type `FILEPATH`; value should be `C:/boost/lib64-msvc-14.1/boost_unit_test_framework-vc141-mt-x64-1_66.lib`
 - Once CMake has configured successfully, hit "Generate" to create the `.sln` file.

4.8 Additional Information

If you are facing issues during the build process of mlpack, you may take a look at other third-party tutorials for Windows, but they may be out of date:

Github wiki Windows Build page
 Keon's tutorial for mlpack 2.0.3
 Kirizaki's tutorial for mlpack 2

Chapter 5

mlpack command-line quickstart guide

5.1 Introduction

This page describes how you can quickly get started using mlpack from the command-line and gives a few examples of usage, and pointers to deeper documentation.

This quickstart guide is also available for **Python** (p. 59).

5.2 Installing mlpack

Installing the mlpack is straightforward and can be done with your system's package manager.

For instance, for Ubuntu or Debian the command is simply

```
sudo apt-get install mlpack-bin
```

On Fedora or Red Hat:

```
sudo dnf install mlpack
```

If you use a different distribution, mlpack may be packaged under a different name. And if it is not packaged, you can use a Docker image from Dockerhub:

```
docker run -it mlpack/mlpack /bin/bash
```

This Docker image has mlpack already built and installed.

If you prefer to build mlpack from scratch, see **Building mlpack From Source** (p. 21).

5.3 Simple mlpack quickstart example

As a really simple example of how to use mlpack from the command-line, let's do some simple classification on a subset of the standard machine learning `covertypes` dataset. We'll first split the dataset into a training set and a testing set, then we'll train an mlpack random forest on the training data, and finally we'll print the accuracy of the random forest on the test dataset.

You can copy-paste this code directly into your shell to run it.

```
# Get the dataset and unpack it.
wget https://www.mlpack.org/datasets/covertypes-small.data.csv.gz
wget https://www.mlpack.org/datasets/covertypes-small.labels.csv.gz
gunzip covertypes-small.data.csv.gz covertypes-small.labels.csv.gz

# Split the dataset; 70% into a training set and 30% into a test set.
# Each of these options has a shorthand single-character option but here we type
# it all out for clarity.
mlpack_preprocess_split \
  --input_file covertypes-small.data.csv \
  --input_labels_file covertypes-small.labels.csv \
  --training_file covertypes-small.train.csv \
  --training_labels_file covertypes-small.train.labels.csv \
  --test_file covertypes-small.test.csv \
  --test_labels_file covertypes-small.test.labels.csv \
  --test_ratio 0.3 \
  --verbose

# Train a random forest.
mlpack_random_forest \
  --training_file covertypes-small.train.csv \
  --labels_file covertypes-small.train.labels.csv \
  --num_trees 10 \
  --minimum_leaf_size 3 \
  --print_training_accuracy \
  --output_model_file rf-model.bin \
  --verbose

# Now predict the labels of the test points and print the accuracy.
# Also, save the test set predictions to the file 'predictions.csv'.
mlpack_random_forest \
  --input_model_file rf-model.bin \
  --test_file covertypes-small.test.csv \
  --test_labels_file covertypes-small.test.labels.csv \
  --predictions_file predictions.csv \
  --verbose
```

We can see by looking at the output that we achieve reasonably good accuracy on the test dataset (80%+). The file `predictions.csv` could also be used by other tools; for instance, we can easily calculate the number of points that were predicted incorrectly:

```
$ diff -U 0 predictions.csv covertypes-small.test.labels.csv | grep '^@@"' | wc -l
```

It's easy to modify the code above to do more complex things, or to use different mlpack learners, or to interface with other machine learning toolkits.

5.4 What else does mlpack implement?

The example above has only shown a little bit of the functionality of mlpack. Lots of other commands are available with different functionality. A full list of commands and full documentation for each can be found on the following page:

- [CLI documentation](#)

For more information on what mlpack does, see <https://www.mlpack.org/>. Next, let's go through another example for providing movie recommendations with mlpack.

5.5 Using mlpack for movie recommendations

In this example, we'll train a collaborative filtering model using mlpack's `mlpack_cf` program. We'll train this on the MovieLens dataset from <https://grouplens.org/datasets/movielens/>, and then we'll use the model that we train to give recommendations.

You can copy-paste this code directly into the command line to run it.

```
wget https://www.mlpack.org/datasets/ml-20m/ratings-only.csv.gz
wget https://www.mlpack.org/datasets/ml-20m/movies.csv.gz
gunzip ratings-only.csv.gz
gunzip movies.csv.gz

# Hold out 10% of the dataset into a test set so we can evaluate performance.
mlpack_preprocess_split \
  --input_file ratings-only.csv \
  --training_file ratings-train.csv \
  --test_file ratings-test.csv \
  --test_ratio 0.1 \
  --verbose

# Train the model. Change the rank to increase/decrease the complexity of the
# model.
mlpack_cf \
  --training_file ratings-train.csv \
  --test_file ratings-test.csv \
  --rank 10 \
  --algorithm RegSVD \
  --output_model_file cf-model.bin \
  --verbose

# Now query the 5 top movies for user 1.
echo "1" > query.csv;
mlpack_cf \
  --input_model_file cf-model.bin \
  --query_file query.csv \
  --recommendations 10 \
  --output_file recommendations.csv \
  --verbose

# Get the names of the movies for user 1.
echo "Recommendations for user 1:"
for i in `seq 1 10`; do
  item=`cat recommendations.csv | awk -F',' '{ print '$i' }'`;
  head -n $((($item + 2)) movies.csv | tail -1 | \
    sed 's/^[^,]*,[^,]*,/' | \
    sed 's/\/(.*)\,.*$/1/' | sed 's/"/g';
done
```

Here is some example output, showing that user 1 seems to have good taste in movies:

```
Recommendations for user 1:
Casablanca (1942)
Pan's Labyrinth (Laberinto del fauno, El) (2006)
Godfather, The (1972)
Answer This! (2010)
Life Is Beautiful (La Vita è bella) (1997)
Adventures of Tintin, The (2011)
Dark Knight, The (2008)
Out for Justice (1991)
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)
Schindler's List (1993)
```

5.6 Next steps with mlpack

Now that you have done some simple work with mlpack, you have seen how it can easily plug into a data science production workflow for the command line. A great thing to do next would be to look at more documentation for the mlpack command-line programs:

- `mlpack command-line program documentation`

Also, mlpack is much more flexible from C++ and allows much greater functionality. So, more complicated tasks are possible if you are willing to write C++. To get started learning about mlpack in C++, the following resources might be helpful:

- `mlpack C++ tutorials`
- `mlpack build and installation guide`
- `Simple sample C++ mlpack programs`
- `mlpack Doxygen documentation homepage`

Chapter 6

Cross-Validation

6.1 Introduction

mlpack implements cross-validation support for its learning algorithms, for a variety of performance measures. Cross-validation is useful for determining an estimate of how well the learner will generalize to un-seen test data. It is a commonly used part of the data science pipeline.

In short, given some learner and some performance measure, we wish to get an average of the performance measure given different splits of the dataset into training data and validation data. The learner is trained on the training data, and the performance measure is evaluated on the validation data.

mlpack currently implements two easy-to-use forms of cross-validation:

- **simple cross-validation**, where we simply desire the performance measure on a single split of the data into a training set and validation set
- **k-fold cross-validation**, where we split the data k ways and desire the average performance measure on each of the k splits of the data

In this tutorial we will see the usage examples and details of the cross-validation module. Because the cross-validation code is generic and can be used with any learner and performance measure, any use of the cross-validation code in mlpack has to be in C++.

This tutorial is split into the following sections:

- **Simple cross-validation examples** (p. 36) Simple cross-validation examples
 - **10-fold cross-validation on softmax regression** (p. 36) 10-fold cross-validation on softmax regression
 - **10-fold cross-validation on weighted decision trees** (p. 37) 10-fold cross-validation on weighted decision trees
 - **10-fold cross-validation with categorical decision trees** (p. 37) 10-fold cross-validation with categorical decision trees
 - **Simple cross-validation for linear regression** (p. 38) Simple cross-validation for linear regression
- **Performance measures** (p. 38) Performance measures
- **The KFoldCV and SimpleCV classes** (p. 39) The **KFoldCV** (p. 1134) and **SimpleCV** (p. 1151) classes
- **Further references** (p. 41) Further reference

6.2 Simple cross-validation examples

6.2.1 10-fold cross-validation on softmax regression

Suppose we have some data to train and validate on, as defined below:

```
// 100-point 6-dimensional random dataset.
arma::mat data = arma::randu<arma::mat>(6, 100);
// Random labels in the [0, 4] interval.
arma::Row<size_t> labels =
    arma::randi<arma::Row<size_t>>(100, arma::distr_param(0, 4));
size_t numClasses = 5;
```

The code above generates an 100-point random 6-dimensional dataset with 5 classes.

To run 10-fold cross-validation for softmax regression with accuracy as a performance measure, we can write the following piece of code.

```
KFoldCV<SoftmaxRegression, Accuracy> cv(10, data, labels, numClasses);
double lambda = 0.1;
double softmaxAccuracy = cv.Evaluate(lambda);
```

Note that the `Evaluate` method of **KFoldCV** (p. 1134) takes any hyperparameters of an algorithm—that is, anything that is not `data`, `labels`, `numClasses`, `datasetInfo`, or `weights` (those last three may not be present for every algorithm type). To be more specific, in this example the `Evaluate` method relies on the following **SoftmaxRegression** (p. 1811) constructor:

```
template<typename OptimizerType = mlpack::optimization::L_BFGS>
SoftmaxRegression(const arma::mat& data,
    const arma::Row<size_t>& labels,
    const size_t numClasses,
    const double lambda = 0.0001,
    const bool fitIntercept = false,
    OptimizerType optimizer = OptimizerType());
```

which has the parameter `lambda` after three conventional arguments (`data`, `labels` and `numClasses`). We can skip passing `fitIntercept` and `optimizer` since there are the default values. (Technically, we don't even need to pass `lambda` since there is a default value.)

In general to cross-validate you need to specify what machine learning algorithm and metric you are going to use, and then to pass some conventional data-related parameters into one of the cross-validation constructors and all other parameters (which are generally hyperparameters) into the `Evaluate` method.

6.2.2 10-fold cross-validation on weighted decision trees

In the following example we will cross-validate **DecisionTree** (p. 2065) with weights. This is very similar to the previous example, except that we also have instance weights for each point in the dataset. We can generate weights for the dataset from the previous example with the code below:

```
// Random weights for every point from the code snippet above.
arma::rowvec weights = arma::randu<arma::mat>(1, 100);
```

Given those weights for each point, we can now perform cross-validation by also passing the weights to the constructor of **KFoldCV** (p. 1134) :

```
KFoldCV<DecisionTree<>, Accuracy> cv2(10, data, labels, numClasses, weights);
size_t minimumLeafSize = 8;
double weightedDecisionTreeAccuracy = cv2.Evaluate(minimumLeafSize);
```

As with the previous example, internally this call to `cv2.Evaluate()` relies on the following **DecisionTree** (p. 2065) constructor:

```
template<typename MatType, typename LabelsType, typename WeightsType>
DecisionTree(MatType&& data,
             LabelsType&& labels,
             const size_t numClasses,
             WeightsType&& weights,
             const size_t minimumLeafSize = 10,
             const std::enable_if_t<arma::is_arma_type<
                 typename std::remove_reference<WeightsType>::type>::value>*
                 = 0>);
```

6.2.3 10-fold cross-validation with categorical decision trees

DecisionTree (p. 2065) models can be constructed in multiple other ways. For example, if we have a dataset with both categorical and numerical features, we can also perform cross-validation by using the associated **data::DatasetInfo** (p. 376) object. Thus, given some **data::DatasetInfo** (p. 376) object called `datasetInfo` (that perhaps was produced by a call to **data::Load()** (p. 379)), we can perform k-fold cross-validation in a similar manner to the other examples:

```
KFoldCV<DecisionTree<>, Accuracy> cv3(10, data, datasetInfo, labels,
    numClasses);
double decisionTreeWithDIAccuracy = cv3.Evaluate(minimumLeafSize);
```

This particular call to `cv3.Evaluate()` relies on the following **DecisionTree** (p. 2065) constructor:

```
template<typename MatType, typename LabelsType>
DecisionTree(MatType&& data,
             const data::DatasetInfo& datasetInfo,
             LabelsType&& labels,
             const size_t numClasses,
             const size_t minimumLeafSize = 10);
```

6.2.4 Simple cross-validation for linear regression

SimpleCV (p. 1151) has the same interface as **KFoldCV** (p. 1134), except it takes as one of its arguments a proportion (from 0 to 1) of data used as a validation set. For example, to validate **LinearRegression** (p. 1789) with 20% of the data used in the validation set we can write the following code.

```
// Random responses for every point from the code snippet in the beginning of
// the tutorial.
arma::rowvec responses = arma::randu<arma::rowvec>(100);

SimpleCV<LinearRegression, MSE> cv4(0.2, data, responses);
double lrLambda = 0.05;
double lrMSE = cv4.Evaluate(lrLambda);
```

6.3 Performance measures

The cross-validation classes require a performance measure to be specified. **mlpack** has a number of performance measures implemented; below is a list:

- **mlpack::cv::Accuracy** (p. 1127): a simple measure of accuracy
- **mlpack::cv::F1** (p. 1132): the **F1** (p. 1132) score; depends on an averaging strategy
- **mlpack::cv::MSE** (p. 1143): minimum squared error (for regression problems)
- **mlpack::cv::Precision** (p. 1145): the precision, for classification problems
- **mlpack::cv::Recall** (p. 1147): the recall, for classification problems

In addition, it is not difficult to implement a custom performance measure. A class following the structure below can be used:

```
class CustomMeasure
{
    //
    // This evaluates the metric given a trained model and a set of data (with
    // labels or responses) to evaluate on. The data parameter will be a type of
    // Armadillo matrix, and the labels will be the labels that go with the model.
    //
    // If you know that your model is a classification model (and thus that
    // ResponsesType will be arma::Row<size_t>), it is ok to replace the
    // ResponsesType template parameter with arma::Row<size_t>.
    //
    template<typename MAlgorithm, typename DataType, typename ResponsesType>
    static double Evaluate(MAlgorithm& model,
                          const DataType& data,
                          const ResponsesType& labels)
    {
        // Inside the method you should call model.Predict() and compare the
        // values with the labels, in order to get the desired performance measure
        // and return it.
    }
};
```

Once this is implemented, then **CustomMeasure** (or whatever the class is called) is easy to use as a custom performance measure with **KFoldCV** (p. 1134) or **SimpleCV** (p. 1151).

6.4 The KFoldCV and SimpleCV classes

This section provides details about the **KFoldCV** (p.1134) and **SimpleCV** (p.1151) classes. The cross-validation infrastructure is based on heavy amounts of template metaprogramming, so that any **mlpack** learner and any performance measure can be used. Both classes have two required template parameters and one optional parameter:

- **MLAlgorithm**: the type of learner to be used
- **Metric**: the performance measure to be evaluated
- **MatType**: the type of matrix used to store the data

In addition, there are two more template parameters, but these are automatically extracted from the given **MLAlgorithm** class, and users should not need to specify these parameters except when using an unconventional type like `arma::fmat` for data points.

The general structure of the **KFoldCV** (p.1134) and **SimpleCV** (p.1151) classes is split into two parts:

- The constructor: create the object, and store the data for the **MLAlgorithm** training.
- The `Evaluate()` method: take any non-data parameters for the **MLAlgorithm** and calculate the desired performance measure.

This split is important because it defines the API: all data-related parameters are passed to the constructor, whereas algorithm hyperparameters are passed to the `Evaluate()` method.

6.4.1 The KFoldCV and SimpleCV constructors

There are six constructors available for **KFoldCV** (p.1134) and **SimpleCV** (p.1151), each tailored for a different learning situation. Each is given below for the **KFoldCV** (p.1134) class, but the same constructors are also available for the **SimpleCV** (p.1151) class, with the exception that instead of specifying `k`, the number of folds, the **SimpleCV** (p.1151) class takes a parameter between 0 and 1 specifying the percentage of the dataset to use as a validation set.

- `KFoldCV(k, xs, ys)`: this is for unweighted regression applications and two-class classification applications; `xs` is the dataset and `ys` are the responses or labels for each point in the dataset.
- `KFoldCV(k, xs, ys, numClasses)`: this is for unweighted classification applications; `xs` is the dataset, `ys` are the class labels for each data point, and `numClasses` is the number of classes in the dataset.
- `KFoldCV(k, xs, datasetInfo, ys, numClasses)`: this is for unweighted categorical/numeric classification applications; `xs` is the dataset, `datasetInfo` is a **data::DatasetInfo** (p.376) object that holds the types of each dimension in the dataset, `ys` are the class labels for each data point, and `numClasses` is the number of classes in the dataset.
- `KFoldCV(k, xs, ys, weights)`: this is for weighted regression or two-class classification applications; `xs` is the dataset, `ys` are the responses or labels for each point in the dataset, and `weights` are the weights for each point in the dataset.

- `KFoldCV(k, xs, ys, numClasses, weights)`: this is for weighted classification applications; `xs` is the dataset, `ys` are the class labels for each point in the dataset; `numClasses` is the number of classes in the dataset, and `weights` holds the weights for each point in the dataset.
- `KFoldCV(k, xs, datasetInfo, ys, numClasses, weights)`: this is for weighted categorical/numeric classification applications; `xs` is the dataset, `datasetInfo` is a `data::DatasetInfo` (p. 376) object that holds the types of each dimension in the dataset, `ys` are the class labels for each data point, `numClasses` is the number of classes in each dataset, and `weights` holds the weights for each point in the dataset.

Note that the constructor you should use is the constructor that most closely matches the constructor of the machine learning algorithm you would like performance measures of. So, for instance, if you are doing multi-class softmax regression, you could call the constructor `"SoftmaxRegression(xs, ys, numClasses)"`. Therefore, for **KFoldCV** (p. 1134) you would call the constructor `"KFoldCV(k, xs, ys, numClasses)"` and for **SimpleCV** (p. 1151) you would call the constructor `"SimpleCV(pct, xs, ys, numClasses)"`.

6.4.2 The Evaluate() method

The other method that **KFoldCV** (p. 1134) and **SimpleCV** (p. 1151) have is the method to actually calculate the performance measure: `Evaluate()`. The `Evaluate()` method takes any hyperparameters that would follow the data arguments to the constructor or `Train()` method of the given `MLAlgorithm`. The `Evaluate()` method takes no more arguments than that, and returns the desired performance measure on the dataset.

Therefore, let us suppose that we are interested in cross-validating the performance of a softmax regression model, and that we have constructed the appropriate **KFoldCV** (p. 1134) object using the code below:

```
KFoldCV<SoftmaxRegression, Precision> cv(k, data, labels, numClasses);
```

The **SoftmaxRegression** (p. 1811) class has the constructor

```
template<typename OptimizerType = mlpack::optimization::L_BFGS>
SoftmaxRegression(const arma::mat& data,
                  const arma::Row<size_t>& labels,
                  const size_t numClasses,
                  const double lambda = 0.0001,
                  const bool fitIntercept = false,
                  OptimizerType optimizer = OptimizerType());
```

Note that all parameters after `numClasses` are optional. This means that we can specify none or any of them in our call to `Evaluate()`. Below is some example code showing three different ways we can call `Evaluate()` with the `cv` object from the code snippet above.

```
// First, call with all defaults.
double result1 = cv.Evaluate();

// Next, call with lambda set to 0.1 and fitIntercept set to true.
double result2 = cv.Evaluate(0.1, true);

// Lastly, create a custom optimizer to use for optimization, and use a lambda
// value of 0.5 and fit no intercept.
optimization::SGD<> sgd(0.05, 50000); // Step size of 0.05, 50k max iterations.
double result3 = cv.Evaluate(0.5, false, sgd);
```

The same general idea applies to any `MLAlgorithm`: all hyperparameters must be passed to the `Evaluate()` method of **KFoldCV** (p. 1134) or **SimpleCV** (p. 1151).

6.5 Further references

For further documentation, please see the associated Doxygen documentation for each of the relevant classes:

- **mlpack::cv::SimpleCV** (p. 1151)
- **mlpack::cv::KFoldCV** (p. 1134)
- **mlpack::cv::Accuracy** (p. 1127)
- **mlpack::cv::F1** (p. 1132)
- **mlpack::cv::MSE** (p. 1143)
- **mlpack::cv::Precision** (p. 1145)
- **mlpack::cv::Recall** (p. 1147)

If you are interested in implementing a different cross-validation strategy than k-fold cross-validation or simple cross-validation, take a look at the implementations of each of those classes to guide your implementation.

In addition, the **hyperparameter tuner** (p. 49) documentation may also be relevant.

Chapter 7

File formats and loading data in mlpack

7.1 Introduction

mlpack supports a wide variety of data and model formats for use in both its command-line programs and in C++ programs using mlpack via the **mlpack::data::Load()** (p. 379) function. This tutorial discusses the formats that are supported and how to use them.

7.2 Simple examples to load data in C++

The example code snippets below load data from different formats into an Armadillo matrix object (`arma::mat`) or model when using C++.

```
using namespace mlpack;

arma::mat matrix1;
data::Load("dataset.csv", matrix1);

using namespace mlpack;

arma::mat matrix2;
data::Load("dataset.bin", matrix2);

using namespace mlpack;

arma::mat matrix3;
data::Load("dataset.h5", matrix3);

using namespace mlpack;

// ARFF loading is a little different, since sometimes mapping has to be done
// for string types.
arma::mat matrix4;
data::DatasetInfo datasetInfo;
data::Load("dataset.arff", matrix4, datasetInfo);

// The datasetInfo object now holds information about each dimension.

using namespace mlpack;

regression::LogisticRegression lr;
data::Load("model.bin", "logistic_regression_model", lr);
```

7.3 Supported dataset types

Datasets in mlpack are represented internally as sparse or dense numeric matrices (specifically, as `arma::mat` or `arma::sp_mat` or similar). This means that when datasets are loaded from file, they must be converted to a suitable numeric representation. Therefore, in general, datasets on disk should contain only numeric features in order to be loaded successfully by mlpack.

The types of datasets that mlpack can load are roughly the same as the types of matrices that Armadillo can load. However, the load functionality that mlpack provides **only supports loading dense datasets**. When datasets are loaded by mlpack, **the file's type is detected using the file's extension**. mlpack supports the following file types:

- csv (comma-separated values), denoted by .csv or .txt
- tsv (tab-separated values), denoted by .tsv, .csv, or .txt
- ASCII (raw ASCII, with space-separated values), denoted by .txt
- Armadillo ASCII (Armadillo's text format with a header), denoted by .txt
- PGM, denoted by .pgm
- PPM, denoted by .ppm
- Armadillo binary, denoted by .bin
- Raw binary, denoted by .bin (**note: this will be loaded as one-dimensional data, which is likely not what is desired.**)
- HDF5, denoted by .hdf, .hdf5, .h5, or .he5 (**note: HDF5 must be enabled in the Armadillo configuration**)
- ARFF, denoted by .arff (**note: this is not supported by all mlpack command-line programs ; see Categorical features and command line programs (p. 46)**)

Datasets that are loaded by mlpack should be stored with **one row for one point** and **one column for one dimension**. Therefore, a dataset with three two-dimensional points $(0, 1)$, $(3, 1)$, and $(5, -5)$ would be stored in a csv file as:

```
0, 1
3, 1
5, -5
```

As noted earlier, for command-line programs, the format is automatically detected at load time. Therefore, a dataset can be loaded in many ways:

```
$ mlpack_logistic_regression -t dataset.csv -v
[INFO ] Loading 'dataset.csv' as CSV data. Size is 32 x 37749.
...

$ mlpack_logistic_regression -t dataset.txt -v
[INFO ] Loading 'dataset.txt' as raw ASCII formatted data. Size is 32 x 37749.
...

$ mlpack_logistic_regression -t dataset.h5 -v
[INFO ] Loading 'dataset.h5' as HDF5 data. Size is 32 x 37749.
...
```

Similarly, the format to save to is detected by the extension of the given filename.

7.4 Loading simple matrices in C++

When C++ is being written, the `mlpack::data::Load()` (p. 379) and `mlpack::data::Save()` (p. 386) functions are used to load and save datasets, respectively. These functions should be preferred over the built-in Armadillo `.load()` and `.save()` functions.

Matrices in `mlpack` are column-major, meaning that each column should correspond to a point in the dataset and each row should correspond to a dimension; for more information, see **Matrices in `mlpack`** (p. 57). This is at odds with how the data is stored in files; therefore, a transposition is required during load and save. The `mlpack::data::Load()` (p. 379) and `mlpack::data::Save()` (p. 386) functions do this automatically (unless otherwise specified), which is why they are preferred over the Armadillo functions.

To load a matrix from file, the call is straightforward. After creating a matrix object, the data can be loaded:

```
arma::mat dataset; // The data will be loaded into this matrix.
mlpack::data::Load("dataset.csv", dataset);
```

Saving matrices is equally straightforward. The code below generates a random matrix with 10 points in 3 dimensions and saves it to a file as HDF5.

```
// 3 dimensions (rows), with 10 points (columns).
arma::mat dataset = arma::randu<arma::mat>(3, 10);
mlpack::data::Save("dataset.h5", dataset);
```

As with the command-line programs, the type of data to be loaded is automatically detected from the filename extension. For more details, see the `mlpack::data::Load()` (p. 379) and `mlpack::data::Save()` (p. 386) documentation.

7.5 Dealing with sparse matrices

As mentioned earlier, support for loading sparse matrices in `mlpack` is not available at this time. To use a sparse matrix with `mlpack` code, you will have to write a C++ program instead of using any of the command-line tools, because the command-line tools all use dense datasets internally. (There is one exception: the `mlpack_cf` program, for collaborative filtering, loads sparse coordinate lists.)

In addition, the `mlpack::data::Load()` (p. 379) function does not support loading any sparse format; so the best idea is to use undocumented Armadillo functionality to load coordinate lists. Suppose you have a coordinate list file like the one below:

```
$ cat cl.csv
0 0 0.332
1 3 3.126
4 4 1.333
```

This represents a 5x5 matrix with three nonzero elements. We can load this using Armadillo:

```
arma::sp_mat matrix;
matrix.load("cl.csv", arma::coord_ascii);
matrix = matrix.t(); // We must transpose after load!
```

The transposition after loading is necessary if the coordinate list is in row-major format (that is, if each row in the matrix represents a point and each column represents a feature). Be sure that the matrix you use with `mlpack` methods has points as columns and features as rows! See **Matrices in `mlpack`** (p. 57) for more information.

7.6 Categorical features and command line programs

In some situations it is useful to represent data not just as a numeric matrix but also as categorical data (i.e. with numeric but unordered categories). This support is useful for, e.g., decision trees and other models that support categorical features.

In some machine learning situations, such as, e.g., decision trees, categorical data can be used. Categorical data might look like this (in CSV format):

```
0, 1, "true", 3
5, -2, "false", 5
2, 2, "true", 4
3, -1, "true", 3
4, 4, "not sure", 0
0, 7, "false", 6
```

In the example above, the third dimension (which takes values "true", "false", and "not sure") is categorical. mlpack can load and work with this data, but the strings must be mapped to numbers, because all dataset in mlpack are represented by Armadillo matrix objects.

From the perspective of an mlpack command-line program, this support is transparent; mlpack will attempt to load the data file, and if it detects entries in the file that are not numeric, it will map them to numbers and then print, for each dimension, the number of mappings. For instance, if we run the `mlpack_hoeffding_tree` program (which supports categorical data) on the dataset above (stored as `dataset.csv`), we receive this output during loading:

```
$ mlpack_hoeffding_tree -t dataset.csv -l dataset.labels.csv -v
[INFO ] Loading 'dataset.csv' as CSV data. Size is 6 x 4.
[INFO ] 0 mappings in dimension 0.
[INFO ] 0 mappings in dimension 1.
[INFO ] 3 mappings in dimension 2.
[INFO ] 0 mappings in dimension 3.
...
```

Currently, only the `mlpack_hoeffding_tree` program supports loading categorical data, and this is also the only program that supports loading an ARFF dataset.

7.7 Categorical features and C++

When writing C++, loading categorical data is slightly more tricky: the mappings from strings to integers must be preserved. This is the purpose of the `mlpack::data::DatasetInfo` (p. 376) class, which stores these mappings and can be used to load and save time to apply and de-apply the mappings.

When loading a dataset with categorical data, the overload of `mlpack::data::Load()` (p. 379) that takes an `mlpack::data::DatasetInfo` (p. 376) object should be used. An example is below:

```
arma::mat dataset; // Load into this matrix.
mlpack::data::DatasetInfo info; // Store information about dataset in this.

// Load the ARFF dataset.
mlpack::data::Load("dataset.arff", dataset, info);
```

After this load completes, the `info` object will hold the information about the mappings necessary to load the dataset. It is possible to re-use the `DatasetInfo` object to load another dataset with the same mappings. This is useful when, for instance, both a training and test set are being loaded, and it is necessary that the mappings from strings to integers for categorical features are identical. An example is given below.

```
arma::mat trainingData; // Load training data into this matrix.
mlpack::data::DatasetInfo info; // This will store the mappings.

// Load the training data, and create the mappings in the 'info' object.
mlpack::data::Load("training_data.arff", trainingData, info);

// Load the test data, but re-use the 'info' object with the already initialized
// mappings. This means that the same mappings will be applied to the test set.
mlpack::data::Load("test_data.arff", trainingData, info);
```

When saving data, pass the same `DatasetInfo` object it was loaded with in order to unmap the categorical features correctly. The example below demonstrates this functionality: it loads the dataset, increments all non-categorical features by 1, and then saves the dataset with the same `DatasetInfo` it was loaded with.

```
arma::mat dataset; // Load data into this matrix.
mlpack::data::DatasetInfo info; // This will store the mappings.

// Load the dataset.
mlpack::data::Load("dataset.tsv", dataset, info);

// Loop over all features, and add 1 to all non-categorical features.
for (size_t i = 0; i < info.Dimensionality(); ++i)
{
    // The Type() function returns whether or not the data is numeric or
    // categorical.
    if (info.Type(i) != mlpack::data::Datatype::categorical)
        dataset.row(i) += 1.0;
}

// Save the modified dataset using the same DatasetInfo.
mlpack::data::Save("dataset-new.tsv", dataset, info);
```

There is more functionality to the `DatasetInfo` class; for more information, see the `mlpack::data::DatasetInfo` (p. 376) documentation.

7.8 Loading and saving models

Using `boost::serialization` (p. 251), `mlpack` is able to load and save machine learning models with ease. These models can currently be saved in three formats:

- binary (.bin); this is not human-readable, but it is small
- text (.txt); this is sort of human-readable and relatively small
- xml (.xml); this is human-readable but very verbose and large

The type of file to save is determined by the given file extension, as with the other loading and saving functionality in `mlpack`. Below is an example where a dataset stored as TSV and labels stored as ASCII text are used to train a logistic regression model, which is then saved to `model.xml`.

```
$ mlpack_logistic_regression -t training_dataset.tsv -l training_labels.txt \
> -M model.xml
```

Many `mlpack` command-line programs have support for loading and saving models through the `-input_model_file` (-m) and `-output_model_file` (-M) options; for more information, see the documentation for each program (accessible by passing `-help` as a parameter).

7.9 Loading and saving models in C++

mlpack uses the `boost::serialization` (p. 251) library internally to perform loading and saving of models, and provides convenience overloads of `mlpack::data::Load()` (p. 379) and `mlpack::data::Save()` (p. 386) to load and save these models.

To be serializable, a class must implement the method

```
template<typename Archive>
void serialize(Archive& ar, const unsigned int version);
```

Note

For more information on this method and how it works, see the `boost::serialization` (p. 251) documentation at <http://www.boost.org/libs/serialization/doc/>.

Examples of `serialize()` methods can be found in most classes; one fairly straightforward example is found in the `mlpack::math::Range` class (p. 1557). A more complex example is found in the `mlpack::tree::BinarySpace↵ Tree` class (p. 2018).

Using the `mlpack::data::Load()` (p. 379) and `mlpack::data::Save()` (p. 386) classes is easy if the type being saved has a `serialize()` method implemented: simply call either function with a filename, a name for the object to save, and the object itself. The example below, for instance, creates an `mlpack::math::Range` (p. 409) object and saves it as `range.txt`. Then, that range is loaded from file into another `mlpack::math::Range` (p. 409) object.

```
// Create range and save it.
mlpack::math::Range r(0.0, 5.0);
mlpack::data::Save("range.txt", "range", r);

// Load into new range.
mlpack::math::Range newRange;
mlpack::data::Load("range.txt", "range", newRange);
```

It is important to be sure that you load the appropriate type; if you save, for instance, an `mlpack::regression::↵ LogisticRegression` (p. 1795) object and attempt to load it as an `mlpack::math::Range` (p. 409) object, the load will fail and an exception will be thrown. (When the object is saved as binary (.bin), it is possible that the load will not fail, but instead load with mangled data, which is perhaps even worse!)

7.10 Final notes

If the examples here are unclear, it would be worth looking into the ways that `mlpack::data::Load()` (p. 379) and `mlpack::data::Save()` (p. 386) are used in the code. Some example files that may be useful to this end:

- `src/mlpack/methods/logistic_regression/logistic_regression_main.cpp`
- `src/mlpack/methods/hoeffding_trees/hoeffding_tree_main.cpp`
- `src/mlpack/methods/neighbor_search/knn_main.cpp`

If you are interested in adding support for more data types to mlpack, it would be preferable to add the support upstream to Armadillo instead, so that may be a better direction to go first. Then very little code modification for mlpack will be necessary.

Chapter 8

Hyper-Parameter Tuning

8.1 Introduction

mlpack implements a generic hyperparameter tuner that is able to tune both continuous and discrete parameters of various different algorithms. This is an important task—the performance of many machine learning algorithms can be highly dependent on the hyperparameters that are chosen for that algorithm. (One example: the choice of k for a k -nearest-neighbors classifier.)

This hyper-parameter tuner is built on the same general concept as the cross-validation classes (see the **cross-validation tutorial** (p.35)): given some machine learning algorithm, some data, some performance measure, and a set of hyperparameters, attempt to find the hyperparameter set that best optimizes the performance measure on the given data with the given algorithm.

mlpack's implementation of hyperparameter tuning is flexible, and is built in a way that supports many algorithms and many optimizers. At the time of this writing, complex hyperparameter optimization techniques are not available, but the hyperparameter tuner does support these, should they be implemented in the future.

In this tutorial we will see the usage examples of the hyper-parameter tuning module, and also more details about the **HyperParameterTuner** (p.1375) class.

8.2 Basic Usage

The interface of the hyper-parameter tuning module is quite similar to the interface of the **cross-validation module** (p.35). To construct a **HyperParameterTuner** (p.1375) object you need to specify as template parameters what machine learning algorithm, cross-validation strategy, performance measure, and optimization strategy (Grid↵Search will be used by default) you are going to use. Then, you must pass the same arguments as for the cross-validation classes: the data and labels (or responses) to use are given to the constructor, and the possible hyperparameter values are given to the **HyperParameterTuner::Optimize()** (p.1378) method, which returns the best algorithm configuration as a `std::tuple<>`.

Let's see some examples.

Suppose we have the following data to train and validate on.

```
// 100-point 5-dimensional random dataset.
arma::mat data = arma::randu<arma::mat>(5, 100);
// Noisy responses retrieved by a random linear transformation of data.
arma::rowvec responses = arma::randu<arma::rowvec>(5) * data +
    0.1 * arma::randn<arma::rowvec>(100);
```

Given the dataset above, we can use the following code to try to find a good `lambda` value for **LinearRegression** (p. 1789). Here we use **SimpleCV** (p. 1151) instead of k-fold cross-validation to save computation time.

```
// Using 80% of data for training and remaining 20% for assessing MSE.
double validationSize = 0.2;
HyperParameterTuner<LinearRegression, MSE, SimpleCV> hpt(validationSize,
    data, responses);

// Finding a good value for lambda from the discrete set of values 0.0, 0.001,
// 0.01, 0.1, and 1.0.
arma::vec lambdas{0.0, 0.001, 0.01, 0.1, 1.0};
double bestLambda;
std::tie(bestLambda) = hpt.Optimize(lambdas);
```

In this example we have used `GridSearch` (the default optimizer) to find a good value for the `lambda` hyper-parameter. For that we have specified what values should be tried.

8.3 Fixed Arguments

When some hyper-parameters should not be optimized, you can specify values for them with the **Fixed()** (p. 398) method as in the following example of trying to find good `lambda1` and `lambda2` values for **LARS** (p. 1783) (least-angle regression).

```
HyperParameterTuner<LARS, MSE, SimpleCV> hpt2(validationSize, data,
    responses);

// The hyper-parameter tuner should not try to change the transposeData or
// useCholesky parameters.
bool transposeData = true;
bool useCholesky = false;

// We wish only to search for the best lambda1 and lambda2 values.
arma::vec lambda1Set{0.0, 0.001, 0.01, 0.1, 1.0};
arma::vec lambda2Set{0.0, 0.002, 0.02, 0.2, 2.0};

double bestLambda1, bestLambda2;
std::tie(bestLambda1, bestLambda2) = hpt2.Optimize(Fixed(transposeData),
    Fixed(useCholesky), lambda1Set, lambda2Set);
```

Note that for the call to `hpt2.Optimize()`, we have used the same order of arguments as they appear in the corresponding **LARS** (p. 1783) constructor:

```
LARS(const arma::mat& data,
    const arma::rowvec& responses,
    const bool transposeData = true,
    const bool useCholesky = false,
    const double lambda1 = 0.0,
    const double lambda2 = 0.0,
    const double tolerance = 1e-16);
```

8.4 Gradient-Based Optimization

In some cases we may wish to optimize a hyperparameter over the space of all possible real values, instead of providing a grid in which to search. Alternately, we may know approximately optimal values from a grid search for real-valued hyperparameters, but wish to further tune those values.

In this case, we can use a gradient-based optimizer for hyperparameter search. In the following example, we try to optimize the `lambda1` and `lambda2` hyper-parameters for **LARS** (p. 1783) with the `GradientDescent` optimizer.

```
HyperParameterTuner<LARS, MSE, SimpleCV, GradientDescent> hpt3(validationSize,
    data, responses);

// GradientDescent can be adjusted in the following way.
hpt3.Optimizer().StepSize() = 0.1;
hpt3.Optimizer().Tolerance() = 1e-15;

// We can set up values used for calculating gradients.
hpt3.RelativeDelta() = 0.01;
hpt3.MinDelta() = 1e-10;

double initialLambda1 = 0.001;
double initialLambda2 = 0.002;

double bestGDLambda1, bestGDLambda2;
std::tie(bestGDLambda1, bestGDLambda2) = hpt3.Optimize(Fixed(transposeData),
    Fixed(useCholesky), initialLambda1, initialLambda2);
```

8.5 The HyperParameterTuner class

The **HyperParameterTuner** (p. 1375) class is very similar to the **KFoldCV** (p. 1134) and **SimpleCV** (p. 1151) classes (see the "cross-validation tutorial" for more information on those two classes), but there are a few important differences.

First, the **HyperParameterTuner** (p. 1375) accepts five different hyperparameters; only the first three of these are required:

- **MLAlgorithm** This is the algorithm to be used.
- **Metric** This is the performance measure to be used; see **Performance measures** (p. 38) for more information.
- **CVType** This is the type of cross-validation to be used for evaluating the performance measure; this should be **KFoldCV** (p. 1134) or **SimpleCV** (p. 1151).
- **OptimizerType** This is the type of optimizer to use; it can be `GridSearch` or a gradient-based optimizer.
- **MatType** This is the type of data matrix to use. The default is `arma::mat`. This only needs to be changed if you are specifically using sparse data, or if you want to use a numeric type other than `double`.

The last two template parameters are automatically inferred by the **HyperParameterTuner** (p. 1375) and should not need to be manually specified, unless an unconventional data type like `arma::fmat` is being used for data points.

Typically, **SimpleCV** (p. 1151) is a good choice for **CVType** because it takes so much less time to compute than full **KFoldCV** (p. 1134); however, the disadvantage is that **SimpleCV** (p. 1151) might give a somewhat more noisy estimate of the performance measure on unseen test data.

The constructor for the **HyperParameterTuner** (p. 1375) is called with exactly the same arguments as the corresponding **CVType** that has been chosen. For more information on that, please see the **cross-validation constructor tutorial** (p. 39). As an example, if we are using **SimpleCV** (p. 1151) and wish to hold out 20% of the dataset as a validation set, we might construct a **HyperParameterTuner** (p. 1375) like this:

```
// We will use LinearRegression as the MLAlgorithm, and MSE as the performance
// measure. Our dataset is 'dataset' and the responses are 'responses'.
HyperParameterTuner<LinearRegression, MSE, SimpleCV> hpt(0.2, dataset,
    responses);
```

Next, we must set up the hyperparameters to be optimized. If we are doing a grid search with the GridSearch optimizer (the default), then we only need to pass a `std::vector` (for non-numeric hyperparameters) or an `arma::vec` (for numeric hyperparameters) containing all of the possible choices that we wish to search over.

For instance, a set of numeric values might be chosen like this, for the `lambda` parameter (of type `double`):

```
arma::vec lambdaSet = arma::vec("0.0 0.1 0.5 1.0");
```

Similarly, a set of non-numeric values might be chosen like this, for the `intercept` parameter:

```
std::vector<bool> interceptSet = { false, true };
```

Once all of these are set up, the `HyperParameterTuner::Optimize()` (p. 1378) method may be called to find the best set of hyperparameters:

```
bool intercept;
double lambda;
std::tie(lambda, intercept) = hpt.Optimize(lambdaSet, interceptSet);
```

Alternately, the `Fixed()` (p. 398) method (detailed in the **Fixed arguments** (p. 50) section) can be used to fix the values of some parameters.

For continuous optimizers like GradientDescent, a range does not need to be specified but instead only a single value. See the **Gradient-Based Optimization** (p. 51) section for more details.

8.6 Further documentation

For more information on the `HyperParameterTuner` (p. 1375) class, see the `mlpack::hpt::HyperParameterTuner` (p. 1375) class documentation and the **cross-validation tutorial** (p. 35).

Chapter 9

Writing an mpack binding

9.1 Introduction

This tutorial gives some simple examples of how to write an mpack binding that can be compiled for multiple languages. These bindings make up the core of how most users will interact with mpack.

mpack provides the following:

- **mpack::Log** (p. 1538), for debugging / informational / warning / fatal output
- **mpack::CLI** (p. 1117), for parsing command line options or other option

Each of those classes are well-documented, and that documentation should be consulted for further reference.

First, we'll discuss the logging infrastructure, which is useful for giving output that users can see.

9.2 Simple Logging Example

mpack has four logging levels:

- `Log::Debug`
- `Log::Info`
- `Log::Warn`
- `Log::Fatal`

Output to `Log::Debug` does not show (and has no performance penalty) when mlpack is compiled without debugging symbols. Output to `Log::Info` is only shown when the program is run with the `-verbose` (or `-v`) flag. `Log::Warn` is always shown, and `Log::Fatal` will throw a `std::runtime_error` exception, after a newline is sent to it. If mlpack was compiled with debugging symbols, `Log::Fatal` will also print a backtrace, if the necessary libraries are available.

Here is a simple example binding, and its output. Note that instead of `int main()`, we use `static void mlpackMain()`. This is because the automatic binding generator (see [mlpack automatic bindings to other languages](#) (p. 3)) will set up the environment and once that is done, it will call `mlpackMain()`.

```
#include <mlpack/core.hpp>
#include <mlpack/core/util/cli.hpp>
// This definition below means we will only compile for the CLI.
#define BINDING_TYPE BINDING_TYPE_CLI
#include <mlpack/core/util/mlpack_main.hpp>

using namespace mlpack;

static void mlpackMain()
{
    Log::Debug << "Compiled with debugging symbols." << std::endl;

    Log::Info << "Some test informational output." << std::endl;

    Log::Warn << "A warning!" << std::endl;

    Log::Fatal << "Program has crashed." << std::endl;

    Log::Warn << "Made it!" << std::endl;
}
```

Assuming mlpack is installed on the system and the code above is saved in `test.cpp`, this program can be compiled with the following command:

```
$ g++ -o test test.cpp -DDEBUG -g -rdynamic -lmlpack
```

Since we compiled with `-DDEBUG`, if we run the program as below, the following output is shown:

```
$ ./test --verbose
[DEBUG] Compiled with debugging symbols.
[INFO ] Some test informational output.
[WARN ] A warning!
[FATAL] [bt]: (1) /absolute/path/to/file/example.cpp:6: function()
[FATAL] Program has crashed.
terminate called after throwing an instance of 'std::runtime_error'
  what():  fatal error; see Log::Fatal output
Aborted
```

The flags `-g` and `-rdynamic` are only necessary for providing a backtrace. If those flags are not given during compilation, the following output would be shown:

```
$ ./test --verbose
[DEBUG] Compiled with debugging symbols.
[INFO ] Some test informational output.
[WARN ] A warning!
[FATAL] Cannot give backtrace because program was compiled without: -g -rdynamic
[FATAL] For a backtrace, recompile with: -g -rdynamic.
[FATAL] Program has crashed.
terminate called after throwing an instance of 'std::runtime_error'
  what():  fatal error; see Log::Fatal output
Aborted
```

The last warning is not reached, because `Log::Fatal` terminates the program.

Without debugging symbols (i.e. without `-g` and `-DDEBUG`) and without `-verbose`, the following is shown:

```
$ ./test
[WARN ] A warning!
[FATAL] Program has crashed.
terminate called after throwing an instance of 'std::runtime_error'
  what():  fatal error; see Log::Fatal output
Aborted
```

These four outputs can be very useful for both providing informational output and debugging output for your `mlpack` program.

9.3 Simple CLI Example

Through the `mlpack::CLI` (p. 1117) object, command-line parameters can be easily added with the `PROGRAM_INFO`, `PARAM_INT`, `PARAM_DOUBLE`, `PARAM_STRING`, and `PARAM_FLAG` macros.

Here is a sample use of those macros, extracted from `methods/pca/pca_main.cpp`. (Some details have been omitted from the snippet below.)

```
#include <mlpack/core.hpp>
#include <mlpack/core/util/cli.hpp>
#include <mlpack/core/util/mlpack_main.hpp>

// Document program.
PROGRAM_INFO("Principal Components Analysis",
  // Short description.
  "An implementation of several strategies for principal components analysis "
  "(PCA), a common preprocessing step. Given a dataset and a desired new "
  "dimensionality, this can reduce the dimensionality of the data using the "
  "linear transformation determined by PCA.",
  // Long description.
  "This program performs principal components analysis on the given dataset "
  "using the exact, randomized, randomized block Krylov, or QUIC SVD method. "
  "It will transform the data onto its principal components, optionally "
  "performing dimensionality reduction by ignoring the principal components "
  "with the smallest eigenvalues.",
  // "See also" section for generated documentation.
  SEE_ALSO("Principal component analysis on Wikipedia",
    "https://en.wikipedia.org/wiki/Principal_component_analysis"),
  SEE_ALSO("mlpack::pca::PCA C++ class documentation",
    "@doxygen/classmlpack_1_1pca_1_1PCA.html"));

// Parameters for program.
PARAM_MATRIX_IN_REQ("input", "Input dataset to perform PCA on.", "i");
PARAM_MATRIX_OUT("output", "Matrix to save modified dataset to.", "o");
PARAM_INT_IN("new_dimensionality", "Desired dimensionality of output dataset.",
  "d", 0);

using namespace mlpack;

static void mlpackMain()
{
  // Load input dataset.
  arma::mat& dataset = CLI::GetParam<arma::mat>("input");

  size_t newDimension = CLI::GetParam<int>("new_dimensionality");

  ...

  // Now save the results.
  if (CLI::HasParam("output"))
    CLI::GetParam<arma::mat>("output") = std::move(dataset);
}
```

Documentation is automatically generated using those macros, and when the program is run with `--help` the following is displayed:

```
$ mlpack_pca --help
Principal Components Analysis

This program performs principal components analysis on the given dataset. It
will transform the data onto its principal components, optionally performing
dimensionality reduction by ignoring the principal components with the
smallest eigenvalues.

Required options:

--input_file [string]      Input dataset to perform PCA on.
--output_file [string]     Matrix to save modified dataset to.

Options:

--help (-h)               Default help info.
--info [string]           Get help on a specific module or option.
                          Default value "".
--new_dimensionality [int] Desired dimensionality of output dataset.
                          Default value 0.
--verbose (-v)            Display informational messages and the full list
                          of parameters and timers at the end of
                          execution.
```

The **mlpack::CLI** (p. 1117) documentation can be consulted for further and complete documentation. Also useful is to look at other example bindings, found in `src/mlpack/methods/`.

Chapter 10

Matrices in mlpack

10.1 Introduction

mlpack uses Armadillo matrices for matrix support. Armadillo is a fast C++ matrix library which makes use of advanced template techniques to provide the fastest possible matrix operations.

Documentation on Armadillo can be found on their website:

<http://arma.sourceforge.net/docs.html>

Nonetheless, there are a few further caveats for mlpack Armadillo usage.

10.2 Column-major Matrices

Armadillo matrices are stored in a column-major format; this means that on disk, each column is located in contiguous memory.

This means that, for the vast majority of machine learning methods, it is faster to store observations as columns and dimensions as rows. This is counter to most standard machine learning texts!

Major implications of this are for linear algebra. For instance, the covariance of a matrix is typically

$$C = X^T X$$

but for a column-wise matrix, it is

$$C = X X^T$$

and this is very important to keep in mind! If your mlpack code is not working, this may be a factor in why.

10.3 Loading Matrices

mlpack provides a **data::Load()** (p. 379) and **data::Save()** (p. 386) function, which should be used instead of Armadillo's loading and saving functions.

Most machine learning data is stored in row-major format; a CSV, for example, will generally have one observation per line and each column will correspond to a dimension.

The **data::Load()** (p. 379) and **data::Save()** (p. 386) functions transpose the matrix upon loading, meaning that the following CSV:

```
$ cat data.csv
3,3,3,3,0
3,4,4,3,0
3,4,4,3,0
3,3,4,3,0
3,6,4,3,0
2,4,4,3,0
2,4,4,1,0
3,3,3,2,0
3,4,4,2,0
3,4,4,2,0
3,3,4,2,0
3,6,4,2,0
2,4,4,2,0
```

is actually loaded with 5 rows and 13 columns, not 13 rows and 5 columns like the CSV is written. More information on mlpack's loading functionality can be found in **File formats and loading data in mlpack** (p. 43).

This is important to remember!

Chapter 11

mlpack in Python quickstart guide

11.1 Introduction

This page describes how you can quickly get started using mlpack from Python and gives a few examples of usage, and pointers to deeper documentation.

This quickstart guide is also available for **the command-line** (p. 31).

11.2 Installing mlpack

Installing the mlpack bindings for Python is straightforward. It's easy to use conda or pip to do this:

```
pip install mlpack/mlpack3
```

```
conda install -c mlpack mlpack
```

Otherwise, we can build the Python bindings from scratch, as follows. First we have to install the dependencies (the code below is for Ubuntu), then we can build and install mlpack. You can copy-paste the commands into your shell.

```
sudo apt-get install libboost-all-dev g++ cmake libarmadillo-dev python-pip wget
sudo pip install cython setuptools distutils numpy pandas
wget https://www.mlpack.org/files/mlpack-3.1.1.tar.gz
tar -xvzpf mlpack-3.1.1.tar.gz
mkdir -p mlpack-3.1.1/build/ && cd mlpack-3.1.1/build/
cmake ../ && make -j4 && sudo make install
```

More information on the build process and details can be found on the **Building mlpack From Source** (p. 21) page. You may also need to set the environment variable `LD_LIBRARY_PATH` to include `/usr/local/lib/` on most Linux systems.

```
export LD_LIBRARY_PATH=/usr/local/lib/
```

You can also use the mlpack Docker image on Dockerhub, which has all of the Python bindings pre-installed:

```
docker run -it mlpack/mlpack /bin/bash
```

11.3 Simple mlpack quickstart example

As a really simple example of how to use mlpack from Python, let's do some simple classification on a subset of the standard machine learning `covertypes` dataset. We'll first split the dataset into a training set and a testing set, then we'll train an mlpack random forest on the training data, and finally we'll print the accuracy of the random forest on the test dataset.

You can copy-paste this code directly into Python to run it.

```
import mlpack
import pandas as pd
import numpy as np

# Load the dataset from an online URL.  Replace with 'covertypes.csv.gz' if you
# want to use on the full dataset.
df = pd.read_csv('http://www.mlpack.org/datasets/covertypes-small.csv.gz')

# Split the labels.
labels = df['label']
dataset = df.drop('label', 1)

# Split the dataset using mlpack.  The output comes back as a dictionary,
# which we'll unpack for clarity of code.
output = mlpack.preprocess_split(input=dataset,
                                input_labels=labels,
                                test_ratio=0.3)

training_set = output['training']
training_labels = output['training_labels']
test_set = output['test']
test_labels = output['test_labels']

# Train a random forest.
output = mlpack.random_forest(training=training_set,
                              labels=training_labels,
                              print_training_accuracy=True,
                              num_trees=10,
                              minimum_leaf_size=3)

random_forest = output['output_model']

# Predict the labels of the test points.
output = mlpack.random_forest(input_model=random_forest,
                              test=test_set)

# Now print the accuracy.  The 'probabilities' output could also be used
# to generate an ROC curve.
correct = np.sum(
    output['predictions'] == np.reshape(test_labels, (test_labels.shape[0],)))
print(str(correct) + ' correct out of ' + str(len(test_labels)) + ' (' +
      str(100 * float(correct) / float(len(test_labels))) + '%).')
```

We can see that we achieve reasonably good accuracy on the test dataset (80%+); if we use the full `covertypes.csv.gz`, the accuracy should increase significantly (but training will take longer).

It's easy to modify the code above to do more complex things, or to use different mlpack learners, or to interface with other machine learning toolkits.

11.4 What else does mlpack implement?

The example above has only shown a little bit of the functionality of mlpack. Lots of other commands are available with different functionality. A full list of each of these commands and full documentation can be found on the following page:

- [Python documentation](#)

For more information on what mlpack does, see <https://www.mlpack.org/>. Next, let's go through another example for providing movie recommendations with mlpack.

11.5 Using mlpack for movie recommendations

In this example, we'll train a collaborative filtering model using mlpack's `cf()` method. We'll train this on the MovieLens dataset from <https://grouplens.org/datasets/movielens/>, and then we'll use the model that we train to give recommendations.

You can copy-paste this code directly into Python to run it.

```
import mlpack
import pandas as pd
import numpy as np

# First, load the MovieLens dataset. This is taken from files.grouplens.org/
# but reposted on mlpack.org as unpacked and slightly preprocessed data.
ratings = pd.read_csv('http://www.mlpack.org/datasets/ml-20m/ratings-only.csv.gz')
movies = pd.read_csv('http://www.mlpack.org/datasets/ml-20m/movies.csv.gz')

# Hold out 10% of the dataset into a test set so we can evaluate performance.
output = mlpack.preprocess_split(input=ratings, test_ratio=0.1, verbose=True)
ratings_train = output['training']
ratings_test = output['test']

# Train the model. Change the rank to increase/decrease the complexity of the
# model.
output = mlpack.cf(training=ratings_train,
                   test=ratings_test,
                   rank=10,
                   verbose=True,
                   algorithm='RegSVD')
cf_model = output['output_model']

# Now query the 5 top movies for user 1.
output = mlpack.cf(input_model=cf_model,
                   query=[[1]],
                   recommendations=10,
                   verbose=True)

# Get the names of the movies for user 1.
print("Recommendations for user 1:")
for i in range(10):
    print(" " + str(i) + ": " + str(movies.loc[movies['movieId'] ==
        output['output'][0, i]].iloc[0]['title']))
```

Here is some example output, showing that user 1 seems to have good taste in movies:

```
Recommendations for user 1:
0: Casablanca (1942)
1: Pan's Labyrinth (Laberinto del fauno, El) (2006)
2: Godfather, The (1972)
3: Answer This! (2010)
4: Life Is Beautiful (La Vita è bella) (1997)
5: Adventures of Tintin, The (2011)
6: Dark Knight, The (2008)
7: Out for Justice (1991)
8: Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)
9: Schindler's List (1993)
```

11.6 Next steps with mlpack

Now that you have done some simple work with mlpack, you have seen how it can easily plug into a data science workflow in Python. A great thing to do next would be to look at more documentation for the Python mlpack bindings:

- [Python mlpack binding documentation](#)

Also, mlpack is much more flexible from C++ and allows much greater functionality. So, more complicated tasks are possible if you are willing to write C++ (or perhaps Cython). To get started learning about mlpack in C++, the following resources might be helpful:

- [mlpack C++ tutorials](#)
- [mlpack build and installation guide](#)
- [Simple sample C++ mlpack programs](#)
- [mlpack Doxygen documentation homepage](#)

Chapter 12

Simple Sample mlpack Programs

12.1 Introduction

On this page, several simple mlpack examples are contained, in increasing order of complexity. If you compile from the command-line, be sure that your compiler is in C++11 mode. With modern gcc and clang, this should already be the default.

Note

The command-line programs like `knn_main.cpp` and `logistic_regression_main.cpp` from the directory `src/mlpack/methods/` cannot be compiled easily by hand (the same is true for the individual tests in `src/mlpack/tests/`); instead, those should be compiled with CMake, by running, e.g., `make mlpack_knn` or `make mlpack_test`; see **Building mlpack From Source** (p. 21). However, any program that uses mlpack (and is not a part of the library itself) can be compiled easily with g++ or clang from the command line.

12.2 Covariance Computation

A simple program to compute the covariance of a data matrix ("data.csv"), assuming that the data is already centered, and save it to file.

```
// Includes all relevant components of mlpack.
#include <mlpack/core.hpp>

// Convenience.
using namespace mlpack;

int main()
{
    // First, load the data.
    arma::mat data;
    // Use data::Load() which transposes the matrix.
    data::Load("data.csv", data, true);

    // Now compute the covariance. We assume that the data is already centered.
    // Remember, because the matrix is column-major, the covariance operation is
    // transposed.
    arma::mat cov = data * trans(data) / data.n_cols;

    // Save the output.
    data::Save("cov.csv", cov, true);
}
```

12.3 Nearest Neighbor

This simple program uses the `mlpack::neighbor::NeighborSearch` (p. 1627) object to find the nearest neighbor of each point in a dataset using the L1 metric, and then print the index of the neighbor and the distance of it to stdout.

```
#include <mlpack/core.hpp>
#include <mlpack/methods/neighbor_search/neighbor_search.hpp>

using namespace mlpack;
using namespace mlpack::neighbor; // NeighborSearch and NearestNeighborSort
using namespace mlpack::metric; // ManhattanDistance

int main()
{
    // Load the data from data.csv (hard-coded). Use CLI for simple command-line
    // parameter handling.
    arma::mat data;
    data::Load("data.csv", data, true);

    // Use templates to specify that we want a NeighborSearch object which uses
    // the Manhattan distance.
    NeighborSearch<NearestNeighborSort, ManhattanDistance> nn(data);

    // Create the object we will store the nearest neighbors in.
    arma::Mat<size_t> neighbors;
    arma::mat distances; // We need to store the distance too.

    // Compute the neighbors.
    nn.Search(1, neighbors, distances);

    // Write each neighbor and distance using Log.
    for (size_t i = 0; i < neighbors.n_elem; ++i)
    {
        std::cout << "Nearest neighbor of point " << i << " is point "
                  << neighbors[i] << " and the distance is " << distances[i] << ".\n";
    }
}
```

12.4 Other examples

For more complex examples, it is useful to refer to the main executables, found in `src/mlpack/methods/`. A few are listed below.

- `methods/neighbor_search/knn_main.cpp`
- `methods/neighbor_search/kfn_main.cpp`
- `methods/emst/emst_main.cpp`
- `methods/radical/radical_main.cpp`
- `methods/nca/nca_main.cpp`
- `methods/naive_bayes/nbc_main.cpp`
- `methods/pca/pca_main.cpp`
- `methods/lars/lars_main.cpp`
- `methods/linear_regression/linear_regression_main.cpp`
- `methods/gmm/gmm_main.cpp`
- `methods/kmeans/kmeans_main.cpp`

Chapter 13

Sample C++ ML App for Windows

13.1 Introduction

This tutorial will help you create a sample machine learning app using mlpack/C++. Although this app does not cover all the mlpack capabilities, it will walkthrough several APIs to understand how everything connects. This Windows sample app is created using Visual Studio, but you can easily adapt it to a different platform by following the provided source code.

Note

Before starting, make sure you have built mlpack for Windows following this **Windows guide** (p. 27)

13.2 Creating the VS project

- Open Visual Studio and create a new project (Windows Console Application)
- For this sample, the project is named “sample-ml-app”

13.3 Project Configuration

There are different ways in which you can configure your project to link with dependencies. This configuration is for x64 Debug Mode. If you need Release Mode, please change the paths accordingly (assuming you have built mlpack and dependencies in Release Mode).

- Right click on the project and select Properties, select the x64 Debug profile
- Under C/C++ > General > Additional Include Directories add:
 - C:\boost\boost_1_66_0
 - C:\mlpack\armadillo-8.500.1\include
 - C:\mlpack\mlpack-3.1.1\build\include

- Under Linker > Input > Additional Dependencies add:

```
- C:\mlpack\mlpack-3.1.1\build\Debug\mlpack.lib
- C:\boost\boost_1_66_0\lib64-msvc-14.1\libboost_serialization-vc141-mt-gd-x64-1_66.lib
- C:\boost\boost_1_66_0\lib64-msvc-14.1\libboost_program_options-vc141-mt-gd-x64-1_66.lib
```

- Under Build Events > Post-Build Event > Command Line add:

```
- xcopy /y "C:\mlpack\mlpack-3.1.1\build\Debug\mlpack.dll" $(OutDir)
- xcopy /y "C:\mlpack\mlpack-3.1.1\packages\OpenBLAS.0.2.14.1\lib\native\bin\x64\*.dll" $(OutDir)
```

Note

Recent versions of Visual Studio set "Conformance Mode" enabled by default. This causes some issues with the armadillo library. If you encounter this issue, disable "Conformance Mode" under C/C++ > Language.

13.4 The app goal

This app aims to exercise an end-to-end machine learning workflow. We will cover:

- Loading and preparing a dataset
- Training (using Random Forest as example)
- Computing the training accuracy
- Cross-Validation using K-Fold
- Metrics gathering (accuracy, precision, recall, F1)
- Saving the trained model to disk
- Loading the model
- Classifying a new sample

13.5 Headers and namespaces

For this app, we will need to include the following headers (i.e. add into stdafx.h):

```
#include "mlpack/core.hpp"
#include "mlpack/methods/random_forest/random_forest.hpp"
#include "mlpack/methods/decision_tree/random_dimension_select.hpp"
#include "mlpack/core/cv/k_fold_cv.hpp"
#include "mlpack/core/cv/metrics/accuracy.hpp"
#include "mlpack/core/cv/metrics/precision.hpp"
#include "mlpack/core/cv/metrics/recall.hpp"
#include "mlpack/core/cv/metrics/F1.hpp"
```

Also, we will use the following namespaces:

```
using namespace arma;
using namespace mlpack;
using namespace mlpack::tree;
using namespace mlpack::cv;
```

13.6 Loading the dataset

First step is about loading the dataset. Different dataset file formats are supported, but here we load a CSV dataset, and we assume the labels don't require normalization.

Note

Make sure you update the path to your dataset file. For this sample, you can simply copy "mlpack/tests/data/german.csv" and paste into a new "data" folder in your project directory.

```
mat dataset;
bool loaded = mlpack::data::Load("data/german.csv", dataset);
if (!loaded)
  return -1;
```

Then we need to extract the labels from the last dimension of the dataset and remove the labels from the training set:

```
Row<size_t> labels;
labels = conv_to<Row<size_t>>::from(dataset.row(dataset.n_rows - 1));
dataset.shed_row(dataset.n_rows - 1);
```

We now have our dataset ready for training.

13.7 Training

This app will use a Random Forest classifier. At first we define the classifier parameters and then we create the classifier to train it.

```
const size_t numClasses = 2;
const size_t minimumLeafSize = 5;
const size_t numTrees = 10;

RandomForest<GiniGain, RandomDimensionSelect> rf;

rf = RandomForest<GiniGain, RandomDimensionSelect>(dataset, labels,
  numClasses, numTrees, minimumLeafSize);
```

Now that the training is completed, we quickly compute the training accuracy:

```
Row<size_t> predictions;
rf.Classify(dataset, predictions);
const size_t correct = arma::accu(predictions == labels);
cout << "\nTraining Accuracy: " << (double(correct) / double(labels.n_elem));
```

13.8 Cross-Validating

Instead of training the Random Forest directly, we could also use K-fold cross-validation for training, which will give us a measure of performance on a held-out test set. This can give us a better estimate of how the model will perform when given new data. We also define which metric to use in order to assess the quality of the trained model.

```
const size_t k = 10;
KFoldCV<RandomForest<GiniGain, RandomDimensionSelect>, Accuracy> cv(k,
    dataset, labels, numClasses);
double cvAcc = cv.Evaluate(numTrees, minimumLeafSize);
cout << "\nKFoldCV Accuracy: " << cvAcc;
```

To compute other relevant metrics, such as Precision, Recall and F1:

```
double cvPrecision = Precision<Binary>::Evaluate(rf, dataset, labels);
cout << "\nPrecision: " << cvPrecision;

double cvRecall = Recall<Binary>::Evaluate(rf, dataset, labels);
cout << "\nRecall: " << cvRecall;

double cvF1 = F1<Binary>::Evaluate(rf, dataset, labels);
cout << "\nF1: " << cvF1;
```

13.9 Saving the model

Now that our model is trained and validated, we save it to a file so we can use it later. Here we save the model that was trained using the entire dataset. Alternatively, we could extract the model from the cross-validation stage by using `cv.Model()`

```
mlpack::data::Save("mymodel.xml", "model", rf, false);
```

We can also save the model in `bin` format ("mymodel.bin") which would result in a smaller file.

13.10 Loading the model

In a real-life application, you may want to load a previously trained model to classify new samples. We load the model from a file using:

```
mlpack::data::Load("mymodel.xml", "model", rf);
```

13.11 Classifying a new sample

Finally, the ultimate goal is to classify a new sample using the previously trained model. Since the Random Forest classifier provides both predictions and probabilities, we obtain both.

```
mat sample("2 12 2 13 1 2 2 1 3 24 3 1 1 1 1 0 1 0 1 0 0 0");
mat probabilities;
rf.Classify(sample, predictions, probabilities);
u64 result = predictions.at(0);
cout << "\nClassification result: " << result << " , Probabilities: " <<
    probabilities.at(0) << "/" << probabilities.at(1);
```

13.12 Final thoughts

Building real-life applications and services using machine learning can be challenging. Hopefully, this tutorial provides a good starting point that covers the basic workflow you may need to follow while developing it. You can take a look at the entire source code in the provided sample project located here: "doc/examples/sample-ml-app".

Chapter 14

mlpack Timers

14.1 Introduction

mlpack provides a simple timer interface for the timing of machine learning methods. The results of any timers used during the program are displayed at output by any command-line binding, when `--verbose` is given:

```
$ mlpack_knn -r dataset.csv -n neighbors_out.csv -d distances_out.csv -k 5 -v
<...>
[INFO ] Program timers:
[INFO ]   computing_neighbors: 0.010650s
[INFO ]   loading_data: 0.002567s
[INFO ]   saving_data: 0.001115s
[INFO ]   total_time: 0.149816s
[INFO ]   tree_building: 0.000534s
```

14.2 Timer API

The `mlpack::Timer` (p. 1975) class provides three simple methods:

```
void Timer::Start(const char* name);
void Timer::Stop(const char* name);
timeval Timer::Get(const char* name);
```

Each timer is given a name, and is referenced by that name. You can call `Timer::Start()` and `Timer::Stop()` multiple times for a particular timer name, and the result will be the sum of the runs of the timer. Note that `Timer::Stop()` must be called before `Timer::Start()` is called again, otherwise a `std::runtime_error` exception will be thrown.

A "total_time" timer is run by default for each mlpack program.

14.3 Timer Example

Below is a very simple example of timer usage in code.

```
#include <mlpack/core.hpp>
#include <mlpack/core/util/cli.hpp>
#define BINDING_TYPE BINDING_TYPE_CLI
#include <mlpack/core/util/mlpack_main.hpp>

using namespace mlpack;

void mlpackMain()
{
    // Start a timer.
    Timer::Start("some_timer");

    // Do some things.
    DoSomeStuff();

    // Stop the timer.
    Timer::Stop("some_timer");
}
```

If the `--verbose` flag was given to this executable, the time that `"some_timer"` ran for would be printed at the end of the program's output.

Chapter 15

mlpack version information

15.1 mlpack versions in code

mlpack provides a couple of convenience macros and functions to get the version of mlpack. More information (and straightforward code) can be found in **src/mlpack/core/util/version.hpp** (p. 2746).

The following three macros provide major, minor, and patch versions of mlpack (i.e. for mlpack-x.y.z, 'x' is the major version, 'y' is the minor version, and 'z' is the patch version):

```
MLPACK_VERSION_MAJOR  
MLPACK_VERSION_MINOR  
MLPACK_VERSION_PATCH
```

In addition, the function **mlpack::util::GetVersion()** (p. 469) returns the mlpack version as a string (for instance, "mlpack 1.0.8").

15.2 mlpack executable versions

Each mlpack executable supports the `-version` (or `-V`) option, which will print the version of mlpack used. If the version is not an official release but instead from svn trunk, the version will be "mlpack trunk" (and may have a revision number appended to "trunk").

Chapter 16

Alternating Matrix Factorization tutorial

16.1 Introduction

Alternating Matrix Factorization

Alternating matrix factorization decomposes matrix V in the form $V \approx WH$ where W is called the basis matrix and H is called the encoding matrix. V is taken to be of size $n \times m$ and the obtained W is $n \times r$ and H is $r \times m$. The size r is called the rank of the factorization. Factorization is done by alternately calculating W and H respectively while holding the other matrix constant.

mlpack provides:

- a **simple C++ interface** (p. 74) to perform Alternating Matrix Factorization

16.2 Table of Contents

A list of all the sections this tutorial contains.

- **Introduction** (p. 73)
- **Table of Contents** (p. 73)
- **The 'AMF' class** (p. 74)
 - **Using different termination policies** (p. 74)
 - **Using different initialization policies** (p. 75)
 - **Using different update rules** (p. 75)
 - **Using Non-Negative Matrix Factorization with AMF** (p. 76)
 - **Using Singular Value Decomposition with AMF** (p. 76)
- **Further documentation** (p. 76)

16.3 The 'AMF' class

The AMF class is templated with 3 parameters; the first contains the policy used to determine when the algorithm has converged; the second contains the initialization rule for the W and H matrix; the last contains the update rule to be used during each iteration. This templization allows the user to try various update rules, initialization rules, and termination policies (including ones not supplied with mlpack) for factorization.

The class provides the following method that performs factorization

```
template<typename MatType> double Apply(const MatType& V,
                                       const size_t r,
                                       arma::mat& W,
                                       arma::mat& H);
```

16.3.1 Using different termination policies

The AMF implementation comes with different termination policies to support many implemented algorithms. Every termination policy implements the following method which returns the status of convergence.

```
bool IsConverged(arma::mat& W, arma::mat& H)
```

Below is a list of all the termination policies that mlpack contains.

- **mlpack::amf::SimpleResidueTermination** (p. 529)
- **mlpack::amf::SimpleToleranceTermination** (p. 535)
- **mlpack::amf::ValidationRMSETermination** (p. 549)

In **SimpleResidueTermination**, termination decision depends on two factors, value of residue and number of iteration. If the current value of residue drops below the threshold or the number of iterations goes beyond the threshold, positive termination signal is passed to AMF.

In **SimpleToleranceTermination**, termination criterion is met when the increase in residue value drops below the given tolerance. To accommodate spikes, certain number of successive residue drops are accepted. Secondary termination criterion terminates algorithm when iteration count goes beyond the threshold.

ValidationRMSETermination divides the data into 2 sets, training set and validation set. Entries of validation set are nullified in the input matrix. Termination criterion is met when increase in validation set RMSE value drops below the given tolerance. To accommodate spikes certain number of successive validation RMSE drops are accepted. This upper limit on successive drops can be adjusted with `reverseStepCount`. A secondary termination criterion terminates the algorithm when the iteration count goes above the threshold. Though this termination policy is better measure of convergence than the above 2 termination policies, it may cause a decrease in performance since it is computationally expensive.

On the other hand, **CompleteIncrementalTermination** (p. 504) and **IncompleteIncrementalTermination** (p. 510) are just wrapper classes for other termination policies. These policies are used when AMF is applied with **SVDCompleteIncrementalLearning** (p. 542) and **SVDIncompleteIncrementalLearning** (p. 547), respectively.

16.3.2 Using different initialization policies

mlpack currently has 2 initialization policies implemented for AMF:

- **RandomInitialization** (p. 528)
- **RandomAcolInitialization** (p. 526)

`RandomInitialization` initializes matrices `W` and `H` with random uniform distribution while `RandomAcolInitialization` initializes the `W` matrix by averaging `p` randomly chosen columns of `V`. In the case of `RandomAcolInitialization`, `p` is a template parameter.

To implement their own initialization policy, users need to define the following function in their class.

```
template<typename MatType>
inline static void Initialize(const MatType& V,
                             const size_t r,
                             arma::mat& W,
                             arma::mat& H)
```

16.3.3 Using different update rules

mlpack implements the following update rules for the AMF class:

- **AMFALSUpdate** (p. 516)
- **NMFMultiplicativeDistanceUpdate** (p. 519)
- **NMFMultiplicativeDivergenceUpdate** (p. 523)
- **SVDBatchLearning** (p. 539)
- **SVDIncompleteIncrementalLearning** (p. 547)
- **SVDCompleteIncrementalLearning** (p. 542)

Non-Negative Matrix factorization can be achieved with `NMFALSUpdate`, `NMFMultiplicativeDivergenceUpdate` or `NMFMultiplicativeDistanceUpdate`. `NMFALSUpdate` implements a simple Alternating Least Squares optimization while the other rules implement algorithms given in the paper 'Algorithms for Non-negative Matrix Factorization'.

The remaining update rules perform the singular value decomposition of the matrix `V`. This SVD factorization is optimized for use by mlpack's collaborative filtering code (**Collaborative filtering tutorial** (p. 103)). This use of SVD factorizers for collaborative filtering is described in the paper 'A Guide to Singular Value Decomposition for Collaborative Filtering' by Chih-Chao Ma. For further details about the algorithms refer to the respective class documentation.

16.3.4 Using Non-Negative Matrix Factorization with AMF

The use of AMF for Non-Negative Matrix factorization is simple. The AMF module defines **NMFALSFactorizer** (p. 259) which can be used directly without knowing the internal structure of AMF. For example:

```
#include <mlpack/core.hpp>
#include <mlpack/methods/amf/amf.hpp>

using namespace std;
using namespace arma;
using namespace mlpack::amf;

int main()
{
    NMFALSFactorizer nmf;
    mat W, H;
    mat V = randu<mat>(100, 100);
    double residue = nmf.Apply(V, W, H);
}
```

NMFALSFactorizer uses SimpleResidueTermination, which is most preferred with Non-Negative Matrix factorizers. The initialization of W and H in NMFALSFactorizer is random. The Apply() function returns the residue obtained by comparing the constructed matrix $W * H$ with the original matrix V.

16.3.5 Using Singular Value Decomposition with AMF

mlpack has the following SVD factorizers implemented for AMF:

- **SVDBatchFactorizer** (p. 259)
- **SVDIncompleteIncrementalFactorizer** (p. 260)
- **SVDCompleteIncrementalFactorizer** (p. 260)

Each of these factorizers takes a template parameter MatType, which specifies the type of the matrix V (dense or sparse—these have types arma::mat and arma::sp_mat, respectively). When the matrix to be factorized is relatively sparse, specifying MatType = arma::sp_mat can provide a runtime boost.

```
#include <mlpack/core.hpp>
#include <mlpack/methods/amf/amf.hpp>

using namespace std;
using namespace arma;
using namespace mlpack::amf;

int main()
{
    sp_mat V = randu<sp_mat>(100, 100);
    mat W, H;

    SVDBatchFactorizer<sp_mat> svd;
    double residue = svd.Apply(V, W, H);
}
```

16.4 Further documentation

For further documentation on the AMF class, consult the **complete API documentation** (p. 499).

Chapter 17

Neural Network tutorial

17.1 Introduction

There is vast literature on neural networks and their uses, as well as strategies for choosing initial points effectively, keeping the algorithm from converging in local minima, choosing the best model structure, choosing the best optimizers, and so forth. mlpack implements many of these building blocks, making it very easy to create different neural networks in a modular way.

mlpack currently implements two easy-to-use forms of neural networks: **Feed-Forward Networks** (this includes convolutional neural networks) and **Recurrent Neural Networks**.

17.2 Table of Contents

This tutorial is split into the following sections:

- **Introduction** (p. 77)
- **Table of Contents** (p. 77)
- **Model API** (p. 78)
- **Layer API** (p. 81)
- **Model Setup & Training** (p. 83)
- **Saving & Loading** (p. 84)
- **Extracting Parameters** (p. 86)
- **Further documentation** (p. 86)

17.3 Model API

There are two main neural network classes that are meant to be used as container for neural network layers that **mlpack** implements; each class is suited to a different setting:

- **FFN**: the Feed Forward Network model provides a means to plug layers together in a feed-forward fully connected manner. This is the 'standard' type of deep learning model, and includes convolutional neural networks (CNNs).
- **RNN**: the Recurrent Neural Network model provides a means to consider successive calls to forward as different time-steps in a sequence. This is often used for time sequence modeling tasks, such as predicting the next character in a sequence.

Below is some basic guidance on what should be used. Note that the question of "which algorithm should be used" is a very difficult question to answer, so the guidance below is just that—guidance—and may not be right for a particular problem.

- **Feed-forward Networks** allow signals or inputs to travel one way only. There is no feedback within the network; for instance, the output of any layer does only affect the upcoming layer. That makes Feed-Forward Networks straightforward and very effective. They are extensively used in pattern recognition and are ideally suitable for modeling relationships between a set of input and one or more output variables.
- **Recurrent Networks** allow signals or inputs to travel in both directions by introducing loops in the network. Computations derived from earlier inputs are fed back into the network, which gives the recurrent network some kind of memory. RNNs are currently being used for all kinds of sequential tasks; for instance, time series prediction, sequence labeling, and sequence classification.

In order to facilitate consistent implementations, the **FFN** and **RNN** classes have a number of methods in common:

- **Train()**: trains the initialized model on the given input data. Optionally an optimizer object can be passed to control the optimization process.
- **Predict()**: predicts the responses to a given set of predictors. Note the responses will reflect the output of the specified output layer.
- **Add()**: this method can be used to add a layer to the model.

Note

To be able to optimize the network, both classes implement the **OptimizerFunction** API; see **Optimizer API** for more information. In short, the **FFN** and **RNN** class implement two methods: **Evaluate()** and **Gradient()**. This enables the optimization given some learner and some performance measure.

Similar to the existing layer infrastructure, the **FFN** and **RNN** classes are very extensible, having the following template arguments; which can be modified to change the behavior of the network:

- **OutputLayerType**: this type defines the output layer used to evaluate the network; by default, **NegativeLogLikelihood** is used.

- `InitializationRuleType`: this type defines the method by which initial parameters are set; by default, `RandomInitialization` is used.

```
template<
    typename OutputLayerType = NegativeLogLikelihood<>,
    typename InitializationRuleType = RandomInitialization
>
class FNN;
```

Internally, the `FNN` and `RNN` class keeps an instantiated `OutputLayerType` class (which can be given in the constructor). This is useful for using different loss functions like the Negative-Log-Likelihood function or the `VRClassReward` function, which takes an optional score parameter. Therefore, you can write a non-static `OutputLayerType` class and use it seamlessly in combination with the `FNN` and `RNN` class. The same applies to the `InitializationRuleType` template parameter.

By choosing different components for each of these template classes in conjunction with the `Add()` method, a very arbitrary network object can be constructed.

Below are several examples of how the `FNN` and `RNN` classes might be used. The first examples focus on the `FNN` class, and the last shows how the `RNN` class can be used.

The simplest way to use the `FNN<>` class is to pass in a dataset with the corresponding labels, and receive the classification in return. Note that the dataset must be column-major – that is, one column corresponds to one point. See the **matrices guide** (p. 57) for more information.

The code below builds a simple feed-forward network with the default options, then queries for the assignments for every point in the `queries` matrix.

Note

The number of inputs in the above graph doesn't match with the real number of features in the thyroid dataset and are just used as an abstract representation.

```
// Load the training set.
arma::mat dataset;
data::Load("thyroid_train.csv", dataset, true);

// Split the labels from the training set.
arma::mat trainData = dataset.submat(0, 0, dataset.n_rows - 4,
    dataset.n_cols - 1);

// Split the data from the training set.
arma::mat trainLabelsTemp = dataset.submat(dataset.n_rows - 3, 0,
    dataset.n_rows - 1, dataset.n_cols - 1);

// Initialize the network.
FFN<> model;
model.Add<Linear<> >(trainData.n_rows, 8);
model.Add<SigmoidLayer<> >();
model.Add<Linear<> >(8, 3);
model.Add<LogSoftMax<> >();

// Train the model.
model.Train(trainData, trainLabels);

// Use the Predict method to get the assignments.
arma::mat assignments;
model.Predict(trainData, assignments);
```

Now, the matrix assignments holds the classification of each point in the dataset.

In the next example, we create simple noisy sine sequences, which are trained later on, using the RNN class in the `RNNModel()` method.

```
void GenerateNoisySines(arma::mat& data,
    arma::mat& labels,
    const size_t points,
    const size_t sequences,
    const double noise = 0.3)
{
    arma::colvec x = arma::linspace<arma::Col<double>>(0,
        points - 1, points) / points * 20.0;
    arma::colvec y1 = arma::sin(x + arma::as_scalar(arma::randu(1)) * 3.0);
    arma::colvec y2 = arma::sin(x / 2.0 + arma::as_scalar(arma::randu(1)) * 3.0);

    data = arma::zeros(points, sequences * 2);
    labels = arma::zeros(2, sequences * 2);

    for (size_t seq = 0; seq < sequences; seq++)
    {
        data.col(seq) = arma::randu(points) * noise + y1 +
            arma::as_scalar(arma::randu(1) - 0.5) * noise;
        labels(0, seq) = 1;

        data.col(sequences + seq) = arma::randu(points) * noise + y2 +
            arma::as_scalar(arma::randu(1) - 0.5) * noise;
        labels(1, sequences + seq) = 1;
    }
}

void RNNModel()
{
    const size_t rho = 10;

    // Generate 12 (2 * 6) noisy sines. A single sine contains rho
    // points/features.
    arma::mat input, labelsTemp;
    GenerateNoisySines(input, labelsTemp, rho, 6);

    arma::mat labels = arma::zeros<arma::mat>(rho, labelsTemp.n_cols);
    for (size_t i = 0; i < labelsTemp.n_cols; ++i)
    {
```



```

    const int value = arma::as_scalar(arma::find(
        arma::max(labelsTemp.col(i)) == labelsTemp.col(i), 1)) + 1;
    labels.col(i).fill(value);
}

Add<> add(4);
Linear<> lookup(1, 4);
SigmoidLayer<> sigmoidLayer;
Linear<> linear(4, 4);
Recurrent<> recurrent(add, lookup, linear, sigmoidLayer, rho);

RNN<> model(rho);
model.Add<IdentityLayer<> >();
model.Add(recurrent);
model.Add<Linear<> >(4, 10);
model.Add<LogSoftMax<> >();

StandardSGD opt(0.1, 1, input.n_cols /* 1 epoch */, -100);
model.Train(input, labels, opt);
}

```

For further examples on the usage of the ann classes, see `mlpack models`.

17.4 Layer API

In order to facilitate consistent implementations, we have defined a `LayerType` API that describes all the methods that a `layer` may implement. `mlpack` offers a few variations of this API, each designed to cover some of the model characteristics mentioned in the previous section. Any `layer` requires the implementation of a `Forward()` method. The interface looks like:

```

template<typename eT>
void Forward(const arma::Mat<eT>&& input, arma::Mat<eT>&& output);

```

The method should calculate the output of the layer given the input matrix and store the result in the given output matrix. Next, any `layer` must implement the `Backward()` method, which uses certain computations obtained during the forward pass and should calculate the function $f(x)$ by propagating x backward through f :

```

template<typename eT>
void Backward(const arma::Mat<eT>&& input,
              arma::Mat<eT>&& gy,
              arma::Mat<eT>&& g);

```

Finally, if the layer is differentiable, the layer must also implement a `Gradient()` method:

```

template<typename eT>
void Gradient(const arma::Mat<eT>&& input,
              arma::Mat<eT>&& error,
              arma::Mat<eT>&& gradient);

```

The `Gradient` function should calculate the gradient with respect to the input activations `input` and calculated errors `error` and place the results into the gradient matrix object `gradient` that is passed as an argument.

Note

Note that each method accepts a template parameter `InputType`, `OutputType` or `GradientType`, which may be `arma::mat` (dense Armadillo matrix) or `arma::sp_mat` (sparse Armadillo matrix). This allows support for both sparse-supporting and non-sparse-supporting `layer` without explicitly passing the type.

In addition, each layer must implement the `Parameters()`, `InputParameter()`, `OutputParameter()`, `Delta()` methods, differentiable layer should also provide access to the gradient by implementing the `Gradient()`, `Parameters()` member function. Note each function is a single line that looks like:

```
OutputDataType const& Parameters() const { return weights; }
```

Below is an example that shows each function with some additional boilerplate code.

Note

Note this is not an actual layer but instead an example that exists to show and document all the functions that `mlpack::layer` must implement. For a better overview of the various layers, see `mlpack::ann` (p. 262). Also be aware that the implementations of each of the methods in this example are entirely fake and do not work; this example exists for its API, not its implementation.

Note that layer sometimes have different properties. These properties are known at compile-time through the `mlpack::ann::LayerTraits` (p. 769) class, and some properties may imply the existence (or non-existence) of certain functions. Refer to the `LayerTraits` `LayerTraits` for more documentation on that.

The two template parameters below must be template parameters to the layer, in the order given below. More template parameters are fine, but they must come after the first two.

- `InputDataType`: this defines the internally used input type for example to store the parameter matrix. Note, a layer could be built on a dense matrix or a sparse matrix. All `mlpack` trees should be able to support any Armadillo-compatible matrix type. When the layer is written it should be assumed that `MatType` has the same functionality as `arma::mat`. Note that
- `OutputDataType`: this defines the internally used input type for example to store the parameter matrix. Note, a layer could be built on a dense matrix or a sparse matrix. All `mlpack` trees should be able to support any Armadillo-compatible matrix type. When the layer is written it should be assumed that `MatType` has the same functionality as `arma::mat`.

```
template<typename InputDataType = arma::mat,
        typename OutputDataType = arma::mat>
class ExampleLayer
{
public:
    ExampleLayer(const size_t inSize, const size_t outSize) :
        inputSize(inSize), outputSize(outSize)
    {
        /* Nothing to do here */
    }
}
```

The constructor for `ExampleLayer` will build the layer given the input and output size. Note that, if the input or output size information isn't used internally it's not necessary to provide a specific constructor. Also, one could add additional or other information that are necessary for the layer construction. One example could be:

```
ExampleLayer(const double ratio = 0.5) : ratio(ratio) { /* Nothing to do here*/ }
```

When this constructor is finished, the entire layer will be built and is ready to be used. Next, as pointed out above, each layer has to follow the `LayerType` API, so we must implement some additional functions.

```
template<typename InputType, typename OutputType>
void Forward(const InputType&& input, OutputType&& output)
{
    output = arma::ones(input.n_rows, input.n_cols);
}

template<typename InputType, typename ErrorType, typename GradientType>
void Backward(const InputType&& input, ErrorType&& gy, GradientType&& g)
{
    g = arma::zeros(gy.n_rows, gy.n_cols) + gy;
}

template<typename InputType, typename ErrorType, typename GradientType>
void Gradient(const InputType&& input,
              ErrorType&& error,
              GradientType&& gradient)
{
    gradient = arma::zeros(input.n_rows, input.n_cols) * error;
}
```

The three functions `Forward()`, `Backward()` and `Gradient()` (which is needed for a differentiable layer) contain the main logic of the layer. The following functions are just to access and manipulate the different layer parameters.

```
OutputDataType& Parameters() { return weights; }
InputDataType& InputParameter() { return inputParameter; }
OutputDataType& OutputParameter() { return outputParameter; }
OutputDataType& Delta() { return delta; }
OutputDataType& Gradient() { return gradient; }
```

Since some of these methods return internal class members we have to define them.

```
private:
    size_t inSize, outSize;
    OutputDataType weights, delta, gradient, outputParameter;
    InputDataType inputParameter;
```

Note some members are just here so `ExampleLayer` compiles without warning. For instance, `inSize` is not required to be a member of every type of layer.

There is one last method that is especially interesting for a layer that shares parameter. Since the layer weights are set once the complete model is defined, it's not possible to split the weights during the construction time. To solve this issue, a layer can implement the `Reset()` method which is called once the layer parameter is set.

17.5 Model Setup & Training

Once the base container is selected (FNN or RNN), the `Add` method can be used to add layers to the model. The code below adds two linear layers to the model—the first takes 512 units as input and gives 256 output units, and the second takes 256 units as input and gives 128 output units.

```
FFN<> model;
model.Add<Linear<> >(512, 256);
model.Add<Linear<> >(256, 128);
```

The model is trained on Armadillo matrices. For training a model, you will typically use the `Train()` function:

```
arma::mat trainingSet, trainingLabels;
model.Train(trainingSet, trainingLabels);
```

You can use mlpack's `Load()` (p. 379) function to load a dataset like this:

```
arma::mat trainingSet;
data::Load("dataset.csv", dataset, true);
```

```
$ cat dataset.csv
0, 1, 4
1, 0, 5
1, 1, 1
2, 0, 2
```

The type does not necessarily need to be a CSV; it can be any supported storage format, assuming that it is a coordinate-format file in the format specified above. For more information on mlpack file formats, see the documentation for `mlpack::data::Load()` (p. 379).

Note

It's often a good idea to normalize or standardize your data, for example using:

```
for (size_t i = 0; i < dataset.n_cols; ++i)
  dataset.col(i) /= norm(dataset.col(i), 2);
```

Also, it is possible to retrain a model with new parameters or with a new reference set. This is functionally equivalent to creating a new model.

17.6 Saving & Loading

Using `boost::serialization` (p. 251) (for more information about the internals see `Serialization - Boost C++ Libraries`), mlpack is able to load and save machine learning models with ease. To save a trained neural network to disk. The example below builds a model on the `thyroid` dataset and then saves the model to the file `model.xml` for later use.

```
// Load the training set.
arma::mat dataset;
data::Load("thyroid_train.csv", dataset, true);

// Split the labels from the training set.
arma::mat trainData = dataset.submat(0, 0, dataset.n_rows - 4,
  dataset.n_cols - 1);

// Split the data from the training set.
arma::mat trainLabelsTemp = dataset.submat(dataset.n_rows - 3, 0,
  dataset.n_rows - 1, dataset.n_cols - 1);

// Initialize the network.
```

```
FFN<> model;
model.Add<Linear<> >(trainData.n_rows, 3);
model.Add<SigmoidLayer<> >();
model.Add<LogSoftMax<> >();

// Train the model.
model.Train(trainData, trainLabels);

// Use the Predict method to get the assignments.
arma::mat assignments;
model.Predict(trainData, assignments);

data::Save("model.xml", "model", model, false);
```

After this, the file `model.xml` will be available in the current working directory.

Now, we can look at the output model file, `model.xml` :

```
$ cat model.xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE boost_serialization>
<boost_serialization signature="serialization::archive" version="15">
<model class_id="0" tracking_level="0" version="0">
  <parameter class_id="1" tracking_level="1" version="0" object_id="_0">
    <n_rows>66</n_rows>
    <n_cols>1</n_cols>
    <n_elem>66</n_elem>
    <vec_state>0</vec_state>
    <item>-7.55971528334903642e+00</item>
    <item>-9.95435955058058930e+00</item>
    <item>9.31133928948225353e+00</item>
    <item>-5.36784434861701953e+00</item>
    ...
  </parameter>
  <width>0</width>
  <height>0</height>
  <currentInput object_id="_1">
    <n_rows>0</n_rows>
    <n_cols>0</n_cols>
    <n_elem>0</n_elem>
    <vec_state>0</vec_state>
  </currentInput>
  <network class_id="2" tracking_level="0" version="0">
    <count>3</count>
    <item_version>0</item_version>
    <item class_id="3" tracking_level="0" version="0">
      <which>18</which>
      <value class_id="4" tracking_level="1" version="0" object_id="_2">
        <inSize>21</inSize>
        <outSize>3</outSize>
      </value>
    </item>
    <item>
      <which>2</which>
      <value class_id="5" tracking_level="1" version="0" object_id="_3"></value>
    </item>
    <item>
      <which>20</which>
      <value class_id="6" tracking_level="1" version="0" object_id="_4"></value>
    </item>
  </network>
</model>
</boost_serialization>
```

As you can see, the `<parameter>` section of `model.xml` contains the trained network weights. We can see that this section also contains the network input size, which is 66 rows and 1 column. Note that in this example, we used three different layers, as can be seen by looking at the `<network>` section. Each node has a unique id that is used to reconstruct the model when loading.

The models can also be saved as `.bin` or `.txt`; the `.xml` format provides a human-inspectable format (though the models tend to be quite complex and may be difficult to read). These models can then be re-used to be used for classification or other tasks.

So, instead of saving or training a network, `mlpack` can also load a pre-trained model. For instance, the example below will load the model from `model.xml` and then generate the class predictions for the `thyroid` test dataset.

```
data::Load("thyroid_test.csv", dataset, true);  
arma::mat testData = dataset.submat(0, 0, dataset.n_rows - 4,  
    dataset.n_cols - 1);  
  
data::Load("model.xml", "model", model);  
  
arma::mat predictions;  
model.Predict(testData, predictions);
```

This enables the possibility to distribute a model without having to train it first or simply to save a model for later use. Note that loading will also work on different machines.

17.7 Extracting Parameters

To access the weights from the neural network layers, you can call the following function on any initialized network:

```
model.Parameters();
```

which will return the complete model parameters as an armadillo matrix object; however often it is useful to not only have the parameters for the complete network, but the parameters of a specific layer. Another method, `Model()`, makes this easily possible:

```
model.Model()[1].Parameters();
```

In the example above, we get the weights of the second layer.

17.8 Further documentation

For further documentation on the ann classes, consult the **complete API documentation** (p. 262).

Chapter 18

Approximate furthest neighbor search (mlpack_approx_kfn) tutorial

18.1 Introduction

mlpack implements multiple strategies for approximate furthest neighbor search in its `mlpack_approx_kfn` and `mlpack_kfn` programs (each program corresponds to different techniques). This tutorial discusses what problems these algorithms solve and how to use each of the techniques that **mlpack** implements.

mlpack implements five approximate furthest neighbor search algorithms:

- brute-force search (in `mlpack_kfn`)
- single-tree search (in `mlpack_kfn`)
- dual-tree search (in `mlpack_kfn`)
- query-dependent approximate furthest neighbor (QDAFN) (in `mlpack_approx_kfn`)
- DrusillaSelect (in `mlpack_approx_kfn`)

These methods are described in the following papers:

```
@inproceedings{curtin2013tree,
  title={Tree-Independent Dual-Tree Algorithms},
  author={Curtin, Ryan R. and March, William B. and Ram, Parikshit and Anderson,
    David V. and Gray, Alexander G. and Isbell Jr., Charles L.},
  booktitle={Proceedings of The 30th International Conference on Machine
    Learning (ICML '13)},
  pages={1435--1443},
  year={2013}
}

@incollection{pagh2015approximate,
  title={Approximate furthest neighbor in high dimensions},
  author={Pagh, Rasmus and Silvestri, Francesco and Sivertsen, Johan and Skala,
    Matthew},
  booktitle={Similarity Search and Applications},
  pages={3--14},
  year={2015},
  publisher={Springer}
}
```

```
@incollection{curtin2016fast,
  title={Fast approximate furthest neighbors with data-dependent candidate
    selection},
  author={Curtin, Ryan R., and Gardner, Andrew B.},
  booktitle={Similarity Search and Applications},
  pages={221--235},
  year={2016},
  publisher={Springer}
}

@article{curtin2018exploiting,
  title={Exploiting the structure of furthest neighbor search for fast
    approximate results},
  author={Curtin, Ryan R., and Echauz, Javier, and Gardner, Andrew B.},
  journal={Information Systems},
  year={2018},
  publisher={Elsevier}
}
```

The problem of furthest neighbor search is simple, and is the opposite of the much-more-studied nearest neighbor search problem. Given a set of reference points R (the set in which we are searching), and a set of query points Q (the set of points for which we want the furthest neighbor), our goal is to return the k furthest neighbors for each query point in Q :

$$k\text{-argmax}_{p_r \in R} d(p_q, p_r).$$

In order to solve this problem, **mlpack** provides a number of interfaces.

- two **simple command-line executables** (p. 90) to calculate approximate furthest neighbors
- a simple **C++ class for QDAFN** (p. 98)
- a simple **C++ class for DrusillaSelect** (p. 96)
- a simple C++ class for tree-based and brute-force search

18.2 Table of Contents

A list of all the sections this tutorial contains.

- **Introduction** (p. 87)
- **Table of Contents** (p. 88)
- **Which algorithm should be used?** (p. 89)
- **Command-line 'mlpack_approx_kfn' and 'mlpack_kfn'** (p. 90)
 - **Calculate 5 furthest neighbors with default options** (p. 90)
 - **Specifying algorithm parameters for DrusillaSelect** (p. 91)
 - **Using QDAFN instead of DrusillaSelect** (p. 91)
 - **Printing results quality with exact distances** (p. 92)
 - **Using cached exact distances for quality results** (p. 93)
 - **Using tree-based approximation with mlpack_kfn** (p. 93)

- Different algorithms with `mlpack_kfn` (p. 94)
- Saving a model for later use (p. 95)
- Final command-line program notes (p. 96)
- **DrusillaSelect C++ class** (p. 96)
 - Approximate furthest neighbors with defaults (p. 96)
 - Custom numbers of tables and projections (p. 97)
 - Accessing the candidate set (p. 97)
 - Retraining on a new reference set (p. 98)
 - Running on sparse data (p. 98)
- **QDAFN C++ class** (p. 98)
 - Approximate furthest neighbors with defaults (p. 99)
 - Custom numbers of tables and projections (p. 99)
 - Accessing the candidate set (p. 99)
 - Retraining on a new reference set (p. 100)
 - Running on sparse data (p. 100)
- **KFN C++ class** (p. 100)
 - Simple furthest neighbors example (p. 101)
 - Retraining on a new reference set (p. 101)
 - Searching in single-tree mode (p. 101)
 - Searching in brute-force mode (p. 102)
- **Further documentation** (p. 102)

18.3 Which algorithm should be used?

There are three algorithms for furthest neighbor search that **mlpack** implements, and each is suited to a different setting. Below is some basic guidance on what should be used. Note that the question of "which algorithm should be used" is a very difficult question to answer, so the guidance below is just that—guidance—and may not be right for a particular problem.

- `DrusillaSelect` is very fast and will perform extremely well for datasets with outliers or datasets with structure (like low-dimensional datasets embedded in high dimensions)
- `QDAFN` is a random approach and therefore should be well-suited for datasets with little to no structure
- The tree-based approaches (the `KFN` class and the `mlpack_kfn` program) is best suited for low-dimensional datasets, and is most effective when very small levels of approximation are desired, or when exact results are desired.
- Dual-tree search is most useful when the query set is large and structured (like for all-furthest-neighbor search).
- Single-tree search is more useful when the query set is small.

18.4 Command-line 'mlpack_approx_kfn' and 'mlpack_kfn'

mlpack provides two command-line programs to solve approximate furthest neighbor search:

- `mlpack_approx_kfn`, for the QDAFN and DrusillaSelect approaches
- `mlpack_kfn`, for exact and approximate tree-based approaches

These two programs allow a large number of algorithms to be used to find approximate furthest neighbors. Note that the `mlpack_kfn` program is also documented by the **Command-Line 'mlpack_knn'** (p. 152) section of the **Neighbor↔ Search tutorial (k-nearest-neighbors)** (p. 151) page, as it shares options with the `mlpack_knn` program.

Below are several examples of how the `mlpack_approx_kfn` and `mlpack_kfn` programs might be used. The first examples focus on the `mlpack_approx_kfn` program, and the last few show how `mlpack_kfn` can be used to produce approximate results.

18.4.1 Calculate 5 furthest neighbors with default options

Here we have a query dataset `queries.csv` and a reference dataset `refs.csv` and we wish to find the 5 furthest neighbors of every query point in the reference dataset. We may do that with the `mlpack_approx_kfn` algorithm, using the default of the DrusillaSelect algorithm with default parameters.

```
$ mlpack_approx_kfn -q queries.csv -r refs.csv -v -k 5 -n n.csv -d d.csv
[INFO ] Loading 'refs.csv' as CSV data. Size is 3 x 1000.
[INFO ] Building DrusillaSelect model...
[INFO ] Model built.
[INFO ] Loading 'queries.csv' as CSV data. Size is 3 x 1000.
[INFO ] Searching for 5 furthest neighbors with DrusillaSelect...
[INFO ] Search complete.
[INFO ] Saving CSV data to 'n.csv'.
[INFO ] Saving CSV data to 'd.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   algorithm: ds
[INFO ]   calculate_error: false
[INFO ]   distances_file: d.csv
[INFO ]   exact_distances_file: ""
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_model_file: ""
[INFO ]   k: 5
[INFO ]   neighbors_file: n.csv
[INFO ]   num_projections: 5
[INFO ]   num_tables: 5
[INFO ]   output_model_file: ""
[INFO ]   query_file: queries.csv
[INFO ]   reference_file: refs.csv
[INFO ]   verbose: true
[INFO ]   version: false
[INFO ]
[INFO ] Program timers:
[INFO ]   drusilla_select_construct: 0.000342s
[INFO ]   drusilla_select_search: 0.000780s
[INFO ]   loading_data: 0.010689s
[INFO ]   saving_data: 0.005585s
[INFO ]   total_time: 0.018592s
```

Convenient timers for parts of the program operation are printed. The results, saved in `n.csv` and `d.csv`, indicate the furthest neighbors and distances for each query point. The row of the output file indicates the query point that the results are for. The neighbors are listed from furthest to nearest; so, the 4th element in the 3rd row of `d.csv` indicates the distance between the 3rd query point in `queries.csv` and its approximate 4th furthest neighbor. Similarly, the same element in `n.csv` indicates the index of the approximate 4th furthest neighbor (with respect to `refs.csv`).

18.4.2 Specifying algorithm parameters for DrusillaSelect

The `-p` (`-num_projections`) and `-t` (`-num_tables`) parameters affect the running of the `DrusillaSelect` algorithm and the `QDAFN` algorithm. Specifically, larger values for each of these parameters will search more possible candidate furthest neighbors and produce better results (at the cost of runtime). More details on how each of these parameters works is available in the original papers, the `mlpack` source, or the documentation given by `-help`.

In the example below, we run `DrusillaSelect` to find 4 furthest neighbors using 10 tables and 2 points in each table. In this case we have chosen to omit the `-n n.csv` option, meaning that only the output candidate distances will be written to `d.csv`.

```
$ mlpack_approx_kfn -q queries.csv -r refs.csv -v -k 4 -n n.csv -d d.csv -t 10 -p 2
[INFO ] Loading 'refs.csv' as CSV data. Size is 3 x 1000.
[INFO ] Building DrusillaSelect model...
[INFO ] Model built.
[INFO ] Loading 'queries.csv' as CSV data. Size is 3 x 1000.
[INFO ] Searching for 4 furthest neighbors with DrusillaSelect...
[INFO ] Search complete.
[INFO ] Saving CSV data to 'n.csv'.
[INFO ] Saving CSV data to 'd.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   algorithm: ds
[INFO ]   calculate_error: false
[INFO ]   distances_file: d.csv
[INFO ]   exact_distances_file: ""
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_model_file: ""
[INFO ]   k: 4
[INFO ]   neighbors_file: n.csv
[INFO ]   num_projections: 2
[INFO ]   num_tables: 10
[INFO ]   output_model_file: ""
[INFO ]   query_file: queries.csv
[INFO ]   reference_file: refs.csv
[INFO ]   verbose: true
[INFO ]   version: false
[INFO ]
[INFO ] Program timers:
[INFO ]   drusilla_select_construct: 0.000645s
[INFO ]   drusilla_select_search: 0.000551s
[INFO ]   loading_data: 0.008518s
[INFO ]   saving_data: 0.003734s
[INFO ]   total_time: 0.014019s
```

18.4.3 Using QDAFN instead of DrusillaSelect

The algorithm to be used for approximate furthest neighbor search can be specified with the `-algorithm` (`-a`) option to the `mlpack_approx_kfn` program. Below, we use the `QDAFN` algorithm instead of the default. We leave the `-p` and `-t` options at their defaults—even though `QDAFN` often requires more tables and points to get the same quality of results.

```
$ mlpack_approx_kfn -q queries.csv -r refs.csv -v -k 3 -n n.csv -d d.csv -a qdafn
[INFO ] Loading 'refs.csv' as CSV data. Size is 3 x 1000.
[INFO ] Building QDAFN model...
[INFO ] Model built.
[INFO ] Loading 'queries.csv' as CSV data. Size is 3 x 1000.
[INFO ] Searching for 3 furthest neighbors with QDAFN...
[INFO ] Search complete.
[INFO ] Saving CSV data to 'n.csv'.
[INFO ] Saving CSV data to 'd.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   algorithm: qdafn
[INFO ]   calculate_error: false
```

```
[INFO ] distances_file: d.csv
[INFO ] exact_distances_file: ""
[INFO ] help: false
[INFO ] info: ""
[INFO ] input_model_file: ""
[INFO ] k: 3
[INFO ] neighbors_file: n.csv
[INFO ] num_projections: 5
[INFO ] num_tables: 5
[INFO ] output_model_file: ""
[INFO ] query_file: queries.csv
[INFO ] reference_file: refs.csv
[INFO ] verbose: true
[INFO ] version: false
[INFO ]
[INFO ] Program timers:
[INFO ]   loading_data: 0.008380s
[INFO ]   qdafn_construct: 0.003399s
[INFO ]   qdafn_search: 0.000886s
[INFO ]   saving_data: 0.002253s
[INFO ]   total_time: 0.015465s
```

18.4.4 Printing results quality with exact distances

The `mlpack_approx_kfn` program can calculate the quality of the results if the `-calculate_error` (`-e`) flag is specified. Below we use the program with its default parameters and calculate the error, which is displayed in the output. The error is only calculated for the furthest neighbor, not all `k`; therefore, in this example we have set `-k` to 1.

```
$ mlpack_approx_kfn -q queries.csv -r refs.csv -v -k 1 -e -q -n n.csv
[INFO ] Loading 'refs.csv' as CSV data. Size is 3 x 1000.
[INFO ] Building DrusillaSelect model...
[INFO ] Model built.
[INFO ] Loading 'queries.csv' as CSV data. Size is 3 x 1000.
[INFO ] Searching for 1 furthest neighbors with DrusillaSelect...
[INFO ] Search complete.
[INFO ] Calculating exact distances...
[INFO ] 28891 node combinations were scored.
[INFO ] 37735 base cases were calculated.
[INFO ] Calculation complete.
[INFO ] Average error: 1.08417.
[INFO ] Maximum error: 1.28712.
[INFO ] Minimum error: 1.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   algorithm: ds
[INFO ]   calculate_error: true
[INFO ]   distances_file: ""
[INFO ]   exact_distances_file: ""
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_model_file: ""
[INFO ]   k: 3
[INFO ]   neighbors_file: ""
[INFO ]   num_projections: 5
[INFO ]   num_tables: 5
[INFO ]   output_model_file: ""
[INFO ]   query_file: queries.csv
[INFO ]   reference_file: refs.csv
[INFO ]   verbose: true
[INFO ]   version: false
[INFO ]
[INFO ] Program timers:
[INFO ]   computing_neighbors: 0.001476s
[INFO ]   drusilla_select_construct: 0.000309s
[INFO ]   drusilla_select_search: 0.000495s
[INFO ]   loading_data: 0.008462s
[INFO ]   total_time: 0.011670s
[INFO ]   tree_building: 0.000202s
```

Note that the output includes three lines indicating the error:

```
[INFO ] Average error: 1.08417.
[INFO ] Maximum error: 1.28712.
[INFO ] Minimum error: 1.
```

In this case, a minimum error of 1 indicates an exact result, and over the entire query set the algorithm has returned a furthest neighbor candidate with maximum error 1.28712.

18.4.5 Using cached exact distances for quality results

However, for large datasets, calculating the error may take a long time, because the exact furthest neighbors must be calculated. Therefore, if the exact furthest neighbor distances are already known, they may be passed in with the `-exact_distances_file` (`-x`) option in order to avoid the calculation. In the example below, we assume `exact.csv` contains the exact furthest neighbor distances. We run the `qdafn` algorithm in this example.

Note that the `-e` option must be specified for the `-x` option have any effect.

```
$ mlpack_approx_kfn -q queries.csv -r refs.csv -k 1 -e -x exact.csv -n n.csv -v -a qdafn
[INFO ] Loading 'refs.csv' as CSV data. Size is 3 x 1000.
[INFO ] Building QDAFN model...
[INFO ] Model built.
[INFO ] Loading 'queries.csv' as CSV data. Size is 3 x 1000.
[INFO ] Searching for 1 furthest neighbors with QDAFN...
[INFO ] Search complete.
[INFO ] Loading 'exact.csv' as raw ASCII formatted data. Size is 1 x 1000.
[INFO ] Average error: 1.06914.
[INFO ] Maximum error: 1.67407.
[INFO ] Minimum error: 1.
[INFO ] Saving CSV data to 'n.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   algorithm: qdafn
[INFO ]   calculate_error: true
[INFO ]   distances_file: ""
[INFO ]   exact_distances_file: exact.csv
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_model_file: ""
[INFO ]   k: 1
[INFO ]   neighbors_file: n.csv
[INFO ]   num_projections: 5
[INFO ]   num_tables: 5
[INFO ]   output_model_file: ""
[INFO ]   query_file: queries.csv
[INFO ]   reference_file: refs.csv
[INFO ]   verbose: true
[INFO ]   version: false
[INFO ]
[INFO ] Program timers:
[INFO ]   loading_data: 0.010348s
[INFO ]   qdafn_construct: 0.000318s
[INFO ]   qdafn_search: 0.000793s
[INFO ]   saving_data: 0.000259s
[INFO ]   total_time: 0.012254s
```

18.4.6 Using tree-based approximation with `mlpack_kfn`

The `mlpack_kfn` algorithm allows specifying a desired approximation level with the `-epsilon` (`-e`) option. The parameter must be greater than or equal to 0 and less than 1. A setting of 0 indicates exact search.

The example below runs dual-tree furthest neighbor search (the default algorithm) with the approximation parameter set to 0.5.

```
$ mlpack_kfn -q queries.csv -r refs.csv -v -k 3 -e 0.5 -n n.csv -d d.csv
[INFO ] Loading 'refs.csv' as CSV data. Size is 3 x 1000.
[INFO ] Loaded reference data from 'refs.csv' (3x1000).
[INFO ] Building reference tree...
[INFO ] Tree built.
[INFO ] Loading 'queries.csv' as CSV data. Size is 3 x 1000.
[INFO ] Loaded query data from 'queries.csv' (3x1000).
[INFO ] Searching for 3 neighbors with dual-tree kd-tree search...
[INFO ] 1611 node combinations were scored.
[INFO ] 13938 base cases were calculated.
[INFO ] 1611 node combinations were scored.
[INFO ] 13938 base cases were calculated.
[INFO ] Search complete.
[INFO ] Saving CSV data to 'n.csv'.
[INFO ] Saving CSV data to 'd.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   algorithm: dual_tree
[INFO ]   distances_file: d.csv
[INFO ]   epsilon: 0.5
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_model_file: ""
[INFO ]   k: 3
[INFO ]   leaf_size: 20
[INFO ]   naive: false
[INFO ]   neighbors_file: n.csv
[INFO ]   output_model_file: ""
[INFO ]   percentage: 1
[INFO ]   query_file: queries.csv
[INFO ]   random_basis: false
[INFO ]   reference_file: refs.csv
[INFO ]   seed: 0
[INFO ]   single_mode: false
[INFO ]   tree_type: kd
[INFO ]   true_distances_file: ""
[INFO ]   true_neighbors_file: ""
[INFO ]   verbose: true
[INFO ]   version: false
[INFO ]
[INFO ] Program timers:
[INFO ]   computing_neighbors: 0.000442s
[INFO ]   loading_data: 0.008060s
[INFO ]   saving_data: 0.002850s
[INFO ]   total_time: 0.012667s
[INFO ]   tree_building: 0.000251s
```

Note that the format of the output files `d.csv` and `n.csv` are the same as for `mlpack_approx_kfn`.

18.4.7 Different algorithms with 'mlpack_kfn'

The `mlpack_kfn` program offers a large number of different algorithms that can be used. The `-algorithm (-a)` may be used to specify three main different algorithm types: `naive` (brute-force search), `single_tree` (single-tree search), `dual_tree` (dual-tree search, the default), and `greedy` ("defeatist" greedy search, which goes to one leaf node of the tree then terminates). The example below uses single-tree search to find approximate neighbors with epsilon set to 0.1.

```
mlpack_kfn -q queries.csv -r refs.csv -v -k 3 -e 0.1 -n n.csv -d d.csv -a single_tree
[INFO ] Loading 'refs.csv' as CSV data. Size is 3 x 1000.
[INFO ] Loaded reference data from 'refs.csv' (3x1000).
[INFO ] Building reference tree...
[INFO ] Tree built.
[INFO ] Loading 'queries.csv' as CSV data. Size is 3 x 1000.
[INFO ] Loaded query data from 'queries.csv' (3x1000).
[INFO ] Searching for 3 neighbors with single-tree kd-tree search...
[INFO ] 13240 node combinations were scored.
[INFO ] 15924 base cases were calculated.
[INFO ] Search complete.
[INFO ] Saving CSV data to 'n.csv'.
[INFO ] Saving CSV data to 'd.csv'.
```

```
[INFO ]
[INFO ] Execution parameters:
[INFO ]   algorithm: single_tree
[INFO ]   distances_file: d.csv
[INFO ]   epsilon: 0.1
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_model_file: ""
[INFO ]   k: 3
[INFO ]   leaf_size: 20
[INFO ]   naive: false
[INFO ]   neighbors_file: n.csv
[INFO ]   output_model_file: ""
[INFO ]   percentage: 1
[INFO ]   query_file: queries.csv
[INFO ]   random_basis: false
[INFO ]   reference_file: refs.csv
[INFO ]   seed: 0
[INFO ]   single_mode: false
[INFO ]   tree_type: kd
[INFO ]   true_distances_file: ""
[INFO ]   true_neighbors_file: ""
[INFO ]   verbose: true
[INFO ]   version: false
[INFO ]
[INFO ] Program timers:
[INFO ]   computing_neighbors: 0.000850s
[INFO ]   loading_data: 0.007858s
[INFO ]   saving_data: 0.003445s
[INFO ]   total_time: 0.013084s
[INFO ]   tree_building: 0.000250s
```

18.4.8 Saving a model for later use

The `mlpack_approx_kfn` and `mlpack_kfn` programs both allow models to be saved and loaded for future use. The `-output_model_file` (`-M`) option allows specifying where to save a model, and the `-input_model_file` (`-m`) option allows a model to be loaded instead of trained. So, if you specify `-input_model_file` then you do not need to specify `-reference_file` (`-r`), `-num_projections` (`-p`), or `-num_tables` (`-t`).

The example below saves a model with 10 projections and 5 tables. Note that neither `-query_file` (`-q`) nor `-k` are specified; this run only builds the model and saves it to `model.bin`.

```
$ mlpack_approx_kfn -r refs.csv -t 5 -p 10 -v -M model.bin
[INFO ] Loading 'refs.csv' as CSV data. Size is 3 x 1000.
[INFO ] Building DrusillaSelect model...
[INFO ] Model built.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   algorithm: ds
[INFO ]   calculate_error: false
[INFO ]   distances_file: ""
[INFO ]   exact_distances_file: ""
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_model_file: ""
[INFO ]   k: 0
[INFO ]   neighbors_file: ""
[INFO ]   num_projections: 10
[INFO ]   num_tables: 5
[INFO ]   output_model_file: model.bin
[INFO ]   query_file: ""
[INFO ]   reference_file: refs.csv
[INFO ]   verbose: true
[INFO ]   version: false
[INFO ]
[INFO ] Program timers:
[INFO ]   drusilla_select_construct: 0.000321s
[INFO ]   loading_data: 0.004700s
[INFO ]   total_time: 0.007320s
```

Now, with the model saved, we can run approximate furthest neighbor search on a query set using the saved model:

```
$ mlpack_approx_kfn -m model.bin -q queries.csv -k 3 -d d.csv -n n.csv -v
[INFO ] Loading 'queries.csv' as CSV data. Size is 3 x 1000.
[INFO ] Searching for 3 furthest neighbors with DrusillaSelect...
[INFO ] Search complete.
[INFO ] Saving CSV data to 'n.csv'.
[INFO ] Saving CSV data to 'd.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   algorithm: ds
[INFO ]   calculate_error: false
[INFO ]   distances_file: d.csv
[INFO ]   exact_distances_file: ""
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_model_file: model.bin
[INFO ]   k: 3
[INFO ]   neighbors_file: n.csv
[INFO ]   num_projections: 5
[INFO ]   num_tables: 5
[INFO ]   output_model_file: ""
[INFO ]   query_file: queries.csv
[INFO ]   reference_file: ""
[INFO ]   verbose: true
[INFO ]   version: false
[INFO ]
[INFO ] Program timers:
[INFO ]   drusilla_select_search: 0.000878s
[INFO ]   loading_data: 0.004599s
[INFO ]   saving_data: 0.003006s
[INFO ]   total_time: 0.009234s
```

These options work in the same way for both the `mlpack_approx_kfn` and `mlpack_kfn` programs.

18.4.9 Final command-line program notes

Both the `mlpack_kfn` and `mlpack_approx_kfn` programs contain numerous options not fully documented in these short examples. You can run each program with the `-help` (`-h`) option for more information.

18.5 DrusillaSelect C++ class

mlpack provides a simple `DrusillaSelect` C++ class that can be used inside of C++ programs to perform approximate furthest neighbor search. The class has only one template parameter—`MatType`—which specifies the type of matrix to be used. That means the class can be used with either dense data (of type `arma::mat`) or sparse data (of type `arma::sp_mat`).

The following examples show simple usage of this class.

18.5.1 Approximate furthest neighbors with defaults

The code below builds a `DrusillaSelect` model with default options on the `matrix` dataset, then queries for the approximate furthest neighbor of every point in the `queries` matrix.


```
#include <mlpack/methods/approx_kfn/drusilla_select.hpp>

using namespace mlpack::neighbor;

// The reference dataset.
extern arma::mat dataset;
// The query set.
extern arma::mat queries;

// Construct the model with defaults.
DrusillaSelect<> ds(dataset);

// Query the model, putting output into the following two matrices.
arma::mat distances;
arma::Mat<size_t> neighbors;
ds.Search(queries, 1, neighbors, distances);
```

At the end of this code, both the `distances` and `neighbors` matrices will have number of columns equal to the number of columns in the `queries` matrix. So, each column of the `distances` and `neighbors` matrices are the distances or neighbors of the corresponding column in the `queries` matrix.

18.5.2 Custom numbers of tables and projections

The following example constructs a `DrusillaSelect` model with 10 tables and 5 projections. Once that is done it performs the same task as the previous example.

```
#include <mlpack/methods/approx_kfn/drusilla_select.hpp>

using namespace mlpack::neighbor;

// The reference dataset.
extern arma::mat dataset;
// The query set.
extern arma::mat queries;

// Construct the model with custom parameters.
DrusillaSelect<> ds(dataset, 10, 5);

// Query the model, putting output into the following two matrices.
arma::mat distances;
arma::Mat<size_t> neighbors;
ds.Search(queries, 1, neighbors, distances);
```

18.5.3 Accessing the candidate set

The `DrusillaSelect` algorithm merely scans the reference set and extracts a number of points that will be queried in a brute-force fashion when the `Search()` method is called. We can access this set with the `CandidateSet()` method. The code below prints the fifth point of the candidate set.

```
#include <mlpack/methods/approx_kfn/drusilla_select.hpp>

using namespace mlpack::neighbor;

// The reference dataset.
extern arma::mat dataset;

// Construct the model with custom parameters.
DrusillaSelect<> ds(dataset, 10, 5);

// Print the fifth point of the candidate set.
std::cout << ds.CandidateSet().col(4).t();
```

18.5.4 Retraining on a new reference set

It is possible to retrain a `DrusillaSelect` model with new parameters or with a new reference set. This is functionally equivalent to creating a new model. The example code below creates a first `DrusillaSelect` model using 3 tables and 10 projections, and then retrains this with the same reference set using 10 tables and 3 projections.

```
#include <mlpack/methods/approx_kfn/drusilla_select.hpp>

using namespace mlpack::neighbor;

// The reference dataset.
extern arma::mat dataset;

// Construct the model with initial parameters.
DrusillaSelect<> ds(dataset, 3, 10);

// Now retrain with different parameters.
ds.Train(dataset, 10, 3);
```

18.5.5 Running on sparse data

We can set the template parameter for `DrusillaSelect` to `arma::sp_mat` in order to perform furthest neighbor search on sparse data. This code below creates a `DrusillaSelect` model using 4 tables and 6 projections with sparse input data, then searches for 3 approximate furthest neighbors.

```
#include <mlpack/methods/approx_kfn/drusilla_select.hpp>

using namespace mlpack::neighbor;

// The reference dataset.
extern arma::sp_mat dataset;
// The query dataset.
extern arma::sp_mat querySet;

// Construct the model on sparse data.
DrusillaSelect<arma::sp_mat> ds(dataset, 4, 6);

// Search on query data.
arma::Mat<size_t> neighbors;
arma::mat distances;
ds.Search(querySet, 3, neighbors, distances);
```

18.6 QDAFN C++ class

mlpack also provides a standalone simple `QDAFN` class for furthest neighbor search. The API for this class is virtually identical to the `DrusillaSelect` class, and also has one template parameter to specify the type of matrix to be used (dense or sparse or other).

The following subsections demonstrate usage of the `QDAFN` class in the same way as the previous section's examples for `DrusillaSelect`.

18.6.1 Approximate furthest neighbors with defaults

The code below builds a QDAFN model with default options on the matrix dataset, then queries for the approximate furthest neighbor of every point in the queries matrix.

```
#include <mlpack/methods/approx_kfn/qdafn.hpp>

using namespace mlpack::neighbor;

// The reference dataset.
extern arma::mat dataset;
// The query set.
extern arma::mat queries;

// Construct the model with defaults.
QDAFN<> qd(dataset);

// Query the model, putting output into the following two matrices.
arma::mat distances;
arma::Mat<size_t> neighbors;
qd.Search(queries, 1, neighbors, distances);
```

At the end of this code, both the distances and neighbors matrices will have number of columns equal to the number of columns in the queries matrix. So, each column of the distances and neighbors matrices are the distances or neighbors of the corresponding column in the queries matrix.

18.6.2 Custom numbers of tables and projections

The following example constructs a QDAFN model with 15 tables and 30 projections. Once that is done it performs the same task as the previous example.

```
#include <mlpack/methods/approx_kfn/qdafn.hpp>

using namespace mlpack::neighbor;

// The reference dataset.
extern arma::mat dataset;
// The query set.
extern arma::mat queries;

// Construct the model with custom parameters.
QDAFN<> qdafn(dataset, 15, 30);

// Query the model, putting output into the following two matrices.
arma::mat distances;
arma::Mat<size_t> neighbors;
qdafn.Search(queries, 1, neighbors, distances);
```

18.6.3 Accessing the candidate set

The QDAFN algorithm scans the reference set, extracting points that have been projected onto random directions. Each random direction corresponds to a single table. The QDAFN class stores these points as a vector of matrices, which can be accessed with the CandidateSet() method. The code below prints the fifth point of the candidate set of the third table.

```
#include <mlpack/methods/approx_kfn/qdafn.hpp>

using namespace mlpack::neighbor;

// The reference dataset.
extern arma::mat dataset;

// Construct the model with custom parameters.
QDAFN<> qdafn(dataset, 10, 5);

// Print the fifth point of the candidate set.
std::cout << ds.CandidateSet(2).col(4).t();
```

18.6.4 Retraining on a new reference set

It is possible to retrain a QDAFN model with new parameters or with a new reference set. This is functionally equivalent to creating a new model. The example code below creates a first QDAFN model using 10 tables and 40 projections, and then retrains this with the same reference set using 15 tables and 25 projections.

```
#include <mlpack/methods/approx_kfn/qdafn.hpp>

using namespace mlpack::neighbor;

// The reference dataset.
extern arma::mat dataset;

// Construct the model with initial parameters.
QDAFN<> qdafn(dataset, 3, 10);

// Now retrain with different parameters.
qdafn.Train(dataset, 10, 3);
```

18.6.5 Running on sparse data

We can set the template parameter for QDAFN to `arma::sp_mat` in order to perform furthest neighbor search on sparse data. This code below creates a QDAFN model using 20 tables and 60 projections with sparse input data, then searches for 3 approximate furthest neighbors.

```
#include <mlpack/methods/approx_kfn/qdafn.hpp>

using namespace mlpack::neighbor;

// The reference dataset.
extern arma::sp_mat dataset;
// The query dataset.
extern arma::sp_mat querySet;

// Construct the model on sparse data.
QDAFN<arma::sp_mat> qdafn(dataset, 20, 60);

// Search on query data.
arma::Mat<size_t> neighbors;
arma::mat distances;
qdafn.Search(querySet, 3, neighbors, distances);
```

18.7 KFN C++ class

The extensive `NeighborSearch` class also provides a way to search for approximate furthest neighbors using a different, tree-based technique. For full documentation on this class, see the [NeighborSearch tutorial](#) (p. 151). The KFN class is a convenient typedef of the `NeighborSearch` class that can be used to perform the furthest neighbors task with kd-trees.

In the following subsections, the KFN class is used in short code examples.

18.7.1 Simple furthest neighbors example

The KFN class has construction semantics similar to `DrusillaSelect` and `QDAFN`. The example below constructs a KFN object (which will build the tree on the reference set), but note that the third parameter to the constructor allows us to specify our desired level of approximation. In this example we choose $\epsilon = 0.05$. Then, the code searches for 3 approximate furthest neighbors.

```
#include <mlpack/methods/neighbor_search/neighbor_search.hpp>

using namespace mlpack::neighbor;

// The reference dataset.
extern arma::mat dataset;
// The query set.
extern arma::mat querySet;

// Construct the object, performing the default dual-tree search with
// approximation level epsilon = 0.05.
KFN kfn(dataset, KFN::DUAL_TREE_MODE, 0.05);

// Search for approximate furthest neighbors.
arma::Mat<size_t> neighbors;
arma::mat distances;
kfn.Search(querySet, 3, neighbors, distances);
```

18.7.2 Retraining on a new reference set

Like the `QDAFN` and `DrusillaSelect` classes, the KFN class is capable of retraining on a new reference set. The code below demonstrates this.

```
#include <mlpack/methods/neighbor_search/neighbor_search.hpp>

using namespace mlpack::neighbor;

// The original reference set we train on.
extern arma::mat dataset;
// The new reference set we retrain on.
extern arma::mat newDataset;

// Construct the object with approximation level 0.1.
KFN kfn(dataset, DUAL_TREE_MODE, 0.1);

// Retrain on the new reference set.
kfn.Train(newDataset);
```

18.7.3 Searching in single-tree mode

The particular mode to be used in search can be specified in the constructor. In this example, we use single-tree search (as opposed to the default of dual-tree search).

```
#include <mlpack/methods/neighbor_search/neighbor_search.hpp>

using namespace mlpack::neighbor;

// The reference set.
extern arma::mat dataset;
// The query set.
extern arma::mat querySet;

// Construct the object with approximation level 0.25 and in single tree search
// mode.
KFN kfn(dataset, SINGLE_TREE_MODE, 0.25);

// Search for 5 approximate furthest neighbors.
arma::Mat<size_t> neighbors;
arma::mat distances;
kfn.Search(querySet, 5, neighbors, distances);
```

18.7.4 Searching in brute-force mode

If desired, brute-force search ("naive search") can be used to find the furthest neighbors; however, the result will not be approximate—it will be exact (since every possibility will be considered). The code below performs exact furthest neighbor search by using the `KFN` class in brute-force mode.

```
#include <mlpack/methods/neighbor_search/neighbor_search.hpp>

using namespace mlpack::neighbor;

// The reference set.
extern arma::mat dataset;
// The query set.
extern arma::mat querySet;

// Construct the object in brute-force mode. We can leave the approximation
// parameter to its default (0) since brute-force will provide exact results.
KFN kfn(dataset, NAIVE_MODE);

// Perform the search for 2 furthest neighbors.
arma::Mat<size_t> neighbors;
arma::mat distances;
kfn.Search(querySet, 2, neighbors, distances);
```

18.8 Further documentation

For further documentation on the approximate furthest neighbor facilities offered by **mlpack**, consult the following documentation:

- **NeighborSearch tutorial (k-nearest-neighbors)** (p. 151)
- **QDAFN class documentation** (p. 1666)
- **DrusillaSelect class documentation** (p. 1597)
- **NeighborSearch class documentation** (p. 1627)

Chapter 19

Collaborative filtering tutorial

19.1 Introduction

Collaborative filtering is an increasingly popular approach for recommender systems. A typical formulation of the problem is as follows: there are n users and m items, and each user has rated some of the items. We want to provide each user with a recommendation for an item they have not rated yet, which they are likely to rate highly. In another formulation, we may want to predict a user's rating of an item. This type of problem has been considered extensively, especially in the context of the Netflix prize. The winning approach for the Netflix prize was a collaborative filtering approach which utilized matrix decomposition. More information on their approach can be found in the following paper:

```
@article{koren2009matrix,
  title={Matrix factorization techniques for recommender systems},
  author={Koren, Yehuda and Bell, Robert and Volinsky, Chris},
  journal={Computer},
  number={8},
  pages={30--37},
  year={2009},
  publisher={IEEE}
}
```

The key to this approach is that the data is represented as an incomplete matrix $V \in \Re^{n \times m}$, where V_{ij} represents user i 's rating of item j , if that rating exists. The task, then, is to complete the entries of the matrix.

In the matrix factorization framework, the matrix V is assumed to be low-rank and decomposed into components as $V \approx WH$ according to some heuristic.

In order to solve problems of this form, **mlpack** provides:

- a **simple command-line interface** (p. 104) to perform collaborative filtering
- a **simple C++ interface** (p. 107) to perform collaborative filtering
- an **extensible C++ interface** (p. 109) for implementing new collaborative filtering techniques

19.2 Table of Contents

- **Introduction** (p. 103)
- **Table of Contents** (p. 104)
- **The 'mlpack_cf' program** (p. 104)
 - **Input format for mlpack_cf** (p. 105)
 - **mlpack_cf with default parameters** (p. 105)
 - **Saving mlpack_cf models** (p. 106)
 - **Loading mlpack_cf models** (p. 106)
 - **Specifying rank of mlpack_cf decomposition** (p. 106)
 - **mlpack_cf with single-user recommendation** (p. 106)
 - **mlpack_cf with non-default factorizer** (p. 107)
 - **mlpack_cf with non-default neighborhood size** (p. 107)
- **The 'CF' class** (p. 107)
 - **CF with default parameters** (p. 108)
 - **CF with other factorizers** (p. 108)
 - **Predicting individual user/item ratings** (p. 109)
 - **Other operations with the W and H matrices** (p. 109)
- **Template parameters for the 'CF' class** (p. 109)
- **Further documentation** (p. 110)

19.3 The 'mlpack_cf' program

mlpack provides a command-line program, `mlpack_cf`, which is used to perform collaborative filtering on a given dataset. It can provide neighborhood-based recommendations for users. The algorithm used for matrix factorization is configurable, and the parameters of each algorithm are also configurable.

The following examples detail usage of the `mlpack_cf` program. Note that you can get documentation on all the possible parameters by typing:

```
$ mlpack_cf --help
```


19.3.1 Input format for mlpack_cf

The input file for the `mlpack_cf` program is specified with the `-training_file` or `-t` option. This file is a coordinate-format sparse matrix, similar to the Matrix Market (MM) format. The first coordinate is the user id; the second coordinate is the item id; and the third coordinate is the rating. So, for instance, a dataset with 3 users and 2 items, and ratings between 1 and 5, might look like the following:

```
$ cat dataset.csv
0, 1, 4
1, 0, 5
1, 1, 1
2, 0, 2
```

This dataset has four ratings: user 0 has rated item 1 with a rating of 4; user 1 has rated item 0 with a rating of 5; user 1 has rated item 1 with a rating of 1; and user 2 has rated item 0 with a rating of 2. Note that the user and item indices start from 0, and the identifiers must be numeric indices, and not names.

The type does not necessarily need to be a csv; it can be any supported storage format, assuming that it is a coordinate-format file in the format specified above. For more information on mlpack file formats, see the documentation for `mlpack::data::Load()` (p. 379).

19.3.2 mlpack_cf with default parameters

In this example, we have a dataset from MovieLens, and we want to use `mlpack_cf` with the default parameters, which will provide 5 recommendations for each user, and we wish to save the results in the file `recommendations.csv`. Assuming that our dataset is in the file `MovieLens-100k.csv` and it is in the correct format, we may use the `mlpack_cf` executable as below:

```
$ mlpack_cf -t MovieLens-100k.csv -v -o recommendations.csv
```

The `-v` option provides verbose output, and may be omitted if desired. Now, for each user, we have recommendations in `recommendations.csv`:

```
$ head recommendations.csv
317,422,482,356,495
116,120,180,6,327
312,49,116,99,236
312,116,99,236,285
55,190,317,194,63
171,209,180,175,95
208,0,94,87,57
99,97,0,203,172
257,99,180,287,0
171,203,172,209,88
```

So, for user 0, the top 5 recommended items that user 0 has not rated are items 317, 422, 482, 356, and 495. For user 5, the recommendations are on the sixth line: 171, 209, 180, 175, 95.

The `mlpack_cf` program can be built into a larger recommendation framework, with a preprocessing step that can turn user information and item information into numeric IDs, and a postprocessing step that can map these numeric IDs back to the original information.

19.3.3 Saving `mlpack_cf` models

The `mlpack_cf` program is able to save a particular model for later loading. Saving a model can be done with the `-output_model_file` or `-M` option. The example below builds a CF model on the `MovieLens-100k.csv` dataset, and then saves the model to the file `cf-model.xml` for later usage.

```
$ mlpack_cf -t MovieLens-100k.csv -M cf-model.xml -v
```

The models can also be saved as `.bin` or `.txt`; the `.xml` format provides a human-inspectable format (though the models tend to be quite complex and may be difficult to read). These models can then be re-used to provide specific recommendations for certain users, or other tasks.

19.3.4 Loading `mlpack_cf` models

Instead of training a model, the `mlpack_cf` model can also load a model to provide recommendations, using the `-input_model_file` or `-m` option. For instance, the example below will load the model from `cf-model.xml` and then generate 3 recommendations for each user in the dataset, saving the results to `recommendations.csv`.

```
$ mlpack_cf -m cf-model.xml -v -o recommendations.csv
```

19.3.5 Specifying rank of `mlpack_cf` decomposition

By default, the matrix factorizations in the `mlpack_cf` program decompose the data matrix into two matrices W and H with rank two. Often, this default parameter is not correct, and it makes sense to use a higher-rank decomposition. The rank can be specified with the `-rank` or `-R` parameter:

```
$ mlpack_cf -t MovieLens-100k.csv -R 10 -v
```

In the example above, the data matrix will be decomposed into two matrices of rank 10. In general, higher-rank decompositions will take longer, but will give more accurate predictions.

19.3.6 `mlpack_cf` with single-user recommendation

In the previous two examples, the output file `recommendations.csv` contains one line for each user in the input dataset. But often, recommendations may only be desired for a few users. In that case, we can assemble a file of query users, with one user per line:

```
$ cat query.csv
0
17
31
```

Now, if we run the `mlpack_cf` executable with this query file, we will obtain recommendations for users 0, 17, and 31:

```
$ mlpack_cf -i MovieLens-100k.csv -R 10 -q query.csv -o recommendations.csv
$ cat recommendations.csv
474,356,317,432,473
510,172,204,483,182
0,120,236,257,126
```

19.3.7 `mlpack_cf` with non-default factorizer

The `-algorithm` (or `-a`) parameter controls the factorizer that is used. Several options are available:

- `'NMF'`: non-negative matrix factorization; see `mlpack::amf::AMF<>`
- `'SVDBatch'`: SVD batch factorization
- `'SVDIncompleteIncremental'`: incomplete incremental SVD
- `'SVDCompleteIncremental'`: complete incremental SVD
- `'RegSVD'`: regularized SVD; see `mlpack::svd::RegularizedSVD` (p. 1943)

The default factorizer is `'NMF'`. The example below uses the `'RegSVD'` factorizer:

```
$ mlpack_cf -i MovieLens-100k.csv -R 10 -q query.csv -a RegSVD -o recommendations.csv
```

19.3.8 `mlpack_cf` with non-default neighborhood size

The `mlpack_cf` program produces recommendations using a neighborhood: similar users in the query user's neighborhood will be averaged to produce predictions. The size of this neighborhood is controlled with the `-neighborhood` (or `-n`) option. An example using a neighborhood with 10 similar users is below:

```
$ mlpack_cf -i MovieLens-100k.csv -R 10 -q query.csv -a RegSVD -n 10
```

19.4 The 'CF' class

The `CF` class in **mlpack** offers a simple, flexible API for performing collaborative filtering for recommender systems within C++ applications. In the constructor, the `CF` class takes a coordinate-list dataset and decomposes the matrix according to the specified `FactorizerType` template parameter.

Then, the `GetRecommendations()` function may be called to obtain recommendations for certain users (or all users), and the `W()` and `H()` matrices may be accessed to perform other computations.

The data which the `CF` constructor takes should be an Armadillo matrix (`arma::mat`) with three rows. The first row corresponds to users; the second row corresponds to items; the third column corresponds to the rating. This is a coordinate list format, like the format the `cf` executable takes. The `data::Load()` (p. 379) function can be used to load data.

The following examples detail a few ways that the `CF` class can be used.

19.4.1 CF with default parameters

This example constructs the CF object with default parameters and obtains recommendations for each user, storing the output in the `recommendations` matrix.

```
#include <mlpack/methods/cf/cf.hpp>

using namespace mlpack::cf;

// The coordinate list of ratings that we have.
extern arma::mat data;
// The size of the neighborhood to use to get recommendations.
extern size_t neighborhood;
// The rank of the decomposition.
extern size_t rank;

// Build the CF object and perform the decomposition.
// The constructor takes a default-constructed factorizer, which, by default,
// is of type amf::NMFALSFactorizer.
CF cf(data, amf::NMFALSFactorizer(), neighborhood, rank);

// Store the results in this object.
arma::Mat<size_t> recommendations;

// Get 5 recommendations for all users.
cf.GetRecommendations(5, recommendations);
```

19.4.2 CF with other factorizers

mlpack provides a number of existing factorizers which can be used in place of the default `mlpack::amf::NMFA↔LSFactorizer` (p. 259) (which is non-negative matrix factorization with alternating least squares update rules). These include:

- `mlpack::amf::SVDBatchFactorizer` (p. 259)
- `mlpack::amf::SVDCompleteIncrementalFactorizer` (p. 260)
- `mlpack::amf::SVDIncompleteIncrementalFactorizer` (p. 260)
- `mlpack::amf::NMFALSFactorizer` (p. 259)
- `mlpack::svd::RegularizedSVD` (p. 1943)
- `mlpack::svd::QUIC_SVD` (p. 1932)

The `amf::AMF<>` class has many other possibilities than those listed here; it is a framework for alternating matrix factorization techniques. See the class documentation or **tutorial on AMF** (p. 73) for more information.

The use of another factorizer is straightforward; the example from the previous section is adapted below to use `svd::↔RegularizedSVD`:

```
#include <mlpack/methods/cf/cf.hpp>
#include <mlpack/methods/regularized_svd/regularized_svd.hpp>

using namespace mlpack::cf;

// The coordinate list of ratings that we have.
extern arma::mat data;
// The size of the neighborhood to use to get recommendations.
extern size_t neighborhood;
// The rank of the decomposition.
extern size_t rank;

// Build the CF object and perform the decomposition.
CF cf(data, svd::RegularizedSVD(), neighborhood, rank);

// Store the results in this object.
arma::Mat<size_t> recommendations;

// Get 5 recommendations for all users.
cf.GetRecommendations(5, recommendations);
```

19.4.3 Predicting individual user/item ratings

The `Predict()` method can be used to predict the rating of an item by a certain user, using the same neighborhood-based approach as the `GetRecommendations()` function or the `cf` executable. Below is an example of the use of that function.

The example below will obtain the predicted rating for item 50 by user 12.

```
#include <mlpack/methods/cf/cf.hpp>

using namespace mlpack::cf;

// The coordinate list of ratings that we have.
extern arma::mat data;
// The size of the neighborhood to use to get recommendations.
extern size_t neighborhood;
// The rank of the decomposition.
extern size_t rank;

// Build the CF object and perform the decomposition.
// The constructor takes a default-constructed factorizer, which, by default,
// is of type amf::NMFALSFactorizer.
CF cf(data, amf::NMFALSFactorizer(), neighborhood, rank);

const double prediction = cf.Predict(12, 50); // User 12, item 50.
```

19.4.4 Other operations with the W and H matrices

Sometimes, the raw decomposed W and H matrices can be useful. The example below obtains these matrices, and multiplies them against each other to obtain a reconstructed data matrix with no missing values.

```
#include <mlpack/methods/cf/cf.hpp>

using namespace mlpack::cf;

// The coordinate list of ratings that we have.
extern arma::mat data;
// The size of the neighborhood to use to get recommendations.
extern size_t neighborhood;
// The rank of the decomposition.
extern size_t rank;

// Build the CF object and perform the decomposition.
// The constructor takes a default-constructed factorizer, which, by default,
// is of type amf::NMFALSFactorizer.
CF cf(data, amf::NMFALSFactorizer(), neighborhood, rank);

// References to W and H matrices.
const arma::mat& W = cf.W();
const arma::mat& H = cf.H();

// Multiply the matrices together.
arma::mat reconstructed = W * H;
```

19.5 Template parameters for the 'CF' class

The CF class takes the `FactorizerType` as a template parameter to some of its constructors and to the `Train()` function. The `FactorizerType` class defines the algorithm used for matrix factorization. There are a number of existing factorizers that can be used in **mlpack**; these were detailed in the **'other factorizers' example** (p. 108) of the previous section.

The `FactorizerType` class must implement one of the two following methods:

- `Apply(arma::mat& data, const size_t rank, arma::mat& W, arma::mat& H);`
- `Apply(arma::sp_mat& data, const size_t rank, arma::mat& W, arma::mat& H);`

The difference between these two methods is whether `arma::mat` or `arma::sp_mat` is used as input. If `arma::mat` is used, then the data matrix is a coordinate list with three columns, as in the constructor to the `CF` class. If `arma::sp_mat` is used, then a sparse matrix is passed with the number of rows equal to the number of items and the number of columns equal to the number of users, and each nonzero element in the matrix corresponds to a non-missing rating.

The method that the factorizer implements is specified via the `FactorizerTraits` class, which is a template metaprogramming traits class:

```
template<typename FactorizerType>
struct FactorizerTraits
{
    static const bool UsesCoordinateList = false;
};
```

If `FactorizerTraits<MyFactorizer>::UsesCoordinateList` is `true`, then `CF` will try to call `Apply()` with an `arma::mat` object. Otherwise, `CF` will try to call `Apply()` with an `arma::sp_mat` object. Specifying the value of `UsesCoordinateList` is straightforward; provide this specialization of the `FactorizerTraits` class:

```
template<>
struct FactorizerTraits<MyFactorizer>
{
    static const bool UsesCoordinateList = true; // Set your value here.
};
```

The `Apply()` function also takes a reference to the matrices `W` and `H`. When the `Apply()` function returns, the input data matrix should be decomposed into these two matrices. `W` should have number of rows equal to the number of items and number of columns equal to the `rank` parameter, and `H` should have number of rows equal to the `rank` parameter, and number of columns equal to the number of users.

The `amf::AMF<>` class (p. 499) can be used as a base for factorizers that alternate between updating `W` and updating `H`. A useful reference is the **AMF tutorial** (p. 73).

19.6 Further documentation

Further documentation for the `CF` class may be found in the complete API documentation. In addition, more information on the `AMF` class of factorizers may be found in its **complete API documentation** (p. 499).

Chapter 20

Density Estimation Tree (DET) tutorial

20.1 Introduction

DETs perform the unsupervised task of density estimation using decision trees. Using a trained density estimation tree (DET), the density at any particular point can be estimated very quickly ($O(\log n)$ time, where n is the number of points the tree is built on).

The details of this work is presented in the following paper:

```
@inproceedings{ram2011density,
  title={Density estimation trees},
  author={Ram, P. and Gray, A.G.},
  booktitle={Proceedings of the 17th ACM SIGKDD International Conference on
    Knowledge Discovery and Data Mining},
  pages={627--635},
  year={2011},
  organization={ACM}
}
```

mlpack provides:

- a **simple command-line executable** (p. 112) to perform density estimation and related analyses using DETs
- a **generic C++ class (DTree)** (p. 114) which provides various functionality for the DETs
- a set of functions in the namespace **mlpack::det** (p. 115) to perform cross-validation for the task of density estimation with DETs

20.2 Table of Contents

A list of all the sections this tutorial contains.

- **Introduction** (p. 111)
- **Table of Contents** (p. 111)

- **Command-Line `mlpack_det`** (p. 112)
 - **Plain-vanilla density estimation** (p. 113)
 - **Estimation on a test set** (p. 113)
 - **Computing the variable importance** (p. 114)
 - **Saving trained DETs** (p. 114)
 - **Loading trained DETs** (p. 114)
- **The 'DTree' class** (p. 114)
 - **Public Functions** (p. 115)
- **'namespace `mlpack::det`'** (p. 115)
 - **Utility Functions** (p. 116)
- **Further Documentation** (p. 116)

20.3 Command-Line `mlpack_det`

The command line arguments of this program can be viewed using the `-h` option:

```
$ mlpack_det -h
Density Estimation With Density Estimation Trees

This program performs a number of functions related to Density Estimation
Trees. The optimal Density Estimation Tree (DET) can be trained on a set of
data (specified by --training_file or -t) using cross-validation (with number
of folds specified by --folds). This trained density estimation tree may then
be saved to a model file with the --output_model_file (-M) option.

The variable importances of each dimension may be saved with the --vi_file
(-i) option, and the density estimates on each training point may be saved to
the file specified with the --training_set_estimates_file (-e) option.

This program also can provide density estimates for a set of test points,
specified in the --test_file (-T) file. The density estimation tree used for
this task will be the tree that was trained on the given training points, or a
tree stored in the file given with the --input_model_file (-m) parameter. The
density estimates for the test points may be saved into the file specified
with the --test_set_estimates_file (-E) option.
```

Options:

```
--folds (-f) [int]          The number of folds of cross-validation to
                             perform for the estimation (0 is LOOCV) Default
                             value 10.
--help (-h)                Default help info.
--info [string]             Get help on a specific module or option.
                             Default value ''.
--input_model_file (-m) [string]
                             File containing already trained density
                             estimation tree. Default value ''.
--max_leaf_size (-L) [int]  The maximum size of a leaf in the unpruned,
                             fully grown DET. Default value 10.
--min_leaf_size (-l) [int]  The minimum size of a leaf in the unpruned,
                             fully grown DET. Default value 5.
--output_model_file (-M) [string]
                             File to save trained density estimation tree to.
                             Default value ''.
--test_file (-T) [string]   A set of test points to estimate the density of.
                             Default value ''.
--test_set_estimates_file (-E) [string]
                             The file in which to output the estimates on the
                             test set from the final optimally pruned tree.
                             Default value ''.
```



```

--training_file (-t) [string]
    The data set on which to build a density
    estimation tree. Default value ''.
--training_set_estimates_file (-e) [string]
    The file in which to output the density
    estimates on the training set from the final
    optimally pruned tree. Default value ''.
--verbose (-v)
    Display informational messages and the full list
    of parameters and timers at the end of
    execution.
--version (-V)
    Display the version of mlpack.
--vi_file (-i) [string]
    The file to output the variable importance
    values for each feature. Default value ''.

```

For further information, including relevant papers, citations, and theory, consult the documentation found at <http://www.mlpack.org> or included with your distribution of **mlpack**.

20.3.1 Plain-vanilla density estimation

We can just train a DET on the provided data set *S*. Like all datasets **mlpack** uses, the data should be row-major (**mlpack** transposes data when it is loaded; internally, the data is column-major – see [this page](#) (p. 57) for more information).

```
$ mlpack_det -t dataset.csv -v
```

By default, `mlpack_det` performs 10-fold cross-validation (using the α -pruning regularization for decision trees). To perform LOOCV (leave-one-out cross-validation), which can provide better results but will take longer, use the following command:

```
$ mlpack_det -t dataset.csv -f 0 -v
```

To perform k-fold crossvalidation, use `-f k` (or `-folds k`). There are certain other options available for training. For example, in the construction of the initial tree, you can specify the maximum and minimum leaf sizes. By default, they are 10 and 5 respectively; you can set them using the `-M` (`-max_leaf_size`) and the `-N` (`-min_leaf_size`) options.

```
$ mlpack_det -t dataset.csv -M 20 -N 10
```

In case you want to output the density estimates at the points in the training set, use the `-e` (`-training_set_↔estimates_file`) option to specify the output file to which the estimates will be saved. The first line in `density_↔estimates.txt` will correspond to the density at the first point in the training set. Note that the logarithm of the density estimates are given, which allows smaller estimates to be saved.

```
$ mlpack_det -t dataset.csv -e density_estimates.txt -v
```

20.3.2 Estimation on a test set

Often, it is useful to train a density estimation tree on a training set and then obtain density estimates from the learned estimator for a separate set of test points. The `-T` (`-test_file`) option allows specification of a set of test points, and the `-E` (`-test_set_estimates_file`) option allows specification of the file into which the test set estimates are saved. Note that the logarithm of the density estimates are saved; this allows smaller values to be saved.

```
$ mlpack_det -t dataset.csv -T test_points.csv -E test_density_estimates.txt -v
```

20.3.3 Computing the variable importance

The variable importance (with respect to density estimation) of the different features in the data set can be obtained by using the `-i (-vi_file)` option. This outputs the absolute (as opposed to relative) variable importance of the all the features into the specified file.

```
$ mlpack_det -t dataset.csv -i variable_importance.txt -v
```

20.3.4 Saving trained DETs

The `mlpack_det` program is capable of saving a trained DET to a file for later usage. The `-output_model_file` or `-M` option allows specification of the file to save to. In the example below, a DET trained on `dataset.csv` is saved to the file `det.xml`.

```
$ mlpack_det -t dataset.csv -M det.xml -v
```

20.3.5 Loading trained DETs

A saved DET can be used to perform any of the functionality in the examples above. A saved DET is loaded with the `-input_model_file` or `-m` option. The example below loads a saved DET from `det.xml` and outputs density estimates on the dataset `test_dataset.csv` into the file `estimates.csv`.

```
$ mlpack_det -m det.xml -T test_dataset.csv -E estimates.csv -v
```

20.4 The 'DTree' class

This class implements density estimation trees. Below is a simple example which initializes a density estimation tree.

```
#include <mlpack/methods/det/dtree.hpp>

using namespace mlpack::det;

// The dataset matrix, on which to learn the density estimation tree.
extern arma::Mat<float> data;

// Initialize the tree. This function also creates and saves the bounding box
// of the data. Note that it does not actually build the tree.
DTree<> det(data);
```

20.4.1 Public Functions

The function `Grow()` greedily grows the tree, adding new points to the tree. Note that the points in the dataset will be reordered. This should only be run on a tree which has not already been built. In general, it is more useful to use the **`Trainer()`** (p. 392) function found in 'namespace mlpack::det' (p. 115).

```
// This keeps track of the data during the shuffle that occurs while growing the
// tree.
arma::Col<size_t> oldFromNew(data.n_cols);
for (size_t i = 0; i < data.n_cols; i++)
  oldFromNew[i] = i;

// This function grows the tree down to the leaves. It returns the current
// minimum value of the regularization parameter alpha.
size_t maxLeafSize = 10;
size_t minLeafSize = 5;

double alpha = det.Grow(data, oldFromNew, false, maxLeafSize, minLeafSize);
```

Note that the alternate volume regularization should not be used (see ticket #238).

To estimate the density at a given query point, use the following code. Note that the logarithm of the density is returned.

```
// For a given query, you can obtain the density estimate.
extern arma::Col<float> query;
extern DTree* det;
double estimate = det->ComputeValue(&query);
```

Computing the **variable importance** of each feature for the given DET.

```
// The data matrix and density estimation tree.
extern arma::mat data;
extern DTree* det;

// The variable importances will be saved into this vector.
arma::Col<double> varImps;

// You can obtain the variable importance from the current tree.
det->ComputeVariableImportance(varImps);
```

20.5 'namespace mlpack::det'

The functions in this namespace allows the user to perform tasks with the 'DTree' class. Most importantly, the **`Trainer()`** (p. 392) method allows the full training of a density estimation tree with cross-validation. There are also utility functions which allow printing of leaf membership and variable importance.

20.5.1 Utility Functions

The code below details how to train a density estimation tree with cross-validation.

```
#include <mlpack/methods/det/dt_utils.hpp>

using namespace mlpack::det;

// The dataset matrix, on which to learn the density estimation tree.
extern arma::Mat<float> data;

// The number of folds for cross-validation.
const size_t folds = 10; // Set folds = 0 for LOOCV.

const size_t maxLeafSize = 10;
const size_t minLeafSize = 5;

// Train the density estimation tree with cross-validation.
DTree<>* dtree_opt = Trainer(data, folds, false, maxLeafSize, minLeafSize);
```

Note that the alternate volume regularization should be set to false because it has known bugs (see #238).

To print the class membership of leaves in the tree into a file, see the following code.

```
extern arma::Mat<size_t> labels;
extern DTree* det;
const size_t numClasses = 3; // The number of classes must be known.

extern string leafClassMembershipFile;

PrintLeafMembership(det, data, labels, numClasses, leafClassMembershipFile);
```

Note that you can find the number of classes with `max(labels) + 1`. The variable importance can also be printed to a file in a similar manner.

```
extern DTree* det;

extern string variableImportanceFile;
const size_t numFeatures = data.n_rows;

PrintVariableImportance(det, numFeatures, variableImportanceFile);
```

20.6 Further Documentation

For further documentation on the DTree class, consult the **complete API documentation** (p. 1207).

Chapter 21

EMST Tutorial

21.1 Introduction

The Euclidean Minimum Spanning Tree problem is widely used in machine learning and data mining applications. Given a set S of points in \mathbf{R}^d , our task is to compute lowest weight spanning tree in the complete graph on S with edge weights given by the Euclidean distance between points.

Among other applications, the EMST can be used to compute hierarchical clusterings of data. A *single-linkage clustering* can be obtained from the EMST by deleting all edges longer than a given cluster length. This technique is also referred to as a *Friends-of-Friends* clustering in the astronomy literature.

mlpack includes an implementation of **Dual-Tree Boruvka** which uses *kd*-trees by default; this is the empirically and theoretically fastest EMST algorithm. In addition, the implementation supports the use of different trees via templates. For more details, see the following paper:

```
@inproceedings{march2010fast,
  title={Fast {E}uclidean minimum spanning tree: algorithm, analysis, and
  applications},
  author={March, William B. and Ram, Parikshit and Gray, Alexander G.},
  booktitle={Proceedings of the 16th ACM SIGKDD International Conference on
  Knowledge Discovery and Data Mining (KDD '10)},
  pages={603--612},
  year={2010},
  organization={ACM}
}
```

mlpack provides:

- a **simple command-line executable** (p. 118) to compute the EMST of a given data set
- a **simple C++ interface** (p. 119) to compute the EMST

21.2 Table of Contents

A list of all the sections this tutorial contains.

- **Introduction** (p. 117)
- **Table of Contents** (p. 118)
- **Command-Line 'EMST'** (p. 118)
- **The 'DualTreeBoruvka' class** (p. 119)
- **Further documentation** (p. 119)

21.3 Command-Line 'EMST'

The `mlpack_emst` executable in **mlpack** will compute the EMST of a given set of points and store the resulting edge list to a file.

The output file contains an edge list representation of the MST in an $n-1 \times 3$ matrix, where the first and second columns are labels of points and the third column is the edge weight. The edges are sorted in order of increasing weight.

Below are several examples of simple usage (and the resultant output). The `-v` option is used so that verbose output is given. Further documentation on each individual option can be found by typing

```
$ mlpack_emst --help

$ mlpack_emst --input_file=dataset.csv --output_file=edge_list.csv -v
[INFO ] Reading in data.
[INFO ] Loading 'dataset.csv' as CSV data.
[INFO ] Data read, building tree.
[INFO ] Tree built, running algorithm.
[INFO ] 4 edges found so far.
[INFO ] 5 edges found so far.
[INFO ] Total spanning tree length: 1002.45
[INFO ] Saving CSV data to 'edge_list.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_file: dataset.csv
[INFO ]   leaf_size: 1
[INFO ]   naive: false
[INFO ]   output_file: edge_list.csv
[INFO ]   verbose: true
[INFO ]
[INFO ] Program timers:
[INFO ]   emst/mst_computation: 0.000179s
[INFO ]   emst/tree_building: 0.000061s
[INFO ]   total_time: 0.052641s
```

The code performs at most $\log N$ iterations for N data points. It will print an update on the number of MST edges found after each iteration. Convenient program timers are given for different parts of the calculation at the bottom of the output, as well as the parameters the simulation was run with.

```
$ cat dataset.csv
0, 0
1, 1
3, 3
0.5, 0
1000, 0
1001, 0

$ cat edge_list.csv
0.0000000000e+00,3.0000000000e+00,5.0000000000e-01
4.0000000000e+00,5.0000000000e+00,1.0000000000e+00
1.0000000000e+00,3.0000000000e+00,1.1180339887e+00
1.0000000000e+00,2.0000000000e+00,2.8284271247e+00
2.0000000000e+00,4.0000000000e+00,9.9700451353e+02
```

The input points are labeled 0-5. The output tells us that the MST connects point 0 to point 3, point 4 to point 5, point 1 to point 3, point 1 to point 2, and point 2 to point 4, with the corresponding edge weights given in the third column. The total length of the MST is also given in the verbose output.

Note that it is also possible to compute the EMST using a naive ($O(N^2)$) algorithm for timing and comparison purposes, using the `-naive` option.

21.4 The 'DualTreeBoruvka' class

The 'DualTreeBoruvka' class contains our implementation of the Dual-Tree Boruvka algorithm.

The class has two constructors: the first takes the data set, constructs the tree (where the type of tree constructed is the `TreeType` template parameter), and computes the MST. The second takes data set and an already constructed tree.

The class provides one method that performs the MST computation:

```
void ComputeMST(const arma::mat& results);
```

This method stores the computed MST in the matrix results in the format given above.

21.5 Further documentation

For further documentation on the DualTreeBoruvka class, consult the **complete API documentation** (p. 1272).

Chapter 22

Fast max-kernel search tutorial (fastmks)

22.1 Introduction

The FastMKS algorithm (fast exact max-kernel search) is a recent algorithm proposed in the following papers:

```
@inproceedings{curtin2013fast,
  title={Fast Exact Max-Kernel Search},
  author={Curtin, Ryan R. and Ram, Parikshit and Gray, Alexander G.},
  booktitle={Proceedings of the 2013 SIAM International Conference on Data
    Mining (SDM '13)},
  year={2013},
  pages={1--9}
}

@article{curtin2014dual,
  author = {Curtin, Ryan R. and Ram, Parikshit},
  title = {Dual-tree fast exact max-kernel search},
  journal = {Statistical Analysis and Data Mining},
  volume = {7},
  number = {4},
  publisher = {Wiley Subscription Services, Inc., A Wiley Company},
  issn = {1932-1872},
  url = {http://dx.doi.org/10.1002/sam.11218},
  doi = {10.1002/sam.11218},
  pages = {229--253},
  year = {2014},
}
```

Given a set of query points Q and a set of reference points R , the FastMKS algorithm is a fast dual-tree (or single-tree) algorithm which finds

$$\arg \max_{p_r \in R} K(p_q, p_r)$$

for all points $p_q \in Q$ and for some Mercer kernel $K(\cdot, \cdot)$. A Mercer kernel is a kernel that is positive semidefinite; these are the classes of kernels that can be used with the kernel trick. In short, the positive semidefiniteness of a Mercer kernel means that any kernel matrix (or Gram matrix) created on a dataset must be positive semidefinite.

The FastMKS algorithm builds trees on the datasets Q and R in such a way that explicit representation of the points in the kernel space is unnecessary, by using cover trees (**mlpack::tree::CoverTree** (p. 2040)). This allows the algorithm to be run, for instance, on string kernels, where there is no sensible explicit representation. The **mlpack** implementation allows any type of tree that does not require an explicit representation to be used. For more details, see the paper.

At the time of this writing there is no other fast algorithm for exact max-kernel search. **mlpack** implements both single-tree and dual-tree fast max-kernel search.

mlpack provides:

- a **simple command-line executable** (p. 122) to run FastMKS
- a **C++ interface** (p. 125) to run FastMKS

22.2 Table of Contents

A list of all the sections this tutorial contains.

- **Introduction** (p. 121)
- **Table of Contents** (p. 122)
- **Command-line FastMKS (mlpack_fastmks)** (p. 122)
 - **FastMKS with a linear kernel on one dataset** (p. 123)
 - **FastMKS on a reference and query dataset** (p. 124)
 - **FastMKS with a different kernel** (p. 124)
 - **Using single-tree search or naive search** (p. 124)
 - **Parameters for alternate kernels** (p. 124)
 - **Saving a FastMKS model/tree** (p. 125)
 - **Loading a FastMKS model for further searches** (p. 125)
- **The 'FastMKS' class** (p. 125)
 - **FastMKS on one dataset** (p. 126)
 - **FastMKS with a query and reference dataset** (p. 126)
 - **FastMKS with an initialized kernel** (p. 126)
 - **FastMKS with an already-created tree** (p. 127)
- **Writing a custom kernel for FastMKS** (p. 128)
- **Using other tree types for FastMKS** (p. 128)
- **Running FastMKS on objects** (p. 128)
- **Further documentation** (p. 129)

22.3 Command-line FastMKS (mlpack_fastmks)

mlpack provides a command-line program, `mlpack_fastmks`, which is used to perform FastMKS on a given query and reference dataset. It supports numerous different types of kernels:

- **linear kernel** (p. 1446)
- **polynomial kernel** (p. 1451)
- **cosine distance** (p. 1416)
- **Gaussian kernel** (p. 1424)

- **Epanechnikov kernel** (p. 1417)
- **triangular kernel** (p. 1461)
- **hyperbolic tangent kernel** (p. 1430)

Note that when a shift-invariant kernel is used, the results will be the same as nearest neighbor search, so **KNN** (p. 151) may be a better option. A shift-invariant kernel is a kernel that depends only on the distance between the two input points. The **Gaussian kernel** (p. 1424), **Epanechnikov kernel** (p. 1417), and **triangular kernel** (p. 1461) are instances of shift-invariant kernels. The paper contains more details on this situation. The `mlpack_fastmks` executable still provides these kernels as options, though.

The following examples detail usage of the `mlpack_fastmks` program. Note that you can get documentation on all the possible parameters by typing:

```
$ mlpack_fastmks --help
```

22.3.1 FastMKS with a linear kernel on one dataset

If only one dataset is specified (with `-r` or `-reference_file`), the reference dataset is taken to be both the query and reference datasets. The example below finds the 4 maximum kernels of each point in `dataset.csv`, using the default linear kernel.

```
$ mlpack_fastmks -r dataset.csv -k 4 -v -p products.csv -i indices.csv
```

When the operation completes, the values of the kernels are saved in `products.csv` and the indices of the points which give the maximum kernels are saved in `indices.csv`.

```
$ head indices.csv
762,910,863,890
762,910,426,568
910,762,863,426
762,910,863,426
863,910,614,762
762,863,910,614
762,910,488,568
762,910,863,426
910,762,863,426
863,762,910,614
```

```
$ head products.csv
1.6221652894e+00,1.5998743443e+00,1.5898890769e+00,1.5406789753e+00
1.3387953449e+00,1.3317349486e+00,1.2966613184e+00,1.2774493620e+00
1.6386110476e+00,1.6332029753e+00,1.5952629124e+00,1.5887195330e+00
1.0917545803e+00,1.0820878726e+00,1.0668992636e+00,1.0419838050e+00
1.2272441028e+00,1.2169643942e+00,1.2104597963e+00,1.2067780154e+00
1.5720962456e+00,1.5618504956e+00,1.5609069923e+00,1.5235605095e+00
1.3655478674e+00,1.3548593212e+00,1.3311547298e+00,1.3250728881e+00
2.0119149744e+00,2.0043668067e+00,1.9847289214e+00,1.9298280046e+00
1.1586923205e+00,1.1494586097e+00,1.1274872962e+00,1.1248172766e+00
4.4789820372e-01,4.4618539778e-01,4.4200024852e-01,4.3989721792e-01
```

We can see in this example that for point 0, the point with maximum kernel value is point 762, with a kernel value of 1.622165. For point 3, the point with third largest kernel value is point 863, with a kernel value of 1.0669.

22.3.2 FastMKS on a reference and query dataset

The query points may be different than the reference points. To specify a different query set, the `-q` (or `-query_file`) option is used, as in the example below.

```
$ mlpack_fastmks -q query_set.csv -r reference_set.csv -k 5 -i indices.csv \
> -p products.csv
```

22.3.3 FastMKS with a different kernel

The `mlpack_fastmks` program offers more than just the linear kernel. Valid options are 'linear', 'polynomial', 'cosine', 'gaussian', 'epanechnikov', 'triangular' and 'hyptan' (the hyperbolic tangent kernel). Note that the hyperbolic tangent kernel is provably not a Mercer kernel but is positive semidefinite on most datasets and is commonly used as a kernel. Note also that the Gaussian kernel and other shift-invariant kernels give the same results as nearest neighbor search (see **NeighborSearch tutorial (k-nearest-neighbors)** (p. 151)).

The kernel to use is specified with the `-K` (or `-kernel`) option. The example below uses the cosine similarity as a kernel.

```
$ mlpack_fastmks -r dataset.csv -k 5 -K cosine -i indices.csv -p products.csv -v
```

22.3.4 Using single-tree search or naive search

In some cases, it may be useful to not use the dual-tree FastMKS algorithm. Instead you can specify the `-single` option, indicating that a tree should be built only on the reference set, and then the queries should be processed in a linear scan (instead of in a tree). Alternately, the `-N` (or `-naive`) option makes the program not build trees at all and instead use brute-force search to find the solutions.

The example below uses single-tree search on two datasets with the linear kernel.

```
$ mlpack_fastmks -q query_set.csv -r reference_set.csv --single -k 5 \
> -p products.csv -i indices.csv -K linear
```

The example below uses naive search on one dataset.

```
$ mlpack_fastmks -r reference_set.csv -k 5 -N -p products.csv -i indices.csv
```

22.3.5 Parameters for alternate kernels

Many of the alternate kernel choices have parameters which can be chosen; these are detailed in this section.

- `-w` (`-bandwidth`): this sets the bandwidth of the kernel, and is applicable to the 'gaussian', 'epanechnikov', and 'triangular' kernels. This is the "spread" of the kernel.
- `-d` (`-degree`): this sets the degree of the polynomial kernel (the power to which the result is raised). It is only applicable to the 'polynomial' kernel.
- `-o` (`-offset`): this sets the offset of the kernel, for the 'polynomial' and 'hyptan' kernel. See the **polynomial kernel documentation** (p. 1451) and the **hyperbolic tangent kernel documentation** (p. 1430) for more information.
- `-s` (`-scale`): this sets the scale of the kernel, and is only applicable to the 'hyptan' kernel. See the **hyperbolic tangent kernel documentation** (p. 1430) for more information.

22.3.6 Saving a FastMKS model/tree

The `mlpack_fastmks` program also supports saving a model built on a reference dataset (this model includes the tree, the kernel, and the search parameters). The `-output_model_file` or `-M` option allows one to save these parameters to disk for later usage. An example is below:

```
$ mlpack_fastmks -r reference_set.csv -K cosine -M fastmks_model.xml
```

This example builds a tree on the dataset in `reference_set.csv` using the cosine similarity kernel, and saves the resulting model to `fastmks_model.xml`. This model may then be used in later calls to the `mlpack_fastmks` program.

22.3.7 Loading a FastMKS model for further searches

Supposing that a FastMKS model has been saved with the `-output_model_file` or `-M` parameter, that model can then be later loaded in subsequent calls to the `mlpack_fastmks` program, using the `-input_model_file` or `-m` option. For instance, with a model saved in `fastmks_model.xml` and a query set in `query_set.csv`, we can find 3 max-kernel candidates, saving to `indices.csv` and `kernels.csv`:

```
$ mlpack_fastmks -m fastmks_model.xml -k 3 -i indices.csv -p kernels.csv
```

Loading a model as opposed to building a model is advantageous because the reference tree is already built. So, among other situations, this could be useful in the setting where many different query sets (or many different values of k) will be used.

Note that the kernel cannot be changed in a saved model without rebuilding the model entirely.

22.4 The 'FastMKS' class

The `FastMKS<>` class offers a simple API for use within C++ applications, and allows further flexibility in kernel choice and tree type choice. However, `FastMKS<>` has no default template parameter for the kernel type – that must be manually specified. Choices that `mlpack` provides include:

- `mlpack::kernel::LinearKernel` (p. 1446)
- `mlpack::kernel::PolynomialKernel` (p. 1451)
- `mlpack::kernel::CosineDistance` (p. 1416)
- `mlpack::kernel::GaussianKernel` (p. 1424)
- `mlpack::kernel::EpanechnikovKernel` (p. 1417)
- `mlpack::kernel::TriangularKernel` (p. 1461)
- `mlpack::kernel::HyperbolicTangentKernel` (p. 1430)
- `mlpack::kernel::LaplacianKernel` (p. 1442)
- `mlpack::kernel::PSpectrumStringKernel` (p. 1454)

The following examples use kernels from that list. Writing your own kernel is detailed in **the next section** (p. 128). Remember that when you are using the C++ interface, the data matrices must be column-major. See **Matrices in `mlpack`** (p. 57) for more information.

22.4.1 FastMKS on one dataset

Given only a reference dataset, the following code will run FastMKS with k set to 5.

```
#include <mlpack/methods/fastmks/fastmks.hpp>
#include <mlpack/core/kernels/linear_kernel.hpp>

using namespace mlpack::fastmks;

// The reference dataset, which is column-major.
extern arma::mat data;

// This will initialize the FastMKS object with the linear kernel with default
// options:  $K(x, y) = x^T y$ . The tree is built in the constructor.
FastMKS<LinearKernel> f(data);

// The results will be stored in these matrices.
arma::Mat<size_t> indices;
arma::mat products;

// Run FastMKS.
f.Search(5, indices, products);
```

22.4.2 FastMKS with a query and reference dataset

In this setting we have both a query and reference dataset. We search for 10 maximum kernels.

```
#include <mlpack/methods/fastmks/fastmks.hpp>
#include <mlpack/core/kernels/triangular_kernel.hpp>

using namespace mlpack::fastmks;
using namespace mlpack::kernel;

// The reference and query datasets, which are column-major.
extern arma::mat referenceData;
extern arma::mat queryData;

// This will initialize the FastMKS object with the triangular kernel with
// default options (bandwidth of 1). The reference tree is built in the
// constructor.
FastMKS<TriangularKernel> f(referenceData);

// The results will be stored in these matrices.
arma::Mat<size_t> indices;
arma::mat products;

// Run FastMKS. The query tree is built during the call to Search().
f.Search(queryData, 10, indices, products);
```

22.4.3 FastMKS with an initialized kernel

Often, kernels have parameters which need to be specified. FastMKS<> has constructors which take initialized kernels. Note that temporary kernels cannot be passed as an argument. The example below initializes a PolynomialKernel object and then runs FastMKS with a query and reference dataset.

```
#include <mlpack/methods/fastmks/fastmks.hpp>
#include <mlpack/core/kernels/polynomial_kernel.hpp>

using namespace mlpack::fastmks;
using namespace mlpack::kernel;

// The reference and query datasets, which are column-major.
extern arma::mat referenceData;
extern arma::mat queryData;
```

```
// Initialize the polynomial kernel with degree of 3 and offset of 2.5.
PolynomialKernel pk(3.0, 2.5);

// Create the FastMKS object with the initialized kernel.
FastMKS<PolynomialKernel> f(referenceData, pk);

// The results will be stored in these matrices.
arma::Mat<size_t> indices;
arma::mat products;

// Run FastMKS.
f.Search(queryData, 10, indices, products);
```

The syntax for running FastMKS with one dataset and an initialized kernel is very similar:

```
f.Search(10, indices, products);
```

22.4.4 FastMKS with an already-created tree

By default, `FastMKS<>` uses the cover tree datastructure (see `mlpack::tree::CoverTree` (p. 2040)). Sometimes, it is useful to modify the parameters of the cover tree. In this scenario, a tree must be built outside of the constructor, and then passed to the appropriate `FastMKS<>` constructor. An example on just a reference dataset is shown below, where the base of the cover tree is modified.

We also use an instantiated kernel, but because we are building our own tree, we must use `IPMetric` (p. 1565) so that our tree is built on the metric induced by our kernel function.

```
#include <mlpack/methods/fastmks/fastmks.hpp>
#include <mlpack/core/kernels/polynomial_kernel.hpp>

// The reference dataset, which is column-major.
extern arma::mat data;

// Initialize the polynomial kernel with a degree of 4 and offset of 2.0.
PolynomialKernel pk(4.0, 2.0);

// Create the metric induced by this kernel (because a kernel is not a metric
// and we can't build a tree on a kernel alone).
IPMetric<PolynomialKernel> metric(pk);

// Now build a tree on the reference dataset using the instantiated metric and
// the custom base of 1.5 (default is 1.3). We have to be sure to use the right
// type here -- FastMKS needs the FastMKSStat object as the tree's
// StatisticType.
typedef tree::CoverTree<IPMetric<PolynomialKernel>, tree::FirstPointIsRoot,
    FastMKSStat> TreeType; // Convenience typedef.
TreeType* tree = new TreeType(data, metric, 1.5);

// Now initialize FastMKS with that statistic. We don't need to specify the
// TreeType template parameter since we are still using the default. We don't
// need to pass the kernel because that is contained in the tree.
FastMKS<PolynomialKernel> f(tree);

// The results will be stored in these matrices.
arma::Mat<size_t> indices;
arma::mat products;

// Run FastMKS.
f.Search(10, indices, products);
```

The syntax is similar for the case where different query and reference datasets are given; but trees for both need to be built in the manner specified above. Be sure to build both trees using the same metric (or at least a metric with the exact same parameters).

```
f.Search(queryTree, 10, indices, products);
```

22.5 Writing a custom kernel for FastMKS

While **mlpack** provides some number of kernels in the **mlpack::kernel** (p. 401) namespace, it is easy to create a custom kernel. To satisfy the **KernelType** policy, a class must implement the following methods:

```
// Empty constructor is required.
KernelType();

// Evaluate the kernel between two points.
template<typename VecType>
double Evaluate(const VecType& a, const VecType& b);
```

The template parameter **VecType** is helpful (but not necessary) so that the kernel can be used with both sparse and dense matrices (**arma::sp_mat** and **arma::mat**).

22.6 Using other tree types for FastMKS

The use of the cover tree (see **CoverTree** (p. 2040)) is not necessary for FastMKS, although it is the default tree type. A different type of tree can be specified with the **TreeType** template parameter. However, the tree type is required to have **FastMKSStat** (p. 1303) as the **StatisticType**, and for FastMKS to work, the tree must be built only on kernel evaluations (or distance evaluations in the kernel space via **IPMetric::Evaluate()** (p. 1565)).

Below is an example where a custom tree class, **CustomTree**, is used as the tree type for FastMKS. In this example FastMKS is only run on one dataset.

```
#include <mlpack/methods/fastmks/fastmks.hpp>
#include "custom_tree.hpp"

using namespace mlpack::fastmks;
using namespace mlpack::tree;

// The dataset that FastMKS will be run on.
extern arma::mat data;

// The custom tree type. We'll assume that the first template parameter is the
// statistic type.
typedef CustomTree<FastMKSStat> TreeType;

// The FastMKS constructor will create the tree.
FastMKS<LinearKernel, arma::mat, TreeType> f(data);

// These will hold the results.
arma::Mat<size_t> indices;
arma::mat products;

// Run FastMKS.
f.Search(5, indices, products);
```

22.7 Running FastMKS on objects

FastMKS has a lot of utility on objects which are not representable in some sort of metric space. These objects might be strings, graphs, models, or other objects. For these types of objects, questions based on distance don't really make sense. One good example is with strings. The question "how far is 'dog' from 'Taki Inoue'?" simply doesn't make sense. We can't have a centroid of the terms 'Fritz', 'E28', and 'popsicle'.

However, what we can do is define some sort of kernel on these objects. These kernels generally correspond to some similarity measure, with one example being the p-spectrum string kernel (see `mlpack::kernel::PSpectrumStringKernel` (p. 1454)). Using that, we can say "how similar is 'dog' to 'Taki Inoue'?" and get an actual numerical result by evaluating `K('dog', 'Taki Inoue')` (where `K` is our p-spectrum string kernel).

The only requirement on these kernels is that they are positive definite kernels (or Mercer kernels). For more information on those details, refer to the FastMKS paper.

Remember that FastMKS is a tree-based method. But trees like the binary space tree require centroids – and as we said earlier, centroids often don't make sense with these types of objects. Therefore, we need a type of tree which is built **exclusively** on points in the dataset – those are points which we can evaluate our kernel function on. The cover tree is one example of a type of tree satisfying this condition; its construction will only call the kernel function on two points that are in the dataset.

But, we have one more problem. The `CoverTree` class is built on `arma::mat` objects (dense matrices). Our objects, however, are not necessarily representable in a column of a matrix. To use the example we have been using, strings cannot be represented easily in a matrix because they may all have different lengths.

The way to work around this problem is to create a "fake" data matrix which simply holds indices to objects. A good example of how to do this is detailed in the documentation for the `PSpectrumStringKernel` (p. 1454).

In short, the trick is to make each data matrix one-dimensional and containing linear indices:

```
arma::mat data = "0 1 2 3 4 5 6 7 8";
```

Then, when `Evaluate()` is called on the kernel function, the parameters will be two one-dimensional vectors that simply contain indices to objects. The example below details the process a little better:

```
// This function evaluates the kernel on two Objects (in this example, its
// implementation is not important; the only important thing is that the
// function exists).
double ObjectKernel::Evaluate(const Object& a, const Object& b) const;

template<typename VecType>
double ObjectKernel::Evaluate(const VecType& a, const VecType& b) const
{
    // Extract the indices from the vectors.
    const size_t indexA = size_t(a[0]);
    const size_t indexB = size_t(b[0]);

    // Assume that 'objects' is an array (or std::vector or other container)
    // holding Objects.
    const Object& objectA = objects[indexA];
    const Object& objectB = objects[indexB];

    // Now call the function that does the actual evaluation on the objects and
    // return its result.
    return Evaluate(objectA, objectB);
}
```

As written earlier, the documentation for `PSpectrumStringKernel` (p. 1454) is a good place to consult for further reference on this. That kernel uses two dimensional indices; one dimension represents the index of the string, and the other represents whether it is referring to the query set or the reference set. If your kernel is meant to work on separate query and reference sets, that strategy should be considered.

22.8 Further documentation

For further documentation on the FastMKS class, consult the **complete API documentation** (p. 1280).

Chapter 23

K-Means tutorial (kmeans)

23.1 Introduction

The popular k-means algorithm for clustering has been around since the late 1950s, and the standard algorithm was proposed by Stuart Lloyd in 1957. Given a set of points X , k-means clustering aims to partition each point x_i into a cluster c_j (where $j \leq k$ and k , the number of clusters, is a parameter). The partitioning is done to minimize the objective function

$$\sum_{j=1}^k \sum_{x_i \in c_j} \|x_i - \mu_j\|^2$$

where μ_j is the centroid of cluster c_j . The standard algorithm is a two-step algorithm:

- **Assignment step.** Each point x_i in X is assigned to the cluster whose centroid it is closest to.
- **Update step.** Using the new cluster assignments, the centroids of each cluster are recalculated.

The algorithm has converged when no more assignment changes are happening with each iteration. However, this algorithm can get stuck in local minima of the objective function and is particularly sensitive to the initial cluster assignments. Also, situations can arise where the algorithm will never converge but reaches steady state – for instance, one point may be changing between two cluster assignments.

There is vast literature on the k-means algorithm and its uses, as well as strategies for choosing initial points effectively and keeping the algorithm from converging in local minima. **mlpack** does implement some of these, notably the Bradley-Fayyad algorithm (see the reference below) for choosing refined initial points. Importantly, the C++ `KMeans` class makes it very easy to improve the k-means algorithm in a modular way.

```
@inproceedings{bradley1998refining,
  title={Refining initial points for k-means clustering},
  author={Bradley, Paul S. and Fayyad, Usama M.},
  booktitle={Proceedings of the Fifteenth International Conference on Machine
    Learning (ICML 1998)},
  volume={66},
  year={1998}
}
```

mlpack provides:

- a **simple command-line executable** (p. 132) to run k-means
- a **simple C++ interface** (p. 135) to run k-means
- a **generic, extensible, and powerful C++ class** (p. 138) for complex usage

23.2 Table of Contents

A list of all the sections this tutorial contains.

- **Introduction** (p. 131)
- **Table of Contents** (p. 132)
- **Command-Line 'kmeans'** (p. 132)
 - **Simple k-means clustering** (p. 133)
 - **Saving the resulting centroids** (p. 133)
 - **Allowing empty clusters** (p. 133)
 - **Limiting the maximum number of iterations** (p. 133)
 - **Using Bradley-Fayyad "refined start"** (p. 134)
 - **Using different k-means algorithms** (p. 134)
- **The 'KMeans' class** (p. 135)
 - **Running k-means and getting cluster assignments** (p. 135)
 - **Running k-means and getting centroids of clusters** (p. 135)
 - **Limiting the maximum number of iterations** (p. 136)
 - **Setting initial cluster assignments** (p. 136)
 - **Setting initial cluster centroids** (p. 137)
 - **Running sparse k-means** (p. 138)
- **Template parameters for the 'KMeans' class** (p. 138)
 - **Changing the distance metric used for k-means** (p. 139)
 - **Changing the initial partitioning strategy used for k-means** (p. 139)
 - **Changing the action taken when an empty cluster is encountered** (p. 140)
 - **The LloydStepType template parameter** (p. 141)
- **Further documentation** (p. 141)

23.3 Command-Line 'kmeans'

mlpack provides a command-line executable, `mlpack_kmeans`, to allow easy execution of the k-means algorithm on data. Complete documentation of the executable can be found by typing

```
$ mlpack_kmeans --help
```

As of October 2014, support for overclustering has been removed due to bugs and lack of usage. If this is support you were using, or are interested, please file a bug or get in touch with the **mlpack** developers in some way so that the support can be re-implemented.

Below are several examples demonstrating simple use of the `mlpack_kmeans` executable.

23.3.1 Simple k-means clustering

We want to find 5 clusters using the points in the file `dataset.csv`. By default, if any of the clusters end up empty, that cluster will be reinitialized to contain the point furthest from the cluster with maximum variance. The cluster assignments of each point will be stored in `assignments.csv`. Each row in `assignments.csv` will correspond to the row in `dataset.csv`.

```
$ mlpack_kmeans -c 5 -i dataset.csv -v -o assignments.csv
```

23.3.2 Saving the resulting centroids

Sometimes it is useful to save the centroids of the clusters found by k-means; one example might be for plotting the points. The `-C` (`-centroid_file`) option allows specification of a file into which the centroids will be saved (one centroid per line, if it is a CSV or other text format).

```
$ mlpack_kmeans -c 5 -i dataset.csv -v -o assignments.csv -C centroids.csv
```

23.3.3 Allowing empty clusters

If you would like to allow empty clusters to exist, instead of reinitializing them, simply specify the `-e` (`-allow_empty_clusters`) option. Note that when you save your clusters, even empty clusters will still have centroids. The centroids of the empty cluster will be the same as what they were on the last iteration when the cluster was not empty.

```
$ mlpack_kmeans -c 5 -i dataset.csv -v -e -o assignments.csv -C centroids.csv
```

23.3.4 Allowing empty clusters

If you would like to kill empty clusters, instead of reinitializing them, simply specify the `-E` (`-kill_empty_clusters`) option. Note that when you save your clusters, all the empty clusters will be removed and the final result may contain less than specified number of clusters.

```
$ mlpack_kmeans -c 5 -i dataset.csv -v -E -o assignments.csv -C centroids.csv
```

23.3.5 Limiting the maximum number of iterations

As mentioned earlier, the k-means algorithm can often fail to converge. In such a situation, it may be useful to stop the algorithm by way of limiting the maximum number of iterations. This can be done with the `-m` (`-max_iterations`) parameter, which is set to 1000 by default. If the maximum number of iterations is 0, the algorithm will run until convergence – or potentially forever. The example below sets a maximum of 250 iterations.

```
$ mlpack_kmeans -c 5 -i dataset.csv -v -o assignments.csv -m 250
```

23.3.6 Using Bradley-Fayyad "refined start"

The method proposed by Bradley and Fayyad in their paper "Refining initial points for k-means clustering" is implemented in **mlpack**. This strategy samples points from the dataset and runs k-means clustering on those points multiple times, saving the resulting clusters. Then, k-means clustering is run on those clusters, yielding the original number of clusters. The centroids of those resulting clusters are used as initial centroids for k-means clustering on the entire dataset.

This technique generally gives better initial points than the default random partitioning, but depending on the parameters, it can take much longer. This initialization technique is enabled with the `-r` (`-refined_start`) option. The `-S` (`-samplings`) parameter controls how many samplings of the dataset are performed, and the `-p` (`-percentage`) parameter controls how much of the dataset is randomly sampled for each sampling (it must be between 0.0 and 1.0). For more information on the refined start technique, see the paper referenced in the introduction of this tutorial.

The example below performs k-means clustering, giving 5 clusters, using the refined start technique, sampling 10% of the dataset 25 times to produce the initial centroids.

```
$ mlpack_kmeans -c 5 -i dataset.csv -v -o assignments.csv -r -S 25 -p 0.2
```

23.3.7 Using different k-means algorithms

The `mlpack_kmeans` program implements six different strategies for clustering; each of these gives the exact same results, but will have different runtimes. The particular algorithm to use can be specified with the `-a` or `-algorithm` option. The choices are:

- `naive`: the standard Lloyd iteration; takes $O(kN)$ time per iteration.
- `pelleg-moore`: the 'blacklist' algorithm, which builds a kd-tree on the data. This can be fast when k is small and the dimensionality is reasonably low.
- `elkan`: Elkan's algorithm for k-means, which maintains upper and lower distance bounds between each point and each centroid. This can be very fast, but it does not scale well to the case of large N or k , and uses a lot of memory.
- `hamerly`: Hamerly's algorithm is a variant of Elkan's algorithm that handles memory usage much better and thus can operate with much larger datasets than Elkan's algorithm.
- `dualtree`: The dual-tree algorithm for k-means builds a kd-tree on both the centroids and the points in order to prune away as much work as possible. This algorithm is most effective when both N and k are large.
- `dualtree-covertree`: This is the dual-tree algorithm using cover trees instead of kd-trees. It satisfies the runtime guarantees specified in the dual-tree k-means paper.

In general, the `naive` algorithm will be much slower than the others on datasets that are larger than tiny.

The example below uses the `dualtree` algorithm to perform k-means clustering with 5 clusters on the dataset in `dataset.csv`, using the initial centroids in `initial_centroids.csv`, saving the resulting cluster assignments to `assignments.csv`:

```
$ mlpack_kmeans -i dataset.csv -c 5 -v -I initial_centroids.csv -a dualtree \
> -o assignments.csv
```

23.4 The 'KMeans' class

The `KMeans<>` class (with default template parameters) provides a simple way to run k-means clustering using **mlpack** in C++. The default template parameters for `KMeans<>` will initialize cluster assignments randomly and disallow empty clusters. When an empty cluster is encountered, the point furthest from the cluster with maximum variance is set to the centroid of the empty cluster.

23.4.1 Running k-means and getting cluster assignments

The simplest way to use the `KMeans<>` class is to pass in a dataset and a number of clusters, and receive the cluster assignments in return. Note that the dataset must be column-major – that is, one column corresponds to one point. See **the matrices guide** (p. 57) for more information.

```
#include <mlpack/methods/kmeans/kmeans.hpp>

using namespace mlpack::kmeans;

// The dataset we are clustering.
extern arma::mat data;
// The number of clusters we are getting.
extern size_t clusters;

// The assignments will be stored in this vector.
arma::Row<size_t> assignments;

// Initialize with the default arguments.
KMeans<> k;
k.Cluster(data, clusters, assignments);
```

Now, the vector `assignments` holds the cluster assignments of each point in the dataset.

23.4.2 Running k-means and getting centroids of clusters

Often it is useful to not only have the cluster assignments, but the centroids of each cluster. Another overload of `Cluster()` makes this easily possible:

```
#include <mlpack/methods/kmeans/kmeans.hpp>

using namespace mlpack::kmeans;

// The dataset we are clustering.
extern arma::mat data;
// The number of clusters we are getting.
extern size_t clusters;

// The assignments will be stored in this vector.
arma::Row<size_t> assignments;
// The centroids will be stored in this matrix.
arma::mat centroids;

// Initialize with the default arguments.
KMeans<> k;
k.Cluster(data, clusters, assignments, centroids);
```

Note that the centroids matrix has columns equal to the number of clusters and rows equal to the dimensionality of the dataset. Each column represents the centroid of the according cluster – `centroids.col(0)` represents the centroid of the first cluster.

23.4.3 Limiting the maximum number of iterations

The first argument to the constructor allows specification of the maximum number of iterations. This is useful because often, the k-means algorithm does not converge, and is terminated after a number of iterations. Setting this parameter to 0 indicates that the algorithm will run until convergence – note that in some cases, convergence may never happen. The default maximum number of iterations is 1000.

```
// The first argument is the maximum number of iterations. Here we set it to
// 500 iterations.
KMeans<> k(500);
```

Then you can run `Cluster()` as normal.

23.4.4 Setting initial cluster assignments

If you have an initial guess for the cluster assignments for each point, you can fill the assignments vector with the guess and then pass an extra boolean (`initialAssignmentGuess`) as true to the `Cluster()` method. Below are examples for either overload of `Cluster()`.

```
#include <mlpack/methods/kmeans/kmeans.hpp>

using namespace mlpack::kmeans;

// The dataset we are clustering on.
extern arma::mat dataset;
// The number of clusters we are obtaining.
extern size_t clusters;

// A vector pre-filled with initial assignment guesses.
extern arma::Row<size_t> assignments;

KMeans<> k;

// The boolean set to true indicates that our assignments vector is filled with
// initial guesses.
k.Cluster(dataset, clusters, assignments, true);
```

```
#include <mlpack/methods/kmeans/kmeans.hpp>

using namespace mlpack::kmeans;

// The dataset we are clustering on.
extern arma::mat dataset;
// The number of clusters we are obtaining.
extern size_t clusters;

// A vector pre-filled with initial assignment guesses.
extern arma::Row<size_t> assignments;

// This will hold the centroids of the finished clusters.
arma::mat centroids;

KMeans<> k;

// The boolean set to true indicates that our assignments vector is filled with
// initial guesses.
k.Cluster(dataset, clusters, assignments, centroids, true);
```


Note

If you have a heuristic or algorithm which makes initial guesses, a more elegant solution is to create a new class fulfilling the InitialPartitionPolicy template policy. See **the section about changing the initial partitioning strategy** (p. 139) for more details.

Note

If you set the InitialPartitionPolicy parameter to something other than the default but give an initial cluster assignment guess, the InitialPartitionPolicy will not be used to initialize the algorithm. See **the section about changing the initial partitioning strategy** (p. 139) for more details.

23.4.5 Setting initial cluster centroids

An equally important option to being able to make initial cluster assignment guesses is to make initial cluster centroid guesses without having to assign each point in the dataset to an initial cluster. This is similar to the previous section, but now you must pass two extra booleans – the first (initialAssignmentGuess) as false, indicating that there are not initial cluster assignment guesses, and the second (initialCentroidGuess) as true, indicating that the centroids matrix is filled with initial centroid guesses.

This, of course, only works with the overload of `Cluster()` that takes a matrix to put the resulting centroids in. Below is an example.

```
#include <mlpack/methods/kmeans/kmeans.hpp>

using namespace mlpack::kmeans;

// The dataset we are clustering on.
extern arma::mat dataset;
// The number of clusters we are obtaining.
extern size_t clusters;

// A matrix pre-filled with guesses for the initial cluster centroids.
extern arma::mat centroids;

// This will be filled with the final cluster assignments for each point.
arma::Row<size_t> assignments;

KMeans<> k;

// Remember, the first boolean indicates that we are not giving initial
// assignment guesses, and the second boolean indicates that we are giving
// initial centroid guesses.
k.Cluster(dataset, clusters, assignments, centroids, false, true);
```

Note

If you have a heuristic or algorithm which makes initial guesses, a more elegant solution is to create a new class fulfilling the InitialPartitionPolicy template policy. See **the section about changing the initial partitioning strategy** (p. 139) for more details.

Note

If you set the InitialPartitionPolicy parameter to something other than the default but give an initial cluster centroid guess, the InitialPartitionPolicy will not be used to initialize the algorithm. See **the section about changing the initial partitioning strategy** (p. 139) for more details.

23.4.6 Running sparse k-means

The `Cluster()` function can work on both sparse and dense matrices, so all of the above examples can be used with sparse matrices instead, if the fifth template parameter is modified. Below is a simple example. Note that the centroids are returned as a dense matrix, because the centroids of collections of sparse points are not generally sparse.

```
// The sparse dataset.
extern arma::sp_mat sparseDataset;
// The number of clusters.
extern size_t clusters;

// The assignments will be stored in this vector.
arma::Row<size_t> assignments;
// The centroids of each cluster will be stored in this sparse matrix.
arma::sp_mat sparseCentroids;

// We must change the fifth (and last) template parameter.
KMeans<metric::EuclideanDistance, SampleInitialization, MaxVarianceNewCluster,
      NaiveKMeans, arma::sp_mat> k;
k.Cluster(sparseDataset, clusters, assignments, sparseCentroids);
```

23.5 Template parameters for the 'KMeans' class

The `KMeans<>` class also takes three template parameters, which can be modified to change the behavior of the k-means algorithm. There are three template parameters:

- **MetricType**: controls the distance metric used for clustering (by default, the squared Euclidean distance is used)
- **InitialPartitionPolicy**: the method by which initial clusters are set; by default, **SampleInitialization** (p. 1506) is used
- **EmptyClusterPolicy**: the action taken when an empty cluster is encountered; by default, **MaxVarianceNewCluster** (p. 1489) is used
- **LloydStepType**: this defines the strategy used to make a single Lloyd iteration; by default this is the typical Lloyd iteration specified in **NaiveKMeans** (p. 1491)
- **MatType**: type of data matrix to use for clustering

The class is defined like below:

```
template<
  typename DistanceMetric = mlpack::metric::SquaredEuclideanDistance,
  typename InitialPartitionPolicy = SampleInitialization,
  typename EmptyClusterPolicy = MaxVarianceNewCluster,
  template<class, class> class LloydStepType = NaiveKMeans,
  typename MatType = arma::mat
>
class KMeans;
```

In the following sections, each policy is described further, with examples of how to modify them.

23.5.1 Changing the distance metric used for k-means

Most machine learning algorithms in **mlpack** support modifying the distance metric, and **KMeans<>** is no exception. Similar to **NeighborSearch** (p. 1627) (see [the section in the NeighborSearch tutorial](#) (p. 156)), any class in **mlpack::metric** (p. 426) can be given as an argument. The **mlpack::metric::LMetric** (p. 1569) class is a good example implementation.

A class fulfilling the **MetricType** policy must provide the following two functions:

```
// Empty constructor is required.
MetricType();

// Computer the distance between two points.
template<typename VecType>
double Evaluate(const VecType& a, const VecType& b);
```

Most of the standard metrics that could be used are stateless and therefore the **Evaluate()** method is implemented statically. However, there are metrics, such as the Mahalanobis distance (**mlpack::metric::MahalanobisDistance** (p. 1572)), that store state. To this end, an instantiated **MetricType** object is stored within the **KMeans** class. The example below shows how to pass an instantiated **MahalanobisDistance** in the constructor.

```
// The initialized Mahalanobis distance.
extern mlpack::metric::MahalanobisDistance distance;

// We keep the default arguments for the maximum number of iterations, but pass
// our instantiated metric.
KMeans<mlpack::metric::MahalanobisDistance> k(1000, distance);
```

Note

While the **MetricType** policy only requires two methods, one of which is an empty constructor, more can always be added. **mlpack::metric::MahalanobisDistance** (p. 1572) also has constructors with parameters, because it is a stateful metric.

23.5.2 Changing the initial partitioning strategy used for k-means

There have been many initial cluster strategies for k-means proposed in the literature. Fortunately, the **KMeans<>** class makes it very easy to implement one of these methods and plug it in without needing to modify the existing algorithm code at all.

By default, the **KMeans<>** class uses **mlpack::kmeans::SampleInitialization** (p. 1506), which randomly samples points as initial centroids. However, writing a new policy is simple; it needs to only implement the following functions:

```
// Empty constructor is required.
InitialPartitionPolicy();

// Only *one* of the following two functions is required! You should implement
// whichever you find more convenient to implement.

// This function is called to initialize the clusters and returns centroids.
template<typename MatType>
void Cluster(MatType& data,
            const size_t clusters,
            arma::mat& centroids);

// This function is called to initialize the clusters and returns individual
// point assignments. The centroids will then be calculated from the given
// assignments.
template<typename MatType>
void Cluster(MatType& data,
            const size_t clusters,
            arma::Row<size_t> assignments);
```

The templatization of the `Cluster()` function allows both dense and sparse matrices to be passed in. If the desired policy does not work with sparse (or dense) matrices, then the method can be written specifically for one type of matrix – however, be warned that if you try to use `KMeans` with that policy and the wrong type of matrix, you will get many ugly compilation errors!

```
// The Cluster() function specialized for dense matrices.
void Cluster(arma::mat& data,
             const size_t clusters,
             arma::Row<size_t> assignments);
```

Note that only one of the two possible `Cluster()` functions are required. This is because sometimes it is easier to express an initial partitioning policy as something that returns point assignments, and sometimes it is easier to express the policy as something that returns centroids. The `KMeans<>` class will use whichever of these two functions is given; if both are given, the overload that returns centroids will be preferred.

One alternate to the default `SampleInitialization` policy is the `RefinedStart` policy, which is an implementation of the Bradley and Fayyad approach for finding initial points detailed in "Refined initial points for k-means clustering" and other places in this document. Another option is the `RandomPartition` class, which randomly assigns points to clusters, but this may not work very well for most settings. See the documentation for `mlpack::kmeans::RefinedStart` (p. 1502) and `mlpack::kmeans::RandomPartition` (p. 1500) for more information.

If the `Cluster()` method returns point assignments instead of centroids, then valid initial assignments must be returned for every point in the dataset.

As with the `MetricType` template parameter, an initialized `InitialPartitionPolicy` can be passed to the constructor of `KMeans` as a fourth argument.

23.5.3 Changing the action taken when an empty cluster is encountered

Sometimes, during clustering, a situation will arise where a cluster has no points in it. The `KMeans` class allows easy customization of the action to be taken when this occurs. By default, the point furthest from the centroid of the cluster with maximum variance is taken as the centroid of the empty cluster; this is implemented in the `mlpack::kmeans::MaxVarianceNewCluster` (p. 1489) class. Another alternate choice is the `mlpack::kmeans::AllowEmptyClusters` (p. 1464) class, which simply allows empty clusters to persist.

A custom policy can be written and it must implement the following methods:

```
// Empty constructor is required.
EmptyClusterPolicy();

// This function is called when an empty cluster is encountered. emptyCluster
// indicates the cluster which is empty, and then the clusterCounts and
// assignments are meant to be modified by the function. The function should
// return the number of modified points.
template<typename MatType>
size_t EmptyCluster(const MatType& data,
                   const size_t emptyCluster,
                   const MatType& centroids,
                   arma::Col<size_t>& clusterCounts,
                   arma::Row<size_t>& assignments);
```

The `EmptyCluster()` function is called for each cluster that is empty at each iteration of the algorithm. As with `InitialPartitionPolicy`, the `EmptyCluster()` function does not need to be generalized to support both dense and sparse matrices – but usage with the wrong type of matrix will cause compilation errors.

Like the other template parameters to `KMeans`, `EmptyClusterPolicy` implementations that have state can be passed to the constructor of `KMeans` as a fifth argument. See the `kmeans::KMeans` documentation for further details.

23.5.4 The LloydStepType template parameter

The internal algorithm used for a single step of the k-means algorithm can easily be changed; **mlpack** implements several existing classes that satisfy the `LloydStepType` policy:

- `mlpack::kmeans::NaiveKMeans` (p. 1491)
- `mlpack::kmeans::ElkanKMeans` (p. 1478)
- `mlpack::kmeans::HamerlyKMeans` (p. 1479)
- `mlpack::kmeans::PellegMooreKMeans` (p. 1493)
- `mlpack::kmeans::DualTreeKMeans` (p. 1466)

Note that the `LloydStepType` policy is itself a template template parameter, and must accept two template parameters of its own:

- `MetricType`: the type of metric to use
- `MatType`: the type of data matrix to use

The `LloydStepType` policy also mandates three functions:

- a constructor: `LloydStepType(const MatType& dataset, MetricType& metric);`
- an `Iterate()` function:

```
double Iterate(const arma::mat& centroids,
               arma::mat& newCentroids,
               arma::Col<size_t>& counts);
```

- a function to get the number of distance calculations:

```
size_t DistanceCalculations() const { return distanceCalculations; }
```

Note that `Iterate()` does not need to return valid centroids if the cluster is empty. This is because `EmptyClusterPolicy` will handle the empty centroid. This behavior can be used to avoid small amounts of computation.

For examples, see the five aforementioned implementations of classes that satisfy the `LloydStepType` policy.

23.6 Further documentation

For further documentation on the `KMeans` class, consult the **complete API documentation** (p. 1483).

Chapter 24

Linear/ridge regression tutorial (mlpack_linear_regression)

24.1 Introduction

Linear regression and ridge regression are simple machine learning techniques that aim to estimate the parameters of a linear model. Assuming we have n **predictor** points \mathbf{x}_i , $0 \leq i < n$ of dimensionality d and n responses y_i , $0 \leq i < n$, we are trying to estimate the best fit for β_i , $0 \leq i \leq d$ in the linear model

$$y_i = \beta_0 + \sum_{j=1}^d \beta_j x_{ij}$$

for each predictor \mathbf{x}_i and response y_i . If we take each predictor \mathbf{x}_i as a row in the matrix \mathbf{X} and each response y_i as an entry of the vector \mathbf{y} , we can represent the model in vector form:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \beta_0$$

The result of this method is the vector $\boldsymbol{\beta}$, including the offset term (or intercept term) β_0 .

mlpack provides:

- a **simple command-line executable** (p. 144) to perform linear regression or ridge regression
- a **simple C++ interface** (p. 148) to perform linear regression or ridge regression

24.2 Table of Contents

A list of all the sections this tutorial contains.

- **Introduction** (p. 143)
- **Table of Contents** (p. 144)
- **Command-Line 'mlpack_linear_regression'** (p. 144)
 - **One file, generating the function coefficients** (p. 145)
 - **Compute model and predict at the same time** (p. 145)
 - **Prediction using a precomputed model** (p. 146)
 - **Using ridge regression** (p. 147)
- **The 'LinearRegression' class** (p. 148)
 - **Generating a model** (p. 148)
 - **Setting a model** (p. 148)
 - **Load a model from a file** (p. 148)
 - **Prediction** (p. 149)
 - **Setting lambda for ridge regression** (p. 149)
- **Further documentation** (p. 149)

24.3 Command-Line 'mlpack_linear_regression'

The simplest way to perform linear regression or ridge regression in **mlpack** is to use the `mlpack_linear_regression` executable. This program will perform linear regression and place the resultant coefficients into one file.

The output file holds a vector of coefficients in increasing order of dimension; that is, the offset term (β_0), the coefficient for dimension 1 (β_1), then dimension 2 (β_2) and so forth, as well as the intercept. This executable can also predict the y values of a second dataset based on the computed coefficients.

Below are several examples of simple usage (and the resultant output). The `option` is used so that verbose output is given. Further documentation on each individual option can be found by typing

```
$ mlpack_linear_regression --help
```


24.3.1 One file, generating the function coefficients

```
$ mlpack_linear_regression --training_file dataset.csv -v -M lr.xml
[INFO ] Loading 'dataset.csv' as CSV data. Size is 2 x 5.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_model_file: ""
[INFO ]   lambda: 0
[INFO ]   output_model_file: lr.xml
[INFO ]   output_predictions: predictions.csv
[INFO ]   test_file: ""
[INFO ]   training_file: dataset.csv
[INFO ]   training_responses: ""
[INFO ]   verbose: true
[INFO ]   version: false
[INFO ]
[INFO ] Program timers:
[INFO ]   load_regressors: 0.000263s
[INFO ]   loading_data: 0.000220s
[INFO ]   regression: 0.000392s
[INFO ]   total_time: 0.001920s
```

Convenient program timers are given for different parts of the calculation at the bottom of the output, as well as the parameters the simulation was run with. Now, if we look at the output model file, which is `lr.xml`,

```
$ cat dataset.csv
0,0
1,1
2,2
3,3
4,4

$ cat lr.xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE boost_serialization>
<boost_serialization signature="serialization::archive" version="12">
<linearRegressionModel class_id="0" tracking_level="0" version="0">
  <parameters class_id="1" tracking_level="0" version="0">
    <n_rows>2</n_rows>
    <n_cols>1</n_cols>
    <n_elem>2</n_elem>
    <vec_state>1</vec_state>
    <item>-3.97205464519563669e-16</item>
    <item>1.00000000000000022e+00</item>
  </parameters>
  <lambda>0.0000000000000000e+00</lambda>
  <intercept>1</intercept>
</linearRegressionModel>
</boost_serialization>
```

As you can see, the function for this input is $f(y) = 0 + 1x_1$. We can see that the model we have trained catches this; in the `<parameters>` section of `lr.xml`, we can see that there are two elements, which are (approximately) 0 and 1. The first element corresponds to the intercept 0, and the second column corresponds to the coefficient 1 for the variable x_1 . Note that in this example, the regressors for the dataset are the second column. That is, the dataset is one dimensional, and the last column has the y values, or responses, for each row. You can specify these responses in a separate file if you want, using the `-input_responses`, or `-r`, option.

24.3.2 Compute model and predict at the same time

```
$ mlpack_linear_regression --training_file dataset.csv --test_file predict.csv \
> -v
[INFO ] Loading 'dataset.csv' as CSV data. Size is 2 x 5.
[INFO ] Loading 'predict.csv' as raw ASCII formatted data. Size is 1 x 3.
[INFO ] Saving CSV data to 'predictions.csv'.
```

```
[INFO ]
[INFO ] Execution parameters:
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_model_file: ""
[INFO ]   lambda: 0
[INFO ]   output_model_file: ""
[INFO ]   output_predictions: predictions.csv
[INFO ]   test_file: predict.csv
[INFO ]   training_file: dataset.csv
[INFO ]   training_responses: ""
[INFO ]   verbose: true
[INFO ]   version: false
[INFO ]
[INFO ] Program timers:
[INFO ]   load_regressors: 0.000371s
[INFO ]   load_test_points: 0.000229s
[INFO ]   loading_data: 0.000491s
[INFO ]   prediction: 0.000075s
[INFO ]   regression: 0.000449s
[INFO ]   saving_data: 0.000186s
[INFO ]   total_time: 0.002731s
```

```
$ cat dataset.csv
```

```
0,0
1,1
2,2
3,3
4,4
```

```
$ cat predict.csv
```

```
2
3
4
```

```
$ cat predictions.csv
```

```
2.000000000000e+00
3.000000000000e+00
4.000000000000e+00
```

We used the same dataset, so we got the same parameters. The key thing to note about the `predict.csv` dataset is that it has the same dimensionality as the dataset used to create the model, one. If the model generating dataset has d dimensions, so must the dataset we want to predict for.

24.3.3 Prediction using a precomputed model

```
$ mlpack_linear_regression --input_model_file lr.xml --test_file predict.csv -v
```

```
[INFO ] Loading 'predict.csv' as raw ASCII formatted data. Size is 1 x 3.
```

```
[INFO ] Saving CSV data to 'predictions.csv'.
```

```
[INFO ]
```

```
[INFO ] Execution parameters:
```

```
[INFO ]   help: false
```

```
[INFO ]   info: ""
```

```
[INFO ]   input_model_file: lr.xml
```

```
[INFO ]   lambda: 0
```

```
[INFO ]   output_model_file: ""
```

```
[INFO ]   output_predictions: predictions.csv
```

```
[INFO ]   test_file: predict.csv
```

```
[INFO ]   training_file: ""
```

```
[INFO ]   training_responses: ""
```

```
[INFO ]   verbose: true
```

```
[INFO ]   version: false
```

```
[INFO ]
```

```
[INFO ] Program timers:
```

```
[INFO ]   load_model: 0.000264s
```

```
[INFO ]   load_test_points: 0.000186s
```

```
[INFO ]   loading_data: 0.000157s
```

```
[INFO ]   prediction: 0.000098s
```

```
[INFO ]   saving_data: 0.000157s
```

```
[INFO ]   total_time: 0.001688s
```

```
$ cat lr.xml
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
```

```

<!DOCTYPE boost_serialization>
<boost_serialization signature="serialization::archive" version="12">
<linearRegressionModel class_id="0" tracking_level="0" version="0">
  <parameters class_id="1" tracking_level="0" version="0">
    <n_rows>2</n_rows>
    <n_cols>1</n_cols>
    <n_elem>2</n_elem>
    <vec_state>1</vec_state>
    <item>-3.97205464519563669e-16</item>
    <item>1.00000000000000022e+00</item>
  </parameters>
  <lambda>0.00000000000000000e+00</lambda>
  <intercept>1</intercept>
</linearRegressionModel>
</boost_serialization>

$ cat predict.csv
2
3
4

$ cat predictions.csv
2.0000000000e+00
3.0000000000e+00
4.0000000000e+00

```

24.3.4 Using ridge regression

Sometimes, the input matrix of predictors has a covariance matrix that is not invertible, or the system is overdetermined. In this case, ridge regression is useful: it adds a normalization term to the covariance matrix to make it invertible. Ridge regression is a standard technique and documentation for the mathematics behind it can be found anywhere on the Internet. In short, the covariance matrix

$$\mathbf{X}'\mathbf{X}$$

is replaced with

$$\mathbf{X}'\mathbf{X} + \lambda \mathbf{I}$$

where \mathbf{I} is the identity matrix. So, a λ parameter greater than zero should be specified to perform ridge regression, using the `-lambda` (or `-l`) option. An example is given below.

```

$ mlpack_linear_regression --training_file dataset.csv -v --lambda 0.5 -M lr.xml
[INFO ] Loading 'dataset.csv' as CSV data. Size is 2 x 5.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_model_file: ""
[INFO ]   lambda: 0.5
[INFO ]   output_model_file: lr.xml
[INFO ]   output_predictions: predictions.csv
[INFO ]   test_file: ""
[INFO ]   training_file: dataset.csv
[INFO ]   training_responses: ""
[INFO ]   verbose: true
[INFO ]   version: false
[INFO ]
[INFO ] Program timers:
[INFO ]   load_regressors: 0.000210s
[INFO ]   loading_data: 0.000170s
[INFO ]   regression: 0.000332s
[INFO ]   total_time: 0.001835s

```

Further documentation on options should be found by using the `-help` option.

24.4 The 'LinearRegression' class

The 'LinearRegression' class is a simple implementation of linear regression.

Using the LinearRegression class is very simple. It has two available constructors; one for generating a model from a matrix of predictors and a vector of responses, and one for loading an already computed model from a given file.

The class provides one method that performs computation:

```
void Predict(const arma::mat& points, arma::vec& predictions);
```

Once you have generated or loaded a model, you can call this method and pass it a matrix of data points to predict values for using the model. The second parameter, predictions, will be modified to contain the predicted values corresponding to each row of the points matrix.

24.4.1 Generating a model

```
#include <mlpack/methods/linear_regression/linear_regression.hpp>

using namespace mlpack::regression;

arma::mat data; // The dataset itself.
arma::vec responses; // The responses, one row for each row in data.

// Regress.
LinearRegression lr(data, responses);

// Get the parameters, or coefficients.
arma::vec parameters = lr.Parameters();
```

24.4.2 Setting a model

Assuming you already have a model and do not need to create one, this is how you would set the parameters for a LinearRegression instance.

```
arma::vec parameters; // Your model.

LinearRegression lr; // Create a new LinearRegression instance or reuse one.
lr.Parameters() = parameters; // Set the model.
```

24.4.3 Load a model from a file

If you have a generated model in a file somewhere you would like to load and use, you can use `data::Load()` (p. 379) to load it.

```
std::string filename; // The path and name of your file.

LinearRegression lr;
data::Load(filename, "lr_model", lr);
```

24.4.4 Prediction

Once you have generated or loaded a model using one of the aforementioned methods, you can predict values for a dataset.

```
LinearRegression lr();  
// Load or generate your model.  
  
// The dataset we want to predict on; each row is a data point.  
arma::mat points;  
// This will store the predictions; one row for each point.  
arma::vec predictions;  
  
lr.Predict(points, predictions); // Predict.  
  
// Now, the vector 'predictions' will contain the predicted values.
```

24.4.5 Setting lambda for ridge regression

As discussed in **Using ridge regression** (p. 147), ridge regression is useful when the covariance of the predictors is not invertible. The standard constructor can be used to set a value of lambda:

```
#include <mlpack/methods/linear_regression/linear_regression.hpp>  
  
using namespace mlpack::regression;  
  
arma::mat data; // The dataset itself.  
arma::vec responses; // The responses, one row for each row in data.  
  
// Regress, with a lambda of 0.5.  
LinearRegression lr(data, responses, 0.5);  
  
// Get the parameters, or coefficients.  
arma::vec parameters = lr.Parameters();
```

In addition, the `Lambda()` function can be used to get or modify the lambda value:

```
LinearRegression lr;  
lr.Lambda() = 0.5;  
Log::Info << "Lambda is " << lr.Lambda() << "." << std::endl;
```

24.5 Further documentation

For further documentation on the `LinearRegression` class, consult the **complete API documentation** (p. 1789).

Chapter 25

NeighborSearch tutorial (k-nearest-neighbors)

25.1 Introduction

Nearest-neighbors search is a common machine learning task. In this setting, we have a **query** and a **reference** dataset. For each point in the **query** dataset, we wish to know the k points in the **reference** dataset which are closest to the given query point.

Alternately, if the query and reference datasets are the same, the problem can be stated more simply: for each point in the dataset, we wish to know the k nearest points to that point.

mlpack provides:

- a **simple command-line executable** (p. 152) to run nearest-neighbors search (and furthest-neighbors search)
- a **simple C++ interface** (p. 154) to perform nearest-neighbors search (and furthest-neighbors search)
- a **generic, extensible, and powerful C++ class (NeighborSearch)** (p. 156) for complex usage

25.2 Table of Contents

A list of all the sections this tutorial contains.

- **Introduction** (p. 151)
- **Table of Contents** (p. 151)
- **Command-Line 'mlpack_knn'** (p. 152)
 - **One dataset, 5 nearest neighbors** (p. 152)
 - **Query and reference dataset, 10 nearest neighbors** (p. 153)
 - **One dataset, 3 nearest neighbors, leaf size of 15 points** (p. 154)
- **The 'KNN' class** (p. 154)

- 5 nearest neighbors on a single dataset (p. 155)
- 10 nearest neighbors on a query and reference dataset (p. 155)
- Naive (exhaustive) search for 6 nearest neighbors on one dataset (p. 155)
- The extensible 'NeighborSearch' class (p. 156)
 - SortPolicy policy class (p. 156)
 - MetricType policy class (p. 156)
 - MatType policy class (p. 157)
 - TreeType policy class (p. 157)
 - TraverserType policy class (p. 157)
- Further documentation (p. 157)

25.3 Command-Line 'mlpack_knn'

The simplest way to perform nearest-neighbors search in **mlpack** is to use the `mlpack_knn` executable. This program will perform nearest-neighbors search and place the resultant neighbors into one file and the resultant distances into another. The output files are organized such that the first row corresponds to the nearest neighbors of the first query point, with the first column corresponding to the nearest neighbor, and so forth.

Below are several examples of simple usage (and the resultant output). The `-v` option is used so that output is given. Further documentation on each individual option can be found by typing

```
$ mlpack_knn --help
```

25.3.1 One dataset, 5 nearest neighbors

```
$ mlpack_knn -r dataset.csv -n neighbors_out.csv -d distances_out.csv -k 5 -v
[INFO ] Loading 'dataset.csv' as CSV data. Size is 3 x 1000.
[INFO ] Loaded reference data from 'dataset.csv' (3 x 1000).
[INFO ] Building reference tree...
[INFO ] Tree built.
[INFO ] Searching for 5 nearest neighbors with dual-tree kd-tree search...
[INFO ] 18412 node combinations were scored.
[INFO ] 54543 base cases were calculated.
[INFO ] Search complete.
[INFO ] Saving CSV data to 'neighbors_out.csv'.
[INFO ] Saving CSV data to 'distances_out.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   distances_file: distances_out.csv
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_model_file: ""
[INFO ]   k: 5
[INFO ]   leaf_size: 20
[INFO ]   naive: false
[INFO ]   neighbors_file: neighbors_out.csv
[INFO ]   output_model_file: ""
[INFO ]   query_file: ""
[INFO ]   random_basis: false
[INFO ]   reference_file: dataset.csv
[INFO ]   seed: 0
[INFO ]   single_mode: false
[INFO ]   tree_type: kd
[INFO ]   verbose: true
[INFO ]   version: false
[INFO ]
[INFO ] Program timers:
[INFO ]   computing_neighbors: 0.108968s
[INFO ]   loading_data: 0.006495s
[INFO ]   saving_data: 0.003843s
[INFO ]   total_time: 0.126036s
[INFO ]   tree_building: 0.003442s
```


Convenient program timers are given for different parts of the calculation at the bottom of the output, as well as the parameters the simulation was run with. Now, if we look at the output files:

```
$ head neighbors_out.csv
862,344,224,43,885
703,499,805,639,450
867,472,972,380,601
397,319,277,443,323
840,827,865,38,438
732,876,751,492,616
563,222,569,985,940
361,97,928,437,79
547,695,419,961,716
982,113,689,843,634

$ head distances_out.csv
5.986076164057e-02,7.664920518084e-02,1.116050961847e-01,1.155595474371e-01,1.169810085522e-01
7.532635022982e-02,1.012564715841e-01,1.127846944644e-01,1.209584396720e-01,1.216543647014e-01
7.659571546879e-02,1.014588981948e-01,1.025114621511e-01,1.128082429187e-01,1.131659758673e-01
2.079405647909e-02,4.710724516732e-02,7.597622408419e-02,9.17197778898e-02,1.037033340864e-01
7.082206779700e-02,9.002355499742e-02,1.044181406406e-01,1.093149568834e-01,1.139700558608e-01
5.688056488896e-02,9.478072514474e-02,1.085637706630e-01,1.114177921451e-01,1.139370265105e-01
7.882260880455e-02,9.454474078041e-02,9.724494179950e-02,1.023829575445e-01,1.066927013814e-01
7.005321598247e-02,9.131417221561e-02,9.498248889074e-02,9.897964162308e-02,1.121202216165e-01
5.295654132754e-02,5.509877761894e-02,8.108227366619e-02,9.785461174861e-02,1.043968140367e-01
3.992859920333e-02,4.471418646159e-02,7.346053904990e-02,9.181982339584e-02,9.843075910782e-02
```

So, the nearest neighbor to point 0 is point 862, with a distance of 5.986076164057e-02. The second nearest neighbor to point 0 is point 344, with a distance of 7.664920518084e-02. The third nearest neighbor to point 5 is point 751, with a distance of 1.085637706630e-01.

25.3.2 Query and reference dataset, 10 nearest neighbors

```
$ mlpack_knn -q query_dataset.csv -r reference_dataset.csv \
> -n neighbors_out.csv -d distances_out.csv -k 10 -v
[INFO ] Loading 'reference_dataset.csv' as CSV data. Size is 3 x 1000.
[INFO ] Loaded reference data from 'reference_dataset.csv' (3 x 1000).
[INFO ] Building reference tree...
[INFO ] Tree built.
[INFO ] Loading 'query_dataset.csv' as CSV data. Size is 3 x 50.
[INFO ] Loaded query data from 'query_dataset.csv' (3x50).
[INFO ] Searching for 10 nearest neighbors with dual-tree kd-tree search...
[INFO ] Building query tree...
[INFO ] Tree built.
[INFO ] Search complete.
[INFO ] Saving CSV data to 'neighbors_out.csv'.
[INFO ] Saving CSV data to 'distances_out.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   distances_file: distances_out.csv
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_model_file: ""
[INFO ]   k: 10
[INFO ]   leaf_size: 20
[INFO ]   naive: false
[INFO ]   neighbors_file: neighbors_out.csv
[INFO ]   output_model_file: ""
[INFO ]   query_file: query_dataset.csv
[INFO ]   random_basis: false
[INFO ]   reference_file: reference_dataset.csv
[INFO ]   seed: 0
[INFO ]   single_mode: false
[INFO ]   tree_type: kd
[INFO ]   verbose: true
[INFO ]   version: false
[INFO ]
[INFO ] Program timers:
[INFO ]   computing_neighbors: 0.022589s
[INFO ]   loading_data: 0.003572s
[INFO ]   saving_data: 0.000755s
[INFO ]   total_time: 0.032197s
[INFO ]   tree_building: 0.002590s
```

25.3.3 One dataset, 3 nearest neighbors, leaf size of 15 points

```
$ mlpack_knn -r dataset.csv -n neighbors_out.csv -d distances_out.csv -k 3 -l 15 -v
[INFO ] Loading 'dataset.csv' as CSV data. Size is 3 x 1000.
[INFO ] Loaded reference data from 'dataset.csv' (3 x 1000).
[INFO ] Building reference tree...
[INFO ] Tree built.
[INFO ] Searching for 3 nearest neighbors with dual-tree kd-tree search...
[INFO ] 19692 node combinations were scored.
[INFO ] 36263 base cases were calculated.
[INFO ] Search complete.
[INFO ] Saving CSV data to 'neighbors_out.csv'.
[INFO ] Saving CSV data to 'distances_out.csv'.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   distances_file: distances_out.csv
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_model_file: ""
[INFO ]   k: 3
[INFO ]   leaf_size: 15
[INFO ]   naive: false
[INFO ]   neighbors_file: neighbors_out.csv
[INFO ]   output_model_file: ""
[INFO ]   query_file: ""
[INFO ]   random_basis: false
[INFO ]   reference_file: dataset.csv
[INFO ]   seed: 0
[INFO ]   single_mode: false
[INFO ]   tree_type: kd
[INFO ]   verbose: true
[INFO ]   version: false
[INFO ]
[INFO ] Program timers:
[INFO ]   computing_neighbors: 0.059020s
[INFO ]   loading_data: 0.002791s
[INFO ]   saving_data: 0.002369s
[INFO ]   total_time: 0.069277s
[INFO ]   tree_building: 0.002713s
```

Further documentation on options should be found by using the `--help` option.

25.4 The 'KNN' class

The 'KNN' class is, specifically, a typedef of the more extensible NeighborSearch class, querying for nearest neighbors using the Euclidean distance.

```
typedef NeighborSearch<NearestNeighborSort, metric::EuclideanDistance>
    KNN;
```

Using the KNN class is particularly simple; first, the object must be constructed and given a dataset. Then, the method is run, and two matrices are returned: one which holds the indices of the nearest neighbors, and one which holds the distances of the nearest neighbors. These are of the same structure as the output `--neighbors_file` and `--distances_file` for the CLI interface (see above). A handful of examples of simple usage of the KNN class are given below.

25.4.1 5 nearest neighbors on a single dataset

```
#include <mlpack/methods/neighbor_search/neighbor_search.hpp>

using namespace mlpack::neighbor;

// Our dataset matrix, which is column-major.
extern arma::mat data;

KNN a(data);

// The matrices we will store output in.
arma::Mat<size_t> resultingNeighbors;
arma::mat resultingDistances;

a.Search(5, resultingNeighbors, resultingDistances);
```

The output of the search is stored in `resultingNeighbors` and `resultingDistances`.

25.4.2 10 nearest neighbors on a query and reference dataset

```
#include <mlpack/methods/neighbor_search/neighbor_search.hpp>

using namespace mlpack::neighbor;

// Our dataset matrices, which are column-major.
extern arma::mat queryData, referenceData;

KNN a(referenceData);

// The matrices we will store output in.
arma::Mat<size_t> resultingNeighbors;
arma::mat resultingDistances;

a.Search(queryData, 10, resultingNeighbors, resultingDistances);
```

25.4.3 Naive (exhaustive) search for 6 nearest neighbors on one dataset

This example uses the $O(n^2)$ naive search (not the tree-based search).

```
#include <mlpack/methods/neighbor_search/neighbor_search.hpp>

using namespace mlpack::neighbor;

// Our dataset matrix, which is column-major.
extern arma::mat dataset;

KNN a(dataset, true);

// The matrices we will store output in.
arma::Mat<size_t> resultingNeighbors;
arma::mat resultingDistances;

a.Search(6, resultingNeighbors, resultingDistances);
```

Needless to say, naive search can be very slow...

25.5 The extensible 'NeighborSearch' class

The NeighborSearch class is very extensible, having the following template arguments:

```
template<
    typename SortPolicy = NearestNeighborSort,
    typename MetricType = mlpack::metric::EuclideanDistance,
    typename MatType = arma::mat,
    template<typename> TreeMetricType,
    typename TreeStatType,
    typename TreeMatType> class TreeType = tree::KDTree,
    template<typename> RuleType> class TraversalType =
        TreeType<MetricType, NeighborSearchStat<SortPolicy>,
            MatType>::template DualTreeTraverser>
>
class NeighborSearch;
```

By choosing different components for each of these template classes, a very arbitrary neighbor searching object can be constructed. Note that each of these template parameters have defaults, so it is not necessary to specify each one.

25.5.1 SortPolicy policy class

The SortPolicy template parameter allows specification of how the NeighborSearch object will decide which points are to be searched for. The `mlpack::neighbor::NearestNeighborSort` (p. 433) class is a well-documented example. A custom SortPolicy class must implement the same methods which NearestNeighborSort does:

```
static size_t SortDistance(const arma::vec& list, double newDistance);

static bool IsBetter(const double value, const double ref);

template<typename TreeType>
static double BestNodeToNodeDistance(const TreeType* queryNode,
                                     const TreeType* referenceNode);

template<typename TreeType>
static double BestPointToNodeDistance(const arma::vec& queryPoint,
                                     const TreeType* referenceNode);

static const double WorstDistance();

static const double BestDistance();
```

The `mlpack::neighbor::FurthestNeighborSort` (p. 431) class is another implementation, which is used to create the 'KFN' typedef class, which finds the furthest neighbors, as opposed to the nearest neighbors.

25.5.2 MetricType policy class

The MetricType policy class allows the neighbor search to take place in any arbitrary metric space. The `mlpack::metric::LMetric` (p. 1569) class is a good example implementation. A MetricType class must provide the following functions:

```
// Empty constructor is required.
MetricType();

// Compute the distance between two points.
template<typename VecType>
double Evaluate(const VecType& a, const VecType& b);
```

Internally, the NeighborSearch class keeps an instantiated MetricType class (which can be given in the constructor). This is useful for a metric like the Mahalanobis distance (`mlpack::metric::MahalanobisDistance` (p. 1572)), which must store state (the covariance matrix). Therefore, you can write a non-static MetricType class and use it seamlessly with NeighborSearch.

For more information on the MetricType policy, see the documentation [here](#) (p. 177).

25.5.3 MatType policy class

The MatType template parameter specifies the type of data matrix used. This type must implement the same operations as an Armadillo matrix, and so standard choices are `arma::mat` and `arma::sp_mat`.

25.5.4 TreeType policy class

The NeighborSearch class allows great extensibility in the selection of the type of tree used for search. This type must follow the typical mlpack TreeType policy, documented [here](#) (p. 179).

Typical choices might include `mlpack::tree::KDTree` (p. 453), `mlpack::tree::BallTree` (p. 451), `mlpack::tree::↔StandardCoverTree` (p. 460), `mlpack::tree::RTree` (p. 459), or `mlpack::tree::RStarTree` (p. 459). It is easily possible to make your own tree type for use with NeighborSearch; consult the [TreeType documentation](#) (p. 179) for more details.

An example of using the NeighborSearch class with a ball tree is given below.

```
// Construct a NeighborSearch object with ball bounds.
NeighborSearch<
    NearestNeighborSort,
    metric::EuclideanDistance,
    arma::mat,
    tree::BallTree
> neighborSearch(dataset);
```

25.5.5 TraverserType policy class

The last template parameter the NeighborSearch class offers is the TraverserType class. The TraverserType class holds the strategy used to traverse the trees in either single-tree or dual-tree search mode. By default, it is set to use the default traverser of the given TreeType (which is the member `TreeType::DualTreeTraverser`).

This class must implement the following two methods:

```
// Instantiate with a given RuleType.
TraverserType(RuleType& rule);

// Traverse with two trees.
void Traverse(TreeType& queryNode, TreeType& referenceNode);
```

The RuleType class provides the following functions for use in the traverser:

```
// Evaluate the base case between two points.
double BaseCase(const size_t queryIndex, const size_t referenceIndex);

// Score the two nodes to see if they can be pruned, returning DBL_MAX if they
// can be pruned.
double Score(TreeType& queryNode, TreeType& referenceNode);
```

Note also that any traverser given must satisfy the definition of a pruning dual-tree traversal given in the paper "Tree-independent dual-tree algorithms".

25.6 Further documentation

For further documentation on the NeighborSearch class, consult the [complete API documentation](#) (p. 1627).

Chapter 26

RangeSearch tutorial (mlpack_range_search)

26.1 Introduction

Range search is a simple machine learning task which aims to find all the neighbors of a point that fall into a certain range of distances. In this setting, we have a **query** and a **reference** dataset. Given a certain range, for each point in the **query** dataset, we wish to know all points in the **reference** dataset which have distances within that given range to the given query point.

Alternately, if the query and reference datasets are the same, the problem can be stated more simply: for each point in the dataset, we wish to know all points which have distance in the given range to that point.

mlpack provides:

- a **simple command-line executable** (p. 160) to run range search
- a **simple C++ interface** (p. 162) to perform range search
- a **generic, extensible, and powerful C++ class (RangeSearch)** (p. 164) for complex usage

26.2 Table of Contents

A list of all the sections this tutorial contains.

- **Introduction** (p. 159)
- **Table of Contents** (p. 159)
- **The 'mlpack_range_search' command-line executable** (p. 160)
 - **One dataset, points with distance ≤ 0.01** (p. 160)
 - **Query and reference dataset, range [1.0, 1.5]** (p. 161)
 - **One dataset, range [0.7 0.8], leaf size of 15 points** (p. 162)
- **The 'RangeSearch' class** (p. 162)

- Distance less than 2.0 on a single dataset (p. 163)
- Range [3.0, 4.0] on a query and reference dataset (p. 163)
- Naive (exhaustive) search for distance greater than 5.0 on one dataset (p. 163)
- The extensible 'RangeSearch' class (p. 164)
 - MetricType policy class (p. 164)
 - MatType policy class (p. 164)
 - TreeType policy class (p. 164)
- Further documentation (p. 165)

26.3 The 'mlpack_range_search' command-line executable

mlpack provides an executable, `mlpack_range_search`, which can be used to perform range searches quickly and simply from the command-line. This program will perform the range search and place the resulting neighbor index list into one file and their corresponding distances into another file. These files are organized such that the first row corresponds to the neighbors (or distances) of the first query point, and the second row corresponds to the neighbors (or distances) of the second query point, and so forth. The neighbors of a specific point are not arranged in any specific order.

Because a range search may return different numbers of points (including zero), the output file is technically not a valid CSV and may not be loadable by other programs. Therefore, if you need the results in a certain format, it may be better to use the **C++ interface** (p. 162) to manually export the data in the preferred format.

Below are several examples of simple usage (and the resultant output). The '-v' option is used so that output is given. Further documentation on each individual option can be found by typing

```
$ mlpack_range_search --help
```

26.3.1 One dataset, points with distance ≤ 0.01

```
$ mlpack_range_search -r dataset.csv -n neighbors_out.csv -d distances_out.csv \
> -U 0.076 -v
[INFO ] Loading 'dataset.csv' as CSV data. Size is 3 x 1000.
[INFO ] Loaded reference data from 'dataset.csv' (3x1000).
[INFO ] Building reference tree...
[INFO ] Tree built.
[INFO ] Search for points in the range [0, 0.076] with dual-tree kd-tree
search...
[INFO ] Search complete.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   distances_file: distances_out.csv
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_model_file: ""
[INFO ]   leaf_size: 20
[INFO ]   max: 0.01
[INFO ]   min: 0
[INFO ]   naive: false
[INFO ]   neighbors_file: neighbors_out.csv
[INFO ]   output_model_file: ""
[INFO ]   query_file: ""
[INFO ]   random_basis: false
[INFO ]   reference_file: dataset.csv
[INFO ]   seed: 0
[INFO ]   single_mode: false
```



```
[INFO ] tree_type: kd
[INFO ] verbose: true
[INFO ] version: false
[INFO ]
[INFO ] Program timers:
[INFO ]   loading_data: 0.005201s
[INFO ]   range_search/computing_neighbors: 0.017110s
[INFO ]   total_time: 0.033313s
[INFO ]   tree_building: 0.002500s
```

Convenient program timers are given for different parts of the calculation at the bottom of the output, as well as the parameters the simulation was run with. Now, if we look at the output files:

```
$ head neighbors_out.csv
862
703

397, 277, 319
840
732

361
547, 695
113, 982, 689

$ head distances_out.csv
0.0598608
0.0753264

0.0207941, 0.0759762, 0.0471072
0.0708221
0.0568806

0.0700532
0.0529565, 0.0550988
0.0447142, 0.0399286, 0.0734605
```

We can see that only point 862 is within distance 0.076 of point 0. We can also see that point 2 has no points within a distance of 0.076 – that line is empty.

26.3.2 Query and reference dataset, range [1.0, 1.5]

```
$ mlpack_range_search -q query_dataset.csv -r reference_dataset.csv -n \
> neighbors_out.csv -d distances_out.csv -L 1.0 -U 1.5 -v
[INFO ] Loading 'reference_dataset.csv' as CSV data. Size is 3 x 1000.
[INFO ] Loaded reference data from 'reference_dataset.csv' (3x1000).
[INFO ] Building reference tree...
[INFO ] Tree built.
[INFO ] Loading 'query_dataset.csv' as CSV data. Size is 3 x 50.
[INFO ] Loaded query data from 'query_dataset.csv' (3x50).
[INFO ] Search for points in the range [1, 1.5] with dual-tree kd-tree search...
[INFO ] Building query tree...
[INFO ] Tree built.
[INFO ] Search complete.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   distances_file: distances_out.csv
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_model_file: ""
[INFO ]   leaf_size: 20
[INFO ]   max: 1.5
[INFO ]   min: 1
[INFO ]   naive: false
[INFO ]   neighbors_file: neighbors_out.csv
[INFO ]   output_model_file: ""
[INFO ]   query_file: query_dataset.csv
[INFO ]   random_basis: false
[INFO ]   reference_file: reference_dataset.csv
[INFO ]   seed: 0
```

```
[INFO ] single_mode: false
[INFO ] tree_type: kd
[INFO ] verbose: true
[INFO ] version: false
[INFO ] Program timers:
[INFO ]   loading_data: 0.006199s
[INFO ]   range_search/computing_neighbors: 0.024427s
[INFO ]   total_time: 0.045403s
[INFO ]   tree_building: 0.003979s
```

26.3.3 One dataset, range [0.7 0.8], leaf size of 15 points

The **mlpack** implementation of range search is a dual-tree algorithm; when *kd*-trees are used, the leaf size of the tree can be changed. Depending on the characteristics of the dataset, a larger or smaller leaf size can provide faster computation. The leaf size is modifiable through the command-line interface, as shown below.

```
$ mlpack_range_search -r dataset.csv -n neighbors_out.csv -d distances_out.csv \
> -L 0.7 -U 0.8 -l 15 -v
[INFO ] Loading 'dataset.csv' as CSV data. Size is 3 x 1000.
[INFO ] Loaded reference data from 'dataset.csv' (3x1000).
[INFO ] Building reference tree...
[INFO ] Tree built.
[INFO ] Search for points in the range [0.7, 0.8] with dual-tree kd-tree
search...
[INFO ] Search complete.
[INFO ]
[INFO ] Execution parameters:
[INFO ]   distances_file: distances_out.csv
[INFO ]   help: false
[INFO ]   info: ""
[INFO ]   input_model_file: ""
[INFO ]   leaf_size: 15
[INFO ]   max: 0.8
[INFO ]   min: 0.7
[INFO ]   naive: false
[INFO ]   neighbors_file: neighbors_out.csv
[INFO ]   output_model_file: ""
[INFO ]   query_file: ""
[INFO ]   random_basis: false
[INFO ]   reference_file: dataset.csv
[INFO ]   seed: 0
[INFO ]   single_mode: false
[INFO ]   tree_type: kd
[INFO ]   verbose: true
[INFO ]   version: false
[INFO ]
[INFO ] Program timers:
[INFO ]   loading_data: 0.006298s
[INFO ]   range_search/computing_neighbors: 0.411041s
[INFO ]   total_time: 0.539931s
[INFO ]   tree_building: 0.004695s
```

Further documentation on options should be found by using the `--help` option.

26.4 The 'RangeSearch' class

The 'RangeSearch' class is an extensible template class which allows a high level of flexibility. However, all of the template arguments have default parameters, allowing a user to simply use 'RangeSearch<>' for simple usage without worrying about the exact necessary template parameters.

The class bears many similarities to the **NeighborSearch** (p. 151) class; usage generally consists of calling the constructor with one or two datasets, and then calling the 'Search()' method to perform the actual range search.

The 'Search()' method stores the results in two vector-of-vector objects. This is necessary because each query point may have a different number of neighbors in the specified distance range. The structure of those two objects is very similar to the output files `--neighbors_file` and `--distances_file` for the CLI interface (see above). A handful of examples of simple usage of the RangeSearch class are given below.

26.4.1 Distance less than 2.0 on a single dataset

```
#include <mlpack/methods/range_search/range_search.hpp>

using namespace mlpack::range;

// Our dataset matrix, which is column-major.
extern arma::mat data;

RangeSearch<> a(data);

// The vector-of-vector objects we will store output in.
std::vector<std::vector<size_t> > resultingNeighbors;
std::vector<std::vector<double> > resultingDistances;

// The range we will use.
math::Range r(0.0, 2.0); // [0.0, 2.0].

a.Search(r, resultingNeighbors, resultingDistances);
```

The output of the search is stored in `resultingNeighbors` and `resultingDistances`.

26.4.2 Range [3.0, 4.0] on a query and reference dataset

```
#include <mlpack/methods/range_search/range_search.hpp>

using namespace mlpack::range;

// Our dataset matrices, which are column-major.
extern arma::mat queryData, referenceData;

RangeSearch<> a(referenceData);

// The vector-of-vector objects we will store output in.
std::vector<std::vector<size_t> > resultingNeighbors;
std::vector<std::vector<double> > resultingDistances;

// The range we will use.
math::Range r(3.0, 4.0); // [3.0, 4.0].

a.Search(queryData, r, resultingNeighbors, resultingDistances);
```

26.4.3 Naive (exhaustive) search for distance greater than 5.0 on one dataset

This example uses the $O(n^2)$ naive search (not the tree-based search).

```
#include <mlpack/methods/range_search/range_search.hpp>

using namespace mlpack::range;

// Our dataset matrix, which is column-major.
extern arma::mat dataset;

// The 'true' option indicates that we will use naive calculation.
RangeSearch<> a(dataset, true);

// The vector-of-vector objects we will store output in.
std::vector<std::vector<size_t> > resultingNeighbors;
std::vector<std::vector<double> > resultingDistances;

// The range we will use. The upper bound is DBL_MAX.
math::Range r(5.0, DBL_MAX); // [5.0, inf).

a.Search(r, resultingNeighbors, resultingDistances);
```

Needless to say, naive search can be very slow...

26.5 The extensible 'RangeSearch' class

Similar to the **NeighborSearch** class (p. 151), the **RangeSearch** class is very extensible, having the following template arguments:

```
template<typename MetricType = metric::EuclideanDistance,
        typename MatType = arma::mat,
        template<typename TreeMetricType,
                typename TreeStatType,
                typename TreeMatType> class TreeType = tree::KDTree>
class RangeSearch;
```

By choosing different components for each of these template classes, a very arbitrary range searching object can be constructed.

26.5.1 MetricType policy class

The **MetricType** policy class allows the range search to take place in any arbitrary metric space. The **mlpack::metric::LMetric** (p. 1569) class is a good example implementation. A **MetricType** class must provide the following functions:

```
// Empty constructor is required.
MetricType();

// Compute the distance between two points.
template<typename VecType>
double Evaluate(const VecType& a, const VecType& b);
```

Internally, the **RangeSearch** class keeps an instantiated **MetricType** class (which can be given in the constructor). This is useful for a metric like the Mahalanobis distance (**mlpack::metric::MahalanobisDistance** (p. 1572)), which must store state (the covariance matrix). Therefore, you can write a non-static **MetricType** class and use it seamlessly with **RangeSearch**.

26.5.2 MatType policy class

The **MatType** template parameter specifies the type of data matrix used. This type must implement the same operations as an Armadillo matrix, and so standard choices are **arma::mat** and **arma::sp_mat**.

26.5.3 TreeType policy class

The **RangeSearch** class also allows a custom tree to be used. The **TreeType** policy is also used elsewhere in **mlpack** and is documented more thoroughly [here](#) (p. 179).

Typical choices might include **mlpack::tree::KDTree** (p. 453) (the default), **mlpack::tree::BallTree** (p. 451), **mlpack::tree::RTree** (p. 459), **mlpack::tree::RStarTree** (p. 459), or **mlpack::tree::StandardCoverTree** (p. 460). Below is an example that uses the **RangeSearch** class with an R-tree:

```
// Construct a RangeSearch object with ball bounds.
RangeSearch<
    metric::EuclideanDistance,
    arma::mat,
    tree::RTree
> rangeSearch(dataset);
```

For further information on trees, including how to write your own tree for use with **RangeSearch** and other **mlpack** methods, see the **TreeType policy documentation** (p. 179).

26.6 Further documentation

For further documentation on the RangeSearch class, consult the **complete API documentation** (p. 1754).

Chapter 27

Tutorials

27.1 Quickstart Tutorials

These tutorials give very quick "getting started" examples that you can use to get started with mlpack in different languages.

- **mlpack in Python quickstart guide** (p. 59)
- **mlpack command-line quickstart guide** (p. 31)

27.2 Introductory Tutorials

These tutorials introduce the basic concepts of working with mlpack, aimed at developers who want to use and contribute to mlpack but are not sure where to start.

- **Building mlpack From Source** (p. 21)
- **Building mlpack From Source on Windows** (p. 27)
- **File formats and loading data in mlpack** (p. 43)
- **Matrices in mlpack** (p. 57)
- **Writing an mlpack binding** (p. 53)
- **mlpack Timers** (p. 69)
- **Simple Sample mlpack Programs** (p. 63)
- **Sample C++ ML App for Windows** (p. 65)

27.3 Method-specific Tutorials

These tutorials introduce the various methods mlpack offers, aimed at users who want to get started quickly. These tutorials start with simple examples and progress to complex, extensible uses.

- **NeighborSearch tutorial (k-nearest-neighbors)** (p. 151)
- **Linear/ridge regression tutorial (mlpack_linear_regression)** (p. 143)
- **RangeSearch tutorial (mlpack_range_search)** (p. 159)
- **Density Estimation Tree (DET) tutorial** (p. 111)
- **K-Means tutorial (kmeans)** (p. 131)
- **Fast max-kernel search tutorial (fastmks)** (p. 121)
- **EMST Tutorial** (p. 117)
- **Alternating Matrix Factorization tutorial** (p. 73)
- **Collaborative filtering tutorial** (p. 103)
- **Approximate furthest neighbor search (mlpack_approx_kfn) tutorial** (p. 87)
- **Neural Network tutorial** (p. 77)

27.4 Advanced Tutorials

These tutorials discuss some of the more advanced functionality contained in mlpack.

- **mlpack automatic bindings to other languages** (p. 3)
- **Cross-Validation** (p. 35)
- **Hyper-Parameter Tuning** (p. 49)

27.5 Policy Class Documentation

mlpack uses templates to achieve its genericity and flexibility. Some of the template types used by mlpack are common across multiple machine learning algorithms. The links below provide documentation for some of these common types.

- **The MetricType policy in mlpack** (p. 177)
- **The KernelType policy in mlpack** (p. 173)
- **The TreeType policy in mlpack** (p. 179)

Chapter 28

The ElemType policy in mlpack

28.1 Overview

mlpack algorithms should be as generic as possible. Often this means allowing arbitrary metrics or kernels to be used, but this also means allowing any type of data point to be used. This means that **mlpack** classes should support `float`, `double`, and other observation types. Some algorithms support this through the use of a `MatType` template parameter; others will have their own template parameter, `ElemType`.

The `ElemType` template parameter can take any value that can be used by Armadillo (or, specifically, classes like `arma::Mat<>` and others); this encompasses the types

- `double`
- `float`
- `int`
- `unsigned int`
- `std::complex<double>`
- `std::complex<float>`

and other primitive numeric types. Note that Armadillo does not support some integer types for functionality such as matrix decompositions or other more advanced linear algebra. This means that when these integer types are used, some algorithms may fail with Armadillo error messages indicating that those types cannot be used.

28.2 note for developers

If the class has a `MatType` template parameter, `ElemType` can be easily defined as below:

```
typedef typename MatType::elem_type ElemType;
```

and otherwise a template parameter with the name `ElemType` can be used. It is generally a good idea to expose the element type somehow for use by other classes.

Chapter 29

The FunctionType policy in mlpack

29.1 Overview

To represent the various types of loss functions encountered in machine learning problems, mlpack provides the `FunctionType` template parameter in the optimizer interface. The various optimizers available in the core library rely on this policy to gain the necessary information required by the optimizing algorithm.

The `FunctionType` template parameter required by the `Optimizer` class can have additional requirements imposed on it, depending on the type of optimizer used.

29.2 Interface requirements

The most basic requirements for the `FunctionType` parameter are the implementations of two public member functions, with the following interface and semantics

```
// Evaluate the loss function at the given coordinates.
double Evaluate(const arma::mat& coordinates);

// Evaluate the gradient at the given coordinates, where 'gradient' is an
// output parameter for the required gradient.
void Gradient(const arma::mat& coordinates, arma::mat& gradient);
```

Optimizers like SGD and RMSProp require a `DecomposableFunctionType` having the following requirements

```
// Return the number of functions. In a data-dependent function, this would
// return the number of points in the dataset.
size_t NumFunctions();

// Evaluate the 'i' th loss function. For example, for a data-dependent
// function, Evaluate(coordinates, 0) should evaluate the loss function at the
// first point in the dataset.
double Evaluate(const arma::mat& coordinates, const size_t i);
```

```
// Evaluate the gradient of the 'i' th loss function at the given coordinates,
// where 'gradient' is an output parameter for the required gradient.
void Gradient(const arma::mat& coordinates, const size_t i, arma::mat& gradient);
```

ParallelSGD optimizer requires a SparseFunctionType interface. SparseFunctionType requires the gradient to be in a sparse matrix (arma::sp_mat), as ParallelSGD, implemented with the HOGWILD! scheme of unsynchronised updates, is expected to be relevant only in situations where the individual gradients are sparse. So, the interface requires function with the following signatures

```
// Return the number of functions. In a data-dependent function, this would
// return the number of points in the dataset.
size_t NumFunctions();
```

```
// Evaluate the loss function at the given coordinates.
double Evaluate(const arma::mat& coordinates);
```

```
// Evaluate the (sparse) gradient of the 'i' th loss function at the given
// coordinates, where 'gradient' is an output parameter for the required
// gradient.
void Gradient(const arma::mat& coordinates, const size_t i, arma::sp_mat& gradient);
```

The SCD optimizer requires a ResolvableFunctionType interface, to calculate partial gradients with respect to individual features. The optimizer requires the decision variable to be arranged in a particular fashion to allow for disjoint updates. The features should be arranged columnwise in the decision variable. For example, in Softmax←RegressionFunction the decision variable has size numClasses x featureSize (+ 1 if an intercept also needs to be fit). Similarly, for LogisticRegression, the decision variable is a row vector, with the number of columns determined by the dimensionality of the dataset.

The interface expects the following member functions from the function class

```
// Return the number of features in the decision variable.
size_t NumFeatures();
```

```
// Evaluate the loss function at the given coordinates.
double Evaluate(const arma::mat& coordinates);
```

```
// Evaluate the partial gradient of the loss function with respect to the 'j' th
// coordinate at the given coordinates, where 'gradient' is an output parameter
// for the required gradient. The 'gradient' matrix is supposed to be non-zero
// in the jth column, which contains the relevant partial gradient.
void PartialGradient(const arma::mat& coordinates, const size_t j, arma::sp_mat& gradient);
```

Chapter 30

The KernelType policy in mlpack

30.1 Table of Contents

- **Introduction to the KernelType policy** (p. 173)
- **The KernelTraits trait class** (p. 175)
- **List of kernels and classes that use a KernelType** (p. 175)

30.2 Introduction to the KernelType policy

'Kernel methods' make up a large class of machine learning techniques. Each of these methods is characterized by its dependence on a **kernel function**. In rough terms, a kernel function is a general notion of similarity between two points, with its value large when objects are similar and its value small when objects are dissimilar (note that this is not the only interpretation of what a kernel is).

A kernel (or 'Mercer kernel') $\mathcal{K}(\cdot, \cdot)$ takes two objects as input and returns some sort of similarity value. The specific details and properties of kernels are outside the scope of this documentation; for a better introduction to kernels and kernel methods, there are numerous better resources available, including <http://www.eric-kim.net/eric-kim-net/posts/1/kernel-trick.html> "Eric Kim's tutorial".

mlpack implements a number of kernel methods and, accordingly, each of these methods allows arbitrary kernels to be used via the `KernelType` template parameter. Like the **MetricType policy** (p. 177), the requirements are quite simple: a class implementing the `KernelType` policy must have

- an `Evaluate()` function
- a default constructor

The signature of the `Evaluate()` function is straightforward:

```
template<typename VecTypeA, typename VecTypeB>
double Evaluate(const VecTypeA& a, const VecTypeB& b);
```

The function takes two vector arguments, a and b , and returns a `double` that is the evaluation of the kernel between the two arguments. So, for a particular kernel $\mathcal{K}(\cdot, \cdot)$, the `Evaluate()` function should return $\mathcal{K}(a, b)$.

The arguments a and b , of types `VecTypeA` and `VecTypeB`, respectively, will be an Armadillo-like vector type (usually `arma::vec`, `arma::sp_vec`, or similar). In general it should be valid to assume that `VecTypeA` is a class with the same API as `arma::vec`.

Note that for kernels that do not hold any state, the `Evaluate()` method can be marked as `static`.

Overall, the `KernelType` template policy is quite simple (much like the **MetricType** policy (p.177)). Below is an example kernel class, which outputs 1 if the vectors are close and 0 otherwise.

```
class ExampleKernel
{
    // Default constructor is required.
    ExampleKernel() { }

    // The example kernel holds no state, so we can mark Evaluate() as static.
    template<typename VecTypeA, typename VecTypeB>
    static double Evaluate(const VecTypeA& a, const VecTypeB& b)
    {
        // Get how far apart the vectors are (using the Euclidean distance).
        const double distance = arma::norm(a - b);

        if (distance < 0.05) // Less than 0.05 distance is "close".
            return 1;
        else
            return 0;
    }
};
```

Then, this kernel may be easily used inside of mlpack algorithms. For instance, the code below runs kernel PCA (**mlpack::kpca::KernelPCA** (p.1507)) on a random dataset using the `ExampleKernel`. The results are saved to a file called `results.csv`. (Note that this is simply an example to demonstrate usage, and this example kernel isn't actually likely to be useful in practice.)

```
#include <mlpack/core.hpp>
#include <mlpack/methods/kernel_pca/kernel_pca.hpp>
#include "example_kernel.hpp" // Contains the ExampleKernel class.

using namespace mlpack;
using namespace mlpack::kpca;
using namespace arma;

int main()
{
    // Generate the random dataset; 10 dimensions, 5000 points.
    mat dataset = randu<mat>(10, 5000);

    // Instantiate the KernelPCA object with the ExampleKernel kernel type.
    KernelPCA<ExampleKernel> kpca;

    // The dataset will be transformed using kernel PCA with the example kernel to
    // contain only 2 dimensions.
    kpca.Apply(dataset, 2);

    // Save the results to 'results.csv'.
    data::Save(dataset, "results.csv");
}
```

30.3 The KernelTraits trait class

Some algorithms that use kernels can specialize if the kernel fulfills some certain conditions. An example of a condition might be that the kernel is shift-invariant or that the kernel is normalized. In the case of fast max-kernel search (`mlpack::fastmks::FastMKS` (p. 1280)), the computation can be accelerated if the kernel is normalized. For this reason, the `KernelTraits` trait class exists. This allows a kernel to specify via a `const static bool` when these types of conditions are satisfied. **Note that a `KernelTraits` class is not required**, but may be helpful.

The `KernelTraits` trait class is a template class that takes a `KernelType` as a parameter, and exposes `const static bool` values that depend on the kernel. Setting these values is achieved by specialization. The code below provides an example, specializing `KernelTraits` for the `ExampleKernel` from earlier:

```
template<>
class KernelTraits<ExampleKernel>
{
public:
    const static bool IsNormalized = true;
};
```

At this time, there is only one kernel trait that is used in `mlpack` code:

- `IsNormalized` (defaults to `false`): if $K(x, x) = 1 \forall x$, then the kernel is normalized and this should be set to `true`.

30.4 List of kernels and classes that use a `KernelType`

`mlpack` comes with a number of pre-written kernels that satisfy the `KernelType` policy:

- `mlpack::kernel::LinearKernel` (p. 1446)
- `mlpack::kernel::ExampleKernel` (p. 1421) – an example kernel with more documentation
- `mlpack::kernel::GaussianKernel` (p. 1424)
- `mlpack::kernel::HyperbolicTangentKernel` (p. 1430)
- `mlpack::kernel::EpanechnikovKernel` (p. 1417)
- `mlpack::kernel::CosineDistance` (p. 1416)
- `mlpack::kernel::LaplacianKernel` (p. 1442)
- `mlpack::kernel::PolynomialKernel` (p. 1451)
- `mlpack::kernel::TriangularKernel` (p. 1461)
- `mlpack::kernel::SphericalKernel` (p. 1458)
- `mlpack::kernel::PSpectrumStringKernel` (p. 1454) – operates on strings, not vectors

These kernels (or a custom kernel) may be used in a variety of `mlpack` methods:

- `mlpack::kpca::KernelPCA` (p. 1507) - kernel principal components analysis
- `mlpack::fastmks::FastMKS` (p. 1280) - fast max-kernel search
- `mlpack::kernel::NystroemMethod` (p. 1448) - the Nystroem method for sampling
- `mlpack::metric::IPMetric` (p. 1565) - a metric built on a kernel

Chapter 31

The MetricType policy in mlpack

Many machine learning methods operate with some sort of metric, and often, this metric can be any arbitrary metric. For instance, consider the problem of nearest neighbor search; one can find the nearest neighbor of a point with respect to the standard Euclidean distance, or the Manhattan (city-block) distance. The actual search techniques, though, remain the same. And this is true of many machine learning methods: the specific metric that is used can be any valid metric.

mlpack algorithms, when possible, allow the use of an arbitrary metric via the use of the `MetricType` template parameter. Any metric passed as a `MetricType` template parameter will need to have

- an `Evaluate` function
- a default constructor.

The signature of the `Evaluate` function is straightforward:

```
template<typename VecTypeA, typename VecTypeB>
double Evaluate(const VecTypeA& a, const VecTypeB& b);
```

The function takes two vector arguments, `a` and `b`, and returns a `double` that is the evaluation of the metric between the two arguments. So, for a particular metric $d(\cdot, \cdot)$, the `Evaluate()` function should return $d(a, b)$.

The arguments `a` and `b`, of types `VecTypeA` and `VecTypeB`, respectively, will be an Armadillo-like vector type (usually `arma::vec`, `arma::sp_vec`, or similar). In general it should be valid to assume that `VecTypeA` is a class with the same API as `arma::vec`.

Note that for metrics that do not hold any state, the `Evaluate()` method can be marked as `static`.

Overall, the `MetricType` template policy is quite simple (much like the **The KernelType policy in mlpack** (p. 173) `KernelType` policy). Below is an example metric class, which implements the L2 distance:

```
class ExampleMetric
{
    // Default constructor is required.
    ExampleMetric() {}

    // The example metric holds no state, so we can mark Evaluate() as static.
    template<typename VecTypeA, typename VecTypeB>
    static double Evaluate(const VecTypeA& a, const VecTypeB& b)
    {
        // Return the L2 norm of the difference between the points, which is the
        // same as the L2 distance.
        return arma::norm(a - b);
    }
};
```

Then, this metric can easily be used inside of other mlpack algorithms. For example, the code below runs range search on a random dataset with the `ExampleKernel`, by instantiating a `mlpack::range::RangeSearch` (p. 1754) object that uses the `ExampleKernel`. Then, the number of results are printed. The `RangeSearch` class takes three template parameters: `MetricType`, `MatType`, and `TreeType`. (All three have defaults, so we will just leave `MatType` and `TreeType` to their defaults.)

```
#include <mlpack/core.hpp>
#include <mlpack/methods/range_search/range_search.hpp>
#include "example_metric.hpp" // A file that contains ExampleKernel.

using namespace mlpack;
using namespace mlpack::range;
using namespace std;

int main()
{
  // Create a random dataset with 10 dimensions and 5000 points.
  arma::mat data = arma::randu<arma::mat>(10, 5000);

  // Instantiate the RangeSearch object with the ExampleKernel.
  RangeSearch<ExampleKernel> rs(data);

  // These vectors will store the results.
  vector<vector<size_t>> neighbors;
  vector<vector<double>> distances;

  // Create a random 10-dimensional query point.
  arma::vec query = arma::randu<arma::vec>(10);

  // Find those points with distance (according to ExampleMetric) between 1 and
  // 2 from the query point.
  rs.Search(query, math::Range(1.0, 2.0), neighbors, distances);

  // Now, print the number of points inside the desired range. We know that
  // neighbors and distances will have length 1, since there was only one query
  // point.
  cout << neighbors[0].size() << " points within the range [1.0, 2.0] of the "
       << "query point!" << endl;
}
```

mlpack comes with a number of pre-written metrics that satisfy the `MetricType` policy:

- `mlpack::metric::ManhattanDistance` (p. 427)
- `mlpack::metric::EuclideanDistance` (p. 427)
- `mlpack::metric::ChebyshevDistance` (p. 427)
- `mlpack::metric::MahalanobisDistance` (p. 1572)
- `mlpack::metric::LMetric` (p. 1569) (for arbitrary L-metrics)
- `mlpack::metric::IPMetric` (p. 1565) (requires a `KernelType` (p. 173) parameter)

Chapter 32

The TreeType policy in mlpack

32.1 Introduction

Trees are an important data structure in mlpack and are used in a number of the machine learning algorithms that mlpack implements. Often, the use of trees can allow significant acceleration of an algorithm; this is generally done by pruning away large parts of the tree during computation.

Most mlpack algorithms that use trees are not tied to a specific tree but instead allow the user to choose a tree via the `TreeType` template parameter. Any tree passed as a `TreeType` template parameter will need to implement a certain set of functions. In addition, a tree may optionally specify some traits about itself with the `TreeTraits` trait class.

This document aims to clarify the abstractions underlying mlpack trees, list and describe the required functionality of the `TreeType` policy, and point users towards existing types of trees. A table of contents is below:

- **Introduction** (p. 179)
- **What is a tree?** (p. 180)
- **Template parameters required by the TreeType policy** (p. 181)
- **The TreeType API** (p. 182)
- **Rigorous API documentation** (p. 184)
 - **Template parameters** (p. 185)
 - **Constructors and destructors** (p. 186)
 - **Basic tree functionality** (p. 187)
 - **Complex tree functionality and bounds** (p. 188)
 - **Serialization** (p. 190)
- **The TreeTraits trait class** (p. 190)
- **A list of trees in mlpack and more information** (p. 191)

Although this document is long, there may still be errors and unclear areas. If you are having trouble understanding anything, please get in touch on Github or on the mailing list and someone will help you (and possibly update the documentation afterwards).

32.2 What is a tree?

In `mlpack`, we assume that we have some sort of data matrix, which might be sparse or dense (that is, it could be of type `arma::mat` or `arma::sp_mat`, or any variant that implements the Armadillo API). This data matrix corresponds to a collection of points in some space (usually a Euclidean space). A tree is a way of organizing this data matrix in a hierarchical manner—so, points that are nearby should lie in similar nodes.

We can rigorously define what a tree is, using the definition of **space tree** introduced in the following paper:

R.R. Curtin, W.B. March, P. Ram, D.V. Anderson, A.G. Gray, and C.L. Isbell Jr., "Tree-independent dual-tree algorithms," in Proceedings of the 30th International Conference on Machine Learning (ICML '13), pp. 1435–1443, 2013.

The definition is:

A **space tree** on a dataset $S \in \mathcal{R}^{N \times d}$ is an undirected, connected, acyclic, rooted simple graph with the following properties:

- Each node (or vertex) holds a number of points (possibly zero) and is connected to one parent node and a number of child nodes (possibly zero).
- There is one node in every space tree with no parent; this is the root node of the tree.
- Each point in S is contained in at least one node.
- Each node corresponds to some subset of \mathcal{R}^d that contains each point in the node and also the subsets that correspond to each child of the node.

This is really a quite straightforward definition: a tree is hierarchical, and each node corresponds to some region of the input space. Each node may have some number of children, and may hold some number of points. However, there is an important terminology distinction to make: the term **points held by a node** has a different meaning than the term **descendant points held by a node**. The points held in a node are just that—points held only in the node. The descendant points of a node are the combination of the points held in a node with the points held in the node's children and the points held in the node's children's children (and so forth). For the purposes of clarity in all discussions about trees, care is taken to differentiate the terms "descendant point" and "point".

Now, it's also important to note that a point does not *need* to hold any children, and that a node *can* hold the same points as its children (or its parent). Some types of trees do this. For instance, each node in the cover tree holds only one point, and may have a child that holds the same point. As another example, the k *d*-tree holds its points only in the leaves (at the bottom of the tree). More information on space trees can be found in either the "Tree-independent dual-tree algorithms" paper or any of the related literature.

So there is a huge amount of possible variety in the types of trees that can fall into the class of *space trees*. Therefore, it's important to treat them abstractly, and the `TreeType` policy allows us to do just that. All we need to remember is that a node in a tree can be represented as the combination of some points held in the node, some child nodes, and some geometric structure that represents the space that all of the descendant points fall into (this is a restatement of the fourth part of the definition).

32.3 Template parameters required by the TreeType policy

Most everything in mlpack is decomposed into a series of configurable template parameters, and trees are no exception. In order to ease usage of high-level mlpack algorithms, each `TreeType` itself must be a template class taking three parameters:

- `MetricType` – the underlying metric that the tree will be built on (see [the MetricType policy documentation](#) (p. 177))
- `StatisticType` – holds any auxiliary information that individual algorithms may need
- `MatType` – the type of the matrix used to represent the data

The reason that these three template parameters are necessary is so that each `TreeType` can be used as a template template parameter, which can radically simplify the required syntax for instantiating mlpack algorithms. By using template template parameters, a user needs only to write

```
// The RangeSearch class takes a MetricType and a TreeType template parameter.

// This code instantiates RangeSearch with the ManhattanDistance and a
// QuadTree. Note that the QuadTree itself is a template, and takes a
// MetricType, StatisticType, and MatType, just like the policy requires.

// This example ignores the constructor parameters, for the sake of simplicity.
RangeSearch<ManhattanDistance, QuadTree> rs(...);
```

as opposed to the far more complicated alternative, where the user must specify the values of each template parameter of the tree type:

```
// This is a much worse alternative, where the user must specify the template
// arguments of their tree.
RangeSearch<ManhattanDistance,
            QuadTree<ManhattanDistance, EmptyStatistic, arma::mat>> rs(...);
```

Unfortunately, the price to pay for this user convenience is that *every* `TreeType` must have three template parameters, and they must be in exactly that order. Fortunately, there is an additional benefit: we are guaranteed that the tree is built using the same metric as the method (that is, a user can't specify different metric types to the algorithm and to the tree, which they can without template template parameters).

There are two important notes about this:

- Not every possible input of `MetricType`, `StatisticType`, and/or `MatType` necessarily need to be valid or work correctly for each type of tree. For instance, the `QuadTree` is limited to Euclidean metrics and will not work otherwise. Either compile-time static checks or detailed documentation can help keep users from using invalid combinations of template arguments.
- Some types of trees have more template parameters than just these three. One example is the generalized binary space tree, where the bounding shape of each node is easily made into a fourth template parameter (the `BinarySpaceTree` class calls this the `BoundType` parameter), and the procedure used to split a node is easily made into a fifth template parameter (the `BinarySpaceTree` class calls this the `SplitType` parameter). However, the syntax of template template parameters *requires* that the class only has the correct number of template parameters—no more, no less. Fortunately, C++11 allows template typedefs, which can be used to provide partial specialization of template classes:

```
// This is the definition of the BinarySpaceTree class, which has five template
// parameters.
template<typename MetricType,
        typename StatisticType,
        typename MatType,
        typename BoundType,
        typename SplitType>
class BinarySpaceTree;

// The 'using' keyword gives us a template typedef, so we can define the
// MeanSplitKDTree template class, which has three parameters and is a valid
// TreeType policy class.
template<typename MetricType, typename StatisticType, typename MatType>
using MeanSplitKDTree = BinarySpaceTree<MetricType,
                                       StatisticType,
                                       MatType,
                                       HRectBound<MetricType>
                                       MeanSplit<BoundType, MetricType>>;
```

Now, the MeanSplitKDTree class has only three template parameters and can be used as a TreeType policy class in various mlpack algorithms. Many types of trees in mlpack have more than three template parameters and rely on template typedefs to provide simplified TreeType interfaces.

32.4 The TreeType API

As a result of the definition of *space tree* in the previous section, a simplified API presents itself quite easily. However, more complex functionality is often necessary in mlpack, so this leads to more functions being necessary for a class to satisfy the TreeType policy. Combining this with the template parameters required for trees given in the previous section gives us the complete API required for a class implementing the TreeType policy. Below is the minimal set of functions required with minor documentation for each function. (More extensive documentation and explanation is given afterwards.)

```
// The three template parameters will be supplied by the user, and are detailed
// in the previous section.
template<typename MetricType,
        typename StatisticType,
        typename MatType>
class ExampleTree
{
public:

    // This batch constructor does not modify the dataset, and builds the entire
    // tree using a default-constructed MetricType.
    ExampleTree(const MatType& data);

    // This batch constructor does not modify the dataset, and builds the entire
    // tree using the given MetricType.
    ExampleTree(const MatType& data, MetricType& metric);

    // Initialize the tree from a given boost::serialization archive. SFINAE (the
    // second argument) is necessary to ensure that the archive is loading, not
    // saving.
    template<typename Archive>
    ExampleTree(
        Archive& ar,
        const typename boost::enable_if<typename Archive::is_loading>::type* = 0);

    // Release any resources held by the tree.
    ~ExampleTree();

    // ////////////////////////////////// //
    // // Basic functionality // //
    // ////////////////////////////////// //

    // Get the dataset that the tree is built on.
    const MatType& Dataset();

    // Get the metric that the tree is built with.
```

```

MetricType& Metric();

// Get/modify the StatisticType for this node.
StatisticType& Stat();

// Return the parent of the node, or NULL if this is the root.
ExampleTree* Parent();

// Return the number of children held by the node.
size_t NumChildren();
// Return the i'th child held by the node.
ExampleTree& Child(const size_t i);

// Return the number of points held in the node.
size_t NumPoints();
// Return the index of the i'th point held in the node.
size_t Point(const size_t i);

// Return the number of descendant nodes of this node.
size_t NumDescendantNodes();
// Return the i'th descendant node of this node.
ExampleTree& DescendantNode(const size_t i);

// Return the number of descendant points of this node.
size_t NumDescendants();
// Return the index of the i'th descendant point of this node.
size_t Descendant(const size_t i);

// Store the center of the bounding region of the node in the given vector.
void Center(arma::vec& center);

// //////////////////////////////////////// //
// // More complex distance-related functionality // //
// //////////////////////////////////////// //

// Return the distance between the center of this node and the center of
// its parent.
double ParentDistance();

// Return an upper bound on the furthest possible distance between the
// center of the node and any point held in the node.
double FurthestPointDistance();

// Return an upper bound on the furthest possible distance between the
// center of the node and any descendant point of the node.
double FurthestDescendantDistance();

// Return a lower bound on the minimum distance between the center and any
// edge of the node's bounding shape.
double MinimumBoundDistance();

// Return a lower bound on the minimum distance between the given point and
// the node.
template<typename VecType>
double MinDistance(VecType& point);

// Return a lower bound on the minimum distance between the given node and
// this node.
double MinDistance(ExampleTree& otherNode);

// Return an upper bound on the maximum distance between the given point and
// the node.
template<typename VecType>
double MaxDistance(VecType& point);

// Return an upper bound on the maximum distance between the given node and
// this node.
double MaxDistance(ExampleTree& otherNode);

// Return the combined results of MinDistance() and MaxDistance().
template<typename VecType>
math::Range RangedDistance(VecType& point);

// Return the combined results of MinDistance() and MaxDistance().
math::Range RangeDistance(ExampleTree& otherNode);

// //////////////////////////////////////// //
// // Serialization (loading/saving) // //
// //////////////////////////////////////// //

// Return a string representation of the tree.

```

```

std::string ToString() const;

// Serialize the tree (load from the given archive / save to the given
// archive, depending on its type).
template<typename Archive>
void Serialize(Archive& ar, const unsigned int version);

protected:
// A default constructor; only meant to be used by boost::serialization. This
// must be protected so that boost::serialization will work; it does not need
// to return a valid tree.
ExampleTree();

// Friend access must be given for the default constructor.
friend class boost::serialization::access;
};

```

Although this is significantly more complex than the four-item definition of `space tree*` might suggest, it turns out many of these methods are not difficult to implement for most reasonable tree types. It is also important to realize that this is a *minimum* API; you may implement more complex tree types at your leisure (and you may include more template parameters too, though you will have to use template typedefs to provide versions with three parameters; see **the previous section** (p. 181)).

Before diving into the detailed documentation for each function, let us consider a few important points about the implications of this API:

- **Trees are not default-constructible** and should not (in general) provide a default constructor. This helps prevent invalid trees. In general, any instantiated mlpack object should be valid and ready to use—and a tree built on no points is not valid or ready to use.
- **Trees only need to provide batch constructors.** Although many tree types do have algorithms for incremental insertions, in mlpack this is not required because the tree-based algorithms that mlpack implements generally assume fully-built, non-modifiable trees. For this purpose, batch construction is perfectly sufficient. (It's also worth pointing out that for some types of trees, like kd-trees, the cost of a handful of insertions often outweighs the cost of completely rebuilding the tree.)
- **Trees must provide a number of distance bounding functions.** The utility of trees generally stems from the ability to place quick bounds on distance-related quantities. For instance, if all the descendant points of a node are bounded by a ball of radius λ and the center of the node is a point c , then the minimum distance between some point p and any descendant point of the node is equal to the distance between p and c minus the radius λ : $d(p, c) - \lambda$. This is a fast calculation, and (usually) provides a decent bound on the minimum distance between p and any descendant point of the node.
- **Trees need to be able to be serialized.** mlpack uses the `boost::serialization` (p. 251) library for saving and loading objects. Trees—which can be a part of machine learning models—therefore must have the ability to be saved and loaded. Making this all work requires a protected constructor (part of the API) and generally makes it impossible to hold references instead of pointers internally, because if a tree is loaded from a file then it must own the dataset it is built on and the metric it uses (this also means that a destructor must exist for freeing these resources).

Now, we can consider each part of the API more rigorously.

32.5 Rigorous API documentation

This section is divided into five parts:

- **Template parameters** (p. 185)
- **Constructors and destructors** (p. 186)
- **Basic tree functionality** (p. 187)
- **Complex tree functionality and bounds** (p. 188)
- **Serialization** (p. 190)

32.5.1 Template parameters

An earlier section discussed the three different template parameters that are required by the `TreeType` policy.

The **MetricType** policy (p. 177) provides one method that will be useful for tree building and other operations:

```
// This function is required by the MetricType policy.
// Evaluate the metric between two points (which may be of different types).
template<typename VecTypeA, typename VecTypeB>
double Evaluate(const VecTypeA& a, const VecTypeB& b);
```

Note that this method is not necessarily static, so a `MetricType` object should be held internally and its `Evaluate()` method should be called whenever the distance between two points is required. **It is generally a bad idea to hardcode any distance calculation in your tree.** This will make the tree unable to generalize to arbitrary metrics. If your tree must depend on certain assumptions holding about the metric (i.e. the metric is a Euclidean metric), then make that clear in the documentation of the tree, so users do not try to use the tree with an inappropriate metric.

The second template parameter, `StatisticType`, is for auxiliary information that is required by certain algorithms. For instance, consider an algorithm which repeatedly uses the variance of the descendant points of a node. It might be tempting to add a `Variance()` method to the required `TreeType` API, but this quickly leads to code bloat (after all, the API already has quite enough functions as it is). Instead, it is better to create a `StatisticType` class which provides the `Variance()` method, and then call `Stat().Variance()` when the variance is required. This also holds true for cached data members.

Each node should have its own instance of a `StatisticType` class. The `StatisticType` must provide the following constructors:

```
// Default constructor required by the StatisticType policy.
StatisticType();

// This constructor is required by the StatisticType policy.
template<typename TreeType>
StatisticType(TreeType& node);
```

This constructor should be called with `(*this)` after the node is constructed (usually, this ends up being the last line in the constructor of a node).

The last template parameter is the `MatType` parameter. This is generally `arma::mat` or `arma::sp_mat`, but could be any Armadillo type, including matrices that hold data points of different precisions (such as `float` or even `int`). It generally suffices to write `MatType` assuming that `arma::mat` will be used, since the vast majority of the time this will be what is used.

32.5.2 Constructors and destructors

The `TreeType` API requires at least three constructors. Technically, it does not *require* a destructor, but almost certainly your tree class will be doing some memory management internally and should have one (though not always).

The first two constructors are variations of the same idea:

```
// This batch constructor does not modify the dataset, and builds the entire
// tree using a default-constructed MetricType.
ExampleTree(const MatType& data);

// This batch constructor does not modify the dataset, and builds the entire
// tree using the given MetricType.
ExampleTree(const MatType& data, MetricType& metric);
```

All that is required here is that a constructor is available that takes a dataset and optionally an instantiated metric. If no metric is provided, then it should be assumed that the `MetricType` class has a default constructor and a default-constructed metric should be used. The constructor *must* return a valid, fully-constructed, ready-to-use tree that satisfies the definition of *space tree* that was **given earlier** (p. 180).

It is possible to implement both these constructors as one by using `boost::optional`.

The third constructor requires the tree to be initializable from a `boost::serialization` (p. 251) archive:

```
// Initialize the tree from a given boost::serialization archive. SFINAE (the
// second argument) is necessary to ensure that the archive is loading, not
// saving.
template<typename Archive>
ExampleTree(
    Archive& ar,
    const typename boost::enable_if<typename Archive::is_loading>::type* = 0);
```

This has implications on how the tree must be stored. In this case, the dataset is *not yet loaded* and therefore the tree **may be required to have ownership of the data matrix**. This means that realistically the most reasonable way to represent the data matrix internally in a tree class is not with a reference but instead with a pointer. If this is true, then a destructor will be required:

```
// Release any resources held by the tree.
~ExampleTree();
```

and, if the data matrix is represented internally with a pointer, this destructor will need to release the memory for the data matrix (in the case that the tree was created via `boost::serialization` (p. 251)).

Note that these constructors are not necessarily the only constructors that a `TreeType` implementation can provide. One important example of when more constructors are useful is when the tree rearranges points internally; this might be desired for the sake of speed or memory optimization. But to do this with the required constructors would necessarily incur a copy of the data matrix, because the user will pass a `"const MatType&"`. One alternate solution is to provide a constructor which takes an rvalue reference to a `MatType`:

```
template<typename Archive>
ExampleTree(MatType&& data);
```

(and another overload that takes an instantiated metric), and then the user can use `std::move()` to build the tree without copying the data matrix, although the data matrix will be modified:

```
ExampleTree exTree(std::move(dataset));
```

It is, of course, possible to add even more constructors if desired.

32.5.3 Basic tree functionality

The basic functionality of a class implementing the `TreeType` API is quite straightforward and intuitive.

```
// Get the dataset that the tree is built on.
const MatType& Dataset();
```

This should return a `const` reference to the dataset the tree is built on. The fact that this function is required essentially means that each node in the tree must store a pointer to the dataset (this is not the only option, but it is the most obvious option).

```
// Get the metric that the tree is built with.
MetricType& Metric();
```

Each node must also store an instantiated metric or a pointer to one (note that this is required even for metrics that have no state and have a `static Evaluate()` function).

```
// Get/modify the StatisticType for this node.
StatisticType& Stat();
```

As discussed earlier, each node must hold a `StatisticType`; this is accessible through the `Stat()` function.

```
// Return the parent of the node, or NULL if this is the root.
ExampleTree* Parent();

// Return the number of children held by the node.
size_t NumChildren();
// Return the i'th child held by the node.
ExampleTree& Child(const size_t i);

// Return the number of points held in the node.
size_t NumPoints();
// Return the index of the i'th point held in the node.
size_t Point(const size_t i);

// Return the number of descendant nodes of this node.
size_t NumDescendantNodes();
// Return the i'th descendant node of this node.
ExampleTree& DescendantNode(const size_t i);

// Return the number of descendant points of this node.
size_t NumDescendants();
// Return the index of the i'th descendant point of this node.
size_t Descendant(const size_t i);
```

These functions are all fairly self-explanatory. Most algorithms will use the `Parent()`, `Children()`, `NumChildren()`, `Point()`, and `NumPoints()` functions, so care should be taken when implementing those functions to ensure they will be efficient. Note that `Point()` and `Descendant()` should return indices of points, so the actual points can be accessed by calling `"Dataset().col(Point(i))"` for some index `i` (or something similar).

An important note about the `Descendant()` function is that each descendant point should be unique. So if a node holds the point with index 6 and it has one child that holds the points with indices 6 and 7, then `NumDescendants()` should return 2, not 3. The ordering in which the descendants are returned can be arbitrary; so, `Descendant(0)` can return 6 or 7, and `Descendant(1)` should return the other index.

```
// Store the center of the bounding region of the node in the given vector.
void Center(arma::vec& center);
```

The last function, **Center()** (p. 410), should calculate the center of the bounding shape and store it in the given vector. So, for instance, if the tree is a ball tree, then the center is simply the center of the ball. Algorithm writers would be wise to try and avoid the use of **Center()** (p. 410) if possible, since it will necessarily cost a copy of a vector.

32.5.4 Complex tree functionality and bounds

A node in a tree should also be able to calculate various distance-related bounds; these are particularly useful in tree-based algorithms. Note that any of these bounds does not necessarily need to be maximally tight; generally it is more important that each bound can be easily calculated.

Details on each bounding function that the `TreeType` API requires are given below.

```
// Return the distance between the center of this node and the center of
// its parent.
double ParentDistance();
```

Remember that each node corresponds to some region in the space that the dataset lies in. For most tree types this shape is often something geometrically simple: a ball, a cone, a hyperrectangle, a slice, or something similar. The `ParentDistance()` function should return the distance between the center of this node's region and the center of the parent node's region.

In practice this bound is often used in dual-tree (or single-tree) algorithms to place an easy `MinDistance()` (or `MaxDistance()`) bound for a child node; the parent's `MinDistance()` (or `MaxDistance()`) function is called and then adjusted with `ParentDistance()` to provide a possibly loose but efficient bound on what the result of `MinDistance()` (or `MaxDistance()`) would be with the child.

```
// Return an upper bound on the furthest possible distance between the
// center of the node and any point held in the node.
double FurthestPointDistance();

// Return an upper bound on the furthest possible distance between the
// center of the node and any descendant point of the node.
double FurthestDescendantDistance();
```

It is often very useful to be able to bound the radius of a node, which is effectively what `FurthestDescendantDistance()` does. Often it is easiest to simply calculate and cache the furthest descendant distance at tree construction time. Some trees, such as the cover tree, are able to give guarantees that the points held in the node will necessarily be closer than the descendant points; therefore, the `FurthestPointDistance()` function is also useful.

It is permissible to simply have `FurthestPointDistance()` return the result of `FurthestDescendantDistance()`, and that will still be a valid bound, but depending on the type of tree it may be possible to have `FurthestPointDistance()` return a tighter bound.

```
// Return a lower bound on the minimum distance between the center and any
// edge of the node's bounding shape.
double MinimumBoundDistance();
```

This is, admittedly, a somewhat complex and weird quantity. It is one of the less important bounding functions, so it is valid to simply return 0...

The bound is a bound on the minimum distance between the center of the node and any edge of the shape that bounds all of the descendants of the node. So, if the bounding shape is a ball (as in a ball tree or a cover tree), then `MinimumBoundDistance()` should just return the radius of the ball. If the bounding shape is a hypercube (as in a generalized octree), then `MinimumBoundDistance()` should return the side length divided by two. If the bounding shape is a hyperrectangle (as in a kd-tree or a spill tree), then `MinimumBoundDistance()` should return half the side length of the hyperrectangle's smallest side.

```

// Return a lower bound on the minimum distance between the given point and
// the node.
template<typename VecType>
double MinDistance(VecType& point);

// Return a lower bound on the minimum distance between the given node and
// this node.
double MinDistance(ExampleTree& otherNode);

// Return an upper bound on the maximum distance between the given point and
// the node.
template<typename VecType>
double MaxDistance(VecType& point);

// Return an upper bound on the maximum distance between the given node and
// this node.
double MaxDistance(ExampleTree& otherNode);

// Return the combined results of MinDistance() and MaxDistance().
template<typename VecType>
math::Range RangeDistance(VecType& point);

// Return the combined results of MinDistance() and MaxDistance().
math::Range RangeDistance(ExampleTree& otherNode);

```

These six functions are almost without a doubt the most important functionality of a tree. Therefore, it is preferable that these methods be implemented as efficiently as possible, as they may potentially be called many millions of times in a tree-based algorithm. It is also preferable that these bounds be as tight as possible. In tree-based algorithms, these are used for pruning away work, and tighter bounds mean that more pruning is possible.

Of these six functions, there are only really two bounds that are desired here: the *minimum distance* between a node and an object, and the *maximum distance* between a node and an object. The object may be either a vector (usually `arma::vec`) or another tree node.

Consider the first case, where the object is a vector. The result of `MinDistance()` needs to be less than or equal to the true minimum distance, which could be calculated as below:

```

// We assume that we have a vector 'vec', and a tree node 'node'.
double trueMinDist = DBL_MAX;
for (size_t i = 0; i < node.NumDescendants(); ++i)
{
    const double dist = node.Metric().Evaluate(vec,
        node.Dataset().col(node.Descendant(i)));
    if (dist < trueMinDist)
        trueMinDist = dist;
}
// At the end of the loop, trueMinDist will hold the true minimum distance
// between 'vec' and any descendant point of 'node'.

```

Often the bounding shape of a node will allow a quick calculation that will make a reasonable bound. For instance, if the node's bounding shape is a ball with radius r and center ctr , the calculation is simply `"(node.Metric().Evaluate(vec, ctr) - r)".` Usually a good `MinDistance()` or `MaxDistance()` function will make only one call to the `Evaluate()` function of the metric.

The `RangeDistance()` function allows a way for both bounds to be calculated at once. It is possible to implement this as a call to `MinDistance()` followed by a call to `MaxDistance()`, but this may incur more metric `Evaluate()` calls than necessary. Often calculating both bounds at once can be more efficient and can be done with fewer `Evaluate()` calls than calling both `MinDistance()` and `MaxDistance()`.

32.5.5 Serialization

The last two public functions that the `TreeType` API requires are related to serialization and printing.

```
// Return a string representation of the tree.
std::string ToString() const;
```

There are few restrictions on the precise way that the `ToString()` function should operate, but generally it should behave similarly to the `ToString()` function in other mlpack methods. Generally, a user will call `ToString()` when they want to inspect the object and see what it looks like. For a tree, printing the entire tree may be way more information than the user was expecting, so it may be a better option to print either only the node itself or the node plus one or two levels of children.

```
// Serialize the tree (load from the given archive / save to the given
// archive, depending on its type).
template<typename Archive>
void Serialize(Archive& ar, const unsigned int version);

protected:
// A default constructor; only meant to be used by boost::serialization. This
// must be protected so that boost::serialization will work; it does not need
// to return a valid tree.
ExampleTree();

// Friend access must be given for the default constructor.
friend class boost::serialization::access;
```

On the other hand, the specifics of the functionality required for the `Serialize()` function are somewhat more difficult. The `Serialize()` function will be called either when a tree is being saved to disk or loaded from disk. The **boost::serialization** (p.251) documentation is fairly comprehensive, but when writing a `Serialize()` method for mlpack trees you should use `data::CreateNVP()` instead of `BOOST_SERIALIZATION_NVP()`. This is because mlpack classes implement `Serialize()` instead of `serialize()` in order to conform to the mlpack style guidelines, and making this work requires some interesting shim code, which is hidden inside of `data::CreateNVP()`. It may be useful to look at other `Serialize()` methods contained in other mlpack classes as an example.

An important note is that it is very difficult to use references with **boost::serialization** (p.251), because `Serialize()` may be called at any time during the object's lifetime, and references cannot be re-seated. In general this will require the use of pointers, which then require manual memory management. Therefore, be careful that `Serialize()` (and the tree's destructor) properly handle memory management!

32.6 The TreeTraits trait class

Some tree-based algorithms can specialize if the tree fulfills certain conditions. For instance, if the regions represented by two sibling nodes cannot overlap, an algorithm may be able to perform a simpler computation. Based on this reasoning, the `TreeTraits` trait class (much like the **mlpack::kernel::KernelTraits** (p.1433) class) exists in order to allow a tree to specify (via a `const static bool`) when these types of conditions are satisfied. **Note that a `TreeTraits` class is not required**, but may be helpful.

The `TreeTraits` trait class is a template class that takes a `TreeType` as a parameter, and exposes `const static bool` values that depend on the tree. Setting these values is achieved by specialization. The code below shows the default `TreeTraits` values (these are the values that will be used if no specialization is provided for a given `TreeType`).

```
template<typename TreeType>
class TreeTraits
{
public:
    // This is true if the subspaces represented by the children of a node can
    // overlap.
    static const bool HasOverlappingChildren = true;

    // This is true if Point(0) is the centroid of the node.
    static const bool FirstPointIsCentroid = false;

    // This is true if the points contained in the first child of a node
    // (Child(0)) are also contained in that node.
    static const bool HasSelfChildren = false;

    // This is true if the tree rearranges points in the dataset when it is built.
    static const bool RearrangesDataset = false;

    // This is true if the tree always has only two children.
    static const bool BinaryTree = false;
};
```

An example specialization for the `mlpack::tree::KDTree` (p. 453) class is given below. Note that `mlpack::tree::KDTree` (p. 453) is itself a template class (like every class satisfying the `TreeType` policy), so we are specializing to a template parameter.

```
template<typename MetricType,
        typename StatisticType,
        typename MatType>
template<>
class TreeTraits<KDTree<MetricType, StatisticType, MatType>>
{
public:
    // The regions represented by the two children of a node may not overlap.
    static const bool HasOverlappingChildren = false;

    // There is no guarantee that the first point of a node is the centroid.
    static const bool FirstPointIsCentroid = false;

    // Points are not contained at multiple levels (only at the leaves).
    static const bool HasSelfChildren = false;

    // Points are rearranged during the building of the tree.
    static const bool RearrangesDataset = true;

    // The tree is always binary.
    static const bool BinaryTree = true;
};
```

Currently, the traits available are each of the five detailed above. For more information, see the `mlpack::tree::TreeTraits` (p. 2302) documentation.

32.7 A list of trees in mlpack and more information

mlpack contains several ready-to-use implementations of trees that satisfy the `TreeType` policy API:

- `mlpack::tree::KDTree` (p. 453)
- `mlpack::tree::MeanSplitKDTree` (p. 455)
- `mlpack::tree::BallTree` (p. 451)
- `mlpack::tree::MeanSplitBallTree` (p. 455)

- **mlpack::tree::RTree** (p. 459)
- **mlpack::tree::RStarTree** (p. 459)
- **mlpack::tree::StandardCoverTree** (p. 460)

Often, these are template typedefs of more flexible tree classes:

- **mlpack::tree::BinarySpaceTree** (p. 1998) – binary trees, such as the KD-tree and ball tree
- **mlpack::tree::RectangleTree** (p. 2207) – the R tree and variants
- **mlpack::tree::CoverTree** (p. 2040) – the cover tree and variants

Chapter 33

Bug List

Class CLI (p. 1117)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member `PARAM_COL_IN` (p. 2705) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member `PARAM_COL_IN_REQ` (p. 2706) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member `PARAM_COL_OUT` (p. 2707) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member `PARAM_DOUBLE_IN` (p. 2708) (ID, DESC, ALIAS, DEF)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member `PARAM_DOUBLE_IN_REQ` (p. 2708) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member PARAM_DOUBLE_OUT (p. 2709) (ID, DESC)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member PARAM_FLAG (p. 2710) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member PARAM_INT_IN (p. 2711) (ID, DESC, ALIAS, DEF)

Use a forward declaration of the class. The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member PARAM_INT_IN_REQ (p. 2712) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member PARAM_INT_OUT (p. 2712) (ID, DESC)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member PARAM_MATRIX_IN (p. 2714) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member PARAM_MATRIX_IN_REQ (p. 2715) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member PARAM_MATRIX_OUT (p. 2716) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member PARAM_ROW_IN (p. 2720) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member PARAM_ROW_OUT (p. 2721) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the

`PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member **PARAM_STRING_IN** (p. 2721) (ID, DESC, ALIAS, DEF)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member **PARAM_STRING_IN_REQ** (p. 2722) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member **PARAM_STRING_OUT** (p. 2723) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member **PARAM_TMATRIX_IN** (p. 2723) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member **PARAM_TMATRIX_IN_REQ** (p. 2724) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member **PARAM_TMATRIX_OUT** (p. 2725) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member **PARAM_UCOL_IN** (p. 2726) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member **PARAM_UCOL_OUT** (p. 2726) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member **PARAM_UMATRIX_IN** (p. 2727) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member PARAM_UMATRIX_IN_REQ (p. 2728) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member PARAM_UMATRIX_OUT (p. 2729) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member PARAM_UROW_IN (p. 2730) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member PARAM_UROW_OUT (p. 2731) (ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member PARAM_VECTOR_IN (p. 2731) (T, ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member PARAM_VECTOR_IN_REQ (p. 2732) (T, ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member PARAM_VECTOR_OUT (p. 2733) (T, ID, DESC, ALIAS)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Member TUPLE_TYPE (p. 2735)

The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Chapter 34

Namespace Index

34.1 Namespace List

Here is a list of all namespaces with brief descriptions:

boost	
Set the serialization version of the adaboost class	251
boost::serialization	251
ens	252
mlpack	
.hpp	252
mlpack::adaboost	257
mlpack::amf	
Alternating Matrix Factorization	257
mlpack::ann	
Artificial Neural Network	262
mlpack::ann::augmented	274
mlpack::ann::augmented::scorers	274
mlpack::ann::augmented::tasks	276
mlpack::bindings	276
mlpack::bindings::cli	276
mlpack::bindings::markdown	311
mlpack::bindings::python	324
mlpack::bindings::tests	357
mlpack::bound	364
mlpack::bound::addr	365
mlpack::bound::meta	
Metaprogramming utilities	368
mlpack::cf	
Collaborative filtering	368
mlpack::cv	371
mlpack::data	
Functions to load and save matrices and models	373
mlpack::dbscan	390
mlpack::decision_stump	390
mlpack::det	
Density Estimation Trees	390

mlpack::distribution	
Probability distributions	392
mlpack::emst	
Euclidean Minimum Spanning Trees	393
mlpack::fastmks	
Fast max-kernel search	394
mlpack::gmm	
Gaussian Mixture Models	394
mlpack::hmm	
Hidden Markov Models	395
mlpack::hpt	397
mlpack::kde	
Kernel Density Estimation	399
mlpack::kernel	
Kernel functions	401
mlpack::kmeans	
K-Means clustering	402
mlpack::kpca	405
mlpack::lcc	406
mlpack::lmnn	
Large Margin Nearest Neighbor	406
mlpack::math	
Miscellaneous math routines	406
mlpack::matrix_completion	426
mlpack::meanshift	
Mean shift clustering	426
mlpack::metric	426
mlpack::naive_bayes	
The Naive Bayes Classifier	428
mlpack::nca	
Neighborhood Components Analysis	428
mlpack::neighbor	429
mlpack::nn	435
mlpack::pca	437
mlpack::perceptron	438
mlpack::radical	438
mlpack::range	
Range-search routines	439
mlpack::regression	
Regression methods	440
mlpack::rl	440
mlpack::sfinae	443
mlpack::sparse_coding	443
mlpack::svd	444
mlpack::svm	444
mlpack::tree	
Trees and tree-building procedures	444
mlpack::tree::enumerate	464
mlpack::tree::split	465
mlpack::util	466
std	476

Chapter 35

Hierarchical Index

35.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AdaBoost< mlpack::decision_stump::DecisionStump<> >	488
AdaBoost< mlpack::perceptron::Perceptron<> >	488
version< mlpack::adaboost::AdaBoost< WeakLearnerType, MatType > >	477
version< mlpack::ann::BRNN< OutputLayerType, MergeLayerType, MergeOutputType, InitializationRuleType, CustomLayer... > >	478
version< mlpack::ann::FFN< OutputLayerType, InitializationRuleType, CustomLayer... > >	478
version< mlpack::ann::RNN< OutputLayerType, InitializationRuleType, CustomLayer... > >	479
static_visitor	
CopyVisitor< CustomLayers... >	651
AddVisitor< CustomLayers >	565
BackwardVisitor	587
CopyVisitor< CustomLayers >	651
DeleteVisitor	659
DeltaVisitor	660
DeterministicSetVisitor	661
ForwardVisitor	713
GradientSetVisitor	725
GradientUpdateVisitor	727
GradientVisitor	728
GradientZeroVisitor	730
LoadOutputParameterVisitor	788
LossVisitor	800
OutputHeightVisitor	850
OutputParameterVisitor	851
OutputWidthVisitor	852
ParametersSetVisitor	853
ParametersVisitor	854
ResetCellVisitor	907
ResetVisitor	909
RewardSetVisitor	910
RunSetVisitor	921

SaveOutputParameterVisitor	923
SetInputHeightVisitor	934
SetInputWidthVisitor	935
WeightSetVisitor	972
WeightSizeVisitor	973
DeleteVisitor	1058
GetValueVisitor	1060
PredictVisitor< NeighborSearchPolicy, InterpolationPolicy >	1077
RecommendationVisitor< NeighborSearchPolicy, InterpolationPolicy >	1084
DeleteVisitor	1383
DualBiKDE	1384
DualMonoKDE	1386
ModeVisitor	1412
TrainVisitor	1413
AlphaVisitor	1590
BiSearchVisitor< SortPolicy >	1591
BiSearchVisitor< SortPolicy >	1591
DeleteVisitor	1595
DeleteVisitor	1595
EpsilonVisitor	1600
FirstLeafExactVisitor	1601
MonoSearchVisitor	1618
MonoSearchVisitor	1618
NaiveVisitor	1620
ReferenceSetVisitor	1701
ReferenceSetVisitor	1701
SampleAtLeavesVisitor	1702
SearchModeVisitor	1703
SingleModeVisitor	1704
SingleSampleLimitVisitor	1705
TauVisitor	1706
TrainVisitor< SortPolicy >	1707
TrainVisitor< SortPolicy >	1707
BiSearchVisitor	1748
DeleteVisitor	1751
MonoSearchVisitor	1752
NaiveVisitor	1753
ReferenceSetVisitor	1771
SingleModeVisitor	1779
TrainVisitor	1780
template AuxiliarySplitInfo< ElemType >	
DecisionTree< FitnessFunction, NumericSplitType, CategoricalSplitType, DimensionSelectionType, ElemType, NoRecursion >	2065
DatasetMapper< mlpack::data::IncrementPolicy, double >	1172
FastMKS< mlpack::kernel::CosineDistance >	1280
FastMKS< mlpack::kernel::EpanechnikovKernel >	1280
FastMKS< mlpack::kernel::GaussianKernel >	1280
FastMKS< mlpack::kernel::HyperbolicTangentKernel >	1280
FastMKS< mlpack::kernel::LinearKernel >	1280
FastMKS< mlpack::kernel::PolynomialKernel >	1280
FastMKS< mlpack::kernel::TriangularKernel >	1280
HMM< distribution::RegressionDistribution >	1335
HMMRegression	1354
HMM< mlpack::distribution::DiscreteDistribution >	1335

HMM< mlpack::distribution::GaussianDistribution >	1335
HMM< mlpack::gmm::DiagonalGMM >	1335
HMM< mlpack::gmm::GMM >	1335
HRectBound< metric::EuclideanDistance, ElemType >	1018
HRectBound< MetricType >	1018
HRectBound< mlpack::metric::LMetric, ElemType >	1018
InitHMMModel	480
IPMetric< mlpack::kernel::CosineDistance >	1565
IPMetric< mlpack::kernel::EpanechnikovKernel >	1565
IPMetric< mlpack::kernel::GaussianKernel >	1565
IPMetric< mlpack::kernel::HyperbolicTangentKernel >	1565
IPMetric< mlpack::kernel::LinearKernel >	1565
IPMetric< mlpack::kernel::PolynomialKernel >	1565
IPMetric< mlpack::kernel::TriangularKernel >	1565
IsVector< VecType >	482
IsVector< arma::Col< eT > >	483
IsVector< arma::Row< eT > >	484
IsVector< arma::SpCol< eT > >	485
IsVector< arma::SpRow< eT > >	485
IsVector< arma::SpSubview< eT > >	486
IsVector< arma::subview_col< eT > >	487
IsVector< arma::subview_row< eT > >	487
LMetric< TPower, true >	1569
AdaBoost< WeakLearnerType, MatType >	488
AdaBoostModel	494
AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >	499
AverageInitialization	502
CompleteIncrementalTermination< TerminationPolicy >	504
GivenInitialization	508
IncompleteIncrementalTermination< TerminationPolicy >	510
MaxIterationTermination	513
NMFALSUpdate	516
NMFMultiplicativeDistanceUpdate	519
NMFMultiplicativeDivergenceUpdate	523
RandomAcolInitialization< columnsToAverage >	526
RandomInitialization	528
SimpleResidueTermination	529
SimpleToleranceTermination< MatType >	535
SVDBatchLearning	539
SVDCompleteIncrementalLearning< MatType >	542
SVDCompleteIncrementalLearning< arma::sp_mat >	544
SVDIncompleteIncrementalLearning	547
ValidationRMSETermination< MatType >	549
Add< InputDataType, OutputDataType >	553
AddMerge< InputDataType, OutputDataType, CustomLayers >	558
AlphaDropout< InputDataType, OutputDataType >	567
AtrousConvolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, Input← DataType, OutputDataType >	572
AddTask	580
CopyTask	582
SortTask	584
BaseLayer< ActivationFunction, InputDataType, OutputDataType >	588
BatchNorm< InputDataType, OutputDataType >	592
BernoulliDistribution< DataType >	599

BilinearInterpolation< InputDataType, OutputDataType >	605
BinaryRBM	609
BRNN< OutputLayerType, MergeLayerType, MergeOutputType, InitializationRuleType, CustomLayers >	610
Concat< InputDataType, OutputDataType, CustomLayers >	621
Concatenate< InputDataType, OutputDataType >	630
ConcatPerformance< OutputLayerType, InputDataType, OutputDataType >	634
Constant< InputDataType, OutputDataType >	638
ConstInitialization	641
Convolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, InputDataType, OutputDataType >	643
CReLU< InputDataType, OutputDataType >	652
CrossEntropyError< InputDataType, OutputDataType >	656
DiceLoss< InputDataType, OutputDataType >	662
DropConnect< InputDataType, OutputDataType >	666
Dropout< InputDataType, OutputDataType >	673
EarthMoverDistance< InputDataType, OutputDataType >	677
ELU< InputDataType, OutputDataType >	680
FastLSTM< InputDataType, OutputDataType >	685
FFN< OutputLayerType, InitializationRuleType, CustomLayers >	692
FFTConvolution< BorderMode, padLastDim >	705
FlexibleReLU< InputDataType, OutputDataType >	707
FullConvolution	714
GaussianInitialization	715
Glimpse< InputDataType, OutputDataType >	717
GlorotInitializationType< Uniform >	723
GRU< InputDataType, OutputDataType >	731
HardSigmoidFunction	737
HardTanH< InputDataType, OutputDataType >	740
HeInitialization	744
IdentityFunction	746
InitTraits< InitRuleType >	750
InitTraits< KathirvalavakumarSubavathiInitialization >	751
InitTraits< NguyenWidrowInitialization >	752
Join< InputDataType, OutputDataType >	753
KathirvalavakumarSubavathiInitialization	756
KLDivergence< InputDataType, OutputDataType >	759
LayerNorm< InputDataType, OutputDataType >	762
LayerTraits< LayerType >	769
LeakyReLU< InputDataType, OutputDataType >	771
LecunNormalInitialization	774
Linear< InputDataType, OutputDataType >	776
LinearNoBias< InputDataType, OutputDataType >	782
LogisticFunction	789
LogSoftMax< InputDataType, OutputDataType >	792
Lookup< InputDataType, OutputDataType >	795
LSTM< InputDataType, OutputDataType >	801
MaxPooling< InputDataType, OutputDataType >	808
MaxPoolingRule	814
MeanPooling< InputDataType, OutputDataType >	814
MeanPoolingRule	822
MeanSquaredError< InputDataType, OutputDataType >	823
MultiplyConstant< InputDataType, OutputDataType >	825
MultiplyMerge< InputDataType, OutputDataType, CustomLayers >	829
NaiveConvolution< BorderMode >	834

NegativeLogLikelihood< InputDataType, OutputDataType >	837
NetworkInitialization< InitializationRuleType, CustomLayers >	841
NguyenWidrowInitialization	842
OivsInitialization< ActivationFunction >	844
OrthogonalInitialization	848
PReLU< InputDataType, OutputDataType >	855
RandomInitialization	861
RBM< InitializationRuleType, DataType, PolicyType >	864
ReconstructionLoss< InputDataType, OutputDataType, DistType >	880
RectifierFunction	883
Recurrent< InputDataType, OutputDataType, CustomLayers >	886
RecurrentAttention< InputDataType, OutputDataType >	892
ReinforceNormal< InputDataType, OutputDataType >	898
Reparametrization< InputDataType, OutputDataType >	903
RNN< OutputLayerType, InitializationRuleType, CustomLayers >	911
Select< InputDataType, OutputDataType >	924
Sequential< InputDataType, OutputDataType, Residual, CustomLayers >	927
SigmoidCrossEntropyError< InputDataType, OutputDataType >	937
SoftplusFunction	940
SoftsignFunction	943
SpikeSlabRBM	947
Subview< InputDataType, OutputDataType >	947
SVDConvolution< BorderMode >	950
SwishFunction	953
TanhFunction	955
TransposedConvolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, InputDataType, OutputDataType >	959
ValidConvolution	967
VRClassReward< InputDataType, OutputDataType >	967
Backtrace	974
CLIOption< N >	975
ParameterType< T >	977
ParameterType< arma::Col< eT > >	978
ParameterType< arma::Mat< eT > >	978
ParameterType< arma::Row< eT > >	979
ParameterType< std::tuple< mpack::data::DatasetMapper< PolicyType, std::string >, arma::Mat< eT > > >	980
ParameterTypeDeducer< HasSerialize, T >	981
ParameterTypeDeducer< true, T >	981
ProgramDoc	982
BindingInfo	984
MDOption< T >	985
ProgramDocWrapper	986
PyOption< T >	987
ProgramDoc	988
TestOption< N >	990
BallBound< MetricType, VecType >	991
BoundTraits< BoundType >	1002
BoundTraits< BallBound< MetricType, VecType > >	1003
BoundTraits< CellBound< MetricType, ElemType > >	1003
BoundTraits< HollowBallBound< MetricType, ElemType > >	1004
BoundTraits< HRectBound< MetricType, ElemType > >	1005
CellBound< MetricType, ElemType >	1006
HollowBallBound< TMetricType, ElemType >	1006
HRectBound< MetricType, ElemType >	1018

IsLMetric< MetricType >	1028
IsLMetric< metric::LMetric< Power, TakeRoot > >	1029
AverageInterpolation	1030
BatchSVDPolicy	1032
BiasSVDPolicy	1035
CFModel	1042
CFTYPE< DecompositionPolicy, NormalizationType >	1045
CombinedNormalization< NormalizationTypes >	1053
CosineSearch	1056
DummyClass	1059
ItemMeanNormalization	1061
LMetricSearch< TPower >	1064
NMFPolicy	1066
NoNormalization	1070
OverallMeanNormalization	1072
PearsonSearch	1075
RandomizedSVDPolicy	1078
RegressionInterpolation	1085
RegSVDPolicy	1088
SimilarityInterpolation	1092
SVDCompletePolicy	1094
SVDIncompletePolicy	1098
SVDPlusPlusPolicy	1102
SVDWrapper< Factorizer >	1108
UserMeanNormalization	1111
ZScoreNormalization	1114
CLI	1117
Accuracy	1127
CVBase< MLAlgorithm, MatType, PredictionsType, WeightsType >	1129
F1< AS, PositiveClass >	1132
KFoldCV< MLAlgorithm, Metric, MatType, PredictionsType, WeightsType >	1134
MetaInfoExtractor< MLAlgorithm, MT, PT, WT >	1140
MSE	1143
NotFoundMethodForm	1144
Precision< AS, PositiveClass >	1145
Recall< AS, PositiveClass >	1147
SelectMethodForm< MLAlgorithm, HMFs >	1148
SelectMethodForm< MLAlgorithm >	1149
SelectMethodForm< MLAlgorithm >::From< Forms >	1149
SelectMethodForm< MLAlgorithm, HasMethodForm, HMFs... >	1150
SelectMethodForm< MLAlgorithm, HasMethodForm, HMFs... >::From< Forms >	1151
SimpleCV< MLAlgorithm, Metric, MatType, PredictionsType, WeightsType >	1151
TrainForm< MatType, PredictionsType, WeightsType, DatasetInfo, NumClasses >	1157
TrainFormBase4< PT, WT, T1, T2 >	1164
TrainFormBase5< PT, WT, T1, T2, T3 >	1165
TrainFormBase6< PT, WT, T1, T2, T3, T4 >	1167
TrainFormBase7< PT, WT, T1, T2, T3, T4, T5 >	1169
CustomImputation< T >	1170
DatasetMapper< PolicyType, InputType >	1172
HasSerialize< T >	1178
HasSerialize< T >::check< U, V, W >	1180
HasSerializeFunction< T >	1180
Imputer< T, MapperType, StrategyType >	1182
IncrementPolicy	1184

ListwiseDeletion< T >	1187
LoadCSV	1188
MeanImputation< T >	1191
MedianImputation< T >	1192
MissingPolicy	1193
DBSCAN< RangeSearchType, PointSelectionPolicy >	1197
OrderedPointSelection	1200
RandomPointSelection	1201
DecisionStump< MatType >	1202
DTree< MatType, TagType >	1207
PathCacher	1221
DiagonalGaussianDistribution	1226
DiscreteDistribution	1232
GammaDistribution	1238
GaussianDistribution	1246
LaplaceDistribution	1251
RegressionDistribution	1257
DTBRules< MetricType, TreeType >	1264
DTBStat	1269
DualTreeBoruvka< MetricType, MatType, TreeType >	1272
EdgePair	1275
UnionFind	1278
FastMKS< KernelType, MatType, TreeType >	1280
FastMKSSModel	1291
FastMKSRules< KernelType, TreeType >	1298
FastMKSSStat	1303
DiagonalConstraint	1306
DiagonalGMM	1308
EigenvalueRatioConstraint	1317
EMFit< InitialClusteringType, CovarianceConstraintPolicy, Distribution >	1318
GMM	1323
NoConstraint	1332
PositiveDefiniteConstraint	1334
HMM< Distribution >	1335
HMMModel	1350
CVFunction< CVType, MLAlgorithm, TotalArgs, BoundArgs >	1361
DeduceHyperParameterTypes< Args >	1365
DeduceHyperParameterTypes< Args >::ResultHolder< HPTypes >	1366
DeduceHyperParameterTypes< PreFixedArg< T >, Args... >	1366
DeduceHyperParameterTypes< PreFixedArg< T >, Args... >::ResultHolder< HPTypes >	1367
DeduceHyperParameterTypes< T, Args... >	1368
DeduceHyperParameterTypes< T, Args... >::IsCollectionType< Type >	1369
DeduceHyperParameterTypes< T, Args... >::ResultHolder< HPTypes >	1371
DeduceHyperParameterTypes< T, Args... >::ResultHPTYPE< ArgumentType, IsArithmetic >	1372
DeduceHyperParameterTypes< T, Args... >::ResultHPTYPE< ArithmeticType, true >	1372
DeduceHyperParameterTypes< T, Args... >::ResultHPTYPE< CollectionType, false >	1373
FixedArg< T, I >	1373
HyperParameterTuner< MLAlgorithm, Metric, CV, OptimizerType, MatType, PredictionsType, WeightsType >	1375
IsPreFixedArg< T >	1380
PreFixedArg< T >	1381
PreFixedArg< T & >	1382
KDE< KernelType, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTreeTraversalType >	1387
KDEModel	1397
KDERules< MetricType, KernelType, TreeType >	1404

KDEStat	1408
KernelNormalizer	1410
CauchyKernel	1414
CosineDistance	1416
EpanechnikovKernel	1417
ExampleKernel	1421
GaussianKernel	1424
HyperbolicTangentKernel	1430
KernelTraits< KernelType >	1433
KernelTraits< CauchyKernel >	1434
KernelTraits< CosineDistance >	1435
KernelTraits< EpanechnikovKernel >	1436
KernelTraits< GaussianKernel >	1437
KernelTraits< LaplacianKernel >	1438
KernelTraits< SphericalKernel >	1439
KernelTraits< TriangularKernel >	1440
KMeansSelection< ClusteringType, maxIterations >	1441
LaplacianKernel	1442
LinearKernel	1446
NystroemMethod< KernelType, PointSelectionPolicy >	1448
OrderedSelection	1450
PolynomialKernel	1451
PSpectrumStringKernel	1454
RandomSelection	1457
SphericalKernel	1458
TriangularKernel	1461
AllowEmptyClusters	1464
DualTreeKMeans< MetricType, MatType, TreeType >	1466
DualTreeKMeansRules< MetricType, TreeType >	1469
ElkanKMeans< MetricType, MatType >	1478
HamerlyKMeans< MetricType, MatType >	1479
KillEmptyClusters	1481
KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy, LloydStepType, MatType >	1483
MaxVarianceNewCluster	1489
NaiveKMeans< MetricType, MatType >	1491
PellegMooreKMeans< MetricType, MatType >	1493
PellegMooreKMeansRules< MetricType, TreeType >	1495
PellegMooreKMeansStatistic	1498
RandomPartition	1500
RefinedStart	1502
SampleInitialization	1506
KernelPCA< KernelType, KernelRule >	1507
NaiveKernelRule< KernelType >	1511
NystroemKernelRule< KernelType, PointSelectionPolicy >	1512
LocalCoordinateCoding	1513
Constraints< MetricType >	1520
LMNN< MetricType, OptimizerType >	1526
LMNNFunction< MetricType >	1531
Log	1538
ColumnsToBlocks	1541
RangeType< T >	1549
MatrixCompletion	1558
MeanShift< UseKernel, KernelType, MatType >	1561
IPMetric< KernelType >	1565

LMetric< TPower, TTakeRoot >	1569
MahalanobisDistance< TakeRoot >	1572
NaiveBayesClassifier< ModelMatType >	1576
NCA< MetricType, OptimizerType >	1583
SoftmaxErrorFunction< MetricType >	1586
DrusillaSelect< MatType >	1597
FurthestNS	1602
LSHSearch< SortPolicy >	1609
NearestNS	1621
NeighborSearch< SortPolicy, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTreeTraversal← Type >	1627
NeighborSearchRules< SortPolicy, MetricType, TreeType >	1640
NeighborSearchRules< SortPolicy, MetricType, TreeType >::CandidateCmp	1652
NeighborSearchStat< SortPolicy >	1653
NSModel< SortPolicy >	1657
QDAFN< MatType >	1666
RAModel< SortPolicy >	1669
RAQueryStat< SortPolicy >	1678
RASearch< SortPolicy, MetricType, MatType, TreeType >	1681
RASearchRules< SortPolicy, MetricType, TreeType >	1692
RAUtil	1699
SparseAutoencoder	1711
SparseAutoencoderFunction	1717
ExactSVDPolicy	1722
PCA< DecompositionPolicy >	1723
QUICSVDPolicy	1727
RandomizedBlockKrylovSVDPolicy	1730
RandomizedSVDPolicy	1732
Perceptron< LearnPolicy, WeightInitializationPolicy, MatType >	1735
RandomInitialization	1740
SimpleWeightUpdate	1741
ZeroInitialization	1742
Radical	1744
RangeSearch< MetricType, MatType, TreeType >	1754
RangeSearchRules< MetricType, TreeType >	1764
RangeSearchStat	1768
RSModel	1772
LARS	1783
LinearRegression	1789
LogisticRegression< MatType >	1795
LogisticRegressionFunction< MatType >	1803
SoftmaxRegression	1811
SoftmaxRegressionFunction	1818
Acrobot	1825
Acrobot::State	1831
AggregatedPolicy< PolicyType >	1835
AsyncLearning< WorkerType, EnvironmentType, NetworkType, UpdaterType, PolicyType >	1837
CartPole	1842
CartPole::State	1846
ContinuousMountainCar	1850
ContinuousMountainCar::Action	1854
ContinuousMountainCar::State	1855
GreedyPolicy< EnvironmentType >	1858
MountainCar	1861

MountainCar::State	1865
NStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >	1868
OneStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >	1871
OneStepSarsaWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >	1875
Pendulum	1878
Pendulum::Action	1881
Pendulum::State	1882
QLearning< EnvironmentType, NetworkType, UpdaterType, PolicyType, ReplayType >	1885
RandomReplay< EnvironmentType >	1890
RewardClipping< EnvironmentType >	1894
TrainingConfig	1901
MethodFormDetector< Class, MethodForm, AdditionalArgsCount >	1907
MethodFormDetector< Class, MethodForm, 0 >	1907
MethodFormDetector< Class, MethodForm, 1 >	1908
MethodFormDetector< Class, MethodForm, 2 >	1908
MethodFormDetector< Class, MethodForm, 3 >	1909
MethodFormDetector< Class, MethodForm, 4 >	1910
MethodFormDetector< Class, MethodForm, 5 >	1910
MethodFormDetector< Class, MethodForm, 6 >	1911
MethodFormDetector< Class, MethodForm, 7 >	1912
DataDependentRandomInitializer	1913
NothingInitializer	1914
RandomInitializer	1915
SparseCoding	1916
BiasSVD< OptimizerType >	1925
BiasSVDFunction< MatType >	1926
QUIC_SVD	1932
RandomizedBlockKrylovSVD	1934
RandomizedSVD	1937
RegularizedSVD< OptimizerType >	1943
RegularizedSVDFunction< MatType >	1945
SVDPlusPlus< OptimizerType >	1951
SVDPlusPlusFunction< MatType >	1954
LinearSVM< MatType >	1959
LinearSVMFunction< MatType >	1967
Timer	1975
Timers	1977
AllCategoricalSplit< FitnessFunction >	1981
AllCategoricalSplit< FitnessFunction >::AuxiliarySplitInfo< ElemType >	1984
AllDimensionSelect	1984
AxisParallelProjVector	1986
BestBinaryNumericSplit< FitnessFunction >	1989
BestBinaryNumericSplit< FitnessFunction >::AuxiliarySplitInfo< ElemType >	1991
BinaryNumericSplit< FitnessFunction, ObservationType >	1992
BinaryNumericSplitInfo< ObservationType >	1996
BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >	1998
BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::BreadthFirstDualTree← Traverser< RuleType >	2019
BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::DualTreeTraverser< Rule← Type >	2022
BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::SingleTreeTraverser< RuleType >	2026
CategoricalSplitInfo	2028
CompareCosineNode	2029

CosineTree	2030
CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >	2040
CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::DualTreeTraverser< RuleType >	2060
CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::SingleTreeTraverser< RuleType >	2063
DiscreteHilbertValue< TreeElemType >	2079
EmptyStatistic	2080
ExampleTree< MetricType, StatisticType, MatType >	2081
FirstPointIsRoot	2089
GiniGain	2090
GiniImpurity	2092
GreedySingleTreeTraverser< TreeType, RuleType >	2093
HilbertRTreeAuxiliaryInformation< TreeType, HilbertValueType >	2095
HilbertRTreeDescentHeuristic	2101
HilbertRTreeSplit< splitOrder >	2102
HoeffdingCategoricalSplit< FitnessFunction >	2104
HoeffdingNumericSplit< FitnessFunction, ObservationType >	2108
HoeffdingTree< FitnessFunction, NumericSplitType, CategoricalSplitType >	2113
HoeffdingTreeModel	2126
HyperplaneBase< BoundT, ProjVectorT >	2133
InformationGain	2138
IsSpillTree< TreeType >	2140
IsSpillTree< tree::SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > >	2141
MeanSpaceSplit< MetricType, MatType >	2141
MeanSplit< BoundType, MatType >	2142
MeanSplit< BoundType, MatType >::SplitInfo	2145
MidpointSpaceSplit< MetricType, MatType >	2146
MidpointSplit< BoundType, MatType >	2147
MidpointSplit< BoundType, MatType >::SplitInfo	2150
MinimalCoverageSweep< SplitPolicy >	2151
MinimalCoverageSweep< SplitPolicy >::SweepCost< TreeType >	2154
MinimalSplitsNumberSweep< SplitPolicy >	2155
MinimalSplitsNumberSweep< SplitPolicy >::SweepCost< typename >	2157
MultipleRandomDimensionSelect	2158
NoAuxiliaryInformation< TreeType >	2160
NumericSplitInfo< ObservationType >	2165
Octree< MetricType, StatisticType, MatType >	2167
Octree< MetricType, StatisticType, MatType >::DualTreeTraverser< MetricType, StatisticType, MatType >	2184
Octree< MetricType, StatisticType, MatType >::SingleTreeTraverser< RuleType >	2187
Octree< MetricType, StatisticType, MatType >::SplitType::SplitInfo	2189
ProjVector	2190
QueueFrame< TreeType, TraversalInfoType >	2193
RandomDimensionSelect	2194
RandomForest< FitnessFunction, DimensionSelectionType, NumericSplitType, CategoricalSplitType, Elem← Type >	2196
RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >	2207
RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >← ::DualTreeTraverser< MetricType, StatisticType, MatType, SplitType, DescentType, Auxiliary← InformationType >	2234
RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >← :SingleTreeTraverser< RuleType >	2237
RPlusPlusTreeAuxiliaryInformation< TreeType >	2239
RPlusPlusTreeDescentHeuristic	2246
RPlusPlusTreeSplitPolicy	2247
RPlusTreeDescentHeuristic	2250

RPlusTreeSplit< SplitPolicyType, SweepType >	2251
RPlusTreeSplitPolicy	2253
RPTreeMaxSplit< BoundType, MatType >	2255
RPTreeMaxSplit< BoundType, MatType >::SplitInfo	2260
RPTreeMeanSplit< BoundType, MatType >	2261
RPTreeMeanSplit< BoundType, MatType >::SplitInfo	2264
RStarTreeDescentHeuristic	2266
RStarTreeSplit	2267
RTreeDescentHeuristic	2269
RTreeSplit	2271
SpaceSplit< MetricType, MatType >	2272
SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >	2274
SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::SpillDualTreeTraverser< MetricType, StatisticType, MatType, HyperplaneType, SplitType >	2293
SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::SpillSingleTreeTraverser< MetricType, StatisticType, MatType, HyperplaneType, SplitType >	2297
TraversallInfo< TreeType >	2299
TreeTraits< TreeType >	2302
TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::BallBound, SplitType > >	2305
TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::CellBound, SplitType > >	2307
TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::HollowBallBound, SplitType > >	2309
TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, RPTreeMaxSplit > >	2311
TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, RPTreeMeanSplit > >	2313
TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > >	2316
TreeTraits< CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > >	2318
TreeTraits< Octree< MetricType, StatisticType, MatType > >	2321
TreeTraits< RectangleTree< MetricType, StatisticType, MatType, RPlusTreeSplit< SplitPolicyType, SweepType >, DescentType, AuxiliaryInformationType > >	2323
TreeTraits< RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType > >	2326
TreeTraits< SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > >	2328
UBTreeSplit< BoundType, MatType >	2331
VantagePointSplit< BoundType, MatType, MaxNumSamples >	2331
VantagePointSplit< BoundType, MatType, MaxNumSamples >::SplitInfo	2336
XTreeAuxiliaryInformation< TreeType >	2338
XTreeAuxiliaryInformation< TreeType >::SplitHistoryStruct	2345
XTreeSplit	2347
IsStdVector< T >	2349
IsStdVector< std::vector< T, A > >	2349
NullOutputStream	2350
ParamData	2356
PrefixedOutputStream	2361
ProgramDoc	2368
NeighborSearch< neighbor::NearestNeighborSort, metric::LMetric< TPower, true > >	1627
NeighborSearchStat< neighbor::NearestNeighborSort >	1653
DualTreeKMeansStatistic	1473
template AuxiliarySplitInfo< ElemType > DecisionTree< FitnessFunction, NumericSplitType, CategoricalSplitType, DimensionSelectionType, ElemType, NoRecursion >	2065
RangeType< double >	1549
RangeType< ElemType >	1549
RNN< OutputLayerType, InitializationRuleType, CustomLayers... >	911
true_type SigCheck< U, U >	1912

TrainFormBase4< PT, void, const MT &, const PT &>1164
TrainForm< MT, PT, void, false, false >1158
TrainFormBase5< PT, void, const MT &, const data::DatasetInfo &, const PT &>1165
TrainForm< MT, PT, void, true, false >1160
TrainFormBase5< PT, void, const MT &, const PT &, const size_t >1165
TrainForm< MT, PT, void, false, true >1159
TrainFormBase5< PT, WT, const MT &, const PT &, const WT &>1165
TrainForm< MT, PT, WT, false, false >1161
TrainFormBase6< PT, void, const MT &, const data::DatasetInfo &, const PT &, const size_t >1167
TrainForm< MT, PT, void, true, true >1160
TrainFormBase6< PT, WT, const MT &, const data::DatasetInfo &, const PT &, const WT &>1167
TrainForm< MT, PT, WT, true, false >1162
TrainFormBase6< PT, WT, const MT &, const PT &, const size_t, const WT &>1167
TrainForm< MT, PT, WT, false, true >1162
TrainFormBase7< PT, WT, const MT &, const data::DatasetInfo &, const PT &, const size_t, const WT &>1169
TrainForm< MT, PT, WT, true, true >1163
TrainHMMModel	2371

Chapter 36

Class Index

36.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

version< mlpack::adaboost::AdaBoost< WeakLearnerType, MatType > >	477
version< mlpack::ann::BRNN< OutputLayerType, MergeLayerType, MergeOutputType, InitializationRuleType, CustomLayer... > >	478
version< mlpack::ann::FFN< OutputLayerType, InitializationRuleType, CustomLayer... > >	478
version< mlpack::ann::RNN< OutputLayerType, InitializationRuleType, CustomLayer... > >	479
InitHMMModel	480
IsVector< VecType > If value == true, then VecType is some sort of Armadillo vector or subview	482
IsVector< arma::Col< eT > >	483
IsVector< arma::Row< eT > >	484
IsVector< arma::SpCol< eT > >	485
IsVector< arma::SpRow< eT > >	485
IsVector< arma::SpSubview< eT > >	486
IsVector< arma::subview_col< eT > >	487
IsVector< arma::subview_row< eT > >	487
AdaBoost< WeakLearnerType, MatType > The AdaBoost (p. 488) class	488
AdaBoostModel The model to save to disk	494
AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType > This class implements AMF (p. 499) (alternating matrix factorization) on the given matrix V	499
AverageInitialization This initialization rule initializes matrix W and H to root of the average of V, perturbed with uniform noise	502
CompleteIncrementalTermination< TerminationPolicy > This class acts as a wrapper for basic termination policies to be used by SVDCompleteIncrementalLearning (p. 542)	504
GivenInitialization This initialization rule for AMF (p. 499) simply fills the W and H matrices with the matrices given to the constructor of this object	508

IncompleteIncrementalTermination < TerminationPolicy >	
This class acts as a wrapper for basic termination policies to be used by SVDIncompleteIncrementalLearning (p. 547)	510
MaxIterationTermination	
This termination policy only terminates when the maximum number of iterations has been reached	513
NMFALSUpdate	
This class implements a method titled 'Alternating Least Squares' described in the following paper:	516
NMFMultiplicativeDistanceUpdate	
The multiplicative distance update rules for matrices W and H	519
NMFMultiplicativeDivergenceUpdate	
This follows a method described in the paper 'Algorithms for Non-negative	523
RandomAcolInitialization < columnsToAverage >	
This class initializes the W matrix of the AMF (p. 499) algorithm by averaging p randomly chosen columns of V	526
RandomInitialization	
This initialization rule for AMF (p. 499) simply fills the W and H matrices with uniform random noise in [0, 1]	528
SimpleResidueTermination	
This class implements a simple residue-based termination policy	529
SimpleToleranceTermination < MatType >	
This class implements residue tolerance termination policy	535
SVDBatchLearning	
This class implements SVD batch learning with momentum	539
SVDCompleteIncrementalLearning < MatType >	
This class computes SVD using complete incremental batch learning, as described in the following paper:	542
SVDCompleteIncrementalLearning < arma::sp_mat >	
TODO : Merge this template specialized function for sparse matrix using common row_col_iterator	544
SVDIncompleteIncrementalLearning	
This class computes SVD using incomplete incremental batch learning, as described in the following paper:	547
ValidationRMSETermination < MatType >	
This class implements validation termination policy based on RMSE index	549
Add < InputDataType , OutputDataType >	
Implementation of the Add (p. 553) module class	553
AddMerge < InputDataType , OutputDataType , CustomLayers >	
Implementation of the AddMerge (p. 558) module class	558
AddVisitor < CustomLayers >	
AddVisitor (p. 565) exposes the Add() method of the given module	565
AlphaDropout < InputDataType , OutputDataType >	
The alpha - dropout layer is a regularizer that randomly with probability 'ratio' sets input values to alphaDash	567
AtrousConvolution < ForwardConvolutionRule , BackwardConvolutionRule , GradientConvolutionRule , InputDataType , OutputDataType >	
Implementation of the Atrous Convolution (p. 643) class	572
AddTask	
Generator of instances of the binary addition task	580
CopyTask	
Generator of instances of the binary sequence copy task	582
SortTask	
Generator of instances of the sequence sort task	584
BackwardVisitor	
BackwardVisitor (p. 587) executes the Backward() function given the input, error and delta parameter	587

BaseLayer < ActivationFunction , InputDataType , OutputDataType > Implementation of the base layer	588
BatchNorm < InputDataType , OutputDataType > Declaration of the Batch Normalization layer class	592
BernoulliDistribution < DataType > Multiple independent Bernoulli distributions	599
BilinearInterpolation < InputDataType , OutputDataType > Definition and Implementation of the Bilinear Interpolation Layer	605
BinaryRBM For more information, see the following paper:	609
BRNN < OutputLayerType , MergeLayerType , MergeOutputType , InitializationRuleType , CustomLayers > Implementation of a standard bidirectional recurrent neural network container	610
Concat < InputDataType , OutputDataType , CustomLayers > Implementation of the Concat (p. 621) class	621
Concatenate < InputDataType , OutputDataType > Implementation of the Concatenate (p. 630) module class	630
ConcatPerformance < OutputLayerType , InputDataType , OutputDataType > Implementation of the concat performance class	634
Constant < InputDataType , OutputDataType > Implementation of the constant layer	638
ConstInitialization This class is used to initialize weight matrix with constant values	641
Convolution < ForwardConvolutionRule , BackwardConvolutionRule , GradientConvolutionRule , InputDataType , OutputDataType > Implementation of the Convolution (p. 643) class	643
CopyVisitor < CustomLayers > This visitor is to support copy constructor for neural network module	651
CReLU < InputDataType , OutputDataType > A concatenated ReLU has two outputs, one ReLU and one negative ReLU, concatenated together	652
CrossEntropyError < InputDataType , OutputDataType > The cross-entropy performance function measures the network's performance according to the cross-entropy between the input and target distributions	656
DeleteVisitor DeleteVisitor (p. 659) executes the destructor of the instantiated object	659
DeltaVisitor DeltaVisitor (p. 660) exposes the delta parameter of the given module	660
DeterministicSetVisitor DeterministicSetVisitor (p. 661) set the deterministic parameter given the deterministic value	661
DiceLoss < InputDataType , OutputDataType > The dice loss performance function measures the network's performance according to the dice coefficient between the input and target distributions	662
DropConnect < InputDataType , OutputDataType > The DropConnect (p. 666) layer is a regularizer that randomly with probability ratio sets the connection values to zero and scales the remaining elements by factor 1 / (1 - ratio)	666
Dropout < InputDataType , OutputDataType > The dropout layer is a regularizer that randomly with probability 'ratio' sets input values to zero and scales the remaining elements by factor 1 / (1 - ratio) rather than during test time so as to keep the expected sum same	673
EarthMoverDistance < InputDataType , OutputDataType > The earth mover distance function measures the network's performance according to the Kantorovich-Rubinstein duality approximation	677
ELU < InputDataType , OutputDataType > The ELU (p. 680) activation function, defined by	680

FastLSTM < InputDataType , OutputDataType >	
An implementation of a faster version of the Fast LSTM (p. 801) network layer	685
FFN < OutputLayerType , InitializationRuleType , CustomLayers >	
Implementation of a standard feed forward network	692
FFTConvolution < BorderMode , padLastDim >	
Computes the two-dimensional convolution through fft	705
FlexibleReLU < InputDataType , OutputDataType >	
The FlexibleReLU (p. 707) activation function, defined by	707
ForwardVisitor	
ForwardVisitor (p. 713) executes the Forward() function given the input and output parameter	713
FullConvolution	714
GaussianInitialization	
This class is used to initialize weight matrix with a gaussian	715
Glimpse < InputDataType , OutputDataType >	
The glimpse layer returns a retina-like representation (down-scaled cropped images) of increasing scale around a given location in a given image	717
GlorotInitializationType < Uniform >	
This class is used to initialize the weight matrix with the Glorot Initialization method	723
GradientSetVisitor	
GradientSetVisitor (p. 725) update the gradient parameter given the gradient set	725
GradientUpdateVisitor	
GradientUpdateVisitor (p. 727) update the gradient parameter given the gradient set	727
GradientVisitor	
SearchModeVisitor executes the Gradient() method of the given module using the input and delta parameter	728
GradientZeroVisitor	730
GRU < InputDataType , OutputDataType >	
An implementation of a gru network layer	731
HardSigmoidFunction	
The hard sigmoid function, defined by	737
HardTanH < InputDataType , OutputDataType >	
The Hard Tanh activation function, defined by	740
HeInitialization	
This class is used to initialize weight matrix with the He initialization rule given by He et	744
IdentityFunction	
The identity function, defined by	746
InitTraits < InitRuleType >	
This is a template class that can provide information about various initialization methods	750
InitTraits < KathirvalavakumarSubavathiInitialization >	
Initialization traits of the kathirvalavakumar subavath initialization rule	751
InitTraits < NguyenWidrowInitialization >	
Initialization traits of the Nguyen-Widrow initialization rule	752
Join < InputDataType , OutputDataType >	
Implementation of the Join (p. 753) module class	753
KathirvalavakumarSubavathiInitialization	
This class is used to initialize the weight matrix with the method proposed by T	756
KLDivergence < InputDataType , OutputDataType >	
The Kullback–Leibler divergence is often used for continuous distributions (direct regression)	759
LayerNorm < InputDataType , OutputDataType >	
Declaration of the Layer Normalization class	762
LayerTraits < LayerType >	
This is a template class that can provide information about various layers	769
LeakyReLU < InputDataType , OutputDataType >	
The LeakyReLU (p. 771) activation function, defined by	771

LecunNormalInitialization	
This class is used to initialize weight matrix with the Lecun Normalization initialization rule	774
Linear< InputDataType, OutputDataType >	
Implementation of the Linear (p. 776) layer class	776
LinearNoBias< InputDataType, OutputDataType >	
Implementation of the LinearNoBias (p. 782) class	782
LoadOutputParameterVisitor	
LoadOutputParameterVisitor (p. 788) restores the output parameter using the given parameter set	788
LogisticFunction	
The logistic function, defined by	789
LogSoftMax< InputDataType, OutputDataType >	
Implementation of the log softmax layer	792
Lookup< InputDataType, OutputDataType >	
Implementation of the Lookup (p. 795) class	795
LossVisitor	
LossVisitor (p. 800) exposes the Loss() method of the given module	800
LSTM< InputDataType, OutputDataType >	
Implementation of the LSTM (p. 801) module class	801
MaxPooling< InputDataType, OutputDataType >	
Implementation of the MaxPooling (p. 808) layer	808
MaxPoolingRule	814
MeanPooling< InputDataType, OutputDataType >	
Implementation of the MeanPooling (p. 814)	814
MeanPoolingRule	822
MeanSquaredError< InputDataType, OutputDataType >	
The mean squared error performance function measures the network's performance according to the mean of squared errors	823
MultiplyConstant< InputDataType, OutputDataType >	
Implementation of the multiply constant layer	825
MultiplyMerge< InputDataType, OutputDataType, CustomLayers >	
Implementation of the MultiplyMerge (p. 829) module class	829
NaiveConvolution< BorderMode >	
Computes the two-dimensional convolution	834
NegativeLogLikelihood< InputDataType, OutputDataType >	
Implementation of the negative log likelihood layer	837
NetworkInitialization< InitializationRuleType, CustomLayers >	
This class is used to initialize the network with the given initialization rule	841
NguyenWidrowInitialization	
This class is used to initialize the weight matrix with the Nguyen-Widrow method	842
OivsInitialization< ActivationFunction >	
This class is used to initialize the weight matrix with the oivs method	844
OrthogonalInitialization	
This class is used to initialize the weight matrix with the orthogonal matrix initialization	848
OutputHeightVisitor	
OutputHeightVisitor (p. 850) exposes the OutputHeight() method of the given module	850
OutputParameterVisitor	
OutputParameterVisitor (p. 851) exposes the output parameter of the given module	851
OutputWidthVisitor	
OutputWidthVisitor (p. 852) exposes the OutputWidth() method of the given module	852
ParametersSetVisitor	
ParametersSetVisitor (p. 853) update the parameters set using the given matrix	853
ParametersVisitor	
ParametersVisitor (p. 854) exposes the parameters set of the given module and stores the parameters set into the given matrix	854

PReLU < InputDataType , OutputDataType >	
The PReLU (p. 855) activation function, defined by (where alpha is trainable)	855
RandomInitialization	
This class is used to initialize randomly the weight matrix	861
RBM < InitializationRuleType , DataType , PolicyType >	
The implementation of the RBM (p. 864) module	864
ReconstructionLoss < InputDataType , OutputDataType , DistType >	
The reconstruction loss performance function measures the network's performance equal to the negative log probability of the target with the input distribution	880
RectifierFunction	
The rectifier function, defined by	883
Recurrent < InputDataType , OutputDataType , CustomLayers >	
Implementation of the RecurrentLayer class	886
RecurrentAttention < InputDataType , OutputDataType >	
This class implements the Recurrent (p. 886) Model for Visual Attention, using a variety of possible layer implementations	892
ReinforceNormal < InputDataType , OutputDataType >	
Implementation of the reinforce normal layer	898
Reparametrization < InputDataType , OutputDataType >	
Implementation of the Reparametrization (p. 903) layer class	903
ResetCellVisitor	
ResetCellVisitor (p. 907) executes the ResetCell() function	907
ResetVisitor	
ResetVisitor (p. 909) executes the Reset() function	909
RewardSetVisitor	
RewardSetVisitor (p. 910) set the reward parameter given the reward value	910
RNN < OutputLayerType , InitializationRuleType , CustomLayers >	
Implementation of a standard recurrent neural network container	911
RunSetVisitor	
RunSetVisitor (p. 921) set the run parameter given the run value	921
SaveOutputParameterVisitor	
SaveOutputParameterVisitor (p. 923) saves the output parameter into the given parameter set	923
Select < InputDataType , OutputDataType >	
The select module selects the specified column from a given input matrix	924
Sequential < InputDataType , OutputDataType , Residual , CustomLayers >	
Implementation of the Sequential (p. 927) class	927
SetInputHeightVisitor	
SetInputHeightVisitor (p. 934) updates the input height parameter with the given input height	934
SetInputWidthVisitor	
SetInputWidthVisitor (p. 935) updates the input width parameter with the given input width	935
SigmoidCrossEntropyError < InputDataType , OutputDataType >	
The SigmoidCrossEntropyError (p. 937) performance function measures the network's performance according to the cross-entropy function between the input and target distributions	937
SoftplusFunction	
The softplus function, defined by	940
SoftsignFunction	
The softsign function, defined by	943
SpikeSlabRBM	
For more information, see the following paper:	947
Subview < InputDataType , OutputDataType >	
Implementation of the subview layer	947
SVDConvolution < BorderMode >	
Computes the two-dimensional convolution using singular value decomposition	950

SwishFunction	
The swish function, defined by	953
TanhFunction	
The tanh function, defined by	955
TransposedConvolution < ForwardConvolutionRule , BackwardConvolutionRule , GradientConvolutionRule , InputDataType , OutputDataType >	
Implementation of the Transposed Convolution (p. 643) class	959
ValidConvolution	967
VRCClassReward < InputDataType , OutputDataType >	
Implementation of the variance reduced classification reinforcement layer	967
WeightSetVisitor	
WeightSetVisitor (p. 972) update the module parameters given the parameters set	972
WeightSizeVisitor	
WeightSizeVisitor (p. 973) returns the number of weights of the given module	973
Backtrace	
Provides a backtrace	974
CLIOption < N >	
A static object whose constructor registers a parameter with the CLI (p. 1117) class	975
ParameterType < T >	
Utility struct to return the type that boost::program_options should accept for a given input type	977
ParameterType < arma::Col < eT > >	
For vector types, boost::program_options will accept a std::string, not an arma::Col<eT> (since it is not clear how to specify a vector on the command-line)	978
ParameterType < arma::Mat < eT > >	
For matrix types, boost::program_options will accept a std::string, not an arma::mat (since it is not clear how to specify a matrix on the command-line)	978
ParameterType < arma::Row < eT > >	
For row vector types, boost::program_options will accept a std::string, not an arma::Row<eT> (since it is not clear how to specify a vector on the command-line)	979
ParameterType < std::tuple < mlpack::data::DatasetMapper < PolicyType , std::string >, arma::Mat < eT > > >	
For matrix+dataset info types, we should accept a std::string	980
ParameterTypeDeducer < HasSerialize , T >	981
ParameterTypeDeducer < true , T >	981
ProgramDoc	
A static object whose constructor registers program documentation with the CLI (p. 1117) class	982
BindingInfo	
Used by the Markdown documentation generator to store multiple ProgramDoc objects, indexed by both the binding name (i.e.	984
MDOption < T >	
The Markdown option class	985
ProgramDocWrapper	986
PyOption < T >	
The Python option class	987
ProgramDoc	
A static object whose constructor registers program documentation with the CLI (p. 1117) class	988
TestOption < N >	
A static object whose constructor registers a parameter with the CLI (p. 1117) class	990
BallBound < MetricType , VecType >	
Ball bound encloses a set of points at a specific distance (radius) from a specific point (center)	991
BoundTraits < BoundType >	
A class to obtain compile-time traits about BoundType classes	1002
BoundTraits < BallBound < MetricType , VecType > >	
A specialization of BoundTraits (p. 1002) for this bound type	1003

BoundTraits < CellBound < MetricType , ElemType > >	1003
BoundTraits < HollowBallBound < MetricType , ElemType > > A specialization of BoundTraits (p. 1002) for this bound type	1004
BoundTraits < HRectBound < MetricType , ElemType > >	1005
CellBound < MetricType , ElemType > The CellBound (p. 1006) class describes a bound that consists of a number of hyperrectangles	1006
HollowBallBound < TMetricType , ElemType > Hollow ball bound encloses a set of points at a specific distance (radius) from a specific point (center) except points at a specific distance from another point (the center of the hole)	1006
HRectBound < MetricType , ElemType > Hyper-rectangle bound for an L-metric	1018
IsLMetric < MetricType > Utility struct where Value is true if and only if the argument is of type LMetric	1028
IsLMetric < metric::LMetric < Power , TakeRoot > > Specialization for IsLMetric (p. 1028) when the argument is of type LMetric	1029
AverageInterpolation This class performs average interpolation to generate interpolation weights for neighborhood-based collaborative filtering	1030
BatchSVDPolicy Implementation of the Batch SVD policy to act as a wrapper when accessing Batch SVD from within CFTYPE (p. 1045)	1032
BiasSVDPolicy Implementation of the Bias SVD policy to act as a wrapper when accessing Bias SVD from within CFTYPE (p. 1045)	1035
CFModel The model to save to disk	1042
CFTYPE < DecompositionPolicy , NormalizationType > This class implements Collaborative Filtering (CF)	1045
CombinedNormalization < NormalizationTypes > This normalization class performs a sequence of normalization methods on raw ratings	1053
CosineSearch Nearest neighbor search with cosine distance	1056
DeleteVisitor DeleteVisitor (p. 1058) deletes the CFTYPE <> object which is pointed to by the variable cf in class CFModel (p. 1042)	1058
DummyClass This class acts as a dummy class for passing as template parameter	1059
GetValueVisitor GetValueVisitor (p. 1060) returns the pointer which points to the CFTYPE (p. 1045) object	1060
ItemMeanNormalization This normalization class performs item mean normalization on raw ratings	1061
LMetricSearch < TPower > Nearest neighbor search with L _p distance	1064
NMFPolicy Implementation of the NMF policy to act as a wrapper when accessing NMF from within CFTYPE (p. 1045)	1066
NoNormalization This normalization class doesn't perform any normalization	1070
OverallMeanNormalization This normalization class performs overall mean normalization on raw ratings	1072
PearsonSearch Nearest neighbor search with pearson distance (or furthest neighbor search with pearson correlation)	1075

PredictVisitor< NeighborSearchPolicy, InterpolationPolicy >	
PredictVisitor (p. 1077) uses the CFTYPE (p. 1045) object to make predictions on the given combinations of users and items	1077
RandomizedSVDPolicy	
Implementation of the Randomized SVD policy to act as a wrapper when accessing Randomized SVD from within CFTYPE (p. 1045)	1078
RecommendationVisitor< NeighborSearchPolicy, InterpolationPolicy >	
RecommendationVisitor (p. 1084) uses the CFTYPE (p. 1045) object to get recommendations for the given users	1084
RegressionInterpolation	
Implementation of regression-based interpolation method	1085
RegSVDPolicy	
Implementation of the Regularized SVD policy to act as a wrapper when accessing Regularized SVD from within CFTYPE (p. 1045)	1088
SimilarityInterpolation	
With SimilarityInterpolation (p. 1092), interpolation weights are based on similarities between query user and its neighbors	1092
SVDCompletePolicy	
Implementation of the SVD complete incremental policy to act as a wrapper when accessing SVD complete decomposition from within CFTYPE (p. 1045)	1094
SVDIncompletePolicy	
Implementation of the SVD incomplete incremental to act as a wrapper when accessing SVD incomplete incremental from within CFTYPE (p. 1045)	1098
SVDPlusPlusPolicy	
Implementation of the SVDPlusPlus policy to act as a wrapper when accessing SVDPlusPlus from within CFTYPE (p. 1045)	1102
SVDWrapper< Factorizer >	
This class acts as the wrapper for all SVD factorizers which are incompatible with CF module	1108
UserMeanNormalization	
This normalization class performs user mean normalization on raw ratings	1111
ZScoreNormalization	
This normalization class performs z-score normalization on raw ratings	1114
CLI	
Parses the command line for parameters and holds user-specified parameters	1117
Accuracy	
The Accuracy (p. 1127) is a metric of performance for classification algorithms that is equal to a proportion of correctly labeled test items among all ones for given test items	1127
CVBase< MLAlgorithm, MatType, PredictionsType, WeightsType >	
An auxiliary class for cross-validation	1129
F1< AS, PositiveClass >	
F1 (p. 1132) is a metric of performance for classification algorithms that for binary classification is equal to $2 * precision * recall / (precision + recall)$	1132
KFoldCV< MLAlgorithm, Metric, MatType, PredictionsType, WeightsType >	
The class KFoldCV (p. 1134) implements k-fold cross-validation for regression and classification algorithms	1134
MetaInfoExtractor< MLAlgorithm, MT, PT, WT >	
MetaInfoExtractor (p. 1140) is a tool for extracting meta information about a given machine learning algorithm	1140
MSE	
The MeanSquaredError is a metric of performance for regression algorithms that is equal to the mean squared error between predicted values and ground truth (correct) values for given test items	1143
NotFoundMethodForm	1144

Precision < AS, PositiveClass >	
Precision (p. 1145) is a metric of performance for classification algorithms that for binary classification is equal to $tp/(tp + fp)$, where tp and fp are the numbers of true positives and false positives respectively	1145
Recall < AS, PositiveClass >	
Recall (p. 1147) is a metric of performance for classification algorithms that for binary classification is equal to $tp/(tp + fn)$, where tp and fn are the numbers of true positives and false negatives respectively	1147
SelectMethodForm < MLAlgorithm, HMFs >	
A type function that selects a right method form	1148
SelectMethodForm < MLAlgorithm >	1149
SelectMethodForm < MLAlgorithm >::From< Forms >	1149
SelectMethodForm < MLAlgorithm, HasMethodForm, HMFs... >	1150
SelectMethodForm < MLAlgorithm, HasMethodForm, HMFs... >::From< Forms >	1151
SimpleCV < MLAlgorithm, Metric, MatType, PredictionsType, WeightsType >	
SimpleCV (p. 1151) splits data into two sets - training and validation sets - and then runs training on the training set and evaluates performance on the validation set	1151
TrainForm < MatType, PredictionsType, WeightsType, DatasetInfo, NumClasses >	
A wrapper struct for holding a Train form	1157
TrainForm < MT, PT, void, false, false >	1158
TrainForm < MT, PT, void, false, true >	1159
TrainForm < MT, PT, void, true, false >	1160
TrainForm < MT, PT, void, true, true >	1160
TrainForm < MT, PT, WT, false, false >	1161
TrainForm < MT, PT, WT, false, true >	1162
TrainForm < MT, PT, WT, true, false >	1162
TrainForm < MT, PT, WT, true, true >	1163
TrainFormBase4 < PT, WT, T1, T2 >	1164
TrainFormBase5 < PT, WT, T1, T2, T3 >	1165
TrainFormBase6 < PT, WT, T1, T2, T3, T4 >	1167
TrainFormBase7 < PT, WT, T1, T2, T3, T4, T5 >	1169
CustomImputation < T >	
A simple custom imputation class	1170
DatasetMapper < PolicyType, InputType >	
Auxiliary information for a dataset, including mappings to/from strings (or other types) and the datatype of each dimension	1172
HasSerialize < T >	1178
HasSerialize < T >::check< U, V, W >	1180
HasSerializeFunction < T >	1180
Imputer < T, MapperType, StrategyType >	
Given a dataset of a particular datatype, replace user-specified missing value with a variable dependent on the StrategyType and MapperType	1182
IncrementPolicy	
IncrementPolicy (p. 1184) is used as a helper class for DatasetMapper (p. 1172)	1184
ListwiseDeletion < T >	
A complete-case analysis to remove the values containing mappedValue	1187
LoadCSV	
Load the csv file. This class use boost::spirit to implement the parser, please refer to following link http://theboostcpplibaries.com/boost.spirit for quick review	1188
MeanImputation < T >	
A simple mean imputation class	1191
MedianImputation < T >	
This is a class implementation of simple median imputation	1192

MissingPolicy	
MissingPolicy (p. 1193) is used as a helper class for DatasetMapper (p. 1172)	1193
DBSCAN< RangeSearchType, PointSelectionPolicy >	
DBSCAN (p. 1197) (Density-Based Spatial Clustering of Applications with Noise) is a clustering technique described in the following paper:	1197
OrderedPointSelection	
This class can be used to sequentially select the next point to use for DBSCAN (p. 1197)	1200
RandomPointSelection	
This class can be used to randomly select the next point to use for DBSCAN (p. 1197)	1201
DecisionStump< MatType >	
This class implements a decision stump	1202
DTree< MatType, TagType >	
A density estimation tree is similar to both a decision tree and a space partitioning tree (like a kd-tree)	1207
PathCacher	
This class is responsible for caching the path to each node of the tree	1221
DiagonalGaussianDistribution	
A single multivariate Gaussian distribution with diagonal covariance	1226
DiscreteDistribution	
A discrete distribution where the only observations are discrete observations	1232
GammaDistribution	
This class represents the Gamma distribution	1238
GaussianDistribution	
A single multivariate Gaussian distribution	1246
LaplaceDistribution	
The multivariate Laplace distribution centered at 0 has pdf	1251
RegressionDistribution	
A class that represents a univariate conditionally Gaussian distribution	1257
DTBRules< MetricType, TreeType >	1264
DTBStat	
A statistic for use with mlpac trees, which stores the upper bound on distance to nearest neighbors and the component which this node belongs to	1269
DualTreeBoruvka< MetricType, MatType, TreeType >	
Performs the MST calculation using the Dual-Tree Boruvka algorithm, using any type of tree	1272
EdgePair	
An edge pair is simply two indices and a distance	1275
UnionFind	
A Union-Find data structure	1278
FastMKS< KernelType, MatType, TreeType >	
An implementation of fast exact max-kernel search	1280
FastMKSMModel	
A utility struct to contain all the possible FastMKS (p. 1280) models, for use by the mlpac_fastmks program	1291
FastMKSRules< KernelType, TreeType >	
The FastMKSRules (p. 1298) class is a template helper class used by FastMKS (p. 1280) class when performing exact max-kernel search	1298
FastMKStat	
The statistic used in trees with FastMKS (p. 1280)	1303
DiagonalConstraint	
Force a covariance matrix to be diagonal	1306
DiagonalGMM	
A Diagonal Gaussian Mixture Model	1308
EigenvalueRatioConstraint	
Given a vector of eigenvalue ratios, ensure that the covariance matrix always has those eigenvalue ratios	1317

EMFit < InitialClusteringType , CovarianceConstraintPolicy , Distribution >	
This class contains methods which can fit a GMM (p. 1323) to observations using the EM algorithm	1318
GMM	
A Gaussian Mixture Model (GMM (p. 1323))	1323
NoConstraint	
This class enforces no constraint on the covariance matrix	1332
PositiveDefiniteConstraint	
Given a covariance matrix, force the matrix to be positive definite	1334
HMM < Distribution >	
A class that represents a Hidden Markov Model with an arbitrary type of emission distribution	1335
HMMModel	
A serializable HMM (p. 1335) model that also stores the type	1350
HMMRegression	
A class that represents a Hidden Markov Model Regression (HMMR)	1354
CVFunction < CVType , MLAlgorithm , TotalArgs , BoundArgs >	
This wrapper serves for adapting the interface of the cross-validation classes to the one that can be utilized by the mlpack optimizers	1361
DeduceHyperParameterTypes < Args >	
A type function for deducing types of hyper-parameters from types of arguments in the Optimize method in HyperParameterTuner (p. 1375)	1365
DeduceHyperParameterTypes < Args >:: ResultHolder < HPTypes >	1366
DeduceHyperParameterTypes < PreFixedArg < T >, Args... >	
Defining DeduceHyperParameterTypes (p. 1365) for the case when not all argument types have been processed, and the next one is the type of an argument that should be fixed	1366
DeduceHyperParameterTypes < PreFixedArg < T >, Args... >:: ResultHolder < HPTypes >	1367
DeduceHyperParameterTypes < T , Args... >	
Defining DeduceHyperParameterTypes (p. 1365) for the case when not all argument types have been processed, and the next one (T) is a collection type or an arithmetic type	1368
DeduceHyperParameterTypes < T , Args... >:: IsCollectionType < Type >	
A type function to check whether Type is a collection type (for that it should define value_type)	1369
DeduceHyperParameterTypes < T , Args... >:: ResultHolder < HPTypes >	1371
DeduceHyperParameterTypes < T , Args... >:: ResultHPType < ArgumentType , IsArithmetic >	
A type function to deduce the result hyper-parameter type for ArgumentType	1372
DeduceHyperParameterTypes < T , Args... >:: ResultHPType < ArithmeticType , true >	1372
DeduceHyperParameterTypes < T , Args... >:: ResultHPType < CollectionType , false >	1373
FixedArg < T , I >	
A struct for storing information about a fixed argument	1373
HyperParameterTuner < MLAlgorithm , Metric , CV , OptimizerType , MatType , PredictionsType , WeightsType >	
The class HyperParameterTuner (p. 1375) for the given MLAlgorithm utilizes the provided Optimizer to find the values of hyper-parameters that optimize the value of the given Metric	1375
IsPreFixedArg < T >	
A type function for checking whether the given type is PreFixedArg (p. 1381)	1380
PreFixedArg < T >	
A struct for marking arguments as ones that should be fixed (it can be useful for the Optimize method of HyperParameterTuner (p. 1375))	1381
PreFixedArg < T & >	
The specialization of the template for references	1382
DeleteVisitor	1383
DualBiKDE	
DualBiKDE (p. 1384) computes a Kernel Density Estimation on the given KDEType	1384
DualMonoKDE	
DualMonoKDE (p. 1386) computes a Kernel Density Estimation on the given KDEType	1386

KDE < KernelType , MetricType , MatType , TreeType , DualTreeTraversalType , SingleTreeTraversalType >	
The KDE (p. 1387) class is a template class for performing Kernel Density Estimations	1387
KDEModel	1397
KDERules < MetricType , KernelType , TreeType >	
A dual-tree traversal Rules class for kernel density estimation	1404
KDEStat	
Extra data for each node in the tree for the task of kernel density estimation	1408
KernelNormalizer	
KernelNormalizer (p. 1410) holds a set of methods to normalize estimations applying in each case the appropriate kernel normalizer function	1410
ModeVisitor	
ModeVisitor (p. 1412) exposes the Mode() method of the KDEType	1412
TrainVisitor	
TrainVisitor (p. 1413) trains a given KDEType using a reference set	1413
CauchyKernel	
The Cauchy kernel	1414
CosineDistance	
The cosine distance (or cosine similarity)	1416
EpanechnikovKernel	
The Epanechnikov kernel, defined as	1417
ExampleKernel	
An example kernel function	1421
GaussianKernel	
The standard Gaussian kernel	1424
HyperbolicTangentKernel	
Hyperbolic tangent kernel	1430
KernelTraits < KernelType >	
This is a template class that can provide information about various kernels	1433
KernelTraits < CauchyKernel >	
Kernel traits for the Cauchy kernel	1434
KernelTraits < CosineDistance >	
Kernel traits for the cosine distance	1435
KernelTraits < EpanechnikovKernel >	
Kernel traits for the Epanechnikov kernel	1436
KernelTraits < GaussianKernel >	
Kernel traits for the Gaussian kernel	1437
KernelTraits < LaplacianKernel >	
Kernel traits of the Laplacian kernel	1438
KernelTraits < SphericalKernel >	
Kernel traits for the spherical kernel	1439
KernelTraits < TriangularKernel >	
Kernel traits for the triangular kernel	1440
KMeansSelection < ClusteringType , maxIterations >	
Implementation of the kmeans sampling scheme	1441
LaplacianKernel	
The standard Laplacian kernel	1442
LinearKernel	
The simple linear kernel (dot product)	1446
NystroemMethod < KernelType , PointSelectionPolicy >	
.	1448
OrderedSelection	1450
PolynomialKernel	
The simple polynomial kernel	1451

PSpectrumStringKernel	
The p-spectrum string kernel	1454
RandomSelection	1457
SphericalKernel	
The spherical kernel, which is 1 when the distance between the two argument points is less than or equal to the bandwidth, or 0 otherwise	1458
TriangularKernel	
The trivially simple triangular kernel, defined by	1461
AllowEmptyClusters	
Policy which allows K-Means to create empty clusters without any error being reported	1464
DualTreeKMeans< MetricType, MatType, TreeType >	
An algorithm for an exact Lloyd iteration which simply uses dual-tree nearest-neighbor search to find the nearest centroid for each point in the dataset	1466
DualTreeKMeansRules< MetricType, TreeType >	1469
DualTreeKMeansStatistic	1473
ElkanKMeans< MetricType, MatType >	1478
HamerlyKMeans< MetricType, MatType >	1479
KillEmptyClusters	
Policy which allows K-Means to "kill" empty clusters without any error being reported	1481
KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy, LloydStepType, MatType >	
This class implements K-Means clustering, using a variety of possible implementations of Lloyd's algorithm	1483
MaxVarianceNewCluster	
When an empty cluster is detected, this class takes the point furthest from the centroid of the cluster with maximum variance as a new cluster	1489
NaiveKMeans< MetricType, MatType >	
This is an implementation of a single iteration of Lloyd's algorithm for k-means	1491
PellegMooreKMeans< MetricType, MatType >	
An implementation of Pelleg-Moore's 'blacklist' algorithm for k-means clustering	1493
PellegMooreKMeansRules< MetricType, TreeType >	
The rules class for the single-tree Pelleg-Moore kd-tree traversal for k-means clustering	1495
PellegMooreKMeansStatistic	
A statistic for trees which holds the blacklist for Pelleg-Moore k-means clustering (which represents the clusters that cannot possibly own any points in a node)	1498
RandomPartition	
A very simple partitioner which partitions the data randomly into the number of desired clusters	1500
RefinedStart	
A refined approach for choosing initial points for k-means clustering	1502
SampleInitialization	1506
KernelPCA< KernelType, KernelRule >	
This class performs kernel principal components analysis (Kernel PCA), for a given kernel	1507
NaiveKernelRule< KernelType >	1511
NystroemKernelRule< KernelType, PointSelectionPolicy >	1512
LocalCoordinateCoding	
An implementation of Local Coordinate Coding (LCC) that codes data which approximately lives on a manifold using a variation of l1-norm regularized sparse coding; in LCC, the penalty on the absolute value of each point's coefficient for each atom is weighted by the squared distance of that point to that atom	1513
Constraints< MetricType >	
Interface for generating distance based constraints on a given dataset, provided corresponding true labels and a quantity parameter (k) are specified	1520
LMNN< MetricType, OptimizerType >	
An implementation of Large Margin nearest neighbor metric learning technique	1526

LMNNFunction < MetricType >	
The Large Margin Nearest Neighbors function	1531
Log	
Provides a convenient way to give formatted output	1538
ColumnsToBlocks	
Transform the columns of the given matrix into a block format	1541
RangeType < T >	
Simple real-valued range	1549
MatrixCompletion	
This class implements the popular nuclear norm minimization heuristic for matrix completion problems	1558
MeanShift < UseKernel , KernelType , MatType >	
This class implements mean shift clustering	1561
IPMetric < KernelType >	
The inner product metric, IPMetric (p. 1565), takes a given Mercer kernel (KernelType), and when Evaluate() (p. 1567) is called, returns the distance between the two points in kernel space:	1565
LMetric < TPower , TTakeRoot >	
The L _p metric for arbitrary integer p, with an option to take the root	1569
MahalanobisDistance < TakeRoot >	
The Mahalanobis distance, which is essentially a stretched Euclidean distance	1572
NaiveBayesClassifier < ModelMatType >	
The simple Naive Bayes classifier	1576
NCA < MetricType , OptimizerType >	
An implementation of Neighborhood Components Analysis, both a linear dimensionality reduction technique and a distance learning technique	1583
SoftmaxErrorFunction < MetricType >	
The "softmax" stochastic neighbor assignment probability function	1586
AlphaVisitor	
Exposes the Alpha() method of the given RAType	1590
BiSearchVisitor < SortPolicy >	
BiSearchVisitor (p. 1591) executes a bichromatic neighbor search on the given NSType	1591
DeleteVisitor	
DeleteVisitor (p. 1595) deletes the given NSType instance	1595
DrusillaSelect < MatType >	1597
EpsilonVisitor	
EpsilonVisitor (p. 1600) exposes the Epsilon method of the given NSType	1600
FirstLeafExactVisitor	
Exposes the FirstLeafExact() method of the given RAType	1601
FurthestNS	
This class implements the necessary methods for the SortPolicy template parameter of the NeighborSearch (p. 1627) class	1602
LSHSearch < SortPolicy >	
The LSHSearch (p. 1609) class; this class builds a hash on the reference set and uses this hash to compute the distance-approximate nearest-neighbors of the given queries	1609
MonoSearchVisitor	
MonoSearchVisitor (p. 1618) executes a monochromatic neighbor search on the given NSType	1618
NaiveVisitor	
NaiveVisitor (p. 1620) exposes the Naive() method of the given RAType	1620
NearestNS	
This class implements the necessary methods for the SortPolicy template parameter of the NeighborSearch (p. 1627) class	1621

NeighborSearch < SortPolicy , MetricType , MatType , TreeType , DualTreeTraversalType , SingleTreeTraversalType >	
The NeighborSearch (p. 1627) class is a template class for performing distance-based neighbor searches	1627
NeighborSearchRules < SortPolicy , MetricType , TreeType >	
The NeighborSearchRules (p. 1640) class is a template helper class used by NeighborSearch (p. 1627) class when performing distance-based neighbor searches	1640
NeighborSearchRules < SortPolicy , MetricType , TreeType >:: CandidateCmp	
Compare two candidates based on the distance	1652
NeighborSearchStat < SortPolicy >	
Extra data for each node in the tree	1653
NSModel < SortPolicy >	
The NSModel (p. 1657) class provides an easy way to serialize a model, abstracts away the different types of trees, and also reflects the NeighborSearch (p. 1627) API	1657
QDAFN < MatType >	1666
RAModel < SortPolicy >	
The RAModel (p. 1669) class provides an abstraction for the RASearch (p. 1681) class, abstracting away the TreeType parameter and allowing it to be specified at runtime in this class	1669
RAQueryStat < SortPolicy >	
Extra data for each node in the tree	1678
RASearch < SortPolicy , MetricType , MatType , TreeType >	
The RASearch (p. 1681) class: This class provides a generic manner to perform rank-approximate search via random-sampling	1681
RASearchRules < SortPolicy , MetricType , TreeType >	
The RASearchRules (p. 1692) class is a template helper class used by RASearch (p. 1681) class when performing rank-approximate search via random-sampling	1692
RAUtil	1699
ReferenceSetVisitor	
ReferenceSetVisitor (p. 1701) exposes the referenceSet of the given NSType	1701
SampleAtLeavesVisitor	
Exposes the SampleAtLeaves() method of the given RAType	1702
SearchModeVisitor	
SearchModeVisitor (p. 1703) exposes the SearchMode() method of the given NSType	1703
SingleModeVisitor	
Exposes the SingleMode() method of the given RAType	1704
SingleSampleLimitVisitor	
Exposes the SingleSampleLimit() method of the given RAType	1705
TauVisitor	
Exposes the Tau() method of the given RAType	1706
TrainVisitor < SortPolicy >	
TrainVisitor (p. 1707) sets the reference set to a new reference set on the given NSType	1707
SparseAutoencoder	
A sparse autoencoder is a neural network whose aim to learn compressed representations of the data, typically for dimensionality reduction, with a constraint on the activity of the neurons in the network	1711
SparseAutoencoderFunction	
This is a class for the sparse autoencoder objective function	1717
ExactSVDPolicy	
Implementation of the exact SVD policy	1722
PCA < DecompositionPolicy >	
This class implements principal components analysis (PCA (p. 1723))	1723
QUICSVDPolicy	
Implementation of the QUIC-SVD policy	1727

RandomizedBlockKrylovSVDPolicy	
Implementation of the randomized block krylov SVD policy	1730
RandomizedSVDPolicy	
Implementation of the randomized SVD policy	1732
Perceptron< LearnPolicy, WeightInitializationPolicy, MatType >	
This class implements a simple perceptron (i.e., a single layer neural network)	1735
RandomInitialization	
This class is used to initialize weights for the weightVectors matrix in a random manner	1740
SimpleWeightUpdate	1741
ZeroInitialization	
This class is used to initialize the matrix weightVectors to zero	1742
Radical	
An implementation of RADICAL, an algorithm for independent component analysis (ICA)	1744
BiSearchVisitor	
BiSearchVisitor (p. 1748) executes a bichromatic range search on the given RSType	1748
DeleteVisitor	
DeleteVisitor (p. 1751) deletes the given RSType instance	1751
MonoSearchVisitor	
MonoSearchVisitor (p. 1752) executes a monochromatic range search on the given RSType	1752
NaiveVisitor	
NaiveVisitor (p. 1753) exposes the Naive() method of the given RSType	1753
RangeSearch< MetricType, MatType, TreeType >	
The RangeSearch (p. 1754) class is a template class for performing range searches	1754
RangeSearchRules< MetricType, TreeType >	
The RangeSearchRules (p. 1764) class is a template helper class used by RangeSearch (p. 1754) class when performing range searches	1764
RangeSearchStat	
Statistic class for RangeSearch (p. 1754), to be set to the StatisticType of the tree type that range search is being performed with	1768
ReferenceSetVisitor	
ReferenceSetVisitor (p. 1771) exposes the referenceSet of the given RSType	1771
RSModel	1772
SingleModeVisitor	
SingleModeVisitor (p. 1779) exposes the SingleMode() method of the given RSType	1779
TrainVisitor	
TrainVisitor (p. 1780) sets the reference set to a new reference set on the given RSType	1780
LARS	
An implementation of LARS (p. 1783), a stage-wise homotopy-based algorithm for l1-regularized linear regression (LASSO) and l1+l2 regularized linear regression (Elastic Net)	1783
LinearRegression	
A simple linear regression algorithm using ordinary least squares	1789
LogisticRegression< MatType >	
The LogisticRegression (p. 1795) class implements an L2-regularized logistic regression model, and supports training with multiple optimizers and classification	1795
LogisticRegressionFunction< MatType >	
The log-likelihood function for the logistic regression objective function	1803
SoftmaxRegression	
Softmax Regression is a classifier which can be used for classification when the data available can take two or more class values	1811
SoftmaxRegressionFunction	1818
Acrobot	
Implementation of Acrobot (p. 1825) game	1825
Acrobot::State	1831
AggregatedPolicy< PolicyType >	1835

AsyncLearning < WorkerType , EnvironmentType , NetworkType , UpdaterType , PolicyType >	
Wrapper of various asynchronous learning algorithms, e.g	1837
CartPole	
Implementation of Cart Pole task	1842
CartPole::State	
Implementation of the state of Cart Pole	1846
ContinuousMountainCar	
Implementation of Continuous Mountain Car task	1850
ContinuousMountainCar::Action	
Implementation of action of Continuous Mountain Car	1854
ContinuousMountainCar::State	
Implementation of state of Continuous Mountain Car	1855
GreedyPolicy < EnvironmentType >	
Implementation for epsilon greedy policy	1858
MountainCar	
Implementation of Mountain Car task	1861
MountainCar::State	
Implementation of state of Mountain Car	1865
NStepQLearningWorker < EnvironmentType , NetworkType , UpdaterType , PolicyType >	
Forward declaration of NStepQLearningWorker (p. 1868)	1868
OneStepQLearningWorker < EnvironmentType , NetworkType , UpdaterType , PolicyType >	
Forward declaration of OneStepQLearningWorker (p. 1871)	1871
OneStepSarsaWorker < EnvironmentType , NetworkType , UpdaterType , PolicyType >	
Forward declaration of OneStepSarsaWorker (p. 1875)	1875
Pendulum	
Implementation of Pendulum (p. 1878) task	1878
Pendulum::Action	
Implementation of action of Pendulum (p. 1878)	1881
Pendulum::State	
Implementation of state of Pendulum (p. 1878)	1882
QLearning < EnvironmentType , NetworkType , UpdaterType , PolicyType , ReplayType >	
Implementation of various Q-Learning algorithms, such as DQN, double DQN	1885
RandomReplay < EnvironmentType >	
Implementation of random experience replay	1890
RewardClipping < EnvironmentType >	
Interface for clipping the reward to some value between the specified maximum and minimum value	
(Clipping here is implemented as $g_{clipped} = \max(g_{min}, \min(g_{min}, g))$.)	1894
TrainingConfig	1901
MethodFormDetector < Class , MethodForm , AdditionalArgsCount >	1907
MethodFormDetector < Class , MethodForm , 0 >	1907
MethodFormDetector < Class , MethodForm , 1 >	1908
MethodFormDetector < Class , MethodForm , 2 >	1908
MethodFormDetector < Class , MethodForm , 3 >	1909
MethodFormDetector < Class , MethodForm , 4 >	1910
MethodFormDetector < Class , MethodForm , 5 >	1910
MethodFormDetector < Class , MethodForm , 6 >	1911
MethodFormDetector < Class , MethodForm , 7 >	1912
SigCheck < U , U >	
Utility struct for checking signatures	1912
DataDependentRandomInitializer	
A data-dependent random dictionary initializer for SparseCoding (p. 1916)	1913
NothingInitializer	
A DictionaryInitializer for SparseCoding (p. 1916) which does not initialize anything; it is useful for when the dictionary is already known and will be set with SparseCoding::Dictionary() (p. 1920)	1914

RandomInitializer	
A DictionaryInitializer for use with the SparseCoding (p. 1916) class	1915
SparseCoding	
An implementation of Sparse Coding with Dictionary Learning that achieves sparsity via an l1-norm regularizer on the codes (LASSO) or an (l1+l2)-norm regularizer on the codes (the Elastic Net)	1916
BiasSVD< OptimizerType >	
Bias SVD is an improvement on Regularized SVD which is a matrix factorization techniques	1925
BiasSVDFunction< MatType >	
This class contains methods which are used to calculate the cost of BiasSVD (p. 1925)'s objective function, to calculate gradient of parameters with respect to the objective function, etc	1926
QUIC_SVD	
QUIC-SVD is a matrix factorization technique, which operates in a subspace such that A's approximation in that subspace has minimum error(A being the data matrix)	1932
RandomizedBlockKrylovSVD	
Randomized block krylov SVD is a matrix factorization that is based on randomized matrix approximation techniques, developed in in "Randomized Block Krylov Methods for Stronger and Faster Approximate Singular Value Decomposition"	1934
RandomizedSVD	
Randomized SVD is a matrix factorization that is based on randomized matrix approximation techniques, developed in in "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions"	1937
RegularizedSVD< OptimizerType >	
Regularized SVD is a matrix factorization technique that seeks to reduce the error on the training set, that is on the examples for which the ratings have been provided by the users	1943
RegularizedSVDFunction< MatType >	
The data is stored in a matrix of type MatType, so that this class can be used with both dense and sparse matrix types	1945
SVDPlusPlus< OptimizerType >	
SVD++ is a matrix decomposition technique used in collaborative filtering	1951
SVDPlusPlusFunction< MatType >	
This class contains methods which are used to calculate the cost of SVD++'s objective function, to calculate gradient of parameters with respect to the objective function, etc	1954
LinearSVM< MatType >	
The LinearSVM (p. 1959) class implements an L2-regularized support vector machine model, and supports training with multiple optimizers and classification	1959
LinearSVMFunction< MatType >	
The hinge loss function for the linear SVM objective function	1967
Timer	
The timer class provides a way for mpack methods to be timed	1975
Timers	1977
AllCategoricalSplit< FitnessFunction >	
The AllCategoricalSplit (p. 1981) is a splitting function that will split categorical features into many children: one child for each category	1981
AllCategoricalSplit< FitnessFunction >::AuxiliarySplitInfo< ElemType >	1984
AllDimensionSelect	
This dimension selection policy allows any dimension to be selected for splitting	1984
AxisParallelProjVector	
AxisParallelProjVector (p. 1986) defines an axis-parallel projection vector	1986
BestBinaryNumericSplit< FitnessFunction >	
The BestBinaryNumericSplit (p. 1989) is a splitting function for decision trees that will exhaustively search a numeric dimension for the best binary split	1989
BestBinaryNumericSplit< FitnessFunction >::AuxiliarySplitInfo< ElemType >	1991

BinaryNumericSplit < FitnessFunction , ObservationType >	
The BinaryNumericSplit (p. 1992) class implements the numeric feature splitting strategy devised by Gama, Rocha, and Medas in the following paper:	1992
BinaryNumericSplitInfo < ObservationType >	1996
BinarySpaceTree < MetricType , StatisticType , MatType , BoundType , SplitType >	
A binary space partitioning tree, such as a KD-tree or a ball tree	1998
BinarySpaceTree < MetricType , StatisticType , MatType , BoundType , SplitType >:: BreadthFirstDualTreeTraverser < RuleType >	2019
BinarySpaceTree < MetricType , StatisticType , MatType , BoundType , SplitType >:: DualTreeTraverser < RuleType >	
A dual-tree traverser for binary space trees; see <code>dual_tree_traverser.hpp</code>	2022
BinarySpaceTree < MetricType , StatisticType , MatType , BoundType , SplitType >:: SingleTreeTraverser < RuleType >	
A single-tree traverser for binary space trees; see <code>single_tree_traverser.hpp</code> for implementation	2026
CategoricalSplitInfo	2028
CompareCosineNode	2029
CosineTree	2030
CoverTree < MetricType , StatisticType , MatType , RootPointPolicy >	
A cover tree is a tree specifically designed to speed up nearest-neighbor computation in high-dimensional spaces	2040
CoverTree < MetricType , StatisticType , MatType , RootPointPolicy >:: DualTreeTraverser < RuleType >	
A dual-tree cover tree traverser; see <code>dual_tree_traverser.hpp</code>	2060
CoverTree < MetricType , StatisticType , MatType , RootPointPolicy >:: SingleTreeTraverser < RuleType >	
A single-tree cover tree traverser; see <code>single_tree_traverser.hpp</code> for implementation	2063
DecisionTree < FitnessFunction , NumericSplitType , CategoricalSplitType , DimensionSelectionType , ElemType , NoRecursion >	
This class implements a generic decision tree learner	2065
DiscreteHilbertValue < TreeElemType >	
The DiscreteHilbertValue (p. 2079) class stores Hilbert values for all of the points in a Rectangle < Tree (p. 2207) node, and calculates Hilbert values for new points	2079
EmptyStatistic	
Empty statistic if you are not interested in storing statistics in your tree	2080
ExampleTree < MetricType , StatisticType , MatType >	
This is not an actual space tree but instead an example tree that exists to show and document all the functions that mlpack trees must implement	2081
FirstPointIsRoot	
This class is meant to be used as a choice for the policy class RootPointPolicy of the CoverTree (p. 2040) class	2089
GiniGain	
The Gini gain, a measure of set purity usable as a fitness function (FitnessFunction) for decision tree	2090
GiniImpurity	2092
GreedySingleTreeTraverser < TreeType , RuleType >	2093
HilbertRTreeAuxiliaryInformation < TreeType , HilbertValueType >	2095
HilbertRTreeDescentHeuristic	
This class chooses the best child of a node in a Hilbert R tree when inserting a new point	2101
HilbertRTreeSplit < splitOrder >	
The splitting procedure for the Hilbert R tree	2102
HoeffdingCategoricalSplit < FitnessFunction >	
This is the standard Hoeffding-bound categorical feature proposed in the paper below:	2104
HoeffdingNumericSplit < FitnessFunction , ObservationType >	
The HoeffdingNumericSplit (p. 2108) class implements the numeric feature splitting strategy al- luded to by Domingos and Hulten in the following paper:	2108

HoeffdingTree < FitnessFunction , NumericSplitType , CategoricalSplitType >	
The HoeffdingTree (p.2113) object represents all of the necessary information for a Hoeffding-bound-based decision tree	2113
HoeffdingTreeModel	
This class is a serializable Hoeffding tree model that can hold four different types of Hoeffding trees	2126
HyperplaneBase < BoundT , ProjVectorT >	
HyperplaneBase (p.2133) defines a splitting hyperplane based on a projection vector and projection value	2133
InformationGain	
The standard information gain criterion, used for calculating gain in decision trees	2138
IsSpillTree < TreeType >	2140
IsSpillTree < tree::SpillTree < MetricType , StatisticType , MatType , HyperplaneType , SplitType > >	2141
MeanSpaceSplit < MetricType , MatType >	2141
MeanSplit < BoundType , MatType >	
A binary space partitioning tree node is split into its left and right child	2142
MeanSplit < BoundType , MatType >:: SplitInfo	
An information about the partition	2145
MidpointSpaceSplit < MetricType , MatType >	2146
MidpointSplit < BoundType , MatType >	
A binary space partitioning tree node is split into its left and right child	2147
MidpointSplit < BoundType , MatType >:: SplitInfo	
A struct that contains an information about the split	2150
MinimalCoverageSweep < SplitPolicy >	
The MinimalCoverageSweep (p.2151) class finds a partition along which we can split a node according to the coverage of two resulting nodes	2151
MinimalCoverageSweep < SplitPolicy >:: SweepCost < TreeType >	
A struct that provides the type of the sweep cost	2154
MinimalSplitsNumberSweep < SplitPolicy >	
The MinimalSplitsNumberSweep (p.2155) class finds a partition along which we can split a node according to the number of required splits of the node	2155
MinimalSplitsNumberSweep < SplitPolicy >:: SweepCost < typename >	
A struct that provides the type of the sweep cost	2157
MultipleRandomDimensionSelect	
This dimension selection policy allows the selection from a few random dimensions	2158
NoAuxiliaryInformation < TreeType >	2160
NumericSplitInfo < ObservationType >	2165
Octree < MetricType , StatisticType , MatType >	2167
Octree < MetricType , StatisticType , MatType >:: DualTreeTraverser < MetricType , StatisticType , MatType >	
A dual-tree traverser; see <code>dual_tree_traverser.hpp</code>	2184
Octree < MetricType , StatisticType , MatType >:: SingleTreeTraverser < RuleType >	
A single-tree traverser; see <code>single_tree_traverser.hpp</code>	2187
Octree < MetricType , StatisticType , MatType >:: SplitType::SplitInfo	2189
ProjVector	
ProjVector (p.2190) defines a general projection vector (not necessarily axis-parallel)	2190
QueueFrame < TreeType , TraversallInfoType >	2193
RandomDimensionSelect	
This dimension selection policy only selects one single random dimension	2194
RandomForest < FitnessFunction , DimensionSelectionType , NumericSplitType , CategoricalSplitType , ElemType >	2196
RectangleTree < MetricType , StatisticType , MatType , SplitType , DescentType , AuxiliaryInformationType >	
A rectangle type tree tree, such as an R-tree or X-tree	2207

RectangleTree < MetricType , StatisticType , MatType , SplitType , DescentType , AuxiliaryInformation < Type >:: DualTreeTraverser < MetricType , StatisticType , MatType , SplitType , DescentType , AuxiliaryInformationType >	
A dual tree traverser for rectangle type trees	2234
RectangleTree < MetricType , StatisticType , MatType , SplitType , DescentType , AuxiliaryInformation < Type >:: SingleTreeTraverser < RuleType >	
A single traverser for rectangle type trees	2237
RPlusPlusTreeAuxiliaryInformation < TreeType >	2239
RPlusPlusTreeDescentHeuristic	2246
RPlusPlusTreeSplitPolicy	
The RPlusPlusTreeSplitPolicy (p. 2247) helps to determine the subtree into which we should insert a child of an intermediate node that is being split	2247
RPlusTreeDescentHeuristic	2250
RPlusTreeSplit < SplitPolicyType , SweepType >	
The RPlusTreeSplit (p. 2251) class performs the split process of a node on overflow	2251
RPlusTreeSplitPolicy	
The RPlusPlusTreeSplitPolicy (p. 2247) helps to determine the subtree into which we should insert a child of an intermediate node that is being split	2253
RPTreeMaxSplit < BoundType , MatType >	
This class splits a node by a random hyperplane	2255
RPTreeMaxSplit < BoundType , MatType >:: SplitInfo	
An information about the partition	2260
RPTreeMeanSplit < BoundType , MatType >	
This class splits a binary space tree	2261
RPTreeMeanSplit < BoundType , MatType >:: SplitInfo	
An information about the partition	2264
RStarTreeDescentHeuristic	
When descending a RectangleTree (p. 2207) to insert a point, we need to have a way to choose a child node when the point isn't enclosed by any of them	2266
RStarTreeSplit	
A Rectangle Tree has new points inserted at the bottom	2267
RTreeDescentHeuristic	
When descending a RectangleTree (p. 2207) to insert a point, we need to have a way to choose a child node when the point isn't enclosed by any of them	2269
RTreeSplit	
A Rectangle Tree has new points inserted at the bottom	2271
SpaceSplit < MetricType , MatType >	2272
SpillTree < MetricType , StatisticType , MatType , HyperplaneType , SplitType >	
A hybrid spill tree is a variant of binary space trees in which the children of a node can "spill over" each other, and contain shared datapoints	2274
SpillTree < MetricType , StatisticType , MatType , HyperplaneType , SplitType >:: SpillDualTree < Traverser < MetricType , StatisticType , MatType , HyperplaneType , SplitType >	
A generic dual-tree traverser for hybrid spill trees; see spill_dual_tree_traverser.hpp (p. 2683) for implementation	2293
SpillTree < MetricType , StatisticType , MatType , HyperplaneType , SplitType >:: SpillSingleTree < Traverser < MetricType , StatisticType , MatType , HyperplaneType , SplitType >	
A generic single-tree traverser for hybrid spill trees; see spill_single_tree_traverser.hpp (p. 2684) for implementation	2297
TraversalInfo < TreeType >	
The TraversalInfo (p. 2299) class holds traversal information which is used in dual-tree (and single- tree) traversals	2299
TreeTraits < TreeType >	
The TreeTraits (p. 2302) class provides compile-time information on the characteristics of a given tree type	2302

TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::BallBound, SplitType > >	
This is a specialization of the TreeType class to the BallTree tree type	2305
TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::CellBound, SplitType > >	
This is a specialization of the TreeType class to the UBTREE tree type	2307
TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::HollowBallBound, SplitType > >	
This is a specialization of the TreeType class to an arbitrary tree with HollowBallBound (currently only the vantage point tree is supported)	2309
TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, RPTreeMaxSplit > >	
This is a specialization of the TreeType class to the max-split random projection tree	2311
TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, RPTreeMeanSplit > >	
This is a specialization of the TreeType class to the mean-split random projection tree	2313
TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > >	
This is a specialization of the TreeTraits (p. 2302) class to the BinarySpaceTree (p. 1998) tree type	2316
TreeTraits< CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > >	
The specialization of the TreeTraits (p. 2302) class for the CoverTree (p. 2040) tree type	2318
TreeTraits< Octree< MetricType, StatisticType, MatType > >	
This is a specialization of the TreeTraits (p. 2302) class to the Octree (p. 2167) tree type	2321
TreeTraits< RectangleTree< MetricType, StatisticType, MatType, RPlusTreeSplit< SplitPolicyType, SweepType >, DescentType, AuxiliaryInformationType > >	
Since the R+/R++ tree can not have overlapping children, we should define traits for the R+/R++ tree	2323
TreeTraits< RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType > >	
This is a specialization of the TreeType class to the RectangleTree (p. 2207) tree type	2326
TreeTraits< SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > >	
This is a specialization of the TreeType class to the SpillTree (p. 2274) tree type	2328
UBTreeSplit< BoundType, MatType >	
Split a node into two parts according to the median address of points contained in the node	2331
VantagePointSplit< BoundType, MatType, MaxNumSamples >	
The class splits a binary space partitioning tree node according to the median distance to the vantage point	2331
VantagePointSplit< BoundType, MatType, MaxNumSamples >::SplitInfo	
A struct that contains an information about the split	2336
XTreeAuxiliaryInformation< TreeType >	
The XTreeAuxiliaryInformation (p. 2338) class provides information specific to X trees for each node in a RectangleTree (p. 2207)	2338
XTreeAuxiliaryInformation< TreeType >::SplitHistoryStruct	
The X tree requires that the tree records it's "split history"	2345
XTreeSplit	
A Rectangle Tree has new points inserted at the bottom	2347
IsStdVector< T >	
Metaprogramming structure for vector detection	2349
IsStdVector< std::vector< T, A > >	
Metaprogramming structure for vector detection	2349
NullOutputStream	
Used for Log::Debug (p. 1540) when not compiled with debugging symbols	2350
ParamData	
This structure holds all of the information about a single parameter, including its value (which is set when ParseCommandLine() (p. 302) is called)	2356
PrefixedOutputStream	
Allows us to output to an ostream with a prefix at the beginning of each line, in the same way we would output to cout or cerr	2361

ProgramDoc

A static object whose constructor registers program documentation with the **CLI** (p. 1117) class . . . 2368

TrainHMMModel 2371

Chapter 37

File Index

37.1 File List

Here is a list of all files with brief descriptions:

/home/barak/src/git/debian-src/mlpack/doc/guide/	bindings.hpp	2373
/home/barak/src/git/debian-src/mlpack/doc/guide/	build.hpp	2373
/home/barak/src/git/debian-src/mlpack/doc/guide/	build_windows.hpp	2373
/home/barak/src/git/debian-src/mlpack/doc/guide/	cli_quickstart.hpp	2373
/home/barak/src/git/debian-src/mlpack/doc/guide/	cv.hpp	2373
/home/barak/src/git/debian-src/mlpack/doc/guide/	formats.hpp	2374
/home/barak/src/git/debian-src/mlpack/doc/guide/	hpt.hpp	2577
/home/barak/src/git/debian-src/mlpack/doc/guide/	iodoc.hpp	2374
/home/barak/src/git/debian-src/mlpack/doc/guide/	matrices.hpp	2374
/home/barak/src/git/debian-src/mlpack/doc/guide/	python_quickstart.hpp	2374
/home/barak/src/git/debian-src/mlpack/doc/guide/	sample.hpp	2374
/home/barak/src/git/debian-src/mlpack/doc/guide/	sample_ml_app.hpp	2374
/home/barak/src/git/debian-src/mlpack/doc/guide/	timer.hpp	2375
/home/barak/src/git/debian-src/mlpack/doc/guide/	version.hpp	2748
/home/barak/src/git/debian-src/mlpack/doc/policies/	elemtype.hpp	2375
/home/barak/src/git/debian-src/mlpack/doc/policies/	functiontype.hpp	2375
/home/barak/src/git/debian-src/mlpack/doc/policies/	kernels.hpp	2375
/home/barak/src/git/debian-src/mlpack/doc/policies/	metrics.hpp	2375
/home/barak/src/git/debian-src/mlpack/doc/policies/	trees.hpp	2375
/home/barak/src/git/debian-src/mlpack/src/mlpack/	core.hpp	
Include all of the base components required to write mlpack methods, and the main mlpack Doxygen documentation		2535
/home/barak/src/git/debian-src/mlpack/src/mlpack/	prereqs.hpp	
The core includes that mlpack expects; standard C++ includes and Armadillo		3075
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	add_to_po.hpp	2381
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	cli_option.hpp	2382
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	default_param.hpp	2456
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	delete_allocated_memory.hpp	2460
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	end_program.hpp	2462
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	get_allocated_memory.hpp	2463
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	get_param.hpp	2465

/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	get_printable_param.hpp	2470
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	get_printable_param_name.hpp	2475
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	get_printable_param_value.hpp	2478
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	get_printable_type.hpp	2481
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	get_raw_param.hpp	2485
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	map_parameter_name.hpp	2487
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	output_param.hpp	2488
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	parameter_type.hpp	2490
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	parse_command_line.hpp	2491
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	print_doc_functions.hpp	2492
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	print_help.hpp	2496
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	print_type_doc.hpp	2497
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	set_param.hpp	2501
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/	string_type_param.hpp	2503
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/	binding_info.hpp	2504
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/	default_param.hpp	2458
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/	get_binding_name.hpp	2505
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/	get_param.hpp	2467
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/	get_printable_param.hpp	2471
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/	get_printable_param_name.hpp	2477
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/	get_printable_param_value.hpp	2479
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/	get_printable_type.hpp	2482
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/	is_serializable.hpp	2506
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/	md_option.hpp	2507
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/	print_doc_functions.hpp	2494
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/	print_docs.hpp	2508
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/	print_type_doc.hpp	2499
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/	program_doc_wrapper.hpp	2509
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	default_param.hpp	2459
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	get_arma_type.hpp	2510
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	get_cython_type.hpp	2512
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	get_numpy_type.hpp	2513
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	get_numpy_type_char.hpp	2515
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	get_param.hpp	2468
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	get_printable_param.hpp	2472
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	get_printable_type.hpp	2483
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	import_decl.hpp	2516
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	print_class_defn.hpp	2522
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	print_defn.hpp	2524
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	print_doc.hpp	2525
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	print_doc_functions.hpp	2495
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	print_input_processing.hpp	2526
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	print_output_processing.hpp	2528
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	print_pyx.hpp	2530
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	print_type_doc.hpp	2500
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	py_option.hpp	2530
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/	strip_type.hpp	2531
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/mlpack/	arma_util.hpp	2518
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/mlpack/	cli_util.hpp	2519
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/mlpack/	serialization.hpp	2521
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/	clean_memory.hpp	2532
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/	delete_allocated_memory.hpp	2461
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/	get_allocated_memory.hpp	2464
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/	get_param.hpp	2469

/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/ get_printable_param.hpp	2474
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/ ignore_check.hpp	2533
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/ test_option.hpp	2534
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ cv_base.hpp	2535
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ k_fold_cv.hpp	2536
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ meta_info_extractor.hpp	2537
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ simple_cv.hpp	2544
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/ accuracy.hpp	2539
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/ average_strategy.hpp	2539
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/ f1.hpp	2540
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/ facilities.hpp	2541
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/ mse.hpp	2542
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/ precision.hpp	2543
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/ recall.hpp	2543
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ binarize.hpp	2545
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ confusion_matrix.hpp	2546
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ dataset_mapper.hpp	2547
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ extension.hpp	2548
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ format.hpp	2549
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ has_serialize.hpp	2549
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ imputer.hpp	2553
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ is_naninf.hpp	2554
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ load.hpp	2555
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ load_arff.hpp	2556
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ load_csv.hpp	2557
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ normalize_labels.hpp	2561
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ one_hot_encoding.hpp	2562
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ save.hpp	2563
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ serialization_template_version.hpp	2564
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ split_data.hpp	2565
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/ custom_imputation.hpp	2550
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/ listwise_deletion.hpp	2551
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/ mean_imputation.hpp	2552
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/ median_imputation.hpp	2553
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/map_policies/ datatype.hpp	2558
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/map_policies/ increment_policy.hpp	2559
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/map_policies/ missing_policy.hpp	2560
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/ diagonal_gaussian_distribution.hpp	2566
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/ discrete_distribution.hpp	2567
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/ gamma_distribution.hpp	2568
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/ gaussian_distribution.hpp	2569
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/ laplace_distribution.hpp	2570
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/ regression_distribution.hpp	2571
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ cv_function.hpp	2572
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ deduce_hp_types.hpp	2573
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ fixed.hpp	2574
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ hpt.hpp	2576
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ cauchy_kernel.hpp	2577
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ cosine_distance.hpp	2578
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ epanechnikov_kernel.hpp	2579
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ example_kernel.hpp	2580
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ gaussian_kernel.hpp	2581
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ hyperbolic_tangent_kernel.hpp	2582
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ kernel_traits.hpp	2583

/home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ laplacian_kernel.hpp	2583
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ linear_kernel.hpp	2584
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ polynomial_kernel.hpp	2585
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ pspectrum_string_kernel.hpp	2586
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ spherical_kernel.hpp	2587
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ triangular_kernel.hpp	2588
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/ ccov.hpp	2589
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/ clamp.hpp	
Miscellaneous math clamping routines	2590
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/ columns_to_blocks.hpp	2591
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/ lin_alg.hpp	2592
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/ log_add.hpp	2594
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/ make_alias.hpp	2594
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/ random.hpp	
Miscellaneous math random-related routines	2596
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/ random_basis.hpp	2597
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/ range.hpp	
Definition of the Range class, which represents a simple range with a lower and upper bound	2598
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/ round.hpp	2599
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/ shuffle_data.hpp	2600
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/metrics/ ip_metric.hpp	2601
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/metrics/ lmetric.hpp	2603
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/metrics/ mahalanobis_distance.hpp	2604
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ address.hpp	2604
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ ballbound.hpp	
Bounds that are useful for binary space partitioning trees	2606
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ binary_space_tree.hpp	2608
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ bound_traits.hpp	2637
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ bounds.hpp	
Bounds that are useful for binary space partitioning trees	2638
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ cellbound.hpp	2638
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ cover_tree.hpp	2642
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ enumerate_tree.hpp	2644
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ example_tree.hpp	2645
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ greedy_single_tree_traverser.hpp	2645
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ hollow_ball_bound.hpp	
Bounds that are useful for binary space partitioning trees	2646
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ hrectbound.hpp	
Bounds that are useful for binary space partitioning trees	2647
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ octree.hpp	2650
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ perform_split.hpp	2650
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ rectangle_tree.hpp	2672
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ spill_tree.hpp	2687
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ statistic.hpp	
Definition of the policy type for the statistic class	2688
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ traversal_info.hpp	2689
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ tree_traits.hpp	2690
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ binary_space_tree.hpp	2607
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ breadth_first_dual_tree_traverser.hpp	2608
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ dual_tree_traverser.hpp	2610
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ mean_split.hpp	2613
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ midpoint_split.hpp	2615
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ rp_tree_max_split.hpp	2616

/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ rp_tree_mean_split.hpp	2617
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ single_tree_traverser.hpp	2618
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ traits.hpp	2622
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ typedef.hpp	2627
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ ub_tree_split.hpp	2634
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ vantage_point_split.hpp	2636
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cosine_tree/ cosine_tree.hpp	2640
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/ cover_tree.hpp	2641
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/ dual_tree_traverser.hpp	2611
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/ first_point_is_root.hpp	2643
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/ single_tree_traverser.hpp	2619
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/ traits.hpp	2623
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/ typedef.hpp	2629
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree/ dual_tree_traverser.hpp	2611
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree/ octree.hpp	2648
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree/ single_tree_traverser.hpp	2620
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree/ traits.hpp	2624
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ discrete_hilbert_value.hpp	2651
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ dual_tree_traverser.hpp	2612
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ hilbert_r_tree_auxiliary ←	
information.hpp	2653
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ hilbert_r_tree_descent ←	
heuristic.hpp	2654
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ hilbert_r_tree_split.hpp	2655
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ minimal_coverage_sweep.hpp	2656
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ minimal_splits_number ←	
sweep.hpp	2658
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ no_auxiliary_information.hpp	2659
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ r_plus_plus_tree_auxiliary ←	
information.hpp	2660
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ r_plus_plus_tree_descent ←	
heuristic.hpp	2662
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ r_plus_plus_tree_split ←	
policy.hpp	2663
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ r_plus_tree_descent_heuristic ←	
hpp	2664
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ r_plus_tree_split.hpp	2665
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ r_plus_tree_split_policy.hpp	2666
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ r_star_tree_descent_heuristic ←	
hpp	2667
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ r_star_tree_split.hpp	2668
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ r_tree_descent_heuristic.hpp	2669
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ r_tree_split.hpp	2670
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ rectangle_tree.hpp	2671
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ single_tree_traverser.hpp	2621
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ traits.hpp	2625
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ typedef.hpp	2630
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ x_tree_auxiliary_information ←	
hpp	2673
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ x_tree_split.hpp	2673
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/ hyperplane.hpp	2674
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/ mean_space_split.hpp	2676
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/ midpoint_space_split.hpp	2678
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/ projection_vector.hpp	2679

/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/ space_split.hpp	2680
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/ is_spill_tree.hpp	
Definition of IsSpillTree	2681
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/ spill_dual_tree_traverser.hpp	2683
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/ spill_single_tree_traverser.hpp . .	2684
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/ spill_tree.hpp	2686
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/ traits.hpp	2626
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/ typedef.hpp	2631
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ arma_config.hpp	
This is an autogenerated file which contains the configuration of Armadillo at the time mlpack was built	2690
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ arma_config_check.hpp	2692
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ arma_traits.hpp	2692
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ backtrace.hpp	2693
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ cli.hpp	2694
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ deprecated.hpp	2695
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ gitversion.hpp	2696
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ hyphenate_string.hpp	2696
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ is_std_vector.hpp	2697
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ log.hpp	2698
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ mlpack_main.hpp	2699
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ nulloutstream.hpp	2702
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ param.hpp	2703
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ param_checks.hpp	2736
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ param_data.hpp	2737
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ prefixedoutstream.hpp	2739
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ program_doc.hpp	2740
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ sfnuae_utility.hpp	2741
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ timers.hpp	2745
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ version.hpp	2746
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/adaboost/ adaboost.hpp	2748
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/adaboost/ adaboost_model.hpp	2749
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/ amf.hpp	2750
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/init_rules/ average_init.hpp	2752
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/init_rules/ given_init.hpp	2752
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/init_rules/ random_acol_init.hpp	2753
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/init_rules/ random_init.hpp	2754
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/ complete_incremental_	
termination.hpp	2757
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/ incomplete_incremental_	
termination.hpp	2757
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/ max_iteration_	
termination.hpp	2758
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/ simple_residue_	
termination.hpp	2759
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/ simple_tolerance_	
termination.hpp	2760
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/ validation_rmse_	
termination.hpp	2762
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/ nmf_als.hpp	2762
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/ nmf_mult_dist.hpp	2764
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/ nmf_mult_div.hpp	2765
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/ svd_batch_learning.hpp . .	2766
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/ svd_complete_incremental_	
learning.hpp	2767

/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/ svd_incomplete_incremental_	
_learning.hpp	2769
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/ brnn.hpp	2782
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/ ffn.hpp	2788
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/ rnn.hpp	2857
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/ hard_sigmoid_	
function.hpp	2770
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/ identity_function.hpp	2771
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/ logistic_function.hpp	2772
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/ rectifier_function.hpp	2773
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/ softplus_function.hpp	2774
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/ softsign_function.hpp	2775
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/ swish_function.hpp	2776
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/ tanh_function.hpp	2777
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented/tasks/ add.hpp	2778
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented/tasks/ copy.hpp	2779
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented/tasks/ score.hpp	2780
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented/tasks/ sort.hpp	2781
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/ border_modes.hpp	2783
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/ fft_convolution.hpp	2784
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/ naive_convolution.hpp	2785
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/ svd_convolution.hpp	2786
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/dists/ bernoulli_distribution.hpp	2787
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ const_init.hpp	2789
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ gaussian_init.hpp	2790
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ glorot_init.hpp	2791
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ he_init.hpp	2792
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ init_rules_traits.hpp	2793
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ kathirvalavakumar_subavathi_	
init.hpp	2794
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ lecun_normal_init.hpp	2795
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ network_init.hpp	2796
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ nguyen_widrow_init.hpp	2797
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ oivs_init.hpp	2798
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ orthogonal_init.hpp	2799
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ random_init.hpp	2755
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ add.hpp	2779
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ add_merge.hpp	2800
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ alpha_dropout.hpp	2801
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ atrous_convolution.hpp	2802
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ base_layer.hpp	2803
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ batch_norm.hpp	2805
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ bilinear_interpolation.hpp	2806
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ c_relu.hpp	2807
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ concat.hpp	2807
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ concat_performance.hpp	2809
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ concatenate.hpp	2810
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ constant.hpp	2811
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ convolution.hpp	2812
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ dropconnect.hpp	2813
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ dropout.hpp	2814
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ elu.hpp	2815
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ fast_lstm.hpp	2816
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ flexible_relu.hpp	2817

/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ glimpse.hpp	2818
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ gru.hpp	2819
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ hard_tanh.hpp	2820
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ join.hpp	2821
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ layer.hpp	2822
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ layer_norm.hpp	2823
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ layer_traits.hpp	2824
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ layer_types.hpp	2826
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ leaky_relu.hpp	2828
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ linear.hpp	2829
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ linear_no_bias.hpp	2830
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ log_softmax.hpp	2831
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ lookup.hpp	2832
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ lstm.hpp	2833
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ max_pooling.hpp	2834
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ mean_pooling.hpp	2835
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ multiply_constant.hpp	2836
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ multiply_merge.hpp	2837
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ parametric_relu.hpp	2838
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ recurrent.hpp	2839
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ recurrent_attention.hpp	2840
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ reinforce_normal.hpp	2841
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ reparametrization.hpp	2842
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ select.hpp	2843
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ sequential.hpp	2844
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ subview.hpp	2846
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ transposed_convolution.hpp	2847
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ vr_class_reward.hpp	2848
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/ cross_entropy_error.hpp	2849
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/ dice_loss.hpp	2849
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/ earth_mover_distance.hpp	2850
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/ kl_divergence.hpp	2851
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/ mean_squared_error.hpp	2852
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/ negative_log_likelihood	2853
hpp	2853
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/ reconstruction_loss.hpp	2854
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/ sigmoid_cross_entropy	2854
error.hpp	2854
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/rbm/ rbm.hpp	2855
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/rbm/ rbm_policies.hpp	2856
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ add_visitor.hpp	2858
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ backward_visitor.hpp	2859
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ copy_visitor.hpp	2860
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ delete_visitor.hpp	2861
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ delta_visitor.hpp	2862
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ deterministic_set_visitor.hpp	2863
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ forward_visitor.hpp	2864
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ gradient_set_visitor.hpp	2865
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ gradient_update_visitor.hpp	2866
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ gradient_visitor.hpp	2867
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ gradient_zero_visitor.hpp	2867
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ load_output_parameter_visitor.hpp	2868
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ loss_visitor.hpp	2869
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ output_height_visitor.hpp	2871

/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/	output_parameter_visitor.hpp	2872
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/	output_width_visitor.hpp	2873
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/	parameters_set_visitor.hpp	2875
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/	parameters_visitor.hpp	2875
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/	reset_cell_visitor.hpp	2876
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/	reset_visitor.hpp	2877
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/	reward_set_visitor.hpp	2879
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/	run_set_visitor.hpp	2880
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/	save_output_parameter_visitor.hpp	2881
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/	set_input_height_visitor.hpp	2882
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/	set_input_width_visitor.hpp	2883
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/	weight_set_visitor.hpp	2884
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/	weight_size_visitor.hpp	2885
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/approx_kfn/	drusilla_select.hpp	2887
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/approx_kfn/	qdafn.hpp	2888
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/bias_svd/	bias_svd.hpp	2889
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/bias_svd/	bias_svd_function.hpp	2890
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/block_krylov_svd/	randomized_block_krylov_svd.hpp	2892
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/	cf.hpp	2893
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/	cf_model.hpp	2894
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/	svd_wrapper.hpp	2916
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/	batch_svd	2895
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/	bias_svd_method.hpp	2896
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/	nmf_method.hpp	2898
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/	randomized_svd	2899
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/	regularized_svd	2900
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/	svd_complete	2902
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/	svd_incomplete	2903
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/	svdplusplus	2904
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/interpolation_policies/	average_interpolation	2905
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/interpolation_policies/	regression_interpolation	2906
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/interpolation_policies/	similarity_interpolation	2907
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/neighbor_search_policies/	cosine_search.hpp	2908
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/neighbor_search_policies/	lmetric_search.hpp	2909
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/neighbor_search_policies/	pearson_search	2910
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/	combined_normalization.hpp	2911
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/	item_mean_normalization.hpp	2912
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/	no_normalization.hpp	2913
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/	overall_mean_normalization	2914
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/	user_mean_normalization.hpp	2915
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/	z_score_normalization.hpp	2915

/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/dbscan/ dbscan.hpp	2917
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/dbscan/ ordered_point_selection.hpp	2918
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/dbscan/ random_point_selection.hpp	2919
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_stump/ decision_stump.hpp	2920
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/ all_categorical_split.hpp	2921
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/ all_dimension_select.hpp	2923
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/ best_binary_numeric_split.hpp	2924
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/ decision_tree.hpp	2925
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/ gini_gain.hpp	2927
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/ information_gain.hpp	2928
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/ multiple_random_dimension_ ← select.hpp	2930
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/ random_dimension_select.hpp	2931
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/det/ dt_utils.hpp	2931
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/det/ dtree.hpp	2932
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/ dtb.hpp	2934
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/ dtb_rules.hpp	2935
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/ dtb_stat.hpp	2936
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/ edge_pair.hpp	2937
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/ union_find.hpp	2938
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/fastmks/ fastmks.hpp	2939
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/fastmks/ fastmks_model.hpp	2940
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/fastmks/ fastmks_rules.hpp	2941
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/fastmks/ fastmks_stat.hpp	2942
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/ diagonal_constraint.hpp	2943
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/ diagonal_gmm.hpp	2945
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/ eigenvalue_ratio_constraint.hpp	2946
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/ em_fit.hpp	2947
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/ gmm.hpp	2948
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/ no_constraint.hpp	2949
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/ positive_definite_constraint.hpp	2950
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hmm/ hmm.hpp	2951
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hmm/ hmm_model.hpp	2952
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hmm/ hmm_regression.hpp	2954
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hmm/ hmm_util.hpp	2954
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ binary_numeric_split.hpp	2956
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ binary_numeric_split_info.hpp	2957
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ categorical_split_info.hpp	2958
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ gini_impurity.hpp	2960
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ hoeffding_categorical_split_ ← hpp	2961
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ hoeffding_numeric_split.hpp	2963
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ hoeffding_tree.hpp	2964
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ hoeffding_tree_model.hpp	2965
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ information_gain.hpp	2929
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ numeric_split_info.hpp	2966
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ typedef.hpp	2633
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/ kde.hpp	2967
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/ kde_model.hpp	2968
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/ kde_rules.hpp	2970
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/ kde_stat.hpp	2971
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kernel_pca/ kernel_pca.hpp	2972
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kernel_pca/kernel_rules/ naive_method.hpp	2973
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kernel_pca/kernel_rules/ nystroem_method.hpp	2974

/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ allow_empty_clusters.hpp	2975
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ dual_tree_kmeans.hpp	2976
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ dual_tree_kmeans_rules.hpp	2977
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ dual_tree_kmeans_statistic.hpp	2978
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ elkan_kmeans.hpp	2979
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ hamerly_kmeans.hpp	2980
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ kill_empty_clusters.hpp	2981
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ kmeans.hpp	2981
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ max_variance_new_cluster.hpp	2983
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ naive_kmeans.hpp	2984
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ pelleg_moore_kmeans.hpp	2985
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ pelleg_moore_kmeans_rules.hpp	2986
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ pelleg_moore_kmeans_statistic.hpp	2987
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ random_partition.hpp	2988
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ refined_start.hpp	2989
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ sample_initialization.hpp	2990
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lars/ lars.hpp	2991
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear_regression/ linear_regression.hpp	2992
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear_svm/ linear_svm.hpp	2994
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear_svm/ linear_svm_function.hpp	2995
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lmnn/ constraints.hpp	2996
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lmnn/ lmnn.hpp	2997
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lmnn/ lmnn_function.hpp	2998
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/local_coordinate_coding/ lcc.hpp	2999
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/logistic_regression/ logistic_regression.hpp	3000
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/logistic_regression/ logistic_regression_ function.hpp	3001
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lsh/ lsh_search.hpp	3002
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/matrix_completion/ matrix_completion.hpp	3004
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/mean_shift/ mean_shift.hpp	3005
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/naive_bayes/ naive_bayes_classifier.hpp	3005
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nca/ nca.hpp	3006
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nca/ nca_softmax_error_function.hpp	3007
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ neighbor_search.hpp	3008
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ neighbor_search_rules.hpp	3010
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ neighbor_search_stat.hpp	3011
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ ns_model.hpp	3012
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ typedef.hpp	2633
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ unmap.hpp	3016
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/sort_policies/ furthest_ neighbor_sort.hpp	3014
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/sort_policies/ nearest_ neighbor_sort.hpp	3015
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nystroem_method/ kmeans_selection.hpp	3017
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nystroem_method/ nystroem_method.hpp	2974
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nystroem_method/ ordered_selection.hpp	3019
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nystroem_method/ random_selection.hpp	3019
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/ pca.hpp	3023
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/decomposition_policies/ exact_svd_ method.hpp	3020
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/decomposition_policies/ quic_svd_ method.hpp	3021
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/decomposition_policies/ randomized_ block_krylov_method.hpp	3022

/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/decomposition_policies/	randomized_svd ↵	
_method.hpp		2900
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/	perceptron.hpp	3026
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/initialization_methods/	random ↵	
init.hpp		2756
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/initialization_methods/	zero_init.hpp	3024
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/learning_policies/	simple_weight ↵	
update.hpp		3025
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/quic_svd/	quic_svd.hpp	3027
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/radical/	radical.hpp	3028
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/random_forest/	bootstrap.hpp	3029
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/random_forest/	random_forest.hpp	3030
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/randomized_svd/	randomized_svd.hpp	3031
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/	range_search.hpp	3032
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/	range_search_rules.hpp	3034
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/	range_search_stat.hpp	3035
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/	rs_model.hpp	3036
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/	ra_model.hpp	3038
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/	ra_query_stat.hpp	3039
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/	ra_search.hpp	3041
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/	ra_search_rules.hpp	3042
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/	ra_typedef.hpp	3043
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/	ra_util.hpp	3045
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/regularized_svd/	regularized_svd.hpp	3046
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/regularized_svd/	regularized_svd_function.hpp	3047
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/	async_learning.hpp	3049
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/	q_learning.hpp	3057
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/	training_config.hpp	3059
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/	acrobot ↵	
hpp		3050
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/	cart ↵	
pole.hpp		3051
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/	continuous ↵	
_mountain_car.hpp		3052
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/	mountain ↵	
_car.hpp		3053
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/	pendulum ↵	
hpp		3054
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/	reward ↵	
clipping.hpp		3055
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/policy/	aggregated ↵	
policy.hpp		3056
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/policy/	greedy_policy ↵	
hpp		3056
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/replay/	random_replay ↵	
hpp		3058
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/	n_step_q ↵	
learning_worker.hpp		3060
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/	one_step_q ↵	
learning_worker.hpp		3061
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/	one_step ↵	
sarsa_worker.hpp		3063
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/softmax_regression/	softmax_regression.hpp	3064

/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/softmax_regression/	softmax_regression_↵	
function.hpp		3065
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_autoencoder/	maximal_inputs.hpp	3066
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_autoencoder/	sparse_autoencoder.hpp	3066
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_autoencoder/	sparse_autoencoder_↵	
function.hpp		3067
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_coding/	data_dependent_random_↵	
initializer.hpp		3068
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_coding/	nothing_initializer.hpp	3069
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_coding/	random_initializer.hpp	3070
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_coding/	sparse_coding.hpp	3071
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/svdplusplus/	svdplusplus.hpp	3072
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/svdplusplus/	svdplusplus_function.hpp	3073
/home/barak/src/git/debian-src/mlpack/src/mlpack/tests/	ann_test_tools.hpp	3077
/home/barak/src/git/debian-src/mlpack/src/mlpack/tests/	custom_layer.hpp	3079
/home/barak/src/git/debian-src/mlpack/src/mlpack/tests/	mock_categorical_data.hpp	
Generate categorical dataset for tests		3085
/home/barak/src/git/debian-src/mlpack/src/mlpack/tests/	serialization.hpp	2521
/home/barak/src/git/debian-src/mlpack/src/mlpack/tests/	test_function_tools.hpp	3086
/home/barak/src/git/debian-src/mlpack/src/mlpack/tests/	test_tools.hpp	3087
/home/barak/src/git/debian-src/mlpack/src/mlpack/tests/main_tests/	hmm_test_utils.hpp	3081
/home/barak/src/git/debian-src/mlpack/src/mlpack/tests/main_tests/	range_search_utils.hpp	3082
/home/barak/src/git/debian-src/mlpack/src/mlpack/tests/main_tests/	test_helper.hpp	3084

Chapter 38

Namespace Documentation

38.1 boost Namespace Reference

Set the serialization version of the adaboost class.

Namespaces

- **serialization**

38.1.1 Detailed Description

Set the serialization version of the adaboost class.

Set the serialization version of the RNN class.

Set the serialization version of the FFN class.

Set the serialization version of the BRNN class.

Multiple template arguments makes this ugly...

38.2 boost::serialization Namespace Reference

Classes

- struct **version**< **mlpack::adaboost::AdaBoost**< **WeakLearnerType**, **MatType** > >
- struct **version**< **mlpack::ann::BRNN**< **OutputLayerType**, **MergeLayerType**, **MergeOutputType**, **InitializationRuleType**, **CustomLayer...** > >
- struct **version**< **mlpack::ann::FFN**< **OutputLayerType**, **InitializationRuleType**, **CustomLayer...** > >
- struct **version**< **mlpack::ann::RNN**< **OutputLayerType**, **InitializationRuleType**, **CustomLayer...** > >

38.3 ens Namespace Reference

38.4 mlpack Namespace Reference

.hpp

Namespaces

- **adaboost**
- **amf**
Alternating Matrix Factorization.
- **ann**
Artificial Neural Network.
- **bindings**
- **bound**
- **cf**
Collaborative filtering.
- **cv**
- **data**
Functions to load and save matrices and models.
- **dbscan**
- **decision_stump**
- **det**
Density Estimation Trees.
- **distribution**
Probability distributions.
- **emst**
Euclidean Minimum Spanning Trees.
- **fastmks**
Fast max-kernel search.
- **gmm**
Gaussian Mixture Models.
- **hmm**
Hidden Markov Models.
- **hpt**
- **kde**
Kernel Density Estimation.
- **kernel**
Kernel functions.
- **kmeans**
K-Means clustering.
- **kpca**
- **lcc**
- **lmnn**
Large Margin Nearest Neighbor.
- **math**

Miscellaneous math routines.

- **matrix_completion**
- **meanshift**

Mean shift clustering.

- **metric**
- **naive_bayes**

The Naive Bayes Classifier.

- **nca**

Neighborhood Components Analysis.

- **neighbor**
- **nn**
- **pca**
- **perceptron**
- **radical**
- **range**

Range-search routines.

- **regression**

Regression methods.

- **rl**
- **sfnai**
- **sparse_coding**
- **svd**
- **svm**
- **tree**

Trees and tree-building procedures.

- **util**

Classes

- class **Backtrace**

Provides a backtrace.

- class **CLI**

Parses the command line for parameters and holds user-specified parameters.

- class **Log**

Provides a convenient way to give formatted output.

- class **Timer**

The timer class provides a way for mlpack methods to be timed.

- class **Timers**

Functions

- void **CheckMatrices** (const arma::mat &x, const arma::mat &xmlX, const arma::mat &textX, const arma::mat &binaryX)
- void **CheckMatrices** (const arma::Mat< size_t > &x, const arma::Mat< size_t > &xmlX, const arma::Mat< size_t > &textX, const arma::Mat< size_t > &binaryX)
- void **CheckMatrices** (const arma::cube &x, const arma::cube &xmlX, const arma::cube &textX, const arma::cube &binaryX)

- `template<typename T , typename IArchiveType , typename OArchiveType >`
`void SerializeObject (T &t, T &newT)`
- `template<typename T >`
`void SerializeObjectAll (T &t, T &xmlT, T &textT, T &binaryT)`
- `template<typename T , typename IArchiveType , typename OArchiveType >`
`void SerializePointerObject (T *t, T *&newT)`
- `template<typename T >`
`void SerializePointerObjectAll (T *t, T *&xmlT, T *&textT, T *&binaryT)`
- `template<typename CubeType >`
`void TestAllArmadilloSerialization (arma::Cube< CubeType > &x)`
- `template<typename MatType >`
`void TestAllArmadilloSerialization (MatType &x)`
- `template<typename CubeType , typename IArchiveType , typename OArchiveType >`
`void TestArmadilloSerialization (arma::Cube< CubeType > &x)`
- `template<typename MatType , typename IArchiveType , typename OArchiveType >`
`void TestArmadilloSerialization (MatType &x)`

38.4.1 Detailed Description

.hpp

This class is used to update the weightVectors matrix according to the simple update rule as discussed by Rosenblatt:

Linear algebra utility functions, generally performed on matrices or vectors.

Author

Ryan Curtin

Simple utilities for **boost::serialization** (p. 251).

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

if a vector x has been incorrectly classified by a weight w , then $w = w - x$ and $w' = w' + x$

where w' is the weight vector which correctly classifies x .

38.4.2 Function Documentation

38.4.2.1 CheckMatrices() [1/3]

```
void mlpack::CheckMatrices (
    const arma::mat & x,
    const arma::mat & xmlX,
    const arma::mat & textX,
    const arma::mat & binaryX )
```

Referenced by `SerializePointerObjectAll()`.

38.4.2.2 CheckMatrices() [2/3]

```
void mlpack::CheckMatrices (
    const arma::Mat< size_t > & x,
    const arma::Mat< size_t > & xmlX,
    const arma::Mat< size_t > & textX,
    const arma::Mat< size_t > & binaryX )
```

38.4.2.3 CheckMatrices() [3/3]

```
void mlpack::CheckMatrices (
    const arma::cube & x,
    const arma::cube & xmlX,
    const arma::cube & textX,
    const arma::cube & binaryX )
```

38.4.2.4 SerializeObject()

```
void mlpack::SerializeObject (
    T & t,
    T & newT )
```

Definition at line 193 of file `serialization.hpp`.

References `mlpack::data::binary`, and `FilterFileName()`.

Referenced by `SerializeObjectAll()`.

38.4.2.5 SerializeObjectAll()

```
void mlpack::SerializeObjectAll (
    T & t,
    T & xmlT,
    T & textT,
    T & binaryT )
```

Definition at line 240 of file `serialization.hpp`.

References `SerializeObject()`.

38.4.2.6 SerializePointerObject()

```
void mlpack::SerializePointerObject (
    T * t,
    T *& newT )
```

Definition at line 252 of file `serialization.hpp`.

References `mlpack::data::binary`, and `FilterFileName()`.

Referenced by `SerializePointerObjectAll()`.

38.4.2.7 SerializePointerObjectAll()

```
void mlpack::SerializePointerObjectAll (
    T * t,
    T *& xmlT,
    T *& textT,
    T *& binaryT )
```

Definition at line 297 of file `serialization.hpp`.

References `CheckMatrices()`, and `SerializePointerObject()`.

38.4.2.8 TestAllArmadilloSerialization() [1/2]

```
void mlpack::TestAllArmadilloSerialization (
    arma::Cube< CubeType > & x )
```

Definition at line 106 of file `serialization.hpp`.

References `TestArmadilloSerialization()`.

38.4.2.9 TestAllArmadilloSerialization() [2/2]

```
void mlpack::TestAllArmadilloSerialization (
    MatType & x )
```

Definition at line 180 of file serialization.hpp.

References TestArmadilloSerialization().

38.4.2.10 TestArmadilloSerialization() [1/2]

```
void mlpack::TestArmadilloSerialization (
    arma::Cube< CubeType > & x )
```

Definition at line 33 of file serialization.hpp.

References mlpack::data::binary, and FilterFileName().

Referenced by TestAllArmadilloSerialization().

38.4.2.11 TestArmadilloSerialization() [2/2]

```
void mlpack::TestArmadilloSerialization (
    MatType & x )
```

Definition at line 120 of file serialization.hpp.

References mlpack::data::binary, and FilterFileName().

38.5 mlpack::adaboost Namespace Reference

Classes

- class **AdaBoost**
*The **AdaBoost** (p. 488) class.*
- class **AdaBoostModel**
The model to save to disk.

38.6 mlpack::amf Namespace Reference

Alternating Matrix Factorization.

Classes

- class **AMF**
*This class implements **AMF** (p. 499) (alternating matrix factorization) on the given matrix V .*
- class **AveragelInitialization**
This initialization rule initializes matrix W and H to root of the average of V , perturbed with uniform noise.
- class **CompleteIncrementalTermination**
*This class acts as a wrapper for basic termination policies to be used by **SVDCompleteIncrementalLearning** (p. 542).*
- class **GivenInitialization**
*This initialization rule for **AMF** (p. 499) simply fills the W and H matrices with the matrices given to the constructor of this object.*
- class **IncompleteIncrementalTermination**
*This class acts as a wrapper for basic termination policies to be used by **SVDIncompleteIncrementalLearning** (p. 547).*
- class **MaxIterationTermination**
This termination policy only terminates when the maximum number of iterations has been reached.
- class **NMFALSUpdate**
This class implements a method titled 'Alternating Least Squares' described in the following paper:
- class **NMFMultiplicativeDistanceUpdate**
The multiplicative distance update rules for matrices W and H .
- class **NMFMultiplicativeDivergenceUpdate**
This follows a method described in the paper 'Algorithms for Non-negative.
- class **RandomAcolInitialization**
*This class initializes the W matrix of the **AMF** (p. 499) algorithm by averaging p randomly chosen columns of V .*
- class **RandomInitialization**
*This initialization rule for **AMF** (p. 499) simply fills the W and H matrices with uniform random noise in $[0, 1]$.*
- class **SimpleResidueTermination**
This class implements a simple residue-based termination policy.
- class **SimpleToleranceTermination**
This class implements residue tolerance termination policy.
- class **SVDBatchLearning**
This class implements SVD batch learning with momentum.
- class **SVDCompleteIncrementalLearning**
This class computes SVD using complete incremental batch learning, as described in the following paper:
- class **SVDCompleteIncrementalLearning**< arma::sp_mat >
TODO : Merge this template specialized function for sparse matrix using common row_col_iterator.
- class **SVDIncompleteIncrementalLearning**
This class computes SVD using incomplete incremental batch learning, as described in the following paper:
- class **ValidationRMSETermination**
This class implements validation termination policy based on RMSE index.

Typedefs

- typedef **amf::AMF**< **amf::SimpleResidueTermination**, **amf::RandomAcolInitialization**<>, **amf::NMFA**< **LSUpdate** > **NMFALSFactorizer**
- template<typename MatType = arma::mat>
using **SVDBatchFactorizer** = **amf::AMF**< **amf::SimpleResidueTermination**, **amf::RandomAcol**< **Initialization**<>, **amf::SVDBatchLearning** >

Convenience typedefs.

- `template<class MatType = arma::mat>`
`using SVDCompleteIncrementalFactorizer = amf::AMF< amf::SimpleResidueTermination, amf::RandomAcolInitialization<>, amf::SVDCompleteIncrementalLearning< MatType > >`
SVDCompleteIncrementalFactorizer factorizes given matrix V into two matrices W and H by complete incremental gradient descent.
- `template<class MatType = arma::mat>`
`using SVDIncompleteIncrementalFactorizer = amf::AMF< amf::SimpleResidueTermination, amf::RandomAcolInitialization<>, amf::SVDIncompleteIncrementalLearning >`
SVDIncompleteIncrementalFactorizer factorizes given matrix V into two matrices W and H by incomplete incremental gradient descent.

Functions

- `template<>`
`void SVDBatchLearning::HUpdate< arma::sp_mat > (const arma::sp_mat &V, const arma::mat &W, arma::mat &H)`
- `template<>`
`void SVDBatchLearning::WUpdate< arma::sp_mat > (const arma::sp_mat &V, arma::mat &W, const arma::mat &H)`
TODO : Merge this template specialized function for sparse matrix using common row_col_iterator.
- `template<>`
`void SVDIncompleteIncrementalLearning::HUpdate< arma::sp_mat > (const arma::sp_mat &V, const arma::mat &W, arma::mat &H)`
- `template<>`
`void SVDIncompleteIncrementalLearning::WUpdate< arma::sp_mat > (const arma::sp_mat &V, arma::mat &W, const arma::mat &H)`
TODO : Merge this template specialized function for sparse matrix using common row_col_iterator.

38.6.1 Detailed Description

Alternating Matrix Factorization.

38.6.2 Typedef Documentation

38.6.2.1 NMFALSFactorizer

```
typedef amf::AMF< amf::SimpleResidueTermination, amf::RandomAcolInitialization<>, amf::NMFALSUpdate> NMFALSFactorizer
```

Definition at line 143 of file amf.hpp.

38.6.2.2 SVDBatchFactorizer

```
using SVDBatchFactorizer = amf::AMF< amf::SimpleResidueTermination, amf::RandomAcolInitialization<>,
amf::SVDBatchLearning>
```

Convenience typedefs.

SVDBatchFactorizer factorizes given matrix V into two matrices W and H by gradient descent. SVD batch learning is described in paper 'A Guide to singular Value Decomposition' by Chih-Chao Ma.

See also

SVDBatchLearning (p. 539)

Definition at line 158 of file amf.hpp.

38.6.2.3 SVDCompleteIncrementalFactorizer

```
using SVDCompleteIncrementalFactorizer = amf::AMF< amf::SimpleResidueTermination, amf::↵
RandomAcolInitialization<>, amf::SVDCompleteIncrementalLearning<MatType> >
```

SVDCompleteIncrementalFactorizer factorizes given matrix V into two matrices W and H by complete incremental gradient descent.

SVD complete incremental learning is described in paper 'A Guide to singular Value Decomposition' by Chih-Chao Ma.

See also

SVDCompleteIncrementalLearning (p. 542)

Definition at line 185 of file amf.hpp.

38.6.2.4 SVDIncompleteIncrementalFactorizer

```
using SVDIncompleteIncrementalFactorizer = amf::AMF< amf::SimpleResidueTermination, amf::↵
RandomAcolInitialization<>, amf::SVDIncompleteIncrementalLearning>
```

SVDIncompleteIncrementalFactorizer factorizes given matrix V into two matrices W and H by incomplete incremental gradient descent.

SVD incomplete incremental learning is described in paper 'A Guide to singular Value Decomposition' by Chih-Chao Ma.

See also

SVDIncompleteIncrementalLearning (p. 547)

Definition at line 172 of file amf.hpp.

38.6.3 Function Documentation

38.6.3.1 SVDBatchLearning::HUpdate< arma::sp_mat >()

```
void mlpack::amf::SVDBatchLearning::HUpdate< arma::sp_mat > (
    const arma::sp_mat & V,
    const arma::mat & W,
    arma::mat & H ) [inline]
```

Definition at line 230 of file svd_batch_learning.hpp.

38.6.3.2 SVDBatchLearning::WUpdate< arma::sp_mat >()

```
void mlpack::amf::SVDBatchLearning::WUpdate< arma::sp_mat > (
    const arma::sp_mat & V,
    arma::mat & W,
    const arma::mat & H ) [inline]
```

TODO : Merge this template specialized function for sparse matrix using common row_col_iterator.

WUpdate function specialization for sparse matrix

Definition at line 202 of file svd_batch_learning.hpp.

38.6.3.3 SVDIncompleteIncrementalLearning::HUpdate< arma::sp_mat >()

```
void mlpack::amf::SVDIncompleteIncrementalLearning::HUpdate< arma::sp_mat > (
    const arma::sp_mat & V,
    const arma::mat & W,
    arma::mat & H ) [inline]
```

Definition at line 185 of file svd_incomplete_incremental_learning.hpp.

38.6.3.4 SVDIncompleteIncrementalLearning::WUpdate< arma::sp_mat >()

```
void mlpack::amf::SVDIncompleteIncrementalLearning::WUpdate< arma::sp_mat > (
    const arma::sp_mat & V,
    arma::mat & W,
    const arma::mat & H ) [inline]
```

TODO : Merge this template specialized function for sparse matrix using common row_col_iterator.

template specialized functions for sparse matrices

Definition at line 166 of file svd_incomplete_incremental_learning.hpp.

38.7 mlpack::ann Namespace Reference

Artificial Neural Network.

Namespaces

- **augmented**

Classes

- class **Add**
*Implementation of the **Add** (p. 553) module class.*
- class **AddMerge**
*Implementation of the **AddMerge** (p. 558) module class.*
- class **AddVisitor**
***AddVisitor** (p. 565) exposes the Add() method of the given module.*
- class **AlphaDropout**
The alpha - dropout layer is a regularizer that randomly with probability 'ratio' sets input values to alphaDash.
- class **AtrousConvolution**
*Implementation of the Atrous **Convolution** (p. 643) class.*
- class **BackwardVisitor**
***BackwardVisitor** (p. 587) executes the Backward() function given the input, error and delta parameter.*
- class **BaseLayer**
Implementation of the base layer.
- class **BatchNorm**
Declaration of the Batch Normalization layer class.
- class **BernoulliDistribution**
Multiple independent Bernoulli distributions.
- class **BilinearInterpolation**
Definition and Implementation of the Bilinear Interpolation Layer.
- class **BinaryRBM**
For more information, see the following paper:
- class **BRNN**
Implementation of a standard bidirectional recurrent neural network container.
- class **Concat**
*Implementation of the **Concat** (p. 621) class.*
- class **Concatenate**
*Implementation of the **Concatenate** (p. 630) module class.*
- class **ConcatPerformance**
Implementation of the concat performance class.
- class **Constant**
Implementation of the constant layer.
- class **ConstInitialization**
This class is used to initialize weight matrix with constant values.
- class **Convolution**

Implementation of the **Convolution** (p. 643) class.

- class **CopyVisitor**

This visitor is to support copy constructor for neural network module.

- class **CReLU**

A concatenated ReLU has two outputs, one ReLU and one negative ReLU, concatenated together.

- class **CrossEntropyError**

The cross-entropy performance function measures the network's performance according to the cross-entropy between the input and target distributions.

- class **DeleteVisitor**

DeleteVisitor (p. 659) executes the destructor of the instantiated object.

- class **DeltaVisitor**

DeltaVisitor (p. 660) exposes the delta parameter of the given module.

- class **DeterministicSetVisitor**

DeterministicSetVisitor (p. 661) set the deterministic parameter given the deterministic value.

- class **DiceLoss**

The dice loss performance function measures the network's performance according to the dice coefficient between the input and target distributions.

- class **DropConnect**

The DropConnect (p. 666) layer is a regularizer that randomly with probability ratio sets the connection values to zero and scales the remaining elements by factor $1 / (1 - \text{ratio})$.

- class **Dropout**

The dropout layer is a regularizer that randomly with probability 'ratio' sets input values to zero and scales the remaining elements by factor $1 / (1 - \text{ratio})$ rather than during test time so as to keep the expected sum same.

- class **EarthMoverDistance**

The earth mover distance function measures the network's performance according to the Kantorovich-Rubinstein duality approximation.

- class **ELU**

The ELU (p. 680) activation function, defined by.

- class **FastLSTM**

An implementation of a faster version of the Fast LSTM (p. 801) network layer.

- class **FFN**

Implementation of a standard feed forward network.

- class **FFTConvolution**

Computes the two-dimensional convolution through fft.

- class **FlexibleReLU**

The FlexibleReLU (p. 707) activation function, defined by.

- class **ForwardVisitor**

ForwardVisitor (p. 713) executes the Forward() function given the input and output parameter.

- class **FullConvolution**

- class **GaussianInitialization**

This class is used to initialize weight matrix with a gaussian.

- class **Glimpse**

The glimpse layer returns a retina-like representation (down-scaled cropped images) of increasing scale around a given location in a given image.

- class **GlorotInitializationType**

This class is used to initialize the weight matrix with the Glorot Initialization method.

- class **GradientSetVisitor**

GradientSetVisitor (p. 725) update the gradient parameter given the gradient set.

- class **GradientUpdateVisitor**
GradientUpdateVisitor (p. 727) update the gradient parameter given the gradient set.
- class **GradientVisitor**
SearchModeVisitor executes the *Gradient()* method of the given module using the input and delta parameter.
- class **GradientZeroVisitor**
- class **GRU**
An implementation of a gru network layer.
- class **HardSigmoidFunction**
The hard sigmoid function, defined by.
- class **HardTanH**
The Hard Tanh activation function, defined by.
- class **HeInitialization**
This class is used to initialize weight matrix with the He initialization rule given by He et.
- class **IdentityFunction**
The identity function, defined by.
- class **InitTraits**
This is a template class that can provide information about various initialization methods.
- class **InitTraits< KathirvalavakumarSubavathilInitialization >**
Initialization traits of the kathirvalavakumar subavath initialization rule.
- class **InitTraits< NguyenWidrowInitialization >**
Initialization traits of the Nguyen-Widrow initialization rule.
- class **Join**
*Implementation of the **Join** (p. 753) module class.*
- class **KathirvalavakumarSubavathilInitialization**
This class is used to initialize the weight matrix with the method proposed by T.
- class **KLDivergence**
The Kullback–Leibler divergence is often used for continuous distributions (direct regression).
- class **LayerNorm**
Declaration of the Layer Normalization class.
- class **LayerTraits**
This is a template class that can provide information about various layers.
- class **LeakyReLU**
*The **LeakyReLU** (p. 771) activation function, defined by.*
- class **LecunNormalInitialization**
This class is used to initialize weight matrix with the Lecun Normalization initialization rule.
- class **Linear**
*Implementation of the **Linear** (p. 776) layer class.*
- class **LinearNoBias**
*Implementation of the **LinearNoBias** (p. 782) class.*
- class **LoadOutputParameterVisitor**
LoadOutputParameterVisitor (p. 788) restores the output parameter using the given parameter set.
- class **LogisticFunction**
The logistic function, defined by.
- class **LogSoftMax**
Implementation of the log softmax layer.
- class **Lookup**
*Implementation of the **Lookup** (p. 795) class.*

- class **LossVisitor**
***LossVisitor** (p. 800) exposes the `Loss()` method of the given module.*
- class **LSTM**
*Implementation of the **LSTM** (p. 801) module class.*
- class **MaxPooling**
*Implementation of the **MaxPooling** (p. 808) layer.*
- class **MaxPoolingRule**
- class **MeanPooling**
*Implementation of the **MeanPooling** (p. 814).*
- class **MeanPoolingRule**
- class **MeanSquaredError**
The mean squared error performance function measures the network's performance according to the mean of squared errors.
- class **MultiplyConstant**
Implementation of the multiply constant layer.
- class **MultiplyMerge**
*Implementation of the **MultiplyMerge** (p. 829) module class.*
- class **NaiveConvolution**
Computes the two-dimensional convolution.
- class **NegativeLogLikelihood**
Implementation of the negative log likelihood layer.
- class **NetworkInitialization**
This class is used to initialize the network with the given initialization rule.
- class **NguyenWidrowInitialization**
This class is used to initialize the weight matrix with the Nguyen-Widrow method.
- class **OivsInitialization**
This class is used to initialize the weight matrix with the oivs method.
- class **OrthogonalInitialization**
This class is used to initialize the weight matrix with the orthogonal matrix initialization.
- class **OutputHeightVisitor**
***OutputHeightVisitor** (p. 850) exposes the `OutputHeight()` method of the given module.*
- class **OutputParameterVisitor**
***OutputParameterVisitor** (p. 851) exposes the output parameter of the given module.*
- class **OutputWidthVisitor**
***OutputWidthVisitor** (p. 852) exposes the `OutputWidth()` method of the given module.*
- class **ParametersSetVisitor**
***ParametersSetVisitor** (p. 853) update the parameters set using the given matrix.*
- class **ParametersVisitor**
***ParametersVisitor** (p. 854) exposes the parameters set of the given module and stores the parameters set into the given matrix.*
- class **PReLU**
*The **PReLU** (p. 855) activation function, defined by (where α is trainable)*
- class **RandomInitialization**
This class is used to initialize randomly the weight matrix.
- class **RBM**
*The implementation of the **RBM** (p. 864) module.*
- class **ReconstructionLoss**

The reconstruction loss performance function measures the network's performance equal to the negative log probability of the target with the input distribution.

- class **RectifierFunction**

The rectifier function, defined by.

- class **Recurrent**

Implementation of the *RecurrentLayer* class.

- class **RecurrentAttention**

This class implements the **Recurrent** (p. 886) Model for Visual Attention, using a variety of possible layer implementations.

- class **ReinforceNormal**

Implementation of the reinforce normal layer.

- class **Reparametrization**

Implementation of the **Reparametrization** (p. 903) layer class.

- class **ResetCellVisitor**

ResetCellVisitor (p. 907) executes the *ResetCell()* function.

- class **ResetVisitor**

ResetVisitor (p. 909) executes the *Reset()* function.

- class **RewardSetVisitor**

RewardSetVisitor (p. 910) set the reward parameter given the reward value.

- class **RNN**

Implementation of a standard recurrent neural network container.

- class **RunSetVisitor**

RunSetVisitor (p. 921) set the run parameter given the run value.

- class **SaveOutputParameterVisitor**

SaveOutputParameterVisitor (p. 923) saves the output parameter into the given parameter set.

- class **Select**

The select module selects the specified column from a given input matrix.

- class **Sequential**

Implementation of the **Sequential** (p. 927) class.

- class **SetInputHeightVisitor**

SetInputHeightVisitor (p. 934) updates the input height parameter with the given input height.

- class **SetInputWidthVisitor**

SetInputWidthVisitor (p. 935) updates the input width parameter with the given input width.

- class **SigmoidCrossEntropyError**

The **SigmoidCrossEntropyError** (p. 937) performance function measures the network's performance according to the cross-entropy function between the input and target distributions.

- class **SoftplusFunction**

The softplus function, defined by.

- class **SoftsignFunction**

The softsign function, defined by.

- class **SpikeSlabRBM**

For more information, see the following paper:

- class **Subview**

Implementation of the subview layer.

- class **SVDConvolution**

Computes the two-dimensional convolution using singular value decomposition.

- class **SwishFunction**

The swish function, defined by.

- class **TanhFunction**
The tanh function, defined by.
- class **TransposedConvolution**
*Implementation of the Transposed **Convolution** (p. 643) class.*
- class **ValidConvolution**
- class **VRClassReward**
Implementation of the variance reduced classification reinforcement layer.
- class **WeightSetVisitor**
***WeightSetVisitor** (p. 972) update the module parameters given the parameters set.*
- class **WeightSizeVisitor**
***WeightSizeVisitor** (p. 973) returns the number of weights of the given module.*

Typedefs

- template<class ActivationFunction = LogisticFunction, typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
using **CustomLayer** = **BaseLayer**< ActivationFunction, InputDataType, OutputDataType >
Standard Sigmoid layer.
- template<typename MatType = arma::mat>
using **Embedding** = **Lookup**< MatType, MatType >
- using **GlorotInitialization** = **GlorotInitializationType**< false >
GlorotInitialization uses uniform distribution.
- template<class ActivationFunction = HardSigmoidFunction, typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
using **HardSigmoidLayer** = **BaseLayer**< ActivationFunction, InputDataType, OutputDataType >
Standard HardSigmoid-Layer using the HardSigmoid activation function.
- template<class ActivationFunction = IdentityFunction, typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
using **IdentityLayer** = **BaseLayer**< ActivationFunction, InputDataType, OutputDataType >
Standard Identity-Layer using the identity activation function.
- template<typename... CustomLayers>
using **LayerTypes** = boost::variant< **Add**< arma::mat, arma::mat > *, **AddMerge**< arma::mat, arma::mat > *, **AtrousConvolution**< **NaiveConvolution**< **ValidConvolution** >, **NaiveConvolution**< **FullConvolution** >, **NaiveConvolution**< **ValidConvolution** >, arma::mat, arma::mat > *, **BaseLayer**< **LogisticFunction**, arma::mat, arma::mat > *, **BaseLayer**< **IdentityFunction**, arma::mat, arma::mat > *, **BaseLayer**< **TanhFunction**, arma::mat, arma::mat > *, **BaseLayer**< **RectifierFunction**, arma::mat, arma::mat > *, **BaseLayer**< **SoftplusFunction**, arma::mat, arma::mat > *, **BatchNorm**< arma::mat, arma::mat > *, **BilinearInterpolation**< arma::mat, arma::mat > *, **Concat**< arma::mat, arma::mat > *, **Concatenate**< arma::mat, arma::mat > *, **ConcatPerformance**< **NegativeLogLikelihood**< arma::mat, arma::mat >, arma::mat, arma::mat > *, **Constant**< arma::mat, arma::mat > *, **Convolution**< **NaiveConvolution**< **ValidConvolution** >, **NaiveConvolution**< **FullConvolution** >, **NaiveConvolution**< **ValidConvolution** >, arma::mat, arma::mat > *, **TransposedConvolution**< **NaiveConvolution**< **ValidConvolution** >, **NaiveConvolution**< **FullConvolution** >, **NaiveConvolution**< **ValidConvolution** >, arma::mat, arma::mat > *, **DropConnect**< arma::mat, arma::mat > *, **Dropout**< arma::mat, arma::mat > *, **AlphaDropout**< arma::mat, arma::mat > *, **ELU**< arma::mat, arma::mat > *, **FlexibleReLU**< arma::mat, arma::mat > *, **Glimpse**< arma::mat, arma::mat > *, **HardTanH**< arma::mat, arma::mat > *, **Join**< arma::mat, arma::mat > *, **LayerNorm**< arma::mat, arma::mat > *, **LeakyReLU**< arma::mat, arma::mat > *, **CReLU**< arma::mat, arma::mat > *, **Linear**< arma::mat, arma::mat > *, **LinearNoBias**< arma::mat, arma::mat > *, **LogSoftMax**< arma::mat, arma::mat > *, **Lookup**< arma::mat, arma::mat > *, **LSTM**< arma::mat, arma::mat > *, **GRU**< arma::mat, arma::mat > *, **FastLSTM**< arma::mat, arma::mat > *, **MaxPooling**< arma::mat, arma::mat > *, **MeanPooling**< arma::mat, arma::mat > *, **MultiplyConstant**< arma::mat, arma::mat > *, **MultiplyMerge**< arma::mat, arma::mat > *, **NegativeLogLikelihood**< arma::mat, arma::mat > *, **PReLU**< arma::mat, arma::mat > *, **Recurrent**< arma::mat, arma::mat > *, **RecurrentAttention**< arma::mat, arma::mat > *, **ReinforceNormal**< arma::mat,

```
arma::mat > *, Reparametrization< arma::mat, arma::mat > *, Select< arma::mat, arma::mat > *, Sequential< arma::mat, arma::mat, false > *, Sequential< arma::mat, arma::mat, true > *, Subview< arma::mat, arma::mat > *, VRClassReward< arma::mat, arma::mat > *, CustomLayers *... >
```

- `template<class ActivationFunction = RectifierFunction, typename InputDataType = arma::mat, typename OutputDataType = arma::mat> using ReLULayer = BaseLayer< ActivationFunction, InputDataType, OutputDataType >`

Standard rectified linear unit non-linearity layer.

- `template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat, typename... CustomLayers> using Residual = Sequential< InputDataType, OutputDataType, true, CustomLayers... >`
- `using SELU = ELU< arma::mat, arma::mat >`
- `template<class ActivationFunction = LogisticFunction, typename InputDataType = arma::mat, typename OutputDataType = arma::mat> using SigmoidLayer = BaseLayer< ActivationFunction, InputDataType, OutputDataType >`

Standard Sigmoid-Layer using the logistic activation function.

- `template<class ActivationFunction = SoftplusFunction, typename InputDataType = arma::mat, typename OutputDataType = arma::mat> using SoftPlusLayer = BaseLayer< ActivationFunction, InputDataType, OutputDataType >`

Standard Softplus-Layer using the Softplus activation function.

- `template<class ActivationFunction = TanhFunction, typename InputDataType = arma::mat, typename OutputDataType = arma::mat> using TanHLayer = BaseLayer< ActivationFunction, InputDataType, OutputDataType >`

Standard hyperbolic tangent layer.

- `using XavierInitialization = GlorotInitializationType< true >`

XavierInitialization is the popular name for this method.

Functions

- **HAS_ANY_METHOD_FORM** (Model, HasModelCheck)
- **HAS_MEM_FUNC** (Gradient, HasGradientCheck)
- **HAS_MEM_FUNC** (Deterministic, HasDeterministicCheck)
- **HAS_MEM_FUNC** (Parameters, HasParametersCheck)
- **HAS_MEM_FUNC** (**Add**, HasAddCheck)
- **HAS_MEM_FUNC** (Location, HasLocationCheck)
- **HAS_MEM_FUNC** (Reset, HasResetCheck)
- **HAS_MEM_FUNC** (ResetCell, HasResetCellCheck)
- **HAS_MEM_FUNC** (Reward, HasRewardCheck)
- **HAS_MEM_FUNC** (InputWidth, HasInputWidth)
- **HAS_MEM_FUNC** (InputHeight, HasInputHeight)
- **HAS_MEM_FUNC** (Rho, HasRho)
- **HAS_MEM_FUNC** (Loss, HasLoss)
- **HAS_MEM_FUNC** (Run, HasRunCheck)

38.7.1 Detailed Description

Artificial Neural Network.

38.7.2 Typedef Documentation

38.7.2.1 CustomLayer

```
using CustomLayer = BaseLayer< ActivationFunction, InputDataType, OutputDataType>
```

Standard Sigmoid layer.

Definition at line 31 of file custom_layer.hpp.

38.7.2.2 Embedding

```
using Embedding = Lookup<MatType, MatType>
```

Definition at line 131 of file lookup.hpp.

38.7.2.3 GlorotInitialization

```
using GlorotInitialization = GlorotInitializationType<false>
```

GlorotInitialization uses uniform distribution.

Definition at line 148 of file glorot_init.hpp.

38.7.2.4 HardSigmoidLayer

```
using HardSigmoidLayer = BaseLayer< ActivationFunction, InputDataType, OutputDataType>
```

Standard HardSigmoid-Layer using the HardSigmoid activation function.

Definition at line 185 of file base_layer.hpp.

38.7.2.5 IdentityLayer

```
using IdentityLayer = BaseLayer< ActivationFunction, InputDataType, OutputDataType>
```

Standard Identity-Layer using the identity activation function.

Definition at line 141 of file base_layer.hpp.

38.7.2.6 LayerTypes

```
using LayerTypes = boost::variant< Add<arma::mat, arma::mat>*, AddMerge<arma::mat, arma::mat>*,
AtrousConvolution< NaiveConvolution< ValidConvolution>, NaiveConvolution< FullConvolution>,
NaiveConvolution< ValidConvolution>, arma::mat, arma::mat>*, BaseLayer< LogisticFunction,
arma::mat, arma::mat>*, BaseLayer< IdentityFunction, arma::mat, arma::mat>*, BaseLayer< TanhFunction,
arma::mat, arma::mat>*, BaseLayer< RectifierFunction, arma::mat, arma::mat>*, BaseLayer< SoftplusFunction,
arma::mat, arma::mat>*, BatchNorm<arma::mat, arma::mat>*, BilinearInterpolation<arma::mat, arma::mat>*,
Concat<arma::mat, arma::mat>*, Concatenate<arma::mat, arma::mat>*, ConcatPerformance< NegativeLogLikelihood<arma::mat, arma::mat>, arma::mat,
arma::mat>*, Constant<arma::mat, arma::mat>*, Convolution< NaiveConvolution< ValidConvolution>, NaiveConvolution< FullConvolution>,
NaiveConvolution< ValidConvolution>, arma::mat, arma::mat>*, TransposedConvolution< NaiveConvolution< ValidConvolution>, NaiveConvolution< FullConvolution>,
NaiveConvolution< ValidConvolution>, arma::mat, arma::mat>*, DropConnect<arma::mat, arma::mat>*, Dropout<arma::mat, arma::mat>*, AlphaDropout<arma::mat, arma::mat>*, ELU<arma::mat, arma::mat>*,
FlexibleReLU<arma::mat, arma::mat>*, Glimpse<arma::mat, arma::mat>*, HardTanH<arma::mat, arma::mat>*, Join<arma::mat, arma::mat>*, LayerNorm<arma::mat, arma::mat>*, LeakyReLU<arma::mat, arma::mat>*, CReLU<arma::mat, arma::mat>*, Linear<arma::mat, arma::mat>*,
LinearNoBias<arma::mat, arma::mat>*, LogSoftMax<arma::mat, arma::mat>*, Lookup<arma::mat, arma::mat>*, LSTM<arma::mat, arma::mat>*, GRU<arma::mat, arma::mat>*, FastLSTM<arma::mat, arma::mat>*, MaxPooling<arma::mat, arma::mat>*, MeanPooling<arma::mat, arma::mat>*,
MultiplyConstant<arma::mat, arma::mat>*, MultiplyMerge<arma::mat, arma::mat>*, NegativeLogLikelihood<arma::mat, arma::mat>*, PReLU<arma::mat, arma::mat>*, Recurrent<arma::mat, arma::mat>*, RecurrentAttention<arma::mat, arma::mat>*, ReinforceNormal<arma::mat, arma::mat>*, Reparametrization<arma::mat, arma::mat>*, Select<arma::mat, arma::mat>*, Sequential<arma::mat, arma::mat, false>*, Sequential<arma::mat, arma::mat, true>*, Subview<arma::mat, arma::mat>*, VRClassReward<arma::mat, arma::mat>*, CustomLayers*... >
```

Definition at line 203 of file layer_types.hpp.

38.7.2.7 ReLULayer

```
using ReLULayer = BaseLayer< ActivationFunction, InputDataType, OutputDataType>
```

Standard rectified linear unit non-linearity layer.

Definition at line 152 of file base_layer.hpp.

38.7.2.8 Residual

```
using Residual = Sequential< InputDataType, OutputDataType, true, CustomLayers*...>
```

Definition at line 241 of file sequential.hpp.

38.7.2.9 SELU

```
using SELU = ELU<arma::mat, arma::mat>
```

Definition at line 263 of file elu.hpp.

38.7.2.10 SigmoidLayer

```
using SigmoidLayer = BaseLayer< ActivationFunction, InputDataType, OutputDataType>
```

Standard Sigmoid-Layer using the logistic activation function.

Definition at line 130 of file base_layer.hpp.

38.7.2.11 SoftPlusLayer

```
using SoftPlusLayer = BaseLayer< ActivationFunction, InputDataType, OutputDataType>
```

Standard Softplus-Layer using the Softplus activation function.

Definition at line 174 of file base_layer.hpp.

38.7.2.12 TanHLayer

```
using TanHLayer = BaseLayer< ActivationFunction, InputDataType, OutputDataType>
```

Standard hyperbolic tangent layer.

Definition at line 163 of file base_layer.hpp.

38.7.2.13 XavierInitialization

```
using XavierInitialization = GlorotInitializationType<true>
```

XavierInitialization is the popular name for this method.

Definition at line 143 of file glorot_init.hpp.

38.7.3 Function Documentation

38.7.3.1 HAS_ANY_METHOD_FORM()

```
mlpack::ann::HAS_ANY_METHOD_FORM (
    Model ,
    HasModelCheck )
```

38.7.3.2 HAS_MEM_FUNC() [1/13]

```
mlpack::ann::HAS_MEM_FUNC (
    Gradient ,
    HasGradientCheck )
```

38.7.3.3 HAS_MEM_FUNC() [2/13]

```
mlpack::ann::HAS_MEM_FUNC (
    Deterministic ,
    HasDeterministicCheck )
```

38.7.3.4 HAS_MEM_FUNC() [3/13]

```
mlpack::ann::HAS_MEM_FUNC (
    Parameters ,
    HasParametersCheck )
```

38.7.3.5 HAS_MEM_FUNC() [4/13]

```
mlpack::ann::HAS_MEM_FUNC (
    Add ,
    HasAddCheck )
```


38.7.3.6 HAS_MEM_FUNC() [5/13]

```
mlpack::ann::HAS_MEM_FUNC (
    Location ,
    HasLocationCheck )
```

38.7.3.7 HAS_MEM_FUNC() [6/13]

```
mlpack::ann::HAS_MEM_FUNC (
    Reset ,
    HasResetCheck )
```

38.7.3.8 HAS_MEM_FUNC() [7/13]

```
mlpack::ann::HAS_MEM_FUNC (
    ResetCell ,
    HasResetCellCheck )
```

38.7.3.9 HAS_MEM_FUNC() [8/13]

```
mlpack::ann::HAS_MEM_FUNC (
    Reward ,
    HasRewardCheck )
```

38.7.3.10 HAS_MEM_FUNC() [9/13]

```
mlpack::ann::HAS_MEM_FUNC (
    InputWidth ,
    HasInputWidth )
```

38.7.3.11 HAS_MEM_FUNC() [10/13]

```
mlpack::ann::HAS_MEM_FUNC (
    InputHeight ,
    HasInputHeight )
```

38.7.3.12 HAS_MEM_FUNC() [11/13]

```
mlpack::ann::HAS_MEM_FUNC (
    Rho ,
    HasRho )
```

38.7.3.13 HAS_MEM_FUNC() [12/13]

```
mlpack::ann::HAS_MEM_FUNC (
    Loss ,
    HasLoss )
```

38.7.3.14 HAS_MEM_FUNC() [13/13]

```
mlpack::ann::HAS_MEM_FUNC (
    Run ,
    HasRunCheck )
```

38.8 mlpack::ann::augmented Namespace Reference**Namespaces**

- **scorers**
- **tasks**

38.9 mlpack::ann::augmented::scorers Namespace Reference**Functions**

- `template<typename MatType >`
`double SequencePrecision (arma::field< MatType > trueOutputs, arma::field< MatType > predOutputs, double tol=1e-4)`
Function that computes the sequences precision (number of correct sequences / number of sequences) of model's answer against ground truth answer.

38.9.1 Function Documentation

38.9.1.1 SequencePrecision()

```
double mlpack::ann::augmented::scorers::SequencePrecision (
    arma::field< MatType > trueOutputs,
    arma::field< MatType > predOutputs,
    double tol = 1e-4 )
```

Function that computes the sequences precision (number of correct sequences / number of sequences) of model's answer against ground truth answer.

Parameters

<i>trueOutputs</i>	Ground truth sequences.
<i>predOutputs</i>	Sequences predicted by model.
<i>tol</i>	Minimum absolute difference value which is considered as a model failure.

38.10 mlpack::ann::augmented::tasks Namespace Reference

Classes

- class **AddTask**
Generator of instances of the binary addition task.
- class **CopyTask**
Generator of instances of the binary sequence copy task.
- class **SortTask**
Generator of instances of the sequence sort task.

38.11 mlpack::bindings Namespace Reference

Namespaces

- **cli**
- **markdown**
- **python**
- **tests**

38.12 mlpack::bindings::cli Namespace Reference

Classes

- class **CLIOption**
*A static object whose constructor registers a parameter with the **CLI** (p. 1117) class.*
- struct **ParameterType**
Utility struct to return the type that boost::program_options should accept for a given input type.
- struct **ParameterType< arma::Col< eT > >**
For vector types, boost::program_options will accept a std::string, not an arma::Col<eT> (since it is not clear how to specify a vector on the command-line).
- struct **ParameterType< arma::Mat< eT > >**
For matrix types, boost::program_options will accept a std::string, not an arma::mat (since it is not clear how to specify a matrix on the command-line).
- struct **ParameterType< arma::Row< eT > >**

For row vector types, `boost::program_options` will accept a `std::string`, not an `arma::Row<eT>` (since it is not clear how to specify a vector on the command-line).

- struct **ParameterType**< `std::tuple< mlpack::data::DatasetMapper< PolicyType, std::string >, arma::Mat< eT > >` >

For matrix+dataset info types, we should accept a `std::string`.

- struct **ParameterTypeDeducer**
- struct **ParameterTypeDeducer**< `true`, `T` >
- class **ProgramDoc**

A static object whose constructor registers program documentation with the **CLI** (p. 1117) class.

Functions

- template<typename T >
void **AddToPO** (const `std::string` &boostName, const `std::string` &descr, `boost::program_options::options_↵` description &desc, const typename `boost::disable_if< util::IsStdVector< T >>::type` !=0, const typename `boost::disable_if< std::is_same< T, bool >>::type` !=0)

Add a non-vector option to `boost::program_options`.

- template<typename T >
void **AddToPO** (const `std::string` &boostName, const `std::string` &descr, `boost::program_options::options_↵` description &desc, const typename `boost::enable_if< util::IsStdVector< T >>::type` !=0, const typename `boost::disable_if< std::is_same< T, bool >>::type` !=0)

Add a vector option to `boost::program_options`.

- template<typename T >
void **AddToPO** (const `std::string` &boostName, const `std::string` &descr, `boost::program_options::options_↵` description &desc, const typename `boost::disable_if< util::IsStdVector< T >>::type` !=0, const typename `boost::enable_if< std::is_same< T, bool >>::type` !=0)

Add a boolean option to `boost::program_options`.

- template<typename T >
void **AddToPO** (const `util::ParamData` &d, const void *, void *output)

Add an option to `boost::program_options`.

- template<typename T >
void **DefaultParam** (const `util::ParamData` &data, const void *, void *output)

Return the default value of an option.

- template<typename T >
`std::string` **DefaultParamImpl** (const `util::ParamData` &data, const typename `boost::disable_if< arma::is_↵` `arma_type< T >>::type` !=0, const typename `boost::disable_if< util::IsStdVector< T >>::type` !=0, const typename `boost::disable_if< data::HasSerialize< T >>::type` !=0, const typename `boost::disable_if< std::is_↵` `_same< T, std::string >>::type` !=0, const typename `boost::disable_if< std::is_same< T, std::tuple< mlpack_↵` `::data::DatasetInfo, arma::mat >>>::type` !=0)

Return the default value of an option.

- template<typename T >
`std::string` **DefaultParamImpl** (const `util::ParamData` &data, const typename `boost::enable_if< util::IsStd_↵` `Vector< T >>::type` !=0)

Return the default value of a vector option.

- template<typename T >
`std::string` **DefaultParamImpl** (const `util::ParamData` &data, const typename `boost::enable_if< std::is_same< ↵` `T, std::string >>::type` !=0)

Return the default value of a string option.

- `template<typename T >`
`std::string DefaultParamImpl (const util::ParamData &data, const typename boost::enable_if_c< arma::is_↵`
`_arma_type< T >::value||std::is_same< T, std::tuple< mlpack::data::DatasetInfo, arma::mat >>::value >↵`
`::type ==0)`
Return the default value of a matrix option, a tuple option, a serializable option, or a string option (this returns the default filename, or " if the default is no file).
- `template<typename T >`
`std::string DefaultParamImpl (const util::ParamData &data, const typename boost::disable_if< arma::is_↵`
`arma_type< T >>::type ==0, const typename boost::enable_if< data::HasSerialize< T >>::type ==0)`
Return the default value of a model option (this returns the default filename, or " if the default is no file).
- `template<typename T >`
`void DeleteAllocatedMemory (const util::ParamData &d, const void *, void *)`
- `template<typename T >`
`void DeleteAllocatedMemoryImpl (const util::ParamData &, const typename boost::disable_if< data::Has_↵`
`Serialize< T >>::type ==0, const typename boost::disable_if< arma::is_arma_type< T >>::type ==0)`
- `template<typename T >`
`void DeleteAllocatedMemoryImpl (const util::ParamData &, const typename boost::enable_if< arma::is_↵`
`arma_type< T >>::type ==0)`
- `template<typename T >`
`void DeleteAllocatedMemoryImpl (const util::ParamData &d, const typename boost::disable_if< arma::is_↵`
`arma_type< T >>::type ==0, const typename boost::enable_if< data::HasSerialize< T >>::type ==0)`
- `void EndProgram ()`
Handle command-line program termination.
- `template<typename T >`
`void * GetAllocatedMemory (const util::ParamData &, const typename boost::disable_if< data::Has_↵`
`Serialize< T >>::type ==0, const typename boost::disable_if< arma::is_arma_type< T >>::type ==0)`
- `template<typename T >`
`void * GetAllocatedMemory (const util::ParamData &, const typename boost::enable_if< arma::is_arma_↵`
`type< T >>::type ==0)`
- `template<typename T >`
`void * GetAllocatedMemory (const util::ParamData &d, const typename boost::disable_if< arma::is_arma_↵`
`_type< T >>::type ==0, const typename boost::enable_if< data::HasSerialize< T >>::type ==0)`
- `template<typename T >`
`void GetAllocatedMemory (const util::ParamData &d, const void *, void *output)`
- `std::string GetBindingName (const std::string &bindingName)`
Given the name of a binding, print its command-line name (this returns "mlpack_<bindingName>".
- `template<typename T >`
`T & GetParam (util::ParamData &d, const typename boost::disable_if< arma::is_arma_type< T >>::type ==0,`
`const typename boost::disable_if< data::HasSerialize< T >>::type ==0, const typename boost::disable_if<`
`std::is_same< T, std::tuple< mlpack::data::DatasetInfo, arma::mat >>>::type ==0)`
This overload is called when nothing special needs to happen to the name of the parameter.
- `template<typename T >`
`T & GetParam (util::ParamData &d, const typename boost::enable_if< arma::is_arma_type< T >>::type ==0)`
Return a matrix parameter.
- `template<typename T >`
`T & GetParam (util::ParamData &d, const typename boost::enable_if< std::is_same< T, std::tuple< mlpack_↵`
`::data::DatasetInfo, arma::mat >>>::type ==0)`
Return a matrix/dataset info parameter.
- `template<typename T >`
`T *& GetParam (util::ParamData &d, const typename boost::disable_if< arma::is_arma_type< T >>::type`
`==0, const typename boost::enable_if< data::HasSerialize< T >>::type ==0)`

Return a serializable object.

- template<typename T >
void **GetParam** (const **util::ParamData** &d, const void *, void *output)

Return a parameter casted to the given type.

- template<typename T >
std::string **GetPrintableParam** (const **util::ParamData** &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< **util::IsStdVector**< T >>::type *=0, const typename boost::disable_if< **data::HasSerialize**< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< **data::DatasetInfo**, arma::mat >>>::type *=0)

Print an option.

- template<typename T >
std::string **GetPrintableParam** (const **util::ParamData** &data, const typename std::enable_if< **util::IsStdVector**< T >>::value >::type *=0)

Print a vector option, with spaces between it.

- template<typename T >
std::string **GetPrintableParam** (const **util::ParamData** &data, const typename std::enable_if< arma::is_arma_type< T >>::value||std::is_same< T, std::tuple< **data::DatasetInfo**, arma::mat >>>::value >::type *=0)

Print a matrix/tuple option (this just prints the filename).

- template<typename T >
std::string **GetPrintableParam** (const **util::ParamData** &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< **data::HasSerialize**< T >>::type *=0)

Print a model option (this just prints the filename).

- template<typename T >
void **GetPrintableParam** (const **util::ParamData** &data, const void *, void *output)

Print an option into a std::string.

- template<typename T >
std::string **GetPrintableParamName** (const **util::ParamData** &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< **data::HasSerialize**< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< **data::DatasetInfo**, arma::mat >>>::type *=0)

Get the parameter name for a type that has no special handling.

- template<typename T >
std::string **GetPrintableParamName** (const **util::ParamData** &data, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)

Get the parameter name for a matrix type (where the user has to pass the file that holds the matrix).

- template<typename T >
std::string **GetPrintableParamName** (const **util::ParamData** &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< **data::HasSerialize**< T >>::type *=0)

Get the parameter name for a serializable model type (where the user has to pass the file that holds the matrix).

- template<typename T >
std::string **GetPrintableParamName** (const **util::ParamData** &data, const typename boost::enable_if< std::is_same< T, std::tuple< **data::DatasetInfo**, arma::mat >>>::type *=0)

Get the parameter name for a mapped matrix type (where the user has to pass the file that holds the matrix).

- template<typename T >
void **GetPrintableParamName** (const **util::ParamData** &d, const void *, void *output)

Get the parameter's name as seen by the user.

- template<typename T >
std::string **GetPrintableParamValue** (const **util::ParamData** &data, const std::string &value, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< **data::HasSerialize**< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< **data::DatasetInfo**, arma::mat >>>::type *=0)

Get the parameter name for a type that has no special handling.

- template<typename T >
 std::string **GetPrintableParamValue** (const **util::ParamData** &data, const std::string &value, const typename boost::enable_if< arma::is_arma_type< T >>::type !=0)
Get the parameter name for a matrix type (where the user has to pass the file that holds the matrix).
- template<typename T >
 std::string **GetPrintableParamValue** (const **util::ParamData** &data, const std::string &value, const typename boost::disable_if< arma::is_arma_type< T >>::type !=0, const typename boost::enable_if< **data::HasSerialize**< T >>::type !=0)
Get the parameter name for a serializable model type (where the user has to pass the file that holds the matrix).
- template<typename T >
 std::string **GetPrintableParamValue** (const **util::ParamData** &data, const std::string &value, const typename boost::enable_if< std::is_same< T, std::tuple< **data::DatasetInfo**, arma::mat >>>::type !=0)
Get the parameter name for a mapped matrix type (where the user has to pass the file that holds the matrix).
- template<typename T >
 void **GetPrintableParamValue** (const **util::ParamData** &d, const void *input, void *output)
Get the parameter's name as seen by the user.
- template<typename T >
 std::string **GetPrintableType** (const **util::ParamData** &data, const typename boost::disable_if< arma::is_arma_type< T >>::type !=0, const typename boost::disable_if< **util::IsStdVector**< T >>::type !=0, const typename boost::disable_if< **data::HasSerialize**< T >>::type !=0, const typename boost::disable_if< std::is_same< T, std::tuple< **data::DatasetInfo**, arma::mat >>>::type !=0)
Return a string representing the command-line type of an option.
- template<typename T >
 std::string **GetPrintableType** (const **util::ParamData** &data, const typename std::enable_if< **util::IsStdVector**< T >::value >::type !=0)
Return a string representing the command-line type of a vector.
- template<typename T >
 std::string **GetPrintableType** (const **util::ParamData** &data, const typename std::enable_if< arma::is_arma_type< T >::value >::type !=0)
Return a string representing the command-line type of a matrix option.
- template<typename T >
 std::string **GetPrintableType** (const **util::ParamData** &data, const typename std::enable_if< std::is_same< T, std::tuple< **data::DatasetInfo**, arma::mat >>>::value >::type !=0)
Return a string representing the command-line type of a matrix tuple option.
- template<typename T >
 std::string **GetPrintableType** (const **util::ParamData** &data, const typename boost::disable_if< arma::is_arma_type< T >>::type !=0, const typename boost::enable_if< **data::HasSerialize**< T >>::type !=0)
Return a string representing the command-line type of a model.
- template<typename T >
 void **GetPrintableType** (const **util::ParamData** &data, const void *, void *output)
Print the command-line type of an option into a string.
- template<typename T >
 T & **GetRawParam** (**util::ParamData** &d, const typename boost::disable_if< arma::is_arma_type< T >>::type !=0, const typename boost::disable_if< **data::HasSerialize**< T >>::type !=0, const typename boost::disable_if< std::is_same< T, std::tuple< **mlpack::data::DatasetInfo**, arma::mat >>>::type !=0)
This overload is called when nothing special needs to happen to the name of the parameter.
- template<typename T >
 T & **GetRawParam** (**util::ParamData** &d, const typename boost::enable_if_c< arma::is_arma_type< T >::value || std::is_same< T, std::tuple< **mlpack::data::DatasetInfo**, arma::mat >>>::value >::type !=0)
Return a matrix parameter.

- `template<typename T >`
`T * & GetRawParam (util::ParamData &d, const typename boost::disable_if< arma::is_arma_type< T >>::type ==0, const typename boost::enable_if< data::HasSerialize< T >>::type ==0)`
Return the name of a model parameter.
- `template<typename T >`
`void GetRawParam (const util::ParamData &d, const void *, void *output)`
Return a parameter casted to the given type.
- `template<typename T >`
`bool IgnoreCheck (const T &)`
Return whether or not a runtime check on parameters should be ignored.
- `template<typename T >`
`std::string MapParameterName (const std::string &identifier, const typename boost::disable_if< arma::is_arma_type< T >>::type ==0, const typename boost::disable_if< data::HasSerialize< T >>::type ==0, const typename boost::disable_if< std::is_same< T, std::tuple< mlpack::data::DatasetInfo, arma::mat >>>::type ==0)`
If needed, map the parameter name to the name that is used by boost::program_options.
- `template<typename T >`
`std::string MapParameterName (const std::string &identifier, const typename boost::enable_if_c< arma::is_arma_type< T >::value || std::is_same< T, std::tuple< mlpack::data::DatasetInfo, arma::mat >>::value || data::HasSerialize< T >::value >::type ==0)`
Map the parameter name to the name that is used by boost::program_options.
- `template<typename T >`
`void MapParameterName (const util::ParamData &d, const void *, void *output)`
Map the parameter name to the name seen by boost::program_options.
- `template<typename T >`
`void OutputParam (const util::ParamData &data, const void *, void *)`
Output an option.
- `template<typename T >`
`void OutputParamImpl (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_type< T >>::type ==0, const typename boost::disable_if< util::IsStdVector< T >>::type ==0, const typename boost::disable_if< data::HasSerialize< T >>::type ==0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type ==0)`
Output an option (print to stdout).
- `template<typename T >`
`void OutputParamImpl (const util::ParamData &data, const typename boost::enable_if< util::IsStdVector< T >>::type ==0)`
Output a vector option (print to stdout).
- `template<typename T >`
`void OutputParamImpl (const util::ParamData &data, const typename boost::enable_if< arma::is_arma_type< T >>::type ==0)`
Output a matrix option (this saves it to the given file).
- `template<typename T >`
`void OutputParamImpl (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_type< T >>::type ==0, const typename boost::enable_if< data::HasSerialize< T >>::type ==0)`
Output a serializable class option (this saves it to the given file).
- `template<typename T >`
`void OutputParamImpl (const util::ParamData &data, const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type ==0)`
Output a mapped dataset.
- **PARAM_FLAG** ("help", "Default help info.", "h")

- **PARAM_FLAG** ("verbose", "Display informational messages and the full list of " "parameters and timers at the end of execution.", "v")
- **PARAM_FLAG** ("version", "Display the version of mlpack.", "V")
- **PARAM_STRING_IN** ("info", "Print help on a specific option.", "", "")
- `std::string` **ParamString** (const `std::string` ¶mName)
Print what a user would type to invoke the given option name.
- `void` **ParseCommandLine** (int argc, char **argv)
*Parse the command line, setting all of the options inside of the **CLI** (p. 1117) object to their appropriate given values.*
- `std::string` **PrintDataset** (const `std::string` &dataset)
Print a dataset type parameter (add .csv and return).
- `std::string` **PrintDefault** (const `std::string` ¶mName)
Given a parameter name, print its corresponding default value.
- `void` **PrintHelp** (const `std::string` ¶m="")
Print the help for the given parameter.
- `std::string` **PrintImport** (const `std::string` &bindingName)
*Print any imports for **CLI** (p. 1117) (there are none, so this returns an empty string).*
- `std::string` **PrintModel** (const `std::string` &model)
Print a model type parameter (add .bin and return).
- `std::string` **PrintOutputOptionInfo** ()
Print any special information about output options.
- `std::string` **PrintType** (const `util::ParamData` ¶m)
Print the type of a parameter that a user would specify from the command-line.
- `template<typename T >`
`std::string` **PrintTypeDoc** (const `util::ParamData` &data, const `typename` boost::disable_if< arma::is_arma_↵
type< T >>::type ==0, const `typename` boost::disable_if< `util::IsStdVector`< T >>::type ==0, const `typename`
boost::disable_if< `data::HasSerialize`< T >>::type ==0, const `typename` boost::disable_if< std::is_same< T,
std::tuple< `data::DatasetInfo`, arma::mat >>>::type ==0)
Return a string representing the command-line type of an option.
- `template<typename T >`
`std::string` **PrintTypeDoc** (const `util::ParamData` &data, const `typename` std::enable_if< `util::IsStdVector`< T
>::value >::type ==0)
Return a string representing the command-line type of a vector.
- `template<typename T >`
`std::string` **PrintTypeDoc** (const `util::ParamData` &data, const `typename` std::enable_if< arma::is_arma_type<
T >::value >::type ==0)
Return a string representing the command-line type of a matrix option.
- `template<typename T >`
`std::string` **PrintTypeDoc** (const `util::ParamData` &data, const `typename` std::enable_if< std::is_same< T, std::↵
::tuple< `data::DatasetInfo`, arma::mat >>::value >::type ==0)
Return a string representing the command-line type of a matrix tuple option.
- `template<typename T >`
`std::string` **PrintTypeDoc** (const `util::ParamData` &data, const `typename` boost::disable_if< arma::is_arma_↵
type< T >>::type ==0, const `typename` boost::enable_if< `data::HasSerialize`< T >>::type ==0)
Return a string representing the command-line type of a model.
- `template<typename T >`
`void` **PrintTypeDoc** (const `util::ParamData` &data, const `void` *, `void` *output)
Print the command-line type of an option into a string.
- `std::string` **PrintTypeDocs** ()
Print documentation for each of the types.

- `template<typename T >`
`std::string PrintValue (const T &value, bool quotes)`
Given a parameter type, print the corresponding value.
- `std::string ProcessOptions ()`
Base case for recursion.
- `template<typename T, typename... Args>`
`std::string ProcessOptions (const std::string ¶mName, const T &value, Args... args)`
Print an option for a command-line argument.
- `template<typename... Args>`
`std::string ProgramCall (const std::string &programName, Args... args)`
Given a program name and arguments for it, print what its invocation would be.
- `std::string ProgramCall (const std::string &programName)`
Given a program name, print a program call invocation assuming that all options are specified.
- `template<typename T >`
`void SetParam (util::ParamData &d, const boost::any &value, const typename boost::disable_if< arma::is_↵
arma_type< T >>::type !=0, const typename boost::disable_if< data::HasSerialize< T >>::type !=0, const
typename boost::disable_if< std::is_same< T, std::tuple< mlpack::data::DatasetInfo, arma::mat >>>::type
!=0, const typename boost::disable_if< std::is_same< T, bool >>::type !=0)`
This overload is called when nothing special needs to happen to the name of the parameter.
- `template<typename T >`
`void SetParam (util::ParamData &d, const boost::any &, const typename boost::enable_if< std::is_same< T,
bool >>::type !=0)`
This overload is called to set a boolean.
- `template<typename T >`
`void SetParam (util::ParamData &d, const boost::any &value, const typename std::enable_if< arma::is_arma↵
_type< T >::value||std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>::value >::type !=0)`
Set a matrix parameter, a matrix/dataset info parameter, or a serializable object.
- `template<typename T >`
`void SetParam (util::ParamData &d, const boost::any &value, const typename boost::disable_if< arma::is_↵
arma_type< T >>::type !=0, const typename boost::enable_if< data::HasSerialize< T >>::type !=0)`
Set a serializable object.
- `template<typename T >`
`void SetParam (const util::ParamData &d, const void *input, void *)`
Return a parameter casted to the given type.
- `template<typename T >`
`void StringTypeParam (const util::ParamData &, const void *, void *output)`
Return a string containing the type of a parameter.
- `template<>`
`void StringTypeParam< bool > (const util::ParamData &, const void *, void *output)`
Return "bool".
- `template<>`
`void StringTypeParam< double > (const util::ParamData &, const void *, void *output)`
Return "double".
- `template<>`
`void StringTypeParam< int > (const util::ParamData &, const void *, void *output)`
Return "int".
- `template<>`
`void StringTypeParam< std::string > (const util::ParamData &, const void *, void *output)`
Return "string".

- `template<>`
`void StringTypeParam< std::tuple< mlpack::data::DatasetInfo, arma::mat > > (const util::ParamData &, const void *, void *output)`
Return "string".
- `template<typename T >`
`std::string StringTypeParamImpl (const typename boost::disable_if< util::IsStdVector< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0)`
Return a string containing the type of the parameter.
- `template<typename T >`
`std::string StringTypeParamImpl (const typename boost::enable_if< util::IsStdVector< T >>::type *=0)`
Return a string containing the type of the parameter, for vector options.
- `template<typename T >`
`std::string StringTypeParamImpl (const typename boost::enable_if< data::HasSerialize< T >>::type *=0)`
Return a string containing the type of the parameter,.

38.12.1 Function Documentation

38.12.1.1 AddToPO() [1/4]

```
void mlpack::bindings::cli::AddToPO (
    const std::string & boostName,
    const std::string & descr,
    boost::program_options::options_description & desc,
    const typename boost::disable_if< util::IsStdVector< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, bool >>::type * = 0 )
```

Add a non-vector option to boost::program_options.

Parameters

<i>boostName</i>	The name of the option to add to boost::program_options.
<i>descr</i>	Description string for parameter.
<i>desc</i>	Options description to add parameter to.

Definition at line 33 of file `add_to_po.hpp`.

38.12.1.2 AddToPO() [2/4]

```
void mlpack::bindings::cli::AddToPO (
    const std::string & boostName,
    const std::string & descr,
```

```
boost::program_options::options_description & desc,
const typename boost::enable_if< util::IsStdVector< T >>::type * = 0,
const typename boost::disable_if< std::is_same< T, bool >>::type * = 0 )
```

Add a vector option to boost::program_options.

This overload will use the multitoken() option.

Parameters

<i>boostName</i>	The name of the option to add to boost::program_options.
<i>descr</i>	Description string for parameter.
<i>desc</i>	Options description to add parameter to.

Definition at line 52 of file add_to_po.hpp.

38.12.1.3 AddToPO() [3/4]

```
void mlpack::bindings::cli::AddToPO (
    const std::string & boostName,
    const std::string & descr,
    boost::program_options::options_description & desc,
    const typename boost::disable_if< util::IsStdVector< T >>::type * = 0,
    const typename boost::enable_if< std::is_same< T, bool >>::type * = 0 )
```

Add a boolean option to boost::program_options.

Parameters

<i>boostName</i>	The name of the option to add to boost::program_options.
<i>descr</i>	Description string for parameter.
<i>desc</i>	Options description to add parameter to.

Definition at line 70 of file add_to_po.hpp.

38.12.1.4 AddToPO() [4/4]

```
void mlpack::bindings::cli::AddToPO (
    const util::ParamData & d,
    const void * ,
    void * output )
```

Add an option to boost::program_options.

This is the function meant to be used in the **CLI** (p. 1117) function map.

Parameters

<i>d</i>	Parameter data.
<i>input</i>	Unused void pointer.
<i>output</i>	Void pointer to options_description object.

Definition at line 88 of file `add_to_po.hpp`.

References `ParamData::alias`, `ParamData::desc`, and `ParamData::name`.

38.12.1.5 `DefaultParam()`

```
void mpack::bindings::cli::DefaultParam (
    const    util::ParamData & data,
    const void * ,
    void * output )
```

Return the default value of an option.

This is the function that will be placed into the **CLI** (p. 1117) `functionMap`.

Definition at line 80 of file `default_param.hpp`.

38.12.1.6 `DefaultParamImpl()` [1/5]

```
std::string mpack::bindings::cli::DefaultParamImpl (
    const    util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if<    util::IsStdVector< T >>::type * = 0,
    const typename boost::disable_if<    data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::string >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple<    mpack::data::↵
DatasetInfo, arma::mat >>>::type * = 0 )
```

Return the default value of an option.

This is for regular types.

38.12.1.7 `DefaultParamImpl()` [2/5]

```
std::string mpack::bindings::cli::DefaultParamImpl (
    const    util::ParamData & data,
    const typename boost::enable_if<    util::IsStdVector< T >>::type * = 0 )
```

Return the default value of a vector option.

38.12.1.8 DefaultParamImpl() [3/5]

```
std::string mlpack::bindings::cli::DefaultParamImpl (
    const    util::ParamData & data,
    const typename boost::enable_if< std::is_same< T, std::string >>::type * = 0 )
```

Return the default value of a string option.

38.12.1.9 DefaultParamImpl() [4/5]

```
std::string mlpack::bindings::cli::DefaultParamImpl (
    const    util::ParamData & data,
    const typename boost::enable_if_c< arma::is_arma_type< T >::value||std::is_same< T,
std::tuple< mlpack::data::DatasetInfo, arma::mat >>::value >::type * = 0 )
```

Return the default value of a matrix option, a tuple option, a serializable option, or a string option (this returns the default filename, or "" if the default is no file).

38.12.1.10 DefaultParamImpl() [5/5]

```
std::string mlpack::bindings::cli::DefaultParamImpl (
    const    util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

Return the default value of a model option (this returns the default filename, or "" if the default is no file).

38.12.1.11 DeleteAllocatedMemory()

```
void mlpack::bindings::cli::DeleteAllocatedMemory (
    const    util::ParamData & d,
    const void * ,
    void * )
```

Definition at line 50 of file delete_allocated_memory.hpp.

38.12.1.12 DeleteAllocatedMemoryImpl() [1/3]

```
void mlpack::bindings::cli::DeleteAllocatedMemoryImpl (
    const util::ParamData & ,
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0 )
```

Definition at line 22 of file delete_allocated_memory.hpp.

38.12.1.13 DeleteAllocatedMemoryImpl() [2/3]

```
void mlpack::bindings::cli::DeleteAllocatedMemoryImpl (
    const util::ParamData & ,
    const typename boost::enable_if< arma::is_arma_type< T >>::type * = 0 )
```

Definition at line 31 of file delete_allocated_memory.hpp.

38.12.1.14 DeleteAllocatedMemoryImpl() [3/3]

```
void mlpack::bindings::cli::DeleteAllocatedMemoryImpl (
    const util::ParamData & d,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

Definition at line 39 of file delete_allocated_memory.hpp.

References `ParamData::value`.

38.12.1.15 EndProgram()

```
void mlpack::bindings::cli::EndProgram ( ) [inline]
```

Handle command-line program termination.

If `-help` or `-info` was passed, we won't make it here, so we don't have to write any contingencies for that.

Definition at line 26 of file end_program.hpp.

References `CLI::functionMap`, `Timers::GetAllTimers()`, `CLI::GetSingleton()`, `CLI::HasParam()`, `Log::Info`, `ParamData::input`, `CLI::Parameters()`, `Timers::PrintTimer()`, `Timers::StopAllTimers()`, `CLI::timer`, and `ParamData::tname`.

38.12.1.16 GetAllocatedMemory() [1/4]

```
void* mlpack::bindings::cli::GetAllocatedMemory (
    const    util::ParamData & ,
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0 )
```

Definition at line 23 of file get_allocated_memory.hpp.

38.12.1.17 GetAllocatedMemory() [2/4]

```
void* mlpack::bindings::cli::GetAllocatedMemory (
    const    util::ParamData & ,
    const typename boost::enable_if< arma::is_arma_type< T >>::type * = 0 )
```

Definition at line 32 of file get_allocated_memory.hpp.

38.12.1.18 GetAllocatedMemory() [3/4]

```
void* mlpack::bindings::cli::GetAllocatedMemory (
    const    util::ParamData & d,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

Definition at line 40 of file get_allocated_memory.hpp.

References ParamData::value.

38.12.1.19 GetAllocatedMemory() [4/4]

```
void mlpack::bindings::cli::GetAllocatedMemory (
    const    util::ParamData & d,
    const void * ,
    void * output )
```

Definition at line 52 of file get_allocated_memory.hpp.

38.12.1.20 GetBindingName()

```
std::string mlpack::bindings::cli::GetBindingName (
    const std::string & bindingName ) [inline]
```

Given the name of a binding, print its command-line name (this returns "mlpack_<bindingName>").

38.12.1.21 GetParam() [1/5]

```
T& mlpack::bindings::cli::GetParam (
    util::ParamData & d,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple< mlpack::data::↵
DatasetInfo, arma::mat >>>::type * = 0 )
```

This overload is called when nothing special needs to happen to the name of the parameter.

Parameters

<i>d</i>	ParamData object to get parameter value from.
----------	---

Definition at line 29 of file get_param.hpp.

References ParamData::value.

38.12.1.22 GetParam() [2/5]

```
T& mlpack::bindings::cli::GetParam (
    util::ParamData & d,
    const typename boost::enable_if< arma::is_arma_type< T >>::type * = 0 )
```

Return a matrix parameter.

Parameters

<i>d</i>	ParamData object to get parameter value from.
----------	---

Definition at line 46 of file get_param.hpp.

References ParamData::input, mlpack::data::Load(), ParamData::loaded, ParamData::noTranspose, and ParamData↵
::value.

38.12.1.23 GetParam() [3/5]

```
T& mlpack::bindings::cli::GetParam (
    util::ParamData & d,
    const typename boost::enable_if< std::is_same< T, std::tuple< mlpack::data::↵
DatasetInfo, arma::mat >>>::type * = 0 )
```

Return a matrix/dataset info parameter.

Parameters

<i>d</i>	ParamData object to get parameter value from.
----------	---

Definition at line 77 of file get_param.hpp.

References ParamData::input, mlpack::data::Load(), ParamData::loaded, ParamData::noTranspose, and ParamData↵
::value.

38.12.1.24 GetParam() [4/5]

```
T*& mlpack::bindings::cli::GetParam (
    util::ParamData & d,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

Return a serializable object.

Parameters

<i>d</i>	ParamData object to get parameter value from.
----------	---

Definition at line 103 of file get_param.hpp.

References ParamData::input, mlpack::data::Load(), ParamData::loaded, and ParamData::value.

38.12.1.25 GetParam() [5/5]

```
void mlpack::bindings::cli::GetParam (
    const util::ParamData & d,
    const void * ,
    void * output )
```

Return a parameter casted to the given type.

Type checking does not happen here!

Parameters

<i>d</i>	Parameter information.
<i>input</i>	Unused parameter.
<i>output</i>	Place to store pointer to value.

Definition at line 132 of file `get_param.hpp`.

38.12.1.26 `GetPrintableParam()` [1/5]

```
std::string mlpack::bindings::cli::GetPrintableParam (
    const    util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if<    util::IsStdVector< T >>::type * = 0,
    const typename boost::disable_if<    data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple<    data::DatasetInfo,
arma::mat >>>::type * = 0 )
```

Print an option.

Print an option.

Definition at line 26 of file `get_printable_param.hpp`.

38.12.1.27 `GetPrintableParam()` [2/5]

```
std::string mlpack::bindings::cli::GetPrintableParam (
    const    util::ParamData & data,
    const typename std::enable_if<    util::IsStdVector< T >::value >::type * = 0 )
```

Print a vector option, with spaces between it.

38.12.1.28 `GetPrintableParam()` [3/5]

```
std::string mlpack::bindings::cli::GetPrintableParam (
    const    util::ParamData & data,
    const typename std::enable_if< arma::is_arma_type< T >::value||std::is_same< T, std::
::tuple<    data::DatasetInfo, arma::mat >>>::value >::type * = 0 )
```

Print a matrix/tuple option (this just prints the filename).

38.12.1.29 GetPrintableParam() [4/5]

```
std::string mlpack::bindings::cli::GetPrintableParam (
    const    util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type *   = 0,
    const typename boost::enable_if<    data::HasSerialize< T >>::type *   = 0 )
```

Print a model option (this just prints the filename).

Print a model option (this just prints the filename).

Definition at line 75 of file get_printable_param.hpp.

38.12.1.30 GetPrintableParam() [5/5]

```
void mlpack::bindings::cli::GetPrintableParam (
    const    util::ParamData & data,
    const void * ,
    void * output )
```

Print an option into a std::string.

This should print a short, one-line representation of the object. The string will be stored in the output pointer.

Definition at line 69 of file get_printable_param.hpp.

38.12.1.31 GetPrintableParamName() [1/5]

```
std::string mlpack::bindings::cli::GetPrintableParamName (
    const    util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type *   = 0,
    const typename boost::disable_if<    data::HasSerialize< T >>::type *   = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple<    data::DatasetInfo,
arma::mat >>>::type *   = 0 )
```

Get the parameter name for a type that has no special handling.

38.12.1.32 GetPrintableParamName() [2/5]

```
std::string mlpack::bindings::cli::GetPrintableParamName (
    const    util::ParamData & data,
    const typename boost::enable_if< arma::is_arma_type< T >>::type *   = 0 )
```

Get the parameter name for a matrix type (where the user has to pass the file that holds the matrix).

38.12.1.33 GetPrintableParamName() [3/5]

```
std::string mlpack::bindings::cli::GetPrintableParamName (
    const util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

Get the parameter name for a serializable model type (where the user has to pass the file that holds the matrix).

38.12.1.34 GetPrintableParamName() [4/5]

```
std::string mlpack::bindings::cli::GetPrintableParamName (
    const util::ParamData & data,
    const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo,
    arma::mat >>>::type * = 0 )
```

Get the parameter name for a mapped matrix type (where the user has to pass the file that holds the matrix).

38.12.1.35 GetPrintableParamName() [5/5]

```
void mlpack::bindings::cli::GetPrintableParamName (
    const util::ParamData & d,
    const void * ,
    void * output )
```

Get the parameter's name as seen by the user.

Definition at line 67 of file get_printable_param_name.hpp.

38.12.1.36 GetPrintableParamValue() [1/5]

```
std::string mlpack::bindings::cli::GetPrintableParamValue (
    const util::ParamData & data,
    const std::string & value,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo,
    arma::mat >>>::type * = 0 )
```

Get the parameter name for a type that has no special handling.

38.12.1.37 GetPrintableParamValue() [2/5]

```
std::string mlpack::bindings::cli::GetPrintableParamValue (
    const util::ParamData & data,
    const std::string & value,
    const typename boost::enable_if< arma::is_arma_type< T >>::type * = 0 )
```

Get the parameter name for a matrix type (where the user has to pass the file that holds the matrix).

38.12.1.38 GetPrintableParamValue() [3/5]

```
std::string mlpack::bindings::cli::GetPrintableParamValue (
    const util::ParamData & data,
    const std::string & value,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

Get the parameter name for a serializable model type (where the user has to pass the file that holds the matrix).

38.12.1.39 GetPrintableParamValue() [4/5]

```
std::string mlpack::bindings::cli::GetPrintableParamValue (
    const util::ParamData & data,
    const std::string & value,
    const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo,
    arma::mat >>>::type * = 0 )
```

Get the parameter name for a mapped matrix type (where the user has to pass the file that holds the matrix).

38.12.1.40 GetPrintableParamValue() [5/5]

```
void mlpack::bindings::cli::GetPrintableParamValue (
    const util::ParamData & d,
    const void * input,
    void * output )
```

Get the parameter's name as seen by the user.

Definition at line 71 of file `get_printable_param_value.hpp`.

38.12.1.41 GetPrintableType() [1/6]

```
std::string mlpack::bindings::cli::GetPrintableType (
    const    util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if<    util::IsStdVector< T >>::type * = 0,
    const typename boost::disable_if<    data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple<    data::DatasetInfo,
arma::mat >>>::type * = 0 )
```

Return a string representing the command-line type of an option.

38.12.1.42 GetPrintableType() [2/6]

```
std::string mlpack::bindings::cli::GetPrintableType (
    const    util::ParamData & data,
    const typename std::enable_if<    util::IsStdVector< T >::value >::type * = 0 )
```

Return a string representing the command-line type of a vector.

38.12.1.43 GetPrintableType() [3/6]

```
std::string mlpack::bindings::cli::GetPrintableType (
    const    util::ParamData & data,
    const typename std::enable_if< arma::is_arma_type< T >::value >::type * = 0 )
```

Return a string representing the command-line type of a matrix option.

38.12.1.44 GetPrintableType() [4/6]

```
std::string mlpack::bindings::cli::GetPrintableType (
    const    util::ParamData & data,
    const typename std::enable_if< std::is_same< T, std::tuple<    data::DatasetInfo,
arma::mat >>>::value >::type * = 0 )
```

Return a string representing the command-line type of a matrix tuple option.

38.12.1.45 GetPrintableType() [5/6]

```
std::string mlpack::bindings::cli::GetPrintableType (
    const    util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type *   = 0,
    const typename boost::enable_if<    data::HasSerialize< T >>::type *   = 0 )
```

Return a string representing the command-line type of a model.

38.12.1.46 GetPrintableType() [6/6]

```
void mlpack::bindings::cli::GetPrintableType (
    const    util::ParamData & data,
    const void * ,
    void * output )
```

Print the command-line type of an option into a string.

Definition at line 70 of file `get_printable_type.hpp`.

38.12.1.47 GetRawParam() [1/4]

```
T& mlpack::bindings::cli::GetRawParam (
    util::ParamData & d,
    const typename boost::disable_if< arma::is_arma_type< T >>::type *   = 0,
    const typename boost::disable_if<    data::HasSerialize< T >>::type *   = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple< mlpack::data::↵
DatasetInfo, arma::mat >>>::type *   = 0 )
```

This overload is called when nothing special needs to happen to the name of the parameter.

Definition at line 28 of file `get_raw_param.hpp`.

References `ParamData::value`.

38.12.1.48 GetRawParam() [2/4]

```
T& mlpack::bindings::cli::GetRawParam (
    util::ParamData & d,
    const typename boost::enable_if_c< arma::is_arma_type< T >::value||std::is_same< T,
std::tuple< mlpack::data::DatasetInfo, arma::mat >>::value >::type *   = 0 )
```

Return a matrix parameter.

Definition at line 43 of file `get_raw_param.hpp`.

References `ParamData::value`.

38.12.1.49 GetRawParam() [3/4]

```
T*& mlpack::bindings::cli::GetRawParam (
    util::ParamData & d,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

Return the name of a model parameter.

Definition at line 60 of file `get_raw_param.hpp`.

References `ParamData::value`.

38.12.1.50 GetRawParam() [4/4]

```
void mlpack::bindings::cli::GetRawParam (
    const util::ParamData & d,
    const void * ,
    void * output )
```

Return a parameter casted to the given type.

Type checking does not happen here!

Parameters

<i>d</i>	Parameter information.
<i>input</i>	Unused parameter.
<i>output</i>	Place to store pointer to value.

Definition at line 80 of file `get_raw_param.hpp`.

38.12.1.51 IgnoreCheck()

```
bool mlpack::bindings::cli::IgnoreCheck (
    const T & ) [inline]
```

Return whether or not a runtime check on parameters should be ignored.

We don't ignore any runtime checks for **CLI** (p. 1117) bindings, so this always returns false.

Definition at line 111 of file `print_doc_functions.hpp`.

38.12.1.52 MapParameterName() [1/3]

```
std::string mlpack::bindings::cli::MapParameterName (
    const std::string & identifier,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple< mlpack::data::↵
DatasetInfo, arma::mat >>>::type * = 0 )
```

If needed, map the parameter name to the name that is used by boost::program_options.

This overload simply returns the same name, so it is used for primitive types.

Definition at line 28 of file map_parameter_name.hpp.

Referenced by CLIOption< N >::CLIOption().

38.12.1.53 MapParameterName() [2/3]

```
std::string mlpack::bindings::cli::MapParameterName (
    const std::string & identifier,
    const typename boost::enable_if_c< arma::is_arma_type< T >::value||std::is_same< T,
std::tuple< mlpack::data::DatasetInfo, arma::mat >>::value|| data::HasSerialize< T >::value >↵
::type * = 0 )
```

Map the parameter name to the name that is used by boost::program_options.

This overload addresses matrices and models, where the parameter name has "_file" appended to it (since a filename will be provided).

Definition at line 44 of file map_parameter_name.hpp.

38.12.1.54 MapParameterName() [3/3]

```
void mlpack::bindings::cli::MapParameterName (
    const util::ParamData & d,
    const void * ,
    void * output )
```

Map the parameter name to the name seen by boost::program_options.

Parameters

<i>d</i>	Parameter data.
<i>input</i>	Unused parameter.
<i>output</i>	Pointer to std::string that will hold the mapped name.

Definition at line 63 of file map_parameter_name.hpp.

References ParamData::name.

38.12.1.55 OutputParam()

```
void mlpack::bindings::cli::OutputParam (
    const    util::ParamData & data,
    const void * ,
    void * )
```

Output an option.

This is the function that will be called by the **CLI** (p. 1117) module.

Definition at line 74 of file output_param.hpp.

38.12.1.56 OutputParamImpl() [1/5]

```
void mlpack::bindings::cli::OutputParamImpl (
    const    util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if<    util::IsStdVector< T >>::type * = 0,
    const typename boost::disable_if<    data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple<    data::DatasetInfo,
    arma::mat >>>::type * = 0 )
```

Output an option (print to stdout).

38.12.1.57 OutputParamImpl() [2/5]

```
void mlpack::bindings::cli::OutputParamImpl (
    const    util::ParamData & data,
    const typename boost::enable_if<    util::IsStdVector< T >>::type * = 0 )
```

Output a vector option (print to stdout).

38.12.1.58 OutputParamImpl() [3/5]

```
void mlpack::bindings::cli::OutputParamImpl (
    const util::ParamData & data,
    const typename boost::enable_if< arma::is_arma_type< T >>::type * = 0 )
```

Output a matrix option (this saves it to the given file).

38.12.1.59 OutputParamImpl() [4/5]

```
void mlpack::bindings::cli::OutputParamImpl (
    const util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

Output a serializable class option (this saves it to the given file).

38.12.1.60 OutputParamImpl() [5/5]

```
void mlpack::bindings::cli::OutputParamImpl (
    const util::ParamData & data,
    const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo,
    arma::mat >>>::type * = 0 )
```

Output a mapped dataset.

38.12.1.61 PARAM_FLAG() [1/3]

```
mlpack::bindings::cli::PARAM_FLAG (
    "help" ,
    "Default help info." ,
    "h" )
```

38.12.1.62 PARAM_FLAG() [2/3]

```
mlpack::bindings::cli::PARAM_FLAG (
    "verbose" ,
    "Display informational messages and the full list of " "parameters and timers at the
end of execution." ,
    "v" )
```

38.12.1.63 PARAM_FLAG() [3/3]

```
mlpack::bindings::cli::PARAM_FLAG (
    "version" ,
    "Display the version of mlpack." ,
    "v" )
```

38.12.1.64 PARAM_STRING_IN()

```
mlpack::bindings::cli::PARAM_STRING_IN (
    "info" ,
    "Print help on a specific option." ,
    "" ,
    "" )
```

38.12.1.65 ParamString()

```
std::string mlpack::bindings::cli::ParamString (
    const std::string & paramName ) [inline]
```

Print what a user would type to invoke the given option name.

Note that the name *must* exist in the **CLI** (p. 1117) module. (Note that because of the way ProgramInfo is structured, this doesn't mean that all of the **PARAM_*()** declarataions need to come before the **PROGRAM_INFO()** (p. 2733) declaration.)

38.12.1.66 ParseCommandLine()

```
void mlpack::bindings::cli::ParseCommandLine (
    int argc,
    char ** argv )
```

Parse the command line, setting all of the options inside of the **CLI** (p. 1117) object to their appropriate given values.

Definition at line 35 of file parse_command_line.hpp.

References `Log::Debug`, `CLI::didParse`, `Log::Fatal`, `CLI::functionMap`, `CLI::GetSingleton()`, `mlpack::util::GetVersion()`, `CLI::HasParam()`, `PrefixedOutputStream::ignoreInput`, `Log::Info`, `ParamData::name`, `CLI::Parameters()`, `PrintHelp()`, `CLI::ProgramName()`, `ParamData::required`, `ParamData::tname`, `ParamData::value`, and `ParamData::wasPassed`.

38.12.1.67 PrintDataset()

```
std::string mpack::bindings::cli::PrintDataset (
    const std::string & dataset ) [inline]
```

Print a dataset type parameter (add .csv and return).

38.12.1.68 PrintDefault()

```
std::string mpack::bindings::cli::PrintDefault (
    const std::string & paramName ) [inline]
```

Given a parameter name, print its corresponding default value.

38.12.1.69 PrintHelp()

```
void mpack::bindings::cli::PrintHelp (
    const std::string & param = "" )
```

Print the help for the given parameter.

If no parameter is specified, then help will be printed for all parameters.

Parameters

<i>param</i>	Parameter name to print help for.
--------------	-----------------------------------

Referenced by ParseCommandLine().

38.12.1.70 PrintImport()

```
std::string mpack::bindings::cli::PrintImport (
    const std::string & bindingName ) [inline]
```

Print any imports for **CLI** (p. 1117) (there are none, so this returns an empty string).

38.12.1.71 PrintModel()

```
std::string mlpack::bindings::cli::PrintModel (
    const std::string & model ) [inline]
```

Print a model type parameter (add .bin and return).

38.12.1.72 PrintOutputOptionInfo()

```
std::string mlpack::bindings::cli::PrintOutputOptionInfo ( ) [inline]
```

Print any special information about output options.

38.12.1.73 PrintType()

```
std::string mlpack::bindings::cli::PrintType (
    const util::ParamData & param ) [inline]
```

Print the type of a parameter that a user would specify from the command-line.

38.12.1.74 PrintTypeDoc() [1/6]

```
std::string mlpack::bindings::cli::PrintTypeDoc (
    const util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if< util::IsStdVector< T >>::type * = 0,
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo,
arma::mat >>>::type * = 0 )
```

Return a string representing the command-line type of an option.

38.12.1.75 PrintTypeDoc() [2/6]

```
std::string mlpack::bindings::cli::PrintTypeDoc (
    const util::ParamData & data,
    const typename std::enable_if< util::IsStdVector< T >::value >::type * = 0 )
```

Return a string representing the command-line type of a vector.

38.12.1.76 PrintTypeDoc() [3/6]

```
std::string mlpack::bindings::cli::PrintTypeDoc (
    const util::ParamData & data,
    const typename std::enable_if< arma::is_arma_type< T >::value >::type * = 0 )
```

Return a string representing the command-line type of a matrix option.

38.12.1.77 PrintTypeDoc() [4/6]

```
std::string mlpack::bindings::cli::PrintTypeDoc (
    const util::ParamData & data,
    const typename std::enable_if< std::is_same< T, std::tuple< data::DatasetInfo,
    arma::mat >>::value >::type * = 0 )
```

Return a string representing the command-line type of a matrix tuple option.

38.12.1.78 PrintTypeDoc() [5/6]

```
std::string mlpack::bindings::cli::PrintTypeDoc (
    const util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

Return a string representing the command-line type of a model.

38.12.1.79 PrintTypeDoc() [6/6]

```
void mlpack::bindings::cli::PrintTypeDoc (
    const util::ParamData & data,
    const void * ,
    void * output )
```

Print the command-line type of an option into a string.

Definition at line 72 of file `print_type_doc.hpp`.

38.12.1.80 PrintTypeDocs()

```
std::string mlpack::bindings::cli::PrintTypeDocs ( ) [inline]
```

Print documentation for each of the types.

38.12.1.81 PrintValue()

```
std::string mlpack::bindings::cli::PrintValue (
    const T & value,
    bool quotes ) [inline]
```

Given a parameter type, print the corresponding value.

38.12.1.82 ProcessOptions() [1/2]

```
std::string mlpack::bindings::cli::ProcessOptions ( ) [inline]
```

Base case for recursion.

38.12.1.83 ProcessOptions() [2/2]

```
std::string mlpack::bindings::cli::ProcessOptions (
    const std::string & paramName,
    const T & value,
    Args... args )
```

Print an option for a command-line argument.

38.12.1.84 ProgramCall() [1/2]

```
std::string mlpack::bindings::cli::ProgramCall (
    const std::string & programName,
    Args... args )
```

Given a program name and arguments for it, print what its invocation would be.

38.12.1.85 ProgramCall() [2/2]

```
std::string mlpack::bindings::cli::ProgramCall (
    const std::string & programName ) [inline]
```

Given a program name, print a program call invocation assuming that all options are specified.

38.12.1.86 SetParam() [1/5]

```
void mlpack::bindings::cli::SetParam (
    util::ParamData & d,
    const boost::any & value,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple< mlpack::data::↔
DatasetInfo, arma::mat >>>::type * = 0,
    const typename boost::disable_if< std::is_same< T, bool >>::type * = 0 )
```

This overload is called when nothing special needs to happen to the name of the parameter.

Definition at line 27 of file set_param.hpp.

References ParamData::value.

38.12.1.87 SetParam() [2/5]

```
void mlpack::bindings::cli::SetParam (
    util::ParamData & d,
    const boost::any & ,
    const typename boost::enable_if< std::is_same< T, bool >>::type * = 0 )
```

This overload is called to set a boolean.

Definition at line 44 of file set_param.hpp.

References ParamData::value, and ParamData::wasPassed.

38.12.1.88 SetParam() [3/5]

```
void mlpack::bindings::cli::SetParam (
    util::ParamData & d,
    const boost::any & value,
    const typename std::enable_if< arma::is_arma_type< T >::value||std::is_same< T, std::
::tuple< data::DatasetInfo, arma::mat >>::value >::type * = 0 )
```

Set a matrix parameter, a matrix/dataset info parameter, or a serializable object.

These set the filename referring to the parameter.

Definition at line 58 of file set_param.hpp.

References ParamData::value.

38.12.1.89 SetParam() [4/5]

```
void mlpack::bindings::cli::SetParam (
    util::ParamData & d,
    const boost::any & value,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

Set a serializable object.

This sets the filename referring to the parameter.

Definition at line 76 of file set_param.hpp.

References ParamData::value.

38.12.1.90 SetParam() [5/5]

```
void mlpack::bindings::cli::SetParam (
    const util::ParamData & d,
    const void * input,
    void * )
```

Return a parameter casted to the given type.

Type checking does not happen here!

Parameters

<i>d</i>	Parameter information.
<i>input</i>	Unused parameter.
<i>output</i>	Place to store pointer to value.

Definition at line 97 of file set_param.hpp.

38.12.1.91 StringTypeParam()

```
void mlpack::bindings::cli::StringTypeParam (
    const    util::ParamData & ,
    const void * ,
    void * output )
```

Return a string containing the type of a parameter.

This overload is used if we don't have a primitive type.

Definition at line 51 of file string_type_param.hpp.

References StringTypeParam< bool >(), StringTypeParam< double >(), and StringTypeParam< int >().

38.12.1.92 StringTypeParam< bool >()

```
void mlpack::bindings::cli::StringTypeParam< bool > (
    const    util::ParamData & ,
    const void * ,
    void * output ) [inline]
```

Return "bool".

Referenced by StringTypeParam().

38.12.1.93 StringTypeParam< double >()

```
void mlpack::bindings::cli::StringTypeParam< double > (
    const    util::ParamData & ,
    const void * ,
    void * output ) [inline]
```

Return "double".

Referenced by StringTypeParam().

38.12.1.94 StringTypeParam< int >()

```
void mlpack::bindings::cli::StringTypeParam< int > (
    const util::ParamData & ,
    const void * ,
    void * output ) [inline]
```

Return "int".

Referenced by StringTypeParam().

38.12.1.95 StringTypeParam< std::string >()

```
void mlpack::bindings::cli::StringTypeParam< std::string > (
    const util::ParamData & ,
    const void * ,
    void * output ) [inline]
```

Return "string".

38.12.1.96 StringTypeParam< std::tuple< mlpack::data::DatasetInfo, arma::mat > >()

```
void mlpack::bindings::cli::StringTypeParam< std::tuple< mlpack::data::DatasetInfo, arma::mat > > (
    const util::ParamData & ,
    const void * ,
    void * output ) [inline]
```

Return "string";.

38.12.1.97 StringTypeParamImpl() [1/3]

```
std::string mlpack::bindings::cli::StringTypeParamImpl (
    const typename boost::disable_if< util::IsStdVector< T >>::type * = 0,
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0 )
```

Return a string containing the type of the parameter.

38.12.1.98 StringTypeParamImpl() [2/3]

```
std::string mlpack::bindings::cli::StringTypeParamImpl (
    const typename boost::enable_if< util::IsStdVector< T >>::type * = 0 )
```

Return a string containing the type of the parameter, for vector options.

38.12.1.99 StringTypeParamImpl() [3/3]

```
std::string mlpack::bindings::cli::StringTypeParamImpl (
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

Return a string containing the type of the parameter,.

38.13 mlpack::bindings::markdown Namespace Reference

Classes

- class **BindingInfo**
The **BindingInfo** (p. 984) class is used by the Markdown documentation generator to store multiple ProgramDoc objects, indexed by both the binding name (i.e.
- class **MDOption**
The Markdown option class.
- class **ProgramDocWrapper**

Functions

- template<typename T >
void **DefaultParam** (const util::ParamData &data, const void *, void *output)
Print the default value of a parameter into the output string.
- std::string **GetBindingName** (const std::string &bindingName)
Given the name of the binding, print the name for the current language (as given by **BindingInfo** (p. 984)).
- std::string **GetBindingName** (const std::string &language, const std::string &name)
Given a language name and a binding name, return the name of that binding for that language.
- template<typename T >
void **GetParam** (const util::ParamData &d, const void *, void *output)
All Markdown binding types are exactly what is held in the ParamData, so no special handling is necessary.
- template<typename T >
std::string **GetPrintableParam** (const util::ParamData &data, const typename boost::disable_if< arma::is_↵
arma_type< T >>::type *=0, const typename boost::disable_if< util::IsStdVector< T >>::type *=0, const
typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< std::is_↵
_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)
Print an option of a simple type.

- `template<typename T >`
`std::string GetPrintableParam (const util::ParamData &data, const typename boost::enable_if< util::IsStd↵`
`Vector< T >>::type !=0)`
Print a vector option, with spaces between it.
- `template<typename T >`
`std::string GetPrintableParam (const util::ParamData &data, const typename boost::enable_if< arma::is_↵`
`arma_type< T >>::type !=0)`
Print a matrix option (this prints its size).
- `template<typename T >`
`std::string GetPrintableParam (const util::ParamData &data, const typename boost::disable_if< arma::is_↵`
`arma_type< T >>::type !=0, const typename boost::enable_if< data::HasSerialize< T >>::type !=0)`
Print a serializable class option (this prints the class name).
- `template<typename T >`
`std::string GetPrintableParam (const util::ParamData &data, const typename boost::enable_if< std::is_same<`
`T, std::tuple< data::DatasetInfo, arma::mat >>>::type !=0)`
Print a combination DatasetInfo/matrix parameter.
- `template<typename T >`
`void GetPrintableParam (const util::ParamData &data, const void *, void *output)`
Print an option into a std::string.
- `template<typename T >`
`std::string GetPrintableParamName (const util::ParamData &data, const typename boost::disable_if< arma↵`
`::is_arma_type< T >>::type !=0, const typename boost::disable_if< data::HasSerialize< T >>::type !=0,`
`const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type !=0)`
Get the parameter name for a type that has no special handling.
- `template<typename T >`
`std::string GetPrintableParamName (const util::ParamData &data, const typename boost::enable_if< arma↵`
`::is_arma_type< T >>::type !=0)`
Get the parameter name for a matrix type (where the user has to pass the file that holds the matrix).
- `template<typename T >`
`std::string GetPrintableParamName (const util::ParamData &data, const typename boost::enable_if< arma↵`
`::is_arma_type< T >>::type !=0, const typename boost::enable_if< data::HasSerialize< T >>::type !=0)`
Get the parameter name for a serializable model type (where the user has to pass the file that holds the matrix).
- `template<typename T >`
`std::string GetPrintableParamName (const util::ParamData &data, const typename boost::enable_if< std↵`
`::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type !=0)`
Get the parameter name for a mapped matrix type (where the user has to pass the file that holds the matrix).
- `template<typename T >`
`void GetPrintableParamName (const util::ParamData &d, const void *, void *output)`
Get the parameter's name as seen by the user.
- `template<typename T >`
`std::string GetPrintableParamValue (const util::ParamData &data, const std::string &value, const typename`
`boost::disable_if< arma::is_arma_type< T >>::type !=0, const typename boost::disable_if< data::Has↵`
`Serialize< T >>::type !=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo,`
`arma::mat >>>::type !=0)`
Get the parameter name for a type that has no special handling.
- `template<typename T >`
`std::string GetPrintableParamValue (const util::ParamData &data, const std::string &value, const typename`
`boost::enable_if< arma::is_arma_type< T >>::type !=0)`
Get the parameter name for a matrix type (where the user has to pass the file that holds the matrix).

- `template<typename T >`
`std::string GetPrintableParamValue (const util::ParamData &data, const std::string &value, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)`
Get the parameter name for a serializable model type (where the user has to pass the file that holds the matrix).
- `template<typename T >`
`std::string GetPrintableParamValue (const util::ParamData &data, const std::string &value, const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)`
Get the parameter name for a mapped matrix type (where the user has to pass the file that holds the matrix).
- `template<typename T >`
`void GetPrintableParamValue (const util::ParamData &d, const void *input, void *output)`
Get the parameter's name as seen by the user.
- `template<typename T >`
`void GetPrintableType (const util::ParamData &data, const void *, void *output)`
Print the type of a parameter into the output string.
- `template<typename T >`
`std::string GetPrintableType (const util::ParamData &data)`
Print the type of a parameter.
- `template<typename T >`
`bool IgnoreCheck (const T &t)`
Return whether or not a runtime check on parameters should be ignored.
- `template<typename T >`
`bool IsSerializable (const typename boost::disable_if< data::HasSerialize< T >>::type *=0)`
Return false, because the type is not serializable.
- `template<typename T >`
`bool IsSerializable (const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)`
Return false, because even though the type is serializable, it is an Armadillo type not an mlpack model.
- `template<typename T >`
`bool IsSerializable (const typename boost::enable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0)`
Return true, because the type is serializable.
- `template<typename T >`
`void IsSerializable (const util::ParamData &, const void *, void *output)`
Return whether or not the type is serializable.
- `std::string ParamString (const std::string ¶mName)`
Print what a user would type to invoke the given option name.
- `std::string ParamType (const util::ParamData &d)`
Print the user-encountered type of an option.
- `std::string PrintDataset (const std::string &dataset)`
Print a dataset type parameter (add .csv and return).
- `std::string PrintDefault (const std::string ¶mName)`
Print the default value of an option, unless it is required (in which case Markdown italicized '-' is printed).
- `std::string PrintImport (const std::string &bindingName)`
Print any imports that need to be done before using the binding.
- `std::string PrintLanguage (const std::string &language)`
Print the name of the given language.
- `std::string PrintModel (const std::string &model)`
Print a model type parameter (add .bin and return).
- `std::string PrintOutputOptionInfo (const std::string &language)`

Print any special information about output options.

- `template<typename T >`
`std::string PrintTypeDoc (const util::ParamData &data)`
Print the type of a parameter into the output string.
- `std::string PrintTypeDocs ()`
Print details about the different types for a language.
- `template<typename T >`
`std::string PrintValue (const T &value, bool quotes)`
Given a parameter type, print the corresponding value.
- `template<typename... Args>`
`std::string ProgramCall (const std::string &programName, Args... args)`
Given a program name and arguments for it, print what its invocation would be.
- `std::string ProgramCall (const std::string &programName)`
Given a program name, print a call assuming that all arguments are specified.

38.13.1 Function Documentation

38.13.1.1 DefaultParam()

```
void mlpack::bindings::markdown::DefaultParam (
    const util::ParamData & data,
    const void * ,
    void * output )
```

Print the default value of a parameter into the output string.

The type printed depends on the current setting of **BindingInfo::Language()** (p. 985).

Definition at line 31 of file `default_param.hpp`.

References **BindingInfo::Language()**.

38.13.1.2 GetBindingName() [1/2]

```
std::string mlpack::bindings::markdown::GetBindingName (
    const std::string & bindingName ) [inline]
```

Given the name of the binding, print the name for the current language (as given by **BindingInfo** (p. 984)).

38.13.1.3 GetBindingName() [2/2]

```
std::string mlpack::bindings::markdown::GetBindingName (
    const std::string & language,
    const std::string & name )
```

Given a language name and a binding name, return the name of that binding for that language.

Note that if a new language is added to the mlpack bindings, this method will need to be updated so that documentation can be successfully generated for that language.

38.13.1.4 GetParam()

```
void mlpack::bindings::markdown::GetParam (
    const util::ParamData & d,
    const void * ,
    void * output )
```

All Markdown binding types are exactly what is held in the ParamData, so no special handling is necessary.

Definition at line 26 of file `get_param.hpp`.

References `ParamData::value`.

38.13.1.5 GetPrintableParam() [1/6]

```
std::string mlpack::bindings::markdown::GetPrintableParam (
    const util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if< util::IsStdVector< T >>::type * = 0,
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo,
    arma::mat >>>::type * = 0 )
```

Print an option of a simple type.

Print an option.

Definition at line 26 of file `get_printable_param.hpp`.

38.13.1.6 GetPrintableParam() [2/6]

```
std::string mlpack::bindings::markdown::GetPrintableParam (
    const util::ParamData & data,
    const typename boost::enable_if< util::IsStdVector< T >>::type * = 0 )
```

Print a vector option, with spaces between it.

Definition at line 43 of file `get_printable_param.hpp`.

38.13.1.7 GetPrintableParam() [3/6]

```
std::string mlpack::bindings::markdown::GetPrintableParam (
    const util::ParamData & data,
    const typename boost::enable_if< arma::is_arma_type< T >>::type * = 0 )
```

Print a matrix option (this prints its size).

Print a matrix option (this just prints the filename).

Definition at line 59 of file `get_printable_param.hpp`.

38.13.1.8 GetPrintableParam() [4/6]

```
std::string mlpack::bindings::markdown::GetPrintableParam (
    const util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

Print a serializable class option (this prints the class name).

Print a serializable class option (this just prints the filename).

Print a model option (this just prints the filename).

Definition at line 75 of file `get_printable_param.hpp`.

38.13.1.9 GetPrintableParam() [5/6]

```
std::string mlpack::bindings::markdown::GetPrintableParam (
    const util::ParamData & data,
    const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo,
arma::mat >>>::type * = 0 )
```

Print a combination DatasetInfo/matrix parameter.

Print a mapped matrix option (this just prints the filename).

Definition at line 89 of file get_printable_param.hpp.

38.13.1.10 GetPrintableParam() [6/6]

```
void mlpack::bindings::markdown::GetPrintableParam (
    const util::ParamData & data,
    const void * ,
    void * output )
```

Print an option into a std::string.

This should print a short, one-line representation of the object. The string will be stored in the output pointer.

Parameters

<i>data</i>	Parameter data struct.
<i>input</i>	Unused parameter.
<i>output</i>	Output storage for the string.

Definition at line 114 of file get_printable_param.hpp.

38.13.1.11 GetPrintableParamName() [1/5]

```
std::string mlpack::bindings::markdown::GetPrintableParamName (
    const util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo,
arma::mat >>>::type * = 0 )
```

Get the parameter name for a type that has no special handling.

38.13.1.12 GetPrintableParamName() [2/5]

```
std::string mlpack::bindings::markdown::GetPrintableParamName (
    const util::ParamData & data,
    const typename boost::enable_if< arma::is_arma_type< T >>::type * = 0 )
```

Get the parameter name for a matrix type (where the user has to pass the file that holds the matrix).

38.13.1.13 GetPrintableParamName() [3/5]

```
std::string mlpack::bindings::markdown::GetPrintableParamName (
    const util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

Get the parameter name for a serializable model type (where the user has to pass the file that holds the matrix).

38.13.1.14 GetPrintableParamName() [4/5]

```
std::string mlpack::bindings::markdown::GetPrintableParamName (
    const util::ParamData & data,
    const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo,
    arma::mat >>>::type * = 0 )
```

Get the parameter name for a mapped matrix type (where the user has to pass the file that holds the matrix).

38.13.1.15 GetPrintableParamName() [5/5]

```
void mlpack::bindings::markdown::GetPrintableParamName (
    const util::ParamData & d,
    const void * ,
    void * output )
```

Get the parameter's name as seen by the user.

Definition at line 67 of file `get_printable_param_name.hpp`.

38.13.1.16 GetPrintableParamValue() [1/5]

```
std::string mlpack::bindings::markdown::GetPrintableParamValue (
    const util::ParamData & data,
    const std::string & value,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo,
arma::mat >>>::type * = 0 )
```

Get the parameter name for a type that has no special handling.

38.13.1.17 GetPrintableParamValue() [2/5]

```
std::string mlpack::bindings::markdown::GetPrintableParamValue (
    const util::ParamData & data,
    const std::string & value,
    const typename boost::enable_if< arma::is_arma_type< T >>::type * = 0 )
```

Get the parameter name for a matrix type (where the user has to pass the file that holds the matrix).

38.13.1.18 GetPrintableParamValue() [3/5]

```
std::string mlpack::bindings::markdown::GetPrintableParamValue (
    const util::ParamData & data,
    const std::string & value,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

Get the parameter name for a serializable model type (where the user has to pass the file that holds the matrix).

38.13.1.19 GetPrintableParamValue() [4/5]

```
std::string mlpack::bindings::markdown::GetPrintableParamValue (
    const util::ParamData & data,
    const std::string & value,
    const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo,
arma::mat >>>::type * = 0 )
```

Get the parameter name for a mapped matrix type (where the user has to pass the file that holds the matrix).

38.13.1.20 GetPrintableParamValue() [5/5]

```
void mlpack::bindings::markdown::GetPrintableParamValue (
    const util::ParamData & d,
    const void * input,
    void * output )
```

Get the parameter's name as seen by the user.

Definition at line 71 of file `get_printable_param_value.hpp`.

38.13.1.21 GetPrintableType() [1/2]

```
void mlpack::bindings::markdown::GetPrintableType (
    const util::ParamData & data,
    const void * ,
    void * output )
```

Print the type of a parameter into the output string.

The type printed depends on the current setting of **BindingInfo::Language()** (p. 985).

Definition at line 30 of file `get_printable_type.hpp`.

References **BindingInfo::Language()**.

38.13.1.22 GetPrintableType() [2/2]

```
std::string mlpack::bindings::markdown::GetPrintableType (
    const util::ParamData & data )
```

Print the type of a parameter.

The type printed depends on the current setting of **BindingInfo::Language()** (p. 985).

Definition at line 56 of file `get_printable_type.hpp`.

38.13.1.23 IgnoreCheck()

```
bool mpack::bindings::markdown::IgnoreCheck (
    const T & ) [inline]
```

Return whether or not a runtime check on parameters should be ignored.

We don't ignore any runtime checks for **CLI** (p. 1117) bindings, so this always returns false.

Return whether or not a runtime check on parameters should be ignored.

For test bindings, we do not ignore any checks, so this always returns false.

Definition at line 111 of file print_doc_functions.hpp.

38.13.1.24 IsSerializable() [1/4]

```
bool mpack::bindings::markdown::IsSerializable (
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0 )
```

Return false, because the type is not serializable.

Definition at line 25 of file is_serializable.hpp.

38.13.1.25 IsSerializable() [2/4]

```
bool mpack::bindings::markdown::IsSerializable (
    const typename boost::enable_if< arma::is_arma_type< T >>::type * = 0 )
```

Return false, because even though the type is serializable, it is an Armadillo type not an mpack model.

Definition at line 36 of file is_serializable.hpp.

38.13.1.26 IsSerializable() [3/4]

```
bool mpack::bindings::markdown::IsSerializable (
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0 )
```

Return true, because the type is serializable.

Definition at line 46 of file is_serializable.hpp.

38.13.1.27 IsSerializable() [4/4]

```
void mlpack::bindings::markdown::IsSerializable (
    const    util::ParamData & ,
    const void * ,
    void * output )
```

Return whether or not the type is serializable.

Definition at line 57 of file `is_serializable.hpp`.

38.13.1.28 ParamString()

```
std::string mlpack::bindings::markdown::ParamString (
    const std::string & paramName ) [inline]
```

Print what a user would type to invoke the given option name.

Note that the name *must* exist in the **CLI** (p. 1117) module. (Note that because of the way ProgramInfo is structured, this doesn't mean that all of the `PARAM_*`() declarataions need to come before the **PROGRAM_INFO()** (p. 2733) declaration.)

38.13.1.29 ParamType()

```
std::string mlpack::bindings::markdown::ParamType (
    const    util::ParamData & d ) [inline]
```

Print the user-encountered type of an option.

38.13.1.30 PrintDataset()

```
std::string mlpack::bindings::markdown::PrintDataset (
    const std::string & dataset ) [inline]
```

Print a dataset type parameter (add .csv and return).

38.13.1.31 PrintDefault()

```
std::string mlpack::bindings::markdown::PrintDefault (
    const std::string & paramName ) [inline]
```

Print the default value of an option, unless it is required (in which case Markdown italicized '–' is printed).

38.13.1.32 PrintImport()

```
std::string mlpack::bindings::markdown::PrintImport (
    const std::string & bindingName ) [inline]
```

Print any imports that need to be done before using the binding.

38.13.1.33 PrintLanguage()

```
std::string mlpack::bindings::markdown::PrintLanguage (
    const std::string & language ) [inline]
```

Print the name of the given language.

38.13.1.34 PrintModel()

```
std::string mlpack::bindings::markdown::PrintModel (
    const std::string & model ) [inline]
```

Print a model type parameter (add .bin and return).

38.13.1.35 PrintOutputOptionInfo()

```
std::string mlpack::bindings::markdown::PrintOutputOptionInfo (
    const std::string & language ) [inline]
```

Print any special information about output options.

38.13.1.36 PrintTypeDoc()

```
std::string mlpack::bindings::markdown::PrintTypeDoc (
    const util::ParamData & data )
```

Print the type of a parameter into the output string.

The type printed depends on the current setting of **BindingInfo::Language()** (p. 985).

Definition at line 30 of file `print_type_doc.hpp`.

References **BindingInfo::Language()**.

38.13.1.37 PrintTypeDocs()

```
std::string mlpack::bindings::markdown::PrintTypeDocs ( ) [inline]
```

Print details about the different types for a language.

38.13.1.38 PrintValue()

```
std::string mlpack::bindings::markdown::PrintValue (
    const T & value,
    bool quotes ) [inline]
```

Given a parameter type, print the corresponding value.

38.13.1.39 ProgramCall() [1/2]

```
std::string mlpack::bindings::markdown::ProgramCall (
    const std::string & programName,
    Args... args )
```

Given a program name and arguments for it, print what its invocation would be.

38.13.1.40 ProgramCall() [2/2]

```
std::string mlpack::bindings::markdown::ProgramCall (
    const std::string & programName ) [inline]
```

Given a program name, print a call assuming that all arguments are specified.

38.14 mlpack::bindings::python Namespace Reference

Classes

- class **PyOption**

The Python option class.

Functions

- `template<typename T >`
`void DefaultParam (const util::ParamData &data, const void *, void *output)`
Return the default value of an option.
- `template<typename T >`
`std::string DefaultParamImpl (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< util::IsStdVector< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::string >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< mlpack::data::DatasetInfo, arma::mat >>::type *=0)`
Return the default value of an option.
- `template<typename T >`
`std::string DefaultParamImpl (const util::ParamData &data, const typename boost::enable_if< util::IsStdVector< T >>::type *=0)`
Return the default value of a vector option.
- `template<typename T >`
`std::string DefaultParamImpl (const util::ParamData &data, const typename boost::enable_if< std::is_same< T, std::string >>::type *=0)`
Return the default value of a string option.
- `template<typename T >`
`std::string DefaultParamImpl (const util::ParamData &data, const typename boost::enable_if_c< arma::is_arma_type< T >::value || std::is_same< T, std::tuple< mlpack::data::DatasetInfo, arma::mat >>::value >>::type *=0)`
Return the default value of a matrix option, a tuple option, a serializable option, or a string option (this returns the default filename, or "" if the default is no file).
- `template<typename T >`
`std::string DefaultParamImpl (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)`
Return the default value of a model option (this returns the default filename, or "" if the default is no file).
- `template<typename T >`
`std::string GetArmaType ()`
This is used for arma::Mat<> types; it will return "mat" for matrices, "row" for row vectors, and "col" for column vectors.
- `std::string GetBindingName (const std::string &bindingName)`
Given the name of a binding, print its Python name.
- `template<typename T >`
`std::string GetCythonType (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0)`
- `template<typename T >`
`std::string GetCythonType (const util::ParamData &d, const typename boost::enable_if< util::IsStdVector< T >>::type *=0)`
- `template<typename T >`
`std::string GetCythonType (const util::ParamData &d, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)`
- `template<typename T >`
`std::string GetCythonType (const util::ParamData &d, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)`
- `template<>`
`std::string GetCythonType< bool > (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< bool >>::type *, const typename boost::disable_if< data::HasSerialize< bool >>::type *, const typename boost::disable_if< arma::is_arma_type< bool >>::type *)`

- `template<>`
`std::string GetCythonType< double > (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< double >>::type *, const typename boost::disable_if< data::HasSerialize< double >>::type *, const typename boost::disable_if< arma::is_arma_type< double >>::type *)`
- `template<>`
`std::string GetCythonType< int > (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< int >>::type *, const typename boost::disable_if< data::HasSerialize< int >>::type *, const typename boost::disable_if< arma::is_arma_type< int >>::type *)`
- `template<>`
`std::string GetCythonType< size_t > (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< size_t >>::type *, const typename boost::disable_if< data::HasSerialize< size_t >>::type *, const typename boost::disable_if< arma::is_arma_type< size_t >>::type *)`
- `template<>`
`std::string GetCythonType< std::string > (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< std::string >>::type *, const typename boost::disable_if< data::HasSerialize< std::string >>::type *, const typename boost::disable_if< arma::is_arma_type< std::string >>::type *)`
- `template<typename T >`
`std::string GetNumpyType ()`
- `template<>`
`std::string GetNumpyType< double > ()`
- `template<>`
`std::string GetNumpyType< size_t > ()`
- `template<typename T >`
`std::string GetNumpyTypeChar ()`
- `template<>`
`std::string GetNumpyTypeChar< arma::Col< size_t > > ()`
- `template<>`
`std::string GetNumpyTypeChar< arma::mat > ()`
- `template<>`
`std::string GetNumpyTypeChar< arma::Mat< size_t > > ()`
- `template<>`
`std::string GetNumpyTypeChar< arma::Row< size_t > > ()`
- `template<>`
`std::string GetNumpyTypeChar< arma::rowvec > ()`
- `template<>`
`std::string GetNumpyTypeChar< arma::vec > ()`
- `template<typename T >`
`void GetParam (const util::ParamData &d, const void *, void *output)`
All Python binding types are exactly what is held in the ParamData, so no special handling is necessary.
- `template<typename T >`
`std::string GetPrintableParam (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< util::IsStdVector< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)`
Print an option of a simple type.
- `template<typename T >`
`std::string GetPrintableParam (const util::ParamData &data, const typename boost::enable_if< util::IsStdVector< T >>::type *=0)`
Print a vector option, with spaces between it.
- `template<typename T >`
`std::string GetPrintableParam (const util::ParamData &data, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)`

Print a matrix option (this prints its size).

- `template<typename T >`
`std::string GetPrintableParam (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)`

Print a serializable class option (this prints the class name).

- `template<typename T >`
`std::string GetPrintableParam (const util::ParamData &data, const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)`

Print a combination DatasetInfo/matrix parameter.

- `template<typename T >`
`void GetPrintableParam (const util::ParamData &data, const void *, void *output)`

Print an option into a std::string.

- `template<typename T >`
`std::string GetPrintableType (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)`
- `template<typename T >`
`std::string GetPrintableType (const util::ParamData &d, const typename boost::enable_if< util::IsStdVector< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)`
- `template<typename T >`
`std::string GetPrintableType (const util::ParamData &, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)`
- `template<typename T >`
`std::string GetPrintableType (const util::ParamData &, const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)`
- `template<typename T >`
`std::string GetPrintableType (const util::ParamData &d, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)`
- `template<typename T >`
`void GetPrintableType (const util::ParamData &d, const void *, void *output)`
- `template<>`
`std::string GetPrintableType< bool > (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< bool >>::type *, const typename boost::disable_if< data::HasSerialize< bool >>::type *, const typename boost::disable_if< arma::is_arma_type< bool >>::type *, const typename boost::disable_if< std::is_same< bool, std::tuple< data::DatasetInfo, arma::mat >>>::type *)`
- `template<>`
`std::string GetPrintableType< double > (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< double >>::type *, const typename boost::disable_if< data::HasSerialize< double >>::type *, const typename boost::disable_if< arma::is_arma_type< double >>::type *, const typename boost::disable_if< std::is_same< double, std::tuple< data::DatasetInfo, arma::mat >>>::type *)`
- `template<>`
`std::string GetPrintableType< int > (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< int >>::type *, const typename boost::disable_if< data::HasSerialize< int >>::type *, const typename boost::disable_if< arma::is_arma_type< int >>::type *, const typename boost::disable_if< std::is_same< int, std::tuple< data::DatasetInfo, arma::mat >>>::type *)`
- `template<>`
`std::string GetPrintableType< size_t > (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< size_t >>::type *, const typename boost::disable_if< data::HasSerialize< size_t >>::type *,`

const typename boost::disable_if< arma::is_arma_type< size_t >>::type *, const typename boost::disable_if< std::is_same< size_t, std::tuple< **data::DatasetInfo**, arma::mat >>>::type *)

- template<>

std::string **GetPrintableType**< std::string > (const **util::ParamData** &, const typename boost::disable_if< **util::isStdVector**< std::string >>::type *, const typename boost::disable_if< **data::HasSerialize**< std::string >>::type *, const typename boost::disable_if< arma::is_arma_type< std::string >>::type *, const typename boost::disable_if< std::is_same< std::string, std::tuple< **data::DatasetInfo**, arma::mat >>>::type *)
- bool **IgnoreCheck** (const std::string ¶mName)

Print whether or not we should ignore a check on the given parameter.
- bool **IgnoreCheck** (const std::vector< std::string > &constraints)

Print whether or not we should ignore a check on the given set of constraints.
- bool **IgnoreCheck** (const std::vector< std::pair< std::string, bool >> &constraints, const std::string ¶mName)

Print whether or not we should ignore a check on the given set of constraints.
- template<typename T >

void **ImportDecl** (const **util::ParamData** &d, const size_t indent, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< **data::HasSerialize**< T >>::type *=0)

For a serializable type, print a cppclass definition.
- template<typename T >

void **ImportDecl** (const **util::ParamData** &, const size_t, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< **data::HasSerialize**< T >>::type *=0)

For a non-serializable type, print nothing.
- template<typename T >

void **ImportDecl** (const **util::ParamData** &, const size_t, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)

For a matrix type, print nothing.
- template<typename T >

void **ImportDecl** (const **util::ParamData** &d, const void *indent, void *)

Print the cppclass definition for a serializable model; print nothing for a non-serializable type.
- std::string **ParamString** (const std::string ¶mName)

Given the parameter name, determine what it would actually be when passed to the command line.
- template<typename T >

void **PrintClassDefn** (const **util::ParamData** &, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< **data::HasSerialize**< T >>::type *=0)

Non-serializable models don't require any special definitions, so this prints nothing.
- template<typename T >

void **PrintClassDefn** (const **util::ParamData** &, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)

Matrices don't require any special definitions, so this prints nothing.
- template<typename T >

void **PrintClassDefn** (const **util::ParamData** &d, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< **data::HasSerialize**< T >>::type *=0)

Serializable models require a special class definition.
- template<typename T >

void **PrintClassDefn** (const **util::ParamData** &d, const void *, void *)

Print the class definition to stdout.
- std::string **PrintDataset** (const std::string &datasetName)

Given the name of a matrix, print it.
- std::string **PrintDefault** (const std::string ¶mName)

Given a parameter name, print its corresponding default value.

- `template<typename T >`
`void PrintDefn (const util::ParamData &d, const void *, void *)`
Print the definition for a Python binding parameter to stdout.
- `template<typename T >`
`void PrintDoc (const util::ParamData &d, const void *input, void *)`
Print the docstring documentation for a given parameter.
- `std::string PrintImport (const std::string &bindingName)`
Print any import information for the Python binding.
- `std::string PrintInputOptions ()`
- `template<typename T , typename... Args>`
`std::string PrintInputOptions (const std::string ¶mName, const T &value, Args... args)`
Print an input option.
- `template<typename T >`
`void PrintInputProcessing (const util::ParamData &d, const size_t indent, const typename boost::disable_if<
util::IsStdVector< T >>::type ==0, const typename boost::disable_if< arma::is_arma_type< T >>::type ==0,
const typename boost::disable_if< data::HasSerialize< T >>::type ==0, const typename boost::disable_if<
std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type ==0)`
Print input processing for a standard option type.
- `template<typename T >`
`void PrintInputProcessing (const util::ParamData &d, const size_t indent, const typename boost::disable_if<
arma::is_arma_type< T >>::type ==0, const typename boost::disable_if< data::HasSerialize< T >>::type
==0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type
==0, const typename boost::enable_if< util::IsStdVector< T >>::type ==0)`
Print input processing for a vector type.
- `template<typename T >`
`void PrintInputProcessing (const util::ParamData &d, const size_t indent, const typename boost::disable_if<
util::IsStdVector< T >>::type ==0, const typename boost::enable_if< arma::is_arma_type< T >>::type ==0)`
Print input processing for a matrix type.
- `template<typename T >`
`void PrintInputProcessing (const util::ParamData &d, const size_t indent, const typename boost::disable_if<
util::IsStdVector< T >>::type ==0, const typename boost::disable_if< arma::is_arma_type< T >>::type ==0,
const typename boost::enable_if< data::HasSerialize< T >>::type ==0)`
Print input processing for a serializable type.
- `template<typename T >`
`void PrintInputProcessing (const util::ParamData &d, const size_t indent, const typename boost::disable_if<
util::IsStdVector< T >>::type ==0, const typename boost::enable_if< std::is_same< T, std::tuple< data::←
DatasetInfo, arma::mat >>>::type ==0)`
Print input processing for a matrix/DatasetInfo type.
- `template<typename T >`
`void PrintInputProcessing (const util::ParamData &d, const void *input, void *)`
Given parameter information and the current number of spaces for indentation, print the code to process the input to cout.
- `std::string PrintModel (const std::string &modelName)`
Given the name of a model, print it.
- `std::string PrintOutputOptionInfo ()`
Print any special information about output options.
- `std::string PrintOutputOptions ()`
- `template<typename T , typename... Args>`
`std::string PrintOutputOptions (const std::string ¶mName, const T &value, Args... args)`

- `template<typename T >`
`void PrintOutputProcessing (const util::ParamData &d, const size_t indent, const bool onlyOutput, const`
`typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< data::`
`::HasSerialize< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::`
`DatasetInfo, arma::mat >>>::type *=0)`
Print output processing for a regular parameter type.
- `template<typename T >`
`void PrintOutputProcessing (const util::ParamData &d, const size_t indent, const bool onlyOutput, const`
`typename boost::enable_if< arma::is_arma_type< T >>::type *=0)`
Print output processing for a matrix type.
- `template<typename T >`
`void PrintOutputProcessing (const util::ParamData &d, const size_t indent, const bool onlyOutput, const`
`typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)`
Print output processing for a dataset info / matrix combination.
- `template<typename T >`
`void PrintOutputProcessing (const util::ParamData &d, const size_t indent, const bool onlyOutput, const`
`typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::`
`::HasSerialize< T >>::type *=0)`
Print output processing for a serializable model.
- `template<typename T >`
`void PrintOutputProcessing (const util::ParamData &d, const void *input, void *)`
Given parameter information and the current number of spaces for indentation, print the code to process the output to
cout.
- `void PrintPYX (const util::ProgramDoc &programInfo, const std::string &mainFilename, const std::string`
`&functionName)`
Given a list of parameter definition and program documentation, print a generated .pyx file to stdout.
- `template<typename T >`
`std::string PrintTypeDoc (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_`
`type< T >>::type *=0, const typename boost::disable_if< util::IsStdVector< T >>::type *=0, const typename`
`boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< std::is_same< T,`
`std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)`
Return a string representing the command-line type of an option.
- `template<typename T >`
`std::string PrintTypeDoc (const util::ParamData &data, const typename std::enable_if< util::IsStdVector< T`
`>::value >::type *=0)`
Return a string representing the command-line type of a vector.
- `template<typename T >`
`std::string PrintTypeDoc (const util::ParamData &data, const typename std::enable_if< arma::is_arma_type<`
`T >::value >::type *=0)`
Return a string representing the command-line type of a matrix option.
- `template<typename T >`
`std::string PrintTypeDoc (const util::ParamData &data, const typename std::enable_if< std::is_same< T, std::`
`tuple< data::DatasetInfo, arma::mat >>>::value >::type *=0)`
Return a string representing the command-line type of a matrix tuple option.
- `template<typename T >`
`std::string PrintTypeDoc (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_`
`type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)`
Return a string representing the command-line type of a model.
- `template<typename T >`
`void PrintTypeDoc (const util::ParamData &data, const void *, void *output)`
Print the command-line type of an option into a string.

- `template<typename T >`
`std::string PrintValue (const T &value, bool quotes)`
Given a parameter type, print the corresponding value.
- `template<>`
`std::string PrintValue (const bool &value, bool quotes)`
- `template<typename... Args>`
`std::string ProgramCall (const std::string & programName, Args... args)`
Given a name of a binding and a variable number of arguments (and their contents), print the corresponding function call.
- `std::string ProgramCall (const std::string & programName)`
Given the name of a binding, print a program call assuming that all options are specified.
- `template<typename T >`
`void SerializeIn (T *t, const std::string &str, const std::string &name)`
- `template<typename T >`
`std::string SerializeOut (T *t, const std::string &name)`
- `void StripType (const std::string &inputType, std::string &strippedType, std::string &printedType, std::string &defaultsType)`
Given an input type like, e.g., "LogisticRegression<>", return three types that can be used in Python code.

Variables

- `std::string programName`

38.14.1 Function Documentation

38.14.1.1 DefaultParam()

```
void mlpack::bindings::python::DefaultParam (
    const util::ParamData & data,
    const void * ,
    void * output )
```

Return the default value of an option.

This is the function that will be placed into the **CLI** (p. 1117) `functionMap`.

Definition at line 80 of file `default_param.hpp`.

38.14.1.2 DefaultParamImpl() [1/5]

```
std::string mlpack::bindings::python::DefaultParamImpl (
    const    util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type *   = 0,
    const typename boost::disable_if<    util::IsStdVector< T >>::type *   = 0,
    const typename boost::disable_if<    data::HasSerialize< T >>::type *   = 0,
    const typename boost::disable_if< std::is_same< T, std::string >>::type *   = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple<    mlpack::data::↵
DatasetInfo, arma::mat >>>::type *   = 0 )
```

Return the default value of an option.

This is for regular types.

38.14.1.3 DefaultParamImpl() [2/5]

```
std::string mlpack::bindings::python::DefaultParamImpl (
    const    util::ParamData & data,
    const typename boost::enable_if<    util::IsStdVector< T >>::type *   = 0 )
```

Return the default value of a vector option.

38.14.1.4 DefaultParamImpl() [3/5]

```
std::string mlpack::bindings::python::DefaultParamImpl (
    const    util::ParamData & data,
    const typename boost::enable_if< std::is_same< T, std::string >>::type *   = 0 )
```

Return the default value of a string option.

38.14.1.5 DefaultParamImpl() [4/5]

```
std::string mlpack::bindings::python::DefaultParamImpl (
    const    util::ParamData & data,
    const typename boost::enable_if_c< arma::is_arma_type< T >::value||std::is_same< T,
std::tuple<    mlpack::data::DatasetInfo, arma::mat >>::value >::type *   = 0 )
```

Return the default value of a matrix option, a tuple option, a serializable option, or a string option (this returns the default filename, or " if the default is no file).

38.14.1.6 DefaultParamImpl() [5/5]

```
std::string mlpack::bindings::python::DefaultParamImpl (
    const    util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if<    data::HasSerialize< T >>::type * = 0 )
```

Return the default value of a model option (this returns the default filename, or "" if the default is no file).

38.14.1.7 GetArmaType()

```
std::string mlpack::bindings::python::GetArmaType ( ) [inline]
```

This is used for arma::Mat<> types; it will return "mat" for matrices, "row" for row vectors, and "col" for column vectors.

Definition at line 28 of file get_arma_type.hpp.

38.14.1.8 GetBindingName()

```
std::string mlpack::bindings::python::GetBindingName (
    const std::string & bindingName ) [inline]
```

Given the name of a binding, print its Python name.

38.14.1.9 GetCythonType() [1/4]

```
std::string mlpack::bindings::python::GetCythonType (
    const    util::ParamData & ,
    const typename boost::disable_if<    util::IsStdVector< T >>::type * = 0,
    const typename boost::disable_if<    data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0 ) [inline]
```

Definition at line 24 of file get_cython_type.hpp.

38.14.1.10 GetCythonType() [2/4]

```
std::string mlpack::bindings::python::GetCythonType (
    const    util::ParamData & d,
    const typename boost::enable_if<    util::IsStdVector< T >>::type * = 0 ) [inline]
```

Definition at line 84 of file get_cython_type.hpp.

38.14.1.11 GetCythonType() [3/4]

```
std::string mlpack::bindings::python::GetCythonType (
    const    util::ParamData & d,
    const typename boost::enable_if< arma::is_arma_type< T >>::type *   = 0 ) [inline]
```

Definition at line 92 of file `get_cython_type.hpp`.

38.14.1.12 GetCythonType() [4/4]

```
std::string mlpack::bindings::python::GetCythonType (
    const    util::ParamData & d,
    const typename boost::disable_if< arma::is_arma_type< T >>::type *   = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type *   = 0 ) [inline]
```

Definition at line 106 of file `get_cython_type.hpp`.

References `ParamData::cppType`.

38.14.1.13 GetCythonType< bool >()

```
std::string mlpack::bindings::python::GetCythonType< bool > (
    const    util::ParamData & ,
    const typename boost::disable_if< util::IsStdVector< bool >>::type * ,
    const typename boost::disable_if< data::HasSerialize< bool >>::type * ,
    const typename boost::disable_if< arma::is_arma_type< bool >>::type *   ) [inline]
```

Definition at line 74 of file `get_cython_type.hpp`.

38.14.1.14 GetCythonType< double >()

```
std::string mlpack::bindings::python::GetCythonType< double > (
    const    util::ParamData & ,
    const typename boost::disable_if< util::IsStdVector< double >>::type * ,
    const typename boost::disable_if< data::HasSerialize< double >>::type * ,
    const typename boost::disable_if< arma::is_arma_type< double >>::type *   ) [inline]
```

Definition at line 44 of file `get_cython_type.hpp`.

38.14.1.15 GetCythonType< int >()

```
std::string mlpack::bindings::python::GetCythonType< int > (
    const    util::ParamData & ,
    const typename boost::disable_if<    util::IsStdVector< int >>::type * ,
    const typename boost::disable_if<    data::HasSerialize< int >>::type * ,
    const typename boost::disable_if< arma::is_arma_type< int >>::type * ) [inline]
```

Definition at line 34 of file get_cython_type.hpp.

38.14.1.16 GetCythonType< size_t >()

```
std::string mlpack::bindings::python::GetCythonType< size_t > (
    const    util::ParamData & ,
    const typename boost::disable_if<    util::IsStdVector< size_t >>::type * ,
    const typename boost::disable_if<    data::HasSerialize< size_t >>::type * ,
    const typename boost::disable_if< arma::is_arma_type< size_t >>::type * ) [inline]
```

Definition at line 64 of file get_cython_type.hpp.

38.14.1.17 GetCythonType< std::string >()

```
std::string mlpack::bindings::python::GetCythonType< std::string > (
    const    util::ParamData & ,
    const typename boost::disable_if<    util::IsStdVector< std::string >>::type * ,
    const typename boost::disable_if<    data::HasSerialize< std::string >>::type * ,
    const typename boost::disable_if< arma::is_arma_type< std::string >>::type * )
[inline]
```

Definition at line 54 of file get_cython_type.hpp.

38.14.1.18 GetNumpyType()

```
std::string mlpack::bindings::python::GetNumpyType ( ) [inline]
```

Definition at line 22 of file get_numpy_type.hpp.

38.14.1.19 GetNumpyType< double >()

```
std::string  mlpack::bindings::python::GetNumpyType< double > ( )  [inline]
```

Definition at line 28 of file get_numpy_type.hpp.

38.14.1.20 GetNumpyType< size_t >()

```
std::string  mlpack::bindings::python::GetNumpyType< size_t > ( )  [inline]
```

Definition at line 34 of file get_numpy_type.hpp.

38.14.1.21 GetNumpyTypeChar()

```
std::string  mlpack::bindings::python::GetNumpyTypeChar ( )  [inline]
```

Definition at line 23 of file get_numpy_type_char.hpp.

38.14.1.22 GetNumpyTypeChar< arma::Col< size_t > >()

```
std::string  mlpack::bindings::python::GetNumpyTypeChar< arma::Col< size_t > > ( )  [inline]
```

Definition at line 36 of file get_numpy_type_char.hpp.

38.14.1.23 GetNumpyTypeChar< arma::mat >()

```
std::string  mlpack::bindings::python::GetNumpyTypeChar< arma::mat > ( )  [inline]
```

Definition at line 49 of file get_numpy_type_char.hpp.

38.14.1.24 GetNumpyTypeChar< arma::Mat< size_t > >()

```
std::string  mlpack::bindings::python::GetNumpyTypeChar< arma::Mat< size_t > > ( )  [inline]
```

Definition at line 30 of file get_numpy_type_char.hpp.

38.14.1.25 GetNumpyTypeChar< arma::Row< size_t > >()

```
std::string mlpack::bindings::python::GetNumpyTypeChar< arma::Row< size_t > > ( ) [inline]
```

Definition at line 42 of file get_numpy_type_char.hpp.

38.14.1.26 GetNumpyTypeChar< arma::rowvec >()

```
std::string mlpack::bindings::python::GetNumpyTypeChar< arma::rowvec > ( ) [inline]
```

Definition at line 61 of file get_numpy_type_char.hpp.

38.14.1.27 GetNumpyTypeChar< arma::vec >()

```
std::string mlpack::bindings::python::GetNumpyTypeChar< arma::vec > ( ) [inline]
```

Definition at line 55 of file get_numpy_type_char.hpp.

38.14.1.28 GetParam()

```
void mlpack::bindings::python::GetParam (
    const util::ParamData & d,
    const void * ,
    void * output )
```

All Python binding types are exactly what is held in the ParamData, so no special handling is necessary.

Definition at line 26 of file get_param.hpp.

References ParamData::value.

38.14.1.29 GetPrintableParam() [1/6]

```
std::string mlpack::bindings::python::GetPrintableParam (
    const    util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type *   = 0,
    const typename boost::disable_if<    util::IsStdVector< T >>::type *   = 0,
    const typename boost::disable_if<    data::HasSerialize< T >>::type *   = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple<    data::DatasetInfo,
arma::mat >>>::type *   = 0 )
```

Print an option of a simple type.

Print an option.

Definition at line 26 of file `get_printable_param.hpp`.

References `ParamData::value`.

38.14.1.30 GetPrintableParam() [2/6]

```
std::string mlpack::bindings::python::GetPrintableParam (
    const    util::ParamData & data,
    const typename boost::enable_if<    util::IsStdVector< T >>::type *   = 0 )
```

Print a vector option, with spaces between it.

Definition at line 43 of file `get_printable_param.hpp`.

References `ParamData::value`.

38.14.1.31 GetPrintableParam() [3/6]

```
std::string mlpack::bindings::python::GetPrintableParam (
    const    util::ParamData & data,
    const typename boost::enable_if< arma::is_arma_type< T >>::type *   = 0 )
```

Print a matrix option (this prints its size).

Print a matrix option (this just prints the filename).

Definition at line 59 of file `get_printable_param.hpp`.

References `ParamData::value`.

38.14.1.32 GetPrintableParam() [4/6]

```
std::string mlpack::bindings::python::GetPrintableParam (
    const    util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type *   = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type *   = 0 )
```

Print a serializable class option (this prints the class name).

Print a serializable class option (this just prints the filename).

Print a model option (this just prints the filename).

Definition at line 75 of file get_printable_param.hpp.

References ParamData::cppType, and ParamData::value.

38.14.1.33 GetPrintableParam() [5/6]

```
std::string mlpack::bindings::python::GetPrintableParam (
    const    util::ParamData & data,
    const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo,
arma::mat >>>::type *   = 0 )
```

Print a combination DatasetInfo/matrix parameter.

Print a mapped matrix option (this just prints the filename).

Definition at line 89 of file get_printable_param.hpp.

References ParamData::value.

38.14.1.34 GetPrintableParam() [6/6]

```
void mlpack::bindings::python::GetPrintableParam (
    const    util::ParamData & data,
    const void * ,
    void * output )
```

Print an option into a std::string.

This should print a short, one-line representation of the object. The string will be stored in the output pointer.

Parameters

<i>data</i>	Parameter data struct.
<i>input</i>	Unused parameter.
<i>output</i>	Output storage for the string.

Definition at line 114 of file `get_printable_param.hpp`.

38.14.1.35 `GetPrintableType()` [1/6]

```
std::string mlpack::bindings::python::GetPrintableType (
    const util::ParamData & ,
    const typename boost::disable_if< util::IsStdVector< T >>::type * = 0,
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo,
arma::mat >>>::type * = 0 ) [inline]
```

38.14.1.36 `GetPrintableType()` [2/6]

```
std::string mlpack::bindings::python::GetPrintableType (
    const util::ParamData & d,
    const typename boost::enable_if< util::IsStdVector< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo,
arma::mat >>>::type * = 0 ) [inline]
```

38.14.1.37 `GetPrintableType()` [3/6]

```
std::string mlpack::bindings::python::GetPrintableType (
    const util::ParamData & ,
    const typename boost::enable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo,
arma::mat >>>::type * = 0 ) [inline]
```

38.14.1.38 `GetPrintableType()` [4/6]

```
std::string mlpack::bindings::python::GetPrintableType (
    const util::ParamData & ,
    const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo,
arma::mat >>>::type * = 0 ) [inline]
```

38.14.1.39 GetPrintableType() [5/6]

```
std::string mlpack::bindings::python::GetPrintableType (
    const    util::ParamData & d,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if<    data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple<    data::DatasetInfo,
arma::mat >>>::type * = 0 ) [inline]
```

38.14.1.40 GetPrintableType() [6/6]

```
void mlpack::bindings::python::GetPrintableType (
    const    util::ParamData & d,
    const void * ,
    void * output )
```

Definition at line 106 of file get_printable_type.hpp.

38.14.1.41 GetPrintableType< bool >()

```
std::string mlpack::bindings::python::GetPrintableType< bool > (
    const    util::ParamData & ,
    const typename boost::disable_if<    util::IsStdVector< bool >>::type * ,
    const typename boost::disable_if<    data::HasSerialize< bool >>::type * ,
    const typename boost::disable_if< arma::is_arma_type< bool >>::type * ,
    const typename boost::disable_if< std::is_same< bool, std::tuple<    data::Dataset←
Info, arma::mat >>>::type * ) [inline]
```

38.14.1.42 GetPrintableType< double >()

```
std::string mlpack::bindings::python::GetPrintableType< double > (
    const    util::ParamData & ,
    const typename boost::disable_if<    util::IsStdVector< double >>::type * ,
    const typename boost::disable_if<    data::HasSerialize< double >>::type * ,
    const typename boost::disable_if< arma::is_arma_type< double >>::type * ,
    const typename boost::disable_if< std::is_same< double, std::tuple<    data::Dataset←
Info, arma::mat >>>::type * ) [inline]
```

38.14.1.43 `GetPrintableType< int >()`

```
std::string mlpack::bindings::python::GetPrintableType< int > (
    const util::ParamData & ,
    const typename boost::disable_if< util::IsStdVector< int >>::type * ,
    const typename boost::disable_if< data::HasSerialize< int >>::type * ,
    const typename boost::disable_if< arma::is_arma_type< int >>::type * ,
    const typename boost::disable_if< std::is_same< int, std::tuple< data::DatasetInfo,
arma::mat >>>::type * ) [inline]
```

38.14.1.44 `GetPrintableType< size_t >()`

```
std::string mlpack::bindings::python::GetPrintableType< size_t > (
    const util::ParamData & ,
    const typename boost::disable_if< util::IsStdVector< size_t >>::type * ,
    const typename boost::disable_if< data::HasSerialize< size_t >>::type * ,
    const typename boost::disable_if< arma::is_arma_type< size_t >>::type * ,
    const typename boost::disable_if< std::is_same< size_t, std::tuple< data::Dataset←
Info, arma::mat >>>::type * ) [inline]
```

38.14.1.45 `GetPrintableType< std::string >()`

```
std::string mlpack::bindings::python::GetPrintableType< std::string > (
    const util::ParamData & ,
    const typename boost::disable_if< util::IsStdVector< std::string >>::type * ,
    const typename boost::disable_if< data::HasSerialize< std::string >>::type * ,
    const typename boost::disable_if< arma::is_arma_type< std::string >>::type * ,
    const typename boost::disable_if< std::is_same< std::string, std::tuple< data::←
DatasetInfo, arma::mat >>>::type * ) [inline]
```

38.14.1.46 `IgnoreCheck()` [1/3]

```
bool mlpack::bindings::python::IgnoreCheck (
    const std::string & paramName ) [inline]
```

Print whether or not we should ignore a check on the given parameter.

For Python bindings, we ignore any checks on output parameters, so if paramName is an output parameter, this returns true.

38.14.1.47 IgnoreCheck() [2/3]

```
bool mlpack::bindings::python::IgnoreCheck (
    const std::vector< std::string > & constraints ) [inline]
```

Print whether or not we should ignore a check on the given set of constraints.

For Python bindings, we ignore any checks on output parameters, so if any parameter is an output parameter, this returns true.

38.14.1.48 IgnoreCheck() [3/3]

```
bool mlpack::bindings::python::IgnoreCheck (
    const std::vector< std::pair< std::string, bool >> & constraints,
    const std::string & paramName ) [inline]
```

Print whether or not we should ignore a check on the given set of constraints.

For Python bindings, we ignore any checks on output parameters, so if any constraint parameter or the main parameter are output parameters, this returns true.

38.14.1.49 ImportDecl() [1/4]

```
void mlpack::bindings::python::ImportDecl (
    const util::ParamData & d,
    const size_t indent,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

For a serializable type, print a cppclass definition.

This will give output of the form:

```
cdef cppclass Type: Type() nogil
```

Definition at line 26 of file import_decl.hpp.

References ParamData::cppType, and StripType().

38.14.1.50 ImportDecl() [2/4]

```
void mlpack::bindings::python::ImportDecl (
    const util::ParamData & ,
    const size_t ,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0 )
```

For a non-serializable type, print nothing.

Definition at line 53 of file import_decl.hpp.

38.14.1.51 ImportDecl() [3/4]

```
void mlpack::bindings::python::ImportDecl (
    const    util::ParamData & ,
    const size_t ,
    const typename boost::enable_if< arma::is_arma_type< T >>::type *   = 0 )
```

For a matrix type, print nothing.

Definition at line 66 of file `import_decl.hpp`.

38.14.1.52 ImportDecl() [4/4]

```
void mlpack::bindings::python::ImportDecl (
    const    util::ParamData & d,
    const void * indent,
    void * )
```

Print the cppclass definition for a serializable model; print nothing for a non-serializable type.

Parameters

<i>d</i>	Parameter info struct.
<i>input</i>	Pointer to <code>size_t</code> indicating indent.
<i>output</i>	Unused parameter.

Definition at line 83 of file `import_decl.hpp`.

38.14.1.53 ParamString()

```
std::string mlpack::bindings::python::ParamString (
    const std::string & paramName ) [inline]
```

Given the parameter name, determine what it would actually be when passed to the command line.

38.14.1.54 PrintClassDefn() [1/4]

```
void mlpack::bindings::python::PrintClassDefn (
    const    util::ParamData & ,
    const typename boost::disable_if< arma::is_arma_type< T >>::type *   = 0,
    const typename boost::disable_if<    data::HasSerialize< T >>::type *   = 0 )
```

Non-serializable models don't require any special definitions, so this prints nothing.

Definition at line 26 of file `print_class_defn.hpp`.

38.14.1.55 PrintClassDefn() [2/4]

```
void mlpack::bindings::python::PrintClassDefn (
    const    util::ParamData & ,
    const typename boost::enable_if< arma::is_arma_type< T >>::type *   = 0 )
```

Matrices don't require any special definitions, so this prints nothing.

Definition at line 38 of file `print_class_defn.hpp`.

38.14.1.56 PrintClassDefn() [3/4]

```
void mlpack::bindings::python::PrintClassDefn (
    const    util::ParamData & d,
    const typename boost::disable_if< arma::is_arma_type< T >>::type *   = 0,
    const typename boost::enable_if<    data::HasSerialize< T >>::type *   = 0 )
```

Serializable models require a special class definition.

This will produce code like:

```
cdef class <ModelType>Type: cdef <ModelType>* modelptr

def cinit(self): self.modelptr = new <ModelType>()

def dealloc(self): del self.modelptr

def getstate(self): return SerializeOut(self.modelptr, "<ModelType>")

def setstate(self, state): SerializeIn(self.modelptr, state, "<ModelType>")

def reduce_ex(self): return (self.__class__, (), self.__getstate__())
```

Definition at line 49 of file `print_class_defn.hpp`.

References `ParamData::cppType`, and `StripType()`.

38.14.1.57 PrintClassDefn() [4/4]

```
void mlpack::bindings::python::PrintClassDefn (
    const    util::ParamData & d,
    const void * ,
    void *   )
```

Print the class definition to stdout.

Only serializable models require a different class definition, so anything else does nothing.

Parameters

<i>d</i>	Parameter data.
<i>input</i>	Unused parameter.
<i>output</i>	Unused parameter.

Definition at line 112 of file `print_class_defn.hpp`.

38.14.1.58 `PrintDataset()`

```
std::string mlpack::bindings::python::PrintDataset (
    const std::string & datasetName ) [inline]
```

Given the name of a matrix, print it.

Here we do not need to modify anything.

38.14.1.59 `PrintDefault()`

```
std::string mlpack::bindings::python::PrintDefault (
    const std::string & paramName ) [inline]
```

Given a parameter name, print its corresponding default value.

38.14.1.60 `PrintDefn()`

```
void mlpack::bindings::python::PrintDefn (
    const util::ParamData & d,
    const void * ,
    void * )
```

Print the definition for a Python binding parameter to stdout.

This is the definition in the function declaration.

Definition at line 26 of file `print_defn.hpp`.

References `ParamData::name`, and `ParamData::required`.

38.14.1.61 `PrintDoc()`

```
void mlpack::bindings::python::PrintDoc (
    const util::ParamData & d,
    const void * input,
    void * )
```

Print the docstring documentation for a given parameter.

You are responsible for setting up the line—this does not handle indentation or anything. This is meant to produce a line of documentation describing a single parameter.

The indent parameter (`void* input`, which should be a pointer to a `size_t`) should be passed to know how much to indent for a new line.

Parameters

<i>d</i>	Parameter data struct.
<i>input</i>	Pointer to size_t containing indent.
<i>output</i>	Unused parameter.

Definition at line 36 of file print_doc.hpp.

References ParamData::cppType, ParamData::desc, mlpack::util::HyphenateString(), ParamData::name, and ParamData::required.

38.14.1.62 PrintImport()

```
std::string mlpack::bindings::python::PrintImport (
    const std::string & bindingName ) [inline]
```

Print any import information for the Python binding.

38.14.1.63 PrintInputOptions() [1/2]

```
std::string mlpack::bindings::python::PrintInputOptions ( ) [inline]
```

38.14.1.64 PrintInputOptions() [2/2]

```
std::string mlpack::bindings::python::PrintInputOptions (
    const std::string & paramName,
    const T & value,
    Args... args )
```

Print an input option.

This will throw an exception if the parameter does not exist in **CLI** (p. 1117). For a parameter 'x' with value '5', this will print something like x=5.

38.14.1.65 PrintInputProcessing() [1/6]

```
void mlpack::bindings::python::PrintInputProcessing (
    const util::ParamData & d,
    const size_t indent,
    const typename boost::disable_if< util::IsStdVector< T >>::type * = 0,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo,
arma::mat >>>::type * = 0 )
```

Print input processing for a standard option type.

This gives us code like:

Detect if the parameter was passed; set if so.

```
if param_name is not None: if isinstance(param_name, int): SetParam[int](<const string>=""> 'param_name', param_name)
else: raise TypeError("'param_name' must have type 'list'")
else: raise TypeError("'param_name' must have type 'list'")
```

Definition at line 31 of file print_input_processing.hpp.

References ParamData::name, and ParamData::required.

38.14.1.66 PrintInputProcessing() [2/6]

```
void mlpack::bindings::python::PrintInputProcessing (
    const util::ParamData & d,
    const size_t indent,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo,
arma::mat >>>::type * = 0,
    const typename boost::enable_if< util::IsStdVector< T >>::type * = 0 )
```

Print input processing for a vector type.

```
This gives us code like: if param_name is not None: if isinstance(param_name, list): if len(param_name) > 0: if
isinstance(param_name[0], str): SetParam[vector[string]](<const string>=""> 'param_name', param_name)
else: raise TypeError("'param_name' must have type 'list of str'")
else: raise TypeError("'param_name' must have type 'list'")
```

Definition at line 164 of file print_input_processing.hpp.

References ParamData::name, and ParamData::required.

38.14.1.67 PrintInputProcessing() [3/6]

```
void mlpack::bindings::python::PrintInputProcessing (
    const    util::ParamData & d,
    const size_t indent,
    const typename boost::disable_if<    util::IsStdVector< T >>::type *    = 0,
    const typename boost::enable_if< arma::is_arma_type< T >>::type *    = 0 )
```

Print input processing for a matrix type.

This gives us code like:

Detect if the parameter was passed; set if so.

```
if param_name is not None: param_name_tuple = to_matrix(param_name) if param_name_tuple[0].shape[0] == 1 or
param_name_tuple[0].shape[1] == 1: param_name_tuple[0].shape = (param_name_tuple[0].size,) param_name_mat
= arma_numpy.numpy_to_mat_s(param_name_tuple[0], param_name_tuple[1]) SetParam[mat](<const string>="">
'param_name', dereference(param_name_mat)) CLI.SetPassed (p. 1125)(<const string>=""> 'param_name')
```

Definition at line 251 of file print_input_processing.hpp.

References ParamData::name, and ParamData::required.

38.14.1.68 PrintInputProcessing() [4/6]

```
void mlpack::bindings::python::PrintInputProcessing (
    const    util::ParamData & d,
    const size_t indent,
    const typename boost::disable_if<    util::IsStdVector< T >>::type *    = 0,
    const typename boost::disable_if< arma::is_arma_type< T >>::type *    = 0,
    const typename boost::enable_if<    data::HasSerialize< T >>::type *    = 0 )
```

Print input processing for a serializable type.

This gives us code like:

Detect if the parameter was passed; set if so.

```
if param_name is not None: try: SetParamPtr[Model]('param_name', (<ModelType?> param_name).modelptr, CLI.
HasParam (p. 1123)('copy_all_inputs')) except TypeError as e: if type(param_name).__name__ == "ModelType": Set
ParamPtr[Model]('param_name', (<ModelType> param_name).modelptr, CLI.HasParam (p. 1123)('copy_all_inputs'))
else: raise e CLI.SetPassed (p. 1125)(<const string>=""> 'param_name')
```

Definition at line 372 of file print_input_processing.hpp.

References ParamData::cppType, ParamData::name, ParamData::required, and StripType().

38.14.1.69 PrintInputProcessing() [5/6]

```
void mlpack::bindings::python::PrintInputProcessing (
    const util::ParamData & d,
    const size_t indent,
    const typename boost::disable_if< util::IsStdVector< T >>::type * = 0,
    const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo,
arma::mat >>>::type * = 0 )
```

Print input processing for a matrix/DatasetInfo type.

We want to generate code like the following:

```
if param_name is not None: param_name_tuple = to_matrix_with_info(param_name) if len(param_name_←
tuple[0].shape) < 2: param_name_tuple[0].shape = (param_name_tuple[0].size,) param_name_mat = arma_←
_numpy.numpy_to_matrix_d(param_name_tuple[0]) SetParamWithInfo[mat](<const string>=""> 'param_name',
dereference(param_name_mat), &param_name_tuple[1][0]) CLI.SetPassed (p. 1125)(<const string>=""> 'param_←
name')
```

Definition at line 445 of file print_input_processing.hpp.

References ParamData::name, and ParamData::required.

38.14.1.70 PrintInputProcessing() [6/6]

```
void mlpack::bindings::python::PrintInputProcessing (
    const util::ParamData & d,
    const void * input,
    void * )
```

Given parameter information and the current number of spaces for indentation, print the code to process the input to cout.

This code assumes that data.input is true, and should not be called when data.input is false.

The number of spaces to indent should be passed through the input pointer.

Parameters

<i>d</i>	Parameter data struct.
<i>input</i>	Pointer to size_t holding the indentation.
<i>output</i>	Unused parameter.

Definition at line 525 of file print_input_processing.hpp.

38.14.1.71 PrintModel()

```
std::string mlpack::bindings::python::PrintModel (
    const std::string & modelName ) [inline]
```

Given the name of a model, print it.

Here we do not need to modify anything.

38.14.1.72 PrintOutputOptionInfo()

```
std::string mlpack::bindings::python::PrintOutputOptionInfo ( ) [inline]
```

Print any special information about output options.

38.14.1.73 PrintOutputOptions() [1/2]

```
std::string mlpack::bindings::python::PrintOutputOptions ( ) [inline]
```

38.14.1.74 PrintOutputOptions() [2/2]

```
std::string mlpack::bindings::python::PrintOutputOptions (
    const std::string & paramName,
    const T & value,
    Args... args )
```

38.14.1.75 PrintOutputProcessing() [1/5]

```
void mlpack::bindings::python::PrintOutputProcessing (
    const util::ParamData & d,
    const size_t indent,
    const bool onlyOutput,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo,
    arma::mat >>>::type * = 0 )
```

Print output processing for a regular parameter type.

This gives us code like:

```
result = CLI.GetParam (p.1122)[int]('param_name')
```

This gives us code like:

```
result['param_name'] = CLI.GetParam (p.1122)[int]('param_name')
```

Definition at line 29 of file print_output_processing.hpp.

References ParamData::name.

38.14.1.76 PrintOutputProcessing() [2/5]

```
void mlpack::bindings::python::PrintOutputProcessing (
    const util::ParamData & d,
    const size_t indent,
    const bool onlyOutput,
    const typename boost::enable_if< arma::is_arma_type< T >>::type * = 0 )
```

Print output processing for a matrix type.

This gives us code like:

```
result = arma_numpy.mat_to_numpy_X(CLI.GetParam (p.1122)[mat]("name"))
```

where X indicates the type to convert to.

This gives us code like:

```
result['param_name'] = arma_numpy.mat_to_numpy_X(CLI.GetParam (p.1122)[mat]('name'))
```

where X indicates the type to convert to.

Definition at line 85 of file `print_output_processing.hpp`.

References `ParamData::name`.

38.14.1.77 PrintOutputProcessing() [3/5]

```
void mlpack::bindings::python::PrintOutputProcessing (
    const util::ParamData & d,
    const size_t indent,
    const bool onlyOutput,
    const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo,
    arma::mat >>>::type * = 0 )
```

Print output processing for a dataset info / matrix combination.

This gives us code like:

```
result = arma_numpy.mat_to_numpy_X(GetParamWithInfo[mat]('name'))
```

This gives us code like:

```
result['param_name'] = arma_numpy.mat_to_numpy_X(GetParamWithInfo[mat]('name'))
```

Definition at line 127 of file `print_output_processing.hpp`.

References `ParamData::name`.

38.14.1.78 PrintOutputProcessing() [4/5]

```
void mlpack::bindings::python::PrintOutputProcessing (
    const    util::ParamData & d,
    const size_t indent,
    const bool onlyOutput,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

Print output processing for a serializable model.

This gives us code like:

```
result = ModelType() (<ModelType?> result).modelptr = GetParamPtr[Model]('name')
```

But we also have to check to ensure there aren't any input model parameters of the same type that could have the same model pointer. So we need to loop through all input parameters that have the same type, and double-check.

This gives us code like:

```
result['name'] = ModelType() (<ModelType?> result['name']).modelptr = GetParamPtr[Model]('name'))
```

But we also have to check to ensure there aren't any input model parameters of the same type that could have the same model pointer. So we need to loop through all input parameters that have the same type, and double-check.

Definition at line 169 of file print_output_processing.hpp.

References ParamData::cppType, ParamData::input, ParamData::name, CLI::Parameters(), ParamData::required, and StripType().

38.14.1.79 PrintOutputProcessing() [5/5]

```
void mlpack::bindings::python::PrintOutputProcessing (
    const    util::ParamData & d,
    const void * input,
    void * )
```

Given parameter information and the current number of spaces for indentation, print the code to process the output to cout.

This code assumes that data.input is false, and should not be called when data.input is true. If this is the only output, the results will be different.

The input pointer should be a pointer to a std::tuple<size_t, bool> where the first element is the indentation and the second element is a boolean representing whether or not this is the only output parameter.

Parameters

<i>d</i>	Parameter data struct.
<i>input</i>	Pointer to size_t holding the indentation.
<i>output</i>	Unused parameter.

Definition at line 298 of file `print_output_processing.hpp`.

38.14.1.80 PrintPYX()

```
void mlpack::bindings::python::PrintPYX (
    const    util::ProgramDoc & programInfo,
    const std::string & mainFilename,
    const std::string & functionName )
```

Given a list of parameter definition and program documentation, print a generated .pyx file to stdout.

Parameters

<i>parameters</i>	List of parameters the program will use (from CLI (p. 1117)).
<i>programInfo</i>	Documentation for the program.
<i>mainFilename</i>	Filename of the main program (i.e. "/path/to/pca_main.cpp").
<i>functionName</i>	Name of the function (i.e. "pca").

38.14.1.81 PrintTypeDoc() [1/6]

```
std::string mlpack::bindings::python::PrintTypeDoc (
    const    util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if<    util::IsStdVector< T >>::type * = 0,
    const typename boost::disable_if<    data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple<    data::DatasetInfo,
    arma::mat >>>::type * = 0 )
```

Return a string representing the command-line type of an option.

38.14.1.82 PrintTypeDoc() [2/6]

```
std::string mlpack::bindings::python::PrintTypeDoc (
    const    util::ParamData & data,
    const typename std::enable_if<    util::IsStdVector< T >::value >::type * = 0 )
```

Return a string representing the command-line type of a vector.

38.14.1.83 PrintTypeDoc() [3/6]

```
std::string mlpack::bindings::python::PrintTypeDoc (
    const util::ParamData & data,
    const typename std::enable_if< arma::is_arma_type< T >::value >::type * = 0 )
```

Return a string representing the command-line type of a matrix option.

38.14.1.84 PrintTypeDoc() [4/6]

```
std::string mlpack::bindings::python::PrintTypeDoc (
    const util::ParamData & data,
    const typename std::enable_if< std::is_same< T, std::tuple< data::DatasetInfo,
    arma::mat >>::value >::type * = 0 )
```

Return a string representing the command-line type of a matrix tuple option.

38.14.1.85 PrintTypeDoc() [5/6]

```
std::string mlpack::bindings::python::PrintTypeDoc (
    const util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

Return a string representing the command-line type of a model.

38.14.1.86 PrintTypeDoc() [6/6]

```
void mlpack::bindings::python::PrintTypeDoc (
    const util::ParamData & data,
    const void * ,
    void * output )
```

Print the command-line type of an option into a string.

Definition at line 72 of file `print_type_doc.hpp`.

38.14.1.87 PrintValue() [1/2]

```
std::string mlpack::bindings::python::PrintValue (
    const T & value,
    bool quotes ) [inline]
```

Given a parameter type, print the corresponding value.

38.14.1.88 PrintValue() [2/2]

```
std::string mlpack::bindings::python::PrintValue (
    const bool & value,
    bool quotes ) [inline]
```

38.14.1.89 ProgramCall() [1/2]

```
std::string mlpack::bindings::python::ProgramCall (
    const std::string & programName,
    Args... args )
```

Given a name of a binding and a variable number of arguments (and their contents), print the corresponding function call.

38.14.1.90 ProgramCall() [2/2]

```
std::string mlpack::bindings::python::ProgramCall (
    const std::string & programName ) [inline]
```

Given the name of a binding, print a program call assuming that all options are specified.

38.14.1.91 SerializeIn()

```
void mlpack::bindings::python::SerializeIn (
    T * t,
    const std::string & str,
    const std::string & name )
```

Definition at line 34 of file `serialization.hpp`.

38.14.1.92 SerializeOut()

```
std::string mlpack::bindings::python::SerializeOut (
    T * t,
    const std::string & name )
```

Definition at line 22 of file serialization.hpp.

38.14.1.93 StripType()

```
void mlpack::bindings::python::StripType (
    const std::string & inputType,
    std::string & strippedType,
    std::string & printedType,
    std::string & defaultsType ) [inline]
```

Given an input type like, e.g., "LogisticRegression<>", return three types that can be used in Python code.

strippedType will be a type with no template parameters (e.g. "LogisticRegression"), printedType will be a printable type with the template parameters (e.g. "LogisticRegression[]"), and defaultsType will be a printable type with a default template parameter (e.g. "LogisticRegression[T=*]") that can be used for class definitions.

Definition at line 28 of file strip_type.hpp.

Referenced by ImportDecl(), PrintClassDefn(), PrintInputProcessing(), and PrintOutputProcessing().

38.14.2 Variable Documentation

38.14.2.1 programName

```
std::string programName
```

38.15 mlpack::bindings::tests Namespace Reference

Classes

- class **ProgramDoc**
A static object whose constructor registers program documentation with the **CLI** (p. 1117) class.
- class **TestOption**
A static object whose constructor registers a parameter with the **CLI** (p. 1117) class.

Functions

- void **CleanMemory** ()
*Delete any unique pointers that are held by the **CLI** (p. 1117) object.*
- template<typename T >
void **DeleteAllocatedMemory** (const **util::ParamData** &d, const void *, void *)
- template<typename T >
void **DeleteAllocatedMemoryImpl** (const **util::ParamData** &, const typename boost::disable_if< **data::HasSerialize**< T >>::type *=0, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0)
- template<typename T >
void **DeleteAllocatedMemoryImpl** (const **util::ParamData** &, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)
- template<typename T >
void **DeleteAllocatedMemoryImpl** (const **util::ParamData** &d, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< **data::HasSerialize**< T >>::type *=0)
- template<typename T >
void * **GetAllocatedMemory** (const **util::ParamData** &, const typename boost::disable_if< **data::HasSerialize**< T >>::type *=0, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0)
- template<typename T >
void * **GetAllocatedMemory** (const **util::ParamData** &, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)
- template<typename T >
void * **GetAllocatedMemory** (const **util::ParamData** &d, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< **data::HasSerialize**< T >>::type *=0)
- template<typename T >
void **GetAllocatedMemory** (const **util::ParamData** &d, const void *, void *output)
- template<typename T >
T & **GetParam** (**util::ParamData** &d)
This overload is called when nothing special needs to happen to the name of the parameter.
- template<typename T >
void **GetParam** (const **util::ParamData** &d, const void *, void *output)
Return a parameter casted to the given type.
- template<typename T >
std::string **GetPrintableParam** (const **util::ParamData** &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< **util::IsStdVector**< T >>::type *=0, const typename boost::disable_if< **data::HasSerialize**< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< **data::DatasetInfo**, arma::mat >>>::type *=0)
Print an option.
- template<typename T >
std::string **GetPrintableParam** (const **util::ParamData** &data, const typename boost::enable_if< **util::IsStdVector**< T >>::type *=0)
Print a vector option, with spaces between it.
- template<typename T >
std::string **GetPrintableParam** (const **util::ParamData** &data, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)
Print a matrix option (this just prints the filename).
- template<typename T >
std::string **GetPrintableParam** (const **util::ParamData** &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< **data::HasSerialize**< T >>::type *=0)
Print a serializable class option (this just prints the filename).

- template<typename T >
std::string **GetPrintableParam** (const **util::ParamData** &data, const typename boost::enable_if< std::is_same< T, std::tuple< **data::DatasetInfo**, arma::mat >>>::type !=0)
Print a mapped matrix option (this just prints the filename).
- template<typename T >
void **GetPrintableParam** (const **util::ParamData** &data, const void *, void *output)
Print an option into a std::string.
- template<typename T >
bool **IgnoreCheck** (const T &)
Return whether or not a parameter check should be ignored.

Variables

- std::string **programName**

38.15.1 Function Documentation

38.15.1.1 CleanMemory()

```
void mlpack::bindings::tests::CleanMemory ( )
```

Delete any unique pointers that are held by the **CLI** (p. 1117) object.

Referenced by **KDEModel::KernelType()**, and **RSModel::RandomBasis()**.

38.15.1.2 DeleteAllocatedMemory()

```
void mlpack::bindings::tests::DeleteAllocatedMemory (
    const util::ParamData & d,
    const void * ,
    void * )
```

Definition at line 49 of file `delete_allocated_memory.hpp`.

38.15.1.3 DeleteAllocatedMemoryImpl() [1/3]

```
void mlpack::bindings::tests::DeleteAllocatedMemoryImpl (
    const util::ParamData & ,
    const typename boost::disable_if< data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0 )
```

Definition at line 22 of file `delete_allocated_memory.hpp`.

38.15.1.4 DeleteAllocatedMemoryImpl() [2/3]

```
void mlpack::bindings::tests::DeleteAllocatedMemoryImpl (
    const    util::ParamData & ,
    const typename boost::enable_if< arma::is_arma_type< T >>::type *   = 0 )
```

Definition at line 31 of file delete_allocated_memory.hpp.

38.15.1.5 DeleteAllocatedMemoryImpl() [3/3]

```
void mlpack::bindings::tests::DeleteAllocatedMemoryImpl (
    const    util::ParamData & d,
    const typename boost::disable_if< arma::is_arma_type< T >>::type *   = 0,
    const typename boost::enable_if<    data::HasSerialize< T >>::type *   = 0 )
```

Definition at line 39 of file delete_allocated_memory.hpp.

References ParamData::value.

38.15.1.6 GetAllocatedMemory() [1/4]

```
void* mlpack::bindings::tests::GetAllocatedMemory (
    const    util::ParamData & ,
    const typename boost::disable_if<    data::HasSerialize< T >>::type *   = 0,
    const typename boost::disable_if< arma::is_arma_type< T >>::type *   = 0 )
```

Definition at line 23 of file get_allocated_memory.hpp.

38.15.1.7 GetAllocatedMemory() [2/4]

```
void* mlpack::bindings::tests::GetAllocatedMemory (
    const    util::ParamData & ,
    const typename boost::enable_if< arma::is_arma_type< T >>::type *   = 0 )
```

Definition at line 32 of file get_allocated_memory.hpp.

38.15.1.8 GetAllocatedMemory() [3/4]

```
void* mlpack::bindings::tests::GetAllocatedMemory (
    const util::ParamData & d,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::enable_if< data::HasSerialize< T >>::type * = 0 )
```

Definition at line 40 of file `get_allocated_memory.hpp`.

References `ParamData::value`.

38.15.1.9 GetAllocatedMemory() [4/4]

```
void mlpack::bindings::tests::GetAllocatedMemory (
    const util::ParamData & d,
    const void * ,
    void * output )
```

Definition at line 50 of file `get_allocated_memory.hpp`.

38.15.1.10 GetParam() [1/2]

```
T& mlpack::bindings::tests::GetParam (
    util::ParamData & d )
```

This overload is called when nothing special needs to happen to the name of the parameter.

Definition at line 26 of file `get_param.hpp`.

References `ParamData::value`.

38.15.1.11 GetParam() [2/2]

```
void mlpack::bindings::tests::GetParam (
    const util::ParamData & d,
    const void * ,
    void * output )
```

Return a parameter casted to the given type.

Type checking does not happen here!

Parameters

<i>d</i>	Parameter information.
<i>input</i>	Unused parameter.
<i>output</i>	Place to store pointer to value.

Definition at line 41 of file `get_param.hpp`.

38.15.1.12 `GetPrintableParam()` [1/6]

```
std::string mlpack::bindings::tests::GetPrintableParam (
    const    util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type * = 0,
    const typename boost::disable_if<    util::IsStdVector< T >>::type * = 0,
    const typename boost::disable_if<    data::HasSerialize< T >>::type * = 0,
    const typename boost::disable_if< std::is_same< T, std::tuple<    data::DatasetInfo,
arma::mat >>>::type * = 0 )
```

Print an option.

Print an option.

Definition at line 26 of file `get_printable_param.hpp`.

References `ParamData::value`.

38.15.1.13 `GetPrintableParam()` [2/6]

```
std::string mlpack::bindings::tests::GetPrintableParam (
    const    util::ParamData & data,
    const typename boost::enable_if<    util::IsStdVector< T >>::type * = 0 )
```

Print a vector option, with spaces between it.

Definition at line 43 of file `get_printable_param.hpp`.

References `ParamData::value`.

38.15.1.14 GetPrintableParam() [3/6]

```
std::string mlpack::bindings::tests::GetPrintableParam (
    const    util::ParamData & data,
    const typename boost::enable_if< arma::is_arma_type< T >>::type *   = 0 )
```

Print a matrix option (this just prints the filename).

Print a matrix option (this just prints the filename).

Definition at line 59 of file get_printable_param.hpp.

References ParamData::value.

38.15.1.15 GetPrintableParam() [4/6]

```
std::string mlpack::bindings::tests::GetPrintableParam (
    const    util::ParamData & data,
    const typename boost::disable_if< arma::is_arma_type< T >>::type *   = 0,
    const typename boost::enable_if<    data::HasSerialize< T >>::type *   = 0 )
```

Print a serializable class option (this just prints the filename).

Print a serializable class option (this just prints the filename).

Print a model option (this just prints the filename).

Definition at line 75 of file get_printable_param.hpp.

References ParamData::cppType, and ParamData::value.

38.15.1.16 GetPrintableParam() [5/6]

```
std::string mlpack::bindings::tests::GetPrintableParam (
    const    util::ParamData & data,
    const typename boost::enable_if< std::is_same< T, std::tuple<    data::DatasetInfo,
arma::mat >>>::type *   = 0 )
```

Print a mapped matrix option (this just prints the filename).

Print a mapped matrix option (this just prints the filename).

Definition at line 89 of file get_printable_param.hpp.

References ParamData::value.

38.15.1.17 GetPrintableParam() [6/6]

```
void mlpack::bindings::tests::GetPrintableParam (
    const    util::ParamData & data,
    const void * ,
    void * output )
```

Print an option into a std::string.

This should print a short, one-line representation of the object. The string will be stored in the output pointer.

Definition at line 76 of file get_printable_param.hpp.

38.15.1.18 IgnoreCheck()

```
bool mlpack::bindings::tests::IgnoreCheck (
    const T & ) [inline]
```

Return whether or not a parameter check should be ignored.

Return whether or not a runtime check on parameters should be ignored.

For test bindings, we do not ignore any checks, so this always returns false.

Definition at line 24 of file ignore_check.hpp.

38.15.2 Variable Documentation

38.15.2.1 programName

```
std::string programName
```

38.16 mlpack::bound Namespace Reference

Namespaces

- **addr**
- **meta**

Metaprogramming utilities.

Classes

- class **BallBound**
Ball bound encloses a set of points at a specific distance (radius) from a specific point (center).
- struct **BoundTraits**
A class to obtain compile-time traits about BoundType classes.
- struct **BoundTraits**< **BallBound**< **MetricType**, **VecType** > >
*A specialization of **BoundTraits** (p. 1002) for this bound type.*
- struct **BoundTraits**< **CellBound**< **MetricType**, **ElemType** > >
- struct **BoundTraits**< **HollowBallBound**< **MetricType**, **ElemType** > >
*A specialization of **BoundTraits** (p. 1002) for this bound type.*
- struct **BoundTraits**< **HRectBound**< **MetricType**, **ElemType** > >
- class **CellBound**
*The **CellBound** (p. 1006) class describes a bound that consists of a number of hyperrectangles.*
- class **HollowBallBound**
Hollow ball bound encloses a set of points at a specific distance (radius) from a specific point (center) except points at a specific distance from another point (the center of the hole).
- class **HRectBound**
Hyper-rectangle bound for an L-metric.

38.17 mlpack::bound::addr Namespace Reference

Functions

- template<typename AddressType , typename VecType >
void **AddressToPoint** (VecType &point, const AddressType &address)
Translate the address to the point.
- template<typename AddressType1 , typename AddressType2 >
int **CompareAddresses** (const AddressType1 &addr1, const AddressType2 &addr2)
Compare two addresses.
- template<typename AddressType1 , typename AddressType2 , typename AddressType3 >
bool **Contains** (const AddressType1 &address, const AddressType2 &loBound, const AddressType3 &hiBound)
Returns true if an address is contained between two other addresses.
- template<typename AddressType , typename VecType >
void **PointToAddress** (AddressType &address, const VecType &point)
Calculate the address of a point.

38.17.1 Function Documentation

38.17.1.1 AddressToPoint()

```
void mlpack::bound::addr::AddressToPoint (
    VecType & point,
    const AddressType & address )
```

Translate the address to the point.

Be careful, the point and the address variables should be equal-sized and the type of the address should correspond to the type of the vector.

The function makes the backward transform to the function above.

Parameters

<i>address</i>	An address to translate.
<i>point</i>	The point that corresponds to the address.

Definition at line 153 of file address.hpp.

38.17.1.2 CompareAddresses()

```
int mlpack::bound::addr::CompareAddresses (
    const AddressType1 & addr1,
    const AddressType2 & addr2 )
```

Compare two addresses.

The function returns 1 if the first address is greater than the second one, -1 if the first address is less than the second one, otherwise the function returns 0.

Definition at line 233 of file address.hpp.

Referenced by Contains().

38.17.1.3 Contains()

```
bool mlpack::bound::addr::Contains (
    const AddressType1 & address,
    const AddressType2 & loBound,
    const AddressType3 & hiBound )
```

Returns true if an address is contained between two other addresses.

Definition at line 256 of file address.hpp.

References CompareAddresses().

Referenced by HRectBound< MetricType >::Metric().

38.17.1.4 PointToAddress()

```
void mlpack::bound::addr::PointToAddress (
    AddressType & address,
    const VecType & point )
```

Calculate the address of a point.

Be careful, the point and the address variables should be equal-sized and the type of the address should correspond to the type of the vector.

The function maps each floating point coordinate to an equal-sized unsigned integer datatype in such a way that the transform preserves the ordering (i.e. lower floating point values correspond to lower integers). Thus, the mapping saves the exponent and the mantissa of each floating point value consequently, furthermore the exponent is stored before the mantissa. In the case of negative numbers the resulting integer value should be inverted. In the multi-dimensional case, after we transform the representation, we have to interleave the bits of the new representation across all the elements in the address vector.

Parameters

<i>address</i>	The resulting address.
<i>point</i>	The point that is being translated to the address.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Definition at line 57 of file address.hpp.

38.18 mlpack::bound::meta Namespace Reference

Metaprogramming utilities.

Classes

- struct **IsLMetric**
Utility struct where Value is true if and only if the argument is of type LMetric.
- struct **IsLMetric**< **metric::LMetric**< **Power**, **TakeRoot** > >
*Specialization for **IsLMetric** (p. 1028) when the argument is of type LMetric.*

38.18.1 Detailed Description

Metaprogramming utilities.

38.19 mlpack::cf Namespace Reference

Collaborative filtering.

Classes

- class **AverageInterpolation**
This class performs average interpolation to generate interpolation weights for neighborhood-based collaborative filtering.
- class **BatchSVDPolicy**
*Implementation of the Batch SVD policy to act as a wrapper when accessing Batch SVD from within **CFTYPE** (p. 1045).*
- class **BiasSVDPolicy**
*Implementation of the Bias SVD policy to act as a wrapper when accessing Bias SVD from within **CFTYPE** (p. 1045).*
- class **CFModel**
The model to save to disk.
- class **CFTYPE**

This class implements Collaborative Filtering (CF).

- class **CombinedNormalization**

This normalization class performs a sequence of normalization methods on raw ratings.

- class **CosineSearch**

Nearest neighbor search with cosine distance.

- class **DeleteVisitor**

***DeleteVisitor** (p. 1058) deletes the **CFTYPE** object which is pointed to by the variable **cf** in class **CFModel** (p. 1042).*

- class **DummyClass**

This class acts as a dummy class for passing as template parameter.

- class **GetValueVisitor**

***GetValueVisitor** (p. 1060) returns the pointer which points to the **CFTYPE** (p. 1045) object.*

- class **ItemMeanNormalization**

This normalization class performs item mean normalization on raw ratings.

- class **LMetricSearch**

Nearest neighbor search with L_p distance.

- class **NMFPolicy**

*Implementation of the NMF policy to act as a wrapper when accessing NMF from within **CFTYPE** (p. 1045).*

- class **NoNormalization**

This normalization class doesn't perform any normalization.

- class **OverallMeanNormalization**

This normalization class performs overall mean normalization on raw ratings.

- class **PearsonSearch**

Nearest neighbor search with pearson distance (or furthest neighbor search with pearson correlation).

- class **PredictVisitor**

***PredictVisitor** (p. 1077) uses the **CFTYPE** (p. 1045) object to make predictions on the given combinations of users and items.*

- class **RandomizedSVDPolicy**

*Implementation of the Randomized SVD policy to act as a wrapper when accessing Randomized SVD from within **CFTYPE** (p. 1045).*

- class **RecommendationVisitor**

***RecommendationVisitor** (p. 1084) uses the **CFTYPE** (p. 1045) object to get recommendations for the given users.*

- class **RegressionInterpolation**

Implementation of regression-based interpolation method.

- class **RegSVDPolicy**

*Implementation of the Regularized SVD policy to act as a wrapper when accessing Regularized SVD from within **CFTYPE** (p. 1045).*

- class **SimilarityInterpolation**

*With **SimilarityInterpolation** (p. 1092), interpolation weights are based on similarities between query user and its neighbors.*

- class **SVDCompletePolicy**

*Implementation of the SVD complete incremental policy to act as a wrapper when accessing SVD complete decomposition from within **CFTYPE** (p. 1045).*

- class **SVDIncompletePolicy**

*Implementation of the SVD incomplete incremental to act as a wrapper when accessing SVD incomplete incremental from within **CFTYPE** (p. 1045).*

- class **SVDPlusPlusPolicy**

*Implementation of the SVDPlusPlus policy to act as a wrapper when accessing SVDPlusPlus from within **CFTYPE** (p. 1045).*

- class **SVDWrapper**

This class acts as the wrapper for all SVD factorizers which are incompatible with CF module.

- class **UserMeanNormalization**

This normalization class performs user mean normalization on raw ratings.

- class **ZScoreNormalization**

This normalization class performs z-score normalization on raw ratings.

Typedefs

- typedef **SVDWrapper**< **DummyClass** > **ArmaSVDFactorizer**

add simple typedefs

- using **EuclideanSearch** = **LMetricSearch**< 2 >

38.19.1 Detailed Description

Collaborative filtering.

38.19.2 Typedef Documentation

38.19.2.1 ArmaSVDFactorizer

```
typedef SVDWrapper< DummyClass> ArmaSVDFactorizer
```

add simple typedefs

Definition at line 86 of file svd_wrapper.hpp.

38.19.2.2 EuclideanSearch

```
using EuclideanSearch = LMetricSearch<2>
```

Definition at line 79 of file lmetric_search.hpp.

38.20 mlpack::cv Namespace Reference

Classes

- class **Accuracy**

The **Accuracy** (p. 1127) is a metric of performance for classification algorithms that is equal to a proportion of correctly labeled test items among all ones for given test items.

- class **CVBase**

An auxiliary class for cross-validation.

- class **F1**

F1 (p. 1132) is a metric of performance for classification algorithms that for binary classification is equal to $2 * precision * recall / (precision + recall)$.

- class **KFoldCV**

The class **KFoldCV** (p. 1134) implements k-fold cross-validation for regression and classification algorithms.

- class **MetalInfoExtractor**

MetalInfoExtractor (p. 1140) is a tool for extracting meta information about a given machine learning algorithm.

- class **MSE**

The **MeanSquaredError** is a metric of performance for regression algorithms that is equal to the mean squared error between predicted values and ground truth (correct) values for given test items.

- struct **NotFoundMethodForm**

- class **Precision**

Precision (p. 1145) is a metric of performance for classification algorithms that for binary classification is equal to $tp / (tp + fp)$, where tp and fp are the numbers of true positives and false positives respectively.

- class **Recall**

Recall (p. 1147) is a metric of performance for classification algorithms that for binary classification is equal to $tp / (tp + fn)$, where tp and fn are the numbers of true positives and false negatives respectively.

- struct **SelectMethodForm**

A type function that selects a right method form.

- struct **SelectMethodForm**< **MLAlgorithm** >

- struct **SelectMethodForm**< **MLAlgorithm**, **HasMethodForm**, **HMFs...** >

- class **SimpleCV**

SimpleCV (p. 1151) splits data into two sets - training and validation sets - and then runs training on the training set and evaluates performance on the validation set.

- struct **TrainForm**

A wrapper struct for holding a Train form.

- struct **TrainForm**< **MT**, **PT**, **void**, **false**, **false** >

- struct **TrainForm**< **MT**, **PT**, **void**, **false**, **true** >

- struct **TrainForm**< **MT**, **PT**, **void**, **true**, **false** >

- struct **TrainForm**< **MT**, **PT**, **void**, **true**, **true** >

- struct **TrainForm**< **MT**, **PT**, **WT**, **false**, **false** >

- struct **TrainForm**< **MT**, **PT**, **WT**, **false**, **true** >

- struct **TrainForm**< **MT**, **PT**, **WT**, **true**, **false** >

- struct **TrainForm**< **MT**, **PT**, **WT**, **true**, **true** >

- struct **TrainFormBase4**

- struct **TrainFormBase5**

- struct **TrainFormBase6**

- struct **TrainFormBase7**

Enumerations

- enum **AverageStrategy** {
Binary,
Micro,
Macro }

This enum declares possible strategies for averaging that can be used in some metrics like precision, recall, and F1.

Functions

- template<typename DataType >
void **AssertSizes** (const DataType &data, const arma::Row< size_t > &labels, const std::string &caller←
Description)

Assert there is the same number of the given data points and labels.

38.20.1 Enumeration Type Documentation

38.20.1.1 AverageStrategy

enum **AverageStrategy**

This enum declares possible strategies for averaging that can be used in some metrics like precision, recall, and **F1** (p. 1132).

The "Binary" strategy means binary classification is going to be used, and there is no need to average.

Enumerator

Binary	
Micro	
Macro	

Definition at line 25 of file average_strategy.hpp.

38.20.2 Function Documentation

38.20.2.1 AssertSizes()

```
void mlpack::cv::AssertSizes (
    const DataType & data,
```

```
const arma::Row< size_t > & labels,
const std::string & callerDescription )
```

Assert there is the same number of the given data points and labels.

Parameters

<i>data</i>	Column-major data.
<i>labels</i>	Labels.
<i>callerDescription</i>	A description of the caller that can be used for error generation.

Definition at line 29 of file facilities.hpp.

38.21 mlpack::data Namespace Reference

Functions to load and save matrices and models.

Classes

- class **CustomImputation**
A simple custom imputation class.
- class **DatasetMapper**
Auxiliary information for a dataset, including mappings to/from strings (or other types) and the datatype of each dimension.
- struct **HasSerialize**
- struct **HasSerializeFunction**
- class **Imputer**
Given a dataset of a particular datatype, replace user-specified missing value with a variable dependent on the Strategy↔ Type and MapperType.
- class **IncrementPolicy**
IncrementPolicy (p. 1184) is used as a helper class for DatasetMapper (p. 1172).
- class **ListwiseDeletion**
A complete-case analysis to remove the values containing mappedValue.
- class **LoadCSV**
Load the csv file. This class use boost::spirit to implement the parser, please refer to following link [http↔://theboostcpplibraries.com/boost.spirit](http://theboostcpplibraries.com/boost.spirit) for quick review.
- class **MeanImputation**
A simple mean imputation class.
- class **MedianImputation**
This is a class implementation of simple median imputation.
- class **MissingPolicy**
MissingPolicy (p. 1193) is used as a helper class for DatasetMapper (p. 1172).

Typedefs

- using **DatasetInfo** = **DatasetMapper**< **data::IncrementPolicy** >

Enumerations

- enum **Datatype** : bool {
numeric = 0,
categorical = 1 }

The Datatype enum specifies the types of data mpack algorithms can use.

- enum **format** {
autodetect,
text,
xml,
binary }

*Define the formats we can read through **boost::serialization** (p. 251).*

Functions

- template<typename T >
void **Binarize** (const arma::Mat< T > &input, arma::Mat< T > &output, const double threshold)
Given an input dataset and threshold, set values greater than threshold to 1 and values less than or equal to the threshold to 0.
- template<typename T >
void **Binarize** (const arma::Mat< T > &input, arma::Mat< T > &output, const double threshold, const size_t dimension)
Given an input dataset and threshold, set values greater than threshold to 1 and values less than or equal to the threshold to 0.
- template<typename eT >
void **ConfusionMatrix** (const arma::Row< size_t > predictors, const arma::Row< size_t > responses, arma::Mat< eT > &output, const size_t numClasses)
A confusion matrix is a summary of prediction results on a classification problem.
- std::string **Extension** (const std::string &filename)
- HAS_EXACT_METHOD_FORM** (serialize, HasSerializeCheck)
- template<typename T >
bool **IsNaNInf** (T &val, const std::string &token)
See if the token is a NaN or an Inf, and if so, set the value accordingly and return a boolean representing whether or not it is.
- template<typename eT >
bool **Load** (const std::string &filename, arma::Mat< eT > &matrix, const bool fatal=false, const bool transpose=true)
Loads a matrix from file, guessing the filetype from the extension.
- template<typename eT >
bool **Load** (const std::string &filename, arma::Col< eT > &vec, const bool fatal=false)
Don't document these with doxygen; these declarations aren't helpful to users.
- template<typename eT >
bool **Load** (const std::string &filename, arma::Row< eT > &rowvec, const bool fatal=false)
Load a row vector from a file, guessing the filetype from the extension.
- template<typename eT, typename PolicyType >
bool **Load** (const std::string &filename, arma::Mat< eT > &matrix, **DatasetMapper**< PolicyType > &info, const bool fatal=false, const bool transpose=true)
*Loads a matrix from a file, guessing the filetype from the extension and mapping categorical features with a **DatasetMapper** (p. 1172) object.*

- `template<typename T >`
`bool Load (const std::string &filename, const std::string &name, T &t, const bool fatal=false, format f=format←
::autodetect)`
Don't document these with doxygen; they aren't helpful for users to know about.
- `template<typename eT >`
`void LoadARFF (const std::string &filename, arma::Mat< eT > &matrix)`
A utility function to load an ARFF dataset as numeric features (that is, as an Armadillo matrix without any modification).
- `template<typename eT, typename PolicyType >`
`void LoadARFF (const std::string &filename, arma::Mat< eT > &matrix, DatasetMapper< PolicyType > &info)`
A utility function to load an ARFF dataset as numeric and categorical features, using the DatasetInfo structure for mapping.
- `template<typename eT, typename RowType >`
`void NormalizeLabels (const RowType &labelsIn, arma::Row< size_t > &labels, arma::Col< eT > &mapping)`
Given a set of labels of a particular datatype, convert them to unsigned labels in the range [0, n) where n is the number of different labels.
- `template<typename eT, typename RowType >`
`void OneHotEncoding (const RowType &labelsIn, arma::Mat< eT > &output)`
Given a set of labels of a particular datatype, convert them to binary vector.
- `template<typename eT >`
`void RevertLabels (const arma::Row< size_t > &labels, const arma::Col< eT > &mapping, arma::Row< eT > &labelsOut)`
Given a set of labels that have been mapped to the range [0, n), map them back to the original labels given by the 'mapping' vector.
- `template<typename eT >`
`bool Save (const std::string &filename, const arma::Mat< eT > &matrix, const bool fatal=false, bool transpose=true)`
Saves a matrix to file, guessing the filetype from the extension.
- `template<typename T >`
`bool Save (const std::string &filename, const std::string &name, T &t, const bool fatal=false, format f=format←
::autodetect)`
Saves a model to file, guessing the filetype from the extension, or, optionally, saving the specified format.
- `template<typename T, typename U >`
`void Split (const arma::Mat< T > &input, const arma::Row< U > &inputLabel, arma::Mat< T > &trainData, arma::Mat< T > &testData, arma::Row< U > &trainLabel, arma::Row< U > &testLabel, const double testRatio)`
Given an input dataset and labels, split into a training set and test set.
- `template<typename T >`
`void Split (const arma::Mat< T > &input, arma::Mat< T > &trainData, arma::Mat< T > &testData, const double testRatio)`
Given an input dataset, split into a training set and test set.
- `template<typename T, typename U >`
`std::tuple< arma::Mat< T >, arma::Mat< T >, arma::Row< U >, arma::Row< U > > Split (const arma::Mat< T > &input, const arma::Row< U > &inputLabel, const double testRatio)`
Given an input dataset and labels, split into a training set and test set.
- `template<typename T >`
`std::tuple< arma::Mat< T >, arma::Mat< T > > Split (const arma::Mat< T > &input, const double testRatio)`
Given an input dataset, split into a training set and test set.

38.21.1 Detailed Description

Functions to load and save matrices and models.

Functions to load and save matrices.

38.21.2 Typedef Documentation

38.21.2.1 DatasetInfo

```
typedef DatasetMapper< IncrementPolicy, std::string > DatasetInfo
```

Definition at line 196 of file dataset_mapper.hpp.

38.21.3 Enumeration Type Documentation

38.21.3.1 Datatype

```
enum Datatype : bool
```

The Datatype enum specifies the types of data mlpack algorithms can use.

The vast majority of mlpack algorithms can only use numeric data (i.e. float/double/etc.), but some algorithms can use categorical data, specified via this Datatype enum and the **DatasetMapper** (p. 1172) class.

Enumerator

numeric	
categorical	

Definition at line 24 of file datatype.hpp.

38.21.3.2 format

```
enum format
```

Define the formats we can read through **boost::serialization** (p. 251).

Enumerator

autodetect	
text	
xml	
binary	

Definition at line 20 of file format.hpp.

38.21.4 Function Documentation

38.21.4.1 Binarize() [1/2]

```
void mlpack::data::Binarize (
    const arma::Mat< T > & input,
    arma::Mat< T > & output,
    const double threshold )
```

Given an input dataset and threshold, set values greater than threshold to 1 and values less than or equal to the threshold to 0.

This overload applies the changes to all dimensions.

```
arma::Mat<double> input = loadData();
arma::Mat<double> output;
double threshold = 0.5;

// Binarize the whole Matrix. All positive values in will be set to 1 and
// the values less than or equal to 0.5 will become 0.
Binarize<double>(input, output, threshold);
```

Parameters

<i>input</i>	Input matrix to Binarize.
<i>output</i>	Matrix you want to save binarized data into.
<i>threshold</i>	Threshold can by any number.

Definition at line 41 of file binarize.hpp.

References `omp_size_t`.

38.21.4.2 Binarize() [2/2]

```
void mlpack::data::Binarize (
    const arma::Mat< T > & input,
    arma::Mat< T > & output,
    const double threshold,
    const size_t dimension )
```

Given an input dataset and threshold, set values greater than threshold to 1 and values less than or equal to the threshold to 0.

This overload takes a dimension and applys the changes to the given dimension.

```
arma::Mat<double> input = loadData();
arma::Mat<double> output;
double threshold = 0.5;
size_t dimension = 0;

// Binarize the first dimension. All positive values in the first dimension
// will be set to 1 and the values less than or equal to 0 will become 0.
Binarize<double>(input, output, threshold, dimension);
```

Parameters

<i>input</i>	Input matrix to Binarize.
<i>output</i>	Matrix you want to save binarized data into.
<i>threshold</i>	Threshold can by any number.
<i>dimension</i>	Feature to apply the Binarize function.

Definition at line 77 of file binarize.hpp.

References `omp_size_t`.

38.21.4.3 ConfusionMatrix()

```
void mlpack::data::ConfusionMatrix (
    const arma::Row< size_t > predictors,
    const arma::Row< size_t > responses,
    arma::Mat< eT > & output,
    const size_t numClasses )
```

A confusion matrix is a summary of prediction results on a classification problem.

The number of correct and incorrect predictions are summarized by count and broken down by each class. For example, for 2 classes, the function call will be

```
ConfusionMatrix(predictors, responses, output, 2)
```

In this case, the output matrix will be of size 2 * 2:

	0	1
0	TP	FN
1	FP	TN

The confusion matrix for two labels will look like what is shown above. In this confusion matrix, TP represents the number of true positives, FP represents the number of false positives, FN represents the number of false negatives, and TN represents the number of true negatives.

When generalizing to 2 or more classes, the row index of the confusion matrix represents the predicted classes and column index represents the actual class.

Parameters

<i>predictors</i>	Vector of data points.
<i>responses</i>	The measured data for each point.
<i>output</i>	Matrix which is represented as confusion matrix.
<i>numClasses</i>	Number of classes.

38.21.4.4 Extension()

```
std::string mlpack::data::Extension (
    const std::string & filename ) [inline]
```

Definition at line 21 of file extension.hpp.

38.21.4.5 HAS_EXACT_METHOD_FORM()

```
mlpack::data::HAS_EXACT_METHOD_FORM (
    serialize ,
    HasSerializeCheck )
```

38.21.4.6 IsNaNInf()

```
bool mlpack::data::IsNaNInf (
    T & val,
    const std::string & token ) [inline]
```

See if the token is a NaN or an Inf, and if so, set the value accordingly and return a boolean representing whether or not it is.

Definition at line 27 of file is_naninf.hpp.

38.21.4.7 Load() [1/5]

```
bool mpack::data::Load (
    const std::string & filename,
    arma::Mat< eT > & matrix,
    const bool fatal = false,
    const bool transpose = true )
```

Loads a matrix from file, guessing the filetype from the extension.

This will transpose the matrix at load time (unless the transpose parameter is set to false). If the filetype cannot be determined, an error will be given.

The supported types of files are the same as found in Armadillo:

- CSV (csv_ascii), denoted by .csv, or optionally .txt
- TSV (raw_ascii), denoted by .tsv, .csv, or .txt
- ASCII (raw_ascii), denoted by .txt
- Armadillo ASCII (arma_ascii), also denoted by .txt
- PGM (pgm_binary), denoted by .pgm
- PPM (ppm_binary), denoted by .ppm
- Raw binary (raw_binary), denoted by .bin
- Armadillo binary (arma_binary), denoted by .bin
- HDF5, denoted by .hdf, .hdf5, .h5, or .he5

If the file extension is not one of those types, an error will be given. This is preferable to Armadillo's default behavior of loading an unknown filetype as raw_binary, which can have very confusing effects.

If the parameter 'fatal' is set to true, a std::runtime_error exception will be thrown if the matrix does not load successfully. The parameter 'transpose' controls whether or not the matrix is transposed after loading. In most cases, because data is generally stored in a row-major format and mpack requires column-major matrices, this should be left at its default value of 'true'.

Parameters

<i>filename</i>	Name of file to load.
<i>matrix</i>	Matrix to load contents of file into.
<i>fatal</i>	If an error should be reported as fatal (default false).
<i>transpose</i>	If true, transpose the matrix after loading.

Returns

Boolean value indicating success or failure of load.

Referenced by mpack::bindings::cli::GetParam().

38.21.4.8 Load() [2/5]

```
bool mlpack::data::Load (
    const std::string & filename,
    arma::Col< eT > & vec,
    const bool fatal = false )
```

Don't document these with doxygen; these declarations aren't helpful to users.

Load a column vector from a file, guessing the filetype from the extension.

The supported types of files are the same as found in Armadillo:

- CSV (csv_ascii), denoted by .csv, or optionally .txt
- TSV (raw_ascii), denoted by .tsv, .csv, or .txt
- ASCII (raw_ascii), denoted by .txt
- Armadillo ASCII (arma_ascii), also denoted by .txt
- PGM (pgm_binary), denoted by .pgm
- PPM (ppm_binary), denoted by .ppm
- Raw binary (raw_binary), denoted by .bin
- Armadillo binary (arma_binary), denoted by .bin
- HDF5, denoted by .hdf, .hdf5, .h5, or .he5

If the file extension is not one of those types, an error will be given. This is preferable to Armadillo's default behavior of loading an unknown filetype as raw_binary, which can have very confusing effects.

If the parameter 'fatal' is set to true, a std::runtime_error exception will be thrown if the matrix does not load successfully.

Parameters

<i>filename</i>	Name of file to load.
<i>colvec</i>	Column vector to load contents of file into.
<i>fatal</i>	If an error should be reported as fatal (default false).

Returns

Boolean value indicating success or failure of load.

38.21.4.9 Load() [3/5]

```
bool mlpack::data::Load (
    const std::string & filename,
```

```
arma::Row< eT > & rowvec,
const bool fatal = false )
```

Load a row vector from a file, guessing the filetype from the extension.

The supported types of files are the same as found in Armadillo:

- CSV (csv_ascii), denoted by .csv, or optionally .txt
- TSV (raw_ascii), denoted by .tsv, .csv, or .txt
- ASCII (raw_ascii), denoted by .txt
- Armadillo ASCII (arma_ascii), also denoted by .txt
- PGM (pgm_binary), denoted by .pgm
- PPM (ppm_binary), denoted by .ppm
- Raw binary (raw_binary), denoted by .bin
- Armadillo binary (arma_binary), denoted by .bin
- HDF5, denoted by .hdf, .hdf5, .h5, or .he5

If the file extension is not one of those types, an error will be given. This is preferable to Armadillo's default behavior of loading an unknown filetype as raw_binary, which can have very confusing effects.

If the parameter 'fatal' is set to true, a std::runtime_error exception will be thrown if the matrix does not load successfully.

Parameters

<i>filename</i>	Name of file to load.
<i>colvec</i>	Column vector to load contents of file into.
<i>fatal</i>	If an error should be reported as fatal (default false).

Returns

Boolean value indicating success or failure of load.

38.21.4.10 Load() [4/5]

```
bool mlpack::data::Load (
    const std::string & filename,
    arma::Mat< eT > & matrix,
    DatasetMapper< PolicyType > & info,
    const bool fatal = false,
    const bool transpose = true )
```

Loads a matrix from a file, guessing the filetype from the extension and mapping categorical features with a **DatasetMapper** (p. 1172) object.

This will transpose the matrix (unless the transpose parameter is set to false). This particular overload of **Load()** (p. 379) can only load text-based formats, such as those given below:

- CSV (csv_ascii), denoted by .csv, or optionally .txt
- TSV (raw_ascii), denoted by .tsv, .csv, or .txt
- ASCII (raw_ascii), denoted by .txt

If the file extension is not one of those types, an error will be given. This is preferable to Armadillo's default behavior of loading an unknown filetype as raw_binary, which can have very confusing effects.

If the parameter 'fatal' is set to true, a std::runtime_error exception will be thrown if the matrix does not load successfully. The parameter 'transpose' controls whether or not the matrix is transposed after loading. In most cases, because data is generally stored in a row-major format and mlpack requires column-major matrices, this should be left at its default value of 'true'.

The **DatasetMapper** (p. 1172) object passed to this function will be re-created, so any mappings from previous loads will be lost.

Parameters

<i>filename</i>	Name of file to load.
<i>matrix</i>	Matrix to load contents of file into.
<i>info</i>	DatasetMapper (p. 1172) object to populate with mappings and data types.
<i>fatal</i>	If an error should be reported as fatal (default false).
<i>transpose</i>	If true, transpose the matrix after loading.

Returns

Boolean value indicating success or failure of load.

38.21.4.11 Load() [5/5]

```
bool mlpack::data::Load (
    const std::string & filename,
    const std::string & name,
    T & t,
    const bool fatal = false,
    format f = format::autodetect )
```

Don't document these with doxygen; they aren't helpful for users to know about.

Load a model from a file, guessing the filetype from the extension, or, optionally, loading the specified format. If automatic extension detection is used and the filetype cannot be determined, an error will be given.

The supported types of files are the same as what is supported by the **boost::serialization** (p. 251) library:

- text, denoted by .txt
- xml, denoted by .xml
- binary, denoted by .bin

The format parameter can take any of the values in the 'format' enum: 'format::autodetect', 'format::text', 'format::xml', and 'format::binary'. The autodetect functionality operates on the file extension (so, "file.txt" would be autodetected as text).

The name parameter should be specified to indicate the name of the structure to be loaded. This should be the same as the name that was used to save the structure (otherwise, the loading procedure will fail).

If the parameter 'fatal' is set to true, then an exception will be thrown in the event of load failure. Otherwise, the method will return false and the relevant error information will be printed to **Log::Warn** (p. 1541).

38.21.4.12 LoadARFF() [1/2]

```
void mlpack::data::LoadARFF (
    const std::string & filename,
    arma::Mat< eT > & matrix )
```

A utility function to load an ARFF dataset as numeric features (that is, as an Armadillo matrix without any modification).

An exception will be thrown if any features are non-numeric.

38.21.4.13 LoadARFF() [2/2]

```
void mlpack::data::LoadARFF (
    const std::string & filename,
    arma::Mat< eT > & matrix,
    DatasetMapper< PolicyType > & info )
```

A utility function to load an ARFF dataset as numeric and categorical features, using the DatasetInfo structure for mapping.

An exception will be thrown upon failure.

A pre-existing DatasetInfo object can be passed in, but if the dimensionality of the given DatasetInfo object (info.Dimensionality()) does not match the dimensionality of the data, a std::invalid_argument exception will be thrown. If an empty DatasetInfo object is given (constructed with the default constructor or otherwise, so that info.Dimensionality() is 0), it will be set to the right dimensionality.

This ability to pass in pre-existing DatasetInfo objects is very necessary when, e.g., loading a test set after training. If the same DatasetInfo from loading the training set is not used, then the test set may be loaded with different mappings—which can cause horrible problems!

Parameters

<i>filename</i>	Name of ARFF file to load.
<i>matrix</i>	Matrix to load data into.
<i>info</i>	DatasetInfo object; can be default-constructed or pre-existing from another call to LoadARFF() (p. 384).

38.21.4.14 NormalizeLabels()

```
void mlpack::data::NormalizeLabels (
    const RowType & labelsIn,
    arma::Row< size_t > & labels,
    arma::Col< eT > & mapping )
```

Given a set of labels of a particular datatype, convert them to unsigned labels in the range [0, n) where n is the number of different labels.

Also, a reverse mapping from the new label to the old value is stored in the 'mapping' vector.

Parameters

<i>labelsIn</i>	Input labels of arbitrary datatype.
<i>labels</i>	Vector that unsigned labels will be stored in.
<i>mapping</i>	Reverse mapping to convert new labels back to old labels.

38.21.4.15 OneHotEncoding()

```
void mlpack::data::OneHotEncoding (
    const RowType & labelsIn,
    arma::Mat< eT > & output )
```

Given a set of labels of a particular datatype, convert them to binary vector.

The categorical values be mapped to integer values. Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1.

Parameters

<i>labels↔ In</i>	Input labels of arbitrary datatype.
<i>output</i>	Binary matrix.

38.21.4.16 RevertLabels()

```
void mlpack::data::RevertLabels (
    const arma::Row< size_t > & labels,
```

```
const arma::Col< eT > & mapping,
arma::Row< eT > & labelsOut )
```

Given a set of labels that have been mapped to the range [0, n), map them back to the original labels given by the 'mapping' vector.

Parameters

<i>labels</i>	Set of normalized labels to convert.
<i>mapping</i>	Mapping to use to convert labels.
<i>labelsOut</i>	Vector to store new labels in.

38.21.4.17 Save() [1/2]

```
bool mlpack::data::Save (
    const std::string & filename,
    const arma::Mat< eT > & matrix,
    const bool fatal = false,
    bool transpose = true )
```

Saves a matrix to file, guessing the filetype from the extension.

This will transpose the matrix at save time. If the filetype cannot be determined, an error will be given.

The supported types of files are the same as found in Armadillo:

- CSV (csv_ascii), denoted by .csv, or optionally .txt
- ASCII (raw_ascii), denoted by .txt
- Armadillo ASCII (arma_ascii), also denoted by .txt
- PGM (pgm_binary), denoted by .pgm
- PPM (ppm_binary), denoted by .ppm
- Raw binary (raw_binary), denoted by .bin
- Armadillo binary (arma_binary), denoted by .bin
- HDF5 (hdf5_binary), denoted by .hdf5, .hdf, .h5, or .he5

If the file extension is not one of those types, an error will be given. If the 'fatal' parameter is set to true, a `std::runtime_error` exception will be thrown upon failure. If the 'transpose' parameter is set to true, the matrix will be transposed before saving. Generally, because mlpack stores matrices in a column-major format and most datasets are stored on disk as row-major, this parameter should be left at its default value of 'true'.

Parameters

<i>filename</i>	Name of file to save to.
<i>matrix</i>	Matrix to save into file.
<i>fatal</i>	If an error should be reported as fatal (default false).
<i>transpose</i>	If true, transpose the matrix before saving.

Returns

Boolean value indicating success or failure of save.

38.21.4.18 Save() [2/2]

```
bool mlpack::data::Save (
    const std::string & filename,
    const std::string & name,
    T & t,
    const bool fatal = false,
    format f = format::autodetect )
```

Saves a model to file, guessing the filetype from the extension, or, optionally, saving the specified format.

If automatic extension detection is used and the filetype cannot be determined, an error will be given.

The supported types of files are the same as what is supported by the **boost::serialization** (p. 251) library:

- text, denoted by .txt
- xml, denoted by .xml
- binary, denoted by .bin

The format parameter can take any of the values in the 'format' enum: 'format::autodetect', 'format::text', 'format::xml', and 'format::binary'. The autodetect functionality operates on the file extension (so, "file.txt" would be autodetected as text).

The name parameter should be specified to indicate the name of the structure to be saved. If **Load()** (p. 379) is later called on the generated file, the name used to load should be the same as the name used for this call to **Save()** (p. 386).

If the parameter 'fatal' is set to true, then an exception will be thrown in the event of a save failure. Otherwise, the method will return false and the relevant error information will be printed to **Log::Warn** (p. 1541).

38.21.4.19 Split() [1/4]

```
void mlpack::data::Split (
    const arma::Mat< T > & input,
    const arma::Row< U > & inputLabel,
    arma::Mat< T > & trainData,
    arma::Mat< T > & testData,
    arma::Row< U > & trainLabel,
    arma::Row< U > & testLabel,
    const double testRatio )
```

Given an input dataset and labels, split into a training set and test set.

Example usage below. This overload places the split dataset into the four output parameters given (trainData, testData, trainLabel, and testLabel).

```

arma::mat input = loadData();
arma::Row<size_t> label = loadLabel();
arma::mat trainData;
arma::mat testData;
arma::Row<size_t> trainLabel;
arma::Row<size_t> testLabel;
math::RandomSeed(100); // Set the seed if you like.

// Split the dataset into a training and test set, with 30% of the data being
// held out for the test set.
Split(input, label, trainData,
      testData, trainLabel, testLabel, 0.3);

```

Parameters

<i>input</i>	Input dataset to split.
<i>label</i>	Input labels to split.
<i>trainData</i>	Matrix to store training data into.
<i>testData</i>	Matrix to store test data into.
<i>trainLabel</i>	Vector to store training labels into.
<i>testLabel</i>	Vector to store test labels into.
<i>testRatio</i>	Percentage of dataset to use for test set (between 0 and 1).

Definition at line 49 of file `split_data.hpp`.

Referenced by `Split()`.

38.21.4.20 Split() [2/4]

```

void mlpack::data::Split (
    const arma::Mat< T > & input,
    arma::Mat< T > & trainData,
    arma::Mat< T > & testData,
    const double testRatio )

```

Given an input dataset, split into a training set and test set.

Example usage below. This overload places the split dataset into the two output parameters given (`trainData`, `testData`).

```

arma::mat input = loadData();
arma::mat trainData;
arma::mat testData;
math::RandomSeed(100); // Set the seed if you like.

// Split the dataset into a training and test set, with 30% of the data being
// held out for the test set.
Split(input, trainData, testData, 0.3);

```

Parameters

<i>input</i>	Input dataset to split.
<i>trainData</i>	Matrix to store training data into.
<i>testData</i>	Matrix to store test data into.
<i>testRatio</i>	Percentage of dataset to use for test set (between 0 and 1).

Definition at line 103 of file split_data.hpp.

38.21.4.21 Split() [3/4]

```
std::tuple<arma::Mat<T>, arma::Mat<T>, arma::Row<U>, arma::Row<U> > mlpack::data::Split (
    const arma::Mat< T > & input,
    const arma::Row< U > & inputLabel,
    const double testRatio )
```

Given an input dataset and labels, split into a training set and test set.

Example usage below. This overload returns the split dataset as a `std::tuple` with four elements: an `arma::Mat<T>` containing the training data, an `arma::Mat<T>` containing the test data, an `arma::Row<U>` containing the training labels, and an `arma::Row<U>` containing the test labels.

```
arma::mat input = loadData();
arma::Row<size_t> label = loadLabel();
auto splitResult = Split(input, label, 0.2);
```

Parameters

<i>input</i>	Input dataset to split.
<i>label</i>	Input labels to split.
<i>testRatio</i>	Percentage of dataset to use for test set (between 0 and 1).

Returns

`std::tuple` containing `trainData` (`arma::Mat<T>`), `testData` (`arma::Mat<T>`), `trainLabel` (`arma::Row<U>`), and `testLabel` (`arma::Row<U>`).

Definition at line 148 of file split_data.hpp.

References `Split()`.

38.21.4.22 Split() [4/4]

```
std::tuple<arma::Mat<T>, arma::Mat<T> > mlpack::data::Split (
    const arma::Mat< T > & input,
    const double testRatio )
```

Given an input dataset, split into a training set and test set.

Example usage below. This overload returns the split dataset as a `std::tuple` with two elements: an `arma::Mat<T>` containing the training data and an `arma::Mat<T>` containing the test data.

```
arma::mat input = loadData();
auto splitResult = Split(input, 0.2);
```

Parameters

<i>input</i>	Input dataset to split.
<i>testRatio</i>	Percentage of dataset to use for test set (between 0 and 1).

Returns

std::tuple containing trainData (arma::Mat<T>) and testData (arma::Mat<T>).

Definition at line 184 of file split_data.hpp.

References Split().

38.22 mlpack::dbscan Namespace Reference

Classes

- class **DBSCAN**
DBSCAN (p. 1197) (*Density-Based Spatial Clustering of Applications with Noise*) is a clustering technique described in the following paper:
- class **OrderedPointSelection**
This class can be used to sequentially select the next point to use for DBSCAN (p. 1197).
- class **RandomPointSelection**
This class can be used to randomly select the next point to use for DBSCAN (p. 1197).

38.23 mlpack::decision_stump Namespace Reference

Classes

- class **DecisionStump**
This class implements a decision stump.

38.24 mlpack::det Namespace Reference

Density Estimation Trees.

Classes

- class **DTree**
A density estimation tree is similar to both a decision tree and a space partitioning tree (like a kd-tree).
- class **PathCacher**
This class is responsible for caching the path to each node of the tree.

Functions

- `template<typename MatType, typename TagType >`
`void PrintLeafMembership (DTree< MatType, TagType > *dtree, const MatType &data, const arma::Mat< size_t > &labels, const size_t numClasses, const std::string &leafClassMembershipFile="")`
Print the membership of leaves of a density estimation tree given the labels and number of classes.
- `template<typename MatType, typename TagType >`
`void PrintVariableImportance (const DTree< MatType, TagType > *dtree, const std::string viFile="")`
Print the variable importance of each dimension of a density estimation tree.
- `template<typename MatType, typename TagType >`
`DTree< MatType, TagType > * Trainer (MatType &dataset, const size_t folds, const bool useVolumeReg=false, const size_t maxLeafSize=10, const size_t minLeafSize=5, const std::string unprunedTreeOutput="", const bool skipPruning=false)`
Train the optimal decision tree using cross-validation with the given number of folds.

38.24.1 Detailed Description

Density Estimation Trees.

38.24.2 Function Documentation

38.24.2.1 PrintLeafMembership()

```
void mlpack::det::PrintLeafMembership (
    DTree< MatType, TagType > * dtree,
    const MatType & data,
    const arma::Mat< size_t > & labels,
    const size_t numClasses,
    const std::string & leafClassMembershipFile = "" )
```

Print the membership of leaves of a density estimation tree given the labels and number of classes.

Optionally, pass the name of a file to print this information to (otherwise stdout is used).

Parameters

<i>dtree</i>	Tree to print membership of.
<i>data</i>	Dataset tree is built upon.
<i>labels</i>	Class labels of dataset.
<i>numClasses</i>	Number of classes in dataset.
<i>leafClassMembershipFile</i>	Name of file to print to (optional).

38.24.2.2 PrintVariableImportance()

```
void mlpack::det::PrintVariableImportance (
    const DTree< MatType, TagType > * dtree,
    const std::string viFile = "" )
```

Print the variable importance of each dimension of a density estimation tree.

Optionally, pass the name of a file to print this information to (otherwise stdout is used).

Parameters

<i>dtree</i>	Density tree to use.
<i>viFile</i>	Name of file to print to (optional).

38.24.2.3 Trainer()

```
DTree<MatType, TagType>* mlpack::det::Trainer (
    MatType & dataset,
    const size_t folds,
    const bool useVolumeReg = false,
    const size_t maxLeafSize = 10,
    const size_t minLeafSize = 5,
    const std::string unprunedTreeOutput = "",
    const bool skipPruning = false )
```

Train the optimal decision tree using cross-validation with the given number of folds.

Optionally, give a filename to print the unpruned tree to. This initializes a tree on the heap, so you are responsible for deleting it.

Parameters

<i>dataset</i>	Dataset for the tree to use.
<i>folds</i>	Number of folds to use for cross-validation.
<i>useVolumeReg</i>	If true, use volume regularization.
<i>maxLeafSize</i>	Maximum number of points allowed in a leaf.
<i>minLeafSize</i>	Minimum number of points allowed in a leaf.
<i>unprunedTreeOutput</i>	Filename to print unpruned tree to (optional).

38.25 mlpack::distribution Namespace Reference

Probability distributions.

Classes

- class **DiagonalGaussianDistribution**
A single multivariate Gaussian distribution with diagonal covariance.
- class **DiscreteDistribution**
A discrete distribution where the only observations are discrete observations.
- class **GammaDistribution**
This class represents the Gamma distribution.
- class **GaussianDistribution**
A single multivariate Gaussian distribution.
- class **LaplaceDistribution**
The multivariate Laplace distribution centered at 0 has pdf.
- class **RegressionDistribution**
A class that represents a univariate conditionally Gaussian distribution.

38.25.1 Detailed Description

Probability distributions.

38.26 mlpack::emst Namespace Reference

Euclidean Minimum Spanning Trees.

Classes

- class **DTBRules**
- class **DTBStat**
A statistic for use with mlpack trees, which stores the upper bound on distance to nearest neighbors and the component which this node belongs to.
- class **DualTreeBoruvka**
Performs the MST calculation using the Dual-Tree Boruvka algorithm, using any type of tree.
- class **EdgePair**
An edge pair is simply two indices and a distance.
- class **UnionFind**
A Union-Find data structure.

38.26.1 Detailed Description

Euclidean Minimum Spanning Trees.

38.27 mlpack::fastmks Namespace Reference

Fast max-kernel search.

Classes

- class **FastMKS**
An implementation of fast exact max-kernel search.
- class **FastMKSMoel**
*A utility struct to contain all the possible **FastMKS** (p. 1280) models, for use by the mlpack_fastmks program.*
- class **FastMKSRules**
*The **FastMKSRules** (p. 1298) class is a template helper class used by **FastMKS** (p. 1280) class when performing exact max-kernel search.*
- class **FastMKStat**
*The statistic used in trees with **FastMKS** (p. 1280).*

38.27.1 Detailed Description

Fast max-kernel search.

38.28 mlpack::gmm Namespace Reference

Gaussian Mixture Models.

Classes

- class **DiagonalConstraint**
Force a covariance matrix to be diagonal.
- class **DiagonalGMM**
A Diagonal Gaussian Mixture Model.
- class **EigenvalueRatioConstraint**
Given a vector of eigenvalue ratios, ensure that the covariance matrix always has those eigenvalue ratios.
- class **EMFit**
*This class contains methods which can fit a **GMM** (p. 1323) to observations using the EM algorithm.*
- class **GMM**
*A Gaussian Mixture Model (**GMM** (p. 1323)).*
- class **NoConstraint**
This class enforces no constraint on the covariance matrix.
- class **PositiveDefiniteConstraint**
Given a covariance matrix, force the matrix to be positive definite.

38.28.1 Detailed Description

Gaussian Mixture Models.

38.29 mlpack::hmm Namespace Reference

Hidden Markov Models.

Classes

- class **HMM**
A class that represents a Hidden Markov Model with an arbitrary type of emission distribution.
- class **HMMModel**
*A serializable **HMM** (p. 1335) model that also stores the type.*
- class **HMMRegression**
A class that represents a Hidden Markov Model Regression (HMMR).

Enumerations

- enum **HMMType** : char {
DiscreteHMM = 0,
GaussianHMM,
GaussianMixtureModelHMM,
DiagonalGaussianMixtureModelHMM,
DiscreteHMM = 0,
GaussianHMM,
GaussianMixtureModelHMM,
DiagonalGaussianMixtureModelHMM }
- enum **HMMType** : char {
DiscreteHMM = 0,
GaussianHMM,
GaussianMixtureModelHMM,
DiagonalGaussianMixtureModelHMM,
DiscreteHMM = 0,
GaussianHMM,
GaussianMixtureModelHMM,
DiagonalGaussianMixtureModelHMM }
HMMType, to be stored on disk.

Functions

- template<typename ActionType, typename ExtraInfoType = void>
void **LoadHMMAndPerformAction** (const std::string &modelFile, ExtraInfoType *x=NULL)
ActionType should implement static void Apply(HMMType&).
- template<typename HMMType >
void **SaveHMM** (**HMMType** &hmm, const std::string &modelFile)
*Save an **HMM** (p. 1335) to a file.*

38.29.1 Detailed Description

Hidden Markov Models.

38.29.2 Enumeration Type Documentation

38.29.2.1 HMMType [1/2]

```
enum HMMType : char
```

Enumerator

DiscreteHMM	
GaussianHMM	
GaussianMixtureModelHMM	
DiagonalGaussianMixtureModelHMM	
DiscreteHMM	
GaussianHMM	
GaussianMixtureModelHMM	
DiagonalGaussianMixtureModelHMM	

Definition at line 22 of file hmm_model.hpp.

38.29.2.2 HMMType [2/2]

```
enum HMMType : char
```

HMMType, to be stored on disk.

This is of type char, which is one byte. (I'm not sure what will happen on systems where one byte is not eight bits.)

Enumerator

DiscreteHMM	
GaussianHMM	
GaussianMixtureModelHMM	
DiagonalGaussianMixtureModelHMM	
DiscreteHMM	
GaussianHMM	
GaussianMixtureModelHMM	
DiagonalGaussianMixtureModelHMM	

Definition at line 22 of file hmm_util.hpp.

38.29.3 Function Documentation

38.29.3.1 LoadHMMAndPerformAction()

```
void mlpack::hmm::LoadHMMAndPerformAction (
    const std::string & modelFile,
    ExtraInfoType * x = NULL )
```

ActionType should implement static void Apply(HMMType&).

38.29.3.2 SaveHMM()

```
void mlpack::hmm::SaveHMM (
    HMMType & hmm,
    const std::string & modelFile )
```

Save an **HMM** (p. 1335) to a file.

The file must also encode what type of **HMM** (p. 1335) is being stored.

38.30 mlpack::hpt Namespace Reference

Classes

- class **CVFunction**
This wrapper serves for adapting the interface of the cross-validation classes to the one that can be utilized by the mlpack optimizers.
- struct **DeduceHyperParameterTypes**
*A type function for deducing types of hyper-parameters from types of arguments in the Optimize method in **Hyper↔ParameterTuner** (p. 1375).*
- struct **DeduceHyperParameterTypes< PreFixedArg< T >, Args... >**
*Defining **DeduceHyperParameterTypes** (p. 1365) for the case when not all argument types have been processed, and the next one is the type of an argument that should be fixed.*
- struct **DeduceHyperParameterTypes< T, Args... >**
*Defining **DeduceHyperParameterTypes** (p. 1365) for the case when not all argument types have been processed, and the next one (T) is a collection type or an arithmetic type.*
- struct **FixedArg**
A struct for storing information about a fixed argument.
- class **HyperParameterTuner**

The class **HyperParameterTuner** (p. 1375) for the given *MLAlgorithm* utilizes the provided *Optimizer* to find the values of hyper-parameters that optimize the value of the given *Metric*.

- class **IsPreFixedArg**

A type function for checking whether the given type is **PreFixedArg** (p. 1381).

- struct **PreFixedArg**

A struct for marking arguments as ones that should be fixed (it can be useful for the *Optimize* method of **HyperParameterTuner** (p. 1375)).

- struct **PreFixedArg< T & >**

The specialization of the template for references.

Typedefs

- template<typename... Args>

using **TupleOfHyperParameters** = typename **DeduceHyperParameterTypes**< Args... >::TupleType

A short alias for deducing types of hyper-parameters from types of arguments in the *Optimize* method in **HyperParameterTuner** (p. 1375).

Functions

- template<typename T >

PreFixedArg< T > Fixed (T &&value)

Mark the given argument as one that should be fixed.

38.30.1 Typedef Documentation

38.30.1.1 TupleOfHyperParameters

```
using TupleOfHyperParameters = typename DeduceHyperParameterTypes<Args...>::TupleType
```

A short alias for deducing types of hyper-parameters from types of arguments in the *Optimize* method in **HyperParameterTuner** (p. 1375).

Definition at line 127 of file `deduce_hp_types.hpp`.

38.30.2 Function Documentation

38.30.2.1 Fixed()

```
PreFixedArg<T> mlpack::hpt::Fixed (
    T && value )
```

Mark the given argument as one that should be fixed.

It can be applied to arguments that are passed to the Optimize method of **HyperParameterTuner** (p. 1375).

The implementation avoids data copying. If the passed argument is an l-value reference, we store it as a const l-value rereference inside the returned **PreFixedArg** (p. 1381) object. If the passed argument is an r-value reference, light-weight copying (by taking possession of the r-value) will be made during the initialization of the returned **PreFixedArg** (p. 1381) object.

Definition at line 36 of file fixed.hpp.

38.31 mlpack::kde Namespace Reference

Kernel Density Estimation.

Classes

- class **DeleteVisitor**
- class **DualBiKDE**
 - DualBiKDE* (p. 1384) computes a Kernel Density Estimation on the given KDEType.
- class **DualMonoKDE**
 - DualMonoKDE* (p. 1386) computes a Kernel Density Estimation on the given KDEType.
- class **KDE**
 - The KDE* (p. 1387) class is a template class for performing Kernel Density Estimations.
- class **KDEModel**
- class **KDERules**
 - A dual-tree traversal Rules class for kernel density estimation.*
- class **KDEStat**
 - Extra data for each node in the tree for the task of kernel density estimation.*
- class **KernelNormalizer**
 - KernelNormalizer* (p. 1410) holds a set of methods to normalize estimations applying in each case the appropriate kernel normalizer function.
- class **ModeVisitor**
 - ModeVisitor* (p. 1412) exposes the Mode() method of the KDEType.
- class **TrainVisitor**
 - TrainVisitor* (p. 1413) trains a given KDEType using a reference set.

Typedefs

- `template<typename KernelType, template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType> using KDEType = KDE< KernelType, metric::EuclideanDistance, arma::mat, TreeType, TreeType< metric::EuclideanDistance, kde::KDEStat, arma::mat >::template DualTreeTraverser, TreeType< metric::EuclideanDistance, kde::KDEStat, arma::mat >::template SingleTreeTraverser >`
- Alias template.*

Enumerations

- enum **KDEMode** {
DUAL_TREE_MODE,
SINGLE_TREE_MODE }

KDEMode represents the ways in which KDE algorithm can be executed.

38.31.1 Detailed Description

Kernel Density Estimation.

38.31.2 Typedef Documentation

38.31.2.1 KDEType

```
using KDEType = KDE<KernelType, metric::EuclideanDistance, arma::mat, TreeType, TreeType<
metric::EuclideanDistance, kde::KDEStat, arma::mat>::template DualTreeTraverser, TreeType< metric<
::EuclideanDistance, kde::KDEStat, arma::mat>::template SingleTreeTraverser>
```

Alias template.

Definition at line 45 of file kde_model.hpp.

38.31.3 Enumeration Type Documentation

38.31.3.1 KDEMode

```
enum KDEMode
```

KDEMode represents the ways in which **KDE** (p. 1387) algorithm can be executed.

Enumerator

DUAL_TREE_MODE	
SINGLE_TREE_MODE	

Definition at line 25 of file kde.hpp.

38.32 mlpack::kernel Namespace Reference

Kernel functions.

Classes

- class **CauchyKernel**
The Cauchy kernel.
- class **CosineDistance**
The cosine distance (or cosine similarity).
- class **EpanechnikovKernel**
The Epanechnikov kernel, defined as.
- class **ExampleKernel**
An example kernel function.
- class **GaussianKernel**
The standard Gaussian kernel.
- class **HyperbolicTangentKernel**
Hyperbolic tangent kernel.
- class **KernelTraits**
This is a template class that can provide information about various kernels.
- class **KernelTraits< CauchyKernel >**
Kernel traits for the Cauchy kernel.
- class **KernelTraits< CosineDistance >**
Kernel traits for the cosine distance.
- class **KernelTraits< EpanechnikovKernel >**
Kernel traits for the Epanechnikov kernel.
- class **KernelTraits< GaussianKernel >**
Kernel traits for the Gaussian kernel.
- class **KernelTraits< LaplacianKernel >**
Kernel traits of the Laplacian kernel.
- class **KernelTraits< SphericalKernel >**
Kernel traits for the spherical kernel.
- class **KernelTraits< TriangularKernel >**
Kernel traits for the triangular kernel.
- class **KMeansSelection**
Implementation of the kmeans sampling scheme.
- class **LaplacianKernel**
The standard Laplacian kernel.
- class **LinearKernel**
The simple linear kernel (dot product).
- class **NystroemMethod**
- class **OrderedSelection**
- class **PolynomialKernel**
The simple polynomial kernel.
- class **PSpectrumStringKernel**
The p-spectrum string kernel.

- class **RandomSelection**
- class **SphericalKernel**

The spherical kernel, which is 1 when the distance between the two argument points is less than or equal to the bandwidth, or 0 otherwise.

- class **TriangularKernel**

The trivially simple triangular kernel, defined by.

38.32.1 Detailed Description

Kernel functions.

This namespace contains kernel functions, which evaluate some kernel function $K(x, y)$ for some arbitrary vectors x and y of the same dimension. The single restriction on the function $K(x, y)$ is that it must satisfy Mercer's condition:

$$\int \int K(x, y) g(x) g(y) dx dy \geq 0$$

for all square integrable functions $g(x)$.

The kernels in this namespace all implement the KernelType policy. For more information, see **The KernelType policy documentation** (p. 173).

38.33 mlpack::kmeans Namespace Reference

K-Means clustering.

Classes

- class **AllowEmptyClusters**

Policy which allows K-Means to create empty clusters without any error being reported.

- class **DualTreeKMeans**

An algorithm for an exact Lloyd iteration which simply uses dual-tree nearest-neighbor search to find the nearest centroid for each point in the dataset.

- class **DualTreeKMeansRules**
- class **DualTreeKMeansStatistic**
- class **ElkanKMeans**
- class **HamerlyKMeans**
- class **KillEmptyClusters**

Policy which allows K-Means to "kill" empty clusters without any error being reported.

- class **KMeans**

This class implements K-Means clustering, using a variety of possible implementations of Lloyd's algorithm.

- class **MaxVarianceNewCluster**

When an empty cluster is detected, this class takes the point furthest from the centroid of the cluster with maximum variance as a new cluster.

- class **NaiveKMeans**

This is an implementation of a single iteration of Lloyd's algorithm for k-means.

- class **PellegMooreKMeans**

An implementation of Pelleg-Moore's 'blacklist' algorithm for k-means clustering.

- class **PellegMooreKMeansRules**

The rules class for the single-tree Pelleg-Moore kd-tree traversal for k-means clustering.

- class **PellegMooreKMeansStatistic**

A statistic for trees which holds the blacklist for Pelleg-Moore k-means clustering (which represents the clusters that cannot possibly own any points in a node).

- class **RandomPartition**

A very simple partitioner which partitions the data randomly into the number of desired clusters.

- class **RefinedStart**

A refined approach for choosing initial points for k-means clustering.

- class **SampleInitialization**

Typedefs

- template<typename MetricType , typename MatType >
using **CoverTreeDualTreeKMeans** = **DualTreeKMeans**< MetricType, MatType, **tree::StandardCoverTree** >
*A template typedef for the **DualTreeKMeans** (p. 1466) algorithm with the cover tree type.*
- template<typename MetricType , typename MatType >
using **DefaultDualTreeKMeans** = **DualTreeKMeans**< MetricType, MatType >
*A template typedef for the **DualTreeKMeans** (p. 1466) algorithm with the default tree type (a kd-tree).*

Functions

- template<typename TreeType >
void **HideChild** (TreeType &node, const size_t child, const typename **std::enable_if_t**< ! **tree::TreeTraits**< TreeType >::BinaryTree > *junk=0)
Utility function for hiding children.
- template<typename TreeType >
void **HideChild** (TreeType &node, const size_t child, const typename **std::enable_if_t**< **tree::TreeTraits**< TreeType >::BinaryTree > *junk=0)
Utility function for hiding children.
- template<typename TreeType >
void **RestoreChildren** (TreeType &node, const typename **std::enable_if_t**< ! **tree::TreeTraits**< TreeType >↵
::BinaryTree > *junk=0)
Utility function for restoring children to a non-binary tree.
- template<typename TreeType >
void **RestoreChildren** (TreeType &node, const typename **std::enable_if_t**< **tree::TreeTraits**< TreeType >↵
::BinaryTree > *junk=0)
Utility function for restoring children to a binary tree.

38.33.1 Detailed Description

K-Means clustering.

38.33.2 Typedef Documentation

38.33.2.1 CoverTreeDualTreeKMeans

```
using CoverTreeDualTreeKMeans = DualTreeKMeans<MetricType, MatType, tree::StandardCoverTree>
```

A template typedef for the **DualTreeKMeans** (p. 1466) algorithm with the cover tree type.

Definition at line 170 of file dual_tree_kmeans.hpp.

38.33.2.2 DefaultDualTreeKMeans

```
using DefaultDualTreeKMeans = DualTreeKMeans<MetricType, MatType>
```

A template typedef for the **DualTreeKMeans** (p. 1466) algorithm with the default tree type (a kd-tree).

Definition at line 164 of file dual_tree_kmeans.hpp.

38.33.3 Function Documentation

38.33.3.1 HideChild() [1/2]

```
void mlpack::kmeans::HideChild (
    TreeType & node,
    const size_t child,
    const typename std::enable_if_t< ! tree::TreeTraits< TreeType >::BinaryTree > *
junk = 0 )
```

Utility function for hiding children.

This actually does something, and is called if the tree is not a binary tree.

Referenced by DualTreeKMeans< MetricType, MatType, TreeType >::DistanceCalculations().

38.33.3.2 HideChild() [2/2]

```
void mlpack::kmeans::HideChild (
    TreeType & node,
    const size_t child,
    const typename std::enable_if_t< tree::TreeTraits< TreeType >::BinaryTree > * junk
    = 0 )
```

Utility function for hiding children.

This is called when the tree is a binary tree, and does nothing, because we don't hide binary children in this way.

38.33.3.3 RestoreChildren() [1/2]

```
void mlpack::kmeans::RestoreChildren (
    TreeType & node,
    const typename std::enable_if_t<! tree::TreeTraits< TreeType >::BinaryTree > *
    junk = 0 )
```

Utility function for restoring children to a non-binary tree.

Referenced by DualTreeKMeans< MetricType, MatType, TreeType >::DistanceCalculations().

38.33.3.4 RestoreChildren() [2/2]

```
void mlpack::kmeans::RestoreChildren (
    TreeType & node,
    const typename std::enable_if_t< tree::TreeTraits< TreeType >::BinaryTree > * junk
    = 0 )
```

Utility function for restoring children to a binary tree.

38.34 mlpack::kpca Namespace Reference

Classes

- class **KernelPCA**

This class performs kernel principal components analysis (Kernel PCA), for a given kernel.

- class **NaiveKernelRule**
- class **NystroemKernelRule**

38.35 mlpack::lcc Namespace Reference

Classes

- class **LocalCoordinateCoding**

An implementation of Local Coordinate Coding (LCC) that codes data which approximately lives on a manifold using a variation of l1-norm regularized sparse coding; in LCC, the penalty on the absolute value of each point's coefficient for each atom is weighted by the squared distance of that point to that atom.

38.36 mlpack::lmnn Namespace Reference

Large Margin Nearest Neighbor.

Classes

- class **Constraints**

Interface for generating distance based constraints on a given dataset, provided corresponding true labels and a quantity parameter (k) are specified.

- class **LMNN**

An implementation of Large Margin nearest neighbor metric learning technique.

- class **LMNNFunction**

The Large Margin Nearest Neighbors function.

38.36.1 Detailed Description

Large Margin Nearest Neighbor.

38.37 mlpack::math Namespace Reference

Miscellaneous math routines.

Classes

- class **ColumnsToBlocks**

Transform the columns of the given matrix into a block format.

- class **RangeType**

Simple real-valued range.

Typedefs

- typedef **RangeType**< double > **Range**

*3.0.0 TODO: break reverse-compatibility by changing **RangeType** (p. 1549) to **Range**.*

Functions

- template<typename T >
T::elem_type **AccuLog** (const T &x)
Sum a vector of log values.
- void **Center** (const arma::mat &x, arma::mat &xCentered)
Creates a centered matrix, where centering is done by subtracting the sum over the columns (a column vector) from each column of the matrix.
- double **ClampNonNegative** (const double d)
Forces a number to be non-negative, turning negative numbers into zero.
- double **ClampNonPositive** (const double d)
Forces a number to be non-positive, turning positive numbers into zero.
- double **ClampRange** (double value, const double rangeMin, const double rangeMax)
Clamp a number between a particular range.
- template<typename ElemType >
void **ClearAlias** (arma::Mat< ElemType > &mat)
Clear an alias so that no data is overwritten.
- template<typename ElemType >
void **ClearAlias** (arma::SpMat< ElemType > &)
Clear an alias for a sparse matrix.
- template<typename eT >
arma::Mat< eT > **ColumnCovariance** (const arma::Mat< eT > &A, const size_t norm_type=0)
- template<typename T >
arma::Mat< std::complex< T > > **ColumnCovariance** (const arma::Mat< std::complex< T > > &A, const size_t norm_type=0)
- void **FixedRandomSeed** ()
Set the random seed to a fixed number.
- template<typename T >
T **LogAdd** (T x, T y)
Internal log-addition.
- template<typename ElemType >
arma::Cube< ElemType > **MakeAlias** (arma::Cube< ElemType > &input, const bool strict=true)
Make an alias of a dense cube.
- template<typename ElemType >
arma::Mat< ElemType > **MakeAlias** (arma::Mat< ElemType > &input, const bool strict=true)
Make an alias of a dense matrix.
- template<typename ElemType >
arma::Row< ElemType > **MakeAlias** (arma::Row< ElemType > &input, const bool strict=true)
Make an alias of a dense row.
- template<typename ElemType >
arma::Col< ElemType > **MakeAlias** (arma::Col< ElemType > &input, const bool strict=true)
Make an alias of a dense column.
- template<typename ElemType >
arma::SpMat< ElemType > **MakeAlias** (const arma::SpMat< ElemType > &input, const bool=true)
Make a copy of a sparse matrix (an alias is not possible).
- template<typename ElemType >
arma::SpRow< ElemType > **MakeAlias** (const arma::SpRow< ElemType > &input, const bool=true)
Make a copy of a sparse row (an alias is not possible).
- template<typename ElemType >
arma::SpCol< ElemType > **MakeAlias** (const arma::SpCol< ElemType > &input, const bool=true)

Make a copy of a sparse column (an alias is not possible).

- void **ObtainDistinctSamples** (const size_t loInclusive, const size_t hiExclusive, const size_t maxNumSamples, arma::uvec &distinctSamples)

Obtains no more than maxNumSamples distinct samples.

- void **Orthogonalize** (const arma::mat &x, arma::mat &W)

Orthogonalize x and return the result in W, using eigendecomposition.

- void **Orthogonalize** (arma::mat &x)

Orthogonalize x in-place.

- double **RandBernoulli** (const double input)

Generates a 0/1 specified by the input.

- int **RandInt** (const int hiExclusive)

Generates a uniform random integer.

- int **RandInt** (const int lo, const int hiExclusive)

Generates a uniform random integer.

- double **RandNormal** ()

Generates a normally distributed random number with mean 0 and variance 1.

- double **RandNormal** (const double mean, const double variance)

Generates a normally distributed random number with specified mean and variance.

- double **Random** ()

Generates a uniform random number between 0 and 1.

- double **Random** (const double lo, const double hi)

Generates a uniform random number in the specified range.

- void **RandomBasis** (arma::mat &basis, const size_t d)

Create a random d-dimensional orthogonal basis, storing it in the given matrix.

- void **RandomSeed** (const size_t seed)

*Set the random seed used by the random functions (**Random()** (p. 418) and **RandInt()** (p. 417)).*

- void **RandVector** (arma::vec &v)

Overwrites a dimension-N vector to a random vector on the unit sphere in R^N .

- void **RemoveRows** (const arma::mat &input, const std::vector< size_t > &rowsToRemove, arma::mat &output)

Remove a certain set of rows in a matrix while copying to a second matrix.

- template<typename MatType, typename LabelsType >

```
void ShuffleData (const MatType &inputPoints, const LabelsType &inputLabels, MatType &outputPoints,
LabelsType &outputLabels, const std::enable_if_t<!arma::is_SpMat< MatType >::value > *==0, const std::enable_if_t<!arma::is_Cube< MatType >::value > *==0)
```

Shuffle a dataset and associated labels (or responses).

- template<typename MatType, typename LabelsType >

```
void ShuffleData (const MatType &inputPoints, const LabelsType &inputLabels, MatType &outputPoints,
LabelsType &outputLabels, const std::enable_if_t< arma::is_SpMat< MatType >::value > *==0, const std::enable_if_t< arma::is_Cube< MatType >::value > *==0)
```

Shuffle a sparse dataset and associated labels (or responses).

- template<typename MatType, typename LabelsType >

```
void ShuffleData (const MatType &inputPoints, const LabelsType &inputLabels, MatType &outputPoints,
LabelsType &outputLabels, const std::enable_if_t<!arma::is_SpMat< MatType >::value > *==0, const std::enable_if_t< arma::is_Cube< MatType >::value > *==0, const std::enable_if_t< arma::is_Cube< LabelsType >::value > *==0)
```

Shuffle a cube-shaped dataset and associated labels (or responses) which are also cube-shaped.

- `template<typename MatType, typename LabelsType, typename WeightsType >`
`void ShuffleData (const MatType &inputPoints, const LabelsType &inputLabels, const WeightsType &inputWeights, MatType &outputPoints, LabelsType &outputLabels, WeightsType &outputWeights, const std::enable_if_t<!arma::is_SpMat< MatType >::value > * = 0, const std::enable_if_t<!arma::is_Cube< MatType >::value > * = 0)`
Shuffle a dataset and associated labels (or responses) and weights.
- `template<typename MatType, typename LabelsType, typename WeightsType >`
`void ShuffleData (const MatType &inputPoints, const LabelsType &inputLabels, const WeightsType &inputWeights, MatType &outputPoints, LabelsType &outputLabels, WeightsType &outputWeights, const std::enable_if_t<arma::is_SpMat< MatType >::value > * = 0, const std::enable_if_t<!arma::is_Cube< MatType >::value > * = 0)`
Shuffle a sparse dataset and associated labels (or responses) and weights.
- `template<typename T >`
`T Sign (const T x)`
Signum function.
- `void Smat (const arma::vec &input, arma::mat &output)`
The inverse of Svec.
- `void Svec (const arma::mat &input, arma::vec &output)`
Upper triangular representation of a symmetric matrix, scaled such that, $\text{dot}(\text{Svec}(A), \text{Svec}(B)) == \text{dot}(A, B)$ for symmetric A, B .
- `void Svec (const arma::sp_mat &input, arma::sp_vec &output)`
- `size_t SvecIndex (size_t i, size_t j, size_t n)`
*Return the index such that $A[i,j] == \text{factr}(i, j) * \text{svec}(A)[\text{pos}(i, j)]$, where $\text{factr}(i, j) = \sqrt{2}$ if $i \neq j$ and 1 otherwise.*
- `void SymKronId (const arma::mat &A, arma::mat &op)`
*If A is a symmetric matrix, then **SymKronId** returns an operator Op such that.*
- `void VectorPower (arma::vec &vec, const double power)`
Auxiliary function to raise vector elements to a specific power.
- `void WhitenUsingEig (const arma::mat &x, arma::mat &xWhitened, arma::mat &whiteningMatrix)`
Whitens a matrix using the eigendecomposition of the covariance matrix.
- `void WhitenUsingSVD (const arma::mat &x, arma::mat &xWhitened, arma::mat &whiteningMatrix)`
Whitens a matrix using the singular value decomposition of the covariance matrix.

Variables

- `MLPACK_EXPORT std::mt19937 randGen`
MLPACK_EXPORT is required for global variables; it exports the symbols correctly on Windows.
- `MLPACK_EXPORT std::normal_distribution randNormalDist`
- `MLPACK_EXPORT std::uniform_real_distribution randUniformDist`

38.37.1 Detailed Description

Miscellaneous math routines.

38.37.2 Typedef Documentation

38.37.2.1 Range

```
typedef RangeType<double> Range
```

3.0.0 TODO: break reverse-compatibility by changing **RangeType** (p. 1549) to **Range**.

Definition at line 19 of file range.hpp.

38.37.3 Function Documentation

38.37.3.1 AccuLog()

```
T::elem_type mlpack::math::AccuLog (
    const T & x )
```

Sum a vector of log values.

(T should be an Armadillo type.)

Parameters

<i>x</i>	vector of log values
----------	----------------------

Returns

$\log(e^{x_0} + e^{x_1} + \dots)$

38.37.3.2 Center()

```
void mlpack::math::Center (
    const arma::mat & x,
    arma::mat & xCentered )
```

Creates a centered matrix, where centering is done by subtracting the sum over the columns (a column vector) from each column of the matrix.

Parameters

<i>x</i>	Input matrix
<i>xCentered</i>	Matrix to write centered output into

Referenced by NystroemKernelRule< KernelType, PointSelectionPolicy >::ApplyKernelMatrix(), and HRectBound< MetricType >::Metric().

38.37.3.3 ClampNonNegative()

```
double mlpack::math::ClampNonNegative (
    const double d ) [inline]
```

Forces a number to be non-negative, turning negative numbers into zero.

Avoids branching costs (this is a measurable improvement).

Parameters

d	Double to clamp.
-----	------------------

Returns

0 if $d < 0$, d otherwise.

Definition at line 28 of file clamp.hpp.

Referenced by ClampRange().

38.37.3.4 ClampNonPositive()

```
double mlpack::math::ClampNonPositive (
    const double d ) [inline]
```

Forces a number to be non-positive, turning positive numbers into zero.

Avoids branching costs (this is a measurable improvement).

Parameters

d	Double to clamp.
0	if $d > 0$, d otherwise.

Definition at line 40 of file clamp.hpp.

Referenced by ClampRange().

38.37.3.5 ClampRange()

```
double mlpack::math::ClampRange (
    double value,
    const double rangeMin,
    const double rangeMax ) [inline]
```

Clamp a number between a particular range.

Parameters

<i>value</i>	The number to clamp.
<i>rangeMin</i>	The first of the range.
<i>rangeMax</i>	The last of the range.

Returns

`max(rangeMin, min(rangeMax, d)).`

Definition at line 53 of file `clamp.hpp`.

References `ClampNonNegative()`, and `ClampNonPositive()`.

38.37.3.6 ClearAlias() [1/2]

```
void mlpack::math::ClearAlias (
    arma::Mat< ElemType > & mat )
```

Clear an alias so that no data is overwritten.

This resets the matrix if it is an alias (and does nothing otherwise).

Definition at line 110 of file `make_alias.hpp`.

38.37.3.7 ClearAlias() [2/2]

```
void mlpack::math::ClearAlias (
    arma::SpMat< ElemType > & )
```

Clear an alias for a sparse matrix.

This does nothing because no sparse matrices can have aliases.

Definition at line 121 of file `make_alias.hpp`.

38.37.3.8 ColumnCovariance() [1/2]

```
arma::Mat<eT> mlpack::math::ColumnCovariance (
    const arma::Mat< eT > & A,
    const size_t norm_type = 0 ) [inline]
```

38.37.3.9 ColumnCovariance() [2/2]

```
arma::Mat< std::complex<T> > mlpack::math::ColumnCovariance (
    const arma::Mat< std::complex< T > > & A,
    const size_t norm_type = 0 ) [inline]
```

38.37.3.10 FixedRandomSeed()

```
void mlpack::math::FixedRandomSeed ( ) [inline]
```

Set the random seed to a fixed number.

This function is used in binding tests to set a fixed random seed before calling `mlpack()`. In this way we can test whether a certain parameter makes a difference to execution of **CLI** (p. 1117) binding. Refer to pull request #1306 for discussion on this function.

Definition at line 59 of file `random.hpp`.

38.37.3.11 LogAdd()

```
T mlpack::math::LogAdd (
    T x,
    T y )
```

Internal log-addition.

Parameters

x	log value
y	log value

Returns

$\log(e^x + e^y)$

38.37.3.12 MakeAlias() [1/7]

```
arma::Cube<ElemType> mlpack::math::MakeAlias (
    arma::Cube< ElemType > & input,
    const bool strict = true )
```

Make an alias of a dense cube.

If strict is true, then the alias cannot be resized or pointed at new memory.

Definition at line 24 of file make_alias.hpp.

38.37.3.13 MakeAlias() [2/7]

```
arma::Mat<ElemType> mlpack::math::MakeAlias (
    arma::Mat< ElemType > & input,
    const bool strict = true )
```

Make an alias of a dense matrix.

If strict is true, then the alias cannot be resized or pointed at new memory.

Definition at line 37 of file make_alias.hpp.

38.37.3.14 MakeAlias() [3/7]

```
arma::Row<ElemType> mlpack::math::MakeAlias (
    arma::Row< ElemType > & input,
    const bool strict = true )
```

Make an alias of a dense row.

If strict is true, then the alias cannot be resized or pointed at new memory.

Definition at line 50 of file make_alias.hpp.

38.37.3.15 MakeAlias() [4/7]

```
arma::Col<ElemType> mlpack::math::MakeAlias (
    arma::Col< ElemType > & input,
    const bool strict = true )
```

Make an alias of a dense column.

If strict is true, then the alias cannot be resized or pointed at new memory.

Definition at line 62 of file make_alias.hpp.

38.37.3.16 MakeAlias() [5/7]

```
arma::SpMat<ElemType> mlpack::math::MakeAlias (
    const arma::SpMat< ElemType > & input,
    const bool = true )
```

Make a copy of a sparse matrix (an alias is not possible).

The strict parameter is ignored.

Definition at line 74 of file make_alias.hpp.

38.37.3.17 MakeAlias() [6/7]

```
arma::SpRow<ElemType> mlpack::math::MakeAlias (
    const arma::SpRow< ElemType > & input,
    const bool = true )
```

Make a copy of a sparse row (an alias is not possible).

The strict parameter is ignored.

Definition at line 86 of file make_alias.hpp.

38.37.3.18 MakeAlias() [7/7]

```
arma::SpCol<ElemType> mlpack::math::MakeAlias (
    const arma::SpCol< ElemType > & input,
    const bool = true )
```

Make a copy of a sparse column (an alias is not possible).

The strict parameter is ignored.

Definition at line 98 of file make_alias.hpp.

38.37.3.19 ObtainDistinctSamples()

```
void mlpack::math::ObtainDistinctSamples (
    const size_t loInclusive,
    const size_t hiExclusive,
    const size_t maxNumSamples,
    arma::uvec & distinctSamples ) [inline]
```

Obtains no more than maxNumSamples distinct samples.

Each sample belongs to [loInclusive, hiExclusive).

Parameters

<i>loInclusive</i>	The lower bound (inclusive).
<i>hiExclusive</i>	The high bound (exclusive).
<i>maxNumSamples</i>	The maximum number of samples to obtain.
<i>distinctSamples</i>	The samples that will be obtained.

Definition at line 141 of file random.hpp.

References RandInt().

38.37.3.20 Orthogonalize() [1/2]

```
void mlpack::math::Orthogonalize (
    const arma::mat & x,
    arma::mat & W )
```

Orthogonalize x and return the result in W, using eigendecomposition.

We will be using the formula $W = x(x^T x)^{-0.5}$.

38.37.3.21 Orthogonalize() [2/2]

```
void mlpack::math::Orthogonalize (
    arma::mat & x )
```

Orthogonalize x in-place.

This could be sped up by a custom implementation.

38.37.3.22 RandBernoulli()

```
double mlpack::math::RandBernoulli (
    const double input ) [inline]
```

Generates a 0/1 specified by the input.

Definition at line 87 of file random.hpp.

References Random().

38.37.3.23 RandInt() [1/2]

```
int mlpack::math::RandInt (
    const int hiExclusive ) [inline]
```

Generates a uniform random integer.

Definition at line 98 of file random.hpp.

References randUniformDist.

Referenced by RandomDimensionSelect::Begin(), MultipleRandomDimensionSelect::Begin(), SampleInitialization::Cluster(), DataDependentRandomInitializer::Initialize(), RandomAcolInitialization< columnsToAverage >::Initialize(), ObtainDistinctSamples(), GreedyPolicy< EnvironmentType >::Sample(), RandomSelection::Select(), and RandomPointSelection::Select().

38.37.3.24 RandInt() [2/2]

```
int mlpack::math::RandInt (
    const int lo,
    const int hiExclusive ) [inline]
```

Generates a uniform random integer.

Definition at line 106 of file random.hpp.

References randUniformDist.

38.37.3.25 RandNormal() [1/2]

```
double mlpack::math::RandNormal ( ) [inline]
```

Generates a normally distributed random number with mean 0 and variance 1.

Definition at line 115 of file random.hpp.

References randNormalDist.

Referenced by GaussianInitialization::Initialize().

38.37.3.26 RandNormal() [2/2]

```
double mlpack::math::RandNormal (
    const double mean,
    const double variance ) [inline]
```

Generates a normally distributed random number with specified mean and variance.

Parameters

<i>mean</i>	Mean of distribution.
<i>variance</i>	Variance of distribution.

Definition at line 127 of file random.hpp.

References randNormalDist.

38.37.3.27 Random() [1/2]

```
double mlpack::math::Random ( ) [inline]
```

Generates a uniform random number between 0 and 1.

Definition at line 71 of file random.hpp.

References randUniformDist.

Referenced by Pendulum::InitialSample(), ContinuousMountainCar::InitialSample(), MockCategoricalData(), RandBernoulli(), GreedyPolicy< EnvironmentType >::Sample(), and Acrobot::Torque().

38.37.3.28 Random() [2/2]

```
double mlpack::math::Random (
    const double lo,
    const double hi ) [inline]
```

Generates a uniform random number in the specified range.

Definition at line 79 of file random.hpp.

References randUniformDist.

38.37.3.29 RandomBasis()

```
void mlpack::math::RandomBasis (
    arma::mat & basis,
    const size_t d )
```

Create a random d-dimensional orthogonal basis, storing it in the given matrix.

Parameters

<i>basis</i>	Matrix to store basis in.
<i>d</i>	Desired number of dimensions in the basis.

38.37.3.30 RandomSeed()

```
void mlpack::math::RandomSeed (
    const size_t seed ) [inline]
```

Set the random seed used by the random functions (**Random()** (p. 418) and **RandInt()** (p. 417)).

The seed is casted to a 32-bit integer before being given to the random number generator, but a `size_t` is taken as a parameter for API consistency.

Parameters

<i>seed</i>	Seed for the random number generator.
-------------	---------------------------------------

Definition at line 40 of file random.hpp.

38.37.3.31 RandVector()

```
void mlpack::math::RandVector (
    arma::vec & v )
```

Overwrites a dimension-N vector to a random vector on the unit sphere in R^N .

38.37.3.32 RemoveRows()

```
void mlpack::math::RemoveRows (
    const arma::mat & input,
    const std::vector< size_t > & rowsToRemove,
    arma::mat & output )
```

Remove a certain set of rows in a matrix while copying to a second matrix.

Parameters

<i>input</i>	Input matrix to copy.
<i>rowsToRemove</i>	Vector containing indices of rows to be removed.
<i>output</i>	Matrix to copy non-removed rows into.

38.37.3.33 ShuffleData() [1/5]

```
void mlpack::math::ShuffleData (
    const MatType & inputPoints,
    const LabelsType & inputLabels,
    MatType & outputPoints,
    LabelsType & outputLabels,
    const std::enable_if_t<!arma::is_SpMat< MatType >::value > * = 0,
    const std::enable_if_t<!arma::is_Cube< MatType >::value > * = 0 )
```

Shuffle a dataset and associated labels (or responses).

It is expected that inputPoints and inputLabels have the same number of columns (so, be sure that inputLabels, if it is a vector, is a row vector).

Shuffled data will be output into outputPoints and outputLabels.

Definition at line 28 of file shuffle_data.hpp.

38.37.3.34 ShuffleData() [2/5]

```
void mlpack::math::ShuffleData (
    const MatType & inputPoints,
    const LabelsType & inputLabels,
    MatType & outputPoints,
    LabelsType & outputLabels,
    const std::enable_if_t< arma::is_SpMat< MatType >::value > * = 0,
    const std::enable_if_t<!arma::is_Cube< MatType >::value > * = 0 )
```

Shuffle a sparse dataset and associated labels (or responses).

It is expected that inputPoints and inputLabels have the same number of columns (so, be sure that inputLabels, if it is a vector, is a row vector).

Shuffled data will be output into outputPoints and outputLabels.

Definition at line 51 of file shuffle_data.hpp.

38.37.3.35 ShuffleData() [3/5]

```
void mlpack::math::ShuffleData (
    const MatType & inputPoints,
    const LabelsType & inputLabels,
    MatType & outputPoints,
    LabelsType & outputLabels,
    const std::enable_if_t<!arma::is_SpMat< MatType >::value > * = 0,
    const std::enable_if_t< arma::is_Cube< MatType >::value > * = 0,
    const std::enable_if_t< arma::is_Cube< LabelsType >::value > * = 0 )
```

Shuffle a cube-shaped dataset and associated labels (or responses) which are also cube-shaped.

It is expected that inputPoints and inputLabels have the same number of columns.

Shuffled data will be output into outputPoints and outputLabels.

Definition at line 103 of file shuffle_data.hpp.

38.37.3.36 ShuffleData() [4/5]

```
void mlpack::math::ShuffleData (
    const MatType & inputPoints,
    const LabelsType & inputLabels,
    const WeightsType & inputWeights,
    MatType & outputPoints,
    LabelsType & outputLabels,
    WeightsType & outputWeights,
    const std::enable_if_t<!arma::is_SpMat< MatType >::value > * = 0,
    const std::enable_if_t<!arma::is_Cube< MatType >::value > * = 0 )
```

Shuffle a dataset and associated labels (or responses) and weights.

It is expected that inputPoints and inputLabels and inputWeights have the same number of columns (so, be sure that inputLabels, if it is a vector, is a row vector).

Shuffled data will be output into outputPoints and outputLabels and outputWeights.

Definition at line 160 of file shuffle_data.hpp.

38.37.3.37 ShuffleData() [5/5]

```
void mlpack::math::ShuffleData (
    const MatType & inputPoints,
    const LabelsType & inputLabels,
    const WeightsType & inputWeights,
    MatType & outputPoints,
    LabelsType & outputLabels,
    WeightsType & outputWeights,
    const std::enable_if_t< arma::is_SpMat< MatType >::value > * = 0,
    const std::enable_if_t<!arma::is_Cube< MatType >::value > * = 0 )
```

Shuffle a sparse dataset and associated labels (or responses) and weights.

It is expected that inputPoints and inputLabels and inputWeights have the same number of columns (so, be sure that inputLabels, if it is a vector, is a row vector).

Shuffled data will be output into outputPoints and outputLabels and outputWeights.

Definition at line 188 of file shuffle_data.hpp.

38.37.3.38 Sign()

```
T mlpack::math::Sign (
    const T x )
```

Signum function.

Return 1 if $x > 0$; return 0 if $x = 0$; return -1 if $x < 0$. Return type are the same as input type.

Parameters

<i>x</i>	Number of any type.
----------	---------------------

Definition at line 135 of file `lin_alg.hpp`.

38.37.3.39 Smat()

```
void mlpack::math::Smat (
    const arma::vec & input,
    arma::mat & output )
```

The inverse of Svec.

That is, $\text{Smat}(\text{Svec}(A)) == A$.

Parameters

<i>input</i>	
<i>output</i>	A symmetric matrix

38.37.3.40 Svec() [1/2]

```
void mlpack::math::Svec (
    const arma::mat & input,
    arma::vec & output )
```

Upper triangular representation of a symmetric matrix, scaled such that, $\text{dot}(\text{Svec}(A), \text{Svec}(B)) == \text{dot}(A, B)$ for symmetric A, B .

Specifically,

$\text{Svec}(K) = [K_{11}, \sqrt{2} K_{12}, \dots, \sqrt{2} K_{1n}, K_{22}, \dots, \sqrt{2} K_{2n}, \dots, K_{nn}]^T$

Parameters

<i>input</i>	A symmetric matrix
<i>output</i>	

38.37.3.41 Svec() [2/2]

```
void mlpack::math::Svec (
    const arma::sp_mat & input,
    arma::sp_vec & output )
```

38.37.3.42 SvecIndex()

```
size_t mlpack::math::SvecIndex (
    size_t i,
    size_t j,
    size_t n ) [inline]
```

Return the index such that $A[i,j] == \text{factr}(i, j) * \text{svec}(A)[\text{pos}(i, j)]$, where $\text{factr}(i, j) = \sqrt{2}$ if $i \neq j$ and 1 otherwise.

Parameters

<i>i</i>	
<i>j</i>	
<i>n</i>	

38.37.3.43 SymKronId()

```
void mlpack::math::SymKronId (
    const arma::mat & A,
    arma::mat & op )
```

If *A* is a symmetric matrix, then SymKronId returns an operator *Op* such that.

$Op * \text{svec}(X) == \text{svec}(0.5 * (AX + XA))$

for every symmetric matrix *X*

Parameters

<i>A</i>	
<i>op</i>	

38.37.3.44 VectorPower()

```
void mlpack::math::VectorPower (
```



```
arma::vec & vec,  
const double power )
```

Auxiliary function to raise vector elements to a specific power.

The sign is ignored in the power operation and then re-added. Useful for eigenvalues.

38.37.3.45 WhitenUsingEig()

```
void mlpack::math::WhitenUsingEig (  
    const arma::mat & x,  
    arma::mat & xWhitened,  
    arma::mat & whiteningMatrix )
```

Whitens a matrix using the eigendecomposition of the covariance matrix.

Whitening means the covariance matrix of the result is the identity matrix.

38.37.3.46 WhitenUsingSVD()

```
void mlpack::math::WhitenUsingSVD (  
    const arma::mat & x,  
    arma::mat & xWhitened,  
    arma::mat & whiteningMatrix )
```

Whitens a matrix using the singular value decomposition of the covariance matrix.

Whitening means the covariance matrix of the result is the identity matrix.

38.37.4 Variable Documentation

38.37.4.1 randGen

```
MLPACK_EXPORT std::mt19937 randGen
```

MLPACK_EXPORT is required for global variables; it exports the symbols correctly on Windows.

38.37.4.2 randNormalDist

```
MLPACK_EXPORT std::normal_distribution randNormalDist
```

Referenced by RandNormal().

38.37.4.3 randUniformDist

MLPACK_EXPORT std::uniform_real_distribution randUniformDist

Referenced by RandInt(), and Random().

38.38 mlpack::matrix_completion Namespace Reference

Classes

- class **MatrixCompletion**

This class implements the popular nuclear norm minimization heuristic for matrix completion problems.

38.39 mlpack::meanshift Namespace Reference

Mean shift clustering.

Classes

- class **MeanShift**

This class implements mean shift clustering.

38.39.1 Detailed Description

Mean shift clustering.

38.40 mlpack::metric Namespace Reference

Classes

- class **IPMetric**

*The inner product metric, **IPMetric** (p. 1565), takes a given Mercer kernel (KernelType), and when **Evaluate()** (p. 1567) is called, returns the distance between the two points in kernel space:*

- class **LMetric**

The L_p metric for arbitrary integer p , with an option to take the root.

- class **MahalanobisDistance**

The Mahalanobis distance, which is essentially a stretched Euclidean distance.

Typedefs

- typedef **LMetric**< INT_MAX, false > **ChebyshevDistance**
The L-infinity distance.
- typedef **LMetric**< 2, true > **EuclideanDistance**
The Euclidean (L2) distance.
- typedef **LMetric**< 1, false > **ManhattanDistance**
The Manhattan (L1) distance.
- typedef **LMetric**< 2, false > **SquaredEuclideanDistance**
The squared Euclidean (L2) distance.

38.40.1 Typedef Documentation

38.40.1.1 ChebyshevDistance

```
typedef LMetric<INT_MAX, false> ChebyshevDistance
```

The L-infinity distance.

Definition at line 117 of file lmetric.hpp.

38.40.1.2 EuclideanDistance

```
typedef LMetric<2, true> EuclideanDistance
```

The Euclidean (L2) distance.

Definition at line 112 of file lmetric.hpp.

38.40.1.3 ManhattanDistance

```
typedef LMetric<1, false> ManhattanDistance
```

The Manhattan (L1) distance.

Definition at line 101 of file lmetric.hpp.

38.40.1.4 SquaredEuclideanDistance

```
typedef LMetric<2, false> SquaredEuclideanDistance
```

The squared Euclidean (L2) distance.

Note that this is not technically a metric! But it can sometimes be used when distances are required.

Definition at line 107 of file lmetric.hpp.

38.41 **mlpack::naive_bayes** Namespace Reference

The Naive Bayes Classifier.

Classes

- class **NaiveBayesClassifier**
The simple Naive Bayes classifier.

38.41.1 Detailed Description

The Naive Bayes Classifier.

38.42 **mlpack::nca** Namespace Reference

Neighborhood Components Analysis.

Classes

- class **NCA**
An implementation of Neighborhood Components Analysis, both a linear dimensionality reduction technique and a distance learning technique.
- class **SoftmaxErrorFunction**
The "softmax" stochastic neighbor assignment probability function.

38.42.1 Detailed Description

Neighborhood Components Analysis.

38.43 mlpack::neighbor Namespace Reference

Classes

- class **AlphaVisitor**
Exposes the Alpha() method of the given RAType.
- class **BiSearchVisitor**
***BiSearchVisitor** (p. 1591) executes a bichromatic neighbor search on the given NSType.*
- class **DeleteVisitor**
***DeleteVisitor** (p. 1595) deletes the given NSType instance.*
- class **DrusillaSelect**
- class **EpsilonVisitor**
***EpsilonVisitor** (p. 1600) exposes the Epsilon method of the given NSType.*
- class **FirstLeafExactVisitor**
Exposes the FirstLeafExact() method of the given RAType.
- class **FurthestNS**
*This class implements the necessary methods for the SortPolicy template parameter of the **NeighborSearch** (p. 1627) class.*
- class **LSHSearch**
*The **LSHSearch** (p. 1609) class; this class builds a hash on the reference set and uses this hash to compute the distance-approximate nearest-neighbors of the given queries.*
- class **MonoSearchVisitor**
***MonoSearchVisitor** (p. 1618) executes a monochromatic neighbor search on the given NSType.*
- class **NaiveVisitor**
***NaiveVisitor** (p. 1620) exposes the Naive() method of the given RAType.*
- class **NearestNS**
*This class implements the necessary methods for the SortPolicy template parameter of the **NeighborSearch** (p. 1627) class.*
- class **NeighborSearch**
*The **NeighborSearch** (p. 1627) class is a template class for performing distance-based neighbor searches.*
- class **NeighborSearchRules**
*The **NeighborSearchRules** (p. 1640) class is a template helper class used by **NeighborSearch** (p. 1627) class when performing distance-based neighbor searches.*
- class **NeighborSearchStat**
Extra data for each node in the tree.
- class **NSModel**
*The **NSModel** (p. 1657) class provides an easy way to serialize a model, abstracts away the different types of trees, and also reflects the **NeighborSearch** (p. 1627) API.*
- class **QDAFN**
- class **RAModel**
*The **RAModel** (p. 1669) class provides an abstraction for the **RASearch** (p. 1681) class, abstracting away the TreeType parameter and allowing it to be specified at runtime in this class.*
- class **RAQueryStat**
Extra data for each node in the tree.
- class **RASearch**
*The **RASearch** (p. 1681) class: This class provides a generic manner to perform rank-approximate search via random-sampling.*
- class **RASearchRules**

The **RASearchRules** (p. 1692) class is a template helper class used by **RASearch** (p. 1681) class when performing rank-approximate search via random-sampling.

- class **RAUtil**
- class **ReferenceSetVisitor**

ReferenceSetVisitor (p. 1701) exposes the `referenceSet` of the given `NSType`.

- class **SampleAtLeavesVisitor**

Exposes the `SampleAtLeaves()` method of the given `RAType`.

- class **SearchModeVisitor**

SearchModeVisitor (p. 1703) exposes the `SearchMode()` method of the given `NSType`.

- class **SingleModeVisitor**

Exposes the `SingleMode()` method of the given `RAType`.

- class **SingleSampleLimitVisitor**

Exposes the `SingleSampleLimit()` method of the given `RAType`.

- class **TauVisitor**

Exposes the `Tau()` method of the given `RAType`.

- class **TrainVisitor**

TrainVisitor (p. 1707) sets the reference set to a new reference set on the given `NSType`.

Typedefs

- `template<template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType = tree::SPTree> using DefeatistKNN = NeighborSearch< NearestNeighborSort, metric::EuclideanDistance, arma::mat, TreeType, TreeType< metric::EuclideanDistance, NeighborSearchStat< NearestNeighborSort >, arma::mat >::mat >::template DefeatistDualTreeTraverser, TreeType< metric::EuclideanDistance, NeighborSearchStat< NearestNeighborSort >, arma::mat >::template DefeatistSingleTreeTraverser >`

The **DefeatistKNN** class is the *k*-nearest-neighbors method considering defeatist search.

- using **FurthestNeighborSort = FurthestNS**
- typedef **NeighborSearch< FurthestNeighborSort, metric::EuclideanDistance > KFN**

The **KFN** class is the *k*-furthest-neighbors method.

- typedef **NeighborSearch< NearestNeighborSort, metric::EuclideanDistance > KNN**

The **KNN** class is the *k*-nearest-neighbors method.

- typedef **RASearch< FurthestNeighborSort > KRAFN**

The **KRAFN** class is the *k*-rank-approximate-furthest-neighbors method.

- typedef **RASearch KRANN**

The **KRANN** class is the *k*-rank-approximate-nearest-neighbors method.

- using **NearestNeighborSort = NearestNS**

- `template<typename SortPolicy, template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType> using NSType = NeighborSearch< SortPolicy, metric::EuclideanDistance, arma::mat, TreeType, TreeType< metric::EuclideanDistance, NeighborSearchStat< SortPolicy >, arma::mat >::template DualTreeTraverser >`

Alias template for euclidean neighbor search.

- `template<typename SortPolicy, template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType> using RAType = RASearch< SortPolicy, metric::EuclideanDistance, arma::mat, TreeType >`

Alias template for **RASearch** (p. 1681).

- typedef **DefeatistKNN< tree::SPTree > SpillKNN**

The **SpillKNN** class is the *k*-nearest-neighbors method considering defeatist search on *SPTree*.

Enumerations

- enum **NeighborSearchMode** {
NAIVE_MODE,
SINGLE_TREE_MODE,
DUAL_TREE_MODE,
GREEDY_SINGLE_TREE_MODE }

NeighborSearchMode represents the different neighbor search modes available.

Functions

- void **Unmap** (const arma::Mat< size_t > &neighbors, const arma::mat &distances, const std::vector< size_t > &referenceMap, const std::vector< size_t > &queryMap, arma::Mat< size_t > &neighborsOut, arma::mat &distancesOut, const bool squareRoot=false)

Assuming that the datasets have been mapped using the referenceMap and the queryMap (such as during kd-tree construction), unmap the columns of the distances and neighbors matrices into neighborsOut and distancesOut, and also unmap the entries in each row of neighbors.

- void **Unmap** (const arma::Mat< size_t > &neighbors, const arma::mat &distances, const std::vector< size_t > &referenceMap, arma::Mat< size_t > &neighborsOut, arma::mat &distancesOut, const bool squareRoot=false)

Assuming that the datasets have been mapped using referenceMap (such as during kd-tree construction), unmap the columns of the distances and neighbors matrices into neighborsOut and distancesOut, and also unmap the entries in each row of neighbors.

38.43.1 Typedef Documentation

38.43.1.1 DefeatistKNN

```
using DefeatistKNN = NeighborSearch< NearestNeighborSort, metric::EuclideanDistance, arma::mat, TreeType, TreeType< metric::EuclideanDistance, NeighborSearchStat< NearestNeighborSort>, arma::mat>::template DefeatistDualTreeTraverser, TreeType< metric::EuclideanDistance, NeighborSearchStat< NearestNeighborSort>, arma::mat>::template DefeatistSingleTreeTraverser>
```

The DefeatistKNN class is the k-nearest-neighbors method considering defeatist search.

It returns L2 distances (Euclidean distances) for each of the k nearest neighbors found.

Template Parameters

<i>TreeType</i>	The tree type to use; must adhere to the TreeType API, and implement Defeatist Traversers.
-----------------	--

Definition at line 60 of file typedef.hpp.

38.43.1.2 FurthestNeighborSort

```
using FurthestNeighborSort = FurthestNS
```

Definition at line 201 of file furthest_neighbor_sort.hpp.

38.43.1.3 KFN

```
typedef NeighborSearch< FurthestNeighborSort, metric::EuclideanDistance> KFN
```

The KFN class is the k-furthest-neighbors method.

It returns L2 distances (Euclidean distances) for each of the k furthest neighbors.

Definition at line 38 of file typedef.hpp.

38.43.1.4 KNN

```
typedef NeighborSearch< NearestNeighborSort, metric::EuclideanDistance> KNN
```

The KNN class is the k-nearest-neighbors method.

It returns L2 distances (Euclidean distances) for each of the k nearest neighbors.

Definition at line 32 of file typedef.hpp.

38.43.1.5 KRAFN

```
typedef RASearch< FurthestNeighborSort> KRAFN
```

The KRAFN class is the k-rank-approximate-farthest-neighbors method.

It returns L2 distances for each of the k rank-approximate farthest-neighbors.

The approximation is controlled with two parameters (see allkrann_main.cpp) which can be specified at search time. So the tree building is done only once while the search can be performed multiple times with different approximation levels.

Definition at line 47 of file ra_typedef.hpp.

38.43.1.6 KRANN

```
typedef RASearch KRANN
```

The KRANN class is the k-rank-approximate-nearest-neighbors method.

It returns L2 distances for each of the k rank-approximate nearest-neighbors.

The approximation is controlled with two parameters (see allkrann_main.cpp) which can be specified at search time. So the tree building is done only once while the search can be performed multiple times with different approximation levels.

Definition at line 36 of file ra_typedef.hpp.

38.43.1.7 NearestNeighborSort

```
using NearestNeighborSort = NearestNS
```

Definition at line 200 of file nearest_neighbor_sort.hpp.

38.43.1.8 NSType

```
using NSType = NeighborSearch<SortPolicy, metric::EuclideanDistance, arma::mat, TreeType,  
TreeType< metric::EuclideanDistance, NeighborSearchStat<SortPolicy>, arma::mat>::template Dual<  
TreeTraverser>
```

Alias template for euclidean neighbor search.

Definition at line 42 of file ns_model.hpp.

38.43.1.9 RAType

```
using RAType = RASearch<SortPolicy, metric::EuclideanDistance, arma::mat, TreeType>
```

Alias template for **RASearch** (p. 1681).

Definition at line 37 of file ra_model.hpp.

38.43.1.10 SpillKNN

```
typedef DefeatistKNN< tree::SPTree> SpillKNN
```

The SpillKNN class is the k-nearest-neighbors method considering defeatist search on SPTree.

It returns L2 distances (Euclidean distances) for each of the k nearest neighbors found.

Definition at line 67 of file typedef.hpp.

38.43.2 Enumeration Type Documentation

38.43.2.1 NeighborSearchMode

```
enum NeighborSearchMode
```

NeighborSearchMode represents the different neighbor search modes available.

Enumerator

NAIVE_MODE	
SINGLE_TREE_MODE	
DUAL_TREE_MODE	
GREEDY_SINGLE_TREE_MODE	

Definition at line 38 of file neighbor_search.hpp.

38.43.3 Function Documentation

38.43.3.1 Unmap() [1/2]

```
void mlpack::neighbor::Unmap (
    const arma::Mat< size_t > & neighbors,
    const arma::mat & distances,
    const std::vector< size_t > & referenceMap,
    const std::vector< size_t > & queryMap,
    arma::Mat< size_t > & neighborsOut,
    arma::mat & distancesOut,
    const bool squareRoot = false )
```

Assuming that the datasets have been mapped using the `referenceMap` and the `queryMap` (such as during kd-tree construction), unmap the columns of the distances and neighbors matrices into `neighborsOut` and `distancesOut`, and also unmap the entries in each row of neighbors.

This is useful for the dual-tree case.

Parameters

<i>neighbors</i>	Matrix of neighbors resulting from neighbor search.
<i>distances</i>	Matrix of distances resulting from neighbor search.
<i>referenceMap</i>	Mapping of reference set to old points.
<i>queryMap</i>	Mapping of query set to old points.
<i>neighborsOut</i>	Matrix to store unmapped neighbors into.
<i>distancesOut</i>	Matrix to store unmapped distances into.
<i>squareRoot</i>	If true, take the square root of the distances.

38.43.3.2 Unmap() [2/2]

```
void mlpack::neighbor::Unmap (
    const arma::Mat< size_t > & neighbors,
    const arma::mat & distances,
    const std::vector< size_t > & referenceMap,
    arma::Mat< size_t > & neighborsOut,
    arma::mat & distancesOut,
    const bool squareRoot = false )
```

Assuming that the datasets have been mapped using `referenceMap` (such as during kd-tree construction), unmap the columns of the distances and neighbors matrices into `neighborsOut` and `distancesOut`, and also unmap the entries in each row of neighbors.

This is useful for the single-tree case.

Parameters

<i>neighbors</i>	Matrix of neighbors resulting from neighbor search.
<i>distances</i>	Matrix of distances resulting from neighbor search.
<i>referenceMap</i>	Mapping of reference set to old points.
<i>neighborsOut</i>	Matrix to store unmapped neighbors into.
<i>distancesOut</i>	Matrix to store unmapped distances into.
<i>squareRoot</i>	If true, take the square root of the distances.

38.44 mlpack::nn Namespace Reference

Classes

- class **SparseAutoencoder**
A sparse autoencoder is a neural network whose aim to learn compressed representations of the data, typically for dimensionality reduction, with a constraint on the activity of the neurons in the network.
- class **SparseAutoencoderFunction**
This is a class for the sparse autoencoder objective function.

Functions

- void **MaximalInputs** (const arma::mat ¶meters, arma::mat &output)
Given a parameters matrix from an autoencoder, maximize the hidden units of the parameters, storing the maximal inputs in the given output matrix.
- void **NormalizeColByMax** (const arma::mat &input, arma::mat &output)
Normalize each column of the input matrix by its maximum value, if that maximum value is not zero.

38.44.1 Function Documentation

38.44.1.1 MaximalInputs()

```
void mlpack::nn::MaximalInputs (
    const arma::mat & parameters,
    arma::mat & output )
```

Given a parameters matrix from an autoencoder, maximize the hidden units of the parameters, storing the maximal inputs in the given output matrix.

Details can be found on the 'Visualizing a Trained Autoencoder' page of the Stanford UFLDL tutorial:

http://deeplearning.stanford.edu/wiki/index.php/Main_Page

This function is based on the implementation (display_network.m) from the "Exercise: Sparse Autoencoder" page of the UFLDL tutorial:

http://deeplearning.stanford.edu/wiki/index.php/Exercise:Sparse_Autoencoder

Example usage of this function can be seen below. Note that this function can work with the ColumnsToBlocks class in order to reshape the maximal inputs for visualization, as in the UFLDL tutorial. The code below demonstrates this.

```
arma::mat data; // Data matrix.
const size_t vSize = 64; // Size of visible layer, depends on the data.
const size_t hSize = 25; // Size of hidden layer, depends on requirements.

const size_t numBasis = 5; // Parameter required for L-BFGS algorithm.
const size_t numIterations = 100; // Maximum number of iterations.

// Use an instantiated optimizer for the training.
SparseAutoencoder<L_BFGS> encoder(data, vSize, hSize);

arma::mat maximalInput; // Store the features learned by sparse autoencoder
mlpack::nn::MaximalInputs(encoder.Parameters(), maximalInput);

arma::mat outputs;
const bool scale = true;

ColumnsToBlocks ctb(5,5);
arma::mat output;
ctb.Transform(maximalInput, output);
// Save the output as PGM, for visualization.
output.save(fileName, arma::pgm_binary);
```

Precondition

Layout of parameters

The layout of the parameters matrix should be same as following

```
//      vSize  1
//      |      |
// hSize|  w1  |bl|
//      |_____|_|
//      |      |
// hSize|  w2'  |  |
//      |_____|_|
//      1|  b2'  |  |
```

Also, the square root of vSize must be an integer (i.e. vSize must be a perfect square).

Parameters

<i>parameters</i>	The parameters of the autoencoder.
<i>output</i>	Matrix to store the maximal inputs in.

38.44.1.2 NormalizeColByMax()

```
void mlpack::nn::NormalizeColByMax (
    const arma::mat & input,
    arma::mat & output )
```

Normalize each column of the input matrix by its maximum value, if that maximum value is not zero.

Parameters

<i>input</i>	The input data to normalize.
<i>output</i>	A matrix to store the input data in after normalization.

38.45 mlpack::pca Namespace Reference**Classes**

- class **ExactSVDPolicy**
Implementation of the exact SVD policy.
- class **PCA**
*This class implements principal components analysis (**PCA** (p. 1723)).*
- class **QUICSVDPolicy**

Implementation of the QUIC-SVD policy.

- class **RandomizedBlockKrylovSVDPolicy**

Implementation of the randomized block krylov SVD policy.

- class **RandomizedSVDPolicy**

Implementation of the randomized SVD policy.

38.46 mlpack::perceptron Namespace Reference

Classes

- class **Perceptron**

This class implements a simple perceptron (i.e., a single layer neural network).

- class **RandomInitialization**

This class is used to initialize weights for the weightVectors matrix in a random manner.

- class **SimpleWeightUpdate**

- class **ZeroInitialization**

This class is used to initialize the matrix weightVectors to zero.

38.47 mlpack::radical Namespace Reference

Classes

- class **Radical**

An implementation of RADICAL, an algorithm for independent component analysis (ICA).

Functions

- void **WhitenFeatureMajorMatrix** (const arma::mat &matX, arma::mat &matXWhitened, arma::mat &mat←Whitening)

38.47.1 Function Documentation

38.47.1.1 WhitenFeatureMajorMatrix()

```
void mlpack::radical::WhitenFeatureMajorMatrix (
    const arma::mat & matX,
    arma::mat & matXWhitened,
    arma::mat & matWhitening )
```

Referenced by Radical::Sweeps().

38.48 mlpack::range Namespace Reference

Range-search routines.

Classes

- class **BiSearchVisitor**
***BiSearchVisitor** (p. 1748) executes a bichromatic range search on the given RSType.*
- class **DeleteVisitor**
***DeleteVisitor** (p. 1751) deletes the given RSType instance.*
- class **MonoSearchVisitor**
***MonoSearchVisitor** (p. 1752) executes a monochromatic range search on the given RSType.*
- class **NaiveVisitor**
***NaiveVisitor** (p. 1753) exposes the Naive() method of the given RSType.*
- class **RangeSearch**
*The **RangeSearch** (p. 1754) class is a template class for performing range searches.*
- class **RangeSearchRules**
*The **RangeSearchRules** (p. 1764) class is a template helper class used by **RangeSearch** (p. 1754) class when performing range searches.*
- class **RangeSearchStat**
*Statistic class for **RangeSearch** (p. 1754), to be set to the StatisticType of the tree type that range search is being performed with.*
- class **ReferenceSetVisitor**
***ReferenceSetVisitor** (p. 1771) exposes the referenceSet of the given RSType.*
- class **RSModel**
- class **SingleModeVisitor**
***SingleModeVisitor** (p. 1779) exposes the SingleMode() method of the given RSType.*
- class **TrainVisitor**
***TrainVisitor** (p. 1780) sets the reference set to a new reference set on the given RSType.*

Typedefs

- template<template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType>
 using **RSType** = **RangeSearch**< **metric::EuclideanDistance**, arma::mat, TreeType >
Alias template for Range Search.

38.48.1 Detailed Description

Range-search routines.

38.48.2 Typedef Documentation

38.48.2.1 RSType

```
using RSType = RangeSearch< metric::EuclideanDistance, arma::mat, TreeType>
```

Alias template for Range Search.

Definition at line 34 of file rs_model.hpp.

38.49 mlpack::regression Namespace Reference

Regression methods.

Classes

- class **LARS**
*An implementation of **LARS** (p. 1783), a stage-wise homotopy-based algorithm for l1-regularized linear regression (LASSO) and l1+l2 regularized linear regression (Elastic Net).*
- class **LinearRegression**
A simple linear regression algorithm using ordinary least squares.
- class **LogisticRegression**
*The **LogisticRegression** (p. 1795) class implements an L2-regularized logistic regression model, and supports training with multiple optimizers and classification.*
- class **LogisticRegressionFunction**
The log-likelihood function for the logistic regression objective function.
- class **SoftmaxRegression**
Softmax Regression is a classifier which can be used for classification when the data available can take two or more class values.
- class **SoftmaxRegressionFunction**

38.49.1 Detailed Description

Regression methods.

38.50 mlpack::rl Namespace Reference

Classes

- class **Acrobot**
*Implementation of **Acrobot** (p. 1825) game.*
- class **AggregatedPolicy**
- class **AsyncLearning**
Wrapper of various asynchronous learning algorithms, e.g.
- class **CartPole**

- Implementation of Cart Pole task.*
- class **ContinuousMountainCar**
 - Implementation of Continuous Mountain Car task.*
- class **GreedyPolicy**
 - Implementation for epsilon greedy policy.*
- class **MountainCar**
 - Implementation of Mountain Car task.*
- class **NStepQLearningWorker**
 - Forward declaration of **NStepQLearningWorker** (p. 1868).*
- class **OneStepQLearningWorker**
 - Forward declaration of **OneStepQLearningWorker** (p. 1871).*
- class **OneStepSarsaWorker**
 - Forward declaration of **OneStepSarsaWorker** (p. 1875).*
- class **Pendulum**
 - Implementation of **Pendulum** (p. 1878) task.*
- class **QLearning**
 - Implementation of various Q-Learning algorithms, such as DQN, double DQN.*
- class **RandomReplay**
 - Implementation of random experience replay.*
- class **RewardClipping**
 - Interface for clipping the reward to some value between the specified maximum and minimum value (Clipping here is implemented as $g_{clipped} = \max(g_{min}, \min(g_{min}, g))$.)*
- class **TrainingConfig**

Typedefs

- typedef **Acrobot Acrobot**
 - Add an alias for backward compatibility.*
- template<typename EnvironmentType , typename NetworkType , typename UpdaterType , typename PolicyType >
 using **NStepQLearning = AsyncLearning< NStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >, EnvironmentType, NetworkType, UpdaterType, PolicyType >**
 - Convenient typedef for async n step q-learning.*
- template<typename EnvironmentType , typename NetworkType , typename UpdaterType , typename PolicyType >
 using **OneStepQLearning = AsyncLearning< OneStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >, EnvironmentType, NetworkType, UpdaterType, PolicyType >**
 - Convenient typedef for async one step q-learning.*
- template<typename EnvironmentType , typename NetworkType , typename UpdaterType , typename PolicyType >
 using **OneStepSarsa = AsyncLearning< OneStepSarsaWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >, EnvironmentType, NetworkType, UpdaterType, PolicyType >**
 - Convenient typedef for async one step Sarsa.*

38.50.1 Typedef Documentation

38.50.1.1 Acrobat

```
typedef Acrobot Acrobat
```

Add an alias for backward compatibility.

Definition at line 357 of file acrobot.hpp.

38.50.1.2 NStepQLearning

```
using NStepQLearning = AsyncLearning< NStepQLearningWorker<EnvironmentType, NetworkType, UpdaterType, PolicyType>, EnvironmentType, NetworkType, UpdaterType, PolicyType>
```

Convenient typedef for async n step q-learning.

Template Parameters

<i>EnvironmentType</i>	The type of the reinforcement learning task.
<i>NetworkType</i>	The type of the network model.
<i>UpdaterType</i>	The type of the optimizer.
<i>PolicyType</i>	The type of the behavior policy.

Definition at line 233 of file async_learning.hpp.

38.50.1.3 OneStepQLearning

```
using OneStepQLearning = AsyncLearning< OneStepQLearningWorker<EnvironmentType, NetworkType, UpdaterType, PolicyType>, EnvironmentType, NetworkType, UpdaterType, PolicyType>
```

Convenient typedef for async one step q-learning.

Template Parameters

<i>EnvironmentType</i>	The type of the reinforcement learning task.
<i>NetworkType</i>	The type of the network model.
<i>UpdaterType</i>	The type of the optimizer.
<i>PolicyType</i>	The type of the behavior policy.

Definition at line 197 of file async_learning.hpp.

38.50.1.4 OneStepSarsa

```
using OneStepSarsa = AsyncLearning< OneStepSarsaWorker<EnvironmentType, NetworkType, UpdaterType, PolicyType>, EnvironmentType, NetworkType, UpdaterType, PolicyType>
```

Convenient typedef for async one step Sarsa.

Template Parameters

<i>EnvironmentType</i>	The type of the reinforcement learning task.
<i>NetworkType</i>	The type of the network model.
<i>UpdaterType</i>	The type of the optimizer.
<i>PolicyType</i>	The type of the behavior policy.

Definition at line 215 of file `async_learning.hpp`.

38.51 mlpack::sfinae Namespace Reference

Classes

- struct **MethodFormDetector**
- struct **MethodFormDetector**< **Class**, **MethodForm**, 0 >
- struct **MethodFormDetector**< **Class**, **MethodForm**, 1 >
- struct **MethodFormDetector**< **Class**, **MethodForm**, 2 >
- struct **MethodFormDetector**< **Class**, **MethodForm**, 3 >
- struct **MethodFormDetector**< **Class**, **MethodForm**, 4 >
- struct **MethodFormDetector**< **Class**, **MethodForm**, 5 >
- struct **MethodFormDetector**< **Class**, **MethodForm**, 6 >
- struct **MethodFormDetector**< **Class**, **MethodForm**, 7 >
- struct **SigCheck**

Utility struct for checking signatures.

38.52 mlpack::sparse_coding Namespace Reference

Classes

- class **DataDependentRandomInitializer**
*A data-dependent random dictionary initializer for **SparseCoding** (p. 1916).*
- class **NothingInitializer**
*A DictionaryInitializer for **SparseCoding** (p. 1916) which does not initialize anything; it is useful for when the dictionary is already known and will be set with **SparseCoding::Dictionary()** (p. 1920).*
- class **RandomInitializer**
*A DictionaryInitializer for use with the **SparseCoding** (p. 1916) class.*
- class **SparseCoding**
An implementation of Sparse Coding with Dictionary Learning that achieves sparsity via an l_1 -norm regularizer on the codes (LASSO) or an (l_1+l_2) -norm regularizer on the codes (the Elastic Net).

38.53 mlpack::svd Namespace Reference

Classes

- class **BiasSVD**
Bias SVD is an improvement on Regularized SVD which is a matrix factorization techniques.
- class **BiasSVDFunction**
*This class contains methods which are used to calculate the cost of **BiasSVD** (p. 1925)'s objective function, to calculate gradient of parameters with respect to the objective function, etc.*
- class **QUIC_SVD**
QUIC-SVD is a matrix factorization technique, which operates in a subspace such that A's approximation in that subspace has minimum error(A being the data matrix).
- class **RandomizedBlockKrylovSVD**
Randomized block krylov SVD is a matrix factorization that is based on randomized matrix approximation techniques, developed in in "Randomized Block Krylov Methods for Stronger and Faster Approximate Singular Value Decomposition".
- class **RandomizedSVD**
Randomized SVD is a matrix factorization that is based on randomized matrix approximation techniques, developed in in "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions".
- class **RegularizedSVD**
Regularized SVD is a matrix factorization technique that seeks to reduce the error on the training set, that is on the examples for which the ratings have been provided by the users.
- class **RegularizedSVDFunction**
The data is stored in a matrix of type MatType, so that this class can be used with both dense and sparse matrix types.
- class **SVDPlusPlus**
SVD++ is a matrix decomposition technique used in collaborative filtering.
- class **SVDPlusPlusFunction**
This class contains methods which are used to calculate the cost of SVD++'s objective function, to calculate gradient of parameters with respect to the objective function, etc.

38.54 mlpack::svm Namespace Reference

Classes

- class **LinearSVM**
*The **LinearSVM** (p. 1959) class implements an L2-regularized support vector machine model, and supports training with multiple optimizers and classification.*
- class **LinearSVMFunction**
The hinge loss function for the linear SVM objective function.

38.55 mlpack::tree Namespace Reference

Trees and tree-building procedures.

Namespaces

- **enumerate**
- **split**

Classes

- class **AllCategoricalSplit**
*The **AllCategoricalSplit** (p. 1981) is a splitting function that will split categorical features into many children: one child for each category.*
- class **AllDimensionSelect**
This dimension selection policy allows any dimension to be selected for splitting.
- class **AxisParallelProjVector**
***AxisParallelProjVector** (p. 1986) defines an axis-parallel projection vector.*
- class **BestBinaryNumericSplit**
*The **BestBinaryNumericSplit** (p. 1989) is a splitting function for decision trees that will exhaustively search a numeric dimension for the best binary split.*
- class **BinaryNumericSplit**
*The **BinaryNumericSplit** (p. 1992) class implements the numeric feature splitting strategy devised by Gama, Rocha, and Medas in the following paper:*
- class **BinaryNumericSplitInfo**
- class **BinarySpaceTree**
A binary space partitioning tree, such as a KD-tree or a ball tree.
- class **CategoricalSplitInfo**
- class **CompareCosineNode**
- class **CosineTree**
- class **CoverTree**
A cover tree is a tree specifically designed to speed up nearest-neighbor computation in high-dimensional spaces.
- class **DecisionTree**
This class implements a generic decision tree learner.
- class **DiscreteHilbertValue**
*The **DiscreteHilbertValue** (p. 2079) class stores Hilbert values for all of the points in a **RectangleTree** (p. 2207) node, and calculates Hilbert values for new points.*
- class **EmptyStatistic**
Empty statistic if you are not interested in storing statistics in your tree.
- class **ExampleTree**
This is not an actual space tree but instead an example tree that exists to show and document all the functions that mlpack trees must implement.
- class **FirstPointIsRoot**
*This class is meant to be used as a choice for the policy class RootPointPolicy of the **CoverTree** (p. 2040) class.*
- class **GiniGain**
The Gini gain, a measure of set purity usable as a fitness function (FitnessFunction) for decision trees.
- class **GiniImpurity**
- class **GreedySingleTreeTraverser**
- class **HilbertRTreeAuxiliaryInformation**
- class **HilbertRTreeDescentHeuristic**
This class chooses the best child of a node in a Hilbert R tree when inserting a new point.
- class **HilbertRTreeSplit**

The splitting procedure for the Hilbert R tree.

- class **HoeffdingCategoricalSplit**

This is the standard Hoeffding-bound categorical feature proposed in the paper below:

- class **HoeffdingNumericSplit**

*The **HoeffdingNumericSplit** (p. 2108) class implements the numeric feature splitting strategy alluded to by Domingos and Hulten in the following paper:*

- class **HoeffdingTree**

*The **HoeffdingTree** (p. 2113) object represents all of the necessary information for a Hoeffding-bound-based decision tree.*

- class **HoeffdingTreeModel**

This class is a serializable Hoeffding tree model that can hold four different types of Hoeffding trees.

- class **HyperplaneBase**

***HyperplaneBase** (p. 2133) defines a splitting hyperplane based on a projection vector and projection value.*

- class **InformationGain**

The standard information gain criterion, used for calculating gain in decision trees.

- struct **IsSpillTree**

- struct **IsSpillTree**< **tree::SpillTree**< **MetricType**, **StatisticType**, **MatType**, **HyperplaneType**, **SplitType** > >

- class **MeanSpaceSplit**

- class **MeanSplit**

A binary space partitioning tree node is split into its left and right child.

- class **MidpointSpaceSplit**

- class **MidpointSplit**

A binary space partitioning tree node is split into its left and right child.

- class **MinimalCoverageSweep**

*The **MinimalCoverageSweep** (p. 2151) class finds a partition along which we can split a node according to the coverage of two resulting nodes.*

- class **MinimalSplitsNumberSweep**

*The **MinimalSplitsNumberSweep** (p. 2155) class finds a partition along which we can split a node according to the number of required splits of the node.*

- class **MultipleRandomDimensionSelect**

This dimension selection policy allows the selection from a few random dimensions.

- class **NoAuxiliaryInformation**

- class **NumericSplitInfo**

- class **Octree**

- class **ProjVector**

***ProjVector** (p. 2190) defines a general projection vector (not necessarily axis-parallel).*

- struct **QueueFrame**

- class **RandomDimensionSelect**

This dimension selection policy only selects one single random dimension.

- class **RandomForest**

- class **RectangleTree**

A rectangle type tree tree, such as an R-tree or X-tree.

- class **RPlusPlusTreeAuxiliaryInformation**

- class **RPlusPlusTreeDescentHeuristic**

- class **RPlusPlusTreeSplitPolicy**

*The **RPlusPlusTreeSplitPolicy** (p. 2247) helps to determine the subtree into which we should insert a child of an intermediate node that is being split.*

- class **RPlusTreeDescentHeuristic**

- class **RPlusTreeSplit**

The **RPlusTreeSplit** (p. 2251) class performs the split process of a node on overflow.

- class **RPlusTreeSplitPolicy**

The **RPlusPlusTreeSplitPolicy** (p. 2247) helps to determine the subtree into which we should insert a child of an intermediate node that is being split.

- class **RPTreeMaxSplit**

This class splits a node by a random hyperplane.

- class **RPTreeMeanSplit**

This class splits a binary space tree.

- class **RStarTreeDescentHeuristic**

When descending a **RectangleTree** (p. 2207) to insert a point, we need to have a way to choose a child node when the point isn't enclosed by any of them.

- class **RStarTreeSplit**

A Rectangle Tree has new points inserted at the bottom.

- class **RTreeDescentHeuristic**

When descending a **RectangleTree** (p. 2207) to insert a point, we need to have a way to choose a child node when the point isn't enclosed by any of them.

- class **RTreeSplit**

A Rectangle Tree has new points inserted at the bottom.

- class **SpaceSplit**

- class **SpillTree**

A hybrid spill tree is a variant of binary space trees in which the children of a node can "spill over" each other, and contain shared datapoints.

- class **TraversalInfo**

The **TraversalInfo** (p. 2299) class holds traversal information which is used in dual-tree (and single-tree) traversals.

- class **TreeTraits**

The **TreeTraits** (p. 2302) class provides compile-time information on the characteristics of a given tree type.

- class **TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::BallBound, SplitType > >**

This is a specialization of the **TreeType** class to the **BallTree** tree type.

- class **TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::CellBound, SplitType > >**

This is a specialization of the **TreeType** class to the **UBTree** tree type.

- class **TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::HollowBallBound, SplitType > >**

This is a specialization of the **TreeType** class to an arbitrary tree with **HollowBallBound** (currently only the vantage point tree is supported).

- class **TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, RPTreeMaxSplit > >**

This is a specialization of the **TreeType** class to the max-split random projection tree.

- class **TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, RPTreeMeanSplit > >**

This is a specialization of the **TreeType** class to the mean-split random projection tree.

- class **TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > >**

This is a specialization of the **TreeTraits** (p. 2302) class to the **BinarySpaceTree** (p. 1998) tree type.

- class **TreeTraits< CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > >**

The specialization of the **TreeTraits** (p. 2302) class for the **CoverTree** (p. 2040) tree type.

- class **TreeTraits< Octree< MetricType, StatisticType, MatType > >**

This is a specialization of the **TreeTraits** (p. 2302) class to the **Octree** (p. 2167) tree type.

- class **TreeTraits**< **RectangleTree**< **MetricType**, **StatisticType**, **MatType**, **RPlusTreeSplit**< **SplitPolicyType**, **SweepType** >, **DescentType**, **AuxiliaryInformationType** > >
Since the R+/R++ tree can not have overlapping children, we should define traits for the R+/R++ tree.
- class **TreeTraits**< **RectangleTree**< **MetricType**, **StatisticType**, **MatType**, **SplitType**, **DescentType**, **AuxiliaryInformationType** > >
*This is a specialization of the TreeType class to the **RectangleTree** (p. 2207) tree type.*
- class **TreeTraits**< **SpillTree**< **MetricType**, **StatisticType**, **MatType**, **HyperplaneType**, **SplitType** > >
*This is a specialization of the TreeType class to the **SpillTree** (p. 2274) tree type.*
- class **UBTreeSplit**
Split a node into two parts according to the median address of points contained in the node.
- class **VantagePointSplit**
The class splits a binary space partitioning tree node according to the median distance to the vantage point.
- class **XTreeAuxiliaryInformation**
*The **XTreeAuxiliaryInformation** (p. 2338) class provides information specific to X trees for each node in a **RectangleTree** (p. 2207).*
- class **XTreeSplit**
A Rectangle Tree has new points inserted at the bottom.

Typedefs

- template<typename MetricType >
using **AxisOrthogonalHyperplane** = **HyperplaneBase**< **bound::HRectBound**< MetricType >, **Axis**< **ParallelProjVector** >
AxisOrthogonalHyperplane represents a hyperplane orthogonal to an axis.
- template<typename MetricType , typename StatisticType , typename MatType >
using **BallTree** = **BinarySpaceTree**< MetricType, StatisticType, MatType, **bound::BallBound**, **MidpointSplit** >
A midpoint-split ball tree.
- template<typename FitnessFunction >
using **BinaryDoubleNumericSplit** = **BinaryNumericSplit**< FitnessFunction, double >
- typedef boost::heap::priority_queue< **CosineTree** *, boost::heap::compare< **CompareCosineNode** > > **CosineNodeQueue**
- template<typename FitnessFunction = GiniGain, template< typename > class NumericSplitType = BestBinaryNumericSplit, template< typename > class CategoricalSplitType = AllCategoricalSplit, typename DimensionSelectType = AllDimensionSelect, typename ElemType = double>
using **DecisionStump** = **DecisionTree**< FitnessFunction, NumericSplitType, CategoricalSplitType, DimensionSelectType, ElemType, false >
Convenience typedef for decision stumps (single level decision trees).
- template<typename TreeType >
using **DiscreteHilbertRTreeAuxiliaryInformation** = **HilbertRTreeAuxiliaryInformation**< TreeType, **DiscreteHilbertValue** >
The Hilbert R-tree, a variant of the R tree with an ordering along the Hilbert curve.
- template<typename MetricType , typename StatisticType , typename MatType >
using **HilbertRTree** = **RectangleTree**< MetricType, StatisticType, MatType, **HilbertRTreeSplit**< 2 >, **Hilbert**< **RTreeDescentHeuristic**, **DiscreteHilbertRTreeAuxiliaryInformation** >
- template<typename FitnessFunction >
using **HoeffdingDoubleNumericSplit** = **HoeffdingNumericSplit**< FitnessFunction, double >
Convenience typedef.
- typedef StreamingDecisionTree< **HoeffdingTree**<> > **HoeffdingTreeType**

- `template<typename MetricType >`
`using Hyperplane = HyperplaneBase< bound::BallBound< MetricType >, ProjVector >`
Hyperplane represents a general hyperplane (not necessarily axis-orthogonal).
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using KDTree = BinarySpaceTree< MetricType, StatisticType, MatType, bound::HRectBound, MidpointSplit`
`>`
The standard midpoint-split kd-tree.
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using MaxRPTree = BinarySpaceTree< MetricType, StatisticType, MatType, bound::HRectBound, RPTree↵`
`MaxSplit >`
A max-split random projection tree.
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using MeanSplitBallTree = BinarySpaceTree< MetricType, StatisticType, MatType, bound::BallBound,`
`MeanSplit >`
A mean-split ball tree.
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using MeanSplitKDTree = BinarySpaceTree< MetricType, StatisticType, MatType, bound::HRectBound,`
`MeanSplit >`
A mean-split kd-tree.
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using MeanSPTree = SpillTree< MetricType, StatisticType, MatType, AxisOrthogonalHyperplane, Mean↵`
`SpaceSplit >`
A mean-split hybrid spill tree.
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using NonOrtMeanSPTree = SpillTree< MetricType, StatisticType, MatType, Hyperplane, MeanSpaceSplit`
`>`
A mean-split hybrid spill tree considering general splitting hyperplanes (not necessarily axis-orthogonal).
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using NonOrtSPTree = SpillTree< MetricType, StatisticType, MatType, Hyperplane, MidpointSpaceSplit >`
A hybrid spill tree considering general splitting hyperplanes (not necessarily axis-orthogonal).
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using RPlusPlusTree = RectangleTree< MetricType, StatisticType, MatType, RPlusTreeSplit< RPlus↵`
`PlusTreeSplitPolicy, MinimalSplitsNumberSweep >, RPlusPlusTreeDescentHeuristic, RPlusPlusTree↵`
`AuxiliaryInformation >`
The R++ tree, a variant of the R+ tree with maximum bounding rectangles.
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using RPlusTree = RectangleTree< MetricType, StatisticType, MatType, RPlusTreeSplit< RPlusTreeSplit↵`
`Policy, MinimalCoverageSweep >, RPlusTreeDescentHeuristic, NoAuxiliaryInformation >`
The R+ tree, a variant of the R tree that avoids overlapping rectangles.
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using RPTree = BinarySpaceTree< MetricType, StatisticType, MatType, bound::HRectBound, RPTree↵`
`MeanSplit >`
A mean-split random projection tree.
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using RStarTree = RectangleTree< MetricType, StatisticType, MatType, RStarTreeSplit, RStarTree↵`
`DescentHeuristic, NoAuxiliaryInformation >`
The R-tree, a more recent variant of the R tree.*
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using RTree = RectangleTree< MetricType, StatisticType, MatType, RTreeSplit, RTreeDescentHeuristic,`
`NoAuxiliaryInformation >`

An implementation of the R tree that satisfies the TreeType policy API.

- `template<typename MetricType , typename StatisticType , typename MatType >`
`using SPTree = SpillTree< MetricType, StatisticType, MatType, AxisOrthogonalHyperplane, Midpoint↵`
`SpaceSplit >`

The hybrid spill tree.

- `template<typename MetricType , typename StatisticType , typename MatType >`
`using StandardCoverTree = CoverTree< MetricType, StatisticType, MatType, FirstPointsRoot >`

The standard cover tree, as detailed in the original cover tree paper:

- `template<typename MetricType , typename StatisticType , typename MatType >`
`using UBTree = BinarySpaceTree< MetricType, StatisticType, MatType, bound::CellBound, UBTreeSplit >`

The Universal B-tree.

- `template<typename MetricType , typename StatisticType , typename MatType >`
`using VPTree = BinarySpaceTree< MetricType, StatisticType, MatType, bound::HollowBallBound, VP↵`
`TreeSplit >`

- `template<typename BoundType , typename MatType = arma::mat>`
`using VPTreeSplit = VantagePointSplit< BoundType, MatType, 100 >`

The vantage point tree (which is also called the metric tree).

- `template<typename MetricType , typename StatisticType , typename MatType >`
`using XTree = RectangleTree< MetricType, StatisticType, MatType, XTreeSplit, RTreeDescentHeuristic,`
`XTreeAuxiliaryInformation >`

The X-tree, a variant of the R tree with supernodes.

Functions

- `template<bool UseWeights, typename MatType , typename LabelsType , typename WeightsType >`
`void Bootstrap (const MatType &dataset, const LabelsType &labels, const WeightsType &weights, MatType`
`&bootstrapDataset, LabelsType &bootstrapLabels, WeightsType &bootstrapWeights)`

Given a dataset, create another dataset via bootstrap sampling, with labels.

- `template<class TreeType , class Walker >`
`void EnumerateTree (TreeType *tree, Walker &walker)`

Traverses all nodes of the tree, including the inner ones.

Variables

- `const double MAX_OVERLAP = 0.2`

The X-tree paper says that a maximum allowable overlap of 20% works well.

38.55.1 Detailed Description

Trees and tree-building procedures.

38.55.2 Typedef Documentation

38.55.2.1 AxisOrthogonalHyperplane

```
using AxisOrthogonalHyperplane = HyperplaneBase< bound::HRectBound<MetricType>, AxisParallel↵
ProjVector>
```

AxisOrthogonalHyperplane represents a hyperplane orthogonal to an axis.

Definition at line 145 of file hyperplane.hpp.

38.55.2.2 BallTree

```
using BallTree = BinarySpaceTree<MetricType, StatisticType, MatType, bound::BallBound, Midpoint↵
Split>
```

A midpoint-split ball tree.

This tree holds its points only in the leaves, similar to the KDTree and MeanSplitKDTree. However, the bounding shape of each node is a ball, not a hyper-rectangle. This can make the ball tree advantageous in some higher-dimensional situations and for some datasets. The tree construction algorithm here is the same as Omohundro's 'K-d construction algorithm', except the splitting value is the midpoint, not the median. This can result in trees that better reflect the data, although they may be unbalanced.

```
@techreport{omohundro1989five,
  author={S.M. Omohundro},
  title={Five balltree construction algorithms},
  year={1989},
  institution={University of California, Berkeley International Computer
    Science Institute Technical Reports},
  number={TR-89-063}
}
```

This template typedef satisfies the TreeType policy API.

See also

The TreeType policy in mpack (p.179), **BinarySpaceTree** (p.1998), **KDTree** (p.453), **MeanSplitBallTree** (p.455)

Definition at line 112 of file typedef.hpp.

38.55.2.3 BinaryDoubleNumericSplit

```
using BinaryDoubleNumericSplit = BinaryNumericSplit<FitnessFunction, double>
```

Definition at line 128 of file binary_numeric_split.hpp.

38.55.2.4 CosineNodeQueue

```
typedef boost::heap::priority_queue< CosineTree*, boost::heap::compare< CompareCosineNode> >
CosineNodeQueue
```

Definition at line 23 of file cosine_tree.hpp.

38.55.2.5 DecisionStump

```
using DecisionStump = DecisionTree<FitnessFunction, NumericSplitType, CategoricalSplitType,
DimensionSelectType, ElemType, false>
```

Convenience typedef for decision stumps (single level decision trees).

Definition at line 510 of file decision_tree.hpp.

38.55.2.6 DiscreteHilbertRTreeAuxiliaryInformation

```
using DiscreteHilbertRTreeAuxiliaryInformation = HilbertRTreeAuxiliaryInformation<TreeType,
DiscreteHilbertValue>
```

The Hilbert R-tree, a variant of the R tree with an ordering along the Hilbert curve.

This template typedef satisfies the TreeType policy API.

```
@inproceedings{kamel1994r,
  author = {Kamel, Ibrahim and Faloutsos, Christos},
  title = {Hilbert R-tree: An Improved R-tree Using Fractals},
  booktitle = {Proceedings of the 20th International Conference on Very Large Data Bases},
  series = {VLDB '94},
  year = {1994},
  isbn = {1-55860-153-8},
  pages = {500--509},
  numpages = {10},
  url = {http://dl.acm.org/citation.cfm?id=645920.673001},
  acmid = {673001},
  publisher = {Morgan Kaufmann Publishers Inc.},
  address = {San Francisco, CA, USA}
}
```

See also

The TreeType policy in mlpack (p. 179), **RTree** (p. 459), **DiscreteHilbertRTree**

Definition at line 128 of file typedef.hpp.

38.55.2.7 HilbertRTree

```
using HilbertRTree = RectangleTree<MetricType, StatisticType, MatType, HilbertRTreeSplit<2>,
HilbertRTreeDescentHeuristic, DiscreteHilbertRTreeAuxiliaryInformation>
```

Definition at line 136 of file typedef.hpp.

38.55.2.8 HoeffdingDoubleNumericSplit

```
using HoeffdingDoubleNumericSplit = HoeffdingNumericSplit<FitnessFunction, double>
```

Convenience typedef.

Definition at line 148 of file hoeffding_numeric_split.hpp.

38.55.2.9 HoeffdingTreeType

```
typedef StreamingDecisionTree< HoeffdingTree<> > HoeffdingTreeType
```

Definition at line 21 of file typedef.hpp.

38.55.2.10 Hyperplane

```
using Hyperplane = HyperplaneBase< bound::BallBound<MetricType>, ProjVector>
```

Hyperplane represents a general hyperplane (not necessarily axis-orthogonal).

Definition at line 151 of file hyperplane.hpp.

38.55.2.11 KDTree

```
using KDTree = BinarySpaceTree<MetricType, StatisticType, MatType, bound::HRectBound, Midpoint↵  
Split>
```

The standard midpoint-split kd-tree.

This is not the original formulation by Bentley but instead the later formulation by Deng and Moore, which only holds points in the leaves of the tree. When recursively splitting nodes, the KDTree class select the dimension with maximum variance to split on, and picks the midpoint of the range in that dimension as the value on which to split nodes.

For more information, see the following papers.

```
@article{bentley1975multidimensional,  
  title={Multidimensional binary search trees used for associative searching},  
  author={Bentley, J.L.},  
  journal={Communications of the ACM},  
  volume={18},  
  number={9},  
  pages={509--517},  
  year={1975},  
  publisher={ACM}  
}  
  
@inproceedings{deng1995multiresolution,  
  title={Multiresolution instance-based learning},  
  author={Deng, K. and Moore, A.W.},  
  booktitle={Proceedings of the 1995 International Joint Conference on AI  
    (IJCAI-95)},  
  pages={1233--1239},  
  year={1995}  
}
```

This template typedef satisfies the TreeType policy API.

See also

The TreeType policy in mlpack (p. 179), **BinarySpaceTree** (p. 1998), **MeanSplitKDTree** (p. 455)

Definition at line 63 of file typedef.hpp.

38.55.2.12 MaxRPTree

```
using MaxRPTree = BinarySpaceTree<MetricType, StatisticType, MatType, bound::HRectBound, RP↵  
TreeMaxSplit>
```

A max-split random projection tree.

When recursively splitting nodes, the MaxSplitRPTree class selects a random hyperplane and splits a node by the hyperplane. The tree holds points in leaf nodes. In contrast to the k-d tree, children of a MaxSplitRPTree node may overlap.

```
@inproceedings{dasgupta2008,
  author = {Dasgupta, Sanjoy and Freund, Yoav},
  title = {Random Projection Trees and Low Dimensional Manifolds},
  booktitle = {Proceedings of the Fortieth Annual ACM Symposium on Theory of
    Computing},
  series = {STOC '08},
  year = {2008},
  pages = {537--546},
  numpages = {10},
  publisher = {ACM},
  address = {New York, NY, USA},
}
```

This template typedef satisfies the TreeType policy API.

See also

The TreeType policy in mlpack (p.179), **BinarySpaceTree** (p.1998), **BallTree** (p.451), **MeanSplitKDTree** (p.455)

Definition at line 232 of file typedef.hpp.

38.55.2.13 MeanSplitBallTree

```
using MeanSplitBallTree = BinarySpaceTree<MetricType, StatisticType, MatType, bound::Ball↔  
Bound, MeanSplit>
```

A mean-split ball tree.

This tree, like the BallTree, holds its points only in the leaves. The tree construction algorithm here is the same as Omohundro's 'K-dc onstruction algorithm', except the splitting value is the mean, not the median. This can result in trees that better reflect the data, although they may be unbalanced.

```
@techreport{omohundro1989five,
  author={S.M. Omohundro},
  title={Five balltree construction algorithms},
  year={1989},
  institution={University of California, Berkeley International Computer
    Science Institute Technical Reports},
  number={TR-89-063}
}
```

This template typedef satisfies the TreeType policy API.

See also

The TreeType policy in mlpack (p.179), **BinarySpaceTree** (p.1998), **BallTree** (p.451), **MeanSplitKDTree** (p.455)

Definition at line 141 of file typedef.hpp.

38.55.2.14 MeanSplitKdTree

```
using MeanSplitKdTree = BinarySpaceTree<MetricType, StatisticType, MatType, bound::HRectBound, MeanSplit>
```

A mean-split kd-tree.

This is the same as the KdTree, but this particular implementation will use the mean of the data in the split dimension as the value on which to split, instead of the midpoint. This can sometimes give better performance, but it is not always clear which type of tree is best.

This template typedef satisfies the TreeType policy API.

See also

The TreeType policy in mlpack (p. 179), **BinarySpaceTree** (p. 1998), **KdTree** (p. 453)

Definition at line 80 of file typedef.hpp.

38.55.2.15 MeanSPTree

```
using MeanSPTree = SpillTree<MetricType, StatisticType, MatType, AxisOrthogonalHyperplane, MeanSpaceSplit>
```

A mean-split hybrid spill tree.

This is the same as the SPTree, but this particular implementation will use the mean of the data in the split dimension as the value on which to split, instead of the midpoint. This can sometimes give better performance, but it is not always clear which type of tree is best.

This template typedef satisfies the TreeType policy API.

See also

The TreeType policy in mlpack (p. 179), **SpillTree** (p. 2274), **SPTree** (p. 460)

Definition at line 80 of file typedef.hpp.

38.55.2.16 NonOrtMeanSPTree

```
using NonOrtMeanSPTree = SpillTree<MetricType, StatisticType, MatType, Hyperplane, MeanSpace↵
Split>
```

A mean-split hybrid spill tree considering general splitting hyperplanes (not necessarily axis-orthogonal).

This is the same as the NonOrtSPTree, but this particular implementation will use the mean of the data in the split projection as the value on which to split, instead of the midpoint. This can sometimes give better performance, but it is not always clear which type of tree is best.

This template typedef satisfies the TreeType policy API.

See also

The TreeType policy in mlpack (p. 179), **SpillTree** (p. 2274), **MeanSPTree** (p. 456), **NonOrtSPTree** (p. 457)

Definition at line 119 of file typedef.hpp.

38.55.2.17 NonOrtSPTree

```
using NonOrtSPTree = SpillTree<MetricType, StatisticType, MatType, Hyperplane, MidpointSpace↵
Split>
```

A hybrid spill tree considering general splitting hyperplanes (not necessarily axis-orthogonal).

This particular implementation will consider the midpoint of the projection of the data in the vector determined by the farthest pair of points. This can sometimes give better performance, but generally it doesn't because it takes $O(d)$ to calculate the projection of the query point when deciding which node to traverse, while when using a axis-orthogonal hyperplane, as SPTree does, we can do it in $O(1)$.

This template typedef satisfies the TreeType policy API.

See also

The TreeType policy in mlpack (p. 179), **SpillTree** (p. 2274), **SPTree** (p. 460)

Definition at line 100 of file typedef.hpp.

38.55.2.18 RPlusPlusTree

```
using RPlusPlusTree = RectangleTree<MetricType, StatisticType, MatType, RPlusTreeSplit< RPlusTreeSplitPolicy, MinimalSplitsNumberSweep>, RPlusPlusTreeDescentHeuristic, RPlusPlusTreeAuxiliaryInformation>
```

The R++ tree, a variant of the R+ tree with maximum buonding rectangles.

This template typedef satisfies the TreeType policy API.

```
@inproceedings{sumak2014r,
  author = {{\v{S}}um{\a}k, Martin and Gursk{\y}, Peter},
  title = {R++-Tree: An Efficient Spatial Access Method for Highly Redundant Point Data},
  booktitle = {New Trends in Databases and Information Systems: 17th East European Conference on Advances in Databases and Information Systems},
  year = {2014},
  isbn = {978-3-319-01863-8},
  pages = {37--44},
  publisher = {Springer International Publishing},
}
```

See also

The TreeType policy in mpack (p. 179), **RTree** (p. 459), **RTree** (p. 459), **RPlusTree** (p. 458), **RPlusPlusTree** (p. 457)

Definition at line 197 of file typedef.hpp.

38.55.2.19 RPlusTree

```
using RPlusTree = RectangleTree<MetricType, StatisticType, MatType, RPlusTreeSplit< RPlusTreeSplitPolicy, MinimalCoverageSweep>, RPlusTreeDescentHeuristic, NoAuxiliaryInformation>
```

The R+ tree, a variant of the R tree that avoids overlapping rectangles.

The implementation is modified from the original paper implementation. This template typedef satisfies the TreeType policy API.

```
@inproceedings{sellis1987r,
  author = {Sellis, Timos K. and Roussopoulos, Nick and Faloutsos, Christos},
  title = {The R+-Tree: A Dynamic Index for Multi-Dimensional Objects},
  booktitle = {Proceedings of the 13th International Conference on Very Large Data Bases},
  series = {VLDB '87},
  year = {1987},
  isbn = {0-934613-46-X},
  pages = {507--518},
  numpages = {12},
  publisher = {Morgan Kaufmann Publishers Inc.},
  address = {San Francisco, CA, USA},
}
```

See also

The TreeType policy in mpack (p. 179), **RTree** (p. 459), **RTree** (p. 459), **RPlusTree** (p. 458)

Definition at line 168 of file typedef.hpp.

38.55.2.20 RPTree

```
using RPTree = BinarySpaceTree<MetricType, StatisticType, MatType, bound::HRectBound, RPTree↵
MeanSplit>
```

A mean-split random projection tree.

When recursively splitting nodes, the RPTree class may perform one of two different kinds of split. Depending on the diameter and the average distance between points, the node may be split by a random hyperplane or according to the distance from the mean point. The tree holds points in leaf nodes. In contrast to the k-d tree, children of a MaxSplitR↵PTree node may overlap.

```
@inproceedings{dasgupta2008,
  author = {Dasgupta, Sanjoy and Freund, Yoav},
  title = {Random Projection Trees and Low Dimensional Manifolds},
  booktitle = {Proceedings of the Fortieth Annual ACM Symposium on Theory of
    Computing},
  series = {STOC '08},
  year = {2008},
  pages = {537--546},
  numpages = {10},
  publisher = {ACM},
  address = {New York, NY, USA},
}
```

This template typedef satisfies the TreeType policy API.

See also

The TreeType policy in mlpack (p.179), **BinarySpaceTree** (p.1998), **BallTree** (p.451), **MeanSplitKDTree** (p.455)

Definition at line 266 of file typedef.hpp.

38.55.2.21 RStarTree

```
using RStarTree = RectangleTree<MetricType, StatisticType, MatType, RStarTreeSplit, RStar↵
TreeDescentHeuristic, NoAuxiliaryInformation>
```

The R*-tree, a more recent variant of the R tree.

This template typedef satisfies the TreeType policy API.

```
@inproceedings{beckmann1990r,
  title={The R*-tree: an efficient and robust access method for points and
    rectangles},
  author={Beckmann, N. and Kriegel, H.-P. and Schneider, R. and Seeger, B.},
  booktitle={Proceedings of the 1990 ACM SIGMOD International Conference on
    Management of Data (SIGMOD '90)},
  volume={19},
  number={2},
  year={1990},
  publisher={ACM}
}
```

See also

The TreeType policy in mlpack (p.179), **RTree** (p.459)

Definition at line 75 of file typedef.hpp.

38.55.2.22 RTree

```
using RTree = RectangleTree<MetricType, StatisticType, MatType, RTreeSplit, RTreeDescent↵
Heuristic, NoAuxiliaryInformation>
```

An implementation of the R tree that satisfies the TreeType policy API.

This is the same R-tree structure as proposed by Guttman:

```
@inproceedings{guttman1984r,
  title={R-trees: a dynamic index structure for spatial searching},
  author={Guttman, A.},
  booktitle={Proceedings of the 1984 ACM SIGMOD International Conference on
    Management of Data (SIGMOD '84)},
  volume={14},
  number={2},
  year={1984},
  publisher={ACM}
}
```

See also

The TreeType policy in mlpack (p. 179), **RStarTree** (p. 459)

Definition at line 47 of file typedef.hpp.

38.55.2.23 SPTree

```
using SPTree = SpillTree<MetricType, StatisticType, MatType, AxisOrthogonalHyperplane, Midpoint↵
SpaceSplit>
```

The hybrid spill tree.

It is a variant of metric-trees in which the children of a node can "spill over" onto each other, and contain shared datapoints.

When recursively splitting nodes, the SPTree class select the dimension with maximum width to split on, and picks the midpoint of the range in that dimension as the value on which to split nodes.

In each case a "overlapping buffer" is defined, included points at a distance less than tau from the decision boundary defined by the midpoint.

For each node, we first split the points considering the overlapping buffer. If either of its children contains more than rho fraction of the total points we undo the overlapping splitting. Instead a conventional partition is used. In this way, we can ensure that each split reduces the number of points of a node by at least a constant factor.

For more information, see the following paper.

```
@inproceedings{
  author = {Ting Liu, Andrew W. Moore, Alexander Gray and Ke Yang},
  title = {An Investigation of Practical Approximate Nearest Neighbor
    Algorithms},
  booktitle = {Advances in Neural Information Processing Systems 17},
  year = {2005},
  pages = {825--832}
}
```

This template typedef satisfies the TreeType policy API.

See also

The TreeType policy in mlpack (p. 179), **SpillTree** (p. 2274), **MeanSPTree** (p. 456)

Definition at line 62 of file typedef.hpp.

38.55.2.24 StandardCoverTree

```
using StandardCoverTree = CoverTree<MetricType, StatisticType, MatType, FirstPointIsRoot>
```

The standard cover tree, as detailed in the original cover tree paper:

```
@inproceedings{
  author={Beygelzimer, A. and Kakade, S. and Langford, J.},
  title={Cover trees for nearest neighbor},
  booktitle={Proceedings of the 23rd International Conference on Machine
    Learning (ICML 2006)},
  pages={97--104},
  year={2006}
}
```

This template typedef satisfies the requirements of the TreeType API.

See also

The TreeType policy in mlpack (p. 179), **CoverTree** (p. 2040)

Definition at line 42 of file typedef.hpp.

38.55.2.25 UBTree

```
using UBTree = BinarySpaceTree<MetricType, StatisticType, MatType, bound::CellBound, UBTree←  
Split>
```

The Universal B-tree.

When recursively splitting nodes, the class calculates addresses of all points and splits each node according to the median address. Children may overlap since the implementation of a tighter bound requires a lot of arithmetic operations. In order to get a tighter bound increase the CellBound::maxNumBounds constant.

```
@inproceedings{bayer1997,
  author = {Bayer, Rudolf},
  title = {The Universal B-Tree for Multidimensional Indexing: General
    Concepts},
  booktitle = {Proceedings of the International Conference on Worldwide
    Computing and Its Applications},
  series = {WWCA '97},
  year = {1997},
  isbn = {3-540-63343-X},
  pages = {198--209},
  numpages = {12},
  publisher = {Springer-Verlag},
  address = {London, UK, UK},
}
```

This template typedef satisfies the TreeType policy API.

See also

The TreeType policy in mlpack (p. 179), **BinarySpaceTree** (p. 1998), **BallTree** (p. 451), **MeanSplitKDTree** (p. 455)

Definition at line 301 of file typedef.hpp.

38.55.2.26 VPTree

```
using VPTree = BinarySpaceTree<MetricType, StatisticType, MatType, bound::HollowBallBound, V↔
PTreeSplit>
```

Definition at line 199 of file typedef.hpp.

38.55.2.27 VPTreeSplit

```
using VPTreeSplit = VantagePointSplit<BoundType, MatType, 100>
```

The vantage point tree (which is also called the metric tree.

Vantage point trees and metric trees were invented independently by Yianilos an Uhlmann) is a kind of the binary space tree. When recursively splitting nodes, the VPTree class selects the vantage point and splits the node according to the distance to this point. Thus, points that are closer to the vantage point form the inner subtree. Other points form the outer subtree. The vantage point is contained in the first (inner) node.

This implementation differs from the original algorithms. Namely, vantage points are not contained in intermediate nodes. The tree has points only in the leaves of the tree.

For more information, see the following papers.

```
@inproceedings{yianilos1993vptrees,
  author = {Yianilos, Peter N.},
  title = {Data Structures and Algorithms for Nearest Neighbor Search in
    General Metric Spaces},
  booktitle = {Proceedings of the Fourth Annual ACM-SIAM Symposium on
    Discrete Algorithms},
  series = {SODA '93},
  year = {1993},
  isbn = {0-89871-313-7},
  pages = {311--321},
  numpages = {11},
  publisher = {Society for Industrial and Applied Mathematics},
  address = {Philadelphia, PA, USA}
}

@article{uhlmann1991metrictrees,
  author = {Jeffrey K. Uhlmann},
  title = {Satisfying general proximity / similarity queries with metric
    trees},
  journal = {Information Processing Letters},
  volume = {40},
  number = {4},
  pages = {175 - 179},
  year = {1991},
}
```

This template typedef satisfies the TreeType policy API.

See also

The TreeType policy in mlpack (p. 179), **BinarySpaceTree** (p. 1998), **VantagePointTree**, **VPTree** (p. 461)

Definition at line 192 of file typedef.hpp.

38.55.2.28 XTree

```
using XTree = RectangleTree<MetricType, StatisticType, MatType, XTreeSplit, RTreeDescent↵
Heuristic, XTreeAuxiliaryInformation>
```

The X-tree, a variant of the R tree with supernodes.

This template typedef satisfies the TreeType policy API.

```
@inproceedings{berchtold1996r,
  title = {The X-Tree: An Index Structure for High--Dimensional Data},
  author = {Berchtold, Stefan and Keim, Daniel A. and Kriegel, Hans-Peter},
  booktitle = {Proc. 22th Int. Conf. on Very Large Databases (VLDB'96), Bombay, India},
  editor = {Vijayaraman, T. and Buchmann, Alex and Mohan, C. and Sarda, N.},
  pages = {28--39},
  year = {1996},
  publisher = {Morgan Kaufmann}
}
```

See also

The TreeType policy in mlpack (p. 179), **RTree** (p. 459), **RStarTree** (p. 459)

Definition at line 101 of file typedef.hpp.

38.55.3 Function Documentation

38.55.3.1 Bootstrap()

```
void mlpack::tree::Bootstrap (
    const MatType & dataset,
    const LabelsType & labels,
    const WeightsType & weights,
    MatType & bootstrapDataset,
    LabelsType & bootstrapLabels,
    WeightsType & bootstrapWeights )
```

Given a dataset, create another dataset via bootstrap sampling, with labels.

Definition at line 26 of file bootstrap.hpp.

38.55.3.2 EnumerateTree()

```
void mlpack::tree::EnumerateTree (
    TreeType * tree,
    Walker & walker ) [inline]
```

Traverses all nodes of the tree, including the inner ones.

On each node two methods of the `enum` are called:

`Enter(TreeType* node, TreeType* parent);` `Leave(TreeType* node, TreeType* parent);`

Parameters

<i>walker</i>	An instance of custom class, receiver of the enumeration.
---------------	---

Definition at line 55 of file `enumerate_tree.hpp`.

References `mlpack::tree::enumerate::EnumerateTreeImpl()`.

38.55.4 Variable Documentation

38.55.4.1 MAX_OVERLAP

```
const double MAX_OVERLAP = 0.2
```

The X-tree paper says that a maximum allowable overlap of 20% works well.

This code should eventually be refactored so as to avoid polluting `mlpack::tree` (p. 444) with this random double.

Definition at line 29 of file `x_tree_split.hpp`.

38.56 mlpack::tree::enumerate Namespace Reference

Functions

- `template<class TreeType , class Walker >`
`void EnumerateTreeImpl (TreeType *tree, Walker &walker, bool root)`

38.56.1 Function Documentation

38.56.1.1 EnumerateTreeImpl()

```
void mlpack::tree::enumerate::EnumerateTreeImpl (
    TreeType * tree,
    Walker & walker,
    bool root )
```

Definition at line 24 of file `enumerate_tree.hpp`.

Referenced by `mlpack::tree::EnumerateTree()`.

38.57 mlpack::tree::split Namespace Reference

Functions

- template<typename MatType , typename SplitType >
size_t **PerformSplit** (MatType &data, const size_t begin, const size_t count, const typename SplitType::SplitInfo &splitInfo)

This function implements the default split behavior i.e.

- template<typename MatType , typename SplitType >
size_t **PerformSplit** (MatType &data, const size_t begin, const size_t count, const typename SplitType::SplitInfo &splitInfo, std::vector< size_t > &oldFromNew)

This function implements the default split behavior i.e.

38.57.1 Function Documentation

38.57.1.1 PerformSplit() [1/2]

```
size_t mlpack::tree::split::PerformSplit (
    MatType & data,
    const size_t begin,
    const size_t count,
    const typename SplitType::SplitInfo & splitInfo )
```

This function implements the default split behavior i.e.

it rearranges points according to the split information. The SplitType::AssignToLeftNode() function is used in order to determine the child that contains any particular point.

Parameters

<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitInfo</i>	The information about the split.

Definition at line 36 of file perform_split.hpp.

References Log::Assert().

38.57.1.2 PerformSplit() [2/2]

```
size_t mlpack::tree::split::PerformSplit (
    MatType & data,
```

```

const size_t begin,
const size_t count,
const typename SplitType::SplitInfo & splitInfo,
std::vector< size_t > & oldFromNew )

```

This function implements the default split behavior i.e.

it rearranges points according to the split information. The `SplitType::AssignToLeftNode()` function is used in order to determine the child that contains any particular point. The function takes care of indices and returns the list of changed indices.

Parameters

<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitInfo</i>	The information about the split.
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.

Definition at line 101 of file `perform_split.hpp`.

References `Log::Assert()`.

38.58 mlpack::util Namespace Reference

Classes

- struct **IsStdVector**
Metaprogramming structure for vector detection.
- struct **IsStdVector**< `std::vector`< **T**, **A** > >
Metaprogramming structure for vector detection.
- class **NullOutputStream**
*Used for **Log::Debug** (p. 1540) when not compiled with debugging symbols.*
- struct **ParamData**
*This structure holds all of the information about a single parameter, including its value (which is set when **ParseCommandLine()** (p. 302) is called).*
- class **PrefixedOutputStream**
Allows us to output to an ostream with a prefix at the beginning of each line, in the same way we would output to cout or cerr.
- class **ProgramDoc**
*A static object whose constructor registers program documentation with the **CLI** (p. 1117) class.*

Functions

- void **DisableBacktrace** ()
Disable backtraces.
- void **DisableVerbose** ()
Turn verbose output off.
- void **EnableTimers** ()
Enable timing.
- void **EnableVerbose** ()
Turn verbose output on.
- template<typename T >
T * **GetParamPtr** (const std::string ¶mName)
Return a pointer.
- template<typename T >
T & **GetParamWithInfo** (const std::string ¶mName)
Return the matrix part of a matrix + dataset info parameter.
- std::string **GetVersion** ()
This will return either "mlpack x.y.z" or "mlpack master-XXXXXXX" depending on whether or not this is a stable version of mlpack or a git repository.
- std::string **HyphenateString** (const std::string &str, int padding)
** Hyphenate a string or split it onto multiple 80-character lines, with some * amount of padding on each line.*
- void **ReportIgnoredParam** (const std::vector< std::pair< std::string, bool >> &constraints, const std::string ¶mName)
Report that a parameter is ignored, if each of the constraints given are satisfied.
- void **ReportIgnoredParam** (const std::string ¶mName, const std::string &reason)
If the given parameter is passed, report that it is ignored, supplying a custom reason.
- void **RequireAtLeastOnePassed** (const std::vector< std::string > &constraints, const bool fatal=true, const std::string &customErrorMessage="")
*Require that at least one of the given parameters in the constraints set was passed to the **CLI** (p. 1117) object; otherwise, issue a warning or fatal error, optionally with the given custom error message.*
- void **RequireNoneOrAllPassed** (const std::vector< std::string > &constraints, const bool fatal=true, const std::string &customErrorMessage="")
*Require that either none or all of the given parameters in the constraints set were passed to the **CLI** (p. 1117) object; otherwise, issue a warning or fatal error, optionally with the given custom error message.*
- void **RequireOnlyOnePassed** (const std::vector< std::string > &constraints, const bool fatal=true, const std::string &customErrorMessage="")
*Require that only one of the given parameters in the constraints set was passed to the **CLI** (p. 1117) object; otherwise, issue a warning or fatal error, optionally with the given custom error message.*
- template<typename T >
void **RequireParamInSet** (const std::string ¶mName, const std::vector< T > &set, const bool fatal, const std::string &errorMessage)
Require that a given parameter is in a set of allowable parameters.
- template<typename T >
void **RequireParamValue** (const std::string ¶mName, const std::function< bool(T)> &conditional, const bool fatal, const std::string &errorMessage)
Require that a given parameter satisfies the given conditional function.
- void **ResetTimers** ()
Reset the status of all timers.
- template<typename T >
void **SetInputParam** (const std::string &name, T &&value)

Utility function that is used in binding tests for setting a parameter and marking it as passed; it uses copy semantics for lvalues and move semantics for rvalues.

- `template<typename T >`
`void SetParam (const std::string &identifier, T &value)`
Set the parameter to the given value.
- `template<typename T >`
`void SetParamPtr (const std::string &identifier, T *value, const bool copy)`
Set the parameter to the given value, given that the type is a pointer.
- `template<typename T >`
`void SetParamWithInfo (const std::string &identifier, T &matrix, const bool *dims)`
Set the parameter (which is a matrix/DatasetInfo tuple) to the given value.

38.58.1 Function Documentation

38.58.1.1 DisableBacktrace()

```
void mlpack::util::DisableBacktrace ( ) [inline]
```

Disable backtraces.

Definition at line 144 of file `cli_util.hpp`.

References `PrefixedOutputStream::backtrace`, and `Log::Fatal`.

38.58.1.2 DisableVerbose()

```
void mlpack::util::DisableVerbose ( ) [inline]
```

Turn verbose output off.

Definition at line 136 of file `cli_util.hpp`.

References `PrefixedOutputStream::ignoreInput`, and `Log::Info`.

38.58.1.3 EnableTimers()

```
void mlpack::util::EnableTimers ( ) [inline]
```

Enable timing.

Definition at line 161 of file `cli_util.hpp`.

References `Timer::EnableTiming()`.

38.58.1.4 EnableVerbose()

```
void mlpack::util::EnableVerbose ( ) [inline]
```

Turn verbose output on.

Definition at line 128 of file cli_util.hpp.

References PrefixedOutputStream::ignoreInput, and Log::Info.

38.58.1.5 GetParamPtr()

```
T* mlpack::util::GetParamPtr (
    const std::string & paramName )
```

Return a pointer.

This function exists to work around Cython's seeming lack of support for template pointer types.

Definition at line 109 of file cli_util.hpp.

38.58.1.6 GetParamWithInfo()

```
T& mlpack::util::GetParamWithInfo (
    const std::string & paramName )
```

Return the matrix part of a matrix + dataset info parameter.

Definition at line 118 of file cli_util.hpp.

38.58.1.7 GetVersion()

```
std::string mlpack::util::GetVersion ( )
```

This will return either "mlpack x.y.z" or "mlpack master-XXXXXXX" depending on whether or not this is a stable version of mlpack or a git repository.

Referenced by mlpack::bindings::cli::ParseCommandLine().

38.58.1.8 HyphenateString()

```
std::string mlpack::util::HyphenateString (
    const std::string & str,
    int padding ) [inline]
```

* Hyphenate a string or split it onto multiple 80-character lines, with some * amount of padding on each line.

This is used for option output. * *

Parameters

<i>str</i>	String to hyphenate (splits are on ' '). *
<i>padding</i>	Amount of padding on the left for each new line.

Definition at line 25 of file hyphenate_string.hpp.

Referenced by `mlpack::bindings::python::PrintDoc()`.

38.58.1.9 ReportIgnoredParam() [1/2]

```
void mlpack::util::ReportIgnoredParam (
    const std::vector< std::pair< std::string, bool >> & constraints,
    const std::string & paramName )
```

Report that a parameter is ignored, if each of the constraints given are satisfied.

The constraints should be a set of string/bool pairs. If all of the constraints are true, and the given parameter in 'paramName' is passed, then a warning will be issued noting that the parameter is ignored. The warning will go to **Log::Warn** (p. 1541).

Parameters

<i>constraints</i>	Set of constraints.
<i>paramName</i>	Name of parameter to check.

38.58.1.10 ReportIgnoredParam() [2/2]

```
void mlpack::util::ReportIgnoredParam (
    const std::string & paramName,
    const std::string & reason )
```

If the given parameter is passed, report that it is ignored, supplying a custom reason.

The reason should specify, in short and clear terms, why the parameter is ignored. So, for example, the output may be similar to:

```
--iterations (-i) ignored because <reason>.
```

and in this case a good reason might be "SGD is not being used as an optimizer". Be sure that when you write the reason, the full message makes sense.

Parameters

<i>paramName</i>	Name of parameter to check.
<i>reason</i>	Reason that parameter is ignored, if it is passed.

38.58.1.11 RequireAtLeastOnePassed()

```
void mlpack::util::RequireAtLeastOnePassed (
    const std::vector< std::string > & constraints,
    const bool fatal = true,
    const std::string & customErrorMessage = "" )
```

Require that at least one of the given parameters in the constraints set was passed to the **CLI** (p. 1117) object; otherwise, issue a warning or fatal error, optionally with the given custom error message.

This uses the correct binding type name for each parameter (i.e. '-parameter' for **CLI** (p. 1117) bindings, 'parameter' for Python bindings).

This can be used with a set of only one constraint and the output is still sensible.

If you use a custom error message, be aware that the given output will be similar to, for example:

```
Should pass one of '--codes_file (-c)', '--dictionary_file (-d)', or
'--output_model_file (-M)'; <custom error message>!
```

so when you write your custom error message, be sure that the sentence makes sense. The custom error message should not have a capitalized first character and no ending punctuation (a '!' will be added by this function).

Parameters

<i>constraints</i>	Set of parameters from which only one should be passed.
<i>fatal</i>	If true, output goes to Log::Fatal (p. 1540) instead of Log::Warn (p. 1541) and an exception is thrown.
<i>customErrorMessage</i>	Error message to append.

38.58.1.12 RequireNoneOrAllPassed()

```
void mlpack::util::RequireNoneOrAllPassed (
    const std::vector< std::string > & constraints,
    const bool fatal = true,
    const std::string & customErrorMessage = "" )
```

Require that either none or all of the given parameters in the constraints set were passed to the **CLI** (p. 1117) object; otherwise, issue a warning or fatal error, optionally with the given custom error message.

This uses the correct binding type name for each parameter (i.e. '-parameter' for **CLI** (p. 1117) bindings, 'parameter' for Python bindings).

If you use a custom error message, be aware that the given output will be similar to, for example:

```
Must pass none or all of '--codes_file (-c)', '--dictionary_file (-d)', and
'--output_model_file (-M)'; <custom error message>!
```

so when you write your custom error message, be sure that the sentence makes sense. The custom error message should not have a capitalized first character and no ending punctuation (a '!' will be added by this function).

Parameters

<i>constraints</i>	Set of parameters of which none or all should be passed.
<i>fatal</i>	If true, output goes to Log::Fatal (p. 1540) instead of Log::Warn (p. 1541) and an exception is thrown.
<i>customErrorMessage</i>	Error message to append.

38.58.1.13 RequireOnlyOnePassed()

```
void mlpack::util::RequireOnlyOnePassed (
    const std::vector< std::string > & constraints,
    const bool fatal = true,
    const std::string & customErrorMessage = "" )
```

Require that only one of the given parameters in the constraints set was passed to the **CLI** (p. 1117) object; otherwise, issue a warning or fatal error, optionally with the given custom error message.

This uses the correct binding type name for each parameter (i.e. '-parameter' for **CLI** (p. 1117) bindings, 'parameter' for Python bindings).

If you use a custom error message, be aware that the given output will be similar to, for example:

```
Must specify one of '--reference_file (-r)' or '--input_model_file (-m)';
<custom error message here>!
```

so when you write your custom error message, be sure that the sentence makes sense. The custom error message should not have a capitalized first character and no ending punctuation (a '!' will be added by this function).

Parameters

<i>constraints</i>	Set of parameters from which only one should be passed.
<i>fatal</i>	If true, output goes to Log::Fatal (p. 1540) instead of Log::Warn (p. 1541) and an exception is thrown.
<i>customErrorMessage</i>	Error message to append.

38.58.1.14 RequireParamInSet()

```
void mlpack::util::RequireParamInSet (
    const std::string & paramName,
    const std::vector< T > & set,
    const bool fatal,
    const std::string & errorMessage )
```

Require that a given parameter is in a set of allowable parameters.

This is probably most useful with `T = std::string`. If `fatal` is true, then an exception is thrown. An error message is not optional and must be specified. The error message does *not* need to specify the values in the set; this function will already output them. So, for example, the output may be similar to:

```
Invalid value of '--weak_learner (-w)' specified ('something'); <error
message>; must be one of 'decision_stump', or 'perceptron'!
```

so when you write the error message, make sure that the message makes sense. For example, in the message above, a good error message might be "unknown weak learner type".

Template Parameters

<i>T</i>	Type of parameter.
----------	--------------------

Parameters

<i>paramName</i>	Name of parameter to check.
<i>set</i>	Set of valid values for parameter.
<i>fatal</i>	If true, an exception is thrown and output goes to Log::Fatal (p. 1540).
<i>errorMessage</i>	Error message to output.

38.58.1.15 RequireParamValue()

```
void mlpack::util::RequireParamValue (
    const std::string & paramName,
    const std::function< bool(T)> & conditional,
    const bool fatal,
    const std::string & errorMessage )
```

Require that a given parameter satisfies the given conditional function.

This is useful for, e.g., checking that a given parameter is greater than 0. If `fatal` is true, then an exception is thrown. An error message is not optional and must be specified. The error message should specify, in clear terms, what the value of the parameter *should* be. So, for example, the output may be similar to:

Invalid value of '`--iterations (-i)`' specified (-1); <error message>!

and in this case a good error message might be "number of iterations must be positive". Be sure that when you write the error message, the message makes sense.

Template Parameters

<i>T</i>	Type of parameter to check.
----------	-----------------------------

Parameters

<i>paramName</i>	Name of parameter to check.
<i>conditional</i>	Function to use to check parameter value; should return 'true' if the parameter value is okay.
<i>fatal</i>	If true, an exception is thrown and output goes to Log::Fatal (p. 1540).
<i>errorMessage</i>	Error message to output.

38.58.1.16 ResetTimers()

```
void mlpack::util::ResetTimers ( ) [inline]
```

Reset the status of all timers.

Definition at line 152 of file cli_util.hpp.

References CLI::GetSingleton(), Timers::Reset(), and CLI::timer.

38.58.1.17 SetInputParam()

```
void mlpack::util::SetInputParam (
    const std::string & name,
    T && value )
```

Utility function that is used in binding tests for setting a parameter and marking it as passed; it uses copy semantics for lvalues and move semantics for rvalues.

Parameters

<i>name</i>	Name of parameter to set.
<i>value</i>	Value to set parameter to.

Definition at line 29 of file test_helper.hpp.

References CLI::SetPassed().

38.58.1.18 SetParam()

```
void mlpack::util::SetParam (
    const std::string & identifier,
    T & value ) [inline]
```

Set the parameter to the given value.

This function exists to work around Cython's lack of support for lvalue references.

Parameters

<i>identifier</i>	Name of parameter.
<i>value</i>	Value to set parameter to.

Definition at line 32 of file cli_util.hpp.

38.58.1.19 SetParamPtr()

```
void mlpack::util::SetParamPtr (
    const std::string & identifier,
    T * value,
    const bool copy ) [inline]
```

Set the parameter to the given value, given that the type is a pointer.

This function exists to work around both Cython's lack of support for lvalue references and also its seeming lack of support for template pointer types.

Parameters

<i>identifier</i>	Name of parameter.
<i>value</i>	Value to set parameter to.
<i>copy</i>	Whether or not the object should be copied.

Definition at line 48 of file cli_util.hpp.

38.58.1.20 SetParamWithInfo()

```
void mlpack::util::SetParamWithInfo (
    const std::string & identifier,
    T & matrix,
    const bool * dims ) [inline]
```

Set the parameter (which is a matrix/DatasetInfo tuple) to the given value.

Definition at line 59 of file cli_util.hpp.

References mlpack::data::categorical, DatasetMapper< PolicyType, InputType >::MapString(), and DatasetMapper< PolicyType, InputType >::Type().

38.59 std Namespace Reference

Typedefs

- `template<bool B, class T = void>`
using **enable_if_t** = typename enable_if< B, T >::type

38.59.1 Typedef Documentation

38.59.1.1 enable_if_t

```
using enable_if_t = typename enable_if<B, T>::type
```

Definition at line 58 of file prereqs.hpp.

Chapter 39

Class Documentation

39.1 `version< mlpack::adaboost::AdaBoost< WeakLearnerType, MatType > >` Struct Template Reference

Public Member Functions

- **BOOST_STATIC_CONSTANT** (int, value=1)

39.1.1 Detailed Description

```
template<typename WeakLearnerType, typename MatType>
struct boost::serialization::version< mlpack::adaboost::AdaBoost< WeakLearnerType, MatType > >
```

Definition at line 184 of file `adaboost.hpp`.

39.1.2 Member Function Documentation

39.1.2.1 `BOOST_STATIC_CONSTANT()`

```
BOOST_STATIC_CONSTANT (
    int ,
    value = 1 )
```

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/adaboost/ adaboost.hpp`

39.2 `version< mlpack::ann::BRNN< OutputLayerType, MergeLayerType, MergeOutputType, InitializationRuleType, CustomLayer... > > Struct Template Reference`

Public Member Functions

- **BOOST_STATIC_CONSTANT** (int, value=1)

39.2.1 Detailed Description

```
template<typename OutputLayerType, typename InitializationRuleType, typename MergeLayerType, typename MergeOutputType, type-
name... CustomLayer>
struct boost::serialization::version< mlpack::ann::BRNN< OutputLayerType, MergeLayerType, MergeOutputType, InitializationRule<
Type, CustomLayer... > >
```

Definition at line 396 of file `brnn.hpp`.

39.2.2 Member Function Documentation

39.2.2.1 `BOOST_STATIC_CONSTANT()`

```
BOOST_STATIC_CONSTANT (
    int ,
    value = 1 )
```

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/brnn.hpp`

39.3 `version< mlpack::ann::FFN< OutputLayerType, InitializationRuleType, CustomLayer... > > Struct Template Reference`

Public Member Functions

- **BOOST_STATIC_CONSTANT** (int, value=1)

39.3.1 Detailed Description

```
template<typename OutputLayerType, typename InitializationRuleType, typename... CustomLayer>
struct boost::serialization::version< mlpack::ann::FFN< OutputLayerType, InitializationRuleType, CustomLayer... > >
```

Definition at line 477 of file `ffn.hpp`.

39.3.2 Member Function Documentation

39.3.2.1 BOOST_STATIC_CONSTANT()

```
BOOST_STATIC_CONSTANT (
    int ,
    value = 1 )
```

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/ **ffn.hpp**

39.4 version< mlpack::ann::RNN< OutputLayerType, InitializationRuleType, CustomLayer... > > Struct Template Reference

Public Member Functions

- **BOOST_STATIC_CONSTANT** (int, value=1)

39.4.1 Detailed Description

```
template<typename OutputLayerType, typename InitializationRuleType, typename... CustomLayer>
struct boost::serialization::version< mlpack::ann::RNN< OutputLayerType, InitializationRuleType, CustomLayer... > >
```

Definition at line 409 of file rnn.hpp.

39.4.2 Member Function Documentation

39.4.2.1 BOOST_STATIC_CONSTANT()

```
BOOST_STATIC_CONSTANT (
    int ,
    value = 1 )
```

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/ **rnn.hpp**

39.5 InitHMMModel Struct Reference

Static Public Member Functions

- `template<typename HMMType >`
`static void Apply (HMMType &hmm, vector< mat > *trainSeq)`
- `static void Create (HMM< DiscreteDistribution > &hmm, vector< mat > &trainSeq, size_t states, double tolerance=1e-05)`
Helper function to create discrete HMM.
- `static void Create (HMM< GaussianDistribution > &hmm, vector< mat > &trainSeq, size_t states, double tolerance=1e-05)`
- `static void Create (HMM< GMM > &hmm, vector< mat > &trainSeq, size_t states, double tolerance=1e-05)`
- `static void Create (HMM< DiagonalGMM > &hmm, vector< mat > &trainSeq, size_t states, double tolerance=1e-05)`
Helper function to create Diagonal GMM HMM.
- `static void RandomInitialize (vector< DiscreteDistribution > &e)`
Helper function for discrete emission distributions.
- `static void RandomInitialize (vector< GaussianDistribution > &e)`
- `static void RandomInitialize (vector< GMM > &e)`
- `static void RandomInitialize (vector< DiagonalGMM > &e)`
Helper function for diagonal GMM emission distributions.

39.5.1 Detailed Description

Definition at line 21 of file `hmm_test_utils.hpp`.

39.5.2 Member Function Documentation

39.5.2.1 Apply()

```
static void Apply (
    HMMType & hmm,
    vector< mat > * trainSeq ) [inline], [static]
```

Definition at line 24 of file `hmm_test_utils.hpp`.

References `Create()`, and `RandomInitialize()`.

39.5.2.2 Create() [1/4]

```
static void Create (
    HMM< DiscreteDistribution > & hmm,
    vector< mat > & trainSeq,
    size_t states,
    double tolerance = 1e-05 ) [inline], [static]
```

Helper function to create discrete HMM.

Definition at line 37 of file `hmm_test_utils.hpp`.

Referenced by `Apply()`.

39.5.2.3 Create() [2/4]

```
static void Create (
    HMM< GaussianDistribution > & hmm,
    vector< mat > & trainSeq,
    size_t states,
    double tolerance = 1e-05 ) [inline], [static]
```

Definition at line 58 of file `hmm_test_utils.hpp`.

39.5.2.4 Create() [3/4]

```
static void Create (
    HMM< GMM > & hmm,
    vector< mat > & trainSeq,
    size_t states,
    double tolerance = 1e-05 ) [inline], [static]
```

Definition at line 82 of file `hmm_test_utils.hpp`.

39.5.2.5 Create() [4/4]

```
static void Create (
    HMM< DiagonalGMM > & hmm,
    vector< mat > & trainSeq,
    size_t states,
    double tolerance = 1e-05 ) [inline], [static]
```

Helper function to create Diagonal GMM HMM.

Definition at line 109 of file `hmm_test_utils.hpp`.

39.5.2.6 RandomInitialize() [1/4]

```
static void RandomInitialize (
    vector< DiscreteDistribution > & e ) [inline], [static]
```

Helper function for discrete emission distributions.

Definition at line 136 of file `hmm_test_utils.hpp`.

Referenced by `Apply()`.

39.5.2.7 RandomInitialize() [2/4]

```
static void RandomInitialize (
    vector< GaussianDistribution > & e ) [inline], [static]
```

Definition at line 145 of file `hmm_test_utils.hpp`.

39.5.2.8 RandomInitialize() [3/4]

```
static void RandomInitialize (
    vector< GMM > & e ) [inline], [static]
```

Definition at line 157 of file `hmm_test_utils.hpp`.

39.5.2.9 RandomInitialize() [4/4]

```
static void RandomInitialize (
    vector< DiagonalGMM > & e ) [inline], [static]
```

Helper function for diagonal GMM emission distributions.

Definition at line 180 of file `hmm_test_utils.hpp`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/tests/main_tests/ hmm_test_utils.hpp`

39.6 IsVector< VecType > Struct Template Reference

If value == true, then `VecType` is some sort of Armadillo vector or subview.

Static Public Attributes

- static const bool **value** = false

39.6.1 Detailed Description

```
template<typename VecType>
struct IsVector< VecType >
```

If value == true, then VecType is some sort of Armadillo vector or subview.

You might use this struct like this:

```
// Only accepts VecTypes that are actually Armadillo vector types.
template<typename VecType>
void Function(const VecType& argumentA,
              typename std::enable_if_t<IsVector<VecType>::value>* = 0);
```

The use of the enable_if_t object allows the compiler to instantiate Function() only if VecType is one of the Armadillo vector types. It has a default argument because it isn't meant to be used in either the function call or the function body.

Definition at line 35 of file arma_traits.hpp.

39.6.2 Member Data Documentation

39.6.2.1 value

```
const bool value = false [static]
```

Definition at line 37 of file arma_traits.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **arma_traits.hpp**

39.7 IsVector< arma::Col< eT > > Struct Template Reference

Static Public Attributes

- static const bool **value** = true

39.7.1 Detailed Description

```
template<typename eT>
struct IsVector< arma::Col< eT > >
```

Definition at line 44 of file arma_traits.hpp.

39.7.2 Member Data Documentation

39.7.2.1 value

```
const bool value = true [static]
```

Definition at line 46 of file arma_traits.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **arma_traits.hpp**

39.8 IsVector< arma::Row< eT > > Struct Template Reference

Static Public Attributes

- static const bool **value** = true

39.8.1 Detailed Description

```
template<typename eT>
struct IsVector< arma::Row< eT > >
```

Definition at line 58 of file arma_traits.hpp.

39.8.2 Member Data Documentation

39.8.2.1 value

```
const bool value = true [static]
```

Definition at line 60 of file arma_traits.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **arma_traits.hpp**

39.9 IsVector< arma::SpCol< eT > > Struct Template Reference

Static Public Attributes

- static const bool **value** = true

39.9.1 Detailed Description

```
template<typename eT>  
struct IsVector< arma::SpCol< eT > >
```

Definition at line 51 of file arma_traits.hpp.

39.9.2 Member Data Documentation

39.9.2.1 value

```
const bool value = true [static]
```

Definition at line 53 of file arma_traits.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **arma_traits.hpp**

39.10 IsVector< arma::SpRow< eT > > Struct Template Reference

Static Public Attributes

- static const bool **value** = true

39.10.1 Detailed Description

```
template<typename eT>
struct lsVector< arma::SpRow< eT > >
```

Definition at line 65 of file arma_traits.hpp.

39.10.2 Member Data Documentation

39.10.2.1 value

```
const bool value = true [static]
```

Definition at line 67 of file arma_traits.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **arma_traits.hpp**

39.11 lsVector< arma::SpSubview< eT > > Struct Template Reference

Static Public Attributes

- static const bool **value** = true

39.11.1 Detailed Description

```
template<typename eT>
struct lsVector< arma::SpSubview< eT > >
```

Definition at line 89 of file arma_traits.hpp.

39.11.2 Member Data Documentation

39.11.2.1 value

```
const bool value = true [static]
```

Definition at line 91 of file `arma_traits.hpp`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ arma_traits.hpp`

39.12 `IsVector< arma::subview_col< eT > >` Struct Template Reference

Static Public Attributes

- static const bool **value** = true

39.12.1 Detailed Description

```
template<typename eT>
struct IsVector< arma::subview_col< eT > >
```

Definition at line 72 of file `arma_traits.hpp`.

39.12.2 Member Data Documentation

39.12.2.1 value

```
const bool value = true [static]
```

Definition at line 74 of file `arma_traits.hpp`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ arma_traits.hpp`

39.13 `IsVector< arma::subview_row< eT > >` Struct Template Reference

Static Public Attributes

- static const bool **value** = true

39.13.1 Detailed Description

```
template<typename eT>
struct lsVector< arma::subview_row< eT > >
```

Definition at line 79 of file arma_traits.hpp.

39.13.2 Member Data Documentation

39.13.2.1 value

```
const bool value = true [static]
```

Definition at line 81 of file arma_traits.hpp.

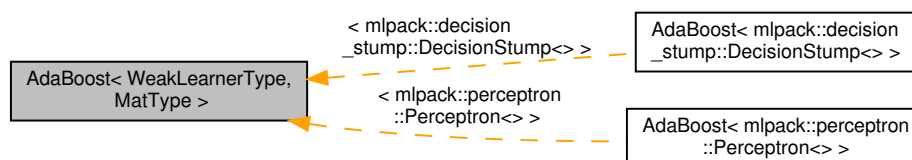
The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **arma_traits.hpp**

39.14 AdaBoost< WeakLearnerType, MatType > Class Template Reference

The **AdaBoost** (p. 488) class.

Inheritance diagram for AdaBoost< WeakLearnerType, MatType >:



Public Member Functions

- **AdaBoost** (const MatType &data, const arma::Row< size_t > &labels, const size_t numClasses, const WeakLearnerType &other, const size_t iterations=100, const double tolerance=1e-6)
Constructor.
- **AdaBoost** (const double tolerance=1e-6)
*Create the **AdaBoost** (p. 488) object without training.*
- double **Alpha** (const size_t i) const
Get the weights for the given weak learner.
- double & **Alpha** (const size_t i)
Modify the weight for the given weak learner (be careful!).
- void **Classify** (const MatType &test, arma::Row< size_t > &predictedLabels)
Classify the given test points.
- size_t **NumClasses** () const
Get the number of classes this model is trained on.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
*Serialize the **AdaBoost** (p. 488) model.*
- double **Tolerance** () const
Get the tolerance for stopping the optimization during training.
- double & **Tolerance** ()
Modify the tolerance for stopping the optimization during training.
- double **Train** (const MatType &data, const arma::Row< size_t > &labels, const size_t numClasses, const WeakLearnerType &learner, const size_t iterations=100, const double tolerance=1e-6)
*Train **AdaBoost** (p. 488) on the given dataset.*
- const WeakLearnerType & **WeakLearner** (const size_t i) const
Get the given weak learner.
- WeakLearnerType & **WeakLearner** (const size_t i)
Modify the given weak learner (be careful!).
- size_t **WeakLearners** () const
Get the number of weak learners in the model.

39.14.1 Detailed Description

```
template<typename WeakLearnerType = mlpack::perceptron::Perceptron<>, typename MatType = arma::mat>
class mlpack::adaboost::AdaBoost< WeakLearnerType, MatType >
```

The **AdaBoost** (p. 488) class.

AdaBoost (p. 488) is a boosting algorithm, meaning that it combines an ensemble of weak learners to produce a strong learner. For more information on **AdaBoost** (p. 488), see the following paper:

```
@article{schapire1999improved,
  author = {Schapire, Robert E. and Singer, Yoram},
  title = {Improved Boosting Algorithms Using Confidence-rated Predictions},
  journal = {Machine Learning},
  volume = {37},
  number = {3},
  month = dec,
  year = {1999},
  issn = {0885-6125},
  pages = {297--336},
}
```

This class is general, and can be used with any type of weak learner, so long as the learner implements the following functions:

```
// A boosting constructor, which learns using the training parameters of the
// given other WeakLearner, but uses the given instance weights for training.
WeakLearner(WeakLearner& other,
            const MatType& data,
            const arma::Row<size_t>& labels,
            const arma::rowvec& weights);

// Given the test points, classify them and output predictions into
// predictedLabels.
void Classify(const MatType& data, arma::Row<size_t>& predictedLabels);
```

For more information on and examples of weak learners, see `perceptron::Perceptron<>` and `decision_stump::DecisionStump<>`.

Template Parameters

<i>MatType</i>	Data matrix type (i.e. <code>arma::mat</code> or <code>arma::sp_mat</code>).
<i>WeakLearnerType</i>	Type of weak learner to use.

Definition at line 81 of file `adaboost.hpp`.

39.14.2 Constructor & Destructor Documentation

39.14.2.1 AdaBoost() [1/2]

```
AdaBoost (
    const MatType & data,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const WeakLearnerType & other,
    const size_t iterations = 100,
    const double tolerance = 1e-6 )
```

Constructor.

This runs the AdaBoost.MH algorithm to provide a trained boosting model. This constructor takes an already-initialized weak learner; all other weak learners will learn with the same parameters as the given weak learner.

Parameters

<i>data</i>	Input data.
<i>labels</i>	Corresponding labels.
<i>iterations</i>	Number of boosting rounds.
<i>tol</i>	The tolerance for change in values of <i>rt</i> .
<i>other</i>	Weak learner that has already been initialized.

39.14.2.2 AdaBoost() [2/2]

```
AdaBoost (  
    const double tolerance = 1e-6 )
```

Create the **AdaBoost** (p. 488) object without training.

Be sure to call **Train()** (p. 492) before calling **Classify()** (p. 491)!

39.14.3 Member Function Documentation

39.14.3.1 Alpha() [1/2]

```
double Alpha (  
    const size_t i ) const [inline]
```

Get the weights for the given weak learner.

Definition at line 121 of file adaboost.hpp.

39.14.3.2 Alpha() [2/2]

```
double& Alpha (  
    const size_t i ) [inline]
```

Modify the weight for the given weak learner (be careful!).

Definition at line 123 of file adaboost.hpp.

39.14.3.3 Classify()

```
void Classify (  
    const MatType & test,  
    arma::Row< size_t > & predictedLabels )
```

Classify the given test points.

Parameters

<i>test</i>	Testing data.
<i>predictedLabels</i>	Vector in which to the predicted labels of the test set will be stored.

Referenced by AdaBoost< mlpack::perceptron::Perceptron<> >::WeakLearner().

39.14.3.4 NumClasses()

```
size_t NumClasses ( ) const [inline]
```

Get the number of classes this model is trained on.

Definition at line 115 of file adaboost.hpp.

39.14.3.5 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the **AdaBoost** (p. 488) model.

Referenced by AdaBoost< mlpack::perceptron::Perceptron<> >::WeakLearner().

39.14.3.6 Tolerance() [1/2]

```
double Tolerance ( ) const [inline]
```

Get the tolerance for stopping the optimization during training.

Definition at line 110 of file adaboost.hpp.

39.14.3.7 Tolerance() [2/2]

```
double& Tolerance ( ) [inline]
```

Modify the tolerance for stopping the optimization during training.

Definition at line 112 of file adaboost.hpp.

39.14.3.8 Train()

```
double Train (
    const MatType & data,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const WeakLearnerType & learner,
    const size_t iterations = 100,
    const double tolerance = 1e-6 )
```

Train **AdaBoost** (p. 488) on the given dataset.

This method takes an initialized **WeakLearnerType**; the parameters for this weak learner will be used to train each of the weak learners during **AdaBoost** (p. 488) training. Note that this will completely overwrite any model that has already been trained with this object.

Parameters

<i>data</i>	Dataset to train on.
<i>labels</i>	Labels for each point in the dataset.
<i>learner</i>	Learner to use for training.

Returns

The upper bound for training error.

Referenced by AdaBoost< mlpack::perceptron::Perceptron<> >::WeakLearner().

39.14.3.9 WeakLearner() [1/2]

```
const WeakLearnerType& WeakLearner (
    const size_t i ) const [inline]
```

Get the given weak learner.

Definition at line 126 of file adaboost.hpp.

39.14.3.10 WeakLearner() [2/2]

```
WeakLearnerType& WeakLearner (
    const size_t i ) [inline]
```

Modify the given weak learner (be careful!).

Definition at line 128 of file adaboost.hpp.

39.14.3.11 WeakLearners()

```
size_t WeakLearners ( ) const [inline]
```

Get the number of weak learners in the model.

Definition at line 118 of file `adaboost.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/adaboost/ adaboost.hpp`

39.15 AdaBoostModel Class Reference

The model to save to disk.

Public Types

- enum **WeakLearnerTypes** {
 DECISION_STUMP,
 PERCEPTRON }

Public Member Functions

- **AdaBoostModel** ()
 Create an empty **AdaBoost** (p. 488) model.
- **AdaBoostModel** (const arma::Col< size_t > &mappings, const size_t weakLearnerType)
 Create the **AdaBoost** (p. 488) model with the given mappings and type.
- **AdaBoostModel** (const **AdaBoostModel** &other)
 Copy constructor.
- **AdaBoostModel** (**AdaBoostModel** &&other)
 Move constructor.
- **~AdaBoostModel** ()
 Clean up memory.
- void **Classify** (const arma::mat &testData, arma::Row< size_t > &predictions)
 Classify test points.
- size_t **Dimensionality** () const
 Get the dimensionality of the model.
- size_t & **Dimensionality** ()
 Modify the dimensionality of the model.
- const arma::Col< size_t > & **Mappings** () const
 Get the mappings.
- arma::Col< size_t > & **Mappings** ()
 Modify the mappings.
- **AdaBoostModel** & **operator=** (const **AdaBoostModel** &other)

Copy assignment operator.

- `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`

Serialize the model.

- `void Train (const arma::mat &data, const arma::Row< size_t > &labels, const size_t numClasses, const size_t iterations, const double tolerance)`

Train the model.

- `size_t WeakLearnerType () const`

Get the weak learner type.

- `size_t & WeakLearnerType ()`

Modify the weak learner type.

39.15.1 Detailed Description

The model to save to disk.

Definition at line 26 of file `adaboost_model.hpp`.

39.15.2 Member Enumeration Documentation

39.15.2.1 WeakLearnerTypes

`enum WeakLearnerTypes`

Enumerator

DECISION_STUMP	
PERCEPTRON	

Definition at line 29 of file `adaboost_model.hpp`.

39.15.3 Constructor & Destructor Documentation

39.15.3.1 AdaBoostModel() [1/4]

`AdaBoostModel ()`

Create an empty **AdaBoost** (p. 488) model.

39.15.3.2 AdaBoostModel() [2/4]

```
AdaBoostModel (
    const arma::Col< size_t > & mappings,
    const size_t weakLearnerType )
```

Create the **AdaBoost** (p. 488) model with the given mappings and type.

39.15.3.3 AdaBoostModel() [3/4]

```
AdaBoostModel (
    const AdaBoostModel & other )
```

Copy constructor.

39.15.3.4 AdaBoostModel() [4/4]

```
AdaBoostModel (
    AdaBoostModel && other )
```

Move constructor.

39.15.3.5 ~AdaBoostModel()

```
~ AdaBoostModel ( )
```

Clean up memory.

39.15.4 Member Function Documentation

39.15.4.1 Classify()

```
void Classify (
    const arma::mat & testData,
    arma::Row< size_t > & predictions )
```

Classify test points.

Referenced by AdaBoostModel::Dimensionality().

39.15.4.2 Dimensionality() [1/2]

```
size_t Dimensionality ( ) const [inline]
```

Get the dimensionality of the model.

Definition at line 78 of file `adaboost_model.hpp`.

39.15.4.3 Dimensionality() [2/2]

```
size_t& Dimensionality ( ) [inline]
```

Modify the dimensionality of the model.

Definition at line 80 of file `adaboost_model.hpp`.

References `AdaBoostModel::Classify()`, and `AdaBoostModel::Train()`.

39.15.4.4 Mappings() [1/2]

```
const arma::Col<size_t>& Mappings ( ) const [inline]
```

Get the mappings.

Definition at line 68 of file `adaboost_model.hpp`.

39.15.4.5 Mappings() [2/2]

```
arma::Col<size_t>& Mappings ( ) [inline]
```

Modify the mappings.

Definition at line 70 of file `adaboost_model.hpp`.

39.15.4.6 operator=()

```
AdaBoostModel& operator= (
    const AdaBoostModel & other )
```

Copy assignment operator.

39.15.4.7 `serialize()`

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the model.

Definition at line 94 of file `adaboost_model.hpp`.

39.15.4.8 `Train()`

```
void Train (
    const arma::mat & data,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const size_t iterations,
    const double tolerance )
```

Train the model.

Referenced by `AdaBoostModel::Dimensionality()`.

39.15.4.9 `WeakLearnerType()` [1/2]

```
size_t WeakLearnerType ( ) const [inline]
```

Get the weak learner type.

Definition at line 73 of file `adaboost_model.hpp`.

39.15.4.10 `WeakLearnerType()` [2/2]

```
size_t& WeakLearnerType ( ) [inline]
```

Modify the weak learner type.

Definition at line 75 of file `adaboost_model.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/adaboost/ adaboost_model.hpp`

39.16 AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType > Class Template Reference

This class implements **AMF** (p. 499) (alternating matrix factorization) on the given matrix V.

Public Member Functions

- **AMF** (const TerminationPolicyType &terminationPolicy=TerminationPolicyType(), const InitializationRuleType &initializeRule=InitializationRuleType(), const UpdateRuleType &update=UpdateRuleType())
*Create the **AMF** (p. 499) object and (optionally) set the parameters which **AMF** (p. 499) will run with.*
- template<typename MatType >
double **Apply** (const MatType &V, const size_t r, arma::mat &W, arma::mat &H)
Apply Alternating Matrix Factorization to the provided matrix.
- const InitializationRuleType & **InitializeRule** () const
Access the initialization rule.
- InitializationRuleType & **InitializeRule** ()
Modify the initialization rule.
- const TerminationPolicyType & **TerminationPolicy** () const
Access the termination policy.
- TerminationPolicyType & **TerminationPolicy** ()
Modify the termination policy.
- const UpdateRuleType & **Update** () const
Access the update rule.
- UpdateRuleType & **Update** ()
Modify the update rule.

39.16.1 Detailed Description

```
template<typename TerminationPolicyType = SimpleResidueTermination, typename InitializationRuleType = RandomAcol,
        Initialization<>, typename UpdateRuleType = NMFMultiplicativeDistanceUpdate>
class mlpack::amf::AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >
```

This class implements **AMF** (p. 499) (alternating matrix factorization) on the given matrix V.

Alternating matrix factorization decomposes V in the form $V \approx WH$ where W is called the basis matrix and H is called the encoding matrix. V is taken to be of size n x m and the obtained W is n x r and H is r x m. The size r is called the rank of the factorization.

The implementation requires three template types; the first contains the policy used to determine when the algorithm has converged; the second contains the initialization rule for the W and H matrix; the last contains the update rule to be used during each iteration. This templization allows the user to try various update rules, initialization rules, and termination policies (including ones not supplied with mlpack) for factorization. By default, the template parameters to **AMF** (p. 499) implement non-negative matrix factorization with the multiplicative distance update.

A simple example of how to run **AMF** (p. 499) (or NMF) is shown below.

```
extern arma::mat V; // Matrix that we want to perform LMF on.
size_t r = 10; // Rank of decomposition
arma::mat W; // Basis matrix
arma::mat H; // Encoding matrix

AMF<> amf; // Default options: NMF with multiplicative distance update rules.
amf.Apply(V, r, W, H);
```

Template Parameters

<i>TerminationPolicy</i>	The policy to use for determining when the factorization has converged.
<i>InitializationRule</i>	The initialization rule for initializing W and H matrix.
<i>UpdateRule</i>	The update rule for calculating W and H matrix at each iteration.

See also

NMFMultiplicativeDistanceUpdate (p. 519), **SimpleResidueTermination** (p. 529)

Definition at line 78 of file amf.hpp.

39.16.2 Constructor & Destructor Documentation

39.16.2.1 AMF()

```
AMF (
    const TerminationPolicyType & terminationPolicy = TerminationPolicyType(),
    const InitializationRuleType & initializeRule = InitializationRuleType(),
    const UpdateRuleType & update = UpdateRuleType() )
```

Create the **AMF** (p. 499) object and (optionally) set the parameters which **AMF** (p. 499) will run with.

The minimum residue refers to the root mean square of the difference between two subsequent iterations of the product $W * H$. A low residue indicates that subsequent iterations are not producing much change in W and H. Once the residue goes below the specified minimum residue, the algorithm terminates.

Parameters

<i>initializationRule</i>	Optional instantiated InitializationRule object for initializing the W and H matrices.
<i>updateRule</i>	Optional instantiated UpdateRule object; this parameter is useful when the update rule for the W and H vector has state that it needs to store (i.e. HUpdate() and WUpdate() are not static functions).
<i>terminationPolicy</i>	Optional instantiated TerminationPolicy object.

39.16.3 Member Function Documentation

39.16.3.1 Apply()

```
double Apply (
    const MatType & V,
    const size_t r,
    arma::mat & W,
    arma::mat & H )
```

Apply Alternating Matrix Factorization to the provided matrix.

Parameters

V	Input matrix to be factorized.
W	Basis matrix to be output.
H	Encoding matrix to output.
r	Rank r of the factorization.

Referenced by NMFPolicy::Apply(), BatchSVDPolicy::Apply(), SVDCompletePolicy::Apply(), and SVDIncompletePolicy::Apply().

39.16.3.2 InitializeRule() [1/2]

```
const InitializationRuleType& InitializeRule ( ) const [inline]
```

Access the initialization rule.

Definition at line 122 of file amf.hpp.

39.16.3.3 InitializeRule() [2/2]

```
InitializationRuleType& InitializeRule ( ) [inline]
```

Modify the initialization rule.

Definition at line 125 of file amf.hpp.

39.16.3.4 TerminationPolicy() [1/2]

```
const TerminationPolicyType& TerminationPolicy ( ) const [inline]
```

Access the termination policy.

Definition at line 116 of file amf.hpp.

39.16.3.5 TerminationPolicy() [2/2]

```
TerminationPolicyType& TerminationPolicy ( ) [inline]
```

Modify the termination policy.

Definition at line 119 of file amf.hpp.

39.16.3.6 Update() [1/2]

```
const UpdateRuleType& Update ( ) const [inline]
```

Access the update rule.

Definition at line 128 of file amf.hpp.

39.16.3.7 Update() [2/2]

```
UpdateRuleType& Update ( ) [inline]
```

Modify the update rule.

Definition at line 130 of file amf.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/ **amf.hpp**

39.17 AveragelInitialization Class Reference

This initialization rule initializes matrix W and H to root of the average of V, perturbed with uniform noise.

Public Member Functions

- **AveragelInitialization** ()
- `template<typename Archive >`
`void serialize (Archive &, const unsigned int)`
Serialize the object (in this case, there is nothing to do).

Static Public Member Functions

- `template<typename MatType >`
`static void Initialize (const MatType &V, const size_t r, arma::mat &W, arma::mat &H)`
Initialize the matrices W and H to the average value of V with uniform random noise added.

39.17.1 Detailed Description

This initialization rule initializes matrix W and H to root of the average of V, perturbed with uniform noise.

Uniform noise is generated by Armadillo's 'randu' function. For better performance, the lowest element of the matrix is subtracted from the average before dividing it by the factorization rank. This computed value is added with the random noise.

Definition at line 27 of file `average_init.hpp`.

39.17.2 Constructor & Destructor Documentation

39.17.2.1 AverageInitialization()

```
AverageInitialization ( ) [inline]
```

Definition at line 31 of file `average_init.hpp`.

39.17.3 Member Function Documentation

39.17.3.1 Initialize()

```
static void Initialize (
    const MatType & V,
    const size_t r,
    arma::mat & W,
    arma::mat & H ) [inline], [static]
```

Initialize the matrices W and H to the average value of V with uniform random noise added.

Parameters

<i>V</i>	Input matrix.
<i>r</i>	Rank of matrix.
<i>W</i>	W matrix, to be initialized.
<i>H</i>	H matrix, to be initialized.

Definition at line 43 of file average_init.hpp.

39.17.3.2 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serialize the object (in this case, there is nothing to do).

Definition at line 79 of file average_init.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/init_rules/ **average_init.hpp**

39.18 CompleteIncrementalTermination< TerminationPolicy > Class Template Reference

This class acts as a wrapper for basic termination policies to be used by **SVDCompleteIncrementalLearning** (p. 542).

Public Member Functions

- **CompleteIncrementalTermination** (TerminationPolicy tPolicy=TerminationPolicy())
Empty constructor.
- const double & **Index** () const
Get current value of residue.
- template<class MatType >
void **Initialize** (const MatType &V)
Initializes the termination policy before stating the factorization.
- void **Initialize** (const arma::sp_mat &V)
Initializes the termination policy before stating the factorization.
- bool **IsConverged** (arma::mat &W, arma::mat &H)
Check if termination criterion is met, if the current iteration means that each point has been visited.
- const size_t & **Iteration** () const
Get current iteration count.
- const size_t & **MaxIterations** () const
Access upper limit of iteration count.
- size_t & **MaxIterations** ()
Modify maximum number of iterations.
- const TerminationPolicy & **TPolicy** () const
Access the wrapped termination policy.
- TerminationPolicy & **TPolicy** ()
Modify the wrapped termination policy.

39.18.1 Detailed Description

```
template<class TerminationPolicy>
class mlpack::amf::CompleteIncrementalTermination< TerminationPolicy >
```

This class acts as a wrapper for basic termination policies to be used by **SVDCCompleteIncrementalLearning** (p. 542).

This class calls the wrapped class functions after every n calls to main class functions where n is the number of non-zero entries in the matrix being factorized. This is necessary for **SVDCCompleteIncrementalLearning** (p. 542), because otherwise **IsConverged()** (p. 506) is called after every point, which is very slow.

See also

AMF (p. 499), **SVDCCompleteIncrementalLearning** (p. 542)

Definition at line 29 of file complete_incremental_termination.hpp.

39.18.2 Constructor & Destructor Documentation

39.18.2.1 CompleteIncrementalTermination()

```
CompleteIncrementalTermination (
    TerminationPolicy tPolicy = TerminationPolicy() ) [inline]
```

Empty constructor.

Parameters

<i>tPolicy</i>	object of wrapped class.
----------------	--------------------------

Definition at line 37 of file complete_incremental_termination.hpp.

39.18.3 Member Function Documentation

39.18.3.1 Index()

```
const double& Index ( ) const [inline]
```

Get current value of residue.

Definition at line 92 of file complete_incremental_termination.hpp.

39.18.3.2 Initialize() [1/2]

```
void Initialize (
    const MatType & V ) [inline]
```

Initializes the termination policy before stating the factorization.

Parameters

<i>V</i>	Input matrix to be factorized.
----------	--------------------------------

Definition at line 47 of file complete_incremental_termination.hpp.

39.18.3.3 Initialize() [2/2]

```
void Initialize (
    const arma::sp_mat & V ) [inline]
```

Initializes the termination policy before stating the factorization.

This is a specialization for sparse matrices.

Parameters

<i>V</i>	Input matrix to be factorized.
----------	--------------------------------

Definition at line 62 of file complete_incremental_termination.hpp.

39.18.3.4 IsConverged()

```
bool IsConverged (
    arma::mat & W,
    arma::mat & H ) [inline]
```

Check if termination criterion is met, if the current iteration means that each point has been visited.

Parameters

<i>W</i>	Basis matrix of output.
<i>H</i>	Encoding matrix of output.

Definition at line 78 of file complete_incremental_termination.hpp.

39.18.3.5 Iteration()

```
const size_t& Iteration ( ) const [inline]
```

Get current iteration count.

Definition at line 95 of file complete_incremental_termination.hpp.

39.18.3.6 MaxIterations() [1/2]

```
const size_t& MaxIterations ( ) const [inline]
```

Access upper limit of iteration count.

Definition at line 98 of file complete_incremental_termination.hpp.

39.18.3.7 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Modify maximum number of iterations.

Definition at line 100 of file complete_incremental_termination.hpp.

39.18.3.8 TPolicy() [1/2]

```
const TerminationPolicy& TPolicy ( ) const [inline]
```

Access the wrapped termination policy.

Definition at line 103 of file complete_incremental_termination.hpp.

39.18.3.9 TPolicy() [2/2]

```
TerminationPolicy& TPolicy ( ) [inline]
```

Modify the wrapped termination policy.

Definition at line 105 of file complete_incremental_termination.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/ **complete_incremental_termination.hpp**↔

39.19 GivenInitialization Class Reference

This initialization rule for **AMF** (p. 499) simply fills the W and H matrices with the matrices given to the constructor of this object.

Public Member Functions

- **GivenInitialization** ()
- **GivenInitialization** (const arma::mat &w, const arma::mat &h)
- **GivenInitialization** (const arma::mat &&w, const arma::mat &&h)
- template<typename MatType >
void **Initialize** (const MatType &V, const size_t r, arma::mat &W, arma::mat &H)
Fill W and H with random uniform noise.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the object (in this case, there is nothing to serialize).

39.19.1 Detailed Description

This initialization rule for **AMF** (p. 499) simply fills the W and H matrices with the matrices given to the constructor of this object.

Note that this object does not use std::move() during the **Initialize()** (p. 509) method, so it can be reused for multiple **AMF** (p. 499) objects, but will incur copies of the W and H matrices.

Definition at line 27 of file given_init.hpp.

39.19.2 Constructor & Destructor Documentation

39.19.2.1 GivenInitialization() [1/3]

```
GivenInitialization ( ) [inline]
```

Definition at line 31 of file given_init.hpp.

39.19.2.2 GivenInitialization() [2/3]

```
GivenInitialization (
    const arma::mat & w,
    const arma::mat & h ) [inline]
```

Definition at line 34 of file given_init.hpp.

39.19.2.3 GivenInitialization() [3/3]

```
GivenInitialization (
    const arma::mat && w,
    const arma::mat && h ) [inline]
```

Definition at line 38 of file given_init.hpp.

39.19.3 Member Function Documentation**39.19.3.1 Initialize()**

```
void Initialize (
    const MatType & V,
    const size_t r,
    arma::mat & W,
    arma::mat & H ) [inline]
```

Fill W and H with random uniform noise.

Parameters

<i>V</i>	Input matrix.
<i>r</i>	Rank of decomposition.
<i>W</i>	W matrix, to be filled with random noise.
<i>H</i>	H matrix, to be filled with random noise.

Definition at line 52 of file given_init.hpp.

References Log::Fatal.

39.19.3.2 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the object (in this case, there is nothing to serialize).

Definition at line 90 of file given_init.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/init_rules/ **given_init.hpp**

39.20 IncompleteIncrementalTermination< TerminationPolicy > Class Template Reference

This class acts as a wrapper for basic termination policies to be used by **SVDIncompleteIncrementalLearning** (p. 547).

Public Member Functions

- **IncompleteIncrementalTermination** (TerminationPolicy tPolicy=TerminationPolicy())
Empty constructor.
- const double & **Index** () const
Get current value of residue.
- template<class MatType >
void **Initialize** (const MatType &V)
Initializes the termination policy before stating the factorization.
- bool **IsConverged** (arma::mat &W, arma::mat &H)
Check if termination criterio is met.
- const size_t & **Iteration** () const
Get current iteration count.
- size_t **MaxIterations** () const
Access maximum number of iterations.
- size_t & **MaxIterations** ()
Modify maximum number of iterations.
- const TerminationPolicy & **TPolicy** () const
Access the wrapped termination policy.
- TerminationPolicy & **TPolicy** ()
Modify the wrapped termination policy.

39.20.1 Detailed Description

```
template<class TerminationPolicy>
class mlpack::amf::IncompleteIncrementalTermination< TerminationPolicy >
```

This class acts as a wrapper for basic termination policies to be used by **SVDIncompleteIncrementalLearning** (p. 547).

This class calls the wrapped class functions after every *n* calls to main class functions where *n* is the number of rows.

See also

AMF (p. 499), **SVDIncompleteIncrementalLearning** (p. 547)

Definition at line 28 of file `incomplete_incremental_termination.hpp`.

39.20.2 Constructor & Destructor Documentation

39.20.2.1 IncompleteIncrementalTermination()

```
IncompleteIncrementalTermination (
    TerminationPolicy tPolicy = TerminationPolicy() ) [inline]
```

Empty constructor.

Parameters

<i>tPolicy</i>	object of wrapped class.
----------------	--------------------------

Definition at line 36 of file `incomplete_incremental_termination.hpp`.

39.20.3 Member Function Documentation

39.20.3.1 Index()

```
const double& Index ( ) const [inline]
```

Get current value of residue.

Definition at line 75 of file `incomplete_incremental_termination.hpp`.

39.20.3.2 Initialize()

```
void Initialize (
    const MatType & V ) [inline]
```

Initializes the termination policy before stating the factorization.

Parameters

<i>V</i>	Input matrix to be factorized.
----------	--------------------------------

Definition at line 46 of file incomplete_incremental_termination.hpp.

39.20.3.3 IsConverged()

```
bool IsConverged (
    arma::mat & W,
    arma::mat & H ) [inline]
```

Check if termination criterio is met.

Parameters

<i>W</i>	Basis matrix of output.
<i>H</i>	Encoding matrix of output.

Definition at line 61 of file incomplete_incremental_termination.hpp.

39.20.3.4 Iteration()

```
const size_t& Iteration ( ) const [inline]
```

Get current iteration count.

Definition at line 78 of file incomplete_incremental_termination.hpp.

39.20.3.5 MaxIterations() [1/2]

```
size_t MaxIterations ( ) const [inline]
```

Access maximum number of iterations.

Definition at line 81 of file incomplete_incremental_termination.hpp.

39.20.3.6 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Modify maximum number of iterations.

Definition at line 83 of file `incomplete_incremental_termination.hpp`.

39.20.3.7 TPolicy() [1/2]

```
const TerminationPolicy& TPolicy ( ) const [inline]
```

Access the wrapped termination policy.

Definition at line 86 of file `incomplete_incremental_termination.hpp`.

39.20.3.8 TPolicy() [2/2]

```
TerminationPolicy& TPolicy ( ) [inline]
```

Modify the wrapped termination policy.

Definition at line 88 of file `incomplete_incremental_termination.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/ incomplete_incremental_termination.hpp`↵

39.21 MaxIterationTermination Class Reference

This termination policy only terminates when the maximum number of iterations has been reached.

Public Member Functions

- **MaxIterationTermination** (const size_t maxIterations)
Construct the termination policy with the given number of iterations allowed (default 1000).
- size_t **Index** ()
Return something similar to the residue, which in this case is just the number of iterations left, since we don't have access to anything else.
- template<typename MatType >
void **Initialize** (const MatType &)
Initialize for the given matrix V (there is nothing to do).
- bool **IsConverged** (const arma::mat &, const arma::mat &)
Check if convergence has occurred.
- size_t **Iteration** () const
Get the current iteration.
- size_t & **Iteration** ()
Modify the current iteration.
- size_t **MaxIterations** () const
Get the maximum number of iterations.
- size_t & **MaxIterations** ()
Modify the maximum number of iterations.

39.21.1 Detailed Description

This termination policy only terminates when the maximum number of iterations has been reached.

Definition at line 23 of file max_iteration_termination.hpp.

39.21.2 Constructor & Destructor Documentation

39.21.2.1 MaxIterationTermination()

```
MaxIterationTermination (  
    const size_t maxIterations ) [inline]
```

Construct the termination policy with the given number of iterations allowed (default 1000).

If maxIterations is 0, then termination will never occur.

Parameters

<i>maxIterations</i>	Maximum number of allowed iterations.
----------------------	---------------------------------------

Definition at line 33 of file max_iteration_termination.hpp.

References Log::Warn.

39.21.3 Member Function Documentation

39.21.3.1 Index()

```
size_t Index ( ) [inline]
```

Return something similar to the residue, which in this case is just the number of iterations left, since we don't have access to anything else.

Definition at line 60 of file max_iteration_termination.hpp.

39.21.3.2 Initialize()

```
void Initialize (
    const MatType & ) [inline]
```

Initialize for the given matrix V (there is nothing to do).

Definition at line 47 of file max_iteration_termination.hpp.

39.21.3.3 IsConverged()

```
bool IsConverged (
    const arma::mat & ,
    const arma::mat & ) [inline]
```

Check if convergence has occurred.

Definition at line 52 of file max_iteration_termination.hpp.

39.21.3.4 Iteration() [1/2]

```
size_t Iteration ( ) const [inline]
```

Get the current iteration.

Definition at line 66 of file max_iteration_termination.hpp.

39.21.3.5 Iteration() [2/2]

```
size_t& Iteration ( ) [inline]
```

Modify the current iteration.

Definition at line 68 of file max_iteration_termination.hpp.

39.21.3.6 MaxIterations() [1/2]

```
size_t MaxIterations ( ) const [inline]
```

Get the maximum number of iterations.

Definition at line 71 of file max_iteration_termination.hpp.

39.21.3.7 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Modify the maximum number of iterations.

Definition at line 73 of file max_iteration_termination.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/ **max_iteration_termination.**↔
hpp

39.22 NMFALSUpdate Class Reference

This class implements a method titled 'Alternating Least Squares' described in the following paper:

Public Member Functions

- **NMFALSUpdate ()**
Empty constructor required for the UpdateRule template.
- template<typename MatType >
void **Initialize** (const MatType &, const size_t)
Set initial values for the factorization.
- template<typename Archive >
void **serialize** (Archive &, const unsigned int)
Serialize the object (in this case, there is nothing to serialize).

Static Public Member Functions

- template<typename MatType >
static void **HUpdate** (const MatType &V, const arma::mat &W, arma::mat &H)
The update rule for the encoding matrix H.
- template<typename MatType >
static void **WUpdate** (const MatType &V, arma::mat &W, const arma::mat &H)
The update rule for the basis matrix W.

39.22.1 Detailed Description

This class implements a method titled 'Alternating Least Squares' described in the following paper:

```
@article{paatero1994positive,
  title={Positive matrix factorization: A non-negative factor model with
    optimal utilization of error estimates of data values},
  author={Paatero, P. and Tapper, U.},
  journal={Environmetrics},
  volume={5},
  number={2},
  pages={111--126},
  year={1994}
}
```

It uses the least squares projection formula to reduce the error value of $\sqrt{\sum_i \sum_j (V - WH)^2}$ by alternately calculating W and H respectively while holding the other matrix constant.

Definition at line 41 of file nmf_als.hpp.

39.22.2 Constructor & Destructor Documentation

39.22.2.1 NMFALSUpdate()

```
NMFALSUpdate ( ) [inline]
```

Empty constructor required for the UpdateRule template.

Definition at line 45 of file nmf_als.hpp.

39.22.3 Member Function Documentation

39.22.3.1 HUpdate()

```
static void HUpdate (
    const MatType & V,
    const arma::mat & W,
    arma::mat & H ) [inline], [static]
```

The update rule for the encoding matrix H.

The formula used is

$$H = \frac{W^T V}{W^T W}$$

The function takes in all the matrices and only changes the value of the H matrix.

Parameters

<i>V</i>	Input matrix to be factorized.
<i>W</i>	Basis matrix.
<i>H</i>	Encoding matrix to be updated.

Definition at line 105 of file nmf_als.hpp.

39.22.3.2 Initialize()

```
void Initialize (
    const MatType & ,
    const size_t ) [inline]
```

Set initial values for the factorization.

In this case, we don't need to set anything.

Definition at line 52 of file nmf_als.hpp.

39.22.3.3 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serialize the object (in this case, there is nothing to serialize).

Definition at line 123 of file nmf_als.hpp.

39.22.3.4 WUpdate()

```
static void WUpdate (
    const MatType & V,
    arma::mat & W,
    const arma::mat & H ) [inline], [static]
```

The update rule for the basis matrix W.

The formula used is

$$W^T = \frac{HV^T}{HH^T}$$

The function takes in all the matrices and only changes the value of the W matrix.

Parameters

<i>V</i>	Input matrix to be factorized.
<i>W</i>	Basis matrix to be updated.
<i>H</i>	Encoding matrix.

Definition at line 72 of file nmf_als.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/ **nmf_als.hpp**

39.23 NMFMultiplicativeDistanceUpdate Class Reference

The multiplicative distance update rules for matrices W and H.

Public Member Functions

- **NMFMultiplicativeDistanceUpdate** ()
- template<typename MatType >
void **Initialize** (const MatType &, const size_t)
Initialize the factorization.
- template<typename Archive >
void **serialize** (Archive &, const unsigned int)
Serialize the object (in this case, there is nothing to serialize).

Static Public Member Functions

- `template<typename MatType >`
`static void HUpdate (const MatType &V, const arma::mat &W, arma::mat &H)`
The update rule for the encoding matrix H.
- `template<typename MatType >`
`static void WUpdate (const MatType &V, arma::mat &W, const arma::mat &H)`
The update rule for the basis matrix W.

39.23.1 Detailed Description

The multiplicative distance update rules for matrices W and H.

This follows a method described in the following paper:

```
@inproceedings{lee2001algorithms,
  title={Algorithms for non-negative matrix factorization},
  author={Lee, D.D. and Seung, H.S.},
  booktitle={Advances in Neural Information Processing Systems 13
    (NIPS 2000)},
  pages={556--562},
  year={2001}
}
```

This is a multiplicative rule that ensures that the Frobenius norm $\sqrt{\sum_i \sum_j (V - WH)^2}$ is non-increasing between subsequent iterations. Both of the update rules for W and H are defined in this file.

Definition at line 39 of file `nmf_mult_dist.hpp`.

39.23.2 Constructor & Destructor Documentation

39.23.2.1 NMFMultiplicativeDistanceUpdate()

```
NFMultiplicativeDistanceUpdate ( ) [inline]
```

Definition at line 43 of file `nmf_mult_dist.hpp`.

39.23.3 Member Function Documentation

39.23.3.1 HUpdate()

```
static void HUpdate (
    const MatType & V,
    const arma::mat & W,
    arma::mat & H ) [inline], [static]
```

The update rule for the encoding matrix H.

The formula used is

$$H_{a\mu} \leftarrow H_{a\mu} \frac{(W^T V)_{a\mu}}{(W^T W H)_{a\mu}}$$

The function takes in all the matrices and only changes the value of the H matrix.

Parameters

V	Input matrix to be factorized.
W	Basis matrix.
H	Encoding matrix to be updated.

Definition at line 92 of file nmf_mult_dist.hpp.

39.23.3.2 Initialize()

```
void Initialize (
    const MatType & ,
    const size_t ) [inline]
```

Initialize the factorization.

These update rules hold no information, so the input parameters are ignored.

Definition at line 50 of file nmf_mult_dist.hpp.

39.23.3.3 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serialize the object (in this case, there is nothing to serialize).

Definition at line 101 of file nmf_mult_dist.hpp.

39.23.3.4 WUpdate()

```
static void WUpdate (
    const MatType & V,
    arma::mat & W,
    const arma::mat & H ) [inline], [static]
```

The update rule for the basis matrix W.

The formula used is

$$W_{ia} \leftarrow W_{ia} \frac{(VH^T)_{ia}}{(WHH^T)_{ia}}$$

The function takes in all the matrices and only changes the value of the W matrix.

Parameters

V	Input matrix to be factorized.
W	Basis matrix to be updated.
H	Encoding matrix.

Definition at line 70 of file nmf_mult_dist.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/ **nmf_mult_dist.hpp**

39.24 NMFMultiplicativeDivergenceUpdate Class Reference

This follows a method described in the paper 'Algorithms for Non-negative.

Public Member Functions

- **NMFMultiplicativeDivergenceUpdate** ()
- template<typename MatType >
void **Initialize** (const MatType &, const size_t)
Initialize the factorization.
- template<typename Archive >
void **serialize** (Archive &, const unsigned int)
Serialize the object (in this case, there is nothing to serialize).

Static Public Member Functions

- template<typename MatType >
static void **HUpdate** (const MatType &V, const arma::mat &W, arma::mat &H)
The update rule for the encoding matrix H.
- template<typename MatType >
static void **WUpdate** (const MatType &V, arma::mat &W, const arma::mat &H)
The update rule for the basis matrix W.

39.24.1 Detailed Description

This follows a method described in the paper 'Algorithms for Non-negative.

```
@inproceedings{lee2001algorithms,
  title={Algorithms for non-negative matrix factorization},
  author={Lee, D.D. and Seung, H.S.},
  booktitle={Advances in Neural Information Processing Systems 13
    (NIPS 2000)},
  pages={556--562},
  year={2001}
}
```

This is a multiplicative rule that ensures that the Kullback–Leibler divergence

$$\sum_i \sum_j (V_{ij} \log \frac{V_{ij}}{(WH)_{ij}} - V_{ij} + (WH)_{ij})$$

is non-increasing between subsequent iterations. Both of the update rules for W and H are defined in this file.

This set of update rules is not meant to work with sparse matrices. Using sparse matrices often causes NaNs in the output, so other choices of update rules are better in that situation.

Definition at line 48 of file nmf_mult_div.hpp.

39.24.2 Constructor & Destructor Documentation

39.24.2.1 NMFMultiplicativeDivergenceUpdate()

```
NMFMultiplicativeDivergenceUpdate ( ) [inline]
```

Definition at line 52 of file nmf_mult_div.hpp.

39.24.3 Member Function Documentation

39.24.3.1 HUpdate()

```
static void HUpdate (
    const MatType & V,
    const arma::mat & W,
    arma::mat & H ) [inline], [static]
```

The update rule for the encoding matrix H.

The formula used is

$$H_{a\mu} \leftarrow H_{a\mu} \frac{\sum_i W_{ia} V_{i\mu} / (WH)_{i\mu}}{\sum_k H_{ka}}$$

The function takes in all the matrices and only changes the value of the H matrix.

Parameters

V	Input matrix to be factorized.
W	Basis matrix.
H	Encoding matrix to updated.

Definition at line 124 of file nmf_mult_div.hpp.

39.24.3.2 Initialize()

```
void Initialize (
    const MatType & ,
    const size_t ) [inline]
```

Initialize the factorization.

These rules don't store any state, so the input values are ignore.

Definition at line 59 of file nmf_mult_div.hpp.

39.24.3.3 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serialize the object (in this case, there is nothing to serialize).

Definition at line 154 of file nmf_mult_div.hpp.

39.24.3.4 WUpdate()

```
static void WUpdate (
    const MatType & V,
    arma::mat & W,
    const arma::mat & H ) [inline], [static]
```

The update rule for the basis matrix W.

The formula used is

$$W_{ia} \leftarrow W_{ia} \frac{\sum_{\mu} H_{a\mu} V_{i\mu} / (WH)_{i\mu}}{\sum_{\nu} H_{a\nu}}$$

The function takes in all the matrices and only changes the value of the W matrix.

Parameters

V	Input matrix to be factorized.
W	Basis matrix to be updated.
H	Encoding matrix.

Definition at line 80 of file nmf_mult_div.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/ **nmf_mult_div.hpp**

39.25 RandomAcolInitialization< columnsToAverage > Class Template Reference

This class initializes the W matrix of the **AMF** (p. 499) algorithm by averaging p randomly chosen columns of V .

Public Member Functions

- **RandomAcolInitialization** ()
- template<typename Archive >
void **serialize** (Archive &, const unsigned int)
Serialize the object (in this case, there is nothing to serialize).

Static Public Member Functions

- template<typename MatType >
static void **Initialize** (const MatType & V , const size_t r , arma::mat & W , arma::mat & H)

39.25.1 Detailed Description

```
template<size_t columnsToAverage = 5>
class mlpack::amf::RandomAcolInitialization< columnsToAverage >
```

This class initializes the W matrix of the **AMF** (p. 499) algorithm by averaging p randomly chosen columns of V .

In this case, p is a template parameter. H is then filled using a uniform distribution in the range $[0, 1]$.

This simple initialization is the "random Acol initialization" found in the following paper:

```
@techreport{langville2014algorithms,
  title = {Algorithms, Initializations, and Convergence for the Nonnegative
    Matrix Factorization},
  author = {Langville, A.N. and Meyer, C.D. and Albright, R. and Cox, J. and
    Duling, D.},
  year = {2014},
  institution = {NCSU Technical Report Math 81706}
}
```

Template Parameters

<i>columnsToAverage</i>	The number of random columns to average for each column of <i>W</i> .
-------------------------	---

Definition at line 44 of file random_acol_init.hpp.

39.25.2 Constructor & Destructor Documentation

39.25.2.1 RandomAcolInitialization()

```
RandomAcolInitialization ( ) [inline]
```

Definition at line 48 of file random_acol_init.hpp.

39.25.3 Member Function Documentation

39.25.3.1 Initialize()

```
static void Initialize (
    const MatType & V,
    const size_t r,
    arma::mat & W,
    arma::mat & H ) [inline], [static]
```

Definition at line 52 of file random_acol_init.hpp.

References `mlpack::math::RandInt()`, and `Log::Warn`.

39.25.3.2 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serialize the object (in this case, there is nothing to serialize).

Definition at line 88 of file random_acol_init.hpp.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/init_rules/ random_acol_init.hpp`

39.26 RandomInitialization Class Reference

This initialization rule for **AMF** (p. 499) simply fills the W and H matrices with uniform random noise in $[0, 1]$.

Public Member Functions

- **RandomInitialization** ()
- `template<typename Archive >`
`void serialize (Archive &, const unsigned int)`
Serialize the object (in this case, there is nothing to serialize).

Static Public Member Functions

- `template<typename MatType >`
`static void Initialize (const MatType &V, const size_t r, arma::mat &W, arma::mat &H)`
Fill W and H with random uniform noise.

39.26.1 Detailed Description

This initialization rule for **AMF** (p. 499) simply fills the W and H matrices with uniform random noise in $[0, 1]$.

Definition at line 25 of file `random_init.hpp`.

39.26.2 Constructor & Destructor Documentation

39.26.2.1 RandomInitialization()

```
RandomInitialization ( ) [inline]
```

Definition at line 29 of file `random_init.hpp`.

39.26.3 Member Function Documentation

39.26.3.1 Initialize()

```
static void Initialize (
    const MatType & V,
    const size_t r,
    arma::mat & W,
    arma::mat & H ) [inline], [static]
```

Fill W and H with random uniform noise.

Parameters

V	Input matrix.
r	Rank of decomposition.
W	W matrix, to be filled with random noise.
H	H matrix, to be filled with random noise.

Definition at line 40 of file random_init.hpp.

39.26.3.2 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serialize the object (in this case, there is nothing to serialize).

Definition at line 56 of file random_init.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/init_rules/ **random_init.hpp**

39.27 SimpleResidueTermination Class Reference

This class implements a simple residue-based termination policy.

Public Member Functions

- **SimpleResidueTermination** (const double **minResidue**=1e-5, const size_t maxIterations=10000)
*Construct the **SimpleResidueTermination** (p. 529) object with the given minimum residue (or the default) and the given maximum number of iterations (or the default).*
- const double & **Index** () const
Get current value of residue.
- template<typename MatType >
void **Initialize** (const MatType &V)
Initializes the termination policy before stating the factorization.
- bool **IsConverged** (arma::mat &W, arma::mat &H)
Check if termination criterion is met.
- const size_t & **Iteration** () const
Get current iteration count.
- const size_t & **MaxIterations** () const
Access max iteration count.
- size_t & **MaxIterations** ()
- const double & **MinResidue** () const
Access minimum residue value.
- double & **MinResidue** ()

Public Attributes

- `size_t iteration`
current iteration count
- `size_t maxIterations`
iteration threshold
- `double minResidue`
residue threshold
- `size_t nm`
- `double normOld`
norm of previous iteration
- `double residue`
current value of residue

39.27.1 Detailed Description

This class implements a simple residue-based termination policy.

The termination decision depends on two factors: the value of the residue (the difference between the norm of WH this iteration and the previous iteration), and the number of iterations. If the current value of residue drops below the threshold or the number of iterations goes above the iteration limit, **IsConverged()** (p. 531) will return true. This class is meant for use with the **AMF** (p. 499) (alternating matrix factorization) class.

See also

AMF (p. 499)

Definition at line 31 of file `simple_residue_termination.hpp`.

39.27.2 Constructor & Destructor Documentation

39.27.2.1 SimpleResidueTermination()

```
SimpleResidueTermination (  
    const double minResidue = 1e-5,  
    const size_t maxIterations = 10000 ) [inline]
```

Construct the **SimpleResidueTermination** (p. 529) object with the given minimum residue (or the default) and the given maximum number of iterations (or the default).

0 indicates no iteration limit.

Parameters

<i>minResidue</i>	Minimum residue for termination.
<i>maxIterations</i>	Maximum number of iterations.

Definition at line 42 of file simple_residue_termination.hpp.

39.27.3 Member Function Documentation**39.27.3.1 Index()**

```
const double& Index ( ) const [inline]
```

Get current value of residue.

Definition at line 90 of file simple_residue_termination.hpp.

References SimpleResidueTermination::residue.

39.27.3.2 Initialize()

```
void Initialize (
    const MatType & V ) [inline]
```

Initializes the termination policy before stating the factorization.

Parameters

<i>V</i>	Input matrix being factorized.
----------	--------------------------------

Definition at line 52 of file simple_residue_termination.hpp.

References SimpleResidueTermination::iteration, SimpleResidueTermination::nm, SimpleResidueTermination::norm↔Old, and SimpleResidueTermination::residue.

39.27.3.3 IsConverged()

```
bool IsConverged (
    arma::mat & W,
    arma::mat & H ) [inline]
```

Check if termination criterion is met.

Parameters

W	Basis matrix of output.
H	Encoding matrix of output.

Definition at line 68 of file simple_residue_termination.hpp.

References [Log::Info](#), [SimpleResidueTermination::iteration](#), [SimpleResidueTermination::maxIterations](#), [SimpleResidueTermination::minResidue](#), [SimpleResidueTermination::normOld](#), and [SimpleResidueTermination::residue](#).

39.27.3.4 Iteration()

```
const size_t& Iteration ( ) const [inline]
```

Get current iteration count.

Definition at line 93 of file simple_residue_termination.hpp.

References [SimpleResidueTermination::iteration](#).

39.27.3.5 MaxIterations() [1/2]

```
const size_t& MaxIterations ( ) const [inline]
```

Access max iteration count.

Definition at line 96 of file simple_residue_termination.hpp.

References [SimpleResidueTermination::maxIterations](#).

39.27.3.6 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Definition at line 97 of file simple_residue_termination.hpp.

References [SimpleResidueTermination::maxIterations](#).

39.27.3.7 MinResidue() [1/2]

```
const double& MinResidue ( ) const [inline]
```

Access minimum residue value.

Definition at line 100 of file simple_residue_termination.hpp.

References SimpleResidueTermination::minResidue.

39.27.3.8 MinResidue() [2/2]

```
double& MinResidue ( ) [inline]
```

Definition at line 101 of file simple_residue_termination.hpp.

References SimpleResidueTermination::minResidue.

39.27.4 Member Data Documentation

39.27.4.1 iteration

```
size_t iteration
```

current iteration count

Definition at line 112 of file simple_residue_termination.hpp.

Referenced by SimpleResidueTermination::Initialize(), SimpleResidueTermination::IsConverged(), and SimpleResidueTermination::Iteration().

39.27.4.2 maxIterations

```
size_t maxIterations
```

iteration threshold

Definition at line 107 of file simple_residue_termination.hpp.

Referenced by SimpleResidueTermination::IsConverged(), and SimpleResidueTermination::MaxIterations().

39.27.4.3 minResidue

`double minResidue`

residue threshold

Definition at line 105 of file `simple_residue_termination.hpp`.

Referenced by `SimpleResidueTermination::IsConverged()`, and `SimpleResidueTermination::MinResidue()`.

39.27.4.4 nm

`size_t nm`

Definition at line 116 of file `simple_residue_termination.hpp`.

Referenced by `SimpleResidueTermination::Initialize()`.

39.27.4.5 normOld

`double normOld`

norm of previous iteration

Definition at line 114 of file `simple_residue_termination.hpp`.

Referenced by `SimpleResidueTermination::Initialize()`, and `SimpleResidueTermination::IsConverged()`.

39.27.4.6 residue

`double residue`

current value of residue

Definition at line 110 of file `simple_residue_termination.hpp`.

Referenced by `SimpleResidueTermination::Index()`, `SimpleResidueTermination::Initialize()`, and `SimpleResidueTermination::IsConverged()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/termination.hpp`

simple_residue_↵

39.28 SimpleToleranceTermination< MatType > Class Template Reference

This class implements residue tolerance termination policy.

Public Member Functions

- **SimpleToleranceTermination** (const double tolerance=1e-5, const size_t maxIterations=10000, const size_t reverseStepTolerance=3)
empty constructor
- const double & **Index** () const
Get current value of residue.
- void **Initialize** (const MatType &V)
Initializes the termination policy before stating the factorization.
- bool **IsConverged** (arma::mat &W, arma::mat &H)
Check if termination criterio is met.
- const size_t & **Iteration** () const
Get current iteration count.
- const size_t & **MaxIterations** () const
Access upper limit of iteration count.
- size_t & **MaxIterations** ()
- const double & **Tolerance** () const
Access tolerance value.
- double & **Tolerance** ()

39.28.1 Detailed Description

```
template<class MatType>
class mlpack::amf::SimpleToleranceTermination< MatType >
```

This class implements residue tolerance termination policy.

Termination criterion is met when increase in residue value drops below the given tolerance. To accommodate spikes certain number of successive residue drops are accepted. This upper imit on successive drops can be adjusted with reverseStepCount. Secondary termination criterion terminates algorithm when iteration count goes above the threshold.

See also

AMF (p. 499)

Definition at line 31 of file simple_tolerance_termination.hpp.

39.28.2 Constructor & Destructor Documentation

39.28.2.1 SimpleToleranceTermination()

```
SimpleToleranceTermination (
    const double tolerance = 1e-5,
    const size_t maxIterations = 10000,
    const size_t reverseStepTolerance = 3 ) [inline]
```

empty constructor

Definition at line 35 of file simple_tolerance_termination.hpp.

39.28.3 Member Function Documentation

39.28.3.1 Index()

```
const double& Index ( ) const [inline]
```

Get current value of residue.

Definition at line 149 of file simple_tolerance_termination.hpp.

39.28.3.2 Initialize()

```
void Initialize (
    const MatType & V ) [inline]
```

Initializes the termination policy before stating the factorization.

Parameters

V	Input matrix to be factorized.
-----	--------------------------------

Definition at line 47 of file simple_tolerance_termination.hpp.

39.28.3.3 IsConverged()

```
bool IsConverged (
    arma::mat & W,
    arma::mat & H ) [inline]
```


Check if termination criterio is met.

Parameters

<i>W</i>	Basis matrix of output.
<i>H</i>	Encoding matrix of output.

Definition at line 69 of file simple_tolerance_termination.hpp.

References [Log::Info](#).

39.28.3.4 Iteration()

```
const size_t& Iteration ( ) const [inline]
```

Get current iteration count.

Definition at line 152 of file simple_tolerance_termination.hpp.

39.28.3.5 MaxIterations() [1/2]

```
const size_t& MaxIterations ( ) const [inline]
```

Access upper limit of iteration count.

Definition at line 155 of file simple_tolerance_termination.hpp.

39.28.3.6 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Definition at line 156 of file simple_tolerance_termination.hpp.

39.28.3.7 Tolerance() [1/2]

```
const double& Tolerance ( ) const [inline]
```

Access tolerance value.

Definition at line 159 of file simple_tolerance_termination.hpp.

39.28.3.8 Tolerance() [2/2]

```
double& Tolerance ( ) [inline]
```

Definition at line 160 of file simple_tolerance_termination.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/**simple_tolerance_termination.hpp** ↩

39.29 SVDBatchLearning Class Reference

This class implements SVD batch learning with momentum.

Public Member Functions

- **SVDBatchLearning** (double u=0.0002, double kw=0, double kh=0, double momentum=0.9)
SVD Batch learning constructor.
- template<typename MatType >
void **HUpdate** (const MatType &V, const arma::mat &W, arma::mat &H)
The update rule for the encoding matrix H.
- template<typename MatType >
void **Initialize** (const MatType &dataset, const size_t rank)
Initialize parameters before factorization.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the SVDBatch object.
- template<typename MatType >
void **WUpdate** (const MatType &V, arma::mat &W, const arma::mat &H)
The update rule for the basis matrix W.

39.29.1 Detailed Description

This class implements SVD batch learning with momentum.

This procedure is described in the following paper:

```
@techreport{ma2008guide,
  title={A Guide to Singular Value Decomposition for Collaborative
    Filtering},
  author={Ma, Chih-Chao},
  year={2008},
  institution={Department of Computer Science, National Taiwan University}
}
```

This class implements 'Algorithm 4' as given in the paper.

The factorizer decomposes the matrix V into two matrices W and H such that sum of sum of squared error between V and W * H is minimum. This optimization is performed with gradient descent. To make gradient descent faster, momentum is added.

Definition at line 41 of file svd_batch_learning.hpp.

39.29.2 Constructor & Destructor Documentation

39.29.2.1 SVDBatchLearning()

```
SVDBatchLearning (
    double u = 0.0002,
    double kw = 0,
    double kh = 0,
    double momentum = 0.9 ) [inline]
```

SVD Batch learning constructor.

Parameters

<i>u</i>	step value used in batch learning
<i>kw</i>	regularization constant for W matrix
<i>kh</i>	regularization constant for H matrix
<i>momentum</i>	momentum applied to batch learning process

Definition at line 52 of file `svd_batch_learning.hpp`.

39.29.3 Member Function Documentation

39.29.3.1 HUpdate()

```
void HUpdate (
    const MatType & V,
    const arma::mat & W,
    arma::mat & H ) [inline]
```

The update rule for the encoding matrix H.

The function takes in all the matrices and only changes the value of the H matrix.

Parameters

<i>V</i>	Input matrix to be factorized.
<i>W</i>	Basis matrix.
<i>H</i>	Encoding matrix to be updated.

Definition at line 133 of file `svd_batch_learning.hpp`.

39.29.3.2 Initialize()

```
void Initialize (
    const MatType & dataset,
    const size_t rank ) [inline]
```

Initialize parameters before factorization.

This function must be called before a new factorization. This resets the internally-held momentum.

Parameters

<i>dataset</i>	Input matrix to be factorized.
<i>rank</i>	rank of factorization

Definition at line 69 of file `svd_batch_learning.hpp`.

39.29.3.3 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the SVDBatch object.

Definition at line 169 of file `svd_batch_learning.hpp`.

39.29.3.4 WUpdate()

```
void WUpdate (
    const MatType & V,
    arma::mat & W,
    const arma::mat & H ) [inline]
```

The update rule for the basis matrix W .

The function takes in all the matrices and only changes the value of the W matrix.

Parameters

V	Input matrix to be factorized.
W	Basis matrix to be updated.
H	Encoding matrix.

Definition at line 88 of file `svd_batch_learning.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/ svd_batch_learning.hpp`

39.30 **SVDCompleteIncrementalLearning**< MatType > Class Template Reference

This class computes SVD using complete incremental batch learning, as described in the following paper:

Public Member Functions

- **SVDCompleteIncrementalLearning** (double $u=0.0001$, double $kw=0$, double $kh=0$)
*Initialize the **SVDCompleteIncrementalLearning** (p. 542) class with the given parameters.*
- void **HUpdate** (const MatType &V, const arma::mat &W, arma::mat &H)
The update rule for the encoding matrix H .
- void **Initialize** (const MatType &, const size_t)
Initialize parameters before factorization.
- void **WUpdate** (const MatType &V, arma::mat &W, const arma::mat &H)
The update rule for the basis matrix W .

39.30.1 Detailed Description

```
template<class MatType>
class mlpack::amf::SVDCompleteIncrementalLearning< MatType >
```

This class computes SVD using complete incremental batch learning, as described in the following paper:

```
@techreport{ma2008guide,
  title={A Guide to Singular Value Decomposition for Collaborative
    Filtering},
  author={Ma, Chih-Chao},
  year={2008},
  institution={Department of Computer Science, National Taiwan University}
}
```

This class implements 'Algorithm 3' given in the paper. Complete incremental learning is an extreme case of incremental learning, where feature vectors are updated after looking at each single element in the input matrix (V). This approach differs from incomplete incremental learning where feature vectors are updated after seeing columns of elements in the input matrix.

See also

SVDIncompleteIncrementalLearning (p. 547)

Definition at line 45 of file `svd_complete_incremental_learning.hpp`.

39.30.2 Constructor & Destructor Documentation

39.30.2.1 SVDCompleteIncrementalLearning()

```
SVDCompleteIncrementalLearning (
    double u = 0.0001,
    double kw = 0,
    double kh = 0 ) [inline]
```

Initialize the **SVDCompleteIncrementalLearning** (p. 542) class with the given parameters.

Parameters

<i>u</i>	Step value used in batch learning.
<i>kw</i>	Regularization constant for W matrix.
<i>kh</i>	Regularization constant for H matrix.

Definition at line 56 of file svd_complete_incremental_learning.hpp.

39.30.3 Member Function Documentation

39.30.3.1 HUpdate()

```
void HUpdate (
    const MatType & V,
    const arma::mat & W,
    arma::mat & H ) [inline]
```

The update rule for the encoding matrix H.

The function takes in all the matrices and only changes the value of the H matrix.

Parameters

<i>V</i>	Input matrix to be factorized.
<i>W</i>	Basis matrix.
<i>H</i>	Encoding matrix to be updated.

Definition at line 123 of file svd_complete_incremental_learning.hpp.

39.30.3.2 Initialize()

```
void Initialize (
    const MatType & ,
    const size_t ) [inline]
```

Initialize parameters before factorization.

This function must be called before a new factorization. For this initialization, the input parameters are unnecessary; we are only setting the current element index to 0.

Parameters

<i>dataset</i>	Input matrix to be factorized.
<i>rank</i>	rank of factorization

Definition at line 72 of file `svd_complete_incremental_learning.hpp`.

39.30.3.3 WUpdate()

```
void WUpdate (
    const MatType & V,
    arma::mat & W,
    const arma::mat & H ) [inline]
```

The update rule for the basis matrix W .

The function takes in all the matrices and only changes the value of the W matrix.

Parameters

V	Input matrix to be factorized.
W	Basis matrix to be updated.
H	Encoding matrix.

Definition at line 87 of file `svd_complete_incremental_learning.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/ svd_complete_incremental_↵
learning.hpp`

39.31 SVDCompleteIncrementalLearning< arma::sp_mat > Class Template Reference

TODO : Merge this template specialized function for sparse matrix using common `row_col_iterator`.

Public Member Functions

- **SVDCompleteIncrementalLearning** (double u=0.01, double kw=0, double kh=0)
- **~SVDCompleteIncrementalLearning** ()
- void **HUpdate** (const arma::sp_mat &, const arma::mat &W, arma::mat &H)
The update rule for the encoding matrix H.
- void **Initialize** (const arma::sp_mat &dataset, const size_t rank)
- void **WUpdate** (const arma::sp_mat &V, arma::mat &W, const arma::mat &H)
The update rule for the basis matrix W.

39.31.1 Detailed Description

```
template<>
class mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >
```

TODO : Merge this template specialized function for sparse matrix using common row_col_iterator.

template specialized functions for sparse matrices

Definition at line 169 of file svd_complete_incremental_learning.hpp.

39.31.2 Constructor & Destructor Documentation

39.31.2.1 SVDCompleteIncrementalLearning()

```
SVDCompleteIncrementalLearning (
    double u = 0.01,
    double kw = 0,
    double kh = 0 ) [inline]
```

Definition at line 172 of file svd_complete_incremental_learning.hpp.

39.31.2.2 ~SVDCompleteIncrementalLearning()

```
~SVDCompleteIncrementalLearning ( ) [inline]
```

Definition at line 178 of file svd_complete_incremental_learning.hpp.

39.31.3 Member Function Documentation

39.31.3.1 HUpdate()

```
void HUpdate (
    const arma::sp_mat & ,
    const arma::mat & W,
    arma::mat & H ) [inline]
```

The update rule for the encoding matrix *H*.

The function takes in all the matrices and only changes the value of the *H* matrix.

Parameters

<i>V</i>	Input matrix to be factorized.
<i>W</i>	Basis matrix.
<i>H</i>	Encoding matrix to be updated.

Definition at line 238 of file `svd_complete_incremental_learning.hpp`.

39.31.3.2 Initialize()

```
void Initialize (
    const arma::sp_mat & dataset,
    const size_t rank ) [inline]
```

Definition at line 183 of file `svd_complete_incremental_learning.hpp`.

39.31.3.3 WUpdate()

```
void WUpdate (
    const arma::sp_mat & V,
    arma::mat & W,
    const arma::mat & H ) [inline]
```

The update rule for the basis matrix *W*.

The function takes in all the matrices and only changes the value of the *W* matrix.

Parameters

<i>V</i>	Input matrix to be factorized.
<i>W</i>	Basis matrix to be updated.
<i>H</i>	Encoding matrix.

Definition at line 202 of file `svd_complete_incremental_learning.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/ svd_complete_incremental_↵
learning.hpp`

39.32 SVDIncompleteIncrementalLearning Class Reference

This class computes SVD using incomplete incremental batch learning, as described in the following paper:

Public Member Functions

- **SVDIncompleteIncrementalLearning** (double $u=0.001$, double $kw=0$, double $kh=0$)
*Initialize the parameters of **SVDIncompleteIncrementalLearning** (p. 547).*
- `template<typename MatType >`
`void HUpdate (const MatType &V, const arma::mat &W, arma::mat &H)`
The update rule for the encoding matrix H.
- `template<typename MatType >`
`void Initialize (const MatType &, const size_t)`
Initialize parameters before factorization.
- `template<typename MatType >`
`void WUpdate (const MatType &V, arma::mat &W, const arma::mat &H)`
The update rule for the basis matrix W.

39.32.1 Detailed Description

This class computes SVD using incomplete incremental batch learning, as described in the following paper:

```
@techreport{ma2008guide,
  title={A Guide to Singular Value Decomposition for Collaborative
    Filtering},
  author={Ma, Chih-Chao},
  year={2008},
  institution={Department of Computer Science, National Taiwan University}
}
```

This class implements 'Algorithm 2' as given in the paper. Incremental learning modifies only some feature values in W and H after scanning part of the input matrix (V). This differs from batch learning, which considers every element in V for each update of W and H . The regularization technique is also different: in incomplete incremental learning, regularization takes into account the number of elements in a given column of V .

See also

SVDBatchLearning (p. 539)

Definition at line 43 of file `svd_incomplete_incremental_learning.hpp`.

39.32.2 Constructor & Destructor Documentation

39.32.2.1 SVDIncompleteIncrementalLearning()

```
SVDIncompleteIncrementalLearning (
    double u = 0.001,
    double kw = 0,
    double kh = 0 ) [inline]
```

Initialize the parameters of **SVDIncompleteIncrementalLearning** (p. 547).

Parameters

<i>u</i>	Step value used in batch learning.
<i>kw</i>	Regularization constant for W matrix.
<i>kh</i>	Regularization constant for H matrix.

Definition at line 53 of file svd_incomplete_incremental_learning.hpp.

39.32.3 Member Function Documentation

39.32.3.1 HUpdate()

```
void HUpdate (
    const MatType & V,
    const arma::mat & W,
    arma::mat & H ) [inline]
```

The update rule for the encoding matrix H.

The function takes in all the matrices and only changes the value of the H matrix.

Parameters

<i>V</i>	Input matrix to be factorized.
<i>W</i>	Basis matrix.
<i>H</i>	Encoding matrix to be updated.

Definition at line 121 of file svd_incomplete_incremental_learning.hpp.

39.32.3.2 Initialize()

```
void Initialize (
    const MatType & ,
    const size_t ) [inline]
```

Initialize parameters before factorization.

This function must be called before a new factorization. This simply sets the column being considered to 0, so the input matrix and rank are not used.

Parameters

<i>dataset</i>	Input matrix to be factorized.
<i>rank</i>	rank of factorization

Definition at line 70 of file svd_incomplete_incremental_learning.hpp.

39.32.3.3 WUpdate()

```
void WUpdate (
    const MatType & V,
    arma::mat & W,
    const arma::mat & H ) [inline]
```

The update rule for the basis matrix W.

The function takes in all the matrices and only changes the value of the W matrix.

Parameters

<i>V</i>	Input matrix to be factorized.
<i>W</i>	Basis matrix to be updated.
<i>H</i>	Encoding matrix.

Definition at line 86 of file svd_incomplete_incremental_learning.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/ **svd_incomplete_incremental_**↵
_learning.hpp

39.33 ValidationRMSETermination< MatType > Class Template Reference

This class implements validation termination policy based on RMSE index.

Public Member Functions

- **ValidationRMSETermination** (MatType &V, size_t num_test_points, double tolerance=1e-5, size_t max←Iterations=10000, size_t reverseStepTolerance=3)
Create a validation set according to given parameters and nullifies this set in data matrix(training set).
- const double & **Index** () const
Get current value of residue.
- void **Initialize** (const MatType &)
Initializes the termination policy before stating the factorization.
- bool **IsConverged** (arma::mat &W, arma::mat &H)
Check if termination criterio is met.
- const size_t & **Iteration** () const
Get current iteration count.
- const size_t & **MaxIterations** () const
Access upper limit of iteration count.
- size_t & **MaxIterations** ()
- const size_t & **NumTestPoints** () const
Get number of validation points.
- const double & **Tolerance** () const
Access tolerance value.
- double & **Tolerance** ()

39.33.1 Detailed Description

```
template<class MatType>
class mlpack::amf::ValidationRMSETermination< MatType >
```

This class implements validation termination policy based on RMSE index.

The input data matrix is divided into 2 sets, training set and validation set. Entries of validation set are nullified in the input matrix. Termination criterion is met when increase in validation set RMSe value drops below the given tolerance. To accommodate spikes certain number of successive validation RMSE drops are accepted. This upper imit on successive drops can be adjusted with reverseStepCount. Secondary termination criterion terminates algorithm when iteration count goes above the threshold.

Note

The input matrix is modified by this termination policy.

See also

AMF (p. 499)

Definition at line 37 of file validation_rmse_termination.hpp.

39.33.2 Constructor & Destructor Documentation

39.33.2.1 ValidationRMSETermination()

```
ValidationRMSETermination (
    MatType & V,
    size_t num_test_points,
    double tolerance = 1e-5,
    size_t maxIterations = 10000,
    size_t reverseStepTolerance = 3 ) [inline]
```

Create a validation set according to given parameters and nullifies this set in data matrix(training set).

Parameters

<i>V</i>	Input matrix to be factorized.
<i>num_test_points</i>	number of validation test points
<i>maxIterations</i>	max iteration count before termination
<i>reverseStepTolerance</i>	max successive RMSE drops allowed

Definition at line 49 of file validation_rmse_termination.hpp.

39.33.3 Member Function Documentation

39.33.3.1 Index()

```
const double& Index ( ) const [inline]
```

Get current value of residue.

Definition at line 187 of file validation_rmse_termination.hpp.

39.33.3.2 Initialize()

```
void Initialize (
    const MatType & ) [inline]
```

Initializes the termination policy before stating the factorization.

Parameters

V	Input matrix to be factorized.
-----	--------------------------------

Definition at line 94 of file validation_rmse_termination.hpp.

39.33.3.3 IsConverged()

```
bool IsConverged (
    arma::mat &  $W$ ,
    arma::mat &  $H$  ) [inline]
```

Check if termination criterio is met.

Parameters

W	Basis matrix of output.
H	Encoding matrix of output.

Definition at line 114 of file validation_rmse_termination.hpp.

39.33.3.4 Iteration()

```
const size_t& Iteration ( ) const [inline]
```

Get current iteration count.

Definition at line 190 of file validation_rmse_termination.hpp.

39.33.3.5 MaxIterations() [1/2]

```
const size_t& MaxIterations ( ) const [inline]
```

Access upper limit of iteration count.

Definition at line 196 of file validation_rmse_termination.hpp.

39.33.3.6 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Definition at line 197 of file validation_rmse_termination.hpp.

39.33.3.7 NumTestPoints()

```
const size_t& NumTestPoints ( ) const [inline]
```

Get number of validation points.

Definition at line 193 of file validation_rmse_termination.hpp.

39.33.3.8 Tolerance() [1/2]

```
const double& Tolerance ( ) const [inline]
```

Access tolerance value.

Definition at line 200 of file validation_rmse_termination.hpp.

39.33.3.9 Tolerance() [2/2]

```
double& Tolerance ( ) [inline]
```

Definition at line 201 of file validation_rmse_termination.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/
termination.hpp **validation_rmse_↵**

39.34 Add< InputDataType, OutputDataType > Class Template Reference

Implementation of the **Add** (p. 553) module class.

Public Member Functions

- **Add** (const size_t outSize=0)
Create the **Add** (p. 553) object using the specified number of output units.
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, const arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename eT >
void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- template<typename eT >
void **Gradient** (const arma::Mat< eT > &&, arma::Mat< eT > &&error, arma::Mat< eT > &&gradient)
Calculate the gradient using the output delta and the input activation.
- OutputDataType const & **Gradient** () const
Get the gradient.
- OutputDataType & **Gradient** ()
Modify the gradient.
- OutputDataType const & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- OutputDataType const & **Parameters** () const
Get the parameters.
- OutputDataType & **Parameters** ()
Modify the parameters.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.34.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::Add< InputDataType, OutputDataType >
```

Implementation of the **Add** (p. 553) module class.

The **Add** (p. 553) module applies a bias term to the incoming data.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 34 of file add.hpp.

39.34.2 Constructor & Destructor Documentation

39.34.2.1 Add()

```
Add (
    const size_t outSize = 0 )
```

Create the **Add** (p. 553) object using the specified number of output units.

Parameters

<i>outSize</i>	The number of output units.
----------------	-----------------------------

39.34.3 Member Function Documentation

39.34.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
    const arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.34.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 91 of file add.hpp.

39.34.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 93 of file add.hpp.

39.34.3.4 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.34.3.5 Gradient() [1/3]

```
void Gradient (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && error,
    arma::Mat< eT > && gradient )
```

Calculate the gradient using the output delta and the input activation.

Parameters

<i>input</i>	The propagated input.
<i>error</i>	The calculated error.
<i>gradient</i>	The calculated gradient.

39.34.3.6 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 96 of file add.hpp.

39.34.3.7 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 98 of file add.hpp.

References Add< InputDataType, OutputDataType >::serialize().

39.34.3.8 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 86 of file add.hpp.

39.34.3.9 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 88 of file add.hpp.

39.34.3.10 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 81 of file add.hpp.

39.34.3.11 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 83 of file add.hpp.

39.34.3.12 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by `Add< InputDataType, OutputDataType >::Gradient()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ add.hpp`

39.35 AddMerge< InputDataType, OutputDataType, CustomLayers > Class Template Reference

Implementation of the **AddMerge** (p. 558) module class.

Public Member Functions

- **AddMerge** (const bool model=false, const bool run=true)
*Create the **AddMerge** (p. 558) object using the specified parameters.*
- **~AddMerge** ()
Destructor to release allocated memory.
- template<class LayerType , class... Args>
void **Add** (Args... args)
- void **Add** (**LayerTypes**< CustomLayers... > layer)
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards trough f .
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g, const size_t index)
*This is the overload of **Backward()** (p. 561) that runs only a specific layer with the given input.*
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename InputType , typename OutputType >
void **Forward** (InputType &&, OutputType &&output)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- template<typename eT >
void **Gradient** (arma::Mat< eT > &&input, arma::Mat< eT > &&error, arma::Mat< eT > &&gradient)
- template<typename eT >
void **Gradient** (arma::Mat< eT > &&input, arma::Mat< eT > &&error, arma::Mat< eT > &&gradient, const size_t index)
- InputDataType const & **InputParameter** () const
Get the input parameter.
- InputDataType & **InputParameter** ()
Modify the input parameter.
- std::vector< **LayerTypes**< CustomLayers... > > & **Model** ()
Return the model modules.
- OutputDataType const & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- OutputDataType const & **Parameters** () const
Get the parameters.
- OutputDataType & **Parameters** ()
Modify the parameters.
- bool **Run** () const
Get the value of run parameter.
- bool & **Run** ()
Modify the value of run parameter.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.35.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat, typename... CustomLayers>
class mlpack::ann::AddMerge< InputDataType, OutputDataType, CustomLayers >
```

Implementation of the **AddMerge** (p. 558) module class.

The **AddMerge** (p. 558) class accumulates the output of various modules.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>CustomLayers</i>	Additional custom layers that can be added.

Definition at line 42 of file add_merge.hpp.

39.35.2 Constructor & Destructor Documentation

39.35.2.1 AddMerge()

```
AddMerge (
    const bool model = false,
    const bool run = true )
```

Create the **AddMerge** (p. 558) object using the specified parameters.

Parameters

<i>model</i>	Expose all the network modules.
<i>run</i>	Call the Forward/Backward method before the output is merged.

39.35.2.2 ~AddMerge()

```
~ AddMerge ( )
```

Destructor to release allocated memory.

39.35.3 Member Function Documentation

39.35.3.1 Add() [1/2]

```
void Add (
    Args... args ) [inline]
```

Definition at line 128 of file add_merge.hpp.

39.35.3.2 Add() [2/2]

```
void Add (
    LayerTypes< CustomLayers... > layer ) [inline]
```

Definition at line 135 of file add_merge.hpp.

39.35.3.3 Backward() [1/2]

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.35.3.4 Backward() [2/2]

```
void Backward (
    const arma::Mat< eT > && ,
```

```

arma::Mat< eT > && gy,
arma::Mat< eT > && g,
const size_t index )

```

This is the overload of **Backward()** (p. 561) that runs only a specific layer with the given input.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.
<i>The</i>	index of the layer to run.

39.35.3.5 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 148 of file add_merge.hpp.

39.35.3.6 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 150 of file add_merge.hpp.

39.35.3.7 Forward()

```

void Forward (
    InputType && ,
    OutputType && output )

```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.35.3.8 Gradient() [1/2]

```
void Gradient (
    arma::Mat< eT > && input,
    arma::Mat< eT > && error,
    arma::Mat< eT > && gradient )
```

39.35.3.9 Gradient() [2/2]

```
void Gradient (
    arma::Mat< eT > && input,
    arma::Mat< eT > && error,
    arma::Mat< eT > && gradient,
    const size_t index )
```

39.35.3.10 InputParameter() [1/2]

```
InputDataType const& InputParameter ( ) const [inline]
```

Get the input parameter.

Definition at line 138 of file add_merge.hpp.

39.35.3.11 InputParameter() [2/2]

```
InputDataType& InputParameter ( ) [inline]
```

Modify the input parameter.

Definition at line 140 of file add_merge.hpp.

39.35.3.12 Model()

```
std::vector< LayerTypes<CustomLayers...> >& Model ( ) [inline]
```

Return the model modules.

Definition at line 153 of file add_merge.hpp.

39.35.3.13 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 143 of file add_merge.hpp.

39.35.3.14 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 145 of file add_merge.hpp.

39.35.3.15 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 164 of file add_merge.hpp.

39.35.3.16 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 166 of file add_merge.hpp.

39.35.3.17 Run() [1/2]

```
bool Run ( ) const [inline]
```

Get the value of run parameter.

Definition at line 169 of file add_merge.hpp.

39.35.3.18 Run() [2/2]

```
bool& Run ( ) [inline]
```

Modify the value of run parameter.

Definition at line 171 of file add_merge.hpp.

References AddMerge< InputDataType, OutputDataType, CustomLayers >::serialize().

39.35.3.19 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by AddMerge< InputDataType, OutputDataType, CustomLayers >::Run().

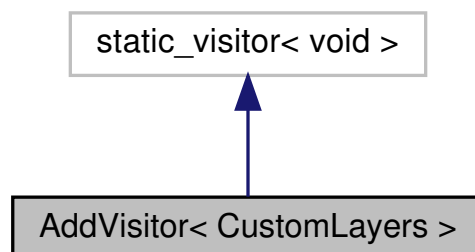
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **add_merge.hpp**

39.36 AddVisitor< CustomLayers > Class Template Reference

AddVisitor (p. 565) exposes the Add() method of the given module.

Inheritance diagram for AddVisitor< CustomLayers >:



Public Member Functions

- `template<typename T >`
AddVisitor (T newLayer)
Exposes the Add() method of the given module.
- `template<typename LayerType >`
void operator() (LayerType *layer) const
Exposes the Add() method.

39.36.1 Detailed Description

```
template<typename... CustomLayers>
class mlpack::ann::AddVisitor< CustomLayers >
```

AddVisitor (p. 565) exposes the Add() method of the given module.

Definition at line 28 of file add_visitor.hpp.

39.36.2 Constructor & Destructor Documentation

39.36.2.1 AddVisitor()

```
AddVisitor (
    T newLayer )
```

Exposes the Add() method of the given module.

39.36.3 Member Function Documentation

39.36.3.1 operator>()

```
void operator() (
    LayerType * layer ) const
```

Exposes the Add() method.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **add_visitor.hpp**

39.37 AlphaDropout< InputDataType, OutputDataType > Class Template Reference

The alpha - dropout layer is a regularizer that randomly with probability 'ratio' sets input values to alphaDash.

Public Member Functions

- **AlphaDropout** (const double ratio=0.5, const double alphaDash=-alpha *lambda)
Create the Alpha_Dropout object using the specified ratio.
- double **A** () const
Value to be multiplied with x for affine transformation.
- double **AlphaDash** () const
Value of alphaDash.
- double **B** () const
*Value to be added to a*x for affine transformation.*
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of the alpha_dropout layer.
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- bool **Deterministic** () const
The value of the deterministic parameter.
- bool & **Deterministic** ()
Modify the value of the deterministic parameter.
- template<typename eT >
void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Ordinary feed forward pass of the alpha_dropout layer.
- OutputDataType const & **Mask** () const
Get the mask.
- OutputDataType const & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- double **Ratio** () const
The probability of setting a value to alphaDash.
- void **Ratio** (const double r)
Modify the probability of setting a value to alphaDash.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.37.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::AlphaDropout< InputDataType, OutputDataType >
```

The alpha - dropout layer is a regularizer that randomly with probability 'ratio' sets input values to alphaDash.

The alpha - dropout layer is mostly used for SELU activation function where successive layers don't have same mean and variance.

For more information, see the following.

```
@article{Klambauer2017,
  author = {Gunter Klambauer and Thomas Unterthiner and
            Andreas Mayr},
  title = {Self-Normalizing Neural Networks},
  journal = {Advances in Neural Information Processing Systems},
  year = {2017}
}
```

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 50 of file alpha_dropout.hpp.

39.37.2 Constructor & Destructor Documentation

39.37.2.1 AlphaDropout()

```
AlphaDropout (
    const double ratio = 0.5,
    const double alphaDash = -alpha * lambda )
```

Create the Alpha_Dropout object using the specified ratio.

Parameters

<i>ratio</i>	The probability of setting a value to alphaDash.
<i>alphaDash</i>	The dropout scaling parameter.

39.37.3 Member Function Documentation

39.37.3.1 A()

```
double A ( ) const [inline]
```

Value to be multiplied with x for affine transformation.

Definition at line 102 of file alpha_dropout.hpp.

39.37.3.2 AlphaDash()

```
double AlphaDash ( ) const [inline]
```

Value of alphaDash.

Definition at line 108 of file alpha_dropout.hpp.

39.37.3.3 B()

```
double B ( ) const [inline]
```

Value to be added to a*x for affine transformation.

Definition at line 105 of file alpha_dropout.hpp.

39.37.3.4 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of the alpha_dropout layer.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.37.3.5 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 89 of file alpha_dropout.hpp.

39.37.3.6 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 91 of file alpha_dropout.hpp.

39.37.3.7 Deterministic() [1/2]

```
bool Deterministic ( ) const [inline]
```

The value of the deterministic parameter.

Definition at line 94 of file alpha_dropout.hpp.

39.37.3.8 Deterministic() [2/2]

```
bool& Deterministic ( ) [inline]
```

Modify the value of the deterministic parameter.

Definition at line 96 of file alpha_dropout.hpp.

39.37.3.9 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of the alpha_dropout layer.

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.37.3.10 Mask()

```
OutputDataType const& Mask ( ) const [inline]
```

Get the mask.

Definition at line 111 of file alpha_dropout.hpp.

39.37.3.11 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 84 of file alpha_dropout.hpp.

39.37.3.12 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 86 of file alpha_dropout.hpp.

39.37.3.13 Ratio() [1/2]

```
double Ratio ( ) const [inline]
```

The probability of setting a value to alphaDash.

Definition at line 99 of file alpha_dropout.hpp.

39.37.3.14 Ratio() [2/2]

```
void Ratio (
    const double r ) [inline]
```

Modify the probability of setting a value to alphaDash.

As 'a' and 'b' depend on 'ratio', modify them as well.

Definition at line 115 of file alpha_dropout.hpp.

References AlphaDropout< InputDataType, OutputDataType >::serialize().

39.37.3.15 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by AlphaDropout< InputDataType, OutputDataType >::Ratio().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **alpha_dropout.hpp**

39.38 AtrousConvolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, InputDataType, OutputDataType > Class Template Reference

Implementation of the Atrous **Convolution** (p. 643) class.

Public Member Functions

- **AtrousConvolution** ()

*Create the **AtrousConvolution** (p. 572) object.*

- **AtrousConvolution** (const size_t inSize, const size_t outSize, const size_t kW, const size_t kH, const size_t dW=1, const size_t dH=1, const size_t padW=0, const size_t padH=0, const size_t inputWidth=0, const size_t inputHeight=0, const size_t dilationW=1, const size_t dilationH=1)

*Create the **AtrousConvolution** (p. 572) object using the specified number of input maps, output maps, filter size, stride, dilation and padding parameter.*

- template<typename eT >

void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

- OutputDataType const & **Delta** () const

Get the delta.

- OutputDataType & **Delta** ()

Modify the delta.

- template<typename eT >

void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

- template<typename eT >

void **Gradient** (const arma::Mat< eT > &&, arma::Mat< eT > &&error, arma::Mat< eT > &&gradient)

- OutputDataType const & **Gradient** () const

Get the gradient.

- OutputDataType & **Gradient** ()

Modify the gradient.

- size_t const & **InputHeight** () const

Get the input height.

- size_t & **InputHeight** ()

Modify the input height.

- size_t const & **InputWidth** () const

Get the input width.

- size_t & **InputWidth** ()

Modify input the width.

- size_t const & **OutputHeight** () const

Get the output height.

- size_t & **OutputHeight** ()

Modify the output height.

- OutputDataType const & **OutputParameter** () const

Get the output parameter.

- OutputDataType & **OutputParameter** ()

Modify the output parameter.

- size_t const & **OutputWidth** () const

Get the output width.

- size_t & **OutputWidth** ()

Modify the output width.

- OutputDataType const & **Parameters** () const

Get the parameters.

- OutputDataType & **Parameters** ()

Modify the parameters.

- void **Reset** ()
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)

Serialize the layer.

39.38.1 Detailed Description

```
template<typename ForwardConvolutionRule = NaiveConvolution<ValidConvolution>, typename BackwardConvolutionRule =
NaiveConvolution<FullConvolution>, typename GradientConvolutionRule = NaiveConvolution<ValidConvolution>, typename
InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::AtrousConvolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, InputData↵
Type, OutputDataType >
```

Implementation of the Atrous **Convolution** (p. 643) class.

The Atrous **Convolution** (p. 643) class represents a single layer of a neural network. Atrous (or Dilated) Convolutions are just simple convolutions applied to input with the defined, spaces included between the kernel cells, in order to capture a larger field of reception, without having to increase discrete kernel sizes.

Template Parameters

<i>ForwardConvolutionRule</i>	Atrous Convolution (p. 643) to perform forward process.
<i>BackwardConvolutionRule</i>	Atrous Convolution (p. 643) to perform backward process.
<i>GradientConvolutionRule</i>	Atrous Convolution (p. 643) to calculate gradient.
<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 50 of file atrous_convolution.hpp.

39.38.2 Constructor & Destructor Documentation

39.38.2.1 AtrousConvolution() [1/2]

```
AtrousConvolution ( )
```

Create the **AtrousConvolution** (p. 572) object.

39.38.2.2 AtrousConvolution() [2/2]

```

AtrousConvolution (
    const size_t inSize,
    const size_t outSize,
    const size_t kW,
    const size_t kH,
    const size_t dW = 1,
    const size_t dH = 1,
    const size_t padW = 0,
    const size_t padH = 0,
    const size_t inputWidth = 0,
    const size_t inputHeight = 0,
    const size_t dilationW = 1,
    const size_t dilationH = 1 )

```

Create the **AtrousConvolution** (p. 572) object using the specified number of input maps, output maps, filter size, stride, dilation and padding parameter.

Parameters

<i>inSize</i>	The number of input maps.
<i>outSize</i>	The number of output maps.
<i>kW</i>	Width of the filter/kernel.
<i>kH</i>	Height of the filter/kernel.
<i>dW</i>	Stride of filter application in the x direction.
<i>dH</i>	Stride of filter application in the y direction.
<i>padW</i>	Padding width of the input.
<i>padH</i>	Padding height of the input.
<i>inputWidth</i>	The widht of the input data.
<i>inputHeight</i>	The height of the input data.
<i>dilationW</i>	The space between the cells of filters in x direction.
<i>dilationH</i>	The space between the cells of filters in y direction.

39.38.3 Member Function Documentation

39.38.3.1 Backward()

```

void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )

```

Ordinary feed backward pass of a neural network, calculating the function f(x) by propagating x backwards through f. Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.38.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 139 of file `atrous_convolution.hpp`.

39.38.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 141 of file `atrous_convolution.hpp`.

39.38.3.4 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.38.3.5 Gradient() [1/3]

```
void Gradient (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && error,
    arma::Mat< eT > && gradient )
```

39.38.3.6 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 144 of file atrous_convolution.hpp.

39.38.3.7 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 146 of file atrous_convolution.hpp.

39.38.3.8 InputHeight() [1/2]

```
size_t const& InputHeight ( ) const [inline]
```

Get the input height.

Definition at line 154 of file atrous_convolution.hpp.

39.38.3.9 InputHeight() [2/2]

```
size_t& InputHeight ( ) [inline]
```

Modify the input height.

Definition at line 156 of file atrous_convolution.hpp.

39.38.3.10 InputWidth() [1/2]

```
size_t const& InputWidth ( ) const [inline]
```

Get the input width.

Definition at line 149 of file `atrous_convolution.hpp`.

39.38.3.11 InputWidth() [2/2]

```
size_t& InputWidth ( ) [inline]
```

Modify input the width.

Definition at line 151 of file `atrous_convolution.hpp`.

39.38.3.12 OutputHeight() [1/2]

```
size_t const& OutputHeight ( ) const [inline]
```

Get the output height.

Definition at line 164 of file `atrous_convolution.hpp`.

39.38.3.13 OutputHeight() [2/2]

```
size_t& OutputHeight ( ) [inline]
```

Modify the output height.

Definition at line 166 of file `atrous_convolution.hpp`.

References `AtrousConvolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, InputDataType, OutputDataType >::serialize()`.

39.38.3.14 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 134 of file atrous_convolution.hpp.

39.38.3.15 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 136 of file atrous_convolution.hpp.

39.38.3.16 OutputWidth() [1/2]

```
size_t const& OutputWidth ( ) const [inline]
```

Get the output width.

Definition at line 159 of file atrous_convolution.hpp.

39.38.3.17 OutputWidth() [2/2]

```
size_t& OutputWidth ( ) [inline]
```

Modify the output width.

Definition at line 161 of file atrous_convolution.hpp.

39.38.3.18 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 129 of file atrous_convolution.hpp.

39.38.3.19 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 131 of file `atrous_convolution.hpp`.

39.38.3.20 Reset()

```
void Reset ( )
```

39.38.3.21 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by `AtrousConvolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, InputDataType, OutputDataType >::OutputHeight()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ atrous_convolution.hpp`

39.39 AddTask Class Reference

Generator of instances of the binary addition task.

Public Member Functions

- **AddTask** (const size_t bitLen)
Creates an instance of the binary addition task.
- void **Generate** (arma::field< arma::mat > &input, arma::field< arma::mat > &labels, const size_t batchSize, const bool fixedLength=false) const
Generate dataset of a given size.
- void **Generate** (arma::mat &input, arma::mat &labels, const size_t batchSize) const
Generate dataset of a given size and store it in arma::mat object.

39.39.1 Detailed Description

Generator of instances of the binary addition task.

The parameters are:

- maximum binary length;

Every element of sequence is encoded as 1-dimensional vector (possible vector elements are {0, 1, 0.5} - the latter corresponds to '+' sign'). Generated datasets are compliant with mlpack format - every dataset element is shaped as a vector of length 3 * (sequence length),

Example of generated dataset (binary length = 2):

- Input sequence: [0,1,0,0,0,1,0,1,0,1,0,0]
- Output sequences: [0,1,0,0,1,0]

Definition at line 42 of file add.hpp.

39.39.2 Constructor & Destructor Documentation

39.39.2.1 AddTask()

```
AddTask (
    const size_t bitLen )
```

Creates an instance of the binary addition task.

Parameters

<i>bitLen</i>	Maximum binary length of added numbers.
---------------	---

39.39.3 Member Function Documentation

39.39.3.1 Generate() [1/2]

```
void Generate (
    arma::field< arma::mat > & input,
```

```
arma::field< arma::mat > & labels,
const size_t batchSize,
const bool fixedLength = false ) const
```

Generate dataset of a given size.

Parameters

<i>input</i>	The variable to store input sequences.
<i>labels</i>	The variable to store output sequences.
<i>batchSize</i>	The dataset size.
<i>fixedLength</i>	Flag that indicates whether the method should return sequences of even length.

39.39.3.2 Generate() [2/2]

```
void Generate (
    arma::mat & input,
    arma::mat & labels,
    const size_t batchSize ) const
```

Generate dataset of a given size and store it in arma::mat object.

Parameters

<i>input</i>	The variable to store input sequences.
<i>labels</i>	The variable to store output sequences.
<i>batchSize</i>	The dataset size.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented/tasks/ **add.hpp**

39.40 CopyTask Class Reference

Generator of instances of the binary sequence copy task.

Public Member Functions

- **CopyTask** (const size_t maxLength, const size_t nRepeats, const bool addSeparator=false)
Creates an instance of the sequence copy task.
- void **Generate** (arma::field< arma::mat > &input, arma::field< arma::mat > &labels, const size_t batchSize, bool fixedLength=false) const
Generate dataset of a given size.
- void **Generate** (arma::mat &input, arma::mat &labels, const size_t batchSize) const
Generate dataset of a given size and store it in arma::mat object.

39.40.1 Detailed Description

Generator of instances of the binary sequence copy task.

The parameters are:

- maximum sequence length;
- number of sequence repetitions.

Input/output sequences are aligned to have the same length: input sequence is padded with zeros from the right end, output sequence is padded with zeros from the left end. The sequences are formed of 2-dimensional vectors of the format [sequence element, input flag], where input flag = 0 iff first vector element is a sequence element.

Generated datasets are compliant with mpack format - every dataset element is shaped as a vector of length (elem-length) * (input sequence length + target sequence length), where elem-length is 2 for input sequences and 1 for output sequences.

Example of generated dataset (sequence length = 3, repetition count = 2):

- Input sequence: [1,0,0,0,1,0,0,1,0,1,0,1,0,1,0,1]
- Output sequences: [0,0,0,1,0,1,1,0,1]

Definition at line 48 of file copy.hpp.

39.40.2 Constructor & Destructor Documentation

39.40.2.1 CopyTask()

```
CopyTask (
    const size_t maxLength,
    const size_t nRepeats,
    const bool addSeparator = false )
```

Creates an instance of the sequence copy task.

Parameters

<i>maxLength</i>	Maximum length of sequence that has to be repeated by model.
<i>nRepeats</i>	Number of repeats required to solve the task.
<i>addSeparator</i>	Flag indicating whether generator should emit separating symbol after input sequence.

39.40.3 Member Function Documentation

39.40.3.1 Generate() [1/2]

```
void Generate (
    arma::field< arma::mat > & input,
    arma::field< arma::mat > & labels,
    const size_t batchSize,
    bool fixedLength = false ) const
```

Generate dataset of a given size.

Parameters

<i>input</i>	The variable to store input sequences.
<i>labels</i>	The variable to store output sequences.
<i>batchSize</i>	The dataset size.

39.40.3.2 Generate() [2/2]

```
void Generate (
    arma::mat & input,
    arma::mat & labels,
    const size_t batchSize ) const
```

Generate dataset of a given size and store it in arma::mat object.

Parameters

<i>input</i>	The variable to store input sequences.
<i>labels</i>	The variable to store output sequences.
<i>batchSize</i>	The dataset size.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented/tasks/ **copy.hpp**

39.41 SortTask Class Reference

Generator of instances of the sequence sort task.

Public Member Functions

- **SortTask** (const size_t maxLength, const size_t bitLen, bool addSeparator=false)
Creates an instance of the sequence sort task.
- void **Generate** (arma::field< arma::mat > &input, arma::field< arma::mat > &labels, const size_t batchSize, bool fixedLength=false) const
Generate dataset of a given size.
- void **Generate** (arma::mat &input, arma::mat &labels, const size_t batchSize) const
Generate dataset of a given size and store it in arma::mat object.

39.41.1 Detailed Description

Generator of instances of the sequence sort task.

The parameters are:

- maximum sequence length;
- binary length of sequence elements.

Generated datasets are compliant with mlpack format - every dataset element is shaped as a vector of length (binary length) * (sequence length).

Example of generated dataset (sequence length = 3, binary length = 2):

- Input sequences: [1,1,0,0,0,1] (three numbers in the sequence are 11, 00, and 01)
- Output sequences: [0,0,0,1,1,1] (00, 01, 11 - reordering of the numbers above in the ascending order)

Definition at line 41 of file sort.hpp.

39.41.2 Constructor & Destructor Documentation

39.41.2.1 SortTask()

```
SortTask (
    const size_t maxLength,
    const size_t bitLen,
    bool addSeparator = false )
```

Creates an instance of the sequence sort task.

Parameters

<i>maxLength</i>	Maximum length of the number sequence.
<i>bitLen</i>	Binary length of sorted numbers.
<i>addSeparator</i>	Flag indicating whether generator should emit separating symbol after input sequence.

39.41.3 Member Function Documentation

39.41.3.1 **Generate()** [1/2]

```
void Generate (
    arma::field< arma::mat > & input,
    arma::field< arma::mat > & labels,
    const size_t batchSize,
    bool fixedLength = false ) const
```

Generate dataset of a given size.

Parameters

<i>input</i>	The variable to store input sequences.
<i>labels</i>	The variable to store output sequences.
<i>batchSize</i>	The dataset size.
<i>fixedLength</i>	Flag indicating whether generator should emit sequences of pairwise equal length.

39.41.3.2 **Generate()** [2/2]

```
void Generate (
    arma::mat & input,
    arma::mat & labels,
    const size_t batchSize ) const
```

Generate dataset of a given size and store it in arma::mat object.

Parameters

<i>input</i>	The variable to store input sequences.
<i>labels</i>	The variable to store output sequences.
<i>batchSize</i>	The dataset size.

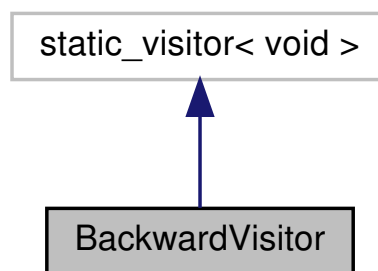
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented/tasks/ **sort.hpp**

39.42 BackwardVisitor Class Reference

BackwardVisitor (p. 587) executes the Backward() function given the input, error and delta parameter.

Inheritance diagram for BackwardVisitor:



Public Member Functions

- **BackwardVisitor** (arma::mat &&input, arma::mat &&error, arma::mat &&delta)
Execute the Backward() function given the input, error and delta parameter.
- **BackwardVisitor** (arma::mat &&input, arma::mat &&error, arma::mat &&delta, const size_t index)
Execute the Backward() function for the layer with the specified index.
- template<typename LayerType >
void **operator()** (LayerType *layer) const
Execute the Backward() function.

39.42.1 Detailed Description

BackwardVisitor (p. 587) executes the Backward() function given the input, error and delta parameter.

Definition at line 28 of file backward_visitor.hpp.

39.42.2 Constructor & Destructor Documentation

39.42.2.1 BackwardVisitor() [1/2]

```
BackwardVisitor (
    arma::mat && input,
    arma::mat && error,
    arma::mat && delta )
```

Execute the Backward() function given the input, error and delta parameter.

39.42.2.2 BackwardVisitor() [2/2]

```
BackwardVisitor (
    arma::mat && input,
    arma::mat && error,
    arma::mat && delta,
    const size_t index )
```

Execute the Backward() function for the layer with the specified index.

39.42.3 Member Function Documentation

39.42.3.1 operator>()

```
void operator() (
    LayerType * layer ) const
```

Execute the Backward() function.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **backward_visitor.hpp**

39.43 BaseLayer< ActivationFunction, InputDataType, OutputDataType > Class Template Reference

Implementation of the base layer.

Public Member Functions

- **BaseLayer** ()
Create the **BaseLayer** (p. 588) object.
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename InputType , typename OutputType >
void **Forward** (const InputType &&input, OutputType &&output)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- OutputDataType const & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- template<typename Archive >
void **serialize** (Archive &, const unsigned int)
Serialize the layer.

39.43.1 Detailed Description

```
template<class ActivationFunction = LogisticFunction, typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::BaseLayer< ActivationFunction, InputDataType, OutputDataType >
```

Implementation of the base layer.

The base layer works as a metaclass which attaches various functions to the embedding layer.

A few convenience typedefs are given:

- SigmoidLayer
- IdentityLayer
- ReLULayer
- TanHLayer

Template Parameters

<i>ActivationFunction</i>	Activation function used for the embedding layer.
<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 49 of file base_layer.hpp.

39.43.2 Constructor & Destructor Documentation

39.43.2.1 BaseLayer()

```
BaseLayer ( ) [inline]
```

Create the **BaseLayer** (p. 588) object.

Definition at line 55 of file base_layer.hpp.

39.43.3 Member Function Documentation

39.43.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g ) [inline]
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

Definition at line 83 of file base_layer.hpp.

39.43.3.2 Delta() ^[1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 98 of file base_layer.hpp.

39.43.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 100 of file base_layer.hpp.

39.43.3.4 Forward()

```
void Forward (
    const InputType && input,
    OutputType && output ) [inline]
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

Definition at line 68 of file base_layer.hpp.

39.43.3.5 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 93 of file base_layer.hpp.

39.43.3.6 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 95 of file base_layer.hpp.

39.43.3.7 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serialize the layer.

Definition at line 106 of file base_layer.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **base_layer.hpp**

39.44 BatchNorm< InputDataType, OutputDataType > Class Template Reference

Declaration of the Batch Normalization layer class.

Public Member Functions

- **BatchNorm** ()
*Create the **BatchNorm** (p. 592) object.*
- **BatchNorm** (const size_t size, const double eps=1e-8)
*Create the **BatchNorm** (p. 592) layer object for a specified number of input units.*
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Backward pass through the layer.
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- bool **Deterministic** () const
Get the value of deterministic parameter.
- bool & **Deterministic** ()
Modify the value of deterministic parameter.

- `template<typename eT >`
`void Forward (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)`
Forward pass of the Batch Normalization layer.
- `template<typename eT >`
`void Gradient (const arma::Mat< eT > &&input, arma::Mat< eT > &&error, arma::Mat< eT > &&gradient)`
Calculate the gradient using the output delta and the input activations.
- `OutputDataType const & Gradient () const`
Get the gradient.
- `OutputDataType & Gradient ()`
Modify the gradient.
- `OutputDataType const & OutputParameter () const`
Get the output parameter.
- `OutputDataType & OutputParameter ()`
Modify the output parameter.
- `OutputDataType const & Parameters () const`
Get the parameters.
- `OutputDataType & Parameters ()`
Modify the parameters.
- `void Reset ()`
Reset the layer parameters.
- `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`
Serialize the layer.
- `OutputDataType TrainingMean ()`
Get the mean over the training data.
- `OutputDataType TrainingVariance ()`
Get the variance over the training data.

39.44.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::BatchNorm< InputDataType, OutputDataType >
```

Declaration of the Batch Normalization layer class.

The layer transforms the input data into zero mean and unit variance and then scales and shifts the data by parameters, gamma and beta respectively. These parameters are learnt by the network.

If deterministic is false (training), the mean and variance over the batch is calculated and the data is normalized. If it is set to true (testing) then the mean and variance accrued over the training set is used.

For more information, refer to the following paper,

```
@article{Ioffe15,
  author    = {Sergey Ioffe and
              Christian Szegedy},
  title     = {Batch Normalization: Accelerating Deep Network Training by
              Reducing Internal Covariate Shift},
  journal   = {CoRR},
  volume    = {abs/1502.03167},
  year      = {2015},
  url       = {http://arxiv.org/abs/1502.03167},
  eprint    = {1502.03167},
}
```

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 56 of file batch_norm.hpp.

39.44.2 Constructor & Destructor Documentation

39.44.2.1 BatchNorm() [1/2]

```
BatchNorm ( )
```

Create the **BatchNorm** (p. 592) object.

39.44.2.2 BatchNorm() [2/2]

```
BatchNorm (
    const size_t size,
    const double eps = 1e-8 )
```

Create the **BatchNorm** (p. 592) layer object for a specified number of input units.

Parameters

<i>size</i>	The number of input units.
<i>eps</i>	The epsilon added to variance to ensure numerical stability.

39.44.3 Member Function Documentation

39.44.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && input,
```

```
arma::Mat< eT > && gy,  
arma::Mat< eT > && g )
```

Backward pass through the layer.

Parameters

<i>input</i>	The input activations
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.44.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 121 of file batch_norm.hpp.

39.44.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 123 of file batch_norm.hpp.

39.44.3.4 Deterministic() [1/2]

```
bool Deterministic ( ) const [inline]
```

Get the value of deterministic parameter.

Definition at line 131 of file batch_norm.hpp.

39.44.3.5 Deterministic() [2/2]

```
bool& Deterministic ( ) [inline]
```

Modify the value of deterministic parameter.

Definition at line 133 of file batch_norm.hpp.

39.44.3.6 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Forward pass of the Batch Normalization layer.

Transforms the input data into zero mean and unit variance, scales the data by a factor gamma and shifts it by beta.

Parameters

<i>input</i>	Input data for the layer
<i>output</i>	Resulting output activations.

39.44.3.7 Gradient() [1/3]

```
void Gradient (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && error,
    arma::Mat< eT > && gradient )
```

Calculate the gradient using the output delta and the input activations.

Parameters

<i>input</i>	The input activations
<i>error</i>	The calculated error
<i>gradient</i>	The calculated gradient.

39.44.3.8 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 126 of file batch_norm.hpp.

39.44.3.9 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 128 of file batch_norm.hpp.

39.44.3.10 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 116 of file batch_norm.hpp.

39.44.3.11 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 118 of file batch_norm.hpp.

39.44.3.12 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 111 of file batch_norm.hpp.

39.44.3.13 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 113 of file batch_norm.hpp.

39.44.3.14 Reset()

```
void Reset ( )
```

Reset the layer parameters.

39.44.3.15 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by BatchNorm< InputDataType, OutputDataType >::TrainingVariance().

39.44.3.16 TrainingMean()

```
OutputDataType TrainingMean ( ) [inline]
```

Get the mean over the training data.

Definition at line 136 of file batch_norm.hpp.

39.44.3.17 TrainingVariance()

```
OutputDataType TrainingVariance ( ) [inline]
```

Get the variance over the training data.

Definition at line 139 of file batch_norm.hpp.

References BatchNorm< InputDataType, OutputDataType >::serialize().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **batch_norm.hpp**

39.45 BernoulliDistribution< DataType > Class Template Reference

Multiple independent Bernoulli distributions.

Public Member Functions

- **BernoulliDistribution** ()
Default constructor, which creates a Bernoulli distribution with zero dimension.
 - **BernoulliDistribution** (const DataType &¶m, const bool applyLogistic=true, const double eps=1e-10)
Create multiple independent Bernoulli distributions whose p values are given by the param parameter.
 - const DataType & **Logits** () const
Return the logits matrix.
 - DataType & **Logits** ()
Return a modifiable copy of the pre probability matrix.
 - double **LogProbability** (const DataType &&observation) const
Return the log probabilities of the given matrix of observations.
 - void **LogProbBackward** (const DataType &&observation, DataType &&output) const
Stores the gradient of the log probabilities of the observations in the output matrix.
 - double **Probability** (const DataType &&observation) const
Return the probabilities of the given matrix of observations.
 - const DataType & **Probability** () const
Return the probability matrix.
 - DataType & **Probability** ()
Return a modifiable copy of the probability matrix.
 - DataType **Sample** () const
Return a matrix of randomly generated samples according to the probability distributions defined by this object.
 - template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
- Serialize the distribution.*

39.45.1 Detailed Description

```
template<typename DataType = arma::mat>
class mlpack::ann::BernoulliDistribution< DataType >
```

Multiple independent Bernoulli distributions.

Bernoulli distribution is the discrete probability distribution of a random variable which takes the value 1 with probability p and the value 0 with probability $q = 1 - p$. In this implementation, the p values of the distributions are given by the param matrix.

Template Parameters

<i>DataType</i>	Type of the input data. (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
-----------------	--

Definition at line 34 of file bernoulli_distribution.hpp.

39.45.2 Constructor & Destructor Documentation

39.45.2.1 BernoulliDistribution() [1/2]

```
BernoulliDistribution ( )
```

Default constructor, which creates a Bernoulli distribution with zero dimension.

39.45.2.2 BernoulliDistribution() [2/2]

```
BernoulliDistribution (
    const DataType && param,
    const bool applyLogistic = true,
    const double eps = 1e-10 )
```

Create multiple independent Bernoulli distributions whose p values are given by the param parameter.

Thus, we create `nofRows * nofColumns` distributions. The shape of the matrix of distributions is the same as the shape of the param matrix as each element of the param matrix parameterizes one Bernoulli distribution. This is used in the ANN module to define distribution for each feature in each batch, where number of features becomes `nofRows` and batch size becomes `nofColumns`.

`applyLogistic` has to be true if all the elements of param matrix are not in the range [0, 1].

Parameters

<i>param</i>	The matrix of probabilities or pre probabilities of the multiple distributions.
<i>applyLogistic</i>	If true, we apply Logistic function to the param matrix (pre probability) to get probability.
<i>eps</i>	The minimum value used for computing logarithms and denominators.

39.45.3 Member Function Documentation

39.45.3.1 Logits() [1/2]

```
const DataType& Logits ( ) const [inline]
```

Return the logits matrix.

Definition at line 108 of file `bernoulli_distribution.hpp`.

39.45.3.2 Logits() [2/2]

```
DataType& Logits ( ) [inline]
```

Return a modifiable copy of the pre probability matrix.

Definition at line 111 of file bernoulli_distribution.hpp.

39.45.3.3 LogProbability()

```
double LogProbability (
    const DataType && observation ) const
```

Return the log probabilities of the given matrix of observations.

Parameters

<i>observation</i>	The observation matrix.
--------------------	-------------------------

Referenced by BernoulliDistribution< DataType >::Probability().

39.45.3.4 LogProbBackward()

```
void LogProbBackward (
    const DataType && observation,
    DataType && output ) const
```

Stores the gradient of the log probabilities of the observations in the output matrix.

Parameters

<i>observation</i>	The observation matrix.
<i>output</i>	The output matrix where the gradients are stored.

Referenced by BernoulliDistribution< DataType >::Probability().

39.45.3.5 Probability() [1/3]

```
double Probability (
    const DataType && observation ) const [inline]
```

Return the probabilities of the given matrix of observations.

Parameters

<i>observation</i>	The observation matrix.
--------------------	-------------------------

Definition at line 72 of file bernoulli_distribution.hpp.

References `BernoulliDistribution< DataType >::LogProbability()`, `BernoulliDistribution< DataType >::LogProb←Backward()`, and `BernoulliDistribution< DataType >::Sample()`.

39.45.3.6 Probability() [2/3]

```
const DataType& Probability ( ) const [inline]
```

Return the probability matrix.

Definition at line 102 of file bernoulli_distribution.hpp.

39.45.3.7 Probability() [3/3]

```
DataType& Probability ( ) [inline]
```

Return a modifiable copy of the probability matrix.

Definition at line 105 of file bernoulli_distribution.hpp.

39.45.3.8 Sample()

```
DataType Sample ( ) const
```

Return a matrix of randomly generated samples according to the probability distributions defined by this object.

Returns

Matrix(integer) of random samples from the multiple Bernoulli distributions.

Referenced by `BernoulliDistribution< DataType >::Probability()`.

39.45.3.9 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the distribution.

Definition at line 117 of file bernoulli_distribution.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/dists/ **bernoulli_distribution.hpp**

39.46 BilinearInterpolation< InputDataType, OutputDataType > Class Template Reference

Definition and Implementation of the Bilinear Interpolation Layer.

Public Member Functions

- **BilinearInterpolation ()**
Create the Bilinear Interpolation object.
- **BilinearInterpolation** (const size_t inRowSize, const size_t inColSize, const size_t outRowSize, const size_t outColSize, const size_t depth)
The constructor for the Bilinear Interpolation.
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gradient, arma::Mat< eT > &&output)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename eT >
void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Forward pass through the layer.
- OutputDataType const & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.46.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::BilinearInterpolation< InputDataType, OutputDataType >
```

Definition and Implementation of the Bilinear Interpolation Layer.

Bilinear Interpolation is an mathematical technique, primarily used for scaling purposes. It is an extension of linear interpolation, for interpolating functions of two variables on a rectangular grid. The key idea is to perform linear interpolation first in one direction (e.g., along x-axis), and then again in the other direction (i.e., y-axis), on four different known points in the grid. This way, we represent any arbitrary point, present within the grid, as a function of those four points.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 39 of file bilinear_interpolation.hpp.

39.46.2 Constructor & Destructor Documentation

39.46.2.1 BilinearInterpolation() [1/2]

```
BilinearInterpolation ( )
```

Create the Bilinear Interpolation object.

39.46.2.2 BilinearInterpolation() [2/2]

```
BilinearInterpolation (
    const size_t inRowSize,
    const size_t inColSize,
    const size_t outRowSize,
    const size_t outColSize,
    const size_t depth )
```

The constructor for the Bilinear Interpolation.

Parameters

<i>inRowSize</i>	Number of input rows.
<i>inColSize</i>	Number of input columns.
<i>outRowSize</i>	Number of output rows.
<i>outColSize</i>	Number of output columns.
<i>depth</i>	Number of input slices.

39.46.3 Member Function Documentation

39.46.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gradient,
    arma::Mat< eT > && output )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass. Since the layer does not have any learn-able parameters, we just have to down-sample the gradient to make its size compatible with the input size.

Parameters

<i>input</i>	The input matrix.
<i>gradient</i>	The computed backward gradient.
<i>output</i>	The resulting down-sampled output.

39.46.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 92 of file `bilinear_interpolation.hpp`.

39.46.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 94 of file `bilinear_interpolation.hpp`.

References `BilinearInterpolation< InputDataType, OutputDataType >::serialize()`.

39.46.3.4 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Forward pass through the layer.

The layer interpolates the matrix using the given Bilinear Interpolation method.

Parameters

<i>input</i>	The input matrix.
<i>output</i>	The resulting interpolated output matrix.

39.46.3.5 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 87 of file `bilinear_interpolation.hpp`.

39.46.3.6 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 89 of file `bilinear_interpolation.hpp`.

39.46.3.7 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by `BilinearInterpolation< InputDataType, OutputDataType >::Delta()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ bilinear_interpolation.hpp`

39.47 BinaryRBM Class Reference

For more information, see the following paper:

39.47.1 Detailed Description

For more information, see the following paper:

```
@article{Hinton10,
  author    = {Geoffrey Hinton},
  title     = {A Practical Guide to Training Restricted Boltzmann Machines},
  year      = {2010},
  url       = {https://www.cs.toronto.edu/~hinton/absps/guideTR.pdf}
}
```

Definition at line 30 of file `rbm_policies.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/rbm/ rbm_policies.hpp`

39.48 **BRNN**< **OutputLayerType**, **MergeLayerType**, **MergeOutputType**, **InitializationRuleType**, **CustomLayers** > Class Template Reference

Implementation of a standard bidirectional recurrent neural network container.

Public Types

- using **NetworkType** = **BRNN**< **OutputLayerType**, **MergeLayerType**, **MergeOutputType**, **InitializationRuleType**, **CustomLayers**... >
Convenience typedef for the internal model construction.

Public Member Functions

- **BRNN** (const size_t rho, const bool single=false, **OutputLayerType** outputLayer=**OutputLayerType**(), **MergeLayerType** mergeLayer=**MergeLayerType**(), **MergeOutputType** mergeOutput=**MergeOutputType**(), **InitializationRuleType** initializeRule=**InitializationRuleType**())
*Create the **BRNN** (p. 610) object.*
- template<class LayerType, class... Args>
void **Add** (Args... args)
- void **Add** (**LayerTypes**< **CustomLayers**... > layer)
- double **Evaluate** (const arma::mat ¶meters, const size_t begin, const size_t batchSize, const bool deterministic)
Evaluate the bidirectional recurrent neural network with the given parameters.
- double **Evaluate** (const arma::mat ¶meters, const size_t begin, const size_t batchSize)
Evaluate the bidirectional recurrent neural network with the given parameters.
- template<typename GradType >
double **EvaluateWithGradient** (const arma::mat ¶meters, const size_t begin, GradType &gradient, const size_t batchSize)
Evaluate the bidirectional recurrent neural network with the given parameters.

- void **Gradient** (const arma::mat ¶meters, const size_t begin, arma::mat &gradient, const size_t batchSize)
Evaluate the gradient of the bidirectional recurrent neural network with the given parameters, and with respect to only one point in the dataset.
- size_t **NumFunctions** () const
Return the number of separable functions. (number of predictor points).
- const arma::mat & **Parameters** () const
Return the initial point for the optimization.
- arma::mat & **Parameters** ()
Modify the initial point for the optimization.
- void **Predict** (arma::cube predictors, arma::cube &results, const size_t batchSize=256)
Predict the responses to a given set of predictors.
- const arma::cube & **Predictors** () const
Get the matrix of data points (predictors).
- arma::cube & **Predictors** ()
Modify the matrix of data points (predictors).
- void **Reset** ()
Reset the state of the network.
- void **ResetParameters** ()
Reset the module information (weights/parameters).
- const arma::cube & **Responses** () const
Get the matrix of responses to the input data points.
- arma::cube & **Responses** ()
Modify the matrix of responses to the input data points.
- const size_t & **Rho** () const
Return the maximum length of backpropagation through time.
- size_t & **Rho** ()
Modify the maximum length of backpropagation through time.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the model.
- void **Shuffle** ()
Shuffle the order of function visitation.
- template<typename OptimizerType >
double **Train** (arma::cube predictors, arma::cube responses, OptimizerType &optimizer)
Train the bidirectional recurrent neural network on the given input data using the given optimizer.
- template<typename OptimizerType = ens::StandardSGD>
double **Train** (arma::cube predictors, arma::cube responses)
Train the bidirectional recurrent neural network on the given input data.

39.48.1 Detailed Description

```
template<typename OutputLayerType = NegativeLogLikelihood<>, typename MergeLayerType = Concat<>, typename MergeOutputType = LogSoftMax<>, typename InitializationRuleType = RandomInitialization, typename... CustomLayers>
class mlpack::ann::BRNN< OutputLayerType, MergeLayerType, MergeOutputType, InitializationRuleType, CustomLayers >
```

Implementation of a standard bidirectional recurrent neural network container.

Template Parameters

<i>OutputLayerType</i>	The output layer type used to evaluate the network.
<i>InitializationRuleType</i>	Rule used to initialize the weight matrix.

Definition at line 47 of file brnn.hpp.

39.48.2 Member Typedef Documentation

39.48.2.1 NetworkType

```
using NetworkType = BRNN<OutputLayerType, MergeLayerType, MergeOutputType, InitializationRuleType, CustomLayers...>
```

Convenience typedef for the internal model construction.

Definition at line 55 of file brnn.hpp.

39.48.3 Constructor & Destructor Documentation

39.48.3.1 BRNN()

```
BRNN (
    const size_t rho,
    const bool single = false,
    OutputLayerType outputLayer = OutputLayerType(),
    MergeLayerType mergeLayer = MergeLayerType(),
    MergeOutputType mergeOutput = MergeOutputType(),
    InitializationRuleType initializeRule = InitializationRuleType() )
```

Create the **BRNN** (p. 610) object.

Optionally, specify which initialize rule and performance function should be used.

If you want to pass in a parameter and discard the original parameter object, be sure to use `std::move` to avoid unnecessary copy.

Parameters

<i>rho</i>	Maximum number of steps to backpropagate through time (BPTT).
<i>single</i>	Predict only the last element of the input sequence.
<i>outputLayer</i>	Output layer used to evaluate the network.
<i>initializeRule</i>	Optional instantiated InitializationRule object for initializing the network parameter.

39.48.4 Member Function Documentation

39.48.4.1 Add() [1/2]

```
void Add (
    Args... args )
```

39.48.4.2 Add() [2/2]

```
void Add (
    LayerTypes< CustomLayers... > layer )
```

39.48.4.3 Evaluate() [1/2]

```
double Evaluate (
    const arma::mat & parameters,
    const size_t begin,
    const size_t batchSize,
    const bool deterministic )
```

Evaluate the bidirectional recurrent neural network with the given parameters.

This function is usually called by the optimizer to train the model.

Parameters

<i>parameters</i>	Matrix model parameters.
<i>begin</i>	Index of the starting point to use for objective function evaluation.
<i>batchSize</i>	Number of points to be passed at a time to use for objective function evaluation.
<i>deterministic</i>	Whether or not to train or test the model. Note some layer act differently in training or testing mode.

39.48.4.4 Evaluate() [2/2]

```
double Evaluate (
    const arma::mat & parameters,
```

```

    const size_t begin,
    const size_t batchSize )

```

Evaluate the bidirectional recurrent neural network with the given parameters.

This function is usually called by the optimizer to train the model. This just calls the other overload of **Evaluate()** (p. 613) with `deterministic = true`.

Parameters

<i>parameters</i>	Matrix model parameters.
<i>begin</i>	Index of the starting point to use for objective function evaluation.
<i>batchSize</i>	Number of points to be passed at a time to use for objective function evaluation.

39.48.4.5 EvaluateWithGradient()

```

double EvaluateWithGradient (
    const arma::mat & parameters,
    const size_t begin,
    GradType & gradient,
    const size_t batchSize )

```

Evaluate the bidirectional recurrent neural network with the given parameters.

This function is usually called by the optimizer to train the model. This just calls the other overload of **Evaluate()** (p. 613) with `deterministic = true`.

Parameters

<i>parameters</i>	Matrix model parameters.
<i>begin</i>	Index of the starting point to use for objective function evaluation.
<i>gradient</i>	Matrix to output gradient into.
<i>batchSize</i>	Number of points to be passed at a time to use for objective function evaluation.

39.48.4.6 Gradient()

```

void Gradient (
    const arma::mat & parameters,
    const size_t begin,
    arma::mat & gradient,
    const size_t batchSize )

```

Evaluate the gradient of the bidirectional recurrent neural network with the given parameters, and with respect to only one point in the dataset.

This is useful for optimizers such as SGD, which require a separable objective function.

Parameters

<i>parameters</i>	Matrix of the model parameters to be optimized.
<i>begin</i>	Index of the starting point to use for objective function gradient evaluation.
<i>gradient</i>	Matrix to output gradient into.
<i>batchSize</i>	Number of points to be processed as a batch for objective function gradient evaluation.

39.48.4.7 NumFunctions()

```
size_t NumFunctions ( ) const [inline]
```

Return the number of separable functions. (number of predictor points).

Definition at line 249 of file brnn.hpp.

39.48.4.8 Parameters() [1/2]

```
const arma::mat& Parameters ( ) const [inline]
```

Return the initial point for the optimization.

Definition at line 252 of file brnn.hpp.

39.48.4.9 Parameters() [2/2]

```
arma::mat& Parameters ( ) [inline]
```

Modify the initial point for the optimization.

Definition at line 254 of file brnn.hpp.

39.48.4.10 Predict()

```
void Predict (
    arma::cube predictors,
    arma::cube & results,
    const size_t batchSize = 256 )
```

Predict the responses to a given set of predictors.

The responses will reflect the output of the given output layer as returned by the output layer function.

If you want to pass in a parameter and discard the original parameter object, be sure to use `std::move` to avoid unnecessary copy.

The format of the data should be as follows:

- each slice should correspond to a time step
- each column should correspond to a data point
- each row should correspond to a dimension So, e.g., `predictors(i, j, k)` is the *i*'th dimension of the *j*'th data point at time slice *k*. The responses will be in the same format.

Parameters

<i>predictors</i>	Input predictors.
<i>results</i>	Matrix to put output predictions of responses into.
<i>batchSize</i>	Number of points to predict at once.

39.48.4.11 Predictors() [1/2]

```
const arma::cube& Predictors ( ) const [inline]
```

Get the matrix of data points (predictors).

Definition at line 267 of file `brnn.hpp`.

39.48.4.12 Predictors() [2/2]

```
arma::cube& Predictors ( ) [inline]
```

Modify the matrix of data points (predictors).

Definition at line 269 of file `brnn.hpp`.

References `BRNN< OutputLayerType, MergeLayerType, MergeOutputType, InitializationRuleType, CustomLayers >::Reset()`, `BRNN< OutputLayerType, MergeLayerType, MergeOutputType, InitializationRuleType, CustomLayers >::ResetParameters()`, and `BRNN< OutputLayerType, MergeLayerType, MergeOutputType, InitializationRuleType, CustomLayers >::serialize()`.

39.48.4.13 Reset()

```
void Reset ( )
```

Reset the state of the network.

This ensures that all internally-held gradients are set to 0, all memory cells are reset, and the parameters matrix is the right size.

Referenced by `BRNN< OutputLayerType, MergeLayerType, MergeOutputType, InitializationRuleType, CustomLayers >::Predictors()`.

39.48.4.14 ResetParameters()

```
void ResetParameters ( )
```

Reset the module information (weights/parameters).

Referenced by `BRNN< OutputLayerType, MergeLayerType, MergeOutputType, InitializationRuleType, CustomLayers >::Predictors()`.

39.48.4.15 Responses() [1/2]

```
const arma::cube& Responses ( ) const [inline]
```

Get the matrix of responses to the input data points.

Definition at line 262 of file `brnn.hpp`.

39.48.4.16 Responses() [2/2]

```
arma::cube& Responses ( ) [inline]
```

Modify the matrix of responses to the input data points.

Definition at line 264 of file `brnn.hpp`.

39.48.4.17 Rho() [1/2]

```
const size_t& Rho ( ) const [inline]
```

Return the maximum length of backpropagation through time.

Definition at line 257 of file brnn.hpp.

39.48.4.18 Rho() [2/2]

```
size_t& Rho ( ) [inline]
```

Modify the maximum length of backpropagation through time.

Definition at line 259 of file brnn.hpp.

39.48.4.19 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the model.

Referenced by BRNN< OutputLayerType, MergeLayerType, MergeOutputType, InitializationRuleType, CustomLayers >::Predictors().

39.48.4.20 Shuffle()

```
void Shuffle ( )
```

Shuffle the order of function visitation.

This may be called by the optimizer.

39.48.4.21 Train() [1/2]

```
double Train (
    arma::cube predictors,
    arma::cube responses,
    OptimizerType & optimizer )
```

Train the bidirectional recurrent neural network on the given input data using the given optimizer.

This will use the existing model parameters as a starting point for the optimization. If this is not what you want, then you should access the parameters vector directly with **Parameters()** (p. 616) and modify it as desired.

If you want to pass in a parameter and discard the original parameter object, be sure to use `std::move` to avoid unnecessary copy.

The format of the data should be as follows:

- each slice should correspond to a time step
- each column should correspond to a data point
- each row should correspond to a dimension So, e.g., `predictors(i, j, k)` is the *i*'th dimension of the *j*'th data point at time slice *k*.

Template Parameters

<i>OptimizerType</i>	Type of optimizer to use to train the model.
----------------------	--

Parameters

<i>predictors</i>	Input training variables.
<i>responses</i>	Outputs results from input training variables.
<i>optimizer</i>	Instantiated optimizer used to train the model.

39.48.4.22 Train() [2/2]

```
double Train (
    arma::cube predictors,
    arma::cube responses )
```

Train the bidirectional recurrent neural network on the given input data.

By default, the SGD optimization algorithm is used, but others can be specified (such as `ens::RMSprop`).

This will use the existing model parameters as a starting point for the optimization. If this is not what you want, then you should access the parameters vector directly with **Parameters()** (p. 616) and modify it as desired.

If you want to pass in a parameter and discard the original parameter object, be sure to use `std::move` to avoid unnecessary copy.

The format of the data should be as follows:

- each slice should correspond to a time step
- each column should correspond to a data point
- each row should correspond to a dimension So, e.g., `predictors(i, j, k)` is the *i*'th dimension of the *j*'th data point at time slice *k*.

Template Parameters

<i>OptimizerType</i>	Type of optimizer to use to train the model.
----------------------	--

Parameters

<i>predictors</i>	Input training variables.
<i>responses</i>	Outputs results from input training variables.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/ brnn.hpp`

39.49 Concat< InputDataType, OutputDataType, CustomLayers > Class Template Reference

Implementation of the **Concat** (p. 621) class.

Public Member Functions

- **Concat** (const bool model=false, const bool run=true)
*Create the **Concat** (p. 621) object using the specified parameters.*
- **Concat** (arma::Row< size_t > &inputSize, const size_t axis, const bool model=false, const bool run=true)
*Create the **Concat** (p. 621) object using the specified parameters.*
- **~Concat** ()
Destroy the layers held by the model.
- `template<typename LayerType >`
`void Add (const LayerType &layer)`
- `template<class LayerType , class... Args>`
`void Add (Args... args)`
- `void Add (LayerTypes< CustomLayers... > layer)`

- `template<typename eT >`
`void Backward (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)`
Ordinary feed backward pass of a neural network, using 3rd-order tensors as input, calculating the function $f(x)$ by propagating x backwards through f .
- `template<typename eT >`
`void Backward (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g, const size_t index)`
*This is the overload of **Backward()** (p. 625) that runs only a specific layer with the given input.*
- `arma::mat const & Delta () const`
Get the delta.e.
- `arma::mat & Delta ()`
Modify the delta.
- `template<typename eT >`
`void Forward (arma::Mat< eT > &&input, arma::Mat< eT > &&output)`
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- `template<typename eT >`
`void Gradient (arma::Mat< eT > &&, arma::Mat< eT > &&error, arma::Mat< eT > &&)`
- `template<typename eT >`
`void Gradient (arma::Mat< eT > &&input, arma::Mat< eT > &&error, arma::Mat< eT > &&gradient, const size_t index)`
- `arma::mat const & Gradient () const`
Get the gradient.
- `arma::mat & Gradient ()`
Modify the gradient.
- `arma::mat const & InputParameter () const`
- `arma::mat & InputParameter ()`
Modify the input parameter.
- `std::vector< LayerTypes< CustomLayers... > > & Model ()`
Return the model modules.
- `arma::mat const & OutputParameter () const`
Get the output parameter.
- `arma::mat & OutputParameter ()`
Modify the output parameter.
- `const arma::mat & Parameters () const`
Return the initial point for the optimization.
- `arma::mat & Parameters ()`
Modify the initial point for the optimization.
- `bool Run () const`
Get the value of run parameter.
- `bool & Run ()`
Modify the value of run parameter.
- `template<typename Archive >`
`void serialize (Archive &, const unsigned int)`
Serialize the layer.

39.49.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat, typename... CustomLayers>
class mlpack::ann::Concat< InputDataType, OutputDataType, CustomLayers >
```

Implementation of the **Concat** (p. 621) class.

The **Concat** (p. 621) class works as a feed-forward fully connected network container which plugs various layers together.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>CustomLayers</i>	Additional custom layers if required.

Definition at line 45 of file concat.hpp.

39.49.2 Constructor & Destructor Documentation

39.49.2.1 Concat() [1/2]

```
Concat (
    const bool model = false,
    const bool run = true )
```

Create the **Concat** (p. 621) object using the specified parameters.

Parameters

<i>model</i>	Expose all network modules.
<i>run</i>	Call the Forward/Backward method before the output is merged.

39.49.2.2 Concat() [2/2]

```
Concat (
    arma::Row< size_t > & inputSize,
    const size_t axis,
    const bool model = false,
    const bool run = true )
```

Create the **Concat** (p. 621) object using the specified parameters.

Parameters

<i>inputSize</i>	A vector denoting input size of each layer added.
<i>axis</i>	Concat (p. 621) axis.
<i>model</i>	Expose all network modules.
<i>run</i>	Call the Forward/Backward method before the output is merged.

39.49.2.3 ~Concat()

~ **Concat** ()

Destroy the layers held by the model.

39.49.3 Member Function Documentation

39.49.3.1 Add() [1/3]

```
void Add (
    const LayerType & layer ) [inline]
```

Definition at line 147 of file concat.hpp.

39.49.3.2 Add() [2/3]

```
void Add (
    Args... args ) [inline]
```

Definition at line 155 of file concat.hpp.

39.49.3.3 Add() [3/3]

```
void Add (
    LayerTypes< CustomLayers... > layer ) [inline]
```

Definition at line 162 of file concat.hpp.

39.49.3.4 Backward() [1/2]

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, using 3rd-order tensors as input, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.49.3.5 Backward() [2/2]

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g,
    const size_t index )
```

This is the overload of **Backward()** (p. 625) that runs only a specific layer with the given input.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.
<i>The</i>	index of the layer to run.

39.49.3.6 Delta() [1/2]

```
arma::mat const& Delta ( ) const [inline]
```

Get the delta.e.

Definition at line 195 of file concat.hpp.

39.49.3.7 Delta() [2/2]

```
arma::mat& Delta ( ) [inline]
```

Modify the delta.

Definition at line 197 of file concat.hpp.

39.49.3.8 Forward()

```
void Forward (
    arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.49.3.9 Gradient() [1/4]

```
void Gradient (
    arma::Mat< eT > && ,
    arma::Mat< eT > && error,
    arma::Mat< eT > && )
```

39.49.3.10 Gradient() [2/4]

```
void Gradient (
    arma::Mat< eT > && input,
    arma::Mat< eT > && error,
    arma::Mat< eT > && gradient,
    const size_t index )
```

39.49.3.11 Gradient() [3/4]

```
arma::mat const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 200 of file concat.hpp.

39.49.3.12 Gradient() [4/4]

```
arma::mat& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 202 of file concat.hpp.

References Concat< InputDataType, OutputDataType, CustomLayers >::serialize().

39.49.3.13 InputParameter() [1/2]

```
arma::mat const& InputParameter ( ) const [inline]
```

Definition at line 185 of file concat.hpp.

39.49.3.14 InputParameter() [2/2]

```
arma::mat& InputParameter ( ) [inline]
```

Modify the input parameter.

Definition at line 187 of file concat.hpp.

39.49.3.15 Model()

```
std::vector< LayerTypes<CustomLayers...> >& Model ( ) [inline]
```

Return the model modules.

Definition at line 165 of file concat.hpp.

39.49.3.16 OutputParameter() [1/2]

```
arma::mat const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 190 of file concat.hpp.

39.49.3.17 OutputParameter() [2/2]

```
arma::mat& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 192 of file concat.hpp.

39.49.3.18 Parameters() [1/2]

```
const arma::mat& Parameters ( ) const [inline]
```

Return the initial point for the optimization.

Definition at line 176 of file concat.hpp.

39.49.3.19 Parameters() [2/2]

```
arma::mat& Parameters ( ) [inline]
```

Modify the initial point for the optimization.

Definition at line 178 of file concat.hpp.

39.49.3.20 Run() [1/2]

```
bool Run ( ) const [inline]
```

Get the value of run parameter.

Definition at line 181 of file concat.hpp.

39.49.3.21 Run() [2/2]

```
bool& Run ( ) [inline]
```

Modify the value of run parameter.

Definition at line 183 of file concat.hpp.

39.49.3.22 serialize()

```
void serialize (
    Archive & ,
    const unsigned int )
```

Serialize the layer.

Referenced by Concat< InputDataType, OutputDataType, CustomLayers >::Gradient().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **concat.hpp**

39.50 Concatenate< InputDataType, OutputDataType > Class Template Reference

Implementation of the **Concatenate** (p. 630) module class.

Public Member Functions

- **Concatenate** ()
*Create the **Concatenate** (p. 630) object using the specified number of output units.*
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, const arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- OutputDataType const & **Concat** () const
Get the concat matrix.
- OutputDataType & **Concat** ()
Modify the delta.
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename eT >
void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

- OutputDataType const & **OutputParameter** () const

Get the output parameter.

- OutputDataType & **OutputParameter** ()

Modify the output parameter.

- OutputDataType const & **Parameters** () const

Get the parameters.

- OutputDataType & **Parameters** ()

Modify the parameters.

- template<typename Archive >

void **serialize** (Archive &, const unsigned int)

Serialize the layer.

39.50.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::Concatenate< InputDataType, OutputDataType >
```

Implementation of the **Concatenate** (p. 630) module class.

The **Concatenate** (p. 630) module concatenates a constant given matrix to the incoming data. Note: Users need to use the **Concat()** (p. 632) function to provide the concat matrix.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 36 of file concatenate.hpp.

39.50.2 Constructor & Destructor Documentation

39.50.2.1 Concatenate()

Concatenate ()

Create the **Concatenate** (p. 630) object using the specified number of output units.

39.50.3 Member Function Documentation

39.50.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
    const arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.50.3.2 Concat() [1/2]

```
OutputDataType const& Concat ( ) const [inline]
```

Get the concat matrix.

Definition at line 84 of file concatenate.hpp.

39.50.3.3 Concat() [2/2]

```
OutputDataType& Concat ( ) [inline]
```

Modify the delta.

Definition at line 86 of file concatenate.hpp.

39.50.3.4 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 79 of file concatenate.hpp.

39.50.3.5 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 81 of file concatenate.hpp.

39.50.3.6 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.50.3.7 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 74 of file concatenate.hpp.

39.50.3.8 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 76 of file concatenate.hpp.

39.50.3.9 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 69 of file concatenate.hpp.

39.50.3.10 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 71 of file concatenate.hpp.

39.50.3.11 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serialize the layer.

Definition at line 92 of file concatenate.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **concatenate.hpp**

39.51 ConcatPerformance< OutputLayerType, InputDataType, OutputDataType > Class Template Reference

Implementation of the concat performance class.

Public Member Functions

- **ConcatPerformance** (const size_t inSize=0, OutputLayerType &&outputLayer=OutputLayerType())
*Create the **ConcatPerformance** (p. 634) object.*
 - template<typename eT >
void **Backward** (const arma::Mat< eT > &&input, const arma::Mat< eT > &&target, arma::Mat< eT > &&output)
Ordinary feed backward pass of a neural network.
 - OutputDataType & **Delta** () const
Get the delta.
 - OutputDataType & **Delta** ()
Modify the delta.
 - template<typename eT >
double **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&target)
 - OutputDataType & **OutputParameter** () const
Get the output parameter.
 - OutputDataType & **OutputParameter** ()
Modify the output parameter.
 - template<typename Archive >
void **serialize** (Archive &, const unsigned int)
- Serialize the layer.*

39.51.1 Detailed Description

```
template<typename OutputLayerType = NegativeLogLikelihood<>, typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
```

```
class mlpack::ann::ConcatPerformance< OutputLayerType, InputDataType, OutputDataType >
```

Implementation of the concat performance class.

The class works as a feed-forward fully connected network container which plugs performance layers together.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 39 of file concat_performance.hpp.

39.51.2 Constructor & Destructor Documentation

39.51.2.1 ConcatPerformance()

```
ConcatPerformance (
    const size_t inSize = 0,
    OutputLayerType && outputLayer = OutputLayerType() )
```

Create the **ConcatPerformance** (p. 634) object.

Parameters

<i>inSize</i>	The number of inputs.
<i>outputLayer</i>	Output layer used to evaluate the network.

39.51.3 Member Function Documentation

39.51.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && input,
    const arma::Mat< eT > && target,
    arma::Mat< eT > && output )
```

Ordinary feed backward pass of a neural network.

The negative log likelihood layer expects that the input contains log-probabilities for each class. The layer also expects a class index, in the range between 1 and the number of classes, as target when calling the Forward function.

Parameters

<i>input</i>	The propagated input activation.
<i>target</i>	The target vector, that contains the class index in the range between 1 and the number of classes.
<i>output</i>	The calculated error.

39.51.3.2 Delta() [1/2]

```
OutputDataType& Delta ( ) const [inline]
```

Get the delta.

Definition at line 81 of file concat_performance.hpp.

39.51.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 83 of file concat_performance.hpp.

References ConcatPerformance< OutputLayerType, InputDataType, OutputDataType >::serialize().

39.51.3.4 Forward()

```
double Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && target )
```

39.51.3.5 OutputParameter() [1/2]

```
OutputDataType& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 76 of file concat_performance.hpp.

39.51.3.6 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 78 of file concat_performance.hpp.

39.51.3.7 serialize()

```
void serialize (
    Archive & ,
    const unsigned int )
```

Serialize the layer.

Referenced by ConcatPerformance< OutputLayerType, InputDataType, OutputDataType >::Delta().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **concat_performance.hpp**

39.52 Constant< InputDataType, OutputDataType > Class Template Reference

Implementation of the constant layer.

Public Member Functions

- **Constant** (const size_t outSize=0, const double scalar=0.0)
*Create the **Constant** (p. 638) object that outputs a given constant scalar value given any input value.*
- template<typename DataType >
void **Backward** (const DataType &&, DataType &&, DataType &&g)
Ordinary feed backward pass of a neural network.
- OutputDataType & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename InputType , typename OutputType >
void **Forward** (const InputType &&input, OutputType &&output)
Ordinary feed forward pass of a neural network.
- OutputDataType & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.52.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::Constant< InputDataType, OutputDataType >
```

Implementation of the constant layer.

The constant layer outputs a given constant value given any input value.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 34 of file constant.hpp.

39.52.2 Constructor & Destructor Documentation

39.52.2.1 Constant()

```

Constant (
    const size_t outSize = 0,
    const double scalar = 0.0 )

```

Create the **Constant** (p. 638) object that outputs a given constant scalar value given any input value.

Parameters

<i>outSize</i>	The number of output units.
<i>scalar</i>	The constant value used to create the constant output.

39.52.3 Member Function Documentation

39.52.3.1 Backward()

```

void Backward (
    const DataType && ,
    DataType && ,
    DataType && g )

```

Ordinary feed backward pass of a neural network.

The backward pass of the constant layer is returns always a zero output error matrix.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.52.3.2 Delta() [1/2]

```

OutputDataType& Delta ( ) const [inline]

```

Get the delta.

Definition at line 75 of file constant.hpp.

39.52.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 77 of file constant.hpp.

References `Constant< InputDataType, OutputDataType >::serialize()`.

39.52.3.4 Forward()

```
void Forward (
    const InputType && input,
    OutputType && output )
```

Ordinary feed forward pass of a neural network.

The forward pass fills the output with the specified constant parameter.

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.52.3.5 OutputParameter() [1/2]

```
OutputDataType& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 70 of file constant.hpp.

39.52.3.6 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 72 of file constant.hpp.

39.52.3.7 `serialize()`

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by `Constant< InputDataType, OutputDataType >::Delta()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ constant.hpp`

39.53 ConstInitialization Class Reference

This class is used to initialize weight matrix with constant values.

Public Member Functions

- **ConstInitialization** (const double initVal=0)
Create the ConstantInitialization object.
- `template<typename eT >`
`void Initialize (arma::Mat< eT > &W, const size_t rows, const size_t cols)`
Initialize the elements of the specified weight matrix.
- `template<typename eT >`
`void Initialize (arma::Cube< eT > &W, const size_t rows, const size_t cols, const size_t slices)`
Initialize the elements of the specified weight (3rd order tensor).
- `double const & InitValue () const`
Get the initialization value.
- `double & initValue ()`
Modify the initialization value.

39.53.1 Detailed Description

This class is used to initialize weight matrix with constant values.

Definition at line 25 of file `const_init.hpp`.

39.53.2 Constructor & Destructor Documentation

39.53.2.1 ConstInitialization()

```
ConstInitialization (
    const double initVal = 0 ) [inline]
```

Create the ConstantInitialization object.

Definition at line 31 of file const_init.hpp.

39.53.3 Member Function Documentation

39.53.3.1 Initialize() [1/2]

```
void Initialize (
    arma::Mat< eT > & W,
    const size_t rows,
    const size_t cols ) [inline]
```

Initialize the elements of the specified weight matrix.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.

Definition at line 42 of file const_init.hpp.

39.53.3.2 Initialize() [2/2]

```
void Initialize (
    arma::Cube< eT > & W,
    const size_t rows,
    const size_t cols,
    const size_t slices ) [inline]
```

Initialize the elements of the specified weight (3rd order tensor).

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.

Definition at line 56 of file const_init.hpp.

39.53.3.3 InitValue()

```
double const& InitValue ( ) const [inline]
```

Get the initialization value.

Definition at line 66 of file const_init.hpp.

39.53.3.4 initValue()

```
double& initValue ( ) [inline]
```

Modify the initialization value.

Definition at line 68 of file const_init.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ **const_init.hpp**

39.54 Convolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, InputDataType, OutputDataType > Class Template Reference

Implementation of the **Convolution** (p. 643) class.

Public Member Functions

- **Convolution** ()
*Create the **Convolution** (p. 643) object.*
- **Convolution** (const size_t inSize, const size_t outSize, const size_t kW, const size_t kH, const size_t dW=1, const size_t dH=1, const size_t padW=0, const size_t padH=0, const size_t inputWidth=0, const size_t inputHeight=0)
*Create the **Convolution** (p. 643) object using the specified number of input maps, output maps, filter size, stride and padding parameter.*
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, calculating the function f(x) by propagating x backwards through f.
- OutputDataType const & **Delta** () const
Get the delta.

- OutputDataType & **Delta** ()
Modify the delta.
- template<typename eT >
void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- template<typename eT >
void **Gradient** (const arma::Mat< eT > &&, arma::Mat< eT > &&error, arma::Mat< eT > &&gradient)
- OutputDataType const & **Gradient** () const
Get the gradient.
- OutputDataType & **Gradient** ()
Modify the gradient.
- size_t const & **InputHeight** () const
Get the input height.
- size_t & **InputHeight** ()
Modify the input height.
- InputDataType const & **InputParameter** () const
Get the input parameter.
- InputDataType & **InputParameter** ()
Modify the input parameter.
- size_t const & **InputWidth** () const
Get the input width.
- size_t & **InputWidth** ()
Modify input the width.
- size_t const & **OutputHeight** () const
Get the output height.
- size_t & **OutputHeight** ()
Modify the output height.
- OutputDataType const & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- size_t const & **OutputWidth** () const
Get the output width.
- size_t & **OutputWidth** ()
Modify the output width.
- OutputDataType const & **Parameters** () const
Get the parameters.
- OutputDataType & **Parameters** ()
Modify the parameters.
- void **Reset** ()
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.54.1 Detailed Description

```
template<typename ForwardConvolutionRule = NaiveConvolution<ValidConvolution>, typename BackwardConvolutionRule =
NaiveConvolution<FullConvolution>, typename GradientConvolutionRule = NaiveConvolution<ValidConvolution>, typename
InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::Convolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, InputDataType,
OutputDataType >
```

Implementation of the **Convolution** (p. 643) class.

The **Convolution** (p. 643) class represents a single layer of a neural network.

Template Parameters

<i>ForwardConvolutionRule</i>	Convolution (p. 643) to perform forward process.
<i>BackwardConvolutionRule</i>	Convolution (p. 643) to perform backward process.
<i>GradientConvolutionRule</i>	Convolution (p. 643) to calculate gradient.
<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 46 of file convolution.hpp.

39.54.2 Constructor & Destructor Documentation

39.54.2.1 Convolution() [1/2]

```
Convolution ( )
```

Create the **Convolution** (p. 643) object.

39.54.2.2 Convolution() [2/2]

```
Convolution (
    const size_t inSize,
    const size_t outSize,
    const size_t kW,
    const size_t kH,
    const size_t dW = 1,
    const size_t dH = 1,
    const size_t padW = 0,
    const size_t padH = 0,
    const size_t inputWidth = 0,
    const size_t inputHeight = 0 )
```

Create the **Convolution** (p. 643) object using the specified number of input maps, output maps, filter size, stride and padding parameter.

Parameters

<i>inSize</i>	The number of input maps.
<i>outSize</i>	The number of output maps.
<i>kW</i>	Width of the filter/kernel.
<i>kH</i>	Height of the filter/kernel.
<i>dW</i>	Stride of filter application in the x direction.
<i>dH</i>	Stride of filter application in the y direction.
<i>padW</i>	Padding width of the input.
<i>padH</i>	Padding height of the input.
<i>inputWidth</i>	The width of the input data.
<i>inputHeight</i>	The height of the input data.

39.54.3 Member Function Documentation

39.54.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.54.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 135 of file convolution.hpp.

39.54.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 137 of file convolution.hpp.

39.54.3.4 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.54.3.5 Gradient() [1/3]

```
void Gradient (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && error,
    arma::Mat< eT > && gradient )
```

39.54.3.6 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 140 of file convolution.hpp.

39.54.3.7 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 142 of file convolution.hpp.

39.54.3.8 InputHeight() [1/2]

```
size_t const& InputHeight ( ) const [inline]
```

Get the input height.

Definition at line 150 of file convolution.hpp.

39.54.3.9 InputHeight() [2/2]

```
size_t& InputHeight ( ) [inline]
```

Modify the input height.

Definition at line 152 of file convolution.hpp.

39.54.3.10 InputParameter() [1/2]

```
InputDataType const& InputParameter ( ) const [inline]
```

Get the input parameter.

Definition at line 125 of file convolution.hpp.

39.54.3.11 InputParameter() [2/2]

```
InputDataType& InputParameter ( ) [inline]
```

Modify the input parameter.

Definition at line 127 of file convolution.hpp.

39.54.3.12 InputWidth() [1/2]

```
size_t const& InputWidth ( ) const [inline]
```

Get the input width.

Definition at line 145 of file convolution.hpp.

39.54.3.13 InputWidth() [2/2]

```
size_t& InputWidth ( ) [inline]
```

Modify input the width.

Definition at line 147 of file convolution.hpp.

39.54.3.14 OutputHeight() [1/2]

```
size_t const& OutputHeight ( ) const [inline]
```

Get the output height.

Definition at line 160 of file convolution.hpp.

39.54.3.15 OutputHeight() [2/2]

```
size_t& OutputHeight ( ) [inline]
```

Modify the output height.

Definition at line 162 of file convolution.hpp.

References Convolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, InputDataType, OutputDataType >::serialize().

39.54.3.16 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 130 of file convolution.hpp.

39.54.3.17 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 132 of file convolution.hpp.

39.54.3.18 OutputWidth() [1/2]

```
size_t const& OutputWidth ( ) const [inline]
```

Get the output width.

Definition at line 155 of file convolution.hpp.

39.54.3.19 OutputWidth() [2/2]

```
size_t& OutputWidth ( ) [inline]
```

Modify the output width.

Definition at line 157 of file convolution.hpp.

39.54.3.20 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 120 of file convolution.hpp.

39.54.3.21 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 122 of file convolution.hpp.

39.54.3.22 Reset()

```
void Reset ( )
```

39.54.3.23 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by Convolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, Input←DataType, OutputDataType >::OutputHeight().

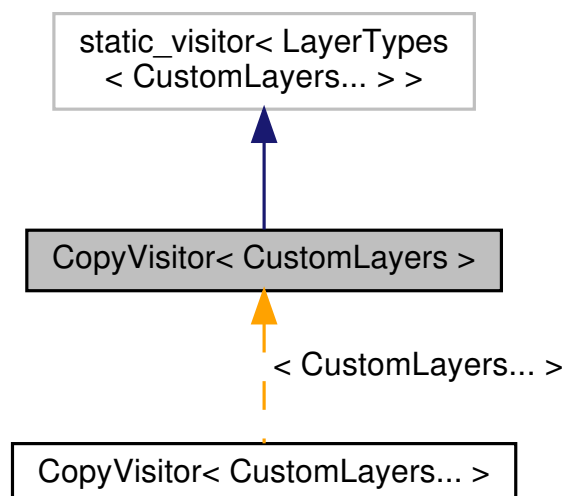
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **convolution.hpp**

39.55 CopyVisitor< CustomLayers > Class Template Reference

This visitor is to support copy constructor for neural network module.

Inheritance diagram for CopyVisitor< CustomLayers >:



Public Member Functions

- `template<typename LayerType >`
LayerTypes< CustomLayers... > **operator()** (LayerType *) const

39.55.1 Detailed Description

```
template<typename... CustomLayers>
class mlpack::ann::CopyVisitor< CustomLayers >
```

This visitor is to support copy constructor for neural network module.

We want a layer-wise copy rather than simple duplicate the pointer.

Definition at line 26 of file `copy_visitor.hpp`.

39.55.2 Member Function Documentation

39.55.2.1 `operator()`

```
LayerTypes<CustomLayers...> operator() (
    LayerType * ) const
```

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ copy_visitor.hpp`

39.56 CReLU< InputDataType, OutputDataType > Class Template Reference

A concatenated ReLU has two outputs, one ReLU and one negative ReLU, concatenated together.

Public Member Functions

- **CReLU** ()
*Create the **CReLU** (p. 652) object.*
- template<typename DataType >
void **Backward** (const DataType &&input, DataType &&gy, DataType &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename InputType , typename OutputType >
void **Forward** (const InputType &&input, OutputType &&output)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- OutputDataType const & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- template<typename Archive >
void **serialize** (Archive &, const unsigned int)
Serialize the layer.

39.56.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::CReLU< InputDataType, OutputDataType >
```

A concatenated ReLU has two outputs, one ReLU and one negative ReLU, concatenated together.

In other words, for positive x it produces $[x, 0]$, and for negative x it produces $[0, x]$. Because it has two outputs, **CReLU** (p. 652) doubles the output dimension.

Note: The **CReLU** (p. 652) doubles the output size.

For more information, see the following.

```
@inproceedings{ICML2016,
  title = {Understanding and Improving Convolutional Neural Networks
    via Concatenated Rectified Linear Units},
  author = {LWenling Shang, Kihyuk Sohn, Diogo Almeida, Honglak Lee},
  year = {2016}
}
```

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 49 of file c_relu.hpp.

39.56.2 Constructor & Destructor Documentation

39.56.2.1 CReLU()

CReLU ()

Create the **CReLU** (p. 652) object.

39.56.3 Member Function Documentation

39.56.3.1 Backward()

```
void Backward (
    const DataType && input,
    DataType && gy,
    DataType && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.56.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 86 of file c_relu.hpp.

39.56.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 88 of file c_relu.hpp.

References CReLU< InputDataType, OutputDataType >::serialize().

39.56.3.4 Forward()

```
void Forward (
    const InputType && input,
    OutputType && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Works only for 2D Tenosrs.

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.56.3.5 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 81 of file c_relu.hpp.

39.56.3.6 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 83 of file c_relu.hpp.

39.56.3.7 serialize()

```
void serialize (
    Archive & ,
    const unsigned int )
```

Serialize the layer.

Referenced by CReLU< InputDataType, OutputDataType >::Delta().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **c_relu.hpp**

39.57 CrossEntropyError< InputDataType, OutputDataType > Class Template Reference

The cross-entropy performance function measures the network's performance according to the cross-entropy between the input and target distributions.

Public Member Functions

- **CrossEntropyError** (const double eps=1e-10)
*Create the **CrossEntropyError** (p. 656) object.*
- template<typename InputType , typename TargetType , typename OutputType >
void **Backward** (const InputType &&input, const TargetType &&target, OutputType &&output)
Ordinary feed backward pass of a neural network.
- double **Eps** () const
Get the epsilon.
- double & **Eps** ()
Modify the epsilon.
- template<typename InputType , typename TargetType >
double **Forward** (const InputType &&input, const TargetType &&target)
Computes the cross-entropy function.
- OutputDataType & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.57.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::CrossEntropyError< InputDataType, OutputDataType >
```

The cross-entropy performance function measures the network's performance according to the cross-entropy between the input and target distributions.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 34 of file cross_entropy_error.hpp.

39.57.2 Constructor & Destructor Documentation

39.57.2.1 CrossEntropyError()

```
CrossEntropyError (
    const double eps = 1e-10 )
```

Create the **CrossEntropyError** (p. 656) object.

Parameters

<i>eps</i>	The minimum value used for computing logarithms and denominators in a numerically stable way.
------------	---

39.57.3 Member Function Documentation

39.57.3.1 Backward()

```
void Backward (
    const InputType && input,
    const TargetType && target,
    OutputType && output )
```

Ordinary feed backward pass of a neural network.

Parameters

<i>input</i>	The propagated input activation.
<i>target</i>	The target vector.
<i>output</i>	The calculated error.

39.57.3.2 Eps() [1/2]

```
double Eps ( ) const [inline]
```

Get the epsilon.

Definition at line 72 of file `cross_entropy_error.hpp`.

39.57.3.3 Eps() [2/2]

```
double& Eps ( ) [inline]
```

Modify the epsilon.

Definition at line 74 of file `cross_entropy_error.hpp`.

References `CrossEntropyError< InputDataType, OutputDataType >::serialize()`.

39.57.3.4 Forward()

```
double Forward (
    const InputType && input,
    const TargetType && target )
```

Computes the cross-entropy function.

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>target</i>	The target vector.

39.57.3.5 OutputParameter() [1/2]

```
OutputDataType& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 67 of file `cross_entropy_error.hpp`.

39.57.3.6 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 69 of file `cross_entropy_error.hpp`.

39.57.3.7 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by `CrossEntropyError< InputDataType, OutputDataType >::Eps()`.

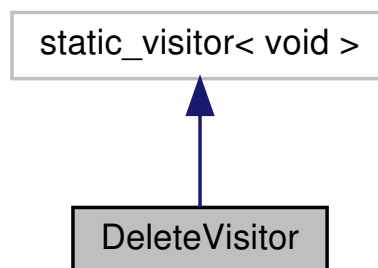
The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/ cross_entropy_error.hpp`

39.58 DeleteVisitor Class Reference

DeleteVisitor (p. 659) executes the destructor of the instantiated object.

Inheritance diagram for DeleteVisitor:



Public Member Functions

- `template<typename LayerType >`
`void operator() (LayerType *layer) const`
Execute the destructor.

39.58.1 Detailed Description

DeleteVisitor (p. 659) executes the destructor of the instantiated object.

Definition at line 27 of file delete_visitor.hpp.

39.58.2 Member Function Documentation

39.58.2.1 operator()

```
void operator() (
    LayerType * layer ) const
```

Execute the destructor.

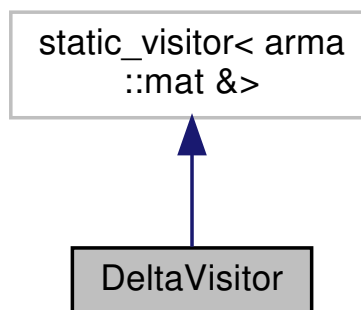
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **delete_visitor.hpp**

39.59 DeltaVisitor Class Reference

DeltaVisitor (p. 660) exposes the delta parameter of the given module.

Inheritance diagram for DeltaVisitor:



Public Member Functions

- `template<typename LayerType >`
`arma::mat & operator() (LayerType *layer) const`
Return the delta parameter.

39.59.1 Detailed Description

DeltaVisitor (p. 660) exposes the delta parameter of the given module.

Definition at line 27 of file delta_visitor.hpp.

39.59.2 Member Function Documentation

39.59.2.1 operator>()

```
arma::mat& operator() (
    LayerType * layer ) const
```

Return the delta parameter.

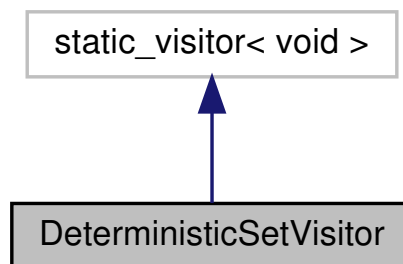
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **delta_visitor.hpp**

39.60 DeterministicSetVisitor Class Reference

DeterministicSetVisitor (p. 661) set the deterministic parameter given the deterministic value.

Inheritance diagram for DeterministicSetVisitor:



Public Member Functions

- **DeterministicSetVisitor** (const bool deterministic=true)
Set the deterministic parameter given the current deterministic value.
- template<typename LayerType >
void **operator()** (LayerType *layer) const
Set the deterministic parameter.

39.60.1 Detailed Description

DeterministicSetVisitor (p. 661) set the deterministic parameter given the deterministic value.

Definition at line 28 of file `deterministic_set_visitor.hpp`.

39.60.2 Constructor & Destructor Documentation

39.60.2.1 DeterministicSetVisitor()

```
DeterministicSetVisitor (  
    const bool deterministic = true )
```

Set the deterministic parameter given the current deterministic value.

39.60.3 Member Function Documentation

39.60.3.1 operator()

```
void operator() (  
    LayerType * layer ) const
```

Set the deterministic parameter.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ deterministic_set_visitor.hpp`

39.61 DiceLoss< InputDataType, OutputDataType > Class Template Reference

The dice loss performance function measures the network's performance according to the dice coefficient between the input and target distributions.

Public Member Functions

- **DiceLoss** (const double smooth=1)
*Create the **DiceLoss** (p. 662) object.*
- template<typename InputType , typename TargetType , typename OutputType >
void **Backward** (const InputType &&input, const TargetType &&target, OutputType &&output)
Ordinary feed backward pass of a neural network.
- template<typename InputType , typename TargetType >
double **Forward** (const InputType &&input, const TargetType &&target)
Computes the dice loss function.
- OutputDataType & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.
- double **Smooth** () const
Get the smooth.
- double & **Smooth** ()
Modify the smooth.

39.61.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::DiceLoss< InputDataType, OutputDataType >
```

The dice loss performance function measures the network's performance according to the dice coefficient between the input and target distributions.

For more information see the following.

{Milletari2016, author = {Fausto Milletari and Nassir Navab and Seyed{-}Ahmad Ahmadi}, title = {V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation}, journal = {CoRR}, volume = {abs/1606.04797}, year = {2016}, url = {http://arxiv.org/abs/1606.04797}, archivePrefix = {arXiv}, eprint = {1606.04797}, }

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 48 of file dice_loss.hpp.

39.61.2 Constructor & Destructor Documentation

39.61.2.1 DiceLoss()

```
DiceLoss (  
    const double smooth = 1 )
```

Create the **DiceLoss** (p. 662) object.

Parameters

<i>smooth</i>	The Laplace smoothing parameter.
---------------	----------------------------------

39.61.3 Member Function Documentation

39.61.3.1 Backward()

```
void Backward (  
    const InputType && input,  
    const TargetType && target,  
    OutputType && output )
```

Ordinary feed backward pass of a neural network.

Parameters

<i>input</i>	The propagated input activation.
<i>target</i>	The target vector.
<i>output</i>	The calculated error.

39.61.3.2 Forward()

```
double Forward (  
    const InputType && input,  
    const TargetType && target )
```

Computes the dice loss function.

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>target</i>	The target vector.

39.61.3.3 OutputParameter() [1/2]

```
OutputDataType& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 80 of file dice_loss.hpp.

39.61.3.4 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 82 of file dice_loss.hpp.

39.61.3.5 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by DiceLoss< InputDataType, OutputDataType >::Smooth().

39.61.3.6 Smooth() [1/2]

```
double Smooth ( ) const [inline]
```

Get the smooth.

Definition at line 85 of file dice_loss.hpp.

39.61.3.7 Smooth() [2/2]

```
double& Smooth ( ) [inline]
```

Modify the smooth.

Definition at line 87 of file dice_loss.hpp.

References DiceLoss< InputDataType, OutputDataType >::serialize().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/ **dice_loss.hpp**

39.62 DropConnect< InputDataType, OutputDataType > Class Template Reference

The **DropConnect** (p. 666) layer is a regularizer that randomly with probability ratio sets the connection values to zero and scales the remaining elements by factor $1/(1 - \text{ratio})$.

Public Member Functions

- **DropConnect** ()
*Create the **DropConnect** (p. 666) object.*
- **DropConnect** (const size_t inSize, const size_t outSize, const double ratio=0.5)
*Creates the **DropConnect** (p. 666) Layer as a **Linear** (p. 776) Object that takes input size, output size and ratio as parameter.*
- **~DropConnect** ()
- template<typename eT >
void **Backward** (arma::Mat< eT > &&input, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
*Ordinary feed backward pass of the **DropConnect** (p. 666) layer.*
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- bool **Deterministic** () const
The value of the deterministic parameter.
- bool & **Deterministic** ()
Modify the value of the deterministic parameter.
- template<typename eT >
void **Forward** (arma::Mat< eT > &&input, arma::Mat< eT > &&output)
*Ordinary feed forward pass of the **DropConnect** (p. 666) layer.*
- template<typename eT >
void **Gradient** (arma::Mat< eT > &&input, arma::Mat< eT > &&error, arma::Mat< eT > &&g)
Calculate the gradient using the output delta and the input activation.
- OutputDataType const & **Gradient** () const
Get the gradient.

- OutputDataType & **Gradient** ()
Modify the gradient.
 - std::vector< **LayerTypes**<> > & **Model** ()
Get the model modules.
 - OutputDataType const & **OutputParameter** () const
Get the output parameter.
 - OutputDataType & **OutputParameter** ()
Modify the output parameter.
 - OutputDataType const & **Parameters** () const
Get the parameters.
 - OutputDataType & **Parameters** ()
Modify the parameters.
 - double **Ratio** () const
The probability of setting a value to zero.
 - void **Ratio** (const double r)
Modify the probability of setting a value to zero.
 - template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
- Serialize the layer.*

39.62.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::DropConnect< InputDataType, OutputDataType >
```

The **DropConnect** (p.666) layer is a regularizer that randomly with probability ratio sets the connection values to zero and scales the remaining elements by factor $1 / (1 - \text{ratio})$.

The output is scaled with $1 / (1 - p)$ when deterministic is false. In the deterministic mode(during testing), the layer just computes the output. The output is computed according to the input layer. If no input layer is given, it will take a linear layer as default.

Note: During training you should set deterministic to false and during testing you should set deterministic to true.

For more information, see the following.

```
@inproceedings{WanICML2013,
  title={Regularization of Neural Networks using DropConnect},
  booktitle = {Proceedings of the 30th International Conference on Machine
    Learning(ICML - 13)},
  author = {Li Wan and Matthew Zeiler and Sixin Zhang and Yann L. Cun and
    Rob Fergus},
  year = {2013}
}
```

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 62 of file dropconnect.hpp.

39.62.2 Constructor & Destructor Documentation

39.62.2.1 DropConnect() [1/2]

```
DropConnect ( )
```

Create the **DropConnect** (p. 666) object.

39.62.2.2 DropConnect() [2/2]

```
DropConnect (  
    const size_t inSize,  
    const size_t outSize,  
    const double ratio = 0.5 )
```

Creates the **DropConnect** (p. 666) Layer as a **Linear** (p. 776) Object that takes input size, output size and ratio as parameter.

Parameters

<i>inSize</i>	The number of input units.
<i>outSize</i>	The number of output units.
<i>ratio</i>	The probability of setting a value to zero.

39.62.2.3 ~DropConnect()

```
~ DropConnect ( )
```

39.62.3 Member Function Documentation

39.62.3.1 Backward()

```
void Backward (
    arma::Mat< eT > && input,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of the **DropConnect** (p. 666) layer.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.62.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 129 of file dropconnect.hpp.

39.62.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 131 of file dropconnect.hpp.

39.62.3.4 Deterministic() [1/2]

```
bool Deterministic ( ) const [inline]
```

The value of the deterministic parameter.

Definition at line 139 of file dropconnect.hpp.

39.62.3.5 Deterministic() [2/2]

```
bool& Deterministic ( ) [inline]
```

Modify the value of the deterministic parameter.

Definition at line 142 of file dropconnect.hpp.

39.62.3.6 Forward()

```
void Forward (
    arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of the **DropConnect** (p. 666) layer.

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.62.3.7 Gradient() [1/3]

```
void Gradient (
    arma::Mat< eT > && input,
    arma::Mat< eT > && error,
    arma::Mat< eT > && )
```

Calculate the gradient using the output delta and the input activation.

Parameters

<i>input</i>	The propagated input.
<i>d</i>	The calculated error.
<i>g</i>	The calculated gradient.

39.62.3.8 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 134 of file dropconnect.hpp.

39.62.3.9 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 136 of file dropconnect.hpp.

39.62.3.10 Model()

```
std::vector< LayerTypes<> >& Model ( ) [inline]
```

Get the model modules.

Definition at line 116 of file dropconnect.hpp.

39.62.3.11 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 124 of file dropconnect.hpp.

39.62.3.12 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 126 of file dropconnect.hpp.

39.62.3.13 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 119 of file dropconnect.hpp.

39.62.3.14 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 121 of file dropconnect.hpp.

39.62.3.15 Ratio() [1/2]

```
double Ratio ( ) const [inline]
```

The probability of setting a value to zero.

Definition at line 145 of file dropconnect.hpp.

39.62.3.16 Ratio() [2/2]

```
void Ratio (
    const double r ) [inline]
```

Modify the probability of setting a value to zero.

Definition at line 148 of file dropconnect.hpp.

References `DropConnect< InputDataType, OutputDataType >::serialize()`.

39.62.3.17 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by DropConnect< InputDataType, OutputDataType >::Ratio().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **dropconnect.hpp**

39.63 Dropout< InputDataType, OutputDataType > Class Template Reference

The dropout layer is a regularizer that randomly with probability 'ratio' sets input values to zero and scales the remaining elements by factor $1 / (1 - \text{ratio})$ rather than during test time so as to keep the expected sum same.

Public Member Functions

- **Dropout** (const double ratio=0.5)
*Create the **Dropout** (p. 673) object using the specified ratio parameter.*
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of the dropout layer.
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- bool **Deterministic** () const
The value of the deterministic parameter.
- bool & **Deterministic** ()
Modify the value of the deterministic parameter.
- template<typename eT >
void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Ordinary feed forward pass of the dropout layer.
- OutputDataType const & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- double **Ratio** () const
The probability of setting a value to zero.
- void **Ratio** (const double r)
Modify the probability of setting a value to zero.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.63.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::Dropout< InputDataType, OutputDataType >
```

The dropout layer is a regularizer that randomly with probability 'ratio' sets input values to zero and scales the remaining elements by factor $1 / (1 - \text{ratio})$ rather than during test time so as to keep the expected sum same.

In the deterministic mode (during testing), there is no change in the input.

Note: During training you should set deterministic to false and during testing you should set deterministic to true.

For more information, see the following.

```
@article{Hinton2012,
  author = {Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky,
            Ilya Sutskever, Ruslan Salakhutdinov},
  title  = {Improving neural networks by preventing co-adaptation of feature
            detectors},
  journal = {CoRR},
  volume = {abs/1207.0580},
  year   = {2012},
}
```

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 52 of file dropout.hpp.

39.63.2 Constructor & Destructor Documentation

39.63.2.1 Dropout()

```
Dropout (
    const double ratio = 0.5 )
```

Create the **Dropout** (p. 673) object using the specified ratio parameter.

Parameters

<i>ratio</i>	The probability of setting a value to zero.
--------------	---

39.63.3 Member Function Documentation

39.63.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of the dropout layer.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.63.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 89 of file dropout.hpp.

39.63.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 91 of file dropout.hpp.

39.63.3.4 Deterministic() [1/2]

```
bool Deterministic ( ) const [inline]
```

The value of the deterministic parameter.

Definition at line 94 of file dropout.hpp.

39.63.3.5 Deterministic() [2/2]

```
bool& Deterministic ( ) [inline]
```

Modify the value of the deterministic parameter.

Definition at line 96 of file dropout.hpp.

39.63.3.6 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of the dropout layer.

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.63.3.7 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 84 of file dropout.hpp.

39.63.3.8 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 86 of file dropout.hpp.

39.63.3.9 Ratio() [1/2]

```
double Ratio ( ) const [inline]
```

The probability of setting a value to zero.

Definition at line 99 of file dropout.hpp.

39.63.3.10 Ratio() [2/2]

```
void Ratio (
    const double r ) [inline]
```

Modify the probability of setting a value to zero.

Definition at line 102 of file dropout.hpp.

References Dropout< InputDataType, OutputDataType >::serialize().

39.63.3.11 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by Dropout< InputDataType, OutputDataType >::Ratio().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **dropout.hpp**

39.64 EarthMoverDistance< InputDataType, OutputDataType > Class Template Reference

The earth mover distance function measures the network's performance according to the Kantorovich-Rubinstein duality approximation.

Public Member Functions

- **EarthMoverDistance** ()
Create the **EarthMoverDistance** (p. 677) object.
- template<typename InputType , typename TargetType , typename OutputType >
void **Backward** (const InputType &&input, const TargetType &&target, OutputType &&output)
Ordinary feed backward pass of a neural network.
- template<typename InputType , typename TargetType >
double **Forward** (const InputType &&input, const TargetType &&target)
Ordinary feed forward pass of a neural network.
- OutputDataType & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.64.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::EarthMoverDistance< InputDataType, OutputDataType >
```

The earth mover distance function measures the network's performance according to the Kantorovich-Rubinstein duality approximation.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 33 of file earth_mover_distance.hpp.

39.64.2 Constructor & Destructor Documentation

39.64.2.1 EarthMoverDistance()

EarthMoverDistance ()

Create the **EarthMoverDistance** (p. 677) object.

39.64.3 Member Function Documentation

39.64.3.1 Backward()

```
void Backward (
    const InputType && input,
    const TargetType && target,
    OutputType && output )
```

Ordinary feed backward pass of a neural network.

Parameters

<i>input</i>	The propagated input activation.
<i>target</i>	The target vector.
<i>output</i>	The calculated error.

39.64.3.2 Forward()

```
double Forward (
    const InputType && input,
    const TargetType && target )
```

Ordinary feed forward pass of a neural network.

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>target</i>	The target vector.

39.64.3.3 OutputParameter() [1/2]

```
OutputDataType& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 63 of file earth_mover_distance.hpp.

39.64.3.4 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 65 of file earth_mover_distance.hpp.

References EarthMoverDistance< InputDataType, OutputDataType >::serialize().

39.64.3.5 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by EarthMoverDistance< InputDataType, OutputDataType >::OutputParameter().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/ **earth_mover_distance.hpp**

39.65 ELU< InputDataType, OutputDataType > Class Template Reference

The **ELU** (p. 680) activation function, defined by.

Public Member Functions

- **ELU** ()
*Create the **ELU** (p. 680) object.*
- **ELU** (const double alpha)
*Create the **ELU** (p. 680) object using the specified parameter.*
- double const & **Alpha** () const
Get the non zero gradient.
- double & **Alpha** ()
Modify the non zero gradient.
- template<typename DataType >
void **Backward** (const DataType &&input, DataType &&gy, DataType &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()

Modify the delta.

- `template<typename InputType , typename OutputType >`

`void Forward (const InputType &&input, OutputType &&output)`

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

- `double const & Lambda () const`

Get the lambda parameter.

- `OutputDataType const & OutputParameter () const`

Get the output parameter.

- `OutputDataType & OutputParameter ()`

Modify the output parameter.

- `template<typename Archive >`

`void serialize (Archive &ar, const unsigned int)`

Serialize the layer.

39.65.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::ELU< InputDataType, OutputDataType >
```

The **ELU** (p. 680) activation function, defined by.

$$f(x) = \begin{cases} x & : x > 0 \\ \alpha(e^x - 1) & : x \leq 0 \end{cases}$$

$$f'(x) = \begin{cases} 1 & : x > 0 \\ f(x) + \alpha & : x \leq 0 \end{cases}$$

For more information, read the following paper:

```
@article{Clevert2015,
  author = {Djork{-}Arn{\'{e}} Clevert and Thomas Unterthiner and
    Sepp Hochreiter},
  title = {Fast and Accurate Deep Network Learning by Exponential Linear
    Units (ELUs)},
  journal = {CoRR},
  year = {2015}
}
```

The SELU activation function is defined by

$$f(x) = \begin{cases} \lambda * x & : x > 0 \\ \lambda * \alpha(e^x - 1) & : x \leq 0 \end{cases}$$

$$f'(x) = \begin{cases} \lambda & : x > 0 \\ f(x) + \lambda * \alpha & : x \leq 0 \end{cases}$$

For more information, read the following paper:

```
@article{Klambauer2017,
  author = {Gunter Klambauer and Thomas Unterthiner and
    Andreas Mayr},
  title = {Self-Normalizing Neural Networks},
  journal = {Advances in Neural Information Processing Systems},
  year = {2017}
}
```

In the deterministic mode, there is no computation of the derivative.

Note

During training deterministic should be set to false and during testing/inference deterministic should be set to true. Make sure to use SELU activation function with normalized inputs and weights initialized with Lecun Normal Initialization.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 109 of file elu.hpp.

39.65.2 Constructor & Destructor Documentation**39.65.2.1 ELU()** [1/2]

```
ELU ( )
```

Create the **ELU** (p. 680) object.

NOTE: Use this constructor for SELU activation function.

39.65.2.2 ELU() [2/2]

```
ELU (
    const double alpha )
```

Create the **ELU** (p. 680) object using the specified parameter.

The non zero gradient for negative inputs can be adjusted by specifying the **ELU** (p. 680) hyperparameter alpha (alpha > 0).

Note

Use this constructor for **ELU** (p. 680) activation function.

Parameters

<i>alpha</i>	Scale parameter for the negative factor.
--------------	--

39.65.3 Member Function Documentation

39.65.3.1 Alpha() [1/2]

```
double const& Alpha ( ) const [inline]
```

Get the non zero gradient.

Definition at line 162 of file elu.hpp.

39.65.3.2 Alpha() [2/2]

```
double& Alpha ( ) [inline]
```

Modify the non zero gradient.

Definition at line 164 of file elu.hpp.

39.65.3.3 Backward()

```
void Backward (
    const DataType && input,
    DataType && gy,
    DataType && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation $f(x)$.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.65.3.4 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 157 of file elu.hpp.

39.65.3.5 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 159 of file elu.hpp.

39.65.3.6 Forward()

```
void Forward (
    const InputType && input,
    OutputType && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.65.3.7 Lambda()

```
double const& Lambda ( ) const [inline]
```

Get the lambda parameter.

Definition at line 167 of file elu.hpp.

References `ELU< InputDataType, OutputDataType >::serialize()`.

39.65.3.8 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 152 of file elu.hpp.

39.65.3.9 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 154 of file elu.hpp.

39.65.3.10 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by ELU< InputDataType, OutputDataType >::Lambda().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **elu.hpp**

39.66 FastLSTM< InputDataType, OutputDataType > Class Template Reference

An implementation of a faster version of the Fast **LSTM** (p. 801) network layer.

Public Types

- typedef OutputDataType::elem_type **ElemType**
- typedef InputDataType::elem_type **InputElemType**

Public Member Functions

- **FastLSTM** ()
*Create the Fast **LSTM** (p. 801) object.*
 - **FastLSTM** (const size_t inSize, const size_t outSize, const size_t rho=std::numeric_limits< size_t >::max())
*Create the Fast **LSTM** (p. 801) layer object using the specified parameters.*
 - template<typename InputType , typename ErrorType , typename GradientType >
void **Backward** (const InputType &&input, ErrorType &&gy, GradientType &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
 - OutputDataType const & **Delta** () const
Get the delta.
 - OutputDataType & **Delta** ()
Modify the delta.
 - template<typename InputType , typename OutputType >
void **Forward** (InputType &&input, OutputType &&output)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
 - template<typename InputType , typename ErrorType , typename GradientType >
void **Gradient** (InputType &&input, ErrorType &&error, GradientType &&gradient)
 - OutputDataType const & **Gradient** () const
Get the gradient.
 - OutputDataType & **Gradient** ()
Modify the gradient.
 - OutputDataType const & **OutputParameter** () const
Get the output parameter.
 - OutputDataType & **OutputParameter** ()
Modify the output parameter.
 - OutputDataType const & **Parameters** () const
Get the parameters.
 - OutputDataType & **Parameters** ()
Modify the parameters.
 - void **Reset** ()
 - void **ResetCell** (const size_t size)
 - size_t **Rho** () const
Get the maximum number of steps to backpropagate through time (BPTT).
 - size_t & **Rho** ()
Modify the maximum number of steps to backpropagate through time (BPTT).
 - template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
- Serialize the layer.*

39.66.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::FastLSTM< InputDataType, OutputDataType >
```

An implementation of a faster version of the Fast **LSTM** (p. 801) network layer.

Basically by combining the calculation of the input, forget, output gates and hidden state in a single step. The standard formula changes as follows:

$$i = \text{sigmoid}(W \cdot x + W \cdot h + b) \quad (39.1)$$

$$f = \text{sigmoid}(W \cdot x + W \cdot h + b) \quad (39.2)$$

$$z = \tanh(W \cdot x + W \cdot h + b) \quad (39.3)$$

$$c = f \cdot c + i \cdot z \quad (39.4)$$

$$o = \text{sigmoid}(W \cdot x + W \cdot h + b) \quad (39.5)$$

$$h = o \cdot \tanh(c) \quad (39.6)$$

Note that **FastLSTM** (p. 685) network layer does not use peephole connections between the cell and gates.

For more information, see the following.

```
@article{Hochreiter1997,
  author = {Hochreiter, Sepp and Schmidhuber, J\"{u}rgen},
  title = {Long Short-term Memory},
  journal = {Neural Comput.},
  year = {1997}
}
```

See also

LSTM (p. 801) for a standard implementation of the **LSTM** (p. 801) layer.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 61 of file fast_lstm.hpp.

39.66.2 Member Typedef Documentation

39.66.2.1 ElemType

```
typedef OutputDataType::elem_type ElemType
```

Definition at line 66 of file fast_lstm.hpp.

39.66.2.2 InputElemType

```
typedef InputDataType::elem_type InputElemType
```

Definition at line 65 of file fast_lstm.hpp.

39.66.3 Constructor & Destructor Documentation

39.66.3.1 FastLSTM() [1/2]

```
FastLSTM ( )
```

Create the Fast **LSTM** (p. 801) object.

39.66.3.2 FastLSTM() [2/2]

```
FastLSTM (
    const size_t inSize,
    const size_t outSize,
    const size_t rho = std::numeric_limits< size_t >::max() )
```

Create the Fast **LSTM** (p. 801) layer object using the specified parameters.

Parameters

<i>inSize</i>	The number of input units.
<i>outSize</i>	The number of output units.
<i>rho</i>	Maximum number of steps to backpropagate through time (BPTT).

39.66.4 Member Function Documentation

39.66.4.1 Backward()

```
void Backward (
    const InputType && input,
```



```
ErrorType && gy,  
GradientType && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.66.4.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 147 of file fast_lstm.hpp.

39.66.4.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 149 of file fast_lstm.hpp.

39.66.4.4 Forward()

```
void Forward (  
    InputType && input,  
    OutputType && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.66.4.5 Gradient() [1/3]

```
void Gradient (
    InputType && input,
    ErrorType && error,
    GradientType && gradient )
```

39.66.4.6 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 152 of file fast_lstm.hpp.

39.66.4.7 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 154 of file fast_lstm.hpp.

References FastLSTM< InputDataType, OutputDataType >::serialize().

39.66.4.8 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 142 of file fast_lstm.hpp.

39.66.4.9 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 144 of file fast_lstm.hpp.

39.66.4.10 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 137 of file fast_lstm.hpp.

39.66.4.11 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 139 of file fast_lstm.hpp.

39.66.4.12 Reset()

```
void Reset ( )
```

39.66.4.13 ResetCell()

```
void ResetCell (
    const size_t size )
```

39.66.4.14 Rho() [1/2]

```
size_t Rho ( ) const [inline]
```

Get the maximum number of steps to backpropagate through time (BPTT).

Definition at line 132 of file fast_lstm.hpp.

39.66.4.15 Rho() [2/2]

```
size_t& Rho ( ) [inline]
```

Modify the maximum number of steps to backpropagate through time (BPTT).

Definition at line 134 of file fast_lstm.hpp.

39.66.4.16 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by FastLSTM< InputDataType, OutputDataType >::Gradient().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **fast_lstm.hpp**

39.67 FFN< OutputLayerType, InitializationRuleType, CustomLayers > Class Template Reference

Implementation of a standard feed forward network.

Public Types

- using **NetworkType** = **FFN**< OutputLayerType, InitializationRuleType >
Convenience typedef for the internal model construction.

Public Member Functions

- **FFN** (OutputLayerType outputLayer=OutputLayerType(), InitializationRuleType initializeRule=InitializationRuleType())
*Create the **FFN** (p. 692) object.*
- **FFN** (const **FFN** &)
Copy constructor.
- **FFN** (**FFN** &&)
Move constructor.
- **~FFN** ()
Destructor to release allocated memory.
- template<class LayerType , class... Args>
void **Add** (Args... args)
- void **Add** (**LayerTypes**< CustomLayers... > layer)
- double **Backward** (arma::mat targets, arma::mat &gradients)
Perform the backward pass of the data in real batch mode.
- double **Evaluate** (arma::mat predictors, arma::mat responses)
Evaluate the feedforward network with the given predictors and responses.
- double **Evaluate** (const arma::mat ¶meters)
Evaluate the feedforward network with the given parameters.
- double **Evaluate** (const arma::mat ¶meters, const size_t begin, const size_t batchSize, const bool deterministic)
Evaluate the feedforward network with the given parameters, but using only a number of data points.
- double **Evaluate** (const arma::mat ¶meters, const size_t begin, const size_t batchSize)
Evaluate the feedforward network with the given parameters, but using only a number of data points.
- template<typename GradType >
double **EvaluateWithGradient** (const arma::mat ¶meters, GradType &gradient)
Evaluate the feedforward network with the given parameters.
- template<typename GradType >
double **EvaluateWithGradient** (const arma::mat ¶meters, const size_t begin, GradType &gradient, const size_t batchSize)
Evaluate the feedforward network with the given parameters, but using only a number of data points.
- void **Forward** (arma::mat inputs, arma::mat &results)
Perform the forward pass of the data in real batch mode.
- void **Forward** (arma::mat inputs, arma::mat &results, const size_t begin, const size_t end)
Perform a partial forward pass of the data.
- void **Gradient** (const arma::mat ¶meters, const size_t begin, arma::mat &gradient, const size_t batchSize)
Evaluate the gradient of the feedforward network with the given parameters, and with respect to only a number of points in the dataset.
- size_t **NumFunctions** () const
Return the number of separable functions (the number of predictor points).
- **FFN** & **operator=** (**FFN**)
Copy/move assignment operator.
- const arma::mat & **Parameters** () const
Return the initial point for the optimization.
- arma::mat & **Parameters** ()
Modify the initial point for the optimization.
- void **Predict** (arma::mat predictors, arma::mat &results)

- Predict the responses to a given set of predictors.*
- const arma::mat & **Predictors** () const
Get the matrix of data points (predictors).
- arma::mat & **Predictors** ()
Modify the matrix of data points (predictors).
- void **ResetParameters** ()
Reset the module information (weights/parameters).
- const arma::mat & **Responses** () const
Get the matrix of responses to the input data points.
- arma::mat & **Responses** ()
Modify the matrix of responses to the input data points.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the model.
- void **Shuffle** ()
Shuffle the order of function visitation.
- template<typename OptimizerType >
double **Train** (arma::mat predictors, arma::mat responses, OptimizerType &optimizer)
Train the feedforward network on the given input data using the given optimizer.
- template<typename OptimizerType = ens::RMSProp>
double **Train** (arma::mat predictors, arma::mat responses)
Train the feedforward network on the given input data.

39.67.1 Detailed Description

template<typename OutputLayerType = NegativeLogLikelihood<>, typename InitializationRuleType = RandomInitialization, typename... CustomLayers>

class mlpack::ann::FFN< OutputLayerType, InitializationRuleType, CustomLayers >

Implementation of a standard feed forward network.

Template Parameters

<i>OutputLayerType</i>	The output layer type used to evaluate the network.
<i>InitializationRuleType</i>	Rule used to initialize the weight matrix.
<i>CustomLayers</i>	Any set of custom layers that could be a part of the feed forward network.

Definition at line 51 of file ffn.hpp.

39.67.2 Member Typedef Documentation

39.67.2.1 NetworkType

```
using NetworkType = FFN<OutputLayerType, InitializationRuleType>
```

Convenience typedef for the internal model construction.

Definition at line 55 of file ffnn.hpp.

39.67.3 Constructor & Destructor Documentation

39.67.3.1 FFN() [1/3]

```
FFN (
    OutputLayerType outputLayer = OutputLayerType(),
    InitializationRuleType initializeRule = InitializationRuleType() )
```

Create the **FFN** (p. 692) object.

Optionally, specify which initialize rule and performance function should be used.

If you want to pass in a parameter and discard the original parameter object, be sure to use `std::move` to avoid unnecessary copy.

Parameters

<i>outputLayer</i>	Output layer used to evaluate the network.
<i>initializeRule</i>	Optional instantiated InitializationRule object for initializing the network parameter.

39.67.3.2 FFN() [2/3]

```
FFN (
    const FFN< OutputLayerType, InitializationRuleType, CustomLayers > & )
```

Copy constructor.

39.67.3.3 FFN() [3/3]

```
FFN (
    FFN< OutputLayerType, InitializationRuleType, CustomLayers > && )
```

Move constructor.

39.67.3.4 ~FFN()

~ **FFN** ()

Destructor to release allocated memory.

39.67.4 Member Function Documentation

39.67.4.1 Add() [1/2]

```
void Add (
    Args... args ) [inline]
```

Definition at line 251 of file ffnn.hpp.

39.67.4.2 Add() [2/2]

```
void Add (
    LayerTypes< CustomLayers... > layer ) [inline]
```

Definition at line 258 of file ffnn.hpp.

39.67.4.3 Backward()

```
double Backward (
    arma::mat targets,
    arma::mat & gradients )
```

Perform the backward pass of the data in real batch mode.

Forward and Backward should be used as a pair, and they are designed mainly for advanced users. User should try to use Predict and Train unless those two functions can't satisfy some special requirements.

Parameters

<i>targets</i>	The training target.
<i>gradients</i>	Computed gradients.

Returns

Training error of the current pass.

Referenced by FFN< OutputLayerType, InitializationRuleType, CustomLayers >::Predictors().

39.67.4.4 Evaluate() [1/4]

```
double Evaluate (
    arma::mat predictors,
    arma::mat responses )
```

Evaluate the feedforward network with the given predictors and responses.

This functions is usually used to monitor progress while training.

Parameters

<i>predictors</i>	Input variables.
<i>responses</i>	Target outputs for input variables.

39.67.4.5 Evaluate() [2/4]

```
double Evaluate (
    const arma::mat & parameters )
```

Evaluate the feedforward network with the given parameters.

This function is usually called by the optimizer to train the model.

Parameters

<i>parameters</i>	Matrix model parameters.
<i>deterministic</i>	Whether or not to train or test the model. Note some layer act differently in training or testing mode.

39.67.4.6 Evaluate() [3/4]

```
double Evaluate (
    const arma::mat & parameters,
```

```

    const size_t begin,
    const size_t batchSize,
    const bool deterministic )

```

Evaluate the feedforward network with the given parameters, but using only a number of data points.

This is useful for optimizers such as SGD, which require a separable objective function.

Parameters

<i>parameters</i>	Matrix model parameters.
<i>begin</i>	Index of the starting point to use for objective function evaluation.
<i>batchSize</i>	Number of points to be passed at a time to use for objective function evaluation.
<i>deterministic</i>	Whether or not to train or test the model. Note some layer act differently in training or testing mode.

39.67.4.7 Evaluate() [4/4]

```

double Evaluate (
    const arma::mat & parameters,
    const size_t begin,
    const size_t batchSize )

```

Evaluate the feedforward network with the given parameters, but using only a number of data points.

This is useful for optimizers such as SGD, which require a separable objective function. This just calls the overload of **Evaluate()** (p. 697) with `deterministic = true`.

Parameters

<i>parameters</i>	Matrix model parameters.
<i>begin</i>	Index of the starting point to use for objective function evaluation.
<i>batchSize</i>	Number of points to be passed at a time to use for objective function evaluation.

39.67.4.8 EvaluateWithGradient() [1/2]

```

double EvaluateWithGradient (
    const arma::mat & parameters,
    GradType & gradient )

```

Evaluate the feedforward network with the given parameters.

This function is usually called by the optimizer to train the model. This just calls the overload of **EvaluateWithGradient()** (p. 698) with `batchSize = 1`.

Parameters

<i>parameters</i>	Matrix model parameters.
<i>gradient</i>	Matrix to output gradient into.

39.67.4.9 EvaluateWithGradient() [2/2]

```
double EvaluateWithGradient (
    const arma::mat & parameters,
    const size_t begin,
    GradType & gradient,
    const size_t batchSize )
```

Evaluate the feedforward network with the given parameters, but using only a number of data points.

This is useful for optimizers such as SGD, which require a separable objective function.

Parameters

<i>parameters</i>	Matrix model parameters.
<i>begin</i>	Index of the starting point to use for objective function evaluation.
<i>gradient</i>	Matrix to output gradient into.
<i>batchSize</i>	Number of points to be passed at a time to use for objective function evaluation.

39.67.4.10 Forward() [1/2]

```
void Forward (
    arma::mat inputs,
    arma::mat & results )
```

Perform the forward pass of the data in real batch mode.

Forward and Backward should be used as a pair, and they are designed mainly for advanced users. User should try to use Predict and Train unless those two functions can't satisfy some special requirements.

Parameters

<i>inputs</i>	The input data.
<i>results</i>	The predicted results.

Referenced by FFN< OutputLayerType, InitializationRuleType, CustomLayers >::Predictors().

39.67.4.11 Forward() [2/2]

```
void Forward (
    arma::mat inputs,
    arma::mat & results,
    const size_t begin,
    const size_t end )
```

Perform a partial forward pass of the data.

This function is meant for the cases when users require a forward pass only through certain layers and not the entire network.

Parameters

<i>inputs</i>	The input data for the specified first layer.
<i>results</i>	The predicted results from the specified last layer.
<i>begin</i>	The index of the first layer.
<i>end</i>	The index of the last layer.

39.67.4.12 Gradient()

```
void Gradient (
    const arma::mat & parameters,
    const size_t begin,
    arma::mat & gradient,
    const size_t batchSize )
```

Evaluate the gradient of the feedforward network with the given parameters, and with respect to only a number of points in the dataset.

This is useful for optimizers such as SGD, which require a separable objective function.

Parameters

<i>parameters</i>	Matrix of the model parameters to be optimized.
<i>begin</i>	Index of the starting point to use for objective function gradient evaluation.
<i>gradient</i>	Matrix to output gradient into.
<i>batchSize</i>	Number of points to be processed as a batch for objective function gradient evaluation.

Referenced by FFN< OutputLayerType, InitializationRuleType, CustomLayers >::Predictors().

39.67.4.13 NumFunctions()

```
size_t NumFunctions ( ) const [inline]
```

Return the number of separable functions (the number of predictor points).

Definition at line 261 of file `ffn.hpp`.

39.67.4.14 operator=()

```
FFN& operator= (
    FFN< OutputLayerType, InitializationRuleType, CustomLayers > )
```

Copy/move assignment operator.

39.67.4.15 Parameters() [1/2]

```
const arma::mat& Parameters ( ) const [inline]
```

Return the initial point for the optimization.

Definition at line 264 of file `ffn.hpp`.

39.67.4.16 Parameters() [2/2]

```
arma::mat& Parameters ( ) [inline]
```

Modify the initial point for the optimization.

Definition at line 266 of file `ffn.hpp`.

39.67.4.17 Predict()

```
void Predict (
    arma::mat predictors,
    arma::mat & results )
```

Predict the responses to a given set of predictors.

The responses will reflect the output of the given output layer as returned by the output layer function.

If you want to pass in a parameter and discard the original parameter object, be sure to use `std::move` to avoid unnecessary copy.

Parameters

<i>predictors</i>	Input predictors.
<i>results</i>	Matrix to put output predictions of responses into.

39.67.4.18 Predictors() [1/2]

```
const arma::mat& Predictors ( ) const [inline]
```

Get the matrix of data points (predictors).

Definition at line 274 of file `ffn.hpp`.

39.67.4.19 Predictors() [2/2]

```
arma::mat& Predictors ( ) [inline]
```

Modify the matrix of data points (predictors).

Definition at line 276 of file `ffn.hpp`.

References `FFN< OutputLayerType, InitializationRuleType, CustomLayers >::Backward()`, `FFN< OutputLayerType, InitializationRuleType, CustomLayers >::Forward()`, `FFN< OutputLayerType, InitializationRuleType, CustomLayers >::Gradient()`, `FFN< OutputLayerType, InitializationRuleType, CustomLayers >::ResetParameters()`, and `FFN< OutputLayerType, InitializationRuleType, CustomLayers >::serialize()`.

39.67.4.20 ResetParameters()

```
void ResetParameters ( )
```

Reset the module information (weights/parameters).

Referenced by `FFN< OutputLayerType, InitializationRuleType, CustomLayers >::Predictors()`.

39.67.4.21 Responses() [1/2]

```
const arma::mat& Responses ( ) const [inline]
```

Get the matrix of responses to the input data points.

Definition at line 269 of file `ffn.hpp`.

39.67.4.22 Responses() [2/2]

```
arma::mat& Responses ( ) [inline]
```

Modify the matrix of responses to the input data points.

Definition at line 271 of file `ffn.hpp`.

39.67.4.23 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the model.

Referenced by `FFN< OutputLayerType, InitializationRuleType, CustomLayers >::Predictors()`.

39.67.4.24 Shuffle()

```
void Shuffle ( )
```

Shuffle the order of function visitation.

This may be called by the optimizer.

39.67.4.25 Train() [1/2]

```
double Train (
    arma::mat predictors,
    arma::mat responses,
    OptimizerType & optimizer )
```

Train the feedforward network on the given input data using the given optimizer.

This will use the existing model parameters as a starting point for the optimization. If this is not what you want, then you should access the parameters vector directly with **Parameters()** (p. 701) and modify it as desired.

If you want to pass in a parameter and discard the original parameter object, be sure to use `std::move` to avoid unnecessary copy.

Template Parameters

<i>OptimizerType</i>	Type of optimizer to use to train the model.
----------------------	--

Parameters

<i>predictors</i>	Input training variables.
<i>responses</i>	Outputs results from input training variables.
<i>optimizer</i>	Instantiated optimizer used to train the model.

Returns

The final objective of the trained model (NaN or Inf on error).

39.67.4.26 Train() [2/2]

```
double Train (
    arma::mat predictors,
    arma::mat responses )
```

Train the feedforward network on the given input data.

By default, the RMSProp optimization algorithm is used, but others can be specified (such as ens::SGD).

This will use the existing model parameters as a starting point for the optimization. If this is not what you want, then you should access the parameters vector directly with **Parameters()** (p. 701) and modify it as desired.

If you want to pass in a parameter and discard the original parameter object, be sure to use std::move to avoid unnecessary copy.

Template Parameters

<i>OptimizerType</i>	Type of optimizer to use to train the model.
----------------------	--

Parameters

<i>predictors</i>	Input training variables.
<i>responses</i>	Outputs results from input training variables.

Returns

The final objective of the trained model (NaN or Inf on error).

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/ **ffn.hpp**

39.68 FFTConvolution< BorderMode, padLastDim > Class Template Reference

Computes the two-dimensional convolution through fft.

Static Public Member Functions

- template<typename eT , typename Border = BorderMode>
static std::enable_if< std::is_same< Border, **ValidConvolution** >::value, void >::type **Convolution** (const arma::Mat< eT > &input, const arma::Mat< eT > &filter, arma::Mat< eT > &output)
- template<typename eT , typename Border = BorderMode>
static std::enable_if< std::is_same< Border, **FullConvolution** >::value, void >::type **Convolution** (const arma::Mat< eT > &input, const arma::Mat< eT > &filter, arma::Mat< eT > &output)
- template<typename eT >
static void **Convolution** (const arma::Cube< eT > &input, const arma::Cube< eT > &filter, arma::Cube< eT > &output)
- template<typename eT >
static void **Convolution** (const arma::Mat< eT > &input, const arma::Cube< eT > &filter, arma::Cube< eT > &output)
- template<typename eT >
static void **Convolution** (const arma::Cube< eT > &input, const arma::Mat< eT > &filter, arma::Cube< eT > &output)

39.68.1 Detailed Description

```
template<typename BorderMode = FullConvolution, const bool padLastDim = false>
class mlpack::ann::FFTConvolution< BorderMode, padLastDim >
```

Computes the two-dimensional convolution through fft.

This class allows specification of the type of the border type. The convolution can be computed with the valid border type or the full border type (default).

FullConvolution (p. 714): returns the full two-dimensional convolution. **ValidConvolution** (p. 967): returns only those parts of the convolution that are computed without the zero-padded edges.

Template Parameters

<i>BorderMode</i>	Type of the border mode (FullConvolution (p. 714) or ValidConvolution (p. 967)).
<i>padLastDim</i>	Pad the last dimension of the input to to turn it from odd to even.

Definition at line 37 of file `fft_convolution.hpp`.

39.68.2 Member Function Documentation

39.68.2.1 Convolution() [1/5]

```
static std::enable_if< std::is_same<Border, ValidConvolution>::value, void>::type Convolution
(
    const arma::Mat< eT > & input,
    const arma::Mat< eT > & filter,
    arma::Mat< eT > & output ) [inline], [static]
```

Definition at line 54 of file `fft_convolution.hpp`.

Referenced by `SVDCConvolution< BorderMode >::Convolution()`, and `FFTConvolution< BorderMode, padLastDim >::Convolution()`.

39.68.2.2 Convolution() [2/5]

```
static std::enable_if< std::is_same<Border, FullConvolution>::value, void>::type Convolution (
    const arma::Mat< eT > & input,
    const arma::Mat< eT > & filter,
    arma::Mat< eT > & output ) [inline], [static]
```

Definition at line 89 of file `fft_convolution.hpp`.

39.68.2.3 Convolution() [3/5]

```
static void Convolution (
    const arma::Cube< eT > & input,
    const arma::Cube< eT > & filter,
    arma::Cube< eT > & output ) [inline], [static]
```

Definition at line 135 of file `fft_convolution.hpp`.

References `FFTConvolution< BorderMode, padLastDim >::Convolution()`.

39.68.2.4 Convolution() [4/5]

```
static void Convolution (
    const arma::Mat< eT > & input,
    const arma::Cube< eT > & filter,
    arma::Cube< eT > & output ) [inline], [static]
```

Definition at line 166 of file `fft_convolution.hpp`.

References `FFTConvolution< BorderMode, padLastDim >::Convolution()`.

39.68.2.5 Convolution() [5/5]

```
static void Convolution (
    const arma::Cube< eT > & input,
    const arma::Mat< eT > & filter,
    arma::Cube< eT > & output ) [inline], [static]
```

Definition at line 194 of file `fft_convolution.hpp`.

References `FFTConvolution< BorderMode, padLastDim >::Convolution()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/fft_convolution.hpp`

39.69 FlexibleReLU< InputDataType, OutputDataType > Class Template Reference

The **FlexibleReLU** (p. 707) activation function, defined by.

Public Member Functions

- **FlexibleReLU** (const double alpha=0)
*Create the **FlexibleReLU** (p. 707) object using the specified parameters.*
- double const & **Alpha** () const
Get the parameter controlling the range of the relu function.
- double & **Alpha** ()
Modify the parameter controlling the range of the relu function.
- template<typename DataType >
void **Backward** (const DataType &&input, DataType &&gy, DataType &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()

Modify the delta.

- `template<typename InputType , typename OutputType >`
`void Forward (const InputType &&input, OutputType &&output)`
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- `template<typename eT >`
`void Gradient (const arma::Mat< eT > &&input, arma::Mat< eT > &&error, arma::Mat< eT > &&gradient)`
Calculate the gradient using the output delta and the input activation.
- `OutputDataType const & Gradient () const`
Get the gradient.
- `OutputDataType & Gradient ()`
Modify the gradient.
- `OutputDataType const & OutputParameter () const`
Get the output parameter.
- `OutputDataType & OutputParameter ()`
Modify the output parameter.
- `OutputDataType const & Parameters () const`
Get the parameters.
- `OutputDataType & Parameters ()`
Modify the parameters.
- `void Reset ()`
Reset the layer parameter.
- `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`
Serialize the layer.

39.69.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::FlexibleReLU< InputDataType, OutputDataType >
```

The **FlexibleReLU** (p. 707) activation function, defined by.

$$f(x) = \max(0, x) + \alpha$$

$$f'(x) = \begin{cases} 1 & : x > 0 \\ 0 & : x \leq 0 \end{cases}$$

For more information, read the following paper:

```
@article{Qiu2018,
  author = {Suo Qiu, Xiangmin Xu and Bolun Cai},
  title = {FReLU: Flexible Rectified Linear Units for Improving
    Convolutional Neural Networks}
  journal = {arxiv preprint},
  URL = {https://arxiv.org/abs/1706.08098},
  year = {2018}
}
```

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mar, arma::sp_mat or arma::cube)
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube)

Definition at line 59 of file flexible_relu.hpp.

39.69.2 Constructor & Destructor Documentation

39.69.2.1 FlexibleReLU()

```
FlexibleReLU (  
    const double alpha = 0 )
```

Create the **FlexibleReLU** (p. 707) object using the specified parameters.

The non zero parameter can be adjusted by specifying the parameter alpha which controls the range of the relu function. (Default alpha = 0) This parameter is trainable.

Parameters

<i>alpha</i>	Parameter for adjusting the range of the relu function.
--------------	---

39.69.3 Member Function Documentation

39.69.3.1 Alpha() [1/2]

```
double const& Alpha ( ) const [inline]
```

Get the parameter controlling the range of the relu function.

Definition at line 134 of file flexible_relu.hpp.

39.69.3.2 Alpha() [2/2]

```
double& Alpha ( ) [inline]
```

Modify the parameter controlling the range of the relu function.

Definition at line 136 of file flexible_relu.hpp.

References FlexibleReLU< InputDataType, OutputDataType >::serialize().

39.69.3.3 Backward()

```
void Backward (
    const DataType && input,
    DataType && gy,
    DataType && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.69.3.4 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 124 of file flexible_relu.hpp.

39.69.3.5 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 126 of file flexible_relu.hpp.

39.69.3.6 Forward()

```
void Forward (
    const InputType && input,
    OutputType && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.69.3.7 Gradient() [1/3]

```
void Gradient (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && error,
    arma::Mat< eT > && gradient )
```

Calculate the gradient using the output delta and the input activation.

Parameters

<i>input</i>	The input parameter used for calculating the gradient.
<i>error</i>	The calculated error.
<i>gradient</i>	The calculated gradient.

39.69.3.8 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 129 of file flexible_relu.hpp.

39.69.3.9 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 131 of file flexible_relu.hpp.

39.69.3.10 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 119 of file flexible_relu.hpp.

39.69.3.11 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 121 of file flexible_relu.hpp.

39.69.3.12 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 114 of file flexible_relu.hpp.

39.69.3.13 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 116 of file flexible_relu.hpp.

39.69.3.14 Reset()

```
void Reset ( )
```

Reset the layer parameter.

39.69.3.15 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by FlexibleReLU< InputDataType, OutputDataType >::Alpha().

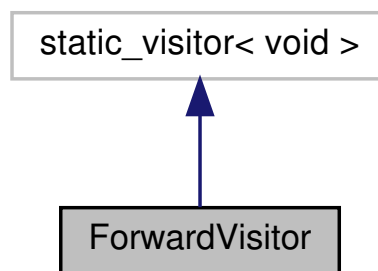
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **flexible_relu.hpp**

39.70 ForwardVisitor Class Reference

ForwardVisitor (p. 713) executes the Forward() function given the input and output parameter.

Inheritance diagram for ForwardVisitor:

**Public Member Functions**

- **ForwardVisitor** (arma::mat &&input, arma::mat &&output)
Execute the Foward() function given the input and output parameter.
- template<typename LayerType >
void **operator()** (LayerType *layer) const
Execute the Foward() function.

39.70.1 Detailed Description

ForwardVisitor (p. 713) executes the Forward() function given the input and output parameter.

Definition at line 28 of file forward_visitor.hpp.

39.70.2 Constructor & Destructor Documentation

39.70.2.1 ForwardVisitor()

```
ForwardVisitor (  
    arma::mat && input,  
    arma::mat && output )
```

Execute the Foward() function given the input and output parameter.

39.70.3 Member Function Documentation

39.70.3.1 operator()()

```
void operator() (  
    LayerType * layer ) const
```

Execute the Foward() function.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **forward_visitor.hpp**

39.71 FullConvolution Class Reference

39.71.1 Detailed Description

Definition at line 22 of file border_modes.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/ **border_modes.hpp**

39.72 GaussianInitialization Class Reference

This class is used to initialize weight matrix with a gaussian.

Public Member Functions

- **GaussianInitialization** (const double mean=0, const double variance=1)
Initialize the gaussian with the given mean and variance.
- template<typename eT >
void **Initialize** (arma::Mat< eT > &W, const size_t rows, const size_t cols)
Initialize the elements weight matrix using a Gaussian Distribution.
- template<typename eT >
void **Initialize** (arma::Cube< eT > &W, const size_t rows, const size_t cols, const size_t slices)
Initialize randomly the elements of the specified weight 3rd order tensor.

39.72.1 Detailed Description

This class is used to initialize weight matrix with a gaussian.

Definition at line 28 of file gaussian_init.hpp.

39.72.2 Constructor & Destructor Documentation

39.72.2.1 GaussianInitialization()

```
GaussianInitialization (  
    const double mean = 0,  
    const double variance = 1 ) [inline]
```

Initialize the gaussian with the given mean and variance.

Parameters

<i>mean</i>	Mean of the gaussian.
<i>variance</i>	Variance of the gaussian.

Definition at line 37 of file gaussian_init.hpp.

39.72.3 Member Function Documentation

39.72.3.1 Initialize() [1/2]

```
void Initialize (
    arma::Mat< eT > & W,
    const size_t rows,
    const size_t cols ) [inline]
```

Initialize the elements weight matrix using a Gaussian Distribution.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.

Definition at line 51 of file gaussian_init.hpp.

References `mlpack::math::RandNormal()`.

Referenced by `GlorotInitializationType< Uniform >::Initialize()`.

39.72.3.2 Initialize() [2/2]

```
void Initialize (
    arma::Cube< eT > & W,
    const size_t rows,
    const size_t cols,
    const size_t slices ) [inline]
```

Initialize randomly the elements of the specified weight 3rd order tensor.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.
<i>slice</i>	Numbers of slices.

Definition at line 71 of file gaussian_init.hpp.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ gaussian_init.hpp`

39.73 Glimpse< InputDataType, OutputDataType > Class Template Reference

The glimpse layer returns a retina-like representation (down-scaled cropped images) of increasing scale around a given location in a given image.

Public Member Functions

- **Glimpse** (const size_t inSize=0, const size_t size=0, const size_t depth=3, const size_t scale=2, const size_t inputWidth=0, const size_t inputHeight=0)
Create the GlimpseLayer object using the specified ratio and rescale parameter.
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of the glimpse layer.
- OutputDataType & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- bool **Deterministic** () const
Get the value of the deterministic parameter.
- bool & **Deterministic** ()
Modify the value of the deterministic parameter.
- template<typename eT >
void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Ordinary feed forward pass of the glimpse layer.
- size_t const & **InputHeight** () const
Get the input height.
- size_t & **InputHeight** ()
Modify the input height.
- size_t const & **InputWidth** () const
Get the input width.
- size_t & **InputWidth** ()
Modify input the width.
- void **Location** (const arma::mat &location)
Set the location the x and y coordinate of the center of the output glimpse.
- size_t const & **OutputHeight** () const
Get the output height.
- size_t & **OutputHeight** ()
Modify the output height.
- OutputDataType & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- size_t const & **OutputWidth** () const
Get the output width.
- size_t & **OutputWidth** ()
Modify the output width.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.73.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::Glimpse< InputDataType, OutputDataType >
```

The glimpse layer returns a retina-like representation (down-scaled cropped images) of increasing scale around a given location in a given image.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 87 of file glimpse.hpp.

39.73.2 Constructor & Destructor Documentation

39.73.2.1 Glimpse()

```
Glimpse (
    const size_t inSize = 0,
    const size_t size = 0,
    const size_t depth = 3,
    const size_t scale = 2,
    const size_t inputWidth = 0,
    const size_t inputHeight = 0 )
```

Create the GlimpseLayer object using the specified ratio and rescale parameter.

Parameters

<i>inSize</i>	The size of the input units.
<i>size</i>	The used glimpse size (height = width).
<i>depth</i>	The number of patches to crop per glimpse.
<i>scale</i>	The scaling factor used to create the increasing retina-like representation.
<i>inputWidth</i>	The input width of the given input data.
<i>inputHeight</i>	The input height of the given input data.

39.73.3 Member Function Documentation

39.73.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of the glimpse layer.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.73.3.2 Delta() [1/2]

```
OutputDataType& Delta ( ) const [inline]
```

Get the delta.

Definition at line 136 of file glimpse.hpp.

39.73.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 138 of file glimpse.hpp.

39.73.3.4 Deterministic() [1/2]

```
bool Deterministic ( ) const [inline]
```

Get the value of the deterministic parameter.

Definition at line 168 of file glimpse.hpp.

39.73.3.5 Deterministic() [2/2]

```
bool& Deterministic ( ) [inline]
```

Modify the value of the deterministic parameter.

Definition at line 170 of file glimpse.hpp.

References `MeanPoolingRule::Pooling()`, and `MeanPoolingRule::Unpooling()`.

39.73.3.6 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of the glimpse layer.

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.73.3.7 InputHeight() [1/2]

```
size_t const& InputHeight ( ) const [inline]
```

Get the input height.

Definition at line 153 of file glimpse.hpp.

39.73.3.8 InputHeight() [2/2]

```
size_t& InputHeight ( ) [inline]
```

Modify the input height.

Definition at line 155 of file glimpse.hpp.

39.73.3.9 InputWidth() [1/2]

```
size_t const& InputWidth ( ) const [inline]
```

Get the input width.

Definition at line 148 of file glimpse.hpp.

39.73.3.10 InputWidth() [2/2]

```
size_t& InputWidth ( ) [inline]
```

Modify input the width.

Definition at line 150 of file glimpse.hpp.

39.73.3.11 Location()

```
void Location (
    const arma::mat & location ) [inline]
```

Set the locationthe x and y coordinate of the center of the output glimpse.

Definition at line 142 of file glimpse.hpp.

39.73.3.12 OutputHeight() [1/2]

```
size_t const& OutputHeight ( ) const [inline]
```

Get the output height.

Definition at line 163 of file glimpse.hpp.

39.73.3.13 OutputHeight() [2/2]

```
size_t& OutputHeight ( ) [inline]
```

Modify the output height.

Definition at line 165 of file glimpse.hpp.

39.73.3.14 OutputParameter() [1/2]

```
OutputDataType& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 131 of file `glimpse.hpp`.

39.73.3.15 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 133 of file `glimpse.hpp`.

39.73.3.16 OutputWidth() [1/2]

```
size_t const& OutputWidth ( ) const [inline]
```

Get the output width.

Definition at line 158 of file `glimpse.hpp`.

39.73.3.17 OutputWidth() [2/2]

```
size_t& OutputWidth ( ) [inline]
```

Modify the output width.

Definition at line 160 of file `glimpse.hpp`.

39.73.3.18 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ glimpse.hpp`

39.74 GlorotInitializationType< Uniform > Class Template Reference

This class is used to initialize the weight matrix with the Glorot Initialization method.

Public Member Functions

- **GlorotInitializationType** ()
Initialize the Glorot initialization object.
- template<typename eT >
void **Initialize** (arma::Mat< eT > &W, const size_t rows, const size_t cols)
Initialize the elements weight matrix with glorot initialization method.
- template<typename eT >
void **Initialize** (arma::Cube< eT > &W, const size_t rows, const size_t cols, const size_t slices)
Initialize the elements of the specified weight 3rd order tensor with glorot initialization method.
- template<>
void **Initialize** (arma::Mat< eT > &W, const size_t rows, const size_t cols)
- template<>
void **Initialize** (arma::Mat< eT > &W, const size_t rows, const size_t cols)

39.74.1 Detailed Description

```
template<bool Uniform = true>
class mlpack::ann::GlorotInitializationType< Uniform >
```

This class is used to initialize the weight matrix with the Glorot Initialization method.

The method is defined by

$$\text{Var}[w_i] = \frac{2}{n_i + n_{i+1}}$$

$$w_i \sim \text{U}\left[-\frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}, \frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}\right]$$

where n_{i+1} is the number of neurons in the outgoing layer, n_i represents the number of neurons in the ingoing layer. Here Normal Distribution may also be used if needed

For more information, see the following paper.

```
@inproceedings {pmlr-v9-glorot10a,
  title      = {Understanding the difficulty of training
               deep feedforward neural networks},
  author     = {Xavier Glorot and Yoshua Bengio},
  booktitle  = {Proceedings of the Thirteenth International Conference
               on Artificial Intelligence and Statistics},
  year      = {2010}
}
```

Definition at line 55 of file glorot_init.hpp.

39.74.2 Constructor & Destructor Documentation

39.74.2.1 GlorotInitializationType()

```
GlorotInitializationType ( ) [inline]
```

Initialize the Glorot initialization object.

Definition at line 61 of file `glorot_init.hpp`.

39.74.3 Member Function Documentation

39.74.3.1 Initialize() [1/4]

```
void Initialize (
    arma::Mat< eT > & W,
    const size_t rows,
    const size_t cols )
```

Initialize the elements weight matrix with glorot initialization method.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.

39.74.3.2 Initialize() [2/4]

```
void Initialize (
    arma::Cube< eT > & W,
    const size_t rows,
    const size_t cols,
    const size_t slices ) [inline]
```

Initialize the elements of the specified weight 3rd order tensor with glorot initialization method.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.
<i>slice</i>	Numbers of slices.

Definition at line 125 of file `glorot_init.hpp`.

39.74.3.3 Initialize() [3/4]

```
void Initialize (
    arma::Mat< eT > & W,
    const size_t rows,
    const size_t cols ) [inline]
```

Definition at line 96 of file `glorot_init.hpp`.

References `GaussianInitialization::Initialize()`.

39.74.3.4 Initialize() [4/4]

```
void Initialize (
    arma::Mat< eT > & W,
    const size_t rows,
    const size_t cols ) [inline]
```

Definition at line 110 of file `glorot_init.hpp`.

References `RandomInitialization::Initialize()`.

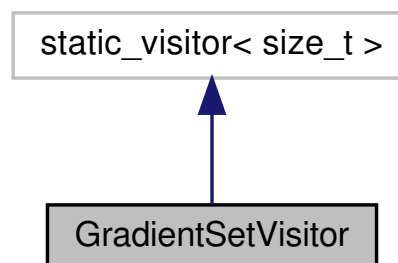
The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ glorot_init.hpp`

39.75 GradientSetVisitor Class Reference

GradientSetVisitor (p. 725) update the gradient parameter given the gradient set.

Inheritance diagram for GradientSetVisitor:



Public Member Functions

- **GradientSetVisitor** (arma::mat &&gradient, size_t offset=0)
Update the gradient parameter given the gradient set.
- template<typename LayerType >
size_t **operator()** (LayerType *layer) const
Update the gradient parameter.

39.75.1 Detailed Description

GradientSetVisitor (p. 725) update the gradient parameter given the gradient set.

Definition at line 26 of file gradient_set_visitor.hpp.

39.75.2 Constructor & Destructor Documentation

39.75.2.1 GradientSetVisitor()

```
GradientSetVisitor (
    arma::mat && gradient,
    size_t offset = 0 )
```

Update the gradient parameter given the gradient set.

39.75.3 Member Function Documentation

39.75.3.1 operator()

```
size_t operator() (
    LayerType * layer ) const
```

Update the gradient parameter.

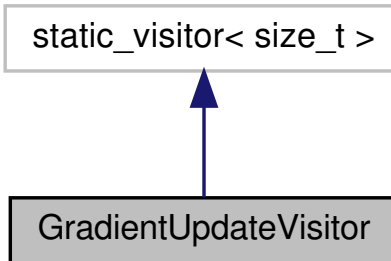
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **gradient_set_visitor.hpp**

39.76 GradientUpdateVisitor Class Reference

GradientUpdateVisitor (p. 727) update the gradient parameter given the gradient set.

Inheritance diagram for GradientUpdateVisitor:



Public Member Functions

- **GradientUpdateVisitor** (arma::mat &&gradient, size_t offset=0)
Update the gradient parameter given the gradient set.
- template<typename LayerType >
 size_t **operator()** (LayerType *layer) const
Update the gradient parameter.

39.76.1 Detailed Description

GradientUpdateVisitor (p. 727) update the gradient parameter given the gradient set.

Definition at line 26 of file gradient_update_visitor.hpp.

39.76.2 Constructor & Destructor Documentation

39.76.2.1 GradientUpdateVisitor()

```

GradientUpdateVisitor (
    arma::mat && gradient,
    size_t offset = 0 )
  
```

Update the gradient parameter given the gradient set.

39.76.3 Member Function Documentation

39.76.3.1 operator()

```
size_t operator() (
    LayerType * layer ) const
```

Update the gradient parameter.

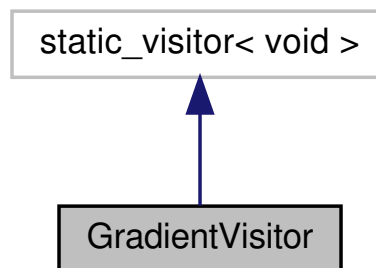
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **gradient_update_visitor.hpp**

39.77 GradientVisitor Class Reference

SearchModeVisitor executes the Gradient() method of the given module using the input and delta parameter.

Inheritance diagram for GradientVisitor:



Public Member Functions

- **GradientVisitor** (arma::mat &&input, arma::mat &&delta)
Executes the Gradient() method of the given module using the input and delta parameter.
- **GradientVisitor** (arma::mat &&input, arma::mat &&delta, const size_t index)
Executes the Gradient() method for the layer with the specified index.
- template<typename LayerType >
void **operator()** (LayerType *layer) const
Executes the Gradient() method.

39.77.1 Detailed Description

SearchModeVisitor executes the Gradient() method of the given module using the input and delta parameter.

Definition at line 28 of file gradient_visitor.hpp.

39.77.2 Constructor & Destructor Documentation

39.77.2.1 GradientVisitor() [1/2]

```
GradientVisitor (  
    arma::mat && input,  
    arma::mat && delta )
```

Executes the Gradient() method of the given module using the input and delta parameter.

39.77.2.2 GradientVisitor() [2/2]

```
GradientVisitor (  
    arma::mat && input,  
    arma::mat && delta,  
    const size_t index )
```

Executes the Gradient() method for the layer with the specified index.

39.77.3 Member Function Documentation

39.77.3.1 operator()()

```
void operator() (  
    LayerType * layer ) const
```

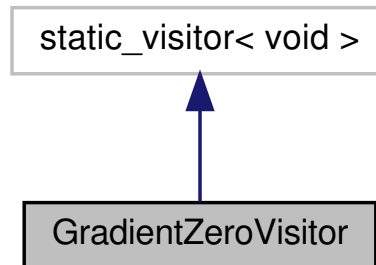
Executes the Gradient() method.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **gradient_visitor.hpp**

39.78 GradientZeroVisitor Class Reference

Inheritance diagram for GradientZeroVisitor:



Public Member Functions

- **GradientZeroVisitor ()**
Set the gradient to zero for the given module.
- template<typename LayerType >
void **operator()** (LayerType *layer) const
Set the gradient to zero.

39.78.1 Detailed Description

Definition at line 27 of file gradient_zero_visitor.hpp.

39.78.2 Constructor & Destructor Documentation

39.78.2.1 GradientZeroVisitor()

```
GradientZeroVisitor ( )
```

Set the gradient to zero for the given module.

39.78.3 Member Function Documentation

39.78.3.1 operator>()

```
void operator() (
    LayerType * layer ) const
```

Set the gradient to zero.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **gradient_zero_visitor.hpp**

39.79 GRU< InputDataType, OutputDataType > Class Template Reference

An implementation of a gru network layer.

Public Member Functions

- **GRU** ()
*Create the **GRU** (p. 731) object.*
- **GRU** (const size_t inSize, const size_t outSize, const size_t rho=std::numeric_limits< size_t >::max())
*Create the **GRU** (p. 731) layer object using the specified parameters.*
- **~GRU** ()
*Delete the **GRU** (p. 731) and the layers it holds.*
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, calculating the function f(x) by propagating x backwards trough f.
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- bool **Deterministic** () const
The value of the deterministic parameter.
- bool & **Deterministic** ()
Modify the value of the deterministic parameter.
- template<typename eT >
void **Forward** (arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Ordinary feed forward pass of a neural network, evaluating the function f(x) by propagating the activity forward through f.
- template<typename eT >
void **Gradient** (arma::Mat< eT > &&input, arma::Mat< eT > &&, arma::Mat< eT > &&)
Ordinary feed forward pass of a neural network, evaluating the function f(x) by propagating the activity forward through f.
- OutputDataType const & **Gradient** () const
Get the gradient.
- OutputDataType & **Gradient** ()
Modify the gradient.
- std::vector< **LayerTypes**<> > & **Model** ()
Get the model modules.

- OutputDataType const & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- OutputDataType const & **Parameters** () const
Get the parameters.
- OutputDataType & **Parameters** ()
Modify the parameters.
- void **ResetCell** (const size_t size)
- size_t **Rho** () const
Get the maximum number of steps to backpropagate through time (BPTT).
- size_t & **Rho** ()
Modify the maximum number of steps to backpropagate through time (BPTT).
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.79.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::GRU< InputDataType, OutputDataType >
```

An implementation of a gru network layer.

This cell can be used in **RNN** (p. 911) networks.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 57 of file gru.hpp.

39.79.2 Constructor & Destructor Documentation

39.79.2.1 GRU() [1/2]

```
GRU ( )
```

Create the **GRU** (p. 731) object.

39.79.2.2 GRU() [2/2]

```
GRU (
    const size_t inSize,
    const size_t outSize,
    const size_t rho = std::numeric_limits< size_t >::max() )
```

Create the **GRU** (p. 731) layer object using the specified parameters.

Parameters

<i>inSize</i>	The number of input units.
<i>outSize</i>	The number of output units.
<i>rho</i>	Maximum number of steps to backpropagate through time (BPTT).

39.79.2.3 ~GRU()

```
~ GRU ( )
```

Delete the **GRU** (p. 731) and the layers it holds.

39.79.3 Member Function Documentation

39.79.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.79.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 144 of file gru.hpp.

39.79.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 146 of file gru.hpp.

39.79.3.4 Deterministic() [1/2]

```
bool Deterministic ( ) const [inline]
```

The value of the deterministic parameter.

Definition at line 124 of file gru.hpp.

39.79.3.5 Deterministic() [2/2]

```
bool& Deterministic ( ) [inline]
```

Modify the value of the deterministic parameter.

Definition at line 126 of file gru.hpp.

39.79.3.6 Forward()

```
void Forward (
    arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.79.3.7 Gradient() [1/3]

```
void Gradient (
    arma::Mat< eT > && input,
    arma::Mat< eT > && ,
    arma::Mat< eT > && )
```

39.79.3.8 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 149 of file gru.hpp.

39.79.3.9 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 151 of file gru.hpp.

39.79.3.10 Model()

```
std::vector< LayerTypes<> >& Model ( ) [inline]
```

Get the model modules.

Definition at line 154 of file gru.hpp.

References GRU< InputDataType, OutputDataType >::serialize().

39.79.3.11 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 139 of file gru.hpp.

39.79.3.12 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 141 of file gru.hpp.

39.79.3.13 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 134 of file gru.hpp.

39.79.3.14 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 136 of file gru.hpp.

39.79.3.15 ResetCell()

```
void ResetCell (
    const size_t size )
```


39.79.3.16 Rho() [1/2]

```
size_t Rho ( ) const [inline]
```

Get the maximum number of steps to backpropagate through time (BPTT).

Definition at line 129 of file gru.hpp.

39.79.3.17 Rho() [2/2]

```
size_t& Rho ( ) [inline]
```

Modify the maximum number of steps to backpropagate through time (BPTT).

Definition at line 131 of file gru.hpp.

39.79.3.18 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by GRU< InputDataType, OutputDataType >::Model().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **gru.hpp**

39.80 HardSigmoidFunction Class Reference

The hard sigmoid function, defined by.

Static Public Member Functions

- static double **Deriv** (const double y)
Computes the first derivatives of hard sigmoid function.
- template<typename InputVecType , typename OutputVecType >
static void **Deriv** (const InputVecType &y, OutputVecType &x)
Computes the first derivatives of the hard sigmoid function.
- static double **Fn** (const double x)
Computes the hard sigmoid function.
- template<typename InputVecType , typename OutputVecType >
static void **Fn** (const InputVecType &x, OutputVecType &y)
Computes the hard sigmoid function.

39.80.1 Detailed Description

The hard sigmoid function, defined by.

$$\begin{aligned} f(x) &= \min(1, \max(0, 0.2 * x + 0.5)) \\ f'(x) &= \begin{cases} 0.0 & : x = 0, 1 \\ 0.2 & \end{cases} \end{aligned}$$

Definition at line 34 of file `hard_sigmoid_function.hpp`.

39.80.2 Member Function Documentation

39.80.2.1 `Deriv()` [1/2]

```
static double Deriv (  
    const double y ) [inline], [static]
```

Computes the first derivatives of hard sigmoid function.

Parameters

<i>y</i>	Input data.
----------	-------------

Returns

$f'(x)$

Definition at line 69 of file `hard_sigmoid_function.hpp`.

Referenced by `HardSigmoidFunction::Deriv()`.

39.80.2.2 `Deriv()` [2/2]

```
static void Deriv (  
    const InputVecType & y,  
    OutputVecType & x ) [inline], [static]
```

Computes the first derivatives of the hard sigmoid function.

Parameters

<i>y</i>	Input activations.
<i>x</i>	The resulting derivatives.

Definition at line 85 of file `hard_sigmoid_function.hpp`.

References `HardSigmoidFunction::Deriv()`.

39.80.2.3 Fn() [1/2]

```
static double Fn (  
    const double x ) [inline], [static]
```

Computes the hard sigmoid function.

Parameters

<i>x</i>	Input data.
----------	-------------

Returns

$f(x)$.

Definition at line 43 of file `hard_sigmoid_function.hpp`.

Referenced by `HardSigmoidFunction::Fn()`.

39.80.2.4 Fn() [2/2]

```
static void Fn (  
    const InputVecType & x,  
    OutputVecType & y ) [inline], [static]
```

Computes the hard sigmoid function.

Parameters

<i>x</i>	Input data.
<i>y</i>	The resulting output activations.

Definition at line 55 of file `hard_sigmoid_function.hpp`.

References `HardSigmoidFunction::Fn()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/ hard_sigmoid_function.↵
hpp`

39.81 `HardTanH< InputDataType, OutputDataType >` Class Template Reference

The Hard Tanh activation function, defined by.

Public Member Functions

- **HardTanH** (const double maxValue=1, const double minValue=-1)
*Create the **HardTanH** (p. 740) object using the specified parameters.*
- `template<typename DataType >`
`void Backward (const DataType &&input, DataType &&gy, DataType &&g)`
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- `OutputDataType const & Delta () const`
Get the delta.
- `OutputDataType & Delta ()`
Modify the delta.
- `template<typename InputType, typename OutputType >`
`void Forward (const InputType &&input, OutputType &&output)`
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- `double const & MaxValue () const`
Get the maximum value.
- `double & MaxValue ()`
Modify the maximum value.
- `double const & MinValue () const`
Get the minimum value.
- `double & MinValue ()`
Modify the minimum value.
- `OutputDataType const & OutputParameter () const`
Get the output parameter.
- `OutputDataType & OutputParameter ()`
Modify the output parameter.
- `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`
Serialize the layer.

39.81.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::HardTanH< InputDataType, OutputDataType >
```

The Hard Tanh activation function, defined by.

$$f(x) = \begin{cases} \max & : x > \maxValue \\ \min & : x \leq \minValue \\ x & : otherwise \end{cases}$$

$$f'(x) = \begin{cases} 0 & : x > \maxValue \\ 0 & : x \leq \minValue \\ 1 & : otherwise \end{cases}$$

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 49 of file hard_tanh.hpp.

39.81.2 Constructor & Destructor Documentation

39.81.2.1 HardTanH()

```
HardTanH (
    const double maxValue = 1,
    const double minValue = -1 )
```

Create the **HardTanH** (p. 740) object using the specified parameters.

The range of the linear region can be adjusted by specifying the `maxValue` and `minValue`. Default (`maxValue = 1`, `minValue = -1`).

Parameters

<i>maxValue</i>	Range of the linear region maximum value.
<i>minValue</i>	Range of the linear region minimum value.

39.81.3 Member Function Documentation

39.81.3.1 Backward()

```
void Backward (
    const DataType && input,
    DataType && gy,
    DataType && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.81.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 92 of file `hard_tanh.hpp`.

39.81.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 94 of file `hard_tanh.hpp`.

39.81.3.4 Forward()

```
void Forward (
    const InputType && input,
    OutputType && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.81.3.5 MaxValue() [1/2]

```
double const& MaxValue ( ) const [inline]
```

Get the maximum value.

Definition at line 97 of file hard_tanh.hpp.

39.81.3.6 MaxValue() [2/2]

```
double& MaxValue ( ) [inline]
```

Modify the maximum value.

Definition at line 99 of file hard_tanh.hpp.

39.81.3.7 MinValue() [1/2]

```
double const& MinValue ( ) const [inline]
```

Get the minimum value.

Definition at line 102 of file hard_tanh.hpp.

39.81.3.8 MinValue() [2/2]

```
double& MinValue ( ) [inline]
```

Modify the minimum value.

Definition at line 104 of file hard_tanh.hpp.

References HardTanH< InputDataType, OutputDataType >::serialize().

39.81.3.9 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 87 of file `hard_tanh.hpp`.

39.81.3.10 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 89 of file `hard_tanh.hpp`.

39.81.3.11 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by `HardTanH< InputDataType, OutputDataType >::MinValue()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ hard_tanh.hpp`

39.82 HeInitialization Class Reference

This class is used to initialize weight matrix with the He initialization rule given by He et.

Public Member Functions

- **HeInitialization ()**
*Initialize the **HeInitialization** (p. 744) object.*
- void **Initialize** (arma::mat &W, const size_t rows, const size_t cols)
Initialize the elements of the weight matrix with the He initialization rule.
- void **Initialize** (arma::cube &W, const size_t rows, const size_t cols, const size_t slices)
Initialize the elements of the specified weight 3rd order tensor with He initialization rule.

39.82.1 Detailed Description

This class is used to initialize weight matrix with the He initialization rule given by He et.

al. for neural networks. The He initialization initializes weights of the neural network to better suit the rectified activation units.

For more information, the following paper can be referred to:

```
@article{Delving2015,
  title   = {Delving Deep into Rectifiers: Surpassing Human-Level Performance
             on ImageNet Classification},
  author  = {Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun},
  journal = {2015 IEEE International Conference on Computer Vision (ICCV)},
  year    = {2015},
  pages   = {1026-1034}
}
```

Definition at line 45 of file he_init.hpp.

39.82.2 Constructor & Destructor Documentation

39.82.2.1 HeInitialization()

```
HeInitialization ( ) [inline]
```

Initialize the **HeInitialization** (p. 744) object.

Definition at line 51 of file he_init.hpp.

39.82.3 Member Function Documentation

39.82.3.1 Initialize() [1/2]

```
void Initialize (
    arma::mat & W,
    const size_t rows,
    const size_t cols ) [inline]
```

Initialize the elements of the weight matrix with the He initialization rule.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.

Definition at line 64 of file `he_init.hpp`.

Referenced by `HeInitialization::Initialize()`.

39.82.3.2 Initialize() [2/2]

```
void Initialize (
    arma::cube & W,
    const size_t rows,
    const size_t cols,
    const size_t slices ) [inline]
```

Initialize the elements of the specified weight 3rd order tensor with He initialization rule.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.
<i>slice</i>	Numbers of slices.

Definition at line 90 of file `he_init.hpp`.

References `HeInitialization::Initialize()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ he_init.hpp`

39.83 IdentityFunction Class Reference

The identity function, defined by.

Static Public Member Functions

- static double **Deriv** (const double)
Computes the first derivative of the identity function.
- template<typename InputVecType , typename OutputVecType >
static void **Deriv** (const InputVecType &y, OutputVecType &x)
Computes the first derivatives of the identity function.
- template<typename eT >
static void **Deriv** (const arma::Cube< eT > &y, arma::Cube< eT > &x)
Computes the first derivatives of the identity function using a 3rd order tensor as input.
- static double **Fn** (const double x)
Computes the identity function.
- template<typename InputVecType , typename OutputVecType >
static void **Fn** (const InputVecType &x, OutputVecType &y)
Computes the identity function.

39.83.1 Detailed Description

The identity function, defined by.

$$\begin{aligned} f(x) &= x \\ f'(x) &= 1 \end{aligned}$$

Definition at line 28 of file identity_function.hpp.

39.83.2 Member Function Documentation

39.83.2.1 Deriv() [1/3]

```
static double Deriv (
    const double ) [inline], [static]
```

Computes the first derivative of the identity function.

Parameters

x	Input data.
---	-------------

Returns $f'(x)$

Definition at line 60 of file identity_function.hpp.

39.83.2.2 Deriv() [2/3]

```
static void Deriv (
    const InputVecType & y,
    OutputVecType & x ) [inline], [static]
```

Computes the first derivatives of the identity function.

Parameters

<i>y</i>	Input activations.
<i>x</i>	The resulting derivatives.

Definition at line 72 of file identity_function.hpp.

39.83.2.3 Deriv() [3/3]

```
static void Deriv (
    const arma::Cube< eT > & y,
    arma::Cube< eT > & x ) [inline], [static]
```

Computes the first derivatives of the identity function using a 3rd order tensor as input.

Parameters

<i>y</i>	Input activations.
<i>x</i>	The resulting derivatives.

Definition at line 85 of file identity_function.hpp.

39.83.2.4 Fn() [1/2]

```
static double Fn (
    const double x ) [inline], [static]
```

Computes the identity function.

Parameters

<i>x</i>	Input data.
----------	-------------

Returns

$f(x)$.

Definition at line 37 of file `identity_function.hpp`.

39.83.2.5 Fn() [2/2]

```
static void Fn (
    const InputVecType & x,
    OutputVecType & y ) [inline], [static]
```

Computes the identity function.

Parameters

<i>x</i>	Input data.
<i>y</i>	The resulting output activation.

Definition at line 49 of file `identity_function.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/ identity_function.hpp`

39.84 InitTraits< InitRuleType > Class Template Reference

This is a template class that can provide information about various initialization methods.

Static Public Attributes

- static const bool **UseLayer** = true

This is true if the initialization method is used for a single layer.

39.84.1 Detailed Description

```
template<typename InitRuleType>
class mlpack::ann::InitTraits< InitRuleType >
```

This is a template class that can provide information about various initialization methods.

By default, this class will provide the weakest possible assumptions on the initialization method, and each initialization method should override values as necessary. If a initialization method doesn't need to override a value, then there's no need to write a **InitTraits** (p. 750) specialization for that class.

Definition at line 28 of file `init_rules_traits.hpp`.

39.84.2 Member Data Documentation

39.84.2.1 UseLayer

```
const bool UseLayer = true [static]
```

This is true if the initialization method is used for a single layer.

Definition at line 34 of file `init_rules_traits.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ init_rules_traits.hpp`

39.85 InitTraits< KathirvalavakumarSubavathInitialization > Class Template Reference

Initialization traits of the kathirvalavakumar subavath initialization rule.

Static Public Attributes

- static const bool **UseLayer** = false
The kathirvalavakumar subavath initialization rule is applied over the entire network.

39.85.1 Detailed Description

```
template<>
class mlpack::ann::InitTraits< KathirvalavakumarSubavathInitialization >
```

Initialization traits of the kathirvalavakumar subavath initialization rule.

Definition at line 123 of file `kathirvalavakumar_subavathi_init.hpp`.

39.85.2 Member Data Documentation

39.85.2.1 UseLayer

```
const bool UseLayer = false [static]
```

The kathirvalavakumar subavath initialization rule is applied over the entire network.

Definition at line 128 of file kathirvalavakumar_subavathi_init.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ **kathirvalavakumar_subavathi_↔
init.hpp**

39.86 InitTraits< NguyenWidrowInitialization > Class Template Reference

Initialization traits of the Nguyen-Widrow initialization rule.

Static Public Attributes

- static const bool **UseLayer** = false
The Nguyen-Widrow initialization rule is applied over the entire network.

39.86.1 Detailed Description

```
template<>
class mlpack::ann::InitTraits< NguyenWidrowInitialization >
```

Initialization traits of the Nguyen-Widrow initialization rule.

Definition at line 116 of file nguyen_widrow_init.hpp.

39.86.2 Member Data Documentation

39.86.2.1 UseLayer

```
const bool UseLayer = false [static]
```

The Nguyen-Widrow initialization rule is applied over the entire network.

Definition at line 120 of file `nguyen_widrow_init.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ nguyen_widrow_init.hpp`

39.87 Join< InputDataType, OutputDataType > Class Template Reference

Implementation of the **Join** (p. 753) module class.

Public Member Functions

- **Join** ()
*Create the **Join** (p. 753) object.*
- `template<typename eT >`
`void Backward (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)`
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- `OutputDataType const & Delta () const`
Get the delta.
- `OutputDataType & Delta ()`
Modify the delta.
- `template<typename InputType, typename OutputType >`
`void Forward (const InputType &&input, OutputType &&output)`
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- `OutputDataType const & OutputParameter () const`
Get the output parameter.
- `OutputDataType & OutputParameter ()`
Modify the output parameter.
- `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`
Serialize the layer.

39.87.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::Join< InputDataType, OutputDataType >
```

Implementation of the **Join** (p. 753) module class.

The **Join** (p. 753) class accumulates the output of various modules.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 33 of file join.hpp.

39.87.2 Constructor & Destructor Documentation

39.87.2.1 Join()

```
Join ( )
```

Create the **Join** (p. 753) object.

39.87.3 Member Function Documentation

39.87.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.87.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 69 of file join.hpp.

39.87.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 71 of file join.hpp.

References Join< InputDataType, OutputDataType >::serialize().

39.87.3.4 Forward()

```
void Forward (
    const InputType && input,
    OutputType && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.87.3.5 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 64 of file join.hpp.

39.87.3.6 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 66 of file join.hpp.

39.87.3.7 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by Join< InputDataType, OutputDataType >::Delta().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **join.hpp**

39.88 KathirvalavakumarSubavathilInitialization Class Reference

This class is used to initialize the weight matrix with the method proposed by T.

Public Member Functions

- template<typename eT >
KathirvalavakumarSubavathilInitialization (const arma::Mat< eT > &data, const double s)
Initialize the random initialization rule with the given values.
- template<typename eT >
void **Initialize** (arma::Mat< eT > &W, const size_t rows, const size_t cols)
Initialize the elements of the specified weight matrix with the Kathirvalavakumar-Subavathi method.
- template<typename eT >
void **Initialize** (arma::Cube< eT > &W, const size_t rows, const size_t cols, const size_t slices)
Initialize the elements of the specified weight 3rd order tensor with the Kathirvalavakumar-Subavathi method.

39.88.1 Detailed Description

This class is used to initialize the weight matrix with the method proposed by T.

Kathirvalavakumar and S. Subavathi. The method is based on sensitivity analysis using using cauchy's inequality. The method is defined by

$$\begin{aligned}\bar{s} &= f^{-1}(t) \\ \Theta_p^1 &\leq \bar{s} \sqrt{\frac{3}{I \sum_{i=1}^I (x_{ip}^2)}} \\ \Theta^1 &= \min(\Theta_p^1); p = 1, 2, \dots, P \\ -\Theta^1 &\leq w_i^1 \leq \Theta^1\end{aligned}$$

where I is the number of inputs including the bias, p refers the pattern considered in training, f is the transfer function and $\{s\}$ is the active region in which the derivative of the activation function is greater than 4% of the maximum derivatives.

Definition at line 60 of file kathirvalavakumar_subavathi_init.hpp.

39.88.2 Constructor & Destructor Documentation

39.88.2.1 KathirvalavakumarSubavathiInitialization()

```
KathirvalavakumarSubavathiInitialization (
    const arma::Mat< eT > & data,
    const double s ) [inline]
```

Initialize the random initialization rule with the given values.

Parameters

<i>data</i>	The input patterns.
<i>s</i>	Parameter that defines the active region.

Definition at line 70 of file kathirvalavakumar_subavathi_init.hpp.

39.88.3 Member Function Documentation

39.88.3.1 Initialize() [1/2]

```
void Initialize (
    arma::Mat< eT > & W,
    const size_t rows,
    const size_t cols ) [inline]
```

Initialize the elements of the specified weight matrix with the Kathirvalavakumar-Subavathi method.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.

Definition at line 85 of file `kathirvalavakumar_subavathi_init.hpp`.

References `RandomInitialization::Initialize()`.

Referenced by `KathirvalavakumarSubavathiInitialization::Initialize()`.

39.88.3.2 Initialize() [2/2]

```
void Initialize (
    arma::Cube< eT > & W,
    const size_t rows,
    const size_t cols,
    const size_t slices ) [inline]
```

Initialize the elements of the specified weight 3rd order tensor with the Kathirvalavakumar-Subavathi method.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.

Definition at line 102 of file `kathirvalavakumar_subavathi_init.hpp`.

References `KathirvalavakumarSubavathiInitialization::Initialize()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/kathirvalavakumar_subavathi_init.hpp` ↩

39.89 KLDivergence< InputDataType, OutputDataType > Class Template Reference

The Kullback–Leibler divergence is often used for continuous distributions (direct regression).

Public Member Functions

- **KLDivergence** (const bool takeMean=false)
Create the Kullback–Leibler Divergence object with the specified parameters.
- template<typename InputType , typename TargetType , typename OutputType >
void **Backward** (const InputType &&input, const TargetType &&target, OutputType &&output)
Ordinary feed backward pass of a neural network.
- template<typename InputType , typename TargetType >
double **Forward** (const InputType &&input, const TargetType &&target)
Computes the Kullback–Leibler divergence error function.
- OutputDataType & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the loss function.
- bool **TakeMean** () const
Get the value of takeMean.
- bool & **TakeMean** ()
Modify the value of takeMean.

39.89.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::KLDivergence< InputDataType, OutputDataType >
```

The Kullback–Leibler divergence is often used for continuous distributions (direct regression).

For more information, see the following paper.

```
article{Kullback1951,
  title   = {On Information and Sufficiency},
  author  = {S. Kullback, R.A. Leibler},
  journal = {The Annals of Mathematical Statistics},
  year    = {1951}
}
```

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 45 of file kl_divergence.hpp.

39.89.2 Constructor & Destructor Documentation

39.89.2.1 KLDivergence()

```
KLDivergence (  
    const bool takeMean = false )
```

Create the Kullback–Leibler Divergence object with the specified parameters.

Parameters

<i>takeMean</i>	Boolean variable to specify whether to take mean or not.
-----------------	--

39.89.3 Member Function Documentation

39.89.3.1 Backward()

```
void Backward (  
    const InputType && input,  
    const TargetType && target,  
    OutputType && output )
```

Ordinary feed backward pass of a neural network.

Parameters

<i>input</i>	The propagated input activation.
<i>target</i>	The target vector.
<i>output</i>	The calculated error.

39.89.3.2 Forward()

```
double Forward (  
    const InputType && input,  
    const TargetType && target )
```


Computes the Kullback–Leibler divergence error function.

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>target</i>	Target data to compare with.

39.89.3.3 OutputParameter() [1/2]

```
OutputDataType& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 78 of file kl_divergence.hpp.

39.89.3.4 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 80 of file kl_divergence.hpp.

39.89.3.5 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the loss function.

Referenced by KLDivergence< InputDataType, OutputDataType >::TakeMean().

39.89.3.6 TakeMean() [1/2]

```
bool TakeMean ( ) const [inline]
```

Get the value of takeMean.

Definition at line 83 of file kl_divergence.hpp.

39.89.3.7 TakeMean() [2/2]

```
bool& TakeMean ( ) [inline]
```

Modify the value of takeMean.

Definition at line 85 of file kl_divergence.hpp.

References KLDivergence< InputDataType, OutputDataType >::serialize().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/ **kl_divergence.hpp**

39.90 LayerNorm< InputDataType, OutputDataType > Class Template Reference

Declaration of the Layer Normalization class.

Public Member Functions

- **LayerNorm** ()
*Create the **LayerNorm** (p. 762) object.*
- **LayerNorm** (const size_t size, const double eps=1e-8)
*Create the **LayerNorm** (p. 762) object for a specified number of input units.*
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Backward pass through the layer.
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename eT >
void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Forward pass of Layer Normalization.
- template<typename eT >
void **Gradient** (const arma::Mat< eT > &&input, arma::Mat< eT > &&error, arma::Mat< eT > &&gradient)
Calculate the gradient using the output delta and the input activations.
- OutputDataType const & **Gradient** () const
Get the gradient.
- OutputDataType & **Gradient** ()
Modify the gradient.
- OutputDataType **Mean** ()
Get the mean across single training data.
- OutputDataType const & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()

Modify the output parameter.

- OutputDataType const & **Parameters** () const

Get the parameters.

- OutputDataType & **Parameters** ()

Modify the parameters.

- void **Reset** ()

Reset the layer parameters.

- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)

Serialize the layer.

- OutputDataType **Variance** ()

Get the variance across single training data.

39.90.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::LayerNorm< InputDataType, OutputDataType >
```

Declaration of the Layer Normalization class.

The layer transforms the input data into zero mean and unit variance and then scales and shifts the data by parameters, gamma and beta respectively over a single training data. These parameters are learnt by the network. Layer Normalization is different from Batch Normalization in the way that normalization is done for individual training cases, and the mean and standard deviations are computed across the layer dimensions, as opposed to across the batch.

For more information, refer to the following papers,

```
@article{Ba16,
  author    = {Jimmy Lei Ba, Jamie Ryan Kiros and Geoffrey E. Hinton},
  title     = {Layer Normalization},
  volume    = {abs/1607.06450},
  year      = {2016},
  url       = {http://arxiv.org/abs/1607.06450},
  eprint    = {1607.06450},
}
```

```
@article{Ioffe15,
  author    = {Sergey Ioffe and
              Christian Szegedy},
  title     = {Batch Normalization: Accelerating Deep Network Training by
              Reducing Internal Covariate Shift},
  journal   = {CoRR},
  volume    = {abs/1502.03167},
  year      = {2015},
  url       = {http://arxiv.org/abs/1502.03167},
  eprint    = {1502.03167},
}
```

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 65 of file layer_norm.hpp.

39.90.2 Constructor & Destructor Documentation

39.90.2.1 LayerNorm() [1/2]

```
LayerNorm ( )
```

Create the **LayerNorm** (p. 762) object.

39.90.2.2 LayerNorm() [2/2]

```
LayerNorm (
    const size_t size,
    const double eps = 1e-8 )
```

Create the **LayerNorm** (p. 762) object for a specified number of input units.

Parameters

<i>size</i>	The number of input units.
<i>eps</i>	The epsilon added to variance to ensure numerical stability.

39.90.3 Member Function Documentation

39.90.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Backward pass through the layer.

Parameters

<i>input</i>	The input activations.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.90.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 130 of file layer_norm.hpp.

39.90.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 132 of file layer_norm.hpp.

39.90.3.4 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Forward pass of Layer Normalization.

Transforms the input data into zero mean and unit variance, scales the data by a factor gamma and shifts it by beta.

Parameters

<i>input</i>	Input data for the layer.
<i>output</i>	Resulting output activations.

39.90.3.5 Gradient() [1/3]

```
void Gradient (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && error,
    arma::Mat< eT > && gradient )
```

Calculate the gradient using the output delta and the input activations.

Parameters

<i>input</i>	The input activations.
<i>error</i>	The calculated error.
<i>gradient</i>	The calculated gradient.

39.90.3.6 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 135 of file layer_norm.hpp.

39.90.3.7 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 137 of file layer_norm.hpp.

39.90.3.8 Mean()

```
OutputDataType Mean ( ) [inline]
```

Get the mean across single training data.

Definition at line 140 of file layer_norm.hpp.

39.90.3.9 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 125 of file layer_norm.hpp.

39.90.3.10 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 127 of file layer_norm.hpp.

39.90.3.11 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 120 of file layer_norm.hpp.

39.90.3.12 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 122 of file layer_norm.hpp.

39.90.3.13 Reset()

```
void Reset ( )
```

Reset the layer parameters.

39.90.3.14 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by LayerNorm< InputDataType, OutputDataType >::Variance().

39.90.3.15 Variance()

```
OutputDataType Variance ( ) [inline]
```

Get the variance across single training data.

Definition at line 143 of file layer_norm.hpp.

References LayerNorm< InputDataType, OutputDataType >::serialize().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **layer_norm.hpp**

39.91 LayerTraits< LayerType > Class Template Reference

This is a template class that can provide information about various layers.

Static Public Attributes

- static const bool **IsBiasLayer** = false
This is true if the layer is a bias layer.
- static const bool **IsBinary** = false
This is true if the layer is a binary layer.
- static const bool **IsConnection** = false
- static const bool **IsLSTMLayer** = false
- static const bool **IsOutputLayer** = false
This is true if the layer is an output layer.

39.91.1 Detailed Description

```
template<typename LayerType>  
class mlpack::ann::LayerTraits< LayerType >
```

This is a template class that can provide information about various layers.

By default, this class will provide the weakest possible assumptions on layer, and each layer should override values as necessary. If a layer doesn't need to override a value, then there's no need to write a **LayerTraits** (p. 769) specialization for that class.

Definition at line 29 of file layer_traits.hpp.

39.91.2 Member Data Documentation

39.91.2.1 IsBiasLayer

```
const bool IsBiasLayer = false [static]
```

This is true if the layer is a bias layer.

Definition at line 45 of file layer_traits.hpp.

39.91.2.2 IsBinary

```
const bool IsBinary = false [static]
```

This is true if the layer is a binary layer.

Definition at line 35 of file layer_traits.hpp.

39.91.2.3 IsConnection

```
const bool IsConnection = false [static]
```

Definition at line 55 of file layer_traits.hpp.

39.91.2.4 IsLSTMLayer

```
const bool IsLSTMLayer = false [static]
```

Definition at line 50 of file layer_traits.hpp.

39.91.2.5 IsOutputLayer

```
const bool IsOutputLayer = false [static]
```

This is true if the layer is an output layer.

Definition at line 40 of file layer_traits.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **layer_traits.hpp**

39.92 LeakyReLU< InputDataType, OutputDataType > Class Template Reference

The **LeakyReLU** (p. 771) activation function, defined by.

Public Member Functions

- **LeakyReLU** (const double alpha=0.03)
*Create the **LeakyReLU** (p. 771) object using the specified parameters.*
- double const & **Alpha** () const
Get the non zero gradient.
- double & **Alpha** ()
Modify the non zero gradient.
- template<typename DataType >
void **Backward** (const DataType &&input, DataType &&gy, DataType &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename InputType , typename OutputType >
void **Forward** (const InputType &&input, OutputType &&output)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- OutputDataType const & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.92.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::LeakyReLU< InputDataType, OutputDataType >
```

The **LeakyReLU** (p. 771) activation function, defined by.

$$f(x) = \max(x, \alpha * x)$$

$$f'(x) = \begin{cases} 1 & : x > 0 \\ \alpha & : x \leq 0 \end{cases}$$

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 44 of file leaky_relu.hpp.

39.92.2 Constructor & Destructor Documentation

39.92.2.1 LeakyReLU()

```
LeakyReLU (
    const double alpha = 0.03 )
```

Create the **LeakyReLU** (p. 771) object using the specified parameters.

The non zero gradient can be adjusted by specifying the parameter alpha in the range 0 to 1. Default (alpha = 0.03)

Parameters

<i>alpha</i>	Non zero gradient
--------------	-------------------

39.92.3 Member Function Documentation

39.92.3.1 Alpha() [1/2]

```
double const& Alpha ( ) const [inline]
```

Get the non zero gradient.

Definition at line 89 of file leaky_relu.hpp.

39.92.3.2 Alpha() [2/2]

```
double& Alpha ( ) [inline]
```

Modify the non zero gradient.

Definition at line 91 of file leaky_relu.hpp.

References `LeakyReLU< InputDataType, OutputDataType >::serialize()`.

39.92.3.3 Backward()

```
void Backward (
    const DataType && input,
    DataType && gy,
    DataType && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.92.3.4 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 84 of file leaky_relu.hpp.

39.92.3.5 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 86 of file leaky_relu.hpp.

39.92.3.6 Forward()

```
void Forward (
    const InputType && input,
    OutputType && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.92.3.7 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 79 of file leaky_relu.hpp.

39.92.3.8 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 81 of file leaky_relu.hpp.

39.92.3.9 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by LeakyReLU< InputDataType, OutputDataType >::Alpha().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **leaky_relu.hpp**

39.93 LecunNormalInitialization Class Reference

This class is used to initialize weight matrix with the Lecun Normalization initialization rule.

Public Member Functions

- **LecunNormalInitialization** ()
*Initialize the **LecunNormalInitialization** (p. 774) object.*
- void **Initialize** (arma::mat &W, const size_t rows, const size_t cols)
Initialize the elements of the weight matrix with the Lecun Normal initialization rule.
- void **Initialize** (arma::cube &W, const size_t rows, const size_t cols, const size_t slices)
Initialize the elements of the specified weight 3rd order tensor with Lecun Normal initialization rule.

39.93.1 Detailed Description

This class is used to initialize weight matrix with the Lecun Normalization initialization rule.

For more information, the following papers can be referred to:

```
@inproceedings{Klambauer2017,
  title = {Self-Normalizing Neural Networks.},
  author = {Klambauer, Günter and Unterthiner, Thomas
    and Mayr, Andreas and Hochreiter, Sepp},
  pages = {972-981},
  year = {2017}
}

@inproceedings{LeCun1998,
  title = {Efficient BackProp},
  author = {LeCun, Yann and Bottou, L{\'e}on and Orr, Genevieve B.
    and M{\"u}ller, Klaus-Robert},
  year = {1998},
  pages = {9--50}
}
```

Definition at line 49 of file lecun_normal_init.hpp.

39.93.2 Constructor & Destructor Documentation

39.93.2.1 LecunNormalInitialization()

```
LecunNormalInitialization ( ) [inline]
```

Initialize the **LecunNormalInitialization** (p. 774) object.

Definition at line 55 of file lecun_normal_init.hpp.

39.93.3 Member Function Documentation

39.93.3.1 Initialize() [1/2]

```
void Initialize (
    arma::mat & W,
    const size_t rows,
    const size_t cols ) [inline]
```

Initialize the elements of the weight matrix with the Lecun Normal initialization rule.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.

Definition at line 68 of file `lecun_normal_init.hpp`.

Referenced by `LecunNormalInitialization::Initialize()`.

39.93.3.2 Initialize() [2/2]

```
void Initialize (
    arma::cube & W,
    const size_t rows,
    const size_t cols,
    const size_t slices ) [inline]
```

Initialize the elements of the specified weight 3rd order tensor with Lecun Normal initialization rule.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.
<i>slice</i>	Numbers of slices.

Definition at line 96 of file `lecun_normal_init.hpp`.

References `LecunNormalInitialization::Initialize()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/lecun_normal_init.hpp`

39.94 Linear< InputDataType, OutputDataType > Class Template Reference

Implementation of the **Linear** (p. 776) layer class.

Public Member Functions

- **Linear** ()
*Create the **Linear** (p. 776) object.*
 - **Linear** (const size_t inSize, const size_t outSize)
*Create the **Linear** (p. 776) layer object using the specified number of units.*
 - template<typename eT >
void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
 - OutputDataType const & **Delta** () const
Get the delta.
 - OutputDataType & **Delta** ()
Modify the delta.
 - template<typename eT >
void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
 - template<typename eT >
void **Gradient** (const arma::Mat< eT > &&input, arma::Mat< eT > &&error, arma::Mat< eT > &&gradient)
 - OutputDataType const & **Gradient** () const
Get the gradient.
 - OutputDataType & **Gradient** ()
Modify the gradient.
 - InputDataType const & **InputParameter** () const
Get the input parameter.
 - InputDataType & **InputParameter** ()
Modify the input parameter.
 - OutputDataType const & **OutputParameter** () const
Get the output parameter.
 - OutputDataType & **OutputParameter** ()
Modify the output parameter.
 - OutputDataType const & **Parameters** () const
Get the parameters.
 - OutputDataType & **Parameters** ()
Modify the parameters.
 - void **Reset** ()
 - template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
- Serialize the layer.*

39.94.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::Linear< InputDataType, OutputDataType >
```

Implementation of the **Linear** (p. 776) layer class.

The **Linear** (p. 776) class represents a single layer of a neural network.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 59 of file layer_types.hpp.

39.94.2 Constructor & Destructor Documentation

39.94.2.1 Linear() [1/2]

```
Linear ( )
```

Create the **Linear** (p. 776) object.

39.94.2.2 Linear() [2/2]

```
Linear (
    const size_t inSize,
    const size_t outSize )
```

Create the **Linear** (p. 776) layer object using the specified number of units.

Parameters

<i>inSize</i>	The number of input units.
<i>outSize</i>	The number of output units.

39.94.3 Member Function Documentation

39.94.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
```

```
arma::Mat< eT > && gy,
arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.94.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 107 of file linear.hpp.

39.94.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 109 of file linear.hpp.

39.94.3.4 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.94.3.5 Gradient() [1/3]

```
void Gradient (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && error,
    arma::Mat< eT > && gradient )
```

39.94.3.6 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 112 of file linear.hpp.

39.94.3.7 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 114 of file linear.hpp.

References `Linear< InputDataType, OutputDataType >::serialize()`.

39.94.3.8 InputParameter() [1/2]

```
InputDataType const& InputParameter ( ) const [inline]
```

Get the input parameter.

Definition at line 97 of file linear.hpp.

39.94.3.9 InputParameter() [2/2]

```
InputDataType& InputParameter ( ) [inline]
```

Modify the input parameter.

Definition at line 99 of file linear.hpp.

39.94.3.10 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 102 of file linear.hpp.

39.94.3.11 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 104 of file linear.hpp.

39.94.3.12 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 92 of file linear.hpp.

39.94.3.13 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 94 of file linear.hpp.

39.94.3.14 Reset()

```
void Reset ( )
```

39.94.3.15 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by `Linear< InputDataType, OutputDataType >::Gradient()`.

The documentation for this class was generated from the following files:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ layer_types.hpp`
- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ linear.hpp`

39.95 LinearNoBias< InputDataType, OutputDataType > Class Template Reference

Implementation of the **LinearNoBias** (p. 782) class.

Public Member Functions

- **LinearNoBias** ()
*Create the **LinearNoBias** (p. 782) object.*
- **LinearNoBias** (const size_t inSize, const size_t outSize)
*Create the **LinearNoBias** (p. 782) object using the specified number of units.*
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename eT >
void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- template<typename eT >
void **Gradient** (const arma::Mat< eT > &&input, arma::Mat< eT > &&error, arma::Mat< eT > &&gradient)
• OutputDataType const & **Gradient** () const
Get the gradient.

- OutputDataType & **Gradient** ()
Modify the gradient.
 - InputDataType const & **InputParameter** () const
Get the input parameter.
 - InputDataType & **InputParameter** ()
Modify the input parameter.
 - OutputDataType const & **OutputParameter** () const
Get the output parameter.
 - OutputDataType & **OutputParameter** ()
Modify the output parameter.
 - OutputDataType const & **Parameters** () const
Get the parameters.
 - OutputDataType & **Parameters** ()
Modify the parameters.
 - void **Reset** ()
 - template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
- Serialize the layer.*

39.95.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::LinearNoBias< InputDataType, OutputDataType >
```

Implementation of the **LinearNoBias** (p. 782) class.

The **LinearNoBias** (p. 782) class represents a single layer of a neural network.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 60 of file layer_types.hpp.

39.95.2 Constructor & Destructor Documentation

39.95.2.1 LinearNoBias() [1/2]

```
LinearNoBias ( )
```

Create the **LinearNoBias** (p. 782) object.

39.95.2.2 LinearNoBias() [2/2]

```
LinearNoBias (  
    const size_t inSize,  
    const size_t outSize )
```

Create the **LinearNoBias** (p. 782) object using the specified number of units.

Parameters

<i>inSize</i>	The number of input units.
<i>outSize</i>	The number of output units.

39.95.3 Member Function Documentation

39.95.3.1 Backward()

```
void Backward (  
    const arma::Mat< eT > && ,  
    arma::Mat< eT > && gy,  
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.95.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 106 of file linear_no_bias.hpp.

39.95.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 108 of file linear_no_bias.hpp.

39.95.3.4 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.95.3.5 Gradient() [1/3]

```
void Gradient (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && error,
    arma::Mat< eT > && gradient )
```

39.95.3.6 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 111 of file linear_no_bias.hpp.

39.95.3.7 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 113 of file linear_no_bias.hpp.

References LinearNoBias< InputDataType, OutputDataType >::serialize().

39.95.3.8 InputParameter() [1/2]

```
InputDataType const& InputParameter ( ) const [inline]
```

Get the input parameter.

Definition at line 96 of file linear_no_bias.hpp.

39.95.3.9 InputParameter() [2/2]

```
InputDataType& InputParameter ( ) [inline]
```

Modify the input parameter.

Definition at line 98 of file linear_no_bias.hpp.

39.95.3.10 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 101 of file linear_no_bias.hpp.

39.95.3.11 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 103 of file linear_no_bias.hpp.

39.95.3.12 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 91 of file linear_no_bias.hpp.

39.95.3.13 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 93 of file linear_no_bias.hpp.

39.95.3.14 Reset()

```
void Reset ( )
```

39.95.3.15 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by LinearNoBias< InputDataType, OutputDataType >::Gradient().

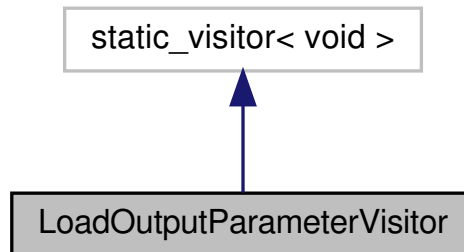
The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **layer_types.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **linear_no_bias.hpp**

39.96 LoadOutputParameterVisitor Class Reference

LoadOutputParameterVisitor (p. 788) restores the output parameter using the given parameter set.

Inheritance diagram for LoadOutputParameterVisitor:



Public Member Functions

- **LoadOutputParameterVisitor** (std::vector< arma::mat > &¶meter)
Restore the output parameter given a parameter set.
- template<typename LayerType >
 void **operator()** (LayerType *layer) const
Restore the output parameter.

39.96.1 Detailed Description

LoadOutputParameterVisitor (p. 788) restores the output parameter using the given parameter set.

Definition at line 28 of file load_output_parameter_visitor.hpp.

39.96.2 Constructor & Destructor Documentation

39.96.2.1 LoadOutputParameterVisitor()

```

LoadOutputParameterVisitor (
    std::vector< arma::mat > && parameter )
  
```

Restore the output parameter given a parameter set.

39.96.3 Member Function Documentation

39.96.3.1 operator>()

```
void operator() (
    LayerType * layer ) const
```

Restore the output parameter.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **load_output_parameter_visitor.hpp**

39.97 LogisticFunction Class Reference

The logistic function, defined by.

Static Public Member Functions

- static double **Deriv** (const double y)
Computes the first derivative of the logistic function.
- template<typename InputVecType , typename OutputVecType >
static void **Deriv** (const InputVecType &y, OutputVecType &x)
Computes the first derivatives of the logistic function.
- template<typename eT >
static double **Fn** (const eT x)
Computes the logistic function.
- template<typename InputVecType , typename OutputVecType >
static void **Fn** (const InputVecType &x, OutputVecType &y)
Computes the logistic function.
- static double **Inv** (const double y)
Computes the inverse of the logistic function.
- template<typename InputVecType , typename OutputVecType >
static void **Inv** (const InputVecType &y, OutputVecType &x)
Computes the inverse of the logistic function.

39.97.1 Detailed Description

The logistic function, defined by.

$$\begin{aligned} f(x) &= \frac{1}{1 + e^{-x}} \\ f'(x) &= f(x) * (1 - f(x)) \\ f^{-1}(y) &= \ln\left(\frac{y}{1 - y}\right) \end{aligned}$$

Definition at line 29 of file logistic_function.hpp.

39.97.2 Member Function Documentation

39.97.2.1 Deriv() [1/2]

```
static double Deriv (  
    const double y ) [inline], [static]
```

Computes the first derivative of the logistic function.

Parameters

<i>x</i>	Input data.
----------	-------------

Returns

$f'(x)$

Definition at line 70 of file logistic_function.hpp.

39.97.2.2 Deriv() [2/2]

```
static void Deriv (  
    const InputVecType & y,  
    OutputVecType & x ) [inline], [static]
```

Computes the first derivatives of the logistic function.

Parameters

<i>y</i>	Input activations.
<i>x</i>	The resulting derivatives.

Definition at line 82 of file logistic_function.hpp.

39.97.2.3 Fn() [1/2]

```
static double Fn (  
    const eT x ) [inline], [static]
```

Computes the logistic function.

Parameters

<i>x</i>	Input data.
----------	-------------

Returns

$f(x)$.

Definition at line 39 of file logistic_function.hpp.

39.97.2.4 Fn() [2/2]

```
static void Fn (  
    const InputVecType & x,  
    OutputVecType & y ) [inline], [static]
```

Computes the logistic function.

Parameters

<i>x</i>	Input data.
<i>y</i>	The resulting output activation.

Definition at line 59 of file logistic_function.hpp.

39.97.2.5 Inv() [1/2]

```
static double Inv (  
    const double y ) [inline], [static]
```

Computes the inverse of the logistic function.

Parameters

<i>y</i>	Input data.
----------	-------------

Returns

$f^{-1}(y)$

Definition at line 93 of file logistic_function.hpp.

39.97.2.6 Inv() [2/2]

```
static void Inv (
    const InputVecType & y,
    OutputVecType & x ) [inline], [static]
```

Computes the inverse of the logistic function.

Parameters

<i>y</i>	Input data.
----------	-------------

Returns

x The resulting inverse of the input data.

Definition at line 105 of file `logistic_function.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/ logistic_function.hpp`

39.98 LogSoftMax< InputDataType, OutputDataType > Class Template Reference

Implementation of the log softmax layer.

Public Member Functions

- **LogSoftMax** ()
Create the LogSoftmax object.
- `template<typename eT >`
`void Backward (const arma::Mat< eT > &&input, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)`
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- `InputDataType & Delta () const`
Get the delta.
- `InputDataType & Delta ()`
Modify the delta.
- `template<typename InputType , typename OutputType >`
`void Forward (const InputType &&input, OutputType &&output)`
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- `OutputDataType & OutputParameter () const`
Get the output parameter.
- `OutputDataType & OutputParameter ()`
Modify the output parameter.
- `template<typename Archive >`
`void serialize (Archive &, const unsigned int)`
Serialize the layer.

39.98.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::LogSoftMax< InputDataType, OutputDataType >
```

Implementation of the log softmax layer.

The log softmax loss layer computes the multinomial logistic loss of the softmax of its inputs. This layer is meant to be used in combination with the negative log likelihood layer (NegativeLogLikelihoodLayer), which expects that the input contains log-probabilities for each class.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 36 of file log_softmax.hpp.

39.98.2 Constructor & Destructor Documentation

39.98.2.1 LogSoftMax()

```
LogSoftMax ( )
```

Create the LogSoftmax object.

39.98.3 Member Function Documentation

39.98.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.98.3.2 Delta() [1/2]

```
InputDataType& Delta ( ) const [inline]
```

Get the delta.

Definition at line 74 of file log_softmax.hpp.

39.98.3.3 Delta() [2/2]

```
InputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 76 of file log_softmax.hpp.

References `LogSoftMax< InputDataType, OutputDataType >::serialize()`.

39.98.3.4 Forward()

```
void Forward (
    const InputType && input,
    OutputType && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.98.3.5 OutputParameter() [1/2]

```
OutputDataType& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 69 of file log_softmax.hpp.

39.98.3.6 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 71 of file log_softmax.hpp.

39.98.3.7 serialize()

```
void serialize (
    Archive & ,
    const unsigned int )
```

Serialize the layer.

Referenced by LogSoftMax< InputDataType, OutputDataType >::Delta().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ log_softmax.hpp

39.99 Lookup< InputDataType, OutputDataType > Class Template Reference

Implementation of the **Lookup** (p. 795) class.

Public Member Functions

- **Lookup** (const size_t inSize=0, const size_t outSize=0)
*Create the **Lookup** (p. 795) object using the specified number of input and output units.*
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, const arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename eT >
void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- template<typename eT >
void **Gradient** (const arma::Mat< eT > &&input, arma::Mat< eT > &&error, arma::Mat< eT > &&gradient)
Get the gradient.
- OutputDataType const & **Gradient** () const
Get the gradient.
- OutputDataType & **Gradient** ()
Modify the gradient.
- OutputDataType const & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- OutputDataType const & **Parameters** () const
Get the parameters.
- OutputDataType & **Parameters** ()
Modify the parameters.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.99.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::Lookup< InputDataType, OutputDataType >
```

Implementation of the **Lookup** (p. 795) class.

The **Lookup** (p. 795) class is a particular convolution, where the width of the convolution is 1.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 35 of file lookup.hpp.

39.99.2 Constructor & Destructor Documentation

39.99.2.1 Lookup()

```
Lookup (  
    const size_t inSize = 0,  
    const size_t outSize = 0 )
```

Create the **Lookup** (p. 795) object using the specified number of input and output units.

Parameters

<i>inSize</i>	The number of input units.
<i>outSize</i>	The number of output units.

39.99.3 Member Function Documentation

39.99.3.1 Backward()

```
void Backward (  
    const arma::Mat< eT > && ,  
    const arma::Mat< eT > && gy,  
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.99.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 94 of file lookup.hpp.

39.99.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 96 of file lookup.hpp.

39.99.3.4 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.99.3.5 Gradient() [1/3]

```
void Gradient (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && error,
    arma::Mat< eT > && gradient )
```

39.99.3.6 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 99 of file lookup.hpp.

39.99.3.7 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 101 of file lookup.hpp.

References Lookup< InputDataType, OutputDataType >::serialize().

39.99.3.8 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 89 of file lookup.hpp.

39.99.3.9 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 91 of file lookup.hpp.

39.99.3.10 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 84 of file lookup.hpp.

39.99.3.11 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 86 of file lookup.hpp.

39.99.3.12 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by `Lookup< InputDataType, OutputDataType >::Gradient()`.

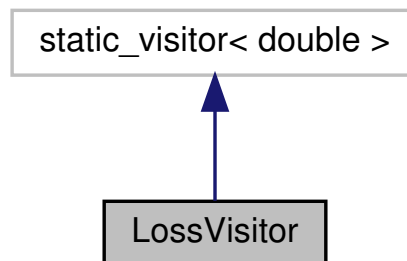
The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ lookup.hpp`

39.100 LossVisitor Class Reference

LossVisitor (p. 800) exposes the `Loss()` method of the given module.

Inheritance diagram for LossVisitor:

**Public Member Functions**

- `template<typename LayerType >`
`double operator() (LayerType *layer) const`
Return the Loss.

39.100.1 Detailed Description

LossVisitor (p. 800) exposes the Loss() method of the given module.

Definition at line 26 of file loss_visitor.hpp.

39.100.2 Member Function Documentation

39.100.2.1 operator>()

```
double operator() (
    LayerType * layer ) const
```

Return the Loss.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **loss_visitor.hpp**

39.101 LSTM< InputDataType, OutputDataType > Class Template Reference

Implementation of the **LSTM** (p. 801) module class.

Public Member Functions

- **LSTM** ()
*Create the **LSTM** (p. 801) object.*
- **LSTM** (const size_t inSize, const size_t outSize, const size_t rho=std::numeric_limits< size_t >::max())
*Create the **LSTM** (p. 801) layer object using the specified parameters.*
- template<typename InputType , typename ErrorType , typename GradientType >
void **Backward** (const InputType &&input, ErrorType &&gy, GradientType &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename InputType , typename OutputType >
void **Forward** (InputType &&input, OutputType &&output)
Ordinary feed-forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- template<typename InputType , typename OutputType >
void **Forward** (InputType &&input, OutputType &&output, OutputType &&cellState, bool useCellState=false)

Ordinary feed-forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

- `template<typename InputType , typename ErrorType , typename GradientType >`
`void Gradient (InputType &&input, ErrorType &&error, GradientType &&gradient)`
- `OutputDataType const & Gradient () const`
Get the gradient.
- `OutputDataType & Gradient ()`
Modify the gradient.
- `OutputDataType const & OutputParameter () const`
Get the output parameter.
- `OutputDataType & OutputParameter ()`
Modify the output parameter.
- `OutputDataType const & Parameters () const`
Get the parameters.
- `OutputDataType & Parameters ()`
Modify the parameters.
- `void Reset ()`
- `void ResetCell (const size_t size)`
- `size_t Rho () const`
Get the maximum number of steps to backpropagate through time (BPTT).
- `size_t & Rho ()`
Modify the maximum number of steps to backpropagate through time (BPTT).
- `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`
Serialize the layer.

39.101.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::LSTM< InputDataType, OutputDataType >
```

Implementation of the **LSTM** (p. 801) module class.

The implementation corresponds to the following algorithm:

$$i = \text{sigmoid}(W \cdot x + W \cdot h + W \cdot c + b) \quad (39.7)$$

$$f = \text{sigmoid}(W \cdot x + W \cdot h + W \cdot c + b) \quad (39.8)$$

$$z = \tanh(W \cdot x + W \cdot h + b) \quad (39.9)$$

$$c = f \cdot c + i \cdot z \quad (39.10)$$

$$o = \text{sigmoid}(W \cdot x + W \cdot h + W \cdot c + b) \quad (39.11)$$

$$h = o \cdot \tanh(c) \quad (39.12)$$

For more information, see the following.

```
@article{Graves2013,
  author = {Alex Graves and Abdel{-}rahman Mohamed and Geoffrey E. Hinton},
  title = {Speech Recognition with Deep Recurrent Neural Networks},
  journal = {CoRR},
  year = {2013},
  url = {http://arxiv.org/abs/1303.5778},
}
```

See also

FastLSTM (p. 685) for a faster **LSTM** (p. 801) version which combines the calculation of the input, forget, output gates and hidden state in a single step.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 61 of file layer_types.hpp.

39.101.2 Constructor & Destructor Documentation

39.101.2.1 LSTM() ^[1/2]

LSTM ()

Create the **LSTM** (p. 801) object.

39.101.2.2 LSTM() ^[2/2]

```
LSTM (
    const size_t inSize,
    const size_t outSize,
    const size_t rho = std::numeric_limits< size_t >::max() )
```

Create the **LSTM** (p. 801) layer object using the specified parameters.

Parameters

<i>inSize</i>	The number of input units.
<i>outSize</i>	The number of output units.
<i>rho</i>	Maximum number of steps to backpropagate through time (BPTT).

39.101.3 Member Function Documentation

39.101.3.1 Backward()

```
void Backward (
    const InputType && input,
    ErrorType && gy,
    GradientType && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.101.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 155 of file `Istm.hpp`.

39.101.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 157 of file `Istm.hpp`.

39.101.3.4 Forward() [1/2]

```
void Forward (
    InputType && input,
    OutputType && output )
```

Ordinary feed-forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.101.3.5 Forward() [2/2]

```
void Forward (
    InputType && input,
    OutputType && output,
    OutputType && cellState,
    bool useCellState = false )
```

Ordinary feed-forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.
<i>cellState</i>	Cell state of the LSTM (p. 801).
<i>useCellState</i>	Use the cellState passed in the LSTM (p. 801) cell.

39.101.3.6 Gradient() [1/3]

```
void Gradient (
    InputType && input,
    ErrorType && error,
    GradientType && gradient )
```

39.101.3.7 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 160 of file lstm.hpp.

39.101.3.8 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 162 of file lstm.hpp.

References LSTM< InputDataType, OutputDataType >::serialize().

39.101.3.9 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 150 of file lstm.hpp.

39.101.3.10 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 152 of file lstm.hpp.

39.101.3.11 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 145 of file lstm.hpp.

39.101.3.12 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 147 of file lstm.hpp.

39.101.3.13 Reset()

```
void Reset ( )
```

39.101.3.14 ResetCell()

```
void ResetCell (
    const size_t size )
```

39.101.3.15 Rho() [1/2]

```
size_t Rho ( ) const [inline]
```

Get the maximum number of steps to backpropagate through time (BPTT).

Definition at line 140 of file `Istm.hpp`.

39.101.3.16 Rho() [2/2]

```
size_t& Rho ( ) [inline]
```

Modify the maximum number of steps to backpropagate through time (BPTT).

Definition at line 142 of file `Istm.hpp`.

39.101.3.17 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by LSTM< InputDataType, OutputDataType >::Gradient().

The documentation for this class was generated from the following files:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ layer_types.hpp`
- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ Istm.hpp`

39.102 MaxPooling< InputDataType, OutputDataType > Class Template Reference

Implementation of the **MaxPooling** (p. 808) layer.

Public Member Functions

- **MaxPooling** ()
*Create the **MaxPooling** (p. 808) object.*
- **MaxPooling** (const size_t kW, const size_t kH, const size_t dW=1, const size_t dH=1, const bool floor=true)
*Create the **MaxPooling** (p. 808) object using the specified number of units.*
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, using 3rd-order tensors as input, calculating the function f(x) by propagating x backwards through f.
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- bool **Deterministic** () const
Get the value of the deterministic parameter.
- bool & **Deterministic** ()
Modify the value of the deterministic parameter.
- template<typename eT >
void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Ordinary feed forward pass of a neural network, evaluating the function f(x) by propagating the activity forward through f.
- size_t const & **InputHeight** () const
Get the height.
- size_t & **InputHeight** ()
Modify the height.
- size_t const & **InputWidth** () const
Get the width.
- size_t & **InputWidth** ()
Modify the width.
- size_t const & **OutputHeight** () const
Get the height.
- size_t & **OutputHeight** ()
Modify the height.
- OutputDataType const & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- size_t const & **OutputWidth** () const
Get the width.
- size_t & **OutputWidth** ()
Modify the width.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.102.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::MaxPooling< InputDataType, OutputDataType >
```

Implementation of the **MaxPooling** (p. 808) layer.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 52 of file max_pooling.hpp.

39.102.2 Constructor & Destructor Documentation

39.102.2.1 MaxPooling() [1/2]

```
MaxPooling ( )
```

Create the **MaxPooling** (p. 808) object.

39.102.2.2 MaxPooling() [2/2]

```
MaxPooling (
    const size_t kW,
    const size_t kH,
    const size_t dW = 1,
    const size_t dH = 1,
    const bool floor = true )
```

Create the **MaxPooling** (p. 808) object using the specified number of units.

Parameters

<i>kW</i>	Width of the pooling window.
<i>kH</i>	Height of the pooling window.
<i>dW</i>	Width of the stride operation.
<i>dH</i>	Width of the stride operation.
<i>floor</i>	Rounding operator (floor or ceil).

39.102.3 Member Function Documentation

39.102.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, using 3rd-order tensors as input, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.102.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 103 of file max_pooling.hpp.

39.102.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 105 of file max_pooling.hpp.

39.102.3.4 Deterministic() [1/2]

```
bool Deterministic ( ) const [inline]
```

Get the value of the deterministic parameter.

Definition at line 128 of file max_pooling.hpp.

39.102.3.5 Deterministic() [2/2]

```
bool& Deterministic ( ) [inline]
```

Modify the value of the deterministic parameter.

Definition at line 130 of file max_pooling.hpp.

39.102.3.6 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.102.3.7 InputHeight() [1/2]

```
size_t const& InputHeight ( ) const [inline]
```

Get the height.

Definition at line 113 of file max_pooling.hpp.

39.102.3.8 InputHeight() [2/2]

```
size_t& InputHeight ( ) [inline]
```

Modify the height.

Definition at line 115 of file max_pooling.hpp.

39.102.3.9 InputWidth() [1/2]

```
size_t const& InputWidth ( ) const [inline]
```

Get the width.

Definition at line 108 of file max_pooling.hpp.

39.102.3.10 InputWidth() [2/2]

```
size_t& InputWidth ( ) [inline]
```

Modify the width.

Definition at line 110 of file max_pooling.hpp.

39.102.3.11 OutputHeight() [1/2]

```
size_t const& OutputHeight ( ) const [inline]
```

Get the height.

Definition at line 123 of file max_pooling.hpp.

39.102.3.12 OutputHeight() [2/2]

```
size_t& OutputHeight ( ) [inline]
```

Modify the height.

Definition at line 125 of file max_pooling.hpp.

39.102.3.13 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 98 of file max_pooling.hpp.

39.102.3.14 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 100 of file max_pooling.hpp.

39.102.3.15 OutputWidth() [1/2]

```
size_t const& OutputWidth ( ) const [inline]
```

Get the width.

Definition at line 118 of file max_pooling.hpp.

39.102.3.16 OutputWidth() [2/2]

```
size_t& OutputWidth ( ) [inline]
```

Modify the width.

Definition at line 120 of file max_pooling.hpp.

39.102.3.17 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **max_pooling.hpp**

39.103 MaxPoolingRule Class Reference

Public Member Functions

- `template<typename MatType >`
`size_t Pooling (const MatType &input)`

39.103.1 Detailed Description

Definition at line 25 of file `max_pooling.hpp`.

39.103.2 Member Function Documentation

39.103.2.1 Pooling()

```
size_t Pooling (  
    const MatType & input ) [inline]
```

Definition at line 34 of file `max_pooling.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ max_pooling.hpp`

39.104 MeanPooling< InputDataType, OutputDataType > Class Template Reference

Implementation of the **MeanPooling** (p. 814).

Public Member Functions

- **MeanPooling** ()
*Create the **MeanPooling** (p. 814) object.*
- **MeanPooling** (const size_t kW, const size_t kH, const size_t dW=1, const size_t dH=1, const bool floor=true)
*Create the **MeanPooling** (p. 814) object using the specified number of units.*
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, using 3rd-order tensors as input, calculating the function f(x) by propagating x backwards through f.
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- bool **Deterministic** () const
Get the value of the deterministic parameter.
- bool & **Deterministic** ()
Modify the value of the deterministic parameter.
- template<typename eT >
void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Ordinary feed forward pass of a neural network, evaluating the function f(x) by propagating the activity forward through f.
- size_t const & **InputHeight** () const
Get the height.
- size_t & **InputHeight** ()
Modify the height.
- size_t const & **InputWidth** () const
Get the width.
- size_t & **InputWidth** ()
Modify the width.
- size_t const & **OutputHeight** () const
Get the height.
- size_t & **OutputHeight** ()
Modify the height.
- OutputDataType const & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- size_t const & **OutputWidth** () const
Get the width.
- size_t & **OutputWidth** ()
Modify the width.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.104.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>  
class mlpack::ann::MeanPooling< InputDataType, OutputDataType >
```

Implementation of the **MeanPooling** (p. 814).

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 33 of file mean_pooling.hpp.

39.104.2 Constructor & Destructor Documentation

39.104.2.1 MeanPooling() [1/2]

```
MeanPooling ( )
```

Create the **MeanPooling** (p. 814) object.

39.104.2.2 MeanPooling() [2/2]

```
MeanPooling (
    const size_t kW,
    const size_t kH,
    const size_t dW = 1,
    const size_t dH = 1,
    const bool floor = true )
```

Create the **MeanPooling** (p. 814) object using the specified number of units.

Parameters

<i>kW</i>	Width of the pooling window.
<i>kH</i>	Height of the pooling window.
<i>dW</i>	Width of the stride operation.
<i>dH</i>	Width of the stride operation.

39.104.3 Member Function Documentation

39.104.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, using 3rd-order tensors as input, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.104.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 83 of file mean_pooling.hpp.

39.104.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 85 of file mean_pooling.hpp.

39.104.3.4 Deterministic() [1/2]

```
bool Deterministic ( ) const [inline]
```

Get the value of the deterministic parameter.

Definition at line 108 of file mean_pooling.hpp.

39.104.3.5 Deterministic() [2/2]

```
bool& Deterministic ( ) [inline]
```

Modify the value of the deterministic parameter.

Definition at line 110 of file mean_pooling.hpp.

References MeanPooling< InputDataType, OutputDataType >::serialize().

39.104.3.6 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.104.3.7 InputHeight() [1/2]

```
size_t const& InputHeight ( ) const [inline]
```

Get the height.

Definition at line 93 of file mean_pooling.hpp.

39.104.3.8 InputHeight() [2/2]

```
size_t& InputHeight ( ) [inline]
```

Modify the height.

Definition at line 95 of file mean_pooling.hpp.

39.104.3.9 InputWidth() [1/2]

```
size_t const& InputWidth ( ) const [inline]
```

Get the width.

Definition at line 88 of file mean_pooling.hpp.

39.104.3.10 InputWidth() [2/2]

```
size_t& InputWidth ( ) [inline]
```

Modify the width.

Definition at line 90 of file mean_pooling.hpp.

39.104.3.11 OutputHeight() [1/2]

```
size_t const& OutputHeight ( ) const [inline]
```

Get the height.

Definition at line 103 of file mean_pooling.hpp.

39.104.3.12 OutputHeight() [2/2]

```
size_t& OutputHeight ( ) [inline]
```

Modify the height.

Definition at line 105 of file mean_pooling.hpp.

39.104.3.13 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 78 of file mean_pooling.hpp.

39.104.3.14 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 80 of file mean_pooling.hpp.

39.104.3.15 OutputWidth() [1/2]

```
size_t const& OutputWidth ( ) const [inline]
```

Get the width.

Definition at line 98 of file mean_pooling.hpp.

39.104.3.16 OutputWidth() [2/2]

```
size_t& OutputWidth ( ) [inline]
```

Modify the width.

Definition at line 100 of file mean_pooling.hpp.

39.104.3.17 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by MeanPooling< InputDataType, OutputDataType >::Deterministic().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **mean_pooling.hpp**

39.105 MeanPoolingRule Class Reference

Public Member Functions

- template<typename MatType >
double **Pooling** (const MatType &input)
- template<typename MatType >
void **Unpooling** (const MatType &input, const double value, MatType &output)

39.105.1 Detailed Description

Definition at line 42 of file glimpse.hpp.

39.105.2 Member Function Documentation

39.105.2.1 Pooling()

```
double Pooling (  
    const MatType & input ) [inline]
```

Definition at line 51 of file glimpse.hpp.

Referenced by Glimpse< InputDataType, OutputDataType >::Deterministic().

39.105.2.2 Unpooling()

```
void Unpooling (  
    const MatType & input,  
    const double value,  
    MatType & output ) [inline]
```

Definition at line 64 of file glimpse.hpp.

Referenced by Glimpse< InputDataType, OutputDataType >::Deterministic().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **glimpse.hpp**

39.106 MeanSquaredError< InputDataType, OutputDataType > Class Template Reference

The mean squared error performance function measures the network's performance according to the mean of squared errors.

Public Member Functions

- **MeanSquaredError** ()
*Create the **MeanSquaredError** (p. 823) object.*
- template<typename InputType , typename TargetType , typename OutputType >
void **Backward** (const InputType &&input, const TargetType &&target, OutputType &&output)
Ordinary feed backward pass of a neural network.
- template<typename InputType , typename TargetType >
double **Forward** (const InputType &&input, const TargetType &&target)
Computes the mean squared error function.
- OutputDataType & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.106.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::MeanSquaredError< InputDataType, OutputDataType >
```

The mean squared error performance function measures the network's performance according to the mean of squared errors.

Template Parameters

<i>ActivationFunction</i>	Activation function used for the embedding layer.
<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 34 of file mean_squared_error.hpp.

39.106.2 Constructor & Destructor Documentation

39.106.2.1 MeanSquaredError()

MeanSquaredError ()

Create the **MeanSquaredError** (p. 823) object.

39.106.3 Member Function Documentation

39.106.3.1 Backward()

```
void Backward (
    const InputType && input,
    const TargetType && target,
    OutputType && output )
```

Ordinary feed backward pass of a neural network.

Parameters

<i>input</i>	The propagated input activation.
<i>target</i>	The target vector.
<i>output</i>	The calculated error.

39.106.3.2 Forward()

```
double Forward (
    const InputType && input,
    const TargetType && target )
```

Computes the mean squared error function.

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>target</i>	The target vector.

39.106.3.3 OutputParameter() [1/2]

```
OutputDataType& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 63 of file mean_squared_error.hpp.

39.106.3.4 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 65 of file mean_squared_error.hpp.

References MeanSquaredError< InputDataType, OutputDataType >::serialize().

39.106.3.5 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by MeanSquaredError< InputDataType, OutputDataType >::OutputParameter().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/ **mean_squared_error.hpp**

39.107 MultiplyConstant< InputDataType, OutputDataType > Class Template Reference

Implementation of the multiply constant layer.

Public Member Functions

- **MultiplyConstant** (const double scalar=1.0)
*Create the **MultiplyConstant** (p. 825) object.*
- template<typename DataType >
void **Backward** (const DataType &&, DataType &&gy, DataType &&g)
Ordinary feed backward pass of a neural network.
- OutputDataType & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename InputType , typename OutputType >
void **Forward** (const InputType &&input, OutputType &&output)
Ordinary feed forward pass of a neural network.
- OutputDataType & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.107.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::MultiplyConstant< InputDataType, OutputDataType >
```

Implementation of the multiply constant layer.

The multiply constant layer multiplies the input by a (non-learnable) constant.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 34 of file multiply_constant.hpp.

39.107.2 Constructor & Destructor Documentation

39.107.2.1 MultiplyConstant()

```
MultiplyConstant (
    const double scalar = 1.0 )
```

Create the **MultiplyConstant** (p. 825) object.

39.107.3 Member Function Documentation

39.107.3.1 Backward()

```
void Backward (
    const DataType && ,
    DataType && gy,
    DataType && g )
```

Ordinary feed backward pass of a neural network.

The backward pass multiplies the error with the specified constant scalar value.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.107.3.2 Delta() [1/2]

```
OutputDataType& Delta ( ) const [inline]
```

Get the delta.

Definition at line 69 of file multiply_constant.hpp.

39.107.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 71 of file multiply_constant.hpp.

References `MultiplyConstant< InputDataType, OutputDataType >::serialize()`.

39.107.3.4 Forward()

```
void Forward (
    const InputType && input,
    OutputType && output )
```

Ordinary feed forward pass of a neural network.

Multiply the input with the specified constant scalar value.

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.107.3.5 OutputParameter() [1/2]

```
OutputDataType& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 64 of file multiply_constant.hpp.

39.107.3.6 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 66 of file multiply_constant.hpp.

39.107.3.7 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by `MultiplyConstant< InputDataType, OutputDataType >::Delta()`.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **multiply_constant.hpp**

39.108 MultiplyMerge< InputDataType, OutputDataType, CustomLayers > Class Template Reference

Implementation of the **MultiplyMerge** (p. 829) module class.

Public Member Functions

- **MultiplyMerge** (const bool model=false, const bool run=true)
*Create the **MultiplyMerge** (p. 829) object using the specified parameters.*
- **~MultiplyMerge** ()
Destructor to release allocated memory.
- template<class LayerType , class... Args>
void **Add** (Args... args)
- void **Add** (**LayerTypes**< CustomLayers... > layer)
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f , using the results from the feed forward pass.
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename InputType , typename OutputType >
void **Forward** (InputType &&, OutputType &&output)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- template<typename eT >
void **Gradient** (arma::Mat< eT > &&input, arma::Mat< eT > &&error, arma::Mat< eT > &&gradient)
- OutputDataType const & **Gradient** () const
Get the gradient.
- OutputDataType & **Gradient** ()
Modify the gradient.
- std::vector< **LayerTypes**< CustomLayers... > > & **Model** ()
Return the model modules.
- OutputDataType const & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- OutputDataType const & **Parameters** () const
Get the parameters.
- OutputDataType & **Parameters** ()
Modify the parameters.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
- *Serialize the layer.*

39.108.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat, typename... CustomLayers>
class mlpack::ann::MultiplyMerge< InputDataType, OutputDataType, CustomLayers >
```

Implementation of the **MultiplyMerge** (p. 829) module class.

The **MultiplyMerge** (p. 829) class multiplies the output of various modules element-wise.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>CustomLayers</i>	Additional custom layers that can be added.

Definition at line 141 of file layer_types.hpp.

39.108.2 Constructor & Destructor Documentation

39.108.2.1 MultiplyMerge()

```
MultiplyMerge (
    const bool model = false,
    const bool run = true )
```

Create the **MultiplyMerge** (p. 829) object using the specified parameters.

Parameters

<i>model</i>	Expose all the network modules.
<i>run</i>	Call the Forward/Backward method before the output is merged.

39.108.2.2 ~MultiplyMerge()

```
~ MultiplyMerge ( )
```

Destructor to release allocated memory.

39.108.3 Member Function Documentation

39.108.3.1 Add() [1/2]

```
void Add (
    Args... args ) [inline]
```

Definition at line 98 of file multiply_merge.hpp.

39.108.3.2 Add() [2/2]

```
void Add (
    LayerTypes< CustomLayers... > layer ) [inline]
```

Definition at line 105 of file multiply_merge.hpp.

39.108.3.3 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f , using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.108.3.4 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 113 of file multiply_merge.hpp.

39.108.3.5 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 115 of file multiply_merge.hpp.

39.108.3.6 Forward()

```
void Forward (
    InputType && ,
    OutputType && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.108.3.7 Gradient() [1/3]

```
void Gradient (
    arma::Mat< eT > && input,
    arma::Mat< eT > && error,
    arma::Mat< eT > && gradient )
```

39.108.3.8 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 118 of file multiply_merge.hpp.

39.108.3.9 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 120 of file multiply_merge.hpp.

39.108.3.10 Model()

```
std::vector< LayerTypes<CustomLayers...> >& Model ( ) [inline]
```

Return the model modules.

Definition at line 123 of file multiply_merge.hpp.

39.108.3.11 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 108 of file multiply_merge.hpp.

39.108.3.12 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 110 of file multiply_merge.hpp.

39.108.3.13 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 134 of file multiply_merge.hpp.

39.108.3.14 Parameters() [2/2]

OutputDataType& Parameters () [inline]

Modify the parameters.

Definition at line 136 of file multiply_merge.hpp.

References MultiplyMerge< InputDataType, OutputDataType, CustomLayers >::serialize().

39.108.3.15 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by MultiplyMerge< InputDataType, OutputDataType, CustomLayers >::Parameters().

The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **layer_types.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **multiply_merge.hpp**

39.109 NaiveConvolution< BorderMode > Class Template Reference

Computes the two-dimensional convolution.

Static Public Member Functions

- template<typename eT, typename Border = BorderMode>
static std::enable_if< std::is_same< Border, **ValidConvolution** >::value, void >::type **Convolution** (const arma::Mat< eT > &input, const arma::Mat< eT > &filter, arma::Mat< eT > &output, const size_t dW=1, const size_t dH=1, const size_t dilationW=1, const size_t dilationH=1)
- template<typename eT, typename Border = BorderMode>
static std::enable_if< std::is_same< Border, **FullConvolution** >::value, void >::type **Convolution** (const arma::Mat< eT > &input, const arma::Mat< eT > &filter, arma::Mat< eT > &output, const size_t dW=1, const size_t dH=1, const size_t dilationW=1, const size_t dilationH=1)
- template<typename eT >
static void **Convolution** (const arma::Cube< eT > &input, const arma::Cube< eT > &filter, arma::Cube< eT > &output, const size_t dW=1, const size_t dH=1, const size_t dilationW=1, const size_t dilationH=1)
- template<typename eT >
static void **Convolution** (const arma::Mat< eT > &input, const arma::Cube< eT > &filter, arma::Cube< eT > &output, const size_t dW=1, const size_t dH=1, const size_t dilationW=1, const size_t dilationH=1)
- template<typename eT >
static void **Convolution** (const arma::Cube< eT > &input, const arma::Mat< eT > &filter, arma::Cube< eT > &output, const size_t dW=1, const size_t dH=1, const size_t dilationW=1, const size_t dilationH=1)

39.109.1 Detailed Description

```
template<typename BorderMode = FullConvolution>
class mlpack::ann::NaiveConvolution< BorderMode >
```

Computes the two-dimensional convolution.

This class allows specification of the type of the border type. The convolution can be compute with the valid border type of the full border type (default).

FullConvolution (p. 714): returns the full two-dimensional convolution. **ValidConvolution** (p. 967): returns only those parts of the convolution that are computed without the zero-padded edges.

Template Parameters

<i>BorderMode</i>	Type of the border mode (FullConvolution (p. 714) or ValidConvolution (p. 967)).
-------------------	---

Definition at line 35 of file naive_convolution.hpp.

39.109.2 Member Function Documentation

39.109.2.1 Convolution() [1/5]

```
static std::enable_if< std::is_same<Border, ValidConvolution>::value, void>::type Convolution
(
    const arma::Mat< eT > & input,
    const arma::Mat< eT > & filter,
    arma::Mat< eT > & output,
    const size_t dW = 1,
    const size_t dH = 1,
    const size_t dilationW = 1,
    const size_t dilationH = 1 ) [inline], [static]
```

Definition at line 52 of file naive_convolution.hpp.

Referenced by `SVDCConvolution< BorderMode >::Convolution()`, and `NaiveConvolution< BorderMode >::Convolution()`.

39.109.2.2 Convolution() [2/5]

```
static std::enable_if< std::is_same<Border, FullConvolution>::value, void>::type Convolution (
    const arma::Mat< eT > & input,
    const arma::Mat< eT > & filter,
    arma::Mat< eT > & output,
    const size_t dW = 1,
    const size_t dH = 1,
    const size_t dilationW = 1,
    const size_t dilationH = 1 ) [inline], [static]
```

Definition at line 98 of file `naive_convolution.hpp`.

References `NaiveConvolution< BorderMode >::Convolution()`.

39.109.2.3 Convolution() [3/5]

```
static void Convolution (
    const arma::Cube< eT > & input,
    const arma::Cube< eT > & filter,
    arma::Cube< eT > & output,
    const size_t dW = 1,
    const size_t dH = 1,
    const size_t dilationW = 1,
    const size_t dilationH = 1 ) [inline], [static]
```

Definition at line 151 of file `naive_convolution.hpp`.

References `NaiveConvolution< BorderMode >::Convolution()`.

39.109.2.4 Convolution() [4/5]

```
static void Convolution (
    const arma::Mat< eT > & input,
    const arma::Cube< eT > & filter,
    arma::Cube< eT > & output,
    const size_t dW = 1,
    const size_t dH = 1,
    const size_t dilationW = 1,
    const size_t dilationH = 1 ) [inline], [static]
```

Definition at line 187 of file `naive_convolution.hpp`.

References `NaiveConvolution< BorderMode >::Convolution()`.

39.109.2.5 Convolution() [5/5]

```
static void Convolution (
    const arma::Cube< eT > & input,
    const arma::Mat< eT > & filter,
    arma::Cube< eT > & output,
    const size_t dW = 1,
    const size_t dH = 1,
    const size_t dilationW = 1,
    const size_t dilationH = 1 ) [inline], [static]
```

Definition at line 223 of file naive_convolution.hpp.

References NaiveConvolution< BorderMode >::Convolution().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/ **naive_convolution.hpp**

39.110 NegativeLogLikelihood< InputDataType, OutputDataType > Class Template Reference

Implementation of the negative log likelihood layer.

Public Member Functions

- **NegativeLogLikelihood** ()
Create the NegativeLogLikelihoodLayer object.
- template<typename InputType , typename TargetType , typename OutputType >
void **Backward** (const InputType &&input, const TargetType &&target, OutputType &&output)
Ordinary feed backward pass of a neural network.
- OutputDataType & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename InputType , typename TargetType >
double **Forward** (const InputType &&input, TargetType &&target)
Computes the Negative log likelihood.
- InputDataType & **InputParameter** () const
Get the input parameter.
- InputDataType & **InputParameter** ()
Modify the input parameter.
- OutputDataType & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- template<typename Archive >
void **serialize** (Archive &, const unsigned int)
Serialize the layer.

39.110.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::NegativeLogLikelihood< InputDataType, OutputDataType >
```

Implementation of the negative log likelihood layer.

The negative log likelihood layer expects that the input contains log-probabilities for each class. The layer also expects a class index, in the range between 1 and the number of classes, as target when calling the Forward function.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 35 of file negative_log_likelihood.hpp.

39.110.2 Constructor & Destructor Documentation

39.110.2.1 NegativeLogLikelihood()

```
NegativeLogLikelihood ( )
```

Create the NegativeLogLikelihoodLayer object.

39.110.3 Member Function Documentation

39.110.3.1 Backward()

```
void Backward (
    const InputType && input,
    const TargetType && target,
    OutputType && output )
```

Ordinary feed backward pass of a neural network.

The negative log likelihood layer expects that the input contains log-probabilities for each class. The layer also expects a class index, in the range between 1 and the number of classes, as target when calling the Forward function.

Parameters

<i>input</i>	The propagated input activation.
<i>target</i>	The target vector, that contains the class index in the range between 1 and the number of classes.
<i>output</i>	The calculated error.

39.110.3.2 Delta() [1/2]

```
OutputDataType& Delta ( ) const [inline]
```

Get the delta.

Definition at line 80 of file `negative_log_likelihood.hpp`.

39.110.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 82 of file `negative_log_likelihood.hpp`.

References `NegativeLogLikelihood< InputDataType, OutputDataType >::serialize()`.

39.110.3.4 Forward()

```
double Forward (
    const InputType && input,
    TargetType && target )
```

Computes the Negative log likelihood.

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>target</i>	The target vector, that contains the class index in the range between 1 and the number of classes.

39.110.3.5 InputParameter() [1/2]

```
InputDataType& InputParameter ( ) const [inline]
```

Get the input parameter.

Definition at line 70 of file `negative_log_likelihood.hpp`.

39.110.3.6 InputParameter() [2/2]

```
InputDataType& InputParameter ( ) [inline]
```

Modify the input parameter.

Definition at line 72 of file `negative_log_likelihood.hpp`.

39.110.3.7 OutputParameter() [1/2]

```
OutputDataType& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 75 of file `negative_log_likelihood.hpp`.

39.110.3.8 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 77 of file `negative_log_likelihood.hpp`.

39.110.3.9 serialize()

```
void serialize (
    Archive & ,
    const unsigned int )
```

Serialize the layer.

Referenced by `NegativeLogLikelihood< InputDataType, OutputDataType >::Delta()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/ negative_log_likelihood.hpp`

39.111 NetworkInitialization< InitializationRuleType, CustomLayers > Class Template Reference

This class is used to initialize the network with the given initialization rule.

Public Member Functions

- **NetworkInitialization** (const InitializationRuleType &initializeRule=InitializationRuleType())
Use the given initialization rule to initialize the specified network.
- void **Initialize** (const std::vector< **LayerTypes**< CustomLayers... > > &network, arma::mat ¶meter, size_t parameterOffset=0)
Initialize the specified network and store the results in the given parameter.

39.111.1 Detailed Description

```
template<typename InitializationRuleType, typename... CustomLayers>
class mlpack::ann::NetworkInitialization< InitializationRuleType, CustomLayers >
```

This class is used to initialize the network with the given initialization rule.

Definition at line 33 of file network_init.hpp.

39.111.2 Constructor & Destructor Documentation

39.111.2.1 NetworkInitialization()

```
NetworkInitialization (
    const InitializationRuleType & initializeRule = InitializationRuleType() ) [inline]
```

Use the given initialization rule to initialize the specified network.

Parameters

<i>initializeRule</i>	Rule to initialize the given network.
-----------------------	---------------------------------------

Definition at line 41 of file network_init.hpp.

39.111.3 Member Function Documentation

39.111.3.1 Initialize()

```
void Initialize (
    const std::vector< LayerTypes< CustomLayers... > > & network,
    arma::mat & parameter,
    size_t parameterOffset = 0 ) [inline]
```

Initialize the specified network and store the results in the given parameter.

Parameters

<i>network</i>	Network that should be initialized.
<i>parameter</i>	The network parameter.

Definition at line 55 of file network_init.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ **network_init.hpp**

39.112 NguyenWidrowInitialization Class Reference

This class is used to initialize the weight matrix with the Nguyen-Widrow method.

Public Member Functions

- **NguyenWidrowInitialization** (const double lowerBound=-0.5, const double upperBound=0.5)
Initialize the random initialization rule with the given lower bound and upper bound.
- template<typename eT >
void **Initialize** (arma::Mat< eT > &W, const size_t rows, const size_t cols)
Initialize the elements of the specified weight matrix with the Nguyen-Widrow method.
- template<typename eT >
void **Initialize** (arma::Cube< eT > &W, const size_t rows, const size_t cols, const size_t slices)
Initialize the elements of the specified weight 3rd order tensor with the Nguyen-Widrow method.

39.112.1 Detailed Description

This class is used to initialize the weight matrix with the Nguyen-Widrow method.

The method is defined by

$$\begin{aligned} \gamma &\leq w_i \leq \gamma \\ \beta &= 0.7H^{\frac{1}{2}} \\ n &= \sqrt{\sum_{i=0} Iw_i^2} \\ w_i &= \frac{\beta w_i}{n} \end{aligned}$$

Where H is the number of neurons in the outgoing layer, I represents the number of neurons in the ingoing layer and gamma defines the random interval that is used to initialize the weights with a random value in a specific range.

Definition at line 53 of file `nguyen_widrow_init.hpp`.

39.112.2 Constructor & Destructor Documentation

39.112.2.1 NguyenWidrowInitialization()

```
NguyenWidrowInitialization (
    const double lowerBound = -0.5,
    const double upperBound = 0.5 ) [inline]
```

Initialize the random initialization rule with the given lower bound and upper bound.

Parameters

<i>lowerBound</i>	The number used as lower bound.
<i>upperBound</i>	The number used as upper bound.

Definition at line 63 of file `nguyen_widrow_init.hpp`.

39.112.3 Member Function Documentation

39.112.3.1 Initialize() [1/2]

```
void Initialize (
    arma::Mat< eT > & W,
    const size_t rows,
    const size_t cols ) [inline]
```

Initialize the elements of the specified weight matrix with the Nguyen-Widrow method.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.

Definition at line 76 of file `nguyen_widrow_init.hpp`.

References `RandomInitialization::Initialize()`.

Referenced by `NguyenWidrowInitialization::Initialize()`.

39.112.3.2 Initialize() [2/2]

```
void Initialize (
    arma::Cube< eT > & W,
    const size_t rows,
    const size_t cols,
    const size_t slices ) [inline]
```

Initialize the elements of the specified weight 3rd order tensor with the Nguyen-Widrow method.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.
<i>slices</i>	Number of slices.

Definition at line 95 of file `nguyen_widrow_init.hpp`.

References `NguyenWidrowInitialization::Initialize()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ nguyen_widrow_init.hpp`

39.113 OivsInitialization< ActivationFunction > Class Template Reference

This class is used to initialize the weight matrix with the oivs method.

Public Member Functions

- **OivsInitialization** (const double epsilon=0.1, const int k=5, const double gamma=0.9)
Initialize the random initialization rule with the given values.
- template<typename eT >
void **Initialize** (arma::Mat< eT > &W, const size_t rows, const size_t cols)
Initialize the elements of the specified weight matrix with the oivs method.
- template<typename eT >
void **Initialize** (arma::Cube< eT > &W, const size_t rows, const size_t cols, const size_t slices)
Initialize the elements of the specified weight 3rd order tensor with the oivs method.

39.113.1 Detailed Description

```
template<class ActivationFunction = LogisticFunction>
class mlpack::ann::OivsInitialization< ActivationFunction >
```

This class is used to initialize the weight matrix with the oivs method.

The method is based on the equations representing the characteristics of the information transformation mechanism of a node. The method is defined by

$$\begin{aligned}
 b &= |F^{-1}(1 - \epsilon) - f^{-1}(\epsilon)| \\
 \hat{w} &= \frac{b}{k \cdot n} \\
 \gamma &\leq a_i \leq \gamma \\
 w_i &= \hat{w} \cdot \sqrt{a_i + 1}
 \end{aligned}$$

Where f is the transfer function epsilon, k custom parameters, n the number of neurons in the outgoing layer and gamma a parameter that defines the random interval.

Template Parameters

<i>ActivationFunction</i>	The activation function used for the oivs method.
---------------------------	---

Definition at line 59 of file oivs_init.hpp.

39.113.2 Constructor & Destructor Documentation

39.113.2.1 OivsInitialization()

```
OivsInitialization (
    const double epsilon = 0.1,
```

```
const int k = 5,
const double gamma = 0.9 ) [inline]
```

Initialize the random initialization rule with the given values.

Parameters

<i>epsilon</i>	Parameter to control the activation region.
<i>k</i>	Parameter to control the activation region width.
<i>gamma</i>	Parameter to define the uniform random range.

Definition at line 69 of file oivs_init.hpp.

39.113.3 Member Function Documentation

39.113.3.1 Initialize() [1/2]

```
void Initialize (
    arma::Mat< eT > & W,
    const size_t rows,
    const size_t cols ) [inline]
```

Initialize the elements of the specified weight matrix with the oivs method.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.

Definition at line 86 of file oivs_init.hpp.

References RandomInitialization::Initialize().

Referenced by OivsInitialization< ActivationFunction >::Initialize().

39.113.3.2 Initialize() [2/2]

```
void Initialize (
    arma::Cube< eT > & W,
    const size_t rows,
```

```
const size_t cols,  
const size_t slices ) [inline]
```

Initialize the elements of the specified weight 3rd order tensor with the oivs method.

Parameters

<i>W</i>	3rd order tensor to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.
<i>slices</i>	Number of slices.

Definition at line 104 of file oivs_init.hpp.

References `OivsInitialization< ActivationFunction >::Initialize()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ oivs_init.hpp`

39.114 OrthogonalInitialization Class Reference

This class is used to initialize the weight matrix with the orthogonal matrix initialization.

Public Member Functions

- **OrthogonalInitialization** (const double gain=1.0)
Initialize the orthogonal matrix initialization rule with the given gain.
- `template<typename eT >`
`void Initialize (arma::Mat< eT > &W, const size_t rows, const size_t cols)`
Initialize the elements of the specified weight matrix with the orthogonal matrix initialization method.
- `template<typename eT >`
`void Initialize (arma::Cube< eT > &W, const size_t rows, const size_t cols, const size_t slices)`
Initialize the elements of the specified weight 3rd order tensor with the orthogonal matrix initialization method.

39.114.1 Detailed Description

This class is used to initialize the weight matrix with the orthogonal matrix initialization.

Definition at line 24 of file orthogonal_init.hpp.

39.114.2 Constructor & Destructor Documentation

39.114.2.1 OrthogonalInitialization()

```
OrthogonalInitialization (  
    const double gain = 1.0 ) [inline]
```

Initialize the orthogonal matrix initialization rule with the given gain.

Parameters

<i>gain</i>	The gain value.
-------------	-----------------

Definition at line 32 of file orthogonal_init.hpp.

39.114.3 Member Function Documentation

39.114.3.1 Initialize() [1/2]

```
void Initialize (
    arma::Mat< eT > & W,
    const size_t rows,
    const size_t cols ) [inline]
```

Initialize the elements of the specified weight matrix with the orthogonal matrix initialization method.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.

Definition at line 43 of file orthogonal_init.hpp.

Referenced by OrthogonalInitialization::Initialize().

39.114.3.2 Initialize() [2/2]

```
void Initialize (
    arma::Cube< eT > & W,
    const size_t rows,
    const size_t cols,
    const size_t slices ) [inline]
```

Initialize the elements of the specified weight 3rd order tensor with the orthogonal matrix initialization method.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.
<i>slices</i>	Number of slices.

Definition at line 62 of file orthogonal_init.hpp.

References OrthogonalInitialization::Initialize().

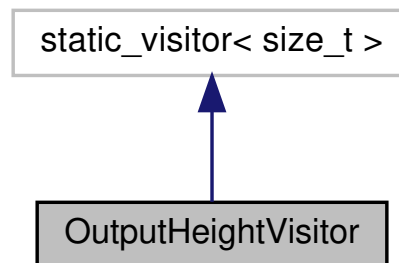
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ **orthogonal_init.hpp**

39.115 OutputHeightVisitor Class Reference

OutputHeightVisitor (p. 850) exposes the OutputHeight() method of the given module.

Inheritance diagram for OutputHeightVisitor:



Public Member Functions

- template<typename LayerType >
size_t **operator()** (LayerType *layer) const
Return the output height.

39.115.1 Detailed Description

OutputHeightVisitor (p. 850) exposes the OutputHeight() method of the given module.

Definition at line 27 of file output_height_visitor.hpp.

39.115.2 Member Function Documentation

39.115.2.1 operator()

```
size_t operator() (
    LayerType * layer ) const
```

Return the output height.

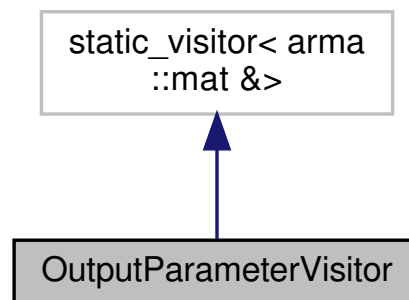
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **output_height_visitor.hpp**

39.116 OutputParameterVisitor Class Reference

OutputParameterVisitor (p. 851) exposes the output parameter of the given module.

Inheritance diagram for OutputParameterVisitor:



Public Member Functions

- template<typename LayerType >
arma::mat & **operator()** (LayerType *layer) const
Return the output parameter set.

39.116.1 Detailed Description

OutputParameterVisitor (p. 851) exposes the output parameter of the given module.

Definition at line 27 of file output_parameter_visitor.hpp.

39.116.2 Member Function Documentation

39.116.2.1 operator()

```
arma::mat& operator() (
    LayerType * layer ) const
```

Return the output parameter set.

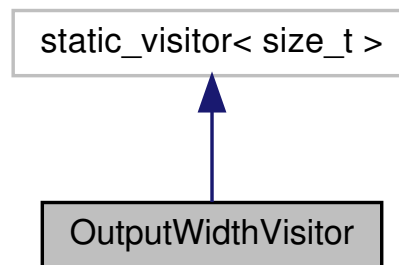
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **output_parameter_visitor.hpp**

39.117 OutputWidthVisitor Class Reference

OutputWidthVisitor (p. 852) exposes the OutputWidth() method of the given module.

Inheritance diagram for OutputWidthVisitor:



Public Member Functions

- template<typename LayerType >
size_t **operator()** (LayerType *layer) const
Return the output width.

39.117.1 Detailed Description

OutputWidthVisitor (p. 852) exposes the OutputWidth() method of the given module.

Definition at line 27 of file output_width_visitor.hpp.

39.117.2 Member Function Documentation

39.117.2.1 operator()()

```
size_t operator() (
    LayerType * layer ) const
```

Return the output width.

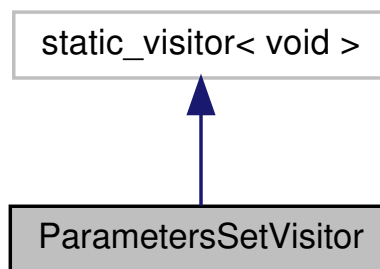
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **output_width_visitor.hpp**

39.118 ParametersSetVisitor Class Reference

ParametersSetVisitor (p. 853) update the parameters set using the given matrix.

Inheritance diagram for ParametersSetVisitor:



Public Member Functions

- **ParametersSetVisitor** (arma::mat &¶meters)
Update the parameters set given the parameters matrix.
- template<typename LayerType >
void **operator()** (LayerType *layer) const
Update the parameters set.

39.118.1 Detailed Description

ParametersSetVisitor (p. 853) update the parameters set using the given matrix.

Definition at line 27 of file parameters_set_visitor.hpp.

39.118.2 Constructor & Destructor Documentation

39.118.2.1 ParametersSetVisitor()

```
ParametersSetVisitor (
    arma::mat && parameters )
```

Update the parameters set given the parameters matrix.

39.118.3 Member Function Documentation

39.118.3.1 operator()()

```
void operator() (
    LayerType * layer ) const
```

Update the parameters set.

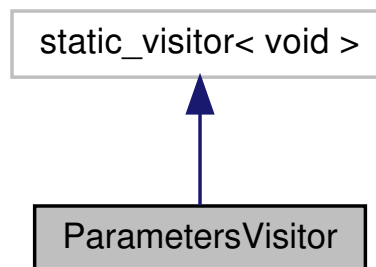
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **parameters_set_visitor.hpp**

39.119 ParametersVisitor Class Reference

ParametersVisitor (p. 854) exposes the parameters set of the given module and stores the parameters set into the given matrix.

Inheritance diagram for ParametersVisitor:



Public Member Functions

- **ParametersVisitor** (arma::mat &¶meters)
Store the parameters set into the given parameters matrix.
- template<typename LayerType >
 void **operator()** (LayerType *layer) const
Set the parameters set.

39.119.1 Detailed Description

ParametersVisitor (p. 854) exposes the parameters set of the given module and stores the parameters set into the given matrix.

Definition at line 28 of file parameters_visitor.hpp.

39.119.2 Constructor & Destructor Documentation

39.119.2.1 ParametersVisitor()

```
ParametersVisitor (  
    arma::mat && parameters )
```

Store the parameters set into the given parameters matrix.

39.119.3 Member Function Documentation

39.119.3.1 operator>()

```
void operator() (  
    LayerType * layer ) const
```

Set the parameters set.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **parameters_visitor.hpp**

39.120 PReLU< InputDataType, OutputDataType > Class Template Reference

The **PReLU** (p. 855) activation function, defined by (where alpha is trainable)

Public Member Functions

- **PReLU** (const double userAlpha=0.03)
*Create the **PReLU** (p. 855) object using the specified parameters.*
 - double const & **Alpha** () const
Get the non zero gradient.
 - double & **Alpha** ()
Modify the non zero gradient.
 - template<typename DataType >
void **Backward** (const DataType &&input, DataType &&gy, DataType &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
 - OutputDataType const & **Delta** () const
Get the delta.
 - OutputDataType & **Delta** ()
Modify the delta.
 - template<typename InputType , typename OutputType >
void **Forward** (const InputType &&input, OutputType &&output)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
 - template<typename eT >
void **Gradient** (const arma::Mat< eT > &&input, arma::Mat< eT > &&error, arma::Mat< eT > &&gradient)
Calculate the gradient using the output delta and the input activation.
 - OutputDataType const & **Gradient** () const
Get the gradient.
 - OutputDataType & **Gradient** ()
Modify the gradient.
 - OutputDataType const & **OutputParameter** () const
Get the output parameter.
 - OutputDataType & **OutputParameter** ()
Modify the output parameter.
 - OutputDataType const & **Parameters** () const
Get the parameters.
 - OutputDataType & **Parameters** ()
Modify the parameters.
 - void **Reset** ()
 - template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
- Serialize the layer.*

39.120.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::PReLU< InputDataType, OutputDataType >
```

The **PReLU** (p. 855) activation function, defined by (where alpha is trainable)

$$f(x) = \max(x, \alpha * x)$$

$$f'(x) = \begin{cases} 1 & : x > 0 \\ \alpha & : x \leq 0 \end{cases}$$

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 45 of file parametric_relu.hpp.

39.120.2 Constructor & Destructor Documentation

39.120.2.1 PReLU()

```
PReLU (  
    const double userAlpha = 0.03 )
```

Create the **PReLU** (p. 855) object using the specified parameters.

The non zero gradient can be adjusted by specifying the parameter alpha in the range 0 to 1. Default (alpha = 0.03). This parameter is trainable.

Parameters

<i>alpha</i>	Non zero gradient
--------------	-------------------

39.120.3 Member Function Documentation

39.120.3.1 Alpha() [1/2]

```
double const& Alpha ( ) const [inline]
```

Get the non zero gradient.

Definition at line 118 of file parametric_relu.hpp.

39.120.3.2 Alpha() [2/2]

```
double& Alpha ( ) [inline]
```

Modify the non zero gradient.

Definition at line 120 of file parametric_relu.hpp.

References PReLU< InputDataType, OutputDataType >::serialize().

39.120.3.3 Backward()

```
void Backward (
    const DataType && input,
    DataType && gy,
    DataType && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.120.3.4 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 108 of file parametric_relu.hpp.

39.120.3.5 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 110 of file parametric_relu.hpp.

39.120.3.6 Forward()

```
void Forward (
    const InputType && input,
    OutputType && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.120.3.7 Gradient() [1/3]

```
void Gradient (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && error,
    arma::Mat< eT > && gradient )
```

Calculate the gradient using the output delta and the input activation.

Parameters

<i>input</i>	The input parameter used for calculating the gradient.
<i>error</i>	The calculated error.
<i>gradient</i>	The calculated gradient.

39.120.3.8 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 113 of file parametric_relu.hpp.

39.120.3.9 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 115 of file parametric_relu.hpp.

39.120.3.10 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 103 of file parametric_relu.hpp.

39.120.3.11 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 105 of file parametric_relu.hpp.

39.120.3.12 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 98 of file parametric_relu.hpp.

39.120.3.13 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 100 of file parametric_relu.hpp.

39.120.3.14 Reset()

```
void Reset ( )
```

39.120.3.15 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by PReLU< InputDataType, OutputDataType >::Alpha().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **parametric_relu.hpp**

39.121 RandomInitialization Class Reference

This class is used to initialize randomly the weight matrix.

Public Member Functions

- **RandomInitialization** (const double lowerBound=-1, const double upperBound=1)
Initialize the random initialization rule with the given lower bound and upper bound.
- **RandomInitialization** (const double bound)
Initialize the random initialization rule with the given bound.
- template<typename eT >
void **Initialize** (arma::Mat< eT > &W, const size_t rows, const size_t cols)
Initialize randomly the elements of the specified weight matrix.
- template<typename eT >
void **Initialize** (arma::Cube< eT > &W, const size_t rows, const size_t cols, const size_t slices)
Initialize randomly the elements of the specified weight 3rd order tensor.

39.121.1 Detailed Description

This class is used to initialize randomly the weight matrix.

Definition at line 24 of file random_init.hpp.

39.121.2 Constructor & Destructor Documentation

39.121.2.1 RandomInitialization() [1/2]

```
RandomInitialization (  
    const double lowerBound = -1,  
    const double upperBound = 1 ) [inline]
```

Initialize the random initialization rule with the given lower bound and upper bound.

Parameters

<i>lowerBound</i>	The number used as lower bound.
<i>upperBound</i>	The number used as upper bound.

Definition at line 34 of file random_init.hpp.

39.121.2.2 RandomInitialization() [2/2]

```
RandomInitialization (
    const double bound ) [inline]
```

Initialize the random initialization rule with the given bound.

Using the negative of the bound as lower bound and the positive bound as upper bound.

Parameters

<i>bound</i>	The number used as lower bound
--------------	--------------------------------

Definition at line 45 of file random_init.hpp.

39.121.3 Member Function Documentation

39.121.3.1 Initialize() [1/2]

```
void Initialize (
    arma::Mat< eT > & W,
    const size_t rows,
    const size_t cols ) [inline]
```

Initialize randomly the elements of the specified weight matrix.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.

Definition at line 56 of file random_init.hpp.

Referenced by `RandomInitialization::Initialize()`, `NguyenWidrowInitialization::Initialize()`, `KathirvalavakumarSubavathiInitialization::Initialize()`, `OivsInitialization< ActivationFunction >::Initialize()`, `GlorotInitializationType< Uniform >::Initialize()`, and `JacobianTest()`.

39.121.3.2 Initialize() [2/2]

```
void Initialize (
    arma::Cube< eT > & W,
    const size_t rows,
    const size_t cols,
    const size_t slices ) [inline]
```

Initialize randomly the elements of the specified weight 3rd order tensor.

Parameters

<i>W</i>	Weight matrix to initialize.
<i>rows</i>	Number of rows.
<i>cols</i>	Number of columns.

Definition at line 70 of file `random_init.hpp`.

References `RandomInitialization::Initialize()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/ random_init.hpp`

39.122 RBM< InitializationRuleType, DataType, PolicyType > Class Template Reference

The implementation of the **RBM** (p. 864) module.

Public Types

- `typedef DataType::elem_type ElemType`
- `using NetworkType = RBM< InitializationRuleType, DataType, PolicyType >`

Public Member Functions

- **RBM** (arma::Mat< **ElemType** > predictors, InitializationRuleType initializeRule, const size_t visibleSize, const size_t hiddenSize, const size_t batchSize=1, const size_t numSteps=1, const size_t negSteps=1, const size_t poolSize=2, const **ElemType** slabPenalty=8, const **ElemType** radius=1, const bool persistence=false)
Initialize all the parameters of the network using initializeRule.
- double **Evaluate** (const arma::Mat< **ElemType** > ¶meters, const size_t i, const size_t batchSize)
*Evaluate the **RBM** (p. 864) network with the given parameters.*
- template<typename Policy = PolicyType>
std::enable_if< std::is_same< Policy, **BinaryRBM** >::value, double >::type **FreeEnergy** (arma::Mat< **ElemType** > &&input)
*This function calculates the free energy of the **BinaryRBM** (p. 609).*
- template<typename Policy = PolicyType>
std::enable_if< std::is_same< Policy, **SpikeSlabRBM** >::value, double >::type **FreeEnergy** (arma::Mat< **ElemType** > &&input)
*This function calculates the free energy of the **SpikeSlabRBM** (p. 947).*
- void **Gibbs** (arma::Mat< **ElemType** > &&input, arma::Mat< **ElemType** > &&output, const size_t steps=SIZE_MAX)
This function does the k-step Gibbs Sampling.
- void **Gradient** (const arma::Mat< **ElemType** > ¶meters, const size_t i, arma::Mat< **ElemType** > &gradient, const size_t batchSize)
*Calculates the gradients for the **RBM** (p. 864) network.*
- DataType const & **HiddenBias** () const
Return the hidden bias of the network.
- DataType & **HiddenBias** ()
Modify the hidden bias of the network.
- template<typename Policy = PolicyType>
std::enable_if< std::is_same< Policy, **BinaryRBM** >::value, void >::type **HiddenMean** (DataType &&input, DataType &&output)
The function calculates the mean for the hidden layer.
- template<typename Policy = PolicyType>
std::enable_if< std::is_same< Policy, **SpikeSlabRBM** >::value, void >::type **HiddenMean** (DataType &&input, DataType &&output)
The function calculates the mean of the Normal distribution of $P(s|v, h)$.
- size_t const & **HiddenSize** () const
Get the hidden size.
- size_t **NumFunctions** () const
Return the number of separable functions (the number of predictor points).
- size_t **NumSteps** () const
Return the number of steps of Gibbs Sampling.
- const arma::Mat< **ElemType** > & **Parameters** () const
Return the parameters of the network.
- arma::Mat< **ElemType** > & **Parameters** ()
Modify the parameters of the network.
- template<typename Policy = PolicyType>
std::enable_if< std::is_same< Policy, **BinaryRBM** >::value, void >::type **Phase** (DataType &&input, DataType &&gradient)
*Calculates the gradient of the **RBM** (p. 864) network on the provided input.*

- `template<typename Policy = PolicyType>`
`std::enable_if< std::is_same< Policy, SpikeSlabRBM >::value, void >::type Phase (DataType &&input, Data←`
`Type &&gradient)`
*Calculates the gradient of the **RBM** (p. 864) network on the provided input.*
- `size_t const & PoolSize () const`
Get the pool size.
- `template<typename Policy = PolicyType>`
`std::enable_if< std::is_same< Policy, BinaryRBM >::value, void >::type Reset ()`
- `template<typename Policy = PolicyType>`
`std::enable_if< std::is_same< Policy, SpikeSlabRBM >::value, void >::type Reset ()`
- `template<typename Policy = PolicyType>`
`std::enable_if< std::is_same< Policy, BinaryRBM >::value, void >::type SampleHidden (arma::Mat< Elem←`
`Type > &&input, arma::Mat< ElemType > &&output)`
This function samples the hidden layer given the visible layer using Bernoulli function.
- `template<typename Policy = PolicyType>`
`std::enable_if< std::is_same< Policy, SpikeSlabRBM >::value, void >::type SampleHidden (arma::Mat<`
`ElemType > &&input, arma::Mat< ElemType > &&output)`
*This function samples the slab outputs from the Normal distribution with mean given by: $h_i \wedge \{-1\} * W_i^T * v$ and variance: σ_i^2 .*
- `template<typename Policy = PolicyType>`
`std::enable_if< std::is_same< Policy, SpikeSlabRBM >::value, void >::type SampleSlab (DataType &&slab←`
`Mean, DataType &&slab)`
*The function samples from the Normal distribution of $P(s|v, h)$, where the mean is given by: $h_i \wedge \{-1\} * W_i^T * v$ and variance is given by: σ_i^2 .*
- `template<typename Policy = PolicyType>`
`std::enable_if< std::is_same< Policy, SpikeSlabRBM >::value, void >::type SampleSpike (DataType`
`&&spikeMean, DataType &&spike)`
The function samples the spike function using Bernoulli distribution.
- `template<typename Policy = PolicyType>`
`std::enable_if< std::is_same< Policy, BinaryRBM >::value, void >::type SampleVisible (arma::Mat< Elem←`
`Type > &&input, arma::Mat< ElemType > &&output)`
This function samples the visible layer given the hidden layer using Bernoulli function.
- `template<typename Policy = PolicyType>`
`std::enable_if< std::is_same< Policy, SpikeSlabRBM >::value, void >::type SampleVisible (arma::Mat<`
`ElemType > &&input, arma::Mat< ElemType > &&output)`
*Sample Hidden function samples the slab outputs from the Normal distribution with mean given by: $h_i \wedge \{-1\} * W_i^T * v$ and variance: σ_i^2 .*
- `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`
Serialize the model.
- `void Shuffle ()`
Shuffle the order of function visitation.
- `template<typename Policy = PolicyType>`
`std::enable_if< std::is_same< Policy, SpikeSlabRBM >::value, void >::type SlabMean (DataType &&visible,`
`DataType &&spike, DataType &&slabMean)`
*The function calculates the mean of Normal distribution of $P(s|v, h)$, where the mean is given by: $h_i \wedge \{-1\} * W_i^T * v$.*
- `ElemType const & SlabPenalty () const`
Get the regularizer associated with slab variables.
- `DataType const & SpikeBias () const`
Get the regularizer associated with spike variables.

- **DataType & SpikeBias ()**
Modify the regularizer associated with spike variables.
- `template<typename Policy = PolicyType>`
`std::enable_if< std::is_same< Policy, SpikeSlabRBM >::value, void >::type SpikeMean (DataType &&visible,`
`DataType &&spikeMean)`
The function calculates the mean of the distribution $P(h|v)$, where mean is given by: $\sum_i (v^T W_i - 1) W_i^T v + b_i$.
- `template<typename OptimizerType >`
`double Train (OptimizerType &optimizer)`
*Train the **RBM** (p. 864) on the given input data.*
- **DataType const & VisibleBias () const**
Return the visible bias of the network.
- **DataType & VisibleBias ()**
Modify the visible bias of the network.
- `template<typename Policy = PolicyType>`
`std::enable_if< std::is_same< Policy, BinaryRBM >::value, void >::type VisibleMean (DataType &&input,`
`DataType &&output)`
The function calculates the mean for the visible layer.
- `template<typename Policy = PolicyType>`
`std::enable_if< std::is_same< Policy, SpikeSlabRBM >::value, void >::type VisibleMean (DataType &&input,`
`DataType &&output)`
The function calculates the mean of the Normal distribution of $P(v|s, h)$.
- **DataType const & VisiblePenalty () const**
Get the regularizer associated with visible variables.
- **DataType & VisiblePenalty ()**
Modify the regularizer associated with visible variables.
- **size_t const & VisibleSize () const**
Get the visible size.
- `arma::Cube< ElemType > const & Weight () const`
Get the weights of the network.
- `arma::Cube< ElemType > & Weight ()`
Modify the weights of the network.

39.122.1 Detailed Description

```
template<typename InitializationRuleType, typename DataType = arma::mat, typename PolicyType = BinaryRBM>
class mlpack::ann::RBM< InitializationRuleType, DataType, PolicyType >
```

The implementation of the **RBM** (p. 864) module.

A Restricted Boltzmann Machines (**RBM** (p. 864)) is a generative stochastic artificial neural network that can learn a probability distribution over its set of inputs. RBMs have found applications in dimensionality reduction, classification, collaborative filtering, feature learning and topic modelling. They can be trained in either supervised or unsupervised ways, depending on the task. They are a variant of Boltzmann machines, with the restriction that the neurons must form a bipartite graph.

Template Parameters

<i>InitializationRuleType</i>	Rule used to initialize the network.
<i>DataType</i>	The type of matrix to be used.
<i>PolicyType</i>	The RBM (p. 864) variant to be used (BinaryRBM (p. 609) or SpikeSlabRBM (p. 947)).

Definition at line 38 of file `rbm.hpp`.

39.122.2 Member Typedef Documentation

39.122.2.1 ElemType

```
typedef DataType::elem_type ElemType
```

Definition at line 42 of file `rbm.hpp`.

39.122.2.2 NetworkType

```
using NetworkType = RBM<InitializationRuleType, DataType, PolicyType>
```

Definition at line 41 of file `rbm.hpp`.

39.122.3 Constructor & Destructor Documentation

39.122.3.1 RBM()

```
RBM (
    arma::Mat< ElemType > predictors,
    InitializationRuleType initializeRule,
    const size_t visibleSize,
    const size_t hiddenSize,
    const size_t batchSize = 1,
    const size_t numSteps = 1,
    const size_t negSteps = 1,
    const size_t poolSize = 2,
    const ElemType slabPenalty = 8,
    const ElemType radius = 1,
    const bool persistence = false )
```

Initialize all the parameters of the network using `initializeRule`.

Parameters

<i>predictors</i>	Training data to be used.
<i>initializeRule</i>	InitializationRule object for initializing the network parameter.
<i>visibleSize</i>	Number of visible neurons.
<i>hiddenSize</i>	Number of hidden neurons.
<i>batchSize</i>	Batch size to be used for training.
<i>numSteps</i>	Number of Gibbs Sampling steps.
<i>negSteps</i>	Number of negative samples to average negative gradient.
<i>poolSize</i>	Number of hidden neurons to pool together.
<i>slabPenalty</i>	Regulariser of slab variables.
<i>radius</i>	Feasible regions for visible layer samples.
<i>persistence</i>	Indicates whether to use Persistent CD or not.

39.122.4 Member Function Documentation

39.122.4.1 Evaluate()

```
double Evaluate (
    const arma::Mat< ElemType > & parameters,
    const size_t i,
    const size_t batchSize )
```

Evaluate the **RB**M (p. 864) network with the given parameters.

The function is needed for monitoring the progress of the network.

Parameters

<i>parameters</i>	Matrix model parameters.
<i>i</i>	Index of the data point.
<i>batchSize</i>	Variable to store the present number of inputs.

39.122.4.2 FreeEnergy() [1/2]

```
std::enable_if<std::is_same<Policy, BinaryRBM>::value, double>::type FreeEnergy (
    arma::Mat< ElemType > && input )
```

This function calculates the free energy of the **BinaryRBM** (p. 609).

The free energy is given by: $-b^T v - \sum_{i=1}^M \log(1 + e^{c_j + v^T W_j})$.

Parameters

<i>input</i>	The visible neurons.
--------------	----------------------

39.122.4.3 FreeEnergy() [2/2]

```
std::enable_if<std::is_same<Policy, SpikeSlabRBM>::value, double>::type FreeEnergy (
    arma::Mat< ElemType > && input )
```

This function calculates the free energy of the **SpikeSlabRBM** (p. 947).

The free energy is given by:
$$- \sum_{i=1}^N \left(\sum_{m=1}^K \log \left(\sum_{j=1}^M \exp \left(w_{ij} + b_i + \sum_{m=1}^K a_m \cdot \text{sign}(w_{ij}) \right) \right) \right)$$

Parameters

<i>input</i>	The visible layer neurons.
--------------	----------------------------

39.122.4.4 Gibbs()

```
void Gibbs (
    arma::Mat< ElemType > && input,
    arma::Mat< ElemType > && output,
    const size_t steps = SIZE_MAX )
```

This function does the k-step Gibbs Sampling.

Parameters

<i>input</i>	Input to the Gibbs function.
<i>output</i>	Used for storing the negative sample.
<i>steps</i>	Number of Gibbs Sampling steps taken.

39.122.4.5 Gradient()

```
void Gradient (
    const arma::Mat< ElemType > & parameters,
    const size_t i,
```

```
arma::Mat< ElemType > & gradient,
const size_t batchSize )
```

Calculates the gradients for the **RBM** (p. 864) network.

Parameters

<i>parameters</i>	The current parameters of the network.
<i>i</i>	Index of the data point.
<i>gradient</i>	Variable to store the present gradient.
<i>batchSize</i>	Variable to store the present number of inputs.

39.122.4.6 HiddenBias() [1/2]

```
DataType const& HiddenBias ( ) const [inline]
```

Return the hidden bias of the network.

Definition at line 349 of file rbm.hpp.

39.122.4.7 HiddenBias() [2/2]

```
DataType& HiddenBias ( ) [inline]
```

Modify the hidden bias of the network.

Definition at line 351 of file rbm.hpp.

39.122.4.8 HiddenMean() [1/2]

```
std::enable_if<std::is_same<Policy, BinaryRBM>::value, void>::type HiddenMean (
    DataType && input,
    DataType && output )
```

The function calculates the mean for the hidden layer.

Parameters

<i>input</i>	Visible neurons.
<i>output</i>	Hidden neuron activations.

39.122.4.9 HiddenMean() [2/2]

```
std::enable_if<std::is_same<Policy, SpikeSlabRBM>::value, void>::type HiddenMean (
    DataType && input,
    DataType && output )
```

The function calculates the mean of the Normal distribution of $P(s|v, h)$.

The mean is given by: $h_i \cdot \{-1\} \cdot W_i^T \cdot v$ The variance is given by: $\{-1\}$

Parameters

<i>input</i>	Visible layer neurons.
<i>output</i>	Consists of both the spike samples and slab samples.

39.122.4.10 HiddenSize()

```
size_t const& HiddenSize ( ) const [inline]
```

Get the hidden size.

Definition at line 369 of file rbm.hpp.

39.122.4.11 NumFunctions()

```
size_t NumFunctions ( ) const [inline]
```

Return the number of separable functions (the number of predictor points).

Definition at line 328 of file rbm.hpp.

39.122.4.12 NumSteps()

```
size_t NumSteps ( ) const [inline]
```

Return the number of steps of Gibbs Sampling.

Definition at line 331 of file rbm.hpp.

39.122.4.13 Parameters() [1/2]

```
const arma::Mat< ElemType>& Parameters ( ) const [inline]
```

Return the parameters of the network.

Definition at line 334 of file rbm.hpp.

39.122.4.14 Parameters() [2/2]

```
arma::Mat< ElemType>& Parameters ( ) [inline]
```

Modify the parameters of the network.

Definition at line 336 of file rbm.hpp.

39.122.4.15 Phase() [1/2]

```
std::enable_if<std::is_same<Policy, BinaryRBM>::value, void>::type Phase (
    DataType && input,
    DataType && gradient )
```

Calculates the gradient of the **RBM** (p. 864) network on the provided input.

Parameters

<i>input</i>	The provided input data.
<i>gradient</i>	Stores the gradient of the RBM (p. 864) network.

39.122.4.16 Phase() [2/2]

```
std::enable_if<std::is_same<Policy, SpikeSlabRBM>::value, void>::type Phase (
    DataType && input,
    DataType && gradient )
```

Calculates the gradient of the **RBM** (p. 864) network on the provided input.

Parameters

<i>input</i>	The provided input data.
<i>gradient</i>	Stores the gradient of the RBM (p. 864) network.

39.122.4.17 PoolSize()

```
size_t const& PoolSize ( ) const [inline]
```

Get the pool size.

Definition at line 371 of file `rbm.hpp`.

References `RBM< InitializationRuleType, DataType, PolicyType >::serialize()`.

39.122.4.18 Reset() [1/2]

```
std::enable_if<std::is_same<Policy, BinaryRBM>::value, void>::type Reset ( )
```

39.122.4.19 Reset() [2/2]

```
std::enable_if<std::is_same<Policy, SpikeSlabRBM>::value, void>::type Reset ( )
```

39.122.4.20 SampleHidden() [1/2]

```
std::enable_if<std::is_same<Policy, BinaryRBM>::value, void>::type SampleHidden (
    arma::Mat< ElemType > && input,
    arma::Mat< ElemType > && output )
```

This function samples the hidden layer given the visible layer using Bernoulli function.

Parameters

<i>input</i>	Visible layer input.
<i>output</i>	The sampled hidden layer.

39.122.4.21 SampleHidden() [2/2]

```
std::enable_if<std::is_same<Policy, SpikeSlabRBM>::value, void>::type SampleHidden (
```

```
arma::Mat< ElemType > && input,
arma::Mat< ElemType > && output )
```

This function samples the slab outputs from the Normal distribution with mean given by: $\mathbf{h}_i^{*-1} * \mathbf{W}_i^T * \mathbf{v}$ and variance: σ_i^{-1} .

Parameters

<i>input</i>	Consists of both visible and spike variables.
<i>output</i>	Sampled slab neurons.

39.122.4.22 SampleSlab()

```
std::enable_if<std::is_same<Policy, SpikeSlabRBM>::value, void>::type SampleSlab (
    DataType && slabMean,
    DataType && slab )
```

The function samples from the Normal distribution of $P(\mathbf{s}|\mathbf{v}, \mathbf{h})$, where the mean is given by: $\mathbf{h}_i^{*-1} * \mathbf{W}_i^T * \mathbf{v}$ and variance is given by: σ_i^{-1} .

Parameters

<i>slabMean</i>	Mean of the Normal distribution of the slab neurons.
<i>slab</i>	Sampled slab variable from the Normal distribution.

39.122.4.23 SampleSpike()

```
std::enable_if<std::is_same<Policy, SpikeSlabRBM>::value, void>::type SampleSpike (
    DataType && spikeMean,
    DataType && spike )
```

The function samples the spike function using Bernoulli distribution.

Parameters

<i>spikeMean</i>	Indicates $P(\mathbf{h} \mathbf{v})$.
<i>spike</i>	Sampled binary spike variables.

39.122.4.24 SampleVisible() [1/2]

```
std::enable_if<std::is_same<Policy, BinaryRBM>::value, void>::type SampleVisible (
    arma::Mat< ElemType > && input,
    arma::Mat< ElemType > && output )
```

This function samples the visible layer given the hidden layer using Bernoulli function.

Parameters

<i>input</i>	Hidden layer of the network.
<i>output</i>	The sampled visible layer.

39.122.4.25 SampleVisible() [2/2]

```
std::enable_if<std::is_same<Policy, SpikeSlabRBM>::value, void>::type SampleVisible (
    arma::Mat< ElemType > && input,
    arma::Mat< ElemType > && output )
```

Sample Hidden function samples the slab outputs from the Normal distribution with mean given by: $\mathbf{h}_i^{\leftarrow -1} * \mathbf{W}_i^{\leftarrow} \mathbf{T} * \mathbf{v}$ and variance: $\mathbf{\Sigma}^{\leftarrow -1}$.

Parameters

<i>input</i>	Hidden layer of the network.
<i>output</i>	The sampled visible layer.

39.122.4.26 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the model.

Referenced by `RBM< InitializationRuleType, DataType, PolicyType >::PoolSize()`.

39.122.4.27 Shuffle()

```
void Shuffle ( )
```

Shuffle the order of function visitation.

This may be called by the optimizer.

39.122.4.28 SlabMean()

```
std::enable_if<std::is_same<Policy, SpikeSlabRBM>::value, void>::type SlabMean (
    DataType && visible,
    DataType && spike,
    DataType && slabMean )
```

The function calculates the mean of Normal distribution of $P(s|v, h)$, where the mean is given by: $h_i^{-1} * W_i^T * v$.

Parameters

<i>visible</i>	The visible layer neurons.
<i>spike</i>	The spike variables from hidden layer.
<i>slabMean</i>	The mean of the Normal distribution of slab neurons.

39.122.4.29 SlabPenalty()

```
ElemType const& SlabPenalty ( ) const [inline]
```

Get the regularizer associated with slab variables.

Definition at line 359 of file rbm.hpp.

39.122.4.30 SpikeBias() [1/2]

```
DataType const& SpikeBias ( ) const [inline]
```

Get the regularizer associated with spike variables.

Definition at line 354 of file rbm.hpp.

39.122.4.31 SpikeBias() [2/2]

```
DataType& SpikeBias ( ) [inline]
```

Modify the regularizer associated with spike variables.

Definition at line 356 of file rbm.hpp.

39.122.4.32 SpikeMean()

```
std::enable_if<std::is_same<Policy, SpikeSlabRBM>::value, void>::type SpikeMean (
    DataType && visible,
    DataType && spikeMean )
```

The function calculates the mean of the distribution $P(h|v)$, where mean is given by: $\text{sigm}(v^T * W_i^{-1} * W_i^T * v + b_i)$.

Parameters

<i>visible</i>	The visible layer neurons.
<i>spikeMean</i>	Indicates $P(h v)$.

39.122.4.33 Train()

```
double Train (
    OptimizerType & optimizer )
```

Train the **RBM** (p. 864) on the given input data.

This will use the existing model parameters as a starting point for the optimization. If this is not what you want, then you should access the parameters vector directly with **Parameters()** (p. 873) and modify it as desired.

Parameters

<i>optimizer</i>	Optimizer type.
------------------	-----------------

Returns

The final objective of the trained model (NaN or Inf on error).

39.122.4.34 VisibleBias() [1/2]

```
DataType const& VisibleBias ( ) const [inline]
```

Return the visible bias of the network.

Definition at line 344 of file rbm.hpp.

39.122.4.35 VisibleBias() [2/2]

```
DataType& VisibleBias ( ) [inline]
```

Modify the visible bias of the network.

Definition at line 346 of file rbm.hpp.

39.122.4.36 VisibleMean() [1/2]

```
std::enable_if<std::is_same<Policy, BinaryRBM>::value, void>::type VisibleMean (
    DataType && input,
    DataType && output )
```

The function calculates the mean for the visible layer.

Parameters

<i>input</i>	Hidden neurons from the hidden layer of the network.
<i>output</i>	Visible neuron activations.

39.122.4.37 VisibleMean() [2/2]

```
std::enable_if<std::is_same<Policy, SpikeSlabRBM>::value, void>::type VisibleMean (
    DataType && input,
    DataType && output )
```

The function calculates the mean of the Normal distribution of $P(v|s, h)$.

The mean is given by: $\sum_{i=1}^N W_i * s_i * h_i$

Parameters

<i>input</i>	Consists of both the spike and slab variables.
<i>output</i>	Mean of the of the Normal distribution.

39.122.4.38 VisiblePenalty() [1/2]

```
DataType const& VisiblePenalty ( ) const [inline]
```

Get the regularizer associated with visible variables.

Definition at line 362 of file `rbm.hpp`.

39.122.4.39 VisiblePenalty() [2/2]

```
DataType& VisiblePenalty ( ) [inline]
```

Modify the regularizer associated with visible variables.

Definition at line 364 of file `rbm.hpp`.

39.122.4.40 VisibleSize()

```
size_t const& VisibleSize ( ) const [inline]
```

Get the visible size.

Definition at line 367 of file `rbm.hpp`.

39.122.4.41 Weight() [1/2]

```
arma::Cube< ElemType> const& Weight ( ) const [inline]
```

Get the weights of the network.

Definition at line 339 of file `rbm.hpp`.

39.122.4.42 Weight() [2/2]

```
arma::Cube< ElemType>& Weight ( ) [inline]
```

Modify the weights of the network.

Definition at line 341 of file `rbm.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/rbm/ rbm.hpp`

39.123 ReconstructionLoss< InputDataType, OutputDataType, DistType > Class Template Reference

The reconstruction loss performance function measures the network's performance equal to the negative log probability of the target with the input distribution.

Public Member Functions

- **ReconstructionLoss** ()
*Create the **ReconstructionLoss** (p. 880) object.*
 - template<typename InputType , typename TargetType , typename OutputType >
void **Backward** (const InputType &&input, const TargetType &&target, OutputType &&output)
Ordinary feed backward pass of a neural network.
 - template<typename InputType , typename TargetType >
double **Forward** (const InputType &&input, const TargetType &&target)
Computes the reconstruction loss.
 - OutputDataType & **OutputParameter** () const
Get the output parameter.
 - OutputDataType & **OutputParameter** ()
Modify the output parameter.
 - template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
- Serialize the layer.*

39.123.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat, typename DistType = BernoulliDistribution<InputDataType>>>
class mlpack::ann::ReconstructionLoss< InputDataType, OutputDataType, DistType >
```

The reconstruction loss performance function measures the network's performance equal to the negative log probability of the target with the input distribution.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>DistType</i>	The type of distribution parametrized by the input.

Definition at line 37 of file reconstruction_loss.hpp.

39.123.2 Constructor & Destructor Documentation

39.123.2.1 ReconstructionLoss()

```
ReconstructionLoss ( )
```

Create the **ReconstructionLoss** (p. 880) object.

39.123.3 Member Function Documentation

39.123.3.1 Backward()

```
void Backward (
    const InputType && input,
    const TargetType && target,
    OutputType && output )
```

Ordinary feed backward pass of a neural network.

Parameters

<i>input</i>	The propagated input activation.
<i>target</i>	The target matrix.
<i>output</i>	The calculated error.

39.123.3.2 Forward()

```
double Forward (
    const InputType && input,
    const TargetType && target )
```

Computes the reconstruction loss.

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>target</i>	The target matrix.

39.123.3.3 OutputParameter() [1/2]

```
OutputDataType& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 67 of file reconstruction_loss.hpp.

39.123.3.4 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 69 of file reconstruction_loss.hpp.

References ReconstructionLoss< InputDataType, OutputDataType, DistType >::serialize().

39.123.3.5 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by ReconstructionLoss< InputDataType, OutputDataType, DistType >::OutputParameter().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/ **reconstruction_loss.hpp**

39.124 RectifierFunction Class Reference

The rectifier function, defined by.

Static Public Member Functions

- static double **Deriv** (const double y)
Computes the first derivative of the rectifier function.
- template<typename InputType , typename OutputType >
static void **Deriv** (const InputType &y, OutputType &x)
Computes the first derivatives of the rectifier function.
- static double **Fn** (const double x)
Computes the rectifier function.
- template<typename eT >
static void **Fn** (const arma::Mat< eT > &x, arma::Mat< eT > &y)
Computes the rectifier function using a dense matrix as input.
- template<typename eT >
static void **Fn** (const arma::Cube< eT > &x, arma::Cube< eT > &y)
Computes the rectifier function using a 3rd-order tensor as input.

39.124.1 Detailed Description

The rectifier function, defined by.

$$\begin{aligned} f(x) &= \max(0, x) \\ f'(x) &= \begin{cases} 1 & : x > 0 \\ 0 & : x \leq 0 \end{cases} \end{aligned}$$

Definition at line 45 of file rectifier_function.hpp.

39.124.2 Member Function Documentation

39.124.2.1 Deriv() [1/2]

```
static double Deriv (  
    const double y ) [inline], [static]
```

Computes the first derivative of the rectifier function.

Parameters

x	Input data.
---	-------------

Returns

$f'(x)$

Definition at line 91 of file rectifier_function.hpp.

Referenced by RectifierFunction::Deriv().

39.124.2.2 Deriv() [2/2]

```
static void Deriv (  
    const InputType & y,  
    OutputType & x ) [inline], [static]
```

Computes the first derivatives of the rectifier function.

Parameters

<i>y</i>	Input activations.
<i>x</i>	The resulting derivatives.

Definition at line 103 of file rectifier_function.hpp.

References RectifierFunction::Deriv().

39.124.2.3 Fn() [1/3]

```
static double Fn (  
    const double x ) [inline], [static]
```

Computes the rectifier function.

Parameters

<i>x</i>	Input data.
----------	-------------

Returns

$f(x)$.

Definition at line 54 of file rectifier_function.hpp.

39.124.2.4 Fn() [2/3]

```
static void Fn (  
    const arma::Mat< eT > & x,  
    arma::Mat< eT > & y ) [inline], [static]
```

Computes the rectifier function using a dense matrix as input.

Parameters

<i>x</i>	Input data.
<i>y</i>	The resulting output activation.

Definition at line 66 of file rectifier_function.hpp.

39.124.2.5 Fn() [3/3]

```
static void Fn (
    const arma::Cube< eT > & x,
    arma::Cube< eT > & y ) [inline], [static]
```

Computes the rectifier function using a 3rd-order tensor as input.

Parameters

<i>x</i>	Input data.
<i>y</i>	The resulting output activation.

Definition at line 79 of file rectifier_function.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/ **rectifier_function.hpp**

39.125 Recurrent< InputDataType, OutputDataType, CustomLayers > Class Template Reference

Implementation of the RecurrentLayer class.

Public Member Functions

- **Recurrent** ()
*Default constructor—this will create a **Recurrent** (p. 886) object that can't be used, so be careful! Make sure to set all the parameters before use.*
- **Recurrent** (const **Recurrent** &)
Copy constructor.
- template<typename StartModuleType, typename InputModuleType, typename FeedbackModuleType, typename TransferModuleType >
Recurrent (const StartModuleType &start, const InputModuleType &input, const FeedbackModuleType &feedback, const TransferModuleType &transfer, const size_t rho)
*Create the **Recurrent** (p. 886) object using the specified modules.*
- ~**Recurrent** ()
Destructor to release allocated memory.
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- bool **Deterministic** () const

- The value of the deterministic parameter.*

 - bool & **Deterministic** ()

Modify the value of the deterministic parameter.
- template<typename eT >
void **Forward** (arma::Mat< eT > &&input, arma::Mat< eT > &&output)

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- template<typename eT >
void **Gradient** (arma::Mat< eT > &&input, arma::Mat< eT > &&error, arma::Mat< eT > &&)
- OutputDataType const & **Gradient** () const

Get the gradient.
- OutputDataType & **Gradient** ()

Modify the gradient.
- std::vector< **LayerTypes**< CustomLayers... > > & **Model** ()

Get the model modules.
- OutputDataType const & **OutputParameter** () const

Get the output parameter.
- OutputDataType & **OutputParameter** ()

Modify the output parameter.
- OutputDataType const & **Parameters** () const

Get the parameters.
- OutputDataType & **Parameters** ()

Modify the parameters.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)

Serialize the layer.

39.125.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat, typename... CustomLayers>
class mlpack::ann::Recurrent< InputDataType, OutputDataType, CustomLayers >
```

Implementation of the RecurrentLayer class.

Recurrent (p. 886) layers can be used similarly to feed-forward layers.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 89 of file layer_types.hpp.

39.125.2 Constructor & Destructor Documentation

39.125.2.1 **Recurrent()** [1/3]

```
Recurrent ( )
```

Default constructor—this will create a **Recurrent** (p. 886) object that can't be used, so be careful! Make sure to set all the parameters before use.

39.125.2.2 **~Recurrent()**

```
~ Recurrent ( )
```

Destructor to release allocated memory.

39.125.2.3 **Recurrent()** [2/3]

```
Recurrent (
    const Recurrent< InputDataType, OutputDataType, CustomLayers > & )
```

Copy constructor.

39.125.2.4 **Recurrent()** [3/3]

```
Recurrent (
    const StartModuleType & start,
    const InputModuleType & input,
    const FeedbackModuleType & feedback,
    const TransferModuleType & transfer,
    const size_t rho )
```

Create the **Recurrent** (p. 886) object using the specified modules.

Parameters

<i>start</i>	The start module.
<i>input</i>	The input module.
<i>feedback</i>	The feedback module.
<i>transfer</i>	The transfer module.
<i>rho</i>	Maximum number of steps to backpropagate through time (BPTT).

39.125.3 Member Function Documentation

39.125.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.125.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 133 of file recurrent.hpp.

39.125.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 135 of file recurrent.hpp.

39.125.3.4 Deterministic() [1/2]

```
bool Deterministic ( ) const [inline]
```

The value of the deterministic parameter.

Definition at line 118 of file recurrent.hpp.

39.125.3.5 Deterministic() [2/2]

```
bool& Deterministic ( ) [inline]
```

Modify the value of the deterministic parameter.

Definition at line 120 of file recurrent.hpp.

39.125.3.6 Forward()

```
void Forward (
    arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.125.3.7 Gradient() [1/3]

```
void Gradient (
    arma::Mat< eT > && input,
    arma::Mat< eT > && error,
    arma::Mat< eT > && )
```

39.125.3.8 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 138 of file recurrent.hpp.

39.125.3.9 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 140 of file recurrent.hpp.

References Recurrent< InputDataType, OutputDataType, CustomLayers >::serialize().

39.125.3.10 Model()

```
std::vector< LayerTypes<CustomLayers...> >& Model ( ) [inline]
```

Get the model modules.

Definition at line 115 of file recurrent.hpp.

39.125.3.11 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 128 of file recurrent.hpp.

39.125.3.12 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 130 of file recurrent.hpp.

39.125.3.13 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 123 of file recurrent.hpp.

39.125.3.14 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 125 of file recurrent.hpp.

39.125.3.15 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by Recurrent< InputDataType, OutputDataType, CustomLayers >::Gradient().

The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **layer_types.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **recurrent.hpp**

39.126 RecurrentAttention< InputDataType, OutputDataType > Class Template Reference

This class implements the **Recurrent** (p. 886) Model for Visual Attention, using a variety of possible layer implementations.

Public Member Functions

- **RecurrentAttention** ()

*Default constructor: this will not give a usable **RecurrentAttention** (p. 892) object, so be sure to set all the parameters before use.*

- template<typename RNNModuleType, typename ActionModuleType >

RecurrentAttention (const size_t outSize, const RNNModuleType &rnn, const ActionModuleType &action, const size_t rho)

*Create the **RecurrentAttention** (p. 892) object using the specified modules.*

- template<typename eT >

void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

- OutputDataType const & **Delta** () const

Get the delta.

- OutputDataType & **Delta** ()

Modify the delta.

- bool **Deterministic** () const

The value of the deterministic parameter.

- bool & **Deterministic** ()

Modify the value of the deterministic parameter.

- template<typename eT >

void **Forward** (arma::Mat< eT > &&input, arma::Mat< eT > &&output)

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

- template<typename eT >

void **Gradient** (arma::Mat< eT > &&, arma::Mat< eT > &&, arma::Mat< eT > &&)

- OutputDataType const & **Gradient** () const

Get the gradient.

- OutputDataType & **Gradient** ()

Modify the gradient.

- std::vector< **LayerTypes**<> > & **Model** ()

Get the model modules.

- OutputDataType const & **OutputParameter** () const

Get the output parameter.

- OutputDataType & **OutputParameter** ()

Modify the output parameter.

- OutputDataType const & **Parameters** () const

Get the parameters.

- OutputDataType & **Parameters** ()

Modify the parameters.

- template<typename Archive >

void **serialize** (Archive &ar, const unsigned int)

Serialize the layer.

39.126.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::RecurrentAttention< InputDataType, OutputDataType >
```

This class implements the **Recurrent** (p. 886) Model for Visual Attention, using a variety of possible layer implementations.

For more information, see the following paper.

```
@article{MnihHGK14,
  title={Recurrent Models of Visual Attention},
  author={Volodymyr Mnih, Nicolas Heess, Alex Graves, Koray Kavukcuoglu},
  journal={CoRR},
  volume={abs/1406.6247},
  year={2014}
}
```

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 135 of file layer_types.hpp.

39.126.2 Constructor & Destructor Documentation

39.126.2.1 RecurrentAttention() [1/2]

```
RecurrentAttention ( )
```

Default constructor: this will not give a usable **RecurrentAttention** (p. 892) object, so be sure to set all the parameters before use.

39.126.2.2 RecurrentAttention() [2/2]

```
RecurrentAttention (
  const size_t outSize,
  const RNNModuleType & rnn,
  const ActionModuleType & action,
  const size_t rho )
```

Create the **RecurrentAttention** (p. 892) object using the specified modules.

Parameters

<i>start</i>	The module output size.
<i>start</i>	The recurrent neural network module.
<i>start</i>	The action module.
<i>rho</i>	Maximum number of steps to backpropagate through time (BPTT).

39.126.3 Member Function Documentation

39.126.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.126.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 133 of file recurrent_attention.hpp.

39.126.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 135 of file recurrent_attention.hpp.

39.126.3.4 Deterministic() [1/2]

```
bool Deterministic ( ) const [inline]
```

The value of the deterministic parameter.

Definition at line 118 of file recurrent_attention.hpp.

39.126.3.5 Deterministic() [2/2]

```
bool& Deterministic ( ) [inline]
```

Modify the value of the deterministic parameter.

Definition at line 120 of file recurrent_attention.hpp.

39.126.3.6 Forward()

```
void Forward (
    arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.126.3.7 Gradient() [1/3]

```
void Gradient (
    arma::Mat< eT > && ,
    arma::Mat< eT > && ,
    arma::Mat< eT > && )
```


39.126.3.8 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 138 of file recurrent_attention.hpp.

39.126.3.9 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 140 of file recurrent_attention.hpp.

References RecurrentAttention< InputDataType, OutputDataType >::serialize().

39.126.3.10 Model()

```
std::vector< LayerTypes<> >& Model ( ) [inline]
```

Get the model modules.

Definition at line 115 of file recurrent_attention.hpp.

39.126.3.11 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 128 of file recurrent_attention.hpp.

39.126.3.12 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 130 of file recurrent_attention.hpp.

39.126.3.13 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 123 of file recurrent_attention.hpp.

39.126.3.14 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 125 of file recurrent_attention.hpp.

39.126.3.15 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by RecurrentAttention< InputDataType, OutputDataType >::Gradient().

The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **layer_types.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **recurrent_attention.hpp**

39.127 ReinforceNormal< InputDataType, OutputDataType > Class Template Reference

Implementation of the reinforce normal layer.

Public Member Functions

- **ReinforceNormal** (const double stdev=1.0)
*Create the **ReinforceNormal** (p. 898) object.*
- template<typename DataType >
void **Backward** (const DataType &&input, DataType &&, DataType &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- OutputDataType & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- bool **Deterministic** () const
Get the value of the deterministic parameter.
- bool & **Deterministic** ()
Modify the value of the deterministic parameter.
- template<typename eT >
void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- OutputDataType & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- double **Reward** () const
Get the value of the reward parameter.
- double & **Reward** ()
Modify the value of the deterministic parameter.
- template<typename Archive >
void **serialize** (Archive &, const unsigned int)
Serialize the layer.

39.127.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::ReinforceNormal< InputDataType, OutputDataType >
```

Implementation of the reinforce normal layer.

The reinforce normal layer implements the REINFORCE algorithm for the normal distribution.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 34 of file reinforce_normal.hpp.

39.127.2 Constructor & Destructor Documentation

39.127.2.1 ReinforceNormal()

```
ReinforceNormal (
    const double stdev = 1.0 )
```

Create the **ReinforceNormal** (p. 898) object.

Parameters

<i>stdev</i>	Standard deviation used during the forward and backward pass.
--------------	---

39.127.3 Member Function Documentation

39.127.3.1 Backward()

```
void Backward (
    const DataType && input,
    DataType && ,
    DataType && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.127.3.2 Delta() [1/2]

```
OutputDataType& Delta ( ) const [inline]
```

Get the delta.

Definition at line 72 of file reinforce_normal.hpp.

39.127.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 74 of file reinforce_normal.hpp.

39.127.3.4 Deterministic() [1/2]

```
bool Deterministic ( ) const [inline]
```

Get the value of the deterministic parameter.

Definition at line 77 of file reinforce_normal.hpp.

39.127.3.5 Deterministic() [2/2]

```
bool& Deterministic ( ) [inline]
```

Modify the value of the deterministic parameter.

Definition at line 79 of file reinforce_normal.hpp.

39.127.3.6 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.127.3.7 OutputParameter() [1/2]

```
OutputDataType& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 67 of file reinforce_normal.hpp.

39.127.3.8 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 69 of file reinforce_normal.hpp.

39.127.3.9 Reward() [1/2]

```
double Reward ( ) const [inline]
```

Get the value of the reward parameter.

Definition at line 82 of file reinforce_normal.hpp.

39.127.3.10 Reward() [2/2]

```
double& Reward ( ) [inline]
```

Modify the value of the deterministic parameter.

Definition at line 84 of file reinforce_normal.hpp.

References ReinforceNormal< InputDataType, OutputDataType >::serialize().

39.127.3.11 serialize()

```
void serialize (
    Archive & ,
    const unsigned int )
```

Serialize the layer.

Referenced by ReinforceNormal< InputDataType, OutputDataType >::Reward().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **reinforce_normal.hpp**

39.128 Reparametrization< InputDataType, OutputDataType > Class Template Reference

Implementation of the **Reparametrization** (p. 903) layer class.

Public Member Functions

- **Reparametrization** ()
*Create the **Reparametrization** (p. 903) object.*
- **Reparametrization** (const size_t latentSize, const bool stochastic=true, const bool includeKl=true, const double beta=1)
*Create the **Reparametrization** (p. 903) layer object using the specified sample vector size.*
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, calculating the function f(x) by propagating x backwards through f.
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename eT >
void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Ordinary feed forward pass of a neural network, evaluating the function f(x) by propagating the activity forward through f.
- double **Loss** ()
Get the KL divergence with standard normal.
- OutputDataType const & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- size_t const & **OutputSize** () const
Get the output size.
- size_t & **OutputSize** ()
Modify the output size.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.128.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::Reparametrization< InputDataType, OutputDataType >
```

Implementation of the **Reparametrization** (p. 903) layer class.

This layer samples from the given parameters of a normal distribution.

This class also supports beta-VAE, a state-of-the-art framework for automated discovery of interpretable factorised latent representations from raw image data in a completely unsupervised manner.

For more information, refer the following paper.

```
@article{ICLR2017,
  title   = {beta-VAE: Learning basic visual concepts with a constrained
            variational framework},
  author  = {Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess,
            Xavier Glorot, Matthew Botvinick, Shakir Mohamed and
            Alexander Lerchner | Google DeepMind},
  journal = {2017 International Conference on Learning Representations (ICLR)},
  year    = {2017}
}
```

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 70 of file layer_types.hpp.

39.128.2 Constructor & Destructor Documentation

39.128.2.1 Reparametrization() [1/2]

```
Reparametrization ( )
```

Create the **Reparametrization** (p. 903) object.

39.128.2.2 Reparametrization() [2/2]

```
Reparametrization (
    const size_t latentSize,
    const bool stochastic = true,
    const bool includeKl = true,
    const double beta = 1 )
```

Create the **Reparametrization** (p. 903) layer object using the specified sample vector size.

Parameters

<i>latentSize</i>	The number of output latent units.
<i>stochastic</i>	Whether we want random sample or constant.
<i>includeKl</i>	Whether we want to include KL loss in backward function.
<i>beta</i>	The beta (hyper)parameter for beta-VAE mentioned above.

39.128.3 Member Function Documentation

39.128.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards trough f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.128.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 104 of file reparametrization.hpp.

39.128.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 106 of file reparametrization.hpp.

39.128.3.4 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.128.3.5 Loss()

```
double Loss ( ) [inline]
```

Get the KL divergence with standard normal.

Definition at line 114 of file reparametrization.hpp.

References `Reparametrization< InputDataType, OutputDataType >::serialize()`.

39.128.3.6 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 99 of file reparametrization.hpp.

39.128.3.7 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 101 of file reparametrization.hpp.

39.128.3.8 OutputSize() [1/2]

```
size_t const& OutputSize ( ) const [inline]
```

Get the output size.

Definition at line 109 of file reparametrization.hpp.

39.128.3.9 OutputSize() [2/2]

```
size_t& OutputSize ( ) [inline]
```

Modify the output size.

Definition at line 111 of file reparametrization.hpp.

39.128.3.10 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by Reparametrization< InputDataType, OutputDataType >::Loss().

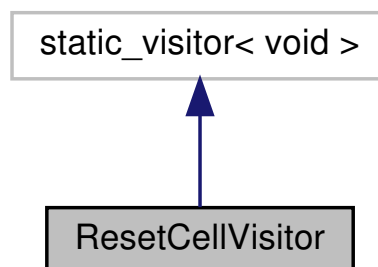
The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **layer_types.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **reparametrization.hpp**

39.129 ResetCellVisitor Class Reference

ResetCellVisitor (p. 907) executes the ResetCell() function.

Inheritance diagram for ResetCellVisitor:



Public Member Functions

- **ResetCellVisitor** (const size_t size)
Reset the cell using the given size.
- template<typename LayerType >
void **operator()** (LayerType *layer) const
Execute the ResetCell() function.

39.129.1 Detailed Description

ResetCellVisitor (p. 907) executes the ResetCell() function.

Definition at line 26 of file reset_cell_visitor.hpp.

39.129.2 Constructor & Destructor Documentation

39.129.2.1 ResetCellVisitor()

```
ResetCellVisitor (  
    const size_t size )
```

Reset the cell using the given size.

39.129.3 Member Function Documentation

39.129.3.1 operator()

```
void operator() (  
    LayerType * layer ) const
```

Execute the ResetCell() function.

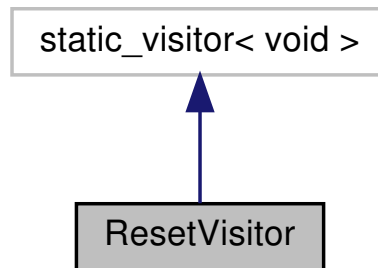
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **reset_cell_visitor.hpp**

39.130 ResetVisitor Class Reference

ResetVisitor (p. 909) executes the `Reset()` function.

Inheritance diagram for `ResetVisitor`:



Public Member Functions

- `template<typename LayerType >`
`void operator() (LayerType *layer) const`
Execute the `Reset()` function.

39.130.1 Detailed Description

ResetVisitor (p. 909) executes the `Reset()` function.

Definition at line 26 of file `reset_visitor.hpp`.

39.130.2 Member Function Documentation

39.130.2.1 `operator()`

```
void operator() (  
    LayerType * layer ) const
```

Execute the `Reset()` function.

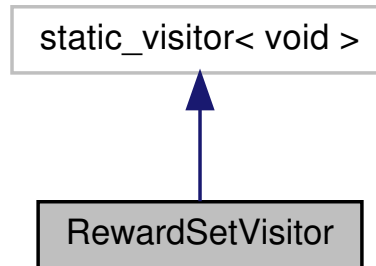
The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ reset_visitor.hpp`

39.131 RewardSetVisitor Class Reference

RewardSetVisitor (p. 910) set the reward parameter given the reward value.

Inheritance diagram for RewardSetVisitor:



Public Member Functions

- **RewardSetVisitor** (const double reward)
Set the reward parameter given the reward value.
- template<typename LayerType >
void **operator()** (LayerType *layer) const
Set the reward parameter.

39.131.1 Detailed Description

RewardSetVisitor (p. 910) set the reward parameter given the reward value.

Definition at line 26 of file reward_set_visitor.hpp.

39.131.2 Constructor & Destructor Documentation

39.131.2.1 RewardSetVisitor()

```

RewardSetVisitor (
    const double reward )
  
```

Set the reward parameter given the reward value.

39.131.3 Member Function Documentation

39.131.3.1 operator()

```
void operator() (
    LayerType * layer ) const
```

Set the reward parameter.

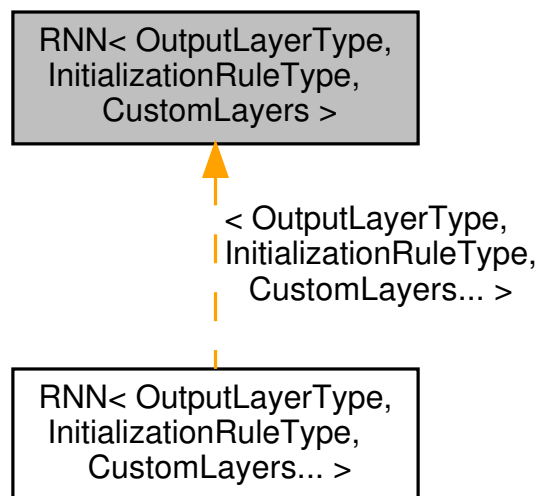
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **reward_set_visitor.hpp**

39.132 RNN< OutputLayerType, InitializationRuleType, CustomLayers > Class Template Reference

Implementation of a standard recurrent neural network container.

Inheritance diagram for RNN< OutputLayerType, InitializationRuleType, CustomLayers >:



Public Types

- using **NetworkType** = `RNN< OutputLayerType, InitializationRuleType, CustomLayers... >`
Convenience typedef for the internal model construction.

Public Member Functions

- **RNN** (const size_t rho, const bool single=false, OutputLayerType outputLayer=OutputLayerType(), InitializationRuleType initializeRule=InitializationRuleType())
*Create the **RNN** (p. 911) object.*
- **~RNN** ()
Destructor to release allocated memory.
- template<class LayerType , class... Args>
void **Add** (Args... args)
- void **Add** (**LayerTypes**< CustomLayers... > layer)
- double **Evaluate** (const arma::mat ¶meters, const size_t begin, const size_t batchSize, const bool deterministic)
Evaluate the recurrent neural network with the given parameters.
- double **Evaluate** (const arma::mat ¶meters, const size_t begin, const size_t batchSize)
Evaluate the recurrent neural network with the given parameters.
- template<typename GradType >
double **EvaluateWithGradient** (const arma::mat ¶meters, const size_t begin, GradType &gradient, const size_t batchSize)
Evaluate the recurrent neural network with the given parameters.
- void **Gradient** (const arma::mat ¶meters, const size_t begin, arma::mat &gradient, const size_t batchSize)
Evaluate the gradient of the recurrent neural network with the given parameters, and with respect to only one point in the dataset.
- size_t **NumFunctions** () const
Return the number of separable functions (the number of predictor points).
- const arma::mat & **Parameters** () const
Return the initial point for the optimization.
- arma::mat & **Parameters** ()
Modify the initial point for the optimization.
- void **Predict** (arma::cube predictors, arma::cube &results, const size_t batchSize=256)
Predict the responses to a given set of predictors.
- const arma::cube & **Predictors** () const
Get the matrix of data points (predictors).
- arma::cube & **Predictors** ()
Modify the matrix of data points (predictors).
- void **Reset** ()
Reset the state of the network.
- void **ResetParameters** ()
Reset the module information (weights/parameters).
- const arma::cube & **Responses** () const
Get the matrix of responses to the input data points.
- arma::cube & **Responses** ()
Modify the matrix of responses to the input data points.
- const size_t & **Rho** () const
Return the maximum length of backpropagation through time.
- size_t & **Rho** ()
Modify the maximum length of backpropagation through time.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)

- Serialize the model.*
 - void **Shuffle** ()
 - Shuffle the order of function visitation.*
- template<typename OptimizerType >
 double **Train** (arma::cube predictors, arma::cube responses, OptimizerType &optimizer)
 - Train the recurrent neural network on the given input data using the given optimizer.*
- template<typename OptimizerType = ens::StandardSGD>
 double **Train** (arma::cube predictors, arma::cube responses)
 - Train the recurrent neural network on the given input data.*

39.132.1 Detailed Description

```
template<typename OutputLayerType = NegativeLogLikelihood<>, typename InitializationRuleType = RandomInitialization, type-
name... CustomLayers>
class mlpack::ann::RNN< OutputLayerType, InitializationRuleType, CustomLayers >
```

Implementation of a standard recurrent neural network container.

Template Parameters

<i>OutputLayerType</i>	The output layer type used to evaluate the network.
<i>InitializationRuleType</i>	Rule used to initialize the weight matrix.

Definition at line 44 of file rnn.hpp.

39.132.2 Member Typedef Documentation

39.132.2.1 NetworkType

```
using NetworkType = RNN<OutputLayerType, InitializationRuleType, CustomLayers...>
```

Convenience typedef for the internal model construction.

Definition at line 50 of file rnn.hpp.

39.132.3 Constructor & Destructor Documentation

39.132.3.1 RNN()

```

RNN (
    const size_t rho,
    const bool single = false,
    OutputLayerType outputLayer = OutputLayerType(),
    InitializationRuleType initializeRule = InitializationRuleType() )

```

Create the **RNN** (p. 911) object.

Optionally, specify which initialize rule and performance function should be used.

If you want to pass in a parameter and discard the original parameter object, be sure to use `std::move` to avoid unnecessary copy.

Parameters

<i>rho</i>	Maximum number of steps to backpropagate through time (BPTT).
<i>single</i>	Predict only the last element of the input sequence.
<i>outputLayer</i>	Output layer used to evaluate the network.
<i>initializeRule</i>	Optional instantiated InitializationRule object for initializing the network parameter.

39.132.3.2 ~RNN()

```
~ RNN ( )
```

Destructor to release allocated memory.

39.132.4 Member Function Documentation

39.132.4.1 Add() [1/2]

```

void Add (
    Args... args ) [inline]

```

Definition at line 233 of file rnn.hpp.

39.132.4.2 Add() [2/2]

```
void Add (
    LayerTypes< CustomLayers... > layer ) [inline]
```

Definition at line 240 of file rnn.hpp.

39.132.4.3 Evaluate() [1/2]

```
double Evaluate (
    const arma::mat & parameters,
    const size_t begin,
    const size_t batchSize,
    const bool deterministic )
```

Evaluate the recurrent neural network with the given parameters.

This function is usually called by the optimizer to train the model.

Parameters

<i>parameters</i>	Matrix model parameters.
<i>begin</i>	Index of the starting point to use for objective function evaluation.
<i>batchSize</i>	Number of points to be passed at a time to use for objective function evaluation.
<i>deterministic</i>	Whether or not to train or test the model. Note some layer act differently in training or testing mode.

39.132.4.4 Evaluate() [2/2]

```
double Evaluate (
    const arma::mat & parameters,
    const size_t begin,
    const size_t batchSize )
```

Evaluate the recurrent neural network with the given parameters.

This function is usually called by the optimizer to train the model. This just calls the other overload of **Evaluate()** (p. 915) with `deterministic = true`.

Parameters

<i>parameters</i>	Matrix model parameters.
<i>begin</i>	Index of the starting point to use for objective function evaluation.
<i>batchSize</i>	Number of points to be passed at a time to use for objective function evaluation.

39.132.4.5 EvaluateWithGradient()

```
double EvaluateWithGradient (
    const arma::mat & parameters,
    const size_t begin,
    GradType & gradient,
    const size_t batchSize )
```

Evaluate the recurrent neural network with the given parameters.

This function is usually called by the optimizer to train the model.

Parameters

<i>parameters</i>	Matrix model parameters.
<i>begin</i>	Index of the starting point to use for objective function evaluation.
<i>gradient</i>	Matrix to output gradient into.
<i>batchSize</i>	Number of points to be passed at a time to use for objective function evaluation.

39.132.4.6 Gradient()

```
void Gradient (
    const arma::mat & parameters,
    const size_t begin,
    arma::mat & gradient,
    const size_t batchSize )
```

Evaluate the gradient of the recurrent neural network with the given parameters, and with respect to only one point in the dataset.

This is useful for optimizers such as SGD, which require a separable objective function.

Parameters

<i>parameters</i>	Matrix of the model parameters to be optimized.
<i>begin</i>	Index of the starting point to use for objective function gradient evaluation.
<i>gradient</i>	Matrix to output gradient into.
<i>batchSize</i>	Number of points to be processed as a batch for objective function gradient evaluation.

Referenced by RNN< OutputLayerType, InitializationRuleType, CustomLayers... >::Predictors().

39.132.4.7 NumFunctions()

```
size_t NumFunctions ( ) const [inline]
```

Return the number of separable functions (the number of predictor points).

Definition at line 243 of file rnn.hpp.

39.132.4.8 Parameters() [1/2]

```
const arma::mat& Parameters ( ) const [inline]
```

Return the initial point for the optimization.

Definition at line 246 of file rnn.hpp.

39.132.4.9 Parameters() [2/2]

```
arma::mat& Parameters ( ) [inline]
```

Modify the initial point for the optimization.

Definition at line 248 of file rnn.hpp.

39.132.4.10 Predict()

```
void Predict (
    arma::cube predictors,
    arma::cube & results,
    const size_t batchSize = 256 )
```

Predict the responses to a given set of predictors.

The responses will reflect the output of the given output layer as returned by the output layer function.

If you want to pass in a parameter and discard the original parameter object, be sure to use `std::move` to avoid unnecessary copy.

The format of the data should be as follows:

- each slice should correspond to a time step
- each column should correspond to a data point
- each row should correspond to a dimension So, e.g., `predictors(i, j, k)` is the *i*'th dimension of the *j*'th data point at time slice *k*. The responses will be in the same format.

Parameters

<i>predictors</i>	Input predictors.
<i>results</i>	Matrix to put output predictions of responses into.
<i>batchSize</i>	Number of points to predict at once.

39.132.4.11 Predictors() [1/2]

```
const arma::cube& Predictors ( ) const [inline]
```

Get the matrix of data points (predictors).

Definition at line 261 of file rnn.hpp.

39.132.4.12 Predictors() [2/2]

```
arma::cube& Predictors ( ) [inline]
```

Modify the matrix of data points (predictors).

Definition at line 263 of file rnn.hpp.

39.132.4.13 Reset()

```
void Reset ( )
```

Reset the state of the network.

This ensures that all internally-held gradients are set to 0, all memory cells are reset, and the parameters matrix is the right size.

Referenced by `RNN< OutputLayerType, InitializationRuleType, CustomLayers... >::Predictors()`.

39.132.4.14 ResetParameters()

```
void ResetParameters ( )
```

Reset the module information (weights/parameters).

Referenced by `RNN< OutputLayerType, InitializationRuleType, CustomLayers... >::Predictors()`.

39.132.4.15 Responses() [1/2]

```
const arma::cube& Responses ( ) const [inline]
```

Get the matrix of responses to the input data points.

Definition at line 256 of file rnn.hpp.

39.132.4.16 Responses() [2/2]

```
arma::cube& Responses ( ) [inline]
```

Modify the matrix of responses to the input data points.

Definition at line 258 of file rnn.hpp.

39.132.4.17 Rho() [1/2]

```
const size_t& Rho ( ) const [inline]
```

Return the maximum length of backpropagation through time.

Definition at line 251 of file rnn.hpp.

39.132.4.18 Rho() [2/2]

```
size_t& Rho ( ) [inline]
```

Modify the maximum length of backpropagation through time.

Definition at line 253 of file rnn.hpp.

39.132.4.19 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the model.

Referenced by RNN< OutputLayerType, InitializationRuleType, CustomLayers... >::Predictors().

39.132.4.20 Shuffle()

```
void Shuffle ( )
```

Shuffle the order of function visitation.

This may be called by the optimizer.

39.132.4.21 Train() [1/2]

```
double Train (
    arma::cube predictors,
    arma::cube responses,
    OptimizerType & optimizer )
```

Train the recurrent neural network on the given input data using the given optimizer.

This will use the existing model parameters as a starting point for the optimization. If this is not what you want, then you should access the parameters vector directly with **Parameters()** (p. 917) and modify it as desired.

If you want to pass in a parameter and discard the original parameter object, be sure to use `std::move` to avoid unnecessary copy.

The format of the data should be as follows:

- each slice should correspond to a time step
- each column should correspond to a data point
- each row should correspond to a dimension So, e.g., `predictors(i, j, k)` is the *i*'th dimension of the *j*'th data point at time slice *k*.

Template Parameters

<i>OptimizerType</i>	Type of optimizer to use to train the model.
----------------------	--

Parameters

<i>predictors</i>	Input training variables.
<i>responses</i>	Outputs results from input training variables.
<i>optimizer</i>	Instantiated optimizer used to train the model.

Returns

The final objective of the trained model (NaN or Inf on error).

39.132.4.22 Train() [2/2]

```
double Train (
    arma::cube predictors,
    arma::cube responses )
```

Train the recurrent neural network on the given input data.

By default, the SGD optimization algorithm is used, but others can be specified (such as `ens::RMSprop`).

This will use the existing model parameters as a starting point for the optimization. If this is not what you want, then you should access the parameters vector directly with **Parameters()** (p. 917) and modify it as desired.

If you want to pass in a parameter and discard the original parameter object, be sure to use `std::move` to avoid unnecessary copy.

The format of the data should be as follows:

- each slice should correspond to a time step
- each column should correspond to a data point
- each row should correspond to a dimension So, e.g., `predictors(i, j, k)` is the *i*'th dimension of the *j*'th data point at time slice *k*.

Template Parameters

<i>OptimizerType</i>	Type of optimizer to use to train the model.
----------------------	--

Parameters

<i>predictors</i>	Input training variables.
<i>responses</i>	Outputs results from input training variables.

Returns

The final objective of the trained model (NaN or Inf on error).

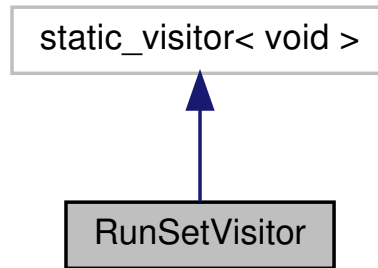
The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/ rnn.hpp`

39.133 RunSetVisitor Class Reference

RunSetVisitor (p. 921) set the run parameter given the run value.

Inheritance diagram for RunSetVisitor:



Public Member Functions

- **RunSetVisitor** (const bool run=true)
Set the run parameter given the current run value.
- template<typename LayerType >
 void **operator()** (LayerType *layer) const
Set the run parameter.

39.133.1 Detailed Description

RunSetVisitor (p. 921) set the run parameter given the run value.

Definition at line 28 of file run_set_visitor.hpp.

39.133.2 Constructor & Destructor Documentation

39.133.2.1 RunSetVisitor()

```

RunSetVisitor (
    const bool run = true )
  
```

Set the run parameter given the current run value.

39.133.3 Member Function Documentation

39.133.3.1 operator()

```
void operator() (
    LayerType * layer ) const
```

Set the run parameter.

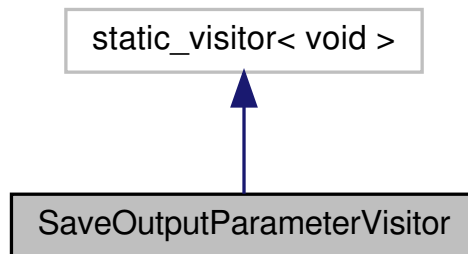
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **run_set_visitor.hpp**

39.134 SaveOutputParameterVisitor Class Reference

SaveOutputParameterVisitor (p. 923) saves the output parameter into the given parameter set.

Inheritance diagram for SaveOutputParameterVisitor:



Public Member Functions

- **SaveOutputParameterVisitor** (std::vector< arma::mat > &¶meter)
Save the output parameter into the given parameter set.
- template<typename LayerType >
 void **operator()** (LayerType *layer) const
Save the output parameter.

39.134.1 Detailed Description

SaveOutputParameterVisitor (p. 923) saves the output parameter into the given parameter set.

Definition at line 27 of file save_output_parameter_visitor.hpp.

39.134.2 Constructor & Destructor Documentation

39.134.2.1 SaveOutputParameterVisitor()

```
SaveOutputParameterVisitor (
    std::vector< arma::mat > && parameter )
```

Save the output parameter into the given parameter set.

39.134.3 Member Function Documentation

39.134.3.1 operator>()

```
void operator() (
    LayerType * layer ) const
```

Save the output parameter.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **save_output_parameter_visitor.hpp**

39.135 Select< InputDataType, OutputDataType > Class Template Reference

The select module selects the specified column from a given input matrix.

Public Member Functions

- **Select** (const size_t index=0, const size_t elements=0)
*Create the **Select** (p. 924) object.*
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- OutputDataType & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename eT >
void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- OutputDataType & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.135.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::Select< InputDataType, OutputDataType >
```

The select module selects the specified column from a given input matrix.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 32 of file select.hpp.

39.135.2 Constructor & Destructor Documentation

39.135.2.1 Select()

```
Select (
    const size_t index = 0,
    const size_t elements = 0 )
```

Create the **Select** (p. 924) object.

Parameters

<i>index</i>	The column which should be extracted from the given input.
<i>elements</i>	The number of elements that should be used.

39.135.3 Member Function Documentation

39.135.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, calculating the function f(x) by propagating x backwards through f.

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.135.3.2 Delta() [1/2]

```
OutputDataType& Delta ( ) const [inline]
```

Get the delta.

Definition at line 73 of file select.hpp.

39.135.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 75 of file select.hpp.

References `Select< InputDataType, OutputDataType >::serialize()`.

39.135.3.4 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.135.3.5 OutputParameter() [1/2]

```
OutputDataType& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 68 of file select.hpp.

39.135.3.6 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 70 of file select.hpp.

39.135.3.7 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by `Select< InputDataType, OutputDataType >::Delta()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ select.hpp`

39.136 Sequential< InputDataType, OutputDataType, Residual, CustomLayers > Class Template Reference

Implementation of the **Sequential** (p. 927) class.

Public Member Functions

- **Sequential** (const bool model=true)
*Create the **Sequential** (p. 927) object using the specified parameters.*
- **~Sequential** ()
*Destroy the **Sequential** (p. 927) object.*
- template<class LayerType , class... Args>
void **Add** (Args... args)
- void **Add** (**LayerTypes**< CustomLayers... > layer)
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, using 3rd-order tensors as input, calculating the function $f(x)$ by propagating x backwards through f .
- void **DeleteModules** ()
- arma::mat const & **Delta** () const
Get the delta.
- arma::mat & **Delta** ()
Modify the delta.
- template<typename eT >
void **Forward** (arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- template<typename eT >
void **Gradient** (arma::Mat< eT > &&input, arma::Mat< eT > &&error, arma::Mat< eT > &&)
- arma::mat const & **Gradient** () const
Get the gradient.
- arma::mat & **Gradient** ()
Modify the gradient.
- arma::mat const & **InputParameter** () const
Get the input parameter.
- arma::mat & **InputParameter** ()
Modify the input parameter.
- std::vector< **LayerTypes**< CustomLayers... > > & **Model** ()
Return the model modules.
- arma::mat const & **OutputParameter** () const
Get the output parameter.
- arma::mat & **OutputParameter** ()
Modify the output parameter.
- const arma::mat & **Parameters** () const
Return the initial point for the optimization.
- arma::mat & **Parameters** ()
Modify the initial point for the optimization.
- template<typename Archive >
void **serialize** (Archive &, const unsigned int)
- *Serialize the layer.*

39.136.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat, bool Residual = false, typename... CustomLayers>
class mlpack::ann::Sequential< InputDataType, OutputDataType, Residual, CustomLayers >
```

Implementation of the **Sequential** (p. 927) class.

The sequential class works as a feed-forward fully connected network container which plugs various layers together.

This class can also be used as a container for a residual block. In that case, the sizes of the input and output matrices of this class should be equal. A typedef has been added for use as a Residual<> class.

For more information, refer the following paper.

```
@article{He15,
  author    = {Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun},
  title     = {Deep Residual Learning for Image Recognition},
  year      = {2015},
  url       = {https://arxiv.org/abs/1512.03385},
  eprint    = {1512.03385},
}
```

Note: If this class is used as the first layer of a network, it should be preceded by IdentityLayer<>.

Note: This class should at least have two layers for a call to its **Gradient()** (p. 932) function.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>Residual</i>	If true, use the object as a Residual block.

Definition at line 83 of file layer_types.hpp.

39.136.2 Constructor & Destructor Documentation

39.136.2.1 Sequential()

```
Sequential (
    const bool model = true )
```

Create the **Sequential** (p. 927) object using the specified parameters.

Parameters

<i>model</i>	Expose the all network modules.
--------------	---------------------------------

39.136.2.2 ~Sequential()

~ **Sequential** ()

Destroy the **Sequential** (p. 927) object.

39.136.3 Member Function Documentation

39.136.3.1 Add() [1/2]

```
void Add (
    Args... args ) [inline]
```

Definition at line 126 of file sequential.hpp.

39.136.3.2 Add() [2/2]

```
void Add (
    LayerTypes< CustomLayers... > layer ) [inline]
```

Definition at line 133 of file sequential.hpp.

References Sequential< InputDataType, OutputDataType, Residual, CustomLayers >::DeleteModules().

39.136.3.3 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, using 3rd-order tensors as input, calculating the function f(x) by propagating x backwards through f.

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.136.3.4 DeleteModules()

```
void DeleteModules ( )
```

Referenced by Sequential< InputDataType, OutputDataType, Residual, CustomLayers >::Add().

39.136.3.5 Delta() [1/2]

```
arma::mat const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 167 of file sequential.hpp.

39.136.3.6 Delta() [2/2]

```
arma::mat& Delta ( ) [inline]
```

Modify the delta.

Definition at line 169 of file sequential.hpp.

39.136.3.7 Forward()

```
void Forward (
    arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.136.3.8 Gradient() [1/3]

```
void Gradient (
    arma::Mat< eT > && input,
    arma::Mat< eT > && error,
    arma::Mat< eT > && )
```

39.136.3.9 Gradient() [2/3]

```
arma::mat const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 172 of file sequential.hpp.

39.136.3.10 Gradient() [3/3]

```
arma::mat& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 174 of file sequential.hpp.

References Sequential< InputDataType, OutputDataType, Residual, CustomLayers >::serialize().

39.136.3.11 InputParameter() [1/2]

```
arma::mat const& InputParameter ( ) const [inline]
```

Get the input parameter.

Definition at line 157 of file sequential.hpp.

39.136.3.12 InputParameter() [2/2]

```
arma::mat& InputParameter ( ) [inline]
```

Modify the input parameter.

Definition at line 159 of file sequential.hpp.

39.136.3.13 Model()

```
std::vector< LayerTypes<CustomLayers...> >& Model ( ) [inline]
```

Return the model modules.

Definition at line 141 of file sequential.hpp.

39.136.3.14 OutputParameter() [1/2]

```
arma::mat const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 162 of file sequential.hpp.

39.136.3.15 OutputParameter() [2/2]

```
arma::mat& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 164 of file sequential.hpp.

39.136.3.16 Parameters() [1/2]

```
const arma::mat& Parameters ( ) const [inline]
```

Return the initial point for the optimization.

Definition at line 152 of file sequential.hpp.

39.136.3.17 Parameters() [2/2]

```
arma::mat& Parameters ( ) [inline]
```

Modify the initial point for the optimization.

Definition at line 154 of file sequential.hpp.

39.136.3.18 serialize()

```
void serialize (
    Archive & ,
    const unsigned int )
```

Serialize the layer.

Referenced by Sequential< InputDataType, OutputDataType, Residual, CustomLayers >::Gradient().

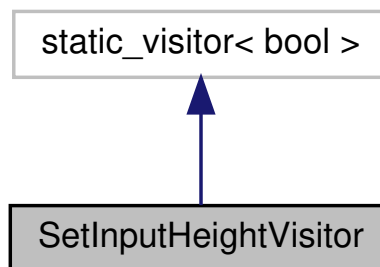
The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **layer_types.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **sequential.hpp**

39.137 SetInputHeightVisitor Class Reference

SetInputHeightVisitor (p. 934) updates the input height parameter with the given input height.

Inheritance diagram for SetInputHeightVisitor:

**Public Member Functions**

- **SetInputHeightVisitor** (const size_t inputHeight=0, const bool reset=false)
Update the input height parameter with the given input height.
- template<typename LayerType >
 bool **operator()** (LayerType *layer) const
Update the input height parameter.

39.137.1 Detailed Description

SetInputHeightVisitor (p. 934) updates the input height parameter with the given input height.

Definition at line 27 of file `set_input_height_visitor.hpp`.

39.137.2 Constructor & Destructor Documentation

39.137.2.1 SetInputHeightVisitor()

```
SetInputHeightVisitor (  
    const size_t inputHeight = 0,  
    const bool reset = false )
```

Update the input height parameter with the given input height.

39.137.3 Member Function Documentation

39.137.3.1 operator()()

```
bool operator() (  
    LayerType * layer ) const
```

Update the input height parameter.

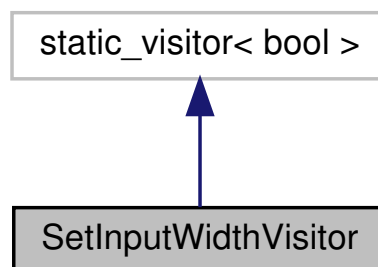
The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ set_input_height_visitor.hpp`

39.138 SetInputWidthVisitor Class Reference

SetInputWidthVisitor (p. 935) updates the input width parameter with the given input width.

Inheritance diagram for SetInputWidthVisitor:



Public Member Functions

- **SetInputWidthVisitor** (const size_t inputWidth=0, const bool reset=false)
Update the input width parameter with the given input width.
- template<typename LayerType >
 bool **operator()** (LayerType *layer) const
Update the input width parameter.

39.138.1 Detailed Description

SetInputWidthVisitor (p. 935) updates the input width parameter with the given input width.

Definition at line 27 of file set_input_width_visitor.hpp.

39.138.2 Constructor & Destructor Documentation

39.138.2.1 SetInputWidthVisitor()

```
SetInputWidthVisitor (
    const size_t inputWidth = 0,
    const bool reset = false )
```

Update the input width parameter with the given input width.

39.138.3 Member Function Documentation

39.138.3.1 operator()()

```
bool operator() (
    LayerType * layer ) const
```

Update the input width parameter.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **set_input_width_visitor.hpp**

39.139 SigmoidCrossEntropyError< InputDataType, OutputDataType > Class Template Reference

The **SigmoidCrossEntropyError** (p. 937) performance function measures the network's performance according to the cross-entropy function between the input and target distributions.

Public Member Functions

- **SigmoidCrossEntropyError** ()
*Create the **SigmoidCrossEntropyError** (p. 937) object.*
- template<typename InputType , typename TargetType , typename OutputType >
void **Backward** (const InputType &&input, const TargetType &&target, OutputType &&output)
Ordinary feed backward pass of a neural network.
- template<typename InputType , typename TargetType >
double **Forward** (const InputType &&input, const TargetType &&target)
Computes the Sigmoid CrossEntropy Error functions.
- OutputDataType & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.139.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::SigmoidCrossEntropyError< InputDataType, OutputDataType >
```

The **SigmoidCrossEntropyError** (p. 937) performance function measures the network's performance according to the cross-entropy function between the input and target distributions.

This function calculates the cross entropy given the real values instead of providing the sigmoid activations. The function uses this equivalent formulation: $\max(x, 0) - x * z + \log(1 + e^{-|x|})$ where x = input and z = target.

For more information, see the following paper.

```
@article{Janocha2017
  title   = {On Loss Functions for Deep Neural Networks in Classification},
  author  = {Katarzyna Janocha, Wojciech Marian Czarnecki},
  url     = {http://arxiv.org/abs/1702.05659},
  journal = {CoRR},
  eprint  = {arXiv:1702.05659},
  year    = {2017}
}
```

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 52 of file sigmoid_cross_entropy_error.hpp.

39.139.2 Constructor & Destructor Documentation

39.139.2.1 SigmoidCrossEntropyError()

```
SigmoidCrossEntropyError ( )
```

Create the **SigmoidCrossEntropyError** (p. 937) object.

39.139.3 Member Function Documentation

39.139.3.1 Backward()

```
void Backward (
    const InputType && input,
    const TargetType && target,
    OutputType && output ) [inline]
```

Ordinary feed backward pass of a neural network.

Parameters

<i>input</i>	The propagated input activation.
<i>target</i>	The target vector.
<i>output</i>	The calculated error.

39.139.3.2 Forward()

```
double Forward (
```

```
const InputType && input,
const TargetType && target ) [inline]
```

Computes the Sigmoid CrossEntropy Error functions.

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>target</i>	The target vector.

39.139.3.3 OutputParameter() [1/2]

```
OutputDataType& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 82 of file sigmoid_cross_entropy_error.hpp.

39.139.3.4 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 84 of file sigmoid_cross_entropy_error.hpp.

References SigmoidCrossEntropyError< InputDataType, OutputDataType >::serialize().

39.139.3.5 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by SigmoidCrossEntropyError< InputDataType, OutputDataType >::OutputParameter().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/ **sigmoid_cross_entropy_error.hpp**

39.140 SoftplusFunction Class Reference

The softplus function, defined by.

Static Public Member Functions

- static double **Deriv** (const double y)
Computes the first derivative of the softplus function.
- template<typename InputType , typename OutputType >
static void **Deriv** (const InputType &y, OutputType &x)
Computes the first derivatives of the softplus function.
- static double **Fn** (const double x)
Computes the softplus function.
- template<typename InputType , typename OutputType >
static void **Fn** (const InputType &x, OutputType &y)
Computes the softplus function.
- static double **Inv** (const double y)
Computes the inverse of the softplus function.
- template<typename InputType , typename OutputType >
static void **Inv** (const InputType &y, OutputType &x)
Computes the inverse of the softplus function.

39.140.1 Detailed Description

The softplus function, defined by.

$$\begin{aligned} f(x) &= \ln(1 + e^x) \\ f'(x) &= \frac{1}{1 + e^{-x}} \\ f^{-1}(y) &= \ln(e^y - 1) \end{aligned}$$

Definition at line 43 of file softplus_function.hpp.

39.140.2 Member Function Documentation

39.140.2.1 Deriv() [1/2]

```
static double Deriv (
    const double y ) [inline], [static]
```

Computes the first derivative of the softplus function.

Parameters

y	Input data.
-----	-------------

Returns $f'(x)$

Definition at line 80 of file softplus_function.hpp.

39.140.2.2 Deriv() [2/2]

```
static void Deriv (  
    const InputType & y,  
    OutputType & x ) [inline], [static]
```

Computes the first derivatives of the softplus function.

Parameters

y	Input activations.
x	The resulting derivatives.

Definition at line 92 of file softplus_function.hpp.

39.140.2.3 Fn() [1/2]

```
static double Fn (  
    const double x ) [inline], [static]
```

Computes the softplus function.

Parameters

x	Input data.
-----	-------------

Returns $f(x)$.

Definition at line 52 of file softplus_function.hpp.

Referenced by SoftplusFunction::Fn().

39.140.2.4 Fn() [2/2]

```
static void Fn (
    const InputType & x,
    OutputType & y ) [inline], [static]
```

Computes the softplus function.

Parameters

x	Input data.
y	The resulting output activation.

Definition at line 66 of file softplus_function.hpp.

References SoftplusFunction::Fn().

39.140.2.5 Inv() [1/2]

```
static double Inv (
    const double y ) [inline], [static]
```

Computes the inverse of the softplus function.

Parameters

y	Input data.
-----	-------------

Returns

$f^{\{-1\}}(y)$

Definition at line 103 of file softplus_function.hpp.

Referenced by SoftplusFunction::Inv().

39.140.2.6 Inv() [2/2]

```
static void Inv (
    const InputType & y,
    OutputType & x ) [inline], [static]
```

Computes the inverse of the softplus function.

Parameters

<i>y</i>	Input data.
<i>x</i>	The resulting inverse of the input data.

Definition at line 115 of file softplus_function.hpp.

References SoftplusFunction::Inv().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/ **softplus_function.hpp**

39.141 SoftsignFunction Class Reference

The softsign function, defined by.

Static Public Member Functions

- static double **Deriv** (const double y)
Computes the first derivative of the softsign function.
- template<typename InputVecType , typename OutputVecType >
static void **Deriv** (const InputVecType &y, OutputVecType &x)
Computes the first derivatives of the softsign function.
- static double **Fn** (const double x)
Computes the softsign function.
- template<typename InputVecType , typename OutputVecType >
static void **Fn** (const InputVecType &x, OutputVecType &y)
Computes the softsign function.
- static double **Inv** (const double y)
Computes the inverse of the softsign function.
- template<typename InputVecType , typename OutputVecType >
static void **Inv** (const InputVecType &y, OutputVecType &x)
Computes the inverse of the softsign function.

39.141.1 Detailed Description

The softsign function, defined by.

$$\begin{aligned} f(x) &= \frac{x}{1 + |x|} \\ f'(x) &= (1 - |x|)^2 \\ f(x) &= \begin{cases} -\frac{y}{y-1} & : x > 0 \\ \frac{x}{1+x} & : x \leq 0 \end{cases} \end{aligned}$$

Definition at line 47 of file softsign_function.hpp.

39.141.2 Member Function Documentation

39.141.2.1 Deriv() [1/2]

```
static double Deriv (
    const double y ) [inline], [static]
```

Computes the first derivative of the softsign function.

Parameters

<i>y</i>	Input data.
----------	-------------

Returns

$f'(x)$

Definition at line 84 of file softsign_function.hpp.

39.141.2.2 Deriv() [2/2]

```
static void Deriv (
    const InputVecType & y,
    OutputVecType & x ) [inline], [static]
```

Computes the first derivatives of the softsign function.

Parameters

<i>y</i>	Input activations.
<i>x</i>	The resulting derivatives.

Definition at line 96 of file softsign_function.hpp.

39.141.2.3 Fn() [1/2]

```
static double Fn (  
    const double x ) [inline], [static]
```

Computes the softsign function.

Parameters

<i>x</i>	Input data.
----------	-------------

Returns

$f(x)$.

Definition at line 56 of file softsign_function.hpp.

Referenced by SoftsignFunction::Fn().

39.141.2.4 Fn() [2/2]

```
static void Fn (  
    const InputVecType & x,  
    OutputVecType & y ) [inline], [static]
```

Computes the softsign function.

Parameters

<i>x</i>	Input data.
<i>y</i>	The resulting output activation.

Definition at line 70 of file softsign_function.hpp.

References SoftsignFunction::Fn().

39.141.2.5 Inv() [1/2]

```
static double Inv (
    const double y ) [inline], [static]
```

Computes the inverse of the softsign function.

Parameters

<i>y</i>	Input data.
----------	-------------

Returns

$f^{\{-1\}}(y)$

Definition at line 107 of file softsign_function.hpp.

Referenced by SoftsignFunction::Inv().

39.141.2.6 Inv() [2/2]

```
static void Inv (
    const InputVecType & y,
    OutputVecType & x ) [inline], [static]
```

Computes the inverse of the softsign function.

Parameters

<i>y</i>	Input data.
<i>x</i>	The resulting inverse of the input data.

Definition at line 122 of file softsign_function.hpp.

References SoftsignFunction::Inv().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/ **softsign_function.hpp**

39.142 SpikeSlabRBM Class Reference

For more information, see the following paper:

39.142.1 Detailed Description

For more information, see the following paper:

```
@article{Courville11,
  author = {Aaron Courville, James Bergstra and Yoshua Bengio},
  title  = {A Spike and Slab Restricted Boltzmann Machine},
  year   = {2011},
  url    = {http://proceedings.mlr.press/v15/courville11a/courville11a.pdf}
}
```

Definition at line 44 of file rbm_policies.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/rbm/ **rbm_policies.hpp**

39.143 Subview< InputDataType, OutputDataType > Class Template Reference

Implementation of the subview layer.

Public Member Functions

- **Subview** (const size_t inSize=1, const size_t beginRow=0, const size_t endRow=0, const size_t beginCol=0, const size_t endCol=0)
*Create the **Subview** (p. 947) layer object using the specified range of input to accept.*
- template<typename eT >
void **Backward** (arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename InputType, typename OutputType >
void **Forward** (InputType &&input, OutputType &&output)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- OutputDataType const & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the layer.

39.143.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::Subview< InputDataType, OutputDataType >
```

Implementation of the subview layer.

The subview layer modifies the input to a submatrix of required size.

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 34 of file subview.hpp.

39.143.2 Constructor & Destructor Documentation

39.143.2.1 Subview()

```
Subview (
    const size_t inSize = 1,
    const size_t beginRow = 0,
    const size_t endRow = 0,
    const size_t beginCol = 0,
    const size_t endCol = 0 ) [inline]
```

Create the **Subview** (p. 947) layer object using the specified range of input to accept.

Parameters

<i>inSize</i>	Width of sample.
<i>beginRow</i>	Starting row index.
<i>endRow</i>	Ending row index.
<i>beginCol</i>	Starting column index.
<i>endCol</i>	Ending column index.

Definition at line 47 of file subview.hpp.

39.143.3 Member Function Documentation

39.143.3.1 Backward()

```
void Backward (
    arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g ) [inline]
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

Definition at line 115 of file subview.hpp.

39.143.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 128 of file subview.hpp.

39.143.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 130 of file subview.hpp.

39.143.3.4 Forward()

```
void Forward (
    InputType && input,
    OutputType && output ) [inline]
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

Definition at line 69 of file subview.hpp.

39.143.3.5 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 123 of file subview.hpp.

39.143.3.6 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 125 of file subview.hpp.

39.143.3.7 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the layer.

Definition at line 136 of file subview.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **subview.hpp**

39.144 SVDConvolution< BorderMode > Class Template Reference

Computes the two-dimensional convolution using singular value decomposition.

Static Public Member Functions

- template<typename eT >
static void **Convolution** (const arma::Mat< eT > &input, const arma::Mat< eT > &filter, arma::Mat< eT > &output)
- template<typename eT >
static void **Convolution** (const arma::Cube< eT > &input, const arma::Cube< eT > &filter, arma::Cube< eT > &output)
- template<typename eT >
static void **Convolution** (const arma::Mat< eT > &input, const arma::Cube< eT > &filter, arma::Cube< eT > &output)
- template<typename eT >
static void **Convolution** (const arma::Cube< eT > &input, const arma::Mat< eT > &filter, arma::Cube< eT > &output)

39.144.1 Detailed Description

```
template<typename BorderMode = FullConvolution>
class mlpack::ann::SVDConvolution< BorderMode >
```

Computes the two-dimensional convolution using singular value decomposition.

This class allows specification of the type of the border type. The convolution can be computed with the valid border type of the full border type (default).

FullConvolution (p. 714): returns the full two-dimensional convolution. **ValidConvolution** (p. 967): returns only those parts of the convolution that are computed without the zero-padded edges.

Template Parameters

<i>BorderMode</i>	Type of the border mode (FullConvolution (p. 714) or ValidConvolution (p. 967)).
-------------------	---

Definition at line 38 of file svd_convolution.hpp.

39.144.2 Member Function Documentation

39.144.2.1 Convolution() [1/4]

```
static void Convolution (
    const arma::Mat< eT > & input,
    const arma::Mat< eT > & filter,
    arma::Mat< eT > & output ) [inline], [static]
```

Definition at line 56 of file svd_convolution.hpp.

References NaiveConvolution< BorderMode >::Convolution(), and FFTConvolution< BorderMode, padLastDim >::Convolution().

Referenced by SVDCConvolution< BorderMode >::Convolution().

39.144.2.2 Convolution() [2/4]

```
static void Convolution (
    const arma::Cube< eT > & input,
    const arma::Cube< eT > & filter,
    arma::Cube< eT > & output ) [inline], [static]
```

Definition at line 122 of file svd_convolution.hpp.

References SVDCConvolution< BorderMode >::Convolution().

39.144.2.3 Convolution() [3/4]

```
static void Convolution (
    const arma::Mat< eT > & input,
    const arma::Cube< eT > & filter,
    arma::Cube< eT > & output ) [inline], [static]
```

Definition at line 152 of file svd_convolution.hpp.

References SVDCConvolution< BorderMode >::Convolution().

39.144.2.4 Convolution() [4/4]

```
static void Convolution (
    const arma::Cube< eT > & input,
    const arma::Mat< eT > & filter,
    arma::Cube< eT > & output ) [inline], [static]
```

Definition at line 181 of file svd_convolution.hpp.

References SVDCConvolution< BorderMode >::Convolution().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/ **svd_convolution.hpp**

39.145 SwishFunction Class Reference

The swish function, defined by.

Static Public Member Functions

- static double **Deriv** (const double y)
Computes the first derivative of the swish function.
- template<typename InputVecType , typename OutputVecType >
static void **Deriv** (const InputVecType &y, OutputVecType &x)
Computes the first derivatives of the swish function.
- static double **Fn** (const double x)
Computes the swish function.
- template<typename eT >
static void **Fn** (const arma::Mat< eT > &x, arma::Mat< eT > &y)
Computes the swish function using a matrix as input.
- template<typename InputVecType , typename OutputVecType >
static void **Fn** (const InputVecType &x, OutputVecType &y)
Computes the swish function.

39.145.1 Detailed Description

The swish function, defined by.

$$\begin{aligned} f(x) &= x \cdot \sigma(x) \\ f'(x) &= f(x) + \sigma(x)(1 - f(x)) \\ \sigma(x) &= \frac{1}{1 + e^{-x}} \end{aligned}$$

Definition at line 30 of file swish_function.hpp.

39.145.2 Member Function Documentation

39.145.2.1 Deriv() [1/2]

```
static double Deriv (
    const double y ) [inline], [static]
```

Computes the first derivative of the swish function.

Parameters

y	Input data.
-----	-------------

Returns $f'(x)$

Definition at line 77 of file swish_function.hpp.

39.145.2.2 Deriv() [2/2]

```
static void Deriv (  
    const InputVecType & y,  
    OutputVecType & x ) [inline], [static]
```

Computes the first derivatives of the swish function.

Parameters

y	Input activations.
x	The resulting derivatives.

Definition at line 90 of file swish_function.hpp.

39.145.2.3 Fn() [1/3]

```
static double Fn (  
    const double x ) [inline], [static]
```

Computes the swish function.

Parameters

x	Input data.
-----	-------------

Returns $f(x)$.

Definition at line 39 of file swish_function.hpp.

Referenced by SwishFunction::Fn().

39.145.2.4 Fn() [2/3]

```
static void Fn (  
    const arma::Mat< eT > & x,  
    arma::Mat< eT > & y ) [inline], [static]
```

Computes the swish function using a matrix as input.

Parameters

<i>x</i>	Input data.
<i>y</i>	The resulting output activation.

Definition at line 51 of file swish_function.hpp.

39.145.2.5 Fn() [3/3]

```
static void Fn (  
    const InputVecType & x,  
    OutputVecType & y ) [inline], [static]
```

Computes the swish function.

Parameters

<i>x</i>	Input data.
<i>y</i>	The resulting output activation.

Definition at line 63 of file swish_function.hpp.

References SwishFunction::Fn().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/ **swish_function.hpp**

39.146 TanhFunction Class Reference

The tanh function, defined by.

Static Public Member Functions

- static double **Deriv** (const double y)
Computes the first derivative of the tanh function.
- template<typename InputVecType , typename OutputVecType >
static void **Deriv** (const InputVecType &y, OutputVecType &x)
Computes the first derivatives of the tanh function.
- static double **Fn** (const double x)
Computes the tanh function.
- template<typename InputVecType , typename OutputVecType >
static void **Fn** (const InputVecType &x, OutputVecType &y)
Computes the tanh function.
- static double **Inv** (const double y)
Computes the inverse of the tanh function.
- template<typename InputVecType , typename OutputVecType >
static void **Inv** (const InputVecType &y, OutputVecType &x)
Computes the inverse of the tanh function.

39.146.1 Detailed Description

The tanh function, defined by.

$$\begin{aligned} f(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}} \\ f'(x) &= 1 - \tanh^2(x) \\ f^{-1}(x) &= \arctan(x) \end{aligned}$$

Definition at line 29 of file tanh_function.hpp.

39.146.2 Member Function Documentation

39.146.2.1 Deriv() [1/2]

```
static double Deriv (
    const double y ) [inline], [static]
```

Computes the first derivative of the tanh function.

Parameters

<i>y</i>	Input data.
----------	-------------

Returns $f'(x)$

Definition at line 61 of file tanh_function.hpp.

39.146.2.2 Deriv() [2/2]

```
static void Deriv (  
    const InputVecType & y,  
    OutputVecType & x ) [inline], [static]
```

Computes the first derivatives of the tanh function.

Parameters

y	Input data.
x	The resulting derivatives.

Definition at line 73 of file tanh_function.hpp.

39.146.2.3 Fn() [1/2]

```
static double Fn (  
    const double x ) [inline], [static]
```

Computes the tanh function.

Parameters

x	Input data.
---	-------------

Returns $f(x)$.

Definition at line 38 of file tanh_function.hpp.

39.146.2.4 Fn() [2/2]

```
static void Fn (
    const InputVecType & x,
    OutputVecType & y ) [inline], [static]
```

Computes the tanh function.

Parameters

<i>x</i>	Input data.
<i>y</i>	The resulting output activation.

Definition at line 50 of file tanh_function.hpp.

39.146.2.5 Inv() [1/2]

```
static double Inv (
    const double y ) [inline], [static]
```

Computes the inverse of the tanh function.

Parameters

<i>y</i>	Input data.
----------	-------------

Returns

$f^{-1}(x)$

Definition at line 84 of file tanh_function.hpp.

39.146.2.6 Inv() [2/2]

```
static void Inv (
    const InputVecType & y,
    OutputVecType & x ) [inline], [static]
```

Computes the inverse of the tanh function.

Parameters

y	Input data.
x	The resulting inverse of the input data.

Definition at line 96 of file tanh_function.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/ **tanh_function.hpp**

39.147 TransposedConvolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, InputDataType, OutputDataType > Class Template Reference

Implementation of the Transposed **Convolution** (p. 643) class.

Public Member Functions

- **TransposedConvolution** ()
*Create the Transposed **Convolution** (p. 643) object.*
- **TransposedConvolution** (const size_t inSize, const size_t outSize, const size_t kW, const size_t kH, const size_t dW=1, const size_t dH=1, const size_t padW=0, const size_t padH=0, const size_t inputWidth=0, const size_t inputHeight=0)
*Create the Transposed **Convolution** (p. 643) object using the specified number of input maps, output maps, filter size, stride and padding parameter.*
- template<typename eT >
void **Backward** (const arma::Mat< eT > &&, arma::Mat< eT > &&gy, arma::Mat< eT > &&g)
Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .
- OutputDataType const & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.
- template<typename eT >
void **Forward** (const arma::Mat< eT > &&input, arma::Mat< eT > &&output)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- template<typename eT >
void **Gradient** (const arma::Mat< eT > &&, arma::Mat< eT > &&error, arma::Mat< eT > &&gradient)
Get the gradient.
- OutputDataType const & **Gradient** () const
Modify the gradient.
- size_t const & **InputHeight** () const
Get the input height.

- `size_t & InputHeight ()`
Modify the input height.
- `InputDataType const & InputParameter () const`
Get the input parameter.
- `InputDataType & InputParameter ()`
Modify the input parameter.
- `size_t const & InputWidth () const`
Get the input width.
- `size_t & InputWidth ()`
Modify input the width.
- `size_t const & OutputHeight () const`
Get the output height.
- `size_t & OutputHeight ()`
Modify the output height.
- `OutputDataType const & OutputParameter () const`
Get the output parameter.
- `OutputDataType & OutputParameter ()`
Modify the output parameter.
- `size_t const & OutputWidth () const`
Get the output width.
- `size_t & OutputWidth ()`
Modify the output width.
- `OutputDataType const & Parameters () const`
Get the parameters.
- `OutputDataType & Parameters ()`
Modify the parameters.
- `void Reset ()`
- `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`
Serialize the layer.

39.147.1 Detailed Description

```
template<typename ForwardConvolutionRule = NaiveConvolution<ValidConvolution>, typename BackwardConvolutionRule =
NaiveConvolution<FullConvolution>, typename GradientConvolutionRule = NaiveConvolution<ValidConvolution>, typename
InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::TransposedConvolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, Input<↔
DataType, OutputDataType >
```

Implementation of the Transposed **Convolution** (p. 643) class.

The Transposed **Convolution** (p. 643) class represents a single layer of a neural network.

Template Parameters

<i>ForwardConvolutionRule</i>	Convolution (p. 643) to perform forward process.
<i>BackwardConvolutionRule</i>	Convolution (p. 643) to perform backward process.
<i>GradientConvolutionRule</i>	Convolution (p. 643) to calculate gradient.
<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 120 of file layer_types.hpp.

39.147.2 Constructor & Destructor Documentation

39.147.2.1 TransposedConvolution() [1/2]

TransposedConvolution ()

Create the Transposed **Convolution** (p. 643) object.

39.147.2.2 TransposedConvolution() [2/2]

```
TransposedConvolution (
    const size_t inSize,
    const size_t outSize,
    const size_t kW,
    const size_t kH,
    const size_t dW = 1,
    const size_t dH = 1,
    const size_t padW = 0,
    const size_t padH = 0,
    const size_t inputWidth = 0,
    const size_t inputHeight = 0 )
```

Create the Transposed **Convolution** (p. 643) object using the specified number of input maps, output maps, filter size, stride and padding parameter.

Parameters

<i>inSize</i>	The number of input maps.
<i>outSize</i>	The number of output maps.
<i>kW</i>	Width of the filter/kernel.
<i>kH</i>	Height of the filter/kernel.
<i>dW</i>	Stride of filter application in the x direction.
<i>dH</i>	Stride of filter application in the y direction.
<i>padW</i>	Padding width of the input.
<i>padH</i>	Padding height of the input.
<i>inputWidth</i>	The width of the input data.
<i>inputHeight</i>	The height of the input data.

39.147.3 Member Function Documentation

39.147.3.1 Backward()

```
void Backward (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && gy,
    arma::Mat< eT > && g )
```

Ordinary feed backward pass of a neural network, calculating the function $f(x)$ by propagating x backwards through f .

Using the results from the feed forward pass.

Parameters

<i>input</i>	The propagated input activation.
<i>gy</i>	The backpropagated error.
<i>g</i>	The calculated gradient.

39.147.3.2 Delta() [1/2]

```
OutputDataType const& Delta ( ) const [inline]
```

Get the delta.

Definition at line 136 of file `transposed_convolution.hpp`.

39.147.3.3 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 138 of file `transposed_convolution.hpp`.

39.147.3.4 Forward()

```
void Forward (
    const arma::Mat< eT > && input,
    arma::Mat< eT > && output )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data used for evaluating the specified function.
<i>output</i>	Resulting output activation.

39.147.3.5 Gradient() [1/3]

```
void Gradient (
    const arma::Mat< eT > && ,
    arma::Mat< eT > && error,
    arma::Mat< eT > && gradient )
```

39.147.3.6 Gradient() [2/3]

```
OutputDataType const& Gradient ( ) const [inline]
```

Get the gradient.

Definition at line 141 of file `transposed_convolution.hpp`.

39.147.3.7 Gradient() [3/3]

```
OutputDataType& Gradient ( ) [inline]
```

Modify the gradient.

Definition at line 143 of file `transposed_convolution.hpp`.

39.147.3.8 InputHeight() [1/2]

```
size_t const& InputHeight ( ) const [inline]
```

Get the input height.

Definition at line 151 of file `transposed_convolution.hpp`.

39.147.3.9 InputHeight() [2/2]

```
size_t& InputHeight ( ) [inline]
```

Modify the input height.

Definition at line 153 of file `transposed_convolution.hpp`.

39.147.3.10 InputParameter() [1/2]

```
InputDataType const& InputParameter ( ) const [inline]
```

Get the input parameter.

Definition at line 126 of file `transposed_convolution.hpp`.

39.147.3.11 InputParameter() [2/2]

```
InputDataType& InputParameter ( ) [inline]
```

Modify the input parameter.

Definition at line 128 of file `transposed_convolution.hpp`.

39.147.3.12 InputWidth() [1/2]

```
size_t const& InputWidth ( ) const [inline]
```

Get the input width.

Definition at line 146 of file `transposed_convolution.hpp`.

39.147.3.13 InputWidth() [2/2]

```
size_t& InputWidth ( ) [inline]
```

Modify input the width.

Definition at line 148 of file `transposed_convolution.hpp`.

39.147.3.14 OutputHeight() [1/2]

```
size_t const& OutputHeight ( ) const [inline]
```

Get the output height.

Definition at line 161 of file `transposed_convolution.hpp`.

39.147.3.15 OutputHeight() [2/2]

```
size_t& OutputHeight ( ) [inline]
```

Modify the output height.

Definition at line 163 of file `transposed_convolution.hpp`.

References `TransposedConvolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, InputDataType, OutputDataType >::serialize()`.

39.147.3.16 OutputParameter() [1/2]

```
OutputDataType const& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 131 of file `transposed_convolution.hpp`.

39.147.3.17 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 133 of file `transposed_convolution.hpp`.

39.147.3.18 OutputWidth() [1/2]

```
size_t const& OutputWidth ( ) const [inline]
```

Get the output width.

Definition at line 156 of file `transposed_convolution.hpp`.

39.147.3.19 OutputWidth() [2/2]

```
size_t& OutputWidth ( ) [inline]
```

Modify the output width.

Definition at line 158 of file `transposed_convolution.hpp`.

39.147.3.20 Parameters() [1/2]

```
OutputDataType const& Parameters ( ) const [inline]
```

Get the parameters.

Definition at line 121 of file `transposed_convolution.hpp`.

39.147.3.21 Parameters() [2/2]

```
OutputDataType& Parameters ( ) [inline]
```

Modify the parameters.

Definition at line 123 of file `transposed_convolution.hpp`.

39.147.3.22 Reset()

```
void Reset ( )
```

39.147.3.23 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the layer.

Referenced by TransposedConvolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, InputDataType, OutputDataType >::OutputHeight().

The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **layer_types.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **transposed_convolution.hpp**

39.148 ValidConvolution Class Reference

39.148.1 Detailed Description

Definition at line 28 of file border_modes.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/ **border_modes.hpp**

39.149 VRClassReward< InputDataType, OutputDataType > Class Template Reference

Implementation of the variance reduced classification reinforcement layer.

Public Member Functions

- **VRClassReward** (const double scale=1, const bool sizeAverage=true)
*Create the **VRClassReward** (p. 967) object.*
- template<class LayerType, class... Args>
void **Add** (Args... args)
- void **Add** (**LayerTypes**<> layer)
- template<typename InputType, typename TargetType, typename OutputType >
void **Backward** (const InputType &&input, const TargetType &&target, OutputType &&output)
Ordinary feed backward pass of a neural network.
- OutputDataType & **Delta** () const
Get the delta.
- OutputDataType & **Delta** ()
Modify the delta.

- bool **Deterministic** () const
Get the value of the deterministic parameter.
- bool & **Deterministic** ()
Modify the value of the deterministic parameter.
- template<typename InputType , typename TargetType >
double **Forward** (const InputType &&input, const TargetType &&target)
Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .
- OutputDataType & **OutputParameter** () const
Get the output parameter.
- OutputDataType & **OutputParameter** ()
Modify the output parameter.
- template<typename Archive >
void **serialize** (Archive &, const unsigned int)
Serialize the layer.

39.149.1 Detailed Description

```
template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat>
class mlpack::ann::VRClassReward< InputDataType, OutputDataType >
```

Implementation of the variance reduced classification reinforcement layer.

This layer is meant to be used in combination with the reinforce normal layer (ReinforceNormalLayer), which expects that an reward: (1 for success, 0 otherwise).

Template Parameters

<i>InputDataType</i>	Type of the input data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).
<i>OutputDataType</i>	Type of the output data (arma::colvec, arma::mat, arma::sp_mat or arma::cube).

Definition at line 64 of file layer_types.hpp.

39.149.2 Constructor & Destructor Documentation

39.149.2.1 VRClassReward()

```
VRClassReward (
    const double scale = 1,
    const bool sizeAverage = true )
```

Create the **VRClassReward** (p. 967) object.

Parameters

<i>scale</i>	Parameter used to scale the reward.
<i>sizeAverage</i>	Take the average over all batches.

39.149.3 Member Function Documentation

39.149.3.1 Add() [1/2]

```
void Add (
    Args... args ) [inline]
```

Definition at line 97 of file vr_class_reward.hpp.

39.149.3.2 Add() [2/2]

```
void Add (
    LayerTypes<> layer ) [inline]
```

Definition at line 104 of file vr_class_reward.hpp.

References VRClassReward< InputDataType, OutputDataType >::serialize().

39.149.3.3 Backward()

```
void Backward (
    const InputType && input,
    const TargetType && target,
    OutputType && output )
```

Ordinary feed backward pass of a neural network.

The negative log likelihood layer expects that the input contains log-probabilities for each class. The layer also expects a class index, in the range between 1 and the number of classes, as target when calling the Forward function.

Parameters

<i>input</i>	The propagated input activation.
<i>target</i>	The target vector, that contains the class index in the range between 1 and the number of classes.
<i>output</i>	The calculated error.

39.149.3.4 Delta() [1/2]

```
OutputDataType& Delta ( ) const [inline]
```

Get the delta.

Definition at line 82 of file `vr_class_reward.hpp`.

39.149.3.5 Delta() [2/2]

```
OutputDataType& Delta ( ) [inline]
```

Modify the delta.

Definition at line 84 of file `vr_class_reward.hpp`.

39.149.3.6 Deterministic() [1/2]

```
bool Deterministic ( ) const [inline]
```

Get the value of the deterministic parameter.

Definition at line 87 of file `vr_class_reward.hpp`.

39.149.3.7 Deterministic() [2/2]

```
bool& Deterministic ( ) [inline]
```

Modify the value of the deterministic parameter.

Definition at line 89 of file `vr_class_reward.hpp`.

39.149.3.8 Forward()

```
double Forward (
    const InputType && input,
    const TargetType && target )
```

Ordinary feed forward pass of a neural network, evaluating the function $f(x)$ by propagating the activity forward through f .

Parameters

<i>input</i>	Input data that contains the log-probabilities for each class.
<i>target</i>	The target vector, that contains the class index in the range between 1 and the number of classes.

39.149.3.9 OutputParameter() [1/2]

```
OutputDataType& OutputParameter ( ) const [inline]
```

Get the output parameter.

Definition at line 77 of file vr_class_reward.hpp.

39.149.3.10 OutputParameter() [2/2]

```
OutputDataType& OutputParameter ( ) [inline]
```

Modify the output parameter.

Definition at line 79 of file vr_class_reward.hpp.

39.149.3.11 serialize()

```
void serialize (
    Archive & ,
    const unsigned int )
```

Serialize the layer.

Referenced by VRClassReward< InputDataType, OutputDataType >::Add().

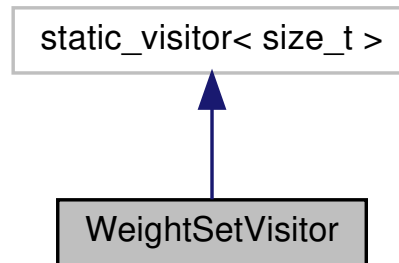
The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **layer_types.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/ **vr_class_reward.hpp**

39.150 WeightSetVisitor Class Reference

WeightSetVisitor (p. 972) update the module parameters given the parameters set.

Inheritance diagram for WeightSetVisitor:



Public Member Functions

- **WeightSetVisitor** (arma::mat &&weight, const size_t offset=0)
Update the parameters given the parameters set and offset.
- template<typename LayerType >
size_t **operator()** (LayerType *layer) const
Update the parameters set.

39.150.1 Detailed Description

WeightSetVisitor (p. 972) update the module parameters given the parameters set.

Definition at line 26 of file weight_set_visitor.hpp.

39.150.2 Constructor & Destructor Documentation

39.150.2.1 WeightSetVisitor()

```

WeightSetVisitor (
    arma::mat && weight,
    const size_t offset = 0 )
  
```

Update the parameters given the parameters set and offset.

39.150.3 Member Function Documentation

39.150.3.1 operator()()

```
size_t operator() (
    LayerType * layer ) const
```

Update the parameters set.

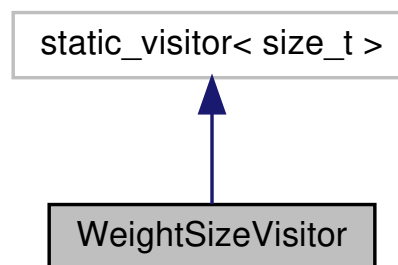
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **weight_set_visitor.hpp**

39.151 WeightSizeVisitor Class Reference

WeightSizeVisitor (p. 973) returns the number of weights of the given module.

Inheritance diagram for WeightSizeVisitor:



Public Member Functions

- template<typename LayerType >
size_t **operator()** (LayerType *layer) const
Return the number of weights.

39.151.1 Detailed Description

WeightSizeVisitor (p. 973) returns the number of weights of the given module.

Definition at line 27 of file weight_size_visitor.hpp.

39.151.2 Member Function Documentation

39.151.2.1 operator()()

```
size_t operator() (
    LayerType * layer ) const
```

Return the number of weights.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/ **weight_size_visitor.hpp**

39.152 Backtrace Class Reference

Provides a backtrace.

Public Member Functions

- **Backtrace** ()
Constructor initialize fields and call `GetAddress` to retrieve addresses for each frame of backtrace.
- std::string **ToString** ()
Returns string of backtrace.

39.152.1 Detailed Description

Provides a backtrace.

The **Backtrace** (p. 974) class retrieve addresses of each called function from the stack and decode file name, function & line number. Retrieved information can be printed in form:

```
[b]: (count) /directory/to/file.cpp:function(args):line_number
```

Backtrace (p. 974) is printed always when **Log::Assert** (p. 1539) failed. An example is given below.

```
if (!someImportantCondition())
{
    Log::Fatal << "someImportantCondition() is not satisfied! Terminating.";
    Log::Fatal << std::endl;
}
```

Note

Log::Assert (p. 1539) will not be shown when compiling in non-debug mode.

See also

PrefixOutStream, **Log** (p. 1538)

Definition at line 46 of file backtrace.hpp.

39.152.2 Constructor & Destructor Documentation

39.152.2.1 Backtrace()

Backtrace ()

Constructor initialize fields and call GetAddress to retrieve addresses for each frame of backtrace.

Parameters

<i>maxDepth</i>	Maximum depth of backtrace. Default 32 steps.
-----------------	---

39.152.3 Member Function Documentation

39.152.3.1 ToString()

`std::string ToString ()`

Returns string of backtrace.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **backtrace.hpp**

39.153 CLIOption< N > Class Template Reference

A static object whose constructor registers a parameter with the **CLI** (p. 1117) class.

Public Member Functions

- **CLIOption** (const N defaultValue, const std::string &identifier, const std::string &description, const std::string &alias, const std::string &cppName, const bool required=false, const bool input=true, const bool noTranspose=false, const std::string &="")

Construct an Option object.

39.153.1 Detailed Description

```
template<typename N>
class mlpack::bindings::cli::CLIOption< N >
```

A static object whose constructor registers a parameter with the **CLI** (p. 1117) class.

This should not be used outside of **CLI** (p. 1117) itself, and you should use the **PARAM_FLAG()** (p. 301), **PARAM_DOUBLE()**, **PARAM_INT()**, **PARAM_STRING()**, or other similar macros to declare these objects instead of declaring them directly.

See also

core/util/cli.hpp (p. 2694), **mlpack::CLI** (p. 1117)

Definition at line 47 of file cli_option.hpp.

39.153.2 Constructor & Destructor Documentation

39.153.2.1 CLIOption()

```
CLIOption (
    const N defaultValue,
    const std::string & identifier,
    const std::string & description,
    const std::string & alias,
    const std::string & cppName,
    const bool required = false,
    const bool input = true,
    const bool noTranspose = false,
    const std::string & testName ) [inline]
```

Construct an Option object.

When constructed, it will register itself with **CLI** (p. 1117).

Parameters

<i>defaultValue</i>	Default value this parameter will be initialized to (for flags, this should be false, for instance).
<i>identifier</i>	The name of the option (no dashes in front; for <code>-help</code> , we would pass "help").
<i>description</i>	A short string describing the option.
<i>alias</i>	Short name of the parameter. "" for no alias.
<i>cppName</i>	Name of the C++ type of this parameter (i.e. "int").
<i>required</i>	Whether or not the option is required at runtime.
<i>input</i>	Whether or not the option is an input option.
<i>noTranspose</i>	If the parameter is a matrix and this is true, then the matrix will not be transposed on loading.
<i>testName</i>	Is not used and added for compatibility reasons.

Definition at line 67 of file `cli_option.hpp`.

References `CLI::Add()`, `ParamData::alias`, `BASH_CLEAR`, `BASH_RED`, `ParamData::cppType`, `ParamData::desc`, `CLI::functionMap`, `CLI::GetSingleton()`, `ParamData::input`, `ParamData::loaded`, `mlpack::bindings::cli::MapParameterName()`, `ParamData::name`, `ParamData::noTranspose`, `CLI::Parameters()`, `ParamData::persistent`, `ParamData::required`, `ParamData::tname`, `TYPENAME`, `ParamData::value`, and `ParamData::wasPassed`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/ cli_option.hpp`

39.154 `ParameterType< T >` Struct Template Reference

Utility struct to return the type that `boost::program_options` should accept for a given input type.

Public Types

- typedef `ParameterTypeDeducer< data::HasSerialize< T >::value, T >:: type type`

39.154.1 Detailed Description

```
template<typename T>
struct mlpack::bindings::cli::ParameterType< T >
```

Utility struct to return the type that `boost::program_options` should accept for a given input type.

In general, there is no change from the input type, but in some cases this may be another type.

Definition at line 42 of file `parameter_type.hpp`.

39.154.2 Member Typedef Documentation

39.154.2.1 `type`

```
typedef ParameterTypeDeducer< data::HasSerialize<T>::value, T>:: type type
```

Definition at line 45 of file `parameter_type.hpp`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/ parameter_type.hpp`

39.155 `ParameterType< arma::Col< eT > >` Struct Template Reference

For vector types, `boost::program_options` will accept a `std::string`, not an `arma::Col<eT>` (since it is not clear how to specify a vector on the command-line).

Public Types

- `typedef std::string type`

39.155.1 Detailed Description

```
template<typename eT>
struct mlpack::bindings::cli::ParameterType< arma::Col< eT > >
```

For vector types, `boost::program_options` will accept a `std::string`, not an `arma::Col<eT>` (since it is not clear how to specify a vector on the command-line).

Definition at line 54 of file `parameter_type.hpp`.

39.155.2 Member Typedef Documentation

39.155.2.1 `type`

```
typedef std::string type
```

Definition at line 56 of file `parameter_type.hpp`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/parameter_type.hpp`

39.156 `ParameterType< arma::Mat< eT > >` Struct Template Reference

For matrix types, `boost::program_options` will accept a `std::string`, not an `arma::mat` (since it is not clear how to specify a matrix on the command-line).

Public Types

- `typedef std::string type`

39.156.1 Detailed Description

```
template<typename eT>
struct mlpack::bindings::cli::ParameterType< arma::Mat< eT > >
```

For matrix types, `boost::program_options` will accept a `std::string`, not an `arma::mat` (since it is not clear how to specify a matrix on the command-line).

Definition at line 77 of file `parameter_type.hpp`.

39.156.2 Member Typedef Documentation

39.156.2.1 `type`

```
typedef std::string type
```

Definition at line 79 of file `parameter_type.hpp`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/parameter_type.hpp`

39.157 `ParameterType< arma::Row< eT > >` Struct Template Reference

For row vector types, `boost::program_options` will accept a `std::string`, not an `arma::Row<eT>` (since it is not clear how to specify a vector on the command-line).

Public Types

- `typedef std::string type`

39.157.1 Detailed Description

```
template<typename eT>
struct mlpack::bindings::cli::ParameterType< arma::Row< eT > >
```

For row vector types, `boost::program_options` will accept a `std::string`, not an `arma::Row<eT>` (since it is not clear how to specify a vector on the command-line).

Definition at line 66 of file `parameter_type.hpp`.

39.157.2 Member Typedef Documentation

39.157.2.1 type

```
typedef std::string type
```

Definition at line 68 of file parameter_type.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/ **parameter_type.hpp**

39.158 **ParameterType**< std::tuple< mlpack::data::DatasetMapper< PolicyType, std::string >, arma::Mat< eT > > > **Struct Template Reference**

For matrix+dataset info types, we should accept a std::string.

Public Types

- typedef std::string **type**

39.158.1 Detailed Description

```
template<typename eT, typename PolicyType>
struct mlpack::bindings::cli::ParameterType< std::tuple< mlpack::data::DatasetMapper< PolicyType, std::string >, arma::Mat< eT > > >
> > >
```

For matrix+dataset info types, we should accept a std::string.

Definition at line 86 of file parameter_type.hpp.

39.158.2 Member Typedef Documentation

39.158.2.1 type

```
typedef std::string type
```

Definition at line 89 of file parameter_type.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/ **parameter_type.hpp**

39.159 ParameterTypeDeducer< HasSerialize, T > Struct Template Reference

Public Types

- typedef T **type**

39.159.1 Detailed Description

```
template<bool HasSerialize, typename T>  
struct mlpack::bindings::cli::ParameterTypeDeducer< HasSerialize, T >
```

Definition at line 24 of file parameter_type.hpp.

39.159.2 Member Typedef Documentation

39.159.2.1 type

```
typedef T type
```

Definition at line 26 of file parameter_type.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/ **parameter_type.hpp**

39.160 ParameterTypeDeducer< true, T > Struct Template Reference

Public Types

- typedef std::string **type**

39.160.1 Detailed Description

```
template<typename T>
struct mlpack::bindings::cli::ParameterTypeDeducer< true, T >
```

Definition at line 31 of file parameter_type.hpp.

39.160.2 Member Typedef Documentation

39.160.2.1 type

```
typedef std::string  type
```

Definition at line 33 of file parameter_type.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/ **parameter_type.hpp**

39.161 ProgramDoc Class Reference

A static object whose constructor registers program documentation with the **CLI** (p. 1117) class.

Public Member Functions

- **ProgramDoc** (const std::string & **programName**, const std::string & **documentation**)
*Construct a **ProgramDoc** (p. 982) object.*

Public Attributes

- std::string **documentation**
Documentation for what the program does.
- std::string **programName**
The name of the program.

39.161.1 Detailed Description

A static object whose constructor registers program documentation with the **CLI** (p. 1117) class.

This should not be used outside of **CLI** (p. 1117) itself, and you should use the **PROGRAM_INFO()** (p. 2733) macro to declare these objects. Only one **ProgramDoc** (p. 982) object should ever exist.

See also

core/util/cli.hpp (p. 2694), **mlpack::CLI** (p. 1117)

Definition at line 174 of file cli_option.hpp.

39.161.2 Constructor & Destructor Documentation

39.161.2.1 ProgramDoc()

```
ProgramDoc (
    const std::string & programName,
    const std::string & documentation )
```

Construct a **ProgramDoc** (p. 982) object.

When constructed, it will register itself with **CLI** (p. 1117).

Parameters

<i>programName</i>	Short string representing the name of the program.
<i>documentation</i>	Long string containing documentation on how to use the program and what it is. No newline characters are necessary; this is taken care of by CLI (p. 1117) later.

39.161.3 Member Data Documentation

39.161.3.1 documentation

```
std::string documentation
```

Documentation for what the program does.

Definition at line 192 of file cli_option.hpp.

39.161.3.2 `programName`

```
std::string programName
```

The name of the program.

Definition at line 190 of file `cli_option.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/ cli_option.hpp`

39.162 `BindingInfo` Class Reference

The **`BindingInfo`** (p. 984) class is used by the Markdown documentation generator to store multiple `ProgramDoc` objects, indexed by both the binding name (i.e.

Static Public Member Functions

- static **`util::ProgramDoc & GetProgramDoc`** (const std::string &bindingName)
Return a `ProgramDoc` object for a given `bindingName`.
- static std::string & **`Language`** ()
Get or modify the current language (don't set it to something invalid!).
- static void **`RegisterProgramDoc`** (const std::string &bindingName, const **`util::ProgramDoc`** &programDoc)
Register a `ProgramDoc` object with the given `bindingName`.

39.162.1 Detailed Description

The **`BindingInfo`** (p. 984) class is used by the Markdown documentation generator to store multiple `ProgramDoc` objects, indexed by both the binding name (i.e.

"knn") and the language (i.e. "cli").

Definition at line 30 of file `binding_info.hpp`.

39.162.2 Member Function Documentation

39.162.2.1 GetProgramDoc()

```
static util::ProgramDoc& GetProgramDoc (
    const std::string & bindingName ) [static]
```

Return a ProgramDoc object for a given bindingName.

39.162.2.2 Language()

```
static std::string& Language ( ) [static]
```

Get or modify the current language (don't set it to something invalid!).

Referenced by `mlpack::bindings::markdown::DefaultParam()`, `mlpack::bindings::markdown::GetPrintableType()`, and `mlpack::bindings::markdown::PrintTypeDoc()`.

39.162.2.3 RegisterProgramDoc()

```
static void RegisterProgramDoc (
    const std::string & bindingName,
    const util::ProgramDoc & programDoc ) [static]
```

Register a ProgramDoc object with the given bindingName.

Referenced by `ProgramDocWrapper::ProgramDocWrapper()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/ binding_info.hpp`

39.163 MDOption< T > Class Template Reference

The Markdown option class.

Public Member Functions

- **MDOption** (const T defaultValue, const std::string &identifier, const std::string &description, const std::string &alias, const std::string &cppName, const bool required=false, const bool input=true, const bool noTranspose=false, const std::string &bindingName="")

Construct an **MDOption** (p. 985) object.

39.163.1 Detailed Description

```
template<typename T>
class mlpack::bindings::markdown::MDOption< T >
```

The Markdown option class.

Definition at line 33 of file md_option.hpp.

39.163.2 Constructor & Destructor Documentation

39.163.2.1 MDOption()

```
MDOption (
    const T defaultValue,
    const std::string & identifier,
    const std::string & description,
    const std::string & alias,
    const std::string & cppName,
    const bool required = false,
    const bool input = true,
    const bool noTranspose = false,
    const std::string & bindingName = "" ) [inline]
```

Construct an **MDOption** (p. 985) object.

When constructed, it will register itself with **CLI** (p. 1117). The testName parameter is not used and added for compatibility reasons.

Definition at line 41 of file md_option.hpp.

References CLI::Add(), ParamData::alias, CLI::ClearSettings(), ParamData::cppType, ParamData::desc, CLI::function↵Map, CLI::GetSingleton(), ParamData::input, ParamData::loaded, ParamData::name, ParamData::noTranspose, ParamData::persistent, ParamData::required, CLI::RestoreSettings(), CLI::StoreSettings(), ParamData::tname, T↵YPENAME, ParamData::value, and ParamData::wasPassed.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/ **md_option.hpp**

39.164 ProgramDocWrapper Class Reference

Public Member Functions

- **ProgramDocWrapper** (const std::string &bindingName, const std::string &programName, const std::string &shortDocumentation, const std::function< std::string()> &documentation, const std::vector< std::pair< std::↵string, std::string >> &seeAlso)

Construct a ProgramDoc object and register it with **BindingInfo::RegisterProgramDoc()** (p. 985).

39.164.1 Detailed Description

Definition at line 22 of file program_doc_wrapper.hpp.

39.164.2 Constructor & Destructor Documentation

39.164.2.1 ProgramDocWrapper()

```
ProgramDocWrapper (
    const std::string & bindingName,
    const std::string & programName,
    const std::string & shortDocumentation,
    const std::function< std::string()> & documentation,
    const std::vector< std::pair< std::string, std::string >> & seeAlso ) [inline]
```

Construct a ProgramDoc object and register it with **BindingInfo::RegisterProgramDoc()** (p. 985).

Definition at line 29 of file program_doc_wrapper.hpp.

References BindingInfo::RegisterProgramDoc().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/ **program_doc_wrapper.hpp**

39.165 PyOption< T > Class Template Reference

The Python option class.

Public Member Functions

- **PyOption** (const T defaultValue, const std::string &identifier, const std::string &description, const std::string &alias, const std::string &cppName, const bool required=false, const bool input=true, const bool noTranspose=false, const std::string &="")

Construct a **PyOption** (p. 987) object.

39.165.1 Detailed Description

```
template<typename T>
class mlpack::bindings::python::PyOption< T >
```

The Python option class.

Definition at line 37 of file py_option.hpp.

39.165.2 Constructor & Destructor Documentation

39.165.2.1 PyOption()

```
PyOption (
    const T defaultValue,
    const std::string & identifier,
    const std::string & description,
    const std::string & alias,
    const std::string & cppName,
    const bool required = false,
    const bool input = true,
    const bool noTranspose = false,
    const std::string & testName = "" ) [inline]
```

Construct a **PyOption** (p. 987) object.

When constructed, it will register itself with **CLI** (p. 1117). The `testName` parameter is not used and added for compatibility reasons.

Definition at line 45 of file `py_option.hpp`.

References `CLI::Add()`, `ParamData::alias`, `CLI::ClearSettings()`, `ParamData::cppType`, `ParamData::desc`, `CLI::function↵Map`, `CLI::GetSingleton()`, `ParamData::input`, `ParamData::loaded`, `ParamData::name`, `ParamData::noTranspose`, `ParamData::persistent`, `ParamData::required`, `CLI::RestoreSettings()`, `CLI::StoreSettings()`, `ParamData::tname`, `T↵YPENAME`, `ParamData::value`, and `ParamData::wasPassed`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/ py_option.hpp`

39.166 ProgramDoc Class Reference

A static object whose constructor registers program documentation with the **CLI** (p. 1117) class.

Public Member Functions

- **ProgramDoc** (const std::string & **programName**, const std::string & **documentation**)
*Construct a **ProgramDoc** (p. 988) object.*

Public Attributes

- std::string **documentation**
Documentation for what the program does.
- std::string **programName**
The name of the program.

39.166.1 Detailed Description

A static object whose constructor registers program documentation with the **CLI** (p. 1117) class.

This should not be used outside of **CLI** (p. 1117) itself, and you should use the **PROGRAM_INFO()** (p. 2733) macro to declare these objects. Only one **ProgramDoc** (p. 988) object should ever exist.

See also

core/util/cli.hpp (p. 2694), **mlpack::CLI** (p. 1117)

Definition at line 119 of file test_option.hpp.

39.166.2 Constructor & Destructor Documentation

39.166.2.1 ProgramDoc()

```
ProgramDoc (  
    const std::string & programName,  
    const std::string & documentation )
```

Construct a **ProgramDoc** (p. 988) object.

When constructed, it will register itself with **CLI** (p. 1117).

Parameters

<i>programName</i>	Short string representing the name of the program.
<i>documentation</i>	Long string containing documentation on how to use the program and what it is. No newline characters are necessary; this is taken care of by CLI (p. 1117) later.

39.166.3 Member Data Documentation

39.166.3.1 documentation

```
std::string documentation
```

Documentation for what the program does.

Definition at line 137 of file test_option.hpp.

39.166.3.2 `programName`

```
std::string programName
```

The name of the program.

Definition at line 135 of file `test_option.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/ test_option.hpp`

39.167 `TestOption< N >` Class Template Reference

A static object whose constructor registers a parameter with the **CLI** (p. 1117) class.

Public Member Functions

- **TestOption** (const N defaultValue, const std::string &identifier, const std::string &description, const std::string &alias, const std::string &cppName, const bool required=false, const bool input=true, const bool noTranspose=false, const std::string &testName="")

Construct an Option object.

39.167.1 Detailed Description

```
template<typename N>
class mlpack::bindings::tests::TestOption< N >
```

A static object whose constructor registers a parameter with the **CLI** (p. 1117) class.

This should not be used outside of **CLI** (p. 1117) itself, and you should use the **PARAM_FLAG()** (p. 2710), **PARAM_DOUBLE()**, **PARAM_INT()**, **PARAM_STRING()**, or other similar macros to declare these objects instead of declaring them directly.

See also

core/util/cli.hpp (p. 2694), **mlpack::CLI** (p. 1117)

Definition at line 40 of file `test_option.hpp`.

39.167.2 Constructor & Destructor Documentation

39.167.2.1 TestOption()

```

TestOption (
    const N defaultValue,
    const std::string & identifier,
    const std::string & description,
    const std::string & alias,
    const std::string & cppName,
    const bool required = false,
    const bool input = true,
    const bool noTranspose = false,
    const std::string & testName = "" ) [inline]

```

Construct an Option object.

When constructed, it will register itself with **CLI** (p. 1117).

Parameters

<i>defaultValue</i>	Default value this parameter will be initialized to (for flags, this should be false, for instance).
<i>identifier</i>	The name of the option (no dashes in front; for <code>-help</code> , we would pass "help").
<i>description</i>	A short string describing the option.
<i>alias</i>	Short name of the parameter. "" for no alias.
<i>cppName</i>	Name of the C++ type of this parameter (i.e. "int").
<i>required</i>	Whether or not the option is required at runtime.
<i>input</i>	Whether or not the option is an input option.
<i>noTranspose</i>	If the parameter is a matrix and this is true, then the matrix will not be transposed on loading.
<i>testName</i>	Name of the test (used for identifying which binding test this option belongs to)

Definition at line 61 of file `test_option.hpp`.

References `CLI::Add()`, `ParamData::alias`, `CLI::ClearSettings()`, `ParamData::cppType`, `ParamData::desc`, `CLI::function↔Map`, `CLI::GetSingleton()`, `ParamData::input`, `ParamData::loaded`, `ParamData::name`, `ParamData::noTranspose`, `ParamData::persistent`, `ParamData::required`, `CLI::RestoreSettings()`, `CLI::SetPassed()`, `CLI::StoreSettings()`, `Param↔Data::tname`, `YPENAME`, `ParamData::value`, and `ParamData::wasPassed`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/ test_option.hpp`

39.168 BallBound< MetricType, VecType > Class Template Reference

Ball bound encloses a set of points at a specific distance (radius) from a specific point (center).

Public Types

- `typedef VecType::elem_type ElemType`
The underlying data type.
- `typedef VecType Vec`
A public version of the vector type.

Public Member Functions

- **BallBound** ()
Empty Constructor.
- **BallBound** (const size_t dimension)
Create the ball bound with the specified dimensionality.
- **BallBound** (const **ElemType** radius, const VecType ¢er)
Create the ball bound with the specified radius and center.
- **BallBound** (const **BallBound** &other)
Copy constructor. To prevent memory leaks.
- **BallBound** (**BallBound** &&other)
Move constructor: take possession of another bound.
- **~BallBound** ()
Destructor to release allocated memory.
- const VecType & **Center** () const
Get the center point of the ball.
- VecType & **Center** ()
Modify the center point of the ball.
- void **Center** (VecType ¢er) const
*Place the center of **BallBound** (p. 991) into the given vector.*
- bool **Contains** (const VecType &point) const
Determines if a point is within this bound.
- **ElemType Diameter** () const
Returns the diameter of the ballbound.
- size_t **Dim** () const
Get the dimensionality of the ball.
- template<typename OtherVecType >
ElemType MaxDistance (const OtherVecType &point, typename **std::enable_if_t**< **IsVector**< OtherVecType >::value > * = 0) const
Computes maximum distance.
- **ElemType MaxDistance** (const **BallBound** &other) const
Computes maximum distance.
- const MetricType & **Metric** () const
Returns the distance metric used in this bound.
- MetricType & **Metric** ()
Modify the distance metric used in this bound.
- template<typename OtherVecType >
ElemType MinDistance (const OtherVecType &point, typename **std::enable_if_t**< **IsVector**< OtherVecType >::value > * = 0) const
Calculates minimum bound-to-point squared distance.

- **ElemType MinDistance** (const **BallBound** &other) const
Calculates minimum bound-to-bound squared distance.
- **ElemType MinWidth** () const
Get the minimum width of the bound (this is same as the diameter).
- **BallBound & operator=** (const **BallBound** &other)
For the same reason as the copy constructor: to prevent memory leaks.
- **math::RangeType< ElemType > operator[]** (const size_t i) const
Get the range in a certain dimension.
- const **BallBound & operator|=** (const **BallBound** &other)
Expand the bound to include the given node.
- template<typename MatType >
const **BallBound & operator|=** (const MatType &data)
Expand the bound to include the given point.
- **ElemType Radius** () const
Get the radius of the ball.
- **ElemType & Radius** ()
Modify the radius of the ball.
- template<typename OtherVecType >
math::RangeType< ElemType > RangeDistance (const OtherVecType &other, typename **std::enable_if_t< IsVector< OtherVecType >::value > !=0**) const
Calculates minimum and maximum bound-to-point distance.
- **math::RangeType< ElemType > RangeDistance** (const **BallBound** &other) const
Calculates minimum and maximum bound-to-bound distance.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int version)
Serialize the bound.

39.168.1 Detailed Description

```
template<typename MetricType = metric::LMetric<2, true>, typename VecType = arma::vec>
class mlpack::bound::BallBound< MetricType, VecType >
```

Ball bound encloses a set of points at a specific distance (radius) from a specific point (center).

MetricType is the custom metric type that defaults to the Euclidean (L2) distance.

Template Parameters

<i>MetricType</i>	metric type used in the distance measure.
<i>VecType</i>	Type of vector (arma::vec or arma::sp_vec or similar).

Definition at line 32 of file ballbound.hpp.

39.168.2 Member Typedef Documentation

39.168.2.1 ElemType

```
typedef VecType::elem_type ElemType
```

The underlying data type.

Definition at line 36 of file ballbound.hpp.

39.168.2.2 Vec

```
typedef VecType Vec
```

A public version of the vector type.

Definition at line 38 of file ballbound.hpp.

39.168.3 Constructor & Destructor Documentation

39.168.3.1 BallBound() [1/5]

```
BallBound ( )
```

Empty Constructor.

39.168.3.2 BallBound() [2/5]

```
BallBound (  
    const size_t dimension )
```

Create the ball bound with the specified dimensionality.

Parameters

<i>dimension</i>	Dimensionality of ball bound.
------------------	-------------------------------

39.168.3.3 BallBound() [3/5]

```
BallBound (
    const ElemType radius,
    const VecType & center )
```

Create the ball bound with the specified radius and center.

Parameters

<i>radius</i>	Radius of ball bound.
<i>center</i>	Center of ball bound.

39.168.3.4 BallBound() [4/5]

```
BallBound (
    const BallBound< MetricType, VecType > & other )
```

Copy constructor. To prevent memory leaks.

39.168.3.5 BallBound() [5/5]

```
BallBound (
    BallBound< MetricType, VecType > && other )
```

Move constructor: take possession of another bound.

39.168.3.6 ~BallBound()

```
~ BallBound ( )
```

Destructor to release allocated memory.

39.168.4 Member Function Documentation

39.168.4.1 Center() [1/3]

```
const VecType& Center ( ) const [inline]
```

Get the center point of the ball.

Definition at line 93 of file ballbound.hpp.

Referenced by ProjVector::Project().

39.168.4.2 Center() [2/3]

```
VecType& Center ( ) [inline]
```

Modify the center point of the ball.

Definition at line 95 of file ballbound.hpp.

39.168.4.3 Center() [3/3]

```
void Center (
    VecType & center ) const [inline]
```

Place the center of **BallBound** (p. 991) into the given vector.

Parameters

<i>center</i>	Vector which the centroid will be written to.
---------------	---

Definition at line 121 of file ballbound.hpp.

References BallBound< MetricType, VecType >::MaxDistance(), BallBound< MetricType, VecType >::MinDistance(), BallBound< MetricType, VecType >::operator|=(), and BallBound< MetricType, VecType >::RangeDistance().

39.168.4.4 Contains()

```
bool Contains (
    const VecType & point ) const
```

Determines if a point is within this bound.

Parameters

<i>point</i>	Point to check the condition.
--------------	-------------------------------

Referenced by BallBound< MetricType, VecType >::MinWidth().

39.168.4.5 Diameter()

```
ElemType Diameter ( ) const [inline]
```

Returns the diameter of the ballbound.

Definition at line 197 of file ballbound.hpp.

39.168.4.6 Dim()

```
size_t Dim ( ) const [inline]
```

Get the dimensionality of the ball.

Definition at line 98 of file ballbound.hpp.

39.168.4.7 MaxDistance() [1/2]

```
ElemType MaxDistance (
    const OtherVecType & point,
    typename std::enable_if_t< IsVector< OtherVecType >::value > * = 0 ) const
```

Computes maximum distance.

Parameters

<i>point</i>	Point to which the maximum distance is requested.
--------------	---

Referenced by BallBound< MetricType, VecType >::Center().

39.168.4.8 MaxDistance() [2/2]

```
ElemType MaxDistance (
    const BallBound< MetricType, VecType > & other ) const
```

Computes maximum distance.

Parameters

<i>other</i>	Bound to which the maximum distance is requested.
--------------	---

39.168.4.9 Metric() [1/2]

```
const MetricType& Metric ( ) const [inline]
```

Returns the distance metric used in this bound.

Definition at line 200 of file ballbound.hpp.

39.168.4.10 Metric() [2/2]

```
MetricType& Metric ( ) [inline]
```

Modify the distance metric used in this bound.

Definition at line 202 of file ballbound.hpp.

References `BallBound< MetricType, VecType >::serialize()`.

39.168.4.11 MinDistance() [1/2]

```
ElemType MinDistance (
    const OtherVecType & point,
    typename std::enable_if_t< IsVector< OtherVecType >::value > * = 0 ) const
```

Calculates minimum bound-to-point squared distance.

Parameters

<i>point</i>	Point to which the minimum distance is requested.
--------------	---

Referenced by BallBound< MetricType, VecType >::Center().

39.168.4.12 MinDistance() [2/2]

```
ElemType MinDistance (
    const BallBound< MetricType, VecType > & other ) const
```

Calculates minimum bound-to-bound squared distance.

Parameters

<i>other</i>	Bound to which the minimum distance is requested.
--------------	---

39.168.4.13 MinWidth()

```
ElemType MinWidth ( ) const [inline]
```

Get the minimum width of the bound (this is same as the diameter).

For ball bounds, width along all dimensions remain same.

Definition at line 104 of file ballbound.hpp.

References BallBound< MetricType, VecType >::Contains(), and BallBound< MetricType, VecType >::operator[]().

39.168.4.14 operator=()

```
BallBound& operator= (
    const BallBound< MetricType, VecType > & other )
```

For the same reason as the copy constructor: to prevent memory leaks.

39.168.4.15 operator[]()

```
math::RangeType< ElemType> operator[] (
    const size_t i ) const
```

Get the range in a certain dimension.

Referenced by BallBound< MetricType, VecType >::MinWidth().

39.168.4.16 `operator" |=()` [1/2]

```
const BallBound& operator|= (
    const BallBound< MetricType, VecType > & other )
```

Expand the bound to include the given node.

Referenced by `BallBound< MetricType, VecType >::Center()`.

39.168.4.17 `operator" |=()` [2/2]

```
const BallBound& operator|= (
    const MatType & data )
```

Expand the bound to include the given point.

The centroid is recalculated to be the center of all of the given points.

Template Parameters

<i>MatType</i>	Type of matrix; could be <code>arma::mat</code> , <code>arma::spmat</code> , or a vector.
<i>data</i>	Data points to add.

39.168.4.18 `Radius()` [1/2]

```
ElemType Radius ( ) const [inline]
```

Get the radius of the ball.

Definition at line 88 of file `ballbound.hpp`.

Referenced by `ProjVector::Project()`.

39.168.4.19 `Radius()` [2/2]

```
ElemType& Radius ( ) [inline]
```

Modify the radius of the ball.

Definition at line 90 of file `ballbound.hpp`.

39.168.4.20 RangeDistance() [1/2]

```
math::RangeType< ElemType> RangeDistance (
    const OtherVecType & other,
    typename std::enable_if_t< IsVector< OtherVecType >::value > * = 0 ) const
```

Calculates minimum and maximum bound-to-point distance.

Parameters

<i>point</i>	Point to which the minimum and maximum distances are requested.
--------------	---

Referenced by BallBound< MetricType, VecType >::Center().

39.168.4.21 RangeDistance() [2/2]

```
math::RangeType< ElemType> RangeDistance (
    const BallBound< MetricType, VecType > & other ) const
```

Calculates minimum and maximum bound-to-bound distance.

Example: bound1.MinDistanceSq(other) for minimum distance.

Parameters

<i>other</i>	Bound to which the minimum and maximum distances are requested.
--------------	---

39.168.4.22 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int version )
```

Serialize the bound.

Referenced by BallBound< MetricType, VecType >::Metric().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ **ballbound.hpp**

39.169 BoundTraits< BoundType > Struct Template Reference

A class to obtain compile-time traits about BoundType classes.

Static Public Attributes

- static const bool **HasTightBounds** = false
If true, then the bounds for each dimension are tight.

39.169.1 Detailed Description

```
template<typename BoundType>
struct mlpack::bound::BoundTraits< BoundType >
```

A class to obtain compile-time traits about BoundType classes.

If you are writing your own BoundType class, you should make a template specialization in order to set the values correctly.

See also

TreeTraits, KernelTraits

Definition at line 26 of file bound_traits.hpp.

39.169.2 Member Data Documentation

39.169.2.1 HasTightBounds

```
const bool HasTightBounds = false [static]
```

If true, then the bounds for each dimension are tight.

If false, then the bounds for each dimension may be looser than the range of all points held in the bound. This defaults to false.

Definition at line 31 of file bound_traits.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ **bound_traits.hpp**

39.170 BoundTraits< BallBound< MetricType, VecType > > Struct Template Reference

A specialization of **BoundTraits** (p. 1002) for this bound type.

Static Public Attributes

- static const bool **HasTightBounds** = false
These bounds are potentially loose in some dimensions.

39.170.1 Detailed Description

```
template<typename MetricType, typename VecType>
struct mpack::bound::BoundTraits< BallBound< MetricType, VecType > >
```

A specialization of **BoundTraits** (p. 1002) for this bound type.

Definition at line 211 of file ballbound.hpp.

39.170.2 Member Data Documentation

39.170.2.1 HasTightBounds

```
const bool HasTightBounds = false [static]
```

These bounds are potentially loose in some dimensions.

Definition at line 214 of file ballbound.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ **ballbound.hpp**

39.171 BoundTraits< CellBound< MetricType, ElemType > > Struct Template Reference

Static Public Attributes

- static const bool **HasTightBounds** = true
These bounds are always tight for each dimension.

39.171.1 Detailed Description

```
template<typename MetricType, typename ElemType>
struct mlpack::bound::BoundTraits< CellBound< MetricType, ElemType > >
```

Definition at line 317 of file cellbound.hpp.

39.171.2 Member Data Documentation

39.171.2.1 HasTightBounds

```
const bool HasTightBounds = true [static]
```

These bounds are always tight for each dimension.

Definition at line 320 of file cellbound.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ **cellbound.hpp**

39.172 BoundTraits< HollowBallBound< MetricType, ElemType > > Struct Template Reference

A specialization of **BoundTraits** (p. 1002) for this bound type.

Static Public Attributes

- static const bool **HasTightBounds** = false
These bounds are potentially loose in some dimensions.

39.172.1 Detailed Description

```
template<typename MetricType, typename ElemType>
struct mlpack::bound::BoundTraits< HollowBallBound< MetricType, ElemType > >
```

A specialization of **BoundTraits** (p. 1002) for this bound type.

Definition at line 240 of file hollow_ball_bound.hpp.

39.172.2 Member Data Documentation

39.172.2.1 HasTightBounds

```
const bool HasTightBounds = false [static]
```

These bounds are potentially loose in some dimensions.

Definition at line 243 of file hollow_ball_bound.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ **hollow_ball_bound.hpp**

39.173 BoundTraits< HRectBound< MetricType, ElemType > > Struct Template Reference

Static Public Attributes

- static const bool **HasTightBounds** = true
These bounds are always tight for each dimension.

39.173.1 Detailed Description

```
template<typename MetricType, typename ElemType>  
struct mlpack::bound::BoundTraits< HRectBound< MetricType, ElemType > >
```

Definition at line 247 of file hrectbound.hpp.

39.173.2 Member Data Documentation

39.173.2.1 HasTightBounds

```
const bool HasTightBounds = true [static]
```

These bounds are always tight for each dimension.

Definition at line 250 of file hrectbound.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ **hrectbound.hpp**

39.174 `CellBound< MetricType, ElemType >` Class Template Reference

The **CellBound** (p. 1006) class describes a bound that consists of a number of hyperrectangles.

39.174.1 Detailed Description

```
template<typename MetricType = metric::LMetric<2, true>, typename ElemType = double>
class mlpack::bound::CellBound< MetricType, ElemType >
```

The **CellBound** (p. 1006) class describes a bound that consists of a number of hyperrectangles.

These hyperrectangles do not overlap each other. The bound is limited by an outer hyperrectangle and two addresses, the lower address and the high address. Thus, the bound contains all points included between the lower and the high addresses. The class caches the minimum bounding rectangle, the lower and the high addresses and the hyperrectangles that are described by the addresses.

The notion of addresses is described in the following paper.

```
@inproceedings{bayer1997,
  author = {Bayer, Rudolf},
  title = {The Universal B-Tree for Multidimensional Indexing: General
    Concepts},
  booktitle = {Proceedings of the International Conference on Worldwide
    Computing and Its Applications},
  series = {WWCA '97},
  year = {1997},
  isbn = {3-540-63343-X},
  pages = {198--209},
  numpages = {12},
  publisher = {Springer-Verlag},
  address = {London, UK, UK},
}
```

Definition at line 75 of file `cellbound.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ cellbound.hpp`

39.175 `HollowBallBound< TMetricType, ElemType >` Class Template Reference

Hollow ball bound encloses a set of points at a specific distance (radius) from a specific point (center) except points at a specific distance from another point (the center of the hole).

Public Types

- typedef `TMetricType` **MetricType**
A public version of the metric type.

Public Member Functions

- **HollowBallBound** ()
Empty Constructor.
- **HollowBallBound** (const size_t dimension)
Create the ball bound with the specified dimensionality.
- template<typename VecType >
HollowBallBound (const ElemType innerRadius, const ElemType outerRadius, const VecType ¢er)
Create the ball bound with the specified radius and center.
- **HollowBallBound** (const **HollowBallBound** &other)
Copy constructor. To prevent memory leaks.
- **HollowBallBound** (**HollowBallBound** &&other)
Move constructor: take possession of another bound.
- **~HollowBallBound** ()
Destructor to release allocated memory.
- const arma::Col< ElemType > & **Center** () const
Get the center point of the ball.
- arma::Col< ElemType > & **Center** ()
Modify the center point of the ball.
- template<typename VecType >
void **Center** (VecType ¢er) const
*Place the center of **BallBound** (p. 991) into the given vector.*
- template<typename VecType >
bool **Contains** (const VecType &point) const
Determines if a point is within this bound.
- bool **Contains** (const **HollowBallBound** &other) const
Determines if another bound is within this bound.
- ElemType **Diameter** () const
Returns the diameter of the ballbound.
- size_t **Dim** () const
Get the dimensionality of the ball.
- const arma::Col< ElemType > & **HollowCenter** () const
Get the center point of the hollow.
- arma::Col< ElemType > & **HollowCenter** ()
Modify the center point of the hollow.
- ElemType **InnerRadius** () const
Get the innner radius of the ball.
- ElemType & **InnerRadius** ()
Modify the inner radius of the ball.
- template<typename VecType >
ElemType **MaxDistance** (const VecType &point, typename std::enable_if_t< IsVector< VecType >::value > !=0) const
Computes maximum distance.
- ElemType **MaxDistance** (const **HollowBallBound** &other) const
Computes maximum distance.
- const **MetricType** & **Metric** () const
Returns the distance metric used in this bound.
- **MetricType** & **Metric** ()

- Modify the distance metric used in this bound.*

 - `template<typename VecType >`
`ElemType MinDistance (const VecType &point, typename std::enable_if_t< IsVector< VecType >::value > * = 0) const`
Calculates minimum bound-to-point squared distance
 - `ElemType MinDistance (const HollowBallBound &other) const`
Calculates minimum bound-to-bound squared distance.
 - `ElemType MinWidth () const`
Get the minimum width of the bound (this is same as the diameter).
 - `HollowBallBound & operator= (const HollowBallBound &other)`
For the same reason as the copy constructor: to prevent memory leaks.
 - `math::RangeType< ElemType > operator[] (const size_t i) const`
Get the range in a certain dimension.
 - `template<typename MatType >`
`const HollowBallBound & operator|= (const MatType &data)`
Expand the bound to include the given point.
 - `const HollowBallBound & operator|= (const HollowBallBound &other)`
Expand the bound to include the given bound.
 - `ElemType OuterRadius () const`
Get the outer radius of the ball.
 - `ElemType & OuterRadius ()`
Modify the outer radius of the ball.
 - `template<typename VecType >`
`math::RangeType< ElemType > RangeDistance (const VecType &other, typename std::enable_if_t< IsVector< VecType >::value > * = 0) const`
Calculates minimum and maximum bound-to-point distance.
 - `math::RangeType< ElemType > RangeDistance (const HollowBallBound &other) const`
Calculates minimum and maximum bound-to-bound distance.
 - `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int version)`
Serialize the bound.

39.175.1 Detailed Description

```
template<typename TMetricType = metric::LMetric<2, true>, typename ElemType = double>
class mpack::bound::HollowBallBound< TMetricType, ElemType >
```

Hollow ball bound encloses a set of points at a specific distance (radius) from a specific point (center) except points at a specific distance from another point (the center of the hole).

MetricType is the custom metric type that defaults to the Euclidean (L2) distance.

Template Parameters

<i>TMetricType</i>	metric type used in the distance measure.
<i>ElemType</i>	Type of element (float or double or similar).

Definition at line 33 of file hollow_ball_bound.hpp.

39.175.2 Member Typedef Documentation

39.175.2.1 MetricType

```
typedef TMetricType MetricType
```

A public version of the metric type.

Definition at line 37 of file hollow_ball_bound.hpp.

39.175.3 Constructor & Destructor Documentation

39.175.3.1 HollowBallBound() [1/5]

```
HollowBallBound ( )
```

Empty Constructor.

39.175.3.2 HollowBallBound() [2/5]

```
HollowBallBound (  
    const size_t dimension )
```

Create the ball bound with the specified dimensionality.

Parameters

<i>dimension</i>	Dimensionality of ball bound.
------------------	-------------------------------

39.175.3.3 HollowBallBound() [3/5]

```
HollowBallBound (  

```

```
const ElemType innerRadius,
const ElemType outerRadius,
const VecType & center )
```

Create the ball bound with the specified radius and center.

Parameters

<i>innerRadius</i>	Inner radius of ball bound.
<i>outerRadius</i>	Outer radius of ball bound.
<i>center</i>	Center of ball bound.

39.175.3.4 HollowBallBound() [4/5]

```
HollowBallBound (
    const HollowBallBound< TMetricType, ElemType > & other )
```

Copy constructor. To prevent memory leaks.

39.175.3.5 HollowBallBound() [5/5]

```
HollowBallBound (
    HollowBallBound< TMetricType, ElemType > && other )
```

Move constructor: take possession of another bound.

39.175.3.6 ~HollowBallBound()

```
~ HollowBallBound ( )
```

Destructor to release allocated memory.

39.175.4 Member Function Documentation

39.175.4.1 Center() [1/3]

```
const arma::Col<ElemType>& Center ( ) const [inline]
```

Get the center point of the ball.

Definition at line 103 of file hollow_ball_bound.hpp.

39.175.4.2 Center() [2/3]

```
arma::Col<ElemType>& Center ( ) [inline]
```

Modify the center point of the ball.

Definition at line 105 of file hollow_ball_bound.hpp.

39.175.4.3 Center() [3/3]

```
void Center (
    VecType & center ) const [inline]
```

Place the center of **BallBound** (p. 991) into the given vector.

Parameters

<i>center</i>	Vector which the centroid will be written to.
---------------	---

Definition at line 145 of file hollow_ball_bound.hpp.

References `HollowBallBound< TMetricType, ElemType >::MaxDistance()`, `HollowBallBound< TMetricType, ElemType >::MinDistance()`, `HollowBallBound< TMetricType, ElemType >::operator|=(())`, and `HollowBallBound< TMetricType, ElemType >::RangeDistance()`.

39.175.4.4 Contains() [1/2]

```
bool Contains (
    const VecType & point ) const
```

Determines if a point is within this bound.

Parameters

<i>point</i>	Point to check the condition.
--------------	-------------------------------

Referenced by `HollowBallBound< TMetricType, ElemType >::MinWidth()`.

39.175.4.5 Contains() [2/2]

```
bool Contains (
    const HollowBallBound< TMetricType, ElemType > & other ) const
```

Determines if another bound is within this bound.

Parameters

<i>other</i>	Bound to check the condition.
--------------	-------------------------------

39.175.4.6 Diameter()

```
ElemType Diameter ( ) const [inline]
```

Returns the diameter of the ballbound.

Definition at line 226 of file `hollow_ball_bound.hpp`.

References `RangeType< T >::Hi()`.

39.175.4.7 Dim()

```
size_t Dim ( ) const [inline]
```

Get the dimensionality of the ball.

Definition at line 113 of file `hollow_ball_bound.hpp`.

39.175.4.8 HollowCenter() [1/2]

```
const arma::Col<ElemType>& HollowCenter ( ) const [inline]
```

Get the center point of the hollow.

Definition at line 108 of file hollow_ball_bound.hpp.

39.175.4.9 HollowCenter() [2/2]

```
arma::Col<ElemType>& HollowCenter ( ) [inline]
```

Modify the center point of the hollow.

Definition at line 110 of file hollow_ball_bound.hpp.

39.175.4.10 InnerRadius() [1/2]

```
ElemType InnerRadius ( ) const [inline]
```

Get the innner radius of the ball.

Definition at line 98 of file hollow_ball_bound.hpp.

References `RangeType< T >::Lo()`.

39.175.4.11 InnerRadius() [2/2]

```
ElemType& InnerRadius ( ) [inline]
```

Modify the inner radius of the ball.

Definition at line 100 of file hollow_ball_bound.hpp.

References `RangeType< T >::Lo()`.

39.175.4.12 MaxDistance() [1/2]

```
ElemType MaxDistance (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const
```

Computes maximum distance.

Parameters

<i>point</i>	Point to which the maximum distance is requested.
--------------	---

Referenced by `HollowBallBound< TMetricType, ElemType >::Center()`.

39.175.4.13 MaxDistance() [2/2]

```
ElemType MaxDistance (
    const HollowBallBound< TMetricType, ElemType > & other ) const
```

Computes maximum distance.

Parameters

<i>other</i>	Bound to which the maximum distance is requested.
--------------	---

39.175.4.14 Metric() [1/2]

```
const MetricType& Metric ( ) const [inline]
```

Returns the distance metric used in this bound.

Definition at line 229 of file `hollow_ball_bound.hpp`.

39.175.4.15 Metric() [2/2]

```
MetricType& Metric ( ) [inline]
```

Modify the distance metric used in this bound.

Definition at line 231 of file `hollow_ball_bound.hpp`.

References `HollowBallBound< TMetricType, ElemType >::serialize()`.

39.175.4.16 MinDistance() [1/2]

```
ElemType MinDistance (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const
```

Calculates minimum bound-to-point squared distance

Parameters

<i>point</i>	Point to which the minimum distance is requested.
--------------	---

Referenced by `HollowBallBound< TMetricType, ElemType >::Center()`.

39.175.4.17 MinDistance() [2/2]

```
ElemType MinDistance (
    const HollowBallBound< TMetricType, ElemType > & other ) const
```

Calculates minimum bound-to-bound squared distance.

Parameters

<i>other</i>	Bound to which the minimum distance is requested.
--------------	---

39.175.4.18 MinWidth()

```
ElemType MinWidth ( ) const [inline]
```

Get the minimum width of the bound (this is same as the diameter).

For ball bounds, width along all dimensions remain same.

Definition at line 119 of file `hollow_ball_bound.hpp`.

References `HollowBallBound< TMetricType, ElemType >::Contains()`, `RangeType< T >::Hi()`, and `HollowBallBound< TMetricType, ElemType >::operator[]()`.

39.175.4.19 operator=()

```
HollowBallBound& operator= (
    const HollowBallBound< TMetricType, ElemType > & other )
```

For the same reason as the copy constructor: to prevent memory leaks.

39.175.4.20 `operator[]()`

```
math::RangeType<ElemType> operator[] (
    const size_t i ) const
```

Get the range in a certain dimension.

Referenced by `HollowBallBound< TMetricType, ElemType >::MinWidth()`.

39.175.4.21 `operator" |=()` [1/2]

```
const HollowBallBound& operator|= (
    const MatType & data )
```

Expand the bound to include the given point.

The centroid will not be moved.

Template Parameters

<i>MatType</i>	Type of matrix; could be <code>arma::mat</code> , <code>arma::spmat</code> , or a vector.
<i>data</i>	Data points to add.

Referenced by `HollowBallBound< TMetricType, ElemType >::Center()`.

39.175.4.22 `operator" |=()` [2/2]

```
const HollowBallBound& operator|= (
    const HollowBallBound< TMetricType, ElemType > & other )
```

Expand the bound to include the given bound.

The centroid will not be moved.

Template Parameters

<i>MatType</i>	Type of matrix; could be <code>arma::mat</code> , <code>arma::spmat</code> , or a vector.
<i>data</i>	Data points to add.

39.175.4.23 OuterRadius() [1/2]

```
ElemType OuterRadius ( ) const [inline]
```

Get the outer radius of the ball.

Definition at line 93 of file hollow_ball_bound.hpp.

References `RangeType< T >::Hi()`.

39.175.4.24 OuterRadius() [2/2]

```
ElemType& OuterRadius ( ) [inline]
```

Modify the outer radius of the ball.

Definition at line 95 of file hollow_ball_bound.hpp.

References `RangeType< T >::Hi()`.

39.175.4.25 RangeDistance() [1/2]

```
math::RangeType<ElemType> RangeDistance (
    const VecType & other,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const
```

Calculates minimum and maximum bound-to-point distance.

Parameters

<i>point</i>	Point to which the minimum and maximum distances are requested.
--------------	---

Referenced by `HollowBallBound< TMetricType, ElemType >::Center()`.

39.175.4.26 RangeDistance() [2/2]

```
math::RangeType<ElemType> RangeDistance (
    const HollowBallBound< TMetricType, ElemType > & other ) const
```

Calculates minimum and maximum bound-to-bound distance.

Example: `bound1.MinDistanceSq(other)` for minimum distance.

Parameters

<i>other</i>	Bound to which the minimum and maximum distances are requested.
--------------	---

39.175.4.27 `serialize()`

```
void serialize (
    Archive & ar,
    const unsigned int version )
```

Serialize the bound.

Referenced by `HollowBallBound< TMetricType, ElemType >::Metric()`.

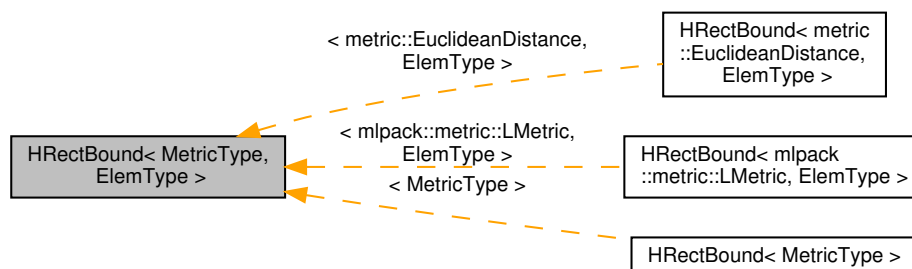
The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ hollow_ball_bound.hpp`

39.176 `HRectBound< MetricType, ElemType >` Class Template Reference

Hyper-rectangle bound for an L-metric.

Inheritance diagram for `HRectBound< MetricType, ElemType >`:



Public Member Functions

- **HRectBound** ()
Empty constructor; creates a bound of dimensionality 0.
- **HRectBound** (const size_t dimension)
Initializes to specified dimensionality with each dimension the empty set.
- **HRectBound** (const **HRectBound** &other)
Copy constructor; necessary to prevent memory leaks.
- **HRectBound** (**HRectBound** &&other)
Move constructor: take possession of another bound's information.
- **~HRectBound** ()
Destructor: clean up memory.
- void **Center** (arma::Col< ElemType > ¢er) const
Calculates the center of the range, placing it into the given vector.
- void **Clear** ()
Resets all dimensions to the empty set (so that this bound contains nothing).
- template<typename VecType >
bool **Contains** (const VecType &point) const
Determines if a point is within this bound.
- bool **Contains** (const **HRectBound** &bound) const
Determines if this bound partially contains a bound.
- ElemType **Diameter** () const
Returns the diameter of the hyperrectangle (that is, the longest diagonal).
- size_t **Dim** () const
Gets the dimensionality.
- template<typename VecType >
ElemType **MaxDistance** (const VecType &point, typename std::enable_if_t< IsVector< VecType >::value >*=0) const
Calculates maximum bound-to-point squared distance.
- ElemType **MaxDistance** (const **HRectBound** &other) const
Computes maximum distance.
- const MetricType & **Metric** () const
Get the instantiated metric associated with the bound.
- MetricType & **Metric** ()
Modify the instantiated metric associated with the bound.
- template<typename VecType >
ElemType **MinDistance** (const VecType &point, typename std::enable_if_t< IsVector< VecType >::value >*=0) const
Calculates minimum bound-to-point distance.
- ElemType **MinDistance** (const **HRectBound** &other) const
Calculates minimum bound-to-bound distance.
- ElemType **MinWidth** () const
Get the minimum width of the bound.
- ElemType & **MinWidth** ()
Modify the minimum width of the bound.
- **HRectBound** **operator &** (const **HRectBound** &bound) const
Returns the intersection of this bound and another.
- **HRectBound** & **operator &=** (const **HRectBound** &bound)

- Intersects this bound with another.*
- **HRectBound & operator=** (const **HRectBound** &other)
Same as copy constructor; necessary to prevent memory leaks.
- **math::RangeType**< ElemType > & **operator[]** (const size_t i)
Get the range for a particular dimension.
- const **math::RangeType**< ElemType > & **operator[]** (const size_t i) const
Modify the range for a particular dimension. No bounds checking.
- template<typename MatType >
HRectBound & operator|= (const MatType &data)
Expands this region to include new points.
- **HRectBound & operator|=** (const **HRectBound** &other)
Expands this region to encompass another bound.
- ElemType **Overlap** (const **HRectBound** &bound) const
Returns the volume of overlap of this bound and another.
- **math::RangeType**< ElemType > **RangeDistance** (const **HRectBound** &other) const
Calculates minimum and maximum bound-to-bound distance.
- template<typename VecType >
math::RangeType< ElemType > **RangeDistance** (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > !=0) const
Calculates minimum and maximum bound-to-point distance.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int version)
Serialize the bound object.
- ElemType **Volume** () const
Calculate the volume of the hyperrectangle.

39.176.1 Detailed Description

```
template<typename MetricType = metric::LMetric<2, true>, typename ElemType = double>
class mlpack::bound::HRectBound< MetricType, ElemType >
```

Hyper-rectangle bound for an L-metric.

This should be used in conjunction with the LMetric class. Be sure to use the same template parameters for LMetric as you do for **HRectBound** (p. 1018) – otherwise odd results may occur.

Template Parameters

<i>MetricType</i>	Type of metric to use; must be of type LMetric.
<i>ElemType</i>	Element type (double/float/int/etc.).

Definition at line 54 of file hrectbound.hpp.

39.176.2 Constructor & Destructor Documentation

39.176.2.1 HRectBound() [1/4]

```
HRectBound ( )
```

Empty constructor; creates a bound of dimensionality 0.

39.176.2.2 HRectBound() [2/4]

```
HRectBound (
    const size_t dimension )
```

Initializes to specified dimensionality with each dimension the empty set.

Parameters

<i>dimension</i>	Dimensionality of bound.
------------------	--------------------------

39.176.2.3 HRectBound() [3/4]

```
HRectBound (
    const HRectBound< MetricType, ElemType > & other )
```

Copy constructor; necessary to prevent memory leaks.

39.176.2.4 HRectBound() [4/4]

```
HRectBound (
    HRectBound< MetricType, ElemType > && other )
```

Move constructor: take possession of another bound's information.

39.176.2.5 ~HRectBound()

```
~ HRectBound ( )
```

Destructor: clean up memory.

39.176.3 Member Function Documentation

39.176.3.1 Center()

```
void Center (
    arma::Col< ElemType > & center ) const
```

Calculates the center of the range, placing it into the given vector.

Parameters

<i>center</i>	Vector which the center will be written to.
---------------	---

Referenced by `RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::Center()`, and `Octree< MetricType, StatisticType, MatType >::Center()`.

39.176.3.2 Clear()

```
void Clear ( )
```

Resets all dimensions to the empty set (so that this bound contains nothing).

39.176.3.3 Contains() [1/2]

```
bool Contains (
    const VecType & point ) const
```

Determines if a point is within this bound.

Parameters

<i>point</i>	Point to check the condition.
--------------	-------------------------------

39.176.3.4 Contains() [2/2]

```
bool Contains (
    const HRectBound< MetricType, ElemType > & bound ) const
```

Determines if this bound partially contains a bound.

Parameters

<i>other</i>	Bound to check the condition.
--------------	-------------------------------

39.176.3.5 Diameter()

```
ElemType Diameter ( ) const
```

Returns the diameter of the hyperrectangle (that is, the longest diagonal).

39.176.3.6 Dim()

```
size_t Dim ( ) const [inline]
```

Gets the dimensionality.

Definition at line 92 of file hrectbound.hpp.

39.176.3.7 MaxDistance() [1/2]

```
ElemType MaxDistance (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const
```

Calculates maximum bound-to-point squared distance.

Parameters

<i>point</i>	Point to which the maximum distance is requested.
--------------	---

Referenced by RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::MaxDistance().

39.176.3.8 MaxDistance() [2/2]

```
ElemType MaxDistance (
    const HRectBound< MetricType, ElemType > & other ) const
```

Computes maximum distance.

Parameters

<i>other</i>	Bound to which the maximum distance is requested.
--------------	---

39.176.3.9 Metric() [1/2]

```
const MetricType& Metric ( ) const [inline]
```

Get the instantiated metric associated with the bound.

Definition at line 107 of file hrectbound.hpp.

39.176.3.10 Metric() [2/2]

```
MetricType& Metric ( ) [inline]
```

Modify the instantiated metric associated with the bound.

Definition at line 109 of file hrectbound.hpp.

39.176.3.11 MinDistance() [1/2]

```
ElemType MinDistance (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const
```

Calculates minimum bound-to-point distance.

Parameters

<i>point</i>	Point to which the minimum distance is requested.
--------------	---

Referenced by RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::MinDistance().

39.176.3.12 MinDistance() [2/2]

```
ElemType MinDistance (
    const HRectBound< MetricType, ElemType > & other ) const
```

Calculates minimum bound-to-bound distance.

Parameters

<i>other</i>	Bound to which the minimum distance is requested.
--------------	---

39.176.3.13 MinWidth() [1/2]

```
ElemType MinWidth ( ) const [inline]
```

Get the minimum width of the bound.

Definition at line 102 of file hrectbound.hpp.

Referenced by RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::MinimumBoundDistance().

39.176.3.14 MinWidth() [2/2]

```
ElemType& MinWidth ( ) [inline]
```

Modify the minimum width of the bound.

Definition at line 104 of file hrectbound.hpp.

39.176.3.15 operator &()

```
HRectBound operator& (
    const HRectBound< MetricType, ElemType > & bound ) const
```

Returns the intersection of this bound and another.

39.176.3.16 operator &=()

```
HRectBound& operator&= (
    const HRectBound< MetricType, ElemType > & bound )
```

Intersects this bound with another.

39.176.3.17 operator=()

```
HRectBound& operator= (
    const HRectBound< MetricType, ElemType > & other )
```

Same as copy constructor; necessary to prevent memory leaks.

39.176.3.18 operator[]() [1/2]

```
math::RangeType<ElemType>& operator[] (
    const size_t i ) [inline]
```

Get the range for a particular dimension.

No bounds checking. Be careful: this may make **MinWidth()** (p. 1025) invalid.

Definition at line 96 of file hrectbound.hpp.

39.176.3.19 operator[]() [2/2]

```
const math::RangeType<ElemType>& operator[] (
    const size_t i ) const [inline]
```

Modify the range for a particular dimension. No bounds checking.

Definition at line 98 of file hrectbound.hpp.

39.176.3.20 operator" |=() [1/2]

```
HRectBound& operator|= (
    const MatType & data )
```

Expands this region to include new points.

Template Parameters

<i>MatType</i>	Type of matrix; could be Mat, SpMat, a subview, or just a vector.
----------------	---

Parameters

<i>data</i>	Data points to expand this region to include.
-------------	---

39.176.3.21 operator" |=() [2/2]

```
HRectBound& operator|= (
    const HRectBound< MetricType, ElemType > & other )
```

Expands this region to encompass another bound.

39.176.3.22 Overlap()

```
ElemType Overlap (
    const HRectBound< MetricType, ElemType > & bound ) const
```

Returns the volume of overlap of this bound and another.

39.176.3.23 RangeDistance() [1/2]

```
math::RangeType<ElemType> RangeDistance (
    const HRectBound< MetricType, ElemType > & other ) const
```

Calculates minimum and maximum bound-to-bound distance.

Parameters

<i>other</i>	Bound to which the minimum and maximum distances are requested.
--------------	---

Referenced by RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::RangeDistance().

39.176.3.24 RangeDistance() [2/2]

```

math::RangeType<ElemType> RangeDistance (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const

```

Calculates minimum and maximum bound-to-point distance.

Parameters

<i>point</i>	Point to which the minimum and maximum distances are requested.
--------------	---

39.176.3.25 serialize()

```

void serialize (
    Archive & ar,
    const unsigned int version )

```

Serialize the bound object.

39.176.3.26 Volume()

```
ElemType Volume ( ) const
```

Calculate the volume of the hyperrectangle.

Returns

Volume of the hyperrectangle.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ **hrectbound.hpp**

39.177 IsLMetric< MetricType > Struct Template Reference

Utility struct where Value is true if and only if the argument is of type LMetric.

Static Public Attributes

- static const bool **Value** = false

39.177.1 Detailed Description

```
template<typename MetricType>
struct mpack::bound::meta::IsLMetric< MetricType >
```

Utility struct where Value is true if and only if the argument is of type LMetric.

Definition at line 30 of file hrectbound.hpp.

39.177.2 Member Data Documentation

39.177.2.1 Value

```
const bool Value = false [static]
```

Definition at line 32 of file hrectbound.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ **hrectbound.hpp**

39.178 IsLMetric< metric::LMetric< Power, TakeRoot > > Struct Template Reference

Specialization for **IsLMetric** (p. 1028) when the argument is of type LMetric.

Static Public Attributes

- static const bool **Value** = true

39.178.1 Detailed Description

```
template<int Power, bool TakeRoot>
struct mpack::bound::meta::IsLMetric< metric::LMetric< Power, TakeRoot > >
```

Specialization for **IsLMetric** (p. 1028) when the argument is of type LMetric.

Definition at line 37 of file hrectbound.hpp.

39.178.2 Member Data Documentation

39.178.2.1 Value

```
const bool Value = true [static]
```

Definition at line 39 of file hrectbound.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ **hrectbound.hpp**

39.179 AverageInterpolation Class Reference

This class performs average interpolation to generate interpolation weights for neighborhood-based collaborative filtering.

Public Member Functions

- **AverageInterpolation** ()
- **AverageInterpolation** (const arma::sp_mat &)

This constructor is needed for interface consistency.
- template<typename VectorType , typename DecompositionPolicy >
 void **GetWeights** (VectorType &&weights, const DecompositionPolicy &, const size_t, const arma::Col< size_t > &neighbors, const arma::vec &, const arma::sp_mat &)

Interpolation weights are identical and sum up to one.

39.179.1 Detailed Description

This class performs average interpolation to generate interpolation weights for neighborhood-based collaborative filtering.

An example of how to use **AverageInterpolation** (p. 1030) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

CFType<> cf(data);

// Generate 10 recommendations for all users.
cf.template GetRecommendations<EuclideanSearch,
    AverageInterpolation>(10, recommendations);
```

Definition at line 39 of file average_interpolation.hpp.

39.179.2 Constructor & Destructor Documentation

39.179.2.1 AverageInterpolation() [1/2]

```
AverageInterpolation ( ) [inline]
```

Definition at line 43 of file `average_interpolation.hpp`.

39.179.2.2 AverageInterpolation() [2/2]

```
AverageInterpolation (
    const arma::sp_mat & ) [inline]
```

This constructor is needed for interface consistency.

Definition at line 48 of file `average_interpolation.hpp`.

39.179.3 Member Function Documentation

39.179.3.1 GetWeights()

```
void GetWeights (
    VectorType && weights,
    const DecompositionPolicy & ,
    const size_t ,
    const arma::Col< size_t > & neighbors,
    const arma::vec & ,
    const arma::sp_mat & ) [inline]
```

Interpolation weights are identical and sum up to one.

After getting the weights, CF algorithm multiplies each neighbor's rating by its corresponding weight and sums them to get predicted rating.

Parameters

<i>weights</i>	Resulting interpolation weights. The size of weights should be set to the number of neighbors before calling GetWeights() (p. 1031).
<i>decomposition</i>	Decomposition object.
<i>queryUser</i>	Queried user.
<i>neighbors</i>	Neighbors of queried user.
<i>similarities</i>	Similarities between query user and neighbors.
<i>cleanedData</i>	Sparse rating matrix.

Definition at line 65 of file average_interpolation.hpp.

References Log::Fatal.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/interpolation_policies/ **average_interpolation.**↩
hpp

39.180 BatchSVDPolicy Class Reference

Implementation of the Batch SVD policy to act as a wrapper when accessing Batch SVD from within **CFTYPE** (p. 1045).

Public Member Functions

- template<typename MatType >
void **Apply** (const MatType &, const arma::sp_mat &cleanedData, const size_t rank, const size_t maxIterations, const double minResidue, const bool mit)
Apply Collaborative Filtering to the provided data set using the batch SVD method.
- template<typename NeighborSearchPolicy >
void **GetNeighborhood** (const arma::Col< size_t > &users, const size_t numUsersForSimilarity, arma::Mat< size_t > &neighborhood, arma::mat &similarities) const
Get the neighborhood and corresponding similarities for a set of users.
- double **GetRating** (const size_t user, const size_t item) const
Return predicted rating given user ID and item ID.
- void **GetRatingOfUser** (const size_t user, arma::vec &rating) const
Get predicted ratings for a user.
- const arma::mat & **H** () const
Get the User Matrix.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialization.
- const arma::mat & **W** () const
Get the Item Matrix.

39.180.1 Detailed Description

Implementation of the Batch SVD policy to act as a wrapper when accessing Batch SVD from within **CFTYPE** (p. 1045).

An example of how to use **BatchSVDPolicy** (p. 1032) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

CFTYPE<BatchSVDPolicy> cf(data);

// Generate 10 recommendations for all users.
cf.GetRecommendations(10, recommendations);
```

Definition at line 43 of file batch_svd_method.hpp.

39.180.2 Member Function Documentation

39.180.2.1 Apply()

```
void Apply (
    const MatType & ,
    const arma::sp_mat & cleanedData,
    const size_t rank,
    const size_t maxIterations,
    const double minResidue,
    const bool mit ) [inline]
```

Apply Collaborative Filtering to the provided data set using the batch SVD method.

Parameters

<i>data</i>	Data matrix: dense matrix (coordinate lists) or sparse matrix(cleaned).
<i>cleanedData</i>	item user table in form of sparse matrix.
<i>rank</i>	Rank parameter for matrix factorization.
<i>maxIterations</i>	Maximum number of iterations.
<i>minResidue</i>	Residue required to terminate.
<i>mit</i>	Whether to terminate only when maxIterations is reached.

Definition at line 59 of file batch_svd_method.hpp.

References `AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::Apply()`.

39.180.2.2 GetNeighborhood()

```
void GetNeighborhood (
    const arma::Col< size_t > & users,
    const size_t numUsersForSimilarity,
    arma::Mat< size_t > & neighborhood,
    arma::mat & similarities ) const [inline]
```

Get the neighborhood and corresponding similarities for a set of users.

Template Parameters

<i>NeighborSearchPolicy</i>	The policy to perform neighbor search.
-----------------------------	--

Parameters

<i>users</i>	Users whose neighborhood is to be computed.
<i>numUsersForSimilarity</i>	The number of neighbors returned for each user.
<i>neighborhood</i>	Neighbors represented by user IDs.
<i>similarities</i>	Similarity between each user and each of its neighbors.

Definition at line 123 of file batch_svd_method.hpp.

39.180.2.3 GetRating()

```
double GetRating (
    const size_t user,
    const size_t item ) const [inline]
```

Return predicted rating given user ID and item ID.

Parameters

<i>user</i>	User ID.
<i>item</i>	Item ID.

Definition at line 93 of file batch_svd_method.hpp.

39.180.2.4 GetRatingOfUser()

```
void GetRatingOfUser (
    const size_t user,
    arma::vec & rating ) const [inline]
```

Get predicted ratings for a user.

Parameters

<i>user</i>	User ID.
<i>rating</i>	Resulting rating vector.

Definition at line 105 of file batch_svd_method.hpp.

39.180.2.5 H()

```
const arma::mat& H ( ) const [inline]
```

Get the User Matrix.

Definition at line 152 of file `batch_svd_method.hpp`.

39.180.2.6 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialization.

Definition at line 158 of file `batch_svd_method.hpp`.

39.180.2.7 W()

```
const arma::mat& W ( ) const [inline]
```

Get the Item Matrix.

Definition at line 150 of file `batch_svd_method.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/ batch_svd_method.hpp`

39.181 BiasSVDPolicy Class Reference

Implementation of the Bias SVD policy to act as a wrapper when accessing Bias SVD from within **CFTYPE** (p. 1045).

Public Member Functions

- **BiasSVDPolicy** (const size_t maxIterations=10, const double alpha=0.02, const double lambda=0.05)
Use Bias SVD method to perform collaborative filtering.
- double **Alpha** () const
Get learning rate.
- double & **Alpha** ()
Modify learning rate.
- void **Apply** (const arma::mat &data, const arma::sp_mat &, const size_t rank, const size_t maxIterations, const double, const bool)
Apply Collaborative Filtering to the provided data set using the bias SVD.
- template<typename NeighborSearchPolicy >
void **GetNeighborhood** (const arma::Col< size_t > &users, const size_t numUsersForSimilarity, arma::Mat< size_t > &neighborhood, arma::mat &similarities) const
Get the neighborhood and corresponding similarities for a set of users.
- double **GetRating** (const size_t user, const size_t item) const
Return predicted rating given user ID and item ID.
- void **GetRatingOfUser** (const size_t user, arma::vec &rating) const
Get predicted ratings for a user.
- const arma::mat & **H** () const
Get the User Matrix.
- double **Lambda** () const
Get regularization parameter.
- double & **Lambda** ()
Modify regularization parameter.
- size_t **MaxIterations** () const
Get the number of iterations.
- size_t & **MaxIterations** ()
Modify the number of iterations.
- const arma::vec & **P** () const
Get the Item Bias Vector.
- const arma::vec & **Q** () const
Get the User Bias Vector.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialization.
- const arma::mat & **W** () const
Get the Item Matrix.

39.181.1 Detailed Description

Implementation of the Bias SVD policy to act as a wrapper when accessing Bias SVD from within **CFTYPE** (p. 1045).

An example of how to use **BiasSVDPolicy** (p. 1035) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

CType<BiasSVDPolicy> cf(data);

// Generate 10 recommendations for all users.
cf.GetRecommendations(10, recommendations);
```

Definition at line 41 of file bias_svd_method.hpp.

39.181.2 Constructor & Destructor Documentation

39.181.2.1 BiasSVDPolicy()

```
BiasSVDPolicy (
    const size_t maxIterations = 10,
    const double alpha = 0.02,
    const double lambda = 0.05 ) [inline]
```

Use Bias SVD method to perform collaborative filtering.

Parameters

<i>maxIterations</i>	Number of iterations.
<i>alpha</i>	Learning rate for optimization.
<i>Regularization</i>	parameter for optimization.

Definition at line 51 of file bias_svd_method.hpp.

39.181.3 Member Function Documentation

39.181.3.1 Alpha() ^[1/2]

```
double Alpha ( ) const [inline]
```

Get learning rate.

Definition at line 154 of file bias_svd_method.hpp.

39.181.3.2 Alpha() [2/2]

```
double& Alpha ( ) [inline]
```

Modify learning rate.

Definition at line 156 of file `bias_svd_method.hpp`.

39.181.3.3 Apply()

```
void Apply (
    const arma::mat & data,
    const arma::sp_mat & ,
    const size_t rank,
    const size_t maxIterations,
    const double ,
    const bool ) [inline]
```

Apply Collaborative Filtering to the provided data set using the bias SVD.

Parameters

<i>data</i>	Data matrix: dense matrix (coordinate lists) or sparse matrix(cleaned).
<i>cleanedData</i>	item user table in form of sparse matrix.
<i>rank</i>	Rank parameter for matrix factorization.
<i>maxIterations</i>	Maximum number of iterations.
<i>minResidue</i>	Residue required to terminate.
<i>mit</i>	Whether to terminate only when maxIterations is reached.

Definition at line 73 of file `bias_svd_method.hpp`.

References `BiasSVD< OptimizerType >::Apply()`.

39.181.3.4 GetNeighborhood()

```
void GetNeighborhood (
    const arma::Col< size_t > & users,
    const size_t numUsersForSimilarity,
    arma::Mat< size_t > & neighborhood,
    arma::mat & similarities ) const [inline]
```

Get the neighborhood and corresponding similarities for a set of users.

Template Parameters

<i>NeighborSearchPolicy</i>	The policy to perform neighbor search.
-----------------------------	--

Parameters

<i>users</i>	Users whose neighborhood is to be computed.
<i>numUsersForSimilarity</i>	The number of neighbors returned for each user.
<i>neighborhood</i>	Neighbors represented by user IDs.
<i>similarities</i>	Similarity between each user and each of its neighbors.

Definition at line 122 of file `bias_svd_method.hpp`.

39.181.3.5 GetRating()

```
double GetRating (
    const size_t user,
    const size_t item ) const [inline]
```

Return predicted rating given user ID and item ID.

Parameters

<i>user</i>	User ID.
<i>item</i>	Item ID.

Definition at line 91 of file `bias_svd_method.hpp`.

39.181.3.6 GetRatingOfUser()

```
void GetRatingOfUser (
    const size_t user,
    arma::vec & rating ) const [inline]
```

Get predicted ratings for a user.

Parameters

<i>user</i>	User ID.
<i>rating</i>	Resulting rating vector.

Definition at line 104 of file bias_svd_method.hpp.

39.181.3.7 H()

```
const arma::mat& H ( ) const [inline]
```

Get the User Matrix.

Definition at line 142 of file bias_svd_method.hpp.

39.181.3.8 Lambda() [1/2]

```
double Lambda ( ) const [inline]
```

Get regularization parameter.

Definition at line 159 of file bias_svd_method.hpp.

39.181.3.9 Lambda() [2/2]

```
double& Lambda ( ) [inline]
```

Modify regularization parameter.

Definition at line 161 of file bias_svd_method.hpp.

39.181.3.10 MaxIterations() [1/2]

```
size_t MaxIterations ( ) const [inline]
```

Get the number of iterations.

Definition at line 149 of file bias_svd_method.hpp.

39.181.3.11 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Modify the number of iterations.

Definition at line 151 of file bias_svd_method.hpp.

39.181.3.12 P()

```
const arma::vec& P ( ) const [inline]
```

Get the Item Bias Vector.

Definition at line 146 of file bias_svd_method.hpp.

39.181.3.13 Q()

```
const arma::vec& Q ( ) const [inline]
```

Get the User Bias Vector.

Definition at line 144 of file bias_svd_method.hpp.

39.181.3.14 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialization.

Definition at line 167 of file bias_svd_method.hpp.

39.181.3.15 W()

```
const arma::mat& W ( ) const [inline]
```

Get the Item Matrix.

Definition at line 140 of file bias_svd_method.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/ **bias_svd_method.hpp**

39.182 CFModel Class Reference

The model to save to disk.

Public Member Functions

- **CFModel** ()
Create an empty CF model.
- **~CFModel** ()
Clean up memory.
- template<typename DecompositionPolicy >
const **CFTYPE**< DecompositionPolicy > * **CFPtr** () const
Get the pointer to CFTYPE<> object.
- template<typename NeighborSearchPolicy , typename InterpolationPolicy >
void **GetRecommendations** (const size_t numRecs, arma::Mat< size_t > &recommendations, const arma::Mat< size_t > &users)
Compute recommendations for query users.
- template<typename NeighborSearchPolicy , typename InterpolationPolicy >
void **GetRecommendations** (const size_t numRecs, arma::Mat< size_t > &recommendations)
Compute recommendations for all users.
- template<typename NeighborSearchPolicy , typename InterpolationPolicy >
void **Predict** (const arma::Mat< size_t > &combinations, arma::vec &predictions)
Make predictions.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the model.
- template<typename DecompositionPolicy , typename MatType >
void **Train** (const MatType &data, const size_t numUsersForSimilarity, const size_t rank, const size_t maxIterations, const double minResidue, const bool mit)
Train the model.

39.182.1 Detailed Description

The model to save to disk.

Definition at line 110 of file cf_model.hpp.

39.182.2 Constructor & Destructor Documentation

39.182.2.1 CFModel()

```
CFModel ( ) [inline]
```

Create an empty CF model.

Definition at line 129 of file cf_model.hpp.

39.182.2.2 ~CFModel()

```
~ CFModel ( )
```

Clean up memory.

39.182.3 Member Function Documentation

39.182.3.1 CFPtr()

```
const CFTYPE<DecompositionPolicy>* CFPtr ( ) const
```

Get the pointer to CType<> object.

39.182.3.2 GetRecommendations() [1/2]

```
void GetRecommendations (
    const size_t numRecs,
    arma::Mat< size_t > & recommendations,
    const arma::Col< size_t > & users )
```

Compute recommendations for query users.

39.182.3.3 GetRecommendations() [2/2]

```
void GetRecommendations (
    const size_t numRecs,
    arma::Mat< size_t > & recommendations )
```

Compute recommendations for all users.

39.182.3.4 Predict()

```
void Predict (
    const arma::Mat< size_t > & combinations,
    arma::vec & predictions )
```

Make predictions.

39.182.3.5 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the model.

39.182.3.6 Train()

```
void Train (
    const MatType & data,
    const size_t numUsersForSimilarity,
    const size_t rank,
    const size_t maxIterations,
    const double minResidue,
    const bool mit )
```

Train the model.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/ **cf_model.hpp**

39.183 CType< DecompositionPolicy, NormalizationType > Class Template Reference

This class implements Collaborative Filtering (CF).

Public Member Functions

- **CType** (const size_t numUsersForSimilarity=5, const size_t rank=0)
*Initialize the **CType** (p. 1045) object without performing any factorization.*
- template<typename MatType >
CType (const MatType &data, const DecompositionPolicy &decomposition=DecompositionPolicy(), const size_t numUsersForSimilarity=5, const size_t rank=0, const size_t maxIterations=1000, const double minResidue=1e-5, const bool mit=false)
*Initialize the **CType** (p. 1045) object using any decomposition method, immediately factorizing the given data to create a model.*
- const arma::sp_mat & **CleanedData** () const
Get the cleaned data matrix.
- const DecompositionPolicy & **Decomposition** () const
Gets decomposition object.
- template<typename NeighborSearchPolicy = EuclideanSearch, typename InterpolationPolicy = AverageInterpolation>
void **GetRecommendations** (const size_t numRecs, arma::Mat< size_t > &recommendations)
Generates the given number of recommendations for all users.
- template<typename NeighborSearchPolicy = EuclideanSearch, typename InterpolationPolicy = AverageInterpolation>
void **GetRecommendations** (const size_t numRecs, arma::Mat< size_t > &recommendations, const arma::Col< size_t > &users)
Generates the given number of recommendations for the specified users.
- const NormalizationType & **Normalization** () const
Get the normalization object.
- void **NumUsersForSimilarity** (const size_t num)
Sets number of users for calculating similarity.
- size_t **NumUsersForSimilarity** () const

Gets number of users for calculating similarity.

- template<typename NeighborSearchPolicy = EuclideanSearch, typename InterpolationPolicy = AverageInterpolation>
double **Predict** (const size_t user, const size_t item) const

Predict the rating of an item by a particular user.

- template<typename NeighborSearchPolicy = EuclideanSearch, typename InterpolationPolicy = AverageInterpolation>
void **Predict** (const arma::Mat< size_t > &combinations, arma::vec &predictions) const

Predict ratings for each user-item combination in the given coordinate list matrix.

- void **Rank** (const size_t rankValue)

Sets rank parameter for matrix factorization.

- size_t **Rank** () const

Gets rank parameter for matrix factorization.

- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)

*Serialize the **CFTYPE** (p. 1045) model to the given archive.*

- void **Train** (const arma::mat &data, const DecompositionPolicy &decomposition, const size_t maxIterations=1000, const double minResidue=1e-5, const bool mit=false)

*Train the **CFTYPE** (p. 1045) model (i.e.*

- void **Train** (const arma::sp_mat &data, const DecompositionPolicy &decomposition, const size_t maxIterations=1000, const double minResidue=1e-5, const bool mit=false)

*Train the **CFTYPE** (p. 1045) model (i.e.*

Static Public Member Functions

- static void **CleanData** (const arma::mat &data, arma::sp_mat &cleanedData)

Converts the User, Item, Value Matrix to User-Item Table.

39.183.1 Detailed Description

```
template<typename DecompositionPolicy = NMFPolicy, typename NormalizationType = NoNormalization>
class mlpack::cf::CFTYPE< DecompositionPolicy, NormalizationType >
```

This class implements Collaborative Filtering (CF).

This implementation presently supports Alternating Least Squares (ALS) for collaborative filtering.

A simple example of how to run Collaborative Filtering is shown below.

```
extern arma::mat data; // (user, item, rating) table
extern arma::Col<size_t> users; // users seeking recommendations
arma::Mat<size_t> recommendations; // Recommendations

CFTYPE<> cf(data); // Default options.

// Generate 10 recommendations for all users.
cf.GetRecommendations(10, recommendations);

// Generate 10 recommendations for specified users.
cf.GetRecommendations(10, recommendations, users);
```

The data matrix is a (user, item, rating) table. Each column in the matrix should have three rows. The first represents the user; the second represents the item; and the third represents the rating. The user and item, while they are in a matrix that holds doubles, should hold integer (or size_t) values. The user and item indices are assumed to start at 0.

Template Parameters

<i>DecompositionPolicy</i>	The policy used to decompose the rating matrix. It also provides methods to compute prediction and neighborhood.
<i>NormalizationType</i>	The type of normalization performed on raw data. Data is normalized before calling Train() (p. 1052) method. Predicted rating is denormalized before return.

Definition at line 70 of file cf.hpp.

39.183.2 Constructor & Destructor Documentation

39.183.2.1 CType() [1/2]

```
CType (
    const size_t numUsersForSimilarity = 5,
    const size_t rank = 0 )
```

Initialize the **CType** (p. 1045) object without performing any factorization.

Be sure to call **Train()** (p. 1052) before calling **GetRecommendations()** (p. 1048) or any other functions!

39.183.2.2 CType() [2/2]

```
CType (
    const MatType & data,
    const DecompositionPolicy & decomposition = DecompositionPolicy(),
    const size_t numUsersForSimilarity = 5,
    const size_t rank = 0,
    const size_t maxIterations = 1000,
    const double minResidue = 1e-5,
    const bool mit = false )
```

Initialize the **CType** (p. 1045) object using any decomposition method, immediately factorizing the given data to create a model.

There are parameters that can be set; default values are provided for each of them. If the rank is left unset (or is set to 0), a simple density-based heuristic will be used to choose a rank.

The provided dataset can be a coordinate list; that is, a 3-row matrix where each column corresponds to a (user, item, rating) entry in the matrix or a sparse matrix representing (user, item) table.

Template Parameters

<i>MatType</i>	The type of input matrix, which is expected to be either arma::mat (table of (user, item, rating)) or arma::sp_mat (sparse rating matrix where row is item and column is user).
----------------	---

Parameters

<i>data</i>	Data matrix: dense matrix (coordinate lists) or sparse matrix(cleaned).
<i>decomposition</i>	Instantiated DecompositionPolicy object.
<i>numUsersForSimilarity</i>	Size of the neighborhood.
<i>rank</i>	Rank parameter for matrix factorization.
<i>maxIterations</i>	Maximum number of iterations.
<i>minResidue</i>	Residue required to terminate.
<i>mit</i>	Whether to terminate only when maxIterations is reached.

39.183.3 Member Function Documentation

39.183.3.1 CleanData()

```
static void CleanData (
    const arma::mat & data,
    arma::sp_mat & cleanedData ) [static]
```

Converts the User, Item, Value Matrix to User-Item Table.

Referenced by CFTYPE< DecompositionPolicy, NormalizationType >::Normalization().

39.183.3.2 CleanedData()

```
const arma::sp_mat& CleanedData ( ) const [inline]
```

Get the cleaned data matrix.

Definition at line 180 of file cf.hpp.

39.183.3.3 Decomposition()

```
const DecompositionPolicy& Decomposition ( ) const [inline]
```

Gets decomposition object.

Definition at line 177 of file cf.hpp.

39.183.3.4 GetRecommendations() [1/2]

```
void GetRecommendations (
    const size_t numRecs,
    arma::Mat< size_t > & recommendations )
```

Generates the given number of recommendations for all users.

Template Parameters

<i>NeighborSearchPolicy</i>	The policy used to search neighbors of query set in referece set.
<i>InterpolationPolicy</i>	The policy used to calculate interpolation weights.

Parameters

<i>numRecs</i>	Number of Recommendations.
<i>recommendations</i>	Matrix to save recommendations into.

Referenced by CType< DecompositionPolicy, NormalizationType >::Normalization().

39.183.3.5 GetRecommendations() [2/2]

```
void GetRecommendations (
    const size_t numRecs,
    arma::Mat< size_t > & recommendations,
    const arma::Col< size_t > & users )
```

Generates the given number of recommendations for the specified users.

Template Parameters

<i>NeighborSearchPolicy</i>	The policy used to search neighbors of query set in referece set.
<i>InterpolationPolicy</i>	The policy used to calculate interpolation weights.

Parameters

<i>numRecs</i>	Number of Recommendations.
<i>recommendations</i>	Matrix to save recommendations.
<i>users</i>	Users for which recommendations are to be generated.

39.183.3.6 Normalization()

```
const NormalizationType& Normalization ( ) const [inline]
```

Get the normalization object.

Definition at line 183 of file cf.hpp.

References CType< DecompositionPolicy, NormalizationType >::CleanData(), CType< DecompositionPolicy, NormalizationType >::GetRecommendations(), CType< DecompositionPolicy, NormalizationType >::Predict(), and CType< DecompositionPolicy, NormalizationType >::serialize().

39.183.3.7 NumUsersForSimilarity() [1/2]

```
void NumUsersForSimilarity (
    const size_t num ) [inline]
```

Sets number of users for calculating similarity.

Definition at line 147 of file cf.hpp.

References Log::Warn.

39.183.3.8 NumUsersForSimilarity() [2/2]

```
size_t NumUsersForSimilarity ( ) const [inline]
```

Gets number of users for calculating similarity.

Definition at line 159 of file cf.hpp.

39.183.3.9 Predict() [1/2]

```
double Predict (
    const size_t user,
    const size_t item ) const
```

Predict the rating of an item by a particular user.

Template Parameters

<i>NeighborSearchPolicy</i>	The policy used to search neighbors of query set in referece set.
<i>InterpolationPolicy</i>	The policy used to calculate interpolation weights.

Parameters

<i>user</i>	User to predict for.
<i>item</i>	Item to predict for.

Referenced by CType< DecompositionPolicy, NormalizationType >::Normalization().

39.183.3.10 Predict() [2/2]

```
void Predict (
    const arma::Mat< size_t > & combinations,
    arma::vec & predictions ) const
```

Predict ratings for each user-item combination in the given coordinate list matrix.

The matrix 'combinations' should have two rows and number of columns equal to the number of desired predictions. The first element of each column corresponds to the user index, and the second element of each column corresponds to the item index. The output vector 'predictions' will have length equal to combinations.n_cols, and predictions[i] will be equal to the prediction for the user/item combination in combinations.col(i).

Template Parameters

<i>NeighborSearchPolicy</i>	The policy used to search neighbors of query set in referece set.
<i>InterpolationPolicy</i>	The policy used to calculate interpolation weights.

Parameters

<i>combinations</i>	User/item combinations to predict.
<i>predictions</i>	Predicted ratings for each user/item combination.

39.183.3.11 Rank() [1/2]

```
void Rank (
    const size_t rankValue ) [inline]
```

Sets rank parameter for matrix factorization.

Definition at line 165 of file cf.hpp.

39.183.3.12 Rank() [2/2]

```
size_t Rank ( ) const [inline]
```

Gets rank parameter for matrix factorization.

Definition at line 171 of file cf.hpp.

39.183.3.13 `serialize()`

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the **CFTYPE** (p. 1045) model to the given archive.

Referenced by `CFTYPE< DecompositionPolicy, NormalizationType >::Normalization()`.

39.183.3.14 `Train()` [1/2]

```
void Train (
    const arma::mat & data,
    const DecompositionPolicy & decomposition,
    const size_t maxIterations = 1000,
    const double minResidue = 1e-5,
    const bool mit = false )
```

Train the **CFTYPE** (p. 1045) model (i.e.

factorize the input matrix) using the parameters that have already been set for the model (specifically, the rank parameter), and optionally, using the given `DecompositionPolicy`.

Parameters

<i>data</i>	Input dataset; dense matrix (coordinate lists).
<i>decomposition</i>	Instantiated <code>DecompositionPolicy</code> object.
<i>maxIterations</i>	Maximum number of iterations.
<i>minResidue</i>	Residue required to terminate.
<i>mit</i>	Whether to terminate only when maxIterations is reached.

39.183.3.15 `Train()` [2/2]

```
void Train (
    const arma::sp_mat & data,
    const DecompositionPolicy & decomposition,
    const size_t maxIterations = 1000,
    const double minResidue = 1e-5,
    const bool mit = false )
```

Train the **CFTYPE** (p. 1045) model (i.e.

factorize the input matrix) using the parameters that have already been set for the model (specifically, the rank parameter), and optionally, using the given `DecompositionPolicy`.

Parameters

<i>data</i>	Input dataset; sparse matrix (user item table).
<i>decomposition</i>	Instantiated DecompositionPolicy object.
<i>maxIterations</i>	Maximum number of iterations.
<i>minResidue</i>	Residue required to terminate.
<i>mit</i>	Whether to terminate only when maxIterations is reached.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/ **cf.hpp**

39.184 CombinedNormalization< NormalizationTypes > Class Template Reference

This normalization class performs a sequence of normalization methods on raw ratings.

Public Types

- using **TupleType** = std::tuple< NormalizationTypes... >

Public Member Functions

- **CombinedNormalization** ()
- double **Denormalize** (const size_t user, const size_t item, const double rating) const
*Denormalize rating by calling **Denormalize()** (p. 1054) in each normalization object.*
- void **Denormalize** (const arma::Mat< size_t > &combinations, arma::vec &predictions) const
*Denormalize rating by calling **Denormalize()** (p. 1054) in each normalization object.*
- const **TupleType** & **Normalizations** () const
Return normalizations tuple.
- template<typename MatType >
void **Normalize** (MatType &data)
*Normalize the data by calling **Normalize()** (p. 1056) in each normalization object.*
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int version)
Serialization.

39.184.1 Detailed Description

```
template<typename... NormalizationTypes>
class mlpack::cf::CombinedNormalization< NormalizationTypes >
```

This normalization class performs a sequence of normalization methods on raw ratings.

An example of how to use **CombinedNormalization** (p. 1053) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

CFTYPE<NMFPolicy,
    CombinedNormalization<
        OverallMeanNormalization,
        UserMeanNormalization,
        ItemMeanNormalization>> cf(data);

// Generate 10 recommendations for all users.
cf.GetRecommendations(10, recommendations);
```

Definition at line 44 of file combined_normalization.hpp.

39.184.2 Member Typedef Documentation

39.184.2.1 TupleType

```
using TupleType = std::tuple<NormalizationTypes...>
```

Definition at line 47 of file combined_normalization.hpp.

39.184.3 Constructor & Destructor Documentation

39.184.3.1 CombinedNormalization()

```
CombinedNormalization ( ) [inline]
```

Definition at line 50 of file combined_normalization.hpp.

39.184.4 Member Function Documentation

39.184.4.1 Denormalize() [1/2]

```
double Denormalize (
    const size_t user,
    const size_t item,
    const double rating ) const [inline]
```

Denormalize rating by calling **Denormalize()** (p. 1054) in each normalization object.

Note that the order of objects calling **Denormalize()** (p. 1054) should be the reversed order of objects calling **Normalize()** (p. 1056).

Parameters

<i>user</i>	User ID.
<i>item</i>	Item ID.
<i>rating</i>	Computed rating before denormalization.

Definition at line 72 of file combined_normalization.hpp.

Referenced by CombinedNormalization< NormalizationTypes >::serialize().

39.184.4.2 Denormalize() [2/2]

```
void Denormalize (
    const arma::Mat< size_t > & combinations,
    arma::vec & predictions ) const [inline]
```

Denormalize rating by calling **Denormalize()** (p. 1054) in each normalization object.

Note that the order of objects calling **Denormalize()** (p. 1054) should be the reversed order of objects calling **Normalize()** (p. 1056).

Parameters

<i>combinations</i>	User/Item combinations.
<i>predictions</i>	Predicted ratings for each user/item combination.

Definition at line 87 of file combined_normalization.hpp.

39.184.4.3 Normalizations()

```
const TupleType& Normalizations ( ) const [inline]
```

Return normalizations tuple.

Definition at line 96 of file combined_normalization.hpp.

39.184.4.4 Normalize()

```
void Normalize (
    MatType & data ) [inline]
```

Normalize the data by calling **Normalize()** (p. 1056) in each normalization object.

Parameters

<i>data</i>	Input dataset.
-------------	----------------

Definition at line 58 of file combined_normalization.hpp.

Referenced by CombinedNormalization< NormalizationTypes >::serialize().

39.184.4.5 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int version ) [inline]
```

Serialization.

Definition at line 105 of file combined_normalization.hpp.

References CombinedNormalization< NormalizationTypes >::Denormalize(), and CombinedNormalization< NormalizationTypes >::Normalize().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/ **combined_normalization.hpp**

39.185 CosineSearch Class Reference

Nearest neighbor search with cosine distance.

Public Member Functions

- **CosineSearch** (const arma::mat &referenceSet)
Constructor with reference set.
- void **Search** (const arma::mat &query, const size_t k, arma::Mat< size_t > &neighbors, arma::mat &similarities)
Given a set of query points, find the nearest k neighbors, and return similarities.

39.185.1 Detailed Description

Nearest neighbor search with cosine distance.

Note that, with normalized vectors, neighbor search with cosine distance is equivalent to neighbor search with Euclidean distance. Therefore, instead of performing neighbor search directly with cosine distance, we first normalize all vectors to unit length, and then use **neighbor::KNN** (p. 432) (i.e. NeighborSearch with Euclidean distance, KDTree). Cosine similarities are calculated from Euclidean distance.

An example of how to use **CosineSearch** (p. 1056) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

CFType<> cf(data);

// Generate 10 recommendations for all users.
cf.template GetRecommendations<CosineSearch>(10, recommendations);
```

Definition at line 44 of file cosine_search.hpp.

39.185.2 Constructor & Destructor Documentation

39.185.2.1 CosineSearch()

```
CosineSearch (
    const arma::mat & referenceSet ) [inline]
```

Constructor with reference set.

All vectors in reference set are normalized to unit length.

Parameters

Set	of reference points.
-----	----------------------

Definition at line 53 of file cosine_search.hpp.

References NeighborSearch< SortPolicy, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTreeTraversalType >::Train().

39.185.3 Member Function Documentation

39.185.3.1 Search()

```
void Search (
    const arma::mat & query,
    const size_t k,
    arma::Mat< size_t > & neighbors,
    arma::mat & similarities ) [inline]
```

Given a set of query points, find the nearest k neighbors, and return similarities.

Similarities are non-negative and no larger than one.

Parameters

<i>query</i>	A set of query points.
<i>k</i>	Number of neighbors to search.
<i>neighbors</i>	Nearest neighbors.
<i>similarites</i>	Similarities between query point and its neighbors.

Definition at line 70 of file cosine_search.hpp.

References NeighborSearch< SortPolicy, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTreeTraversalType >::Search().

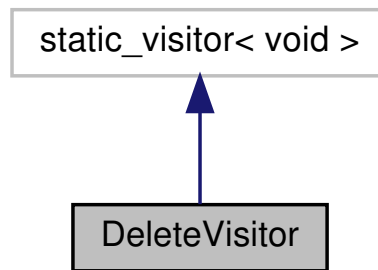
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/neighbor_search_policies/ **cosine_search.hpp**

39.186 DeleteVisitor Class Reference

DeleteVisitor (p. 1058) deletes the CFTYPE<> object which is pointed to by the variable cf in class **CFModel** (p. 1042).

Inheritance diagram for DeleteVisitor:



Public Member Functions

- `template<typename DecompositionPolicy >`
`void operator() (CType< DecompositionPolicy > *c) const`
*Delete **CType** (p. 1045) object.*

39.186.1 Detailed Description

DeleteVisitor (p. 1058) deletes the `CType<>` object which is pointed to by the variable `cf` in class **CFModel** (p. 1042).

Definition at line 34 of file `cf_model.hpp`.

39.186.2 Member Function Documentation

39.186.2.1 `operator()`

```
void operator() (
    CType< DecompositionPolicy > * c ) const
```

Delete **CType** (p. 1045) object.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/ cf_model.hpp`

39.187 DummyClass Class Reference

This class acts as a dummy class for passing as template parameter.

39.187.1 Detailed Description

This class acts as a dummy class for passing as template parameter.

Passing this class as a template parameter to class **SVDWrapper** (p. 1108) will force **SVDWrapper** (p. 1108) to use Armadillo's SVD implementation.

Definition at line 27 of file `svd_wrapper.hpp`.

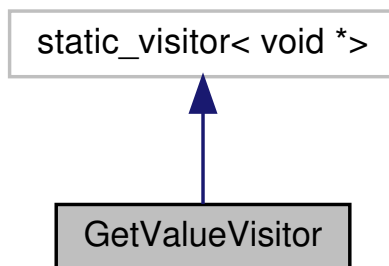
The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/ svd_wrapper.hpp`

39.188 GetValueVisitor Class Reference

GetValueVisitor (p. 1060) returns the pointer which points to the **CType** (p. 1045) object.

Inheritance diagram for GetValueVisitor:



Public Member Functions

- `template<typename DecompositionPolicy >`
`void * operator() (CType< DecompositionPolicy > *c) const`
Return stored pointer as void type.*

39.188.1 Detailed Description

GetValueVisitor (p. 1060) returns the pointer which points to the **CType** (p. 1045) object.

Definition at line 45 of file `cf_model.hpp`.

39.188.2 Member Function Documentation

39.188.2.1 operator()()

```
void* operator() (
    CFType< DecompositionPolicy > * c ) const
```

Return stored pointer as void* type.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/ **cf_model.hpp**

39.189 ItemMeanNormalization Class Reference

This normalization class performs item mean normalization on raw ratings.

Public Member Functions

- **ItemMeanNormalization** ()
- double **Denormalize** (const size_t, const size_t item, const double rating) const
Denormalize computed rating by adding item mean.
- void **Denormalize** (const arma::Mat< size_t > &combinations, arma::vec &predictions) const
Denormalize computed rating by adding item mean.
- const arma::vec & **Mean** () const
Return item mean.
- void **Normalize** (arma::mat &data)
Normalize the data by subtracting item mean from each of existing ratings.
- void **Normalize** (arma::sp_mat &cleanedData)
Normalize the data by subtracting item mean from each of existing ratings.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialization.

39.189.1 Detailed Description

This normalization class performs item mean normalization on raw ratings.

An example of how to use **ItemMeanNormalization** (p. 1061) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

// Use ItemMeanNormalization as normalization method.
CFType<NMFPolicy, ItemMeanNormalization> cf(data);

// Generate 10 recommendations for all users.
cf.GetRecommendations(10, recommendations);
```

Definition at line 39 of file item_mean_normalization.hpp.

39.189.2 Constructor & Destructor Documentation

39.189.2.1 ItemMeanNormalization()

```
ItemMeanNormalization ( ) [inline]
```

Definition at line 43 of file item_mean_normalization.hpp.

39.189.3 Member Function Documentation

39.189.3.1 Denormalize() [1/2]

```
double Denormalize (
    const size_t ,
    const size_t item,
    const double rating ) const [inline]
```

Denormalize computed rating by adding item mean.

Parameters

<i>user</i>	User ID.
<i>item</i>	Item ID.
<i>rating</i>	Computed rating before denormalization.

Definition at line 127 of file item_mean_normalization.hpp.

39.189.3.2 Denormalize() [2/2]

```
void Denormalize (
    const arma::Mat< size_t > & combinations,
    arma::vec & predictions ) const [inline]
```

Denormalize computed rating by adding item mean.

Parameters

<i>combinations</i>	User/Item combinations.
<i>predictions</i>	Predicted ratings for each user/item combination.

Definition at line 140 of file `item_mean_normalization.hpp`.

39.189.3.3 Mean()

```
const arma::vec& Mean ( ) const [inline]
```

Return item mean.

Definition at line 153 of file `item_mean_normalization.hpp`.

39.189.3.4 Normalize() [1/2]

```
void Normalize (
    arma::mat & data ) [inline]
```

Normalize the data by subtracting item mean from each of existing ratings.

Parameters

<i>data</i>	Input dataset in the form of coordinate list.
-------------	---

Definition at line 50 of file `item_mean_normalization.hpp`.

39.189.3.5 Normalize() [2/2]

```
void Normalize (
    arma::sp_mat & cleanedData ) [inline]
```

Normalize the data by subtracting item mean from each of existing ratings.

Parameters

<i>cleanedData</i>	Input data as a sparse matrix.
--------------------	--------------------------------

Definition at line 90 of file `item_mean_normalization.hpp`.

39.189.3.6 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialization.

Definition at line 159 of file `item_mean_normalization.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/ item_mean_normalization.hpp`

39.190 LMetricSearch< TPower > Class Template Reference

Nearest neighbor search with L_p distance.

Public Types

- using **NeighborSearchType** = `neighbor::NeighborSearch< neighbor::NearestNeighborSort, metric::L \leftarrow Metric< TPower, true > >`

Public Member Functions

- **LMetricSearch** (const arma::mat &referenceSet)
- void **Search** (const arma::mat &query, const size_t k, arma::Mat< size_t > &neighbors, arma::mat &similarities)
Given a set of query points, find the nearest k neighbors, and return similarites.

39.190.1 Detailed Description

```
template<int TPower>
class mlpack::cf::LMetricSearch< TPower >
```

Nearest neighbor search with L_p distance.

An example of how to use **LMetricSearch** (p. 1064) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

CFType<> cf(data);

// Generate 10 recommendations for all users.
cf.template GetRecommendations<LMetricSearch<2>>>(10, recommendations);
```


Template Parameters

<i>TPower</i>	Power of metric.
---------------	------------------

Definition at line 42 of file lmetric_search.hpp.

39.190.2 Member Typedef Documentation

39.190.2.1 NeighborSearchType

```
using NeighborSearchType = neighbor::NeighborSearch< neighbor::NearestNeighborSort, metric::↔
LMetric<TPower, true> >
```

Definition at line 47 of file lmetric_search.hpp.

39.190.3 Constructor & Destructor Documentation

39.190.3.1 LMetricSearch()

```
LMetricSearch (
    const arma::mat & referenceSet ) [inline]
```

Parameters

<i>Set</i>	of reference points.
------------	----------------------

Definition at line 52 of file lmetric_search.hpp.

39.190.4 Member Function Documentation

39.190.4.1 Search()

```
void Search (
    const arma::mat & query,
```

```
const size_t k,
arma::Mat< size_t > & neighbors,
arma::mat & similarities ) [inline]
```

Given a set of query points, find the nearest k neighbors, and return similarities.

Similarities are non-negative and no larger than one.

Parameters

<i>query</i>	A set of query points.
<i>k</i>	Number of neighbors to search.
<i>neighbors</i>	Nearest neighbors.
<i>similarities</i>	Similarities between query point and its neighbors.

Definition at line 64 of file `lmetric_search.hpp`.

References `NeighborSearch< SortPolicy, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTreeTraversalType >::Search()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/neighbor_search_policies/ lmetric_search.hpp`

39.191 NMFPolicy Class Reference

Implementation of the NMF policy to act as a wrapper when accessing NMF from within **CType** (p. 1045).

Public Member Functions

- `template<typename MatType >`
void **Apply** (const MatType &, const arma::sp_mat &cleanedData, const size_t rank, const size_t maxIterations, const double minResidue, const bool mit)
Apply Collaborative Filtering to the provided dataset using NMF method.
- `template<typename NeighborSearchPolicy >`
void **GetNeighborhood** (const arma::Col< size_t > &users, const size_t numUsersForSimilarity, arma::Mat< size_t > &neighborhood, arma::mat &similarities) const
Get the neighborhood and corresponding similarities for a set of users.
- double **GetRating** (const size_t user, const size_t item) const
Return predicted rating given user ID and item ID.
- void **GetRatingOfUser** (const size_t user, arma::vec &rating) const
Get predicted ratings for a user.
- const arma::mat & **H** () const
Get the User Matrix.
- `template<typename Archive >`
void **serialize** (Archive &ar, const unsigned int)
Serialization.
- const arma::mat & **W** () const
Get the Item Matrix.

39.191.1 Detailed Description

Implementation of the NMF policy to act as a wrapper when accessing NMF from within **CType** (p. 1045).

An example of how to use **NMFPolicy** (p. 1066) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

CType<NMFPolicy> cf(data);

// Generate 10 recommendations for all users.
cf.GetRecommendations(10, recommendations);
```

Definition at line 43 of file nmf_method.hpp.

39.191.2 Member Function Documentation

39.191.2.1 Apply()

```
void Apply (
    const MatType & ,
    const arma::sp_mat & cleanedData,
    const size_t rank,
    const size_t maxIterations,
    const double minResidue,
    const bool mit ) [inline]
```

Apply Collaborative Filtering to the provided dataset using NMF method.

Parameters

<i>data</i>	Data matrix: dense matrix (coordinate lists) or sparse matrix(cleaned).
<i>cleanedData</i>	item user table in form of sparse matrix.
<i>rank</i>	Rank parameter for matrix factorization.
<i>maxIterations</i>	Maximum number of iterations.
<i>minResidue</i>	Residue required to terminate.
<i>mit</i>	Whether to terminate only when maxIterations is reached.

Definition at line 58 of file nmf_method.hpp.

References `AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::Apply()`.

39.191.2.2 GetNeighborhood()

```
void GetNeighborhood (
    const arma::Col< size_t > & users,
    const size_t numUsersForSimilarity,
    arma::Mat< size_t > & neighborhood,
    arma::mat & similarities ) const [inline]
```

Get the neighborhood and corresponding similarities for a set of users.

Template Parameters

<i>NeighborSearchPolicy</i>	The policy to perform neighbor search.
-----------------------------	--

Parameters

<i>users</i>	Users whose neighborhood is to be computed.
<i>numUsersForSimilarity</i>	The number of neighbors returned for each user.
<i>neighborhood</i>	Neighbors represented by user IDs.
<i>similarities</i>	Similarity between each user and each of its neighbors.

Definition at line 120 of file nmf_method.hpp.

39.191.2.3 GetRating()

```
double GetRating (
    const size_t user,
    const size_t item ) const [inline]
```

Return predicted rating given user ID and item ID.

Parameters

<i>user</i>	User ID.
<i>item</i>	Item ID.

Definition at line 90 of file nmf_method.hpp.

39.191.2.4 GetRatingOfUser()

```
void GetRatingOfUser (
    const size_t user,
    arma::vec & rating ) const [inline]
```

Get predicted ratings for a user.

Parameters

<i>user</i>	User ID.
<i>rating</i>	Resulting rating vector.

Definition at line 102 of file nmf_method.hpp.

39.191.2.5 H()

```
const arma::mat& H ( ) const [inline]
```

Get the User Matrix.

Definition at line 149 of file nmf_method.hpp.

39.191.2.6 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialization.

Definition at line 155 of file nmf_method.hpp.

39.191.2.7 W()

```
const arma::mat& W ( ) const [inline]
```

Get the Item Matrix.

Definition at line 147 of file nmf_method.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/ **nmf_method.hpp**

39.192 NoNormalization Class Reference

This normalization class doesn't perform any normalization.

Public Member Functions

- **NoNormalization** ()
- double **Denormalize** (const size_t, const size_t, const double rating) const
Do nothing.
- void **Denormalize** (const arma::Mat< size_t > &, const arma::vec &) const
Do nothing.
- template<typename MatType >
void **Normalize** (const MatType &) const
Do nothing.
- template<typename Archive >
void **serialize** (Archive &, const unsigned int)
Serialization.

39.192.1 Detailed Description

This normalization class doesn't perform any normalization.

It is the default normalization type for CF class.

Definition at line 25 of file no_normalization.hpp.

39.192.2 Constructor & Destructor Documentation

39.192.2.1 NoNormalization()

```
NoNormalization ( ) [inline]
```

Definition at line 29 of file no_normalization.hpp.

39.192.3 Member Function Documentation

39.192.3.1 Denormalize() [1/2]

```
double Denormalize (
    const size_t ,
    const size_t ,
    const double rating ) const [inline]
```

Do nothing.

Parameters

<i>user</i>	User ID.
<i>item</i>	Item ID.
<i>rating</i>	Computed rating before denormalization.

Definition at line 46 of file no_normalization.hpp.

39.192.3.2 Denormalize() [2/2]

```
void Denormalize (
    const arma::Mat< size_t > & ,
    const arma::vec & ) const [inline]
```

Do nothing.

Parameters

<i>combinations</i>	User/Item combinations.
<i>predictions</i>	Predicted ratings for each user/item combination.

Definition at line 59 of file no_normalization.hpp.

39.192.3.3 Normalize()

```
void Normalize (
    const MatType & ) const [inline]
```

Do nothing.

Parameters

<i>data</i>	Input dataset.
-------------	----------------

Definition at line 37 of file no_normalization.hpp.

39.192.3.4 serialize()

```
void serialize (
```

```
Archive & ,
const unsigned int ) [inline]
```

Serialization.

Definition at line 67 of file no_normalization.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/ **no_normalization.hpp**

39.193 OverallMeanNormalization Class Reference

This normalization class performs overall mean normalization on raw ratings.

Public Member Functions

- **OverallMeanNormalization** ()
- double **Denormalize** (const size_t, const size_t, const double rating) const
Denormalize computed rating by adding mean.
- void **Denormalize** (const arma::Mat< size_t > &, arma::vec &predictions) const
Denormalize computed rating by adding mean.
- double **Mean** () const
Return mean.
- void **Normalize** (arma::mat &data)
Normalize the data by subtracting the mean of all existing ratings.
- void **Normalize** (arma::sp_mat &cleanedData)
Normalize the data by subtracting the mean of all existing ratings.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialization.

39.193.1 Detailed Description

This normalization class performs overall mean normalization on raw ratings.

An example of how to use **OverallMeanNormalization** (p. 1072) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

// Use OverallMeanNormalization as normalization method.
CFType<NMFPolicy, OverallMeanNormalization> cf(data);

// Generate 10 recommendations for all users.
cf.GetRecommendations(10, recommendations);
```

Definition at line 39 of file overall_mean_normalization.hpp.

39.193.2 Constructor & Destructor Documentation

39.193.2.1 OverallMeanNormalization()

```
OverallMeanNormalization ( ) [inline]
```

Definition at line 43 of file overall_mean_normalization.hpp.

39.193.3 Member Function Documentation

39.193.3.1 Denormalize() [1/2]

```
double Denormalize (
    const size_t ,
    const size_t ,
    const double rating ) const [inline]
```

Denormalize computed rating by adding mean.

Parameters

<i>user</i>	User ID.
<i>item</i>	Item ID.
<i>rating</i>	Computed rating before denormalization.

Definition at line 100 of file overall_mean_normalization.hpp.

39.193.3.2 Denormalize() [2/2]

```
void Denormalize (
    const arma::Mat< size_t > & ,
    arma::vec & predictions ) const [inline]
```

Denormalize computed rating by adding mean.

Parameters

<i>combinations</i>	User/Item combinations.
<i>predictions</i>	Predicted ratings for each user/item combination.

Definition at line 113 of file overall_mean_normalization.hpp.

39.193.3.3 Mean()

```
double Mean ( ) const [inline]
```

Return mean.

Definition at line 122 of file overall_mean_normalization.hpp.

39.193.3.4 Normalize() [1/2]

```
void Normalize (
    arma::mat & data ) [inline]
```

Normalize the data by subtracting the mean of all existing ratings.

Parameters

<i>data</i>	Input dataset in the form of coordinate list.
-------------	---

Definition at line 50 of file overall_mean_normalization.hpp.

39.193.3.5 Normalize() [2/2]

```
void Normalize (
    arma::sp_mat & cleanedData ) [inline]
```

Normalize the data by subtracting the mean of all existing ratings.

Parameters

<i>cleanedData</i>	Input data as a sparse matrix.
--------------------	--------------------------------

Definition at line 68 of file overall_mean_normalization.hpp.

39.193.3.6 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialization.

Definition at line 131 of file overall_mean_normalization.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/ **overall_mean_normalization.hpp**

39.194 PearsonSearch Class Reference

Nearest neighbor search with pearson distance (or furthest neighbor search with pearson correlation).

Public Member Functions

- **PearsonSearch** (const arma::mat &referenceSet)
Constructor with reference set.
- void **Search** (const arma::mat &query, const size_t k, arma::Mat< size_t > &neighbors, arma::mat &similarities)
Given a set of query points, find the nearest k neighbors, and return similarities.

39.194.1 Detailed Description

Nearest neighbor search with pearson distance (or furthest neighbor search with pearson correlation).

Note that, with normalized vectors, neighbor search with pearson distance is equivalent to neighbor search with Euclidean distance. Therefore, instead of performing neighbor search directly with pearson distance, we first normalize all vectors, and then use **neighbor::KNN** (p. 432) (i.e. NeighborSearch with Euclidean distance, KDTree). Pearson correlation are calculated from Euclidean distance.

An example of how to use **PearsonSearch** (p. 1075) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

CFTYPE<> cf(data);

// Generate 10 recommendations for all users.
cf.template GetRecommendations<PearsonSearch>(10, recommendations);
```

Definition at line 45 of file pearson_search.hpp.

39.194.2 Constructor & Destructor Documentation

39.194.2.1 PearsonSearch()

```
PearsonSearch (
    const arma::mat & referenceSet ) [inline]
```

Constructor with reference set.

In order to use **neighbor::KNN** (p. 432)(i.e. NeighborSearch with Euclidean distance, KDTree), we need to normalize all vectors in referenceSet. For each vector x , we first subtract $\text{mean}(x)$ from each element in x . Then, we normalize the vector to unit length.

Parameters

<i>Set</i>	of reference points.
------------	----------------------

Definition at line 57 of file pearson_search.hpp.

References NeighborSearch< SortPolicy, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTree↵ TraversalType >::Train().

39.194.3 Member Function Documentation

39.194.3.1 Search()

```
void Search (
    const arma::mat & query,
    const size_t k,
    arma::Mat< size_t > & neighbors,
    arma::mat & similarities ) [inline]
```

Given a set of query points, find the nearest k neighbors, and return similarities.

Similarities are non-negative and no larger than one.

Parameters

<i>query</i>	A set of query points.
<i>k</i>	Number of neighbors to search.
<i>neighbors</i>	Nearest neighbors.
<i>similarites</i>	Similarities between query point and its neighbors.

Definition at line 78 of file pearson_search.hpp.

References NeighborSearch< SortPolicy, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTreeTraversalType >::Search().

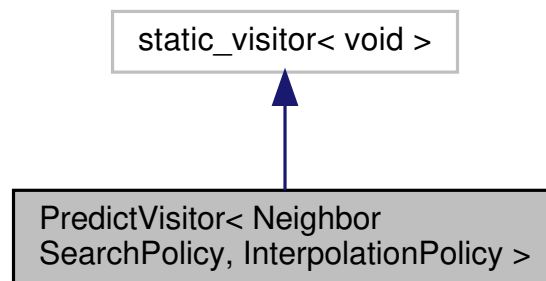
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/neighbor_search_policies/ **pearson_search.hpp**

39.195 PredictVisitor< NeighborSearchPolicy, InterpolationPolicy > Class Template Reference

PredictVisitor (p. 1077) uses the **CFTYPE** (p. 1045) object to make predictions on the given combinations of users and items.

Inheritance diagram for PredictVisitor< NeighborSearchPolicy, InterpolationPolicy >:



Public Member Functions

- **PredictVisitor** (const arma::Mat< size_t > &combinations, arma::vec &predictions)
Visitor constructor.
- template<typename DecompositionPolicy >
void **operator()** (**CFTYPE**< DecompositionPolicy > *c) const
Predict ratings for each user-item combination.

39.195.1 Detailed Description

```
template<typename NeighborSearchPolicy, typename InterpolationPolicy>
class mlpack::cf::PredictVisitor< NeighborSearchPolicy, InterpolationPolicy >
```

PredictVisitor (p. 1077) uses the **CFTYPE** (p. 1045) object to make predictions on the given combinations of users and items.

Definition at line 59 of file cf_model.hpp.

39.195.2 Constructor & Destructor Documentation

39.195.2.1 PredictVisitor()

```
PredictVisitor (
    const arma::Mat< size_t > & combinations,
    arma::vec & predictions )
```

Visitor constructor.

39.195.3 Member Function Documentation

39.195.3.1 operator>()

```
void operator() (
    CFTYPE< DecompositionPolicy > * c ) const
```

Predict ratings for each user-item combination.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/ **cf_model.hpp**

39.196 RandomizedSVDPolicy Class Reference

Implementation of the Randomized SVD policy to act as a wrapper when accessing Randomized SVD from within **CFTYPE** (p. 1045).

Public Member Functions

- **RandomizedSVDPolicy** (const size_t iteratedPower=0, const size_t maxIterations=2)
Use randomized SVD method to perform collaborative filtering.
- template<typename MatType >
void **Apply** (const MatType &, const arma::sp_mat &cleanedData, const size_t rank, const size_t maxIterations, const double, const bool)
Apply Collaborative Filtering to the provided data set using the randomized SVD.
- template<typename NeighborSearchPolicy >
void **GetNeighborhood** (const arma::Col< size_t > &users, const size_t numUsersForSimilarity, arma::Mat< size_t > &neighborhood, arma::mat &similarities) const
Get the neighborhood and corresponding similarities for a set of users.
- double **GetRating** (const size_t user, const size_t item) const
Return predicted rating given user ID and item ID.
- void **GetRatingOfUser** (const size_t user, arma::vec &rating) const
Get predicted ratings for a user.
- const arma::mat & **H** () const
Get the User Matrix.
- size_t **IteratedPower** () const
Get the size of the normalized power iterations.
- size_t & **IteratedPower** ()
Modify the size of the normalized power iterations.
- size_t **MaxIterations** () const
Get the number of iterations.
- size_t & **MaxIterations** ()
Modify the number of iterations.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialization.
- const arma::mat & **W** () const
Get the Item Matrix.

39.196.1 Detailed Description

Implementation of the Randomized SVD policy to act as a wrapper when accessing Randomized SVD from within **CFTYPE** (p. 1045).

An example of how to use **RandomizedSVDPolicy** (p. 1078) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

CFTYPE<RandomizedSVDPolicy> cf(data);

// Generate 10 recommendations for all users.
cf.GetRecommendations(10, recommendations);
```

Definition at line 41 of file randomized_svd_method.hpp.

39.196.2 Constructor & Destructor Documentation

39.196.2.1 RandomizedSVDPolicy()

```
RandomizedSVDPolicy (
    const size_t iteratedPower = 0,
    const size_t maxIterations = 2 ) [inline]
```

Use randomized SVD method to perform collaborative filtering.

Parameters

<i>iteratedPower</i>	Size of the normalized power iterations (Default: rank + 2).
<i>maxIterations</i>	Number of iterations for the power method (Default: 2).

Definition at line 52 of file randomized_svd_method.hpp.

39.196.3 Member Function Documentation

39.196.3.1 Apply()

```
void Apply (
    const MatType & ,
    const arma::sp_mat & cleanedData,
    const size_t rank,
    const size_t maxIterations,
    const double ,
    const bool ) [inline]
```

Apply Collaborative Filtering to the provided data set using the randomized SVD.

Parameters

<i>data</i>	Data matrix: dense matrix (coordinate lists) or sparse matrix(cleaned).
<i>cleanedData</i>	item user table in form of sparse matrix.
<i>rank</i>	Rank parameter for matrix factorization.
<i>maxIterations</i>	Maximum number of iterations.
<i>minResidue</i>	Residue required to terminate.
<i>mit</i>	Whether to terminate only when maxIterations is reached.

Definition at line 73 of file randomized_svd_method.hpp.

References RandomizedSVD::Apply().

39.196.3.2 GetNeighborhood()

```
void GetNeighborhood (
    const arma::Col< size_t > & users,
    const size_t numUsersForSimilarity,
    arma::Mat< size_t > & neighborhood,
    arma::mat & similarities ) const [inline]
```

Get the neighborhood and corresponding similarities for a set of users.

Template Parameters

<i>NeighborSearchPolicy</i>	The policy to perform neighbor search.
-----------------------------	--

Parameters

<i>users</i>	Users whose neighborhood is to be computed.
<i>numUsersForSimilarity</i>	The number of neighbors returned for each user.
<i>neighborhood</i>	Neighbors represented by user IDs.
<i>similarities</i>	Similarity between each user and each of its neighbors.

Definition at line 129 of file randomized_svd_method.hpp.

39.196.3.3 GetRating()

```
double GetRating (
    const size_t user,
    const size_t item ) const [inline]
```

Return predicted rating given user ID and item ID.

Parameters

<i>user</i>	User ID.
<i>item</i>	Item ID.

Definition at line 99 of file randomized_svd_method.hpp.

39.196.3.4 GetRatingOfUser()

```
void GetRatingOfUser (
    const size_t user,
    arma::vec & rating ) const [inline]
```

Get predicted ratings for a user.

Parameters

<i>user</i>	User ID.
<i>rating</i>	Resulting rating vector.

Definition at line 111 of file randomized_svd_method.hpp.

39.196.3.5 H()

```
const arma::mat& H ( ) const [inline]
```

Get the User Matrix.

Definition at line 158 of file randomized_svd_method.hpp.

39.196.3.6 IteratedPower() [1/2]

```
size_t IteratedPower ( ) const [inline]
```

Get the size of the normalized power iterations.

Definition at line 161 of file randomized_svd_method.hpp.

39.196.3.7 IteratedPower() [2/2]

```
size_t& IteratedPower ( ) [inline]
```

Modify the size of the normalized power iterations.

Definition at line 163 of file randomized_svd_method.hpp.

39.196.3.8 MaxIterations() [1/2]

```
size_t MaxIterations ( ) const [inline]
```

Get the number of iterations.

Definition at line 166 of file randomized_svd_method.hpp.

39.196.3.9 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Modify the number of iterations.

Definition at line 168 of file randomized_svd_method.hpp.

39.196.3.10 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialization.

Definition at line 174 of file randomized_svd_method.hpp.

39.196.3.11 W()

```
const arma::mat& W ( ) const [inline]
```

Get the Item Matrix.

Definition at line 156 of file randomized_svd_method.hpp.

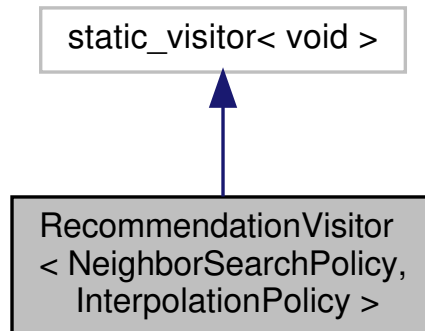
The documentation for this class was generated from the following file:

- [/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/randomized_svd_method.hpp](#) ↩

39.197 RecommendationVisitor< NeighborSearchPolicy, InterpolationPolicy > Class Template Reference

RecommendationVisitor (p. 1084) uses the **CType** (p. 1045) object to get recommendations for the given users.

Inheritance diagram for RecommendationVisitor< NeighborSearchPolicy, InterpolationPolicy >:



Public Member Functions

- **RecommendationVisitor** (const size_t numRecs, arma::Mat< size_t > &recommendations, const arma::Col< size_t > &users, const bool usersGiven)

Visitor constructor.

- template<typename DecompositionPolicy >
void **operator()** (**CType**< DecompositionPolicy > *c) const

Generates the given number of recommendations.

39.197.1 Detailed Description

```
template<typename NeighborSearchPolicy, typename InterpolationPolicy>
class mlpack::cf::RecommendationVisitor< NeighborSearchPolicy, InterpolationPolicy >
```

RecommendationVisitor (p. 1084) uses the **CType** (p. 1045) object to get recommendations for the given users.

Definition at line 83 of file `cf_model.hpp`.

39.197.2 Constructor & Destructor Documentation

39.197.2.1 RecommendationVisitor()

```

RecommendationVisitor (
    const size_t numRecs,
    arma::Mat< size_t > & recommendations,
    const arma::Col< size_t > & users,
    const bool usersGiven )

```

Visitor constructor.

39.197.3 Member Function Documentation

39.197.3.1 operator>()

```

void operator() (
    CFType< DecompositionPolicy > * c ) const

```

Generates the given number of recommendations.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/ **cf_model.hpp**

39.198 RegressionInterpolation Class Reference

Implementation of regression-based interpolation method.

Public Member Functions

- **RegressionInterpolation** ()
Empty Constructor.
- **RegressionInterpolation** (const arma::sp_mat &cleanedData)
Use cleanedData to perform necessary preprocessing.
- template<typename VectorType , typename DecompositionPolicy >
void **GetWeights** (VectorType &&weights, const DecompositionPolicy &decomposition, const size_t queryUser, const arma::Col< size_t > &neighbors, const arma::vec &, const arma::sp_mat &cleanedData)
The regression-based interpolation problem can be solved by a linear system of equations.

39.198.1 Detailed Description

Implementation of regression-based interpolation method.

Predicting a user's rating r_{iu} by it's neighbors' ratings can be regarded as solving linear regression of r_{iu} on r_{iv} , where v are u 's neighbors.

An example of how to use **RegressionInterpolation** (p. 1085) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

CFType<> cf(data);

// Generate 10 recommendations for all users.
cf.template GetRecommendations<
    EuclideanSearch,
    RegressionInterpolation>(10, recommendations);
```

For more information, see the following paper.

```
@inproceedings{bell2007improved,
  title={Improved neighborhood-based collaborative filtering},
  author={Bell, Robert M and Koren, Yehuda},
  booktitle={KDD cup and workshop at the 13th ACM SIGKDD international
    conference on knowledge discovery and data mining},
  pages={7--14},
  year={2007},
  organization={Citeseer}
}
```

Definition at line 56 of file regression_interpolation.hpp.

39.198.2 Constructor & Destructor Documentation

39.198.2.1 RegressionInterpolation() [1/2]

```
RegressionInterpolation ( ) [inline]
```

Empty Constructor.

Definition at line 62 of file regression_interpolation.hpp.

39.198.2.2 RegressionInterpolation() [2/2]

```
RegressionInterpolation (
    const arma::sp_mat & cleanedData ) [inline]
```

Use cleanedData to perform necessary preprocessing.

Parameters

<i>cleanedData</i>	Sparse rating matrix.
--------------------	-----------------------

Definition at line 69 of file regression_interpolation.hpp.

39.198.3 Member Function Documentation

39.198.3.1 GetWeights()

```
void GetWeights (
    VectorType && weights,
    const DecompositionPolicy & decomposition,
    const size_t queryUser,
    const arma::Col< size_t > & neighbors,
    const arma::vec & ,
    const arma::sp_mat & cleanedData ) [inline]
```

The regression-based interpolation problem can be solved by a linear system of equations.

This method first calculates the coefficients and constant terms for the equations and then solve the equations. The solution of the linear system of equations is the resulting interpolation weights (the first parameter). After getting the weights, CF algorithm multiplies each neighbor's rating by its corresponding weight and sums them to get predicted rating.

Parameters

<i>weights</i>	Resulting interpolation weights. The size of weights should be set to the number of neighbors before calling GetWeights() (p. 1087).
<i>decomposition</i>	Decomposition object.
<i>queryUser</i>	Queried user.
<i>neighbors</i>	Neighbors of queried user.
<i>similarities</i>	Similarities between query user and neighbors.
<i>cleanedData</i>	Sparse rating matrix.

Definition at line 95 of file regression_interpolation.hpp.

References Log::Fatal.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/interpolation_policies/**regression_interpolation.**↔
hpp

39.199 RegSVDPolicy Class Reference

Implementation of the Regularized SVD policy to act as a wrapper when accessing Regularized SVD from within **CF**↩
Type (p. 1045).

Public Member Functions

- **RegSVDPolicy** (const size_t maxIterations=10)
Use regularized SVD method to perform collaborative filtering.
- void **Apply** (const arma::mat &data, const arma::sp_mat &, const size_t rank, const size_t maxIterations, const double, const bool)
Apply Collaborative Filtering to the provided data set using the regularized SVD.
- template<typename NeighborSearchPolicy >
void **GetNeighborhood** (const arma::Col< size_t > &users, const size_t numUsersForSimilarity, arma::Mat< size_t > &neighborhood, arma::mat &similarities) const
Get the neighborhood and corresponding similarities for a set of users.
- double **GetRating** (const size_t user, const size_t item) const
Return predicted rating given user ID and item ID.
- void **GetRatingOfUser** (const size_t user, arma::vec &rating) const
Get predicted ratings for a user.
- const arma::mat & **H** () const
Get the User Matrix.
- size_t **MaxIterations** () const
Get the number of iterations.
- size_t & **MaxIterations** ()
Modify the number of iterations.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialization.
- const arma::mat & **W** () const
Get the Item Matrix.

39.199.1 Detailed Description

Implementation of the Regularized SVD policy to act as a wrapper when accessing Regularized SVD from within **CF**↩
Type (p. 1045).

An example of how to use **RegSVDPolicy** (p. 1088) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

CFType<RegSVDPolicy> cf(data);

// Generate 10 recommendations for all users.
cf.GetRecommendations(10, recommendations);
```

Definition at line 41 of file regularized_svd_method.hpp.

39.199.2 Constructor & Destructor Documentation

39.199.2.1 RegSVDPolicy()

```
RegSVDPolicy (
    const size_t maxIterations = 10 ) [inline]
```

Use regularized SVD method to perform collaborative filtering.

Parameters

<i>maxIterations</i>	Number of iterations for the power method (Default: 2).
----------------------	---

Definition at line 50 of file `regularized_svd_method.hpp`.

39.199.3 Member Function Documentation

39.199.3.1 Apply()

```
void Apply (
    const arma::mat & data,
    const arma::sp_mat & ,
    const size_t rank,
    const size_t maxIterations,
    const double ,
    const bool ) [inline]
```

Apply Collaborative Filtering to the provided data set using the regularized SVD.

Parameters

<i>data</i>	Data matrix: dense matrix (coordinate lists) or sparse matrix(cleaned).
<i>cleanedData</i>	item user table in form of sparse matrix.
<i>rank</i>	Rank parameter for matrix factorization.
<i>maxIterations</i>	Maximum number of iterations.
<i>minResidue</i>	Residue required to terminate.
<i>mit</i>	Whether to terminate only when <i>maxIterations</i> is reached.

Definition at line 68 of file `regularized_svd_method.hpp`.

References RegularizedSVD< OptimizerType >::Apply().

39.199.3.2 GetNeighborhood()

```
void GetNeighborhood (
    const arma::Col< size_t > & users,
    const size_t numUsersForSimilarity,
    arma::Mat< size_t > & neighborhood,
    arma::mat & similarities ) const [inline]
```

Get the neighborhood and corresponding similarities for a set of users.

Template Parameters

<i>NeighborSearchPolicy</i>	The policy to perform neighbor search.
-----------------------------	--

Parameters

<i>users</i>	Users whose neighborhood is to be computed.
<i>numUsersForSimilarity</i>	The number of neighbors returned for each user.
<i>neighborhood</i>	Neighbors represented by user IDs.
<i>similarities</i>	Similarity between each user and each of its neighbors.

Definition at line 116 of file regularized_svd_method.hpp.

39.199.3.3 GetRating()

```
double GetRating (
    const size_t user,
    const size_t item ) const [inline]
```

Return predicted rating given user ID and item ID.

Parameters

<i>user</i>	User ID.
<i>item</i>	Item ID.

Definition at line 86 of file regularized_svd_method.hpp.

39.199.3.4 GetRatingOfUser()

```
void GetRatingOfUser (
    const size_t user,
    arma::vec & rating ) const [inline]
```

Get predicted ratings for a user.

Parameters

<i>user</i>	User ID.
<i>rating</i>	Resulting rating vector.

Definition at line 98 of file regularized_svd_method.hpp.

39.199.3.5 H()

```
const arma::mat& H ( ) const [inline]
```

Get the User Matrix.

Definition at line 145 of file regularized_svd_method.hpp.

39.199.3.6 MaxIterations() [1/2]

```
size_t MaxIterations ( ) const [inline]
```

Get the number of iterations.

Definition at line 148 of file regularized_svd_method.hpp.

39.199.3.7 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Modify the number of iterations.

Definition at line 150 of file regularized_svd_method.hpp.

39.199.3.8 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialization.

Definition at line 156 of file regularized_svd_method.hpp.

39.199.3.9 W()

```
const arma::mat& W ( ) const [inline]
```

Get the Item Matrix.

Definition at line 143 of file regularized_svd_method.hpp.

The documentation for this class was generated from the following file:

- [/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/regularized_svd_method.hpp](#) ↩

39.200 SimilarityInterpolation Class Reference

With **SimilarityInterpolation** (p. 1092), interpolation weights are based on similarities between query user and its neighbors.

Public Member Functions

- **SimilarityInterpolation** ()
- **SimilarityInterpolation** (const arma::sp_mat &)

This constructor is needed for interface consistency.
- `template<typename VectorType , typename DecompositionPolicy >`
void GetWeights (VectorType &&weights, const DecompositionPolicy &, const size_t, const arma::Col< size_t > &neighbors, const arma::vec &similarities, const arma::sp_mat &)

Interpolation weights are computed as normalized similarities.

39.200.1 Detailed Description

With **SimilarityInterpolation** (p. 1092), interpolation weights are based on similarities between query user and its neighbors.

All interpolation weights sum up to one.

An example of how to use **SimilarityInterpolation** (p. 1092) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

CType<> cf(data);

// Generate 10 recommendations for all users.
cf.template GetRecommendations<
    EuclideanSearch,
    SimilarityInterpolation>(10, recommendations);
```

Definition at line 41 of file similarity_interpolation.hpp.

39.200.2 Constructor & Destructor Documentation

39.200.2.1 SimilarityInterpolation() [1/2]

```
SimilarityInterpolation ( ) [inline]
```

Definition at line 45 of file similarity_interpolation.hpp.

39.200.2.2 SimilarityInterpolation() [2/2]

```
SimilarityInterpolation (
    const arma::sp_mat & ) [inline]
```

This constructor is needed for interface consistency.

Definition at line 50 of file similarity_interpolation.hpp.

39.200.3 Member Function Documentation

39.200.3.1 GetWeights()

```
void GetWeights (
    VectorType && weights,
    const DecompositionPolicy & ,
    const size_t ,
    const arma::Col< size_t > & neighbors,
    const arma::vec & similarities,
    const arma::sp_mat & ) [inline]
```

Interpolation weights are computed as normalized similarities.

After getting the weights, CF algorithm multiplies each neighbor's rating by its corresponding weight and sums them to get predicted rating.

Parameters

<i>weights</i>	Resulting interpolation weights. The size of weights should be set to the number of neighbors before calling GetWeights() (p. 1093).
<i>decomposition</i>	Decomposition object.
<i>queryUser</i>	Queried user.
<i>neighbors</i>	Neighbors of queried user.
<i>similarities</i>	Similarities between query user and neighbors.
<i>cleanedData</i>	Sparse rating matrix.

Definition at line 67 of file similarity_interpolation.hpp.

References Log::Fatal.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/interpolation_policies/ **similarity_interpolation.**↩
hpp

39.201 SVDCompletePolicy Class Reference

Implementation of the SVD complete incremental policy to act as a wrapper when accessing SVD complete decomposition from within **CFTYPE** (p. 1045).

Public Member Functions

- template<typename MatType >
void **Apply** (const MatType &, const arma::sp_mat &cleanedData, const size_t rank, const size_t maxIterations, const double minResidue, const bool mit)
Apply Collaborative Filtering to the provided data set using the SVD complete incremental policy.

- `template<typename NeighborSearchPolicy >`
`void GetNeighborhood (const arma::Col< size_t > &users, const size_t numUsersForSimilarity, arma::Mat< size_t > &neighborhood, arma::mat &similarities) const`
Get the neighborhood and corresponding similarities for a set of users.
- `double GetRating (const size_t user, const size_t item) const`
Return predicted rating given user ID and item ID.
- `void GetRatingOfUser (const size_t user, arma::vec &rating) const`
Get predicted ratings for a user.
- `const arma::mat & H () const`
Get the User Matrix.
- `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`
Serialization.
- `const arma::mat & W () const`
Get the Item Matrix.

39.201.1 Detailed Description

Implementation of the SVD complete incremental policy to act as a wrapper when accessing SVD complete decomposition from within **CFTYPE** (p. 1045).

An example of how to use **SVDCompletePolicy** (p. 1094) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

CFTYPE<SVDCompletePolicy> cf(data);

// Generate 10 recommendations for all users.
cf.GetRecommendations(10, recommendations);
```

Definition at line 44 of file `svd_complete_method.hpp`.

39.201.2 Member Function Documentation

39.201.2.1 Apply()

```
void Apply (
    const MatType & ,
    const arma::sp_mat & cleanedData,
    const size_t rank,
    const size_t maxIterations,
    const double minResidue,
    const bool mit ) [inline]
```

Apply Collaborative Filtering to the provided data set using the SVD complete incremental policy.

Parameters

<i>data</i>	Data matrix: dense matrix (coordinate lists) or sparse matrix(cleaned).
<i>cleanedData</i>	item user table in form of sparse matrix.
<i>rank</i>	Rank parameter for matrix factorization.
<i>maxIterations</i>	Maximum number of iterations.
<i>minResidue</i>	Residue required to terminate.
<i>mit</i>	Whether to terminate only when maxIterations is reached.

Definition at line 60 of file `svd_complete_method.hpp`.

References `AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::Apply()`.

39.201.2.2 GetNeighborhood()

```
void GetNeighborhood (
    const arma::Col< size_t > & users,
    const size_t numUsersForSimilarity,
    arma::Mat< size_t > & neighborhood,
    arma::mat & similarities ) const [inline]
```

Get the neighborhood and corresponding similarities for a set of users.

Template Parameters

<i>NeighborSearchPolicy</i>	The policy to perform neighbor search.
-----------------------------	--

Parameters

<i>users</i>	Users whose neighborhood is to be computed.
<i>numUsersForSimilarity</i>	The number of neighbors returned for each user.
<i>neighborhood</i>	Neighbors represented by user IDs.
<i>similarities</i>	Similarity between each user and each of its neighbors.

Definition at line 126 of file `svd_complete_method.hpp`.

39.201.2.3 GetRating()

```
double GetRating (
    const size_t user,
    const size_t item ) const [inline]
```

Return predicted rating given user ID and item ID.

Parameters

<i>user</i>	User ID.
<i>item</i>	Item ID.

Definition at line 96 of file `svd_complete_method.hpp`.

39.201.2.4 GetRatingOfUser()

```
void GetRatingOfUser (
    const size_t user,
    arma::vec & rating ) const [inline]
```

Get predicted ratings for a user.

Parameters

<i>user</i>	User ID.
<i>rating</i>	Resulting rating vector.

Definition at line 108 of file `svd_complete_method.hpp`.

39.201.2.5 H()

```
const arma::mat& H ( ) const [inline]
```

Get the User Matrix.

Definition at line 155 of file `svd_complete_method.hpp`.

39.201.2.6 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialization.

Definition at line 161 of file `svd_complete_method.hpp`.

39.201.2.7 W()

```
const arma::mat& W ( ) const [inline]
```

Get the Item Matrix.

Definition at line 153 of file svd_complete_method.hpp.

The documentation for this class was generated from the following file:

- [/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/method.hpp](#) **svd_complete_↔**

39.202 SVDIncompletePolicy Class Reference

Implementation of the SVD incomplete incremental to act as a wrapper when accessing SVD incomplete incremental from within **CType** (p. 1045).

Public Member Functions

- `template<typename MatType >`
`void Apply (const MatType &, const arma::sp_mat &cleanedData, const size_t rank, const size_t maxIterations, const double minResidue, const bool mit)`
Apply Collaborative Filtering to the provided data set using the SVD incomplete incremental method.
- `template<typename NeighborSearchPolicy >`
`void GetNeighborhood (const arma::Col< size_t > &users, const size_t numUsersForSimilarity, arma::Mat< size_t > &neighborhood, arma::mat &similarities) const`
Get the neighborhood and corresponding similarities for a set of users.
- `double GetRating (const size_t user, const size_t item) const`
Return predicted rating given user ID and item ID.
- `void GetRatingOfUser (const size_t user, arma::vec &rating) const`
Get predicted ratings for a user.
- `const arma::mat & H () const`
Get the User Matrix.
- `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`
Serialization.
- `const arma::mat & W () const`
Get the Item Matrix.

39.202.1 Detailed Description

Implementation of the SVD incomplete incremental to act as a wrapper when accessing SVD incomplete incremental from within **CType** (p. 1045).

An example of how to use **SVDIncompletePolicy** (p. 1098) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

CType<SVDIncompletePolicy> cf(data);

// Generate 10 recommendations for all users.
cf.GetRecommendations(10, recommendations);
```

Definition at line 44 of file `svd_incomplete_method.hpp`.

39.202.2 Member Function Documentation

39.202.2.1 Apply()

```
void Apply (
    const MatType & ,
    const arma::sp_mat & cleanedData,
    const size_t rank,
    const size_t maxIterations,
    const double minResidue,
    const bool mit ) [inline]
```

Apply Collaborative Filtering to the provided data set using the SVD incomplete incremental method.

Parameters

<i>data</i>	Data matrix: dense matrix (coordinate lists) or sparse matrix(cleaned).
<i>cleanedData</i>	item user table in form of sparse matrix.
<i>rank</i>	Rank parameter for matrix factorization.
<i>maxIterations</i>	Maximum number of iterations.
<i>minResidue</i>	Residue required to terminate.
<i>mit</i>	Whether to terminate only when maxIterations is reached.

Definition at line 60 of file `svd_incomplete_method.hpp`.

References `AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >::Apply()`.

39.202.2.2 GetNeighborhood()

```
void GetNeighborhood (
    const arma::Col< size_t > & users,
    const size_t numUsersForSimilarity,
    arma::Mat< size_t > & neighborhood,
    arma::mat & similarities ) const [inline]
```

Get the neighborhood and corresponding similarities for a set of users.

Template Parameters

<i>NeighborSearchPolicy</i>	The policy to perform neighbor search.
-----------------------------	--

Parameters

<i>users</i>	Users whose neighborhood is to be computed.
<i>numUsersForSimilarity</i>	The number of neighbors returned for each user.
<i>neighborhood</i>	Neighbors represented by user IDs.
<i>similarities</i>	Similarity between each user and each of its neighbors.

Definition at line 125 of file `svd_incomplete_method.hpp`.

39.202.2.3 GetRating()

```
double GetRating (
    const size_t user,
    const size_t item ) const [inline]
```

Return predicted rating given user ID and item ID.

Parameters

<i>user</i>	User ID.
<i>item</i>	Item ID.

Definition at line 95 of file `svd_incomplete_method.hpp`.

39.202.2.4 GetRatingOfUser()

```
void GetRatingOfUser (
    const size_t user,
    arma::vec & rating ) const [inline]
```

Get predicted ratings for a user.

Parameters

<i>user</i>	User ID.
<i>rating</i>	Resulting rating vector.

Definition at line 107 of file `svd_incomplete_method.hpp`.

39.202.2.5 `H()`

```
const arma::mat& H ( ) const [inline]
```

Get the User Matrix.

Definition at line 154 of file `svd_incomplete_method.hpp`.

39.202.2.6 `serialize()`

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialization.

Definition at line 160 of file `svd_incomplete_method.hpp`.

39.202.2.7 `W()`

```
const arma::mat& W ( ) const [inline]
```

Get the Item Matrix.

Definition at line 152 of file `svd_incomplete_method.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/svd_incomplete_method.hpp` ↩

39.203 SVDPlusPlusPolicy Class Reference

Implementation of the SVDPlusPlus policy to act as a wrapper when accessing SVDPlusPlus from within **CType** (p. 1045).

Public Member Functions

- **SVDPlusPlusPolicy** (const size_t maxIterations=10, const double alpha=0.001, const double lambda=0.1)
Use SVDPlusPlus method to perform collaborative filtering.
- double **Alpha** () const
Get learning rate.
- double & **Alpha** ()
Modify learning rate.
- void **Apply** (const arma::mat &data, const arma::sp_mat &, const size_t rank, const size_t maxIterations, const double, const bool)
Apply Collaborative Filtering to the provided data set using the svdplusplus.
- template<typename NeighborSearchPolicy >
void **GetNeighborhood** (const arma::Col< size_t > &users, const size_t numUsersForSimilarity, arma::Mat< size_t > &neighborhood, arma::mat &similarities) const
Get the neighborhood and corresponding similarities for a set of users.
- double **GetRating** (const size_t user, const size_t item) const
Return predicted rating given user ID and item ID.
- void **GetRatingOfUser** (const size_t user, arma::vec &rating) const
Get predicted ratings for a user.
- const arma::mat & **H** () const
Get the User Matrix.
- const arma::sp_mat & **ImplicitData** () const
Get Implicit Feedback Data.
- double **Lambda** () const
Get regularization parameter.
- double & **Lambda** ()
Modify regularization parameter.
- size_t **MaxIterations** () const
Get the number of iterations.
- size_t & **MaxIterations** ()
Modify the number of iterations.
- const arma::vec & **P** () const
Get the Item Bias Vector.
- const arma::vec & **Q** () const
Get the User Bias Vector.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialization.
- const arma::mat & **W** () const
Get the Item Matrix.
- const arma::mat & **Y** () const
Get the Item Implicit Matrix.

39.203.1 Detailed Description

Implementation of the SVDPlusPlus policy to act as a wrapper when accessing SVDPlusPlus from within **CType** (p. 1045).

An example of how to use **SVDPlusPlusPolicy** (p. 1102) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

CType<SVDPlusPlusPolicy> cf(data);

// Generate 10 recommendations for all users.
cf.GetRecommendations(10, recommendations);
```

Definition at line 41 of file svdplusplus_method.hpp.

39.203.2 Constructor & Destructor Documentation

39.203.2.1 SVDPlusPlusPolicy()

```
SVDPlusPlusPolicy (
    const size_t maxIterations = 10,
    const double alpha = 0.001,
    const double lambda = 0.1 ) [inline]
```

Use SVDPlusPlus method to perform collaborative filtering.

Parameters

<i>maxIterations</i>	Number of iterations.
<i>alpha</i>	Learning rate for optimization.
<i>Regularization</i>	parameter for optimization.

Definition at line 51 of file svdplusplus_method.hpp.

39.203.3 Member Function Documentation

39.203.3.1 Alpha() [1/2]

```
double Alpha ( ) const [inline]
```

Get learning rate.

Definition at line 193 of file svdplusplus_method.hpp.

39.203.3.2 Alpha() [2/2]

```
double& Alpha ( ) [inline]
```

Modify learning rate.

Definition at line 195 of file svdplusplus_method.hpp.

39.203.3.3 Apply()

```
void Apply (
    const arma::mat & data,
    const arma::sp_mat & ,
    const size_t rank,
    const size_t maxIterations,
    const double ,
    const bool ) [inline]
```

Apply Collaborative Filtering to the provided data set using the svdplusplus.

Parameters

<i>data</i>	Data matrix: dense matrix (coordinate lists) or sparse matrix(cleaned).
<i>cleanedData</i>	item user table in form of sparse matrix.
<i>rank</i>	Rank parameter for matrix factorization.
<i>maxIterations</i>	Maximum number of iterations.
<i>minResidue</i>	Residue required to terminate.
<i>mit</i>	Whether to terminate only when maxIterations is reached.

Definition at line 73 of file svdplusplus_method.hpp.

References SVDPlusPlus< OptimizerType >::Apply(), and SVDPlusPlus< OptimizerType >::CleanData().

39.203.3.4 GetNeighborhood()

```
void GetNeighborhood (
    const arma::Col< size_t > & users,
    const size_t numUsersForSimilarity,
    arma::Mat< size_t > & neighborhood,
    arma::mat & similarities ) const [inline]
```

Get the neighborhood and corresponding similarities for a set of users.

Template Parameters

<i>NeighborSearchPolicy</i>	The policy to perform neighbor search.
-----------------------------	--

Parameters

<i>users</i>	Users whose neighborhood is to be computed.
<i>numUsersForSimilarity</i>	The number of neighbors returned for each user.
<i>neighborhood</i>	Neighbors represented by user IDs.
<i>similarities</i>	Similarity between each user and each of its neighbors.

Definition at line 157 of file svdplusplus_method.hpp.

39.203.3.5 GetRating()

```
double GetRating (
    const size_t user,
    const size_t item ) const [inline]
```

Return predicted rating given user ID and item ID.

Parameters

<i>user</i>	User ID.
<i>item</i>	Item ID.

Definition at line 96 of file svdplusplus_method.hpp.

39.203.3.6 GetRatingOfUser()

```
void GetRatingOfUser (
    const size_t user,
    arma::vec & rating ) const [inline]
```

Get predicted ratings for a user.

Parameters

<i>user</i>	User ID.
<i>rating</i>	Resulting rating vector.

Definition at line 124 of file svdplusplus_method.hpp.

39.203.3.7 H()

```
const arma::mat& H ( ) const [inline]
```

Get the User Matrix.

Definition at line 177 of file svdplusplus_method.hpp.

39.203.3.8 ImplicitData()

```
const arma::sp_mat& ImplicitData ( ) const [inline]
```

Get Implicit Feedback Data.

Definition at line 185 of file svdplusplus_method.hpp.

39.203.3.9 Lambda() [1/2]

```
double Lambda ( ) const [inline]
```

Get regularization parameter.

Definition at line 198 of file svdplusplus_method.hpp.

39.203.3.10 Lambda() [2/2]

```
double& Lambda ( ) [inline]
```

Modify regularization parameter.

Definition at line 200 of file svdplusplus_method.hpp.

39.203.3.11 MaxIterations() [1/2]

```
size_t MaxIterations ( ) const [inline]
```

Get the number of iterations.

Definition at line 188 of file svdplusplus_method.hpp.

39.203.3.12 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Modify the number of iterations.

Definition at line 190 of file svdplusplus_method.hpp.

39.203.3.13 P()

```
const arma::vec& P ( ) const [inline]
```

Get the Item Bias Vector.

Definition at line 181 of file svdplusplus_method.hpp.

39.203.3.14 Q()

```
const arma::vec& Q ( ) const [inline]
```

Get the User Bias Vector.

Definition at line 179 of file svdplusplus_method.hpp.

39.203.3.15 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialization.

Definition at line 206 of file svdplusplus_method.hpp.

39.203.3.16 `W()`

```
const arma::mat& W ( ) const [inline]
```

Get the Item Matrix.

Definition at line 175 of file `svdplusplus_method.hpp`.

39.203.3.17 `Y()`

```
const arma::mat& Y ( ) const [inline]
```

Get the Item Implicit Matrix.

Definition at line 183 of file `svdplusplus_method.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/ svdplusplus_method.h`↩
`hpp`

39.204 SVDWrapper< Factorizer > Class Template Reference

This class acts as the wrapper for all SVD factorizers which are incompatible with CF module.

Public Member Functions

- **SVDWrapper** (const Factorizer &factorizer=Factorizer())
- double **Apply** (const arma::mat &V, arma::mat &W, arma::mat &sigma, arma::mat &H) const
Factorizer function which takes SVD of the given matrix and returns the frobenius norm of error.
- double **Apply** (const arma::mat &V, size_t r, arma::mat &W, arma::mat &H) const
Factorizer function which computes SVD and returns matrices as required by CF module.

39.204.1 Detailed Description

```
template<class Factorizer = DummyClass>
class mlpack::cf::SVDWrapper< Factorizer >
```

This class acts as the wrapper for all SVD factorizers which are incompatible with CF module.

Normally SVD factorizers implement Apply method which takes matrix V and factorizes it into P, sigma and Q where $V = P * \text{sigma} * \text{trans}(Q)$. But CF module requires factorization to be $V = W * H$. This class multiplies P and sigma and takes the first 'r' eigenvectors out where 'r' is the rank of factorization. Q matrix is transposed and trimmed to support the rank of factorization. The Factorizer class should implement Apply which takes matrices P, sigma, Q and V as their parameter respectively.

Definition at line 40 of file `svd_wrapper.hpp`.

39.204.2 Constructor & Destructor Documentation

39.204.2.1 SVDWrapper()

```
SVDWrapper (
    const Factorizer & factorizer = Factorizer() ) [inline]
```

Definition at line 44 of file svd_wrapper.hpp.

39.204.3 Member Function Documentation

39.204.3.1 Apply() [1/2]

```
double Apply (
    const arma::mat & V,
    arma::mat & W,
    arma::mat & sigma,
    arma::mat & H ) const
```

Factorizer function which takes SVD of the given matrix and returns the frobenius norm of error.

Parameters

<i>V</i>	input matrix
<i>W</i>	first unitary matrix
<i>sigma</i>	eigenvalue matrix
<i>H</i>	second unitary matrix

Note

$$V = W * \text{sigma} * \text{arma::trans}(H)$$

39.204.3.2 Apply() [2/2]

```
double Apply (
    const arma::mat & V,
    size_t r,
```

```
arma::mat & W,  
arma::mat & H ) const
```

Factorizer function which computes SVD and returns matrices as required by CF module.

Parameters

V	input matrix
W	first unitary matrix
H	second unitary matrix

Note

$$V = W * H$$

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/ **svd_wrapper.hpp**

39.205 UserMeanNormalization Class Reference

This normalization class performs user mean normalization on raw ratings.

Public Member Functions

- **UserMeanNormalization** ()
- double **Denormalize** (const size_t user, const size_t, const double rating) const
Denormalize computed rating by adding user mean.
- void **Denormalize** (const arma::Mat< size_t > &combinations, arma::vec &predictions) const
Denormalize computed rating by adding user mean.
- const arma::vec & **Mean** () const
Return user mean.
- void **Normalize** (arma::mat &data)
Normalize the data by subtracting user mean from each of existing ratings.
- void **Normalize** (arma::sp_mat &cleanedData)
Normalize the data by subtracting user mean from each of existing rating.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialization.

39.205.1 Detailed Description

This normalization class performs user mean normalization on raw ratings.

An example of how to use **UserMeanNormalization** (p. 1111) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

// Use UserMeanNormalization as normalization method.
CFType<NMFPolicy, UserMeanNormalization> cf(data);

// Generate 10 recommendations for all users.
cf.GetRecommendations(10, recommendations);
```

Definition at line 39 of file user_mean_normalization.hpp.

39.205.2 Constructor & Destructor Documentation

39.205.2.1 UserMeanNormalization()

UserMeanNormalization () [inline]

Definition at line 43 of file user_mean_normalization.hpp.

39.205.3 Member Function Documentation

39.205.3.1 Denormalize() [1/2]

```
double Denormalize (
    const size_t user,
    const size_t ,
    const double rating ) const [inline]
```

Denormalize computed rating by adding user mean.

Parameters

<i>user</i>	User ID.
<i>item</i>	Item ID.
<i>rating</i>	Computed rating before denormalization.

Definition at line 127 of file user_mean_normalization.hpp.

39.205.3.2 Denormalize() [2/2]

```
void Denormalize (
    const arma::Mat< size_t > & combinations,
    arma::vec & predictions ) const [inline]
```

Denormalize computed rating by adding user mean.

Parameters

<i>combinations</i>	User/Item combinations.
<i>predictions</i>	Predicted ratings for each user/item combination.

Definition at line 140 of file `user_mean_normalization.hpp`.

39.205.3.3 Mean()

```
const arma::vec& Mean ( ) const [inline]
```

Return user mean.

Definition at line 153 of file `user_mean_normalization.hpp`.

39.205.3.4 Normalize() [1/2]

```
void Normalize (
    arma::mat & data ) [inline]
```

Normalize the data by subtracting user mean from each of existing ratings.

Parameters

<i>data</i>	Input dataset in the form of coordinate list.
-------------	---

Definition at line 50 of file `user_mean_normalization.hpp`.

39.205.3.5 Normalize() [2/2]

```
void Normalize (
    arma::sp_mat & cleanedData ) [inline]
```

Normalize the data by subtracting user mean from each of existing rating.

Parameters

<i>cleanedData</i>	Input data as a sparse matrix.
--------------------	--------------------------------

Definition at line 90 of file `user_mean_normalization.hpp`.

39.205.3.6 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialization.

Definition at line 159 of file `user_mean_normalization.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/ user_mean_normalization.hpp`

39.206 ZScoreNormalization Class Reference

This normalization class performs z-score normalization on raw ratings.

Public Member Functions

- **ZScoreNormalization** ()
- double **Denormalize** (const size_t, const size_t, const double rating) const
Denormalize computed rating by adding mean and multiplying stddev.
- void **Denormalize** (const arma::Mat< size_t > &, arma::vec &predictions) const
Denormalize computed rating by adding mean and multiplying stddev.
- double **Mean** () const
Return mean.
- void **Normalize** (arma::mat &data)
Normalize the data to zero mean and one standard deviation.
- void **Normalize** (arma::sp_mat &cleanedData)
Normalize the data to zero mean and one standard deviation.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialization.
- double **Stddev** () const
Return stddev.

39.206.1 Detailed Description

This normalization class performs z-score normalization on raw ratings.

An example of how to use **ZScoreNormalization** (p. 1114) in CF is shown below:

```
extern arma::mat data; // data is a (user, item, rating) table.
// Users for whom recommendations are generated.
extern arma::Col<size_t> users;
arma::Mat<size_t> recommendations; // Resulting recommendations.

// Use ZScoreNormalization as normalization method.
CFType<NMFPolicy, ZScoreNormalization> cf(data);

// Generate 10 recommendations for all users.
cf.GetRecommendations(10, recommendations);
```

Definition at line 38 of file `z_score_normalization.hpp`.

39.206.2 Constructor & Destructor Documentation

39.206.2.1 ZScoreNormalization()

```
ZScoreNormalization ( ) [inline]
```

Definition at line 42 of file `z_score_normalization.hpp`.

39.206.3 Member Function Documentation

39.206.3.1 Denormalize() [1/2]

```
double Denormalize (
    const size_t ,
    const size_t ,
    const double rating ) const [inline]
```

Denormalize computed rating by adding mean and multiplying stddev.

Parameters

<i>user</i>	User ID.
<i>item</i>	Item ID.
<i>rating</i>	Computed rating before denormalization.

Definition at line 110 of file `z_score_normalization.hpp`.

39.206.3.2 Denormalize() [2/2]

```
void Denormalize (
    const arma::Mat< size_t > & ,
    arma::vec & predictions ) const [inline]
```

Denormalize computed rating by adding mean and multiplying stddev.

Parameters

<i>combinations</i>	User/Item combinations.
<i>predictions</i>	Predicted ratings for each user/item combination.

Definition at line 123 of file `z_score_normalization.hpp`.

39.206.3.3 `Mean()`

```
double Mean ( ) const [inline]
```

Return mean.

Definition at line 132 of file `z_score_normalization.hpp`.

39.206.3.4 `Normalize()` [1/2]

```
void Normalize (
    arma::mat & data ) [inline]
```

Normalize the data to zero mean and one standard deviation.

Parameters

<i>data</i>	Input dataset in the form of coordinate list.
-------------	---

Definition at line 49 of file `z_score_normalization.hpp`.

References `Log::Fatal`.

39.206.3.5 `Normalize()` [2/2]

```
void Normalize (
    arma::sp_mat & cleanedData ) [inline]
```

Normalize the data to zero mean and one standard deviation.

Parameters

<i>cleanedData</i>	Input data as a sparse matrix.
--------------------	--------------------------------

Definition at line 76 of file `z_score_normalization.hpp`.

References `Log::Fatal`.

39.206.3.6 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialization.

Definition at line 149 of file `z_score_normalization.hpp`.

39.206.3.7 Stddev()

```
double Stddev ( ) const [inline]
```

Return stddev.

Definition at line 140 of file `z_score_normalization.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/ z_score_normalization.hpp`

39.207 CLI Class Reference

Parses the command line for parameters and holds user-specified parameters.

Public Types

- `typedef std::map< std::string, std::map< std::string, void(*) (const util::ParamData &, const void *, void *)> > >`
FunctionMapType
Map for functions and types.

Static Public Member Functions

- static void **Add** (**util::ParamData** &&d)
Adds a parameter to the hierarchy; use the `PARAM_` macros instead of this (i.e.*
- static std::map< char, std::string > & **Aliases** ()
*Return a modifiable list of aliases that **CLI** (p. 1117) knows about.*
- static void **ClearSettings** ()
Clear all of the settings, removing all parameters and function mappings.
- template<typename T >
static T & **GetParam** (const std::string &identifier)
Get the value of type T found while parsing.
- template<typename T >
static std::string **GetPrintableParam** (const std::string &identifier)
Cast the given parameter of the given type to a short, printable std::string, for use in status messages.
- template<typename T >
static T & **GetRawParam** (const std::string &identifier)
*Get the raw value of the parameter before any processing that **GetParam()** (p. 1122) might normally do.*
- static **CLI** & **GetSingleton** ()
Retrieve the singleton.
- static bool **HasParam** (const std::string &identifier)
See if the specified flag was found while parsing.
- static std::map< std::string, **util::ParamData** > & **Parameters** ()
*Return a modifiable list of parameters that **CLI** (p. 1117) knows about.*
- static std::string **ProgramName** ()
*Get the program name as set by the **PROGRAM_INFO()** (p. 2733) macro.*
- static void **RegisterProgramDoc** (**util::ProgramDoc** * doc)
Registers a ProgramDoc object, which contains documentation about the program.
- static void **RestoreSettings** (const std::string &name, const bool fatal=true)
Restore all of the parameters and function mappings of the given name, if they exist.
- static void **SetPassed** (const std::string &name)
Mark a particular parameter as passed.
- static void **StoreSettings** (const std::string &name)
Take all parameters and function mappings and store them, under the given name.

Public Attributes

- bool **didParse**
*True, if **CLI** (p. 1117) was used to parse command line options.*
- **util::ProgramDoc** * **doc**
Pointer to the ProgramDoc object.
- **FunctionMapType** **functionMap**
- std::string **programName**
Holds the name of the program for `--version`.
- **Timers** **timer**
Holds the timer objects.

39.207.1 Detailed Description

Parses the command line for parameters and holds user-specified parameters.

The **CLI** (p. 1117) class is a subsystem by which parameters for machine learning methods can be specified and accessed. In conjunction with the macros `PARAM_DOUBLE`, `PARAM_INT`, `PARAM_STRING`, `PARAM_FLAG`, and others, this class aims to make user configurability of mlpack methods very easy. There are only three methods in **CLI** (p. 1117) that a user should need: **CLI::ParseCommandLine()** (p. 302), **CLI::GetParam()** (p. 1122), and **CLI::HasParam()** (p. 1123) (in addition to the `PARAM_*`() macros).

39.207.2 Adding parameters to a program

```
$ ./executable --bar=5
```

Note

The `=` is optional; a space can also be used.

A parameter is specified by using one of the following macros (this is not a complete list; see `core/io/cli.hpp`):

- **PARAM_FLAG**(ID, DESC, ALIAS) (p. 2710)
- **PARAM_DOUBLE**(ID, DESC, ALIAS, DEF)
- **PARAM_INT**(ID, DESC, ALIAS, DEF)
- **PARAM_STRING**(ID, DESC, ALIAS, DEF)

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Short description of the parameter (one/two sentences).
<i>ALIAS</i>	An alias for the parameter.
<i>DEF</i>	Default value of the parameter.

The flag (boolean) type automatically defaults to false; it is specified merely as a flag on the command line (no `'=true'` is required).

Here is an example of a few parameters being defined; this is for the KNN executable (`methods/neighbor_search/knn_main.cpp`):

```
PARAM_STRING_REQ("reference_file", "File containing the reference dataset.",
    "r");
PARAM_STRING_REQ("distances_file", "File to output distances into.", "d");
PARAM_STRING_REQ("neighbors_file", "File to output neighbors into.", "n");
PARAM_INT_REQ("k", "Number of furthest neighbors to find.", "k");
PARAM_STRING("query_file", "File containing query points (optional).", "q",
    "");
PARAM_INT("leaf_size", "Leaf size for tree building.", "l", 20);
```

```
PARAM_FLAG("naive", "If true, O(n^2) naive mode is used for computation.",
           "N");
PARAM_FLAG("single_mode", "If true, single-tree search is used (as opposed "
           "to dual-tree search.", "s");
```

More documentation is available on the `PARAM_*`() macros in the documentation for `core/io/cli.hpp`.

39.207.3 Documenting the program itself

In addition to allowing documentation for each individual parameter and module, the **PROGRAM_INFO()** (p. 2733) macro provides support for documenting the program itself. There should only be one instance of the **PROGRAM_INFO()** (p. 2733) macro. Below is an example:

```
PROGRAM_INFO("Maximum Variance Unfolding", "This program performs maximum "
           "variance unfolding on the given dataset, writing a lower-dimensional "
           "unfolded dataset to the given output file.");
```

This description should be verbose, and explain to a non-expert user what the program does and how to use it. If relevant, paper citations should be included.

39.207.4 Parsing the command line with CLI

To have **CLI** (p. 1117) parse the command line at the beginning of code execution, only a call to **ParseCommandLine()** (p. 302) is necessary:

```
int main(int argc, char** argv)
{
    CLI::ParseCommandLine(argc, argv);

    ...
}
```

CLI (p. 1117) provides `–help` and `–info` options which give nicely formatted documentation of each option; the documentation is generated from the `DESC` arguments in the `PARAM_*`() macros.

39.207.5 Getting parameters with CLI

When the parameters have been defined, the next important thing is how to access them. For this, the **HasParam()** (p. 1123) and **GetParam()** (p. 1122) methods are used. For instance, to see if the user passed the flag (boolean) "naive":

```
if (CLI::HasParam("naive"))
{
    Log::Info << "Naive has been passed!" << std::endl;
}
```

To get the value of a parameter, such as a string, use **GetParam**:

```
const std::string filename = CLI::GetParam<std::string>("filename");
```


Note

Options should only be defined in files which define `main()` (that is, main executables). If options are defined elsewhere, they may be spuriously included into other executables and confuse users. Similarly, if your executable has options which you did not define, it is probably because the option is defined somewhere else and included in your executable.

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 166 of file `cli.hpp`.

39.207.6 Member Typedef Documentation

39.207.6.1 FunctionMapType

```
typedef std::map<std::string, std::map<std::string, void (*)(const util::ParamData&, const void*, void*)> > FunctionMapType
```

Map for functions and types.

Use as `functionMap["typename"]["functionName"]`.

Definition at line 289 of file `cli.hpp`.

39.207.7 Member Function Documentation

39.207.7.1 Add()

```
static void Add (  
    util::ParamData && d ) [static]
```

Adds a parameter to the hierarchy; use the `PARAM_*`() macros instead of this (i.e.

`PARAM_INT()`).

Parameters

<i>d</i>	Utility structure holding parameter data.
----------	---

Referenced by `CLIOption< N >::CLIOption()`, `MDOption< T >::MDOption()`, `PyOption< T >::PyOption()`, and `TestOption< N >::TestOption()`.

39.207.7.2 Aliases()

```
static std::map<char, std::string>& Aliases ( ) [static]
```

Return a modifiable list of aliases that **CLI** (p. 1117) knows about.

39.207.7.3 ClearSettings()

```
static void ClearSettings ( ) [static]
```

Clear all of the settings, removing all parameters and function mappings.

Referenced by `MDOption< T >::MDOption()`, `PyOption< T >::PyOption()`, and `TestOption< N >::TestOption()`.

39.207.7.4 GetParam()

```
static T& GetParam (
    const std::string & identifier ) [static]
```

Get the value of type `T` found while parsing.

You can set the value using this reference safely.

Parameters

<i>identifier</i>	The name of the parameter in question.
-------------------	--

39.207.7.5 GetPrintableParam()

```
static std::string GetPrintableParam (
```

```
const std::string & identifier ) [static]
```

Cast the given parameter of the given type to a short, printable `std::string`, for use in status messages.

Ideally the message returned here should be only a handful of characters, and certainly no longer than one line.

Parameters

<i>identifier</i>	The name of the parameter in question.
-------------------	--

39.207.7.6 GetRawParam()

```
static T& GetRawParam (
    const std::string & identifier ) [static]
```

Get the raw value of the parameter before any processing that **GetParam()** (p. 1122) might normally do.

So, e.g., for command-line programs, this does not perform any data loading or manipulation like **GetParam()** (p. 1122) does. So if you want to access a matrix or model (or similar) parameter before it is loaded, this is the method to use.

Parameters

<i>identifier</i>	The name of the parameter in question.
-------------------	--

39.207.7.7 GetSingleton()

```
static CLI& GetSingleton ( ) [static]
```

Retrieve the singleton.

As an end user, if you are just using the **CLI** (p. 1117) object, you should not need to use this function—the other static functions should be sufficient.

In this case, the singleton is used to store data for the static methods, as there is no point in defining static methods only to have users call private instance methods.

Returns

The singleton instance for use in the static methods.

Referenced by `CLIOption< N >::CLIOption()`, `mlpack::bindings::cli::EndProgram()`, `MDOption< T >::MDOption()`, `mlpack::bindings::cli::ParseCommandLine()`, `PyOption< T >::PyOption()`, `mlpack::util::ResetTimers()`, and `TestOption< N >::TestOption()`.

39.207.7.8 HasParam()

```
static bool HasParam (
    const std::string & identifier ) [static]
```

See if the specified flag was found while parsing.

Parameters

<i>identifier</i>	The name of the parameter in question.
-------------------	--

Referenced by `mlpack::bindings::cli::EndProgram()`, and `mlpack::bindings::cli::ParseCommandLine()`.

39.207.7.9 Parameters()

```
static std::map<std::string, util::ParamData>& Parameters ( ) [static]
```

Return a modifiable list of parameters that **CLI** (p. 1117) knows about.

Referenced by `CLIOption< N >::CLIOption()`, `mlpack::bindings::cli::EndProgram()`, `mlpack::bindings::cli::ParseCommandLine()`, and `mlpack::bindings::python::PrintOutputProcessing()`.

39.207.7.10 ProgramName()

```
static std::string ProgramName ( ) [static]
```

Get the program name as set by the **PROGRAM_INFO()** (p. 2733) macro.

Referenced by `mlpack::bindings::cli::ParseCommandLine()`.

39.207.7.11 RegisterProgramDoc()

```
static void RegisterProgramDoc (
    util::ProgramDoc * doc ) [static]
```

Registers a `ProgramDoc` object, which contains documentation about the program.

If this method has been called before (that is, if two `ProgramDocs` are instantiated in the program), a fatal error will occur.

Parameters

<i>doc</i>	Pointer to the ProgramDoc object.
------------	-----------------------------------

39.207.7.12 RestoreSettings()

```
static void RestoreSettings (
    const std::string & name,
    const bool fatal = true ) [static]
```

Restore all of the parameters and function mappings of the given name, if they exist.

A `std::invalid_argument` exception will be thrown if `fatal` is true and no settings with the given name have been stored (with **StoreSettings()** (p. 1125)).

Parameters

<i>name</i>	Name of settings to restore.
<i>fatal</i>	Whether to throw an exception on an unknown name.

Referenced by `MDOption< T >::MDOption()`, `PyOption< T >::PyOption()`, and `TestOption< N >::TestOption()`.

39.207.7.13 SetPassed()

```
static void SetPassed (
    const std::string & name ) [static]
```

Mark a particular parameter as passed.

Parameters

<i>name</i>	Name of the parameter.
-------------	------------------------

Referenced by `mlpack::util::SetInputParam()`, and `TestOption< N >::TestOption()`.

39.207.7.14 StoreSettings()

```
static void StoreSettings (
    const std::string & name ) [static]
```

Take all parameters and function mappings and store them, under the given name.

This can later be restored with **RestoreSettings()** (p. 1125). If settings have already been saved under the given name, they will be overwritten. This also clears the current parameters and function map.

Parameters

<i>name</i>	Name of settings to save.
-------------	---------------------------

Referenced by MDOption< T >::MDOption(), PyOption< T >::PyOption(), and TestOption< N >::TestOption().

39.207.8 Member Data Documentation

39.207.8.1 didParse

```
bool didParse
```

True, if **CLI** (p. 1117) was used to parse command line options.

Definition at line 299 of file cli.hpp.

Referenced by mlpack::bindings::cli::ParseCommandLine().

39.207.8.2 doc

```
util::ProgramDoc* doc
```

Pointer to the ProgramDoc object.

Definition at line 312 of file cli.hpp.

39.207.8.3 functionMap

```
FunctionMapType functionMap
```

Definition at line 290 of file cli.hpp.

Referenced by CLIOption< N >::CLIOption(), mlpack::bindings::cli::EndProgram(), MDOption< T >::MDOption(), mlpack::bindings::cli::ParseCommandLine(), PyOption< T >::PyOption(), and TestOption< N >::TestOption().

39.207.8.4 `programName`

```
std::string programName
```

Holds the name of the program for `--version`.

This is the true program name (`argv[0]`) not what is given in `ProgramDoc`.

Definition at line 303 of file `cli.hpp`.

39.207.8.5 `timer`

```
Timers timer
```

Holds the timer objects.

Definition at line 306 of file `cli.hpp`.

Referenced by `mlpack::bindings::cli::EndProgram()`, and `mlpack::util::ResetTimers()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ cli.hpp`

39.208 Accuracy Class Reference

The **Accuracy** (p. 1127) is a metric of performance for classification algorithms that is equal to a proportion of correctly labeled test items among all ones for given test items.

Static Public Member Functions

- `template<typename MLEAlgorithm , typename DataType >`
`static double Evaluate (MLEAlgorithm &model, const DataType &data, const arma::Row< size_t > &labels)`
Run classification and calculate accuracy.

Static Public Attributes

- `static const bool NeedsMinimization = false`
Information for hyper-parameter tuning code.

39.208.1 Detailed Description

The **Accuracy** (p. 1127) is a metric of performance for classification algorithms that is equal to a proportion of correctly labeled test items among all ones for given test items.

Definition at line 25 of file accuracy.hpp.

39.208.2 Member Function Documentation

39.208.2.1 Evaluate()

```
static double Evaluate (
    MLAlgorithm & model,
    const DataType & data,
    const arma::Row< size_t > & labels ) [static]
```

Run classification and calculate accuracy.

Parameters

<i>model</i>	A classification model.
<i>data</i>	Column-major data containing test items.
<i>labels</i>	Ground truth (correct) labels for the test items.

39.208.3 Member Data Documentation

39.208.3.1 NeedsMinimization

```
const bool NeedsMinimization = false [static]
```

Information for hyper-parameter tuning code.

It indicates that we want to maximize the metric.

Definition at line 44 of file accuracy.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/ **accuracy.hpp**

39.209 CVBase< MLAlgorithm, MatType, PredictionsType, WeightsType > Class Template Reference

An auxiliary class for cross-validation.

Public Types

- using **MIE** = **MetalInfoExtractor**< MLAlgorithm, MatType, PredictionsType, WeightsType >
A short alias for **MetalInfoExtractor** (p. 1140).

Public Member Functions

- **CVBase** ()
Assert that *MLAlgorithm* doesn't take any additional basic parameters like *numClasses*.
- **CVBase** (const size_t numClasses)
Assert that *MLAlgorithm* takes the *numClasses* parameter and store it.
- **CVBase** (const **data::DatasetInfo** &datasetInfo, const size_t numClasses)
Assert that *MLAlgorithm* takes the *numClasses* parameter and a **data::DatasetInfo** (p. 376) parameter and store them.
- template<typename... MLAlgorithmArgs>
MLAlgorithm Train (const MatType &xs, const PredictionsType &ys, const MLAlgorithmArgs &... args)
Train *MLAlgorithm* with given data points, predictions, and hyperparameters depending on what **CVBase** (p. 1129) constructor has been called.
- template<typename... MLAlgorithmArgs>
MLAlgorithm Train (const MatType &xs, const PredictionsType &ys, const WeightsType &weights, const MLAlgorithmArgs &... args)
Train *MLAlgorithm* with given data points, predictions, weights, and hyperparameters depending on what **CVBase** (p. 1129) constructor has been called.

Static Public Member Functions

- static void **AssertDataConsistency** (const MatType &xs, const PredictionsType &ys)
Assert there is the equal number of data points and predictions.
- static void **AssertWeightsConsistency** (const MatType &xs, const WeightsType &weights)
Assert weighted learning is supported and there is the equal number of data points and weights.

39.209.1 Detailed Description

```
template<typename MLAlgorithm, typename MatType, typename PredictionsType, typename WeightsType>
class mlpack::cv::CVBase< MLAlgorithm, MatType, PredictionsType, WeightsType >
```

An auxiliary class for cross-validation.

It serves to handle basic non-data constructor parameters of a machine learning algorithm (like *datasetInfo* or *numClasses*) and to assert that the machine learning algorithm and data satisfy certain conditions.

This class is not meant to be used directly by users. To cross-validate rather use end-user classes like **SimpleCV** (p. 1151) or **KFoldCV** (p. 1134).

Template Parameters

<i>MLAlgorithm</i>	A machine learning algorithm.
<i>MatType</i>	The type of data.
<i>PredictionsType</i>	The type of predictions (labels/responses).
<i>WeightsType</i>	The type of weights. It supposed to be void* when weights are not supported.

Definition at line 39 of file cv_base.hpp.

39.209.2 Member Typedef Documentation

39.209.2.1 MIE

```
using MIE = MetaInfoExtractor<MLAlgorithm, MatType, PredictionsType, WeightsType>
```

A short alias for **MetaInfoExtractor** (p. 1140).

Definition at line 44 of file cv_base.hpp.

39.209.3 Constructor & Destructor Documentation

39.209.3.1 CVBase() [1/3]

```
CVBase ( )
```

Assert that MLAlgorithm doesn't take any additional basic parameters like numClasses.

39.209.3.2 CVBase() [2/3]

```
CVBase (
    const size_t numClasses )
```

Assert that MLAlgorithm takes the numClasses parameter and store it.

Parameters

<i>numClasses</i>	Number of classes in the dataset.
-------------------	-----------------------------------

39.209.3.3 CVBase() [3/3]

```
CVBase (
    const data::DatasetInfo & datasetInfo,
    const size_t numClasses )
```

Assert that MLAlgorithm takes the numClasses parameter and a **data::DatasetInfo** (p. 376) parameter and store them.

Parameters

<i>datasetInfo</i>	Type information for each dimension of the dataset.
<i>numClasses</i>	Number of classes in the dataset.

39.209.4 Member Function Documentation

39.209.4.1 AssertDataConsistency()

```
static void AssertDataConsistency (
    const MatType & xs,
    const PredictionsType & ys ) [static]
```

Assert there is the equal number of data points and predictions.

39.209.4.2 AssertWeightsConsistency()

```
static void AssertWeightsConsistency (
    const MatType & xs,
    const WeightsType & weights ) [static]
```

Assert weighted learning is supported and there is the equal number of data points and weights.

39.209.4.3 Train() [1/2]

```
MLAlgorithm Train (
    const MatType & xs,
    const PredictionsType & ys,
    const MLAlgorithmArgs &... args )
```

Train MLAlgorithm with given data points, predictions, and hyperparameters depending on what **CVBase** (p.1129) constructor has been called.

39.209.4.4 Train() [2/2]

```
MLAlgorithm Train (
    const MatType & xs,
    const PredictionsType & ys,
    const WeightsType & weights,
    const MLAlgorithmArgs &... args )
```

Train MLAlgorithm with given data points, predictions, weights, and hyperparameters depending on what **CVBase** (p. 1129) constructor has been called.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **cv_base.hpp**

39.210 F1 < AS, PositiveClass > Class Template Reference

F1 (p. 1132) is a metric of performance for classification algorithms that for binary classification is equal to $2 * \textit{precision} * \textit{recall} / (\textit{precision} + \textit{recall})$.

Static Public Member Functions

- template<typename MLAlgorithm, typename DataType>
static double **Evaluate** (MLAlgorithm &model, const DataType &data, const arma::Row< size_t > &labels)
*Run classification and calculate **F1** (p. 1132).*

Static Public Attributes

- static const bool **NeedsMinimization** = false
Information for hyper-parameter tuning code.

39.210.1 Detailed Description

```
template<AverageStrategy AS, size_t PositiveClass = 1>
class mlpack::cv::F1< AS, PositiveClass >
```

F1 (p. 1132) is a metric of performance for classification algorithms that for binary classification is equal to $2 * precision * recall / (precision + recall)$.

For multiclass classification the **F1** (p. 1132) metric can be used with the following strategies for averaging.

1. Micro. The result is calculated by the above formula, but microaveraged precision and microaveraged recall are used.
2. Macro. **F1** (p. 1132) is calculated for each class (with values used for calculation of macroaveraged precision and macroaveraged recall), and then the **F1** (p. 1132) values are averaged.

In the case of multiclass classification it is assumed that there are instances of every label from 0 to max(labels) among input data points.

The returned value for **F1** (p. 1132) will be zero if both precision and recall turn out to be zeros.

Template Parameters

<i>AS</i>	An average strategy.
<i>PositiveClass</i>	In the case of binary classification (AS = Binary) positives are assumed to have labels equal to this value.

Definition at line 45 of file f1.hpp.

39.210.2 Member Function Documentation

39.210.2.1 Evaluate()

```
static double Evaluate (
    MLAlgorithm & model,
    const DataType & data,
    const arma::Row< size_t > & labels ) [static]
```

Run classification and calculate **F1** (p. 1132).

Parameters

<i>model</i>	A classification model.
<i>data</i>	Column-major data containing test items.
<i>labels</i>	Ground truth (correct) labels for the test items.

39.210.3 Member Data Documentation

39.210.3.1 NeedsMinimization

```
const bool NeedsMinimization = false [static]
```

Information for hyper-parameter tuning code.

It indicates that we want to maximize the metric.

Definition at line 64 of file f1.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/ **f1.hpp**

39.211 KFoldCV< MLAlgorithm, Metric, MatType, PredictionsType, WeightsType > Class Template Reference

The class **KFoldCV** (p. 1134) implements k-fold cross-validation for regression and classification algorithms.

Public Member Functions

- **KFoldCV** (const size_t k, const MatType &xs, const PredictionsType &ys, const bool shuffle=true)
This constructor can be used for regression algorithms and for binary classification algorithms.
- **KFoldCV** (const size_t k, const MatType &xs, const PredictionsType &ys, const size_t numClasses, const bool shuffle=true)
This constructor can be used for multiclass classification algorithms.
- **KFoldCV** (const size_t k, const MatType &xs, const **data::DatasetInfo** &datasetInfo, const PredictionsType &ys, const size_t numClasses, const bool shuffle=true)
*This constructor can be used for multiclass classification algorithms that can take a **data::DatasetInfo** (p. 376) parameter.*
- **KFoldCV** (const size_t k, const MatType &xs, const PredictionsType &ys, const WeightsType &weights, const bool shuffle=true)
This constructor can be used for regression and binary classification algorithms that support weighted learning.
- **KFoldCV** (const size_t k, const MatType &xs, const PredictionsType &ys, const size_t numClasses, const WeightsType &weights, const bool shuffle=true)
This constructor can be used for multiclass classification algorithms that support weighted learning.
- **KFoldCV** (const size_t k, const MatType &xs, const **data::DatasetInfo** &datasetInfo, const PredictionsType &ys, const size_t numClasses, const WeightsType &weights, const bool shuffle=true)
*This constructor can be used for multiclass classification algorithms that can take a **data::DatasetInfo** (p. 376) parameter and support weighted learning.*
- template<typename... MLAlgorithmArgs>
double **Evaluate** (const MLAlgorithmArgs &...args)

Run k-fold cross-validation.

- **MLAlgorithm & Model ()**

Access and modify a model from the last run of k-fold cross-validation.

- `template<bool Enabled = !Base::MIE::SupportsWeights, typename = typename std::enable_if<Enabled>::type>`

`void Shuffle ()`

Shuffle the data.

- `template<bool Enabled = Base::MIE::SupportsWeights, typename = typename std::enable_if<Enabled>::type, typename = void>`

`void Shuffle ()`

Shuffle the data.

39.211.1 Detailed Description

```
template<typename MLAlgorithm, typename Metric, typename MatType = arma::mat, typename PredictionsType = typename MetaInfoExtractor<MLAlgorithm, MatType>::PredictionsType, typename WeightsType = typename MetaInfoExtractor<MLAlgorithm, MatType, PredictionsType>::WeightsType>
class mlpack::cv::KFoldCV< MLAlgorithm, Metric, MatType, PredictionsType, WeightsType >
```

The class **KFoldCV** (p. 1134) implements k-fold cross-validation for regression and classification algorithms.

To construct a **KFoldCV** (p. 1134) object you need to pass the `k` parameter and arguments that specify data. For example, you can run 10-fold cross-validation for SoftmaxRegression in the following way.

```
// 100-point 5-dimensional random dataset.
arma::mat data = arma::randu<arma::mat>(5, 100);
// Random labels in the [0, 4] interval.
arma::Row<size_t> labels =
    arma::randi<arma::Row<size_t>>(100, arma::distr_param(0, 4));
size_t numClasses = 5;

KFoldCV<SoftmaxRegression<>, Accuracy> cv(10, data, labels, numClasses);

double lambda = 0.1;
double softmaxAccuracy = cv.Evaluate(lambda);
```

Before calling **Evaluate()** (p. 1139), it is possible to shuffle the data by calling the **Shuffle()** (p. 1139) function. Shuffling is performed at construction time if the parameter `shuffle` is set to `true` in the constructor.

Template Parameters

<i>MLAlgorithm</i>	A machine learning algorithm.
<i>Metric</i>	A metric to assess the quality of a trained model.
<i>MatType</i>	The type of data.
<i>PredictionsType</i>	The type of predictions (should be passed when the predictions type is a template parameter in Train methods of MLAlgorithm).
<i>WeightsType</i>	The type of weights (should be passed when weighted learning is supported, and the weights type is a template parameter in Train methods of MLAlgorithm).

Definition at line 65 of file `k_fold_cv.hpp`.

39.211.2 Constructor & Destructor Documentation

39.211.2.1 KFoldCV() [1/6]

```
KFoldCV (
    const size_t k,
    const MatType & xs,
    const PredictionsType & ys,
    const bool shuffle = true )
```

This constructor can be used for regression algorithms and for binary classification algorithms.

Parameters

<i>k</i>	Number of folds (should be at least 2).
<i>xs</i>	Data points to cross-validate on.
<i>ys</i>	Predictions (labels for classification algorithms and responses for regression algorithms) for each data point.
<i>shuffle</i>	Whether or not to shuffle the data during construction.

39.211.2.2 KFoldCV() [2/6]

```
KFoldCV (
    const size_t k,
    const MatType & xs,
    const PredictionsType & ys,
    const size_t numClasses,
    const bool shuffle = true )
```

This constructor can be used for multiclass classification algorithms.

Parameters

<i>k</i>	Number of folds (should be at least 2).
<i>xs</i>	Data points to cross-validate on.
<i>ys</i>	Labels for each data point.
<i>numClasses</i>	Number of classes in the dataset.
<i>shuffle</i>	Whether or not to shuffle the data during construction.

39.211.2.3 KFoldCV() [3/6]

```

KFoldCV (
    const size_t k,
    const MatType & xs,
    const data::DatasetInfo & datasetInfo,
    const PredictionsType & ys,
    const size_t numClasses,
    const bool shuffle = true )

```

This constructor can be used for multiclass classification algorithms that can take a **data::DatasetInfo** (p. 376) parameter.

Parameters

<i>k</i>	Number of folds (should be at least 2).
<i>xs</i>	Data points to cross-validate on.
<i>datasetInfo</i>	Type information for each dimension of the dataset.
<i>ys</i>	Labels for each data point.
<i>numClasses</i>	Number of classes in the dataset.
<i>shuffle</i>	Whether or not to shuffle the data during construction.

39.211.2.4 KFoldCV() [4/6]

```

KFoldCV (
    const size_t k,
    const MatType & xs,
    const PredictionsType & ys,
    const WeightsType & weights,
    const bool shuffle = true )

```

This constructor can be used for regression and binary classification algorithms that support weighted learning.

Parameters

<i>k</i>	Number of folds (should be at least 2).
<i>xs</i>	Data points to cross-validate on.
<i>ys</i>	Predictions (labels for classification algorithms and responses for regression algorithms) for each data point.
<i>weights</i>	Observation weights (for boosting).
<i>shuffle</i>	Whether or not to shuffle the data during construction.

39.211.2.5 KFoldCV() [5/6]

```

KFoldCV (
    const size_t k,
    const MatType & xs,
    const PredictionsType & ys,
    const size_t numClasses,
    const WeightsType & weights,
    const bool shuffle = true )

```

This constructor can be used for multiclass classification algorithms that support weighted learning.

Parameters

<i>k</i>	Number of folds (should be at least 2).
<i>xs</i>	Data points to cross-validate on.
<i>ys</i>	Labels for each data point.
<i>numClasses</i>	Number of classes in the dataset.
<i>weights</i>	Observation weights (for boosting).
<i>shuffle</i>	Whether or not to shuffle the data during construction.

39.211.2.6 KFoldCV() [6/6]

```

KFoldCV (
    const size_t k,
    const MatType & xs,
    const data::DatasetInfo & datasetInfo,
    const PredictionsType & ys,
    const size_t numClasses,
    const WeightsType & weights,
    const bool shuffle = true )

```

This constructor can be used for multiclass classification algorithms that can take a **data::DatasetInfo** (p. 376) parameter and support weighted learning.

Parameters

<i>k</i>	Number of folds (should be at least 2).
<i>xs</i>	Data points to cross-validate on.
<i>datasetInfo</i>	Type information for each dimension of the dataset.
<i>ys</i>	Labels for each data point.
<i>numClasses</i>	Number of classes in the dataset.
<i>weights</i>	Observation weights (for boosting).
<i>shuffle</i>	Whether or not to shuffle the data during construction.

39.211.3 Member Function Documentation

39.211.3.1 Evaluate()

```
double Evaluate (
    const MLAlgorithmArgs &... args )
```

Run k-fold cross-validation.

Parameters

<i>args</i>	Arguments for MLAlgorithm (in addition to the passed ones in the constructor).
-------------	--

39.211.3.2 Model()

```
MLAlgorithm& Model ( )
```

Access and modify a model from the last run of k-fold cross-validation.

39.211.3.3 Shuffle() [1/2]

```
void Shuffle ( )
```

Shuffle the data.

This overload is called if weights are not supported by the model type.

39.211.3.4 Shuffle() [2/2]

```
void Shuffle ( )
```

Shuffle the data.

This overload is called if weights are supported by the model type.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **k_fold_cv.hpp**

39.212 MetalInfoExtractor< MLAlgorithm, MT, PT, WT > Class Template Reference

MetalInfoExtractor (p. 1140) is a tool for extracting meta information about a given machine learning algorithm.

Public Types

- using **PredictionsType** = typename Select< **TF1**, **TF2**, **TF3**, **TF4**, **TF5** >::Type::PredictionsType
The type of predictions used in MLAlgorithm.
- using **WeightsType** = typename Select< **WTF1**, **WTF2**, **WTF3**, **WTF4**, **WTF5** >::Type::WeightsType
The type of weights used in MLAlgorithm.

Static Public Attributes

- static const bool **IsSupported** = lstd::is_same< **PredictionsType**, void*>::value
An indication whether PredictionsType has been identified (i.e.
- static const bool **SupportsWeights** = lstd::is_same< **WeightsType**, void*>::value
An indication whether MLAlgorithm supports weighted learning.
- static const bool **TakesDatasetInfo** = Selects< **TF5**>::value
*An indication whether MLAlgorithm takes a **data::DatasetInfo** (p. 376) parameter.*
- static const bool **TakesNumClasses** = Selects< **TF4**, **TF5**>::value
An indication whether MLAlgorithm takes the numClasses (size_t) parameter.

39.212.1 Detailed Description

```
template<typename MLAlgorithm, typename MT = arma::mat, typename PT = arma::Row<size_t>, typename WT = arma::rowvec>
class mlpack::cv::MetalInfoExtractor< MLAlgorithm, MT, PT, WT >
```

MetalInfoExtractor (p. 1140) is a tool for extracting meta information about a given machine learning algorithm.

It can be used to automatically extract the type of predictions and weights (if weighted learning is supported), whether the machine learning algorithm takes a DatasetInfo parameter or a numClasses parameter.

The following assumptions are made about the machine learning algorithm.

1. All needed information can be extracted from signatures of Train methods.
2. The machine learning algorithm contains either only non-templated Train methods or only templated ones.
3. Train methods that can be used for extraction of needed information should be distinguishable by a number of arguments (for more information read discussion in <https://github.com/mlpack/mlpack/issues/929>).
4. If weighted learning is supported, passing weights is an option rather than a requirement.

Template Parameters

<i>MLAlgorithm</i>	A machine learning algorithm to investigate.
<i>MT</i>	The type of data.
<i>PT</i>	The type of predictions (should be passed when the predictions type is a template parameter in Train methods of MLAlgorithm).
<i>WT</i>	The type of weights (should be passed when weighted learning is supported, and the weights type is a template parameter in Train methods of MLAlgorithm).

Definition at line 272 of file meta_info_extractor.hpp.

39.212.2 Member Typedef Documentation

39.212.2.1 PredictionsType

```
using PredictionsType = typename Select< TF1, TF2, TF3, TF4, TF5>::Type::PredictionsType
```

The type of predictions used in MLAlgorithm.

It is equal to void* if the extraction fails.

Definition at line 319 of file meta_info_extractor.hpp.

39.212.2.2 WeightsType

```
using WeightsType = typename Select< WTF1, WTF2, WTF3, WTF4, WTF5>::Type::WeightsType
```

The type of weights used in MLAlgorithm.

It is equal to void* if the extraction fails.

Definition at line 326 of file meta_info_extractor.hpp.

39.212.3 Member Data Documentation

39.212.3.1 IsSupported

```
const bool IsSupported = !std::is_same< PredictionsType, void*>::value [static]
```

An indication whether PredictionsType has been identified (i.e.

MLAlgorithm is supported by **MetaInfoExtractor** (p. 1140)).

Definition at line 332 of file meta_info_extractor.hpp.

39.212.3.2 SupportsWeights

```
const bool SupportsWeights = !std::is_same< WeightsType, void*>::value [static]
```

An indication whether MLAlgorithm supports weighted learning.

Definition at line 337 of file meta_info_extractor.hpp.

39.212.3.3 TakesDatasetInfo

```
const bool TakesDatasetInfo = Selects< TF5>::value [static]
```

An indication whether MLAlgorithm takes a **data::DatasetInfo** (p. 376) parameter.

Definition at line 342 of file meta_info_extractor.hpp.

39.212.3.4 TakesNumClasses

```
const bool TakesNumClasses = Selects< TF4, TF5>::value [static]
```

An indication whether MLAlgorithm takes the numClasses (size_t) parameter.

Definition at line 347 of file meta_info_extractor.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **meta_info_extractor.hpp**

39.213 MSE Class Reference

The MeanSquaredError is a metric of performance for regression algorithms that is equal to the mean squared error between predicted values and ground truth (correct) values for given test items.

Static Public Member Functions

- `template<typename MLAlgorithm , typename DataType , typename ResponsesType >`
`static double Evaluate (MLAlgorithm &model, const DataType &data, const ResponsesType &responses)`
Run prediction and calculate the mean squared error.

Static Public Attributes

- `static const bool NeedsMinimization = true`
Information for hyper-parameter tuning code.

39.213.1 Detailed Description

The MeanSquaredError is a metric of performance for regression algorithms that is equal to the mean squared error between predicted values and ground truth (correct) values for given test items.

Definition at line 25 of file mse.hpp.

39.213.2 Member Function Documentation

39.213.2.1 Evaluate()

```
static double Evaluate (
    MLAlgorithm & model,
    const DataType & data,
    const ResponsesType & responses ) [static]
```

Run prediction and calculate the mean squared error.

Parameters

<i>model</i>	A regression model.
<i>data</i>	Column-major data containing test items.
<i>responses</i>	Ground truth (correct) target values for the test items, should be either a row vector or a column-major matrix.

39.213.3 Member Data Documentation

39.213.3.1 NeedsMinimization

```
const bool NeedsMinimization = true [static]
```

Information for hyper-parameter tuning code.

It indicates that we want to minimize the measurement.

Definition at line 45 of file mse.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/ **mse.hpp**

39.214 NotFoundMethodForm Struct Reference

Public Types

- using **PredictionsType** = void *
- using **WeightsType** = void *

39.214.1 Detailed Description

Definition at line 180 of file meta_info_extractor.hpp.

39.214.2 Member Typedef Documentation

39.214.2.1 PredictionsType

```
using PredictionsType = void*
```

Definition at line 182 of file meta_info_extractor.hpp.

39.214.2.2 WeightsType

```
using WeightsType = void*
```

Definition at line 183 of file meta_info_extractor.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ meta_info_extractor.hpp

39.215 Precision< AS, PositiveClass > Class Template Reference

Precision (p. 1145) is a metric of performance for classification algorithms that for binary classification is equal to $tp/(tp + fp)$, where tp and fp are the numbers of true positives and false positives respectively.

Static Public Member Functions

- template<typename MLAlgorithm, typename DataType >
static double **Evaluate** (MLAlgorithm &model, const DataType &data, const arma::Row< size_t > &labels)
Run classification and calculate precision.

Static Public Attributes

- static const bool **NeedsMinimization** = false
Information for hyper-parameter tuning code.

39.215.1 Detailed Description

```
template<AverageStrategy AS, size_t PositiveClass = 1>  
class mlpack::cv::Precision< AS, PositiveClass >
```

Precision (p. 1145) is a metric of performance for classification algorithms that for binary classification is equal to $tp/(tp + fp)$, where tp and fp are the numbers of true positives and false positives respectively.

For multiclass classification the precision metric can be used with the following strategies for averaging.

1. Micro. If there are $N + 1$ classes in total, the result is equal to

$$(tp_0 + tp_1 + \dots + tp_N)/(tp_0 + tp_1 + \dots + tp_N + fp_0 + fp_1 + \dots + fp_N),$$

where tp_i and fp_i are the numbers of true positives and false positives respectively for the class (label) i .

2. Macro. If there are $N + 1$ classes in total, the result is equal to the mean of the values

$$tp_0/(tp_0 + fp_0), tp_1/(tp_1 + fp_1), \dots, tp_N/(tp_N + fp_N),$$

where tp_i and fp_i are the numbers of true positives and false positives respectively for the class (label) i .

Template Parameters

<i>AS</i>	An average strategy.
<i>PositiveClass</i>	In the case of binary classification (AS = Binary) positives are assumed to have labels equal to this value.

Definition at line 48 of file precision.hpp.

39.215.2 Member Function Documentation

39.215.2.1 Evaluate()

```
static double Evaluate (
    MlAlgorithm & model,
    const DataType & data,
    const arma::Row< size_t > & labels ) [static]
```

Run classification and calculate precision.

Parameters

<i>model</i>	A classification model.
<i>data</i>	Column-major data containing test items.
<i>labels</i>	Ground truth (correct) labels for the test items.

39.215.3 Member Data Documentation

39.215.3.1 NeedsMinimization

```
const bool NeedsMinimization = false [static]
```

Information for hyper-parameter tuning code.

It indicates that we want to maximize the metric.

Definition at line 67 of file precision.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/ **precision.hpp**

39.216 Recall< AS, PositiveClass > Class Template Reference

Recall (p. 1147) is a metric of performance for classification algorithms that for binary classification is equal to $tp/(tp + fn)$, where tp and fn are the numbers of true positives and false negatives respectively.

Static Public Member Functions

- `template<typename MlAlgorithm , typename DataType >`
`static double Evaluate (MlAlgorithm &model, const DataType &data, const arma::Row< size_t > &labels)`
Run classification and calculate recall.

Static Public Attributes

- `static const bool NeedsMinimization = false`
Information for hyper-parameter tuning code.

39.216.1 Detailed Description

```
template<AverageStrategy AS, size_t PositiveClass = 1>
class mlpack::cv::Recall< AS, PositiveClass >
```

Recall (p. 1147) is a metric of performance for classification algorithms that for binary classification is equal to $tp/(tp + fn)$, where tp and fn are the numbers of true positives and false negatives respectively.

For multiclass classification the recall metric can be used with the following strategies for averaging.

1. Micro. If there are $N + 1$ classes in total, the result is equal to

$$(tp_0 + tp_1 + \dots + tp_N)/(tp_0 + tp_1 + \dots + tp_N + fn_0 + fn_1 + \dots + fn_N),$$

where tp_i and fn_i are the numbers of true positives and false negatives respectively for the class (label) i .

2. Macro. If there are $N + 1$ classes in total, the result is equal to the mean of the values

$$tp_0/(tp_0 + fn_0), tp_1/(tp_1 + fn_1), \dots, tp_N/(tp_N + fn_N),$$

where tp_i and fn_i are the numbers of true positives and false negatives respectively for the class (label) i .

Template Parameters

AS	An average strategy.
PositiveClass	In the case of binary classification (AS = Binary) positives are assumed to have labels equal to this value.

Definition at line 48 of file recall.hpp.

39.216.2 Member Function Documentation

39.216.2.1 Evaluate()

```
static double Evaluate (
    MLAlgorithm & model,
    const DataType & data,
    const arma::Row< size_t > & labels ) [static]
```

Run classification and calculate recall.

Parameters

<i>model</i>	A classification model.
<i>data</i>	Column-major data containing test items.
<i>labels</i>	Ground truth (correct) labels for the test items.

39.216.3 Member Data Documentation

39.216.3.1 NeedsMinimization

```
const bool NeedsMinimization = false [static]
```

Information for hyper-parameter tuning code.

It indicates that we want to maximize the metric.

Definition at line 67 of file recall.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/ **recall.hpp**

39.217 SelectMethodForm< MLAlgorithm, HMFs > Struct Template Reference

A type function that selects a right method form.

39.217.1 Detailed Description

```
template<typename MLAlgorithm, template< class, template< class... > class, size_t > class... HMFs>
struct mlpack::cv::SelectMethodForm< MLAlgorithm, HMFs >
```

A type function that selects a right method form.

As parameters it takes a machine learning algorithm, a set of HasMethodForm structs, and a set of method forms. Method forms are passed to the internal struct From. The result type can be accessed through the Type member of the From struct.

The implementation basically loops through all combinations of the HasMethodForm structs and the method forms. It stops when a right combination succeeds, or when there are no more combinations.

Definition at line 198 of file meta_info_extractor.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **meta_info_extractor.hpp**

39.218 SelectMethodForm< MLAlgorithm > Struct Template Reference

Classes

- struct **From**

39.218.1 Detailed Description

```
template<typename MLAlgorithm>
struct mlpack::cv::SelectMethodForm< MLAlgorithm >
```

Definition at line 234 of file meta_info_extractor.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **meta_info_extractor.hpp**

39.219 SelectMethodForm< MLAlgorithm >::From< Forms > Struct Template Reference

Public Types

- using **Type** = **NotFoundMethodForm**

39.219.1 Detailed Description

```
template<typename MAlgorithm>
template<typename... Forms>
struct mlpack::cv::SelectMethodForm< MAlgorithm >::From< Forms >
```

Definition at line 237 of file meta_info_extractor.hpp.

39.219.2 Member Typedef Documentation

39.219.2.1 Type

```
using Type = NotFoundMethodForm
```

Definition at line 239 of file meta_info_extractor.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **meta_info_extractor.hpp**

39.220 SelectMethodForm< MAlgorithm, HasMethodForm, HMFs... > Struct Template Reference

Classes

- class **From**

39.220.1 Detailed Description

```
template<typename MAlgorithm, template< class, template< class... > class, size_t > class HasMethodForm, template< class,
template< class... > class, size_t > class... HMFs>
struct mlpack::cv::SelectMethodForm< MAlgorithm, HasMethodForm, HMFs... >
```

Definition at line 203 of file meta_info_extractor.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **meta_info_extractor.hpp**

39.221 SelectMethodForm< MLAlgorithm, HasMethodForm, HMFs... >::From< Forms > Class Template Reference

Public Types

- using **Type** = typename Implementation< Forms... >:: **Type**

39.221.1 Detailed Description

```
template<typename MLAlgorithm, template< class, template< class... > class, size_t > class HasMethodForm, template< class,
template< class... > class, size_t > class... HMFs>
template<typename... Forms>
class mlpack::cv::SelectMethodForm< MLAlgorithm, HasMethodForm, HMFs... >::From< Forms >
```

Definition at line 206 of file meta_info_extractor.hpp.

39.221.2 Member Typedef Documentation

39.221.2.1 Type

```
using Type = typename Implementation<Forms...>:: Type
```

Definition at line 229 of file meta_info_extractor.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **meta_info_extractor.hpp**

39.222 SimpleCV< MLAlgorithm, Metric, MatType, PredictionsType, WeightsType > Class Template Reference

SimpleCV (p. 1151) splits data into two sets - training and validation sets - and then runs training on the training set and evaluates performance on the validation set.

Public Member Functions

- `template<typename MatInType , typename PredictionsInType >`
SimpleCV (const double validationSize, MatInType &&xs, PredictionsInType &&ys)
This constructor can be used for regression algorithms and for binary classification algorithms.
- `template<typename MatInType , typename PredictionsInType >`
SimpleCV (const double validationSize, MatInType &&xs, PredictionsInType &&ys, const size_t numClasses)
This constructor can be used for multiclass classification algorithms.
- `template<typename MatInType , typename PredictionsInType >`
SimpleCV (const double validationSize, MatInType &&xs, const **data::DatasetInfo** &datasetInfo, PredictionsInType &&ys, const size_t numClasses)
*This constructor can be used for multiclass classification algorithms that can take a **data::DatasetInfo** (p. 376) parameter.*
- `template<typename MatInType , typename PredictionsInType , typename WeightsInType >`
SimpleCV (const double validationSize, MatInType &&xs, PredictionsInType &&ys, WeightsInType &&weights)
This constructor can be used for regression and binary classification algorithms that support weighted learning.
- `template<typename MatInType , typename PredictionsInType , typename WeightsInType >`
SimpleCV (const double validationSize, MatInType &&xs, PredictionsInType &&ys, const size_t numClasses, WeightsInType &&weights)
This constructor can be used for multiclass classification algorithms that support weighted learning.
- `template<typename MatInType , typename PredictionsInType , typename WeightsInType >`
SimpleCV (const double validationSize, MatInType &&xs, const **data::DatasetInfo** &datasetInfo, PredictionsInType &&ys, const size_t numClasses, WeightsInType &&weights)
*This constructor can be used for multiclass classification algorithms that can take a **data::DatasetInfo** (p. 376) parameter and support weighted learning.*
- `template<typename... MLAlgorithmArgs>`
double **Evaluate** (const MLAlgorithmArgs &... args)
Train on the training set and assess performance on the validation set by using the class Metric.
- `MLAlgorithm & Model ()`
Access and modify the last trained model.

39.222.1 Detailed Description

```
template<typename MLAlgorithm, typename Metric, typename MatType = arma::mat, typename PredictionsType = typename MetaInfoExtractor<MLAlgorithm, MatType>::PredictionsType, typename WeightsType = typename MetaInfoExtractor<MLAlgorithm, MatType, PredictionsType>::WeightsType>
class mlpack::cv::SimpleCV< MLAlgorithm, Metric, MatType, PredictionsType, WeightsType >
```

SimpleCV (p. 1151) splits data into two sets - training and validation sets - and then runs training on the training set and evaluates performance on the validation set.

To construct a **SimpleCV** (p. 1151) object you need to pass the validationSize parameter and arguments that specify data. For example, SoftmaxRegression can be validated in the following way.

```
// 100-point 5-dimensional random dataset.
arma::mat data = arma::randu<arma::mat>(5, 100);
// Random labels in the [0, 4] interval.
arma::Row<size_t> labels =
    arma::randi<arma::Row<size_t>>(100, arma::distr_param(0, 4));
size_t numClasses = 5;

double validationSize = 0.2;
SimpleCV<SoftmaxRegression<>, Accuracy> cv(validationSize, data, labels,
    numClasses);

double lambda = 0.1;
double softmaxAccuracy = cv.Evaluate(lambda);
```


In the example above, 80% of the passed dataset will be used for training, and remaining 20% will be used for calculating the accuracy metric.

Template Parameters

<i>MLAlgorithm</i>	A machine learning algorithm.
<i>Metric</i>	A metric to assess the quality of a trained model.
<i>MatType</i>	The type of data.
<i>PredictionsType</i>	The type of predictions (should be passed when the predictions type is a template parameter in Train methods of the given MLAlgorithm; arma::Row<size_t> will be used otherwise).
<i>WeightsType</i>	The type of weights (should be passed when weighted learning is supported, and the weights type is a template parameter in Train methods of the given MLAlgorithm; arma::vec will be used otherwise).

Definition at line 68 of file simple_cv.hpp.

39.222.2 Constructor & Destructor Documentation

39.222.2.1 SimpleCV() [1/6]

```
SimpleCV (
    const double validationSize,
    MatInType && xs,
    PredictionsInType && ys )
```

This constructor can be used for regression algorithms and for binary classification algorithms.

Parameters

<i>validationSize</i>	A proportion (between 0 and 1) of data used as a validation set.
<i>xs</i>	Data points to cross-validate on.
<i>ys</i>	Predictions (labels for classification algorithms and responses for regression algorithms) for each data point.

Template Parameters

<i>MatInType</i>	A type that can be converted to MatType.
<i>PredictionsInType</i>	A type that can be converted to PredictionsType.

39.222.2.2 SimpleCV() [2/6]

```
SimpleCV (
    const double validationSize,
    MatInType && xs,
    PredictionsInType && ys,
    const size_t numClasses )
```

This constructor can be used for multiclass classification algorithms.

Parameters

<i>validationSize</i>	A proportion (between 0 and 1) of data used as a validation set.
<i>xs</i>	Data points to cross-validate on.
<i>ys</i>	Labels for each data point.
<i>numClasses</i>	Number of classes in the dataset.

Template Parameters

<i>MatInType</i>	A type that can be converted to MatType.
<i>PredictionsInType</i>	A type that can be converted to PredictionsType.

39.222.2.3 SimpleCV() [3/6]

```
SimpleCV (
    const double validationSize,
    MatInType && xs,
    const data::DatasetInfo & datasetInfo,
    PredictionsInType && ys,
    const size_t numClasses )
```

This constructor can be used for multiclass classification algorithms that can take a **data::DatasetInfo** (p. 376) parameter.

Parameters

<i>validationSize</i>	A proportion (between 0 and 1) of data used as a validation set.
<i>xs</i>	Data points to cross-validate on.
<i>datasetInfo</i>	Type information for each dimension of the dataset.
<i>ys</i>	Labels for each data point.
<i>numClasses</i>	Number of classes in the dataset.

Template Parameters

<i>MatInType</i>	A type that can be converted to MatType.
------------------	--

Template Parameters

<i>PredictionsInType</i>	A type that can be converted to PredictionsType.
--------------------------	--

39.222.2.4 SimpleCV() [4/6]

```
SimpleCV (
    const double validationSize,
    MatInType && xs,
    PredictionsInType && ys,
    WeightsInType && weights )
```

This constructor can be used for regression and binary classification algorithms that support weighted learning.

Parameters

<i>validationSize</i>	A proportion (between 0 and 1) of data used as a validation set.
<i>xs</i>	Data points to cross-validate on.
<i>ys</i>	Predictions (labels for classification algorithms and responses for regression algorithms) for each data point.
<i>weights</i>	Observation weights (for boosting).

Template Parameters

<i>MatInType</i>	A type that can be converted to MatType.
<i>PredictionsInType</i>	A type that can be converted to PredictionsType.
<i>WeightsInType</i>	A type that can be converted to WeightsType.

39.222.2.5 SimpleCV() [5/6]

```
SimpleCV (
    const double validationSize,
    MatInType && xs,
    PredictionsInType && ys,
    const size_t numClasses,
    WeightsInType && weights )
```

This constructor can be used for multiclass classification algorithms that support weighted learning.

Parameters

<i>validationSize</i>	A proportion (between 0 and 1) of data used as a validation set.
-----------------------	--

Parameters

<i>xs</i>	Data points to cross-validate on.
<i>ys</i>	Labels for each data point.
<i>numClasses</i>	Number of classes in the dataset.
<i>weights</i>	Observation weights (for boosting).

Template Parameters

<i>MatInType</i>	A type that can be converted to MatType.
<i>PredictionsInType</i>	A type that can be converted to PredictionsType.
<i>WeightsInType</i>	A type that can be converted to WeightsType.

39.222.2.6 SimpleCV() [6/6]

```
SimpleCV (
    const double validationSize,
    MatInType && xs,
    const data::DatasetInfo & datasetInfo,
    PredictionsInType && ys,
    const size_t numClasses,
    WeightsInType && weights )
```

This constructor can be used for multiclass classification algorithms that can take a **data::DatasetInfo** (p. 376) parameter and support weighted learning.

Parameters

<i>validationSize</i>	A proportion (between 0 and 1) of data used as a validation set.
<i>xs</i>	Data points to cross-validate on.
<i>datasetInfo</i>	Type information for each dimension of the dataset.
<i>ys</i>	Labels for each data point.
<i>numClasses</i>	Number of classes in the dataset.
<i>weights</i>	Observation weights (for boosting).

Template Parameters

<i>MatInType</i>	A type that can be converted to MatType.
<i>PredictionsInType</i>	A type that can be converted to PredictionsType.
<i>WeightsInType</i>	A type that can be converted to WeightsType.

39.222.3 Member Function Documentation

39.222.3.1 Evaluate()

```
double Evaluate (
    const MLAlgorithmArgs &... args )
```

Train on the training set and assess performance on the validation set by using the class Metric.

Parameters

<i>args</i>	Arguments for the given MLAlgorithm taken by its constructor (in addition to the passed ones in the SimpleCV (p. 1151) constructor).
-------------	---

39.222.3.2 Model()

```
MLAlgorithm& Model ( )
```

Access and modify the last trained model.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **simple_cv.hpp**

39.223 TrainForm< MatType, PredictionsType, WeightsType, DatasetInfo, NumClasses > Struct Template Reference

A wrapper struct for holding a Train form.

39.223.1 Detailed Description

```
template<typename MatType, typename PredictionsType, typename WeightsType, bool DatasetInfo, bool NumClasses>
struct mlpack::cv::TrainForm< MatType, PredictionsType, WeightsType, DatasetInfo, NumClasses >
```

A wrapper struct for holding a Train form.

Template Parameters

<i>MatType</i>	The type of data.
<i>PredictionsType</i>	The type of predictions.
<i>WeightsType</i>	The type of weights.
<i>DatasetInfo</i>	An indicator whether a data::DatasetInfo (p. 376) parameter should be present.
<i>NumClasses</i>	An indicator whether the numClasses parameter should be present.

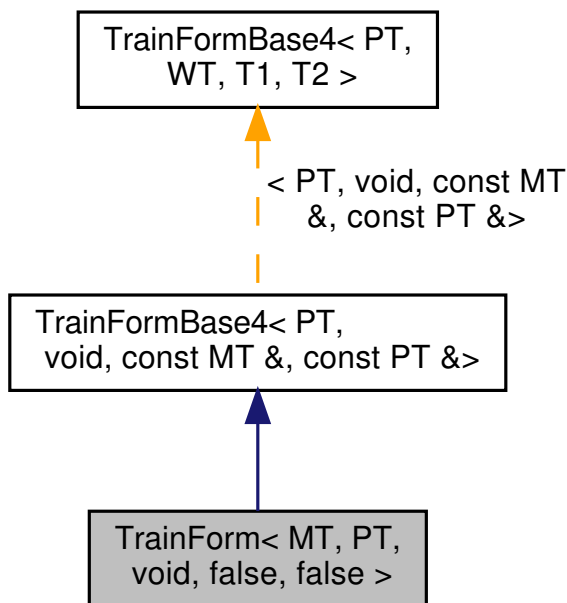
Definition at line 39 of file meta_info_extractor.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **meta_info_extractor.hpp**

39.224 TrainForm< MT, PT, void, false, false > Struct Template Reference

Inheritance diagram for TrainForm< MT, PT, void, false, false >:



Additional Inherited Members

39.224.1 Detailed Description

```
template<typename MT, typename PT>
struct mlpack::cv::TrainForm< MT, PT, void, false, false >
```

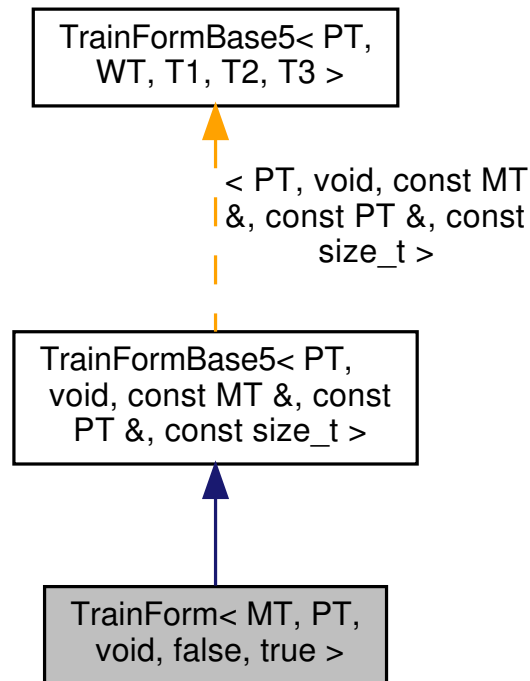
Definition at line 100 of file meta_info_extractor.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **meta_info_extractor.hpp**

39.225 TrainForm< MT, PT, void, false, true > Struct Template Reference

Inheritance diagram for TrainForm< MT, PT, void, false, true >:



Additional Inherited Members

39.225.1 Detailed Description

```
template<typename MT, typename PT>
struct mlpack::cv::TrainForm< MT, PT, void, false, true >
```

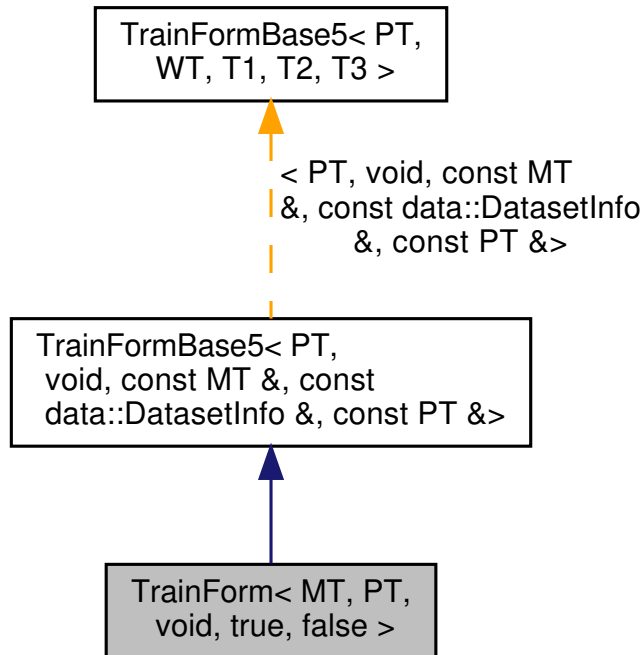
Definition at line 116 of file `meta_info_extractor.hpp`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ meta_info_extractor.hpp`

39.226 TrainForm< MT, PT, void, true, false > Struct Template Reference

Inheritance diagram for TrainForm< MT, PT, void, true, false >:



Additional Inherited Members

39.226.1 Detailed Description

```
template<typename MT, typename PT>
struct mpack::cv::TrainForm< MT, PT, void, true, false >
```

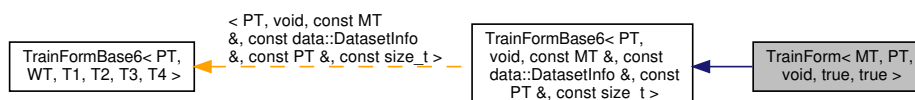
Definition at line 104 of file meta_info_extractor.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **meta_info_extractor.hpp**

39.227 TrainForm< MT, PT, void, true, true > Struct Template Reference

Inheritance diagram for TrainForm< MT, PT, void, true, true >:



Additional Inherited Members

39.227.1 Detailed Description

```
template<typename MT, typename PT>
struct mpack::cv::TrainForm< MT, PT, void, true, true >
```

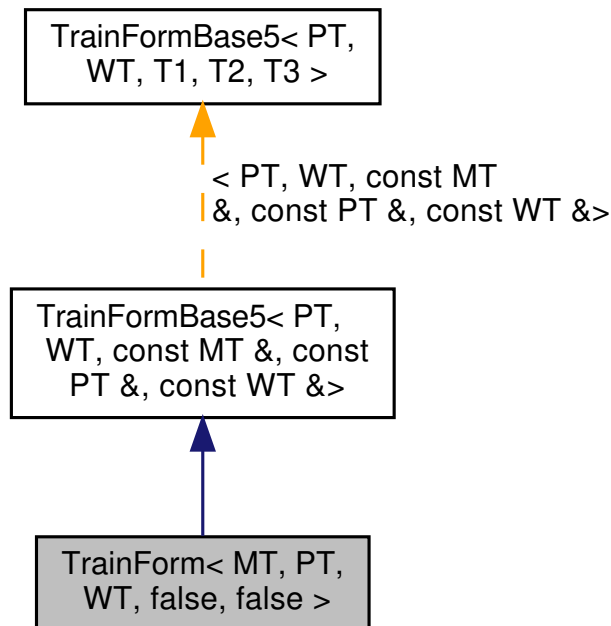
Definition at line 120 of file meta_info_extractor.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **meta_info_extractor.hpp**

39.228 TrainForm< MT, PT, WT, false, false > Struct Template Reference

Inheritance diagram for TrainForm< MT, PT, WT, false, false >:



Additional Inherited Members

39.228.1 Detailed Description

```
template<typename MT, typename PT, typename WT>
struct mpack::cv::TrainForm< MT, PT, WT, false, false >
```

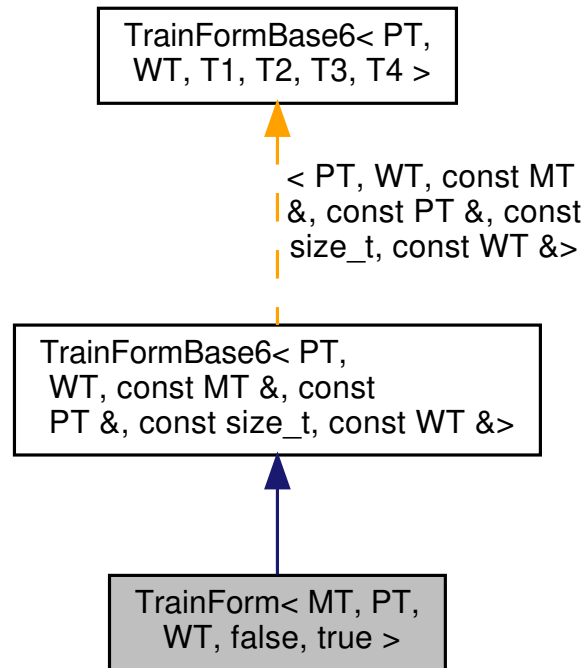
Definition at line 108 of file meta_info_extractor.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **meta_info_extractor.hpp**

39.229 TrainForm< MT, PT, WT, false, true > Struct Template Reference

Inheritance diagram for TrainForm< MT, PT, WT, false, true >:



Additional Inherited Members

39.229.1 Detailed Description

```
template<typename MT, typename PT, typename WT>
struct mlpack::cv::TrainForm< MT, PT, WT, false, true >
```

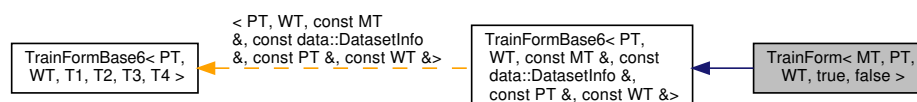
Definition at line 124 of file `meta_info_extractor.hpp`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ meta_info_extractor.hpp`

39.230 TrainForm< MT, PT, WT, true, false > Struct Template Reference

Inheritance diagram for TrainForm< MT, PT, WT, true, false >:



Additional Inherited Members

39.230.1 Detailed Description

```
template<typename MT, typename PT, typename WT>
struct mlpack::cv::TrainForm< MT, PT, WT, true, false >
```

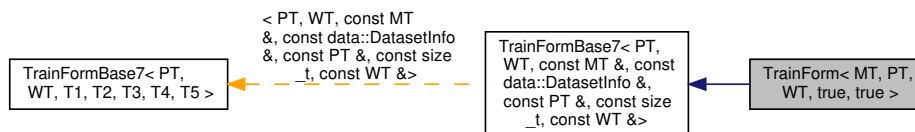
Definition at line 112 of file meta_info_extractor.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **meta_info_extractor.hpp**

39.231 TrainForm< MT, PT, WT, true, true > Struct Template Reference

Inheritance diagram for TrainForm< MT, PT, WT, true, true >:



Additional Inherited Members

39.231.1 Detailed Description

```
template<typename MT, typename PT, typename WT>
struct mlpack::cv::TrainForm< MT, PT, WT, true, true >
```

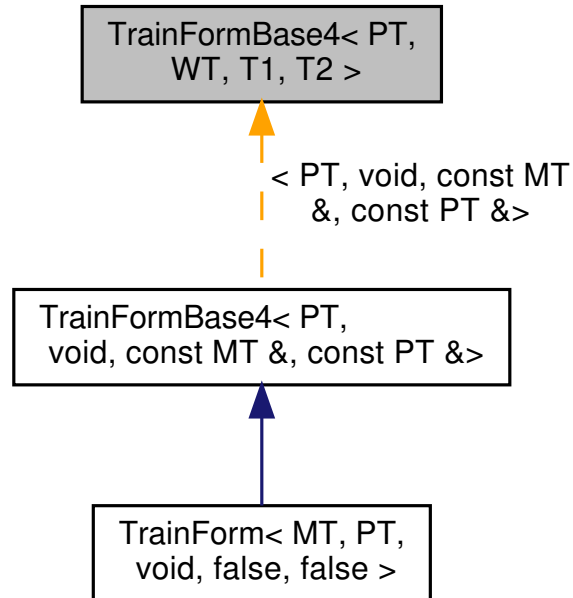
Definition at line 128 of file meta_info_extractor.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **meta_info_extractor.hpp**

39.232 TrainFormBase4< PT, WT, T1, T2 > Struct Template Reference

Inheritance diagram for TrainFormBase4< PT, WT, T1, T2 >:



Public Types

- using **PredictionsType** = PT
- template<typename Class , typename RT , typename... Ts>
using **Type** = RT(Class::*)(T1, T2, Ts...)
- using **WeightsType** = WT

Static Public Attributes

- static const size_t **MinNumberOfAdditionalArgs** = 1

39.232.1 Detailed Description

```
template<typename PT, typename WT, typename T1, typename T2>
struct mpack::cv::TrainFormBase4< PT, WT, T1, T2 >
```

Definition at line 46 of file meta_info_extractor.hpp.

39.232.2 Member Typedef Documentation

39.232.2.1 PredictionsType

```
using PredictionsType = PT
```

Definition at line 48 of file meta_info_extractor.hpp.

39.232.2.2 Type

```
using Type = RT(Class::*)(T1, T2, Ts...)
```

Definition at line 55 of file meta_info_extractor.hpp.

39.232.2.3 WeightsType

```
using WeightsType = WT
```

Definition at line 49 of file meta_info_extractor.hpp.

39.232.3 Member Data Documentation

39.232.3.1 MinNumberOfAdditionalArgs

```
const size_t MinNumberOfAdditionalArgs = 1 [static]
```

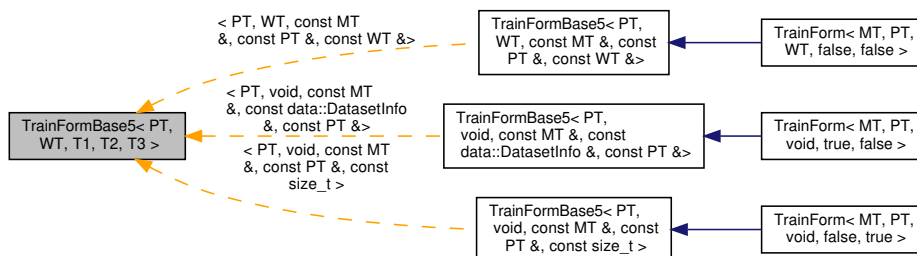
Definition at line 52 of file meta_info_extractor.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ meta_info_extractor.hpp

39.233 TrainFormBase5< PT, WT, T1, T2, T3 > Struct Template Reference

Inheritance diagram for TrainFormBase5< PT, WT, T1, T2, T3 >:



Public Types

- using **PredictionsType** = PT
- template<typename Class , typename RT , typename... Ts>
using **Type** = RT(Class::*)(T1, T2, T3, Ts...)
- using **WeightsType** = WT

Static Public Attributes

- static const size_t **MinNumberOfAdditionalArgs** = 1

39.233.1 Detailed Description

```
template<typename PT, typename WT, typename T1, typename T2, typename T3>  
struct mpack::cv::TrainFormBase5< PT, WT, T1, T2, T3 >
```

Definition at line 59 of file meta_info_extractor.hpp.

39.233.2 Member Typedef Documentation

39.233.2.1 PredictionsType

```
using PredictionsType = PT
```

Definition at line 61 of file meta_info_extractor.hpp.

39.233.2.2 Type

```
using Type = RT(Class::*)(T1, T2, T3, Ts...)
```

Definition at line 68 of file meta_info_extractor.hpp.

39.233.2.3 WeightsType

```
using WeightsType = WT
```

Definition at line 62 of file meta_info_extractor.hpp.

39.233.3 Member Data Documentation

39.233.3.1 MinNumberOfAdditionalArgs

```
const size_t MinNumberOfAdditionalArgs = 1 [static]
```

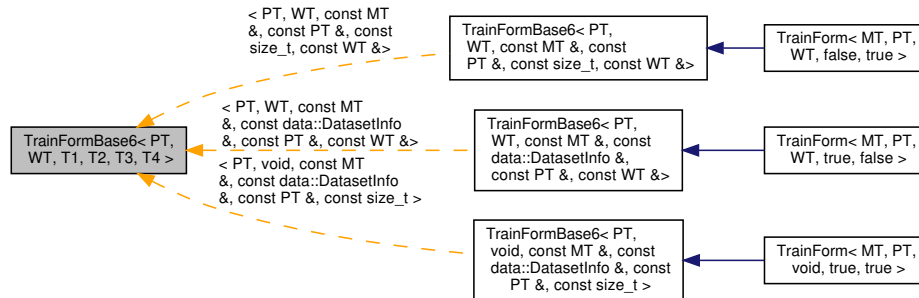
Definition at line 65 of file meta_info_extractor.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **meta_info_extractor.hpp**

39.234 TrainFormBase6< PT, WT, T1, T2, T3, T4 > Struct Template Reference

Inheritance diagram for TrainFormBase6< PT, WT, T1, T2, T3, T4 >:



Public Types

- using **PredictionsType** = PT
- template<typename Class, typename RT, typename... Ts>
using **Type** = RT(Class::*)(T1, T2, T3, T4, Ts...)
- using **WeightsType** = WT

Static Public Attributes

- static const size_t **MinNumberOfAdditionalArgs** = 1

39.234.1 Detailed Description

```
template<typename PT, typename WT, typename T1, typename T2, typename T3, typename T4>
struct mlpack::cv::TrainFormBase6< PT, WT, T1, T2, T3, T4 >
```

Definition at line 73 of file meta_info_extractor.hpp.

39.234.2 Member Typedef Documentation

39.234.2.1 PredictionsType

```
using PredictionsType = PT
```

Definition at line 75 of file meta_info_extractor.hpp.

39.234.2.2 Type

```
using Type = RT(Class::*)(T1, T2, T3, T4, Ts...)
```

Definition at line 82 of file meta_info_extractor.hpp.

39.234.2.3 WeightsType

```
using WeightsType = WT
```

Definition at line 76 of file meta_info_extractor.hpp.

39.234.3 Member Data Documentation

39.234.3.1 MinNumberOfAdditionalArgs

```
const size_t MinNumberOfAdditionalArgs = 1 [static]
```

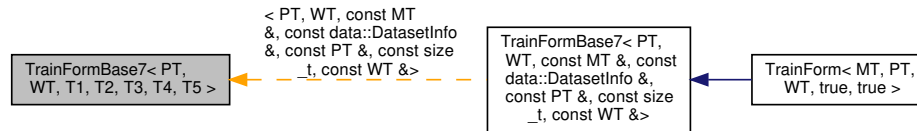
Definition at line 79 of file meta_info_extractor.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **meta_info_extractor.hpp**

39.235 TrainFormBase7< PT, WT, T1, T2, T3, T4, T5 > Struct Template Reference

Inheritance diagram for TrainFormBase7< PT, WT, T1, T2, T3, T4, T5 >:



Public Types

- using **PredictionsType** = PT
- template<typename Class , typename RT , typename... Ts>
using **Type** = RT(Class::*)(T1, T2, T3, T4, T5, Ts...)
- using **WeightsType** = WT

Static Public Attributes

- static const size_t **MinNumberOfAdditionalArgs** = 1

39.235.1 Detailed Description

```
template<typename PT, typename WT, typename T1, typename T2, typename T3, typename T4, typename T5>
struct mlpack::cv::TrainFormBase7< PT, WT, T1, T2, T3, T4, T5 >
```

Definition at line 87 of file meta_info_extractor.hpp.

39.235.2 Member Typedef Documentation

39.235.2.1 PredictionsType

```
using PredictionsType = PT
```

Definition at line 89 of file meta_info_extractor.hpp.

39.235.2.2 Type

```
using Type = RT(Class::*)(T1, T2, T3, T4, T5, Ts...)
```

Definition at line 96 of file meta_info_extractor.hpp.

39.235.2.3 WeightsType

```
using WeightsType = WT
```

Definition at line 90 of file meta_info_extractor.hpp.

39.235.3 Member Data Documentation

39.235.3.1 MinNumberOfAdditionalArgs

```
const size_t MinNumberOfAdditionalArgs = 1 [static]
```

Definition at line 93 of file meta_info_extractor.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/ **meta_info_extractor.hpp**

39.236 CustomImputation< T > Class Template Reference

A simple custom imputation class.

Public Member Functions

- **CustomImputation** (T customValue)
- void **Impute** (arma::Mat< T > &input, const T &mappedValue, const size_t dimension, const bool column↔Major=true)

Impute function searches through the input looking for mappedValue and replaces it with the user-defined custom value of the given dimension.

39.236.1 Detailed Description

```
template<typename T>
class mlpack::data::CustomImputation< T >
```

A simple custom imputation class.

Template Parameters

<i>T</i>	Type of armadillo matrix
----------	--------------------------

Definition at line 24 of file custom_imputation.hpp.

39.236.2 Constructor & Destructor Documentation

39.236.2.1 CustomImputation()

```
CustomImputation (
    T customValue ) [inline]
```

Definition at line 27 of file custom_imputation.hpp.

39.236.3 Member Function Documentation

39.236.3.1 Impute()

```
void Impute (
    arma::Mat< T > & input,
    const T & mappedValue,
    const size_t dimension,
    const bool columnMajor = true ) [inline]
```

Impute function searches through the input looking for mappedValue and replaces it with the user-defined custom value of the given dimension.

The result is overwritten to the input, not creating any copy. Custom value must be set when initializing the **CustomImputation** (p. 1170) object.

Parameters

<i>input</i>	Matrix that contains mappedValue.
<i>mappedValue</i>	Value that the user wants to get rid of.
<i>dimension</i>	Index of the dimension of the mappedValue.
<i>columnMajor</i>	State of whether the input matrix is columnMajor or not.

Definition at line 44 of file custom_imputation.hpp.

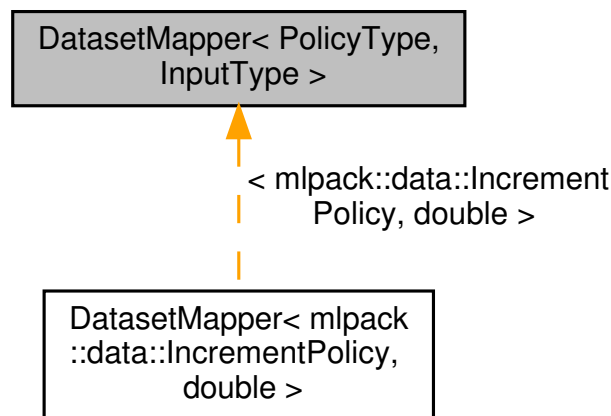
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/ **custom_imputation.hpp**

39.237 DatasetMapper< PolicyType, InputType > Class Template Reference

Auxiliary information for a dataset, including mappings to/from strings (or other types) and the datatype of each dimension.

Inheritance diagram for DatasetMapper< PolicyType, InputType >:



Public Member Functions

- **DatasetMapper** (const size_t dimensionality=0)
Create the **DatasetMapper** (p. 1172) object with the given dimensionality.
- **DatasetMapper** (PolicyType &policy, const size_t dimensionality=0)
Create the **DatasetMapper** (p. 1172) object with the given policy and dimensionality.
- size_t **Dimensionality** () const
Get the dimensionality of the **DatasetMapper** (p. 1172) object (that is, how many dimensions it has information for).
- template<typename T >
void **MapFirstPass** (const InputType &input, const size_t dimension)
Preprocessing: during a first pass of the data, pass the input on to the MapPolicy if they are needed.
- template<typename T >
T **MapString** (const InputType &input, const size_t dimension)
Given the input and the dimension to which it belongs, return its numeric mapping.
- size_t **NumMappings** (const size_t dimension) const
Get the number of mappings for a particular dimension.
- template<typename T >
size_t **NumUnmappings** (const T value, const size_t dimension) const
Get the number of possible unmappings for a string in a given dimension.
- const PolicyType & **Policy** () const

- Return the policy of the mapper.*
- PolicyType & **Policy** ()
 - Modify the policy of the mapper (be careful!).*
- void **Policy** (PolicyType &&policy)
 - Modify (Replace) the policy of the mapper with a new policy.*
- template<typename Archive >
 - void **serialize** (Archive &ar, const unsigned int)
 - Serialize the dataset information.*
- void **SetDimensionality** (const size_t dimensionality)
 - Set the dimensionality of an existing **DatasetMapper** (p. 1172) object.*
- **Datatype Type** (const size_t dimension) const
 - Return the type of a given dimension (numeric or categorical).*
- **Datatype & Type** (const size_t dimension)
 - Modify the type of a given dimension (be careful!).*
- template<typename T >
 - const InputType & **UnmapString** (const T value, const size_t dimension, const size_t unmappingIndex=0) const
 - Return the input that corresponds to a given value in a given dimension.*
- PolicyType::MappedType **UnmapValue** (const InputType &input, const size_t dimension)
 - Return the value that corresponds to a given input in a given dimension.*

39.237.1 Detailed Description

```
template<typename PolicyType, typename InputType = std::string>
class mlpack::data::DatasetMapper< PolicyType, InputType >
```

Auxiliary information for a dataset, including mappings to/from strings (or other types) and the datatype of each dimension.

DatasetMapper (p. 1172) objects are optionally produced by **data::Load()** (p. 379), and store the type of each dimension (**Datatype::numeric** or **Datatype::categorical**) as well as mappings from strings to unsigned integers and vice versa.

DatasetMapper (p. 1172) objects can also map from arbitrary types; the type to map from can be specified with the **InputType** template parameter. By default, the **InputType** parameter is **std::string**.

Template Parameters

<i>PolicyType</i>	Mapping policy used to specify MapString() (p. 1175).
<i>InputType</i>	Type of input to be mapped.

Definition at line 41 of file `dataset_mapper.hpp`.

39.237.2 Constructor & Destructor Documentation

39.237.2.1 DatasetMapper() [1/2]

```
DatasetMapper (
    const size_t dimensionality = 0 ) [explicit]
```

Create the **DatasetMapper** (p. 1172) object with the given dimensionality.

Note that the dimensionality cannot be changed later; you will have to create a new **DatasetMapper** (p. 1172) object.

39.237.2.2 DatasetMapper() [2/2]

```
DatasetMapper (
    PolicyType & policy,
    const size_t dimensionality = 0 ) [explicit]
```

Create the **DatasetMapper** (p. 1172) object with the given policy and dimensionality.

Note that the dimensionality cannot be changed later; you will have to create a new **DatasetMapper** (p. 1172) object. Policy can be modified by the modifier.

39.237.3 Member Function Documentation**39.237.3.1 Dimensionality()**

```
size_t Dimensionality ( ) const
```

Get the dimensionality of the **DatasetMapper** (p. 1172) object (that is, how many dimensions it has information for).

If this object was created by a call to **mlpack::data::Load()** (p. 379), then the dimensionality will be the same as the number of rows (dimensions) in the dataset.

39.237.3.2 MapFirstPass()

```
void MapFirstPass (
    const InputType & input,
    const size_t dimension )
```

Preprocessing: during a first pass of the data, pass the input on to the MapPolicy if they are needed.

Parameters

<i>input</i>	Input to map.
<i>dimension</i>	Dimension to map for.

39.237.3.3 MapString()

```
T MapString (
    const InputType & input,
    const size_t dimension )
```

Given the input and the dimension to which it belongs, return its numeric mapping.

If no mapping yet exists, the input is added to the list of mappings for the given dimension. The dimension parameter refers to the index of the dimension of the string (i.e. the row in the dataset).

Template Parameters

<i>T</i>	Numeric type to map to (int/double/float/etc.).
----------	---

Parameters

<i>input</i>	Input to find/create mapping for.
<i>dimension</i>	Index of the dimension of the string.

Referenced by MockCategoricalData(), and mlpack::util::SetParamWithInfo().

39.237.3.4 NumMappings()

```
size_t NumMappings (
    const size_t dimension ) const
```

Get the number of mappings for a particular dimension.

If the dimension is numeric, then this will return 0.

39.237.3.5 NumUnmappings()

```
size_t NumUnmappings (
    const T value,
    const size_t dimension ) const
```

Get the number of possible unmappings for a string in a given dimension.

39.237.3.6 Policy() [1/3]

```
const PolicyType& Policy ( ) const
```

Return the policy of the mapper.

Referenced by DatasetMapper< mpack::data::IncrementPolicy, double >::serialize().

39.237.3.7 Policy() [2/3]

```
PolicyType& Policy ( )
```

Modify the policy of the mapper (be careful!).

39.237.3.8 Policy() [3/3]

```
void Policy (
    PolicyType && policy )
```

Modify (Replace) the policy of the mapper with a new policy.

39.237.3.9 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the dataset information.

Definition at line 154 of file dataset_mapper.hpp.

39.237.3.10 SetDimensionality()

```
void SetDimensionality (
    const size_t dimensionality )
```

Set the dimensionality of an existing **DatasetMapper** (p. 1172) object.

This resets all mappings (but not the PolicyType).

Parameters

<i>dimensionality</i>	New dimensionality.
-----------------------	---------------------

Referenced by LoadCSV::GetTransposeMatrixSize().

39.237.3.11 Type() [1/2]

```
Datatype Type (
    const size_t dimension ) const
```

Return the type of a given dimension (numeric or categorical).

Referenced by MockCategoricalData(), and mlpack::util::SetParamWithInfo().

39.237.3.12 Type() [2/2]

```
Datatype& Type (
    const size_t dimension )
```

Modify the type of a given dimension (be careful!).

39.237.3.13 UnmapString()

```
const InputType& UnmapString (
    const T value,
    const size_t dimension,
    const size_t unmappingIndex = 0 ) const
```

Return the input that corresponds to a given value in a given dimension.

If the value is not a valid mapping in the given dimension, a `std::invalid_argument` is thrown. Note that this does not remove the mapping.

If the mapping is non-unique (i.e. many strings can map to the same value), then you can pass a different value for `unmappingIndex` to get a different string that maps to the given value. `unmappingIndex` should be in the range from 0 to `(NumUnmappings(value, dimension) - 1)`.

If the mapping is unique (which it is for `DatasetInfo`), then the `unmappingIndex` parameter can be left as the default.

Parameters

<i>value</i>	Mapped value for input.
<i>dimension</i>	Dimension to unmap string from.
<i>unmappingIndex</i>	Index of non-unique unmapping (optional).

39.237.3.14 UnmapValue()

```
PolicyType::MappedType UnmapValue (
    const InputType & input,
    const size_t dimension )
```

Return the value that corresponds to a given input in a given dimension.

If the value is not a valid mapping in the given dimension, a `std::invalid_argument` is thrown. Note that this does not remove the mapping.

Parameters

<i>input</i>	Mapped input for value.
<i>dimension</i>	Dimension to unmap input from.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ dataset_mapper.hpp`

39.238 HasSerialize< T > Struct Template Reference

Classes

- struct **check**

Public Types

- typedef char **no**[2]
- typedef char **yes**[1]

Static Public Member Functions

- template<typename U >
static **yes** & **chk** (**check**< U, typename **std::enable_if_t**< std::is_class< U >:: **value** > *, typename **std::enable_if_t**< **HasSerializeFunction**< U >:: **value** > *> *)
- template<typename >
static **no** & **chk** (...)

Static Public Attributes

- static const bool **value** = (sizeof(**chk**<T>(0)) == sizeof(**yes**))

39.238.1 Detailed Description

```
template<typename T>
struct mlpack::data::HasSerialize< T >
```

Definition at line 45 of file has_serialize.hpp.

39.238.2 Member Typedef Documentation

39.238.2.1 no

```
typedef char no[2]
```

Definition at line 49 of file has_serialize.hpp.

39.238.2.2 yes

```
typedef char yes[1]
```

Definition at line 48 of file has_serialize.hpp.

39.238.3 Member Function Documentation

39.238.3.1 chk() [1/2]

```
static yes& chk (
    check< U, typename std::enable_if_t< std::is_class< U >:: value > *, typename
std::enable_if_t< HasSerializeFunction< U >:: value > * > * ) [static]
```

39.238.3.2 `chk()` [2/2]

```
static no& chk (
    ... ) [static]
```

39.238.4 Member Data Documentation**39.238.4.1** `value`

```
const bool value = (sizeof( chk<T>(0)) == sizeof( yes)) [static]
```

Definition at line 57 of file `has_serialize.hpp`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ has_serialize.hpp`

39.239 `HasSerialize< T >::check< U, V, W >` Struct Template Reference**39.239.1** Detailed Description

```
template<typename T>
template<typename U, typename V, typename W>
struct mlpack::data::HasSerialize< T >::check< U, V, W >
```

Definition at line 50 of file `has_serialize.hpp`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ has_serialize.hpp`

39.240 `HasSerializeFunction< T >` Struct Template Reference**Public Types**

- `template<typename C >`
using **NonStaticSerialize** = `void(C::*)(boost::archive::xml_oarchive &, const unsigned int)`
- `template<typename >`
using **StaticSerialize** = `void(*)(boost::archive::xml_oarchive &, const unsigned int)`

Static Public Attributes

- static const bool **value**

39.240.1 Detailed Description

```
template<typename T>
struct mlpack::data::HasSerializeFunction< T >
```

Definition at line 31 of file has_serialize.hpp.

39.240.2 Member Typedef Documentation

39.240.2.1 NonStaticSerialize

```
using NonStaticSerialize = void(C::*)(boost::archive::xml_oarchive&, const unsigned int)
```

Definition at line 35 of file has_serialize.hpp.

39.240.2.2 StaticSerialize

```
using StaticSerialize = void(*) (boost::archive::xml_oarchive&, const unsigned int)
```

Definition at line 38 of file has_serialize.hpp.

39.240.3 Member Data Documentation

39.240.3.1 value

```
const bool value [static]
```

Initial value:

```
= HasSerializeCheck<T, NonStaticSerialize>::value ||
  HasSerializeCheck<T, StaticSerialize>::value
```

Definition at line 40 of file has_serialize.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ **has_serialize.hpp**

39.241 `Imputer< T, MapperType, StrategyType >` Class Template Reference

Given a dataset of a particular datatype, replace user-specified missing value with a variable dependent on the StrategyType and MapperType.

Public Member Functions

- **Imputer** (MapperType mapper, bool columnMajor=true)
- **Imputer** (MapperType mapper, StrategyType strategy, bool columnMajor=true)
- void **Impute** (arma::Mat< T > &input, const std::string &missingValue, const size_t dimension)
Given an input dataset, replace missing values of a dimension with given imputation strategy.
- const MapperType & **Mapper** () const
Get the mapper.
- MapperType & **Mapper** ()
Modify the given mapper.
- const StrategyType & **Strategy** () const
Get the strategy.
- StrategyType & **Strategy** ()
Modify the given strategy.

39.241.1 Detailed Description

```
template<typename T, typename MapperType, typename StrategyType>
class mlpack::data::Imputer< T, MapperType, StrategyType >
```

Given a dataset of a particular datatype, replace user-specified missing value with a variable dependent on the StrategyType and MapperType.

Template Parameters

<i>T</i>	Type of armadillo matrix used for imputation strategy.
<i>MapperType</i>	DatasetMapper (p. 1172) that is used to hold dataset information.
<i>StrategyType</i>	Imputation strategy used.

Definition at line 33 of file imputer.hpp.

39.241.2 Constructor & Destructor Documentation

39.241.2.1 Imputer() [1/2]

```

Imputer (
    MapperType mapper,
    bool columnMajor = true ) [inline]

```

Definition at line 36 of file imputer.hpp.

39.241.2.2 Imputer() [2/2]

```

Imputer (
    MapperType mapper,
    StrategyType strategy,
    bool columnMajor = true ) [inline]

```

Definition at line 43 of file imputer.hpp.

39.241.3 Member Function Documentation

39.241.3.1 Impute()

```

void Impute (
    arma::Mat< T > & input,
    const std::string & missingValue,
    const size_t dimension ) [inline]

```

Given an input dataset, replace missing values of a dimension with given imputation strategy.

This function does not produce output matrix, but overwrites the result into the input matrix.

Parameters

<i>input</i>	Input dataset to apply imputation. missingValue User defined missing value; it can be anything.
<i>dimension</i>	Dimension to apply the imputation.

Definition at line 60 of file imputer.hpp.

39.241.3.2 Mapper() [1/2]

```

const MapperType& Mapper ( ) const [inline]

```

Get the mapper.

Definition at line 75 of file imputer.hpp.

39.241.3.3 Mapper() [2/2]

```
MapperType& Mapper ( ) [inline]
```

Modify the given mapper.

Definition at line 78 of file imputer.hpp.

39.241.3.4 Strategy() [1/2]

```
const StrategyType& Strategy ( ) const [inline]
```

Get the strategy.

Definition at line 69 of file imputer.hpp.

39.241.3.5 Strategy() [2/2]

```
StrategyType& Strategy ( ) [inline]
```

Modify the given strategy.

Definition at line 72 of file imputer.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ **imputer.hpp**

39.242 IncrementPolicy Class Reference

IncrementPolicy (p. 1184) is used as a helper class for **DatasetMapper** (p. 1172).

Public Types

- using **MappedType** = size_t

Public Member Functions

- **IncrementPolicy** (const bool forceAllMappings=false)
- template<typename T , typename InputType >
void **MapFirstPass** (const InputType &input, const size_t dim, std::vector< **Datatype** > &types)
Determine if the dimension is numeric or categorical.
- template<typename MapType , typename T , typename InputType >
T **MapString** (const InputType &input, const size_t dimension, MapType &maps, std::vector< **Datatype** > &types)
*Given the input and the dimension to which the it belongs, and the maps and types given by the **DatasetMapper** (p. 1172) class, returns its numeric mapping.*

Static Public Attributes

- static const bool **NeedsFirstPass** = true
We do need a first pass over the data to set the dimension types right.

39.242.1 Detailed Description

IncrementPolicy (p. 1184) is used as a helper class for **DatasetMapper** (p. 1172).

It tells how the strings should be mapped. Purpose of this policy is to map all dimension if one of the variables in a dimension turns out to be a categorical variable. **IncrementPolicy** (p. 1184) maps strings to incrementing unsigned integers (size_t). The first input to be mapped will be mapped to 0, the next to 1 and so on.

If the 'forceAllMappings' parameter is set to true, this will always map. Otherwise, inputs will only be mapped if they cannot be cast to the output type via a stringstream extraction.

Definition at line 33 of file increment_policy.hpp.

39.242.2 Member Typedef Documentation

39.242.2.1 MappedType

```
using MappedType = size_t
```

Definition at line 40 of file increment_policy.hpp.

39.242.3 Constructor & Destructor Documentation

39.242.3.1 IncrementPolicy()

```
IncrementPolicy (
    const bool forceAllMappings = false ) [inline]
```

Definition at line 36 of file `increment_policy.hpp`.

39.242.4 Member Function Documentation

39.242.4.1 MapFirstPass()

```
void MapFirstPass (
    const InputType & input,
    const size_t dim,
    std::vector< Datatype > & types ) [inline]
```

Determine if the dimension is numeric or categorical.

Definition at line 49 of file `increment_policy.hpp`.

References `mlpack::data::categorical`.

39.242.4.2 MapString()

```
T MapString (
    const InputType & input,
    const size_t dimension,
    MapType & maps,
    std::vector< Datatype > & types ) [inline]
```

Given the input and the dimension to which the it belongs, and the maps and types given by the **DatasetMapper** (p. 1172) class, returns its numeric mapping.

If no mapping yet exists, the input is added to the list of mappings for the given dimension. This function is used as a helper function for **DatasetMapper** (p. 1172) class.

Template Parameters

<i>MapType</i>	Type of <code>unordered_map</code> that contains mapped value pairs
----------------	---

Parameters

<i>input</i>	Input to find/create mapping for.
<i>dimension</i>	Index of the dimension of the input.
<i>maps</i>	Unordered map given by the DatasetMapper (p. 1172).
<i>types</i>	Vector containing the type information about each dimensions.

Definition at line 90 of file increment_policy.hpp.

References `mlpack::data::categorical`, and `mlpack::data::numeric`.

39.242.5 Member Data Documentation

39.242.5.1 NeedsFirstPass

```
const bool NeedsFirstPass = true [static]
```

We do need a first pass over the data to set the dimension types right.

Definition at line 43 of file increment_policy.hpp.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/map_policies/ increment_policy.hpp`

39.243 ListwiseDeletion< T > Class Template Reference

A complete-case analysis to remove the values containing mappedValue.

Public Member Functions

- void **Impute** (`arma::Mat< T > &input`, `const T &mappedValue`, `const size_t dimension`, `const bool column↔Major=true`)

Impute function searches through the input looking for mappedValue and remove the whole row or column.

39.243.1 Detailed Description

```
template<typename T>
class mlpack::data::ListwiseDeletion< T >
```

A complete-case analysis to remove the values containing mappedValue.

Removes all data for a case that has one or more missing values.

Template Parameters

<i>T</i>	Type of armadillo matrix
----------	--------------------------

Definition at line 25 of file listwise_deletion.hpp.

39.243.2 Member Function Documentation

39.243.2.1 Impute()

```
void Impute (
    arma::Mat< T > & input,
    const T & mappedValue,
    const size_t dimension,
    const bool columnMajor = true ) [inline]
```

Impute function searches through the input looking for mappedValue and remove the whole row or column.

The result is overwritten to the input.

Parameters

<i>input</i>	Matrix that contains mappedValue.
<i>mappedValue</i>	Value that the user wants to get rid of.
<i>dimension</i>	Index of the dimension of the mappedValue.
<i>columnMajor</i>	State of whether the input matrix is columnMajor or not.

Definition at line 37 of file listwise_deletion.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/ **listwise_deletion.hpp**

39.244 LoadCSV Class Reference

Load the csv file. This class use boost::spirit to implement the parser, please refer to following link <http://theboostcpplibraries.com/boost.spirit> for quick review.

Public Member Functions

- **LoadCSV** (const std::string &file)
Construct the **LoadCSV** (p. 1188) object on the given file.
- template<typename T , typename MapPolicy >
void **GetMatrixSize** (size_t &rows, size_t &cols, **DatasetMapper**< MapPolicy > &info)
Peek at the file to determine the number of rows and columns in the matrix, assuming a non-transposed matrix.
- template<typename T , typename MapPolicy >
void **GetTransposeMatrixSize** (size_t &rows, size_t &cols, **DatasetMapper**< MapPolicy > &info)
Peek at the file to determine the number of rows and columns in the matrix, assuming a transposed matrix.
- template<typename T , typename PolicyType >
void **Load** (arma::Mat< T > &inout, **DatasetMapper**< PolicyType > &infoSet, const bool transpose=true)
Load the file into the given matrix with the given **DatasetMapper** (p. 1172) object.

39.244.1 Detailed Description

Load the csv file. This class uses boost::spirit to implement the parser, please refer to following link <http://theboostcpplibraries.com/boost.spirit> for quick review.

Definition at line 36 of file load_csv.hpp.

39.244.2 Constructor & Destructor Documentation

39.244.2.1 LoadCSV()

```
LoadCSV (
    const std::string & file )
```

Construct the **LoadCSV** (p. 1188) object on the given file.

This will construct the rules necessary for loading and attempt to open the file.

39.244.3 Member Function Documentation

39.244.3.1 GetMatrixSize()

```
void GetMatrixSize (
    size_t & rows,
    size_t & cols,
    DatasetMapper< MapPolicy > & info ) [inline]
```

Peek at the file to determine the number of rows and columns in the matrix, assuming a non-transposed matrix.

This will also take a first pass over the data for **DatasetMapper** (p. 1172), if MapPolicy::NeedsFirstPass is true. The info object will be re-initialized with the correct dimensionality.

Parameters

<i>rows</i>	Variable to be filled with the number of rows.
<i>cols</i>	Variable to be filled with the number of columns.
<i>info</i>	DatasetMapper (p. 1172) object to use for first pass.

Definition at line 78 of file load_csv.hpp.

39.244.3.2 GetTransposeMatrixSize()

```
void GetTransposeMatrixSize (
    size_t & rows,
    size_t & cols,
    DatasetMapper< MapPolicy > & info ) [inline]
```

Peek at the file to determine the number of rows and columns in the matrix, assuming a transposed matrix.

This will also take a first pass over the data for **DatasetMapper** (p. 1172), if MapPolicy::NeedsFirstPass is true. The info object will be re-initialized with the correct dimensionality.

Parameters

<i>rows</i>	Variable to be filled with the number of rows.
<i>cols</i>	Variable to be filled with the number of columns.
<i>info</i>	DatasetMapper (p. 1172) object to use for first pass.

Definition at line 151 of file load_csv.hpp.

References DatasetMapper< PolicyType, InputType >::SetDimensionality().

39.244.3.3 Load()

```
void Load (
    arma::Mat< T > & inout,
    DatasetMapper< PolicyType > & infoSet,
    const bool transpose = true ) [inline]
```

Load the file into the given matrix with the given **DatasetMapper** (p. 1172) object.

Throws exceptions on errors.

Parameters

<i>inout</i>	Matrix to load into.
<i>infoSet</i>	DatasetMapper (p. 1172) to use while loading.
<i>transpose</i>	If true, the matrix should be transposed on loading (default).

Definition at line 55 of file load_csv.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/ **load_csv.hpp**

39.245 MeanImputation< T > Class Template Reference

A simple mean imputation class.

Public Member Functions

- void **Impute** (arma::Mat< T > &input, const T &mappedValue, const size_t dimension, const bool column↔Major=true)

Impute function searches through the input looking for mappedValue and replaces it with the mean of the given dimension.

39.245.1 Detailed Description

```
template<typename T>
class mlpack::data::MeanImputation< T >
```

A simple mean imputation class.

Template Parameters

<i>T</i>	Type of armadillo matrix
----------	--------------------------

Definition at line 24 of file mean_imputation.hpp.

39.245.2 Member Function Documentation

39.245.2.1 Impute()

```
void Impute (
    arma::Mat< T > & input,
    const T & mappedValue,
    const size_t dimension,
    const bool columnMajor = true ) [inline]
```

Impute function searches through the input looking for mappedValue and replaces it with the mean of the given dimension.

The result is overwritten to the input matrix.

Parameters

<i>input</i>	Matrix that contains mappedValue.
<i>mappedValue</i>	Value that the user wants to get rid of.
<i>dimension</i>	Index of the dimension of the mappedValue.
<i>columnMajor</i>	State of whether the input matrix is columnMajor or not.

Definition at line 37 of file mean_imputation.hpp.

References Log::Fatal.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/ **mean_imputation.hpp**

39.246 MedianImputation< T > Class Template Reference

This is a class implementation of simple median imputation.

Public Member Functions

- void **Impute** (arma::Mat< T > &input, const T &mappedValue, const size_t dimension, const bool columnMajor=true)
Impute function searches through the input looking for mappedValue and replaces it with the median of the given dimension.

39.246.1 Detailed Description

```
template<typename T>
class mlpack::data::MedianImputation< T >
```

This is a class implementation of simple median imputation.

replace missing value with middle or average of middle values

Template Parameters

<i>T</i>	Type of armadillo matrix
----------	--------------------------

Definition at line 25 of file median_imputation.hpp.

39.246.2 Member Function Documentation

39.246.2.1 Impute()

```
void Impute (
    arma::Mat< T > & input,
    const T & mappedValue,
    const size_t dimension,
    const bool columnMajor = true ) [inline]
```

Impute function searches through the input looking for mappedValue and replaces it with the median of the given dimension.

The result is overwritten to the input matrix.

Parameters

<i>input</i>	Matrix that contains mappedValue.
<i>mappedValue</i>	Value that the user wants to get rid of.
<i>dimension</i>	Index of the dimension of the mappedValue.
<i>columnMajor</i>	State of whether the input matrix is columnMajor or not.

Definition at line 38 of file median_imputation.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/ **median_imputation.hpp**

39.247 MissingPolicy Class Reference

MissingPolicy (p. 1193) is used as a helper class for **DatasetMapper** (p. 1172).

Public Types

- using **MappedType** = double

Public Member Functions

- **MissingPolicy** ()
- **MissingPolicy** (`std::set< std::string > missingSet`)
*Create the **MissingPolicy** (p. 1193) object with the given **missingSet**.*
- `template<typename T >`
`void MapFirstPass (const std::string &, const size_t)`
*There is nothing for us to do here, but this is required by the **MapPolicy** type.*
- `template<typename MapType , typename T >`
`T MapString (const std::string &string, const size_t dimension, MapType &maps, std::vector< Datatype > &)`
*Given the string and the dimension to which it belongs by the user, and the maps and types given by the **DatasetMapper** (p. 1172) class, returns its numeric mapping.*

Static Public Attributes

- static const bool **NeedsFirstPass** = false
This doesn't need a first pass over the data to set up.

39.247.1 Detailed Description

MissingPolicy (p. 1193) is used as a helper class for **DatasetMapper** (p. 1172).

It tells how the strings should be mapped. Purpose of this policy is to map all user-defined missing variables into maps so that users can decide what to do with the corrupted data. User-defined missing variables are given by the **missingSet**. Note that **MissingPolicy** (p. 1193) does not change type of features.

Definition at line 30 of file `missing_policy.hpp`.

39.247.2 Member Typedef Documentation

39.247.2.1 MappedType

```
using MappedType = double
```

Definition at line 34 of file `missing_policy.hpp`.

39.247.3 Constructor & Destructor Documentation

39.247.3.1 MissingPolicy() [1/2]

```
MissingPolicy ( ) [inline]
```

Definition at line 36 of file missing_policy.hpp.

39.247.3.2 MissingPolicy() [2/2]

```
MissingPolicy (  
    std::set< std::string > missingSet ) [inline], [explicit]
```

Create the **MissingPolicy** (p. 1193) object with the given missingSet.

Note that the missingSet cannot be changed later; you will have to create a new **MissingPolicy** (p. 1193) object.

Parameters

<i>missingSet</i>	Set of strings that should be mapped.
-------------------	---------------------------------------

Definition at line 48 of file missing_policy.hpp.

39.247.4 Member Function Documentation

39.247.4.1 MapFirstPass()

```
void MapFirstPass (  
    const std::string & ,  
    const size_t ) [inline]
```

There is nothing for us to do here, but this is required by the MapPolicy type.

Definition at line 62 of file missing_policy.hpp.

39.247.4.2 MapString()

```
T MapString (
    const std::string & string,
    const size_t dimension,
    MapType & maps,
    std::vector< Datatype > & ) [inline]
```

Given the string and the dimension to which it belongs by the user, and the maps and types given by the **DatasetMapper** (p. 1172) class, returns its numeric mapping.

If no mapping yet exists and the string is included in the missingSet, the string is added to the list of mappings for the given dimension. This function is used as a helper function for **DatasetMapper** (p. 1172) class.

Template Parameters

<i>MapType</i>	Type of unordered_map that contains mapped value pairs
----------------	--

Parameters

<i>string</i>	String to find/create mapping for.
<i>dimension</i>	Index of the dimension of the string.
<i>maps</i>	Unordered map given by the DatasetMapper (p. 1172).
<i>types</i>	Vector containing the type information about each dimensions.

Definition at line 82 of file missing_policy.hpp.

39.247.5 Member Data Documentation

39.247.5.1 NeedsFirstPass

```
const bool NeedsFirstPass = false [static]
```

This doesn't need a first pass over the data to set up.

Definition at line 55 of file missing_policy.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/map_policies/ **missing_policy.hpp**

39.248 DBSCAN< RangeSearchType, PointSelectionPolicy > Class Template Reference

DBSCAN (p. 1197) (Density-Based Spatial Clustering of Applications with Noise) is a clustering technique described in the following paper:

Public Member Functions

- **DBSCAN** (const double epsilon, const size_t minPoints, const bool batchMode=true, RangeSearchType range↔ Search=RangeSearchType(), PointSelectionPolicy pointSelector=PointSelectionPolicy())
*Construct the **DBSCAN** (p. 1197) object with the given parameters.*
- template<typename MatType >
size_t **Cluster** (const MatType &data, arma::mat ¢roids)
*Performs **DBSCAN** (p. 1197) clustering on the data, returning number of clusters and also the centroid of each cluster.*
- template<typename MatType >
size_t **Cluster** (const MatType &data, arma::Row< size_t > &assignments)
*Performs **DBSCAN** (p. 1197) clustering on the data, returning number of clusters and also the list of cluster assignments.*
- template<typename MatType >
size_t **Cluster** (const MatType &data, arma::Row< size_t > &assignments, arma::mat ¢roids)
*Performs **DBSCAN** (p. 1197) clustering on the data, returning number of clusters, the centroid of each cluster and also the list of cluster assignments.*

39.248.1 Detailed Description

```
template<typename RangeSearchType = range::RangeSearch<>, typename PointSelectionPolicy = OrderedPointSelection>
class mlpack::dbscan::DBSCAN< RangeSearchType, PointSelectionPolicy >
```

DBSCAN (p. 1197) (Density-Based Spatial Clustering of Applications with Noise) is a clustering technique described in the following paper:

```
@inproceedings{ester1996density,
  title={A density-based algorithm for discovering clusters in large spatial
    databases with noise.},
  author={Ester, M. and Kriegel, H.-P. and Sander, J. and Xu, X.},
  booktitle={Proceedings of the Second International Conference on Knowledge
    Discovery and Data Mining (KDD '96)},
  pages={226--231},
  year={1996}
}
```

The **DBSCAN** (p. 1197) algorithm iteratively clusters points using range searches with a specified radius parameter. This implementation allows configuration of the range search technique used and the point selection strategy by means of template parameters.

Template Parameters

<i>RangeSearchType</i>	Class to use for range searching.
<i>PointSelectionPolicy</i>	Strategy for selecting next point to cluster with.

Definition at line 53 of file dbscan.hpp.

39.248.2 Constructor & Destructor Documentation

39.248.2.1 DBSCAN()

```
DBSCAN (
    const double epsilon,
    const size_t minPoints,
    const bool batchMode = true,
    RangeSearchType rangeSearch = RangeSearchType(),
    PointSelectionPolicy pointSelector = PointSelectionPolicy() )
```

Construct the **DBSCAN** (p. 1197) object with the given parameters.

The batchMode parameter should be set to false in the case where RAM issues will be encountered (i.e. if the dataset is very large or if epsilon is large). When batchMode is false, each point will be searched iteratively, which could be slower but will use less memory.

Parameters

<i>epsilon</i>	Size of range query.
<i>minPoints</i>	Minimum number of points for each cluster.
<i>batchMode</i>	If true, all points are searched in batch.
<i>rangeSearch</i>	Optional instantiated RangeSearch object.
<i>pointSelector</i>	OptionL instantiated PointSelectionPolicy object.

39.248.3 Member Function Documentation

39.248.3.1 Cluster() [1/3]

```
size_t Cluster (
    const MatType & data,
    arma::mat & centroids )
```

Performs **DBSCAN** (p. 1197) clustering on the data, returning number of clusters and also the centroid of each cluster.

Template Parameters

<i>MatType</i>	Type of matrix (arma::mat or arma::sp_mat).
----------------	---

Parameters

<i>data</i>	Dataset to cluster.
<i>centroids</i>	Matrix in which centroids are stored.

39.248.3.2 Cluster() [2/3]

```
size_t Cluster (
    const MatType & data,
    arma::Row< size_t > & assignments )
```

Performs **DBSCAN** (p. 1197) clustering on the data, returning number of clusters and also the list of cluster assignments.

If assignments[i] == SIZE_MAX, then the point is considered "noise".

Template Parameters

<i>MatType</i>	Type of matrix (arma::mat or arma::sp_mat).
----------------	---

Parameters

<i>data</i>	Dataset to cluster.
<i>assignments</i>	Vector to store cluster assignments.

39.248.3.3 Cluster() [3/3]

```
size_t Cluster (
    const MatType & data,
    arma::Row< size_t > & assignments,
    arma::mat & centroids )
```

Performs **DBSCAN** (p. 1197) clustering on the data, returning number of clusters, the centroid of each cluster and also the list of cluster assignments.

If assignments[i] == SIZE_MAX, then the point is considered "noise".

Template Parameters

<i>MatType</i>	Type of matrix (arma::mat or arma::sp_mat).
----------------	---

Parameters

<i>data</i>	Dataset to cluster.
<i>assignments</i>	Vector to store cluster assignments.
<i>centroids</i>	Matrix in which centroids are stored.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/dbscan/ **dbscan.hpp**

39.249 OrderedPointSelection Class Reference

This class can be used to sequentially select the next point to use for **DBSCAN** (p. 1197).

Static Public Member Functions

- template<typename MatType >
static size_t **Select** (const size_t point, const MatType &)
Select the next point to use, sequentially.

39.249.1 Detailed Description

This class can be used to sequentially select the next point to use for **DBSCAN** (p. 1197).

Definition at line 23 of file `ordered_point_selection.hpp`.

39.249.2 Member Function Documentation

39.249.2.1 Select()

```
static size_t Select (
    const size_t point,
    const MatType & ) [inline], [static]
```

Select the next point to use, sequentially.

Parameters

<i>unvisited</i>	Bitset indicating which points are unvisited.
<i>data</i>	Unused data.

Definition at line 33 of file `ordered_point_selection.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/dbscan/ ordered_point_selection.hpp`

39.250 RandomPointSelection Class Reference

This class can be used to randomly select the next point to use for **DBSCAN** (p. 1197).

Public Member Functions

- `template<typename MatType >`
`size_t Select (const size_t, const MatType &data)`
Select the next point to use, randomly.

39.250.1 Detailed Description

This class can be used to randomly select the next point to use for **DBSCAN** (p. 1197).

Definition at line 23 of file `random_point_selection.hpp`.

39.250.2 Member Function Documentation

39.250.2.1 Select()

```
size_t Select (
    const size_t ,
    const MatType & data ) [inline]
```

Select the next point to use, randomly.

Parameters

<i>point</i>	Unused data.
<i>data</i>	Dataset to cluster.

Definition at line 33 of file random_point_selection.hpp.

References `mlpack::math::RandInt()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/dbscan/ random_point_selection.hpp`

39.251 DecisionStump< MatType > Class Template Reference

This class implements a decision stump.

Public Member Functions

- **DecisionStump** (const MatType &data, const arma::Row< size_t > &labels, const size_t numClasses, const size_t bucketSize=10)
Constructor.
- **DecisionStump** (const **DecisionStump**<> &other, const MatType &data, const arma::Row< size_t > &labels, const size_t numClasses, const arma::rowvec &weights)
Alternate constructor which copies the parameters bucketSize and classes from an already initiated decision stump, other.
- **DecisionStump** ()
Create a decision stump without training.
- const arma::Col< size_t > **BinLabels** () const
Access the labels for each split bin.
- arma::Col< size_t > & **BinLabels** ()
Modify the labels for each split bin (be careful!).
- void **Classify** (const MatType &test, arma::Row< size_t > &predictedLabels)

Classification function.

- `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`
Serialize the decision stump.
- `const arma::vec & Split () const`
Access the splitting values.
- `arma::vec & Split ()`
Modify the splitting values (be careful!).
- `size_t SplitDimension () const`
Access the splitting dimension.
- `size_t & SplitDimension ()`
Modify the splitting dimension (be careful!).
- `double Train (const MatType &data, const arma::Row< size_t > &labels, const size_t numClasses, const size_t bucketSize)`
Train the decision stump on the given data.
- `double Train (const MatType &data, const arma::Row< size_t > &labels, const arma::rowvec &weights, const size_t numClasses, const size_t bucketSize)`
Train the decision stump on the given data, with the given weights.

39.251.1 Detailed Description

```
template<typename MatType = arma::mat>
class mlpack::decision_stump::DecisionStump< MatType >
```

This class implements a decision stump.

It constructs a single level decision tree, i.e., a decision stump. It uses entropy to decide splitting ranges.

The stump is parameterized by a splitting dimension (the dimension on which points are split), a vector of bin split values, and a vector of labels for each bin. Bin *i* is specified by the range `[split[i], split[i + 1])`. The last bin has range up to `(split[i + 1]` does not exist in that case). Points that are below the first bin will take the label of the first bin.

Template Parameters

<i>MatType</i>	Type of matrix that is being used (sparse or dense).
----------------	--

Definition at line 34 of file `decision_stump.hpp`.

39.251.2 Constructor & Destructor Documentation

39.251.2.1 DecisionStump() [1/3]

```
DecisionStump (
    const MatType & data,
```

```
const arma::Row< size_t > & labels,
const size_t numClasses,
const size_t bucketSize = 10 )
```

Constructor.

Train on the provided data. Generate a decision stump from data.

Parameters

<i>data</i>	Input, training data.
<i>labels</i>	Labels of training data.
<i>numClasses</i>	Number of distinct classes in labels.
<i>bucketSize</i>	Minimum size of bucket when splitting.

39.251.2.2 DecisionStump() [2/3]

```
DecisionStump (
    const DecisionStump<> & other,
    const MatType & data,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const arma::rowvec & weights )
```

Alternate constructor which copies the parameters bucketSize and classes from an already initiated decision stump, other.

It appropriately sets the weight vector.

Parameters

<i>other</i>	The other initiated Decision Stump object from which we copy the values.
<i>data</i>	The data on which to train this object on.
<i>labels</i>	The labels of data.
<i>weights</i>	Weight vector to use while training. For boosting purposes.

39.251.2.3 DecisionStump() [3/3]

```
DecisionStump ( )
```

Create a decision stump without training.

This stump will not be useful and will always return a class of 0 for anything that is to be classified, so it would be a prudent idea to call **Train()** (p. 1206) after using this constructor.

39.251.3 Member Function Documentation

39.251.3.1 BinLabels() [1/2]

```
const arma::Col<size_t> BinLabels ( ) const [inline]
```

Access the labels for each split bin.

Definition at line 130 of file decision_stump.hpp.

39.251.3.2 BinLabels() [2/2]

```
arma::Col<size_t>& BinLabels ( ) [inline]
```

Modify the labels for each split bin (be careful!).

Definition at line 132 of file decision_stump.hpp.

References DecisionStump< MatType >::serialize(), and DecisionStump< MatType >::Train().

39.251.3.3 Classify()

```
void Classify (
    const MatType & test,
    arma::Row< size_t > & predictedLabels )
```

Classification function.

After training, classify test, and put the predicted classes in predictedLabels.

Parameters

<i>test</i>	Testing data or data to classify.
<i>predictedLabels</i>	Vector to store the predicted classes after classifying test data.

39.251.3.4 serialize()

```
void serialize (
```

```
Archive & ar,
const unsigned int )
```

Serialize the decision stump.

Referenced by `DecisionStump< MatType >::BinLabels()`.

39.251.3.5 Split() [1/2]

```
const arma::vec& Split ( ) const [inline]
```

Access the splitting values.

Definition at line 125 of file `decision_stump.hpp`.

39.251.3.6 Split() [2/2]

```
arma::vec& Split ( ) [inline]
```

Modify the splitting values (be careful!).

Definition at line 127 of file `decision_stump.hpp`.

39.251.3.7 SplitDimension() [1/2]

```
size_t SplitDimension ( ) const [inline]
```

Access the splitting dimension.

Definition at line 120 of file `decision_stump.hpp`.

39.251.3.8 SplitDimension() [2/2]

```
size_t& SplitDimension ( ) [inline]
```

Modify the splitting dimension (be careful!).

Definition at line 122 of file `decision_stump.hpp`.

39.251.3.9 Train() [1/2]

```
double Train (
    const MatType & data,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const size_t bucketSize )
```

Train the decision stump on the given data.

This completely overwrites any previous training data, so after training the stump may be completely different.

Parameters

<i>data</i>	Dataset to train on.
<i>labels</i>	Labels for each point in the dataset.
<i>numClasses</i>	Number of classes in the dataset.
<i>bucketSize</i>	Minimum size of bucket when splitting.

Returns

The final entropy after splitting.

Referenced by DecisionStump< MatType >::BinLabels().

39.251.3.10 Train() [2/2]

```
double Train (
    const MatType & data,
    const arma::Row< size_t > & labels,
    const arma::rowvec & weights,
    const size_t numClasses,
    const size_t bucketSize )
```

Train the decision stump on the given data, with the given weights.

This completely overwrites any previous training data, so after training the stump may be completely different.

Parameters

<i>data</i>	Dataset to train on.
<i>labels</i>	Labels for each point in the dataset.
<i>weights</i>	Weights for each point in the dataset.
<i>numClasses</i>	Number of classes in the dataset.
<i>bucketSize</i>	Minimum size of bucket when splitting.

Returns

The final entropy after splitting.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_stump/ **decision_stump.hpp**

39.252 DTree< MatType, TagType > Class Template Reference

A density estimation tree is similar to both a decision tree and a space partitioning tree (like a kd-tree).

Public Types

- `typedef MatType::elem_type ElemType`
The actual, underlying type we're working with.
- `typedef arma::Col< ElemType > StatType`
The statistic type we are holding.
- `typedef MatType::vec_type VecType`
The type of vector we are using.

Public Member Functions

- **DTree** ()
Create an empty density estimation tree.
- **DTree** (const **DTree** &obj)
Create a tree that is the copy of the given tree.
- **DTree** (**DTree** &&obj)
Create a tree by taking ownership of another tree (move constructor).
- **DTree** (const **StatType** &maxVals, const **StatType** &minVals, const size_t totalPoints)
Create a density estimation tree with the given bounds and the given number of total points.
- **DTree** (MatType &data)
Create a density estimation tree on the given data.
- **DTree** (const **StatType** &maxVals, const **StatType** &minVals, const size_t start, const size_t end, const double logNegError)
Create a child node of a density estimation tree given the bounding box specified by maxVals and minVals, using the size given in start and end and the specified error.
- **DTree** (const **StatType** &maxVals, const **StatType** &minVals, const size_t totalPoints, const size_t start, const size_t end)
Create a child node of a density estimation tree given the bounding box specified by maxVals and minVals, using the size given in start and end, and calculating the error with the total number of points given.
- **~DTree** ()
Clean up memory allocated by the tree.
- double **AlphaUpper** () const
Return the upper part of the alpha sum.
- TagType **BucketTag** () const
Return the current bucket's ID, if leaf, or -1 otherwise.
- **DTree** & **Child** (const size_t child) const
Return the specified child (0 will be left, 1 will be right).
- **DTree** *& **ChildPtr** (const size_t child)
- double **ComputeValue** (const **VecType** &query) const
Compute the logarithm of the density estimate of a given query point.
- void **ComputeVariableImportance** (arma::vec &importances) const
Compute the variable importance of each dimension in the learned tree.
- size_t **End** () const
Return the first index of a point not contained in this node.
- TagType **FindBucket** (const **VecType** &query) const
Return the tag of the leaf containing the query.

- double **Grow** (MatType &data, arma::Col< size_t > &oldFromNew, const bool useVolReg=false, const size_t maxLeafSize=10, const size_t minLeafSize=5)
Greedily expand the tree.
- **DTree * Left** () const
Return the left child.
- double **LogNegativeError** (const size_t totalPoints) const
Compute the log-negative-error for this point, given the total number of points in the dataset.
- double **LogNegError** () const
Return the log negative error of this node.
- double **LogVolume** () const
Return the inverse of the volume of this node.
- const **StatType & MaxVals** () const
Return the maximum values.
- const **StatType & MinVals** () const
Return the minimum values.
- size_t **NumChildren** () const
Return the number of children in this node.
- **DTree & operator=** (const **DTree** &obj)
Copy the given tree.
- **DTree & operator=** (**DTree** &&obj)
Take ownership of the given tree (move operator).
- double **PruneAndUpdate** (const double oldAlpha, const size_t points, const bool useVolReg=false)
Perform alpha pruning on a tree.
- double **Ratio** () const
Return the ratio of points in this node to the points in the whole dataset.
- **DTree * Right** () const
Return the right child.
- bool **Root** () const
Return whether or not this is the root of the tree.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the density estimation tree.
- size_t **SplitDim** () const
Return the split dimension of this node.
- **ElemType SplitValue** () const
Return the split value of this node.
- size_t **Start** () const
Return the starting index of points contained in this node.
- size_t **SubtreeLeaves** () const
Return the number of leaves which are descendants of this node.
- double **SubtreeLeavesLogNegError** () const
Return the log negative error of all descendants of this node.
- TagType **TagTree** (const TagType &tag=0, bool everyNode=false)
*Index the buckets for possible usage later; this results in every leaf in the tree having a specific tag (accessible with **BucketTag()** (p. 1214)).*
- bool **WithinRange** (const **VecType** &query) const
Return whether a query point is within the range of this node.

39.252.1 Detailed Description

```
template<typename MatType = arma::mat, typename TagType = int>
class mlpack::det::DTree< MatType, TagType >
```

A density estimation tree is similar to both a decision tree and a space partitioning tree (like a kd-tree).

Each leaf represents a constant-density hyper-rectangle. The tree is constructed in such a way as to minimize the integrated square error between the probability distribution of the tree and the observed probability distribution of the data. Because the tree is similar to a decision tree, the density estimation tree can provide very fast density estimates for a given point.

For more information, see the following paper:

```
@incollection{ram2011,
  author = {Ram, Parikshit and Gray, Alexander G.},
  title = {Density estimation trees},
  booktitle = {{Proceedings of the 17th ACM SIGKDD International Conference
    on Knowledge Discovery and Data Mining}},
  series = {KDD '11},
  year = {2011},
  pages = {627--635}
}
```

Definition at line 46 of file dtree.hpp.

39.252.2 Member Typedef Documentation

39.252.2.1 ElemType

```
typedef MatType::elem_type ElemType
```

The actual, underlying type we're working with.

Definition at line 50 of file dtree.hpp.

39.252.2.2 StatType

```
typedef arma::Col< ElemType> StatType
```

The statistic type we are holding.

Definition at line 54 of file dtree.hpp.

39.252.2.3 VecType

```
typedef MatType::vec_type VecType
```

The type of vector we are using.

Definition at line 52 of file dtree.hpp.

39.252.3 Constructor & Destructor Documentation

39.252.3.1 DTree() [1/7]

```
DTree ( )
```

Create an empty density estimation tree.

39.252.3.2 DTree() [2/7]

```
DTree (
    const DTree< MatType, TagType > & obj )
```

Create a tree that is the copy of the given tree.

Parameters

<i>obj</i>	Tree to copy.
------------	---------------

39.252.3.3 DTree() [3/7]

```
DTree (
    DTree< MatType, TagType > && obj )
```

Create a tree by taking ownership of another tree (move constructor).

Parameters

<i>obj</i>	Tree to take ownership of.
------------	----------------------------

39.252.3.4 DTree() [4/7]

```

DTree (
    const StatType & maxVals,
    const StatType & minVals,
    const size_t totalPoints )

```

Create a density estimation tree with the given bounds and the given number of total points.

Children will not be created.

Parameters

<i>maxVals</i>	Maximum values of the bounding box.
<i>minVals</i>	Minimum values of the bounding box.
<i>totalPoints</i>	Total number of points in the dataset.

39.252.3.5 DTree() [5/7]

```

DTree (
    MatType & data )

```

Create a density estimation tree on the given data.

Children will be created following the procedure outlined in the paper. The data will be modified; it will be reordered similar to the way BinarySpaceTree modifies datasets.

Parameters

<i>data</i>	Dataset to build tree on.
-------------	---------------------------

39.252.3.6 DTree() [6/7]

```

DTree (
    const StatType & maxVals,
    const StatType & minVals,
    const size_t start,
    const size_t end,
    const double logNegError )

```

Create a child node of a density estimation tree given the bounding box specified by `maxVals` and `minVals`, using the size given in `start` and `end` and the specified error.

Children of this node will not be created recursively.

Parameters

<i>maxVals</i>	Upper bound of bounding box.
<i>minVals</i>	Lower bound of bounding box.
<i>start</i>	Start of points represented by this node in the data matrix.
<i>end</i>	End of points represented by this node in the data matrix.
<i>error</i>	log-negative error of this node.

39.252.3.7 DTree() [7/7]

```
DTree (
    const StatType & maxVals,
    const StatType & minVals,
    const size_t totalPoints,
    const size_t start,
    const size_t end )
```

Create a child node of a density estimation tree given the bounding box specified by `maxVals` and `minVals`, using the size given in `start` and `end`, and calculating the error with the total number of points given.

Children of this node will not be created recursively.

Parameters

<i>maxVals</i>	Upper bound of bounding box.
<i>minVals</i>	Lower bound of bounding box.
<i>start</i>	Start of points represented by this node in the data matrix.
<i>end</i>	End of points represented by this node in the data matrix.

39.252.3.8 ~DTree()

```
~ DTree ( )
```

Clean up memory allocated by the tree.

39.252.4 Member Function Documentation

39.252.4.1 AlphaUpper()

```
double AlphaUpper ( ) const [inline]
```

Return the upper part of the alpha sum.

Definition at line 306 of file dtree.hpp.

39.252.4.2 BucketTag()

```
TagType BucketTag ( ) const [inline]
```

Return the current bucket's ID, if leaf, or -1 otherwise.

Definition at line 308 of file dtree.hpp.

39.252.4.3 Child()

```
DTree& Child (
    const size_t child ) const [inline]
```

Return the specified child (0 will be left, 1 will be right).

If the index is greater than 1, this will return the right child.

Parameters

<i>child</i>	Index of child to return.
--------------	---------------------------

Definition at line 318 of file dtree.hpp.

39.252.4.4 ChildPtr()

```
DTree*& ChildPtr (
    const size_t child ) [inline]
```

Definition at line 320 of file dtree.hpp.

39.252.4.5 ComputeValue()

```
double ComputeValue (
    const VecType & query ) const
```

Compute the logarithm of the density estimate of a given query point.

Parameters

<i>query</i>	Point to estimate density of.
--------------	-------------------------------

39.252.4.6 ComputeVariableImportance()

```
void ComputeVariableImportance (
    arma::vec & importances ) const
```

Compute the variable importance of each dimension in the learned tree.

Parameters

<i>importances</i>	Vector to store the calculated importances in.
--------------------	--

39.252.4.7 End()

```
size_t End ( ) const [inline]
```

Return the first index of a point not contained in this node.

Definition at line 283 of file dtree.hpp.

39.252.4.8 FindBucket()

```
TagType FindBucket (
    const VecType & query ) const
```

Return the tag of the leaf containing the query.

This is useful for generating class memberships.

Parameters

<i>query</i>	Query to search for.
--------------	----------------------

39.252.4.9 Grow()

```
double Grow (
    MatType & data,
    arma::Col< size_t > & oldFromNew,
    const bool useVolReg = false,
    const size_t maxLeafSize = 10,
    const size_t minLeafSize = 5 )
```

Greedily expand the tree.

The points in the dataset will be reordered during tree growth.

Parameters

<i>data</i>	Dataset to build tree on.
<i>oldFromNew</i>	Mappings from old points to new points.
<i>useVolReg</i>	If true, volume regularization is used.
<i>maxLeafSize</i>	Maximum size of a leaf.
<i>minLeafSize</i>	Minimum size of a leaf.

39.252.4.10 Left()

```
DTree* Left ( ) const [inline]
```

Return the left child.

Definition at line 300 of file dtree.hpp.

39.252.4.11 LogNegativeError()

```
double LogNegativeError (
    const size_t totalPoints ) const
```

Compute the log-negative-error for this point, given the total number of points in the dataset.

Parameters

<i>totalPoints</i>	Total number of points in the dataset.
--------------------	--

39.252.4.12 LogNegError()

```
double LogNegError ( ) const [inline]
```

Return the log negative error of this node.

Definition at line 289 of file dtree.hpp.

39.252.4.13 LogVolume()

```
double LogVolume ( ) const [inline]
```

Return the inverse of the volume of this node.

Definition at line 298 of file dtree.hpp.

39.252.4.14 MaxVals()

```
const StatType& MaxVals ( ) const [inline]
```

Return the maximum values.

Definition at line 323 of file dtree.hpp.

39.252.4.15 MinVals()

```
const StatType& MinVals ( ) const [inline]
```

Return the minimum values.

Definition at line 326 of file dtree.hpp.

References DTree< MatType, TagType >::serialize().

39.252.4.16 NumChildren()

```
size_t NumChildren ( ) const [inline]
```

Return the number of children in this node.

Definition at line 310 of file dtree.hpp.

39.252.4.17 operator=() [1/2]

```
DTree& operator= (
    const DTree< MatType, TagType > & obj )
```

Copy the given tree.

Parameters

<i>obj</i>	Tree to copy.
------------	---------------

39.252.4.18 operator=() [2/2]

```
DTree& operator= (
    DTree< MatType, TagType > && obj )
```

Take ownership of the given tree (move operator).

Parameters

<i>obj</i>	Tree to take ownership of.
------------	----------------------------

39.252.4.19 PruneAndUpdate()

```
double PruneAndUpdate (
    const double oldAlpha,
    const size_t points,
    const bool useVolReg = false )
```

Perform alpha pruning on a tree.

Returns the new value of alpha.

Parameters

<i>oldAlpha</i>	Old value of alpha.
<i>points</i>	Total number of points in dataset.
<i>useVolReg</i>	If true, volume regularization is used.

Returns

New value of alpha.

39.252.4.20 Ratio()

```
double Ratio ( ) const [inline]
```

Return the ratio of points in this node to the points in the whole dataset.

Definition at line 296 of file dtree.hpp.

39.252.4.21 Right()

```
DTree* Right ( ) const [inline]
```

Return the right child.

Definition at line 302 of file dtree.hpp.

39.252.4.22 Root()

```
bool Root ( ) const [inline]
```

Return whether or not this is the root of the tree.

Definition at line 304 of file dtree.hpp.

39.252.4.23 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the density estimation tree.

Referenced by DTree< MatType, TagType >::MinVals().

39.252.4.24 SplitDim()

```
size_t SplitDim ( ) const [inline]
```

Return the split dimension of this node.

Definition at line 285 of file dtree.hpp.

39.252.4.25 SplitValue()

```
ElementType SplitValue ( ) const [inline]
```

Return the split value of this node.

Definition at line 287 of file dtree.hpp.

39.252.4.26 Start()

```
size_t Start ( ) const [inline]
```

Return the starting index of points contained in this node.

Definition at line 281 of file dtree.hpp.

39.252.4.27 SubtreeLeaves()

```
size_t SubtreeLeaves ( ) const [inline]
```

Return the number of leaves which are descendants of this node.

Definition at line 293 of file dtree.hpp.

39.252.4.28 SubtreeLeavesLogNegError()

```
double SubtreeLeavesLogNegError ( ) const [inline]
```

Return the log negative error of all descendants of this node.

Definition at line 291 of file `dtree.hpp`.

39.252.4.29 TagTree()

```
TagType TagTree (
    const TagType & tag = 0,
    bool everyNode = false )
```

Index the buckets for possible usage later; this results in every leaf in the tree having a specific tag (accessible with **BucketTag()** (p. 1214)).

This function calls itself recursively. The tag is incremented with `operator++()`, so any `TagType` overriding it will do.

Parameters

<i>tag</i>	Tag for the next leaf; leave at 0 for the initial call.
<i>everyNode</i>	Whether to increment on every node, not just leaves.

39.252.4.30 WithinRange()

```
bool WithinRange (
    const VecType & query ) const
```

Return whether a query point is within the range of this node.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/det/ dtree.hpp`

39.253 PathCacher Class Reference

This class is responsible for caching the path to each node of the tree.

Public Types

- enum **PathFormat** {
 FormatLR,
 FormatLR_ID,
 FormatID_LR }

Possible formats to use for output.

Public Member Functions

- template<typename MatType >
 PathCacher (**PathFormat** fmt, **DTree**< MatType, int > *tree)
 *Construct a **PathCacher** (p. 1221) object on the given tree with the given format.*
- template<typename MatType >
 void **Enter** (const **DTree**< MatType, int > *node, const **DTree**< MatType, int > *parent)
 Enter a given node.
- template<typename MatType >
 void **Leave** (const **DTree**< MatType, int > *node, const **DTree**< MatType, int > *parent)
 Leave the given node.
- size_t **NumNodes** () const
 Get the number of nodes in the path cache.
- int **ParentOf** (int tag) const
 Get the parent tag of a given tag.
- const std::string & **PathFor** (int tag) const
 Return the constructed path for a given tag.

Protected Types

- typedef std::vector< std::pair< int, std::string > > **PathCacheType**
- typedef std::list< std::pair< bool, int > > **PathType**

Protected Member Functions

- std::string **BuildString** ()

Protected Attributes

- **PathFormat** format
- **PathType** path
- **PathCacheType** pathCache

39.253.1 Detailed Description

This class is responsible for caching the path to each node of the tree.

Its instance is provided to **EnumerateTree()** (p. 463) utility ONCE and it caches the paths to all the leafs and then easily (and quickly) retrieves these paths for each test entry.

Definition at line 79 of file dt_utils.hpp.

39.253.2 Member Typedef Documentation

39.253.2.1 PathCacheType

```
typedef std::vector<std::pair<int, std::string> > PathCacheType [protected]
```

Definition at line 135 of file dt_utils.hpp.

39.253.2.2 PathType

```
typedef std::list<std::pair<bool, int> > PathType [protected]
```

Definition at line 134 of file dt_utils.hpp.

39.253.3 Member Enumeration Documentation

39.253.3.1 PathFormat

```
enum PathFormat
```

Possible formats to use for output.

Enumerator

FormatLR	Print only whether we went left or right.
FormatLR_ID	Print the direction, then the tag of the node.
FormatID_LR	Print the tag of the node, then the direction.

Definition at line 85 of file dt_utils.hpp.

39.253.4 Constructor & Destructor Documentation

39.253.4.1 PathCacher()

```
PathCacher (  
    PathFormat fmt,  
    DTree< MatType, int > * tree )
```

Construct a **PathCacher** (p. 1221) object on the given tree with the given format.

Parameters

<i>fmt</i>	Format to use for output.
<i>tree</i>	Tree to cache paths in.

39.253.5 Member Function Documentation

39.253.5.1 BuildString()

```
std::string BuildString ( ) [inline], [protected]
```

39.253.5.2 Enter()

```
void Enter (  
    const DTree< MatType, int > * node,  
    const DTree< MatType, int > * parent )
```

Enter a given node.

39.253.5.3 Leave()

```
void Leave (  
    const DTree< MatType, int > * node,  
    const DTree< MatType, int > * parent )
```

Leave the given node.

39.253.5.4 NumNodes()

```
size_t NumNodes ( ) const [inline]
```

Get the number of nodes in the path cache.

Definition at line 131 of file dt_utils.hpp.

References PathCacher::pathCache.

39.253.5.5 ParentOf()

```
int ParentOf (
    int tag ) const [inline]
```

Get the parent tag of a given tag.

39.253.5.6 PathFor()

```
const std::string& PathFor (
    int tag ) const [inline]
```

Return the constructed path for a given tag.

39.253.6 Member Data Documentation

39.253.6.1 format

```
PathFormat format [protected]
```

Definition at line 138 of file dt_utils.hpp.

39.253.6.2 path

```
PathType path [protected]
```

Definition at line 137 of file dt_utils.hpp.

39.253.6.3 pathCache

PathCacheType pathCache [protected]

Definition at line 139 of file dt_utils.hpp.

Referenced by PathCacher::NumNodes().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/det/ **dt_utils.hpp**

39.254 DiagonalGaussianDistribution Class Reference

A single multivariate Gaussian distribution with diagonal covariance.

Public Member Functions

- **DiagonalGaussianDistribution** ()
Default constructor, which creates a Gaussian with zero dimension.
- **DiagonalGaussianDistribution** (const size_t dimension)
Create a Gaussian Distribution with zero mean and diagonal covariance with the given dimensionality.
- **DiagonalGaussianDistribution** (const arma::vec &mean, const arma::vec &covariance)
Create a Gaussian distribution with the given mean and diagonal covariance.
- const arma::vec & **Covariance** () const
Return the covariance matrix.
- void **Covariance** (const arma::vec &covariance)
Set the covariance matrix.
- void **Covariance** (arma::vec &&covariance)
Set the covariance matrix using move assignment.
- size_t **Dimensionality** () const
Return the dimensionality of this distribution.
- double **LogProbability** (const arma::vec &observation) const
Return the log probability of the given observation.
- void **LogProbability** (const arma::mat &observations, arma::vec &logProbabilities) const
Calculate the multivariate Gaussian log probability density function for each data point (column) in the given matrix.
- const arma::vec & **Mean** () const
Return the mean.
- arma::vec & **Mean** ()
Return a modifiable copy of the mean.
- double **Probability** (const arma::vec &observation) const
Return the probability of the given observation.
- void **Probability** (const arma::mat &x, arma::vec &probabilities) const
Calculate the multivariate Gaussian probability density function for each data point (column) in the given matrix.
- arma::vec **Random** () const

Return a randomly generated observation according to the probability distribution defined by this object.

- `template<typename Archive >`

`void serialize (Archive &ar, const unsigned int)`

Serialize the distribution.

- `void Train (const arma::mat &observations)`

Estimate the Gaussian distribution directly from the given observations.

- `void Train (const arma::mat &observations, const arma::vec &probabilities)`

Estimate the Gaussian distribution from the given observations, taking into account the probability of each observation actually being from this distribution.

39.254.1 Detailed Description

A single multivariate Gaussian distribution with diagonal covariance.

Definition at line 21 of file `diagonal_gaussian_distribution.hpp`.

39.254.2 Constructor & Destructor Documentation

39.254.2.1 DiagonalGaussianDistribution() [1/3]

```
DiagonalGaussianDistribution ( ) [inline]
```

Default constructor, which creates a Gaussian with zero dimension.

Definition at line 38 of file `diagonal_gaussian_distribution.hpp`.

Referenced by `DiagonalGaussianDistribution::DiagonalGaussianDistribution()`.

39.254.2.2 DiagonalGaussianDistribution() [2/3]

```
DiagonalGaussianDistribution (
    const size_t dimension ) [inline]
```

Create a Gaussian Distribution with zero mean and diagonal covariance with the given dimensionality.

Parameters

<i>dimension</i>	Number of dimensions.
------------------	-----------------------

Definition at line 46 of file `diagonal_gaussian_distribution.hpp`.

References `DiagonalGaussianDistribution::DiagonalGaussianDistribution()`.

39.254.2.3 `DiagonalGaussianDistribution()` [3/3]

```
DiagonalGaussianDistribution (
    const arma::vec & mean,
    const arma::vec & covariance )
```

Create a Gaussian distribution with the given mean and diagonal covariance.

Parameters

<i>mean</i>	Mean of distribution.
<i>covariance</i>	Covariance of distribution.

39.254.3 Member Function Documentation

39.254.3.1 `Covariance()` [1/3]

```
const arma::vec& Covariance ( ) const [inline]
```

Return the covariance matrix.

Definition at line 133 of file `diagonal_gaussian_distribution.hpp`.

39.254.3.2 `Covariance()` [2/3]

```
void Covariance (
    const arma::vec & covariance )
```

Set the covariance matrix.

39.254.3.3 `Covariance()` [3/3]

```
void Covariance (
    arma::vec && covariance )
```

Set the covariance matrix using move assignment.

39.254.3.4 Dimensionality()

```
size_t Dimensionality ( ) const [inline]
```

Return the dimensionality of this distribution.

Definition at line 64 of file diagonal_gaussian_distribution.hpp.

39.254.3.5 LogProbability() [1/2]

```
double LogProbability (
    const arma::vec & observation ) const
```

Return the log probability of the given observation.

Referenced by DiagonalGaussianDistribution::Probability().

39.254.3.6 LogProbability() [2/2]

```
void LogProbability (
    const arma::mat & observations,
    arma::vec & logProbabilities ) const
```

Calculate the multivariate Gaussian log probability density function for each data point (column) in the given matrix.

Parameters

<i>observations</i>	Matrix of observations.
<i>probabilities</i>	Output log probabilities for each input observation.

39.254.3.7 Mean() [1/2]

```
const arma::vec& Mean ( ) const [inline]
```

Return the mean.

Definition at line 127 of file diagonal_gaussian_distribution.hpp.

39.254.3.8 Mean() [2/2]

```
arma::vec& Mean ( ) [inline]
```

Return a modifiable copy of the mean.

Definition at line 130 of file `diagonal_gaussian_distribution.hpp`.

39.254.3.9 Probability() [1/2]

```
double Probability (
    const arma::vec & observation ) const [inline]
```

Return the probability of the given observation.

Definition at line 67 of file `diagonal_gaussian_distribution.hpp`.

References `DiagonalGaussianDistribution::LogProbability()`.

39.254.3.10 Probability() [2/2]

```
void Probability (
    const arma::mat & x,
    arma::vec & probabilities ) const [inline]
```

Calculate the multivariate Gaussian probability density function for each data point (column) in the given matrix.

Parameters

<i>x</i>	Matrix of observations.
<i>probabilities</i>	Output probabilities for each input observation.

Definition at line 82 of file `diagonal_gaussian_distribution.hpp`.

References `DiagonalGaussianDistribution::LogProbability()`, `DiagonalGaussianDistribution::Random()`, and `DiagonalGaussianDistribution::Train()`.

39.254.3.11 Random()

```
arma::vec Random ( ) const
```

Return a randomly generated observation according to the probability distribution defined by this object.

Returns

Random observation from this Diagonal Gaussian distribution.

Referenced by `DiagonalGaussianDistribution::Probability()`.

39.254.3.12 `serialize()`

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the distribution.

Definition at line 143 of file `diagonal_gaussian_distribution.hpp`.

39.254.3.13 `Train()` [1/2]

```
void Train (
    const arma::mat & observations )
```

Estimate the Gaussian distribution directly from the given observations.

Parameters

<i>observations</i>	Matrix of observations.
---------------------	-------------------------

Referenced by `DiagonalGaussianDistribution::Probability()`.

39.254.3.14 `Train()` [2/2]

```
void Train (
    const arma::mat & observations,
    const arma::vec & probabilities )
```

Estimate the Gaussian distribution from the given observations, taking into account the probability of each observation actually being from this distribution.

Parameters

<i>observations</i>	Matrix of observations.
<i>probabilities</i>	List of probability of the each observation being from this distribution.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/ **diagonal_gaussian_distribution.hpp**

39.255 DiscreteDistribution Class Reference

A discrete distribution where the only observations are discrete observations.

Public Member Functions

- **DiscreteDistribution** ()
Default constructor, which creates a distribution that has no observations.
- **DiscreteDistribution** (const size_t numObservations)
Define the discrete distribution as having numObservations possible observations.
- **DiscreteDistribution** (const arma::Col< size_t > &numObservations)
Define the multidimensional discrete distribution as having numObservations possible observations.
- **DiscreteDistribution** (const std::vector< arma::vec > &probabilities)
Define the multidimensional discrete distribution as having the given probabilities for each observation.
- size_t **Dimensionality** () const
Get the dimensionality of the distribution.
- double **LogProbability** (const arma::vec &observation) const
Return the log probability of the given observation.
- void **LogProbability** (const arma::mat &x, arma::vec &logProbabilities) const
*Returns the **Log** (p. 1538) probability of the given matrix.*
- arma::vec & **Probabilities** (const size_t dim=0)
Return the vector of probabilities for the given dimension.
- const arma::vec & **Probabilities** (const size_t dim=0) const
Modify the vector of probabilities for the given dimension.
- double **Probability** (const arma::vec &observation) const
Return the probability of the given observation.
- void **Probability** (const arma::mat &x, arma::vec &probabilities) const
Calculates the Discrete probability density function for each data point (column) in the given matrix.
- arma::vec **Random** () const
Return a randomly generated observation (one-dimensional vector; one observation) according to the probability distribution defined by this object.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the distribution.
- void **Train** (const arma::mat &observations)
Estimate the probability distribution directly from the given observations.
- void **Train** (const arma::mat &observations, const arma::vec &probabilities)
Estimate the probability distribution from the given observations, taking into account the probability of each observation actually being from this distribution.

39.255.1 Detailed Description

A discrete distribution where the only observations are discrete observations.

This is useful (for example) with discrete Hidden Markov Models, where observations are non-negative integers representing specific emissions.

No bounds checking is performed for observations, so if an invalid observation is passed (i.e. `observation > numObservations`), a crash will probably occur.

This distribution only supports one-dimensional observations, so when passing an `arma::vec` as an observation, it should only have one dimension (`vec.n_rows == 1`). Any additional dimensions will simply be ignored.

Note

This class, like every other class in `mlpack`, uses `arma::vec` to represent observations. While a discrete distribution only has positive integers (`size_t`) as observations, these can be converted to doubles (which is what `arma::vec` holds). This distribution internally converts those doubles back into `size_t` before comparisons.

Definition at line 46 of file `discrete_distribution.hpp`.

39.255.2 Constructor & Destructor Documentation

39.255.2.1 DiscreteDistribution() [1/4]

```
DiscreteDistribution ( ) [inline]
```

Default constructor, which creates a distribution that has no observations.

Definition at line 53 of file `discrete_distribution.hpp`.

39.255.2.2 DiscreteDistribution() [2/4]

```
DiscreteDistribution (
    const size_t numObservations ) [inline]
```

Define the discrete distribution as having `numObservations` possible observations.

The probability in each state will be set to $(1 / \text{numObservations})$.

Parameters

<i>numObservations</i>	Number of possible observations this distribution can have.
------------------------	---

Definition at line 64 of file discrete_distribution.hpp.

39.255.2.3 DiscreteDistribution() [3/4]

```
DiscreteDistribution (
    const arma::Col< size_t > & numObservations ) [inline]
```

Define the multidimensional discrete distribution as having numObservations possible observations.

The probability in each state will be set to (1 / numObservations of each dimension).

Parameters

<i>numObservations</i>	Number of possible observations this distribution can have.
------------------------	---

Definition at line 77 of file discrete_distribution.hpp.

39.255.2.4 DiscreteDistribution() [4/4]

```
DiscreteDistribution (
    const std::vector< arma::vec > & probabilities ) [inline]
```

Define the multidimensional discrete distribution as having the given probabilities for each observation.

Parameters

<i>probabilities</i>	Probabilities of each possible observation.
----------------------	---

Definition at line 99 of file discrete_distribution.hpp.

39.255.3 Member Function Documentation

39.255.3.1 Dimensionality()

```
size_t Dimensionality ( ) const [inline]
```

Get the dimensionality of the distribution.

Definition at line 118 of file `discrete_distribution.hpp`.

39.255.3.2 LogProbability() [1/2]

```
double LogProbability (
    const arma::vec & observation ) const [inline]
```

Return the log probability of the given observation.

If the observation is greater than the number of possible observations, then a crash will probably occur – bounds checking is not performed.

Parameters

<i>observation</i>	Observation to return the log probability of.
--------------------	---

Returns

Log (p. 1538) probability of the given observation.

Definition at line 167 of file `discrete_distribution.hpp`.

References `DiscreteDistribution::Probability()`.

39.255.3.3 LogProbability() [2/2]

```
void LogProbability (
    const arma::mat & x,
    arma::vec & logProbabilities ) const [inline]
```

Returns the **Log** (p. 1538) probability of the given matrix.

These values are stored in `logProbabilities`.

Parameters

<i>x</i>	List of observations.
<i>logProbabilities</i>	Output log-probabilities for each input observation.

Definition at line 195 of file `discrete_distribution.hpp`.

References `DiscreteDistribution::Probability()`, `DiscreteDistribution::Random()`, and `DiscreteDistribution::Train()`.

39.255.3.4 Probabilities() [1/2]

```
arma::vec& Probabilities (
    const size_t dim = 0 ) [inline]
```

Return the vector of probabilities for the given dimension.

Definition at line 233 of file `discrete_distribution.hpp`.

39.255.3.5 Probabilities() [2/2]

```
const arma::vec& Probabilities (
    const size_t dim = 0 ) const [inline]
```

Modify the vector of probabilities for the given dimension.

Definition at line 235 of file `discrete_distribution.hpp`.

39.255.3.6 Probability() [1/2]

```
double Probability (
    const arma::vec & observation ) const [inline]
```

Return the probability of the given observation.

If the observation is greater than the number of possible observations, then a crash will probably occur – bounds checking is not performed.

Parameters

<i>observation</i>	Observation to return the probability of.
--------------------	---

Returns

Probability of the given observation.

Definition at line 128 of file `discrete_distribution.hpp`.

References Log::Fatal.

Referenced by DiscreteDistribution::LogProbability(), and DiscreteDistribution::Probability().

39.255.3.7 Probability() [2/2]

```
void Probability (
    const arma::mat & x,
    arma::vec & probabilities ) const [inline]
```

Calculates the Discrete probability density function for each data point (column) in the given matrix.

Parameters

<i>x</i>	List of observations.
<i>probabilities</i>	Output probabilities for each input observation.

Definition at line 180 of file discrete_distribution.hpp.

References DiscreteDistribution::Probability().

39.255.3.8 Random()

```
arma::vec Random ( ) const
```

Return a randomly generated observation (one-dimensional vector; one observation) according to the probability distribution defined by this object.

Returns

Random observation.

Referenced by DiscreteDistribution::LogProbability(), MockCategoricalData(), and AggregatedPolicy< PolicyType >::←Sample().

39.255.3.9 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the distribution.

Definition at line 242 of file discrete_distribution.hpp.

39.255.3.10 Train() [1/2]

```
void Train (
    const arma::mat & observations )
```

Estimate the probability distribution directly from the given observations.

If any of the observations is greater than numObservations, a crash is likely to occur.

Parameters

<i>observations</i>	List of observations.
---------------------	-----------------------

Referenced by DiscreteDistribution::LogProbability().

39.255.3.11 Train() [2/2]

```
void Train (
    const arma::mat & observations,
    const arma::vec & probabilities )
```

Estimate the probability distribution from the given observations, taking into account the probability of each observation actually being from this distribution.

Parameters

<i>observations</i>	List of observations.
<i>probabilities</i>	List of probabilities that each observation is actually from this distribution.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/ **discrete_distribution.hpp**

39.256 GammaDistribution Class Reference

This class represents the Gamma distribution.

Public Member Functions

- **GammaDistribution** (const size_t dimensionality=0)
Construct the Gamma distribution with the given number of dimensions (default 0); each parameter will be initialized to 0.

- **GammaDistribution** (const arma::mat &data, const double tol=1e-8)
Construct the Gamma distribution, training on the given parameters.
- **GammaDistribution** (const arma::vec &alpha, const arma::vec &beta)
Construct the Gamma distribution given two vectors alpha and beta.
- **~GammaDistribution** ()
Destructor.
- double **Alpha** (const size_t dim) const
Get the alpha parameter of the given dimension.
- double & **Alpha** (const size_t dim)
Modify the alpha parameter of the given dimension.
- double **Beta** (const size_t dim) const
Get the beta parameter of the given dimension.
- double & **Beta** (const size_t dim)
Modify the beta parameter of the given dimension.
- size_t **Dimensionality** () const
Get the dimensionality of the distribution.
- void **LogProbability** (const arma::mat &observations, arma::vec &logProbabilities) const
This function returns the logarithm of the probability of a group of observations.
- double **LogProbability** (double x, const size_t dim) const
This function returns the logarithm of the probability of a single observation.
- void **Probability** (const arma::mat &observations, arma::vec &probabilities) const
This function returns the probability of a group of observations.
- double **Probability** (double x, const size_t dim) const
This is a shortcut to the Probability(arma::mat&, arma::vec&) function for when we want to evaluate only the probability of one dimension of the gamma.
- arma::vec **Random** () const
This function returns an observation of this distribution.
- void **Train** (const arma::mat &rdata, const double tol=1e-8)
This function trains (fits distribution parameters) to new data or the dataset the object owns.
- void **Train** (const arma::mat &observations, const arma::vec &probabilities, const double tol=1e-8)
Fits an alpha and beta parameter according to observation probabilities.
- void **Train** (const arma::vec &logMeanxVec, const arma::vec &meanLogxVec, const arma::vec &meanxVec, const double tol=1e-8)
This function trains (fits distribution parameters) to a dataset with pre-computed statistics logMeanx, meanLogx, meanx for each dimension.

39.256.1 Detailed Description

This class represents the Gamma distribution.

It supports training a Gamma distribution on a given dataset and accessing the fitted alpha and beta parameters.

This class supports multidimensional Gamma distributions; however, it is assumed that each dimension is independent; therefore, a multidimensional Gamma distribution here may be seen as a set of independent single-dimensional Gamma distributions—and the parameters are estimated under this assumption.

The estimation algorithm used can be found in the following paper:

```
@techreport{minka2002estimating,
  title={Estimating a {G}amma distribution},
  author={Minka, Thomas P.},
  institution={Microsoft Research},
  address={Cambridge, U.K.},
  year={2002}
}
```

Definition at line 52 of file gamma_distribution.hpp.

39.256.2 Constructor & Destructor Documentation

39.256.2.1 GammaDistribution() [1/3]

```
GammaDistribution (
    const size_t dimensionality = 0 )
```

Construct the Gamma distribution with the given number of dimensions (default 0); each parameter will be initialized to 0.

Parameters

<i>dimensionality</i>	Number of dimensions.
-----------------------	-----------------------

39.256.2.2 GammaDistribution() [2/3]

```
GammaDistribution (
    const arma::mat & data,
    const double tol = 1e-8 )
```

Construct the Gamma distribution, training on the given parameters.

Parameters

<i>data</i>	Data to train the distribution on.
<i>tol</i>	Convergence tolerance. This is <i>not</i> an absolute measure: It will stop the approximation once the <i>change</i> in the value is smaller than tol.

39.256.2.3 GammaDistribution() [3/3]

```
GammaDistribution (
```



```
const arma::vec & alpha,
const arma::vec & beta )
```

Construct the Gamma distribution given two vectors alpha and beta.

Parameters

<i>alpha</i>	The vector of alphas, one per dimension.
<i>beta</i>	The vector of betas, one per dimension.

39.256.2.4 ~GammaDistribution()

```
~ GammaDistribution ( ) [inline]
```

Destructor.

Definition at line 84 of file gamma_distribution.hpp.

References GammaDistribution::LogProbability(), GammaDistribution::Probability(), GammaDistribution::Random(), and GammaDistribution::Train().

39.256.3 Member Function Documentation

39.256.3.1 Alpha() [1/2]

```
double Alpha (
    const size_t dim ) const [inline]
```

Get the alpha parameter of the given dimension.

Definition at line 197 of file gamma_distribution.hpp.

39.256.3.2 Alpha() [2/2]

```
double& Alpha (
    const size_t dim ) [inline]
```

Modify the alpha parameter of the given dimension.

Definition at line 199 of file gamma_distribution.hpp.

39.256.3.3 Beta() [1/2]

```
double Beta (
    const size_t dim ) const [inline]
```

Get the beta parameter of the given dimension.

Definition at line 202 of file gamma_distribution.hpp.

39.256.3.4 Beta() [2/2]

```
double& Beta (
    const size_t dim ) [inline]
```

Modify the beta parameter of the given dimension.

Definition at line 204 of file gamma_distribution.hpp.

39.256.3.5 Dimensionality()

```
size_t Dimensionality ( ) const [inline]
```

Get the dimensionality of the distribution.

Definition at line 207 of file gamma_distribution.hpp.

39.256.3.6 LogProbability() [1/2]

```
void LogProbability (
    const arma::mat & observations,
    arma::vec & logProbabilities ) const
```

This function returns the logarithm of the probability of a group of observations.

The logarithm of the probability of a value x is

$$\log\left(\frac{x^{(\alpha-1)}}{\Gamma(\alpha)\beta^\alpha}e^{-\frac{x}{\beta}}\right)$$

for one dimension. This implementation assumes each dimension is independent, so the product rule is used.

Parameters

<i>observations</i>	Matrix of observations, one per column.
<i>logProbabilities</i>	Column vector of log probabilities, one per observation.

Referenced by GammaDistribution::~GammaDistribution().

39.256.3.7 LogProbability() [2/2]

```
double LogProbability (
    double x,
    const size_t dim ) const
```

This function returns the logarithm of the probability of a single observation.

Parameters

<i>x</i>	The 1-dimensional observation.
<i>dim</i>	The dimension for which to calculate the probability.

39.256.3.8 Probability() [1/2]

```
void Probability (
    const arma::mat & observations,
    arma::vec & probabilities ) const
```

This function returns the probability of a group of observations.

The probability of the value x is

$$\frac{x^{(\alpha-1)}}{\Gamma(\alpha)\beta^\alpha} e^{-\frac{x}{\beta}}$$

for one dimension. This implementation assumes each dimension is independent, so the product rule is used.

Parameters

<i>observations</i>	Matrix of observations, one per column.
<i>probabilities</i>	Column vector of probabilities, one per observation.

Referenced by GammaDistribution::~~GammaDistribution().

39.256.3.9 Probability() [2/2]

```
double Probability (
    double x,
    const size_t dim ) const
```

This is a shortcut to the Probability(arma::mat&, arma::vec&) function for when we want to evaluate only the probability of one dimension of the gamma.

Parameters

<i>x</i>	The 1-dimensional observation.
<i>dim</i>	The dimension for which to calculate the probability.

39.256.3.10 Random()

```
arma::vec Random ( ) const
```

This function returns an observation of this distribution.

Referenced by GammaDistribution::~~GammaDistribution().

39.256.3.11 Train() [1/3]

```
void Train (
    const arma::mat & rdata,
    const double tol = 1e-8 )
```

This function trains (fits distribution parameters) to new data or the dataset the object owns.

Parameters

<i>rdata</i>	Reference data to fit parameters to.
<i>tol</i>	Convergence tolerance. This is <i>not</i> an absolute measure: It will stop the approximation once the <i>change</i> in the value is smaller than tol.

Referenced by GammaDistribution::~~GammaDistribution().

39.256.3.12 Train() [2/3]

```
void Train (
    const arma::mat & observations,
    const arma::vec & probabilities,
    const double tol = 1e-8 )
```

Fits an alpha and beta parameter according to observation probabilities.

This method is not yet implemented.

Parameters

<i>observations</i>	The reference data, one observation per column.
<i>probabilities</i>	The probability of each observation. One value per column of the observations matrix.
<i>tol</i>	Convergence tolerance. This is <i>not</i> an absolute measure: It will stop the approximation once the <i>change</i> in the value is smaller than tol.

39.256.3.13 Train() [3/3]

```
void Train (
    const arma::vec & logMeanxVec,
    const arma::vec & meanLogxVec,
    const arma::vec & meanxVec,
    const double tol = 1e-8 )
```

This function trains (fits distribution parameters) to a dataset with pre-computed statistics logMeanx, meanLogx, meanx for each dimension.

Parameters

<i>logMeanxVec</i>	Is each dimension's logarithm of the mean (log(mean(x))).
<i>meanLogxVec</i>	Is each dimension's mean of logarithms (mean(log(x))).
<i>meanxVec</i>	Is each dimension's mean (mean(x)).
<i>tol</i>	Convergence tolerance. This is <i>not</i> an absolute measure: It will stop the approximation once the <i>change</i> in the value is smaller than tol.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/ **gamma_distribution.hpp**

39.257 GaussianDistribution Class Reference

A single multivariate Gaussian distribution.

Public Member Functions

- **GaussianDistribution** ()
Default constructor, which creates a Gaussian with zero dimension.
- **GaussianDistribution** (const size_t dimension)
Create a Gaussian distribution with zero mean and identity covariance with the given dimensionality.
- **GaussianDistribution** (const arma::vec &mean, const arma::mat &covariance)
Create a Gaussian distribution with the given mean and covariance.
- const arma::mat & **Covariance** () const
Return the covariance matrix.
- void **Covariance** (const arma::mat &covariance)
Set the covariance.
- void **Covariance** (arma::mat &&covariance)
- size_t **Dimensionality** () const
Return the dimensionality of this distribution.
- double **LogProbability** (const arma::vec &observation) const
Return the log probability of the given observation.
- void **LogProbability** (const arma::mat &x, arma::vec &logProbabilities) const
*Returns the **Log** (p. 1538) probability of the given matrix.*
- const arma::vec & **Mean** () const
Return the mean.
- arma::vec & **Mean** ()
Return a modifiable copy of the mean.
- double **Probability** (const arma::vec &observation) const
Return the probability of the given observation.
- void **Probability** (const arma::mat &x, arma::vec &probabilities) const
Calculates the multivariate Gaussian probability density function for each data point (column) in the given matrix.
- arma::vec **Random** () const
Return a randomly generated observation according to the probability distribution defined by this object.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the distribution.
- void **Train** (const arma::mat &observations)
Estimate the Gaussian distribution directly from the given observations.
- void **Train** (const arma::mat &observations, const arma::vec &probabilities)
Estimate the Gaussian distribution from the given observations, taking into account the probability of each observation actually being from this distribution.

39.257.1 Detailed Description

A single multivariate Gaussian distribution.

Definition at line 24 of file gaussian_distribution.hpp.

39.257.2 Constructor & Destructor Documentation

39.257.2.1 GaussianDistribution() [1/3]

```
GaussianDistribution ( ) [inline]
```

Default constructor, which creates a Gaussian with zero dimension.

Definition at line 45 of file gaussian_distribution.hpp.

Referenced by GaussianDistribution::GaussianDistribution().

39.257.2.2 GaussianDistribution() [2/3]

```
GaussianDistribution (  
    const size_t dimension ) [inline]
```

Create a Gaussian distribution with zero mean and identity covariance with the given dimensionality.

Definition at line 51 of file gaussian_distribution.hpp.

References GaussianDistribution::GaussianDistribution().

39.257.2.3 GaussianDistribution() [3/3]

```
GaussianDistribution (  
    const arma::vec & mean,  
    const arma::mat & covariance )
```

Create a Gaussian distribution with the given mean and covariance.

covariance is expected to be positive definite.

39.257.3 Member Function Documentation

39.257.3.1 Covariance() [1/3]

```
const arma::mat& Covariance ( ) const [inline]
```

Return the covariance matrix.

Definition at line 162 of file `gaussian_distribution.hpp`.

Referenced by `RegressionDistribution::RegressionDistribution()`.

39.257.3.2 Covariance() [2/3]

```
void Covariance (
    const arma::mat & covariance )
```

Set the covariance.

39.257.3.3 Covariance() [3/3]

```
void Covariance (
    arma::mat && covariance )
```

39.257.3.4 Dimensionality()

```
size_t Dimensionality ( ) const [inline]
```

Return the dimensionality of this distribution.

Definition at line 69 of file `gaussian_distribution.hpp`.

39.257.3.5 LogProbability() [1/2]

```
double LogProbability (
    const arma::vec & observation ) const
```

Return the log probability of the given observation.

Referenced by `GaussianDistribution::Probability()`.

39.257.3.6 LogProbability() [2/2]

```
void LogProbability (
    const arma::mat & x,
    arma::vec & logProbabilities ) const [inline]
```

Returns the **Log** (p. 1538) probability of the given matrix.

These values are stored in `logProbabilities`.

Parameters

<i>x</i>	List of observations.
<i>logProbabilities</i>	Output log probabilities for each input observation.

Definition at line 108 of file gaussian_distribution.hpp.

References GaussianDistribution::Random(), and GaussianDistribution::Train().

39.257.3.7 Mean() [1/2]

```
const arma::vec& Mean ( ) const [inline]
```

Return the mean.

Definition at line 152 of file gaussian_distribution.hpp.

39.257.3.8 Mean() [2/2]

```
arma::vec& Mean ( ) [inline]
```

Return a modifiable copy of the mean.

Definition at line 157 of file gaussian_distribution.hpp.

39.257.3.9 Probability() [1/2]

```
double Probability (
    const arma::vec & observation ) const [inline]
```

Return the probability of the given observation.

Definition at line 74 of file gaussian_distribution.hpp.

References GaussianDistribution::LogProbability().

Referenced by GaussianDistribution::Probability().

39.257.3.10 Probability() [2/2]

```
void Probability (
    const arma::mat & x,
    arma::vec & probabilities ) const [inline]
```

Calculates the multivariate Gaussian probability density function for each data point (column) in the given matrix.

Parameters

<i>x</i>	List of observations.
<i>probabilities</i>	Output probabilities for each input observation.

Definition at line 91 of file gaussian_distribution.hpp.

References GaussianDistribution::Probability().

39.257.3.11 Random()

```
arma::vec Random ( ) const
```

Return a randomly generated observation according to the probability distribution defined by this object.

Returns

Random observation from this Gaussian distribution.

Referenced by LogisticRegressionTestData(), and GaussianDistribution::LogProbability().

39.257.3.12 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the distribution.

Definition at line 175 of file gaussian_distribution.hpp.

39.257.3.13 Train() [1/2]

```
void Train (
    const arma::mat & observations )
```

Estimate the Gaussian distribution directly from the given observations.

Parameters

<i>observations</i>	List of observations.
---------------------	-----------------------

Referenced by GaussianDistribution::LogProbability().

39.257.3.14 Train() [2/2]

```
void Train (
    const arma::mat & observations,
    const arma::vec & probabilities )
```

Estimate the Gaussian distribution from the given observations, taking into account the probability of each observation actually being from this distribution.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/ **gaussian_distribution.hpp**

39.258 LaplaceDistribution Class Reference

The multivariate Laplace distribution centered at 0 has pdf.

Public Member Functions

- **LaplaceDistribution** ()
Default constructor, which creates a Laplace distribution with zero dimension and zero scale parameter.
- **LaplaceDistribution** (const size_t dimensionality, const double scale)
Construct the Laplace distribution with the given scale and dimensionality.
- **LaplaceDistribution** (const arma::vec &mean, const double scale)
Construct the Laplace distribution with the given mean and scale parameter.
- size_t **Dimensionality** () const
Return the dimensionality of this distribution.
- void **Estimate** (const arma::mat &observations)
Estimate the Laplace distribution directly from the given observations.
- void **Estimate** (const arma::mat &observations, const arma::vec &probabilities)
Estimate the Laplace distribution from the given observations, taking into account the probability of each observation actually being from this distribution.
- double **LogProbability** (const arma::vec &observation) const
Return the log probability of the given observation.
- void **LogProbability** (const arma::mat &x, arma::vec &logProbabilities) const
Evaluate log probability density function of given observation.

- `const arma::vec & Mean () const`
Return the mean.
- `arma::vec & Mean ()`
Modify the mean.
- `double Probability (const arma::vec &observation) const`
Return the probability of the given observation.
- `void Probability (const arma::mat &x, arma::vec &probabilities) const`
Evaluate probability density function of given observation.
- `arma::vec Random () const`
Return a randomly generated observation according to the probability distribution defined by this object.
- `double Scale () const`
Return the scale parameter.
- `double & Scale ()`
Modify the scale parameter.
- `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`
Serialize the distribution.

39.258.1 Detailed Description

The multivariate Laplace distribution centered at 0 has pdf.

$$f(x|\theta) = \frac{1}{2\theta} \exp\left(-\frac{\|x - \mu\|}{\theta}\right)$$

given scale parameter θ and mean μ . This implementation assumes a diagonal covariance, but a rewrite to support arbitrary covariances is possible.

See the following paper for more information on the non-diagonal-covariance Laplace distribution and estimation techniques:

```
@article{eltoft2006multivariate,
  title={On the Multivariate Laplace Distribution},
  author={Eltoft, Torbjørn and Kim, Taesu and Lee, Te-Won},
  journal={IEEE Signal Processing Letters},
  volume={13},
  number={5},
  pages={300--304},
  year={2006}
}
```

Note that because of the diagonal covariance restriction, much of the algebra in the paper above becomes simplified, and the PDF takes roughly the same form as the univariate case.

Definition at line 50 of file `laplace_distribution.hpp`.

39.258.2 Constructor & Destructor Documentation

39.258.2.1 LaplaceDistribution() [1/3]

```
LaplaceDistribution ( ) [inline]
```

Default constructor, which creates a Laplace distribution with zero dimension and zero scale parameter.

Definition at line 57 of file laplace_distribution.hpp.

39.258.2.2 LaplaceDistribution() [2/3]

```
LaplaceDistribution (
    const size_t dimensionality,
    const double scale ) [inline]
```

Construct the Laplace distribution with the given scale and dimensionality.

The mean is initialized to zero.

Parameters

<i>dimensionality</i>	Dimensionality of distribution.
<i>scale</i>	Scale of distribution.

Definition at line 66 of file laplace_distribution.hpp.

39.258.2.3 LaplaceDistribution() [3/3]

```
LaplaceDistribution (
    const arma::vec & mean,
    const double scale ) [inline]
```

Construct the Laplace distribution with the given mean and scale parameter.

Parameters

<i>mean</i>	Mean of distribution.
<i>scale</i>	Scale of distribution.

Definition at line 76 of file laplace_distribution.hpp.

39.258.3 Member Function Documentation

39.258.3.1 Dimensionality()

```
size_t Dimensionality ( ) const [inline]
```

Return the dimensionality of this distribution.

Definition at line 80 of file `laplace_distribution.hpp`.

39.258.3.2 Estimate() [1/2]

```
void Estimate (
    const arma::mat & observations )
```

Estimate the Laplace distribution directly from the given observations.

Parameters

<i>observations</i>	List of observations.
---------------------	-----------------------

Referenced by `LaplaceDistribution::Random()`.

39.258.3.3 Estimate() [2/2]

```
void Estimate (
    const arma::mat & observations,
    const arma::vec & probabilities )
```

Estimate the Laplace distribution from the given observations, taking into account the probability of each observation actually being from this distribution.

39.258.3.4 LogProbability() [1/2]

```
double LogProbability (
    const arma::vec & observation ) const
```

Return the log probability of the given observation.

Parameters

<i>observation</i>	Point to evaluate logarithm of probability.
--------------------	---

Referenced by LaplaceDistribution::LogProbability(), and LaplaceDistribution::Probability().

39.258.3.5 LogProbability() [2/2]

```
void LogProbability (
    const arma::mat & x,
    arma::vec & logProbabilities ) const [inline]
```

Evaluate log probability density function of given observation.

Parameters

<i>x</i>	List of observations.
<i>logProbabilities</i>	Output probabilities for each input observation.

Definition at line 113 of file laplace_distribution.hpp.

References LaplaceDistribution::LogProbability().

39.258.3.6 Mean() [1/2]

```
const arma::vec& Mean ( ) const [inline]
```

Return the mean.

Definition at line 163 of file laplace_distribution.hpp.

39.258.3.7 Mean() [2/2]

```
arma::vec& Mean ( ) [inline]
```

Modify the mean.

Definition at line 165 of file laplace_distribution.hpp.

39.258.3.8 Probability() [1/2]

```
double Probability (
    const arma::vec & observation ) const [inline]
```

Return the probability of the given observation.

Parameters

<i>observation</i>	Point to evaluate probability at.
--------------------	-----------------------------------

Definition at line 87 of file laplace_distribution.hpp.

References LaplaceDistribution::LogProbability().

39.258.3.9 Probability() [2/2]

```
void Probability (
    const arma::mat & x,
    arma::vec & probabilities ) const
```

Evaluate probability density function of given observation.

Parameters

<i>x</i>	List of observations.
<i>probabilities</i>	Output probabilities for each input observation.

39.258.3.10 Random()

```
arma::vec Random ( ) const [inline]
```

Return a randomly generated observation according to the probability distribution defined by this object.

This is inlined for speed.

Returns

Random observation from this Laplace distribution.

Definition at line 128 of file laplace_distribution.hpp.

References LaplaceDistribution::Estimate().

39.258.3.11 Scale() [1/2]

```
double Scale ( ) const [inline]
```

Return the scale parameter.

Definition at line 168 of file `laplace_distribution.hpp`.

39.258.3.12 Scale() [2/2]

```
double& Scale ( ) [inline]
```

Modify the scale parameter.

Definition at line 170 of file `laplace_distribution.hpp`.

39.258.3.13 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the distribution.

Definition at line 176 of file `laplace_distribution.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/ laplace_distribution.hpp`

39.259 RegressionDistribution Class Reference

A class that represents a univariate conditionally Gaussian distribution.

Public Member Functions

- **RegressionDistribution** ()
Default constructor, which creates a Gaussian with zero dimension.
- **mlpack_deprecated RegressionDistribution** (const arma::mat &predictors, const arma::vec &responses)
Create a Conditional Gaussian distribution with conditional mean function obtained by running RegressionFunction on predictors, responses.
- **RegressionDistribution** (const arma::mat &predictors, const arma::rowvec &responses)
Create a Conditional Gaussian distribution with conditional mean function obtained by running RegressionFunction on predictors, responses.
- size_t **Dimensionality** () const
Return the dimensionality.
- const **GaussianDistribution & Err** () const
Return error distribution.
- **GaussianDistribution & Err** ()
Modify error distribution.
- double **LogProbability** (const arma::vec &observation) const
Evaluate log probability density function of given observation.
- const arma::vec & **Parameters** () const
Return the parameters (the b vector).
- **mlpack_deprecated void Predict** (const arma::mat &points, arma::vec &predictions) const
Calculate y_i for each data point in points.
- void **Predict** (const arma::mat &points, arma::rowvec &predictions) const
Calculate y_i for each data point in points.
- double **Probability** (const arma::vec &observation) const
Evaluate probability density function of given observation.
- const **regression::LinearRegression & Rf** () const
Return regression function.
- **regression::LinearRegression & Rf** ()
Modify regression function.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the distribution.
- void **Train** (const arma::mat &observations)
Estimate the Gaussian distribution directly from the given observations.
- **mlpack_deprecated void Train** (const arma::mat &observations, const arma::vec &weights)
Estimate parameters using provided observation weights.
- void **Train** (const arma::mat &observations, const arma::rowvec &weights)
Estimate parameters using provided observation weights.

39.259.1 Detailed Description

A class that represents a univariate conditionally Gaussian distribution.

Can be used as an emission distribution with the hmm class to implement HMM regression (HMMR) as described in <https://www.ima.umn.edu/preprints/January1994/1195.pdf> The hmm observations should have the dependent variable in the first row, with the independent variables in the other rows.

Definition at line 31 of file regression_distribution.hpp.

39.259.2 Constructor & Destructor Documentation

39.259.2.1 RegressionDistribution() [1/3]

```
RegressionDistribution ( ) [inline]
```

Default constructor, which creates a Gaussian with zero dimension.

Definition at line 43 of file `regression_distribution.hpp`.

39.259.2.2 RegressionDistribution() [2/3]

```
mlpack_deprecated RegressionDistribution (  
    const arma::mat & predictors,  
    const arma::vec & responses ) [inline]
```

Create a Conditional Gaussian distribution with conditional mean function obtained by running `RegressionFunction` on `predictors`, `responses`.

Parameters

<i>predictors</i>	Matrix of predictors (X).
<i>responses</i>	Vector of responses (y).

Definition at line 52 of file `regression_distribution.hpp`.

39.259.2.3 RegressionDistribution() [3/3]

```
RegressionDistribution (  
    const arma::mat & predictors,  
    const arma::rowvec & responses ) [inline]
```

Create a Conditional Gaussian distribution with conditional mean function obtained by running `RegressionFunction` on `predictors`, `responses`.

Parameters

<i>predictors</i>	Matrix of predictors (X).
<i>responses</i>	Vector of responses (y).

Definition at line 64 of file `regression_distribution.hpp`.

References `LinearRegression::ComputeError()`, `GaussianDistribution::Covariance()`, and `LinearRegression::Train()`.

39.259.3 Member Function Documentation

39.259.3.1 Dimensionality()

```
size_t Dimensionality ( ) const [inline]
```

Return the dimensionality.

Definition at line 156 of file `regression_distribution.hpp`.

References `LinearRegression::Parameters()`.

39.259.3.2 Err() [1/2]

```
const GaussianDistribution& Err ( ) const [inline]
```

Return error distribution.

Definition at line 90 of file `regression_distribution.hpp`.

39.259.3.3 Err() [2/2]

```
GaussianDistribution& Err ( ) [inline]
```

Modify error distribution.

Definition at line 92 of file `regression_distribution.hpp`.

References `mlpack_deprecated`, `RegressionDistribution::Probability()`, and `RegressionDistribution::Train()`.

39.259.3.4 LogProbability()

```
double LogProbability (
    const arma::vec & observation ) const [inline]
```

Evaluate log probability density function of given observation.

Parameters

<i>observation</i>	Point to evaluate log probability at.
--------------------	---------------------------------------

Definition at line 130 of file regression_distribution.hpp.

References `mlpack_deprecated`, `RegressionDistribution::Predict()`, and `RegressionDistribution::Probability()`.

39.259.3.5 Parameters()

```
const arma::vec& Parameters ( ) const [inline]
```

Return the parameters (the *b* vector).

Definition at line 153 of file regression_distribution.hpp.

References `LinearRegression::Parameters()`.

39.259.3.6 Predict() [1/2]

```
mlpack_deprecated void Predict (
    const arma::mat & points,
    arma::vec & predictions ) const
```

Calculate y_i for each data point in *points*.

Parameters

<i>points</i>	The data points to calculate with.
<i>predictions</i>	Y, will contain calculated values on completion.

Referenced by `RegressionDistribution::LogProbability()`.

39.259.3.7 Predict() [2/2]

```
void Predict (
    const arma::mat & points,
    arma::rowvec & predictions ) const
```

Calculate y_i for each data point in *points*.

Parameters

<i>points</i>	The data points to calculate with.
<i>predictions</i>	Y, will contain calculated values on completion.

39.259.3.8 Probability()

```
double Probability (
    const arma::vec & observation ) const
```

Evaluate probability density function of given observation.

Parameters

<i>observation</i>	Point to evaluate probability at.
--------------------	-----------------------------------

Referenced by RegressionDistribution::Err(), and RegressionDistribution::LogProbability().

39.259.3.9 Rf() [1/2]

```
const regression::LinearRegression& Rf ( ) const [inline]
```

Return regression function.

Definition at line 85 of file regression_distribution.hpp.

39.259.3.10 Rf() [2/2]

```
regression::LinearRegression& Rf ( ) [inline]
```

Modify regression function.

Definition at line 87 of file regression_distribution.hpp.

39.259.3.11 `serialize()`

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the distribution.

Definition at line 78 of file `regression_distribution.hpp`.

39.259.3.12 `Train()` [1/3]

```
void Train (
    const arma::mat & observations )
```

Estimate the Gaussian distribution directly from the given observations.

Parameters

<i>observations</i>	List of observations.
---------------------	-----------------------

Referenced by `RegressionDistribution::Err()`.

39.259.3.13 `Train()` [2/3]

```
mlpack_deprecated void Train (
    const arma::mat & observations,
    const arma::vec & weights )
```

Estimate parameters using provided observation weights.

Parameters

<i>observations</i>	List of observations.
<i>weights</i>	Probability that given observation is from distribution.

39.259.3.14 `Train()` [3/3]

```
void Train (
    const arma::mat & observations,
    const arma::rowvec & weights )
```

Estimate parameters using provided observation weights.

Parameters

<i>observations</i>	List of observations.
<i>weights</i>	Probability that given observation is from distribution.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/ **regression_distribution.hpp**

39.260 DTBRules< MetricType, TreeType > Class Template Reference

Public Types

- typedef **tree::TraversallInfo**< TreeType > **TraversallInfoType**

Public Member Functions

- **DTBRules** (const arma::mat &dataSet, **UnionFind** &connections, arma::vec &neighborsDistances, arma::Col< size_t > &neighborsInComponent, arma::Col< size_t > &neighborsOutComponent, MetricType &metric)
- double **BaseCase** (const size_t queryIndex, const size_t referenceIndex)
- size_t **BaseCases** () const
Get the number of base cases performed.
- size_t & **BaseCases** ()
Modify the number of base cases performed.
- double **Rescore** (const size_t queryIndex, TreeType &referenceNode, const double oldScore)
Re-evaluate the score for recursion order.
- double **Rescore** (TreeType &queryNode, TreeType &referenceNode, const double oldScore) const
Re-evaluate the score for recursion order.
- double **Score** (const size_t queryIndex, TreeType &referenceNode)
Get the score for recursion order.
- double **Score** (TreeType &queryNode, TreeType &referenceNode)
Get the score for recursion order.
- size_t **Scores** () const
Get the number of node combinations that have been scored.
- size_t & **Scores** ()
Modify the number of node combinations that have been scored.
- const **TraversallInfoType** & **TraversallInfo** () const
- **TraversallInfoType** & **TraversallInfo** ()

39.260.1 Detailed Description

```
template<typename MetricType, typename TreeType>
class mlpack::emst::DTBRules< MetricType, TreeType >
```

Definition at line 23 of file dtb_rules.hpp.

39.260.2 Member Typedef Documentation

39.260.2.1 TraversalInfoType

```
typedef tree::TraversalInfo<TreeType> TraversalInfoType
```

Definition at line 85 of file dtb_rules.hpp.

39.260.3 Constructor & Destructor Documentation

39.260.3.1 DTBRules()

```
DTBRules (
    const arma::mat & dataSet,
    UnionFind & connections,
    arma::vec & neighborsDistances,
    arma::Col< size_t > & neighborsInComponent,
    arma::Col< size_t > & neighborsOutComponent,
    MetricType & metric )
```

39.260.4 Member Function Documentation

39.260.4.1 BaseCase()

```
double BaseCase (
    const size_t queryIndex,
    const size_t referenceIndex )
```

39.260.4.2 BaseCases() [1/2]

```
size_t BaseCases ( ) const [inline]
```

Get the number of base cases performed.

Definition at line 91 of file dtb_rules.hpp.

39.260.4.3 BaseCases() [2/2]

```
size_t& BaseCases ( ) [inline]
```

Modify the number of base cases performed.

Definition at line 93 of file dtb_rules.hpp.

39.260.4.4 Rescore() [1/2]

```
double Rescore (
    const size_t queryIndex,
    TreeType & referenceNode,
    const double oldScore )
```

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.
<i>oldScore</i>	Old score produced by Score() (p. 1267) (or Rescore() (p. 1266)).

39.260.4.5 Rescore() [2/2]

```
double Rescore (
    TreeType & queryNode,
```

```
TreeType & referenceNode,
const double oldScore ) const
```

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.
<i>oldScore</i>	Old score produced by Socre() (or Rescore() (p. 1266)).

39.260.4.6 Score() [1/2]

```
double Score (
    const size_t queryIndex,
    TreeType & referenceNode )
```

Get the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.

39.260.4.7 Score() [2/2]

```
double Score (
    TreeType & queryNode,
    TreeType & referenceNode )
```

Get the score for recursion order.

A low score indicates priority for recursionm while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.

39.260.4.8 Scores() [1/2]

```
size_t Scores ( ) const [inline]
```

Get the number of node combinations that have been scored.

Definition at line 96 of file dtb_rules.hpp.

39.260.4.9 Scores() [2/2]

```
size_t& Scores ( ) [inline]
```

Modify the number of node combinations that have been scored.

Definition at line 98 of file dtb_rules.hpp.

39.260.4.10 TraversalInfo() [1/2]

```
const TraversalInfoType& TraversalInfo ( ) const [inline]
```

Definition at line 87 of file dtb_rules.hpp.

39.260.4.11 TraversalInfo() [2/2]

```
TraversalInfoType& TraversalInfo ( ) [inline]
```

Definition at line 88 of file dtb_rules.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/ **dtb_rules.hpp**

39.261 DTBStat Class Reference

A statistic for use with mlpack trees, which stores the upper bound on distance to nearest neighbors and the component which this node belongs to.

Public Member Functions

- **DTBStat** ()
A generic initializer.
- template<typename TreeType >
DTBStat (const TreeType &node)
This is called when a node is finished initializing.
- double **Bound** () const
Get the total bound for pruning.
- double & **Bound** ()
Modify the total bound for pruning.
- int **ComponentMembership** () const
Get the component membership of this node.
- int & **ComponentMembership** ()
Modify the component membership of this node.
- double **MaxNeighborDistance** () const
Get the maximum neighbor distance.
- double & **MaxNeighborDistance** ()
Modify the maximum neighbor distance.
- double **MinNeighborDistance** () const
Get the minimum neighbor distance.
- double & **MinNeighborDistance** ()
Modify the minimum neighbor distance.

39.261.1 Detailed Description

A statistic for use with mlpack trees, which stores the upper bound on distance to nearest neighbors and the component which this node belongs to.

Definition at line 24 of file dtb_stat.hpp.

39.261.2 Constructor & Destructor Documentation

39.261.2.1 DTBStat() [1/2]

```
DTBStat ( ) [inline]
```

A generic initializer.

Sets the maximum neighbor distance to its default, and the component membership to -1 (no component).

Definition at line 49 of file dtb_stat.hpp.

39.261.2.2 DTBStat() [2/2]

```
DTBStat (
    const TreeType & node ) [inline]
```

This is called when a node is finished initializing.

We set the maximum neighbor distance to its default, and if possible, we set the component membership of the node (if it has only one point and no children).

Parameters

<i>node</i>	Node that has been finished.
-------------	------------------------------

Definition at line 63 of file dtb_stat.hpp.

39.261.3 Member Function Documentation**39.261.3.1 Bound()** [1/2]

```
double Bound ( ) const [inline]
```

Get the total bound for pruning.

Definition at line 82 of file dtb_stat.hpp.

39.261.3.2 Bound() [2/2]

```
double& Bound ( ) [inline]
```

Modify the total bound for pruning.

Definition at line 84 of file dtb_stat.hpp.

39.261.3.3 ComponentMembership() [1/2]

```
int ComponentMembership ( ) const [inline]
```

Get the component membership of this node.

Definition at line 87 of file dtb_stat.hpp.

39.261.3.4 ComponentMembership() [2/2]

```
int& ComponentMembership ( ) [inline]
```

Modify the component membership of this node.

Definition at line 89 of file dtb_stat.hpp.

39.261.3.5 MaxNeighborDistance() [1/2]

```
double MaxNeighborDistance ( ) const [inline]
```

Get the maximum neighbor distance.

Definition at line 72 of file dtb_stat.hpp.

39.261.3.6 MaxNeighborDistance() [2/2]

```
double& MaxNeighborDistance ( ) [inline]
```

Modify the maximum neighbor distance.

Definition at line 74 of file dtb_stat.hpp.

39.261.3.7 MinNeighborDistance() [1/2]

```
double MinNeighborDistance ( ) const [inline]
```

Get the minimum neighbor distance.

Definition at line 77 of file dtb_stat.hpp.

39.261.3.8 MinNeighborDistance() [2/2]

```
double& MinNeighborDistance ( ) [inline]
```

Modify the minimum neighbor distance.

Definition at line 79 of file dtb_stat.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/ **dtb_stat.hpp**

39.262 DualTreeBoruvka< MetricType, MatType, TreeType > Class Template Reference

Performs the MST calculation using the Dual-Tree Boruvka algorithm, using any type of tree.

Public Types

- typedef TreeType< MetricType, **DTBStat**, MatType > **Tree**
Convenience typedef.

Public Member Functions

- **DualTreeBoruvka** (const MatType &dataset, const bool naive=false, const MetricType metric=MetricType())
Create the tree from the given dataset.
- **DualTreeBoruvka** (**Tree** *tree, const MetricType metric=MetricType())
*Create the **DualTreeBoruvka** (p. 1272) object with an already initialized tree.*
- **~DualTreeBoruvka** ()
Delete the tree, if it was created inside the object.
- void **ComputeMST** (arma::mat &results)
Iteratively find the nearest neighbor of each component until the MST is complete.

39.262.1 Detailed Description

```
template<typename MetricType = metric::EuclideanDistance, typename MatType = arma::mat, template< typename TreeMetricType,
typename TreeStatType, typename TreeMatType > class TreeType = tree::KDTree>
class mlpack::emst::DualTreeBoruvka< MetricType, MatType, TreeType >
```

Performs the MST calculation using the Dual-Tree Boruvka algorithm, using any type of tree.

For more information on the algorithm, see the following citation:

```
@inproceedings{
  author = {March, W.B., Ram, P., and Gray, A.G.},
  title = {{Fast Euclidean Minimum Spanning Tree: Algorithm, Analysis,
    Applications.}},
  booktitle = {Proceedings of the 16th ACM SIGKDD International Conference
    on Knowledge Discovery and Data Mining}
  series = {KDD 2010},
  year = {2010}
}
```

General usage of this class might be like this:

```
extern arma::mat data; // We want to find the MST of this dataset.
DualTreeBoruvka<> dtb(data); // Create the tree with default options.

// Find the MST.
arma::mat mstResults;
dtb.ComputeMST(mstResults);
```

More advanced usage of the class can use different types of trees, pass in an already-built tree, or compute the MST using the $O(n^2)$ naive algorithm.

Template Parameters

<i>MetricType</i>	The metric to use.
<i>MatType</i>	The type of data matrix to use.
<i>TreeType</i>	Type of tree to use. This should follow the TreeType policy API.

Definition at line 83 of file dtb.hpp.

39.262.2 Member Typedef Documentation

39.262.2.1 Tree

```
typedef TreeType<MetricType, DTBStat, MatType> Tree
```

Convenience typedef.

Definition at line 87 of file dtb.hpp.

39.262.3 Constructor & Destructor Documentation

39.262.3.1 DualTreeBoruvka() [1/2]

```
DualTreeBoruvka (
    const MatType & dataset,
    const bool naive = false,
    const MetricType metric = MetricType() )
```

Create the tree from the given dataset.

This copies the dataset to an internal copy, because tree-building modifies the dataset.

Parameters

<i>data</i>	Dataset to build a tree for.
<i>naive</i>	Whether the computation should be done in $O(n^2)$ naive mode.
<i>metric</i>	An optional instantiated metric to use.

39.262.3.2 DualTreeBoruvka() [2/2]

```
DualTreeBoruvka (
    Tree * tree,
    const MetricType metric = MetricType() )
```

Create the **DualTreeBoruvka** (p. 1272) object with an already initialized tree.

This will not copy the dataset, and can save a little processing power. Naive mode is not available as an option for this constructor; instead, to run naive computation, construct a tree with all the points in one leaf (i.e. leafSize = number of points).

Note

Because tree-building (at least with BinarySpaceTree) modifies the ordering of a matrix, be sure you pass the modified matrix to this object! In addition, mapping the points of the matrix back to their original indices is not done when this constructor is used.

Parameters

<i>tree</i>	Pre-built tree.
<i>metric</i>	An optional instantiated metric to use.

39.262.3.3 `~DualTreeBoruvka()`

`~ DualTreeBoruvka ()`

Delete the tree, if it was created inside the object.

39.262.4 Member Function Documentation

39.262.4.1 `ComputeMST()`

```
void ComputeMST (
    arma::mat & results )
```

Iteratively find the nearest neighbor of each component until the MST is complete.

The results will be a 3xN matrix (with N equal to the number of edges in the minimum spanning tree). The first row will contain the lesser index of the edge; the second row will contain the greater index of the edge; and the third row will contain the distance between the two edges.

Parameters

<i>results</i>	Matrix which results will be stored in.
----------------	---

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/ dtb.hpp`

39.263 EdgePair Class Reference

An edge pair is simply two indices and a distance.

Public Member Functions

- **EdgePair** (const size_t lesser, const size_t greater, const double dist)
*Initialize an **EdgePair** (p. 1275) with two indices and a distance.*
- double **Distance** () const
Get the distance.

- double & **Distance** ()
Modify the distance.
- size_t **Greater** () const
Get the greater index.
- size_t & **Greater** ()
Modify the greater index.
- size_t **Lesser** () const
Get the lesser index.
- size_t & **Lesser** ()
Modify the lesser index.

39.263.1 Detailed Description

An edge pair is simply two indices and a distance.

It is used as the basic element of an edge list when computing a minimum spanning tree.

Definition at line 28 of file edge_pair.hpp.

39.263.2 Constructor & Destructor Documentation

39.263.2.1 EdgePair()

```
EdgePair (
    const size_t lesser,
    const size_t greater,
    const double dist ) [inline]
```

Initialize an **EdgePair** (p. 1275) with two indices and a distance.

The indices are called lesser and greater, implying that they be sorted before calling Init. However, this is not necessary for functionality; it is just a way to keep the edge list organized in other code.

Definition at line 45 of file edge_pair.hpp.

References Log::Assert().

39.263.3 Member Function Documentation

39.263.3.1 Distance() [1/2]

```
double Distance ( ) const [inline]
```

Get the distance.

Definition at line 63 of file edge_pair.hpp.

39.263.3.2 Distance() [2/2]

```
double& Distance ( ) [inline]
```

Modify the distance.

Definition at line 65 of file edge_pair.hpp.

39.263.3.3 Greater() [1/2]

```
size_t Greater ( ) const [inline]
```

Get the greater index.

Definition at line 58 of file edge_pair.hpp.

39.263.3.4 Greater() [2/2]

```
size_t& Greater ( ) [inline]
```

Modify the greater index.

Definition at line 60 of file edge_pair.hpp.

39.263.3.5 Lesser() [1/2]

```
size_t Lesser ( ) const [inline]
```

Get the lesser index.

Definition at line 53 of file edge_pair.hpp.

39.263.3.6 Lesser() [2/2]

```
size_t& Lesser ( ) [inline]
```

Modify the lesser index.

Definition at line 55 of file edge_pair.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/ **edge_pair.hpp**

39.264 UnionFind Class Reference

A Union-Find data structure.

Public Member Functions

- **UnionFind** (const size_t size)
Construct the object with the given size.
- **~UnionFind** ()
Destroy the object (nothing to do).
- size_t **Find** (const size_t x)
Returns the component containing an element.
- void **Union** (const size_t x, const size_t y)
Union the components containing x and y.

39.264.1 Detailed Description

A Union-Find data structure.

See Cormen, Rivest, & Stein for details. The structure tracks the components of a graph. Each point in the graph is initially in its own component. Calling Union(x, y) unites the components indexed by x and y. Find(x) returns the index of the component containing point x.

Definition at line 30 of file union_find.hpp.

39.264.2 Constructor & Destructor Documentation

39.264.2.1 UnionFind()

```
UnionFind (
    const size_t size ) [inline]
```

Construct the object with the given size.

Definition at line 38 of file union_find.hpp.

39.264.2.2 ~UnionFind()

```
~ UnionFind ( ) [inline]
```

Destroy the object (nothing to do).

Definition at line 48 of file union_find.hpp.

39.264.3 Member Function Documentation

39.264.3.1 Find()

```
size_t Find (
    const size_t x ) [inline]
```

Returns the component containing an element.

Parameters

x	the component to be found
---	---------------------------

Returns

The index of the component containing x

Definition at line 56 of file union_find.hpp.

Referenced by UnionFind::Union().

39.264.3.2 Union()

```
void Union (
    const size_t x,
    const size_t y ) [inline]
```

Union the components containing x and y.

Parameters

x	one component
y	the other component

Definition at line 76 of file union_find.hpp.

References UnionFind::Find().

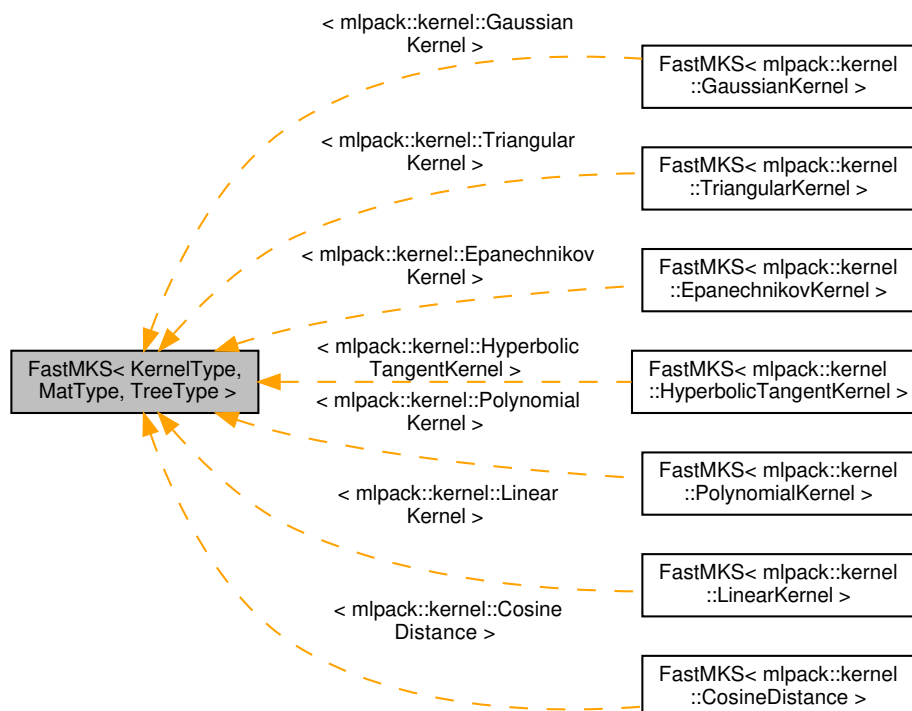
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/ **union_find.hpp**

39.265 FastMKS< KernelType, MatType, TreeType > Class Template Reference

An implementation of fast exact max-kernel search.

Inheritance diagram for FastMKS< KernelType, MatType, TreeType >:



Public Types

- typedef TreeType< **metric::IPMetric**< KernelType >, **FastMKSStat**, MatType > **Tree**

Convenience typedef.

Public Member Functions

- **FastMKS** (const bool singleMode=false, const bool naive=false)
*Create the **FastMKS** (p. 1280) object with an empty reference set and default kernel.*
- **FastMKS** (const MatType &referenceSet, const bool singleMode=false, const bool naive=false)
*Create the **FastMKS** (p. 1280) object with the given reference set (this is the set that is searched).*
- **FastMKS** (const MatType &referenceSet, KernelType &kernel, const bool singleMode=false, const bool naive=false)
*Create the **FastMKS** (p. 1280) object using the reference set (this is the set that is searched) with an initialized kernel.*
- **FastMKS** (MatType &&referenceSet, const bool singleMode=false, const bool naive=false)
*Create the **FastMKS** (p. 1280) object with the given reference set (this is the set that is searched), taking ownership of the reference set.*
- **FastMKS** (MatType &&referenceSet, KernelType &kernel, const bool singleMode=false, const bool naive=false)
*Create the **FastMKS** (p. 1280) object using the reference set (this is the set that is searched) with an initialized kernel, taking ownership of the reference set.*
- **FastMKS** (**Tree** *referenceTree, const bool singleMode=false)
*Create the **FastMKS** (p. 1280) object with an already-initialized tree built on the reference points.*
- **FastMKS** (const **FastMKS** &other)
Copy the parameters of the given model.
- **FastMKS** (**FastMKS** &&other)
Take ownership of the given model.
- **~FastMKS** ()
*Destructor for the **FastMKS** (p. 1280) object.*
- const **metric::IPMetric**< KernelType > & **Metric** () const
Get the inner-product metric induced by the given kernel.
- **metric::IPMetric**< KernelType > & **Metric** ()
Modify the inner-product metric induced by the given kernel.
- bool **Naive** () const
Get whether or not brute-force (naive) search is used.
- bool & **Naive** ()
Modify whether or not brute-force (naive) search is used.
- **FastMKS** & **operator=** (const **FastMKS** &other)
Assign this model to be a copy of the given model.
- void **Search** (const MatType &querySet, const size_t k, arma::Mat< size_t > &indices, arma::mat &kernels)
Search for the points in the reference set with maximum kernel evaluation to each point in the given query set.
- void **Search** (**Tree** *querySet, const size_t k, arma::Mat< size_t > &indices, arma::mat &kernels)
Search for the points in the reference set with maximum kernel evaluation to each point in the query set corresponding to the given pre-built query tree.
- void **Search** (const size_t k, arma::Mat< size_t > &indices, arma::mat &products)
Search for the maximum inner products of the query set (or if no query set was passed, the reference set is used).
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)

- *Serialize the model.*
- `bool SingleMode () const`
Get whether or not single-tree search is used.
- `bool & SingleMode ()`
Modify whether or not single-tree search is used.
- `void Train (const MatType &referenceSet)`
*"Train" the **FastMKS** (p. 1280) model on the given reference set (this will just build a tree, if the current search mode is not naive mode).*
- `void Train (const MatType &referenceSet, KernelType &kernel)`
*"Train" the **FastMKS** (p. 1280) model on the given reference set and use the given kernel.*
- `void Train (MatType &&referenceSet)`
*"Train" the **FastMKS** (p. 1280) model on the given reference set (this will just build a tree, if the current search mode is not naive mode).*
- `void Train (MatType &&referenceSet, KernelType &kernel)`
*"Train" the **FastMKS** (p. 1280) model on the given reference set and use the given kernel.*
- `void Train (Tree *referenceTree)`
*Train the **FastMKS** (p. 1280) model on the given reference tree.*

39.265.1 Detailed Description

```
template<typename KernelType, typename MatType = arma::mat, template< typename TreeMetricType, typename TreeStatType, type-
name TreeMatType > class TreeType = tree::StandardCoverTree>
class mlpack::fastmks::FastMKS< KernelType, MatType, TreeType >
```

An implementation of fast exact max-kernel search.

Given a query dataset and a reference dataset (or optionally just a reference dataset which is also used as the query dataset), fast exact max-kernel search finds, for each point in the query dataset, the k points in the reference set with maximum kernel value $K(p_q, p_r)$, where k is a specified parameter and $K()$ is a Mercer kernel.

For more information, see the following paper.

```
@inproceedings{curtin2013fast,
  title={Fast Exact Max-Kernel Search},
  author={Curtin, Ryan R. and Ram, Parikshit and Gray, Alexander G.},
  booktitle={Proceedings of the 2013 SIAM International Conference on Data
    Mining (SDM 13)},
  year={2013}
}
```

This class allows specification of the type of kernel and also of the type of tree. **FastMKS** (p. 1280) can be run on kernels that work on arbitrary objects – however, this only works with cover trees and other trees that are built only on points in the dataset (and not centroids of regions or anything like that).

Template Parameters

<i>KernelType</i>	Type of kernel to run FastMKS (p. 1280) with.
<i>MatType</i>	Type of data matrix (usually <code>arma::mat</code>).
<i>TreeType</i>	Type of tree to run FastMKS (p. 1280) with; it must satisfy the <code>TreeType</code> policy API.

Definition at line 63 of file fastmks.hpp.

39.265.2 Member Typedef Documentation

39.265.2.1 Tree

```
typedef TreeType< metric::IPMetric<KernelType>, FastMKStat, MatType> Tree
```

Convenience typedef.

Definition at line 67 of file fastmks.hpp.

39.265.3 Constructor & Destructor Documentation

39.265.3.1 FastMKS() [1/8]

```
FastMKS (
    const bool singleMode = false,
    const bool naive = false )
```

Create the **FastMKS** (p. 1280) object with an empty reference set and default kernel.

Make sure to call **Train()** (p. 1289) before **Search()** (p. 1287) is called!

Parameters

<i>singleMode</i>	Whether or not to run single-tree search.
<i>naive</i>	Whether or not to run brute-force (naive) search.

39.265.3.2 FastMKS() [2/8]

```
FastMKS (
    const MatType & referenceSet,
    const bool singleMode = false,
    const bool naive = false )
```

Create the **FastMKS** (p. 1280) object with the given reference set (this is the set that is searched).

Optionally, specify whether or not single-tree search or naive (brute-force) search should be used.

Parameters

<i>referenceSet</i>	Set of reference data.
<i>singleMode</i>	Whether or not to run single-tree search.
<i>naive</i>	Whether or not to run brute-force (naive) search.

39.265.3.3 **FastMKS()** [3/8]

```
FastMKS (
    const MatType & referenceSet,
    KernelType & kernel,
    const bool singleMode = false,
    const bool naive = false )
```

Create the **FastMKS** (p. 1280) object using the reference set (this is the set that is searched) with an initialized kernel.

This is useful for when the kernel stores state. Optionally, specify whether or not single-tree search or naive (brute-force) search should be used.

Parameters

<i>referenceSet</i>	Reference set of data for FastMKS (p. 1280).
<i>kernel</i>	Initialized kernel.
<i>single</i>	Whether or not to run single-tree search.
<i>naive</i>	Whether or not to run brute-force (naive) search.

39.265.3.4 **FastMKS()** [4/8]

```
FastMKS (
    MatType && referenceSet,
    const bool singleMode = false,
    const bool naive = false )
```

Create the **FastMKS** (p. 1280) object with the given reference set (this is the set that is searched), taking ownership of the reference set.

Optionally, specify whether or not single-tree search or naive (brute-force) search should be used.

Parameters

<i>referenceSet</i>	Set of reference data.
<i>singleMode</i>	Whether or not to run single-tree search.
<i>naive</i>	Whether or not to run brute-force (naive) search.

39.265.3.5 FastMKS() [5/8]

```

FastMKS (
    MatType && referenceSet,
    KernelType & kernel,
    const bool singleMode = false,
    const bool naive = false )

```

Create the **FastMKS** (p. 1280) object using the reference set (this is the set that is searched) with an initialized kernel, taking ownership of the reference set.

This is useful for when the kernel stores state. Optionally, specify whether or not single-tree search or naive (brute-force) search should be used.

Parameters

<i>referenceSet</i>	Reference set of data for FastMKS (p. 1280).
<i>kernel</i>	Initialized kernel.
<i>single</i>	Whether or not to run single-tree search.
<i>naive</i>	Whether or not to run brute-force (naive) search.

39.265.3.6 FastMKS() [6/8]

```

FastMKS (
    Tree * referenceTree,
    const bool singleMode = false )

```

Create the **FastMKS** (p. 1280) object with an already-initialized tree built on the reference points.

Be sure that the tree is built with the metric type IPMetric<KernelType>. Optionally, whether or not to run single-tree search can be specified. Brute-force search is not available with this constructor since a tree is given (use one of the other constructors).

Parameters

<i>referenceTree</i>	Tree built on reference data.
<i>single</i>	Whether or not to run single-tree search.
<i>naive</i>	Whether or not to run brute-force (naive) search.

39.265.3.7 FastMKS() [7/8]

```
FastMKS (
    const FastMKS< KernelType, MatType, TreeType > & other )
```

Copy the parameters of the given model.

39.265.3.8 FastMKS() [8/8]

```
FastMKS (
    FastMKS< KernelType, MatType, TreeType > && other )
```

Take ownership of the given model.

39.265.3.9 ~FastMKS()

```
~ FastMKS ( )
```

Destructor for the **FastMKS** (p. 1280) object.

39.265.4 Member Function Documentation**39.265.4.1 Metric()** [1/2]

```
const metric::IPMetric<KernelType>& Metric ( ) const [inline]
```

Get the inner-product metric induced by the given kernel.

Definition at line 287 of file fastmks.hpp.

39.265.4.2 Metric() [2/2]

```
metric::IPMetric<KernelType>& Metric ( ) [inline]
```

Modify the inner-product metric induced by the given kernel.

Definition at line 289 of file fastmks.hpp.

39.265.4.3 Naive() [1/2]

```
bool Naive ( ) const [inline]
```

Get whether or not brute-force (naive) search is used.

Definition at line 297 of file fastmks.hpp.

39.265.4.4 Naive() [2/2]

```
bool& Naive ( ) [inline]
```

Modify whether or not brute-force (naive) search is used.

Definition at line 299 of file fastmks.hpp.

39.265.4.5 operator=()

```
FastMKS& operator= (
    const FastMKS< KernelType, MatType, TreeType > & other )
```

Assign this model to be a copy of the given model.

39.265.4.6 Search() [1/3]

```
void Search (
    const MatType & querySet,
    const size_t k,
    arma::Mat< size_t > & indices,
    arma::mat & kernels )
```

Search for the points in the reference set with maximum kernel evaluation to each point in the given query set.

The resulting kernel evaluations are stored in the kernels matrix, and the corresponding point indices are stored in the indices matrix. The results for each point in the query set are stored in the corresponding column of the kernels and products matrices; for instance, the index of the point with maximum kernel evaluation to point 4 in the query set will be stored in row 0 and column 4 of the indices matrix.

If querySet only contains a few points, the extra overhead of building a tree to perform dual-tree search may not be warranted, and it may be faster to use single-tree search, either by setting singleMode to false in the constructor or with **SingleMode()** (p. 1289).

Parameters

<i>querySet</i>	Set of query points (can be a single point).
<i>k</i>	The number of maximum kernels to find.
<i>indices</i>	Matrix to store resulting indices of max-kernel search in.
<i>kernels</i>	Matrix to store resulting max-kernel values in.

39.265.4.7 Search() [2/3]

```
void Search (
    Tree * querySet,
    const size_t k,
    arma::Mat< size_t > & indices,
    arma::mat & kernels )
```

Search for the points in the reference set with maximum kernel evaluation to each point in the query set corresponding to the given pre-built query tree.

The resulting kernel evaluations are stored in the kernels matrix, and the corresponding point indices are stored in the indices matrix. The results for each point in the query set are stored in the corresponding column of the kernels and products matrices; for instance, the index of the point with maximum kernel evaluation to point 4 in the query set will be stored in row 0 and column 4 of the indices matrix.

This will throw an exception if called while the **FastMKS** (p. 1280) object has 'single' set to true.

Be aware that if your tree modifies the original input matrix, the results here are with respect to the modified input matrix (that is, queryTree->Dataset()).

Parameters

<i>queryTree</i>	Tree built on query points.
<i>k</i>	The number of maximum kernels to find.
<i>indices</i>	Matrix to store resulting indices of max-kernel search in.
<i>kernels</i>	Matrix to store resulting max-kernel values in.

39.265.4.8 Search() [3/3]

```
void Search (
    const size_t k,
    arma::Mat< size_t > & indices,
    arma::mat & products )
```

Search for the maximum inner products of the query set (or if no query set was passed, the reference set is used).

The resulting maximum inner products are stored in the products matrix and the corresponding point indices are stores in the indices matrix. The results for each point in the query set are stored in the corresponding column of the indices and products matrices; for instance, the index of the point with maximum inner product to point 4 in the query set will be stored in row 0 and column 4 of the indices matrix.

Parameters

<i>k</i>	The number of maximum kernels to find.
<i>indices</i>	Matrix to store resulting indices of max-kernel search in.
<i>products</i>	Matrix to store resulting max-kernel values in.

39.265.4.9 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the model.

Referenced by FastMKS< mlpack::kernel::CosineDistance >::Naive().

39.265.4.10 SingleMode() [1/2]

```
bool SingleMode ( ) const [inline]
```

Get whether or not single-tree search is used.

Definition at line 292 of file fastmks.hpp.

39.265.4.11 SingleMode() [2/2]

```
bool& SingleMode ( ) [inline]
```

Modify whether or not single-tree search is used.

Definition at line 294 of file fastmks.hpp.

39.265.4.12 Train() [1/5]

```
void Train (
    const MatType & referenceSet )
```

"Train" the **FastMKS** (p. 1280) model on the given reference set (this will just build a tree, if the current search mode is not naive mode).

Parameters

<i>referenceSet</i>	Set of reference points.
---------------------	--------------------------

39.265.4.13 **Train()** [2/5]

```
void Train (
    const MatType & referenceSet,
    KernelType & kernel )
```

"Train" the **FastMKS** (p. 1280) model on the given reference set and use the given kernel.

This will just build a tree and replace the metric, if the current search mode is not naive mode.

Parameters

<i>referenceSet</i>	Set of reference points.
<i>kernel</i>	Kernel to use for search.

39.265.4.14 **Train()** [3/5]

```
void Train (
    MatType && referenceSet )
```

"Train" the **FastMKS** (p. 1280) model on the given reference set (this will just build a tree, if the current search mode is not naive mode).

This takes ownership of the reference set.

Parameters

<i>referenceSet</i>	Set of reference points.
---------------------	--------------------------

39.265.4.15 **Train()** [4/5]

```
void Train (
    MatType && referenceSet,
    KernelType & kernel )
```

"Train" the **FastMKS** (p. 1280) model on the given reference set and use the given kernel.

This will just build a tree and replace the metric, if the current search mode is not naive mode. This takes ownership of the reference set.

Parameters

<i>referenceSet</i>	Set of reference points.
<i>kernel</i>	Kernel to use for search.

39.265.4.16 Train() [5/5]

```
void Train (
    Tree * referenceTree )
```

Train the **FastMKS** (p. 1280) model on the given reference tree.

This takes ownership of the tree, so you do not need to delete it! This will throw an exception if the model is searching in naive mode (i.e. if **Naive()** (p. 1287) == true).

Parameters

<i>tree</i>	Tree to use as reference data.
-------------	--------------------------------

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/fastmks/ **fastmks.hpp**

39.266 FastMKModel Class Reference

A utility struct to contain all the possible **FastMKS** (p. 1280) models, for use by the mlpack_fastmks program.

Public Types

- enum **KernelTypes** {
LINEAR_KERNEL,
POLYNOMIAL_KERNEL,
COSINE_DISTANCE,
GAUSSIAN_KERNEL,
EPANECHNIKOV_KERNEL,
TRIANGULAR_KERNEL,
HYPTAN_KERNEL }

A list of all the kernels we support.

Public Member Functions

- **FastMKModel** (const int kernelType= **LINEAR_KERNEL**)
*Create the **FastMKModel** (p. 1291) with the given kernel type.*
- **FastMKModel** (const **FastMKModel** &other)
Copy constructor.
- **FastMKModel** (**FastMKModel** &&other)
Move constructor.
- **~FastMKModel** ()
Clean memory.
- template<typename TKernelType >
void **BuildModel** (arma::mat &&referenceData, TKernelType &kernel, const bool singleMode, const bool naive, const double base)
Build the model on the given reference set.
- int **KernelType** () const
Get the kernel type.
- int & **KernelType** ()
Modify the kernel type.
- bool **Naive** () const
Get whether or not naive search is used.
- bool & **Naive** ()
Set whether or not naive search is used.
- **FastMKModel** & **operator=** (const **FastMKModel** &other)
Copy assignment operator.
- void **Search** (const arma::mat &querySet, const size_t k, arma::Mat< size_t > &indices, arma::mat &kernels, const double base)
Search with a different query set.
- void **Search** (const size_t k, arma::Mat< size_t > &indices, arma::mat &kernels)
Search with the reference set as the query set.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the model.
- bool **SingleMode** () const
Get whether or not single-tree search is used.
- bool & **SingleMode** ()
Set whether or not single-tree search is used.

39.266.1 Detailed Description

A utility struct to contain all the possible **FastMKS** (p. 1280) models, for use by the `mlpack_fastmks` program.

Definition at line 34 of file `fastmks_model.hpp`.

39.266.2 Member Enumeration Documentation

39.266.2.1 KernelTypes

enum **KernelTypes**

A list of all the kernels we support.

Enumerator

LINEAR_KERNEL	
POLYNOMIAL_KERNEL	
COSINE_DISTANCE	
GAUSSIAN_KERNEL	
EPANECHNIKOV_KERNEL	
TRIANGULAR_KERNEL	
HYPTAN_KERNEL	

Definition at line 38 of file fastmks_model.hpp.

39.266.3 Constructor & Destructor Documentation

39.266.3.1 FastMKModel() [1/3]

```
FastMKModel (  
    const int kernelType = LINEAR_KERNEL )
```

Create the **FastMKModel** (p. 1291) with the given kernel type.

39.266.3.2 FastMKModel() [2/3]

```
FastMKModel (  
    const FastMKModel & other )
```

Copy constructor.

39.266.3.3 FastMKModel() [3/3]

```
FastMKModel (  
    FastMKModel && other )
```

Move constructor.

39.266.3.4 ~FastMKModel()

~ **FastMKModel** ()

Clean memory.

39.266.4 Member Function Documentation

39.266.4.1 BuildModel()

```
void BuildModel (
    arma::mat && referenceData,
    TKernelType & kernel,
    const bool singleMode,
    const bool naive,
    const double base )
```

Build the model on the given reference set.

Make sure `kernelType` is equal to the correct entry in `KernelTypes` for the given `KernelType` class!

39.266.4.2 KernelType() [1/2]

```
int KernelType ( ) const [inline]
```

Get the kernel type.

Definition at line 90 of file `fastmks_model.hpp`.

39.266.4.3 KernelType() [2/2]

```
int& KernelType ( ) [inline]
```

Modify the kernel type.

Definition at line 92 of file `fastmks_model.hpp`.

References `FastMKModel::Search()`, and `FastMKModel::serialize()`.

39.266.4.4 Naive() [1/2]

```
bool Naive ( ) const
```

Get whether or not naive search is used.

39.266.4.5 Naive() [2/2]

```
bool& Naive ( )
```

Set whether or not naive search is used.

39.266.4.6 operator=()

```
FastMKModel& operator= (
    const FastMKModel & other )
```

Copy assignment operator.

39.266.4.7 Search() [1/2]

```
void Search (
    const arma::mat & querySet,
    const size_t k,
    arma::Mat< size_t > & indices,
    arma::mat & kernels,
    const double base )
```

Search with a different query set.

Parameters

<i>querySet</i>	Set to search with.
<i>k</i>	Number of max-kernel candidates to search for.
<i>indices</i>	A matrix in which to store the indices of max-kernel candidates.
<i>kernels</i>	A matrix in which to store the max-kernel candidate kernel values.
<i>base</i>	Base to use for cover tree building (if in dual-tree search mode).

Referenced by `FastMKModel::KernelType()`.

39.266.4.8 Search() [2/2]

```
void Search (
    const size_t k,
    arma::Mat< size_t > & indices,
    arma::mat & kernels )
```

Search with the reference set as the query set.

Parameters

<i>k</i>	Number of max-kernel candidates to search for.
<i>indices</i>	A matrix in which to store the indices of max-kernel candidates.
<i>kernels</i>	A matrix in which to store the max-kernel candidate kernel values.

39.266.4.9 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the model.

Referenced by FastMKModel::KernelType().

39.266.4.10 SingleMode() [1/2]

```
bool SingleMode ( ) const
```

Get whether or not single-tree search is used.

39.266.4.11 SingleMode() [2/2]

```
bool& SingleMode ( )
```

Set whether or not single-tree search is used.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/fastmks/ **fastmks_model.hpp**

39.267 FastMKSRules< KernelType, TreeType > Class Template Reference

The **FastMKSRules** (p. 1298) class is a template helper class used by **FastMKS** (p. 1280) class when performing exact max-kernel search.

Public Types

- typedef **tree::TraversallInfo**< TreeType > **TraversallInfoType**

Public Member Functions

- **FastMKSRules** (const typename TreeType::Mat &referenceSet, const typename TreeType::Mat &querySet, const size_t k, KernelType &kernel)
*Construct the **FastMKSRules** (p. 1298) object.*
- double **BaseCase** (const size_t queryIndex, const size_t referenceIndex)
Compute the base case (kernel value) between two points.
- size_t **BaseCases** () const
*Get the number of times **BaseCase()** (p. 1300) was called.*
- size_t & **BaseCases** ()
*Modify the number of times **BaseCase()** (p. 1300) was called.*
- void **GetResults** (arma::Mat< size_t > &indices, arma::mat &products)
Store the list of candidates for each query point in the given matrices.
- double **Rescore** (const size_t queryIndex, TreeType &referenceNode, const double oldScore) const
Re-evaluate the score for recursion order.
- double **Rescore** (TreeType &queryNode, TreeType &referenceNode, const double oldScore) const
Re-evaluate the score for recursion order.
- double **Score** (const size_t queryIndex, TreeType &referenceNode)
Get the score for recursion order.
- double **Score** (TreeType &queryNode, TreeType &referenceNode)
Get the score for recursion order.
- size_t **Scores** () const
*Get the number of times **Score()** (p. 1301) was called.*
- size_t & **Scores** ()
*Modify the number of times **Score()** (p. 1301) was called.*
- const **TraversallInfoType** & **TraversallInfo** () const
- **TraversallInfoType** & **TraversallInfo** ()

39.267.1 Detailed Description

```
template<typename KernelType, typename TreeType>
class mlpack::fastmks::FastMKSRules< KernelType, TreeType >
```

The **FastMKSRules** (p. 1298) class is a template helper class used by **FastMKS** (p. 1280) class when performing exact max-kernel search.

For each point in the query dataset, it keeps track of the k best candidates in the reference dataset.

Template Parameters

<i>KernelType</i>	Type of kernel to run FastMKS (p. 1280) with.
<i>TreeType</i>	Type of tree to run FastMKS (p. 1280) with; it must satisfy the <i>TreeType</i> policy API.

Definition at line 34 of file `fastmks_rules.hpp`.

39.267.2 Member Typedef Documentation

39.267.2.1 TraversalInfoType

```
typedef tree::TraversalInfo<TreeType> TraversalInfoType
```

Definition at line 122 of file `fastmks_rules.hpp`.

39.267.3 Constructor & Destructor Documentation

39.267.3.1 FastMKSRules()

```
FastMKSRules (
    const typename TreeType::Mat & referenceSet,
    const typename TreeType::Mat & querySet,
    const size_t k,
    KernelType & kernel )
```

Construct the **FastMKSRules** (p. 1298) object.

This is usually done from within the **FastMKS** (p. 1280) class at search time.

Parameters

<i>referenceSet</i>	Set of reference data.
<i>querySet</i>	Set of query data.
<i>k</i>	Number of candidates to search for.
<i>kernel</i>	Kernel to run FastMKS (p. 1280) with.

39.267.4 Member Function Documentation

39.267.4.1 BaseCase()

```
double BaseCase (
    const size_t queryIndex,
    const size_t referenceIndex )
```

Compute the base case (kernel value) between two points.

39.267.4.2 BaseCases() [1/2]

```
size_t BaseCases ( ) const [inline]
```

Get the number of times **BaseCase()** (p. 1300) was called.

Definition at line 113 of file fastmks_rules.hpp.

39.267.4.3 BaseCases() [2/2]

```
size_t& BaseCases ( ) [inline]
```

Modify the number of times **BaseCase()** (p. 1300) was called.

Definition at line 115 of file fastmks_rules.hpp.

39.267.4.4 GetResults()

```
void GetResults (
    arma::Mat< size_t > & indices,
    arma::mat & products )
```

Store the list of candidates for each query point in the given matrices.

Parameters

<i>indices</i>	Matrix storing lists of candidate for each query point.
<i>products</i>	Matrix storing kernel value for each candidate.

39.267.4.5 Rescore() [1/2]

```
double Rescore (
    const size_t queryIndex,
    TreeType & referenceNode,
    const double oldScore ) const
```

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that a node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.
<i>oldScore</i>	Old score produced by Score() (p. 1301) (or Rescore() (p. 1301)).

39.267.4.6 Rescore() [2/2]

```
double Rescore (
    TreeType & queryNode,
    TreeType & referenceNode,
    const double oldScore ) const
```

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that a node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

Parameters

<i>queryNode</i>	Candidate query node to be recursed into.
<i>referenceNode</i>	Candidate reference node to be recursed into.
<i>oldScore</i>	Old score produced by Score() (p. 1301) (or Rescore() (p. 1301)).

39.267.4.7 Score() [1/2]

```
double Score (
```

```
const size_t queryIndex,
TreeType & referenceNode )
```

Get the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate to be recursed into.

39.267.4.8 Score() [2/2]

```
double Score (
    TreeType & queryNode,
    TreeType & referenceNode )
```

Get the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryNode</i>	Candidate query node to be recursed into.
<i>referenceNode</i>	Candidate reference node to be recursed into.

39.267.4.9 Scores() [1/2]

```
size_t Scores ( ) const [inline]
```

Get the number of times **Score()** (p. 1301) was called.

Definition at line 118 of file fastmks_rules.hpp.

39.267.4.10 Scores() [2/2]

```
size_t& Scores ( ) [inline]
```

Modify the number of times **Score()** (p. 1301) was called.

Definition at line 120 of file fastmks_rules.hpp.

39.267.4.11 TraversalInfo() [1/2]

```
const TraversalInfoType& TraversalInfo ( ) const [inline]
```

Definition at line 124 of file fastmks_rules.hpp.

39.267.4.12 TraversalInfo() [2/2]

```
TraversalInfoType& TraversalInfo ( ) [inline]
```

Definition at line 125 of file fastmks_rules.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/fastmks/ **fastmks_rules.hpp**

39.268 FastMKSSStat Class Reference

The statistic used in trees with **FastMKS** (p. 1280).

Public Member Functions

- **FastMKSSStat** ()
Default initialization.
- template<typename TreeType >
FastMKSSStat (const TreeType &node)
Initialize this statistic for the given tree node.
- double **Bound** () const
Get the bound.
- double & **Bound** ()
Modify the bound.
- double **LastKernel** () const
Get the last kernel evaluation.
- double & **LastKernel** ()
Modify the last kernel evaluation.
- void * **LastKernelNode** () const
Get the address of the node corresponding to the last distance evaluation.
- void *& **LastKernelNode** ()
Modify the address of the node corresponding to the last distance evaluation.
- double **SelfKernel** () const
Get the self-kernel.
- double & **SelfKernel** ()
Modify the self-kernel.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the statistic.

39.268.1 Detailed Description

The statistic used in trees with **FastMKS** (p. 1280).

This stores both the bound and the self-kernels for each node in the tree.

Definition at line 25 of file `fastmks_stat.hpp`.

39.268.2 Constructor & Destructor Documentation

39.268.2.1 `FastMKStat()` [1/2]

```
FastMKStat ( ) [inline]
```

Default initialization.

Definition at line 31 of file `fastmks_stat.hpp`.

39.268.2.2 `FastMKStat()` [2/2]

```
FastMKStat (  
    const TreeType & node ) [inline]
```

Initialize this statistic for the given tree node.

The TreeType's metric better be IPMetric with some kernel type (that is, `Metric().Kernel()` must exist).

Parameters

<i>node</i>	Node that this statistic is built for.
-------------	--

Definition at line 46 of file `fastmks_stat.hpp`.

39.268.3 Member Function Documentation

39.268.3.1 Bound() [1/2]

```
double Bound ( ) const [inline]
```

Get the bound.

Definition at line 86 of file fastmks_stat.hpp.

39.268.3.2 Bound() [2/2]

```
double& Bound ( ) [inline]
```

Modify the bound.

Definition at line 88 of file fastmks_stat.hpp.

39.268.3.3 LastKernel() [1/2]

```
double LastKernel ( ) const [inline]
```

Get the last kernel evaluation.

Definition at line 91 of file fastmks_stat.hpp.

39.268.3.4 LastKernel() [2/2]

```
double& LastKernel ( ) [inline]
```

Modify the last kernel evaluation.

Definition at line 93 of file fastmks_stat.hpp.

39.268.3.5 LastKernelNode() [1/2]

```
void* LastKernelNode ( ) const [inline]
```

Get the address of the node corresponding to the last distance evaluation.

Definition at line 96 of file fastmks_stat.hpp.

39.268.3.6 LastKernelNode() [2/2]

```
void*& LastKernelNode ( ) [inline]
```

Modify the address of the node corresponding to the last distance evaluation.

Definition at line 99 of file fastmks_stat.hpp.

39.268.3.7 SelfKernel() [1/2]

```
double SelfKernel ( ) const [inline]
```

Get the self-kernel.

Definition at line 81 of file fastmks_stat.hpp.

39.268.3.8 SelfKernel() [2/2]

```
double& SelfKernel ( ) [inline]
```

Modify the self-kernel.

Definition at line 83 of file fastmks_stat.hpp.

39.268.3.9 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the statistic.

Definition at line 103 of file fastmks_stat.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/fastmks/ **fastmks_stat.hpp**

39.269 DiagonalConstraint Class Reference

Force a covariance matrix to be diagonal.

Static Public Member Functions

- static void **ApplyConstraint** (arma::mat &covariance)
Force a covariance matrix to be diagonal.
- static void **ApplyConstraint** (arma::vec &diagCovariance)
Apply the diagonal constraint to the given diagonal covariance matrix (which is represented as a vector), and ensure each value on the diagonal is at least 1e-10.
- template<typename Archive >
static void **serialize** (Archive &, const unsigned int)
Serialize the constraint (which holds nothing, so, nothing to do).

39.269.1 Detailed Description

Force a covariance matrix to be diagonal.

Definition at line 23 of file diagonal_constraint.hpp.

39.269.2 Member Function Documentation

39.269.2.1 ApplyConstraint() [1/2]

```
static void ApplyConstraint (  
    arma::mat & covariance ) [inline], [static]
```

Force a covariance matrix to be diagonal.

Definition at line 27 of file diagonal_constraint.hpp.

39.269.2.2 ApplyConstraint() [2/2]

```
static void ApplyConstraint (  
    arma::vec & diagCovariance ) [inline], [static]
```

Apply the diagonal constraint to the given diagonal covariance matrix (which is represented as a vector), and ensure each value on the diagonal is at least 1e-10.

Definition at line 38 of file diagonal_constraint.hpp.

39.269.2.3 serialize()

```
static void serialize (
    Archive & ,
    const unsigned int ) [inline], [static]
```

Serialize the constraint (which holds nothing, so, nothing to do).

Definition at line 47 of file diagonal_constraint.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/ **diagonal_constraint.hpp**

39.270 DiagonalGMM Class Reference

A Diagonal Gaussian Mixture Model.

Public Member Functions

- **DiagonalGMM** ()
Create an empty Diagonal Gaussian Mixture Model, with zero gaussians.
- **DiagonalGMM** (const size_t gaussians, const size_t dimensionality)
*Create a **GMM** (p. 1323) with the given number of Gaussians, each of which have the specified dimensionality.*
- **DiagonalGMM** (const std::vector< **distribution::DiagonalGaussianDistribution** > &dists, const arma::vec &weights)
*Create a **DiagonalGMM** (p. 1308) with the given dists and weights.*
- **DiagonalGMM** (const **DiagonalGMM** &other)
Copy constructor for DiagonalGMMs.
- void **Classify** (const arma::mat &observations, arma::Row< size_t > &labels) const
*Classify the given observations as being from an individual component in this **DiagonalGMM** (p. 1308).*
- const **distribution::DiagonalGaussianDistribution** & **Component** (size_t i) const
Return a const reference to a component distribution.
- **distribution::DiagonalGaussianDistribution** & **Component** (size_t i)
Return a reference to a component distribution.
- size_t **Dimensionality** () const
Return the dimensionality of the model.
- size_t **Gaussians** () const
Return the number of Gaussians in the model.
- double **LogProbability** (const arma::vec &observation) const
Return the log probability that the given observation came from this distribution.
- double **LogProbability** (const arma::vec &observation, const size_t component) const
Return the log probability that the given observation came from the given Gaussian component in this distribution.
- **DiagonalGMM** & **operator=** (const **DiagonalGMM** &other)
Copy operator for DiagonalGMMs.

- double **Probability** (const arma::vec &observation) const
Return the probability that the given observation came from this distribution.
- double **Probability** (const arma::vec &observation, const size_t component) const
Return the probability that the given observation came from the given Gaussian component in this distribution.
- arma::vec **Random** () const
Return a randomly generated observation according to the probability distribution defined by this object.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
*Serialize the **DiagonalGMM** (p. 1308).*
- template<typename FittingType = EMFit<kmeans::KMeans<>, DiagonalConstraint, distribution::DiagonalGaussianDistribution>>
double **Train** (const arma::mat &observations, const size_t trials=1, const bool useExistingModel=false, FittingType←
Type fitter=FittingType())
Estimate the probability distribution directly from the given observations, using the given algorithm in the FittingType class to fit the data.
- template<typename FittingType = EMFit<kmeans::KMeans<>, DiagonalConstraint, distribution::DiagonalGaussianDistribution>>
double **Train** (const arma::mat &observations, const arma::vec &probabilities, const size_t trials=1, const bool useExistingModel=false, FittingType fitter=FittingType())
Estimate the probability distribution directly from the given observations, taking into account the probability of each observation actually being from this distribution, and using the given algorithm in the FittingType class to fit the data.
- const arma::vec & **Weights** () const
Return a const reference to the a priori weights of each Gaussian.
- arma::vec & **Weights** ()
Return a reference to the a priori weights of each Gaussian.

39.270.1 Detailed Description

A Diagonal Gaussian Mixture Model.

This class uses maximum likelihood loss functions to estimate the parameters of the **DiagonalGMM** (p. 1308) on a given dataset via the given fitting mechanism, defined by the FittingType template parameter. The **DiagonalGMM** (p. 1308) can be trained using normal data, or data with probabilities of being from this **GMM** (p. 1323) (see **DiagonalGMM::Train()** (p. 1315) for more information). The **DiagonalGMM** (p. 1308) is the same as **GMM** (p. 1323) except for wrapping gmm_diag class.

The **Train()** (p. 1315) method uses a template type 'FittingType'. The FittingType template class must provide a way for the **DiagonalGMM** (p. 1308) to train on data. It must provide the following two functions:

```
void Estimate(
    const arma::mat& observations,
    std::vector<distribution::DiagonalGaussianDistribution>& dists,
    arma::vec& weights);

void Estimate(
    const arma::mat& observations,
    const arma::vec& probabilities,
    std::vector<distribution::DiagonalGaussianDistribution>& dists,
    arma::vec& weights);
```

Example use:

```
// Set up a mixture of 5 gaussians in a 4-dimensional space.
DiagonalGMM g(5, 4);

// Train the DiagonalGMM given the data observations, using the default
// EM fitting mechanism.

g.Train(data);

// Get the probability of 'observation' being observed from this
// DiagonalGMM.
double probability = g.Probability(observation);

// Get a random observation from the DiagonalGMM.
arma::vec observation = g.Random();
```

Definition at line 74 of file diagonal_gmm.hpp.

39.270.2 Constructor & Destructor Documentation

39.270.2.1 DiagonalGMM() [1/4]

```
DiagonalGMM ( ) [inline]
```

Create an empty Diagonal Gaussian Mixture Model, with zero gaussians.

Definition at line 92 of file diagonal_gmm.hpp.

References Log::Debug.

Referenced by DiagonalGMM::DiagonalGMM().

39.270.2.2 DiagonalGMM() [2/4]

```
DiagonalGMM (
    const size_t gaussians,
    const size_t dimensionality )
```

Create a **GMM** (p. 1323) with the given number of Gaussians, each of which have the specified dimensionality.

The means and covariances will be set to 0.

Parameters

<i>gaussians</i>	Number of Gaussians in this DiagonalGMM (p. 1308).
<i>dimensionality</i>	Dimensionality of each Gaussian.

39.270.2.3 DiagonalGMM() [3/4]

```
DiagonalGMM (
    const std::vector< distribution::DiagonalGaussianDistribution > & dists,
    const arma::vec & weights ) [inline]
```

Create a **DiagonalGMM** (p. 1308) with the given *dists* and *weights*.

Parameters

<i>dists</i>	Distributions of the model.
<i>weights</i>	Weights of the model.

Definition at line 118 of file `diagonal_gmm.hpp`.

References `DiagonalGMM::DiagonalGMM()`, and `DiagonalGMM::operator=()`.

39.270.2.4 DiagonalGMM() [4/4]

```
DiagonalGMM (
    const DiagonalGMM & other )
```

Copy constructor for **DiagonalGMMs**.

39.270.3 Member Function Documentation

39.270.3.1 Classify()

```
void Classify (
    const arma::mat & observations,
    arma::Row< size_t > & labels ) const
```

Classify the given observations as being from an individual component in this **DiagonalGMM** (p. 1308).

The resultant classifications are stored in the 'labels' object, and each label will be between 0 and (**Gaussians()** (p. 1312) - 1). Supposing that a point was classified with label 2, and that our **DiagonalGMM** (p. 1308) object was called 'dgmm', one could access the relevant Gaussian distribution as follows:

```
arma::vec mean = dgmm.Means()[2];
arma::mat covariance = dgmm.Covariances()[2];
double priorWeight = dgmm.Weights()[2];
```

Parameters

<i>observations</i>	Matrix of observations to classify.
<i>labels</i>	Object which will be filled with labels.

Referenced by DiagonalGMM::Weights().

39.270.3.2 Component() [1/2]

```
const distribution::DiagonalGaussianDistribution& Component (
    size_t i ) const [inline]
```

Return a const reference to a component distribution.

Parameters

<i>i</i>	Index of component.
----------	---------------------

Definition at line 141 of file diagonal_gmm.hpp.

39.270.3.3 Component() [2/2]

```
distribution::DiagonalGaussianDistribution& Component (
    size_t i ) [inline]
```

Return a reference to a component distribution.

Parameters

<i>i</i>	Index of component.
----------	---------------------

Definition at line 151 of file diagonal_gmm.hpp.

39.270.3.4 Dimensionality()

```
size_t Dimensionality ( ) const [inline]
```

Return the dimensionality of the model.

Definition at line 134 of file diagonal_gmm.hpp.

39.270.3.5 Gaussians()

```
size_t Gaussians ( ) const [inline]
```

Return the number of Gaussians in the model.

Definition at line 132 of file diagonal_gmm.hpp.

39.270.3.6 LogProbability() [1/2]

```
double LogProbability (
    const arma::vec & observation ) const
```

Return the log probability that the given observation came from this distribution.

Parameters

<i>observation</i>	Observation to evaluate the probability of.
--------------------	---

Referenced by DiagonalGMM::Weights().

39.270.3.7 LogProbability() [2/2]

```
double LogProbability (
    const arma::vec & observation,
    const size_t component ) const
```

Return the log probability that the given observation came from the given Gaussian component in this distribution.

Parameters

<i>observation</i>	Observation to evaluate the probability of.
<i>component</i>	Index of the component of the DiagonalGMM (p. 1308).

39.270.3.8 operator=()

```
DiagonalGMM& operator= (
    const DiagonalGMM & other )
```

Copy operator for DiagonalGMMs.

Referenced by DiagonalGMM::DiagonalGMM().

39.270.3.9 Probability() [1/2]

```
double Probability (
    const arma::vec & observation ) const
```

Return the probability that the given observation came from this distribution.

Parameters

<i>observation</i>	Observation to evaluate the probability of.
--------------------	---

Referenced by DiagonalGMM::Weights().

39.270.3.10 Probability() [2/2]

```
double Probability (
    const arma::vec & observation,
    const size_t component ) const
```

Return the probability that the given observation came from the given Gaussian component in this distribution.

Parameters

<i>observation</i>	Observation to evaluate the probability of.
<i>component</i>	Index of the component of the DiagonalGMM (p. 1308).

39.270.3.11 Random()

```
arma::vec Random ( ) const
```

Return a randomly generated observation according to the probability distribution defined by this object.

Returns

Random observation from this **DiagonalGMM** (p. 1308).

Referenced by DiagonalGMM::Weights().

39.270.3.12 `serialize()`

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the **DiagonalGMM** (p. 1308).

Referenced by `DiagonalGMM::Weights()`.

39.270.3.13 `Train()` [1/2]

```
double Train (
    const arma::mat & observations,
    const size_t trials = 1,
    const bool useExistingModel = false,
    FittingType fitter = FittingType() )
```

Estimate the probability distribution directly from the given observations, using the given algorithm in the `FittingType` class to fit the data.

The fitting will be performed 'trials' times; from these trials, the model with the greatest log-likelihood will be selected. By default, only one trial is performed. The log-likelihood of the best fitting is returned.

Optionally, the existing model can be used as an initial model for the estimation by setting 'useExistingModel' to true. If the fitting procedure is deterministic after the initial position is given, then 'trials' should be set to 1.

Parameters

<i>observations</i>	Observations of the model.
<i>trials</i>	Number of trials to perform; the model in these trials with the greatest log-likelihood will be selected.
<i>useExistingModel</i>	If true, the existing model is used as an initial model for the estimation.
<i>fitter</i>	Fitting type that estimates observations.

Returns

The log-likelihood of the best fit.

Referenced by `DiagonalGMM::Weights()`.

39.270.3.14 `Train()` [2/2]

```
double Train (
    const arma::mat & observations,
```

```
const arma::vec & probabilities,
const size_t trials = 1,
const bool useExistingModel = false,
FittingType fitter = FittingType() )
```

Estimate the probability distribution directly from the given observations, taking into account the probability of each observation actually being from this distribution, and using the given algorithm in the FittingType class to fit the data.

The fitting will be performed 'trials' times; from these trials, the model with the greatest log-likelihood will be selected. By default, only one trial is performed. The log-likelihood of the best fitting is returned.

Optionally, the existing model can be used as an initial model for the estimation by setting 'useExistingModel' to true. If the fitting procedure is deterministic after the initial position is given, then 'trials' should be set to 1.

Parameters

<i>observations</i>	Observations of the model.
<i>probabilities</i>	Probability of each observation being from this distribution.
<i>trials</i>	Number of trials to perform; the model in these trials with the greatest log-likelihood will be selected.
<i>useExistingModel</i>	If true, the existing model is used as an initial model for the estimation.
<i>fitter</i>	Fitting type that estimates observations.

Returns

The log-likelihood of the best fit.

39.270.3.15 Weights() [1/2]

```
const arma::vec& Weights ( ) const [inline]
```

Return a const reference to the a priori weights of each Gaussian.

Definition at line 157 of file diagonal_gmm.hpp.

39.270.3.16 Weights() [2/2]

```
arma::vec& Weights ( ) [inline]
```

Return a reference to the a priori weights of each Gaussian.

Definition at line 159 of file diagonal_gmm.hpp.

References DiagonalGMM::Classify(), DiagonalGMM::LogProbability(), DiagonalGMM::Probability(), DiagonalGMM::↵ Random(), DiagonalGMM::serialize(), and DiagonalGMM::Train().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/ **diagonal_gmm.hpp**

39.271 EigenvalueRatioConstraint Class Reference

Given a vector of eigenvalue ratios, ensure that the covariance matrix always has those eigenvalue ratios.

Public Member Functions

- **EigenvalueRatioConstraint** (const arma::vec &ratios)
*Create the **EigenvalueRatioConstraint** (p. 1317) object with the given vector of eigenvalue ratios.*
 - void **ApplyConstraint** (arma::mat &covariance) const
Apply the eigenvalue ratio constraint to the given covariance matrix.
 - void **ApplyConstraint** (arma::vec &diagCovariance) const
Apply the eigenvalue ratio constraint to the given diagonal covariance matrix (represented as a vector).
 - template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
- Serialize the constraint.*

39.271.1 Detailed Description

Given a vector of eigenvalue ratios, ensure that the covariance matrix always has those eigenvalue ratios.

When you create this object, make sure that the vector of ratios that you pass does not go out of scope, because this object holds a reference to that vector instead of copying it. (This doesn't apply if you are deserializing the object from a file.)

Definition at line 27 of file eigenvalue_ratio_constraint.hpp.

39.271.2 Constructor & Destructor Documentation

39.271.2.1 EigenvalueRatioConstraint()

```
EigenvalueRatioConstraint (  
    const arma::vec & ratios ) [inline]
```

Create the **EigenvalueRatioConstraint** (p. 1317) object with the given vector of eigenvalue ratios.

These ratios are with respect to the first eigenvalue, which is the largest eigenvalue, so the first element of the vector should be 1. In addition, all other elements should be less than or equal to 1.

Definition at line 36 of file eigenvalue_ratio_constraint.hpp.

References Log::Fatal, and Log::Warn.

39.271.3 Member Function Documentation

39.271.3.1 ApplyConstraint() [1/2]

```
void ApplyConstraint (
    arma::mat & covariance ) const [inline]
```

Apply the eigenvalue ratio constraint to the given covariance matrix.

Definition at line 62 of file eigenvalue_ratio_constraint.hpp.

39.271.3.2 ApplyConstraint() [2/2]

```
void ApplyConstraint (
    arma::vec & diagCovariance ) const [inline]
```

Apply the eigenvalue ratio constraint to the given diagonal covariance matrix (represented as a vector).

Definition at line 83 of file eigenvalue_ratio_constraint.hpp.

39.271.3.3 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the constraint.

Definition at line 102 of file eigenvalue_ratio_constraint.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/ **eigenvalue_ratio_constraint.hpp**

39.272 EMFit< InitialClusteringType, CovarianceConstraintPolicy, Distribution > Class Template Reference

This class contains methods which can fit a **GMM** (p. 1323) to observations using the EM algorithm.

Public Member Functions

- **EMFit** (const size_t maxIterations=300, const double tolerance=1e-10, InitialClusteringType clusterer=InitialClusteringType(), CovarianceConstraintPolicy constraint=CovarianceConstraintPolicy())
*Construct the **EMFit** (p. 1318) object, optionally passing an InitialClusteringType object (just in case it needs to store state).*
- const InitialClusteringType & **Clusterer** () const
Get the clusterer.
- InitialClusteringType & **Clusterer** ()
Modify the clusterer.
- const CovarianceConstraintPolicy & **Constraint** () const
Get the covariance constraint policy class.
- CovarianceConstraintPolicy & **Constraint** ()
Modify the covariance constraint policy class.
- void **Estimate** (const arma::mat &observations, std::vector< Distribution > &dists, arma::vec &weights, const bool useInitialModel=false)
*Fit the observations to a Gaussian mixture model (**GMM** (p. 1323)) using the EM algorithm.*
- void **Estimate** (const arma::mat &observations, const arma::vec &probabilities, std::vector< Distribution > &dists, arma::vec &weights, const bool useInitialModel=false)
*Fit the observations to a Gaussian mixture model (**GMM** (p. 1323)) using the EM algorithm, taking into account the probabilities of each point being from this mixture.*
- size_t **MaxIterations** () const
Get the maximum number of iterations of the EM algorithm.
- size_t & **MaxIterations** ()
Modify the maximum number of iterations of the EM algorithm.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int version)
Serialize the fitter.
- double **Tolerance** () const
Get the tolerance for the convergence of the EM algorithm.
- double & **Tolerance** ()
Modify the tolerance for the convergence of the EM algorithm.

39.272.1 Detailed Description

```
template<typename InitialClusteringType = kmeans::KMeans<>, typename CovarianceConstraintPolicy = PositiveDefiniteConstraint,
typename Distribution = distribution::GaussianDistribution>
class mlpack::gmm::EMFit< InitialClusteringType, CovarianceConstraintPolicy, Distribution >
```

This class contains methods which can fit a **GMM** (p. 1323) to observations using the EM algorithm.

It requires an initial clustering mechanism, which is by default the KMeans algorithm. The clustering mechanism must implement the following method:

- void Cluster(const arma::mat& observations, const size_t clusters, arma::Row<size_t>& assignments);

This method should create 'clusters' clusters, and return the assignment of each point to a cluster.

Definition at line 45 of file em_fit.hpp.

39.272.2 Constructor & Destructor Documentation

39.272.2.1 EMFit()

```
EMFit (
    const size_t maxIterations = 300,
    const double tolerance = 1e-10,
    InitialClusteringType clusterer = InitialClusteringType(),
    CovarianceConstraintPolicy constraint = CovarianceConstraintPolicy() )
```

Construct the **EMFit** (p. 1318) object, optionally passing an InitialClusteringType object (just in case it needs to store state).

Setting the maximum number of iterations to 0 means that the EM algorithm will iterate until convergence (with the given tolerance).

The parameter *forcePositive* controls whether or not the covariance matrices are checked for positive definiteness at each iteration. This could be a time-consuming task, so, if you know your data is well-behaved, you can set it to false and save some runtime.

Parameters

<i>maxIterations</i>	Maximum number of iterations for EM.
<i>tolerance</i>	Log-likelihood tolerance required for convergence.
<i>clusterer</i>	Object which will perform the initial clustering.
<i>constraint</i>	Constraint policy of covariance.

39.272.3 Member Function Documentation

39.272.3.1 Clusterer() [1/2]

```
const InitialClusteringType& Clusterer ( ) const [inline]
```

Get the clusterer.

Definition at line 113 of file `em_fit.hpp`.

39.272.3.2 Clusterer() [2/2]

```
InitialClusteringType& Clusterer ( ) [inline]
```

Modify the clusterer.

Definition at line 115 of file em_fit.hpp.

39.272.3.3 Constraint() [1/2]

```
const CovarianceConstraintPolicy& Constraint ( ) const [inline]
```

Get the covariance constraint policy class.

Definition at line 118 of file em_fit.hpp.

39.272.3.4 Constraint() [2/2]

```
CovarianceConstraintPolicy& Constraint ( ) [inline]
```

Modify the covariance constraint policy class.

Definition at line 120 of file em_fit.hpp.

39.272.3.5 Estimate() [1/2]

```
void Estimate (
    const arma::mat & observations,
    std::vector< Distribution > & dists,
    arma::vec & weights,
    const bool useInitialModel = false )
```

Fit the observations to a Gaussian mixture model (**GMM** (p. 1323)) using the EM algorithm.

The size of the vectors (indicating the number of components) must already be set. Optionally, if useInitialModel is set to true, then the model given in the means, covariances, and weights parameters is used as the initial model, instead of using the InitialClusteringType::Cluster() option.

Parameters

<i>observations</i>	List of observations to train on.
<i>means</i>	Vector to store trained means in.
<i>covariances</i>	Vector to store trained covariances in.
<i>weights</i>	Vector to store a priori weights in.
<i>useInitialModel</i>	If true, the given model is used for the initial clustering.

39.272.3.6 Estimate() [2/2]

```
void Estimate (
    const arma::mat & observations,
    const arma::vec & probabilities,
    std::vector< Distribution > & dists,
    arma::vec & weights,
    const bool useInitialModel = false )
```

Fit the observations to a Gaussian mixture model (**GMM** (p. 1323)) using the EM algorithm, taking into account the probabilities of each point being from this mixture.

The size of the vectors (indicating the number of components) must already be set. Optionally, if `useInitialModel` is set to true, then the model given in the means, covariances, and weights parameters is used as the initial model, instead of using the `InitialClusteringType::Cluster()` option.

Parameters

<i>observations</i>	List of observations to train on.
<i>probabilities</i>	Probability of each point being from this model.
<i>means</i>	Vector to store trained means in.
<i>covariances</i>	Vector to store trained covariances in.
<i>weights</i>	Vector to store a priori weights in.
<i>useInitialModel</i>	If true, the given model is used for the initial clustering.

39.272.3.7 MaxIterations() [1/2]

```
size_t MaxIterations ( ) const [inline]
```

Get the maximum number of iterations of the EM algorithm.

Definition at line 123 of file `em_fit.hpp`.

39.272.3.8 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Modify the maximum number of iterations of the EM algorithm.

Definition at line 125 of file `em_fit.hpp`.

39.272.3.9 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int version )
```

Serialize the fitter.

Referenced by EMFit< InitialClusteringType, CovarianceConstraintPolicy, Distribution >::Tolerance().

39.272.3.10 Tolerance() [1/2]

```
double Tolerance ( ) const [inline]
```

Get the tolerance for the convergence of the EM algorithm.

Definition at line 128 of file em_fit.hpp.

39.272.3.11 Tolerance() [2/2]

```
double& Tolerance ( ) [inline]
```

Modify the tolerance for the convergence of the EM algorithm.

Definition at line 130 of file em_fit.hpp.

References EMFit< InitialClusteringType, CovarianceConstraintPolicy, Distribution >::serialize().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/ **em_fit.hpp**

39.273 GMM Class Reference

A Gaussian Mixture Model (**GMM** (p. 1323)).

Public Member Functions

- **GMM** ()
Create an empty Gaussian Mixture Model, with zero gaussians.
- **GMM** (const size_t gaussians, const size_t dimensionality)
*Create a **GMM** (p. 1323) with the given number of Gaussians, each of which have the specified dimensionality.*
- **GMM** (const std::vector< **distribution::GaussianDistribution** > &dists, const arma::vec &weights)
*Create a **GMM** (p. 1323) with the given dists and weights.*
- **GMM** (const **GMM** &other)
Copy constructor for GMMs.
- void **Classify** (const arma::mat &observations, arma::Row< size_t > &labels) const
*Classify the given observations as being from an individual component in this **GMM** (p. 1323).*
- const **distribution::GaussianDistribution** & **Component** (size_t i) const
Return a const reference to a component distribution.
- **distribution::GaussianDistribution** & **Component** (size_t i)
Return a reference to a component distribution.
- size_t **Dimensionality** () const
Return the dimensionality of the model.
- size_t **Gaussians** () const
Return the number of gaussians in the model.
- double **LogProbability** (const arma::vec &observation) const
Return the log probability that the given observation came from this distribution.
- double **LogProbability** (const arma::vec &observation, const size_t component) const
Return the log probability that the given observation came from the given Gaussian component in this distribution.
- **GMM** & **operator=** (const **GMM** &other)
Copy operator for GMMs.
- double **Probability** (const arma::vec &observation) const
Return the probability that the given observation came from this distribution.
- double **Probability** (const arma::vec &observation, const size_t component) const
Return the probability that the given observation came from the given Gaussian component in this distribution.
- arma::vec **Random** () const
Return a randomly generated observation according to the probability distribution defined by this object.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
*Serialize the **GMM** (p. 1323).*
- template<typename FittingType = EMFit<>>
double **Train** (const arma::mat &observations, const size_t trials=1, const bool useExistingModel=false, FittingType fitter=FittingType())
Estimate the probability distribution directly from the given observations, using the given algorithm in the FittingType class to fit the data.
- template<typename FittingType = EMFit<>>
double **Train** (const arma::mat &observations, const arma::vec &probabilities, const size_t trials=1, const bool useExistingModel=false, FittingType fitter=FittingType())
Estimate the probability distribution directly from the given observations, taking into account the probability of each observation actually being from this distribution, and using the given algorithm in the FittingType class to fit the data.
- const arma::vec & **Weights** () const
Return a const reference to the a priori weights of each Gaussian.
- arma::vec & **Weights** ()
Return a reference to the a priori weights of each Gaussian.

39.273.1 Detailed Description

A Gaussian Mixture Model (**GMM** (p. 1323)).

This class uses maximum likelihood loss functions to estimate the parameters of the **GMM** (p. 1323) on a given dataset via the given fitting mechanism, defined by the `FittingType` template parameter. The **GMM** (p. 1323) can be trained using normal data, or data with probabilities of being from this **GMM** (p. 1323) (see **GMM::Train()** (p. 1330) for more information).

The **Train()** (p. 1330) method uses a template type 'FittingType'. The `FittingType` template class must provide a way for the **GMM** (p. 1323) to train on data. It must provide the following two functions:

```
void Estimate(const arma::mat& observations,
              std::vector<distribution::GaussianDistribution>& dists,
              arma::vec& weights);

void Estimate(const arma::mat& observations,
              const arma::vec& probabilities,
              std::vector<distribution::GaussianDistribution>& dists,
              arma::vec& weights);
```

These functions should produce a trained **GMM** (p. 1323) from the given observations and probabilities. These may modify the size of the model (by increasing the size of the mean and covariance vectors as well as the weight vectors), but the method should expect that these vectors are already set to the size of the **GMM** (p. 1323) as specified in the constructor.

For a sample implementation, see the **EMFit** (p. 1318) class; this class uses the EM algorithm to train a **GMM** (p. 1323), and is the default fitting type for the **Train()** (p. 1330) method.

The **GMM** (p. 1323), once trained, can be used to generate random points from the distribution and estimate the probability of points being from the distribution. The parameters of the **GMM** (p. 1323) can be obtained through the accessors and mutators.

Example use:

```
// Set up a mixture of 5 gaussians in a 4-dimensional space.
GMM g(5, 4);

// Train the GMM given the data observations, using the default EM fitting
// mechanism.
g.Train(data);

// Get the probability of 'observation' being observed from this GMM.
double probability = g.Probability(observation);

// Get a random observation from the GMM.
arma::vec observation = g.Random();
```

Definition at line 78 of file `gmm.hpp`.

39.273.2 Constructor & Destructor Documentation

39.273.2.1 GMM() [1/4]

```
GMM ( ) [inline]
```

Create an empty Gaussian Mixture Model, with zero gaussians.

Definition at line 96 of file gmm.hpp.

References Log::Debug.

Referenced by GMM::GMM().

39.273.2.2 GMM() [2/4]

```
GMM (
    const size_t gaussians,
    const size_t dimensionality )
```

Create a **GMM** (p. 1323) with the given number of Gaussians, each of which have the specified dimensionality.

The means and covariances will be set to 0.

Parameters

<i>gaussians</i>	Number of Gaussians in this GMM (p. 1323).
<i>dimensionality</i>	Dimensionality of each Gaussian.

39.273.2.3 GMM() [3/4]

```
GMM (
    const std::vector< distribution::GaussianDistribution > & dists,
    const arma::vec & weights ) [inline]
```

Create a **GMM** (p. 1323) with the given dists and weights.

Parameters

<i>dists</i>	Distributions of the model.
<i>weights</i>	Weights of the model.

Definition at line 122 of file gmm.hpp.

References GMM::GMM(), and GMM::operator=().

39.273.2.4 GMM() [4/4]

```
GMM (
    const GMM & other )
```

Copy constructor for GMMs.

39.273.3 Member Function Documentation

39.273.3.1 Classify()

```
void Classify (
    const arma::mat & observations,
    arma::Row< size_t > & labels ) const
```

Classify the given observations as being from an individual component in this **GMM** (p. 1323).

The resultant classifications are stored in the 'labels' object, and each label will be between 0 and (**Gaussians()** (p. 1328) - 1). Supposing that a point was classified with label 2, and that our **GMM** (p. 1323) object was called 'gmm', one could access the relevant Gaussian distribution as follows:

```
arma::vec mean = gmm.Means()[2];
arma::mat covariance = gmm.Covariances()[2];
double priorWeight = gmm.Weights()[2];
```

Parameters

<i>observations</i>	List of observations to classify.
<i>labels</i>	Object which will be filled with labels.

Referenced by **GMM::Weights()**.

39.273.3.2 Component() [1/2]

```
const distribution::GaussianDistribution& Component (
    size_t i ) const [inline]
```

Return a const reference to a component distribution.

Parameters

<i>i</i>	Index of component.
----------	---------------------

Definition at line 145 of file gmm.hpp.

39.273.3.3 Component() [2/2]

```
distribution::GaussianDistribution& Component (
    size_t i ) [inline]
```

Return a reference to a component distribution.

Parameters

<i>i</i>	Index of component.
----------	---------------------

Definition at line 152 of file gmm.hpp.

39.273.3.4 Dimensionality()

```
size_t Dimensionality ( ) const [inline]
```

Return the dimensionality of the model.

Definition at line 138 of file gmm.hpp.

39.273.3.5 Gaussians()

```
size_t Gaussians ( ) const [inline]
```

Return the number of gaussians in the model.

Definition at line 136 of file gmm.hpp.

39.273.3.6 LogProbability() [1/2]

```
double LogProbability (
    const arma::vec & observation ) const
```

Return the log probability that the given observation came from this distribution.

Parameters

<i>observation</i>	Observation to evaluate the probability of.
--------------------	---

Referenced by GMM::Weights().

39.273.3.7 LogProbability() [2/2]

```
double LogProbability (
    const arma::vec & observation,
    const size_t component ) const
```

Return the log probability that the given observation came from the given Gaussian component in this distribution.

Parameters

<i>observation</i>	Observation to evaluate the probability of.
<i>component</i>	Index of the component of the GMM (p. 1323) to be considered.

39.273.3.8 operator=()

```
GMM& operator= (
    const GMM & other )
```

Copy operator for GMMs.

Referenced by GMM::GMM().

39.273.3.9 Probability() [1/2]

```
double Probability (
    const arma::vec & observation ) const
```

Return the probability that the given observation came from this distribution.

Parameters

<i>observation</i>	Observation to evaluate the probability of.
--------------------	---

Referenced by GMM::Weights().

39.273.3.10 Probability() [2/2]

```
double Probability (
    const arma::vec & observation,
    const size_t component ) const
```

Return the probability that the given observation came from the given Gaussian component in this distribution.

Parameters

<i>observation</i>	Observation to evaluate the probability of.
<i>component</i>	Index of the component of the GMM (p. 1323) to be considered.

39.273.3.11 Random()

```
arma::vec Random ( ) const
```

Return a randomly generated observation according to the probability distribution defined by this object.

Returns

Random observation from this **GMM** (p. 1323).

Referenced by GMM::Weights().

39.273.3.12 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the **GMM** (p. 1323).

Referenced by GMM::Weights().

39.273.3.13 Train() [1/2]

```
double Train (
    const arma::mat & observations,
    const size_t trials = 1,
    const bool useExistingModel = false,
    FittingType fitter = FittingType() )
```

Estimate the probability distribution directly from the given observations, using the given algorithm in the FittingType class to fit the data.

The fitting will be performed 'trials' times; from these trials, the model with the greatest log-likelihood will be selected. By default, only one trial is performed. The log-likelihood of the best fitting is returned.

Optionally, the existing model can be used as an initial model for the estimation by setting 'useExistingModel' to true. If the fitting procedure is deterministic after the initial position is given, then 'trials' should be set to 1.

Template Parameters

<i>FittingType</i>	The type of fitting method which should be used (EMFit<> is suggested).
--------------------	---

Parameters

<i>observations</i>	Observations of the model.
<i>trials</i>	Number of trials to perform; the model in these trials with the greatest log-likelihood will be selected.
<i>useExistingModel</i>	If true, the existing model is used as an initial model for the estimation.

Returns

The log-likelihood of the best fit.

Referenced by GMM::Weights().

39.273.3.14 Train() [2/2]

```
double Train (
    const arma::mat & observations,
    const arma::vec & probabilities,
    const size_t trials = 1,
    const bool useExistingModel = false,
    FittingType fitter = FittingType() )
```

Estimate the probability distribution directly from the given observations, taking into account the probability of each observation actually being from this distribution, and using the given algorithm in the FittingType class to fit the data.

The fitting will be performed 'trials' times; from these trials, the model with the greatest log-likelihood will be selected. By default, only one trial is performed. The log-likelihood of the best fitting is returned.

Optionally, the existing model can be used as an initial model for the estimation by setting 'useExistingModel' to true. If the fitting procedure is deterministic after the initial position is given, then 'trials' should be set to 1.

Parameters

<i>observations</i>	Observations of the model.
<i>probabilities</i>	Probability of each observation being from this distribution.
<i>trials</i>	Number of trials to perform; the model in these trials with the greatest log-likelihood will be selected.
<i>useExistingModel</i>	If true, the existing model is used as an initial model for the estimation.

Returns

The log-likelihood of the best fit.

39.273.3.15 **Weights()** [1/2]

```
const arma::vec& Weights ( ) const [inline]
```

Return a const reference to the a priori weights of each Gaussian.

Definition at line 155 of file gmm.hpp.

39.273.3.16 **Weights()** [2/2]

```
arma::vec& Weights ( ) [inline]
```

Return a reference to the a priori weights of each Gaussian.

Definition at line 157 of file gmm.hpp.

References GMM::Classify(), GMM::LogProbability(), GMM::Probability(), GMM::Random(), GMM::serialize(), and GMM::Train().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/ **gmm.hpp**

39.274 **NoConstraint Class Reference**

This class enforces no constraint on the covariance matrix.

Static Public Member Functions

- static void **ApplyConstraint** (const arma::mat &)
Do nothing, and do not modify the covariance matrix.
- template<typename Archive >
static void **serialize** (Archive &, const unsigned int)
Serialize the object (nothing to do).

39.274.1 Detailed Description

This class enforces no constraint on the covariance matrix.

It's faster this way, although depending on your situation you may end up with a non-invertible covariance matrix.

Definition at line 25 of file no_constraint.hpp.

39.274.2 Member Function Documentation

39.274.2.1 ApplyConstraint()

```
static void ApplyConstraint (  
    const arma::mat & ) [inline], [static]
```

Do nothing, and do not modify the covariance matrix.

Definition at line 29 of file no_constraint.hpp.

39.274.2.2 serialize()

```
static void serialize (  
    Archive & ,  
    const unsigned int ) [inline], [static]
```

Serialize the object (nothing to do).

Definition at line 33 of file no_constraint.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/ **no_constraint.hpp**

39.275 PositiveDefiniteConstraint Class Reference

Given a covariance matrix, force the matrix to be positive definite.

Static Public Member Functions

- static void **ApplyConstraint** (arma::mat &covariance)
Apply the positive definiteness constraint to the given covariance matrix, and ensure each value on the diagonal is at least 1e-50.
- static void **ApplyConstraint** (arma::vec &diagCovariance)
Apply the positive definiteness constraint to the given diagonal covariance matrix (which is represented as a vector), and ensure each value on the diagonal is at least 1e-50.
- template<typename Archive >
static void **serialize** (Archive &, const unsigned int)
Serialize the constraint (which stores nothing, so, nothing to do).

39.275.1 Detailed Description

Given a covariance matrix, force the matrix to be positive definite.

Also force a minimum value on the diagonal, so that even if the matrix is invertible, it doesn't cause problems with Cholesky decompositions. The forcing here is also done in order to bring the condition number of the matrix under 1e5 (10k), which should help with numerical stability.

Definition at line 27 of file positive_definite_constraint.hpp.

39.275.2 Member Function Documentation

39.275.2.1 ApplyConstraint() [1/2]

```
static void ApplyConstraint (
    arma::mat & covariance ) [inline], [static]
```

Apply the positive definiteness constraint to the given covariance matrix, and ensure each value on the diagonal is at least 1e-50.

Parameters

<i>covariance</i>	Covariance matrix.
-------------------	--------------------

Definition at line 36 of file positive_definite_constraint.hpp.

39.275.2.2 ApplyConstraint() [2/2]

```
static void ApplyConstraint (
    arma::vec & diagCovariance ) [inline], [static]
```

Apply the positive definiteness constraint to the given diagonal covariance matrix (which is represented as a vector), and ensure each value on the diagonal is at least 1e-50.

Definition at line 70 of file positive_definite_constraint.hpp.

39.275.2.3 serialize()

```
static void serialize (
    Archive & ,
    const unsigned int ) [inline], [static]
```

Serialize the constraint (which stores nothing, so, nothing to do).

Definition at line 95 of file positive_definite_constraint.hpp.

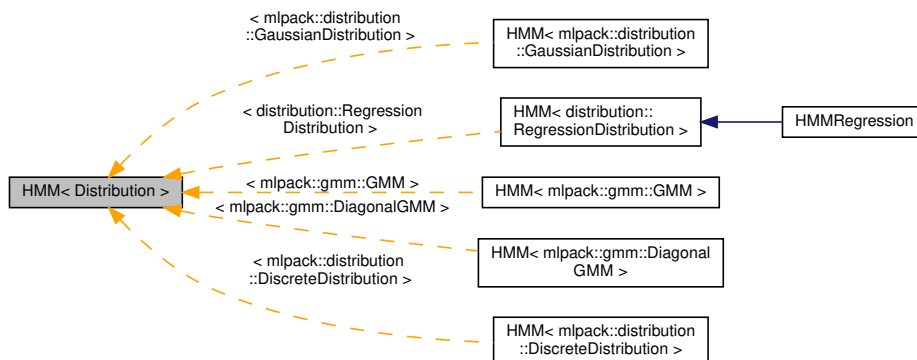
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/ **positive_definite_constraint.hpp**

39.276 HMM< Distribution > Class Template Reference

A class that represents a Hidden Markov Model with an arbitrary type of emission distribution.

Inheritance diagram for HMM< Distribution >:



Public Member Functions

- **HMM** (const size_t states=0, const Distribution emissions=Distribution(), const double tolerance=1e-5)
Create the Hidden Markov Model with the given number of hidden states and the given default distribution for emissions.
- **HMM** (const arma::vec &initial, const arma::mat & **transition**, const std::vector< Distribution > & **emission**, const double tolerance=1e-5)
Create the Hidden Markov Model with the given initial probability vector, the given transition matrix, and the given emission distributions.
- size_t **Dimensionality** () const
Get the dimensionality of observations.
- size_t & **Dimensionality** ()
Set the dimensionality of observations.
- const std::vector< Distribution > & **Emission** () const
Return the emission distributions.
- std::vector< Distribution > & **Emission** ()
Return a modifiable emission probability matrix reference.
- double **Estimate** (const arma::mat &dataSeq, arma::mat &stateProb, arma::mat &forwardProb, arma::mat &backwardProb, arma::vec &scales) const
Estimate the probabilities of each hidden state at each time step for each given data observation, using the Forward-↔ Backward algorithm.
- double **Estimate** (const arma::mat &dataSeq, arma::mat &stateProb) const
Estimate the probabilities of each hidden state at each time step of each given data observation, using the Forward-↔ Backward algorithm.
- void **Filter** (const arma::mat &dataSeq, arma::mat &filterSeq, size_t ahead=0) const
HMM (p. 1335) filtering.
- void **Generate** (const size_t length, arma::mat &dataSequence, arma::Row< size_t > &stateSequence, const size_t startState=0) const
Generate a random data sequence of the given length.
- const arma::vec & **Initial** () const
Return the vector of initial state probabilities.
- arma::vec & **Initial** ()
Modify the vector of initial state probabilities.
- double **LogEstimate** (const arma::mat &dataSeq, arma::mat &stateLogProb, arma::mat &forwardLogProb, arma::mat &backwardLogProb, arma::vec &logScales) const
Estimate the probabilities of each hidden state at each time step for each given data observation, using the Forward-↔ Backward algorithm.
- double **LogLikelihood** (const arma::mat &dataSeq) const
Compute the log-likelihood of the given data sequence.
- double **Predict** (const arma::mat &dataSeq, arma::Row< size_t > &stateSeq) const
Compute the most probable hidden state sequence for the given data sequence, using the Viterbi algorithm, returning the log-likelihood of the most likely state sequence.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int version)
Serialize the object.
- void **Smooth** (const arma::mat &dataSeq, arma::mat &smoothSeq) const
HMM (p. 1335) smoothing.
- double **Tolerance** () const
Get the tolerance of the Baum-Welch algorithm.
- double & **Tolerance** ()

Modify the tolerance of the Baum-Welch algorithm.

- double **Train** (const std::vector< arma::mat > &dataSeq)
Train the model using the Baum-Welch algorithm, with only the given unlabeled observations.
- void **Train** (const std::vector< arma::mat > &dataSeq, const std::vector< arma::Row< size_t > > &stateSeq)
Train the model using the given labeled observations; the transition and emission matrices are directly estimated.
- const arma::mat & **Transition** () const
Return the transition matrix.
- arma::mat & **Transition** ()
Return a modifiable transition matrix reference.

Protected Member Functions

- void **Backward** (const arma::mat &dataSeq, const arma::vec &logScales, arma::mat &backwardLogProb) const
The Backward algorithm (part of the Forward-Backward algorithm).
- void **Forward** (const arma::mat &dataSeq, arma::vec &logScales, arma::mat &forwardLogProb) const
The Forward algorithm (part of the Forward-Backward algorithm).

Protected Attributes

- std::vector< Distribution > **emission**
Set of emission probability distributions; one for each state.
- arma::mat **transition**
Transition probability matrix.

39.276.1 Detailed Description

```
template<typename Distribution = distribution::DiscreteDistribution>
class mpack::hmm::HMM< Distribution >
```

A class that represents a Hidden Markov Model with an arbitrary type of emission distribution.

This **HMM** (p. 1335) class supports training (supervised and unsupervised), prediction of state sequences via the Viterbi algorithm, estimation of state probabilities, generation of random sequences, and calculation of the log-likelihood of a given sequence.

The template parameter, *Distribution*, specifies the distribution which the emissions follow. The class should implement the following functions:

```
class Distribution
{
public:
    // The type of observation used by this distribution.
    typedef something DataType;

    // Return the probability of the given observation.
    double Probability(const DataType& observation) const;

    // Estimate the distribution based on the given observations.
    double Train(const std::vector<DataType>& observations);

    // Estimate the distribution based on the given observations, given also
    // the probability of each observation coming from this distribution.
    double Train(const std::vector<DataType>& observations,
                const std::vector<double>& probabilities);
};
```

See the `mlpack::distribution::DiscreteDistribution` (p. 1232) class for an example. One would use the `DiscreteDistribution` class when the observations are non-negative integers. Other distributions could be Gaussians, a mixture of Gaussians (GMM), or any other probability distribution implementing the four Distribution functions.

Usage of the **HMM** (p. 1335) class generally involves either training an **HMM** (p. 1335) or loading an already-known **HMM** (p. 1335) and taking probability measurements of sequences. Example code for supervised training of a Gaussian **HMM** (p. 1335) (that is, where the emission output distribution is a single Gaussian for each hidden state) is given below.

```
extern arma::mat observations; // Each column is an observation.
extern arma::Row<size_t> states; // Hidden states for each observation.
// Create an untrained HMM with 5 hidden states and default (N(0, 1))
// Gaussian distributions with the dimensionality of the dataset.
HMM<GaussianDistribution> hmm(5, GaussianDistribution(observations.n_rows));

// Train the HMM (the labels could be omitted to perform unsupervised
// training).
hmm.Train(observations, states);
```

Once initialized, the **HMM** (p. 1335) can evaluate the probability of a certain sequence (with **LogLikelihood()** (p. 1346)), predict the most likely sequence of hidden states (with **Predict()** (p. 1346)), generate a sequence (with **Generate()** (p. 1343)), or estimate the probabilities of each state for a sequence of observations (with **Train()** (p. 1348)).

Template Parameters

<i>Distribution</i>	Type of emission distribution for this HMM (p. 1335).
---------------------	--

Definition at line 85 of file `hmm.hpp`.

39.276.2 Constructor & Destructor Documentation

39.276.2.1 HMM() [1/2]

```
HMM (
    const size_t states = 0,
    const Distribution emissions = Distribution(),
    const double tolerance = 1e-5 )
```

Create the Hidden Markov Model with the given number of hidden states and the given default distribution for emissions.

The dimensionality of the observations is taken from the emissions variable, so it is important that the given default emission distribution is set with the correct dimensionality. Alternately, set the dimensionality with **Dimensionality()** (p. 1341). Optionally, the tolerance for convergence of the Baum-Welch algorithm can be set.

By default, the transition matrix and initial probability vector are set to contain equal probability for each state.

Parameters

<i>states</i>	Number of states.
<i>emissions</i>	Default distribution for emissions.
<i>tolerance</i>	Tolerance for convergence of training algorithm (Baum-Welch).

39.276.2.2 HMM() [2/2]

```

HMM (
    const arma::vec & initial,
    const arma::mat & transition,
    const std::vector< Distribution > & emission,
    const double tolerance = 1e-5 )

```

Create the Hidden Markov Model with the given initial probability vector, the given transition matrix, and the given emission distributions.

The dimensionality of the observations of the **HMM** (p. 1335) are taken from the given emission distributions. Alternately, the dimensionality can be set with **Dimensionality()** (p. 1341).

The initial state probability vector should have length equal to the number of states, and each entry represents the probability of being in the given state at time $T = 0$ (the beginning of a sequence).

The transition matrix should be such that $T(i, j)$ is the probability of transition to state i from state j . The columns of the matrix should sum to 1.

The emission matrix should be such that $E(i, j)$ is the probability of emission i while in state j . The columns of the matrix should sum to 1.

Optionally, the tolerance for convergence of the Baum-Welch algorithm can be set.

Parameters

<i>initial</i>	Initial state probabilities.
<i>transition</i>	Transition matrix.
<i>emission</i>	Emission distributions.
<i>tolerance</i>	Tolerance for convergence of training algorithm (Baum-Welch).

39.276.3 Member Function Documentation

39.276.3.1 Backward()

```

void Backward (
    const arma::mat & dataSeq,
    const arma::vec & logScales,
    arma::mat & backwardLogProb ) const [protected]

```

The Backward algorithm (part of the Forward-Backward algorithm).

Computes backward probabilities for each state for each observation in the given data sequence, using the scaling factors found (presumably) by **Forward()** (p. 1343). The returned matrix has rows equal to the number of hidden states and columns equal to the number of observations.

Parameters

<i>dataSeq</i>	Data sequence to compute probabilities for.
<i>scales</i>	Vector of scaling factors.
<i>backwardProb</i>	Matrix in which backward probabilities will be saved.

Referenced by HMM< mlpack::distribution::DiscreteDistribution >::Tolerance().

39.276.3.2 Dimensionality() [1/2]

```
size_t Dimensionality ( ) const [inline]
```

Get the dimensionality of observations.

Definition at line 341 of file hmm.hpp.

39.276.3.3 Dimensionality() [2/2]

```
size_t& Dimensionality ( ) [inline]
```

Set the dimensionality of observations.

Definition at line 343 of file hmm.hpp.

39.276.3.4 Emission() [1/2]

```
const std::vector<Distribution>& Emission ( ) const [inline]
```

Return the emission distributions.

Definition at line 336 of file hmm.hpp.

39.276.3.5 Emission() [2/2]

```
std::vector<Distribution>& Emission ( ) [inline]
```

Return a modifiable emission probability matrix reference.

Definition at line 338 of file hmm.hpp.

39.276.3.6 Estimate() [1/2]

```
double Estimate (
    const arma::mat & dataSeq,
    arma::mat & stateProb,
    arma::mat & forwardProb,
    arma::mat & backwardProb,
    arma::vec & scales ) const
```

Estimate the probabilities of each hidden state at each time step for each given data observation, using the Forward-↔ Backward algorithm.

Each matrix which is returned has columns equal to the number of data observations, and rows equal to the number of hidden states in the model. The log-likelihood of the most probable sequence is returned.

Parameters

<i>dataSeq</i>	Sequence of observations.
<i>stateProb</i>	Matrix in which the probabilities of each state at each time interval will be stored.
<i>forwardProb</i>	Matrix in which the forward probabilities of each state at each time interval will be stored.
<i>backwardProb</i>	Matrix in which the backward probabilities of each state at each time interval will be stored.
<i>scales</i>	Vector in which the scaling factors at each time interval will be stored.

Returns

Log-likelihood of most likely state sequence.

39.276.3.7 Estimate() [2/2]

```
double Estimate (
    const arma::mat & dataSeq,
    arma::mat & stateProb ) const
```

Estimate the probabilities of each hidden state at each time step of each given data observation, using the Forward-↔ Backward algorithm.

The returned matrix of state probabilities has columns equal to the number of data observations, and rows equal to the number of hidden states in the model. The log-likelihood of the most probable sequence is returned.

Parameters

<i>dataSeq</i>	Sequence of observations.
<i>stateProb</i>	Probabilities of each state at each time interval.

Returns

Log-likelihood of most likely state sequence.

39.276.3.8 Filter()

```
void Filter (
    const arma::mat & dataSeq,
    arma::mat & filterSeq,
    size_t ahead = 0 ) const
```

HMM (p. 1335) filtering.

Computes the k-step-ahead expected emission at each time conditioned only on prior observations. That is $E\{Y[t+k] \mid Y[0], \dots, Y[t]\}$. The returned matrix has columns equal to the number of observations. Note that the expectation may not be meaningful for discrete emissions.

Parameters

<i>dataSeq</i>	Sequence of observations.
<i>filterSeq</i>	Vector in which the expected emission sequence will be stored.
<i>ahead</i>	Number of steps ahead (k) for expectations.

39.276.3.9 Forward()

```
void Forward (
    const arma::mat & dataSeq,
    arma::vec & logScales,
    arma::mat & forwardLogProb ) const [protected]
```

The Forward algorithm (part of the Forward-Backward algorithm).

Computes forward probabilities for each state for each observation in the given data sequence. The returned matrix has rows equal to the number of hidden states and columns equal to the number of observations.

Parameters

<i>dataSeq</i>	Data sequence to compute probabilities for.
<i>scales</i>	Vector in which scaling factors will be saved.
<i>forwardProb</i>	Matrix in which forward probabilities will be saved.

Referenced by HMM< mlpack::distribution::DiscreteDistribution >::Tolerance().

39.276.3.10 Generate()

```
void Generate (
    const size_t length,
    arma::mat & dataSequence,
    arma::Row< size_t > & stateSequence,
    const size_t startState = 0 ) const
```

Generate a random data sequence of the given length.

The data sequence is stored in the `dataSequence` parameter, and the state sequence is stored in the `stateSequence` parameter. Each column of `dataSequence` represents a random observation.

Parameters

<i>length</i>	Length of random sequence to generate.
<i>dataSequence</i>	Vector to store data in.
<i>stateSequence</i>	Vector to store states in.
<i>startState</i>	Hidden state to start sequence in (default 0).

39.276.3.11 Initial() [1/2]

```
const arma::vec& Initial ( ) const [inline]
```

Return the vector of initial state probabilities.

Definition at line 326 of file `hmm.hpp`.

39.276.3.12 Initial() [2/2]

```
arma::vec& Initial ( ) [inline]
```

Modify the vector of initial state probabilities.

Definition at line 328 of file `hmm.hpp`.

39.276.3.13 LogEstimate()

```
double LogEstimate (
    const arma::mat & dataSeq,
    arma::mat & stateLogProb,
    arma::mat & forwardLogProb,
    arma::mat & backwardLogProb,
    arma::vec & logScales ) const
```

Estimate the probabilities of each hidden state at each time step for each given data observation, using the Forward-↔ Backward algorithm.

Each matrix which is returned has columns equal to the number of data observations, and rows equal to the number of hidden states in the model. The log-likelihood of the most probable sequence is returned.

Parameters

<i>dataSeq</i>	Sequence of observations.
<i>stateProb</i>	Matrix in which the log probabilities of each state at each time interval will be stored.
<i>forwardProb</i>	Matrix in which the forward log probabilities of each state at each time interval will be stored.
<i>backwardProb</i>	Matrix in which the backward log probabilities of each state at each time interval will be stored.
<i>scales</i>	Vector in which the log of scaling factors at each time interval will be stored.

Returns

Log-likelihood of most likely state sequence.

39.276.3.14 LogLikelihood()

```
double LogLikelihood (
    const arma::mat & dataSeq ) const
```

Compute the log-likelihood of the given data sequence.

Parameters

<i>dataSeq</i>	Data sequence to evaluate the likelihood of.
----------------	--

Returns

Log-likelihood of the given sequence.

39.276.3.15 Predict()

```
double Predict (
    const arma::mat & dataSeq,
    arma::Row< size_t > & stateSeq ) const
```

Compute the most probable hidden state sequence for the given data sequence, using the Viterbi algorithm, returning the log-likelihood of the most likely state sequence.

Parameters

<i>dataSeq</i>	Sequence of observations.
<i>stateSeq</i>	Vector in which the most probable state sequence will be stored.

Returns

Log-likelihood of most probable state sequence.

39.276.3.16 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int version )
```

Serialize the object.

Referenced by HMM< mlpack::distribution::DiscreteDistribution >::Tolerance().

39.276.3.17 Smooth()

```
void Smooth (
    const arma::mat & dataSeq,
    arma::mat & smoothSeq ) const
```

HMM (p. 1335) smoothing.

Computes expected emission at each time conditioned on all observations. That is $E\{Y[t] \mid Y[0], \dots, Y[T]\}$. The returned matrix has columns equal to the number of observations. Note that the expectation may not be meaningful for discrete emissions.

Parameters

<i>dataSeq</i>	Sequence of observations.
<i>smoothSeq</i>	Vector in which the expected emission sequence will be stored.

39.276.3.18 Tolerance() [1/2]

```
double Tolerance ( ) const [inline]
```

Get the tolerance of the Baum-Welch algorithm.

Definition at line 346 of file hmm.hpp.

39.276.3.19 Tolerance() [2/2]

```
double& Tolerance ( ) [inline]
```

Modify the tolerance of the Baum-Welch algorithm.

Definition at line 348 of file `hmm.hpp`.

39.276.3.20 Train() [1/2]

```
double Train (
    const std::vector< arma::mat > & dataSeq )
```

Train the model using the Baum-Welch algorithm, with only the given unlabeled observations.

Instead of giving a guess transition and emission matrix here, do that in the constructor. Each matrix in the vector of data sequences holds an individual data sequence; each point in each individual data sequence should be a column in the matrix. The number of rows in each matrix should be equal to the dimensionality of the **HMM** (p. 1335) (which is set in the constructor).

It is preferable to use the other overload of **Train()** (p. 1348), with labeled data. That will produce much better results. However, if labeled data is unavailable, this will work. In addition, it is possible to use **Train()** (p. 1348) with labeled data first, and then continue to train the model using this overload of **Train()** (p. 1348) with unlabeled data.

The tolerance of the Baum-Welch algorithm can be set either in the constructor or with the **Tolerance()** (p. 1347) method. When the change in log-likelihood of the model between iterations is less than the tolerance, the Baum-Welch algorithm terminates.

Note

Train() (p. 1348) can be called multiple times with different sequences; each time it is called, it uses the current parameters of the **HMM** (p. 1335) as a starting point for training.

Parameters

<i>dataSeq</i>	Vector of observation sequences.
----------------	----------------------------------

Returns

Log-likelihood of state sequence.

39.276.3.21 Train() [2/2]

```
void Train (
    const std::vector< arma::mat > & dataSeq,
    const std::vector< arma::Row< size_t > > & stateSeq )
```

Train the model using the given labeled observations; the transition and emission matrices are directly estimated.

Each matrix in the vector of data sequences corresponds to a vector in the vector of state sequences. Each point in each individual data sequence should be a column in the matrix, and its state should be the corresponding element in the state sequence vector. For instance, `dataSeq[0].col(3)` corresponds to the fourth observation in the first data sequence, and its state is `stateSeq[0][3]`. The number of rows in each matrix should be equal to the dimensionality of the **HMM** (p. 1335) (which is set in the constructor).

Note

Train() (p. 1348) can be called multiple times with different sequences; each time it is called, it uses the current parameters of the **HMM** (p. 1335) as a starting point for training.

Parameters

<i>dataSeq</i>	Vector of observation sequences.
<i>stateSeq</i>	Vector of state sequences, corresponding to each observation.

39.276.3.22 Transition() [1/2]

```
const arma::mat& Transition ( ) const [inline]
```

Return the transition matrix.

Definition at line 331 of file `hmm.hpp`.

39.276.3.23 Transition() [2/2]

```
arma::mat& Transition ( ) [inline]
```

Return a modifiable transition matrix reference.

Definition at line 333 of file `hmm.hpp`.

39.276.4 Member Data Documentation

39.276.4.1 emission

```
std::vector<Distribution> emission [protected]
```

Set of emission probability distributions; one for each state.

Definition at line 388 of file hmm.hpp.

Referenced by HMM< mlpack::distribution::DiscreteDistribution >::Emission().

39.276.4.2 transition

```
arma::mat transition [protected]
```

Transition probability matrix.

Definition at line 391 of file hmm.hpp.

Referenced by HMM< mlpack::distribution::DiscreteDistribution >::Transition().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hmm/ **hmm.hpp**

39.277 HMMModel Class Reference

A serializable **HMM** (p. 1335) model that also stores the type.

Public Member Functions

- **HMMModel** (const **HMMType** type=HMMType::DiscreteHMM)
Construct a model of the given type.
- **HMMModel** (const **HMMModel** &other)
Copy another model.
- **HMMModel** (**HMMModel** &&other)
Take ownership of another model.
- **~HMMModel** ()
Clean memory.
- **HMM**< **gmm::DiagonalGMM** > * **DiagGMMHMM** ()
- **HMM**< **distribution::DiscreteDistribution** > * **DiscreteHMM** ()
Accessor methods for discreteHMM, gaussianHMM, gmmHMM, and diagGMMHMM.
- **HMM**< **distribution::GaussianDistribution** > * **GaussianHMM** ()
- **HMM**< **gmm::GMM** > * **GMMHMM** ()
- **HMMModel** & **operator=** (const **HMMModel** &other)
Copy assignment operator.
- template<typename ActionType , typename ExtraInfoType >
void **PerformAction** (ExtraInfoType *x)
*Given a functor type, perform that functor with the optional extra info on the **HMM** (p. 1335).*
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int version)
Serialize the model.
- **HMMType** **Type** ()

39.277.1 Detailed Description

A serializable **HMM** (p. 1335) model that also stores the type.

Definition at line 33 of file `hmm_model.hpp`.

39.277.2 Constructor & Destructor Documentation

39.277.2.1 HMMModel() [1/3]

```
HMMModel (
    const HMMType type = HMMType::DiscreteHMM ) [inline]
```

Construct a model of the given type.

Definition at line 49 of file `hmm_model.hpp`.

References `mlpack::hmm::DiagonalGaussianMixtureModelHMM`, `mlpack::hmm::DiscreteHMM`, `mlpack::hmm::GaussianHMM`, and `mlpack::hmm::GaussianMixtureModelHMM`.

39.277.2.2 HMMModel() [2/3]

```
HMMModel (
    const HMMModel & other ) [inline]
```

Copy another model.

Definition at line 67 of file `hmm_model.hpp`.

References `mlpack::hmm::DiagonalGaussianMixtureModelHMM`, `mlpack::hmm::DiscreteHMM`, `mlpack::hmm::GaussianHMM`, and `mlpack::hmm::GaussianMixtureModelHMM`.

39.277.2.3 HMMModel() [3/3]

```
HMMModel (
    HMMModel && other ) [inline]
```

Take ownership of another model.

Definition at line 87 of file `hmm_model.hpp`.

References `mlpack::hmm::DiscreteHMM`.

39.277.2.4 ~HMMModel()

```
~ HMMModel ( ) [inline]
```

Clean memory.

Definition at line 133 of file hmm_model.hpp.

39.277.3 Member Function Documentation

39.277.3.1 DiagGMMHMM()

```
HMM< gmm::DiagonalGMM>* DiagGMMHMM ( ) [inline]
```

Definition at line 219 of file hmm_model.hpp.

References `BOOST_CLASS_VERSION()`.

39.277.3.2 DiscreteHMM()

```
HMM< distribution::DiscreteDistribution>* DiscreteHMM ( ) [inline]
```

Accessor methods for discreteHMM, gaussianHMM, gmmHMM, and diagGMMHMM.

Note that an instantiation of this class will only contain one type of **HMM** (p. 1335) (as indicated by the "type" instance variable) - the other two pointers will be NULL.

For instance, if the **HMMModel** (p. 1350) object holds a discrete **HMM** (p. 1335), then: type → DiscreteHMM gaussian↔ HMM → NULL gmmHMM → NULL diagGMMHMM → NULL discreteHMM → **HMM**<DiscreteDistribution> object and hence, calls to **GMMHMM()** (p. 1352), **DiagGMMHMM()** (p. 1352) and **GaussianHMM()** (p. 1352) will return NULL. Only the call to **DiscreteHMM()** (p. 1352) will return a non NULL pointer.

Hence, in practice, a user should be careful to first check the type of **HMM** (p. 1335) (by calling the **Type()** (p. 1353) accessor) and then perform subsequent actions, to avoid null pointer dereferences.

Definition at line 216 of file hmm_model.hpp.

39.277.3.3 GaussianHMM()

```
HMM< distribution::GaussianDistribution>* GaussianHMM ( ) [inline]
```

Definition at line 217 of file hmm_model.hpp.

39.277.3.4 GMMHMM()

```
HMM< gmm::GMM>* GMMHMM ( ) [inline]
```

Definition at line 218 of file `hmm_model.hpp`.

39.277.3.5 operator=()

```
HMMModel& operator= (
    const HMMModel & other ) [inline]
```

Copy assignment operator.

Definition at line 102 of file `hmm_model.hpp`.

References `mlpack::hmm::DiagonalGaussianMixtureModelHMM`, `mlpack::hmm::DiscreteHMM`, `mlpack::hmm::GaussianHMM`, and `mlpack::hmm::GaussianMixtureModelHMM`.

39.277.3.6 PerformAction()

```
void PerformAction (
    ExtraInfoType * x ) [inline]
```

Given a functor type, perform that functor with the optional extra info on the **HMM** (p. 1335).

Definition at line 147 of file `hmm_model.hpp`.

References `mlpack::hmm::DiagonalGaussianMixtureModelHMM`, `mlpack::hmm::DiscreteHMM`, `mlpack::hmm::GaussianHMM`, and `mlpack::hmm::GaussianMixtureModelHMM`.

39.277.3.7 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int version ) [inline]
```

Serialize the model.

Definition at line 161 of file `hmm_model.hpp`.

References `mlpack::hmm::DiagonalGaussianMixtureModelHMM`, `mlpack::hmm::DiscreteHMM`, `mlpack::hmm::GaussianHMM`, and `mlpack::hmm::GaussianMixtureModelHMM`.

39.277.3.8 Type()

```
HMMType Type ( ) [inline]
```

Definition at line 195 of file `hmm_model.hpp`.

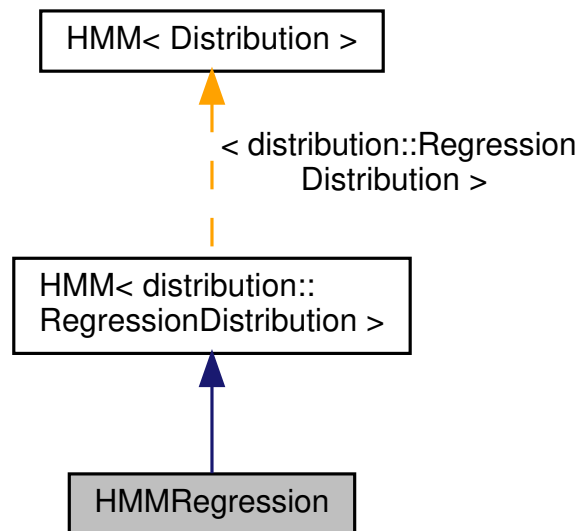
The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hmm/ hmm_model.hpp`

39.278 HMMRegression Class Reference

A class that represents a Hidden Markov Model Regression (HMMR).

Inheritance diagram for HMMRegression:



Public Member Functions

- **HMMRegression** (const size_t states, const **distribution::RegressionDistribution** emissions, const double tolerance=1e-5)
Create the Hidden Markov Model Regression with the given number of hidden states and the given default regression emission.
- **HMMRegression** (const arma::vec &initial, const arma::mat & **transition**, const std::vector< **distribution::RegressionDistribution** > & **emission**, const double tolerance=1e-5)
Create the Hidden Markov Model Regression with the given initial probability vector, the given transition matrix, and the given regression emission distributions.
- double **Estimate** (const arma::mat &predictors, const arma::vec &responses, arma::mat &stateProb, arma::mat &forwardProb, arma::mat &backwardProb, arma::vec &scales) const

Estimate the probabilities of each hidden state at each time step for each given data observation, using the Forward-Backward algorithm.

- double **Estimate** (const arma::mat &predictors, const arma::vec &responses, arma::mat &stateProb) const

Estimate the probabilities of each hidden state at each time step of each given data observation, using the Forward-Backward algorithm.

- void **Filter** (const arma::mat &predictors, const arma::vec &responses, arma::vec &filterSeq, size_t ahead=0) const

HMMR filtering.

- double **LogLikelihood** (const arma::mat &predictors, const arma::vec &responses) const

Compute the log-likelihood of the given predictors and responses.

- double **Predict** (const arma::mat &predictors, const arma::vec &responses, arma::Row< size_t > &stateSeq) const

Compute the most probable hidden state sequence for the given predictors and responses, using the Viterbi algorithm, returning the log-likelihood of the most likely state sequence.

- void **Smooth** (const arma::mat &predictors, const arma::vec &responses, arma::vec &smoothSeq) const

HMM (p. 1335) smoothing.

- void **Train** (const std::vector< arma::mat > &predictors, const std::vector< arma::vec > &responses)

Train the model using the Baum-Welch algorithm, with only the given predictors and responses.

- void **Train** (const std::vector< arma::mat > &predictors, const std::vector< arma::vec > &responses, const std::vector< arma::Row< size_t > > &stateSeq)

Train the model using the given labeled observations; the transition and regression emissions are directly estimated.

Additional Inherited Members

39.278.1 Detailed Description

A class that represents a Hidden Markov Model Regression (HMMR).

HMMR is an extension of Hidden Markov Models to regression analysis. The method is described in (Fridman, 1993) <https://www.ima.umn.edu/preprints/January1994/1195.pdf> An HMMR is a linear regression model whose coefficients are determined by a finite-state Markov chain. The error terms are conditionally independently normally distributed with zero mean and state-dependent variance. Let Q_t be a finite-state Markov chain, X_t a vector of predictors and Y_t a response. The HMMR is $Y_t = X_t \{Q_t\} + \{Q_t\}$

This HMMR class supports training (supervised and unsupervised), prediction of state sequences via the Viterbi algorithm, estimation of state probabilities, filtering and smoothing of responses, and calculation of the log-likelihood of a given sequence.

Usage of the HMMR class generally involves either training an HMMR or loading an already-known HMMR and using to filter a sequence. Example code for supervised training of an HMMR is given below.

```
// Each column is a vector of predictors for a single observation.
arma::mat predictors(5, 100, arma::fill::randn);
// Responses for each observation
arma::vec responses(100, arma::fill::randn);

// Create an untrained HMMR with 3 hidden states
RegressionDistribution rd(predictors, responses);
arma::mat transition("0.5 0.5;" "0.5 0.5;");
std::vector<RegressionDistribution> emissions(2,rd);
HMMRegression hmmr("0.9 0.1", transition, emissions);

// Train the HMM (supply a state sequence to perform supervised training)
std::vector<arma::mat> predictorsSeq(1, predictors);
std::vector< arma::vec> responsesSeq(1, responses);
hmmr.Train(predictorsSeq, responsesSeq);
hmm.Train(observations, states);
```

Once initialized, the HMMR can evaluate the probability of a certain sequence (with **LogLikelihood()** (p. 1359)), predict the most likely sequence of hidden states (with **Predict()** (p. 1359)), estimate the probabilities of each state for a sequence of observations (with **Estimate()** (p. 1357)), or perform filtering or smoothing of observations.

Definition at line 69 of file `hmm_regression.hpp`.

39.278.2 Constructor & Destructor Documentation

39.278.2.1 HMMRegression() [1/2]

```
HMMRegression (
    const size_t states,
    const distribution::RegressionDistribution emissions,
    const double tolerance = 1e-5 ) [inline]
```

Create the Hidden Markov Model Regression with the given number of hidden states and the given default regression emission.

The dimensionality of the observations is taken from the emissions variable, so it is important that the given default emission distribution is set with the correct dimensionality. Alternately, set the dimensionality with **Dimensionality()** (p. 1341). Optionally, the tolerance for convergence of the Baum-Welch algorithm can be set.

By default, the transition matrix and initial probability vector are set to contain equal probability for each state.

Parameters

<i>states</i>	Number of states.
<i>emissions</i>	Default distribution for emissions.
<i>tolerance</i>	Tolerance for convergence of training algorithm (Baum-Welch).

Definition at line 89 of file `hmm_regression.hpp`.

39.278.2.2 HMMRegression() [2/2]

```
HMMRegression (
    const arma::vec & initial,
    const arma::mat & transition,
    const std::vector< distribution::RegressionDistribution > & emission,
    const double tolerance = 1e-5 ) [inline]
```

Create the Hidden Markov Model Regression with the given initial probability vector, the given transition matrix, and the given regression emission distributions.

The dimensionality of the observations of the HMMR are taken from the given emission distributions. Alternately, the dimensionality can be set with **Dimensionality()** (p. 1341).

The initial state probability vector should have length equal to the number of states, and each entry represents the probability of being in the given state at time $T = 0$ (the beginning of a sequence).

The transition matrix should be such that $T(i, j)$ is the probability of transition to state i from state j . The columns of the matrix should sum to 1.

Optionally, the tolerance for convergence of the Baum-Welch algorithm can be set.

Parameters

<i>initial</i>	Initial state probabilities.
<i>transition</i>	Transition matrix.
<i>emission</i>	Emission distributions.
<i>tolerance</i>	Tolerance for convergence of training algorithm (Baum-Welch).

Definition at line 119 of file `hmm_regression.hpp`.

References `HMMRegression::Estimate()`, `HMMRegression::Filter()`, `HMMRegression::LogLikelihood()`, `HMMRegression::Predict()`, `HMMRegression::Smooth()`, and `HMMRegression::Train()`.

39.278.3 Member Function Documentation

39.278.3.1 Estimate() [1/2]

```
double Estimate (
    const arma::mat & predictors,
    const arma::vec & responses,
    arma::mat & stateProb,
    arma::mat & forwardProb,
    arma::mat & backwardProb,
    arma::vec & scales ) const
```

Estimate the probabilities of each hidden state at each time step for each given data observation, using the Forward-Backward algorithm.

Each matrix which is returned has columns equal to the number of data observations, and rows equal to the number of hidden states in the model. The log-likelihood of the most probable sequence is returned.

Parameters

<i>predictors</i>	Vector of predictor sequences.
<i>responses</i>	Vector of response sequences.
<i>stateProb</i>	Matrix in which the probabilities of each state at each time interval will be stored.
<i>forwardProb</i>	Matrix in which the forward probabilities of each state at each time interval will be stored.
<i>backwardProb</i>	Matrix in which the backward probabilities of each state at each time interval will be stored.
<i>scales</i>	Vector in which the scaling factors at each time interval will be stored.

Returns

Log-likelihood of most likely state sequence.

Referenced by HMMRegression::HMMRegression().

39.278.3.2 Estimate() [2/2]

```
double Estimate (
    const arma::mat & predictors,
    const arma::vec & responses,
    arma::mat & stateProb ) const
```

Estimate the probabilities of each hidden state at each time step of each given data observation, using the Forward-↔ Backward algorithm.

The returned matrix of state probabilities has columns equal to the number of data observations, and rows equal to the number of hidden states in the model. The log-likelihood of the most probable sequence is returned.

Parameters

<i>predictors</i>	Vector of predictor sequences.
<i>responses</i>	Vector of response sequences.
<i>stateProb</i>	Probabilities of each state at each time interval.

Returns

Log-likelihood of most likely state sequence.

39.278.3.3 Filter()

```
void Filter (
    const arma::mat & predictors,
    const arma::vec & responses,
    arma::vec & filterSeq,
    size_t ahead = 0 ) const
```

HMMR filtering.

Computes the k-step-ahead expected response at each time conditioned only on prior observations. That is $E\{Y[t+k] \mid Y[0], \dots, Y[t]\}$. The returned matrix has columns equal to the number of observations. Note that the expectation may not be meaningful for discrete emissions.

Parameters

<i>predictors</i>	Vector of predictor sequences.
<i>responses</i>	Vector of response sequences.
<i>initial</i>	Distribution of initial state.
<i>ahead</i>	Number of steps ahead (k) for expectations.
<i>filterSeq</i>	Vector in which the expected emission sequence will be stored.

Referenced by HMMRegression::HMMRegression().

39.278.3.4 LogLikelihood()

```
double LogLikelihood (
    const arma::mat & predictors,
    const arma::vec & responses ) const
```

Compute the log-likelihood of the given predictors and responses.

Parameters

<i>predictors</i>	Vector of predictor sequences.
<i>responses</i>	Vector of response sequences.

Returns

Log-likelihood of the given sequence.

Referenced by HMMRegression::HMMRegression().

39.278.3.5 Predict()

```
double Predict (
    const arma::mat & predictors,
    const arma::vec & responses,
    arma::Row< size_t > & stateSeq ) const
```

Compute the most probable hidden state sequence for the given predictors and responses, using the Viterbi algorithm, returning the log-likelihood of the most likely state sequence.

Parameters

<i>predictors</i>	Vector of predictor sequences.
<i>responses</i>	Vector of response sequences.
<i>stateSeq</i>	Vector in which the most probable state sequence will be stored.

Returns

Log-likelihood of most probable state sequence.

Referenced by `HMMRegression::HMMRegression()`.

39.278.3.6 Smooth()

```
void Smooth (
    const arma::mat & predictors,
    const arma::vec & responses,
    arma::vec & smoothSeq ) const
```

HMM (p. 1335) smoothing.

Computes expected emission at each time conditioned on all observations. That is $E\{Y[t] \mid Y[0], \dots, Y[T]\}$. The returned matrix has columns equal to the number of observations. Note that the expectation may not be meaningful for discrete emissions.

Parameters

<i>predictors</i>	Vector of predictor sequences.
<i>responses</i>	Vector of response sequences..
<i>initial</i>	Distribution of initial state.
<i>smoothSeq</i>	Vector in which the expected emission sequence will be stored.

Referenced by `HMMRegression::HMMRegression()`.

39.278.3.7 Train() [1/2]

```
void Train (
    const std::vector< arma::mat > & predictors,
    const std::vector< arma::vec > & responses )
```

Train the model using the Baum-Welch algorithm, with only the given predictors and responses.

Instead of giving a guess transition and emission here, do that in the constructor. Each matrix in the vector of predictors corresponds to an individual data sequence, and likewise for each vec in the vector of responses. The number of rows in each matrix of predictors plus one should be equal to the dimensionality of the **HMM** (p. 1335) (which is set in the constructor).

It is preferable to use the other overload of **Train()** (p. 1360), with labeled data. That will produce much better results. However, if labeled data is unavailable, this will work. In addition, it is possible to use **Train()** (p. 1360) with labeled data first, and then continue to train the model using this overload of **Train()** (p. 1360) with unlabeled data.

The tolerance of the Baum-Welch algorithm can be set either in the constructor or with the **Tolerance()** (p. 1347) method. When the change in log-likelihood of the model between iterations is less than the tolerance, the Baum-Welch algorithm terminates.

Note

Train() (p. 1360) can be called multiple times with different sequences; each time it is called, it uses the current parameters of the **HMM** (p. 1335) as a starting point for training.

Parameters

<i>predictors</i>	Vector of predictor sequences.
<i>responses</i>	Vector of response sequences.

Referenced by HMMRegression::HMMRegression().

39.278.3.8 Train() [2/2]

```
void Train (
    const std::vector< arma::mat > & predictors,
    const std::vector< arma::vec > & responses,
    const std::vector< arma::Row< size_t > > & stateSeq )
```

Train the model using the given labeled observations; the transition and regression emissions are directly estimated.

Each matrix in the vector of predictors corresponds to an individual data sequence, and likewise for each vec in the vector of responses. The number of rows in each matrix of predictors plus one should be equal to the dimensionality of the **HMM** (p. 1335) (which is set in the constructor).

Note

Train() (p. 1360) can be called multiple times with different sequences; each time it is called, it uses the current parameters of the HMMR as a starting point for training.

Parameters

<i>predictors</i>	Vector of predictor sequences.
<i>responses</i>	Vector of response sequences.
<i>stateSeq</i>	Vector of state sequences, corresponding to each observation.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hmm/ **hmm_regression.hpp**

39.279 CVFunction< CVType, MLAlgorithm, TotalArgs, BoundArgs > Class Template Reference

This wrapper serves for adapting the interface of the cross-validation classes to the one that can be utilized by the mlpack optimizers.

Public Member Functions

- **CVFunction** (CVType &cv, **data::DatasetMapper**< **data::IncrementPolicy**, double > &datasetInfo, const double relativeDelta, const double minDelta, const BoundArgs &... args)
*Initialize a **CVFunction** (p. 1361) object.*
- **MLAlgorithm** & **BestModel** ()
Access and modify the best model so far.
- double **Evaluate** (const arma::mat ¶meters)
Run cross-validation with the bound and passed parameters.
- void **Gradient** (const arma::mat ¶meters, arma::mat &gradient)
*Evaluate numerically the gradient of the **CVFunction** (p. 1361) with the given parameters.*

39.279.1 Detailed Description

```
template<typename CVType, typename MLAlgorithm, size_t TotalArgs, typename... BoundArgs>
class mlpack::hpt::CVFunction< CVType, MLAlgorithm, TotalArgs, BoundArgs >
```

This wrapper serves for adapting the interface of the cross-validation classes to the one that can be utilized by the mlpack optimizers.

This class is not supposed to be used directly by users. To tune hyper-parameters see **HyperParameterTuner** (p. 1375).

Template Parameters

<i>CVType</i>	A cross-validation strategy.
<i>MLAlgorithm</i>	The machine learning algorithm used in cross-validation.
<i>TotalArgs</i>	The total number of arguments that are supposed to be passed to the Evaluate method of a CVType object.
<i>BoundArgs</i>	Types of arguments (wrapped into the BoundArg struct) that should be passed into the Evaluate method of a CVType object but are not going to be passed into the Evaluate method of a CVFunction (p. 1361) object.

Definition at line 39 of file cv_function.hpp.

39.279.2 Constructor & Destructor Documentation

39.279.2.1 CVFunction()

```
CVFunction (
    CVType & cv,
    data::DatasetMapper< data::IncrementPolicy, double > & datasetInfo,
    const double relativeDelta,
```

```
const double minDelta,  
const BoundArgs &... args )
```

Initialize a **CVFunction** (p. 1361) object.

Parameters

<i>cv</i>	A cross-validation object.
<i>datasetInfo</i>	Information on each parameter (categorical/numeric). Contains mappings from optimizer-passed <code>size_t</code> indices to double values that should be used.
<i>relativeDelta</i>	Relative increase of arguments for calculation of partial derivatives (by the definition). The exact increase for some particular argument is equal to the absolute value of the argument multiplied by the relative increase (see also the documentation for the <code>minDelta</code> parameter).
<i>minDelta</i>	Minimum increase of arguments for calculation of partial derivatives (by the definition). This value is going to be used when it is greater than the increase calculated with the rules described in the documentation for the <code>relativeDelta</code> parameter.
<i>BoundArgs</i>	Arguments that should be passed into the Evaluate method of the CVType object but are not going to be passed into the Evaluate method of this object.

39.279.3 Member Function Documentation

39.279.3.1 BestModel()

```
MLAlgorithm& BestModel ( ) [inline]
```

Access and modify the best model so far.

Definition at line 87 of file `cv_function.hpp`.

References `CVFunction< CVType, MLAlgorithm, TotalArgs, BoundArgs >::Evaluate()`.

39.279.3.2 Evaluate()

```
double Evaluate (
    const arma::mat & parameters )
```

Run cross-validation with the bound and passed parameters.

Parameters

<i>parameters</i>	Arguments (rather than the bound arguments) that should be passed into the Evaluate method of the CVType object.
-------------------	--

Referenced by `CVFunction< CVType, MLAlgorithm, TotalArgs, BoundArgs >::BestModel()`.

39.279.3.3 Gradient()

```
void Gradient (
    const arma::mat & parameters,
    arma::mat & gradient )
```

Evaluate numerically the gradient of the **CVFunction** (p. 1361) with the given parameters.

Parameters

<i>parameters</i>	Arguments (rather than the bound arguments) that should be passed into the Evaluate method of the CVType object.
<i>gradient</i>	Vector to output the gradient into.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ **cv_function.hpp**

39.280 DeduceHyperParameterTypes< Args > Struct Template Reference

A type function for deducing types of hyper-parameters from types of arguments in the Optimize method in **Hyper↔ParameterTuner** (p. 1375).

Classes

- struct **ResultHolder**

39.280.1 Detailed Description

```
template<typename... Args>
struct mlpack::hpt::DeduceHyperParameterTypes< Args >
```

A type function for deducing types of hyper-parameters from types of arguments in the Optimize method in **Hyper↔ParameterTuner** (p. 1375).

We start by putting all types of the arguments into Args, and then process each of them one by one and put results into the internal struct **ResultHolder** (p. 1366). By the end Args become empty, while **ResultHolder** (p. 1366) holds the tuple type of hyper-parameters.

Here we declare and define **DeduceHyperParameterTypes** (p. 1365) for the end phase when Args are empty (all argument types have been processed).

Definition at line 34 of file deduce_hp_types.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ **deduce_hp_types.hpp**

39.281 DeduceHyperParameterTypes< Args >::ResultHolder< HPTypes > Struct Template Reference

Public Types

- using **TupleType** = std::tuple< HPTypes... >

39.281.1 Detailed Description

```
template<typename... Args>
template<typename... HPTypes>
struct mlpack::hpt::DeduceHyperParameterTypes< Args >::ResultHolder< HPTypes >
```

Definition at line 37 of file deduce_hp_types.hpp.

39.281.2 Member Typedef Documentation

39.281.2.1 TupleType

```
using TupleType = std::tuple<HPTypes...>
```

Definition at line 39 of file deduce_hp_types.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ **deduce_hp_types.hpp**

39.282 DeduceHyperParameterTypes< PreFixedArg< T >, Args... > Struct Template Reference

Defining **DeduceHyperParameterTypes** (p. 1365) for the case when not all argument types have been processed, and the next one is the type of an argument that should be fixed.

Classes

- struct **ResultHolder**

Public Types

- using **TupleType** = typename ResultHolder<>:: **TupleType**

39.282.1 Detailed Description

```
template<typename T, typename... Args>
struct mlpack::hpt::DeduceHyperParameterTypes< PreFixedArg< T >, Args... >
```

Defining **DeduceHyperParameterTypes** (p. 1365) for the case when not all argument types have been processed, and the next one is the type of an argument that should be fixed.

Definition at line 109 of file deduce_hp_types.hpp.

39.282.2 Member Typedef Documentation

39.282.2.1 TupleType

```
using TupleType = typename ResultHolder<>:: TupleType
```

Definition at line 118 of file deduce_hp_types.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ **deduce_hp_types.hpp**

39.283 DeduceHyperParameterTypes< PreFixedArg< T >, Args... >::ResultHolder< HP←Types > Struct Template Reference

Public Types

- using **TupleType** = typename **DeduceHyperParameterTypes**< Args... >::template ResultHolder< HPTypes... >:: **TupleType**

39.283.1 Detailed Description

```
template<typename T, typename... Args>
template<typename... HPTypes>
struct mlpack::hpt::DeduceHyperParameterTypes< PreFixedArg< T >, Args... >::ResultHolder< HPTypes >
```

Definition at line 112 of file deduce_hp_types.hpp.

39.283.2 Member Typedef Documentation

39.283.2.1 TupleType

```
using TupleType = typename DeduceHyperParameterTypes<Args...>::template ResultHolder<HPTypes...>↔
:: TupleType
```

Definition at line 115 of file deduce_hp_types.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ **deduce_hp_types.hpp**

39.284 DeduceHyperParameterTypes< T, Args... > Struct Template Reference

Defining **DeduceHyperParameterTypes** (p. 1365) for the case when not all argument types have been processed, and the next one (T) is a collection type or an arithmetic type.

Classes

- struct **IsCollectionType**
A type function to check whether Type is a collection type (for that it should define value_type).
- struct **ResultHolder**
- struct **ResultHPType**
A type function to deduce the result hyper-parameter type for ArgumentType.
- struct **ResultHPType**< **ArithmeticType**, **true** >
- struct **ResultHPType**< **CollectionType**, **false** >

Public Types

- using **TupleType** = typename ResultHolder<>:: **TupleType**

39.284.1 Detailed Description

```
template<typename T, typename... Args>
struct mlpack::hpt::DeduceHyperParameterTypes< T, Args... >
```

Defining **DeduceHyperParameterTypes** (p. 1365) for the case when not all argument types have been processed, and the next one (T) is a collection type or an arithmetic type.

Definition at line 49 of file deduce_hp_types.hpp.

39.284.2 Member Typedef Documentation

39.284.2.1 TupleType

```
using TupleType = typename ResultHolder<>:: TupleType
```

Definition at line 100 of file deduce_hp_types.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ **deduce_hp_types.hpp**

39.285 DeduceHyperParameterTypes< T, Args... >::IsCollectionType< Type > Struct Template Reference

A type function to check whether Type is a collection type (for that it should define value_type).

Public Types

- using **No** = char[2]
- using **Yes** = char[1]

Static Public Member Functions

- template<typename TypeToCheck >
static **Yes & Check** (typename TypeToCheck::value_type *)
- template<typename >
static **No & Check** (...)

Static Public Attributes

- static const bool **value**

39.285.1 Detailed Description

```
template<typename T, typename... Args>  
template<typename Type>  
struct mlpack::hpt::DeduceHyperParameterTypes< T, Args... >::IsCollectionType< Type >
```

A type function to check whether Type is a collection type (for that it should define value_type).

Definition at line 69 of file deduce_hp_types.hpp.

39.285.2 Member Typedef Documentation

39.285.2.1 No

```
using No = char[2]
```

Definition at line 72 of file deduce_hp_types.hpp.

39.285.2.2 Yes

```
using Yes = char[1]
```

Definition at line 71 of file deduce_hp_types.hpp.

39.285.3 Member Function Documentation

39.285.3.1 Check() [1/2]

```
static Yes& Check (  
    typename TypeToCheck::value_type * ) [static]
```

39.285.3.2 Check() [2/2]

```
static No& Check (  
    ... ) [static]
```

39.285.4 Member Data Documentation

39.285.4.1 value

```
const bool value [static]
```

Initial value:

```
=  
    sizeof(decltype(Check<Type>(0))) == sizeof(Yes)
```

Definition at line 79 of file deduce_hp_types.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ **deduce_hp_types.hpp**

39.286 DeduceHyperParameterTypes< T, Args... >::ResultHolder< HPTypes > Struct Template Reference

Public Types

- using **TupleType** = typename **DeduceHyperParameterTypes**< Args... >::template ResultHolder< HPTypes..., typename ResultHPTYPE< T >::Type >:: **TupleType**

39.286.1 Detailed Description

```
template<typename T, typename... Args>  
template<typename... HPTypes>  
struct mlpack::hpt::DeduceHyperParameterTypes< T, Args... >::ResultHolder< HPTypes >
```

Definition at line 94 of file deduce_hp_types.hpp.

39.286.2 Member Typedef Documentation

39.286.2.1 TupleType

```
using TupleType = typename DeduceHyperParameterTypes<Args...>::template ResultHolder<HPTypes...,  
typename ResultHPTYPE<T>::Type>:: TupleType
```

Definition at line 97 of file deduce_hp_types.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ **deduce_hp_types.hpp**

39.287 DeduceHyperParameterTypes< T, Args... >::ResultHPTType< ArgumentType, IsArithmetic > Struct Template Reference

A type function to deduce the result hyper-parameter type for ArgumentType.

39.287.1 Detailed Description

```
template<typename T, typename... Args>
template<typename ArgumentType, bool IsArithmetic = std::is_arithmetic<ArgumentType>::value>
struct mlpack::hpt::DeduceHyperParameterTypes< T, Args... >::ResultHPTType< ArgumentType, IsArithmetic >
```

A type function to deduce the result hyper-parameter type for ArgumentType.

Definition at line 56 of file deduce_hp_types.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ **deduce_hp_types.hpp**

39.288 DeduceHyperParameterTypes< T, Args... >::ResultHPTType< ArithmeticType, true > Struct Template Reference

Public Types

- using **Type** = ArithmeticType

39.288.1 Detailed Description

```
template<typename T, typename... Args>
template<typename ArithmeticType>
struct mlpack::hpt::DeduceHyperParameterTypes< T, Args... >::ResultHPTType< ArithmeticType, true >
```

Definition at line 59 of file deduce_hp_types.hpp.

39.288.2 Member Typedef Documentation

39.288.2.1 Type

```
using Type = ArithmeticType
```

Definition at line 61 of file deduce_hp_types.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ **deduce_hp_types.hpp**

39.289 DeduceHyperParameterTypes< T, Args... >::ResultHPTType< CollectionType, false > Struct Template Reference

Public Types

- using **Type** = typename CollectionType::value_type

39.289.1 Detailed Description

```
template<typename T, typename... Args>
template<typename CollectionType>
struct mlpack::hpt::DeduceHyperParameterTypes< T, Args... >::ResultHPTType< CollectionType, false >
```

Definition at line 84 of file deduce_hp_types.hpp.

39.289.2 Member Typedef Documentation

39.289.2.1 Type

```
using Type = typename CollectionType::value_type
```

Definition at line 90 of file deduce_hp_types.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ **deduce_hp_types.hpp**

39.290 FixedArg< T, I > Struct Template Reference

A struct for storing information about a fixed argument.

Public Attributes

- `const T & value`

The value of the fixed argument.

Static Public Attributes

- `static const size_t index = I`

The index of the fixed argument.

39.290.1 Detailed Description

```
template<typename T, size_t I>
struct mlpack::hpt::FixedArg< T, I >
```

A struct for storing information about a fixed argument.

Objects of this type are supposed to be passed into the **CVFunction** (p. 1361) constructor.

This struct is not meant to be used directly by users. Rather use the **mlpack::hpt::Fixed** (p. 398) function.

Template Parameters

<i>T</i>	The type of the fixed argument.
<i>I</i>	The index of the fixed argument.

Definition at line 52 of file fixed.hpp.

39.290.2 Member Data Documentation

39.290.2.1 index

```
const size_t index = I [static]
```

The index of the fixed argument.

Definition at line 55 of file fixed.hpp.

39.290.2.2 value

```
const T& value
```

The value of the fixed argument.

Definition at line 58 of file fixed.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ **fixed.hpp**

39.291 HyperParameterTuner< MLAlgorithm, Metric, CV, OptimizerType, MatType, PredictionsType, WeightsType > Class Template Reference

The class **HyperParameterTuner** (p. 1375) for the given MLAlgorithm utilizes the provided Optimizer to find the values of hyper-parameters that optimize the value of the given Metric.

Public Member Functions

- template<typename... CVArgs>
HyperParameterTuner (const CVArgs &...args)
*Create a **HyperParameterTuner** (p. 1375) object by passing constructor arguments for the given cross-validation strategy (the CV class).*
- const MLAlgorithm & **BestModel** () const
Get the best model from the last run.
- MLAlgorithm & **BestModel** ()
Modify the best model from the last run.
- double **BestObjective** () const
Get the performance measurement of the best model from the last run.
- double **MinDelta** () const
Get minimum increase of arguments for calculation of partial derivatives (by the definition) in gradient-based optimization.
- double & **MinDelta** ()
Modify minimum increase of arguments for calculation of partial derivatives (by the definition) in gradient-based optimization.
- template<typename... Args>
TupleOfHyperParameters< Args... > **Optimize** (const Args &... args)
Find the best hyper-parameters by using the given Optimizer.
- OptimizerType & **Optimizer** ()
Access and modify the optimizer.
- double **RelativeDelta** () const
Get relative increase of arguments for calculation of partial derivatives (by the definition) in gradient-based optimization.
- double & **RelativeDelta** ()
Modify relative increase of arguments for calculation of partial derivatives (by the definition) in gradient-based optimization.

39.291.1 Detailed Description

```
template<typename MlAlgorithm, typename Metric, template< typename, typename, typename, typename > class CV,
typename OptimizerType = ens::GridSearch, typename MatType = arma::mat, typename PredictionsType = typename cv::MetalInfo<↵
Extractor<MlAlgorithm, MatType>::PredictionsType, typename WeightsType = typename cv::MetalInfoExtractor<MlAlgorithm, Mat<↵
Type, PredictionsType>::WeightsType>
class mlpack::hpt::HyperParameterTuner< MlAlgorithm, Metric, CV, OptimizerType, MatType, PredictionsType, WeightsType >
```

The class **HyperParameterTuner** (p. 1375) for the given **MlAlgorithm** utilizes the provided **Optimizer** to find the values of hyper-parameters that optimize the value of the given **Metric**.

The value of the **Metric** is calculated by performing cross-validation with the provided cross-validation strategy.

To construct a **HyperParameterTuner** (p. 1375) object you need to pass the same arguments as for construction of an object of the given **CV** class. For example, we can use the following code to try to find a good **lambda** value for **LinearRegression**.

```
// 100-point 5-dimensional random dataset.
arma::mat data = arma::randu<arma::mat>(5, 100);
// Noisy responses retrieved by a random linear transformation of data.
arma::rowvec responses = arma::randu<arma::rowvec>(5) * data +
    0.1 * arma::randn<arma::rowvec>(100);

// Using 80% of data for training and remaining 20% for assessing MSE.
double validationSize = 0.2;
HyperParameterTuner<LinearRegression, MSE, SimpleCV> hpt(validationSize,
    data, responses);

// Finding the best value for lambda from the values 0.0, 0.001, 0.01, 0.1,
// and 1.0.
arma::vec lambdas{0.0, 0.001, 0.01, 0.1, 1.0};
double bestLambda;
std::tie(bestLambda) = hpt.Optimize(lambdas);
```

When some hyper-parameters should not be optimized, you can specify values for them with the **Fixed** function as in the following example of finding good **lambda1** and **lambda2** values for **LARS**.

```
HyperParameterTuner<LARS, MSE, SimpleCV> hpt2(validationSize, data,
    responses);

bool transposeData = true;
bool useCholesky = false;
arma::vec lambda1Set{0.0, 0.001, 0.01, 0.1, 1.0};
arma::vec lambda2Set{0.0, 0.002, 0.02, 0.2, 2.0};

double bestLambda1, bestLambda2;
std::tie(bestLambda1, bestLambda2) = hpt2.Optimize(Fixed(transposeData),
    Fixed(useCholesky), lambda1Set, lambda2Set);
```

Template Parameters

<i>MlAlgorithm</i>	A machine learning algorithm.
<i>Metric</i>	A metric to assess the quality of a trained model.
<i>CV</i>	A cross-validation strategy used to assess a set of hyper-parameters.
<i>OptimizerType</i>	An optimization strategy (GridSearch and GradientDescent are supported).
<i>MatType</i>	The type of data.
<i>PredictionsType</i>	The type of predictions (should be passed when the predictions type is a template parameter in Train methods of the given MlAlgorithm ; arma::Row<size_t> will be used otherwise).
<i>WeightsType</i>	The type of weights (should be passed when weighted learning is supported, and the weights type is a template parameter in Train methods of the given MlAlgorithm ; arma::vec will be used otherwise).

Definition at line 96 of file hpt.hpp.

39.291.2 Constructor & Destructor Documentation

39.291.2.1 HyperParameterTuner()

```
HyperParameterTuner (  
    const CVArgs &... args )
```

Create a **HyperParameterTuner** (p. 1375) object by passing constructor arguments for the given cross-validation strategy (the CV class).

Parameters

<i>args</i>	Constructor arguments for the given cross-validation strategy (the CV class).
-------------	---

39.291.3 Member Function Documentation

39.291.3.1 BestModel() [1/2]

```
const MLAlgorithm& BestModel ( ) const [inline]
```

Get the best model from the last run.

Definition at line 184 of file hpt.hpp.

39.291.3.2 BestModel() [2/2]

```
MLAlgorithm& BestModel ( ) [inline]
```

Modify the best model from the last run.

Definition at line 187 of file hpt.hpp.

39.291.3.3 BestObjective()

```
double BestObjective ( ) const [inline]
```

Get the performance measurement of the best model from the last run.

Definition at line 181 of file hpt.hpp.

39.291.3.4 MinDelta() [1/2]

```
double MinDelta ( ) const [inline]
```

Get minimum increase of arguments for calculation of partial derivatives (by the definition) in gradient-based optimization.

This value is going to be used when it is greater than the increase calculated with the rules described in the documentation for **RelativeDelta()** (p. 1379).

The default value is 1e-10.

Definition at line 142 of file hpt.hpp.

39.291.3.5 MinDelta() [2/2]

```
double& MinDelta ( ) [inline]
```

Modify minimum increase of arguments for calculation of partial derivatives (by the definition) in gradient-based optimization.

This value is going to be used when it is greater than the increase calculated with the rules described in the documentation for **RelativeDelta()** (p. 1379).

The default value is 1e-10.

Definition at line 152 of file hpt.hpp.

References `HyperParameterTuner< MLAlgorithm, Metric, CV, OptimizerType, MatType, PredictionsType, WeightsType >::Optimize()`.

39.291.3.6 Optimize()

```
TupleOfHyperParameters<Args...> Optimize (
    const Args &... args )
```

Find the best hyper-parameters by using the given Optimizer.

For each hyper-parameter one of the following should be passed as an argument.

1. A set of values to choose from (when using GridSearch as an optimizer). The set of values should be an STL-compatible container (it should provide begin() and end() methods returning iterators).
2. A starting value (when using any other optimizer than GridSearch).
3. A value fixed by using the function **mlpack::hpt::Fixed** (p.398). In this case the hyper-parameter will not be optimized.

All arguments should be passed in the same order as if the corresponding hyper-parameters would be passed into the Evaluate method of the given CV class (in the order as they appear in the constructor(s) of the given MLAlgorithm). Also, arguments for all required hyper-parameters (ones that don't have default values in the corresponding MLAlgorithm constructor) should be provided.

The method returns a tuple of values for hyper-parameters that haven't been fixed.

Parameters

<i>args</i>	Arguments corresponding to hyper-parameters (see the method description for more information).
-------------	--

Referenced by HyperParameterTuner< MLAlgorithm, Metric, CV, OptimizerType, MatType, PredictionsType, WeightsType >::MinDelta().

39.291.3.7 Optimizer()

```
OptimizerType& Optimizer ( ) [inline]
```

Access and modify the optimizer.

Definition at line 110 of file hpt.hpp.

39.291.3.8 RelativeDelta() [1/2]

```
double RelativeDelta ( ) const [inline]
```

Get relative increase of arguments for calculation of partial derivatives (by the definition) in gradient-based optimization.

The exact increase for some particular argument is equal to the absolute value of the argument multiplied by the relative increase (see also the documentation for **MinDelta()** (p. 1378)).

The default value is 0.01.

Definition at line 121 of file hpt.hpp.

39.291.3.9 RelativeDelta() [2/2]

```
double& RelativeDelta ( ) [inline]
```

Modify relative increase of arguments for calculation of partial derivatives (by the definition) in gradient-based optimization.

The exact increase for some particular argument is equal to the absolute value of the argument multiplied by the relative increase (see also the documentation for **MinDelta()** (p. 1378)).

The default value is 0.01.

Definition at line 132 of file hpt.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ **hpt.hpp**

39.292 IsPreFixedArg< T > Class Template Reference

A type function for checking whether the given type is **PreFixedArg** (p. 1381).

Static Public Attributes

- static const bool **value** = Implementation<typename std::decay<T>::type>::value

39.292.1 Detailed Description

```
template<typename T>
class mlpack::hpt::IsPreFixedArg< T >
```

A type function for checking whether the given type is **PreFixedArg** (p. 1381).

Definition at line 96 of file fixed.hpp.

39.292.2 Member Data Documentation

39.292.2.1 value

```
const bool value = Implementation<typename std::decay<T>::type>::value [static]
```

Definition at line 105 of file fixed.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ **fixed.hpp**

39.293 PreFixedArg< T > Struct Template Reference

A struct for marking arguments as ones that should be fixed (it can be useful for the Optimize method of **Hyper**↔**ParameterTuner** (p. 1375)).

Public Types

- using **Type** = T

Public Attributes

- const T **value**

39.293.1 Detailed Description

```
template<typename T>
struct mlpack::hpt::PreFixedArg< T >
```

A struct for marking arguments as ones that should be fixed (it can be useful for the Optimize method of **Hyper**↔**ParameterTuner** (p. 1375)).

Arguments of this type are supposed to be converted into structs of the type **FixedArg** (p. 1373) by adding information about argument positions.

This struct is not meant to be used directly by users. Rather use the **mlpack::hpt::Fixed** (p. 398) function.

Definition at line 23 of file fixed.hpp.

39.293.2 Member Typedef Documentation

39.293.2.1 Type

```
using Type = T
```

Definition at line 73 of file fixed.hpp.

39.293.3 Member Data Documentation

39.293.3.1 value

```
const T value
```

Definition at line 75 of file fixed.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ **fixed.hpp**

39.294 PreFixedArg< T & > Struct Template Reference

The specialization of the template for references.

Public Types

- using **Type** = T

Public Attributes

- const T & **value**

39.294.1 Detailed Description

```
template<typename T>  
struct mlpack::hpt::PreFixedArg< T & >
```

The specialization of the template for references.

This struct is not meant to be used directly by users. Rather use the **mlpack::hpt::Fixed** (p. 398) function.

Definition at line 85 of file fixed.hpp.

39.294.2 Member Typedef Documentation

39.294.2.1 Type

```
using Type = T
```

Definition at line 87 of file fixed.hpp.

39.294.3 Member Data Documentation

39.294.3.1 value

```
const T& value
```

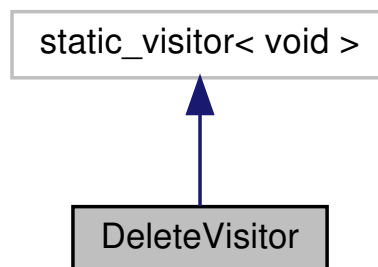
Definition at line 89 of file fixed.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/ **fixed.hpp**

39.295 DeleteVisitor Class Reference

Inheritance diagram for DeleteVisitor:



Public Member Functions

- `template<typename KDEType >`
`void operator() (KDEType *kde) const`
Delete KDEType instance.

39.295.1 Detailed Description

Definition at line 185 of file kde_model.hpp.

39.295.2 Member Function Documentation

39.295.2.1 operator>()

```
void operator() (
    KDEType * kde ) const
```

Delete KDEType instance.

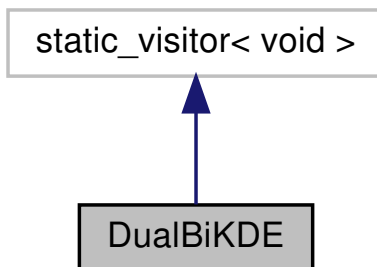
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/ **kde_model.hpp**

39.296 DualBiKDE Class Reference

DualBiKDE (p. 1384) computes a Kernel Density Estimation on the given KDEType.

Inheritance diagram for DualBiKDE:



Public Types

- `template<typename KernelType , template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType>`
`using KDETypeT = KDEType< KernelType, TreeType >`

Alias template necessary for Visual C++ compiler.

Public Member Functions

- **DualBiKDE** (arma::mat &&querySet, arma::vec &estimations)
***DualBiKDE** (p. 1384) constructor. Takes ownership of the given querySet.*
- template<typename KernelType, template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType>
void **operator()** (**KDETypeT**< KernelType, TreeType > *kde) const
*Default **DualBiKDE** (p. 1384) on some KDEType.*

39.296.1 Detailed Description

DualBiKDE (p. 1384) computes a Kernel Density Estimation on the given KDEType.

It performs a bichromatic **KDE** (p. 1387).

Definition at line 118 of file kde_model.hpp.

39.296.2 Member Typedef Documentation

39.296.2.1 KDETypeT

```
using KDETypeT = KDEType<KernelType, TreeType>
```

Alias template necessary for Visual C++ compiler.

Definition at line 136 of file kde_model.hpp.

39.296.3 Constructor & Destructor Documentation

39.296.3.1 DualBiKDE()

```
DualBiKDE (  
    arma::mat && querySet,  
    arma::vec & estimations )
```

DualBiKDE (p. 1384) constructor. Takes ownership of the given querySet.

39.296.4 Member Function Documentation

39.296.4.1 operator()()

```
void operator() (
    KDETypeT< KernelType, TreeType > * kde ) const
```

Default **DualBiKDE** (p. 1384) on some KDEType.

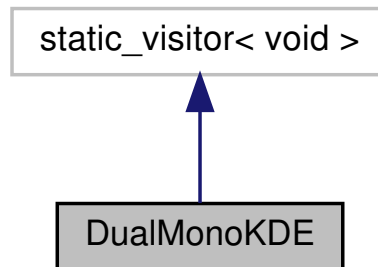
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/ **kde_model.hpp**

39.297 DualMonoKDE Class Reference

DualMonoKDE (p. 1386) computes a Kernel Density Estimation on the given KDEType.

Inheritance diagram for DualMonoKDE:



Public Types

- `template<typename KernelType , template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType> using KDETypeT = KDEType< KernelType, TreeType >`
Alias template necessary for Visual C++ compiler.

Public Member Functions

- **DualMonoKDE** (arma::vec &estimations)
DualMonoKDE (p. 1386) constructor.
- `template<typename KernelType , template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType> void operator() (KDETypeT< KernelType, TreeType > *kde) const`
Default DualMonoKDE (p. 1386) on some KDEType.

39.297.1 Detailed Description

DualMonoKDE (p. 1386) computes a Kernel Density Estimation on the given KDEType.

It performs a monochromatic **KDE** (p. 1387).

Definition at line 87 of file `kde_model.hpp`.

39.297.2 Member Typedef Documentation

39.297.2.1 KDETypeT

```
using KDETypeT = KDEType<KernelType, TreeType>
```

Alias template necessary for Visual C++ compiler.

Definition at line 99 of file kde_model.hpp.

39.297.3 Constructor & Destructor Documentation

39.297.3.1 DualMonoKDE()

```
DualMonoKDE (
    arma::vec & estimations )
```

DualMonoKDE (p. 1386) constructor.

39.297.4 Member Function Documentation

39.297.4.1 operator>()

```
void operator() (
    KDETypeT< KernelType, TreeType > * kde ) const
```

Default **DualMonoKDE** (p. 1386) on some KDEType.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/ **kde_model.hpp**

39.298 KDE< KernelType, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTreeTraversalType > Class Template Reference

The **KDE** (p. 1387) class is a template class for performing Kernel Density Estimations.

Public Types

- typedef TreeType< MetricType, **kde::KDEStat**, MatType > **Tree**
Convenience typedef.

Public Member Functions

- **KDE** (const double relError=0.05, const double absError=0, KernelType kernel=KernelType(), const **KDEMode** mode= **DUAL_TREE_MODE**, MetricType metric=MetricType())
*Initialize **KDE** (p. 1387) object using custom instantiated Metric and Kernel objects.*
- **KDE** (const **KDE** &other)
*Construct **KDE** (p. 1387) object as a copy of the given model.*
- **KDE** (**KDE** &&other)
*Construct **KDE** (p. 1387) object taking ownership of the given model.*
- ~**KDE** ()
*Destroy the **KDE** (p. 1387) object.*
- double **AbsoluteError** () const
Get absolute error tolerance.
- void **AbsoluteError** (const double newError)
Modify absolute error tolerance ($0 \leq \text{newError}$).
- void **Evaluate** (MatType querySet, arma::vec &estimations)
Estimate density of each point in the query set given the data of the reference set.
- void **Evaluate** (**Tree** *queryTree, const std::vector< size_t > &oldFromNewQueries, arma::vec &estimations)
Estimate density of each point in the query set given the data of an already created query tree.
- void **Evaluate** (arma::vec &estimations)
Estimate density of each point in the reference set given the data of the reference set.
- bool **IsTrained** () const
*Check whether **KDE** (p. 1387) model is trained or not.*
- const KernelType & **Kernel** () const
Get the kernel.
- KernelType & **Kernel** ()
Modify the kernel.
- const MetricType & **Metric** () const
Get the metric.
- MetricType & **Metric** ()
Modify the metric.
- **KDEMode** **Mode** () const
*Get the mode of **KDE** (p. 1387).*
- **KDEMode** & **Mode** ()
*Modify the mode of **KDE** (p. 1387).*
- **KDE** & **operator=** (**KDE** other)
*Copy a **KDE** (p. 1387) model.*
- bool **OwnsReferenceTree** () const
*Check whether reference tree is owned by the **KDE** (p. 1387) model.*
- **Tree** * **ReferenceTree** ()
Get the reference tree.
- double **RelativeError** () const

Get relative error tolerance.

- void **RelativeError** (const double newError)
Modify relative error tolerance (0 <= newError <= 1).
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the model.
- void **Train** (MatType referenceSet)
Trains the KDE (p. 1387) model.
- void **Train** (**Tree** *referenceTree, std::vector< size_t > *oldFromNewReferences)
Trains the KDE (p. 1387) model.

39.298.1 Detailed Description

```
template<typename KernelType = kernel::GaussianKernel, typename MetricType = mlpack::metric::EuclideanDistance, typename
MatType = arma::mat, template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType = tree<
::KDTree, template< typename RuleType > class DualTreeTraversalType = TreeType<MetricType, kde::KDEStat, MatType>::template
DualTreeTraverser, template< typename RuleType > class SingleTreeTraversalType = TreeType<MetricType, kde::KDEStat, Mat<
Type>::template SingleTreeTraverser>
class mlpack::kde::KDE< KernelType, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTreeTraversalType >
```

The **KDE** (p. 1387) class is a template class for performing Kernel Density Estimations.

In statistics, kernel density estimation is a way to estimate the probability density function of a variable in a non parametric way. This implementation performs this estimation using a tree-independent dual-tree algorithm. Details about this algorithm are available in **KDERules** (p. 1404).

Template Parameters

<i>KernelType</i>	Kernel function to use for KDE (p. 1387) calculations.
<i>MetricType</i>	Metric to use for KDE (p. 1387) calculations.
<i>MatType</i>	Type of data to use.
<i>TreeType</i>	Type of tree to use; must satisfy the TreeType policy API.
<i>DualTreeTraversalType</i>	Type of dual-tree traversal to use.
<i>SingleTreeTraversalType</i>	Type of single-tree traversal to use.

Definition at line 59 of file kde.hpp.

39.298.2 Member Typedef Documentation

39.298.2.1 Tree

```
typedef TreeType<MetricType, kde::KDEStat, MatType> Tree
```

Convenience typedef.

Definition at line 63 of file kde.hpp.

39.298.3 Constructor & Destructor Documentation

39.298.3.1 KDE() [1/3]

```
KDE (
    const double relError = 0.05,
    const double absError = 0,
    KernelType kernel = KernelType(),
    const KDEMode mode = DUAL_TREE_MODE,
    MetricType metric = MetricType() )
```

Initialize **KDE** (p. 1387) object using custom instantiated Metric and Kernel objects.

Parameters

<i>relError</i>	Relative error tolerance of the model.
<i>absError</i>	Absolute error tolerance of the model.
<i>kernel</i>	Instantiated kernel object.
<i>mode</i>	Mode for the algorithm.
<i>metric</i>	Instantiated metric object.

39.298.3.2 KDE() [2/3]

```
KDE (
    const KDE< KernelType, MetricType, MatType, TreeType, DualTreeTraversalType, Single↵
TreeTraversalType > & other )
```

Construct **KDE** (p. 1387) object as a copy of the given model.

This may be computationally intensive!

Parameters

<i>other</i>	KDE (p. 1387) object to copy.
--------------	--------------------------------------

39.298.3.3 KDE() [3/3]

```
KDE (
    KDE< KernelType, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTree↵
TraversalType > && other )
```

Construct **KDE** (p. 1387) object taking ownership of the given model.

Parameters

<i>other</i>	KDE (p. 1387) object to take ownership of.
--------------	---

39.298.3.4 ~KDE()

`~ KDE ()`

Destroy the **KDE** (p. 1387) object.

If this object created any trees, they will be deleted. If you created the trees then you have to delete them yourself.

39.298.4 Member Function Documentation

39.298.4.1 AbsoluteError() [1/2]

```
double AbsoluteError ( ) const [inline]
```

Get absolute error tolerance.

Definition at line 200 of file kde.hpp.

39.298.4.2 AbsoluteError() [2/2]

```
void AbsoluteError (
    const double newError )
```

Modify absolute error tolerance (0 <= newError).

39.298.4.3 Evaluate() [1/3]

```
void Evaluate (
    MatType querySet,
    arma::vec & estimations )
```

Estimate density of each point in the query set given the data of the reference set.

The result is stored in an estimations vector. Estimations might not be normalized.

- Dimension of each point in the query set must match the dimension of each point in the reference set.
- Use `std::move` if the query set is no longer needed.

Precondition

The model has to be previously trained.

Parameters

<i>querySet</i>	Set of query points to get the density of.
<i>estimations</i>	Object which will hold the density of each query point.

39.298.4.4 Evaluate() [2/3]

```
void Evaluate (
    Tree * queryTree,
    const std::vector< size_t > & oldFromNewQueries,
    arma::vec & estimations )
```

Estimate density of each point in the query set given the data of an already created query tree.

The result is stored in an estimations vector. Estimations might not be normalized.

- Dimension of each point in the queryTree dataset must match the dimension of each point in the reference set.
- Use std::move if the query tree is no longer needed.

Precondition

The model has to be previously trained and mode has to be dual-tree.

Parameters

<i>queryTree</i>	Tree of query points to get the density of.
<i>oldFromNewQueries</i>	Mappings of query points to the tree dataset.
<i>estimations</i>	Object which will hold the density of each query point.

39.298.4.5 Evaluate() [3/3]

```
void Evaluate (
    arma::vec & estimations )
```

Estimate density of each point in the reference set given the data of the reference set.

It does not compute the estimation of a point with itself. The result is stored in an estimations vector. Estimations might not be normalized.

Precondition

The model has to be previously trained.

Parameters

<i>estimations</i>	Object which will hold the density of each reference point.
--------------------	---

39.298.4.6 IsTrained()

```
bool IsTrained ( ) const [inline]
```

Check whether **KDE** (p. 1387) model is trained or not.

Definition at line 209 of file kde.hpp.

39.298.4.7 Kernel() [1/2]

```
const KernelType& Kernel ( ) const [inline]
```

Get the kernel.

Definition at line 179 of file kde.hpp.

39.298.4.8 Kernel() [2/2]

```
KernelType& Kernel ( ) [inline]
```

Modify the kernel.

Definition at line 182 of file kde.hpp.

39.298.4.9 Metric() [1/2]

```
const MetricType& Metric ( ) const [inline]
```

Get the metric.

Definition at line 185 of file kde.hpp.

39.298.4.10 Metric() [2/2]

```
MetricType& Metric ( ) [inline]
```

Modify the metric.

Definition at line 188 of file kde.hpp.

39.298.4.11 Mode() [1/2]

```
KDEMode Mode ( ) const [inline]
```

Get the mode of **KDE** (p. 1387).

Definition at line 212 of file kde.hpp.

39.298.4.12 Mode() [2/2]

```
KDEMode& Mode ( ) [inline]
```

Modify the mode of **KDE** (p. 1387).

Definition at line 215 of file kde.hpp.

References KDE< KernelType, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTreeTraversalType >↵
::serialize().

39.298.4.13 operator=()

```
KDE& operator= (
    KDE< KernelType, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTree↵
    TraversalType > other )
```

Copy a **KDE** (p. 1387) model.

Use std::move if the object to copy is no longer needed.

Parameters

<i>other</i>	KDE (p. 1387) model to copy.
--------------	-------------------------------------

39.298.4.14 OwnsReferenceTree()

```
bool OwnsReferenceTree ( ) const [inline]
```

Check whether reference tree is owned by the **KDE** (p. 1387) model.

Definition at line 206 of file kde.hpp.

39.298.4.15 ReferenceTree()

```
Tree* ReferenceTree ( ) [inline]
```

Get the reference tree.

Definition at line 191 of file kde.hpp.

39.298.4.16 RelativeError() [1/2]

```
double RelativeError ( ) const [inline]
```

Get relative error tolerance.

Definition at line 194 of file kde.hpp.

39.298.4.17 RelativeError() [2/2]

```
void RelativeError (
    const double newError )
```

Modify relative error tolerance ($0 \leq \text{newError} \leq 1$).

39.298.4.18 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the model.

Referenced by KDE< KernelType, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTreeTraversalType >::Mode().

39.298.4.19 Train() [1/2]

```
void Train (
    MatType referenceSet )
```

Trains the **KDE** (p. 1387) model.

It builds a tree using a reference set.

Use std::move if the reference set is no longer needed.

Parameters

<i>referenceSet</i>	Set of reference data.
---------------------	------------------------

39.298.4.20 Train() [2/2]

```
void Train (
    Tree * referenceTree,
    std::vector< size_t > * oldFromNewReferences )
```

Trains the **KDE** (p. 1387) model.

Sets the reference tree to an already created tree.

- If TreeTraits<TreeType>::RearrangesDataset is false then it is possible to use an empty oldFromNewReferences vector.

Parameters

<i>referenceTree</i>	Built reference tree.
<i>oldFromNewReferences</i>	Permutations of reference points obtained during tree generation.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/ **kde.hpp**

39.299 KDEModel Class Reference

Public Types

- enum **KernelTypes** {
GAUSSIAN_KERNEL,
EPANECHNIKOV_KERNEL,
LAPLACIAN_KERNEL,
SPHERICAL_KERNEL,
TRIANGULAR_KERNEL }
- enum **TreeTypes** {
KD_TREE,
BALL_TREE,
COVER_TREE,
OCTREE,
R_TREE }

Public Member Functions

- **KDEModel** (const double bandwidth=1.0, const double relError=0.05, const double absError=0, const **KernelTypes** kernelType=KernelTypes::GAUSSIAN_KERNEL, const **TreeTypes** treeType=TreeTypes::KD_TREE)
Initialize **KDEModel** (p. 1397).
- **KDEModel** (const **KDEModel** &other)
Copy constructor of the given model.
- **KDEModel** (**KDEModel** &&other)
Move constructor of the given model. Takes ownership of the model.
- **~KDEModel** ()
Destroy the **KDEModel** (p. 1397) object.
- double **AbsoluteError** () const
Get the absolute error tolerance.
- double & **AbsoluteError** ()
Modify the absolute error tolerance.
- double **Bandwidth** () const
Get the bandwidth of the kernel.
- double & **Bandwidth** ()
Modify the bandwidth of the kernel.
- void **BuildModel** (arma::mat &&referenceSet)
Build the **KDE** (p. 1387) model with the given parameters and then trains it with the given reference data.
- void **Evaluate** (arma::mat &&querySet, arma::vec &estimations)
Perform kernel density estimation on the given query set.
- void **Evaluate** (arma::vec &estimations)
Perform kernel density estimation on the reference set.

- **KernelTypes** **KernelType** () const
Get the kernel type of the model.
- **KernelTypes** & **KernelType** ()
Modify the kernel type of the model.
- **KDEMode** **Mode** () const
Get the mode of the model.
- **KDEMode** & **Mode** ()
Modify the mode of the model.
- **KDEModel** & **operator=** (**KDEModel** other)
Copy the given model.
- double **RelativeError** () const
Get the relative error tolerance.
- double & **RelativeError** ()
Modify the relative error tolerance.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
*Serialize the **KDE** (p. 1387) model.*
- **TreeTypes** **TreeType** () const
Get the tree type of the model.
- **TreeTypes** & **TreeType** ()
Modify the tree type of the model.

39.299.1 Detailed Description

Definition at line 193 of file kde_model.hpp.

39.299.2 Member Enumeration Documentation

39.299.2.1 KernelTypes

enum **KernelTypes**

Enumerator

GAUSSIAN_KERNEL	
EPANECHNIKOV_KERNEL	
LAPLACIAN_KERNEL	
SPHERICAL_KERNEL	
TRIANGULAR_KERNEL	

Definition at line 205 of file kde_model.hpp.

39.299.2.2 TreeTypes

```
enum TreeTypes
```

Enumerator

KD_TREE	
BALL_TREE	
COVER_TREE	
OCTREE	
R_TREE	

Definition at line 196 of file kde_model.hpp.

39.299.3 Constructor & Destructor Documentation

39.299.3.1 KDEModel() [1/3]

```
KDEModel (
    const double bandwidth = 1.0,
    const double relError = 0.05,
    const double absError = 0,
    const KernelTypes kernelType = KernelTypes::GAUSSIAN_KERNEL,
    const TreeTypes treeType = TreeTypes::KD_TREE )
```

Initialize **KDEModel** (p. 1397).

Parameters

<i>bandwidth</i>	Bandwidth to use for the kernel.
<i>relError</i>	Maximum relative error tolerance for each point in the model. For example, 0.05 means that each value must be within 5% of the true KDE (p. 1387) value.
<i>absError</i>	Maximum absolute error tolerance for each point in the model. For example, 0.1 means that for each point the value can have a maximum error of 0.1 units.
<i>kernelType</i>	Type of kernel to use.
<i>treeType</i>	Type of tree to use.

39.299.3.2 KDEMModel() [2/3]

```
KDEMModel (
    const KDEMModel & other )
```

Copy constructor of the given model.

39.299.3.3 KDEMModel() [3/3]

```
KDEMModel (
    KDEMModel && other )
```

Move constructor of the given model. Takes ownership of the model.

39.299.3.4 ~KDEMModel()

```
~ KDEMModel ( )
```

Destroy the **KDEMModel** (p. 1397) object.

39.299.4 Member Function Documentation**39.299.4.1 AbsoluteError()** [1/2]

```
double AbsoluteError ( ) const [inline]
```

Get the absolute error tolerance.

Definition at line 315 of file kde_model.hpp.

39.299.4.2 AbsoluteError() [2/2]

```
double& AbsoluteError ( ) [inline]
```

Modify the absolute error tolerance.

Definition at line 318 of file kde_model.hpp.

39.299.4.3 Bandwidth() [1/2]

```
double Bandwidth ( ) const [inline]
```

Get the bandwidth of the kernel.

Definition at line 303 of file kde_model.hpp.

39.299.4.4 Bandwidth() [2/2]

```
double& Bandwidth ( ) [inline]
```

Modify the bandwidth of the kernel.

Definition at line 306 of file kde_model.hpp.

39.299.4.5 BuildModel()

```
void BuildModel (
    arma::mat && referenceSet )
```

Build the **KDE** (p. 1387) model with the given parameters and then trains it with the given reference data.

Takes possession of the reference set to avoid a copy, so the reference set will not be usable after this.

Parameters

<i>referenceSet</i>	Set of reference points.
---------------------	--------------------------

39.299.4.6 Evaluate() [1/2]

```
void Evaluate (
    arma::mat && querySet,
    arma::vec & estimations )
```

Perform kernel density estimation on the given query set.

Takes possession of the query set to avoid a copy, so the query set will not be usable after this. If possible, it returns normalized estimations.

Precondition

The model has to be previously created with BuildModel.

Parameters

<i>querySet</i>	Set of query points.
<i>estimations</i>	Vector where the results will be stored in the same order as the query points.

39.299.4.7 Evaluate() [2/2]

```
void Evaluate (
    arma::vec & estimations )
```

Perform kernel density estimation on the reference set.

If possible, it returns normalized estimations.

Precondition

The model has to be previously created with BuildModel.

Parameters

<i>estimations</i>	Vector where the results will be stored in the same order as the query points.
--------------------	--

39.299.4.8 KernelType() [1/2]

```
KernelTypes KernelType ( ) const [inline]
```

Get the kernel type of the model.

Definition at line 327 of file kde_model.hpp.

39.299.4.9 KernelType() [2/2]

```
KernelTypes& KernelType ( ) [inline]
```

Modify the kernel type of the model.

Definition at line 330 of file kde_model.hpp.

References mlpack::bindings::tests::CleanMemory().

39.299.4.10 Mode() [1/2]

```
KDEMode Mode ( ) const
```

Get the mode of the model.

39.299.4.11 Mode() [2/2]

```
KDEMode& Mode ( )
```

Modify the mode of the model.

39.299.4.12 operator=()

```
KDEModel& operator= (
    KDEModel other )
```

Copy the given model.

Use std::move if the object to copy is no longer needed.

Parameters

<i>other</i>	KDEModel (p. 1397) to copy.
--------------	------------------------------------

39.299.4.13 RelativeError() [1/2]

```
double RelativeError ( ) const [inline]
```

Get the relative error tolerance.

Definition at line 309 of file kde_model.hpp.

39.299.4.14 RelativeError() [2/2]

```
double& RelativeError ( ) [inline]
```

Modify the relative error tolerance.

Definition at line 312 of file kde_model.hpp.

39.299.4.15 `serialize()`

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the **KDE** (p. 1387) model.

39.299.4.16 `TreeType()` [1/2]

```
TreeTypes TreeType ( ) const [inline]
```

Get the tree type of the model.

Definition at line 321 of file `kde_model.hpp`.

39.299.4.17 `TreeType()` [2/2]

```
TreeTypes& TreeType ( ) [inline]
```

Modify the tree type of the model.

Definition at line 324 of file `kde_model.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/ kde_model.hpp`

39.300 `KDERules< MetricType, KernelType, TreeType >` Class Template Reference

A dual-tree traversal Rules class for kernel density estimation.

Public Types

- typedef `tree::TraversallInfo< TreeType >` **TraversallInfoType**

Public Member Functions

- **KDERules** (const arma::mat &referenceSet, const arma::mat &querySet, arma::vec &densities, const double relError, const double absError, MetricType &metric, KernelType &kernel, const bool sameSet)
*Construct **KDERules** (p. 1404).*
- double **BaseCase** (const size_t queryIndex, const size_t referenceIndex)
Base Case.
- size_t **BaseCases** () const
Get the number of base cases.
- double **Rescore** (const size_t queryIndex, TreeType &referenceNode, const double oldScore) const
SingleTree Score.
- double **Rescore** (TreeType &queryNode, TreeType &referenceNode, const double oldScore) const
DoubleTree Rescore.
- double **Score** (const size_t queryIndex, TreeType &referenceNode)
SingleTree Rescore.
- double **Score** (TreeType &queryNode, TreeType &referenceNode)
DoubleTree Score.
- size_t **Scores** () const
Get the number of scores.
- const **TraversalInfoType** & **TraversalInfo** () const
Get traversal information.
- **TraversalInfoType** & **TraversalInfo** ()
Modify traversal information.

39.300.1 Detailed Description

```
template<typename MetricType, typename KernelType, typename TreeType>
class mlpack::kde::KDERules< MetricType, KernelType, TreeType >
```

A dual-tree traversal Rules class for kernel density estimation.

This contains the **Score()** (p. 1407) and **BaseCase()** (p. 1406) implementations.

Definition at line 26 of file kde_rules.hpp.

39.300.2 Member Typedef Documentation

39.300.2.1 TraversalInfoType

```
typedef tree::TraversalInfo<TreeType> TraversalInfoType
```

Definition at line 70 of file kde_rules.hpp.

39.300.3 Constructor & Destructor Documentation

39.300.3.1 KDERules()

```
KDERules (
    const arma::mat & referenceSet,
    const arma::mat & querySet,
    arma::vec & densities,
    const double relError,
    const double absError,
    MetricType & metric,
    KernelType & kernel,
    const bool sameSet )
```

Construct **KDERules** (p. 1404).

Parameters

<i>referenceSet</i>	Reference set data.
<i>querySet</i>	Query set data.
<i>densities</i>	Vector where estimations will be written.
<i>relError</i>	Relative error tolerance.
<i>absError</i>	Absolute error tolerance.
<i>metric</i>	Instantiated metric.
<i>kernel</i>	Instantiated kernel.
<i>sameSet</i>	True if query and reference sets are the same (monochromatic evaluation).

39.300.4 Member Function Documentation

39.300.4.1 BaseCase()

```
double BaseCase (
    const size_t queryIndex,
    const size_t referenceIndex )
```

Base Case.

39.300.4.2 BaseCases()

```
size_t BaseCases ( ) const [inline]
```

Get the number of base cases.

Definition at line 79 of file kde_rules.hpp.

39.300.4.3 Rescore() [1/2]

```
double Rescore (
    const size_t queryIndex,
    TreeType & referenceNode,
    const double oldScore ) const
```

SingleTree Score.

39.300.4.4 Rescore() [2/2]

```
double Rescore (
    TreeType & queryNode,
    TreeType & referenceNode,
    const double oldScore ) const
```

DoubleTree Rescore.

39.300.4.5 Score() [1/2]

```
double Score (
    const size_t queryIndex,
    TreeType & referenceNode )
```

SingleTree Rescore.

39.300.4.6 Score() [2/2]

```
double Score (
    TreeType & queryNode,
    TreeType & referenceNode )
```

DoubleTree Score.

39.300.4.7 Scores()

```
size_t Scores ( ) const [inline]
```

Get the number of scores.

Definition at line 82 of file kde_rules.hpp.

39.300.4.8 TraversalInfo() [1/2]

```
const TraversalInfoType& TraversalInfo ( ) const [inline]
```

Get traversal information.

Definition at line 73 of file kde_rules.hpp.

39.300.4.9 TraversalInfo() [2/2]

```
TraversalInfoType& TraversalInfo ( ) [inline]
```

Modify traversal information.

Definition at line 76 of file kde_rules.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/ **kde_rules.hpp**

39.301 KDEStat Class Reference

Extra data for each node in the tree for the task of kernel density estimation.

Public Member Functions

- **KDEStat** ()
Initialize the statistic.
- template<typename TreeType >
KDEStat (TreeType &node)
Initialization for a fully initialized node.
- const arma::vec & **Centroid** () const
Get the centroid of the node.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the statistic to/from an archive.
- void **SetCentroid** (arma::vec newCentroid)
Modify the centroid of the node.
- bool **ValidCentroid** () const
Get whether the centroid is valid.

39.301.1 Detailed Description

Extra data for each node in the tree for the task of kernel density estimation.

Definition at line 24 of file kde_stat.hpp.

39.301.2 Constructor & Destructor Documentation

39.301.2.1 KDEStat() [1/2]

```
KDEStat ( ) [inline]
```

Initialize the statistic.

Definition at line 28 of file kde_stat.hpp.

39.301.2.2 KDEStat() [2/2]

```
KDEStat (
    TreeType & node ) [inline]
```

Initialization for a fully initialized node.

Definition at line 32 of file kde_stat.hpp.

39.301.3 Member Function Documentation

39.301.3.1 Centroid()

```
const arma::vec& Centroid ( ) const [inline]
```

Get the centroid of the node.

Definition at line 47 of file kde_stat.hpp.

39.301.3.2 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the statistic to/from an archive.

Definition at line 67 of file kde_stat.hpp.

39.301.3.3 SetCentroid()

```
void SetCentroid (
    arma::vec newCentroid ) [inline]
```

Modify the centroid of the node.

Definition at line 56 of file kde_stat.hpp.

39.301.3.4 ValidCentroid()

```
bool ValidCentroid ( ) const [inline]
```

Get whether the centroid is valid.

Definition at line 63 of file kde_stat.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/ **kde_stat.hpp**

39.302 KernelNormalizer Class Reference

KernelNormalizer (p. 1410) holds a set of methods to normalize estimations applying in each case the appropriate kernel normalizer function.

Static Public Member Functions

- `template<typename KernelType >`
 static void **ApplyNormalizer** (KernelType &, const size_t, arma::vec &, const typename std::enable_if< !HasNormalizer< KernelType, double(KernelType::*)(size_t)>::value >::type !=0)
Normalization not needed.
- `template<typename KernelType >`
 static void **ApplyNormalizer** (KernelType &kernel, const size_t dimension, arma::vec &estimations, const typename std::enable_if< HasNormalizer< KernelType, double(KernelType::*)(size_t)>::value >::type !=0)
Normalize kernels that have normalizer.

39.302.1 Detailed Description

KernelNormalizer (p. 1410) holds a set of methods to normalize estimations applying in each case the appropriate kernel normalizer function.

Definition at line 51 of file `kde_model.hpp`.

39.302.2 Member Function Documentation

39.302.2.1 ApplyNormalizer() [1/2]

```
static void ApplyNormalizer (
    KernelType & ,
    const size_t ,
    arma::vec & ,
    const typename std::enable_if< !HasNormalizer< KernelType, double(KernelType::*) (size_t)>::value >::type * = 0 ) [inline], [static]
```

Normalization not needed.

Definition at line 60 of file `kde_model.hpp`.

39.302.2.2 ApplyNormalizer() [2/2]

```
static void ApplyNormalizer (
    KernelType & kernel,
    const size_t dimension,
    arma::vec & estimations,
    const typename std::enable_if< HasNormalizer< KernelType, double(KernelType::*) (size_t)>::value >::type * = 0 ) [inline], [static]
```

Normalize kernels that have normalizer.

Definition at line 71 of file `kde_model.hpp`.

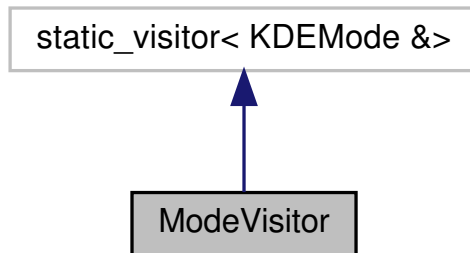
The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/ kde_model.hpp`

39.303 ModeVisitor Class Reference

ModeVisitor (p. 1412) exposes the Mode() method of the KDEType.

Inheritance diagram for ModeVisitor:



Public Member Functions

- template<typename KDEType >
KDEMode & operator() (KDEType *kde) const
Return mode of KDEType instance.

39.303.1 Detailed Description

ModeVisitor (p. 1412) exposes the Mode() method of the KDEType.

Definition at line 177 of file kde_model.hpp.

39.303.2 Member Function Documentation

39.303.2.1 operator()()

```

KDEMode& operator() (
    KDEType * kde ) const
  
```

Return mode of KDEType instance.

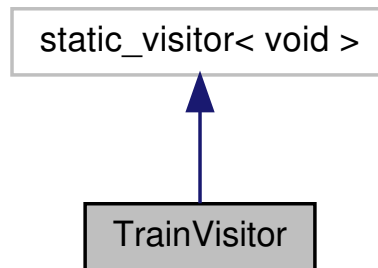
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/ **kde_model.hpp**

39.304 TrainVisitor Class Reference

TrainVisitor (p. 1413) trains a given KDEType using a reference set.

Inheritance diagram for TrainVisitor:



Public Member Functions

- **TrainVisitor** (arma::mat &&referenceSet)
***TrainVisitor** (p. 1413) constructor. Takes ownership of the given referenceSet.*
- template<typename KernelType , template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType>
void **operator()** (KDEType< KernelType, TreeType > *kde) const
*Default **TrainVisitor** (p. 1413) on some KDEType.*

39.304.1 Detailed Description

TrainVisitor (p. 1413) trains a given KDEType using a reference set.

Definition at line 154 of file kde_model.hpp.

39.304.2 Constructor & Destructor Documentation

39.304.2.1 TrainVisitor()

```

TrainVisitor (
    arma::mat && referenceSet )
  
```

TrainVisitor (p. 1413) constructor. Takes ownership of the given referenceSet.

39.304.3 Member Function Documentation

39.304.3.1 operator>()

```
void operator() (
    KDEType< KernelType, TreeType > * kde ) const
```

Default **TrainVisitor** (p. 1413) on some KDEType.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/ **kde_model.hpp**

39.305 CauchyKernel Class Reference

The Cauchy kernel.

Public Member Functions

- **CauchyKernel** (double bandwidth=1.0)
Construct the Cauchy kernel; by default, the bandwidth is 1.0.
- template<typename VecTypeA , typename VecTypeB >
double **Evaluate** (const VecTypeA &a, const VecTypeB &b)
Evaluation of the Cauchy kernel.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the kernel.

39.305.1 Detailed Description

The Cauchy kernel.

Given two vector x , y , and a bandwidth σ (set in the constructor),

$$K(x, y) = \frac{1}{1 + \left(\frac{\|x - y\|}{\sigma}\right)^2}.$$

For more details, see the following published paper:

```
@inproceedings{Basak2008,
  title={A least square kernel machine with box constraints},
  author={Basak, Jayanta},
  booktitle={Pattern Recognition, 2008. ICPR 2008. 19th International
    Conference on},
  pages={1--4},
  year={2008},
  organization={IEEE}
}
```

Definition at line 44 of file `cauchy_kernel.hpp`.

39.305.2 Constructor & Destructor Documentation

39.305.2.1 CauchyKernel()

```
CauchyKernel (
    double bandwidth = 1.0 ) [inline]
```

Construct the Cauchy kernel; by default, the bandwidth is 1.0.

Definition at line 50 of file `cauchy_kernel.hpp`.

39.305.3 Member Function Documentation

39.305.3.1 Evaluate()

```
double Evaluate (
    const VecTypeA & a,
    const VecTypeB & b ) [inline]
```

Evaluation of the Cauchy kernel.

This could be generalized to use any distance metric, not the Euclidean distance, but for now, the Euclidean distance is used.

Template Parameters

<i>VecTypeA</i>	Type of first vector (arma::vec, arma::sp_vec).
<i>VecTypeB</i>	Type of second vector (arma::vec, arma::sp_vec).

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

$K(a, b)$.

Definition at line 65 of file `cauchy_kernel.hpp`.

References `LMetric< TPower, TTakeRoot >::Evaluate()`.

39.305.3.2 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the kernel.

Definition at line 75 of file `cauchy_kernel.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ cauchy_kernel.hpp`

39.306 CosineDistance Class Reference

The cosine distance (or cosine similarity).

Public Member Functions

- `template<typename Archive >`
`void serialize (Archive &, const unsigned int)`
Serialize the class (there's nothing to save).

Static Public Member Functions

- `template<typename VecTypeA , typename VecTypeB >`
`static double Evaluate (const VecTypeA &a, const VecTypeB &b)`
Computes the cosine distance between two points.

39.306.1 Detailed Description

The cosine distance (or cosine similarity).

It is defined by

$$d(a,b) = \frac{a^T b}{||a|| ||b||}$$

and this class assumes the standard L2 inner product.

Definition at line 31 of file `cosine_distance.hpp`.

39.306.2 Member Function Documentation

39.306.2.1 Evaluate()

```
static double Evaluate (
    const VecTypeA & a,
    const VecTypeB & b ) [static]
```

Computes the cosine distance between two points.

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

$d(a, b)$.

39.306.2.2 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serialize the class (there's nothing to save).

Definition at line 46 of file cosine_distance.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ **cosine_distance.hpp**

39.307 EpanechnikovKernel Class Reference

The Epanechnikov kernel, defined as.

Public Member Functions

- **EpanechnikovKernel** (const double bandwidth=1.0)
Instantiate the Epanechnikov kernel with the given bandwidth (default 1.0).
- template<typename VecTypeA , typename VecTypeB >
double **ConvolutionIntegral** (const VecTypeA &a, const VecTypeB &b)
Obtains the convolution integral [integral of $K(\|x-a\|) K(\|b-x\|) dx$] for the two vectors.
- template<typename VecTypeA , typename VecTypeB >
double **Evaluate** (const VecTypeA &a, const VecTypeB &b) const
Evaluate the Epanechnikov kernel on the given two inputs.
- double **Evaluate** (const double distance) const
Evaluate the Epanechnikov kernel given that the distance between the two input points is known.
- double **Gradient** (const double distance) const
Evaluate the Gradient of Epanechnikov kernel given that the distance between the two input points is known.
- double **GradientForSquaredDistance** (const double distanceSquared) const
Evaluate the Gradient of Epanechnikov kernel given that the squared distance between the two input points is known.
- double **Normalizer** (const size_t dimension)
Compute the normalizer of this Epanechnikov kernel for the given dimension.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int version)
Serialize the kernel.

39.307.1 Detailed Description

The Epanechnikov kernel, defined as.

$$K(x, y) = \max\{0, 1 - \|x - y\|_2^2 / b^2\}$$

where b is the bandwidth the of the kernel (defaults to 1.0).

Definition at line 30 of file epanechnikov_kernel.hpp.

39.307.2 Constructor & Destructor Documentation

39.307.2.1 EpanechnikovKernel()

```
EpanechnikovKernel (
    const double bandwidth = 1.0 ) [inline]
```

Instantiate the Epanechnikov kernel with the given bandwidth (default 1.0).

Parameters

<i>bandwidth</i>	Bandwidth of the kernel.
------------------	--------------------------

Definition at line 38 of file epanechnikov_kernel.hpp.

References `EpanechnikovKernel::ConvolutionIntegral()`, `EpanechnikovKernel::Evaluate()`, `EpanechnikovKernel::↵ Gradient()`, `EpanechnikovKernel::GradientForSquaredDistance()`, `EpanechnikovKernel::Normalizer()`, and `Epanechnikov↵ Kernel::serialize()`.

39.307.3 Member Function Documentation

39.307.3.1 ConvolutionIntegral()

```
double ConvolutionIntegral (
    const VecTypeA & a,
    const VecTypeB & b )
```

Obtains the convolution integral [integral of $K(\|x-a\|) K(\|b-x\|) dx$] for the two vectors.

Template Parameters

<i>VecType</i>	Type of vector (arma::vec, arma::spvec should be expected).
----------------	---

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

the convolution integral value.

Referenced by EpanechnikovKernel::EpanechnikovKernel().

39.307.3.2 Evaluate() [1/2]

```
double Evaluate (
    const VecTypeA & a,
    const VecTypeB & b ) const
```

Evaluate the Epanechnikov kernel on the given two inputs.

Template Parameters

<i>VecTypeA</i>	Type of first vector.
<i>VecTypeB</i>	Type of second vector.

Parameters

<i>a</i>	One input vector.
<i>b</i>	The other input vector.

Referenced by EpanechnikovKernel::EpanechnikovKernel().

39.307.3.3 Evaluate() [2/2]

```
double Evaluate (
    const double distance ) const
```

Evaluate the Epanechnikov kernel given that the distance between the two input points is known.

39.307.3.4 Gradient()

```
double Gradient (
    const double distance ) const
```

Evaluate the Gradient of Epanechnikov kernel given that the distance between the two input points is known.

Referenced by EpanechnikovKernel::EpanechnikovKernel().

39.307.3.5 GradientForSquaredDistance()

```
double GradientForSquaredDistance (
    const double distanceSquared ) const
```

Evaluate the Gradient of Epanechnikov kernel given that the squared distance between the two input points is known.

Referenced by EpanechnikovKernel::EpanechnikovKernel().

39.307.3.6 Normalizer()

```
double Normalizer (
    const size_t dimension )
```

Compute the normalizer of this Epanechnikov kernel for the given dimension.

Parameters

<i>dimension</i>	Dimension to calculate the normalizer for.
------------------	--

Referenced by EpanechnikovKernel::EpanechnikovKernel().

39.307.3.7 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int version )
```

Serialize the kernel.

Referenced by EpanechnikovKernel::EpanechnikovKernel().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ **epanechnikov_kernel.hpp**

39.308 ExampleKernel Class Reference

An example kernel function.

Public Member Functions

- **ExampleKernel** ()
The default constructor, which takes no parameters.
- template<typename Archive >
void **serialize** (Archive &, const unsigned int)
Serializes the kernel.

Static Public Member Functions

- template<typename VecTypeA , typename VecTypeB >
static double **ConvolutionIntegral** (const VecTypeA &, const VecTypeB &)
Obtains the convolution integral $\int K(\|x-a\|)K(\|b-x\|)dx$ for the two vectors.
- template<typename VecTypeA , typename VecTypeB >
static double **Evaluate** (const VecTypeA &, const VecTypeB &)
Evaluates the kernel function for two given vectors.
- static double **Normalizer** ()
Obtains the normalizing volume for the kernel with dimension \$dimension\$.

39.308.1 Detailed Description

An example kernel function.

This is not a useful kernel, but it implements the two functions necessary to satisfy the Kernel policy (so that a class can be used whenever an mpack method calls for a `typename Kernel` template parameter).

All that is necessary is a constructor and an **Evaluate()** (p. 1423) function. More methods could be added; for instance, one useful idea is a constructor which takes parameters for a kernel (for instance, the width of the Gaussian for a Gaussian kernel). However, mpack methods cannot count on these various constructors existing, which is why most methods allow passing an already-instantiated kernel object (and by default the method will construct the kernel with the default constructor). So, for instance,

```
GaussianKernel k(5.0);
KernelPCA<GaussianKernel> kpca(dataset, k);
```

will set up kernel PCA using a Gaussian kernel with a width of 5.0, but

```
KernelPCA<GaussianKernel> kpca(dataset);
```

will create the kernel with the default constructor. It is important (but not strictly mandatory) that your default constructor still gives a working kernel.

Note

Not all kernels require state. For instance, the regular dot product needs no parameters. In that case, no local variables are necessary and **Evaluate()** (p. 1423) can (and should) be declared static. However, for greater generalization, mpack methods expect all kernels to require state and hence must store instantiated kernel functions; this is why a default constructor is necessary.

Definition at line 77 of file `example_kernel.hpp`.

39.308.2 Constructor & Destructor Documentation

39.308.2.1 ExampleKernel()

```
ExampleKernel ( ) [inline]
```

The default constructor, which takes no parameters.

Because our simple example kernel has no internal parameters that need to be stored, the constructor does not need to do anything. For a more complex example, see the **GaussianKernel** (p. 1424), which stores an internal parameter.

Definition at line 86 of file example_kernel.hpp.

39.308.3 Member Function Documentation

39.308.3.1 ConvolutionIntegral()

```
static double ConvolutionIntegral (
    const VecTypeA & ,
    const VecTypeB & ) [inline], [static]
```

Obtains the convolution integral $[\text{integral } K(\|x-a\|)K(\|b-x\|)dx]$ for the two vectors.

In this case, because our simple example kernel has no internal parameters, we can declare the function static. For a more complex example which cannot be declared static, see the **GaussianKernel** (p. 1424), which stores an internal parameter.

Template Parameters

<i>VecTypeA</i>	Type of first vector (arma::vec, arma::sp_vec should be expected).
<i>VecTypeB</i>	Type of second vector (arma::vec, arma::sp_vec).

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

the convolution integral value.

Definition at line 127 of file example_kernel.hpp.

39.308.3.2 Evaluate()

```
static double Evaluate (
    const VecTypeA & ,
    const VecTypeB & ) [inline], [static]
```

Evaluates the kernel function for two given vectors.

In this case, because our simple example kernel has no internal parameters, we can declare the function static. For a more complex example which cannot be declared static, see the **GaussianKernel** (p. 1424), which stores an internal parameter.

Template Parameters

<i>VecTypeA</i>	Type of first vector (arma::vec, arma::sp_vec should be expected).
<i>VecTypeB</i>	Type of second vector (arma::vec, arma::sp_vec).

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

$K(a, b)$.

Definition at line 102 of file example_kernel.hpp.

39.308.3.3 Normalizer()

```
static double Normalizer ( ) [inline], [static]
```

Obtains the normalizing volume for the kernel with dimension $\$dimension\$$.

In this case, because our simple example kernel has no internal parameters, we can declare the function static. For a more complex example which cannot be declared static, see the **GaussianKernel** (p. 1424), which stores an internal parameter.

Parameters

<i>dimension</i>	the dimension of the space.
------------------	-----------------------------

Returns

the normalization constant.

Definition at line 140 of file example_kernel.hpp.

39.308.3.4 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serializes the kernel.

In this case, the kernel has no members, so we do not need to do anything at all.

Definition at line 110 of file example_kernel.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ **example_kernel.hpp**

39.309 GaussianKernel Class Reference

The standard Gaussian kernel.

Public Member Functions

- **GaussianKernel** ()
Default constructor; sets bandwidth to 1.0.
- **GaussianKernel** (const double bandwidth)
Construct the Gaussian kernel with a custom bandwidth.
- double **Bandwidth** () const
Get the bandwidth.
- void **Bandwidth** (const double bandwidth)
Modify the bandwidth.
- template<typename VecTypeA , typename VecTypeB >
double **ConvolutionIntegral** (const VecTypeA &a, const VecTypeB &b)
Obtain a convolution integral of the Gaussian kernel.
- template<typename VecTypeA , typename VecTypeB >
double **Evaluate** (const VecTypeA &a, const VecTypeB &b) const
Evaluation of the Gaussian kernel.
- double **Evaluate** (const double t) const
Evaluation of the Gaussian kernel given the distance between two points.

- double **Gamma** () const
Get the precalculated constant.
- double **Gradient** (const double t) const
Evaluation of the gradient of Gaussian kernel given the distance between two points.
- double **GradientForSquaredDistance** (const double t) const
Evaluation of the gradient of Gaussian kernel given the squared distance between two points.
- double **Normalizer** (const size_t dimension)
Obtain the normalization constant of the Gaussian kernel.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the kernel.

39.309.1 Detailed Description

The standard Gaussian kernel.

Given two vectors x , y , and a bandwidth μ (set in the constructor),

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\mu^2}\right).$$

The implementation is all in the header file because it is so simple.

Definition at line 34 of file gaussian_kernel.hpp.

39.309.2 Constructor & Destructor Documentation

39.309.2.1 GaussianKernel() [1/2]

```
GaussianKernel ( ) [inline]
```

Default constructor; sets bandwidth to 1.0.

Definition at line 40 of file gaussian_kernel.hpp.

39.309.2.2 GaussianKernel() [2/2]

```
GaussianKernel (
    const double bandwidth ) [inline]
```

Construct the Gaussian kernel with a custom bandwidth.

Parameters

<i>bandwidth</i>	The bandwidth of the kernel (μ).
------------------	--

Definition at line 48 of file gaussian_kernel.hpp.

39.309.3 Member Function Documentation**39.309.3.1 Bandwidth()** [1/2]

```
double Bandwidth ( ) const [inline]
```

Get the bandwidth.

Definition at line 135 of file gaussian_kernel.hpp.

39.309.3.2 Bandwidth() [2/2]

```
void Bandwidth (
    const double bandwidth ) [inline]
```

Modify the bandwidth.

This takes an argument because we must update the precalculated constant (gamma).

Definition at line 139 of file gaussian_kernel.hpp.

39.309.3.3 ConvolutionIntegral()

```
double ConvolutionIntegral (
    const VecTypeA & a,
    const VecTypeB & b ) [inline]
```

Obtain a convolution integral of the Gaussian kernel.

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

The convolution integral.

Definition at line 127 of file gaussian_kernel.hpp.

References GaussianKernel::Evaluate(), LMetric< TPower, TTakeRoot >::Evaluate(), and GaussianKernel::←Normalizer().

39.309.3.4 Evaluate() [1/2]

```
double Evaluate (
    const VecTypeA & a,
    const VecTypeB & b ) const [inline]
```

Evaluation of the Gaussian kernel.

This could be generalized to use any distance metric, not the Euclidean distance, but for now, the Euclidean distance is used.

Template Parameters

<i>VecType</i>	Type of vector (likely arma::vec or arma::spvec).
----------------	---

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

K(a, b) using the bandwidth (μ) specified in the constructor.

Definition at line 65 of file gaussian_kernel.hpp.

References LMetric< TPower, TTakeRoot >::Evaluate().

Referenced by GaussianKernel::ConvolutionIntegral().

39.309.3.5 Evaluate() [2/2]

```
double Evaluate (
    const double t ) const [inline]
```

Evaluation of the Gaussian kernel given the distance between two points.

Parameters

t	The distance between the two points the kernel is evaluated on.
-----	---

Returns

K(t) using the bandwidth (μ) specified in the constructor.

Definition at line 78 of file gaussian_kernel.hpp.

39.309.3.6 Gamma()

```
double Gamma ( ) const [inline]
```

Get the precalculated constant.

Definition at line 146 of file gaussian_kernel.hpp.

39.309.3.7 Gradient()

```
double Gradient (
    const double t ) const [inline]
```

Evaluation of the gradient of Gaussian kernel given the distance between two points.

Parameters

t	The distance between the two points the kernel is evaluated on.
-----	---

Returns

K(t) using the bandwidth (μ) specified in the constructor.

Definition at line 92 of file gaussian_kernel.hpp.

39.309.3.8 GradientForSquaredDistance()

```
double GradientForSquaredDistance (
    const double t ) const [inline]
```

Evaluation of the gradient of Gaussian kernel given the squared distance between two points.

Parameters

<i>t</i>	The squared distance between the two points
----------	---

Returns

$K(t)$ using the bandwidth (μ) specified in the constructor.

Definition at line 104 of file gaussian_kernel.hpp.

39.309.3.9 Normalizer()

```
double Normalizer (
    const size_t dimension ) [inline]
```

Obtain the normalization constant of the Gaussian kernel.

Parameters

<i>dimension</i>	
------------------	--

Returns

the normalization constant

Definition at line 114 of file gaussian_kernel.hpp.

References `M_PI`.

Referenced by `GaussianKernel::ConvolutionIntegral()`.

39.309.3.10 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the kernel.

Definition at line 150 of file gaussian_kernel.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ **gaussian_kernel.hpp**

39.310 HyperbolicTangentKernel Class Reference

Hyperbolic tangent kernel.

Public Member Functions

- **HyperbolicTangentKernel** ()
This constructor sets the default scale to 1.0 and offset to 0.0.
- **HyperbolicTangentKernel** (double scale, double offset)
Construct the hyperbolic tangent kernel with custom scale factor and offset.
- template<typename VecTypeA , typename VecTypeB >
double **Evaluate** (const VecTypeA &a, const VecTypeB &b)
Evaluate the hyperbolic tangent kernel.
- double **Offset** () const
Get offset for the kernel.
- double & **Offset** ()
Modify offset for the kernel.
- double **Scale** () const
Get scale factor.
- double & **Scale** ()
Modify scale factor.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the kernel.

39.310.1 Detailed Description

Hyperbolic tangent kernel.

For any two vectors x, y and a given scale s and offset t

$$K(x, y) = \tanh(s \langle x, y \rangle + t)$$

Definition at line 28 of file hyperbolic_tangent_kernel.hpp.

39.310.2 Constructor & Destructor Documentation

39.310.2.1 HyperbolicTangentKernel() [1/2]

```
HyperbolicTangentKernel ( ) [inline]
```

This constructor sets the default scale to 1.0 and offset to 0.0.

Definition at line 34 of file hyperbolic_tangent_kernel.hpp.

39.310.2.2 HyperbolicTangentKernel() [2/2]

```
HyperbolicTangentKernel (
    double scale,
    double offset ) [inline]
```

Construct the hyperbolic tangent kernel with custom scale factor and offset.

Parameters

<i>scale</i>	Scaling factor for $\langle x, y \rangle$.
<i>offset</i>	Kernel offset.

Definition at line 44 of file hyperbolic_tangent_kernel.hpp.

39.310.3 Member Function Documentation

39.310.3.1 Evaluate()

```
double Evaluate (
    const VecTypeA & a,
    const VecTypeB & b ) [inline]
```

Evaluate the hyperbolic tangent kernel.

This evaluation uses Armadillo's dot() function.

Template Parameters

<i>VecTypeA</i>	Type of first vector (should be arma::vec or arma::sp_vec).
<i>VecTypeB</i>	Type of second vector (arma::vec / arma::sp_vec).

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

$K(a, b)$.

Definition at line 60 of file hyperbolic_tangent_kernel.hpp.

39.310.3.2 Offset() [1/2]

```
double Offset ( ) const [inline]
```

Get offset for the kernel.

Definition at line 71 of file hyperbolic_tangent_kernel.hpp.

39.310.3.3 Offset() [2/2]

```
double& Offset ( ) [inline]
```

Modify offset for the kernel.

Definition at line 73 of file hyperbolic_tangent_kernel.hpp.

39.310.3.4 Scale() [1/2]

```
double Scale ( ) const [inline]
```

Get scale factor.

Definition at line 66 of file hyperbolic_tangent_kernel.hpp.

39.310.3.5 Scale() [2/2]

```
double& Scale ( ) [inline]
```

Modify scale factor.

Definition at line 68 of file hyperbolic_tangent_kernel.hpp.

39.310.3.6 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the kernel.

Definition at line 77 of file hyperbolic_tangent_kernel.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ **hyperbolic_tangent_kernel.hpp**

39.311 KernelTraits< KernelType > Class Template Reference

This is a template class that can provide information about various kernels.

Static Public Attributes

- static const bool **IsNormalized** = false
If true, then the kernel is normalized: $K(x, x) = K(y, y) = 1$ for all x .
- static const bool **UsesSquaredDistance** = false
If true, then the kernel include a squared distance, $\|x - y\|^2$.

39.311.1 Detailed Description

```
template<typename KernelType>
class mlpack::kernel::KernelTraits< KernelType >
```

This is a template class that can provide information about various kernels.

By default, this class will provide the weakest possible assumptions on kernels, and each kernel should override values as necessary. If a kernel doesn't need to override a value, then there's no need to write a **KernelTraits** (p. 1433) specialization for that class.

Definition at line 27 of file kernel_traits.hpp.

39.311.2 Member Data Documentation

39.311.2.1 IsNormalized

```
const bool IsNormalized = false [static]
```

If true, then the kernel is normalized: $K(x, x) = K(y, y) = 1$ for all x .

Definition at line 33 of file kernel_traits.hpp.

39.311.2.2 UsesSquaredDistance

```
const bool UsesSquaredDistance = false [static]
```

If true, then the kernel include a squared distance, $\|x - y\|^2$.

Definition at line 38 of file kernel_traits.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ **kernel_traits.hpp**

39.312 KernelTraits< CauchyKernel > Class Template Reference

Kernel traits for the Cauchy kernel.

Static Public Attributes

- static const bool **IsNormalized** = true
The Cauchy kernel is normalized: $K(x, x) = 1$ for all x .

39.312.1 Detailed Description

```
template<>
class mlpack::kernel::KernelTraits< CauchyKernel >
```

Kernel traits for the Cauchy kernel.

Definition at line 87 of file cauchy_kernel.hpp.

39.312.2 Member Data Documentation

39.312.2.1 IsNormalized

```
const bool IsNormalized = true [static]
```

The Cauchy kernel is normalized: $K(x, x) = 1$ for all x .

Definition at line 91 of file `cauchy_kernel.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ cauchy_kernel.hpp`

39.313 KernelTraits< CosineDistance > Class Template Reference

Kernel traits for the cosine distance.

Static Public Attributes

- static const bool **IsNormalized** = true
The cosine kernel is normalized: $K(x, x) = 1$ for all x .
- static const bool **UsesSquaredDistance** = false
The cosine kernel doesn't include a squared distance.

39.313.1 Detailed Description

```
template<>  
class mlpack::kernel::KernelTraits< CosineDistance >
```

Kernel traits for the cosine distance.

Definition at line 51 of file `cosine_distance.hpp`.

39.313.2 Member Data Documentation

39.313.2.1 IsNormalized

```
const bool IsNormalized = true [static]
```

The cosine kernel is normalized: $K(x, x) = 1$ for all x .

Definition at line 55 of file cosine_distance.hpp.

39.313.2.2 UsesSquaredDistance

```
const bool UsesSquaredDistance = false [static]
```

The cosine kernel doesn't include a squared distance.

Definition at line 58 of file cosine_distance.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ **cosine_distance.hpp**

39.314 KernelTraits< EpanechnikovKernel > Class Template Reference

Kernel traits for the Epanechnikov kernel.

Static Public Attributes

- static const bool **IsNormalized** = true
The Epanechnikov kernel is normalized: $K(x, x) = 1$ for all x .
- static const bool **UsesSquaredDistance** = true
The Epanechnikov kernel includes a squared distance.

39.314.1 Detailed Description

```
template<>
class mlpack::kernel::KernelTraits< EpanechnikovKernel >
```

Kernel traits for the Epanechnikov kernel.

Definition at line 107 of file epanechnikov_kernel.hpp.

39.314.2 Member Data Documentation

39.314.2.1 IsNormalized

```
const bool IsNormalized = true [static]
```

The Epanechnikov kernel is normalized: $K(x, x) = 1$ for all x .

Definition at line 111 of file epanechnikov_kernel.hpp.

39.314.2.2 UsesSquaredDistance

```
const bool UsesSquaredDistance = true [static]
```

The Epanechnikov kernel includes a squared distance.

Definition at line 113 of file epanechnikov_kernel.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ epanechnikov_kernel.hpp

39.315 KernelTraits< GaussianKernel > Class Template Reference

Kernel traits for the Gaussian kernel.

Static Public Attributes

- static const bool **IsNormalized** = true
The Gaussian kernel is normalized: $K(x, x) = 1$ for all x .
- static const bool **UsesSquaredDistance** = true
The Gaussian kernel includes a squared distance.

39.315.1 Detailed Description

```
template<>
class mlpack::kernel::KernelTraits< GaussianKernel >
```

Kernel traits for the Gaussian kernel.

Definition at line 167 of file gaussian_kernel.hpp.

39.315.2 Member Data Documentation

39.315.2.1 IsNormalized

```
const bool IsNormalized = true [static]
```

The Gaussian kernel is normalized: $K(x, x) = 1$ for all x .

Definition at line 171 of file gaussian_kernel.hpp.

39.315.2.2 UsesSquaredDistance

```
const bool UsesSquaredDistance = true [static]
```

The Gaussian kernel includes a squared distance.

Definition at line 173 of file gaussian_kernel.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ **gaussian_kernel.hpp**

39.316 KernelTraits< LaplacianKernel > Class Template Reference

Kernel traits of the Laplacian kernel.

Static Public Attributes

- static const bool **IsNormalized** = true
The Laplacian kernel is normalized: $K(x, x) = 1$ for all x .
- static const bool **UsesSquaredDistance** = false
The Laplacian kernel doesn't include a squared distance.

39.316.1 Detailed Description

```
template<>
class mlpack::kernel::KernelTraits< LaplacianKernel >
```

Kernel traits of the Laplacian kernel.

Definition at line 113 of file laplacian_kernel.hpp.

39.316.2 Member Data Documentation

39.316.2.1 IsNormalized

```
const bool IsNormalized = true [static]
```

The Laplacian kernel is normalized: $K(x, x) = 1$ for all x .

Definition at line 117 of file laplacian_kernel.hpp.

39.316.2.2 UsesSquaredDistance

```
const bool UsesSquaredDistance = false [static]
```

The Laplacian kernel doesn't include a squared distance.

Definition at line 119 of file laplacian_kernel.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ **laplacian_kernel.hpp**

39.317 KernelTraits< SphericalKernel > Class Template Reference

Kernel traits for the spherical kernel.

Static Public Attributes

- static const bool **IsNormalized** = true
The spherical kernel is normalized: $K(x, x) = 1$ for all x .
- static const bool **UsesSquaredDistance** = false
The spherical kernel doesn't include a squared distance.

39.317.1 Detailed Description

```
template<>  
class mlpack::kernel::KernelTraits< SphericalKernel >
```

Kernel traits for the spherical kernel.

Definition at line 123 of file spherical_kernel.hpp.

39.317.2 Member Data Documentation

39.317.2.1 IsNormalized

```
const bool IsNormalized = true [static]
```

The spherical kernel is normalized: $K(x, x) = 1$ for all x .

Definition at line 127 of file spherical_kernel.hpp.

39.317.2.2 UsesSquaredDistance

```
const bool UsesSquaredDistance = false [static]
```

The spherical kernel doesn't include a squared distance.

Definition at line 129 of file spherical_kernel.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ **spherical_kernel.hpp**

39.318 KernelTraits< TriangularKernel > Class Template Reference

Kernel traits for the triangular kernel.

Static Public Attributes

- static const bool **IsNormalized** = true
The triangular kernel is normalized: $K(x, x) = 1$ for all x .
- static const bool **UsesSquaredDistance** = false
The triangular kernel doesn't include a squared distance.

39.318.1 Detailed Description

```
template<>
class mlpack::kernel::KernelTraits< TriangularKernel >
```

Kernel traits for the triangular kernel.

Definition at line 108 of file triangular_kernel.hpp.

39.318.2 Member Data Documentation

39.318.2.1 IsNormalized

```
const bool IsNormalized = true [static]
```

The triangular kernel is normalized: $K(x, x) = 1$ for all x .

Definition at line 112 of file triangular_kernel.hpp.

39.318.2.2 UsesSquaredDistance

```
const bool UsesSquaredDistance = false [static]
```

The triangular kernel doesn't include a squared distance.

Definition at line 114 of file triangular_kernel.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ **triangular_kernel.hpp**

39.319 KMeansSelection< ClusteringType, maxIterations > Class Template Reference

Implementation of the kmeans sampling scheme.

Static Public Member Functions

- static const arma::mat * **Select** (const arma::mat &data, const size_t m)
Use the K-Means clustering method to select the specified number of points in the dataset.

39.319.1 Detailed Description

```
template<typename ClusteringType = kmeans::KMeans<>, size_t maxIterations = 5>
class mlpack::kernel::KMeansSelection< ClusteringType, maxIterations >
```

Implementation of the kmeans sampling scheme.

Template Parameters

<i>ClusteringType</i>	Type of clustering.
<i>maxIterations</i>	Maximum number of iterations allowed before giving up.

Definition at line 29 of file kmeans_selection.hpp.

39.319.2 Member Function Documentation

39.319.2.1 Select()

```
static const arma::mat* Select (
    const arma::mat & data,
    const size_t m ) [inline], [static]
```

Use the K-Means clustering method to select the specified number of points in the dataset.

You are responsible for deleting the returned matrix!

Parameters

<i>data</i>	Dataset to sample from.
<i>m</i>	Number of points to select.

Returns

Matrix pointer in which centroids are stored.

Definition at line 40 of file kmeans_selection.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nystroem_method/ **kmeans_selection.hpp**

39.320 LaplacianKernel Class Reference

The standard Laplacian kernel.

Public Member Functions

- **LaplacianKernel** ()
Default constructor; sets bandwidth to 1.0.
- **LaplacianKernel** (double bandwidth)
Construct the Laplacian kernel with a custom bandwidth.
- double **Bandwidth** () const
Get the bandwidth.
- double & **Bandwidth** ()
Modify the bandwidth.
- template<typename VecTypeA , typename VecTypeB >
double **Evaluate** (const VecTypeA &a, const VecTypeB &b) const
Evaluation of the Laplacian kernel.
- double **Evaluate** (const double t) const
Evaluation of the Laplacian kernel given the distance between two points.
- double **Gradient** (const double t) const
Evaluation of the gradient of the Laplacian kernel given the distance between two points.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the kernel.

39.320.1 Detailed Description

The standard Laplacian kernel.

Given two vectors x , y , and a bandwidth μ (set in the constructor),

$$K(x, y) = \exp\left(-\frac{\|x - y\|}{\mu}\right).$$

The implementation is all in the header file because it is so simple.

Definition at line 30 of file laplacian_kernel.hpp.

39.320.2 Constructor & Destructor Documentation

39.320.2.1 LaplacianKernel() [1/2]

```
LaplacianKernel ( ) [inline]
```

Default constructor; sets bandwidth to 1.0.

Definition at line 36 of file laplacian_kernel.hpp.

39.320.2.2 LaplacianKernel() [2/2]

```
LaplacianKernel (  
    double bandwidth ) [inline]
```

Construct the Laplacian kernel with a custom bandwidth.

Parameters

<i>bandwidth</i>	The bandwidth of the kernel (μ).
------------------	--

Definition at line 44 of file laplacian_kernel.hpp.

39.320.3 Member Function Documentation

39.320.3.1 Bandwidth() [1/2]

```
double Bandwidth ( ) const [inline]
```

Get the bandwidth.

Definition at line 95 of file laplacian_kernel.hpp.

39.320.3.2 Bandwidth() [2/2]

```
double& Bandwidth ( ) [inline]
```

Modify the bandwidth.

Definition at line 97 of file laplacian_kernel.hpp.

39.320.3.3 Evaluate() [1/2]

```
double Evaluate (
    const VecTypeA & a,
    const VecTypeB & b ) const [inline]
```

Evaluation of the Laplacian kernel.

This could be generalized to use any distance metric, not the Euclidean distance, but for now, the Euclidean distance is used.

Template Parameters

<i>VecTypeA</i>	Type of first vector (likely arma::vec or arma::sp_vec).
<i>VecTypeB</i>	Type of second vector (arma::vec / arma::sp_vec).

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

K(a, b) using the bandwidth (μ) specified in the constructor.

Definition at line 61 of file laplacian_kernel.hpp.

References LMetric< TPower, TTakeRoot >::Evaluate().

39.320.3.4 Evaluate() [2/2]

```
double Evaluate (  
    const double t ) const [inline]
```

Evaluation of the Laplacian kernel given the distance between two points.

Parameters

<i>t</i>	The distance between the two points the kernel should be evaluated on.
----------	--

Returns

K(t) using the bandwidth (μ) specified in the constructor.

Definition at line 75 of file laplacian_kernel.hpp.

39.320.3.5 Gradient()

```
double Gradient (  
    const double t ) const [inline]
```

Evaluation of the gradient of the Laplacian kernel given the distance between two points.

Parameters

<i>t</i>	The distance between the two points the kernel should be evaluated on.
----------	--

Returns

$K(t)$ using the bandwidth (μ) specified in the constructor.

Definition at line 90 of file laplacian_kernel.hpp.

39.320.3.6 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the kernel.

Definition at line 101 of file laplacian_kernel.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ **laplacian_kernel.hpp**

39.321 LinearKernel Class Reference

The simple linear kernel (dot product).

Public Member Functions

- **LinearKernel** ()
This constructor does nothing; the linear kernel has no parameters to store.
- template<typename Archive >
void **serialize** (Archive &, const unsigned int)
Serialize the kernel (it has no members... do nothing).

Static Public Member Functions

- template<typename VecTypeA , typename VecTypeB >
static double **Evaluate** (const VecTypeA &a, const VecTypeB &b)
Simple evaluation of the dot product.

39.321.1 Detailed Description

The simple linear kernel (dot product).

For any two vectors x and y ,

$$K(x, y) = x^T y$$

This kernel has no parameters and therefore the evaluation can be static.

Definition at line 32 of file linear_kernel.hpp.

39.321.2 Constructor & Destructor Documentation

39.321.2.1 LinearKernel()

```
LinearKernel ( ) [inline]
```

This constructor does nothing; the linear kernel has no parameters to store.

Definition at line 39 of file linear_kernel.hpp.

39.321.3 Member Function Documentation

39.321.3.1 Evaluate()

```
static double Evaluate (
    const VecTypeA & a,
    const VecTypeB & b ) [inline], [static]
```

Simple evaluation of the dot product.

This evaluation uses Armadillo's dot() function.

Template Parameters

<i>VecTypeA</i>	Type of first vector (should be arma::vec or arma::sp_vec).
<i>VecTypeB</i>	Type of second vector (arma::vec / arma::sp_vec).

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

$K(a, b)$.

Definition at line 53 of file linear_kernel.hpp.

39.321.3.2 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serialize the kernel (it has no members... do nothing).

Definition at line 60 of file linear_kernel.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ **linear_kernel.hpp**

39.322 NystroemMethod< KernelType, PointSelectionPolicy > Class Template Reference

Public Member Functions

- **NystroemMethod** (const arma::mat &data, KernelType &kernel, const size_t rank)
*Create the **NystroemMethod** (p. 1448) object.*
- void **Apply** (arma::mat &output)
*Apply the low-rank factorization to obtain an output matrix G such that $K' = G * G^T$.*
- void **GetKernelMatrix** (const arma::mat *data, arma::mat &miniKernel, arma::mat &semiKernel)
Construct the kernel matrix with matrix that contains the selected points.
- void **GetKernelMatrix** (const arma::Col< size_t > &selectedPoints, arma::mat &miniKernel, arma::mat &semiKernel)
Construct the kernel matrix with the selected points.

39.322.1 Detailed Description

```
template<typename KernelType, typename PointSelectionPolicy = KMeansSelection<>>
class mlpack::kernel::NystroemMethod< KernelType, PointSelectionPolicy >
```

Definition at line 28 of file nystroem_method.hpp.

39.322.2 Constructor & Destructor Documentation

39.322.2.1 NystroemMethod()

```
NystroemMethod (
    const arma::mat & data,
    KernelType & kernel,
    const size_t rank )
```

Create the **NystroemMethod** (p. 1448) object.

The constructor here does not really do anything.

Parameters

<i>data</i>	Data matrix.
<i>kernel</i>	Kernel to be used for computation.
<i>rank</i>	Rank to be used for matrix approximation.

39.322.3 Member Function Documentation

39.322.3.1 Apply()

```
void Apply (
    arma::mat & output )
```

Apply the low-rank factorization to obtain an output matrix G such that $K' = G * G^T$.

Parameters

<i>output</i>	Matrix to store kernel approximation into.
---------------	--

Referenced by NystroemKernelRule< KernelType, PointSelectionPolicy >::ApplyKernelMatrix().

39.322.3.2 GetKernelMatrix() [1/2]

```
void GetKernelMatrix (
    const arma::mat & data,
```

```
arma::mat & miniKernel,
arma::mat & semiKernel )
```

Construct the kernel matrix with matrix that contains the selected points.

Parameters

<i>data</i>	Data matrix pointer.
<i>miniKernel</i>	to store the constructed mini-kernel matrix in.
<i>miniKernel</i>	to store the constructed semi-kernel matrix in.

39.322.3.3 GetKernelMatrix() [2/2]

```
void GetKernelMatrix (
    const arma::Col< size_t > & selectedPoints,
    arma::mat & miniKernel,
    arma::mat & semiKernel )
```

Construct the kernel matrix with the selected points.

Parameters

<i>points</i>	Indices of selected points.
<i>miniKernel</i>	to store the constructed mini-kernel matrix in.
<i>miniKernel</i>	to store the constructed semi-kernel matrix in.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nystroem_method/ **nystroem_method.hpp**

39.323 OrderedSelection Class Reference

Static Public Member Functions

- static const arma::Col< size_t > **Select** (const arma::mat &, const size_t m)
Select the specified number of points in the dataset.

39.323.1 Detailed Description

Definition at line 22 of file ordered_selection.hpp.

39.323.2 Member Function Documentation

39.323.2.1 Select()

```
static const arma::Col<size_t> Select (
    const arma::mat & ,
    const size_t m ) [inline], [static]
```

Select the specified number of points in the dataset.

Parameters

<i>data</i>	Dataset to sample from.
<i>m</i>	Number of points to select.

Returns

Indices of selected points from the dataset.

Definition at line 32 of file `ordered_selection.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nystroem_method/ ordered_selection.hpp`

39.324 PolynomialKernel Class Reference

The simple polynomial kernel.

Public Member Functions

- **PolynomialKernel** (const double degree=2.0, const double offset=0.0)
Construct the Polynomial Kernel with the given offset and degree.
- const double & **Degree** () const
Get the degree of the polynomial.
- double & **Degree** ()
Modify the degree of the polynomial.
- template<typename VecTypeA , typename VecTypeB >
double **Evaluate** (const VecTypeA &a, const VecTypeB &b) const
Simple evaluation of the dot product.
- const double & **Offset** () const
Get the offset of the dot product of the arguments.
- double & **Offset** ()
Modify the offset of the dot product of the arguments.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the kernel.

39.324.1 Detailed Description

The simple polynomial kernel.

For any two vectors x , y , *degree* and *offset*,

$$K(x, y) = (x^T * y + offset)^{degree}.$$

Definition at line 28 of file polynomial_kernel.hpp.

39.324.2 Constructor & Destructor Documentation

39.324.2.1 PolynomialKernel()

```
PolynomialKernel (
    const double degree = 2.0,
    const double offset = 0.0 ) [inline]
```

Construct the Polynomial Kernel with the given offset and degree.

If the arguments are omitted, the default degree is 2 and the default offset is 0.

Parameters

<i>offset</i>	Offset of the dot product of the arguments.
<i>degree</i>	Degree of the polynomial.

Definition at line 38 of file polynomial_kernel.hpp.

39.324.3 Member Function Documentation

39.324.3.1 Degree() [1/2]

```
const double& Degree ( ) const [inline]
```

Get the degree of the polynomial.

Definition at line 61 of file polynomial_kernel.hpp.

39.324.3.2 Degree() [2/2]

```
double& Degree ( ) [inline]
```

Modify the degree of the polynomial.

Definition at line 63 of file polynomial_kernel.hpp.

39.324.3.3 Evaluate()

```
double Evaluate (
    const VecTypeA & a,
    const VecTypeB & b ) const [inline]
```

Simple evaluation of the dot product.

This evaluation uses Armadillo's dot() function.

Template Parameters

<i>VecTypeA</i>	Type of first vector (should be arma::vec or arma::sp_vec).
<i>VecTypeB</i>	Type of second vector (arma::vec / arma::sp_vec).

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

K(a, b).

Definition at line 55 of file polynomial_kernel.hpp.

39.324.3.4 Offset() [1/2]

```
const double& Offset ( ) const [inline]
```

Get the offset of the dot product of the arguments.

Definition at line 66 of file polynomial_kernel.hpp.

39.324.3.5 Offset() [2/2]

```
double& Offset ( ) [inline]
```

Modify the offset of the dot product of the arguments.

Definition at line 68 of file polynomial_kernel.hpp.

39.324.3.6 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the kernel.

Definition at line 72 of file polynomial_kernel.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ **polynomial_kernel.hpp**

39.325 PSpectrumStringKernel Class Reference

The p-spectrum string kernel.

Public Member Functions

- **PSpectrumStringKernel** (const std::vector< std::vector< std::string > > &datasets, const size_t p)
*Initialize the **PSpectrumStringKernel** (p. 1454) with the given string datasets.*
- const std::vector< std::vector< std::map< std::string, int > > > & **Counts** () const
Access the lists of substrings.
- std::vector< std::vector< std::map< std::string, int > > > & **Counts** ()
Modify the lists of substrings.
- template<typename VecType >
double **Evaluate** (const VecType &a, const VecType &b) const
Evaluate the kernel for the string indices given.
- size_t **P** () const
Access the value of p.
- size_t & **P** ()
Modify the value of p.

39.325.1 Detailed Description

The p-spectrum string kernel.

Given a length p , the p-spectrum kernel finds the contiguous subsequence match count between two strings. The kernel will take every possible substring of length p of one string and count how many times it appears in the other string.

The string kernel, when created, must be passed a reference to a series of string datasets (`std::vector<std::vector<std::string>>&`). This is because mlpack only supports datasets which are Armadillo matrices – and a dataset of variable-length strings cannot be easily cast into an Armadillo matrix.

Therefore, once the **PSpectrumStringKernel** (p. 1454) is created with a reference to the string datasets, a "fake" Armadillo data matrix must be created, which simply holds indices to the strings they represent. This "fake" matrix has two rows and n columns (where n is the number of strings in the dataset). The first row holds the index of the dataset (remember, the kernel can have multiple datasets), and the second row holds the index of the string. A fake matrix containing only strings from dataset 0 might look like this:

```
[[0 0 0 0 0 0 0 0] [0 1 2 3 4 5 6 7 8]]
```

This fake matrix is then given to the machine learning method, which will eventually call `PSpectrumStringKernel::Evaluate(a, b)`, where a and b are two columns of the fake matrix. The string kernel will then map these fake columns back to the strings they represent, and then correctly evaluate the kernel.

Unfortunately, not every machine learning method will work with this kernel. Only machine learning methods which do not ever operate on the explicit representation of points can use this kernel. So, for instance, one cannot build a kd-tree on strings, because the `BinarySpaceTree<>` class will split the data according to the fake data matrix – resulting in a meaningless tree. This kernel was originally written for the FastMKS method; so, at the very least, it will work with that.

Definition at line 65 of file `pspectrum_string_kernel.hpp`.

39.325.2 Constructor & Destructor Documentation

39.325.2.1 PSpectrumStringKernel()

```
PSpectrumStringKernel (
    const std::vector< std::vector< std::string > > & datasets,
    const size_t p )
```

Initialize the **PSpectrumStringKernel** (p. 1454) with the given string datasets.

For more information on this, see the general class documentation.

Parameters

<i>datasets</i>	Sets of string data.
<i>p</i>	The length of substrings to search.

39.325.3 Member Function Documentation

39.325.3.1 Counts() [1/2]

```
const std::vector<std::vector<std::map<std::string, int> > >> Counts ( ) const [inline]
```

Access the lists of substrings.

Definition at line 93 of file pspectrum_string_kernel.hpp.

39.325.3.2 Counts() [2/2]

```
std::vector<std::vector<std::map<std::string, int> > >> Counts ( ) [inline]
```

Modify the lists of substrings.

Definition at line 96 of file pspectrum_string_kernel.hpp.

39.325.3.3 Evaluate()

```
double Evaluate (
    const VecType & a,
    const VecType & b ) const
```

Evaluate the kernel for the string indices given.

As mentioned in the class documentation, a and b should be 2-element vectors, where the first element contains the index of the dataset and the second element contains the index of the string. Therefore, if [2 3] is passed for a, the string used will be datasets[2][3] (datasets is of type std::vector<std::vector<std::string> >&).

Parameters

<i>a</i>	Index of string and dataset for first string.
<i>b</i>	Index of string and dataset for second string.

39.325.3.4 P() [1/2]

```
size_t P ( ) const [inline]
```


Access the value of `p`.

Definition at line 100 of file `pspectrum_string_kernel.hpp`.

39.325.3.5 `P()` [2/2]

```
size_t& P ( ) [inline]
```

Modify the value of `p`.

Definition at line 102 of file `pspectrum_string_kernel.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ pspectrum_string_kernel.hpp`

39.326 RandomSelection Class Reference

Static Public Member Functions

- static const arma::Col< size_t > **Select** (const arma::mat &data, const size_t m)
Randomly select the specified number of points in the dataset.

39.326.1 Detailed Description

Definition at line 22 of file `random_selection.hpp`.

39.326.2 Member Function Documentation

39.326.2.1 `Select()`

```
static const arma::Col<size_t> Select (
    const arma::mat & data,
    const size_t m ) [inline], [static]
```

Randomly select the specified number of points in the dataset.

Parameters

<i>data</i>	Dataset to sample from.
<i>m</i>	Number of points to select.

Returns

Indices of selected points from the dataset.

Definition at line 32 of file random_selection.hpp.

References `mlpack::math::RandInt()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nystroem_method/ random_selection.hpp`

39.327 SphericalKernel Class Reference

The spherical kernel, which is 1 when the distance between the two argument points is less than or equal to the bandwidth, or 0 otherwise.

Public Member Functions

- **SphericalKernel** (const double bandwidth=1.0)
*Construct the **SphericalKernel** (p. 1458) with the given bandwidth.*
- `template<typename VecTypeA , typename VecTypeB >`
`double ConvolutionIntegral (const VecTypeA &a, const VecTypeB &b) const`
Obtains the convolution integral $[\int K(\|x-a\|)K(\|b-x\|)dx]$ for the two vectors.
- `template<typename VecTypeA , typename VecTypeB >`
`double Evaluate (const VecTypeA &a, const VecTypeB &b) const`
Evaluate the spherical kernel with the given two vectors.
- `double Evaluate (const double t) const`
Evaluate the kernel when only a distance is given, not two points.
- `double Gradient (double t)`
- `double Normalizer (size_t dimension) const`
- `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`
Serialize the object.

39.327.1 Detailed Description

The spherical kernel, which is 1 when the distance between the two argument points is less than or equal to the bandwidth, or 0 otherwise.

Definition at line 23 of file spherical_kernel.hpp.

39.327.2 Constructor & Destructor Documentation

39.327.2.1 SphericalKernel()

```
SphericalKernel (
    const double bandwidth = 1.0 ) [inline]
```

Construct the **SphericalKernel** (p. 1458) with the given bandwidth.

Definition at line 29 of file spherical_kernel.hpp.

39.327.3 Member Function Documentation

39.327.3.1 ConvolutionIntegral()

```
double ConvolutionIntegral (
    const VecTypeA & a,
    const VecTypeB & b ) const [inline]
```

Obtains the convolution integral $[\text{integral } K(\|x-a\|)K(\|b-x\|)dx]$ for the two vectors.

Template Parameters

<i>VecTypeA</i>	Type of first vector (arma::vec, arma::sp_vec should be expected).
<i>VecTypeB</i>	Type of second vector.

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

the convolution integral value.

Definition at line 62 of file spherical_kernel.hpp.

References `LMetric< TPower, TTakeRoot >::Evaluate()`, `Log::Fatal`, and `SphericalKernel::Normalizer()`.

39.327.3.2 Evaluate() [1/2]

```
double Evaluate (
    const VecTypeA & a,
    const VecTypeB & b ) const [inline]
```

Evaluate the spherical kernel with the given two vectors.

Template Parameters

<i>VecTypeA</i>	Type of first vector.
<i>VecTypeB</i>	Type of second vector.

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

The kernel evaluation between the two vectors.

Definition at line 44 of file spherical_kernel.hpp.

References `LMetric< TPower, TTakeRoot >::Evaluate()`.

39.327.3.3 Evaluate() [2/2]

```
double Evaluate (
    const double t ) const [inline]
```

Evaluate the kernel when only a distance is given, not two points.

Parameters

<i>t</i>	Argument to kernel.
----------	---------------------

Definition at line 99 of file spherical_kernel.hpp.

39.327.3.4 Gradient()

```
double Gradient (
    double t ) [inline]
```

Definition at line 103 of file spherical_kernel.hpp.

39.327.3.5 Normalizer()

```
double Normalizer (
    size_t dimension ) const [inline]
```

Definition at line 88 of file spherical_kernel.hpp.

References `M_PI`.

Referenced by `SphericalKernel::ConvolutionIntegral()`.

39.327.3.6 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the object.

Definition at line 110 of file spherical_kernel.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ **spherical_kernel.hpp**

39.328 TriangularKernel Class Reference

The trivially simple triangular kernel, defined by.

Public Member Functions

- **TriangularKernel** (const double bandwidth=1.0)
Initialize the triangular kernel with the given bandwidth (default 1.0).
- double **Bandwidth** () const
Get the bandwidth of the kernel.
- double & **Bandwidth** ()
Modify the bandwidth of the kernel.
- template<typename VecTypeA , typename VecTypeB >
double **Evaluate** (const VecTypeA &a, const VecTypeB &b) const
Evaluate the triangular kernel for the two given vectors.
- double **Evaluate** (const double distance) const
Evaluate the triangular kernel given that the distance between the two points is known.
- double **Gradient** (const double distance) const
Evaluate the gradient of triangular kernel given that the distance between the two points is known.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the kernel.

39.328.1 Detailed Description

The trivially simple triangular kernel, defined by.

$$K(x, y) = \max\{0, 1 - \frac{\|x - y\|_2}{b}\}$$

where b is the bandwidth of the kernel.

Definition at line 30 of file triangular_kernel.hpp.

39.328.2 Constructor & Destructor Documentation

39.328.2.1 TriangularKernel()

```
TriangularKernel (
    const double bandwidth = 1.0 ) [inline]
```

Initialize the triangular kernel with the given bandwidth (default 1.0).

Parameters

<i>bandwidth</i>	Bandwidth of the triangular kernel.
------------------	-------------------------------------

Definition at line 38 of file triangular_kernel.hpp.

39.328.3 Member Function Documentation

39.328.3.1 Bandwidth() [1/2]

```
double Bandwidth ( ) const [inline]
```

Get the bandwidth of the kernel.

Definition at line 90 of file triangular_kernel.hpp.

39.328.3.2 Bandwidth() [2/2]

```
double& Bandwidth ( ) [inline]
```

Modify the bandwidth of the kernel.

Definition at line 92 of file triangular_kernel.hpp.

39.328.3.3 Evaluate() [1/2]

```
double Evaluate (
    const VecTypeA & a,
    const VecTypeB & b ) const [inline]
```

Evaluate the triangular kernel for the two given vectors.

Template Parameters

<i>VecTypeA</i>	Type of first vector.
<i>VecTypeB</i>	Type of second vector.

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Definition at line 49 of file triangular_kernel.hpp.

References `LMetric< TPower, TTakeRoot >::Evaluate()`.

39.328.3.4 Evaluate() [2/2]

```
double Evaluate (
    const double distance ) const [inline]
```

Evaluate the triangular kernel given that the distance between the two points is known.

Parameters

<i>distance</i>	The distance between the two points.
-----------------	--------------------------------------

Definition at line 61 of file triangular_kernel.hpp.

39.328.3.5 Gradient()

```
double Gradient (
    const double distance ) const [inline]
```

Evaluate the gradient of triangular kernel given that the distance between the two points is known.

Parameters

<i>distance</i>	The distance between the two points.
-----------------	--------------------------------------

Definition at line 73 of file triangular_kernel.hpp.

39.328.3.6 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the kernel.

Definition at line 96 of file triangular_kernel.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/ **triangular_kernel.hpp**

39.329 AllowEmptyClusters Class Reference

Policy which allows K-Means to create empty clusters without any error being reported.

Public Member Functions

- **AllowEmptyClusters** ()
Default constructor required by EmptyClusterPolicy policy.
- `template<typename Archive >`
`void serialize (Archive &, const unsigned int)`
Serialize the empty cluster policy (nothing to do).

Static Public Member Functions

- `template<typename MetricType , typename MatType >`
`static force_inline void EmptyCluster (const MatType &, const size_t emptyCluster, const arma::mat &oldCentroids, arma::mat &newCentroids, arma::Col< size_t > &, MetricType &, const size_t)`

This function allows empty clusters to persist simply by leaving the empty cluster in its last position.

39.329.1 Detailed Description

Policy which allows K-Means to create empty clusters without any error being reported.

Definition at line 25 of file `allow_empty_clusters.hpp`.

39.329.2 Constructor & Destructor Documentation

39.329.2.1 AllowEmptyClusters()

```
AllowEmptyClusters ( ) [inline]
```

Default constructor required by EmptyClusterPolicy policy.

Definition at line 29 of file `allow_empty_clusters.hpp`.

39.329.3 Member Function Documentation

39.329.3.1 EmptyCluster()

```
static force_inline void EmptyCluster (
    const MatType & ,
    const size_t emptyCluster,
    const arma::mat & oldCentroids,
    arma::mat & newCentroids,
    arma::Col< size_t > & ,
    MetricType & ,
    const size_t ) [inline], [static]
```

This function allows empty clusters to persist simply by leaving the empty cluster in its last position.

Template Parameters

<i>MatType</i>	Type of data (arma::mat or arma::spmat).
----------------	--

Parameters

<i>data</i>	Dataset on which clustering is being performed.
<i>emptyCluster</i>	Index of cluster which is empty.
<i>oldCentroids</i>	Centroids of each cluster (one per column) at the start of the iteration.
<i>newCentroids</i>	Centroids of each cluster (one per column) at the end of the iteration.
<i>clusterCounts</i>	Number of points in each cluster.
<i>assignments</i>	Cluster assignments of each point.
<i>iteration</i>	Number of iteration.

Returns

Number of points changed (0).

Definition at line 49 of file allow_empty_clusters.hpp.

39.329.3.2 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serialize the empty cluster policy (nothing to do).

Definition at line 64 of file allow_empty_clusters.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ **allow_empty_clusters.hpp**

39.330 DualTreeKMeans< MetricType, MatType, TreeType > Class Template Reference

An algorithm for an exact Lloyd iteration which simply uses dual-tree nearest-neighbor search to find the nearest centroid for each point in the dataset.

Public Types

- `template<typename TreeMetricType, typename IgnoredStatType, typename TreeMatType > using NNSTreeType = TreeType< TreeMetricType, DualTreeKMeansStatistic, TreeMatType >`
- `typedef TreeType< MetricType, DualTreeKMeansStatistic, MatType > Tree`
Convenience typedef.

Public Member Functions

- **DualTreeKMeans** (const MatType &dataset, MetricType &metric)
*Construct the **DualTreeKMeans** (p. 1466) object, which will construct a tree on the points.*
- **~DualTreeKMeans** ()
*Delete the tree constructed by the **DualTreeKMeans** (p. 1466) object.*
- `size_t DistanceCalculations () const`
Return the number of distance calculations.
- `size_t & DistanceCalculations ()`
Modify the number of distance calculations.
- `double Iterate (const arma::mat ¢roids, arma::mat &newCentroids, arma::Col< size_t > &counts)`
Run a single iteration of the dual-tree nearest neighbor algorithm for k-means, updating the given centroids into the new← Centroids matrix.

39.330.1 Detailed Description

```
template<typename MetricType, typename MatType, template< typename TreeMetricType, typename TreeStatType, typename Tree←
MatType > class TreeType = tree::KDTree>
class mlpack::kmeans::DualTreeKMeans< MetricType, MatType, TreeType >
```

An algorithm for an exact Lloyd iteration which simply uses dual-tree nearest-neighbor search to find the nearest centroid for each point in the dataset.

The conditions under which this will perform best are probably limited to the case where k is close to the number of points in the dataset, and the number of iterations of the k-means algorithm will be few.

Definition at line 41 of file dual_tree_kmeans.hpp.

39.330.2 Member Typedef Documentation

39.330.2.1 NNSTreeType

```
using NNSTreeType = TreeType<TreeMetricType, DualTreeKMeansStatistic, TreeMatType>
```

Definition at line 51 of file dual_tree_kmeans.hpp.

39.330.2.2 Tree

```
typedef TreeType<MetricType, DualTreeKMeansStatistic, MatType> Tree
```

Convenience typedef.

Definition at line 45 of file dual_tree_kmeans.hpp.

39.330.3 Constructor & Destructor Documentation

39.330.3.1 DualTreeKMeans()

```
DualTreeKMeans (  
    const MatType & dataset,  
    MetricType & metric )
```

Construct the **DualTreeKMeans** (p. 1466) object, which will construct a tree on the points.

39.330.3.2 ~DualTreeKMeans()

```
~ DualTreeKMeans ( )
```

Delete the tree constructed by the **DualTreeKMeans** (p. 1466) object.

39.330.4 Member Function Documentation

39.330.4.1 DistanceCalculations() [1/2]

```
size_t DistanceCalculations ( ) const [inline]
```

Return the number of distance calculations.

Definition at line 77 of file dual_tree_kmeans.hpp.

39.330.4.2 DistanceCalculations() [2/2]

```
size_t& DistanceCalculations ( ) [inline]
```

Modify the number of distance calculations.

Definition at line 79 of file dual_tree_kmeans.hpp.

References `mlpack::kmeans::HideChild()`, and `mlpack::kmeans::RestoreChildren()`.

39.330.4.3 Iterate()

```
double Iterate (
    const arma::mat & centroids,
    arma::mat & newCentroids,
    arma::Col< size_t > & counts )
```

Run a single iteration of the dual-tree nearest neighbor algorithm for k-means, updating the given centroids into the `newCentroids` matrix.

Parameters

<i>centroids</i>	Current cluster centroids.
<i>newCentroids</i>	New cluster centroids.
<i>counts</i>	Current counts, to be overwritten with new counts.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ dual_tree_kmeans.hpp`

39.331 DualTreeKMeansRules< MetricType, TreeType > Class Template Reference

Public Types

- typedef `tree::TraversallInfo< TreeType > TraversallInfoType`

Public Member Functions

- **DualTreeKMeansRules** (const arma::mat ¢roids, const arma::mat &dataset, arma::Row< size_t > &assignments, arma::vec &upperBounds, arma::vec &lowerBounds, MetricType &metric, const std::vector< bool > &prunedPoints, const std::vector< size_t > &oldFromNewCentroids, std::vector< bool > &visited)
- double **BaseCase** (const size_t queryIndex, const size_t referenceIndex)
- size_t **BaseCases** () const

- `size_t & BaseCases ()`
- `double Rescore (const size_t queryIndex, TreeType &referenceNode, const double oldScore)`
- `double Rescore (TreeType &queryNode, TreeType &referenceNode, const double oldScore)`
- `double Score (const size_t queryIndex, TreeType &referenceNode)`
- `double Score (TreeType &queryNode, TreeType &referenceNode)`
- `size_t Scores () const`
- `size_t & Scores ()`
- `TraversalInfoType & TraversalInfo ()`
- `const TraversalInfoType & TraversalInfo () const`

39.331.1 Detailed Description

```
template<typename MetricType, typename TreeType>
class mlpack::kmeans::DualTreeKMeansRules< MetricType, TreeType >
```

Definition at line 23 of file `dual_tree_kmeans_rules.hpp`.

39.331.2 Member Typedef Documentation

39.331.2.1 TraversalInfoType

```
typedef tree::TraversalInfo<TreeType> TraversalInfoType
```

Definition at line 47 of file `dual_tree_kmeans_rules.hpp`.

39.331.3 Constructor & Destructor Documentation

39.331.3.1 DualTreeKMeansRules()

```
DualTreeKMeansRules (
    const arma::mat & centroids,
    const arma::mat & dataset,
    arma::Row< size_t > & assignments,
    arma::vec & upperBounds,
    arma::vec & lowerBounds,
    MetricType & metric,
    const std::vector< bool > & prunedPoints,
    const std::vector< size_t > & oldFromNewCentroids,
    std::vector< bool > & visited )
```

39.331.4 Member Function Documentation

39.331.4.1 BaseCase()

```
double BaseCase (
    const size_t queryIndex,
    const size_t referenceIndex )
```

39.331.4.2 BaseCases() [1/2]

```
size_t BaseCases ( ) const [inline]
```

Definition at line 52 of file dual_tree_kmeans_rules.hpp.

39.331.4.3 BaseCases() [2/2]

```
size_t& BaseCases ( ) [inline]
```

Definition at line 53 of file dual_tree_kmeans_rules.hpp.

39.331.4.4 Rescore() [1/2]

```
double Rescore (
    const size_t queryIndex,
    TreeType & referenceNode,
    const double oldScore )
```

39.331.4.5 Rescore() [2/2]

```
double Rescore (
    TreeType & queryNode,
    TreeType & referenceNode,
    const double oldScore )
```

39.331.4.6 Score() [1/2]

```
double Score (
    const size_t queryIndex,
    TreeType & referenceNode )
```

39.331.4.7 Score() [2/2]

```
double Score (
    TreeType & queryNode,
    TreeType & referenceNode )
```

39.331.4.8 Scores() [1/2]

```
size_t Scores ( ) const [inline]
```

Definition at line 55 of file `dual_tree_kmeans_rules.hpp`.

39.331.4.9 Scores() [2/2]

```
size_t& Scores ( ) [inline]
```

Definition at line 56 of file `dual_tree_kmeans_rules.hpp`.

39.331.4.10 TraversalInfo() [1/2]

```
TraversalInfoType& TraversalInfo ( ) [inline]
```

Definition at line 49 of file `dual_tree_kmeans_rules.hpp`.

39.331.4.11 TraversalInfo() [2/2]

```
const TraversalInfoType& TraversalInfo ( ) const [inline]
```

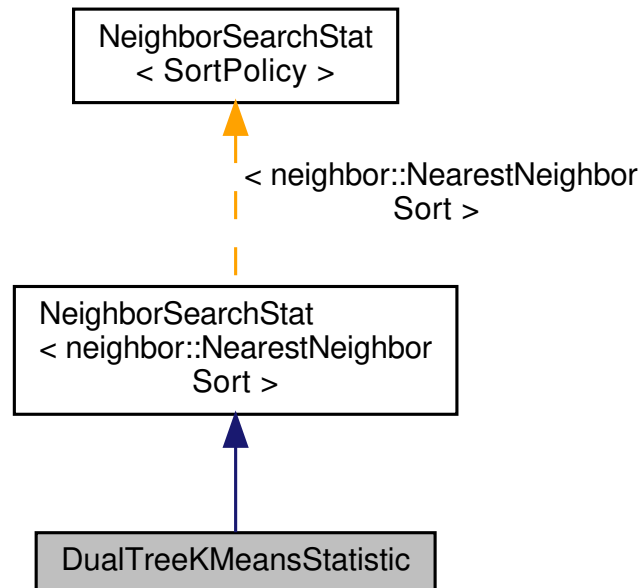
Definition at line 50 of file `dual_tree_kmeans_rules.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ dual_tree_kmeans_rules.hpp`

39.332 DualTreeKMeansStatistic Class Reference

Inheritance diagram for DualTreeKMeansStatistic:



Public Member Functions

- **DualTreeKMeansStatistic** ()
- `template<typename TreeType >`
DualTreeKMeansStatistic (TreeType &node)
- `const arma::vec & Centroid` () const
- `arma::vec & Centroid` ()
- `double LowerBound` () const
- `double & LowerBound` ()
- `size_t NumTrueChildren` () const
- `size_t Owner` () const
- `size_t & Owner` ()
- `size_t Pruned` () const
- `size_t & Pruned` ()
- `double StaticLowerBoundMovement` () const
- `double & StaticLowerBoundMovement` ()
- `bool StaticPruned` () const
- `bool & StaticPruned` ()
- `double StaticUpperBoundMovement` () const
- `double & StaticUpperBoundMovement` ()
- `void * TrueChild` (const size_t i) const
- `void *& TrueChild` (const size_t i)
- `void * TrueParent` () const
- `void *& TrueParent` ()
- `double UpperBound` () const
- `double & UpperBound` ()

39.332.1 Detailed Description

Definition at line 20 of file dual_tree_kmeans_statistic.hpp.

39.332.2 Constructor & Destructor Documentation

39.332.2.1 DualTreeKMeansStatistic() [1/2]

```
DualTreeKMeansStatistic ( ) [inline]
```

Definition at line 24 of file dual_tree_kmeans_statistic.hpp.

39.332.2.2 DualTreeKMeansStatistic() [2/2]

```
DualTreeKMeansStatistic (
    TreeType & node ) [inline]
```

Definition at line 40 of file dual_tree_kmeans_statistic.hpp.

39.332.3 Member Function Documentation

39.332.3.1 Centroid() [1/2]

```
const arma::vec& Centroid ( ) const [inline]
```

Definition at line 81 of file dual_tree_kmeans_statistic.hpp.

39.332.3.2 Centroid() [2/2]

```
arma::vec& Centroid ( ) [inline]
```

Definition at line 82 of file dual_tree_kmeans_statistic.hpp.

39.332.3.3 LowerBound() [1/2]

```
double LowerBound ( ) const [inline]
```

Definition at line 78 of file dual_tree_kmeans_statistic.hpp.

39.332.3.4 LowerBound() [2/2]

```
double& LowerBound ( ) [inline]
```

Definition at line 79 of file dual_tree_kmeans_statistic.hpp.

39.332.3.5 NumTrueChildren()

```
size_t NumTrueChildren ( ) const [inline]
```

Definition at line 105 of file dual_tree_kmeans_statistic.hpp.

39.332.3.6 Owner() [1/2]

```
size_t Owner ( ) const [inline]
```

Definition at line 84 of file dual_tree_kmeans_statistic.hpp.

39.332.3.7 Owner() [2/2]

```
size_t& Owner ( ) [inline]
```

Definition at line 85 of file dual_tree_kmeans_statistic.hpp.

39.332.3.8 Pruned() [1/2]

```
size_t Pruned ( ) const [inline]
```

Definition at line 87 of file dual_tree_kmeans_statistic.hpp.

39.332.3.9 Pruned() [2/2]

```
size_t& Pruned ( ) [inline]
```

Definition at line 88 of file dual_tree_kmeans_statistic.hpp.

39.332.3.10 StaticLowerBoundMovement() [1/2]

```
double StaticLowerBoundMovement ( ) const [inline]
```

Definition at line 96 of file dual_tree_kmeans_statistic.hpp.

39.332.3.11 StaticLowerBoundMovement() [2/2]

```
double& StaticLowerBoundMovement ( ) [inline]
```

Definition at line 97 of file dual_tree_kmeans_statistic.hpp.

39.332.3.12 StaticPruned() [1/2]

```
bool StaticPruned ( ) const [inline]
```

Definition at line 90 of file dual_tree_kmeans_statistic.hpp.

39.332.3.13 StaticPruned() [2/2]

```
bool& StaticPruned ( ) [inline]
```

Definition at line 91 of file dual_tree_kmeans_statistic.hpp.

39.332.3.14 StaticUpperBoundMovement() [1/2]

```
double StaticUpperBoundMovement ( ) const [inline]
```

Definition at line 93 of file dual_tree_kmeans_statistic.hpp.

39.332.3.15 StaticUpperBoundMovement() [2/2]

```
double& StaticUpperBoundMovement ( ) [inline]
```

Definition at line 94 of file dual_tree_kmeans_statistic.hpp.

39.332.3.16 TrueChild() [1/2]

```
void* TrueChild (
    const size_t i ) const [inline]
```

Definition at line 102 of file dual_tree_kmeans_statistic.hpp.

39.332.3.17 TrueChild() [2/2]

```
void*& TrueChild (
    const size_t i ) [inline]
```

Definition at line 103 of file dual_tree_kmeans_statistic.hpp.

39.332.3.18 TrueParent() [1/2]

```
void* TrueParent ( ) const [inline]
```

Definition at line 99 of file dual_tree_kmeans_statistic.hpp.

39.332.3.19 TrueParent() [2/2]

```
void*& TrueParent ( ) [inline]
```

Definition at line 100 of file dual_tree_kmeans_statistic.hpp.

39.332.3.20 UpperBound() [1/2]

```
double UpperBound ( ) const [inline]
```

Definition at line 75 of file dual_tree_kmeans_statistic.hpp.

39.332.3.21 UpperBound() [2/2]

```
double& UpperBound ( ) [inline]
```

Definition at line 76 of file dual_tree_kmeans_statistic.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ **dual_tree_kmeans_statistic.hpp**

39.333 ElkanKMeans< MetricType, MatType > Class Template Reference**Public Member Functions**

- **ElkanKMeans** (const MatType &dataset, MetricType &metric)
*Construct the **ElkanKMeans** (p. 1478) object, which must store several sets of bounds.*
- size_t **DistanceCalculations** () const
- double **Iterate** (const arma::mat ¢roids, arma::mat &newCentroids, arma::Col< size_t > &counts)
Run a single iteration of Elkan's algorithm, updating the given centroids into the newCentroids matrix.

39.333.1 Detailed Description

```
template<typename MetricType, typename MatType>
class mlpack::kmeans::ElkanKMeans< MetricType, MatType >
```

Definition at line 19 of file elkan_kmeans.hpp.

39.333.2 Constructor & Destructor Documentation

39.333.2.1 ElkanKMeans()

```
ElkanKMeans (
    const MatType & dataset,
    MetricType & metric )
```

Construct the **ElkanKMeans** (p. 1478) object, which must store several sets of bounds.

39.333.3 Member Function Documentation

39.333.3.1 DistanceCalculations()

```
size_t DistanceCalculations ( ) const [inline]
```

Definition at line 39 of file elkan_kmeans.hpp.

39.333.3.2 Iterate()

```
double Iterate (
    const arma::mat & centroids,
    arma::mat & newCentroids,
    arma::Col< size_t > & counts )
```

Run a single iteration of Elkan's algorithm, updating the given centroids into the newCentroids matrix.

Parameters

<i>centroids</i>	Current cluster centroids.
<i>newCentroids</i>	New cluster centroids.
<i>counts</i>	Current counts, to be overwritten with new counts.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ **elkan_kmeans.hpp**

39.334 HamerlyKMeans< MetricType, MatType > Class Template Reference

Public Member Functions

- **HamerlyKMeans** (const MatType &dataset, MetricType &metric)

Construct the **HamerlyKMeans** (p. 1479) object, which must store several sets of bounds.

- `size_t` **DistanceCalculations** () const
- `double` **Iterate** (const arma::mat ¢roids, arma::mat &newCentroids, arma::Col< size_t > &counts)

Run a single iteration of Hamerly's algorithm, updating the given centroids into the newCentroids matrix.

39.334.1 Detailed Description

```
template<typename MetricType, typename MatType>
class mlpack::kmeans::HamerlyKMeans< MetricType, MatType >
```

Definition at line 19 of file hamerly_kmeans.hpp.

39.334.2 Constructor & Destructor Documentation

39.334.2.1 HamerlyKMeans()

```
HamerlyKMeans (
    const MatType & dataset,
    MetricType & metric )
```

Construct the **HamerlyKMeans** (p. 1479) object, which must store several sets of bounds.

39.334.3 Member Function Documentation

39.334.3.1 DistanceCalculations()

```
size_t DistanceCalculations ( ) const [inline]
```

Definition at line 40 of file hamerly_kmeans.hpp.

39.334.3.2 Iterate()

```
double Iterate (
    const arma::mat & centroids,
    arma::mat & newCentroids,
    arma::Col< size_t > & counts )
```

Run a single iteration of Hamerly's algorithm, updating the given centroids into the newCentroids matrix.

Parameters

<i>centroids</i>	Current cluster centroids.
<i>newCentroids</i>	New cluster centroids.
<i>counts</i>	Current counts, to be overwritten with new counts.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ **hamerly_kmeans.hpp**

39.335 KillEmptyClusters Class Reference

Policy which allows K-Means to "kill" empty clusters without any error being reported.

Public Member Functions

- **KillEmptyClusters** ()
Default constructor required by EmptyClusterPolicy policy.
- template<typename Archive >
void **serialize** (Archive &, const unsigned int)
Serialize the empty cluster policy (nothing to do).

Static Public Member Functions

- template<typename MetricType , typename MatType >
static **force_inline** void **EmptyCluster** (const MatType &, const size_t emptyCluster, const arma::mat &, arma::mat &newCentroids, arma::Col< size_t > &clusterCounts, MetricType &, const size_t)
This function sets an empty cluster found during k-means to all DBL_MAX (i.e.

39.335.1 Detailed Description

Policy which allows K-Means to "kill" empty clusters without any error being reported.

This means the centroids will be filled with DBL_MAX.

Definition at line 25 of file kill_empty_clusters.hpp.

39.335.2 Constructor & Destructor Documentation

39.335.2.1 KillEmptyClusters()

```
KillEmptyClusters ( ) [inline]
```

Default constructor required by EmptyClusterPolicy policy.

Definition at line 29 of file kill_empty_clusters.hpp.

39.335.3 Member Function Documentation

39.335.3.1 EmptyCluster()

```
static force_inline void EmptyCluster (
    const MatType & ,
    const size_t emptyCluster,
    const arma::mat & ,
    arma::mat & newCentroids,
    arma::Col< size_t > & clusterCounts,
    MetricType & ,
    const size_t ) [inline], [static]
```

This function sets an empty cluster found during k-means to all DBL_MAX (i.e.

an invalid "dead" cluster).

Template Parameters

<i>MatType</i>	Type of data (arma::mat or arma::spmat).
----------------	--

Parameters

<i>data</i>	Dataset on which clustering is being performed.
<i>emptyCluster</i>	Index of cluster which is empty.
<i>oldCentroids</i>	Centroids of each cluster (one per column) at the start of the iteration.
<i>newCentroids</i>	Centroids of each cluster (one per column) at the end of the iteration.
<i>clusterCounts</i>	Number of points in each cluster.
<i>assignments</i>	Cluster assignments of each point.
<i>iteration</i>	Number of iteration.

Returns

Number of points changed (0).

Definition at line 49 of file kill_empty_clusters.hpp.

39.335.3.2 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serialize the empty cluster policy (nothing to do).

Definition at line 68 of file kill_empty_clusters.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ **kill_empty_clusters.hpp**

39.336 KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy, LloydStepType, MatType > Class Template Reference

This class implements K-Means clustering, using a variety of possible implementations of Lloyd's algorithm.

Public Member Functions

- **KMeans** (const size_t maxIterations=1000, const MetricType metric=MetricType(), const InitialPartitionPolicy partitioner=InitialPartitionPolicy(), const EmptyClusterPolicy emptyClusterAction=EmptyClusterPolicy())
Create a K-Means object and (optionally) set the parameters which K-Means will be run with.
- void **Cluster** (const MatType &data, const size_t clusters, arma::Row< size_t > &assignments, const bool initialGuess=false)
Perform k-means clustering on the data, returning a list of cluster assignments.
- void **Cluster** (const MatType &data, size_t clusters, arma::mat ¢roids, const bool initialGuess=false)
Perform k-means clustering on the data, returning the centroids of each cluster in the centroids matrix.
- void **Cluster** (const MatType &data, const size_t clusters, arma::Row< size_t > &assignments, arma::mat ¢roids, const bool initialAssignmentGuess=false, const bool initialCentroidGuess=false)
Perform k-means clustering on the data, returning a list of cluster assignments and also the centroids of each cluster.
- const EmptyClusterPolicy & **EmptyClusterAction** () const
Get the empty cluster policy.
- EmptyClusterPolicy & **EmptyClusterAction** ()
Modify the empty cluster policy.
- size_t **MaxIterations** () const
Get the maximum number of iterations.
- size_t & **MaxIterations** ()
Set the maximum number of iterations.
- const MetricType & **Metric** () const
Get the distance metric.
- MetricType & **Metric** ()

Modify the distance metric.

- `const InitialPartitionPolicy & Partitioner () const`

Get the initial partitioning policy.

- `InitialPartitionPolicy & Partitioner ()`

Modify the initial partitioning policy.

- `template<typename Archive >
void serialize (Archive &ar, const unsigned int version)`

Serialize the k-means object.

39.336.1 Detailed Description

```
template<typename MetricType = metric::EuclideanDistance, typename InitialPartitionPolicy = SampleInitialization, typename EmptyClusterPolicy = MaxVarianceNewCluster, template< class, class > class LloydStepType = NaiveKMeans, typename MatType = arma::mat>
class mlpack::kmeans::KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy, LloydStepType, MatType >
```

This class implements K-Means clustering, using a variety of possible implementations of Lloyd's algorithm.

Four template parameters can (optionally) be supplied: the distance metric to use, the policy for how to find the initial partition of the data, the actions to be taken when an empty cluster is encountered, and the implementation of a single Lloyd step to use.

A simple example of how to run K-Means clustering is shown below.

```
extern arma::mat data; // Dataset we want to run K-Means on.
arma::Row<size_t> assignments; // Cluster assignments.
arma::mat centroids; // Cluster centroids.

KMeans<> k; // Default options.
k.Cluster(data, 3, assignments, centroids); // 3 clusters.

// Cluster using the Manhattan distance, 100 iterations maximum, saving only
// the centroids.
KMeans<metric::ManhattanDistance> k(100);
k.Cluster(data, 6, centroids); // 6 clusters.
```

Template Parameters

<i>MetricType</i>	The distance metric to use for this KMeans (p. 1483); see metric::LMetric (p. 1569) for an example.
<i>InitialPartitionPolicy</i>	Initial partitioning policy; must implement a default constructor and either 'void Cluster(const arma::mat&, const size_t, arma::Row<size_t>&)' or 'void Cluster(const arma::mat&, const size_t, arma::mat&)'.
<i>EmptyClusterPolicy</i>	Policy for what to do on an empty cluster; must implement a default constructor and 'void EmptyCluster(const arma::mat& data, const size_t emptyCluster, const arma::mat& oldCentroids, arma::mat& newCentroids, arma::Col<size_t>& counts, MetricType& metric, const size_t iteration)'.
<i>LloydStepType</i>	Implementation of single Lloyd step to use.

See also

RandomPartition (p. 1500), **SampleInitialization** (p. 1506), **RefinedStart** (p. 1502), **AllowEmptyClusters** (p. 1464), **MaxVarianceNewCluster** (p. 1489), **NaiveKMeans** (p. 1491), **ElkanKMeans** (p. 1478)

Definition at line 73 of file kmeans.hpp.

39.336.2 Constructor & Destructor Documentation

39.336.2.1 KMeans()

```
KMeans (  
    const size_t maxIterations = 1000,  
    const MetricType metric = MetricType(),  
    const InitialPartitionPolicy partitioner = InitialPartitionPolicy(),  
    const EmptyClusterPolicy emptyClusterAction = EmptyClusterPolicy() )
```

Create a K-Means object and (optionally) set the parameters which K-Means will be run with.

Parameters

<i>maxIterations</i>	Maximum number of iterations allowed before giving up (0 is valid, but the algorithm may never terminate).
<i>metric</i>	Optional MetricType object; for when the metric has state it needs to store.
<i>partitioner</i>	Optional InitialPartitionPolicy object; for when a specially initialized partitioning policy is required.
<i>emptyClusterAction</i>	Optional EmptyClusterPolicy object; for when a specially initialized empty cluster policy is required.

39.336.3 Member Function Documentation

39.336.3.1 Cluster() [1/3]

```
void Cluster (  
    const MatType & data,  
    const size_t clusters,  
    arma::Row< size_t > & assignments,  
    const bool initialGuess = false )
```

Perform k-means clustering on the data, returning a list of cluster assignments.

Optionally, the vector of assignments can be set to an initial guess of the cluster assignments; to do this, set initialGuess to true.

Template Parameters

<i>MatType</i>	Type of matrix (arma::mat or arma::sp_mat).
----------------	---

Parameters

<i>data</i>	Dataset to cluster.
<i>clusters</i>	Number of clusters to compute.
<i>assignments</i>	Vector to store cluster assignments in.
<i>initialGuess</i>	If true, then it is assumed that assignments has a list of initial cluster assignments.

39.336.3.2 Cluster() [2/3]

```
void Cluster (
    const MatType & data,
    size_t clusters,
    arma::mat & centroids,
    const bool initialGuess = false )
```

Perform k-means clustering on the data, returning the centroids of each cluster in the centroids matrix.

Optionally, the initial centroids can be specified by filling the centroids matrix with the initial centroids and specifying initialGuess = true.

Template Parameters

<i>MatType</i>	Type of matrix (arma::mat or arma::sp_mat).
----------------	---

Parameters

<i>data</i>	Dataset to cluster.
<i>clusters</i>	Number of clusters to compute.
<i>centroids</i>	Matrix in which centroids are stored.
<i>initialGuess</i>	If true, then it is assumed that centroids contains the initial cluster centroids.

39.336.3.3 Cluster() [3/3]

```
void Cluster (
    const MatType & data,
    const size_t clusters,
```

```
arma::Row< size_t > & assignments,
arma::mat & centroids,
const bool initialAssignmentGuess = false,
const bool initialCentroidGuess = false )
```

Perform k-means clustering on the data, returning a list of cluster assignments and also the centroids of each cluster.

Optionally, the vector of assignments can be set to an initial guess of the cluster assignments; to do this, set `initialAssignmentGuess` to true. Another way to set initial cluster guesses is to fill the centroids matrix with the centroid guesses, and then set `initialCentroidGuess` to true. `initialAssignmentGuess` supersedes `initialCentroidGuess`, so if both are set to true, the assignments vector is used.

Template Parameters

<i>MatType</i>	Type of matrix (arma::mat or arma::sp_mat).
----------------	---

Parameters

<i>data</i>	Dataset to cluster.
<i>clusters</i>	Number of clusters to compute.
<i>assignments</i>	Vector to store cluster assignments in.
<i>centroids</i>	Matrix in which centroids are stored.
<i>initialAssignmentGuess</i>	If true, then it is assumed that assignments has a list of initial cluster assignments.
<i>initialCentroidGuess</i>	If true, then it is assumed that centroids contains the initial centroids of each cluster.

39.336.3.4 EmptyClusterAction() [1/2]

```
const EmptyClusterPolicy& EmptyClusterAction ( ) const [inline]
```

Get the empty cluster policy.

Definition at line 174 of file kmeans.hpp.

39.336.3.5 EmptyClusterAction() [2/2]

```
EmptyClusterPolicy& EmptyClusterAction ( ) [inline]
```

Modify the empty cluster policy.

Definition at line 177 of file kmeans.hpp.

References `KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy, LloydStepType, MatType >::serialize()`.

39.336.3.6 MaxIterations() [1/2]

```
size_t MaxIterations ( ) const [inline]
```

Get the maximum number of iterations.

Definition at line 159 of file kmeans.hpp.

39.336.3.7 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Set the maximum number of iterations.

Definition at line 161 of file kmeans.hpp.

39.336.3.8 Metric() [1/2]

```
const MetricType& Metric ( ) const [inline]
```

Get the distance metric.

Definition at line 164 of file kmeans.hpp.

39.336.3.9 Metric() [2/2]

```
MetricType& Metric ( ) [inline]
```

Modify the distance metric.

Definition at line 166 of file kmeans.hpp.

39.336.3.10 Partitioner() [1/2]

```
const InitialPartitionPolicy& Partitioner ( ) const [inline]
```

Get the initial partitioning policy.

Definition at line 169 of file kmeans.hpp.

39.336.3.11 Partitioner() [2/2]

```
InitialPartitionPolicy& Partitioner ( ) [inline]
```

Modify the initial partitioning policy.

Definition at line 171 of file kmeans.hpp.

39.336.3.12 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int version )
```

Serialize the k-means object.

Referenced by `KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy, LloydStepType, MatType >::EmptyClusterAction()`.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ **kmeans.hpp**

39.337 MaxVarianceNewCluster Class Reference

When an empty cluster is detected, this class takes the point furthest from the centroid of the cluster with maximum variance as a new cluster.

Public Member Functions

- **MaxVarianceNewCluster ()**

Default constructor required by EmptyClusterPolicy.

- `template<typename MetricType , typename MatType >`
void EmptyCluster (const MatType &data, const size_t emptyCluster, const arma::mat &oldCentroids, arma::mat &newCentroids, arma::Col< size_t > &clusterCounts, MetricType &metric, const size_t iteration)

Take the point furthest from the centroid of the cluster with maximum variance to be a new cluster.

- `template<typename Archive >`
void serialize (Archive &ar, const unsigned int version)

Serialize the object.

39.337.1 Detailed Description

When an empty cluster is detected, this class takes the point furthest from the centroid of the cluster with maximum variance as a new cluster.

Definition at line 26 of file `max_variance_new_cluster.hpp`.

39.337.2 Constructor & Destructor Documentation

39.337.2.1 MaxVarianceNewCluster()

```
MaxVarianceNewCluster ( ) [inline]
```

Default constructor required by EmptyClusterPolicy.

Definition at line 30 of file `max_variance_new_cluster.hpp`.

References `MaxVarianceNewCluster::EmptyCluster()`, and `MaxVarianceNewCluster::serialize()`.

39.337.3 Member Function Documentation

39.337.3.1 EmptyCluster()

```
void EmptyCluster (
    const MatType & data,
    const size_t emptyCluster,
    const arma::mat & oldCentroids,
    arma::mat & newCentroids,
    arma::Col< size_t > & clusterCounts,
    MetricType & metric,
    const size_t iteration )
```

Take the point furthest from the centroid of the cluster with maximum variance to be a new cluster.

Template Parameters

<i>MatType</i>	Type of data (arma::mat or arma::sp_mat).
----------------	---

Parameters

<i>data</i>	Dataset on which clustering is being performed.
<i>emptyCluster</i>	Index of cluster which is empty.
<i>oldCentroids</i>	Centroids of each cluster (one per column), at the start of the iteration.
<i>newCentroids</i>	Centroids of each cluster (one per column), at the end of the iteration. This will be modified!
<i>clusterCounts</i>	Number of points in each cluster.
<i>assignments</i>	Cluster assignments of each point.

Returns

Number of points changed.

Referenced by MaxVarianceNewCluster::MaxVarianceNewCluster().

39.337.3.2 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int version )
```

Serialize the object.

Referenced by MaxVarianceNewCluster::MaxVarianceNewCluster().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ **max_variance_new_cluster.hpp**

39.338 NaiveKMeans< MetricType, MatType > Class Template Reference

This is an implementation of a single iteration of Lloyd's algorithm for k-means.

Public Member Functions

- **NaiveKMeans** (const MatType &dataset, MetricType &metric)
*Construct the **NaiveKMeans** (p. 1491) object with the given dataset and metric.*
- size_t **DistanceCalculations** () const
- double **Iterate** (const arma::mat ¢roids, arma::mat &newCentroids, arma::Col< size_t > &counts)
Run a single iteration of the Lloyd algorithm, updating the given centroids into the newCentroids matrix.

39.338.1 Detailed Description

```
template<typename MetricType, typename MatType>
class mlpack::kmeans::NaiveKMeans< MetricType, MatType >
```

This is an implementation of a single iteration of Lloyd's algorithm for k-means.

If your intention is to run the full k-means algorithm, you are looking for the **mlpack::kmeans::KMeans** (p. 1483) class instead of this one. This class is used by **KMeans** (p. 1483) as the actual implementation of the Lloyd iteration.

Parameters

<i>MetricType</i>	Type of metric used with this implementation.
<i>MatType</i>	Matrix type (arma::mat or arma::sp_mat).

Definition at line 32 of file naive_kmeans.hpp.

39.338.2 Constructor & Destructor Documentation

39.338.2.1 NaiveKMeans()

```
NaiveKMeans (
    const MatType & dataset,
    MetricType & metric )
```

Construct the **NaiveKMeans** (p. 1491) object with the given dataset and metric.

Parameters

<i>dataset</i>	Dataset.
<i>metric</i>	Instantiated metric.

39.338.3 Member Function Documentation

39.338.3.1 DistanceCalculations()

```
size_t DistanceCalculations ( ) const [inline]
```

Definition at line 57 of file naive_kmeans.hpp.

39.338.3.2 Iterate()

```
double Iterate (
    const arma::mat & centroids,
    arma::mat & newCentroids,
    arma::Col< size_t > & counts )
```

Run a single iteration of the Lloyd algorithm, updating the given centroids into the newCentroids matrix.

If any cluster is empty (that is, if any cluster has no points assigned to it), then the centroid associated with that cluster may be filled with invalid data (it will be corrected later).

Parameters

<i>centroids</i>	Current cluster centroids.
<i>newCentroids</i>	New cluster centroids.
<i>counts</i>	Number of points in each cluster at the end of the iteration.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ **naive_kmeans.hpp**

39.339 PellegMooreKMeans< MetricType, MatType > Class Template Reference

An implementation of Pelleg-Moore's 'blacklist' algorithm for k-means clustering.

Public Types

- typedef **tree::KDTree**< MetricType, **PellegMooreKMeansStatistic**, MatType > **TreeType**
Convenience typedef for the tree.

Public Member Functions

- **PellegMooreKMeans** (const MatType &dataset, MetricType &metric)
*Construct the **PellegMooreKMeans** (p. 1493) object, which must construct a tree.*
- **~PellegMooreKMeans** ()
*Delete the tree constructed by the **PellegMooreKMeans** (p. 1493) object.*
- size_t **DistanceCalculations** () const
Return the number of distance calculations.
- size_t & **DistanceCalculations** ()
Modify the number of distance calculations.
- double **Iterate** (const arma::mat ¢roids, arma::mat &newCentroids, arma::Col< size_t > &counts)
Run a single iteration of the Pelleg-Moore blacklist algorithm, updating the given centroids into the newCentroids matrix.

39.339.1 Detailed Description

```
template<typename MetricType, typename MatType>
class mlpack::kmeans::PellegMooreKMeans< MetricType, MatType >
```

An implementation of Pelleg-Moore's 'blacklist' algorithm for k-means clustering.

This algorithm builds a kd-tree on the data points and traverses it in order to determine the closest clusters to each point.

For more information on the algorithm, see

```
@inproceedings{pelleg1999accelerating,
  title={Accelerating exact k-means algorithms with geometric reasoning},
  author={Pelleg, Dan and Moore, Andrew W.},
  booktitle={Proceedings of the Fifth ACM SIGKDD International Conference
    on Knowledge Discovery and Data Mining (KDD '99)},
  pages={277--281},
  year={1999},
  organization={ACM}
}
```

Definition at line 42 of file pelleg_moore_kmeans.hpp.

39.339.2 Member Typedef Documentation

39.339.2.1 TreeType

```
typedef tree::KDTree<MetricType, PellegMooreKMeansStatistic, MatType> TreeType
```

Convenience typedef for the tree.

Definition at line 74 of file pelleg_moore_kmeans.hpp.

39.339.3 Constructor & Destructor Documentation

39.339.3.1 PellegMooreKMeans()

```
PellegMooreKMeans (  
    const MatType & dataset,  
    MetricType & metric )
```

Construct the **PellegMooreKMeans** (p. 1493) object, which must construct a tree.

39.339.3.2 ~PellegMooreKMeans()

```
~ PellegMooreKMeans ( )
```

Delete the tree constructed by the **PellegMooreKMeans** (p. 1493) object.

39.339.4 Member Function Documentation

39.339.4.1 DistanceCalculations() [1/2]

```
size_t DistanceCalculations ( ) const [inline]
```

Return the number of distance calculations.

Definition at line 68 of file pelleg_moore_kmeans.hpp.

39.339.4.2 DistanceCalculations() [2/2]

```
size_t& DistanceCalculations ( ) [inline]
```

Modify the number of distance calculations.

Definition at line 70 of file pelleg_moore_kmeans.hpp.

39.339.4.3 Iterate()

```
double Iterate (
    const arma::mat & centroids,
    arma::mat & newCentroids,
    arma::Col< size_t > & counts )
```

Run a single iteration of the Pelleg-Moore blacklist algorithm, updating the given centroids into the newCentroids matrix.

Parameters

<i>centroids</i>	Current cluster centroids.
<i>newCentroids</i>	New cluster centroids.
<i>counts</i>	Current counts, to be overwritten with new counts.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ **pelleg_moore_kmeans.hpp**

39.340 PellegMooreKMeansRules< MetricType, TreeType > Class Template Reference

The rules class for the single-tree Pelleg-Moore kd-tree traversal for k-means clustering.

Public Member Functions

- **PellegMooreKMeansRules** (const typename TreeType::Mat &dataset, const arma::mat ¢roids, arma::mat &newCentroids, arma::Col< size_t > &counts, MetricType &metric)
*Create the **PellegMooreKMeansRules** (p. 1495) object.*
- double **BaseCase** (const size_t queryIndex, const size_t referenceIndex)
*The **BaseCase()** (p. 1496) function for this single-tree algorithm does nothing.*
- size_t **DistanceCalculations** () const
Get the number of distance calculations that have been performed.
- size_t & **DistanceCalculations** ()
Modify the number of distance calculations that have been performed.
- double **Rescore** (const size_t queryIndex, TreeType &referenceNode, const double oldScore)
Rescore to determine if a node can be pruned.
- double **Score** (const size_t queryIndex, TreeType &referenceNode)
Determine if a cluster can be pruned, and if not, perform point-to-cluster comparisons.

39.340.1 Detailed Description

```
template<typename MetricType, typename TreeType>
class mlpack::kmeans::PellegMooreKMeansRules< MetricType, TreeType >
```

The rules class for the single-tree Pelleg-Moore kd-tree traversal for k-means clustering.

Although `TreeType` is a free template parameter, this particular implementation is specialized to trees with hyper-rectangle bounds due to the pruning rule used to determine if one cluster dominates a node with respect to another cluster.

Our implementation here abuses the single-tree algorithm abstractions a little bit. Instead of doing a traversal for a particular query point, in this case we consider all clusters at once—so the query point is entirely ignored during in **BaseCase()** (p. 1496) and **Score()** (p. 1498).

Definition at line 33 of file `pelleg_moore_kmeans_rules.hpp`.

39.340.2 Constructor & Destructor Documentation

39.340.2.1 PellegMooreKMeansRules()

```
PellegMooreKMeansRules (
    const typename TreeType::Mat & dataset,
    const arma::mat & centroids,
    arma::mat & newCentroids,
    arma::Col< size_t > & counts,
    MetricType & metric )
```

Create the **PellegMooreKMeansRules** (p. 1495) object.

Parameters

<i>dataset</i>	The dataset that the tree is built on.
<i>centroids</i>	The current centroids.
<i>newCentroids</i>	New centroids after this iteration (output).
<i>counts</i>	Current cluster counts, to be replaced with new cluster counts.
<i>metric</i>	Instantiated metric.

39.340.3 Member Function Documentation

39.340.3.1 BaseCase()

```
double BaseCase (
    const size_t queryIndex,
    const size_t referenceIndex )
```

The **BaseCase()** (p. 1496) function for this single-tree algorithm does nothing.

Instead, point-to-cluster comparisons are handled as necessary in **Score()** (p. 1498).

Parameters

<i>queryIndex</i>	Index of query point (fake, will be ignored).
<i>referenceIndex</i>	Index of reference point.

39.340.3.2 DistanceCalculations() [1/2]

```
size_t DistanceCalculations ( ) const [inline]
```

Get the number of distance calculations that have been performed.

Definition at line 84 of file pelleg_moore_kmeans_rules.hpp.

39.340.3.3 DistanceCalculations() [2/2]

```
size_t& DistanceCalculations ( ) [inline]
```

Modify the number of distance calculations that have been performed.

Definition at line 86 of file pelleg_moore_kmeans_rules.hpp.

39.340.3.4 Rescore()

```
double Rescore (
    const size_t queryIndex,
    TreeType & referenceNode,
    const double oldScore )
```

Rescore to determine if a node can be pruned.

In this case, a node can never be pruned during rescoring, so this just returns oldScore.

Parameters

<i>queryIndex</i>	Index of query point (fake, will be ignored).
<i>referenceNode</i>	Node containing points in the dataset.
<i>oldScore</i>	Resulting score from Score() (p. 1498).

39.340.3.5 Score()

```
double Score (
    const size_t queryIndex,
    TreeType & referenceNode )
```

Determine if a cluster can be pruned, and if not, perform point-to-cluster comparisons.

The point-to-cluster comparisons are performed here and not in **BaseCase()** (p. 1496) because of the complexity of managing the blacklist.

Parameters

<i>queryIndex</i>	Index of query point (fake, will be ignored).
<i>referenceNode</i>	Node containing points in the dataset.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ **pelleg_moore_kmeans_rules.hpp**

39.341 PellegMooreKMeansStatistic Class Reference

A statistic for trees which holds the blacklist for Pelleg-Moore k-means clustering (which represents the clusters that cannot possibly own any points in a node).

Public Member Functions

- **PellegMooreKMeansStatistic** ()
Initialize the statistic without a node (this does nothing).
- template<typename TreeType >
PellegMooreKMeansStatistic (TreeType &node)
Initialize the statistic for a node; this calculates the centroid and caches it.
- const arma::uvec & **Blacklist** () const
Get the cluster blacklist.
- arma::uvec & **Blacklist** ()

Modify the cluster blacklist.

- `const arma::vec & Centroid () const`

Get the node's centroid.

- `arma::vec & Centroid ()`

Modify the node's centroid (be careful!).

39.341.1 Detailed Description

A statistic for trees which holds the blacklist for Pelleg-Moore k-means clustering (which represents the clusters that cannot possibly own any points in a node).

Definition at line 24 of file `pelleg_moore_kmeans_statistic.hpp`.

39.341.2 Constructor & Destructor Documentation

39.341.2.1 PellegMooreKMeansStatistic() [1/2]

```
PellegMooreKMeansStatistic ( ) [inline]
```

Initialize the statistic without a node (this does nothing).

Definition at line 28 of file `pelleg_moore_kmeans_statistic.hpp`.

39.341.2.2 PellegMooreKMeansStatistic() [2/2]

```
PellegMooreKMeansStatistic (
    TreeType & node ) [inline]
```

Initialize the statistic for a node; this calculates the centroid and caches it.

Definition at line 33 of file `pelleg_moore_kmeans_statistic.hpp`.

39.341.3 Member Function Documentation

39.341.3.1 Blacklist() [1/2]

```
const arma::uvec& Blacklist ( ) const [inline]
```

Get the cluster blacklist.

Definition at line 57 of file `pelleg_moore_kmeans_statistic.hpp`.

39.341.3.2 Blacklist() [2/2]

```
arma::uvec& Blacklist ( ) [inline]
```

Modify the cluster blacklist.

Definition at line 59 of file `pelleg_moore_kmeans_statistic.hpp`.

39.341.3.3 Centroid() [1/2]

```
const arma::vec& Centroid ( ) const [inline]
```

Get the node's centroid.

Definition at line 62 of file `pelleg_moore_kmeans_statistic.hpp`.

39.341.3.4 Centroid() [2/2]

```
arma::vec& Centroid ( ) [inline]
```

Modify the node's centroid (be careful!).

Definition at line 64 of file `pelleg_moore_kmeans_statistic.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/pelleg_moore_kmeans_statistic.hpp`

39.342 RandomPartition Class Reference

A very simple partitioner which partitions the data randomly into the number of desired clusters.

Public Member Functions

- **RandomPartition** ()
Empty constructor, required by the InitialPartitionPolicy policy.
- template<typename Archive >
void **serialize** (Archive &, const unsigned int)
Serialize the partitioner (nothing to do).

Static Public Member Functions

- template<typename MatType >
static void **Cluster** (const MatType &data, const size_t clusters, arma::Row< size_t > &assignments)
Partition the given dataset into the given number of clusters.

39.342.1 Detailed Description

A very simple partitioner which partitions the data randomly into the number of desired clusters.

It has no parameters, and so an instance of the class is not even necessary.

Definition at line 26 of file random_partition.hpp.

39.342.2 Constructor & Destructor Documentation

39.342.2.1 RandomPartition()

```
RandomPartition ( ) [inline]
```

Empty constructor, required by the InitialPartitionPolicy policy.

Definition at line 30 of file random_partition.hpp.

39.342.3 Member Function Documentation

39.342.3.1 Cluster()

```
static void Cluster (  
    const MatType & data,  
    const size_t clusters,  
    arma::Row< size_t > & assignments ) [inline], [static]
```

Partition the given dataset into the given number of clusters.

Assignments are random, and the number of points in each cluster should be equal (or approximately equal).

Template Parameters

<i>MatType</i>	Type of data (arma::mat or arma::sp_mat).
----------------	---

Parameters

<i>data</i>	Dataset to partition.
<i>clusters</i>	Number of clusters to split dataset into.
<i>assignments</i>	Vector to store cluster assignments into. Values will be between 0 and (clusters - 1).

Definition at line 44 of file random_partition.hpp.

39.342.3.2 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serialize the partitioner (nothing to do).

Definition at line 55 of file random_partition.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ **random_partition.hpp**

39.343 RefinedStart Class Reference

A refined approach for choosing initial points for k-means clustering.

Public Member Functions

- **RefinedStart** (const size_t samplings=100, const double percentage=0.02)
*Create the **RefinedStart** (p. 1502) object, optionally specifying parameters for the number of samplings to perform and the percentage of the dataset to use in each sampling.*
- template<typename MatType >
void **Cluster** (const MatType &data, const size_t clusters, arma::mat ¢roids) const
Partition the given dataset into the given number of clusters according to the random sampling scheme outlined in Bradley and Fayyad's paper, and return centroids.
- template<typename MatType >
void **Cluster** (const MatType &data, const size_t clusters, arma::Row< size_t > &assignments) const

Partition the given dataset into the given number of clusters according to the random sampling scheme outlined in Bradley and Fayyad's paper, and return point assignments.

- double **Percentage** () const
Get the percentage of the data used by each subsampling.
- double & **Percentage** ()
Modify the percentage of the data used by each subsampling.
- size_t **Samplings** () const
Get the number of samplings that will be performed.
- size_t & **Samplings** ()
Modify the number of samplings that will be performed.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the object.

39.343.1 Detailed Description

A refined approach for choosing initial points for k-means clustering.

This approach runs k-means several times on random subsets of the data, and then clusters those solutions to select refined initial cluster assignments. It is an implementation of the following paper:

```
{bradley1998refining, title={Refining initial points for k-means clustering}, author={Bradley, Paul S and Fayyad, Usama M}, booktitle={Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)}, volume={66}, year={1998} }
```

Definition at line 37 of file refined_start.hpp.

39.343.2 Constructor & Destructor Documentation

39.343.2.1 RefinedStart()

```
RefinedStart (  
    const size_t samplings = 100,  
    const double percentage = 0.02 ) [inline]
```

Create the **RefinedStart** (p. 1502) object, optionally specifying parameters for the number of samplings to perform and the percentage of the dataset to use in each sampling.

Definition at line 45 of file refined_start.hpp.

References `RefinedStart::Cluster()`.

39.343.3 Member Function Documentation

39.343.3.1 Cluster() [1/2]

```
void Cluster (
    const MatType & data,
    const size_t clusters,
    arma::mat & centroids ) const
```

Partition the given dataset into the given number of clusters according to the random sampling scheme outlined in Bradley and Fayyad's paper, and return centroids.

Template Parameters

<i>MatType</i>	Type of data (arma::mat or arma::sp_mat).
----------------	---

Parameters

<i>data</i>	Dataset to partition.
<i>clusters</i>	Number of clusters to split dataset into.
<i>centroids</i>	Matrix to store centroids into.

Referenced by RefinedStart::RefinedStart().

39.343.3.2 Cluster() [2/2]

```
void Cluster (
    const MatType & data,
    const size_t clusters,
    arma::Row< size_t > & assignments ) const
```

Partition the given dataset into the given number of clusters according to the random sampling scheme outlined in Bradley and Fayyad's paper, and return point assignments.

Template Parameters

<i>MatType</i>	Type of data (arma::mat or arma::sp_mat).
----------------	---

Parameters

<i>data</i>	Dataset to partition.
<i>clusters</i>	Number of clusters to split dataset into.
<i>assignments</i>	Vector to store cluster assignments into. Values will be between 0 and (clusters - 1).

39.343.3.3 Percentage() [1/2]

```
double Percentage ( ) const [inline]
```

Get the percentage of the data used by each subsampling.

Definition at line 86 of file `refined_start.hpp`.

39.343.3.4 Percentage() [2/2]

```
double& Percentage ( ) [inline]
```

Modify the percentage of the data used by each subsampling.

Definition at line 88 of file `refined_start.hpp`.

39.343.3.5 Samplings() [1/2]

```
size_t Samplings ( ) const [inline]
```

Get the number of samplings that will be performed.

Definition at line 81 of file `refined_start.hpp`.

39.343.3.6 Samplings() [2/2]

```
size_t& Samplings ( ) [inline]
```

Modify the number of samplings that will be performed.

Definition at line 83 of file `refined_start.hpp`.

39.343.3.7 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the object.

Definition at line 92 of file `refined_start.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ refined_start.hpp`

39.344 SampleInitialization Class Reference

Public Member Functions

- **SampleInitialization** ()
Empty constructor, required by the InitialPartitionPolicy type definition.

Static Public Member Functions

- `template<typename MatType >`
`static void Cluster (const MatType &data, const size_t clusters, arma::mat ¢roids)`
Initialize the centroids matrix by randomly sampling points from the data matrix.

39.344.1 Detailed Description

Definition at line 23 of file `sample_initialization.hpp`.

39.344.2 Constructor & Destructor Documentation

39.344.2.1 SampleInitialization()

```
SampleInitialization ( ) [inline]
```

Empty constructor, required by the InitialPartitionPolicy type definition.

Definition at line 27 of file `sample_initialization.hpp`.

39.344.3 Member Function Documentation

39.344.3.1 Cluster()

```
static void Cluster (
    const MatType & data,
    const size_t clusters,
    arma::mat & centroids ) [inline], [static]
```

Initialize the centroids matrix by randomly sampling points from the data matrix.

Parameters

<i>data</i>	Dataset.
<i>clusters</i>	Number of clusters.
<i>centroids</i>	Matrix to put initial centroids into.

Definition at line 38 of file sample_initialization.hpp.

References `mlpack::math::RandInt()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/ sample_initialization.hpp`

39.345 KernelPCA< KernelType, KernelRule > Class Template Reference

This class performs kernel principal components analysis (Kernel PCA), for a given kernel.

Public Member Functions

- **KernelPCA** (const KernelType kernel=KernelType(), const bool centerTransformedData=false)
*Construct the **KernelPCA** (p. 1507) object, optionally passing a kernel.*
- void **Apply** (const arma::mat &data, arma::mat &transformedData, arma::vec &eigval, arma::mat &eigvec, const size_t newDimension)
Apply Kernel Principal Components Analysis to the provided data set.
- void **Apply** (const arma::mat &data, arma::mat &transformedData, arma::vec &eigval, arma::mat &eigvec)
Apply Kernel Principal Components Analysis to the provided data set.
- void **Apply** (const arma::mat &data, arma::mat &transformedData, arma::vec &eigval)
Apply Kernel Principal Component Analysis to the provided data set.
- void **Apply** (arma::mat &data, const size_t newDimension)
Apply dimensionality reduction using Kernel Principal Component Analysis to the provided data set.
- bool **CenterTransformedData** () const
Return whether or not the transformed data is centered.
- bool & **CenterTransformedData** ()
Return whether or not the transformed data is centered.
- const KernelType & **Kernel** () const
Get the kernel.
- KernelType & **Kernel** ()
Modify the kernel.

39.345.1 Detailed Description

```
template<typename KernelType, typename KernelRule = NaiveKernelRule<KernelType>>
class mlpack::kpca::KernelPCA< KernelType, KernelRule >
```

This class performs kernel principal components analysis (Kernel PCA), for a given kernel.

This is a standard machine learning technique and is well-documented on the Internet and in standard texts. It is often used as a dimensionality reduction technique, and can also be useful in mapping linearly inseparable classes of points to different spaces where they are linearly separable.

The performance of the method is highly dependent on the kernel choice. There are numerous available kernels in the **mlpack::kernel** (p. 401) namespace (see files in `mlpack/core/kernels/`) and it is easy to write your own; see other implementations for examples.

Definition at line 40 of file `kernel_pca.hpp`.

39.345.2 Constructor & Destructor Documentation

39.345.2.1 KernelPCA()

```
KernelPCA (
    const KernelType kernel = KernelType(),
    const bool centerTransformedData = false )
```

Construct the **KernelPCA** (p. 1507) object, optionally passing a kernel.

Optionally, the transformed data can be centered about the origin; to do this, pass 'true' for `centerTransformedData`. This will take slightly longer (but not much).

Parameters

<i>kernel</i>	Kernel to be used for computation.
<i>centerTransformedData</i>	Center transformed data.

39.345.3 Member Function Documentation

39.345.3.1 Apply() [1/4]

```
void Apply (
    const arma::mat & data,
```

```

    arma::mat & transformedData,
    arma::vec & eigval,
    arma::mat & eigvec,
    const size_t newDimension )

```

Apply Kernel Principal Components Analysis to the provided data set.

Parameters

<i>data</i>	Data matrix.
<i>transformedData</i>	Matrix to output results into.
<i>eigval</i>	KPCA eigenvalues will be written to this vector.
<i>eigvec</i>	KPCA eigenvectors will be written to this matrix.
<i>newDimension</i>	New dimension for the dataset.

39.345.3.2 Apply() [2/4]

```

void Apply (
    const arma::mat & data,
    arma::mat & transformedData,
    arma::vec & eigval,
    arma::mat & eigvec )

```

Apply Kernel Principal Components Analysis to the provided data set.

Parameters

<i>data</i>	Data matrix.
<i>transformedData</i>	Matrix to output results into.
<i>eigval</i>	KPCA eigenvalues will be written to this vector.
<i>eigvec</i>	KPCA eigenvectors will be written to this matrix.

39.345.3.3 Apply() [3/4]

```

void Apply (
    const arma::mat & data,
    arma::mat & transformedData,
    arma::vec & eigval )

```

Apply Kernel Principal Component Analysis to the provided data set.

Parameters

<i>data</i>	Data matrix.
<i>transformedData</i>	Matrix to output results into.
<i>eigval</i>	KPCA eigenvalues will be written to this vector.

39.345.3.4 Apply() [4/4]

```
void Apply (
    arma::mat & data,
    const size_t newDimension )
```

Apply dimensionality reduction using Kernel Principal Component Analysis to the provided data set.

The data matrix will be modified in-place. Note that the dimension can be larger than the existing dimension because KPCA works on the kernel matrix, not the covariance matrix. This means the new dimension can be as large as the number of points (columns) in the dataset. Note that if you specify newDimension to be larger than the current dimension of the data (the number of rows), then it's not really "dimensionality reduction"...

Parameters

<i>data</i>	Data matrix.
<i>newDimension</i>	New dimension for the dataset.

39.345.3.5 CenterTransformedData() [1/2]

```
bool CenterTransformedData ( ) const [inline]
```

Return whether or not the transformed data is centered.

Definition at line 115 of file kernel_pca.hpp.

39.345.3.6 CenterTransformedData() [2/2]

```
bool& CenterTransformedData ( ) [inline]
```

Return whether or not the transformed data is centered.

Definition at line 117 of file kernel_pca.hpp.

39.345.3.7 Kernel() [1/2]

```
const KernelType& Kernel ( ) const [inline]
```

Get the kernel.

Definition at line 110 of file kernel_pca.hpp.

39.345.3.8 Kernel() [2/2]

```
KernelType& Kernel ( ) [inline]
```

Modify the kernel.

Definition at line 112 of file kernel_pca.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kernel_pca/ **kernel_pca.hpp**

39.346 NaiveKernelRule< KernelType > Class Template Reference**Static Public Member Functions**

- static void **ApplyKernelMatrix** (const arma::mat &data, arma::mat &transformedData, arma::vec &eigval, arma::mat &eigvec, const size_t, KernelType kernel=KernelType())

Construct the exact kernel matrix.

39.346.1 Detailed Description

```
template<typename KernelType>
class mlpack::kpca::NaiveKernelRule< KernelType >
```

Definition at line 22 of file naive_method.hpp.

39.346.2 Member Function Documentation**39.346.2.1 ApplyKernelMatrix()**

```
static void ApplyKernelMatrix (
    const arma::mat & data,
    arma::mat & transformedData,
    arma::vec & eigval,
    arma::mat & eigvec,
    const size_t ,
    KernelType kernel = KernelType() ) [inline], [static]
```

Construct the exact kernel matrix.

Parameters

<i>data</i>	Input data points.
<i>transformedData</i>	Matrix to output results into.
<i>eigval</i>	KPCA eigenvalues will be written to this vector.
<i>eigvec</i>	KPCA eigenvectors will be written to this matrix.
<i>rank</i>	Rank to be used for matrix approximation.
<i>kernel</i>	Kernel to be used for computation.

Definition at line 35 of file `naive_method.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kernel_pca/kernel_rules/ naive_method.hpp`

39.347 NystroemKernelRule< KernelType, PointSelectionPolicy > Class Template Reference

Static Public Member Functions

- static void **ApplyKernelMatrix** (const arma::mat &data, arma::mat &transformedData, arma::vec &eigval, arma::mat &eigvec, const size_t rank, KernelType kernel=KernelType())
Construct the kernel matrix approximation using the nystroem method.

39.347.1 Detailed Description

```
template<typename KernelType, typename PointSelectionPolicy = kernel::KMeansSelection<>>
class mlpack::kcca::NystroemKernelRule< KernelType, PointSelectionPolicy >
```

Definition at line 27 of file `nystroem_method.hpp`.

39.347.2 Member Function Documentation

39.347.2.1 ApplyKernelMatrix()

```
static void ApplyKernelMatrix (
    const arma::mat & data,
    arma::mat & transformedData,
    arma::vec & eigval,
    arma::mat & eigvec,
    const size_t rank,
    KernelType kernel = KernelType() ) [inline], [static]
```

Construct the kernel matrix approximation using the nystroem method.

Parameters

<i>data</i>	Input data points.
<i>transformedData</i>	Matrix to output results into.
<i>eigval</i>	KPCA eigenvalues will be written to this vector.
<i>eigvec</i>	KPCA eigenvectors will be written to this matrix.
<i>rank</i>	Rank to be used for matrix approximation.
<i>kernel</i>	Kernel to be used for computation.

Definition at line 40 of file nystroem_method.hpp.

References NystroemMethod< KernelType, PointSelectionPolicy >::Apply(), and mlpack::math::Center().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kernel_pca/kernel_rules/ **nystroem_method.hpp**

39.348 LocalCoordinateCoding Class Reference

An implementation of Local Coordinate Coding (LCC) that codes data which approximately lives on a manifold using a variation of l1-norm regularized sparse coding; in LCC, the penalty on the absolute value of each point's coefficient for each atom is weighted by the squared distance of that point to that atom.

Public Member Functions

- template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer>
LocalCoordinateCoding (const arma::mat &data, const size_t atoms, const double lambda, const size_t maxIterations=0, const double tolerance=0.01, const DictionaryInitializer &initializer=DictionaryInitializer())
*Set the parameters to **LocalCoordinateCoding** (p. 1513), and train the dictionary.*
- **LocalCoordinateCoding** (const size_t atoms=0, const double lambda=0.0, const size_t maxIterations=0, const double tolerance=0.01)
*Set the parameters to **LocalCoordinateCoding** (p. 1513).*
- size_t **Atoms** () const
Get the number of atoms.
- size_t & **Atoms** ()
Modify the number of atoms.
- const arma::mat & **Dictionary** () const
Accessor for dictionary.
- arma::mat & **Dictionary** ()
Mutator for dictionary.
- void **Encode** (const arma::mat &data, arma::mat &codes)
Code each point via distance-weighted LARS.
- double **Lambda** () const
Get the L1 regularization parameter.
- double & **Lambda** ()

- Modify the L1 regularization parameter.
 - `size_t MaxIterations () const`
 Get the maximum number of iterations.
 - `size_t & MaxIterations ()`
 Modify the maximum number of iterations.
 - `double Objective (const arma::mat &data, const arma::mat &codes, const arma::uvec &adjacencies) const`
 Compute objective function given the list of adjacencies.
 - `void OptimizeDictionary (const arma::mat &data, const arma::mat &codes, const arma::uvec &adjacencies)`
 Learn dictionary by solving linear system.
 - `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`
 Serialize the model.
 - `double Tolerance () const`
 Get the objective tolerance.
 - `double & Tolerance ()`
 Modify the objective tolerance.
 - `template<typename DictionaryInitializer = sparse_coding::DataDependentRandomInitializer>`
`double Train (const arma::mat &data, const DictionaryInitializer &initializer=DictionaryInitializer())`
 Run local coordinate coding.

39.348.1 Detailed Description

An implementation of Local Coordinate Coding (LCC) that codes data which approximately lives on a manifold using a variation of l1-norm regularized sparse coding; in LCC, the penalty on the absolute value of each point's coefficient for each atom is weighted by the squared distance of that point to that atom.

Let d be the number of dimensions in the original space, m the number of training points, and k the number of atoms in the dictionary (the dimension of the learned feature space). The training data X is a d -by- m matrix where each column is a point and each row is a dimension. The dictionary D is a d -by- k matrix, and the sparse codes matrix Z is a k -by- m matrix. This program seeks to minimize the objective: $\min_{\{D,Z\}} \|X - DZ\|_{\text{Fro}}^2$

- $\lambda \sum_{i=1}^m \sum_{j=1}^k \text{dist}(X_i, D_j)^2 Z_{ij}$ where $\lambda > 0$.

This problem is solved by an algorithm that alternates between a dictionary learning step and a sparse coding step. The dictionary learning step updates the dictionary D by solving a linear system (note that the objective is a positive definite quadratic program). The sparse coding step involves solving a large number of weighted l1-norm regularized linear regression problems; this can be done efficiently using LARS, an algorithm that can solve the LASSO (paper below).

The papers are listed below.

```
@incollection{NIPS2009_0719,
  title = {Nonlinear Learning using Local Coordinate Coding},
  author = {Kai Yu and Tong Zhang and Yihong Gong},
  booktitle = {Advances in Neural Information Processing Systems 22},
  editor = {Y. Bengio and D. Schuurmans and J. Lafferty and C. K. I. Williams
    and A. Culotta},
  pages = {2223--2231},
  year = {2009}
}
```

```
@article{efron2004least,
  title={Least angle regression},
  author={Efron, B. and Hastie, T. and Johnstone, I. and Tibshirani, R.},
  journal={The Annals of statistics},
  volume={32},
  number={2},
  pages={407--499},
  year={2004},
  publisher={Institute of Mathematical Statistics}
}
```

Definition at line 79 of file lcc.hpp.

39.348.2 Constructor & Destructor Documentation

39.348.2.1 LocalCoordinateCoding() [1/2]

```
LocalCoordinateCoding (
    const arma::mat & data,
    const size_t atoms,
    const double lambda,
    const size_t maxIterations = 0,
    const double tolerance = 0.01,
    const DictionaryInitializer & initializer = DictionaryInitializer() )
```

Set the parameters to **LocalCoordinateCoding** (p. 1513), and train the dictionary.

This constructor will also initialize the dictionary using the given DictionaryInitializer before training.

If you want to initialize the dictionary to a custom matrix, consider either writing your own DictionaryInitializer class (with void Initialize(const arma::mat& data, arma::mat& dictionary) function), or call the constructor that does not take a data matrix, then call **Dictionary()** (p. 1516) to set the dictionary matrix to a matrix of your choosing, and then call **Train()** (p. 1519) with **sparse_coding::NothingInitializer** (p. 1914) (i.e. Train<sparse_coding::NothingInitializer>(data)).

Parameters

<i>data</i>	Data matrix.
<i>atoms</i>	Number of atoms in dictionary.
<i>lambda</i>	Regularization parameter for weighted l1-norm penalty.
<i>maxIterations</i>	Maximum number of iterations for training (0 runs until convergence).
<i>tolerance</i>	Tolerance for the objective function.

39.348.2.2 LocalCoordinateCoding() [2/2]

```
LocalCoordinateCoding (
    const size_t atoms = 0,
```

```
const double lambda = 0.0,
const size_t maxIterations = 0,
const double tolerance = 0.01 )
```

Set the parameters to **LocalCoordinateCoding** (p. 1513).

This constructor will not train the model, and a subsequent call to **Train()** (p. 1519) will be required before the model can encode points with **Encode()** (p. 1517). The default values for atoms and lambda should be changed if you intend to train the model!

Parameters

<i>atoms</i>	Number of atoms in dictionary.
<i>lambda</i>	Regularization parameter for weighted l1-norm penalty.
<i>maxIterations</i>	Maximum number of iterations for training (0 runs until convergence).
<i>tolerance</i>	Tolerance for the objective function.

39.348.3 Member Function Documentation

39.348.3.1 Atoms() [1/2]

```
size_t Atoms ( ) const [inline]
```

Get the number of atoms.

Definition at line 175 of file lcc.hpp.

39.348.3.2 Atoms() [2/2]

```
size_t& Atoms ( ) [inline]
```

Modify the number of atoms.

Definition at line 177 of file lcc.hpp.

39.348.3.3 Dictionary() [1/2]

```
const arma::mat& Dictionary ( ) const [inline]
```

Accessor for dictionary.

Definition at line 180 of file lcc.hpp.

39.348.3.4 Dictionary() [2/2]

```
arma::mat& Dictionary ( ) [inline]
```

Mutator for dictionary.

Definition at line 182 of file lcc.hpp.

39.348.3.5 Encode()

```
void Encode (
    const arma::mat & data,
    arma::mat & codes )
```

Code each point via distance-weighted LARS.

Parameters

<i>data</i>	Matrix containing points to encode.
<i>codes</i>	Output matrix to store codes in.

39.348.3.6 Lambda() [1/2]

```
double Lambda ( ) const [inline]
```

Get the L1 regularization parameter.

Definition at line 185 of file lcc.hpp.

39.348.3.7 Lambda() [2/2]

```
double& Lambda ( ) [inline]
```

Modify the L1 regularization parameter.

Definition at line 187 of file lcc.hpp.

39.348.3.8 MaxIterations() [1/2]

```
size_t MaxIterations ( ) const [inline]
```

Get the maximum number of iterations.

Definition at line 190 of file lcc.hpp.

39.348.3.9 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Modify the maximum number of iterations.

Definition at line 192 of file lcc.hpp.

39.348.3.10 Objective()

```
double Objective (
    const arma::mat & data,
    const arma::mat & codes,
    const arma::uvec & adjacencies ) const
```

Compute objective function given the list of adjacencies.

39.348.3.11 OptimizeDictionary()

```
void OptimizeDictionary (
    const arma::mat & data,
    const arma::mat & codes,
    const arma::uvec & adjacencies )
```

Learn dictionary by solving linear system.

Parameters

<i>adjacencies</i>	Indices of entries (unrolled column by column) of the coding matrix Z that are non-zero (the adjacency matrix for the bipartite graph of points and atoms)
--------------------	--

39.348.3.12 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the model.

Referenced by LocalCoordinateCoding::Tolerance().

39.348.3.13 Tolerance() [1/2]

```
double Tolerance ( ) const [inline]
```

Get the objective tolerance.

Definition at line 195 of file lcc.hpp.

39.348.3.14 Tolerance() [2/2]

```
double& Tolerance ( ) [inline]
```

Modify the objective tolerance.

Definition at line 197 of file lcc.hpp.

References LocalCoordinateCoding::serialize().

39.348.3.15 Train()

```
double Train (
    const arma::mat & data,
    const DictionaryInitializer & initializer = DictionaryInitializer() )
```

Run local coordinate coding.

Parameters

<i>nIterations</i>	Maximum number of iterations to run algorithm.
<i>objTolerance</i>	Tolerance of objective function. When the objective function changes by a value lower than this tolerance, the optimization terminates.

Returns

The final objective value.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/local_coordinate_coding/ **lcc.hpp**

39.349 Constraints< MetricType > Class Template Reference

Interface for generating distance based constraints on a given dataset, provided corresponding true labels and a quantity parameter (k) are specified.

Public Types

- typedef **neighbor::NeighborSearch**< **neighbor::NearestNeighborSort**, MetricType > **KNN**
Convenience typedef.

Public Member Functions

- **Constraints** (const arma::mat &dataset, const arma::Row< size_t > &labels, const size_t k)
*Constructor for creating a **Constraints** (p. 1520) instance.*
- void **Impostors** (arma::Mat< size_t > &outputMatrix, const arma::mat &dataset, const arma::Row< size_t > &labels, const arma::vec &norms)
Calculates k differently labeled nearest neighbors for each datapoint and writes them back to passed matrix.
- void **Impostors** (arma::Mat< size_t > &outputNeighbors, arma::mat &outputDistance, const arma::mat &dataset, const arma::Row< size_t > &labels, const arma::vec &norms)
Calculates k differently labeled nearest neighbors & distances to impostors for each datapoint and writes them back to passed matrices.
- void **Impostors** (arma::Mat< size_t > &outputMatrix, const arma::mat &dataset, const arma::Row< size_t > &labels, const arma::vec &norms, const size_t begin, const size_t batchSize)
Calculates k differently labeled nearest neighbors for a batch of dataset and writes them back to passed matrix.
- void **Impostors** (arma::Mat< size_t > &outputNeighbors, arma::mat &outputDistance, const arma::mat &dataset, const arma::Row< size_t > &labels, const arma::vec &norms, const size_t begin, const size_t batchSize)
Calculates k differently labeled nearest neighbors & distances to impostors for a batch of dataset and writes them back to passed matrices.
- void **Impostors** (arma::Mat< size_t > &outputNeighbors, arma::mat &outputDistance, const arma::mat &dataset, const arma::Row< size_t > &labels, const arma::vec &norms, const arma::uvec &points, const size_t numPoints)
Calculates k differently labeled nearest neighbors & distances to impostors for some points of dataset and writes them back to passed matrices.
- const size_t & **K** () const
Get the number of target neighbors (k).
- size_t & **K** ()
Modify the number of target neighbors (k).

- const bool & **PreCalulated** () const
Access the boolean value of precalculated.
- bool & **PreCalulated** ()
Modify the value of precalculated.
- void **TargetNeighbors** (arma::Mat< size_t > &outputMatrix, const arma::mat &dataset, const arma::Row< size_t > &labels, const arma::vec &norms)
Calculates k similar labeled nearest neighbors and stores them into the passed matrix.
- void **TargetNeighbors** (arma::Mat< size_t > &outputMatrix, const arma::mat &dataset, const arma::Row< size_t > &labels, const arma::vec &norms, const size_t begin, const size_t batchSize)
Calculates k similar labeled nearest neighbors for a batch of dataset and stores them into the passed matrix.
- void **Triplets** (arma::Mat< size_t > &outputMatrix, const arma::mat &dataset, const arma::Row< size_t > &labels, const arma::vec &norms)
Generate triplets {i, j, l} for each datapoint i and writes back generated triplets to matrix passed.

39.349.1 Detailed Description

```
template<typename MetricType = metric::SquaredEuclideanDistance>
```

```
class mlpack::lmnn::Constraints< MetricType >
```

Interface for generating distance based constraints on a given dataset, provided corresponding true labels and a quantity parameter (k) are specified.

Class provides **TargetNeighbors()** (p. 1525) (Used for calculating target neighbors of each data point), **Impostors()** (p. 1522) (used for calculating impostors of each data point) and **Triplets()** (p. 1526) (Generates sets of {dataset, target neighbors, impostors} triplets.)

Definition at line 31 of file constraints.hpp.

39.349.2 Member Typedef Documentation

39.349.2.1 KNN

```
typedef neighbor::NeighborSearch< neighbor::NearestNeighborSort, MetricType> KNN
```

Convenience typedef.

Definition at line 36 of file constraints.hpp.

39.349.3 Constructor & Destructor Documentation

39.349.3.1 Constraints()

```
Constraints (
    const arma::mat & dataset,
    const arma::Row< size_t > & labels,
    const size_t k )
```

Constructor for creating a **Constraints** (p. 1520) instance.

Parameters

<i>dataset</i>	Input dataset.
<i>labels</i>	Input dataset labels.
<i>k</i>	Number of target neighbors, impostors & triplets.

39.349.4 Member Function Documentation

39.349.4.1 `Impostors()` [1/5]

```
void Impostors (
    arma::Mat< size_t > & outputMatrix,
    const arma::mat & dataset,
    const arma::Row< size_t > & labels,
    const arma::vec & norms )
```

Calculates k differently labeled nearest neighbors for each datapoint and writes them back to passed matrix.

Parameters

<i>outputMatrix</i>	Coordinates matrix to store impostors.
<i>dataset</i>	Input dataset.
<i>labels</i>	Input dataset labels.

39.349.4.2 `Impostors()` [2/5]

```
void Impostors (
    arma::Mat< size_t > & outputNeighbors,
    arma::mat & outputDistance,
    const arma::mat & dataset,
    const arma::Row< size_t > & labels,
    const arma::vec & norms )
```

Calculates k differently labeled nearest neighbors & distances to impostors for each datapoint and writes them back to passed matrices.

Parameters

<i>outputNeighbors</i>	Coordinates matrix to store impostors.
<i>outputDistance</i>	matrix to store distance.
<i>dataset</i>	Input dataset.
<i>labels</i>	Input dataset labels.

39.349.4.3 Impostors() [3/5]

```
void Impostors (
    arma::Mat< size_t > & outputMatrix,
    const arma::mat & dataset,
    const arma::Row< size_t > & labels,
    const arma::vec & norms,
    const size_t begin,
    const size_t batchSize )
```

Calculates k differently labeled nearest neighbors for a batch of dataset and writes them back to passed matrix.

Parameters

<i>outputMatrix</i>	Coordinates matrix to store impostors.
<i>dataset</i>	Input dataset.
<i>labels</i>	Input dataset labels.
<i>begin</i>	Index of the initial point of dataset.
<i>batchSize</i>	Number of data points to use.

39.349.4.4 Impostors() [4/5]

```
void Impostors (
    arma::Mat< size_t > & outputNeighbors,
    arma::mat & outputDistance,
    const arma::mat & dataset,
    const arma::Row< size_t > & labels,
    const arma::vec & norms,
    const size_t begin,
    const size_t batchSize )
```

Calculates k differently labeled nearest neighbors & distances to impostors for a batch of dataset and writes them back to passed matrices.

Parameters

<i>outputNeighbors</i>	Coordinates matrix to store impostors.
<i>outputDistance</i>	matrix to store distance.
<i>dataset</i>	Input dataset.
<i>labels</i>	Input dataset labels.
<i>begin</i>	Index of the initial point of dataset.
<i>batchSize</i>	Number of data points to use.

39.349.4.5 Impostors() [5/5]

```

void Impostors (
    arma::Mat< size_t > & outputNeighbors,
    arma::mat & outputDistance,
    const arma::mat & dataset,
    const arma::Row< size_t > & labels,
    const arma::vec & norms,
    const arma::uvec & points,
    const size_t numPoints )

```

Calculates k differently labeled nearest neighbors & distances to impostors for some points of dataset and writes them back to passed matrices.

Parameters

<i>outputNeighbors</i>	Coordinates matrix to store impostors.
<i>outputDistance</i>	matrix to store distance.
<i>dataset</i>	Input dataset.
<i>labels</i>	Input dataset labels.
<i>points</i>	Indices of data points to calculate impostors on.
<i>numPoints</i>	Number of points to actually calculate impostors on.

39.349.4.6 K() [1/2]

```
const size_t& K ( ) const [inline]
```

Get the number of target neighbors (k).

Definition at line 177 of file constraints.hpp.

39.349.4.7 K() [2/2]

```
size_t& K ( ) [inline]
```

Modify the number of target neighbors (k).

Definition at line 179 of file constraints.hpp.

39.349.4.8 PreCalulated() [1/2]

```
const bool& PreCalulated ( ) const [inline]
```

Access the boolean value of precalculated.

Definition at line 182 of file constraints.hpp.

39.349.4.9 PreCalulated() [2/2]

```
bool& PreCalulated ( ) [inline]
```

Modify the value of precalculated.

Definition at line 184 of file constraints.hpp.

39.349.4.10 TargetNeighbors() [1/2]

```
void TargetNeighbors (
    arma::Mat< size_t > & outputMatrix,
    const arma::mat & dataset,
    const arma::Row< size_t > & labels,
    const arma::vec & norms )
```

Calculates k similar labeled nearest neighbors and stores them into the passed matrix.

Parameters

<i>outputMatrix</i>	Coordinates matrix to store target neighbors.
<i>dataset</i>	Input dataset.
<i>labels</i>	Input dataset labels.

39.349.4.11 TargetNeighbors() [2/2]

```
void TargetNeighbors (
    arma::Mat< size_t > & outputMatrix,
    const arma::mat & dataset,
    const arma::Row< size_t > & labels,
    const arma::vec & norms,
```

```
const size_t begin,
const size_t batchSize )
```

Calculates k similar labeled nearest neighbors for a batch of dataset and stores them into the passed matrix.

Parameters

<i>outputMatrix</i>	Coordinates matrix to store target neighbors.
<i>dataset</i>	Input dataset.
<i>labels</i>	Input dataset labels.
<i>begin</i>	Index of the initial point of dataset.
<i>batchSize</i>	Number of data points to use.

39.349.4.12 Triplets()

```
void Triplets (
    arma::Mat< size_t > & outputMatrix,
    const arma::mat & dataset,
    const arma::Row< size_t > & labels,
    const arma::vec & norms )
```

Generate triplets {i, j, l} for each datapoint i and writes back generated triplets to matrix passed.

Parameters

<i>outputMatrix</i>	Coordinates matrix to store triplets.
<i>dataset</i>	Input dataset.
<i>labels</i>	Input dataset labels.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lmnn/ **constraints.hpp**

39.350 LMNN< MetricType, OptimizerType > Class Template Reference

An implementation of Large Margin nearest neighbor metric learning technique.

Public Member Functions

- **LMNN** (const arma::mat &dataset, const arma::Row< size_t > &labels, const size_t k, const MetricType metric=MetricType())

Initialize the **LMNN** (p. 1526) object, passing a dataset (distance metric is learned using this dataset) and labels.

- `const arma::mat & Dataset () const`
Get the dataset reference.
- `const size_t & K () const`
Access the value of *k*.
- `size_t K ()`
Modify the value of *k*.
- `const arma::Row< size_t > & Labels () const`
Get the labels reference.
- `void LearnDistance (arma::mat &outputMatrix)`
Perform Large Margin Nearest Neighbors metric learning.
- `const OptimizerType & Optimizer () const`
Get the optimizer.
- `OptimizerType & Optimizer ()`
- `const size_t & Range () const`
Access the range value.
- `size_t & Range ()`
Modify the range value.
- `const double & Regularization () const`
Access the regularization value.
- `double & Regularization ()`
Modify the regularization value.

39.350.1 Detailed Description

```
template<typename MetricType = metric::SquaredEuclideanDistance, typename OptimizerType = ens::AMSGrad>
class mlpack::lmnn::LMNN< MetricType, OptimizerType >
```

An implementation of Large Margin nearest neighbor metric learning technique.

The method seeks to improve clustering & classification algorithms on a dataset by transforming the dataset representation in a more convenient form for them. It introduces the concept of target neighbors and impostors, focusing on the idea that the distance between impostors and the perimeters established by target neighbors should be large and that between target neighbors and data point should be small. It requires the knowledge of target neighbors beforehand. Moreover, target neighbors once initialized remain same.

For more details, see the following published paper:

```
@ARTICLE{weinberger09distance,
  author = {Weinberger, K.Q. and Saul, L.K.},
  title = {{Distance metric learning for large margin nearest neighbor
    classification}},
  journal = {The Journal of Machine Learning Research},
  year = {2009},
  volume = {10},
  pages = {207--244},
  publisher = {MIT Press}
}
```

Template Parameters

<i>MetricType</i>	The type of metric to use for computation.
<i>OptimizerType</i>	Optimizer to use for developing distance.

Definition at line 55 of file `lmnn.hpp`.

39.350.2 Constructor & Destructor Documentation

39.350.2.1 LMNN()

```
LMNN (
    const arma::mat & dataset,
    const arma::Row< size_t > & labels,
    const size_t k,
    const MetricType metric = MetricType() )
```

Initialize the **LMNN** (p. 1526) object, passing a dataset (distance metric is learned using this dataset) and labels.

Initialization will copy both dataset and labels matrices to internal copies.

Parameters

<i>dataset</i>	Input dataset.
<i>labels</i>	Input dataset labels.
<i>k</i>	Number of targets to consider.
<i>metric</i>	Type of metric used for computation.

39.350.3 Member Function Documentation

39.350.3.1 Dataset()

```
const arma::mat& Dataset ( ) const [inline]
```

Get the dataset reference.

Definition at line 86 of file `lmnn.hpp`.

39.350.3.2 K() [1/2]

```
const size_t& K ( ) const [inline]
```

Access the value of k.

Definition at line 102 of file `lmnn.hpp`.

39.350.3.3 K() [2/2]

```
size_t K ( ) [inline]
```

Modify the value of k.

Definition at line 104 of file `lmnn.hpp`.

39.350.3.4 Labels()

```
const arma::Row<size_t>& Labels ( ) const [inline]
```

Get the labels reference.

Definition at line 89 of file `lmnn.hpp`.

39.350.3.5 LearnDistance()

```
void LearnDistance (
    arma::mat & outputMatrix )
```

Perform Large Margin Nearest Neighbors metric learning.

The output distance matrix is written into the passed reference. If the **LearnDistance()** (p. 1529) is called with an `outputMatrix` with correct dimensions, then that matrix will be used as the starting point for optimization.

Parameters

<i>outputMatrix</i>	Covariance matrix of Mahalanobis distance.
---------------------	--

39.350.3.6 Optimizer() [1/2]

```
const OptimizerType& Optimizer ( ) const [inline]
```

Get the optimizer.

Definition at line 107 of file Imnn.hpp.

39.350.3.7 Optimizer() [2/2]

```
OptimizerType& Optimizer ( ) [inline]
```

Definition at line 108 of file Imnn.hpp.

39.350.3.8 Range() [1/2]

```
const size_t& Range ( ) const [inline]
```

Access the range value.

Definition at line 97 of file Imnn.hpp.

39.350.3.9 Range() [2/2]

```
size_t& Range ( ) [inline]
```

Modify the range value.

Definition at line 99 of file Imnn.hpp.

39.350.3.10 Regularization() [1/2]

```
const double& Regularization ( ) const [inline]
```

Access the regularization value.

Definition at line 92 of file Imnn.hpp.

39.350.3.11 Regularization() [2/2]

```
double& Regularization ( ) [inline]
```

Modify the regularization value.

Definition at line 94 of file lmnn.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lmnn/ **lmnn.hpp**

39.351 LMNNFunction< MetricType > Class Template Reference

The Large Margin Nearest Neighbors function.

Public Member Functions

- **LMNNFunction** (const arma::mat &dataset, const arma::Row< size_t > &labels, size_t k, double regularization, size_t range, MetricType metric=MetricType())
*Constructor for **LMNNFunction** (p. 1531) class.*
- const arma::mat & **Dataset** () const
Return the dataset passed into the constructor.
- double **Evaluate** (const arma::mat &transformation)
*Evaluate the **LMNN** (p. 1526) function for the given transformation matrix.*
- double **Evaluate** (const arma::mat &transformation, const size_t begin, const size_t batchSize=1)
*Evaluate the **LMNN** (p. 1526) objective function for the given transformation matrix on the given batch size from a given initial point of the dataset.*
- template<typename GradType >
double **EvaluateWithGradient** (const arma::mat &transformation, GradType &gradient)
*Evaluate the **LMNN** (p. 1526) objective function together with gradient for the given transformation matrix.*
- template<typename GradType >
double **EvaluateWithGradient** (const arma::mat &transformation, const size_t begin, GradType &gradient, const size_t batchSize=1)
*Evaluate the **LMNN** (p. 1526) objective function together with gradient for the given transformation matrix on the given batch size, from a given initial point of the dataset.*
- const arma::mat & **GetInitialPoint** () const
Return the initial point for the optimization.
- template<typename GradType >
void **Gradient** (const arma::mat &transformation, GradType &gradient)
*Evaluate the gradient of the **LMNN** (p. 1526) function for the given transformation matrix.*
- template<typename GradType >
void **Gradient** (const arma::mat &transformation, const size_t begin, GradType &gradient, const size_t batchSize=1)
*Evaluate the gradient of the **LMNN** (p. 1526) function for the given transformation matrix on the given batch size, from a given initial point of the dataset.*
- const size_t & **K** () const

- *Access the value of k.*
- `size_t & K ()`
- *Modify the value of k.*
- `size_t NumFunctions () const`
- *Get the number of functions the objective function can be decomposed into.*
- `const size_t & Range () const`
- *Access the value of range.*
- `size_t & Range ()`
- *Modify the value of k.*
- `const double & Regularization () const`
- *Access the regularization value.*
- `double & Regularization ()`
- *Modify the regularization value.*
- `void Shuffle ()`
- *Shuffle the points in the dataset.*

39.351.1 Detailed Description

```
template<typename MetricType = metric::SquaredEuclideanDistance>
class mlpack::lmnn::LMNNFunction< MetricType >
```

The Large Margin Nearest Neighbors function.

The actual function is

$$(M) = \{ij\} \{ij\} \|L x_i - L x_j\|^2 + c_{ij} \{ij\} (1 - y_{ij}) [1 + \|L x_i - L x_j\|^2 - \|L x_i - L x_l\|^2]_{+}$$

where x_n represents a point and A is the current scaling matrix.

This class is more flexible than the original paper, allowing an arbitrary metric function to be used in place of $\|A x_i - A x_j\|^2$, meaning that the squared Euclidean distance is not the only allowed metric for **LMNN** (p. 1526). However, that is probably the best way to use this class.

In addition to the standard **Evaluate()** (p. 1533) and **Gradient()** (p. 1535) functions which mlpack optimizers use, overloads of **Evaluate()** (p. 1533) and **Gradient()** (p. 1535) are given which only operate on one point in the dataset. This is useful for optimizers like stochastic gradient descent (see `ens::SGD`).

Definition at line 46 of file `lmnn_function.hpp`.

39.351.2 Constructor & Destructor Documentation

39.351.2.1 LMNNFunction()

```
LMNNFunction (
    const arma::mat & dataset,
    const arma::Row< size_t > & labels,
    size_t k,
    double regularization,
    size_t range,
    MetricType metric = MetricType() )
```

Constructor for **LMNNFunction** (p. 1531) class.

Parameters

<i>dataset</i>	Input dataset.
<i>labels</i>	Input dataset labels.
<i>k</i>	Number of target neighbors to be used.
<i>regularization</i>	Regularization value.
<i>range</i>	Range after which impostors need to be recalculated.
<i>metric</i>	Type of metric used for computation.

39.351.3 Member Function Documentation

39.351.3.1 Dataset()

```
const arma::mat& Dataset ( ) const [inline]
```

Return the dataset passed into the constructor.

Definition at line 175 of file `lmnn_function.hpp`.

39.351.3.2 Evaluate() [1/2]

```
double Evaluate (
    const arma::mat & transformation )
```

Evaluate the **LMNN** (p. 1526) function for the given transformation matrix.

This is the non-separable implementation, where the objective function is not decomposed into the sum of several objective functions.

Parameters

<i>transformation</i>	Transformation matrix of Mahalanobis distance.
-----------------------	--

39.351.3.3 Evaluate() [2/2]

```
double Evaluate (
    const arma::mat & transformation,
```

```
const size_t begin,
const size_t batchSize = 1 )
```

Evaluate the **LMNN** (p. 1526) objective function for the given transformation matrix on the given batch size from a given initial point of the dataset.

This is the separable implementation, where the objective function is decomposed into the sum of many objective functions, and here, only one of those constituent objective functions is returned.

Parameters

<i>transformation</i>	Transformation matrix of Mahalanobis distance.
<i>begin</i>	Index of the initial point to use for objective function.
<i>batchSize</i>	Number of points to use for objective function.

39.351.3.4 EvaluateWithGradient() [1/2]

```
double EvaluateWithGradient (
    const arma::mat & transformation,
    GradType & gradient )
```

Evaluate the **LMNN** (p. 1526) objective function together with gradient for the given transformation matrix.

This is the non-separable implementation, where the objective function is not decomposed into the sum of several objective functions.

Template Parameters

<i>GradType</i>	The type of the gradient out-param.
-----------------	-------------------------------------

Parameters

<i>transformation</i>	Transformation matrix of Mahalanobis distance.
<i>gradient</i>	Matrix to store the calculated gradient in.

39.351.3.5 EvaluateWithGradient() [2/2]

```
double EvaluateWithGradient (
    const arma::mat & transformation,
    const size_t begin,
    GradType & gradient,
    const size_t batchSize = 1 )
```

Evaluate the **LMNN** (p. 1526) objective function together with gradient for the given transformation matrix on the given batch size, from a given initial point of the dataset.

This is the separable implementation, where the objective function is decomposed into the sum of many objective functions, and here, only one of those constituent objective functions is returned. The type of the gradient parameter is a template argument to allow the computation of a sparse gradient.

Template Parameters

<i>GradType</i>	The type of the gradient out-param.
-----------------	-------------------------------------

Parameters

<i>transformation</i>	Transformation matrix of Mahalanobis distance.
<i>begin</i>	Index of the initial point to use for objective function.
<i>gradient</i>	Matrix to store the calculated gradient in.
<i>batchSize</i>	Number of points to use for objective function.

39.351.3.6 GetInitialPoint()

```
const arma::mat& GetInitialPoint ( ) const [inline]
```

Return the initial point for the optimization.

Definition at line 166 of file `lmnn_function.hpp`.

39.351.3.7 Gradient() [1/2]

```
void Gradient (
    const arma::mat & transformation,
    GradType & gradient )
```

Evaluate the gradient of the **LMNN** (p. 1526) function for the given transformation matrix.

This is the non-separable implementation, where the objective function is not decomposed into the sum of several objective functions.

Template Parameters

<i>GradType</i>	The type of the gradient out-param.
-----------------	-------------------------------------

Parameters

<i>transformation</i>	Transformation matrix of Mahalanobis distance.
<i>gradient</i>	Matrix to store the calculated gradient in.

39.351.3.8 Gradient() [2/2]

```
void Gradient (
    const arma::mat & transformation,
    const size_t begin,
    GradType & gradient,
    const size_t batchSize = 1 )
```

Evaluate the gradient of the **LMNN** (p. 1526) function for the given transformation matrix on the given batch size, from a given initial point of the dataset.

This is the separable implementation, where the objective function is decomposed into the sum of many objective functions, and here, only one of those constituent objective functions is returned. The type of the gradient parameter is a template argument to allow the computation of a sparse gradient.

Template Parameters

<i>GradType</i>	The type of the gradient out-param.
-----------------	-------------------------------------

Parameters

<i>transformation</i>	Transformation matrix of Mahalanobis distance.
<i>begin</i>	Index of the initial point to use for objective function.
<i>gradient</i>	Matrix to store the calculated gradient in.
<i>batchSize</i>	Number of points to use for objective function.

39.351.3.9 K() [1/2]

```
const size_t& K ( ) const [inline]
```

Access the value of k.

Definition at line 183 of file `lmnn_function.hpp`.

39.351.3.10 K() [2/2]

```
size_t& K ( ) [inline]
```

Modify the value of k.

Definition at line 185 of file lmnn_function.hpp.

39.351.3.11 NumFunctions()

```
size_t NumFunctions ( ) const [inline]
```

Get the number of functions the objective function can be decomposed into.

This is just the number of points in the dataset.

Definition at line 172 of file lmnn_function.hpp.

39.351.3.12 Range() [1/2]

```
const size_t& Range ( ) const [inline]
```

Access the value of range.

Definition at line 188 of file lmnn_function.hpp.

39.351.3.13 Range() [2/2]

```
size_t& Range ( ) [inline]
```

Modify the value of k.

Definition at line 190 of file lmnn_function.hpp.

39.351.3.14 Regularization() [1/2]

```
const double& Regularization ( ) const [inline]
```

Access the regularization value.

Definition at line 178 of file lmnn_function.hpp.

39.351.3.15 Regularization() [2/2]

```
double& Regularization ( ) [inline]
```

Modify the regularization value.

Definition at line 180 of file `lmnn_function.hpp`.

39.351.3.16 Shuffle()

```
void Shuffle ( )
```

Shuffle the points in the dataset.

This may be used by optimizers.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lmnn/lmnn_function.hpp`

39.352 Log Class Reference

Provides a convenient way to give formatted output.

Static Public Member Functions

- static void **Assert** (bool condition, const std::string &message="Assert Failed.")
Checks if the specified condition is true.

Static Public Attributes

- static std::ostream & **cout**
Reference to cout, if necessary.
- static MLPACK_EXPORT **util::NullOutputStream Debug**
MLPACK_EXPORT is required for global variables, so that they are properly exported by the Windows compiler.
- static MLPACK_EXPORT **util::PrefixedOutputStream Fatal**
Prints fatal messages prefixed with [FATAL], then terminates the program.
- static MLPACK_EXPORT **util::PrefixedOutputStream Info**
Prints informational messages if -verbose is specified, prefixed with [INFO].
- static MLPACK_EXPORT **util::PrefixedOutputStream Warn**
Prints warning messages prefixed with [WARN].

39.352.1 Detailed Description

Provides a convenient way to give formatted output.

The **Log** (p. 1538) class has four members which can be used in the same way ostream can be used:

- **Log::Debug** (p. 1540)
- **Log::Info** (p. 1540)
- **Log::Warn** (p. 1541)
- **Log::Fatal** (p. 1540)

Each of these will prefix a tag to the output (for easy filtering), and the fatal output will terminate the program when a newline is encountered. An example is given below.

```
Log::Info << "Checking a condition." << std::endl;
if (!someCondition())
    Log::Warn << "someCondition() is not satisfied!" << std::endl;
Log::Info << "Checking an important condition." << std::endl;
if (!someImportantCondition())
{
    Log::Fatal << "someImportantCondition() is not satisfied! Terminating.";
    Log::Fatal << std::endl;
}
```

Any messages sent to **Log::Debug** (p. 1540) will not be shown when compiling in non-debug mode. Messages to **Log::Info** (p. 1540) will only be shown when the `-verbose` flag is given to the program (or rather, the **CLI** (p. 1117) class).

See also

PrefixOutputStream, NullOutputStream, **CLI** (p. 1117)

Definition at line 56 of file log.hpp.

39.352.2 Member Function Documentation

39.352.2.1 Assert()

```
static void Assert (
    bool condition,
    const std::string & message = "Assert Failed." ) [static]
```

Checks if the specified condition is true.

If not, halts program execution and prints a custom error message. Does nothing in non-debug mode.

Referenced by `EdgePair::EdgePair()`, and `mlpack::tree::split::PerformSplit()`.

39.352.3 Member Data Documentation

39.352.3.1 cout

```
std::ostream& cout [static]
```

Reference to cout, if necessary.

Definition at line 93 of file log.hpp.

39.352.3.2 Debug

```
MLPACK_EXPORT util::NullOutputStream Debug [static]
```

MLPACK_EXPORT is required for global variables, so that they are properly exported by the Windows compiler.

Dumps debug output into the bit nether regions.

Definition at line 79 of file log.hpp.

Referenced by DiagonalGMM::DiagonalGMM(), GMM::GMM(), and mlpack::bindings::cli::ParseCommandLine().

39.352.3.3 Fatal

```
MLPACK_EXPORT util::PrefixedOutputStream Fatal [static]
```

Prints fatal messages prefixed with [FATAL], then terminates the program.

Definition at line 90 of file log.hpp.

Referenced by SphericalKernel::ConvolutionIntegral(), mlpack::util::DisableBacktrace(), EigenvalueRatioConstraint::EigenvalueRatioConstraint(), AverageInterpolation::GetWeights(), SimilarityInterpolation::GetWeights(), RegressionInterpolation::GetWeights(), MeanImputation< T >::Impute(), GivenInitialization::Initialize(), ZScoreNormalization::Normalize(), mlpack::bindings::cli::ParseCommandLine(), and DiscreteDistribution::Probability().

39.352.3.4 Info

```
MLPACK_EXPORT util::PrefixedOutputStream Info [static]
```

Prints informational messages if `--verbose` is specified, prefixed with `[INFO]`.

Definition at line 84 of file `log.hpp`.

Referenced by `mlpack::util::DisableVerbose()`, `mlpack::util::EnableVerbose()`, `mlpack::bindings::cli::EndProgram()`, `SimpleResidueTermination::IsConverged()`, `SimpleToleranceTermination< MatType >::IsConverged()`, and `mlpack::bindings::cli::ParseCommandLine()`.

39.352.3.5 Warn

```
MLPACK_EXPORT util::PrefixedOutputStream Warn [static]
```

Prints warning messages prefixed with `[WARN]`.

Definition at line 87 of file `log.hpp`.

Referenced by `EigenvalueRatioConstraint::EigenvalueRatioConstraint()`, `RandomAcolInitialization< columnsToAverage >::Initialize()`, `MaxIterationTermination::MaxIterationTermination()`, and `CFTYPE< DecompositionPolicy, NormalizationType >::NumUsersForSimilarity()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ log.hpp`

39.353 ColumnsToBlocks Class Reference

Transform the columns of the given matrix into a block format.

Public Member Functions

- **ColumnsToBlocks** (size_t rows, size_t cols, size_t blockHeight=0, size_t blockWidth=0)
*Constructor a **ColumnsToBlocks** (p. 1541) object with the given parameters.*
- void **BlockHeight** (const size_t value)
Set the height of each block; see the constructor for more details.
- size_t **BlockHeight** () const
Get the block height.
- void **BlockWidth** (size_t value)
Set the width of each block; see the constructor for more details.
- size_t **BlockWidth** () const
Get the block width.
- void **BufSize** (const size_t value)
Modify the buffer size (the size of the margin around each column of the input).
- size_t **BufSize** () const
Get the buffer size.
- void **BufValue** (const double value)
Modify the value used for buffer cells; the default is -1.
- double **BufValue** () const
Get the value used for buffer cells.
- void **Cols** (const size_t value)
Set the number of blocks per column.
- size_t **Cols** () const
Return the number of blocks per column.
- void **MaxRange** (const double value)
*Set the maximum of the range the input will be scaled to, if scaling is enabled (see **Scale()** (p. 1548)).*
- double **MaxRange** () const
*Get the maximum of the range the input will be scaled to, if scaling is enabled (see **Scale()** (p. 1548)).*
- void **MinRange** (const double value)
*Set the minimum of the range the input will be scaled to, if scaling is enabled (see **Scale()** (p. 1548)).*
- double **MinRange** () const
*Get the minimum of the range the input will be scaled to, if scaling is enabled (see **Scale()** (p. 1548)).*
- void **Rows** (const size_t value)
Set the number of blocks per row.
- size_t **Rows** () const
Modify the number of blocks per row.
- void **Scale** (const bool value)
*Set whether or not scaling is enabled (see also **MaxRange()** (p. 1546) and **MinRange()** (p. 1547)).*
- bool **Scale** () const
*Get whether or not scaling is enabled (see also **MaxRange()** (p. 1546) and **MinRange()** (p. 1547)).*
- void **Transform** (const arma::mat &maximallInputs, arma::mat &output)
Transform the columns of the input matrix into blocks.

39.353.1 Detailed Description

Transform the columns of the given matrix into a block format.

This could be useful with the `mlpack::nn::MaximalInputs()` (p. 436) function, if your training samples are images. Roughly speaking, given a matrix

`[[A] [B] [C] [D]]`

then the **ColumnsToBlocks** (p. 1541) class can transform this to something like

`[[m m m m m] [m A m B m] [m m m m m] [m C m D m] [m m m m m]]`

where A through D are vectors and may themselves be reshaped by **ColumnsToBlocks** (p. 1541).

An example usage of the **ColumnsToBlocks** (p. 1541) class with the output of **MaximalInputs()** (p. 436) is given below; this assumes that the images are square, and will return a matrix with a one-element margin, with each maximal input (that is, each column of the maximalInput matrix) as a square block in the output matrix. 5 rows and columns of blocks will be in the output matrix.

```
// We assume we have a sparse autoencoder 'encoder'.
arma::mat maximalInput; // Store the features learned by sparse autoencoder
mlpack::nn::MaximalInputs(encoder.Parameters(), maximalInput);

arma::mat outputs;
const bool scale = true;

ColumnsToBlocks ctb(5, 5);
arma::mat output;
ctb.Transform(maximalInput, output);
// You can save the output as a pgm, this may help you visualize the training
// results.
output.save(fileName, arma::pgm_binary);
```

Another example of usage is given below, on a sample matrix.

```
// This matrix has two columns.
arma::mat input;
input << -1.0000 << 0.1429 << arma::endr
      << -0.7143 << 0.4286 << arma::endr
      << -0.4286 << 0.7143 << arma::endr
      << -0.1429 << 1.0000 << arma::endr;

arma::mat output;
ColumnsToBlocks ctb(1, 2);
ctb.Transform(input, output);

// The columns of the input will be reshaped as a square which is
// surrounded by padding value -1 (this value could be changed with the
// BufValue() method):
// -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000
// -1.0000 -1.0000 -1.0000 -0.4286 -1.0000 0.1429 0.7143 -1.0000
// -1.0000 -0.7143 -0.1429 -1.0000 0.4286 1.0000 -1.0000
// -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000

// Now, let's change some parameters; let's have each input column output not
// as a square, but as a 4x1 vector.
ctb.BlockWidth(1);
ctb.BlockHeight(4);
ctb.Transform(input, output);

// The output here will be similar, but each maximal input is 4x1:
// -1.0000 -1.0000 -1.0000 -1.0000 -1.0000
// -1.0000 -1.0000 -1.0000 0.1429 -1.0000
// -1.0000 -0.7143 -1.0000 0.4286 -1.0000
// -1.0000 -0.4286 -1.0000 0.7143 -1.0000
// -1.0000 -0.1429 -1.0000 1.0000 -1.0000
// -1.0000 -1.0000 -1.0000 -1.0000 -1.0000
```

The **ColumnsToBlocks** (p. 1541) class can also, depending on the parameters, scale the input to a given range (useful for exporting to PGM, for instance), and also set the buffer size and value. See the **Scale()** (p. 1548), **MinRange()** (p. 1547), **MaxRange()** (p. 1546), **BufSize()** (p. 1545), and **BufValue()** (p. 1546) methods for more details.

Definition at line 106 of file `columns_to_blocks.hpp`.

39.353.2 Constructor & Destructor Documentation

39.353.2.1 ColumnsToBlocks()

```
ColumnsToBlocks (
    size_t rows,
    size_t cols,
    size_t blockHeight = 0,
    size_t blockWidth = 0 )
```

Constructor a **ColumnsToBlocks** (p. 1541) object with the given parameters.

The rows and cols parameters control the number of blocks per row and column of the output matrix, respectively, and the blockHeight and blockWidth parameters control the size of the individual blocks. If blockHeight and blockWidth are specified, then (blockHeight * blockWidth) must be equal to the number of rows in the input matrix when **Transform()** (p. 1548) is called. If blockHeight and blockWidth are not specified, then the square root of the number of rows of the input matrix will be taken when **Transform()** (p. 1548) is called and that will be used as the block width and height.

Note that the **ColumnsToBlocks** (p. 1541) object can also scale the inputs to a given range; see **Scale()** (p. 1548), **MinRange()** (p. 1547), and **MaxRange()** (p. 1546), and the buffer (margin) size can also be set with **BufSize()** (p. 1545), and the value used for the buffer can be set with **BufValue()** (p. 1546).

Parameters

<i>rows</i>	Number of blocks in each column of the output matrix.
<i>cols</i>	Number of blocks in each row of the output matrix.
<i>blockHeight</i>	Height of each block.
<i>blockWidth</i>	Width of each block.

Warning

blockHeight * blockWidth must be equal to maximalInputs.n_rows.

39.353.3 Member Function Documentation

39.353.3.1 BlockHeight() [1/2]

```
void BlockHeight (
    const size_t value ) [inline]
```

Set the height of each block; see the constructor for more details.

Definition at line 149 of file columns_to_blocks.hpp.

39.353.3.2 BlockHeight() [2/2]

```
size_t BlockHeight ( ) const [inline]
```

Get the block height.

Definition at line 151 of file columns_to_blocks.hpp.

39.353.3.3 BlockWidth() [1/2]

```
void BlockWidth (
    size_t value ) [inline]
```

Set the width of each block; see the constructor for more details.

Definition at line 154 of file columns_to_blocks.hpp.

39.353.3.4 BlockWidth() [2/2]

```
size_t BlockWidth ( ) const [inline]
```

Get the block width.

Definition at line 156 of file columns_to_blocks.hpp.

39.353.3.5 BufSize() [1/2]

```
void BufSize (
    const size_t value ) [inline]
```

Modify the buffer size (the size of the margin around each column of the input).

The default value is 1.

Definition at line 160 of file columns_to_blocks.hpp.

39.353.3.6 BufSize() [2/2]

```
size_t BufSize ( ) const [inline]
```

Get the buffer size.

Definition at line 162 of file columns_to_blocks.hpp.

39.353.3.7 BufValue() [1/2]

```
void BufValue (
    const double value ) [inline]
```

Modify the value used for buffer cells; the default is -1.

Definition at line 165 of file columns_to_blocks.hpp.

39.353.3.8 BufValue() [2/2]

```
double BufValue ( ) const [inline]
```

Get the value used for buffer cells.

Definition at line 167 of file columns_to_blocks.hpp.

39.353.3.9 Cols() [1/2]

```
void Cols (
    const size_t value ) [inline]
```

Set the number of blocks per column.

Definition at line 196 of file columns_to_blocks.hpp.

39.353.3.10 Cols() [2/2]

```
size_t Cols ( ) const [inline]
```

Return the number of blocks per column.

Definition at line 198 of file columns_to_blocks.hpp.

39.353.3.11 MaxRange() [1/2]

```
void MaxRange (
    const double value ) [inline]
```

Set the maximum of the range the input will be scaled to, if scaling is enabled (see **Scale()** (p. 1548)).

Definition at line 171 of file columns_to_blocks.hpp.

39.353.3.12 MaxRange() [2/2]

```
double MaxRange ( ) const [inline]
```

Get the maximum of the range the input will be scaled to, if scaling is enabled (see **Scale()** (p. 1548)).

Definition at line 174 of file columns_to_blocks.hpp.

39.353.3.13 MinRange() [1/2]

```
void MinRange (
    const double value ) [inline]
```

Set the minimum of the range the input will be scaled to, if scaling is enabled (see **Scale()** (p. 1548)).

Definition at line 178 of file columns_to_blocks.hpp.

39.353.3.14 MinRange() [2/2]

```
double MinRange ( ) const [inline]
```

Get the minimum of the range the input will be scaled to, if scaling is enabled (see **Scale()** (p. 1548)).

Definition at line 181 of file columns_to_blocks.hpp.

39.353.3.15 Rows() [1/2]

```
void Rows (
    const size_t value ) [inline]
```

Set the number of blocks per row.

Definition at line 191 of file columns_to_blocks.hpp.

39.353.3.16 Rows() [2/2]

```
size_t Rows ( ) const [inline]
```

Modify the number of blocks per row.

Definition at line 193 of file columns_to_blocks.hpp.

39.353.3.17 Scale() [1/2]

```
void Scale (
    const bool value ) [inline]
```

Set whether or not scaling is enabled (see also **MaxRange()** (p. 1546) and **MinRange()** (p. 1547)).

Definition at line 185 of file columns_to_blocks.hpp.

39.353.3.18 Scale() [2/2]

```
bool Scale ( ) const [inline]
```

Get whether or not scaling is enabled (see also **MaxRange()** (p. 1546) and **MinRange()** (p. 1547)).

Definition at line 188 of file columns_to_blocks.hpp.

39.353.3.19 Transform()

```
void Transform (
    const arma::mat & maximalInputs,
    arma::mat & output )
```

Transform the columns of the input matrix into blocks.

If blockHeight and blockWidth were not specified in the constructor (and **BlockHeight()** (p. 1544) and **BlockWidth()** (p. 1545) were not called), then the number of rows in the input matrix must be a perfect square.

Parameters

<i>input</i>	Input matrix to transform.
<i>output</i>	Matrix to store transformed output in.

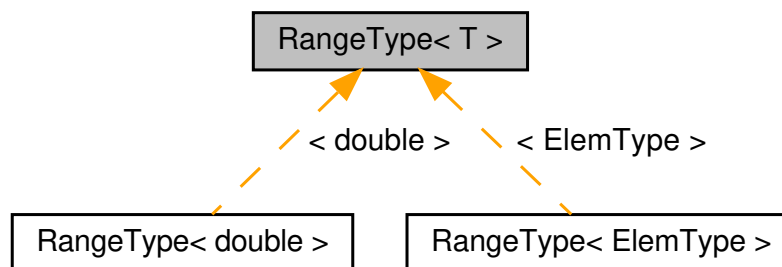
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/ **columns_to_blocks.hpp**

39.354 RangeType< T > Class Template Reference

Simple real-valued range.

Inheritance diagram for RangeType< T >:



Public Member Functions

- **RangeType** ()
The upper bound.
- **RangeType** (const T point)
- **RangeType** (const T lo, const T hi)
Initializes to specified range.
- bool **Contains** (const T d) const
Determines if a point is contained within the range.
- bool **Contains** (const **RangeType** &r) const
Determines if another range overlaps with this one.
- T **Hi** () const
Get the upper bound.
- T & **Hi** ()
Modify the upper bound.
- T **Lo** () const
Get the lower bound.
- T & **Lo** ()
Modify the lower bound.
- T **Mid** () const

Gets the midpoint of this range.

- **RangeType operator &** (const **RangeType** &rhs) const

Shrinks this range to be the overlap with another range; this makes an empty set if there is no overlap.

- **RangeType & operator &=** (const **RangeType** &rhs)

Shrinks this range to be the overlap with another range; this makes an empty set if there is no overlap.

- bool **operator!=** (const **RangeType** &rhs) const

Compare with another range for strict equality.

- **RangeType operator*** (const T d) const

Scale the bounds by the given double.

- **RangeType & operator*=** (const T d)

Scale the bounds by the given double.

- bool **operator<** (const **RangeType** &rhs) const

Compare with another range.

- bool **operator==** (const **RangeType** &rhs) const

Compare with another range for strict equality.

- bool **operator>** (const **RangeType** &rhs) const

Compare with another range.

- **RangeType operator|** (const **RangeType** &rhs) const

Expands this range to include another range.

- **RangeType & operator|=** (const **RangeType** &rhs)

Expands this range to include another range.

- template<typename Archive >

void **serialize** (Archive &ar, const unsigned int version)

Serialize the range object.

- T **Width** () const

Gets the span of the range (hi - lo).

Friends

- template<typename TT >

RangeType< TT > operator* (const TT d, const **RangeType< TT >** &r)

Scale the bounds by the given double.

39.354.1 Detailed Description

```
template<typename T = double>
class mpack::math::RangeType< T >
```

Simple real-valued range.

It contains an upper and lower bound.

Note that until mpack 3.0.0, this class is named `RangeType<>` and for the specification where T is double, you can use `math::Range` (p. 409). As of mpack 3.0.0, this class will be renamed `math::Range<>` (p. 409).

Template Parameters

<i>T</i>	type of element held by this range.
----------	-------------------------------------

Definition at line 19 of file range.hpp.

39.354.2 Constructor & Destructor Documentation

39.354.2.1 RangeType() [1/3]

```
RangeType ( ) [inline]
```

The upper bound.

Initialize to an empty set (where lo > hi).

39.354.2.2 RangeType() [2/3]

```
RangeType (
    const T point ) [inline]
```

39.354.2.3 RangeType() [3/3]

```
RangeType (
    const T lo,
    const T hi ) [inline]
```

Initializes to specified range.

Parameters

<i>lo</i>	Lower bound of the range.
<i>hi</i>	Upper bound of the range.

39.354.3 Member Function Documentation

39.354.3.1 Contains() [1/2]

```
bool Contains (
    const T d ) const [inline]
```

Determines if a point is contained within the range.

Parameters

<i>d</i>	Point to check.
----------	-----------------

Referenced by `RangeType< ElemType >::Hi()`.

39.354.3.2 Contains() [2/2]

```
bool Contains (
    const RangeType< T > & r ) const [inline]
```

Determines if another range overlaps with this one.

Parameters

<i>r</i>	Other range.
----------	--------------

Returns

true if ranges overlap at all.

39.354.3.3 Hi() [1/2]

```
T Hi ( ) const [inline]
```

Get the upper bound.

Definition at line 66 of file range.hpp.

Referenced by `HollowBallBound< TMetricType, ElemType >::Diameter()`, `HollowBallBound< TMetricType, ElemType >::MinWidth()`, and `HollowBallBound< TMetricType, ElemType >::OuterRadius()`.

39.354.3.4 Hi() [2/2]

```
T& Hi ( ) [inline]
```

Modify the upper bound.

Definition at line 68 of file range.hpp.

39.354.3.5 Lo() [1/2]

```
T Lo ( ) const [inline]
```

Get the lower bound.

Definition at line 61 of file range.hpp.

Referenced by HollowBallBound< TMetricType, ElemType >::InnerRadius().

39.354.3.6 Lo() [2/2]

```
T& Lo ( ) [inline]
```

Modify the lower bound.

Definition at line 63 of file range.hpp.

39.354.3.7 Mid()

```
T Mid ( ) const [inline]
```

Gets the midpoint of this range.

Referenced by RangeType< ElemType >::Hi().

39.354.3.8 operator &()

```
RangeType operator& (  
    const RangeType< T > & rhs ) const [inline]
```

Shrinks this range to be the overlap with another range; this makes an empty set if there is no overlap.

Parameters

<i>rhs</i>	Other range.
------------	--------------

Referenced by `RangeType< ElemType >::Hi()`.

39.354.3.9 `operator &=()`

```
RangeType& operator&= (
    const RangeType< T > & rhs ) [inline]
```

Shrinks this range to be the overlap with another range; this makes an empty set if there is no overlap.

Parameters

<i>rhs</i>	Other range.
------------	--------------

Referenced by `RangeType< ElemType >::Hi()`.

39.354.3.10 `operator !=()`

```
bool operator!= (
    const RangeType< T > & rhs ) const [inline]
```

Compare with another range for strict equality.

Parameters

<i>rhs</i>	Other range.
------------	--------------

Referenced by `RangeType< ElemType >::Hi()`.

39.354.3.11 `operator*()`

```
RangeType operator* (
    const T d ) const [inline]
```

Scale the bounds by the given double.

Parameters

<i>d</i>	Scaling factor.
----------	-----------------

Referenced by RangeType< ElemType >::Hi().

39.354.3.12 operator*=()

```
RangeType& operator*= (
    const T d ) [inline]
```

Scale the bounds by the given double.

Parameters

<i>d</i>	Scaling factor.
----------	-----------------

Referenced by RangeType< ElemType >::Hi().

39.354.3.13 operator<()

```
bool operator< (
    const RangeType< T > & rhs ) const [inline]
```

Compare with another range.

For Range objects x and y, $x < y$ means that x is strictly less than y and does not overlap at all.

Parameters

<i>rhs</i>	Other range.
------------	--------------

Referenced by RangeType< ElemType >::Hi().

39.354.3.14 operator==()

```
bool operator== (
    const RangeType< T > & rhs ) const [inline]
```

Compare with another range for strict equality.

Parameters

<i>rhs</i>	Other range.
------------	--------------

Referenced by `RangeType< ElemType >::Hi()`.

39.354.3.15 `operator>()`

```
bool operator> (
    const RangeType< T > & rhs ) const [inline]
```

Compare with another range.

For Range objects x and y, $x < y$ means that x is strictly less than y and does not overlap at all.

Parameters

<i>rhs</i>	Other range.
------------	--------------

Referenced by `RangeType< ElemType >::Hi()`.

39.354.3.16 `operator" |()`

```
RangeType operator| (
    const RangeType< T > & rhs ) const [inline]
```

Expands this range to include another range.

Parameters

<i>rhs</i>	Range to include.
------------	-------------------

Referenced by `RangeType< ElemType >::Hi()`.

39.354.3.17 `operator" |=()`

```
RangeType& operator|= (
    const RangeType< T > & rhs ) [inline]
```

Expands this range to include another range.

Parameters

<i>rhs</i>	Range to include.
------------	-------------------

Referenced by RangeType< ElemType >::Hi().

39.354.3.18 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int version )
```

Serialize the range object.

Referenced by RangeType< ElemType >::Hi().

39.354.3.19 Width()

```
T Width ( ) const [inline]
```

Gets the span of the range (hi - lo).

Referenced by RangeType< ElemType >::Hi().

39.354.4 Friends And Related Function Documentation

39.354.4.1 operator*

```
RangeType<TT> operator* (
    const TT d,
    const RangeType< TT > & r ) [friend]
```

Scale the bounds by the given double.

Parameters

<i>d</i>	Scaling factor.
----------	-----------------

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/ **range.hpp**

39.355 MatrixCompletion Class Reference

This class implements the popular nuclear norm minimization heuristic for matrix completion problems.

Public Member Functions

- **MatrixCompletion** (const size_t m, const size_t n, const arma::umat &indices, const arma::vec &values, const size_t r)
Construct a matrix completion problem, specifying the maximum rank of the solution.
- **MatrixCompletion** (const size_t m, const size_t n, const arma::umat &indices, const arma::vec &values, const arma::mat &initialPoint)
Construct a matrix completion problem, specifying the initial point of the optimization.
- **MatrixCompletion** (const size_t m, const size_t n, const arma::umat &indices, const arma::vec &values)
Construct a matrix completion problem.
- void **Recover** (arma::mat &recovered)
Solve the underlying SDP to fill in the remaining values.
- const ens::LRSDP< ens::SDP< arma::sp_mat > > & **Sdp** () const
Return the underlying SDP.
- ens::LRSDP< ens::SDP< arma::sp_mat > > & **Sdp** ()
Modify the underlying SDP.

39.355.1 Detailed Description

This class implements the popular nuclear norm minimization heuristic for matrix completion problems.

That is, given known values M_{ij} 's, the following optimization problem (semi-definite program) is solved to fill in the remaining unknown values of X

$$\min ||X||_* \text{ subj to } X_{ij} = M_{ij}$$

where $||X||_*$ denotes the nuclear norm (sum of singular values of X).

For a theoretical treatment of the conditions necessary for exact recovery, see the following paper:

A Simpler Approach to Matrix Completion. Benjamin Recht. JMLR 11. <http://arxiv.org/pdf/0910.0651v2.pdf>

An example of how to use this class is shown below:

```
size_t m, n;           // size of unknown matrix
arma::umat indices;    // contains the known indices [2 x n_entries]
arma::vec values;      // contains the known values [n_entries]
arma::mat recovered;   // will contain the completed matrix

MatrixCompletion mc(m, n, indices, values);
mc.Recover(recovered);
```

See also

LRSDP

Definition at line 52 of file matrix_completion.hpp.

39.355.2 Constructor & Destructor Documentation

39.355.2.1 MatrixCompletion() [1/3]

```
MatrixCompletion (
    const size_t m,
    const size_t n,
    const arma::umat & indices,
    const arma::vec & values,
    const size_t r )
```

Construct a matrix completion problem, specifying the maximum rank of the solution.

Parameters

<i>m</i>	Number of rows of original matrix.
<i>n</i>	Number of columns of original matrix.
<i>indices</i>	Matrix containing the indices of the known entries (must be [2 x p]).
<i>values</i>	Vector containing the values of the known entries (must be length p).
<i>r</i>	Maximum rank of solution.

39.355.2.2 MatrixCompletion() [2/3]

```
MatrixCompletion (
    const size_t m,
    const size_t n,
    const arma::umat & indices,
    const arma::vec & values,
    const arma::mat & initialPoint )
```

Construct a matrix completion problem, specifying the initial point of the optimization.

Parameters

<i>m</i>	Number of rows of original matrix.
<i>n</i>	Number of columns of original matrix.
<i>indices</i>	Matrix containing the indices of the known entries (must be [2 x p]).
<i>values</i>	Vector containing the values of the known entries (must be length p).
<i>initialPoint</i>	Starting point for the SDP optimization.

39.355.2.3 MatrixCompletion() [3/3]

```

MatrixCompletion (
    const size_t m,
    const size_t n,
    const arma::umat & indices,
    const arma::vec & values )

```

Construct a matrix completion problem.

Parameters

<i>m</i>	Number of rows of original matrix.
<i>n</i>	Number of columns of original matrix.
<i>indices</i>	Matrix containing the indices of the known entries (must be [2 x p]).
<i>values</i>	Vector containing the values of the known entries (must be length p).

39.355.3 Member Function Documentation**39.355.3.1 Recover()**

```

void Recover (
    arma::mat & recovered )

```

Solve the underlying SDP to fill in the remaining values.

Parameters

<i>recovered</i>	Will contain the completed matrix.
------------------	------------------------------------

39.355.3.2 Sdp() [1/2]

```

const ens::LRSDP<ens::SDP<arma::sp_mat> >& Sdp ( ) const [inline]

```

Return the underlying SDP.

Definition at line 114 of file `matrix_completion.hpp`.

39.355.3.3 Sdp() [2/2]

```
ens::LRSDP<ens::SDP<arma::sp_mat> >& Sdp ( ) [inline]
```

Modify the underlying SDP.

Definition at line 119 of file matrix_completion.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/matrix_completion/ **matrix_completion.hpp**

39.356 MeanShift< UseKernel, KernelType, MatType > Class Template Reference

This class implements mean shift clustering.

Public Member Functions

- **MeanShift** (const double radius=0, const size_t maxIterations=1000, const KernelType kernel=KernelType())
Create a mean shift object and set the parameters which mean shift will be run with.
- void **Cluster** (const MatType &data, arma::Row< size_t > &assignments, arma::mat ¢roids, bool force←
Convergence=true, bool useSeeds=true)
Perform mean shift clustering on the data, returning a list of cluster assignments and centroids.
- double **EstimateRadius** (const MatType &data, const double ratio=0.2)
Give an estimation of radius based on given dataset.
- const KernelType & **Kernel** () const
Get the kernel.
- KernelType & **Kernel** ()
Modify the kernel.
- size_t **MaxIterations** () const
Get the maximum number of iterations.
- size_t & **MaxIterations** ()
Set the maximum number of iterations.
- double **Radius** () const
Get the radius.
- void **Radius** (double radius)
Set the radius.

39.356.1 Detailed Description

```
template<bool UseKernel = false, typename KernelType = kernel::GaussianKernel, typename MatType = arma::mat>
class mlpack::meanshift::MeanShift< UseKernel, KernelType, MatType >
```

This class implements mean shift clustering.

For each point in dataset, apply mean shift algorithm until maximum iterations or convergence. Then remove duplicate centroids.

A simple example of how to run mean shift clustering is shown below.

```
extern arma::mat data; // Dataset we want to run mean shift on.
arma::Row<size_t> assignments; // Cluster assignments.
arma::mat centroids; // Cluster centroids.
bool forceConvergence = true; // Flag whether to force each centroid seed
to converge regardless of maxIterations.
```

```
MeanShift<> meanShift();
meanShift.Cluster(dataset, assignments, centroids, forceConvergence);
```

Template Parameters

<i>UseKernel</i>	Use kernel or mean to calculate new centroid. If false, KernelType will be ignored.
<i>KernelType</i>	The kernel to use.
<i>MatType</i>	The type of matrix the data is stored in.

Definition at line 51 of file mean_shift.hpp.

39.356.2 Constructor & Destructor Documentation

39.356.2.1 MeanShift()

```
MeanShift (
    const double radius = 0,
    const size_t maxIterations = 1000,
    const KernelType kernel = KernelType() )
```

Create a mean shift object and set the parameters which mean shift will be run with.

Parameters

<i>radius</i>	If distance of two centroids is less than it, one will be removed. If this value isn't positive, an estimation will be given when clustering.
<i>maxIterations</i>	Maximum number of iterations allowed before giving up iterations will terminate.
<i>kernel</i>	Optional KernelType object.

39.356.3 Member Function Documentation

39.356.3.1 Cluster()

```
void Cluster (
    const MatType & data,
    arma::Row< size_t > & assignments,
    arma::mat & centroids,
    bool forceConvergence = true,
    bool useSeeds = true )
```

Perform mean shift clustering on the data, returning a list of cluster assignments and centroids.

Template Parameters

<i>MatType</i>	Type of matrix.
----------------	-----------------

Parameters

<i>data</i>	Dataset to cluster.
<i>assignments</i>	Vector to store cluster assignments in.
<i>centroids</i>	Matrix in which centroids are stored.
<i>forceConvergence</i>	Flag whether to force each centroid seed to converge regardless of maxIterations.

39.356.3.2 EstimateRadius()

```
double EstimateRadius (
    const MatType & data,
    const double ratio = 0.2 )
```

Give an estimation of radius based on given dataset.

Parameters

<i>data</i>	Dataset for estimation.
<i>ratio</i>	Percentage of dataset to use for nearest neighbor search.

39.356.3.3 Kernel() [1/2]

```
const KernelType& Kernel ( ) const [inline]
```

Get the kernel.

Definition at line 105 of file mean_shift.hpp.

39.356.3.4 Kernel() [2/2]

```
KernelType& Kernel ( ) [inline]
```

Modify the kernel.

Definition at line 107 of file mean_shift.hpp.

39.356.3.5 MaxIterations() [1/2]

```
size_t MaxIterations ( ) const [inline]
```

Get the maximum number of iterations.

Definition at line 95 of file mean_shift.hpp.

39.356.3.6 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Set the maximum number of iterations.

Definition at line 97 of file mean_shift.hpp.

39.356.3.7 Radius() [1/2]

```
double Radius ( ) const [inline]
```

Get the radius.

Definition at line 100 of file mean_shift.hpp.

39.356.3.8 Radius() [2/2]

```
void Radius (
    double radius )
```

Set the radius.

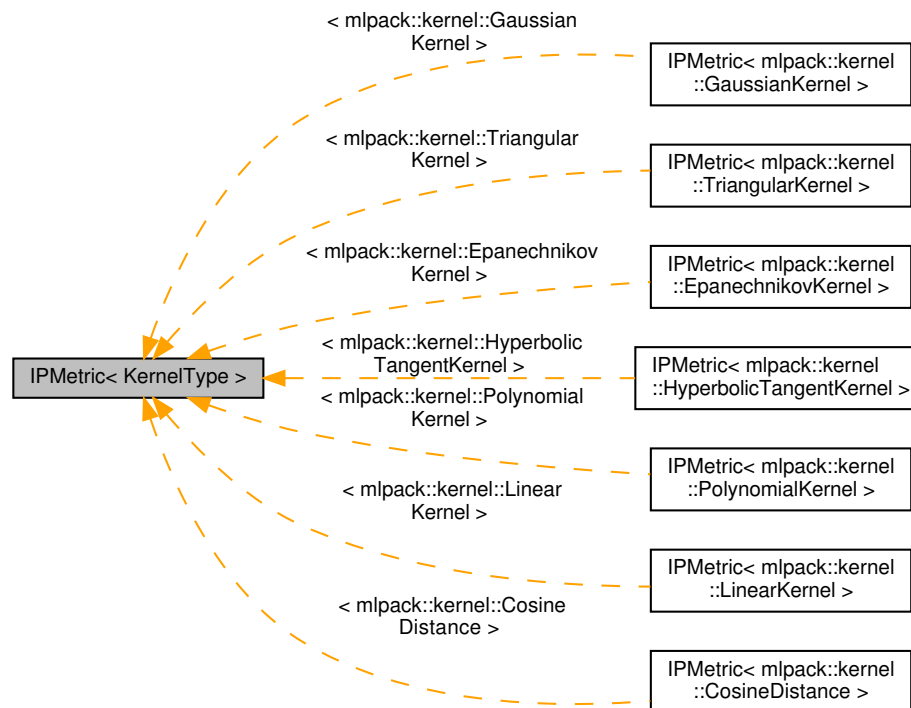
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/mean_shift/ **mean_shift.hpp**

39.357 IPMetric< KernelType > Class Template Reference

The inner product metric, **IPMetric** (p. 1565), takes a given Mercer kernel (KernelType), and when **Evaluate()** (p. 1567) is called, returns the distance between the two points in kernel space:

Inheritance diagram for IPMetric< KernelType >:



Public Member Functions

- **IPMetric** ()
Create the **IPMetric** (p. 1565) without an instantiated kernel.
- **IPMetric** (KernelType &kernel)
Create the **IPMetric** (p. 1565) with an instantiated kernel.

- **IPMetric** (const **IPMetric** &other)
Copy the parameters of the given metric.
- **~IPMetric** ()
*Destroy the **IPMetric** (p. 1565) object.*
- template<typename VecTypeA , typename VecTypeB >
VecTypeA::elem_type **Evaluate** (const VecTypeA &a, const VecTypeB &b)
Evaluate the metric.
- const KernelType & **Kernel** () const
Get the kernel.
- KernelType & **Kernel** ()
Modify the kernel.
- **IPMetric** & **operator=** (const **IPMetric** &other)
Assign this metric to be a copy of the given metric.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int version)
Serialize the metric.

39.357.1 Detailed Description

```
template<typename KernelType>
class mlpack::metric::IPMetric< KernelType >
```

The inner product metric, **IPMetric** (p. 1565), takes a given Mercer kernel (KernelType), and when **Evaluate()** (p. 1567) is called, returns the distance between the two points in kernel space:

$$d(x, y) = \sqrt{K(x, x) + K(y, y) - 2K(x, y)}.$$

Template Parameters

<i>KernelType</i>	Type of Kernel to use. This must be a Mercer kernel (positive definite), otherwise the metric may not be valid.
-------------------	---

Definition at line 32 of file ip_metric.hpp.

39.357.2 Constructor & Destructor Documentation

39.357.2.1 IPMetric() [1/3]

```
IPMetric ( )
```

Create the **IPMetric** (p. 1565) without an instantiated kernel.

39.357.2.2 IPMetric() [2/3]

```
IPMetric (
    KernelType & kernel )
```

Create the **IPMetric** (p. 1565) with an instantiated kernel.

39.357.2.3 ~IPMetric()

```
~ IPMetric ( )
```

Destroy the **IPMetric** (p. 1565) object.

39.357.2.4 IPMetric() [3/3]

```
IPMetric (
    const IPMetric< KernelType > & other )
```

Copy the parameters of the given metric.

39.357.3 Member Function Documentation

39.357.3.1 Evaluate()

```
VecTypeA::elem_type Evaluate (
    const VecTypeA & a,
    const VecTypeB & b )
```

Evaluate the metric.

Template Parameters

<i>VecTypeA</i>	Type of first vector.
<i>VecTypeB</i>	Type of second vector.

Parameters

<i>a</i>	First vector.
----------	---------------

Parameters

<i>b</i>	Second vector.
----------	----------------

Returns

Distance between the two points in kernel space.

39.357.3.2 Kernel() [1/2]

```
const KernelType& Kernel ( ) const [inline]
```

Get the kernel.

Definition at line 63 of file ip_metric.hpp.

39.357.3.3 Kernel() [2/2]

```
KernelType& Kernel ( ) [inline]
```

Modify the kernel.

Definition at line 65 of file ip_metric.hpp.

39.357.3.4 operator=()

```
IPMetric& operator= (
    const IPMetric< KernelType > & other )
```

Assign this metric to be a copy of the given metric.

39.357.3.5 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int version )
```

Serialize the metric.

Referenced by IPMetric< mlpack::kernel::CosineDistance >::Kernel().

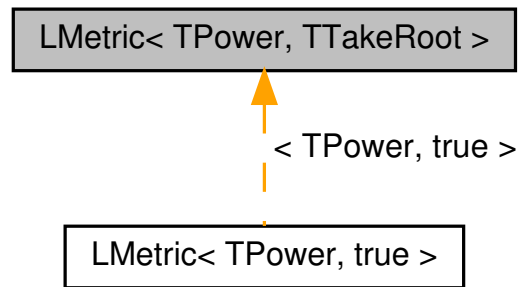
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/metrics/ ip_metric.hpp

39.358 LMetric< TPower, TTakeRoot > Class Template Reference

The L_p metric for arbitrary integer p, with an option to take the root.

Inheritance diagram for LMetric< TPower, TTakeRoot >:



Public Member Functions

- **LMetric** ()
- `template<typename Archive >`
`void serialize (Archive &, const unsigned int)`
Serialize the metric (nothing to do).

Static Public Member Functions

- `template<typename VecTypeA , typename VecTypeB >`
`static VecTypeA::elem_type Evaluate (const VecTypeA &a, const VecTypeB &b)`
Computes the distance between two points.

Static Public Attributes

- static const int **Power** = TPower
The power of the metric.
- static const bool **TakeRoot** = TTakeRoot
Whether or not the root is taken.

39.358.1 Detailed Description

```
template<int TPower, bool TTakeRoot = true>
class mlpack::metric::LMetric< TPower, TTakeRoot >
```

The L_p metric for arbitrary integer p , with an option to take the root.

This class implements the standard L_p metric for two arbitrary vectors x and y of dimensionality n :

$$d(x, y) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}}.$$

The value of p is given as a template parameter.

In addition, the function $d(x, y)$ can be simplified, neglecting the p -root calculation. This is done by specifying the `TakeRoot` template parameter to be false. Then,

$$d(x, y) = \sum_{i=1}^n |x_i - y_i|^p$$

It is faster to compute that distance, so `TakeRoot` is by default off. However, when `TakeRoot` is false, the distance given is not actually a true metric – it does not satisfy the triangle inequality. Some `mlpack` methods do not require the triangle inequality to operate correctly (such as the `BinarySpaceTree`), but setting `TakeRoot = false` in some cases will cause incorrect results.

A few convenience typedefs are given:

- `ManhattanDistance`
- `EuclideanDistance`
- `SquaredEuclideanDistance`

Template Parameters

<i>Power</i>	Power of metric; i.e. <code>Power = 1</code> gives the L1-norm (Manhattan distance).
<i>TakeRoot</i>	If true, the <code>Power</code> 'th root of the result is taken before it is returned. Setting this to false causes the metric to not satisfy the Triangle Inequality (be careful!).

Definition at line 63 of file `lmetric.hpp`.

39.358.2 Constructor & Destructor Documentation

39.358.2.1 LMetric()

```
LMetric ( ) [inline]
```

Definition at line 70 of file lmetric.hpp.

39.358.3 Member Function Documentation

39.358.3.1 Evaluate()

```
static VecTypeA::elem_type Evaluate (
    const VecTypeA & a,
    const VecTypeB & b ) [static]
```

Computes the distance between two points.

Template Parameters

<i>VecTypeA</i>	Type of first vector (generally arma::vec or arma::sp_vec).
<i>VecTypeB</i>	Type of second vector.

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Returns

Distance between vectors a and b.

Referenced by SphericalKernel::ConvolutionIntegral(), GaussianKernel::ConvolutionIntegral(), SphericalKernel::Evaluate(), TriangularKernel::Evaluate(), LaplacianKernel::Evaluate(), GaussianKernel::Evaluate(), CauchyKernel::Evaluate(), and LMetric< TPower, true >::LMetric().

39.358.3.2 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serialize the metric (nothing to do).

Definition at line 88 of file lmetric.hpp.

39.358.4 Member Data Documentation

39.358.4.1 Power

```
const int Power = TPower [static]
```

The power of the metric.

Definition at line 91 of file lmetric.hpp.

39.358.4.2 TakeRoot

```
const bool TakeRoot = TTakeRoot [static]
```

Whether or not the root is taken.

Definition at line 93 of file lmetric.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/metrics/ **lmetric.hpp**

39.359 MahalanobisDistance< TakeRoot > Class Template Reference

The Mahalanobis distance, which is essentially a stretched Euclidean distance.

Public Member Functions

- **MahalanobisDistance** ()
Initialize the Mahalanobis distance with the empty matrix as covariance.
- **MahalanobisDistance** (const size_t dimensionality)
Initialize the Mahalanobis distance with the identity matrix of the given dimensionality.
- **MahalanobisDistance** (arma::mat covariance)
Initialize the Mahalanobis distance with the given covariance matrix.
- const arma::mat & **Covariance** () const
Access the covariance matrix.
- arma::mat & **Covariance** ()
Modify the covariance matrix.
- template<typename VecTypeA , typename VecTypeB >
double **Evaluate** (const VecTypeA &a, const VecTypeB &b)
Evaluate the distance between the two given points using this Mahalanobis distance.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int version)
Serialize the Mahalanobis distance.

39.359.1 Detailed Description

```
template<bool TakeRoot = true>
class mlpack::metric::MahalanobisDistance< TakeRoot >
```

The Mahalanobis distance, which is essentially a stretched Euclidean distance.

Given a square covariance matrix Q of size $d \times d$, where d is the dimensionality of the points it will be evaluating, and given two vectors x and y also of dimensionality d ,

$$d(x, y) = \sqrt{(x - y)^T Q (x - y)}$$

where Q is the covariance matrix.

Because each evaluation multiplies $(x_1 - x_2)$ by the covariance matrix, it is typically much quicker to use an **LMetric** (p. 1569) and simply stretch the actual dataset itself before performing any evaluations. However, this class is provided for convenience.

If you wish to use the KNN class or other tree-based algorithms with this distance, it is recommended to instead stretch the dataset first, by decomposing $Q = L^T L$ (perhaps via a Cholesky decomposition), and then multiply the data by L . If you still wish to use the KNN class with a custom distance anyway, you will need to use a different tree type than the default KDTree, which only works with the **LMetric** (p. 1569) class.

Similar to the **LMetric** (p. 1569) class, this offers a template parameter `TakeRoot` which, when set to false, will instead evaluate the distance

$$d(x, y) = (x - y)^T Q (x - y)$$

which is faster to evaluate.

Template Parameters

<i>TakeRoot</i>	If true, takes the root of the output. It is slightly faster to leave this at the default of false, but this means the metric may not satisfy the triangle inequality and may not be usable for methods that expect a true metric.
-----------------	--

Definition at line 60 of file mahalanobis_distance.hpp.

39.359.2 Constructor & Destructor Documentation

39.359.2.1 MahalanobisDistance() [1/3]

```
MahalanobisDistance ( ) [inline]
```

Initialize the Mahalanobis distance with the empty matrix as covariance.

Don't call **Evaluate()** (p. 1575) until you set the covariance with **Covariance()** (p. 1575)!

Definition at line 67 of file mahalanobis_distance.hpp.

39.359.2.2 MahalanobisDistance() [2/3]

```
MahalanobisDistance (
    const size_t dimensionality ) [inline]
```

Initialize the Mahalanobis distance with the identity matrix of the given dimensionality.

Parameters

<i>dimensionality</i>	Dimesnsionality of the covariance matrix.
-----------------------	---

Definition at line 75 of file mahalanobis_distance.hpp.

39.359.2.3 MahalanobisDistance() [3/3]

```
MahalanobisDistance (
    arma::mat covariance ) [inline]
```

Initialize the Mahalanobis distance with the given covariance matrix.

The given covariance matrix will be copied (this is not optimal).

Parameters

<i>covariance</i>	The covariance matrix to use for this distance.
-------------------	---

Definition at line 84 of file mahalanobis_distance.hpp.

References MahalanobisDistance< TakeRoot >::Evaluate().

39.359.3 Member Function Documentation

39.359.3.1 Covariance() [1/2]

```
const arma::mat& Covariance ( ) const [inline]
```

Access the covariance matrix.

Returns

Constant reference to the covariance matrix.

Definition at line 104 of file mahalanobis_distance.hpp.

39.359.3.2 Covariance() [2/2]

```
arma::mat& Covariance ( ) [inline]
```

Modify the covariance matrix.

Returns

Reference to the covariance matrix.

Definition at line 111 of file mahalanobis_distance.hpp.

References MahalanobisDistance< TakeRoot >::serialize().

39.359.3.3 Evaluate()

```
double Evaluate (
    const VecTypeA & a,
    const VecTypeB & b )
```

Evaluate the distance between the two given points using this Mahalanobis distance.

If the covariance matrix has not been set (i.e. if you used the empty constructor and did not later modify the covariance matrix), calling this method will probably result in a crash.

Parameters

<i>a</i>	First vector.
<i>b</i>	Second vector.

Referenced by MahalanobisDistance< TakeRoot >::MahalanobisDistance().

39.359.3.4 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int version )
```

Serialize the Mahalanobis distance.

Referenced by MahalanobisDistance< TakeRoot >::Covariance().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/metrics/ **mahalanobis_distance.hpp**

39.360 NaiveBayesClassifier< ModelMatType > Class Template Reference

The simple Naive Bayes classifier.

Public Types

- typedef ModelMatType::elem_type **ElemType**

Public Member Functions

- template<typename MatType >
NaiveBayesClassifier (const MatType &data, const arma::Row< size_t > &labels, const size_t numClasses, const bool incrementalVariance=false)
Initializes the classifier as per the input and then trains it by calculating the sample mean and variances.
- **NaiveBayesClassifier** (const size_t dimensionality=0, const size_t numClasses=0)
Initialize the Naive Bayes classifier without performing training.
- template<typename VecType >
size_t **Classify** (const VecType &point) const
*Classify the given point, using the trained **NaiveBayesClassifier** (p. 1576) model.*
- template<typename VecType , typename ProbabilitiesVecType >
void **Classify** (const VecType &point, size_t &prediction, ProbabilitiesVecType &probabilities) const
*Classify the given point using the trained **NaiveBayesClassifier** (p. 1576) model and also return estimates of the probability for each class in the given vector.*
- template<typename MatType >
void **Classify** (const MatType &data, arma::Row< size_t > &predictions) const
*Classify the given points using the trained **NaiveBayesClassifier** (p. 1576) model.*

- `template<typename MatType , typename ProbabilitiesMatType >`
`void Classify (const MatType &data, arma::Row< size_t > &predictions, ProbabilitiesMatType &probabilities)`
`const`
*Classify the given points using the trained **NaiveBayesClassifier** (p. 1576) model and also return estimates of the probabilities for each class in the given matrix.*
- `const ModelMatType & Means () const`
Get the sample means for each class.
- `ModelMatType & Means ()`
Modify the sample means for each class.
- `const ModelMatType & Probabilities () const`
Get the prior probabilities for each class.
- `ModelMatType & Probabilities ()`
Modify the prior probabilities for each class.
- `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`
Serialize the classifier.
- `template<typename MatType >`
`void Train (const MatType &data, const arma::Row< size_t > &labels, const size_t numClasses, const bool incremental=true)`
Train the Naive Bayes classifier on the given dataset.
- `template<typename VecType >`
`void Train (const VecType &point, const size_t label)`
Train the Naive Bayes classifier on the given point.
- `const ModelMatType & Variances () const`
Get the sample variances for each class.
- `ModelMatType & Variances ()`
Modify the sample variances for each class.

39.360.1 Detailed Description

```
template<typename ModelMatType = arma::mat>
class mlpack::naive_bayes::NaiveBayesClassifier< ModelMatType >
```

The simple Naive Bayes classifier.

This class trains on the data by calculating the sample mean and variance of the features with respect to each of the labels, and also the class probabilities. The class labels are assumed to be positive integers (starting with 0), and are expected to be the last row of the data input to the constructor.

Mathematically, it computes $P(X_i = x_i \mid Y = y_j)$ for each feature X_i for each of the labels y_j . Along with this, it also computes the class probabilities $P(Y = y_j)$.

For classifying a data point (x_1, x_2, \dots, x_n) , it computes the following: $\arg \max_y (P(Y = y) * P(X_1 = x_1 \mid Y = y) * \dots * P(X_n = x_n \mid Y = y))$

Example use:

```
extern arma::mat training_data, testing_data;
NaiveBayesClassifier<> nbc(training_data, 5);
arma::vec results;

nbc.Classify(testing_data, results);
```

The `ModelMatType` template parameter specifies the internal matrix type that **NaiveBayesClassifier** (p. 1576) will use to hold the means, variances, and weights that make up the Naive Bayes model. This can be `arma::mat`, `arma::fmat`, or any other Armadillo (or Armadillo-compatible) object. Because `ModelMatType` may be different than the type of the data the model is trained on, now training is possible with subviews, sparse matrices, or anything else, while still storing the model as a `ModelMatType` internally.

Template Parameters

<i>ModelMatType</i>	Internal matrix type to use to store the model.
---------------------	---

Definition at line 58 of file `naive_bayes_classifier.hpp`.

39.360.2 Member Typedef Documentation

39.360.2.1 ElemType

```
typedef ModelMatType::elem_type ElemType
```

Definition at line 62 of file `naive_bayes_classifier.hpp`.

39.360.3 Constructor & Destructor Documentation

39.360.3.1 NaiveBayesClassifier() [1/2]

```
NaiveBayesClassifier (
    const MatType & data,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const bool incrementalVariance = false )
```

Initializes the classifier as per the input and then trains it by calculating the sample mean and variances.

Example use:

```
extern arma::mat training_data, testing_data;
extern arma::Row<size_t> labels;
NaiveBayesClassifier nbc(training_data, labels, 5);
```

Parameters

<i>data</i>	Training data points.
<i>labels</i>	Labels corresponding to training data points.
<i>numClasses</i>	Number of classes in this classifier.
<i>incrementalVariance</i>	If true, an incremental algorithm is used to calculate the variance; this can prevent loss of precision in some cases, but will be somewhat slower to calculate.

39.360.3.2 NaiveBayesClassifier() [2/2]

```
NaiveBayesClassifier (
    const size_t dimensionality = 0,
    const size_t numClasses = 0 )
```

Initialize the Naive Bayes classifier without performing training.

All of the parameters of the model will be initialized to zero. Be sure to use **Train()** (p. 1582) before calling **Classify()** (p. 1579), otherwise the results may be meaningless.

39.360.4 Member Function Documentation

39.360.4.1 Classify() [1/4]

```
size_t Classify (
    const VecType & point ) const
```

Classify the given point, using the trained **NaiveBayesClassifier** (p. 1576) model.

The predicted label is returned.

Parameters

<i>point</i>	Point to classify.
--------------	--------------------

39.360.4.2 Classify() [2/4]

```
void Classify (
    const VecType & point,
```

```
size_t & prediction,
ProbabilitiesVecType & probabilities ) const
```

Classify the given point using the trained **NaiveBayesClassifier** (p. 1576) model and also return estimates of the probability for each class in the given vector.

Parameters

<i>point</i>	Point to classify.
<i>prediction</i>	This will be set to the predicted class of the point.
<i>probabilities</i>	This will be filled with class probabilities for the point.

39.360.4.3 Classify() [3/4]

```
void Classify (
    const MatType & data,
    arma::Row< size_t > & predictions ) const
```

Classify the given points using the trained **NaiveBayesClassifier** (p. 1576) model.

The predicted labels for each point are stored in the given vector.

```
arma::mat test_data; // each column is a test point
arma::Row<size_t> results;
...
nbc.Classify(test_data, results);
```

Parameters

<i>data</i>	List of data points.
<i>predictions</i>	Vector that class predictions will be placed into.

39.360.4.4 Classify() [4/4]

```
void Classify (
    const MatType & data,
    arma::Row< size_t > & predictions,
    ProbabilitiesMatType & probabilities ) const
```

Classify the given points using the trained **NaiveBayesClassifier** (p. 1576) model and also return estimates of the probabilities for each class in the given matrix.

The predicted labels for each point are stored in the given vector.

```
arma::mat test_data; // each column is a test point
arma::Row<size_t> results;
arma::mat resultsProbs;
...
nbc.Classify(test_data, results, resultsProbs);
```

Parameters

<i>data</i>	Set of points to classify.
<i>predictions</i>	This will be filled with predictions for each point.
<i>probabilities</i>	This will be filled with class probabilities for each point. Each row represents a point.

Template Parameters

<i>MatType</i>	Type of data to be classified.
<i>ProbabilitiesMatType</i>	Type to store output probabilities in.

39.360.4.5 Means() [1/2]

```
const ModelMatType& Means ( ) const [inline]
```

Get the sample means for each class.

Definition at line 200 of file naive_bayes_classifier.hpp.

39.360.4.6 Means() [2/2]

```
ModelMatType& Means ( ) [inline]
```

Modify the sample means for each class.

Definition at line 202 of file naive_bayes_classifier.hpp.

39.360.4.7 Probabilities() [1/2]

```
const ModelMatType& Probabilities ( ) const [inline]
```

Get the prior probabilities for each class.

Definition at line 210 of file naive_bayes_classifier.hpp.

39.360.4.8 Probabilities() [2/2]

```
ModelMatType& Probabilities ( ) [inline]
```

Modify the prior probabilities for each class.

Definition at line 212 of file `naive_bayes_classifier.hpp`.

References `NaiveBayesClassifier< ModelMatType >::serialize()`.

39.360.4.9 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the classifier.

Referenced by `NaiveBayesClassifier< ModelMatType >::Probabilities()`.

39.360.4.10 Train() [1/2]

```
void Train (
    const MatType & data,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const bool incremental = true )
```

Train the Naive Bayes classifier on the given dataset.

If the incremental algorithm is used, the current model is used as a starting point (this is the default). If the incremental algorithm is not used, then the current model is ignored and the new model will be trained only on the given data. Note that even if the incremental algorithm is not used, the data must have the same dimensionality and number of classes that the model was initialized with. If you want to change the dimensionality or number of classes, either re-initialize or call **Means()** (p. 1581), **Variances()** (p. 1583), and **Probabilities()** (p. 1581) individually to set them to the right size.

Parameters

<i>data</i>	The dataset to train on.
<i>labels</i>	The labels for the dataset.
<i>numClasses</i>	The numbe of classes in the dataset.
<i>incremental</i>	Whether or not to use the incremental algorithm for training.

39.360.4.11 Train() [2/2]

```
void Train (
    const VecType & point,
    const size_t label )
```

Train the Naive Bayes classifier on the given point.

This will use the incremental algorithm for updating the model parameters. The data must be the same dimensionality as the existing model parameters.

Parameters

<i>point</i>	Data point to train on.
<i>label</i>	Label of data point.

39.360.4.12 Variances() [1/2]

```
const ModelMatType& Variances ( ) const [inline]
```

Get the sample variances for each class.

Definition at line 205 of file naive_bayes_classifier.hpp.

39.360.4.13 Variances() [2/2]

```
ModelMatType& Variances ( ) [inline]
```

Modify the sample variances for each class.

Definition at line 207 of file naive_bayes_classifier.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/naive_bayes/ **naive_bayes_classifier.hpp**

39.361 NCA< MetricType, OptimizerType > Class Template Reference

An implementation of Neighborhood Components Analysis, both a linear dimensionality reduction technique and a distance learning technique.

Public Member Functions

- **NCA** (const arma::mat &dataset, const arma::Row< size_t > &labels, MetricType metric=MetricType())
Construct the Neighborhood Components Analysis object.
- const arma::mat & **Dataset** () const
Get the dataset reference.
- const arma::Row< size_t > & **Labels** () const
Get the labels reference.
- void **LearnDistance** (arma::mat &outputMatrix)
Perform Neighborhood Components Analysis.
- const OptimizerType & **Optimizer** () const
Get the optimizer.
- OptimizerType & **Optimizer** ()

39.361.1 Detailed Description

```
template<typename MetricType = metric::SquaredEuclideanDistance, typename OptimizerType = ens::StandardSGD>
class mlpack::nca::NCA< MetricType, OptimizerType >
```

An implementation of Neighborhood Components Analysis, both a linear dimensionality reduction technique and a distance learning technique.

The method seeks to improve k-nearest-neighbor classification on a dataset by scaling the dimensions. The method is nonparametric, and does not require a value of k. It works by using stochastic ("soft") neighbor assignments and using optimization techniques over the gradient of the accuracy of the neighbor assignments.

For more details, see the following published paper:

```
@inproceedings{Goldberger2004,
  author = {Goldberger, Jacob and Roweis, Sam and Hinton, Geoff and
    Salakhutdinov, Ruslan},
  booktitle = {Advances in Neural Information Processing Systems 17},
  pages = {513--520},
  publisher = {MIT Press},
  title = {{Neighbourhood Components Analysis}},
  year = {2004}
}
```

Definition at line 49 of file nca.hpp.

39.361.2 Constructor & Destructor Documentation

39.361.2.1 NCA()

```
NCA (
    const arma::mat & dataset,
    const arma::Row< size_t > & labels,
    MetricType metric = MetricType() )
```

Construct the Neighborhood Components Analysis object.

This simply stores the reference to the dataset and labels as well as the parameters for optimization before the actual optimization is performed.

Parameters

<i>dataset</i>	Input dataset.
<i>labels</i>	Input dataset labels.
<i>stepSize</i>	Step size for stochastic gradient descent.
<i>maxIterations</i>	Maximum iterations for stochastic gradient descent.
<i>tolerance</i>	Tolerance for termination of stochastic gradient descent.
<i>shuffle</i>	Whether or not to shuffle the dataset during SGD.
<i>metric</i>	Instantiated metric to use.

39.361.3 Member Function Documentation

39.361.3.1 Dataset()

```
const arma::mat& Dataset ( ) const [inline]
```

Get the dataset reference.

Definition at line 81 of file nca.hpp.

39.361.3.2 Labels()

```
const arma::Row<size_t>& Labels ( ) const [inline]
```

Get the labels reference.

Definition at line 83 of file nca.hpp.

39.361.3.3 LearnDistance()

```
void LearnDistance (
    arma::mat & outputMatrix )
```

Perform Neighborhood Components Analysis.

The output distance learning matrix is written into the passed reference. If **LearnDistance()** (p. 1585) is called with an outputMatrix which has the correct size (dataset.n_rows x dataset.n_rows), that matrix will be used as the starting point for optimization.

Parameters

<i>output_matrix</i>	Covariance matrix of Mahalanobis distance.
----------------------	--

39.361.3.4 **Optimizer()** [1/2]

```
const OptimizerType& Optimizer ( ) const [inline]
```

Get the optimizer.

Definition at line 86 of file nca.hpp.

39.361.3.5 **Optimizer()** [2/2]

```
OptimizerType& Optimizer ( ) [inline]
```

Definition at line 87 of file nca.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nca/ **nca.hpp**

39.362 **SoftmaxErrorFunction< MetricType > Class Template Reference**

The "softmax" stochastic neighbor assignment probability function.

Public Member Functions

- **SoftmaxErrorFunction** (const arma::mat &dataset, const arma::Row< size_t > &labels, MetricType metric=MetricType())
Initialize with the given kernel; useful when the kernel has some state to store, which is set elsewhere.
- double **Evaluate** (const arma::mat &covariance)
Evaluate the softmax function for the given covariance matrix.
- double **Evaluate** (const arma::mat &covariance, const size_t begin, const size_t batchSize=1)
Evaluate the softmax objective function for the given covariance matrix on the given batch size from a given initial point of the dataset.
- const arma::mat **GetInitialPoint** () const
Get the initial point.
- void **Gradient** (const arma::mat &covariance, arma::mat &gradient)
Evaluate the gradient of the softmax function for the given covariance matrix.
- template<typename GradType >
void **Gradient** (const arma::mat &covariance, const size_t begin, GradType &gradient, const size_t batchSize=1)
Evaluate the gradient of the softmax function for the given covariance matrix on the given batch size, from a given initial point of the dataset.
- size_t **NumFunctions** () const
Get the number of functions the objective function can be decomposed into.
- void **Shuffle** ()
Shuffle the dataset.

39.362.1 Detailed Description

```
template<typename MetricType = metric::SquaredEuclideanDistance>
class mlpack::nca::SoftmaxErrorFunction< MetricType >
```

The "softmax" stochastic neighbor assignment probability function.

The actual function is

$$p_{ij} = (\exp(-\|A x_i - A x_j\|^2)) / (\sum_{k \neq i} (\exp(-\|A x_i - A x_k\|^2)))$$

where x_n represents a point and A is the current scaling matrix.

This class is more flexible than the original paper, allowing an arbitrary metric function to be used in place of $\|A x_i - A x_j\|^2$, meaning that the squared Euclidean distance is not the only allowed metric for **NCA** (p. 1583). However, that is probably the best way to use this class.

In addition to the standard **Evaluate()** (p. 1588) and **Gradient()** (p. 1588) functions which mlpack optimizers use, overloads of **Evaluate()** (p. 1588) and **Gradient()** (p. 1588) are given which only operate on one point in the dataset. This is useful for optimizers like stochastic gradient descent (see `mlpack::optimization::SGD`).

Definition at line 43 of file `nca_softmax_error_function.hpp`.

39.362.2 Constructor & Destructor Documentation

39.362.2.1 SoftmaxErrorFunction()

```
SoftmaxErrorFunction (
    const arma::mat & dataset,
    const arma::Row< size_t > & labels,
    MetricType metric = MetricType() )
```

Initialize with the given kernel; useful when the kernel has some state to store, which is set elsewhere.

If no kernel is given, an empty kernel is used; this way, you can call the constructor with no arguments. A reference to the dataset we will be optimizing over is also required.

Parameters

<i>dataset</i>	Matrix containing the dataset.
<i>labels</i>	Vector of class labels for each point in the dataset.
<i>kernel</i>	Instantiated kernel (optional).

39.362.3 Member Function Documentation

39.362.3.1 Evaluate() [1/2]

```
double Evaluate (
    const arma::mat & covariance )
```

Evaluate the softmax function for the given covariance matrix.

This is the non-separable implementation, where the objective function is not decomposed into the sum of several objective functions.

Parameters

<i>covariance</i>	Covariance matrix of Mahalanobis distance.
-------------------	--

39.362.3.2 Evaluate() [2/2]

```
double Evaluate (
    const arma::mat & covariance,
    const size_t begin,
    const size_t batchSize = 1 )
```

Evaluate the softmax objective function for the given covariance matrix on the given batch size from a given initial point of the dataset.

This is the separable implementation, where the objective function is decomposed into the sum of many objective functions, and here, only one of those constituent objective functions is returned.

Parameters

<i>covariance</i>	Covariance matrix of Mahalanobis distance.
<i>begin</i>	Index of the initial point to use for objective function.
<i>batchSize</i>	Number of points to use for objective function.

39.362.3.3 GetInitialPoint()

```
const arma::mat GetInitialPoint ( ) const
```

Get the initial point.

39.362.3.4 Gradient() [1/2]

```
void Gradient (
    const arma::mat & covariance,
    arma::mat & gradient )
```

Evaluate the gradient of the softmax function for the given covariance matrix.

This is the non-separable implementation, where the objective function is not decomposed into the sum of several objective functions.

Parameters

<i>covariance</i>	Covariance matrix of Mahalanobis distance.
<i>gradient</i>	Matrix to store the calculated gradient in.

39.362.3.5 Gradient() [2/2]

```
void Gradient (
    const arma::mat & covariance,
    const size_t begin,
    GradType & gradient,
    const size_t batchSize = 1 )
```

Evaluate the gradient of the softmax function for the given covariance matrix on the given batch size, from a given initial point of the dataset.

This is the separable implementation, where the objective function is decomposed into the sum of many objective functions, and here, only one of those constituent objective functions is returned. The type of the gradient parameter is a template argument to allow the computation of a sparse gradient.

Template Parameters

<i>GradType</i>	The type of the gradient out-param.
-----------------	-------------------------------------

Parameters

<i>covariance</i>	Covariance matrix of Mahalanobis distance.
<i>begin</i>	Index of the initial point to use for objective function.
<i>batchSize</i>	Number of points to use for objective function.
<i>gradient</i>	Matrix to store the calculated gradient in.

39.362.3.6 NumFunctions()

```
size_t NumFunctions ( ) const [inline]
```

Get the number of functions the objective function can be decomposed into.

This is just the number of points in the dataset.

Definition at line 130 of file `nca_softmax_error_function.hpp`.

39.362.3.7 Shuffle()

```
void Shuffle ( )
```

Shuffle the dataset.

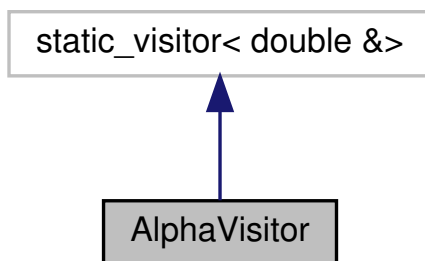
The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nca/ nca_softmax_error_function.hpp`

39.363 AlphaVisitor Class Reference

Exposes the `Alpha()` method of the given `RAType`.

Inheritance diagram for `AlphaVisitor`:

**Public Member Functions**

- `template<typename RAType >`
`double & operator() (RAType *ra) const`
Return Alpha parameter.

39.363.1 Detailed Description

Exposes the Alpha() method of the given RAType.

Definition at line 198 of file ra_model.hpp.

39.363.2 Member Function Documentation

39.363.2.1 operator()()

```
double& operator() (
    RAType * ra ) const
```

Return Alpha parameter.

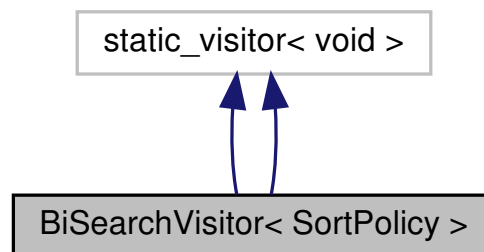
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ ra_model.hpp

39.364 BiSearchVisitor< SortPolicy > Class Template Reference

BiSearchVisitor (p. 1591) executes a bichromatic neighbor search on the given NSType.

Inheritance diagram for BiSearchVisitor< SortPolicy >:



Public Types

- `template<template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType> using NSTypeT = NSType< SortPolicy, TreeType >`
Alias template necessary for visual c++ compiler.
- `template<template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType> using RATypeT = RAType< SortPolicy, TreeType >`
Alias template necessary for visual c++ compiler.

Public Member Functions

- **BiSearchVisitor** (const arma::mat &querySet, const size_t k, arma::Mat< size_t > &neighbors, arma::mat &distances, const size_t leafSize)
*Construct the **BiSearchVisitor** (p. 1591).*
- **BiSearchVisitor** (const arma::mat &querySet, const size_t k, arma::Mat< size_t > &neighbors, arma::mat &distances, const size_t leafSize, const double tau, const double rho)
*Construct the **BiSearchVisitor** (p. 1591).*
- template<template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType>
void **operator()** (**RATypeT**< TreeType > *ra) const
Default Bichromatic neighbor search on the given RAType instance.
- void **operator()** (**RATypeT**< **tree::KDTree** > *ra) const
Bichromatic search on the given RAType specialized for KDTrees.
- void **operator()** (**RATypeT**< **tree::Octree** > *ra) const
Bichromatic search on the given RAType specialized for octrees.
- template<template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType>
void **operator()** (**NSTypeT**< TreeType > *ns) const
Default Bichromatic neighbor search on the given NSType instance.
- void **operator()** (**NSTypeT**< **tree::KDTree** > *ns) const
Bichromatic neighbor search on the given NSType specialized for KDTrees.
- void **operator()** (**NSTypeT**< **tree::BallTree** > *ns) const
Bichromatic neighbor search on the given NSType specialized for BallTrees.
- void **operator()** (**SpillKNN** *ns) const
Bichromatic neighbor search specialized for SPTrees.
- void **operator()** (**NSTypeT**< **tree::Octree** > *ns) const
Bichromatic neighbor search specialized for octrees.

39.364.1 Detailed Description

```
template<typename SortPolicy>
class mlpack::neighbor::BiSearchVisitor< SortPolicy >
```

BiSearchVisitor (p. 1591) executes a bichromatic neighbor search on the given NSType.

BiSearchVisitor (p. 1591) executes a bichromatic neighbor search on the given RAType.

We use template specialization to differentiate those tree types that accept leafSize as a parameter. In these cases, before doing neighbor search, a query tree with proper leafSize is built from the querySet.

We use template specialization to differentiate those tree types types that accept leafSize as a parameter. In these cases, before doing neighbor search a query tree with proper leafSize is built from the querySet.

Definition at line 80 of file ns_model.hpp.

39.364.2 Member Typedef Documentation

39.364.2.1 NTypeT

```
using NTypeT = NType<SortPolicy, TreeType>
```

Alias template necessary for visual c++ compiler.

Definition at line 107 of file ns_model.hpp.

39.364.2.2 RTypeT

```
using RTypeT = RType<SortPolicy, TreeType>
```

Alias template necessary for visual c++ compiler.

Definition at line 98 of file ra_model.hpp.

39.364.3 Constructor & Destructor Documentation

39.364.3.1 BiSearchVisitor() [1/2]

```
BiSearchVisitor (
    const arma::mat & querySet,
    const size_t k,
    arma::Mat< size_t > & neighbors,
    arma::mat & distances,
    const size_t leafSize,
    const double tau,
    const double rho )
```

Construct the **BiSearchVisitor** (p. 1591).

39.364.3.2 BiSearchVisitor() [2/2]

```
BiSearchVisitor (
    const arma::mat & querySet,
    const size_t k,
    arma::Mat< size_t > & neighbors,
    arma::mat & distances,
    const size_t leafSize )
```

Construct the **BiSearchVisitor** (p. 1591).

39.364.4 Member Function Documentation

39.364.4.1 operator>() [1/8]

```
void operator() (
    RATypeT< TreeType > * ra ) const
```

Default Bichromatic neighbor search on the given RAType instance.

39.364.4.2 operator>() [2/8]

```
void operator() (
    RATypeT< tree::KDTree > * ra ) const
```

Bichromatic search on the given RAType specialized for KDTrees.

39.364.4.3 operator>() [3/8]

```
void operator() (
    RATypeT< tree::Octree > * ra ) const
```

Bichromatic search on the given RAType specialized for octrees.

39.364.4.4 operator>() [4/8]

```
void operator() (
    NSTypeT< TreeType > * ns ) const
```

Default Bichromatic neighbor search on the given NSType instance.

39.364.4.5 operator>() [5/8]

```
void operator() (
    NSTypeT< tree::KDTree > * ns ) const
```

Bichromatic neighbor search on the given NSType specialized for KDTrees.

39.364.4.6 operator>() [6/8]

```
void operator() (
    NSTypeT< tree::BallTree > * ns ) const
```

Bichromatic neighbor search on the given NSType specialized for BallTrees.

39.364.4.7 operator>() [7/8]

```
void operator() (
    SpillKNN * ns ) const
```

Bichromatic neighbor search specialized for SPTrees.

39.364.4.8 operator>() [8/8]

```
void operator() (
    NSTypeT< tree::Octree > * ns ) const
```

Bichromatic neighbor search specialized for octrees.

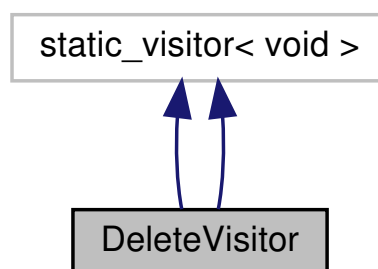
The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ **ns_model.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ **ra_model.hpp**

39.365 DeleteVisitor Class Reference

DeleteVisitor (p. 1595) deletes the given NSType instance.

Inheritance diagram for DeleteVisitor:



Public Member Functions

- `template<typename NSType >`
`void operator() (NSType *ns) const`
Delete the NSType object.
- `template<typename RAType >`
`void operator() (RAType *ra) const`
Delete the RAType Object.

39.365.1 Detailed Description

DeleteVisitor (p. 1595) deletes the given NSType instance.

DeleteVisitor (p. 1595) deletes the give RAType Instance.

Definition at line 229 of file `ns_model.hpp`.

39.365.2 Member Function Documentation

39.365.2.1 `operator()` [1/2]

```
void operator() (
    NSType * ns ) const
```

Delete the NSType object.

39.365.2.2 `operator()` [2/2]

```
void operator() (
    RAType * ra ) const
```

Delete the RAType Object.

The documentation for this class was generated from the following files:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ ns_model.hpp`
- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ ra_model.hpp`

39.366 DrusillaSelect< MatType > Class Template Reference

Public Member Functions

- **DrusillaSelect** (const MatType &referenceSet, const size_t l, const size_t m)
*Construct the **DrusillaSelect** (p. 1597) object with the given reference set (this is the set that will be searched).*
- **DrusillaSelect** (const size_t l, const size_t m)
*Construct the **DrusillaSelect** (p. 1597) object with no given reference set.*
- const arma::Col< size_t > & **CandidateIndices** () const
Access the indices of points in the candidate set.
- arma::Col< size_t > & **CandidateIndices** ()
Modify the indices of points in the candidate set. Be careful!
- const MatType & **CandidateSet** () const
Access the candidate set.
- MatType & **CandidateSet** ()
Modify the candidate set. Be careful!
- void **Search** (const MatType &querySet, const size_t k, arma::Mat< size_t > &neighbors, arma::mat &distances)
Search for the k furthest neighbors of the given query set.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the model.
- void **Train** (const MatType &referenceSet, const size_t l=0, const size_t m=0)
Build the set of candidate points on the given reference set.

39.366.1 Detailed Description

```
template<typename MatType = arma::mat>
class mpack::neighbor::DrusillaSelect< MatType >
```

Definition at line 39 of file drusilla_select.hpp.

39.366.2 Constructor & Destructor Documentation

39.366.2.1 DrusillaSelect() [1/2]

```
DrusillaSelect (
    const MatType & referenceSet,
    const size_t l,
    const size_t m )
```

Construct the **DrusillaSelect** (p. 1597) object with the given reference set (this is the set that will be searched).

The resulting set of candidate points that will be searched at query time will have size l*m.

Parameters

<i>referenceSet</i>	Set of reference data.
<i>l</i>	Number of projections.
<i>m</i>	Number of elements to store for each projection.

39.366.2.2 **DrusillaSelect()** [2/2]

```
DrusillaSelect (
    const size_t l,
    const size_t m )
```

Construct the **DrusillaSelect** (p. 1597) object with no given reference set.

Be sure to call **Train()** (p. 1600) before calling **Search()** (p. 1599)!

Parameters

<i>l</i>	Number of projections.
<i>m</i>	Number of elements to store for each projection.

39.366.3 Member Function Documentation

39.366.3.1 **CandidateIndices()** [1/2]

```
const arma::Col<size_t>& CandidateIndices ( ) const [inline]
```

Access the indices of points in the candidate set.

Definition at line 108 of file drusilla_select.hpp.

39.366.3.2 **CandidateIndices()** [2/2]

```
arma::Col<size_t>& CandidateIndices ( ) [inline]
```

Modify the indices of points in the candidate set. Be careful!

Definition at line 110 of file drusilla_select.hpp.

39.366.3.3 CandidateSet() [1/2]

```
const MatType& CandidateSet ( ) const [inline]
```

Access the candidate set.

Definition at line 103 of file drusilla_select.hpp.

39.366.3.4 CandidateSet() [2/2]

```
MatType& CandidateSet ( ) [inline]
```

Modify the candidate set. Be careful!

Definition at line 105 of file drusilla_select.hpp.

39.366.3.5 Search()

```
void Search (
    const MatType & querySet,
    const size_t k,
    arma::Mat< size_t > & neighbors,
    arma::mat & distances )
```

Search for the k furthest neighbors of the given query set.

(The query set can contain just one point: that is okay.) The results will be stored in the given neighbors and distances matrices, in the same format as the **NeighborSearch** (p. 1627) and **LSHSearch** (p. 1609) classes. That is, each column in the neighbors and distances matrices will refer to a single query point, and the k'th row in that column will refer to the k'th candidate neighbor or distance for that query point.

Parameters

<i>querySet</i>	Set of query points to search.
<i>k</i>	Number of furthest neighbors to search for.
<i>neighbors</i>	Matrix to store resulting neighbors in.
<i>distances</i>	Matrix to store resulting distances in.

39.366.3.6 serialize()

```
void serialize (
```

```
Archive & ar,
const unsigned int )
```

Serialize the model.

39.366.3.7 Train()

```
void Train (
    const MatType & referenceSet,
    const size_t l = 0,
    const size_t m = 0 )
```

Build the set of candidate points on the given reference set.

If *l* and *m* are left unspecified, then the values set in the constructor will be used instead.

Parameters

<i>referenceSet</i>	Set to extract candidate points from.
<i>l</i>	Number of projections.
<i>m</i>	Number of elements to store for each projection.

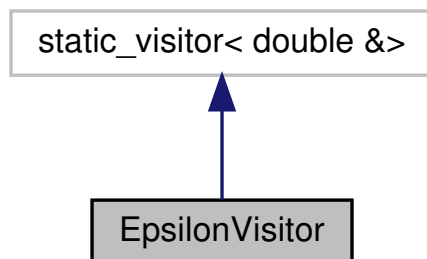
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/approx_kfn/ **drusilla_select.hpp**

39.367 EpsilonVisitor Class Reference

EpsilonVisitor (p. 1600) exposes the Epsilon method of the given NSType.

Inheritance diagram for EpsilonVisitor:



Public Member Functions

- `template<typename NSType >
double & operator() (NSType *ns) const`
Return epsilon, the approximation parameter.

39.367.1 Detailed Description

EpsilonVisitor (p. 1600) exposes the Epsilon method of the given NSType.

Definition at line 207 of file ns_model.hpp.

39.367.2 Member Function Documentation

39.367.2.1 operator>()

```
double& operator() (
    NSType * ns ) const
```

Return epsilon, the approximation parameter.

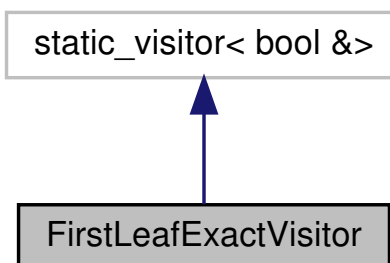
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ **ns_model.hpp**

39.368 FirstLeafExactVisitor Class Reference

Exposes the FirstLeafExact() method of the given RType.

Inheritance diagram for FirstLeafExactVisitor:



Public Member Functions

- template<typename RType >
bool & **operator()** (RType *ra) const

39.368.1 Detailed Description

Exposes the FirstLeafExact() method of the given RType.

Definition at line 177 of file ra_model.hpp.

39.368.2 Member Function Documentation

39.368.2.1 operator>()

```
bool& operator() (
    RType * ra ) const
```

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ **ra_model.hpp**

39.369 FurthestNS Class Reference

This class implements the necessary methods for the SortPolicy template parameter of the **NeighborSearch** (p. 1627) class.

Static Public Member Functions

- static double **BestDistance** ()
Return what should represent the best possible distance with this particular sort policy.
- template<typename TreeType >
static double **BestNodeToNodeDistance** (const TreeType *queryNode, const TreeType *referenceNode)
Return the best possible distance between two nodes.
- template<typename TreeType >
static double **BestNodeToNodeDistance** (const TreeType *queryNode, const TreeType *referenceNode, const double centerToCenterDistance)
Return the best possible distance between two nodes, given that the distance between the centers of the two nodes has already been calculated.

- `template<typename TreeType >`
`static double BestNodeToNodeDistance (const TreeType *queryNode, const TreeType *referenceNode, const TreeType *referenceChildNode, const double centerToCenterDistance)`
Return the best possible distance between the query node and the reference child node given the base case distance between the query node and the reference node.
- `template<typename VecType , typename TreeType >`
`static double BestPointToNodeDistance (const VecType &queryPoint, const TreeType *referenceNode)`
Return the best possible distance between a node and a point.
- `template<typename VecType , typename TreeType >`
`static double BestPointToNodeDistance (const VecType &queryPoint, const TreeType *referenceNode, const double pointToCenterDistance)`
Return the best possible distance between a point and a node, given that the distance between the point and the center of the node has already been calculated.
- `static double CombineBest (const double a, const double b)`
Return the best combination of the two distances.
- `static double CombineWorst (const double a, const double b)`
Return the worst combination of the two distances.
- `static double ConvertToDistance (const double score)`
Convert the given score back to a distance.
- `static double ConvertToScore (const double distance)`
Convert the given distance to a score.
- `template<typename VecType , typename TreeType >`
`static size_t GetBestChild (const VecType &queryPoint, TreeType &referenceNode)`
Return the best child according to this sort policy.
- `template<typename TreeType >`
`static size_t GetBestChild (const TreeType &queryNode, TreeType &referenceNode)`
Return the best child according to this sort policy.
- `static bool IsBetter (const double value, const double ref)`
Return whether or not value is "better" than ref.
- `static double Relax (const double value, const double epsilon)`
Return the given value relaxed.
- `static double WorstDistance ()`
Return what should represent the worst possible distance with this particular sort policy.

39.369.1 Detailed Description

This class implements the necessary methods for the SortPolicy template parameter of the **NeighborSearch** (p. 1627) class.

The sorting policy here is that the minimum distance is the best (so, when used with **NeighborSearch** (p. 1627), the output is furthest neighbors).

Definition at line 27 of file `furthest_neighbor_sort.hpp`.

39.369.2 Member Function Documentation

39.369.2.1 BestDistance()

```
static double BestDistance ( ) [inline], [static]
```

Return what should represent the best possible distance with this particular sort policy.

In our case, this should be the maximum possible distance, DBL_MAX.

Returns

DBL_MAX

Definition at line 138 of file `furthest_neighbor_sort.hpp`.

39.369.2.2 BestNodeToNodeDistance() [1/3]

```
static double BestNodeToNodeDistance (
    const TreeType * queryNode,
    const TreeType * referenceNode ) [static]
```

Return the best possible distance between two nodes.

In our case, this is the maximum distance between the two tree nodes using the given distance function.

Referenced by `FurthestNS::IsBetter()`.

39.369.2.3 BestNodeToNodeDistance() [2/3]

```
static double BestNodeToNodeDistance (
    const TreeType * queryNode,
    const TreeType * referenceNode,
    const double centerToCenterDistance ) [static]
```

Return the best possible distance between two nodes, given that the distance between the centers of the two nodes has already been calculated.

This is used in conjunction with trees that have self-children (like cover trees).

39.369.2.4 BestNodeToNodeDistance() [3/3]

```
static double BestNodeToNodeDistance (
    const TreeType * queryNode,
    const TreeType * referenceNode,
    const TreeType * referenceChildNode,
    const double centerToCenterDistance ) [static]
```

Return the best possible distance between the query node and the reference child node given the base case distance between the query node and the reference node.

`TreeType::ParentDistance()` must be implemented to use this.

Parameters

<i>queryNode</i>	Query node.
<i>referenceNode</i>	Reference node.
<i>referenceChildNode</i>	Child of reference node which is being inspected.
<i>centerToCenterDistance</i>	Distance between centers of query node and reference node.

39.369.2.5 BestPointToNodeDistance() [1/2]

```
static double BestPointToNodeDistance (
    const VecType & queryPoint,
    const TreeType * referenceNode ) [static]
```

Return the best possible distance between a node and a point.

In our case, this is the maximum distance between the tree node and the point using the given distance function.

Referenced by FurthestNS::IsBetter().

39.369.2.6 BestPointToNodeDistance() [2/2]

```
static double BestPointToNodeDistance (
    const VecType & queryPoint,
    const TreeType * referenceNode,
    const double pointToCenterDistance ) [static]
```

Return the best possible distance between a point and a node, given that the distance between the point and the center of the node has already been calculated.

This is used in conjunction with trees that have self-children (like cover trees).

39.369.2.7 CombineBest()

```
static double CombineBest (
    const double a,
    const double b ) [inline], [static]
```

Return the best combination of the two distances.

Definition at line 143 of file furthest_neighbor_sort.hpp.

39.369.2.8 CombineWorst()

```
static double CombineWorst (  
    const double a,  
    const double b ) [inline], [static]
```

Return the worst combination of the two distances.

Definition at line 153 of file furthest_neighbor_sort.hpp.

39.369.2.9 ConvertToDistance()

```
static double ConvertToDistance (  
    const double score ) [inline], [static]
```

Convert the given score back to a distance.

This is the inverse of the operation of converting a distance to a score, and again, for furthest neighbor search, corresponds to inverting the value.

Definition at line 193 of file furthest_neighbor_sort.hpp.

References FurthestNS::ConvertToScore().

39.369.2.10 ConvertToScore()

```
static double ConvertToScore (  
    const double distance ) [inline], [static]
```

Convert the given distance to a score.

Lower scores are better, but for furthest neighbor search, larger distances are better. Therefore we must invert the given distance.

Definition at line 178 of file furthest_neighbor_sort.hpp.

Referenced by FurthestNS::ConvertToDistance().

39.369.2.11 GetBestChild() [1/2]

```
static size_t GetBestChild (
    const VecType & queryPoint,
    TreeType & referenceNode ) [inline], [static]
```

Return the best child according to this sort policy.

In this case it will return the one with the maximum distance.

Definition at line 107 of file furthest_neighbor_sort.hpp.

39.369.2.12 GetBestChild() [2/2]

```
static size_t GetBestChild (
    const TreeType & queryNode,
    TreeType & referenceNode ) [inline], [static]
```

Return the best child according to this sort policy.

In this case it will return the one with the maximum distance.

Definition at line 117 of file furthest_neighbor_sort.hpp.

39.369.2.13 IsBetter()

```
static bool IsBetter (
    const double value,
    const double ref ) [inline], [static]
```

Return whether or not value is "better" than ref.

In this case, that means that the value is greater than or equal to the reference.

Parameters

<i>value</i>	Value to compare
<i>ref</i>	Value to compare with

Returns

bool indicating whether or not (value >= ref).

Definition at line 39 of file furthest_neighbor_sort.hpp.

References `FurthestNS::BestNodeToNodeDistance()`, and `FurthestNS::BestPointToNodeDistance()`.

39.369.2.14 `Relax()`

```
static double Relax (
    const double value,
    const double epsilon ) [inline], [static]
```

Return the given value relaxed.

Parameters

<i>value</i>	Value to relax.
<i>epsilon</i>	Relative error (non-negative).

Returns

double Value relaxed.

Definition at line 164 of file `furthest_neighbor_sort.hpp`.

39.369.2.15 `WorstDistance()`

```
static double WorstDistance ( ) [inline], [static]
```

Return what should represent the worst possible distance with this particular sort policy.

In our case, this should be the minimum possible distance, 0.

Returns

0

Definition at line 129 of file `furthest_neighbor_sort.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/sort_policies/furthest_neighbor_sort.hpp`↩

39.370 LSHSearch< SortPolicy > Class Template Reference

The **LSHSearch** (p. 1609) class; this class builds a hash on the reference set and uses this hash to compute the distance-approximate nearest-neighbors of the given queries.

Public Member Functions

- **LSHSearch** (arma::mat referenceSet, const arma::cube &projections, const double hashWidth=0.0, const size_t secondHashSize=99901, const size_t bucketSize=500)
This function initializes the LSH class.
- **LSHSearch** (arma::mat referenceSet, const size_t numProj, const size_t numTables, const double hashWidth=0.0, const size_t secondHashSize=99901, const size_t bucketSize=500)
This function initializes the LSH class.
- **LSHSearch** ()
Create an untrained LSH model.
- **LSHSearch** (const **LSHSearch** &other)
Copy the given LSH model.
- **LSHSearch** (**LSHSearch** &&other)
Take ownership of the given LSH model.
- size_t **BucketSize** () const
Get the bucket size of the second hash.
- size_t **DistanceEvaluations** () const
Return the number of distance evaluations performed.
- size_t & **DistanceEvaluations** ()
Modify the number of distance evaluations performed.
- size_t **NumProjections** () const
Get the number of projections.
- const arma::mat & **Offsets** () const
Get the offsets 'b' for each of the projections. (One 'b' per column.)
- **LSHSearch** & **operator=** (const **LSHSearch** &other)
Copy the given LSH model.
- **LSHSearch** & **operator=** (**LSHSearch** &&other)
Take ownership of the given LSH model.
- const arma::cube & **Projections** ()
Get the projection tables.
- void **Projections** (const arma::cube &projTables)
Change the projection tables (this retrains the LSH model).
- const arma::mat & **ReferenceSet** () const
Return the reference dataset.
- void **Search** (const arma::mat &querySet, const size_t k, arma::Mat< size_t > &resultingNeighbors, arma::mat &distances, const size_t numTablesToSearch=0, const size_t T=0)
Compute the nearest neighbors of the points in the given query set and store the output in the given matrices.
- void **Search** (const size_t k, arma::Mat< size_t > &resultingNeighbors, arma::mat &distances, const size_t numTablesToSearch=0, size_t T=0)
Compute the nearest neighbors and store the output in the given matrices.
- const std::vector< arma::Col< size_t > > & **SecondHashTable** () const

Get the second hash table.

- const arma::vec & **SecondHashWeights** () const

Get the weights of the second hash.

- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int version)

Serialize the LSH model.

- void **Train** (arma::mat referenceSet, const size_t numProj, const size_t numTables, const double hashWidth=0.0, const size_t secondHashSize=99901, const size_t bucketSize=500, const arma::cube &projection=arma::cube())

Train the LSH model on the given dataset.

Static Public Member Functions

- static double **ComputeRecall** (const arma::Mat< size_t > &foundNeighbors, const arma::Mat< size_t > &realNeighbors)

*Compute the recall (% of neighbors found) given the neighbors returned by **LSHSearch::Search** (p. 1615) and a "ground truth" set of neighbors.*

39.370.1 Detailed Description

```
template<typename SortPolicy = NearestNeighborSort>
class mlpack::neighbor::LSHSearch< SortPolicy >
```

The **LSHSearch** (p. 1609) class; this class builds a hash on the reference set and uses this hash to compute the distance-approximate nearest-neighbors of the given queries.

Template Parameters

<i>SortPolicy</i>	The sort policy for distances; see NearestNeighborSort.
-------------------	---

Definition at line 64 of file lsh_search.hpp.

39.370.2 Constructor & Destructor Documentation

39.370.2.1 LSHSearch() [1/5]

```
LSHSearch (
    arma::mat referenceSet,
    const arma::cube & projections,
    const double hashWidth = 0.0,
    const size_t secondHashSize = 99901,
    const size_t bucketSize = 500 )
```

This function initializes the LSH class.

It builds the hash on the reference set with 2-stable distributions. See the individual functions performing the hashing for details on how the hashing is done. In order to avoid copying the reference set, it is suggested to pass that parameter with `std::move()`.

Parameters

<i>referenceSet</i>	Set of reference points and the set of queries.
<i>projections</i>	Cube of projection tables. For a cube of size (a, b, c) we set numProj = a, numTables = c. b is the reference set dimensionality.
<i>hashWidth</i>	The width of hash for every table. If 0 (the default) is provided, then the hash width is automatically obtained by computing the average pairwise distance of 25 pairs. This should be a reasonable upper bound on the nearest-neighbor distance in general.
<i>secondHashSize</i>	The size of the second hash table. This should be a large prime number.
<i>bucketSize</i>	The size of the bucket in the second hash table. This is the maximum number of points that can be hashed into single bucket. A value of 0 indicates that there is no limit (so the second hash table can be arbitrarily large—be careful!).

39.370.2.2 LSHSearch() [2/5]

```

LSHSearch (
    arma::mat referenceSet,
    const size_t numProj,
    const size_t numTables,
    const double hashWidth = 0.0,
    const size_t secondHashSize = 99901,
    const size_t bucketSize = 500 )

```

This function initializes the LSH class.

It builds the hash one the reference set using the provided projections. See the individual functions performing the hashing for details on how the hashing is done. In order to avoid copying the reference set, consider passing the set with `std::move()`.

Parameters

<i>referenceSet</i>	Set of reference points and the set of queries.
<i>numProj</i>	Number of projections in each hash table (anything between 10-50 might be a decent choice).
<i>numTables</i>	Total number of hash tables (anything between 10-20 should suffice).
<i>hashWidth</i>	The width of hash for every table. If 0 (the default) is provided, then the hash width is automatically obtained by computing the average pairwise distance of 25 pairs. This should be a reasonable upper bound on the nearest-neighbor distance in general.
<i>secondHashSize</i>	The size of the second hash table. This should be a large prime number.
<i>bucketSize</i>	The size of the bucket in the second hash table. This is the maximum number of points that can be hashed into single bucket. A value of 0 indicates that there is no limit (so the second hash table can be arbitrarily large—be careful!).

39.370.2.3 LSHSearch() [3/5]

```
LSHSearch ( )
```

Create an untrained LSH model.

Be sure to call **Train()** (p. 1617) before calling **Search()** (p. 1615); otherwise, an exception will be thrown when **Search()** (p. 1615) is called.

39.370.2.4 LSHSearch() [4/5]

```
LSHSearch (
    const LSHSearch< SortPolicy > & other )
```

Copy the given LSH model.

Parameters

<i>other</i>	Other LSH model to copy.
--------------	--------------------------

39.370.2.5 LSHSearch() [5/5]

```
LSHSearch (
    LSHSearch< SortPolicy > && other )
```

Take ownership of the given LSH model.

Parameters

<i>other</i>	Other LSH model to take ownership of.
--------------	---------------------------------------

39.370.3 Member Function Documentation**39.370.3.1 BucketSize()**

```
size_t BucketSize ( ) const [inline]
```

Get the bucket size of the second hash.

Definition at line 281 of file lsh_search.hpp.

39.370.3.2 ComputeRecall()

```
static double ComputeRecall (
    const arma::Mat< size_t > & foundNeighbors,
    const arma::Mat< size_t > & realNeighbors ) [static]
```

Compute the recall (% of neighbors found) given the neighbors returned by **LSHSearch::Search** (p. 1615) and a "ground truth" set of neighbors.

The recall returned will be in the range [0, 1].

Parameters

<i>foundNeighbors</i>	Set of neighbors to compute recall of.
<i>realNeighbors</i>	Set of "ground truth" neighbors to compute recall against.

39.370.3.3 DistanceEvaluations() [1/2]

```
size_t DistanceEvaluations ( ) const [inline]
```

Return the number of distance evaluations performed.

Definition at line 264 of file lsh_search.hpp.

39.370.3.4 DistanceEvaluations() [2/2]

```
size_t& DistanceEvaluations ( ) [inline]
```

Modify the number of distance evaluations performed.

Definition at line 266 of file lsh_search.hpp.

39.370.3.5 NumProjections()

```
size_t NumProjections ( ) const [inline]
```

Get the number of projections.

Definition at line 272 of file lsh_search.hpp.

39.370.3.6 Offsets()

```
const arma::mat& Offsets ( ) const [inline]
```

Get the offsets 'b' for each of the projections. (One 'b' per column.)

Definition at line 275 of file lsh_search.hpp.

39.370.3.7 operator=() [1/2]

```
LSHSearch& operator= (
    const LSHSearch< SortPolicy > & other )
```

Copy the given LSH model.

Parameters

<i>other</i>	Other LSH model to copy.
--------------	--------------------------

39.370.3.8 operator=() [2/2]

```
LSHSearch& operator= (
    LSHSearch< SortPolicy > && other )
```

Take ownership of the given LSH model.

Parameters

<i>other</i>	Other LSH model to take ownership of.
--------------	---------------------------------------

39.370.3.9 Projections() [1/2]

```
const arma::cube& Projections ( ) [inline]
```

Get the projection tables.

Definition at line 288 of file lsh_search.hpp.

39.370.3.10 Projections() [2/2]

```
void Projections (
    const arma::cube & projTables ) [inline]
```

Change the projection tables (this retrains the LSH model).

Definition at line 291 of file lsh_search.hpp.

References BOOST_TEMPLATE_CLASS_VERSION(), and LSHSearch< SortPolicy >::Train().

39.370.3.11 ReferenceSet()

```
const arma::mat& ReferenceSet ( ) const [inline]
```

Return the reference dataset.

Definition at line 269 of file lsh_search.hpp.

39.370.3.12 Search() [1/2]

```
void Search (
    const arma::mat & querySet,
    const size_t k,
    arma::Mat< size_t > & resultingNeighbors,
    arma::mat & distances,
    const size_t numTablesToSearch = 0,
    const size_t T = 0 )
```

Compute the nearest neighbors of the points in the given query set and store the output in the given matrices.

The matrices will be set to the size of n columns by k rows, where n is the number of points in the query dataset and k is the number of neighbors being searched for.

Parameters

<i>querySet</i>	Set of query points.
<i>k</i>	Number of neighbors to search for.
<i>resultingNeighbors</i>	Matrix storing lists of neighbors for each query point.
<i>distances</i>	Matrix storing distances of neighbors for each query point.
<i>numTablesToSearch</i>	This parameter allows the user to have control over the number of hash tables to be searched. This allows the user to pick the number of tables it can afford for the time available without having to build hashing for every table size. By default, this is set to zero in which case all tables are considered.
<i>T</i>	The number of additional probing bins to examine with multiprobe LSH. If $T = 0$, classic single-probe LSH is run (default).

39.370.3.13 Search() [2/2]

```
void Search (
    const size_t k,
    arma::Mat< size_t > & resultingNeighbors,
    arma::mat & distances,
    const size_t numTablesToSearch = 0,
    size_t T = 0 )
```

Compute the nearest neighbors and store the output in the given matrices.

The matrices will be set to the size of n columns by k rows, where n is the number of points in the query dataset and k is the number of neighbors being searched for.

Parameters

<i>k</i>	Number of neighbors to search for.
<i>resultingNeighbors</i>	Matrix storing lists of neighbors for each query point.
<i>distances</i>	Matrix storing distances of neighbors for each query point.
<i>numTablesToSearch</i>	This parameter allows the user to have control over the number of hash tables to be searched. This allows the user to pick the number of tables it can afford for the time available without having to build hashing for every table size. By default, this is set to zero in which case all tables are considered.

39.370.3.14 SecondHashTable()

```
const std::vector<arma::Col<size_t> >& SecondHashTable ( ) const [inline]
```

Get the second hash table.

Definition at line 284 of file `lsh_search.hpp`.

39.370.3.15 SecondHashWeights()

```
const arma::vec& SecondHashWeights ( ) const [inline]
```

Get the weights of the second hash.

Definition at line 278 of file lsh_search.hpp.

39.370.3.16 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int version )
```

Serialize the LSH model.

Parameters

<i>ar</i>	Archive to serialize to.
-----------	--------------------------

39.370.3.17 Train()

```
void Train (
    arma::mat referenceSet,
    const size_t numProj,
    const size_t numTables,
    const double hashWidth = 0.0,
    const size_t secondHashSize = 99901,
    const size_t bucketSize = 500,
    const arma::cube & projection = arma::cube() )
```

Train the LSH model on the given dataset.

If a correctly-sized projection cube is not provided, this means building new hash tables. Otherwise, we use the projections provided by the user. In order to avoid copying the reference set, consider passing that parameter with `std::move()`.

Parameters

<i>referenceSet</i>	Set of reference points and the set of queries.
<i>numProj</i>	Number of projections in each hash table (anything between 10-50 might be a decent choice).
<i>numTables</i>	Total number of hash tables (anything between 10-20 should suffice).
<i>hashWidth</i>	The width of hash for every table. If 0 (the default) is provided, then the hash width is automatically obtained by computing the average pairwise distance of 25 pairs. This should be a reasonable upper bound on the nearest-neighbor distance in general.

Parameters

<i>secondHashSize</i>	The size of the second hash table. This should be a large prime number.
<i>bucketSize</i>	The size of the bucket in the second hash table. This is the maximum number of points that can be hashed into single bucket. A value of 0 indicates that there is no limit (so the second hash table can be arbitrarily large—be careful!).
<i>projections</i>	Cube of projection tables. For a cube of size (a, b, c) we set numProj = a, numTables = c. b is the reference set dimensionality.

Referenced by LSHSearch< SortPolicy >::Projections().

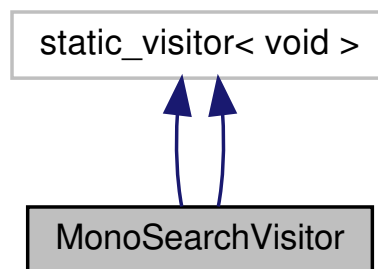
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lsh/ **lsh_search.hpp**

39.371 MonoSearchVisitor Class Reference

MonoSearchVisitor (p. 1618) executes a monochromatic neighbor search on the given NSType.

Inheritance diagram for MonoSearchVisitor:



Public Member Functions

- **MonoSearchVisitor** (const size_t k, arma::Mat< size_t > &neighbors, arma::mat &distances)
Construct the **MonoSearchVisitor** (p. 1618) object with the given parameters.
- **MonoSearchVisitor** (const size_t k, arma::Mat< size_t > &neighbors, arma::mat &distances)
Construct the **MonoSearchVisitor** (p. 1618) object with the given parameters.
- template<typename RType >
void **operator()** (RType *ra) const
Perform monochromatic nearest neighbor search.
- template<typename NSType >
void **operator()** (NSType *ns) const
Perform monochromatic nearest neighbor search.

39.371.1 Detailed Description

MonoSearchVisitor (p. 1618) executes a monochromatic neighbor search on the given NType.

MonoSearchVisitor (p. 1618) executes a monochromatic neighbor search on the given RType.

We don't make any difference for different instantiations of NType.

We don't make any difference for different instantiation of RType.

Definition at line 48 of file ns_model.hpp.

39.371.2 Constructor & Destructor Documentation

39.371.2.1 MonoSearchVisitor() [1/2]

```
MonoSearchVisitor (  
    const size_t k,  
    arma::Mat< size_t > & neighbors,  
    arma::mat & distances ) [inline]
```

Construct the **MonoSearchVisitor** (p. 1618) object with the given parameters.

Definition at line 64 of file ns_model.hpp.

39.371.2.2 MonoSearchVisitor() [2/2]

```
MonoSearchVisitor (  
    const size_t k,  
    arma::Mat< size_t > & neighbors,  
    arma::mat & distances ) [inline]
```

Construct the **MonoSearchVisitor** (p. 1618) object with the given parameters.

Definition at line 59 of file ra_model.hpp.

39.371.3 Member Function Documentation

39.371.3.1 operator>() [1/2]

```
void operator() (
    RAType * ra ) const
```

Perform monochromatic nearest neighbor search.

39.371.3.2 operator>() [2/2]

```
void operator() (
    NSType * ns ) const
```

Perform monochromatic nearest neighbor search.

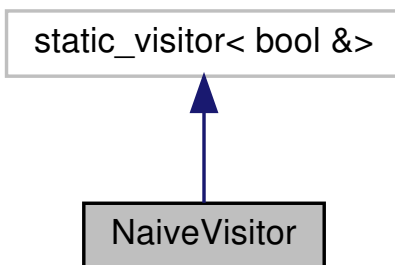
The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ **ns_model.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ **ra_model.hpp**

39.372 NaiveVisitor Class Reference

NaiveVisitor (p. 1620) exposes the Naive() method of the given RAType.

Inheritance diagram for NaiveVisitor:

**Public Member Functions**

- template<typename RAType >
bool & **operator()** (**RAType** *ra) const

*Get a reference to the naive parameter of the given **RASearch** (p. 1681) object.*

39.372.1 Detailed Description

NaiveVisitor (p. 1620) exposes the `Naive()` method of the given `RAType`.

Definition at line 252 of file `ra_model.hpp`.

39.372.2 Member Function Documentation

39.372.2.1 `operator()()`

```
bool& operator() (
    RAType * ra ) const
```

Get a reference to the naive parameter of the given **RASearch** (p. 1681) object.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ ra_model.hpp`

39.373 NearestNS Class Reference

This class implements the necessary methods for the `SortPolicy` template parameter of the **NeighborSearch** (p. 1627) class.

Static Public Member Functions

- static double **BestDistance** ()
Return what should represent the best possible distance with this particular sort policy.
- template<typename TreeType >
static double **BestNodeToNodeDistance** (const TreeType *queryNode, const TreeType *referenceNode)
Return the best possible distance between two nodes.
- template<typename TreeType >
static double **BestNodeToNodeDistance** (const TreeType *queryNode, const TreeType *referenceNode, const double centerToCenterDistance)
Return the best possible distance between two nodes, given that the distance between the centers of the two nodes has already been calculated.
- template<typename TreeType >
static double **BestNodeToNodeDistance** (const TreeType *queryNode, const TreeType *referenceNode, const TreeType *referenceChildNode, const double centerToCenterDistance)
Return the best possible distance between the query node and the reference child node given the base case distance between the query node and the reference node.

- `template<typename VecType , typename TreeType >`
`static double BestPointToNodeDistance (const VecType &queryPoint, const TreeType *referenceNode)`
Return the best possible distance between a node and a point.
- `template<typename VecType , typename TreeType >`
`static double BestPointToNodeDistance (const VecType &queryPoint, const TreeType *referenceNode, const double pointToCenterDistance)`
Return the best possible distance between a point and a node, given that the distance between the point and the center of the node has already been calculated.
- `static double CombineBest (const double a, const double b)`
Return the best combination of the two distances.
- `static double CombineWorst (const double a, const double b)`
Return the worst combination of the two distances.
- `static double ConvertToDistance (const double score)`
Convert the given score to a distance.
- `static double ConvertToScore (const double distance)`
Convert the given distance into a score.
- `template<typename VecType , typename TreeType >`
`static size_t GetBestChild (const VecType &queryPoint, TreeType &referenceNode)`
Return the best child according to this sort policy.
- `template<typename TreeType >`
`static size_t GetBestChild (const TreeType &queryNode, TreeType &referenceNode)`
Return the best child according to this sort policy.
- `static bool IsBetter (const double value, const double ref)`
Return whether or not value is "better" than ref.
- `static double Relax (const double value, const double epsilon)`
Return the given value relaxed.
- `static double WorstDistance ()`
Return what should represent the worst possible distance with this particular sort policy.

39.373.1 Detailed Description

This class implements the necessary methods for the SortPolicy template parameter of the **NeighborSearch** (p. 1627) class.

The sorting policy here is that the minimum distance is the best (so, when used with **NeighborSearch** (p. 1627), the output is nearest neighbors).

This class is also meant to serve as a guide to implement a custom SortPolicy. All of the methods implemented here must be implemented by any other SortPolicy classes.

Definition at line 31 of file nearest_neighbor_sort.hpp.

39.373.2 Member Function Documentation

39.373.2.1 BestDistance()

```
static double BestDistance ( ) [inline], [static]
```

Return what should represent the best possible distance with this particular sort policy.

In our case, this should be the minimum possible distance, 0.0.

Returns

0.0

Definition at line 142 of file nearest_neighbor_sort.hpp.

39.373.2.2 BestNodeToNodeDistance() [1/3]

```
static double BestNodeToNodeDistance (
    const TreeType * queryNode,
    const TreeType * referenceNode ) [static]
```

Return the best possible distance between two nodes.

In our case, this is the minimum distance between the two tree nodes using the given distance function.

Referenced by NearestNS::IsBetter().

39.373.2.3 BestNodeToNodeDistance() [2/3]

```
static double BestNodeToNodeDistance (
    const TreeType * queryNode,
    const TreeType * referenceNode,
    const double centerToCenterDistance ) [static]
```

Return the best possible distance between two nodes, given that the distance between the centers of the two nodes has already been calculated.

This is used in conjunction with trees that have self-children (like cover trees).

39.373.2.4 BestNodeToNodeDistance() [3/3]

```
static double BestNodeToNodeDistance (
    const TreeType * queryNode,
    const TreeType * referenceNode,
    const TreeType * referenceChildNode,
    const double centerToCenterDistance ) [static]
```

Return the best possible distance between the query node and the reference child node given the base case distance between the query node and the reference node.

TreeType::ParentDistance() must be implemented to use this.

Parameters

<i>queryNode</i>	Query node.
<i>referenceNode</i>	Reference node.
<i>referenceChildNode</i>	Child of reference node which is being inspected.
<i>centerToCenterDistance</i>	Distance between centers of query node and reference node.

39.373.2.5 BestPointToNodeDistance() [1/2]

```
static double BestPointToNodeDistance (
    const VecType & queryPoint,
    const TreeType * referenceNode ) [static]
```

Return the best possible distance between a node and a point.

In our case, this is the minimum distance between the tree node and the point using the given distance function.

Referenced by NearestNS::IsBetter().

39.373.2.6 BestPointToNodeDistance() [2/2]

```
static double BestPointToNodeDistance (
    const VecType & queryPoint,
    const TreeType * referenceNode,
    const double pointToCenterDistance ) [static]
```

Return the best possible distance between a point and a node, given that the distance between the point and the center of the node has already been calculated.

This is used in conjunction with trees that have self-children (like cover trees).

39.373.2.7 CombineBest()

```
static double CombineBest (
    const double a,
    const double b ) [inline], [static]
```

Return the best combination of the two distances.

Definition at line 147 of file nearest_neighbor_sort.hpp.

39.373.2.8 CombineWorst()

```
static double CombineWorst (
    const double a,
    const double b ) [inline], [static]
```

Return the worst combination of the two distances.

Definition at line 155 of file nearest_neighbor_sort.hpp.

39.373.2.9 ConvertToDistance()

```
static double ConvertToDistance (
    const double score ) [inline], [static]
```

Convert the given score to a distance.

This is the inverse of the operation provided by **ConvertToScore()** (p. 1625). For nearest neighbor search, there is no need for any change.

Definition at line 192 of file nearest_neighbor_sort.hpp.

39.373.2.10 ConvertToScore()

```
static double ConvertToScore (
    const double distance ) [inline], [static]
```

Convert the given distance into a score.

Lower scores are better, so in the case of nearest neighbor sort where lower distances are better, we just return the distance.

Definition at line 182 of file nearest_neighbor_sort.hpp.

39.373.2.11 GetBestChild() [1/2]

```
static size_t GetBestChild (
    const VecType & queryPoint,
    TreeType & referenceNode ) [inline], [static]
```

Return the best child according to this sort policy.

In this case it will return the one with the minimum distance.

Definition at line 111 of file nearest_neighbor_sort.hpp.

39.373.2.12 GetBestChild() [2/2]

```
static size_t GetBestChild (
    const TreeType & queryNode,
    TreeType & referenceNode ) [inline], [static]
```

Return the best child according to this sort policy.

In this case it will return the one with the minimum distance.

Definition at line 121 of file nearest_neighbor_sort.hpp.

39.373.2.13 IsBetter()

```
static bool IsBetter (
    const double value,
    const double ref ) [inline], [static]
```

Return whether or not value is "better" than ref.

In this case, that means that the value is less than or equal to the reference.

Parameters

<i>value</i>	Value to compare
<i>ref</i>	Value to compare with

Returns

bool indicating whether or not ($value \leq ref$).

Definition at line 43 of file nearest_neighbor_sort.hpp.

References NearestNS::BestNodeToNodeDistance(), and NearestNS::BestPointToNodeDistance().

39.373.2.14 Relax()

```
static double Relax (
    const double value,
    const double epsilon ) [inline], [static]
```

Return the given value relaxed.

Parameters

<i>value</i>	Value to relax.
<i>epsilon</i>	Relative error (non-negative).

Returns

double Value relaxed.

Definition at line 170 of file nearest_neighbor_sort.hpp.

39.373.2.15 WorstDistance()

```
static double WorstDistance ( ) [inline], [static]
```

Return what should represent the worst possible distance with this particular sort policy.

In our case, this should be the maximum possible distance, DBL_MAX.

Returns

DBL_MAX

Definition at line 133 of file nearest_neighbor_sort.hpp.

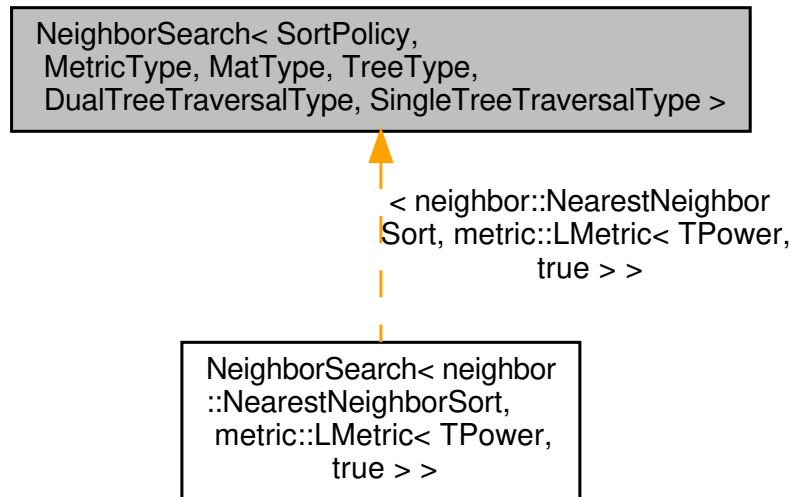
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/sort_policies/ **nearest_neighbor_↵
_sort.hpp**

39.374 NeighborSearch< SortPolicy, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTreeTraversalType > Class Template Reference↵

The **NeighborSearch** (p. 1627) class is a template class for performing distance-based neighbor searches.

Inheritance diagram for NeighborSearch< SortPolicy, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTreeTraversalType >:



Public Types

- typedef TreeType< MetricType, **NeighborSearchStat**< SortPolicy >, MatType > **Tree**
Convenience typedef.

Public Member Functions

- **NeighborSearch** (MatType referenceSet, const **NeighborSearchMode** mode= **DUAL_TREE_MODE**, const double epsilon=0, const MetricType metric=MetricType())
*Initialize the **NeighborSearch** (p. 1627) object, passing a reference dataset (this is the dataset which is searched).*
- **NeighborSearch** (**Tree** referenceTree, const **NeighborSearchMode** mode= **DUAL_TREE_MODE**, const double epsilon=0, const MetricType metric=MetricType())
*Initialize the **NeighborSearch** (p. 1627) object with a copy of the given pre-constructed reference tree (this is the tree built on the points that will be searched).*
- **NeighborSearch** (const **NeighborSearchMode** mode= **DUAL_TREE_MODE**, const double epsilon=0, const MetricType metric=MetricType())
*Create a **NeighborSearch** (p. 1627) object without any reference data.*
- **NeighborSearch** (const **NeighborSearch** &other)
*Construct the **NeighborSearch** (p. 1627) object by copying the given **NeighborSearch** (p. 1627) object.*
- **NeighborSearch** (**NeighborSearch** &&other)
*Construct the **NeighborSearch** (p. 1627) object by taking ownership of the given **NeighborSearch** (p. 1627) object.*
- **~NeighborSearch** ()
*Delete the **NeighborSearch** (p. 1627) object.*
- size_t **BaseCases** () const
Return the total number of base case evaluations performed during the last search.
- double **Epsilon** () const
Access the relative error to be considered in approximate search.

- double & **Epsilon** ()
Modify the relative error to be considered in approximate search.
- **NeighborSearch** & **operator=** (const **NeighborSearch** &other)
*Copy the given **NeighborSearch** (p. 1627) object.*
- **NeighborSearch** & **operator=** (**NeighborSearch** &&other)
*Take ownership of the given **NeighborSearch** (p. 1627) object.*
- const MatType & **ReferenceSet** () const
Access the reference dataset.
- const **Tree** & **ReferenceTree** () const
Access the reference tree.
- **Tree** & **ReferenceTree** ()
Modify the reference tree.
- size_t **Scores** () const
Return the number of node combination scores during the last search.
- void **Search** (const MatType &querySet, const size_t k, arma::Mat< size_t > &neighbors, arma::mat &distances)
For each point in the query set, compute the nearest neighbors and store the output in the given matrices.
- void **Search** (**Tree** &queryTree, const size_t k, arma::Mat< size_t > &neighbors, arma::mat &distances, bool sameSet=false)
Given a pre-built query tree, search for the nearest neighbors of each point in the query tree, storing the output in the given matrices.
- void **Search** (const size_t k, arma::Mat< size_t > &neighbors, arma::mat &distances)
Search for the nearest neighbors of every point in the reference set.
- **NeighborSearchMode** **SearchMode** () const
Access the search mode.
- **NeighborSearchMode** & **SearchMode** ()
Modify the search mode.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
*Serialize the **NeighborSearch** (p. 1627) model.*
- void **Train** (MatType referenceSet)
Set the reference set to a new reference set, and build a tree if necessary.
- void **Train** (**Tree** referenceTree)
Set the reference tree to a new reference tree.

Static Public Member Functions

- static double **EffectiveError** (arma::mat &foundDistances, arma::mat &realDistances)
Calculate the average relative error (effective error) between the distances calculated and the true distances provided.
- static double **Recall** (arma::Mat< size_t > &foundNeighbors, arma::Mat< size_t > &realNeighbors)
Calculate the recall (% of neighbors found) given the list of found neighbors and the true set of neighbors.

39.374.1 Detailed Description

```
template<typename SortPolicy = NearestNeighborSort, typename MetricType = mlpack::metric::EuclideanDistance, typename MatType = arma::mat, template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType = tree::KDTree, template< typename RuleType > class DualTreeTraversalType = TreeType<MetricType, NeighborSearchStat<SortPolicy>, MatType>::template DualTreeTraverser, template< typename RuleType > class SingleTreeTraversalType = TreeType<MetricType, NeighborSearchStat<SortPolicy>, MatType>::template SingleTreeTraverser>
class mlpack::neighbor::NeighborSearch< SortPolicy, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTreeTraversalType >
```

The **NeighborSearch** (p. 1627) class is a template class for performing distance-based neighbor searches.

It takes a query dataset and a reference dataset (or just a reference dataset) and, for each point in the query dataset, finds the *k* neighbors in the reference dataset which have the 'best' distance according to a given sorting policy. A constructor is given which takes only a reference dataset, and if that constructor is used, the given reference dataset is also used as the query dataset.

The template parameters *SortPolicy* and *Metric* define the sort function used and the metric (distance function) used. More information on those classes can be found in the *NearestNeighborSort* class and the **kernel::ExampleKernel** (p. 1421) class.

Template Parameters

<i>SortPolicy</i>	The sort policy for distances; see <i>NearestNeighborSort</i> .
<i>MetricType</i>	The metric to use for computation.
<i>MatType</i>	The type of data matrix.
<i>TreeType</i>	The tree type to use; must adhere to the <i>TreeType</i> API.
<i>DualTreeTraversalType</i>	The type of dual tree traversal to use (defaults to the tree's default traverser).
<i>SingleTreeTraversalType</i>	The type of single tree traversal to use (defaults to the tree's default traverser).

Definition at line 83 of file *neighbor_search.hpp*.

39.374.2 Member Typedef Documentation

39.374.2.1 Tree

```
typedef TreeType<MetricType, NeighborSearchStat<SortPolicy>, MatType> Tree
```

Convenience typedef.

Definition at line 87 of file *neighbor_search.hpp*.

39.374.3 Constructor & Destructor Documentation

39.374.3.1 NeighborSearch() [1/5]

```
NeighborSearch (
    MatType referenceSet,
    const NeighborSearchMode mode = DUAL_TREE_MODE,
    const double epsilon = 0,
    const MetricType metric = MetricType() )
```

Initialize the **NeighborSearch** (p. 1627) object, passing a reference dataset (this is the dataset which is searched).

Optionally, perform the computation in a different mode. An initialized distance metric can be given, for cases where the metric has internal data (i.e. the distance::MahalanobisDistance class).

This method will move the matrices to internal copies, which are rearranged during tree-building. You can avoid creating an extra copy by pre-constructing the trees, passing std::move(yourReferenceSet).

Parameters

<i>referenceSet</i>	Set of reference points.
<i>mode</i>	Neighbor search mode.
<i>epsilon</i>	Relative approximate error (non-negative).
<i>metric</i>	An optional instance of the MetricType class.

39.374.3.2 NeighborSearch() [2/5]

```
NeighborSearch (
    Tree referenceTree,
    const NeighborSearchMode mode = DUAL_TREE_MODE,
    const double epsilon = 0,
    const MetricType metric = MetricType() )
```

Initialize the **NeighborSearch** (p. 1627) object with a copy of the given pre-constructed reference tree (this is the tree built on the points that will be searched).

Optionally, choose to use single-tree mode. Naive mode is not available as an option for this constructor. Additionally, an instantiated distance metric can be given, for cases where the distance metric holds data.

This method will copy the given tree. When copies must absolutely be avoided, you can avoid this copy, while taking ownership of the given tree, by passing std::move(yourReferenceTree)

Note

Mapping the points of the matrix back to their original indices is not done when this constructor is used, so if the tree type you are using maps points (like BinarySpaceTree), then you will have to perform the re-mapping manually.

Parameters

<i>referenceTree</i>	Pre-built tree for reference points.
<i>mode</i>	Neighbor search mode.
<i>epsilon</i>	Relative approximate error (non-negative).
<i>metric</i>	Instantiated distance metric.

39.374.3.3 NeighborSearch() [3/5]

```

NeighborSearch (
    const NeighborSearchMode mode = DUAL_TREE_MODE,
    const double epsilon = 0,
    const MetricType metric = MetricType() )

```

Create a **NeighborSearch** (p. 1627) object without any reference data.

If **Search()** (p. 1636) is called before a reference set is set with **Train()** (p. 1638), an exception will be thrown.

Parameters

<i>mode</i>	Neighbor search mode.
<i>epsilon</i>	Relative approximate error (non-negative).
<i>metric</i>	Instantiated metric.

39.374.3.4 NeighborSearch() [4/5]

```

NeighborSearch (
    const NeighborSearch< SortPolicy, MetricType, MatType, TreeType, DualTreeTraversal↵
Type, SingleTreeTraversalType > & other )

```

Construct the **NeighborSearch** (p. 1627) object by copying the given **NeighborSearch** (p. 1627) object.

Parameters

<i>other</i>	NeighborSearch (p. 1627) object to copy.
--------------	---

39.374.3.5 NeighborSearch() [5/5]

```

NeighborSearch (

```



```
NeighborSearch< SortPolicy, MetricType, MatType, TreeType, DualTreeTraversalType,
SingleTreeTraversalType > && other )
```

Construct the **NeighborSearch** (p. 1627) object by taking ownership of the given **NeighborSearch** (p. 1627) object.

Parameters

<i>other</i>	NeighborSearch (p. 1627) object to take ownership of.
--------------	--

39.374.3.6 ~NeighborSearch()

```
~ NeighborSearch ( )
```

Delete the **NeighborSearch** (p. 1627) object.

The tree is the only member we are responsible for deleting. The others will take care of themselves.

39.374.4 Member Function Documentation

39.374.4.1 BaseCases()

```
size_t BaseCases ( ) const [inline]
```

Return the total number of base case evaluations performed during the last search.

Definition at line 310 of file neighbor_search.hpp.

39.374.4.2 EffectiveError()

```
static double EffectiveError (
    arma::mat & foundDistances,
    arma::mat & realDistances ) [static]
```

Calculate the average relative error (effective error) between the distances calculated and the true distances provided.

The input matrices must have the same size.

Cases where the true distance is zero (the same point) or the calculated distance is SortPolicy::WorstDistance() (didn't find enough points) will be ignored.

Parameters

<i>foundDistances</i>	Matrix storing lists of calculated distances for each query point.
<i>realDistances</i>	Matrix storing lists of true best distances for each query point.

Returns

Average relative error.

39.374.4.3 Epsilon() [1/2]

```
double Epsilon ( ) const [inline]
```

Access the relative error to be considered in approximate search.

Definition at line 321 of file neighbor_search.hpp.

39.374.4.4 Epsilon() [2/2]

```
double& Epsilon ( ) [inline]
```

Modify the relative error to be considered in approximate search.

Definition at line 323 of file neighbor_search.hpp.

39.374.4.5 operator=() [1/2]

```
NeighborSearch& operator= (
    const NeighborSearch< SortPolicy, MetricType, MatType, TreeType, DualTreeTraversal↔
Type, SingleTreeTraversalType > & other )
```

Copy the given **NeighborSearch** (p. 1627) object.

Parameters

<i>other</i>	NeighborSearch (p. 1627) object to copy.
--------------	---

39.374.4.6 operator=() [2/2]

```
NeighborSearch& operator= (
    NeighborSearch< SortPolicy, MetricType, MatType, TreeType, DualTreeTraversalType,
    SingleTreeTraversalType > && other )
```

Take ownership of the given **NeighborSearch** (p. 1627) object.

Parameters

<i>other</i>	NeighborSearch (p. 1627) object to take ownership of.
--------------	--

39.374.4.7 Recall()

```
static double Recall (
    arma::Mat< size_t > & foundNeighbors,
    arma::Mat< size_t > & realNeighbors ) [static]
```

Calculate the recall (% of neighbors found) given the list of found neighbors and the true set of neighbors.

The recall returned will be in the range [0, 1].

Parameters

<i>foundNeighbors</i>	Matrix storing lists of calculated neighbors for each query point.
<i>realNeighbors</i>	Matrix storing lists of true best neighbors for each query point.

Returns

Recall.

39.374.4.8 ReferenceSet()

```
const MatType& ReferenceSet ( ) const [inline]
```

Access the reference dataset.

Definition at line 326 of file neighbor_search.hpp.

39.374.4.9 ReferenceTree() [1/2]

```
const Tree& ReferenceTree ( ) const [inline]
```

Access the reference tree.

Definition at line 329 of file neighbor_search.hpp.

39.374.4.10 ReferenceTree() [2/2]

```
Tree& ReferenceTree ( ) [inline]
```

Modify the reference tree.

Definition at line 331 of file neighbor_search.hpp.

39.374.4.11 Scores()

```
size_t Scores ( ) const [inline]
```

Return the number of node combination scores during the last search.

Definition at line 313 of file neighbor_search.hpp.

39.374.4.12 Search() [1/3]

```
void Search (
    const MatType & querySet,
    const size_t k,
    arma::Mat< size_t > & neighbors,
    arma::mat & distances )
```

For each point in the query set, compute the nearest neighbors and store the output in the given matrices.

The matrices will be set to the size of n columns by k rows, where n is the number of points in the query dataset and k is the number of neighbors being searched for.

If querySet contains only a few query points, the extra cost of building a tree on the points for dual-tree search may not be warranted, and it may be worthwhile to set singleMode = false (either in the constructor or with SingleMode()).

Parameters

<i>querySet</i>	Set of query points (can be just one point).
<i>k</i>	Number of neighbors to search for.
<i>neighbors</i>	Matrix storing lists of neighbors for each query point.
<i>distances</i>	Matrix storing distances of neighbors for each query point.

Referenced by LMetricSearch< TPower >::Search(), CosineSearch::Search(), and PearsonSearch::Search().

39.374.4.13 Search() [2/3]

```
void Search (
    Tree & queryTree,
    const size_t k,
    arma::Mat< size_t > & neighbors,
    arma::mat & distances,
    bool sameSet = false )
```

Given a pre-built query tree, search for the nearest neighbors of each point in the query tree, storing the output in the given matrices.

The matrices will be set to the size of n columns by k rows, where n is the number of points in the query dataset and k is the number of neighbors being searched for.

Note that if you are calling **Search()** (p. 1636) multiple times with a single query tree, you need to reset the bounds in the statistic of each query node, otherwise the result may be wrong! You can do this by calling TreeType::Stat()::Reset() on each node in the query tree.

Parameters

<i>queryTree</i>	Tree built on query points.
<i>k</i>	Number of neighbors to search for.
<i>neighbors</i>	Matrix storing lists of neighbors for each query point.
<i>distances</i>	Matrix storing distances of neighbors for each query point.
<i>sameSet</i>	Denotes whether or not the reference and query sets are the same.

39.374.4.14 Search() [3/3]

```
void Search (
    const size_t k,
    arma::Mat< size_t > & neighbors,
    arma::mat & distances )
```

Search for the nearest neighbors of every point in the reference set.

This is basically equivalent to calling any other overload of **Search()** (p. 1636) with the reference set as the query set; so, this lets you do all-k-nearest-neighbors search. The results are stored in the given matrices. The matrices will be set to the size of n columns by k rows, where n is the number of points in the query dataset and k is the number of neighbors being searched for.

Parameters

<i>k</i>	Number of neighbors to search for.
<i>neighbors</i>	Matrix storing lists of neighbors for each query point.
<i>distances</i>	Matrix storing distances of neighbors for each query point.

39.374.4.15 SearchMode() [1/2]

```
NeighborSearchMode SearchMode ( ) const [inline]
```

Access the search mode.

Definition at line 316 of file neighbor_search.hpp.

39.374.4.16 SearchMode() [2/2]

```
NeighborSearchMode& SearchMode ( ) [inline]
```

Modify the search mode.

Definition at line 318 of file neighbor_search.hpp.

39.374.4.17 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the **NeighborSearch** (p. 1627) model.

Referenced by NeighborSearch< neighbor::NearestNeighborSort, metric::LMetric< TPower, true > >::Reference←Tree().

39.374.4.18 Train() [1/2]

```
void Train (
    MatType referenceSet )
```

Set the reference set to a new reference set, and build a tree if necessary.

The dataset is copied by default, but the copy can be avoided by transferring the ownership of the dataset using `std::move()`. This method is called '**Train()** (p. 1638)' in order to match the rest of the mlpack abstractions, even though calling this "training" is maybe a bit of a stretch.

Parameters

<i>referenceSet</i>	New set of reference data.
---------------------	----------------------------

Referenced by `CosineSearch::CosineSearch()`, and `PearsonSearch::PearsonSearch()`.

39.374.4.19 `Train()` [2/2]

```
void Train (
    Tree referenceTree )
```

Set the reference tree to a new reference tree.

The tree is copied by default, but the copy can be avoided by using `std::move()` to transfer the ownership of the tree. This method is called '**Train()**' (p. 1638) in order to match the rest of the `mlpack` abstractions, even though calling this "training" is maybe a bit of a stretch.

Parameters

<i>referenceTree</i>	Pre-built tree for reference points.
----------------------	--------------------------------------

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/neighbor_search.hpp`

39.375 `NeighborSearchRules< SortPolicy, MetricType, TreeType >` Class Template Reference

The **NeighborSearchRules** (p. 1640) class is a template helper class used by **NeighborSearch** (p. 1627) class when performing distance-based neighbor searches.

Classes

- struct **CandidateCmp**
Compare two candidates based on the distance.

Public Types

- typedef **tree::TraversallInfo< TreeType >** **TraversallInfoType**
Convenience typedef.

Public Member Functions

- **NeighborSearchRules** (const typename TreeType::Mat & **referenceSet**, const typename TreeType::Mat & **querySet**, const size_t **k**, MetricType & **metric**, const double **epsilon**=0, const bool **sameSet**=false)
*Construct the **NeighborSearchRules** (p. 1640) object.*
- double **BaseCase** (const size_t queryIndex, const size_t referenceIndex)
Get the distance from the query point to the reference point.
- size_t **BaseCases** () const
Get the number of base cases that have been performed.
- size_t & **BaseCases** ()
Modify the number of base cases that have been performed.
- size_t **GetBestChild** (const size_t queryIndex, TreeType &referenceNode)
Get the child node with the best score.
- size_t **GetBestChild** (const TreeType &queryNode, TreeType &referenceNode)
Get the child node with the best score.
- void **GetResults** (arma::Mat< size_t > &neighbors, arma::mat &distances)
Store the list of candidates for each query point in the given matrices.
- double **Rescore** (const size_t queryIndex, TreeType &referenceNode, const double oldScore) const
Re-evaluate the score for recursion order.
- double **Rescore** (TreeType &queryNode, TreeType &referenceNode, const double oldScore) const
Re-evaluate the score for recursion order.
- double **Score** (const size_t queryIndex, TreeType &referenceNode)
Get the score for recursion order.
- double **Score** (TreeType &queryNode, TreeType &referenceNode)
Get the score for recursion order.
- size_t **Scores** () const
Get the number of scores that have been performed.
- size_t & **Scores** ()
Modify the number of scores that have been performed.
- const **TraversallInfoType** & **TraversallInfo** () const
Get the traversal info.
- **TraversallInfoType** & **TraversallInfo** ()
Modify the traversal info.

Protected Types

- typedef std::pair< double, size_t > **Candidate**
Candidate represents a possible candidate neighbor (distance, index).
- typedef std::priority_queue< **Candidate**, std::vector< **Candidate** >, **CandidateCmp** > **CandidateList**
Use a priority queue to represent the list of candidate neighbors.

Protected Member Functions

- double **CalculateBound** (TreeType &queryNode) const
Recalculate the bound for a given query node.
- void **InsertNeighbor** (const size_t queryIndex, const size_t neighbor, const double distance)
Helper function to insert a point into the list of candidate points.

Protected Attributes

- `size_t` **baseCases**
The number of base cases that have been performed.
- `std::vector< CandidateList >` **candidates**
Set of candidate neighbors for each point.
- `const double` **epsilon**
Relative error to be considered in approximate search.
- `const size_t` **k**
Number of neighbors to search for.
- `double` **lastBaseCase**
The last base case result.
- `size_t` **lastQueryIndex**
*The last query point **BaseCase()** (p. 1644) was called with.*
- `size_t` **lastReferenceIndex**
*The last reference point **BaseCase()** (p. 1644) was called with.*
- `MetricType` & **metric**
The instantiated metric.
- `const TreeType::Mat` & **querySet**
The query set.
- `const TreeType::Mat` & **referenceSet**
The reference set.
- `bool` **sameSet**
Denotes whether or not the reference and query sets are the same.
- `size_t` **scores**
The number of scores that have been performed.
- `TraversalInfoType` **traversalInfo**
*Traversal info for the parent combination; this is updated by the traversal before each call to **Score()** (p. 1647).*

39.375.1 Detailed Description

```
template<typename SortPolicy, typename MetricType, typename TreeType>
class mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >
```

The **NeighborSearchRules** (p. 1640) class is a template helper class used by **NeighborSearch** (p. 1627) class when performing distance-based neighbor searches.

For each point in the query dataset, it keeps track of the k neighbors in the reference dataset which have the 'best' distance according to a given sorting policy.

Template Parameters

<i>SortPolicy</i>	The sort policy for distances.
<i>MetricType</i>	The metric to use for computation.
<i>TreeType</i>	The tree type to use; must adhere to the <code>TreeType</code> API.

Definition at line 35 of file neighbor_search_rules.hpp.

39.375.2 Member Typedef Documentation

39.375.2.1 Candidate

```
typedef std::pair<double, size_t> Candidate [protected]
```

Candidate represents a possible candidate neighbor (distance, index).

Definition at line 168 of file neighbor_search_rules.hpp.

39.375.2.2 CandidateList

```
typedef std::priority_queue< Candidate, std::vector< Candidate>, CandidateCmp> CandidateList [protected]
```

Use a priority queue to represent the list of candidate neighbors.

Definition at line 180 of file neighbor_search_rules.hpp.

39.375.2.3 TraversalInfoType

```
typedef tree::TraversalInfo<TreeType> TraversalInfoType
```

Convenience typedef.

Definition at line 153 of file neighbor_search_rules.hpp.

39.375.3 Constructor & Destructor Documentation

39.375.3.1 NeighborSearchRules()

```
NeighborSearchRules (  
    const typename TreeType::Mat & referenceSet,  
    const typename TreeType::Mat & querySet,  
    const size_t k,  
    MetricType & metric,  
    const double epsilon = 0,  
    const bool sameSet = false )
```

Construct the **NeighborSearchRules** (p. 1640) object.

This is usually done from within the **NeighborSearch** (p. 1627) class at search time.

Parameters

<i>referenceSet</i>	Set of reference data.
<i>querySet</i>	Set of query data.
<i>k</i>	Number of neighbors to search for.
<i>metric</i>	Instantiated metric.
<i>epsilon</i>	Relative approximate error.
<i>sameSet</i>	If true, the query and reference set are taken to be the same, and a query point will not return itself in the results.

39.375.4 Member Function Documentation

39.375.4.1 BaseCase()

```
double BaseCase (
    const size_t queryIndex,
    const size_t referenceIndex )
```

Get the distance from the query point to the reference point.

This will update the list of candidates with the new point if appropriate and will track the number of base cases (number of points evaluated).

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceIndex</i>	Index of reference point.

39.375.4.2 BaseCases() [1/2]

```
size_t BaseCases ( ) const [inline]
```

Get the number of base cases that have been performed.

Definition at line 143 of file neighbor_search_rules.hpp.

References NeighborSearchRules< SortPolicy, MetricType, TreeType >::baseCases.

39.375.4.3 BaseCases() [2/2]

```
size_t& BaseCases ( ) [inline]
```

Modify the number of base cases that have been performed.

Definition at line 145 of file neighbor_search_rules.hpp.

References NeighborSearchRules< SortPolicy, MetricType, TreeType >::baseCases.

39.375.4.4 CalculateBound()

```
double CalculateBound (
    TreeType & queryNode ) const [protected]
```

Recalculate the bound for a given query node.

39.375.4.5 GetBestChild() [1/2]

```
size_t GetBestChild (
    const size_t queryIndex,
    TreeType & referenceNode )
```

Get the child node with the best score.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.

39.375.4.6 GetBestChild() [2/2]

```
size_t GetBestChild (
    const TreeType & queryNode,
    TreeType & referenceNode )
```

Get the child node with the best score.

Parameters

<i>queryNode</i>	Node to be considered.
<i>referenceNode</i>	Candidate node to be recursed into.

39.375.4.7 GetResults()

```
void GetResults (
    arma::Mat< size_t > & neighbors,
    arma::mat & distances )
```

Store the list of candidates for each query point in the given matrices.

Parameters

<i>neighbors</i>	Matrix storing lists of neighbors for each query point.
<i>distances</i>	Matrix storing distances of neighbors for each query point.

39.375.4.8 InsertNeighbor()

```
void InsertNeighbor (
    const size_t queryIndex,
    const size_t neighbor,
    const double distance ) [protected]
```

Helper function to insert a point into the list of candidate points.

Parameters

<i>queryIndex</i>	Index of point whose neighbors we are inserting into.
<i>neighbor</i>	Index of reference point which is being inserted.
<i>distance</i>	Distance from query point to reference point.

39.375.4.9 Rescore() [1/2]

```
double Rescore (
    const size_t queryIndex,
```

```
TreeType & referenceNode,
const double oldScore ) const
```

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.
<i>oldScore</i>	Old score produced by Score() (p. 1647) (or Rescore() (p. 1646)).

39.375.4.10 Rescore() [2/2]

```
double Rescore (
    TreeType & queryNode,
    TreeType & referenceNode,
    const double oldScore ) const
```

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.
<i>oldScore</i>	Old score produced by Socre() (or Rescore() (p. 1646)).

39.375.4.11 Score() [1/2]

```
double Score (
    const size_t queryIndex,
    TreeType & referenceNode )
```

Get the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.

39.375.4.12 Score() [2/2]

```
double Score (
    TreeType & queryNode,
    TreeType & referenceNode )
```

Get the score for recursion order.

A low score indicates priority for recursionm while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.

39.375.4.13 Scores() [1/2]

```
size_t Scores ( ) const [inline]
```

Get the number of scores that have been performed.

Definition at line 148 of file neighbor_search_rules.hpp.

References NeighborSearchRules< SortPolicy, MetricType, TreeType >::scores.

39.375.4.14 Scores() [2/2]

```
size_t& Scores ( ) [inline]
```

Modify the number of scores that have been performed.

Definition at line 150 of file neighbor_search_rules.hpp.

References NeighborSearchRules< SortPolicy, MetricType, TreeType >::scores.

39.375.4.15 TraversalInfo() [1/2]

```
const TraversalInfoType& TraversalInfo ( ) const [inline]
```

Get the traversal info.

Definition at line 156 of file neighbor_search_rules.hpp.

References NeighborSearchRules< SortPolicy, MetricType, TreeType >::traversalInfo.

39.375.4.16 TraversalInfo() [2/2]

```
TraversalInfoType& TraversalInfo ( ) [inline]
```

Modify the traversal info.

Definition at line 158 of file neighbor_search_rules.hpp.

References NeighborSearchRules< SortPolicy, MetricType, TreeType >::traversalInfo.

39.375.5 Member Data Documentation

39.375.5.1 baseCases

```
size_t baseCases [protected]
```

The number of base cases that have been performed.

Definition at line 205 of file neighbor_search_rules.hpp.

Referenced by NeighborSearchRules< SortPolicy, MetricType, TreeType >::BaseCases().

39.375.5.2 candidates

```
std::vector< CandidateList> candidates [protected]
```

Set of candidate neighbors for each point.

Definition at line 183 of file neighbor_search_rules.hpp.

39.375.5.3 epsilon

```
const double epsilon [protected]
```

Relative error to be considered in approximate search.

Definition at line 195 of file neighbor_search_rules.hpp.

39.375.5.4 k

```
const size_t k [protected]
```

Number of neighbors to search for.

Definition at line 186 of file neighbor_search_rules.hpp.

39.375.5.5 lastBaseCase

```
double lastBaseCase [protected]
```

The last base case result.

Definition at line 202 of file neighbor_search_rules.hpp.

39.375.5.6 lastQueryIndex

```
size_t lastQueryIndex [protected]
```

The last query point **BaseCase()** (p. 1644) was called with.

Definition at line 198 of file neighbor_search_rules.hpp.

39.375.5.7 lastReferenceIndex

```
size_t lastReferenceIndex [protected]
```

The last reference point **BaseCase()** (p. 1644) was called with.

Definition at line 200 of file neighbor_search_rules.hpp.

39.375.5.8 metric

```
MetricType& metric [protected]
```

The instantiated metric.

Definition at line 189 of file neighbor_search_rules.hpp.

39.375.5.9 querySet

```
const TreeType::Mat& querySet [protected]
```

The query set.

Definition at line 165 of file neighbor_search_rules.hpp.

39.375.5.10 referenceSet

```
const TreeType::Mat& referenceSet [protected]
```

The reference set.

Definition at line 162 of file neighbor_search_rules.hpp.

39.375.5.11 sameSet

```
bool sameSet [protected]
```

Denotes whether or not the reference and query sets are the same.

Definition at line 192 of file neighbor_search_rules.hpp.

39.375.5.12 scores

```
size_t scores [protected]
```

The number of scores that have been performed.

Definition at line 207 of file neighbor_search_rules.hpp.

Referenced by NeighborSearchRules< SortPolicy, MetricType, TreeType >::Scores().

39.375.5.13 traversalInfo

TraversalInfoType traversalInfo [protected]

Traversal info for the parent combination; this is updated by the traversal before each call to **Score()** (p. 1647).

Definition at line 211 of file neighbor_search_rules.hpp.

Referenced by NeighborSearchRules< SortPolicy, MetricType, TreeType >::TraversalInfo().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ **neighbor_search_rules.hpp**

39.376 NeighborSearchRules< SortPolicy, MetricType, TreeType >::CandidateCmp Struct Reference

Compare two candidates based on the distance.

Public Member Functions

- bool **operator()** (const **Candidate** &c1, const **Candidate** &c2)

39.376.1 Detailed Description

```
template<typename SortPolicy, typename MetricType, typename TreeType>
struct mlpack::neighbor::NeighborSearchRules< SortPolicy, MetricType, TreeType >::CandidateCmp
```

Compare two candidates based on the distance.

Definition at line 171 of file neighbor_search_rules.hpp.

39.376.2 Member Function Documentation

39.376.2.1 operator()()

```
bool operator() (
    const Candidate & c1,
    const Candidate & c2 ) [inline]
```

Definition at line 172 of file neighbor_search_rules.hpp.

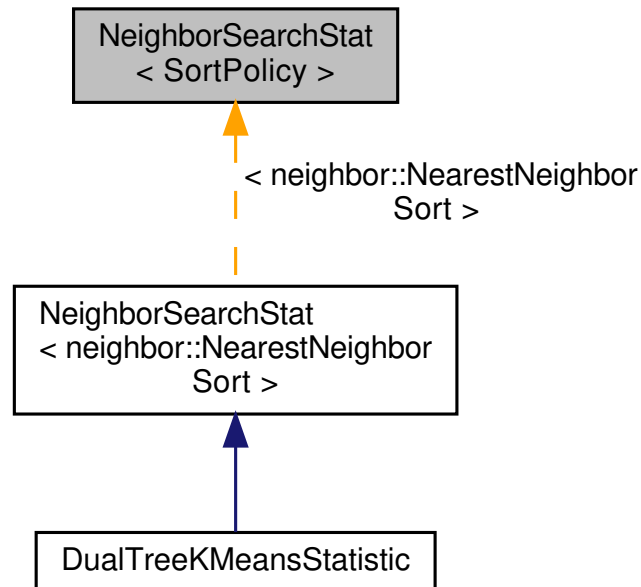
The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ **neighbor_search_rules.hpp**

39.377 NeighborSearchStat< SortPolicy > Class Template Reference

Extra data for each node in the tree.

Inheritance diagram for NeighborSearchStat< SortPolicy >:



Public Member Functions

- **NeighborSearchStat** ()
Initialize the statistic with the worst possible distance according to our sorting policy.
- template<typename TreeType >
NeighborSearchStat (TreeType &)
Initialization for a fully initialized node.
- double **AuxBound** () const
Get the aux bound.
- double & **AuxBound** ()
Modify the aux bound.
- double **FirstBound** () const
Get the first bound.
- double & **FirstBound** ()
Modify the first bound.
- double **LastDistance** () const
Get the last distance calculation.
- double & **LastDistance** ()
Modify the last distance calculation.
- void **Reset** ()
Reset statistic parameters to initial values.
- double **SecondBound** () const

Get the second bound.

- double & **SecondBound** ()

Modify the second bound.

- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)

Serialize the statistic to/from an archive.

39.377.1 Detailed Description

```
template<typename SortPolicy>
class mlpack::neighbor::NeighborSearchStat< SortPolicy >
```

Extra data for each node in the tree.

For neighbor searches, each node only needs to store a bound on neighbor distances.

Definition at line 26 of file neighbor_search_stat.hpp.

39.377.2 Constructor & Destructor Documentation

39.377.2.1 NeighborSearchStat() [1/2]

```
NeighborSearchStat ( ) [inline]
```

Initialize the statistic with the worst possible distance according to our sorting policy.

Definition at line 48 of file neighbor_search_stat.hpp.

39.377.2.2 NeighborSearchStat() [2/2]

```
NeighborSearchStat (
    TreeType & ) [inline]
```

Initialization for a fully initialized node.

In this case, we don't need to worry about the node.

Definition at line 59 of file neighbor_search_stat.hpp.

39.377.3 Member Function Documentation

39.377.3.1 AuxBound() [1/2]

```
double AuxBound ( ) const [inline]
```

Get the aux bound.

Definition at line 85 of file neighbor_search_stat.hpp.

39.377.3.2 AuxBound() [2/2]

```
double& AuxBound ( ) [inline]
```

Modify the aux bound.

Definition at line 87 of file neighbor_search_stat.hpp.

39.377.3.3 FirstBound() [1/2]

```
double FirstBound ( ) const [inline]
```

Get the first bound.

Definition at line 77 of file neighbor_search_stat.hpp.

39.377.3.4 FirstBound() [2/2]

```
double& FirstBound ( ) [inline]
```

Modify the first bound.

Definition at line 79 of file neighbor_search_stat.hpp.

39.377.3.5 LastDistance() [1/2]

```
double LastDistance ( ) const [inline]
```

Get the last distance calculation.

Definition at line 89 of file neighbor_search_stat.hpp.

39.377.3.6 LastDistance() [2/2]

```
double& LastDistance ( ) [inline]
```

Modify the last distance calculation.

Definition at line 91 of file neighbor_search_stat.hpp.

39.377.3.7 Reset()

```
void Reset ( ) [inline]
```

Reset statistic parameters to initial values.

Definition at line 68 of file neighbor_search_stat.hpp.

39.377.3.8 SecondBound() [1/2]

```
double SecondBound ( ) const [inline]
```

Get the second bound.

Definition at line 81 of file neighbor_search_stat.hpp.

39.377.3.9 SecondBound() [2/2]

```
double& SecondBound ( ) [inline]
```

Modify the second bound.

Definition at line 83 of file neighbor_search_stat.hpp.

39.377.3.10 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the statistic to/from an archive.

Definition at line 95 of file neighbor_search_stat.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ **neighbor_search_stat.hpp**

39.378 NSModel< SortPolicy > Class Template Reference

The **NSModel** (p. 1657) class provides an easy way to serialize a model, abstracts away the different types of trees, and also reflects the **NeighborSearch** (p. 1627) API.

Public Types

- enum **TreeTypes** {
 KD_TREE,
 COVER_TREE,
 R_TREE,
 R_STAR_TREE,
 BALL_TREE,
 X_TREE,
 HILBERT_R_TREE,
 R_PLUS_TREE,
 R_PLUS_PLUS_TREE,
 VP_TREE,
 RP_TREE,
 MAX_RP_TREE,
 SPILL_TREE,
 UB_TREE,
 OCTREE }

Enum type to identify each accepted tree type.

Public Member Functions

- **NSModel** (**TreeTypes** treeType=TreeTypes::KD_TREE, bool randomBasis=false)
*Initialize the **NSModel** (p. 1657) with the given type and whether or not a random basis should be used.*
- **NSModel** (const **NSModel** &other)
*Copy the given **NSModel** (p. 1657).*
- **NSModel** (**NSModel** &&other)
*Take ownership of the given **NSModel** (p. 1657).*
- **~NSModel** ()
Clean memory, if necessary.
- void **BuildModel** (arma::mat &&referenceSet, const size_t leafSize, const **NeighborSearchMode** searchMode, const double epsilon=0)
Build the reference tree.
- const arma::mat & **Dataset** () const
Expose the dataset.
- double **Epsilon** () const
Expose Epsilon.
- double & **Epsilon** ()
- size_t **LeafSize** () const
Expose leafSize.
- size_t & **LeafSize** ()
- **NSModel** & **operator=** (const **NSModel** &other)
*Copy the given **NSModel** (p. 1657).*
- **NSModel** & **operator=** (**NSModel** &&other)
*Take ownership of the given **NSModel** (p. 1657).*
- bool **RandomBasis** () const
Expose randomBasis.
- bool & **RandomBasis** ()
- double **Rho** () const
Expose rho.
- double & **Rho** ()
- void **Search** (arma::mat &&querySet, const size_t k, arma::Mat< size_t > &neighbors, arma::mat &distances)
Perform neighbor search. The query set will be reordered.
- void **Search** (const size_t k, arma::Mat< size_t > &neighbors, arma::mat &distances)
Perform monochromatic neighbor search.
- **NeighborSearchMode** **SearchMode** () const
Expose SearchMode.
- **NeighborSearchMode** & **SearchMode** ()
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the neighbor search model.
- double **Tau** () const
Expose tau.
- double & **Tau** ()
- std::string **TreeName** () const
Return a string representation of the current tree type.
- **TreeTypes** **TreeType** () const
Expose treeType.
- **TreeTypes** & **TreeType** ()

39.378.1 Detailed Description

```
template<typename SortPolicy>
class mlpack::neighbor::NSModel< SortPolicy >
```

The **NSModel** (p. 1657) class provides an easy way to serialize a model, abstracts away the different types of trees, and also reflects the **NeighborSearch** (p. 1627) API.

This class is meant to be used by the command-line `mlpack_knn` and `mlpack_kfn` programs, and thus does not have the same complete functionality and flexibility as the **NeighborSearch** (p. 1627) class. So if you are using it outside of `mlpack_knn` and `mlpack_kfn`, be aware that it is limited!

Template Parameters

<i>SortPolicy</i>	The sort policy for distances; see <code>NearestNeighborSort</code> .
-------------------	---

Definition at line 248 of file `ns_model.hpp`.

39.378.2 Member Enumeration Documentation

39.378.2.1 TreeTypes

```
enum TreeTypes
```

Enum type to identify each accepted tree type.

Enumerator

KD_TREE	
COVER_TREE	
R_TREE	
R_STAR_TREE	
BALL_TREE	
X_TREE	
HILBERT_R_TREE	
R_PLUS_TREE	
R_PLUS_PLUS_TREE	
VP_TREE	
RP_TREE	
MAX_RP_TREE	
SPILL_TREE	
UB_TREE	
OCTREE	

Definition at line 252 of file ns_model.hpp.

39.378.3 Constructor & Destructor Documentation

39.378.3.1 NSModel() [1/3]

```
NSModel (
    TreeTypes treeType = TreeTypes::KD_TREE,
    bool randomBasis = false )
```

Initialize the **NSModel** (p. 1657) with the given type and whether or not a random basis should be used.

Parameters

<i>treeType</i>	Type of tree to use.
<i>randomBasis</i>	Whether or not to project the points onto a random basis before searching.

39.378.3.2 NSModel() [2/3]

```
NSModel (
    const NSModel< SortPolicy > & other )
```

Copy the given **NSModel** (p. 1657).

Parameters

<i>other</i>	Model to copy.
--------------	----------------

39.378.3.3 NSModel() [3/3]

```
NSModel (
    NSModel< SortPolicy > && other )
```

Take ownership of the given **NSModel** (p. 1657).

Parameters

<i>other</i>	Model to take ownership of.
--------------	-----------------------------

39.378.3.4 ~NSModel()

`~ NSModel ()`

Clean memory, if necessary.

39.378.4 Member Function Documentation

39.378.4.1 BuildModel()

```
void BuildModel (
    arma::mat && referenceSet,
    const size_t leafSize,
    const NeighborSearchMode searchMode,
    const double epsilon = 0 )
```

Build the reference tree.

39.378.4.2 Dataset()

```
const arma::mat& Dataset ( ) const
```

Expose the dataset.

39.378.4.3 Epsilon() [1/2]

```
double Epsilon ( ) const
```

Expose Epsilon.

39.378.4.4 Epsilon() [2/2]

```
double& Epsilon ( )
```

39.378.4.5 LeafSize() [1/2]

```
size_t LeafSize ( ) const [inline]
```

Expose leafSize.

Definition at line 367 of file ns_model.hpp.

39.378.4.6 LeafSize() [2/2]

```
size_t& LeafSize ( ) [inline]
```

Definition at line 368 of file ns_model.hpp.

39.378.4.7 operator=() [1/2]

```
NSModel& operator= (
    const NSModel< SortPolicy > & other )
```

Copy the given **NSModel** (p. 1657).

Parameters

<i>other</i>	Model to copy.
--------------	----------------

39.378.4.8 operator=() [2/2]

```
NSModel& operator= (
    NSModel< SortPolicy > && other )
```

Take ownership of the given **NSModel** (p. 1657).

Parameters

<i>other</i>	Model to take ownership of.
--------------	-----------------------------

39.378.4.9 RandomBasis() [1/2]

```
bool RandomBasis ( ) const [inline]
```

Expose randomBasis.

Definition at line 383 of file ns_model.hpp.

39.378.4.10 RandomBasis() [2/2]

```
bool& RandomBasis ( ) [inline]
```

Definition at line 384 of file ns_model.hpp.

References BOOST_TEMPLATE_CLASS_VERSION().

39.378.4.11 Rho() [1/2]

```
double Rho ( ) const [inline]
```

Expose rho.

Definition at line 375 of file ns_model.hpp.

39.378.4.12 Rho() [2/2]

```
double& Rho ( ) [inline]
```

Definition at line 376 of file ns_model.hpp.

39.378.4.13 Search() [1/2]

```
void Search (
    arma::mat && querySet,
    const size_t k,
    arma::Mat< size_t > & neighbors,
    arma::mat & distances )
```

Perform neighbor search. The query set will be reordered.

39.378.4.14 Search() [2/2]

```
void Search (
    const size_t k,
    arma::Mat< size_t > & neighbors,
    arma::mat & distances )
```

Perform monochromatic neighbor search.

39.378.4.15 SearchMode() [1/2]

```
NeighborSearchMode SearchMode ( ) const
```

Expose SearchMode.

39.378.4.16 SearchMode() [2/2]

```
NeighborSearchMode& SearchMode ( )
```

39.378.4.17 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the neighbor search model.

39.378.4.18 `Tau()` [1/2]

```
double Tau ( ) const [inline]
```

Expose tau.

Definition at line 371 of file ns_model.hpp.

39.378.4.19 `Tau()` [2/2]

```
double& Tau ( ) [inline]
```

Definition at line 372 of file ns_model.hpp.

39.378.4.20 `TreeName()`

```
std::string TreeName ( ) const
```

Return a string representation of the current tree type.

39.378.4.21 `TreeType()` [1/2]

```
TreeTypes TreeType ( ) const [inline]
```

Expose treeType.

Definition at line 379 of file ns_model.hpp.

39.378.4.22 `TreeType()` [2/2]

```
TreeTypes& TreeType ( ) [inline]
```

Definition at line 380 of file ns_model.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ **ns_model.hpp**

39.379 QDAFN< MatType > Class Template Reference

Public Member Functions

- **QDAFN** (const size_t l, const size_t m)
*Construct the **QDAFN** (p. 1666) object but do not train it.*
- **QDAFN** (const MatType &referenceSet, const size_t l, const size_t m)
*Construct the **QDAFN** (p. 1666) object with the given reference set (this is the set that will be searched).*
- const MatType & **CandidateSet** (const size_t t) const
Get the candidate set for the given projection table.
- MatType & **CandidateSet** (const size_t t)
Modify the candidate set for the given projection table. Careful!
- size_t **NumProjections** () const
Get the number of projections.
- void **Search** (const MatType &querySet, const size_t k, arma::Mat< size_t > &neighbors, arma::mat &distances)
Search for the k furthest neighbors of the given query set.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the model.
- void **Train** (const MatType &referenceSet, const size_t l=0, const size_t m=0)
*Train the **QDAFN** (p. 1666) model on the given reference set, optionally setting new parameters for the number of projections/tables (l) and the number of elements stored for each projection/table (m).*

39.379.1 Detailed Description

```
template<typename MatType = arma::mat>
class mlpack::neighbor::QDAFN< MatType >
```

Definition at line 34 of file qdafn.hpp.

39.379.2 Constructor & Destructor Documentation

39.379.2.1 QDAFN() [1/2]

```
QDAFN (
    const size_t l,
    const size_t m )
```

Construct the **QDAFN** (p. 1666) object but do not train it.

Be sure to call **Train()** (p. 1668) before calling **Search()** (p. 1668).

Parameters

<i>l</i>	Number of projections.
<i>m</i>	Number of elements to store for each projection.

39.379.2.2 QDAFN() [2/2]

```
QDAFN (
    const MatType & referenceSet,
    const size_t l,
    const size_t m )
```

Construct the **QDAFN** (p. 1666) object with the given reference set (this is the set that will be searched).

Parameters

<i>referenceSet</i>	Set of reference data.
<i>l</i>	Number of projections.
<i>m</i>	Number of elements to store for each projection.

39.379.3 Member Function Documentation

39.379.3.1 CandidateSet() [1/2]

```
const MatType& CandidateSet (
    const size_t t ) const [inline]
```

Get the candidate set for the given projection table.

Definition at line 90 of file qdafn.hpp.

39.379.3.2 CandidateSet() [2/2]

```
MatType& CandidateSet (
    const size_t t ) [inline]
```

Modify the candidate set for the given projection table. Careful!

Definition at line 92 of file qdafn.hpp.

39.379.3.3 NumProjections()

```
size_t NumProjections ( ) const [inline]
```

Get the number of projections.

Definition at line 87 of file qdafn.hpp.

39.379.3.4 Search()

```
void Search (
    const MatType & querySet,
    const size_t k,
    arma::Mat< size_t > & neighbors,
    arma::mat & distances )
```

Search for the k furthest neighbors of the given query set.

(The query set can contain just one point, that is okay.) The results will be stored in the given neighbors and distances matrices, in the same format as the mlpack **NeighborSearch** (p. 1627) and **LSHSearch** (p. 1609) classes.

39.379.3.5 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the model.

39.379.3.6 Train()

```
void Train (
    const MatType & referenceSet,
    const size_t l = 0,
    const size_t m = 0 )
```

Train the **QDAFN** (p. 1666) model on the given reference set, optionally setting new parameters for the number of projections/tables (l) and the number of elements stored for each projection/table (m).

Parameters

<i>referenceSet</i>	Reference set to train on.
<i>l</i>	Number of projections.
<i>m</i>	Number of elements to store for each projection.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/approx_kfn/ **qdafn.hpp**

39.380 RAModel< SortPolicy > Class Template Reference

The **RAModel** (p. 1669) class provides an abstraction for the **RASearch** (p. 1681) class, abstracting away the `TreeType` parameter and allowing it to be specified at runtime in this class.

Public Types

- enum **TreeTypes** {
KD_TREE,
COVER_TREE,
R_TREE,
R_STAR_TREE,
X_TREE,
HILBERT_R_TREE,
R_PLUS_TREE,
R_PLUS_PLUS_TREE,
UB_TREE,
OCTREE }

*The list of tree types we can use with **RASearch** (p. 1681).*

Public Member Functions

- **RAModel** (**TreeTypes** treeType=`TreeTypes::KD_TREE`, bool randomBasis=false)
*Initialize the **RAModel** (p. 1669) with the given type and whether or not a random basis should be used.*
- **RAModel** (const **RAModel** &other)
*Copy the given **RAModel** (p. 1669).*
- **RAModel** (**RAModel** &&other)
*Take ownership of the given **RAModel** (p. 1669).*
- **~RAModel** ()
Clean memory, if necessary.
- double **Alpha** () const
Get the desired success probability.
- double & **Alpha** ()
Modify the desired success probability.
- void **BuildModel** (arma::mat &&referenceSet, const size_t leafSize, const bool naive, const bool singleMode)
Build the reference tree.
- const arma::mat & **Dataset** () const
Expose the dataset.
- bool **FirstLeafExact** () const
Get whether or not we traverse to the first leaf without approximation.
- bool & **FirstLeafExact** ()

- Modify whether or not we traverse to the first leaf without approximation.*

 - `size_t LeafSize () const`
Get the leaf size (only relevant when the kd-tree is used).
 - `size_t & LeafSize ()`
Modify the leaf size (only relevant when the kd-tree is used).
 - `bool Naive () const`
Get whether or not naive search is being used.
 - `bool & Naive ()`
Modify whether or not naive search is being used.
 - `RAModel & operator= (const RAModel &other)`
Copy the given RAModel (p. 1669).
 - `RAModel & operator= (RAModel &&other)`
Take ownership of the given RAModel (p. 1669).
 - `bool RandomBasis () const`
Get whether or not a random basis is being used.
 - `bool & RandomBasis ()`
Modify whether or not a random basis is being used.
 - `bool SampleAtLeaves () const`
Get whether or not sampling is done at the leaves.
 - `bool & SampleAtLeaves ()`
Modify whether or not sampling is done at the leaves.
 - `void Search (arma::mat &&querySet, const size_t k, arma::Mat< size_t > &neighbors, arma::mat &distances)`
Perform rank-approximate neighbor search, taking ownership of the query set.
 - `void Search (const size_t k, arma::Mat< size_t > &neighbors, arma::mat &distances)`
Perform rank-approximate neighbor search, using the reference set as the query set.
 - `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`
Serialize the model.
 - `bool SingleMode () const`
Get whether or not single-tree search is being used.
 - `bool & SingleMode ()`
Modify whether or not single-tree search is being used.
 - `size_t SingleSampleLimit () const`
Get the limit on the size of a node that can be approximated.
 - `size_t & SingleSampleLimit ()`
Modify the limit on the size of a node that can be approximation.
 - `double Tau () const`
Get the rank-approximation in percentile of the data.
 - `double & Tau ()`
Modify the rank-approximation in percentile of the data.
 - `std::string TreeName () const`
Get the name of the tree type.
 - `TreeTypes TreeType () const`
Get the type of tree being used.
 - `TreeTypes & TreeType ()`
Modify the type of tree being used.

39.380.1 Detailed Description

```
template<typename SortPolicy>
class mlpack::neighbor::RAModel< SortPolicy >
```

The **RAModel** (p. 1669) class provides an abstraction for the **RASearch** (p. 1681) class, abstracting away the `TreeType` parameter and allowing it to be specified at runtime in this class.

This class is written for the sake of the 'allkrann' program, but is not necessarily restricted to that use.

Parameters

<i>SortPolicy</i>	Sorting policy for neighbor searching (see RASearch (p. 1681)).
-------------------	--

Definition at line 271 of file `ra_model.hpp`.

39.380.2 Member Enumeration Documentation

39.380.2.1 TreeTypes

```
enum TreeTypes
```

The list of tree types we can use with **RASearch** (p. 1681).

Does not include ball trees; see #338.

Enumerator

KD_TREE	
COVER_TREE	
R_TREE	
R_STAR_TREE	
X_TREE	
HILBERT_R_TREE	
R_PLUS_TREE	
R_PLUS_PLUS_TREE	
UB_TREE	
OCTREE	

Definition at line 278 of file `ra_model.hpp`.

39.380.3 Constructor & Destructor Documentation

39.380.3.1 **RAModel**() [1/3]

```
RAModel (
    TreeTypes treeType = TreeTypes::KD_TREE,
    bool randomBasis = false )
```

Initialize the **RAModel** (p. 1669) with the given type and whether or not a random basis should be used.

39.380.3.2 **RAModel**() [2/3]

```
RAModel (
    const RAModel< SortPolicy > & other )
```

Copy the given **RAModel** (p. 1669).

Parameters

<i>other</i>	RAModel (p. 1669) to copy.
--------------	-----------------------------------

39.380.3.3 **RAModel**() [3/3]

```
RAModel (
    RAModel< SortPolicy > && other )
```

Take ownership of the given **RAModel** (p. 1669).

Parameters

<i>other</i>	RAModel (p. 1669) to take ownership of.
--------------	--

39.380.3.4 **~RAModel**()

```
~RAModel ( )
```

Clean memory, if necessary.

39.380.4 Member Function Documentation

39.380.4.1 Alpha() [1/2]

```
double Alpha ( ) const
```

Get the desired success probability.

39.380.4.2 Alpha() [2/2]

```
double& Alpha ( )
```

Modify the desired success probability.

39.380.4.3 BuildModel()

```
void BuildModel (
    arma::mat && referenceSet,
    const size_t leafSize,
    const bool naive,
    const bool singleMode )
```

Build the reference tree.

39.380.4.4 Dataset()

```
const arma::mat& Dataset ( ) const
```

Expose the dataset.

39.380.4.5 FirstLeafExact() [1/2]

```
bool FirstLeafExact ( ) const
```

Get whether or not we traverse to the first leaf without approximation.

39.380.4.6 FirstLeafExact() [2/2]

```
bool& FirstLeafExact ( )
```

Modify whether or not we traverse to the first leaf without approximation.

39.380.4.7 LeafSize() [1/2]

```
size_t LeafSize ( ) const
```

Get the leaf size (only relevant when the kd-tree is used).

39.380.4.8 LeafSize() [2/2]

```
size_t& LeafSize ( )
```

Modify the leaf size (only relevant when the kd-tree is used).

39.380.4.9 Naive() [1/2]

```
bool Naive ( ) const
```

Get whether or not naive search is being used.

39.380.4.10 Naive() [2/2]

```
bool& Naive ( )
```

Modify whether or not naive search is being used.

39.380.4.11 operator=() [1/2]

```
RAModel& operator= (
    const RAModel< SortPolicy > & other )
```

Copy the given **RAModel** (p. 1669).

Parameters

<i>other</i>	RAModel (p. 1669) to copy.
--------------	-----------------------------------

39.380.4.12 operator=() [2/2]

```
RAModel& operator= (
    RAModel< SortPolicy > && other )
```

Take ownership of the given **RAModel** (p. 1669).

Parameters

<i>other</i>	RAModel (p. 1669) to take ownership of.
--------------	--

39.380.4.13 RandomBasis() [1/2]

```
bool RandomBasis ( ) const
```

Get whether or not a random basis is being used.

39.380.4.14 RandomBasis() [2/2]

```
bool& RandomBasis ( )
```

Modify whether or not a random basis is being used.

Be sure to rebuild the model using **BuildModel()** (p. 1673).

39.380.4.15 SampleAtLeaves() [1/2]

```
bool SampleAtLeaves ( ) const
```

Get whether or not sampling is done at the leaves.

39.380.4.16 SampleAtLeaves() [2/2]

```
bool& SampleAtLeaves ( )
```

Modify whether or not sampling is done at the leaves.

39.380.4.17 Search() [1/2]

```
void Search (
    arma::mat && querySet,
    const size_t k,
    arma::Mat< size_t > & neighbors,
    arma::mat & distances )
```

Perform rank-approximate neighbor search, taking ownership of the query set.

39.380.4.18 Search() [2/2]

```
void Search (
    const size_t k,
    arma::Mat< size_t > & neighbors,
    arma::mat & distances )
```

Perform rank-approximate neighbor search, using the reference set as the query set.

39.380.4.19 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the model.

39.380.4.20 SingleMode() [1/2]

```
bool SingleMode ( ) const
```

Get whether or not single-tree search is being used.

39.380.4.21 SingleMode() [2/2]

```
bool& SingleMode ( )
```

Modify whether or not single-tree search is being used.

39.380.4.22 SingleSampleLimit() [1/2]

```
size_t SingleSampleLimit ( ) const
```

Get the limit on the size of a node that can be approximated.

39.380.4.23 SingleSampleLimit() [2/2]

```
size_t& SingleSampleLimit ( )
```

Modify the limit on the size of a node that can be approximation.

39.380.4.24 Tau() [1/2]

```
double Tau ( ) const
```

Get the rank-approximation in percentile of the data.

39.380.4.25 Tau() [2/2]

```
double& Tau ( )
```

Modify the rank-approximation in percentile of the data.

39.380.4.26 TreeName()

```
std::string TreeName ( ) const
```

Get the name of the tree type.

39.380.4.27 **TreeType()** [1/2]

```
TreeTypes TreeType ( ) const
```

Get the type of tree being used.

39.380.4.28 **TreeType()** [2/2]

```
TreeTypes& TreeType ( )
```

Modify the type of tree being used.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ **ra_model.hpp**

39.381 **RAQueryStat< SortPolicy >** Class Template Reference

Extra data for each node in the tree.

Public Member Functions

- **RAQueryStat** ()
Initialize the statistic with the worst possible distance according to our sorting policy.
- template<typename TreeType >
RAQueryStat (const TreeType &)
Initialization for a node.
- double **Bound** () const
Get the bound.
- double & **Bound** ()
Modify the bound.
- size_t **NumSamplesMade** () const
Get the number of samples made.
- size_t & **NumSamplesMade** ()
Modify the number of samples made.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the statistic.

39.381.1 Detailed Description

```
template<typename SortPolicy>
class mlpack::neighbor::RAQueryStat< SortPolicy >
```

Extra data for each node in the tree.

For neighbor searches, each node only needs to store a bound on neighbor distances.

Every query is required to make a minimum number of samples to guarantee the desired approximation error. The 'numSamplesMade' keeps track of the minimum number of samples made by all queries in the node in question.

Definition at line 35 of file ra_query_stat.hpp.

39.381.2 Constructor & Destructor Documentation

39.381.2.1 RAQueryStat() [1/2]

```
RAQueryStat ( ) [inline]
```

Initialize the statistic with the worst possible distance according to our sorting policy.

Definition at line 42 of file ra_query_stat.hpp.

39.381.2.2 RAQueryStat() [2/2]

```
RAQueryStat (
    const TreeType & ) [inline]
```

Initialization for a node.

Definition at line 48 of file ra_query_stat.hpp.

39.381.3 Member Function Documentation

39.381.3.1 Bound() [1/2]

```
double Bound ( ) const [inline]
```

Get the bound.

Definition at line 54 of file `ra_query_stat.hpp`.

39.381.3.2 Bound() [2/2]

```
double& Bound ( ) [inline]
```

Modify the bound.

Definition at line 56 of file `ra_query_stat.hpp`.

39.381.3.3 NumSamplesMade() [1/2]

```
size_t NumSamplesMade ( ) const [inline]
```

Get the number of samples made.

Definition at line 59 of file `ra_query_stat.hpp`.

39.381.3.4 NumSamplesMade() [2/2]

```
size_t& NumSamplesMade ( ) [inline]
```

Modify the number of samples made.

Definition at line 61 of file `ra_query_stat.hpp`.

39.381.3.5 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the statistic.

Definition at line 65 of file `ra_query_stat.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ra_query_stat.hpp`

39.382 RASearch< SortPolicy, MetricType, MatType, TreeType > Class Template Reference

The **RASearch** (p. 1681) class: This class provides a generic manner to perform rank-approximate search via random-sampling.

Public Types

- typedef TreeType< MetricType, **RAQueryStat**< SortPolicy >, MatType > **Tree**
Convenience typedef.

Public Member Functions

- **RASearch** (MatType referenceSet, const bool naive=false, const bool singleMode=false, const double tau=5, const double alpha=0.95, const bool sampleAtLeaves=false, const bool firstLeafExact=false, const size_t singleSampleLimit=20, const MetricType metric=MetricType())
*Initialize the **RASearch** (p. 1681) object, passing both a reference dataset (this is the dataset that will be searched).*
- **RASearch** (**Tree** *referenceTree, const bool singleMode=false, const double tau=5, const double alpha=0.95, const bool sampleAtLeaves=false, const bool firstLeafExact=false, const size_t singleSampleLimit=20, const MetricType metric=MetricType())
*Initialize the **RASearch** (p. 1681) object with the given pre-constructed reference tree.*
- **RASearch** (const bool naive=false, const bool singleMode=false, const double tau=5, const double alpha=0.95, const bool sampleAtLeaves=false, const bool firstLeafExact=false, const size_t singleSampleLimit=20, const MetricType metric=MetricType())
*Create an **RASearch** (p. 1681) object with no reference data.*
- **~RASearch** ()
*Delete the **RASearch** (p. 1681) object.*
- double **Alpha** () const
Get the desired success probability.
- double & **Alpha** ()
Modify the desired success probability.
- bool **FirstLeafExact** () const
Get whether or not we traverse to the first leaf without approximation.
- bool & **FirstLeafExact** ()
Modify whether or not we traverse to the first leaf without approximation.
- bool **Naive** () const
Get whether or not naive (brute-force) search is used.
- bool & **Naive** ()
Modify whether or not naive (brute-force) search is used.
- const MatType & **ReferenceSet** () const
Access the reference set.
- void **ResetQueryTree** (**Tree** *queryTree) const
*This function recursively resets the **RAQueryStat** (p. 1678) of the given query tree to set 'bound' to SortPolicy::WorstDistance and 'numSamplesMade' to 0.*
- bool **SampleAtLeaves** () const
Get whether or not sampling is done at the leaves.
- bool & **SampleAtLeaves** ()

- Modify whether or not sampling is done at the leaves.*
- void **Search** (const MatType &querySet, const size_t k, arma::Mat< size_t > &neighbors, arma::mat &distances)
Compute the rank approximate nearest neighbors of each query point in the query set and store the output in the given matrices.
- void **Search** (**Tree** *queryTree, const size_t k, arma::Mat< size_t > &neighbors, arma::mat &distances)
Compute the rank approximate nearest neighbors of each point in the pre-built query tree and store the output in the given matrices.
- void **Search** (const size_t k, arma::Mat< size_t > &neighbors, arma::mat &distances)
Compute the rank approximate nearest neighbors of each point in the reference set (that is, the query set is taken to be the reference set), and store the output in the given matrices.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the object.
- bool **SingleMode** () const
Get whether or not single-tree search is used.
- bool & **SingleMode** ()
Modify whether or not single-tree search is used.
- size_t **SingleSampleLimit** () const
Get the limit on the size of a node that can be approximated.
- size_t & **SingleSampleLimit** ()
Modify the limit on the size of a node that can be approximation.
- double **Tau** () const
Get the rank-approximation in percentile of the data.
- double & **Tau** ()
Modify the rank-approximation in percentile of the data.
- void **Train** (MatType referenceSet)
"Train" the model on the given reference set.
- void **Train** (**Tree** *referenceTree)
Set the reference tree to a new reference tree.

39.382.1 Detailed Description

```
template<typename SortPolicy = NearestNeighborSort, typename MetricType = metric::EuclideanDistance, typename MatType =
arma::mat, template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType = tree::KDTree>
class mlpack::neighbor::RASearch< SortPolicy, MetricType, MatType, TreeType >
```

The **RASearch** (p. 1681) class: This class provides a generic manner to perform rank-approximate search via random-sampling.

If the 'naive' option is chosen, this rank-approximate search will be done by randomly sampling from the whole set. If the 'naive' option is not chosen, the sampling is done in a stratified manner in the tree as mentioned in the algorithms in Figure 2 of the following paper:

```
{ram2009rank, title={{Rank-Approximate Nearest Neighbor Search: Retaining Meaning and Speed in High Dimen-
sions}}, author={{Ram, P. and Lee, D. and Ouyang, H. and Gray, A. G.}}, booktitle={{Advances of Neural Information
Processing Systems}}, year={2009} }
```

RASearch (p. 1681) is currently known to not work with ball trees (#356).

Template Parameters

<i>SortPolicy</i>	The sort policy for distances; see NearestNeighborSort.
<i>MetricType</i>	The metric to use for computation.
<i>TreeType</i>	The tree type to use.

Definition at line 71 of file ra_search.hpp.

39.382.2 Member Typedef Documentation

39.382.2.1 Tree

```
typedef TreeType<MetricType, RAQueryStat<SortPolicy>, MatType> Tree
```

Convenience typedef.

Definition at line 75 of file ra_search.hpp.

39.382.3 Constructor & Destructor Documentation

39.382.3.1 RASearch() [1/3]

```
RASearch (
    MatType referenceSet,
    const bool naive = false,
    const bool singleMode = false,
    const double tau = 5,
    const double alpha = 0.95,
    const bool sampleAtLeaves = false,
    const bool firstLeafExact = false,
    const size_t singleSampleLimit = 20,
    const MetricType metric = MetricType() )
```

Initialize the **RASearch** (p. 1681) object, passing both a reference dataset (this is the dataset that will be searched).

Optionally, perform the computation in naive mode or single-tree mode. An initialized distance metric can be given, for cases where the metric has internal data (i.e. the distance::MahalanobisDistance class).

This method will copy the matrices to internal copies, which are rearranged during tree-building. If you don't need to keep the reference dataset, you can use std::move() to remove the overhead of making copies. Using std::move() transfers the ownership of the dataset.

tau, the rank-approximation parameter, specifies that we are looking for k neighbors with probability alpha of being in the top tau percent of nearest neighbors. So, as an example, if our dataset has 1000 points, and we want 5 nearest neighbors with 95% probability of being in the top 5% of nearest neighbors (or, the top 50 nearest neighbors), we set k = 5, tau = 5, and alpha = 0.95.

The method will fail (and throw a std::invalid_argument exception) if the value of tau is too low: tau must be set such that the number of points in the corresponding percentile of the data is greater than k. Thus, if we choose tau = 0.1 with a dataset of 1000 points and k = 5, then we are attempting to choose 5 nearest neighbors out of the closest 1 point – this is invalid.

Parameters

<i>referenceSet</i>	Set of reference points.
<i>naive</i>	If true, the rank-approximate search will be performed by directly sampling the whole set instead of using the stratified sampling on the tree.
<i>singleMode</i>	If true, single-tree search will be used (as opposed to dual-tree search). This is useful when Search() (p. 1688) will be called with few query points.
<i>metric</i>	An optional instance of the MetricType class.
<i>tau</i>	The rank-approximation in percentile of the data. The default value is 5%.
<i>alpha</i>	The desired success probability. The default value is 0.95.
<i>sampleAtLeaves</i>	Sample at leaves for faster but less accurate computation. This defaults to 'false'.
<i>firstLeafExact</i>	Traverse to the first leaf without approximation. This can ensure that the query definitely finds its (near) duplicate if there exists one. This defaults to 'false' for now.
<i>singleSampleLimit</i>	The limit on the largest node that can be approximated by sampling. This defaults to 20.

39.382.3.2 **RASearch()** [2/3]

```

RASearch (
    Tree * referenceTree,
    const bool singleMode = false,
    const double tau = 5,
    const double alpha = 0.95,
    const bool sampleAtLeaves = false,
    const bool firstLeafExact = false,
    const size_t singleSampleLimit = 20,
    const MetricType metric = MetricType() )

```

Initialize the **RASearch** (p. 1681) object with the given pre-constructed reference tree.

It is assumed that the points in the tree's dataset correspond to the reference set. Optionally, choose to use single-tree mode. Naive mode is not available as an option for this constructor; instead, to run naive computation, use a different constructor. Additionally, an instantiated distance metric can be given, for cases where the distance metric holds data.

There is no copying of the data matrices in this constructor (because tree-building is not necessary), so this is the constructor to use when copies absolutely must be avoided.

tau, the rank-approximation parameter, specifies that we are looking for k neighbors with probability alpha of being in the top tau percent of nearest neighbors. So, as an example, if our dataset has 1000 points, and we want 5 nearest neighbors with 95% probability of being in the top 5% of nearest neighbors (or, the top 50 nearest neighbors), we set k = 5, tau = 5, and alpha = 0.95.

The method will fail (and throw a `std::invalid_argument` exception) if the value of tau is too low: tau must be set such that the number of points in the corresponding percentile of the data is greater than k. Thus, if we choose tau = 0.1 with a dataset of 1000 points and k = 5, then we are attempting to choose 5 nearest neighbors out of the closest 1 point – this is invalid.

Note

Tree-building may (at least with `BinarySpaceTree`) modify the ordering of a matrix, so be aware that the results you get from **Search()** (p. 1688) will correspond to the modified matrix.

Parameters

<i>referenceTree</i>	Pre-built tree for reference points.
<i>singleMode</i>	Whether single-tree computation should be used (as opposed to dual-tree computation).
<i>tau</i>	The rank-approximation in percentile of the data. The default value is 5%.
<i>alpha</i>	The desired success probability. The default value is 0.95.
<i>sampleAtLeaves</i>	Sample at leaves for faster but less accurate computation. This defaults to 'false'.
<i>firstLeafExact</i>	Traverse to the first leaf without approximation. This can ensure that the query definitely finds its (near) duplicate if there exists one. This defaults to 'false' for now.
<i>singleSampleLimit</i>	The limit on the largest node that can be approximated by sampling. This defaults to 20.
<i>metric</i>	Instantiated distance metric.

39.382.3.3 RASearch() [3/3]

```

RASearch (
    const bool naive = false,
    const bool singleMode = false,
    const double tau = 5,
    const double alpha = 0.95,
    const bool sampleAtLeaves = false,
    const bool firstLeafExact = false,
    const size_t singleSampleLimit = 20,
    const MetricType metric = MetricType() )

```

Create an **RASearch** (p. 1681) object with no reference data.

If **Search()** (p. 1688) is called before a reference set is set with **Train()** (p. 1691), an exception will be thrown.

Parameters

<i>naive</i>	Whether naive (brute-force) search should be used.
<i>singleMode</i>	Whether single-tree computation should be used (as opposed to dual-tree computation).
<i>tau</i>	The rank-approximation in percentile of the data. The default value is 5%.
<i>alpha</i>	The desired success probability. The default value is 0.95.
<i>sampleAtLeaves</i>	Sample at leaves for faster but less accurate computation. This defaults to 'false'.
<i>firstLeafExact</i>	Traverse to the first leaf without approximation. This can ensure that the query definitely finds its (near) duplicate if there exists one. This defaults to 'false' for now.
<i>singleSampleLimit</i>	The limit on the largest node that can be approximated by sampling. This defaults to 20.
<i>metric</i>	Instantiated distance metric.

39.382.3.4 ~RASearch()

```

~ RASearch ( )

```

Delete the **RASearch** (p. 1681) object.

The tree is the only member we are responsible for deleting. The others will take care of themselves.

39.382.4 Member Function Documentation

39.382.4.1 Alpha() [1/2]

```
double Alpha ( ) const [inline]
```

Get the desired success probability.

Definition at line 338 of file ra_search.hpp.

39.382.4.2 Alpha() [2/2]

```
double& Alpha ( ) [inline]
```

Modify the desired success probability.

Definition at line 340 of file ra_search.hpp.

39.382.4.3 FirstLeafExact() [1/2]

```
bool FirstLeafExact ( ) const [inline]
```

Get whether or not we traverse to the first leaf without approximation.

Definition at line 348 of file ra_search.hpp.

39.382.4.4 FirstLeafExact() [2/2]

```
bool& FirstLeafExact ( ) [inline]
```

Modify whether or not we traverse to the first leaf without approximation.

Definition at line 350 of file ra_search.hpp.

39.382.4.5 Naive() [1/2]

```
bool Naive ( ) const [inline]
```

Get whether or not naive (brute-force) search is used.

Definition at line 323 of file ra_search.hpp.

39.382.4.6 Naive() [2/2]

```
bool& Naive ( ) [inline]
```

Modify whether or not naive (brute-force) search is used.

Definition at line 325 of file ra_search.hpp.

39.382.4.7 ReferenceSet()

```
const MatType& ReferenceSet ( ) const [inline]
```

Access the reference set.

Definition at line 320 of file ra_search.hpp.

39.382.4.8 ResetQueryTree()

```
void ResetQueryTree (
    Tree * queryTree ) const
```

This function recursively resets the **RAQueryStat** (p. 1678) of the given query tree to set 'bound' to SortPolicy::Worst↔ Distance and 'numSamplesMade' to 0.

This allows a user to perform multiple searches with the same query tree, possibly with different levels of approximation without requiring to build a new pair of trees for every new (approximate) search.

If **Search()** (p. 1688) is called multiple times with the same query tree without calling **ResetQueryTree()** (p. 1687), the results may not satisfy the theoretical guarantees provided by the rank-approximate neighbor search algorithm.

Parameters

<i>queryTree</i>	Tree whose statistics should be reset.
------------------	--

39.382.4.9 SampleAtLeaves() [1/2]

```
bool SampleAtLeaves ( ) const [inline]
```

Get whether or not sampling is done at the leaves.

Definition at line 343 of file `ra_search.hpp`.

39.382.4.10 SampleAtLeaves() [2/2]

```
bool& SampleAtLeaves ( ) [inline]
```

Modify whether or not sampling is done at the leaves.

Definition at line 345 of file `ra_search.hpp`.

39.382.4.11 Search() [1/3]

```
void Search (
    const MatType & querySet,
    const size_t k,
    arma::Mat< size_t > & neighbors,
    arma::mat & distances )
```

Compute the rank approximate nearest neighbors of each query point in the query set and store the output in the given matrices.

The matrices will be set to the size of n columns by k rows, where n is the number of points in the query dataset and k is the number of neighbors being searched for.

If `querySet` is small or only contains one point, it can be faster to do single-tree search; single-tree search can be set with the **SingleMode()** (p. 1690) function or in the constructor.

Parameters

<i>querySet</i>	Set of query points (can be a single point).
<i>k</i>	Number of neighbors to search for.
<i>neighbors</i>	Matrix storing lists of neighbors for each query point.
<i>distances</i>	Matrix storing distances of neighbors for each query point.

39.382.4.12 Search() [2/3]

```
void Search (
    Tree * queryTree,
    const size_t k,
    arma::Mat< size_t > & neighbors,
    arma::mat & distances )
```

Compute the rank approximate nearest neighbors of each point in the pre-built query tree and store the output in the given matrices.

The matrices will be set to the size of n columns by k rows, where n is the number of points in the query dataset and k is the number of neighbors being searched for.

If singleMode or naive is enabled, then this method will throw a `std::invalid_argument` exception; calling this function implies a dual-tree algorithm.

Note

If the tree type you are using modifies the data matrix, be aware that the results returned from this function will be with respect to the modified data matrix.

Parameters

<i>queryTree</i>	Tree built on query points.
<i>k</i>	Number of neighbors to search for.
<i>neighbors</i>	Matrix storing lists of neighbors for each query point.
<i>distances</i>	Matrix storing distances of neighbors for each query point.

39.382.4.13 Search() [3/3]

```
void Search (
    const size_t k,
    arma::Mat< size_t > & neighbors,
    arma::mat & distances )
```

Compute the rank approximate nearest neighbors of each point in the reference set (that is, the query set is taken to be the reference set), and store the output in the given matrices.

The matrices will be set to the size of n columns by k rows, where n is the number of points in the query dataset and k is the number of neighbors being searched for.

Parameters

<i>k</i>	Number of neighbors to search for.
<i>neighbors</i>	Matrix storing lists of neighbors for each point.
<i>distances</i>	Matrix storing distances of neighbors for each query point.

39.382.4.14 `serialize()`

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the object.

Referenced by `RASearch< SortPolicy, MetricType, MatType, TreeType >::SingleSampleLimit()`.

39.382.4.15 `SingleMode()` [1/2]

```
bool SingleMode ( ) const [inline]
```

Get whether or not single-tree search is used.

Definition at line 328 of file `ra_search.hpp`.

39.382.4.16 `SingleMode()` [2/2]

```
bool& SingleMode ( ) [inline]
```

Modify whether or not single-tree search is used.

Definition at line 330 of file `ra_search.hpp`.

39.382.4.17 `SingleSampleLimit()` [1/2]

```
size_t SingleSampleLimit ( ) const [inline]
```

Get the limit on the size of a node that can be approximated.

Definition at line 353 of file `ra_search.hpp`.

39.382.4.18 SingleSampleLimit() [2/2]

```
size_t& SingleSampleLimit ( ) [inline]
```

Modify the limit on the size of a node that can be approximation.

Definition at line 355 of file ra_search.hpp.

References RASearch< SortPolicy, MetricType, MatType, TreeType >::serialize().

39.382.4.19 Tau() [1/2]

```
double Tau ( ) const [inline]
```

Get the rank-approximation in percentile of the data.

Definition at line 333 of file ra_search.hpp.

39.382.4.20 Tau() [2/2]

```
double& Tau ( ) [inline]
```

Modify the rank-approximation in percentile of the data.

Definition at line 335 of file ra_search.hpp.

39.382.4.21 Train() [1/2]

```
void Train (
    MatType referenceSet )
```

"Train" the model on the given reference set.

If tree-based search is being used (if **Naive()** (p. 1687) is false), the reference tree is rebuilt. Thus, a copy of the reference dataset is made. If you don't need to keep the dataset, you can avoid copying by using `std::move()`. This transfers the ownership of the dataset.

Parameters

<i>referenceSet</i>	New reference set to use.
---------------------	---------------------------

39.382.4.22 Train() [2/2]

```
void Train (
    Tree * referenceTree )
```

Set the reference tree to a new reference tree.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ **ra_search.hpp**

39.383 RASearchRules< SortPolicy, MetricType, TreeType > Class Template Reference

The **RASearchRules** (p. 1692) class is a template helper class used by **RASearch** (p. 1681) class when performing rank-approximate search via random-sampling.

Public Types

- typedef **tree::TraversallInfo**< TreeType > **TraversallInfoType**

Public Member Functions

- **RASearchRules** (const arma::mat &referenceSet, const arma::mat &querySet, const size_t k, MetricType &metric, const double tau=5, const double **alpha**=0.95, const bool naive=false, const bool sampleAtLeaves=false, const bool firstLeafExact=false, const size_t singleSampleLimit=20, const bool sameSet=false)
*Construct the **RASearchRules** (p. 1692) object.*
- double **BaseCase** (const size_t queryIndex, const size_t referenceIndex)
Get the distance from the query point to the reference point.
- void **GetResults** (arma::Mat< size_t > &neighbors, arma::mat &distances)
Store the list of candidates for each query point in the given matrices.
- size_t **NumDistComputations** ()
- size_t **NumEffectiveSamples** ()
- double **Rescore** (const size_t queryIndex, TreeType &referenceNode, const double oldScore)
Re-evaluate the score for recursion order.
- double **Rescore** (TreeType &queryNode, TreeType &referenceNode, const double oldScore)
Re-evaluate the score for recursion order.
- double **Score** (const size_t queryIndex, TreeType &referenceNode)
Get the score for recursion order.
- double **Score** (const size_t queryIndex, TreeType &referenceNode, const double baseCaseResult)
Get the score for recursion order.
- double **Score** (TreeType &queryNode, TreeType &referenceNode)
Get the score for recursion order.
- double **Score** (TreeType &queryNode, TreeType &referenceNode, const double baseCaseResult)
Get the score for recursion order, passing the base case result (in the situation where it may be needed to calculate the recursion order).
- const **TraversallInfoType** & **TraversallInfo** () const
- **TraversallInfoType** & **TraversallInfo** ()

39.383.1 Detailed Description

```
template<typename SortPolicy, typename MetricType, typename TreeType>
class mlpack::neighbor::RASearchRules< SortPolicy, MetricType, TreeType >
```

The **RASearchRules** (p. 1692) class is a template helper class used by **RASearch** (p. 1681) class when performing rank-approximate search via random-sampling.

Template Parameters

<i>SortPolicy</i>	The sort policy for distances.
<i>MetricType</i>	The metric to use for computation.
<i>TreeType</i>	The tree type to use; must adhere to the TreeType API.

Definition at line 33 of file ra_search_rules.hpp.

39.383.2 Member Typedef Documentation

39.383.2.1 TraversalInfoType

```
typedef tree::TraversalInfo<TreeType> TraversalInfoType
```

Definition at line 239 of file ra_search_rules.hpp.

39.383.3 Constructor & Destructor Documentation

39.383.3.1 RASearchRules()

```
RASearchRules (
    const arma::mat & referenceSet,
    const arma::mat & querySet,
    const size_t k,
    MetricType & metric,
    const double tau = 5,
    const double alpha = 0.95,
    const bool naive = false,
    const bool sampleAtLeaves = false,
    const bool firstLeafExact = false,
    const size_t singleSampleLimit = 20,
    const bool sameSet = false )
```

Construct the **RASearchRules** (p. 1692) object.

This is usually done from within the **RASearch** (p. 1681) class at search time.

Parameters

<i>referenceSet</i>	Set of reference data.
<i>querySet</i>	Set of query data.
<i>k</i>	Number of neighbors to search for.
<i>metric</i>	Instantiated metric.
<i>tau</i>	The rank-approximation in percentile of the data.
<i>alpha</i>	The desired success probability.
<i>naive</i>	If true, the rank-approximate search will be performed by directly sampling the whole set instead of using the stratified sampling on the tree.
<i>sampleAtLeaves</i>	Sample at leaves for faster but less accurate computation.
<i>firstLeafExact</i>	Traverse to the first leaf without approximation.
<i>singleSampleLimit</i>	The limit on the largest node that can be approximated by sampling.
<i>sameSet</i>	If true, the query and reference set are taken to be the same, and a query point will not return itself in the results.

39.383.4 Member Function Documentation

39.383.4.1 BaseCase()

```
double BaseCase (
    const size_t queryIndex,
    const size_t referenceIndex )
```

Get the distance from the query point to the reference point.

This will update the list of candidates with the new point if appropriate.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceIndex</i>	Index of reference point.

39.383.4.2 GetResults()

```
void GetResults (
    arma::Mat< size_t > & neighbors,
    arma::mat & distances )
```

Store the list of candidates for each query point in the given matrices.

Parameters

<i>neighbors</i>	Matrix storing lists of neighbors for each query point.
<i>distances</i>	Matrix storing distances of neighbors for each query point.

39.383.4.3 NumDistComputations()

```
size_t NumDistComputations ( ) [inline]
```

Definition at line 230 of file `ra_search_rules.hpp`.

39.383.4.4 NumEffectiveSamples()

```
size_t NumEffectiveSamples ( ) [inline]
```

Definition at line 231 of file `ra_search_rules.hpp`.

39.383.4.5 Rescore() [1/2]

```
double Rescore (
    const size_t queryIndex,
    TreeType & referenceNode,
    const double oldScore )
```

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while `DBL_MAX` indicates that the node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

For rank-approximation, it also checks if the number of samples left for a query to satisfy the rank constraint is small enough at this point of the algorithm, then this node is approximated by sampling and given a new score of '`DBL_MAX`'.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.
<i>oldScore</i>	Old score produced by Score() (p. 1696) (or Rescore() (p. 1695)).

39.383.4.6 Rescore() [2/2]

```
double Rescore (
    TreeType & queryNode,
    TreeType & referenceNode,
    const double oldScore )
```

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

For the rank-approximation, we check if the referenceNode can be approximated by sampling. If it can be, enough samples are made for every query in the queryNode. No further query-tree traversal is performed.

The 'NumSamplesMade' query stat is propagated up the tree. And then if pruning occurs (by distance or by sampling), the 'NumSamplesMade' stat is not propagated down the tree. If no pruning occurs, the stat is propagated down the tree.

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.
<i>oldScore</i>	Old score produced by Socre() (or Rescore() (p. 1695)).

39.383.4.7 Score() [1/4]

```
double Score (
    const size_t queryIndex,
    TreeType & referenceNode )
```

Get the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

For rank-approximation, the scoring function first checks if pruning by distance is possible. If yes, then the node is given the score of 'DBL_MAX' and the expected number of samples from that node are added to the number of samples made for the query.

If no, then the function tries to see if the node can be pruned by approximation. If number of samples required from this node is small enough, then that number of samples are acquired from this node and the score is set to be 'DBL_MAX'.

If the pruning by approximation is not possible either, the algorithm continues with the usual tree-traversal.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.

Referenced by RASearchRules< SortPolicy, MetricType, TreeType >::TraverseAllInfo().

39.383.4.8 Score() [2/4]

```
double Score (
    const size_t queryIndex,
    TreeType & referenceNode,
    const double baseCaseResult )
```

Get the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

For rank-approximation, the scoring function first checks if pruning by distance is possible. If yes, then the node is given the score of 'DBL_MAX' and the expected number of samples from that node are added to the number of samples made for the query.

If no, then the function tries to see if the node can be pruned by approximation. If number of samples required from this node is small enough, then that number of samples are acquired from this node and the score is set to be 'DBL_MAX'.

If the pruning by approximation is not possible either, the algorithm continues with the usual tree-traversal.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.
<i>baseCaseResult</i>	Result of BaseCase(queryIndex, referenceNode).

39.383.4.9 Score() [3/4]

```
double Score (
    TreeType & queryNode,
    TreeType & referenceNode )
```

Get the score for recursion order.

A low score indicates priority for recursion while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

For the rank-approximation, we check if the `referenceNode` can be approximated by sampling. If it can be, enough samples are made for every query in the `queryNode`. No further query-tree traversal is performed.

The 'NumSamplesMade' query stat is propagated up the tree. And then if pruning occurs (by distance or by sampling), the 'NumSamplesMade' stat is not propagated down the tree. If no pruning occurs, the stat is propagated down the tree.

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.

39.383.4.10 Score() [4/4]

```
double Score (
    TreeType & queryNode,
    TreeType & referenceNode,
    const double baseCaseResult )
```

Get the score for recursion order, passing the base case result (in the situation where it may be needed to calculate the recursion order).

A low score indicates priority for recursion, while `DBL_MAX` indicates that the node should not be recursed into at all (it should be pruned).

For the rank-approximation, we check if the `referenceNode` can be approximated by sampling. If it can be, enough samples are made for every query in the `queryNode`. No further query-tree traversal is performed.

The 'NumSamplesMade' query stat is propagated up the tree. And then if pruning occurs (by distance or by sampling), the 'NumSamplesMade' stat is not propagated down the tree. If no pruning occurs, the stat is propagated down the tree.

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.
<i>baseCaseResult</i>	Result of <code>BaseCase(queryIndex, referenceNode)</code> .

39.383.4.11 TraversalInfo() [1/2]

```
const TraversalInfoType& TraversalInfo ( ) const [inline]
```

Definition at line 241 of file `ra_search_rules.hpp`.

39.383.4.12 TraversalInfo() [2/2]

```
TraversalInfoType& TraversalInfo ( ) [inline]
```

Definition at line 242 of file ra_search_rules.hpp.

References RASearchRules< SortPolicy, MetricType, TreeType >::Score().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ **ra_search_rules.hpp**

39.384 RAUtil Class Reference

Static Public Member Functions

- static size_t **MinimumSamplesReqd** (const size_t n, const size_t k, const double tau, const double **alpha**)
Compute the minimum number of samples required to guarantee the given rank-approximation and success probability.
- static void **ObtainDistinctSamples** (const size_t numSamples, const size_t rangeUpperBound, arma::uvec &distinctSamples)
Pick up desired number of samples (with replacement) from a given range of integers so that only the distinct samples are returned from the range [0 - specified upper bound)
- static double **SuccessProbability** (const size_t n, const size_t k, const size_t m, const size_t t)
Compute the success probability of obtaining 'k'-neighbors from a set of size 'n' within the top 't' neighbors if 'm' samples are made.

39.384.1 Detailed Description

Definition at line 21 of file ra_util.hpp.

39.384.2 Member Function Documentation

39.384.2.1 MinimumSamplesReqd()

```
static size_t MinimumSamplesReqd (
    const size_t n,
    const size_t k,
    const double tau,
    const double alpha ) [static]
```

Compute the minimum number of samples required to guarantee the given rank-approximation and success probability.

Parameters

<i>n</i>	Size of the set to be sampled from.
<i>k</i>	The number of neighbors required within the rank-approximation.
<i>tau</i>	The rank-approximation in percentile of the data.
<i>alpha</i>	The success probability desired.

39.384.2.2 ObtainDistinctSamples()

```
static void ObtainDistinctSamples (
    const size_t numSamples,
    const size_t rangeUpperBound,
    arma::uvec & distinctSamples ) [static]
```

Pick up desired number of samples (with replacement) from a given range of integers so that only the distinct samples are returned from the range [0 - specified upper bound)

Parameters

<i>numSamples</i>	Number of random samples.
<i>rangeUpperBound</i>	The upper bound on the range of integers.
<i>distinctSamples</i>	The list of the distinct samples.

39.384.2.3 SuccessProbability()

```
static double SuccessProbability (
    const size_t n,
    const size_t k,
    const size_t m,
    const size_t t ) [static]
```

Compute the success probability of obtaining 'k'-neighbors from a set of size 'n' within the top 't' neighbors if 'm' samples are made.

Parameters

<i>n</i>	Size of the set being sampled from.
<i>k</i>	The number of neighbors required within the rank-approximation.
<i>m</i>	The number of random samples.
<i>t</i>	The desired rank-approximation.

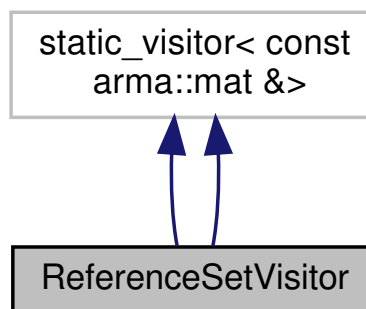
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ **ra_util.hpp**

39.385 ReferenceSetVisitor Class Reference

ReferenceSetVisitor (p. 1701) exposes the referenceSet of the given NType.

Inheritance diagram for ReferenceSetVisitor:



Public Member Functions

- `template<typename NType >`
`const arma::mat & operator() (NType *ns) const`
Return the reference set.
- `template<typename RType >`
`const arma::mat & operator() (RType *ra) const`
Return the reference set.

39.385.1 Detailed Description

ReferenceSetVisitor (p. 1701) exposes the referenceSet of the given NType.

Exposes the referenceSet of the given RType.

Definition at line 218 of file `ns_model.hpp`.

39.385.2 Member Function Documentation

39.385.2.1 operator>() [1/2]

```
const arma::mat& operator() (
    NSType * ns ) const
```

Return the reference set.

39.385.2.2 operator>() [2/2]

```
const arma::mat& operator() (
    RAType * ra ) const
```

Return the reference set.

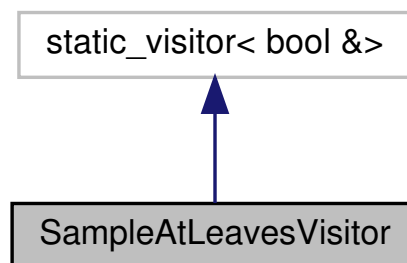
The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ **ns_model.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ **ra_model.hpp**

39.386 SampleAtLeavesVisitor Class Reference

Exposes the SampleAtLeaves() method of the given RAType.

Inheritance diagram for SampleAtLeavesVisitor:

**Public Member Functions**

- template<typename RAType >
bool & **operator()** (**RAType** *) const
Return SampleAtLeaves (whether or not sampling is done at leaves).

39.386.1 Detailed Description

Exposes the SampleAtLeaves() method of the given RType.

Definition at line 187 of file ra_model.hpp.

39.386.2 Member Function Documentation

39.386.2.1 operator()()

```
bool& operator() (
    RType * ) const
```

Return SampleAtLeaves (whether or not sampling is done at leaves).

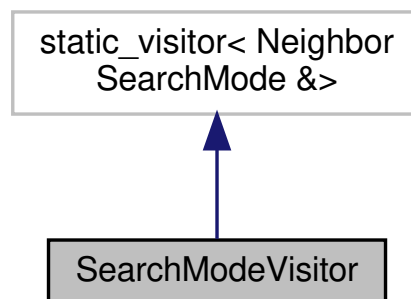
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ **ra_model.hpp**

39.387 SearchModeVisitor Class Reference

SearchModeVisitor (p. 1703) exposes the SearchMode() method of the given NType.

Inheritance diagram for SearchModeVisitor:



Public Member Functions

- `template<typename NType >`
NeighborSearchMode & operator() (NType *ns) const
Return the search mode.

39.387.1 Detailed Description

SearchModeVisitor (p. 1703) exposes the `SearchMode()` method of the given `NSType`.

Definition at line 196 of file `ns_model.hpp`.

39.387.2 Member Function Documentation

39.387.2.1 `operator()()`

```
NeighborSearchMode& operator() (
    NSType * ns ) const
```

Return the search mode.

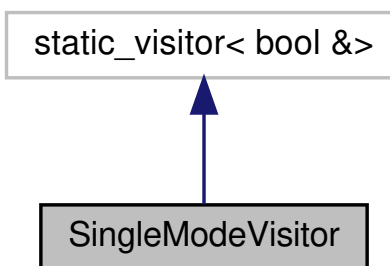
The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ ns_model.hpp`

39.388 SingleModeVisitor Class Reference

Exposes the `SingleMode()` method of the given `RAType`.

Inheritance diagram for `SingleModeVisitor`:



Public Member Functions

- `template<typename RAType >`
`bool & operator() (RAType *ra) const`

*Get a reference to the `SingleMode` parameter of the given **RASearch** (p. 1681) object.*

39.388.1 Detailed Description

Exposes the SingleMode() method of the given RAType.

Definition at line 220 of file ra_model.hpp.

39.388.2 Member Function Documentation

39.388.2.1 operator()()

```
bool& operator() (
    RAType * ra ) const
```

Get a reference to the SingleMode parameter of the given **RASearch** (p. 1681) object.

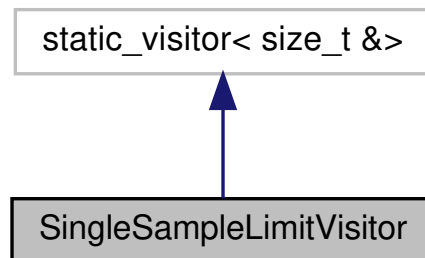
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ **ra_model.hpp**

39.389 SingleSampleLimitVisitor Class Reference

Exposes the SingleSampleLimit() method of the given RAType.

Inheritance diagram for SingleSampleLimitVisitor:



Public Member Functions

- template<typename RAType >
size_t & **operator()** (**RAType** *ra) const

39.389.1 Detailed Description

Exposes the SingleSampleLimit() method of the given RAType.

Definition at line 167 of file ra_model.hpp.

39.389.2 Member Function Documentation

39.389.2.1 operator()()

```
size_t& operator() (
    RAType * ra ) const
```

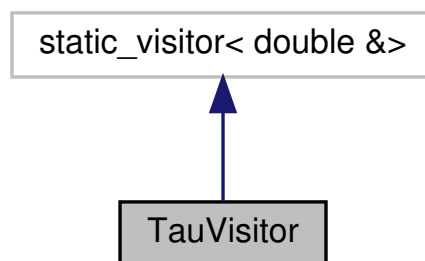
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ ra_model.hpp

39.390 TauVisitor Class Reference

Exposes the Tau() method of the given RAType.

Inheritance diagram for TauVisitor:



Public Member Functions

- template<typename RAType >
double & **operator()** (RAType *ra) const
Get a reference to the Tau parameter.

39.390.1 Detailed Description

Exposes the Tau() method of the given RAType.

Definition at line 209 of file ra_model.hpp.

39.390.2 Member Function Documentation

39.390.2.1 operator()()

```
double& operator() (
    RAType * ra ) const
```

Get a reference to the Tau parameter.

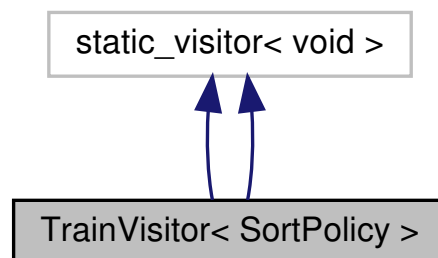
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ ra_model.hpp

39.391 TrainVisitor< SortPolicy > Class Template Reference

TrainVisitor (p. 1707) sets the reference set to a new reference set on the given NSType.

Inheritance diagram for TrainVisitor< SortPolicy >:



Public Types

- `template<template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType> using NSTypeT = NSType< SortPolicy, TreeType >`
Alias template necessary for visual c++ compiler.
- `template<template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType> using RATypeT = RAType< SortPolicy, TreeType >`
Alias template necessary for visual c++ compiler.

Public Member Functions

- **TrainVisitor** (arma::mat &&referenceSet, const size_t leafSize)
Construct the **TrainVisitor** (p. 1707) object with the given reference set, leafSize for BinarySpaceTrees.
- **TrainVisitor** (arma::mat &&referenceSet, const size_t leafSize, const double tau, const double rho)
Construct the **TrainVisitor** (p. 1707) object with the given reference set, leafSize for BinarySpaceTrees, and tau and rho for spill trees.
- template<template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType>
void **operator()** (**RATypeT**< TreeType > *ra) const
Default Train on the given RAType instance.
- void **operator()** (**RATypeT**< **tree::KDTree** > *ra) const
Train on the given RAType specialized for KDTrees.
- void **operator()** (**RATypeT**< **tree::Octree** > *ra) const
Train on the given RAType specialized for Octrees.
- template<template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType>
void **operator()** (**NSTypeT**< TreeType > *ns) const
Default Train on the given NSType instance.
- void **operator()** (**NSTypeT**< **tree::KDTree** > *ns) const
Train on the given NSType specialized for KDTrees.
- void **operator()** (**NSTypeT**< **tree::BallTree** > *ns) const
Train on the given NSType specialized for BallTrees.
- void **operator()** (**SpillKNN** *ns) const
Train specialized for SPTrees.
- void **operator()** (**NSTypeT**< **tree::Octree** > *ns) const
Train specialized for octrees.

39.391.1 Detailed Description

```
template<typename SortPolicy>
class mlpack::neighbor::TrainVisitor< SortPolicy >
```

TrainVisitor (p. 1707) sets the reference set to a new reference set on the given NSType.

TrainVisitor (p. 1707) sets the reference set to a new reference set on the given RAType.

We use template specialization to differentiate those tree types that accept leafSize as a parameter. In these cases, a reference tree with proper leafSize is built from the referenceSet.

We use template specialization to differentiate those trees that accept leafSize as a parameter. In these cases, a reference tree with proper leafSize is built from the referenceSet.

Definition at line 35 of file neighbor_search.hpp.

39.391.2 Member Typedef Documentation

39.391.2.1 NTypeT

```
using NTypeT = NType<SortPolicy, TreeType>
```

Alias template necessary for visual c++ compiler.

Definition at line 165 of file ns_model.hpp.

39.391.2.2 RTypeT

```
using RTypeT = RType<SortPolicy, TreeType>
```

Alias template necessary for visual c++ compiler.

Definition at line 144 of file ra_model.hpp.

39.391.3 Constructor & Destructor Documentation

39.391.3.1 TrainVisitor() [1/2]

```
TrainVisitor (
    arma::mat && referenceSet,
    const size_t leafSize,
    const double tau,
    const double rho )
```

Construct the **TrainVisitor** (p. 1707) object with the given reference set, leafSize for BinarySpaceTrees, and tau and rho for spill trees.

39.391.3.2 TrainVisitor() [2/2]

```
TrainVisitor (
    arma::mat && referenceSet,
    const size_t leafSize )
```

Construct the **TrainVisitor** (p. 1707) object with the given reference set, leafSize for BinarySpaceTrees.

39.391.4 Member Function Documentation

39.391.4.1 operator() [1/8]

```
void operator() (
    RATypeT< TreeType > * ra ) const
```

Default Train on the given RAType instance.

39.391.4.2 operator() [2/8]

```
void operator() (
    RATypeT< tree::KDTree > * ra ) const
```

Train on the given RAType specialized for KDTrees.

39.391.4.3 operator() [3/8]

```
void operator() (
    RATypeT< tree::Octree > * ra ) const
```

Train on the given RAType specialized for Octrees.

39.391.4.4 operator() [4/8]

```
void operator() (
    NSTypeT< TreeType > * ns ) const
```

Default Train on the given NSType instance.

39.391.4.5 operator() [5/8]

```
void operator() (
    NSTypeT< tree::KDTree > * ns ) const
```

Train on the given NSType specialized for KDTrees.

39.391.4.6 operator>() [6/8]

```
void operator() (
    NSTypeT< tree::BallTree > * ns ) const
```

Train on the given NSType specialized for BallTrees.

39.391.4.7 operator>() [7/8]

```
void operator() (
    SpillKNN * ns ) const
```

Train specialized for SPTrees.

39.391.4.8 operator>() [8/8]

```
void operator() (
    NSTypeT< tree::Octree > * ns ) const
```

Train specialized for octrees.

The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ **neighbor_search.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ **ns_model.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ **ra_model.hpp**

39.392 SparseAutoencoder Class Reference

A sparse autoencoder is a neural network whose aim to learn compressed representations of the data, typically for dimensionality reduction, with a constraint on the activity of the neurons in the network.

Public Member Functions

- `template<typename OptimizerType = ens::L_BFGS>`
SparseAutoencoder (const arma::mat &data, const size_t visibleSize, const size_t hiddenSize, const double lambda=0.0001, const double beta=3, const double rho=0.01, OptimizerType optimizer=OptimizerType())
Construct the sparse autoencoder model with the given training data.
- `void Beta (const double b)`
Sets the KL divergence parameter.
- `double Beta () const`
Gets the KL divergence parameter.
- `void GetNewFeatures (arma::mat &data, arma::mat &features)`
Transforms the provided data into the representation learned by the sparse autoencoder.
- `void HiddenSize (const size_t hidden)`
Sets size of the hidden layer.
- `size_t HiddenSize () const`
Gets the size of the hidden layer.
- `void Lambda (const double l)`
Sets the L2-regularization parameter.
- `double Lambda () const`
Gets the L2-regularization parameter.
- `void Rho (const double r)`
Sets the sparsity parameter.
- `double Rho () const`
Gets the sparsity parameter.
- `void Sigmoid (const arma::mat &x, arma::mat &output) const`
Returns the elementwise sigmoid of the passed matrix, where the sigmoid function of a real number 'x' is $1 / (1 + \exp(-x))$.
- `void VisibleSize (const size_t visible)`
Sets size of the visible layer.
- `size_t VisibleSize () const`
Gets size of the visible layer.

39.392.1 Detailed Description

A sparse autoencoder is a neural network whose aim to learn compressed representations of the data, typically for dimensionality reduction, with a constraint on the activity of the neurons in the network.

Sparse autoencoders can be stacked together to learn a hierarchy of features, which provide a better representation of the data for classification. This is a method used in the recently developed field of deep learning. More technical details about the model can be found on the following webpage:

http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial

An example of how to use the interface is shown below:


```

arma::mat data; // Data matrix.
const size_t vSize = 64; // Size of visible layer, depends on the data.
const size_t hSize = 25; // Size of hidden layer, depends on requirements.

// Train the model using default options.
SparseAutoencoder encoder1(data, vSize, hSize);

const size_t numBasis = 5; // Parameter required for L-BFGS algorithm.
const size_t numIterations = 100; // Maximum number of iterations.

// Use an instantiated optimizer for the training.
SparseAutoencoderFunction saf(data, vSize, hSize);
L_BFGS<SparseAutoencoderFunction> optimizer(saf, numBasis, numIterations);
SparseAutoencoder<L_BFGS> encoder2(optimizer);

arma::mat features1, features2; // Matrices for storing new representations.

// Get new representations from the trained models.
encoder1.GetNewFeatures(data, features1);
encoder2.GetNewFeatures(data, features2);

```

This implementation allows the use of arbitrary mlpack optimizers via the `OptimizerType` template parameter.

Definition at line 63 of file `sparse_autoencoder.hpp`.

39.392.2 Constructor & Destructor Documentation

39.392.2.1 SparseAutoencoder()

```

SparseAutoencoder (
    const arma::mat & data,
    const size_t visibleSize,
    const size_t hiddenSize,
    const double lambda = 0.0001,
    const double beta = 3,
    const double rho = 0.01,
    OptimizerType optimizer = OptimizerType() )

```

Construct the sparse autoencoder model with the given training data.

This will train the model. The parameters 'lambda', 'beta' and 'rho' can be set optionally. Changing these parameters will have an effect on regularization and sparsity of the model.

Template Parameters

<i>OptimizerType</i>	The optimizer to use.
----------------------	-----------------------

Parameters

<i>data</i>	Input data with each column as one example.
<i>visibleSize</i>	Size of input vector expected at the visible layer.
<i>hiddenSize</i>	Size of input vector expected at the hidden layer.

Parameters

<i>lambda</i>	L2-regularization parameter.
<i>beta</i>	KL divergence parameter.
<i>rho</i>	Sparsity parameter.

39.392.3 Member Function Documentation**39.392.3.1 Beta()** [1/2]

```
void Beta (
    const double b ) [inline]
```

Sets the KL divergence parameter.

Definition at line 147 of file sparse_autoencoder.hpp.

39.392.3.2 Beta() [2/2]

```
double Beta ( ) const [inline]
```

Gets the KL divergence parameter.

Definition at line 153 of file sparse_autoencoder.hpp.

39.392.3.3 GetNewFeatures()

```
void GetNewFeatures (
    arma::mat & data,
    arma::mat & features )
```

Transforms the provided data into the representation learned by the sparse autoencoder.

The function basically performs a feedforward computation using the learned weights, and returns the hidden layer activations.

Parameters

<i>data</i>	Matrix of the provided data.
<i>features</i>	The hidden layer representation of the provided data.

39.392.3.4 HiddenSize() [1/2]

```
void HiddenSize (
    const size_t hidden ) [inline]
```

Sets size of the hidden layer.

Definition at line 123 of file sparse_autoencoder.hpp.

39.392.3.5 HiddenSize() [2/2]

```
size_t HiddenSize ( ) const [inline]
```

Gets the size of the hidden layer.

Definition at line 129 of file sparse_autoencoder.hpp.

39.392.3.6 Lambda() [1/2]

```
void Lambda (
    const double l ) [inline]
```

Sets the L2-regularization parameter.

Definition at line 135 of file sparse_autoencoder.hpp.

39.392.3.7 Lambda() [2/2]

```
double Lambda ( ) const [inline]
```

Gets the L2-regularization parameter.

Definition at line 141 of file sparse_autoencoder.hpp.

39.392.3.8 Rho() [1/2]

```
void Rho (
    const double r ) [inline]
```

Sets the sparsity parameter.

Definition at line 159 of file sparse_autoencoder.hpp.

39.392.3.9 Rho() [2/2]

```
double Rho ( ) const [inline]
```

Gets the sparsity parameter.

Definition at line 165 of file sparse_autoencoder.hpp.

39.392.3.10 Sigmoid()

```
void Sigmoid (
    const arma::mat & x,
    arma::mat & output ) const [inline]
```

Returns the elementwise sigmoid of the passed matrix, where the sigmoid function of a real number 'x' is $[1 / (1 + \exp(-x))]$.

Parameters

<i>x</i>	Matrix of real values for which we require the sigmoid activation.
----------	--

Definition at line 105 of file sparse_autoencoder.hpp.

39.392.3.11 VisibleSize() [1/2]

```
void VisibleSize (
    const size_t visible ) [inline]
```

Sets size of the visible layer.

Definition at line 111 of file sparse_autoencoder.hpp.

39.392.3.12 VisibleSize() [2/2]

```
size_t VisibleSize ( ) const [inline]
```

Gets size of the visible layer.

Definition at line 117 of file sparse_autoencoder.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_autoencoder/ **sparse_autoencoder.hpp**

39.393 SparseAutoencoderFunction Class Reference

This is a class for the sparse autoencoder objective function.

Public Member Functions

- **SparseAutoencoderFunction** (const arma::mat &data, const size_t visibleSize, const size_t hiddenSize, const double lambda=0.0001, const double beta=3, const double rho=0.01)
Construct the sparse autoencoder objective function with the given parameters.
- void **Beta** (const double b)
Sets the KL divergence parameter.
- double **Beta** () const
Gets the KL divergence parameter.
- double **Evaluate** (const arma::mat ¶meters) const
Evaluates the objective function of the sparse autoencoder model using the given parameters.
- const arma::mat & **GetInitialPoint** () const
Return the initial point for the optimization.
- void **Gradient** (const arma::mat ¶meters, arma::mat &gradient) const
Evaluates the gradient values of the objective function given the current set of parameters.
- void **HiddenSize** (const size_t hidden)
Sets size of the hidden layer.
- size_t **HiddenSize** () const
Gets the size of the hidden layer.
- const arma::mat **InitializeWeights** ()
Initializes the parameters of the model to suitable values.
- void **Lambda** (const double l)
Sets the L2-regularization parameter.
- double **Lambda** () const
Gets the L2-regularization parameter.
- void **Rho** (const double r)
Sets the sparsity parameter.
- double **Rho** () const
Gets the sparsity parameter.
- void **Sigmoid** (const arma::mat &x, arma::mat &output) const
Returns the elementwise sigmoid of the passed matrix, where the sigmoid function of a real number 'x' is $[1 / (1 + \exp(-x))]$.
- void **VisibleSize** (const size_t visible)
Sets size of the visible layer.
- size_t **VisibleSize** () const
Gets size of the visible layer.

39.393.1 Detailed Description

This is a class for the sparse autoencoder objective function.

It can be used to create learning models like self-taught learning, stacked autoencoders, conditional random fields (CRFs), and so forth.

Definition at line 26 of file `sparse_autoencoder_function.hpp`.

39.393.2 Constructor & Destructor Documentation

39.393.2.1 SparseAutoencoderFunction()

```
SparseAutoencoderFunction (
    const arma::mat & data,
    const size_t visibleSize,
    const size_t hiddenSize,
    const double lambda = 0.0001,
    const double beta = 3,
    const double rho = 0.01 )
```

Construct the sparse autoencoder objective function with the given parameters.

Parameters

<i>data</i>	The data matrix.
<i>visibleSize</i>	Size of input vector expected at the visible layer.
<i>hiddenSize</i>	Size of input vector expected at the hidden layer.
<i>lambda</i>	L2-regularization parameter.
<i>beta</i>	KL divergence parameter.
<i>rho</i>	Sparsity parameter.

39.393.3 Member Function Documentation

39.393.3.1 Beta() [1/2]

```
void Beta (
    const double b ) [inline]
```

Sets the KL divergence parameter.

Definition at line 124 of file `sparse_autoencoder_function.hpp`.

39.393.3.2 Beta() [2/2]

```
double Beta ( ) const [inline]
```

Gets the KL divergence parameter.

Definition at line 130 of file `sparse_autoencoder_function.hpp`.

39.393.3.3 Evaluate()

```
double Evaluate (
    const arma::mat & parameters ) const
```

Evaluates the objective function of the sparse autoencoder model using the given parameters.

The cost function has terms for the reconstruction error, regularization cost and the sparsity cost. The objective function takes a low value when the model is able to reconstruct the data well using weights which are low in value and when the average activations of neurons in the hidden layers agrees well with the sparsity parameter 'rho'.

Parameters

<i>parameters</i>	Current values of the model parameters.
-------------------	---

39.393.3.4 GetInitialPoint()

```
const arma::mat& GetInitialPoint ( ) const [inline]
```

Return the initial point for the optimization.

Definition at line 85 of file `sparse_autoencoder_function.hpp`.

39.393.3.5 Gradient()

```
void Gradient (
    const arma::mat & parameters,
    arma::mat & gradient ) const
```

Evaluates the gradient values of the objective function given the current set of parameters.

The function performs a feedforward pass and computes the error in reconstructing the data points. It then uses the backpropagation algorithm to compute the gradient values.

Parameters

<i>parameters</i>	Current values of the model parameters.
<i>gradient</i>	Matrix where gradient values will be stored.

39.393.3.6 HiddenSize() [1/2]

```
void HiddenSize (
    const size_t hidden ) [inline]
```

Sets size of the hidden layer.

Definition at line 100 of file sparse_autoencoder_function.hpp.

39.393.3.7 HiddenSize() [2/2]

```
size_t HiddenSize ( ) const [inline]
```

Gets the size of the hidden layer.

Definition at line 106 of file sparse_autoencoder_function.hpp.

39.393.3.8 InitializeWeights()

```
const arma::mat InitializeWeights ( )
```

Initializes the parameters of the model to suitable values.

39.393.3.9 Lambda() [1/2]

```
void Lambda (
    const double l ) [inline]
```

Sets the L2-regularization parameter.

Definition at line 112 of file sparse_autoencoder_function.hpp.

39.393.3.10 Lambda() [2/2]

```
double Lambda ( ) const [inline]
```

Gets the L2-regularization parameter.

Definition at line 118 of file `sparse_autoencoder_function.hpp`.

39.393.3.11 Rho() [1/2]

```
void Rho (
    const double r ) [inline]
```

Sets the sparsity parameter.

Definition at line 136 of file `sparse_autoencoder_function.hpp`.

39.393.3.12 Rho() [2/2]

```
double Rho ( ) const [inline]
```

Gets the sparsity parameter.

Definition at line 142 of file `sparse_autoencoder_function.hpp`.

39.393.3.13 Sigmoid()

```
void Sigmoid (
    const arma::mat & x,
    arma::mat & output ) const [inline]
```

Returns the elementwise sigmoid of the passed matrix, where the sigmoid function of a real number 'x' is $[1 / (1 + \exp(-x))]$.

Parameters

x	Matrix of real values for which we require the sigmoid activation.
---	--

Definition at line 79 of file `sparse_autoencoder_function.hpp`.

39.393.3.14 VisibleSize() [1/2]

```
void VisibleSize (
    const size_t visible ) [inline]
```

Sets size of the visible layer.

Definition at line 88 of file `sparse_autoencoder_function.hpp`.

39.393.3.15 VisibleSize() [2/2]

```
size_t VisibleSize ( ) const [inline]
```

Gets size of the visible layer.

Definition at line 94 of file `sparse_autoencoder_function.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_autoencoder/function.hpp` **sparse_autoencoder_↔**

39.394 ExactSVDPolicy Class Reference

Implementation of the exact SVD policy.

Public Member Functions

- void **Apply** (const arma::mat &data, const arma::mat ¢eredData, arma::mat &transformedData, arma::vec &eigVal, arma::mat &eigvec, const size_t)

Apply Principal Component Analysis to the provided data set using the exact SVD method.

39.394.1 Detailed Description

Implementation of the exact SVD policy.

Definition at line 27 of file `exact_svd_method.hpp`.

39.394.2 Member Function Documentation**39.394.2.1 Apply()**

```
void Apply (
    const arma::mat & data,
    const arma::mat & centeredData,
    arma::mat & transformedData,
    arma::vec & eigVal,
    arma::mat & eigvec,
    const size_t ) [inline]
```

Apply Principal Component Analysis to the provided data set using the exact SVD method.

Parameters

<i>data</i>	Data matrix.
<i>centeredData</i>	Centered data matrix.
<i>transformedData</i>	Matrix to put results of PCA (p. 1723) into.
<i>eigVal</i>	Vector to put eigenvalues into.
<i>eigvec</i>	Matrix to put eigenvectors (loadings) into.
<i>rank</i>	Rank of the decomposition.

Definition at line 41 of file `exact_svd_method.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/decomposition_policies/ exact_svd_method.↵
hpp`

39.395 PCA< DecompositionPolicy > Class Template Reference

This class implements principal components analysis (**PCA** (p. 1723)).

Public Member Functions

- **PCA** (const bool scaleData=false, const DecompositionPolicy &decomposition=DecompositionPolicy())
*Create the **PCA** (p. 1723) object, specifying if the data should be scaled in each dimension by standard deviation when **PCA** (p. 1723) is performed.*
- void **Apply** (const arma::mat &data, arma::mat &transformedData, arma::vec &eigVal, arma::mat &eigvec)
Apply Principal Component Analysis to the provided data set.
- void **Apply** (const arma::mat &data, arma::mat &transformedData, arma::vec &eigVal)
Apply Principal Component Analysis to the provided data set.
- double **Apply** (arma::mat &data, const size_t newDimension)
*Use **PCA** (p. 1723) for dimensionality reduction on the given dataset.*
- double **Apply** (arma::mat &data, const int newDimension)
This overload is here to make sure int gets casted right to size_t.
- double **Apply** (arma::mat &data, const double varRetained)
*Use **PCA** (p. 1723) for dimensionality reduction on the given dataset.*
- bool **ScaleData** () const
*Get whether or not this **PCA** (p. 1723) object will scale (by standard deviation) the data when **PCA** (p. 1723) is performed.*
- bool & **ScaleData** ()
*Modify whether or not this **PCA** (p. 1723) object will scale (by standard deviation) the data when **PCA** (p. 1723) is performed.*

39.395.1 Detailed Description

```
template<typename DecompositionPolicy = ExactSVDPolicy>
class mlpack::pca::PCA< DecompositionPolicy >
```

This class implements principal components analysis (**PCA** (p. 1723)).

This is a common, widely-used technique that is often used for either dimensionality reduction or transforming data into a better basis. Further information on **PCA** (p. 1723) can be found in almost any statistics or machine learning textbook, and all over the internet.

Definition at line 33 of file pca.hpp.

39.395.2 Constructor & Destructor Documentation

39.395.2.1 PCA()

```
PCA (
    const bool scaleData = false,
    const DecompositionPolicy & decomposition = DecompositionPolicy() )
```

Create the **PCA** (p. 1723) object, specifying if the data should be scaled in each dimension by standard deviation when **PCA** (p. 1723) is performed.

Parameters

<i>scaleData</i>	Whether or not to scale the data.
------------------	-----------------------------------

39.395.3 Member Function Documentation

39.395.3.1 Apply() [1/5]

```
void Apply (
    const arma::mat & data,
    arma::mat & transformedData,
    arma::vec & eigVal,
    arma::mat & eigvec )
```

Apply Principal Component Analysis to the provided data set.

It is safe to pass the same matrix reference for both data and transformedData.

Parameters

<i>data</i>	Data matrix.
<i>transformedData</i>	Matrix to put results of PCA (p. 1723) into.
<i>eigval</i>	Vector to put eigenvalues into.
<i>eigvec</i>	Matrix to put eigenvectors (loadings) into.

Referenced by PCA< DecompositionPolicy >::Apply().

39.395.3.2 Apply() [2/5]

```
void Apply (
    const arma::mat & data,
    arma::mat & transformedData,
    arma::vec & eigVal )
```

Apply Principal Component Analysis to the provided data set.

It is safe to pass the same matrix reference for both data and transformedData.

Parameters

<i>data</i>	Data matrix.
<i>transformedData</i>	Matrix to store results of PCA (p. 1723) in.
<i>eigVal</i>	Vector to put eigenvalues into.

39.395.3.3 Apply() [3/5]

```
double Apply (
    arma::mat & data,
    const size_t newDimension )
```

Use **PCA** (p. 1723) for dimensionality reduction on the given dataset.

This will save the newDimension largest principal components of the data and remove the rest. The parameter returned is the amount of variance of the data that is retained; this is a value between 0 and 1. For instance, a value of 0.9 indicates that 90% of the variance present in the data was retained.

Parameters

<i>data</i>	Data matrix.
<i>newDimension</i>	New dimension of the data.

Returns

Amount of the variance of the data retained (between 0 and 1).

39.395.3.4 Apply() [4/5]

```
double Apply (
    arma::mat & data,
    const int newDimension ) [inline]
```

This overload is here to make sure int gets casted right to size_t.

Definition at line 85 of file pca.hpp.

References `PCA< DecompositionPolicy >::Apply()`.

39.395.3.5 Apply() [5/5]

```
double Apply (
    arma::mat & data,
    const double varRetained )
```

Use **PCA** (p. 1723) for dimensionality reduction on the given dataset.

This will save as many dimensions as necessary to retain at least the given amount of variance (specified by parameter `varRetained`). The amount should be between 0 and 1; if the amount is 0, then only 1 dimension will be retained. If the amount is 1, then all dimensions will be retained.

The method returns the actual amount of variance retained, which will always be greater than or equal to the `varRetained` parameter.

Parameters

<i>data</i>	Data matrix.
<i>varRetained</i>	Lower bound on amount of variance to retain; should be between 0 and 1.

Returns

Actual amount of variance retained (between 0 and 1).

39.395.3.6 ScaleData() [1/2]

```
bool ScaleData ( ) const [inline]
```

Get whether or not this **PCA** (p. 1723) object will scale (by standard deviation) the data when **PCA** (p. 1723) is performed.

Definition at line 109 of file pca.hpp.

Referenced by PCA< DecompositionPolicy >::ScaleData().

39.395.3.7 ScaleData() [2/2]

```
bool& ScaleData ( ) [inline]
```

Modify whether or not this **PCA** (p. 1723) object will scale (by standard deviation) the data when **PCA** (p. 1723) is performed.

Definition at line 112 of file pca.hpp.

References PCA< DecompositionPolicy >::ScaleData().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/ **pca.hpp**

39.396 QUICSVDPolicy Class Reference

Implementation of the QUIC-SVD policy.

Public Member Functions

- **QUICSVDPolicy** (const double epsilon=0.03, const double delta=0.1)
*Use QUIC-SVD method to perform the principal components analysis (**PCA** (p. 1723)).*
- void **Apply** (const arma::mat &data, const arma::mat ¢eredData, arma::mat &transformedData, arma::vec &eigVal, arma::mat &eigvec, const size_t)
Apply Principal Component Analysis to the provided data set using the QUIC-SVD method.
- double **Delta** () const
Get the cumulative probability for Monte Carlo error lower bound.
- double & **Delta** ()
Modify the cumulative probability for Monte Carlo error lower bound.
- double **Epsilon** () const
Get the error tolerance fraction for calculated subspace.
- double & **Epsilon** ()
Modify the error tolerance fraction for calculated subspace.

39.396.1 Detailed Description

Implementation of the QUIC-SVD policy.

Definition at line 26 of file quic_svd_method.hpp.

39.396.2 Constructor & Destructor Documentation

39.396.2.1 QUICSVDPolicy()

```
QUICSVDPolicy (
    const double epsilon = 0.03,
    const double delta = 0.1 ) [inline]
```

Use QUIC-SVD method to perform the principal components analysis (**PCA** (p. 1723)).

Parameters

<i>epsilon</i>	Error tolerance fraction for calculated subspace.
<i>delta</i>	Cumulative probability for Monte Carlo error lower bound.

Definition at line 35 of file quic_svd_method.hpp.

39.396.3 Member Function Documentation

39.396.3.1 Apply()

```
void Apply (
    const arma::mat & data,
    const arma::mat & centeredData,
    arma::mat & transformedData,
    arma::vec & eigVal,
    arma::mat & eigvec,
    const size_t ) [inline]
```

Apply Principal Component Analysis to the provided data set using the QUIC-SVD method.

Parameters

<i>data</i>	Data matrix.
-------------	--------------

Parameters

<i>centeredData</i>	Centered data matrix.
<i>transformedData</i>	Matrix to put results of PCA (p. 1723) into.
<i>eigVal</i>	Vector to put eigenvalues into.
<i>eigvec</i>	Matrix to put eigenvectors (loadings) into.
<i>rank</i>	Rank of the decomposition.

Definition at line 53 of file quic_svd_method.hpp.

39.396.3.2 Delta() [1/2]

```
double Delta ( ) const [inline]
```

Get the cumulative probability for Monte Carlo error lower bound.

Definition at line 81 of file quic_svd_method.hpp.

39.396.3.3 Delta() [2/2]

```
double& Delta ( ) [inline]
```

Modify the cumulative probability for Monte Carlo error lower bound.

Definition at line 83 of file quic_svd_method.hpp.

39.396.3.4 Epsilon() [1/2]

```
double Epsilon ( ) const [inline]
```

Get the error tolerance fraction for calculated subspace.

Definition at line 76 of file quic_svd_method.hpp.

39.396.3.5 Epsilon() [2/2]

```
double& Epsilon ( ) [inline]
```

Modify the error tolerance fraction for calculated subspace.

Definition at line 78 of file quic_svd_method.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/decomposition_policies/ **quic_svd_method.hpp**

39.397 RandomizedBlockKrylovSVDPolicy Class Reference

Implementation of the randomized block krylov SVD policy.

Public Member Functions

- **RandomizedBlockKrylovSVDPolicy** (const size_t maxIterations=2, const size_t blockSize=0)
*Use randomized block krylov SVD method to perform the principal components analysis (**PCA** (p. 1723)).*
- void **Apply** (const arma::mat &data, const arma::mat ¢eredData, arma::mat &transformedData, arma::vec &eigVal, arma::mat &eigvec, const size_t rank)
Apply Principal Component Analysis to the provided data set using the randomized block krylov SVD method.
- size_t **BlockSize** () const
Get the block size.
- size_t & **BlockSize** ()
Modify the block size.
- size_t **MaxIterations** () const
Get the number of iterations for the power method.
- size_t & **MaxIterations** ()
Modify the number of iterations for the power method.

39.397.1 Detailed Description

Implementation of the randomized block krylov SVD policy.

Definition at line 26 of file randomized_block_krylov_method.hpp.

39.397.2 Constructor & Destructor Documentation**39.397.2.1 RandomizedBlockKrylovSVDPolicy()**

```
RandomizedBlockKrylovSVDPolicy (
    const size_t maxIterations = 2,
    const size_t blockSize = 0 ) [inline]
```

Use randomized block krylov SVD method to perform the principal components analysis (**PCA** (p. 1723)).

Parameters

<i>maxIterations</i>	Number of iterations for the power method (Default: 2).
<i>blockSize</i>	The block size, must be \geq rank (Default: rank + 10).

Definition at line 37 of file randomized_block_krylov_method.hpp.

39.397.3 Member Function Documentation

39.397.3.1 Apply()

```
void Apply (
    const arma::mat & data,
    const arma::mat & centeredData,
    arma::mat & transformedData,
    arma::vec & eigVal,
    arma::mat & eigvec,
    const size_t rank ) [inline]
```

Apply Principal Component Analysis to the provided data set using the randomized block krylov SVD method.

Parameters

<i>data</i>	Data matrix.
<i>centeredData</i>	Centered data matrix.
<i>transformedData</i>	Matrix to put results of PCA (p. 1723) into.
<i>eigVal</i>	Vector to put eigenvalues into.
<i>eigvec</i>	Matrix to put eigenvectors (loadings) into.
<i>rank</i>	Rank of the decomposition.

Definition at line 56 of file randomized_block_krylov_method.hpp.

References RandomizedBlockKrylovSVD::Apply().

39.397.3.2 BlockSize() [1/2]

```
size_t BlockSize ( ) const [inline]
```

Get the block size.

Definition at line 86 of file randomized_block_krylov_method.hpp.

39.397.3.3 BlockSize() [2/2]

```
size_t& BlockSize ( ) [inline]
```

Modify the block size.

Definition at line 88 of file randomized_block_krylov_method.hpp.

39.397.3.4 MaxIterations() [1/2]

```
size_t MaxIterations ( ) const [inline]
```

Get the number of iterations for the power method.

Definition at line 81 of file randomized_block_krylov_method.hpp.

39.397.3.5 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Modify the number of iterations for the power method.

Definition at line 83 of file randomized_block_krylov_method.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/decomposition_policies/ **randomized_block_krylov_method.hpp**

39.398 RandomizedSVDPolicy Class Reference

Implementation of the randomized SVD policy.

Public Member Functions

- **RandomizedSVDPolicy** (const size_t iteratedPower=0, const size_t maxIterations=2)
*Use randomized SVD method to perform the principal components analysis (**PCA** (p. 1723)).*
- void **Apply** (const arma::mat &data, const arma::mat ¢eredData, arma::mat &transformedData, arma::vec &eigVal, arma::mat &eigvec, const size_t rank)
Apply Principal Component Analysis to the provided data set using the randomized SVD.
- size_t **IteratedPower** () const
Get the size of the normalized power iterations.
- size_t & **IteratedPower** ()
Modify the size of the normalized power iterations.
- size_t **MaxIterations** () const
Get the number of iterations for the power method.
- size_t & **MaxIterations** ()
Modify the number of iterations for the power method.

39.398.1 Detailed Description

Implementation of the randomized SVD policy.

Definition at line 26 of file randomized_svd_method.hpp.

39.398.2 Constructor & Destructor Documentation

39.398.2.1 RandomizedSVDPolicy()

```
RandomizedSVDPolicy (
    const size_t iteratedPower = 0,
    const size_t maxIterations = 2 ) [inline]
```

Use randomized SVD method to perform the principal components analysis (**PCA** (p. 1723)).

Parameters

<i>iteratedPower</i>	Size of the normalized power iterations (Default: rank + 2).
<i>maxIterations</i>	Number of iterations for the power method (Default: 2).

Definition at line 38 of file randomized_svd_method.hpp.

39.398.3 Member Function Documentation

39.398.3.1 Apply()

```
void Apply (
    const arma::mat & data,
    const arma::mat & centeredData,
    arma::mat & transformedData,
    arma::vec & eigVal,
    arma::mat & eigvec,
    const size_t rank ) [inline]
```

Apply Principal Component Analysis to the provided data set using the randomized SVD.

Parameters

<i>data</i>	Data matrix.
-------------	--------------

Parameters

<i>centeredData</i>	Centered data matrix.
<i>transformedData</i>	Matrix to put results of PCA (p. 1723) into.
<i>eigVal</i>	Vector to put eigenvalues into.
<i>eigvec</i>	Matrix to put eigenvectors (loadings) into.
<i>rank</i>	Rank of the decomposition.

Definition at line 57 of file randomized_svd_method.hpp.

References RandomizedSVD::Apply().

39.398.3.2 IteratedPower() [1/2]

```
size_t IteratedPower ( ) const [inline]
```

Get the size of the normalized power iterations.

Definition at line 81 of file randomized_svd_method.hpp.

39.398.3.3 IteratedPower() [2/2]

```
size_t& IteratedPower ( ) [inline]
```

Modify the size of the normalized power iterations.

Definition at line 83 of file randomized_svd_method.hpp.

39.398.3.4 MaxIterations() [1/2]

```
size_t MaxIterations ( ) const [inline]
```

Get the number of iterations for the power method.

Definition at line 86 of file randomized_svd_method.hpp.

39.398.3.5 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Modify the number of iterations for the power method.

Definition at line 88 of file randomized_svd_method.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/decomposition_policies/ **randomized_svd_method.hpp**

39.399 Perceptron< LearnPolicy, WeightInitializationPolicy, MatType > Class Template Reference

This class implements a simple perceptron (i.e., a single layer neural network).

Public Member Functions

- **Perceptron** (const size_t numClasses=0, const size_t dimensionality=0, const size_t maxIterations=1000)
Constructor: create the perceptron with the given number of classes and initialize the weight matrix, but do not perform any training.
- **Perceptron** (const MatType &data, const arma::Row< size_t > &labels, const size_t numClasses, const size_t maxIterations=1000)
Constructor: constructs the perceptron by building the weights matrix, which is later used in classification.
- **Perceptron** (const **Perceptron** &other, const MatType &data, const arma::Row< size_t > &labels, const size_t numClasses, const arma::rowvec &instanceWeights)
Alternate constructor which copies parameters from an already initiated perceptron.
- const arma::vec & **Biases** () const
Get the biases.
- arma::vec & **Biases** ()
Modify the biases. You had better know what you are doing!
- void **Classify** (const MatType &test, arma::Row< size_t > &predictedLabels)
Classification function.
- size_t **MaxIterations** () const
Get the maximum number of iterations.
- size_t & **MaxIterations** ()
Modify the maximum number of iterations.
- size_t **NumClasses** () const
Get the number of classes this perceptron has been trained for.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the perceptron.
- void **Train** (const MatType &data, const arma::Row< size_t > &labels, const size_t numClasses, const arma::rowvec &instanceWeights=arma::rowvec())
*Train the perceptron on the given data for up to the maximum number of iterations (specified in the constructor or through **MaxIterations**() (p. 1738)).*
- const arma::mat & **Weights** () const
Get the weight matrix.
- arma::mat & **Weights** ()
Modify the weight matrix. You had better know what you are doing!

39.399.1 Detailed Description

```
template<typename LearnPolicy = SimpleWeightUpdate, typename WeightInitializationPolicy = ZeroInitialization, typename MatType = arma::mat>
class mlpack::perceptron::Perceptron< LearnPolicy, WeightInitializationPolicy, MatType >
```

This class implements a simple perceptron (i.e., a single layer neural network).

It converges if the supplied training dataset is linearly separable.

Template Parameters

<i>LearnPolicy</i>	Options of SimpleWeightUpdate (p. 1741) and GradientDescent .
<i>WeightInitializationPolicy</i>	Option of ZeroInitialization (p. 1742) and RandomInitialization (p. 1740).

Definition at line 36 of file perceptron.hpp.

39.399.2 Constructor & Destructor Documentation

39.399.2.1 Perceptron() [1/3]

```
Perceptron (
    const size_t numClasses = 0,
    const size_t dimensionality = 0,
    const size_t maxIterations = 1000 )
```

Constructor: create the perceptron with the given number of classes and initialize the weight matrix, but do not perform any training.

(Call the **Train()** (p. 1739) function to perform training.)

Parameters

<i>numClasses</i>	Number of classes in the dataset.
<i>dimensionality</i>	Dimensionality of the dataset.
<i>maxIterations</i>	Maximum number of iterations for the perceptron learning algorithm.

39.399.2.2 Perceptron() [2/3]

```
Perceptron (
    const MatType & data,
```



```
const arma::Row< size_t > & labels,
const size_t numClasses,
const size_t maxIterations = 1000 )
```

Constructor: constructs the perceptron by building the weights matrix, which is later used in classification.

The number of classes should be specified separately, and the labels vector should contain values in the range [0, numClasses - 1]. The **data::NormalizeLabels()** (p. 385) function can be used if the labels vector does not contain values in the required range.

Parameters

<i>data</i>	Input, training data.
<i>labels</i>	Labels of dataset.
<i>numClasses</i>	Number of classes in the dataset.
<i>maxIterations</i>	Maximum number of iterations for the perceptron learning algorithm.

39.399.2.3 Perceptron() [3/3]

```
Perceptron (
    const Perceptron< LearnPolicy, WeightInitializationPolicy, MatType > & other,
    const MatType & data,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const arma::rowvec & instanceWeights )
```

Alternate constructor which copies parameters from an already initiated perceptron.

Parameters

<i>other</i>	The other initiated Perceptron (p. 1735) object from which we copy the values from.
<i>data</i>	The data on which to train this Perceptron (p. 1735) object on.
<i>labels</i>	The labels of data.
<i>numClasses</i>	Number of classes in the data.
<i>instanceWeights</i>	Weight vector to use while training. For boosting purposes.

39.399.3 Member Function Documentation

39.399.3.1 Biases() [1/2]

```
const arma::vec& Biases ( ) const [inline]
```

Get the biases.

Definition at line 139 of file perceptron.hpp.

39.399.3.2 Biases() [2/2]

```
arma::vec& Biases ( ) [inline]
```

Modify the biases. You had better know what you are doing!

Definition at line 141 of file perceptron.hpp.

39.399.3.3 Classify()

```
void Classify (
    const MatType & test,
    arma::Row< size_t > & predictedLabels )
```

Classification function.

After training, use the weights matrix to classify test, and put the predicted classes in predictedLabels.

Parameters

<i>test</i>	Testing data or data to classify.
<i>predictedLabels</i>	Vector to store the predicted classes after classifying test.

39.399.3.4 MaxIterations() [1/2]

```
size_t MaxIterations ( ) const [inline]
```

Get the maximum number of iterations.

Definition at line 126 of file perceptron.hpp.

39.399.3.5 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Modify the maximum number of iterations.

Definition at line 128 of file perceptron.hpp.

39.399.3.6 NumClasses()

```
size_t NumClasses ( ) const [inline]
```

Get the number of classes this perceptron has been trained for.

Definition at line 131 of file perceptron.hpp.

39.399.3.7 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the perceptron.

39.399.3.8 Train()

```
void Train (
    const MatType & data,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const arma::rowvec & instanceWeights = arma::rowvec() )
```

Train the perceptron on the given data for up to the maximum number of iterations (specified in the constructor or through **MaxIterations()** (p. 1738)).

A single iteration corresponds to a single pass through the data, so if you want to pass through the dataset only once, set **MaxIterations()** (p. 1738) to 1.

This training does not reset the model weights, so you can call **Train()** (p. 1739) on multiple datasets sequentially.

Parameters

<i>data</i>	Dataset on which training should be performed.
<i>labels</i>	Labels of the dataset.
<i>numClasses</i>	Number of classes in the data.
<i>instanceWeights</i>	Cost matrix. Stores the cost of mispredicting instances. This is useful for boosting.

39.399.3.9 Weights() [1/2]

```
const arma::mat& Weights ( ) const [inline]
```

Get the weight matrix.

Definition at line 134 of file perceptron.hpp.

39.399.3.10 Weights() [2/2]

```
arma::mat& Weights ( ) [inline]
```

Modify the weight matrix. You had better know what you are doing!

Definition at line 136 of file perceptron.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/ **perceptron.hpp**

39.400 RandomInitialization Class Reference

This class is used to initialize weights for the weightVectors matrix in a random manner.

Public Member Functions

- **RandomInitialization** ()

Static Public Member Functions

- static void **Initialize** (arma::mat &weights, arma::vec &biases, const size_t numFeatures, const size_t num←
Classes)

39.400.1 Detailed Description

This class is used to initialize weights for the weightVectors matrix in a random manner.

Definition at line 24 of file random_init.hpp.

39.400.2 Constructor & Destructor Documentation

39.400.2.1 RandomInitialization()

```
RandomInitialization ( ) [inline]
```

Definition at line 27 of file random_init.hpp.

39.400.3 Member Function Documentation

39.400.3.1 Initialize()

```
static void Initialize (
    arma::mat & weights,
    arma::vec & biases,
    const size_t numFeatures,
    const size_t numClasses ) [inline], [static]
```

Definition at line 29 of file random_init.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/initialization_methods/ **random_init.hpp**

39.401 SimpleWeightUpdate Class Reference

Public Member Functions

- template<typename VecType >
void **UpdateWeights** (const VecType &trainingPoint, arma::mat &weights, arma::vec &biases, const size_t incorrectClass, const size_t correctClass, const double instanceWeight=1.0)

This function is called to update the weightVectors matrix.

39.401.1 Detailed Description

Definition at line 30 of file `simple_weight_update.hpp`.

39.401.2 Member Function Documentation

39.401.2.1 UpdateWeights()

```
void UpdateWeights (
    const VecType & trainingPoint,
    arma::mat & weights,
    arma::vec & biases,
    const size_t incorrectClass,
    const size_t correctClass,
    const double instanceWeight = 1.0 ) [inline]
```

This function is called to update the `weightVectors` matrix.

It decreases the weights of the incorrectly classified class while increasing the weight of the correct class it should have been classified to.

Template Parameters

<i>Type</i>	of vector (should be an Armadillo vector like <code>arma::vec</code> or <code>arma::sp_vec</code> or something similar).
-------------	--

Parameters

<i>trainingPoint</i>	Point that was misclassified.
<i>weights</i>	Matrix of weights.
<i>biases</i>	Vector of biases.
<i>incorrectClass</i>	Index of class that the point was incorrectly classified as.
<i>correctClass</i>	Index of the true class of the point.
<i>instanceWeight</i>	Weight to be given to this particular point during training (this is useful for boosting).

Definition at line 50 of file `simple_weight_update.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/learning_policies/simple_weight_update.hpp` ↩

39.402 ZeroInitialization Class Reference

This class is used to initialize the matrix `weightVectors` to zero.

Public Member Functions

- **ZeroInitialization** ()

Static Public Member Functions

- static void **Initialize** (arma::mat &weights, arma::vec &biases, const size_t numFeatures, const size_t numClasses)

39.402.1 Detailed Description

This class is used to initialize the matrix weightVectors to zero.

Definition at line 23 of file zero_init.hpp.

39.402.2 Constructor & Destructor Documentation

39.402.2.1 ZeroInitialization()

```
ZeroInitialization ( ) [inline]
```

Definition at line 26 of file zero_init.hpp.

39.402.3 Member Function Documentation

39.402.3.1 Initialize()

```
static void Initialize (
    arma::mat & weights,
    arma::vec & biases,
    const size_t numFeatures,
    const size_t numClasses ) [inline], [static]
```

Definition at line 28 of file zero_init.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/initialization_methods/ **zero_init.hpp**

39.403 Radical Class Reference

An implementation of RADICAL, an algorithm for independent component analysis (ICA).

Public Member Functions

- **Radical** (const double noiseStdDev=0.175, const size_t replicates=30, const size_t angles=150, const size_t sweeps=0, const size_t m=0)
Set the parameters to RADICAL.
- size_t **Angles** () const
Get the number of angles considered during brute-force search.
- size_t & **Angles** ()
Modify the number of angles considered during brute-force search.
- void **CopyAndPerturb** (arma::mat &xNew, const arma::mat &x) const
*Make replicates of each data point (the number of replicates is set in either the constructor or with **Replicates()** (p. 1747)) and perturb data with Gaussian noise with standard deviation noiseStdDev.*
- void **DoRadical** (const arma::mat &matX, arma::mat &matY, arma::mat &matW)
Run RADICAL.
- double **DoRadical2D** (const arma::mat &matX)
Two-dimensional version of RADICAL.
- double **NoiseStdDev** () const
Get the standard deviation of the additive Gaussian noise.
- double & **NoiseStdDev** ()
Modify the standard deviation of the additive Gaussian noise.
- size_t **Replicates** () const
Get the number of Gaussian-perturbed replicates used per point.
- size_t & **Replicates** ()
Modify the number of Gaussian-perturbed replicates used per point.
- size_t **Sweeps** () const
Get the number of sweeps.
- size_t & **Sweeps** ()
Modify the number of sweeps.
- double **Vasicek** (arma::vec &x) const
Vasicek's m-spacing estimator of entropy, with overlap modification from (Learned-Miller and Fisher, 2003).

39.403.1 Detailed Description

An implementation of RADICAL, an algorithm for independent component analysis (ICA).

Let X be a matrix where each column is a point and each row a dimension. The goal is to find a square unmixing matrix W such that $Y = W X$ and the rows of Y are independent components.

For more details, see the following paper:

```
@article{learned2003ica,
  title = {ICA Using Spacings Estimates of Entropy},
  author = {Learned-Miller, E.G. and Fisher III, J.W.},
  journal = {Journal of Machine Learning Research},
  volume = {4},
  pages = {1271--1295},
  year = {2003}
}
```

Definition at line 43 of file radical.hpp.

39.403.2 Constructor & Destructor Documentation

39.403.2.1 Radical()

```
Radical (
    const double noiseStdDev = 0.175,
    const size_t replicates = 30,
    const size_t angles = 150,
    const size_t sweeps = 0,
    const size_t m = 0 )
```

Set the parameters to RADICAL.

Parameters

<i>noiseStdDev</i>	Standard deviation of the Gaussian noise added to the replicates of the data points during Radical2D
<i>replicates</i>	Number of Gaussian-perturbed replicates to use (per point) in Radical2D
<i>angles</i>	Number of angles to consider in brute-force search during Radical2D
<i>sweeps</i>	Number of sweeps. Each sweep calls Radical2D once for each pair of dimensions
<i>m</i>	The variable m from Vasicek's m-spacing estimator of entropy.

39.403.3 Member Function Documentation

39.403.3.1 Angles() [1/2]

```
size_t Angles ( ) const [inline]
```

Get the number of angles considered during brute-force search.

Definition at line 105 of file radical.hpp.

39.403.3.2 Angles() [2/2]

```
size_t& Angles ( ) [inline]
```

Modify the number of angles considered during brute-force search.

Definition at line 107 of file radical.hpp.

39.403.3.3 CopyAndPerturb()

```
void CopyAndPerturb (
    arma::mat & xNew,
    const arma::mat & x ) const
```

Make replicates of each data point (the number of replicates is set in either the constructor or with **Replicates()** (p. 1747)) and perturb data with Gaussian noise with standard deviation noiseStdDev.

39.403.3.4 DoRadical()

```
void DoRadical (
    const arma::mat & matX,
    arma::mat & matY,
    arma::mat & matW )
```

Run RADICAL.

Parameters

<i>matX</i>	Input data into the algorithm - a matrix where each column is a point and each row is a dimension.
<i>matY</i>	Estimated independent components - a matrix where each column is a point and each row is an estimated independent component.
<i>matW</i>	Estimated unmixing matrix, where $\text{matY} = \text{matW} * \text{matX}$.

39.403.3.5 DoRadical2D()

```
double DoRadical2D (
    const arma::mat & matX )
```

Two-dimensional version of RADICAL.

39.403.3.6 NoiseStdDev() [1/2]

```
double NoiseStdDev ( ) const [inline]
```

Get the standard deviation of the additive Gaussian noise.

Definition at line 95 of file radical.hpp.

39.403.3.7 NoiseStdDev() [2/2]

```
double& NoiseStdDev ( ) [inline]
```

Modify the standard deviation of the additive Gaussian noise.

Definition at line 97 of file radical.hpp.

39.403.3.8 Replicates() [1/2]

```
size_t Replicates ( ) const [inline]
```

Get the number of Gaussian-perturbed replicates used per point.

Definition at line 100 of file radical.hpp.

39.403.3.9 Replicates() [2/2]

```
size_t& Replicates ( ) [inline]
```

Modify the number of Gaussian-perturbed replicates used per point.

Definition at line 102 of file radical.hpp.

39.403.3.10 Sweeps() [1/2]

```
size_t Sweeps ( ) const [inline]
```

Get the number of sweeps.

Definition at line 110 of file radical.hpp.

39.403.3.11 Sweeps() [2/2]

```
size_t& Sweeps ( ) [inline]
```

Modify the number of sweeps.

Definition at line 112 of file radical.hpp.

References `mlpack::radical::WhitenFeatureMajorMatrix()`.

39.403.3.12 Vasicek()

```
double Vasicek (
    arma::vec & x ) const
```

Vasicek's m-spacing estimator of entropy, with overlap modification from (Learned-Miller and Fisher, 2003).

Parameters

x	Empirical sample (one-dimensional) over which to estimate entropy.
---	--

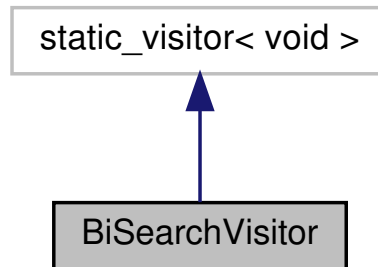
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/radical/ **radical.hpp**

39.404 BiSearchVisitor Class Reference

BiSearchVisitor (p. 1748) executes a bichromatic range search on the given RSType.

Inheritance diagram for BiSearchVisitor:



Public Types

- `template<template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType> using RSTypeT = RSType< TreeType >`
Alias template necessary for visual c++ compiler.

Public Member Functions

- **BiSearchVisitor** (const arma::mat &querySet, const **math::Range** &range, std::vector< std::vector< size_t >> &neighbors, std::vector< std::vector< double >> &distances, const size_t leafSize)
*Construct the **BiSearchVisitor** (p. 1748).*
- `template<template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType> void operator() (RSTypeT< TreeType > *rs) const`
Default Bichromatic range search on the given RSType instance.
- `void operator() (RSTypeT< tree::KDTree > *rs) const`
Bichromatic range search on the given RSType specialized for KDTrees.
- `void operator() (RSTypeT< tree::BallTree > *rs) const`
Bichromatic range search on the given RSType specialized for BallTrees.
- `void operator() (RSTypeT< tree::Octree > *rs) const`
Bichromatic range search specialized for octrees.

39.404.1 Detailed Description

BiSearchVisitor (p. 1748) executes a bichromatic range search on the given RSType.

We use template specialization to differentiate those tree types that accept leafSize as a parameter. In these cases, before doing range search, a query tree with proper leafSize is built from the querySet.

Definition at line 71 of file rs_model.hpp.

39.404.2 Member Typedef Documentation

39.404.2.1 RSTypeT

```
using RSTypeT = RSType<TreeType>
```

Alias template necessary for visual c++ compiler.

Definition at line 94 of file rs_model.hpp.

39.404.3 Constructor & Destructor Documentation

39.404.3.1 BiSearchVisitor()

```
BiSearchVisitor (  
    const arma::mat & querySet,  
    const math::Range & range,  
    std::vector< std::vector< size_t >> & neighbors,  
    std::vector< std::vector< double >> & distances,  
    const size_t leafSize )
```

Construct the **BiSearchVisitor** (p. 1748).

39.404.4 Member Function Documentation

39.404.4.1 operator>() [1/4]

```
void operator() (
    RSTypeT< TreeType > * rs ) const
```

Default Bichromatic range search on the given RSType instance.

39.404.4.2 operator>() [2/4]

```
void operator() (
    RSTypeT< tree::KDTree > * rs ) const
```

Bichromatic range search on the given RSType specialized for KDTrees.

39.404.4.3 operator>() [3/4]

```
void operator() (
    RSTypeT< tree::BallTree > * rs ) const
```

Bichromatic range search on the given RSType specialized for BallTrees.

39.404.4.4 operator>() [4/4]

```
void operator() (
    RSTypeT< tree::Octree > * rs ) const
```

Bichromatic range search specialized for octrees.

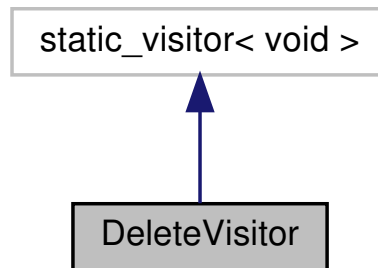
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/ **rs_model.hpp**

39.405 DeleteVisitor Class Reference

DeleteVisitor (p. 1751) deletes the given RSType instance.

Inheritance diagram for DeleteVisitor:



Public Member Functions

- `template<typename RSType >`
`void operator() (RSType *rs) const`
Delete the RSType object.

39.405.1 Detailed Description

DeleteVisitor (p. 1751) deletes the given RSType instance.

Definition at line 177 of file `rs_model.hpp`.

39.405.2 Member Function Documentation

39.405.2.1 `operator()()`

```
void operator() (  
    RSType * rs ) const
```

Delete the RSType object.

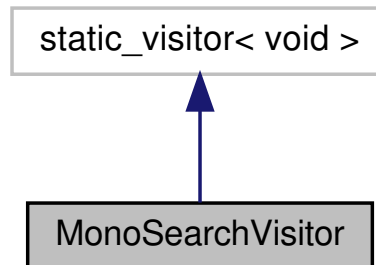
The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/ rs_model.hpp`

39.406 MonoSearchVisitor Class Reference

MonoSearchVisitor (p. 1752) executes a monochromatic range search on the given RSType.

Inheritance diagram for MonoSearchVisitor:



Public Member Functions

- **MonoSearchVisitor** (const **math::Range** &range, std::vector< std::vector< size_t >> &neighbors, std::vector< std::vector< double >> &distances)
Construct the **MonoSearchVisitor** (p. 1752) with the given parameters.
- template<typename RSType >
void **operator()** (**RSType** *rs) const
Perform monochromatic search with the given **RangeSearch** (p. 1754) object.

39.406.1 Detailed Description

MonoSearchVisitor (p. 1752) executes a monochromatic range search on the given RSType.

Range Search is performed on the reference set itself, no querySet.

Definition at line 40 of file rs_model.hpp.

39.406.2 Constructor & Destructor Documentation

39.406.2.1 MonoSearchVisitor()

```

MonoSearchVisitor (
    const math::Range & range,
    std::vector< std::vector< size_t >> & neighbors,
    std::vector< std::vector< double >> & distances ) [inline]

```

Construct the **MonoSearchVisitor** (p. 1752) with the given parameters.

Definition at line 56 of file rs_model.hpp.

39.406.3 Member Function Documentation

39.406.3.1 operator>()

```
void operator() (
    RSType * rs ) const
```

Perform monochromatic search with the given **RangeSearch** (p. 1754) object.

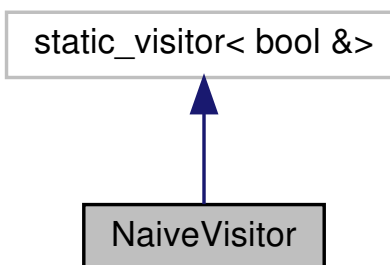
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/ **rs_model.hpp**

39.407 NaiveVisitor Class Reference

NaiveVisitor (p. 1753) exposes the Naive() method of the given RSType.

Inheritance diagram for NaiveVisitor:



Public Member Functions

- template<typename RSType >
bool & **operator()** (RSType *rs) const
*Get a reference to the naive parameter of the given **RangeSearch** (p. 1754) object.*

39.407.1 Detailed Description

NaiveVisitor (p. 1753) exposes the Naive() method of the given RSType.

Definition at line 202 of file rs_model.hpp.

39.407.2 Member Function Documentation

39.407.2.1 operator{ }()

```
bool& operator() (
    RSType * rs ) const
```

Get a reference to the naive parameter of the given **RangeSearch** (p. 1754) object.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/ **rs_model.hpp**

39.408 RangeSearch< MetricType, MatType, TreeType > Class Template Reference

The **RangeSearch** (p. 1754) class is a template class for performing range searches.

Public Types

- typedef TreeType< MetricType, **RangeSearchStat**, MatType > **Tree**
Convenience typedef.

Public Member Functions

- **RangeSearch** (MatType referenceSet, const bool naive=false, const bool singleMode=false, const MetricType metric=MetricType())
*Initialize the **RangeSearch** (p. 1754) object with a given reference dataset (this is the dataset which is searched).*
- **RangeSearch** (**Tree** *referenceTree, const bool singleMode=false, const MetricType metric=MetricType())
*Initialize the **RangeSearch** (p. 1754) object with the given pre-constructed reference tree (this is the tree built on the reference set, which is the set that is searched).*
- **RangeSearch** (const bool naive=false, const bool singleMode=false, const MetricType metric=MetricType())
*Initialize the **RangeSearch** (p. 1754) object without any reference data.*
- **RangeSearch** (const **RangeSearch** &other)
*Construct the **RangeSearch** (p. 1754) model as a copy of the given model.*
- **RangeSearch** (**RangeSearch** &&other)
*Construct the **RangeSearch** (p. 1754) model by taking ownership of the given model.*
- **~RangeSearch** ()
*Destroy the **RangeSearch** (p. 1754) object.*
- size_t **BaseCases** () const
Get the number of base cases during the last search.
- bool **Naive** () const

- Get whether naive search is being used.*

 - `bool & Naive ()`

Modify whether naive search is being used.
- `RangeSearch & operator= (RangeSearch other)`

Copy the given RangeSearch (p. 1754) model.
- `const MatType & ReferenceSet () const`

Return the reference set.
- `Tree * ReferenceTree ()`

Return the reference tree (or NULL if in naive mode).
- `size_t Scores () const`

Get the number of scores during the last search.
- `void Search (const MatType &querySet, const math::Range &range, std::vector< std::vector< size_t >> &neighbors, std::vector< std::vector< double >> &distances)`

Search for all reference points in the given range for each point in the query set, returning the results in the neighbors and distances objects.
- `void Search (Tree *queryTree, const math::Range &range, std::vector< std::vector< size_t >> &neighbors, std::vector< std::vector< double >> &distances)`

Given a pre-built query tree, search for all reference points in the given range for each point in the query set, returning the results in the neighbors and distances objects.
- `void Search (const math::Range &range, std::vector< std::vector< size_t >> &neighbors, std::vector< std::vector< double >> &distances)`

Search for all points in the given range for each point in the reference set (which was passed to the constructor), returning the results in the neighbors and distances objects.
- `template<typename Archive > void serialize (Archive &ar, const unsigned int version)`

Serialize the model.
- `bool SingleMode () const`

Get whether single-tree search is being used.
- `bool & SingleMode ()`

Modify whether single-tree search is being used.
- `void Train (MatType referenceSet)`

Set the reference set to a new reference set, and build a tree if necessary.
- `void Train (Tree *referenceTree)`

Set the reference tree to a new reference tree.

39.408.1 Detailed Description

```
template<typename MetricType = metric::EuclideanDistance, typename MatType = arma::mat, template< typename TreeMetricType,
typename TreeStatType, typename TreeMatType > class TreeType = tree::KDTree>
class mlpack::range::RangeSearch< MetricType, MatType, TreeType >
```

The **RangeSearch** (p. 1754) class is a template class for performing range searches.

It is implemented in the style of a generalized tree-independent dual-tree algorithm; for more details on the actual algorithm, see the **RangeSearchRules** (p. 1764) class.

Template Parameters

<i>MetricType</i>	Metric to use for range search calculations.
<i>MatType</i>	Type of data to use.
<i>TreeType</i>	Type of tree to use; must satisfy the <i>TreeType</i> policy API.

Definition at line 42 of file `range_search.hpp`.

39.408.2 Member Typedef Documentation

39.408.2.1 Tree

```
typedef TreeType<MetricType, RangeSearchStat, MatType> Tree
```

Convenience typedef.

Definition at line 46 of file `range_search.hpp`.

39.408.3 Constructor & Destructor Documentation

39.408.3.1 RangeSearch() [1/5]

```
RangeSearch (
    MatType referenceSet,
    const bool naive = false,
    const bool singleMode = false,
    const MetricType metric = MetricType() )
```

Initialize the **RangeSearch** (p. 1754) object with a given reference dataset (this is the dataset which is searched).

Optionally, perform the computation in naive mode or single-tree mode. Additionally, an instantiated metric can be given, for cases where the distance metric holds data.

This method will move the matrices to internal copies, which are rearranged during tree-building. You can avoid creating an extra copy by pre-constructing the trees and passing them in using `std::move`.

Parameters

<i>referenceSet</i>	Reference dataset.
<i>naive</i>	Whether the computation should be done in $O(n^2)$ naive mode.
<i>singleMode</i>	Whether single-tree computation should be used (as opposed to dual-tree computation).
<i>metric</i>	Instantiated distance metric.

39.408.3.2 RangeSearch() [2/5]

```

RangeSearch (
    Tree * referenceTree,
    const bool singleMode = false,
    const MetricType metric = MetricType() )

```

Initialize the **RangeSearch** (p. 1754) object with the given pre-constructed reference tree (this is the tree built on the reference set, which is the set that is searched).

Optionally, choose to use single-tree mode, which will not build a tree on query points. Naive mode is not available as an option for this constructor. Additionally, an instantiated distance metric can be given, for cases where the distance metric holds data.

There is no copying of the data matrices in this constructor (because tree-building is not necessary), so this is the constructor to use when copies absolutely must be avoided.

Note

Because tree-building (at least with BinarySpaceTree) modifies the ordering of a matrix, be aware that mapping of the points back to their original indices is not done when this constructor is used.

Parameters

<i>referenceTree</i>	Pre-built tree for reference points.
<i>referenceSet</i>	Set of reference points corresponding to referenceTree.
<i>singleMode</i>	Whether single-tree computation should be used (as opposed to dual-tree computation).
<i>metric</i>	Instantiated distance metric.

39.408.3.3 RangeSearch() [3/5]

```

RangeSearch (
    const bool naive = false,
    const bool singleMode = false,
    const MetricType metric = MetricType() )

```

Initialize the **RangeSearch** (p. 1754) object without any reference data.

If the monochromatic **Search()** (p. 1760) is called before a reference set is set with **Train()** (p. 1763), no results will be returned (since the reference set is empty).

Parameters

<i>naive</i>	Whether to use naive search.
<i>singleMode</i>	Whether single-tree computation should be used (as opposed to dual-tree computation).
<i>metric</i>	Instantiated metric.

39.408.3.4 **RangeSearch()** [4/5]

```
RangeSearch (
    const RangeSearch< MetricType, MatType, TreeType > & other )
```

Construct the **RangeSearch** (p. 1754) model as a copy of the given model.

Note that this may be computationally intensive!

Parameters

<i>other</i>	RangeSearch (p. 1754) model to copy.
--------------	---

39.408.3.5 **RangeSearch()** [5/5]

```
RangeSearch (
    RangeSearch< MetricType, MatType, TreeType > && other )
```

Construct the **RangeSearch** (p. 1754) model by taking ownership of the given model.

Parameters

<i>other</i>	RangeSearch (p. 1754) model to take ownership of.
--------------	--

39.408.3.6 **~RangeSearch()**

```
~ RangeSearch ( )
```

Destroy the **RangeSearch** (p. 1754) object.

If trees were created, they will be deleted.

39.408.4 **Member Function Documentation**

39.408.4.1 BaseCases()

```
size_t BaseCases ( ) const [inline]
```

Get the number of base cases during the last search.

Definition at line 276 of file range_search.hpp.

39.408.4.2 Naive() [1/2]

```
bool Naive ( ) const [inline]
```

Get whether naive search is being used.

Definition at line 271 of file range_search.hpp.

39.408.4.3 Naive() [2/2]

```
bool& Naive ( ) [inline]
```

Modify whether naive search is being used.

Definition at line 273 of file range_search.hpp.

39.408.4.4 operator=()

```
RangeSearch& operator= (
    RangeSearch< MetricType, MatType, TreeType > other )
```

Copy the given **RangeSearch** (p. 1754) model.

Use std::move to pass in the model if the old copy is no longer needed.

Parameters

<i>other</i>	RangeSearch (p. 1754) model to copy.
--------------	---

39.408.4.5 ReferenceSet()

```
const MatType& ReferenceSet ( ) const [inline]
```

Return the reference set.

Definition at line 285 of file range_search.hpp.

39.408.4.6 ReferenceTree()

```
Tree* ReferenceTree ( ) [inline]
```

Return the reference tree (or NULL if in naive mode).

Definition at line 288 of file range_search.hpp.

39.408.4.7 Scores()

```
size_t Scores ( ) const [inline]
```

Get the number of scores during the last search.

Definition at line 278 of file range_search.hpp.

References RangeSearch< MetricType, MatType, TreeType >::serialize().

39.408.4.8 Search() [1/3]

```
void Search (
    const MatType & querySet,
    const math::Range & range,
    std::vector< std::vector< size_t >> & neighbors,
    std::vector< std::vector< double >> & distances )
```

Search for all reference points in the given range for each point in the query set, returning the results in the neighbors and distances objects.

Each entry in the external vector corresponds to a query point. Each of these entries holds a vector which contains the indices and distances of the reference points falling into the given range.

That is:

- neighbors.size() and distances.size() both equal the number of query points.
- neighbors[i] contains the indices of all the points in the reference set which have distances inside the given range to query point i.
- distances[i] contains all of the distances corresponding to the indices contained in neighbors[i].
- neighbors[i] and distances[i] are not sorted in any particular order.

Parameters

<i>querySet</i>	Set of query points to search with.
<i>range</i>	Range of distances in which to search.
<i>neighbors</i>	Object which will hold the list of neighbors for each point which fell into the given range, for each query point.
<i>distances</i>	Object which will hold the list of distances for each point which fell into the given range, for each query point.

39.408.4.9 Search() [2/3]

```
void Search (
    Tree * queryTree,
    const math::Range & range,
    std::vector< std::vector< size_t >> & neighbors,
    std::vector< std::vector< double >> & distances )
```

Given a pre-built query tree, search for all reference points in the given range for each point in the query set, returning the results in the neighbors and distances objects.

Each entry in the external vector corresponds to a query point. Each of these entries holds a vector which contains the indices and distances of the reference points falling into the given range.

That is:

- neighbors.size() and distances.size() both equal the number of query points.
- neighbors[i] contains the indices of all the points in the reference set which have distances inside the given range to query point i.
- distances[i] contains all of the distances corresponding to the indices contained in neighbors[i].
- neighbors[i] and distances[i] are not sorted in any particular order.

If either naive or singleMode are set to true, this will throw an invalid_argument exception; passing in a query tree implies dual-tree search.

If you want to use the reference tree as the query tree, instead call the overload of **Search()** (p. 1760) that does not take a query set.

Parameters

<i>queryTree</i>	Tree built on query points.
<i>range</i>	Range of distances in which to search.
<i>neighbors</i>	Object which will hold the list of neighbors for each point which fell into the given range, for each query point.
<i>distances</i>	Object which will hold the list of distances for each point which fell into the given range, for each query point.

39.408.4.10 Search() [3/3]

```
void Search (
    const math::Range & range,
    std::vector< std::vector< size_t >> & neighbors,
    std::vector< std::vector< double >> & distances )
```

Search for all points in the given range for each point in the reference set (which was passed to the constructor), returning the results in the neighbors and distances objects.

This means that the query set and the reference set are the same.

Each entry in the external vector corresponds to a query point. Each of these entries holds a vector which contains the indices and distances of the reference points falling into the given range.

That is:

- neighbors.size() and distances.size() both equal the number of query points.
- neighbors[i] contains the indices of all the points in the reference set which have distances inside the given range to query point i.
- distances[i] contains all of the distances corresponding to the indices contained in neighbors[i].
- neighbors[i] and distances[i] are not sorted in any particular order.

Parameters

<i>queryTree</i>	Tree built on query points.
<i>range</i>	Range of distances in which to search.
<i>neighbors</i>	Object which will hold the list of neighbors for each point which fell into the given range, for each query point.
<i>distances</i>	Object which will hold the list of distances for each point which fell into the given range, for each query point.

39.408.4.11 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int version )
```

Serialize the model.

Referenced by RangeSearch< MetricType, MatType, TreeType >::Scores().

39.408.4.12 SingleMode() [1/2]

```
bool SingleMode ( ) const [inline]
```

Get whether single-tree search is being used.

Definition at line 266 of file range_search.hpp.

39.408.4.13 SingleMode() [2/2]

```
bool& SingleMode ( ) [inline]
```

Modify whether single-tree search is being used.

Definition at line 268 of file range_search.hpp.

39.408.4.14 Train() [1/2]

```
void Train (
    MatType referenceSet )
```

Set the reference set to a new reference set, and build a tree if necessary.

This method is called '**Train()** (p. 1763)' in order to match the rest of the mlpack abstractions, even though calling this "training" is maybe a bit of a stretch.

Use std::move to pass in the reference set if the old copy is no longer needed.

Parameters

<i>referenceSet</i>	New set of reference data.
---------------------	----------------------------

39.408.4.15 Train() [2/2]

```
void Train (
    Tree * referenceTree )
```

Set the reference tree to a new reference tree.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/ **range_search.hpp**

39.409 RangeSearchRules< MetricType, TreeType > Class Template Reference

The **RangeSearchRules** (p. 1764) class is a template helper class used by **RangeSearch** (p. 1754) class when performing range searches.

Public Types

- typedef **tree::TraversallInfo**< TreeType > **TraversallInfoType**

Public Member Functions

- **RangeSearchRules** (const arma::mat &referenceSet, const arma::mat &querySet, const **math::Range** &range, std::vector< std::vector< size_t > > &neighbors, std::vector< std::vector< double > > &distances, MetricType &metric, const bool sameSet=false)
*Construct the **RangeSearchRules** (p. 1764) object.*
- double **BaseCase** (const size_t queryIndex, const size_t referenceIndex)
Compute the base case between the given query point and reference point.
- size_t **BaseCases** () const
Get the number of base cases.
- double **Rescore** (const size_t queryIndex, TreeType &referenceNode, const double oldScore) const
Re-evaluate the score for recursion order.
- double **Rescore** (TreeType &queryNode, TreeType &referenceNode, const double oldScore) const
Re-evaluate the score for recursion order.
- double **Score** (const size_t queryIndex, TreeType &referenceNode)
Get the score for recursion order.
- double **Score** (TreeType &queryNode, TreeType &referenceNode)
Get the score for recursion order.
- size_t **Scores** () const
Get the number of scores (that is, calls to RangeDistance()).
- const **TraversallInfoType** & **TraversallInfo** () const
- **TraversallInfoType** & **TraversallInfo** ()

39.409.1 Detailed Description

```
template<typename MetricType, typename TreeType>
class mpack::range::RangeSearchRules< MetricType, TreeType >
```

The **RangeSearchRules** (p. 1764) class is a template helper class used by **RangeSearch** (p. 1754) class when performing range searches.

Template Parameters

<i>MetricType</i>	The metric to use for computation.
<i>TreeType</i>	The tree type to use; must adhere to the TreeType API.

Definition at line 28 of file range_search_rules.hpp.

39.409.2 Member Typedef Documentation

39.409.2.1 TraversalInfoType

```
typedef tree::TraversalInfo<TreeType> TraversalInfoType
```

Definition at line 110 of file range_search_rules.hpp.

39.409.3 Constructor & Destructor Documentation

39.409.3.1 RangeSearchRules()

```
RangeSearchRules (
    const arma::mat & referenceSet,
    const arma::mat & querySet,
    const math::Range & range,
    std::vector< std::vector< size_t > > & neighbors,
    std::vector< std::vector< double > > & distances,
    MetricType & metric,
    const bool sameSet = false )
```

Construct the **RangeSearchRules** (p. 1764) object.

This is usually done from within the **RangeSearch** (p. 1754) class at search time.

Parameters

<i>referenceSet</i>	Set of reference data.
<i>querySet</i>	Set of query data.
<i>range</i>	Range to search for.
<i>neighbors</i>	Vector to store resulting neighbors in.
<i>distances</i>	Vector to store resulting distances in.
<i>metric</i>	Instantiated metric.
<i>sameSet</i>	If true, the query and reference set are taken to be the same, and a query point will not return itself in the results.

39.409.4 Member Function Documentation

39.409.4.1 BaseCase()

```
double BaseCase (
    const size_t queryIndex,
    const size_t referenceIndex )
```

Compute the base case between the given query point and reference point.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceIndex</i>	Index of reference point.

39.409.4.2 BaseCases()

```
size_t BaseCases ( ) const [inline]
```

Get the number of base cases.

Definition at line 116 of file range_search_rules.hpp.

39.409.4.3 Rescore() [1/2]

```
double Rescore (
    const size_t queryIndex,
    TreeType & referenceNode,
    const double oldScore ) const
```

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.
<i>oldScore</i>	Old score produced by Score() (p. 1767) (or Rescore() (p. 1766)).

39.409.4.4 Rescore() [2/2]

```
double Rescore (
    TreeType & queryNode,
    TreeType & referenceNode,
    const double oldScore ) const
```

Re-evaluate the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned). This is used when the score has already been calculated, but another recursion may have modified the bounds for pruning. So the old score is checked against the new pruning bound.

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.
<i>oldScore</i>	Old score produced by Score() (p. 1767) (or Rescore() (p. 1766)).

39.409.4.5 Score() [1/2]

```
double Score (
    const size_t queryIndex,
    TreeType & referenceNode )
```

Get the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Candidate node to be recursed into.

39.409.4.6 Score() [2/2]

```
double Score (
    TreeType & queryNode,
    TreeType & referenceNode )
```

Get the score for recursion order.

A low score indicates priority for recursion, while DBL_MAX indicates that the node should not be recursed into at all (it should be pruned).

Parameters

<i>queryNode</i>	Candidate query node to recurse into.
<i>referenceNode</i>	Candidate reference node to recurse into.

39.409.4.7 Scores()

```
size_t Scores ( ) const [inline]
```

Get the number of scores (that is, calls to RangeDistance()).

Definition at line 118 of file range_search_rules.hpp.

39.409.4.8 TraversalInfo() [1/2]

```
const TraversalInfoType& TraversalInfo ( ) const [inline]
```

Definition at line 112 of file range_search_rules.hpp.

39.409.4.9 TraversalInfo() [2/2]

```
TraversalInfoType& TraversalInfo ( ) [inline]
```

Definition at line 113 of file range_search_rules.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/ **range_search_rules.hpp**

39.410 RangeSearchStat Class Reference

Statistic class for **RangeSearch** (p. 1754), to be set to the `StatisticType` of the tree type that range search is being performed with.

Public Member Functions

- **RangeSearchStat** ()
Initialize the statistic.
- template<typename TreeType >
RangeSearchStat (TreeType &)
Initialize the statistic given a tree node that this statistic belongs to.
- double **LastDistance** () const
Get the last distance evaluation.
- double & **LastDistance** ()
Modify the last distance evaluation.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the statistic.

39.410.1 Detailed Description

Statistic class for **RangeSearch** (p. 1754), to be set to the StatisticType of the tree type that range search is being performed with.

This class just holds the last visited node and the corresponding base case result.

Definition at line 26 of file range_search_stat.hpp.

39.410.2 Constructor & Destructor Documentation

39.410.2.1 RangeSearchStat() [1/2]

```
RangeSearchStat ( ) [inline]
```

Initialize the statistic.

Definition at line 32 of file range_search_stat.hpp.

39.410.2.2 RangeSearchStat() [2/2]

```
RangeSearchStat (  
    TreeType & ) [inline]
```

Initialize the statistic given a tree node that this statistic belongs to.

In this case, we ignore the node.

Definition at line 39 of file range_search_stat.hpp.

39.410.3 Member Function Documentation

39.410.3.1 LastDistance() [1/2]

```
double LastDistance ( ) const [inline]
```

Get the last distance evaluation.

Definition at line 43 of file range_search_stat.hpp.

39.410.3.2 LastDistance() [2/2]

```
double& LastDistance ( ) [inline]
```

Modify the last distance evaluation.

Definition at line 45 of file range_search_stat.hpp.

39.410.3.3 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the statistic.

Definition at line 49 of file range_search_stat.hpp.

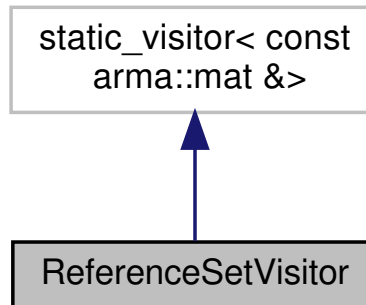
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/ **range_search_stat.hpp**

39.411 ReferenceSetVisitor Class Reference

ReferenceSetVisitor (p. 1771) exposes the referenceSet of the given RSType.

Inheritance diagram for ReferenceSetVisitor:



Public Member Functions

- `template<typename RSType >`
`const arma::mat & operator() (RSType *rs) const`
Return the reference set.

39.411.1 Detailed Description

ReferenceSetVisitor (p. 1771) exposes the referenceSet of the given RSType.

Definition at line 166 of file `rs_model.hpp`.

39.411.2 Member Function Documentation

39.411.2.1 operator()()

```
const arma::mat& operator() (
    RSType * rs ) const
```

Return the reference set.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/ rs_model.hpp`

39.412 RSMModel Class Reference

Public Types

- enum **TreeTypes** {
KD_TREE,
COVER_TREE,
R_TREE,
R_STAR_TREE,
BALL_TREE,
X_TREE,
HILBERT_R_TREE,
R_PLUS_TREE,
R_PLUS_PLUS_TREE,
VP_TREE,
RP_TREE,
MAX_RP_TREE,
UB_TREE,
OCTREE }

Public Member Functions

- **RSMModel** (const **TreeTypes** treeType=TreeTypes::KD_TREE, const bool randomBasis=false)
*Initialize the **RSMModel** (p. 1772) with the given type and whether or not a random basis should be used.*
- **RSMModel** (const **RSMModel** &other)
*Copy the given **RSMModel** (p. 1772).*
- **RSMModel** (**RSMModel** &&other)
*Take ownership of the given **RSMModel** (p. 1772).*
- **~RSMModel** ()
Clean memory, if necessary.
- void **BuildModel** (arma::mat &&referenceSet, const size_t leafSize, const bool naive, const bool singleMode)
Build the reference tree on the given dataset with the given parameters.
- const arma::mat & **Dataset** () const
Expose the dataset.
- size_t **LeafSize** () const
Get the leaf size (applicable to everything but the cover tree).
- size_t & **LeafSize** ()
Modify the leaf size (applicable to everything but the cover tree).
- bool **Naive** () const
Get whether the model is in naive search mode.
- bool & **Naive** ()
Modify whether the model is in naive search mode.
- **RSMModel** & **operator=** (**RSMModel** other)
*Copy the given **RSMModel** (p. 1772).*
- bool **RandomBasis** () const
Get whether a random basis is used.
- bool & **RandomBasis** ()
Modify whether a random basis is used (don't do this after the model has been built).

- void **Search** (arma::mat &&querySet, const **math::Range** &range, std::vector< std::vector< size_t >> &neighbors, std::vector< std::vector< double >> &distances)
Perform range search.
- void **Search** (const **math::Range** &range, std::vector< std::vector< size_t >> &neighbors, std::vector< std::vector< double >> &distances)
Perform monochromatic range search, with the reference set as the query set.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the range search model.
- bool **SingleMode** () const
Get whether the model is in single-tree search mode.
- bool & **SingleMode** ()
Modify whether the model is in single-tree search mode.
- **TreeTypes TreeType** () const
Get the type of tree.
- **TreeTypes & TreeType** ()
Modify the type of tree (don't do this after the model has been built).

39.412.1 Detailed Description

Definition at line 212 of file rs_model.hpp.

39.412.2 Member Enumeration Documentation

39.412.2.1 TreeTypes

enum **TreeTypes**

Enumerator

KD_TREE	
COVER_TREE	
R_TREE	
R_STAR_TREE	
BALL_TREE	
X_TREE	
HILBERT_R_TREE	
R_PLUS_TREE	
R_PLUS_PLUS_TREE	
VP_TREE	
RP_TREE	
MAX_RP_TREE	
UB_TREE	
OCTREE	

Definition at line 215 of file rs_model.hpp.

39.412.3 Constructor & Destructor Documentation

39.412.3.1 `RModel()` [1/3]

```
RModel (
    const TreeTypes treeType = TreeTypes::KD_TREE,
    const bool randomBasis = false )
```

Initialize the **RModel** (p. 1772) with the given type and whether or not a random basis should be used.

Parameters

<i>treeType</i>	Type of tree to use.
<i>randomBasis</i>	Whether or not to use a random basis.

39.412.3.2 `RModel()` [2/3]

```
RModel (
    const RModel & other )
```

Copy the given **RModel** (p. 1772).

Parameters

<i>other</i>	RModel (p. 1772) to copy.
--------------	----------------------------------

39.412.3.3 `RModel()` [3/3]

```
RModel (
    RModel && other )
```

Take ownership of the given **RModel** (p. 1772).

Parameters

<i>other</i>	RModel (p. 1772) to take ownership of.
--------------	---

39.412.3.4 ~RSModel()

`~ RSModel ()`

Clean memory, if necessary.

39.412.4 Member Function Documentation

39.412.4.1 BuildModel()

```
void BuildModel (
    arma::mat && referenceSet,
    const size_t leafSize,
    const bool naive,
    const bool singleMode )
```

Build the reference tree on the given dataset with the given parameters.

This takes possession of the reference set to avoid a copy.

Parameters

<i>referenceSet</i>	Set of reference points.
<i>leafSize</i>	Leaf size of tree (ignored for the cover tree).
<i>naive</i>	Whether naive search should be used.
<i>singleMode</i>	Whether single-tree search should be used.

39.412.4.2 Dataset()

```
const arma::mat& Dataset ( ) const
```

Expose the dataset.

39.412.4.3 LeafSize() [1/2]

```
size_t LeafSize ( ) const [inline]
```

Get the leaf size (applicable to everything but the cover tree).

Definition at line 319 of file rs_model.hpp.

39.412.4.4 LeafSize() [2/2]

```
size_t& LeafSize ( ) [inline]
```

Modify the leaf size (applicable to everything but the cover tree).

Definition at line 321 of file rs_model.hpp.

39.412.4.5 Naive() [1/2]

```
bool Naive ( ) const
```

Get whether the model is in naive search mode.

39.412.4.6 Naive() [2/2]

```
bool& Naive ( )
```

Modify whether the model is in naive search mode.

39.412.4.7 operator=()

```
RModel& operator= (
    RModel other )
```

Copy the given **RModel** (p. 1772).

Use `std::move` to pass in the model if the old copy is no longer needed.

Parameters

<i>other</i>	RSModel (p. 1772) to copy.
--------------	-----------------------------------

39.412.4.8 RandomBasis() [1/2]

```
bool RandomBasis ( ) const [inline]
```

Get whether a random basis is used.

Definition at line 329 of file `rs_model.hpp`.

39.412.4.9 RandomBasis() [2/2]

```
bool& RandomBasis ( ) [inline]
```

Modify whether a random basis is used (don't do this after the model has been built).

Definition at line 332 of file `rs_model.hpp`.

References `mlpack::bindings::tests::CleanMemory()`.

39.412.4.10 Search() [1/2]

```
void Search (
    arma::mat && querySet,
    const math::Range & range,
    std::vector< std::vector< size_t >> & neighbors,
    std::vector< std::vector< double >> & distances )
```

Perform range search.

This takes possession of the query set, so the query set will not be usable after the search. For more information on the output format, see **RangeSearch<>::Search()** (p. 1760).

Parameters

<i>querySet</i>	Set of query points.
<i>range</i>	Range to search for.
<i>neighbors</i>	Output: neighbors falling within the desired range.
<i>distances</i>	Output: distances of neighbors.

39.412.4.11 Search() [2/2]

```
void Search (
    const math::Range & range,
    std::vector< std::vector< size_t >> & neighbors,
    std::vector< std::vector< double >> & distances )
```

Perform monochromatic range search, with the reference set as the query set.

For more information on the output format, see **RangeSearch<>::Search()** (p. 1760).

Parameters

<i>range</i>	Range to search for.
<i>neighbors</i>	Output: neighbors falling within the desired range.
<i>distances</i>	Output: distances of neighbors.

39.412.4.12 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the range search model.

39.412.4.13 SingleMode() [1/2]

```
bool SingleMode ( ) const
```

Get whether the model is in single-tree search mode.

39.412.4.14 SingleMode() [2/2]

```
bool& SingleMode ( )
```

Modify whether the model is in single-tree search mode.

39.412.4.15 TreeType() [1/2]

```
TreeTypes TreeType ( ) const [inline]
```

Get the type of tree.

Definition at line 324 of file rs_model.hpp.

39.412.4.16 TreeType() [2/2]

```
TreeTypes& TreeType ( ) [inline]
```

Modify the type of tree (don't do this after the model has been built).

Definition at line 326 of file rs_model.hpp.

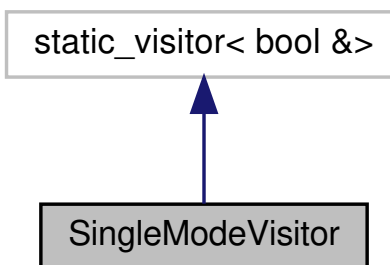
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/ **rs_model.hpp**

39.413 SingleModeVisitor Class Reference

SingleModeVisitor (p. 1779) exposes the SingleMode() method of the given RSType.

Inheritance diagram for SingleModeVisitor:



Public Member Functions

- template<typename RSType >
bool & **operator()** (**RSType** *rs) const

Get a reference to the singleMode parameter of the given RangeSeach object.

39.413.1 Detailed Description

SingleModeVisitor (p. 1779) exposes the SingleMode() method of the given RSType.

Definition at line 188 of file rs_model.hpp.

39.413.2 Member Function Documentation

39.413.2.1 operator()()

```
bool& operator() (
    RSType * rs ) const
```

Get a reference to the singleMode parameter of the given RangeSeach object.

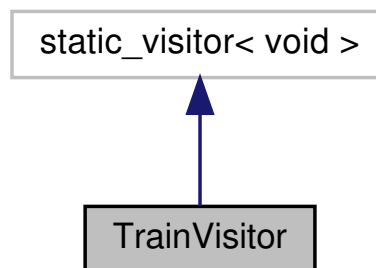
The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/ **rs_model.hpp**

39.414 TrainVisitor Class Reference

TrainVisitor (p. 1780) sets the reference set to a new reference set on the given RSType.

Inheritance diagram for TrainVisitor:



Public Types

- `template<template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType>`
`using RSTypeT = RSType< TreeType >`
Alias template necessary for visual c++ compiler.

Public Member Functions

- **TrainVisitor** (arma::mat &&referenceSet, const size_t leafSize)
Construct the **TrainVisitor** (p. 1780) object with the given reference set, leafSize.
- template<template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType>
void **operator()** (**RSTypeT**< TreeType > *rs) const
Default Train on the given RSType instance.
- void **operator()** (**RSTypeT**< **tree::KDTree** > *rs) const
Train on the given RSType specialized for KDTrees.
- void **operator()** (**RSTypeT**< **tree::BallTree** > *rs) const
Train on the given RSType specialized for BallTrees.
- void **operator()** (**RSTypeT**< **tree::Octree** > *rs) const
Train specialized for octrees.

39.414.1 Detailed Description

TrainVisitor (p. 1780) sets the reference set to a new reference set on the given RSType.

We use template specialization to differentiate those tree types that accept leafSize as a parameter. In these cases, a reference tree with proper leafSize is built from the referenceSet.

Definition at line 125 of file rs_model.hpp.

39.414.2 Member Typedef Documentation

39.414.2.1 RSTypeT

```
using RSTypeT = RSType<TreeType>
```

Alias template necessary for visual c++ compiler.

Definition at line 141 of file rs_model.hpp.

39.414.3 Constructor & Destructor Documentation

39.414.3.1 TrainVisitor()

```
TrainVisitor (  
    arma::mat && referenceSet,  
    const size_t leafSize )
```

Construct the **TrainVisitor** (p. 1780) object with the given reference set, leafSize.

39.414.4 Member Function Documentation

39.414.4.1 `operator()()` [1/4]

```
void operator() (
    RSTypeT< TreeType > * rs ) const
```

Default Train on the given RSType instance.

39.414.4.2 `operator()()` [2/4]

```
void operator() (
    RSTypeT< tree::KDTree > * rs ) const
```

Train on the given RSType specialized for KDTrees.

39.414.4.3 `operator()()` [3/4]

```
void operator() (
    RSTypeT< tree::BallTree > * rs ) const
```

Train on the given RSType specialized for BallTrees.

39.414.4.4 `operator()()` [4/4]

```
void operator() (
    RSTypeT< tree::Octree > * rs ) const
```

Train specialized for octrees.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/ **rs_model.hpp**

39.415 LARS Class Reference

An implementation of **LARS** (p. 1783), a stage-wise homotopy-based algorithm for l_1 -regularized linear regression (LASSO) and l_1+l_2 regularized linear regression (Elastic Net).

Public Member Functions

- **LARS** (const bool useCholesky=false, const double lambda1=0.0, const double lambda2=0.0, const double tolerance=1e-16)

*Set the parameters to **LARS** (p. 1783).*

- **LARS** (const bool useCholesky, const arma::mat &gramMatrix, const double lambda1=0.0, const double lambda2=0.0, const double tolerance=1e-16)

*Set the parameters to **LARS** (p. 1783), and pass in a precalculated Gram matrix.*

- **LARS** (const arma::mat &data, const arma::rowvec &responses, const bool transposeData=true, const bool useCholesky=false, const double lambda1=0.0, const double lambda2=0.0, const double tolerance=1e-16)

*Set the parameters to **LARS** (p. 1783) and run training.*

- **LARS** (const arma::mat &data, const arma::rowvec &responses, const bool transposeData, const bool useCholesky, const arma::mat &gramMatrix, const double lambda1=0.0, const double lambda2=0.0, const double tolerance=1e-16)

*Set the parameters to **LARS** (p. 1783), pass in a precalculated Gram matrix, and run training.*

- const std::vector< size_t > & **ActiveSet** () const

Access the set of active dimensions.

- const arma::vec & **Beta** () const

Access the solution coefficients.

- const std::vector< arma::vec > & **BetaPath** () const

Access the set of coefficients after each iteration; the solution is the last element.

- const std::vector< double > & **LambdaPath** () const

Access the set of values for lambda1 after each iteration; the solution is the last element.

- const arma::mat & **MatUtriCholFactor** () const

Access the upper triangular cholesky factor.

- void **Predict** (const arma::mat &points, arma::rowvec &predictions, const bool rowMajor=false) const

*Predict y_i for each data point in the given data matrix using the currently-trained **LARS** (p. 1783) model.*

- template<typename Archive >

void **serialize** (Archive &ar, const unsigned int)

*Serialize the **LARS** (p. 1783) model.*

- double **Train** (const arma::mat &data, const arma::rowvec &responses, arma::vec &beta, const bool transposeData=true)

*Run **LARS** (p. 1783).*

- double **Train** (const arma::mat &data, const arma::rowvec &responses, const bool transposeData=true)

*Run **LARS** (p. 1783).*

39.415.1 Detailed Description

An implementation of **LARS** (p. 1783), a stage-wise homotopy-based algorithm for l1-regularized linear regression (LASSO) and l1+l2 regularized linear regression (Elastic Net).

Let X be a matrix where each row is a point and each column is a dimension and let y be a vector of responses.

The Elastic Net problem is to solve

$$\min_{\beta} 0.5 \|X\beta - y\|_2^2 + \lambda_1 \|\beta\|_1 + 0.5\lambda_2 \|\beta\|_2^2$$

where β is the vector of regression coefficients.

If $\lambda_1 > 0$ and $\lambda_2 = 0$, the problem is the LASSO. If $\lambda_1 > 0$ and $\lambda_2 > 0$, the problem is the elastic net. If $\lambda_1 = 0$ and $\lambda_2 > 0$, the problem is ridge regression. If $\lambda_1 = 0$ and $\lambda_2 = 0$, the problem is unregularized linear regression.

Note: This algorithm is not recommended for use (in terms of efficiency) when $\lambda_1 = 0$.

For more details, see the following papers:

```
@article{efron2004least,
  title={Least angle regression},
  author={Efron, B. and Hastie, T. and Johnstone, I. and Tibshirani, R.},
  journal={The Annals of statistics},
  volume={32},
  number={2},
  pages={407--499},
  year={2004},
  publisher={Institute of Mathematical Statistics}
}

@article{zou2005regularization,
  title={Regularization and variable selection via the elastic net},
  author={Zou, H. and Hastie, T.},
  journal={Journal of the Royal Statistical Society Series B},
  volume={67},
  number={2},
  pages={301--320},
  year={2005},
  publisher={Royal Statistical Society}
}
```

Definition at line 89 of file lars.hpp.

39.415.2 Constructor & Destructor Documentation

39.415.2.1 LARS() [1/4]

```
LARS (
    const bool useCholesky = false,
    const double lambda1 = 0.0,
    const double lambda2 = 0.0,
    const double tolerance = 1e-16 )
```

Set the parameters to **LARS** (p. 1783).

Both lambda1 and lambda2 default to 0.

Parameters

<i>useCholesky</i>	Whether or not to use Cholesky decomposition when solving linear system (as opposed to using the full Gram matrix).
<i>lambda1</i>	Regularization parameter for l1-norm penalty.
<i>lambda2</i>	Regularization parameter for l2-norm penalty.
<i>tolerance</i>	Run until the maximum correlation of elements in $(X^T y)$ is less than this.

39.415.2.2 LARS() [2/4]

```
LARS (
    const bool useCholesky,
    const arma::mat & gramMatrix,
    const double lambda1 = 0.0,
    const double lambda2 = 0.0,
    const double tolerance = 1e-16 )
```

Set the parameters to **LARS** (p. 1783), and pass in a precalculated Gram matrix.

Both lambda1 and lambda2 default to 0.

Parameters

<i>useCholesky</i>	Whether or not to use Cholesky decomposition when solving linear system (as opposed to using the full Gram matrix).
<i>gramMatrix</i>	Gram matrix.
<i>lambda1</i>	Regularization parameter for l1-norm penalty.
<i>lambda2</i>	Regularization parameter for l2-norm penalty.
<i>tolerance</i>	Run until the maximum correlation of elements in $(X^T y)$ is less than this.

39.415.2.3 LARS() [3/4]

```
LARS (
    const arma::mat & data,
    const arma::rowvec & responses,
    const bool transposeData = true,
    const bool useCholesky = false,
    const double lambda1 = 0.0,
    const double lambda2 = 0.0,
    const double tolerance = 1e-16 )
```

Set the parameters to **LARS** (p. 1783) and run training.

Both lambda1 and lambda2 are set by default to 0.

Parameters

<i>data</i>	Input data.
<i>responses</i>	A vector of targets.
<i>transposeData</i>	Should be true if the input data is column-major and false otherwise.
<i>useCholesky</i>	Whether or not to use Cholesky decomposition when solving linear system (as opposed to using the full Gram matrix).
<i>lambda1</i>	Regularization parameter for l1-norm penalty.
<i>lambda2</i>	Regularization parameter for l2-norm penalty.
<i>tolerance</i>	Run until the maximum correlation of elements in $(X^T y)$ is less than this.

39.415.2.4 LARS() [4/4]

```

LARS (
    const arma::mat & data,
    const arma::rowvec & responses,
    const bool transposeData,
    const bool useCholesky,
    const arma::mat & gramMatrix,
    const double lambda1 = 0.0,
    const double lambda2 = 0.0,
    const double tolerance = 1e-16 )

```

Set the parameters to **LARS** (p. 1783), pass in a precalculated Gram matrix, and run training.

Both lambda1 and lambda2 are set by default to 0.

Parameters

<i>data</i>	Input data.
<i>responses</i>	A vector of targets.
<i>transposeData</i>	Should be true if the input data is column-major and false otherwise.
<i>useCholesky</i>	Whether or not to use Cholesky decomposition when solving linear system (as opposed to using the full Gram matrix).
<i>gramMatrix</i>	Gram matrix.
<i>lambda1</i>	Regularization parameter for l1-norm penalty.
<i>lambda2</i>	Regularization parameter for l2-norm penalty.
<i>tolerance</i>	Run until the maximum correlation of elements in $(X^T y)$ is less than this.

39.415.3 Member Function Documentation

39.415.3.1 ActiveSet()

```
const std::vector<size_t>& ActiveSet ( ) const [inline]
```

Access the set of active dimensions.

Definition at line 225 of file lars.hpp.

39.415.3.2 Beta()

```
const arma::vec& Beta ( ) const [inline]
```

Access the solution coefficients.

Definition at line 232 of file lars.hpp.

39.415.3.3 BetaPath()

```
const std::vector<arma::vec>& BetaPath ( ) const [inline]
```

Access the set of coefficients after each iteration; the solution is the last element.

Definition at line 229 of file lars.hpp.

39.415.3.4 LambdaPath()

```
const std::vector<double>& LambdaPath ( ) const [inline]
```

Access the set of values for lambda1 after each iteration; the solution is the last element.

Definition at line 236 of file lars.hpp.

39.415.3.5 MatUtriCholFactor()

```
const arma::mat& MatUtriCholFactor ( ) const [inline]
```

Access the upper triangular cholesky factor.

Definition at line 239 of file lars.hpp.

References LARS::serialize().

39.415.3.6 Predict()

```
void Predict (
    const arma::mat & points,
    arma::rowvec & predictions,
    const bool rowMajor = false ) const
```

Predict y_i for each data point in the given data matrix using the currently-trained **LARS** (p. 1783) model.

Parameters

<i>points</i>	The data points to regress on.
<i>predictions</i>	y, which will contained calculated values on completion.
<i>rowMajor</i>	Should be true if the data points matrix is row-major and false otherwise.

39.415.3.7 `serialize()`

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the **LARS** (p. 1783) model.

Referenced by `LARS::MatUtriCholFactor()`.

39.415.3.8 `Train()` [1/2]

```
double Train (
    const arma::mat & data,
    const arma::rowvec & responses,
    arma::vec & beta,
    const bool transposeData = true )
```

Run **LARS** (p. 1783).

The input matrix (like all mpack matrices) should be column-major – each column is an observation and each row is a dimension. However, because **LARS** (p. 1783) is more efficient on a row-major matrix, this method will (internally) transpose the matrix. If this transposition is not necessary (i.e., you want to pass in a row-major matrix), pass 'false' for the `transposeData` parameter.

Parameters

<i>data</i>	Column-major input data (or row-major input data if <code>rowMajor = true</code>).
<i>responses</i>	A vector of targets.
<i>beta</i>	Vector to store the solution (the coefficients) in.
<i>transposeData</i>	Set to false if the data is row-major.

Returns

The final absolute maximum correlation.

39.415.3.9 Train() [2/2]

```
double Train (
    const arma::mat & data,
    const arma::rowvec & responses,
    const bool transposeData = true )
```

Run **LARS** (p. 1783).

The input matrix (like all mlpack matrices) should be column-major – each column is an observation and each row is a dimension. However, because **LARS** (p. 1783) is more efficient on a row-major matrix, this method will (internally) transpose the matrix. If this transposition is not necessary (i.e., you want to pass in a row-major matrix), pass 'false' for the transposeData parameter.

Parameters

<i>data</i>	Input data.
<i>responses</i>	A vector of targets.
<i>transposeData</i>	Should be true if the input data is column-major and false otherwise.

Returns

The final absolute maximum correlation.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lars/ **lars.hpp**

39.416 LinearRegression Class Reference

A simple linear regression algorithm using ordinary least squares.

Public Member Functions

- **LinearRegression** (const arma::mat &predictors, const arma::rowvec &responses, const double lambda=0, const bool intercept=true)
Creates the model.
- **LinearRegression** (const arma::mat &predictors, const arma::rowvec &responses, const arma::rowvec &weights, const double lambda=0, const bool intercept=true)
Creates the model with weighted learning.
- **LinearRegression** ()
Empty constructor.
- double **ComputeError** (const arma::mat &points, const arma::rowvec &responses) const
Calculate the L2 squared error on the given predictors and responses using this linear regression model.
- bool **Intercept** () const

- Return whether or not an intercept term is used in the model.*
- double **Lambda** () const
Return the Tikhonov regularization parameter for ridge regression.
- double & **Lambda** ()
Modify the Tikhonov regularization parameter for ridge regression.
- const arma::vec & **Parameters** () const
Return the parameters (the b vector).
- arma::vec & **Parameters** ()
Modify the parameters (the b vector).
- void **Predict** (const arma::mat &points, arma::rowvec &predictions) const
Calculate y_i for each data point in points.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the model.
- double **Train** (const arma::mat &predictors, const arma::rowvec &responses, const bool intercept=true)
*Train the **LinearRegression** (p. 1789) model on the given data.*
- double **Train** (const arma::mat &predictors, const arma::rowvec &responses, const arma::rowvec &weights, const bool intercept=true)
*Train the **LinearRegression** (p. 1789) model on the given data and weights.*

39.416.1 Detailed Description

A simple linear regression algorithm using ordinary least squares.

Optionally, this class can perform ridge regression, if the lambda parameter is set to a number greater than zero.

Definition at line 26 of file linear_regression.hpp.

39.416.2 Constructor & Destructor Documentation

39.416.2.1 LinearRegression() [1/3]

```
LinearRegression (
    const arma::mat & predictors,
    const arma::rowvec & responses,
    const double lambda = 0,
    const bool intercept = true )
```

Creates the model.

Parameters

<i>predictors</i>	X, matrix of data points.
<i>responses</i>	y, the measured data for each point in X.
<i>lambda</i>	Regularization constant for ridge regression.
<i>intercept</i>	Whether or not to include an intercept term.

39.416.2.2 LinearRegression() [2/3]

```

LinearRegression (
    const arma::mat & predictors,
    const arma::rowvec & responses,
    const arma::rowvec & weights,
    const double lambda = 0,
    const bool intercept = true )

```

Creates the model with weighted learning.

Parameters

<i>predictors</i>	X, matrix of data points.
<i>responses</i>	y, the measured data for each point in X.
<i>weights</i>	Observation weights (for boosting).
<i>lambda</i>	Regularization constant for ridge regression.
<i>intercept</i>	Whether or not to include an intercept term.

39.416.2.3 LinearRegression() [3/3]

```

LinearRegression ( ) [inline]

```

Empty constructor.

This gives a non-working model, so make sure **Train()** (p. 1794) is called (or make sure the model parameters are set) before calling **Predict()** (p. 1793)!

Definition at line 62 of file linear_regression.hpp.

References LinearRegression::ComputeError(), LinearRegression::Predict(), and LinearRegression::Train().

39.416.3 Member Function Documentation

39.416.3.1 ComputeError()

```
double ComputeError (
    const arma::mat & points,
    const arma::rowvec & responses ) const
```

Calculate the L2 squared error on the given predictors and responses using this linear regression model.

This calculation returns

$$(1/n) * \|y - XB\|_2^2$$

where y is the responses vector, X is the matrix of predictors, and B is the parameters of the trained linear regression model.

As this number decreases to 0, the linear regression fit is better.

Parameters

<i>points</i>	Matrix of predictors (X).
<i>responses</i>	Transposed vector of responses (y^T).

Referenced by `LinearRegression::LinearRegression()`, and `RegressionDistribution::RegressionDistribution()`.

39.416.3.2 Intercept()

```
bool Intercept ( ) const [inline]
```

Return whether or not an intercept term is used in the model.

Definition at line 137 of file `linear_regression.hpp`.

39.416.3.3 Lambda() [1/2]

```
double Lambda ( ) const [inline]
```

Return the Tikhonov regularization parameter for ridge regression.

Definition at line 132 of file `linear_regression.hpp`.

39.416.3.4 Lambda() [2/2]

```
double& Lambda ( ) [inline]
```

Modify the Tikhonov regularization parameter for ridge regression.

Definition at line 134 of file linear_regression.hpp.

39.416.3.5 Parameters() [1/2]

```
const arma::vec& Parameters ( ) const [inline]
```

Return the parameters (the b vector).

Definition at line 127 of file linear_regression.hpp.

Referenced by RegressionDistribution::Dimensionality(), and RegressionDistribution::Parameters().

39.416.3.6 Parameters() [2/2]

```
arma::vec& Parameters ( ) [inline]
```

Modify the parameters (the b vector).

Definition at line 129 of file linear_regression.hpp.

39.416.3.7 Predict()

```
void Predict (
    const arma::mat & points,
    arma::rowvec & predictions ) const
```

Calculate y_i for each data point in points.

Parameters

<i>points</i>	the data points to calculate with.
<i>predictions</i>	y, will contain calculated values on completion.

Referenced by LinearRegression::LinearRegression().

39.416.3.8 `serialize()`

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the model.

Definition at line 143 of file `linear_regression.hpp`.

39.416.3.9 `Train()` [1/2]

```
double Train (
    const arma::mat & predictors,
    const arma::rowvec & responses,
    const bool intercept = true )
```

Train the **LinearRegression** (p. 1789) model on the given data.

Careful! This will completely ignore and overwrite the existing model. This particular implementation does not have an incremental training algorithm. To set the regularization parameter `lambda`, call **Lambda()** (p. 1792) or set a different value in the constructor.

Parameters

<i>predictors</i>	X, the matrix of data points to train the model on.
<i>responses</i>	y, the responses to the data points.
<i>intercept</i>	Whether or not to fit an intercept term.

Returns

The least squares error after training.

Referenced by `LinearRegression::LinearRegression()`, and `RegressionDistribution::RegressionDistribution()`.

39.416.3.10 `Train()` [2/2]

```
double Train (
    const arma::mat & predictors,
    const arma::rowvec & responses,
```

```
const arma::rowvec & weights,
const bool intercept = true )
```

Train the **LinearRegression** (p. 1789) model on the given data and weights.

Careful! This will completely ignore and overwrite the existing model. This particular implementation does not have an incremental training algorithm. To set the regularization parameter lambda, call **Lambda()** (p. 1792) or set a different value in the constructor.

Parameters

<i>predictors</i>	X, the matrix of data points to train the model on.
<i>responses</i>	y, the responses to the data points.
<i>weights</i>	Observation weights (for boosting).
<i>intercept</i>	Whether or not to fit an intercept term.

Returns

The least squares error after training.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear_regression/ **linear_regression.hpp**

39.417 LogisticRegression< MatType > Class Template Reference

The **LogisticRegression** (p. 1795) class implements an L2-regularized logistic regression model, and supports training with multiple optimizers and classification.

Public Member Functions

- **LogisticRegression** (const MatType &predictors, const arma::Row< size_t > &responses, const double lambda=0)
Construct the **LogisticRegression** (p. 1795) class with the given labeled training data.
- **LogisticRegression** (const MatType &predictors, const arma::Row< size_t > &responses, const arma::rowvec &initialPoint, const double lambda=0)
Construct the **LogisticRegression** (p. 1795) class with the given labeled training data.
- **LogisticRegression** (const size_t dimensionality=0, const double lambda=0)
Construct the **LogisticRegression** (p. 1795) class without performing any training.
- template<typename OptimizerType >
LogisticRegression (const MatType &predictors, const arma::Row< size_t > &responses, OptimizerType &optimizer, const double lambda)
Construct the **LogisticRegression** (p. 1795) class with the given labeled training data.
- template<typename VecType >
size_t **Classify** (const VecType &point, const double decisionBoundary=0.5) const
Classify the given point.

- void **Classify** (const MatType &dataset, arma::Row< size_t > &labels, const double decisionBoundary=0.5) const
Classify the given points, returning the predicted labels for each point.
- void **Classify** (const MatType &dataset, arma::mat &probabilities) const
Classify the given points, returning class probabilities for each point.
- double **ComputeAccuracy** (const MatType &predictors, const arma::Row< size_t > &responses, const double decisionBoundary=0.5) const
Compute the accuracy of the model on the given predictors and responses, optionally using the given decision boundary.
- double **ComputeError** (const MatType &predictors, const arma::Row< size_t > &responses) const
Compute the error of the model.
- const double & **Lambda** () const
Return the lambda value for L2-regularization.
- double & **Lambda** ()
Modify the lambda value for L2-regularization.
- const arma::rowvec & **Parameters** () const
Return the parameters (the b vector).
- arma::rowvec & **Parameters** ()
Modify the parameters (the b vector).
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the model.
- template<typename OptimizerType = ens::L_BFGS>
double **Train** (const MatType &predictors, const arma::Row< size_t > &responses)
*Train the **LogisticRegression** (p. 1795) model on the given input data.*
- template<typename OptimizerType >
double **Train** (const MatType &predictors, const arma::Row< size_t > &responses, OptimizerType &optimizer)
*Train the **LogisticRegression** (p. 1795) model with the given instantiated optimizer.*

39.417.1 Detailed Description

```
template<typename MatType = arma::mat>
class mlpack::regression::LogisticRegression< MatType >
```

The **LogisticRegression** (p. 1795) class implements an L2-regularized logistic regression model, and supports training with multiple optimizers and classification.

The class supports different observation types via the MatType template parameter; for instance, logistic regression can be performed on sparse datasets by specifying arma::sp_mat as the MatType parameter.

LogisticRegression (p. 1795) can be used for general classification tasks, but the class is restricted to support only two classes. For multiclass logistic regression, see **mlpack::regression::SoftmaxRegression** (p. 1811).

Template Parameters

<i>MatType</i>	Type of data matrix.
----------------	----------------------

Definition at line 39 of file logistic_regression.hpp.

39.417.2 Constructor & Destructor Documentation

39.417.2.1 LogisticRegression() [1/4]

```
LogisticRegression (
    const MatType & predictors,
    const arma::Row< size_t > & responses,
    const double lambda = 0 )
```

Construct the **LogisticRegression** (p. 1795) class with the given labeled training data.

This will train the model. Optionally, specify *lambda*, which is the penalty parameter for L2-regularization. If not specified, it is set to 0, which results in standard (unregularized) logistic regression.

It is not possible to set a custom optimizer with this constructor. Either use a constructor that does not train and call **Train()** (p. 1802) with a custom optimizer type, or use the constructor that takes an instantiated optimizer. (This unfortunate situation is a language restriction of C++.)

Parameters

<i>predictors</i>	Input training variables.
<i>responses</i>	Outputs resulting from input training variables.
<i>lambda</i>	L2-regularization parameter.

39.417.2.2 LogisticRegression() [2/4]

```
LogisticRegression (
    const MatType & predictors,
    const arma::Row< size_t > & responses,
    const arma::rowvec & initialPoint,
    const double lambda = 0 )
```

Construct the **LogisticRegression** (p. 1795) class with the given labeled training data.

This will train the model. Optionally, specify *lambda*, which is the penalty parameter for L2-regularization. If not specified, it is set to 0, which results in standard (unregularized) logistic regression.

It is not possible to set a custom optimizer with this constructor. Either use a constructor that does not train and call **Train()** (p. 1802) with a custom optimizer type, or use the constructor that takes an instantiated optimizer. (This unfortunate situation is a language restriction of C++.)

Parameters

<i>predictors</i>	Input training variables.
<i>responses</i>	Outputs results from input training variables.
<i>initialPoint</i>	Initial model to train with.
<i>lambda</i>	L2-regularization parameter.

39.417.2.3 `LogisticRegression()` [3/4]

```
LogisticRegression (
    const size_t dimensionality = 0,
    const double lambda = 0 )
```

Construct the **LogisticRegression** (p. 1795) class without performing any training.

The dimensionality of the data (which will be used to set the size of the parameters vector) must be specified, and all of the parameters in the model will be set to 0. Note that the dimensionality may be changed later by directly modifying the parameters vector (using **Parameters()** (p. 1801)).

Parameters

<i>dimensionality</i>	Dimensionality of the data.
<i>lambda</i>	L2-regularization parameter.

39.417.2.4 `LogisticRegression()` [4/4]

```
LogisticRegression (
    const MatType & predictors,
    const arma::Row< size_t > & responses,
    OptimizerType & optimizer,
    const double lambda )
```

Construct the **LogisticRegression** (p. 1795) class with the given labeled training data.

This will train the model. This overload takes an already instantiated optimizer (which holds the **LogisticRegression**↔**Function** (p. 1803) error function, which must also be instantiated), so that the optimizer can be configured before the training is run by this constructor. The update policy of the optimizer can be set through the policy argument. The predictors and responses and initial point are all taken from the error function contained in the optimizer.

Parameters

<i>predictors</i>	Input training variables.
<i>responses</i>	Outputs results from input training variables.
<i>optimizer</i>	Instantiated optimizer with instantiated error function.
<i>lambda</i>	L2-regularization parameter.

39.417.3 Member Function Documentation

39.417.3.1 Classify() [1/3]

```
size_t Classify (
    const VecType & point,
    const double decisionBoundary = 0.5 ) const
```

Classify the given point.

The predicted label is returned. Optionally, specify the decision boundary; logistic regression returns a value between 0 and 1. If the value is greater than the decision boundary, the response is taken to be 1; otherwise, it is 0. By default the decision boundary is 0.5.

Parameters

<i>point</i>	Point to classify.
<i>decisionBoundary</i>	Decision boundary (default 0.5).

Returns

Predicted label of point.

Referenced by LogisticRegression< MatType >::Lambda().

39.417.3.2 Classify() [2/3]

```
void Classify (
    const MatType & dataset,
    arma::Row< size_t > & labels,
    const double decisionBoundary = 0.5 ) const
```

Classify the given points, returning the predicted labels for each point.

Optionally, specify the decision boundary; logistic regression returns a value between 0 and 1. If the value is greater than the decision boundary, the response is taken to be 1; otherwise, it is 0. By default the decision boundary is 0.5.

Parameters

<i>dataset</i>	Set of points to classify.
<i>labels</i>	Predicted labels for each point.
<i>decisionBoundary</i>	Decision boundary (default 0.5).

39.417.3.3 Classify() [3/3]

```
void Classify (
    const MatType & dataset,
    arma::mat & probabilities ) const
```

Classify the given points, returning class probabilities for each point.

Parameters

<i>dataset</i>	Set of points to classify.
<i>probabilities</i>	Class probabilities for each point (output).

39.417.3.4 ComputeAccuracy()

```
double ComputeAccuracy (
    const MatType & predictors,
    const arma::Row< size_t > & responses,
    const double decisionBoundary = 0.5 ) const
```

Compute the accuracy of the model on the given predictors and responses, optionally using the given decision boundary.

The responses should be either 0 or 1. Logistic regression returns a value between 0 and 1. If the value is greater than the decision boundary, the response is taken to be 1; otherwise, it is 0. By default, the decision boundary is 0.5.

The accuracy is returned as a percentage, between 0 and 100.

Parameters

<i>predictors</i>	Input predictors.
<i>responses</i>	Vector of responses.
<i>decisionBoundary</i>	Decision boundary (default 0.5).

Returns

Percentage of responses that are predicted correctly.

Referenced by LogisticRegression< MatType >::Lambda().

39.417.3.5 ComputeError()

```
double ComputeError (
    const MatType & predictors,
    const arma::Row< size_t > & responses ) const
```


Compute the error of the model.

This returns the negative objective function of the logistic regression log-likelihood function. For the model to be optimal, the negative log-likelihood function should be minimized.

Parameters

<i>predictors</i>	Input predictors.
<i>responses</i>	Vector of responses.

Referenced by LogisticRegression< MatType >::Lambda().

39.417.3.6 Lambda() [1/2]

```
const double& Lambda ( ) const [inline]
```

Return the lambda value for L2-regularization.

Definition at line 162 of file logistic_regression.hpp.

39.417.3.7 Lambda() [2/2]

```
double& Lambda ( ) [inline]
```

Modify the lambda value for L2-regularization.

Definition at line 164 of file logistic_regression.hpp.

References LogisticRegression< MatType >::Classify(), LogisticRegression< MatType >::ComputeAccuracy(), LogisticRegression< MatType >::ComputeError(), and LogisticRegression< MatType >::serialize().

39.417.3.8 Parameters() [1/2]

```
const arma::rowvec& Parameters ( ) const [inline]
```

Return the parameters (the b vector).

Definition at line 157 of file logistic_regression.hpp.

39.417.3.9 Parameters() [2/2]

```
arma::rowvec& Parameters ( ) [inline]
```

Modify the parameters (the **b** vector).

Definition at line 159 of file `logistic_regression.hpp`.

39.417.3.10 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the model.

Referenced by `LogisticRegression< MatType >::Lambda()`.

39.417.3.11 Train() [1/2]

```
double Train (
    const MatType & predictors,
    const arma::Row< size_t > & responses )
```

Train the **LogisticRegression** (p. 1795) model on the given input data.

By default, the L-BFGS optimization algorithm is used, but others can be specified (such as `ens::SGD`).

This will use the existing model parameters as a starting point for the optimization. If this is not what you want, then you should access the parameters vector directly with **Parameters()** (p. 1801) and modify it as desired.

Template Parameters

<i>OptimizerType</i>	Type of optimizer to use to train the model.
----------------------	--

Parameters

<i>predictors</i>	Input training variables.
<i>responses</i>	Outputs results from input training variables.

Returns

The final objective of the trained model (NaN or Inf on error)

39.417.3.12 Train() [2/2]

```
double Train (
    const MatType & predictors,
    const arma::Row< size_t > & responses,
    OptimizerType & optimizer )
```

Train the **LogisticRegression** (p. 1795) model with the given instantiated optimizer.

Using this overload allows configuring the instantiated optimizer before training is performed.

Note that the initial point of the optimizer (optimizer.Function().GetInitialPoint()) will be used as the initial point of the optimization, overwriting any existing trained model. If you don't want to overwrite the existing model, set optimizer.Function().GetInitialPoint() to the current parameters vector, accessible via **Parameters()** (p. 1801).

Parameters

<i>predictors</i>	Input training variables.
<i>responses</i>	Outputs results from input training variables.
<i>optimizer</i>	Instantiated optimizer with instantiated error function.

Returns

The final objective of the trained model (NaN or Inf on error)

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/logistic_regression/ **logistic_regression.hpp**

39.418 LogisticRegressionFunction< MatType > Class Template Reference

The log-likelihood function for the logistic regression objective function.

Public Member Functions

- **LogisticRegressionFunction** (const MatType &predictors, const arma::Row< size_t > &responses, const double lambda=0)
Creates the **LogisticRegressionFunction** (p. 1803).

- **LogisticRegressionFunction** (const MatType &predictors, const arma::Row< size_t > &responses, const arma::vec &initialPoint, const double lambda=0)
*Creates the **LogisticRegressionFunction** (p. 1803) with initialPoint.*
- double **Evaluate** (const arma::mat ¶meters) const
Evaluate the logistic regression log-likelihood function with the given parameters.
- double **Evaluate** (const arma::mat ¶meters, const size_t begin, const size_t batchSize=1) const
Evaluate the logistic regression log-likelihood function with the given parameters using the given batch size from the given point index.
- template<typename GradType >
double **EvaluateWithGradient** (const arma::mat ¶meters, GradType &gradient) const
Evaluate the objective function and gradient of the logistic regression log-likelihood function simultaneously with the given parameters.
- template<typename GradType >
double **EvaluateWithGradient** (const arma::mat ¶meters, const size_t begin, GradType &gradient, const size_t batchSize=1) const
Evaluate the objective function and gradient of the logistic regression log-likelihood function simultaneously with the given parameters, for the given batch size from a given point in the dataset.
- const arma::mat & **GetInitialPoint** () const
Return the initial point for the optimization.
- void **Gradient** (const arma::mat ¶meters, arma::mat &gradient) const
Evaluate the gradient of the logistic regression log-likelihood function with the given parameters.
- template<typename GradType >
void **Gradient** (const arma::mat ¶meters, const size_t begin, GradType &gradient, const size_t batchSize=1) const
Evaluate the gradient of the logistic regression log-likelihood function with the given parameters, for the given batch size from a given point in the dataset.
- const arma::mat & **InitialPoint** () const
Return the initial point for the optimization.
- arma::mat & **InitialPoint** ()
Modify the initial point for the optimization.
- const double & **Lambda** () const
Return the regularization parameter (lambda).
- double & **Lambda** ()
Modify the regularization parameter (lambda).
- size_t **NumFeatures** () const
Return the number of features(add 1 for the intercept term).
- size_t **NumFunctions** () const
Return the number of separable functions (the number of predictor points).
- void **PartialGradient** (const arma::mat ¶meters, const size_t j, arma::sp_mat &gradient) const
Evaluate the gradient of the logistic regression log-likelihood function with the given parameters, and with respect to only one feature in the dataset.
- const MatType & **Predictors** () const
Return the matrix of predictors.
- const arma::Row< size_t > & **Responses** () const
Return the vector of responses.
- void **Shuffle** ()
Shuffle the order of function visitation.

39.418.1 Detailed Description

```
template<typename MatType = arma::mat>
class mlpack::regression::LogisticRegressionFunction< MatType >
```

The log-likelihood function for the logistic regression objective function.

This is used by various mlpack optimizers to train a logistic regression model.

Definition at line 28 of file logistic_regression_function.hpp.

39.418.2 Constructor & Destructor Documentation

39.418.2.1 LogisticRegressionFunction() [1/2]

```
LogisticRegressionFunction (
    const MatType & predictors,
    const arma::Row< size_t > & responses,
    const double lambda = 0 )
```

Creates the **LogisticRegressionFunction** (p. 1803).

Parameters

<i>predictors</i>	The matrix of data points.
<i>responses</i>	The measured data for each point in predictors.
<i>lambda</i>	Regularization constant for ridge regression.

39.418.2.2 LogisticRegressionFunction() [2/2]

```
LogisticRegressionFunction (
    const MatType & predictors,
    const arma::Row< size_t > & responses,
    const arma::vec & initialPoint,
    const double lambda = 0 )
```

Creates the **LogisticRegressionFunction** (p. 1803) with *initialPoint*.

Parameters

<i>predictors</i>	The matrix of data points.
-------------------	----------------------------

Parameters

<i>responses</i>	The measured data for each point in predictors.
<i>initialPoint</i>	Point from which to start the optimization.
<i>lambda</i>	Regularization constant for ridge regression.

39.418.3 Member Function Documentation

39.418.3.1 Evaluate() [1/2]

```
double Evaluate (
    const arma::mat & parameters ) const
```

Evaluate the logistic regression log-likelihood function with the given parameters.

Note that if a point has 0 probability of being classified directly with the given parameters, then **Evaluate()** (p. 1806) will return nan (this is kind of a corner case and should not happen for reasonable models).

The optimum (minimum) of this function is 0.0, and occurs when each point is classified correctly with very high probability.

Parameters

<i>parameters</i>	Vector of logistic regression parameters.
-------------------	---

Referenced by LogisticRegressionFunction< MatType >::Responses().

39.418.3.2 Evaluate() [2/2]

```
double Evaluate (
    const arma::mat & parameters,
    const size_t begin,
    const size_t batchSize = 1 ) const
```

Evaluate the logistic regression log-likelihood function with the given parameters using the given batch size from the given point index.

This is useful for optimizers such as SGD, which require a separable objective function. Note that if the points have 0 probability of being classified correctly with the given parameters, then **Evaluate()** (p. 1806) will return nan (this is kind of a corner case and should not happen for reasonable models).

The optimum (minimum) of this function is 0.0, and occurs when the points are classified correctly with very high probability.

Parameters

<i>parameters</i>	Vector of logistic regression parameters.
<i>begin</i>	Index of the starting point to use for objective function evaluation.
<i>batchSize</i>	Number of points to be passed at a time to use for objective function evaluation.

39.418.3.3 EvaluateWithGradient() [1/2]

```
double EvaluateWithGradient (
    const arma::mat & parameters,
    GradType & gradient ) const
```

Evaluate the objective function and gradient of the logistic regression log-likelihood function simultaneously with the given parameters.

Referenced by LogisticRegressionFunction< MatType >::Responses().

39.418.3.4 EvaluateWithGradient() [2/2]

```
double EvaluateWithGradient (
    const arma::mat & parameters,
    const size_t begin,
    GradType & gradient,
    const size_t batchSize = 1 ) const
```

Evaluate the objective function and gradient of the logistic regression log-likelihood function simultaneously with the given parameters, for the given batch size from a given point in the dataset.

39.418.3.5 GetInitialPoint()

```
const arma::mat& GetInitialPoint ( ) const [inline]
```

Return the initial point for the optimization.

Definition at line 172 of file logistic_regression_function.hpp.

39.418.3.6 Gradient() [1/2]

```
void Gradient (
    const arma::mat & parameters,
    arma::mat & gradient ) const
```

Evaluate the gradient of the logistic regression log-likelihood function with the given parameters.

Parameters

<i>parameters</i>	Vector of logistic regression parameters.
<i>gradient</i>	Vector to output gradient into.

Referenced by `LogisticRegressionFunction< MatType >::Responses()`.

39.418.3.7 Gradient() [2/2]

```
void Gradient (
    const arma::mat & parameters,
    const size_t begin,
    GradType & gradient,
    const size_t batchSize = 1 ) const
```

Evaluate the gradient of the logistic regression log-likelihood function with the given parameters, for the given batch size from a given point in the dataset.

This is useful for optimizers such as SGD, which require a separable objective function.

Parameters

<i>parameters</i>	Vector of logistic regression parameters.
<i>begin</i>	Index of the starting point to use for objective function gradient evaluation.
<i>gradient</i>	Vector to output gradient into.
<i>batchSize</i>	Number of points to be processed as a batch for objective function gradient evaluation.

39.418.3.8 InitialPoint() [1/2]

```
const arma::mat& InitialPoint ( ) const [inline]
```

Return the initial point for the optimization.

Definition at line 56 of file `logistic_regression_function.hpp`.

39.418.3.9 InitialPoint() [2/2]

```
arma::mat& InitialPoint ( ) [inline]
```

Modify the initial point for the optimization.

Definition at line 58 of file `logistic_regression_function.hpp`.

39.418.3.10 Lambda() [1/2]

```
const double& Lambda ( ) const [inline]
```

Return the regularization parameter (lambda).

Definition at line 61 of file logistic_regression_function.hpp.

39.418.3.11 Lambda() [2/2]

```
double& Lambda ( ) [inline]
```

Modify the regularization parameter (lambda).

Definition at line 63 of file logistic_regression_function.hpp.

39.418.3.12 NumFeatures()

```
size_t NumFeatures ( ) const [inline]
```

Return the number of features(add 1 for the intercept term).

Definition at line 178 of file logistic_regression_function.hpp.

39.418.3.13 NumFunctions()

```
size_t NumFunctions ( ) const [inline]
```

Return the number of separable functions (the number of predictor points).

Definition at line 175 of file logistic_regression_function.hpp.

39.418.3.14 PartialGradient()

```
void PartialGradient (
    const arma::mat & parameters,
    const size_t j,
    arma::sp_mat & gradient ) const
```

Evaluate the gradient of the logistic regression log-likelihood function with the given parameters, and with respect to only one feature in the dataset.

This is useful for optimizers such as SCD, which require partial gradients.

Parameters

<i>parameters</i>	Vector of logistic regression parameters.
<i>j</i>	Index of the feature with respect to which the gradient is to be computed.
<i>gradient</i>	Sparse matrix to output gradient into.

Referenced by `LogisticRegressionFunction< MatType >::Responses()`.

39.418.3.15 Predictors()

```
const MatType& Predictors ( ) const [inline]
```

Return the matrix of predictors.

Definition at line 66 of file `logistic_regression_function.hpp`.

39.418.3.16 Responses()

```
const arma::Row<size_t>& Responses ( ) const [inline]
```

Return the vector of responses.

Definition at line 68 of file `logistic_regression_function.hpp`.

References `LogisticRegressionFunction< MatType >::Evaluate()`, `LogisticRegressionFunction< MatType >::EvaluateWithGradient()`, `LogisticRegressionFunction< MatType >::Gradient()`, `LogisticRegressionFunction< MatType >::PartialGradient()`, and `LogisticRegressionFunction< MatType >::Shuffle()`.

39.418.3.17 Shuffle()

```
void Shuffle ( )
```

Shuffle the order of function visitation.

This may be called by the optimizer.

Referenced by `LogisticRegressionFunction< MatType >::Responses()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/logistic_regression/ logistic_regression_function.`
`hpp`

39.419 SoftmaxRegression Class Reference

Softmax Regression is a classifier which can be used for classification when the data available can take two or more class values.

Public Member Functions

- **SoftmaxRegression** (const size_t inputSize=0, const size_t numClasses=0, const bool fitIntercept=false)
*Initialize the **SoftmaxRegression** (p. 1811) without performing training.*
- template<typename OptimizerType = ens::L_BFGS>
SoftmaxRegression (const arma::mat &data, const arma::Row< size_t > &labels, const size_t numClasses, const double lambda=0.0001, const bool fitIntercept=false, OptimizerType optimizer=OptimizerType())
*Construct the **SoftmaxRegression** (p. 1811) class with the provided data and labels.*
- void **Classify** (const arma::mat &dataset, arma::Row< size_t > &labels) const
Classify the given points, returning the predicted labels for each point.
- template<typename VecType >
size_t **Classify** (const VecType &point) const
Classify the given point.
- void **Classify** (const arma::mat &dataset, arma::Row< size_t > &labels, arma::mat &probabilites) const
Classify the given points, returning class probabilities and predicted class label for each point.
- void **Classify** (const arma::mat &dataset, arma::mat &probabilities) const
Classify the given points, returning class probabilities for each point.
- double **ComputeAccuracy** (const arma::mat &testData, const arma::Row< size_t > &labels) const
Computes accuracy of the learned model given the feature data and the labels associated with each data point.
- size_t **FeatureSize** () const
Gets the features size of the training data.
- bool **FitIntercept** () const
Gets the intercept term flag. We can't change this after training.
- double & **Lambda** ()
Sets the regularization parameter.
- double **Lambda** () const
Gets the regularization parameter.
- size_t & **NumClasses** ()
Sets the number of classes.
- size_t **NumClasses** () const
Gets the number of classes.
- arma::mat & **Parameters** ()
Get the model parameters.
- const arma::mat & **Parameters** () const
Get the model parameters.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
*Serialize the **SoftmaxRegression** (p. 1811) model.*
- template<typename OptimizerType = ens::L_BFGS>
double **Train** (const arma::mat &data, const arma::Row< size_t > &labels, const size_t numClasses, OptimizerType optimizer=OptimizerType())
Train the softmax regression with the given training data.

39.419.1 Detailed Description

Softmax Regression is a classifier which can be used for classification when the data available can take two or more class values.

It is a generalization of Logistic Regression (which is used only for binary classification). The model has a different set of parameters for each class, but can be easily converted into a vectorized implementation as has been done in this module. The model can be used for direct classification of feature data or in conjunction with unsupervised learning methods. More technical details about the model can be found on the following webpage:

http://ufldl.stanford.edu/wiki/index.php/Softmax_Regression

An example on how to use the interface is shown below:

```
arma::mat trainData; // Training data matrix.
arma::Row<size_t> labels; // Labels associated with the data.
const size_t inputSize = 1000; // Size of input feature vector.
const size_t numClasses = 10; // Number of classes.
const double lambda = 0.0001; // L2-Regularization parameter.

const size_t numBasis = 5; // Parameter required for L-BFGS algorithm.
const size_t numIterations = 100; // Maximum number of iterations.

// Train the model using an instantiated optimizer for the training.
SoftmaxRegression regressor(trainData.n_rows, numClasses);
ens::L_BFGS optimizer(numBasis, numIterations);
regressor.Train(trainData, labels, numClasses, std::move(optimizer));

arma::mat testData; // Test data matrix.
arma::Row<size_t> predictions; // Vectors to store predictions in.

// Obtain predictions from both the learned models.
regressor.Classify(testData, predictions);
```

Definition at line 59 of file softmax_regression.hpp.

39.419.2 Constructor & Destructor Documentation

39.419.2.1 SoftmaxRegression() [1/2]

```
SoftmaxRegression (
    const size_t inputSize = 0,
    const size_t numClasses = 0,
    const bool fitIntercept = false )
```

Initialize the **SoftmaxRegression** (p. 1811) without performing training.

Default value of lambda is 0.0001. Be sure to use **Train()** (p. 1817) before calling **Classify()** (p. 1813) or **ComputeAccuracy()** (p. 1815), otherwise the results may be meaningless.

Parameters

<i>inputSize</i>	Size of the input feature vector.
<i>numClasses</i>	Number of classes for classification.
<i>fitIntercept</i>	add intercept term or not.

39.419.2.2 SoftmaxRegression() [2/2]

```

SoftmaxRegression (
    const arma::mat & data,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const double lambda = 0.0001,
    const bool fitIntercept = false,
    OptimizerType optimizer = OptimizerType() )

```

Construct the **SoftmaxRegression** (p. 1811) class with the provided data and labels.

This will train the model. Optionally, the parameter 'lambda' can be passed, which controls the amount of L2-regularization in the objective function. By default, the model takes a small value.

Template Parameters

<i>OptimizerType</i>	Desired optimizer type.
----------------------	-------------------------

Parameters

<i>data</i>	Input training features. Each column associate with one sample
<i>labels</i>	Labels associated with the feature data.
<i>inputSize</i>	Size of the input feature vector.
<i>numClasses</i>	Number of classes for classification.
<i>optimizer</i>	Desired optimizer.
<i>lambda</i>	L2-regularization constant.
<i>fitIntercept</i>	add intercept term or not.

39.419.3 Member Function Documentation

39.419.3.1 Classify() [1/4]

```

void Classify (
    const arma::mat & dataset,

```

```
arma::Row< size_t > & labels ) const
```

Classify the given points, returning the predicted labels for each point.

The function calculates the probabilities for every class, given a data point. It then chooses the class which has the highest probability among all.

Parameters

<i>dataset</i>	Set of points to classify.
<i>labels</i>	Predicted labels for each point.

39.419.3.2 Classify() [2/4]

```
size_t Classify (
    const VecType & point ) const
```

Classify the given point.

The predicted class label is returned. The function calculates the probabilities for every class, given the point. It then chooses the class which has the highest probability among all.

Parameters

<i>point</i>	Point to be classified.
--------------	-------------------------

Returns

Predicted class label of the point.

39.419.3.3 Classify() [3/4]

```
void Classify (
    const arma::mat & dataset,
    arma::Row< size_t > & labels,
    arma::mat & probabilites ) const
```

Classify the given points, returning class probabilities and predicted class label for each point.

The function calculates the probabilities for every class, given a data point. It then chooses the class which has the highest probability among all.

Parameters

<i>dataset</i>	Matrix of data points to be classified.
<i>labels</i>	Predicted labels for each point.
<i>probabilities</i>	Class probabilities for each point.

39.419.3.4 Classify() [4/4]

```
void Classify (
    const arma::mat & dataset,
    arma::mat & probabilities ) const
```

Classify the given points, returning class probabilities for each point.

Parameters

<i>dataset</i>	Matrix of data points to be classified.
<i>probabilities</i>	Class probabilities for each point.

39.419.3.5 ComputeAccuracy()

```
double ComputeAccuracy (
    const arma::mat & testData,
    const arma::Row< size_t > & labels ) const
```

Computes accuracy of the learned model given the feature data and the labels associated with each data point.

Predictions are made using the provided data and are compared with the actual labels.

Parameters

<i>testData</i>	Matrix of data points using which predictions are made.
<i>labels</i>	Vector of labels associated with the data.

39.419.3.6 FeatureSize()

```
size_t FeatureSize ( ) const [inline]
```

Gets the features size of the training data.

Definition at line 190 of file softmax_regression.hpp.

39.419.3.7 FitIntercept()

```
bool FitIntercept ( ) const [inline]
```

Gets the intercept term flag. We can't change this after training.

Definition at line 182 of file softmax_regression.hpp.

39.419.3.8 Lambda() [1/2]

```
double& Lambda ( ) [inline]
```

Sets the regularization parameter.

Definition at line 177 of file softmax_regression.hpp.

39.419.3.9 Lambda() [2/2]

```
double Lambda ( ) const [inline]
```

Gets the regularization parameter.

Definition at line 179 of file softmax_regression.hpp.

39.419.3.10 NumClasses() [1/2]

```
size_t& NumClasses ( ) [inline]
```

Sets the number of classes.

Definition at line 172 of file softmax_regression.hpp.

39.419.3.11 NumClasses() [2/2]

```
size_t NumClasses ( ) const [inline]
```

Gets the number of classes.

Definition at line 174 of file softmax_regression.hpp.

39.419.3.12 Parameters() [1/2]

```
arma::mat& Parameters ( ) [inline]
```

Get the model parameters.

Definition at line 185 of file softmax_regression.hpp.

39.419.3.13 Parameters() [2/2]

```
const arma::mat& Parameters ( ) const [inline]
```

Get the model parameters.

Definition at line 187 of file softmax_regression.hpp.

39.419.3.14 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the **SoftmaxRegression** (p. 1811) model.

Definition at line 198 of file softmax_regression.hpp.

39.419.3.15 Train()

```
double Train (
    const arma::mat & data,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    OptimizerType optimizer = OptimizerType() )
```

Train the softmax regression with the given training data.

Template Parameters

<i>OptimizerType</i>	Desired optimizer type.
----------------------	-------------------------

Parameters

<i>data</i>	Input data with each column as one example.
<i>labels</i>	Labels associated with the feature data.
<i>numClasses</i>	Number of classes for classification.
<i>optimizer</i>	Desired optimizer.

Returns

Objective value of the final point.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/softmax_regression/ **softmax_regression.hpp**

39.420 SoftmaxRegressionFunction Class Reference

Public Member Functions

- **SoftmaxRegressionFunction** (const arma::mat &data, const arma::Row< size_t > &labels, const size_t numClasses, const double lambda=0.0001, const bool fitIntercept=false)
Construct the Softmax Regression objective function with the given parameters.
- double **Evaluate** (const arma::mat ¶meters) const
Evaluates the objective function of the softmax regression model using the given parameters.
- double **Evaluate** (const arma::mat ¶meters, const size_t start, const size_t batchSize=1) const
Evaluate the objective function of the softmax regression model for a subset of the data points using the given parameters.
- bool **FitIntercept** () const
Gets the intercept flag.
- void **GetGroundTruthMatrix** (const arma::Row< size_t > &labels, arma::sp_mat &groundTruth)
Constructs the ground truth label matrix with the passed labels.
- const arma::mat & **GetInitialPoint** () const
Return the initial point for the optimization.
- void **GetProbabilitiesMatrix** (const arma::mat ¶meters, arma::mat &probabilities, const size_t start, const size_t batchSize) const
Evaluate the probabilities matrix with the passed parameters.
- void **Gradient** (const arma::mat ¶meters, arma::mat &gradient) const
Evaluates the gradient values of the objective function given the current set of parameters.
- void **Gradient** (const arma::mat ¶meters, const size_t start, arma::mat &gradient, const size_t batchSize=1) const
Evaluate the gradient of the objective function given the current set of parameters, on a subset of the data.

- `const arma::mat InitializeWeights ()`
Initializes the parameters of the model to suitable values.
- `double & Lambda ()`
Sets the regularization parameter.
- `double Lambda () const`
Gets the regularization parameter.
- `size_t NumClasses () const`
Gets the number of classes.
- `size_t NumFeatures () const`
Gets the features size of the training data.
- `void PartialGradient (const arma::mat ¶meters, size_t j, arma::sp_mat &gradient) const`
Evaluates the gradient values of the objective function given the current set of parameters for a single feature indexed by j.
- `void Shuffle ()`
Shuffle the dataset.

Static Public Member Functions

- `static const arma::mat InitializeWeights (const size_t featureSize, const size_t numClasses, const bool fit↵ Intercept=false)`
Initialize Softmax Regression weights (trainable parameters) with the given parameters.
- `static void InitializeWeights (arma::mat &weights, const size_t featureSize, const size_t numClasses, const bool fitIntercept=false)`
Initialize Softmax Regression weights (trainable parameters) with the given parameters.

39.420.1 Detailed Description

Definition at line 21 of file softmax_regression_function.hpp.

39.420.2 Constructor & Destructor Documentation

39.420.2.1 SoftmaxRegressionFunction()

```
SoftmaxRegressionFunction (
    const arma::mat & data,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const double lambda = 0.0001,
    const bool fitIntercept = false )
```

Construct the Softmax Regression objective function with the given parameters.

Parameters

<i>data</i>	Input training data, each column associate with one sample
<i>labels</i>	Labels associated with the feature data.
<i>inputSize</i>	Size of the input feature vector.
<i>numClasses</i>	Number of classes for classification.
<i>lambda</i>	L2-regularization constant.
<i>fitIntercept</i>	Intercept term flag.

39.420.3 Member Function Documentation

39.420.3.1 Evaluate() [1/2]

```
double Evaluate (
    const arma::mat & parameters ) const
```

Evaluates the objective function of the softmax regression model using the given parameters.

The cost function has terms for the log likelihood error and the regularization cost. The objective function takes a low value when the model generalizes well for the given training data, while having small parameter values.

Parameters

<i>parameters</i>	Current values of the model parameters.
-------------------	---

39.420.3.2 Evaluate() [2/2]

```
double Evaluate (
    const arma::mat & parameters,
    const size_t start,
    const size_t batchSize = 1 ) const
```

Evaluate the objective function of the softmax regression model for a subset of the data points using the given parameters.

The cost function has terms for the log likelihood error and the regularization cost. The objective function takes a low value when the model generalizes well for the given training data, while having small parameter values.

Parameters

<i>parameters</i>	Current values of the model parameters.
<i>start</i>	First index of the data points to use.
<i>batchSize</i>	Number of data points to evaluate objective for.

39.420.3.3 FitIntercept()

```
bool FitIntercept ( ) const [inline]
```

Gets the intercept flag.

Definition at line 185 of file softmax_regression_function.hpp.

39.420.3.4 GetGroundTruthMatrix()

```
void GetGroundTruthMatrix (
    const arma::Row< size_t > & labels,
    arma::sp_mat & groundTruth )
```

Constructs the ground truth label matrix with the passed labels.

Parameters

<i>labels</i>	Labels associated with the training data.
<i>groundTruth</i>	Pointer to arma::mat which stores the computed matrix.

39.420.3.5 GetInitialPoint()

```
const arma::mat& GetInitialPoint ( ) const [inline]
```

Return the initial point for the optimization.

Definition at line 168 of file softmax_regression_function.hpp.

39.420.3.6 GetProbabilitiesMatrix()

```
void GetProbabilitiesMatrix (
    const arma::mat & parameters,
    arma::mat & probabilities,
    const size_t start,
    const size_t batchSize ) const
```

Evaluate the probabilities matrix with the passed parameters.

$\text{probabilities}(i, j) = \exp(*\text{data}_j) / \sum_k(\exp(*\text{data}_j))$. It represents the probability of data_j belongs to class i .

Parameters

<i>parameters</i>	Current values of the model parameters.
<i>probabilities</i>	Pointer to arma::mat which stores the probabilities.
<i>start</i>	Index of point to start at.
<i>batchSize</i>	Number of points to calculate probabilities for.

39.420.3.7 Gradient() [1/2]

```
void Gradient (
    const arma::mat & parameters,
    arma::mat & gradient ) const
```

Evaluates the gradient values of the objective function given the current set of parameters.

The function calculates the probabilities for each class given the parameters, and computes the gradients based on the difference from the ground truth.

Parameters

<i>parameters</i>	Current values of the model parameters.
<i>gradient</i>	Matrix where gradient values will be stored.

39.420.3.8 Gradient() [2/2]

```
void Gradient (
    const arma::mat & parameters,
    const size_t start,
    arma::mat & gradient,
    const size_t batchSize = 1 ) const
```

Evaluate the gradient of the objective function given the current set of parameters, on a subset of the data.

The function calculates the probabilities for each class given the parameters, and computes the gradients based on the difference from the ground truth.

Parameters

<i>parameters</i>	Current values of the model parameters.
<i>start</i>	First index of the data points to use.
<i>gradient</i>	Matrix to store gradient into.
<i>batchSize</i>	Number of data points to evaluate gradient for.

39.420.3.9 InitializeWeights() [1/3]

```
const arma::mat InitializeWeights ( )
```

Initializes the parameters of the model to suitable values.

39.420.3.10 InitializeWeights() [2/3]

```
static const arma::mat InitializeWeights (
    const size_t featureSize,
    const size_t numClasses,
    const bool fitIntercept = false ) [static]
```

Initialize Softmax Regression weights (trainable parameters) with the given parameters.

Parameters

<i>featureSize</i>	The number of features in the training set.
<i>numClasses</i>	Number of classes for classification.
<i>fitIntercept</i>	If true, an intercept is fitted.

Returns

Initialized model weights.

39.420.3.11 InitializeWeights() [3/3]

```
static void InitializeWeights (
    arma::mat & weights,
    const size_t featureSize,
    const size_t numClasses,
    const bool fitIntercept = false ) [static]
```

Initialize Softmax Regression weights (trainable parameters) with the given parameters.

Parameters

<i>weights</i>	This will be filled with the initialized model weights.
<i>featureSize</i>	The number of features in the training set.
<i>numClasses</i>	Number of classes for classification.
<i>fitIntercept</i>	Intercept term flag.

39.420.3.12 Lambda() [1/2]

```
double& Lambda ( ) [inline]
```

Sets the regularization parameter.

Definition at line 180 of file softmax_regression_function.hpp.

39.420.3.13 Lambda() [2/2]

```
double Lambda ( ) const [inline]
```

Gets the regularization parameter.

Definition at line 182 of file softmax_regression_function.hpp.

39.420.3.14 NumClasses()

```
size_t NumClasses ( ) const [inline]
```

Gets the number of classes.

Definition at line 171 of file softmax_regression_function.hpp.

39.420.3.15 NumFeatures()

```
size_t NumFeatures ( ) const [inline]
```

Gets the features size of the training data.

Definition at line 174 of file softmax_regression_function.hpp.

39.420.3.16 PartialGradient()

```
void PartialGradient (
    const arma::mat & parameters,
    size_t j,
    arma::sp_mat & gradient ) const
```

Evaluates the gradient values of the objective function given the current set of parameters for a single feature indexed by j.

Parameters

<i>parameters</i>	Current values of the model parameters.
<i>j</i>	The index of the feature with respect to which the partial gradient is to be computed.
<i>gradient</i>	Out param for the gradient value.

39.420.3.17 Shuffle()

```
void Shuffle ( )
```

Shuffle the dataset.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/softmax_regression/function.hpp` **softmax_regression_↵**

39.421 Acrobot Class Reference

Implementation of **Acrobot** (p. 1825) game.

Classes

- class **State**

Public Types

- enum **Action** {
 negativeTorque,
 zeroTorque,
 positiveTorque,
 size }

Public Member Functions

- **Acrobot** (const double gravity=9.81, const double linkLength1=1.0, const double linkLength2=1.0, const double linkMass1=1.0, const double linkMass2=1.0, const double linkCom1=0.5, const double linkCom2=0.5, const double linkMoi=1.0, const double maxVel1=4 * **M_PI**, const double maxVel2=9 * **M_PI**, const double dt=0.2, const double doneReward=0)
*Construct a **Acrobot** (p. 1825) instance using the given constants.*
- arma::colvec **Dsdt** (arma::colvec state, const double torque) const
This is the ordinary differential equations required for estimation of nextState through RK4 method.
- **State** **InitialSample** () const
This function does random initialization of state space.
- bool **IsTerminal** (const **State** &state) const
This function checks if the acrobot has reached the terminal state.
- arma::colvec **Rk4** (const arma::colvec state, const double torque) const
This function calls the RK4 iterative method to estimate the next state based on given ordinary differential equation.
- double **Sample** (const **State** &state, const **Action** &action, **State** &nextState) const
*Dynamics of the **Acrobot** (p. 1825) System.*
- double **Sample** (const **State** &state, const **Action** &action) const
*Dynamics of the **Acrobot** (p. 1825) System.*
- double **Torque** (const **Action** &action) const
This function calculates the torque for a particular action.
- double **Wrap** (double value, const double minimum, const double maximum) const
Wrap funtion is required to truncate the angle value from -180 to 180.

39.421.1 Detailed Description

Implementation of **Acrobot** (p. 1825) game.

Acrobot (p. 1825) is a 2-link pendulum with only the second joint actuated. Initially, both links point downwards. The goal is to swing the end-effector at a height at least the length of one link above the base. Both links can swing freely and can pass by each other, i.e., they don't collide when they have the same angle.

Definition at line 28 of file acrobot.hpp.

39.421.2 Member Enumeration Documentation

39.421.2.1 Action

enum **Action**

Enumerator

negativeTorque	
zeroTorque	
positiveTorque	
size	

Definition at line 88 of file `acrobot.hpp`.

39.421.3 Constructor & Destructor Documentation

39.421.3.1 Acrobat()

```
Acrobot (
    const double gravity = 9.81,
    const double linkLength1 = 1.0,
    const double linkLength2 = 1.0,
    const double linkMass1 = 1.0,
    const double linkMass2 = 1.0,
    const double linkCom1 = 0.5,
    const double linkCom2 = 0.5,
    const double linkMoi = 1.0,
    const double maxVel1 = 4 * M_PI,
    const double maxVel2 = 9 * M_PI,
    const double dt = 0.2,
    const double doneReward = 0 ) [inline]
```

Construct a **Acrobot** (p. 1825) instance using the given constants.

Parameters

<i>gravity</i>	The gravity parameter.
<i>linkLength1</i>	The length of link 1.
<i>linkLength2</i>	The length of link 2.
<i>linkMass1</i>	The mass of link 1.
<i>linkMass2</i>	The mass of link 2.
<i>linkCom1</i>	The position of the center of mass of link 1.
<i>linkCom2</i>	The position of the center of mass of link 2.
<i>linkMoi</i>	The moments of inertia for both link.
<i>maxVel1</i>	The max angular velocity of link1.
<i>maxVel2</i>	The max angular velocity of link2.
<i>dt</i>	The differential value.

Definition at line 113 of file `acrobot.hpp`.

39.421.4 Member Function Documentation

39.421.4.1 Dsdt()

```
arma::colvec Dsdt (
    arma::colvec state,
    const double torque ) const [inline]
```

This is the ordinary differential equations required for estimation of nextState through RK4 method.

Parameters

<i>state</i>	Current State (p. 1831).
<i>torque</i>	The torque Applied.

Definition at line 218 of file acrobot.hpp.

References M_PI.

Referenced by Acrobot::Rk4().

39.421.4.2 InitialSample()

```
State InitialSample ( ) const [inline]
```

This function does random initialization of state space.

Definition at line 195 of file acrobot.hpp.

References Acrobot::State::State().

39.421.4.3 IsTerminal()

```
bool IsTerminal (
    const State & state ) const [inline]
```

This function checks if the acrobot has reached the terminal state.

Parameters

<i>state</i>	The current State (p. 1831).
--------------	-------------------------------------

Definition at line 205 of file acrobot.hpp.

References Acrobot::State::Theta1(), and Acrobot::State::Theta2().

Referenced by Acrobot::Sample().

39.421.4.4 Rk4()

```
arma::colvec Rk4 (
    const arma::colvec state,
    const double torque ) const [inline]
```

This function calls the RK4 iterative method to estimate the next state based on given ordinary differential equation.

Parameters

<i>state</i>	The current State (p. 1831).
<i>torque</i>	The torque applied.

Definition at line 305 of file acrobot.hpp.

References Acrobot::Dsdt().

Referenced by Acrobot::Sample().

39.421.4.5 Sample() [1/2]

```
double Sample (
    const State & state,
    const Action & action,
    State & nextState ) const [inline]
```

Dynamics of the **Acrobot** (p. 1825) System.

To get reward and next state based on current state and current action. Always return -1 reward.

Parameters

<i>state</i>	The current State (p. 1831).
<i>action</i>	The action taken.
<i>nextState</i>	The next state.

Returns

reward, it's always -1.0.

The value of angular velocity is bounded in min and max value.

If the acrobot reaches a terminal state, it should be given a positive reward. This will ensure that the agent learns the goal of the game.

Definition at line 148 of file acrobot.hpp.

References `Acrobot::State::AngularVelocity1()`, `Acrobot::State::AngularVelocity2()`, `Acrobot::IsTerminal()`, `M_PI`, `Acrobot::Rk4()`, `Acrobot::State::Theta1()`, `Acrobot::State::Theta2()`, `Acrobot::Torque()`, and `Acrobot::Wrap()`.

Referenced by `Acrobot::Sample()`.

39.421.4.6 `Sample()` [2/2]

```
double Sample (
    const State & state,
    const Action & action ) const [inline]
```

Dynamics of the **Acrobot** (p. 1825) System.

To get reward and next state based on current state and current action. This function calls the `Sample` function to estimate the next state return reward for taking a particular action.

Parameters

<i>state</i>	The current State (p. 1831).
<i>action</i>	The action taken.
<i>nextState</i>	The next state.

Definition at line 186 of file acrobot.hpp.

References `Acrobot::Sample()`.

39.421.4.7 `Torque()`

```
double Torque (
    const Action & action ) const [inline]
```

This function calculates the torque for a particular action.

0 : negative torque, 1 : zero torque, 2 : positive torque.

Parameters

<i>Action</i>	action taken.
---------------	---------------

Definition at line 291 of file acrobot.hpp.

References `mlpack::math::Random()`.

Referenced by `Acrobot::Sample()`.

39.421.4.8 Wrap()

```
double Wrap (
    double value,
    const double minimum,
    const double maximum ) const [inline]
```

Wrap function is required to truncate the angle value from -180 to 180.

This function will make sure that value will always be between minimum to maximum.

Parameters

<i>value</i>	Scalar value to wrap.
<i>minimum</i>	Minimum range of wrap.
<i>maximum</i>	Maximum range of wrap.

Definition at line 267 of file acrobot.hpp.

Referenced by `Acrobot::Sample()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/ acrobot.hpp`

39.422 Acrobot::State Class Reference

Public Member Functions

- **State** ()
Construct a state instance.
- **State** (const arma::colvec &data)
Construct a state instance from given data.
- double **AngularVelocity1** () const
Get value of Angular velocity (one).
- double & **AngularVelocity1** ()
Modify the angular velocity (one).

- double **AngularVelocity2** () const
Get value of Angular velocity (two).
- double & **AngularVelocity2** ()
Modify the angular velocity (two).
- arma::colvec & **Data** ()
Modify the state representation.
- const arma::colvec & **Encode** () const
Encode the state to a column vector.
- double **Theta1** () const
Get value of theta (one).
- double & **Theta1** ()
Modify value of theta (one).
- double **Theta2** () const
Get value of theta (two).
- double & **Theta2** ()
Modify value of theta (two).

Static Public Attributes

- static constexpr size_t **dimension** = 4
Dimension of the encoded state.

39.422.1 Detailed Description

Definition at line 35 of file acrobot.hpp.

39.422.2 Constructor & Destructor Documentation

39.422.2.1 State() [1/2]

```
State ( ) [inline]
```

Construct a state instance.

Definition at line 41 of file acrobot.hpp.

Referenced by Acrobot::InitialSample().

39.422.2.2 State() [2/2]

```
State (
    const arma::colvec & data ) [inline]
```

Construct a state instance from given data.

Parameters

<i>data</i>	Data for the theta and angular velocity of two links.
-------------	---

Definition at line 48 of file acrobot.hpp.

39.422.3 Member Function Documentation

39.422.3.1 AngularVelocity1() [1/2]

```
double AngularVelocity1 ( ) const [inline]
```

Get value of Angular velocity (one).

Definition at line 65 of file acrobot.hpp.

Referenced by Acrobot::Sample().

39.422.3.2 AngularVelocity1() [2/2]

```
double& AngularVelocity1 ( ) [inline]
```

Modify the angular velocity (one).

Definition at line 67 of file acrobot.hpp.

39.422.3.3 AngularVelocity2() [1/2]

```
double AngularVelocity2 ( ) const [inline]
```

Get value of Angular velocity (two).

Definition at line 70 of file acrobot.hpp.

Referenced by Acrobot::Sample().

39.422.3.4 AngularVelocity2() [2/2]

```
double& AngularVelocity2 ( ) [inline]
```

Modify the angular velocity (two).

Definition at line 72 of file acrobot.hpp.

39.422.3.5 Data()

```
arma::colvec& Data ( ) [inline]
```

Modify the state representation.

Definition at line 52 of file acrobot.hpp.

39.422.3.6 Encode()

```
const arma::colvec& Encode ( ) const [inline]
```

Encode the state to a column vector.

Definition at line 75 of file acrobot.hpp.

39.422.3.7 Theta1() [1/2]

```
double Theta1 ( ) const [inline]
```

Get value of theta (one).

Definition at line 55 of file acrobot.hpp.

Referenced by Acrobot::IsTerminal(), and Acrobot::Sample().

39.422.3.8 Theta1() [2/2]

```
double& Theta1 ( ) [inline]
```

Modify value of theta (one).

Definition at line 57 of file acrobot.hpp.

39.422.3.9 Theta2() [1/2]

```
double Theta2 ( ) const [inline]
```

Get value of theta (two).

Definition at line 60 of file acrobot.hpp.

Referenced by Acrobot::IsTerminal(), and Acrobot::Sample().

39.422.3.10 Theta2() [2/2]

```
double& Theta2 ( ) [inline]
```

Modify value of theta (two).

Definition at line 62 of file acrobot.hpp.

39.422.4 Member Data Documentation**39.422.4.1 dimension**

```
constexpr size_t dimension = 4 [static]
```

Dimension of the encoded state.

Definition at line 78 of file acrobot.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/ **acrobot.hpp**

39.423 AggregatedPolicy< PolicyType > Class Template Reference**Public Types**

- using **ActionType** = typename PolicyType::ActionType
Convenient typedef for action.

Public Member Functions

- **AggregatedPolicy** (std::vector< PolicyType > policies, const arma::colvec &distribution)
- void **Anneal** ()
Exploration probability will anneal at each step.
- **ActionType Sample** (const arma::colvec &actionValue, bool deterministic=false)
Sample an action based on given action values.

39.423.1 Detailed Description

```
template<typename PolicyType>
class mlpack::rl::AggregatedPolicy< PolicyType >
```

Template Parameters

<i>PolicyType</i>	The type of the child policy.
-------------------	-------------------------------

Definition at line 27 of file aggregated_policy.hpp.

39.423.2 Member Typedef Documentation

39.423.2.1 ActionType

```
using ActionType = typename PolicyType::ActionType
```

Convenient typedef for action.

Definition at line 31 of file aggregated_policy.hpp.

39.423.3 Constructor & Destructor Documentation

39.423.3.1 AggregatedPolicy()

```
AggregatedPolicy (  
    std::vector< PolicyType > policies,  
    const arma::colvec & distribution ) [inline]
```

Parameters

<i>policies</i>	Child policies.
<i>distribution</i>	Probability distribution for each child policy. User should make sure its size is same as the number of policies and the sum of its element is equal to 1.

Definition at line 39 of file aggregated_policy.hpp.

39.423.4 Member Function Documentation

39.423.4.1 Anneal()

```
void Anneal ( ) [inline]
```

Exploration probability will anneal at each step.

Definition at line 63 of file aggregated_policy.hpp.

39.423.4.2 Sample()

```
ActionType Sample (
    const arma::colvec & actionValue,
    bool deterministic = false ) [inline]
```

Sample an action based on given action values.

Parameters

<i>actionValue</i>	Values for each action.
<i>deterministic</i>	Always select the action greedily.

Returns

Sampled action.

Definition at line 52 of file aggregated_policy.hpp.

References `DiscreteDistribution::Random()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/policy/policy.hpp` **aggregated_↔**

39.424 AsyncLearning< WorkerType, EnvironmentType, NetworkType, UpdaterType, PolicyType > Class Template Reference

Wrapper of various asynchronous learning algorithms, e.g.

Public Member Functions

- **AsyncLearning** (**TrainingConfig** config, NetworkType network, PolicyType policy, UpdaterType updater=UpdaterType(), EnvironmentType environment=EnvironmentType())
Construct an instance of the given async learning algorithm.
- **TrainingConfig** & **Config** ()
Get training config.
- const **TrainingConfig** & **Config** () const
Modify training config.
- EnvironmentType & **Environment** ()
Get the environment.
- const EnvironmentType & **Environment** () const
Modify the environment.
- NetworkType & **Network** ()
Get learning network.
- const NetworkType & **Network** () const
Modify learning network.
- PolicyType & **Policy** ()
Get behavior policy.
- const PolicyType & **Policy** () const
Modify behavior policy.
- template<typename Measure >
void **Train** (Measure &measure)
Starting async training.
- UpdaterType & **Updater** ()
Get optimizer.
- const UpdaterType & **Updater** () const
Modify optimizer.

39.424.1 Detailed Description

```
template<typename WorkerType, typename EnvironmentType, typename NetworkType, typename UpdaterType, typename PolicyType>
class mlpack::rl::AsyncLearning< WorkerType, EnvironmentType, NetworkType, UpdaterType, PolicyType >
```

Wrapper of various asynchronous learning algorithms, e.g.

async one-step Q-learning, async one-step Sarsa, async n-step Q-learning and async advantage actor-critic.

For more details, see the following:

```
@inproceedings{mnih2016asynchronous,
  title   = {Asynchronous methods for deep reinforcement learning},
  author  = {Mnih, Volodymyr and Badia, Adria Puigdomenech and Mirza,
            Mehdi and Graves, Alex and Lillicrap, Timothy and Harley,
            Tim and Silver, David and Kavukcuoglu, Koray},
  booktitle = {International Conference on Machine Learning},
  pages   = {1928--1937},
  year    = {2016}
}
```

Template Parameters

<i>WorkerType</i>	The type of the worker.
<i>EnvironmentType</i>	The type of reinforcement learning task.
<i>NetworkType</i>	The type of the network model.
<i>UpdaterType</i>	The type of the optimizer.
<i>PolicyType</i>	The type of the behavior policy.

Definition at line 57 of file `async_learning.hpp`.

39.424.2 Constructor & Destructor Documentation

39.424.2.1 AsyncLearning()

```
AsyncLearning (
    TrainingConfig config,
    NetworkType network,
    PolicyType policy,
    UpdaterType updater = UpdaterType(),
    EnvironmentType environment = EnvironmentType() )
```

Construct an instance of the given async learning algorithm.

Parameters

<i>config</i>	Hyper-parameters for training.
<i>network</i>	The network model.
<i>policy</i>	The behavior policy.
<i>updater</i>	The optimizer.
<i>environment</i>	The reinforcement learning task.

39.424.3 Member Function Documentation

39.424.3.1 Config() [1/2]

```
TrainingConfig& Config ( ) [inline]
```

Get training config.

Definition at line 92 of file `async_learning.hpp`.

39.424.3.2 Config() [2/2]

```
const TrainingConfig& Config ( ) const [inline]
```

Modify training config.

Definition at line 94 of file `async_learning.hpp`.

39.424.3.3 Environment() [1/2]

```
EnvironmentType& Environment ( ) [inline]
```

Get the environment.

Definition at line 112 of file `async_learning.hpp`.

39.424.3.4 Environment() [2/2]

```
const EnvironmentType& Environment ( ) const [inline]
```

Modify the environment.

Definition at line 114 of file `async_learning.hpp`.

39.424.3.5 Network() [1/2]

```
NetworkType& Network ( ) [inline]
```

Get learning network.

Definition at line 97 of file `async_learning.hpp`.

39.424.3.6 Network() [2/2]

```
const NetworkType& Network ( ) const [inline]
```

Modify learning network.

Definition at line 99 of file `async_learning.hpp`.

39.424.3.7 Policy() [1/2]

```
PolicyType& Policy ( ) [inline]
```

Get behavior policy.

Definition at line 102 of file async_learning.hpp.

39.424.3.8 Policy() [2/2]

```
const PolicyType& Policy ( ) const [inline]
```

Modify behavior policy.

Definition at line 104 of file async_learning.hpp.

39.424.3.9 Train()

```
void Train (
    Measure & measure )
```

Starting async training.

Template Parameters

<i>Measure</i>	<p>The type of the measurement. It should be a callable object like</p> <pre>bool foo(double reward);</pre> <p>where reward is the total reward of a deterministic test episode, and the return value should indicate whether the training process is completed.</p>
----------------	--

Parameters

<i>measure</i>	The measurement instance.
----------------	---------------------------

39.424.3.10 Updater() [1/2]

```
UpdaterType& Updater ( ) [inline]
```

Get optimizer.

Definition at line 107 of file `async_learning.hpp`.

39.424.3.11 Updater() [2/2]

```
const UpdaterType& Updater ( ) const [inline]
```

Modify optimizer.

Definition at line 109 of file `async_learning.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/async_learning.hpp`

39.425 CartPole Class Reference

Implementation of Cart Pole task.

Classes

- class **State**
Implementation of the state of Cart Pole.

Public Types

- enum **Action** {
 backward,
 forward,
 size }
Implementation of action of Cart Pole.

Public Member Functions

- **CartPole** (const double gravity=9.8, const double massCart=1.0, const double massPole=0.1, const double length=0.5, const double forceMag=10.0, const double tau=0.02, const double thetaThresholdRadians=12 *2 *3.1416/360, const double xThreshold=2.4, const double doneReward=0.0)
Construct a Cart Pole instance using the given constants.
- **State InitialSample** () const
Initial state representation is randomly generated within [-0.05, 0.05].
- bool **IsTerminal** (const **State** &state) const
Whether given state is a terminal state.
- double **Sample** (const **State** &state, const **Action** &action, **State** &nextState) const
Dynamics of Cart Pole instance.
- double **Sample** (const **State** &state, const **Action** &action) const
Dynamics of Cart Pole.

39.425.1 Detailed Description

Implementation of Cart Pole task.

Definition at line 26 of file cart_pole.hpp.

39.425.2 Member Enumeration Documentation

39.425.2.1 Action

enum **Action**

Implementation of action of Cart Pole.

Enumerator

backward	
forward	
size	

Definition at line 87 of file cart_pole.hpp.

39.425.3 Constructor & Destructor Documentation

39.425.3.1 CartPole()

```
CartPole (  
    const double gravity = 9.8,  
    const double massCart = 1.0,  
    const double massPole = 0.1,  
    const double length = 0.5,  
    const double forceMag = 10.0,  
    const double tau = 0.02,  
    const double thetaThresholdRadians = 12 * 2 * 3.1416 / 360,  
    const double xThreshold = 2.4,  
    const double doneReward = 0.0 ) [inline]
```

Construct a Cart Pole instance using the given constants.

Parameters

<i>gravity</i>	The gravity constant.
<i>massCart</i>	The mass of the cart.
<i>massPole</i>	The mass of the pole.
<i>length</i>	The length of the pole.
<i>forceMag</i>	The magnitude of the applied force.
<i>tau</i>	The time interval.
<i>thetaThresholdRadians</i>	The maximum angle.
<i>xThreshold</i>	The maximum position.

Definition at line 108 of file cart_pole.hpp.

39.425.4 Member Function Documentation

39.425.4.1 InitialSample()

```
State InitialSample ( ) const [inline]
```

Initial state representation is randomly generated within [-0.05, 0.05].

Returns

Initial state for each episode.

Definition at line 192 of file cart_pole.hpp.

References `CartPole::State::State()`.

39.425.4.2 IsTerminal()

```
bool IsTerminal (
    const State & state ) const [inline]
```

Whether given state is a terminal state.

Parameters

<i>state</i>	The desired state.
--------------	--------------------

Returns

true if state is a terminal state, otherwise false.

Definition at line 203 of file cart_pole.hpp.

References `CartPole::State::Angle()`, and `CartPole::State::Position()`.

Referenced by `CartPole::Sample()`.

39.425.4.3 Sample() [1/2]

```
double Sample (
    const State & state,
    const Action & action,
    State & nextState ) const [inline]
```

Dynamics of Cart Pole instance.

Get reward and next state based on current state and current action.

Parameters

<i>state</i>	The current state.
<i>action</i>	The current action.
<i>nextState</i>	The next state.

Returns

reward, it's always 1.0.

It is important to note that if the cartpole is falling down, it should be penalized.

When done is false, it means that the cartpole has fallen down. For this case the reward is 1.0.

Definition at line 139 of file cart_pole.hpp.

References `CartPole::State::Angle()`, `CartPole::State::AngularVelocity()`, `CartPole::IsTerminal()`, `CartPole::State::Position()`, and `CartPole::State::Velocity()`.

Referenced by `CartPole::Sample()`.

39.425.4.4 Sample() [2/2]

```
double Sample (
    const State & state,
    const Action & action ) const [inline]
```

Dynamics of Cart Pole.

Get reward based on current state and current action.

Parameters

<i>state</i>	The current state.
<i>action</i>	The current action.

Returns

reward, it's always 1.0.

Definition at line 181 of file cart_pole.hpp.

References `CartPole::Sample()`.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/ **cart_pole.hpp**

39.426 CartPole::State Class Reference

Implementation of the state of Cart Pole.

Public Member Functions

- **State** ()
Construct a state instance.
- **State** (const arma::colvec &data)
Construct a state instance from given data.
- double **Angle** () const
Get the angle.
- double & **Angle** ()
Modify the angle.
- double **AngularVelocity** () const
Get the angular velocity.
- double & **AngularVelocity** ()
Modify the angular velocity.
- arma::colvec & **Data** ()
Modify the internal representation of the state.
- const arma::colvec & **Encode** () const
Encode the state to a column vector.
- double **Position** () const
Get the position.
- double & **Position** ()
Modify the position.
- double **Velocity** () const
Get the velocity.
- double & **Velocity** ()
Modify the velocity.

Static Public Attributes

- static constexpr size_t **dimension** = 4
Dimension of the encoded state.

39.426.1 Detailed Description

Implementation of the state of Cart Pole.

Each state is a tuple vector (position, velocity, angle, angular velocity).

Definition at line 33 of file cart_pole.hpp.

39.426.2 Constructor & Destructor Documentation

39.426.2.1 State() [1/2]

```
State ( ) [inline]
```

Construct a state instance.

Definition at line 39 of file cart_pole.hpp.

Referenced by CartPole::InitialSample().

39.426.2.2 State() [2/2]

```
State (
    const arma::colvec & data ) [inline]
```

Construct a state instance from given data.

Parameters

<i>data</i>	Data for the position, velocity, angle and angular velocity.
-------------	--

Definition at line 47 of file cart_pole.hpp.

39.426.3 Member Function Documentation

39.426.3.1 Angle() [1/2]

```
double Angle ( ) const [inline]
```

Get the angle.

Definition at line 64 of file cart_pole.hpp.

Referenced by CartPole::IsTerminal(), and CartPole::Sample().

39.426.3.2 Angle() [2/2]

```
double& Angle ( ) [inline]
```

Modify the angle.

Definition at line 66 of file cart_pole.hpp.

39.426.3.3 AngularVelocity() [1/2]

```
double AngularVelocity ( ) const [inline]
```

Get the angular velocity.

Definition at line 69 of file cart_pole.hpp.

Referenced by CartPole::Sample().

39.426.3.4 AngularVelocity() [2/2]

```
double& AngularVelocity ( ) [inline]
```

Modify the angular velocity.

Definition at line 71 of file cart_pole.hpp.

39.426.3.5 Data()

```
arma::colvec& Data ( ) [inline]
```

Modify the internal representation of the state.

Definition at line 51 of file cart_pole.hpp.

39.426.3.6 Encode()

```
const arma::colvec& Encode ( ) const [inline]
```

Encode the state to a column vector.

Definition at line 74 of file cart_pole.hpp.

39.426.3.7 Position() [1/2]

```
double Position ( ) const [inline]
```

Get the position.

Definition at line 54 of file cart_pole.hpp.

Referenced by CartPole::IsTerminal(), and CartPole::Sample().

39.426.3.8 Position() [2/2]

```
double& Position ( ) [inline]
```

Modify the position.

Definition at line 56 of file cart_pole.hpp.

39.426.3.9 Velocity() [1/2]

```
double Velocity ( ) const [inline]
```

Get the velocity.

Definition at line 59 of file cart_pole.hpp.

Referenced by CartPole::Sample().

39.426.3.10 Velocity() [2/2]

```
double& Velocity ( ) [inline]
```

Modify the velocity.

Definition at line 61 of file cart_pole.hpp.

39.426.4 Member Data Documentation**39.426.4.1 dimension**

```
constexpr size_t dimension = 4 [static]
```

Dimension of the encoded state.

Definition at line 77 of file cart_pole.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/ **cart_pole.hpp**

39.427 ContinuousMountainCar Class Reference

Implementation of Continuous Mountain Car task.

Classes

- struct **Action**
Implementation of action of Continuous Mountain Car.
- class **State**
Implementation of state of Continuous Mountain Car.

Public Member Functions

- **ContinuousMountainCar** (const double positionMin=-1.2, const double positionMax=0.6, const double positionGoal=0.45, const double velocityMin=-0.07, const double velocityMax=0.07, const double power=0.0015)
Construct a Continuous Mountain Car instance using the given constant.
- **State InitialSample** () const
Initial position is randomly generated within [-0.6, -0.4].
- bool **IsTerminal** (const **State** &state) const
Whether given state is a terminal state.
- double **Sample** (const **State** &state, const **Action** &action, **State** &nextState) const
Dynamics of Continuous Mountain Car.
- double **Sample** (const **State** &state, const **Action** &action) const
Dynamics of Continuous Mountain Car.

39.427.1 Detailed Description

Implementation of Continuous Mountain Car task.

Definition at line 28 of file continuous_mountain_car.hpp.

39.427.2 Constructor & Destructor Documentation

39.427.2.1 ContinuousMountainCar()

```
ContinuousMountainCar (  
    const double positionMin = -1.2,  
    const double positionMax = 0.6,  
    const double positionGoal = 0.45,  
    const double velocityMin = -0.07,  
    const double velocityMax = 0.07,  
    const double power = 0.0015 ) [inline]
```

Construct a Continuous Mountain Car instance using the given constant.

Parameters

<i>positionMin</i>	Minimum legal position.
<i>positionMax</i>	Maximum legal position.
<i>positionGoal</i>	Final target position.
<i>velocityMin</i>	Minimum legal velocity.
<i>velocityMax</i>	Maximum legal velocity.
<i>power</i>	Power generated by car.

Definition at line 101 of file continuous_mountain_car.hpp.

39.427.3 Member Function Documentation

39.427.3.1 InitialSample()

```
State InitialSample ( ) const [inline]
```

Initial position is randomly generated within [-0.6, -0.4].

Initial velocity is 0.

Returns

Initial state for each episode.

Definition at line 171 of file continuous_mountain_car.hpp.

References ContinuousMountainCar::State::Position(), mpack::math::Random(), and ContinuousMountainCar::State::Velocity().

39.427.3.2 IsTerminal()

```
bool IsTerminal (
    const State & state ) const [inline]
```

Whether given state is a terminal state.

Parameters

<i>state</i>	desired state.
--------------	----------------

Returns

true if state is a terminal state, otherwise false.

Definition at line 185 of file continuous_mountain_car.hpp.

References ContinuousMountainCar::State::Position().

Referenced by ContinuousMountainCar::Sample().

39.427.3.3 Sample() [1/2]

```
double Sample (
    const State & state,
    const Action & action,
    State & nextState ) const [inline]
```

Dynamics of Continuous Mountain Car.

Get reward and next state based on current state and current action.

Parameters

<i>state</i>	The current state.
<i>action</i>	The current action.
<i>nextState</i>	The next state.

Returns

reward, it's always -1.0.

Definition at line 124 of file continuous_mountain_car.hpp.

References ContinuousMountainCar::Action::action, ContinuousMountainCar::IsTerminal(), ContinuousMountainCar::State::Position(), and ContinuousMountainCar::State::Velocity().

Referenced by ContinuousMountainCar::Sample().

39.427.3.4 Sample() [2/2]

```
double Sample (
    const State & state,
    const Action & action ) const [inline]
```

Dynamics of Continuous Mountain Car.

Get reward based on current state and current action.

Parameters

<i>state</i>	The current state.
<i>action</i>	The current action.

Returns

reward, it's always -1.0.

Definition at line 159 of file continuous_mountain_car.hpp.

References ContinuousMountainCar::Sample().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/ **continuous_↵**
_mountain_car.hpp

39.428 ContinuousMountainCar::Action Struct Reference

Implementation of action of Continuous Mountain Car.

Public Attributes

- double **action** [1]
- const int **size** = 1

39.428.1 Detailed Description

Implementation of action of Continuous Mountain Car.

In Continuous mountain car gain, the action represents the force to be applied. This value is bounded in range -1.0 to 1.0. Unlike the simple mountain car environment, where action space has a discrete value, continuous mountain car has continuous action space value.

Definition at line 84 of file continuous_mountain_car.hpp.

39.428.2 Member Data Documentation

39.428.2.1 action

```
double action[1]
```

Definition at line 86 of file continuous_mountain_car.hpp.

Referenced by ContinuousMountainCar::Sample().

39.428.2.2 size

```
const int size = 1
```

Definition at line 88 of file continuous_mountain_car.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/ **continuous_↵**
_mountain_car.hpp

39.429 ContinuousMountainCar::State Class Reference

Implementation of state of Continuous Mountain Car.

Public Member Functions

- **State** ()
Construct a state instance.
- **State** (const arma::colvec &data)
Construct a state based on the given data.
- arma::colvec & **Data** ()
Modify the internal representation of the state.
- const arma::colvec & **Encode** () const
Encode the state to a column vector.
- double **Position** () const
Get the position.
- double & **Position** ()
Modify the position.
- double **Velocity** () const
Get the velocity.
- double & **Velocity** ()
Modify the velocity.

Static Public Attributes

- static constexpr size_t **dimension** = 2
Dimension of the encoded state.

39.429.1 Detailed Description

Implementation of state of Continuous Mountain Car.

Each state is a (velocity, position) vector.

Definition at line 35 of file continuous_mountain_car.hpp.

39.429.2 Constructor & Destructor Documentation

39.429.2.1 State() [1/2]

```
State ( ) [inline]
```

Construct a state instance.

Definition at line 41 of file continuous_mountain_car.hpp.

39.429.2.2 State() [2/2]

```
State (
    const arma::colvec & data ) [inline]
```

Construct a state based on the given data.

Parameters

<i>data</i>	Data for the velocity and position.
-------------	-------------------------------------

Definition at line 49 of file continuous_mountain_car.hpp.

39.429.3 Member Function Documentation

39.429.3.1 Data()

```
arma::colvec& Data ( ) [inline]
```

Modify the internal representation of the state.

Definition at line 53 of file continuous_mountain_car.hpp.

39.429.3.2 Encode()

```
const arma::colvec& Encode ( ) const [inline]
```

Encode the state to a column vector.

Definition at line 66 of file continuous_mountain_car.hpp.

39.429.3.3 Position() [1/2]

```
double Position ( ) const [inline]
```

Get the position.

Definition at line 61 of file continuous_mountain_car.hpp.

Referenced by ContinuousMountainCar::InitialSample(), ContinuousMountainCar::IsTerminal(), and ContinuousMountainCar::Sample().

39.429.3.4 Position() [2/2]

```
double& Position ( ) [inline]
```

Modify the position.

Definition at line 63 of file continuous_mountain_car.hpp.

39.429.3.5 Velocity() [1/2]

```
double Velocity ( ) const [inline]
```

Get the velocity.

Definition at line 56 of file continuous_mountain_car.hpp.

Referenced by ContinuousMountainCar::InitialSample(), and ContinuousMountainCar::Sample().

39.429.3.6 Velocity() [2/2]

```
double& Velocity ( ) [inline]
```

Modify the velocity.

Definition at line 58 of file continuous_mountain_car.hpp.

39.429.4 Member Data Documentation**39.429.4.1 dimension**

```
constexpr size_t dimension = 2 [static]
```

Dimension of the encoded state.

Definition at line 69 of file continuous_mountain_car.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/ **continuous_↵**
_mountain_car.hpp

39.430 GreedyPolicy< EnvironmentType > Class Template Reference

Implementation for epsilon greedy policy.

Public Types

- using **ActionType** = typename EnvironmentType::Action
Convenient typedef for action.

Public Member Functions

- **GreedyPolicy** (const double initialEpsilon, const size_t annealInterval, const double minEpsilon, const double decayRate=1.0)
Constructor for epsilon greedy policy class.
- void **Anneal** ()
Exploration probability will anneal at each step.
- const double & **Epsilon** () const
- **ActionType Sample** (const arma::colvec &actionValue, bool deterministic=false)
Sample an action based on given action values.

39.430.1 Detailed Description

```
template<typename EnvironmentType>
class mlpack::rl::GreedyPolicy< EnvironmentType >
```

Implementation for epsilon greedy policy.

In general we will select an action greedily based on the action value, however sometimes we will also randomly select an action to encourage exploration.

Template Parameters

<i>EnvironmentType</i>	The reinforcement learning task.
------------------------	----------------------------------

Definition at line 31 of file greedy_policy.hpp.

39.430.2 Member Typedef Documentation

39.430.2.1 ActionType

```
using ActionType = typename EnvironmentType::Action
```

Convenient typedef for action.

Definition at line 35 of file greedy_policy.hpp.

39.430.3 Constructor & Destructor Documentation

39.430.3.1 GreedyPolicy()

```
GreedyPolicy (  
    const double initialEpsilon,  
    const size_t annealInterval,  
    const double minEpsilon,  
    const double decayRate = 1.0 ) [inline]
```

Constructor for epsilon greedy policy class.

Parameters

<i>initialEpsilon</i>	The initial probability to explore (select a random action).
<i>annealInterval</i>	The steps during which the probability to explore will anneal.
<i>minEpsilon</i>	Epsilon will never be less than this value.
<i>decayRate</i>	How much to change the model in response to the estimated error each time the model weights are updated.

Definition at line 48 of file greedy_policy.hpp.

39.430.4 Member Function Documentation

39.430.4.1 Anneal()

```
void Anneal ( ) [inline]
```

Exploration probability will anneal at each step.

Definition at line 80 of file greedy_policy.hpp.

39.430.4.2 Epsilon()

```
const double& Epsilon ( ) const [inline]
```

Returns

Current possibility to explore.

Definition at line 89 of file greedy_policy.hpp.

39.430.4.3 Sample()

```
ActionType Sample (  
    const arma::colvec & actionValue,  
    bool deterministic = false ) [inline]
```

Sample an action based on given action values.

Parameters

<i>actionValue</i>	Values for each action.
<i>deterministic</i>	Always select the action greedily.

Returns

Sampled action.

Definition at line 64 of file greedy_policy.hpp.

References `mlpack::math::RandInt()`, and `mlpack::math::Random()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/policy/greedy_policy.hpp`

39.431 MountainCar Class Reference

Implementation of Mountain Car task.

Classes

- class **State**
Implementation of state of Mountain Car.

Public Types

- enum **Action** {
 backward,
 stop,
 forward,
 size }
Implementation of action of Mountain Car.

Public Member Functions

- **MountainCar** (const double positionMin=-1.2, const double positionMax=0.6, const double positionGoal=0.5, const double velocityMin=-0.07, const double velocityMax=0.07, const double doneReward=0)
Construct a Mountain Car instance using the given constant.
- **State InitialSample** () const
Initial position is randomly generated within [-0.6, -0.4].
- bool **IsTerminal** (const **State** &state) const
Whether given state is a terminal state.
- double **Sample** (const **State** &state, const **Action** &action, **State** &nextState) const
Dynamics of Mountain Car.
- double **Sample** (const **State** &state, const **Action** &action) const
Dynamics of Mountain Car.

39.431.1 Detailed Description

Implementation of Mountain Car task.

Definition at line 27 of file mountain_car.hpp.

39.431.2 Member Enumeration Documentation

39.431.2.1 Action

enum **Action**

Implementation of action of Mountain Car.

Enumerator

backward	
stop	
forward	
size	Track the size of the action space.

Definition at line 78 of file mountain_car.hpp.

39.431.3 Constructor & Destructor Documentation

39.431.3.1 MountainCar()

```
MountainCar (  
    const double positionMin = -1.2,  
    const double positionMax = 0.6,  
    const double positionGoal = 0.5,  
    const double velocityMin = -0.07,  
    const double velocityMax = 0.07,  
    const double doneReward = 0 ) [inline]
```

Construct a Mountain Car instance using the given constant.

Parameters

<i>positionMin</i>	Minimum legal position.
--------------------	-------------------------

Parameters

<i>positionMax</i>	Maximum legal position.
<i>positionGoal</i>	Final target position.
<i>velocityMin</i>	Minimum legal velocity.
<i>velocityMax</i>	Maximum legal velocity.

Definition at line 97 of file mountain_car.hpp.

39.431.4 Member Function Documentation

39.431.4.1 InitialSample()

```
State InitialSample ( ) const [inline]
```

Initial position is randomly generated within [-0.6, -0.4].

Initial velocity is 0.

Returns

Initial state for each episode.

Definition at line 172 of file mountain_car.hpp.

References MountainCar::State::Position(), and MountainCar::State::Velocity().

39.431.4.2 IsTerminal()

```
bool IsTerminal (
    const State & state ) const [inline]
```

Whether given state is a terminal state.

Parameters

<i>state</i>	desired state.
--------------	----------------

Returns

true if state is a terminal state, otherwise false.

Definition at line 186 of file mountain_car.hpp.

References MountainCar::State::Position().

Referenced by MountainCar::Sample().

39.431.4.3 Sample() [1/2]

```
double Sample (
    const State & state,
    const Action & action,
    State & nextState ) const [inline]
```

Dynamics of Mountain Car.

Get reward and next state based on current state and current action.

Parameters

<i>state</i>	The current state.
<i>action</i>	The current action.
<i>nextState</i>	The next state.

Returns

reward, it's always -1.0.

If done is true , it means that car has reached its goal. To make sure that the agent learns this, we will give some positive reward to the agent. If the agent doesn't reach the terminal state, then we will give a -1.0 reward to penalize the agent to take that step.

Definition at line 120 of file mountain_car.hpp.

References MountainCar::IsTerminal(), MountainCar::State::Position(), and MountainCar::State::Velocity().

Referenced by MountainCar::Sample().

39.431.4.4 Sample() [2/2]

```
double Sample (
    const State & state,
    const Action & action ) const [inline]
```

Dynamics of Mountain Car.

Get reward based on current state and current action.

Parameters

<i>state</i>	The current state.
<i>action</i>	The current action.

Returns

reward, it's always -1.0.

Definition at line 160 of file mountain_car.hpp.

References MountainCar::Sample().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/ **mountain_car.hpp**

39.432 MountainCar::State Class Reference

Implementation of state of Mountain Car.

Public Member Functions

- **State** ()
Construct a state instance.
- **State** (const arma::colvec &data)
Construct a state based on the given data.
- arma::colvec & **Data** ()
Modify the internal representation of the state.
- const arma::colvec & **Encode** () const
Encode the state to a column vector.
- double **Position** () const
Get the position.
- double & **Position** ()
Modify the position.
- double **Velocity** () const
Get the velocity.
- double & **Velocity** ()
Modify the velocity.

Static Public Attributes

- static constexpr size_t **dimension** = 2
Dimension of the encoded state.

39.432.1 Detailed Description

Implementation of state of Mountain Car.

Each state is a (velocity, position) vector.

Definition at line 34 of file mountain_car.hpp.

39.432.2 Constructor & Destructor Documentation

39.432.2.1 State() [1/2]

```
State ( ) [inline]
```

Construct a state instance.

Definition at line 40 of file mountain_car.hpp.

39.432.2.2 State() [2/2]

```
State (
    const arma::colvec & data ) [inline]
```

Construct a state based on the given data.

Parameters

<i>data</i>	Data for the velocity and position.
-------------	-------------------------------------

Definition at line 48 of file mountain_car.hpp.

39.432.3 Member Function Documentation

39.432.3.1 Data()

```
arma::colvec& Data ( ) [inline]
```

Modify the internal representation of the state.

Definition at line 52 of file mountain_car.hpp.

39.432.3.2 Encode()

```
const arma::colvec& Encode ( ) const [inline]
```

Encode the state to a column vector.

Definition at line 65 of file mountain_car.hpp.

39.432.3.3 Position() [1/2]

```
double Position ( ) const [inline]
```

Get the position.

Definition at line 60 of file mountain_car.hpp.

Referenced by MountainCar::InitialSample(), MountainCar::IsTerminal(), and MountainCar::Sample().

39.432.3.4 Position() [2/2]

```
double& Position ( ) [inline]
```

Modify the position.

Definition at line 62 of file mountain_car.hpp.

39.432.3.5 Velocity() [1/2]

```
double Velocity ( ) const [inline]
```

Get the velocity.

Definition at line 55 of file mountain_car.hpp.

Referenced by MountainCar::InitialSample(), and MountainCar::Sample().

39.432.3.6 Velocity() [2/2]

```
double& Velocity ( ) [inline]
```

Modify the velocity.

Definition at line 57 of file mountain_car.hpp.

39.432.4 Member Data Documentation**39.432.4.1 dimension**

```
constexpr size_t dimension = 2 [static]
```

Dimension of the encoded state.

Definition at line 68 of file mountain_car.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/ **mountain_↔
car.hpp**

39.433 NStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType > Class Template Reference

Forward declaration of **NStepQLearningWorker** (p. 1868).

Public Types

- using **ActionType** = typename EnvironmentType::Action
- using **StateType** = typename EnvironmentType::State
- using **TransitionType** = std::tuple< **StateType**, **ActionType**, double, **StateType** >

Public Member Functions

- **NStepQLearningWorker** (const UpdaterType &updater, const EnvironmentType &environment, const **TrainingConfig** &config, bool deterministic)
Construct N-step Q-Learning worker with the given parameters and environment.
- void **Initialize** (NetworkType &learningNetwork)
Initialize the worker.
- bool **Step** (NetworkType &learningNetwork, NetworkType &targetNetwork, size_t &totalSteps, PolicyType &policy, double &totalReward)
The agent will execute one step.

39.433.1 Detailed Description

```
template<typename EnvironmentType, typename NetworkType, typename UpdaterType, typename PolicyType>
class mlpack::rl::NStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >
```

Forward declaration of **NStepQLearningWorker** (p. 1868).

N-step Q-Learning worker.

Template Parameters

<i>EnvironmentType</i>	The type of the reinforcement learning task.
<i>NetworkType</i>	The type of the network model.
<i>UpdaterType</i>	The type of the optimizer.
<i>PolicyType</i>	The type of the behavior policy.

Definition at line 179 of file `async_learning.hpp`.

39.433.2 Member Typedef Documentation

39.433.2.1 ActionType

```
using ActionType = typename EnvironmentType::Action
```

Definition at line 39 of file `n_step_q_learning_worker.hpp`.

39.433.2.2 StateType

```
using StateType = typename EnvironmentType::State
```

Definition at line 38 of file `n_step_q_learning_worker.hpp`.

39.433.2.3 TransitionType

```
using TransitionType = std::tuple< StateType, ActionType, double, StateType>
```

Definition at line 40 of file `n_step_q_learning_worker.hpp`.

39.433.3 Constructor & Destructor Documentation

39.433.3.1 NStepQLearningWorker()

```
NStepQLearningWorker (  
    const UpdaterType & updater,  
    const EnvironmentType & environment,  
    const TrainingConfig & config,  
    bool deterministic ) [inline]
```

Construct N-step Q-Learning worker with the given parameters and environment.

Parameters

<i>updater</i>	The optimizer.
<i>environment</i>	The reinforcement learning task.
<i>config</i>	Hyper-parameters.
<i>deterministic</i>	Whether it should be deterministic.

Definition at line 51 of file `n_step_q_learning_worker.hpp`.

39.433.4 Member Function Documentation

39.433.4.1 Initialize()

```
void Initialize (  
    NetworkType & learningNetwork ) [inline]
```

Initialize the worker.

Parameters

<i>learningNetwork</i>	The shared network.
------------------------	---------------------

Definition at line 67 of file `n_step_q_learning_worker.hpp`.

39.433.4.2 Step()

```
bool Step (
    NetworkType & learningNetwork,
    NetworkType & targetNetwork,
    size_t & totalSteps,
    PolicyType & policy,
    double & totalReward ) [inline]
```

The agent will execute one step.

Parameters

<i>learningNetwork</i>	The shared learning network.
<i>targetNetwork</i>	The shared target network.
<i>totalSteps</i>	The shared counter for total steps.
<i>policy</i>	The shared behavior policy.
<i>totalReward</i>	This will be the episode return if the episode ends after this step. Otherwise this is invalid.

Returns

Indicate whether current episode ends after this step.

Definition at line 86 of file `n_step_q_learning_worker.hpp`.

References `TrainingConfig::Discount()`, `TrainingConfig::GradientLimit()`, `TrainingConfig::StepLimit()`, `TrainingConfig::↵StepSize()`, `TrainingConfig::TargetNetworkSyncInterval()`, and `TrainingConfig::UpdateInterval()`.

The documentation for this class was generated from the following files:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/ async_learning.hpp`
- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/ n_step_q_↵learning_worker.hpp`

39.434 OneStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, Policy↵Type > Class Template Reference

Forward declaration of **OneStepQLearningWorker** (p. 1871).

Public Types

- using **ActionType** = typename EnvironmentType::Action
- using **StateType** = typename EnvironmentType::State
- using **TransitionType** = std::tuple< **StateType**, **ActionType**, double, **StateType** >

Public Member Functions

- **OneStepQLearningWorker** (const UpdaterType &updater, const EnvironmentType &environment, const TrainingConfig &config, bool deterministic)
Construct one step Q-Learning worker with the given parameters and environment.
- void **Initialize** (NetworkType &learningNetwork)
Initialize the worker.
- bool **Step** (NetworkType &learningNetwork, NetworkType &targetNetwork, size_t &totalSteps, PolicyType &policy, double &totalReward)
The agent will execute one step.

39.434.1 Detailed Description

```
template<typename EnvironmentType, typename NetworkType, typename UpdaterType, typename PolicyType>
class mlpack::rl::OneStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >
```

Forward declaration of **OneStepQLearningWorker** (p. 1871).

One step Q-Learning worker.

Template Parameters

<i>EnvironmentType</i>	The type of the reinforcement learning task.
<i>NetworkType</i>	The type of the network model.
<i>UpdaterType</i>	The type of the optimizer.
<i>PolicyType</i>	The type of the behavior policy.
<i>EnvironmentType</i>	The type of the reinforcement learning task.
<i>NetworkType</i>	The type of the network model.
<i>UpdaterType</i>	The type of the optimizer.
<i>PolicyType</i>	The type of the behavior policy. *

Definition at line 147 of file `async_learning.hpp`.

39.434.2 Member Typedef Documentation

39.434.2.1 ActionType

```
using ActionType = typename EnvironmentType::Action
```

Definition at line 39 of file `one_step_q_learning_worker.hpp`.

39.434.2.2 StateType

```
using StateType = typename EnvironmentType::State
```

Definition at line 38 of file one_step_q_learning_worker.hpp.

39.434.2.3 TransitionType

```
using TransitionType = std::tuple< StateType, ActionType, double, StateType>
```

Definition at line 40 of file one_step_q_learning_worker.hpp.

39.434.3 Constructor & Destructor Documentation

39.434.3.1 OneStepQLearningWorker()

```
OneStepQLearningWorker (  
    const UpdaterType & updater,  
    const EnvironmentType & environment,  
    const TrainingConfig & config,  
    bool deterministic ) [inline]
```

Construct one step Q-Learning worker with the given parameters and environment.

Parameters

<i>updater</i>	The optimizer.
<i>environment</i>	The reinforcement learning task.
<i>config</i>	Hyper-parameters.
<i>deterministic</i>	Whether it should be deterministic.

Definition at line 51 of file one_step_q_learning_worker.hpp.

39.434.4 Member Function Documentation

39.434.4.1 Initialize()

```
void Initialize (
    NetworkType & learningNetwork ) [inline]
```

Initialize the worker.

Parameters

<i>learningNetwork</i>	The shared network.
------------------------	---------------------

Definition at line 67 of file one_step_q_learning_worker.hpp.

39.434.4.2 Step()

```
bool Step (
    NetworkType & learningNetwork,
    NetworkType & targetNetwork,
    size_t & totalSteps,
    PolicyType & policy,
    double & totalReward ) [inline]
```

The agent will execute one step.

Parameters

<i>learningNetwork</i>	The shared learning network.
<i>targetNetwork</i>	The shared target network.
<i>totalSteps</i>	The shared counter for total steps.
<i>policy</i>	The shared behavior policy.
<i>totalReward</i>	This will be the episode return if the episode ends after this step. Otherwise this is invalid.

Returns

Indicate whether current episode ends after this step.

Definition at line 86 of file one_step_q_learning_worker.hpp.

References [TrainingConfig::Discount\(\)](#), [TrainingConfig::GradientLimit\(\)](#), [TrainingConfig::StepLimit\(\)](#), [TrainingConfig::↔StepSize\(\)](#), [TrainingConfig::TargetNetworkSyncInterval\(\)](#), and [TrainingConfig::UpdateInterval\(\)](#).

The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/ **async_learning.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/ **one_step_q↔learning_worker.hpp**

39.435 OneStepSarsaWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType > Class Template Reference

Forward declaration of **OneStepSarsaWorker** (p. 1875).

Public Types

- using **ActionType** = typename EnvironmentType::Action
- using **StateType** = typename EnvironmentType::State
- using **TransitionType** = std::tuple< **StateType**, **ActionType**, double, **StateType**, **ActionType** >

Public Member Functions

- **OneStepSarsaWorker** (const UpdaterType &updater, const EnvironmentType &environment, const **TrainingConfig** &config, bool deterministic)
Construct one step sarsa worker with the given parameters and environment.
- void **Initialize** (NetworkType &learningNetwork)
Initialize the worker.
- bool **Step** (NetworkType &learningNetwork, NetworkType &targetNetwork, size_t &totalSteps, PolicyType &policy, double &totalReward)
The agent will execute one step.

39.435.1 Detailed Description

```
template<typename EnvironmentType, typename NetworkType, typename UpdaterType, typename PolicyType>
class mlpack::rl::OneStepSarsaWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >
```

Forward declaration of **OneStepSarsaWorker** (p. 1875).

One step Sarsa worker.

Template Parameters

<i>EnvironmentType</i>	The type of the reinforcement learning task.
<i>NetworkType</i>	The type of the network model.
<i>UpdaterType</i>	The type of the optimizer.
<i>PolicyType</i>	The type of the behavior policy.

Definition at line 163 of file `async_learning.hpp`.

39.435.2 Member Typedef Documentation

39.435.2.1 ActionType

```
using ActionType = typename EnvironmentType::Action
```

Definition at line 39 of file one_step_sarsa_worker.hpp.

39.435.2.2 StateType

```
using StateType = typename EnvironmentType::State
```

Definition at line 38 of file one_step_sarsa_worker.hpp.

39.435.2.3 TransitionType

```
using TransitionType = std::tuple< StateType, ActionType, double, StateType, ActionType>
```

Definition at line 41 of file one_step_sarsa_worker.hpp.

39.435.3 Constructor & Destructor Documentation

39.435.3.1 OneStepSarsaWorker()

```
OneStepSarsaWorker (
    const UpdaterType & updater,
    const EnvironmentType & environment,
    const TrainingConfig & config,
    bool deterministic ) [inline]
```

Construct one step sarsa worker with the given parameters and environment.

Parameters

<i>updater</i>	The optimizer.
<i>environment</i>	The reinforcement learning task.
<i>config</i>	Hyper-parameters.
<i>deterministic</i>	Whether it should be deterministic.

Definition at line 52 of file one_step_sarsa_worker.hpp.

39.435.4 Member Function Documentation

39.435.4.1 Initialize()

```
void Initialize (
    NetworkType & learningNetwork ) [inline]
```

Initialize the worker.

Parameters

<i>learningNetwork</i>	The shared network.
------------------------	---------------------

Definition at line 68 of file one_step_sarsa_worker.hpp.

39.435.4.2 Step()

```
bool Step (
    NetworkType & learningNetwork,
    NetworkType & targetNetwork,
    size_t & totalSteps,
    PolicyType & policy,
    double & totalReward ) [inline]
```

The agent will execute one step.

Parameters

<i>learningNetwork</i>	The shared learning network.
<i>targetNetwork</i>	The shared target network.
<i>totalSteps</i>	The shared counter for total steps.
<i>policy</i>	The shared behavior policy.
<i>totalReward</i>	This will be the episode return if the episode ends after this step. Otherwise this is invalid.

Returns

Indicate whether current episode ends after this step.

Definition at line 87 of file one_step_sarsa_worker.hpp.

References TrainingConfig::Discount(), TrainingConfig::GradientLimit(), TrainingConfig::StepLimit(), TrainingConfig::StepSize(), TrainingConfig::TargetNetworkSyncInterval(), and TrainingConfig::UpdateInterval().

The documentation for this class was generated from the following files:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/ async_learning.hpp`
- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/ one_step_sarsa_↵
worker.hpp`

39.436 Pendulum Class Reference

Implementation of **Pendulum** (p. 1878) task.

Classes

- struct **Action**
*Implementation of action of **Pendulum** (p. 1878).*
- class **State**
*Implementation of state of **Pendulum** (p. 1878).*

Public Member Functions

- **Pendulum** (const double maxAngularVelocity=8, const double maxTorque=2.0, const double dt=0.05)
*Construct a **Pendulum** (p. 1878) instance using the given values.*
- double **AngleNormalize** (double theta) const
This function calculates the normalized angle for a particular theta.
- **State InitialSample** () const
Initial theta is randomly generated within $[-\pi, \pi]$.
- double **Sample** (const **State** &state, const **Action** &action, **State** &nextState) const
*Dynamics of **Pendulum** (p. 1878).*
- double **Sample** (const **State** &state, const **Action** &action) const
*Dynamics of **Pendulum** (p. 1878).*

39.436.1 Detailed Description

Implementation of **Pendulum** (p. 1878) task.

The inverted pendulum swingup problem is a classic problem in the control literature. In this version of the problem, the pendulum starts in a random position, and the goal is to swing it up so it stays upright

Definition at line 30 of file pendulum.hpp.

39.436.2 Constructor & Destructor Documentation

39.436.2.1 Pendulum()

```
Pendulum (
    const double maxAngularVelocity = 8,
    const double maxTorque = 2.0,
    const double dt = 0.05 ) [inline]
```

Construct a **Pendulum** (p. 1878) instance using the given values.

Parameters

<i>maxAngularVelocity</i>	Maximum angular velocity.
<i>maxTorque</i>	Maximum torque.
<i>dt</i>	The differential value.

Definition at line 97 of file pendulum.hpp.

39.436.3 Member Function Documentation

39.436.3.1 AngleNormalize()

```
double AngleNormalize (  
    double theta ) const [inline]
```

This function calculates the normalized anlge for a particular theta.

Parameters

<i>theta</i>	The un-normalized angle.
--------------	--------------------------

Definition at line 180 of file pendulum.hpp.

References `M_PI`.

Referenced by `Pendulum::Sample()`.

39.436.3.2 InitialSample()

```
State InitialSample ( ) const [inline]
```

Initial theta is randomly generated within $[-\pi, \pi]$.

Initial angular velocity is randomly generated within $[-1, 1]$.

Returns

Initial state for each episode.

Definition at line 167 of file pendulum.hpp.

References `Pendulum::State::AngularVelocity()`, `M_PI`, `mlpack::math::Random()`, and `Pendulum::State::Theta()`.

39.436.3.3 Sample() [1/2]

```
double Sample (
    const State & state,
    const Action & action,
    State & nextState ) const [inline]
```

Dynamics of **Pendulum** (p. 1878).

Get reward and next state based on current state and current action.

Parameters

<i>state</i>	The current state.
<i>action</i>	The current action.
<i>nextState</i>	The next state.

Returns

reward, The reward for taking the action taken for current state.

Definition at line 114 of file pendulum.hpp.

References `Pendulum::Action::action`, `Pendulum::AngleNormalize()`, `Pendulum::State::AngularVelocity()`, `M_PI`, and `Pendulum::State::Theta()`.

Referenced by `Pendulum::Sample()`.

39.436.3.4 Sample() [2/2]

```
double Sample (
    const State & state,
    const Action & action ) const [inline]
```

Dynamics of **Pendulum** (p. 1878).

Get reward based on current state and current action

Parameters

<i>state</i>	The current state.
<i>action</i>	The current action.

Returns

reward, The reward.

Definition at line 155 of file pendulum.hpp.

References Pendulum::Sample().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/ **pendulum.hpp**

39.437 Pendulum::Action Struct Reference

Implementation of action of **Pendulum** (p. 1878).

Public Attributes

- double **action** [1]
- const int **size** = 1

39.437.1 Detailed Description

Implementation of action of **Pendulum** (p. 1878).

In **Pendulum** (p. 1878), the action represents the torque to be applied. This value is bounded in range -2.0 to 2.0 by default.

Definition at line 83 of file pendulum.hpp.

39.437.2 Member Data Documentation

39.437.2.1 action

```
double action[1]
```

Definition at line 85 of file pendulum.hpp.

Referenced by Pendulum::Sample().

39.437.2.2 size

```
const int size = 1
```

Definition at line 87 of file pendulum.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/ **pendulum.hpp**

39.438 Pendulum::State Class Reference

Implementation of state of **Pendulum** (p. 1878).

Public Member Functions

- **State** ()
Construct a state instance.
- **State** (const arma::colvec &data)
Construct a state based on the given data.
- double **AngularVelocity** () const
Get the angular velocity.
- double & **AngularVelocity** ()
Modify the value of angular velocity.
- arma::colvec & **Data** ()
Modify the internal representation of the state.
- const arma::colvec & **Encode** () const
Encode the state to a column vector.
- double **Theta** () const
Get the theta.
- double & **Theta** ()
Modify the value of theta.

Static Public Attributes

- static constexpr size_t **dimension** = 2
Dimension of the encoded state.

39.438.1 Detailed Description

Implementation of state of **Pendulum** (p. 1878).

Each state is a (theta, angular velocity) vector.

Definition at line 37 of file pendulum.hpp.

39.438.2 Constructor & Destructor Documentation

39.438.2.1 State() [1/2]

```
State ( ) [inline]
```

Construct a state instance.

Definition at line 43 of file pendulum.hpp.

39.438.2.2 State() [2/2]

```
State (
    const arma::colvec & data ) [inline]
```

Construct a state based on the given data.

Parameters

<i>data</i>	Data for the theta and angular velocity.
-------------	--

Definition at line 51 of file pendulum.hpp.

39.438.3 Member Function Documentation

39.438.3.1 AngularVelocity() [1/2]

```
double AngularVelocity ( ) const [inline]
```

Get the angular velocity.

Definition at line 63 of file pendulum.hpp.

Referenced by Pendulum::InitialSample(), and Pendulum::Sample().

39.438.3.2 AngularVelocity() [2/2]

```
double& AngularVelocity ( ) [inline]
```

Modify the value of angular velocity.

Definition at line 65 of file pendulum.hpp.

39.438.3.3 Data()

```
arma::colvec& Data ( ) [inline]
```

Modify the internal representation of the state.

Definition at line 55 of file pendulum.hpp.

39.438.3.4 Encode()

```
const arma::colvec& Encode ( ) const [inline]
```

Encode the state to a column vector.

Definition at line 68 of file pendulum.hpp.

39.438.3.5 Theta() [1/2]

```
double Theta ( ) const [inline]
```

Get the theta.

Definition at line 58 of file pendulum.hpp.

Referenced by Pendulum::InitialSample(), and Pendulum::Sample().

39.438.3.6 Theta() [2/2]

```
double& Theta ( ) [inline]
```

Modify the value of theta.

Definition at line 60 of file pendulum.hpp.

39.438.4 Member Data Documentation

39.438.4.1 dimension

```
constexpr size_t dimension = 2 [static]
```

Dimension of the encoded state.

Definition at line 71 of file pendulum.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/ **pendulum.hpp**

39.439 QLearning< EnvironmentType, NetworkType, UpdaterType, PolicyType, ReplayType > Class Template Reference

Implementation of various Q-Learning algorithms, such as DQN, double DQN.

Public Types

- using **ActionType** = typename EnvironmentType::Action
Convenient typedef for action.
- using **StateType** = typename EnvironmentType::State
Convenient typedef for state.

Public Member Functions

- **QLearning** (**TrainingConfig** config, NetworkType network, PolicyType policy, ReplayType replayMethod, UpdaterType updater=UpdaterType(), EnvironmentType environment=EnvironmentType())
*Create the **QLearning** (p. 1885) object with given settings.*
- bool & **Deterministic** ()
Modify the training mode / test mode indicator.
- const bool & **Deterministic** () const
Get the indicator of training mode / test mode.
- EnvironmentType & **Environment** ()
Modify the environment in which the agent is.
- const EnvironmentType & **Environment** () const
Get the environment in which the agent is.
- double **Episode** ()
Execute an episode.

- `const NetworkType & Network () const`
Return the learning network.
- `NetworkType & Network ()`
Modify the learning network.
- `StateType & State ()`
Modify the state of the agent.
- `const StateType & State () const`
Get the state of the agent.
- `double Step ()`
Execute a step in an episode.
- `const size_t & TotalSteps () const`

39.439.1 Detailed Description

```
template<typename EnvironmentType, typename NetworkType, typename UpdaterType, typename PolicyType, typename ReplayType
= RandomReplay<EnvironmentType>>
class mlpack::rl::QLearning< EnvironmentType, NetworkType, UpdaterType, PolicyType, ReplayType >
```

Implementation of various Q-Learning algorithms, such as DQN, double DQN.

For more details, see the following:

```
@article{Mnih2013,
  author    = {Volodymyr Mnih and
              Koray Kavukcuoglu and
              David Silver and
              Alex Graves and
              Ioannis Antonoglou and
              Daan Wierstra and
              Martin A. Riedmiller},
  title     = {Playing Atari with Deep Reinforcement Learning},
  journal   = {CoRR},
  year      = {2013},
  url       = {http://arxiv.org/abs/1312.5602}
}
```

Template Parameters

<i>EnvironmentType</i>	The environment of the reinforcement learning task.
<i>NetworkType</i>	The network to compute action value.
<i>UpdaterType</i>	How to apply gradients when training.
<i>PolicyType</i>	Behavior policy of the agent.
<i>ReplayType</i>	Experience replay method.

Definition at line 57 of file `q_learning.hpp`.

39.439.2 Member Typedef Documentation

39.439.2.1 ActionType

```
using ActionType = typename EnvironmentType::Action
```

Convenient typedef for action.

Definition at line 64 of file q_learning.hpp.

39.439.2.2 StateType

```
using StateType = typename EnvironmentType::State
```

Convenient typedef for state.

Definition at line 61 of file q_learning.hpp.

39.439.3 Constructor & Destructor Documentation

39.439.3.1 QLearning()

```
QLearning (
    TrainingConfig config,
    NetworkType network,
    PolicyType policy,
    ReplayType replayMethod,
    UpdaterType updater = UpdaterType(),
    EnvironmentType environment = EnvironmentType() )
```

Create the **QLearning** (p. 1885) object with given settings.

If you want to pass in a parameter and discard the original parameter object, be sure to use `std::move` to avoid unnecessary copy.

Parameters

<i>config</i>	Hyper-parameters for training.
<i>network</i>	The network to compute action value.
<i>policy</i>	Behavior policy of the agent.
<i>replayMethod</i>	Experience replay method.
<i>updater</i>	How to apply gradients when training.
<i>environment</i>	Reinforcement learning task.

39.439.4 Member Function Documentation

39.439.4.1 Deterministic() [1/2]

```
bool& Deterministic ( ) [inline]
```

Modify the training mode / test mode indicator.

Definition at line 114 of file q_learning.hpp.

39.439.4.2 Deterministic() [2/2]

```
const bool& Deterministic ( ) const [inline]
```

Get the indicator of training mode / test mode.

Definition at line 116 of file q_learning.hpp.

39.439.4.3 Environment() [1/2]

```
EnvironmentType& Environment ( ) [inline]
```

Modify the environment in which the agent is.

Definition at line 109 of file q_learning.hpp.

39.439.4.4 Environment() [2/2]

```
const EnvironmentType& Environment ( ) const [inline]
```

Get the environment in which the agent is.

Definition at line 111 of file q_learning.hpp.

39.439.4.5 Episode()

```
double Episode ( )
```

Execute an episode.

Returns

Return of the episode.

39.439.4.6 Network() [1/2]

```
const NetworkType& Network ( ) const [inline]
```

Return the learning network.

Definition at line 119 of file q_learning.hpp.

39.439.4.7 Network() [2/2]

```
NetworkType& Network ( ) [inline]
```

Modify the learning network.

Definition at line 121 of file q_learning.hpp.

39.439.4.8 State() [1/2]

```
StateType& State ( ) [inline]
```

Modify the state of the agent.

Definition at line 104 of file q_learning.hpp.

39.439.4.9 State() [2/2]

```
const StateType& State ( ) const [inline]
```

Get the state of the agent.

Definition at line 106 of file `q_learning.hpp`.

39.439.4.10 Step()

```
double Step ( )
```

Execute a step in an episode.

Returns

Reward for the step.

39.439.4.11 TotalSteps()

```
const size_t& TotalSteps ( ) const [inline]
```

Returns

Total steps from beginning.

Definition at line 101 of file `q_learning.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/ q_learning.hpp`

39.440 RandomReplay< EnvironmentType > Class Template Reference

Implementation of random experience replay.

Public Types

- using **ActionType** = typename EnvironmentType::Action
Convenient typedef for action.
- using **StateType** = typename EnvironmentType::State
Convenient typedef for state.

Public Member Functions

- **RandomReplay** (const size_t batchSize, const size_t capacity, const size_t dimension=StateType::dimension)
Construct an instance of random experience replay class.
- void **Sample** (arma::mat &sampledStates, arma::icolvec &sampledActions, arma::colvec &sampledRewards, arma::mat &sampledNextStates, arma::icolvec &isTerminal)
Sample some experiences.
- const size_t & **Size** ()
Get the number of transitions in the memory.
- void **Store** (const **StateType** &state, **ActionType** action, double reward, const **StateType** &nextState, bool isEnd)
Store the given experience.

39.440.1 Detailed Description

```
template<typename EnvironmentType>
class mlpack::rl::RandomReplay< EnvironmentType >
```

Implementation of random experience replay.

At each time step, interactions between the agent and the environment will be saved to a memory buffer. When necessary, we can simply sample previous experiences from the buffer to train the agent. Typically this would be a random sample and the memory will be a First-In-First-Out buffer.

For more information, see the following.

```
@phdthesis{lin1993reinforcement,
  title = {Reinforcement learning for robots using neural networks},
  author = {Lin, Long-Ji},
  year = {1993},
  school = {Fujitsu Laboratories Ltd}
}
```

Template Parameters

<i>EnvironmentType</i>	Desired task.
------------------------	---------------

Definition at line 43 of file random_replay.hpp.

39.440.2 Member Typedef Documentation

39.440.2.1 ActionType

```
using ActionType = typename EnvironmentType::Action
```

Convenient typedef for action.

Definition at line 47 of file random_replay.hpp.

39.440.2.2 StateType

```
using StateType = typename EnvironmentType::State
```

Convenient typedef for state.

Definition at line 50 of file random_replay.hpp.

39.440.3 Constructor & Destructor Documentation

39.440.3.1 RandomReplay()

```
RandomReplay (
    const size_t batchSize,
    const size_t capacity,
    const size_t dimension = StateType::dimension ) [inline]
```

Construct an instance of random experience replay class.

Parameters

<i>batchSize</i>	Number of examples returned at each sample.
<i>capacity</i>	Total memory size in terms of number of examples.
<i>dimension</i>	The dimension of an encoded state.

Definition at line 59 of file random_replay.hpp.

39.440.4 Member Function Documentation

39.440.4.1 Sample()

```
void Sample (
    arma::mat & sampledStates,
```

```

        arma::icolvec & sampledActions,
        arma::colvec & sampledRewards,
        arma::mat & sampledNextStates,
        arma::icolvec & isTerminal ) [inline]

```

Sample some experiences.

Parameters

<i>sampledStates</i>	Sampled encoded states.
<i>sampledActions</i>	Sampled actions.
<i>sampledRewards</i>	Sampled rewards.
<i>sampledNextStates</i>	Sampled encoded next states.
<i>isTerminal</i>	Indicate whether corresponding next state is terminal state.

Definition at line 111 of file random_replay.hpp.

39.440.4.2 Size()

```
const size_t& Size ( ) [inline]
```

Get the number of transitions in the memory.

Returns

Actual used memory size

Definition at line 133 of file random_replay.hpp.

39.440.4.3 Store()

```

void Store (
    const StateType & state,
    ActionType action,
    double reward,
    const StateType & nextState,
    bool isEnd ) [inline]

```

Store the given experience.

Parameters

<i>state</i>	Given state.
<i>action</i>	Given action.
<i>reward</i>	Given reward.
<i>nextState</i>	Given next state.
<i>isEnd</i>	Whether next state is terminal state.

Definition at line 82 of file random_replay.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/replay/ **random_replay.hpp**

39.441 **RewardClipping**< **EnvironmentType** > Class Template Reference

Interface for clipping the reward to some value between the specified maximum and minimum value (Clipping here is implemented as $g_{\text{clipped}} = \max(g_{\text{min}}, \min(g_{\text{min}}, g))$.)

Public Types

- using **Action** = typename **EnvironmentType**::Action
Convenient typedef for action.
- using **State** = typename **EnvironmentType**::State
Convenient typedef for state.

Public Member Functions

- **RewardClipping** (**EnvironmentType** &environment, const double minReward=-1.0, const double maxReward=1.0)
*Constructor for creating a **RewardClipping** (p. 1894) instance.*
- **EnvironmentType** & **Environment** () const
Get the environment.
- **EnvironmentType** & **Environment** ()
Modify the environment.
- **State** **InitialSample** ()
The InitialSample method is called by the environment to initialize the starting state.
- bool **IsTerminal** (const **State** &state) const
Checks whether given state is a terminal state.
- double **MaxReward** () const
Get the maximum reward value.
- double & **MaxReward** ()
Modify the maximum reward value.
- double **MinReward** () const
Get the minimum reward value.
- double & **MinReward** ()
Modify the minimum reward value.
- double **Sample** (const **State** &state, const **Action** &action, **State** &nextState) const
Dynamics of Environment.
- double **Sample** (const **State** &state, const **Action** &action) const
Dynamics of Environment.

39.441.1 Detailed Description

```
template<typename EnvironmentType>  
class mlpack::rl::RewardClipping< EnvironmentType >
```

Interface for clipping the reward to some value between the specified maximum and minimum value (Clipping here is implemented as $g_{\text{clipped}} = \max(g_{\text{min}}, \min(g_{\text{min}}, g))$.)

Template Parameters

<i>EnvironmentType</i>	A type of Environment that is being wrapped.
------------------------	--

Definition at line 29 of file reward_clipping.hpp.

39.441.2 Member Typedef Documentation

39.441.2.1 Action

```
using Action = typename EnvironmentType::Action
```

Convenient typedef for action.

Definition at line 36 of file reward_clipping.hpp.

39.441.2.2 State

```
using State = typename EnvironmentType::State
```

Convenient typedef for state.

Definition at line 33 of file reward_clipping.hpp.

39.441.3 Constructor & Destructor Documentation

39.441.3.1 RewardClipping()

```
RewardClipping (  
    EnvironmentType & environment,  
    const double minReward = -1.0,  
    const double maxReward = 1.0 ) [inline]
```

Constructor for creating a **RewardClipping** (p. 1894) instance.

Parameters

<i>minReward</i>	Minimum possible value of clipped reward.
<i>maxReward</i>	Maximum possible value of clipped reward.
<i>environment</i>	An instance of the environment used for actual simulations.

Definition at line 46 of file reward_clipping.hpp.

39.441.4 Member Function Documentation

39.441.4.1 Environment() [1/2]

```
EnvironmentType& Environment ( ) const [inline]
```

Get the environment.

Definition at line 112 of file reward_clipping.hpp.

39.441.4.2 Environment() [2/2]

```
EnvironmentType& Environment ( ) [inline]
```

Modify the environment.

Definition at line 114 of file reward_clipping.hpp.

39.441.4.3 InitialSample()

```
State InitialSample ( ) [inline]
```

The InitialSample method is called by the environment to initialize the starting state.

Returns whatever Initial Sample is returned by the environment.

Definition at line 61 of file reward_clipping.hpp.

39.441.4.4 IsTerminal()

```
bool IsTerminal (
    const State & state ) const [inline]
```

Checks whether given state is a terminal state.

Returns the value by calling the environment method.

Parameters

<i>state</i>	desired state.
--------------	----------------

Returns

true if state is a terminal state, otherwise false.

Definition at line 73 of file reward_clipping.hpp.

39.441.4.5 MaxReward() [1/2]

```
double MaxReward ( ) const [inline]
```

Get the maximum reward value.

Definition at line 122 of file reward_clipping.hpp.

39.441.4.6 MaxReward() [2/2]

```
double& MaxReward ( ) [inline]
```

Modify the maximum reward value.

Definition at line 124 of file reward_clipping.hpp.

39.441.4.7 MinReward() [1/2]

```
double MinReward ( ) const [inline]
```

Get the minimum reward value.

Definition at line 117 of file reward_clipping.hpp.

39.441.4.8 MinReward() [2/2]

```
double& MinReward ( ) [inline]
```

Modify the minimum reward value.

Definition at line 119 of file reward_clipping.hpp.

39.441.4.9 Sample() [1/2]

```
double Sample (
    const State & state,
    const Action & action,
    State & nextState ) const [inline]
```

Dynamics of Environment.

The rewards returned from the base environment are clipped according the maximum and minimum values specified.

Parameters

<i>state</i>	The current state.
<i>action</i>	The current action.
<i>nextState</i>	The next state.

Returns

clippedReward, Reward clipped between [minReward, maxReward].

Definition at line 87 of file reward_clipping.hpp.

Referenced by RewardClipping< EnvironmentType >::Sample().

39.441.4.10 Sample() [2/2]

```
double Sample (
    const State & state,
    const Action & action ) const [inline]
```

Dynamics of Environment.

The rewards returned from the base environment are clipped according the maximum and minimum values specified.

Parameters

<i>state</i>	The current state.
<i>action</i>	The current action.

Returns

clippedReward, Reward clipped between [minReward, maxReward].

Definition at line 105 of file reward_clipping.hpp.

References RewardClipping< EnvironmentType >::Sample().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/**reward_clipping.hpp** ↩

39.442 TrainingConfig Class Reference

Public Member Functions

- **TrainingConfig** ()
- **TrainingConfig** (size_t numWorkers, size_t updateInterval, size_t targetNetworkSyncInterval, size_t stepLimit, size_t explorationSteps, double stepSize, double discount, double gradientLimit, bool doubleQLearning)
- double **Discount** () const
Get the discount rate for future reward.
- double & **Discount** ()
Modify the discount rate for future reward.
- bool **DoubleQLearning** () const
Get the indicator of double q-learning.
- bool & **DoubleQLearning** ()
Modify the indicator of double q-learning.
- size_t **ExplorationSteps** () const
Get the exploration steps.
- size_t & **ExplorationSteps** ()
Modify the exploration steps.
- double **GradientLimit** () const
Get the limit of update gradient.
- double & **GradientLimit** ()
Modify the limit of update gradient.
- size_t **NumWorkers** () const
Get the amount of workers.
- size_t & **NumWorkers** ()
Modify the amount of workers.
- size_t **StepLimit** () const
Get the maximum steps of each episode.
- size_t & **StepLimit** ()
Modify the maximum steps of each episode.
- double **StepSize** () const
Get the step size of the optimizer.
- double & **StepSize** ()
Modify the step size of the optimizer.
- size_t **TargetNetworkSyncInterval** () const
Get the interval for syncing target network.
- size_t & **TargetNetworkSyncInterval** ()
Modify the interval for syncing target network.
- size_t **UpdateInterval** () const
Get the update interval.
- size_t & **UpdateInterval** ()
Modify the update interval.

39.442.1 Detailed Description

Definition at line 19 of file training_config.hpp.

39.442.2 Constructor & Destructor Documentation

39.442.2.1 TrainingConfig() [1/2]

```
TrainingConfig ( ) [inline]
```

Definition at line 22 of file training_config.hpp.

39.442.2.2 TrainingConfig() [2/2]

```
TrainingConfig (
    size_t numWorkers,
    size_t updateInterval,
    size_t targetNetworkSyncInterval,
    size_t stepLimit,
    size_t explorationSteps,
    double stepSize,
    double discount,
    double gradientLimit,
    bool doubleQLearning ) [inline]
```

Definition at line 31 of file training_config.hpp.

39.442.3 Member Function Documentation

39.442.3.1 Discount() [1/2]

```
double Discount ( ) const [inline]
```

Get the discount rate for future reward.

Definition at line 87 of file training_config.hpp.

Referenced by NStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step(), OneStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step(), and OneStepSarsaWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step().

39.442.3.2 Discount() [2/2]

```
double& Discount ( ) [inline]
```

Modify the discount rate for future reward.

Definition at line 89 of file training_config.hpp.

39.442.3.3 DoubleQLearning() [1/2]

```
bool DoubleQLearning ( ) const [inline]
```

Get the indicator of double q-learning.

Definition at line 97 of file training_config.hpp.

39.442.3.4 DoubleQLearning() [2/2]

```
bool& DoubleQLearning ( ) [inline]
```

Modify the indicator of double q-learning.

Definition at line 99 of file training_config.hpp.

39.442.3.5 ExplorationSteps() [1/2]

```
size_t ExplorationSteps ( ) const [inline]
```

Get the exploration steps.

Definition at line 77 of file training_config.hpp.

39.442.3.6 ExplorationSteps() [2/2]

```
size_t& ExplorationSteps ( ) [inline]
```

Modify the exploration steps.

Definition at line 79 of file training_config.hpp.

39.442.3.7 GradientLimit() [1/2]

```
double GradientLimit ( ) const [inline]
```

Get the limit of update gradient.

Definition at line 92 of file training_config.hpp.

Referenced by `NStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step()`, `OneStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step()`, and `OneStepSarsaWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step()`.

39.442.3.8 GradientLimit() [2/2]

```
double& GradientLimit ( ) [inline]
```

Modify the limit of update gradient.

Definition at line 94 of file training_config.hpp.

39.442.3.9 NumWorkers() [1/2]

```
size_t NumWorkers ( ) const [inline]
```

Get the amount of workers.

Definition at line 53 of file training_config.hpp.

39.442.3.10 NumWorkers() [2/2]

```
size_t& NumWorkers ( ) [inline]
```

Modify the amount of workers.

Definition at line 55 of file training_config.hpp.

39.442.3.11 StepLimit() [1/2]

```
size_t StepLimit ( ) const [inline]
```

Get the maximum steps of each episode.

Definition at line 69 of file training_config.hpp.

Referenced by NStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step(), OneStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step(), and OneStepSarsaWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step().

39.442.3.12 StepLimit() [2/2]

```
size_t& StepLimit ( ) [inline]
```

Modify the maximum steps of each episode.

Setting it to 0 means no limit.

Definition at line 74 of file training_config.hpp.

39.442.3.13 StepSize() [1/2]

```
double StepSize ( ) const [inline]
```

Get the step size of the optimizer.

Definition at line 82 of file training_config.hpp.

Referenced by NStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step(), OneStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step(), and OneStepSarsaWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step().

39.442.3.14 StepSize() [2/2]

```
double& StepSize ( ) [inline]
```

Modify the step size of the optimizer.

Definition at line 84 of file training_config.hpp.

39.442.3.15 TargetNetworkSyncInterval() [1/2]

```
size_t TargetNetworkSyncInterval ( ) const [inline]
```

Get the interval for syncing target network.

Definition at line 63 of file training_config.hpp.

Referenced by NStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step(), OneStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step(), and OneStepSarsaWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step().

39.442.3.16 TargetNetworkSyncInterval() [2/2]

```
size_t& TargetNetworkSyncInterval ( ) [inline]
```

Modify the interval for syncing target network.

Definition at line 66 of file training_config.hpp.

39.442.3.17 UpdateInterval() [1/2]

```
size_t UpdateInterval ( ) const [inline]
```

Get the update interval.

Definition at line 58 of file training_config.hpp.

Referenced by NStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step(), OneStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step(), and OneStepSarsaWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >::Step().

39.442.3.18 UpdateInterval() [2/2]

```
size_t& UpdateInterval ( ) [inline]
```

Modify the update interval.

Definition at line 60 of file training_config.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/ **training_config.hpp**

39.443 MethodFormDetector< Class, MethodForm, AdditionalArgsCount > Struct Template Reference

39.443.1 Detailed Description

```
template<typename Class, template< typename... > class MethodForm, size_t AdditionalArgsCount>
struct mlpack::sfinae::MethodFormDetector< Class, MethodForm, AdditionalArgsCount >
```

Definition at line 45 of file sfinae_utility.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **sfinae_utility.hpp**

39.444 MethodFormDetector< Class, MethodForm, 0 > Struct Template Reference

Public Member Functions

- void **operator()** (MethodForm< Class >)

39.444.1 Detailed Description

```
template<typename Class, template< typename... > class MethodForm>
struct mlpack::sfinae::MethodFormDetector< Class, MethodForm, 0 >
```

Definition at line 48 of file sfinae_utility.hpp.

39.444.2 Member Function Documentation

39.444.2.1 operator>()

```
void operator() (
    MethodForm< Class > )
```

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **sfinae_utility.hpp**

39.445 MethodFormDetector< Class, MethodForm, 1 > Struct Template Reference

Public Member Functions

- template<class T1 >
void **operator()** (MethodForm< Class, T1 >)

39.445.1 Detailed Description

```
template<typename Class, template< typename... > class MethodForm>  
struct mlpack::sfinae::MethodFormDetector< Class, MethodForm, 1 >
```

Definition at line 54 of file sfinae_utility.hpp.

39.445.2 Member Function Documentation

39.445.2.1 operator>()

```
void operator() (  
    MethodForm< Class, T1 > )
```

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **sfinae_utility.hpp**

39.446 MethodFormDetector< Class, MethodForm, 2 > Struct Template Reference

Public Member Functions

- template<class T1 , class T2 >
void **operator()** (MethodForm< Class, T1, T2 >)

39.446.1 Detailed Description

```
template<typename Class, template< typename... > class MethodForm>  
struct mlpack::sfinae::MethodFormDetector< Class, MethodForm, 2 >
```

Definition at line 61 of file sfinae_utility.hpp.

39.446.2 Member Function Documentation

39.446.2.1 operator()()

```
void operator() (
    MethodForm< Class, T1, T2 > )
```

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **sfinae_utility.hpp**

39.447 MethodFormDetector< Class, MethodForm, 3 > Struct Template Reference

Public Member Functions

- template<class T1 , class T2 , class T3 >
void **operator()** (MethodForm< Class, T1, T2, T3 >)

39.447.1 Detailed Description

```
template<typename Class, template< typename... > class MethodForm>
struct mlpack::sfinae::MethodFormDetector< Class, MethodForm, 3 >
```

Definition at line 68 of file sfinae_utility.hpp.

39.447.2 Member Function Documentation

39.447.2.1 operator()()

```
void operator() (
    MethodForm< Class, T1, T2, T3 > )
```

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **sfinae_utility.hpp**

39.448 MethodFormDetector< Class, MethodForm, 4 > Struct Template Reference

Public Member Functions

- `template<class T1 , class T2 , class T3 , class T4 >`
`void operator() (MethodForm< Class, T1, T2, T3, T4 >)`

39.448.1 Detailed Description

```
template<typename Class, template< typename... > class MethodForm>
struct mlpack::sfinae::MethodFormDetector< Class, MethodForm, 4 >
```

Definition at line 75 of file `sfinae_utility.hpp`.

39.448.2 Member Function Documentation

39.448.2.1 `operator()`

```
void operator() (
    MethodForm< Class, T1, T2, T3, T4 > )
```

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ sfinae_utility.hpp`

39.449 MethodFormDetector< Class, MethodForm, 5 > Struct Template Reference

Public Member Functions

- `template<class T1 , class T2 , class T3 , class T4 , class T5 >`
`void operator() (MethodForm< Class, T1, T2, T3, T4, T5 >)`

39.449.1 Detailed Description

```
template<typename Class, template< typename... > class MethodForm>
struct mlpack::sfinae::MethodFormDetector< Class, MethodForm, 5 >
```

Definition at line 82 of file `sfinae_utility.hpp`.

39.449.2 Member Function Documentation

39.449.2.1 operator()()

```
void operator() (
    MethodForm< Class, T1, T2, T3, T4, T5 > )
```

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **sfinae_utility.hpp**

39.450 MethodFormDetector< Class, MethodForm, 6 > Struct Template Reference

Public Member Functions

- template<class T1 , class T2 , class T3 , class T4 , class T5 , class T6 >
void **operator()** (MethodForm< Class, T1, T2, T3, T4, T5, T6 >)

39.450.1 Detailed Description

```
template<typename Class, template< typename... > class MethodForm>
struct mlpack::sfinae::MethodFormDetector< Class, MethodForm, 6 >
```

Definition at line 89 of file sfinae_utility.hpp.

39.450.2 Member Function Documentation

39.450.2.1 operator()()

```
void operator() (
    MethodForm< Class, T1, T2, T3, T4, T5, T6 > )
```

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **sfinae_utility.hpp**

39.451 MethodFormDetector< Class, MethodForm, 7 > Struct Template Reference

Public Member Functions

- `template<class T1 , class T2 , class T3 , class T4 , class T5 , class T6 , class T7 >`
`void operator() (MethodForm< Class, T1, T2, T3, T4, T5, T6, T7 >)`

39.451.1 Detailed Description

```
template<typename Class, template< typename... > class MethodForm>
struct mlpack::sfinae::MethodFormDetector< Class, MethodForm, 7 >
```

Definition at line 96 of file `sfinae_utility.hpp`.

39.451.2 Member Function Documentation

39.451.2.1 operator>()

```
void operator() (
    MethodForm< Class, T1, T2, T3, T4, T5, T6, T7 > )
```

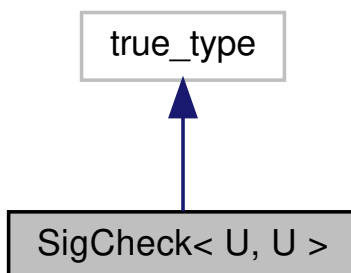
The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ sfinae_utility.hpp`

39.452 SigCheck< U, U > Struct Template Reference

Utility struct for checking signatures.

Inheritance diagram for `SigCheck< U, U >`:



39.452.1 Detailed Description

```
template<typename U, U>
struct mlpack::sfinae::SigCheck< U, U >
```

Utility struct for checking signatures.

Definition at line 103 of file sfinae_utility.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **sfinae_utility.hpp**

39.453 DataDependentRandomInitializer Class Reference

A data-dependent random dictionary initializer for **SparseCoding** (p. 1916).

Static Public Member Functions

- static void **Initialize** (const arma::mat &data, const size_t atoms, arma::mat &dictionary)
Initialize the dictionary by adding together three random observations from the data, and then normalizing the atom.

39.453.1 Detailed Description

A data-dependent random dictionary initializer for **SparseCoding** (p. 1916).

This creates random dictionary atoms by adding three random observations from the data together, and then normalizing the atom.

Definition at line 26 of file data_dependent_random_initializer.hpp.

39.453.2 Member Function Documentation

39.453.2.1 Initialize()

```
static void Initialize (
    const arma::mat & data,
    const size_t atoms,
    arma::mat & dictionary ) [inline], [static]
```

Initialize the dictionary by adding together three random observations from the data, and then normalizing the atom.

This implementation is simple enough to be included with the definition.


Parameters

<i>data</i>	Dataset to initialize the dictionary with.
<i>atoms</i>	Number of atoms in dictionary.
<i>dictionary</i>	Dictionary to initialize.

Definition at line 38 of file `data_dependent_random_initializer.hpp`.

References `mlpack::math::RandInt()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_coding/data_dependent_random_initializer.hpp` 

39.454 NothingInitializer Class Reference

A DictionaryInitializer for **SparseCoding** (p. 1916) which does not initialize anything; it is useful for when the dictionary is already known and will be set with **SparseCoding::Dictionary()** (p. 1920).

Static Public Member Functions

- static void **Initialize** (const arma::mat &, const size_t, arma::mat &)
This function does not initialize the dictionary.

39.454.1 Detailed Description

A DictionaryInitializer for **SparseCoding** (p. 1916) which does not initialize anything; it is useful for when the dictionary is already known and will be set with **SparseCoding::Dictionary()** (p. 1920).

Definition at line 27 of file `nothing_initializer.hpp`.

39.454.2 Member Function Documentation

39.454.2.1 Initialize()

```
static void Initialize (
    const arma::mat & ,
    const size_t ,
    arma::mat & ) [inline], [static]
```

This function does not initialize the dictionary.

This will cause problems for **SparseCoding** (p. 1916) if the dictionary is not set manually before running the method.

Definition at line 35 of file `nothing_initializer.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_coding/ nothing_initializer.hpp`

39.455 RandomInitializer Class Reference

A DictionaryInitializer for use with the **SparseCoding** (p. 1916) class.

Static Public Member Functions

- static void **Initialize** (const arma::mat &data, const size_t atoms, arma::mat &dictionary)
Initialize the dictionary randomly from a normal distribution, such that each atom has a norm of 1.

39.455.1 Detailed Description

A DictionaryInitializer for use with the **SparseCoding** (p. 1916) class.

This provides a random, normally distributed dictionary, such that each atom has a norm of 1.

Definition at line 25 of file `random_initializer.hpp`.

39.455.2 Member Function Documentation

39.455.2.1 Initialize()

```
static void Initialize (
    const arma::mat & data,
    const size_t atoms,
    arma::mat & dictionary ) [inline], [static]
```

Initialize the dictionary randomly from a normal distribution, such that each atom has a norm of 1.

This is simple enough to be included with the definition.

Parameters

<i>data</i>	Dataset to use for initialization.
<i>atoms</i>	Number of atoms (columns) in the dictionary.
<i>dictionary</i>	Dictionary to initialize.

Definition at line 37 of file random_initializer.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_coding/ **random_initializer.hpp**

39.456 SparseCoding Class Reference

An implementation of Sparse Coding with Dictionary Learning that achieves sparsity via an l_1 -norm regularizer on the codes (LASSO) or an (l_1+l_2) -norm regularizer on the codes (the Elastic Net).

Public Member Functions

- `template<typename DictionaryInitializer = DataDependentRandomInitializer>`
SparseCoding (const arma::mat &data, const size_t atoms, const double lambda1, const double lambda2=0, const size_t maxIterations=0, const double objTolerance=0.01, const double newtonTolerance=1e-6, const DictionaryInitializer &initializer=DictionaryInitializer())
*Set the parameters to **SparseCoding** (p. 1916).*
- **SparseCoding** (const size_t atoms=0, const double lambda1=0, const double lambda2=0, const size_t maxIterations=0, const double objTolerance=0.01, const double newtonTolerance=1e-6)
*Set the parameters to **SparseCoding** (p. 1916).*
- size_t **Atoms** () const
Access the number of atoms.
- size_t & **Atoms** ()
Modify the number of atoms.
- const arma::mat & **Dictionary** () const
Access the dictionary.
- arma::mat & **Dictionary** ()
Modify the dictionary.
- void **Encode** (const arma::mat &data, arma::mat &codes)
Sparse code each point in the given dataset via LARS, using the current dictionary and store the encoded data in the codes matrix.
- double **Lambda1** () const
Access the L_1 regularization term.
- double & **Lambda1** ()
Modify the L_1 regularization term.
- double **Lambda2** () const
Access the L_2 regularization term.
- double & **Lambda2** ()

- Modify the L2 regularization term.
 - `size_t MaxIterations () const`
 Get the maximum number of iterations.
 - `size_t & MaxIterations ()`
 Modify the maximum number of iterations.
 - `double NewtonTolerance () const`
 Get the tolerance for Newton's method (dictionary optimization step).
 - `double & NewtonTolerance ()`
 Modify the tolerance for Newton's method (dictionary optimization step).
 - `double Objective (const arma::mat &data, const arma::mat &codes) const`
 Compute the objective function.
 - `double ObjTolerance () const`
 Get the objective tolerance.
 - `double & ObjTolerance ()`
 Modify the objective tolerance.
 - `double OptimizeDictionary (const arma::mat &data, const arma::mat &codes, const arma::uvec &adjacencies)`
 Learn dictionary via Newton method based on Lagrange dual.
 - `void ProjectDictionary ()`
 Project each atom of the dictionary back onto the unit ball, if necessary.
 - `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`
 Serialize the sparse coding model.
 - `template<typename DictionaryInitializer = DataDependentRandomInitializer>`
`double Train (const arma::mat &data, const DictionaryInitializer &initializer=DictionaryInitializer())`
 Train the sparse coding model on the given dataset.

39.456.1 Detailed Description

An implementation of Sparse Coding with Dictionary Learning that achieves sparsity via an l1-norm regularizer on the codes (LASSO) or an (l1+l2)-norm regularizer on the codes (the Elastic Net).

Let d be the number of dimensions in the original space, m the number of training points, and k the number of atoms in the dictionary (the dimension of the learned feature space). The training data X is a d -by- m matrix where each column is a point and each row is a dimension. The dictionary D is a d -by- k matrix, and the sparse codes matrix Z is a k -by- m matrix. This program seeks to minimize the objective:

$$\min_{D,Z} 0.5 \|X - DZ\|_F^2 + \lambda_1 \sum_{i=1}^m \|Z_i\|_1 + 0.5 \lambda_2 \sum_{i=1}^m \|Z_i\|_2^2$$

subject to $\|D_j\|_2 \leq 1$ for $1 \leq j \leq k$ where typically $\lambda_1 > 0$ and $\lambda_2 = 0$.

This problem is solved by an algorithm that alternates between a dictionary learning step and a sparse coding step. The dictionary learning step updates the dictionary D using a Newton method based on the Lagrange dual (see the paper below for details). The sparse coding step involves solving a large number of sparse linear regression problems; this can be done efficiently using LARS, an algorithm that can solve the LASSO or the Elastic Net (papers below).

Here are those papers:

```

@incollection{lee2007efficient,
  title = {Efficient sparse coding algorithms},
  author = {Honglak Lee and Alexis Battle and Rajat Raina and Andrew Y. Ng},
  booktitle = {Advances in Neural Information Processing Systems 19},
  editor = {B. Schölkopf and J. Platt and T. Hoffman},
  publisher = {MIT Press},
  address = {Cambridge, MA},
  pages = {801--808},
  year = {2007}
}

@article{efron2004least,
  title={Least angle regression},
  author={Efron, B. and Hastie, T. and Johnstone, I. and Tibshirani, R.},
  journal={The Annals of Statistics},
  volume={32},
  number={2},
  pages={407--499},
  year={2004},
  publisher={Institute of Mathematical Statistics}
}

@article{zou2005regularization,
  title={Regularization and variable selection via the elastic net},
  author={Zou, H. and Hastie, T.},
  journal={Journal of the Royal Statistical Society Series B},
  volume={67},
  number={2},
  pages={301--320},
  year={2005},
  publisher={Royal Statistical Society}
}

```

Note that the implementation here does not use the feature-sign search algorithm from Honglak Lee's paper, but instead the LARS algorithm suggested in that paper.

When **Train()** (p. 1924) is called, the dictionary is initialized using the `DictionaryInitializationPolicy` class. Possible choices include the **RandomInitializer** (p. 1915), which provides an entirely random dictionary, the **DataDependentRandomInitializer** (p. 1913), which provides a random dictionary based loosely on characteristics of the dataset, and the **NothingInitializer** (p. 1914), which does not initialize the dictionary – instead, the user should set the dictionary using the **Dictionary()** (p. 1920) mutator method.

Once a dictionary is trained with **Train()** (p. 1924), another matrix may be encoded with the **Encode()** (p. 1921) function.

Template Parameters

<i>DictionaryInitializationPolicy</i>	The class to use to initialize the dictionary; must have 'void Initialize(const arma::mat& data, arma::mat& dictionary)' function.
---------------------------------------	--

Definition at line 115 of file `sparse_coding.hpp`.

39.456.2 Constructor & Destructor Documentation

39.456.2.1 SparseCoding() [1/2]

```

SparseCoding (
    const arma::mat & data,
    const size_t atoms,
    const double lambda1,
    const double lambda2 = 0,
    const size_t maxIterations = 0,
    const double objTolerance = 0.01,
    const double newtonTolerance = 1e-6,
    const DictionaryInitializer & initializer = DictionaryInitializer() )

```

Set the parameters to **SparseCoding** (p. 1916).

lambda2 defaults to 0. This constructor will train the model. If that is not desired, call the other constructor that does not take a data matrix. This constructor will also initialize the dictionary using the given DictionaryInitializer before training.

If you want to initialize the dictionary to a custom matrix, consider either writing your own DictionaryInitializer class (with void Initialize(const arma::mat& data, arma::mat& dictionary) function), or call the constructor that does not take a data matrix, then call **Dictionary()** (p. 1920) to set the dictionary matrix to a matrix of your choosing, and then call **Train()** (p. 1924) with **NothingInitializer** (p. 1914) (i.e. Train<NothingInitializer>(data)).

Parameters

<i>data</i>	Data matrix.
<i>atoms</i>	Number of atoms in dictionary.
<i>lambda1</i>	Regularization parameter for l1-norm penalty.
<i>lambda2</i>	Regularization parameter for l2-norm penalty.
<i>maxIterations</i>	Maximum number of iterations to run algorithm. If 0, the algorithm will run until convergence (or forever).
<i>objTolerance</i>	Tolerance for objective function. When an iteration of the algorithm produces an improvement smaller than this, the algorithm will terminate.
<i>newtonTolerance</i>	Tolerance for the Newton's method dictionary optimization step.

39.456.2.2 SparseCoding() [2/2]

```

SparseCoding (
    const size_t atoms = 0,
    const double lambda1 = 0,
    const double lambda2 = 0,
    const size_t maxIterations = 0,
    const double objTolerance = 0.01,
    const double newtonTolerance = 1e-6 )

```

Set the parameters to **SparseCoding** (p. 1916).

lambda2 defaults to 0. This constructor will not train the model, and a subsequent call to **Train()** (p. 1924) will be required before the model can encode points with **Encode()** (p. 1921).

Parameters

<i>atoms</i>	Number of atoms in dictionary.
<i>lambda1</i>	Regularization parameter for l1-norm penalty.
<i>lambda2</i>	Regularization parameter for l2-norm penalty.
<i>maxIterations</i>	Maximum number of iterations to run algorithm. If 0, the algorithm will run until convergence (or forever).
<i>objTolerance</i>	Tolerance for objective function. When an iteration of the algorithm produces an improvement smaller than this, the algorithm will terminate.
<i>newtonTolerance</i>	Tolerance for the Newton's method dictionary optimization step.

39.456.3 Member Function Documentation

39.456.3.1 Atoms() [1/2]

```
size_t Atoms ( ) const [inline]
```

Access the number of atoms.

Definition at line 227 of file sparse_coding.hpp.

39.456.3.2 Atoms() [2/2]

```
size_t& Atoms ( ) [inline]
```

Modify the number of atoms.

Definition at line 229 of file sparse_coding.hpp.

39.456.3.3 Dictionary() [1/2]

```
const arma::mat& Dictionary ( ) const [inline]
```

Access the dictionary.

Definition at line 222 of file sparse_coding.hpp.

39.456.3.4 Dictionary() [2/2]

```
arma::mat& Dictionary ( ) [inline]
```

Modify the dictionary.

Definition at line 224 of file `sparse_coding.hpp`.

39.456.3.5 Encode()

```
void Encode (
    const arma::mat & data,
    arma::mat & codes )
```

Sparse code each point in the given dataset via LARS, using the current dictionary and store the encoded data in the codes matrix.

Parameters

<i>data</i>	Input data matrix to be encoded.
<i>codes</i>	Output codes matrix.

39.456.3.6 Lambda1() [1/2]

```
double Lambda1 ( ) const [inline]
```

Access the L1 regularization term.

Definition at line 232 of file `sparse_coding.hpp`.

39.456.3.7 Lambda1() [2/2]

```
double& Lambda1 ( ) [inline]
```

Modify the L1 regularization term.

Definition at line 234 of file `sparse_coding.hpp`.

39.456.3.8 Lambda2() [1/2]

```
double Lambda2 ( ) const [inline]
```

Access the L2 regularization term.

Definition at line 237 of file sparse_coding.hpp.

39.456.3.9 Lambda2() [2/2]

```
double& Lambda2 ( ) [inline]
```

Modify the L2 regularization term.

Definition at line 239 of file sparse_coding.hpp.

39.456.3.10 MaxIterations() [1/2]

```
size_t MaxIterations ( ) const [inline]
```

Get the maximum number of iterations.

Definition at line 242 of file sparse_coding.hpp.

39.456.3.11 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Modify the maximum number of iterations.

Definition at line 244 of file sparse_coding.hpp.

39.456.3.12 NewtonTolerance() [1/2]

```
double NewtonTolerance ( ) const [inline]
```

Get the tolerance for Newton's method (dictionary optimization step).

Definition at line 252 of file sparse_coding.hpp.

39.456.3.13 NewtonTolerance() [2/2]

```
double& NewtonTolerance ( ) [inline]
```

Modify the tolerance for Newton's method (dictionary optimization step).

Definition at line 254 of file `sparse_coding.hpp`.

References `SparseCoding::serialize()`.

39.456.3.14 Objective()

```
double Objective (
    const arma::mat & data,
    const arma::mat & codes ) const
```

Compute the objective function.

39.456.3.15 ObjTolerance() [1/2]

```
double ObjTolerance ( ) const [inline]
```

Get the objective tolerance.

Definition at line 247 of file `sparse_coding.hpp`.

39.456.3.16 ObjTolerance() [2/2]

```
double& ObjTolerance ( ) [inline]
```

Modify the objective tolerance.

Definition at line 249 of file `sparse_coding.hpp`.

39.456.3.17 OptimizeDictionary()

```
double OptimizeDictionary (
    const arma::mat & data,
    const arma::mat & codes,
    const arma::uvec & adjacencies )
```

Learn dictionary via Newton method based on Lagrange dual.

Parameters

<i>data</i>	Data matrix.
<i>codes</i>	Matrix of codes.
<i>adjacencies</i>	Indices of entries (unrolled column by column) of the coding matrix Z that are non-zero (the adjacency matrix for the bipartite graph of points and atoms).

Returns

the norm of the gradient of the Lagrange dual with respect to the dual variables

39.456.3.18 ProjectDictionary()

```
void ProjectDictionary ( )
```

Project each atom of the dictionary back onto the unit ball, if necessary.

39.456.3.19 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the sparse coding model.

Referenced by SparseCoding::NewtonTolerance().

39.456.3.20 Train()

```
double Train (
    const arma::mat & data,
    const DictionaryInitializer & initializer = DictionaryInitializer() )
```

Train the sparse coding model on the given dataset.

Returns

The final objective value.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_coding/ **sparse_coding.hpp**

39.457 BiasSVD< OptimizerType > Class Template Reference

Bias SVD is an improvement on Regularized SVD which is a matrix factorization techniques.

Public Member Functions

- **BiasSVD** (const size_t iterations=10, const double alpha=0.02, const double lambda=0.05)
Constructor of Bias SVD.
- void **Apply** (const arma::mat &data, const size_t rank, arma::mat &u, arma::mat &v, arma::vec &p, arma::vec &q)
Trains the model and obtains user/item matrices and user/item bias.

39.457.1 Detailed Description

```
template<typename OptimizerType = ens::StandardSGD>
class mlpack::svd::BiasSVD< OptimizerType >
```

Bias SVD is an improvement on Regularized SVD which is a matrix factorization techniques.

Bias SVD outputs user/item latent vectors and user/item bias, so that $r_{iu} = b_i + b_u + p_i * q_u$, where b, p, q are bias, item latent, user latent respectively. Parameters are optimized by Stochastic Gradient Descent (SGD). The updates also penalize the learning of large feature values by means of regularization.

An example of how to use the interface is shown below:

```
arma::mat data; // Rating data in the form of coordinate list.

const size_t rank = 10; // Rank used for the decomposition.
const size_t iterations = 10; // Number of iterations used for optimization.

const double alpha = 0.005 // Learning rate for the SGD optimizer.
const double lambda = 0.02 // Regularization parameter for the optimization.

// Make a BiasSVD object.
BiasSVD<> biasSVD(iterations, alpha, lambda);

arma::mat u, v; // Item and User matrices.
arma::vec p, q; // Item and User bias.

// Use the Apply() method to get a factorization.
rSVD.Apply(data, rank, u, v, p, q);
```

Definition at line 57 of file bias_svd.hpp.

39.457.2 Constructor & Destructor Documentation

39.457.2.1 BiasSVD()

```
BiasSVD (
    const size_t iterations = 10,
    const double alpha = 0.02,
    const double lambda = 0.05 )
```

Constructor of Bias SVD.

By default SGD optimizer is used in **BiasSVD** (p. 1925). The optimizer uses a template specialization of Optimize().

Parameters

<i>iterations</i>	Number of optimization iterations.
<i>alpha</i>	Learning rate for the SGD optimizer.
<i>lambda</i>	Regularization parameter for the optimization.

39.457.3 Member Function Documentation

39.457.3.1 Apply()

```
void Apply (
    const arma::mat & data,
    const size_t rank,
    arma::mat & u,
    arma::mat & v,
    arma::vec & p,
    arma::vec & q )
```

Trains the model and obtains user/item matrices and user/item bias.

Parameters

<i>data</i>	Rating data matrix.
<i>rank</i>	Rank parameter to be used for optimization.
<i>u</i>	Item matrix obtained on decomposition.
<i>v</i>	User matrix obtained on decomposition.
<i>p</i>	Item bias.
<i>q</i>	User bias.

Referenced by BiasSVDPolicy::Apply().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/bias_svd/ **bias_svd.hpp**

39.458 BiasSVDFunction< MatType > Class Template Reference

This class contains methods which are used to calculate the cost of **BiasSVD** (p. 1925)'s objective function, to calculate gradient of parameters with respect to the objective function, etc.

Public Member Functions

- **BiasSVDFunction** (const MatType &data, const size_t rank, const double lambda)
*Constructor for **BiasSVDFunction** (p. 1926) class.*
- const arma::mat & **Dataset** () const
Return the dataset passed into the constructor.
- double **Evaluate** (const arma::mat ¶meters) const
Evaluates the cost function over all examples in the data.
- double **Evaluate** (const arma::mat ¶meters, const size_t start, const size_t batchSize=1) const
Evaluates the cost function for one training example.
- const arma::mat & **GetInitialPoint** () const
Return the initial point for the optimization.
- void **Gradient** (const arma::mat ¶meters, arma::mat &gradient) const
Evaluates the full gradient of the cost function over all the training examples.
- template<typename GradType >
void **Gradient** (const arma::mat ¶meters, const size_t start, GradType &gradient, const size_t batchSize=1) const
Evaluates the gradient of the cost function over one training example.
- double **Lambda** () const
Return the regularization parameters.
- size_t **NumFunctions** () const
Return the number of training examples. Useful for SGD optimizer.
- size_t **NumItems** () const
Return the number of items in the data.
- size_t **NumUsers** () const
Return the number of users in the data.
- size_t **Rank** () const
Return the rank used for the factorization.
- void **Shuffle** ()
Shuffle the points in the dataset.

39.458.1 Detailed Description

```
template<typename MatType = arma::mat>
class mlpack::svd::BiasSVDFunction< MatType >
```

This class contains methods which are used to calculate the cost of **BiasSVD** (p. 1925)'s objective function, to calculate gradient of parameters with respect to the objective function, etc.

Template Parameters

<i>MatType</i>	The matrix type of the dataset.
----------------	---------------------------------

Definition at line 31 of file bias_svd_function.hpp.

39.458.2 Constructor & Destructor Documentation

39.458.2.1 BiasSVDFunction()

```
BiasSVDFunction (
    const MatType & data,
    const size_t rank,
    const double lambda )
```

Constructor for **BiasSVDFunction** (p. 1926) class.

The constructor calculates the number of users and items in the passed data. It also randomly initializes the parameter values.

Parameters

<i>data</i>	Dataset for which SVD is calculated.
<i>rank</i>	Rank used for matrix factorization.
<i>lambda</i>	Regularization parameter used for optimization.

39.458.3 Member Function Documentation

39.458.3.1 Dataset()

```
const arma::mat& Dataset ( ) const [inline]
```

Return the dataset passed into the constructor.

Definition at line 107 of file `bias_svd_function.hpp`.

39.458.3.2 Evaluate() [1/2]

```
double Evaluate (
    const arma::mat & parameters ) const
```

Evaluates the cost function over all examples in the data.

Parameters

<i>parameters</i>	Parameters(user/item matrices/bias) of the decomposition.
-------------------	---

39.458.3.3 Evaluate() [2/2]

```
double Evaluate (
    const arma::mat & parameters,
    const size_t start,
    const size_t batchSize = 1 ) const
```

Evaluates the cost function for one training example.

Useful for the SGD optimizer abstraction which uses one training example at a time.

Parameters

<i>parameters</i>	Parameters(user/item matrices/bias) of the decomposition.
<i>start</i>	First index of the training examples to be used.
<i>batchSize</i>	Size of batch to evaluate.

39.458.3.4 GetInitialPoint()

```
const arma::mat& GetInitialPoint ( ) const [inline]
```

Return the initial point for the optimization.

Definition at line 104 of file bias_svd_function.hpp.

39.458.3.5 Gradient() [1/2]

```
void Gradient (
    const arma::mat & parameters,
    arma::mat & gradient ) const
```

Evaluates the full gradient of the cost function over all the training examples.

Parameters

<i>parameters</i>	Parameters(user/item matrices/bias) of the decomposition.
<i>gradient</i>	Calculated gradient for the parameters.

39.458.3.6 Gradient() [2/2]

```
void Gradient (
    const arma::mat & parameters,
    const size_t start,
    GradType & gradient,
    const size_t batchSize = 1 ) const
```

Evaluates the gradient of the cost function over one training example.

This function is useful for optimizers like SGD. The type of the gradient parameter is a template argument to allow the computation of a sparse gradient.

Template Parameters

<i>GradType</i>	The type of the gradient out-param.
-----------------	-------------------------------------

Parameters

<i>parameters</i>	Parameters(user/item matrices/bias) of the decomposition.
<i>start</i>	The first index of the training examples to use.
<i>gradient</i>	Calculated gradient for the parameters.
<i>batchSize</i>	Size of batch to calculate gradient for.

39.458.3.7 Lambda()

```
double Lambda ( ) const [inline]
```

Return the regularization parameters.

Definition at line 119 of file bias_svd_function.hpp.

39.458.3.8 NumFunctions()

```
size_t NumFunctions ( ) const [inline]
```

Return the number of training examples. Useful for SGD optimizer.

Definition at line 110 of file `bias_svd_function.hpp`.

39.458.3.9 NumItems()

```
size_t NumItems ( ) const [inline]
```

Return the number of items in the data.

Definition at line 116 of file `bias_svd_function.hpp`.

39.458.3.10 NumUsers()

```
size_t NumUsers ( ) const [inline]
```

Return the number of users in the data.

Definition at line 113 of file `bias_svd_function.hpp`.

39.458.3.11 Rank()

```
size_t Rank ( ) const [inline]
```

Return the rank used for the factorization.

Definition at line 122 of file `bias_svd_function.hpp`.

39.458.3.12 Shuffle()

```
void Shuffle ( )
```

Shuffle the points in the dataset.

This may be used by optimizers.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/bias_svd/ bias_svd_function.hpp`

39.459 QUIC_SVD Class Reference

QUIC-SVD is a matrix factorization technique, which operates in a subspace such that A's approximation in that subspace has minimum error(A being the data matrix).

Public Member Functions

- **QUIC_SVD** (const arma::mat &dataset, arma::mat &u, arma::mat &v, arma::mat &sigma, const double epsilon=0.03, const double delta=0.1)

Constructor which implements the QUIC-SVD algorithm.

- void **ExtractSVD** (arma::mat &u, arma::mat &v, arma::mat &sigma)

This function uses the vector subspace created using a cosine tree to calculate an approximate SVD of the original matrix.

39.459.1 Detailed Description

QUIC-SVD is a matrix factorization technique, which operates in a subspace such that A's approximation in that subspace has minimum error(A being the data matrix).

The subspace is constructed using a cosine tree, which ensures minimum representative rank(and thus a fast running time). It follows a splitting policy based on Length-squared(LS) sampling and constructs the child nodes based on the absolute cosines of the remaining points relative to the pivot. The centroids of the points in the child nodes are added to the subspace span in each step. Each node is then placed into a queue prioritized by its residual error. The subspace approximation error of A after each step is calculated using a Monte Carlo estimate. If the error is below a certain threshold, the method proceeds to calculate the Singular Value Decomposition in the obtained subspace. Otherwise, the same procedure is repeated until we obtain a subspace of sufficiently low error. Technical details can be found in the following paper:

<http://www.cc.gatech.edu/~isbell/papers/isbell-quicsvd-nips-2008.pdf>

An example of how to use the interface is shown below:

```
arma::mat data; // Data matrix.

const double epsilon = 0.01; // Relative error limit of data in subspace.
const double delta = 0.1 // Lower error bound for Monte Carlo estimate.

arma::mat u, v, sigma; // Matrices for the factors. data = u * sigma * v.t()

// Get the factorization in the constructor.
QUIC_SVD(data, u, v, sigma, epsilon, delta);
```

Definition at line 53 of file quic_svd.hpp.

39.459.2 Constructor & Destructor Documentation

39.459.2.1 QUIC_SVD()

```

QUIC_SVD (
    const arma::mat & dataset,
    arma::mat & u,
    arma::mat & v,
    arma::mat & sigma,
    const double epsilon = 0.03,
    const double delta = 0.1 )

```

Constructor which implements the QUIC-SVD algorithm.

The function calls the CosineTree constructor to create a subspace basis, where the original matrix's projection has minimum reconstruction error. The constructor then uses the **ExtractSVD()** (p. 1933) function to calculate the SVD of the original dataset in that subspace.

Parameters

<i>dataset</i>	Matrix for which SVD is calculated.
<i>u</i>	First unitary matrix.
<i>v</i>	Second unitary matrix.
<i>sigma</i>	Diagonal matrix of singular values.
<i>epsilon</i>	Error tolerance fraction for calculated subspace.
<i>delta</i>	Cumulative probability for Monte Carlo error lower bound.

39.459.3 Member Function Documentation

39.459.3.1 ExtractSVD()

```

void ExtractSVD (
    arma::mat & u,
    arma::mat & v,
    arma::mat & sigma )

```

This function uses the vector subspace created using a cosine tree to calculate an approximate SVD of the original matrix.

Parameters

<i>u</i>	First unitary matrix.
<i>v</i>	Second unitary matrix.
<i>sigma</i>	Diagonal matrix of singular values.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/quic_svd/ quic_svd.hpp`

39.460 RandomizedBlockKrylovSVD Class Reference

Randomized block krylov SVD is a matrix factorization that is based on randomized matrix approximation techniques, developed in in "Randomized Block Krylov Methods for Stronger and Faster Approximate Singular Value Decomposition".

Public Member Functions

- **RandomizedBlockKrylovSVD** (const arma::mat &data, arma::mat &u, arma::vec &s, arma::mat &v, const size_t maxIterations=2, const size_t rank=0, const size_t blockSize=0)
Create object for the randomized block krylov SVD method.
- **RandomizedBlockKrylovSVD** (const size_t maxIterations=2, const size_t blockSize=0)
Create object for the randomized block krylov SVD method.
- void **Apply** (const arma::mat &data, arma::mat &u, arma::vec &s, arma::mat &v, const size_t rank)
Apply Principal Component Analysis to the provided data set using the randomized block krylov SVD.
- size_t **BlockSize** () const
Get the block size.
- size_t & **BlockSize** ()
Modify the block size.
- size_t **MaxIterations** () const
Get the number of iterations for the power method.
- size_t & **MaxIterations** ()
Modify the number of iterations for the power method.

39.460.1 Detailed Description

Randomized block krylov SVD is a matrix factorization that is based on randomized matrix approximation techniques, developed in in "Randomized Block Krylov Methods for Stronger and Faster Approximate Singular Value Decomposition".

For more information, see the following.

```
@inproceedings{Musco2015,
  author    = {Cameron Musco and Christopher Musco},
  title     = {Randomized Block Krylov Methods for Stronger and Faster
    Approximate Singular Value Decomposition},
  booktitle = {Advances in Neural Information Processing Systems 28: Annual
    Conference on Neural Information Processing Systems 2015,
    December 7-12, 2015, Montreal, Quebec, Canada},
  pages     = {1396--1404},
  year      = {2015},
}
```

An example of how to use the interface is shown below:

```
arma::mat data; // Rating data in the form of coordinate list.

const size_t rank = 20; // Rank used for the decomposition.

// Make a RandomizedBlockKrylovSVD object.
RandomizedBlockKrylovSVD bSVD();

arma::mat u, s, v;

// Use the Apply() method to get a factorization.
bSVD.Apply(data, u, s, v, rank);
```

Definition at line 58 of file randomized_block_krylov_svd.hpp.

39.460.2 Constructor & Destructor Documentation

39.460.2.1 RandomizedBlockKrylovSVD() [1/2]

```
RandomizedBlockKrylovSVD (
    const arma::mat & data,
    arma::mat & u,
    arma::vec & s,
    arma::mat & v,
    const size_t maxIterations = 2,
    const size_t rank = 0,
    const size_t blockSize = 0 )
```

Create object for the randomized block krylov SVD method.

Parameters

<i>data</i>	Data matrix.
<i>u</i>	First unitary matrix.
<i>v</i>	Second unitary matrix.
<i>s</i>	Diagonal matrix of singular values.
<i>maxIterations</i>	Number of iterations for the power method (Default: 2).
<i>rank</i>	Rank of the approximation (Default: number of rows.)
<i>blockSize</i>	The block size, must be \geq rank (Default: rank + 10).

39.460.2.2 RandomizedBlockKrylovSVD() [2/2]

```
RandomizedBlockKrylovSVD (
    const size_t maxIterations = 2,
    const size_t blockSize = 0 )
```

Create object for the randomized block krylov SVD method.

Parameters

<i>maxIterations</i>	Number of iterations for the power method (Default: 2).
<i>blockSize</i>	The block size, must be \geq rank (Default: rank + 10).

39.460.3 Member Function Documentation

39.460.3.1 Apply()

```
void Apply (
    const arma::mat & data,
    arma::mat & u,
    arma::vec & s,
    arma::mat & v,
    const size_t rank )
```

Apply Principal Component Analysis to the provided data set using the randomized block krylov SVD.

Parameters

<i>data</i>	Data matrix.
<i>u</i>	First unitary matrix.
<i>v</i>	Second unitary matrix.
<i>s</i>	Diagonal matrix of singular values.
<i>rank</i>	Rank of the approximation.

Referenced by RandomizedBlockKrylovSVDPolicy::Apply().

39.460.3.2 BlockSize() [1/2]

```
size_t BlockSize ( ) const [inline]
```

Get the block size.

Definition at line 113 of file randomized_block_krylov_svd.hpp.

39.460.3.3 BlockSize() [2/2]

```
size_t& BlockSize ( ) [inline]
```

Modify the block size.

Definition at line 115 of file randomized_block_krylov_svd.hpp.

39.460.3.4 MaxIterations() [1/2]

```
size_t MaxIterations ( ) const [inline]
```

Get the number of iterations for the power method.

Definition at line 108 of file randomized_block_krylov_svd.hpp.

39.460.3.5 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Modify the number of iterations for the power method.

Definition at line 110 of file randomized_block_krylov_svd.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/block_krylov_svd/ **randomized_block_krylov_svd.hpp** ↩

39.461 RandomizedSVD Class Reference

Randomized SVD is a matrix factorization that is based on randomized matrix approximation techniques, developed in "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions".

Public Member Functions

- **RandomizedSVD** (const arma::mat &data, arma::mat &u, arma::vec &s, arma::mat &v, const size_t iteratedPower=0, const size_t maxIterations=2, const size_t rank=0, const double eps=1e-7)
Create object for the randomized SVD method.
- **RandomizedSVD** (const size_t iteratedPower=0, const size_t maxIterations=2, const double eps=1e-7)
Create object for the randomized SVD method.
- void **Apply** (const arma::sp_mat &data, arma::mat &u, arma::vec &s, arma::mat &v, const size_t rank)
Center the data to apply Principal Component Analysis on given sparse matrix dataset using randomized SVD.
- void **Apply** (const arma::mat &data, arma::mat &u, arma::vec &s, arma::mat &v, const size_t rank)
Center the data to apply Principal Component Analysis on given matrix dataset using randomized SVD.
- template<typename MatType >
void **Apply** (const MatType &data, arma::mat &u, arma::vec &s, arma::mat &v, const size_t rank, MatType rowMean)
Apply Principal Component Analysis to the provided matrix data set using the randomized SVD.
- double **Epsilon** () const
Get the value used for decomposition stability.
- double & **Epsilon** ()
Modify the value used for decomposition stability.
- size_t **IteratedPower** () const
Get the size of the normalized power iterations.
- size_t & **IteratedPower** ()
Modify the size of the normalized power iterations.
- size_t **MaxIterations** () const
Get the number of iterations for the power method.
- size_t & **MaxIterations** ()
Modify the number of iterations for the power method.

39.461.1 Detailed Description

Randomized SVD is a matrix factorization that is based on randomized matrix approximation techniques, developed in "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions".

For more information, see the following.

```
@article{Halko2011,
  author = {Halko, N. and Martinsson, P. G. and Tropp, J. A.},
  title = {Finding Structure with Randomness: Probabilistic Algorithms for
    Constructing Approximate Matrix Decompositions},
  journal = {SIAM Rev.},
  volume = {53},
  year = {2011},
}
```

```
@article{Szlam2014,
  author = {Arthur Szlam Yuval Kluger and Mark Tygert},
  title = {An implementation of a randomized algorithm for principal
    component analysis},
  journal = {CoRR},
  volume = {abs/1412.3510},
  year = {2014},
}
```

An example of how to use the interface is shown below:

```
arma::mat data; // Rating data in the form of coordinate list.

const size_t rank = 20; // Rank used for the decomposition.

// Make a RandomizedSVD object.
RandomizedSVD rSVD();

arma::mat u, s, v;

// Use the Apply() method to get a factorization.
rSVD.Apply(data, u, s, v, rank);
```

Definition at line 66 of file randomized_svd.hpp.

39.461.2 Constructor & Destructor Documentation

39.461.2.1 RandomizedSVD() [1/2]

```
RandomizedSVD (
    const arma::mat & data,
    arma::mat & u,
    arma::vec & s,
    arma::mat & v,
    const size_t iteratedPower = 0,
    const size_t maxIterations = 2,
    const size_t rank = 0,
    const double eps = 1e-7 )
```

Create object for the randomized SVD method.

Parameters

<i>data</i>	Data matrix.
<i>u</i>	First unitary matrix.
<i>v</i>	Second unitary matrix.
<i>sigma</i>	Diagonal matrix of singular values.
<i>iteratedPower</i>	Size of the normalized power iterations (Default: rank + 2).
<i>maxIterations</i>	Number of iterations for the power method (Default: 2).
<i>rank</i>	Rank of the approximation (Default: number of rows.)
<i>eps</i>	The eps coefficient to avoid division by zero (numerical stability).

39.461.2.2 RandomizedSVD() [2/2]

```

RandomizedSVD (
    const size_t iteratedPower = 0,
    const size_t maxIterations = 2,
    const double eps = 1e-7 )

```

Create object for the randomized SVD method.

Parameters

<i>iteratedPower</i>	Size of the normalized power iterations (Default: rank + 2).
<i>maxIterations</i>	Number of iterations for the power method (Default: 2).
<i>eps</i>	The eps coefficient to avoid division by zero (numerical stability).

39.461.3 Member Function Documentation**39.461.3.1 Apply()** [1/3]

```

void Apply (
    const arma::sp_mat & data,
    arma::mat & u,
    arma::vec & s,
    arma::mat & v,
    const size_t rank )

```

Center the data to apply Principal Component Analysis on given sparse matrix dataset using randomized SVD.

Parameters

<i>data</i>	Sparse data matrix.
<i>u</i>	First unitary matrix.
<i>v</i>	Second unitary matrix.
<i>sigma</i>	Diagonal matrix of singular values.
<i>rank</i>	Rank of the approximation.

Referenced by RandomizedSVDPolicy::Apply().

39.461.3.2 Apply() [2/3]

```

void Apply (
    const arma::mat & data,

```

```

    arma::mat & u,
    arma::vec & s,
    arma::mat & v,
    const size_t rank )

```

Center the data to apply Principal Component Analysis on given matrix dataset using randomized SVD.

Parameters

<i>data</i>	Data matrix.
<i>u</i>	First unitary matrix.
<i>v</i>	Second unitary matrix.
<i>sigma</i>	Diagonal matrix of singular values.
<i>rank</i>	Rank of the approximation.

39.461.3.3 Apply() [3/3]

```

void Apply (
    const MatType & data,
    arma::mat & u,
    arma::vec & s,
    arma::mat & v,
    const size_t rank,
    MatType rowMean ) [inline]

```

Apply Principal Component Analysis to the provided matrix data set using the randomized SVD.

Parameters

<i>data</i>	Data matrix.
<i>u</i>	First unitary matrix.
<i>v</i>	Second unitary matrix.
<i>sigma</i>	Diagonal matrix of singular values.
<i>rank</i>	Rank of the approximation.
<i>rowMean</i>	Centered mean value matrix.

Definition at line 151 of file randomized_svd.hpp.

39.461.3.4 Epsilon() [1/2]

```

double Epsilon ( ) const [inline]

```

Get the value used for decomposition stability.

Definition at line 245 of file randomized_svd.hpp.

39.461.3.5 Epsilon() [2/2]

```
double& Epsilon ( ) [inline]
```

Modify the value used for decomposition stability.

Definition at line 247 of file randomized_svd.hpp.

39.461.3.6 IteratedPower() [1/2]

```
size_t IteratedPower ( ) const [inline]
```

Get the size of the normalized power iterations.

Definition at line 235 of file randomized_svd.hpp.

39.461.3.7 IteratedPower() [2/2]

```
size_t& IteratedPower ( ) [inline]
```

Modify the size of the normalized power iterations.

Definition at line 237 of file randomized_svd.hpp.

39.461.3.8 MaxIterations() [1/2]

```
size_t MaxIterations ( ) const [inline]
```

Get the number of iterations for the power method.

Definition at line 240 of file randomized_svd.hpp.

39.461.3.9 MaxIterations() [2/2]

```
size_t& MaxIterations ( ) [inline]
```

Modify the number of iterations for the power method.

Definition at line 242 of file randomized_svd.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/randomized_svd/ **randomized_svd.hpp**

39.462 RegularizedSVD< OptimizerType > Class Template Reference

Regularized SVD is a matrix factorization technique that seeks to reduce the error on the training set, that is on the examples for which the ratings have been provided by the users.

Public Member Functions

- **RegularizedSVD** (const size_t iterations=10, const double alpha=0.01, const double lambda=0.02)
Constructor for Regularized SVD.
- void **Apply** (const arma::mat &data, const size_t rank, arma::mat &u, arma::mat &v)
Obtains the user and item matrices using the provided data and rank.

39.462.1 Detailed Description

```
template<typename OptimizerType = ens::StandardSGD>
class mlpack::svd::RegularizedSVD< OptimizerType >
```

Regularized SVD is a matrix factorization technique that seeks to reduce the error on the training set, that is on the examples for which the ratings have been provided by the users.

It is a fairly straightforward technique where the user and item matrices are updated with the help of Stochastic Gradient Descent(SGD) updates. The updates also penalize the learning of large feature values by means of regularization. More details can be found in the following links:

<http://sifter.org/~simon/journal/20061211.html> <http://www.cs.uic.edu/~liub/KDD-cup-2007/proceedings/Regular-Paterek.pdf>

An example of how to use the interface is shown below:

```
arma::mat data; // Rating data in the form of coordinate list.

const size_t rank = 20; // Rank used for the decomposition.
const size_t iterations = 10; // Number of iterations used for optimization.

const double alpha = 0.01 // Learning rate for the SGD optimizer.
const double lambda = 0.1 // Regularization parameter for the optimization.

// Make a RegularizedSVD object.
RegularizedSVD<> rSVD(iterations, alpha, lambda);

arma::mat u, v; // User and item matrices.

// Use the Apply() method to get a factorization.
rSVD.Apply(data, rank, u, v);
```

Definition at line 58 of file regularized_svd.hpp.

39.462.2 Constructor & Destructor Documentation

39.462.2.1 RegularizedSVD()

```
RegularizedSVD (
    const size_t iterations = 10,
    const double alpha = 0.01,
    const double lambda = 0.02 )
```

Constructor for Regularized SVD.

Obtains the user and item matrices after training on the passed data. The constructor initiates an object of class **RegularizedSVDFunction** (p. 1945) for optimization. It uses the SGD optimizer by default. The optimizer uses a template specialization of `Optimize()`.

Parameters

<i>iterations</i>	Number of optimization iterations.
<i>alpha</i>	Learning rate for the SGD optimizer.
<i>lambda</i>	Regularization parameter for the optimization.

39.462.3 Member Function Documentation

39.462.3.1 Apply()

```
void Apply (
    const arma::mat & data,
    const size_t rank,
    arma::mat & u,
    arma::mat & v )
```

Obtains the user and item matrices using the provided data and rank.

Parameters

<i>data</i>	Rating data matrix.
<i>rank</i>	Rank parameter to be used for optimization.
<i>u</i>	Item matrix obtained on decomposition.
<i>v</i>	User matrix obtained on decomposition.

Referenced by RegSVDPolicy::Apply().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/regularized_svd/ **regularized_svd.hpp**

39.463 RegularizedSVDFunction< MatType > Class Template Reference

The data is stored in a matrix of type MatType, so that this class can be used with both dense and sparse matrix types.

Public Member Functions

- **RegularizedSVDFunction** (const MatType &data, const size_t rank, const double lambda)
*Constructor for **RegularizedSVDFunction** (p. 1945) class.*
- const arma::mat & **Dataset** () const
Return the dataset passed into the constructor.
- double **Evaluate** (const arma::mat ¶meters) const
Evaluates the cost function over all examples in the data.
- double **Evaluate** (const arma::mat ¶meters, const size_t start, const size_t batchSize=1) const
Evaluates the cost function for one training example.
- const arma::mat & **GetInitialPoint** () const
Return the initial point for the optimization.
- void **Gradient** (const arma::mat ¶meters, arma::mat &gradient) const
Evaluates the full gradient of the cost function over all the training examples.
- template<typename GradType >
void **Gradient** (const arma::mat ¶meters, const size_t start, GradType &gradient, const size_t batchSize=1) const
Evaluates the gradient of the cost function over one training example.
- double **Lambda** () const
Return the regularization parameters.
- size_t **NumFunctions** () const
Return the number of training examples. Useful for SGD optimizer.
- size_t **NumItems** () const
Return the number of items in the data.
- size_t **NumUsers** () const
Return the number of users in the data.
- size_t **Rank** () const
Return the rank used for the factorization.
- void **Shuffle** ()
Shuffle the points in the dataset.

39.463.1 Detailed Description

```
template<typename MatType = arma::mat>
class mlpack::svd::RegularizedSVDFunction< MatType >
```

The data is stored in a matrix of type MatType, so that this class can be used with both dense and sparse matrix types.

Template Parameters

<i>MatType</i>	The matrix type of the dataset.
----------------	---------------------------------

Definition at line 29 of file regularized_svd_function.hpp.

39.463.2 Constructor & Destructor Documentation

39.463.2.1 RegularizedSVDFunction()

```
RegularizedSVDFunction (
    const MatType & data,
    const size_t rank,
    const double lambda )
```

Constructor for **RegularizedSVDFunction** (p. 1945) class.

The constructor calculates the number of users and items in the passed data. It also randomly initializes the parameter values.

Parameters

<i>data</i>	Dataset for which SVD is calculated.
<i>rank</i>	Rank used for matrix factorization.
<i>lambda</i>	Regularization parameter used for optimization.

39.463.3 Member Function Documentation

39.463.3.1 Dataset()

```
const arma::mat& Dataset ( ) const [inline]
```

Return the dataset passed into the constructor.

Definition at line 101 of file regularized_svd_function.hpp.

39.463.3.2 Evaluate() [1/2]

```
double Evaluate (
    const arma::mat & parameters ) const
```

Evaluates the cost function over all examples in the data.

Parameters

<i>parameters</i>	Parameters(user/item matrices) of the decomposition.
-------------------	--

39.463.3.3 Evaluate() [2/2]

```
double Evaluate (
    const arma::mat & parameters,
    const size_t start,
    const size_t batchSize = 1 ) const
```

Evaluates the cost function for one training example.

Useful for the SGD optimizer abstraction which uses one training example at a time.

Parameters

<i>parameters</i>	Parameters(user/item matrices) of the decomposition.
<i>start</i>	First index of the training examples to be used.
<i>batchSize</i>	Size of batch to evaluate.

39.463.3.4 GetInitialPoint()

```
const arma::mat& GetInitialPoint ( ) const [inline]
```

Return the initial point for the optimization.

Definition at line 98 of file regularized_svd_function.hpp.

39.463.3.5 Gradient() [1/2]

```
void Gradient (
    const arma::mat & parameters,
    arma::mat & gradient ) const
```

Evaluates the full gradient of the cost function over all the training examples.

Parameters

<i>parameters</i>	Parameters(user/item matrices) of the decomposition.
<i>gradient</i>	Calculated gradient for the parameters.

39.463.3.6 Gradient() [2/2]

```
void Gradient (
    const arma::mat & parameters,
    const size_t start,
    GradType & gradient,
    const size_t batchSize = 1 ) const
```

Evaluates the gradient of the cost function over one training example.

This function is useful for optimizers like SGD. The type of the gradient parameter is a template argument to allow the computation of a sparse gradient.

Template Parameters

<i>GradType</i>	The type of the gradient out-param.
-----------------	-------------------------------------

Parameters

<i>parameters</i>	Parameters(user/item matrices) of the decomposition.
<i>start</i>	The first index of the training examples to use.
<i>gradient</i>	Calculated gradient for the parameters.
<i>batchSize</i>	Size of batch to calculate gradient for.

39.463.3.7 Lambda()

```
double Lambda ( ) const [inline]
```

Return the regularization parameters.

Definition at line 113 of file regularized_svd_function.hpp.

39.463.3.8 NumFunctions()

```
size_t NumFunctions ( ) const [inline]
```

Return the number of training examples. Useful for SGD optimizer.

Definition at line 104 of file `regularized_svd_function.hpp`.

39.463.3.9 NumItems()

```
size_t NumItems ( ) const [inline]
```

Return the number of items in the data.

Definition at line 110 of file `regularized_svd_function.hpp`.

39.463.3.10 NumUsers()

```
size_t NumUsers ( ) const [inline]
```

Return the number of users in the data.

Definition at line 107 of file `regularized_svd_function.hpp`.

39.463.3.11 Rank()

```
size_t Rank ( ) const [inline]
```

Return the rank used for the factorization.

Definition at line 116 of file `regularized_svd_function.hpp`.

39.463.3.12 Shuffle()

```
void Shuffle ( )
```

Shuffle the points in the dataset.

This may be used by optimizers.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/regularized_svd/regularized_svd_function.hpp`

39.464 SVDPlusPlus< OptimizerType > Class Template Reference

SVD++ is a matrix decomposition technique used in collaborative filtering.

Public Member Functions

- **SVDPlusPlus** (const size_t iterations=10, const double alpha=0.001, const double lambda=0.1)
*Constructor of **SVDPlusPlus** (p. 1951).*
- void **Apply** (const arma::mat &data, const arma::mat &implicitData, const size_t rank, arma::mat &u, arma::mat &v, arma::vec &p, arma::vec &q, arma::mat &y)
Trains the model and obtains user/item matrices, user/item bias, and item implicit matrix.
- void **Apply** (const arma::mat &data, const size_t rank, arma::mat &u, arma::mat &v, arma::vec &p, arma::vec &q, arma::mat &y)
Trains the model and obtains user/item matrices, user/item bias, and item implicit matrix.

Static Public Member Functions

- static void **CleanData** (const arma::mat &implicitData, arma::sp_mat &cleanedData, const arma::mat &data)
Converts the User, Item matrix of implicit data to Item-User Table.

39.464.1 Detailed Description

```
template<typename OptimizerType = ens::StandardSGD>
class mlpack::svd::SVDPlusPlus< OptimizerType >
```

SVD++ is a matrix decomposition technique used in collaborative filtering.

SVD++ is similar to **BiasSVD** (p. 1925), but it is a more expressive model because SVD++ also models implicit feedback. SVD++ outputs user/item latent vectors, user/item bias, and item vectors with regard to implicit feedback. Parameters are optimized by Stochastic Gradient Descent (SGD). The updates also penalize the learning of large feature values by means of regularization.

For more information, see the following paper:

{koren2008factorization, title={Factorization meets the neighborhood: a multifaceted collaborative filtering model}, author={Koren, Yehuda}, booktitle={Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining}, pages={426–434}, year={2008}, organization={ACM} }

An example of how to use the interface is shown below:

```
arma::mat data; // Rating data in the form of coordinate list.

// Implicit feedback data in the form of coordinate list.
arma::mat implicitData;

const size_t rank = 10; // Rank used for the decomposition.
const size_t iterations = 10; // Number of iterations used for optimization.

const double alpha = 0.001 // Learning rate for the SGD optimizer.
const double lambda = 0.1 // Regularization parameter for the optimization.

// Make a SVD++ object.
SVDPlusPlus<> svdPP(iterations, alpha, lambda);

arma::mat u, v; // Item and User matrices.
arma::vec p, q; // Item and User bias.
arma::mat y;    // Item matrix with respect to implicit feedback.

// Use the Apply() method to get a factorization.
svdPP.Apply(data, implicitData, rank, u, v, p, q, y);
```

Definition at line 74 of file svdplusplus.hpp.

39.464.2 Constructor & Destructor Documentation

39.464.2.1 SVDPlusPlus()

```
SVDPlusPlus (
    const size_t iterations = 10,
    const double alpha = 0.001,
    const double lambda = 0.1 )
```

Constructor of **SVDPlusPlus** (p. 1951).

By default SGD optimizer is used in **SVDPlusPlus** (p. 1951). The optimizer uses a template specialization of `Optimize()`.

Parameters

<i>iterations</i>	Number of optimization iterations.
<i>alpha</i>	Learning rate for the SGD optimizer.
<i>lambda</i>	Regularization parameter for the optimization.

39.464.3 Member Function Documentation

39.464.3.1 Apply() [1/2]

```
void Apply (
    const arma::mat & data,
    const arma::mat & implicitData,
    const size_t rank,
    arma::mat & u,
    arma::mat & v,
    arma::vec & p,
    arma::vec & q,
    arma::mat & y )
```

Trains the model and obtains user/item matrices, user/item bias, and item implicit matrix.

Parameters

<i>data</i>	Rating data matrix.
<i>implicitData</i>	Implicit feedback.
<i>rank</i>	Rank parameter to be used for optimization.
<i>u</i>	Item matrix obtained on decomposition.

Parameters

<i>v</i>	User matrix obtained on decomposition.
<i>p</i>	Item bias.
<i>q</i>	User bias.
<i>y</i>	Item matrix with respect to implicit feedback.

Referenced by SVDPlusPlusPolicy::Apply().

39.464.3.2 Apply() [2/2]

```
void Apply (
    const arma::mat & data,
    const size_t rank,
    arma::mat & u,
    arma::mat & v,
    arma::vec & p,
    arma::vec & q,
    arma::mat & y )
```

Trains the model and obtains user/item matrices, user/item bias, and item implicit matrix.

Whether a user rates an item is used as implicit feedback.

Parameters

<i>data</i>	Rating data matrix.
<i>rank</i>	Rank parameter to be used for optimization.
<i>u</i>	Item matrix obtained on decomposition.
<i>v</i>	User matrix obtained on decomposition.
<i>p</i>	Item bias.
<i>q</i>	User bias.
<i>y</i>	Item matrix with respect to implicit feedback. Each column is a latent vector of an item with respect to implicit feedback.

39.464.3.3 CleanData()

```
static void CleanData (
    const arma::mat & implicitData,
    arma::sp_mat & cleanedData,
    const arma::mat & data ) [static]
```

Converts the User, Item matrix of implicit data to Item-User Table.

Referenced by `SVDPlusPlusPolicy::Apply()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/svdplusplus/ svdplusplus.hpp`

39.465 SVDPlusPlusFunction< MatType > Class Template Reference

This class contains methods which are used to calculate the cost of SVD++'s objective function, to calculate gradient of parameters with respect to the objective function, etc.

Public Member Functions

- **SVDPlusPlusFunction** (const MatType &data, const arma::sp_mat &implicitData, const size_t rank, const double lambda)
*Constructor for **SVDPlusPlusFunction** (p. 1954) class.*
- const arma::mat & **Dataset** () const
Return the dataset passed into the constructor.
- double **Evaluate** (const arma::mat ¶meters) const
Evaluates the cost function over all examples in the data.
- double **Evaluate** (const arma::mat ¶meters, const size_t start, const size_t batchSize=1) const
Evaluates the cost function for one training example.
- const arma::mat & **GetInitialPoint** () const
Return the initial point for the optimization.
- void **Gradient** (const arma::mat ¶meters, arma::mat &gradient) const
Evaluates the full gradient of the cost function over all the training examples.
- template<typename GradType >
void **Gradient** (const arma::mat ¶meters, const size_t start, GradType &gradient, const size_t batchSize=1) const
Evaluates the gradient of the cost function over one training example.
- const arma::sp_mat & **ImplicitDataset** () const
Return the implicit data passed into the constructor.
- double **Lambda** () const
Return the regularization parameters.
- size_t **NumFunctions** () const
Return the number of training examples. Useful for SGD optimizer.
- size_t **NumItems** () const
Return the number of items in the data.
- size_t **NumUsers** () const
Return the number of users in the data.
- size_t **Rank** () const
Return the rank used for the factorization.
- void **Shuffle** ()
Shuffle the points in the dataset.

39.465.1 Detailed Description

```
template<typename MatType = arma::mat>
class mlpack::svd::SVDPlusPlusFunction< MatType >
```

This class contains methods which are used to calculate the cost of SVD++'s objective function, to calculate gradient of parameters with respect to the objective function, etc.

Template Parameters

<i>MatType</i>	The matrix type of the dataset.
----------------	---------------------------------

Definition at line 31 of file `svdplusplus_function.hpp`.

39.465.2 Constructor & Destructor Documentation

39.465.2.1 SVDPlusPlusFunction()

```
SVDPlusPlusFunction (
    const MatType & data,
    const arma::sp_mat & implicitData,
    const size_t rank,
    const double lambda )
```

Constructor for **SVDPlusPlusFunction** (p. 1954) class.

The constructor calculates the number of users and items in the passed data. It also randomly initializes the parameter values.

Parameters

<i>data</i>	Dataset for which SVD is calculated.
<i>implicitData</i>	Implicit feedback matrix where a non-zero entry means interaction between a user and an item is observed.
<i>rank</i>	Rank used for matrix factorization.
<i>lambda</i>	Regularization parameter used for optimization.

39.465.3 Member Function Documentation

39.465.3.1 Dataset()

```
const arma::mat& Dataset ( ) const [inline]
```

Return the dataset passed into the constructor.

Definition at line 110 of file svdplusplus_function.hpp.

39.465.3.2 Evaluate() [1/2]

```
double Evaluate (
    const arma::mat & parameters ) const
```

Evaluates the cost function over all examples in the data.

Parameters

<i>parameters</i>	Parameters(user/item matrices, user/item bias, item implicit matrix) of the decomposition.
-------------------	--

39.465.3.3 Evaluate() [2/2]

```
double Evaluate (
    const arma::mat & parameters,
    const size_t start,
    const size_t batchSize = 1 ) const
```

Evaluates the cost function for one training example.

Useful for the SGD optimizer abstraction which uses one training example at a time.

Parameters

<i>parameters</i>	Parameters(user/item matrices, user/item bias, item implicit matrix) of the decomposition.
<i>start</i>	First index of the training examples to be used.
<i>batchSize</i>	Size of batch to evaluate.

39.465.3.4 GetInitialPoint()

```
const arma::mat& GetInitialPoint ( ) const [inline]
```

Return the initial point for the optimization.

Definition at line 107 of file `svdplusplus_function.hpp`.

39.465.3.5 Gradient() [1/2]

```
void Gradient (
    const arma::mat & parameters,
    arma::mat & gradient ) const
```

Evaluates the full gradient of the cost function over all the training examples.

Parameters

<i>parameters</i>	Parameters(user/item matrices, user/item bias, item implicit matrix) of the decomposition.
<i>gradient</i>	Calculated gradient for the parameters.

39.465.3.6 Gradient() [2/2]

```
void Gradient (
    const arma::mat & parameters,
    const size_t start,
    GradType & gradient,
    const size_t batchSize = 1 ) const
```

Evaluates the gradient of the cost function over one training example.

This function is useful for optimizers like SGD. The type of the gradient parameter is a template argument to allow the computation of a sparse gradient.

Template Parameters

<i>GradType</i>	The type of the gradient out-param.
-----------------	-------------------------------------

Parameters

<i>parameters</i>	Parameters(user/item matrices, user/item bias, item implicit matrix) of the decomposition.
<i>start</i>	The first index of the training examples to use.
<i>gradient</i>	Calculated gradient for the parameters.
<i>batchSize</i>	Size of batch to calculate gradient for.

39.465.3.7 ImplicitDataset()

```
const arma::sp_mat& ImplicitDataset ( ) const [inline]
```

Return the implicit data passed into the constructor.

Definition at line 113 of file svdplusplus_function.hpp.

39.465.3.8 Lambda()

```
double Lambda ( ) const [inline]
```

Return the regularization parameters.

Definition at line 125 of file svdplusplus_function.hpp.

39.465.3.9 NumFunctions()

```
size_t NumFunctions ( ) const [inline]
```

Return the number of training examples. Useful for SGD optimizer.

Definition at line 116 of file svdplusplus_function.hpp.

39.465.3.10 NumItems()

```
size_t NumItems ( ) const [inline]
```

Return the number of items in the data.

Definition at line 122 of file svdplusplus_function.hpp.

39.465.3.11 NumUsers()

```
size_t NumUsers ( ) const [inline]
```

Return the number of users in the data.

Definition at line 119 of file svdplusplus_function.hpp.

39.465.3.12 Rank()

```
size_t Rank ( ) const [inline]
```

Return the rank used for the factorization.

Definition at line 128 of file svdplusplus_function.hpp.

39.465.3.13 Shuffle()

```
void Shuffle ( )
```

Shuffle the points in the dataset.

This may be used by optimizers.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/svdplusplus/ **svdplusplus_function.hpp**

39.466 LinearSVM< MatType > Class Template Reference

The **LinearSVM** (p. 1959) class implements an L2-regularized support vector machine model, and supports training with multiple optimizers and classification.

Public Member Functions

- template<typename OptimizerType = ens::L_BFGS>
LinearSVM (const MatType &data, const arma::Row< size_t > &labels, const size_t numClasses=2, const double lambda=0.0001, const double delta=1.0, const bool fitIntercept=false, OptimizerType optimizer=OptimizerType())
*Construct the **LinearSVM** (p. 1959) class with the provided data and labels.*
- **LinearSVM** (const size_t inputSize, const size_t numClasses=0, const double lambda=0.0001, const double delta=1.0, const bool fitIntercept=false)
Initialize the Linear SVM without performing training.
- void **Classify** (const MatType &data, arma::Row< size_t > &labels) const
Classify the given points, returning the predicted labels for each point.
- void **Classify** (const MatType &data, arma::Row< size_t > &labels, arma::mat &scores) const
Classify the given points, returning class scores and predicted class label for each point.
- void **Classify** (const MatType &data, arma::mat &scores) const
Classify the given points, returning class scores for each point.
- template<typename VecType >
size_t **Classify** (const VecType &point) const
Classify the given point.

- double **ComputeAccuracy** (const MatType &testData, const arma::Row< size_t > &testLabels) const
Computes accuracy of the learned model given the feature data and the labels associated with each data point.
- size_t **FeatureSize** () const
Gets the features size of the training data.
- double & **Lambda** ()
Sets the regularization parameter.
- double **Lambda** () const
Gets the regularization parameter.
- size_t & **NumClasses** ()
Sets the number of classes.
- size_t **NumClasses** () const
Gets the number of classes.
- arma::mat & **Parameters** ()
Set the model parameters.
- const arma::mat & **Parameters** () const
Get the model parameters.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
*Serialize the **LinearSVM** (p. 1959) model.*
- template<typename OptimizerType = ens::L_BFGS>
double **Train** (const MatType &data, const arma::Row< size_t > &labels, const size_t numClasses=2, OptimizerType optimizer=OptimizerType())
Train the Linear SVM with the given training data.

39.466.1 Detailed Description

```
template<typename MatType = arma::mat>
class mlpack::svm::LinearSVM< MatType >
```

The **LinearSVM** (p. 1959) class implements an L2-regularized support vector machine model, and supports training with multiple optimizers and classification.

The class supports different observation types via the MatType template parameter; for instance, support vector classification can be performed on sparse datasets by specifying arma::sp_mat as the MatType parameter.

Linear SVM can be used for general classification tasks which will work on multiclass classification. More technical details about the model can be found from the following:

```
@inproceedings{weston1999support,
  title      = {Support vector machines for multi-class pattern
               recognition.},
  author     = {Weston, Jason and Watkins, Chris},
  booktitle  = {Proceedings of the 7th European Symposium on Artificial Neural
               Networks (ESANN '99)},
  volume     = {99},
  pages      = {219--224},
  year       = {1999}
}
```



```
@article{cortes1995support,
title      = {Support-vector networks},
author     = {Cortes, Corinna and Vapnik, Vladimir},
journal    = {Machine Learning},
volume     = {20},
number     = {3},
pages      = {273--297},
year       = {1995},
publisher  = {Springer}
}
```

An example on how to use the interface is shown below:

```
arma::mat train_data; // Training data matrix.
arma::Row<size_t> labels; // Labels associated with the data.
const size_t inputSize = 1000; // Size of input feature vector.
const size_t numClasses = 5; // Number of classes.

// Train the model using default options.
LinearSVM<> lsvm(train_data, labels, inputSize, numClasses, lambda,
    delta, L_BFGS());

arma::mat test_data;
arma::Row<size_t> predictions;
lsvm.Classify(test_data, predictions);
```

Template Parameters

<i>MatType</i>	Type of data matrix.
----------------	----------------------

Definition at line 80 of file linear_svm.hpp.

39.466.2 Constructor & Destructor Documentation

39.466.2.1 LinearSVM() [1/2]

```
LinearSVM (
    const MatType & data,
    const arma::Row< size_t > & labels,
    const size_t numClasses = 2,
    const double lambda = 0.0001,
    const double delta = 1.0,
    const bool fitIntercept = false,
    OptimizerType optimizer = OptimizerType() )
```

Construct the **LinearSVM** (p. 1959) class with the provided data and labels.

This will train the model. Optionally, the parameter 'lambda' can be passed, which controls the amount of L2-regularization in the objective function. By default, the model takes a small value.

Template Parameters

<i>OptimizerType</i>	Desired differentiable separable optimizer
----------------------	--

Parameters

<i>data</i>	Input training features. Each column associate with one sample
<i>labels</i>	Labels associated with the feature data.
<i>numClasses</i>	Number of classes for classification.
<i>lambda</i>	L2-regularization constant. delta Margin of difference between correct class and other classes.
<i>optimizer</i>	Desired optimizer.

39.466.2.2 LinearSVM() [2/2]

```

LinearSVM (
    const size_t inputSize,
    const size_t numClasses = 0,
    const double lambda = 0.0001,
    const double delta = 1.0,
    const bool fitIntercept = false )

```

Initialize the Linear SVM without performing training.

Default value of lambda is 0.0001. Be sure to use **Train()** (p. 1966) before calling **Classify()** (p. 1962) or **ComputeAccuracy()** (p. 1964), otherwise the results may be meaningless.

Parameters

<i>inputSize</i>	Size of the input feature vector.
<i>numClasses</i>	Number of classes for classification.
<i>lambda</i>	L2-regularization constant. delta Margin of difference between correct class and other classes.
<i>fitIntercept</i>	add intercept term or not.

39.466.3 Member Function Documentation

39.466.3.1 Classify() [1/4]

```

void Classify (
    const MatType & data,
    arma::Row< size_t > & labels ) const

```

Classify the given points, returning the predicted labels for each point.

The function calculates the probabilities for every class, given a data point. It then chooses the class which has the highest probability among all.

Parameters

<i>data</i>	Set of points to classify.
<i>labels</i>	Predicted labels for each point.

39.466.3.2 Classify() [2/4]

```
void Classify (
    const MatType & data,
    arma::Row< size_t > & labels,
    arma::mat & scores ) const
```

Classify the given points, returning class scores and predicted class label for each point.

The function calculates the scores for every class, given a data point. It then chooses the class which has the highest probability among all.

Parameters

<i>data</i>	Matrix of data points to be classified.
<i>labels</i>	Predicted labels for each point.
<i>scores</i>	Class probabilities for each point.

39.466.3.3 Classify() [3/4]

```
void Classify (
    const MatType & data,
    arma::mat & scores ) const
```

Classify the given points, returning class scores for each point.

Parameters

<i>data</i>	Matrix of data points to be classified.
<i>scores</i>	Class scores for each point.

39.466.3.4 Classify() [4/4]

```
size_t Classify (
    const VecType & point ) const
```

Classify the given point.

The predicted class label is returned. The function calculates the scores for every class, given the point. It then chooses the class which has the highest probability among all.

Parameters

<i>point</i>	Point to be classified.
--------------	-------------------------

Returns

Predicted class label of the point.

39.466.3.5 ComputeAccuracy()

```
double ComputeAccuracy (
    const MatType & testData,
    const arma::Row< size_t > & testLabels ) const
```

Computes accuracy of the learned model given the feature data and the labels associated with each data point.

Predictions are made using the provided data and are compared with the actual labels.

Parameters

<i>testData</i>	Matrix of data points using which predictions are made.
<i>testLabels</i>	Vector of labels associated with the data.

Returns

Accuracy of the model.

39.466.3.6 FeatureSize()

```
size_t FeatureSize ( ) const [inline]
```

Gets the features size of the training data.

Definition at line 216 of file linear_svm.hpp.

39.466.3.7 Lambda() [1/2]

```
double& Lambda ( ) [inline]
```

Sets the regularization parameter.

Definition at line 206 of file linear_svm.hpp.

39.466.3.8 Lambda() [2/2]

```
double Lambda ( ) const [inline]
```

Gets the regularization parameter.

Definition at line 208 of file linear_svm.hpp.

39.466.3.9 NumClasses() [1/2]

```
size_t& NumClasses ( ) [inline]
```

Sets the number of classes.

Definition at line 201 of file linear_svm.hpp.

39.466.3.10 NumClasses() [2/2]

```
size_t NumClasses ( ) const [inline]
```

Gets the number of classes.

Definition at line 203 of file linear_svm.hpp.

39.466.3.11 Parameters() [1/2]

```
arma::mat& Parameters ( ) [inline]
```

Set the model parameters.

Definition at line 211 of file linear_svm.hpp.

39.466.3.12 Parameters() [2/2]

```
const arma::mat& Parameters ( ) const [inline]
```

Get the model parameters.

Definition at line 213 of file linear_svm.hpp.

39.466.3.13 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the **LinearSVM** (p. 1959) model.

Definition at line 224 of file linear_svm.hpp.

39.466.3.14 Train()

```
double Train (
    const MatType & data,
    const arma::Row< size_t > & labels,
    const size_t numClasses = 2,
    OptimizerType optimizer = OptimizerType() )
```

Train the Linear SVM with the given training data.

Template Parameters

<i>OptimizerType</i>	Desired optimizer
----------------------	-------------------

Parameters

<i>data</i>	Input training features. Each column associate with one sample
<i>labels</i>	Labels associated with the feature data.
<i>numClasses</i>	Number of classes for classification.
<i>lambda</i>	L2-regularization constant.
<i>optimizer</i>	Desired optimizer.

Returns

Objective value of the final point.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear_svm/ **linear_svm.hpp**

39.467 LinearSVMFunction< MatType > Class Template Reference

The hinge loss function for the linear SVM objective function.

Public Member Functions

- **LinearSVMFunction** (const MatType &dataset, const arma::Row< size_t > &labels, const size_t numClasses, const double lambda=0.0001, const double delta=1.0, const bool fitIntercept=false)
Construct the Linear SVM objective function with given parameters.
- const arma::sp_mat & **Dataset** () const
Get the dataset.
- arma::sp_mat & **Dataset** ()
Modify the dataset.
- double **Evaluate** (const arma::mat ¶meters)
Evaluate the hinge loss function for all the datapoints.
- double **Evaluate** (const arma::mat ¶meters, const size_t firstId, const size_t batchSize=1)
Evaluate the hinge loss function on the specified datapoints.
- template<typename GradType >
double **EvaluateWithGradient** (const arma::mat ¶meters, GradType &gradient) const
Evaluate the gradient of the hinge loss function, following the LinearFunctionType requirements on the Gradient function followed by evaluation of the hinge loss function on all the datapoints.
- template<typename GradType >
double **EvaluateWithGradient** (const arma::mat ¶meters, const size_t firstId, GradType &gradient, const size_t batchSize=1) const
Evaluate the gradient of the hinge loss function, following the LinearFunctionType requirements on the Gradient function followed by evaluation of the hinge loss function on the specified datapoints.
- bool **FitIntercept** () const
Gets the intercept flag.
- void **GetGroundTruthMatrix** (const arma::Row< size_t > &labels, arma::sp_mat &groundTruth)
Constructs the ground truth label matrix with the passed labels.
- template<typename GradType >
void **Gradient** (const arma::mat ¶meters, GradType &gradient)
Evaluate the gradient of the hinge loss function following the LinearFunctionType requirements on the Gradient function.
- template<typename GradType >
void **Gradient** (const arma::mat ¶meters, const size_t firstId, GradType &gradient, const size_t batchSize=1)
Evaluate the gradient of the hinge loss function, following the LinearFunctionType requirements on the Gradient function.
- const arma::mat & **InitialPoint** () const
Return the initial point for the optimization.

- `arma::mat & InitialPoint ()`
Modify the initial point for the optimization.
- `double & Lambda ()`
Sets the regularization parameter.
- `double Lambda () const`
Gets the regularization parameter.
- `size_t NumFunctions () const`
Return the number of functions.
- `void Shuffle ()`
Shuffle the dataset.

Static Public Member Functions

- `static void InitializeWeights (arma::mat &weights, const size_t featureSize, const size_t numClasses, const bool fitIntercept=false)`
Initialize Linear SVM weights (trainable parameters) with the given parameters.

39.467.1 Detailed Description

```
template<typename MatType = arma::mat>
class mlpack::svm::LinearSVMFunction< MatType >
```

The hinge loss function for the linear SVM objective function.

This is used by various ensmallen optimizers to train the linear SVM model.

Definition at line 28 of file `linear_svm_function.hpp`.

39.467.2 Constructor & Destructor Documentation

39.467.2.1 LinearSVMFunction()

```
LinearSVMFunction (
    const MatType & dataset,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const double lambda = 0.0001,
    const double delta = 1.0,
    const bool fitIntercept = false )
```

Construct the Linear SVM objective function with given parameters.

Parameters

<i>dataset</i>	Input training data, each column associate with one sample
<i>labels</i>	Labels associated with the feature data.
<i>numClasses</i>	Number of classes for classification.
<i>lambda</i>	L2-regularization constant. delta Margin of difference between correct class and other classes.
<i>fitIntercept</i>	Intercept term flag.

39.467.3 Member Function Documentation

39.467.3.1 Dataset() [1/2]

```
const arma::sp_mat& Dataset ( ) const [inline]
```

Get the dataset.

Definition at line 167 of file linear_svm_function.hpp.

39.467.3.2 Dataset() [2/2]

```
arma::sp_mat& Dataset ( ) [inline]
```

Modify the dataset.

Definition at line 169 of file linear_svm_function.hpp.

39.467.3.3 Evaluate() [1/2]

```
double Evaluate (
    const arma::mat & parameters )
```

Evaluate the hinge loss function for all the datapoints.

Parameters

<i>parameters</i>	The parameters of the SVM.
-------------------	----------------------------

Returns

The value of the loss function for the entire dataset.

39.467.3.4 Evaluate() [2/2]

```
double Evaluate (
    const arma::mat & parameters,
    const size_t firstId,
    const size_t batchSize = 1 )
```

Evaluate the hinge loss function on the specified datapoints.

Parameters

<i>parameters</i>	The parameters of the SVM.
<i>firstId</i>	Index of the datapoints to use for function evaluation.
<i>batchSize</i>	Size of batch to process.

Returns

The value of the loss function for the given parameters.

39.467.3.5 EvaluateWithGradient() [1/2]

```
double EvaluateWithGradient (
    const arma::mat & parameters,
    GradType & gradient ) const
```

Evaluate the gradient of the hinge loss function, following the LinearFunctionType requirements on the Gradient function followed by evaluation of the hinge loss function on all the datapoints.

Template Parameters

<i>GradType</i>	Type of the gradient matrix.
-----------------	------------------------------

Parameters

<i>parameters</i>	The parameters of the SVM.
<i>gradient</i>	Linear matrix to output the gradient into.

Returns

The value of the loss function at the given parameters.

39.467.3.6 EvaluateWithGradient() [2/2]

```
double EvaluateWithGradient (
    const arma::mat & parameters,
    const size_t firstId,
    GradType & gradient,
    const size_t batchSize = 1 ) const
```

Evaluate the gradient of the hinge loss function, following the LinearFunctionType requirements on the Gradient function followed by evaluation of the hinge loss function on the specified datapoints.

Template Parameters

<i>GradType</i>	Type of the gradient matrix.
-----------------	------------------------------

Parameters

<i>parameters</i>	The parameters of the SVM.
<i>firstId</i>	Index of the datapoint to use for the gradient and function evaluation.
<i>gradient</i>	Linear matrix to output the gradient into.
<i>batchSize</i>	Size of the batch to process.

Returns

The value of the loss function at the given parameters.

39.467.3.7 FitIntercept()

```
bool FitIntercept ( ) const [inline]
```

Gets the intercept flag.

Definition at line 177 of file linear_svm_function.hpp.

References LinearSVMFunction< MatType >::NumFunctions().

39.467.3.8 GetGroundTruthMatrix()

```
void GetGroundTruthMatrix (
    const arma::Row< size_t > & labels,
    arma::sp_mat & groundTruth )
```

Constructs the ground truth label matrix with the passed labels.

Parameters

<i>labels</i>	Labels associated with the training data.
<i>groundTruth</i>	Pointer to arma::mat which stores the computed matrix.

39.467.3.9 Gradient() [1/2]

```
void Gradient (
    const arma::mat & parameters,
    GradType & gradient )
```

Evaluate the gradient of the hinge loss function following the LinearFunctionType requirements on the Gradient function.

Template Parameters

<i>GradType</i>	Type of the gradient matrix.
-----------------	------------------------------

Parameters

<i>parameters</i>	The parameters of the SVM.
<i>gradient</i>	Linear matrix to output the gradient into.

39.467.3.10 Gradient() [2/2]

```
void Gradient (
    const arma::mat & parameters,
    const size_t firstId,
    GradType & gradient,
    const size_t batchSize = 1 )
```

Evaluate the gradient of the hinge loss function, following the LinearFunctionType requirements on the Gradient function.

Template Parameters

<i>GradType</i>	Type of the gradient matrix.
-----------------	------------------------------

Parameters

<i>parameters</i>	The parameters of the SVM.
<i>firstId</i>	Index of the datapoint to use for the gradient evaluation.
<i>gradient</i>	Linear matrix to output the gradient into.
<i>batchSize</i>	Size of the batch to process.

39.467.3.11 InitializeWeights()

```
static void InitializeWeights (
    arma::mat & weights,
    const size_t featureSize,
    const size_t numClasses,
    const bool fitIntercept = false ) [static]
```

Initialize Linear SVM weights (trainable parameters) with the given parameters.

Parameters

<i>weights</i>	This will be filled with the initialized model weights.
<i>featureSize</i>	The number of features in the training set.
<i>numClasses</i>	Number of classes for classification.
<i>fitIntercept</i>	If true, an intercept is fitted.

Returns

Initialized model weights.

39.467.3.12 InitialPoint() [1/2]

```
const arma::mat& InitialPoint ( ) const [inline]
```

Return the initial point for the optimization.

Definition at line 162 of file linear_svm_function.hpp.

39.467.3.13 InitialPoint() [2/2]

```
arma::mat& InitialPoint ( ) [inline]
```

Modify the initial point for the optimization.

Definition at line 164 of file linear_svm_function.hpp.

39.467.3.14 Lambda() [1/2]

```
double& Lambda ( ) [inline]
```

Sets the regularization parameter.

Definition at line 172 of file linear_svm_function.hpp.

39.467.3.15 Lambda() [2/2]

```
double Lambda ( ) const [inline]
```

Gets the regularization parameter.

Definition at line 174 of file linear_svm_function.hpp.

39.467.3.16 NumFunctions()

```
size_t NumFunctions ( ) const
```

Return the number of functions.

Referenced by LinearSVMFunction< MatType >::FitIntercept().

39.467.3.17 Shuffle()

```
void Shuffle ( )
```

Shuffle the dataset.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear_svm/ **linear_svm_function.hpp**

39.468 Timer Class Reference

The timer class provides a way for mpack methods to be timed.

Static Public Member Functions

- static void **DisableTiming** ()
Disable timing of mpack programs.
- static void **EnableTiming** ()
Enable timing of mpack programs.
- static std::chrono::microseconds **Get** (const std::string &name)
Get the value of the given timer.
- static void **ResetAll** ()
Stop and reset all running timers.
- static void **Start** (const std::string &name)
Start the given timer.
- static void **Stop** (const std::string &name)
Stop the given timer.

39.468.1 Detailed Description

The timer class provides a way for mpack methods to be timed.

The three methods contained in this class allow a named timer to be started and stopped, and its value to be obtained. A named timer is specific to the thread it is running on, so if you start a timer in one thread, it cannot be stopped from a different thread.

Definition at line 45 of file timers.hpp.

39.468.2 Member Function Documentation

39.468.2.1 DisableTiming()

```
static void DisableTiming ( ) [static]
```

Disable timing of mpack programs.

Do not run this while timers are running!

39.468.2.2 EnableTiming()

```
static void EnableTiming ( ) [static]
```

Enable timing of mpack programs.

Do not run this while timers are running!

Referenced by mpack::util::EnableTimers().

39.468.2.3 Get()

```
static std::chrono::microseconds Get (
    const std::string & name ) [static]
```

Get the value of the given timer.

Parameters

<i>name</i>	Name of timer to return value of.
-------------	-----------------------------------

39.468.2.4 ResetAll()

```
static void ResetAll ( ) [static]
```

Stop and reset all running timers.

This removes all knowledge of any existing timers.

39.468.2.5 Start()

```
static void Start (
    const std::string & name ) [static]
```

Start the given timer.

If a timer is started, then stopped, then re-started, then re-stopped, the final value of the timer is the length of both runs – that is, mpack timers are additive for each time they are run, and do not reset.

Note

A `std::runtime_error` exception will be thrown if a timer is started twice.

Parameters

<i>name</i>	Name of timer to be started.
-------------	------------------------------

39.468.2.6 Stop()

```
static void Stop (
    const std::string & name ) [static]
```

Stop the given timer.

Note

A `std::runtime_error` exception will be thrown if a timer is started twice.

Parameters

<i>name</i>	Name of timer to be stopped.
-------------	------------------------------

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **timers.hpp**

39.469 Timers Class Reference

Public Member Functions

- **Timers** ()
Default to disabled.
- `std::atomic< bool > & Enabled ()`
Modify whether or not timing is enabled.
- `bool Enabled () const`
Get whether or not timing is enabled.
- `std::map< std::string, std::chrono::microseconds > GetAllTimers ()`
Returns a copy of all the timers used via this interface.
- `bool GetState (const std::string &timerName, const std::thread::id &threadId=std::thread::id())`
Returns state of the given timer.
- `std::chrono::microseconds GetTimer (const std::string &timerName)`
Returns a copy of the timer specified.
- `void PrintTimer (const std::string &timerName)`
Prints the specified timer.

- void **Reset** ()
Reset the timers.
- void **StartTimer** (const std::string &timerName, const std::thread::id &threadId=std::thread::id())
** Initializes a timer, available like a normal value specified on * the command line.*
- void **StopAllTimers** ()
Stop all timers.
- void **StopTimer** (const std::string &timerName, const std::thread::id &threadId=std::thread::id())
** Halts the timer, and replaces its value with the delta time from its start.*

39.469.1 Detailed Description

Definition at line 97 of file timers.hpp.

39.469.2 Constructor & Destructor Documentation

39.469.2.1 Timers()

```
Timers ( ) [inline]
```

Default to disabled.

Definition at line 101 of file timers.hpp.

39.469.3 Member Function Documentation

39.469.3.1 Enabled() [1/2]

```
std::atomic<bool>& Enabled ( ) [inline]
```

Modify whether or not timing is enabled.

Definition at line 166 of file timers.hpp.

39.469.3.2 Enabled() [2/2]

```
bool Enabled ( ) const [inline]
```

Get whether or not timing is enabled.

Definition at line 168 of file timers.hpp.

39.469.3.3 GetAllTimers()

```
std::map<std::string, std::chrono::microseconds> GetAllTimers ( )
```

Returns a copy of all the timers used via this interface.

Referenced by `mlpack::bindings::cli::EndProgram()`.

39.469.3.4 GetState()

```
bool GetState (
    const std::string & timerName,
    const std::thread::id & threadId = std::thread::id() )
```

Returns state of the given timer.

Parameters

<i>timerName</i>	The name of the timer in question.
<i>threadId</i>	Id of the thread accessing the timer.

39.469.3.5 GetTimer()

```
std::chrono::microseconds GetTimer (
    const std::string & timerName )
```

Returns a copy of the timer specified.

This contains the sum of the timing results for timers that have been stopped with this name.

Parameters

<i>timerName</i>	The name of the timer in question.
------------------	------------------------------------

39.469.3.6 PrintTimer()

```
void PrintTimer (
    const std::string & timerName )
```

Prints the specified timer.

If it took longer than a minute to complete the timer will be displayed in days, hours, and minutes as well.

Parameters

<i>timerName</i>	The name of the timer in question.
------------------	------------------------------------

Referenced by `mlpack::bindings::cli::EndProgram()`.

39.469.3.7 Reset()

```
void Reset ( )
```

Reset the timers.

This stops all running timers and removes them. Whether or not timing is enabled will not be changed.

Referenced by `mlpack::util::ResetTimers()`.

39.469.3.8 StartTimer()

```
void StartTimer (
    const std::string & timerName,
    const std::thread::id & threadId = std::thread::id() )
```

* Initializes a timer, available like a normal value specified on * the command line.

Timers (p. 1977) are of type `timeval`. If a timer is started, then stopped, then re-started, then stopped, the final timer value will be the length of both runs of the timer. * *

Parameters

<i>timerName</i>	The name of the timer in question.
<i>threadId</i>	Id of the thread accessing the timer.

39.469.3.9 StopAllTimers()

```
void StopAllTimers ( )
```

Stop all timers.

Referenced by `mlpack::bindings::cli::EndProgram()`.

39.469.3.10 StopTimer()

```
void StopTimer (
    const std::string & timerName,
    const std::thread::id & threadId = std::thread::id() )
```

* Halts the timer, and replaces its value with the delta time from its start.

* *

Parameters

<i>timerName</i>	The name of the timer in question.
<i>threadId</i>	Id of the thread accessing the timer.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ timers.hpp`

39.470 AllCategoricalSplit< FitnessFunction > Class Template Reference

The **AllCategoricalSplit** (p. 1981) is a splitting function that will split categorical features into many children: one child for each category.

Classes

- class **AuxiliarySplitInfo**

Static Public Member Functions

- `template<typename ElemType >`
`static size_t CalculateDirection (const ElemType &point, const arma::Col< ElemType > &classProbabilities,`
`const AuxiliarySplitInfo< ElemType > &)`
Calculate the direction a point should percolate to.
- `template<typename ElemType >`
`static size_t NumChildren (const arma::Col< ElemType > &classProbabilities, const AuxiliarySplitInfo<`
`ElemType > &)`
Return the number of children in the split.
- `template<bool UseWeights, typename VecType , typename WeightVecType >`
`static double SplitIfBetter (const double bestGain, const VecType &data, const size_t numCategories, const`
`arma::Row< size_t > &labels, const size_t numClasses, const WeightVecType &weights, const size_t minimum←`
`LeafSize, const double minimumGainSplit, arma::Col< typename VecType::elem_type > &classProbabilities,`
`AuxiliarySplitInfo< typename VecType::elem_type > &aux)`
Check if we can split a node.

39.470.1 Detailed Description

```
template<typename FitnessFunction>
class mlpack::tree::AllCategoricalSplit< FitnessFunction >
```

The **AllCategoricalSplit** (p. 1981) is a splitting function that will split categorical features into many children: one child for each category.

Template Parameters

<i>FitnessFunction</i>	Fitness function to evaluate gain with.
------------------------	---

Definition at line 28 of file `all_categorical_split.hpp`.

39.470.2 Member Function Documentation

39.470.2.1 CalculateDirection()

```
static size_t CalculateDirection (
    const ElemType & point,
    const arma::Col< ElemType > & classProbabilities,
    const AuxiliarySplitInfo< ElemType > & ) [static]
```

Calculate the direction a point should percolate to.

Parameters

<i>classProbabilities</i>	Auxiliary information for the split.
<i>aux</i>	(Unused) auxiliary information for the split.

39.470.2.2 NumChildren()

```
static size_t NumChildren (
    const arma::Col< ElemType > & classProbabilities,
    const AuxiliarySplitInfo< ElemType > & ) [static]
```

Return the number of children in the split.

Parameters

<i>classProbabilities</i>	Auxiliary information for the split.
<i>aux</i>	(Unused) auxiliary information for the split.

39.470.2.3 SplitIfBetter()

```
static double SplitIfBetter (
    const double bestGain,
    const VecType & data,
    const size_t numCategories,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const WeightVecType & weights,
    const size_t minimumLeafSize,
    const double minimumGainSplit,
    arma::Col< typename VecType::elem_type > & classProbabilities,
    AuxiliarySplitInfo< typename VecType::elem_type > & aux ) [static]
```

Check if we can split a node.

If we can split a node in a way that improves on 'bestGain', then we return the improved gain. Otherwise we return the value 'bestGain'. If a split is made, then classProbabilities and aux may be modified. For this particular split type, aux will be empty and classProbabilities will hold one element—the number of children.

Parameters

<i>bestGain</i>	Best gain seen so far (we'll only split if we find gain better than this).
<i>data</i>	The dimension of data points to check for a split in.
<i>numCategories</i>	Number of categories in the categorical data.

Parameters

<i>labels</i>	Labels for each point.
<i>numClasses</i>	Number of classes in the dataset.
<i>minimumLeafSize</i>	Minimum number of points in a leaf node for splitting.
<i>classProbabilities</i>	Class probabilities vector, which may be filled with split information a successful split.
<i>aux</i>	Auxiliary split information, which may be modified on a successful split.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/ **all_categorical_split.hpp**

39.471 AllCategoricalSplit< FitnessFunction >::AuxiliarySplitInfo< ElemType > Class Template Reference

39.471.1 Detailed Description

```
template<typename FitnessFunction>
template<typename ElemType>
class mlpack::tree::AllCategoricalSplit< FitnessFunction >::AuxiliarySplitInfo< ElemType >
```

Definition at line 33 of file all_categorical_split.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/ **all_categorical_split.hpp**

39.472 AllDimensionSelect Class Reference

This dimension selection policy allows any dimension to be selected for splitting.

Public Member Functions

- **AllDimensionSelect** ()
*Construct the **AllDimensionSelect** (p. 1984) object for the given number of dimensions.*
- size_t **Begin** ()
Get the first dimension to select from.
- size_t **Dimensions** () const
Get the number of dimensions.
- size_t & **Dimensions** ()
Modify the number of dimensions.
- size_t **End** () const
Get the last dimension to select from.
- size_t **Next** ()
Get the next dimension.

39.472.1 Detailed Description

This dimension selection policy allows any dimension to be selected for splitting.

Definition at line 22 of file all_dimension_select.hpp.

39.472.2 Constructor & Destructor Documentation

39.472.2.1 AllDimensionSelect()

```
AllDimensionSelect ( ) [inline]
```

Construct the **AllDimensionSelect** (p. 1984) object for the given number of dimensions.

Definition at line 28 of file all_dimension_select.hpp.

39.472.3 Member Function Documentation

39.472.3.1 Begin()

```
size_t Begin ( ) [inline]
```

Get the first dimension to select from.

Definition at line 33 of file all_dimension_select.hpp.

39.472.3.2 Dimensions() [1/2]

```
size_t Dimensions ( ) const [inline]
```

Get the number of dimensions.

Definition at line 50 of file all_dimension_select.hpp.

39.472.3.3 Dimensions() [2/2]

```
size_t& Dimensions ( ) [inline]
```

Modify the number of dimensions.

Definition at line 52 of file all_dimension_select.hpp.

39.472.3.4 End()

```
size_t End ( ) const [inline]
```

Get the last dimension to select from.

Definition at line 42 of file all_dimension_select.hpp.

39.472.3.5 Next()

```
size_t Next ( ) [inline]
```

Get the next dimension.

Definition at line 47 of file all_dimension_select.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/ **all_dimension_select.hpp**

39.473 AxisParallelProjVector Class Reference

AxisParallelProjVector (p. 1986) defines an axis-parallel projection vector.

Public Member Functions

- **AxisParallelProjVector** (size_t dim=0)
Create the projection vector based on the specified dimension.
- template<typename VecType >
double **Project** (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > !=0) const
Project the given point on the projection vector.
- template<typename MetricType , typename ElemType >
math::RangeType< ElemType > **Project** (const **bound::HRectBound**< MetricType, ElemType > &bound) const
Project the given hrect bound on the projection vector.
- template<typename MetricType , typename VecType >
math::RangeType< typename VecType::elem_type > **Project** (const **bound::BallBound**< MetricType, VecType > &bound) const
Project the given ball bound on the projection vector.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialization.

39.473.1 Detailed Description

AxisParallelProjVector (p. 1986) defines an axis-parallel projection vector.

We can efficiently project points, simply analyzing a specific dimension.

Definition at line 24 of file projection_vector.hpp.

39.473.2 Constructor & Destructor Documentation

39.473.2.1 AxisParallelProjVector()

```
AxisParallelProjVector (  
    size_t dim = 0 ) [inline]
```

Create the projection vector based on the specified dimension.

Parameters

<i>dim</i>	Dimension to be considered.
------------	-----------------------------

Definition at line 35 of file projection_vector.hpp.

39.473.3 Member Function Documentation

39.473.3.1 Project() [1/3]

```
double Project (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const [inline]
```

Project the given point on the projection vector.

Parameters

<i>point</i>	Point to be projected.
--------------	------------------------

Definition at line 45 of file projection_vector.hpp.

Referenced by ProjVector::Project().

39.473.3.2 Project() [2/3]

```
math::RangeType<ElemType> Project (
    const bound::HRectBound< MetricType, ElemType > & bound ) const [inline]
```

Project the given hrect bound on the projection vector.

Parameters

<i>bound</i>	Bound to be projected.
--------------	------------------------

Returns

Range of projected values.

Definition at line 58 of file projection_vector.hpp.

39.473.3.3 Project() [3/3]

```
math::RangeType<typename VecType::elem_type> Project (
    const bound::BallBound< MetricType, VecType > & bound ) const [inline]
```

Project the given ball bound on the projection vector.

Parameters

<i>bound</i>	Bound to be projected.
--------------	------------------------

Returns

Range of projected values.

Definition at line 71 of file `projection_vector.hpp`.

39.473.3.4 **serialize()**

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialization.

Definition at line 81 of file `projection_vector.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/ projection_vector.hpp`

39.474 **BestBinaryNumericSplit**< **FitnessFunction** > Class Template Reference

The **BestBinaryNumericSplit** (p. 1989) is a splitting function for decision trees that will exhaustively search a numeric dimension for the best binary split.

Classes

- class **AuxiliarySplitInfo**

Static Public Member Functions

- `template<typename ElemType >`
`static size_t CalculateDirection (const ElemType &point, const arma::Col< ElemType > &classProbabilities, const AuxiliarySplitInfo< ElemType > &)`
Given a point, calculate which child it should go to (left or right).
- `template<typename ElemType >`
`static size_t NumChildren (const arma::Col< ElemType > &, const AuxiliarySplitInfo< ElemType > &)`
Returns 2, since the binary split always has two children.
- `template<bool UseWeights, typename VecType, typename WeightVecType >`
`static double SplitIfBetter (const double bestGain, const VecType &data, const arma::Row< size_t > &labels, const size_t numClasses, const WeightVecType &weights, const size_t minimumLeafSize, const double minimumGainSplit, arma::Col< typename VecType::elem_type > &classProbabilities, AuxiliarySplitInfo< typename VecType::elem_type > &aux)`
Check if we can split a node.

39.474.1 Detailed Description

```
template<typename FitnessFunction>
class mlpack::tree::BestBinaryNumericSplit< FitnessFunction >
```

The **BestBinaryNumericSplit** (p. 1989) is a splitting function for decision trees that will exhaustively search a numeric dimension for the best binary split.

Template Parameters

<i>FitnessFunction</i>	Fitness function to use to calculate gain.
------------------------	--

Definition at line 27 of file `best_binary_numeric_split.hpp`.

39.474.2 Member Function Documentation

39.474.2.1 CalculateDirection()

```
static size_t CalculateDirection (
    const ElemType & point,
    const arma::Col< ElemType > & classProbabilities,
    const AuxiliarySplitInfo< ElemType > & ) [static]
```

Given a point, calculate which child it should go to (left or right).

Parameters

<i>point</i>	Point to calculate direction of.
<i>classProbabilities</i>	Auxiliary information for the split.
<i>aux</i>	(Unused) auxiliary information for the split.

Referenced by `BestBinaryNumericSplit< FitnessFunction >::NumChildren()`.

39.474.2.2 NumChildren()

```
static size_t NumChildren (
    const arma::Col< ElemType > & ,
    const AuxiliarySplitInfo< ElemType > & ) [inline], [static]
```

Returns 2, since the binary split always has two children.

Definition at line 69 of file best_binary_numeric_split.hpp.

References BestBinaryNumericSplit< FitnessFunction >::CalculateDirection().

39.474.2.3 SplitIfBetter()

```
static double SplitIfBetter (
    const double bestGain,
    const VecType & data,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const WeightVecType & weights,
    const size_t minimumLeafSize,
    const double minimumGainSplit,
    arma::Col< typename VecType::elem_type > & classProbabilities,
    AuxiliarySplitInfo< typename VecType::elem_type > & aux ) [static]
```

Check if we can split a node.

If we can split a node in a way that improves on 'bestGain', then we return the improved gain. Otherwise we return the value 'bestGain'. If a split is made, then classProbabilities and aux may be modified.

Parameters

<i>bestGain</i>	Best gain seen so far (we'll only split if we find gain better than this).
<i>data</i>	The dimension of data points to check for a split in.
<i>numCategories</i>	Number of categories in the categorical data.
<i>labels</i>	Labels for each point.
<i>numClasses</i>	Number of classes in the dataset.
<i>minimumLeafSize</i>	Minimum number of points in a leaf node for splitting.
<i>classProbabilities</i>	Class probabilities vector, which may be filled with split information a successful split.
<i>aux</i>	Auxiliary split information, which may be modified on a successful split.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/ **best_binary_numeric_split.hpp**

39.475 BestBinaryNumericSplit< FitnessFunction >::AuxiliarySplitInfo< ElemType > Class Template Reference

39.475.1 Detailed Description

```
template<typename FitnessFunction>
template<typename ElemType>
class mlpack::tree::BestBinaryNumericSplit< FitnessFunction >::AuxiliarySplitInfo< ElemType >
```

Definition at line 32 of file best_binary_numeric_split.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/ **best_binary_numeric_split.hpp**

39.476 BinaryNumericSplit< FitnessFunction, ObservationType > Class Template Reference

The **BinaryNumericSplit** (p. 1992) class implements the numeric feature splitting strategy devised by Gama, Rocha, and Medas in the following paper:

Public Types

- typedef **BinaryNumericSplitInfo**< ObservationType > **SplitInfo**
*The splitting information required by the **BinaryNumericSplit** (p. 1992).*

Public Member Functions

- **BinaryNumericSplit** (const size_t numClasses=0)
*Create the **BinaryNumericSplit** (p. 1992) object with the given number of classes.*
- **BinaryNumericSplit** (const size_t numClasses, const **BinaryNumericSplit** &other)
*Create the **BinaryNumericSplit** (p. 1992) object with the given number of classes, using information from the given other split for other parameters.*
- void **EvaluateFitnessFunction** (double &bestFitness, double &secondBestFitness)
Given the points seen so far, evaluate the fitness function, returning the best possible gain of a binary split.
- size_t **MajorityClass** () const
The majority class of the points seen so far.
- double **MajorityProbability** () const
The probability of the majority class given the points seen so far.
- size_t **NumChildren** () const
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the object.
- void **Split** (arma::Col< size_t > &childMajorities, **SplitInfo** &splitInfo)
Given that a split should happen, return the majority classes of the (two) children and an initialized SplitInfo object.
- void **Train** (ObservationType value, const size_t label)
Train on the given value with the given label.

39.476.1 Detailed Description

```
template<typename FitnessFunction, typename ObservationType = double>
class mlpack::tree::BinaryNumericSplit< FitnessFunction, ObservationType >
```

The **BinaryNumericSplit** (p. 1992) class implements the numeric feature splitting strategy devised by Gama, Rocha, and Medas in the following paper:

```
@inproceedings{gama2003accurate,
  title={Accurate Decision Trees for Mining High-Speed Data Streams},
  author={Gama, J. and Rocha, R. and Medas, P.},
  year={2003},
  booktitle={Proceedings of the Ninth ACM SIGKDD International Conference on
    Knowledge Discovery and Data Mining (KDD '03)},
  pages={523--528}
}
```

This splitting procedure builds a binary tree on points it has seen so far, and then **EvaluateFitnessFunction()** (p. 1994) returns the best possible split in $O(n)$ time, where n is the number of samples seen so far. Every split with this split type returns only two splits (greater than or equal to the split point, and less than the split point). The **Train()** (p. 1996) function should take $O(1)$ time.

Template Parameters

<i>FitnessFunction</i>	Fitness function to use for calculating gain.
<i>ObservationType</i>	Type of observation used by this dimension.

Definition at line 47 of file `binary_numeric_split.hpp`.

39.476.2 Member Typedef Documentation

39.476.2.1 SplitInfo

```
typedef BinaryNumericSplitInfo<ObservationType> SplitInfo
```

The splitting information required by the **BinaryNumericSplit** (p. 1992).

Definition at line 51 of file `binary_numeric_split.hpp`.

39.476.3 Constructor & Destructor Documentation

39.476.3.1 BinaryNumericSplit() [1/2]

```
BinaryNumericSplit (
  const size_t numClasses = 0 )
```

Create the **BinaryNumericSplit** (p. 1992) object with the given number of classes.

Parameters

<i>numClasses</i>	Number of classes in dataset.
-------------------	-------------------------------

39.476.3.2 **BinaryNumericSplit()** [2/2]

```

BinaryNumericSplit (
    const size_t numClasses,
    const BinaryNumericSplit< FitnessFunction, ObservationType > & other )

```

Create the **BinaryNumericSplit** (p. 1992) object with the given number of classes, using information from the given other split for other parameters.

In this case, there are no other parameters, but this function is required by the **HoeffdingTree** (p. 2113) class.

39.476.4 Member Function Documentation

39.476.4.1 **EvaluateFitnessFunction()**

```

void EvaluateFitnessFunction (
    double & bestFitness,
    double & secondBestFitness )

```

Given the points seen so far, evaluate the fitness function, returning the best possible gain of a binary split.

Note that this takes $O(n)$ time, where n is the number of points seen so far. So this may not exactly be fast...

The best possible split will be stored in `bestFitness`, and the second best possible split will be stored in `secondBestFitness`.

Parameters

<i>bestFitness</i>	Fitness function value for best possible split.
<i>secondBestFitness</i>	Fitness function value for second best possible split.

39.476.4.2 **MajorityClass()**

```

size_t MajorityClass ( ) const

```

The majority class of the points seen so far.

Referenced by BinaryNumericSplit< FitnessFunction, ObservationType >::NumChildren().

39.476.4.3 MajorityProbability()

```
double MajorityProbability ( ) const
```

The probability of the majority class given the points seen so far.

Referenced by BinaryNumericSplit< FitnessFunction, ObservationType >::NumChildren().

39.476.4.4 NumChildren()

```
size_t NumChildren ( ) const [inline]
```

Definition at line 93 of file binary_numeric_split.hpp.

References BinaryNumericSplit< FitnessFunction, ObservationType >::MajorityClass(), BinaryNumericSplit< FitnessFunction, ObservationType >::MajorityProbability(), BinaryNumericSplit< FitnessFunction, ObservationType >::serialize(), and BinaryNumericSplit< FitnessFunction, ObservationType >::Split().

39.476.4.5 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the object.

Referenced by BinaryNumericSplit< FitnessFunction, ObservationType >::NumChildren().

39.476.4.6 Split()

```
void Split (
    arma::Col< size_t > & childMajorities,
    SplitInfo & splitInfo )
```

Given that a split should happen, return the majority classes of the (two) children and an initialized SplitInfo object.

Parameters

<i>childMajorities</i>	Majority classes of the children after the split.
<i>splitInfo</i>	Split information.

Referenced by BinaryNumericSplit< FitnessFunction, ObservationType >::NumChildren().

39.476.4.7 Train()

```
void Train (
    ObservationType value,
    const size_t label )
```

Train on the given value with the given label.

Parameters

<i>value</i>	The value to train on.
<i>label</i>	The label to train on.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ **binary_numeric_split.hpp**

39.477 BinaryNumericSplitInfo< ObservationType > Class Template Reference

Public Member Functions

- **BinaryNumericSplitInfo** ()
- **BinaryNumericSplitInfo** (const ObservationType &splitPoint)
- template<typename eT >
size_t **CalculateDirection** (const eT &value) const
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the split (save/load the split points).

39.477.1 Detailed Description

```
template<typename ObservationType = double>
class mlpack::tree::BinaryNumericSplitInfo< ObservationType >
```

Definition at line 22 of file binary_numeric_split_info.hpp.

39.477.2 Constructor & Destructor Documentation

39.477.2.1 BinaryNumericSplitInfo() [1/2]

```
BinaryNumericSplitInfo ( ) [inline]
```

Definition at line 25 of file `binary_numeric_split_info.hpp`.

39.477.2.2 BinaryNumericSplitInfo() [2/2]

```
BinaryNumericSplitInfo (
    const ObservationType & splitPoint ) [inline]
```

Definition at line 26 of file `binary_numeric_split_info.hpp`.

39.477.3 Member Function Documentation

39.477.3.1 CalculateDirection()

```
size_t CalculateDirection (
    const eT & value ) const [inline]
```

Definition at line 30 of file `binary_numeric_split_info.hpp`.

39.477.3.2 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the split (save/load the split points).

Definition at line 37 of file `binary_numeric_split_info.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ binary_numeric_split_info.hpp`

39.478 BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > Class Template Reference

A binary space partitioning tree, such as a KD-tree or a ball tree.

Classes

- class **BreadthFirstDualTreeTraverser**
- class **DualTreeTraverser**
A dual-tree traverser for binary space trees; see dual_tree_traverser.hpp.
- class **SingleTreeTraverser**
A single-tree traverser for binary space trees; see single_tree_traverser.hpp for implementation.

Public Types

- typedef MatType::elem_type **ElemType**
The type of element held in MatType.
- typedef MatType **Mat**
So other classes can use TreeType::Mat.
- typedef SplitType< BoundType< MetricType >, MatType > **Split**

Public Member Functions

- **BinarySpaceTree** (const MatType &data, const size_t maxLeafSize=20)
Construct this as the root node of a binary space tree using the given dataset.
- **BinarySpaceTree** (const MatType &data, std::vector< size_t > &oldFromNew, const size_t maxLeafSize=20)
Construct this as the root node of a binary space tree using the given dataset.
- **BinarySpaceTree** (const MatType &data, std::vector< size_t > &oldFromNew, std::vector< size_t > &newFromOld, const size_t maxLeafSize=20)
Construct this as the root node of a binary space tree using the given dataset.
- **BinarySpaceTree** (MatType &&data, const size_t maxLeafSize=20)
Construct this as the root node of a binary space tree using the given dataset.
- **BinarySpaceTree** (MatType &&data, std::vector< size_t > &oldFromNew, const size_t maxLeafSize=20)
Construct this as the root node of a binary space tree using the given dataset.
- **BinarySpaceTree** (MatType &&data, std::vector< size_t > &oldFromNew, std::vector< size_t > &newFromOld, const size_t maxLeafSize=20)
Construct this as the root node of a binary space tree using the given dataset.
- **BinarySpaceTree** (**BinarySpaceTree** *parent, const size_t begin, const size_t count, SplitType< BoundType< MetricType >, MatType > &splitter, const size_t maxLeafSize=20)
Construct this node as a child of the given parent, starting at column begin and using count points.
- **BinarySpaceTree** (**BinarySpaceTree** *parent, const size_t begin, const size_t count, std::vector< size_t > &oldFromNew, SplitType< BoundType< MetricType >, MatType > &splitter, const size_t maxLeafSize=20)
Construct this node as a child of the given parent, starting at column begin and using count points.

- **BinarySpaceTree** (**BinarySpaceTree** *parent, const size_t begin, const size_t count, std::vector< size_t > &oldFromNew, std::vector< size_t > &newFromOld, SplitType< BoundType< MetricType >, MatType > &splitter, const size_t maxLeafSize=20)
Construct this node as a child of the given parent, starting at column begin and using count points.
- **BinarySpaceTree** (const **BinarySpaceTree** &other)
Create a binary space tree by copying the other tree.
- **BinarySpaceTree** (**BinarySpaceTree** &&other)
*Move constructor for a **BinarySpaceTree** (p. 1998); possess all the members of the given tree.*
- template<typename Archive >
BinarySpaceTree (Archive &ar, const typename **std::enable_if_t**< Archive::is_loading::value > *=0)
*Initialize the tree from a **boost::serialization** (p. 251) archive.*
- ~**BinarySpaceTree** ()
Deletes this node, deallocating the memory for the children and calling their destructors in turn.
- size_t **Begin** () const
Return the index of the beginning point of this subset.
- size_t & **Begin** ()
Modify the index of the beginning point of this subset.
- const BoundType< MetricType > & **Bound** () const
Return the bound object for this node.
- BoundType< MetricType > & **Bound** ()
Return the bound object for this node.
- void **Center** (arma::vec ¢er) const
Store the center of the bounding region in the given vector.
- **BinarySpaceTree** & **Child** (const size_t child) const
Return the specified child (0 will be left, 1 will be right).
- **BinarySpaceTree** *& **ChildPtr** (const size_t child)
- size_t **Count** () const
Return the number of points in this subset.
- size_t & **Count** ()
Modify the number of points in this subset.
- const MatType & **Dataset** () const
Get the dataset which the tree is built on.
- MatType & **Dataset** ()
Modify the dataset which the tree is built on. Be careful!
- size_t **Descendant** (const size_t index) const
Return the index (with reference to the dataset) of a particular descendant of this node.
- ElemType **FurthestDescendantDistance** () const
Return the furthest possible descendant distance.
- ElemType **FurthestPointDistance** () const
Return the furthest distance to a point held in this node.
- template<typename VecType >
size_t **GetFurthestChild** (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > *=0)
Return the index of the furthest child node to the given query point.
- size_t **GetFurthestChild** (const **BinarySpaceTree** &queryNode)
Return the index of the furthest child node to the given query node.

- `template<typename VecType >`
`size_t GetNearestChild (const VecType &point, typename std::enable_if_t< IsVector< VecType >::value > ==0)`
Return the index of the nearest child node to the given query point.
- `size_t GetNearestChild (const BinarySpaceTree &queryNode)`
Return the index of the nearest child node to the given query node.
- `bool IsLeaf () const`
Return whether or not this node is a leaf (true if it has no children).
- `BinarySpaceTree * Left () const`
Gets the left child of this node.
- `BinarySpaceTree *& Left ()`
Modify the left child of this node.
- `ElemType MaxDistance (const BinarySpaceTree &other) const`
Return the maximum distance to another node.
- `template<typename VecType >`
`ElemType MaxDistance (const VecType &point, typename std::enable_if_t< IsVector< VecType >::value > ==0) const`
Return the maximum distance to another point.
- `MetricType Metric () const`
Get the metric that the tree uses.
- `ElemType MinDistance (const BinarySpaceTree &other) const`
Return the minimum distance to another node.
- `template<typename VecType >`
`ElemType MinDistance (const VecType &point, typename std::enable_if_t< IsVector< VecType >::value > ==0) const`
Return the minimum distance to another point.
- `ElemType MinimumBoundDistance () const`
Return the minimum distance from the center of the node to any bound edge.
- `size_t NumChildren () const`
Return the number of children in this node.
- `size_t NumDescendants () const`
Return the number of descendants of this node.
- `size_t NumPoints () const`
Return the number of points in this node (0 if not a leaf).
- `BinarySpaceTree * Parent () const`
Gets the parent of this node.
- `BinarySpaceTree *& Parent ()`
Modify the parent of this node.
- `ElemType ParentDistance () const`
Return the distance from the center of this node to the center of the parent node.
- `ElemType & ParentDistance ()`
Modify the distance from the center of this node to the center of the parent node.
- `size_t Point (const size_t index) const`
Return the index (with reference to the dataset) of a particular point in this node.
- `math::RangeType< ElemType > RangeDistance (const BinarySpaceTree &other) const`
Return the minimum and maximum distance to another node.

- `template<typename VecType >`
`math::RangeType< ElemType > RangeDistance` (const VecType &point, typename `std::enable_if_t< IsVector< VecType >::value > !=0`) const
Return the minimum and maximum distance to another point.
- `BinarySpaceTree * Right ()` const
Gets the right child of this node.
- `BinarySpaceTree *& Right ()`
Modify the right child of this node.
- `template<typename Archive >`
`void serialize` (Archive &ar, const unsigned int version)
Serialize the tree.
- `const StatisticType & Stat ()` const
Return the statistic object for this node.
- `StatisticType & Stat ()`
Return the statistic object for this node.

Protected Member Functions

- `BinarySpaceTree ()`
A default constructor.

39.478.1 Detailed Description

```
template<typename MetricType, typename StatisticType = EmptyStatistic, typename MatType = arma::mat, template< typename
BoundMetricType, typename... > class BoundType = bound::HRectBound, template< typename SplitBoundType, typename Split<
MatType > class SplitType = MidpointSplit>
class mlpack::tree::BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >
```

A binary space partitioning tree, such as a KD-tree or a ball tree.

Once the bound and type of dataset is defined, the tree will construct itself. Call the constructor with the dataset to build the tree on, and the entire tree will be built.

This particular tree does not allow growth, so you cannot add or delete nodes from it. If you need to add or delete a node, the better procedure is to rebuild the tree entirely.

This tree does take one runtime parameter in the constructor, which is the max leaf size to be used.

Template Parameters

<i>MetricType</i>	The metric used for tree-building. The BoundType may place restrictions on the metrics that can be used.
<i>StatisticType</i>	Extra data contained in the node. See <code>statistic.hpp</code> (p. 2688) for the necessary skeleton interface.
<i>MatType</i>	The dataset class.
<i>BoundType</i>	The bound used for each node. HRectBound, the default, requires that an LMetric<> is used for MetricType (so, EuclideanDistance, ManhattanDistance, etc.).
<i>SplitType</i>	The class that partitions the dataset/points at a particular node into two parts. Its definition decides the way this split is done.

Definition at line 54 of file `binary_space_tree.hpp`.

39.478.2 Member Typedef Documentation

39.478.2.1 ElemType

```
typedef MatType::elem_type ElemType
```

The type of element held in `MatType`.

Definition at line 60 of file `binary_space_tree.hpp`.

39.478.2.2 Mat

```
typedef MatType Mat
```

So other classes can use `TreeType::Mat`.

Definition at line 58 of file `binary_space_tree.hpp`.

39.478.2.3 Split

```
typedef SplitType<BoundType<MetricType>, MatType> Split
```

Definition at line 62 of file `binary_space_tree.hpp`.

39.478.3 Constructor & Destructor Documentation

39.478.3.1 BinarySpaceTree() [1/13]

```
BinarySpaceTree (  
    const MatType & data,  
    const size_t maxLeafSize = 20 )
```

Construct this as the root node of a binary space tree using the given dataset.

This will copy the input matrix; if you don't want this, consider using the constructor that takes an rvalue reference and use `std::move()`.

Parameters

<i>data</i>	Dataset to create tree from. This will be copied!
<i>maxLeafSize</i>	Size of each leaf in the tree.

39.478.3.2 BinarySpaceTree() [2/13]

```
BinarySpaceTree (
    const MatType & data,
    std::vector< size_t > & oldFromNew,
    const size_t maxLeafSize = 20 )
```

Construct this as the root node of a binary space tree using the given dataset.

This will copy the input matrix and modify its ordering; a mapping of the old point indices to the new point indices is filled. If you don't want the matrix to be copied, consider using the constructor that takes an rvalue reference and use `std::move()`.

Parameters

<i>data</i>	Dataset to create tree from. This will be copied!
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.
<i>maxLeafSize</i>	Size of each leaf in the tree.

39.478.3.3 BinarySpaceTree() [3/13]

```
BinarySpaceTree (
    const MatType & data,
    std::vector< size_t > & oldFromNew,
    std::vector< size_t > & newFromOld,
    const size_t maxLeafSize = 20 )
```

Construct this as the root node of a binary space tree using the given dataset.

This will copy the input matrix and modify its ordering; a mapping of the old point indices to the new point indices is filled, as well as a mapping of the new point indices to the old point indices. If you don't want the matrix to be copied, consider using the constructor that takes an rvalue reference and use `std::move()`.

Parameters

<i>data</i>	Dataset to create tree from. This will be copied!
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.
<i>newFromOld</i>	Vector which will be filled with the new positions for each old point.
<i>maxLeafSize</i>	Size of each leaf in the tree.

39.478.3.4 BinarySpaceTree() [4/13]

```

BinarySpaceTree (
    MatType && data,
    const size_t maxLeafSize = 20 )

```

Construct this as the root node of a binary space tree using the given dataset.

This will take ownership of the data matrix; if you don't want this, consider using the constructor that takes a const reference to a dataset.

Parameters

<i>data</i>	Dataset to create tree from.
<i>maxLeafSize</i>	Size of each leaf in the tree.

39.478.3.5 BinarySpaceTree() [5/13]

```

BinarySpaceTree (
    MatType && data,
    std::vector< size_t > & oldFromNew,
    const size_t maxLeafSize = 20 )

```

Construct this as the root node of a binary space tree using the given dataset.

This will take ownership of the data matrix; a mapping of the old point indices to the new point indices is filled. If you don't want the matrix to have its ownership taken, consider using the constructor that takes a const reference to a dataset.

Parameters

<i>data</i>	Dataset to create tree from.
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.
<i>maxLeafSize</i>	Size of each leaf in the tree.

39.478.3.6 BinarySpaceTree() [6/13]

```

BinarySpaceTree (
    MatType && data,
    std::vector< size_t > & oldFromNew,

```

```
std::vector< size_t > & newFromOld,  
const size_t maxLeafSize = 20 )
```

Construct this as the root node of a binary space tree using the given dataset.

This will take ownership of the data matrix; a mapping of the old point indices to the new point indices is filled, as well as a mapping of the new point indices to the old point indices. If you don't want the matrix to have its ownership taken, consider using the constructor that takes a const reference to a dataset.

Parameters

<i>data</i>	Dataset to create tree from.
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.
<i>newFromOld</i>	Vector which will be filled with the new positions for each old point.
<i>maxLeafSize</i>	Size of each leaf in the tree.

39.478.3.7 BinarySpaceTree() [7/13]

```
BinarySpaceTree (  
    BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > * parent,  
    const size_t begin,  
    const size_t count,  
    SplitType< BoundType< MetricType >, MatType > & splitter,  
    const size_t maxLeafSize = 20 )
```

Construct this node as a child of the given parent, starting at column begin and using count points.

The ordering of that subset of points in the parent's data matrix will be modified! This is used for recursive tree-building by the other constructors which don't specify point indices.

Parameters

<i>parent</i>	Parent of this node. Its dataset will be modified!
<i>begin</i>	Index of point to start tree construction with.
<i>count</i>	Number of points to use to construct tree.
<i>splitter</i>	Instantiated node splitter object.
<i>maxLeafSize</i>	Size of each leaf in the tree.

39.478.3.8 BinarySpaceTree() [8/13]

```
BinarySpaceTree (  
    BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > * parent,
```

```

const size_t begin,
const size_t count,
std::vector< size_t > & oldFromNew,
SplitType< BoundType< MetricType >, MatType > & splitter,
const size_t maxLeafSize = 20 )

```

Construct this node as a child of the given parent, starting at column begin and using count points.

The ordering of that subset of points in the parent's data matrix will be modified! This is used for recursive tree-building by the other constructors which don't specify point indices.

A mapping of the old point indices to the new point indices is filled, but it is expected that the vector is already allocated with size greater than or equal to (begin + count), and if that is not true, invalid memory reads (and writes) will occur.

Parameters

<i>parent</i>	Parent of this node. Its dataset will be modified!
<i>begin</i>	Index of point to start tree construction with.
<i>count</i>	Number of points to use to construct tree.
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.
<i>splitter</i>	Instantiated node splitter object.
<i>maxLeafSize</i>	Size of each leaf in the tree.

39.478.3.9 BinarySpaceTree() [9/13]

BinarySpaceTree (

```

    BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > * parent,
const size_t begin,
const size_t count,
std::vector< size_t > & oldFromNew,
std::vector< size_t > & newFromOld,
SplitType< BoundType< MetricType >, MatType > & splitter,
const size_t maxLeafSize = 20 )

```

Construct this node as a child of the given parent, starting at column begin and using count points.

The ordering of that subset of points in the parent's data matrix will be modified! This is used for recursive tree-building by the other constructors which don't specify point indices.

A mapping of the old point indices to the new point indices is filled, as well as a mapping of the new point indices to the old point indices. It is expected that the vector is already allocated with size greater than or equal to (begin_in + count_in), and if that is not true, invalid memory reads (and writes) will occur.

Parameters

<i>parent</i>	Parent of this node. Its dataset will be modified!
<i>begin</i>	Index of point to start tree construction with.
<i>count</i>	Number of points to use to construct tree.
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.
<i>newFromOld</i>	Vector which will be filled with the new positions for each old point.
<i>maxLeafSize</i>	Size of each leaf in the tree.

39.478.3.10 BinarySpaceTree() [10/13]

```
BinarySpaceTree (
    const BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > &
    other )
```

Create a binary space tree by copying the other tree.

Be careful! This can take a long time and use a lot of memory.

Parameters

<i>other</i>	Tree to be replicated.
--------------	------------------------

39.478.3.11 BinarySpaceTree() [11/13]

```
BinarySpaceTree (
    BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > &&
    other )
```

Move constructor for a **BinarySpaceTree** (p. 1998); possess all the members of the given tree.

39.478.3.12 BinarySpaceTree() [12/13]

```
BinarySpaceTree (
    Archive & ar,
    const typename std::enable_if_t< Archive::is_loading::value > * = 0 )
```

Initialize the tree from a **boost::serialization** (p. 251) archive.

Parameters

<i>ar</i>	Archive to load tree from. Must be an iarchive, not an oarchive.
-----------	--

39.478.3.13 ~BinarySpaceTree()

```
~ BinarySpaceTree ( )
```

Deletes this node, deallocating the memory for the children and calling their destructors in turn.

This will invalidate any pointers or references to any nodes which are children of this one.

39.478.3.14 `BinarySpaceTree()` [13/13]

```
BinarySpaceTree ( ) [protected]
```

A default constructor.

This is meant to only be used with **boost::serialization** (p. 251), which is allowed with the friend declaration below. This does not return a valid tree! The method must be protected, so that the serialization shim can work with the default constructor.

Referenced by `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::Center()`.

39.478.4 Member Function Documentation

39.478.4.1 `Begin()` [1/2]

```
size_t Begin ( ) const [inline]
```

Return the index of the beginning point of this subset.

Definition at line 483 of file `binary_space_tree.hpp`.

39.478.4.2 `Begin()` [2/2]

```
size_t& Begin ( ) [inline]
```

Modify the index of the beginning point of this subset.

Definition at line 485 of file `binary_space_tree.hpp`.

39.478.4.3 Bound() [1/2]

```
const BoundType<MetricType>& Bound ( ) const [inline]
```

Return the bound object for this node.

Definition at line 304 of file `binary_space_tree.hpp`.

Referenced by `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::MaxDistance()`, `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::MinDistance()`, and `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::RangeDistance()`.

39.478.4.4 Bound() [2/2]

```
BoundType<MetricType>& Bound ( ) [inline]
```

Return the bound object for this node.

Definition at line 306 of file `binary_space_tree.hpp`.

39.478.4.5 Center()

```
void Center (
    arma::vec & center ) const [inline]
```

Store the center of the bounding region in the given vector.

Definition at line 493 of file `binary_space_tree.hpp`.

References `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::BinarySpaceTree()`.

39.478.4.6 Child()

```
BinarySpaceTree& Child (
    const size_t child ) const
```

Return the specified child (0 will be left, 1 will be right).

If the index is greater than 1, this will return the right child.

Parameters

<i>child</i>	Index of child to return.
--------------	---------------------------

Referenced by `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::ParentDistance()`.

39.478.4.7 ChildPtr()

```
BinarySpaceTree*& ChildPtr (
    const size_t child ) [inline]
```

Definition at line 405 of file `binary_space_tree.hpp`.

References `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::Descendant()`, `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::NumDescendants()`, `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::NumPoints()`, and `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::Point()`.

39.478.4.8 Count() [1/2]

```
size_t Count ( ) const [inline]
```

Return the number of points in this subset.

Definition at line 488 of file `binary_space_tree.hpp`.

39.478.4.9 Count() [2/2]

```
size_t& Count ( ) [inline]
```

Modify the number of points in this subset.

Definition at line 490 of file `binary_space_tree.hpp`.

39.478.4.10 Dataset() [1/2]

```
const MatType& Dataset ( ) const [inline]
```

Get the dataset which the tree is built on.

Definition at line 332 of file binary_space_tree.hpp.

39.478.4.11 Dataset() [2/2]

```
MatType& Dataset ( ) [inline]
```

Modify the dataset which the tree is built on. Be careful!

Definition at line 334 of file binary_space_tree.hpp.

39.478.4.12 Descendant()

```
size_t Descendant (
    const size_t index ) const
```

Return the index (with reference to the dataset) of a particular descendant of this node.

The index should be greater than zero but less than the number of descendants.

Parameters

<i>index</i>	Index of the descendant.
--------------	--------------------------

Referenced by BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::ChildPtr().

39.478.4.13 FurthestDescendantDistance()

```
ElemType FurthestDescendantDistance ( ) const
```

Return the furthest possible descendant distance.

This returns the maximum distance from the centroid to the edge of the bound and not the empirical quantity which is the actual furthest descendant distance. So the actual furthest descendant distance may be less than what this method returns (but it will never be greater than this).

Referenced by BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::Metric().

39.478.4.14 FurthestPointDistance()

```
ElemType FurthestPointDistance ( ) const
```

Return the furthest distance to a point held in this node.

If this is not a leaf node, then the distance is 0 because the node holds no points.

Referenced by `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::Metric()`.

39.478.4.15 GetFurthestChild() [1/2]

```
size_t GetFurthestChild (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 )
```

Return the index of the furthest child node to the given query point.

If this is a leaf node, it will return **NumChildren()** (p. 2015) (invalid index).

Referenced by `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::Metric()`.

39.478.4.16 GetFurthestChild() [2/2]

```
size_t GetFurthestChild (
    const BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > &
    queryNode )
```

Return the index of the furthest child node to the given query node.

If it can't decide, it will return **NumChildren()** (p. 2015) (invalid index).

39.478.4.17 GetNearestChild() [1/2]

```
size_t GetNearestChild (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 )
```

Return the index of the nearest child node to the given query point.

If this is a leaf node, it will return **NumChildren()** (p. 2015) (invalid index).

Referenced by `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::Metric()`.

39.478.4.18 GetNearestChild() [2/2]

```
size_t GetNearestChild (
    const BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > &
    queryNode )
```

Return the index of the nearest child node to the given query node.

If it can't decide, it will return **NumChildren()** (p.2015) (invalid index).

39.478.4.19 IsLeaf()

```
bool IsLeaf ( ) const
```

Return whether or not this node is a leaf (true if it has no children).

Referenced by **BinarySpaceTree**< MetricType, StatisticType, MatType, BoundType, SplitType >::Stat().

39.478.4.20 Left() [1/2]

```
BinarySpaceTree* Left ( ) const [inline]
```

Gets the left child of this node.

Definition at line 317 of file `binary_space_tree.hpp`.

39.478.4.21 Left() [2/2]

```
BinarySpaceTree*& Left ( ) [inline]
```

Modify the left child of this node.

Definition at line 319 of file `binary_space_tree.hpp`.

39.478.4.22 MaxDistance() [1/2]

```
ElemType MaxDistance (
    const BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > &
    other ) const [inline]
```

Return the maximum distance to another node.

Definition at line 444 of file `binary_space_tree.hpp`.

References **BinarySpaceTree**< MetricType, StatisticType, MatType, BoundType, SplitType >::Bound().

39.478.4.23 MaxDistance() [2/2]

```
ElemType MaxDistance (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const [inline]
```

Return the maximum distance to another point.

Definition at line 466 of file `binary_space_tree.hpp`.

39.478.4.24 Metric()

```
MetricType Metric ( ) const [inline]
```

Get the metric that the tree uses.

Definition at line 337 of file `binary_space_tree.hpp`.

References `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::FurthestDescendantDistance()`, `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::FurthestPointDistance()`, `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::GetFurthestChild()`, `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::GetNearestChild()`, `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::MinimumBoundDistance()`, and `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::NumChildren()`.

39.478.4.25 MinDistance() [1/2]

```
ElemType MinDistance (
    const BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > &
    other ) const [inline]
```

Return the minimum distance to another node.

Definition at line 438 of file `binary_space_tree.hpp`.

References `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::Bound()`.

39.478.4.26 MinDistance() [2/2]

```
ElemType MinDistance (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const [inline]
```

Return the minimum distance to another point.

Definition at line 457 of file `binary_space_tree.hpp`.

39.478.4.27 MinimumBoundDistance()

```
ElemType MinimumBoundDistance ( ) const
```

Return the minimum distance from the center of the node to any bound edge.

Referenced by BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::Metric().

39.478.4.28 NumChildren()

```
size_t NumChildren ( ) const
```

Return the number of children in this node.

Referenced by BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::Metric().

39.478.4.29 NumDescendants()

```
size_t NumDescendants ( ) const
```

Return the number of descendants of this node.

For a non-leaf in a binary space tree, this is the number of points at the descendant leaves. For a leaf, this is the number of points in the leaf.

Referenced by BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::ChildPtr().

39.478.4.30 NumPoints()

```
size_t NumPoints ( ) const
```

Return the number of points in this node (0 if not a leaf).

Referenced by BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::ChildPtr().

39.478.4.31 Parent() [1/2]

```
BinarySpaceTree* Parent ( ) const [inline]
```

Gets the parent of this node.

Definition at line 327 of file `binary_space_tree.hpp`.

39.478.4.32 Parent() [2/2]

```
BinarySpaceTree*& Parent ( ) [inline]
```

Modify the parent of this node.

Definition at line 329 of file `binary_space_tree.hpp`.

39.478.4.33 ParentDistance() [1/2]

```
ElemType ParentDistance ( ) const [inline]
```

Return the distance from the center of this node to the center of the parent node.

Definition at line 392 of file `binary_space_tree.hpp`.

39.478.4.34 ParentDistance() [2/2]

```
ElemType& ParentDistance ( ) [inline]
```

Modify the distance from the center of this node to the center of the parent node.

Definition at line 395 of file `binary_space_tree.hpp`.

References `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::Child()`.

39.478.4.35 Point()

```
size_t Point (
    const size_t index ) const
```

Return the index (with reference to the dataset) of a particular point in this node.

This will happily return invalid indices if the given index is greater than the number of points in this node (obtained with **NumPoints()** (p. 2015)) – be careful.

Parameters

<i>index</i>	Index of point for which a dataset index is wanted.
--------------	---

Referenced by BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::ChildPtr().

39.478.4.36 RangeDistance() [1/2]

```
math::RangeType< ElemType> RangeDistance (
    const BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > &
    other ) const [inline]
```

Return the minimum and maximum distance to another node.

Definition at line 450 of file binary_space_tree.hpp.

References BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::Bound().

39.478.4.37 RangeDistance() [2/2]

```
math::RangeType< ElemType> RangeDistance (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const [inline]
```

Return the minimum and maximum distance to another point.

Definition at line 476 of file binary_space_tree.hpp.

39.478.4.38 Right() [1/2]

```
BinarySpaceTree* Right ( ) const [inline]
```

Gets the right child of this node.

Definition at line 322 of file binary_space_tree.hpp.

39.478.4.39 Right() [2/2]

```
BinarySpaceTree*& Right ( ) [inline]
```

Modify the right child of this node.

Definition at line 324 of file `binary_space_tree.hpp`.

39.478.4.40 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int version )
```

Serialize the tree.

39.478.4.41 Stat() [1/2]

```
const StatisticType& Stat ( ) const [inline]
```

Return the statistic object for this node.

Definition at line 309 of file `binary_space_tree.hpp`.

39.478.4.42 Stat() [2/2]

```
StatisticType& Stat ( ) [inline]
```

Return the statistic object for this node.

Definition at line 311 of file `binary_space_tree.hpp`.

References `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::IsLeaf()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ binary_space_tree.hpp`

39.479 BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::BreadthFirstDualTreeTraverser< RuleType > Class Template Reference

Public Types

- typedef **QueueFrame**< **BinarySpaceTree**, typename RuleType::TraversallInfoType > **QueueFrameType**

Public Member Functions

- **BreadthFirstDualTreeTraverser** (RuleType &rule)
Instantiate the dual-tree traverser with the given rule set.
- size_t **NumBaseCases** () const
Get the number of times a base case was calculated.
- size_t & **NumBaseCases** ()
Modify the number of times a base case was calculated.
- size_t **NumPrunes** () const
Get the number of prunes.
- size_t & **NumPrunes** ()
Modify the number of prunes.
- size_t **NumScores** () const
Get the number of times a node combination was scored.
- size_t & **NumScores** ()
Modify the number of times a node combination was scored.
- size_t **NumVisited** () const
Get the number of visited combinations.
- size_t & **NumVisited** ()
Modify the number of visited combinations.
- void **Traverse** (**BinarySpaceTree** &queryNode, **BinarySpaceTree** &referenceNode)
Traverse the two trees.
- void **Traverse** (**BinarySpaceTree** &queryNode, std::priority_queue< **QueueFrameType** > &referenceQueue)

39.479.1 Detailed Description

```
template<typename MetricType, typename StatisticType = EmptyStatistic, typename MatType = arma::mat, template< typename  
BoundMetricType, typename... > class BoundType = bound::HRectBound, template< typename SplitBoundType, typename SplitType  
MatType > class SplitType = MidpointSplit>  
template<typename RuleType>  
class mlpack::tree::BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::BreadthFirstDualTreeTraverser<  
RuleType >
```

Definition at line 103 of file binary_space_tree.hpp.

39.479.2 Member Typedef Documentation

39.479.2.1 QueueFrameType

```
typedef QueueFrame< BinarySpaceTree, typename RuleType::TraversalInfoType> QueueFrameType
```

Definition at line 53 of file breadth_first_dual_tree_traverser.hpp.

39.479.3 Constructor & Destructor Documentation

39.479.3.1 BreadthFirstDualTreeTraverser()

```
BreadthFirstDualTreeTraverser (
    RuleType & rule )
```

Instantiate the dual-tree traverser with the given rule set.

39.479.4 Member Function Documentation

39.479.4.1 NumBaseCases() [1/2]

```
size_t NumBaseCases ( ) const [inline]
```

Get the number of times a base case was calculated.

Definition at line 83 of file breadth_first_dual_tree_traverser.hpp.

39.479.4.2 NumBaseCases() [2/2]

```
size_t& NumBaseCases ( ) [inline]
```

Modify the number of times a base case was calculated.

Definition at line 85 of file breadth_first_dual_tree_traverser.hpp.

References QueueFrame< TreeType, TraversalInfoType >::traversalInfo.

39.479.4.3 NumPrunes() [1/2]

```
size_t NumPrunes ( ) const [inline]
```

Get the number of prunes.

Definition at line 68 of file breadth_first_dual_tree_traverser.hpp.

39.479.4.4 NumPrunes() [2/2]

```
size_t& NumPrunes ( ) [inline]
```

Modify the number of prunes.

Definition at line 70 of file breadth_first_dual_tree_traverser.hpp.

39.479.4.5 NumScores() [1/2]

```
size_t NumScores ( ) const [inline]
```

Get the number of times a node combination was scored.

Definition at line 78 of file breadth_first_dual_tree_traverser.hpp.

39.479.4.6 NumScores() [2/2]

```
size_t& NumScores ( ) [inline]
```

Modify the number of times a node combination was scored.

Definition at line 80 of file breadth_first_dual_tree_traverser.hpp.

39.479.4.7 NumVisited() [1/2]

```
size_t NumVisited ( ) const [inline]
```

Get the number of visited combinations.

Definition at line 73 of file breadth_first_dual_tree_traverser.hpp.

39.479.4.8 NumVisited() [2/2]

```
size_t& NumVisited ( ) [inline]
```

Modify the number of visited combinations.

Definition at line 75 of file `breadth_first_dual_tree_traverser.hpp`.

39.479.4.9 Traverse() [1/2]

```
void Traverse (
    BinarySpaceTree & queryNode,
    BinarySpaceTree & referenceNode )
```

Traverse the two trees.

This does not reset the number of prunes.

Parameters

<i>queryNode</i>	The query node to be traversed.
<i>referenceNode</i>	The reference node to be traversed.
<i>score</i>	The score of the current node combination.

39.479.4.10 Traverse() [2/2]

```
void Traverse (
    BinarySpaceTree & queryNode,
    std::priority_queue< QueueFrameType > & referenceQueue )
```

The documentation for this class was generated from the following files:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ binary_space_tree.hpp`
- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ breadth_first_dual_tree_traverser.hpp`

39.480 BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::DualTreeTraverser< RuleType > Class Template Reference

A dual-tree traverser for binary space trees; see `dual_tree_traverser.hpp`.

Public Member Functions

- **DualTreeTraverser** (RuleType &rule)
Instantiate the dual-tree traverser with the given rule set.
- size_t **NumBaseCases** () const
Get the number of times a base case was calculated.
- size_t & **NumBaseCases** ()
Modify the number of times a base case was calculated.
- size_t **NumPrunes** () const
Get the number of prunes.
- size_t & **NumPrunes** ()
Modify the number of prunes.
- size_t **NumScores** () const
Get the number of times a node combination was scored.
- size_t & **NumScores** ()
Modify the number of times a node combination was scored.
- size_t **NumVisited** () const
Get the number of visited combinations.
- size_t & **NumVisited** ()
Modify the number of visited combinations.
- void **Traverse** (BinarySpaceTree &queryNode, BinarySpaceTree &referenceNode)
Traverse the two trees.

39.480.1 Detailed Description

```
template<typename MetricType, typename StatisticType = EmptyStatistic, typename MatType = arma::mat, template< typename
BoundMetricType, typename... > class BoundType = bound::HRectBound, template< typename SplitBoundType, typename SplitType>
class SplitType = MidpointSplit>
template<typename RuleType>
class mpack::tree::BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::DualTreeTraverser< RuleType >
```

A dual-tree traverser for binary space trees; see dual_tree_traverser.hpp.

Definition at line 100 of file binary_space_tree.hpp.

39.480.2 Constructor & Destructor Documentation

39.480.2.1 DualTreeTraverser()

```
DualTreeTraverser (
    RuleType & rule )
```

Instantiate the dual-tree traverser with the given rule set.

39.480.3 Member Function Documentation

39.480.3.1 NumBaseCases() [1/2]

```
size_t NumBaseCases ( ) const [inline]
```

Get the number of times a base case was calculated.

Definition at line 67 of file dual_tree_traverser.hpp.

39.480.3.2 NumBaseCases() [2/2]

```
size_t& NumBaseCases ( ) [inline]
```

Modify the number of times a base case was calculated.

Definition at line 69 of file dual_tree_traverser.hpp.

39.480.3.3 NumPrunes() [1/2]

```
size_t NumPrunes ( ) const [inline]
```

Get the number of prunes.

Definition at line 52 of file dual_tree_traverser.hpp.

39.480.3.4 NumPrunes() [2/2]

```
size_t& NumPrunes ( ) [inline]
```

Modify the number of prunes.

Definition at line 54 of file dual_tree_traverser.hpp.

39.480.3.5 NumScores() [1/2]

```
size_t NumScores ( ) const [inline]
```

Get the number of times a node combination was scored.

Definition at line 62 of file dual_tree_traverser.hpp.

39.480.3.6 NumScores() [2/2]

```
size_t& NumScores ( ) [inline]
```

Modify the number of times a node combination was scored.

Definition at line 64 of file dual_tree_traverser.hpp.

39.480.3.7 NumVisited() [1/2]

```
size_t NumVisited ( ) const [inline]
```

Get the number of visited combinations.

Definition at line 57 of file dual_tree_traverser.hpp.

39.480.3.8 NumVisited() [2/2]

```
size_t& NumVisited ( ) [inline]
```

Modify the number of visited combinations.

Definition at line 59 of file dual_tree_traverser.hpp.

39.480.3.9 Traverse()

```
void Traverse (
    BinarySpaceTree & queryNode,
    BinarySpaceTree & referenceNode )
```

Traverse the two trees.

This does not reset the number of prunes.

Parameters

<i>queryNode</i>	The query node to be traversed.
<i>referenceNode</i>	The reference node to be traversed.
<i>score</i>	The score of the current node combination.

The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ **binary_space_tree.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ **dual_tree_traverser.hpp**

39.481 BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::SingleTreeTraverser< RuleType > Class Template Reference

A single-tree traverser for binary space trees; see single_tree_traverser.hpp for implementation.

Public Member Functions

- **SingleTreeTraverser** (RuleType &rule)
Instantiate the single tree traverser with the given rule set.
- size_t **NumPrunes** () const
Get the number of prunes.
- size_t & **NumPrunes** ()
Modify the number of prunes.
- void **Traverse** (const size_t queryIndex, **BinarySpaceTree** &referenceNode)
Traverse the tree with the given point.

39.481.1 Detailed Description

```
template<typename MetricType, typename StatisticType = EmptyStatistic, typename MatType = arma::mat, template< typename
BoundMetricType, typename... > class BoundType = bound::HRectBound, template< typename SplitBoundType, typename Split<
MatType > class SplitType = MidpointSplit>
template<typename RuleType>
class mlpack::tree::BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::SingleTreeTraverser< RuleType
>
```

A single-tree traverser for binary space trees; see single_tree_traverser.hpp for implementation.

Definition at line 96 of file binary_space_tree.hpp.

39.481.2 Constructor & Destructor Documentation

39.481.2.1 SingleTreeTraverser()

```
SingleTreeTraverser (
    RuleType & rule )
```

Instantiate the single tree traverser with the given rule set.

39.481.3 Member Function Documentation

39.481.3.1 NumPrunes() [1/2]

```
size_t NumPrunes ( ) const [inline]
```

Get the number of prunes.

Definition at line 50 of file single_tree_traverser.hpp.

39.481.3.2 NumPrunes() [2/2]

```
size_t& NumPrunes ( ) [inline]
```

Modify the number of prunes.

Definition at line 52 of file single_tree_traverser.hpp.

39.481.3.3 Traverse()

```
void Traverse (
    const size_t queryIndex,
    BinarySpaceTree & referenceNode )
```

Traverse the tree with the given point.

Parameters

<i>queryIndex</i>	The index of the point in the query set which is being used as the query point.
<i>referenceNode</i>	The tree node to be traversed.

The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ **binary_space_tree.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ **single_tree_traverser.hpp**

39.482 CategoricalSplitInfo Class Reference

Public Member Functions

- **CategoricalSplitInfo** (const size_t)
- template<typename Archive >
void **serialize** (Archive &, const unsigned int)
Serialize the object. (Nothing needs to be saved.)

Static Public Member Functions

- template<typename eT >
static size_t **CalculateDirection** (const eT &value)

39.482.1 Detailed Description

Definition at line 20 of file categorical_split_info.hpp.

39.482.2 Constructor & Destructor Documentation

39.482.2.1 CategoricalSplitInfo()

```
CategoricalSplitInfo (  
    const size_t ) [inline]
```

Definition at line 23 of file categorical_split_info.hpp.

39.482.3 Member Function Documentation

39.482.3.1 CalculateDirection()

```
static size_t CalculateDirection (
    const eT & value ) [inline], [static]
```

Definition at line 26 of file categorical_split_info.hpp.

39.482.3.2 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serialize the object. (Nothing needs to be saved.)

Definition at line 35 of file categorical_split_info.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ **categorical_split_info.hpp**

39.483 CompareCosineNode Class Reference

Public Member Functions

- bool **operator()** (const **CosineTree** *a, const **CosineTree** *b) const

39.483.1 Detailed Description

Definition at line 246 of file cosine_tree.hpp.

39.483.2 Member Function Documentation

39.483.2.1 operator>()

```
bool operator() (
    const CosineTree * a,
    const CosineTree * b ) const [inline]
```

Definition at line 250 of file cosine_tree.hpp.

References `CosineTree::L2Error()`.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cosine_tree/ **cosine_tree.hpp**

39.484 CosineTree Class Reference

Public Member Functions

- **CosineTree** (const arma::mat &dataset)
***CosineTree** (p. 2030) constructor for the root node of the tree.*
- **CosineTree** (**CosineTree** &parentNode, const std::vector< size_t > &subIndices)
***CosineTree** (p. 2030) constructor for nodes other than the root node of the tree.*
- **CosineTree** (const arma::mat &dataset, const double epsilon, const double delta)
*Construct the **CosineTree** (p. 2030) and the basis for the given matrix, and passed 'epsilon' and 'delta' parameters.*
- **~CosineTree** ()
*Clean up the **CosineTree** (p. 2030): release allocated memory (including children).*
- void **BasisVector** (arma::vec &bVector)
Set the basis vector of the node.
- arma::vec & **BasisVector** ()
Get the basis vector of the node.
- size_t **BinarySearch** (arma::vec &cDistribution, double value, size_t start, size_t end)
Sample a column based on the cumulative Length-Squared distribution of the cosine node, and a randomly generated value in the range [0, 1].
- void **CalculateCentroid** ()
Calculate centroid of the columns present in the node.
- void **CalculateCosines** (arma::vec &cosines)
Calculate cosines of the columns present in the node, with respect to the sampled splitting point.
- arma::vec & **Centroid** ()
Get pointer to the centroid vector.
- size_t **ColumnSampleLS** ()
Sample a point from the Length-Squared distribution of the cosine node.
- void **ColumnSamplesLS** (std::vector< size_t > &sampledIndices, arma::vec &probabilities, size_t numSamples)
Sample 'numSamples' points from the Length-Squared distribution of the cosine node.
- void **ConstructBasis** (**CosineNodeQueue** &treeQueue)
Constructs the final basis matrix, after the cosine tree construction.
- void **CosineNodeSplit** ()

This function splits the cosine node into two children based on the cosines of the columns contained in the node, with respect to the sampled splitting point.

- double **FrobNormSquared** () const
Get the Frobenius norm squared of columns in the node.
- const arma::mat & **GetDataset** () const
Get pointer to the dataset matrix.
- void **GetFinalBasis** (arma::mat &finalBasis)
Returns the basis of the constructed subspace.
- void **L2Error** (const double error)
Set the Monte Carlo error.
- double **L2Error** () const
Get the Monte Carlo error.
- **CosineTree** * **Left** () const
Get pointer to the left child of the node.
- **CosineTree** *& **Left** ()
Modify the pointer to the left child of the node.
- void **ModifiedGramSchmidt** (**CosineNodeQueue** &treeQueue, arma::vec ¢roid, arma::vec &newBasis←
Vector, arma::vec *addBasisVector=NULL)
Calculates the orthonormalization of the passed centroid, with respect to the current vector subspace.
- double **MonteCarloError** (**CosineTree** *node, **CosineNodeQueue** &treeQueue, arma::vec *addBasis←
Vector1=NULL, arma::vec *addBasisVector2=NULL)
Estimates the squared error of the projection of the input node's matrix onto the current vector subspace.
- size_t **NumColumns** () const
Get number of columns of input matrix in the node.
- **CosineTree** * **Parent** () const
Get pointer to the parent node.
- **CosineTree** *& **Parent** ()
Modify the pointer to the parent node.
- **CosineTree** * **Right** () const
Get pointer to the right child of the node.
- **CosineTree** *& **Right** ()
Modify the pointer to the left child of the node.
- size_t **SplitPointIndex** () const
Get the column index of split point of the node.
- std::vector< size_t > & **VectorIndices** ()
Get the indices of columns in the node.

39.484.1 Detailed Description

Definition at line 29 of file cosine_tree.hpp.

39.484.2 Constructor & Destructor Documentation

39.484.2.1 **CosineTree()** [1/3]

```
CosineTree (
    const arma::mat & dataset )
```

CosineTree (p. 2030) constructor for the root node of the tree.

It initializes the necessary variables required for splitting of the node, and building the tree further. It takes a pointer to the input matrix and calculates the relevant variables using it.

Parameters

<i>dataset</i>	Matrix for which cosine tree is constructed.
----------------	--

39.484.2.2 **CosineTree()** [2/3]

```
CosineTree (
    CosineTree & parentNode,
    const std::vector< size_t > & subIndices )
```

CosineTree (p. 2030) constructor for nodes other than the root node of the tree.

It takes in a pointer to the parent node and a list of column indices which mentions the columns to be included in the node. The function calculate the relevant variables just like the constructor above.

Parameters

<i>parentNode</i>	Pointer to the parent cosine node.
<i>subIndices</i>	Pointer to vector of column indices to be included.

39.484.2.3 **CosineTree()** [3/3]

```
CosineTree (
    const arma::mat & dataset,
    const double epsilon,
    const double delta )
```

Construct the **CosineTree** (p. 2030) and the basis for the given matrix, and passed 'epsilon' and 'delta' parameters.

The **CosineTree** (p. 2030) is constructed by splitting nodes in the direction of maximum error, stored using a priority queue. Basis vectors are added from the left and right children of the split node. The basis vector from a node is the orthonormalized centroid of its columns. The splitting continues till the Monte Carlo estimate of the input matrix's projection on the obtained subspace is less than a fraction of the norm of the input matrix.

Parameters

<i>dataset</i>	Matrix for which the CosineTree (p. 2030) is constructed.
<i>epsilon</i>	Error tolerance fraction for calculated subspace.
<i>delta</i>	Cumulative probability for Monte Carlo error lower bound.

39.484.2.4 \sim CosineTree()

\sim **CosineTree** ()

Clean up the **CosineTree** (p. 2030): release allocated memory (including children).

39.484.3 Member Function Documentation

39.484.3.1 BasisVector() [1/2]

```
void BasisVector (
    arma::vec & bVector ) [inline]
```

Set the basis vector of the node.

Definition at line 186 of file cosine_tree.hpp.

39.484.3.2 BasisVector() [2/2]

```
arma::vec& BasisVector ( ) [inline]
```

Get the basis vector of the node.

Definition at line 189 of file cosine_tree.hpp.

39.484.3.3 BinarySearch()

```
size_t BinarySearch (
    arma::vec & cDistribution,
    double value,
    size_t start,
    size_t end )
```

Sample a column based on the cumulative Length-Squared distribution of the cosine node, and a randomly generated value in the range [0, 1].

Binary search is more efficient than searching linearly for the same. This leads a significant speedup when there are large number of columns to choose from and when a number of samples are to be drawn from the distribution.

Parameters

<i>cDistribution</i>	Cumulative LS distribution of columns in the node.
<i>value</i>	Randomly generated value in the range [0, 1].
<i>start</i>	Starting index of the distribution interval to search in.
<i>end</i>	Ending index of the distribution interval to search in.

39.484.3.4 CalculateCentroid()

```
void CalculateCentroid ( )
```

Calculate centroid of the columns present in the node.

The calculated centroid is used as a basis vector for the cosine tree being constructed.

39.484.3.5 CalculateCosines()

```
void CalculateCosines (
    arma::vec & cosines )
```

Calculate cosines of the columns present in the node, with respect to the sampled splitting point.

The calculated cosine values are useful for splitting the node into its children.

Parameters

<i>cosines</i>	Vector to store the cosine values in.
----------------	---------------------------------------

39.484.3.6 Centroid()

```
arma::vec& Centroid ( ) [inline]
```

Get pointer to the centroid vector.

Definition at line 183 of file cosine_tree.hpp.

39.484.3.7 ColumnSampleLS()

```
size_t ColumnSampleLS ( )
```

Sample a point from the Length-Squared distribution of the cosine node.

The function uses 'l2NormsSquared' to calculate the cumulative probability distribution of the column vectors. The sampling is based on a randomly generated value in the range [0, 1].

39.484.3.8 ColumnSamplesLS()

```
void ColumnSamplesLS (
    std::vector< size_t > & sampledIndices,
    arma::vec & probabilities,
    size_t numSamples )
```

Sample 'numSamples' points from the Length-Squared distribution of the cosine node.

The function uses 'l2NormsSquared' to calculate the cumulative probability distribution of the column vectors. The sampling is based on a randomly generated values in the range [0, 1].

39.484.3.9 ConstructBasis()

```
void ConstructBasis (
    CosineNodeQueue & treeQueue )
```

Constructs the final basis matrix, after the cosine tree construction.

Parameters

<i>treeQueue</i>	Priority queue of cosine nodes.
------------------	---------------------------------

39.484.3.10 CosineNodeSplit()

```
void CosineNodeSplit ( )
```

This function splits the cosine node into two children based on the cosines of the columns contained in the node, with respect to the sampled splitting point.

The function also calls the **CosineTree** (p. 2030) constructor for the children.

39.484.3.11 FrobNormSquared()

```
double FrobNormSquared ( ) const [inline]
```

Get the Frobenius norm squared of columns in the node.

Definition at line 210 of file cosine_tree.hpp.

39.484.3.12 GetDataset()

```
const arma::mat& GetDataset ( ) const [inline]
```

Get pointer to the dataset matrix.

Definition at line 172 of file cosine_tree.hpp.

39.484.3.13 GetFinalBasis()

```
void GetFinalBasis (
    arma::mat & finalBasis ) [inline]
```

Returns the basis of the constructed subspace.

Definition at line 169 of file cosine_tree.hpp.

39.484.3.14 L2Error() [1/2]

```
void L2Error (
    const double error ) [inline]
```

Set the Monte Carlo error.

Definition at line 178 of file cosine_tree.hpp.

Referenced by CompareCosineNode::operator()().

39.484.3.15 L2Error() [2/2]

```
double L2Error ( ) const [inline]
```

Get the Monte Carlo error.

Definition at line 180 of file cosine_tree.hpp.

39.484.3.16 Left() [1/2]

```
CosineTree* Left ( ) const [inline]
```

Get pointer to the left child of the node.

Definition at line 197 of file cosine_tree.hpp.

39.484.3.17 Left() [2/2]

```
CosineTree*& Left ( ) [inline]
```

Modify the pointer to the left child of the node.

Definition at line 199 of file cosine_tree.hpp.

39.484.3.18 ModifiedGramSchmidt()

```
void ModifiedGramSchmidt (
    CosineNodeQueue & treeQueue,
    arma::vec & centroid,
    arma::vec & newBasisVector,
    arma::vec * addBasisVector = NULL )
```

Calculates the orthonormalization of the passed centroid, with respect to the current vector subspace.

Parameters

<i>treeQueue</i>	Priority queue of cosine nodes.
<i>centroid</i>	Centroid of the node being added to the basis.
<i>newBasisVector</i>	Orthonormalized centroid of the node.
<i>addBasisVector</i>	Address to additional basis vector.

39.484.3.19 MonteCarloError()

```
double MonteCarloError (
    CosineTree * node,
    CosineNodeQueue & treeQueue,
    arma::vec * addBasisVector1 = NULL,
    arma::vec * addBasisVector2 = NULL )
```

Estimates the squared error of the projection of the input node's matrix onto the current vector subspace.

A normal distribution is fit using weighted norms of projections of samples drawn from the input node's matrix columns. The error is calculated as the difference between the Frobenius norm of the input node's matrix and lower bound of the normal distribution.

Parameters

<i>node</i>	Node for which Monte Carlo estimate is calculated.
<i>treeQueue</i>	Priority queue of cosine nodes.
<i>addBasisVector1</i>	Address to first additional basis vector.
<i>addBasisVector2</i>	Address to second additional basis vector.

39.484.3.20 NumColumns()

```
size_t NumColumns ( ) const [inline]
```

Get number of columns of input matrix in the node.

Definition at line 207 of file cosine_tree.hpp.

39.484.3.21 Parent() [1/2]

```
CosineTree* Parent ( ) const [inline]
```

Get pointer to the parent node.

Definition at line 192 of file cosine_tree.hpp.

39.484.3.22 Parent() [2/2]

```
CosineTree*& Parent ( ) [inline]
```

Modify the pointer to the parent node.

Definition at line 194 of file cosine_tree.hpp.

39.484.3.23 Right() [1/2]

```
CosineTree* Right ( ) const [inline]
```

Get pointer to the right child of the node.

Definition at line 202 of file cosine_tree.hpp.

39.484.3.24 Right() [2/2]

```
CosineTree*& Right ( ) [inline]
```

Modify the pointer to the left child of the node.

Definition at line 204 of file cosine_tree.hpp.

39.484.3.25 SplitPointIndex()

```
size_t SplitPointIndex ( ) const [inline]
```

Get the column index of split point of the node.

Definition at line 213 of file cosine_tree.hpp.

39.484.3.26 VectorIndices()

```
std::vector<size_t>& VectorIndices ( ) [inline]
```

Get the indices of columns in the node.

Definition at line 175 of file cosine_tree.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cosine_tree/ **cosine_tree.hpp**

39.485 CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > Class Template Reference

A cover tree is a tree specifically designed to speed up nearest-neighbor computation in high-dimensional spaces.

Classes

- class **DualTreeTraverser**
A dual-tree cover tree traverser; see dual_tree_traverser.hpp.
- class **SingleTreeTraverser**
A single-tree cover tree traverser; see single_tree_traverser.hpp for implementation.

Public Types

- template<typename RuleType >
using **BreadthFirstDualTreeTraverser** = **DualTreeTraverser**< RuleType >
- typedef MatType::elem_type **ElemType**
The type held by the matrix type.
- typedef MatType **Mat**
So that other classes can access the matrix type.

Public Member Functions

- **CoverTree** (const MatType &dataset, const **ElemType** base=2.0, MetricType *metric=NULL)
Create the cover tree with the given dataset and given base.
- **CoverTree** (const MatType &dataset, MetricType &metric, const **ElemType** base=2.0)
Create the cover tree with the given dataset and the given instantiated metric.
- **CoverTree** (MatType &&dataset, const **ElemType** base=2.0)
Create the cover tree with the given dataset, taking ownership of the dataset.
- **CoverTree** (MatType &&dataset, MetricType &metric, const **ElemType** base=2.0)
Create the cover tree with the given dataset and the given instantiated metric, taking ownership of the dataset.
- **CoverTree** (const MatType &dataset, const **ElemType** base, const size_t pointIndex, const int scale, **CoverTree** *parent, const **ElemType** parentDistance, arma::Col< size_t > &indices, arma::vec &distances, size_t nearSetSize, size_t &farSetSize, size_t &usedSetSize, MetricType &metric=NULL)
Construct a child cover tree node.
- **CoverTree** (const MatType &dataset, const **ElemType** base, const size_t pointIndex, const int scale, **CoverTree** *parent, const **ElemType** parentDistance, const **ElemType** furthestDescendantDistance, MetricType *metric=NULL)
*Manually construct a cover tree node; no tree assembly is done in this constructor, and children must be added manually (use **Children()** (p. 2051)).*
- **CoverTree** (const **CoverTree** &other)
Create a cover tree from another tree.
- **CoverTree** (**CoverTree** &&other)
Move constructor for a Cover Tree, possess all the members of the given tree.

- template<typename Archive >
CoverTree (Archive &ar, const typename **std::enable_if_t**< Archive::is_loading::value > !=0)
*Create a cover tree from a **boost::serialization** (p. 251) archive.*
- ~**CoverTree** ()
Delete this cover tree node and its children.
- **ElemType Base** () const
Get the base.
- **ElemType & Base** ()
Modify the base; don't do this, you'll break everything.
- void **Center** (arma::vec ¢er) const
Get the center of the node and store it in the given vector.
- const **CoverTree & Child** (const size_t index) const
Get a particular child node.
- **CoverTree & Child** (const size_t index)
Modify a particular child node.
- **CoverTree *& ChildPtr** (const size_t index)
- const std::vector< **CoverTree *** > & **Children** () const
Get the children.
- std::vector< **CoverTree *** > & **Children** ()
Modify the children manually (maybe not a great idea).
- const MatType & **Dataset** () const
Get a reference to the dataset.
- size_t **Descendant** (const size_t index) const
Get the index of a particular descendant point.
- size_t **DistanceComps** () const
- size_t & **DistanceComps** ()
- **ElemType FurthestDescendantDistance** () const
Get the distance from the center of the node to the furthest descendant.
- **ElemType & FurthestDescendantDistance** ()
Modify the distance from the center of the node to the furthest descendant.
- **ElemType FurthestPointDistance** () const
Get the distance to the furthest point. This is always 0 for cover trees.
- template<typename VecType >
size_t **GetFurthestChild** (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > !=0)
Return the index of the furthest child node to the given query point.
- size_t **GetFurthestChild** (const **CoverTree** &queryNode)
Return the index of the furthest child node to the given query node.
- template<typename VecType >
size_t **GetNearestChild** (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > !=0)
Return the index of the nearest child node to the given query point.
- size_t **GetNearestChild** (const **CoverTree** &queryNode)
Return the index of the nearest child node to the given query node.
- bool **IsLeaf** () const
- **ElemType MaxDistance** (const **CoverTree** &other) const
Return the maximum distance to another node.
- **ElemType MaxDistance** (const **CoverTree** &other, const **ElemType** distance) const

Return the maximum distance to another node given that the point-to-point distance has already been calculated.

- **ElemType MaxDistance** (const arma::vec &other) const

Return the maximum distance to another point.

- **ElemType MaxDistance** (const arma::vec &other, const **ElemType** distance) const

Return the maximum distance to another point given that the distance from the center to the point has already been calculated.

- **MetricType & Metric** () const

Get the instantiated metric.

- **ElemType MinDistance** (const **CoverTree** &other) const

Return the minimum distance to another node.

- **ElemType MinDistance** (const **CoverTree** &other, const **ElemType** distance) const

Return the minimum distance to another node given that the point-to-point distance has already been calculated.

- **ElemType MinDistance** (const arma::vec &other) const

Return the minimum distance to another point.

- **ElemType MinDistance** (const arma::vec &other, const **ElemType** distance) const

Return the minimum distance to another point given that the distance from the center to the point has already been calculated.

- **ElemType MinimumBoundDistance** () const

Get the minimum distance from the center to any bound edge (this is the same as furthestDescendantDistance).

- **size_t NumChildren** () const

Get the number of children.

- **size_t NumDescendants** () const

Get the number of descendant points.

- **size_t NumPoints** () const

- **CoverTree * Parent** () const

Get the parent node.

- **CoverTree *& Parent** ()

Modify the parent node.

- **ElemType ParentDistance** () const

Get the distance to the parent.

- **ElemType & ParentDistance** ()

Modify the distance to the parent.

- **size_t Point** () const

Get the index of the point which this node represents.

- **size_t Point** (const size_t) const

For compatibility with other trees; the argument is ignored.

- **math::RangeType< ElemType > RangeDistance** (const **CoverTree** &other) const

Return the minimum and maximum distance to another node.

- **math::RangeType< ElemType > RangeDistance** (const **CoverTree** &other, const **ElemType** distance) const

Return the minimum and maximum distance to another node given that the point-to-point distance has already been calculated.

- **math::RangeType< ElemType > RangeDistance** (const arma::vec &other) const

Return the minimum and maximum distance to another point.

- **math::RangeType< ElemType > RangeDistance** (const arma::vec &other, const **ElemType** distance) const

Return the minimum and maximum distance to another point given that the point-to-point distance has already been calculated.

- **int Scale** () const

Get the scale of this node.

- int & **Scale** ()
Modify the scale of this node. Be careful...
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the tree.
- const StatisticType & **Stat** () const
Get the statistic for this node.
- StatisticType & **Stat** ()
Modify the statistic for this node.

Protected Member Functions

- **CoverTree** ()
A default constructor.

39.485.1 Detailed Description

```
template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
typename RootPointPolicy = FirstPointIsRoot>
class mlpack::tree::CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >
```

A cover tree is a tree specifically designed to speed up nearest-neighbor computation in high-dimensional spaces.

Each non-leaf node references a point and has a nonzero number of children, including a "self-child" which references the same point. A leaf node represents only one point.

The tree can be thought of as a hierarchy with the root node at the top level and the leaf nodes at the bottom level. Each level in the tree has an assigned 'scale' i . The tree follows these two invariants:

- nesting: the level C_i is a subset of the level C_{i-1} .
- covering: all node in level C_{i-1} have at least one node in the level C_i with distance less than or equal to b^i (exactly one of these is a parent of the point in level C_{i-1}).

Note that in the cover tree paper, there is a third invariant (the 'separation invariant'), but that does not apply to our implementation, because we have relaxed the invariant.

The value 'b' refers to the base, which is a parameter of the tree. These three properties make the cover tree very good for fast, high-dimensional nearest-neighbor search.

The theoretical structure of the tree contains many 'implicit' nodes which only have a "self-child" (a child referencing the same point, but at a lower scale level). This practical implementation only constructs explicit nodes – non-leaf nodes with more than one child. A leaf node has no children, and its scale level is INT_MIN.

For more information on cover trees, see

```
@inproceedings{
  author = {Beygelzimer, Alina and Kakade, Sham and Langford, John},
  title = {Cover trees for nearest neighbor},
  booktitle = {Proceedings of the 23rd International Conference on Machine
    Learning},
  series = {ICML '06},
  year = {2006},
  pages = {97--104}
}
```

For information on runtime bounds of the nearest-neighbor computation using cover trees, see the following paper, presented at NIPS 2009:

```
@inproceedings{
  author = {Ram, P., and Lee, D., and March, W.B., and Gray, A.G.},
  title = {Linear-time Algorithms for Pairwise Statistical Problems},
  booktitle = {Advances in Neural Information Processing Systems 22},
  editor = {Y. Bengio and D. Schuurmans and J. Lafferty and C.K.I. Williams
    and A. Culotta},
  pages = {1527--1535},
  year = {2009}
}
```

The **CoverTree** (p.2040) class offers three template parameters; a custom metric type can be used with `MetricType` (this class defaults to the L2-squared metric). The root node's point can be chosen with the `RootPointPolicy`; by default, the **FirstPointIsRoot** (p.2089) policy is used, meaning the first point in the dataset is used. The `StatisticType` policy allows you to define statistics which can be gathered during the creation of the tree.

Template Parameters

<i>MetricType</i>	Metric type to use during tree construction.
<i>RootPointPolicy</i>	Determines which point to use as the root node.
<i>StatisticType</i>	Statistic to be used during tree creation.
<i>MatType</i>	Type of matrix to build the tree on (generally <code>mat</code> or <code>sp_mat</code>).

Definition at line 99 of file `cover_tree.hpp`.

39.485.2 Member Typedef Documentation

39.485.2.1 BreadthFirstDualTreeTraverser

```
using BreadthFirstDualTreeTraverser = DualTreeTraverser<RuleType>
```

Definition at line 264 of file `cover_tree.hpp`.

39.485.2.2 ElemType

```
typedef MatType::elem_type ElemType
```

The type held by the matrix type.

Definition at line 105 of file cover_tree.hpp.

39.485.2.3 Mat

```
typedef MatType Mat
```

So that other classes can access the matrix type.

Definition at line 103 of file cover_tree.hpp.

39.485.3 Constructor & Destructor Documentation

39.485.3.1 CoverTree() [1/10]

```
CoverTree (  
    const MatType & dataset,  
    const ElemType base = 2.0,  
    MetricType * metric = NULL )
```

Create the cover tree with the given dataset and given base.

The dataset will not be modified during the building procedure (unlike **BinarySpaceTree** (p. 1998)).

The last argument will be removed in mpack 1.1.0 (see #274 and #273).

Parameters

<i>dataset</i>	Reference to the dataset to build a tree on.
<i>base</i>	Base to use during tree building (default 2.0).

39.485.3.2 CoverTree() [2/10]

```
CoverTree (  
    const MatType & dataset,
```

```

MetricType & metric,
const ElemType base = 2.0 )

```

Create the cover tree with the given dataset and the given instantiated metric.

Optionally, set the base. The dataset will not be modified during the building procedure (unlike **BinarySpaceTree** (p. 1998)).

Parameters

<i>dataset</i>	Reference to the dataset to build a tree on.
<i>metric</i>	Instantiated metric to use during tree building.
<i>base</i>	Base to use during tree building (default 2.0).

39.485.3.3 **CoverTree()** [3/10]

```

CoverTree (
    MatType && dataset,
    const ElemType base = 2.0 )

```

Create the cover tree with the given dataset, taking ownership of the dataset.

Optionally, set the base.

Parameters

<i>dataset</i>	Reference to the dataset to build a tree on.
<i>base</i>	Base to use during tree building (default 2.0).

39.485.3.4 **CoverTree()** [4/10]

```

CoverTree (
    MatType && dataset,
    MetricType & metric,
    const ElemType base = 2.0 )

```

Create the cover tree with the given dataset and the given instantiated metric, taking ownership of the dataset.

Optionally, set the base.

Parameters

<i>dataset</i>	Reference to the dataset to build a tree on.
<i>metric</i>	Instantiated metric to use during tree building.
<i>base</i>	Base to use during tree building (default 2.0).

39.485.3.5 CoverTree() [5/10]

```

CoverTree (
    const MatType & dataset,
    const ElemType base,
    const size_t pointIndex,
    const int scale,
    CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > * parent,
    const ElemType parentDistance,
    arma::Col< size_t > & indices,
    arma::vec & distances,
    size_t nearSetSize,
    size_t & farSetSize,
    size_t & usedSetSize,
    MetricType & metric = NULL )

```

Construct a child cover tree node.

This constructor is not meant to be used externally, but it could be used to insert another node into a tree. This procedure uses only one vector for the near set, the far set, and the used set (this is to prevent unnecessary memory allocation in recursive calls to this constructor). Therefore, the size of the near set, far set, and used set must be passed in. The near set will be entirely used up, and some of the far set may be used. The value of usedSetSize will be set to the number of points used in the construction of this node, and the value of farSetSize will be modified to reflect the number of points in the far set *after* the construction of this node.

If you are calling this manually, be careful that the given scale is as small as possible, or you may be creating an implicit node in your tree.

Parameters

<i>dataset</i>	Reference to the dataset to build a tree on.
<i>base</i>	Base to use during tree building.
<i>pointIndex</i>	Index of the point this node references.
<i>scale</i>	Scale of this level in the tree.
<i>parent</i>	Parent of this node (NULL indicates no parent).
<i>parentDistance</i>	Distance to the parent node.
<i>indices</i>	Array of indices, ordered [nearSet farSet usedSet]; will be modified to [farSet usedSet].
<i>distances</i>	Array of distances, ordered the same way as the indices. These represent the distances between the point specified by pointIndex and each point in the indices array.
<i>nearSetSize</i>	Size of the near set; if 0, this will be a leaf.
<i>farSetSize</i>	Size of the far set; may be modified (if this node uses any points in the far set).
<i>usedSetSize</i>	The number of points used will be added to this number.

39.485.3.6 **CoverTree()** [6/10]

```

CoverTree (
    const MatType & dataset,
    const ElemType base,
    const size_t pointIndex,
    const int scale,
    CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > * parent,
    const ElemType parentDistance,
    const ElemType furthestDescendantDistance,
    MetricType * metric = NULL )

```

Manually construct a cover tree node; no tree assembly is done in this constructor, and children must be added manually (use **Children()** (p. 2051)).

This constructor is useful when the tree is being "imported" into the **CoverTree** (p. 2040) class after being created in some other manner.

Parameters

<i>dataset</i>	Reference to the dataset this node is a part of.
<i>base</i>	Base that was used for tree building.
<i>pointIndex</i>	Index of the point in the dataset which this node refers to.
<i>scale</i>	Scale of this node's level in the tree.
<i>parent</i>	Parent node (NULL indicates no parent).
<i>parentDistance</i>	Distance to parent node point.
<i>furthestDescendantDistance</i>	Distance to furthest descendant point.
<i>metric</i>	Instantiated metric (optional).

39.485.3.7 **CoverTree()** [7/10]

```

CoverTree (
    const CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > & other )

```

Create a cover tree from another tree.

Be careful! This may use a lot of memory and take a lot of time. This will also make a copy of the dataset.

Parameters

<i>other</i>	Cover tree to copy from.
--------------	--------------------------

39.485.3.8 CoverTree() [8/10]

```

CoverTree (
    CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > && other )

```

Move constructor for a Cover Tree, possess all the members of the given tree.

Parameters

<i>other</i>	Cover Tree to move.
--------------	---------------------

39.485.3.9 CoverTree() [9/10]

```

CoverTree (
    Archive & ar,
    const typename std::enable_if_t< Archive::is_loading::value > * = 0 )

```

Create a cover tree from a **boost::serialization** (p. 251) archive.

39.485.3.10 ~CoverTree()

```

~ CoverTree ( )

```

Delete this cover tree node and its children.

39.485.3.11 CoverTree() [10/10]

```

CoverTree ( ) [protected]

```

A default constructor.

This is meant to only be used with **boost::serialization** (p. 251), which is allowed with the friend declaration below. This does not return a valid tree! This method must be protected, so that the serialization shim can work with the default constructor.

Referenced by CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::Metric().

39.485.4 Member Function Documentation

39.485.4.1 Base() [1/2]

```
ElemType Base ( ) const [inline]
```

Get the base.

Definition at line 304 of file cover_tree.hpp.

39.485.4.2 Base() [2/2]

```
ElemType& Base ( ) [inline]
```

Modify the base; don't do this, you'll break everything.

Definition at line 306 of file cover_tree.hpp.

39.485.4.3 Center()

```
void Center (
    arma::vec & center ) const [inline]
```

Get the center of the node and store it in the given vector.

Definition at line 412 of file cover_tree.hpp.

39.485.4.4 Child() [1/2]

```
const CoverTree& Child (
    const size_t index ) const [inline]
```

Get a particular child node.

Definition at line 278 of file cover_tree.hpp.

39.485.4.5 Child() [2/2]

```
CoverTree& Child (
    const size_t index ) [inline]
```

Modify a particular child node.

Definition at line 280 of file cover_tree.hpp.

39.485.4.6 ChildPtr()

```
CoverTree*& ChildPtr (
    const size_t index ) [inline]
```

Definition at line 282 of file cover_tree.hpp.

39.485.4.7 Children() [1/2]

```
const std::vector< CoverTree*>& Children ( ) const [inline]
```

Get the children.

Definition at line 288 of file cover_tree.hpp.

39.485.4.8 Children() [2/2]

```
std::vector< CoverTree*>& Children ( ) [inline]
```

Modify the children manually (maybe not a great idea).

Definition at line 290 of file cover_tree.hpp.

References CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::Descendant(), and CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::NumDescendants().

39.485.4.9 Dataset()

```
const MatType& Dataset ( ) const [inline]
```

Get a reference to the dataset.

Definition at line 267 of file cover_tree.hpp.

39.485.4.10 Descendant()

```
size_t Descendant (
    const size_t index ) const
```

Get the index of a particular descendant point.

Referenced by CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::Children().

39.485.4.11 DistanceComps() [1/2]

```
size_t DistanceComps ( ) const [inline]
```

Definition at line 555 of file cover_tree.hpp.

39.485.4.12 DistanceComps() [2/2]

```
size_t& DistanceComps ( ) [inline]
```

Definition at line 556 of file cover_tree.hpp.

39.485.4.13 FurthestDescendantDistance() [1/2]

```
ElemType FurthestDescendantDistance ( ) const [inline]
```

Get the distance from the center of the node to the furthest descendant.

Definition at line 401 of file cover_tree.hpp.

39.485.4.14 FurthestDescendantDistance() [2/2]

```
ElemType& FurthestDescendantDistance ( ) [inline]
```

Modify the distance from the center of the node to the furthest descendant.

Definition at line 405 of file cover_tree.hpp.

39.485.4.15 FurthestPointDistance()

```
ElemType FurthestPointDistance ( ) const [inline]
```

Get the distance to the furthest point. This is always 0 for cover trees.

Definition at line 398 of file cover_tree.hpp.

39.485.4.16 GetFurthestChild() [1/2]

```
size_t GetFurthestChild (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 )
```

Return the index of the furthest child node to the given query point.

If this is a leaf node, it will return **NumChildren()** (p. 2056) (invalid index).

Referenced by CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::Stat().

39.485.4.17 GetFurthestChild() [2/2]

```
size_t GetFurthestChild (
    const CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > & queryNode
)
```

Return the index of the furthest child node to the given query node.

If it can't decide, it will return **NumChildren()** (p. 2056) (invalid index).

39.485.4.18 GetNearestChild() [1/2]

```
size_t GetNearestChild (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 )
```

Return the index of the nearest child node to the given query point.

If this is a leaf node, it will return **NumChildren()** (p. 2056) (invalid index).

Referenced by `CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::Stat()`.

39.485.4.19 GetNearestChild() [2/2]

```
size_t GetNearestChild (
    const CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > & queryNode
)
```

Return the index of the nearest child node to the given query node.

If it can't decide, it will return **NumChildren()** (p. 2056) (invalid index).

39.485.4.20 IsLeaf()

```
bool IsLeaf ( ) const [inline]
```

Definition at line 274 of file `cover_tree.hpp`.

39.485.4.21 MaxDistance() [1/4]

```
ElemType MaxDistance (
    const CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > & other )
const
```

Return the maximum distance to another node.

Referenced by `CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::Stat()`.

39.485.4.22 MaxDistance() [2/4]

```
ElemType MaxDistance (
    const CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > & other,
    const ElemType distance ) const
```

Return the maximum distance to another node given that the point-to-point distance has already been calculated.

39.485.4.23 MaxDistance() [3/4]

```
ElemType MaxDistance (
    const arma::vec & other ) const
```

Return the maximum distance to another point.

39.485.4.24 MaxDistance() [4/4]

```
ElemType MaxDistance (
    const arma::vec & other,
    const ElemType distance ) const
```

Return the maximum distance to another point given that the distance from the center to the point has already been calculated.

39.485.4.25 Metric()

```
MetricType& Metric ( ) const [inline]
```

Get the instantiated metric.

Definition at line 418 of file cover_tree.hpp.

References CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::CoverTree().

39.485.4.26 MinDistance() [1/4]

```
ElemType MinDistance (
    const CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > & other )
const
```

Return the minimum distance to another node.

Referenced by `CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::Stat()`.

39.485.4.27 MinDistance() [2/4]

```
ElemType MinDistance (
    const CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > & other,
    const ElemType distance ) const
```

Return the minimum distance to another node given that the point-to-point distance has already been calculated.

39.485.4.28 MinDistance() [3/4]

```
ElemType MinDistance (
    const arma::vec & other ) const
```

Return the minimum distance to another point.

39.485.4.29 MinDistance() [4/4]

```
ElemType MinDistance (
    const arma::vec & other,
    const ElemType distance ) const
```

Return the minimum distance to another point given that the distance from the center to the point has already been calculated.

39.485.4.30 MinimumBoundDistance()

```
ElemType MinimumBoundDistance ( ) const [inline]
```

Get the minimum distance from the center to any bound edge (this is the same as `furthestDescendantDistance`).

Definition at line 409 of file `cover_tree.hpp`.

39.485.4.31 NumChildren()

```
size_t NumChildren ( ) const [inline]
```

Get the number of children.

Definition at line 285 of file cover_tree.hpp.

39.485.4.32 NumDescendants()

```
size_t NumDescendants ( ) const
```

Get the number of descendant points.

Referenced by CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::Children().

39.485.4.33 NumPoints()

```
size_t NumPoints ( ) const [inline]
```

Definition at line 275 of file cover_tree.hpp.

39.485.4.34 Parent() [1/2]

```
CoverTree* Parent ( ) const [inline]
```

Get the parent node.

Definition at line 388 of file cover_tree.hpp.

39.485.4.35 Parent() [2/2]

```
CoverTree*& Parent ( ) [inline]
```

Modify the parent node.

Definition at line 390 of file cover_tree.hpp.

39.485.4.36 ParentDistance() [1/2]

```
ElemType ParentDistance ( ) const [inline]
```

Get the distance to the parent.

Definition at line 393 of file cover_tree.hpp.

39.485.4.37 ParentDistance() [2/2]

```
ElemType& ParentDistance ( ) [inline]
```

Modify the distance to the parent.

Definition at line 395 of file cover_tree.hpp.

39.485.4.38 Point() [1/2]

```
size_t Point ( ) const [inline]
```

Get the index of the point which this node represents.

Definition at line 270 of file cover_tree.hpp.

39.485.4.39 Point() [2/2]

```
size_t Point (
    const size_t ) const [inline]
```

For compatibility with other trees; the argument is ignored.

Definition at line 272 of file cover_tree.hpp.

39.485.4.40 RangeDistance() [1/4]

```
math::RangeType< ElemType> RangeDistance (
    const CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > & other )
const
```

Return the minimum and maximum distance to another node.

Referenced by `CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::Stat()`.

39.485.4.41 RangeDistance() [2/4]

```
math::RangeType< ElemType> RangeDistance (
    const CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > & other,
    const ElemType distance ) const
```

Return the minimum and maximum distance to another node given that the point-to-point distance has already been calculated.

39.485.4.42 RangeDistance() [3/4]

```
math::RangeType< ElemType> RangeDistance (
    const arma::vec & other ) const
```

Return the minimum and maximum distance to another point.

39.485.4.43 RangeDistance() [4/4]

```
math::RangeType< ElemType> RangeDistance (
    const arma::vec & other,
    const ElemType distance ) const
```

Return the minimum and maximum distance to another point given that the point-to-point distance has already been calculated.

39.485.4.44 Scale() [1/2]

```
int Scale ( ) const [inline]
```

Get the scale of this node.

Definition at line 299 of file cover_tree.hpp.

39.485.4.45 Scale() [2/2]

```
int& Scale ( ) [inline]
```

Modify the scale of this node. Be careful...

Definition at line 301 of file cover_tree.hpp.

39.485.4.46 `serialize()`

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the tree.

39.485.4.47 `Stat()` [1/2]

```
const StatisticType& Stat ( ) const [inline]
```

Get the statistic for this node.

Definition at line 309 of file `cover_tree.hpp`.

39.485.4.48 `Stat()` [2/2]

```
StatisticType& Stat ( ) [inline]
```

Modify the statistic for this node.

Definition at line 311 of file `cover_tree.hpp`.

References `CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::GetFurthestChild()`, `CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::GetNearestChild()`, `CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::MaxDistance()`, `CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::MinDistance()`, and `CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::RangeDistance()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/ cover_tree.hpp`

39.486 `CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::DualTreeTraverser< RuleType >` **Class Template Reference**

A dual-tree cover tree traverser; see `dual_tree_traverser.hpp`.

Public Member Functions

- **DualTreeTraverser** (**RuleType** &rule)
Initialize the dual tree traverser with the given rule type.
- size_t **NumBaseCases** () const
- size_t **NumPrunes** () const
Get the number of pruned nodes.
- size_t & **NumPrunes** ()
Modify the number of pruned nodes.
- size_t **NumScores** () const
- size_t **NumVisited** () const
- void **Traverse** (**CoverTree** &queryNode, **CoverTree** &referenceNode)
Traverse the two specified trees.

39.486.1 Detailed Description

```
template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
typename RootPointPolicy = FirstPointIsRoot>
template<typename RuleType>
class mlpack::tree::CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::DualTreeTraverser< RuleType >
```

A dual-tree cover tree traverser; see dual_tree_traverser.hpp.

Definition at line 261 of file cover_tree.hpp.

39.486.2 Constructor & Destructor Documentation

39.486.2.1 DualTreeTraverser()

```
DualTreeTraverser (
    RuleType & rule )
```

Initialize the dual tree traverser with the given rule type.

39.486.3 Member Function Documentation

39.486.3.1 NumBaseCases()

```
size_t NumBaseCases ( ) const [inline]
```

Definition at line 54 of file dual_tree_traverser.hpp.

39.486.3.2 NumPrunes() [1/2]

```
size_t NumPrunes ( ) const [inline]
```

Get the number of pruned nodes.

Definition at line 46 of file dual_tree_traverser.hpp.

39.486.3.3 NumPrunes() [2/2]

```
size_t& NumPrunes ( ) [inline]
```

Modify the number of pruned nodes.

Definition at line 48 of file dual_tree_traverser.hpp.

39.486.3.4 NumScores()

```
size_t NumScores ( ) const [inline]
```

Definition at line 53 of file dual_tree_traverser.hpp.

39.486.3.5 NumVisited()

```
size_t NumVisited ( ) const [inline]
```

Definition at line 52 of file dual_tree_traverser.hpp.

39.486.3.6 Traverse()

```
void Traverse (
    CoverTree & queryNode,
    CoverTree & referenceNode )
```

Traverse the two specified trees.

Parameters

<i>queryNode</i>	Root of query tree.
<i>referenceNode</i>	Root of reference tree.

The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/ **cover_tree.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/ **dual_tree_traverser.hpp**

39.487 CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::SingleTreeTraverser< RuleType > Class Template Reference

A single-tree cover tree traverser; see single_tree_traverser.hpp for implementation.

Public Member Functions

- **SingleTreeTraverser** (RuleType &rule)
Initialize the single tree traverser with the given rule.
- size_t **NumPrunes** () const
Get the number of prunes so far.
- size_t & **NumPrunes** ()
Set the number of prunes (good for a reset to 0).
- void **Traverse** (const size_t queryIndex, **CoverTree** &referenceNode)
Traverse the tree with the given point.

39.487.1 Detailed Description

```
template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
typename RootPointPolicy = FirstPointsRoot>
template<typename RuleType>
class mlpack::tree::CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::SingleTreeTraverser< RuleType >
```

A single-tree cover tree traverser; see single_tree_traverser.hpp for implementation.

Definition at line 257 of file cover_tree.hpp.

39.487.2 Constructor & Destructor Documentation

39.487.2.1 SingleTreeTraverser()

```
SingleTreeTraverser (
    RuleType & rule )
```

Initialize the single tree traverser with the given rule.

39.487.3 Member Function Documentation

39.487.3.1 NumPrunes() [1/2]

```
size_t NumPrunes ( ) const [inline]
```

Get the number of prunes so far.

Definition at line 50 of file `single_tree_traverser.hpp`.

39.487.3.2 NumPrunes() [2/2]

```
size_t& NumPrunes ( ) [inline]
```

Set the number of prunes (good for a reset to 0).

Definition at line 52 of file `single_tree_traverser.hpp`.

39.487.3.3 Traverse()

```
void Traverse (
    const size_t queryIndex,
    CoverTree & referenceNode )
```

Traverse the tree with the given point.

Parameters

<i>queryIndex</i>	The index of the point in the query set which is used as the query point.
<i>referenceNode</i>	The tree node to be traversed.

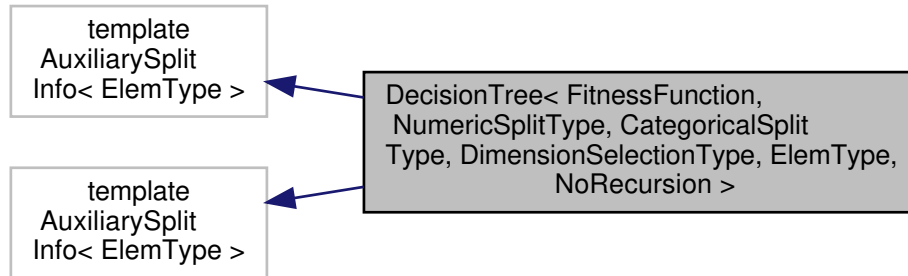
The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/ **cover_tree.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/ **single_tree_traverser.hpp**

39.488 **DecisionTree**< **FitnessFunction**, **NumericSplitType**, **CategoricalSplitType**, **DimensionSelectionType**, **ElemType**, **NoRecursion** > **Class Template Reference**

This class implements a generic decision tree learner.

Inheritance diagram for **DecisionTree**< **FitnessFunction**, **NumericSplitType**, **CategoricalSplitType**, **DimensionSelectionType**, **ElemType**, **NoRecursion** >:



Public Types

- typedef **CategoricalSplitType**< **FitnessFunction** > **CategoricalSplit**
Allow access to the categorical split type.
- typedef **DimensionSelectionType** **DimensionSelection**
Allow access to the dimension selection type.
- typedef **NumericSplitType**< **FitnessFunction** > **NumericSplit**
Allow access to the numeric split type.

Public Member Functions

- template<typename MatType , typename LabelsType >
DecisionTree (MatType data, const **data::DatasetInfo** &datasetInfo, LabelsType labels, const size_t numClasses, const size_t minimumLeafSize=10, const double minimumGainSplit=1e-7, DimensionSelectionType dimensionSelector=DimensionSelectionType())
Construct the decision tree on the given data and labels, where the data can be both numeric and categorical.
- template<typename MatType , typename LabelsType >
DecisionTree (MatType data, LabelsType labels, const size_t numClasses, const size_t minimumLeafSize=10, const double minimumGainSplit=1e-7, DimensionSelectionType dimensionSelector=DimensionSelectionType())
Construct the decision tree on the given data and labels, assuming that the data is all of the numeric type.

- `template<typename MatType , typename LabelsType , typename WeightsType >`
DecisionTree (MatType data, const **data::DatasetInfo** &datasetInfo, LabelsType labels, const size_t numClasses, WeightsType weights, const size_t minimumLeafSize=10, const double minimumGainSplit=1e-7, DimensionSelectionType dimensionSelector=DimensionSelectionType(), const std::enable_if_t< arma::is_arma_type< typename std::remove_reference< WeightsType >::type >::value > !=0)
Construct the decision tree on the given data and labels with weights, where the data can be both numeric and categorical.
- `template<typename MatType , typename LabelsType , typename WeightsType >`
DecisionTree (MatType data, LabelsType labels, const size_t numClasses, WeightsType weights, const size_t minimumLeafSize=10, const double minimumGainSplit=1e-7, DimensionSelectionType dimensionSelector=DimensionSelectionType(), const std::enable_if_t< arma::is_arma_type< typename std::remove_reference< WeightsType >::type >::value > !=0)
Construct the decision tree on the given data and labels with weights, assuming that the data is all of the numeric type.
- **DecisionTree** (const size_t numClasses=1)
Construct a decision tree without training it.
- **DecisionTree** (const **DecisionTree** &other)
Copy another tree.
- **DecisionTree** (**DecisionTree** &&other)
Take ownership of another tree.
- **~DecisionTree** ()
Clean up memory.
- `template<typename VecType >`
size_t **CalculateDirection** (const VecType &point) const
Given a point and that this node is not a leaf, calculate the index of the child node this point would go towards.
- const **DecisionTree** & **Child** (const size_t i) const
Get the child of the given index.
- **DecisionTree** & **Child** (const size_t i)
Modify the child of the given index (be careful!).
- `template<typename VecType >`
size_t **Classify** (const VecType &point) const
Classify the given point, using the entire tree.
- `template<typename VecType >`
void **Classify** (const VecType &point, size_t &prediction, arma::vec &probabilities) const
Classify the given point and also return estimates of the probability for each class in the given vector.
- `template<typename MatType >`
void **Classify** (const MatType &data, arma::Row< size_t > &predictions) const
Classify the given points, using the entire tree.
- `template<typename MatType >`
void **Classify** (const MatType &data, arma::Row< size_t > &predictions, arma::mat &probabilities) const
Classify the given points and also return estimates of the probabilities for each class in the given matrix.
- size_t **NumChildren** () const
Get the number of children.
- size_t **NumClasses** () const
Get the number of classes in the tree.
- **DecisionTree** & **operator=** (const **DecisionTree** &other)
Copy another tree.
- **DecisionTree** & **operator=** (**DecisionTree** &&other)
Take ownership of another tree.
- `template<typename Archive >`
void **serialize** (Archive &ar, const unsigned int)

Serialize the tree.

- `size_t SplitDimension () const`

Get the split dimension (only meaningful if this is a non-leaf in a trained tree).

- `template<typename MatType , typename LabelsType >
double Train (MatType data, const data::DatasetInfo &datasetInfo, LabelsType labels, const size_t numClasses,
const size_t minimumLeafSize=10, const double minimumGainSplit=1e-7, DimensionSelectionType dimensionSelector=DimensionSelectionType())`

Train the decision tree on the given data.

- `template<typename MatType , typename LabelsType >
double Train (MatType data, LabelsType labels, const size_t numClasses, const size_t minimumLeafSize=10,
const double minimumGainSplit=1e-7, DimensionSelectionType dimensionSelector=DimensionSelectionType())`

Train the decision tree on the given data, assuming that all dimensions are numeric.

- `template<typename MatType , typename LabelsType , typename WeightsType >
double Train (MatType data, const data::DatasetInfo &datasetInfo, LabelsType labels, const size_t numClasses,
WeightsType weights, const size_t minimumLeafSize=10, const double minimumGainSplit=1e-7, DimensionSelectionType dimensionSelector=DimensionSelectionType(), const std::enable_if_t< arma::is_arma_type< typename std::remove_reference< WeightsType >::type >::value > !=0)`

Train the decision tree on the given weighted data.

- `template<typename MatType , typename LabelsType , typename WeightsType >
double Train (MatType data, LabelsType labels, const size_t numClasses, WeightsType weights, const size_t minimumLeafSize=10, const double minimumGainSplit=1e-7, DimensionSelectionType dimensionSelector=DimensionSelectionType(), const std::enable_if_t< arma::is_arma_type< typename std::remove_reference< WeightsType >::type >::value > !=0)`

Train the decision tree on the given weighted data, assuming that all dimensions are numeric.

39.488.1 Detailed Description

```
template<typename FitnessFunction = GiniGain, template< typename > class NumericSplitType = BestBinaryNumericSplit,
template< typename > class CategoricalSplitType = AllCategoricalSplit, typename DimensionSelectionType = AllDimensionSelect,
typename ElemType = double, bool NoRecursion = false>
class mlpack::tree::DecisionTree< FitnessFunction, NumericSplitType, CategoricalSplitType, DimensionSelectionType, ElemType,
NoRecursion >
```

This class implements a generic decision tree learner.

Its behavior can be controlled via its template arguments.

The class inherits from the auxiliary split information in order to prevent an empty auxiliary split information struct from taking any extra size.

Definition at line 39 of file `decision_tree.hpp`.

39.488.2 Member Typedef Documentation

39.488.2.1 CategoricalSplit

```
typedef CategoricalSplitType<FitnessFunction> CategoricalSplit
```

Allow access to the categorical split type.

Definition at line 49 of file decision_tree.hpp.

39.488.2.2 DimensionSelection

```
typedef DimensionSelectionType DimensionSelection
```

Allow access to the dimension selection type.

Definition at line 51 of file decision_tree.hpp.

39.488.2.3 NumericSplit

```
typedef NumericSplitType<FitnessFunction> NumericSplit
```

Allow access to the numeric split type.

Definition at line 47 of file decision_tree.hpp.

39.488.3 Constructor & Destructor Documentation

39.488.3.1 DecisionTree() [1/7]

```
DecisionTree (
    MatType data,
    const data::DatasetInfo & datasetInfo,
    LabelsType labels,
    const size_t numClasses,
    const size_t minimumLeafSize = 10,
    const double minimumGainSplit = 1e-7,
    DimensionSelectionType dimensionSelector = DimensionSelectionType() )
```

Construct the decision tree on the given data and labels, where the data can be both numeric and categorical.

Setting minimumLeafSize and minimumGainSplit too small may cause the tree to overfit, but setting them too large may cause it to underfit.

Use std::move if data or labels are no longer needed to avoid copies.

Parameters

<i>data</i>	Dataset to train on.
<i>datasetInfo</i>	Type information for each dimension of the dataset.
<i>labels</i>	Labels for each training point.
<i>numClasses</i>	Number of classes in the dataset.
<i>minimumLeafSize</i>	Minimum number of points in each leaf node.
<i>minimumGainSplit</i>	Minimum gain for the node to split.
<i>dimensionSelector</i>	Instantiated dimension selection policy.

39.488.3.2 DecisionTree() [2/7]

```

DecisionTree (
    MatType data,
    LabelsType labels,
    const size_t numClasses,
    const size_t minimumLeafSize = 10,
    const double minimumGainSplit = 1e-7,
    DimensionSelectionType dimensionSelector = DimensionSelectionType() )

```

Construct the decision tree on the given data and labels, assuming that the data is all of the numeric type.

Setting minimumLeafSize and minimumGainSplit too small may cause the tree to overfit, but setting them too large may cause it to underfit.

Use std::move if data or labels are no longer needed to avoid copies.

Parameters

<i>data</i>	Dataset to train on.
<i>labels</i>	Labels for each training point.
<i>numClasses</i>	Number of classes in the dataset.
<i>minimumLeafSize</i>	Minimum number of points in each leaf node.
<i>minimumGainSplit</i>	Minimum gain for the node to split.
<i>dimensionSelector</i>	Instantiated dimension selection policy.

39.488.3.3 DecisionTree() [3/7]

```

DecisionTree (
    MatType data,
    const data::DatasetInfo & datasetInfo,

```

```

LabelsType labels,
const size_t numClasses,
WeightsType weights,
const size_t minimumLeafSize = 10,
const double minimumGainSplit = 1e-7,
DimensionSelectionType dimensionSelector = DimensionSelectionType(),
const std::enable_if_t< arma::is_arma_type< typename std::remove_reference< WeightsType >::type >::value > * = 0 >

```

Construct the decision tree on the given data and labels with weights, where the data can be both numeric and categorical.

Setting minimumLeafSize and minimumGainSplit too small may cause the tree to overfit, but setting them too large may cause it to underfit.

Use std::move if data, labels or weights are no longer needed to avoid copies.

Parameters

<i>data</i>	Dataset to train on.
<i>datasetInfo</i>	Type information for each dimension of the dataset.
<i>labels</i>	Labels for each training point.
<i>numClasses</i>	Number of classes in the dataset.
<i>weights</i>	The weight list of given label.
<i>minimumLeafSize</i>	Minimum number of points in each leaf node.
<i>minimumGainSplit</i>	Minimum gain for the node to split.
<i>dimensionSelector</i>	Instantiated dimension selection policy.

39.488.3.4 DecisionTree() [4/7]

```

DecisionTree (
    MatType data,
    LabelsType labels,
    const size_t numClasses,
    WeightsType weights,
    const size_t minimumLeafSize = 10,
    const double minimumGainSplit = 1e-7,
    DimensionSelectionType dimensionSelector = DimensionSelectionType(),
    const std::enable_if_t< arma::is_arma_type< typename std::remove_reference< WeightsType >::type >::value > * = 0 >

```

Construct the decision tree on the given data and labels with weights, assuming that the data is all of the numeric type.

Setting minimumLeafSize and minimumGainSplit too small may cause the tree to overfit, but setting them too large may cause it to underfit.

Use std::move if data, labels or weights are no longer needed to avoid copies.

Parameters

<i>data</i>	Dataset to train on.
<i>labels</i>	Labels for each training point.
<i>numClasses</i>	Number of classes in the dataset.
<i>weights</i>	The Weight list of given labels.
<i>minimumLeafSize</i>	Minimum number of points in each leaf node.
<i>minimumGainSplit</i>	Minimum gain for the node to split.
<i>dimensionSelector</i>	Instantiated dimension selection policy.

39.488.3.5 `DecisionTree()` [5/7]

```
DecisionTree (
    const size_t numClasses = 1 )
```

Construct a decision tree without training it.

It will be a leaf node with equal probabilities for each class.

Parameters

<i>numClasses</i>	Number of classes in the dataset.
-------------------	-----------------------------------

39.488.3.6 `DecisionTree()` [6/7]

```
DecisionTree (
    const DecisionTree< FitnessFunction, NumericSplitType, CategoricalSplitType, Dimension←
SelectionType, ElemType, NoRecursion > & other )
```

Copy another tree.

This may use a lot of memory—be sure that it's what you want to do.

Parameters

<i>other</i>	Tree to copy.
--------------	---------------

39.488.3.7 `DecisionTree()` [7/7]

```

DecisionTree (
    DecisionTree< FitnessFunction, NumericSplitType, CategoricalSplitType, Dimension↔
SelectionType, ElemType, NoRecursion > && other )

```

Take ownership of another tree.

Parameters

<i>other</i>	Tree to take ownership of.
--------------	----------------------------

39.488.3.8 `~DecisionTree()`

```

~ DecisionTree ( )

```

Clean up memory.

39.488.4 Member Function Documentation

39.488.4.1 `CalculateDirection()`

```

size_t CalculateDirection (
    const VecType & point ) const

```

Given a point and that this node is not a leaf, calculate the index of the child node this point would go towards.

This method is primarily used by the **Classify()** (p. 2073) function, but it can be used in a standalone sense too.

Parameters

<i>point</i>	Point to classify.
--------------	--------------------

Referenced by `DecisionTree< FitnessFunction, NumericSplitType, CategoricalSplitType, DimensionSelectionType, ElemType, NoRecursion >::SplitDimension()`.

39.488.4.2 Child() [1/2]

```
const DecisionTree& Child (
    const size_t i ) const [inline]
```

Get the child of the given index.

Definition at line 386 of file decision_tree.hpp.

39.488.4.3 Child() [2/2]

```
DecisionTree& Child (
    const size_t i ) [inline]
```

Modify the child of the given index (be careful!).

Definition at line 388 of file decision_tree.hpp.

39.488.4.4 Classify() [1/4]

```
size_t Classify (
    const VecType & point ) const
```

Classify the given point, using the entire tree.

The predicted label is returned.

Parameters

<i>point</i>	Point to classify.
--------------	--------------------

39.488.4.5 Classify() [2/4]

```
void Classify (
    const VecType & point,
    size_t & prediction,
    arma::vec & probabilities ) const
```

Classify the given point and also return estimates of the probability for each class in the given vector.

Parameters

<i>point</i>	Point to classify.
<i>prediction</i>	This will be set to the predicted class of the point.
<i>probabilities</i>	This will be filled with class probabilities for the point.

39.488.4.6 Classify() [3/4]

```
void Classify (
    const MatType & data,
    arma::Row< size_t > & predictions ) const
```

Classify the given points, using the entire tree.

The predicted labels for each point are stored in the given vector.

Parameters

<i>data</i>	Set of points to classify.
<i>predictions</i>	This will be filled with predictions for each point.

39.488.4.7 Classify() [4/4]

```
void Classify (
    const MatType & data,
    arma::Row< size_t > & predictions,
    arma::mat & probabilities ) const
```

Classify the given points and also return estimates of the probabilities for each class in the given matrix.

The predicted labels for each point are stored in the given vector.

Parameters

<i>data</i>	Set of points to classify.
<i>predictions</i>	This will be filled with predictions for each point.
<i>probabilities</i>	This will be filled with class probabilities for each point.

39.488.4.8 NumChildren()

```
size_t NumChildren ( ) const [inline]
```

Get the number of children.

Definition at line 383 of file decision_tree.hpp.

39.488.4.9 NumClasses()

```
size_t NumClasses ( ) const
```

Get the number of classes in the tree.

Referenced by DecisionTree< FitnessFunction, NumericSplitType, CategoricalSplitType, DimensionSelectionType, ElemType, NoRecursion >::SplitDimension().

39.488.4.10 operator=() [1/2]

```
DecisionTree& operator= (
    const DecisionTree< FitnessFunction, NumericSplitType, CategoricalSplitType, DimensionSelectionType, ElemType, NoRecursion > & other )
```

Copy another tree.

This may use a lot of memory—be sure that it's what you want to do.

Parameters

<i>other</i>	Tree to copy.
--------------	---------------

39.488.4.11 operator=() [2/2]

```
DecisionTree& operator= (
    DecisionTree< FitnessFunction, NumericSplitType, CategoricalSplitType, DimensionSelectionType, ElemType, NoRecursion > && other )
```

Take ownership of another tree.

Parameters

<i>other</i>	Tree to take ownership of.
--------------	----------------------------

39.488.4.12 `serialize()`

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the tree.

39.488.4.13 `SplitDimension()`

```
size_t SplitDimension ( ) const [inline]
```

Get the split dimension (only meaningful if this is a non-leaf in a trained tree).

Definition at line 392 of file `decision_tree.hpp`.

References `DecisionTree< FitnessFunction, NumericSplitType, CategoricalSplitType, DimensionSelectionType, ElemType, NoRecursion >::CalculateDirection()`, `DecisionTree< FitnessFunction, NumericSplitType, CategoricalSplitType, DimensionSelectionType, ElemType, NoRecursion >::NumClasses()`, and `DecisionTree< FitnessFunction, NumericSplitType, CategoricalSplitType, DimensionSelectionType, ElemType, NoRecursion >::Train()`.

39.488.4.14 `Train()` [1/4]

```
double Train (
    MatType data,
    const data::DatasetInfo & datasetInfo,
    LabelsType labels,
    const size_t numClasses,
    const size_t minimumLeafSize = 10,
    const double minimumGainSplit = 1e-7,
    DimensionSelectionType dimensionSelector = DimensionSelectionType() )
```

Train the decision tree on the given data.

This will overwrite the existing model. The data may have numeric and categorical types, specified by the `datasetInfo` parameter. Setting `minimumLeafSize` and `minimumGainSplit` too small may cause the tree to overfit, but setting them too large may cause it to underfit.

Use `std::move` if data or labels are no longer needed to avoid copies.

Parameters

<i>data</i>	Dataset to train on.
<i>datasetInfo</i>	Type information for each dimension.
<i>labels</i>	Labels for each training point.
<i>numClasses</i>	Number of classes in the dataset.
<i>weights</i>	Weights of all the labels
<i>minimumLeafSize</i>	Minimum number of points in each leaf node.
<i>minimumGainSplit</i>	Minimum gain for the node to split.
<i>dimensionSelector</i>	Instantiated dimension selection policy.

Returns

The final entropy of decision tree.

Referenced by DecisionTree< FitnessFunction, NumericSplitType, CategoricalSplitType, DimensionSelectionType, ElemType, NoRecursion >::SplitDimension().

39.488.4.15 Train() [2/4]

```
double Train (
    MatType data,
    LabelsType labels,
    const size_t numClasses,
    const size_t minimumLeafSize = 10,
    const double minimumGainSplit = 1e-7,
    DimensionSelectionType dimensionSelector = DimensionSelectionType() )
```

Train the decision tree on the given data, assuming that all dimensions are numeric.

This will overwrite the given model. Setting minimumLeafSize and minimumGainSplit too small may cause the tree to overfit, but setting them too large may cause it to underfit.

Use std::move if data or labels are no longer needed to avoid copies.

Parameters

<i>data</i>	Dataset to train on.
<i>labels</i>	Labels for each training point.
<i>numClasses</i>	Number of classes in the dataset.
<i>weights</i>	Weights of all the labels
<i>minimumLeafSize</i>	Minimum number of points in each leaf node.
<i>minimumGainSplit</i>	Minimum gain for the node to split.
<i>dimensionSelector</i>	Instantiated dimension selection policy.

Returns

The final entropy of decision tree.

39.488.4.16 Train() [3/4]

```
double Train (
    MatType data,
    const data::DatasetInfo & datasetInfo,
    LabelsType labels,
    const size_t numClasses,
    WeightsType weights,
    const size_t minimumLeafSize = 10,
    const double minimumGainSplit = 1e-7,
    DimensionSelectionType dimensionSelector = DimensionSelectionType(),
    const std::enable_if_t< arma::is_arma_type< typename std::remove_reference< WeightsType >::type >::value > * = 0 > )
```

Train the decision tree on the given weighted data.

This will overwrite the existing model. The data may have numeric and categorical types, specified by the datasetInfo parameter. Setting minimumLeafSize and minimumGainSplit too small may cause the tree to overfit, but setting them too large may cause it to underfit.

Use std::move if data, labels or weights are no longer needed to avoid copies.

Parameters

<i>data</i>	Dataset to train on.
<i>datasetInfo</i>	Type information for each dimension.
<i>labels</i>	Labels for each training point.
<i>numClasses</i>	Number of classes in the dataset.
<i>weights</i>	Weights of all the labels
<i>minimumLeafSize</i>	Minimum number of points in each leaf node.
<i>minimumGainSplit</i>	Minimum gain for the node to split.
<i>dimensionSelector</i>	Instantiated dimension selection policy.

Returns

The final entropy of decision tree.

39.488.4.17 Train() [4/4]

```
double Train (
    MatType data,
```

```

LabelsType labels,
const size_t numClasses,
WeightsType weights,
const size_t minimumLeafSize = 10,
const double minimumGainSplit = 1e-7,
DimensionSelectionType dimensionSelector = DimensionSelectionType(),
const std::enable_if_t< arma::is_arma_type< typename std::remove_reference< WeightsType >::type >::value > * = 0 >

```

Train the decision tree on the given weighted data, assuming that all dimensions are numeric.

This will overwrite the given model. Setting minimumLeafSize and minimumGainSplit too small may cause the tree to overfit, but setting them too large may cause it to underfit.

Use std::move if data, labels or weights are no longer needed to avoid copies.

Parameters

<i>data</i>	Dataset to train on.
<i>labels</i>	Labels for each training point.
<i>numClasses</i>	Number of classes in the dataset.
<i>weights</i>	Weights of all the labels
<i>minimumLeafSize</i>	Minimum number of points in each leaf node.
<i>minimumGainSplit</i>	Minimum gain for the node to split.
<i>dimensionSelector</i>	Instantiated dimension selection policy.

Returns

The final entropy of decision tree.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/ **decision_tree.hpp**

39.489 DiscreteHilbertValue< TreeElemType > Class Template Reference

The **DiscreteHilbertValue** (p. 2079) class stores Hilbert values for all of the points in a **RectangleTree** (p. 2207) node, and calculates Hilbert values for new points.

39.489.1 Detailed Description

```

template<typename TreeElemType>
class mlpack::tree::DiscreteHilbertValue< TreeElemType >

```

The **DiscreteHilbertValue** (p. 2079) class stores Hilbert values for all of the points in a **RectangleTree** (p. 2207) node, and calculates Hilbert values for new points.

This implementation calculates the full discrete Hilbert value; for a d-dimensional vector filled with elements of size E, each Hilbert value will take dE space.

Definition at line 29 of file `discrete_hilbert_value.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ discrete_hilbert_value.hpp`

39.490 EmptyStatistic Class Reference

Empty statistic if you are not interested in storing statistics in your tree.

Public Member Functions

- **EmptyStatistic** ()
- `template<typename TreeType >`
EmptyStatistic (TreeType &)
This constructor is called when a node is finished being created.
- **~EmptyStatistic** ()
- `template<typename Archive >`
`void` **serialize** (Archive &, const unsigned int)
Serialize the statistic (there's nothing to be saved).

39.490.1 Detailed Description

Empty statistic if you are not interested in storing statistics in your tree.

Use this as a template for your own.

Definition at line 24 of file `statistic.hpp`.

39.490.2 Constructor & Destructor Documentation

39.490.2.1 EmptyStatistic() [1/2]

```
EmptyStatistic ( ) [inline]
```

Definition at line 27 of file `statistic.hpp`.

39.490.2.2 ~EmptyStatistic()

```
~ EmptyStatistic ( ) [inline]
```

Definition at line 28 of file statistic.hpp.

39.490.2.3 EmptyStatistic() [2/2]

```
EmptyStatistic (
    TreeType & ) [inline]
```

This constructor is called when a node is finished being created.

The node is finished, and its children are finished, but it is not necessarily true that the statistics of other nodes are initialized yet.

Parameters

<i>node</i>	Node which this corresponds to.
-------------	---------------------------------

Definition at line 38 of file statistic.hpp.

39.490.3 Member Function Documentation

39.490.3.1 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serialize the statistic (there's nothing to be saved).

Definition at line 44 of file statistic.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ **statistic.hpp**

39.491 ExampleTree< MetricType, StatisticType, MatType > Class Template Reference

This is not an actual space tree but instead an example tree that exists to show and document all the functions that mlpack trees must implement.

Public Member Functions

- **ExampleTree** (const MatType &dataset, MetricType &metric)
This constructor will build the tree given a dataset and an instantiated metric.
- void **Centroid** (arma::vec ¢roid) const
Fill the given vector with the center of the node.
- const **ExampleTree** & **Child** (const size_t i) const
Return a particular child of this node.
- **ExampleTree** & **Child** (const size_t i)
Modify a particular child of this node.
- size_t **Descendant** (const size_t i) const
Get the index of a particular descendant point.
- double **FurthestDescendantDistance** () const
Get the distance from the center of the node to the furthest descendant point of this node.
- double **MaxDistance** (const MatType &point) const
Return the maximum distance between this node and a point.
- double **MaxDistance** (const **ExampleTree** &other) const
Return the maximum distance between this node and another node.
- const MetricType & **Metric** () const
Get the instantiated metric for this node.
- MetricType & **Metric** ()
Modify the instantiated metric for this node.
- double **MinDistance** (const MatType &point) const
Return the minimum distance between this node and a point.
- double **MinDistance** (const **ExampleTree** &other) const
Return the minimum distance between this node and another node.
- size_t **NumChildren** () const
Return the number of children of this node.
- size_t **NumDescendants** () const
Get the number of descendant points.
- size_t **NumPoints** () const
Return the number of points held in this node.
- **ExampleTree** * **Parent** () const
Return the parent node (NULL if this is the root of the tree).
- double **ParentDistance** () const
Get the distance from the center of this node to the center of the parent node.
- size_t **Point** (const size_t i) const
Return the index of a particular point of this node.
- **math::Range** **RangeDistance** (const MatType &point) const
*Return both the minimum and maximum distances between this node and a point as a **math::Range** (p. 409) object.*
- **math::Range** **RangeDistance** (const **ExampleTree** &other) const
*Return both the minimum and maximum distances between this node and another node as a **math::Range** (p. 409) object.*
- const StatisticType & **Stat** () const
Get the statistic for this node.
- StatisticType & **Stat** ()
Modify the statistic for this node.

39.491.1 Detailed Description

```
template<typename MetricType = metric::LMetric<2, true>, typename StatisticType = EmptyStatistic, typename MatType = arma<
::mat>
class mlpack::tree::ExampleTree< MetricType, StatisticType, MatType >
```

This is not an actual space tree but instead an example tree that exists to show and document all the functions that mlpack trees must implement.

For a better overview of trees, see **The TreeType policy in mlpack** (p. 179). Also be aware that the implementations of each of the methods in this example tree are entirely fake and do not work; this example tree exists for its API, not its implementation.

Note that trees often have different properties. These properties are known at compile-time through the **mlpack::tree::TreeTraits** (p. 2302) class, and some properties may imply the existence (or non-existence) of certain functions. Refer to the **TreeTraits** (p. 2302) for more documentation on that.

The three template parameters below must be template parameters to the tree, in the order given below. More template parameters are fine, but they must come after the first three.

Template Parameters

<i>MetricType</i>	This defines the space in which the tree will be built. For some trees, arbitrary metrics cannot be used, and a template metaprogramming approach should be used to issue a compile-time error if a metric cannot be used with a specific tree type. One example is the tree::BinarySpaceTree (p. 1998) tree type, which cannot work with the metric::IPMetric (p. 1565) class.
<i>StatisticType</i>	A tree node can hold a statistic, which is sometimes useful for various dual-tree algorithms. The tree itself does not need to know anything about how the statistic works, but it needs to hold a <i>StatisticType</i> in each node. It can be assumed that the <i>StatisticType</i> class has a constructor <code>StatisticType(const ExampleTree&)</code> .
<i>MatType</i>	A tree could be built on a dense matrix or a sparse matrix. All mlpack trees should be able to support any Armadillo-compatible matrix type. When the tree is written it should be assumed that <i>MatType</i> has the same functionality as <code>arma::mat</code> .

Definition at line 56 of file `example_tree.hpp`.

39.491.2 Constructor & Destructor Documentation

39.491.2.1 ExampleTree()

```
ExampleTree (
    const MatType & dataset,
    MetricType & metric )
```

This constructor will build the tree given a dataset and an instantiated metric.

Note that the parameter is a `MatType&` and not an `arma::mat&`. The dataset is not modified by the tree-building process (if it is, see the documentation for `mlpack::tree::TreeTraits::RearrangesDataset` (p. 2304) for how to deal with that situation). The `MetricType` parameter is necessary even though some metrics do not hold any state. This is so that the tree does not have to worry about instantiating the metric (if the tree had to worry about this, this would almost certainly incur additional runtime complexity and a larger runtime size of the tree node objects, which is to be avoided). The metric can't be `const`, in case `MetricType::Evaluate()` is non-`const`.

When this constructor is finished, the entire tree will be built and ready to use. The constructor should call the constructor of the statistic for each node that is built (see `tree::EmptyStatistic` (p. 2080) for more information).

Parameters

<i>dataset</i>	The dataset that the tree will be built on.
<i>metric</i>	The instantiated metric to use to build the dataset.

39.491.3 Member Function Documentation

39.491.3.1 Centroid()

```
void Centroid (
    arma::vec & centroid ) const
```

Fill the given vector with the center of the node.

Parameters

<i>centroid</i>	Vector to be filled with the center of the node.
-----------------	--

39.491.3.2 Child() [1/2]

```
const ExampleTree& Child (
    const size_t i ) const
```

Return a particular child of this node.

39.491.3.3 Child() [2/2]

```
ExampleTree& Child (
    const size_t i )
```

Modify a particular child of this node.

39.491.3.4 Descendant()

```
size_t Descendant (
    const size_t i ) const
```

Get the index of a particular descendant point.

The ordering of the descendants does not matter, as long as calling Descendant(0) through Descendant(**NumDescendants()** (p. 2087) - 1) will return the indices of every unique descendant point of the node.

39.491.3.5 FurthestDescendantDistance()

```
double FurthestDescendantDistance ( ) const
```

Get the distance from the center of the node to the furthest descendant point of this node.

This does not necessarily need to be the exact furthest descendant distance but instead can be an upper bound. See the definitions in **The TreeType policy in mlpack** (p. 179) for more information.

39.491.3.6 MaxDistance() [1/2]

```
double MaxDistance (
    const MatType & point ) const
```

Return the maximum distance between this node and a point.

It is not required that the exact maximum distance between the node and the point is returned but instead an upper bound on the maximum distance will suffice. See the definitions in **The TreeType policy in mlpack** (p. 179) for more information.

Parameters

<i>point</i>	Point to return [upper bound on] maximum distance to.
--------------	---

39.491.3.7 MaxDistance() [2/2]

```
double MaxDistance (
    const ExampleTree< MetricType, StatisticType, MatType > & other ) const
```

Return the maximum distance between this node and another node.

It is not required that the exact maximum distance between the two nodes be returned but instead an upper bound on the maximum distance will suffice. See the definitions in **The TreeType policy in mlpack** (p. 179) for more information.

Parameters

<i>node</i>	Node to return [upper bound on] maximum distance to.
-------------	--

39.491.3.8 Metric() [1/2]

```
const MetricType& Metric ( ) const
```

Get the instantiated metric for this node.

39.491.3.9 Metric() [2/2]

```
MetricType& Metric ( )
```

Modify the instantiated metric for this node.

39.491.3.10 MinDistance() [1/2]

```
double MinDistance (
    const MatType & point ) const
```

Return the minimum distance between this node and a point.

It is not required that the exact minimum distance between the node and the point is returned but instead a lower bound on the minimum distance will suffice. See the definitions in **The TreeType policy in mlpack** (p. 179) for more information.

Parameters

<i>point</i>	Point to return [lower bound on] minimum distance to.
--------------	---

39.491.3.11 MinDistance() [2/2]

```
double MinDistance (
    const ExampleTree< MetricType, StatisticType, MatType > & other ) const
```

Return the minimum distance between this node and another node.

It is not required that the exact minimum distance between the two nodes be returned but instead a lower bound on the minimum distance will suffice. See the definitions in **The TreeType policy in mlpack** (p. 179) for more information.

Parameters

<i>node</i>	Node to return [lower bound on] minimum distance to.
-------------	--

39.491.3.12 NumChildren()

```
size_t NumChildren ( ) const
```

Return the number of children of this node.

39.491.3.13 NumDescendants()

```
size_t NumDescendants ( ) const
```

Get the number of descendant points.

This is the number of unique points held in this node plus the number of points held in all descendant nodes. This could be calculated at build-time and cached, or could be calculated at run-time. This may be harder to calculate for trees that may hold points in multiple nodes (like cover trees and spill trees, for instance).

39.491.3.14 NumPoints()

```
size_t NumPoints ( ) const
```

Return the number of points held in this node.

39.491.3.15 Parent()

```
ExampleTree* Parent ( ) const
```

Return the parent node (NULL if this is the root of the tree).

39.491.3.16 ParentDistance()

```
double ParentDistance ( ) const
```

Get the distance from the center of this node to the center of the parent node.

39.491.3.17 Point()

```
size_t Point (
    const size_t i ) const
```

Return the index of a particular point of this node.

mlpack trees do not, in general, hold the actual dataset, and instead just hold the indices of the points they contain. Thus, you might use this function in code like this:

```
arma::vec thirdPoint = dataset.col(treeNode.Point(2));
```

39.491.3.18 RangeDistance() [1/2]

```
math::Range RangeDistance (
    const MatType & point ) const
```

Return both the minimum and maximum distances between this node and a point as a **math::Range** (p. 409) object.

This overload is given because it is possible that, for some tree types, calculation of both at once is faster than a call to **MinDistance()** (p. 2086) then **MaxDistance()** (p. 2085). It is not necessary that the minimum and maximum distances be exact; it is sufficient to return a lower bound on the minimum distance and an upper bound on the maximum distance. See the definitions in **The TreeType policy in mlpack** (p. 179) for more information.

Parameters

<i>point</i>	Point to return [bounds on] minimum and maximum distances to.
--------------	---

39.491.3.19 RangeDistance() [2/2]

```
math::Range RangeDistance (
    const ExampleTree< MetricType, StatisticType, MatType > & other ) const
```


Return both the minimum and maximum distances between this node and another node as a **math::Range** (p. 409) object.

This overload is given because it is possible that, for some tree types, calculation of both at once is faster than a call to **MinDistance()** (p. 2086) then **MaxDistance()** (p. 2085). It is not necessary that the minimum and maximum distances be exact; it is sufficient to return a lower bound on the minimum distance and an upper bound on the maximum distance. See the definitions in **The TreeType policy in mlpack** (p. 179) for more information.

Parameters

<i>node</i>	Node to return [bounds on] minimum and maximum distances to.
-------------	--

39.491.3.20 Stat() [1/2]

```
const StatisticType& Stat ( ) const
```

Get the statistic for this node.

39.491.3.21 Stat() [2/2]

```
StatisticType& Stat ( )
```

Modify the statistic for this node.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ **example_tree.hpp**

39.492 FirstPointIsRoot Class Reference

This class is meant to be used as a choice for the policy class RootPointPolicy of the **CoverTree** (p. 2040) class.

Static Public Member Functions

- `template<typename MatType >`
`static size_t ChooseRoot (const MatType &)`
Return the point to be used as the root point of the cover tree.

39.492.1 Detailed Description

This class is meant to be used as a choice for the policy class RootPointPolicy of the **CoverTree** (p. 2040) class.

This policy determines which point is used for the root node of the cover tree. This particular implementation simply chooses the first point in the dataset as the root. A more complex implementation might choose, for instance, the point with least maximum distance to other points (the closest to the "middle").

Definition at line 29 of file first_point_is_root.hpp.

39.492.2 Member Function Documentation

39.492.2.1 ChooseRoot()

```
static size_t ChooseRoot (
    const MatType & ) [inline], [static]
```

Return the point to be used as the root point of the cover tree.

This just returns 0.

Definition at line 37 of file first_point_is_root.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/ **first_point_is_root.hpp**

39.493 GiniGain Class Reference

The Gini gain, a measure of set purity usable as a fitness function (FitnessFunction) for decision trees.

Static Public Member Functions

- template<bool UseWeights, typename RowType, typename WeightVecType >
static double **Evaluate** (const RowType &labels, const size_t numClasses, const WeightVecType &weights)
Evaluate the Gini impurity on the given set of labels.
- template<bool UseWeights, typename CountType >
static double **EvaluatePtr** (const CountType *counts, const size_t countLength, const CountType totalCount)
Evaluate the Gini impurity given a vector of class weight counts.
- static double **Range** (const size_t numClasses)
Return the range of the Gini impurity for the given number of classes.

39.493.1 Detailed Description

The Gini gain, a measure of set purity usable as a fitness function (FitnessFunction) for decision trees.

This is the exact same thing as the well-known Gini impurity, but negated—since the decision tree will be trying to maximize gain (and the Gini impurity would need to be minimized).

Definition at line 27 of file gini_gain.hpp.

39.493.2 Member Function Documentation

39.493.2.1 Evaluate()

```
static double Evaluate (  
    const RowType & labels,  
    const size_t numClasses,  
    const WeightVecType & weights ) [inline], [static]
```

Evaluate the Gini impurity on the given set of labels.

RowType should be an Armadillo vector that holds size_t objects.

Note that it is possible that due to floating-point representation issues, it is possible that the gain returned can be very slightly greater than 0! Thus, if you are checking for a perfect fit, be sure to use 'gain >= 0.0' not 'gain == 0.0'.

Parameters

<i>labels</i>	Set of labels to evaluate Gini impurity on.
<i>numClasses</i>	Number of classes in the dataset.
<i>weights</i>	Weight of labels.

Definition at line 62 of file gini_gain.hpp.

39.493.2.2 EvaluatePtr()

```
static double EvaluatePtr (  
    const CountType * counts,  
    const size_t countLength,  
    const CountType totalCount ) [inline], [static]
```

Evaluate the Gini impurity given a vector of class weight counts.

Definition at line 34 of file gini_gain.hpp.

39.493.2.3 Range()

```
static double Range (
    const size_t numClasses ) [inline], [static]
```

Return the range of the Gini impurity for the given number of classes.

(That is, the difference between the maximum possible value and the minimum possible value.)

Parameters

<i>numClasses</i>	Number of classes in the dataset.
-------------------	-----------------------------------

Definition at line 203 of file gini_gain.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/ **gini_gain.hpp**

39.494 GinImpurity Class Reference

Static Public Member Functions

- static double **Evaluate** (const arma::Mat< size_t > &counts)
- static double **Range** (const size_t numClasses)
Return the range of the Gini impurity for the given number of classes.

39.494.1 Detailed Description

Definition at line 21 of file gini_impurity.hpp.

39.494.2 Member Function Documentation

39.494.2.1 Evaluate()

```
static double Evaluate (
    const arma::Mat< size_t > & counts ) [inline], [static]
```

Definition at line 24 of file gini_impurity.hpp.

39.494.2.2 Range()

```
static double Range (
    const size_t numClasses ) [inline], [static]
```

Return the range of the Gini impurity for the given number of classes.

(That is, the difference between the maximum possible value and the minimum possible value.)

Definition at line 77 of file gini_impurity.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ **gini_impurity.hpp**

39.495 GreedySingleTreeTraverser< TreeType, RuleType > Class Template Reference

Public Member Functions

- **GreedySingleTreeTraverser** (RuleType &rule)
Instantiate the greedy single tree traverser with the given rule set.
- size_t & **MinBaseCases** ()
Set value of minBaseCases.
- size_t **MinBaseCases** () const
Get value of minBaseCases.
- size_t **NumPrunes** () const
Get the number of prunes.
- void **Traverse** (const size_t queryIndex, TreeType &referenceNode)
Traverse the tree with the given point.

39.495.1 Detailed Description

```
template<typename TreeType, typename RuleType>
class mlpack::tree::GreedySingleTreeTraverser< TreeType, RuleType >
```

Definition at line 23 of file greedy_single_tree_traverser.hpp.

39.495.2 Constructor & Destructor Documentation

39.495.2.1 GreedySingleTreeTraverser()

```
GreedySingleTreeTraverser (  
    RuleType & rule )
```

Instantiate the greedy single tree traverser with the given rule set.

39.495.3 Member Function Documentation

39.495.3.1 MinBaseCases() [1/2]

```
size_t& MinBaseCases ( ) [inline]
```

Set value of minBaseCases.

Definition at line 44 of file greedy_single_tree_traverser.hpp.

39.495.3.2 MinBaseCases() [2/2]

```
size_t MinBaseCases ( ) const [inline]
```

Get value of minBaseCases.

Definition at line 47 of file greedy_single_tree_traverser.hpp.

39.495.3.3 NumPrunes()

```
size_t NumPrunes ( ) const [inline]
```

Get the number of prunes.

Definition at line 41 of file greedy_single_tree_traverser.hpp.

39.495.3.4 Traverse()

```
void Traverse (  
    const size_t queryIndex,  
    TreeType & referenceNode )
```

Traverse the tree with the given point.

Parameters

<i>queryIndex</i>	The index of the point in the query set which is being used as the query point.
<i>referenceNode</i>	The tree node to be traversed.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ **greedy_single_tree_traverser.hpp**

39.496 HilbertRTreeAuxiliaryInformation< TreeType, HilbertValueType > Class Template Reference

Public Types

- typedef TreeType::ElemType **ElemType**
The element type held by the tree.

Public Member Functions

- **HilbertRTreeAuxiliaryInformation** ()
Default constructor.
- **HilbertRTreeAuxiliaryInformation** (const TreeType *node)
Construct this as an auxiliary information for the given node.
- **HilbertRTreeAuxiliaryInformation** (const **HilbertRTreeAuxiliaryInformation** &other, TreeType *tree=NULL, bool deepCopy=true)
Create an auxiliary information object by copying from another object.
- **HilbertRTreeAuxiliaryInformation** (**HilbertRTreeAuxiliaryInformation** &&other)
Create an auxiliary information object by moving from the other node.
- bool **HandleNodeInsertion** (TreeType *node, TreeType *nodeToInsert, bool insertionLevel)
The Hilbert R tree requires to insert nodes according to their Hilbert value.
- bool **HandleNodeRemoval** (TreeType *node, const size_t nodeIndex)
The Hilbert R tree requires all nodes to be arranged according to their Hilbert value.
- bool **HandlePointDeletion** (TreeType *node, const size_t localIndex)
The Hilbert R tree requires all points to be arranged according to their Hilbert value.
- bool **HandlePointInsertion** (TreeType *node, const size_t point)
The Hilbert R tree requires to insert points according to their Hilbert value.
- const HilbertValueType< **ElemType** > & **HilbertValue** () const
Return the largest Hilbert value of a point covered by the node.
- HilbertValueType< **ElemType** > & **HilbertValue** ()
Modify the largest Hilbert value of a point covered by the node.
- void **NullifyData** ()
Clear memory.
- **HilbertRTreeAuxiliaryInformation** & **operator=** (const **HilbertRTreeAuxiliaryInformation** &other)
Copy the auxiliary information.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the information.
- bool **UpdateAuxiliaryInfo** (TreeType *node)
Update the auxiliary information in the node.

Static Public Member Functions

- static const std::vector< TreeType * > **Children** (const TreeType *tree)
Return the children vector of the tree.

39.496.1 Detailed Description

```
template<typename TreeType, template< typename > class HilbertValueType>
class mpack::tree::HilbertRTreeAuxiliaryInformation< TreeType, HilbertValueType >
```

Definition at line 22 of file hilbert_r_tree_auxiliary_information.hpp.

39.496.2 Member Typedef Documentation

39.496.2.1 ElemType

```
typedef TreeType::ElemType ElemType
```

The element type held by the tree.

Definition at line 26 of file hilbert_r_tree_auxiliary_information.hpp.

39.496.3 Constructor & Destructor Documentation

39.496.3.1 HilbertRTreeAuxiliaryInformation() [1/4]

```
HilbertRTreeAuxiliaryInformation ( )
```

Default constructor.

39.496.3.2 HilbertRTreeAuxiliaryInformation() [2/4]

```
HilbertRTreeAuxiliaryInformation (
    const TreeType * node )
```

Construct this as an auxiliary information for the given node.

Parameters

<i>node</i>	The node that stores this auxiliary information.
-------------	--

39.496.3.3 HilbertRTreeAuxiliaryInformation() [3 / 4]

```
HilbertRTreeAuxiliaryInformation (
    const HilbertRTreeAuxiliaryInformation< TreeType, HilbertValueType > & other,
    TreeType * tree = NULL,
    bool deepCopy = true )
```

Create an auxiliary information object by copying from another object.

Parameters

<i>other</i>	Another auxiliary information object from which the information will be copied.
<i>tree</i>	The node that holds the auxiliary information.
<i>deepCopy</i>	If false, the new object uses the same memory (not used here).

39.496.3.4 HilbertRTreeAuxiliaryInformation() [4 / 4]

```
HilbertRTreeAuxiliaryInformation (
    HilbertRTreeAuxiliaryInformation< TreeType, HilbertValueType > && other )
```

Create an auxiliary information object by moving from the other node.

Parameters

<i>other</i>	The object from which the information will be moved.
--------------	--

39.496.4 Member Function Documentation

39.496.4.1 Children()

```
static const std::vector<TreeType*> Children (
    const TreeType * tree ) [inline], [static]
```

Return the children vector of the tree.

Definition at line 124 of file `hilbert_r_tree_auxiliary_information.hpp`.

39.496.4.2 HandleNodeInsertion()

```
bool HandleNodeInsertion (
    TreeType * node,
    TreeType * nodeToInsert,
    bool insertionLevel )
```

The Hilbert R tree requires to insert nodes according to their Hilbert value.

This method should take care of it. It returns false if it does nothing and true if it handles the insertion process.

Parameters

<i>node</i>	The node in which the nodeToInsert is being inserted.
<i>nodeToInsert</i>	The node being inserted.
<i>insertionLevel</i>	The level of the tree at which the nodeToInsert should be inserted.

39.496.4.3 HandleNodeRemoval()

```
bool HandleNodeRemoval (
    TreeType * node,
    const size_t nodeIndex )
```

The Hilbert R tree requires all nodes to be arranged according to their Hilbert value.

This method should take care of saving this property after the deletion process. It returns false if it does nothing and true if it handles the deletion process.

Parameters

<i>node</i>	The node from which the node is being deleted.
<i>nodeIndex</i>	The index of the node being deleted.

39.496.4.4 HandlePointDeletion()

```
bool HandlePointDeletion (
```

```
TreeType * node,
const size_t localIndex )
```

The Hilbert R tree requires all points to be arranged according to their Hilbert value.

This method should take care of saving this property after the deletion process. It returns false if it does nothing and true if it handles the deletion process.

Parameters

<i>node</i>	The node from which the point is being deleted.
<i>localIndex</i>	The index of the point being deleted.

39.496.4.5 HandlePointInsertion()

```
bool HandlePointInsertion (
    TreeType * node,
    const size_t point )
```

The Hilbert R tree requires to insert points according to their Hilbert value.

This method should take care of it. It returns false if it does nothing and true if it handles the insertion process.

Parameters

<i>node</i>	The node in which the point is being inserted.
<i>point</i>	The number of the point being inserted.

39.496.4.6 HilbertValue() [1/2]

```
const HilbertValueType< ElemType>& HilbertValue ( ) const [inline]
```

Return the largest Hilbert value of a point covered by the node.

Definition at line 133 of file hilbert_r_tree_auxiliary_information.hpp.

39.496.4.7 HilbertValue() [2/2]

```
HilbertValueType< ElemType>& HilbertValue ( ) [inline]
```

Modify the largest Hilbert value of a point covered by the node.

Definition at line 136 of file hilbert_r_tree_auxiliary_information.hpp.

References HilbertRTreeAuxiliaryInformation< TreeType, HilbertValueType >::serialize().

39.496.4.8 NullifyData()

```
void NullifyData ( )
```

Clear memory.

39.496.4.9 operator=()

```
HilbertRTreeAuxiliaryInformation& operator= (
    const HilbertRTreeAuxiliaryInformation< TreeType, HilbertValueType > & other )
```

Copy the auxiliary information.

Parameters

<i>other</i>	The object from which the information will be moved.
--------------	--

39.496.4.10 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the information.

Referenced by `HilbertRTreeAuxiliaryInformation< TreeType, HilbertValueType >::HilbertValue()`.

39.496.4.11 UpdateAuxiliaryInfo()

```
bool UpdateAuxiliaryInfo (
    TreeType * node )
```

Update the auxiliary information in the node.

The method returns true if the update should be propagated downward.

Parameters

<i>node</i>	The node in which the auxiliary information being update.
-------------	---

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ hilbert_r_tree_auxiliary_information.↵
hpp`

39.497 HilbertRTreeDescentHeuristic Class Reference

This class chooses the best child of a node in a Hilbert R tree when inserting a new point.

Static Public Member Functions

- `template<typename TreeType >`
`static size_t ChooseDescentNode (const TreeType *node, const size_t point)`
Evaluate the node using a heuristic.
- `template<typename TreeType >`
`static size_t ChooseDescentNode (const TreeType *node, const TreeType *insertedNode)`
Evaluate the node using a heuristic.

39.497.1 Detailed Description

This class chooses the best child of a node in a Hilbert R tree when inserting a new point.

This is done, in this class, by using the Hilbert value of the point to be inserted.

Definition at line 26 of file `hilbert_r_tree_descent_heuristic.hpp`.

39.497.2 Member Function Documentation

39.497.2.1 ChooseDescentNode() [1/2]

```
static size_t ChooseDescentNode (
    const TreeType * node,
    const size_t point ) [static]
```

Evaluate the node using a heuristic.

Returns the number of the node with minimum largest Hilbert value that is greater than the Hilbert value of the point being inserted.

Parameters

<i>node</i>	The node that is being evaluated.
<i>point</i>	The number of the point that is being inserted.

39.497.2.2 ChooseDescentNode() [2/2]

```
static size_t ChooseDescentNode (
    const TreeType * node,
    const TreeType * insertedNode ) [static]
```

Evaluate the node using a heuristic.

Returns the number of the node with minimum largest Hilbert value that is greater than the largest Hilbert value of the point being inserted.

Parameters

<i>node</i>	The node that is being evaluated.
<i>insertedNode</i>	The node that is being inserted.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ **hilbert_r_tree_descent_heuristic.**↵
hpp

39.498 HilbertRTreeSplit< splitOrder > Class Template Reference

The splitting procedure for the Hilbert R tree.

Static Public Member Functions

- template<typename TreeType >
static void **SplitLeafNode** (TreeType *tree, std::vector< bool > &relevels)
Split a leaf node using the "default" algorithm.
- template<typename TreeType >
static bool **SplitNonLeafNode** (TreeType *tree, std::vector< bool > &relevels)
Split a non-leaf node using the "default" algorithm.

39.498.1 Detailed Description

```
template<size_t splitOrder = 2>
class mlpack::tree::HilbertRTreeSplit< splitOrder >
```

The splitting procedure for the Hilbert R tree.

The template parameter `splitOrder` is the order of the splitting policy. The Hilbert R tree splits a node on overflow, turning `splitOrder` nodes into `(splitOrder + 1)` nodes.

Template Parameters

<i>splitOrder</i>	Number of nodes to split.
-------------------	---------------------------

Definition at line 29 of file `hilbert_r_tree_split.hpp`.

39.498.2 Member Function Documentation

39.498.2.1 SplitLeafNode()

```
static void SplitLeafNode (
    TreeType * tree,
    std::vector< bool > & relevels ) [static]
```

Split a leaf node using the "default" algorithm.

If necessary, this split will propagate upwards through the tree.

Parameters

<i>node</i>	The node that is being split.
<i>relevels</i>	Not used.

39.498.2.2 SplitNonLeafNode()

```
static bool SplitNonLeafNode (
    TreeType * tree,
    std::vector< bool > & relevels ) [static]
```

Split a non-leaf node using the "default" algorithm.

If this is a root node, the tree increases in depth.

Parameters

<i>node</i>	The node that is being split.
<i>relevels</i>	Not used.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ **hilbert_r_tree_split.hpp**

39.499 HoeffdingCategoricalSplit< FitnessFunction > Class Template Reference

This is the standard Hoeffding-bound categorical feature proposed in the paper below:

Public Types

- typedef **CategoricalSplitInfo** **SplitInfo**
*The type of split information required by the **HoeffdingCategoricalSplit** (p. 2104).*

Public Member Functions

- **HoeffdingCategoricalSplit** (const size_t numCategories=0, const size_t numClasses=0)
*Create the **HoeffdingCategoricalSplit** (p. 2104) given a number of categories for this dimension and a number of classes.*
- **HoeffdingCategoricalSplit** (const size_t numCategories, const size_t numClasses, const **HoeffdingCategoricalSplit** &other)
*Create the **HoeffdingCategoricalSplit** (p. 2104) given a number of categories for this dimension and a number of classes and another **HoeffdingCategoricalSplit** (p. 2104) to take parameters from.*
- void **EvaluateFitnessFunction** (double &bestFitness, double &secondBestFitness) const
Given the points seen so far, evaluate the fitness function, returning the gain for the best possible split and the second best possible split.
- size_t **MajorityClass** () const
Get the majority class seen so far.
- double **MajorityProbability** () const
Get the probability of the majority class given the points seen so far.
- size_t **NumChildren** () const
Return the number of children, if the node were to split.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the categorical split.
- void **Split** (arma::Col< size_t > &childMajorities, **SplitInfo** &splitInfo)
Gather the information for a split: get the labels of the child majorities, and initialize the SplitInfo object.
- template<typename eT >
void **Train** (eT value, const size_t label)
Train on the given value with the given label.

39.499.1 Detailed Description

```
template<typename FitnessFunction>
class mlpack::tree::HoeffdingCategoricalSplit< FitnessFunction >
```

This is the standard Hoeffding-bound categorical feature proposed in the paper below:

```
@inproceedings{domingos2000mining,
  title={Mining High-Speed Data Streams},
  author={Domingos, P. and Hulten, G.},
  year={2000},
  booktitle={Proceedings of the Sixth ACM SIGKDD International Conference on
    Knowledge Discovery and Data Mining (KDD '00)},
  pages={71--80}
}
```

This class will track the sufficient statistics of the training points it has seen. The HoeffdingSplit class (and other related classes) can use this class to track categorical features and split decision tree nodes.

Template Parameters

<i>FitnessFunction</i>	Fitness function to use for calculating gain.
------------------------	---

Definition at line 44 of file hoeffding_categorical_split.hpp.

39.499.2 Member Typedef Documentation

39.499.2.1 SplitInfo

```
typedef CategoricalSplitInfo SplitInfo
```

The type of split information required by the **HoeffdingCategoricalSplit** (p.2104).

Definition at line 48 of file hoeffding_categorical_split.hpp.

39.499.3 Constructor & Destructor Documentation

39.499.3.1 HoeffdingCategoricalSplit() [1/2]

```
HoeffdingCategoricalSplit (
    const size_t numCategories = 0,
    const size_t numClasses = 0 )
```

Create the **HoeffdingCategoricalSplit** (p.2104) given a number of categories for this dimension and a number of classes.

Parameters

<i>numCategories</i>	Number of categories in this dimension.
<i>numClasses</i>	Number of classes in this dimension.

39.499.3.2 HoeffdingCategoricalSplit() [2/2]

```

HoeffdingCategoricalSplit (
    const size_t numCategories,
    const size_t numClasses,
    const HoeffdingCategoricalSplit< FitnessFunction > & other )

```

Create the **HoeffdingCategoricalSplit** (p.2104) given a number of categories for this dimension and a number of classes and another **HoeffdingCategoricalSplit** (p. 2104) to take parameters from.

In this particular case, there are no parameters to take, but this constructor is required by the **HoeffdingTree** (p. 2113) class.

39.499.4 Member Function Documentation

39.499.4.1 EvaluateFitnessFunction()

```

void EvaluateFitnessFunction (
    double & bestFitness,
    double & secondBestFitness ) const

```

Given the points seen so far, evaluate the fitness function, returning the gain for the best possible split and the second best possible split.

In this splitting technique, we only split one possible way, so *secondBestFitness* will always be 0.

Parameters

<i>bestFitness</i>	The fitness function result for this split.
<i>secondBestFitness</i>	This is always set to 0 (this split only splits one way).

39.499.4.2 MajorityClass()

```

size_t MajorityClass ( ) const

```

Get the majority class seen so far.

Referenced by HoeffdingCategoricalSplit< FitnessFunction >::NumChildren().

39.499.4.3 MajorityProbability()

```
double MajorityProbability ( ) const
```

Get the probability of the majority class given the points seen so far.

Referenced by HoeffdingCategoricalSplit< FitnessFunction >::NumChildren().

39.499.4.4 NumChildren()

```
size_t NumChildren ( ) const [inline]
```

Return the number of children, if the node were to split.

Definition at line 93 of file hoeffding_categorical_split.hpp.

References HoeffdingCategoricalSplit< FitnessFunction >::MajorityClass(), HoeffdingCategoricalSplit< FitnessFunction >::MajorityProbability(), and HoeffdingCategoricalSplit< FitnessFunction >::Split().

39.499.4.5 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the categorical split.

Definition at line 111 of file hoeffding_categorical_split.hpp.

39.499.4.6 Split()

```
void Split (
    arma::Col< size_t > & childMajorities,
    SplitInfo & splitInfo )
```

Gather the information for a split: get the labels of the child majorities, and initialize the SplitInfo object.

Parameters

<i>childMajorities</i>	Majorities of child nodes to be created.
<i>splitInfo</i>	Information for splitting.

Referenced by `HoeffdingCategoricalSplit< FitnessFunction >::NumChildren()`.

39.499.4.7 Train()

```
void Train (
    eT value,
    const size_t label )
```

Train on the given value with the given label.

Parameters

<i>value</i>	Value to train on.
<i>label</i>	Label to train on.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ hoeffding_categorical_split.hpp`

39.500 HoeffdingNumericSplit< FitnessFunction, ObservationType > Class Template Reference

The **HoeffdingNumericSplit** (p. 2108) class implements the numeric feature splitting strategy alluded to by Domingos and Hulten in the following paper:

Public Types

- typedef **NumericSplitInfo**< ObservationType > **SplitInfo**

*The splitting information type required by the **HoeffdingNumericSplit** (p. 2108).*

Public Member Functions

- **HoeffdingNumericSplit** (const size_t numClasses=0, const size_t bins=10, const size_t observationsBefore←Binning=100)
Create the **HoeffdingNumericSplit** (p. 2108) class, and specify some basic parameters about how the binning should take place.
- **HoeffdingNumericSplit** (const size_t numClasses, const **HoeffdingNumericSplit** &other)
Create the **HoeffdingNumericSplit** (p. 2108) class, using the parameters from the given other split object.
- size_t **Bins** () const
Return the number of bins.
- void **EvaluateFitnessFunction** (double &bestFitness, double &secondBestFitness) const
Evaluate the fitness function given what has been calculated so far.
- size_t **MajorityClass** () const
Return the majority class.
- double **MajorityProbability** () const
Return the probability of the majority class.
- size_t **NumChildren** () const
Return the number of children if this node splits on this feature.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the object.
- void **Split** (arma::Col< size_t > &childMajorities, **SplitInfo** &splitInfo) const
Return the majority class of each child to be created, if a split on this dimension was performed.
- void **Train** (ObservationType value, const size_t label)
Train the **HoeffdingNumericSplit** (p. 2108) on the given observed value (remember that this object only cares about the information for a single feature, not an entire point).

39.500.1 Detailed Description

```
template<typename FitnessFunction, typename ObservationType = double>
class mlpack::tree::HoeffdingNumericSplit< FitnessFunction, ObservationType >
```

The **HoeffdingNumericSplit** (p. 2108) class implements the numeric feature splitting strategy alluded to by Domingos and Hulten in the following paper:

```
@inproceedings{domingos2000mining,
  title={Mining High-Speed Data Streams},
  author={Domingos, P. and Hulten, G.},
  year={2000},
  booktitle={Proceedings of the Sixth ACM SIGKDD International Conference on
    Knowledge Discovery and Data Mining (KDD '00)},
  pages={71--80}
}
```

The strategy alluded to is very simple: we discretize the numeric features that we see. But in this case, we don't know how many bins we have, which makes things a little difficult. This class only makes binary splits, and has a maximum number of bins. The binning strategy is simple: the split caches the minimum and maximum value of points seen so far, and when the number of points hits a predefined threshold, the cached minimum-maximum range is equally split into bins, and splitting proceeds in the same way as with the categorical splits. This is a simple and stupid strategy, so don't expect it to be the best possible thing you can do.

Template Parameters

<i>FitnessFunction</i>	Fitness function to use for calculating gain.
<i>ObservationType</i>	Type of observations in this dimension.

Definition at line 53 of file hoeffding_numeric_split.hpp.

39.500.2 Member Typedef Documentation

39.500.2.1 SplitInfo

```
typedef NumericSplitInfo<ObservationType> SplitInfo
```

The splitting information type required by the **HoeffdingNumericSplit** (p. 2108).

Definition at line 57 of file hoeffding_numeric_split.hpp.

39.500.3 Constructor & Destructor Documentation

39.500.3.1 HoeffdingNumericSplit() [1/2]

```
HoeffdingNumericSplit (
    const size_t numClasses = 0,
    const size_t bins = 10,
    const size_t observationsBeforeBinning = 100 )
```

Create the **HoeffdingNumericSplit** (p. 2108) class, and specify some basic parameters about how the binning should take place.

Parameters

<i>numClasses</i>	Number of classes.
<i>bins</i>	Number of bins.
<i>observationsBeforeBinning</i>	Number of points to see before binning is performed.

39.500.3.2 HoeffdingNumericSplit() [2/2]

```
HoeffdingNumericSplit (
    const size_t numClasses,
    const HoeffdingNumericSplit< FitnessFunction, ObservationType > & other )
```

Create the **HoeffdingNumericSplit** (p. 2108) class, using the parameters from the given other split object.

39.500.4 Member Function Documentation

39.500.4.1 Bins()

```
size_t Bins ( ) const [inline]
```

Return the number of bins.

Definition at line 120 of file hoeffding_numeric_split.hpp.

References `HoeffdingNumericSplit< FitnessFunction, ObservationType >::serialize()`.

39.500.4.2 EvaluateFitnessFunction()

```
void EvaluateFitnessFunction (
    double & bestFitness,
    double & secondBestFitness ) const
```

Evaluate the fitness function given what has been calculated so far.

In this case, if binning has not yet been performed, 0 will be returned (i.e., no gain). Because this split can only split one possible way, `secondBestFitness` (the fitness function for the second best possible split) will be set to 0.

Parameters

<i>bestFitness</i>	Value of the fitness function for the best possible split.
<i>secondBestFitness</i>	Value of the fitness function for the second best possible split (always 0 for this split).

39.500.4.3 MajorityClass()

```
size_t MajorityClass ( ) const
```

Return the majority class.

Referenced by `HoeffdingNumericSplit< FitnessFunction, ObservationType >::NumChildren()`.

39.500.4.4 MajorityProbability()

```
double MajorityProbability ( ) const
```

Return the probability of the majority class.

Referenced by `HoeffdingNumericSplit< FitnessFunction, ObservationType >::NumChildren()`.

39.500.4.5 NumChildren()

```
size_t NumChildren ( ) const [inline]
```

Return the number of children if this node splits on this feature.

Definition at line 106 of file `hoeffding_numeric_split.hpp`.

References `HoeffdingNumericSplit< FitnessFunction, ObservationType >::MajorityClass()`, `HoeffdingNumericSplit< FitnessFunction, ObservationType >::MajorityProbability()`, and `HoeffdingNumericSplit< FitnessFunction, ObservationType >::Split()`.

39.500.4.6 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the object.

Referenced by `HoeffdingNumericSplit< FitnessFunction, ObservationType >::Bins()`.

39.500.4.7 Split()

```
void Split (
    arma::Col< size_t > & childMajorities,
    SplitInfo & splitInfo ) const
```

Return the majority class of each child to be created, if a split on this dimension was performed.

Also create the split object.

Referenced by HoeffdingNumericSplit< FitnessFunction, ObservationType >::NumChildren().

39.500.4.8 Train()

```
void Train (
    ObservationType value,
    const size_t label )
```

Train the **HoeffdingNumericSplit** (p.2108) on the given observed value (remember that this object only cares about the information for a single feature, not an entire point).

Parameters

<i>value</i>	Value in the dimension that this HoeffdingNumericSplit (p.2108) refers to.
<i>label</i>	Label of the given point.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ **hoeffding_numeric_split.hpp**

39.501 HoeffdingTree< FitnessFunction, NumericSplitType, CategoricalSplitType > Class Template Reference

The **HoeffdingTree** (p.2113) object represents all of the necessary information for a Hoeffding-bound-based decision tree.

Public Types

- typedef CategoricalSplitType< FitnessFunction > **CategoricalSplit**
Allow access to the categorical split type.
- typedef NumericSplitType< FitnessFunction > **NumericSplit**
Allow access to the numeric split type.

Public Member Functions

- `template<typename MatType >`
HoeffdingTree (const MatType &data, const **data::DatasetInfo** &datasetInfo, const arma::Row< size_t > &labels, const size_t numClasses, const bool batchTraining=true, const double successProbability=0.95, const size_t maxSamples=0, const size_t checkInterval=100, const size_t minSamples=100, const CategoricalSplitType< FitnessFunction > &categoricalSplitIn=CategoricalSplitType< FitnessFunction >(0, 0), const NumericSplitType< FitnessFunction > &numericSplitIn=NumericSplitType< FitnessFunction >(0))
Construct the Hoeffding tree with the given parameters and given training data.
- **HoeffdingTree** (const **data::DatasetInfo** &datasetInfo, const size_t numClasses, const double successProbability=0.95, const size_t maxSamples=0, const size_t checkInterval=100, const size_t minSamples=100, const CategoricalSplitType< FitnessFunction > &categoricalSplitIn=CategoricalSplitType< FitnessFunction >(0, 0), const NumericSplitType< FitnessFunction > &numericSplitIn=NumericSplitType< FitnessFunction >(0), std::unordered_map< size_t, std::pair< size_t, size_t >> *dimensionMappings=NULL)
Construct the Hoeffding tree with the given parameters, but training on no data.
- **HoeffdingTree** ()
Construct a Hoeffding tree with no data and no information.
- **HoeffdingTree** (const **HoeffdingTree** &other)
Copy another tree (warning: this will duplicate the tree entirely, and may use a lot of memory).
- `~HoeffdingTree` ()
Clean up memory.
- `template<typename VecType >`
size_t **CalculateDirection** (const VecType &point) const
Given a point and that this node is not a leaf, calculate the index of the child node this point would go towards.
- size_t **CheckInterval** () const
Get the number of samples before a split check is performed.
- void **CheckInterval** (const size_t checkInterval)
Modify the number of samples before a split check is performed.
- const **HoeffdingTree & Child** (const size_t i) const
Get a child.
- **HoeffdingTree & Child** (const size_t i)
Modify a child.
- `template<typename VecType >`
size_t **Classify** (const VecType &point) const
Classify the given point, using this node and the entire (sub)tree beneath it.
- `template<typename VecType >`
void **Classify** (const VecType &point, size_t &prediction, double &probability) const
Classify the given point and also return an estimate of the probability that the prediction is correct.
- `template<typename MatType >`
void **Classify** (const MatType &data, arma::Row< size_t > &predictions) const
Classify the given points, using this node and the entire (sub)tree beneath it.
- `template<typename MatType >`
void **Classify** (const MatType &data, arma::Row< size_t > &predictions, arma::rowvec &probabilities) const
Classify the given points, using this node and the entire (sub)tree beneath it.
- void **CreateChildren** ()
Given that this node should split, create the children.
- size_t **MajorityClass** () const
Get the majority class.
- size_t & **MajorityClass** ()

- *Modify the majority class.*
- double **MajorityProbability** () const
Get the probability of the majority class (based on training samples).
- double & **MajorityProbability** ()
Modify the probability of the majority class.
- size_t **MaxSamples** () const
Get the maximum number of samples before a split is forced.
- void **MaxSamples** (const size_t maxSamples)
Modify the maximum number of samples before a split is forced.
- size_t **MinSamples** () const
Get the minimum number of samples for a split.
- void **MinSamples** (const size_t minSamples)
Modify the minimum number of samples for a split.
- size_t **NumChildren** () const
Get the number of children.
- size_t **NumDescendants** () const
Get the size of the Hoeffding Tree.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the split.
- size_t **SplitCheck** ()
Check if a split would satisfy the conditions of the Hoeffding bound with the node's specified success probability.
- size_t **SplitDimension** () const
Get the splitting dimension (size_t(-1) if no split).
- double **SuccessProbability** () const
Get the confidence required for a split.
- void **SuccessProbability** (const double successProbability)
Modify the confidence required for a split.
- template<typename MatType >
void **Train** (const MatType &data, const arma::Row< size_t > &labels, const bool batchTraining=true)
Train on a set of points, either in streaming mode or in batch mode, with the given labels.
- template<typename MatType >
void **Train** (const MatType &data, const **data::DatasetInfo** &info, const arma::Row< size_t > &labels, const bool batchTraining=true)
Train on a set of points, either in streaming mode or in batch mode, with the given labels and the given DatasetInfo.
- template<typename VecType >
void **Train** (const VecType &point, const size_t label)
Train on a single point in streaming mode, with the given label.

39.501.1 Detailed Description

```
template<typename FitnessFunction = GinImpurity, template< typename > class NumericSplitType = HoeffdingDoubleNumericSplit,
template< typename > class CategoricalSplitType = HoeffdingCategoricalSplit>
class mlpack::tree::HoeffdingTree< FitnessFunction, NumericSplitType, CategoricalSplitType >
```

The **HoeffdingTree** (p. 2113) object represents all of the necessary information for a Hoeffding-bound-based decision tree.

This class is able to train on samples in streaming settings and batch settings, and perform splits based on the Hoeffding bound. The Hoeffding tree (also known as the "very fast decision tree" – VFDT) is described in the following paper:

```
@inproceedings{domingos2000mining,
  title={Mining High-Speed Data Streams}},
  author={Domingos, P. and Hulten, G.},
  year={2000},
  booktitle={Proceedings of the Sixth ACM SIGKDD International Conference
    on Knowledge Discovery and Data Mining (KDD '00)},
  pages={71--80}
}
```

The class is modular, and takes three template parameters. The first, `FitnessFunction`, is the fitness function that should be used to determine whether a split is beneficial; examples might be **GiniImpurity** (p. 2092) or **Information↔Gain** (p. 2138). The `NumericSplitType` determines how numeric attributes are handled, and the `CategoricalSplitType` determines how categorical attributes are handled. As far as the actual splitting goes, the meat of the splitting procedure will be contained in those two classes.

Template Parameters

<i><code>FitnessFunction</code></i>	Fitness function to use.
<i><code>NumericSplitType</code></i>	Technique for splitting numeric features.
<i><code>CategoricalSplitType</code></i>	Technique for splitting categorical features.

Definition at line 61 of file `hoeffding_tree.hpp`.

39.501.2 Member Typedef Documentation

39.501.2.1 CategoricalSplit

```
typedef CategoricalSplitType<FitnessFunction> CategoricalSplit
```

Allow access to the categorical split type.

Definition at line 67 of file `hoeffding_tree.hpp`.

39.501.2.2 NumericSplit

```
typedef NumericSplitType<FitnessFunction> NumericSplit
```

Allow access to the numeric split type.

Definition at line 65 of file `hoeffding_tree.hpp`.

39.501.3 Constructor & Destructor Documentation

39.501.3.1 HoeffdingTree() [1/4]

```

HoeffdingTree (
    const MatType & data,
    const data::DatasetInfo & datasetInfo,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const bool batchTraining = true,
    const double successProbability = 0.95,
    const size_t maxSamples = 0,
    const size_t checkInterval = 100,
    const size_t minSamples = 100,
    const CategoricalSplitType< FitnessFunction > & categoricalSplitIn = CategoricalSplitType< FitnessFunction >(),
    const NumericSplitType< FitnessFunction > & numericSplitIn = NumericSplitType< FitnessFunction >()
)

```

Construct the Hoeffding tree with the given parameters and given training data.

The tree may be trained either in batch mode (which looks at all points before splitting, and propagates these points to the created children for further training), or in streaming mode, where each point is only considered once. (In general, batch mode will give better-performing trees, but will have higher memory and runtime costs for the same dataset.)

Parameters

<i>data</i>	Dataset to train on.
<i>datasetInfo</i>	Information on the dataset (types of each feature).
<i>labels</i>	Labels of each point in the dataset.
<i>numClasses</i>	Number of classes in the dataset.
<i>batchTraining</i>	Whether or not to train in batch.
<i>successProbability</i>	Probability of success required in Hoeffding bounds before a split can happen.
<i>maxSamples</i>	Maximum number of samples before a split is forced (0 never forces a split); ignored in batch training mode.
<i>checkInterval</i>	Number of samples required before each split; ignored in batch training mode.
<i>minSamples</i>	If the node has seen this many points or fewer, no split will be allowed.

39.501.3.2 HoeffdingTree() [2/4]

```

HoeffdingTree (
    const data::DatasetInfo & datasetInfo,
    const size_t numClasses,
    const double successProbability = 0.95,
    const size_t maxSamples = 0,
    const size_t checkInterval = 100,
    const size_t minSamples = 100,
    const CategoricalSplitType< FitnessFunction > & categoricalSplitIn = CategoricalSplitType< FitnessFunction >(),
    const NumericSplitType< FitnessFunction > & numericSplitIn = NumericSplitType< FitnessFunction >()
)

```

```

        std::unordered_map< size_t, std::pair< size_t, size_t >> * dimensionMappings = NULL
    )

```

Construct the Hoeffding tree with the given parameters, but training on no data.

The dimensionMappings parameter is only used if it is desired that this node does not create its own dimensionMappings object (for instance, if this is a child of another node in the tree).

Parameters

<i>dimensionality</i>	Dimensionality of the dataset.
<i>numClasses</i>	Number of classes in the dataset.
<i>datasetInfo</i>	Information on the dataset (types of each feature).
<i>successProbability</i>	Probability of success required in Hoeffding bound before a split can happen.
<i>maxSamples</i>	Maximum number of samples before a split is forced.
<i>checkInterval</i>	Number of samples required before each split check.
<i>minSamples</i>	If the node has seen this many points or fewer, no split will be allowed.
<i>dimensionMappings</i>	Mappings from dimension indices to positions in numeric and categorical split vectors. If left NULL, a new one will be created.

39.501.3.3 HoeffdingTree() [3 / 4]

```
HoeffdingTree ( )
```

Construct a Hoeffding tree with no data and no information.

Be sure to call **Train()** (p. 2125) before trying to use the tree.

39.501.3.4 HoeffdingTree() [4 / 4]

```

HoeffdingTree (
    const HoeffdingTree< FitnessFunction, NumericSplitType, CategoricalSplitType > &
    other )

```

Copy another tree (warning: this will duplicate the tree entirely, and may use a lot of memory).

Make sure it's what you want before you do it).

Parameters

<i>other</i>	Tree to copy.
--------------	---------------

39.501.3.5 ~HoeffdingTree()

`~ HoeffdingTree ()`

Clean up memory.

39.501.4 Member Function Documentation

39.501.4.1 CalculateDirection()

```
size_t CalculateDirection (
    const VecType & point ) const
```

Given a point and that this node is not a leaf, calculate the index of the child node this point would go towards.

This method is primarily used by the **Classify()** (p. 2120) function, but it can be used in a standalone sense too.

Parameters

<i>point</i>	Point to classify.
--------------	--------------------

Referenced by HoeffdingTree< FitnessFunction, NumericSplitType, CategoricalSplitType >::CheckInterval().

39.501.4.2 CheckInterval() [1/2]

```
size_t CheckInterval ( ) const [inline]
```

Get the number of samples before a split check is performed.

Definition at line 233 of file hoeffding_tree.hpp.

References HoeffdingTree< FitnessFunction, NumericSplitType, CategoricalSplitType >::CalculateDirection(), HoeffdingTree< FitnessFunction, NumericSplitType, CategoricalSplitType >::Classify(), HoeffdingTree< FitnessFunction, NumericSplitType, CategoricalSplitType >::CreateChildren(), HoeffdingTree< FitnessFunction, NumericSplitType, CategoricalSplitType >::NumDescendants(), and HoeffdingTree< FitnessFunction, NumericSplitType, CategoricalSplitType >::serialize().

39.501.4.3 CheckInterval() [2/2]

```
void CheckInterval (
    const size_t checkInterval )
```

Modify the number of samples before a split check is performed.

39.501.4.4 Child() [1/2]

```
const HoeffdingTree& Child (
    const size_t i ) const [inline]
```

Get a child.

Definition at line 213 of file `hoeffding_tree.hpp`.

39.501.4.5 Child() [2/2]

```
HoeffdingTree& Child (
    const size_t i ) [inline]
```

Modify a child.

Definition at line 215 of file `hoeffding_tree.hpp`.

39.501.4.6 Classify() [1/4]

```
size_t Classify (
    const VecType & point ) const
```

Classify the given point, using this node and the entire (sub)tree beneath it.

The predicted label is returned.

Parameters

<i>point</i>	Point to classify.
--------------	--------------------

Returns

Predicted label of point.

Referenced by HoeffdingTree< FitnessFunction, NumericSplitType, CategoricalSplitType >::CheckInterval().

39.501.4.7 Classify() [2/4]

```
void Classify (
    const VecType & point,
    size_t & prediction,
    double & probability ) const
```

Classify the given point and also return an estimate of the probability that the prediction is correct.

(This estimate is simply the probability that a training point was from the majority class in the leaf that this point binned to.)

Parameters

<i>point</i>	Point to classify.
<i>prediction</i>	Predicted label of point.
<i>probability</i>	An estimate of the probability that the prediction is correct.

39.501.4.8 Classify() [3/4]

```
void Classify (
    const MatType & data,
    arma::Row< size_t > & predictions ) const
```

Classify the given points, using this node and the entire (sub)tree beneath it.

The predicted labels for each point are returned.

Parameters

<i>data</i>	Points to classify.
<i>predictions</i>	Predicted labels for each point.

39.501.4.9 Classify() [4/4]

```
void Classify (
    const MatType & data,
    arma::Row< size_t > & predictions,
    arma::rowvec & probabilities ) const
```

Classify the given points, using this node and the entire (sub)tree beneath it.

The predicted labels for each point are returned, as well as an estimate of the probability that the prediction is correct for each point. This estimate is simply the **MajorityProbability()** (p. 2123) for the leaf that each point bins to.

Parameters

<i>data</i>	Points to classify.
<i>predictions</i>	Predicted labels for each point.
<i>probabilities</i>	Probability estimates for each predicted label.

39.501.4.10 CreateChildren()

```
void CreateChildren ( )
```

Given that this node should split, create the children.

Referenced by `HoeffdingTree< FitnessFunction, NumericSplitType, CategoricalSplitType >::CheckInterval()`.

39.501.4.11 MajorityClass() [1/2]

```
size_t MajorityClass ( ) const [inline]
```

Get the majority class.

Definition at line 200 of file `hoeffding_tree.hpp`.

39.501.4.12 MajorityClass() [2/2]

```
size_t& MajorityClass ( ) [inline]
```

Modify the majority class.

Definition at line 202 of file `hoeffding_tree.hpp`.

39.501.4.13 MajorityProbability() [1/2]

```
double MajorityProbability ( ) const [inline]
```

Get the probability of the majority class (based on training samples).

Definition at line 205 of file hoeffding_tree.hpp.

39.501.4.14 MajorityProbability() [2/2]

```
double& MajorityProbability ( ) [inline]
```

Modify the probability of the majority class.

Definition at line 207 of file hoeffding_tree.hpp.

39.501.4.15 MaxSamples() [1/2]

```
size_t MaxSamples ( ) const [inline]
```

Get the maximum number of samples before a split is forced.

Definition at line 228 of file hoeffding_tree.hpp.

39.501.4.16 MaxSamples() [2/2]

```
void MaxSamples (
    const size_t maxSamples )
```

Modify the maximum number of samples before a split is forced.

39.501.4.17 MinSamples() [1/2]

```
size_t MinSamples ( ) const [inline]
```

Get the minimum number of samples for a split.

Definition at line 223 of file hoeffding_tree.hpp.

39.501.4.18 MinSamples() [2/2]

```
void MinSamples (
    const size_t minSamples )
```

Modify the minimum number of samples for a split.

39.501.4.19 NumChildren()

```
size_t NumChildren ( ) const [inline]
```

Get the number of children.

Definition at line 210 of file hoeffding_tree.hpp.

39.501.4.20 NumDescendants()

```
size_t NumDescendants ( ) const
```

Get the size of the Hoeffding Tree.

Referenced by HoeffdingTree< FitnessFunction, NumericSplitType, CategoricalSplitType >::CheckInterval().

39.501.4.21 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the split.

Referenced by HoeffdingTree< FitnessFunction, NumericSplitType, CategoricalSplitType >::CheckInterval().

39.501.4.22 SplitCheck()

```
size_t SplitCheck ( )
```

Check if a split would satisfy the conditions of the Hoeffding bound with the node's specified success probability.

If so, the number of children that would be created is returned. If not, 0 is returned.

39.501.4.23 SplitDimension()

```
size_t SplitDimension ( ) const [inline]
```

Get the splitting dimension (size_t(-1) if no split).

Definition at line 197 of file hoeffding_tree.hpp.

39.501.4.24 SuccessProbability() [1/2]

```
double SuccessProbability ( ) const [inline]
```

Get the confidence required for a split.

Definition at line 218 of file hoeffding_tree.hpp.

39.501.4.25 SuccessProbability() [2/2]

```
void SuccessProbability (
    const double successProbability )
```

Modify the confidence required for a split.

39.501.4.26 Train() [1/3]

```
void Train (
    const MatType & data,
    const arma::Row< size_t > & labels,
    const bool batchTraining = true )
```

Train on a set of points, either in streaming mode or in batch mode, with the given labels.

Parameters

<i>data</i>	Data points to train on.
<i>label</i>	Labels of data points.
<i>batchTraining</i>	If true, perform training in batch.

39.501.4.27 Train() [2/3]

```
void Train (
    const MatType & data,
    const data::DatasetInfo & info,
    const arma::Row< size_t > & labels,
    const bool batchTraining = true )
```

Train on a set of points, either in streaming mode or in batch mode, with the given labels and the given DatasetInfo.

This will reset the tree.

39.501.4.28 Train() [3/3]

```
void Train (
    const VecType & point,
    const size_t label )
```

Train on a single point in streaming mode, with the given label.

Parameters

<i>point</i>	Point to train on.
<i>label</i>	Label of point to train on.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ **hoeffding_tree.hpp**

39.502 HoeffdingTreeModel Class Reference

This class is a serializable Hoeffding tree model that can hold four different types of Hoeffding trees.

Public Types

- typedef **HoeffdingTree**< **GiniImpurity**, **BinaryDoubleNumericSplit**, **HoeffdingCategoricalSplit** > **Gini**↩
BinaryTreeType
Convenience typedef for GINI_BINARY tree type.
- typedef **HoeffdingTree**< **GiniImpurity**, **HoeffdingDoubleNumericSplit**, **HoeffdingCategoricalSplit** >
GiniHoeffdingTreeType
Convenience typedef for GINI_HOEFFDING tree type.
- typedef **HoeffdingTree**< **InformationGain**, **BinaryDoubleNumericSplit**, **HoeffdingCategoricalSplit** >
InfoBinaryTreeType
Convenience typedef for INFO_BINARY tree type.

- typedef **HoeffdingTree**< **InformationGain**, **HoeffdingDoubleNumericSplit**, **HoeffdingCategoricalSplit** > **InfoHoeffdingTreeType**

Convenience typedef for INFO_HOEFFDING tree type.

- enum **TreeType** {
GINI_HOEFFDING,
GINI_BINARY,
INFO_HOEFFDING,
INFO_BINARY }

This enumerates the four types of trees we can hold.

Public Member Functions

- **HoeffdingTreeModel** (const **TreeType** &type= **GINI_HOEFFDING**)
Construct the Hoeffding tree model, but don't initialize any tree.
- **HoeffdingTreeModel** (const **HoeffdingTreeModel** &other)
Copy the Hoeffding tree model from the given other model.
- **HoeffdingTreeModel** (**HoeffdingTreeModel** &&other)
Move the Hoeffding tree model from the given other model.
- **~HoeffdingTreeModel** ()
Clean up the given model.
- void **BuildModel** (const arma::mat &dataset, const **data::DatasetInfo** &datasetInfo, const arma::Row< size_t > &labels, const size_t numClasses, const bool batchTraining, const double successProbability, const size_t maxSamples, const size_t checkInterval, const size_t minSamples, const size_t bins, const size_t observationsBeforeBinning)
Train the model on the given dataset with the given labels.
- void **Classify** (const arma::mat &dataset, arma::Row< size_t > &predictions) const
Using the model, classify the given test points.
- void **Classify** (const arma::mat &dataset, arma::Row< size_t > &predictions, arma::rowvec &probabilities) const
Using the model, classify the given test points, returning class probabilities.
- size_t **NumNodes** () const
Get the number of nodes in the tree.
- **HoeffdingTreeModel** & **operator=** (const **HoeffdingTreeModel** &other)
Copy the Hoeffding tree model from the given other model.
- **HoeffdingTreeModel** & **operator=** (**HoeffdingTreeModel** &&other)
Move the Hoeffding tree model from the given other model.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the model.
- void **Train** (const arma::mat &dataset, const arma::Row< size_t > &labels, const bool batchTraining)
Train in streaming mode on the given dataset.

39.502.1 Detailed Description

This class is a serializable Hoeffding tree model that can hold four different types of Hoeffding trees.

It is meant to be used by the command-line program for Hoeffding trees.

Definition at line 27 of file hoeffding_tree_model.hpp.

39.502.2 Member Typedef Documentation

39.502.2.1 GiniBinaryTreeType

```
typedef HoeffdingTree< GiniImpurity, BinaryDoubleNumericSplit, HoeffdingCategoricalSplit>  
GiniBinaryTreeType
```

Convenience typedef for GINI_BINARY tree type.

Definition at line 44 of file hoeffding_tree_model.hpp.

39.502.2.2 GiniHoeffdingTreeType

```
typedef HoeffdingTree< GiniImpurity, HoeffdingDoubleNumericSplit, HoeffdingCategoricalSplit>  
GiniHoeffdingTreeType
```

Convenience typedef for GINI_HOEFFDING tree type.

Definition at line 41 of file hoeffding_tree_model.hpp.

39.502.2.3 InfoBinaryTreeType

```
typedef HoeffdingTree< InformationGain, BinaryDoubleNumericSplit, HoeffdingCategoricalSplit>  
InfoBinaryTreeType
```

Convenience typedef for INFO_BINARY tree type.

Definition at line 50 of file hoeffding_tree_model.hpp.

39.502.2.4 InfoHoeffdingTreeType

```
typedef HoeffdingTree< InformationGain, HoeffdingDoubleNumericSplit, HoeffdingCategorical↔  
Split> InfoHoeffdingTreeType
```

Convenience typedef for INFO_HOEFFDING tree type.

Definition at line 47 of file hoeffding_tree_model.hpp.

39.502.3 Member Enumeration Documentation

39.502.3.1 TreeType

```
enum TreeType
```

This enumerates the four types of trees we can hold.

Enumerator

GINI_HOEFFDING	
GINI_BINARY	
INFO_HOEFFDING	
INFO_BINARY	

Definition at line 31 of file hoeffding_tree_model.hpp.

39.502.4 Constructor & Destructor Documentation

39.502.4.1 HoeffdingTreeModel() [1/3]

```
HoeffdingTreeModel (  
    const TreeType & type = GINI_HOEFFDING )
```

Construct the Hoeffding tree model, but don't initialize any tree.

Be sure to call **Train()** (p. 2133) before doing anything with the model!

Parameters

<i>type</i>	Type of tree that will be used.
-------------	---------------------------------

39.502.4.2 HoeffdingTreeModel() [2/3]

```
HoeffdingTreeModel (  
    const HoeffdingTreeModel & other )
```

Copy the Hoeffding tree model from the given other model.

Parameters

<i>other</i>	Hoeffding tree model to copy.
--------------	-------------------------------

39.502.4.3 HoeffdingTreeModel() [3/3]

```
HoeffdingTreeModel (
    HoeffdingTreeModel && other )
```

Move the Hoeffding tree model from the given other model.

Parameters

<i>other</i>	Hoeffding tree model to move.
--------------	-------------------------------

39.502.4.4 ~HoeffdingTreeModel()

```
~ HoeffdingTreeModel ( )
```

Clean up the given model.

39.502.5 Member Function Documentation

39.502.5.1 BuildModel()

```
void BuildModel (
    const arma::mat & dataset,
    const data::DatasetInfo & datasetInfo,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const bool batchTraining,
    const double successProbability,
    const size_t maxSamples,
    const size_t checkInterval,
    const size_t minSamples,
    const size_t bins,
    const size_t observationsBeforeBinning )
```

Train the model on the given dataset with the given labels.

This method just passes to the appropriate HoeffdingTree<...> constructor, and will train with one pass over the dataset.

Parameters

<i>dataset</i>	Dataset to train on.
<i>datasetInfo</i>	Information about dimensions of dataset.
<i>labels</i>	Labels for training set.

Parameters

<i>numClasses</i>	Number of classes in dataset.
<i>batchTraining</i>	Whether or not to train in batch.
<i>successProbability</i>	Probability of success required in Hoeffding bound before a split can happen.
<i>maxSamples</i>	Maximum number of samples before a split is forced.
<i>checkInterval</i>	Number of samples required before each split check.
<i>minSamples</i>	If the node has seen this many points or fewer, no split will be allowed.
<i>bins</i>	Number of bins, for Hoeffding numeric split.
<i>observationsBeforeBinning</i>	Number of observations before binning, for Hoeffding numeric split.

39.502.5.2 Classify() [1/2]

```
void Classify (
    const arma::mat & dataset,
    arma::Row< size_t > & predictions ) const
```

Using the model, classify the given test points.

Be sure that **BuildModel()** (p. 2130) has been called first!

Parameters

<i>dataset</i>	Dataset to classify.
<i>predictions</i>	Vector to store predictions for test points in.

39.502.5.3 Classify() [2/2]

```
void Classify (
    const arma::mat & dataset,
    arma::Row< size_t > & predictions,
    arma::rowvec & probabilities ) const
```

Using the model, classify the given test points, returning class probabilities.

Parameters

<i>dataset</i>	Dataset to classify.
<i>predictions</i>	Vector to store predictions for test points in.
<i>probabilities</i>	Vector to store probabilities for test points in.

39.502.5.4 NumNodes()

```
size_t NumNodes ( ) const
```

Get the number of nodes in the tree.

39.502.5.5 operator=() [1/2]

```
HoeffdingTreeModel& operator= (
    const HoeffdingTreeModel & other )
```

Copy the Hoeffding tree model from the given other model.

Parameters

<i>other</i>	Hoeffding tree model to copy.
--------------	-------------------------------

39.502.5.6 operator=() [2/2]

```
HoeffdingTreeModel& operator= (
    HoeffdingTreeModel && other )
```

Move the Hoeffding tree model from the given other model.

Parameters

<i>other</i>	Hoeffding tree model to move.
--------------	-------------------------------

39.502.5.7 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the model.

Definition at line 169 of file hoeffding_tree_model.hpp.

References HoeffdingTreeModel::GINI_BINARY, HoeffdingTreeModel::GINI_HOEFFDING, HoeffdingTreeModel::INFO_BINARY, and HoeffdingTreeModel::INFO_HOEFFDING.

39.502.5.8 Train()

```
void Train (
    const arma::mat & dataset,
    const arma::Row< size_t > & labels,
    const bool batchTraining )
```

Train in streaming mode on the given dataset.

This takes one pass. Be sure that **BuildModel()** (p. 2130) has been called first!

Parameters

<i>dataset</i>	Dataset to train on.
<i>labels</i>	Labels for training set.
<i>batchTraining</i>	Whether or not to train in batch.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ **hoeffding_tree_model.hpp**

39.503 HyperplaneBase< BoundT, ProjVectorT > Class Template Reference

HyperplaneBase (p. 2133) defines a splitting hyperplane based on a projection vector and projection value.

Public Types

- typedef BoundT **BoundType**
Useful typedef for the bound type.
- typedef ProjVectorT **ProjVectorType**
Useful typedef for the projection vector type.

Public Member Functions

- **HyperplaneBase** ()
Empty Constructor.
- **HyperplaneBase** (const **ProjVectorType** &projVect, double splitVal)
Create the hyperplane with the specified projection vector and split value.
- template<typename VecType >
bool **Left** (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > !=0) const
Determine if the given point is to the left of the hyperplane, this means if the projection over the projection vector is negative or zero.
- bool **Left** (const **BoundType** &bound) const
Determine if the given bound is to the left of the hyperplane.
- template<typename VecType >
double **Project** (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > !=0) const
Project the given point on the projection vector and subtract the split value.
- template<typename VecType >
bool **Right** (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > !=0) const
Determine if the given point is to the right of the hyperplane, this means if the projection over the projection vector is positive.
- bool **Right** (const **BoundType** &bound) const
Determine if the given bound is to the right of the hyperplane.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialization.

39.503.1 Detailed Description

```
template<typename BoundT, typename ProjVectorT>
class mlpack::tree::HyperplaneBase< BoundT, ProjVectorT >
```

HyperplaneBase (p. 2133) defines a splitting hyperplane based on a projection vector and projection value.

Template Parameters

<i>BoundT</i>	The bound type considered.
<i>ProjVectorT</i>	Type of projection vector (AxisParallelProjVector (p. 1986), ProjVector (p. 2190)).

Definition at line 30 of file hyperplane.hpp.

39.503.2 Member Typedef Documentation

39.503.2.1 BoundType

```
typedef BoundT BoundType
```

Useful typedef for the bound type.

Definition at line 34 of file hyperplane.hpp.

39.503.2.2 ProjVectorType

```
typedef ProjVectorT ProjVectorType
```

Useful typedef for the projection vector type.

Definition at line 36 of file hyperplane.hpp.

39.503.3 Constructor & Destructor Documentation

39.503.3.1 HyperplaneBase() [1/2]

```
HyperplaneBase ( ) [inline]
```

Empty Constructor.

By default will consider all points to the left.

Definition at line 49 of file hyperplane.hpp.

39.503.3.2 HyperplaneBase() [2/2]

```
HyperplaneBase (  
    const ProjVectorType & projVect,  
    double splitVal ) [inline]
```

Create the hyperplane with the specified projection vector and split value.

Parameters

<i>projVect</i>	Projection vector.
<i>splitVal</i>	Split value.

Definition at line 59 of file hyperplane.hpp.

39.503.4 Member Function Documentation

39.503.4.1 Left() [1/2]

```
bool Left (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const [inline]
```

Determine if the given point is to the left of the hyperplane, this means if the projection over the projection vector is negative or zero.

Parameters

<i>point</i>	Point to be analyzed.
--------------	-----------------------

Definition at line 86 of file hyperplane.hpp.

References HyperplaneBase< BoundT, ProjVectorT >::Project().

39.503.4.2 Left() [2/2]

```
bool Left (
    const BoundType & bound ) const [inline]
```

Determine if the given bound is to the left of the hyperplane.

Parameters

<i>point</i>	Bound to be analyzed.
--------------	-----------------------

Definition at line 110 of file hyperplane.hpp.

39.503.4.3 Project()

```
double Project (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const [inline]
```


Project the given point on the projection vector and subtract the split value.

Parameters

<i>point</i>	Point to be projected.
--------------	------------------------

Definition at line 71 of file hyperplane.hpp.

Referenced by HyperplaneBase< BoundT, ProjVectorT >::Left(), and HyperplaneBase< BoundT, ProjVectorT >::Right().

39.503.4.4 Right() [1/2]

```
bool Right (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const [inline]
```

Determine if the given point is to the right of the hyperplane, this means if the projection over the projection vector is positive.

Parameters

<i>point</i>	Point to be analyzed.
--------------	-----------------------

Definition at line 99 of file hyperplane.hpp.

References HyperplaneBase< BoundT, ProjVectorT >::Project().

39.503.4.5 Right() [2/2]

```
bool Right (
    const BoundType & bound ) const [inline]
```

Determine if the given bound is to the right of the hyperplane.

Parameters

<i>point</i>	Bound to be analyzed.
--------------	-----------------------

Definition at line 122 of file hyperplane.hpp.

39.503.4.6 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialization.

Definition at line 133 of file hyperplane.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/ **hyperplane.hpp**

39.504 InformationGain Class Reference

The standard information gain criterion, used for calculating gain in decision trees.

Static Public Member Functions

- static double **Evaluate** (const arma::Mat< size_t > &counts)
Given the sufficient statistics of a proposed split, calculate the information gain if that split was to be used.
- template<bool UseWeights>
static double **Evaluate** (const arma::Row< size_t > &labels, const size_t numClasses, const arma::Row< double > &weights)
Given a set of labels, calculate the information gain of those labels.
- template<bool UseWeights, typename CountType >
static double **EvaluatePtr** (const CountType *counts, const size_t countLength, const CountType totalCount)
Evaluate the Gini impurity given a vector of class weight counts.
- static double **Range** (const size_t numClasses)
Return the range of the information gain for the given number of classes.
- static double **Range** (const size_t numClasses)
Return the range of the information gain for the given number of classes.

39.504.1 Detailed Description

The standard information gain criterion, used for calculating gain in decision trees.

Definition at line 25 of file information_gain.hpp.

39.504.2 Member Function Documentation**39.504.2.1 Evaluate()** [1/2]

```
static double Evaluate (
    const arma::Mat< size_t > & counts ) [inline], [static]
```

Given the sufficient statistics of a proposed split, calculate the information gain if that split was to be used.

The 'counts' matrix should contain the number of points in each class in each column, so the size of 'counts' is children x classes, where 'children' is the number of child nodes in the proposed split.

Parameters

<i>counts</i>	Matrix of sufficient statistics.
---------------	----------------------------------

Definition at line 31 of file `information_gain.hpp`.

39.504.2.2 Evaluate() [2/2]

```
static double Evaluate (  
    const arma::Row< size_t > & labels,  
    const size_t numClasses,  
    const arma::Row< double > & weights ) [inline], [static]
```

Given a set of labels, calculate the information gain of those labels.

Note that it is possible that due to floating-point representation issues, it is possible that the gain returned can be very slightly greater than 0! Thus, if you are checking for a perfect fit, be sure to use 'gain >= 0.0' not 'gain == 0.0'.

Parameters

<i>labels</i>	Labels of the dataset.
<i>numClasses</i>	Number of classes in the dataset.

Definition at line 59 of file `information_gain.hpp`.

39.504.2.3 EvaluatePtr()

```
static double EvaluatePtr (  
    const CountType * counts,  
    const size_t countLength,  
    const CountType totalCount ) [inline], [static]
```

Evaluate the Gini impurity given a vector of class weight counts.

Definition at line 32 of file `information_gain.hpp`.

39.504.2.4 Range() [1/2]

```
static double Range (  
    const size_t numClasses ) [inline], [static]
```

Return the range of the information gain for the given number of classes.

(That is, the difference between the maximum possible value and the minimum possible value.)

Definition at line 84 of file `information_gain.hpp`.

39.504.2.5 Range() [2/2]

```
static double Range (
    const size_t numClasses ) [inline], [static]
```

Return the range of the information gain for the given number of classes.

(That is, the difference between the maximum possible value and the minimum possible value.)

Parameters

<i>numClasses</i>	Number of classes in the dataset.
-------------------	-----------------------------------

Definition at line 202 of file `information_gain.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/information_gain.hpp`

39.505 IsSpillTree< TreeType > Struct Template Reference**Static Public Attributes**

- static const bool **value** = false

39.505.1 Detailed Description

```
template<typename TreeType>
struct mlpack::tree::IsSpillTree< TreeType >
```

Definition at line 21 of file `is_spill_tree.hpp`.

39.505.2 Member Data Documentation**39.505.2.1 value**

```
const bool value = false [static]
```

Definition at line 23 of file `is_spill_tree.hpp`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/is_spill_tree.hpp`

39.506 IsSpillTree< tree::SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > > Struct Template Reference

Static Public Attributes

- static const bool **value** = true

39.506.1 Detailed Description

```
template<typename MetricType, typename StatisticType, typename MatType, template< typename HyperplaneMetricType > class HyperplaneType, template< typename SplitMetricType, typename SplitMatType > class SplitType>
struct mpack::tree::IsSpillTree< tree::SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > >
```

Definition at line 34 of file is_spill_tree.hpp.

39.506.2 Member Data Documentation

39.506.2.1 value

```
const bool value = true [static]
```

Definition at line 37 of file is_spill_tree.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/ **is_spill_tree.hpp**

39.507 MeanSpaceSplit< MetricType, MatType > Class Template Reference

Static Public Member Functions

- template<typename HyperplaneType >
static bool **SplitSpace** (const typename HyperplaneType::BoundType &bound, const MatType &data, const arma::Col< size_t > &points, HyperplaneType &hyp)

Create a splitting hyperplane considering the mean of the values in a certain projection.

39.507.1 Detailed Description

```
template<typename MetricType, typename MatType>
class mlpack::tree::MeanSpaceSplit< MetricType, MatType >
```

Definition at line 23 of file mean_space_split.hpp.

39.507.2 Member Function Documentation

39.507.2.1 SplitSpace()

```
static bool SplitSpace (
    const typename HyperplaneType::BoundType & bound,
    const MatType & data,
    const arma::Col< size_t > & points,
    HyperplaneType & hyp ) [static]
```

Create a splitting hyperplane considering the mean of the values in a certain projection.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the tree.
<i>points</i>	Vector of indexes of points to be considered.
<i>hyp</i>	Resulting splitting hyperplane.

Returns

Flag to determine if split is possible.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/ **mean_space_split.hpp**

39.508 MeanSplit< BoundType, MatType > Class Template Reference

A binary space partitioning tree node is split into its left and right child.

Classes

- struct **SplitInfo**
An information about the partition.

Static Public Member Functions

- `template<typename VecType >`
`static bool AssignToLeftNode (const VecType &point, const SplitInfo &splitInfo)`
Indicates that a point should be assigned to the left subtree.
- `static size_t PerformSplit (MatType &data, const size_t begin, const size_t count, const SplitInfo &splitInfo)`
Perform the split process according to the information about the split.
- `static size_t PerformSplit (MatType &data, const size_t begin, const size_t count, const SplitInfo &splitInfo, std::vector< size_t > &oldFromNew)`
Perform the split process according to the information about the split and return the list of changed indices.
- `static bool SplitNode (const BoundType &bound, MatType &data, const size_t begin, const size_t count, SplitInfo &splitInfo)`
Find the partition of the node.

39.508.1 Detailed Description

```
template<typename BoundType, typename MatType = arma::mat>
class mpack::tree::MeanSplit< BoundType, MatType >
```

A binary space partitioning tree node is split into its left and right child.

The split is done in the dimension that has the maximum width. The points are divided into two parts based on the mean in this dimension.

Definition at line 29 of file mean_split.hpp.

39.508.2 Member Function Documentation

39.508.2.1 AssignToLeftNode()

```
static bool AssignToLeftNode (
    const VecType & point,
    const SplitInfo & splitInfo ) [inline], [static]
```

Indicates that a point should be assigned to the left subtree.

Parameters

<i>point</i>	The point that is being assigned.
<i>splitInfo</i>	An information about the split.

Definition at line 115 of file mean_split.hpp.

References `MeanSplit< BoundType, MatType >::SplitInfo::splitDimension`, and `MeanSplit< BoundType, MatType >::SplitInfo::splitVal`.

39.508.2.2 PerformSplit() [1/2]

```
static size_t PerformSplit (
    MatType & data,
    const size_t begin,
    const size_t count,
    const SplitInfo & splitInfo ) [inline], [static]
```

Perform the split process according to the information about the split.

This will order the dataset such that points that belong to the left subtree are on the left of the split column, and points from the right subtree are on the right side of the split column.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitInfo</i>	The information about the split.

Definition at line 73 of file `mean_split.hpp`.

39.508.2.3 PerformSplit() [2/2]

```
static size_t PerformSplit (
    MatType & data,
    const size_t begin,
    const size_t count,
    const SplitInfo & splitInfo,
    std::vector< size_t > & oldFromNew ) [inline], [static]
```

Perform the split process according to the information about the split and return the list of changed indices.

This will order the dataset such that points that belong to the left subtree are on the left of the split column, and points from the right subtree are on the right side of the split column.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitInfo</i>	The information about the split.
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.

Definition at line 98 of file mean_split.hpp.

39.508.2.4 SplitNode()

```
static bool SplitNode (
    const BoundType & bound,
    MatType & data,
    const size_t begin,
    const size_t count,
    SplitInfo & splitInfo ) [static]
```

Find the partition of the node.

This method fills up the dimension that will be used to split the node and the value according which the split will be performed.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitInfo</i>	An information about the split. This information contains the dimension and the value.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ **mean_split.hpp**

39.509 MeanSplit< BoundType, MatType >::SplitInfo Struct Reference

An information about the partition.

Public Attributes

- size_t **splitDimension**
The dimension to split the node on.
- double **splitVal**
The split in dimension *splitDimension* is based on this value.

39.509.1 Detailed Description

```
template<typename BoundType, typename MatType = arma::mat>
struct mlpack::tree::MeanSplit< BoundType, MatType >::SplitInfo
```

An information about the partition.

Definition at line 33 of file mean_split.hpp.

39.509.2 Member Data Documentation

39.509.2.1 splitDimension

```
size_t splitDimension
```

The dimension to split the node on.

Definition at line 36 of file mean_split.hpp.

Referenced by MeanSplit< BoundType, MatType >::AssignToLeftNode().

39.509.2.2 splitVal

```
double splitVal
```

The split in dimension splitDimension is based on this value.

Definition at line 38 of file mean_split.hpp.

Referenced by MeanSplit< BoundType, MatType >::AssignToLeftNode().

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ **mean_split.hpp**

39.510 MidpointSpaceSplit< MetricType, MatType > Class Template Reference

Static Public Member Functions

- template<typename HyperplaneType >
static bool **SplitSpace** (const typename HyperplaneType::BoundType &bound, const MatType &data, const arma::Col< size_t > &points, HyperplaneType &hyp)

Create a splitting hyperplane considering the midpoint of the values in a certain projection.

39.510.1 Detailed Description

```
template<typename MetricType, typename MatType>
class mlpack::tree::MidpointSpaceSplit< MetricType, MatType >
```

Definition at line 23 of file midpoint_space_split.hpp.

39.510.2 Member Function Documentation

39.510.2.1 SplitSpace()

```
static bool SplitSpace (
    const typename HyperplaneType::BoundType & bound,
    const MatType & data,
    const arma::Col< size_t > & points,
    HyperplaneType & hyp ) [static]
```

Create a splitting hyperplane considering the midpoint of the values in a certain projection.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the tree.
<i>points</i>	Vector of indexes of points to be considered.
<i>hyp</i>	Resulting splitting hyperplane.

Returns

Flag to determine if split is possible.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/ **midpoint_space_split.hpp**

39.511 MidpointSplit< BoundType, MatType > Class Template Reference

A binary space partitioning tree node is split into its left and right child.

Classes

- struct **SplitInfo**
A struct that contains an information about the split.

Static Public Member Functions

- `template<typename VecType >`
`static bool AssignToLeftNode (const VecType &point, const SplitInfo &splitInfo)`
Indicates that a point should be assigned to the left subtree.
- `static size_t PerformSplit (MatType &data, const size_t begin, const size_t count, const SplitInfo &splitInfo)`
Perform the split process according to the information about the split.
- `static size_t PerformSplit (MatType &data, const size_t begin, const size_t count, const SplitInfo &splitInfo, std::vector< size_t > &oldFromNew)`
Perform the split process according to the information about the split and return the list of changed indices.
- `static bool SplitNode (const BoundType &bound, MatType &data, const size_t begin, const size_t count, SplitInfo &splitInfo)`
Find the partition of the node.

39.511.1 Detailed Description

```
template<typename BoundType, typename MatType = arma::mat>
class mpack::tree::MidpointSplit< BoundType, MatType >
```

A binary space partitioning tree node is split into its left and right child.

The split is done in the dimension that has the maximum width. The points are divided into two parts based on the midpoint in this dimension.

Definition at line 30 of file `midpoint_split.hpp`.

39.511.2 Member Function Documentation

39.511.2.1 AssignToLeftNode()

```
static bool AssignToLeftNode (
    const VecType & point,
    const SplitInfo & splitInfo ) [inline], [static]
```

Indicates that a point should be assigned to the left subtree.

Parameters

<i>point</i>	The point that is being assigned.
<i>splitInfo</i>	An information about the split.

Definition at line 115 of file `midpoint_split.hpp`.

References MidpointSplit< BoundType, MatType >::SplitInfo::splitDimension, and MidpointSplit< BoundType, MatType >::SplitInfo::splitVal.

39.511.2.2 PerformSplit() [1/2]

```
static size_t PerformSplit (
    MatType & data,
    const size_t begin,
    const size_t count,
    const SplitInfo & splitInfo ) [inline], [static]
```

Perform the split process according to the information about the split.

This will order the dataset such that points that belong to the left subtree are on the left of the split column, and points from the right subtree are on the right side of the split column.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitInfo</i>	The information about the split.

Definition at line 73 of file midpoint_split.hpp.

39.511.2.3 PerformSplit() [2/2]

```
static size_t PerformSplit (
    MatType & data,
    const size_t begin,
    const size_t count,
    const SplitInfo & splitInfo,
    std::vector< size_t > & oldFromNew ) [inline], [static]
```

Perform the split process according to the information about the split and return the list of changed indices.

This will order the dataset such that points that belong to the left subtree are on the left of the split column, and points from the right subtree are on the right side of the split column.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitInfo</i>	The information about the split.
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.

Definition at line 98 of file midpoint_split.hpp.

39.511.2.4 SplitNode()

```
static bool SplitNode (
    const BoundType & bound,
    MatType & data,
    const size_t begin,
    const size_t count,
    SplitInfo & splitInfo ) [static]
```

Find the partition of the node.

This method fills up the dimension that will be used to split the node and the value according which the split will be performed.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitInfo</i>	An information about the split. This information contains the dimension and the value.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ **midpoint_split.hpp**

39.512 MidpointSplit< BoundType, MatType >::SplitInfo Struct Reference

A struct that contains an information about the split.

Public Attributes

- size_t **splitDimension**
The dimension to split the node on.
- double **splitVal**
The split in dimension splitDimension is based on this value.

39.512.1 Detailed Description

```
template<typename BoundType, typename MatType = arma::mat>
struct mpack::tree::MidpointSplit< BoundType, MatType >::SplitInfo
```

A struct that contains an information about the split.

Definition at line 34 of file midpoint_split.hpp.

39.512.2 Member Data Documentation

39.512.2.1 splitDimension

```
size_t splitDimension
```

The dimension to split the node on.

Definition at line 37 of file midpoint_split.hpp.

Referenced by MidpointSplit< BoundType, MatType >::AssignToLeftNode().

39.512.2.2 splitVal

```
double splitVal
```

The split in dimension splitDimension is based on this value.

Definition at line 39 of file midpoint_split.hpp.

Referenced by MidpointSplit< BoundType, MatType >::AssignToLeftNode().

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ midpoint_split.hpp

39.513 MinimalCoverageSweep< SplitPolicy > Class Template Reference

The **MinimalCoverageSweep** (p. 2151) class finds a partition along which we can split a node according to the coverage of two resulting nodes.

Classes

- struct **SweepCost**

A struct that provides the type of the sweep cost.

Static Public Member Functions

- template<typename TreeType , typename ElemType >
static bool **CheckLeafSweep** (const TreeType *node, const size_t cutAxis, const ElemType cut)
Check if a leaf node can be split along the axis at the provided coordinate.
- template<typename TreeType , typename ElemType >
static bool **CheckNonLeafSweep** (const TreeType *node, const size_t cutAxis, const ElemType cut)
Check if an intermediate node can be split along the axis at the provided coordinate.
- template<typename TreeType >
static TreeType::ElemType **SweepLeafNode** (const size_t axis, const TreeType *node, typename TreeType::ElemType &axisCut)
Find a suitable partition of a leaf node along the provided axis.
- template<typename TreeType >
static TreeType::ElemType **SweepNonLeafNode** (const size_t axis, const TreeType *node, typename TreeType::ElemType &axisCut)
Find a suitable partition of a non-leaf node along the provided axis.

39.513.1 Detailed Description

```
template<typename SplitPolicy>
class mlpack::tree::MinimalCoverageSweep< SplitPolicy >
```

The **MinimalCoverageSweep** (p. 2151) class finds a partition along which we can split a node according to the coverage of two resulting nodes.

The class finds a partition along a given axis. Moreover, the class evaluates the cost of each split. The cost is proportional to the total coverage of resulting nodes. If the resulting nodes are overflowed the maximum cost is returned.

Template Parameters

<i>SplitPolicy</i>	The class that provides rules for inserting children of a node that is being split into two new subtrees.
--------------------	---

Definition at line 30 of file minimal_coverage_sweep.hpp.

39.513.2 Member Function Documentation

39.513.2.1 CheckLeafSweep()

```
static bool CheckLeafSweep (
    const TreeType * node,
    const size_t cutAxis,
    const ElemType cut ) [static]
```

Check if a leaf node can be split along the axis at the provided coordinate.

Parameters

<i>node</i>	The node that is being split.
<i>cutAxis</i>	The axis that we want to check.
<i>cut</i>	The coordinate that we want to check.

39.513.2.2 CheckNonLeafSweep()

```
static bool CheckNonLeafSweep (
    const TreeType * node,
    const size_t cutAxis,
    const ElemType cut ) [static]
```

Check if an intermediate node can be split along the axis at the provided coordinate.

Parameters

<i>node</i>	The node that is being split.
<i>cutAxis</i>	The axis that we want to check.
<i>cut</i>	The coordinate that we want to check.

39.513.2.3 SweepLeafNode()

```
static TreeType::ElemType SweepLeafNode (
    const size_t axis,
    const TreeType * node,
    typename TreeType::ElemType & axisCut ) [static]
```

Find a suitable partition of a leaf node along the provided axis.

The method returns the cost of the split.

Parameters

<i>axis</i>	The axis along which we are finding a partition.
<i>node</i>	The node that is being split.
<i>axisCut</i>	The coordinate at which the node may be split.

39.513.2.4 SweepNonLeafNode()

```
static TreeType::ElemType SweepNonLeafNode (
    const size_t axis,
    const TreeType * node,
    typename TreeType::ElemType & axisCut ) [static]
```

Find a suitable partition of a non-leaf node along the provided axis.

The method returns the cost of the split.

Parameters

<i>axis</i>	The axis along which we are finding a partition.
<i>node</i>	The node that is being split.
<i>axisCut</i>	The coordinate at which the node may be split.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ **minimal_coverage_sweep.hpp**

39.514 MinimalCoverageSweep< SplitPolicy >::SweepCost< TreeType > Struct Template Reference

A struct that provides the type of the sweep cost.

Public Types

- typedef TreeType::ElemType **type**

39.514.1 Detailed Description

```
template<typename SplitPolicy>
template<typename TreeType>
struct mlpack::tree::MinimalCoverageSweep< SplitPolicy >::SweepCost< TreeType >
```

A struct that provides the type of the sweep cost.

Definition at line 35 of file `minimal_coverage_sweep.hpp`.

39.514.2 Member Typedef Documentation

39.514.2.1 type

```
typedef TreeType::ElemType type
```

Definition at line 37 of file `minimal_coverage_sweep.hpp`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ minimal_coverage_sweep.hpp`

39.515 MinimalSplitsNumberSweep< SplitPolicy > Class Template Reference

The **MinimalSplitsNumberSweep** (p.2155) class finds a partition along which we can split a node according to the number of required splits of the node.

Classes

- struct **SweepCost**
A struct that provides the type of the sweep cost.

Static Public Member Functions

- `template<typename TreeType >`
static `size_t` **SweepLeafNode** (`const size_t axis`, `const TreeType *node`, `typename TreeType::ElemType &axisCut`)
Find a suitable partition of a leaf node along the provided axis.
- `template<typename TreeType >`
static `size_t` **SweepNonLeafNode** (`const size_t axis`, `const TreeType *node`, `typename TreeType::ElemType &axisCut`)
Find a suitable partition of a non-leaf node along the provided axis.

39.515.1 Detailed Description

```
template<typename SplitPolicy>
class mlpack::tree::MinimalSplitsNumberSweep< SplitPolicy >
```

The **MinimalSplitsNumberSweep** (p.2155) class finds a partition along which we can split a node according to the number of required splits of the node.

The class finds a partition along a given axis. Moreover, the class evaluates the cost of each split. The cost is proportional to the number of required splits and the difference of sizes of resulting nodes. If the resulting nodes are overflowed the maximum cost is returned.

Template Parameters

<i>SplitPolicy</i>	The class that provides rules for inserting children of a node that is being split into two new subtrees.
--------------------	---

Definition at line 31 of file `minimal_splits_number_sweep.hpp`.

39.515.2 Member Function Documentation

39.515.2.1 SweepLeafNode()

```
static size_t SweepLeafNode (
    const size_t axis,
    const TreeType * node,
    typename TreeType::ElemType & axisCut ) [static]
```

Find a suitable partition of a leaf node along the provided axis.

The method returns the cost of the split.

Parameters

<i>axis</i>	The axis along which we are finding a partition.
<i>node</i>	The node that is being split.
<i>axisCut</i>	The coordinate at which the node may be split.

39.515.2.2 SweepNonLeafNode()

```
static size_t SweepNonLeafNode (
    const size_t axis,
    const TreeType * node,
    typename TreeType::ElemType & axisCut ) [static]
```

Find a suitable partition of a non-leaf node along the provided axis.

The method returns the cost of the split.

Parameters

<i>axis</i>	The axis along which we are finding a partition.
<i>node</i>	The node that is being split.
<i>axisCut</i>	The coordinate at which the node may be split.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ **minimal_splits_number_sweep.**↵
hpp

39.516 MinimalSplitsNumberSweep< SplitPolicy >::SweepCost< typename > Struct Template Reference

A struct that provides the type of the sweep cost.

Public Types

- typedef size_t **type**

39.516.1 Detailed Description

```
template<typename SplitPolicy>
template<typename>
struct mlpack::tree::MinimalSplitsNumberSweep< SplitPolicy >::SweepCost< typename >
```

A struct that provides the type of the sweep cost.

Definition at line 36 of file minimal_splits_number_sweep.hpp.

39.516.2 Member Typedef Documentation

39.516.2.1 type

```
typedef size_t type
```

Definition at line 38 of file minimal_splits_number_sweep.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ **minimal_splits_number_sweep.**↵
hpp

39.517 MultipleRandomDimensionSelect Class Reference

This dimension selection policy allows the selection from a few random dimensions.

Public Member Functions

- **MultipleRandomDimensionSelect** (const size_t numDimensions=0)
*Instantiate the **MultipleRandomDimensionSelect** (p. 2158) object.*
- size_t **Begin** ()
Get the first random value.
- size_t **Dimensions** () const
Get the number of dimensions.
- size_t & **Dimensions** ()
Set the number of dimensions.
- size_t **End** () const
Get the last random value.
- size_t **Next** ()
Get the next index.

39.517.1 Detailed Description

This dimension selection policy allows the selection from a few random dimensions.

The number of random dimensions to use is specified in the constructor.

Definition at line 23 of file multiple_random_dimension_select.hpp.

39.517.2 Constructor & Destructor Documentation

39.517.2.1 MultipleRandomDimensionSelect()

```
MultipleRandomDimensionSelect (  
    const size_t numDimensions = 0 ) [inline]
```

Instantiate the **MultipleRandomDimensionSelect** (p. 2158) object.

Parameters

<i>numDimensions</i>	Number of random dimensions to try for each split.
----------------------	--

Definition at line 31 of file multiple_random_dimension_select.hpp.

39.517.3 Member Function Documentation

39.517.3.1 Begin()

```
size_t Begin ( ) [inline]
```

Get the first random value.

Definition at line 40 of file `multiple_random_dimension_select.hpp`.

References `mlpack::math::RandInt()`.

39.517.3.2 Dimensions() [1/2]

```
size_t Dimensions ( ) const [inline]
```

Get the number of dimensions.

Definition at line 93 of file `multiple_random_dimension_select.hpp`.

39.517.3.3 Dimensions() [2/2]

```
size_t& Dimensions ( ) [inline]
```

Set the number of dimensions.

Definition at line 95 of file `multiple_random_dimension_select.hpp`.

39.517.3.4 End()

```
size_t End ( ) const [inline]
```

Get the last random value.

Definition at line 82 of file `multiple_random_dimension_select.hpp`.

39.517.3.5 Next()

```
size_t Next ( ) [inline]
```

Get the next index.

Definition at line 87 of file `multiple_random_dimension_select.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/multiple_random_dimension_select.hpp` ↩

39.518 NoAuxiliaryInformation< TreeType > Class Template Reference**Public Member Functions**

- **NoAuxiliaryInformation** ()
Construct the auxiliary information object.
- **NoAuxiliaryInformation** (const TreeType *)
Construct the auxiliary information object.
- **NoAuxiliaryInformation** (const **NoAuxiliaryInformation** &, TreeType *, bool=true)
Construct the auxiliary information object.
- **NoAuxiliaryInformation** (**NoAuxiliaryInformation** &&)
Construct the auxiliary information object.
- bool **HandleNodeInsertion** (TreeType *, TreeType *, bool)
Some tree types require to save some properties at the insertion process.
- bool **HandleNodeRemoval** (TreeType *, const size_t)
Some tree types require to save some properties at the deletion process.
- bool **HandlePointDeletion** (TreeType *, const size_t)
Some tree types require to save some properties at the deletion process.
- bool **HandlePointInsertion** (TreeType *, const size_t)
Some tree types require to save some properties at the insertion process.
- void **NullifyData** ()
Nullify the auxiliary information in order to prevent an invalid free.
- **NoAuxiliaryInformation** & **operator=** (const **NoAuxiliaryInformation** &)
Copy the auxiliary information object.
- template<typename Archive >
void **serialize** (Archive &, const unsigned int)
Serialize the information.
- void **SplitAuxiliaryInfo** (TreeType *, TreeType *, size_t, typename TreeType::ElemType)
The R++ tree requires to split the maximum bounding rectangle of a node that is being split.
- bool **UpdateAuxiliaryInfo** (TreeType *)
Some tree types require to propagate the information upward.

39.518.1 Detailed Description

```
template<typename TreeType>
class mlpack::tree::NoAuxiliaryInformation< TreeType >
```

Definition at line 20 of file no_auxiliary_information.hpp.

39.518.2 Constructor & Destructor Documentation

39.518.2.1 NoAuxiliaryInformation() [1/4]

```
NoAuxiliaryInformation ( ) [inline]
```

Construct the auxiliary information object.

Definition at line 24 of file no_auxiliary_information.hpp.

39.518.2.2 NoAuxiliaryInformation() [2/4]

```
NoAuxiliaryInformation (
    const TreeType * ) [inline]
```

Construct the auxiliary information object.

Definition at line 26 of file no_auxiliary_information.hpp.

39.518.2.3 NoAuxiliaryInformation() [3/4]

```
NoAuxiliaryInformation (
    const NoAuxiliaryInformation< TreeType > & ,
    TreeType * ,
    bool = true ) [inline]
```

Construct the auxiliary information object.

Definition at line 28 of file no_auxiliary_information.hpp.

39.518.2.4 NoAuxiliaryInformation() [4/4]

```
NoAuxiliaryInformation (
    NoAuxiliaryInformation< TreeType > && ) [inline]
```

Construct the auxiliary information object.

Definition at line 32 of file no_auxiliary_information.hpp.

39.518.3 Member Function Documentation

39.518.3.1 HandleNodeInsertion()

```
bool HandleNodeInsertion (
    TreeType * ,
    TreeType * ,
    bool ) [inline]
```

Some tree types require to save some properties at the insertion process.

This method allows the auxiliary information the option of manipulating the tree in order to perform the insertion process. If the auxiliary information does that, then the method should return true; if the method returns false the **RectangleTree** (p. 2207) performs its default behavior.

Parameters

<i>node</i>	The node in which the nodeToInsert is being inserted.
<i>nodeToInsert</i>	The node being inserted.
<i>insertionLevel</i>	The level of the tree at which the nodeToInsert should be inserted.

Definition at line 67 of file no_auxiliary_information.hpp.

39.518.3.2 HandleNodeRemoval()

```
bool HandleNodeRemoval (
    TreeType * ,
    const size_t ) [inline]
```

Some tree types require to save some properties at the deletion process.

This method allows the auxiliary information the option of manipulating the tree in order to perform the deletion process. If the auxiliary information does that, then the method should return true; if the method returns false the **RectangleTree** (p. 2207) performs its default behavior.

Parameters

<i>node</i>	The node from which the node is being deleted.
<i>nodeIndex</i>	The local index of the node being deleted.

Definition at line 99 of file no_auxiliary_information.hpp.

39.518.3.3 HandlePointDeletion()

```
bool HandlePointDeletion (
    TreeType * ,
    const size_t ) [inline]
```

Some tree types require to save some properties at the deletion process.

This method allows the auxiliary information the option of manipulating the tree in order to perform the deletion process. If the auxiliary information does that, then the method should return true; if the method returns false the **RectangleTree** (p. 2207) performs its default behavior.

Parameters

<i>node</i>	The node from which the point is being deleted.
<i>localIndex</i>	The local index of the point being deleted.

Definition at line 84 of file no_auxiliary_information.hpp.

39.518.3.4 HandlePointInsertion()

```
bool HandlePointInsertion (
    TreeType * ,
    const size_t ) [inline]
```

Some tree types require to save some properties at the insertion process.

This method allows the auxiliary information the option of manipulating the tree in order to perform the insertion process. If the auxiliary information does that, then the method should return true; if the method returns false the **RectangleTree** (p. 2207) performs its default behavior.

Parameters

<i>node</i>	The node in which the point is being inserted.
<i>point</i>	The global number of the point being inserted.

Definition at line 50 of file no_auxiliary_information.hpp.

39.518.3.5 NullifyData()

```
void NullifyData ( ) [inline]
```

Nullify the auxiliary information in order to prevent an invalid free.

Definition at line 137 of file no_auxiliary_information.hpp.

39.518.3.6 operator=()

```
NoAuxiliaryInformation& operator= (
    const NoAuxiliaryInformation< TreeType > & ) [inline]
```

Copy the auxiliary information object.

Definition at line 35 of file no_auxiliary_information.hpp.

39.518.3.7 serialize()

```
void serialize (
    Archive & ,
    const unsigned int ) [inline]
```

Serialize the information.

Definition at line 144 of file no_auxiliary_information.hpp.

39.518.3.8 SplitAuxiliaryInfo()

```
void SplitAuxiliaryInfo (
    TreeType * ,
    TreeType * ,
    size_t ,
    typename TreeType::ElemType ) [inline]
```

The R++ tree requires to split the maximum bounding rectangle of a node that is being split.

This method is intended for that. This method is only necessary for an AuxiliaryInformationType that is being used in conjunction with **RPlusTreeSplit** (p.2251).

Parameters

<i>treeOne</i>	The first subtree.
<i>treeTwo</i>	The second subtree.
<i>axis</i>	The axis along which the split is performed.
<i>cut</i>	The coordinate at which the node is split.

Definition at line 127 of file no_auxiliary_information.hpp.

39.518.3.9 UpdateAuxiliaryInfo()

```
bool UpdateAuxiliaryInfo (
    TreeType * ) [inline]
```

Some tree types require to propagate the information upward.

This method should return false if this is not the case. If true is returned, the update will be propagated upward.

Parameters

<i>node</i>	The node in which the auxiliary information being update.
-------------	---

Definition at line 111 of file no_auxiliary_information.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ no_auxiliary_information.hpp

39.519 NumericSplitInfo< ObservationType > Class Template Reference

Public Member Functions

- **NumericSplitInfo** ()
- **NumericSplitInfo** (const arma::Col< ObservationType > &splitPoints)
- template<typename eT >
size_t **CalculateDirection** (const eT &value) const
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the split (save/load the split points).

39.519.1 Detailed Description

```
template<typename ObservationType = double>
class mpack::tree::NumericSplitInfo< ObservationType >
```

Definition at line 21 of file `numeric_split_info.hpp`.

39.519.2 Constructor & Destructor Documentation

39.519.2.1 NumericSplitInfo() [1/2]

```
NumericSplitInfo ( ) [inline]
```

Definition at line 24 of file `numeric_split_info.hpp`.

39.519.2.2 NumericSplitInfo() [2/2]

```
NumericSplitInfo (
    const arma::Col< ObservationType > & splitPoints ) [inline]
```

Definition at line 25 of file `numeric_split_info.hpp`.

39.519.3 Member Function Documentation

39.519.3.1 CalculateDirection()

```
size_t CalculateDirection (
    const eT & value ) const [inline]
```

Definition at line 29 of file `numeric_split_info.hpp`.

39.519.3.2 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the split (save/load the split points).

Definition at line 41 of file numeric_split_info.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/ **numeric_split_info.hpp**

39.520 Octree< MetricType, StatisticType, MatType > Class Template Reference

Classes

- class **DualTreeTraverser**
A dual-tree traverser; see dual_tree_traverser.hpp.
- class **SingleTreeTraverser**
A single-tree traverser; see single_tree_traverser.hpp.

Public Types

- typedef MatType::elem_type **ElemType**
The type of element held in MatType.
- typedef MatType **Mat**
So other classes can use TreeType::Mat.

Public Member Functions

- **Octree** (const MatType &data, const size_t maxLeafSize=20)
Construct this as the root node of an octree on the given dataset.
- **Octree** (const MatType &data, std::vector< size_t > &oldFromNew, const size_t maxLeafSize=20)
Construct this as the root node of an octree on the given dataset.
- **Octree** (const MatType &data, std::vector< size_t > &oldFromNew, std::vector< size_t > &newFromOld, const size_t maxLeafSize=20)
Construct this as the root node of an octree on the given dataset.
- **Octree** (MatType &&data, const size_t maxLeafSize=20)
Construct this as the root node of an octree on the given dataset.
- **Octree** (MatType &&data, std::vector< size_t > &oldFromNew, const size_t maxLeafSize=20)
Construct this as the root node of an octree on the given dataset.
- **Octree** (MatType &&data, std::vector< size_t > &oldFromNew, std::vector< size_t > &newFromOld, const size_t maxLeafSize=20)

Construct this as the root node of an octree on the given dataset.

- **Octree** (**Octree** *parent, const size_t begin, const size_t count, const arma::vec ¢er, const double width, const size_t maxLeafSize=20)

Construct this node as a child of the given parent, starting at column begin and using count points.

- **Octree** (**Octree** *parent, const size_t begin, const size_t count, std::vector< size_t > &oldFromNew, const arma::vec ¢er, const double width, const size_t maxLeafSize=20)

Construct this node as a child of the given parent, starting at column begin and using count points.

- **Octree** (const **Octree** &other)

Copy the given tree.

- **Octree** (**Octree** &&other)

Move the given tree.

- template<typename Archive >

Octree (Archive &ar, const typename **std::enable_if_t**< Archive::is_loading::value > *=0)

*Initialize the tree from a **boost::serialization** (p. 251) archive.*

- ~**Octree** ()

Destroy the tree.

- const **bound::HRectBound**< MetricType > & **Bound** () const

Return the bound object for this node.

- **bound::HRectBound**< MetricType > & **Bound** ()

Modify the bound object for this node.

- void **Center** (arma::vec ¢er) const

Store the center of the bounding region in the given vector.

- const **Octree** & **Child** (const size_t child) const

Return the specified child.

- **Octree** & **Child** (const size_t child)

Return the specified child.

- **Octree** *& **ChildPtr** (const size_t child)

Return the pointer to the given child.

- const MatType & **Dataset** () const

Return the dataset used by this node.

- size_t **Descendant** (const size_t index) const

Return the index (with reference to the dataset) of a particular descendant.

- **ElemType** **FurthestDescendantDistance** () const

Return the furthest possible descendant distance.

- **ElemType** **FurthestPointDistance** () const

Return the furthest distance to a point held in this node.

- template<typename VecType >

size_t **GetFurthestChild** (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > *=0) const

Return the index of the furthest child node to the given query point.

- size_t **GetFurthestChild** (const **Octree** &queryNode) const

Return the index of the furthest child node to the given query node.

- template<typename VecType >

size_t **GetNearestChild** (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > *=0) const

Return the index of the nearest child node to the given query point.

- size_t **GetNearestChild** (const **Octree** &queryNode) const

Return the index of the nearest child node to the given query node.

- bool **IsLeaf** () const
Return whether or not the node is a leaf.
- **ElemType MaxDistance** (const **Octree** &other) const
Return the maximum distance to another node.
- template<typename VecType >
ElemType MaxDistance (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > !=0) const
Return the maximum distance to the given point.
- MetricType **Metric** () const
Return the metric that this tree uses.
- **ElemType MinDistance** (const **Octree** &other) const
Return the minimum distance to another node.
- template<typename VecType >
ElemType MinDistance (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > !=0) const
Return the minimum distance to the given point.
- **ElemType MinimumBoundDistance** () const
Return the minimum distance from the center of the node to any bound edge.
- size_t **NumChildren** () const
Return the number of children in this node.
- size_t **NumDescendants** () const
Return the number of descendants of this node.
- size_t **NumPoints** () const
Return the number of points in this node (0 if not a leaf).
- **Octree * Parent** () const
Get the pointer to the parent.
- **Octree *& Parent** ()
Modify the pointer to the parent (be careful!).
- **ElemType ParentDistance** () const
Return the distance from the center of this node to the center of the parent node.
- **ElemType & ParentDistance** ()
Modify the distance from the center of this node to the center of the parent node.
- size_t **Point** (const size_t index) const
Return the index (with reference to the dataset) of a particular point in this node.
- **math::RangeType**< **ElemType** > **RangeDistance** (const **Octree** &other) const
Return the minimum and maximum distance to another node.
- template<typename VecType >
math::RangeType< **ElemType** > **RangeDistance** (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > !=0) const
Return the minimum and maximum distance to another node.
- template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
Serialize the tree.
- const StatisticType & **Stat** () const
Return the statistic object for this node.
- StatisticType & **Stat** ()
Modify the statistic object for this node.

Protected Member Functions

- **Octree ()**

A default constructor.

39.520.1 Detailed Description

```
template<typename MetricType = metric::EuclideanDistance, typename StatisticType = EmptyStatistic, typename MatType = arma<↵
::mat>
class mlpack::tree::Octree< MetricType, StatisticType, MatType >
```

Definition at line 25 of file octree.hpp.

39.520.2 Member Typedef Documentation

39.520.2.1 ElemType

```
typedef MatType::elem_type ElemType
```

The type of element held in MatType.

Definition at line 31 of file octree.hpp.

39.520.2.2 Mat

```
typedef MatType Mat
```

So other classes can use TreeType::Mat.

Definition at line 29 of file octree.hpp.

39.520.3 Constructor & Destructor Documentation

39.520.3.1 Octree() [1/12]

```
Octree (
    const MatType & data,
    const size_t maxLeafSize = 20 )
```

Construct this as the root node of an octree on the given dataset.

This copies the dataset. If you don't want to copy the input dataset, consider using the constructor that takes an rvalue reference and use `std::move()`.

Parameters

<i>data</i>	Dataset to create tree from. This will be copied!
<i>maxLeafSize</i>	Maximum number of points in a leaf node.

39.520.3.2 Octree() [2/12]

```
Octree (
    const MatType & data,
    std::vector< size_t > & oldFromNew,
    const size_t maxLeafSize = 20 )
```

Construct this as the root node of an octree on the given dataset.

This copies the dataset and modifies its ordering; a mapping of the old point indices to the new point indices is filled. If you don't want the matrix to be copied, consider using the constructor that takes an rvalue reference and use `std::move()`.

Parameters

<i>data</i>	Dataset to create tree from. This will be copied!
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.
<i>maxLeafSize</i>	Maximum number of points in a leaf node.

39.520.3.3 Octree() [3/12]

```
Octree (
    const MatType & data,
    std::vector< size_t > & oldFromNew,
    std::vector< size_t > & newFromOld,
    const size_t maxLeafSize = 20 )
```

Construct this as the root node of an octree on the given dataset.

This copies the dataset and modifies its ordering; a mapping of the old point indices to the new point indices is filled, and a mapping of the new point indices to the old point indices is filled. If you don't want the matrix to be copied, consider using the constructor that takes an rvalue reference and use `std::move()`.

Parameters

<i>data</i>	Dataset to create tree from. This will be copied!
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.
<i>newFromOld</i>	Vector which will be filled with the new positions for each old point.
<i>maxLeafSize</i>	Maximum number of points in a leaf node.

39.520.3.4 Octree() [4/12]

```
Octree (
    MatType && data,
    const size_t maxLeafSize = 20 )
```

Construct this as the root node of an octree on the given dataset.

This will take ownership of the dataset; if you don't want this, consider using the constructor that takes a const reference to the dataset.

Parameters

<i>data</i>	Dataset to create tree from. This will be copied!
<i>maxLeafSize</i>	Maximum number of points in a leaf node.

39.520.3.5 Octree() [5/12]

```
Octree (
    MatType && data,
    std::vector< size_t > & oldFromNew,
    const size_t maxLeafSize = 20 )
```

Construct this as the root node of an octree on the given dataset.

This will take ownership of the dataset; if you don't want this, consider using the constructor that takes a const reference to the dataset. This modifies the ordering of the dataset; a mapping of the old point indices to the new point indices is filled.

Parameters

<i>data</i>	Dataset to create tree from. This will be copied!
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.
<i>maxLeafSize</i>	Maximum number of points in a leaf node.

39.520.3.6 Octree() [6/12]

```
Octree (
    MatType && data,
```

```
std::vector< size_t > & oldFromNew,
std::vector< size_t > & newFromOld,
const size_t maxLeafSize = 20 )
```

Construct this as the root node of an octree on the given dataset.

This will take ownership of the dataset; if you don't want this, consider using the constructor that takes a const reference to the dataset. This modifies the ordering of the dataset; a mapping of the old point indices to the new point indices is filled, and a mapping of the new point indices to the old point indices is filled.

Parameters

<i>data</i>	Dataset to create tree from. This will be copied!
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.
<i>newFromOld</i>	Vector which will be filled with the new positions for each old point.
<i>maxLeafSize</i>	Maximum number of points in a leaf node.

39.520.3.7 Octree() [7/12]

```
Octree (
    Octree< MetricType, StatisticType, MatType > * parent,
    const size_t begin,
    const size_t count,
    const arma::vec & center,
    const double width,
    const size_t maxLeafSize = 20 )
```

Construct this node as a child of the given parent, starting at column begin and using count points.

The ordering of that subset of points in the parent's data matrix will be modified! This is used for recursive tree-building by the other constructors that don't specify point indices.

Parameters

<i>parent</i>	Parent of this node. Its dataset will be modified!
<i>begin</i>	Index of point to start tree construction with.
<i>count</i>	Number of points to use to construct tree.
<i>center</i>	Center of the node (for splitting).
<i>width</i>	Width of the node in each dimension.
<i>maxLeafSize</i>	Maximum number of points in a leaf node.

39.520.3.8 Octree() [8/12]

```
Octree (
```

```

    Octree< MetricType, StatisticType, MatType > * parent,
    const size_t begin,
    const size_t count,
    std::vector< size_t > & oldFromNew,
    const arma::vec & center,
    const double width,
    const size_t maxLeafSize = 20 )

```

Construct this node as a child of the given parent, starting at column *begin* and using *count* points.

The ordering of that subset of points in the parent's data matrix will be modified! This is used for recursive tree-building by the other constructors that don't specify point indices.

A mapping of the old point indices to the new point indices is filled, but it is expected that the vector is already allocated with size greater than or equal to (*begin* + *count*), and if that is not true, invalid memory reads (and writes) will occur.

Parameters

<i>parent</i>	Parent of this node. Its dataset will be modified!
<i>begin</i>	Index of point to start tree construction with.
<i>count</i>	Number of points to use to construct tree.
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.
<i>center</i>	Center of the node (for splitting).
<i>width</i>	Width of the node in each dimension.
<i>maxLeafSize</i>	Maximum number of points in a leaf node.

39.520.3.9 Octree() [9/12]

```

Octree (
    const Octree< MetricType, StatisticType, MatType > & other )

```

Copy the given tree.

Be careful! This may use a lot of memory.

Parameters

<i>other</i>	Tree to copy from.
--------------	--------------------

39.520.3.10 Octree() [10/12]

```

Octree (
    Octree< MetricType, StatisticType, MatType > && other )

```

Move the given tree.

The tree passed as a parameter will be emptied and will not be usable after this call.

Parameters

<i>other</i>	Tree to move.
--------------	---------------

39.520.3.11 Octree() [11/12]

```
Octree (
    Archive & ar,
    const typename std::enable_if_t< Archive::is_loading::value > * = 0 )
```

Initialize the tree from a **boost::serialization** (p. 251) archive.

Parameters

<i>ar</i>	Archive to load tree from. Must be an iarchive, not an oarchive.
-----------	--

39.520.3.12 ~Octree()

```
~ Octree ( )
```

Destroy the tree.

39.520.3.13 Octree() [12/12]

```
Octree ( ) [protected]
```

A default constructor.

This is meant to only be used with **boost::serialization** (p. 251), which is allowed with the friend declaration below. This does not return a valid tree! The method must be protected, so that the serialization shim can work with the default constructor.

Referenced by Octree< MetricType, StatisticType, MatType >::Center().

39.520.4 Member Function Documentation

39.520.4.1 Bound() [1/2]

```
const bound::HRectBound<MetricType>& Bound ( ) const [inline]
```

Return the bound object for this node.

Definition at line 247 of file octree.hpp.

39.520.4.2 Bound() [2/2]

```
bound::HRectBound<MetricType>& Bound ( ) [inline]
```

Modify the bound object for this node.

Definition at line 249 of file octree.hpp.

39.520.4.3 Center()

```
void Center (
    arma::vec & center ) const [inline]
```

Store the center of the bounding region in the given vector.

Definition at line 383 of file octree.hpp.

References HRectBound< MetricType, ElemType >::Center(), Octree< MetricType, StatisticType, MatType >::Octree(), and Octree< MetricType, StatisticType, MatType >::serialize().

39.520.4.4 Child() [1/2]

```
const Octree& Child (
    const size_t child ) const [inline]
```

Return the specified child.

If the index is out of bounds, unspecified behavior will occur.

Definition at line 326 of file octree.hpp.

39.520.4.5 Child() [2/2]

```
Octree& Child (
    const size_t child ) [inline]
```

Return the specified child.

If the index is out of bounds, unspecified behavior will occur.

Definition at line 332 of file octree.hpp.

39.520.4.6 ChildPtr()

```
Octree* ChildPtr (
    const size_t child ) [inline]
```

Return the pointer to the given child.

This allows the child itself to be modified.

Definition at line 338 of file octree.hpp.

References Octree< MetricType, StatisticType, MatType >::Descendant(), Octree< MetricType, StatisticType, MatType >::MaxDistance(), Octree< MetricType, StatisticType, MatType >::MinDistance(), Octree< MetricType, StatisticType, MatType >::NumDescendants(), Octree< MetricType, StatisticType, MatType >::NumPoints(), Octree< MetricType, StatisticType, MatType >::Point(), and Octree< MetricType, StatisticType, MatType >::RangeDistance().

39.520.4.7 Dataset()

```
const MatType& Dataset ( ) const [inline]
```

Return the dataset used by this node.

Definition at line 239 of file octree.hpp.

39.520.4.8 Descendant()

```
size_t Descendant (
    const size_t index ) const
```

Return the index (with reference to the dataset) of a particular descendant.

Referenced by Octree< MetricType, StatisticType, MatType >::ChildPtr().

39.520.4.9 FurthestDescendantDistance()

```
ElemType FurthestDescendantDistance ( ) const
```

Return the furthest possible descendant distance.

This returns the maximum distance from the centroid to the edge of the bound and not the empirical quantity which is the actual furthest descendant distance. So the actual furthest descendant distance may be less than what this method returns (but it will never be greater than this).

Referenced by Octree< MetricType, StatisticType, MatType >::IsLeaf().

39.520.4.10 FurthestPointDistance()

```
ElemType FurthestPointDistance ( ) const
```

Return the furthest distance to a point held in this node.

If this is not a leaf node, then the distance is 0 because the node holds no points.

Referenced by Octree< MetricType, StatisticType, MatType >::IsLeaf().

39.520.4.11 GetFurthestChild() [1/2]

```
size_t GetFurthestChild (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const
```

Return the index of the furthest child node to the given query point.

If this is a leaf node, it will return **NumChildren()** (p. 2181) (invalid index).

Referenced by Octree< MetricType, StatisticType, MatType >::IsLeaf(), and Octree< MetricType, StatisticType, MatType >::Metric().

39.520.4.12 GetFurthestChild() [2/2]

```
size_t GetFurthestChild (
    const Octree< MetricType, StatisticType, MatType > & queryNode ) const
```

Return the index of the furthest child node to the given query node.

If it can't decide, it will return **NumChildren()** (p. 2181) (invalid index).

39.520.4.13 GetNearestChild() [1/2]

```
size_t GetNearestChild (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const
```

Return the index of the nearest child node to the given query point.

If this is a leaf node, it will return **NumChildren()** (p. 2181) (invalid index).

Referenced by Octree< MetricType, StatisticType, MatType >::IsLeaf(), and Octree< MetricType, StatisticType, MatType >::Metric().

39.520.4.14 GetNearestChild() [2/2]

```
size_t GetNearestChild (
    const Octree< MetricType, StatisticType, MatType > & queryNode ) const
```

Return the index of the nearest child node to the given query node.

If it can't decide, it will return **NumChildren()** (p. 2181) (invalid index).

39.520.4.15 IsLeaf()

```
bool IsLeaf ( ) const [inline]
```

Return whether or not the node is a leaf.

Definition at line 283 of file octree.hpp.

References Octree< MetricType, StatisticType, MatType >::FurthestDescendantDistance(), Octree< MetricType, StatisticType, MatType >::FurthestPointDistance(), Octree< MetricType, StatisticType, MatType >::GetFurthestChild(), Octree< MetricType, StatisticType, MatType >::GetNearestChild(), Octree< MetricType, StatisticType, MatType >::MinimumBoundDistance(), and Octree< MetricType, StatisticType, MatType >::NumChildren().

39.520.4.16 MaxDistance() [1/2]

```
ElemType MaxDistance (
    const Octree< MetricType, StatisticType, MatType > & other ) const
```

Return the maximum distance to another node.

Referenced by Octree< MetricType, StatisticType, MatType >::ChildPtr().

39.520.4.17 MaxDistance() [2/2]

```
ElemType MaxDistance (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const
```

Return the maximum distance to the given point.

39.520.4.18 Metric()

```
MetricType Metric ( ) const [inline]
```

Return the metric that this tree uses.

Definition at line 260 of file octree.hpp.

References Octree< MetricType, StatisticType, MatType >::GetFurthestChild(), and Octree< MetricType, StatisticType, MatType >::GetNearestChild().

39.520.4.19 MinDistance() [1/2]

```
ElemType MinDistance (
    const Octree< MetricType, StatisticType, MatType > & other ) const
```

Return the minimum distance to another node.

Referenced by Octree< MetricType, StatisticType, MatType >::ChildPtr().

39.520.4.20 MinDistance() [2/2]

```
ElemType MinDistance (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const
```

Return the minimum distance to the given point.

39.520.4.21 MinimumBoundDistance()

```
ElemType MinimumBoundDistance ( ) const
```

Return the minimum distance from the center of the node to any bound edge.

Referenced by Octree< MetricType, StatisticType, MatType >::IsLeaf().

39.520.4.22 NumChildren()

```
size_t NumChildren ( ) const
```

Return the number of children in this node.

Referenced by Octree< MetricType, StatisticType, MatType >::IsLeaf(), and Octree< MetricType, StatisticType, MatType >::Stat().

39.520.4.23 NumDescendants()

```
size_t NumDescendants ( ) const
```

Return the number of descendants of this node.

Referenced by Octree< MetricType, StatisticType, MatType >::ChildPtr().

39.520.4.24 NumPoints()

```
size_t NumPoints ( ) const
```

Return the number of points in this node (0 if not a leaf).

Referenced by Octree< MetricType, StatisticType, MatType >::ChildPtr().

39.520.4.25 Parent() [1/2]

```
Octree* Parent ( ) const [inline]
```

Get the pointer to the parent.

Definition at line 242 of file octree.hpp.

39.520.4.26 Parent() [2/2]

```
Octree*& Parent ( ) [inline]
```

Modify the pointer to the parent (be careful!).

Definition at line 244 of file octree.hpp.

39.520.4.27 ParentDistance() [1/2]

```
ElementType ParentDistance ( ) const [inline]
```

Return the distance from the center of this node to the center of the parent node.

Definition at line 317 of file octree.hpp.

39.520.4.28 ParentDistance() [2/2]

```
ElementType& ParentDistance ( ) [inline]
```

Modify the distance from the center of this node to the center of the parent node.

Definition at line 320 of file octree.hpp.

39.520.4.29 Point()

```
size_t Point (
    const size_t index ) const
```

Return the index (with reference to the dataset) of a particular point in this node.

If the given index is invalid (i.e. if it is greater than **NumPoints()** (p. 2181)), the indices returned will be invalid.

Referenced by Octree< MetricType, StatisticType, MatType >::ChildPtr().

39.520.4.30 RangeDistance() [1/2]

```
math::RangeType< ElemType> RangeDistance (
    const Octree< MetricType, StatisticType, MatType > & other ) const
```

Return the minimum and maximum distance to another node.

Referenced by Octree< MetricType, StatisticType, MatType >::ChildPtr().

39.520.4.31 RangeDistance() [2/2]

```
math::RangeType< ElemType> RangeDistance (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const
```

Return the minimum and maximum distance to another node.

39.520.4.32 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the tree.

Referenced by Octree< MetricType, StatisticType, MatType >::Center().

39.520.4.33 Stat() [1/2]

```
const StatisticType& Stat ( ) const [inline]
```

Return the statistic object for this node.

Definition at line 252 of file octree.hpp.

39.520.4.34 Stat() [2/2]

```
StatisticType& Stat ( ) [inline]
```

Modify the statistic object for this node.

Definition at line 254 of file octree.hpp.

References Octree< MetricType, StatisticType, MatType >::NumChildren().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree/ **octree.hpp**

39.521 Octree< MetricType, StatisticType, MatType >::DualTreeTraverser< MetricType, StatisticType, MatType > Class Template Reference

A dual-tree traverser; see dual_tree_traverser.hpp.

Public Member Functions

- **DualTreeTraverser** (RuleType &rule)
Instantiate the given dual-tree traverser with the given rule set.
- size_t **NumBaseCases** () const
Get the number of times a base case was computed.
- size_t & **NumBaseCases** ()
Modify the number of times a base case was computed.
- size_t **NumPrunes** () const
Get the number of pruned nodes.
- size_t & **NumPrunes** ()
Modify the number of pruned nodes (i.e. to reset it).
- size_t **NumScores** () const
Get the number of times a node was scored.
- size_t & **NumScores** ()
Modify the number of times a node was scored.
- size_t **NumVisited** () const
Get the number of visited node combinations.
- size_t & **NumVisted** ()
Modify the number of visited node combinations.
- void **Traverse** (Octree &queryNode, Octree &referenceNode)
Traverse the two trees.

39.521.1 Detailed Description

```
template<typename MetricType = metric::EuclideanDistance, typename StatisticType = EmptyStatistic, typename MatType = arma<
::mat>
template<typename MetricType, typename StatisticType, typename MatType>
class mlpack::tree::Octree< MetricType, StatisticType, MatType >::DualTreeTraverser< MetricType, StatisticType, MatType >
```

A dual-tree traverser; see dual_tree_traverser.hpp.

Definition at line 25 of file dual_tree_traverser.hpp.

39.521.2 Constructor & Destructor Documentation

39.521.2.1 DualTreeTraverser()

```
DualTreeTraverser (
    RuleType & rule )
```

Instantiate the given dual-tree traverser with the given rule set.

39.521.3 Member Function Documentation

39.521.3.1 NumBaseCases() [1/2]

```
size_t NumBaseCases ( ) const [inline]
```

Get the number of times a base case was computed.

Definition at line 55 of file dual_tree_traverser.hpp.

39.521.3.2 NumBaseCases() [2/2]

```
size_t& NumBaseCases ( ) [inline]
```

Modify the number of times a base case was computed.

Definition at line 57 of file dual_tree_traverser.hpp.

39.521.3.3 NumPrunes() [1/2]

```
size_t NumPrunes ( ) const [inline]
```

Get the number of pruned nodes.

Definition at line 40 of file dual_tree_traverser.hpp.

39.521.3.4 NumPrunes() [2/2]

```
size_t& NumPrunes ( ) [inline]
```

Modify the number of pruned nodes (i.e. to reset it).

Definition at line 42 of file dual_tree_traverser.hpp.

39.521.3.5 NumScores() [1/2]

```
size_t NumScores ( ) const [inline]
```

Get the number of times a node was scored.

Definition at line 50 of file dual_tree_traverser.hpp.

39.521.3.6 NumScores() [2/2]

```
size_t& NumScores ( ) [inline]
```

Modify the number of times a node was scored.

Definition at line 52 of file dual_tree_traverser.hpp.

39.521.3.7 NumVisited()

```
size_t NumVisited ( ) const [inline]
```

Get the number of visited node combinations.

Definition at line 45 of file dual_tree_traverser.hpp.

39.521.3.8 NumVisted()

```
size_t& NumVisted ( ) [inline]
```

Modify the number of visited node combinations.

Definition at line 47 of file dual_tree_traverser.hpp.

39.521.3.9 Traverse()

```
void Traverse (
    Octree & queryNode,
    Octree & referenceNode )
```

Traverse the two trees.

This does not reset the statistics of the traversals (it just adds to them).

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree/ **dual_tree_traverser.hpp**

39.522 Octree< MetricType, StatisticType, MatType >::SingleTreeTraverser< RuleType > Class Template Reference

A single-tree traverser; see single_tree_traverser.hpp.

Public Member Functions

- **SingleTreeTraverser** (RuleType &rule)
Instantiate the traverser with the given rule set.
- size_t **NumPrunes** () const
Get the number of pruned nodes.
- size_t & **NumPrunes** ()
Modify the number of pruned nodes.
- void **Traverse** (const size_t queryIndex, **Octree** &referenceNode)
Traverse the reference tree with the given query point.

39.522.1 Detailed Description

```
template<typename MetricType = metric::EuclideanDistance, typename StatisticType = EmptyStatistic, typename MatType = arma<
::mat>
template<typename RuleType>
class mlpack::tree::Octree< MetricType, StatisticType, MatType >::SingleTreeTraverser< RuleType >
```

A single-tree traverser; see `single_tree_traverser.hpp`.

Definition at line 35 of file `octree.hpp`.

39.522.2 Constructor & Destructor Documentation

39.522.2.1 SingleTreeTraverser()

```
SingleTreeTraverser (
    RuleType & rule )
```

Instantiate the traverser with the given rule set.

39.522.3 Member Function Documentation

39.522.3.1 NumPrunes() [1/2]

```
size_t NumPrunes ( ) const [inline]
```

Get the number of pruned nodes.

Definition at line 41 of file `single_tree_traverser.hpp`.

39.522.3.2 NumPrunes() [2/2]

```
size_t& NumPrunes ( ) [inline]
```

Modify the number of pruned nodes.

Definition at line 43 of file `single_tree_traverser.hpp`.

39.522.3.3 Traverse()

```
void Traverse (
    const size_t queryIndex,
    Octree & referenceNode )
```

Traverse the reference tree with the given query point.

This does not reset the number of pruned nodes.

Parameters

<i>queryIndex</i>	Index of query point.
<i>referenceNode</i>	Node in reference tree.

The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree/ **octree.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree/ **single_tree_traverser.hpp**

39.523 Octree< MetricType, StatisticType, MatType >::SplitType::SplitInfo Struct Reference

Public Member Functions

- **SplitInfo** (const size_t **d**, const arma::vec &c)
*Create the **SplitInfo** (p. 2189) object.*

Public Attributes

- const arma::vec & **center**
The center of the node.
- size_t **d**
The dimension we are splitting on.

39.523.1 Detailed Description

```
template<typename MetricType = metric::EuclideanDistance, typename StatisticType = EmptyStatistic, typename MatType = arma::mat>
struct mlpack::tree::Octree< MetricType, StatisticType, MatType >::SplitType::SplitInfo
```

Definition at line 433 of file octree.hpp.

39.523.2 Constructor & Destructor Documentation

39.523.2.1 SplitInfo()

```
SplitInfo (
    const size_t d,
    const arma::vec & c ) [inline]
```

Create the **SplitInfo** (p. 2189) object.

Definition at line 436 of file octree.hpp.

39.523.3 Member Data Documentation

39.523.3.1 center

```
const arma::vec& center
```

The center of the node.

Definition at line 441 of file octree.hpp.

39.523.3.2 d

```
size_t d
```

The dimension we are splitting on.

Definition at line 439 of file octree.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree/ **octree.hpp**

39.524 ProjVector Class Reference

ProjVector (p. 2190) defines a general projection vector (not necessarily axis-parallel).

Public Member Functions

- **ProjVector** ()
Empty Constructor.
 - **ProjVector** (const arma::vec &vect)
Create the projection vector based on the specified vector.
 - template<typename VecType >
double **Project** (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > * = 0) const
Project the given point on the projection vector.
 - template<typename MetricType , typename VecType >
math::RangeType< typename VecType::elem_type > **Project** (const **bound::BallBound**< MetricType, VecType > &bound) const
Project the given ball bound on the projection vector.
 - template<typename Archive >
void **serialize** (Archive &ar, const unsigned int)
- Serialization.*

39.524.1 Detailed Description

ProjVector (p. 2190) defines a general projection vector (not necessarily axis-parallel).

Definition at line 91 of file projection_vector.hpp.

39.524.2 Constructor & Destructor Documentation

39.524.2.1 ProjVector() [1/2]

```
ProjVector ( ) [inline]
```

Empty Constructor.

Definition at line 100 of file projection_vector.hpp.

39.524.2.2 ProjVector() [2/2]

```
ProjVector (
    const arma::vec & vect ) [inline]
```

Create the projection vector based on the specified vector.

Parameters

<i>vect</i>	Vector to be considered.
-------------	--------------------------

Definition at line 109 of file projection_vector.hpp.

39.524.3 Member Function Documentation

39.524.3.1 Project() [1/2]

```
double Project (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const [inline]
```

Project the given point on the projection vector.

Parameters

<i>point</i>	Point to be projected.
--------------	------------------------

Definition at line 119 of file `projection_vector.hpp`.

39.524.3.2 `Project()` [2/2]

```
math::RangeType<typename VecType::elem_type> Project (
    const    bound::BallBound< MetricType, VecType > & bound ) const    [inline]
```

Project the given ball bound on the projection vector.

Parameters

<i>bound</i>	Bound to be projected.
--------------	------------------------

Returns

Range of projected values.

Definition at line 132 of file `projection_vector.hpp`.

References `BallBound< MetricType, VecType >::Center()`, `AxisParallelProjVector::Project()`, and `BallBound< MetricType, VecType >::Radius()`.

39.524.3.3 `serialize()`

```
void serialize (
    Archive & ar,
    const unsigned int )    [inline]
```

Serialization.

Definition at line 145 of file `projection_vector.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/ projection_vector.hpp`

39.525 QueueFrame< TreeType, TraversalInfoType > Struct Template Reference

Public Attributes

- `size_t` **queryDepth**
- `TreeType *` **queryNode**
- `TreeType *` **referenceNode**
- `double` **score**
- `TraversalInfoType` **traversalInfo**

39.525.1 Detailed Description

```
template<typename TreeType, typename TraversalInfoType>  
struct mpack::tree::QueueFrame< TreeType, TraversalInfoType >
```

Definition at line 27 of file `breadth_first_dual_tree_traverser.hpp`.

39.525.2 Member Data Documentation

39.525.2.1 queryDepth

```
size_t queryDepth
```

Definition at line 31 of file `breadth_first_dual_tree_traverser.hpp`.

39.525.2.2 queryNode

```
TreeType* queryNode
```

Definition at line 29 of file `breadth_first_dual_tree_traverser.hpp`.

39.525.2.3 referenceNode

```
TreeType* referenceNode
```

Definition at line 30 of file `breadth_first_dual_tree_traverser.hpp`.

39.525.2.4 score

```
double score
```

Definition at line 32 of file `breadth_first_dual_tree_traverser.hpp`.

39.525.2.5 traversalInfo

```
TraversalInfoType traversalInfo
```

Definition at line 33 of file `breadth_first_dual_tree_traverser.hpp`.

Referenced by `BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::BreadthFirstDualTreeTraverser< RuleType >::NumBaseCases()`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/breadth_first_dual_tree_traverser.hpp`

39.526 RandomDimensionSelect Class Reference

This dimension selection policy only selects one single random dimension.

Public Member Functions

- **RandomDimensionSelect ()**
*Construct the **RandomDimensionSelect** (p. 2194) object with the given number of dimensions.*
- `size_t Begin () const`
Get the first dimension to select from.
- `size_t Dimensions () const`
Get the number of dimensions.
- `size_t & Dimensions ()`
Set the number of dimensions.
- `size_t End () const`
Get the last dimension to select from.
- `size_t Next () const`
Get the next (last) dimensions.

39.526.1 Detailed Description

This dimension selection policy only selects one single random dimension.

Definition at line 21 of file `random_dimension_select.hpp`.

39.526.2 Constructor & Destructor Documentation

39.526.2.1 RandomDimensionSelect()

```
RandomDimensionSelect ( ) [inline]
```

Construct the **RandomDimensionSelect** (p. 2194) object with the given number of dimensions.

Definition at line 28 of file random_dimension_select.hpp.

39.526.3 Member Function Documentation

39.526.3.1 Begin()

```
size_t Begin ( ) const [inline]
```

Get the first dimension to select from.

Definition at line 33 of file random_dimension_select.hpp.

References `mlpack::math::RandInt()`.

39.526.3.2 Dimensions() [1/2]

```
size_t Dimensions ( ) const [inline]
```

Get the number of dimensions.

Definition at line 47 of file random_dimension_select.hpp.

39.526.3.3 Dimensions() [2/2]

```
size_t& Dimensions ( ) [inline]
```

Set the number of dimensions.

Definition at line 49 of file random_dimension_select.hpp.

39.526.3.4 End()

```
size_t End ( ) const [inline]
```

Get the last dimension to select from.

Definition at line 38 of file random_dimension_select.hpp.

39.526.3.5 Next()

```
size_t Next ( ) const [inline]
```

Get the next (last) dimensions.

We only allow one dimension, so any 'next' dimension is past our bounds.

Definition at line 44 of file random_dimension_select.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/ **random_dimension_select.hpp**

39.527 RandomForest< FitnessFunction, DimensionSelectionType, NumericSplitType, CategoricalSplitType, ElemType > Class Template Reference

Public Types

- typedef **DecisionTree**< FitnessFunction, NumericSplitType, CategoricalSplitType, DimensionSelectionType, ElemType > **DecisionTreeType**

Allow access to the underlying decision tree type.

Public Member Functions

- **RandomForest** ()

Construct the random forest without any training or specifying the number of trees.

- template<typename MatType >

RandomForest (const MatType &dataset, const arma::Row< size_t > &labels, const size_t numClasses, const size_t numTrees=20, const size_t minimumLeafSize=1, const double minimumGainSplit=1e-7, DimensionSelectionType dimensionSelector=DimensionSelectionType())

Create a random forest, training on the given labeled training data with the given number of trees.

- template<typename MatType >

RandomForest (const MatType &dataset, const **data::DatasetInfo** &datasetInfo, const arma::Row< size_t > &labels, const size_t numClasses, const size_t numTrees=20, const size_t minimumLeafSize=1, const double minimumGainSplit=1e-7, DimensionSelectionType dimensionSelector=DimensionSelectionType())

Create a random forest, training on the given labeled training data with the given dataset info and the given number of trees.

- template<typename MatType >

RandomForest (const MatType &dataset, const arma::Row< size_t > &labels, const size_t numClasses, const arma::rowvec &weights, const size_t numTrees=20, const size_t minimumLeafSize=1, const double minimumGainSplit=1e-7, DimensionSelectionType dimensionSelector=DimensionSelectionType())

Create a random forest, training on the given weighted labeled training data with the given number of trees.

- template<typename MatType >

RandomForest (const MatType &dataset, const **data::DatasetInfo** &datasetInfo, const arma::Row< size_t > &labels, const size_t numClasses, const arma::rowvec &weights, const size_t numTrees=20, const size_t minimumLeafSize=1, const double minimumGainSplit=1e-7, DimensionSelectionType dimensionSelector=DimensionSelectionType())

Create a random forest, training on the given weighted labeled training data with the given dataset info and the given number of trees.

- template<typename VecType >

size_t **Classify** (const VecType &point) const

Predict the class of the given point.

- template<typename VecType >

void **Classify** (const VecType &point, size_t &prediction, arma::vec &probabilities) const

Predict the class of the given point and return the predicted class probabilities for each class.

- template<typename MatType >

void **Classify** (const MatType &data, arma::Row< size_t > &predictions) const

Predict the classes of each point in the given dataset.

- template<typename MatType >

void **Classify** (const MatType &data, arma::Row< size_t > &predictions, arma::mat &probabilities) const

Predict the classes of each point in the given dataset, also returning the predicted class probabilities for each point.

- size_t **NumTrees** () const

Get the number of trees in the forest.

- template<typename Archive >

void **serialize** (Archive &ar, const unsigned int)

Serialize the random forest.

- template<typename MatType >

double **Train** (const MatType &data, const arma::Row< size_t > &labels, const size_t numClasses, const size_t numTrees=20, const size_t minimumLeafSize=1, const double minimumGainSplit=1e-7, DimensionSelectionType dimensionSelector=DimensionSelectionType())

Train the random forest on the given labeled training data with the given number of trees.

- `template<typename MatType >`
`double Train (const MatType &data, const data::DatasetInfo &datasetInfo, const arma::Row< size_t > &labels,`
`const size_t numClasses, const size_t numTrees=20, const size_t minimumLeafSize=1, const double minimumGain←`
`GainSplit=1e-7, DimensionSelectionType dimensionSelector=DimensionSelectionType())`
Train the random forest on the given labeled training data with the given dataset info and the given number of trees.
- `template<typename MatType >`
`double Train (const MatType &data, const arma::Row< size_t > &labels, const size_t numClasses, const arma←`
`::rowvec &weights, const size_t numTrees=20, const size_t minimumLeafSize=1, const double minimumGain←`
`Split=1e-7, DimensionSelectionType dimensionSelector=DimensionSelectionType())`
Train the random forest on the given weighted labeled training data with the given number of trees.
- `template<typename MatType >`
`double Train (const MatType &data, const data::DatasetInfo &datasetInfo, const arma::Row< size_t > &labels,`
`const size_t numClasses, const arma::rowvec &weights, const size_t numTrees=20, const size_t minimum←`
`LeafSize=1, const double minimumGainSplit=1e-7, DimensionSelectionType dimensionSelector=Dimension←`
`SelectionType())`
Train the random forest on the given weighted labeled training data with the given dataset info and the given number of trees.
- `const DecisionTreeType & Tree (const size_t i) const`
Access a tree in the forest.
- `DecisionTreeType & Tree (const size_t i)`
Modify a tree in the forest (be careful!).

39.527.1 Detailed Description

```
template<typename FitnessFunction = GiniGain, typename DimensionSelectionType = MultipleRandomDimensionSelect, template<
typename > class NumericSplitType = BestBinaryNumericSplit, template< typename > class CategoricalSplitType = AllCategorical←
Split, typename ElemType = double>
class mlpack::tree::RandomForest< FitnessFunction, DimensionSelectionType, NumericSplitType, CategoricalSplitType, ElemType >
```

Definition at line 27 of file random_forest.hpp.

39.527.2 Member Typedef Documentation

39.527.2.1 DecisionTreeType

```
typedef DecisionTree<FitnessFunction, NumericSplitType, CategoricalSplitType, DimensionSelection←
Type, ElemType> DecisionTreeType
```

Allow access to the underlying decision tree type.

Definition at line 32 of file random_forest.hpp.

39.527.3 Constructor & Destructor Documentation

39.527.3.1 RandomForest() [1/5]

```
RandomForest ( ) [inline]
```

Construct the random forest without any training or specifying the number of trees.

Predict() will throw an exception until **Train()** (p. 2203) is called.

Definition at line 38 of file random_forest.hpp.

References RandomForest< FitnessFunction, DimensionSelectionType, NumericSplitType, CategoricalSplitType, ElemType >::Classify(), and RandomForest< FitnessFunction, DimensionSelectionType, NumericSplitType, CategoricalSplitType, ElemType >::Train().

39.527.3.2 RandomForest() [2/5]

```
RandomForest (
    const MatType & dataset,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const size_t numTrees = 20,
    const size_t minimumLeafSize = 1,
    const double minimumGainSplit = 1e-7,
    DimensionSelectionType dimensionSelector = DimensionSelectionType() )
```

Create a random forest, training on the given labeled training data with the given number of trees.

The minimumLeafSize and minimumGainSplit parameters are given to each individual decision tree during tree building. Optionally, you may specify a DimensionSelectionType to set parameters for the strategy used to choose dimensions.

Parameters

<i>dataset</i>	Dataset to train on.
<i>labels</i>	Labels for dataset.
<i>numClasses</i>	Number of classes in dataset.
<i>numTrees</i>	Number of trees in the forest.
<i>minimumLeafSize</i>	Minimum number of points in each tree's leaf nodes.
<i>minimumGainSplit</i>	Minimum gain for splitting a decision tree node.
<i>dimensionSelector</i>	Instantiated dimension selection policy.

39.527.3.3 RandomForest() [3/5]

```

RandomForest (
    const MatType & dataset,
    const data::DatasetInfo & datasetInfo,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const size_t numTrees = 20,
    const size_t minimumLeafSize = 1,
    const double minimumGainSplit = 1e-7,
    DimensionSelectionType dimensionSelector = DimensionSelectionType() )

```

Create a random forest, training on the given labeled training data with the given dataset info and the given number of trees.

The minimumLeafSize and minimumGainSplit parameters are given to each individual decision tree during tree building. Optionally, you may specify a DimensionSelectionType to set parameters for the strategy used to choose dimensions. This constructor can be used to train on categorical data.

Parameters

<i>dataset</i>	Dataset to train on.
<i>datasetInfo</i>	Dimension info for the dataset.
<i>labels</i>	Labels for dataset.
<i>numClasses</i>	Number of classes in dataset.
<i>numTrees</i>	Number of trees in the forest.
<i>minimumLeafSize</i>	Minimum number of points in each tree's leaf nodes.
<i>minimumGainSplit</i>	Minimum gain for splitting a decision tree node.
<i>dimensionSelector</i>	Instantiated dimension selection policy.

39.527.3.4 RandomForest() [4/5]

```

RandomForest (
    const MatType & dataset,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const arma::rowvec & weights,
    const size_t numTrees = 20,
    const size_t minimumLeafSize = 1,
    const double minimumGainSplit = 1e-7,
    DimensionSelectionType dimensionSelector = DimensionSelectionType() )

```

Create a random forest, training on the given weighted labeled training data with the given number of trees.

The minimumLeafSize parameter is given to each individual decision tree during tree building.

Parameters

<i>dataset</i>	Dataset to train on.
<i>labels</i>	Labels for dataset.
<i>numClasses</i>	Number of classes in dataset.
<i>weights</i>	Weights (importances) of each point in the dataset.
<i>numTrees</i>	Number of trees in the forest.
<i>minimumLeafSize</i>	Minimum number of points in each tree's leaf nodes.

39.527.3.5 RandomForest() [5/5]

```

RandomForest (
    const MatType & dataset,
    const data::DatasetInfo & datasetInfo,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const arma::rowvec & weights,
    const size_t numTrees = 20,
    const size_t minimumLeafSize = 1,
    const double minimumGainSplit = 1e-7,
    DimensionSelectionType dimensionSelector = DimensionSelectionType() )

```

Create a random forest, training on the given weighted labeled training data with the given dataset info and the given number of trees.

The minimumLeafSize and minimumGainSplit parameters are given to each individual decision tree during tree building. Optionally, you may specify a DimensionSelectionType to set parameters for the strategy used to choose dimensions. This can be used for categorical weighted training.

Parameters

<i>dataset</i>	Dataset to train on.
<i>datasetInfo</i>	Dimension info for the dataset.
<i>labels</i>	Labels for dataset.
<i>numClasses</i>	Number of classes in dataset.
<i>weights</i>	Weights (importances) of each point in the dataset.
<i>numTrees</i>	Number of trees in the forest.
<i>minimumLeafSize</i>	Minimum number of points in each tree's leaf nodes.
<i>minimumGainSplit</i>	Minimum gain for splitting a decision tree node.
<i>dimensionSelector</i>	Instantiated dimension selection policy.

39.527.4 Member Function Documentation

39.527.4.1 Classify() [1/4]

```
size_t Classify (
    const VecType & point ) const
```

Predict the class of the given point.

If the random forest has not been trained, this will throw an exception.

Parameters

<i>point</i>	Point to be classified.
--------------	-------------------------

Referenced by RandomForest< FitnessFunction, DimensionSelectionType, NumericSplitType, CategoricalSplitType, ElemType >::RandomForest().

39.527.4.2 Classify() [2/4]

```
void Classify (
    const VecType & point,
    size_t & prediction,
    arma::vec & probabilities ) const
```

Predict the class of the given point and return the predicted class probabilities for each class.

If the random forest has not been trained, this will throw an exception.

Parameters

<i>point</i>	Point to be classified.
<i>prediction</i>	size_t to store predicted class in.
<i>probabilities</i>	Output vector of class probabilities.

39.527.4.3 Classify() [3/4]

```
void Classify (
    const MatType & data,
    arma::Row< size_t > & predictions ) const
```

Predict the classes of each point in the given dataset.

If the random forest has not been trained, this will throw an exception.

Parameters

<i>data</i>	Dataset to be classified.
<i>predictions</i>	Output predictions for each point in the dataset.

39.527.4.4 Classify() [4/4]

```
void Classify (
    const MatType & data,
    arma::Row< size_t > & predictions,
    arma::mat & probabilities ) const
```

Predict the classes of each point in the given dataset, also returning the predicted class probabilities for each point.

If the random forest has not been trained, this will throw an exception.

Parameters

<i>data</i>	Dataset to be classified.
<i>predictions</i>	Output predictions for each point in the dataset.
<i>probabilities</i>	Output matrix of class probabilities for each point.

39.527.4.5 NumTrees()

```
size_t NumTrees ( ) const [inline]
```

Get the number of trees in the forest.

Definition at line 315 of file random_forest.hpp.

References RandomForest< FitnessFunction, DimensionSelectionType, NumericSplitType, CategoricalSplitType, ElemType >::serialize(), and RandomForest< FitnessFunction, DimensionSelectionType, NumericSplitType, CategoricalSplitType, ElemType >::Train().

39.527.4.6 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the random forest.

Referenced by RandomForest< FitnessFunction, DimensionSelectionType, NumericSplitType, CategoricalSplitType, ElemType >::NumTrees().

39.527.4.7 Train() [1/4]

```
double Train (
    const MatType & data,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const size_t numTrees = 20,
    const size_t minimumLeafSize = 1,
    const double minimumGainSplit = 1e-7,
    DimensionSelectionType dimensionSelector = DimensionSelectionType() )
```

Train the random forest on the given labeled training data with the given number of trees.

The minimumLeafSize and minimumGainSplit parameters are given to each individual decision tree during tree building. Optionally, you may specify a DimensionSelectionType to set parameters for the strategy used to choose dimensions.

Parameters

<i>data</i>	Dataset to train on.
<i>labels</i>	Labels for dataset.
<i>numClasses</i>	Number of classes in dataset.
<i>numTrees</i>	Number of trees in the forest.
<i>minimumLeafSize</i>	Minimum number of points in each tree's leaf nodes.
<i>minimumGainSplit</i>	Minimum gain for splitting a decision tree node.
<i>dimensionSelector</i>	Instantiated dimension selection policy.

Returns

The average entropy of all the decision trees trained under forest.

Referenced by RandomForest< FitnessFunction, DimensionSelectionType, NumericSplitType, CategoricalSplitType, ElemType >::NumTrees(), and RandomForest< FitnessFunction, DimensionSelectionType, NumericSplitType, CategoricalSplitType, ElemType >::RandomForest().

39.527.4.8 Train() [2/4]

```
double Train (
    const MatType & data,
    const data::DatasetInfo & datasetInfo,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const size_t numTrees = 20,
    const size_t minimumLeafSize = 1,
    const double minimumGainSplit = 1e-7,
    DimensionSelectionType dimensionSelector = DimensionSelectionType() )
```

Train the random forest on the given labeled training data with the given dataset info and the given number of trees.

The minimumLeafSize parameter is given to each individual decision tree during tree building. Optionally, you may specify a DimensionSelectionType to set parameters for the strategy used to choose dimensions. This overload can be used to train on categorical data.

Parameters

<i>data</i>	Dataset to train on.
<i>datasetInfo</i>	Dimension info for the dataset.
<i>labels</i>	Labels for dataset.
<i>numClasses</i>	Number of classes in dataset.
<i>numTrees</i>	Number of trees in the forest.
<i>minimumLeafSize</i>	Minimum number of points in each tree's leaf nodes.
<i>minimumGainSplit</i>	Minimum gain for splitting a decision tree node.
<i>dimensionSelector</i>	Instantiated dimension selection policy.

Returns

The average entropy of all the decision trees trained under forest.

39.527.4.9 Train() [3/4]

```
double Train (
    const MatType & data,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const arma::rowvec & weights,
    const size_t numTrees = 20,
    const size_t minimumLeafSize = 1,
    const double minimumGainSplit = 1e-7,
    DimensionSelectionType dimensionSelector = DimensionSelectionType() )
```

Train the random forest on the given weighted labeled training data with the given number of trees.

The minimumLeafSize and minimumGainSplit parameters are given to each individual decision tree during tree building. Optionally, you may specify a DimensionSelectionType to set parameters for the strategy used to choose dimensions.

Parameters

<i>data</i>	Dataset to train on.
<i>labels</i>	Labels for dataset.
<i>numClasses</i>	Number of classes in dataset.
<i>weights</i>	Weights (importances) of each point in the dataset.
<i>numTrees</i>	Number of trees in the forest.
<i>minimumLeafSize</i>	Minimum number of points in each tree's leaf nodes.
<i>minimumGainSplit</i>	Minimum gain for splitting a decision tree node.
<i>dimensionSelector</i>	Instantiated dimension selection policy.

Returns

The average entropy of all the decision trees trained under forest.

39.527.4.10 Train() [4/4]

```
double Train (
    const MatType & data,
    const data::DatasetInfo & datasetInfo,
    const arma::Row< size_t > & labels,
    const size_t numClasses,
    const arma::rowvec & weights,
    const size_t numTrees = 20,
    const size_t minimumLeafSize = 1,
    const double minimumGainSplit = 1e-7,
    DimensionSelectionType dimensionSelector = DimensionSelectionType() )
```

Train the random forest on the given weighted labeled training data with the given dataset info and the given number of trees.

The minimumLeafSize and minimumGainSplit parameters are given to each individual decision tree during tree building. Optionally, you may specify a DimensionSelectionType to set parameters for the strategy used to choose dimensions. This overload can be used for categorical weighted training.

Parameters

<i>data</i>	Dataset to train on.
<i>datasetInfo</i>	Dimension info for the dataset.
<i>labels</i>	Labels for dataset.
<i>numClasses</i>	Number of classes in dataset.
<i>weights</i>	Weights (importances) of each point in the dataset.
<i>numTrees</i>	Number of trees in the forest.
<i>minimumLeafSize</i>	Minimum number of points in each tree's leaf nodes.
<i>minimumGainSplit</i>	Minimum gain for splitting a decision tree node.
<i>dimensionSelector</i>	Instantiated dimension selection policy.

Returns

The average entropy of all the decision trees trained under forest.

39.527.4.11 Tree() [1/2]

```
const DecisionTreeType& Tree (
    const size_t i ) const [inline]
```

Access a tree in the forest.

Definition at line 310 of file random_forest.hpp.

39.527.4.12 Tree() [2/2]

```
DecisionTreeType& Tree (  
    const size_t i ) [inline]
```

Modify a tree in the forest (be careful!).

Definition at line 312 of file random_forest.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/random_forest/ **random_forest.hpp**

39.528 RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType > Class Template Reference

A rectangle type tree tree, such as an R-tree or X-tree.

Classes

- class **DualTreeTraverser**
A dual tree traverser for rectangle type trees.
- class **SingleTreeTraverser**
A single traverser for rectangle type trees.

Public Types

- typedef AuxiliaryInformationType< **RectangleTree** > **AuxiliaryInformation**
The auxiliary information type held by the tree.
- typedef MatType::elem_type **ElemType**
The element type held by the matrix type.
- typedef MatType **Mat**
So other classes can use TreeType::Mat.

Public Member Functions

- **RectangleTree** (const MatType &data, const size_t maxLeafSize=20, const size_t minLeafSize=8, const size_t maxNumChildren=5, const size_t minNumChildren=2, const size_t firstDataIndex=0)
Construct this as the root node of a rectangle type tree using the given dataset.
- **RectangleTree** (MatType &&data, const size_t maxLeafSize=20, const size_t minLeafSize=8, const size_t maxNumChildren=5, const size_t minNumChildren=2, const size_t firstDataIndex=0)
Construct this as the root node of a rectangle tree type using the given dataset, and taking ownership of the given dataset.
- **RectangleTree** (**RectangleTree** *parentNode, const size_t numMaxChildren=0)
Construct this as an empty node with the specified parent.
- **RectangleTree** (const **RectangleTree** &other, const bool deepCopy=true, **RectangleTree** *newParent=NULL)
Create a rectangle tree by copying the other tree.
- **RectangleTree** (**RectangleTree** &&other)
Create a rectangle tree by moving the other tree.
- template<typename Archive >
RectangleTree (Archive &ar, const typename **std::enable_if_t**< Archive::is_loading::value > *=0)
*Construct the tree from a **boost::serialization** (p. 251) archive.*
- ~**RectangleTree** ()
Deletes this node, deallocating the memory for the children and calling their destructors in turn.
- const AuxiliaryInformationType< **RectangleTree** > & **AuxiliaryInfo** () const
Return the auxiliary information object of this node.
- AuxiliaryInformationType< **RectangleTree** > & **AuxiliaryInfo** ()
Modify the split object of this node.
- size_t **Begin** () const
Return the index of the beginning point of this subset.
- size_t & **Begin** ()
Modify the index of the beginning point of this subset.
- const **bound::HRectBound**< MetricType > & **Bound** () const
Return the bound object for this node.
- **bound::HRectBound**< MetricType > & **Bound** ()
Modify the bound object for this node.
- void **Center** (arma::vec ¢er)
Get the centroid of the node and store it in the given vector.
- **RectangleTree** & **Child** (const size_t child) const
Get the specified child.
- **RectangleTree** & **Child** (const size_t child)
Modify the specified child.
- void **CondenseTree** (const arma::vec &point, std::vector< bool > &relevels, const bool usePoint)
Condense the bounding rectangles for this node based on the removal of the point specified by the arma::vec&.
- size_t **Count** () const
Return the number of points in this subset.
- size_t & **Count** ()
Modify the number of points in this subset.
- const MatType & **Dataset** () const
Get the dataset which the tree is built on.
- MatType & **Dataset** ()
Modify the dataset which the tree is built on. Be careful!
- bool **DeletePoint** (const size_t point)

- Deletes a point from the tree and, updates the bounding rectangle.*

 - **bool DeletePoint** (const size_t point, std::vector< bool > &relevels)
- Deletes a point from the tree, updates the bounding rectangle, tracking levels.*

 - **size_t Descendant** (const size_t index) const
- Return the index (with reference to the dataset) of a particular descendant of this node.*

 - **RectangleTree * ExactClone** ()
- Make an exact copy of this node, pointers and everything.*

 - **const RectangleTree * FindByBeginCount** (size_t begin, size_t count) const
- Find a node in this tree by its begin and count (const).*

 - **RectangleTree * FindByBeginCount** (size_t begin, size_t count)
- Find a node in this tree by its begin and count.*

 - **ElemType FurthestDescendantDistance** () const
- Return the furthest possible descendant distance.*

 - **ElemType FurthestPointDistance** () const
- Return the furthest distance to a point held in this node.*

 - **template<typename VecType >**
size_t GetFurthestChild (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > !=0)
- Return the index of the furthest child node to the given query point.*

 - **size_t GetFurthestChild** (const **RectangleTree** &queryNode)
- Return the index of the furthest child node to the given query node.*

 - **template<typename VecType >**
size_t GetNearestChild (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > !=0)
- Return the index of the nearest child node to the given query point.*

 - **size_t GetNearestChild** (const **RectangleTree** &queryNode)
- Return the index of the nearest child node to the given query node.*

 - **void InsertNode** (**RectangleTree** *node, const size_t level, std::vector< bool > &relevels)
- Inserts a node into the tree, tracking which levels have been inserted into.*

 - **void InsertPoint** (const size_t point)
- Inserts a point into the tree.*

 - **void InsertPoint** (const size_t point, std::vector< bool > &relevels)
- Inserts a point into the tree, tracking which levels have been inserted into.*

 - **bool IsLeaf** () const
- Return whether or not this node is a leaf (true if it has no children).*

 - **ElemType MaxDistance** (const **RectangleTree** &other) const
- Return the maximum distance to another node.*

 - **template<typename VecType >**
ElemType MaxDistance (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > !=0) const
- Return the maximum distance to another point.*

 - **size_t MaxLeafSize** () const
- Return the maximum leaf size.*

 - **size_t & MaxLeafSize** ()
- Modify the maximum leaf size.*

 - **size_t MaxNumChildren** () const
- Return the maximum number of children (in a non-leaf node).*

 - **size_t & MaxNumChildren** ()

- Modify the maximum number of children (in a non-leaf node).*

 - **MetricType Metric** () const
 - Get the metric which the tree uses.*
 - **ElemType MinDistance** (const **RectangleTree** &other) const
 - Return the minimum distance to another node.*
 - template<typename VecType >
 - ElemType MinDistance** (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > !=0) const

Return the minimum distance to another point.
 - **ElemType MinimumBoundDistance** () const
 - Return the minimum distance from the center to any edge of the bound.*
 - size_t **MinLeafSize** () const
 - Return the minimum leaf size.*
 - size_t & **MinLeafSize** ()
 - Modify the minimum leaf size.*
 - size_t **MinNumChildren** () const
 - Return the minimum number of children (in a non-leaf node).*
 - size_t & **MinNumChildren** ()
 - Modify the minimum number of children (in a non-leaf node).*
 - void **NullifyData** ()
 - Nullify the auxiliary information.*
 - size_t **NumChildren** () const
 - Return the number of child nodes. (One level beneath this one only.)*
 - size_t & **NumChildren** ()
 - Modify the number of child nodes. Be careful.*
 - size_t **NumDescendants** () const
 - Return the number of descendants of this node.*
 - size_t **NumPoints** () const
 - Return the number of points in this node (returns 0 if this node is not a leaf).*
 - **RectangleTree * Parent** () const
 - Gets the parent of this node.*
 - **RectangleTree *& Parent** ()
 - Modify the parent of this node.*
 - **ElemType ParentDistance** () const
 - Return the distance from the center of this node to the center of the parent node.*
 - **ElemType & ParentDistance** ()
 - Modify the distance from the center of this node to the center of the parent node.*
 - size_t **Point** (const size_t index) const
 - Return the index (with reference to the dataset) of a particular point in this node.*
 - size_t & **Point** (const size_t index)
 - Modify the index of a particular point in this node.*
 - **math::RangeType**< **ElemType** > **RangeDistance** (const **RectangleTree** &other) const
 - Return the minimum and maximum distance to another node.*
 - template<typename VecType >
 - math::RangeType**< **ElemType** > **RangeDistance** (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > !=0) const

Return the minimum and maximum distance to another point.
 - bool **RemoveNode** (const **RectangleTree** *node, std::vector< bool > &relevels)

Removes a node from the tree.

- `template<typename Archive >`
`void serialize (Archive &ar, const unsigned int)`
Serialize the tree.
- `bool ShrinkBoundForBound (const bound::HRectBound< MetricType > &changedBound)`
Shrink the bound object of this node for the removal of a child node.
- `bool ShrinkBoundForPoint (const arma::vec &point)`
Shrink the bound object of this node for the removal of a point.
- `void SoftDelete ()`
Delete this node of the tree, but leave the stuff contained in it intact.
- `const StatisticType & Stat () const`
Return the statistic object for this node.
- `StatisticType & Stat ()`
Modify the statistic object for this node.
- `size_t TreeDepth () const`
Obtains the number of levels below this node in the tree, starting with this.
- `size_t TreeSize () const`
Obtains the number of nodes in the tree, starting with this.

Protected Member Functions

- `RectangleTree ()`
A default constructor.

Protected Attributes

- `friend AuxiliaryInformation`
Give friend access for AuxiliaryInformationType.
- `friend DescentType`
Give friend access for DescentType.
- `friend SplitType`
Give friend access for SplitType.

39.528.1 Detailed Description

```
template<typename MetricType = metric::EuclideanDistance, typename StatisticType = EmptyStatistic, typename MatType = arma::mat,
typename SplitType = RTreeSplit, typename DescentType = RTreeDescentHeuristic, template< typename > class AuxiliaryInformationType = NoAuxiliaryInformation>
class mlpack::tree::RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >
```

A rectangle type tree tree, such as an R-tree or X-tree.

Once the bound and type of dataset is defined, the tree will construct itself. Call the constructor with the dataset to build the tree on, and the entire tree will be built.

This tree does allow growth, so you can add and delete nodes from it.

Template Parameters

<i>MetricType</i>	This <i>must</i> be EuclideanDistance, but the template parameter is required to satisfy the TreeType API.
<i>StatisticType</i>	Extra data contained in the node. See statistic.hpp (p. 2688) for the necessary skeleton interface.
<i>MatType</i>	The dataset class.
<i>SplitType</i>	The type of split to use when inserting points.
<i>DescentType</i>	The heuristic to use when descending the tree to insert points.
<i>AuxiliaryInformationType</i>	An auxiliary information contained in the node. This information depends on the type of the RectangleTree (p. 2207).

Definition at line 54 of file rectangle_tree.hpp.

39.528.2 Member Typedef Documentation

39.528.2.1 AuxiliaryInformation

```
typedef AuxiliaryInformationType< RectangleTree> AuxiliaryInformation
```

The auxiliary information type held by the tree.

Definition at line 66 of file rectangle_tree.hpp.

39.528.2.2 ElemType

```
typedef MatType::elem_type ElemType
```

The element type held by the matrix type.

Definition at line 64 of file rectangle_tree.hpp.

39.528.2.3 Mat

```
typedef MatType Mat
```

So other classes can use TreeType::Mat.

Definition at line 58 of file rectangle_tree.hpp.

39.528.3 Constructor & Destructor Documentation

39.528.3.1 RectangleTree() [1/7]

```
RectangleTree (
    const MatType & data,
    const size_t maxLeafSize = 20,
    const size_t minLeafSize = 8,
    const size_t maxNumChildren = 5,
    const size_t minNumChildren = 2,
    const size_t firstDataIndex = 0 )
```

Construct this as the root node of a rectangle type tree using the given dataset.

This will modify the ordering of the points in the dataset!

Parameters

<i>data</i>	Dataset from which to create the tree. This will be modified!
<i>maxLeafSize</i>	Maximum size of each leaf in the tree.
<i>minLeafSize</i>	Minimum size of each leaf in the tree.
<i>maxNumChildren</i>	The maximum number of child nodes a non-leaf node may have.
<i>minNumChildren</i>	The minimum number of child nodes a non-leaf node may have.
<i>firstDataIndex</i>	The index of the first data point. UNUSED UNLESS WE ADD SUPPORT FOR HAVING A "CENTRAL" DATA MATRIX.

39.528.3.2 RectangleTree() [2/7]

```
RectangleTree (
    MatType && data,
    const size_t maxLeafSize = 20,
    const size_t minLeafSize = 8,
    const size_t maxNumChildren = 5,
    const size_t minNumChildren = 2,
    const size_t firstDataIndex = 0 )
```

Construct this as the root node of a rectangle tree type using the given dataset, and taking ownership of the given dataset.

Parameters

<i>data</i>	Dataset from which to create the tree.
<i>maxLeafSize</i>	Maximum size of each leaf in the tree.
<i>minLeafSize</i>	Minimum size of each leaf in the tree.

Parameters

<i>maxNumChildren</i>	The maximum number of child nodes a non-leaf node may have.
<i>minNumChildren</i>	The minimum number of child nodes a non-leaf node may have.
<i>firstDataIndex</i>	The index of the first data point. UNUSED UNLESS WE ADD SUPPORT FOR HAVING A "CENTRAL" DATA MATRIX.

39.528.3.3 `RectangleTree()` [3/7]

```

RectangleTree (
    RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, Auxiliary↵
InformationType > * parentNode,
    const size_t numMaxChildren = 0 ) [explicit]

```

Construct this as an empty node with the specified parent.

Copying the parameters (maxLeafSize, minLeafSize, maxNumChildren, minNumChildren, firstDataIndex) from the parent.

Parameters

<i>parentNode</i>	The parent of the node that is being constructed.
<i>numMaxChildren</i>	The max number of child nodes (used in x-trees).

39.528.3.4 `RectangleTree()` [4/7]

```

RectangleTree (
    const RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType,
AuxiliaryInformationType > & other,
    const bool deepCopy = true,
    RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, Auxiliary↵
InformationType > * newParent = NULL )

```

Create a rectangle tree by copying the other tree.

Be careful! This can take a long time and use a lot of memory.

Parameters

<i>other</i>	The tree to be copied.
<i>deepCopy</i>	If false, the children are not recursively copied.

39.528.3.5 **RectangleTree**() [5/7]

```
RectangleTree (
    RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType > && other )
```

Create a rectangle tree by moving the other tree.

Parameters

<i>other</i>	The tree to be copied.
--------------	------------------------

39.528.3.6 **RectangleTree**() [6/7]

```
RectangleTree (
    Archive & ar,
    const typename std::enable_if_t< Archive::is_loading::value > * = 0 )
```

Construct the tree from a **boost::serialization** (p. 251) archive.

39.528.3.7 **~RectangleTree**()

```
~RectangleTree ( )
```

Deletes this node, deallocating the memory for the children and calling their destructors in turn.

This will invalidate any younters or references to any nodes which are children of this one.

39.528.3.8 **RectangleTree**() [7/7]

```
RectangleTree ( ) [protected]
```

A default constructor.

This is meant to only be used with **boost::serialization** (p. 251), which is allowed with the friend declaration below. This does not return a valid tree! This method must be protected, so that the serialization shim can work with the default constructor.

Referenced by **RectangleTree**< **MetricType**, **StatisticType**, **MatType**, **SplitType**, **DescentType**, **AuxiliaryInformationType**>::**Count**().

39.528.4 Member Function Documentation

39.528.4.1 AuxiliaryInfo() [1/2]

```
const AuxiliaryInformationType< RectangleTree>& AuxiliaryInfo ( ) const [inline]
```

Return the auxiliary information object of this node.

Definition at line 310 of file rectangle_tree.hpp.

39.528.4.2 AuxiliaryInfo() [2/2]

```
AuxiliaryInformationType< RectangleTree>& AuxiliaryInfo ( ) [inline]
```

Modify the split object of this node.

Definition at line 313 of file rectangle_tree.hpp.

References `RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::IsLeaf()`.

39.528.4.3 Begin() [1/2]

```
size_t Begin ( ) const [inline]
```

Return the index of the beginning point of this subset.

Definition at line 528 of file rectangle_tree.hpp.

39.528.4.4 Begin() [2/2]

```
size_t& Begin ( ) [inline]
```

Modify the index of the beginning point of this subset.

Definition at line 530 of file rectangle_tree.hpp.

39.528.4.5 Bound() [1/2]

```
const bound::HRectBound<MetricType>& Bound ( ) const [inline]
```

Return the bound object for this node.

Definition at line 300 of file rectangle_tree.hpp.

Referenced by RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::MaxDistance(), RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::MinDistance(), and RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::RangeDistance().

39.528.4.6 Bound() [2/2]

```
bound::HRectBound<MetricType>& Bound ( ) [inline]
```

Modify the bound object for this node.

Definition at line 302 of file rectangle_tree.hpp.

39.528.4.7 Center()

```
void Center (
    arma::vec & center ) [inline]
```

Get the centroid of the node and store it in the given vector.

Definition at line 353 of file rectangle_tree.hpp.

References HRectBound< MetricType, ElemType >::Center().

39.528.4.8 Child() [1/2]

```
RectangleTree& Child (
    const size_t child ) const [inline]
```

Get the specified child.

Parameters

<i>child</i>	Index of child to return.
--------------	---------------------------

Definition at line 422 of file rectangle_tree.hpp.

39.528.4.9 Child() [2/2]

```
RectangleTree& Child (
    const size_t child ) [inline]
```

Modify the specified child.

Parameters

<i>child</i>	Index of child to return.
--------------	---------------------------

Definition at line 432 of file rectangle_tree.hpp.

References RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::Descendant(), RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::NumDescendants(), and RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::NumPoints().

39.528.4.10 CondenseTree()

```
void CondenseTree (
    const arma::vec & point,
    std::vector< bool > & relevels,
    const bool usePoint )
```

Condense the bounding rectangles for this node based on the removal of the point specified by the arma::vec&.

This recurses up the tree. If a node goes below the minimum fill, this function will fix the tree.

Parameters

<i>point</i>	The arma::vec& of the point that was removed to require this condensation of the tree.
<i>usePoint</i>	True if we use the optimized version of the algorithm that is possible when we now what point was deleted. False otherwise (eg. if we deleted a node instead of a point).

39.528.4.11 Count() [1/2]

```
size_t Count ( ) const [inline]
```

Return the number of points in this subset.

Definition at line 533 of file rectangle_tree.hpp.

39.528.4.12 Count() [2/2]

```
size_t& Count ( ) [inline]
```

Modify the number of points in this subset.

Definition at line 535 of file rectangle_tree.hpp.

References `RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::RectangleTree()`.

39.528.4.13 Dataset() [1/2]

```
const MatType& Dataset ( ) const [inline]
```

Get the dataset which the tree is built on.

Definition at line 345 of file rectangle_tree.hpp.

39.528.4.14 Dataset() [2/2]

```
MatType& Dataset ( ) [inline]
```

Modify the dataset which the tree is built on. Be careful!

Definition at line 347 of file rectangle_tree.hpp.

39.528.4.15 DeletePoint() [1/2]

```
bool DeletePoint (
    const size_t point )
```

Deletes a point from the tree and, updates the bounding rectangle.

However, the point will be kept in the central dataset. (The user may remove it from there if he wants, but he must not change the indices of the other points.) Returns true if the point is successfully removed and false if it is not. (ie. the point is not in the tree)

39.528.4.16 DeletePoint() [2/2]

```
bool DeletePoint (
    const size_t point,
    std::vector< bool > & relevels )
```

Deletes a point from the tree, updates the bounding rectangle, tracking levels.

However, the point will be kept in the central dataset. (The user may remove it from there if he wants, but he must not change the indices of the other points.) Returns true if the point is successfully removed and false if it is not. (ie. the point is not in the tree)

39.528.4.17 Descendant()

```
size_t Descendant (
    const size_t index ) const
```

Return the index (with reference to the dataset) of a particular descendant of this node.

The index should be greater than zero but less than the number of descendants.

Parameters

<i>index</i>	Index of the descendant.
--------------	--------------------------

Referenced by RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::Child().

39.528.4.18 ExactClone()

```
RectangleTree* ExactClone ( )
```

Make an exact copy of this node, pointers and everything.

39.528.4.19 FindByBeginCount() [1/2]

```
const RectangleTree* FindByBeginCount (
    size_t begin,
    size_t count ) const
```

Find a node in this tree by its begin and count (const).

Every node is uniquely identified by these two numbers. This is useful for communicating position over the network, when pointers would be invalid.

Parameters

<i>begin</i>	The begin() of the node to find.
<i>count</i>	The count() of the node to find.

Returns

The found node, or NULL if not found.

39.528.4.20 FindByBeginCount() [2/2]

```
RectangleTree* FindByBeginCount (
    size_t begin,
    size_t count )
```

Find a node in this tree by its begin and count.

Every node is uniquely identified by these two numbers. This is useful for communicating position over the network, when pointers would be invalid.

Parameters

<i>begin</i>	The begin() of the node to find.
<i>count</i>	The count() of the node to find.

Returns

The found node, or NULL if not found.

39.528.4.21 FurthestDescendantDistance()

```
ElemType FurthestDescendantDistance ( ) const
```

Return the furthest possible descendant distance.

This returns the maximum distance from the centroid to the edge of the bound and not the empirical quantity which is the actual furthest descendant distance. So the actual furthest descendant distance may be less than what this method returns (but it will never be greater than this).

Referenced by RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::NumChildren().

39.528.4.22 FurthestPointDistance()

```
ElemType FurthestPointDistance ( ) const
```

Return the furthest distance to a point held in this node.

If this is not a leaf node, then the distance is 0 because the node holds no points.

Referenced by RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::NumChildren().

39.528.4.23 GetFurthestChild() [1/2]

```
size_t GetFurthestChild (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 )
```

Return the index of the furthest child node to the given query point.

If this is a leaf node, it will return **NumChildren()** (p. 2227) (invalid index).

Referenced by RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::NumChildren().

39.528.4.24 GetFurthestChild() [2/2]

```
size_t GetFurthestChild (
    const RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType,
    AuxiliaryInformationType > & queryNode )
```

Return the index of the furthest child node to the given query node.

If it can't decide, it will return **NumChildren()** (p. 2227) (invalid index).

39.528.4.25 GetNearestChild() [1/2]

```
size_t GetNearestChild (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 )
```

Return the index of the nearest child node to the given query point.

If this is a leaf node, it will return **NumChildren()** (p. 2227) (invalid index).

Referenced by RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::NumChildren().

39.528.4.26 **GetNearestChild()** [2/2]

```
size_t GetNearestChild (
    const RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType,
    AuxiliaryInformationType > & queryNode )
```

Return the index of the nearest child node to the given query node.

If it can't decide, it will return **NumChildren()** (p.2227) (invalid index).

39.528.4.27 **InsertNode()**

```
void InsertNode (
    RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, Auxiliary←
    InformationType > * node,
    const size_t level,
    std::vector< bool > & relevels )
```

Inserts a node into the tree, tracking which levels have been inserted into.

The node will be inserted so that the tree remains valid.

Parameters

<i>node</i>	The node to be inserted.
<i>level</i>	The depth that should match the node where this node is finally inserted. This should be the number returned by calling TreeDepth() (p. 2232) from the node that originally contained "node".
<i>relevels</i>	The levels that have been reinserted to on this top level insertion.

39.528.4.28 **InsertPoint()** [1/2]

```
void InsertPoint (
    const size_t point )
```

Inserts a point into the tree.

Parameters

<i>point</i>	The index of a point in the dataset.
--------------	--------------------------------------

39.528.4.29 InsertPoint() [2/2]

```
void InsertPoint (
    const size_t point,
    std::vector< bool > & relevels )
```

Inserts a point into the tree, tracking which levels have been inserted into.

Parameters

<i>point</i>	The index of a point in the dataset.
<i>relevels</i>	The levels that have been reinserted to on this top level insertion.

39.528.4.30 IsLeaf()

```
bool IsLeaf ( ) const
```

Return whether or not this node is a leaf (true if it has no children).

Referenced by `RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::AuxiliaryInfo()`.

39.528.4.31 MaxDistance() [1/2]

```
ElemType MaxDistance (
    const RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType,
    AuxiliaryInformationType > & other ) const [inline]
```

Return the maximum distance to another node.

Definition at line 478 of file `rectangle_tree.hpp`.

References `RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::Bound()`, and `HRectBound< MetricType, ElemType >::MaxDistance()`.

39.528.4.32 MaxDistance() [2/2]

```
ElemType MaxDistance (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const [inline]
```

Return the maximum distance to another point.

Definition at line 500 of file `rectangle_tree.hpp`.

References `HRectBound< MetricType, ElemType >::MaxDistance()`.

39.528.4.33 MaxLeafSize() [1/2]

```
size_t MaxLeafSize ( ) const [inline]
```

Return the maximum leaf size.

Definition at line 320 of file rectangle_tree.hpp.

39.528.4.34 MaxLeafSize() [2/2]

```
size_t& MaxLeafSize ( ) [inline]
```

Modify the maximum leaf size.

Definition at line 322 of file rectangle_tree.hpp.

39.528.4.35 MaxNumChildren() [1/2]

```
size_t MaxNumChildren ( ) const [inline]
```

Return the maximum number of children (in a non-leaf node).

Definition at line 330 of file rectangle_tree.hpp.

39.528.4.36 MaxNumChildren() [2/2]

```
size_t& MaxNumChildren ( ) [inline]
```

Modify the maximum number of children (in a non-leaf node).

Definition at line 332 of file rectangle_tree.hpp.

39.528.4.37 Metric()

```
MetricType Metric ( ) const [inline]
```

Get the metric which the tree uses.

Definition at line 350 of file rectangle_tree.hpp.

39.528.4.38 MinDistance() [1/2]

```
ElemType MinDistance (
    const RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType,
    AuxiliaryInformationType > & other ) const [inline]
```

Return the minimum distance to another node.

Definition at line 472 of file rectangle_tree.hpp.

References `RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::Bound()`, and `HRectBound< MetricType, ElemType >::MinDistance()`.

39.528.4.39 MinDistance() [2/2]

```
ElemType MinDistance (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const [inline]
```

Return the minimum distance to another point.

Definition at line 491 of file rectangle_tree.hpp.

References `HRectBound< MetricType, ElemType >::MinDistance()`.

39.528.4.40 MinimumBoundDistance()

```
ElemType MinimumBoundDistance ( ) const [inline]
```

Return the minimum distance from the center to any edge of the bound.

Currently, this returns 0, which doesn't break algorithms, but it isn't necessarily correct, either.

Definition at line 408 of file rectangle_tree.hpp.

References `HRectBound< MetricType, ElemType >::MinWidth()`.

39.528.4.41 MinLeafSize() [1/2]

```
size_t MinLeafSize ( ) const [inline]
```

Return the minimum leaf size.

Definition at line 325 of file rectangle_tree.hpp.

39.528.4.42 MinLeafSize() [2/2]

```
size_t& MinLeafSize ( ) [inline]
```

Modify the minimum leaf size.

Definition at line 327 of file rectangle_tree.hpp.

39.528.4.43 MinNumChildren() [1/2]

```
size_t MinNumChildren ( ) const [inline]
```

Return the minimum number of children (in a non-leaf node).

Definition at line 335 of file rectangle_tree.hpp.

39.528.4.44 MinNumChildren() [2/2]

```
size_t& MinNumChildren ( ) [inline]
```

Modify the minimum number of children (in a non-leaf node).

Definition at line 337 of file rectangle_tree.hpp.

39.528.4.45 NullifyData()

```
void NullifyData ( )
```

Nullify the auxiliary information.

Used for memory management. Be careful.

39.528.4.46 NumChildren() [1/2]

```
size_t NumChildren ( ) const [inline]
```

Return the number of child nodes. (One level beneath this one only.)

Definition at line 356 of file rectangle_tree.hpp.

39.528.4.47 NumChildren() [2/2]

```
size_t& NumChildren ( ) [inline]
```

Modify the number of child nodes. Be careful.

Definition at line 358 of file rectangle_tree.hpp.

References `RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::FurthestDescendantDistance()`, `RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::FurthestPointDistance()`, `RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::GetFurthestChild()`, and `RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::GetNearestChild()`.

39.528.4.48 NumDescendants()

```
size_t NumDescendants ( ) const
```

Return the number of descendants of this node.

For a non-leaf in a binary space tree, this is the number of points at the descendant leaves. For a leaf, this is the number of points in the leaf.

Referenced by `RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::Child()`.

39.528.4.49 NumPoints()

```
size_t NumPoints ( ) const
```

Return the number of points in this node (returns 0 if this node is not a leaf).

Referenced by `RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::Child()`.

39.528.4.50 Parent() [1/2]

```
RectangleTree* Parent ( ) const [inline]
```

Gets the parent of this node.

Definition at line 340 of file rectangle_tree.hpp.

39.528.4.51 Parent() [2/2]

```
RectangleTree*& Parent ( ) [inline]
```

Modify the parent of this node.

Definition at line 342 of file rectangle_tree.hpp.

39.528.4.52 ParentDistance() [1/2]

```
ElemType ParentDistance ( ) const [inline]
```

Return the distance from the center of this node to the center of the parent node.

Definition at line 412 of file rectangle_tree.hpp.

39.528.4.53 ParentDistance() [2/2]

```
ElemType& ParentDistance ( ) [inline]
```

Modify the distance from the center of this node to the center of the parent node.

Definition at line 415 of file rectangle_tree.hpp.

39.528.4.54 Point() [1/2]

```
size_t Point (
    const size_t index ) const [inline]
```

Return the index (with reference to the dataset) of a particular point in this node.

This will happily return invalid indices if the given index is greater than the number of points in this node (obtained with **NumPoints()** (p. 2228)) – be careful.

Parameters

<i>index</i>	Index of point for which a dataset index is wanted.
--------------	---

Definition at line 465 of file rectangle_tree.hpp.

39.528.4.55 Point() [2/2]

```
size_t& Point (
    const size_t index ) [inline]
```

Modify the index of a particular point in this node.

Be very careful when you do this! You may make the tree invalid.

Definition at line 469 of file rectangle_tree.hpp.

39.528.4.56 RangeDistance() [1/2]

```
math::RangeType< ElemType> RangeDistance (
    const RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType,
    AuxiliaryInformationType > & other ) const [inline]
```

Return the minimum and maximum distance to another node.

Definition at line 484 of file rectangle_tree.hpp.

References RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::Bound(), and HRectBound< MetricType, ElemType >::RangeDistance().

39.528.4.57 RangeDistance() [2/2]

```
math::RangeType< ElemType> RangeDistance (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const [inline]
```

Return the minimum and maximum distance to another point.

Definition at line 509 of file rectangle_tree.hpp.

References HRectBound< MetricType, ElemType >::RangeDistance(), RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::TreeDepth(), and RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::TreeSize().

39.528.4.58 RemoveNode()

```
bool RemoveNode (
    const RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType,
    AuxiliaryInformationType > * node,
    std::vector< bool > & relevels )
```

Removes a node from the tree.

You are responsible for deleting it if you wish to do so.

39.528.4.59 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int )
```

Serialize the tree.

39.528.4.60 ShrinkBoundForBound()

```
bool ShrinkBoundForBound (
    const bound::HRectBound< MetricType > & changedBound )
```

Shrink the bound object of this node for the removal of a child node.

Parameters

<i>bound</i>	The HRectBound <> & of the bound that was removed to require this shrinking.
--------------	---

Returns

true if the bound needed to be changed, false if it did not.

39.528.4.61 ShrinkBoundForPoint()

```
bool ShrinkBoundForPoint (
    const arma::vec & point )
```

Shrink the bound object of this node for the removal of a point.

Parameters

<i>point</i>	The arma::vec& of the point that was removed to require this shrinking.
--------------	---

Returns

true if the bound needed to be changed, false if it did not.

39.528.4.62 SoftDelete()

```
void SoftDelete ( )
```

Delete this node of the tree, but leave the stuff contained in it intact.

This is used when splitting a node, where the data in this tree is moved to two other trees.

39.528.4.63 Stat() [1/2]

```
const StatisticType& Stat ( ) const [inline]
```

Return the statistic object for this node.

Definition at line 305 of file rectangle_tree.hpp.

39.528.4.64 Stat() [2/2]

```
StatisticType& Stat ( ) [inline]
```

Modify the statistic object for this node.

Definition at line 307 of file rectangle_tree.hpp.

39.528.4.65 TreeDepth()

```
size_t TreeDepth ( ) const
```

Obtains the number of levels below this node in the tree, starting with this.

Referenced by `RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::RangeDistance()`.

39.528.4.66 TreeSize()

```
size_t TreeSize ( ) const
```

Obtains the number of nodes in the tree, starting with this.

Referenced by RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::RangeDistance().

39.528.5 Member Data Documentation

39.528.5.1 AuxiliaryInformation

```
friend AuxiliaryInformation [protected]
```

Give friend access for AuxiliaryInformationType.

Definition at line 564 of file rectangle_tree.hpp.

39.528.5.2 DescentType

```
friend DescentType [protected]
```

Give friend access for DescentType.

Definition at line 558 of file rectangle_tree.hpp.

39.528.5.3 SplitType

```
friend SplitType [protected]
```

Give friend access for SplitType.

Definition at line 561 of file rectangle_tree.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ **rectangle_tree.hpp**

39.529 `RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::DualTreeTraverser< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >` Class Template Reference

A dual tree traverser for rectangle type trees.

Public Member Functions

- **DualTreeTraverser** (RuleType &rule)
Instantiate the dual-tree traverser with the given rule set.
- `size_t` **NumBaseCases** () const
Get the number of times a base case was calculated.
- `size_t` & **NumBaseCases** ()
Modify the number of times a base case was calculated.
- `size_t` **NumPrunes** () const
Get the number of prunes.
- `size_t` & **NumPrunes** ()
Modify the number of prunes.
- `size_t` **NumScores** () const
Get the number of times a node combination was scored.
- `size_t` & **NumScores** ()
Modify the number of times a node combination was scored.
- `size_t` **NumVisited** () const
Get the number of visited combinations.
- `size_t` & **NumVisited** ()
Modify the number of visited combinations.
- `void` **Traverse** (`RectangleTree` &queryNode, `RectangleTree` &referenceNode)
Traverse the two trees.

39.529.1 Detailed Description

```
template<typename MetricType = metric::EuclideanDistance, typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = RTreeSplit, typename DescentType = RTreeDescentHeuristic, template< typename > class AuxiliaryInformationType = NoAuxiliaryInformation>
template<typename MetricType, typename StatisticType, typename MatType, typename SplitType, typename DescentType, template< typename > class AuxiliaryInformationType>
class mlpack::tree::RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::DualTreeTraverser< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >
```

A dual tree traverser for rectangle type trees.

Definition at line 31 of file `dual_tree_traverser.hpp`.

39.529.2 Constructor & Destructor Documentation

39.529.2.1 DualTreeTraverser()

```
DualTreeTraverser (  
    RuleType & rule )
```

Instantiate the dual-tree traverser with the given rule set.

39.529.3 Member Function Documentation

39.529.3.1 NumBaseCases() [1/2]

```
size_t NumBaseCases ( ) const [inline]
```

Get the number of times a base case was calculated.

Definition at line 65 of file dual_tree_traverser.hpp.

39.529.3.2 NumBaseCases() [2/2]

```
size_t& NumBaseCases ( ) [inline]
```

Modify the number of times a base case was calculated.

Definition at line 67 of file dual_tree_traverser.hpp.

39.529.3.3 NumPrunes() [1/2]

```
size_t NumPrunes ( ) const [inline]
```

Get the number of prunes.

Definition at line 50 of file dual_tree_traverser.hpp.

39.529.3.4 NumPrunes() [2/2]

```
size_t& NumPrunes ( ) [inline]
```

Modify the number of prunes.

Definition at line 52 of file dual_tree_traverser.hpp.

39.529.3.5 NumScores() [1/2]

```
size_t NumScores ( ) const [inline]
```

Get the number of times a node combination was scored.

Definition at line 60 of file dual_tree_traverser.hpp.

39.529.3.6 NumScores() [2/2]

```
size_t& NumScores ( ) [inline]
```

Modify the number of times a node combination was scored.

Definition at line 62 of file dual_tree_traverser.hpp.

39.529.3.7 NumVisited() [1/2]

```
size_t NumVisited ( ) const [inline]
```

Get the number of visited combinations.

Definition at line 55 of file dual_tree_traverser.hpp.

39.529.3.8 NumVisited() [2/2]

```
size_t& NumVisited ( ) [inline]
```

Modify the number of visited combinations.

Definition at line 57 of file dual_tree_traverser.hpp.

39.529.3.9 Traverse()

```
void Traverse (
    RectangleTree & queryNode,
    RectangleTree & referenceNode )
```

Traverse the two trees.

This does not reset the number of prunes.

Parameters

<i>queryNode</i>	The query node to be traversed.
<i>referenceNode</i>	The reference node to be traversed.
<i>score</i>	The score of the current node combination.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ **dual_tree_traverser.hpp**

39.530 **RectangleTree**< **MetricType**, **StatisticType**, **MatType**, **SplitType**, **DescentType**, **AuxiliaryInformationType** >::**SingleTreeTraverser**< **RuleType** > **Class Template Reference**

A single traverser for rectangle type trees.

Public Member Functions

- **SingleTreeTraverser** (**RuleType** &rule)
Instantiate the traverser with the given rule set.
- size_t **NumPrunes** () const
Get the number of prunes.
- size_t & **NumPrunes** ()
Modify the number of prunes.
- void **Traverse** (const size_t queryIndex, const **RectangleTree** &referenceNode)
Traverse the tree with the given point.

39.530.1 Detailed Description

```
template<typename MetricType = metric::EuclideanDistance, typename StatisticType = EmptyStatistic, typename MatType = arma::mat, typename SplitType = RTreeSplit, typename DescentType = RTreeDescentHeuristic, template< typename > class AuxiliaryInformationType = NoAuxiliaryInformation>
template<typename RuleType>
class mlpack::tree::RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::SingleTreeTraverser< RuleType >
```

A single traverser for rectangle type trees.

See single_tree_traverser.hpp for implementation.

Definition at line 112 of file rectangle_tree.hpp.

39.530.2 Constructor & Destructor Documentation

39.530.2.1 SingleTreeTraverser()

```
SingleTreeTraverser (  
    RuleType & rule )
```

Instantiate the traverser with the given rule set.

39.530.3 Member Function Documentation

39.530.3.1 NumPrunes() [1/2]

```
size_t NumPrunes ( ) const [inline]
```

Get the number of prunes.

Definition at line 50 of file `single_tree_traverser.hpp`.

39.530.3.2 NumPrunes() [2/2]

```
size_t& NumPrunes ( ) [inline]
```

Modify the number of prunes.

Definition at line 52 of file `single_tree_traverser.hpp`.

39.530.3.3 Traverse()

```
void Traverse (  
    const size_t queryIndex,  
    const RectangleTree & referenceNode )
```

Traverse the tree with the given point.

Parameters

<i>queryIndex</i>	The index of the point in the query set which is being used as the query point.
<i>referenceNode</i>	The tree node to be traversed.

The documentation for this class was generated from the following files:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ **rectangle_tree.hpp**
- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ **single_tree_traverser.hpp**

39.531 RPlusPlusTreeAuxiliaryInformation< TreeType > Class Template Reference

Public Types

- typedef **bound::HRectBound< metric::EuclideanDistance, ElemType > BoundType**
The bound type held by the auxiliary information.
- typedef TreeType::ElemType **ElemType**
The element type held by the tree.

Public Member Functions

- **RPlusPlusTreeAuxiliaryInformation ()**
Construct the auxiliary information object.
- **RPlusPlusTreeAuxiliaryInformation (const TreeType *)**
Construct this as an auxiliary information for the given node.
- **RPlusPlusTreeAuxiliaryInformation (const RPlusPlusTreeAuxiliaryInformation &other, TreeType *tree, bool=true)**
Create an auxiliary information object by copying from another object.
- **RPlusPlusTreeAuxiliaryInformation (RPlusPlusTreeAuxiliaryInformation &&other)**
Create an auxiliary information object by moving from another node.
- bool **HandleNodeInsertion** (TreeType *, TreeType *, bool)
Some tree types require to save some properties at the insertion process.
- bool **HandleNodeRemoval** (TreeType *, const size_t)
Some tree types require to save some properties at the deletion process.
- bool **HandlePointDeletion** (TreeType *, const size_t)
Some tree types require to save some properties at the deletion process.
- bool **HandlePointInsertion** (TreeType *, const size_t)
Some tree types require to save some properties at the insertion process.
- void **NullifyData ()**
Nullify the auxiliary information in order to prevent an invalid free.
- **BoundType & OuterBound ()**
Return the maximum bounding rectangle.
- const **BoundType & OuterBound () const**
Modify the maximum bounding rectangle.

- `template<typename Archive >`
`void serialize (Archive &, const unsigned int)`
Serialize the information.
- `void SplitAuxiliaryInfo (TreeType *treeOne, TreeType *treeTwo, const size_t axis, const ElemType cut)`
The R++ tree requires to split the maximum bounding rectangle of a node that is being split.
- `bool UpdateAuxiliaryInfo (TreeType *)`
Some tree types require to propagate the information upward.

39.531.1 Detailed Description

```
template<typename TreeType>
class mpack::tree::RPlusPlusTreeAuxiliaryInformation< TreeType >
```

Definition at line 24 of file `r_plus_plus_tree_auxiliary_information.hpp`.

39.531.2 Member Typedef Documentation

39.531.2.1 BoundType

```
typedef bound::HRectBound< metric::EuclideanDistance, ElemType> BoundType
```

The bound type held by the auxiliary information.

Definition at line 30 of file `r_plus_plus_tree_auxiliary_information.hpp`.

39.531.2.2 ElemType

```
typedef TreeType::ElemType ElemType
```

The element type held by the tree.

Definition at line 28 of file `r_plus_plus_tree_auxiliary_information.hpp`.

39.531.3 Constructor & Destructor Documentation

39.531.3.1 RPlusPlusTreeAuxiliaryInformation() [1/4]

```
RPlusPlusTreeAuxiliaryInformation ( )
```

Construct the auxiliary information object.

39.531.3.2 RPlusPlusTreeAuxiliaryInformation() [2/4]

```
RPlusPlusTreeAuxiliaryInformation (
    const TreeType * )
```

Construct this as an auxiliary information for the given node.

Parameters

<i>node</i>	The node that stores this auxiliary information.
-------------	--

39.531.3.3 `RPlusPlusTreeAuxiliaryInformation()` [3/4]

```
RPlusPlusTreeAuxiliaryInformation (
    const RPlusPlusTreeAuxiliaryInformation< TreeType > & other,
    TreeType * tree,
    bool    = true )
```

Create an auxiliary information object by copying from another object.

Parameters

<i>other</i>	Another auxiliary information object from which the information will be copied.
<i>tree</i>	The node that holds the auxiliary information.
<i>deepCopy</i>	If false, the new object uses the same memory (not used here).

39.531.3.4 `RPlusPlusTreeAuxiliaryInformation()` [4/4]

```
RPlusPlusTreeAuxiliaryInformation (
    RPlusPlusTreeAuxiliaryInformation< TreeType > && other )
```

Create an auxiliary information object by moving from another node.

Parameters

<i>other</i>	The auxiliary information object from which the information will be moved.
--------------	--

39.531.4 Member Function Documentation

39.531.4.1 `HandleNodeInsertion()`

```
bool HandleNodeInsertion (
    TreeType * ,
```

```
TreeType * ,
bool )
```

Some tree types require to save some properties at the insertion process.

This method allows the auxiliary information the option of manipulating the tree in order to perform the insertion process. If the auxiliary information does that, then the method should return true; if the method returns false the **RectangleTree** (p. 2207) performs its default behavior.

Parameters

<i>node</i>	The node in which the nodeToInsert is being inserted.
<i>nodeToInsert</i>	The node being inserted.
<i>insertionLevel</i>	The level of the tree at which the nodeToInsert should be inserted.

39.531.4.2 HandleNodeRemoval()

```
bool HandleNodeRemoval (
    TreeType * ,
    const size_t )
```

Some tree types require to save some properties at the deletion process.

This method allows the auxiliary information the option of manipulating the tree in order to perform the deletion process. If the auxiliary information does that, then the method should return true; if the method returns false the **RectangleTree** (p. 2207) performs its default behavior.

Parameters

<i>node</i>	The node from which the node is being deleted.
<i>nodeIndex</i>	The local index of the node being deleted.

39.531.4.3 HandlePointDeletion()

```
bool HandlePointDeletion (
    TreeType * ,
    const size_t )
```

Some tree types require to save some properties at the deletion process.

This method allows the auxiliary information the option of manipulating the tree in order to perform the deletion process. If the auxiliary information does that, then the method should return true; if the method returns false the **RectangleTree** (p. 2207) performs its default behavior.

Parameters

<i>node</i>	The node from which the point is being deleted.
<i>localIndex</i>	The local index of the point being deleted.

39.531.4.4 **HandlePointInsertion()**

```
bool HandlePointInsertion (
    TreeType * ,
    const size_t )
```

Some tree types require to save some properties at the insertion process.

This method allows the auxiliary information the option of manipulating the tree in order to perform the insertion process. If the auxiliary information does that, then the method should return true; if the method returns false the **RectangleTree** (p. 2207) performs its default behavior.

Parameters

<i>node</i>	The node in which the point is being inserted.
<i>point</i>	The global number of the point being inserted.

39.531.4.5 **NullifyData()**

```
void NullifyData ( )
```

Nullify the auxiliary information in order to prevent an invalid free.

39.531.4.6 **OuterBound()** [1/2]

```
BoundType& OuterBound ( ) [inline]
```

Return the maximum bounding rectangle.

Definition at line 146 of file `r_plus_plus_tree_auxiliary_information.hpp`.

39.531.4.7 OuterBound() [2/2]

```
const BoundType& OuterBound ( ) const [inline]
```

Modify the maximum bounding rectangle.

Definition at line 149 of file r_plus_plus_tree_auxiliary_information.hpp.

References RPlusPlusTreeAuxiliaryInformation< TreeType >::serialize().

39.531.4.8 serialize()

```
void serialize (
    Archive & ,
    const unsigned int )
```

Serialize the information.

Referenced by RPlusPlusTreeAuxiliaryInformation< TreeType >::OuterBound().

39.531.4.9 SplitAuxiliaryInfo()

```
void SplitAuxiliaryInfo (
    TreeType * treeOne,
    TreeType * treeTwo,
    const size_t axis,
    const ElemType cut )
```

The R++ tree requires to split the maximum bounding rectangle of a node that is being split.

This method is intended for that.

Parameters

<i>treeOne</i>	The first subtree.
<i>treeTwo</i>	The second subtree.
<i>axis</i>	The axis along which the split is performed.
<i>cut</i>	The coordinate at which the node is split.

39.531.4.10 UpdateAuxiliaryInfo()

```
bool UpdateAuxiliaryInfo (
    TreeType * )
```

Some tree types require to propagate the information upward.

This method should return false if this is not the case. If true is returned, the update will be propagated upward.

Parameters

<i>node</i>	The node in which the auxiliary information being update.
-------------	---

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/
information.hpp [r_plus_plus_tree_auxiliary_↔](#)

39.532 RPlusPlusTreeDescentHeuristic Class Reference**Static Public Member Functions**

- template<typename TreeType >
static size_t **ChooseDescentNode** (TreeType *node, const size_t point)
Evaluate the node using a heuristic.
- template<typename TreeType >
static size_t **ChooseDescentNode** (const TreeType *node, const TreeType *insertedNode)
Evaluate the node using a heuristic.

39.532.1 Detailed Description

Definition at line 21 of file r_plus_plus_tree_descent_heuristic.hpp.

39.532.2 Member Function Documentation**39.532.2.1 ChooseDescentNode() [1/2]**

```
static size_t ChooseDescentNode (
    TreeType * node,
    const size_t point ) [static]
```

Evaluate the node using a heuristic.

Returns the number of the node with minimum largest Hilbert value is greater than the Hilbert value of the point being inserted.

Parameters

<i>node</i>	The node that is being evaluated.
<i>point</i>	The number of the point that is being inserted.

39.532.2.2 ChooseDescentNode() [2/2]

```
static size_t ChooseDescentNode (
    const TreeType * node,
    const TreeType * insertedNode ) [static]
```

Evaluate the node using a heuristic.

Returns the number of the node with minimum largest Hilbert value is greater than the largest Hilbert value of the point being inserted.

Parameters

<i>node</i>	The node that is being evaluated.
<i>insertedNode</i>	The node that is being inserted.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/` **r_plus_plus_tree_descent_**
heuristic.hpp

39.533 RPlusPlusTreeSplitPolicy Class Reference

The **RPlusPlusTreeSplitPolicy** (p.2247) helps to determine the subtree into which we should insert a child of an intermediate node that is being split.

Static Public Member Functions

- `template<typename TreeType >`
static const **bound::HRectBound**< **metric::EuclideanDistance**, typename TreeType::ElemType > & **Bound**
(const TreeType &node)
Return the maximum bounding rectangle of the node.
- `template<typename TreeType >`
static int **GetSplitPolicy** (const TreeType &child, const size_t axis, const typename TreeType::ElemType cut)
This method returns SplitRequired if a child of an intermediate node should be split, AssignToFirstTree if the child should be inserted to the first subtree, AssignToSecondTree if the child should be inserted to the second subtree.

Static Public Attributes

- static const int **AssignToFirstTree** = 1
Indicate that the child should be inserted to the first subtree.
- static const int **AssignToSecondTree** = 2
Indicate that the child should be inserted to the second subtree.
- static const int **SplitRequired** = 0
Indicate that the child should be split.

39.533.1 Detailed Description

The **RPlusPlusTreeSplitPolicy** (p.2247) helps to determine the subtree into which we should insert a child of an intermediate node that is being split.

This class is designed for the R++ tree.

Definition at line 25 of file `r_plus_plus_tree_split_policy.hpp`.

39.533.2 Member Function Documentation

39.533.2.1 Bound()

```
static const bound::HRectBound< metric::EuclideanDistance, typename TreeType::ElemType>& Bound (
    const TreeType & node ) [inline], [static]
```

Return the maximum bounding rectangle of the node.

This method should always return the bound that is used for the decision-making in **GetSplitPolicy()** (p. 2248).

Parameters

<i>node</i>	The node whose bound is requested.
-------------	------------------------------------

Definition at line 70 of file `r_plus_plus_tree_split_policy.hpp`.

39.533.2.2 GetSplitPolicy()

```
static int GetSplitPolicy (
    const TreeType & child,
```



```
const size_t axis,
const typename TreeType::ElemType cut ) [inline], [static]
```

This method returns `SplitRequired` if a child of an intermediate node should be split, `AssignToFirstTree` if the child should be inserted to the first subtree, `AssignToSecondTree` if the child should be inserted to the second subtree.

The method makes decision according to the maximum bounding rectangle of the child, the axis along which the intermediate node is being split and the coordinate at which the node is being split.

Parameters

<i>child</i>	A child of the node that is being split.
<i>axis</i>	The axis along which the node is being split.
<i>cut</i>	The coordinate at which the node is being split.

Definition at line 48 of file `r_plus_plus_tree_split_policy.hpp`.

References `RPlusPlusTreeSplitPolicy::SplitRequired`.

39.533.3 Member Data Documentation

39.533.3.1 AssignToFirstTree

```
const int AssignToFirstTree = 1 [static]
```

Indicate that the child should be inserted to the first subtree.

Definition at line 31 of file `r_plus_plus_tree_split_policy.hpp`.

39.533.3.2 AssignToSecondTree

```
const int AssignToSecondTree = 2 [static]
```

Indicate that the child should be inserted to the second subtree.

Definition at line 33 of file `r_plus_plus_tree_split_policy.hpp`.

39.533.3.3 SplitRequired

```
const int SplitRequired = 0 [static]
```

Indicate that the child should be split.

Definition at line 29 of file `r_plus_plus_tree_split_policy.hpp`.

Referenced by `RPlusPlusTreeSplitPolicy::GetSplitPolicy()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/r_plus_plus_tree_split_policy.hpp`

39.534 RPlusTreeDescentHeuristic Class Reference

Static Public Member Functions

- `template<typename TreeType >`
`static size_t ChooseDescentNode (TreeType *node, const size_t point)`
Evaluate the node using a heuristic.
- `template<typename TreeType >`
`static size_t ChooseDescentNode (const TreeType *, const TreeType *)`
Evaluate the node using a heuristic.

39.534.1 Detailed Description

Definition at line 21 of file `r_plus_tree_descent_heuristic.hpp`.

39.534.2 Member Function Documentation

39.534.2.1 ChooseDescentNode() [1/2]

```
static size_t ChooseDescentNode (
    TreeType * node,
    const size_t point ) [static]
```

Evaluate the node using a heuristic.

Returns the number of the node with minimum largest Hilbert value is greater than the Hilbert value of the point being inserted.

Parameters

<i>node</i>	The node that is being evaluated.
<i>point</i>	The number of the point that is being inserted.

39.534.2.2 ChooseDescentNode() [2/2]

```
static size_t ChooseDescentNode (
    const TreeType * ,
    const TreeType * ) [static]
```

Evaluate the node using a heuristic.

Returns the number of the node with minimum largest Hilbert value is greater than the largest Hilbert value of the point being inserted.

Parameters

<i>node</i>	The node that is being evaluated.
<i>insertedNode</i>	The node that is being inserted.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ **r_plus_tree_descent_heuristic.**↵
hpp

39.535 RPlusTreeSplit< SplitPolicyType, SweepType > Class Template Reference

The **RPlusTreeSplit** (p. 2251) class performs the split process of a node on overflow.

Public Types

- typedef SplitPolicyType **SplitPolicy**

Static Public Member Functions

- template<typename TreeType >
static void **SplitLeafNode** (TreeType *tree, std::vector< bool > &relevels)
Split a leaf node using the "default" algorithm.
- template<typename TreeType >
static bool **SplitNonLeafNode** (TreeType *tree, std::vector< bool > &relevels)
Split a non-leaf node using the "default" algorithm.

39.535.1 Detailed Description

```
template<typename SplitPolicyType, template< typename > class SweepType>
class mlpack::tree::RPlusTreeSplit< SplitPolicyType, SweepType >
```

The **RPlusTreeSplit** (p. 2251) class performs the split process of a node on overflow.

Template Parameters

<i>SplitPolicyType</i>	The class that helps to determine the subtree into which we should insert a child node.
<i>SweepType</i>	The class that finds the partition of a node along a given axis. The partition algorithm tries to find a partition along each axis, evaluates each partition and chooses the best one.

Definition at line 32 of file `r_plus_tree_split.hpp`.

39.535.2 Member Typedef Documentation

39.535.2.1 SplitPolicy

```
typedef SplitPolicyType SplitPolicy
```

Definition at line 35 of file `r_plus_tree_split.hpp`.

39.535.3 Member Function Documentation

39.535.3.1 SplitLeafNode()

```
static void SplitLeafNode (
    TreeType * tree,
    std::vector< bool > & relevels ) [static]
```

Split a leaf node using the "default" algorithm.

If necessary, this split will propagate upwards through the tree.

Parameters

<i>node.</i>	The node that is being split.
<i>relevels</i>	Not used.

39.535.3.2 SplitNonLeafNode()

```
static bool SplitNonLeafNode (
    TreeType * tree,
    std::vector< bool > & relevels ) [static]
```

Split a non-leaf node using the "default" algorithm.

If this is a root node, the tree increases in depth.

Parameters

<i>node.</i>	The node that is being split.
<i>relevels</i>	Not used.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ **r_plus_tree_split.hpp**

39.536 RPlusTreeSplitPolicy Class Reference

The **RPlusTreeSplitPolicy** (p.2247) helps to determine the subtree into which we should insert a child of an intermediate node that is being split.

Static Public Member Functions

- template<typename TreeType >
static const **bound::HRectBound**< **metric::EuclideanDistance**, typename TreeType::ElemType > & **Bound**
(const TreeType &node)
Return the minimum bounding rectangle of the node.
- template<typename TreeType >
static int **GetSplitPolicy** (const TreeType &child, const size_t axis, const typename TreeType::ElemType cut)
This method returns SplitRequired if a child of an intermediate node should be split, AssignToFirstTree if the child should be inserted to the first subtree, AssignToSecondTree if the child should be inserted to the second subtree.

Static Public Attributes

- static const int **AssignToFirstTree** = 1
Indicate that the child should be inserted to the first subtree.
- static const int **AssignToSecondTree** = 2
Indicate that the child should be inserted to the second subtree.
- static const int **SplitRequired** = 0
Indicate that the child should be split.

39.536.1 Detailed Description

The **RPlusPlusTreeSplitPolicy** (p.2247) helps to determine the subtree into which we should insert a child of an intermediate node that is being split.

This class is designed for the R+ tree.

Definition at line 25 of file `r_plus_tree_split_policy.hpp`.

39.536.2 Member Function Documentation

39.536.2.1 Bound()

```
static const bound::HRectBound< metric::EuclideanDistance, typename TreeType::ElemType>& Bound (
    const TreeType & node ) [inline], [static]
```

Return the minimum bounding rectangle of the node.

This method should always return the bound that is used for the decision-making in **GetSplitPolicy()** (p. 2254).

Parameters

<i>node</i>	The node whose bound is requested.
-------------	------------------------------------

Definition at line 70 of file `r_plus_tree_split_policy.hpp`.

39.536.2.2 GetSplitPolicy()

```
static int GetSplitPolicy (
    const TreeType & child,
    const size_t axis,
    const typename TreeType::ElemType cut ) [inline], [static]
```

This method returns `SplitRequired` if a child of an intermediate node should be split, `AssignToFirstTree` if the child should be inserted to the first subtree, `AssignToSecondTree` if the child should be inserted to the second subtree.

The method makes desicion according to the minimum bounding rectangle of the child, the axis along which the intermediate node is being split and the coordinate at which the node is being split.

Parameters

<i>child</i>	A child of the node that is being split.
<i>axis</i>	The axis along which the node is being split.
<i>cut</i>	The coordinate at which the node is being split.

Definition at line 48 of file r_plus_tree_split_policy.hpp.

References RPlusTreeSplitPolicy::SplitRequired.

39.536.3 Member Data Documentation

39.536.3.1 AssignToFirstTree

```
const int AssignToFirstTree = 1 [static]
```

Indicate that the child should be inserted to the first subtree.

Definition at line 31 of file r_plus_tree_split_policy.hpp.

39.536.3.2 AssignToSecondTree

```
const int AssignToSecondTree = 2 [static]
```

Indicate that the child should be inserted to the second subtree.

Definition at line 33 of file r_plus_tree_split_policy.hpp.

39.536.3.3 SplitRequired

```
const int SplitRequired = 0 [static]
```

Indicate that the child should be split.

Definition at line 29 of file r_plus_tree_split_policy.hpp.

Referenced by RPlusTreeSplitPolicy::GetSplitPolicy().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ r_plus_tree_split_policy.hpp

39.537 RPTreeMaxSplit< BoundType, MatType > Class Template Reference

This class splits a node by a random hyperplane.

Classes

- struct **SplitInfo**
An information about the partition.

Public Types

- typedef MatType::elem_type **ElemType**
The element type held by the matrix type.

Static Public Member Functions

- template<typename VecType >
static bool **AssignToLeftNode** (const VecType &point, const **SplitInfo** &splitInfo)
Indicates that a point should be assigned to the left subtree.
- static size_t **PerformSplit** (MatType &data, const size_t begin, const size_t count, const **SplitInfo** &splitInfo)
Perform the split process according to the information about the split.
- static size_t **PerformSplit** (MatType &data, const size_t begin, const size_t count, const **SplitInfo** &splitInfo, std::vector< size_t > &oldFromNew)
Perform the split process according to the information about the split and return the list of changed indices.
- static bool **SplitNode** (const BoundType &, MatType &data, const size_t begin, const size_t count, **SplitInfo** &splitInfo)
Split the node by a random hyperplane.

39.537.1 Detailed Description

```
template<typename BoundType, typename MatType = arma::mat>
class mpack::tree::RPTreeMaxSplit< BoundType, MatType >
```

This class splits a node by a random hyperplane.

In order to choose the hyperplane we need to choose the normal to the hyperplane and the position of the hyperplane i.e. the scalar product of the normal and a point.

A point will be assigned to the left subtree if the product of this point and the normal is less or equal to the split value (i.e. the position of the hyperplane).

Definition at line 32 of file rp_tree_max_split.hpp.

39.537.2 Member Typedef Documentation

39.537.2.1 ElemType

```
typedef MatType::elem_type ElemType
```

The element type held by the matrix type.

Definition at line 36 of file `rp_tree_max_split.hpp`.

39.537.3 Member Function Documentation

39.537.3.1 AssignToLeftNode()

```
static bool AssignToLeftNode (
    const VecType & point,
    const SplitInfo & splitInfo ) [inline], [static]
```

Indicates that a point should be assigned to the left subtree.

Parameters

<i>point</i>	The point that is being assigned.
<i>splitInfo</i>	An information about the split.

Definition at line 118 of file `rp_tree_max_split.hpp`.

References `RPTreeMaxSplit< BoundType, MatType >::SplitInfo::direction`, and `RPTreeMaxSplit< BoundType, MatType >::SplitInfo::splitVal`.

39.537.3.2 PerformSplit() [1/2]

```
static size_t PerformSplit (
    MatType & data,
    const size_t begin,
    const size_t count,
    const SplitInfo & splitInfo ) [inline], [static]
```

Perform the split process according to the information about the split.

This will order the dataset such that points that belong to the left subtree are on the left of the split column, and points from the right subtree are on the right side of the split column.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitInfo</i>	The information about the split.

Definition at line 76 of file `rp_tree_max_split.hpp`.

39.537.3.3 PerformSplit() [2/2]

```
static size_t PerformSplit (
    MatType & data,
    const size_t begin,
    const size_t count,
    const SplitInfo & splitInfo,
    std::vector< size_t > & oldFromNew ) [inline], [static]
```

Perform the split process according to the information about the split and return the list of changed indices.

This will order the dataset such that points that belong to the left subtree are on the left of the split column, and points from the right subtree are on the right side of the split column.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitInfo</i>	The information about the split.
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.

Definition at line 101 of file `rp_tree_max_split.hpp`.

39.537.3.4 SplitNode()

```
static bool SplitNode (
    const BoundType & ,
    MatType & data,
    const size_t begin,
```

```
const size_t count,  
    SplitInfo & splitInfo ) [static]
```

Split the node by a random hyperplane.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitInfo</i>	An information about the split. This information contains the direction and the value.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ **rp_tree_max_split.hpp**

39.538 RPTreeMaxSplit< BoundType, MatType >::SplitInfo Struct Reference

An information about the partition.

Public Attributes

- arma::Col< **ElemType** > **direction**
The normal vector to the hyperplane that splits the node.
- **ElemType** **splitVal**
The value according to which the node is being split.

39.538.1 Detailed Description

```
template<typename BoundType, typename MatType = arma::mat>
struct mlpack::tree::RPTreeMaxSplit< BoundType, MatType >::SplitInfo
```

An information about the partition.

Definition at line 38 of file rp_tree_max_split.hpp.

39.538.2 Member Data Documentation

39.538.2.1 direction

```
arma::Col< ElemType> direction
```

The normal vector to the hyperplane that splits the node.

Definition at line 41 of file rp_tree_max_split.hpp.

Referenced by RPTreeMaxSplit< BoundType, MatType >::AssignToLeftNode().

39.538.2.2 splitVal

ElemType splitVal

The value according to which the node is being split.

Definition at line 43 of file rp_tree_max_split.hpp.

Referenced by RPTreeMaxSplit< BoundType, MatType >::AssignToLeftNode().

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ **rp_tree_max_split.hpp**

39.539 RPTreeMeanSplit< BoundType, MatType > Class Template Reference

This class splits a binary space tree.

Classes

- struct **SplitInfo**
An information about the partition.

Public Types

- typedef MatType::elem_type **ElemType**
The element type held by the matrix type.

Static Public Member Functions

- template<typename VecType >
static bool **AssignToLeftNode** (const VecType &point, const **SplitInfo** &splitInfo)
Indicates that a point should be assigned to the left subtree.
- static size_t **PerformSplit** (MatType &data, const size_t begin, const size_t count, const **SplitInfo** &splitInfo)
Perform the split process according to the information about the split.
- static size_t **PerformSplit** (MatType &data, const size_t begin, const size_t count, const **SplitInfo** &splitInfo, std::vector< size_t > &oldFromNew)
Perform the split process according to the information about the split and return the list of changed indices.
- static bool **SplitNode** (const BoundType &, MatType &data, const size_t begin, const size_t count, **SplitInfo** &splitInfo)
Split the node according to the mean value in the dimension with maximum width.

39.539.1 Detailed Description

```
template<typename BoundType, typename MatType = arma::mat>
class mlpack::tree::RPTreeMeanSplit< BoundType, MatType >
```

This class splits a binary space tree.

This class provides two different kinds of split. The mean split (i.e. all points are split by the median of their distance to the mean point) is performed if the average distance between points multiplied by a constant is greater than the diameter of the node. Otherwise, the median split (i.e. the node is split by a random hyperplane) is performed.

Definition at line 33 of file `rp_tree_mean_split.hpp`.

39.539.2 Member Typedef Documentation

39.539.2.1 ElemType

```
typedef MatType::elem_type ElemType
```

The element type held by the matrix type.

Definition at line 37 of file `rp_tree_mean_split.hpp`.

39.539.3 Member Function Documentation

39.539.3.1 AssignToLeftNode()

```
static bool AssignToLeftNode (
    const VecType & point,
    const SplitInfo & splitInfo ) [inline], [static]
```

Indicates that a point should be assigned to the left subtree.

Parameters

<i>point</i>	The point that is being assigned.
<i>splitInfo</i>	An information about the split.

Definition at line 125 of file `rp_tree_mean_split.hpp`.

References RPTreeMeanSplit< BoundType, MatType >::SplitInfo::direction, RPTreeMeanSplit< BoundType, MatType >::SplitInfo::mean, RPTreeMeanSplit< BoundType, MatType >::SplitInfo::meanSplit, and RPTreeMeanSplit< BoundType, MatType >::SplitInfo::splitVal.

39.539.3.2 PerformSplit() [1/2]

```
static size_t PerformSplit (
    MatType & data,
    const size_t begin,
    const size_t count,
    const SplitInfo & splitInfo ) [inline], [static]
```

Perform the split process according to the information about the split.

This will order the dataset such that points that belong to the left subtree are on the left of the split column, and points from the right subtree are on the right side of the split column.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitInfo</i>	The information about the split.

Definition at line 83 of file `rp_tree_mean_split.hpp`.

39.539.3.3 PerformSplit() [2/2]

```
static size_t PerformSplit (
    MatType & data,
    const size_t begin,
    const size_t count,
    const SplitInfo & splitInfo,
    std::vector< size_t > & oldFromNew ) [inline], [static]
```

Perform the split process according to the information about the split and return the list of changed indices.

This will order the dataset such that points that belong to the left subtree are on the left of the split column, and points from the right subtree are on the right side of the split column.

Parameters

<i>bound</i>	The bound used for this node.
--------------	-------------------------------

Parameters

<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitInfo</i>	The information about the split.
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.

Definition at line 108 of file `rp_tree_mean_split.hpp`.

39.539.3.4 SplitNode()

```
static bool SplitNode (
    const BoundType & ,
    MatType & data,
    const size_t begin,
    const size_t count,
    SplitInfo & splitInfo ) [static]
```

Split the node according to the mean value in the dimension with maximum width.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitInfo</i>	An information about the split. This information contains the direction and the value.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ rp_tree_mean_split.hpp`

39.540 RPTreeMeanSplit< BoundType, MatType >::SplitInfo Struct Reference

An information about the partition.

Public Attributes

- `arma::Col< ElemType > direction`
The normal to the hyperplane that will split the node.

- `arma::Col< ElemType > mean`
The mean of some sampled points.
- `bool meanSplit`
Indicates that we should use the mean split algorithm instead of the median split.
- `ElemType splitVal`
The value according to which the split will be performed.

39.540.1 Detailed Description

```
template<typename BoundType, typename MatType = arma::mat>
struct mlpack::tree::RPTreeMeanSplit< BoundType, MatType >::SplitInfo
```

An information about the partition.

Definition at line 39 of file `rp_tree_mean_split.hpp`.

39.540.2 Member Data Documentation

39.540.2.1 direction

```
arma::Col< ElemType> direction
```

The normal to the hyperplane that will split the node.

Definition at line 42 of file `rp_tree_mean_split.hpp`.

Referenced by `RPTreeMeanSplit< BoundType, MatType >::AssignToLeftNode()`.

39.540.2.2 mean

```
arma::Col< ElemType> mean
```

The mean of some sampled points.

Definition at line 44 of file `rp_tree_mean_split.hpp`.

Referenced by `RPTreeMeanSplit< BoundType, MatType >::AssignToLeftNode()`.

39.540.2.3 meanSplit

```
bool meanSplit
```

Indicates that we should use the mean split algorithm instead of the median split.

Definition at line 49 of file `rp_tree_mean_split.hpp`.

Referenced by `RPTreeMeanSplit< BoundType, MatType >::AssignToLeftNode()`.

39.540.2.4 splitVal

```
ElemType splitVal
```

The value according to which the split will be performed.

Definition at line 46 of file `rp_tree_mean_split.hpp`.

Referenced by `RPTreeMeanSplit< BoundType, MatType >::AssignToLeftNode()`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ rp_tree_mean_split.hpp`

39.541 RStarTreeDescentHeuristic Class Reference

When descending a **RectangleTree** (p. 2207) to insert a point, we need to have a way to choose a child node when the point isn't enclosed by any of them.

Static Public Member Functions

- `template<typename TreeType >`
`static size_t ChooseDescentNode (const TreeType *node, const size_t point)`
Evaluate the node using a heuristic.
- `template<typename TreeType >`
`static size_t ChooseDescentNode (const TreeType *node, const TreeType *insertedNode)`

39.541.1 Detailed Description

When descending a **RectangleTree** (p. 2207) to insert a point, we need to have a way to choose a child node when the point isn't enclosed by any of them.

This heuristic is used to do so using the rules for the R* tree.

Definition at line 26 of file `r_star_tree_descent_heuristic.hpp`.

39.541.2 Member Function Documentation

39.541.2.1 ChooseDescentNode() [1/2]

```
static size_t ChooseDescentNode (  
    const TreeType * node,  
    const size_t point ) [static]
```

Evaluate the node using a heuristic.

The heuristic guarantees two things:

1. If point is contained in (or on) bound, the value returned is zero.
2. If the point is not contained in (or on) bound, the value returned is greater than zero.

Parameters

<i>bound</i>	The bound used for the node that is being evaluated.
<i>point</i>	The index of the point that is being inserted.

39.541.2.2 ChooseDescentNode() [2/2]

```
static size_t ChooseDescentNode (  
    const TreeType * node,  
    const TreeType * insertedNode ) [static]
```

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ **r_star_tree_descent_heuristic.hpp**

39.542 RStarTreeSplit Class Reference

A Rectangle Tree has new points inserted at the bottom.

Static Public Member Functions

- `template<typename TreeType >`
`static void PickLeafSplit (TreeType *tree, size_t &bestAxis, size_t &bestIndex)`
Given a node, return the best dimension and the best index to split on.
- `template<typename TreeType >`
`static size_t ReinsertPoints (TreeType *tree, std::vector< bool > &relevels)`
Reinsert any points into the tree, if needed.
- `template<typename TreeType >`
`static void SplitLeafNode (TreeType *tree, std::vector< bool > &relevels)`
Split a leaf node using the algorithm described in "The R-tree: An Efficient and Robust Access method for Points and Rectangles."*
- `template<typename TreeType >`
`static bool SplitNonLeafNode (TreeType *tree, std::vector< bool > &relevels)`
Split a non-leaf node using the "default" algorithm.

39.542.1 Detailed Description

A Rectangle Tree has new points inserted at the bottom.

When these nodes overflow, we split them, moving up the tree and splitting nodes as necessary.

Definition at line 26 of file `r_star_tree_split.hpp`.

39.542.2 Member Function Documentation

39.542.2.1 PickLeafSplit()

```
static void PickLeafSplit (
    TreeType * tree,
    size_t & bestAxis,
    size_t & bestIndex ) [static]
```

Given a node, return the best dimension and the best index to split on.

39.542.2.2 ReinsertPoints()

```
static size_t ReinsertPoints (
    TreeType * tree,
    std::vector< bool > & relevels ) [static]
```

Reinsert any points into the tree, if needed.

This returns the number of points reinserted.

39.542.2.3 SplitLeafNode()

```
static void SplitLeafNode (
    TreeType * tree,
    std::vector< bool > & relevels ) [static]
```

Split a leaf node using the algorithm described in "The R*-tree: An Efficient and Robust Access method for Points and Rectangles.

" If necessary, this split will propagate upwards through the tree.

39.542.2.4 SplitNonLeafNode()

```
static bool SplitNonLeafNode (
    TreeType * tree,
    std::vector< bool > & relevels ) [static]
```

Split a non-leaf node using the "default" algorithm.

If this is a root node, the tree increases in depth.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ **r_star_tree_split.hpp**

39.543 RTreeDescentHeuristic Class Reference

When descending a **RectangleTree** (p. 2207) to insert a point, we need to have a way to choose a child node when the point isn't enclosed by any of them.

Static Public Member Functions

- template<typename TreeType >
static size_t **ChooseDescentNode** (const TreeType *node, const size_t point)
Evaluate the node using a heuristic.
- template<typename TreeType >
static size_t **ChooseDescentNode** (const TreeType *node, const TreeType *insertedNode)
Evaluate the node using a heuristic.

39.543.1 Detailed Description

When descending a **RectangleTree** (p. 2207) to insert a point, we need to have a way to choose a child node when the point isn't enclosed by any of them.

This heuristic is used to do so.

Definition at line 26 of file r_tree_descent_heuristic.hpp.

39.543.2 Member Function Documentation

39.543.2.1 ChooseDescentNode() [1/2]

```
static size_t ChooseDescentNode (
    const TreeType * node,
    const size_t point ) [static]
```

Evaluate the node using a heuristic.

The heuristic guarantees two things:

1. If point is contained in (or on) the bound, the value returned is zero.
2. If the point is not contained in (or on) the bound, the value returned is greater than zero.

Parameters

<i>node</i>	The node that is being evaluated.
<i>point</i>	The index of the point that is being inserted.

39.543.2.2 ChooseDescentNode() [2/2]

```
static size_t ChooseDescentNode (
    const TreeType * node,
    const TreeType * insertedNode ) [static]
```

Evaluate the node using a heuristic.

The heuristic guarantees two things:

1. If point is contained in (or on) the bound, the value returned is zero.
2. If the point is not contained in (or on) the bound, the value returned is greater than zero.

Parameters

<i>node</i>	The node that is being evaluated.
<i>insertedNode</i>	The node that is being inserted.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ **r_tree_descent_heuristic.hpp**

39.544 RTreeSplit Class Reference

A Rectangle Tree has new points inserted at the bottom.

Static Public Member Functions

- template<typename TreeType >
static void **SplitLeafNode** (TreeType *tree, std::vector< bool > &relevels)
Split a leaf node using the "default" algorithm.
- template<typename TreeType >
static bool **SplitNonLeafNode** (TreeType *tree, std::vector< bool > &relevels)
Split a non-leaf node using the "default" algorithm.

39.544.1 Detailed Description

A Rectangle Tree has new points inserted at the bottom.

When these nodes overflow, we split them, moving up the tree and splitting nodes as necessary.

Definition at line 26 of file r_tree_split.hpp.

39.544.2 Member Function Documentation

39.544.2.1 SplitLeafNode()

```
static void SplitLeafNode (  
    TreeType * tree,  
    std::vector< bool > & relevels ) [static]
```

Split a leaf node using the "default" algorithm.

If necessary, this split will propagate upwards through the tree.

39.544.2.2 SplitNonLeafNode()

```
static bool SplitNonLeafNode (
    TreeType * tree,
    std::vector< bool > & relevels ) [static]
```

Split a non-leaf node using the "default" algorithm.

If this is a root node, the tree increases in depth.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ **r_tree_split.hpp**

39.545 SpaceSplit< MetricType, MatType > Class Template Reference

Static Public Member Functions

- static bool **GetProjVector** (const **bound::HRectBound**< MetricType > &bound, const MatType &data, const arma::Col< size_t > &points, **AxisParallelProjVector** &projVector, double &midValue)
Create a projection vector based on the given set of point.
- template<typename BoundType >
static bool **GetProjVector** (const BoundType &bound, const MatType &data, const arma::Col< size_t > &points, **ProjVector** &projVector, double &midValue)
Create a projection vector based on the given set of point.

39.545.1 Detailed Description

```
template<typename MetricType, typename MatType>
class mlpack::tree::SpaceSplit< MetricType, MatType >
```

Definition at line 23 of file space_split.hpp.

39.545.2 Member Function Documentation

39.545.2.1 GetProjVector() [1/2]

```
static bool GetProjVector (
    const bound::HRectBound< MetricType > & bound,
    const MatType & data,
    const arma::Col< size_t > & points,
    AxisParallelProjVector & projVector,
    double & midValue ) [static]
```

Create a projection vector based on the given set of point.

This special case will create an axis-parallel projection vector in the dimension that has the maximum width.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the tree.
<i>points</i>	Vector of indexes of points to be considered.
<i>projVector</i>	Resulting axis-parallel projection vector.
<i>midValue</i>	Mid value in the chosen projection.

Returns

Flag to determine if it is possible.

39.545.2.2 GetProjVector() [2/2]

```
static bool GetProjVector (
    const BoundType & bound,
    const MatType & data,
    const arma::Col< size_t > & points,
    ProjVector & projVector,
    double & midValue ) [static]
```

Create a projection vector based on the given set of point.

We efficiently estimate the farthest pair of points in the given set: p and q, and then consider the projection vector (q - p).

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the tree.
<i>points</i>	Vector of indexes of points to be considered.
<i>projVector</i>	Resulting projection vector.
<i>midValue</i>	Mid value in the chosen projection.

Returns

Flag to determine if it is possible.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/ **space_split.hpp**

39.546 SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > Class Template Reference

A hybrid spill tree is a variant of binary space trees in which the children of a node can "spill over" each other, and contain shared datapoints.

Classes

- class **SpillDualTreeTraverser**
A generic dual-tree traverser for hybrid spill trees; see [spill_dual_tree_traverser.hpp](#) (p. 2683) for implementation.
- class **SpillSingleTreeTraverser**
A generic single-tree traverser for hybrid spill trees; see [spill_single_tree_traverser.hpp](#) (p. 2684) for implementation.

Public Types

- typedef HyperplaneType< MetricType >:: **BoundType** **BoundType**
The bound type.
- template<typename RuleType >
using **DefeatistDualTreeTraverser** = **SpillDualTreeTraverser**< RuleType, true >
A defeatist dual-tree traverser for hybrid spill trees.
- template<typename RuleType >
using **DefeatistSingleTreeTraverser** = **SpillSingleTreeTraverser**< RuleType, true >
A defeatist single-tree traverser for hybrid spill trees.
- template<typename RuleType >
using **DualTreeTraverser** = **SpillDualTreeTraverser**< RuleType, false >
A dual-tree traverser for hybrid spill trees.
- typedef MatType::elem_type **ElemType**
The type of element held in MatType.
- typedef MatType **Mat**
So other classes can use TreeType::Mat.
- template<typename RuleType >
using **SingleTreeTraverser** = **SpillSingleTreeTraverser**< RuleType, false >
A single-tree traverser for hybrid spill trees.

Public Member Functions

- **SpillTree** (const MatType &data, const double tau=0, const size_t maxLeafSize=20, const double rho=0.7)
Construct this as the root node of a hybrid spill tree using the given dataset.
- **SpillTree** (MatType &&data, const double tau=0, const size_t maxLeafSize=20, const double rho=0.7)
Construct this as the root node of a hybrid spill tree using the given dataset.
- **SpillTree** (**SpillTree** *parent, arma::Col< size_t > &points, const double tau=0, const size_t maxLeafSize=20, const double rho=0.7)
Construct this node as a child of the given parent, including the given list of points.
- **SpillTree** (const **SpillTree** &other)
Create a hybrid spill tree by copying the other tree.

- **SpillTree** (**SpillTree** &&other)

*Move constructor for a **SpillTree** (p. 2274); possess all the members of the given tree.*
- template<typename Archive >
 SpillTree (Archive &ar, const typename **std::enable_if_t**< Archive::is_loading::value > *=0)

*Initialize the tree from a **boost::serialization** (p. 251) archive.*
- **~SpillTree** ()

Deletes this node, deallocating the memory for the children and calling their destructors in turn.
- const **BoundType** & **Bound** () const

Return the bound object for this node.
- **BoundType** & **Bound** ()

Return the bound object for this node.
- void **Center** (arma::vec ¢er)

Store the center of the bounding region in the given vector.
- **SpillTree** & **Child** (const size_t child) const

Return the specified child (0 will be left, 1 will be right).
- **SpillTree** *& **ChildPtr** (const size_t child)
- const MatType & **Dataset** () const

Get the dataset which the tree is built on.
- size_t **Descendant** (const size_t index) const

Return the index (with reference to the dataset) of a particular descendant of this node.
- **ElemType** **FurthestDescendantDistance** () const

Return the furthest possible descendant distance.
- **ElemType** **FurthestPointDistance** () const

Return the furthest distance to a point held in this node.
- template<typename VecType >
 size_t **GetFurthestChild** (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > *=0)

Return the index of the furthest child node to the given query point (this is an efficient estimation based on the splitting hyperplane, the node returned is not necessarily the furthest).
- size_t **GetFurthestChild** (const **SpillTree** &queryNode)

Return the index of the furthest child node to the given query node (this is an efficient estimation based on the splitting hyperplane, the node returned is not necessarily the furthest).
- template<typename VecType >
 size_t **GetNearestChild** (const VecType &point, typename **std::enable_if_t**< **IsVector**< VecType >::value > *=0)

Return the index of the nearest child node to the given query point (this is an efficient estimation based on the splitting hyperplane, the node returned is not necessarily the nearest).
- size_t **GetNearestChild** (const **SpillTree** &queryNode)

Return the index of the nearest child node to the given query node (this is an efficient estimation based on the splitting hyperplane, the node returned is not necessarily the nearest).
- const HyperplaneType< MetricType > & **Hyperplane** () const

Get the Hyperplane instance.
- bool **IsLeaf** () const

Return whether or not this node is a leaf (true if it has no children).
- **SpillTree** * **Left** () const

Gets the left child of this node.
- **SpillTree** *& **Left** ()

Modify the left child of this node.
- **ElemType** **MaxDistance** (const **SpillTree** &other) const

- Return the maximum distance to another node.*

 - `template<typename VecType >`
ElemType MaxDistance (const VecType &point, typename **std::enable_if_t< IsVector< VecType >::value >*=0**) const

Return the maximum distance to another point.
- `MetricType Metric` () const
- Get the metric that the tree uses.*

 - **ElemType MinDistance** (const **SpillTree** &other) const

Return the minimum distance to another node.
- `template<typename VecType >`
ElemType MinDistance (const VecType &point, typename **std::enable_if_t< IsVector< VecType >::value >*=0**) const
- Return the minimum distance to another point.*

 - **ElemType MinimumBoundDistance** () const

Return the minimum distance from the center of the node to any bound edge.
- `size_t NumChildren` () const
- Return the number of children in this node.*

 - `size_t NumDescendants` () const

Return the number of descendants of this node.
- `size_t NumPoints` () const
- Return the number of points in this node (0 if not a leaf).*

 - `bool Overlap` () const

Distinguish overlapping nodes from non-overlapping nodes.
- **SpillTree * Parent** () const
- Gets the parent of this node.*

 - **SpillTree *& Parent** ()

Modify the parent of this node.
- **ElemType ParentDistance** () const
- Return the distance from the center of this node to the center of the parent node.*

 - **ElemType & ParentDistance** ()

Modify the distance from the center of this node to the center of the parent node.
- `size_t Point` (const `size_t` index) const
- Return the index (with reference to the dataset) of a particular point in this node.*

 - **math::RangeType< ElemType > RangeDistance** (const **SpillTree** &other) const

Return the minimum and maximum distance to another node.
- `template<typename VecType >`
math::RangeType< ElemType > RangeDistance (const VecType &point, typename **std::enable_if_t< IsVector< VecType >::value >*=0**) const
- Return the minimum and maximum distance to another point.*

 - **SpillTree * Right** () const

Gets the right child of this node.
- **SpillTree *& Right** ()
- Modify the right child of this node.*

 - `template<typename Archive >`
void serialize (Archive &ar, const unsigned int version)

Serialize the tree.
- `const StatisticType & Stat` () const
- Return the statistic object for this node.*

 - `StatisticType & Stat` ()

Return the statistic object for this node.

Static Public Member Functions

- static bool **HasSelfChildren** ()
Returns false: this tree type does not have self children.

Protected Member Functions

- **SpillTree** ()
A default constructor.

39.546.1 Detailed Description

```
template<typename MetricType, typename StatisticType = EmptyStatistic, typename MatType = arma::mat, template< typename  
HyperplaneMetricType > class HyperplaneType = AxisOrthogonalHyperplane, template< typename SplitMetricType, typename SplitType  
MatType > class SplitType = MidpointSpaceSplit>  
class mlpack::tree::SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >
```

A hybrid spill tree is a variant of binary space trees in which the children of a node can "spill over" each other, and contain shared datapoints.

Two new separating planes lplane and rplane are defined, both of which are parallel to the original decision boundary and at a distance tau from it. The region between lplane and rplane is called "overlapping buffer".

For each node, we first split the points considering the overlapping buffer. If either of its children contains more than rho fraction of the total points we undo the overlapping splitting. Instead a conventional partition is used. In this way, we can ensure that each split reduces the number of points of a node by at least a constant factor.

This particular tree does not allow growth, so you cannot add or delete nodes from it. If you need to add or delete a node, the better procedure is to rebuild the tree entirely.

Three runtime parameters are required in the constructor:

- maxLeafSize: Max leaf size to be used.
- tau: Overlapping size.
- rho: Balance threshold.

For more information on spill trees, see

```
@inproceedings{  
  author = {Ting Liu, Andrew W. Moore, Alexander Gray and Ke Yang},  
  title = {An Investigation of Practical Approximate Nearest Neighbor  
    Algorithms},  
  booktitle = {Advances in Neural Information Processing Systems 17},  
  year = {2005},  
  pages = {825--832}  
}
```

Template Parameters

<i>MetricType</i>	The metric used for tree-building.
<i>StatisticType</i>	Extra data contained in the node. See statistic.hpp (p. 2688) for the necessary skeleton interface.
<i>MatType</i>	The dataset class.
<i>HyperplaneType</i>	The splitting hyperplane class.
<i>SplitType</i>	The class that partitions the dataset/points at a particular node into two parts. Its definition decides the way this split is done.

Definition at line 73 of file `spill_tree.hpp`.

39.546.2 Member Typedef Documentation

39.546.2.1 BoundType

```
typedef HyperplaneType<MetricType>:: BoundType BoundType
```

The bound type.

Definition at line 81 of file `spill_tree.hpp`.

39.546.2.2 DefeatistDualTreeTraverser

```
using DefeatistDualTreeTraverser = SpillDualTreeTraverser<RuleType, true>
```

A defeatist dual-tree traverser for hybrid spill trees.

Definition at line 146 of file `spill_tree.hpp`.

39.546.2.3 DefeatistSingleTreeTraverser

```
using DefeatistSingleTreeTraverser = SpillSingleTreeTraverser<RuleType, true>
```

A defeatist single-tree traverser for hybrid spill trees.

Definition at line 138 of file `spill_tree.hpp`.

39.546.2.4 **DualTreeTraverser**

```
using DualTreeTraverser = SpillDualTreeTraverser<RuleType, false>
```

A dual-tree traverser for hybrid spill trees.

Definition at line 142 of file `spill_tree.hpp`.

39.546.2.5 **ElemType**

```
typedef MatType::elem_type ElemType
```

The type of element held in `MatType`.

Definition at line 79 of file `spill_tree.hpp`.

39.546.2.6 **Mat**

```
typedef MatType Mat
```

So other classes can use `TreeType::Mat`.

Definition at line 77 of file `spill_tree.hpp`.

39.546.2.7 **SingleTreeTraverser**

```
using SingleTreeTraverser = SpillSingleTreeTraverser<RuleType, false>
```

A single-tree traverser for hybrid spill trees.

Definition at line 134 of file `spill_tree.hpp`.

39.546.3 **Constructor & Destructor Documentation**

39.546.3.1 **SpillTree()** [1/7]

```
SpillTree (  
    const MatType & data,  
    const double tau = 0,  
    const size_t maxLeafSize = 20,  
    const double rho = 0.7 )
```

Construct this as the root node of a hybrid spill tree using the given dataset.

The dataset will not be modified during the building procedure (unlike **BinarySpaceTree** (p. 1998)).

Parameters

<i>data</i>	Dataset to create tree from.
<i>tau</i>	Overlapping size.
<i>maxLeafSize</i>	Size of each leaf in the tree.
<i>rho</i>	Balance threshold.

39.546.3.2 `SpillTree()` [2/7]

```
SpillTree (
    MatType && data,
    const double tau = 0,
    const size_t maxLeafSize = 20,
    const double rho = 0.7 )
```

Construct this as the root node of a hybrid spill tree using the given dataset.

This will take ownership of the data matrix; if you don't want this, consider using the constructor that takes a const reference to a dataset.

Parameters

<i>data</i>	Dataset to create tree from.
<i>tau</i>	Overlapping size.
<i>maxLeafSize</i>	Size of each leaf in the tree.
<i>rho</i>	Balance threshold.

39.546.3.3 `SpillTree()` [3/7]

```
SpillTree (
    SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > * parent,
    arma::Col< size_t > & points,
    const double tau = 0,
    const size_t maxLeafSize = 20,
    const double rho = 0.7 )
```

Construct this node as a child of the given parent, including the given list of points.

This is used for recursive tree-building by the other constructors which don't specify point indices.

Parameters

<i>parent</i>	Parent of this node.
---------------	----------------------

Parameters

<i>points</i>	Vector of indexes of points to be included in this node.
<i>tau</i>	Overlapping size.
<i>maxLeafSize</i>	Size of each leaf in the tree.
<i>rho</i>	Balance threshold.

39.546.3.4 SpillTree() [4 / 7]

```
SpillTree (
    const SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > &
    other )
```

Create a hybrid spill tree by copying the other tree.

Be careful! This can take a long time and use a lot of memory.

Parameters

<i>other</i>	tree to be replicated.
--------------	------------------------

39.546.3.5 SpillTree() [5 / 7]

```
SpillTree (
    SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > && other
)
```

Move constructor for a **SpillTree** (p. 2274); possess all the members of the given tree.

Parameters

<i>other</i>	tree to be moved.
--------------	-------------------

39.546.3.6 SpillTree() [6 / 7]

```
SpillTree (
    Archive & ar,
    const typename std::enable_if_t< Archive::is_loading::value > * = 0 )
```

Initialize the tree from a **boost::serialization** (p. 251) archive.

Parameters

<i>ar</i>	Archive to load tree from. Must be an iarchive, not an oarchive.
-----------	--

39.546.3.7 ~SpillTree()

`~ SpillTree ()`

Deletes this node, deallocating the memory for the children and calling their destructors in turn.

This will invalidate any pointers or references to any nodes which are children of this one.

39.546.3.8 SpillTree() [7/7]

`SpillTree () [protected]`

A default constructor.

This is meant to only be used with **boost::serialization** (p. 251), which is allowed with the friend declaration below. This does not return a valid tree! The method must be protected, so that the serialization shim can work with the default constructor.

Referenced by `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::Center()`.

39.546.4 Member Function Documentation

39.546.4.1 Bound() [1/2]

`const BoundType& Bound () const [inline]`

Return the bound object for this node.

Definition at line 230 of file `spill_tree.hpp`.

Referenced by `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::MaxDistance()`, `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::MinDistance()`, and `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::RangeDistance()`.

39.546.4.2 Bound() [2/2]

```
BoundType& Bound ( ) [inline]
```

Return the bound object for this node.

Definition at line 232 of file spill_tree.hpp.

39.546.4.3 Center()

```
void Center (
    arma::vec & center ) [inline]
```

Store the center of the bounding region in the given vector.

Definition at line 424 of file spill_tree.hpp.

References SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::SpillTree().

39.546.4.4 Child()

```
SpillTree& Child (
    const size_t child ) const
```

Return the specified child (0 will be left, 1 will be right).

If the index is greater than 1, this will return the right child.

Parameters

<i>child</i>	Index of child to return.
--------------	---------------------------

Referenced by SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::ParentDistance().

39.546.4.5 ChildPtr()

```
SpillTree*& ChildPtr (
    const size_t child ) [inline]
```

Definition at line 343 of file spill_tree.hpp.

References SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::Descendant(), SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::NumDescendants(), SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::NumPoints(), and SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::Point().

39.546.4.6 Dataset()

```
const MatType& Dataset ( ) const [inline]
```

Get the dataset which the tree is built on.

Definition at line 258 of file spill_tree.hpp.

39.546.4.7 Descendant()

```
size_t Descendant (
    const size_t index ) const
```

Return the index (with reference to the dataset) of a particular descendant of this node.

The index should be greater than zero but less than the number of descendants.

Parameters

<i>index</i>	Index of the descendant.
--------------	--------------------------

Referenced by SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::ChildPtr().

39.546.4.8 FurthestDescendantDistance()

```
ElemType FurthestDescendantDistance ( ) const
```

Return the furthest possible descendant distance.

This returns the maximum distance from the centroid to the edge of the bound and not the empirical quantity which is the actual furthest descendant distance. So the actual furthest descendant distance may be less than what this method returns (but it will never be greater than this).

Referenced by SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::Metric().

39.546.4.9 FurthestPointDistance()

```
ElemType FurthestPointDistance ( ) const
```

Return the furthest distance to a point held in this node.

If this is not a leaf node, then the distance is 0 because the node holds no points.

Referenced by `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::Metric()`.

39.546.4.10 GetFurthestChild() [1/2]

```
size_t GetFurthestChild (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 )
```

Return the index of the furthest child node to the given query point (this is an efficient estimation based on the splitting hyperplane, the node returned is not necessarily the furthest).

If this is a leaf node, it will return **NumChildren()** (p. 2289) (invalid index).

Referenced by `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::Metric()`.

39.546.4.11 GetFurthestChild() [2/2]

```
size_t GetFurthestChild (
    const SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > &
    queryNode )
```

Return the index of the furthest child node to the given query node (this is an efficient estimation based on the splitting hyperplane, the node returned is not necessarily the furthest).

If it can't decide it will return **NumChildren()** (p. 2289) (invalid index).

39.546.4.12 GetNearestChild() [1/2]

```
size_t GetNearestChild (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 )
```

Return the index of the nearest child node to the given query point (this is an efficient estimation based on the splitting hyperplane, the node returned is not necessarily the nearest).

If this is a leaf node, it will return **NumChildren()** (p. 2289) (invalid index).

Referenced by `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::Metric()`.

39.546.4.13 GetNearestChild() [2/2]

```
size_t GetNearestChild (
    const SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > &
    queryNode )
```

Return the index of the nearest child node to the given query node (this is an efficient estimation based on the splitting hyperplane, the node returned is not necessarily the nearest).

If it can't decide it will return **NumChildren()** (p. 2289) (invalid index).

39.546.4.14 HasSelfChildren()

```
static bool HasSelfChildren ( ) [inline], [static]
```

Returns false: this tree type does not have self children.

Definition at line 421 of file spill_tree.hpp.

39.546.4.15 Hyperplane()

```
const HyperplaneType<MetricType>& Hyperplane ( ) const [inline]
```

Get the Hyperplane instance.

Definition at line 264 of file spill_tree.hpp.

39.546.4.16 IsLeaf()

```
bool IsLeaf ( ) const
```

Return whether or not this node is a leaf (true if it has no children).

Referenced by SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::Stat().

39.546.4.17 Left() [1/2]

```
SpillTree* Left ( ) const [inline]
```

Gets the left child of this node.

Definition at line 243 of file spill_tree.hpp.

39.546.4.18 Left() [2/2]

```
SpillTree*& Left ( ) [inline]
```

Modify the left child of this node.

Definition at line 245 of file spill_tree.hpp.

39.546.4.19 MaxDistance() [1/2]

```
ElemType MaxDistance (
    const SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > &
    other ) const [inline]
```

Return the maximum distance to another node.

Definition at line 382 of file spill_tree.hpp.

References `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::Bound()`.

39.546.4.20 MaxDistance() [2/2]

```
ElemType MaxDistance (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const [inline]
```

Return the maximum distance to another point.

Definition at line 404 of file spill_tree.hpp.

39.546.4.21 Metric()

```
MetricType Metric ( ) const [inline]
```

Get the metric that the tree uses.

Definition at line 267 of file spill_tree.hpp.

References `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::FurthestDescendantDistance()`, `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::FurthestPointDistance()`, `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::GetFurthestChild()`, `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::GetNearestChild()`, `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::MinimumBoundDistance()`, and `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::NumChildren()`.

39.546.4.22 MinDistance() [1/2]

```
ElemType MinDistance (
    const SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > &
    other ) const [inline]
```

Return the minimum distance to another node.

Definition at line 376 of file spill_tree.hpp.

References SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::Bound().

39.546.4.23 MinDistance() [2/2]

```
ElemType MinDistance (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const [inline]
```

Return the minimum distance to another point.

Definition at line 395 of file spill_tree.hpp.

39.546.4.24 MinimumBoundDistance()

```
ElemType MinimumBoundDistance ( ) const
```

Return the minimum distance from the center of the node to any bound edge.

Referenced by SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::Metric().

39.546.4.25 NumChildren()

```
size_t NumChildren ( ) const
```

Return the number of children in this node.

Referenced by SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::Metric().

39.546.4.26 NumDescendants()

```
size_t NumDescendants ( ) const
```

Return the number of descendants of this node.

For a non-leaf spill tree, this is the number of points at the descendant leaves. For a leaf, this is the number of points in the leaf.

Referenced by `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::ChildPtr()`.

39.546.4.27 NumPoints()

```
size_t NumPoints ( ) const
```

Return the number of points in this node (0 if not a leaf).

Referenced by `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::ChildPtr()`.

39.546.4.28 Overlap()

```
bool Overlap ( ) const [inline]
```

Distinguish overlapping nodes from non-overlapping nodes.

Definition at line 261 of file `spill_tree.hpp`.

39.546.4.29 Parent() [1/2]

```
SpillTree* Parent ( ) const [inline]
```

Gets the parent of this node.

Definition at line 253 of file `spill_tree.hpp`.

39.546.4.30 Parent() [2/2]

```
SpillTree* Parent ( ) [inline]
```

Modify the parent of this node.

Definition at line 255 of file spill_tree.hpp.

39.546.4.31 ParentDistance() [1/2]

```
ElemType ParentDistance ( ) const [inline]
```

Return the distance from the center of this node to the center of the parent node.

Definition at line 330 of file spill_tree.hpp.

39.546.4.32 ParentDistance() [2/2]

```
ElemType& ParentDistance ( ) [inline]
```

Modify the distance from the center of this node to the center of the parent node.

Definition at line 333 of file spill_tree.hpp.

References SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::Child().

39.546.4.33 Point()

```
size_t Point (
    const size_t index ) const
```

Return the index (with reference to the dataset) of a particular point in this node.

This will happily return invalid indices if the given index is greater than the number of points in this node (obtained with **NumPoints()** (p. 2290)) – be careful.

Parameters

<i>index</i>	Index of point for which a dataset index is wanted.
--------------	---

Referenced by `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::ChildPtr()`.

39.546.4.34 `RangeDistance()` [1/2]

```
math::RangeType< ElemType> RangeDistance (
    const SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > &
    other ) const [inline]
```

Return the minimum and maximum distance to another node.

Definition at line 388 of file `spill_tree.hpp`.

References `SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::Bound()`.

39.546.4.35 `RangeDistance()` [2/2]

```
math::RangeType< ElemType> RangeDistance (
    const VecType & point,
    typename std::enable_if_t< IsVector< VecType >::value > * = 0 ) const [inline]
```

Return the minimum and maximum distance to another point.

Definition at line 414 of file `spill_tree.hpp`.

39.546.4.36 `Right()` [1/2]

```
SpillTree* Right ( ) const [inline]
```

Gets the right child of this node.

Definition at line 248 of file `spill_tree.hpp`.

39.546.4.37 `Right()` [2/2]

```
SpillTree& Right ( ) [inline]
```

Modify the right child of this node.

Definition at line 250 of file `spill_tree.hpp`.

39.546.4.38 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int version )
```

Serialize the tree.

39.546.4.39 Stat() [1/2]

```
const StatisticType& Stat ( ) const [inline]
```

Return the statistic object for this node.

Definition at line 235 of file spill_tree.hpp.

39.546.4.40 Stat() [2/2]

```
StatisticType& Stat ( ) [inline]
```

Return the statistic object for this node.

Definition at line 237 of file spill_tree.hpp.

References SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::IsLeaf().

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/ **spill_tree.hpp**

39.547 SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::SpillDualTreeTraverser< MetricType, StatisticType, MatType, HyperplaneType, SplitType > Class Template Reference

A generic dual-tree traverser for hybrid spill trees; see **spill_dual_tree_traverser.hpp** (p. 2683) for implementation.

Public Member Functions

- **SpillDualTreeTraverser** (RuleType &rule)
Instantiate the dual-tree traverser with the given rule set.
- size_t **NumBaseCases** () const
Get the number of times a base case was calculated.
- size_t & **NumBaseCases** ()
Modify the number of times a base case was calculated.
- size_t **NumPrunes** () const
Get the number of prunes.
- size_t & **NumPrunes** ()
Modify the number of prunes.
- size_t **NumScores** () const
Get the number of times a node combination was scored.
- size_t & **NumScores** ()
Modify the number of times a node combination was scored.
- size_t **NumVisited** () const
Get the number of visited combinations.
- size_t & **NumVisited** ()
Modify the number of visited combinations.
- void **Traverse** (**SpillTree** &queryNode, **SpillTree** &referenceNode)
Traverse the two trees.

39.547.1 Detailed Description

```
template<typename MetricType, typename StatisticType = EmptyStatistic, typename MatType = arma::mat, template< typename
HyperplaneMetricType > class HyperplaneType = AxisOrthogonalHyperplane, template< typename SplitMetricType, typename SplitMatType > class SplitType = MidpointSpaceSplit>
template<typename MetricType, typename StatisticType, typename MatType, template< typename HyperplaneMetricType > class
HyperplaneType, template< typename SplitMetricType, typename SplitMatType > class SplitType>
class mpack::tree::SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::SpillDualTreeTraverser< MetricType,
StatisticType, MatType, HyperplaneType, SplitType >
```

A generic dual-tree traverser for hybrid spill trees; see **spill_dual_tree_traverser.hpp** (p.2683) for implementation.

The Defeatist template parameter determines if the traverser must do defeatist search on overlapping nodes.

Definition at line 35 of file **spill_dual_tree_traverser.hpp**.

39.547.2 Constructor & Destructor Documentation

39.547.2.1 SpillDualTreeTraverser()

```
SpillDualTreeTraverser (
    RuleType & rule )
```

Instantiate the dual-tree traverser with the given rule set.

39.547.3 Member Function Documentation

39.547.3.1 NumBaseCases() [1/2]

```
size_t NumBaseCases ( ) const [inline]
```

Get the number of times a base case was calculated.

Definition at line 70 of file spill_dual_tree_traverser.hpp.

39.547.3.2 NumBaseCases() [2/2]

```
size_t& NumBaseCases ( ) [inline]
```

Modify the number of times a base case was calculated.

Definition at line 72 of file spill_dual_tree_traverser.hpp.

39.547.3.3 NumPrunes() [1/2]

```
size_t NumPrunes ( ) const [inline]
```

Get the number of prunes.

Definition at line 55 of file spill_dual_tree_traverser.hpp.

39.547.3.4 NumPrunes() [2/2]

```
size_t& NumPrunes ( ) [inline]
```

Modify the number of prunes.

Definition at line 57 of file spill_dual_tree_traverser.hpp.

39.547.3.5 NumScores() [1/2]

```
size_t NumScores ( ) const [inline]
```

Get the number of times a node combination was scored.

Definition at line 65 of file spill_dual_tree_traverser.hpp.

39.547.3.6 NumScores() [2/2]

```
size_t& NumScores ( ) [inline]
```

Modify the number of times a node combination was scored.

Definition at line 67 of file spill_dual_tree_traverser.hpp.

39.547.3.7 NumVisited() [1/2]

```
size_t NumVisited ( ) const [inline]
```

Get the number of visited combinations.

Definition at line 60 of file spill_dual_tree_traverser.hpp.

39.547.3.8 NumVisited() [2/2]

```
size_t& NumVisited ( ) [inline]
```

Modify the number of visited combinations.

Definition at line 62 of file spill_dual_tree_traverser.hpp.

39.547.3.9 Traverse()

```
void Traverse (
    SpillTree & queryNode,
    SpillTree & referenceNode )
```

Traverse the two trees.

This does not reset the number of prunes.

Parameters

<i>queryNode</i>	The query node to be traversed.
<i>referenceNode</i>	The reference node to be traversed.
<i>score</i>	The score of the current node combination.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/ **spill_dual_tree_traverser.hpp**

39.548 **SpillTree**< **MetricType**, **StatisticType**, **MatType**, **HyperplaneType**, **SplitType** >::**SpillSingleTreeTraverser**< **MetricType**, **StatisticType**, **MatType**, **HyperplaneType**, **SplitType** > **Class Template Reference**

A generic single-tree traverser for hybrid spill trees; see **spill_single_tree_traverser.hpp** (p. 2684) for implementation.

Public Member Functions

- **SpillSingleTreeTraverser** (**RuleType** &rule)
Instantiate the single tree traverser with the given rule set.
- **size_t NumPrunes** () const
Get the number of prunes.
- **size_t & NumPrunes** ()
Modify the number of prunes.
- **void Traverse** (const **size_t** queryIndex, **SpillTree** &referenceNode)
Traverse the tree with the given point.

39.548.1 Detailed Description

```
template<typename MetricType, typename StatisticType = EmptyStatistic, typename MatType = arma::mat, template< typename
HyperplaneMetricType > class HyperplaneType = AxisOrthogonalHyperplane, template< typename SplitMetricType, typename SplitType > class SplitType = MidpointSpaceSplit>
template<typename MetricType, typename StatisticType, typename MatType, template< typename HyperplaneMetricType > class
HyperplaneType, template< typename SplitMetricType, typename SplitMatType > class SplitType>
class mlpack::tree::SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >::SpillSingleTreeTraverser< MetricType, StatisticType, MatType, HyperplaneType, SplitType >
```

A generic single-tree traverser for hybrid spill trees; see **spill_single_tree_traverser.hpp** (p. 2684) for implementation.

The Defeatist template parameter determines if the traverser must do defeatist search on overlapping nodes.

Definition at line 34 of file **spill_single_tree_traverser.hpp**.

39.548.2 Constructor & Destructor Documentation

39.548.2.1 SpillSingleTreeTraverser()

```
SpillSingleTreeTraverser (  
    RuleType & rule )
```

Instantiate the single tree traverser with the given rule set.

39.548.3 Member Function Documentation

39.548.3.1 NumPrunes() [1/2]

```
size_t NumPrunes ( ) const [inline]
```

Get the number of prunes.

Definition at line 53 of file spill_single_tree_traverser.hpp.

39.548.3.2 NumPrunes() [2/2]

```
size_t& NumPrunes ( ) [inline]
```

Modify the number of prunes.

Definition at line 55 of file spill_single_tree_traverser.hpp.

39.548.3.3 Traverse()

```
void Traverse (  
    const size_t queryIndex,  
    SpillTree & referenceNode )
```

Traverse the tree with the given point.

Parameters

<i>queryIndex</i>	The index of the point in the query set which is being used as the query point.
<i>referenceNode</i>	The tree node to be traversed.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/ **spill_single_tree_traverser.hpp**

39.549 TraversalInfo< TreeType > Class Template Reference

The **TraversalInfo** (p. 2299) class holds traversal information which is used in dual-tree (and single-tree) traversals.

Public Member Functions

- **TraversalInfo** ()
*Create the **TraversalInfo** (p. 2299) object and initialize the pointers to NULL.*
- double **LastBaseCase** () const
Get the base case associated with the last node combination.
- double & **LastBaseCase** ()
Modify the base case associated with the last node combination.
- TreeType * **LastQueryNode** () const
Get the last query node.
- TreeType *& **LastQueryNode** ()
Modify the last query node.
- TreeType * **LastReferenceNode** () const
Get the last reference node.
- TreeType *& **LastReferenceNode** ()
Modify the last reference node.
- double **LastScore** () const
Get the score associated with the last query and reference nodes.
- double & **LastScore** ()
Modify the score associated with the last query and reference nodes.

39.549.1 Detailed Description

```
template<typename TreeType>
class mlpack::tree::TraversalInfo< TreeType >
```

The **TraversalInfo** (p. 2299) class holds traversal information which is used in dual-tree (and single-tree) traversals.

A traversal should be updating the members of this class before `Score()` is called. This class should be held as a member of the `RuleType` class and the interface to it should be through a **TraversalInfo()** (p. 2300) method.

The information held by this class is the last node combination visited before the current node combination was recursed into, and the score resulting from when `Score()` was called on that combination. However, this information is identical for a query node and a reference node in a particular node combination, so traversals only need to update the **TraversalInfo** (p. 2299) object in a query node (and the algorithms should only use the **TraversalInfo** (p. 2299) object from a query node).

In general, this auxiliary traversal information is used to try and make a prune without needing to call `BaseCase()` or calculate the distance between nodes. Using this information you can place bounds on the distance between the two nodes quickly.

If the traversal is not updating the members of this class correctly, a likely result is a null pointer dereference. Dual-tree algorithms should assume that the members are set properly and should not need to check for null pointers.

There is one exception, which is the root node combination; the score can be set to 0 and the query and reference nodes can just be set to the root nodes; no algorithm should be able to prune the root combination anyway.

Definition at line 50 of file `traversal_info.hpp`.

39.549.2 Constructor & Destructor Documentation

39.549.2.1 TraversalInfo()

```
TraversalInfo ( ) [inline]
```

Create the **TraversalInfo** (p. 2299) object and initialize the pointers to `NULL`.

Definition at line 56 of file `traversal_info.hpp`.

39.549.3 Member Function Documentation

39.549.3.1 LastBaseCase() [1/2]

```
double LastBaseCase ( ) const [inline]
```

Get the base case associated with the last node combination.

Definition at line 78 of file traversal_info.hpp.

39.549.3.2 LastBaseCase() [2/2]

```
double& LastBaseCase ( ) [inline]
```

Modify the base case associated with the last node combination.

Definition at line 80 of file traversal_info.hpp.

39.549.3.3 LastQueryNode() [1/2]

```
TreeType* LastQueryNode ( ) const [inline]
```

Get the last query node.

Definition at line 63 of file traversal_info.hpp.

39.549.3.4 LastQueryNode() [2/2]

```
TreeType*& LastQueryNode ( ) [inline]
```

Modify the last query node.

Definition at line 65 of file traversal_info.hpp.

39.549.3.5 LastReferenceNode() [1/2]

```
TreeType* LastReferenceNode ( ) const [inline]
```

Get the last reference node.

Definition at line 68 of file traversal_info.hpp.

39.549.3.6 LastReferenceNode() [2/2]

```
TreeType*& LastReferenceNode ( ) [inline]
```

Modify the last reference node.

Definition at line 70 of file traversal_info.hpp.

39.549.3.7 LastScore() [1/2]

```
double LastScore ( ) const [inline]
```

Get the score associated with the last query and reference nodes.

Definition at line 73 of file traversal_info.hpp.

39.549.3.8 LastScore() [2/2]

```
double& LastScore ( ) [inline]
```

Modify the score associated with the last query and reference nodes.

Definition at line 75 of file traversal_info.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ **traversal_info.hpp**

39.550 TreeTraits< TreeType > Class Template Reference

The **TreeTraits** (p. 2302) class provides compile-time information on the characteristics of a given tree type.

Static Public Attributes

- static const bool **BinaryTree** = false
This is true if the tree always has only two children.
- static const bool **FirstPointIsCentroid** = false
This is true if the first point of each node is the centroid of its bound.
- static const bool **HasDuplicatedPoints** = false
This is true if a point can be included in more than one node.
- static const bool **HasOverlappingChildren** = true
This is true if the subspaces represented by the children of a node can overlap.
- static const bool **HasSelfChildren** = false
This is true if the points contained in the first child of a node (Child(0)) are also contained in that node.
- static const bool **RearrangesDataset** = false
This is true if the tree rearranges points in the dataset when it is built.
- static const bool **UniqueNumDescendants** = true
This is true if the NumDescendants() method doesn't include duplicated points.

39.550.1 Detailed Description

```
template<typename TreeType>
class mlpack::tree::TreeTraits< TreeType >
```

The **TreeTraits** (p. 2302) class provides compile-time information on the characteristics of a given tree type.

These include traits such as whether or not a node knows the distance to its parent node, or whether or not the subspaces represented by children can overlap.

These traits can be used for static compile-time optimization:

```
// This if statement will be optimized out at compile time!
if (TreeTraits<TreeType>::HasOverlappingChildren == false)
{
    // Do a simpler computation because no children overlap.
}
else
{
    // Do the full, complex calculation.
}
```

The traits can also be used in conjunction with SFINAE to write specialized versions of functions:

```
template<typename TreeType>
void Compute(TreeType& node,
             std::enable_if_t<
                 TreeTraits<TreeType>::RearrangesDataset>*)
{
    // Computation where special dataset-rearranging tree constructor is
    // called.
}

template<typename TreeType>
void Compute(TreeType& node,
             std::enable_if_t<
                 !TreeTraits<TreeType>::RearrangesDataset>*)
{
    // Computation where normal tree constructor is called.
}
```

In those two examples, the **std::enable_if_t<>** (p. 476) class takes a boolean template parameter which allows that function to be called when the boolean is true.

Each trait must be a static const value and not a function; only const values can be used as template parameters (or constexprs can be used too). By default (the unspecialized implementation of **TreeTraits** (p. 2302)), each parameter is set to make as few assumptions about the tree as possible; so, even if **TreeTraits** (p. 2302) is not specialized for a particular tree type, tree-based algorithms should still work.

When you write your own tree, you must specialize the **TreeTraits** (p. 2302) class to your tree type and set the corresponding values appropriately. See **mlpack/core/tree/binary_space_tree/traits.hpp** (p. 2622) for an example.

Definition at line 77 of file **tree_traits.hpp**.

39.550.2 Member Data Documentation

39.550.2.1 BinaryTree

```
const bool BinaryTree = false [static]
```

This is true if the tree always has only two children.

Definition at line 110 of file tree_traits.hpp.

39.550.2.2 FirstPointIsCentroid

```
const bool FirstPointIsCentroid = false [static]
```

This is true if the first point of each node is the centroid of its bound.

Definition at line 94 of file tree_traits.hpp.

39.550.2.3 HasDuplicatedPoints

```
const bool HasDuplicatedPoints = false [static]
```

This is true if a point can be included in more than one node.

Definition at line 89 of file tree_traits.hpp.

39.550.2.4 HasOverlappingChildren

```
const bool HasOverlappingChildren = true [static]
```

This is true if the subspaces represented by the children of a node can overlap.

Definition at line 84 of file tree_traits.hpp.

39.550.2.5 HasSelfChildren

```
const bool HasSelfChildren = false [static]
```

This is true if the points contained in the first child of a node (Child(0)) are also contained in that node.

Definition at line 100 of file tree_traits.hpp.

39.550.2.6 **RearrangesDataset**

```
const bool RearrangesDataset = false [static]
```

This is true if the tree rearranges points in the dataset when it is built.

Definition at line 105 of file `tree_traits.hpp`.

39.550.2.7 **UniqueNumDescendants**

```
const bool UniqueNumDescendants = true [static]
```

This is true if the `NumDescendants()` method doesn't include duplicated points.

Definition at line 116 of file `tree_traits.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ tree_traits.hpp`

39.551 **TreeTraits**< **BinarySpaceTree**< **MetricType**, **StatisticType**, **MatType**, **bound::BallBound**, **SplitType** > > Class Template Reference

This is a specialization of the `TreeType` class to the `BallTree` tree type.

Static Public Attributes

- static const bool **BinaryTree** = true
- static const bool **FirstPointIsCentroid** = false
- static const bool **HasDuplicatedPoints** = false
- static const bool **HasOverlappingChildren** = true
- static const bool **HasSelfChildren** = false
- static const bool **RearrangesDataset** = true
- static const bool **UniqueNumDescendants** = true

39.551.1 Detailed Description

```
template<typename MetricType, typename StatisticType, typename MatType, template< typename SplitBoundType, typename SplitType>
class SplitType> class SplitType>
class mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::BallBound, SplitType > >
```

This is a specialization of the `TreeType` class to the `BallTree` tree type.

The only difference with general **BinarySpaceTree** (p. 1998) is that `BallTree` can have overlapping children. See `mlpack/core/tree/tree_traits.hpp` (p. 2690) for more information.

Definition at line 187 of file `traits.hpp`.

39.551.2 Member Data Documentation

39.551.2.1 BinaryTree

```
const bool BinaryTree = true [static]
```

Definition at line 196 of file traits.hpp.

39.551.2.2 FirstPointIsCentroid

```
const bool FirstPointIsCentroid = false [static]
```

Definition at line 193 of file traits.hpp.

39.551.2.3 HasDuplicatedPoints

```
const bool HasDuplicatedPoints = false [static]
```

Definition at line 192 of file traits.hpp.

39.551.2.4 HasOverlappingChildren

```
const bool HasOverlappingChildren = true [static]
```

Definition at line 191 of file traits.hpp.

39.551.2.5 HasSelfChildren

```
const bool HasSelfChildren = false [static]
```

Definition at line 194 of file traits.hpp.

39.551.2.6 RearrangesDataset

```
const bool RearrangesDataset = true [static]
```

Definition at line 195 of file traits.hpp.

39.551.2.7 UniqueNumDescendants

```
const bool UniqueNumDescendants = true [static]
```

Definition at line 197 of file traits.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ **traits.hpp**

39.552 TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::CellBound, SplitType > > Class Template Reference

This is a specialization of the TreeType class to the UBTree tree type.

Static Public Attributes

- static const bool **BinaryTree** = true
- static const bool **FirstPointIsCentroid** = false
- static const bool **HasDuplicatedPoints** = false
- static const bool **HasOverlappingChildren** = true
- static const bool **HasSelfChildren** = false
- static const bool **RearrangesDataset** = true
- static const bool **UniqueNumDescendants** = true

39.552.1 Detailed Description

```
template<typename MetricType, typename StatisticType, typename MatType, template< typename SplitBoundType, typename SplitType> class SplitType>  
class mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::CellBound, SplitType > >
```

This is a specialization of the TreeType class to the UBTree tree type.

The only difference with general **BinarySpaceTree** (p.1998) is that UBTree can have overlapping children. See **mlpack/core/tree/tree_traits.hpp** (p.2690) for more information.

Definition at line 235 of file traits.hpp.

39.552.2 Member Data Documentation

39.552.2.1 BinaryTree

```
const bool BinaryTree = true [static]
```

Definition at line 244 of file traits.hpp.

39.552.2.2 FirstPointIsCentroid

```
const bool FirstPointIsCentroid = false [static]
```

Definition at line 241 of file traits.hpp.

39.552.2.3 HasDuplicatedPoints

```
const bool HasDuplicatedPoints = false [static]
```

Definition at line 240 of file traits.hpp.

39.552.2.4 HasOverlappingChildren

```
const bool HasOverlappingChildren = true [static]
```

Definition at line 239 of file traits.hpp.

39.552.2.5 HasSelfChildren

```
const bool HasSelfChildren = false [static]
```

Definition at line 242 of file traits.hpp.

39.552.2.6 RearrangesDataset

```
const bool RearrangesDataset = true [static]
```

Definition at line 243 of file traits.hpp.

39.552.2.7 UniqueNumDescendants

```
const bool UniqueNumDescendants = true [static]
```

Definition at line 245 of file traits.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ **traits.hpp**

39.553 TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::HollowBallBound, SplitType > > Class Template Reference ↩

This is a specialization of the `TreeType` class to an arbitrary tree with `HollowBallBound` (currently only the vantage point tree is supported).

Static Public Attributes

- static const bool **BinaryTree** = true
- static const bool **FirstPointIsCentroid** = false
- static const bool **HasDuplicatedPoints** = false
- static const bool **HasOverlappingChildren** = true
- static const bool **HasSelfChildren** = false
- static const bool **RearrangesDataset** = true
- static const bool **UniqueNumDescendants** = true

39.553.1 Detailed Description

```
template<typename MetricType, typename StatisticType, typename MatType, template< typename SplitBoundType, typename Split↩  
MatType > class SplitType>  
class mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::HollowBallBound, SplitType > >
```

This is a specialization of the `TreeType` class to an arbitrary tree with `HollowBallBound` (currently only the vantage point tree is supported).

The only difference with general **BinarySpaceTree** (p. 1998) is that the tree can have overlapping children.

Definition at line 211 of file traits.hpp.

39.553.2 Member Data Documentation

39.553.2.1 BinaryTree

```
const bool BinaryTree = true [static]
```

Definition at line 220 of file traits.hpp.

39.553.2.2 FirstPointIsCentroid

```
const bool FirstPointIsCentroid = false [static]
```

Definition at line 217 of file traits.hpp.

39.553.2.3 HasDuplicatedPoints

```
const bool HasDuplicatedPoints = false [static]
```

Definition at line 216 of file traits.hpp.

39.553.2.4 HasOverlappingChildren

```
const bool HasOverlappingChildren = true [static]
```

Definition at line 215 of file traits.hpp.

39.553.2.5 HasSelfChildren

```
const bool HasSelfChildren = false [static]
```

Definition at line 218 of file traits.hpp.

39.553.2.6 RearrangesDataset

```
const bool RearrangesDataset = true [static]
```

Definition at line 219 of file traits.hpp.

39.553.2.7 UniqueNumDescendants

```
const bool UniqueNumDescendants = true [static]
```

Definition at line 221 of file traits.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ **traits.hpp**

39.554 TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, RPTreeMaxSplit > > Class Template Reference

This is a specialization of the TreeType class to the max-split random projection tree.

Static Public Attributes

- static const bool **BinaryTree** = true
This is always a binary tree.
- static const bool **FirstPointIsCentroid** = false
There is no guarantee that the first point in a node is its centroid.
- static const bool **HasDuplicatedPoints** = false
The tree has not got duplicated points.
- static const bool **HasOverlappingChildren** = true
Children of a random projection tree node may overlap.
- static const bool **HasSelfChildren** = false
Points are not contained at multiple levels of the binary space tree.
- static const bool **RearrangesDataset** = true
Points are rearranged during building of the tree.
- static const bool **UniqueNumDescendants** = true
Binary space trees don't have duplicated points, so NumDescendants() represents the number of unique descendant points.

39.554.1 Detailed Description

```
template<typename MetricType, typename StatisticType, typename MatType, template< typename BoundMetricType, typename... >
class BoundType>
class mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, RPTreeMaxSplit > >
```

This is a specialization of the `TreeType` class to the max-split random projection tree.

The only difference with general **BinarySpaceTree** (p. 1998) is that the tree can have overlapping children.

Definition at line 85 of file `traits.hpp`.

39.554.2 Member Data Documentation

39.554.2.1 BinaryTree

```
const bool BinaryTree = true [static]
```

This is always a binary tree.

Definition at line 117 of file `traits.hpp`.

39.554.2.2 FirstPointIsCentroid

```
const bool FirstPointIsCentroid = false [static]
```

There is no guarantee that the first point in a node is its centroid.

Definition at line 102 of file `traits.hpp`.

39.554.2.3 HasDuplicatedPoints

```
const bool HasDuplicatedPoints = false [static]
```

The tree has not got duplicated points.

Definition at line 97 of file `traits.hpp`.

39.554.2.4 HasOverlappingChildren

```
const bool HasOverlappingChildren = true [static]
```

Children of a random projection tree node may overlap.

Definition at line 92 of file traits.hpp.

39.554.2.5 HasSelfChildren

```
const bool HasSelfChildren = false [static]
```

Points are not contained at multiple levels of the binary space tree.

Definition at line 107 of file traits.hpp.

39.554.2.6 RearrangesDataset

```
const bool RearrangesDataset = true [static]
```

Points are rearranged during building of the tree.

Definition at line 112 of file traits.hpp.

39.554.2.7 UniqueNumDescendants

```
const bool UniqueNumDescendants = true [static]
```

Binary space trees don't have duplicated points, so NumDescendants() represents the number of unique descendant points.

Definition at line 123 of file traits.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ **traits.hpp**

39.555 TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, RPTreeMeanSplit > > Class Template Reference

This is a specialization of the TreeType class to the mean-split random projection tree.

Static Public Attributes

- static const bool **BinaryTree** = true
This is always a binary tree.
- static const bool **FirstPointIsCentroid** = false
There is no guarantee that the first point in a node is its centroid.
- static const bool **HasDuplicatedPoints** = false
The tree has not got duplicated points.
- static const bool **HasOverlappingChildren** = true
Children of a random projection tree node may overlap.
- static const bool **HasSelfChildren** = false
Points are not contained at multiple levels of the binary space tree.
- static const bool **RearrangesDataset** = true
Points are rearranged during building of the tree.
- static const bool **UniqueNumDescendants** = true
Binary space trees don't have duplicated points, so NumDescendants() represents the number of unique descendant points.

39.555.1 Detailed Description

```
template<typename MetricType, typename StatisticType, typename MatType, template< typename BoundMetricType, typename... >
class BoundType>
class mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, RPTreeMeanSplit > >
```

This is a specialization of the `TreeType` class to the mean-split random projection tree.

The only difference with general **BinarySpaceTree** (p. 1998) is that the tree can have overlapping children.

Definition at line 135 of file `traits.hpp`.

39.555.2 Member Data Documentation

39.555.2.1 BinaryTree

```
const bool BinaryTree = true [static]
```

This is always a binary tree.

Definition at line 167 of file `traits.hpp`.

39.555.2.2 FirstPointIsCentroid

```
const bool FirstPointIsCentroid = false [static]
```

There is no guarantee that the first point in a node is its centroid.

Definition at line 152 of file traits.hpp.

39.555.2.3 HasDuplicatedPoints

```
const bool HasDuplicatedPoints = false [static]
```

The tree has not got duplicated points.

Definition at line 147 of file traits.hpp.

39.555.2.4 HasOverlappingChildren

```
const bool HasOverlappingChildren = true [static]
```

Children of a random projection tree node may overlap.

Definition at line 142 of file traits.hpp.

39.555.2.5 HasSelfChildren

```
const bool HasSelfChildren = false [static]
```

Points are not contained at multiple levels of the binary space tree.

Definition at line 157 of file traits.hpp.

39.555.2.6 RearrangesDataset

```
const bool RearrangesDataset = true [static]
```

Points are rearranged during building of the tree.

Definition at line 162 of file traits.hpp.

39.555.2.7 UniqueNumDescendants

```
const bool UniqueNumDescendants = true [static]
```

Binary space trees don't have duplicated points, so NumDescendants() represents the number of unique descendant points.

Definition at line 173 of file traits.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ **traits.hpp**

39.556 TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > > Class Template Reference

This is a specialization of the **TreeTraits** (p. 2302) class to the **BinarySpaceTree** (p. 1998) tree type.

Static Public Attributes

- static const bool **BinaryTree** = true
This is always a binary tree.
- static const bool **FirstPointIsCentroid** = false
There is no guarantee that the first point in a node is its centroid.
- static const bool **HasDuplicatedPoints** = false
Each binary space tree node doesn't share points with any other node.
- static const bool **HasOverlappingChildren** = false
Each binary space tree node has two children which represent non-overlapping subsets of the space which the node represents.
- static const bool **HasSelfChildren** = false
Points are not contained at multiple levels of the binary space tree.
- static const bool **RearrangesDataset** = true
Points are rearranged during building of the tree.
- static const bool **UniqueNumDescendants** = true
Binary space trees don't have duplicated points, so NumDescendants() represents the number of unique descendant points.

39.556.1 Detailed Description

```
template<typename MetricType, typename StatisticType, typename MatType, template< typename BoundMetricType, typename... >
class BoundType, template< typename SplitBoundType, typename SplitMatType > class SplitType>
class mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > >
```

This is a specialization of the **TreeTraits** (p. 2302) class to the **BinarySpaceTree** (p. 1998) tree type.

It defines characteristics of the binary space tree, and is used to help write tree-independent (but still optimized) tree-based algorithms. See **mlpack/core/tree/tree_traits.hpp** (p. 2690) for more information.

Definition at line 33 of file traits.hpp.

39.556.2 Member Data Documentation

39.556.2.1 BinaryTree

```
const bool BinaryTree = true [static]
```

This is always a binary tree.

Definition at line 67 of file traits.hpp.

39.556.2.2 FirstPointIsCentroid

```
const bool FirstPointIsCentroid = false [static]
```

There is no guarantee that the first point in a node is its centroid.

Definition at line 52 of file traits.hpp.

39.556.2.3 HasDuplicatedPoints

```
const bool HasDuplicatedPoints = false [static]
```

Each binary space tree node doesn't share points with any other node.

Definition at line 47 of file traits.hpp.

39.556.2.4 HasOverlappingChildren

```
const bool HasOverlappingChildren = false [static]
```

Each binary space tree node has two children which represent non-overlapping subsets of the space which the node represents.

Therefore, children are not overlapping.

Definition at line 42 of file traits.hpp.

39.556.2.5 HasSelfChildren

```
const bool HasSelfChildren = false [static]
```

Points are not contained at multiple levels of the binary space tree.

Definition at line 57 of file traits.hpp.

39.556.2.6 RearrangesDataset

```
const bool RearrangesDataset = true [static]
```

Points are rearranged during building of the tree.

Definition at line 62 of file traits.hpp.

39.556.2.7 UniqueNumDescendants

```
const bool UniqueNumDescendants = true [static]
```

Binary space trees don't have duplicated points, so NumDescendants() represents the number of unique descendant points.

Definition at line 73 of file traits.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ **traits.hpp**

39.557 TreeTraits< CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > > Class Template Reference

The specialization of the **TreeTraits** (p. 2302) class for the **CoverTree** (p. 2040) tree type.

Static Public Attributes

- static const bool **BinaryTree** = false
The cover tree is not necessarily a binary tree.
- static const bool **FirstPointIsCentroid** = true
Each cover tree node contains only one point, and that point is its centroid.
- static const bool **HasDuplicatedPoints** = true
Cover trees do have self-children, so points can be included in more than one node.
- static const bool **HasOverlappingChildren** = true
The cover tree (or, this implementation of it) does not require that children represent non-overlapping subsets of the parent node.
- static const bool **HasSelfChildren** = true
Cover trees do have self-children.
- static const bool **RearrangesDataset** = false
Points are not rearranged when the tree is built.
- static const bool **UniqueNumDescendants** = true
NumDescendants() represents the number of unique descendant points.

39.557.1 Detailed Description

```
template<typename MetricType, typename StatisticType, typename MatType, typename RootPointPolicy>
class mlpack::tree::TreeTraits< CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > >
```

The specialization of the **TreeTraits** (p. 2302) class for the **CoverTree** (p. 2040) tree type.

It defines characteristics of the cover tree, and is used to help write tree-independent (but still optimized) tree-based algorithms. See **mlpack/core/tree/tree_traits.hpp** (p. 2690) for more information.

Definition at line 31 of file traits.hpp.

39.557.2 Member Data Documentation

39.557.2.1 BinaryTree

```
const bool BinaryTree = false [static]
```

The cover tree is not necessarily a binary tree.

Definition at line 65 of file traits.hpp.

39.557.2.2 FirstPointIsCentroid

```
const bool FirstPointIsCentroid = true [static]
```

Each cover tree node contains only one point, and that point is its centroid.

Definition at line 50 of file traits.hpp.

39.557.2.3 HasDuplicatedPoints

```
const bool HasDuplicatedPoints = true [static]
```

Cover trees do have self-children, so points can be included in more than one node.

Definition at line 44 of file traits.hpp.

39.557.2.4 HasOverlappingChildren

```
const bool HasOverlappingChildren = true [static]
```

The cover tree (or, this implementation of it) does not require that children represent non-overlapping subsets of the parent node.

Definition at line 38 of file traits.hpp.

39.557.2.5 HasSelfChildren

```
const bool HasSelfChildren = true [static]
```

Cover trees do have self-children.

Definition at line 55 of file traits.hpp.

39.557.2.6 RearrangesDataset

```
const bool RearrangesDataset = false [static]
```

Points are not rearranged when the tree is built.

Definition at line 60 of file traits.hpp.

39.557.2.7 UniqueNumDescendants

```
const bool UniqueNumDescendants = true [static]
```

NumDescendants() represents the number of unique descendant points.

Definition at line 70 of file traits.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/ **traits.hpp**

39.558 TreeTraits< Octree< MetricType, StatisticType, MatType > > Class Template Reference

This is a specialization of the **TreeTraits** (p. 2302) class to the **Octree** (p. 2167) tree type.

Static Public Attributes

- static const bool **BinaryTree** = false
This is not necessarily a binary tree.
- static const bool **FirstPointIsCentroid** = false
There is no guarantee that the first point in a node is its centroid.
- static const bool **HasDuplicatedPoints** = false
Points are not shared across nodes in the octree.
- static const bool **HasOverlappingChildren** = false
No octree nodes will overlap.
- static const bool **HasSelfChildren** = false
Points are not contained at multiple levels of the octree.
- static const bool **RearrangesDataset** = true
Points are rearranged during building of the tree.
- static const bool **UniqueNumDescendants** = true
NumDescendants() represents the number of unique descendant points.

39.558.1 Detailed Description

```
template<typename MetricType, typename StatisticType, typename MatType>
class mlpack::tree::TreeTraits< Octree< MetricType, StatisticType, MatType > >
```

This is a specialization of the **TreeTraits** (p. 2302) class to the **Octree** (p. 2167) tree type.

It defines characteristics of the octree, and is used to help write tree-independent (but still optimized) tree-based algorithms. See **mlpack/core/tree/tree_traits.hpp** (p. 2690) for more information.

Definition at line 29 of file traits.hpp.

39.558.2 Member Data Documentation

39.558.2.1 BinaryTree

```
const bool BinaryTree = false [static]
```

This is not necessarily a binary tree.

Definition at line 60 of file traits.hpp.

39.558.2.2 FirstPointIsCentroid

```
const bool FirstPointIsCentroid = false [static]
```

There is no guarantee that the first point in a node is its centroid.

Definition at line 45 of file traits.hpp.

39.558.2.3 HasDuplicatedPoints

```
const bool HasDuplicatedPoints = false [static]
```

Points are not shared across nodes in the octree.

Definition at line 40 of file traits.hpp.

39.558.2.4 HasOverlappingChildren

```
const bool HasOverlappingChildren = false [static]
```

No octree nodes will overlap.

Definition at line 35 of file traits.hpp.

39.558.2.5 HasSelfChildren

```
const bool HasSelfChildren = false [static]
```

Points are not contained at multiple levels of the octree.

Definition at line 50 of file traits.hpp.

39.558.2.6 RearrangesDataset

```
const bool RearrangesDataset = true [static]
```

Points are rearranged during building of the tree.

Definition at line 55 of file traits.hpp.

39.558.2.7 UniqueNumDescendants

```
const bool UniqueNumDescendants = true [static]
```

NumDescendants() represents the number of unique descendant points.

Definition at line 65 of file traits.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree/ **traits.hpp**

39.559 TreeTraits< RectangleTree< MetricType, StatisticType, MatType, RPlusTreeSplit< SplitPolicyType, SweepType >, DescentType, AuxiliaryInformationType > > Class Template Reference

Since the R+/R++ tree can not have overlapping children, we should define traits for the R+/R++ tree.

Static Public Attributes

- static const bool **BinaryTree** = false
This tree is not necessarily a binary tree.
- static const bool **FirstPointIsCentroid** = false
There is no guarantee that the first point in a node is its centroid.
- static const bool **HasDuplicatedPoints** = false
An R-tree node doesn't share points with another node.
- static const bool **HasOverlappingChildren** = false
The R+/R++ tree can't have overlapping children.
- static const bool **HasSelfChildren** = false
Points are not contained at multiple levels of the R-tree.
- static const bool **RearrangesDataset** = false
Points are rearranged during building of the tree.
- static const bool **UniqueNumDescendants** = true
Rectangle trees don't have duplicated points, so NumDescendants() represents the number of unique descendant points.

39.559.1 Detailed Description

```
template<typename MetricType, typename StatisticType, typename MatType, typename SplitPolicyType, template< typename > class
SweepType, typename DescentType, template< typename > class AuxiliaryInformationType>
class mpack::tree::TreeTraits< RectangleTree< MetricType, StatisticType, MatType, RPlusTreeSplit< SplitPolicyType, SweepType >,
DescentType, AuxiliaryInformationType > >
```

Since the R+/R++ tree can not have overlapping children, we should define traits for the R+/R++ tree.

Definition at line 86 of file traits.hpp.

39.559.2 Member Data Documentation

39.559.2.1 BinaryTree

```
const bool BinaryTree = false [static]
```

This tree is not necessarily a binary tree.

Definition at line 125 of file traits.hpp.

39.559.2.2 FirstPointIsCentroid

```
const bool FirstPointIsCentroid = false [static]
```

There is no guarantee that the first point in a node is its centroid.

Definition at line 108 of file traits.hpp.

39.559.2.3 HasDuplicatedPoints

```
const bool HasDuplicatedPoints = false [static]
```

An R-tree node doesn't share points with another node.

Definition at line 103 of file traits.hpp.

39.559.2.4 HasOverlappingChildren

```
const bool HasOverlappingChildren = false [static]
```

The R+/R++ tree can't have overlapping children.

Definition at line 98 of file traits.hpp.

39.559.2.5 HasSelfChildren

```
const bool HasSelfChildren = false [static]
```

Points are not contained at multiple levels of the R-tree.

Definition at line 113 of file traits.hpp.

39.559.2.6 RearrangesDataset

```
const bool RearrangesDataset = false [static]
```

Points are rearranged during building of the tree.

THIS MAY NOT BE TRUE. IT'S HARD TO DYNAMICALLY INSERT POINTS AND REARRANGE THE MATRIX

Definition at line 120 of file traits.hpp.

39.559.2.7 UniqueNumDescendants

```
const bool UniqueNumDescendants = true [static]
```

Rectangle trees don't have duplicated points, so NumDescendants() represents the number of unique descendant points.

Definition at line 131 of file traits.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ **traits.hpp**

39.560 TreeTraits< RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType > > Class Template Reference

This is a specialization of the TreeType class to the **RectangleTree** (p. 2207) tree type.

Static Public Attributes

- static const bool **BinaryTree** = false
This tree is not necessarily a binary tree.
- static const bool **FirstPointIsCentroid** = false
There is no guarantee that the first point in a node is its centroid.
- static const bool **HasDuplicatedPoints** = false
An R-tree node doesn't share points with another node.
- static const bool **HasOverlappingChildren** = true
An R-tree can have overlapping children.
- static const bool **HasSelfChildren** = false
Points are not contained at multiple levels of the R-tree.
- static const bool **RearrangesDataset** = false
Points are rearranged during building of the tree.
- static const bool **UniqueNumDescendants** = true
Rectangle trees don't have duplicated points, so NumDescendants() represents the number of unique descendant points.

39.560.1 Detailed Description

```
template<typename MetricType, typename StatisticType, typename MatType, typename SplitType, typename DescentType, template<
typename > class AuxiliaryInformationType>
class mlpack::tree::TreeTraits< RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType
> >
```

This is a specialization of the TreeType class to the **RectangleTree** (p. 2207) tree type.

It defines characteristics of the rectangle type trees, and is used to help write tree-independent (but still optimized) tree-based algorithms. See **mlpack/core/tree/tree_traits.hpp** (p. 2690) for more information.

Definition at line 32 of file traits.hpp.

39.560.2 Member Data Documentation

39.560.2.1 BinaryTree

```
const bool BinaryTree = false [static]
```

This tree is not necessarily a binary tree.

Definition at line 66 of file traits.hpp.

39.560.2.2 FirstPointIsCentroid

```
const bool FirstPointIsCentroid = false [static]
```

There is no guarantee that the first point in a node is its centroid.

Definition at line 49 of file traits.hpp.

39.560.2.3 HasDuplicatedPoints

```
const bool HasDuplicatedPoints = false [static]
```

An R-tree node doesn't share points with another node.

Definition at line 44 of file traits.hpp.

39.560.2.4 HasOverlappingChildren

```
const bool HasOverlappingChildren = true [static]
```

An R-tree can have overlapping children.

Definition at line 39 of file traits.hpp.

39.560.2.5 HasSelfChildren

```
const bool HasSelfChildren = false [static]
```

Points are not contained at multiple levels of the R-tree.

Definition at line 54 of file traits.hpp.

39.560.2.6 RearrangesDataset

```
const bool RearrangesDataset = false [static]
```

Points are rearranged during building of the tree.

THIS MAY NOT BE TRUE. IT'S HARD TO DYNAMICALLY INSERT POINTS AND REARRANGE THE MATRIX

Definition at line 61 of file traits.hpp.

39.560.2.7 UniqueNumDescendants

```
const bool UniqueNumDescendants = true [static]
```

Rectangle trees don't have duplicated points, so NumDescendants() represents the number of unique descendant points.

Definition at line 72 of file traits.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ **traits.hpp**

39.561 TreeTraits< SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > > Class Template Reference

This is a specialization of the TreeType class to the **SpillTree** (p. 2274) tree type.

Static Public Attributes

- static const bool **BinaryTree** = true
This is always a binary tree.
- static const bool **FirstPointIsCentroid** = false
There is no guarantee that the first point in a node is its centroid.
- static const bool **HasOverlappingChildren** = true
Each spill tree node has two children which can share points.
- static const bool **HasSelfChildren** = false
Points are not contained at multiple levels of the spill tree.
- static const bool **RearrangesDataset** = false
Points are not rearranged during building of the tree.
- static const bool **UniqueNumDescendants** = false
Spill trees have duplicated points, so NumDescendants() could count a given point twice.

39.561.1 Detailed Description

```
template<typename MetricType, typename StatisticType, typename MatType, template< typename HyperplaneMetricType > class
HyperplaneType, template< typename SplitMetricType, typename SplitMatType > class SplitType>
class mpack::tree::TreeTraits< SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > >
```

This is a specialization of the TreeType class to the **SpillTree** (p. 2274) tree type.

It defines characteristics of the spill tree, and is used to help write tree-independent (but still optimized) tree-based algorithms. See **mlpack/core/tree/tree_traits.hpp** (p. 2690) for more information.

Definition at line 33 of file traits.hpp.

39.561.2 Member Data Documentation

39.561.2.1 BinaryTree

```
const bool BinaryTree = true [static]
```

This is always a binary tree.

Definition at line 61 of file traits.hpp.

39.561.2.2 FirstPointIsCentroid

```
const bool FirstPointIsCentroid = false [static]
```

There is no guarantee that the first point in a node is its centroid.

Definition at line 46 of file traits.hpp.

39.561.2.3 HasOverlappingChildren

```
const bool HasOverlappingChildren = true [static]
```

Each spill tree node has two children which can share points.

Therefore, children can be overlapping.

Definition at line 41 of file traits.hpp.

39.561.2.4 HasSelfChildren

```
const bool HasSelfChildren = false [static]
```

Points are not contained at multiple levels of the spill tree.

Definition at line 51 of file traits.hpp.

39.561.2.5 RearrangesDataset

```
const bool RearrangesDataset = false [static]
```

Points are not rearranged during building of the tree.

Definition at line 56 of file traits.hpp.

39.561.2.6 UniqueNumDescendants

```
const bool UniqueNumDescendants = false [static]
```

Spill trees have duplicated points, so NumDescendants() could count a given point twice.

Definition at line 67 of file traits.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/ **traits.hpp**

39.562 UBTreeSplit< **BoundType**, **MatType** > **Class Template Reference**

Split a node into two parts according to the median address of points contained in the node.

39.562.1 Detailed Description

```
template<typename BoundType, typename MatType = arma::mat>
class mlpack::tree::UBTreeSplit< BoundType, MatType >
```

Split a node into two parts according to the median address of points contained in the node.

The class reorders the dataset such that points with lower addresses belong to the left subtree and points with high addresses belong to the right subtree.

Definition at line 29 of file ub_tree_split.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ **ub_tree_split.hpp**

39.563 VantagePointSplit< **BoundType**, **MatType**, **MaxNumSamples** > **Class Template Reference**

The class splits a binary space partitioning tree node according to the median distance to the vantage point.

Classes

- struct **SplitInfo**
A struct that contains an information about the split.

Public Types

- typedef MatType::elem_type **ElemType**
The matrix element type.
- typedef BoundType::MetricType **MetricType**
The bounding shape type.

Static Public Member Functions

- template<typename VecType >
static bool **AssignToLeftNode** (const VecType &point, const **SplitInfo** &splitInfo)
Indicates that a point should be assigned to the left subtree.
- static size_t **PerformSplit** (MatType &data, const size_t begin, const size_t count, const **SplitInfo** &splitInfo)
Perform the split process according to the information about the split.
- static size_t **PerformSplit** (MatType &data, const size_t begin, const size_t count, const **SplitInfo** &splitInfo, std::vector< size_t > &oldFromNew)
Perform the split process according to the information about the split and return the list of changed indices.
- static bool **SplitNode** (const BoundType &bound, MatType &data, const size_t begin, const size_t count, **SplitInfo** &splitInfo)
Split the node according to the distance to a vantage point.

39.563.1 Detailed Description

```
template<typename BoundType, typename MatType = arma::mat, size_t MaxNumSamples = 100>
class mlpack::tree::VantagePointSplit< BoundType, MatType, MaxNumSamples >
```

The class splits a binary space partitioning tree node according to the median distance to the vantage point.

Thus points that are closer to the vantage point belong to the left subtree and points that are farther from the vantage point belong to the right subtree.

Definition at line 32 of file vantage_point_split.hpp.

39.563.2 Member Typedef Documentation

39.563.2.1 ElemType

```
typedef MatType::elem_type ElemType
```

The matrix element type.

Definition at line 36 of file vantage_point_split.hpp.

39.563.2.2 MetricType

```
typedef BoundType::MetricType MetricType
```

The bounding shape type.

Definition at line 38 of file `vantage_point_split.hpp`.

39.563.3 Member Function Documentation

39.563.3.1 AssignToLeftNode()

```
static bool AssignToLeftNode (
    const VecType & point,
    const SplitInfo & splitInfo ) [inline], [static]
```

Indicates that a point should be assigned to the left subtree.

This method returns true if a point should be assigned to the left subtree, i.e., if the distance from the point to the vantage point is less then the median value. Otherwise it returns false.

Parameters

<i>point</i>	The point that is being assigned.
<i>splitInfo</i>	An information about the split.

Definition at line 138 of file `vantage_point_split.hpp`.

References `VantagePointSplit< BoundType, MatType, MaxNumSamples >::SplitInfo::metric`, `VantagePointSplit< BoundType, MatType, MaxNumSamples >::SplitInfo::mu`, and `VantagePointSplit< BoundType, MatType, MaxNumSamples >::SplitInfo::vantagePoint`.

39.563.3.2 PerformSplit() [1/2]

```
static size_t PerformSplit (
    MatType & data,
    const size_t begin,
    const size_t count,
    const SplitInfo & splitInfo ) [inline], [static]
```

Perform the split process according to the information about the split.

This will order the dataset such that points that belong to the left subtree are on the left of the split column, and points from the right subtree are on the right side of the split column.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitInfo</i>	The information about the split.

Definition at line 93 of file `vantage_point_split.hpp`.

39.563.3.3 PerformSplit() [2/2]

```
static size_t PerformSplit (
    MatType & data,
    const size_t begin,
    const size_t count,
    const SplitInfo & splitInfo,
    std::vector< size_t > & oldFromNew ) [inline], [static]
```

Perform the split process according to the information about the split and return the list of changed indices.

This will order the dataset such that points that belong to the left subtree are on the left of the split column, and points from the right subtree are on the right side of the split column.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitInfo</i>	The information about the split.
<i>oldFromNew</i>	Vector which will be filled with the old positions for each new point.

Definition at line 118 of file `vantage_point_split.hpp`.

39.563.3.4 SplitNode()

```
static bool SplitNode (
    const BoundType & bound,
    MatType & data,
    const size_t begin,
```

```
const size_t count,  
    SplitInfo & splitInfo ) [static]
```

Split the node according to the distance to a vantage point.

Parameters

<i>bound</i>	The bound used for this node.
<i>data</i>	The dataset used by the binary space tree.
<i>begin</i>	Index of the starting point in the dataset that belongs to this node.
<i>count</i>	Number of points in this node.
<i>splitInfo</i>	An information about the split. This information contains the vantage point and the median distance to the vantage point.

Referenced by `VantagePointSplit< BoundType, MatType, MaxNumSamples >::SplitInfo::SplitInfo()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ vantage_point_split.hpp`

39.564 VantagePointSplit< BoundType, MatType, MaxNumSamples >::SplitInfo Struct Reference

A struct that contains an information about the split.

Public Member Functions

- **SplitInfo** ()
- `template<typename VecType >`
SplitInfo (const **MetricType** & **metric**, const VecType & **vantagePoint**, **ElemType** **mu**)

Public Attributes

- const **MetricType** * **metric**
An instance of the MetricType class.
- **ElemType** **mu**
The median distance according to which the node will be split.
- `arma::Col< ElemType >` **vantagePoint**
The vantage point.

39.564.1 Detailed Description

```
template<typename BoundType, typename MatType = arma::mat, size_t MaxNumSamples = 100>
struct mlpack::tree::VantagePointSplit< BoundType, MatType, MaxNumSamples >::SplitInfo
```

A struct that contains an information about the split.

Definition at line 40 of file `vantage_point_split.hpp`.

39.564.2 Constructor & Destructor Documentation

39.564.2.1 SplitInfo() [1/2]

```
SplitInfo ( ) [inline]
```

Definition at line 49 of file vantage_point_split.hpp.

39.564.2.2 SplitInfo() [2/2]

```
SplitInfo (
    const MetricType & metric,
    const VecType & vantagePoint,
    ElemType mu ) [inline]
```

Definition at line 55 of file vantage_point_split.hpp.

References VantagePointSplit< BoundType, MatType, MaxNumSamples >::SplitNode().

39.564.3 Member Data Documentation

39.564.3.1 metric

```
const MetricType* metric
```

An instance of the MetricType class.

Definition at line 47 of file vantage_point_split.hpp.

Referenced by VantagePointSplit< BoundType, MatType, MaxNumSamples >::AssignToLeftNode().

39.564.3.2 mu

```
ElemType mu
```

The median distance according to which the node will be split.

Definition at line 45 of file vantage_point_split.hpp.

Referenced by VantagePointSplit< BoundType, MatType, MaxNumSamples >::AssignToLeftNode().

39.564.3.3 vantagePoint

```
arma::Col< ElemType> vantagePoint
```

The vantage point.

Definition at line 43 of file vantage_point_split.hpp.

Referenced by VantagePointSplit< BoundType, MatType, MaxNumSamples >::AssignToLeftNode().

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ **vantage_point_split.hpp**

39.565 XTreeAuxiliaryInformation< TreeType > Class Template Reference

The **XTreeAuxiliaryInformation** (p. 2338) class provides information specific to X trees for each node in a **Rectangle**↩
Tree (p. 2207).

Classes

- struct **SplitHistoryStruct**
The X tree requires that the tree records it's "split history".

Public Types

- typedef struct **mlpack::tree::XTreeAuxiliaryInformation::SplitHistoryStruct** **SplitHistoryStruct**
The X tree requires that the tree records it's "split history".

Public Member Functions

- **XTreeAuxiliaryInformation** ()
Default constructor.
- **XTreeAuxiliaryInformation** (const TreeType *node)
Construct this with the specified node.
- **XTreeAuxiliaryInformation** (const **XTreeAuxiliaryInformation** &other, TreeType *=NULL, bool=true)
Create an auxiliary information object by copying from another object.
- **XTreeAuxiliaryInformation** (**XTreeAuxiliaryInformation** &&other)
Create an auxiliary information object by moving from the other node.
- bool **HandleNodeInsertion** (TreeType *, TreeType *, bool)
Some tree types require to save some properties at the insertion process.
- bool **HandleNodeRemoval** (TreeType *, const size_t)
Some tree types require to save some properties at the deletion process.
- bool **HandlePointDeletion** (TreeType *, const size_t)

Some tree types require to save some properties at the deletion process.

- bool **HandlePointInsertion** (TreeType *, const size_t)

Some tree types require to save some properties at the insertion process.

- size_t **NormalNodeMaxNumChildren** () const

Return the maximum number of a normal node's children.

- size_t & **NormalNodeMaxNumChildren** ()

Modify the maximum number of a normal node's children.

- void **NullifyData** ()

Nullify the auxiliary information in order to prevent an invalid free.

- **XTreeAuxiliaryInformation** & **operator=** (const **XTreeAuxiliaryInformation** &other)

Copy the auxiliary information object.

- template<typename Archive >

void **serialize** (Archive &ar, const unsigned int)

Serialize the information.

- const **SplitHistoryStruct** & **SplitHistory** () const

Return the split history of the node associated with this object.

- **SplitHistoryStruct** & **SplitHistory** ()

Modify the split history of the node associated with this object.

- bool **UpdateAuxiliaryInfo** (TreeType *)

Some tree types require to propagate the information upward.

39.565.1 Detailed Description

```
template<typename TreeType>
class mlpack::tree::XTreeAuxiliaryInformation< TreeType >
```

The **XTreeAuxiliaryInformation** (p. 2338) class provides information specific to X trees for each node in a **RectangleTree** (p. 2207).

Definition at line 24 of file x_tree_auxiliary_information.hpp.

39.565.2 Member Typedef Documentation

39.565.2.1 SplitHistoryStruct

```
typedef struct mlpack::tree::XTreeAuxiliaryInformation::SplitHistoryStruct SplitHistoryStruct
```

The X tree requires that the tree records it's "split history".

To make this easy, we use the following structure.

39.565.3 Constructor & Destructor Documentation

39.565.3.1 XTreeAuxiliaryInformation() [1/4]

```
XTreeAuxiliaryInformation ( ) [inline]
```

Default constructor.

Definition at line 28 of file `x_tree_auxiliary_information.hpp`.

39.565.3.2 XTreeAuxiliaryInformation() [2/4]

```
XTreeAuxiliaryInformation (
    const TreeType * node ) [inline]
```

Construct this with the specified node.

Parameters

<i>node</i>	The node that stores this auxiliary information.
-------------	--

Definition at line 38 of file `x_tree_auxiliary_information.hpp`.

39.565.3.3 XTreeAuxiliaryInformation() [3/4]

```
XTreeAuxiliaryInformation (
    const XTreeAuxiliaryInformation< TreeType > & other,
    TreeType *   = NULL,
    bool   = true ) [inline]
```

Create an auxiliary information object by copying from another object.

Parameters

<i>other</i>	Another auxiliary information object from which the information will be copied.
<i>tree</i>	The node that holds the auxiliary information.
<i>deepCopy</i>	If false, the new object uses the same memory (not used here).

Definition at line 54 of file `x_tree_auxiliary_information.hpp`.

39.565.3.4 XTreeAuxiliaryInformation() [4/4]

```
XTreeAuxiliaryInformation (
    XTreeAuxiliaryInformation< TreeType > && other ) [inline]
```

Create an auxiliary information object by moving from the other node.

Parameters

<i>other</i>	The object from which the information will be moved.
--------------	--

Definition at line 79 of file x_tree_auxiliary_information.hpp.

39.565.4 Member Function Documentation

39.565.4.1 HandleNodeInsertion()

```
bool HandleNodeInsertion (
    TreeType * ,
    TreeType * ,
    bool ) [inline]
```

Some tree types require to save some properties at the insertion process.

This method allows the auxiliary information the option of manipulating the tree in order to perform the insertion process. If the auxiliary information does that, then the method should return true; if the method returns false the **RectangleTree** (p. 2207) performs its default behavior.

Parameters

<i>node</i>	The node in which the nodeToInsert is being inserted.
<i>nodeToInsert</i>	The node being inserted.
<i>insertionLevel</i>	The level of the tree at which the nodeToInsert should be inserted.

Definition at line 113 of file x_tree_auxiliary_information.hpp.

39.565.4.2 HandleNodeRemoval()

```
bool HandleNodeRemoval (
    TreeType * ,
    const size_t ) [inline]
```

Some tree types require to save some properties at the deletion process.

This method allows the auxiliary information the option of manipulating the tree in order to perform the deletion process. If the auxiliary information does that, then the method should return true; if the method returns false the **RectangleTree** (p. 2207) performs its default behavior.

Parameters

<i>node</i>	The node from which the node is being deleted.
<i>nodeIndex</i>	The local index of the node being deleted.

Definition at line 143 of file `x_tree_auxiliary_information.hpp`.

39.565.4.3 HandlePointDeletion()

```
bool HandlePointDeletion (
    TreeType * ,
    const size_t ) [inline]
```

Some tree types require to save some properties at the deletion process.

This method allows the auxiliary information the option of manipulating the tree in order to perform the deletion process. If the auxiliary information does that, then the method should return true; if the method returns false the **RectangleTree** (p. 2207) performs its default behavior.

Parameters

<i>node</i>	The node from which the point is being deleted.
<i>localIndex</i>	The local index of the point being deleted.

Definition at line 129 of file `x_tree_auxiliary_information.hpp`.

39.565.4.4 HandlePointInsertion()

```
bool HandlePointInsertion (
    TreeType * ,
    const size_t ) [inline]
```

Some tree types require to save some properties at the insertion process.

This method allows the auxiliary information the option of manipulating the tree in order to perform the insertion process. If the auxiliary information does that, then the method should return true; if the method returns false the **RectangleTree** (p. 2207) performs its default behavior.

Parameters

<i>node</i>	The node in which the point is being inserted.
<i>point</i>	The global number of the point being inserted.

Definition at line 96 of file x_tree_auxiliary_information.hpp.

39.565.4.5 NormalNodeMaxNumChildren() [1/2]

```
size_t NormalNodeMaxNumChildren ( ) const [inline]
```

Return the maximum number of a normal node's children.

Definition at line 215 of file x_tree_auxiliary_information.hpp.

Referenced by XTreeAuxiliaryInformation< TreeType >::operator=().

39.565.4.6 NormalNodeMaxNumChildren() [2/2]

```
size_t& NormalNodeMaxNumChildren ( ) [inline]
```

Modify the maximum number of a normal node's children.

Definition at line 217 of file x_tree_auxiliary_information.hpp.

39.565.4.7 NullifyData()

```
void NullifyData ( ) [inline]
```

Nullify the auxiliary information in order to prevent an invalid free.

Definition at line 162 of file x_tree_auxiliary_information.hpp.

39.565.4.8 operator=()

```
XTreeAuxiliaryInformation& operator= (
    const XTreeAuxiliaryInformation< TreeType > & other ) [inline]
```

Copy the auxiliary information object.

Parameters

<i>other</i>	The node from which the information will be copied.
--------------	---

Definition at line 66 of file `x_tree_auxiliary_information.hpp`.

References `XTreeAuxiliaryInformation< TreeType >::NormalNodeMaxNumChildren()`, and `XTreeAuxiliaryInformation< TreeType >::SplitHistory()`.

39.565.4.9 `serialize()`

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Serialize the information.

Definition at line 227 of file `x_tree_auxiliary_information.hpp`.

39.565.4.10 `SplitHistory()` [1/2]

```
const SplitHistoryStruct& SplitHistory ( ) const [inline]
```

Return the split history of the node associated with this object.

Definition at line 219 of file `x_tree_auxiliary_information.hpp`.

Referenced by `XTreeAuxiliaryInformation< TreeType >::operator=()`.

39.565.4.11 `SplitHistory()` [2/2]

```
SplitHistoryStruct& SplitHistory ( ) [inline]
```

Modify the split history of the node associated with this object.

Definition at line 221 of file `x_tree_auxiliary_information.hpp`.

39.565.4.12 `UpdateAuxiliaryInfo()`

```
bool UpdateAuxiliaryInfo (
    TreeType * ) [inline]
```

Some tree types require to propagate the information upward.

This method should return false if this is not the case. If true is returned, the update will be propagated upward.

Parameters

<i>node</i>	The node in which the auxiliary information being update.
-------------	---

Definition at line 154 of file `x_tree_auxiliary_information.hpp`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ x_tree_auxiliary_information.hpp`

39.566 XTreeAuxiliaryInformation< TreeType >::SplitHistoryStruct Struct Reference

The X tree requires that the tree records it's "split history".

Public Member Functions

- **SplitHistoryStruct** (int dim)
- **SplitHistoryStruct** (const **SplitHistoryStruct** &other)
- **SplitHistoryStruct** (**SplitHistoryStruct** &&other)
- **SplitHistoryStruct** & **operator=** (const **SplitHistoryStruct** &other)
- `template<typename Archive >`
void **serialize** (Archive &ar, const unsigned int)

Public Attributes

- `std::vector< bool >` **history**
- int **lastDimension**

39.566.1 Detailed Description

```
template<typename TreeType>
struct mlpack::tree::XTreeAuxiliaryInformation< TreeType >::SplitHistoryStruct
```

The X tree requires that the tree records it's "split history".

To make this easy, we use the following structure.

Definition at line 169 of file `x_tree_auxiliary_information.hpp`.

39.566.2 Constructor & Destructor Documentation

39.566.2.1 SplitHistoryStruct() [1/3]

```
SplitHistoryStruct (  
    int dim ) [inline]
```

Definition at line 174 of file `x_tree_auxiliary_information.hpp`.

Referenced by `XTreeAuxiliaryInformation< TreeType >::SplitHistoryStruct::serialize()`.

39.566.2.2 SplitHistoryStruct() [2/3]

```
SplitHistoryStruct (  
    const SplitHistoryStruct & other ) [inline]
```

Definition at line 180 of file `x_tree_auxiliary_information.hpp`.

39.566.2.3 SplitHistoryStruct() [3/3]

```
SplitHistoryStruct (  
    SplitHistoryStruct && other ) [inline]
```

Definition at line 192 of file `x_tree_auxiliary_information.hpp`.

39.566.3 Member Function Documentation**39.566.3.1 operator=()**

```
SplitHistoryStruct& operator= (  
    const SplitHistoryStruct & other ) [inline]
```

Definition at line 185 of file `x_tree_auxiliary_information.hpp`.

References `XTreeAuxiliaryInformation< TreeType >::SplitHistoryStruct::history`, and `XTreeAuxiliaryInformation< TreeType >::SplitHistoryStruct::lastDimension`.

39.566.3.2 serialize()

```
void serialize (
    Archive & ar,
    const unsigned int ) [inline]
```

Definition at line 200 of file `x_tree_auxiliary_information.hpp`.

References `XTreeAuxiliaryInformation< TreeType >::SplitHistoryStruct::SplitHistoryStruct()`.

39.566.4 Member Data Documentation

39.566.4.1 history

```
std::vector<bool> history
```

Definition at line 172 of file `x_tree_auxiliary_information.hpp`.

Referenced by `XTreeAuxiliaryInformation< TreeType >::SplitHistoryStruct::operator=()`.

39.566.4.2 lastDimension

```
int lastDimension
```

Definition at line 171 of file `x_tree_auxiliary_information.hpp`.

Referenced by `XTreeAuxiliaryInformation< TreeType >::SplitHistoryStruct::operator=()`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ x_tree_auxiliary_information.hpp`

39.567 XTreeSplit Class Reference

A Rectangle Tree has new points inserted at the bottom.

Static Public Member Functions

- `template<typename TreeType >`
`static void SplitLeafNode (TreeType *tree, std::vector< bool > &relevels)`
Split a leaf node using the algorithm described in "The R-tree: An Efficient and Robust Access method for Points and Rectangles."*
- `template<typename TreeType >`
`static bool SplitNonLeafNode (TreeType *tree, std::vector< bool > &relevels)`
Split a non-leaf node using the "default" algorithm.

39.567.1 Detailed Description

A Rectangle Tree has new points inserted at the bottom.

When these nodes overflow, we split them, moving up the tree and splitting nodes as necessary.

Definition at line 36 of file `x_tree_split.hpp`.

39.567.2 Member Function Documentation

39.567.2.1 SplitLeafNode()

```
static void SplitLeafNode (
    TreeType * tree,
    std::vector< bool > & relevels ) [static]
```

Split a leaf node using the algorithm described in "The R*-tree: An Efficient and Robust Access method for Points and Rectangles."

" If necessary, this split will propagate upwards through the tree.

39.567.2.2 SplitNonLeafNode()

```
static bool SplitNonLeafNode (
    TreeType * tree,
    std::vector< bool > & relevels ) [static]
```

Split a non-leaf node using the "default" algorithm.

If this is a root node, the tree increases in depth.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/ x_tree_split.hpp`

39.568 `IsStdVector< T >` Struct Template Reference

Metaprogramming structure for vector detection.

Static Public Attributes

- static const bool **value** = false

39.568.1 Detailed Description

```
template<typename T>
struct mpack::util::IsStdVector< T >
```

Metaprogramming structure for vector detection.

Definition at line 23 of file `is_std_vector.hpp`.

39.568.2 Member Data Documentation

39.568.2.1 **value**

```
const bool value = false [static]
```

Definition at line 23 of file `is_std_vector.hpp`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ is_std_vector.hpp`

39.569 `IsStdVector< std::vector< T, A > >` Struct Template Reference

Metaprogramming structure for vector detection.

Static Public Attributes

- static const bool **value** = true

39.569.1 Detailed Description

```
template<typename T, typename A>
struct mpack::util::IsStdVector< std::vector< T, A > >
```

Metaprogramming structure for vector detection.

Definition at line 27 of file `is_std_vector.hpp`.

39.569.2 Member Data Documentation

39.569.2.1 value

```
const bool value = true [static]
```

Definition at line 27 of file `is_std_vector.hpp`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ is_std_vector.hpp`

39.570 NullOutputStream Class Reference

Used for **Log::Debug** (p. 1540) when not compiled with debugging symbols.

Public Member Functions

- **NullOutputStream** ()
Does nothing.
- **NullOutputStream** (const **NullOutputStream** &)
Does nothing.
- **NullOutputStream** & **operator**<< (bool)
Does nothing.
- **NullOutputStream** & **operator**<< (short)
Does nothing.
- **NullOutputStream** & **operator**<< (unsigned short)
Does nothing.
- **NullOutputStream** & **operator**<< (int)
Does nothing.
- **NullOutputStream** & **operator**<< (unsigned int)
Does nothing.

- **NullOutputStream & operator<<** (long)
Does nothing.
- **NullOutputStream & operator<<** (unsigned long)
Does nothing.
- **NullOutputStream & operator<<** (float)
Does nothing.
- **NullOutputStream & operator<<** (double)
Does nothing.
- **NullOutputStream & operator<<** (long double)
Does nothing.
- **NullOutputStream & operator<<** (void *)
Does nothing.
- **NullOutputStream & operator<<** (const char *)
Does nothing.
- **NullOutputStream & operator<<** (std::string &)
Does nothing.
- **NullOutputStream & operator<<** (std::streambuf *)
Does nothing.
- **NullOutputStream & operator<<** (std::ostream &*)(std::ostream &)
Does nothing.
- **NullOutputStream & operator<<** (std::ios &*)(std::ios &)
Does nothing.
- **NullOutputStream & operator<<** (std::ios_base &*)(std::ios_base &)
Does nothing.
- **template<typename T >**
NullOutputStream & operator<< (const T &)
Does nothing.

39.570.1 Detailed Description

Used for **Log::Debug** (p. 1540) when not compiled with debugging symbols.

This class does nothing and should be optimized out entirely by the compiler.

Definition at line 27 of file nullostream.hpp.

39.570.2 Constructor & Destructor Documentation

39.570.2.1 NullOutputStream() [1/2]

```
NullOutputStream ( ) [inline]
```

Does nothing.

Definition at line 33 of file nullostream.hpp.

39.570.2.2 `NullOutputStream()` [2/2]

```
NullOutputStream (  
    const NullOutputStream & ) [inline]
```

Does nothing.

Definition at line 38 of file `nullostream.hpp`.

39.570.3 **Member Function Documentation****39.570.3.1** `operator<<()` [1/18]

```
NullOutputStream& operator<< (  
    bool ) [inline]
```

Does nothing.

Definition at line 41 of file `nullostream.hpp`.

39.570.3.2 `operator<<()` [2/18]

```
NullOutputStream& operator<< (  
    short ) [inline]
```

Does nothing.

Definition at line 43 of file `nullostream.hpp`.

39.570.3.3 `operator<<()` [3/18]

```
NullOutputStream& operator<< (  
    unsigned short ) [inline]
```

Does nothing.

Definition at line 45 of file `nullostream.hpp`.

39.570.3.4 `operator<<()` [4/18]

```
NullOutputStream& operator<< (  
    int ) [inline]
```

Does nothing.

Definition at line 47 of file nullostream.hpp.

39.570.3.5 `operator<<()` [5/18]

```
NullOutputStream& operator<< (  
    unsigned int ) [inline]
```

Does nothing.

Definition at line 49 of file nullostream.hpp.

39.570.3.6 `operator<<()` [6/18]

```
NullOutputStream& operator<< (  
    long ) [inline]
```

Does nothing.

Definition at line 51 of file nullostream.hpp.

39.570.3.7 `operator<<()` [7/18]

```
NullOutputStream& operator<< (  
    unsigned long ) [inline]
```

Does nothing.

Definition at line 53 of file nullostream.hpp.

39.570.3.8 `operator<<()` [8/18]

```
NullOutputStream& operator<< (  
    float ) [inline]
```

Does nothing.

Definition at line 55 of file nullostream.hpp.

39.570.3.9 `operator<<()` [9/18]

```
NullOutputStream& operator<< (  
    double ) [inline]
```

Does nothing.

Definition at line 57 of file nullostream.hpp.

39.570.3.10 `operator<<()` [10/18]

```
NullOutputStream& operator<< (  
    long double ) [inline]
```

Does nothing.

Definition at line 59 of file nullostream.hpp.

39.570.3.11 `operator<<()` [11/18]

```
NullOutputStream& operator<< (  
    void * ) [inline]
```

Does nothing.

Definition at line 61 of file nullostream.hpp.

39.570.3.12 `operator<<()` [12/18]

```
NullOutputStream& operator<< (  
    const char * ) [inline]
```

Does nothing.

Definition at line 63 of file nullostream.hpp.

39.570.3.13 `operator<<()` [13/18]

```
NullOutputStream& operator<< (  
    std::string & ) [inline]
```

Does nothing.

Definition at line 65 of file nullostream.hpp.

39.570.3.14 `operator<<()` [14/18]

```
NullOutputStream& operator<< (  
    std::streambuf * ) [inline]
```

Does nothing.

Definition at line 67 of file nullostream.hpp.

39.570.3.15 `operator<<()` [15/18]

```
NullOutputStream& operator<< (  
    std::ostream & *) (std::ostream & ) [inline]
```

Does nothing.

Definition at line 69 of file nullostream.hpp.

39.570.3.16 `operator<<()` [16/18]

```
NullOutputStream& operator<< (  
    std::ios & *) (std::ios & ) [inline]
```

Does nothing.

Definition at line 71 of file nullostream.hpp.

39.570.3.17 `operator<<()` [17/18]

```
NullOutputStream& operator<< (  
    std::ios_base & *) (std::ios_base & ) [inline]
```

Does nothing.

Definition at line 73 of file nullostream.hpp.

39.570.3.18 `operator<<()` [18/18]

```
NullOutputStream& operator<< (  
    const T & ) [inline]
```

Does nothing.

Definition at line 78 of file nullostream.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **nullostream.hpp**

39.571 ParamData Struct Reference

This structure holds all of the information about a single parameter, including its value (which is set when **ParseCommandLine()** (p.302) is called).

Public Attributes

- char **alias**
Alias for this parameter.
- std::string **cppType**
The true name of the type, as it would be written in C++.
- std::string **desc**
Description of this parameter, if any.
- bool **input**
True if this option is an input option (otherwise, it is output).
- bool **loaded**
If this is an input parameter that needs extra loading, this indicates whether or not it has been loaded.
- std::string **name**
Name of this parameter.
- bool **noTranspose**
True if this is a matrix that should not be transposed.
- bool **persistent**
If this should be preserved across different settings (i.e.
- bool **required**
True if this option is required.
- std::string **tname**
Type information of this parameter.
- boost::any **value**
The actual value that is held.
- bool **wasPassed**
True if the option was passed to the program.

39.571.1 Detailed Description

This structure holds all of the information about a single parameter, including its value (which is set when **Parse**↔**CommandLine()** (p. 302) is called).

It does not hold any information about whether or not it was passed—that is handled elsewhere. A **ParamData** (p. 2356) struct is only useful in order to get "static" information about a parameter. Note that some parameter types have internal types but also different types that are used by boost::program_options (specifically, matrix and model types map to strings).

This structure is somewhat unwieldy and is likely to be refactored at some point in the future, but for now it does the job fine.

Definition at line 52 of file param_data.hpp.

39.571.2 Member Data Documentation

39.571.2.1 alias

```
char alias
```

Alias for this parameter.

Definition at line 63 of file param_data.hpp.

Referenced by `mlpack::bindings::cli::AddToPO()`, `CLIOption< N >::CLIOption()`, `MDOption< T >::MDOption()`, `PyOption< T >::PyOption()`, and `TestOption< N >::TestOption()`.

39.571.2.2 cppType

```
std::string cppType
```

The true name of the type, as it would be written in C++.

Definition at line 84 of file param_data.hpp.

Referenced by `CLIOption< N >::CLIOption()`, `mlpack::bindings::python::GetCythonType()`, `mlpack::bindings::tests::GetPrintableParam()`, `mlpack::bindings::python::GetPrintableParam()`, `mlpack::bindings::python::ImportDecl()`, `MDOption< T >::MDOption()`, `mlpack::bindings::python::PrintClassDefn()`, `mlpack::bindings::python::PrintDoc()`, `mlpack::bindings::python::PrintInputProcessing()`, `mlpack::bindings::python::PrintOutputProcessing()`, `PyOption< T >::PyOption()`, and `TestOption< N >::TestOption()`.

39.571.2.3 desc

```
std::string desc
```

Description of this parameter, if any.

Definition at line 58 of file param_data.hpp.

Referenced by `mlpack::bindings::cli::AddToPO()`, `CLIOption< N >::CLIOption()`, `MDOption< T >::MDOption()`, `mlpack::bindings::python::PrintDoc()`, `PyOption< T >::PyOption()`, and `TestOption< N >::TestOption()`.

39.571.2.4 input

```
bool input
```

True if this option is an input option (otherwise, it is output).

Definition at line 73 of file param_data.hpp.

Referenced by `CLIOption< N >::CLIOption()`, `mlpack::bindings::cli::EndProgram()`, `mlpack::bindings::cli::GetParam()`, `MDOption< T >::MDOption()`, `mlpack::bindings::python::PrintOutputProcessing()`, `PyOption< T >::PyOption()`, and `TestOption< N >::TestOption()`.

39.571.2.5 loaded

```
bool loaded
```

If this is an input parameter that needs extra loading, this indicates whether or not it has been loaded.

Definition at line 76 of file param_data.hpp.

Referenced by `CLIOption< N >::CLIOption()`, `mlpack::bindings::cli::GetParam()`, `MDOption< T >::MDOption()`, `PyOption< T >::PyOption()`, and `TestOption< N >::TestOption()`.

39.571.2.6 name

```
std::string name
```

Name of this parameter.

This is the name used for `HasParam()` and **`GetParam()`** (p. 290).

Definition at line 56 of file param_data.hpp.

Referenced by `mlpack::bindings::cli::AddToPO()`, `CLIOption< N >::CLIOption()`, `mlpack::bindings::cli::MapParameterName()`, `MDOption< T >::MDOption()`, `mlpack::bindings::cli::ParseCommandLine()`, `mlpack::bindings::python::PrintDefn()`, `mlpack::bindings::python::PrintDoc()`, `mlpack::bindings::python::PrintInputProcessing()`, `mlpack::bindings::python::PrintOutputProcessing()`, `PyOption< T >::PyOption()`, and `TestOption< N >::TestOption()`.

39.571.2.7 noTranspose

```
bool noTranspose
```

True if this is a matrix that should not be transposed.

Ignored if the parameter is not a matrix.

Definition at line 69 of file param_data.hpp.

Referenced by `CLIOption< N >::CLIOption()`, `mlpack::bindings::cli::GetParam()`, `MDOption< T >::MDOption()`, `PyOption< T >::PyOption()`, and `TestOption< N >::TestOption()`.

39.571.2.8 persistent

```
bool persistent
```

If this should be preserved across different settings (i.e.

if it should exist for every binding), this should be set to true.

Definition at line 79 of file param_data.hpp.

Referenced by `CLIOption< N >::CLIOption()`, `MDOption< T >::MDOption()`, `PyOption< T >::PyOption()`, and `TestOption< N >::TestOption()`.

39.571.2.9 required

```
bool required
```

True if this option is required.

Definition at line 71 of file param_data.hpp.

Referenced by `CLIOption< N >::CLIOption()`, `MDOption< T >::MDOption()`, `mlpack::bindings::cli::ParseCommandLine()`, `mlpack::bindings::python::PrintDefn()`, `mlpack::bindings::python::PrintDoc()`, `mlpack::bindings::python::PrintInputProcessing()`, `mlpack::bindings::python::PrintOutputProcessing()`, `PyOption< T >::PyOption()`, and `TestOption< N >::TestOption()`.

39.571.2.10 tname

```
std::string tname
```

Type information of this parameter.

Note that this is **TYPENAME()** (p. 2738) of the user-visible parameter type, not whatever is given by `ParameterType<>`.

Definition at line 61 of file param_data.hpp.

Referenced by `CLIOption< N >::CLIOption()`, `mlpack::bindings::cli::EndProgram()`, `MDOption< T >::MDOption()`, `mlpack::bindings::cli::ParseCommandLine()`, `PyOption< T >::PyOption()`, and `TestOption< N >::TestOption()`.

39.571.2.11 value

```
boost::any value
```

The actual value that is held.

If the user has passed a different type, this may be a tuple containing multiple values.

Definition at line 82 of file param_data.hpp.

Referenced by `CLIOption< N >::CLIOption()`, `mlpack::bindings::cli::DeleteAllocatedMemoryImpl()`, `mlpack::bindings::tests::DeleteAllocatedMemoryImpl()`, `mlpack::bindings::cli::GetAllocatedMemory()`, `mlpack::bindings::tests::GetAllocatedMemory()`, `mlpack::bindings::python::GetParam()`, `mlpack::bindings::tests::GetParam()`, `mlpack::bindings::markdown::GetParam()`, `mlpack::bindings::cli::GetParam()`, `mlpack::bindings::python::GetPrintableParam()`, `mlpack::bindings::tests::GetPrintableParam()`, `mlpack::bindings::cli::GetRawParam()`, `MDOption< T >::MDOption()`, `mlpack::bindings::cli::ParseCommandLine()`, `PyOption< T >::PyOption()`, `mlpack::bindings::cli::SetParam()`, and `TestOption< N >::TestOption()`.

39.571.2.12 wasPassed

```
bool wasPassed
```

True if the option was passed to the program.

Note that `wasPassed` may be set by either **`ParseCommandLine()`** (p. 302) or `SetPassed()`.

Definition at line 66 of file param_data.hpp.

Referenced by `CLIOption< N >::CLIOption()`, `MDOption< T >::MDOption()`, `mlpack::bindings::cli::ParseCommandLine()`, `PyOption< T >::PyOption()`, `mlpack::bindings::cli::SetParam()`, and `TestOption< N >::TestOption()`.

The documentation for this struct was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ param_data.hpp`

39.572 PrefixedOutputStream Class Reference

Allows us to output to an ostream with a prefix at the beginning of each line, in the same way we would output to cout or cerr.

Public Member Functions

- **PrefixOutputStream** (std::ostream & **destination**, const char *prefix, bool **ignoreInput**=false, bool fatal=false, bool **backtrace**=true)
*Set up the **PrefixOutputStream** (p. 2361).*
- **PrefixOutputStream** & **operator**<< (bool val)
Write a bool to the stream.
- **PrefixOutputStream** & **operator**<< (short val)
Write a short to the stream.
- **PrefixOutputStream** & **operator**<< (unsigned short val)
Write an unsigned short to the stream.
- **PrefixOutputStream** & **operator**<< (int val)
Write an int to the stream.
- **PrefixOutputStream** & **operator**<< (unsigned int val)
Write an unsigned int to the stream.
- **PrefixOutputStream** & **operator**<< (long val)
Write a long to the stream.
- **PrefixOutputStream** & **operator**<< (unsigned long val)
Write an unsigned long to the stream.
- **PrefixOutputStream** & **operator**<< (float val)
Write a float to the stream.
- **PrefixOutputStream** & **operator**<< (double val)
Write a double to the stream.
- **PrefixOutputStream** & **operator**<< (long double val)
Write a long double to the stream.
- **PrefixOutputStream** & **operator**<< (void *val)
Write a void pointer to the stream.
- **PrefixOutputStream** & **operator**<< (const char *str)
Write a character array to the stream.
- **PrefixOutputStream** & **operator**<< (std::string &str)
Write a string to the stream.
- **PrefixOutputStream** & **operator**<< (std::streambuf *sb)
Write a streambuf to the stream.
- **PrefixOutputStream** & **operator**<< (std::ostream &(*pf)(std::ostream &))
Write an ostream manipulator function to the stream.
- **PrefixOutputStream** & **operator**<< (std::ios &(*pf)(std::ios &))
Write an ios manipulator function to the stream.
- **PrefixOutputStream** & **operator**<< (std::ios_base &(*pf)(std::ios_base &))
Write an ios_base manipulator function to the stream.
- template<typename T >
PrefixOutputStream & **operator**<< (const T &s)
Write anything else to the stream.

Public Attributes

- bool **backtrace**
If true, on a fatal error, a backtrace will be printed if HAS_BFD_DL is defined.
- std::ostream & **destination**
The output stream that all data is to be sent to; example: std::cout.
- bool **ignoreInput**
Discards input, prints nothing if true.

39.572.1 Detailed Description

Allows us to output to an ostream with a prefix at the beginning of each line, in the same way we would output to cout or cerr.

The prefix is specified in the constructor (as well as the destination ostream). A newline must be passed to the stream, and then the prefix will be prepended to the next line. For example,

```
PrefixedOutputStream ostr(std::cout, "[TEST] ");
ostr << "Hello world I like " << 7.5;
ostr << "...Continue" << std::endl;
ostr << "After the CR\n" << std::endl;
```

would give, on std::cout,

```
[TEST] Hello world I like 7.5...Continue
[TEST] After the CR
[TEST]
```

These objects are used for the **mlpack::Log** (p. 1538) levels (DEBUG, INFO, WARN, and FATAL).

Definition at line 46 of file prefixedostream.hpp.

39.572.2 Constructor & Destructor Documentation

39.572.2.1 PrefixedOutputStream()

```
PrefixedOutputStream (  
    std::ostream & destination,  
    const char * prefix,  
    bool ignoreInput = false,  
    bool fatal = false,  
    bool backtrace = true ) [inline]
```

Set up the **PrefixedOutputStream** (p. 2361).

Parameters

<i>destination</i>	ostream which receives output from this object.
<i>prefix</i>	The prefix to prepend to each line.
<i>ignoreInput</i>	If true, the stream will not be printed.
<i>fatal</i>	If true, a <code>std::runtime_error</code> exception is thrown after printing a newline.
<i>backtrace</i>	If true, attempt to print a backtrace (will only be done if <code>HAS_BFD_DL</code> is defined).

Definition at line 60 of file `prefixedostream.hpp`.

References `PrefixedOutputStream::operator<<()`.

39.572.3 Member Function Documentation

39.572.3.1 `operator<<()` [1/18]

```
PrefixedOutputStream& operator<< (  
    bool val )
```

Write a bool to the stream.

Referenced by `PrefixedOutputStream::PrefixedOutputStream()`.

39.572.3.2 `operator<<()` [2/18]

```
PrefixedOutputStream& operator<< (  
    short val )
```

Write a short to the stream.

39.572.3.3 `operator<<()` [3/18]

```
PrefixedOutputStream& operator<< (  
    unsigned short val )
```

Write an unsigned short to the stream.

39.572.3.4 operator<<() [4/18]

```
PrefixedOutputStream& operator<< (  
    int val )
```

Write an int to the stream.

39.572.3.5 operator<<() [5/18]

```
PrefixedOutputStream& operator<< (  
    unsigned int val )
```

Write an unsigned int to the stream.

39.572.3.6 operator<<() [6/18]

```
PrefixedOutputStream& operator<< (  
    long val )
```

Write a long to the stream.

39.572.3.7 operator<<() [7/18]

```
PrefixedOutputStream& operator<< (  
    unsigned long val )
```

Write an unsigned long to the stream.

39.572.3.8 operator<<() [8/18]

```
PrefixedOutputStream& operator<< (  
    float val )
```

Write a float to the stream.

39.572.3.9 `operator<<()` [9/18]

```
PrefixOutputStream& operator<< (  
    double val )
```

Write a double to the stream.

39.572.3.10 `operator<<()` [10/18]

```
PrefixOutputStream& operator<< (  
    long double val )
```

Write a long double to the stream.

39.572.3.11 `operator<<()` [11/18]

```
PrefixOutputStream& operator<< (  
    void * val )
```

Write a void pointer to the stream.

39.572.3.12 `operator<<()` [12/18]

```
PrefixOutputStream& operator<< (  
    const char * str )
```

Write a character array to the stream.

39.572.3.13 `operator<<()` [13/18]

```
PrefixOutputStream& operator<< (  
    std::string & str )
```

Write a string to the stream.

39.572.3.14 `operator<<()` [14/18]

```
PrefixedOutputStream& operator<< (  
    std::streambuf * sb )
```

Write a streambuf to the stream.

39.572.3.15 `operator<<()` [15/18]

```
PrefixedOutputStream& operator<< (  
    std::ostream &(*) (std::ostream &) pf )
```

Write an ostream manipulator function to the stream.

39.572.3.16 `operator<<()` [16/18]

```
PrefixedOutputStream& operator<< (  
    std::ios &(*) (std::ios &) pf )
```

Write an ios manipulator function to the stream.

39.572.3.17 `operator<<()` [17/18]

```
PrefixedOutputStream& operator<< (  
    std::ios_base &(*) (std::ios_base &) pf )
```

Write an ios_base manipulator function to the stream.

39.572.3.18 `operator<<()` [18/18]

```
PrefixedOutputStream& operator<< (  
    const T & s )
```

Write anything else to the stream.

39.572.4 Member Data Documentation

39.572.4.1 `backtrace`

```
bool backtrace
```

If true, on a fatal error, a backtrace will be printed if `HAS_BFD_DL` is defined.

Definition at line 122 of file `prefixedostream.hpp`.

Referenced by `mlpack::util::DisableBacktrace()`.

39.572.4.2 `destination`

```
std::ostream& destination
```

The output stream that all data is to be sent to; example: `std::cout`.

Definition at line 115 of file `prefixedostream.hpp`.

39.572.4.3 `ignoreInput`

```
bool ignoreInput
```

Discards input, prints nothing if true.

Definition at line 118 of file `prefixedostream.hpp`.

Referenced by `mlpack::util::DisableVerbose()`, `mlpack::util::EnableVerbose()`, and `mlpack::bindings::cli::ParseCommandLine()`.

The documentation for this class was generated from the following file:

- `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ prefixedostream.hpp`

39.573 ProgramDoc Class Reference

A static object whose constructor registers program documentation with the **CLI** (p. 1117) class.

Public Member Functions

- **ProgramDoc** (const std::string & **programName**, const std::string & **shortDocumentation**, const std::function< std::string()> & **documentation**, const std::vector< std::pair< std::string, std::string >> & **seeAlso**)
Construct a **ProgramDoc** (p. 2368) object.
- **ProgramDoc** ()
Construct an empty **ProgramDoc** (p. 2368) object.

Public Attributes

- `std::function< std::string()>` **documentation**
Documentation for what the program does.
- `std::string` **programName**
The name of the program.
- `std::vector< std::pair< std::string, std::string > >` **seeAlso**
Set of see also information.
- `std::string` **shortDocumentation**
The short documentation for the program.

39.573.1 Detailed Description

A static object whose constructor registers program documentation with the **CLI** (p. 1117) class.

This should not be used outside of **CLI** (p. 1117) itself, and you should use the **PROGRAM_INFO()** (p. 2733) macro to declare these objects. Only one **ProgramDoc** (p. 2368) object should ever exist.

See also

core/util/cli.hpp (p. 2694), **mlpack::CLI** (p. 1117)

Definition at line 26 of file `program_doc.hpp`.

39.573.2 Constructor & Destructor Documentation

39.573.2.1 ProgramDoc() [1/2]

```
ProgramDoc (
    const std::string & programName,
    const std::string & shortDocumentation,
    const std::function< std::string()> & documentation,
    const std::vector< std::pair< std::string, std::string >> & seeAlso )
```

Construct a **ProgramDoc** (p. 2368) object.

When constructed, it will register itself with **CLI** (p. 1117), and when the user calls `--help` (or whatever the option is named for the given binding type), the given function that returns a `std::string` will be returned.

Parameters

<i>programName</i>	Short string representing the name of the program.
<i>shortDocumentation</i>	A short two-sentence description of the program, what it does, and what it is useful for.
<i>documentation</i>	Long string containing documentation on how to use the program and what it is. No newline characters are necessary; this is taken care of by CLI (p. 1117) later.
Generated by Doxygen <i>seeAlso</i>	A set of pairs of strings with useful "see also" information; each pair is <code><description, url></code> .

39.573.2.2 ProgramDoc() [2/2]

ProgramDoc ()

Construct an empty **ProgramDoc** (p. 2368) object.

(This is not meant to be used!)

39.573.3 Member Data Documentation

39.573.3.1 documentation

`std::function<std::string()> documentation`

Documentation for what the program does.

Definition at line 59 of file program_doc.hpp.

39.573.3.2 programName

`std::string programName`

The name of the program.

Definition at line 55 of file program_doc.hpp.

39.573.3.3 seeAlso

`std::vector<std::pair<std::string, std::string> > seeAlso`

Set of see also information.

Definition at line 61 of file program_doc.hpp.

39.573.3.4 shortDocumentation

```
std::string shortDocumentation
```

The short documentation for the program.

Definition at line 57 of file program_doc.hpp.

The documentation for this class was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/ **program_doc.hpp**

39.574 TrainHMMModel Struct Reference

Static Public Member Functions

- template<typename HMMType >
static void **Apply** (HMMType &hmm, vector< arma::mat > *trainSeq)

39.574.1 Detailed Description

Definition at line 202 of file hmm_test_utils.hpp.

39.574.2 Member Function Documentation

39.574.2.1 Apply()

```
static void Apply (  
    HMMType & hmm,  
    vector< arma::mat > * trainSeq ) [inline], [static]
```

Definition at line 205 of file hmm_test_utils.hpp.

The documentation for this struct was generated from the following file:

- /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/main_tests/ **hmm_test_utils.hpp**

Chapter 40

File Documentation

40.1 `/home/barak/src/git/debian-src/mlpack/doc/guide/bindings.hpp` File Reference

40.2 `/home/barak/src/git/debian-src/mlpack/doc/guide/build.hpp` File Reference

40.3 `/home/barak/src/git/debian-src/mlpack/doc/guide/build_windows.hpp` File Reference

40.3.1 Detailed Description

Author

German Lancioni
Miguel Canteras
Shikhar Jaiswal

40.4 `/home/barak/src/git/debian-src/mlpack/doc/guide/cli_quickstart.hpp` File Reference

40.4.1 Detailed Description

Author

Ryan Curtin

40.5 `/home/barak/src/git/debian-src/mlpack/doc/guide/cv.hpp` File Reference

Namespaces

- **mlpack**
 .hpp
- **mlpack::cv**

40.6 /home/barak/src/git/debian-src/mlpack/doc/guide/formats.hpp File Reference

40.6.1 Detailed Description

Author

Ryan Curtin

Define the formats that can be used by mlpack's **Load()** (p.379) and **Save()** (p.386) mechanisms via **boost**↔
::serialization (p.251).

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause>↔ for more information.

40.7 /home/barak/src/git/debian-src/mlpack/doc/guide/iodoc.hpp File Reference

40.8 /home/barak/src/git/debian-src/mlpack/doc/guide/matrices.hpp File Reference

40.9 /home/barak/src/git/debian-src/mlpack/doc/guide/python_quickstart.hpp File Reference

40.9.1 Detailed Description

Author

Ryan Curtin

40.10 /home/barak/src/git/debian-src/mlpack/doc/guide/sample.hpp File Reference

40.11 /home/barak/src/git/debian-src/mlpack/doc/guide/sample_ml_app.hpp File Reference

40.11.1 Detailed Description

Author

German Lancioni

40.12 /home/barak/src/git/debian-src/mlpack/doc/guide/timer.hpp File Reference

40.13 /home/barak/src/git/debian-src/mlpack/doc/policies/elemtype.hpp File Reference

40.14 /home/barak/src/git/debian-src/mlpack/doc/policies/functiontype.hpp File Reference

40.15 /home/barak/src/git/debian-src/mlpack/doc/policies/kernels.hpp File Reference

40.16 /home/barak/src/git/debian-src/mlpack/doc/policies/metrics.hpp File Reference

40.17 /home/barak/src/git/debian-src/mlpack/doc/policies/trees.hpp File Reference

40.18 /home/barak/src/git/debian-src/mlpack/doc/tutorials/amf/amf.txt File Reference

Tutorial for how to use the AMF class.

40.18.1 Detailed Description

Tutorial for how to use the AMF class.

Author

Sumedh Ghaisas

40.19 /home/barak/src/git/debian-src/mlpack/doc/tutorials/ann/ann.txt File Reference

Tutorial for how to use the neural network code in mlpack.

40.19.1 Detailed Description

Tutorial for how to use the neural network code in mlpack.

Author

Marcus Edel (<https://kurg.org>)

40.20 /home/barak/src/git/debian-src/mlpack/doc/tutorials/approx_kfn/approx_kfn.txt File Reference

Tutorial for how to use approximate furthest neighbor search in mlpack.

40.20.1 Detailed Description

Tutorial for how to use approximate furthest neighbor search in mlpack.

Author

Ryan Curtin

40.21 /home/barak/src/git/debian-src/mlpack/doc/tutorials/cf/cf.txt File Reference

Tutorial for how to use the CF class and program.

40.21.1 Detailed Description

Tutorial for how to use the CF class and program.

Author

Ryan Curtin

40.22 /home/barak/src/git/debian-src/mlpack/doc/tutorials/det/det.txt File Reference

Tutorial for how to perform density estimation with Density Estimation Trees (DET).

Functions

- $V(t)$ is the volume of the node t and \tilde{V}
- see subsection cli_alt_reg_tut Alternate DET **regularization** The usual regularized error $R_\alpha(t)$ of a node t is given by

Variables

- f is the **set** of leaves in the subtree rooted at t For the purposes of density **estimation**
- this option is not available in DET right **now**
- f is the **set** of leaves in the subtree rooted at t For the purposes of density there is a different form of **regularization**
- f is the **set** of leaves in the subtree rooted at t For the purposes of density there is a different form of we penalize the sum of the inverse of the volumes of the leaves With this very small volume nodes are discouraged unless the data actually warrants it **Thus**

40.22.1 Detailed Description

Tutorial for how to perform density estimation with Density Estimation Trees (DET).

Author

Parikshit Ram

40.22.2 Function Documentation

40.22.2.1 `$V()`

```
f f $V (
      t )
```

Definition at line 351 of file det.txt.

40.22.2.2 `alpha()`

```
see subsection cli_alt_reg_tut Alternate DET regularization The usual regularized error f $R_{\leftarrow}
alpha (
      t )
```

Definition at line 344 of file det.txt.

40.22.3 Variable Documentation

40.22.3.1 `estimation`

`f` is the **set** of leaves in the subtree rooted at `f` `$t f` For the purposes of density estimation

Definition at line 354 of file det.txt.

40.22.3.2 now

this option is not available in DET right now

Definition at line 341 of file det.txt.

40.22.3.3 regularization

f is the **set** of leaves in the subtree rooted at f $\$t f$ For the purposes of density there is a different form of we penalize the sum of the inverse of the volumes of the leaves With this regularization

Definition at line 354 of file det.txt.

40.22.3.4 Thus

f is the **set** of leaves in the subtree rooted at f $\$t f$ For the purposes of density there is a different form of we penalize the sum of the inverse of the volumes of the leaves With this very small volume nodes are discouraged unless the data actually warrants it Thus

Definition at line 354 of file det.txt.

40.23 /home/barak/src/git/debian-src/mlpack/doc/tutorials/emst/emst.txt File Reference

Tutorial for the Euclidean Minimum Spanning Tree algorithm.

40.23.1 Detailed Description

Tutorial for the Euclidean Minimum Spanning Tree algorithm.

Author

Bill March

40.24 /home/barak/src/git/debian-src/mlpack/doc/tutorials/fastmks/fastmks.txt File Reference

Tutorial for how to use FastMKS in mlpack.

40.24.1 Detailed Description

Tutorial for how to use FastMKS in mlpack.

Author

Ryan Curtin

40.25 /home/barak/src/git/debian-src/mlpack/doc/tutorials/kmeans/kmeans.txt File Reference

Tutorial for how to use k-means in mlpack.

40.25.1 Detailed Description

Tutorial for how to use k-means in mlpack.

Author

Ryan Curtin

40.26 /home/barak/src/git/debian-src/mlpack/doc/tutorials/linear_regression/linear_regression.txt File Reference

Tutorial for how to use the LinearRegression class.

40.26.1 Detailed Description

Tutorial for how to use the LinearRegression class.

Author

James Cline

40.27 /home/barak/src/git/debian-src/mlpack/doc/tutorials/neighbor_search/neighbor_search.txt File Reference

Tutorial for how to use the NeighborSearch class.

40.27.1 Detailed Description

Tutorial for how to use the NeighborSearch class.

Author

Ryan Curtin

40.28 `/home/barak/src/git/debian-src/mlpack/doc/tutorials/range_search/range_search.txt` File Reference

Tutorial for how to use the RangeSearch class.

40.28.1 Detailed Description

Tutorial for how to use the RangeSearch class.

Author

Ryan Curtin

40.29 `/home/barak/src/git/debian-src/mlpack/doc/tutorials/tutorials.txt` File Reference

List of mlpack tutorials.

40.29.1 Detailed Description

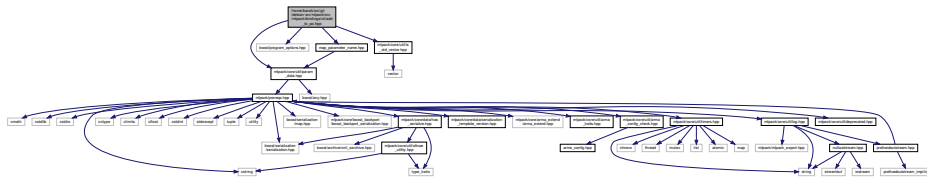
List of mlpack tutorials.

Author

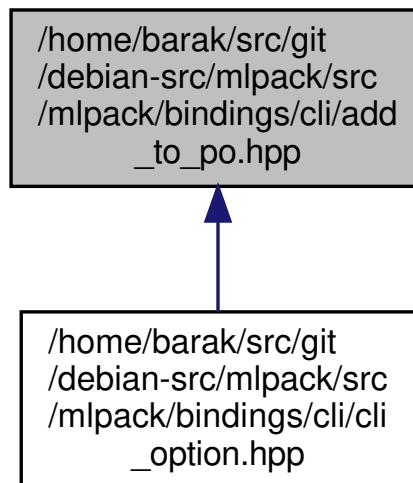
Ryan Curtin

40.30 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/add_to_po.hpp File Reference

Include dependency graph for add_to_po.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::cli**

Functions

- `template<typename T >`
`void AddToPO (const std::string &boostName, const std::string &descr, boost::program_options::options_↵`
`_description &desc, const typename boost::disable_if< util::IsStdVector< T >>::type *=0, const typename`
`boost::disable_if< std::is_same< T, bool >>::type *=0)`
Add a non-vector option to boost::program_options.

Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::cli**

Macros

- **#define BASH_CLEAR** "\033[0m"
- **#define BASH_RED** "\033[0;31m"

40.31.1 Macro Definition Documentation

40.31.1.1 BASH_CLEAR

```
#define BASH_CLEAR "\033[0m"
```

Referenced by `CLIOption< N >::CLIOption()`.

40.31.1.2 BASH_RED

```
#define BASH_RED "\033[0;31m"
```

Referenced by `CLIOption< N >::CLIOption()`.

40.32 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/CMakeLists.txt File Reference

Functions

- **set** (SOURCES add_to_po.hpp cli_option.hpp default_param.hpp default_param_impl.hpp delete_allocated_↵
memory.hpp end_program.hpp get_allocated_memory.hpp get_param.hpp get_raw_param.hpp get_printable_↵
_param.hpp get_printable_param_impl.hpp get_printable_param_name.hpp get_printable_param_name_impl.↵
hpp get_printable_param_value.hpp get_printable_param_value_impl.hpp map_parameter_name.hpp output_↵
_param.hpp output_param_impl.hpp parameter_type.hpp parse_command_line.hpp print_doc_functions.hpp
print_doc_functions_impl.hpp print_help.hpp print_help.cpp print_type_doc.hpp print_type_doc_impl.hpp set_↵
param.hpp string_type_param.hpp string_type_param_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS
\$

40.32.1 Function Documentation

40.32.1.1 `set()` [1/2]

```
set (
    SOURCES add_to_po.hpp cli_option.hpp default_param.hpp default_param_impl.hpp delete_↵
    _allocated_memory.hpp end_program.hpp get_allocated_memory.hpp get_param.hpp get_raw_param.hpp
    get_printable_param.hpp get_printable_param_impl.hpp get_printable_param_name.hpp get_printable_↵
    param_name_impl.hpp get_printable_param_value.hpp get_printable_param_value_impl.hpp map_parameter_↵
    _name.hpp output_param.hpp output_param_impl.hpp parameter_type.hpp parse_command_line.hpp print_↵
    _doc_functions.hpp print_doc_functions_impl.hpp print_help.hpp print_help.cpp print_type_doc.hpp
    print_type_doc_impl.hpp set_param.hpp string_type_param.hpp string_type_param_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

Referenced by `set()`.

40.32.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 38 of file CMakeLists.txt.

References `add_cli_executable()`, `add_executable()`, `BINDING_TYPE`, `BINDING_TYPE_CLI`, `endif()`, and `macro()`.

40.33 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/CMakeLists.txt File Reference

Functions

- **add_subdirectory** (`${dir}`) **endforeach()** **set**(`MARKDOWN_CATEGORIES` `$`
- **set** (`DIRS cli markdown python tests`) **foreach**(`dir` `$`
- `PARENT_SCOPE` **set** (`MLPACK_SRCS` `${MLPACK_SRCS}` `PARENT_SCOPE`) **set**(`MLPACK_PYXS` `$`

40.33.1 Function Documentation

40.33.1.1 `add_subdirectory()`

```
add_subdirectory (
    ${dir} )
```

Definition at line 10 of file CMakeLists.txt.

Referenced by `include_directories()`, and `set()`.

40.33.1.2 `set()` [1/2]

```
set (
    DIRS cli markdown python tests )
```

Definition at line 2 of file CMakeLists.txt.

40.33.1.3 `set()` [2/2]

```
PARENT_SCOPE set (
    MLPACK_SRCS ${MLPACK_SRCS} PARENT_SCOPE )
```

Definition at line 14 of file CMakeLists.txt.

40.34 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/CMakeLists.txt File Reference

Functions

- **macro** (`not_found_return` message) `message(STATUS "$`
- **macro** (`add_markdown_docs` name languages category) `endmacro()` `function(post_markdown_setup)` `endfunction()` `return()` `endmacro()` `if(NOT BUILD_MARKDOWN_BINDINGS)` `not_found_return("Not building Markdown bindings.")` `endif()` `include(MarkdownCategories.cmake)` **set**(`MARKDOWN_CATEGORIES` \$
- `PARENT_SCOPE` **set** (`SOURCES` "binding_info.hpp" "binding_info.cpp" "default_param.hpp" "get_binding_↵
name.hpp" "get_binding_name.cpp" "get_param.hpp" "get_printable_param.hpp" "get_printable_param_name.↵
hpp" "get_printable_param_name_impl.hpp" "get_printable_type.hpp" "md_option.hpp" "print_doc_functions.↵
hpp" "print_doc_functions_impl.hpp" "print_docs.hpp" "print_docs.cpp" "print_type_doc.hpp" "program_doc_↵
wrapper.hpp") `add_custom_target(markdown_copy)` `add_custom_command(TARGET markdown_copy PRE_↵
BUILD COMMAND $`

40.34.1 Function Documentation

40.34.1.1 macro() [1/2]

```
macro (
    not_found_return message )
```

Definition at line 1 of file CMakeLists.txt.

Referenced by `endif()`, and `set()`.

40.34.1.2 macro() [2/2]

```
macro (
    add_markdown_docs name languages category )
```

Definition at line 4 of file CMakeLists.txt.

40.34.1.3 set()

```
PARENT_SCOPE set (
    SOURCES "binding_info.hpp" "binding_info.cpp" "default_param.hpp" "get_binding_↵
name.hpp" "get_binding_name.cpp" "get_param.hpp" "get_printable_param.hpp" "get_printable_param_↵
_name.hpp" "get_printable_param_name_impl.hpp" "get_printable_type.hpp" "md_option.hpp" "print_↵
doc_functions.hpp" "print_doc_functions_impl.hpp" "print_docs.hpp" "print_docs.cpp" "print_type_↵
doc.hpp" "program_doc_wrapper.hpp" )
```

Definition at line 28 of file CMakeLists.txt.

References `add_executable()`, `BINDING_TYPE_MARKDOWN`, `endif()`, `macro()`, and `set()`.

40.35 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/CMakeLists.txt

File Reference

Functions

- not building Python bindings **else** () `message(STATUS "Found Python`
- **endif** () `include($`
- CMake `FindPythonModule` `cmake` **find_python_module** (`distutils`) `if(NOT PY_DISTUTILS)` `not_found_↵`
`return("distutils not found`
- **macro** (`not_found_return message`) `message(STATUS "$`
- **macro** (`add_python_binding name`) `endmacro()` `return()` `endmacro()` `if(NOT BUILD_PYTHON_BINDINGS)` `not_↵`
`_found_return("Not building Python bindings.")` `endif()` `find_package(PythonInterp)` `if(NOT PYTHON_EXECUT_↵`
`ABLE)` `not_found_return("Python not found`
- `PROPERTY INCLUDE_DIRECTORIES` **set** (`CYTHON_INCLDIRS "${CYTHON_INCLUDE_DIRECTORIES}"`)
`if(DEBUG)` `set(DISABLE_CFLAGS "NDEBUG`

40.35.1 Function Documentation

40.35.1.1 `else()`

```
not building Python bindings else ( )
```

Definition at line 20 of file CMakeLists.txt.

40.35.1.2 `endif()`

```
HAS_BFD_DL PARENT_SCOPE endif ( )
```

Definition at line 22 of file CMakeLists.txt.

References `add_executable()`, `BINDING_TYPE_PYX`, and `macro()`.

Referenced by `add_executable()`, `include_directories()`, and `set()`.

40.35.1.3 `find_python_module()`

```
CMake FindPythonModule cmake find_python_module (
    distutils )
```

40.35.1.4 `macro()` [1/2]

```
macro (
    not_found_return message )
```

Definition at line 1 of file CMakeLists.txt.

40.35.1.5 `macro()` [2/2]

```
macro (
    add_python_binding name )
```

40.35.1.6 `set()`

```
PROPERTY INCLUDE_DIRECTORIES set (
    CYTHON_INCLDIRS "${CYTHON_INCLUDE_DIRECTORIES}" )
```

40.36 `/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/tests/CMakeLists.txt` File Reference40.37 `/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/CMakeLists.txt` File Reference

Functions

- **set** (SOURCES clean_memory.hpp clean_memory.cpp test_option.hpp ignore_check.hpp delete_allocated_memory.hpp get_allocated_memory.hpp get_param.hpp get_printable_param.hpp get_printable_param_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.37.1 Function Documentation

40.37.1.1 `set()` [1/2]

```
set (
    SOURCES clean_memory.hpp clean_memory.cpp test_option.hpp ignore_check.hpp delete_
allocated_memory.hpp get_allocated_memory.hpp get_param.hpp get_printable_param.hpp get_printable_
_param_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.37.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 18 of file CMakeLists.txt.

40.38 /home/barak/src/git/debian-src/mlpack/src/mlpack/CMakeLists.txt File Reference

Functions

- **include_directories** ($\text{\${CMAKE_CURRENT_BINARY_DIR}/..}$) **set**(MLPACK_SRCS \$

40.38.1 Function Documentation

40.38.1.1 include_directories()

```
include_directories (
     $\text{\${CMAKE\_CURRENT\_BINARY\_DIR}/..}$  )
```

Definition at line 1 of file CMakeLists.txt.

References `add_subdirectory()`, `endif()`, `MLPACK_VERSION_MAJOR`, `MLPACK_VERSION_MINOR`, and `MLPACK_VERSION_PATCH`.

40.39 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/CMakeLists.txt File Reference

Functions

- **add_subdirectory** ($\text{\${dir}}$) **endforeach**() **set**(MLPACK_SRCS \$
- **set** (DIRS arma_extend boost_backport cv data dists hpt kernels math metrics tree util) **foreach**(dir \$

40.39.1 Function Documentation

40.39.1.1 add_subdirectory()

```
add_subdirectory (
     $\text{\${dir}}$  )
```

Definition at line 17 of file CMakeLists.txt.

40.39.1.2 `set()`

```
set (
    DIRS arma_extend boost_backport cv data dists hpt kernels math metrics tree util )
```

Definition at line 2 of file CMakeLists.txt.

40.40 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/CMakeLists.txt File Reference

Functions

- **add_subdirectory** (metrics) **set**(SOURCES cv_base.hpp cv_base_impl.hpp k_fold_cv.hpp k_fold_cv_impl.hpp meta_info_extractor.hpp simple_cv.hpp simple_cv_impl.hpp) **set**(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.40.1 Function Documentation

40.40.1.1 `add_subdirectory()`

```
add_subdirectory (
    metrics )
```

Definition at line 1 of file CMakeLists.txt.

40.40.1.2 `set()`

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 18 of file CMakeLists.txt.

40.41 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/CMakeLists.txt File Reference

Functions

- **set** (SOURCES accuracy.hpp accuracy_impl.hpp average_strategy.hpp f1.hpp f1_impl.hpp facilities.hpp mse.hpp mse_impl.hpp precision.hpp precision_impl.hpp recall.hpp recall_impl.hpp) **set**(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.41.1 Function Documentation

40.41.1.1 `set()` [1/2]

```
set (
    SOURCES accuracy.hpp accuracy_impl.hpp average_strategy.hpp fl.hpp fl_impl.hpp facilities.↵
    hpp mse.hpp mse_impl.hpp precision.hpp precision_impl.hpp recall.hpp recall_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.41.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 21 of file CMakeLists.txt.

40.42 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/CMakeLists.txt File Reference

Functions

- **set** (SOURCES dataset_mapper.hpp dataset_mapper_impl.hpp extension.hpp format.hpp has_serialize.hpp is_↵
_naninf.hpp load_csv.hpp load_csv.cpp load.hpp load_model_impl.hpp load_vec_impl.hpp load_impl.hpp load.↵
cpp load_arff.hpp load_arff_impl.hpp normalize_labels.hpp normalize_labels_impl.hpp save.hpp save_impl.↵
hpp serialization_template_version.hpp split_data.hpp imputer.hpp binarize.hpp confusion_matrix.hpp one_hot_↵
_encoding.hpp one_hot_encoding_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() **add_**↵
subdirectory(imputation_methods) **add_subdirectory**(map_policies) set(MLPACK_SRCS \$

40.42.1 Function Documentation

40.42.1.1 set() [1/2]

```
set (
    SOURCES dataset_mapper.hpp dataset_mapper_impl.hpp extension.hpp format.hpp has_↵
    serialize.hpp is_naninf.hpp load_csv.hpp load_csv.cpp load.hpp load_model_impl.hpp load_vec_↵
    impl.hpp load_impl.hpp load.cpp load_arff.hpp load_arff_impl.hpp normalize_labels.hpp normalize_↵
    labels_impl.hpp save.hpp save_impl.hpp serialization_template_version.hpp split_data.hpp imputer.↵
    hpp binarize.hpp confusion_matrix.hpp one_hot_encoding.hpp one_hot_encoding_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.42.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 35 of file CMakeLists.txt.

References `add_subdirectory()`.

40.43 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/C↵ MakeLists.txt File Reference

Functions

- **set** (SOURCES custom_imputation.hpp listwise_deletion.hpp mean_imputation.hpp median_imputation.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.43.1 Function Documentation**40.43.1.1 set()** [1/2]

```
set (
    SOURCES custom_imputation.hpp listwise_deletion.hpp mean_imputation.hpp median_↵
    imputation.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.44 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/map_policies/CMakeLists.txt File Reference

40.43.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 13 of file CMakeLists.txt.

40.44 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/map_policies/CMakeLists.txt File Reference

Functions

- **set** (SOURCES increment_policy.hpp missing_policy.hpp datatype.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.44.1 Function Documentation

40.44.1.1 set() [1/2]

```
set (
    SOURCES increment_policy.hpp missing_policy.hpp datatype. hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.44.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 12 of file CMakeLists.txt.

40.45 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/CMakeLists.txt File Reference

Functions

- **set** (SOURCES discrete_distribution.hpp discrete_distribution.cpp gaussian_distribution.hpp gaussian_distribution.cpp laplace_distribution.hpp laplace_distribution.cpp regression_distribution.hpp regression_distribution.cpp gamma_distribution.hpp gamma_distribution.cpp diagonal_gaussian_distribution.hpp diagonal_gaussian_distribution.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.45.1 Function Documentation

40.45.1.1 `set()` [1/2]

```
set (
    SOURCES discrete_distribution.hpp discrete_distribution.cpp gaussian_distribution.↵
    hpp gaussian_distribution.cpp laplace_distribution.hpp laplace_distribution.cpp regression_↵
    distribution.hpp regression_distribution.cpp gamma_distribution.hpp gamma_distribution.cpp diagonal_↵
    _gaussian_distribution.hpp diagonal_gaussian_distribution.  cpp )
```

Definition at line 3 of file CMakeLists.txt.

40.45.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 21 of file CMakeLists.txt.

40.46 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/CMakeLists.txt File Reference

Functions

- **set** (SOURCES cv_function.hpp cv_function_impl.hpp deduce_hp_types.hpp fixed.hpp hpt.hpp hpt_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.46.1 Function Documentation

40.46.1.1 `set()` [1/2]

```
set (
    SOURCES cv_function.hpp cv_function_impl.hpp deduce_hp_types.hpp fixed.hpp hpt.hpp
    hpt_impl.  hpp )
```

Definition at line 1 of file CMakeLists.txt.

40.46.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 12 of file CMakeLists.txt.

40.47 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/CMakeLists.txt File Reference

Functions

- **set** (SOURCES cauchy_kernel.hpp cosine_distance.hpp cosine_distance_impl.hpp epanechnikov_kernel.↵
hpp epanechnikov_kernel_impl.hpp epanechnikov_kernel.cpp example_kernel.hpp gaussian_kernel.↵
hyperbolic_tangent_kernel.hpp kernel_traits.hpp laplacian_kernel.hpp linear_kernel.hpp polynomial_kernel.↵
hpp pspectrum_string_kernel.hpp pspectrum_string_kernel_impl.hpp pspectrum_string_kernel.cpp spherical_↵
kernel.hpp triangular_kernel.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS
\$

40.47.1 Function Documentation

40.47.1.1 `set()` [1/2]

```
set (
    SOURCES cauchy_kernel.hpp cosine_distance.hpp cosine_distance_impl.hpp epanechnikov_↵  
_kernel.hpp epanechnikov_kernel_impl.hpp epanechnikov_kernel.cpp example_kernel.hpp gaussian_↵  
kernel.hpp hyperbolic_tangent_kernel.hpp kernel_traits.hpp laplacian_kernel.hpp linear_kernel.↵  
hpp polynomial_kernel.hpp pspectrum_string_kernel.hpp pspectrum_string_kernel_impl.hpp pspectrum_↵  
_string_kernel.cpp spherical_kernel.hpp triangular_kernel.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.47.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 27 of file CMakeLists.txt.

40.48 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/CMakeLists.txt File Reference

Functions

- **set** (SOURCES clamp.hpp columns_to_blocks.hpp columns_to_blocks.cpp lin_alg.hpp lin_alg_impl.hpp lin_alg_impl.hpp log_add.hpp log_add_impl.hpp make_alias.hpp random.hpp random.cpp random_basis.hpp random_basis.cpp range.hpp range_impl.hpp round.hpp shuffle_data.hpp ccov.hpp ccov_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.48.1 Function Documentation

40.48.1.1 set() [1/2]

```
set (
    SOURCES clamp.hpp columns_to_blocks.hpp columns_to_blocks.cpp lin_alg.hpp lin_alg_
impl.hpp lin_alg.cpp log_add.hpp log_add_impl.hpp make_alias.hpp random.hpp random_
basis.hpp random_basis.cpp range.hpp range_impl.hpp round.hpp shuffle_data.hpp ccov.hpp ccov_impl.
hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.48.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 28 of file CMakeLists.txt.

40.49 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/metrics/CMakeLists.txt File Reference

Functions

- **set** (SOURCES ip_metric.hpp ip_metric_impl.hpp lmetric.hpp lmetric_impl.hpp mahalanobis_distance.hpp mahalanobis_distance_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.49.1 Function Documentation

40.49.1.1 `set()` [1/2]

```
set (
    SOURCES ip_metric.hpp ip_metric_impl.hpp lmetric.hpp lmetric_impl.hpp mahalanobis_↵
distance.hpp mahalanobis_distance_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.49.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file}  )
```

Definition at line 15 of file CMakeLists.txt.

40.50 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/CMakeLists.txt File Reference

Functions

- **set** (SOURCES address.hpp ballbound.hpp ballbound_impl.hpp binary_space_tree.hpp binary_space_↵
_tree/binary_space_tree.hpp binary_space_tree/binary_space_tree_impl.hpp binary_space_tree/breadth_↵
first_dual_tree_traverser.hpp binary_space_tree/breadth_first_dual_tree_traverser_impl.hpp binary_space_↵
_tree/dual_tree_traverser.hpp binary_space_tree/dual_tree_traverser_impl.hpp binary_space_tree/mean_↵
_split.hpp binary_space_tree/mean_split_impl.hpp binary_space_tree/midpoint_split.hpp binary_space_↵
tree/midpoint_split_impl.hpp binary_space_tree/rp_tree_max_split.hpp binary_space_tree/rp_tree_max_↵
split_impl.hpp binary_space_tree/rp_tree_mean_split.hpp binary_space_tree/rp_tree_mean_split_impl.hpp
binary_space_tree/single_tree_traverser.hpp binary_space_tree/single_tree_traverser_impl.hpp binary_space_↵
_tree/vantage_point_split.hpp binary_space_tree/vantage_point_split_impl.hpp binary_space_tree/traits.hpp
binary_space_tree/typedef.hpp binary_space_tree/ub_tree_split.hpp binary_space_tree/ub_tree_split_impl.↵
hpp bounds.hpp bound_traits.hpp cellbound.hpp cellbound_impl.hpp cosine_tree/cosine_tree.hpp cosine_↵
tree/cosine_tree.cpp cover_tree.hpp cover_tree/cover_tree.hpp cover_tree/cover_tree_impl.hpp cover_tree/first_↵
_point_is_root.hpp cover_tree/single_tree_traverser.hpp cover_tree/single_tree_traverser_impl.hpp cover_↵
tree/dual_tree_traverser.hpp cover_tree/dual_tree_traverser_impl.hpp cover_tree/traits.hpp cover_tree/typedef.↵
hpp example_tree.hpp greedy_single_tree_traverser.hpp greedy_single_tree_traverser_impl.hpp hollow_ball_↵
_bound.hpp hollow_ball_bound_impl.hpp hrectbound.hpp hrectbound_impl.hpp octree.hpp octree/octree.hpp
octree/octree_impl.hpp octree/single_tree_traverser.hpp octree/single_tree_traverser_impl.hpp octree/dual_↵
_tree_traverser.hpp octree/dual_tree_traverser_impl.hpp octree/traits.hpp perform_split.hpp rectangle_↵
tree.hpp rectangle_tree/rectangle_tree.hpp rectangle_tree/rectangle_tree_impl.hpp rectangle_tree/single_↵
_tree_traverser.hpp rectangle_tree/single_tree_traverser_impl.hpp rectangle_tree/dual_tree_traverser.hpp

```

rectangle_tree/dual_tree_traverser_impl.hpp rectangle_tree/r_tree_split.hpp rectangle_tree/r_tree_split_impl.↵
hpp rectangle_tree/no_auxiliary_information.hpp rectangle_tree/r_tree_descent_heuristic.hpp rectangle_tree/r_↵
_tree_descent_heuristic_impl.hpp rectangle_tree/r_star_tree_descent_heuristic.hpp rectangle_tree/r_star_↵
tree_descent_heuristic_impl.hpp rectangle_tree/r_star_tree_split.hpp rectangle_tree/r_star_tree_split_impl.↵
hpp rectangle_tree/traits.hpp rectangle_tree/typedef.hpp rectangle_tree/x_tree_split.hpp rectangle_tree/x_↵
tree_split_impl.hpp rectangle_tree/x_tree_auxiliary_information.hpp rectangle_tree/hilbert_r_tree_descent_↵
heuristic.hpp rectangle_tree/hilbert_r_tree_descent_heuristic_impl.hpp rectangle_tree/hilbert_r_tree_split.hpp
rectangle_tree/hilbert_r_tree_split_impl.hpp rectangle_tree/hilbert_r_tree_auxiliary_information.hpp rectangle_↵
_tree/hilbert_r_tree_auxiliary_information_impl.hpp rectangle_tree/discrete_hilbert_value.hpp rectangle_↵
tree/discrete_hilbert_value_impl.hpp rectangle_tree/r_plus_tree_descent_heuristic.hpp rectangle_tree/r_↵
plus_tree_descent_heuristic_impl.hpp rectangle_tree/minimal_coverage_sweep.hpp rectangle_tree/minimal_↵
_coverage_sweep_impl.hpp rectangle_tree/minimal_splits_number_sweep.hpp rectangle_tree/minimal_↵
splits_number_sweep_impl.hpp rectangle_tree/r_plus_tree_split.hpp rectangle_tree/r_plus_tree_split_impl.hpp
rectangle_tree/r_plus_tree_split_policy.hpp rectangle_tree/r_plus_plus_tree_descent_heuristic.hpp rectangle_↵
tree/r_plus_plus_tree_descent_heuristic_impl.hpp rectangle_tree/r_plus_plus_tree_split_policy.hpp rectangle_↵
tree/r_plus_plus_tree_auxiliary_information.hpp rectangle_tree/r_plus_plus_tree_auxiliary_information_impl.hpp
space_split/hyperplane.hpp space_split/mean_space_split.hpp space_split/mean_space_split_impl.hpp space_↵
_split/midpoint_space_split.hpp space_split/midpoint_space_split_impl.hpp space_split/projection_vector.hpp
space_split/space_split.hpp space_split/space_split_impl.hpp spill_tree.hpp spill_tree/is_spill_tree.hpp spill_↵
tree/spill_tree.hpp spill_tree/spill_tree_impl.hpp spill_tree/spill_dual_tree_traverser.hpp spill_tree/spill_dual_↵
tree_traverser_impl.hpp spill_tree/spill_single_tree_traverser.hpp spill_tree/spill_single_tree_traverser_impl.hpp
spill_tree/traits.hpp spill_tree/typedef.hpp statistic.hpp traversal_info.hpp tree_traits.hpp enumerate_tree.hpp)
set(DIR_SRCS) foreach(file $

```

- **set** (DIR_SRCS \${DIR_SRCS}) \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.50.1 Function Documentation

40.50.1.1 set() [1/2]

```

set (
    SOURCES address.hpp ballbound.hpp ballbound_impl.hpp binary_space_tree.hpp binary_↵
_space_tree/binary_space_tree.hpp binary_space_tree/binary_space_tree_impl.hpp binary_space_↵
tree/breadth_first_dual_tree_traverser.hpp binary_space_tree/breadth_first_dual_tree_traverser_↵
_impl.hpp binary_space_tree/dual_tree_traverser.hpp binary_space_tree/dual_tree_traverser_impl.hpp
binary_space_tree/mean_split.hpp binary_space_tree/mean_split_impl.hpp binary_space_tree/midpoint_↵
_split.hpp binary_space_tree/midpoint_split_impl.hpp binary_space_tree/rp_tree_max_split.hpp binary_↵
_space_tree/rp_tree_max_split_impl.hpp binary_space_tree/rp_tree_mean_split.hpp binary_space_↵
_tree/rp_tree_mean_split_impl.hpp binary_space_tree/single_tree_traverser.hpp binary_space_↵
tree/single_tree_traverser_impl.hpp binary_space_tree/vantage_point_split.hpp binary_space_↵
tree/vantage_point_split_impl.hpp binary_space_tree/traits.hpp binary_space_tree/typedef.hpp binary_↵
_space_tree/ub_tree_split.hpp binary_space_tree/ub_tree_split_impl.hpp bounds.hpp bound_traits.hpp
cellbound.hpp cellbound_impl.hpp cosine_tree/cosine_tree.hpp cosine_tree/cosine_tree.cpp cover_↵
_tree.hpp cover_tree/cover_tree.hpp cover_tree/cover_tree_impl.hpp cover_tree/first_point_is_↵
root.hpp cover_tree/single_tree_traverser.hpp cover_tree/single_tree_traverser_impl.hpp cover_↵
tree/dual_tree_traverser.hpp cover_tree/dual_tree_traverser_impl.hpp cover_tree/traits.hpp cover_↵
_tree/typedef.hpp example_tree.hpp greedy_single_tree_traverser.hpp greedy_single_tree_traverser_↵
_impl.hpp hollow_ball_bound.hpp hollow_ball_bound_impl.hpp hrectbound.hpp hrectbound_impl.hpp

```

```

octree.hpp octree/octree.hpp octree/octree_impl.hpp octree/single_tree_traverser.hpp octree/single_
_tree_traverser_impl.hpp octree/dual_tree_traverser.hpp octree/dual_tree_traverser_impl.hpp octree/traits.
hpp perform_split.hpp rectangle_tree.hpp rectangle_tree/rectangle_tree.hpp rectangle_tree/rectangle_
_tree_impl.hpp rectangle_tree/single_tree_traverser.hpp rectangle_tree/single_tree_traverser_
impl.hpp rectangle_tree/dual_tree_traverser.hpp rectangle_tree/dual_tree_traverser_impl.hpp rectangle_
_tree/r_tree_split.hpp rectangle_tree/r_tree_split_impl.hpp rectangle_tree/no_auxiliary_information.
hpp rectangle_tree/r_tree_descent_heuristic.hpp rectangle_tree/r_tree_descent_heuristic_impl.hpp
rectangle_tree/r_star_tree_descent_heuristic.hpp rectangle_tree/r_star_tree_descent_heuristic_
impl.hpp rectangle_tree/r_star_tree_split.hpp rectangle_tree/r_star_tree_split_impl.hpp rectangle_
_tree/traits.hpp rectangle_tree/typedef.hpp rectangle_tree/x_tree_split.hpp rectangle_tree/x_
tree_split_impl.hpp rectangle_tree/x_tree_auxiliary_information.hpp rectangle_tree/hilbert_r_
tree_descent_heuristic.hpp rectangle_tree/hilbert_r_tree_descent_heuristic_impl.hpp rectangle_
_tree/hilbert_r_tree_split.hpp rectangle_tree/hilbert_r_tree_split_impl.hpp rectangle_tree/hilbert_
_r_tree_auxiliary_information.hpp rectangle_tree/hilbert_r_tree_auxiliary_information_impl.hpp
rectangle_tree/discrete_hilbert_value.hpp rectangle_tree/discrete_hilbert_value_impl.hpp rectangle_
_tree/r_plus_tree_descent_heuristic.hpp rectangle_tree/r_plus_tree_descent_heuristic_impl.hpp
rectangle_tree/minimal_coverage_sweep.hpp rectangle_tree/minimal_coverage_sweep_impl.hpp rectangle_
_tree/minimal_splits_number_sweep.hpp rectangle_tree/minimal_splits_number_sweep_impl.hpp rectangle_
_tree/r_plus_tree_split.hpp rectangle_tree/r_plus_tree_split_impl.hpp rectangle_tree/r_plus_tree_
_split_policy.hpp rectangle_tree/r_plus_plus_tree_descent_heuristic.hpp rectangle_tree/r_plus_
plus_tree_descent_heuristic_impl.hpp rectangle_tree/r_plus_plus_tree_split_policy.hpp rectangle_
_tree/r_plus_plus_tree_auxiliary_information.hpp rectangle_tree/r_plus_plus_tree_auxiliary_
information_impl.hpp space_split/hyperplane.hpp space_split/mean_space_split.hpp space_split/mean_
_space_split_impl.hpp space_split/midpoint_space_split.hpp space_split/midpoint_space_split_
impl.hpp space_split/projection_vector.hpp space_split/space_split.hpp space_split/space_split_
impl.hpp spill_tree.hpp spill_tree/is_spill_tree.hpp spill_tree/spill_tree.hpp spill_tree/spill_
_tree_impl.hpp spill_tree/spill_dual_tree_traverser.hpp spill_tree/spill_dual_tree_traverser_
impl.hpp spill_tree/spill_single_tree_traverser.hpp spill_tree/spill_single_tree_traverser_impl.
hpp spill_tree/traits.hpp spill_tree/typedef.hpp statistic.hpp traversal_info.hpp tree_traits.hpp
enumerate_tree.  hpp )

```

Definition at line 3 of file CMakeLists.txt.

40.50.1.2 set() [2/2]

```

set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )

```

Definition at line 132 of file CMakeLists.txt.

40.51 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/CMakeLists.txt File Reference

Functions

- **set** (SOURCES arma_traits.hpp arma_config.hpp arma_config_check.hpp backtrace.hpp backtrace.cpp cli.hpp cli.cpp cli_impl.hpp deprecated.hpp hyphenate_string.hpp is_std_vector.hpp log.hpp log.cpp mlpack_main.hpp nulloutstream.hpp param.hpp param_checks.hpp param_checks_impl.hpp param_data.hpp prefixedoutstream. hpp prefixedoutstream.cpp prefixedoutstream_impl.hpp program_doc.hpp program_doc.cpp sfinae_utility.hpp singletons.cpp timers.hpp timers.cpp version.hpp version.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.51.1 Function Documentation

40.51.1.1 `set()` [1/2]

```
set (
    SOURCES arma_traits.hpp arma_config.hpp arma_config_check.hpp backtrace.hpp backtrace.
cpp cli.hpp cli.cpp cli_impl.hpp deprecated.hpp hyphenate_string.hpp is_std_vector.hpp log.hpp
log.cpp mlpack_main.hpp nullostream.hpp param.hpp param_checks.hpp param_checks_impl.hpp param_
_data.hpp prefixedostream.hpp prefixedostream.cpp prefixedostream_impl.hpp program_doc.hpp
program_doc.cpp sfinae_utility.hpp singletons.cpp timers.hpp timers.cpp version.hpp version.  cpp
)
```

Definition at line 3 of file CMakeLists.txt.

40.51.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 39 of file CMakeLists.txt.

40.52 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/adaboost/CMakeLists.txt

File Reference

Functions

- **set** (SOURCES adaboost.hpp adaboost_impl.hpp adaboost_model.hpp adaboost_model.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.52.1 Function Documentation

40.52.1.1 `set()` [1/2]

```
set (
    SOURCES adaboost.hpp adaboost_impl.hpp adaboost_model.hpp adaboost_model.  cpp )
```

Definition at line 3 of file CMakeLists.txt.

40.52.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 13 of file CMakeLists.txt.

References `add_cli_executable()`.

40.53 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/CMakeLists.txt File Reference

Functions

- **set** (SOURCES amf.hpp amf_impl.hpp) **add_subdirectory**(update_rules) **add_subdirectory**(init_rules) **add_subdirectory**(termination_policies) `set(DIR_SRCS)` `foreach(file $`
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) `endforeach()` `set(MLPACK_SRCS $`

40.53.1 Function Documentation

40.53.1.1 `set()` [1/2]

```
set (
    SOURCES amf.hpp amf_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.53.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 15 of file CMakeLists.txt.

40.54 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/init_rules/CMakeLists.txt File Reference

Functions

- **set** (SOURCES random_init.hpp random_acol_init.hpp average_init.hpp given_init.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.54.1 Function Documentation

40.54.1.1 set() [1/2]

```
set (
    SOURCES random_init.hpp random_acol_init.hpp average_init.hpp given_init. .hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.54.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 13 of file CMakeLists.txt.

40.55 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/CMakeLists.txt File Reference

Functions

- **set** (SOURCES simple_residue_termination.hpp simple_tolerance_termination.hpp validation_rmse_termination.hpp incomplete_incremental_termination.hpp complete_incremental_termination.hpp max_iteration_termination.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.55.1 Function Documentation

40.55.1.1 `set()` [1/2]

```
set (
    SOURCES simple_residue_termination.hpp simple_tolerance_termination.hpp validation_
_rmse_termination.hpp incomplete_incremental_termination.hpp complete_incremental_termination.hpp
max_iteration_termination.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.55.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 15 of file CMakeLists.txt.

40.56 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/CMakeLists.txt File Reference

Functions

- **set** (SOURCES nmf_als.hpp nmf_mult_dist.hpp nmf_mult_div.hpp svd_batch_learning.hpp svd_incomplete_incomplete_learning.hpp svd_complete_incremental_learning.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.56.1 Function Documentation

40.56.1.1 `set()` [1/2]

```
set (
    SOURCES nmf_als.hpp nmf_mult_dist.hpp nmf_mult_div.hpp svd_batch_learning.hpp svd_incomplete_incomplete_learning.hpp svd_complete_incremental_learning.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.56.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 15 of file CMakeLists.txt.

40.57 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/↵ CMakeLists.txt File Reference

Functions

- **set** (SOURCES identity_function.hpp logistic_function.hpp softsign_function.hpp tanh_function.hpp rectifier_↵
function.hpp softplus_function.hpp swish_function.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS
\$

40.57.1 Function Documentation

40.57.1.1 set() [1/2]

```
set (
    SOURCES identity_function.hpp logistic_function.hpp softsign_function.hpp tanh_↵  
function.hpp rectifier_function.hpp softplus_function.hpp swish_function.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.57.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 16 of file CMakeLists.txt.

40.58 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented/CMake↵ Lists.txt File Reference

Functions

- **set** (SOURCES) set(MLPACK_SRCS \$

40.58.1 Function Documentation

40.58.1.1 set()

```
set (
    SOURCES )
```

Definition at line 3 of file CMakeLists.txt.

References `add_subdirectory()`.

40.59 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented/tasks/CMakeLists.txt File Reference

Functions

- **set** (SOURCES add.hpp add_impl.hpp copy.hpp copy_impl.hpp score.hpp score_impl.hpp sort.hpp sort_impl.↵hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.59.1 Function Documentation

40.59.1.1 set() [1/2]

```
set (
    SOURCES add.hpp add_impl.hpp copy.hpp copy_impl.hpp score.hpp score_impl.hpp sort.hpp
    sort_impl. hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.59.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 17 of file CMakeLists.txt.

40.60 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/CMakeLists.txt File Reference

Functions

- **set** (SOURCES ffn.hpp ffn_impl.hpp rnn.hpp rnn_impl.hpp brnn.hpp brnn_impl.hpp) **add_subdirectory**(visitor) **add_subdirectory**(activation_functions) **add_subdirectory**(init_rules) **add_subdirectory**(layer) **add_subdirectory**(loss_functions) **add_subdirectory**(convolution_rules) **add_subdirectory**(rbm) **add_subdirectory**(augmented) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.60.1 Function Documentation

40.60.1.1 set() [1/2]

```
set (
    SOURCES ffn.hpp ffn_impl.hpp rnn.hpp rnn_impl.hpp brnn.hpp brnn_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.60.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 24 of file CMakeLists.txt.

40.61 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/CMakeLists.txt File Reference

Functions

- **set** (SOURCES border_modes.hpp naive_convolution.hpp fft_convolution.hpp svd_convolution.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.61.1 Function Documentation

40.61.1.1 `set()` [1/2]

```
set (
    SOURCES border_modes.hpp naive_convolution.hpp fft_convolution.hpp svd_convolution.
    hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.61.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 13 of file CMakeLists.txt.

40.62 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/dists/CMakeLists.txt File Reference

Functions

- `set` (SOURCES bernoulli_distribution.hpp bernoulli_distribution_impl.hpp) `set`(DIR_SRCS) `foreach`(file \$
- `set` (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) `endforeach`() `set`(MLPACK_SRCS \$

40.62.1 Function Documentation

40.62.1.1 `set()` [1/2]

```
set (
    SOURCES bernoulli_distribution.hpp bernoulli_distribution_impl. hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.62.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 11 of file CMakeLists.txt.

40.63 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/CMakeLists.txt File Reference

Functions

- **set** (SOURCES const_init.hpp gaussian_init.hpp he_init.hpp init_rules_traits.hpp kathirvalavakumar_subavathi_init.hpp lecun_normal_init.hpp network_init.hpp nguyen_widrow_init.hpp oivs_init.hpp orthogonal_init.hpp random_init.hpp glorot_init.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.63.1 Function Documentation

40.63.1.1 set() [1/2]

```
set (
    SOURCES const_init.hpp gaussian_init.hpp he_init.hpp init_rules_traits.hpp kathirvalavakumar_subavathi_init.hpp lecun_normal_init.hpp network_init.hpp nguyen_widrow_init.hpp oivs_init.hpp orthogonal_init.hpp random_init.hpp glorot_init.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.63.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 21 of file CMakeLists.txt.

40.64 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/CMakeLists.txt File Reference

Functions

- **set** (SOURCES add.hpp add_impl.hpp add_merge.hpp add_merge_impl.hpp alpha_dropout.hpp alpha_dropout_impl.hpp atrous_convolution.hpp atrous_convolution_impl.hpp base_layer.hpp batch_norm.hpp batch_norm_impl.hpp bilinear_interpolation.hpp bilinear_interpolation_impl.hpp concat.hpp concat_impl.hpp concat_performance.hpp concat_performance_impl.hpp concatenate.hpp concatenate_impl.hpp constant.hpp constant_impl.hpp convolution.hpp convolution_impl.hpp dropconnect.hpp dropconnect_impl.hpp dropout.hpp dropout_impl.hpp elu.hpp elu_impl.hpp fast_lstm.hpp fast_lstm_impl.hpp flexible_relu.hpp flexible_relu_impl.hpp glimpse.hpp glimpse_impl.hpp gru.hpp gru_impl.hpp hard_tanh.hpp hard_tanh_impl.hpp join.hpp join_impl.hpp layer.hpp layer_norm.hpp layer_norm_impl.hpp layer_traits.hpp layer_types.hpp leaky_relu.hpp leaky_relu_impl.hpp linear.hpp linear_impl.hpp linear_no_bias.hpp linear_no_bias_impl.hpp log_softmax.hpp log_softmax_impl.hpp lookup.hpp lookup_impl.hpp lstm.hpp lstm_impl.hpp max_pooling.hpp max_pooling_impl.hpp mean_pooling.hpp mean_pooling_impl.hpp multiply_constant.hpp multiply_constant_impl.hpp multiply_merge.hpp multiply_merge_impl.hpp parametric_relu.hpp parametric_relu_impl.hpp recurrent.hpp recurrent_impl.hpp recurrent_attention.hpp recurrent_attention_impl.hpp reinforce_normal.hpp reinforce_normal_impl.hpp reparametrization.hpp reparametrization_impl.hpp select.hpp select_impl.hpp sequential.hpp sequential_impl.hpp subview.hpp transposed_convolution.hpp transposed_convolution_impl.hpp vr_class_reward.hpp vr_class_reward_impl.hpp c_relu.hpp c_relu_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.64.1 Function Documentation

40.64.1.1 set() [1/2]

```
set (
    SOURCES add.hpp add_impl.hpp add_merge.hpp add_merge_impl.hpp alpha_dropout.hpp alpha_dropout_impl.hpp atrous_convolution.hpp atrous_convolution_impl.hpp base_layer.hpp batch_norm.hpp batch_norm_impl.hpp bilinear_interpolation.hpp bilinear_interpolation_impl.hpp concat.hpp concat_impl.hpp concat_performance.hpp concat_performance_impl.hpp concatenate.hpp concatenate_impl.hpp constant.hpp constant_impl.hpp convolution.hpp convolution_impl.hpp dropconnect.hpp dropconnect_impl.hpp dropout.hpp dropout_impl.hpp elu.hpp elu_impl.hpp fast_lstm.hpp fast_lstm_impl.hpp flexible_relu.hpp flexible_relu_impl.hpp glimpse.hpp glimpse_impl.hpp gru.hpp gru_impl.hpp hard_tanh.hpp hard_tanh_impl.hpp join.hpp join_impl.hpp layer.hpp layer_norm.hpp layer_norm_impl.hpp layer_traits.hpp layer_types.hpp leaky_relu.hpp leaky_relu_impl.hpp linear.hpp linear_impl.hpp linear_no_bias.hpp linear_no_bias_impl.hpp log_softmax.hpp log_softmax_impl.hpp lookup.hpp lookup_impl.hpp lstm.hpp lstm_impl.hpp max_pooling.hpp max_pooling_impl.hpp mean_pooling.hpp mean_pooling_impl.hpp multiply_constant.hpp multiply_constant_impl.hpp multiply_merge.hpp multiply_merge_impl.hpp parametric_relu.hpp parametric_relu_impl.hpp recurrent.hpp recurrent_impl.hpp recurrent_attention.hpp recurrent_attention_impl.hpp reinforce_normal.hpp reinforce_normal_impl.hpp reparametrization.hpp reparametrization_impl.hpp select.hpp select_impl.hpp sequential.hpp sequential_impl.hpp subview.hpp transposed_convolution.hpp transposed_convolution_impl.hpp vr_class_reward.hpp vr_class_reward_impl.hpp c_relu.hpp c_relu_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.64.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 96 of file CMakeLists.txt.

40.65 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss_functions/CMakeLists.txt File Reference

Functions

- **set** (SOURCES cross_entropy_error.hpp cross_entropy_error_impl.hpp dice_loss.hpp dice_loss_impl.hpp earth_mover_distance.hpp earth_mover_distance_impl.hpp kl_divergence.hpp kl_divergence_impl.hpp mean_squared_error.hpp mean_squared_error_impl.hpp negative_log_likelihood.hpp negative_log_likelihood_impl.hpp reconstruction_loss.hpp reconstruction_loss_impl.hpp sigmoid_cross_entropy_error.hpp sigmoid_cross_entropy_error_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.65.1 Function Documentation

40.65.1.1 set() [1/2]

```
set (
    SOURCES cross_entropy_error.hpp cross_entropy_error_impl.hpp dice_loss.hpp dice_loss_impl.hpp earth_mover_distance.hpp earth_mover_distance_impl.hpp kl_divergence.hpp kl_divergence_impl.hpp mean_squared_error.hpp mean_squared_error_impl.hpp negative_log_likelihood.hpp negative_log_likelihood_impl.hpp reconstruction_loss.hpp reconstruction_loss_impl.hpp sigmoid_cross_entropy_error.hpp sigmoid_cross_entropy_error_impl. hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.65.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 25 of file CMakeLists.txt.

40.66 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/rbm/CMakeLists.txt File Reference

Functions

- **set** (SOURCES rbm.hpp rbm_impl.hpp rbm_policies.hpp spike_slab_rbm_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.66.1 Function Documentation

40.66.1.1 set() [1/2]

```
set (
    SOURCES rbm.hpp rbm_impl.hpp rbm_policies.hpp spike_slab_rbm_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.66.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file}  )
```

Definition at line 13 of file CMakeLists.txt.

40.67 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/CMakeLists.txt File Reference

Functions

- **set** (SOURCES add_visitor.hpp add_visitor_impl.hpp backward_visitor.hpp backward_visitor_impl.hpp copy_↵
visitor.hpp copy_visitor_impl.hpp delete_visitor.hpp delete_visitor_impl.hpp delta_visitor.hpp delta_visitor_impl.↵
hpp deterministic_set_visitor.hpp deterministic_set_visitor_impl.hpp forward_visitor.hpp forward_visitor_impl.hpp
gradient_set_visitor.hpp gradient_set_visitor_impl.hpp gradient_update_visitor.hpp gradient_update_visitor_↵
impl.hpp gradient_visitor.hpp gradient_visitor_impl.hpp gradient_zero_visitor.hpp gradient_zero_visitor_impl.↵
hpp load_output_parameter_visitor.hpp load_output_parameter_visitor_impl.hpp loss_visitor.hpp loss_visitor↵
_impl.hpp output_height_visitor.hpp output_height_visitor_impl.hpp output_parameter_visitor.hpp output_↵
parameter_visitor_impl.hpp output_width_visitor.hpp output_width_visitor_impl.hpp parameters_set_visitor.↵
hpp parameters_set_visitor_impl.hpp parameters_visitor.hpp parameters_visitor_impl.hpp reset_cell_visitor.↵
hpp reset_cell_visitor_impl.hpp reset_visitor.hpp reset_visitor_impl.hpp reward_set_visitor.hpp reward_set_↵
visitor_impl.hpp run_set_visitor.hpp run_set_visitor_impl.hpp save_output_parameter_visitor.hpp save_output↵
_parameter_visitor_impl.hpp set_input_height_visitor.hpp set_input_height_visitor_impl.hpp set_input_width_↵
visitor.hpp set_input_width_visitor_impl.hpp weight_set_visitor.hpp weight_set_visitor_impl.hpp weight_size_↵
visitor.hpp weight_size_visitor_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.67.1 Function Documentation

40.67.1.1 `set()` [1/2]

```
set (
    SOURCES add_visitor.hpp add_visitor_impl.hpp backward_visitor.hpp backward_visitor_↵
impl.hpp copy_visitor.hpp copy_visitor_impl.hpp delete_visitor.hpp delete_visitor_impl.hpp delta_↵
_visitor.hpp delta_visitor_impl.hpp deterministic_set_visitor.hpp deterministic_set_visitor_↵
impl.hpp forward_visitor.hpp forward_visitor_impl.hpp gradient_set_visitor.hpp gradient_set_↵
visitor_impl.hpp gradient_update_visitor.hpp gradient_update_visitor_impl.hpp gradient_visitor.hpp
gradient_visitor_impl.hpp gradient_zero_visitor.hpp gradient_zero_visitor_impl.hpp load_output_↵
parameter_visitor.hpp load_output_parameter_visitor_impl.hpp loss_visitor.hpp loss_visitor_impl.↵
hpp output_height_visitor.hpp output_height_visitor_impl.hpp output_parameter_visitor.hpp output_↵
_parameter_visitor_impl.hpp output_width_visitor.hpp output_width_visitor_impl.hpp parameters_↵
_set_visitor.hpp parameters_set_visitor_impl.hpp parameters_visitor.hpp parameters_visitor_↵
impl.hpp reset_cell_visitor.hpp reset_cell_visitor_impl.hpp reset_visitor.hpp reset_visitor_↵
impl.hpp reward_set_visitor.hpp reward_set_visitor_impl.hpp run_set_visitor.hpp run_set_visitor_↵
_impl.hpp save_output_parameter_visitor.hpp save_output_parameter_visitor_impl.hpp set_input_↵
_height_visitor.hpp set_input_height_visitor_impl.hpp set_input_width_visitor.hpp set_input_↵
width_visitor_impl.hpp weight_set_visitor.hpp weight_set_visitor_impl.hpp weight_size_visitor.hpp
weight_size_visitor_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.67.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 63 of file CMakeLists.txt.

40.68 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/approx_kfn/CMake↵ Lists.txt File Reference

Functions

- **set** (SOURCES drusilla_select.hpp drusilla_select_impl.hpp qdafn.hpp qdafn_impl.hpp) set(DIR_SRCS) fore-
ach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS
\$

40.68.1 Function Documentation

40.68.1.1 `set()` [1/2]

```
set (
    SOURCES drusilla_select.hpp drusilla_select_impl.hpp qdafn.hpp qdafn_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.68.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file}  )
```

Definition at line 15 of file CMakeLists.txt.

40.69 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/bias_svd/CMakeLists.txt File Reference

Functions

- **set** (SOURCES bias_svd.hpp bias_svd_impl.hpp bias_svd_function.hpp bias_svd_function_impl.hpp) set(DIR←_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.69.1 Function Documentation

40.69.1.1 `set()` [1/2]

```
set (
    SOURCES bias_svd.hpp bias_svd_impl.hpp bias_svd_function.hpp bias_svd_function_impl.
    hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.69.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 13 of file CMakeLists.txt.

40.70 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/block_krylov_svd/CMakeLists.txt File Reference

Functions

- **set** (SOURCES randomized_block_krylov_svd.hpp randomized_block_krylov_svd.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.70.1 Function Documentation

40.70.1.1 set() [1/2]

```
set (
    SOURCES randomized_block_krylov_svd.hpp randomized_block_krylov_svd.  cpp )
```

Definition at line 3 of file CMakeLists.txt.

40.70.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 11 of file CMakeLists.txt.

40.71 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/CMakeLists.txt File Reference

Functions

- **set** (SOURCES cf.hpp cf_impl.hpp cf_model.hpp cf_model_impl.hpp svd_wrapper.hpp svd_wrapper_ impl.hpp) **add_subdirectory**(decomposition_policies) **add_subdirectory**(interpolation_policies) **add_ subdirectory**(neighbor_search_policies) **add_subdirectory**(normalization) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.71.1 Function Documentation

40.71.1.1 `set()` [1/2]

```
set (
    SOURCES cf.hpp cf_impl.hpp cf_model.hpp cf_model_impl.hpp svd_wrapper.hpp svd_↵
    wrapper_impl. hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.71.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 20 of file CMakeLists.txt.

References `add_cli_executable()`.

40.72 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_↵ policies/CMakeLists.txt File Reference

Functions

- **set** (SOURCES batch_svd_method.hpp bias_svd_method.hpp nmf_method.hpp randomized_svd_method.hpp regularized_svd_method.hpp svd_complete_method.hpp svd_incomplete_method.hpp svdplusplus_method.↵hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.72.1 Function Documentation

40.72.1.1 `set()` [1/2]

```
set (
    SOURCES batch_svd_method.hpp bias_svd_method.hpp nmf_method.hpp randomized_svd_↵
    method.hpp regularized_svd_method.hpp svd_complete_method.hpp svd_incomplete_method.hpp svdplusplus_↵
    _method. hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.72.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 17 of file CMakeLists.txt.

40.73 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/interpolation_policies/↵ CMakeLists.txt File Reference

Functions

- **set** (SOURCES average_interpolation.hpp similarity_interpolation.hpp regression_interpolation.hpp) set(DIR_↵
SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS
\$

40.73.1 Function Documentation

40.73.1.1 set() [1/2]

```
set (
    SOURCES average_interpolation.hpp similarity_interpolation.hpp regression_interpolation.  
hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.73.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 12 of file CMakeLists.txt.

40.74 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/neighbor_search_↵ policies/CMakeLists.txt File Reference

Functions

- **set** (SOURCES lmetric_search.hpp cosine_search.hpp pearson_search.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS
\$

40.74.1 Function Documentation

40.74.1.1 `set()` [1/2]

```
set (
    SOURCES lmetric_search.hpp cosine_search.hpp pearson_search.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.74.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file}  )
```

Definition at line 12 of file CMakeLists.txt.

40.75 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/CMakeLists.txt File Reference

Functions

- `set` (SOURCES no_normalization.hpp overall_mean_normalization.hpp user_mean_normalization.hpp item_mean_normalization.hpp z_score_normalization.hpp combined_normalization.hpp) `set`(DIR_SRCS) `foreach`(file \$
- `set` (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) `endforeach`() `set`(MLPACK_SRCS \$

40.75.1 Function Documentation

40.75.1.1 `set()` [1/2]

```
set (
    SOURCES no_normalization.hpp overall_mean_normalization.hpp user_mean_normalization.↵
    hpp item_mean_normalization.hpp z_score_normalization.hpp combined_normalization.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.75.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 15 of file CMakeLists.txt.

40.76 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/CMakeLists.txt File Reference

Functions

- **add_subdirectory** (\${dir}) endforeach() **set**(MLPACK_SRCS \$
- **set** (DIRS adaboost amf ann approx_kfn bias_svd block_krylov_svd cf dbscan decision_stump decision_tree det emst fastmks gmm hmm hoeffding_trees kde kernel_pca kmeans lars linear_regression linear_svm lmn local_coordinate_coding logistic_regression lsh matrix_completion mean_shift naive_bayes nca neighbor_search nmf nystroem_method pca perceptron preprocess quic_svd radical random_forest randomized_svd range_search rann regularized_svd reinforcement_learning softmax_regression sparse_autoencoder sparse_coding svdplusplus) foreach(dir \$

40.76.1 Function Documentation

40.76.1.1 add_subdirectory()

```
add_subdirectory (
    ${dir} )
```

Definition at line 55 of file CMakeLists.txt.

40.76.1.2 set()

```
set (
    DIRS adaboost amf ann approx_kfn bias_svd block_krylov_svd cf dbscan decision_↵
stump decision_tree det emst fastmks gmm hmm hoeffding_trees kde kernel_pca kmeans lars linear_↵
_regression linear_svm lmn local_coordinate_coding logistic_regression lsh matrix_completion
mean_shift naive_bayes nca neighbor_search nmf nystroem_method pca perceptron preprocess quic_↵
svd radical random_forest randomized_svd range_search rann regularized_svd reinforcement_learning
softmax_regression sparse_autoencoder sparse_coding svdplusplus )
```

Definition at line 2 of file CMakeLists.txt.

40.77 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/dbscan/CMakeLists.txt File Reference

Functions

- **set** (SOURCES dbscan.hpp dbscan_impl.hpp random_point_selection.hpp ordered_point_selection.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.77.1 Function Documentation

40.77.1.1 **set()** [1/2]

```
set (
    SOURCES dbscan.hpp dbscan_impl.hpp random_point_selection.hpp ordered_point_selection.
   .hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.77.1.2 **set()** [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 13 of file CMakeLists.txt.

References `add_cli_executable()`.

40.78 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_stump/CMakeLists.txt File Reference

Functions

- **set** (SOURCES decision_stump.hpp decision_stump_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.78.1 Function Documentation

40.78.1.1 `set()` [1/2]

```
set (
    SOURCES decision_stump.hpp decision_stump_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.78.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file}  )
```

Definition at line 11 of file CMakeLists.txt.

References `add_cli_executable()`.

40.79 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/CMakeLists.txt File Reference

Functions

- **set** (SOURCES all_dimension_select.hpp decision_tree.hpp decision_tree_impl.hpp all_categorical_split.hpp all_categorical_split_impl.hpp best_binary_numeric_split.hpp best_binary_numeric_split_impl.hpp gini_gain.↵
hpp information_gain.hpp multiple_random_dimension_select.hpp random_dimension_select.hpp) set(DIR_S↵
RCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS
\$

40.79.1 Function Documentation

40.79.1.1 **set()** [1/2]

```
set (
    SOURCES all_dimension_select.hpp decision_tree.hpp decision_tree_impl.hpp all_↵
    categorical_split.hpp all_categorical_split_impl.hpp best_binary_numeric_split.hpp best_binary_↵
    _numeric_split_impl.hpp gini_gain.hpp information_gain.hpp multiple_random_dimension_select.hpp
    random_dimension_select.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.79.1.2 **set()** [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file}  )
```

Definition at line 20 of file CMakeLists.txt.

References `add_cli_executable()`.

40.80 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/det/CMakeLists.txt File Reference

Functions

- **set** (SOURCES dtree.hpp dtree_impl.hpp dt_utils.hpp dt_utils_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.80.1 Function Documentation

40.80.1.1 **set()** [1/2]

```
set (
    SOURCES dtree.hpp dtree_impl.hpp dt_utils.hpp dt_utils_impl.  hpp )
```

Definition at line 4 of file CMakeLists.txt.

40.80.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 17 of file CMakeLists.txt.

References `add_cli_executable()`.

40.81 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/CMakeLists.txt File Reference

Functions

- **set** (SOURCES union_find.hpp dtb.hpp dtb_impl.hpp dtb_rules.hpp dtb_rules_impl.hpp dtb_stat.hpp edge_↔ pair.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.81.1 Function Documentation

40.81.1.1 set() [1/2]

```
set (
    SOURCES union_find.hpp dtb.hpp dtb_impl.hpp dtb_rules.hpp dtb_rules_impl.hpp dtb_↔
stat.hpp edge_pair.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.81.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 18 of file CMakeLists.txt.

References `add_cli_executable()`.

40.82 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/fastmks/CMakeLists.txt File Reference

Functions

- **set** (SOURCES fastmks.hpp fastmks_impl.hpp fastmks_model.hpp fastmks_model_impl.hpp fastmks_model.cpp fastmks_rules.hpp fastmks_rules_impl.hpp fastmks_stat.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.82.1 Function Documentation

40.82.1.1 set() [1/2]

```
set (
    SOURCES fastmks.hpp fastmks_impl.hpp fastmks_model.hpp fastmks_model_impl.hpp fastmks_↵
_model.cpp fastmks_rules.hpp fastmks_rules_impl.hpp fastmks_stat.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.82.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 17 of file CMakeLists.txt.

References `add_cli_executable()`.

40.83 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/CMakeLists.txt File Reference

Functions

- **set** (SOURCES gmm.hpp gmm.cpp gmm_impl.hpp diagonal_gmm.hpp diagonal_gmm.cpp diagonal_gmm_↵
impl.hpp em_fit.hpp em_fit_impl.hpp no_constraint.hpp positive_definite_constraint.hpp diagonal_constraint.hpp
eigenvalue_ratio_constraint.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.83.1 Function Documentation

40.83.1.1 `set()` [1/2]

```
set (
    SOURCES gmm.hpp gmm.cpp gmm_impl.hpp diagonal_gmm.hpp diagonal_gmm.cpp diagonal_gmm_
_impl.hpp em_fit.hpp em_fit_impl.hpp no_constraint.hpp positive_definite_constraint.hpp diagonal_
_constraint.hpp eigenvalue_ratio_constraint.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.83.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 21 of file CMakeLists.txt.

References `add_cli_executable()`.

40.84 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hmm/CMakeLists.txt File Reference

Functions

- **`set`** (SOURCES hmm.hpp hmm_impl.hpp hmm_model.hpp hmm_regression.hpp hmm_regression_impl.hpp hmm_util.hpp hmm_util_impl.hpp) `set(DIR_SRCS) foreach(file $`
- **`set`** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) `endforeach() set(MLPACK_SRCS $`

40.84.1 Function Documentation

40.84.1.1 `set()` [1/2]

```
set (
    SOURCES hmm.hpp hmm_impl.hpp hmm_model.hpp hmm_regression.hpp hmm_regression_impl.hpp
hmm_util.hpp hmm_util_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.84.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 16 of file CMakeLists.txt.

References `add_cli_executable()`.

40.85 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/CMakeLists.txt File Reference

Functions

- **set** (SOURCES binary_numeric_split.hpp binary_numeric_split_impl.hpp binary_numeric_split_info.hpp categorical_split_info.hpp gini_impurity.hpp hoeffding_categorical_split.hpp hoeffding_categorical_split_impl.hpp hoeffding_numeric_split.hpp hoeffding_numeric_split_impl.hpp hoeffding_tree.hpp hoeffding_tree_impl.hpp hoeffding_tree_model.hpp hoeffding_tree_model.cpp information_gain.hpp numeric_split_info.hpp typedef.hpp) `set(DIR_SRCS) foreach(file $`
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) `endforeach() set(MLPACK_SRCS $`

40.85.1 Function Documentation

40.85.1.1 `set()` [1/2]

```
set (
    SOURCES binary_numeric_split.hpp binary_numeric_split_impl.hpp binary_numeric_split_info.hpp categorical_split_info.hpp gini_impurity.hpp hoeffding_categorical_split.hpp hoeffding_categorical_split_impl.hpp hoeffding_numeric_split.hpp hoeffding_numeric_split_impl.hpp hoeffding_tree.hpp hoeffding_tree_impl.hpp hoeffding_tree_model.hpp hoeffding_tree_model.cpp information_gain.hpp numeric_split_info.hpp typedef.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.85.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 25 of file CMakeLists.txt.

References `add_cli_executable()`.

40.86 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/CMakeLists.txt File Reference

Functions

- **set** (SOURCES kde.hpp kde_impl.hpp kde_rules.hpp kde_rules_impl.hpp kde_stat.hpp kde_model.hpp kde_model_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.86.1 Function Documentation

40.86.1.1 set() [1/2]

```
set (
    SOURCES kde.hpp kde_impl.hpp kde_rules.hpp kde_rules_impl.hpp kde_stat.hpp kde_model.hpp
    kde_model_impl.hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.86.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 16 of file CMakeLists.txt.

References `add_cli_executable()`.

40.87 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kernel_pca/CMakeLists.txt File Reference

Functions

- **set** (SOURCES kernel_pca.hpp kernel_pca_impl.hpp) **add_subdirectory**(kernel_rules) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.87.1 Function Documentation

40.87.1.1 `set()` [1/2]

```
set (
    SOURCES kernel_pca.hpp kernel_pca_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.87.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file}  )
```

Definition at line 13 of file CMakeLists.txt.

References `add_cli_executable()`.

40.88 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kernel_pca/kernel_↵rules/CMakeLists.txt File Reference

Functions

- **set** (SOURCES nystroem_method.hpp naive_method.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.88.1 Function Documentation

40.88.1.1 `set()` [1/2]

```
set (
    SOURCES nystroem_method.hpp naive_method.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.88.1.2 **set()** [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 11 of file CMakeLists.txt.

40.89 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/CMakeLists.txt

File Reference

Functions

- **set** (SOURCES allow_empty_clusters.hpp dual_tree_kmeans.hpp dual_tree_kmeans_impl.hpp dual_tree_kmeans_rules.hpp dual_tree_kmeans_rules_impl.hpp dual_tree_kmeans_statistic.hpp elkan_kmeans.hpp elkan_kmeans_impl.hpp hamerly_kmeans.hpp hamerly_kmeans_impl.hpp kill_empty_clusters.hpp kmeans.hpp kmeans_impl.hpp max_variance_new_cluster.hpp max_variance_new_cluster_impl.hpp naive_kmeans.hpp naive_kmeans_impl.hpp pelleg_moore_kmeans.hpp pelleg_moore_kmeans_impl.hpp pelleg_moore_kmeans_rules.hpp pelleg_moore_kmeans_rules_impl.hpp pelleg_moore_kmeans_statistic.hpp random_partition.hpp refined_start.hpp refined_start_impl.hpp sample_initialization.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.89.1 Function Documentation

40.89.1.1 **set()** [1/2]

```
set (
    SOURCES allow_empty_clusters.hpp dual_tree_kmeans.hpp dual_tree_kmeans_impl.hpp dual_tree_kmeans_rules.hpp dual_tree_kmeans_rules_impl.hpp dual_tree_kmeans_statistic.hpp elkan_kmeans.hpp elkan_kmeans_impl.hpp hamerly_kmeans.hpp hamerly_kmeans_impl.hpp kill_empty_clusters.hpp kmeans.hpp kmeans_impl.hpp max_variance_new_cluster.hpp max_variance_new_cluster_impl.hpp naive_kmeans.hpp naive_kmeans_impl.hpp pelleg_moore_kmeans.hpp pelleg_moore_kmeans_impl.hpp pelleg_moore_kmeans_rules.hpp pelleg_moore_kmeans_rules_impl.hpp pelleg_moore_kmeans_statistic.hpp random_partition.hpp refined_start.hpp refined_start_impl.hpp sample_initialization.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.89.1.2 **set()** [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 35 of file CMakeLists.txt.

References `add_cli_executable()`.

40.90 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lars/CMakeLists.txt File Reference

Functions

- **set** (SOURCES lars.hpp lars_impl.hpp lars.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.90.1 Function Documentation

40.90.1.1 **set()** [1/2]

```
set (
    SOURCES lars.hpp lars_impl.hpp lars.  cpp )
```

Definition at line 3 of file CMakeLists.txt.

40.90.1.2 **set()** [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 12 of file CMakeLists.txt.

References `add_cli_executable()`.

40.91 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear_regression/CMakeLists.txt File Reference

Functions

- **set** (SOURCES linear_regression.hpp linear_regression.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.91.1 Function Documentation

40.91.1.1 set() [1/2]

```
set (
    SOURCES linear_regression.hpp linear_regression.  cpp )
```

Definition at line 4 of file CMakeLists.txt.

40.91.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file}  )
```

Definition at line 12 of file CMakeLists.txt.

References `add_cli_executable()`.

40.92 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear_svm/CMakeLists.txt File Reference

Functions

- **set** (SOURCES linear_svm.hpp linear_svm_impl.hpp linear_svm_function.hpp linear_svm_function_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.92.1 Function Documentation

40.92.1.1 set() [1/2]

```
set (
    SOURCES linear_svm.hpp linear_svm_impl.hpp linear_svm_function.hpp linear_svm_↵
function_impl. .hpp )
```

Definition at line 4 of file CMakeLists.txt.

40.92.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 14 of file CMakeLists.txt.

40.93 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lmnn/CMakeLists.txt File Reference

Functions

- `set` (SOURCES lmnn.hpp lmnn_impl.hpp lmnn_function.hpp lmnn_function_impl.hpp constraints.hpp constraints_impl.hpp) `set`(DIR_SRCS) `foreach`(file \$
- `set` (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) `endforeach`() `set`(MLPACK_SRCS \$

40.93.1 Function Documentation

40.93.1.1 `set()` [1/2]

```
set (
    SOURCES lmnn.hpp lmnn_impl.hpp lmnn_function.hpp lmnn_function_impl.hpp constraints.↵
    hpp constraints_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.93.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 15 of file CMakeLists.txt.

References `add_cli_executable()`.

40.94 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/local_coordinate_coding/CMakeLists.txt File Reference

Functions

- **set** (SOURCES lcc.hpp lcc.cpp lcc_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.94.1 Function Documentation

40.94.1.1 set() [1/2]

```
set (
    SOURCES lcc.hpp lcc.cpp lcc_impl. hpp )
```

Definition at line 6 of file CMakeLists.txt.

40.94.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 15 of file CMakeLists.txt.

References `add_cli_executable()`.

40.95 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/logistic_regression/CMakeLists.txt File Reference

Functions

- **set** (SOURCES logistic_regression.hpp logistic_regression_impl.hpp logistic_regression_function.hpp logistic_regression_function_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.95.1 Function Documentation

40.95.1.1 `set()` [1/2]

```
set (
    SOURCES logistic_regression.hpp logistic_regression_impl.hpp logistic_regression_↔
function.hpp logistic_regression_function_impl.  hpp )
```

Definition at line 4 of file CMakeLists.txt.

40.95.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 14 of file CMakeLists.txt.

References `add_cli_executable()`.

40.96 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lsh/CMakeLists.txt File Reference

Functions

- **set** (SOURCES lsh_search.hpp lsh_search_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.96.1 Function Documentation

40.96.1.1 `set()` [1/2]

```
set (
    SOURCES lsh_search.hpp lsh_search_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.96.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 12 of file CMakeLists.txt.

40.97 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/matrix_completion/CMakeLists.txt File Reference

Functions

- **set** (SOURCES matrix_completion.hpp matrix_completion.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.97.1 Function Documentation

40.97.1.1 set() [1/2]

```
set (
    SOURCES matrix_completion.hpp matrix_completion.  cpp )
```

Definition at line 3 of file CMakeLists.txt.

40.97.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 11 of file CMakeLists.txt.

40.98 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/mean_shift/CMakeLists.txt File Reference

Functions

- **set** (SOURCES mean_shift.hpp mean_shift_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.98.1 Function Documentation

40.98.1.1 `set()` [1/2]

```
set (
    SOURCES mean_shift.hpp mean_shift_impl.  _hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.98.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 11 of file CMakeLists.txt.

References `add_cli_executable()`.

40.99 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/naive_bayes/CMakeLists.txt File Reference

Functions

- `set` (SOURCES naive_bayes_classifier.hpp naive_bayes_classifier_impl.hpp) `set`(DIR_SRCS) `foreach`(file \$
- `set` (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) `endforeach`() `set`(MLPACK_SRCS \$

40.99.1 Function Documentation

40.99.1.1 `set()` [1/2]

```
set (
    SOURCES naive_bayes_classifier.hpp naive_bayes_classifier_impl.  _hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.99.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 11 of file CMakeLists.txt.

References `add_cli_executable()`.

40.100 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nca/CMakeLists.txt File Reference

Functions

- **set** (SOURCES nca.hpp nca_impl.hpp nca_softmax_error_function.hpp nca_softmax_error_function_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.100.1 Function Documentation

40.100.1.1 set() [1/2]

```
set (
    SOURCES nca.hpp nca_impl.hpp nca_softmax_error_function.hpp nca_softmax_error_↵
function_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.100.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 13 of file CMakeLists.txt.

References `add_cli_executable()`.

40.101 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/CMakeLists.txt File Reference

Functions

- **set** (SOURCES neighbor_search.hpp neighbor_search_impl.hpp neighbor_search_rules.hpp neighbor_search_rules_impl.hpp neighbor_search_stat.hpp ns_model.hpp ns_model_impl.hpp sort_policies/nearest_neighbor_sort.hpp sort_policies/nearest_neighbor_sort_impl.hpp sort_policies/furthest_neighbor_sort.hpp sort_policies/furthest_neighbor_sort_impl.hpp typedef.hpp unmap.hpp unmap.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.101.1 Function Documentation

40.101.1.1 set() [1/2]

```
set (
    SOURCES neighbor_search.hpp neighbor_search_impl.hpp neighbor_search_rules.hpp neighbor_search_rules_impl.hpp neighbor_search_stat.hpp ns_model.hpp ns_model_impl.hpp sort_policies/nearest_neighbor_sort.hpp sort_policies/nearest_neighbor_sort_impl.hpp sort_policies/furthest_neighbor_sort.hpp sort_policies/furthest_neighbor_sort_impl.hpp typedef.hpp unmap.hpp unmap.  cpp )
```

Definition at line 3 of file CMakeLists.txt.

40.101.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 23 of file CMakeLists.txt.

References `add_cli_executable()`.

40.102 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nmf/CMakeLists.txt File Reference

Functions

- **add_cli_executable** (nmf) **add_python_binding** (nmf) **add_markdown_docs** (nmf "cli

40.102.1 Function Documentation

40.102.1.1 add_cli_executable()

```
add_cli_executable (
    nmf )
```

Referenced by set().

40.103 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nystroem_method/C↵ MakeLists.txt File Reference

Functions

- **set** (SOURCES nystroem_method.hpp nystroem_method_impl.hpp ordered_selection.hpp random_selection.↵
hpp kmeans_selection.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS
\$

40.103.1 Function Documentation

40.103.1.1 set() [1/2]

```
set (
    SOURCES nystroem_method.hpp nystroem_method_impl.hpp ordered_selection.hpp random_↵  
selection.hpp kmeans_selection.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.103.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 14 of file CMakeLists.txt.

40.104 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/CMakeLists.txt File Reference

Functions

- **set** (SOURCES pca.hpp pca_impl.hpp) **add_subdirectory**(decomposition_policies) **set**(DIR_SRCS) **foreach**(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) **endforeach**() **set**(MLPACK_SRCS \$

40.104.1 Function Documentation

40.104.1.1 **set**() [1/2]

```
set (
    SOURCES pca.hpp pca_impl. .hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.104.1.2 **set**() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 15 of file CMakeLists.txt.

References `add_cli_executable()`.

40.105 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/decomposition_policies/CMakeLists.txt File Reference

Functions

- **set** (SOURCES exact_svd_method.hpp randomized_block_krylov_method.hpp randomized_svd_method.hpp quic_svd_method.hpp) **set**(DIR_SRCS) **foreach**(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) **endforeach**() **set**(MLPACK_SRCS \$

40.105.1 Function Documentation

40.105.1.1 `set()` [1/2]

```
set (
    SOURCES exact_svd_method.hpp randomized_block_krylov_method.hpp randomized_svd_↵
method.hpp quic_svd_method. .hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.105.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 13 of file CMakeLists.txt.

40.106 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/CMake↵ Lists.txt File Reference

Functions

- **set** (SOURCES perceptron.hpp perceptron_impl.hpp) **add_subdirectory**(initialization_methods) **add_↵**
subdirectory(learning_policies) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS
\$

40.106.1 Function Documentation

40.106.1.1 `set()` [1/2]

```
set (
    SOURCES perceptron.hpp perceptron_impl. .hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.106.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 14 of file CMakeLists.txt.

References `add_cli_executable()`.

40.107 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/initialization_↵ _methods/CMakeLists.txt File Reference

Functions

- `set` (SOURCES `random_init.hpp zero_init.hpp`) `set(DIR_SRCS)` `foreach(file $`
- `set` (DIR_SRCS `${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file}`) `endforeach()` `set(MLPACK_SRCS`
`$`

40.107.1 Function Documentation

40.107.1.1 `set()` [1/2]

```
set (
    SOURCES random_init.hpp zero_init. hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.107.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 11 of file CMakeLists.txt.

40.108 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/learning_↵ policies/CMakeLists.txt File Reference

Functions

- `set` (SOURCES `simple_weight_update.hpp`) `set(DIR_SRCS)` `foreach(file $`
- `set` (DIR_SRCS `${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file}`) `endforeach()` `set(MLPACK_SRCS`
`$`

40.108.1 Function Documentation

40.108.1.1 `set()` [1/2]

```
set (
    SOURCES simple_weight_update.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.108.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file}  )
```

Definition at line 10 of file CMakeLists.txt.

40.109 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/preprocess/CMake↵ Lists.txt File Reference

Functions

- **set** (SOURCES) `set(DIR_SRCS) foreach(file $`
- **set** (DIR_SRCS `${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file}) endforeach() set(MLPACK_SRCS`
`$`

40.109.1 Function Documentation

40.109.1.1 `set()` [1/2]

```
set (
    SOURCES  )
```

Definition at line 3 of file CMakeLists.txt.

40.109.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 9 of file CMakeLists.txt.

References `add_cli_executable()`.

40.110 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/quic_svd/CMakeLists.txt File Reference

Functions

- `set` (SOURCES quic_svd.hpp quic_svd.cpp) `set`(DIR_SRCS) `foreach`(file \$
- `set` (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) `endforeach`() `set`(MLPACK_SRCS \$

40.110.1 Function Documentation

40.110.1.1 `set()` [1/2]

```
set (
    SOURCES quic_svd.hpp quic_svd.  cpp )
```

Definition at line 3 of file CMakeLists.txt.

40.110.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 11 of file CMakeLists.txt.

40.111 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/radical/CMakeLists.txt File Reference

Functions

- `set` (SOURCES radical.hpp radical.cpp) `set`(DIR_SRCS) `foreach`(file \$
- `set` (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) `endforeach`() `set`(MLPACK_SRCS \$

40.111.1 Function Documentation

40.111.1.1 `set()` [1/2]

```
set (
    SOURCES radical.hpp radical.  cpp )
```

Definition at line 3 of file CMakeLists.txt.

40.111.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 11 of file CMakeLists.txt.

References `add_cli_executable()`.

40.112 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/random_forest/CMakeLists.txt File Reference

Functions

- `set` (SOURCES bootstrap.hpp random_forest.hpp random_forest_impl.hpp) `set`(DIR_SRCS) `foreach`(file \$
- `set` (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) `endforeach`() `set`(MLPACK_SRCS \$

40.112.1 Function Documentation

40.112.1.1 `set()` [1/2]

```
set (
    SOURCES bootstrap.hpp random_forest.hpp random_forest_impl. .hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.112.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 12 of file CMakeLists.txt.

References `add_cli_executable()`.

40.113 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/randomized_svd/CMakeLists.txt File Reference

Functions

- **set** (SOURCES randomized_svd.hpp randomized_svd.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.113.1 Function Documentation

40.113.1.1 `set()` [1/2]

```
set (
    SOURCES randomized_svd.hpp randomized_svd.  cpp )
```

Definition at line 3 of file CMakeLists.txt.

40.113.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 11 of file CMakeLists.txt.

40.114 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/CMakeLists.txt File Reference

Functions

- **set** (SOURCES range_search.hpp range_search_impl.hpp range_search_rules.hpp range_search_rules_impl.hpp range_search_stat.hpp rs_model.hpp rs_model_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.114.1 Function Documentation

40.114.1.1 **set()** [1/2]

```
set (
    SOURCES range_search.hpp range_search_impl.hpp range_search_rules.hpp range_search_
rules_impl.hpp range_search_stat.hpp rs_model.hpp rs_model_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.114.1.2 **set()** [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 16 of file CMakeLists.txt.

References `add_cli_executable()`.

40.115 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/CMakeLists.txt File Reference

Functions

- **set** (SOURCES ra_search.hpp ra_search_impl.hpp ra_search_rules.hpp ra_search_rules_impl.hpp ra_query_
stat.hpp ra_typedef.hpp ra_util.hpp ra_util.cpp ra_model.hpp ra_model_impl.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.115.1 Function Documentation

40.115.1.1 `set()` [1/2]

```
set (
    SOURCES ra_search.hpp ra_search_impl.hpp ra_search_rules.hpp ra_search_rules_impl.hpp
    ra_query_stat.hpp ra_typedef.hpp ra_util.hpp ra_util.cpp ra_model.hpp ra_model_impl.  hpp )
```

Definition at line 4 of file CMakeLists.txt.

40.115.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 31 of file CMakeLists.txt.

40.116 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/regularized_svd/CMakeLists.txt File Reference

Functions

- `set` (SOURCES regularized_svd.hpp regularized_svd_impl.hpp regularized_svd_function.hpp regularized_svd_function_impl.hpp) `set` (DIR_SRCS) `foreach` (file \$
- `set` (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) `endforeach` () `set` (MLPACK_SRCS \$

40.116.1 Function Documentation

40.116.1.1 `set()` [1/2]

```
set (
    SOURCES regularized_svd.hpp regularized_svd_impl.hpp regularized_svd_function.hpp
    regularized_svd_function_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.116.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 13 of file CMakeLists.txt.

40.117 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/↵ CMakeLists.txt File Reference

Functions

- **set** (SOURCES async_learning.hpp async_learning_impl.hpp q_learning.hpp q_learning_impl.hpp training_↵
config.hpp) **add_subdirectory**(environment) **add_subdirectory**(policy) **add_subdirectory**(replay) **add_↵
subdirectory**(worker) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS
\$

40.117.1 Function Documentation

40.117.1.1 set() [1/2]

```
set (
    SOURCES async_learning.hpp async_learning_impl.hpp q_learning.hpp q_learning_impl.hpp
    training_config.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.117.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 19 of file CMakeLists.txt.

40.118 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/↵ CMakeLists.txt File Reference

Functions

- **set** (SOURCES mountain_car.hpp cart_pole.hpp continuous_mountain_car.hpp acrobot.hpp pendulum.hpp reward_clipping.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.118.1 Function Documentation

40.118.1.1 set() [1/2]

```
set (
    SOURCES mountain_car.hpp cart_pole.hpp continuous_mountain_car.hpp acrobot.hpp pendulum.↵
    hpp reward_clipping. hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.118.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 15 of file CMakeLists.txt.

40.119 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/policy/↵ CMakeLists.txt File Reference

Functions

- **set** (SOURCES aggregated_policy.hpp greedy_policy.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.119.1 Function Documentation

40.119.1.1 set() [1/2]

```
set (
    SOURCES aggregated_policy.hpp greedy_policy. .hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.119.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 11 of file CMakeLists.txt.

40.120 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/replay/↵ CMakeLists.txt File Reference

Functions

- **set** (SOURCES random_replay.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.120.1 Function Documentation

40.120.1.1 set() [1/2]

```
set (
    SOURCES random_replay. .hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.120.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 10 of file CMakeLists.txt.

40.121 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/CMakeLists.txt File Reference

Functions

- **set** (SOURCES one_step_q_learning_worker.hpp one_step_sarsa_worker.hpp n_step_q_learning_worker.hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.121.1 Function Documentation

40.121.1.1 **set()** [1/2]

```
set (
    SOURCES one_step_q_learning_worker.hpp one_step_sarsa_worker.hpp n_step_q_learning_↵
worker.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.121.1.2 **set()** [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 12 of file CMakeLists.txt.

40.122 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/softmax_regression/CMakeLists.txt File Reference

Functions

- **set** (SOURCES softmax_regression.hpp softmax_regression.cpp softmax_regression_impl.hpp softmax_↵
regression_function.hpp softmax_regression_function.cpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS \$

40.122.1 Function Documentation

40.122.1.1 `set()` [1/2]

```
set (
    SOURCES softmax_regression.hpp softmax_regression.cpp softmax_regression_impl.hpp
softmax_regression_function.hpp softmax_regression_function.  cpp )
```

Definition at line 3 of file CMakeLists.txt.

40.122.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 14 of file CMakeLists.txt.

References `add_cli_executable()`.

40.123 `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_autoencoder/CMakeLists.txt` File Reference

Functions

- **set** (SOURCES sparse_autoencoder.hpp sparse_autoencoder.cpp sparse_autoencoder_impl.hpp sparse_↵
autoencoder_function.hpp sparse_autoencoder_function.cpp maximal_inputs.hpp maximal_inputs.cpp) set(D↵
IR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS
\$

40.123.1 Function Documentation

40.123.1.1 `set()` [1/2]

```
set (
    SOURCES sparse_autoencoder.hpp sparse_autoencoder.cpp sparse_autoencoder_impl.hpp
sparse_autoencoder_function.hpp sparse_autoencoder_function.cpp maximal_inputs.hpp maximal_inputs.
cpp )
```

Definition at line 3 of file CMakeLists.txt.

40.123.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 16 of file CMakeLists.txt.

40.124 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_coding/CMakeLists.txt File Reference

Functions

- `set` (SOURCES data_dependent_random_initializer.hpp nothing_initializer.hpp random_initializer.hpp sparse_coding.hpp sparse_coding.cpp sparse_coding_impl.hpp) `set(DIR_SRCS)` `foreach(file $`
- `set` (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) `endforeach()` `set(MLPACK_SRCS $`

40.124.1 Function Documentation

40.124.1.1 `set()` [1/2]

```
set (
    SOURCES data_dependent_random_initializer.hpp nothing_initializer.hpp random_initializer.
hpp sparse_coding.hpp sparse_coding.cpp sparse_coding_impl.  hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.124.1.2 `set()` [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 15 of file CMakeLists.txt.

References `add_cli_executable()`.

40.125 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/svdplusplus/CMakeLists.txt File Reference

Functions

- **set** (SOURCES svdplusplus.hpp svdplusplus_impl.hpp svdplusplus_function.hpp svdplusplus_function_impl.↵
hpp) set(DIR_SRCS) foreach(file \$
- **set** (DIR_SRCS \${DIR_SRCS} \${CMAKE_CURRENT_SOURCE_DIR}/\${file}) endforeach() set(MLPACK_SRCS
\$

40.125.1 Function Documentation

40.125.1.1 set() [1/2]

```
set (
    SOURCES svdplusplus.hpp svdplusplus_impl.hpp svdplusplus_function.hpp svdplusplus_↵
function_impl. hpp )
```

Definition at line 3 of file CMakeLists.txt.

40.125.1.2 set() [2/2]

```
set (
    DIR_SRCS ${DIR_SRCS} ${CMAKE_CURRENT_SOURCE_DIR}/${file} )
```

Definition at line 13 of file CMakeLists.txt.

40.126 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/CMakeLists.txt File Reference

Functions

- **add_executable** (mlpack_test_activation_functions_test.cpp adaboost_test.cpp akfn_test.cpp aknn_test.↵
cpp ann_dist_test.cpp ann_layer_test.cpp ann_test_tools.hpp arma_extend_test.cpp armadillo_svd_test.cpp
async_learning_test.cpp augmented_rnn_tasks_test.cpp bias_svd_test.cpp binarize_test.cpp block_krylov_↵
svd_test.cpp cf_test.cpp cli_binding_test.cpp cli_test.cpp convolution_test.cpp convolutional_network_test.cpp
cosine_tree_test.cpp cv_test.cpp dbscan_test.cpp decision_stump_test.cpp decision_tree_test.cpp det_test.↵
cpp distribution_test.cpp drusilla_select_test.cpp emst_test.cpp fastmks_test.cpp feedforward_network_test.↵
cpp gmm_test.cpp hmm_test.cpp hoeffding_tree_test.cpp hpt_test.cpp hyperplane_test.cpp imputation_test.↵
cpp init_rules_test.cpp kde_test.cpp kernel_pca_test.cpp kernel_test.cpp kernel_traits_test.cpp kfn_test.cpp

```

kmeans_test.cpp knn_test.cpp krann_search_test.cpp ksinit_test.cpp lars_test.cpp lin_alg_test.cpp linear_
regression_test.cpp linear_svm_test.cpp lmnn_test.cpp load_save_test.cpp local_coordinate_coding_test.
cpp log_test.cpp logistic_regression_test.cpp loss_functions_test.cpp lsh_test.cpp math_test.cpp matrix_
completion_test.cpp maximal_inputs_test.cpp mean_shift_test.cpp metric_test.cpp mlpack_test.cpp mock_
_categorical_data.hpp nbc_test.cpp nca_test.cpp nmf_test.cpp nystroem_method_test.cpp octree_test.cpp
pca_test.cpp perceptron_test.cpp prefixedostream_test.cpp python_binding_test.cpp q_learning_test.cpp
qdafn_test.cpp quic_svd_test.cpp radical_test.cpp random_forest_test.cpp random_test.cpp randomized_
svd_test.cpp range_search_test.cpp rbm_network_test.cpp rectangle_tree_test.cpp recurrent_network_test.cpp
regularized_svd_test.cpp reward_clipping_test.cpp rl_components_test.cpp serialization.cpp serialization.hpp
serialization_test.cpp sfinae_test.cpp softmax_regression_test.cpp sort_policy_test.cpp sparse_autoencoder_
test.cpp sparse_coding_test.cpp spill_tree_test.cpp split_data_test.cpp svd_batch_test.cpp svd_incremental_
test.cpp svdplusplus_test.cpp termination_policy_test.cpp test_function_tools.hpp test_tools.hpp timer_test.cpp
tree_test.cpp tree_traits_test.cpp ub_tree_test.cpp union_find_test.cpp vantage_point_tree_test.cpp main_
tests/test_helper.hpp main_tests/emst_test.cpp main_tests/adaboost_test.cpp main_tests/approx_kfn_test.cpp
main_tests/cf_test.cpp main_tests/dbscan_test.cpp main_tests/det_test.cpp main_tests/decision_tree_test.cpp
main_tests/decision_stump_test.cpp main_tests/fastmks_test.cpp main_tests/kde_test.cpp main_tests/linear_
_regression_test.cpp main_tests/logistic_regression_test.cpp main_tests/local_coordinate_coding_test.cpp
main_tests/lmnn_test.cpp main_tests/lsh_test.cpp main_tests/mean_shift_test.cpp main_tests/nbc_test.cpp
main_tests/nca_test.cpp main_tests/nmf_test.cpp main_tests/pca_test.cpp main_tests/perceptron_test.cpp
main_tests/preprocess_binarize_test.cpp main_tests/preprocess_imputer_test.cpp main_tests/preprocess_
split_test.cpp main_tests/random_forest_test.cpp main_tests/softmax_regression_test.cpp main_tests/sparse_
_coding_test.cpp main_tests/kmeans_test.cpp main_tests/hoeffding_tree_test.cpp main_tests/hmm_viterbi_
test.cpp main_tests/hmm_train_test.cpp main_tests/hmm_loglik_test.cpp main_tests/hmm_generate_test.cpp
main_tests/radical_test.cpp main_tests/hmm_test_utils.hpp main_tests/kernel_pca_test.cpp main_tests/range_
_search_test.cpp) target_link_libraries(mlpack_test mlpack $

```

40.126.1 Function Documentation

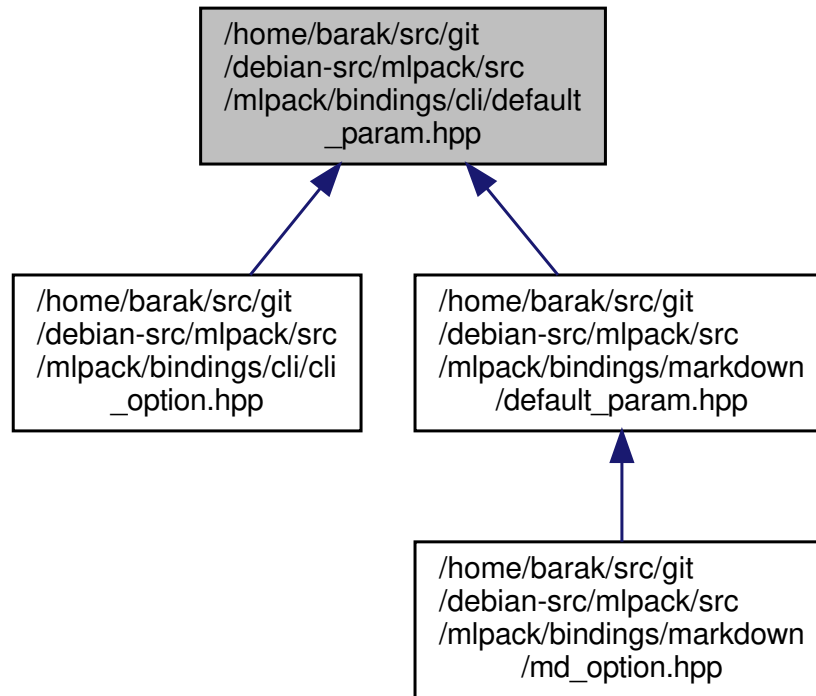
40.126.1.1 add_executable()

```

add_executable (
    mlpack_test activation_functions_test.cpp adaboost_test.cpp akfn_test.cpp aknn_
test.cpp ann_dist_test.cpp ann_layer_test.cpp ann_test_tools.hpp arma_extend_test.cpp armadillo_
_svd_test.cpp async_learning_test.cpp augmented_rnn_tasks_test.cpp bias_svd_test.cpp binarize_
_test.cpp block_krylov_svd_test.cpp cf_test.cpp cli_binding_test.cpp cli_test.cpp convolution_
_test.cpp convolutional_network_test.cpp cosine_tree_test.cpp cv_test.cpp dbscan_test.cpp decision_
_stump_test.cpp decision_tree_test.cpp det_test.cpp distribution_test.cpp drusilla_select_test.cpp
emst_test.cpp fastmks_test.cpp feedforward_network_test.cpp gmm_test.cpp hmm_test.cpp hoeffding_
_tree_test.cpp hpt_test.cpp hyperplane_test.cpp imputation_test.cpp init_rules_test.cpp kde_
_test.cpp kernel_pca_test.cpp kernel_test.cpp kernel_traits_test.cpp kfn_test.cpp kmeans_test.
cpp knn_test.cpp krann_search_test.cpp ksinit_test.cpp lars_test.cpp lin_alg_test.cpp linear_
regression_test.cpp linear_svm_test.cpp lmnn_test.cpp load_save_test.cpp local_coordinate_coding_
_test.cpp log_test.cpp logistic_regression_test.cpp loss_functions_test.cpp lsh_test.cpp math_
_test.cpp matrix_completion_test.cpp maximal_inputs_test.cpp mean_shift_test.cpp metric_test.cpp
mlpack_test.cpp mock_categorical_data.hpp nbc_test.cpp nca_test.cpp nmf_test.cpp nystroem_method_
_test.cpp octree_test.cpp pca_test.cpp perceptron_test.cpp prefixedostream_test.cpp python_
binding_test.cpp q_learning_test.cpp qdafn_test.cpp quic_svd_test.cpp radical_test.cpp random_
_forest_test.cpp random_test.cpp randomized_svd_test.cpp range_search_test.cpp rbm_network_

```


This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::cli**

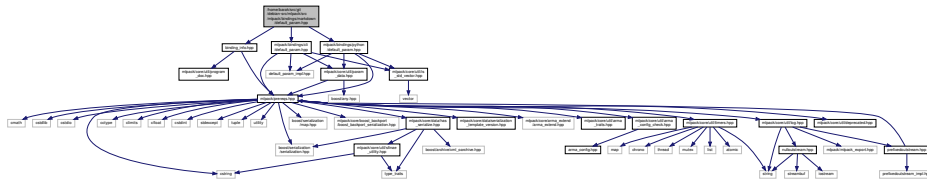
Functions

- template<typename T >
void **DefaultParam** (const util::ParamData &data, const void *, void *output)
Return the default value of an option.
- template<typename T >
std::string **DefaultParamImpl** (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_<_type< T >>::type !=0, const typename boost::disable_if< util::IsStdVector< T >>::type !=0, const typename boost::disable_if< data::HasSerialize< T >>::type !=0, const typename boost::disable_if< std::is_same< T, std::string >>::type !=0, const typename boost::disable_if< std::is_same< T, std::tuple< **mlpack::data::DatasetInfo**, arma::mat >>>::type !=0)
Return the default value of an option.
- template<typename T >
std::string **DefaultParamImpl** (const util::ParamData &data, const typename boost::enable_if< util::IsStdVector< T >>::type !=0)
Return the default value of a vector option.

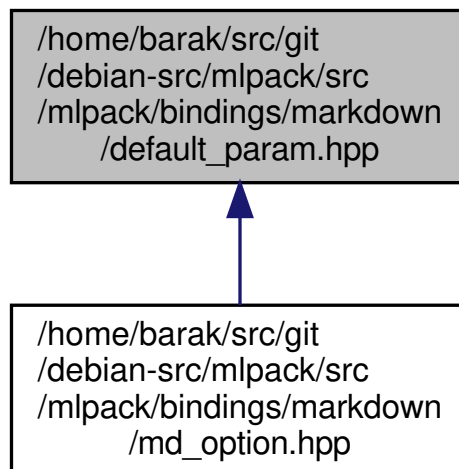
- `template<typename T >`
`std::string DefaultParamImpl (const util::ParamData &data, const typename boost::enable_if< std::is_same<`
`T, std::string >>::type *=0)`
Return the default value of a string option.
- `template<typename T >`
`std::string DefaultParamImpl (const util::ParamData &data, const typename boost::enable_if_c< arma::is_↵`
`arma_type< T >::value||std::is_same< T, std::tuple< mlpack::data::DatasetInfo, arma::mat >>::value >↵`
`::type *=0)`
Return the default value of a matrix option, a tuple option, a serializable option, or a string option (this returns the default filename, or " if the default is no file).
- `template<typename T >`
`std::string DefaultParamImpl (const util::ParamData &data, const typename boost::disable_if< arma::is_arma↵`
`_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)`
Return the default value of a model option (this returns the default filename, or " if the default is no file).

40.128 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/default_↵ param.hpp File Reference

Include dependency graph for default_param.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

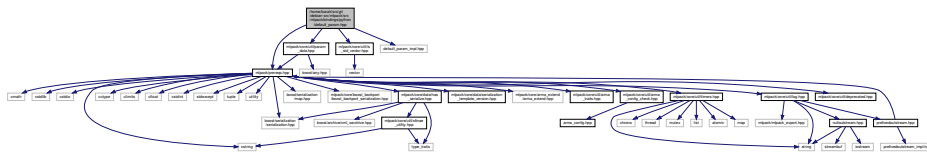
- **mlpack**
 - .hpp
- **mlpack::bindings**
- **mlpack::bindings::markdown**

Functions

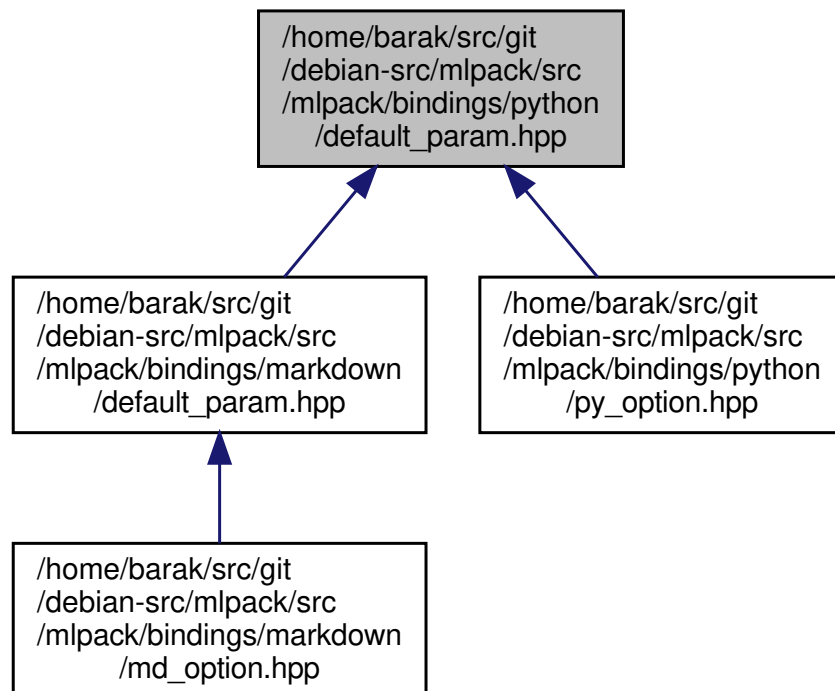
- `template<typename T>`
void DefaultParam (const util::ParamData &data, const void *, void *output)
Print the default value of a parameter into the output string.

40.129 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/default_param.hpp File Reference

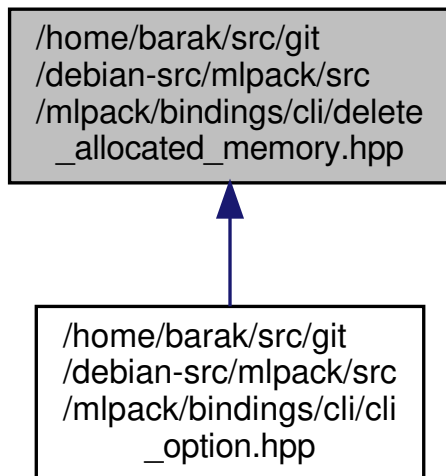
Include dependency graph for default_param.hpp:



This graph shows which files directly or indirectly include this file:



This graph shows which files directly or indirectly include this file:



Namespaces

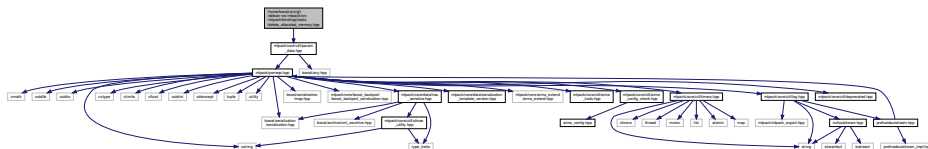
- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::cli**

Functions

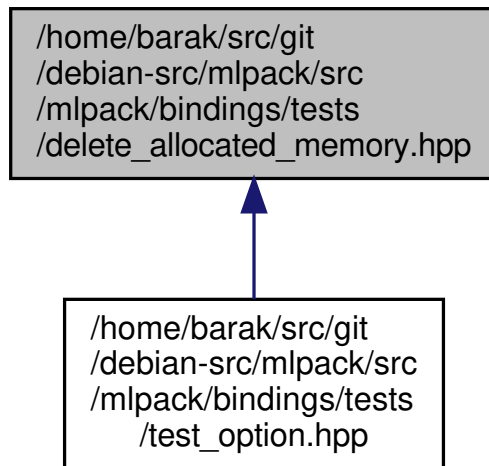
- template<typename T >
 void **DeleteAllocatedMemory** (const util::ParamData &d, const void *, void *)
- template<typename T >
 void **DeleteAllocatedMemoryImpl** (const util::ParamData &, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0)
- template<typename T >
 void **DeleteAllocatedMemoryImpl** (const util::ParamData &, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)
- template<typename T >
 void **DeleteAllocatedMemoryImpl** (const util::ParamData &d, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)

40.131 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/delete_allocated_memory.hpp File Reference

Include dependency graph for delete_allocated_memory.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::tests**

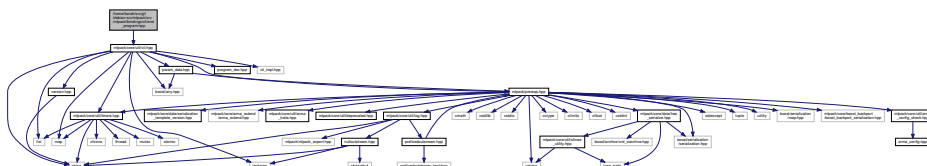
Functions

- `template<typename T >`
 `void DeleteAllocatedMemory (const util::ParamData &d, const void *, void *)`
- `template<typename T >`
 `void DeleteAllocatedMemoryImpl (const util::ParamData &, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0)`
- `template<typename T >`
 `void DeleteAllocatedMemoryImpl (const util::ParamData &, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)`
- `template<typename T >`
 `void DeleteAllocatedMemoryImpl (const util::ParamData &d, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)`

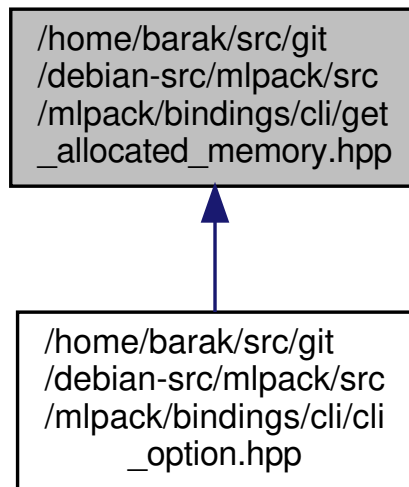
40.132 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/end_program.hpp

File Reference

Include dependency graph for `end_program.hpp`:



This graph shows which files directly or indirectly include this file:



Namespaces

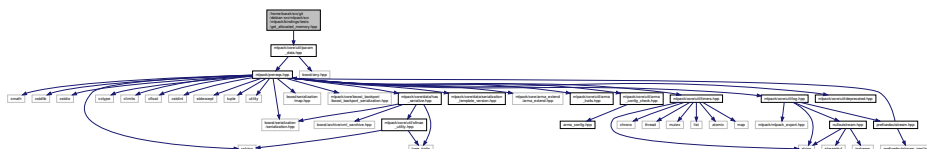
- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::cli**

Functions

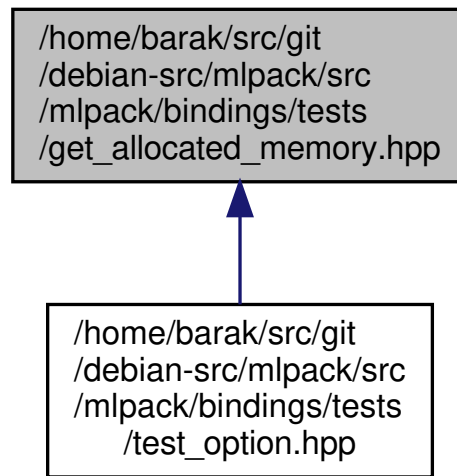
- `template<typename T >`
 `void * GetAllocatedMemory (const util::ParamData &, const typename boost::disable_if< data::HasSerialize< T >>::type !=0, const typename boost::disable_if< arma::is_arma_type< T >>::type !=0)`
- `template<typename T >`
 `void * GetAllocatedMemory (const util::ParamData &, const typename boost::enable_if< arma::is_arma_type< T >>::type !=0)`
- `template<typename T >`
 `void * GetAllocatedMemory (const util::ParamData &d, const typename boost::disable_if< arma::is_arma_type< T >>::type !=0, const typename boost::enable_if< data::HasSerialize< T >>::type !=0)`
- `template<typename T >`
 `void GetAllocatedMemory (const util::ParamData &d, const void *, void *output)`

40.134 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/get_allocated_↵ memory.hpp File Reference

Include dependency graph for `get_allocated_memory.hpp`:



This graph shows which files directly or indirectly include this file:



Namespaces

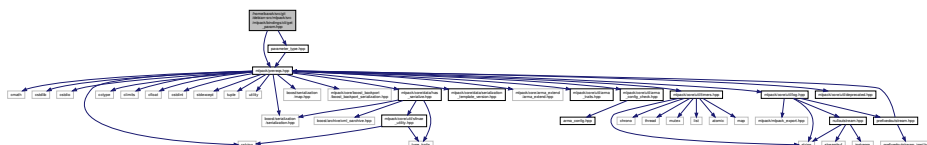
- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::tests**

Functions

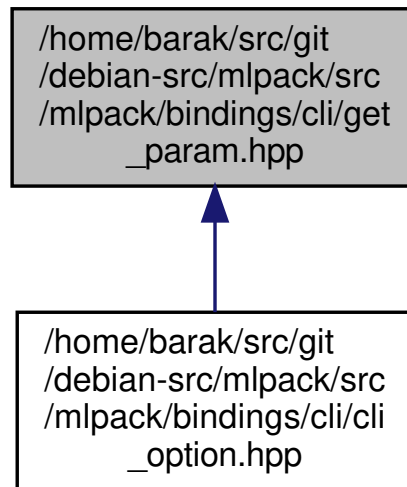
- template<typename T >
void * **GetAllocatedMemory** (const util::ParamData &, const typename boost::disable_if< data::HasSerialize< T >>::type !=0, const typename boost::disable_if< arma::is_arma_type< T >>::type !=0)
- template<typename T >
void * **GetAllocatedMemory** (const util::ParamData &, const typename boost::enable_if< arma::is_arma_type< T >>::type !=0)
- template<typename T >
void * **GetAllocatedMemory** (const util::ParamData &d, const typename boost::disable_if< arma::is_arma_type< T >>::type !=0, const typename boost::enable_if< data::HasSerialize< T >>::type !=0)
- template<typename T >
void **GetAllocatedMemory** (const util::ParamData &d, const void *, void *output)

40.135 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get_param.hpp File Reference

Include dependency graph for get_param.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

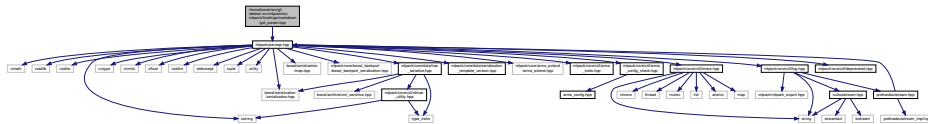
- **mlpack**
 .*.hpp*
- **mlpack::bindings**
- **mlpack::bindings::cli**

Functions

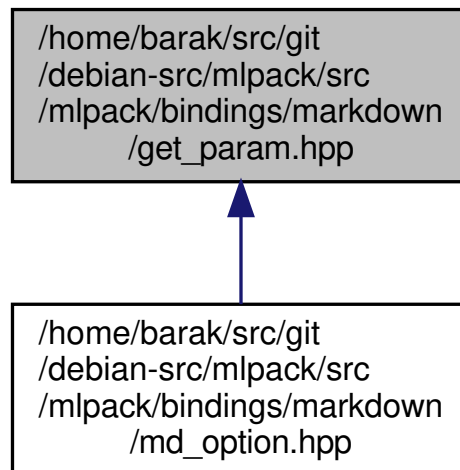
- `template<typename T >`
`T & GetParam (util::ParamData &d, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0,`
`const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if<`
`std::is_same< T, std::tuple< mlpack::data::DatasetInfo, arma::mat >>>::type *=0)`
This overload is called when nothing special needs to happen to the name of the parameter.
- `template<typename T >`
`T & GetParam (util::ParamData &d, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)`
Return a matrix parameter.
- `template<typename T >`
`T & GetParam (util::ParamData &d, const typename boost::enable_if< std::is_same< T, std::tuple< mlpack::data::DatasetInfo, arma::mat >>>::type *=0)`
Return a matrix/dataset info parameter.
- `template<typename T >`
`T * & GetParam (util::ParamData &d, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0,`
`const typename boost::enable_if< data::HasSerialize< T >>::type *=0)`
Return a serializable object.
- `template<typename T >`
`void GetParam (const util::ParamData &d, const void *, void *output)`
Return a parameter casted to the given type.

40.136 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/get_param.hpp File Reference

Include dependency graph for get_param.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::markdown**

Functions

- `template<typename T >`
 void **GetParam** (const util::ParamData &d, const void *, void *output)

All Markdown binding types are exactly what is held in the ParamData, so no special handling is necessary.

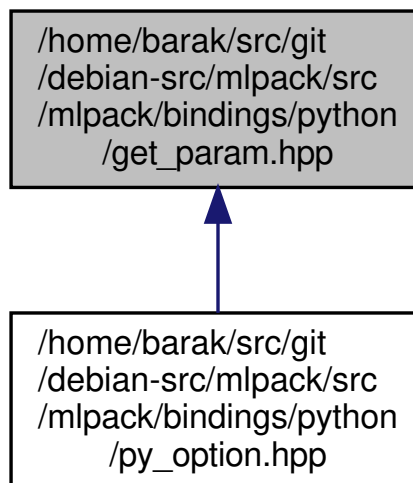
40.137 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/get_param.hpp

File Reference

Include dependency graph for get_param.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::python**

Functions

- `template<typename T >`
 void **GetParam** (const util::ParamData &d, const void *, void *output)

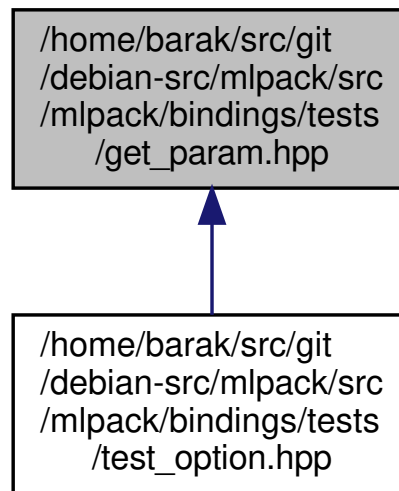
All Python binding types are exactly what is held in the ParamData, so no special handling is necessary.

40.138 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/get_param.hpp File Reference

Include dependency graph for get_param.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

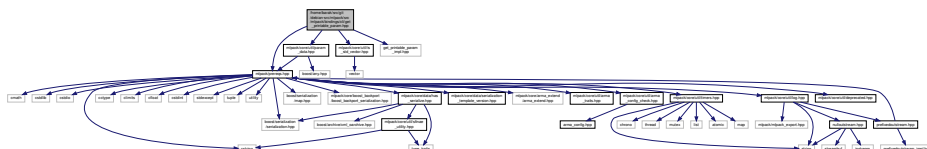
- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::tests**

Functions

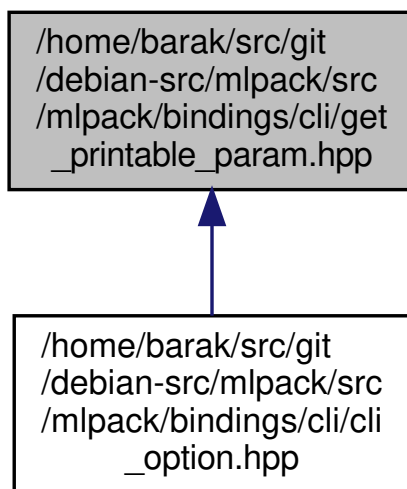
- `template<typename T >`
 T & **GetParam** (util::ParamData &d)
 This overload is called when nothing special needs to happen to the name of the parameter.
- `template<typename T >`
 void **GetParam** (const util::ParamData &d, const void *, void *output)
 Return a parameter casted to the given type.

40.139 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get_printable_param.hpp File Reference

Include dependency graph for get_printable_param.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::cli**

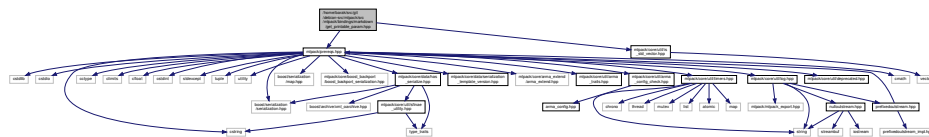
Functions

- `template<typename T >`
`std::string GetPrintableParam (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< util::IsStdVector< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)`
Print an option.

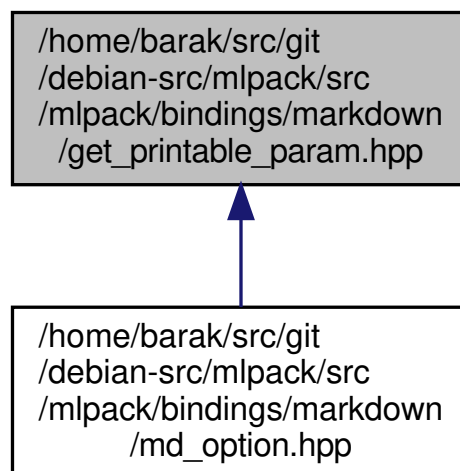
- Print a vector option, with spaces between it.*

_param.hpp File Reference

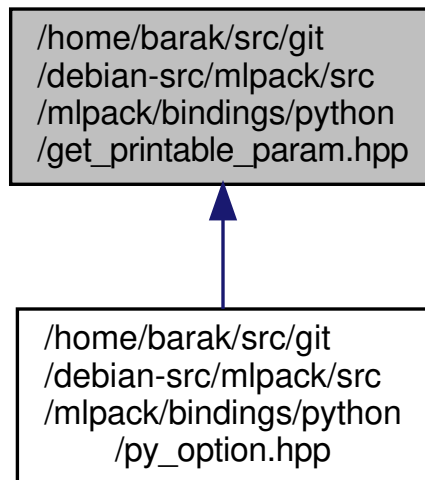
Include dependency graph for get_printable_param.hpp:



This graph shows which files directly or indirectly include this file:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::python**

Functions

- `template<typename T >`
`std::string GetPrintableParam (const util::ParamData &data, const typename boost::disable_if< arma::is_↵`
`arma_type< T >>::type !=0, const typename boost::disable_if< util::IsStdVector< T >>::type !=0, const type-`
`name boost::disable_if< data::HasSerialize< T >>::type !=0, const typename boost::disable_if< std::is_same<`
`T, std::tuple< data::DatasetInfo, arma::mat >>>::type !=0)`
Print an option of a simple type.
- `template<typename T >`
`std::string GetPrintableParam (const util::ParamData &data, const typename boost::enable_if< util::IsStd↵`
`Vector< T >>::type !=0)`
Print a vector option, with spaces between it.
- `template<typename T >`
`std::string GetPrintableParam (const util::ParamData &data, const typename boost::enable_if< arma::is_↵`
`arma_type< T >>::type !=0)`
Print a matrix option (this prints its size).
- `template<typename T >`
`std::string GetPrintableParam (const util::ParamData &data, const typename boost::disable_if< arma::is_↵`
`arma_type< T >>::type !=0, const typename boost::enable_if< data::HasSerialize< T >>::type !=0)`
Print a serializable class option (this prints the class name).
- `template<typename T >`
`std::string GetPrintableParam (const util::ParamData &data, const typename boost::enable_if< std::is_same<`
`T, std::tuple< data::DatasetInfo, arma::mat >>>::type !=0)`

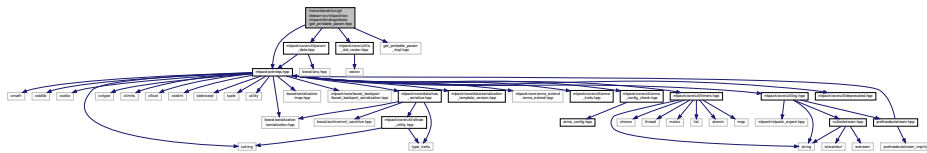
Print a combination DatasetInfo/matrix parameter.

- `template<typename T >`
`void GetPrintableParam (const util::ParamData &data, const void *, void *output)`

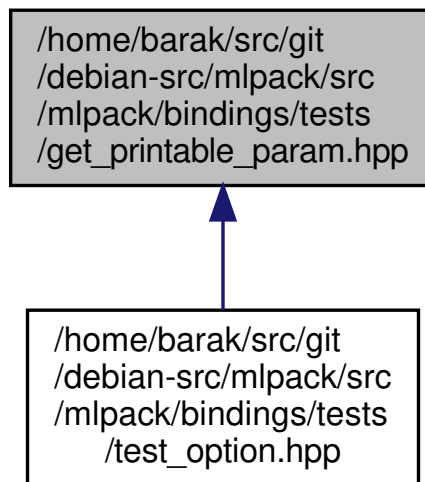
Print an option into a std::string.

40.142 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/get_printable_param.hpp File Reference

Include dependency graph for `get_printable_param.hpp`:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
`.hpp`
- **mlpack::bindings**
- **mlpack::bindings::tests**

Functions

- `template<typename T >`
`std::string GetPrintableParam` (const util::ParamData &data, const typename boost::disable_if< arma::is_↵
`arma_type< T >>::type !=0, const typename boost::disable_if< util::IsStdVector< T >>::type !=0, const type-`
`name boost::disable_if< data::HasSerialize< T >>::type !=0, const typename boost::disable_if< std::is_same<`
`T, std::tuple< data::DatasetInfo, arma::mat >>>::type !=0)`

Print an option.

- `template<typename T >`
`std::string GetPrintableParam` (const util::ParamData &data, const typename boost::enable_if< util::IsStd↵
`Vector< T >>::type !=0)`

Print a vector option, with spaces between it.

- `template<typename T >`
`std::string GetPrintableParam` (const util::ParamData &data, const typename boost::enable_if< arma::is_↵
`arma_type< T >>::type !=0)`

Print a matrix option (this just prints the filename).

- `template<typename T >`
`std::string GetPrintableParam` (const util::ParamData &data, const typename boost::disable_if< arma::is_↵
`arma_type< T >>::type !=0, const typename boost::enable_if< data::HasSerialize< T >>::type !=0)`

Print a serializable class option (this just prints the filename).

- `template<typename T >`
`std::string GetPrintableParam` (const util::ParamData &data, const typename boost::enable_if< std::is_same<
`T, std::tuple< data::DatasetInfo, arma::mat >>>::type !=0)`

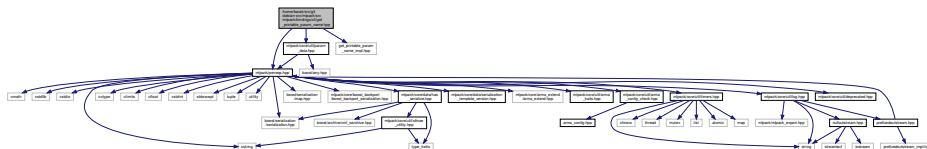
Print a mapped matrix option (this just prints the filename).

- `template<typename T >`
`void GetPrintableParam` (const util::ParamData &data, const void *, void *output)

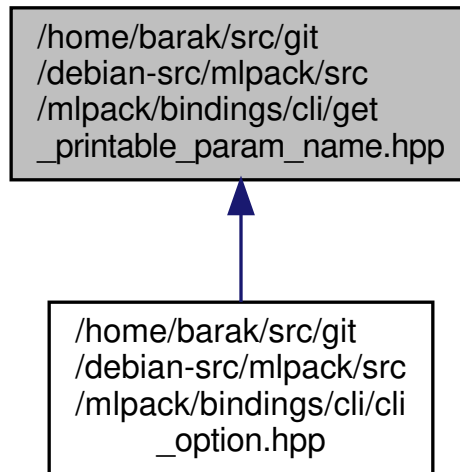
Print an option into a std::string.

40.143 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get_printable_↵ param_name.hpp File Reference

Include dependency graph for get_printable_param_name.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

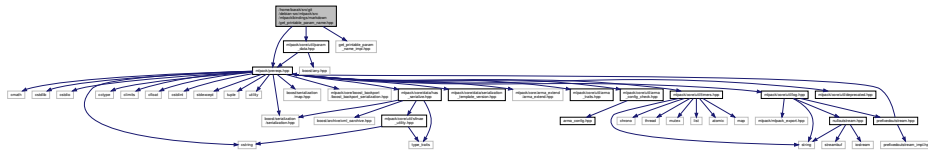
- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::cli**

Functions

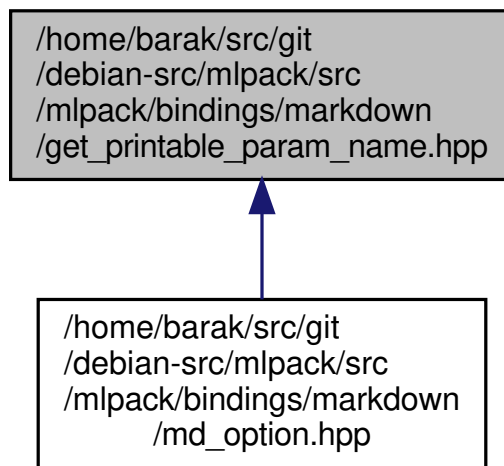
- template<typename T >
std::string **GetPrintableParamName** (const util::ParamData &data, const typename boost::disable_if< arma↵
::is_arma_type< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const
typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)
 Get the parameter name for a type that has no special handling.
- template<typename T >
std::string **GetPrintableParamName** (const util::ParamData &data, const typename boost::enable_if< arma↵
::is_arma_type< T >>::type *=0)
 Get the parameter name for a matrix type (where the user has to pass the file that holds the matrix).
- template<typename T >
std::string **GetPrintableParamName** (const util::ParamData &data, const typename boost::disable_if< arma↵
::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)
 Get the parameter name for a serializable model type (where the user has to pass the file that holds the matrix).
- template<typename T >
std::string **GetPrintableParamName** (const util::ParamData &data, const typename boost::enable_if< std::is↵
_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)
 Get the parameter name for a mapped matrix type (where the user has to pass the file that holds the matrix).
- template<typename T >
void **GetPrintableParamName** (const util::ParamData &d, const void *, void *output)
 Get the parameter's name as seen by the user.

40.144 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/get_printable_param_name.hpp File Reference

Include dependency graph for get_printable_param_name.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::markdown**

Functions

- template<typename T >
std::string **GetPrintableParamName** (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)
 Get the parameter name for a type that has no special handling.
- template<typename T >
std::string **GetPrintableParamName** (const util::ParamData &data, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)

Get the parameter name for a matrix type (where the user has to pass the file that holds the matrix).

- `template<typename T >`
`std::string GetPrintableParamName (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)`

Get the parameter name for a serializable model type (where the user has to pass the file that holds the matrix).

- `template<typename T >`
`std::string GetPrintableParamName (const util::ParamData &data, const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)`

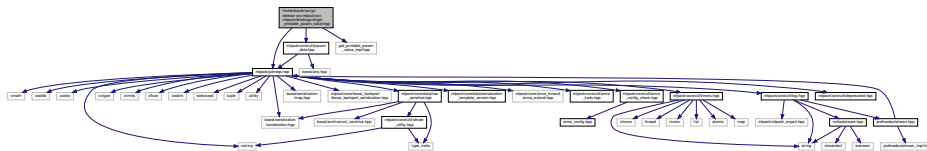
Get the parameter name for a mapped matrix type (where the user has to pass the file that holds the matrix).

- `template<typename T >`
`void GetPrintableParamName (const util::ParamData &d, const void *, void *output)`

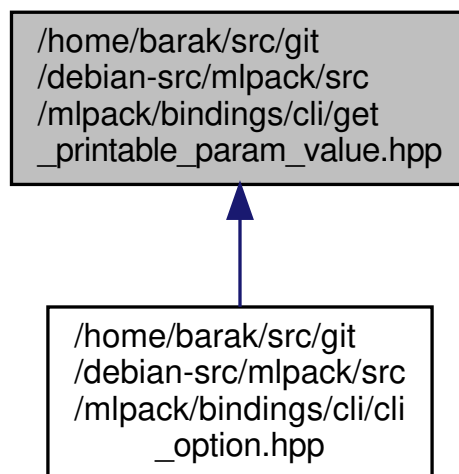
Get the parameter's name as seen by the user.

40.145 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get_printable_param_value.hpp File Reference

Include dependency graph for `get_printable_param_value.hpp`:



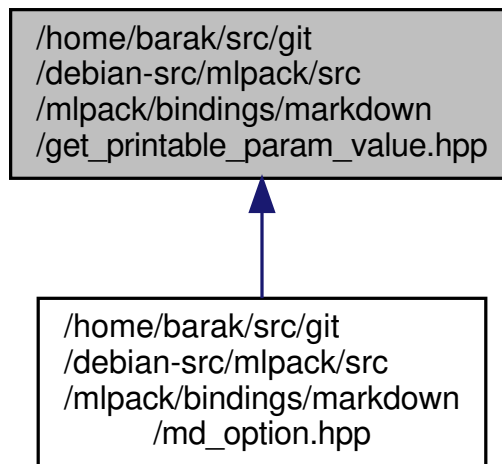
This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
`.hpp`
- **mlpack::bindings**
- **mlpack::bindings::cli**

This graph shows which files directly or indirectly include this file:



Namespaces

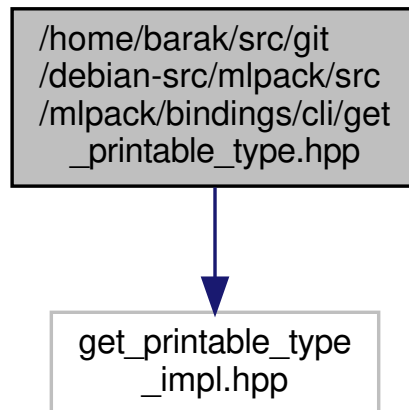
- **mlpack**
 .*.hpp*
- **mlpack::bindings**
- **mlpack::bindings::markdown**

Functions

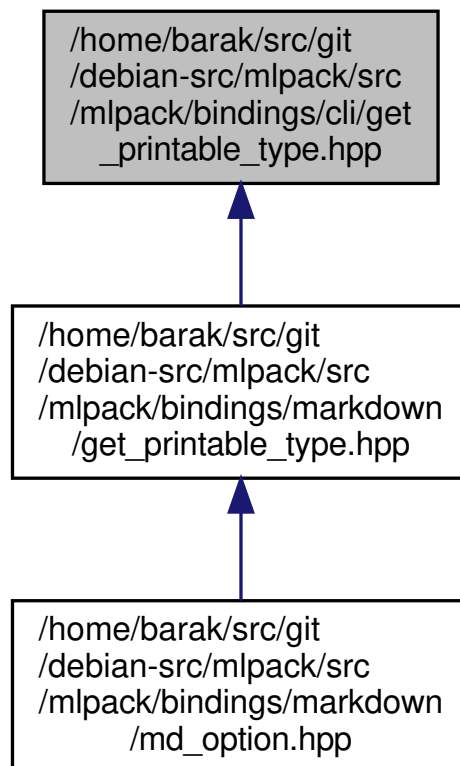
- `template<typename T >`
`std::string GetPrintableParamValue (const util::ParamData &data, const std::string &value, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)`
Get the parameter name for a type that has no special handling.
- `template<typename T >`
`std::string GetPrintableParamValue (const util::ParamData &data, const std::string &value, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)`
Get the parameter name for a matrix type (where the user has to pass the file that holds the matrix).
- `template<typename T >`
`std::string GetPrintableParamValue (const util::ParamData &data, const std::string &value, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)`
Get the parameter name for a serializable model type (where the user has to pass the file that holds the matrix).
- `template<typename T >`
`std::string GetPrintableParamValue (const util::ParamData &data, const std::string &value, const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)`
Get the parameter name for a mapped matrix type (where the user has to pass the file that holds the matrix).
- `template<typename T >`
`void GetPrintableParamValue (const util::ParamData &d, const void *input, void *output)`
Get the parameter's name as seen by the user.

40.147 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get_printable_type.hpp File Reference

Include dependency graph for get_printable_type.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**


```
graph TD; A["/home/barak/src/git  
/debian-src/mlpack/src  
/mlpack/bindings/markdown  
/get_printable_type.hpp"] --> B["/home/barak/src/git  
/debian-src/mlpack/src  
/mlpack/bindings/markdown  
/md_option.hpp"]
```

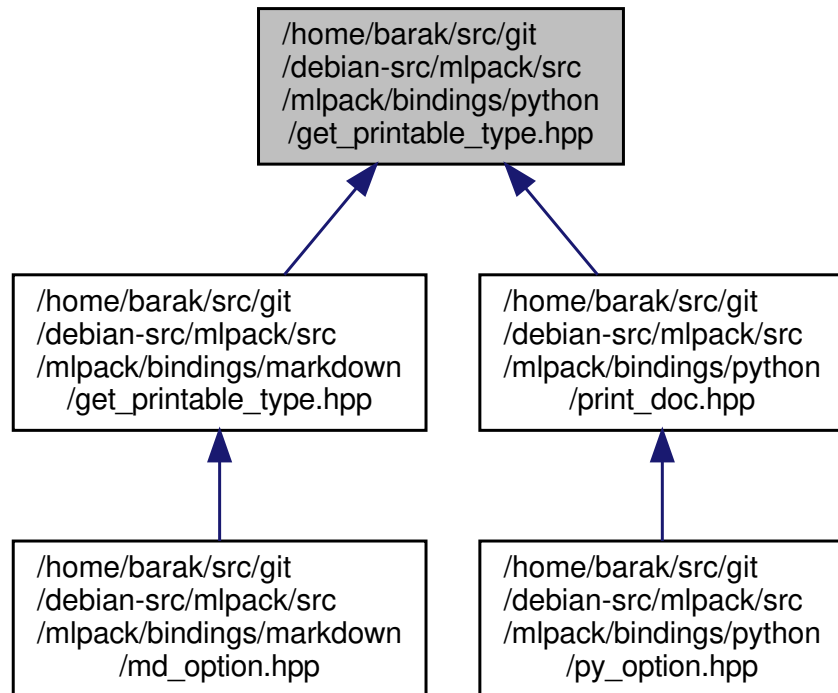
/home/barak/src/git
/debian-src/mlpack/src
/mlpack/bindings/markdown
/get_printable_type.hpp

/home/barak/src/git
/debian-src/mlpack/src
/mlpack/bindings/markdown
/md_option.hpp

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::markdown**

- `template<typename T >`
`void GetPrintableType (const util::ParamData &data, const void *, void *output)`
Print the type of a parameter into the output string.
- `template<typename T >`
`std::string GetPrintableType (const util::ParamData &data)`
Print the type of a parameter.

This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::python**

Functions

- `template<typename T >`
`std::string GetPrintableType (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< T >>::type !=0, const typename boost::disable_if< data::HasSerialize< T >>::type !=0, const typename boost::disable_if< arma::is_arma_type< T >>::type !=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type !=0)`
- `template<typename T >`
`std::string GetPrintableType (const util::ParamData &d, const typename boost::enable_if< util::IsStdVector< T >>::type !=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type !=0)`
- `template<typename T >`
`std::string GetPrintableType (const util::ParamData &, const typename boost::enable_if< arma::is_arma_type< T >>::type !=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type !=0)`
- `template<typename T >`
`std::string GetPrintableType (const util::ParamData &, const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type !=0)`

- template<typename T >
std::string **GetPrintableType** (const util::ParamData &d, const typename boost::disable_if< arma::is_arma_type< T >>::type !=0, const typename boost::enable_if< data::HasSerialize< T >>::type !=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type !=0)
- template<typename T >
void **GetPrintableType** (const util::ParamData &d, const void *, void *output)
- template<>
std::string **GetPrintableType**< **bool** > (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< bool >>::type *, const typename boost::disable_if< data::HasSerialize< bool >>::type *, const typename boost::disable_if< arma::is_arma_type< bool >>::type *, const typename boost::disable_if< std::is_same< bool, std::tuple< data::DatasetInfo, arma::mat >>>::type *)
- template<>
std::string **GetPrintableType**< **double** > (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< double >>::type *, const typename boost::disable_if< data::HasSerialize< double >>::type *, const typename boost::disable_if< arma::is_arma_type< double >>::type *, const typename boost::disable_if< std::is_same< double, std::tuple< data::DatasetInfo, arma::mat >>>::type *)
- template<>
std::string **GetPrintableType**< **int** > (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< int >>::type *, const typename boost::disable_if< data::HasSerialize< int >>::type *, const typename boost::disable_if< arma::is_arma_type< int >>::type *, const typename boost::disable_if< std::is_same< int, std::tuple< data::DatasetInfo, arma::mat >>>::type *)
- template<>
std::string **GetPrintableType**< **size_t** > (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< size_t >>::type *, const typename boost::disable_if< data::HasSerialize< size_t >>::type *, const typename boost::disable_if< arma::is_arma_type< size_t >>::type *, const typename boost::disable_if< std::is_same< size_t, std::tuple< data::DatasetInfo, arma::mat >>>::type *)
- template<>
std::string **GetPrintableType**< **std::string** > (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< std::string >>::type *, const typename boost::disable_if< data::HasSerialize< std::string >>::type *, const typename boost::disable_if< arma::is_arma_type< std::string >>::type *, const typename boost::disable_if< std::is_same< std::string, std::tuple< data::DatasetInfo, arma::mat >>>::type *)

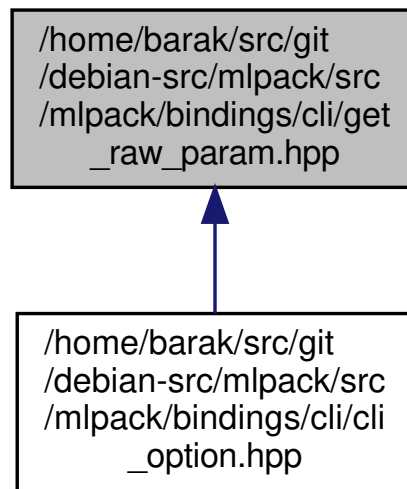
40.150 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get_raw_param.hpp

File Reference

Include dependency graph for get_raw_param.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .*.hpp*
- **mlpack::bindings**
- **mlpack::bindings::cli**

Functions

- `template<typename T >`
T & GetRawParam (util::ParamData &d, const typename boost::disable_if< arma::is_arma_type< T >>::type
 *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if<
 std::is_same< T, std::tuple< **mlpack::data::DatasetInfo**, arma::mat >>>::type *=0)
This overload is called when nothing special needs to happen to the name of the parameter.
- `template<typename T >`
T & GetRawParam (util::ParamData &d, const typename boost::enable_if_c< arma::is_arma_type< T ><←
 ::value||std::is_same< T, std::tuple< **mlpack::data::DatasetInfo**, arma::mat >>::value >>::type *=0)
Return a matrix parameter.
- `template<typename T >`
T *& GetRawParam (util::ParamData &d, const typename boost::disable_if< arma::is_arma_type< T >>::type
 *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)
Return the name of a model parameter.
- `template<typename T >`
void GetRawParam (const util::ParamData &d, const void *, void *output)
Return a parameter casted to the given type.

40.150.1 Detailed Description

Author

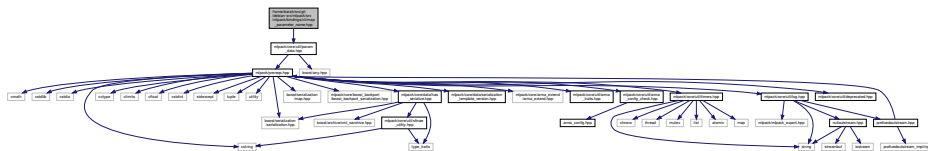
Ryan Curtin

Use template metaprogramming to get the right type of parameter, but without any processing.

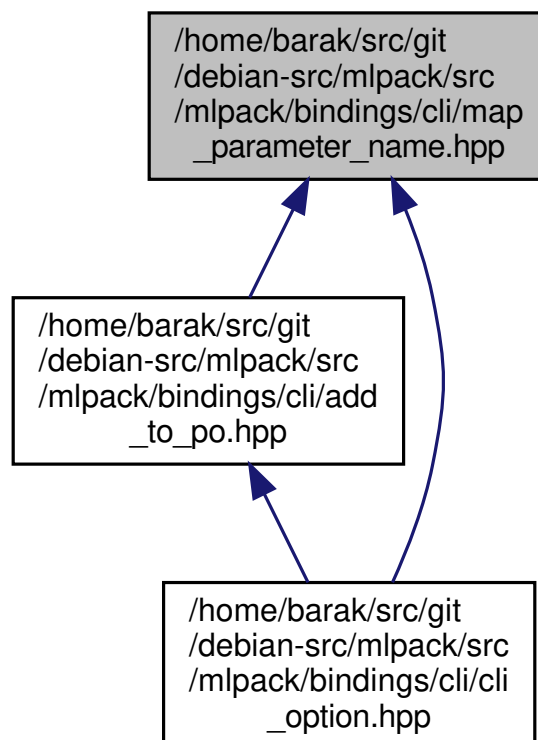
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.151 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/map_parameter_name.hpp File Reference

Include dependency graph for map_parameter_name.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::cli**

Functions

- `template<typename T >`
`std::string MapParameterName (const std::string &identifier, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< mlpack::data::DatasetInfo, arma::mat >>>::type *=0)`
If needed, map the parameter name to the name that is used by boost::program_options.
- `template<typename T >`
`std::string MapParameterName (const std::string &identifier, const typename boost::enable_if_c< arma::is_arma_type< T >::value||std::is_same< T, std::tuple< mlpack::data::DatasetInfo, arma::mat >>>::value||data::HasSerialize< T >::value >::type *=0)`
Map the parameter name to the name that is used by boost::program_options.
- `template<typename T >`
`void MapParameterName (const util::ParamData &d, const void *, void *output)`
Map the parameter name to the name seen by boost::program_options.

40.151.1 Detailed Description

Author

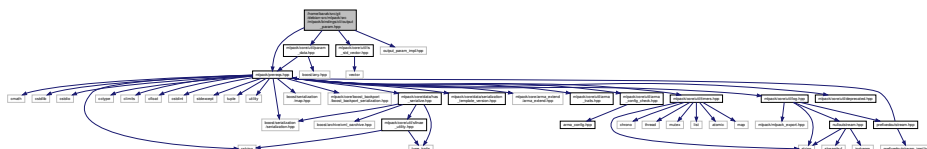
Ryan Curtin

Map a parameter name to what it seen by boost::program_options using template metaprogramming.

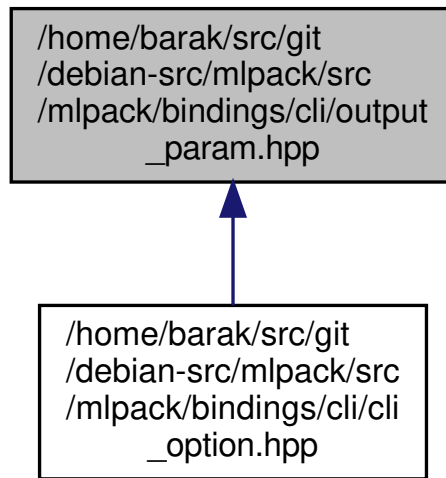
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.152 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/output_param.hpp File Reference

Include dependency graph for output_param.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::cli**

Functions

- template<typename T >
void **OutputParam** (const util::ParamData &data, const void *, void *)
 Output an option.
- template<typename T >
void **OutputParamImpl** (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< util::IsStdVector< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>::type *=0
 Output an option (print to stdout).
- template<typename T >
void **OutputParamImpl** (const util::ParamData &data, const typename boost::enable_if< util::IsStdVector< T >>::type *=0)
 Output a vector option (print to stdout).
- template<typename T >
void **OutputParamImpl** (const util::ParamData &data, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)
 Output a matrix option (this saves it to the given file).
- template<typename T >
void **OutputParamImpl** (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)
 Output a serializable class option (this saves it to the given file).

- `template<typename T >`
`void OutputParamImpl (const util::ParamData &data, const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>::type !=0)`
Output a mapped dataset.

40.152.1 Detailed Description

Author

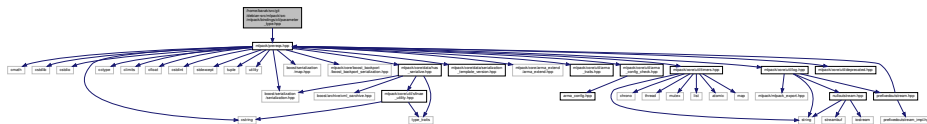
Ryan Curtin

Output a parameter of different types using template metaprogramming.

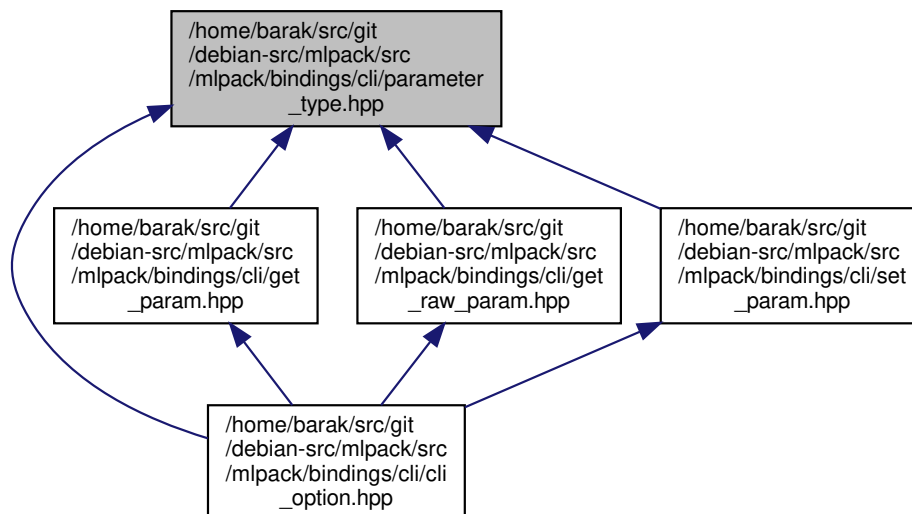
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.153 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/parameter_type.hpp File Reference

Include dependency graph for parameter_type.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct **ParameterType**< **T** >
Utility struct to return the type that boost::program_options should accept for a given input type.
- struct **ParameterType**< **arma::Col**< **eT** > >
For vector types, boost::program_options will accept a std::string, not an arma::Col<eT> (since it is not clear how to specify a vector on the command-line).
- struct **ParameterType**< **arma::Mat**< **eT** > >
For matrix types, boost::program_options will accept a std::string, not an arma::mat (since it is not clear how to specify a matrix on the command-line).
- struct **ParameterType**< **arma::Row**< **eT** > >
For row vector types, boost::program_options will accept a std::string, not an arma::Row<eT> (since it is not clear how to specify a vector on the command-line).
- struct **ParameterType**< **std::tuple**< **mlpack::data::DatasetMapper**< **PolicyType**, **std::string** >, **arma::Mat**< **eT** > > >
For matrix+dataset info types, we should accept a std::string.
- struct **ParameterTypeDeducer**< **HasSerialize**, **T** >
- struct **ParameterTypeDeducer**< **true**, **T** >

Namespaces

- **mlpack**
.hpp
- **mlpack::bindings**
- **mlpack::bindings::cli**

40.153.1 Detailed Description

Author

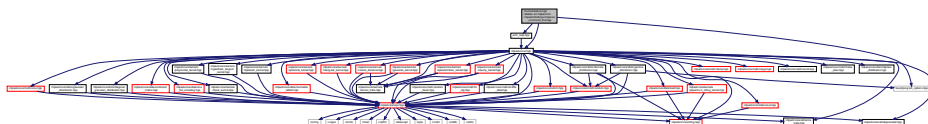
Ryan Curtin

Template metaprogramming structures to find the type (as seen by boost::program_options) of a particular option type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.154 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/parse_command_line.hpp File Reference

Include dependency graph for parse_command_line.hpp:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::cli**

Functions

- **PARAM_FLAG** ("help", "Default help info.", "h")
- **PARAM_FLAG** ("verbose", "Display informational messages and the full list of " "parameters and timers at the end of execution.", "v")
- **PARAM_FLAG** ("version", "Display the version of mlpack.", "V")
- **PARAM_STRING_IN** ("info", "Print help on a specific option.", "", "")
- void **ParseCommandLine** (int argc, char **argv)

*Parse the command line, setting all of the options inside of the **CLI** (p. 1117) object to their appropriate given values.*

40.154.1 Detailed Description

Author

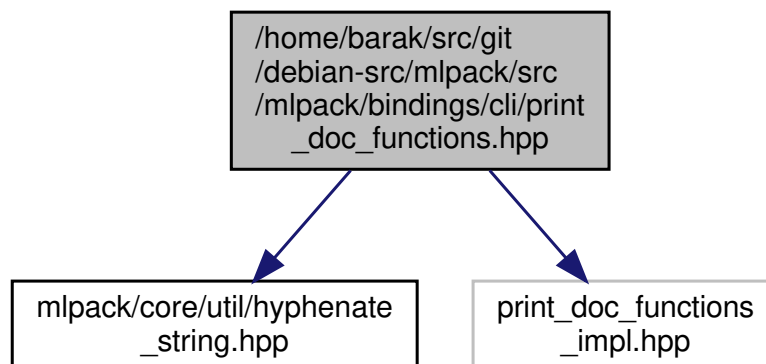
Ryan Curtin
Matthew Amidon

Parse the command line options.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.155 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/print_doc_functions.hpp File Reference

Include dependency graph for print_doc_functions.hpp:



Namespaces

- **mlpack**
.hpp
- **mlpack::bindings**
- **mlpack::bindings::cli**

Functions

- `std::string GetBindingName (const std::string &bindingName)`
Given the name of a binding, print its command-line name (this returns "mlpack_<bindingName>").
- `template<typename T >`
`bool IgnoreCheck (const T &)`
Return whether or not a runtime check on parameters should be ignored.
- `std::string ParamString (const std::string ¶mName)`
Print what a user would type to invoke the given option name.
- `std::string PrintDataset (const std::string &dataset)`
Print a dataset type parameter (add .csv and return).
- `std::string PrintDefault (const std::string ¶mName)`
Given a parameter name, print its corresponding default value.
- `std::string PrintImport (const std::string &bindingName)`
*Print any imports for **CLI** (p. 1117) (there are none, so this returns an empty string).*
- `std::string PrintModel (const std::string &model)`
Print a model type parameter (add .bin and return).
- `std::string PrintOutputOptionInfo ()`
Print any special information about output options.
- `std::string PrintType (const util::ParamData ¶m)`
Print the type of a parameter that a user would specify from the command-line.
- `std::string PrintTypeDocs ()`
Print documentation for each of the types.
- `template<typename T >`
`std::string PrintValue (const T &value, bool quotes)`
Given a parameter type, print the corresponding value.
- `std::string ProcessOptions ()`
Base case for recursion.
- `template<typename T , typename... Args>`
`std::string ProcessOptions (const std::string ¶mName, const T &value, Args... args)`
Print an option for a command-line argument.
- `template<typename... Args>`
`std::string ProgramCall (const std::string &programName, Args... args)`
Given a program name and arguments for it, print what its invocation would be.
- `std::string ProgramCall (const std::string &programName)`
Given a program name, print a program call invocation assuming that all options are specified.

40.156 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/print_doc_↵ functions.hpp File Reference

Include dependency graph for print_doc_functions.hpp:



Namespaces

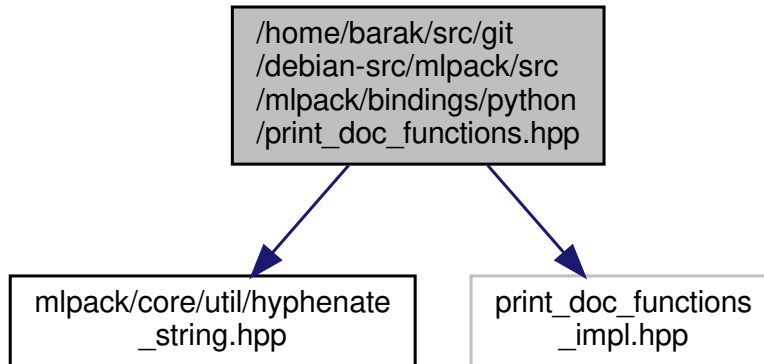
- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::markdown**

Functions

- std::string **GetBindingName** (const std::string &bindingName)
 *Given the name of the binding, print the name for the current language (as given by **BindingInfo** (p. 984)).*
- template<typename T >
 bool **IgnoreCheck** (const T &t)
 Return whether or not a runtime check on parameters should be ignored.
- std::string **ParamString** (const std::string ¶mName)
 Print what a user would type to invoke the given option name.
- std::string **ParamType** (const util::ParamData &d)
 Print the user-encountered type of an option.
- std::string **PrintDataset** (const std::string &dataset)
 Print a dataset type parameter (add .csv and return).
- std::string **PrintDefault** (const std::string ¶mName)
 Print the default value of an option, unless it is required (in which case Markdown italicized '—' is printed).
- std::string **PrintImport** (const std::string &bindingName)
 Print any imports that need to be done before using the binding.
- std::string **PrintLanguage** (const std::string &language)
 Print the name of the given language.
- std::string **PrintModel** (const std::string &model)
 Print a model type parameter (add .bin and return).
- std::string **PrintOutputOptionInfo** (const std::string &language)
 Print any special information about output options.
- std::string **PrintTypeDocs** ()
 Print details about the different types for a language.
- template<typename T >
 std::string **PrintValue** (const T &value, bool quotes)
 Given a parameter type, print the corresponding value.
- template<typename... Args>
 std::string **ProgramCall** (const std::string &programName, Args... args)
 Given a program name and arguments for it, print what its invocation would be.
- std::string **ProgramCall** (const std::string &programName)
 Given a program name, print a call assuming that all arguments are specified.

40.157 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/print_doc_functions.hpp File Reference

Include dependency graph for print_doc_functions.hpp:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::python**

Functions

- **std::string GetBindingName** (const std::string &bindingName)
 Given the name of a binding, print its Python name.
- **bool IgnoreCheck** (const std::string ¶mName)
 Print whether or not we should ignore a check on the given parameter.
- **bool IgnoreCheck** (const std::vector< std::string > &constraints)
 Print whether or not we should ignore a check on the given set of constraints.
- **bool IgnoreCheck** (const std::vector< std::pair< std::string, bool >> &constraints, const std::string ¶mName)
 Print whether or not we should ignore a check on the given set of constraints.
- **std::string ParamString** (const std::string ¶mName)
 Given the parameter name, determine what it would actually be when passed to the command line.
- **std::string PrintDataset** (const std::string &datasetName)
 Given the name of a matrix, print it.
- **std::string PrintDefault** (const std::string ¶mName)
 Given a parameter name, print its corresponding default value.
- **std::string PrintImport** (const std::string &bindingName)
 Print any import information for the Python binding.
- **std::string PrintInputOptions** ()

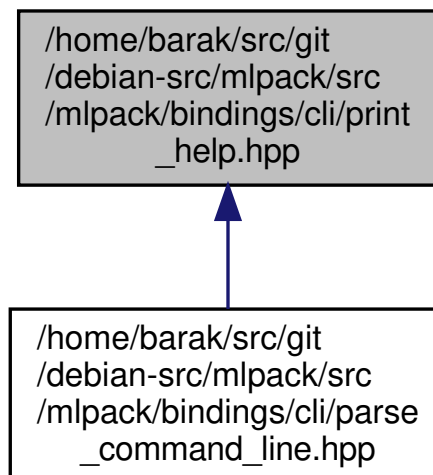
- `template<typename T , typename... Args>`
`std::string PrintInputOptions (const std::string ¶mName, const T &value, Args... args)`
Print an input option.
- `std::string PrintModel (const std::string &modelName)`
Given the name of a model, print it.
- `std::string PrintOutputOptionInfo ()`
Print any special information about output options.
- `std::string PrintOutputOptions ()`
- `template<typename T , typename... Args>`
`std::string PrintOutputOptions (const std::string ¶mName, const T &value, Args... args)`
- `template<typename T >`
`std::string PrintValue (const T &value, bool quotes)`
Given a parameter type, print the corresponding value.
- `template<>`
`std::string PrintValue (const bool &value, bool quotes)`
- `template<typename... Args>`
`std::string ProgramCall (const std::string &programName, Args... args)`
Given a name of a binding and a variable number of arguments (and their contents), print the corresponding function call.
- `std::string ProgramCall (const std::string &programName)`
Given the name of a binding, print a program call assuming that all options are specified.

40.158 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/print_help.hpp File Reference

Include dependency graph for `print_help.hpp`:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::cli**

Functions

- void **PrintHelp** (const std::string ¶m="")
 Print the help for the given parameter.

40.158.1 Detailed Description

Author

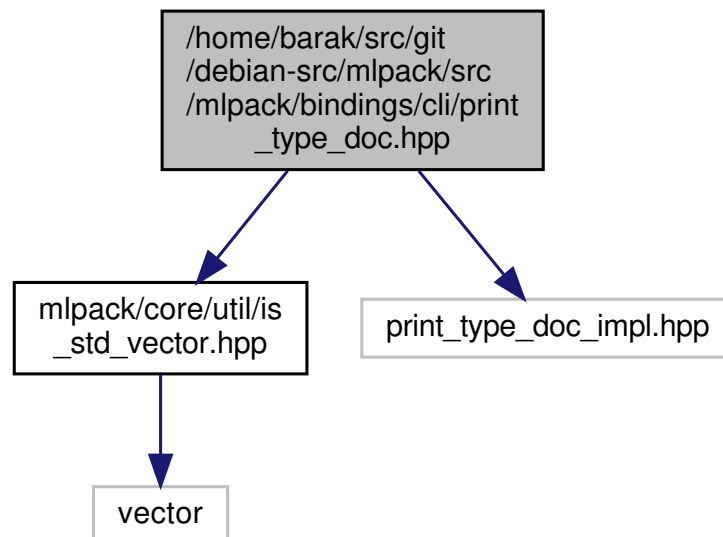
Ryan Curtin

Print help for a command-line program.

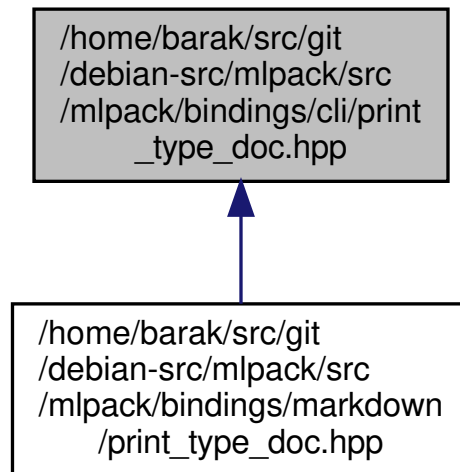
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.159 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/print_type_doc.hpp File Reference

Include dependency graph for print_type_doc.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::cli**

Functions

- template<typename T >
std::string **PrintTypeDoc** (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< util::IsStdVector< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)

 Return a string representing the command-line type of an option.
- template<typename T >
std::string **PrintTypeDoc** (const util::ParamData &data, const typename std::enable_if< util::IsStdVector< T ><::value >::type *=0)

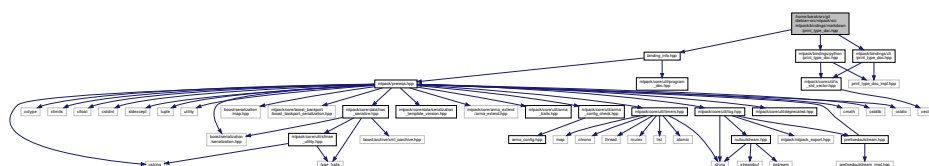
 Return a string representing the command-line type of a vector.
- template<typename T >
std::string **PrintTypeDoc** (const util::ParamData &data, const typename std::enable_if< arma::is_arma_type< T ><::value >::type *=0)

 Return a string representing the command-line type of a matrix option.
- template<typename T >
std::string **PrintTypeDoc** (const util::ParamData &data, const typename std::enable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::value >::type *=0)

 Return a string representing the command-line type of a matrix tuple option.
- template<typename T >
std::string **PrintTypeDoc** (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)

- `template<typename T>`
`void PrintTypeDoc (const util::ParamData &data, const void *, void *output)`
Print the command-line type of an option into a string.

Include dependency graph for print_type_doc.hpp:

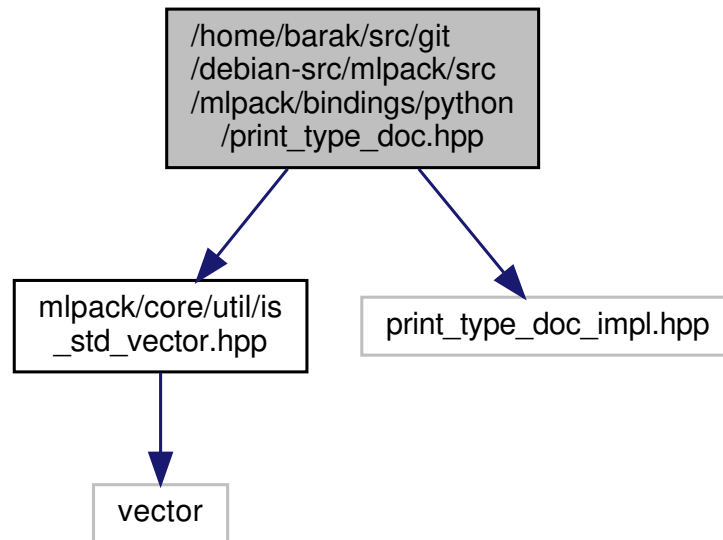


- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::markdown**

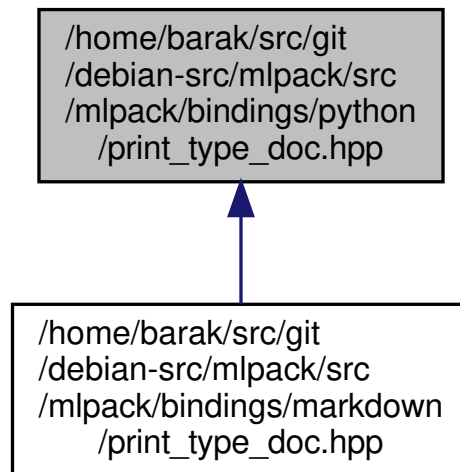
- `template<typename T>`
`std::string PrintTypeDoc (const util::ParamData &data)`
Print the type of a parameter into the output string.

40.161 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/print_type_↔ doc.hpp File Reference

Include dependency graph for print_type_doc.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::python**

Functions

- template<typename T >
 std::string **PrintTypeDoc** (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< util::IsStdVector< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)

 Return a string representing the command-line type of an option.
- template<typename T >
 std::string **PrintTypeDoc** (const util::ParamData &data, const typename std::enable_if< util::IsStdVector< T >>::value >::type *=0)

 Return a string representing the command-line type of a vector.
- template<typename T >
 std::string **PrintTypeDoc** (const util::ParamData &data, const typename std::enable_if< arma::is_arma_type< T >>::value >::type *=0)

 Return a string representing the command-line type of a matrix option.
- template<typename T >
 std::string **PrintTypeDoc** (const util::ParamData &data, const typename std::enable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::value >::type *=0)

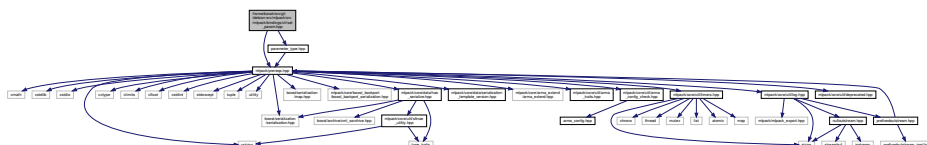
 Return a string representing the command-line type of a matrix tuple option.
- template<typename T >
 std::string **PrintTypeDoc** (const util::ParamData &data, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)

 Return a string representing the command-line type of a model.
- template<typename T >
 void **PrintTypeDoc** (const util::ParamData &data, const void *, void *output)

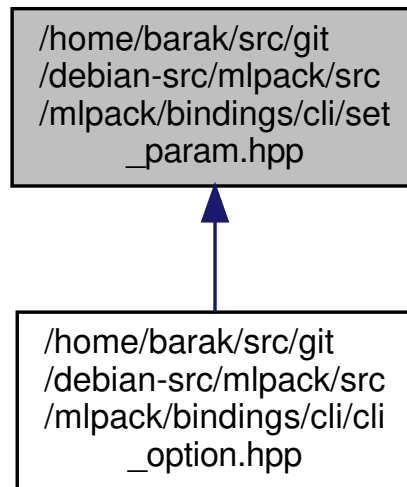
 Print the command-line type of an option into a string.

40.162 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/set_param.hpp File Reference

Include dependency graph for set_param.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

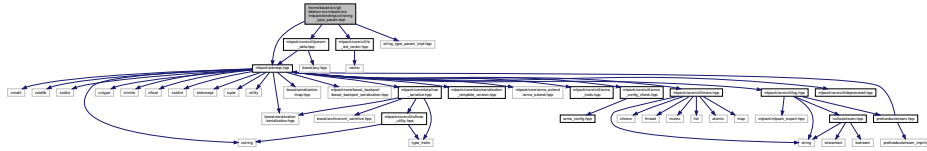
- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::cli**

Functions

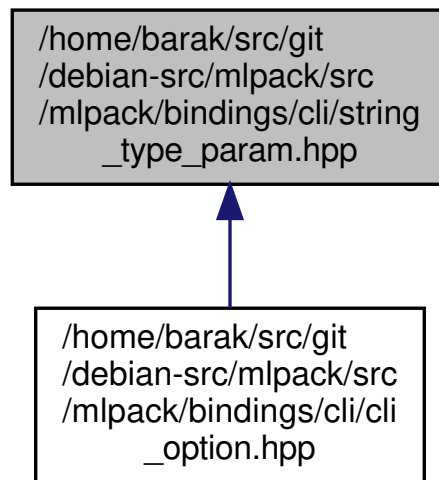
- `template<typename T >`
`void SetParam (util::ParamData &d, const boost::any &value, const typename boost::disable_if< arma::is_↵`
`arma_type< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const`
`typename boost::disable_if< std::is_same< T, std::tuple< mlpack::data::DatasetInfo, arma::mat >>>::type`
`*=0, const typename boost::disable_if< std::is_same< T, bool >>::type *=0)`
This overload is called when nothing special needs to happen to the name of the parameter.
- `template<typename T >`
`void SetParam (util::ParamData &d, const boost::any &, const typename boost::enable_if< std::is_same< T,`
`bool >>::type *=0)`
This overload is called to set a boolean.
- `template<typename T >`
`void SetParam (util::ParamData &d, const boost::any &value, const typename std::enable_if< arma::is_arma↵`
`_type< T >::value||std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>::value >::type *=0)`
Set a matrix parameter, a matrix/dataset info parameter, or a serializable object.
- `template<typename T >`
`void SetParam (util::ParamData &d, const boost::any &value, const typename boost::disable_if< arma::is_↵`
`arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)`
Set a serializable object.
- `template<typename T >`
`void SetParam (const util::ParamData &d, const void *input, void *)`
Return a parameter casted to the given type.

40.163 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/string_type_param.hpp File Reference

Include dependency graph for string_type_param.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::cli**

Functions

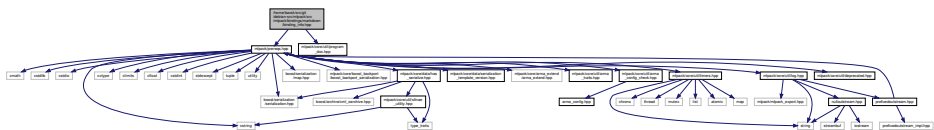
- `template<typename T >`
 void **StringTypeParam** (const util::ParamData &, const void *, void *output)
 Return a string containing the type of a parameter.
- `template<>`
 void **StringTypeParam**< **bool** > (const util::ParamData &, const void *, void *output)
 Return "bool".
- `template<>`
 void **StringTypeParam**< **double** > (const util::ParamData &, const void *, void *output)

- `template<>`
`void StringTypeParam< int > (const util::ParamData &, const void *, void *output)`
Return "int".
- `template<>`
`void StringTypeParam< std::string > (const util::ParamData &, const void *, void *output)`
Return "string".
- `template<>`
`void StringTypeParam< std::tuple< mlpack::data::DatasetInfo, arma::mat > > (const util::ParamData &, const void *, void *output)`
Return "string";.
- `template<typename T >`
`std::string StringTypeParamImpl (const typename boost::disable_if< util::IsStdVector< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0)`
Return a string containing the type of the parameter.
- `template<typename T >`
`std::string StringTypeParamImpl (const typename boost::enable_if< util::IsStdVector< T >>::type *=0)`
Return a string containing the type of the parameter, for vector options.
- `template<typename T >`
`std::string StringTypeParamImpl (const typename boost::enable_if< data::HasSerialize< T >>::type *=0)`
Return a string containing the type of the parameter,.

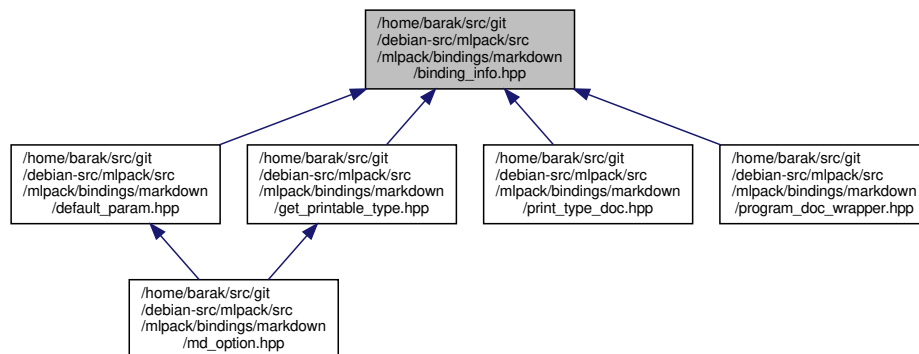
Author
Ryan Curtin

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Include dependency graph for binding_info.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **BindingInfo**

The ***BindingInfo*** (p. 984) class is used by the Markdown documentation generator to store multiple *ProgramDoc* objects, indexed by both the binding name (i.e.

Namespaces

- mlpack**
.hpp
- mlpack::bindings**
- mlpack::bindings::markdown**

40.164.1 Detailed Description

Author

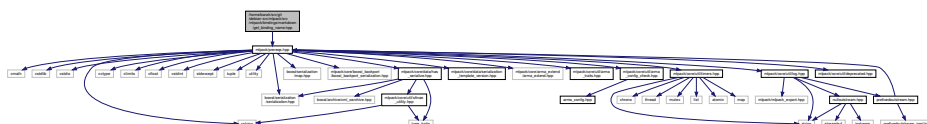
Ryan Curtin

This file defines the `BindingInfo` singleton class that is used specifically for the Markdown bindings to map from a binding name (i.e. "knn") to multiple `ProgramDoc` objects, which are then used to generate the documentation.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.165 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/get_binding_name.hpp File Reference

Include dependency graph for `get_binding_name.hpp`:



Namespaces

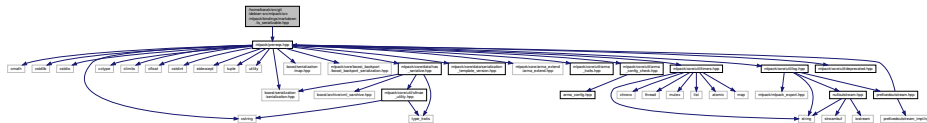
- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::markdown**

Functions

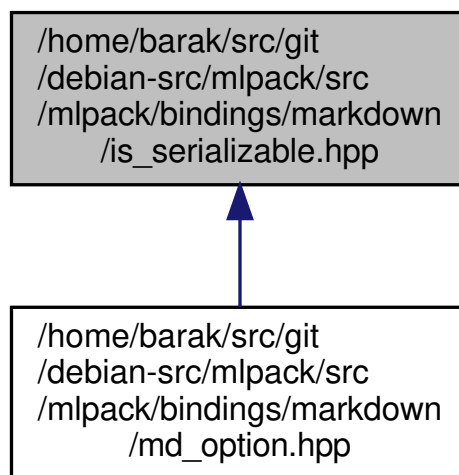
- std::string **GetBindingName** (const std::string &language, const std::string &name)
 Given a language name and a binding name, return the name of that binding for that language.

40.166 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/is_serializable.hpp File Reference

Include dependency graph for is_serializable.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::markdown**

Functions

- template<typename T >
bool **IsSerializable** (const typename boost::disable_if< data::HasSerialize< T >>::type *=0)
Return false, because the type is not serializable.
- template<typename T >
bool **IsSerializable** (const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)
Return false, because even though the type is serializable, it is an Armadillo type not an mlpack model.
- template<typename T >
bool **IsSerializable** (const typename boost::enable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0)
Return true, because the type is serializable.
- template<typename T >
void **IsSerializable** (const util::ParamData &, const void *, void *output)
Return whether or not the type is serializable.

40.166.1 Detailed Description

Author

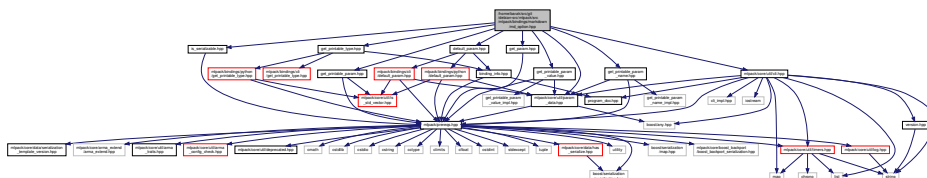
Ryan Curtin

Return a bool noting whether or not a parameter is serializable.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.167 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/markdown/md_option.hpp File Reference

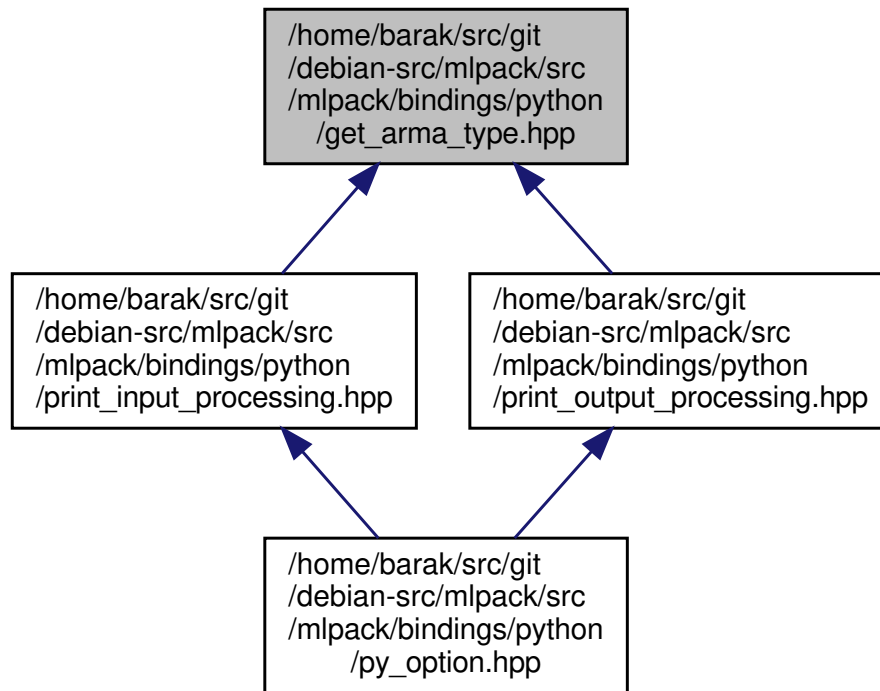
Include dependency graph for md_option.hpp:



Classes

- class **MDOption**< T >
The Markdown option class.

This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::python**

Functions

- `template<typename T >
std::string GetArmaType ()`
 This is used for arma::Mat<> types; it will return "mat" for matrices, "row" for row vectors, and "col" for column vectors.

40.170.1 Detailed Description

Author

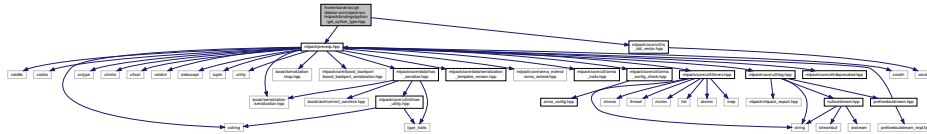
Ryan Curtin

Return "mat", "col", or "row" depending on the type of the given Armadillo object. This is so that the correct overload of `arma_numpy.numpy_to_<type>()` can be called.

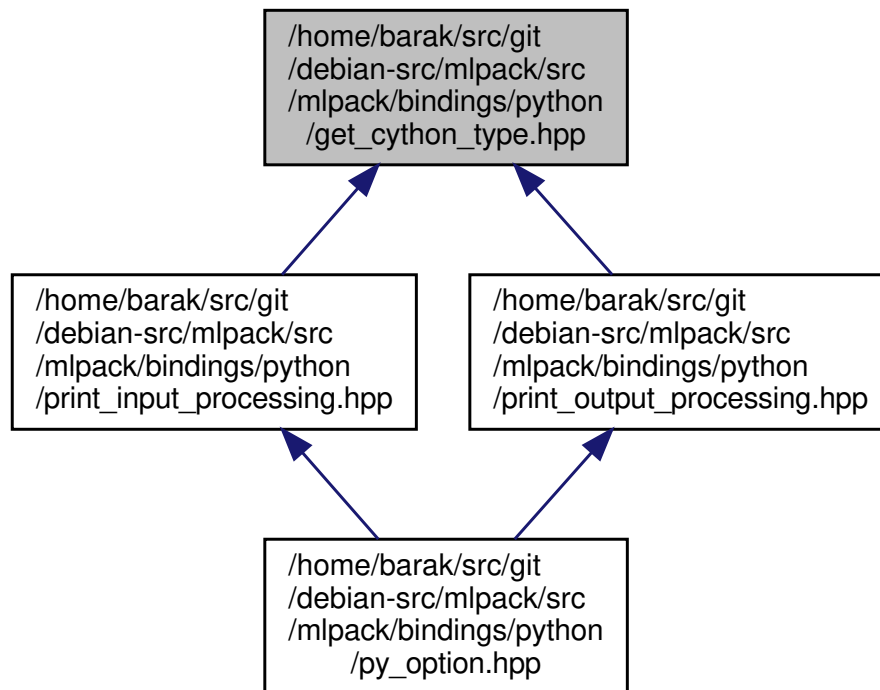
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.171 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/get_cython_type.hpp File Reference

Include dependency graph for get_cython_type.hpp:



This graph shows which files directly or indirectly include this file:



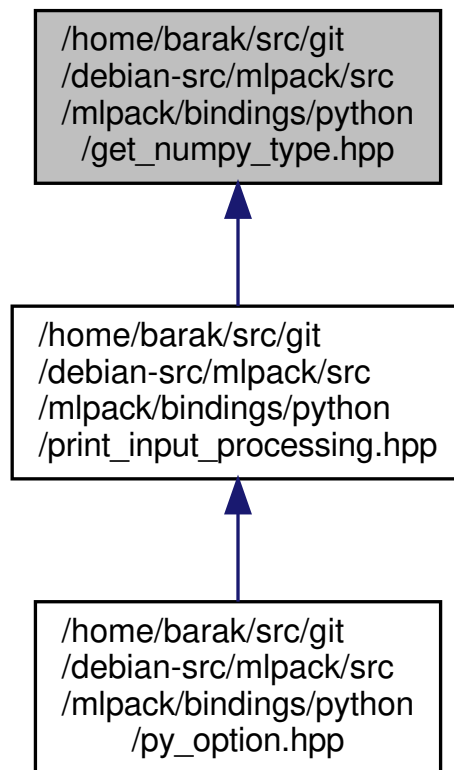
Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::python**

Functions

- `template<typename T>`
`std::string GetCythonType (const util::ParamData &, const typename boost::disable_if< util::IsStdVector< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0)`

This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::python**

Functions

- `template<typename T >`
 `std::string GetNumpyType ()`
- `template<>`
 `std::string GetNumpyType< double > ()`
- `template<>`
 `std::string GetNumpyType< size_t > ()`

40.172.1 Detailed Description

Author

Ryan Curtin

Given a C++ type, return the Python numpy dtype associated with that type.

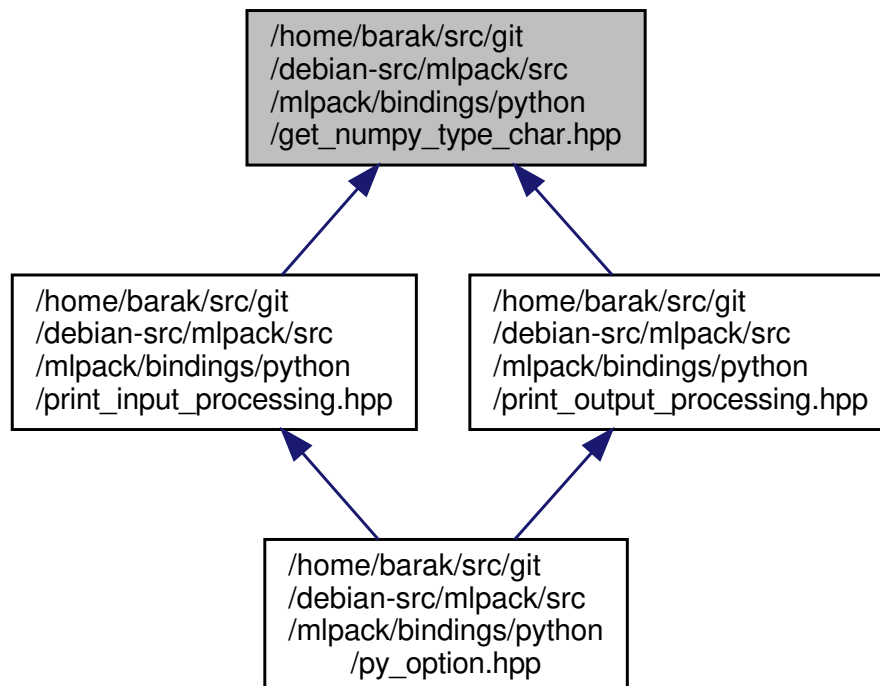
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.173 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/get_numpy_type_char.hpp File Reference

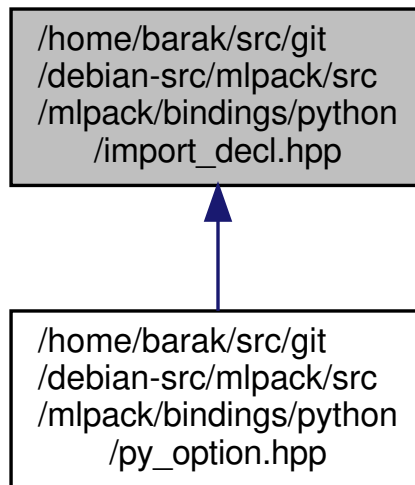
Include dependency graph for get_numpy_type_char.hpp:



This graph shows which files directly or indirectly include this file:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::python**

Functions

- template<typename T >
void **ImportDecl** (const util::ParamData &d, const size_t indent, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)
For a serializable type, print a cppclass definition.
- template<typename T >
void **ImportDecl** (const util::ParamData &, const size_t, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)
For a non-serializable type, print nothing.
- template<typename T >
void **ImportDecl** (const util::ParamData &, const size_t, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)
For a matrix type, print nothing.
- template<typename T >
void **ImportDecl** (const util::ParamData &d, const void *indent, void *)
Print the cppclass definition for a serializable model; print nothing for a non-serializable type.

40.174.1 Detailed Description

Author

Ryan Curtin

For a serializable model, print the class import.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.175 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/mlpack/arma_util.hpp File Reference

Include dependency graph for arma_util.hpp:



Functions

- `template<typename T >`
`T::elem_type * GetMemory (T &m)`
Return the matrix's allocated memory pointer, unless the matrix is using its internal preallocated memory, in which case we copy that and return a pointer to the memory we just made.
- `template<typename T >`
`size_t GetMemState (T &t)`
Get the memory state of the given Armadillo object.
- `template<typename T >`
`void SetMemState (T &t, int state)`
Set the memory state of the given Armadillo object.

40.175.1 Detailed Description

Author

Ryan Curtin

Utility function for Cython to set the memory state of an Armadillo object.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.175.2 Function Documentation

40.175.2.1 GetMemory()

```
T::elem_type* GetMemory (
    T & m ) [inline]
```

Return the matrix's allocated memory pointer, unless the matrix is using its internal preallocated memory, in which case we copy that and return a pointer to the memory we just made.

Definition at line 47 of file arma_util.hpp.

40.175.2.2 GetMemState()

```
size_t GetMemState (
    T & t )
```

Get the memory state of the given Armadillo object.

Definition at line 31 of file arma_util.hpp.

40.175.2.3 SetMemState()

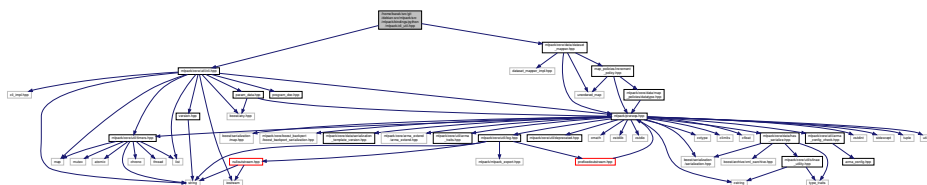
```
void SetMemState (
    T & t,
    int state )
```

Set the memory state of the given Armadillo object.

Definition at line 22 of file arma_util.hpp.

40.176 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/mlpack/cli_↵
util.hpp File Reference

Include dependency graph for cli_util.hpp:



Namespaces

- **mlpack**
 .hpp
- **mlpack::util**

Functions

- void **DisableBacktrace** ()
 Disable backtraces.
- void **DisableVerbose** ()
 Turn verbose output off.
- void **EnableTimers** ()
 Enable timing.
- void **EnableVerbose** ()
 Turn verbose output on.
- template<typename T >
 T * **GetParamPtr** (const std::string ¶mName)
 Return a pointer.
- template<typename T >
 T & **GetParamWithInfo** (const std::string ¶mName)
 Return the matrix part of a matrix + dataset info parameter.
- void **ResetTimers** ()
 Reset the status of all timers.
- template<typename T >
 void **SetParam** (const std::string &identifier, T &value)
 Set the parameter to the given value.
- template<typename T >
 void **SetParamPtr** (const std::string &identifier, T *value, const bool copy)
 Set the parameter to the given value, given that the type is a pointer.
- template<typename T >
 void **SetParamWithInfo** (const std::string &identifier, T &matrix, const bool *dims)
 Set the parameter (which is a matrix/DatasetInfo tuple) to the given value.

40.176.1 Detailed Description

Author

Ryan Curtin

Simple function to work around Cython's lack of support for lvalue references.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.177 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/mlpack/serialization.hpp File Reference

Include dependency graph for serialization.hpp:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::python**

Functions

- `template<typename T >`
 void **SerializeIn** (T *t, const std::string &str, const std::string &name)
- `template<typename T >`
 std::string **SerializeOut** (T *t, const std::string &name)

40.178 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/serialization.hpp File Reference

Include dependency graph for serialization.hpp:



Namespaces

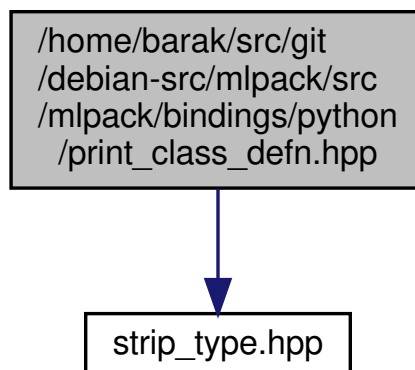
- **mlpack**
 .hpp

Functions

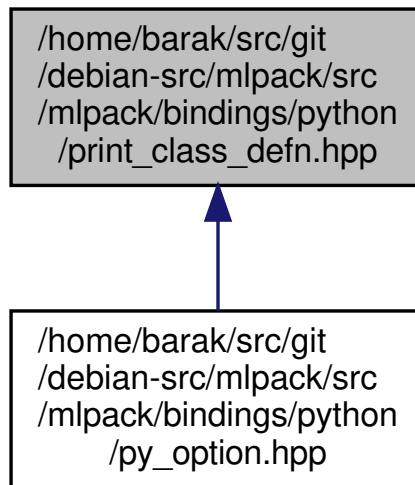
- void **CheckMatrices** (const arma::mat &x, const arma::mat &xmlX, const arma::mat &textX, const arma::mat &binaryX)
- void **CheckMatrices** (const arma::Mat< size_t > &x, const arma::Mat< size_t > &xmlX, const arma::Mat< size_t > &textX, const arma::Mat< size_t > &binaryX)
- void **CheckMatrices** (const arma::cube &x, const arma::cube &xmlX, const arma::cube &textX, const arma::cube &binaryX)
- template<typename T , typename IArchiveType , typename OArchiveType >
void **SerializeObject** (T &t, T &newT)
- template<typename T >
void **SerializeObjectAll** (T &t, T &xmlT, T &textT, T &binaryT)
- template<typename T , typename IArchiveType , typename OArchiveType >
void **SerializePointerObject** (T *t, T *&newT)
- template<typename T >
void **SerializePointerObjectAll** (T *t, T *&xmlT, T *&textT, T *&binaryT)
- template<typename CubeType >
void **TestAllArmadilloSerialization** (arma::Cube< CubeType > &x)
- template<typename MatType >
void **TestAllArmadilloSerialization** (MatType &x)
- template<typename CubeType , typename IArchiveType , typename OArchiveType >
void **TestArmadilloSerialization** (arma::Cube< CubeType > &x)
- template<typename MatType , typename IArchiveType , typename OArchiveType >
void **TestArmadilloSerialization** (MatType &x)

40.179 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/print_class_↵ defn.hpp File Reference

Include dependency graph for print_class_defn.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::python**

Functions

- template<typename T >
void **PrintClassDefn** (const util::ParamData &, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0)

Non-serializable models don't require any special definitions, so this prints nothing.
- template<typename T >
void **PrintClassDefn** (const util::ParamData &, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)

Matrices don't require any special definitions, so this prints nothing.
- template<typename T >
void **PrintClassDefn** (const util::ParamData &d, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)

Serializable models require a special class definition.
- template<typename T >
void **PrintClassDefn** (const util::ParamData &d, const void *, void *)

Print the class definition to stdout.

40.179.1 Detailed Description

Author

Ryan Curtin

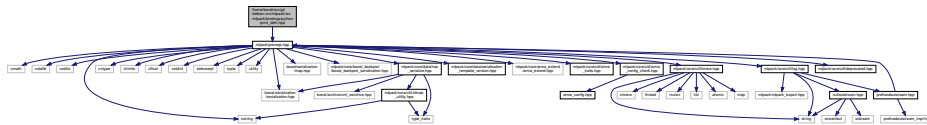
Print the class definition for generating a .pyx binding.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

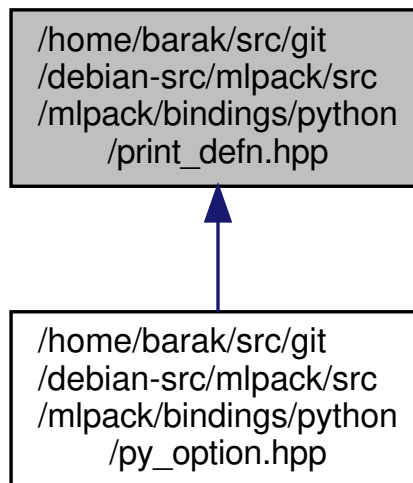
40.180 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/print_defn.hpp

File Reference

Include dependency graph for print_defn.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::python**

Functions

- `template<typename T >`
`void PrintDefn (const util::ParamData &d, const void *, void *)`
Print the definition for a Python binding parameter to stdout.

40.180.1 Detailed Description

Author

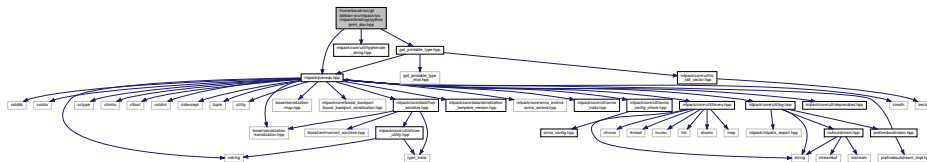
Ryan Curtin

Print the definition of a Python parameter.

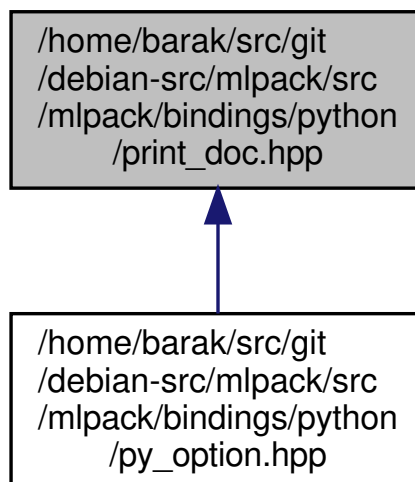
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.181 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/print_doc.hpp File Reference

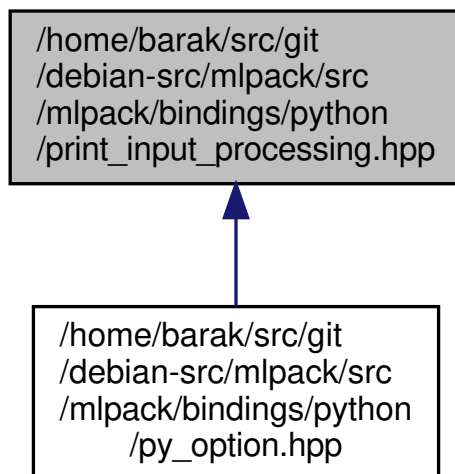
Include dependency graph for `print_doc.hpp`:



This graph shows which files directly or indirectly include this file:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::python**

Functions

- `template<typename T >`
void PrintInputProcessing (const util::ParamData &d, const size_t indent, const typename boost::disable_if< util::IsStdVector< T >>::type *=0, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)
Print input processing for a standard option type.
- `template<typename T >`
void PrintInputProcessing (const util::ParamData &d, const size_t indent, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0, const typename boost::enable_if< util::IsStdVector< T >>::type *=0)
Print input processing for a vector type.
- `template<typename T >`
void PrintInputProcessing (const util::ParamData &d, const size_t indent, const typename boost::disable_if< util::IsStdVector< T >>::type *=0, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)
Print input processing for a matrix type.
- `template<typename T >`
void PrintInputProcessing (const util::ParamData &d, const size_t indent, const typename boost::disable_if< util::IsStdVector< T >>::type *=0, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)
Print input processing for a serializable type.

Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::python**

Functions

- template<typename T >
void **PrintOutputProcessing** (const util::ParamData &d, const size_t indent, const bool onlyOutput, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::disable_if< data::HasSerialize< T >>::type *=0, const typename boost::disable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)

 Print output processing for a regular parameter type.
- template<typename T >
void **PrintOutputProcessing** (const util::ParamData &d, const size_t indent, const bool onlyOutput, const typename boost::enable_if< arma::is_arma_type< T >>::type *=0)

 Print output processing for a matrix type.
- template<typename T >
void **PrintOutputProcessing** (const util::ParamData &d, const size_t indent, const bool onlyOutput, const typename boost::enable_if< std::is_same< T, std::tuple< data::DatasetInfo, arma::mat >>>::type *=0)

 Print output processing for a dataset info / matrix combination.
- template<typename T >
void **PrintOutputProcessing** (const util::ParamData &d, const size_t indent, const bool onlyOutput, const typename boost::disable_if< arma::is_arma_type< T >>::type *=0, const typename boost::enable_if< data::HasSerialize< T >>::type *=0)

 Print output processing for a serializable model.
- template<typename T >
void **PrintOutputProcessing** (const util::ParamData &d, const void *input, void *)

 Given parameter information and the current number of spaces for indentation, print the code to process the output to cout.

40.183.1 Detailed Description

Author

Ryan Curtin

Print the output processing in a Python binding .pyx file for a given parameter.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.184 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/print_pyx.hpp

File Reference

Include dependency graph for print_pyx.hpp:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::python**

Functions

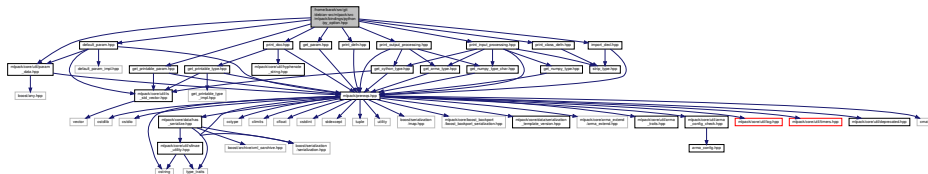
- void **PrintPYX** (const util::ProgramDoc &programInfo, const std::string &mainFilename, const std::string &functionName)

Given a list of parameter definition and program documentation, print a generated .pyx file to stdout.

40.185 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/py_option.hpp

File Reference

Include dependency graph for py_option.hpp:



Classes

- class **PyOption**< T >
 The Python option class.

Namespaces

- **mlpack**
.hpp
- **mlpack::bindings**
- **mlpack::bindings::python**

Variables

- `std::string` **programName**

40.185.1 Detailed Description

Author

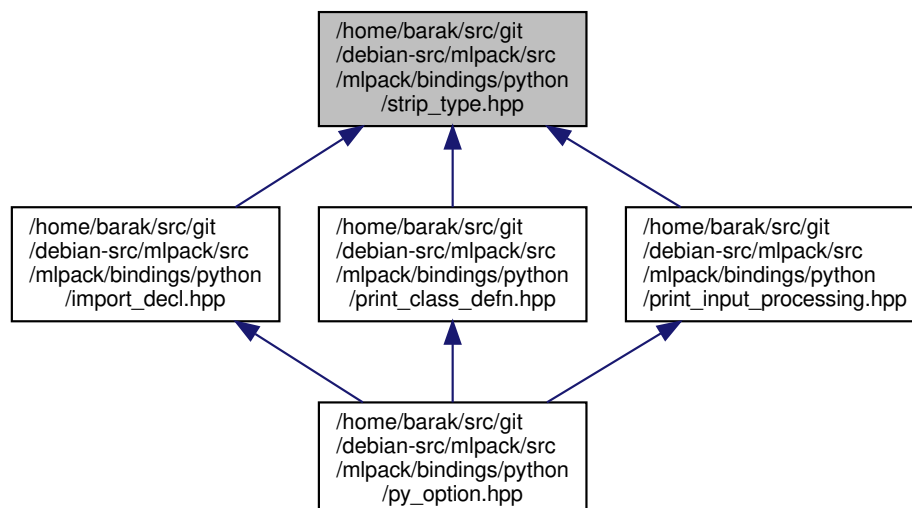
Ryan Curtin

The Python option type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.186 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/strip_type.hpp File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::python**

Functions

- void **StripType** (const std::string &inputType, std::string &strippedType, std::string &printedType, std::string &defaultsType)
 Given an input type like, e.g., "LogisticRegression<>", return three types that can be used in Python code.

40.186.1 Detailed Description

Author

Ryan Curtin

Given a C++ typename that may have template parameters, return stripped and printable versions to be used in Python bindings.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.187 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/clean_memory.hpp File Reference

Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::tests**

Functions

- void **CleanMemory** ()
 *Delete any unique pointers that are held by the **CLI** (p. 1117) object.*

40.187.1 Detailed Description

Author

Ryan Curtin

Delete any unique pointers that are held by the CLI object. This is similar to the code in **end_program.hpp** (p. 2462).

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.188 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/ignore_check.hpp File Reference

Namespaces

- **mlpack**
 .hpp
- **mlpack::bindings**
- **mlpack::bindings::tests**

Functions

- `template<typename T >`
 bool IgnoreCheck (const T &)
 Return whether or not a parameter check should be ignored.

40.188.1 Detailed Description

Author

Ryan Curtin

Implementation of **IgnoreCheck()** (p. 364) for Python bindings.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.190 /home/barak/src/git/debian-src/mlpack/src/mlpack/core.hpp File Reference

Include all of the base components required to write mlpack methods, and the main mlpack Doxygen documentation.

Include dependency graph for core.hpp:



This graph shows which files directly or indirectly include this file:



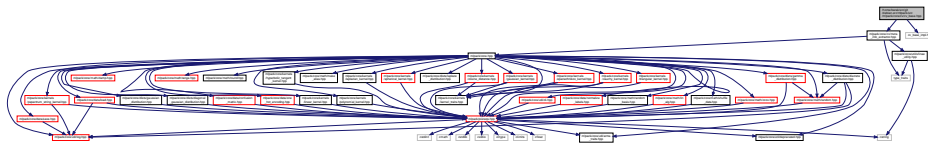
40.190.1 Detailed Description

Include all of the base components required to write mlpack methods, and the main mlpack Doxygen documentation.

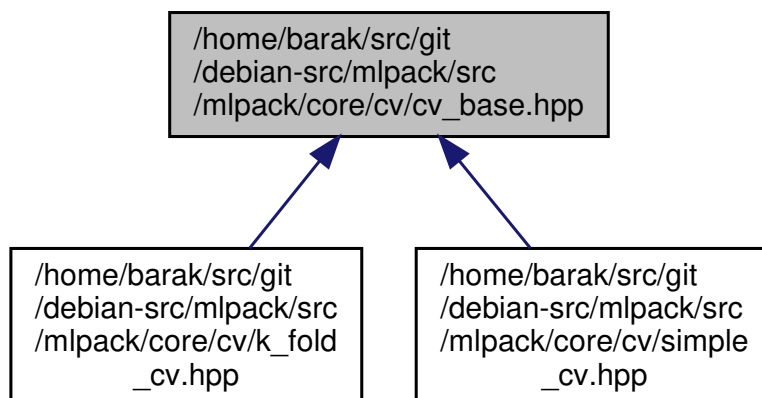
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.191 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/cv_base.hpp File Reference

Include dependency graph for cv_base.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **CVBase**< **MLAlgorithm**, **MatType**, **PredictionsType**, **WeightsType** >

An auxiliary class for cross-validation.

Namespaces

- **mlpack**
 .hpp
- **mlpack::cv**

40.191.1 Detailed Description

Author

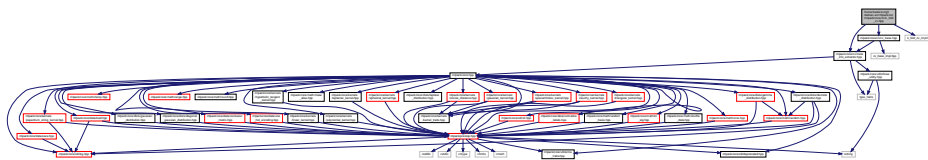
Kirill Mishchenko

A base class for cross-validation.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.192 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/k_fold_cv.hpp File Reference

Include dependency graph for k_fold_cv.hpp:



Classes

- class **KFoldCV**< **MLAlgorithm**, **Metric**, **MatType**, **PredictionsType**, **WeightsType** >

*The class **KFoldCV** (p. 1134) implements k-fold cross-validation for regression and classification algorithms.*

Namespaces

- **mlpack**
 .hpp
- **mlpack::cv**

40.192.1 Detailed Description

Author

Kirill Mishchenko

k-fold cross-validation.

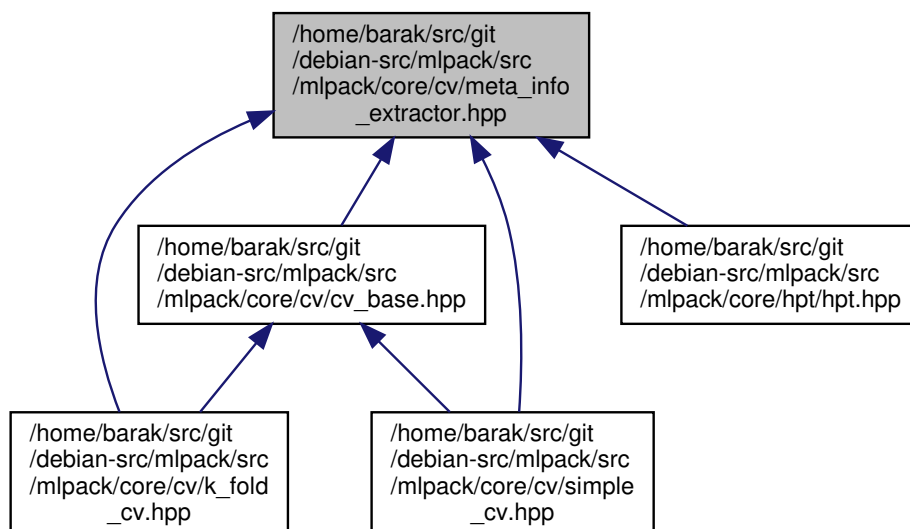
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.193 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/meta_info_extractor.hpp File Reference

Include dependency graph for meta_info_extractor.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **MetaInfoExtractor**< **MLAlgorithm**, **MT**, **PT**, **WT** >
MetaInfoExtractor (p. 1140) is a tool for extracting meta information about a given machine learning algorithm.
- struct **NotFoundMethodForm**
- struct **SelectMethodForm**< **MLAlgorithm**, **HMFs** >
A type function that selects a right method form.
- struct **SelectMethodForm**< **MLAlgorithm** >
- struct **SelectMethodForm**< **MLAlgorithm** >::From< **Forms** >
- struct **SelectMethodForm**< **MLAlgorithm**, **HasMethodForm**, **HMFs...** >
- class **SelectMethodForm**< **MLAlgorithm**, **HasMethodForm**, **HMFs...** >::From< **Forms** >
- struct **TrainForm**< **MatType**, **PredictionsType**, **WeightsType**, **DatasetInfo**, **NumClasses** >
A wrapper struct for holding a Train form.
- struct **TrainForm**< **MT**, **PT**, **void**, **false**, **false** >
- struct **TrainForm**< **MT**, **PT**, **void**, **false**, **true** >
- struct **TrainForm**< **MT**, **PT**, **void**, **true**, **false** >
- struct **TrainForm**< **MT**, **PT**, **void**, **true**, **true** >
- struct **TrainForm**< **MT**, **PT**, **WT**, **false**, **false** >
- struct **TrainForm**< **MT**, **PT**, **WT**, **false**, **true** >
- struct **TrainForm**< **MT**, **PT**, **WT**, **true**, **false** >
- struct **TrainForm**< **MT**, **PT**, **WT**, **true**, **true** >
- struct **TrainFormBase4**< **PT**, **WT**, **T1**, **T2** >
- struct **TrainFormBase5**< **PT**, **WT**, **T1**, **T2**, **T3** >
- struct **TrainFormBase6**< **PT**, **WT**, **T1**, **T2**, **T3**, **T4** >
- struct **TrainFormBase7**< **PT**, **WT**, **T1**, **T2**, **T3**, **T4**, **T5** >

Namespaces

- **mlpack**
.hpp
- **mlpack::cv**

40.193.1 Detailed Description

Author

Kirill Mishchenko

Tools for extracting meta information about machine learning algorithms.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.194 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/accuracy.hpp File Reference

Include dependency graph for accuracy.hpp:



Classes

- class **Accuracy**

The **Accuracy** (p. 1127) is a metric of performance for classification algorithms that is equal to a proportion of correctly labeled test items among all ones for given test items.

Namespaces

- **mlpack**
 .hpp
- **mlpack::cv**

40.194.1 Detailed Description

Author

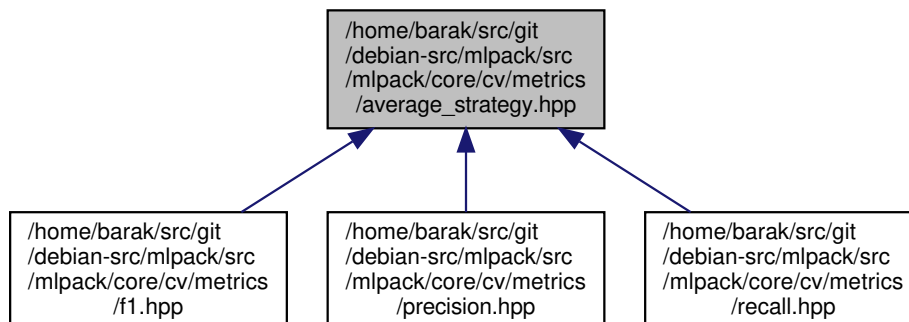
Kirill Mishchenko

The accuracy metric.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.195 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/average_strategy.hpp File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::cv**

Enumerations

- enum **AverageStrategy** {
 Binary,
 Micro,
 Macro }

This enum declares possible strategies for averaging that can be used in some metrics like precision, recall, and F1.

40.195.1 Detailed Description

Author

Kirill Mishchenko

Strategies for averaging.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.196 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/f1.hpp File Reference

Include dependency graph for f1.hpp:



Classes

- class **F1** < **AS**, **PositiveClass** >

F1 (p. 1132) is a metric of performance for classification algorithms that for binary classification is equal to $2 * precision * recall / (precision + recall)$.

Namespaces

- **mlpack**
 .hpp
- **mlpack::cv**

40.196.1 Detailed Description

Author

Kirill Mishchenko

The F1 metric.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.197 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/facilities.hpp File Reference

Include dependency graph for facilities.hpp:



Namespaces

- **mlpack**
 .hpp
- **mlpack::cv**

Functions

- `template<typename DataType >`
 void **AssertSizes** (const DataType &data, const arma::Row< size_t > &labels, const std::string &caller↵
 Description)
 Assert there is the same number of the given data points and labels.

40.197.1 Detailed Description

Author

Kirill Mishchenko

Functionality that is used more than in one metric.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.198 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/mse.hpp File Reference

Include dependency graph for mse.hpp:



Classes

- class **MSE**

The MeanSquaredError is a metric of performance for regression algorithms that is equal to the mean squared error between predicted values and ground truth (correct) values for given test items.

Namespaces

- **mlpack**
 .hpp
- **mlpack::cv**

40.198.1 Detailed Description

Author

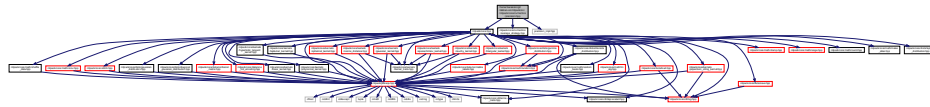
Kirill Mishchenko

The mean squared error (MSE).

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.199 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/precision.hpp File Reference

Include dependency graph for precision.hpp:



Classes

- class **Precision**< **AS**, **PositiveClass** >

Precision (p. 1145) is a metric of performance for classification algorithms that for binary classification is equal to $tp / (tp + fp)$, where tp and fp are the numbers of true positives and false positives respectively.

Namespaces

- **mlpack**
 .hpp
- **mlpack::cv**

40.199.1 Detailed Description

Author

Kirill Mishchenko

The precision metric.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.200 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/metrics/recall.hpp File Reference

Include dependency graph for recall.hpp:



Classes

- class **Recall**< **AS**, **PositiveClass** >

Recall (p. 1147) is a metric of performance for classification algorithms that for binary classification is equal to $tp / (tp + fn)$, where tp and fn are the numbers of true positives and false negatives respectively.

Namespaces

- **mlpack**
 .hpp
- **mlpack::cv**

40.200.1 Detailed Description

Author

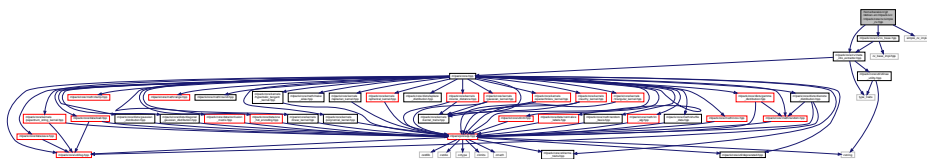
Kirill Mishchenko

The recall metric.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.201 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/simple_cv.hpp File Reference

Include dependency graph for simple_cv.hpp:



Classes

- class **SimpleCV**< **MLAlgorithm**, **Metric**, **MatType**, **PredictionsType**, **WeightsType** >

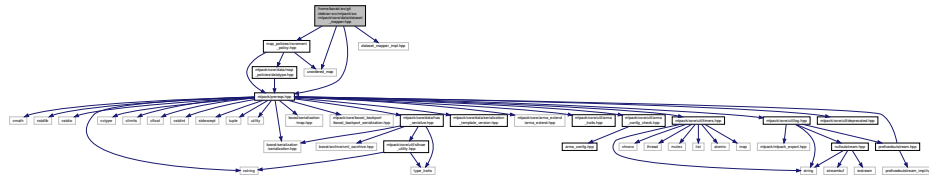
SimpleCV (p. 1151) splits data into two sets - training and validation sets - and then runs training on the training set and evaluates performance on the validation set.

Namespaces

- **mlpack**
 .hpp
- **mlpack::cv**

40.204 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/dataset_mapper.hpp File Reference

Include dependency graph for dataset_mapper.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **DatasetMapper**< **PolicyType**, **InputType** >
Auxiliary information for a dataset, including mappings to/from strings (or other types) and the datatype of each dimension.

Namespaces

- **mlpack**
.hpp
- **mlpack::data**
Functions to load and save matrices and models.

Typedefs

- using **DatasetInfo** = DatasetMapper< data::IncrementPolicy >

40.204.1 Detailed Description

Author

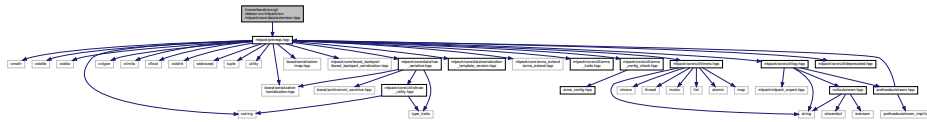
Ryan Curtin
 Keon Kim

Defines the DatasetMapper class, which holds information about a dataset. This is useful when the dataset contains categorical non-numeric features that needs to be mapped to categorical numeric features.

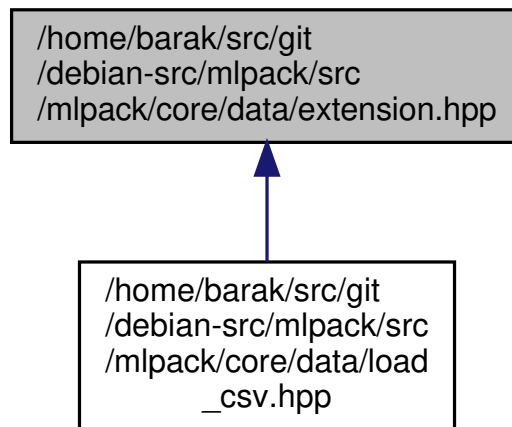
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.205 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/extension.hpp File Reference

Include dependency graph for extension.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::data**
 Functions to load and save matrices and models.

Functions

- std::string **Extension** (const std::string &filename)

40.205.1 Detailed Description

Author

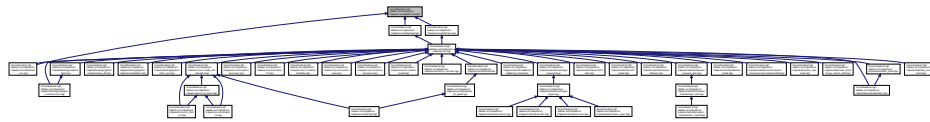
Ryan Curtin

Given a filename, extract its extension. This is used by **data::Load()** (p. 379) and **data::Save()** (p. 386).

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.206 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/format.hpp File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::data**

Functions to load and save matrices and models.

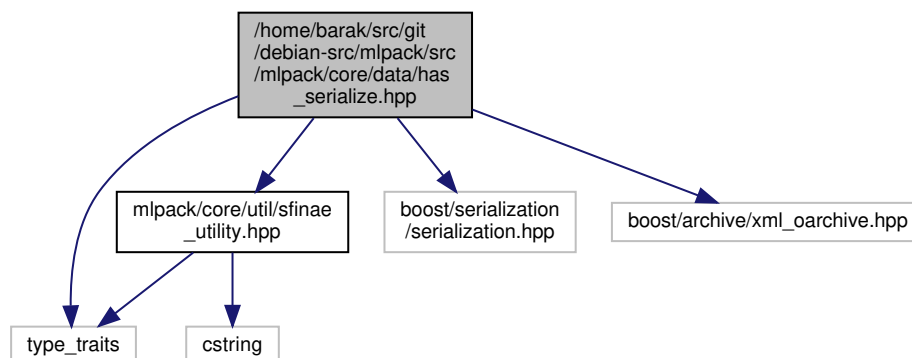
Enumerations

- enum **format** {
 autodetect,
 text,
 xml,
 binary }

*Define the formats we can read through **boost::serialization** (p. 251).*

40.207 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/has_serialize.hpp File Reference

Include dependency graph for has_serialize.hpp:

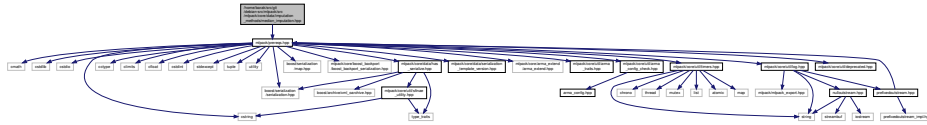


This graph shows which files directly or indirectly include this file:



40.211 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/median_imputation.hpp File Reference

Include dependency graph for median_imputation.hpp:



Classes

- class **MedianImputation**< **T** >
This is a class implementation of simple median imputation.

Namespaces

- **mlpack**
.hpp
- **mlpack::data**
Functions to load and save matrices and models.

40.211.1 Detailed Description

Author

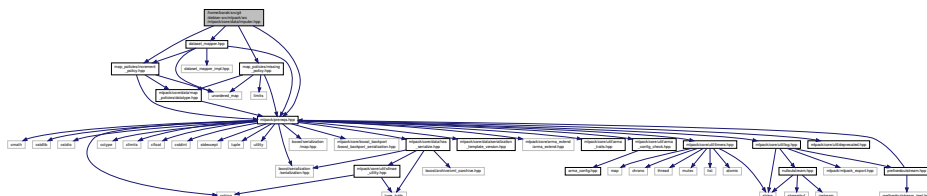
Keon Kim

Definition and Implementation of the MedianImputation class.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.212 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputer.hpp File Reference

Include dependency graph for imputer.hpp:



- `template<typename T>`
`bool IsNaNInf (T &val, const std::string &token)`

40.213.1 Detailed Description

Ryan Curtin

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.214 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/load.hpp File Reference

- **mlpack**
 .hpp
- **mlpack::data**

Generated by Doxygen

Functions

- `template<typename eT >`
`bool Load` (const std::string &filename, arma::Mat< eT > &matrix, const bool fatal=false, const bool transpose=true)
Loads a matrix from file, guessing the filetype from the extension.
- `template<typename eT >`
`bool Load` (const std::string &filename, arma::Col< eT > &vec, const bool fatal=false)
Don't document these with doxygen; these declarations aren't helpful to users.
- `template<typename eT >`
`bool Load` (const std::string &filename, arma::Row< eT > &rowvec, const bool fatal=false)
Load a row vector from a file, guessing the filetype from the extension.
- `template<typename eT, typename PolicyType >`
`bool Load` (const std::string &filename, arma::Mat< eT > &matrix, DatasetMapper< PolicyType > &info, const bool fatal=false, const bool transpose=true)
*Loads a matrix from a file, guessing the filetype from the extension and mapping categorical features with a **DatasetMapper** (p. 1172) object.*
- `template<typename T >`
`bool Load` (const std::string &filename, const std::string &name, T &t, const bool fatal=false, format f=format::autodetect)
Don't document these with doxygen; they aren't helpful for users to know about.

40.214.1 Detailed Description

Author

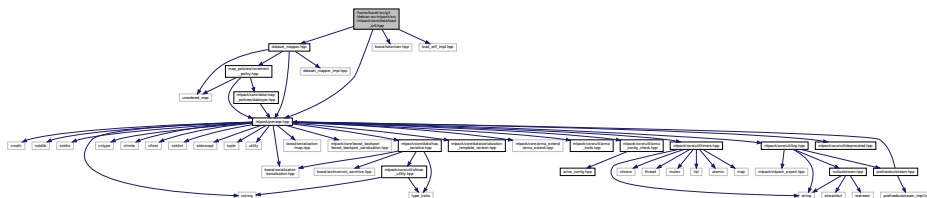
Ryan Curtin

Load an Armadillo matrix from file. This is necessary because Armadillo does not transpose matrices on input, and it allows us to give better error output.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.215 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/load_arff.hpp File Reference

Include dependency graph for load_arff.hpp:



Namespaces

- **mlpack**
 .hpp
- **mlpack::data**
 Functions to load and save matrices and models.

Functions

- template<typename eT >
 void **LoadARFF** (const std::string &filename, arma::Mat< eT > &matrix)
 A utility function to load an ARFF dataset as numeric features (that is, as an Armadillo matrix without any modification).
- template<typename eT, typename PolicyType >
 void **LoadARFF** (const std::string &filename, arma::Mat< eT > &matrix, DatasetMapper< PolicyType > &info)
 A utility function to load an ARFF dataset as numeric and categorical features, using the DatasetInfo structure for mapping.

40.215.1 Detailed Description

Author

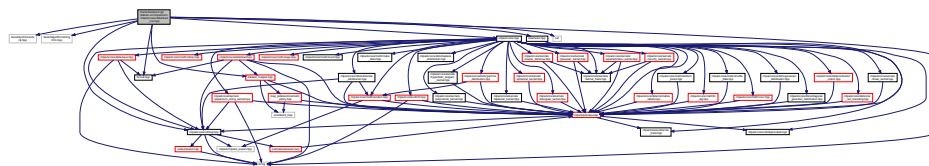
Ryan Curtin

Load an ARFF dataset.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.216 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/load_csv.hpp File Reference

Include dependency graph for load_csv.hpp:

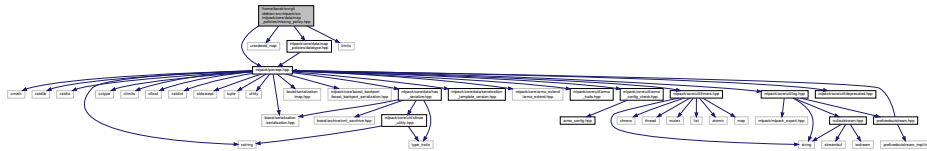


Classes

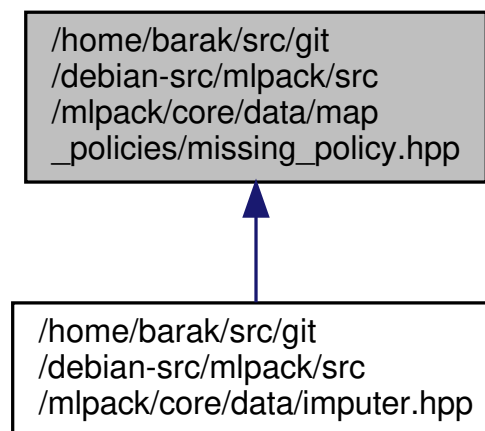
- class **LoadCSV**
 Load the csv file. This class use boost::spirit to implement the parser, please refer to following link <http://theboostcpplibraries.com/boost.spirit> for quick review.

40.219 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/map_policies/missing_policy.hpp File Reference

Include dependency graph for missing_policy.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **MissingPolicy**

MissingPolicy (p. 1193) is used as a helper class for *DatasetMapper* (p. 1172).

Namespaces

- **mlpack**
 .hpp
- **mlpack::data**

Functions to load and save matrices and models.

40.219.1 Detailed Description

Author

Keon Kim

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Author

Keon Kim

Missing map policy for dataset info.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.220 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/normalize_labels.hpp File Reference

Include dependency graph for normalize_labels.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::data**

Functions to load and save matrices and models.

Functions

- template<typename eT , typename RowType >
 void **NormalizeLabels** (const RowType &labelsIn, arma::Row< size_t > &labels, arma::Col< eT > &mapping)
Given a set of labels of a particular datatype, convert them to unsigned labels in the range [0, n) where n is the number of different labels.
- template<typename eT >
 void **RevertLabels** (const arma::Row< size_t > &labels, const arma::Col< eT > &mapping, arma::Row< eT > &labelsOut)
Given a set of labels that have been mapped to the range [0, n), map them back to the original labels given by the 'mapping' vector.

40.220.1 Detailed Description

Author

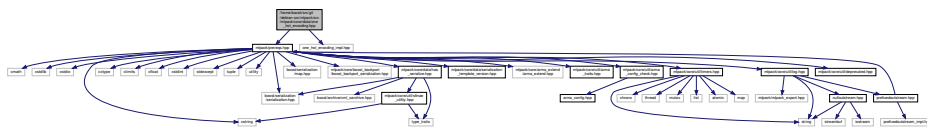
Ryan Curtin

Often labels are not given as {0, 1, 2, ...} but instead {1, 2, ...} or even {-1, 1} or otherwise. The purpose of this function is to normalize labels to {0, 1, 2, ...} and provide a mapping back to those labels.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.221 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/one_hot_encoding.hpp File Reference

Include dependency graph for one_hot_encoding.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- mlpack**
 .hpp
- mlpack::data**
Functions to load and save matrices and models.

Functions

- `template<typename eT , typename RowType >`
`void OneHotEncoding (const RowType &labelsIn, arma::Mat< eT > &output)`

Given a set of labels of a particular datatype, convert them to binary vector.

40.221.1 Detailed Description

Author

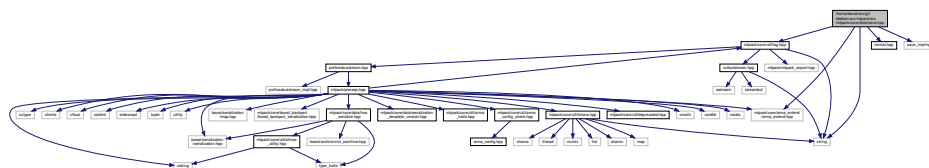
Jeffin Sam

One hot encoding functions. The purpose of this function is to convert categorical variables to binary vectors.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpac. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.222 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/save.hpp File Reference

Include dependency graph for save.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::data**

Functions to load and save matrices and models.

Functions

- `template<typename eT >`
`bool Save (const std::string &filename, const arma::Mat< eT > &matrix, const bool fatal=false, bool transpose=true)`
Saves a matrix to file, guessing the filetype from the extension.
- `template<typename T >`
`bool Save (const std::string &filename, const std::string &name, T &t, const bool fatal=false, format f=format_↵::autodetect)`
Saves a model to file, guessing the filetype from the extension, or, optionally, saving the specified format.

40.222.1 Detailed Description

Author

Ryan Curtin

Save an Armadillo matrix to file. This is necessary because Armadillo does not transpose matrices upon saving, and it allows us to give better error output.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.↵org/licenses/BSD-3-Clause> for more information.

40.223 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/serialization_template_↵_version.hpp File Reference

This graph shows which files directly or indirectly include this file:



Macros

- `#define BOOST_TEMPLATE_CLASS_VERSION(SIGNATURE, T, N)`
*Use this like **BOOST_CLASS_VERSION**() (p. 2953), but for templated classes.*

40.223.1 Detailed Description

Author

Ryan Curtin

A better version of the **BOOST_CLASS_VERSION**() (p. 2953) macro that supports templated classes.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.↵org/licenses/BSD-3-Clause> for more information.

Functions

- `template<typename T , typename U >`
`void Split (const arma::Mat< T > &input, const arma::Row< U > &inputLabel, arma::Mat< T > &trainData, arma::Mat< T > &testData, arma::Row< U > &trainLabel, arma::Row< U > &testLabel, const double testRatio)`
Given an input dataset and labels, split into a training set and test set.
- `template<typename T >`
`void Split (const arma::Mat< T > &input, arma::Mat< T > &trainData, arma::Mat< T > &testData, const double testRatio)`
Given an input dataset, split into a training set and test set.
- `template<typename T , typename U >`
`std::tuple< arma::Mat< T >, arma::Mat< T >, arma::Row< U >, arma::Row< U > > Split (const arma::Mat< T > &input, const arma::Row< U > &inputLabel, const double testRatio)`
Given an input dataset and labels, split into a training set and test set.
- `template<typename T >`
`std::tuple< arma::Mat< T >, arma::Mat< T > > Split (const arma::Mat< T > &input, const double testRatio)`
Given an input dataset, split into a training set and test set.

40.224.1 Detailed Description

Author

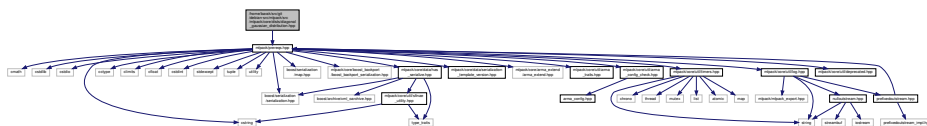
Tham Ngap Wei, Keon Kim

Defines **Split()** (p. 389), a utility function to split a dataset into a training set and a test set.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.225 `/home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/diagonal_gaussian_distribution.hpp` File Reference

Include dependency graph for `diagonal_gaussian_distribution.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **DiagonalGaussianDistribution**
A single multivariate Gaussian distribution with diagonal covariance.

Namespaces

- mlpack**
.hpp
- mlpack::distribution**
Probability distributions.

40.225.1 Detailed Description

Author

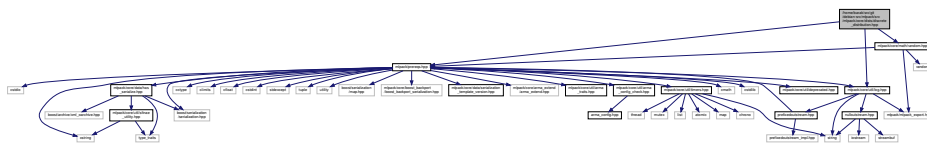
Kim SangYeon

Implementation of the Gaussian distribution with diagonal covariance.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.226 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/discrete_distribution.hpp File Reference

Include dependency graph for discrete_distribution.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **DiscreteDistribution**
A discrete distribution where the only observations are discrete observations.

Namespaces

- **mlpack**
 .hpp
- **mlpack::distribution**
 Probability distributions.

40.227.1 Detailed Description

Author

Yannis Mentekidis
 Rohan Raj

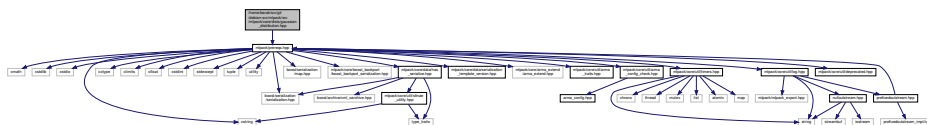
Implementation of a Gamma distribution of multidimensional data that fits gamma parameters (alpha, beta) to data. The fitting is done independently for each dataset dimension (row), based on the assumption each dimension is fully independent.

Based on "Estimating a Gamma Distribution" by Thomas P. Minka: research.microsoft.com/~minka/papers/minka-gamma.pdf

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.228 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/gaussian_distribution.hpp File Reference

Include dependency graph for gaussian_distribution.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **GaussianDistribution**
 A single multivariate Gaussian distribution.

Namespaces

- **mlpack**
 .hpp
- **mlpack::distribution**
 Probability distributions.

40.228.1 Detailed Description

Author

Ryan Curtin
Michael Fox

Implementation of the Gaussian distribution.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.229 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/laplace_distribution.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

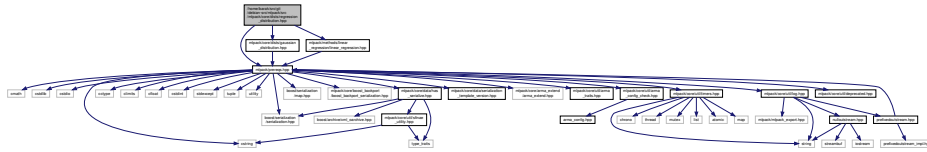
- class **LaplaceDistribution**
 The multivariate Laplace distribution centered at 0 has pdf.

Namespaces

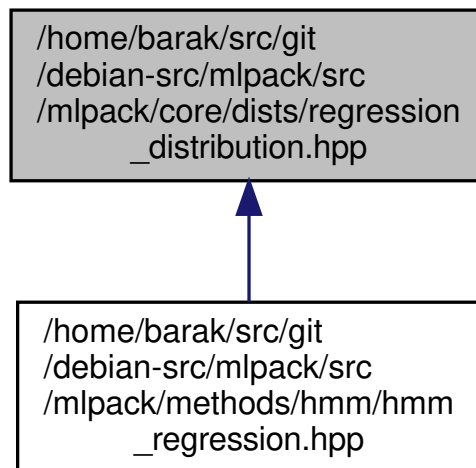
- **mlpack**
 .hpp
- **mlpack::distribution**
 Probability distributions.

40.230 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/regression_distribution.hpp File Reference

Include dependency graph for regression_distribution.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RegressionDistribution**

A class that represents a univariate conditionally Gaussian distribution.

Namespaces

- **mlpack**
.hpp
- **mlpack::distribution**
Probability distributions.

40.230.1 Detailed Description

Author

Michael Fox

Implementation of conditional Gaussian distribution for HMM regression (HMMR).

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.231 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/cv_function.hpp File Reference

Include dependency graph for cv_function.hpp:



Classes

- class **CVFunction**< **CVType**, **MLAlgorithm**, **TotalArgs**, **BoundArgs** >

This wrapper serves for adapting the interface of the cross-validation classes to the one that can be utilized by the mlpack optimizers.

Namespaces

- **mlpack**
 .hpp
- **mlpack::hpt**

40.231.1 Detailed Description

Author

Kirill Mishchenko

A cross-validation wrapper for optimizers.

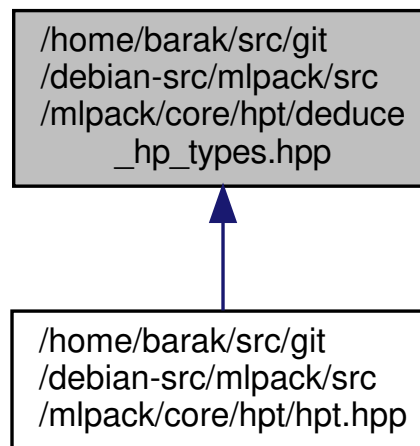
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.232 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/deduce_hp_types.hpp File Reference

Include dependency graph for deduce_hp_types.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct **DeduceHyperParameterTypes**< Args >
A type function for deducing types of hyper-parameters from types of arguments in the Optimize method in **HyperParameterTuner** (p. 1375).
- struct **DeduceHyperParameterTypes**< Args >::ResultHolder< HPTypes >
- struct **DeduceHyperParameterTypes**< PreFixedArg< T >, Args... >
Defining **DeduceHyperParameterTypes** (p. 1365) for the case when not all argument types have been processed, and the next one is the type of an argument that should be fixed.
- struct **DeduceHyperParameterTypes**< PreFixedArg< T >, Args... >::ResultHolder< HPTypes >
- struct **DeduceHyperParameterTypes**< T, Args... >
Defining **DeduceHyperParameterTypes** (p. 1365) for the case when not all argument types have been processed, and the next one (T) is a collection type or an arithmetic type.
- struct **DeduceHyperParameterTypes**< T, Args... >::IsCollectionType< Type >
A type function to check whether Type is a collection type (for that it should define value_type).
- struct **DeduceHyperParameterTypes**< T, Args... >::ResultHolder< HPTypes >
- struct **DeduceHyperParameterTypes**< T, Args... >::ResultHPType< ArgumentType, IsArithmetic >
A type function to deduce the result hyper-parameter type for ArgumentType.
- struct **DeduceHyperParameterTypes**< T, Args... >::ResultHPType< ArithmeticType, true >
- struct **DeduceHyperParameterTypes**< T, Args... >::ResultHPType< CollectionType, false >

Namespaces

- **mlpack**
 .hpp
- **mlpack::hpt**

Typedefs

- `template<typename... Args>`
 using **TupleOfHyperParameters** = typename DeduceHyperParameterTypes< Args... >::TupleType
 *A short alias for deducing types of hyper-parameters from types of arguments in the Optimize method in **HyperParameterTuner** (p. 1375).*

40.232.1 Detailed Description

Author

Kirill Mishchenko

Tools to deduce types of hyper-parameters from types of arguments in the Optimize method in HyperParameterTuner.

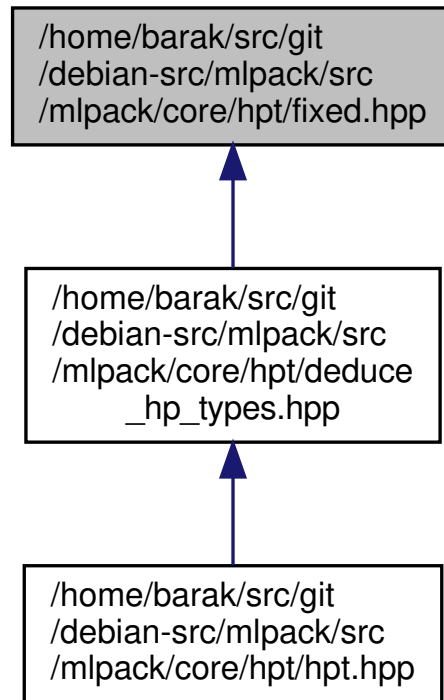
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.233 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/fixed.hpp File Reference

Include dependency graph for fixed.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct **FixedArg**< T, I >
A struct for storing information about a fixed argument.
- class **IsPreFixedArg**< T >
*A type function for checking whether the given type is **PreFixedArg** (p. 1381).*
- struct **PreFixedArg**< T >
*A struct for marking arguments as ones that should be fixed (it can be useful for the Optimize method of **Hyper↔ParameterTuner** (p. 1375)).*
- struct **PreFixedArg**< T >
*A struct for marking arguments as ones that should be fixed (it can be useful for the Optimize method of **Hyper↔ParameterTuner** (p. 1375)).*
- struct **PreFixedArg**< T & >
The specialization of the template for references.

Namespaces

- **mlpack**
.hpp
- **mlpack::hpt**

Functions

- `template<typename T>`
`PreFixedArg< T > Fixed (T &&value)`
Mark the given argument as one that should be fixed.

40.233.1 Detailed Description

Author

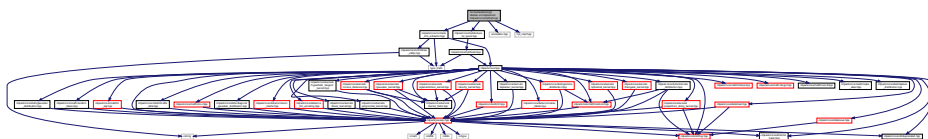
Kirill Mishchenko

Facilities for supporting fixed arguments.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.234 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/hpt.hpp File Reference

Include dependency graph for hpt.hpp:



Classes

- class **HyperParameterTuner**< **MLAlgorithm**, **Metric**, **CV**, **OptimizerType**, **MatType**, **PredictionsType**, **WeightsType** >
*The class **HyperParameterTuner** (p. 1375) for the given **MLAlgorithm** utilizes the provided **Optimizer** to find the values of hyper-parameters that optimize the value of the given **Metric**.*

Namespaces

- **mlpack**
`.hpp`
- **mlpack::hpt**

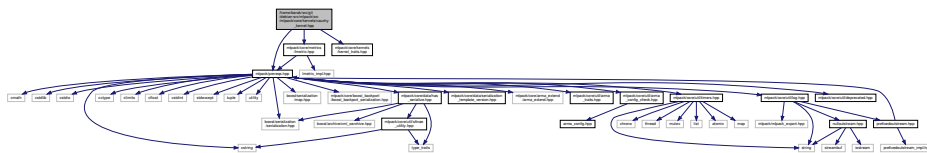
40.235 /home/barak/src/git/debian-src/mlpack/doc/guide/hpt.hpp File Reference

Namespaces

- **mlpack**
 .hpp
- **mlpack::hpt**

40.236 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/cauchy_kernel.hpp File Reference

Include dependency graph for `cauchy_kernel.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **CauchyKernel**
 The Cauchy kernel.
- class **KernelTraits< CauchyKernel >**
 Kernel traits for the Cauchy kernel.

Namespaces

- **mlpack**
 .hpp
- **mlpack::kernel**
 Kernel functions.

The diagram illustrates the structure of a research project titled "Research Project: Exploring the Impact of Digital Marketing on Consumer Behavior". The project is organized into a hierarchical structure with main branches and sub-topics.

Main Branches:

- Introduction**
 - Background and Significance
 - Research Objectives and Scope
 - Thesis Statement
- Literature Review**
 - Digital Marketing Strategies
 - Consumer Behavior Theories
 - Previous Research Findings
 - Research Gaps and Contributions
- Methodology**
 - Research Design
 - Sampling Method
 - Data Collection Methods
 - Measurement Instruments
 - Data Analysis Techniques
- Data Collection**
 - Survey Design and Distribution
 - Interviews and Focus Groups
 - Secondary Data Sources
 - Data Cleaning and Preparation
- Analysis**
 - Descriptive Statistics
 - Correlation Analysis
 - Regression Analysis
 - Mediation and Moderation Analysis
 - Qualitative Data Analysis
- Results**
 - Demographic Characteristics
 - Descriptive Statistics of Variables
 - Correlation Results
 - Regression Results
 - Mediation and Moderation Results
 - Qualitative Findings
- Discussion**
 - Interpretation of Results
 - Implications for Theory and Practice
 - Limitations and Future Research
- Conclusion**
 - Summary of Findings
 - Final Thoughts and Recommendations

The diagram shows a complex network of relationships between these main branches and their sub-topics, with some sub-topics further branching out into more detailed sub-topics, indicating a highly structured and comprehensive research project.

[illegible]

- class **GaussianKernel**
The standard Gaussian kernel.
- class **KernelTraits**< **GaussianKernel** >
Kernel traits for the Gaussian kernel.

- **mlpack**
.hpp
- **mlpack::kernel**
Kernel functions.

Wei Guan
James Cline
Ryan Curtin

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.241 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/hyperbolic_tangent_↵
_kernel.hpp File Reference

Include dependency graph for hyperbolic_tangent_kernel.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **HyperbolicTangentKernel**
Hyperbolic tangent kernel.

Namespaces

- **mlpack**
.hpp
- **mlpack::kernel**
Kernel functions.

40.241.1 Detailed Description

Author

Ajinkya Kale kaleajinkya@gmail.com

Implementation of the hyperbolic tangent kernel.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.242 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/kernel_traits.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **KernelTraits**< **KernelType** >
This is a template class that can provide information about various kernels.

Namespaces

- **mlpack**
.hpp
- **mlpack::kernel**
Kernel functions.

40.242.1 Detailed Description

Author

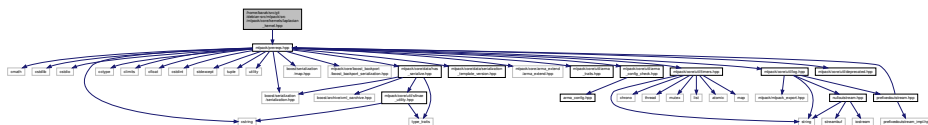
Ryan Curtin

This provides the KernelTraits class, a template class to get information about various kernels.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.243 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/laplacian_kernel.hpp File Reference

Include dependency graph for laplacian_kernel.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **KernelTraits**< **LaplacianKernel** >
Kernel traits of the Laplacian kernel.
- class **LaplacianKernel**
The standard Laplacian kernel.

Namespaces

- **mlpack**
.hpp
- **mlpack::kernel**
Kernel functions.

40.243.1 Detailed Description

Author

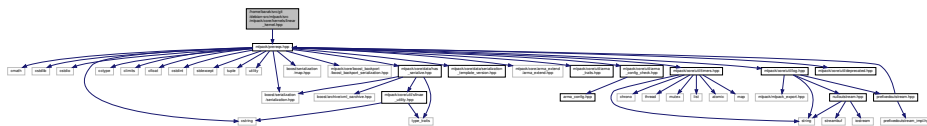
Ajinkya Kale kaleajinkya@gmail.com

Implementation of the Laplacian kernel (LaplacianKernel).

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.244 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/linear_kernel.hpp File Reference

Include dependency graph for linear_kernel.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **LinearKernel**
The simple linear kernel (dot product).

Namespaces

- **mlpack**
.hpp
- **mlpack::kernel**
Kernel functions.

40.244.1 Detailed Description

Author

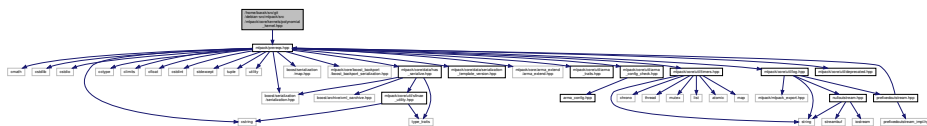
Wei Guan
James Cline
Ryan Curtin

Implementation of the linear kernel (just the standard dot product).

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.245 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/polynomial_kernel.hpp File Reference

Include dependency graph for polynomial_kernel.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::kernel**
 Kernel functions.

40.246.1 Detailed Description

Author

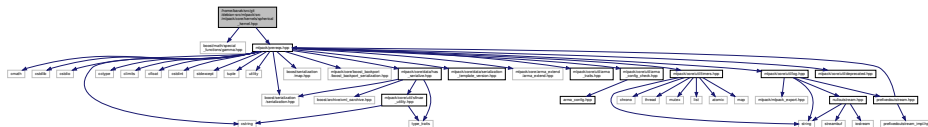
Ryan Curtin

Implementation of the p-spectrum string kernel, created for use with FastMKS. Instead of passing a data matrix to FastMKS which stores the kernels, pass a one-dimensional data matrix (data vector) to FastMKS which stores indices of strings; then, the actual strings are given to the PSpectrumStringKernel at construction time, and the kernel knows to map the indices to actual strings.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.247 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/spherical_kernel.hpp File Reference

Include dependency graph for spherical_kernel.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **KernelTraits< SphericalKernel >**
 Kernel traits for the spherical kernel.
- class **SphericalKernel**
 The spherical kernel, which is 1 when the distance between the two argument points is less than or equal to the bandwidth, or 0 otherwise.

Namespaces

- **mlpack**
.hpp
- **mlpack::kernel**
Kernel functions.

40.247.1 Detailed Description

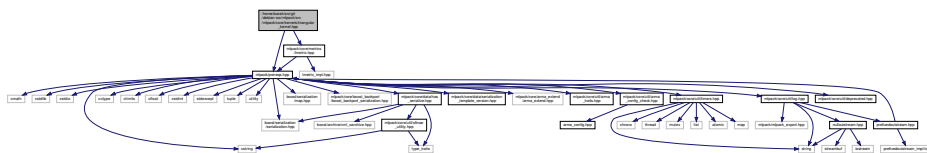
Author

Neil Slagle

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.248 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/triangular_kernel.hpp File Reference

Include dependency graph for triangular_kernel.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **KernelTraits< TriangularKernel >**
Kernel traits for the triangular kernel.
- class **TriangularKernel**
The trivially simple triangular kernel, defined by.

Namespaces

- **mlpack**
.hpp
- **mlpack::kernel**
Kernel functions.

40.248.1 Detailed Description

Author

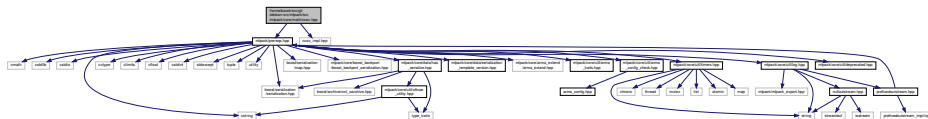
Ryan Curtin

Definition and implementation of the trivially simple triangular kernel.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.249 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/ccov.hpp File Reference

Include dependency graph for ccov.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
.hpp
- **mlpack::math**
Miscellaneous math routines.

Functions

- `template<typename eT >`
`arma::Mat< eT > ColumnCovariance (const arma::Mat< eT > &A, const size_t norm_type=0)`
- `template<typename T >`
`arma::Mat< std::complex< T > > ColumnCovariance (const arma::Mat< std::complex< T > > &A, const size_t norm_type=0)`

40.249.1 Detailed Description

Author

Ryan Curtin
 Conrad Sanderson

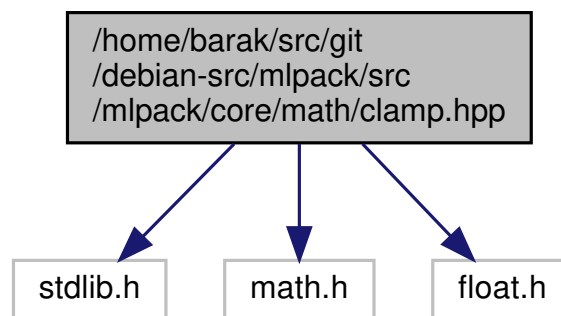
`ColumnCovariance(X)` is same as `cov(trans(X))` but without the cost of computing `trans(X)`

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.250 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/clamp.hpp File Reference

Miscellaneous math clamping routines.

Include dependency graph for `clamp.hpp`:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::math**
 Miscellaneous math routines.

Functions

- double **ClampNonNegative** (const double d)
 Forces a number to be non-negative, turning negative numbers into zero.
- double **ClampNonPositive** (const double d)
 Forces a number to be non-positive, turning positive numbers into zero.
- double **ClampRange** (double value, const double rangeMin, const double rangeMax)
 Clamp a number between a particular range.

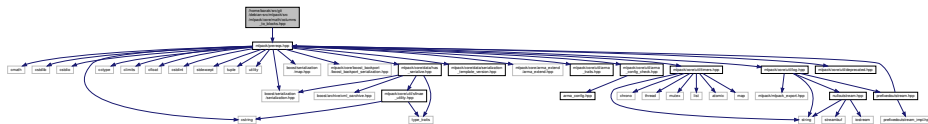
40.250.1 Detailed Description

Miscellaneous math clamping routines.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.251 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/columns_to_blocks.hpp File Reference

Include dependency graph for columns_to_blocks.hpp:



Classes

- class **ColumnsToBlocks**
 Transform the columns of the given matrix into a block format.

Namespaces

- **mlpack**
 .hpp
- **mlpack::math**
 Miscellaneous math routines.

Functions

- void **Center** (const arma::mat &x, arma::mat &xCentered)
Creates a centered matrix, where centering is done by subtracting the sum over the columns (a column vector) from each column of the matrix.
- void **Orthogonalize** (const arma::mat &x, arma::mat &W)
Orthogonalize x and return the result in W, using eigendecomposition.
- void **Orthogonalize** (arma::mat &x)
Orthogonalize x in-place.
- void **RandVector** (arma::vec &v)
Overwrites a dimension-N vector to a random vector on the unit sphere in R^N .
- void **RemoveRows** (const arma::mat &input, const std::vector< size_t > &rowsToRemove, arma::mat &output)
Remove a certain set of rows in a matrix while copying to a second matrix.
- template<typename T >
T **Sign** (const T x)
Signum function.
- void **Smat** (const arma::vec &input, arma::mat &output)
The inverse of Svec.
- void **Svec** (const arma::mat &input, arma::vec &output)
Upper triangular representation of a symmetric matrix, scaled such that, $\text{dot}(\text{Svec}(A), \text{Svec}(B)) == \text{dot}(A, B)$ for symmetric A, B .
- void **Svec** (const arma::sp_mat &input, arma::sp_vec &output)
- size_t **SvecIndex** (size_t i, size_t j, size_t n)
*Return the index such that $A[i,j] == \text{factr}(i, j) * \text{svec}(A)[\text{pos}(i, j)]$, where $\text{factr}(i, j) = \sqrt{2}$ if $i \neq j$ and 1 otherwise.*
- void **SymKronId** (const arma::mat &A, arma::mat &op)
If A is a symmetric matrix, then SymKronId returns an operator Op such that.
- void **VectorPower** (arma::vec &vec, const double power)
Auxiliary function to raise vector elements to a specific power.
- void **WhitenUsingEig** (const arma::mat &x, arma::mat &xWhitened, arma::mat &whiteningMatrix)
Whitens a matrix using the eigendecomposition of the covariance matrix.
- void **WhitenUsingSVD** (const arma::mat &x, arma::mat &xWhitened, arma::mat &whiteningMatrix)
Whitens a matrix using the singular value decomposition of the covariance matrix.

40.252.1 Detailed Description

Author

Nishant Mehta

Linear algebra utilities.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.253 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/log_add.hpp File Reference

Include dependency graph for log_add.hpp:



Namespaces

- **mlpack**
 .hpp
- **mlpack::math**
 Miscellaneous math routines.

Functions

- template<typename T >
 T::elem_type **AccuLog** (const T &x)
 Sum a vector of log values.
- template<typename T >
 T **LogAdd** (T x, T y)
 Internal log-addition.

40.253.1 Detailed Description

Author

Arash Abghari

Functions for logarithmic addition.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.254 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/make_alias.hpp File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
.hpp
- **mlpack::math**
Miscellaneous math routines.

Functions

- `template<typename ElemType >`
`void ClearAlias (arma::Mat< ElemType > &mat)`
Clear an alias so that no data is overwritten.
- `template<typename ElemType >`
`void ClearAlias (arma::SpMat< ElemType > &)`
Clear an alias for a sparse matrix.
- `template<typename ElemType >`
`arma::Cube< ElemType > MakeAlias (arma::Cube< ElemType > &input, const bool strict=true)`
Make an alias of a dense cube.
- `template<typename ElemType >`
`arma::Mat< ElemType > MakeAlias (arma::Mat< ElemType > &input, const bool strict=true)`
Make an alias of a dense matrix.
- `template<typename ElemType >`
`arma::Row< ElemType > MakeAlias (arma::Row< ElemType > &input, const bool strict=true)`
Make an alias of a dense row.
- `template<typename ElemType >`
`arma::Col< ElemType > MakeAlias (arma::Col< ElemType > &input, const bool strict=true)`
Make an alias of a dense column.
- `template<typename ElemType >`
`arma::SpMat< ElemType > MakeAlias (const arma::SpMat< ElemType > &input, const bool=true)`
Make a copy of a sparse matrix (an alias is not possible).
- `template<typename ElemType >`
`arma::SpRow< ElemType > MakeAlias (const arma::SpRow< ElemType > &input, const bool=true)`
Make a copy of a sparse row (an alias is not possible).
- `template<typename ElemType >`
`arma::SpCol< ElemType > MakeAlias (const arma::SpCol< ElemType > &input, const bool=true)`
Make a copy of a sparse column (an alias is not possible).

40.254.1 Detailed Description

Author

Ryan Curtin

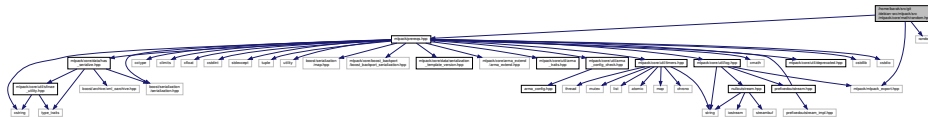
Make an alias of a matrix. For sparse matrices, unfortunately no alias can be made and a copy must be incurred.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.255 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/random.hpp File Reference

Miscellaneous math random-related routines.

Include dependency graph for random.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::math**
 Miscellaneous math routines.

Functions

- void **FixedRandomSeed** ()
 Set the random seed to a fixed number.
- void **ObtainDistinctSamples** (const size_t loInclusive, const size_t hiExclusive, const size_t maxNumSamples, arma::uvec &distinctSamples)
 Obtains no more than maxNumSamples distinct samples.
- double **RandBernoulli** (const double input)
 Generates a 0/1 specified by the input.
- int **RandInt** (const int hiExclusive)
 Generates a uniform random integer.
- int **RandInt** (const int lo, const int hiExclusive)
 Generates a uniform random integer.
- double **RandNormal** ()
 Generates a normally distributed random number with mean 0 and variance 1.
- double **RandNormal** (const double mean, const double variance)
 Generates a normally distributed random number with specified mean and variance.
- double **Random** ()
 Generates a uniform random number between 0 and 1.
- double **Random** (const double lo, const double hi)
 Generates a uniform random number in the specified range.
- void **RandomSeed** (const size_t seed)
 Set the random seed used by the random functions (**Random()** (p. 418) and **RandInt()** (p. 417)).

Variables

- `MLPACK_EXPORT std::mt19937 randGen`
MLPACK_EXPORT is required for global variables; it exports the symbols correctly on Windows.
- `MLPACK_EXPORT std::normal_distribution randNormalDist`
- `MLPACK_EXPORT std::uniform_real_distribution randUniformDist`

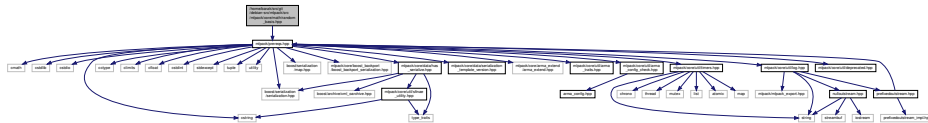
40.255.1 Detailed Description

Miscellaneous math random-related routines.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.256 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/random_basis.hpp File Reference

Include dependency graph for `random_basis.hpp`:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
.hpp
- **mlpack::math**
Miscellaneous math routines.

Functions

- `void RandomBasis(arma::mat &basis, const size_t d)`
Create a random d-dimensional orthogonal basis, storing it in the given matrix.

40.256.1 Detailed Description

Author

Ryan Curtin

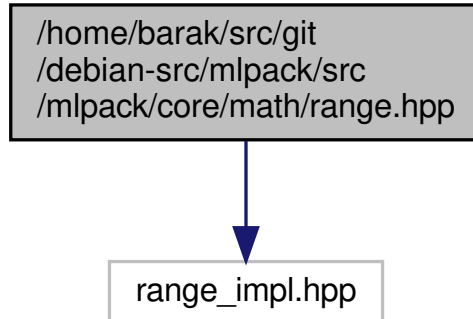
Generate a random d-dimensional basis.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.257 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/range.hpp File Reference

Definition of the Range class, which represents a simple range with a lower and upper bound.

Include dependency graph for range.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RangeType**< T >
Simple real-valued range.
- class **RangeType**< T >
Simple real-valued range.

Namespaces

- **mlpack**
 .hpp
- **mlpack::math**
 Miscellaneous math routines.

Typedefs

- typedef RangeType< double > **Range**
 3.0.0 TODO: break reverse-compatibility by changing **RangeType** (p. 1549) to **Range**.

40.257.1 Detailed Description

Definition of the Range class, which represents a simple range with a lower and upper bound.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.258 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/round.hpp File Reference

This graph shows which files directly or indirectly include this file:



40.258.1 Detailed Description

Author

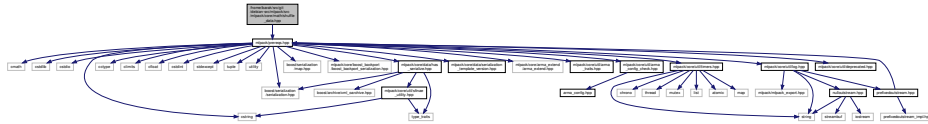
Ryan Curtin

Implementation of round() for use on Visual Studio, where C99 isn't implemented.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.259 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/shuffle_data.hpp File Reference

Include dependency graph for shuffle_data.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::math**
 Miscellaneous math routines.

Functions

- template<typename MatType , typename LabelsType >
 void **ShuffleData** (const MatType &inputPoints, const LabelsType &inputLabels, MatType &outputPoints, LabelsType &outputLabels, const **std::enable_if_t**<!arma::is_SpMat< MatType >::value > *=0, const **std::enable_if_t**<!arma::is_Cube< MatType >::value > *=0)
 Shuffle a dataset and associated labels (or responses).
- template<typename MatType , typename LabelsType >
 void **ShuffleData** (const MatType &inputPoints, const LabelsType &inputLabels, MatType &outputPoints, LabelsType &outputLabels, const **std::enable_if_t**< arma::is_SpMat< MatType >::value > *=0, const **std::enable_if_t**<!arma::is_Cube< MatType >::value > *=0)
 Shuffle a sparse dataset and associated labels (or responses).
- template<typename MatType , typename LabelsType >
 void **ShuffleData** (const MatType &inputPoints, const LabelsType &inputLabels, MatType &outputPoints, LabelsType &outputLabels, const **std::enable_if_t**<!arma::is_SpMat< MatType >::value > *=0, const **std::enable_if_t**< arma::is_Cube< MatType >::value > *=0, const **std::enable_if_t**< arma::is_Cube< LabelsType >::value > *=0)
 Shuffle a cube-shaped dataset and associated labels (or responses) which are also cube-shaped.
- template<typename MatType , typename LabelsType , typename WeightsType >
 void **ShuffleData** (const MatType &inputPoints, const LabelsType &inputLabels, const WeightsType &inputWeights, MatType &outputPoints, LabelsType &outputLabels, WeightsType &outputWeights, const **std::enable_if_t**<!arma::is_SpMat< MatType >::value > *=0, const **std::enable_if_t**<!arma::is_Cube< MatType >::value > *=0)

Shuffle a dataset and associated labels (or responses) and weights.

- `template<typename MatType , typename LabelsType , typename WeightsType >`
`void ShuffleData (const MatType &inputPoints, const LabelsType &inputLabels, const WeightsType &inputWeights, MatType &outputPoints, LabelsType &outputLabels, WeightsType &outputWeights, const std::enable_if_t< arma::is_SpMat< MatType >::value > !=0, const std::enable_if_t<!arma::is_Cube< MatType >::value > !=0)`

Shuffle a sparse dataset and associated labels (or responses) and weights.

40.259.1 Detailed Description

Author

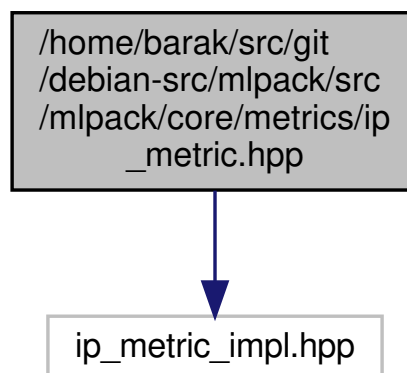
Ryan Curtin

Given data points and labels, shuffle their ordering.

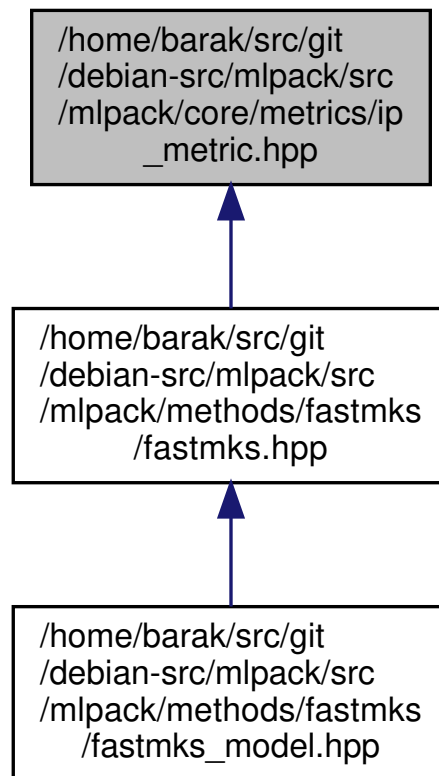
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.260 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/metrics/ip_metric.hpp File Reference

Include dependency graph for ip_metric.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **IPMetric**< **KernelType** >

The inner product metric, **IPMetric** (p. 1565), takes a given Mercer kernel (**KernelType**), and when **Evaluate()** (p. 1567) is called, returns the distance between the two points in kernel space:

Namespaces

- **mlpack**
 .hpp
- **mlpack::metric**

40.260.1 Detailed Description

Author

Ryan Curtin

Inner product induced metric. If given a kernel function, this gives the complementary metric.

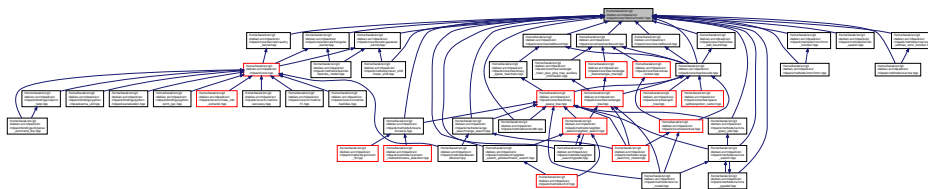
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.261 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/metrics/lmetric.hpp File Reference

Include dependency graph for lmetric.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **LMetric**< **TPower**, **TTakeRoot** >
The L_p metric for arbitrary integer p , with an option to take the root.

Namespaces

- **mlpack**
 .hpp
- **mlpack::metric**

Typedefs

- typedef LMetric< INT_MAX, false > **ChebyshevDistance**
The L -infinity distance.
- typedef LMetric< 2, true > **EuclideanDistance**
The Euclidean (L_2) distance.
- typedef LMetric< 1, false > **ManhattanDistance**
The Manhattan (L_1) distance.
- typedef LMetric< 2, false > **SquaredEuclideanDistance**
The squared Euclidean (L_2) distance.

Namespaces

- **mlpack**
 .hpp
- **mlpack::bound**
- **mlpack::bound::addr**

Functions

- template<typename AddressType , typename VecType >
void **AddressToPoint** (VecType &point, const AddressType &address)
Translate the address to the point.
- template<typename AddressType1 , typename AddressType2 >
int **CompareAddresses** (const AddressType1 &addr1, const AddressType2 &addr2)
Compare two addresses.
- template<typename AddressType1 , typename AddressType2 , typename AddressType3 >
bool **Contains** (const AddressType1 &address, const AddressType2 &loBound, const AddressType3 &hiBound)
Returns true if an address is contained between two other addresses.
- template<typename AddressType , typename VecType >
void **PointToAddress** (AddressType &address, const VecType &point)
Calculate the address of a point.

40.263.1 Detailed Description

Author

Mikhail Lozhnikov

This file contains a series of functions for translating points to addresses and back and functions for comparing addresses.

The notion of addresses is described in the following paper.

```
@inproceedings{bayer1997,
  author = {Bayer, Rudolf},
  title = {The Universal B-Tree for Multidimensional Indexing: General
    Concepts},
  booktitle = {Proceedings of the International Conference on Worldwide
    Computing and Its Applications},
  series = {WWCA '97},
  year = {1997},
  isbn = {3-540-63343-X},
  pages = {198--209},
  numpages = {12},
  publisher = {Springer-Verlag},
  address = {London, UK, UK},
}
```

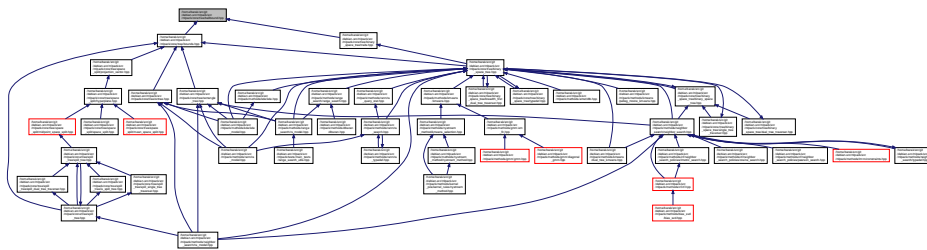
40.264 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ballbound.hpp File Reference

Bounds that are useful for binary space partitioning trees.

Include dependency graph for ballbound.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **BallBound**< **MetricType**, **VecType** >
Ball bound encloses a set of points at a specific distance (radius) from a specific point (center).
- struct **BoundTraits**< **BallBound**< **MetricType**, **VecType** > >
*A specialization of **BoundTraits** (p. 1002) for this bound type.*

Namespaces

- **mlpack**
.hpp
- **mlpack::bound**

40.264.1 Detailed Description

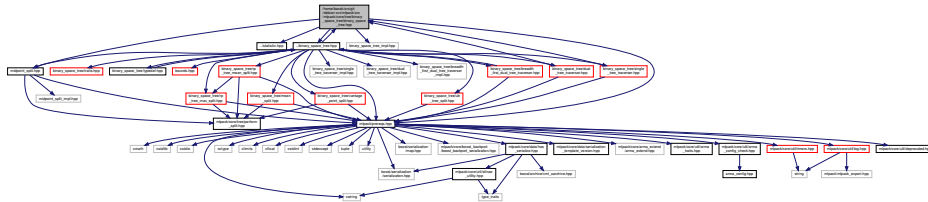
Bounds that are useful for binary space partitioning trees.

Interface to a ball bound that works in arbitrary metric spaces.

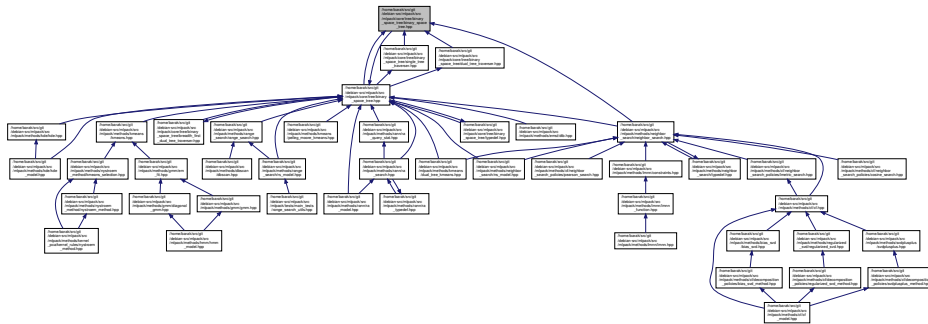
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.265 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/binary_space_tree.hpp File Reference

Include dependency graph for binary_space_tree.hpp:



This graph shows which files directly or indirectly include this file:



Classes

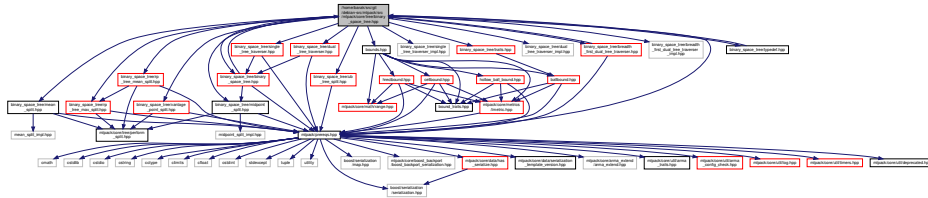
- class **BinarySpaceTree**< **MetricType**, **StatisticType**, **MatType**, **BoundType**, **SplitType** >
A binary space partitioning tree, such as a KD-tree or a ball tree.
- class **BinarySpaceTree**< **MetricType**, **StatisticType**, **MatType**, **BoundType**, **SplitType** >::**BreadthFirstDualTreeTraverser**< **RuleType** >
- class **BinarySpaceTree**< **MetricType**, **StatisticType**, **MatType**, **BoundType**, **SplitType** >::**DualTreeTraverser**< **RuleType** >
A dual-tree traverser for binary space trees; see dual_tree_traverser.hpp.
- class **BinarySpaceTree**< **MetricType**, **StatisticType**, **MatType**, **BoundType**, **SplitType** >::**SingleTreeTraverser**< **RuleType** >
A single-tree traverser for binary space trees; see single_tree_traverser.hpp for implementation.

Namespaces

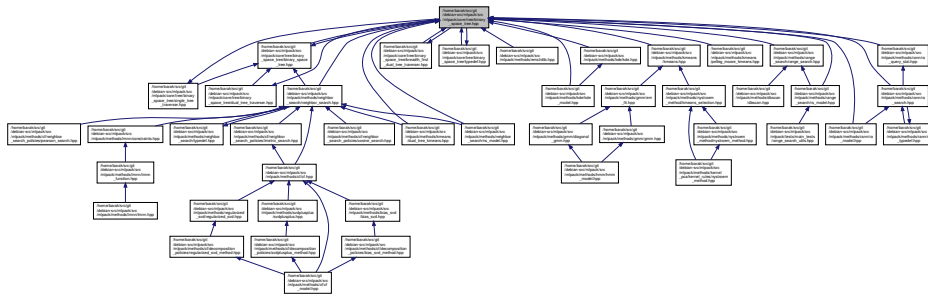
- **mlpack**
.hpp
- **mlpack::tree**
Trees and tree-building procedures.

40.266 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree.hpp
File Reference

Include dependency graph for `binary_space_tree.hpp`:

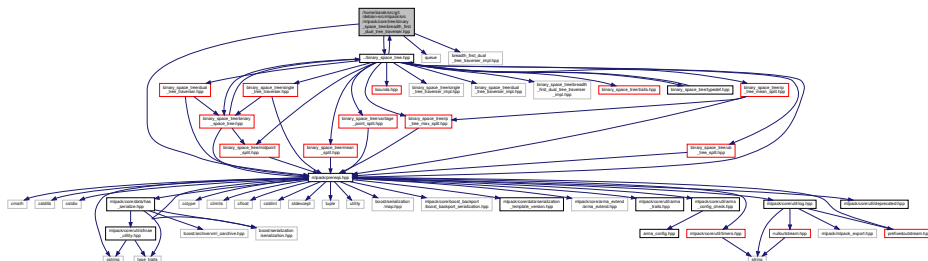


This graph shows which files directly or indirectly include this file:

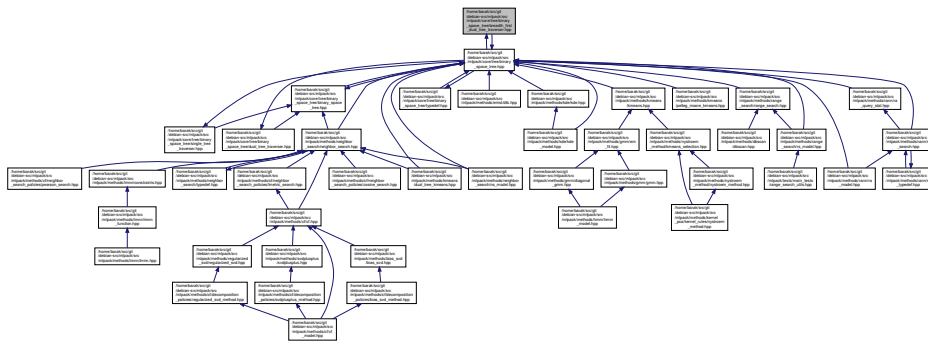


40.267 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/breadth_↵
_first_dual_tree_traverser.hpp File Reference

Include dependency graph for breadth_first_dual_tree_traverser.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **BinarySpaceTree**< **MetricType**, **StatisticType**, **MatType**, **BoundType**, **SplitType** >::**BreadthFirst**↵
DualTreeTraverser< **RuleType** >
- struct **QueueFrame**< **TreeType**, **TraversalInfoType** >

Namespaces

- **mlpack**
.hpp
- **mlpack::tree**
Trees and tree-building procedures.

40.267.1 Detailed Description

Author

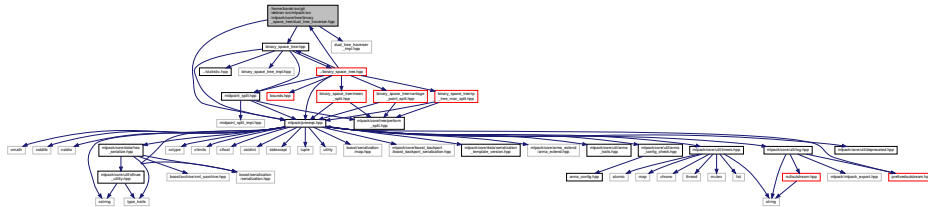
Ryan Curtin

Defines the `BreadthFirstDualTreeTraverser` for the `BinarySpaceTree` tree type. This is a nested class of `BinarySpaceTree` which traverses two trees in a breadth-first manner with a given set of rules which indicate the branches which can be pruned and the order in which to recurse.

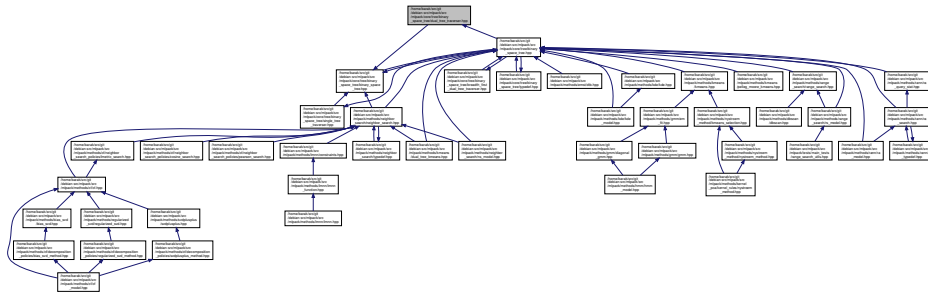
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpak. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.268 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/dual_↵
_tree_traverser.hpp File Reference

Include dependency graph for dual_tree_traverser.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **BinarySpaceTree**< **MetricType**, **StatisticType**, **MatType**, **BoundType**, **SplitType** >::DualTree↵
Traverser< **RuleType** >

A dual-tree traverser for binary space trees; see [dual_tree_traverser.hpp](#).

Namespaces

- **mlpack**
 .hpp
- **mlpack::tree**

Trees and tree-building procedures.

Include dependency graph for dual_tree_traverser.hpp:

[illegible]

- class **CoverTree**< **MetricType**, **StatisticType**, **MatType**, **RootPointPolicy** >::DualTreeTraverser< **RuleType** >

Namespaces

- 40.270 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree/dual_tree_↵
traverser.hpp File Reference

[illegible]

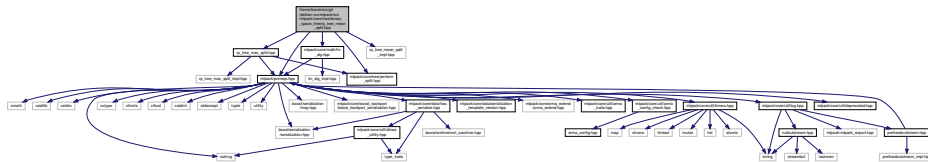
- class **MidpointSplit**< **BoundType**, **MatType** >
A binary space partitioning tree node is split into its left and right child.
- struct **MidpointSplit**< **BoundType**, **MatType** >::**SplitInfo**
A struct that contains an information about the split.

- **mlpack**
.hpp
- **mlpack::tree**
Trees and tree-building procedures.

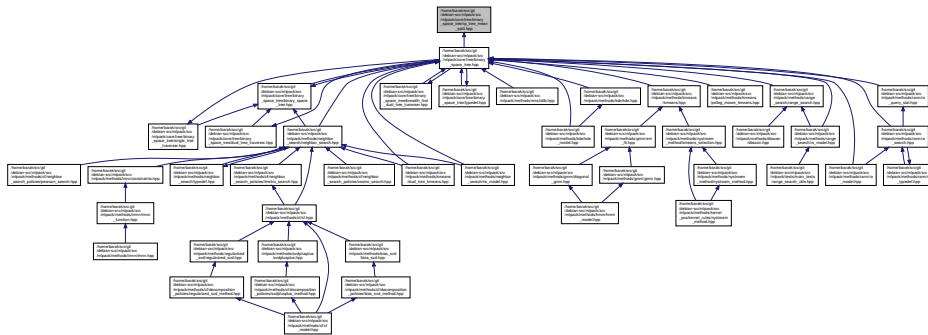
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

tree_mean_split.hpp File Reference

Include dependency graph for `rp_tree_mean_split.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **RPTreeMeanSplit**< **BoundType**, **MatType** >
This class splits a binary space tree.
- struct **RPTreeMeanSplit**< **BoundType**, **MatType** >::**SplitInfo**
An information about the partition.

Namespaces

- **mlpack**
.hpp
- **mlpack::tree**
Trees and tree-building procedures.

40.275.1 Detailed Description

Author

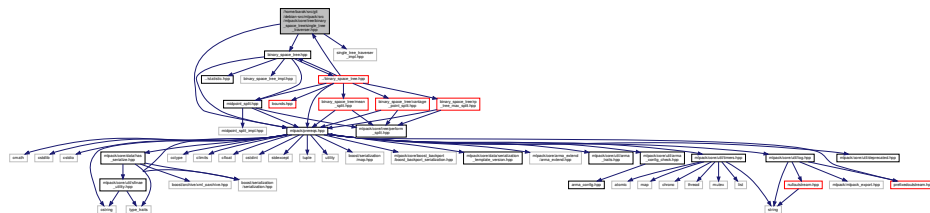
Mikhail Lozhnikov

Definition of class (RPTreeMaxSplit) to split a binary space partition tree.

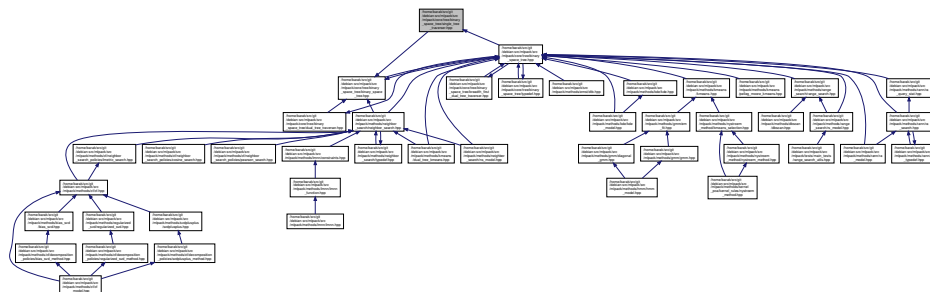
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.276 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/single_↵
_tree_traverser.hpp File Reference

Include dependency graph for single_tree_traverser.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- **class BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::SingleTree**↔
Traverser< RuleType >

A single-tree traverser for binary space trees; see `single_tree_traverser.hpp` for implementation.

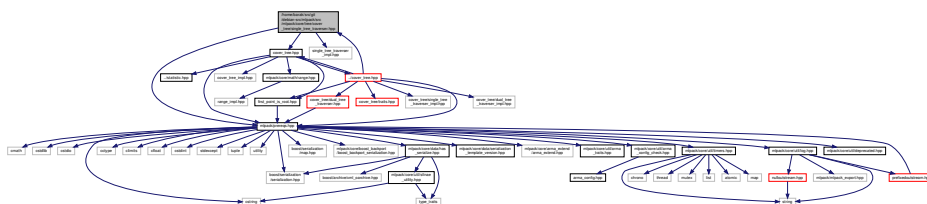
Namespaces

- **mlpack**
.hpp
- **mlpack::tree**

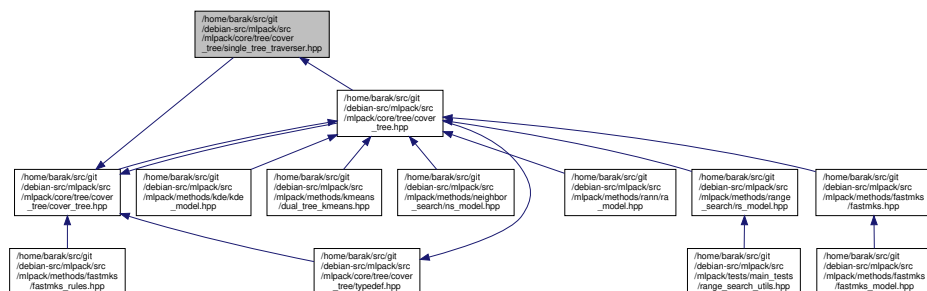
Trees and tree-building procedures.

traverser.hpp File Reference

Include dependency graph for single tree traverser.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **CoverTree**< **MetricType**, **StatisticType**, **MatType**, **RootPointPolicy** >::**SingleTreeTraverser**< **RuleType** >

A single-tree cover tree traverser; see `single_tree_traverser.hpp` for implementation.

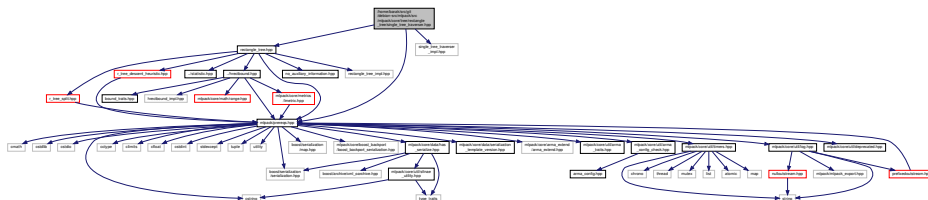
Namespaces

- **mlpack**
 .hpp
- **mlpack::tree**

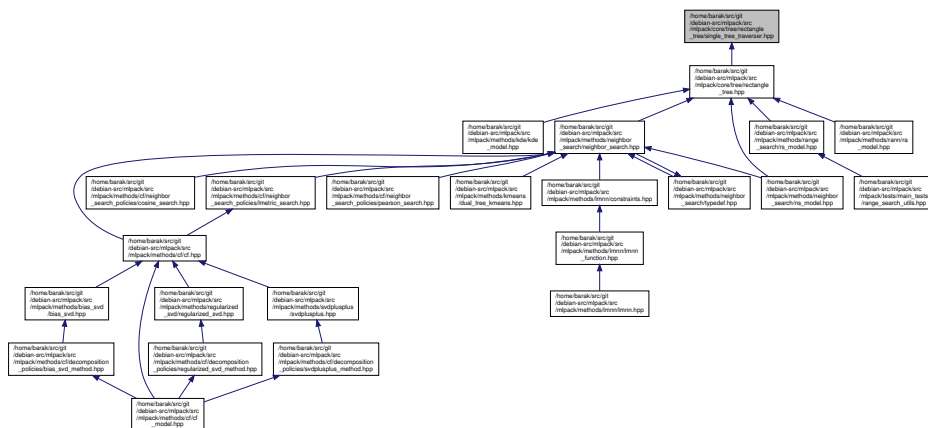
Trees and tree-building procedures.

40.279 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/single_tree_traverser.hpp File Reference

Include dependency graph for single_tree_traverser.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RectangleTree**< **MetricType**, **StatisticType**, **MatType**, **SplitType**, **DescentType**, **Auxiliary**< **InformationType** >::**SingleTreeTraverser**< **RuleType** >

A single traverser for rectangle type trees.

Namespaces

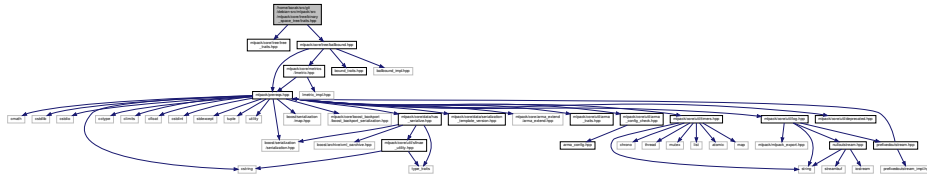
- **mlpack**
 .hpp
- **mlpack::tree**

Trees and tree-building procedures.

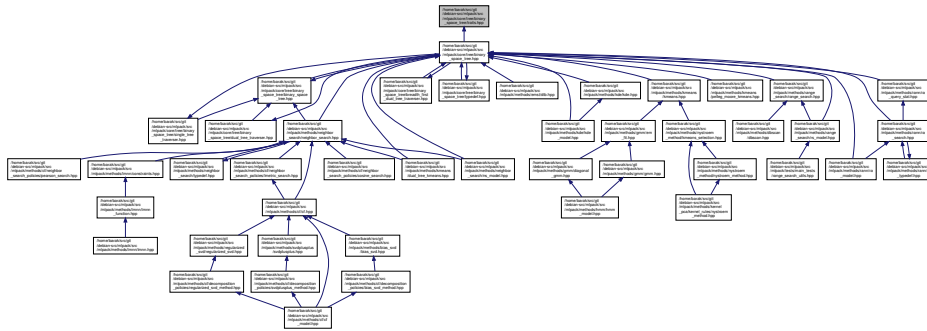
40.280 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/traits.hpp

File Reference

Include dependency graph for traits.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **TreeTraits**< **BinarySpaceTree**< **MetricType**, **StatisticType**, **MatType**, bound::BallBound, SplitType > >

This is a specialization of the TreeType class to the BallTree tree type.

- class **TreeTraits**< **BinarySpaceTree**< **MetricType**, **StatisticType**, **MatType**, bound::CellBound, SplitType > >

This is a specialization of the TreeType class to the UBTree tree type.

- class **TreeTraits**< **BinarySpaceTree**< **MetricType**, **StatisticType**, **MatType**, bound::HollowBallBound, SplitType > >

This is a specialization of the TreeType class to an arbitrary tree with HollowBallBound (currently only the vantage point tree is supported).

- class **TreeTraits**< **BinarySpaceTree**< **MetricType**, **StatisticType**, **MatType**, BoundType, RPTreeMaxSplit > >

This is a specialization of the TreeType class to the max-split random projection tree.

- class **TreeTraits**< **BinarySpaceTree**< **MetricType**, **StatisticType**, **MatType**, BoundType, RPTreeMeanSplit > >

This is a specialization of the TreeType class to the mean-split random projection tree.

- class **TreeTraits**< **BinarySpaceTree**< **MetricType**, **StatisticType**, **MatType**, BoundType, SplitType > >

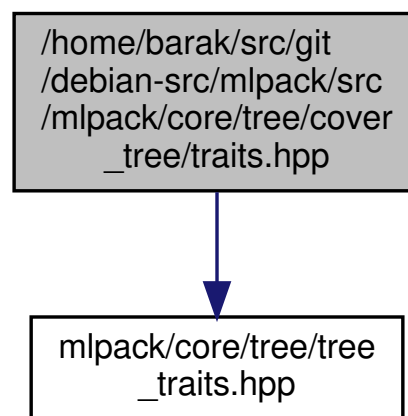
This is a specialization of the TreeTraits (p. 2302) class to the BinarySpaceTree (p. 1998) tree type.

Namespaces

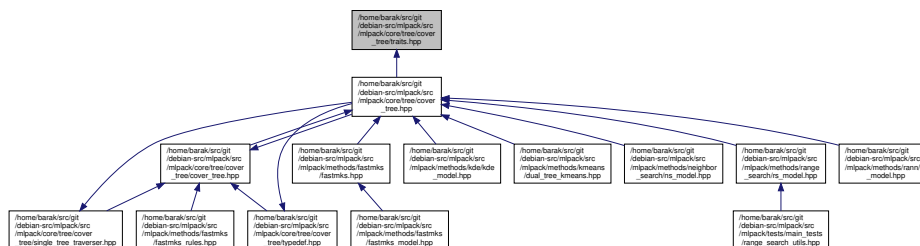
- mlpack**
 .hpp
- mlpack::tree**
 Trees and tree-building procedures.

40.281 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/traits.hpp File Reference

Include dependency graph for traits.hpp:



This graph shows which files directly or indirectly include this file:



Classes

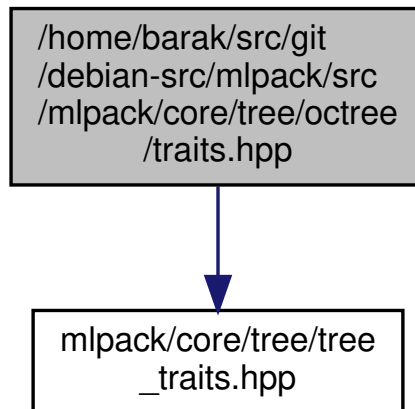
- class **TreeTraits**< **CoverTree**< **MetricType**, **StatisticType**, **MatType**, **RootPointPolicy** > >
 *The specialization of the **TreeTraits** (p. 2302) class for the **CoverTree** (p. 2040) tree type.*

Namespaces

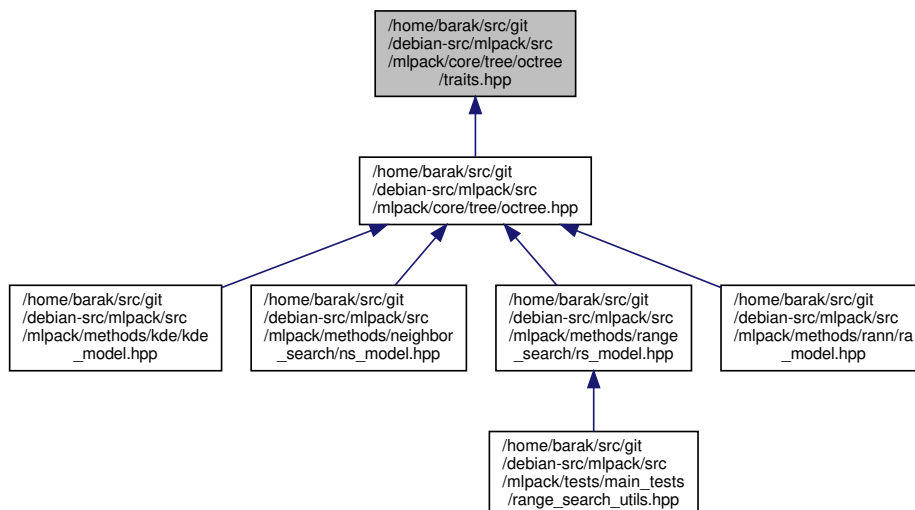
- **mlpack**
 .hpp
- **mlpack::tree**
 Trees and tree-building procedures.

40.282 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree/traits.hpp File Reference

Include dependency graph for traits.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **TreeTraits**< **RectangleTree**< **MetricType**, **StatisticType**, **MatType**, **RPlusTreeSplit**< **SplitPolicyType**, **SweepType** >, **DescentType**, **AuxiliaryInformationType** > >

Since the R+/R++ tree can not have overlapping children, we should define traits for the R+/R++ tree.

- class **TreeTraits**< **RectangleTree**< **MetricType**, **StatisticType**, **MatType**, **SplitType**, **DescentType**, **AuxiliaryInformationType** > >

*This is a specialization of the TreeType class to the **RectangleTree** (p. 2207) tree type.*

Namespaces

- **mlpack**

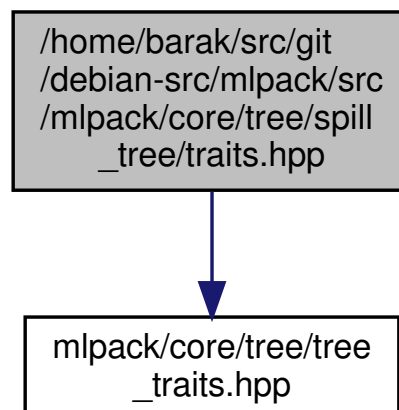
.hpp

- **mlpack::tree**

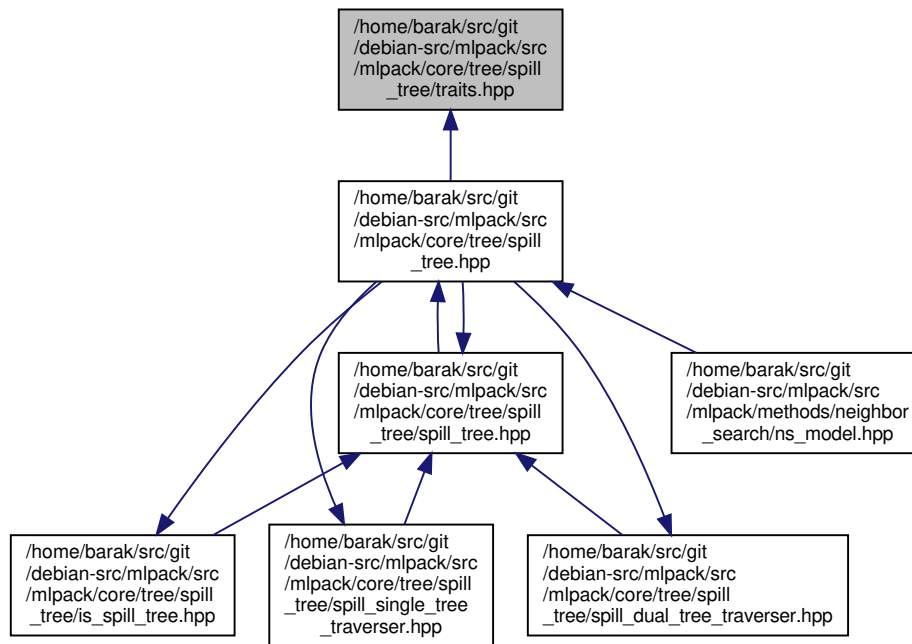
Trees and tree-building procedures.

40.284 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/traits.hpp File Reference

Include dependency graph for traits.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **TreeTraits** < **SpillTree** < **MetricType**, **StatisticType**, **MatType**, **HyperplaneType**, **SplitType** > >

*This is a specialization of the `TreeType` class to the **SpillTree** (p. 2274) tree type.*

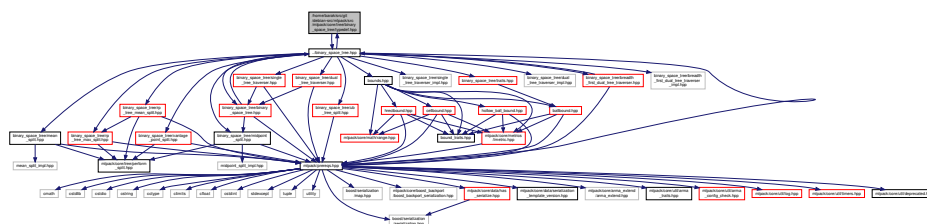
Namespaces

- **mlpack**
 .hpp
- **mlpack::tree**

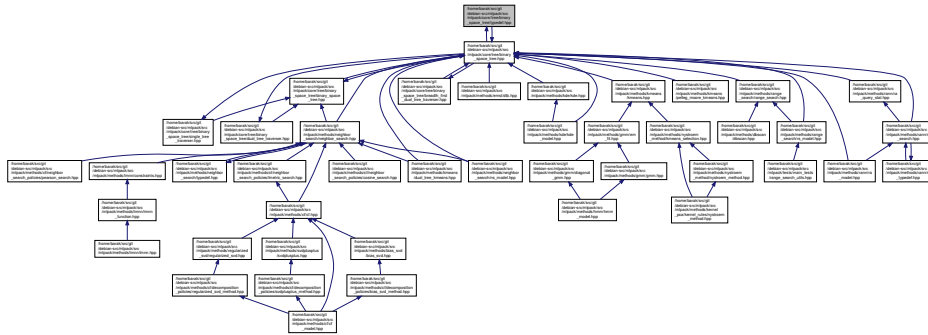
Trees and tree-building procedures.

40.285 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/typedef.hpp
File Reference

Include dependency graph for typedef.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
.hpp
- **mlpack::tree**
Trees and tree-building procedures.

Typedefs

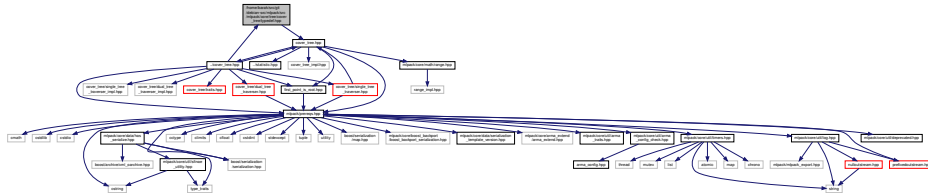
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using BallTree = BinarySpaceTree< MetricType, StatisticType, MatType, bound::BallBound, MidpointSplit >`
A midpoint-split ball tree.
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using KDTree = BinarySpaceTree< MetricType, StatisticType, MatType, bound::HRectBound, MidpointSplit >`
The standard midpoint-split kd-tree.
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using MaxRPTree = BinarySpaceTree< MetricType, StatisticType, MatType, bound::HRectBound, RPTreeMaxSplit >`
A max-split random projection tree.
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using MeanSplitBallTree = BinarySpaceTree< MetricType, StatisticType, MatType, bound::BallBound, MeanSplit >`
A mean-split ball tree.
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using MeanSplitKDTree = BinarySpaceTree< MetricType, StatisticType, MatType, bound::HRectBound, MeanSplit >`
A mean-split kd-tree.
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using RPTree = BinarySpaceTree< MetricType, StatisticType, MatType, bound::HRectBound, RPTreeMeanSplit >`
A mean-split random projection tree.
- `template<typename MetricType , typename StatisticType , typename MatType >`
`using UBTree = BinarySpaceTree< MetricType, StatisticType, MatType, bound::CellBound, UBTreeSplit >`
The Universal B-tree.

- `template<typename MetricType , typename StatisticType , typename MatType >`
`using VPTree = BinarySpaceTree< MetricType, StatisticType, MatType, bound::HollowBallBound, VPTreeSplit >`
- `template<typename BoundType , typename MatType = arma::mat>`
`using VPTreeSplit = VantagePointSplit< BoundType, MatType, 100 >`

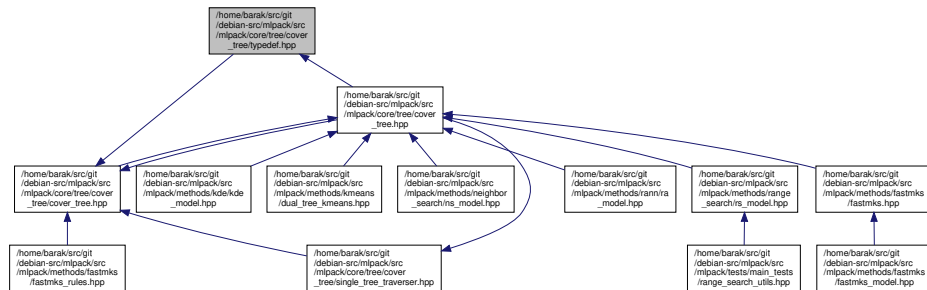
The vantage point tree (which is also called the metric tree).

40.286 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/typedef.hpp File Reference

Include dependency graph for typedef.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
`.hpp`
- **mlpack::tree**
Trees and tree-building procedures.

Typedefs

- `template<typename MetricType , typename StatisticType , typename MatType >`
`using StandardCoverTree = CoverTree< MetricType, StatisticType, MatType, FirstPointIsRoot >`
The standard cover tree, as detailed in the original cover tree paper:

- `template<typename MetricType , typename StatisticType , typename MatType >`
`using RPlusPlusTree = RectangleTree< MetricType, StatisticType, MatType, RPlusTreeSplit< RPlusPlusTree<`
`SplitPolicy, MinimalSplitsNumberSweep >, RPlusPlusTreeDescentHeuristic, RPlusPlusTreeAuxiliaryInformation`
`>`

The R++ tree, a variant of the R+ tree with maximum bounding rectangles.

- `template<typename MetricType , typename StatisticType , typename MatType >`
`using RPlusTree = RectangleTree< MetricType, StatisticType, MatType, RPlusTreeSplit< RPlusTreeSplitPolicy,`
`MinimalCoverageSweep >, RPlusTreeDescentHeuristic, NoAuxiliaryInformation >`

The R+ tree, a variant of the R tree that avoids overlapping rectangles.

- `template<typename MetricType , typename StatisticType , typename MatType >`
`using RStarTree = RectangleTree< MetricType, StatisticType, MatType, RStarTreeSplit, RStarTreeDescent<`
`Heuristic, NoAuxiliaryInformation >`

The R-tree, a more recent variant of the R tree.*

- `template<typename MetricType , typename StatisticType , typename MatType >`
`using RTree = RectangleTree< MetricType, StatisticType, MatType, RTreeSplit, RTreeDescentHeuristic, No<`
`AuxiliaryInformation >`

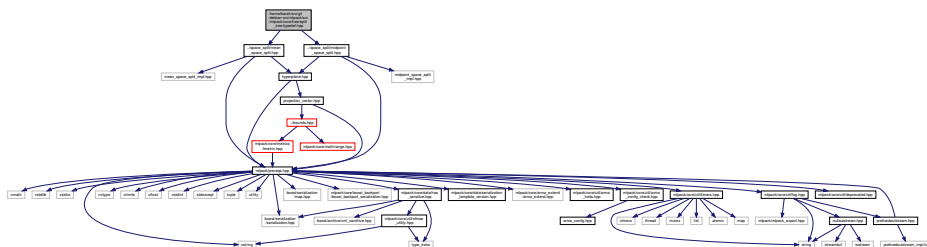
An implementation of the R tree that satisfies the TreeType policy API.

- `template<typename MetricType , typename StatisticType , typename MatType >`
`using XTree = RectangleTree< MetricType, StatisticType, MatType, XTreeSplit, RTreeDescentHeuristic, XTree<`
`AuxiliaryInformation >`

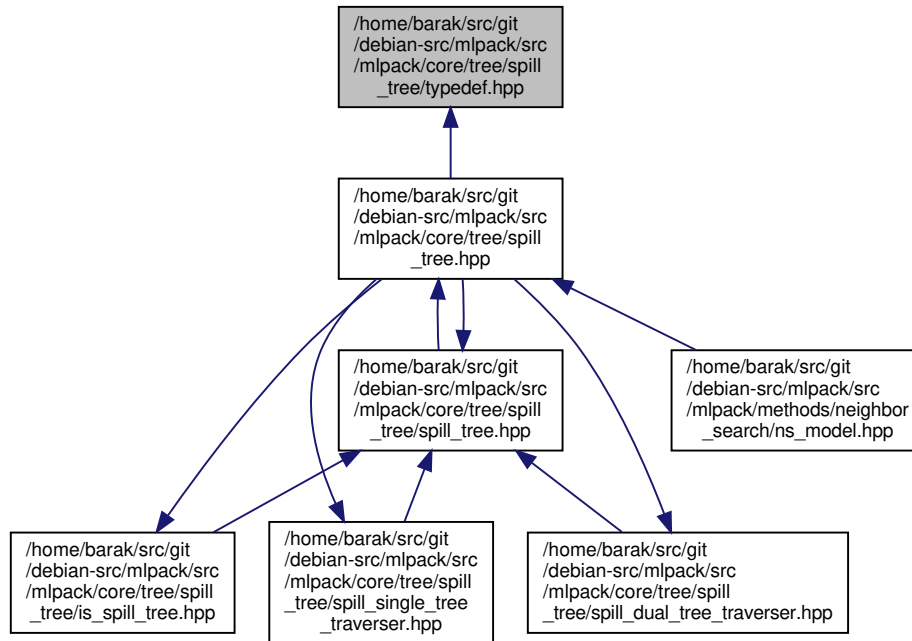
The X-tree, a variant of the R tree with supernodes.

40.288 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/typedef.hpp File Reference

Include dependency graph for typedef.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

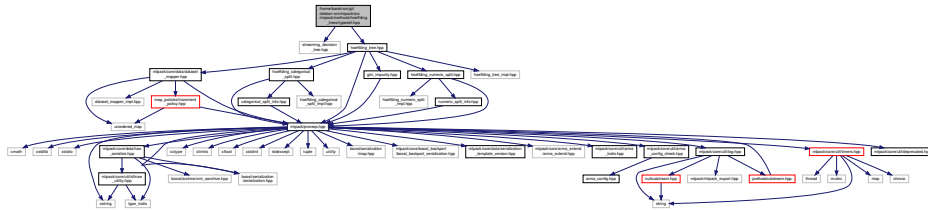
- **mlpack**
 .hpp
- **mlpack::tree**
 Trees and tree-building procedures.

Typedefs

- `template<typename MetricType , typename StatisticType , typename MatType >`
 using **MeanSPTree** = SpillTree< MetricType, StatisticType, MatType, AxisOrthogonalHyperplane, MeanSpaceSplit >
 A mean-split hybrid spill tree.
- `template<typename MetricType , typename StatisticType , typename MatType >`
 using **NonOrtMeanSPTree** = SpillTree< MetricType, StatisticType, MatType, Hyperplane, MeanSpaceSplit >
 A mean-split hybrid spill tree considering general splitting hyperplanes (not necessarily axis-orthogonal).
- `template<typename MetricType , typename StatisticType , typename MatType >`
 using **NonOrtSPTree** = SpillTree< MetricType, StatisticType, MatType, Hyperplane, MidpointSpaceSplit >
 A hybrid spill tree considering general splitting hyperplanes (not necessarily axis-orthogonal).
- `template<typename MetricType , typename StatisticType , typename MatType >`
 using **SPTree** = SpillTree< MetricType, StatisticType, MatType, AxisOrthogonalHyperplane, MidpointSpaceSplit >
 The hybrid spill tree.

40.289 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/typedef.hpp File Reference

Include dependency graph for typedef.hpp:



Namespaces

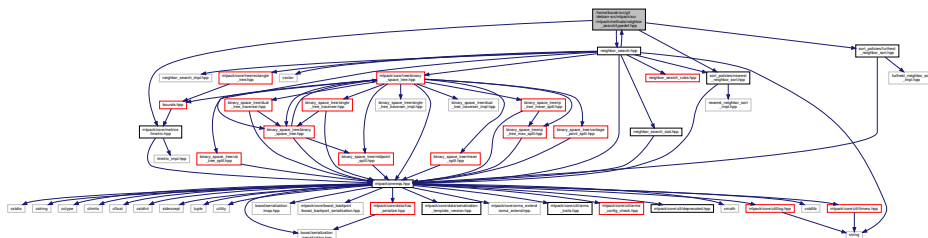
- **mlpack**
 .hpp
- **mlpack::tree**
 Trees and tree-building procedures.

Typedefs

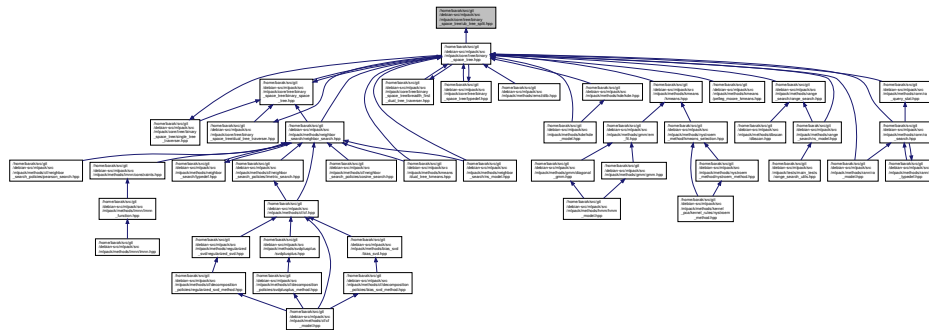
- typedef StreamingDecisionTree< HoeffdingTree<> > **HoeffdingTreeType**

40.290 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/typedef.hpp File Reference

Include dependency graph for typedef.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **UBTreeSplit**< **BoundType**, **MatType** >

Split a node into two parts according to the median address of points contained in the node.

Namespaces

- **mlpack**

.hpp

- **mlpack::tree**

Trees and tree-building procedures.

40.291.1 Detailed Description

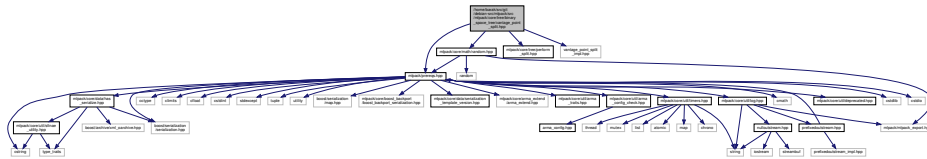
Author

Mikhail Lozhnikov

Definition of UBTreeSplit, a class that splits the space according to the median address of points contained in the node.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Include dependency graph for vantage_point_split.hpp:



- class **VantagePointSplit**< **BoundType**, **MatType**, **MaxNumSamples** >
The class splits a binary space partitioning tree node according to the median distance to the vantage point.
- struct **VantagePointSplit**< **BoundType**, **MatType**, **MaxNumSamples** >::**SplitInfo**
A struct that contains an information about the split.

- **mlpack**
.hpp
- **mlpack::tree**
Trees and tree-building procedures.

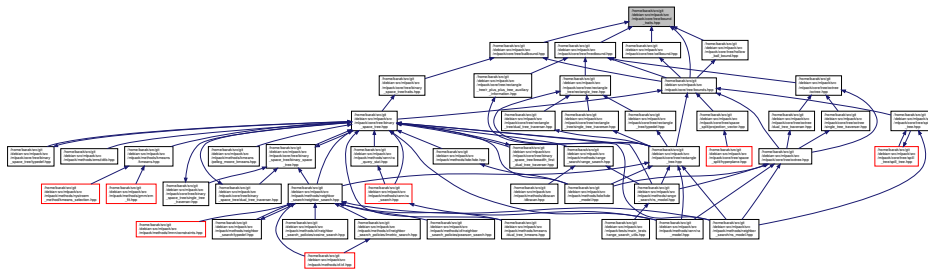
Author

Definition of class `VantagePointSplit`, a class that splits a vantage point tree into two parts using the distance to a certain vantage point.

Generated by Doxygen

40.293 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/bound_traits.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct **BoundTraits**< **BoundType** >
A class to obtain compile-time traits about BoundType classes.

Namespaces

- **mlpack**
.hpp
- **mlpack::bound**

40.293.1 Detailed Description

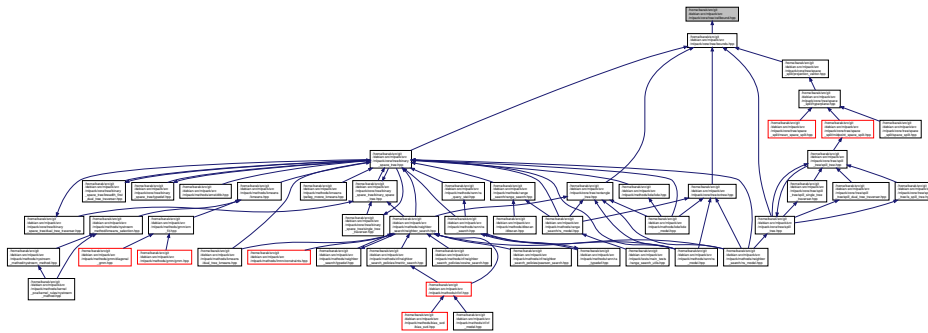
Author

Ryan Curtin

A class for template metaprogramming traits for bounds.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

This graph shows which files directly or indirectly include this file:



Classes

- struct **BoundTraits**< **CellBound**< **MetricType**, **ElemType** > >
- class **CellBound**< **MetricType**, **ElemType** >

The *CellBound* (p. 1006) class describes a bound that consists of a number of hyperrectangles.

Namespaces

- **mlpack**
- **mlpack::bound**

40.295.1 Detailed Description

Author

Mikhail Lozhnikov

Definition of the CellBound class. The class describes a bound that consists of a number of hyperrectangles. These hyperrectangles do not overlap each other. The bound is limited by an outer hyperrectangle and two addresses, the lower address and the high address. Thus, the bound contains all points included between the lower and the high addresses.

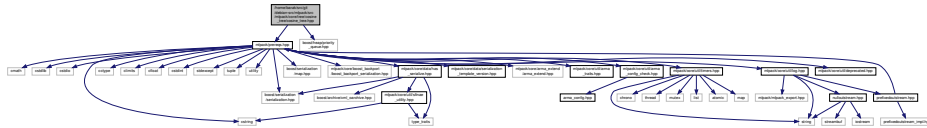
The notion of addresses is described in the following paper.

```
@inproceedings{bayer1997,
  author = {Bayer, Rudolf},
  title = {The Universal B-Tree for Multidimensional Indexing: General
    Concepts},
  booktitle = {Proceedings of the International Conference on Worldwide
    Computing and Its Applications},
  series = {WWCA '97},
  year = {1997},
  isbn = {3-540-63343-X},
  pages = {198--209},
  numpages = {12},
  publisher = {Springer-Verlag},
  address = {London, UK, UK},
}
```

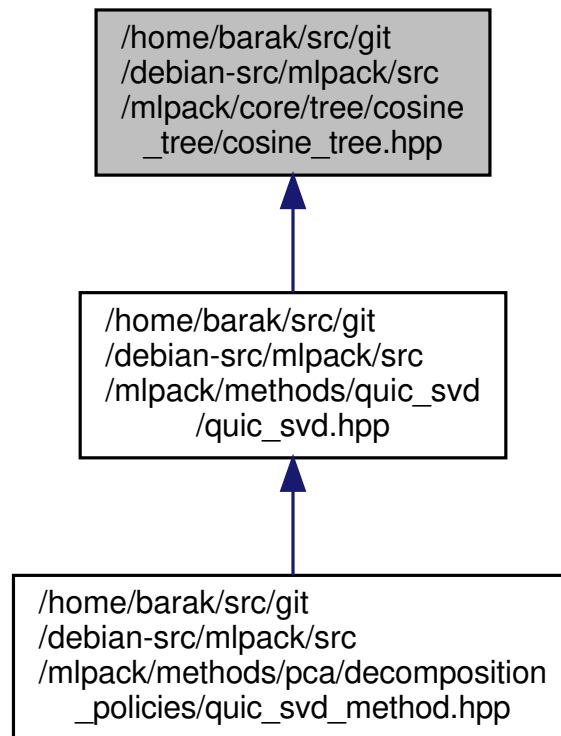
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.296 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cosine_tree/cosine_tree.hpp File Reference

Include dependency graph for cosine_tree.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **CompareCosineNode**
- class **CosineTree**

Namespaces

- **mlpack**
.*.hpp*
- **mlpack::tree**
Trees and tree-building procedures.

Typedefs

- `typedef boost::heap::priority_queue< CosineTree *, boost::heap::compare< CompareCosineNode > > CosineNodeQueue`

40.296.1 Detailed Description

Author

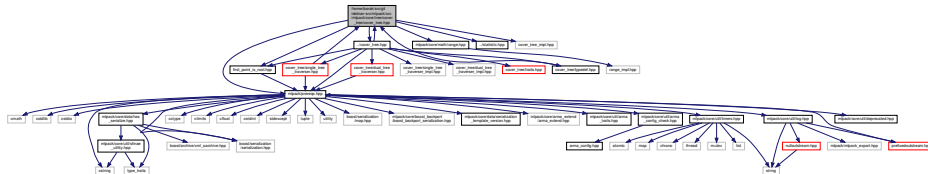
Siddharth Agrawal

Definition of Cosine Tree.

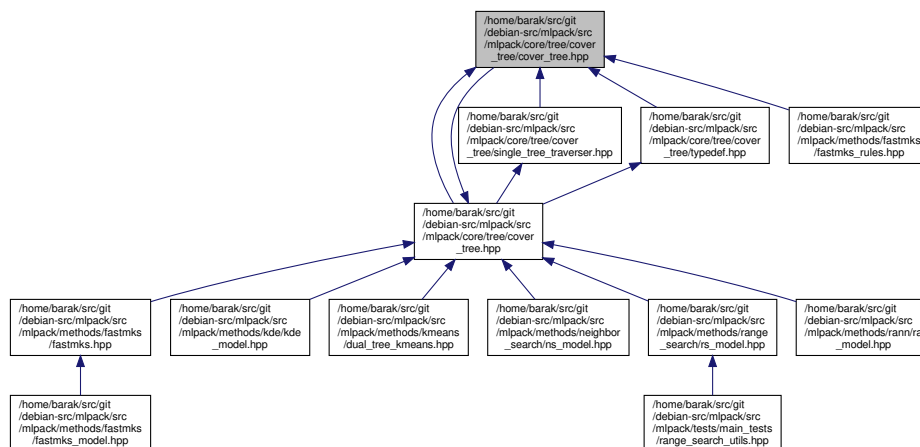
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.297 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/cover_↵
tree.hpp File Reference

Include dependency graph for cover_tree.hpp:



This graph shows which files directly or indirectly include this file:

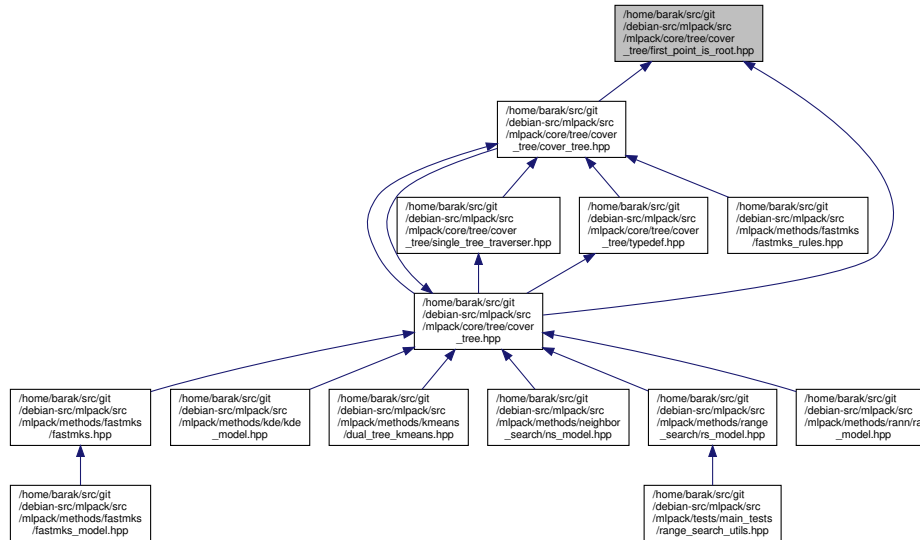


40.299 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree/first_point_is_root.hpp File Reference

Include dependency graph for first_point_is_root.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **FirstPointIsRoot**

This class is meant to be used as a choice for the policy class `RootPointPolicy` of the `CoverTree` (p. 2040) class.

Namespaces

- mlpack**
.hpp
- mlpack::tree**

Trees and tree-building procedures.

40.299.1 Detailed Description

Author

Ryan Curtin

A very simple policy for the cover tree; the first point in the dataset is chosen as the root of the cover tree.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.300 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/enumerate_tree.hpp File Reference

Namespaces

- **mlpack**
.hpp
- **mlpack::tree**
Trees and tree-building procedures.
- **mlpack::tree::enumerate**

Functions

- `template<class TreeType , class Walker >`
`void EnumerateTree (TreeType *tree, Walker &walker)`
Traverses all nodes of the tree, including the inner ones.
- `template<class TreeType , class Walker >`
`void EnumerateTreeImpl (TreeType *tree, Walker &walker, bool root)`

40.300.1 Detailed Description

Author

Ivan (Jonan) Georgiev

This file contains function that performs a simple depth-first walk on the tree calling `Enter` and `Leave` methods of a provided walker.

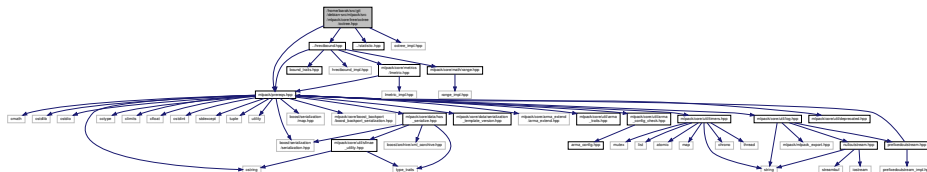
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

- **mlpack**
.hpp
- **mlpack::bound**
- **mlpack::bound::meta**
Metaprogramming utilities.

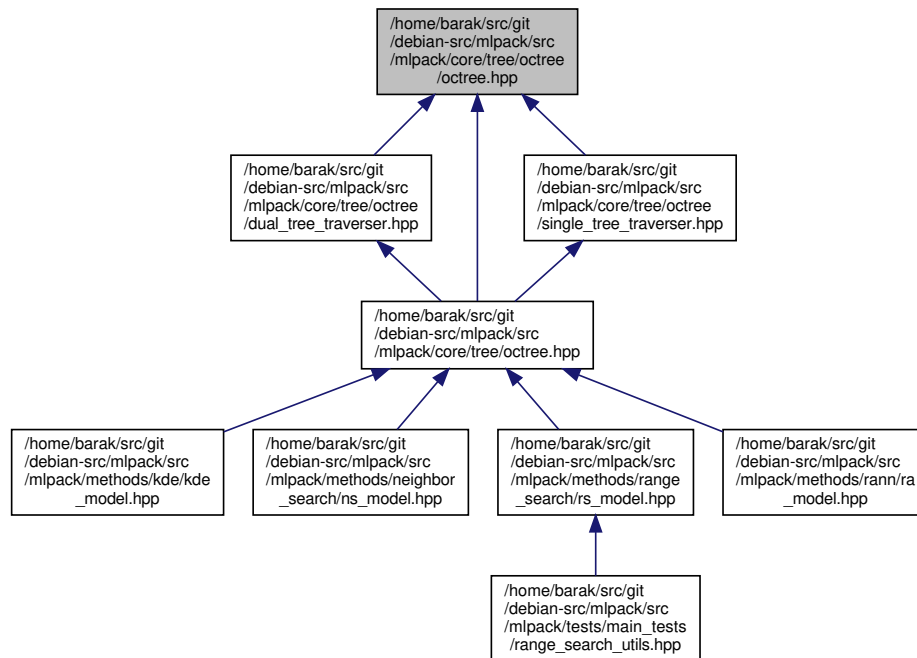
Bounds that are useful for binary space partitioning trees.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpac. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Include dependency graph for octree.hpp:



This graph shows which files directly or indirectly include this file:



Classes

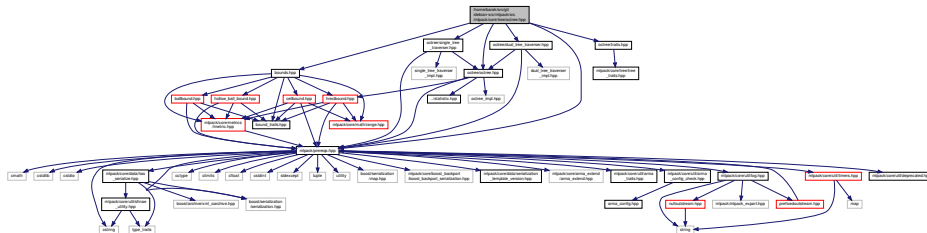
- class **Octree**< **MetricType**, **StatisticType**, **MatType** >
- class **Octree**< **MetricType**, **StatisticType**, **MatType** >::**DualTreeTraverser**< **MetricType**, **StatisticType**, **MatType** >
A dual-tree traverser; see dual_tree_traverser.hpp.
- class **Octree**< **MetricType**, **StatisticType**, **MatType** >::**SingleTreeTraverser**< **RuleType** >
A single-tree traverser; see single_tree_traverser.hpp.
- struct **Octree**< **MetricType**, **StatisticType**, **MatType** >::**SplitType**::**SplitInfo**

Namespaces

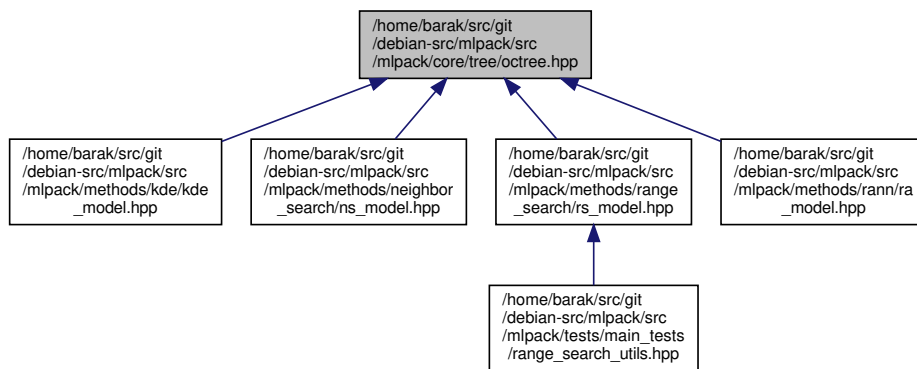
- **mlpack**
.hpp
- **mlpack::tree**
Trees and tree-building procedures.

40.306 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/octree.hpp File Reference

Include dependency graph for octree.hpp:

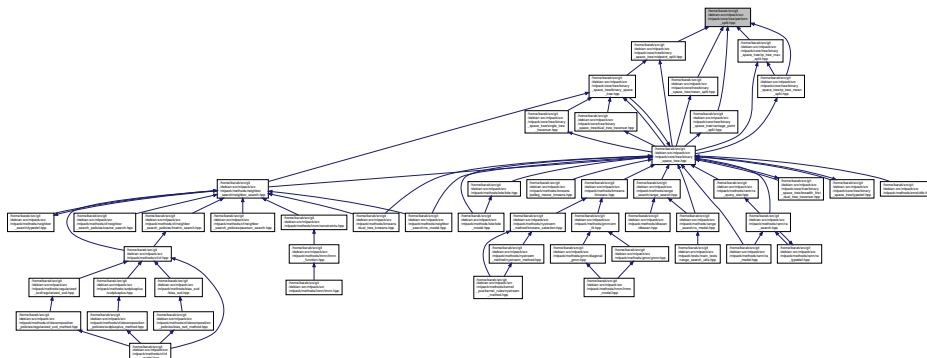


This graph shows which files directly or indirectly include this file:

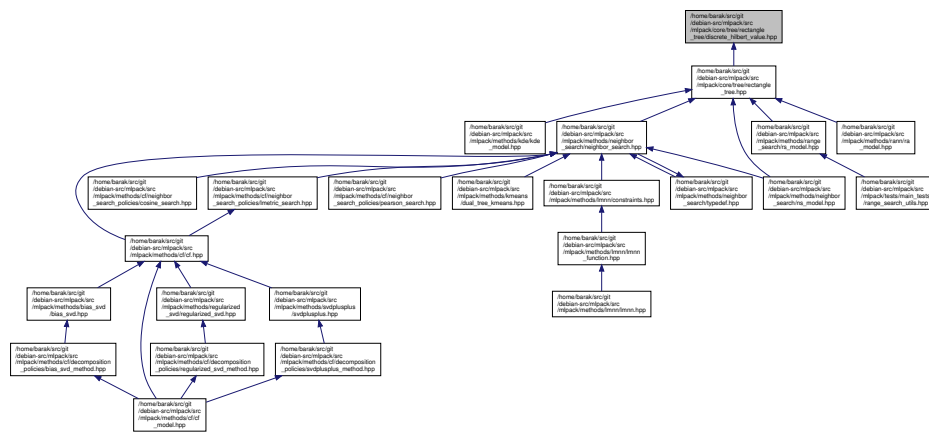


40.307 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/perform_split.hpp File Reference

This graph shows which files directly or indirectly include this file:



This graph shows which files directly or indirectly include this file:



Classes

- class **DiscreteHilbertValue**< **TreeElemType** >

The **DiscreteHilbertValue** (p. 2079) class stores Hilbert values for all of the points in a **RectangleTree** (p. 2207) node, and calculates Hilbert values for new points.

Namespaces

- **mlpack**
 .hpp
- **mlpack::tree**

Trees and tree-building procedures.

40.308.1 Detailed Description

Author

Mikhail Lozhnikov

Definition of the DiscreteHilbertValue class, a class that calculates the ordering of points using the Hilbert curve.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.309.1 Detailed Description

Author

Mikhail Lozhnikov

Definition of the HilbertRTreeAuxiliaryInformation class, a class that provides some Hilbert r-tree specific information about the nodes.

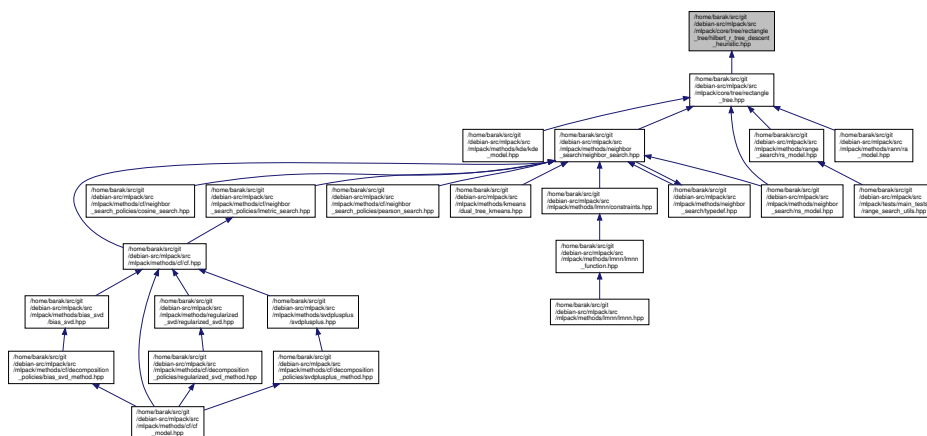
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.310 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/hilbert_r_tree_descent_heuristic.hpp File Reference

Include dependency graph for hilbert_r_tree_descent_heuristic.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **HilbertRTreeDescentHeuristic**

This class chooses the best child of a node in a Hilbert R tree when inserting a new point.

Namespaces

- **mlpack**
 - .hpp
- **mlpack::tree**
 - Trees and tree-building procedures.

40.310.1 Detailed Description

Author

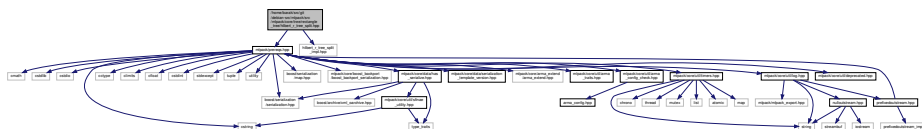
Mikhail Lozhnikov

Definition of HilbertRTreeDescentHeuristic, a class that chooses the best child of a node in an R tree when inserting a new point.

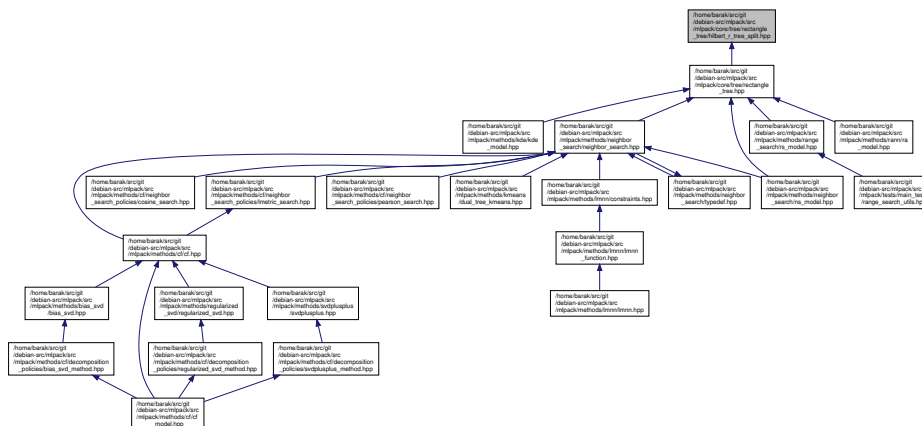
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.311 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/hilbert_r_tree_split.hpp File Reference

Include dependency graph for hilbert_r_tree_split.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **HilbertRTreeSplit**< **splitOrder** >
The splitting procedure for the Hilbert R tree.

Namespaces

- **mlpack**
.hpp
- **mlpack::tree**
Trees and tree-building procedures.

40.311.1 Detailed Description

Author

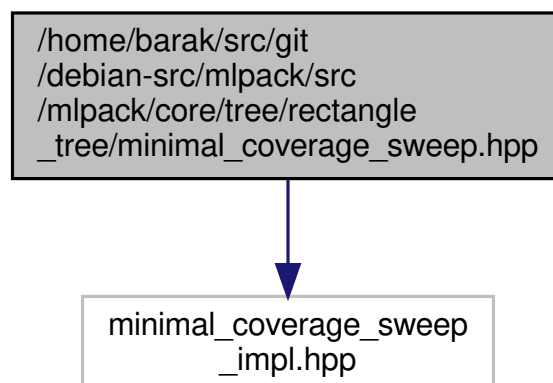
Mikhail Lozhnikov

Definition of the HilbertRTreeSplit class, a class that splits the nodes of an R tree, starting at a leaf node and moving upwards if necessary.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.312 [/home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/minimal_coverage_sweep.hpp](#) File Reference

Include dependency graph for minimal_coverage_sweep.hpp:



- class **MinimalCoverageSweep**< **SplitPolicy** >

- `struct MinimalCoverageSweep< SplitPolicy >::SweepCost< TreeType >`

- **mlpack**
.hpp
- **mlpack::tree**
Trees and tree-building procedures.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

- **mlpack**
.hpp
- **mlpack::tree**
Trees and tree-building procedures.

Author

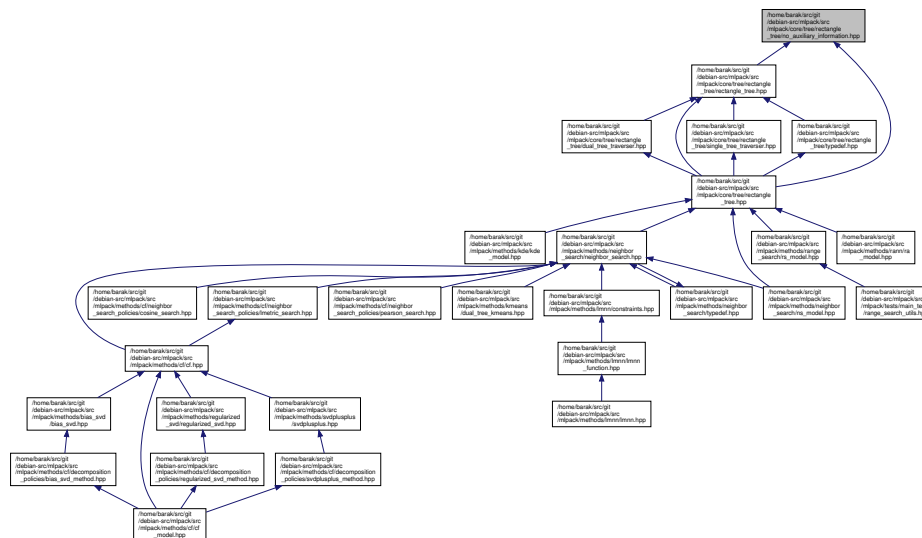
Mikhail Lozhnikov

Definition of the MinimalSplitsNumberSweep class, a class that finds a partition of a node along an axis.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.314 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/no_↵
auxiliary_information.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **NoAuxiliaryInformation**< **TreeType** >

[illegible]

- class **RPlusPlusTreeAuxiliaryInformation**< **TreeType** >

- **mlpack**
.hpp
- **mlpack::tree**

Trees and tree-building procedures.

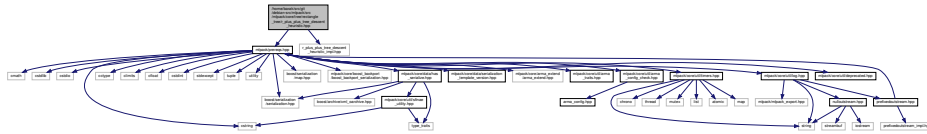
Author

Mikhail Lozhnikov

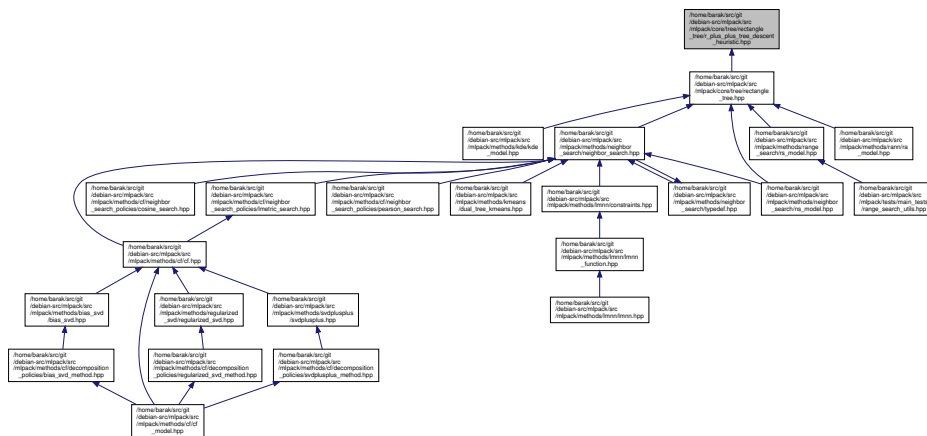
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.316 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/r_plus_plus_tree_descent_heuristic.hpp File Reference

Include dependency graph for r_plus_plus_tree_descent_heuristic.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RPlusPlusTreeDescentHeuristic**

Namespaces

- **mlpack**
 .hpp
- **mlpack::tree**
 Trees and tree-building procedures.

40.316.1 Detailed Description

Author

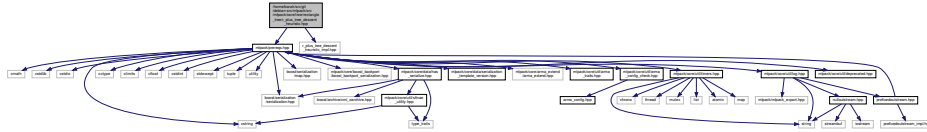
Mikhail Lozhnikov

Definition of RPlusPlusTreeDescentHeuristic, a class that chooses the best child of a node in an R++ tree when inserting a new point.

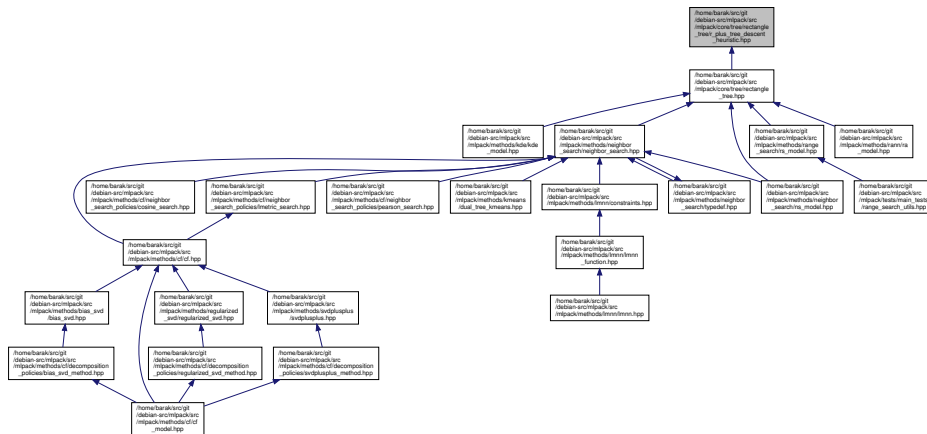
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.318 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/r_plus_tree_descent_heuristic.hpp File Reference

Include dependency graph for r_plus_tree_descent_heuristic.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RPlusTreeDescentHeuristic**

Namespaces

- mlpack**
 .hpp
- mlpack::tree**
Trees and tree-building procedures.

40.318.1 Detailed Description

Author

Mikhail Lozhnikov

Definition of RPlusTreeDescentHeuristic, a class that chooses the best child of a node in an R+ tree when inserting a new point.

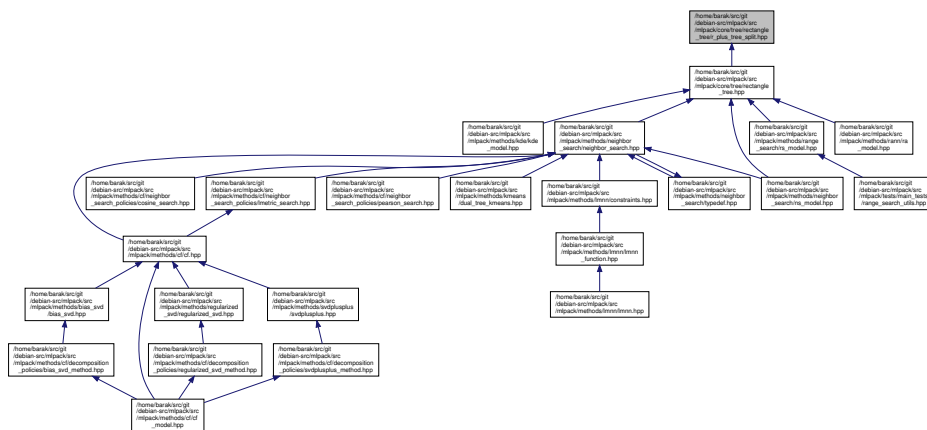
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.319 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/r_plus_tree_split.hpp File Reference

Include dependency graph for r_plus_tree_split.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RPlusTreeSplit**< **SplitPolicyType**, **SweepType** >

The *RPlusTreeSplit* (p. 2251) class performs the split process of a node on overflow.

Namespaces

- mlpack**
.
.hpp
- mlpack::tree**
Trees and tree-building procedures.

40.319.1 Detailed Description

Author

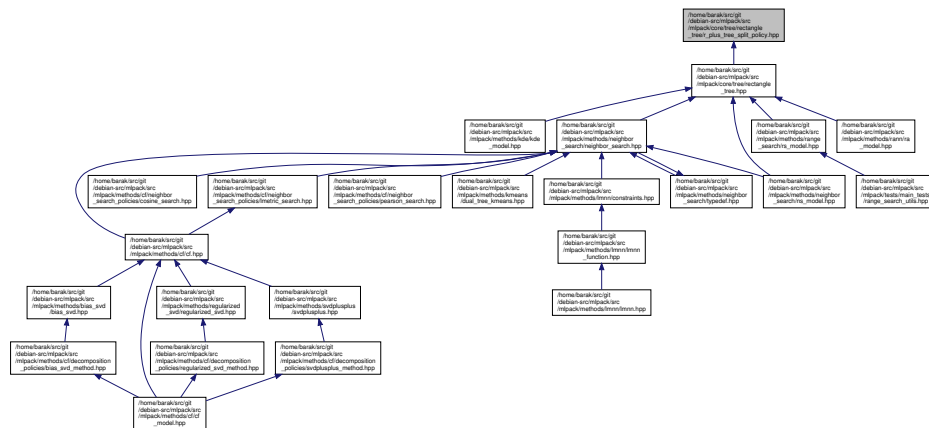
Mikhail Lozhnikov

Definition of the RPlusTreeSplit class, a class that splits the nodes of an R+ (or R++) tree, starting at a leaf node and moving upwards if necessary.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.320 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/r_plus_tree_split_policy.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **RPlusTreeSplitPolicy**

The **RPlusTreeSplitPolicy** (p. 2247) helps to determine the subtree into which we should insert a child of an intermediate node that is being split.

Namespaces

- mlpack**
- mlpack::tree**

Trees and tree-building procedures.

40.320.1 Detailed Description

Author

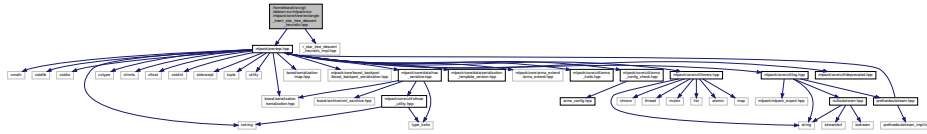
Mikhail Lozhnikov

Definition and implementation of the RPlusTreeSplitPolicy class, a class that helps to determine the subtree into which we should insert an intermediate node.

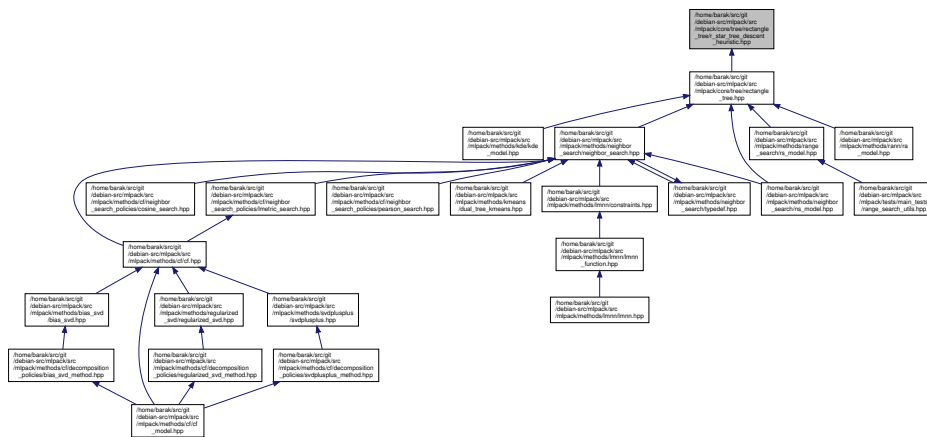
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.321 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/r_star_↵ tree_descent_heuristic.hpp File Reference

Include dependency graph for r_star_tree_descent_heuristic.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RStarTreeDescentHeuristic**

When descending a **RectangleTree** (p. 2207) to insert a point, we need to have a way to choose a child node when the point isn't enclosed by any of them.

Namespaces

- mlpack**
.hpp
- mlpack::tree**
Trees and tree-building procedures.

40.321.1 Detailed Description

Author

Andrew Wells

Definition of RStarTreeDescentHeuristic, a class that chooses the best child of a node in an R tree when inserting a new point.

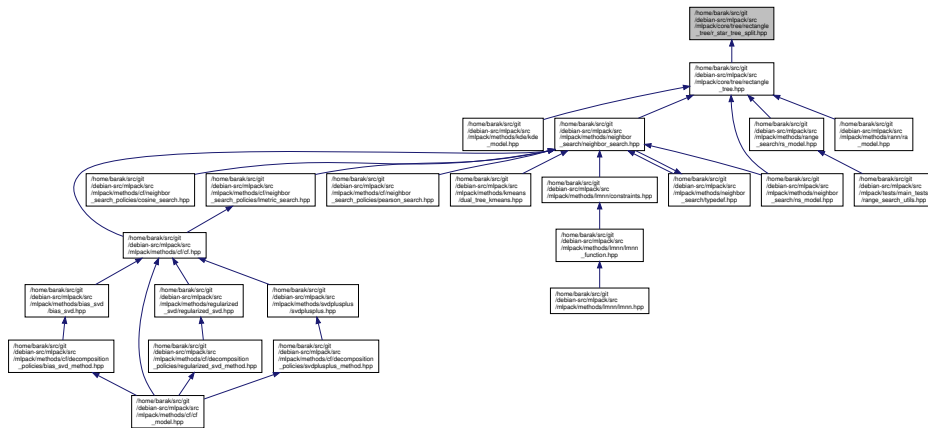
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.322 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree/r_star_↵
tree_split.hpp File Reference

Include dependency graph for r_star_tree_split.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RStarTreeSplit**
A Rectangle Tree has new points inserted at the bottom.

Namespaces

- **mlpack**
.hpp
- **mlpack::tree**
Trees and tree-building procedures.

Classes

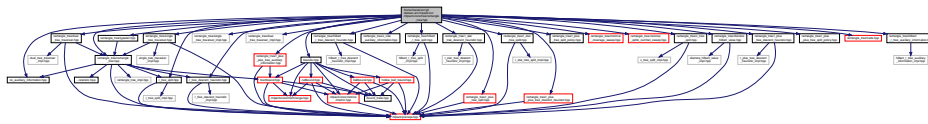
- class **RectangleTree**< **MetricType**, **StatisticType**, **MatType**, **SplitType**, **DescentType**, **Auxiliary**↔**InformationType** >
A rectangle type tree tree, such as an R-tree or X-tree.
- class **RectangleTree**< **MetricType**, **StatisticType**, **MatType**, **SplitType**, **DescentType**, **Auxiliary**↔**InformationType** >::**DualTreeTraverser**< **MetricType**, **StatisticType**, **MatType**, **SplitType**, **DescentType**, **AuxiliaryInformationType** >
A dual tree traverser for rectangle type trees.
- class **RectangleTree**< **MetricType**, **StatisticType**, **MatType**, **SplitType**, **DescentType**, **Auxiliary**↔**InformationType** >::**SingleTreeTraverser**< **RuleType** >
A single traverser for rectangle type trees.

Namespaces

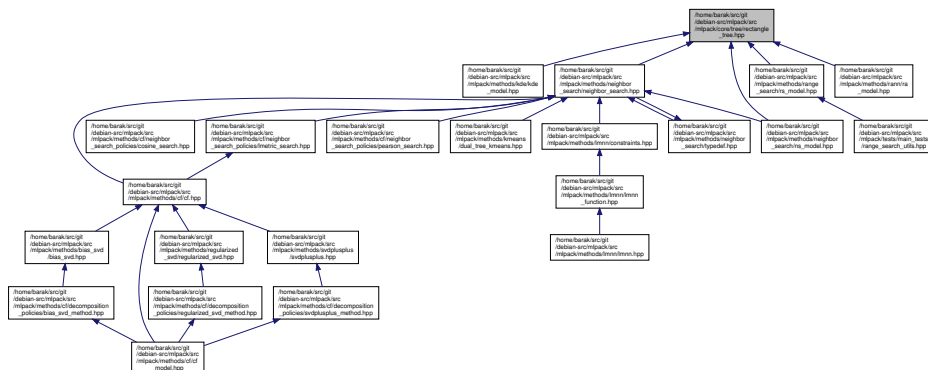
- mlpack**
.hpp
- mlpack::tree**
Trees and tree-building procedures.

40.326 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_tree.hpp File Reference

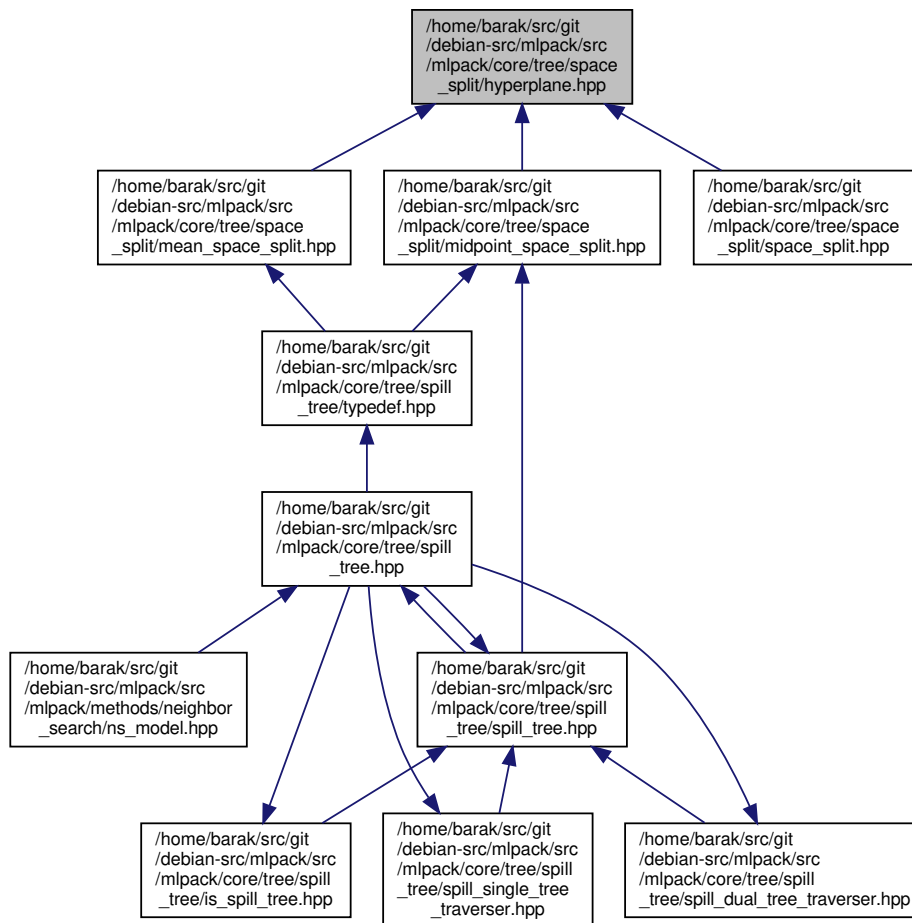
Include dependency graph for rectangle_tree.hpp:



This graph shows which files directly or indirectly include this file:



This graph shows which files directly or indirectly include this file:



Classes

- class **HyperplaneBase**< **BoundT**, **ProjVectorT** >
HyperplaneBase (p. 2133) defines a splitting hyperplane based on a projection vector and projection value.

Namespaces

- mlpack**
.hpp
- mlpack::tree**
Trees and tree-building procedures.

Typedefs

- template<typename MetricType >
using **AxisOrthogonalHyperplane** = HyperplaneBase< bound::HRectBound< MetricType >, AxisParallel< ProjVector >

AxisOrthogonalHyperplane represents a hyperplane orthogonal to an axis.

- `template<typename MetricType >`
`using Hyperplane = HyperplaneBase< bound::BallBound< MetricType >, ProjVector >`

Hyperplane represents a general hyperplane (not necessarily axis-orthogonal).

40.329.1 Detailed Description

Author

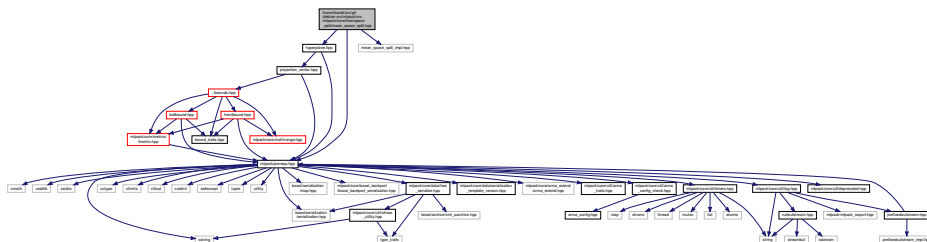
Marcos Pivadori

Definition of Hyperplane and AxisOrthogonalHyperplane.

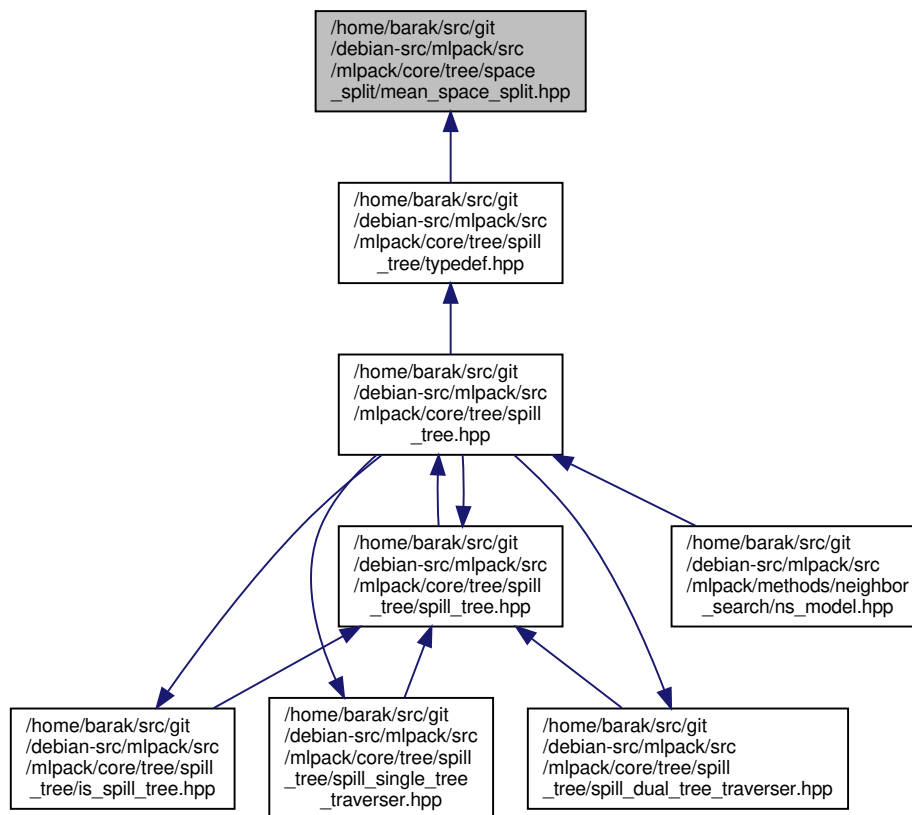
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.330 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/mean_↵ space_split.hpp File Reference

Include dependency graph for mean_space_split.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **MeanSpaceSplit**< **MetricType**, **MatType** >

Namespaces

- **mlpack**
.
.hpp
- **mlpack::tree**
Trees and tree-building procedures.

40.330.1 Detailed Description

Author

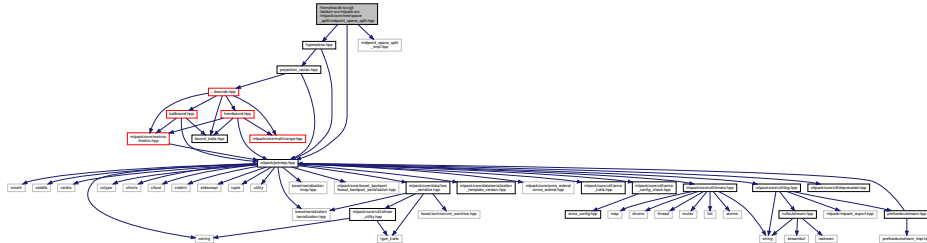
Marcos Pivodori

Definition of MeanSpaceSplit, to create a splitting hyperplane considering the mean of the values in a certain projection.

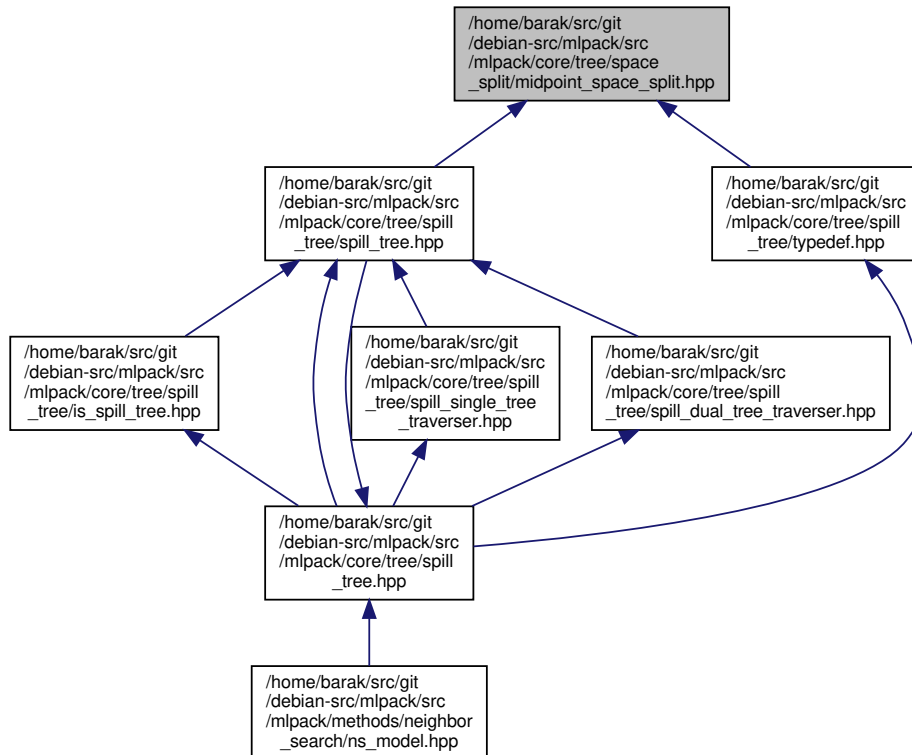
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.331 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/midpoint_↵ space_split.hpp File Reference

Include dependency graph for midpoint_space_split.hpp:



This graph shows which files directly or indirectly include this file:



Classes

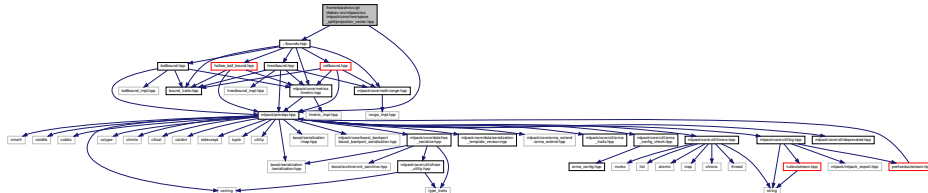
- class **MidpointSpaceSplit**< **MetricType**, **MatType** >

Namespaces

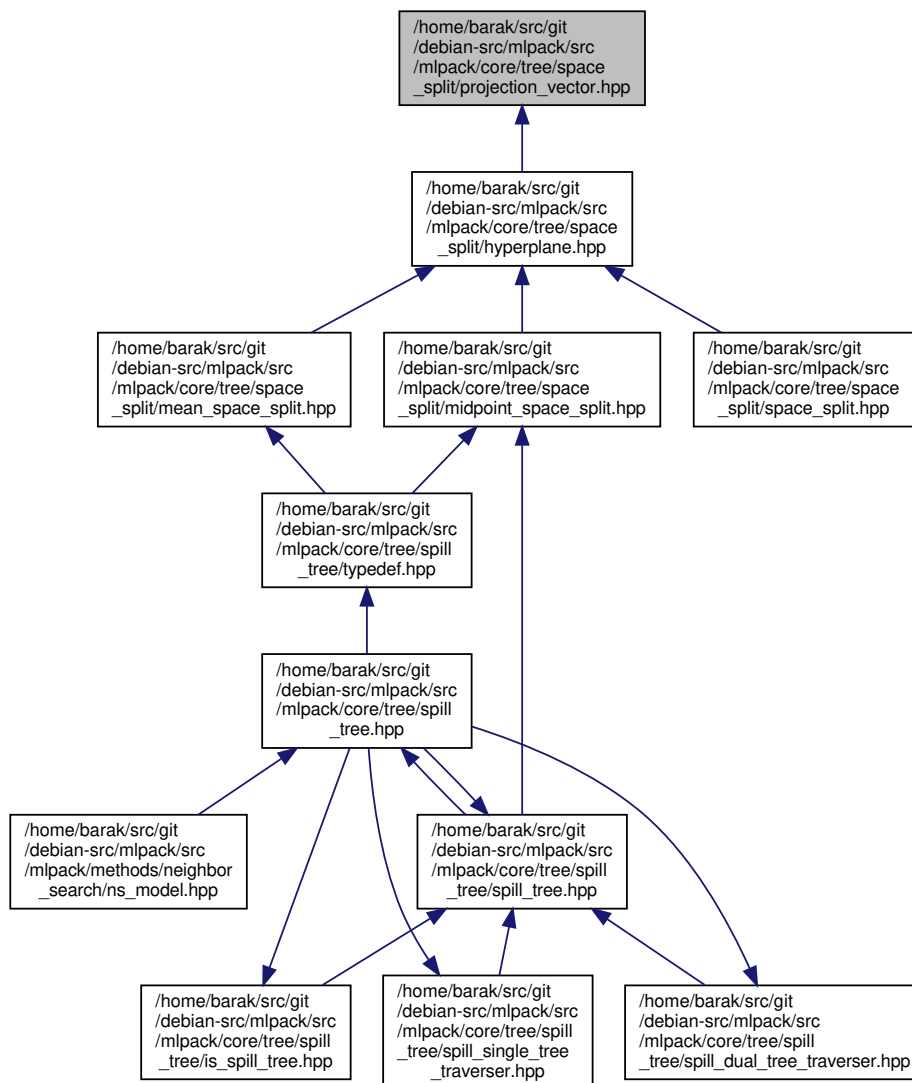
- **mlpack**
 .hpp
- **mlpack::tree**
 Trees and tree-building procedures.

40.332 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_split/projection_vector.hpp File Reference

Include dependency graph for projection_vector.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
.hpp
- **mlpack::tree**

Trees and tree-building procedures.

40.333.1 Detailed Description

Author

Marcos Pividori

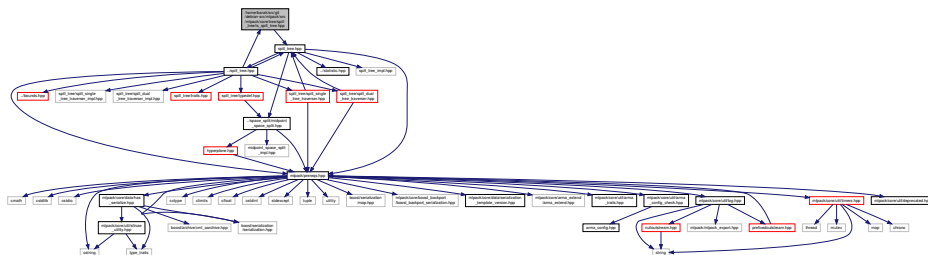
Definition of SpaceSplit, implementing some methods to create a projection vector based on a given set of points.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

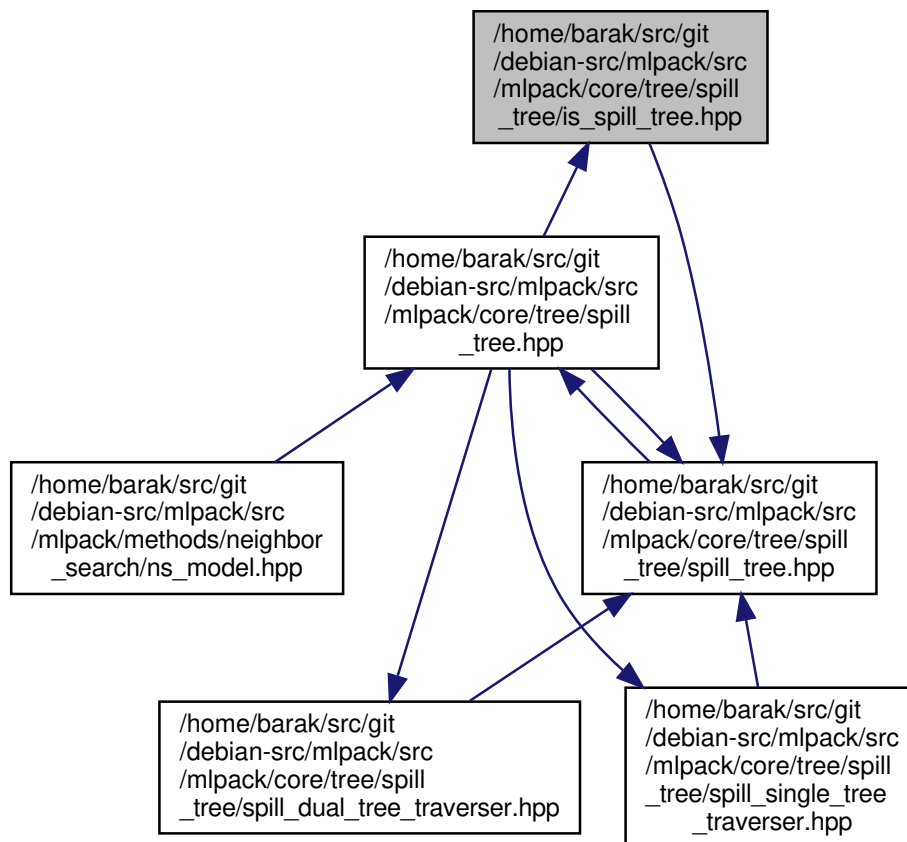
40.334 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/is_spill_↵
tree.hpp File Reference

Definition of IsSpillTree.

Include dependency graph for is_spill_tree.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct **IsSpillTree**< **TreeType** >
- struct **IsSpillTree**< **tree::SpillTree**< **MetricType**, **StatisticType**, **MatType**, **HyperplaneType**, **SplitType** > >

Namespaces

- **mlpack**
 .hpp
- **mlpack::tree**
 Trees and tree-building procedures.

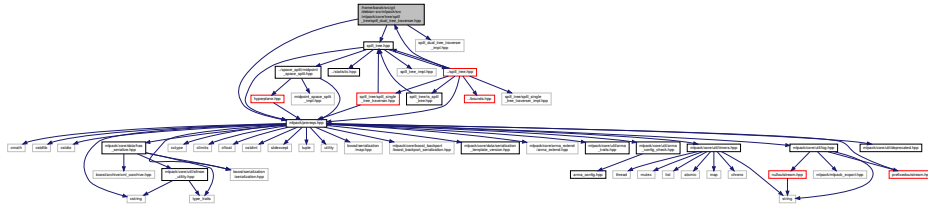
40.334.1 Detailed Description

Definition of IsSpillTree.

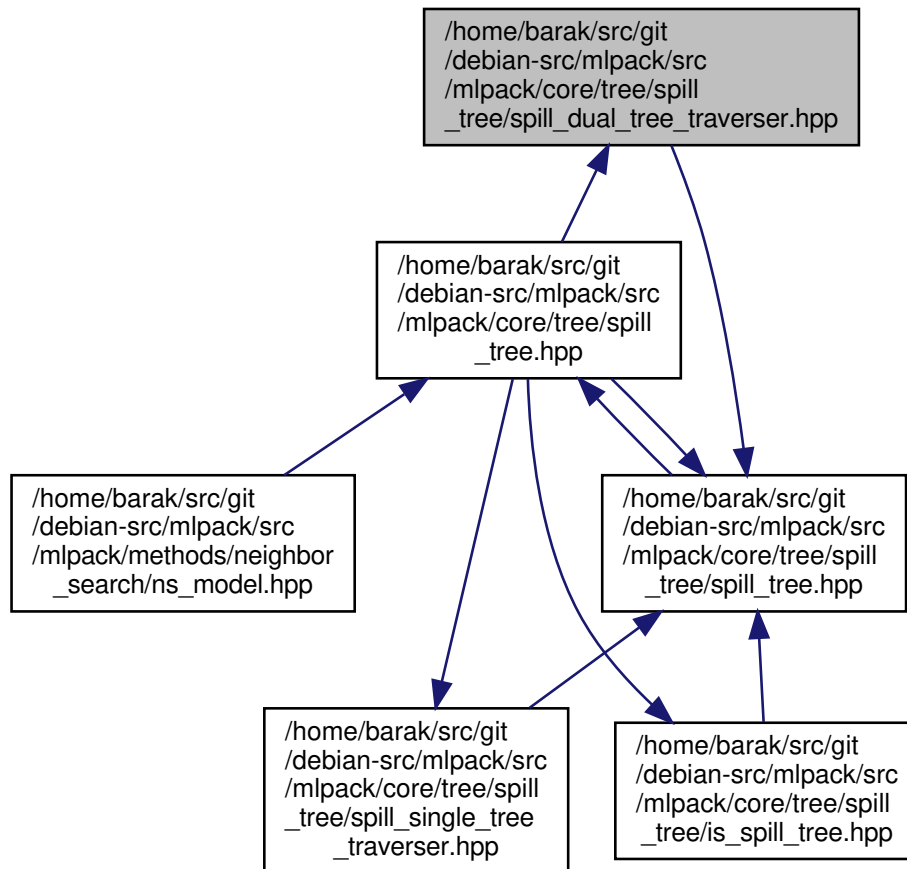
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.335 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/spill_dual_tree_traverser.hpp File Reference

Include dependency graph for spill_dual_tree_traverser.hpp:



This graph shows which files directly or indirectly include this file:

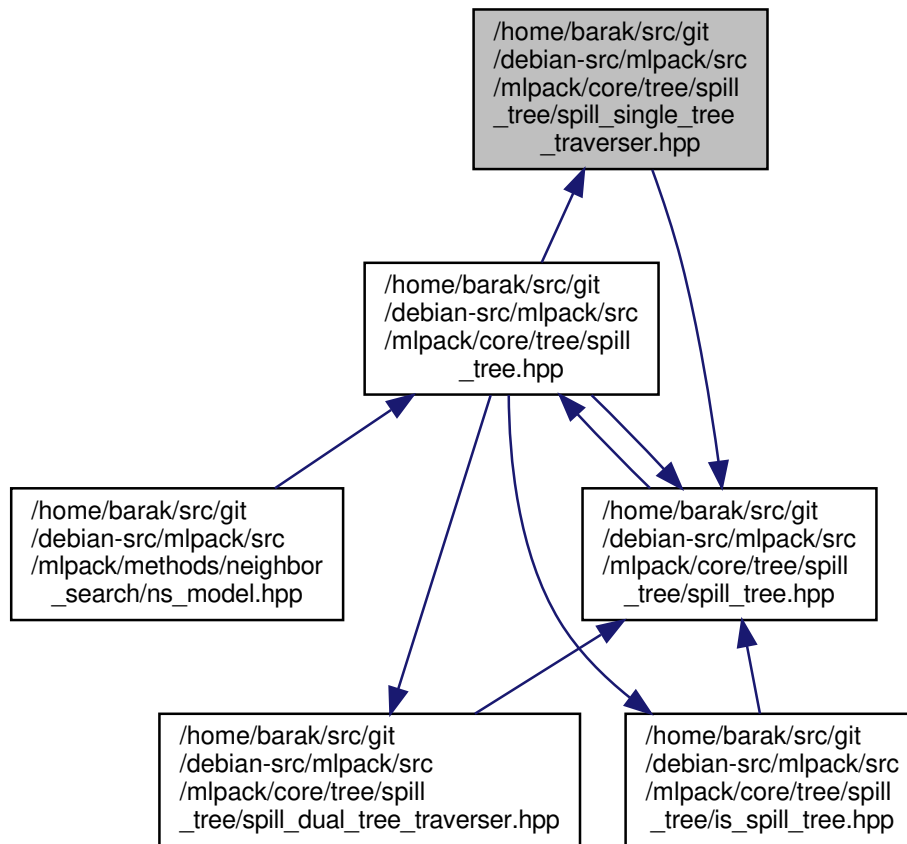


Classes

- class **SpillTree**< **MetricType**, **StatisticType**, **MatType**, **HyperplaneType**, **SplitType** >::**SpillDualTree**↔
Traverser< **MetricType**, **StatisticType**, **MatType**, **HyperplaneType**, **SplitType** >

*A generic dual-tree traverser for hybrid spill trees; see **spill_dual_tree_traverser.hpp** (p. 2683) for implementation.*

This graph shows which files directly or indirectly include this file:



Classes

- class **SpillTree**< **MetricType**, **StatisticType**, **MatType**, **HyperplaneType**, **SplitType** >::**SpillSingleTree**< **Traverser**< **MetricType**, **StatisticType**, **MatType**, **HyperplaneType**, **SplitType** >

*A generic single-tree traverser for hybrid spill trees; see **spill_single_tree_traverser.hpp** (p. 2684) for implementation.*

Namespaces

- **mlpack**
.hpp
- **mlpack::tree**

Trees and tree-building procedures.

40.336.1 Detailed Description

Author

Ryan Curtin
Marcos Pivadori

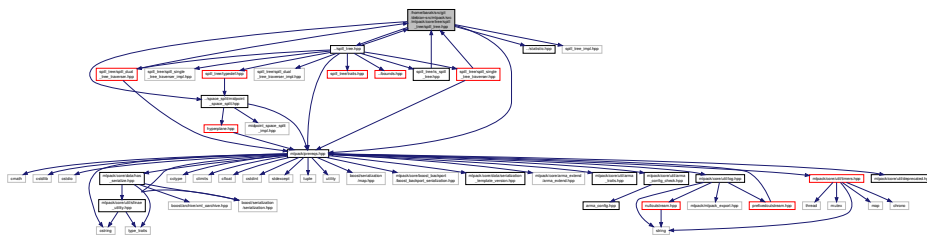
A nested class of SpillTree which traverses the entire tree with a given set of rules which indicate the branches which can be pruned and the order in which to recurse. This traverser is a depth-first traverser. The Defeatist template parameter determines if the traversers must do defeatist search on overlapping nodes.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

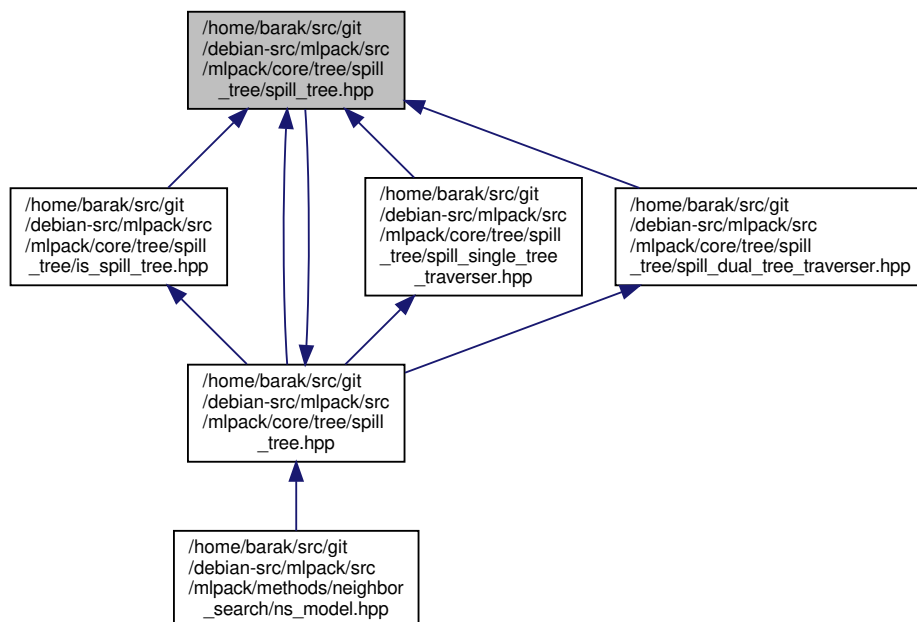
40.337 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree/spill_tree.hpp

File Reference

Include dependency graph for spill_tree.hpp:



This graph shows which files directly or indirectly include this file:



Classes

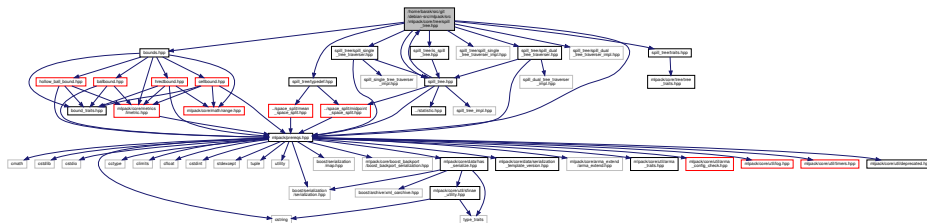
- class **SpillTree**< **MetricType**, **StatisticType**, **MatType**, **HyperplaneType**, **SplitType** >
A hybrid spill tree is a variant of binary space trees in which the children of a node can "spill over" each other, and contain shared datapoints.
- class **SpillTree**< **MetricType**, **StatisticType**, **MatType**, **HyperplaneType**, **SplitType** >::**SpillDualTree**↔
Traverser< **MetricType**, **StatisticType**, **MatType**, **HyperplaneType**, **SplitType** >
A generic dual-tree traverser for hybrid spill trees; see [spill_dual_tree_traverser.hpp](#) (p. 2683) for implementation.
- class **SpillTree**< **MetricType**, **StatisticType**, **MatType**, **HyperplaneType**, **SplitType** >::**SpillSingleTree**↔
Traverser< **MetricType**, **StatisticType**, **MatType**, **HyperplaneType**, **SplitType** >
A generic single-tree traverser for hybrid spill trees; see [spill_single_tree_traverser.hpp](#) (p. 2684) for implementation.

Namespaces

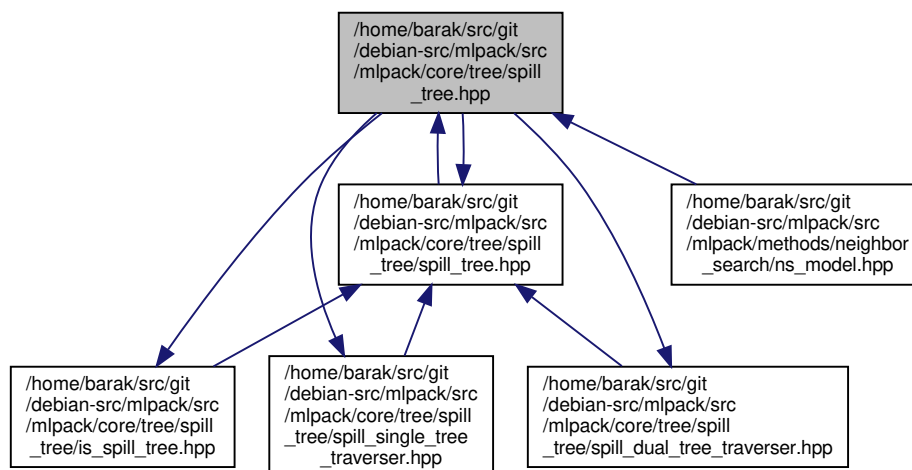
- mlpack**
.hpp
- mlpack::tree**
Trees and tree-building procedures.

40.338 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_tree.hpp File Reference

Include dependency graph for spill_tree.hpp:



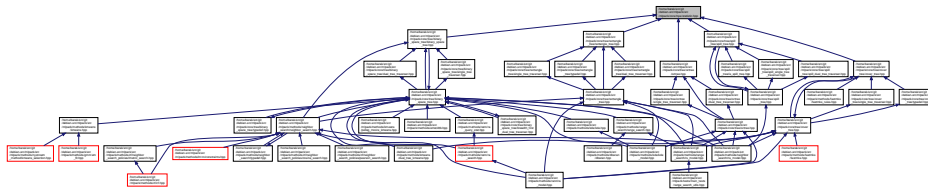
This graph shows which files directly or indirectly include this file:



40.339 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/statistic.hpp File Reference

Definition of the policy type for the statistic class.

This graph shows which files directly or indirectly include this file:



Classes

- class **EmptyStatistic**

Empty statistic if you are not interested in storing statistics in your tree.

Namespaces

- **mlpack**

.hpp

- **mlpack::tree**

Trees and tree-building procedures.

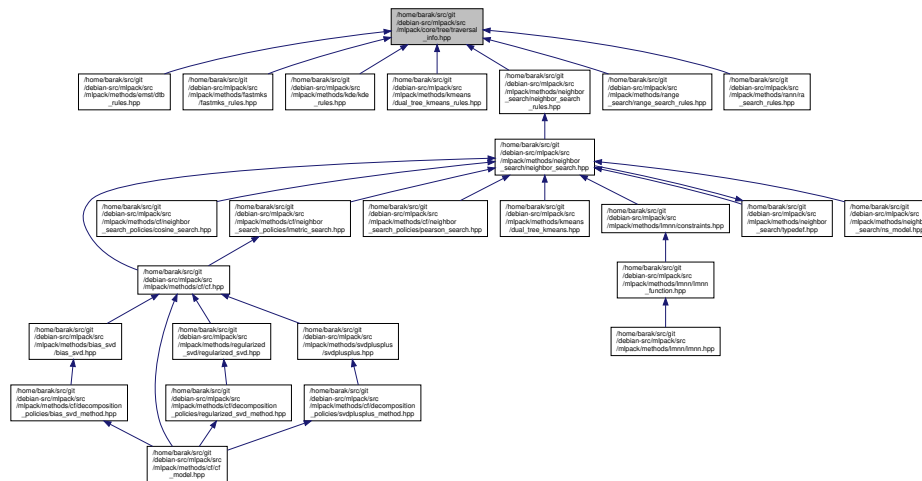
40.339.1 Detailed Description

Definition of the policy type for the statistic class.

You should define your own statistic that looks like EmptyStatistic.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.340 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/traversal_info.hpp File Reference



Classes

The **TraversalInfo** (p. 2299) class holds traversal information which is used in dual-tree (and single-tree) traversals.

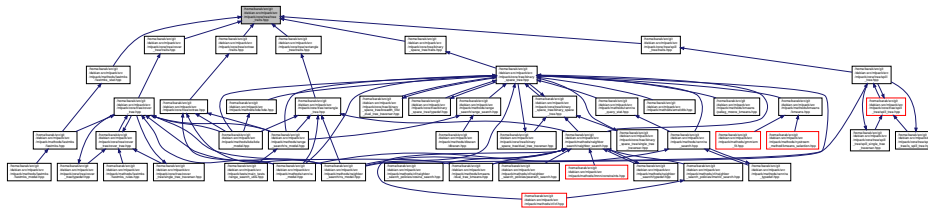
- **mlpack**
.hpp
- **mlpack::tree**
Trees and tree-building procedures.

Author

This class will hold the traversal information for dual-tree traversals. A dual-tree traversal should be updating the members of this class before `Score()` is called.

40.341 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/tree_traits.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **TreeTraits**< **TreeType** >

The **TreeTraits** (p. 2302) class provides compile-time information on the characteristics of a given tree type.

Namespaces

- **mlpack**
 .hpp
- **mlpack::tree**

Trees and tree-building procedures.

40.341.1 Detailed Description

Author

Ryan Curtin

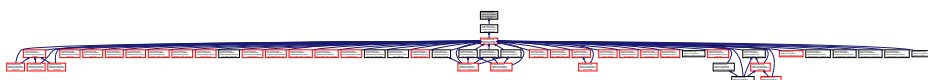
This file implements the basic, unspecialized TreeTraits class, which provides information about tree types. If you create a tree class, you should specialize this class with the characteristics of your tree.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.342 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/arma_config.hpp File Reference

This is an autogenerated file which contains the configuration of Armadillo at the time mlpack was built.

This graph shows which files directly or indirectly include this file:



Macros

- #define **MLPACK_ARMA_NO64BIT_WORD**
- #define **MLPACK_ARMA_USE_OPENMP**

40.342.1 Detailed Description

This is an autogenerated file which contains the configuration of Armadillo at the time mlpack was built.

If you modify anything in here by hand, your warranty is void, your house may catch fire, and we're not going to call the police when your program segfaults so hard that robbers come to your house and take everything you own. If you do decide, against better judgment, to modify anything at all in this file, and you are reporting a bug, be absolutely certain to mention that you've done something stupid in this file first.

In short: don't touch this file.

40.342.2 Macro Definition Documentation

40.342.2.1 MLPACK_ARMA_NO64BIT_WORD

```
#define MLPACK_ARMA_NO64BIT_WORD
```

Definition at line 18 of file arma_config.hpp.

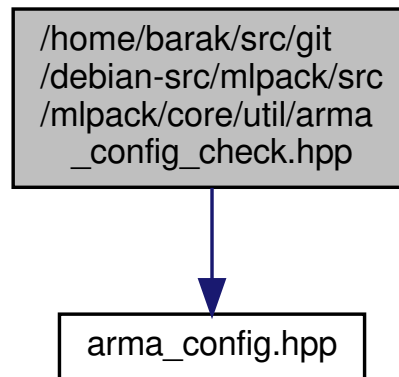
40.342.2.2 MLPACK_ARMA_USE_OPENMP

```
#define MLPACK_ARMA_USE_OPENMP
```

Definition at line 20 of file arma_config.hpp.

40.343 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/arma_config_check.hpp File Reference

Include dependency graph for arma_config_check.hpp:



This graph shows which files directly or indirectly include this file:



40.343.1 Detailed Description

Author

Ryan Curtin

Using the contents of **arma_config.hpp** (p.2690), try to catch the condition where the user has included mlpack with ARMA_64BIT_WORD enabled but mlpack was compiled without ARMA_64BIT_WORD enabled. This should help prevent a long, drawn-out debugging process where nobody can figure out why the stack is getting mangled.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.344 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/arma_traits.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct **IsVector**< **VecType** >
If value == true, then VecType is some sort of Armadillo vector or subview.
- struct **IsVector**< arma::Col< eT > >
- struct **IsVector**< arma::Row< eT > >
- struct **IsVector**< arma::SpCol< eT > >
- struct **IsVector**< arma::SpRow< eT > >
- struct **IsVector**< arma::SpSubview< eT > >
- struct **IsVector**< arma::subview_col< eT > >
- struct **IsVector**< arma::subview_row< eT > >

40.344.1 Detailed Description

Author

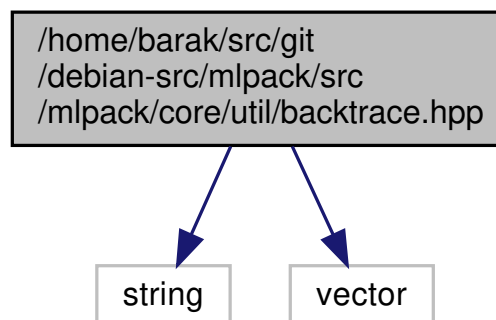
Ryan Curtin

Some traits used for template metaprogramming (SFINAE) with Armadillo types.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.345 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/backtrace.hpp File Reference

Include dependency graph for backtrace.hpp:



Classes

- class **Backtrace**
Provides a backtrace.

40.346.1 Detailed Description

Author

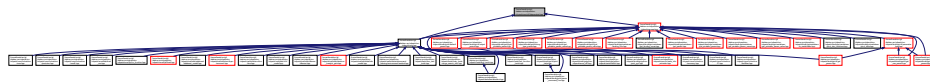
Matthew Amidon

This file implements the CLI subsystem which is intended to replace FX. This can be used more or less regardless of context. In the future, it might be expanded to include file I/O.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.347 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/deprecated.hpp File Reference

This graph shows which files directly or indirectly include this file:



Macros

- `#define mlpack_deprecated`

40.347.1 Detailed Description

Author

Marcos Pivadori.

Definition of the `mlpack_deprecated` macro.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.347.2 Macro Definition Documentation

40.347.2.1 mlpack_deprecated

```
#define mlpack_deprecated
```

Definition at line 22 of file deprecated.hpp.

Referenced by RegressionDistribution::Err(), and RegressionDistribution::LogProbability().

40.348 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/gitversion.hpp File Reference**Variables**

- return mlpack **git**

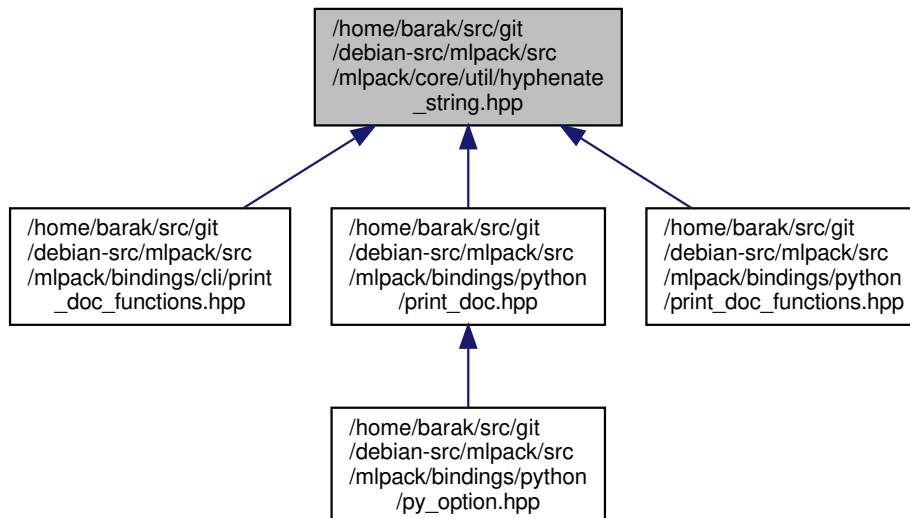
40.348.1 Variable Documentation**40.348.1.1 git**

```
return mlpack git
```

Definition at line 1 of file gitversion.hpp.

40.349 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/hyphenate_string.hpp File Reference

This graph shows which files directly or indirectly include this file:



- **mlpack**
 .hpp
- **mlpack::util**

- `std::string HyphenateString` (const `std::string` &str, int padding)
** Hyphenate a string or split it onto multiple 80-character lines, with some * amount of padding on each line.*

Ryan Curtin

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

```
graph TD; A["/home/barak/src/git  
/debian-src/mlpack/src  
/mlpack/core/util/is_std  
_vector.hpp"] --> B["vector"]
```

A diagram illustrating the file path for the `vector` header. A grey box at the top contains the path: `/home/barak/src/git`, `/debian-src/mlpack/src`, `/mlpack/core/util/is_std`, and `_vector.hpp`. A blue arrow points from this box to a white box at the bottom labeled `vector`.

[illegible]

Classes

- class **Log**
Provides a convenient way to give formatted output.

Namespaces

- mlpack**
.hpp

40.351.1 Detailed Description

Author

Matthew Amidon

Definition of the Log class.

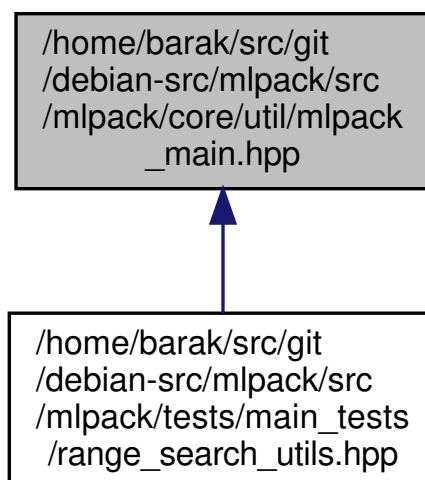
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.352 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/mlpack_main.hpp File Reference

Include dependency graph for mlpack_main.hpp:



This graph shows which files directly or indirectly include this file:



Macros

- `#define BINDING_TYPE BINDING_TYPE_UNKNOWN`
- `#define BINDING_TYPE_CLI 0`
- `#define BINDING_TYPE_MARKDOWN 128`
- `#define BINDING_TYPE_PYX 2`
- `#define BINDING_TYPE_TEST 1`
- `#define BINDING_TYPE_UNKNOWN -1`

40.352.1 Macro Definition Documentation

40.352.1.1 BINDING_TYPE

```
#define BINDING_TYPE BINDING_TYPE_UNKNOWN
```

Definition at line 28 of file `mlpack_main.hpp`.

Referenced by `set()`.

40.352.1.2 BINDING_TYPE_CLI

```
#define BINDING_TYPE_CLI 0
```

Parameters

<code>mlpack_cli_main.hpp</code>	
----------------------------------	--

Author

Ryan Curtin

This file, based on the value of the macro `BINDING_TYPE`, will define the macros necessary to compile an `mlpack` binding for the target language.

This file should *only* be included by a program that is meant to be a command-line program or a binding to another language. This file also includes **`param_checks.hpp`** (p. 2736), which contains functions that are used to check parameter values at runtime.

`mlpack` is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with `mlpack`. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Definition at line 21 of file `mlpack_main.hpp`.

Referenced by `set()`.

40.352.1.3 BINDING_TYPE_MARKDOWN

```
#define BINDING_TYPE_MARKDOWN 128
```

Definition at line 24 of file mlpack_main.hpp.

Referenced by set().

40.352.1.4 BINDING_TYPE_PYX

```
#define BINDING_TYPE_PYX 2
```

Definition at line 23 of file mlpack_main.hpp.

Referenced by endif().

40.352.1.5 BINDING_TYPE_TEST

```
#define BINDING_TYPE_TEST 1
```

Definition at line 22 of file mlpack_main.hpp.

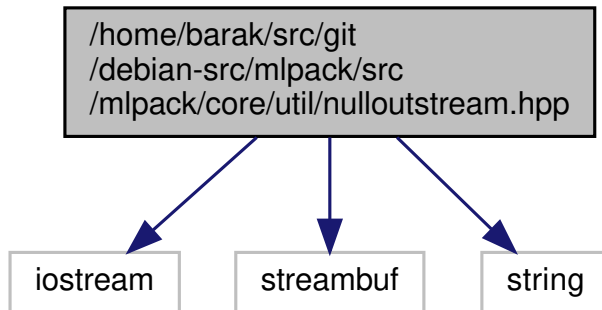
40.352.1.6 BINDING_TYPE_UNKNOWN

```
#define BINDING_TYPE_UNKNOWN -1
```

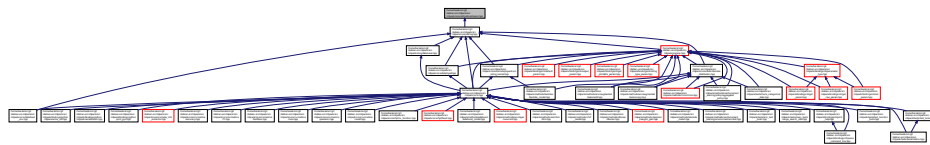
Definition at line 25 of file mlpack_main.hpp.

40.353 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/nullostream.hpp File Reference

Include dependency graph for nullostream.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **NullOutputStream**
Used for **Log::Debug** (p. 1540) when not compiled with debugging symbols.

Namespaces

- **mlpack**
.hpp
- **mlpack::util**

40.353.1 Detailed Description

Author

Ryan Curtin
Matthew Amidon

Definition of the `NullOutputStream` class.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.354 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/param.hpp File Reference

Classes

- class **DatasetMapper**< **PolicyType**, **InputType** >
Auxiliary information for a dataset, including mappings to/from strings (or other types) and the datatype of each dimension.

Namespaces

- **mlpack**
.hpp
- **mlpack::data**
Functions to load and save matrices and models.

Macros

- #define **PARAM_COL**(ID, DESC, ALIAS, REQ, TRANS, IN)
- #define **PARAM_COL_IN**(ID, DESC, ALIAS) **PARAM_COL**(ID, DESC, ALIAS, false, true, true)
Define a vector input parameter (type arma::vec).
- #define **PARAM_COL_IN_REQ**(ID, DESC, ALIAS) **PARAM_COL**(ID, DESC, ALIAS, true, true, true)
Define a required vector input parameter (type arma::vec).
- #define **PARAM_COL_OUT**(ID, DESC, ALIAS) **PARAM_COL**(ID, DESC, ALIAS, false, true, false)
Define a vector output parameter (type arma::vec).
- #define **PARAM_DOUBLE_IN**(ID, DESC, ALIAS, DEF) **PARAM_IN**(double, ID, DESC, ALIAS, DEF, false)
Define a double input parameter.
- #define **PARAM_DOUBLE_IN_REQ**(ID, DESC, ALIAS) **PARAM_IN**(double, ID, DESC, ALIAS, 0.0, true)
Define a required double parameter.
- #define **PARAM_DOUBLE_OUT**(ID, DESC) **PARAM_OUT**(double, ID, DESC, "", 0.0, false)
Define a double output parameter.
- #define **PARAM_FLAG**(ID, DESC, ALIAS) **PARAM_IN**(bool, ID, DESC, ALIAS, false, false);
Define a flag parameter.
- #define **PARAM_IN**(T, ID, DESC, ALIAS, DEF, REQ)
Define an input parameter.
- #define **PARAM_INT_IN**(ID, DESC, ALIAS, DEF) **PARAM_IN**(int, ID, DESC, ALIAS, DEF, false)
Define an integer input parameter.
- #define **PARAM_INT_IN_REQ**(ID, DESC, ALIAS) **PARAM_IN**(int, ID, DESC, ALIAS, 0, true)
Define a required integer input parameter.
- #define **PARAM_INT_OUT**(ID, DESC) **PARAM_OUT**(int, ID, DESC, "", 0, false)
Define an integer output parameter.
- #define **PARAM_MATRIX**(ID, DESC, ALIAS, REQ, TRANS, IN)
- #define **PARAM_MATRIX_AND_INFO_IN**(ID, DESC, ALIAS) **PARAM_IN**(**TUPLE_TYPE**, ID, DESC, ALIAS, **TUPLE_TYPE**(), false)
- #define **PARAM_MATRIX_IN**(ID, DESC, ALIAS) **PARAM_MATRIX**(ID, DESC, ALIAS, false, true, true)
Define a matrix input parameter.

- **#define PARAM_MATRIX_IN_REQ**(ID, DESC, ALIAS) **PARAM_MATRIX**(ID, DESC, ALIAS, true, true, true)
Define a required matrix input parameter.
- **#define PARAM_MATRIX_OUT**(ID, DESC, ALIAS) **PARAM_MATRIX**(ID, DESC, ALIAS, false, true, false)
Define a matrix output parameter.
- **#define PARAM_MODEL**(TYPE, ID, DESC, ALIAS, REQ, IN)
- **#define PARAM_MODEL_IN**(TYPE, ID, DESC, ALIAS) **PARAM_MODEL**(TYPE, ID, DESC, ALIAS, false, true)
Define an input model.
- **#define PARAM_MODEL_IN_REQ**(TYPE, ID, DESC, ALIAS) **PARAM_MODEL**(TYPE, ID, DESC, ALIAS, true, true)
Define a required input model.
- **#define PARAM_MODEL_OUT**(TYPE, ID, DESC, ALIAS) **PARAM_MODEL**(TYPE, ID, DESC, ALIAS, false, false)
Define an output model.
- **#define PARAM_OUT**(T, ID, DESC, ALIAS, DEF, REQ)
- **#define PARAM_ROW**(ID, DESC, ALIAS, REQ, TRANS, IN)
- **#define PARAM_ROW_IN**(ID, DESC, ALIAS) **PARAM_ROW**(ID, DESC, ALIAS, false, true, true)
Define a row vector input parameter (type arma::rowvec).
- **#define PARAM_ROW_OUT**(ID, DESC, ALIAS) **PARAM_ROW**(ID, DESC, ALIAS, false, true, false)
Define a row vector output parameter (type arma::rowvec).
- **#define PARAM_STRING_IN**(ID, DESC, ALIAS, DEF) **PARAM_IN**(std::string, ID, DESC, ALIAS, DEF, false)
Define a string input parameter.
- **#define PARAM_STRING_IN_REQ**(ID, DESC, ALIAS) **PARAM_IN**(std::string, ID, DESC, ALIAS, "", true)
Define a required string parameter.
- **#define PARAM_STRING_OUT**(ID, DESC, ALIAS) **PARAM_OUT**(std::string, ID, DESC, ALIAS, "", false)
Define a string output parameter.
- **#define PARAM_TMATRIX_IN**(ID, DESC, ALIAS) **PARAM_MATRIX**(ID, DESC, ALIAS, false, false, true)
Define a transposed matrix input parameter.
- **#define PARAM_TMATRIX_IN_REQ**(ID, DESC, ALIAS) **PARAM_MATRIX**(ID, DESC, ALIAS, true, false, true)
Define a required transposed matrix input parameter.
- **#define PARAM_TMATRIX_OUT**(ID, DESC, ALIAS) **PARAM_MATRIX**(ID, DESC, ALIAS, false, false, false)
Define a transposed matrix output parameter.
- **#define PARAM_UCOL**(ID, DESC, ALIAS, REQ, TRANS, IN)
- **#define PARAM_UCOL_IN**(ID, DESC, ALIAS) **PARAM_UCOL**(ID, DESC, ALIAS, false, true, true)
Define an unsigned vector input parameter (type arma::Col<size_t>).
- **#define PARAM_UCOL_OUT**(ID, DESC, ALIAS) **PARAM_UCOL**(ID, DESC, ALIAS, false, true, false)
Define an unsigned vector output parameter (type arma::Col<size_t>).
- **#define PARAM_UMATRIX**(ID, DESC, ALIAS, REQ, TRANS, IN)
- **#define PARAM_UMATRIX_IN**(ID, DESC, ALIAS) **PARAM_UMATRIX**(ID, DESC, ALIAS, false, true, true)
Define an unsigned matrix input parameter (arma::Mat<size_t>).
- **#define PARAM_UMATRIX_IN_REQ**(ID, DESC, ALIAS) **PARAM_UMATRIX**(ID, DESC, ALIAS, true, true, true)
Define a required unsigned matrix input parameter (arma::Mat<size_t>).
- **#define PARAM_UMATRIX_OUT**(ID, DESC, ALIAS) **PARAM_UMATRIX**(ID, DESC, ALIAS, false, true, false)
Define an unsigned matrix output parameter (arma::Mat<size_t>).
- **#define PARAM_UROW**(ID, DESC, ALIAS, REQ, TRANS, IN)
- **#define PARAM_UROW_IN**(ID, DESC, ALIAS) **PARAM_UROW**(ID, DESC, ALIAS, false, true, true)
Define an unsigned row vector input parameter (type arma::Row<size_t>).
- **#define PARAM_UROW_OUT**(ID, DESC, ALIAS) **PARAM_UROW**(ID, DESC, ALIAS, false, true, false)
Define an unsigned row vector output parameter (type arma::Row<size_t>).

- **#define PARAM_VECTOR_IN**(T, ID, DESC, ALIAS) **PARAM_IN**(std::vector<T>, ID, DESC, ALIAS, std::vector<T>(), false)
Define a std::vector input parameter.
- **#define PARAM_VECTOR_IN_REQ**(T, ID, DESC, ALIAS) **PARAM_IN**(std::vector<T>, ID, DESC, ALIAS, std::vector<T>(), true);
Define a required vector parameter.
- **#define PARAM_VECTOR_OUT**(T, ID, DESC, ALIAS) **PARAM_OUT**(std::vector<T>, ID, DESC, ALIAS, std::vector<T>(), false)
Define a vector output parameter.
- **#define PROGRAM_INFO**(NAME, SHORT_DESC, DESC, ...)
Document an executable.
- **#define SEE_ALSO**(DESCRIPTION, LINK) {DESCRIPTION, LINK}
Provide a link for a binding's "see also" documentation section, which is primarily (but not necessarily exclusively) used by the Markdown bindings. This link can be specified by calling SEE_ALSO("description", "link"), where "description" is the description of the link and "link" may be one of the following:
- **#define TUPLE_TYPE** std::tuple< **mlpack::data::DatasetInfo**, arma::mat>
Define an input DatasetInfo/matrix parameter.

40.354.1 Detailed Description

Author

Matthew Amidon
 Ryan Curtin

Definition of PARAM_*_IN() and PARAM_*_OUT() macros, as well as the **PROGRAM_INFO**() (p. 2733) macro, which are used to define input and output parameters of command-line programs and bindings to other languages.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.354.2 Macro Definition Documentation

40.354.2.1 PARAM_COL

```
#define PARAM_COL(  
    ID,  
    DESC,  
    ALIAS,  
    REQ,  
    TRANS,  
    IN )
```

Value:

```
static mlpack::util::Option<arma::vec> \  
  JOIN(cli_option_dummy_object_col_, __LINE_) \  
  (arma::vec(), ID, DESC, ALIAS, "arma::vec", REQ, IN, !TRANS, \  
  testName);
```

Definition at line 1140 of file param.hpp.

40.354.2.2 PARAM_COL_IN

```
#define PARAM_COL_IN(
    ID,
    DESC,
    ALIAS ) PARAM_COL(ID, DESC, ALIAS, false, true, true)
```

Define a vector input parameter (type arma::vec).

From the command line, the user can specify the file that holds the vector, using the name of the vector parameter with "_file" appended (and the same alias). So for instance, if the name of the vector parameter was "vec", the user could specify that the "vec" vector was held in vec.csv by giving the parameter:

```
--vec_file vector.csv
```

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 535 of file param.hpp.

40.354.2.3 PARAM_COL_IN_REQ

```
#define PARAM_COL_IN_REQ(
    ID,
    DESC,
    ALIAS ) PARAM_COL(ID, DESC, ALIAS, true, true, true)
```

Define a required vector input parameter (type arma::vec).

From the command line, the user can specify the file that holds the vector, using the name of the vector parameter with "_file" appended (and the same alias). So for instance, if the name of the vector parameter was "vec", the user could specify that the "vec" vector was held in vec.csv by giving the parameter:

```
--vec_file vector.csv
```

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 561 of file `param.hpp`.

40.354.2.4 PARAM_COL_OUT

```
#define PARAM_COL_OUT(
    ID,
    DESC,
    ALIAS )  PARAM_COL(ID, DESC, ALIAS, false, true, false)
```

Define a vector output parameter (type `arma::vec`).

When the program terminates, the vector will be saved to whatever it was set to during the program. From the command-line, the user may specify the file in which to save the output vector using a string option that is the name of the matrix parameter with `"_file"` appended. So, for instance, if the name of the output vector parameter was `"vec"`, the user could specify that the `"vec"` vector should be saved in `vector.csv` by giving the parameter:

```
--vec_file vector.csv
```

The output vector will not be printed on `stdout`, like the other output option types.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 671 of file param.hpp.

40.354.2.5 PARAM_DOUBLE_IN

```
#define PARAM_DOUBLE_IN(  
    ID,  
    DESC,  
    ALIAS,  
    DEF ) PARAM_IN(double, ID, DESC, ALIAS, DEF, false)
```

Define a double input parameter.

The parameter can then be specified on the command line with `-ID=value`.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).
<i>DEF</i>	Default value of the parameter.

See also

mlpack::CLI (p. 1117), **PROGRAM_INFO()** (p. 2733)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 170 of file param.hpp.

40.354.2.6 PARAM_DOUBLE_IN_REQ

```
#define PARAM_DOUBLE_IN_REQ(  
    ID,  
    DESC,  
    ALIAS ) PARAM_IN(double, ID, DESC, ALIAS, 0.0, true)
```

Define a required double parameter.

The parameter must then be specified on the command line with `-ID=value`.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

See also

mlpack::CLI (p. 1117), **PROGRAM_INFO()** (p. 2733)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 986 of file param.hpp.

40.354.2.7 PARAM_DOUBLE_OUT

```
#define PARAM_DOUBLE_OUT(  
    ID,  
    DESC )  PARAM_OUT(double, ID, DESC, "", 0.0, false)
```

Define a double output parameter.

This parameter will be printed on stdout at the end of the program; for instance, if the parameter name is "number" and the value is 5.012, the output on stdout would be of the following form:

```
number: 5.012
```

If the parameter is not set by the end of the program, a fatal runtime error will be issued.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).

See also

mlpack::CLI (p. 1117), **PROGRAM_INFO()** (p. 2733)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case,

the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 198 of file param.hpp.

40.354.2.8 PARAM_FLAG

```
#define PARAM_FLAG(  
    ID,  
    DESC,  
    ALIAS )  PARAM_IN(bool, ID, DESC, ALIAS, false, false);
```

Define a flag parameter.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

See also

mlpack::CLI (p. 1117), **PROGRAM_INFO()** (p. 2733)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 93 of file param.hpp.

40.354.2.9 PARAM_IN

```
#define PARAM_IN(  
    T,  
    ID,  
    DESC,  
    ALIAS,  
    DEF,  
    REQ )
```

Value:

```
static mlpack::util::Option<T> \
  JOIN(JOIN(cli_option_dummy_object_in_, __LINE__), opt) \
  (DEF, ID, DESC, ALIAS, #T, REQ, true, false, testName);
```

Define an input parameter.

Don't use this function; use the other ones above that call it. Note that we are using the **LINE** macro for naming these actual parameters when **COUNTER** does not exist, which is a bit of an ugly hack... but this is the preprocessor, after all. We don't have much choice other than ugliness.

Parameters

<i>T</i>	Type of the parameter.
<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	Alias for this parameter (one letter).
<i>DEF</i>	Default value of the parameter.
<i>REQ</i>	Whether or not parameter is required (boolean value).

Definition at line 1118 of file param.hpp.

40.354.2.10 PARAM_INT_IN

```
#define PARAM_INT_IN(
    ID,
    DESC,
    ALIAS,
    DEF ) PARAM_IN(int, ID, DESC, ALIAS, DEF, false)
```

Define an integer input parameter.

The parameter can then be specified on the command line with `-ID=value`.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).
<i>DEF</i>	Default value of the parameter.

See also

mlpack::CLI (p. 1117), **PROGRAM_INFO()** (p. 2733)

Bug Use a forward declaration of the class. The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—

most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 118 of file param.hpp.

40.354.2.11 PARAM_INT_IN_REQ

```
#define PARAM_INT_IN_REQ(  
    ID,  
    DESC,  
    ALIAS ) PARAM_IN(int, ID, DESC, ALIAS, 0, true)
```

Define a required integer input parameter.

The parameter must then be specified on the command line with `-ID=value`.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

See also

mlpack::CLI (p. 1117), **PROGRAM_INFO()** (p. 2733)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 964 of file param.hpp.

40.354.2.12 PARAM_INT_OUT

```
#define PARAM_INT_OUT(  
    ID,  
    DESC ) PARAM_OUT(int, ID, DESC, "", 0, false)
```

Define an integer output parameter.

This parameter will be printed on stdout at the end of the program; for instance, if the parameter name is "number" and the value is 5, the output on stdout would be of the following form:


```
number: 5
```

If the parameter is not set by the end of the program, a fatal runtime error will be issued.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).

See also

mlpack::CLI (p. 1117), **PROGRAM_INFO()** (p. 2733)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 146 of file param.hpp.

40.354.2.13 PARAM_MATRIX

```
#define PARAM_MATRIX(
    ID,
    DESC,
    ALIAS,
    REQ,
    TRANS,
    IN )
```

Value:

```
static mlpack::util::Option<arma::mat> \
  JOIN(JOIN(cli_option_dummy_object_matrix_, __LINE__), opt) \
  (arma::mat(), ID, DESC, ALIAS, "arma::mat", REQ, IN, !TRANS, \
  testName);
```

Definition at line 1128 of file param.hpp.

40.354.2.14 PARAM_MATRIX_AND_INFO_IN

```
#define PARAM_MATRIX_AND_INFO_IN(
    ID,
    DESC,
    ALIAS )  PARAM_IN( TUPLE_TYPE, ID, DESC, ALIAS,  TUPLE_TYPE(), false)
```

Definition at line 855 of file param.hpp.

40.354.2.15 PARAM_MATRIX_IN

```
#define PARAM_MATRIX_IN(  
    ID,  
    DESC,  
    ALIAS ) PARAM_MATRIX(ID, DESC, ALIAS, false, true, true)
```

Define a matrix input parameter.

From the command line, the user can specify the file that holds the matrix, using the name of the matrix parameter with "_file" appended (and the same alias). So for instance, if the name of the matrix parameter was "mat", the user could specify that the "mat" matrix was held in matrix.csv by giving the parameter

```
--mat_file matrix.csv
```

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 278 of file param.hpp.

40.354.2.16 PARAM_MATRIX_IN_REQ

```
#define PARAM_MATRIX_IN_REQ(  
    ID,  
    DESC,  
    ALIAS ) PARAM_MATRIX(ID, DESC, ALIAS, true, true, true)
```

Define a required matrix input parameter.

From the command line, the user can specify the file that holds the matrix, using the name of the matrix parameter with "_file" appended (and the same alias). So for instance, if the name of the matrix parameter was "mat", the user could specify that the "mat" matrix was held in matrix.csv by giving the parameter

```
--mat_file matrix.csv
```

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 304 of file `param.hpp`.

40.354.2.17 `PARAM_MATRIX_OUT`

```
#define PARAM_MATRIX_OUT(  
    ID,  
    DESC,  
    ALIAS ) PARAM_MATRIX(ID, DESC, ALIAS, false, true, false)
```

Define a matrix output parameter.

When the program terminates, the matrix will be saved to whatever it was set to by `CLI::GetParam<arma::mat>(ID)` during the program. From the command-line, the user may specify the file in which to save the output matrix using a string option that is the name of the matrix parameter with `"_file"` appended. So, for instance, if the name of the output matrix parameter was `"mat"`, the user could specify that the `"mat"` matrix should be saved in `matrix.csv` by giving the parameter

```
--mat_file matrix.csv
```

The output matrix will not be printed on `stdout`, like the other output option types.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they

produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 335 of file param.hpp.

40.354.2.18 PARAM_MODEL

```
#define PARAM_MODEL(  
    TYPE,  
    ID,  
    DESC,  
    ALIAS,  
    REQ,  
    IN )
```

Value:

```
static mlpack::util::Option<TYPE*> \  
  JOIN(JOIN(cli_option_dummy_object_model_, __LINE_), opt) \  
  (nullptr, ID, DESC, ALIAS, #TYPE, REQ, IN, false, \  
   testName);
```

Definition at line 1164 of file param.hpp.

40.354.2.19 PARAM_MODEL_IN

```
#define PARAM_MODEL_IN(  
    TYPE,  
    ID,  
    DESC,  
    ALIAS )  PARAM_MODEL(TYPE, ID, DESC, ALIAS, false, true)
```

Define an input model.

From the command line, the user can specify the file that holds the model, using the name of the model parameter with "_file" appended (and the same alias). So for instance, if the name of the model parameter was "model", the user could specify that the "model" model was held in model.bin by giving the parameter

```
--model_file model.bin
```

Note that the first parameter of this model is the type (the class name) of the model to be loaded. This model type must have a `Serialize()` function; a compilation error (a very long and complex one) will result if the model type does not have the following function:

```
template<typename Archive>  
void Serialize(Archive& ar, const unsigned int version);
```

This is the **boost::serialization** (p. 251) `serialize()` function, just with a capital s for `Serialize()` (see `src/mlpack/core/data/serialization↔_shim.hpp`).

Parameters

<i>TYPE</i>	Type of the model to be loaded.
<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter.
<i>ALIAS</i>	An alias for the parameter (one letter).

Definition at line 887 of file param.hpp.

40.354.2.20 PARAM_MODEL_IN_REQ

```
#define PARAM_MODEL_IN_REQ(
    TYPE,
    ID,
    DESC,
    ALIAS )  PARAM_MODEL(TYPE, ID, DESC, ALIAS, true, true)
```

Define a required input model.

From the command line, the user can specify the file that holds the model, using the name of the model parameter with "_file" appended (and the same alias). So for instance, if the name of the model parameter was "model", the user could specify that the "model" model was held in model.bin by giving the parameter

```
--model_file model.bin
```

Note that the first parameter of this model is the type (the class name) of the model to be loaded. This model type must have a `Serialize()` function; a compilation error (a very long and complex one) will result if the model type does not have the following function:

```
template<typename Archive>
void Serialize(Archive& ar, const unsigned int version);
```

This is the **boost::serialization** (p. 251) `serialize()` function, just with a capital s for `Serialize()` (see `src/mlpack/core/data/serialization/_shim.hpp`).

Parameters

<i>TYPE</i>	Type of the model to be loaded.
<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter.
<i>ALIAS</i>	An alias for the parameter (one letter).

Definition at line 919 of file param.hpp.

40.354.2.21 PARAM_MODEL_OUT

```
#define PARAM_MODEL_OUT(
    TYPE,
    ID,
    DESC,
    ALIAS )  PARAM_MODEL(TYPE, ID, DESC, ALIAS, false, false)
```

Define an output model.

From the command line, the user can specify the file that should hold the model, using the name of the model parameter with "_file" appended (and the same alias). So for instance, if the user desires to save the model to model.bin and the parameter name is "model", they could specify

```
--model_file model.bin
```

The model will be saved at the termination of the program. If you use a parameter of this type, you must call CLI::Destroy() at the end of your program.

Parameters

<i>TYPE</i>	Type of the model to be saved.
<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter.
<i>ALIAS</i>	An alias for the parameter (one letter).

Definition at line 942 of file param.hpp.

40.354.2.22 PARAM_OUT

```
#define PARAM_OUT(
    T,
    ID,
    DESC,
    ALIAS,
    DEF,
    REQ )
```

Value:

```
static mlpack::util::Option<T> \
    JOIN(JOIN(cli_option_dummy_object_out_, __LINE__), opt) \
    (DEF, ID, DESC, ALIAS, #T, REQ, false, false, testName);
```

Definition at line 1123 of file param.hpp.

40.354.2.23 PARAM_ROW

```
#define PARAM_ROW(
    ID,
    DESC,
    ALIAS,
    REQ,
    TRANS,
    IN )
```

Value:

```
static mlpack::util::Option<arma::rowvec> \
    JOIN(cli_option_dummy_object_row_, __LINE__) \
    (arma::rowvec(), ID, DESC, ALIAS, "arma::rowvec", REQ, IN, !TRANS, \
    testName);
```

Definition at line 1152 of file param.hpp.

40.354.2.24 PARAM_ROW_IN

```
#define PARAM_ROW_IN(
    ID,
    DESC,
    ALIAS ) PARAM_ROW(ID, DESC, ALIAS, false, true, true)
```

Define a row vector input parameter (type arma::rowvec).

From the command line, the user can specify the file that holds the vector, using the name of the vector parameter with "_file" appended (and the same alias). So for instance, if the name of the vector parameter was "vec", the user could specify that the "vec" vector was held in vec.csv by giving the parameter:

```
--vec_file vector.csv
```

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 587 of file param.hpp.

40.354.2.25 PARAM_ROW_OUT

```
#define PARAM_ROW_OUT(  
    ID,  
    DESC,  
    ALIAS )  PARAM_ROW(ID, DESC, ALIAS, false, true, false)
```

Define a row vector output parameter (type arma::rowvec).

When the program terminates, the vector will be saved to whatever it was set to during the program. From the command-line, the user may specify the file in which to save the output vector using a string option that is the name of the matrix parameter with "_file" appended. So, for instance, if the name of the output vector parameter was "vec", the user could specify that the "vec" vector should be saved in vector.csv by giving the parameter:

```
--vec_file vector.csv
```

The output vector will not be printed on stdout, like the other output option types.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 702 of file param.hpp.

40.354.2.26 PARAM_STRING_IN

```
#define PARAM_STRING_IN(  
    ID,  
    DESC,  
    ALIAS,  
    DEF )  PARAM_IN(std::string, ID, DESC, ALIAS, DEF, false)
```

Define a string input parameter.

The parameter can then be specified on the command line with **-ID=value**. If **ALIAS** is equal to **DEF_MOD** (which is set using the **PROGRAM_INFO**() (p. 2733) macro), the parameter can be specified with just **-ID=value**.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).
<i>DEF</i>	Default value of the parameter.

See also

mlpack::CLI (p. 1117), **PROGRAM_INFO()** (p. 2733)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 223 of file param.hpp.

40.354.2.27 PARAM_STRING_IN_REQ

```
#define PARAM_STRING_IN_REQ(
    ID,
    DESC,
    ALIAS )  PARAM_IN(std::string, ID, DESC, ALIAS, "", true)
```

Define a required string parameter.

The parameter must then be specified on the command line with **-ID=value**.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

See also

mlpack::CLI (p. 1117), **PROGRAM_INFO()** (p. 2733)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 1008 of file param.hpp.

40.354.2.28 PARAM_STRING_OUT

```
#define PARAM_STRING_OUT(  
    ID,  
    DESC,  
    ALIAS )  PARAM_OUT(std::string, ID, DESC, ALIAS, "", false)
```

Define a string output parameter.

The string will be printed to stdout at the end of the program. For instance, if there was a string output parameter called "something" with value "hello", at the end of the program the output would be of the following form:

```
something: "hello"
```

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

See also

mlpack::CLI (p. 1117), **PROGRAM_INFO()** (p. 2733)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_*()** macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 252 of file param.hpp.

40.354.2.29 PARAM_TMATRIX_IN

```
#define PARAM_TMATRIX_IN(  
    ID,  
    DESC,  
    ALIAS )  PARAM_MATRIX(ID, DESC, ALIAS, false, false, true)
```

Define a transposed matrix input parameter.

This is useful when data is desired in row-major form instead of the usual column-major form. From the command line, the user can specify the file that holds the matrix, using the name of the matrix parameter with "_file" appended (and the same alias). So for instance, if the name of the matrix parameter was "mat", the user could specify that the "mat" matrix was held in matrix.csv by giving the parameter

```
--mat_file matrix.csv
```

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 362 of file `param.hpp`.

40.354.2.30 PARAM_TMATRIX_IN_REQ

```
#define PARAM_TMATRIX_IN_REQ(  
    ID,  
    DESC,  
    ALIAS )  PARAM_MATRIX(ID, DESC, ALIAS, true, false, true)
```

Define a required transposed matrix input parameter.

This is useful when data is desired in row-major form instead of the usual column-major form. From the command line, the user can specify the file that holds the matrix, using the name of the matrix parameter with `"_file"` appended (and the same alias). So for instance, if the name of the matrix parameter was `"mat"`, the user could specify that the `"mat"` matrix was held in `matrix.csv` by giving the parameter

```
--mat_file matrix.csv
```

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 390 of file param.hpp.

40.354.2.31 PARAM_TMATRIX_OUT

```
#define PARAM_TMATRIX_OUT(  
    ID,  
    DESC,  
    ALIAS )  PARAM_MATRIX(ID, DESC, ALIAS, false, false, false)
```

Define a transposed matrix output parameter.

This is useful when data is stored in a row-major form instead of the usual column-major form. When the program terminates, the matrix will be saved to whatever it was set to by `CLI::GetParam<arma::mat>(ID)` during the program. From the command-line, the user may specify the file in which to save the output matrix using a string option that is the name of the matrix parameter with `"_file"` appended. So, for instance, if the name of the output matrix parameter was `"mat"`, the user could specify that the `"mat"` matrix should be saved in `matrix.csv` by giving the parameter

```
--mat_file matrix.csv
```

The output matrix will not be printed on stdout, like the other output option types.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 423 of file param.hpp.

40.354.2.32 PARAM_UCOL

```
#define PARAM_UCOL(  
    ID,  
    DESC,  
    ALIAS,
```

```

    REQ,
    TRANS,
    IN )

```

Value:

```

static mlpack::util::Option<arma::Col<size_t>> \
  JOIN(cli_option_dummy_object_ucol_, __LINE__) \
  (arma::Col<size_t>(), ID, DESC, ALIAS, "arma::Col<size_t>", REQ, IN, \
  !TRANS, testName);

```

Definition at line 1146 of file param.hpp.

40.354.2.33 PARAM_UCOL_IN

```

#define PARAM_UCOL_IN(
    ID,
    DESC,
    ALIAS )  PARAM_UCOL(ID, DESC, ALIAS, false, true, true)

```

Define an unsigned vector input parameter (type `arma::Col<size_t>`).

From the command line, the user can specify the file that holds the vector, using the name of the vector parameter with `"_file"` appended (and the same alias). So for instance, if the name of the vector parameter was `"vec"`, the user could specify that the `"vec"` vector was held in `vec.csv` by giving the parameter:

```
--vec_file vector.csv
```

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 613 of file param.hpp.

40.354.2.34 PARAM_UCOL_OUT

```
#define PARAM_UCOL_OUT(
    ID,
    DESC,
    ALIAS )  PARAM_UCOL(ID, DESC, ALIAS, false, true, false)
```

Define an unsigned vector output parameter (type `arma::Col<size_t>`).

When the program terminates, the vector will be saved to whatever it was set to during the program. From the command-line, the user may specify the file in which to save the output vector using a string option that is the name of the matrix parameter with "_file" appended. So, for instance, if the name of the output vector parameter was "vec", the user could specify that the "vec" vector should be saved in vector.csv by giving the parameter:

```
--vec_file vector.csv
```

The output vector will not be printed on stdout, like the other output option types.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*`() macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 733 of file `param.hpp`.

40.354.2.35 PARAM_UMATRIX

```
#define PARAM_UMATRIX(
    ID,
    DESC,
    ALIAS,
    REQ,
    TRANS,
    IN )
```

Value:

```
static mlpack::util::Option<arma::Mat<size_t>> \
  JOIN(JOIN(cli_option_dummy_object_umatix_, __LINE__), opt) \
  (arma::Mat<size_t>(), ID, DESC, ALIAS, "arma::Mat<size_t>", REQ, IN, \
  !TRANS, testName);
```

Definition at line 1134 of file `param.hpp`.

40.354.2.36 PARAM_UMATRIX_IN

```
#define PARAM_UMATRIX_IN(
    ID,
    DESC,
    ALIAS ) PARAM_UMATRIX(ID, DESC, ALIAS, false, true, true)
```

Define an unsigned matrix input parameter (arma::Mat<size_t>).

From the command line, the user can specify the file that holds the matrix, using the name of the matrix parameter with "_file" appended (and the same alias). So for instance, if the name of the matrix parameter was "mat", the user could specify that the "mat" matrix was held in matrix.csv by giving the parameter

```
--mat_file matrix.csv
```

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 449 of file param.hpp.

40.354.2.37 PARAM_UMATRIX_IN_REQ

```
#define PARAM_UMATRIX_IN_REQ(
    ID,
    DESC,
    ALIAS ) PARAM_UMATRIX(ID, DESC, ALIAS, true, true, true)
```

Define a required unsigned matrix input parameter (arma::Mat<size_t>).

From the command line, the user can specify the file that holds the matrix, using the name of the matrix parameter with "_file" appended (and the same alias). So for instance, if the name of the matrix parameter was "mat", the user could specify that the "mat" matrix was held in matrix.csv by giving the parameter

```
--mat_file matrix.csv
```


Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 476 of file `param.hpp`.

40.354.2.38 PARAM_UMATRIX_OUT

```
#define PARAM_UMATRIX_OUT(  
    ID,  
    DESC,  
    ALIAS )  PARAM_UMATRIX(ID, DESC, ALIAS, false, true, false)
```

Define an unsigned matrix output parameter (`arma::Mat<size_t>`).

When the program terminates, the matrix will be saved to whatever it was set to by `CLI::GetParam<arma::Mat<size_t>>(ID)` during the program. From the command-line, the user may specify the file in which to save the output matrix using a string option that is the name of the matrix parameter with `"_file"` appended. So, for instance, if the name of the output matrix parameter was `"mat"`, the user could specify that the `"mat"` matrix should be saved in `matrix.csv` by giving the parameter

```
--mat_file matrix.csv
```

The output matrix will not be printed on `stdout`, like the other output option types.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they

produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 508 of file param.hpp.

40.354.2.39 PARAM_UROW

```
#define PARAM_UROW(
    ID,
    DESC,
    ALIAS,
    REQ,
    TRANS,
    IN )
```

Value:

```
static mlpack::util::Option<arma::Row<size_t>> \
  JOIN(cli_option_dummy_object_urow_, __LINE__) \
  (arma::Row<size_t>(), ID, DESC, ALIAS, "arma::Row<size_t>", REQ, IN, \
  !TRANS, testName);
```

Definition at line 1158 of file param.hpp.

40.354.2.40 PARAM_UROW_IN

```
#define PARAM_UROW_IN(
    ID,
    DESC,
    ALIAS ) PARAM_UROW(ID, DESC, ALIAS, false, true, true)
```

Define an unsigned row vector input parameter (type `arma::Row<size_t>`).

From the command line, the user can specify the file that holds the vector, using the name of the vector parameter with `"_file"` appended (and the same alias). So for instance, if the name of the vector parameter was `"vec"`, the user could specify that the `"vec"` vector was held in `vec.csv` by giving the parameter:

```
--vec_file vector.csv
```

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 640 of file `param.hpp`.

40.354.2.41 PARAM_UROW_OUT

```
#define PARAM_UROW_OUT(  
    ID,  
    DESC,  
    ALIAS )  PARAM_UROW(ID, DESC, ALIAS, false, true, false)
```

Define an unsigned row vector output parameter (type `arma::Row<size_t>`).

When the program terminates, the vector will be saved to whatever it was set to during the program. From the command-line, the user may specify the file in which to save the output vector using a string option that is the name of the matrix parameter with `"_file"` appended. So, for instance, if the name of the output vector parameter was `"vec"`, the user could specify that the `"vec"` vector should be saved in `vector.csv` by giving the parameter:

```
--vec_file vector.csv
```

The output vector will not be printed on `stdout`, like the other output option types.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 764 of file `param.hpp`.

40.354.2.42 PARAM_VECTOR_IN

```
#define PARAM_VECTOR_IN(
    T,
    ID,
    DESC,
    ALIAS ) PARAM_IN(std::vector<T>, ID, DESC, ALIAS, std::vector<T>(), false)
```

Define a `std::vector` input parameter.

The parameter can then be specified on the command line with `-ID=value1,value2,value3`.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).
<i>DEF</i>	Default value of the parameter.

See also

mlpack::CLI (p. 1117), **PROGRAM_INFO()** (p. 2733)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 788 of file `param.hpp`.

40.354.2.43 PARAM_VECTOR_IN_REQ

```
#define PARAM_VECTOR_IN_REQ(
    T,
    ID,
    DESC,
    ALIAS ) PARAM_IN(std::vector<T>, ID, DESC, ALIAS, std::vector<T>(), true);
```

Define a required vector parameter.

The parameter must then be specified on the command line with `-ID=value1,value2,value3`.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	An alias for the parameter (one letter).

See also

mlpack::CLI (p. 1117), **PROGRAM_INFO()** (p. 2733)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 1031 of file param.hpp.

40.354.2.44 PARAM_VECTOR_OUT

```
#define PARAM_VECTOR_OUT(  
    T,  
    ID,  
    DESC,  
    ALIAS )  PARAM_OUT(std::vector<T>, ID, DESC, ALIAS, std::vector<T>(), false)
```

Define a vector output parameter.

This vector will be printed on stdout at the end of the program; for instance, if the parameter name is "vector" and the vector holds the array { 1, 2, 3, 4 }, the output on stdout would be of the following form:

```
vector: 1, 2, 3, 4
```

If the parameter is not set by the end of the program, a fatal runtime error will be issued.

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).

See also

mlpack::CLI (p. 1117), **PROGRAM_INFO()** (p. 2733)

Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the **PARAM_***() macros. However, not all compilers have this support—most notably, gcc < 4.3. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

Definition at line 817 of file param.hpp.

40.354.2.45 PROGRAM_INFO

```
#define PROGRAM_INFO(
    NAME,
    SHORT_DESC,
    DESC,
    ... )
```

Value:

```
static mlpack::util::ProgramDoc \
cli_programdoc_dummy_object = mlpack::util::ProgramDoc(NAME, SHORT_DESC, \
[]() { return DESC; }, { __VA_ARGS__ } )
```

Document an executable.

Only one instance of this macro should be present in your program! Therefore, use it in the main.cpp (or corresponding executable) in your program.

See also

mlpack::CLI (p. 1117), **PARAM_FLAG()** (p. 2710), **PARAM_INT_IN()** (p. 2711), **PARAM_DOUBLE_IN()** (p. 2708), **PARAM_STRING_IN()** (p. 2721), **PARAM_VECTOR_IN()** (p. 2731), **PARAM_INT_OUT()** (p. 2712), **PARAM_DOUBLE_OUT()** (p. 2709), **PARAM_VECTOR_OUT()** (p. 2733), **PARAM_INT_IN_REQ()** (p. 2712), **PARAM_DOUBLE_IN_REQ()** (p. 2708), **PARAM_STRING_IN_REQ()** (p. 2722), **PARAM_VECTOR_IN_REQ()** (p. 2732), **PARAM_INT_OUT_REQ()**, **PARAM_DOUBLE_OUT_REQ()**, **PARAM_VECTOR_OUT_REQ()**, **PARAM_STRING_OUT_REQ()**.

Parameters

<i>NAME</i>	Short string representing the name of the program.
<i>SHORT_DESC</i>	Short two-sentence description of the program; it should describe what the program implements and does, and a quick overview of how it can be used and what it should be used for.
<i>DESC</i>	Long string describing what the program does and possibly a simple usage example. Newlines should not be used here; this is taken care of by CLI (however, you can explicitly specify newlines to denote new paragraphs). You can also use printing macros like <code>PRINT_PARAM_STRING()</code> , <code>PRINT_DATASET()</code> , and others.
<i>SEE_ALSOS</i>	A set of SEE_ALSO() (p. 2734) macros that are used for generating documentation. See the SEE_ALSO() (p. 2734) macro. This is a varargs argument, so you can add as many SEE_ALSO() (p. 2734)s as you like.

Definition at line 71 of file param.hpp.

40.354.2.46 SEE_ALSO

```
#define SEE_ALSO(
    DESCRIPTION,
    LINK ) {DESCRIPTION, LINK}
```

Provide a link for a binding's "see also" documentation section, which is primarily (but not necessarily exclusively) used by the Markdown bindings. This link can be specified by calling `SEE_ALSO("description", "link")`, where "description" is the description of the link and "link" may be one of the following:

- A direct URL, starting with `http://` or `https://`.
- A page anchor for documentation, referencing another binding by its CMake binding name, i.e. `"#knn"`.
- A link to a Doxygen page, using the mangled Doxygen name after a `'/`, i.e., `"@doxygen/mlpack1_1_adaboost1↔_1_AdaBoost"`.

Definition at line 45 of file `param.hpp`.

40.354.2.47 TUPLE_TYPE

```
#define TUPLE_TYPE std::tuple< mlpack::data::DatasetInfo, arma::mat>
```

Define an input `DatasetInfo`/matrix parameter.

From the command line, the user can specify the file that holds the matrix, using the name of the matrix parameter with `"_file"` appended (and the same alias). So for instance, if the name of the matrix parameter was `"matrix"`, the user could specify that the `"matrix"` matrix was held in `file.csv` by giving the parameter

```
--matrix_file file.csv
```

Then the `DatasetInfo` and matrix type could be accessed with

```
DatasetInfo d = std::move(
    CLI::GetParam<std::tuple<arma::mat, DatasetInfo>>("matrix").get<0>());
arma::mat m = std::move(
    CLI::GetParam<std::tuple<arma::mat, DatasetInfo>>("matrix").get<1>());
```

Parameters

<i>ID</i>	Name of the parameter.
<i>DESC</i>	Quick description of the parameter (1-2 sentences).
<i>ALIAS</i>	One-character string representing the alias of the parameter.

See also

`mlpack::CLI` (p. 1117), `PROGRAM_INFO()` (p. 2733)

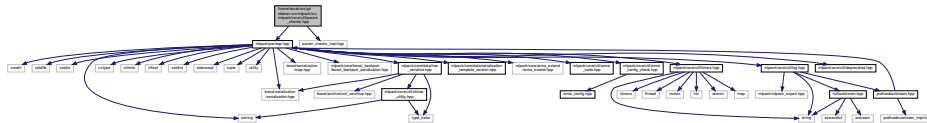
Bug The **COUNTER** variable is used in most cases to guarantee a unique global identifier for options declared using the `PARAM_*` macros. However, not all compilers have this support—most notably, `gcc < 4.3`. In that case, the **LINE** macro is used as an attempt to get a unique global identifier, but collisions are still possible, and they

produce bizarre error messages. See <https://github.com/mlpack/mlpack/issues/100> for more information.

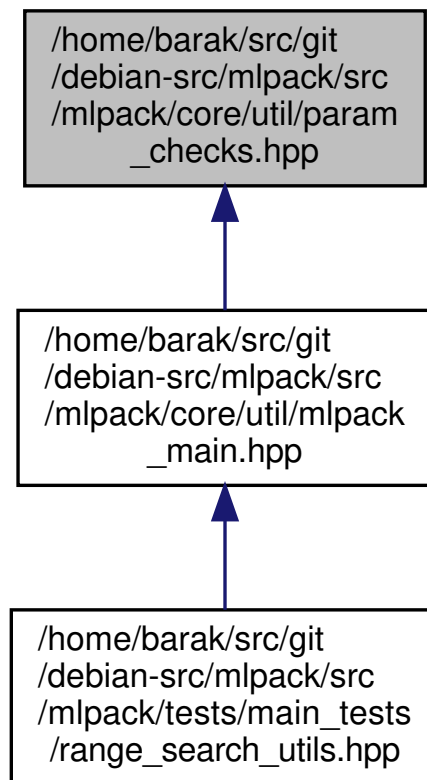
Definition at line 854 of file param.hpp.

40.355 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/param_checks.hpp File Reference

Include dependency graph for param_checks.hpp:



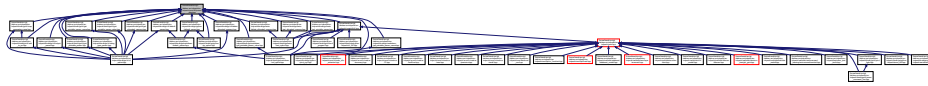
This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::util**

This graph shows which files directly or indirectly include this file:



Classes

- class **DatasetMapper**< **PolicyType**, **InputType** >
Auxiliary information for a dataset, including mappings to/from strings (or other types) and the datatype of each dimension.
- struct **ParamData**
*This structure holds all of the information about a single parameter, including its value (which is set when **ParseCommandLine()** (p. 302) is called).*

Namespaces

- **mlpack**
.hpp
- **mlpack::data**
Functions to load and save matrices and models.
- **mlpack::util**

Macros

- **#define TYPENAME(x)** (std::string(typeid(x).name()))
The TYPENAME macro is used internally to convert a type into a string.

40.356.1 Detailed Description

Author

Ryan Curtin

This defines the structure that holds information for each command-line parameter, as well as utility functions it is used with.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.356.2 Macro Definition Documentation

40.356.2.1 TYPENAME

```
#define TYPENAME(  
    x ) (std::string(typeid(x).name()))
```

The `TYPENAME` macro is used internally to convert a type into a string.

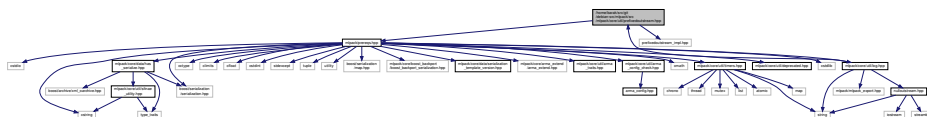
Definition at line 22 of file param_data.hpp.

Referenced by `CLIOption< N >::CLIOption()`, `MDOption< T >::MDOption()`, `PyOption< T >::PyOption()`, and `TestOption< N >::TestOption()`.

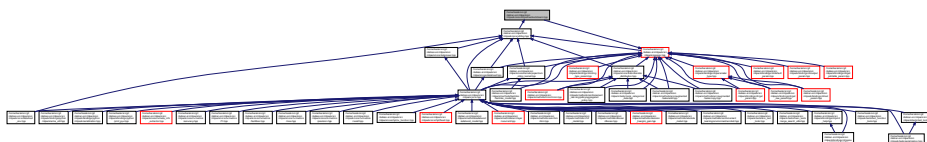
40.357 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/prefixedostream.hpp

File Reference

Include dependency graph for prefixedostream.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **PrefixedOutputStream**

Allows us to output to an ostream with a prefix at the beginning of each line, in the same way we would output to cout or cerr.

Namespaces

- **mlpack**
 .hpp
- **mlpack::util**

40.357.1 Detailed Description

Author

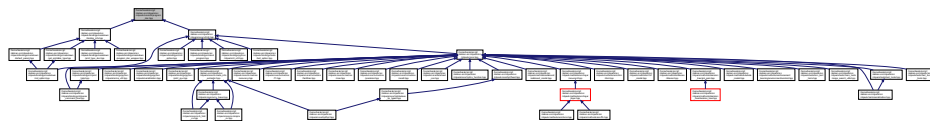
Ryan Curtin
Matthew Amidon

Declaration of the `PrefixedOutputStream` class.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.358 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/program_doc.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **ProgramDoc**
*A static object whose constructor registers program documentation with the **CLI** (p. 1117) class.*

Namespaces

- **mlpack**
.hpp
- **mlpack::util**

40.358.1 Detailed Description

Author

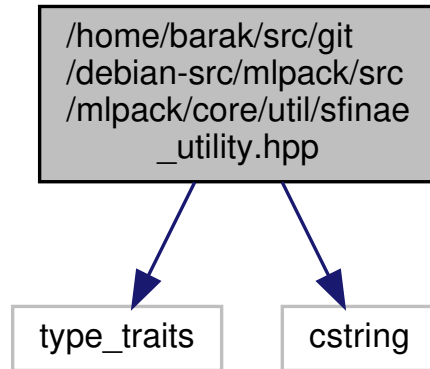
Matthew Amidon

The structure used to store a program's name and documentation.

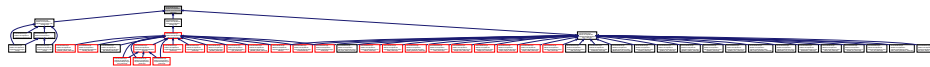
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.359 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/sfinae_utility.hpp File Reference

Include dependency graph for sfinae_utility.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct **MethodFormDetector**< **Class**, **MethodForm**, **AdditionalArgsCount** >
- struct **MethodFormDetector**< **Class**, **MethodForm**, **0** >
- struct **MethodFormDetector**< **Class**, **MethodForm**, **1** >
- struct **MethodFormDetector**< **Class**, **MethodForm**, **2** >
- struct **MethodFormDetector**< **Class**, **MethodForm**, **3** >
- struct **MethodFormDetector**< **Class**, **MethodForm**, **4** >
- struct **MethodFormDetector**< **Class**, **MethodForm**, **5** >
- struct **MethodFormDetector**< **Class**, **MethodForm**, **6** >
- struct **MethodFormDetector**< **Class**, **MethodForm**, **7** >
- struct **SigCheck**< **U**, **U** >

Utility struct for checking signatures.

Namespaces

- **mlpack**
 .hpp
- **mlpack::sfinae**

Macros

- **#define HAS_ANY_METHOD_FORM(FUNC, NAME)**
Constructs a template structure, which will define a boolean static variable, to true, if the passed template parameter, has a member function with the specified name.
- **#define HAS_EXACT_METHOD_FORM(METHOD, NAME) HAS_METHOD_FORM_BASE(SINGLE_ARG(METHOD), SINGLE_ARG(NAME), 0)**
HAS_EXACT_METHOD_FORM generates a template that allows a compile time check whether a given class has a method of the requested form.
- **#define HAS_MEM_FUNC(FUNC, NAME)**
- **#define HAS_METHOD_FORM(METHOD, NAME) HAS_METHOD_FORM_BASE(SINGLE_ARG(METHOD), SINGLE_ARG(NAME), 7)**
HAS_METHOD_FORM generates a template that allows a compile time check for whether a given class has a method of the requested form.
- **#define HAS_METHOD_FORM_BASE(METHOD, NAME, MAXN)**
*Base macro for **HAS_METHOD_FORM()** (p. 2744) and **HAS_EXACT_METHOD_FORM()** (p. 2743) macros.*
- **#define SINGLE_ARG(...) __VA_ARGS__**

40.359.1 Detailed Description

Author

Trironk Kiatkungwanglai, Kirill Mishchenko

This file contains macro utilities for the SFINAE Paradigm. These utilities determine if classes passed in as template parameters contain members at compile time, which is useful for changing functionality depending on what operations an object is capable of performing.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.359.2 Macro Definition Documentation

40.359.2.1 HAS_ANY_METHOD_FORM

```
#define HAS_ANY_METHOD_FORM(  
    FUNC,  
    NAME )
```

Value:

```

template <typename T>
struct NAME
{
    template <typename Q = T>
    static typename
    std::enable_if<std::is_member_function_pointer<decltype(&Q::FUNC)>::value,
                  int>::type
    f(int) { return 1; }

    template <typename Q = T>
    static char f(char) { return 0; }

    static const bool value = sizeof(f<T>(0)) != sizeof(char);
};

```

Constructs a template structure, which will define a boolean static variable, to true, if the passed template parameter, has a member function with the specified name.

The check does not care about the signature or the function parameters.

Parameters

<i>FUNC</i>	the name of the function, whose existence is to be detected
<i>NAME</i>	the name of the structure that will be generated

Use this like: `NAME<ClassName>::value` to check for the existence of the function in the given class name. This can also be used in conjunction with `std::enable_if`.

Definition at line 201 of file `sfinae_utility.hpp`.

40.359.2.2 HAS_EXACT_METHOD_FORM

```

#define HAS_EXACT_METHOD_FORM(
    METHOD,
    NAME ) HAS_METHOD_FORM_BASE( SINGLE_ARG(METHOD), SINGLE_ARG(NAME), 0)

```

`HAS_EXACT_METHOD_FORM` generates a template that allows a compile time check whether a given class has a method of the requested form.

For example, for the following class

```
class A { public: ... Train(const arma::mat&, const arma::Row<size_t>&); ... };
```

and the following form of Train methods

```
template<typename Class> using TrainForm = void(Class::*)(const arma::mat&, const arma::Row<size_t>&);
```

we can check whether the class A has a Train method of the specified form:

```
HAS_METHOD_FORM(Train, HasTrain) (p. 2744); static_assert(HasTrain<A, TrainForm>::value, "value should be true");
```

The class generated by this will only return true values if the signature matches exactly.

Parameters

<i>METHOD</i>	The name of the method to check for.
<i>NAME</i>	The name of the struct to construct.
<i>MAXN</i>	The maximum number of additional arguments.

Definition at line 287 of file `sfinae_utility.hpp`.

40.359.2.3 HAS_MEM_FUNC

```
#define HAS_MEM_FUNC(
    FUNC,
    NAME )
```

Value:

```
template<typename T, typename sig, typename = std::true_type>
struct NAME : std::false_type {};

template<typename T, typename sig>
struct NAME
<
    T,
    sig,
    std::integral_constant<bool, mlpack::sfinae::SigCheck<sig, &T::FUNC>::value>
> : std::true_type {};
```

Definition at line 128 of file `sfinae_utility.hpp`.

40.359.2.4 HAS_METHOD_FORM

```
#define HAS_METHOD_FORM(
    METHOD,
    NAME ) HAS_METHOD_FORM_BASE( SINGLE_ARG(METHOD), SINGLE_ARG(NAME), 7)
```

`HAS_METHOD_FORM` generates a template that allows a compile time check for whether a given class has a method of the requested form.

For example, for the following class

```
class A { public: ... Train(const arma::mat&, const arma::Row<size_t>&, double); ... };
```

and the following form of `Train` methods

```
template<typename Class, typename...Ts> using TrainForm = void(Class::*)(const arma::mat&, const arma::Row<size_t>&, Ts...);
```

we can check whether the class `A` has a `Train` method of the specified form:

```
HAS_METHOD_FORM(Train, HasTrain) (p. 2744); static_assert(HasTrain<A, TrainForm>::value, "value should be true");
```

The class generated by this will also return true values if the given class has a method that also has extra parameters.

Parameters

<i>METHOD</i>	The name of the method to check for.
<i>NAME</i>	The name of the struct to construct.
<i>MAXN</i>	The maximum number of additional arguments.

Definition at line 253 of file sfinae_utility.hpp.

40.359.2.5 HAS_METHOD_FORM_BASE

```
#define HAS_METHOD_FORM_BASE(  
    METHOD,  
    NAME,  
    MAXN )
```

Base macro for **HAS_METHOD_FORM()** (p. 2744) and **HAS_EXACT_METHOD_FORM()** (p. 2743) macros.

Definition at line 143 of file sfinae_utility.hpp.

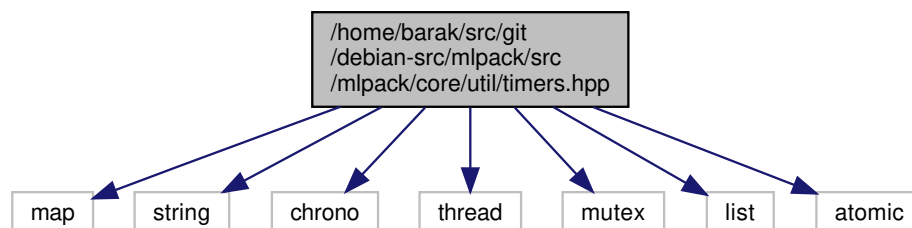
40.359.2.6 SINGLE_ARG

```
#define SINGLE_ARG(  
    ... ) __VA_ARGS__
```

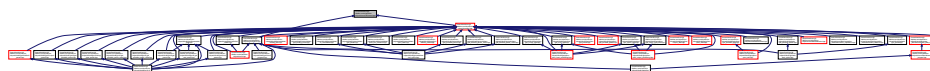
Definition at line 220 of file sfinae_utility.hpp.

40.360 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/util/timers.hpp File Reference

Include dependency graph for timers.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **Timer**
The timer class provides a way for mpack methods to be timed.
- class **Timers**

Namespaces

- **mpack**
.hpp

40.360.1 Detailed Description

Author

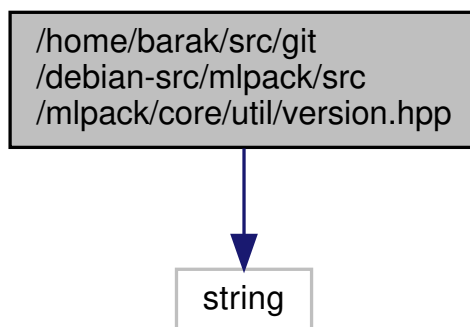
Matthew Amidon
 Marcus Edel
 Ryan Curtin

Timers for mpack.

mpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.361 /home/barak/src/git/debian-src/mpack/src/mpack/core/util/version.hpp File Reference

Include dependency graph for version.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
.hpp
- **mlpack::util**

Macros

- `#define MLPACK_VERSION_MAJOR 3`
- `#define MLPACK_VERSION_MINOR 1`
- `#define MLPACK_VERSION_PATCH 1`

Functions

- `std::string GetVersion ()`
This will return either "mlpack x.y.z" or "mlpack master-XXXXXXX" depending on whether or not this is a stable version of mlpack or a git repository.

40.361.1 Macro Definition Documentation

40.361.1.1 **MLPACK_VERSION_MAJOR**

```
#define MLPACK_VERSION_MAJOR 3
```

Definition at line 19 of file version.hpp.

Referenced by `include_directories()`.

40.361.1.2 **MLPACK_VERSION_MINOR**

```
#define MLPACK_VERSION_MINOR 1
```

Definition at line 20 of file version.hpp.

Referenced by `include_directories()`.

40.361.1.3 MLPACK_VERSION_PATCH

```
#define MLPACK_VERSION_PATCH 1
```

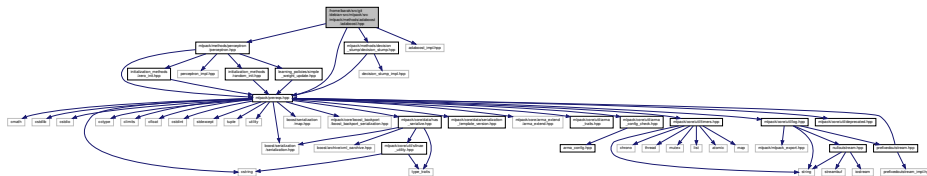
Definition at line 21 of file version.hpp.

Referenced by include_directories().

40.362 /home/barak/src/git/debian-src/mlpack/doc/guide/version.hpp File Reference

40.363 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/adaboost/adaboost.hpp File Reference

Include dependency graph for adaboost.hpp:



Classes

- struct **version**< **mlpack::adaboost::AdaBoost**< **WeakLearnerType**, **MatType** > >
- class **AdaBoost**< **WeakLearnerType**, **MatType** >

The *AdaBoost* (p. 488) class.

Namespaces

- **boost**
Set the serialization version of the adaboost class.
- **boost::serialization**
- **mlpack**
.hpp
- **mlpack::adaboost**

40.363.1 Detailed Description

Author

Udit Saxena

The AdaBoost class. AdaBoost is a boosting algorithm, meaning that it combines an ensemble of weak learners to produce a strong learner. For more information on AdaBoost, see the following paper:

```
@article{schapire1999improved,
  author = {Schapire, Robert E. and Singer, Yoram},
  title = {Improved Boosting Algorithms Using Confidence-rated Predictions},
  journal = {Machine Learning},
  volume = {37},
  number = {3},
  month = dec,
  year = {1999},
  issn = {0885-6125},
  pages = {297--336},
}
```

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.364 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/adaboost/adaboost_model.hpp File Reference

Include dependency graph for adaboost_model.hpp:



Classes

- class **AdaBoostModel**
The model to save to disk.

Namespaces

- **mlpack**
.hpp
- **mlpack::adaboost**

Namespaces

- **mlpack**
 .hpp
- **mlpack::amf**
 Alternating Matrix Factorization.

Typedefs

- `typedef amf::AMF< amf::SimpleResidueTermination, amf::RandomAcolInitialization<>, amf::NMFALSUpdate > NMFALSFactorizer`
- `template<typename MatType = arma::mat>
 using SVDBatchFactorizer = amf::AMF< amf::SimpleResidueTermination, amf::RandomAcolInitialization<>, amf::SVDBatchLearning >`
 Convenience typedefs.
- `template<class MatType = arma::mat>
 using SVDCompleteIncrementalFactorizer = amf::AMF< amf::SimpleResidueTermination, amf::RandomAcolInitialization<>, amf::SVDCompleteIncrementalLearning< MatType > >`
 SVDCompleteIncrementalFactorizer factorizes given matrix V into two matrices W and H by complete incremental gradient descent.
- `template<class MatType = arma::mat>
 using SVDIncompleteIncrementalFactorizer = amf::AMF< amf::SimpleResidueTermination, amf::RandomAcolInitialization<>, amf::SVDIncompleteIncrementalLearning >`
 SVDIncompleteIncrementalFactorizer factorizes given matrix V into two matrices W and H by incomplete incremental gradient descent.

40.365.1 Detailed Description

Author

Sumedh Ghaisas
 Mohan Rajendran
 Ryan Curtin

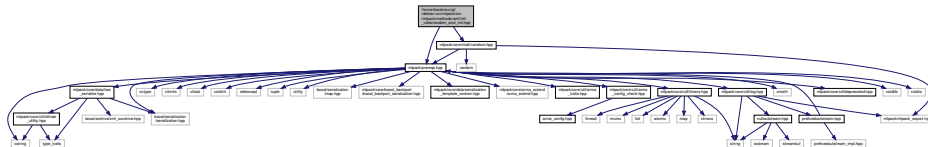
Alternating Matrix Factorization

The AMF (alternating matrix factorization) class, from which more commonly known techniques such as incremental SVD, NMF, and batch-learning SVD can be derived.

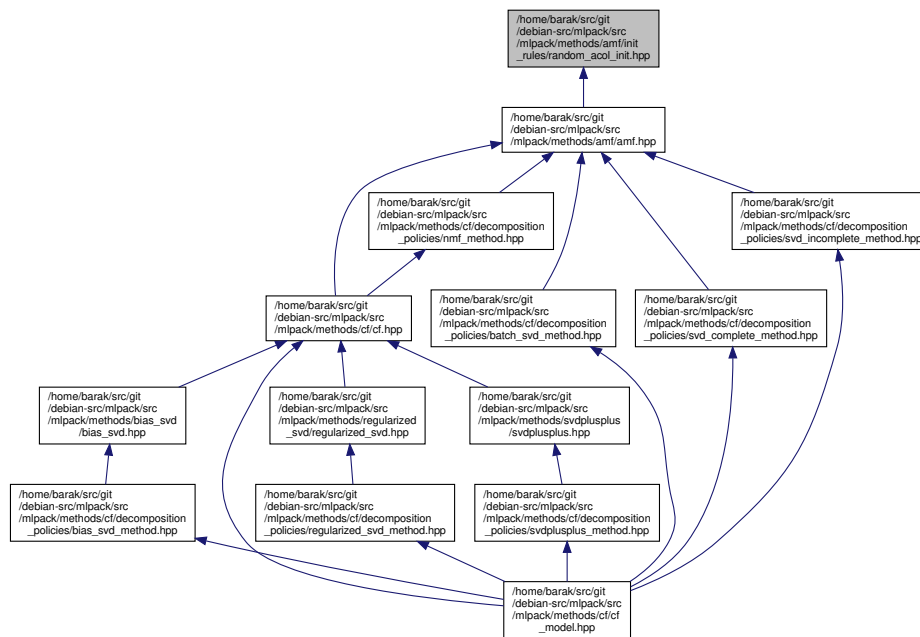
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.368 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/init_rules/random_acol_init.hpp File Reference

Include dependency graph for random_acol_init.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RandomAcolInitialization**< **columnsToAverage** >
*This class initializes the W matrix of the **AMF** (p. 499) algorithm by averaging p randomly chosen columns of V .*

Namespaces

- mlpack**
.hpp
- mlpack::amf**
Alternating Matrix Factorization.

40.368.1 Detailed Description

Author

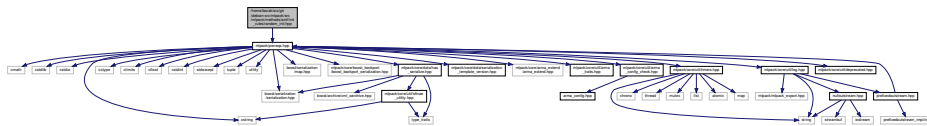
Mohan Rajendran

Initialization rule for Alternating Matrix Factorization.

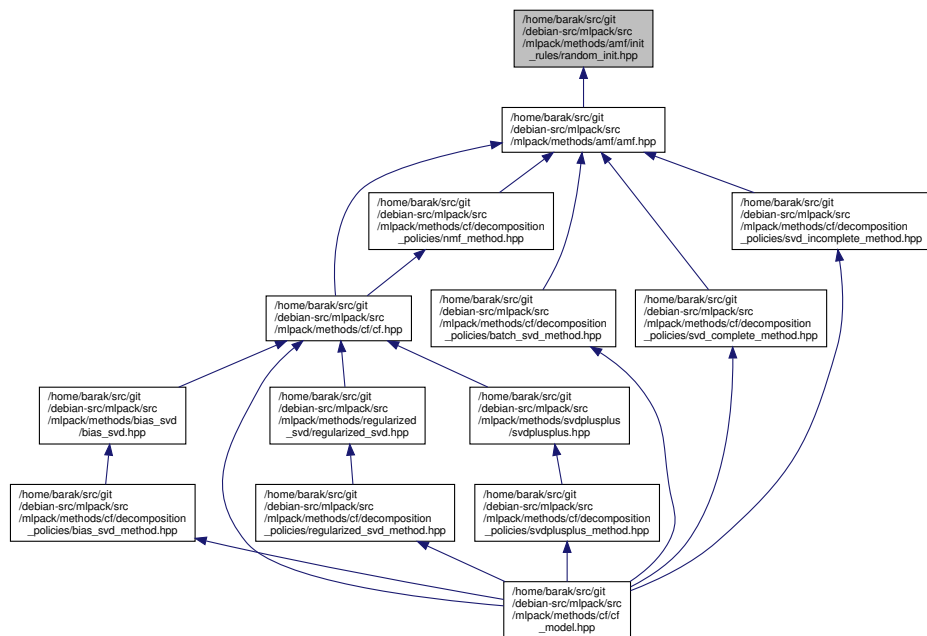
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.369 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/init_rules/random_init.hpp File Reference

Include dependency graph for random_init.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RandomInitialization**

*This initialization rule for **AMF** (p. 499) simply fills the W and H matrices with uniform random noise in $[0, 1]$.*

Classes

- This class acts as a wrapper for basic termination policies to be used by **SVDCompleteIncrementalLearning** (p. 542).*

Alternating Matrix Factorization.

Sumedh Ghaisas

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpak. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

[illegible]

*This class acts as a wrapper for basic termination policies to be used by **SVDIncompleteIncrementalLearning** (p. 547).*

Namespaces

- **mlpack**
 .hpp
- **mlpack::amf**
 Alternating Matrix Factorization.

40.373.1 Detailed Description

Author

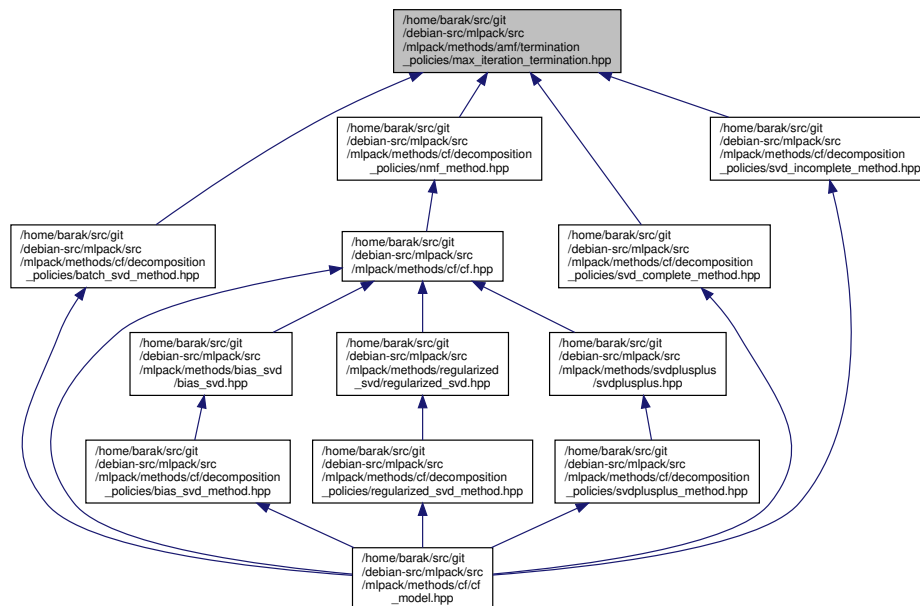
Sumedh Ghaisas

Termination policy used in AMF (Alternating Matrix Factorization).

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.374 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/max_iteration_termination.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **MaxIterationTermination**
 This termination policy only terminates when the maximum number of iterations has been reached.

Namespaces

- **mlpack**
 .hpp
- **mlpack::amf**
 Alternating Matrix Factorization.

40.374.1 Detailed Description

Author

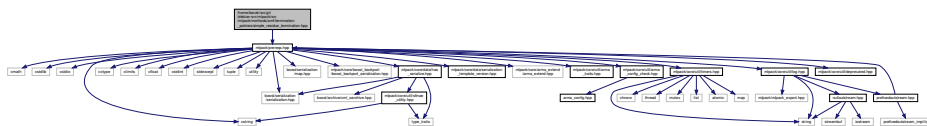
Ryan Curtin

A termination policy which only terminates when the maximum number of iterations is reached.

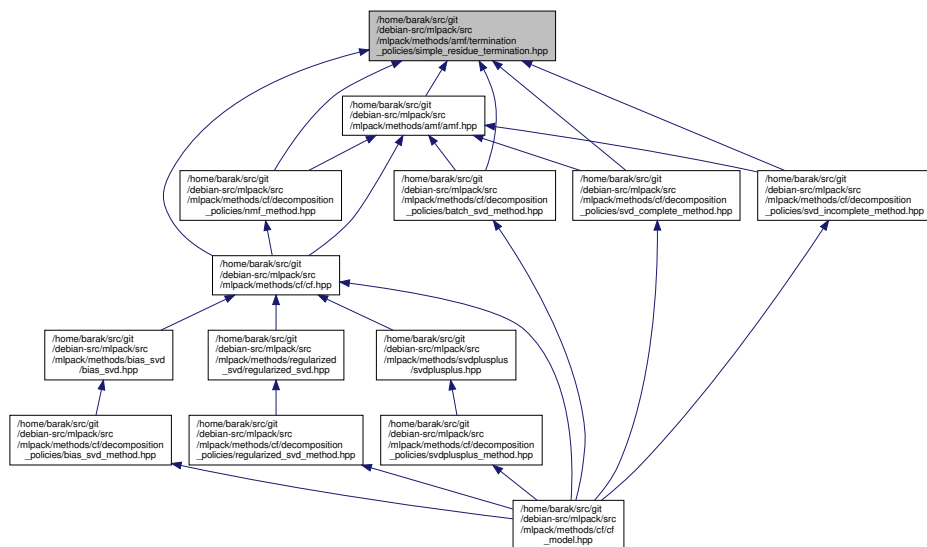
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.375 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/simple_residue_termination.hpp File Reference

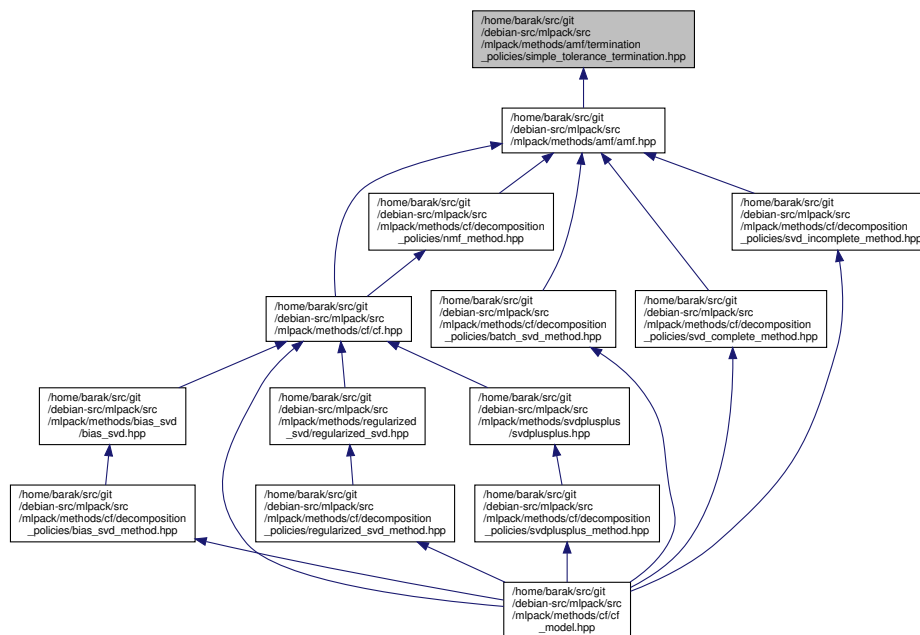
Include dependency graph for simple_residue_termination.hpp:



This graph shows which files directly or indirectly include this file:



This graph shows which files directly or indirectly include this file:



Classes

- class **SimpleToleranceTermination**< MatType >
This class implements residue tolerance termination policy.

Namespaces

- **mlpack**
.hpp
- **mlpack::amf**
Alternating Matrix Factorization.

40.376.1 Detailed Description

Author

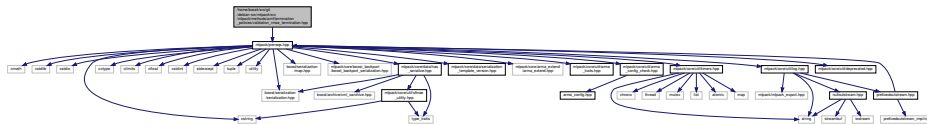
Sumedh Ghaisas

Termination policy used in AMF (Alternating Matrix Factorization).

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.377 [/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/termination_policies/validation_rmse_termination.hpp](#) File Reference

Include dependency graph for validation_rmse_termination.hpp:



Classes

- class **ValidationRMSETermination**< **MatType** >

This class implements validation termination policy based on RMSE index.

Namespaces

- **mlpack**

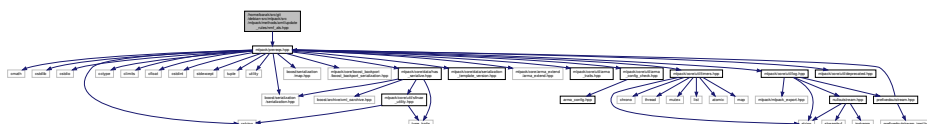
.hpp

- **mlpack::amf**

Alternating Matrix Factorization.

40.378 [/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/nmf_als.hpp](#) File Reference

Include dependency graph for nmf_als.hpp:

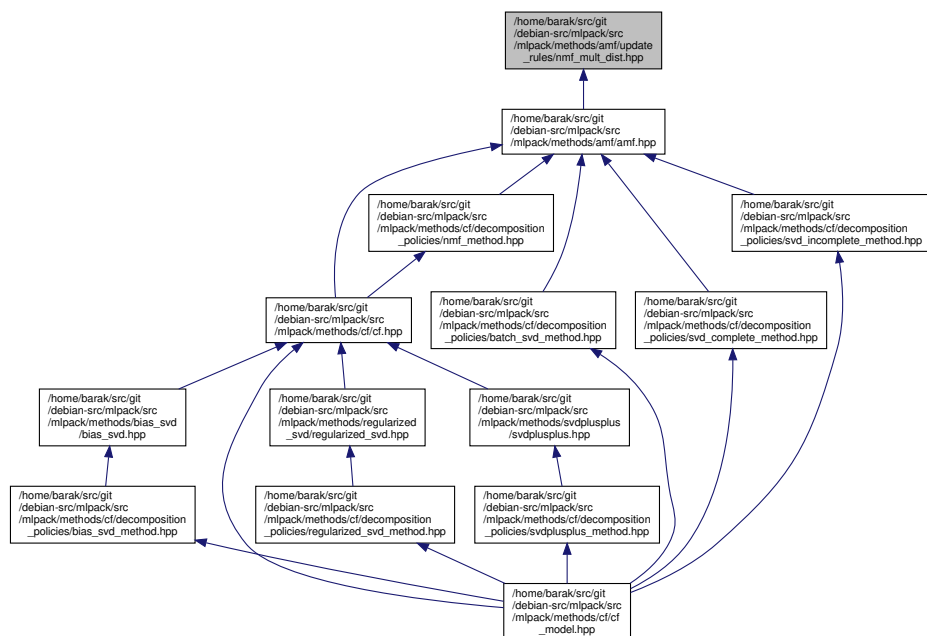


40.379 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/nmf_mult_dist.hpp File Reference

Include dependency graph for nmf_mult_dist.hpp:



This graph shows which files directly or indirectly include this file:



Classes

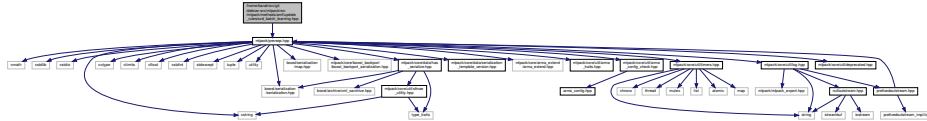
- class **NMFMultiplicativeDistanceUpdate**
The multiplicative distance update rules for matrices W and H .

Namespaces

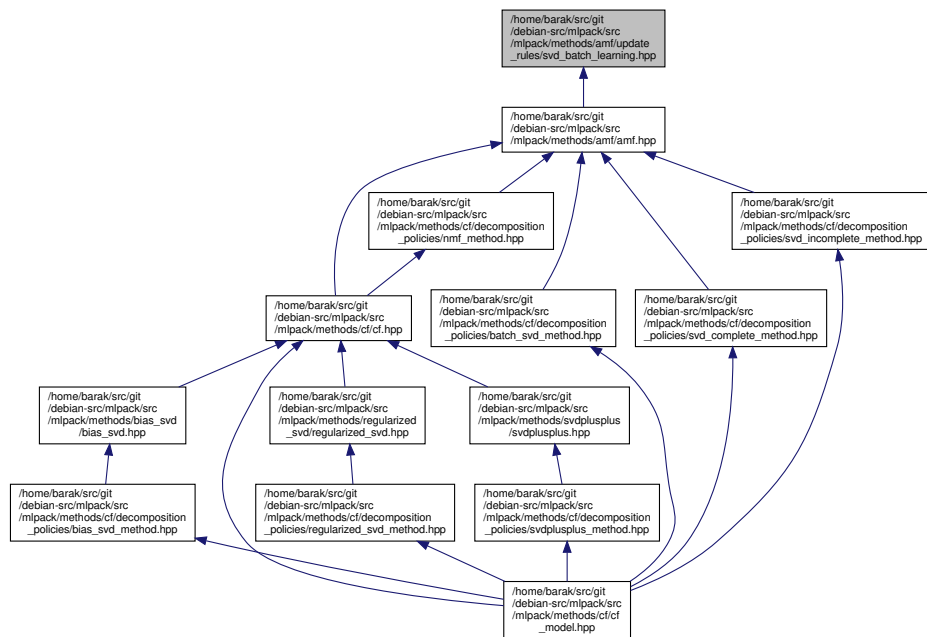
- mlpack**
.hpp
- mlpack::amf**
Alternating Matrix Factorization.

40.381 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/svd_↵
batch_learning.hpp File Reference

Include dependency graph for `svd_batch_learning.hpp`:



This graph shows which files directly or indirectly include this file:



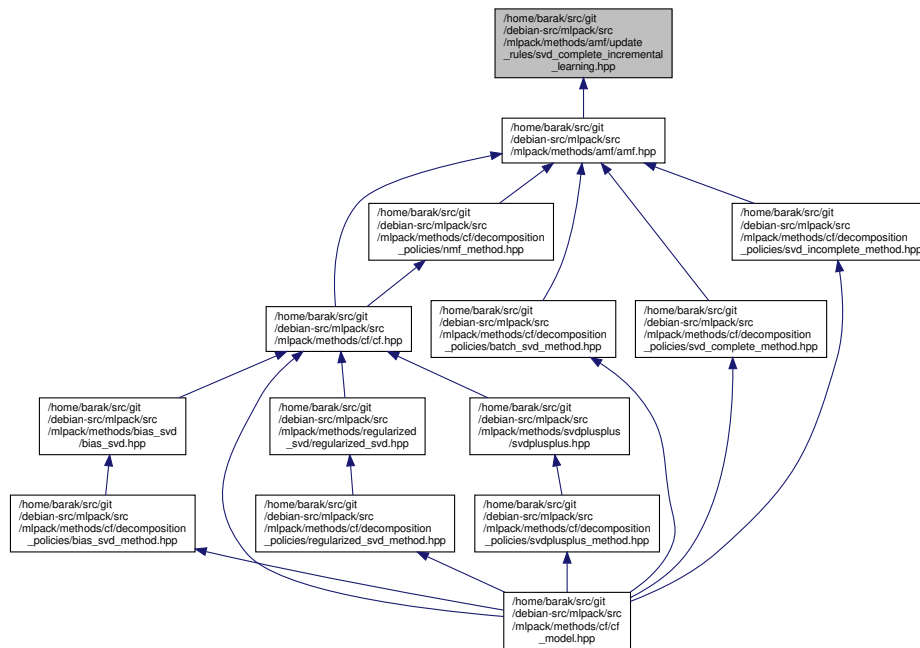
Classes

- class **SVDBatchLearning**
This class implements SVD batch learning with momentum.

Namespaces

- **mlpack**
.hpp
- **mlpack::amf**
Alternating Matrix Factorization.

This graph shows which files directly or indirectly include this file:



Classes

- class **SVDCompleteIncrementalLearning** < **MatType** >

This class computes SVD using complete incremental batch learning, as described in the following paper:

- class **SVDCompleteIncrementalLearning** < **arma::sp_mat** >

TODO : Merge this template specialized function for sparse matrix using common row_col_iterator.

Namespaces

- **mlpack**

.hpp

- **mlpack::amf**

Alternating Matrix Factorization.

40.382.1 Detailed Description

Author

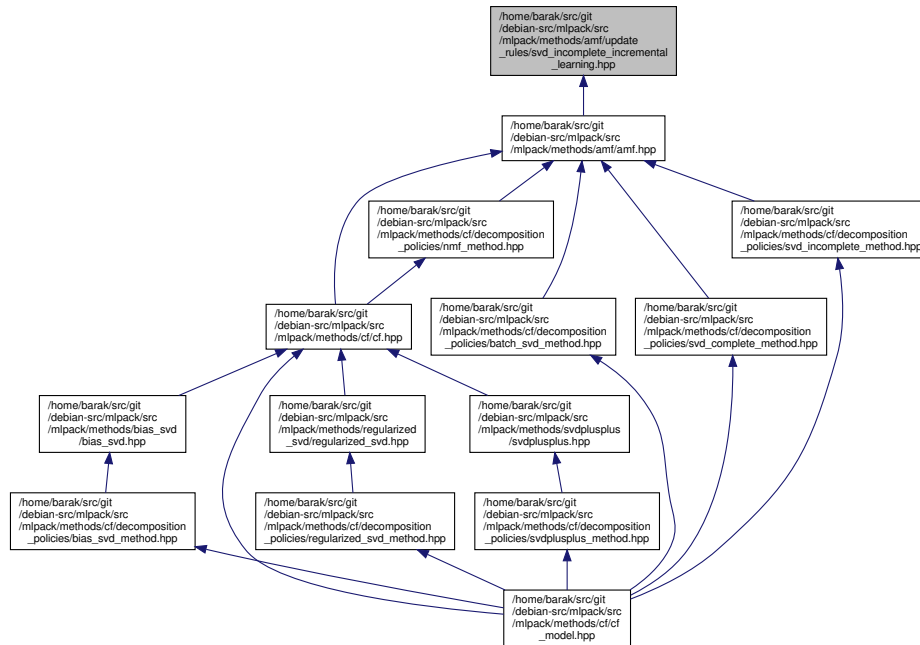
Sumedh Ghaisas

SVD factorizer used in AMF (Alternating Matrix Factorization).

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.383 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/amf/update_rules/svd_incomplete_incremental_learning.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **SVDIncompleteIncrementalLearning**

This class computes SVD using incomplete incremental batch learning, as described in the following paper:

Namespaces

- **mlpack**
 .hpp
- **mlpack::amf**
Alternating Matrix Factorization.

Functions

- template<>
 void **SVDIncompleteIncrementalLearning::HUpdate**< arma::sp_mat > (const arma::sp_mat &V, const arma::mat &W, arma::mat &H)
- template<>
 void **SVDIncompleteIncrementalLearning::WUpdate**< arma::sp_mat > (const arma::sp_mat &V, arma::mat &W, const arma::mat &H)

TODO : Merge this template specialized function for sparse matrix using common row_col_iterator.

40.383.1 Detailed Description

Author

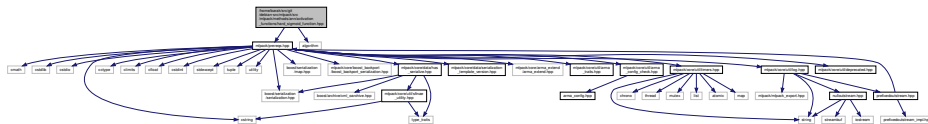
Sumedh Ghaisas

SVD factorizer used in AMF (Alternating Matrix Factorization).

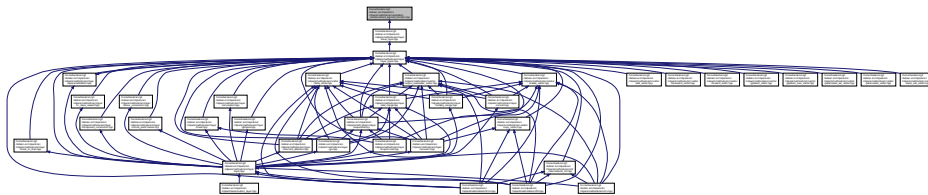
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.384 `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/hard_` `_sigmoid_function.hpp` File Reference

Include dependency graph for `hard_sigmoid_function.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **HardSigmoidFunction**
The hard sigmoid function, defined by.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.384.1 Detailed Description

Author

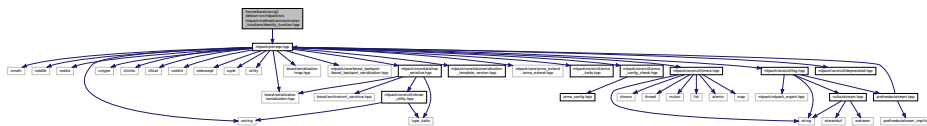
Bishwa Karki

Definition and implementation of the hard sigmoid function.

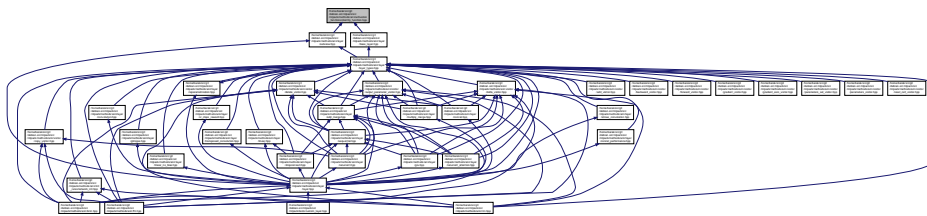
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.385 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/identity_function.hpp File Reference

Include dependency graph for identity_function.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **IdentityFunction**
The identity function, defined by.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.385.1 Detailed Description

Author

Marcus Edel

Definition and implementation of the identity function.

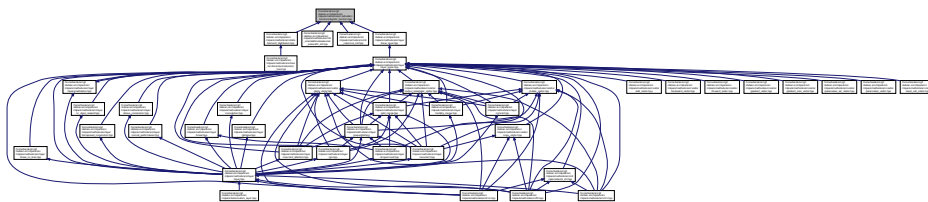
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.386 `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/logistic_function.hpp` File Reference

Include dependency graph for `logistic_function.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **LogisticFunction**
The logistic function, defined by.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.386.1 Detailed Description

Author

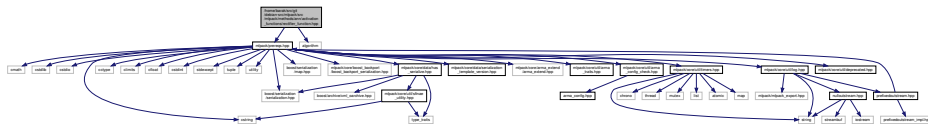
Marcus Edel

Definition and implementation of the logistic function.

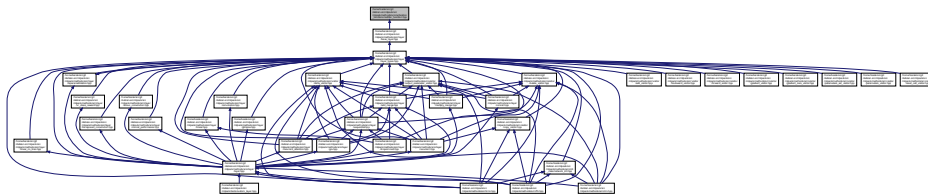
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.387 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/rectifier_↵ _function.hpp File Reference

Include dependency graph for rectifier_function.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RectifierFunction**
The rectifier function, defined by.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.387.1 Detailed Description

Author

Marcus Edel

Definition and implementation of the rectifier function as described by V. Nair and G. E. Hinton.

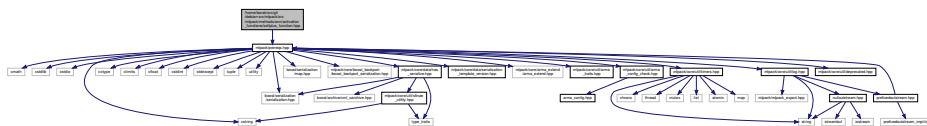
For more information, see the following paper.

```
@misc{NairHinton2010,
  author = {Vinod Nair, Geoffrey E. Hinton},
  title = {Rectified Linear Units Improve Restricted Boltzmann Machines},
  year = {2010}
}
```

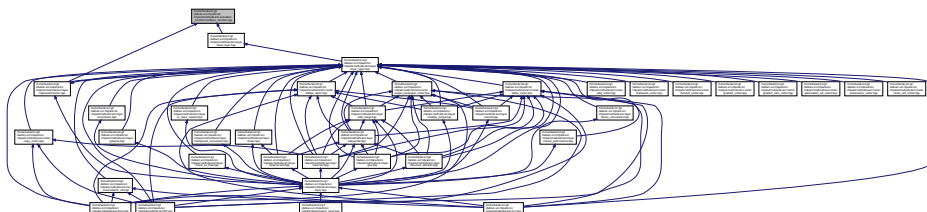
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.388 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/softplus_↵ _function.hpp File Reference

Include dependency graph for softplus_function.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **SoftplusFunction**
The softplus function, defined by.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.388.1 Detailed Description

Author

Vivek Pal

Definition and implementation of the softplus function as described by Charles Dugas, Yoshua Bengio, François Belisle, Claude Nadeau & René Garcia.

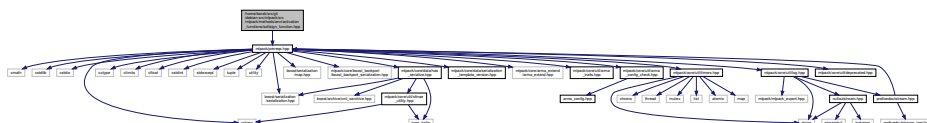
For more information, please see the following paper:

```
@inproceedings{Dugas2001,
  author    = {Dugas, Charles and Bengio, Yoshua and Belisle, Francois and
               Nadeau, Claude and Garcia, Rene},
  title     = {Incorporating Second-Order Functional Knowledge for Better
               Option Pricing},
  booktitle = {Advances in Neural Information Processing Systems},
  year      = {2001}
}
```

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpak. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.389 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/softsign.
_function.hpp File Reference

Include dependency graph for softsign function.hpp:



Classes

- class **SoftsignFunction**
The softsign function, defined by.

Namespaces

- **mlpack**
 .hpp
- **mlpack::ann**
 Artificial Neural Network.

40.389.1 Detailed Description

Author

Marcus Edel

Definition and implementation of the softsign function as described by X. Glorot and Y. Bengio.

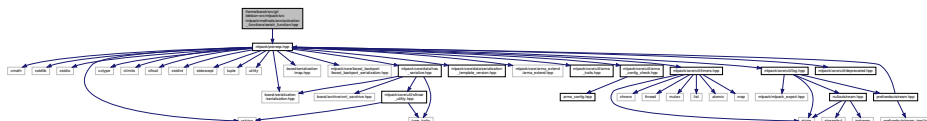
For more information, see the following paper.

```
@inproceedings{GlorotAISTATS2010,
  title={title={Understanding the difficulty of training deep feedforward
neural networks},
  author={Glorot, Xavier and Bengio, Yoshua},
  booktitle={Proceedings of AISTATS 2010},
  year={2010}
}
```

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.390 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/swish_↵ _function.hpp File Reference

Include dependency graph for swish_function.hpp:



Classes

- class **SwishFunction**
 The swish function, defined by.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.390.1 Detailed Description

Author

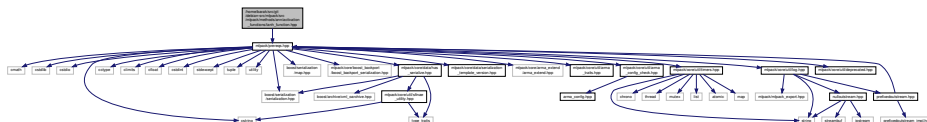
Vivek Pal

Definition and implementation of the Swish function as described by Prajit Ramachandran, Barret Zoph & Quoc V. Le

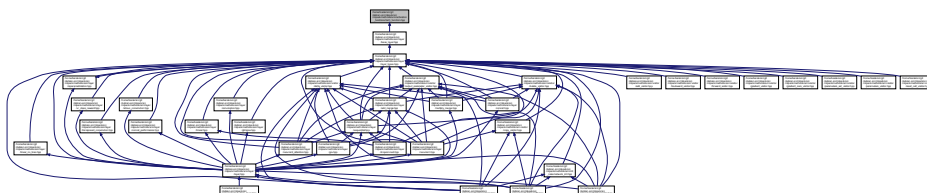
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.391 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation_functions/tanh_function.hpp File Reference

Include dependency graph for tanh_function.hpp:



This graph shows which files directly or indirectly include this file:

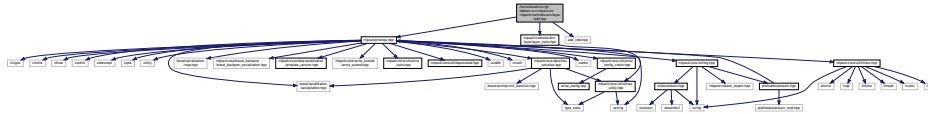


Classes

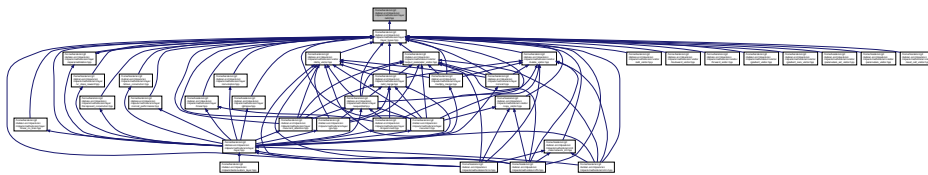
- class **TanhFunction**
The tanh function, defined by.

40.393 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/add.hpp File Reference

Include dependency graph for add.hpp:



This graph shows which files directly or indirectly include this file:



Classes

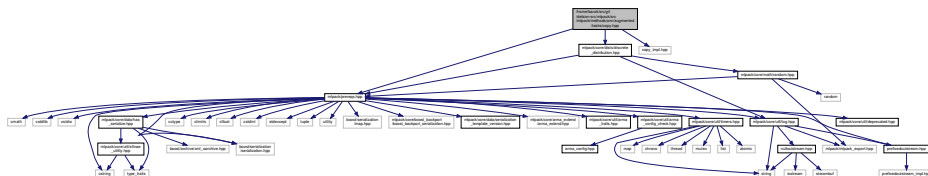
- class **Add**< **InputDataType**, **OutputDataType** >
Implementation of the **Add** (p. 553) module class.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.394 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented/tasks/copy.hpp File Reference

Include dependency graph for copy.hpp:



Functions

- `template<typename MatType >`
`double SequencePrecision (arma::field< MatType > trueOutputs, arma::field< MatType > predOutputs, double tol=1e-4)`

Function that computes the sequences precision (number of correct sequences / number of sequences) of model's answer against ground truth answer.

40.395.1 Detailed Description

Author

Konstantin Sidorov

Definition of scoring functions for sequence prediction problems.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.396 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented/tasks/sort.hpp File Reference

Include dependency graph for sort.hpp:



Classes

- class **SortTask**
Generator of instances of the sequence sort task.

Namespaces

- **mlpack**
`.hpp`
- **mlpack::ann**
Artificial Neural Network.
- **mlpack::ann::augmented**
- **mlpack::ann::augmented::tasks**

40.397.1 Detailed Description

Author

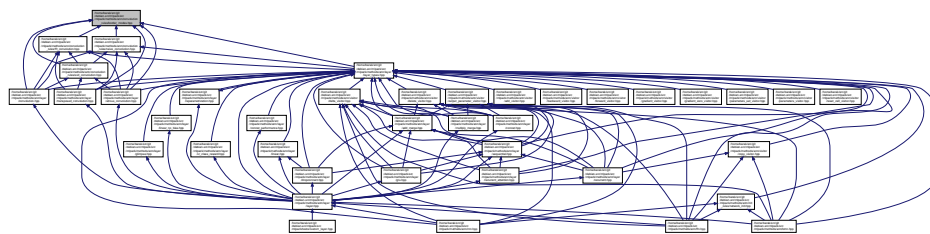
Saksham Bansal

Definition of the BRNN class, which implements bidirectional recurrent neural networks.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.398 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/border_modes.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **FullConvolution**
- class **ValidConvolution**

Namespaces

- **mlpack**
 .hpp
- **mlpack::ann**
 Artificial Neural Network.

40.398.1 Detailed Description

Author

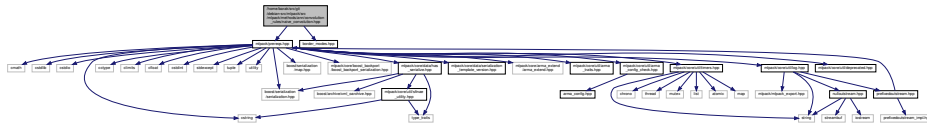
Marcus Edel

This file provides the border modes that can be used to compute different convolutions.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.400 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/naive_convolution.hpp File Reference

Include dependency graph for naive_convolution.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **NaiveConvolution**< **BorderMode** >
Computes the two-dimensional convolution.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.400.1 Detailed Description

Author

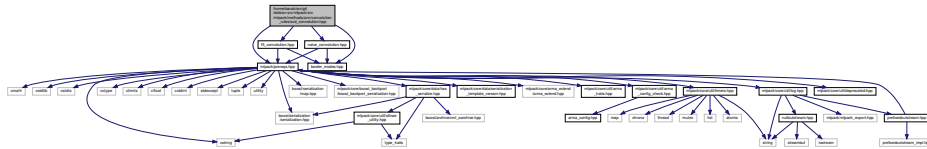
Shangtong Zhang
Marcus Edel

Implementation of the convolution.

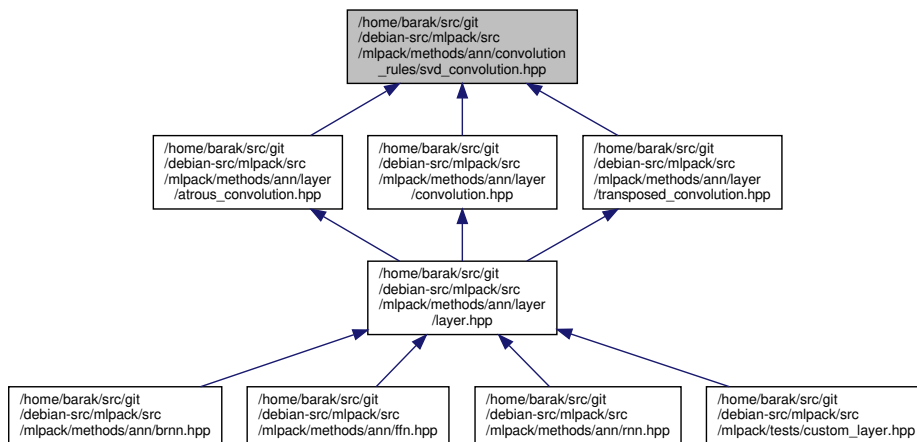
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.401 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution_rules/svd_convolution.hpp File Reference

Include dependency graph for svd_convolution.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **SVDConvolution**< **BorderMode** >
Computes the two-dimensional convolution using singular value decomposition.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.401.1 Detailed Description

Author

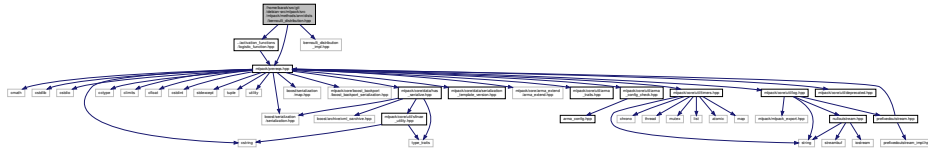
Marcus Edel

Implementation of the convolution using the singular value decomposition to speed up the computation.

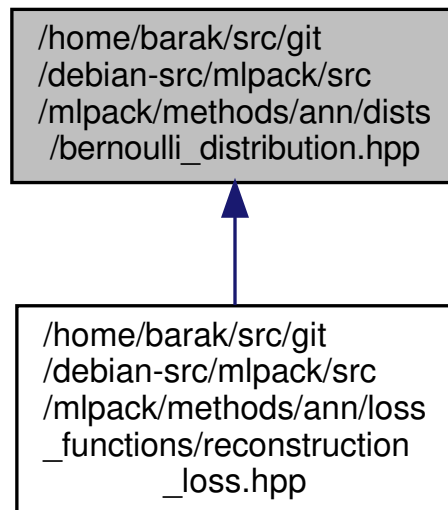
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.402 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/dists/bernoulli_distribution.hpp File Reference

Include dependency graph for bernoulli_distribution.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **BernoulliDistribution**< **DataType** >
Multiple independent Bernoulli distributions.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.402.1 Detailed Description

Author

Atharva Khandait

Definition of the Bernoulli distribution class.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.403 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/ffn.hpp File Reference

Include dependency graph for ffn.hpp:



Classes

- struct **version**< **mlpack::ann::FFN**< **OutputLayerType**, **InitializationRuleType**, **CustomLayer...** > >
- class **FFN**< **OutputLayerType**, **InitializationRuleType**, **CustomLayers** >
Implementation of a standard feed forward network.

Namespaces

- **boost**
Set the serialization version of the adaboost class.
- **boost::serialization**
- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.403.1 Detailed Description

Author

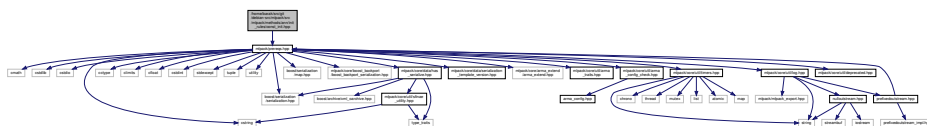
Marcus Edel
Shangtong Zhang

Definition of the FFN class, which implements feed forward neural networks.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

42

Include dependency graph for const_init.hpp:



Classes

- class **ConstInitialization**

This class is used to initialize weight matrix with constant values.

Namespaces

- **mlpack**

.hpp

- **mlpack::ann**

Artificial Neural Network.

40.404.1 Detailed Description

Author

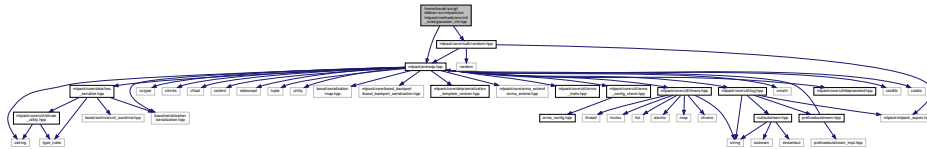
Sumedh Ghaisas

Initialization rule for the neural networks. This simple initialization is performed by assigning a matrix of all constant values to the weight matrix.

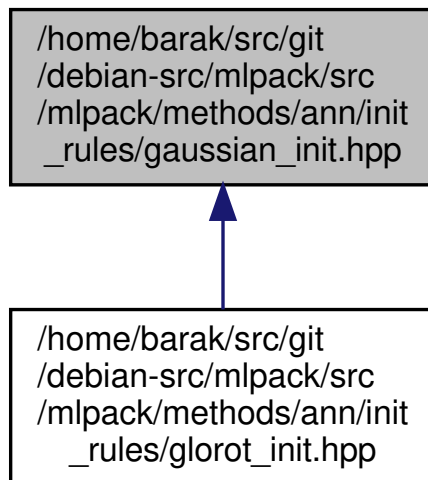
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.405 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/gaussian_init.hpp File Reference

Include dependency graph for gaussian_init.hpp:



This graph shows which files directly or indirectly include this file:



Classes

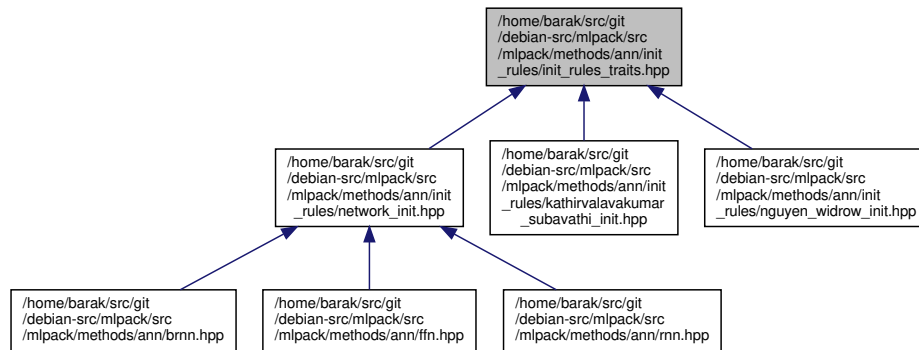
- class **GaussianInitialization**
This class is used to initialize weight matrix with a gaussian.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.408 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/init_rules_traits.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **InitTraits**< **InitRuleType** >

This is a template class that can provide information about various initialization methods.

Namespaces

- **mlpack**

.hpp

- **mlpack::ann**

Artificial Neural Network.

40.408.1 Detailed Description

Author

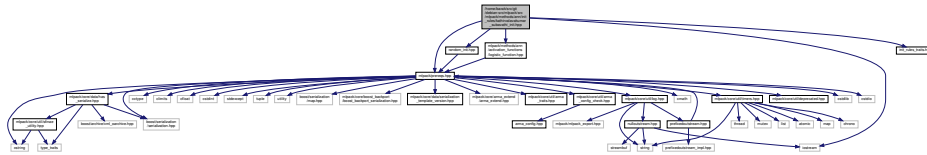
Marcus Edel

This provides the InitTraits class, a template class to get information about various initialization methods.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.409 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/kathirvalavakumar_subavathi_init.hpp File Reference

Include dependency graph for kathirvalavakumar_subavathi_init.hpp:



Classes

- class **InitTraits**< **KathirvalavakumarSubavathiInitialization** >
Initialization traits of the kathirvalavakumar subavathi initialization rule.
- class **KathirvalavakumarSubavathiInitialization**
This class is used to initialize the weight matrix with the method proposed by T.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.409.1 Detailed Description

Author

Marcus Edel

Definition and implementation of the initialization method by T. Kathirvalavakumar and S. Subavathi. This initialization rule is based on sensitivity analysis using cauchy's inequality.

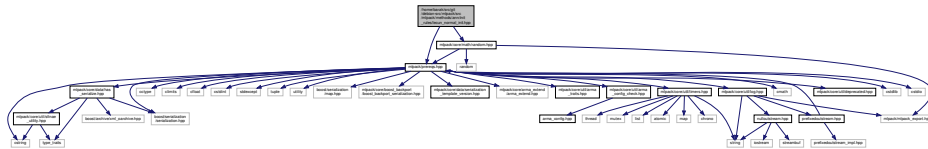
For more information, see the following paper.

```
@inproceedings{KathirvalavakumarJILSA2011,
  title={A New Weight Initialization Method Using Cauchy's Inequality Based
on Sensitivity Analysis},
  author={T. Kathirvalavakumar and S. Subavathi},
  booktitle={Journal of Intelligent Learning Systems and Applications,
Vol. 3 No. 4},
  year={2011}
}
```

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.410 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/lecun_normal_init.hpp File Reference

Include dependency graph for lecu_n_normal_init.hpp:



Classes

- class **LecunNormalInitialization**

This class is used to initialize weight matrix with the Lecun Normalization initialization rule.

Namespaces

- **mlpack**
 `.hpp`
- **mlpack::ann**

Artificial Neural Network.

40.410.1 Detailed Description

Author

Dakshit Agrawal
Prabhat Sharma

Initialization rule given by Lecun et. al. for neural networks and also mentioned in Self Normalizing Networks.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.412 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/nguyen_widrow_init.hpp File Reference

Include dependency graph for `nguyen_widrow_init.hpp`:



Classes

- class **InitTraits**< **NguyenWidrowInitialization** >
Initialization traits of the Nguyen-Widrow initialization rule.
- class **NguyenWidrowInitialization**
This class is used to initialize the weight matrix with the Nguyen-Widrow method.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.412.1 Detailed Description

Author

Marcus Edel

Definition and implementation of the Nguyen-Widrow method. This initialization rule initialize the weights so that the active regions of the neurons are approximately evenly distributed over the input space.

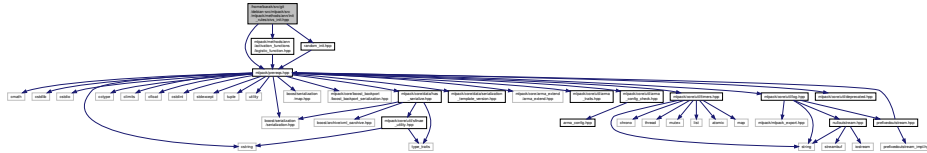
For more information, see the following paper.

```
@inproceedings{NguyenIJCNN1990,
  title={Improving the learning speed of 2-layer neural networks by choosing
  initial values of the adaptive weights},
  booktitle={Neural Networks, 1990., 1990 IJCNN International Joint
  Conference on},
  year={1990}
}
```

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.413 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/oivs_↵ init.hpp File Reference

Include dependency graph for oivs_init.hpp:



Classes

- class **OivsInitialization**< **ActivationFunction** >
This class is used to initialize the weight matrix with the oivs method.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.413.1 Detailed Description

Author

Marcus Edel

Definition and implementation of the Optimal Initial Value Setting method (OIVS). This initialization rule is based on geometrical considerations as described by H. Shimodaira.

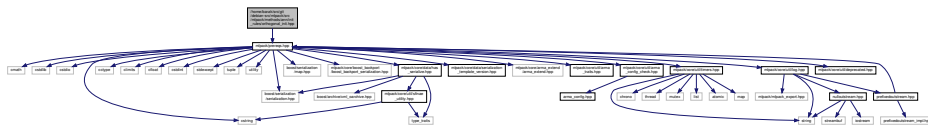
For more information, see the following paper.

```
@inproceedings{ShimodairaICTAI1994,
  title={A weight value initialization method for improving learning
    performance of the backpropagation algorithm in neural networks},
  author={Shimodaira, H.},
  booktitle={Tools with Artificial Intelligence, 1994. Proceedings.,
    Sixth International Conference on},
  year={1994}
}
```

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.414 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/orthogonal_init.hpp File Reference

Include dependency graph for orthogonal_init.hpp:



Classes

- class **OrthogonalInitialization**

This class is used to initialize the weight matrix with the orthogonal matrix initialization.

Namespaces

- **mlpack**
 .hpp
- **mlpack::ann**

Artificial Neural Network.

40.414.1 Detailed Description

Author

Marcus Edel

Definition and implementation of the orthogonal matrix initialization method.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.416.1 Detailed Description

Author

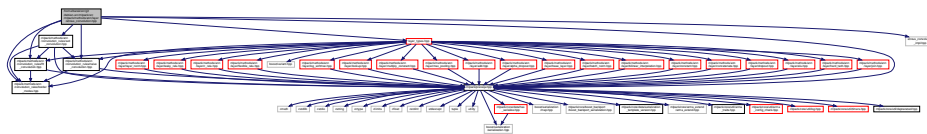
Dakshit Agrawal

Definition of the Alpha-Dropout class, which implements a regularizer that randomly sets units to alpha-dash to prevent them from co-adapting and makes an affine transformation so as to keep the mean and variance of outputs at their original values.

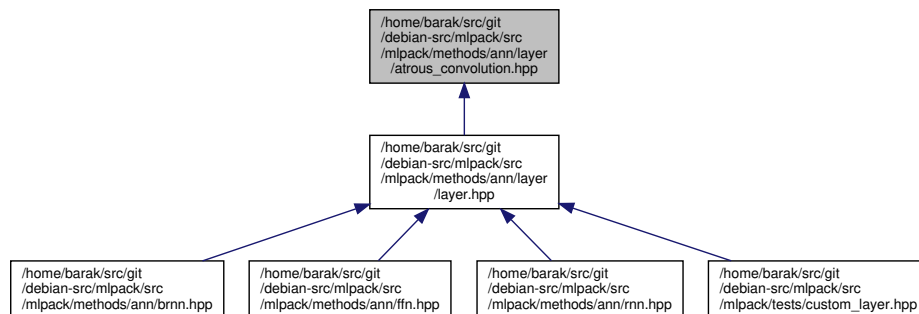
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.417 `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/atrous_convolution.hpp` File Reference

Include dependency graph for `atrous_convolution.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **AtrousConvolution**< **ForwardConvolutionRule**, **BackwardConvolutionRule**, **GradientConvolutionRule**, **InputDataType**, **OutputDataType** >

*Implementation of the Atrous **Convolution** (p. 643) class.*

Namespaces

- **mlpack**
 .hpp
- **mlpack::ann**
 Artificial Neural Network.

40.417.1 Detailed Description

Author

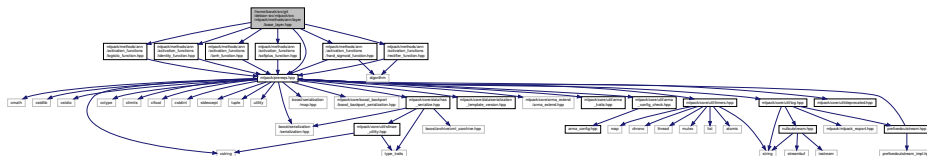
Aarush Gupta
Shikhar Jaiswal

Definition of the Atrous Convolution class.

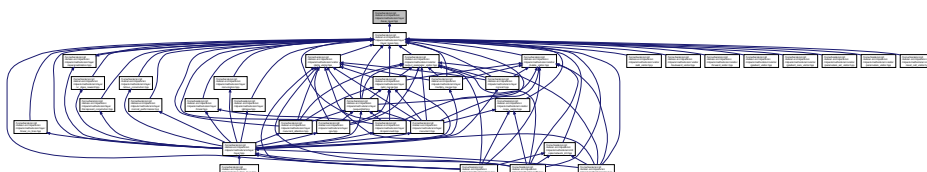
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.418 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/base_layer.hpp File Reference

Include dependency graph for base_layer.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **BaseLayer**< **ActivationFunction**, **InputDataType**, **OutputDataType** >
 Implementation of the base layer.

Namespaces

- **mlpack**
 .hpp
- **mlpack::ann**
 Artificial Neural Network.

Typedefs

- `template<class ActivationFunction = HardSigmoidFunction, typename InputDataType = arma::mat, typename OutputDataType = arma::mat> using HardSigmoidLayer = BaseLayer< ActivationFunction, InputDataType, OutputDataType >`
 Standard HardSigmoid-Layer using the HardSigmoid activation function.
- `template<class ActivationFunction = IdentityFunction, typename InputDataType = arma::mat, typename OutputDataType = arma::mat> using IdentityLayer = BaseLayer< ActivationFunction, InputDataType, OutputDataType >`
 Standard Identity-Layer using the identity activation function.
- `template<class ActivationFunction = RectifierFunction, typename InputDataType = arma::mat, typename OutputDataType = arma::mat> using ReLULayer = BaseLayer< ActivationFunction, InputDataType, OutputDataType >`
 Standard rectified linear unit non-linearity layer.
- `template<class ActivationFunction = LogisticFunction, typename InputDataType = arma::mat, typename OutputDataType = arma::mat> using SigmoidLayer = BaseLayer< ActivationFunction, InputDataType, OutputDataType >`
 Standard Sigmoid-Layer using the logistic activation function.
- `template<class ActivationFunction = SoftplusFunction, typename InputDataType = arma::mat, typename OutputDataType = arma::mat> using SoftPlusLayer = BaseLayer< ActivationFunction, InputDataType, OutputDataType >`
 Standard Softplus-Layer using the Softplus activation function.
- `template<class ActivationFunction = TanhFunction, typename InputDataType = arma::mat, typename OutputDataType = arma::mat> using TanHLayer = BaseLayer< ActivationFunction, InputDataType, OutputDataType >`
 Standard hyperbolic tangent layer.

40.418.1 Detailed Description

Author

Marcus Edel

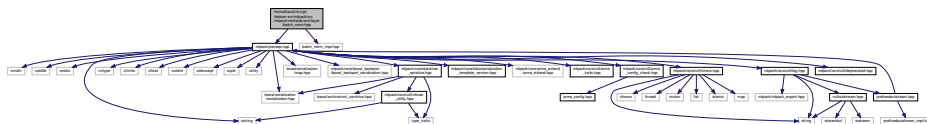
Definition of the BaseLayer class, which attaches various functions to the embedding layer.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

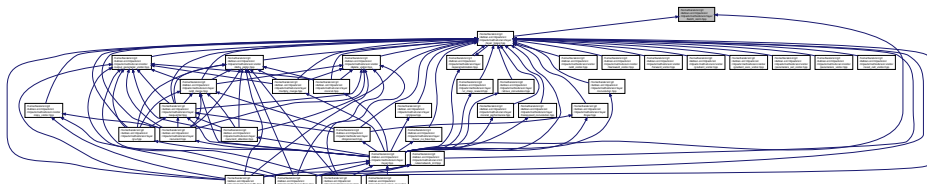
40.419 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/batch_norm.hpp

File Reference

Include dependency graph for batch_norm.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **BatchNorm**< **InputDataType**, **OutputDataType** >
Declaration of the Batch Normalization layer class.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.419.1 Detailed Description

Author

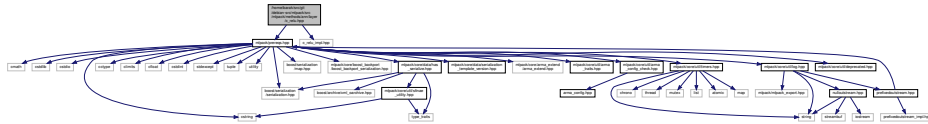
Praveen Ch
Manthan-R-Sheth

Definition of the Batch Normalization layer class.

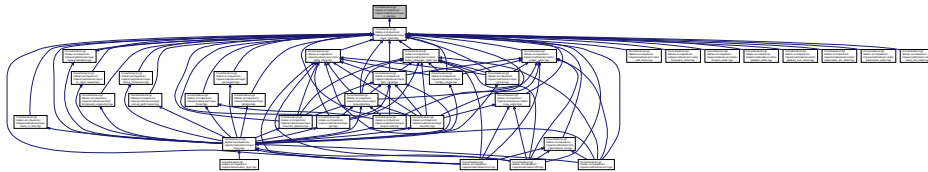
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.421 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/c_relu.hpp File Reference

Include dependency graph for c_relu.hpp:



This graph shows which files directly or indirectly include this file:



Classes

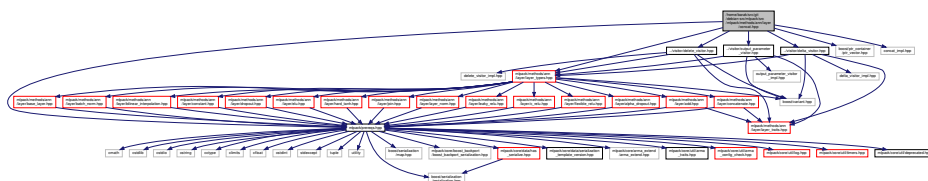
- class **CReLU**< **InputDataType**, **OutputDataType** >
A concatenated ReLU has two outputs, one ReLU and one negative ReLU, concatenated together.

Namespaces

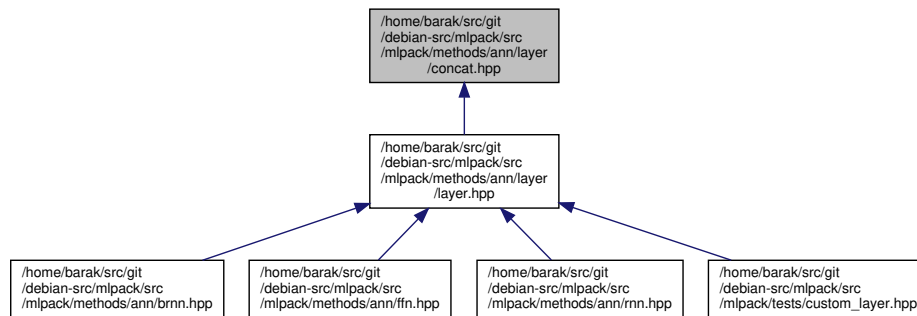
- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.422 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/concat.hpp File Reference

Include dependency graph for concat.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **Concat**< **InputDataType**, **OutputDataType**, **CustomLayers** >

*Implementation of the **Concat** (p. 621) class.*

Namespaces

- **mlpack**

.hpp

- **mlpack::ann**

Artificial Neural Network.

40.422.1 Detailed Description

Author

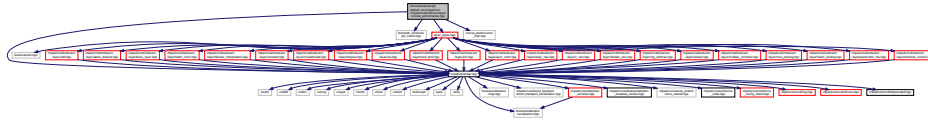
Marcus Edel
Mehul Kumar Nirala

Definition of the Concat class, which acts as a concatenation contain.

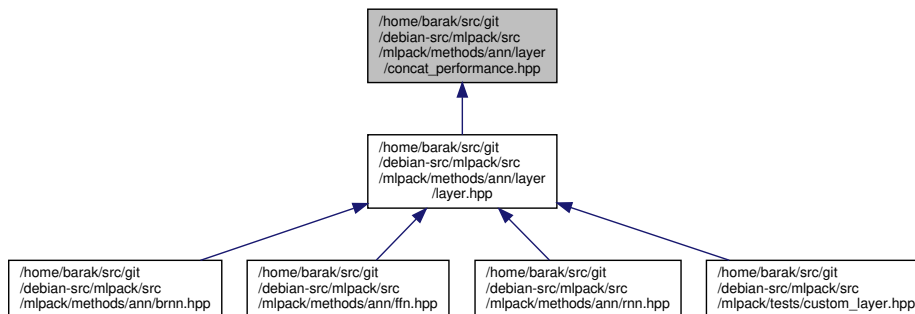
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.423 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/concat_performance.hpp File Reference

Include dependency graph for concat_performance.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **ConcatPerformance**< **OutputLayerType**, **InputDataType**, **OutputDataType** >
Implementation of the concat performance class.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.423.1 Detailed Description

Author

Marcus Edel

Definition of the ConcatPerformance class.

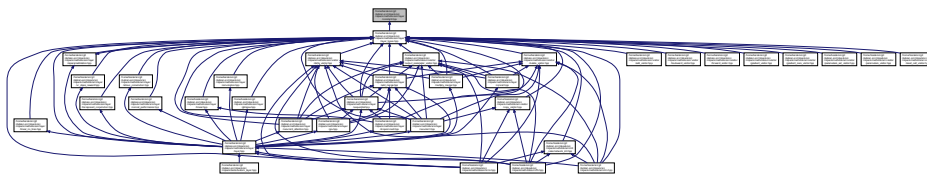
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.425 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/constant.hpp File Reference

Include dependency graph for constant.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **Constant**< **InputDataType**, **OutputDataType** >
Implementation of the constant layer.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.425.1 Detailed Description

Author

Marcus Edel

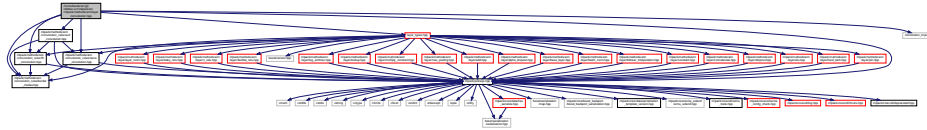
Definition of the Constant class, which outputs a constant value given any input.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

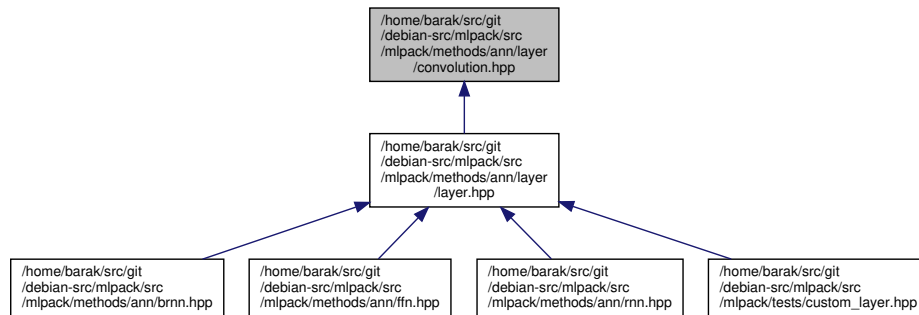
40.426 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/convolution.hpp

File Reference

Include dependency graph for convolution.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **Convolution** < **ForwardConvolutionRule**, **BackwardConvolutionRule**, **GradientConvolutionRule**, **InputDataType**, **OutputDataType** >
*Implementation of the **Convolution** (p. 643) class.*

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.426.1 Detailed Description

Author

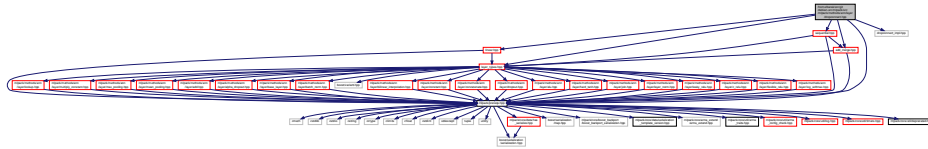
Marcus Edel

Definition of the Convolution module class.

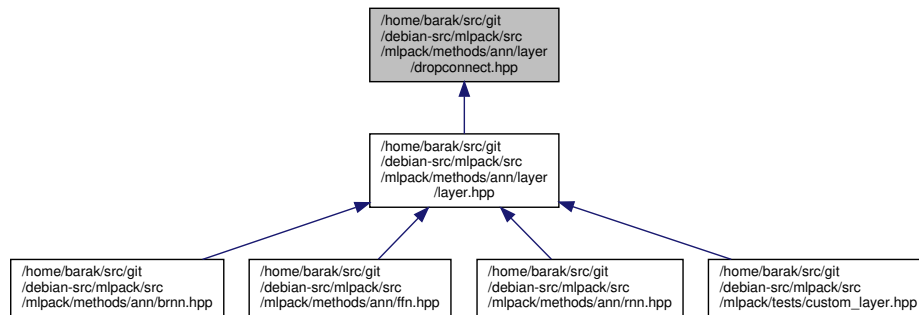
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.427 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/dropconnect.hpp File Reference

Include dependency graph for dropconnect.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **DropConnect**< **InputDataType**, **OutputDataType** >

The **DropConnect** (p. 666) layer is a regularizer that randomly with probability ratio sets the connection values to zero and scales the remaining elements by factor $1/(1 - \text{ratio})$.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.427.1 Detailed Description

Author

Palash Ahuja
Marcus Edel

Definition of the DropConnect class, which implements a regularizer that randomly sets connections to zero. Preventing units from co-adapting.

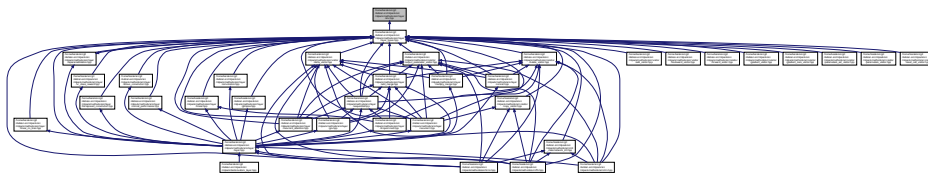
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.429 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/elu.hpp File Reference

Include dependency graph for elu.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **ELU**< **InputDataType**, **OutputDataType** >
The **ELU** (p. 680) activation function, defined by.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

Typedefs

- using **SELU** = ELU< arma::mat, arma::mat >

40.429.1 Detailed Description

Author

Vivek Pal
Dakshit Agrawal

Definition of the ELU activation function as described by Djork-Arne Clevert, Thomas Unterthiner and Sepp Hochreiter.

Definition of the SELU function as introduced by Klambauer et. al. in Self Neural Networks. The SELU activation function keeps the mean and variance of the input invariant.

In short, SELU = lambda * ELU, with 'alpha' and 'lambda' fixed for normalized inputs.

Hence both ELU and SELU are implemented in the same file, with lambda = 1 for ELU function.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Include dependency graph for fast_lstm.hpp:



```

graph TD
    A["/home/barak/src/git  
/debian-src/mlpack/src  
mlpack/methods/ann/brnn.hpp"] --> D["/home/barak/src/git  
/debian-src/mlpack/src  
mlpack/methods/ann/layer.hpp  
layer.hpp"]
    B["/home/barak/src/git  
/debian-src/mlpack/src  
mlpack/methods/ann/layer.hpp"] --> D
    C["/home/barak/src/git  
/debian-src/mlpack/src  
mlpack/methods/ann/rnn.hpp"] --> D
    E["/home/barak/src/git  
/debian-src/mlpack/src  
mlpack/tests/custom_layer.hpp"] --> D
    D --> F["/home/barak/src/git  
/debian-src/mlpack/src  
mlpack/methods/ann/layer  
fast_lstm.hpp"]
  
```

- class **FastLSTM**< **InputDataType**, **OutputDataType** >
*An implementation of a faster version of the Fast **LSTM** (p. 801) network layer.*

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

Marcus Edel

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

- class **FlexibleReLU**< **InputDataType**, **OutputDataType** >
*The **FlexibleReLU** (p. 707) activation function, defined by.*

- **mlpack**
 .hpp
- **mlpack::ann**
 Artificial Neural Network.

Aarush Gupta
Manthan-R-Sheth

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

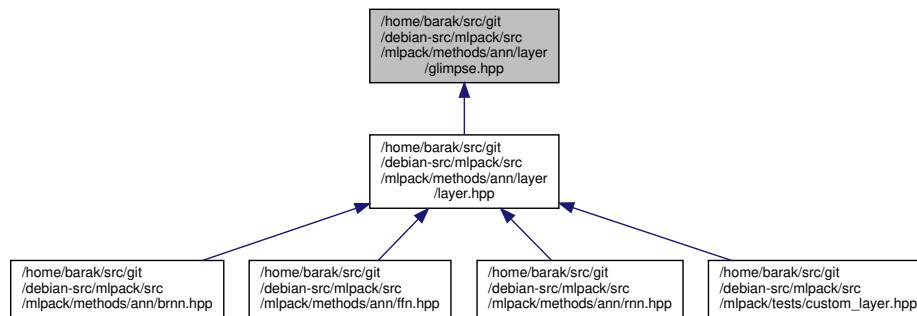
40.432 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/glimpse.hpp

File Reference

Include dependency graph for glimpse.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **Glimpse**< **InputDataType**, **OutputDataType** >

The glimpse layer returns a retina-like representation (down-scaled cropped images) of increasing scale around a given location in a given image.

- class **MeanPoolingRule**

Namespaces

- **mlpack**

.hpp

- **mlpack::ann**

Artificial Neural Network.

40.432.1 Detailed Description

Author

Marcus Edel

Definition of the GlimpseLayer class, which takes an input image and a location to extract a retina-like representation of the input image at different increasing scales.

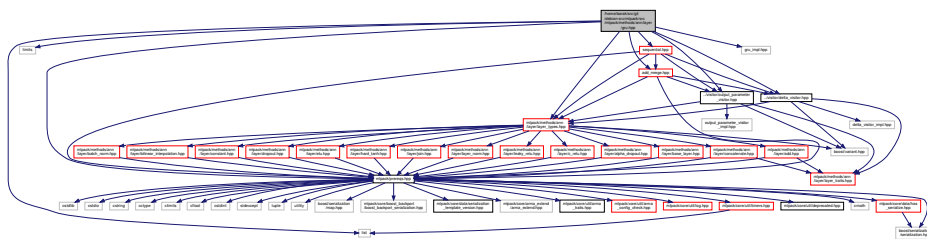
For more information, see the following.

```
@article{CoRR2014,
  author = {Volodymyr Mnih, Nicolas Heess, Alex Graves, Koray Kavukcuoglu},
  title = {Recurrent Models of Visual Attention},
  journal = {CoRR},
  volume = {abs/1406.6247},
  year = {2014},
}
```

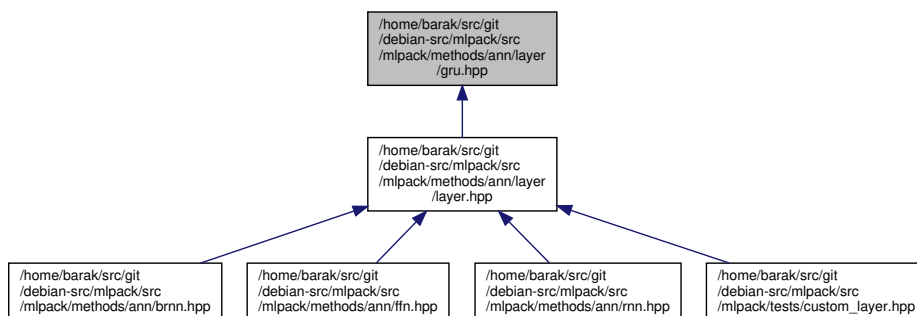
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.433 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/gru.hpp File Reference

Include dependency graph for gru.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **GRU**< **InputDataType**, **OutputDataType** >
An implementation of a gru network layer.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.433.1 Detailed Description

Author

Sumedh Ghaisas

Definition of the GRU layer.

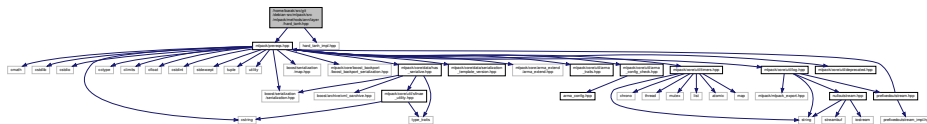
For more information, read the following paper:

```
@inproceedings{chung2015gated,
  title   = {Gated Feedback Recurrent Neural Networks.},
  author  = {Chung, Junyoung and G{\u}l{\c}ehre, Caglar and Cho,
            Kyunghyun and Bengio, Yoshua},
  booktitle = {ICML},
  pages   = {2067--2075},
  year    = {2015}
}
```

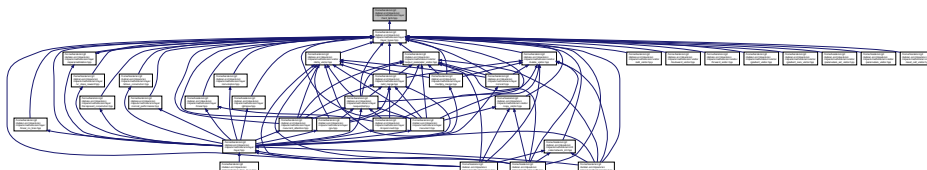
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.434 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/hard_tanh.hpp File Reference

Include dependency graph for hard_tanh.hpp:



This graph shows which files directly or indirectly include this file:



- class **HardTanH**< **InputDataType**, **OutputDataType** >
The Hard Tanh activation function, defined by.

- **mlpack**
 .hpp
- **mlpack::ann**
 Artificial Neural Network.

Author

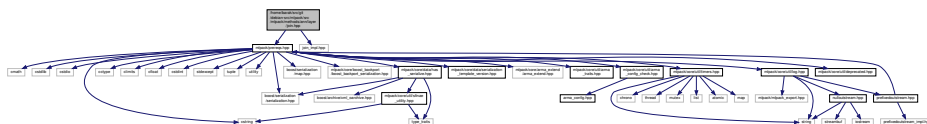
Dhawal Arora

Definition and implementation of the HardTanH layer.

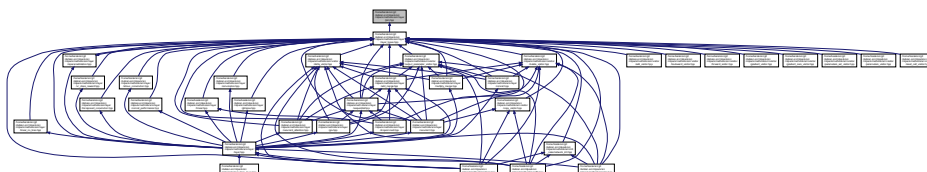
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.435 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/join.hpp File Reference

Include dependency graph for join.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **Join**< **InputDataType**, **OutputDataType** >

Implementation of the **Join** (p. 753) module class.

Namespaces

- **mlpack**

.hpp

- **mlpack::ann**

Artificial Neural Network.

40.435.1 Detailed Description

Author

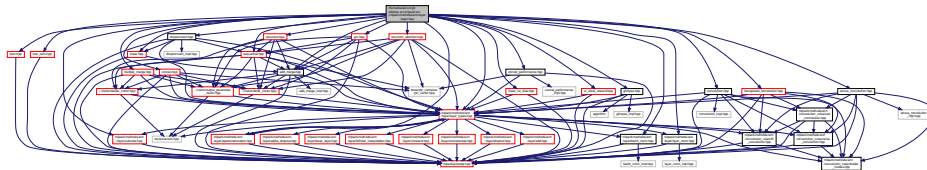
Marcus Edel

Definition of the Join module.

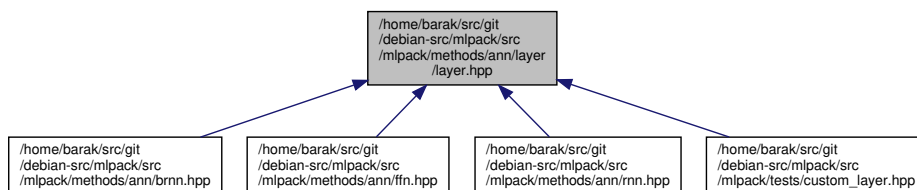
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.436 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/layer.hpp File Reference

Include dependency graph for layer.hpp:



This graph shows which files directly or indirectly include this file:



40.436.1 Detailed Description

Author

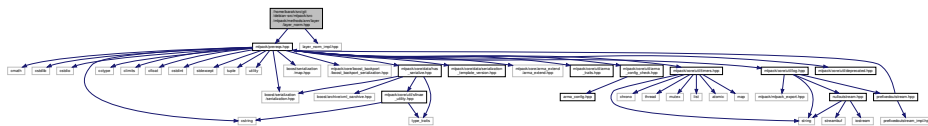
Marcus Edel

This includes various layers to construct a model.

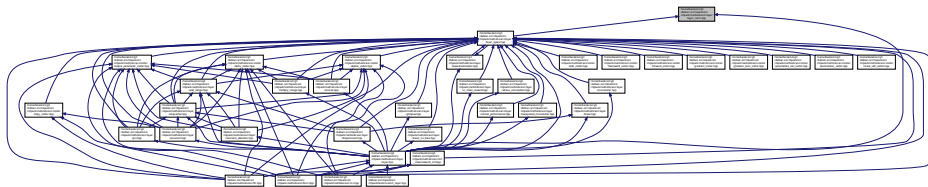
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.437 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/layer_norm.hpp File Reference

Include dependency graph for layer_norm.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **LayerNorm**< **InputDataType**, **OutputDataType** >
Declaration of the Layer Normalization class.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.437.1 Detailed Description

Author

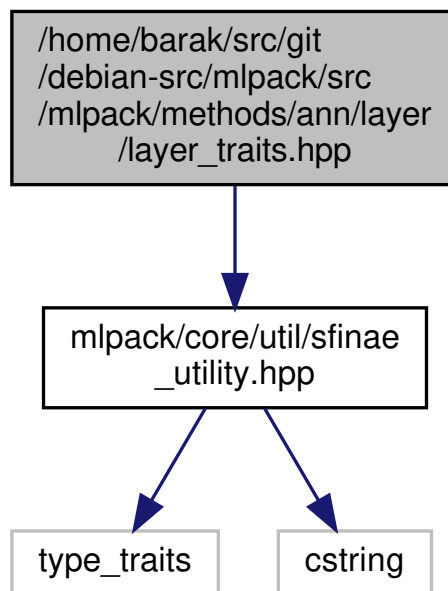
Shikhar Jaiswal

Definition of the Layer Normalization class.

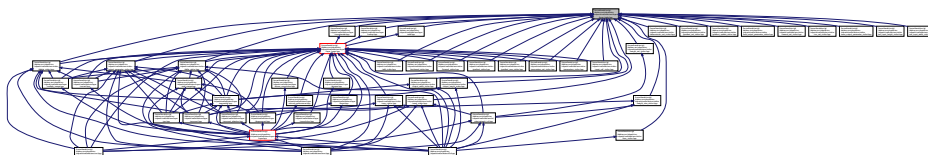
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.438 `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/layer_traits.hpp` File Reference

Include dependency graph for `layer_traits.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **LayerTraits**< **LayerType** >

This is a template class that can provide information about various layers.

Namespaces

- **mlpack**

.hpp

- **mlpack::ann**

Artificial Neural Network.

Functions

- **HAS_ANY_METHOD_FORM** (Model, HasModelCheck)
- **HAS_MEM_FUNC** (Gradient, HasGradientCheck)
- **HAS_MEM_FUNC** (Deterministic, HasDeterministicCheck)
- **HAS_MEM_FUNC** (Parameters, HasParametersCheck)
- **HAS_MEM_FUNC** (Add, HasAddCheck)
- **HAS_MEM_FUNC** (Location, HasLocationCheck)
- **HAS_MEM_FUNC** (Reset, HasResetCheck)
- **HAS_MEM_FUNC** (ResetCell, HasResetCellCheck)
- **HAS_MEM_FUNC** (Reward, HasRewardCheck)
- **HAS_MEM_FUNC** (InputWidth, HasInputWidth)
- **HAS_MEM_FUNC** (InputHeight, HasInputHeight)
- **HAS_MEM_FUNC** (Rho, HasRho)
- **HAS_MEM_FUNC** (Loss, HasLoss)
- **HAS_MEM_FUNC** (Run, HasRunCheck)

40.438.1 Detailed Description

Author

Marcus Edel

This provides the LayerTraits class, a template class to get information about various layers.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.439 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/layer_types.hpp

File Reference

Include dependency graph for layer_types.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **AddMerge**< **InputDataType**, **OutputDataType**, **CustomLayers** >
Implementation of the **AddMerge** (p. 558) module class.
- class **AtrousConvolution**< **ForwardConvolutionRule**, **BackwardConvolutionRule**, **GradientConvolutionRule**, **InputDataType**, **OutputDataType** >
Implementation of the Atrous **Convolution** (p. 643) class.
- class **BatchNorm**< **InputDataType**, **OutputDataType** >
Declaration of the Batch Normalization layer class.
- class **Concat**< **InputDataType**, **OutputDataType**, **CustomLayers** >
Implementation of the **Concat** (p. 621) class.
- class **Concatenate**< **InputDataType**, **OutputDataType** >
Implementation of the **Concatenate** (p. 630) module class.
- class **ConcatPerformance**< **OutputLayerType**, **InputDataType**, **OutputDataType** >
Implementation of the concat performance class.
- class **Convolution**< **ForwardConvolutionRule**, **BackwardConvolutionRule**, **GradientConvolutionRule**, **InputDataType**, **OutputDataType** >
Implementation of the **Convolution** (p. 643) class.
- class **DropConnect**< **InputDataType**, **OutputDataType** >
The **DropConnect** (p. 666) layer is a regularizer that randomly with probability ratio sets the connection values to zero and scales the remaining elements by factor $1/(1 - \text{ratio})$.
- class **FastLSTM**< **InputDataType**, **OutputDataType** >
An implementation of a faster version of the Fast **LSTM** (p. 801) network layer.
- class **Glimpse**< **InputDataType**, **OutputDataType** >
The glimpse layer returns a retina-like representation (down-scaled cropped images) of increasing scale around a given location in a given image.
- class **GRU**< **InputDataType**, **OutputDataType** >
An implementation of a gru network layer.
- class **LayerNorm**< **InputDataType**, **OutputDataType** >

Declaration of the Layer Normalization class.

- class **Linear**< **InputDataType**, **OutputDataType** >
*Implementation of the **Linear** (p. 776) layer class.*
- class **LinearNoBias**< **InputDataType**, **OutputDataType** >
*Implementation of the **LinearNoBias** (p. 782) class.*
- class **LSTM**< **InputDataType**, **OutputDataType** >
*Implementation of the **LSTM** (p. 801) module class.*
- class **MultiplyMerge**< **InputDataType**, **OutputDataType**, **CustomLayers** >
*Implementation of the **MultiplyMerge** (p. 829) module class.*
- class **Recurrent**< **InputDataType**, **OutputDataType**, **CustomLayers** >
Implementation of the RecurrentLayer class.
- class **RecurrentAttention**< **InputDataType**, **OutputDataType** >
*This class implements the **Recurrent** (p. 886) Model for Visual Attention, using a variety of possible layer implementations.*
- class **Reparametrization**< **InputDataType**, **OutputDataType** >
*Implementation of the **Reparametrization** (p. 903) layer class.*
- class **Sequential**< **InputDataType**, **OutputDataType**, **Residual**, **CustomLayers** >
*Implementation of the **Sequential** (p. 927) class.*
- class **TransposedConvolution**< **ForwardConvolutionRule**, **BackwardConvolutionRule**, **GradientConvolutionRule**, **InputDataType**, **OutputDataType** >
*Implementation of the Transposed **Convolution** (p. 643) class.*
- class **VRClassReward**< **InputDataType**, **OutputDataType** >
Implementation of the variance reduced classification reinforcement layer.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

Typedefs

- template<typename... CustomLayers>
using **LayerTypes** = boost::variant< Add< arma::mat, arma::mat > *, AddMerge< arma::mat, arma::mat > *, AtrousConvolution< NaiveConvolution< ValidConvolution >, NaiveConvolution< FullConvolution >, NaiveConvolution< ValidConvolution >, arma::mat, arma::mat > *, BaseLayer< LogisticFunction, arma::mat, arma::mat > *, BaseLayer< IdentityFunction, arma::mat, arma::mat > *, BaseLayer< TanhFunction, arma::mat, arma::mat > *, BaseLayer< RectifierFunction, arma::mat, arma::mat > *, BaseLayer< SoftplusFunction, arma::mat, arma::mat > *, BatchNorm< arma::mat, arma::mat > *, BilinearInterpolation< arma::mat, arma::mat > *, Concat< arma::mat, arma::mat > *, Concatenate< arma::mat, arma::mat > *, ConcatPerformance< NegativeLogLikelihood< arma::mat, arma::mat >, arma::mat, arma::mat > *, Constant< arma::mat, arma::mat > *, Convolution< NaiveConvolution< ValidConvolution >, NaiveConvolution< FullConvolution >, NaiveConvolution< ValidConvolution >, arma::mat, arma::mat > *, TransposedConvolution< NaiveConvolution< ValidConvolution >, NaiveConvolution< FullConvolution >, NaiveConvolution< ValidConvolution >, arma::mat, arma::mat > *, DropConnect< arma::mat, arma::mat > *, Dropout< arma::mat, arma::mat > *, AlphaDropout< arma::mat, arma::mat > *, ELU< arma::mat, arma::mat > *, FlexibleReLU< arma::mat, arma::mat

Namespaces

- **mlpack**
 .hpp
- **mlpack::ann**
 Artificial Neural Network.

40.440.1 Detailed Description

Author

Dhawal Arora

Definition of LeakyReLU layer first introduced in the acoustic model, Andrew L. Maas, Awni Y. Hannun, Andrew Y. Ng, "Rectifier Nonlinearities Improve Neural Network Acoustic Models", 2014

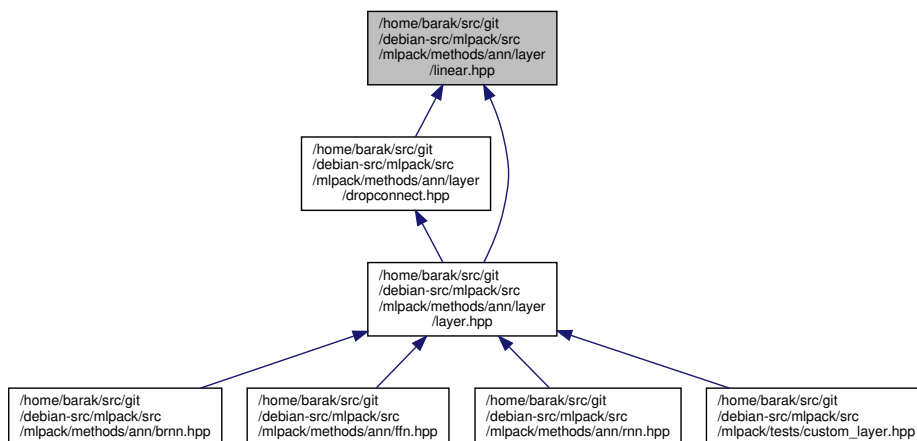
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.441 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear.hpp File Reference

Include dependency graph for linear.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **Lookup**< InputDataType, OutputDataType >
Implementation of the **Lookup** (p. 795) class.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

Typedefs

- `template<typename MatType = arma::mat>`
`using Embedding = Lookup< MatType, MatType >`

40.444.1 Detailed Description

Author

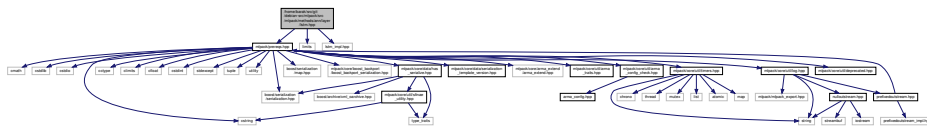
Marcus Edel

Definition of the Lookup class a particular convolution, where the width of the convolution is 1.

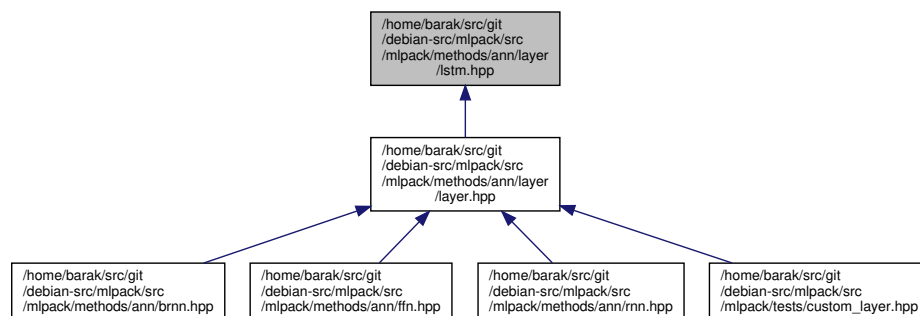
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.445 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/lstm.hpp File Reference

Include dependency graph for lstm.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **MeanPooling**< **InputDataType**, **OutputDataType** >
*Implementation of the **MeanPooling** (p. 814).*

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.447.1 Detailed Description

Author

Marcus Edel
Nilay Jain

Definition of the MeanPooling layer class.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.448 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/multiply_↵ constant.hpp File Reference

Include dependency graph for multiply_constant.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **MultiplyConstant**< **InputDataType**, **OutputDataType** >
Implementation of the multiply constant layer.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.448.1 Detailed Description

Author

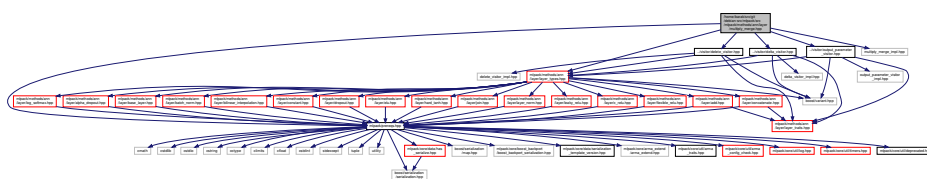
Marcus Edel

Definition of the MultiplyConstantLayer class, which multiplies the input by a (non-learnable) constant.

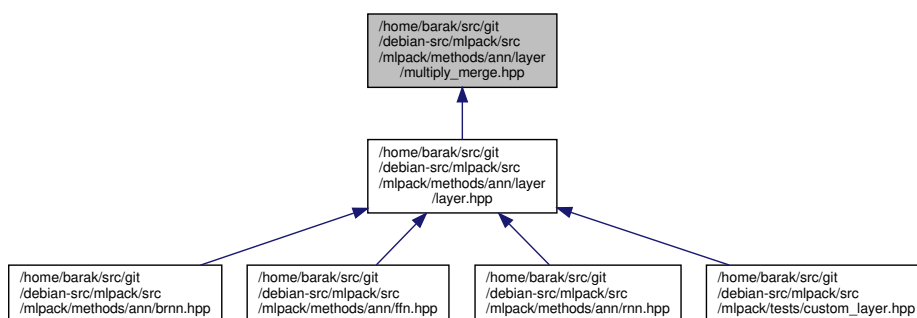
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.449 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/multiply_merge.hpp File Reference

Include dependency graph for multiply_merge.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **PReLU**< **InputDataType**, **OutputDataType** >
*The **PReLU** (p. 855) activation function, defined by (where alpha is trainable)*

Namespaces

- mlpack**
.hpp
- mlpack::ann**
Artificial Neural Network.

40.450.1 Detailed Description

Author

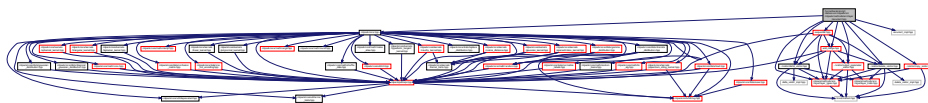
Prasanna Patil

Definition of PReLU layer first introduced in the, Kaiming He, Xiangyu Zhang, Shaoqing, Ren Jian Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification", 2014

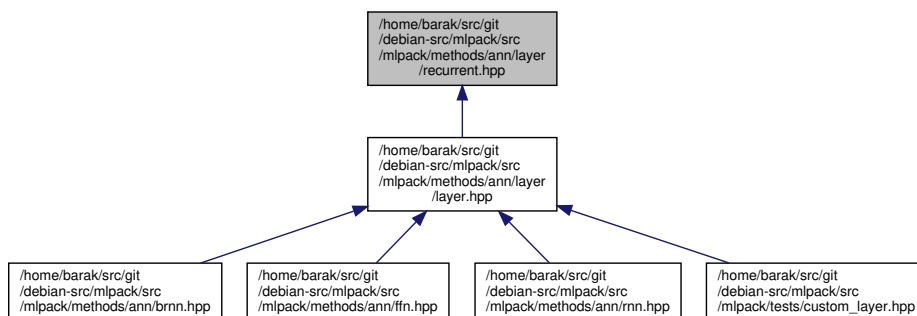
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.451 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/recurrent.hpp File Reference

Include dependency graph for recurrent.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RecurrentAttention**< **InputDataType**, **OutputDataType** >

This class implements the **Recurrent** (p. 886) Model for Visual Attention, using a variety of possible layer implementations.

Namespaces

- mlpack**
.hpp
- mlpack::ann**

Artificial Neural Network.

40.452.1 Detailed Description

Author

Marcus Edel

Definition of the RecurrentAttention class.

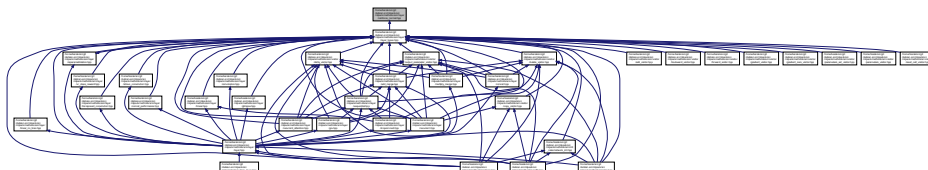
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.453 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/reinforce_normal.hpp File Reference

Include dependency graph for reinforce_normal.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **ReinforceNormal**< **InputDataType**, **OutputDataType** >
Implementation of the reinforce normal layer.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.453.1 Detailed Description

Author

Marcus Edel

Definition of the ReinforceNormalLayer class, which implements the REINFORCE algorithm for the normal distribution.

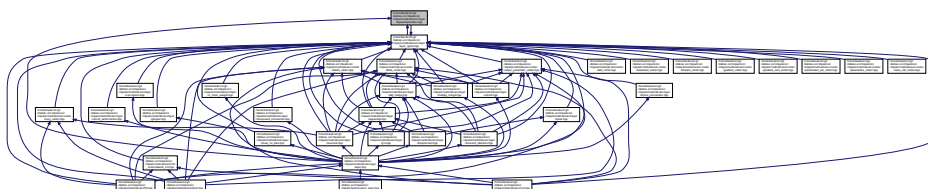
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.454 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/reparametrization.hpp File Reference

Include dependency graph for reparametrization.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **Reparametrization**< **InputDataType**, **OutputDataType** >
Implementation of the **Reparametrization** (p. 903) layer class.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.454.1 Detailed Description

Author

Atharva Khandait

Definition of the Reparametrization layer class which samples from a gaussian distribution.

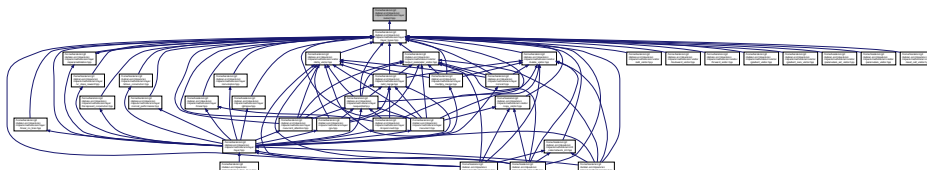
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.455 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/select.hpp File Reference

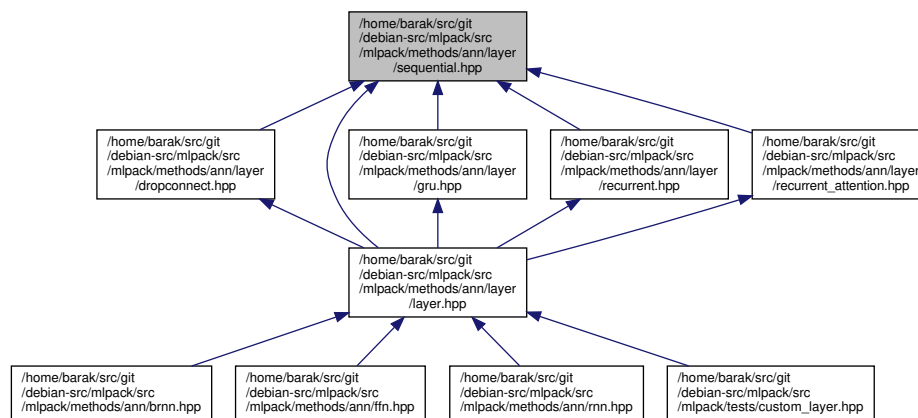
Include dependency graph for select.hpp:



This graph shows which files directly or indirectly include this file:



This graph shows which files directly or indirectly include this file:



Classes

- class **Sequential**< **InputDataType**, **OutputDataType**, **Residual**, **CustomLayers** >
Implementation of the **Sequential** (p. 927) class.

Namespaces

- mlpack**
.hpp
- mlpack::ann**
Artificial Neural Network.

Typedefs

- template<typename InputDataType = arma::mat, typename OutputDataType = arma::mat, typename... CustomLayers>
using **Residual** = Sequential< InputDataType, OutputDataType, true, CustomLayers... >

40.456.1 Detailed Description

Author

Marcus Edel

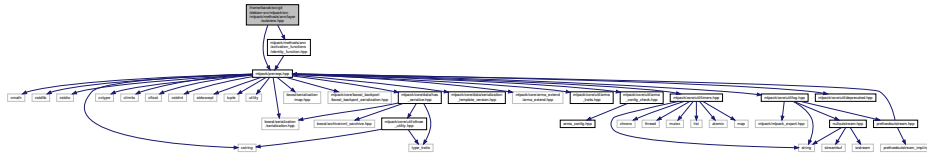
Definition of the Sequential class, which acts as a feed-forward fully connected network container.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

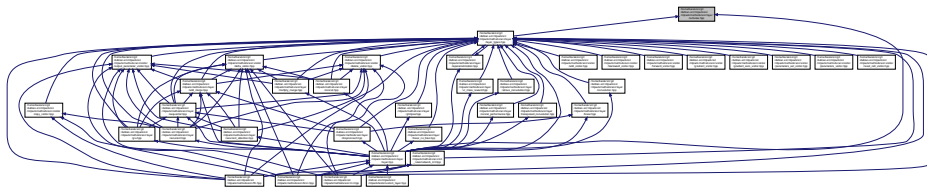
40.457 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/subview.hpp

File Reference

Include dependency graph for subview.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **Subview**< **InputDataType**, **OutputDataType** >
Implementation of the subview layer.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.457.1 Detailed Description

Author

Haritha Nair

Definition of the Subview class, which modifies the input as necessary.

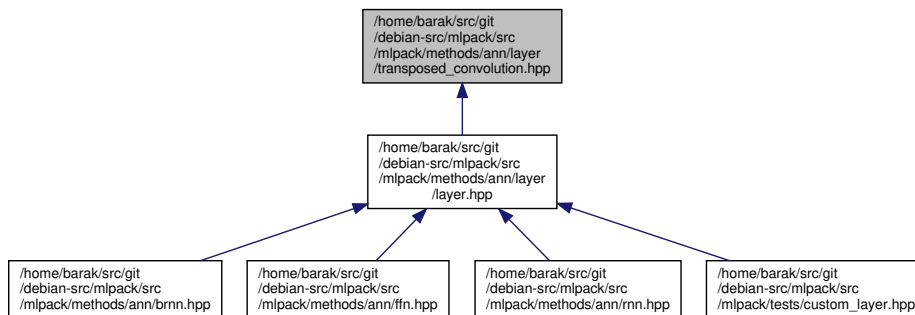
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.458 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/transposed_convolution.hpp File Reference

Include dependency graph for transposed_convolution.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **TransposedConvolution**< **ForwardConvolutionRule**, **BackwardConvolutionRule**, **GradientConvolutionRule**, **InputDataType**, **OutputDataType** >
Implementation of the Transposed **Convolution** (p. 643) class.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.458.1 Detailed Description

Author

Shikhar Jaiswal
Marcus Edel

Definition of the Transposed Convolution module class.

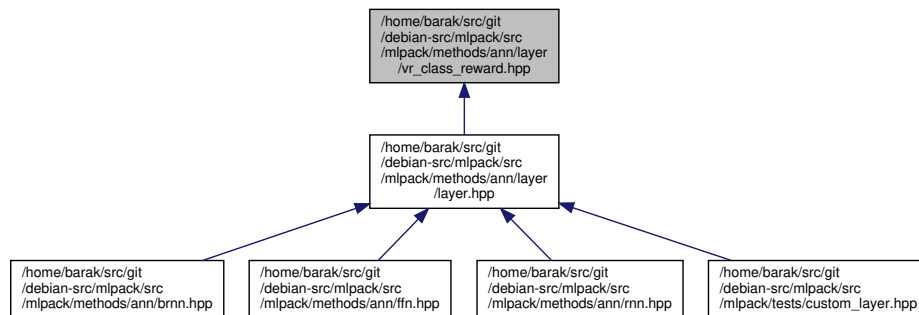
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.459 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/vr_class_reward.hpp File Reference

Include dependency graph for vr_class_reward.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **VRClassReward**< **InputDataType**, **OutputDataType** >
Implementation of the variance reduced classification reinforcement layer.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.459.1 Detailed Description

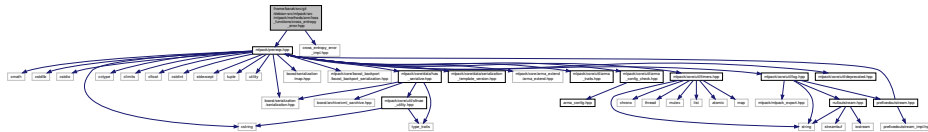
Author

Marcus Edel

Definition of the VRClassReward class, which implements the variance reduced classification reinforcement layer.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Include dependency graph for cross_entropy_error.hpp:



- class **CrossEntropyError**< InputDataType, OutputDataType >
The cross-entropy performance function measures the network's performance according to the cross-entropy between the input and target distributions.

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

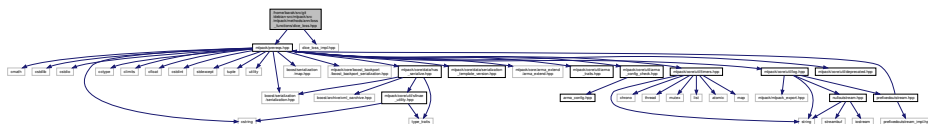
Author

Konstantin Sidorov

Definition of the cross-entropy performance function.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpac. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Include dependency graph for dice_loss.hpp:



Classes

- class **SigmoidCrossEntropyError**< **InputDataType**, **OutputDataType** >

The **SigmoidCrossEntropyError** (p. 937) performance function measures the network's performance according to the cross-entropy function between the input and target distributions.

Namespaces

- **mlpack**

.hpp

- **mlpack::ann**

Artificial Neural Network.

40.467.1 Detailed Description

Author

Kris Singh
Shikhar Jaiswal

Definition of the cross-entropy with logit performance function.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.468 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/rbm/rbm.hpp File Reference

Include dependency graph for rbm.hpp:



Classes

- class **RBM**< **InitializationRuleType**, **DataType**, **PolicyType** >

The implementation of the **RBM** (p. 864) module.

Namespaces

- **mlpack**
 .hpp
- **mlpack::ann**
 Artificial Neural Network.

40.468.1 Detailed Description

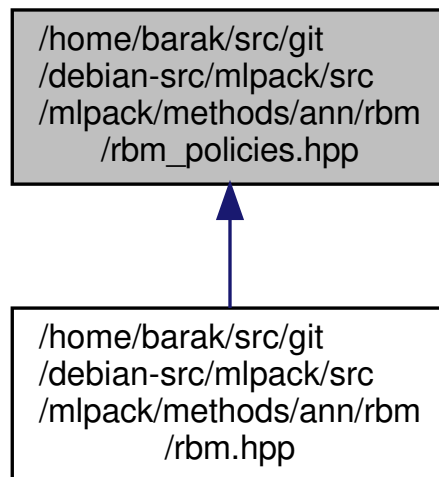
Author

Kris Singh
Shikhar Jaiswal

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.469 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/rbm/rbm_policies.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **BinaryRBM**
 For more information, see the following paper:
- class **SpikeSlabRBM**
 For more information, see the following paper:

40.470.1 Detailed Description

Author

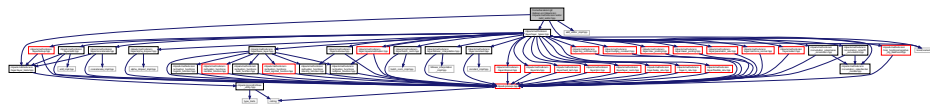
Marcus Edel

Definition of the RNN class, which implements recurrent neural networks.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.471 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/add_visitor.hpp File Reference

Include dependency graph for add_visitor.hpp:



Classes

- class **AddVisitor**< **CustomLayers** >
AddVisitor (p. 565) exposes the *Add()* method of the given module.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.471.1 Detailed Description

Author

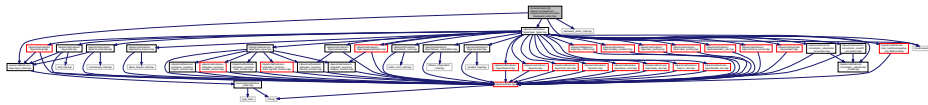
Marcus Edel

This file provides an abstraction for the *Add()* function for different layers and automatically directs any parameter to the right layer type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.472 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/backward_visitor.hpp File Reference

Include dependency graph for backward_visitor.hpp:



Classes

- class **BackwardVisitor**

***BackwardVisitor** (p. 587) executes the Backward() function given the input, error and delta parameter.*

Namespaces

- **mlpack**

.hpp

- **mlpack::ann**

Artificial Neural Network.

40.472.1 Detailed Description

Author

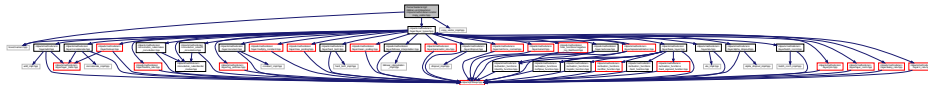
Marcus Edel

This file provides an abstraction for the Backward() function for different layers and automatically directs any parameter to the right layer type.

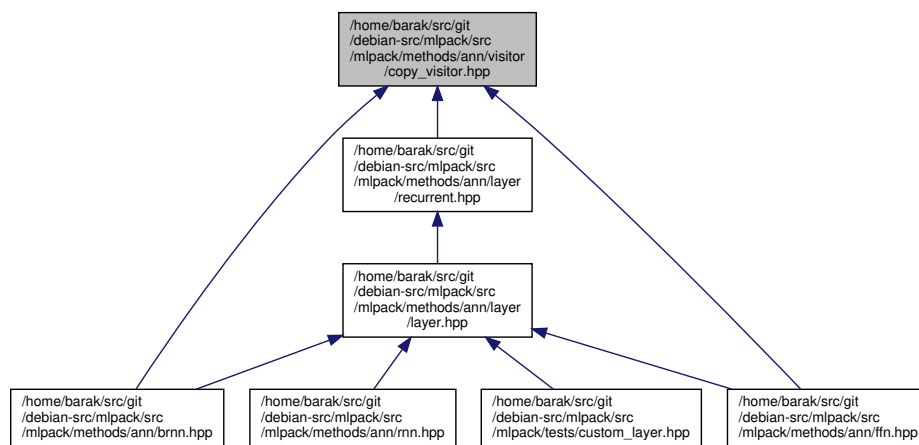
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.473 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/copy_visitor.hpp File Reference

Include dependency graph for copy_visitor.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **CopyVisitor**< **CustomLayers** >
This visitor is to support copy constructor for neural network module.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.473.1 Detailed Description

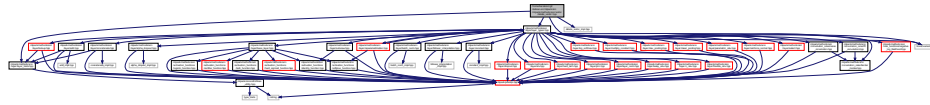
Author

Shangtong Zhang

This file provides an abstraction for copy between layers.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Include dependency graph for delete_visitor.hpp:



- class **DeleteVisitor**

DeleteVisitor (p. 659) executes the destructor of the instantiated object.

- **mlpack**

.hpp

- **mlpack::ann**

Artificial Neural Network.

Author

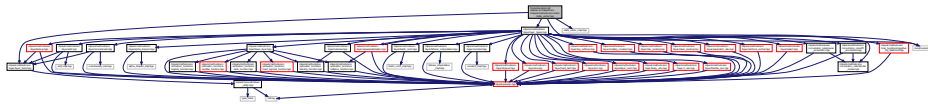
Marcus Edel

This file provides an abstraction for the Delete() function for different layers and automatically directs any parameter to the right layer type.

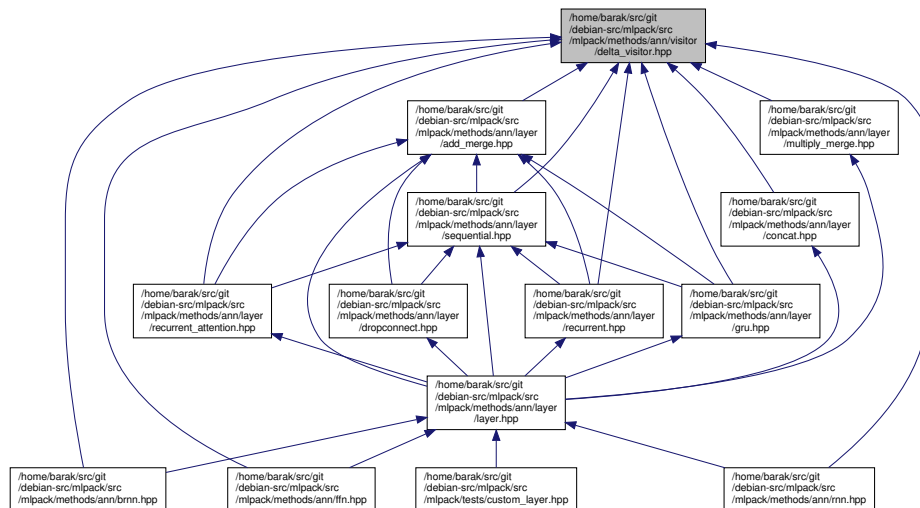
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.475 `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/delta_visitor.hpp` File Reference

Include dependency graph for `delta_visitor.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **DeltaVisitor**

***DeltaVisitor** (p. 660) exposes the delta parameter of the given module.*

Namespaces

- **mlpack**

.hpp

- **mlpack::ann**

Artificial Neural Network.

40.475.1 Detailed Description

Author

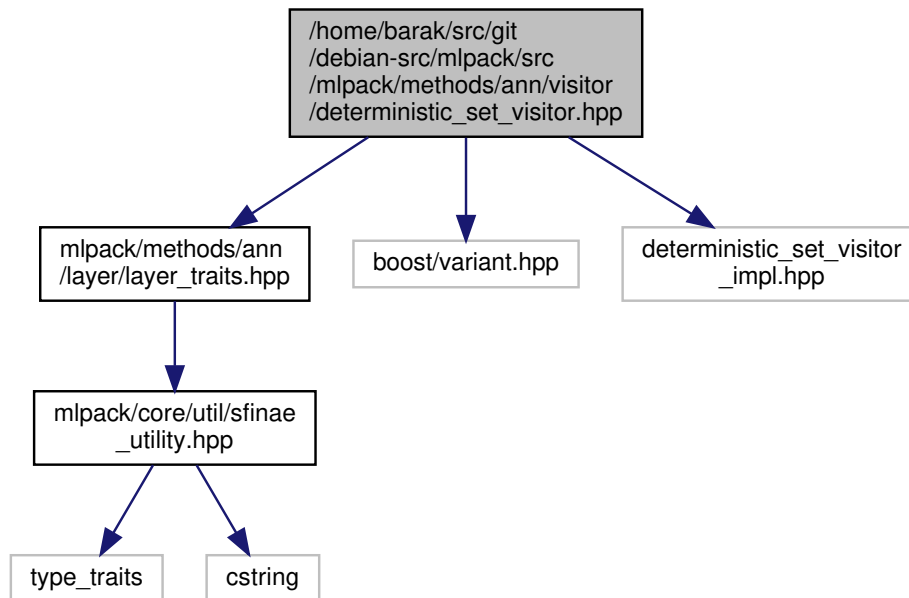
Marcus Edel

This file provides an abstraction for the Delta() function for different layers and automatically directs any parameter to the right layer type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.476 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/deterministic_set_visitor.hpp File Reference

Include dependency graph for deterministic_set_visitor.hpp:



Classes

- class **DeterministicSetVisitor**
DeterministicSetVisitor (p. 661) set the deterministic parameter given the deterministic value.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.476.1 Detailed Description

Author

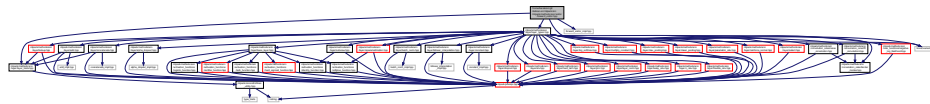
Marcus Edel

This file provides an abstraction for the Deterministic() function for different layers and automatically directs any parameter to the right layer type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.477 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/forward_visitor.hpp File Reference

Include dependency graph for forward_visitor.hpp:



Classes

- class **ForwardVisitor**

ForwardVisitor (p. 713) executes the Forward() function given the input and output parameter.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**

Artificial Neural Network.

40.477.1 Detailed Description

Author

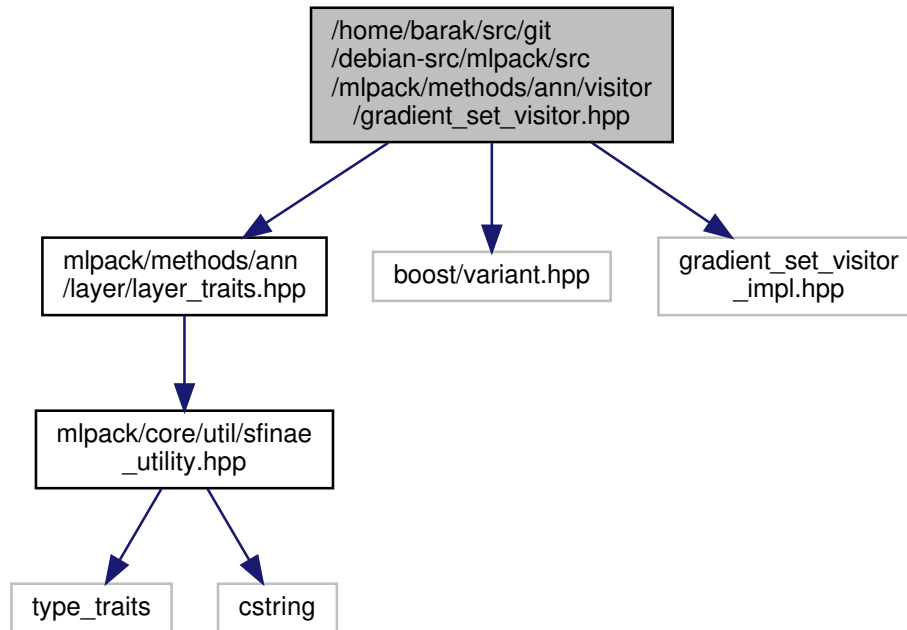
Marcus Edel

This file provides an abstraction for the Forward() function for different layers and automatically directs any parameter to the right layer type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.478 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/gradient_set_visitor.hpp File Reference

Include dependency graph for gradient_set_visitor.hpp:



Classes

- class **GradientSetVisitor**
GradientSetVisitor (p. 725) update the gradient parameter given the gradient set.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.478.1 Detailed Description

Author

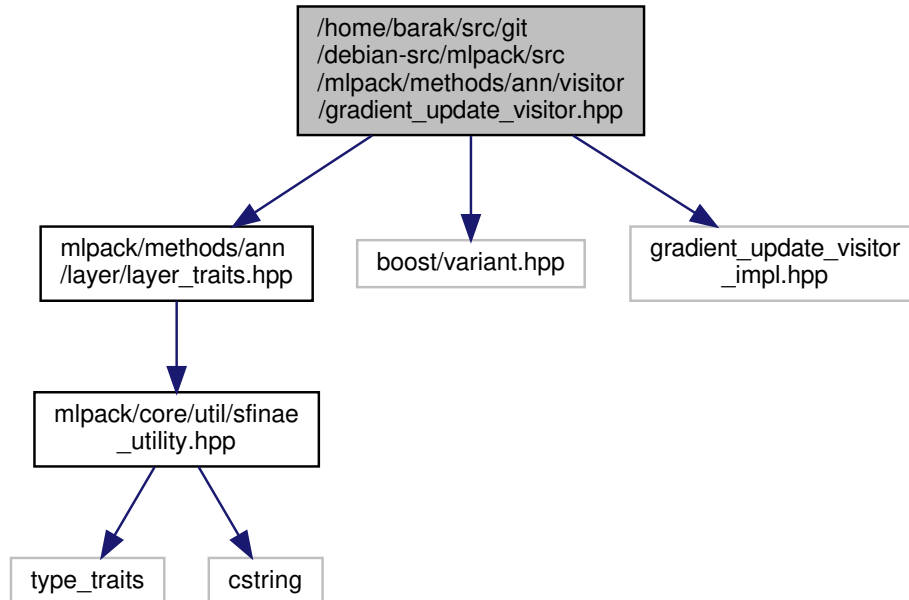
Marcus Edel

This file provides an abstraction for the `Gradient()` function for different layers and automatically directs any parameter to the right layer type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.479 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/gradient_↵ update_visitor.hpp File Reference

Include dependency graph for gradient_update_visitor.hpp:



Classes

- class **GradientUpdateVisitor**
GradientUpdateVisitor (p. 727) update the gradient parameter given the gradient set.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.479.1 Detailed Description

Author

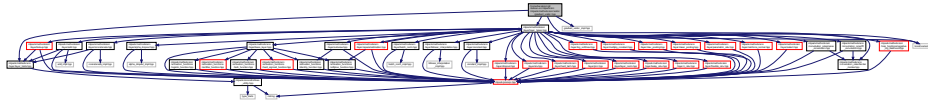
Marcus Edel

This file provides an abstraction for the `Gradient()` function for different layers and automatically directs any parameter to the right layer type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.480 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/gradient_visitor.hpp File Reference

Include dependency graph for gradient_visitor.hpp:



Classes

- class **GradientVisitor**

SearchModeVisitor executes the Gradient() method of the given module using the input and delta parameter.

Namespaces

- **mlpack**
 .hpp
- **mlpack::ann**

Artificial Neural Network.

40.480.1 Detailed Description

Author

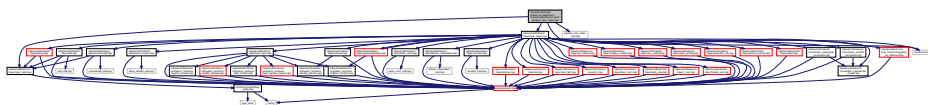
Marcus Edel

This file provides an abstraction for the Gradient() function for different layers and automatically directs any parameter to the right layer type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.481 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/gradient_zero_visitor.hpp File Reference

Include dependency graph for gradient_zero_visitor.hpp:



Classes

- class **GradientZeroVisitor**

Namespaces

- **mlpack**
 .hpp
- **mlpack::ann**
 Artificial Neural Network.

40.481.1 Detailed Description

Author

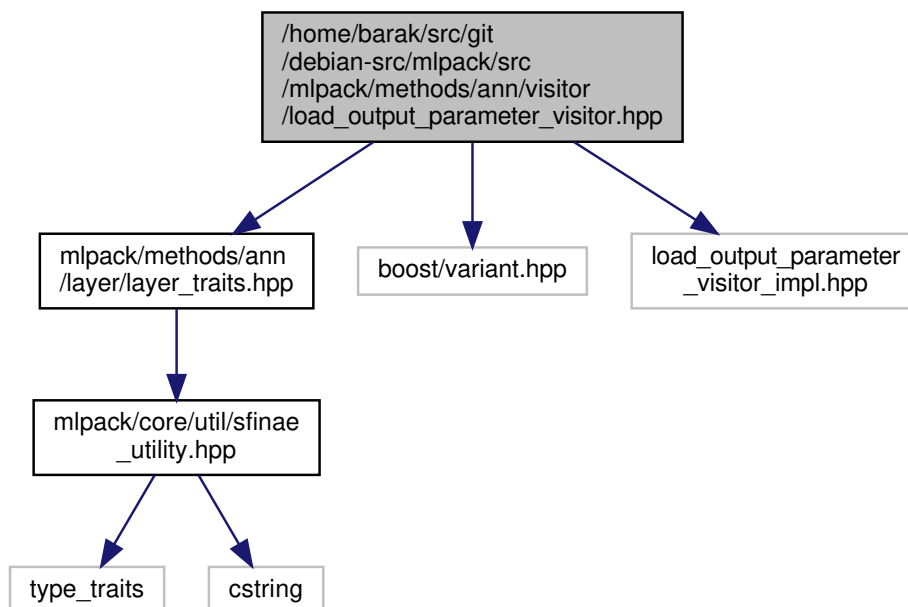
Marcus Edel

This file provides an abstraction for the Gradient() function for different layers and automatically directs any parameter to the right layer type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.482 `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/load_output_parameter_visitor.hpp` File Reference

Include dependency graph for `load_output_parameter_visitor.hpp`:



Classes

- class **LoadOutputParameterVisitor**
LoadOutputParameterVisitor (p. 788) restores the output parameter using the given parameter set.

Namespaces

- mlpack**
.hpp
- mlpack::ann**
Artificial Neural Network.

40.482.1 Detailed Description

Author

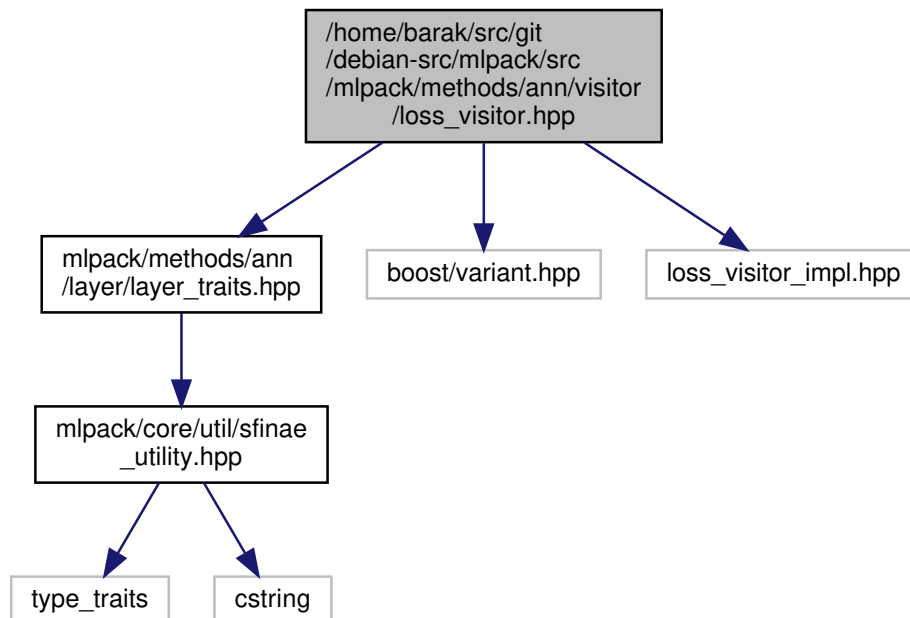
Marcus Edel

This file provides an abstraction for the OutputParameter() function for different layers and automatically directs any parameter to the right layer type.

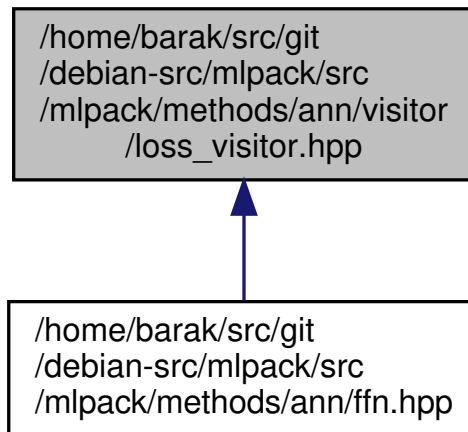
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.483 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/loss_visitor.hpp File Reference

Include dependency graph for loss_visitor.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **LossVisitor**

LossVisitor (p. 800) exposes the `Loss()` method of the given module.

Namespaces

- **mlpack**

.hpp

- **mlpack::ann**

Artificial Neural Network.

40.483.1 Detailed Description

Author

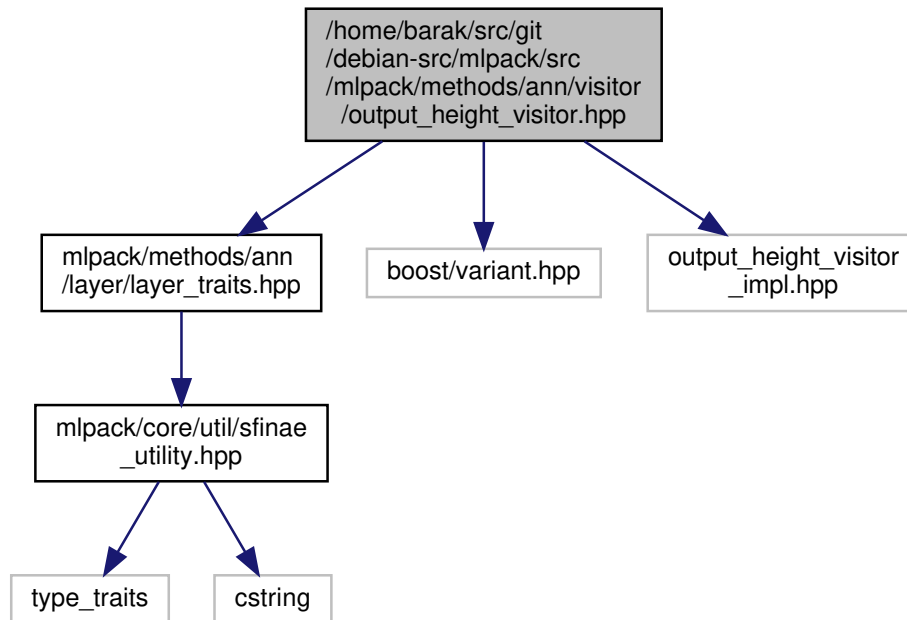
Atharva Khandait

This file provides an abstraction for the `Loss()` function for different layers and automatically directs any parameter to the right layer type.

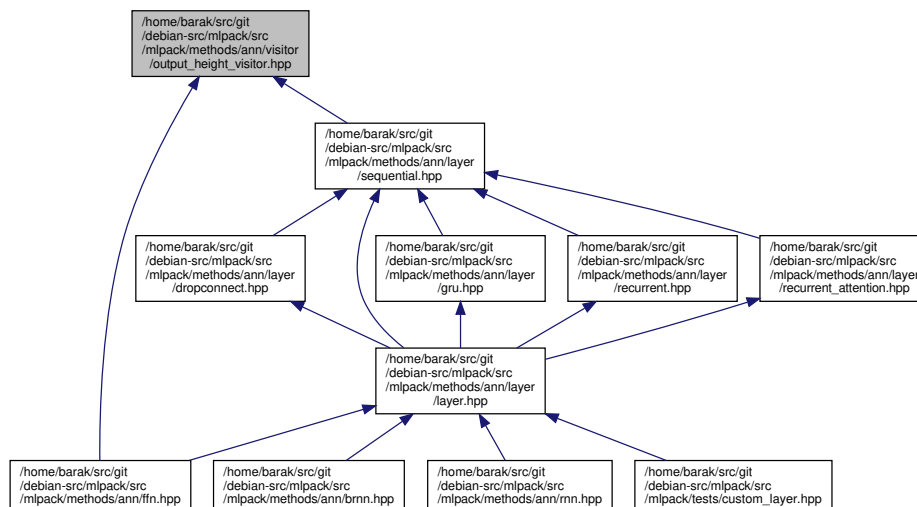
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.484 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/output_height_visitor.hpp File Reference

Include dependency graph for output_height_visitor.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **OutputHeightVisitor**

OutputHeightVisitor (p. 850) exposes the `OutputHeight()` method of the given module.

Namespaces

- **mlpack**
 .hpp
- **mlpack::ann**
 Artificial Neural Network.

40.484.1 Detailed Description

Author

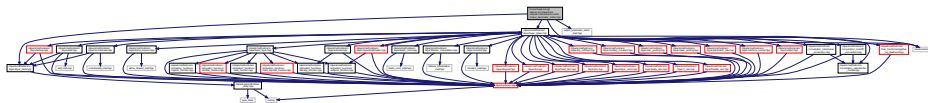
Marcus Edel

This file provides an abstraction for the `OutputHeight()` function for different layers and automatically directs any parameter to the right layer type.

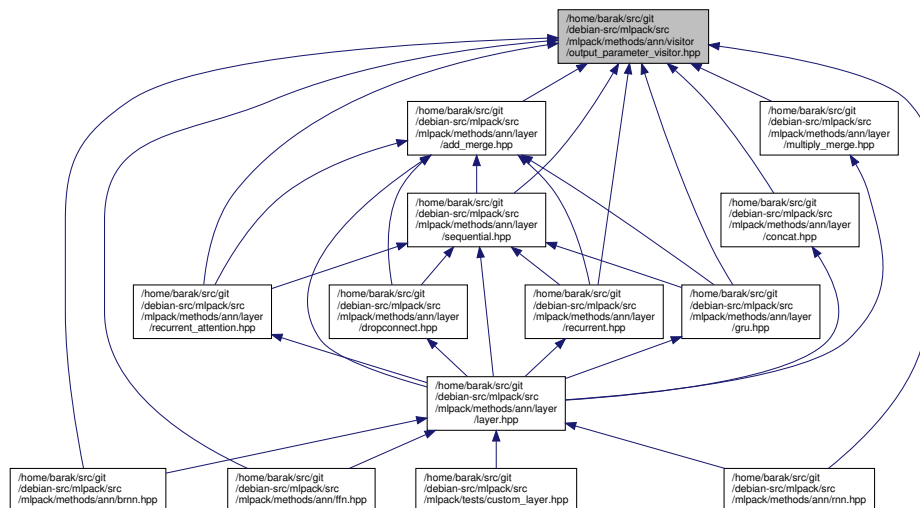
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.485 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/output_parameter_visitor.hpp File Reference

Include dependency graph for `output_parameter_visitor.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **OutputParameterVisitor**
OutputParameterVisitor (p. 851) exposes the output parameter of the given module.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.485.1 Detailed Description

Author

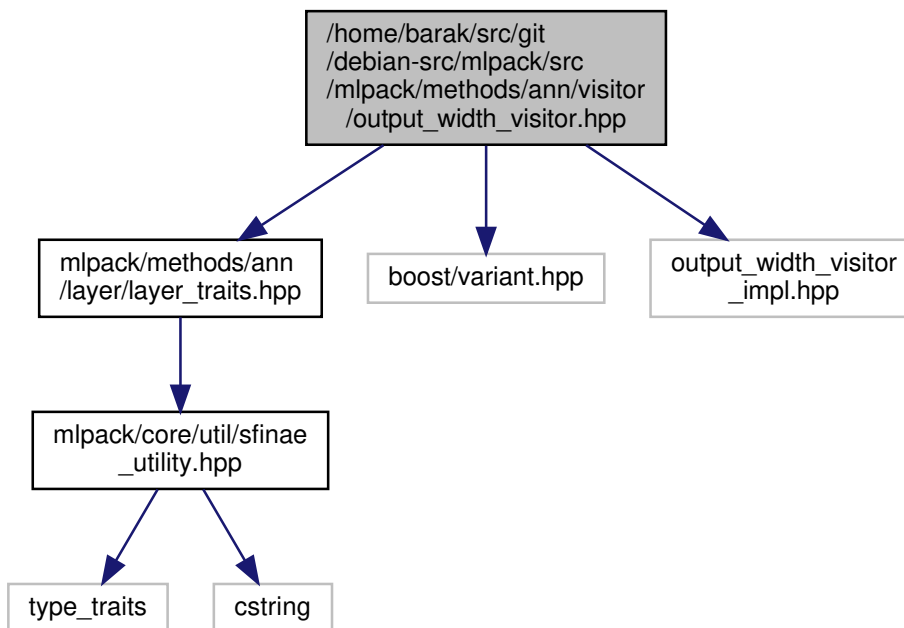
Marcus Edel

This file provides an abstraction for the OutputParameter() function for different layers and automatically directs any parameter to the right layer type.

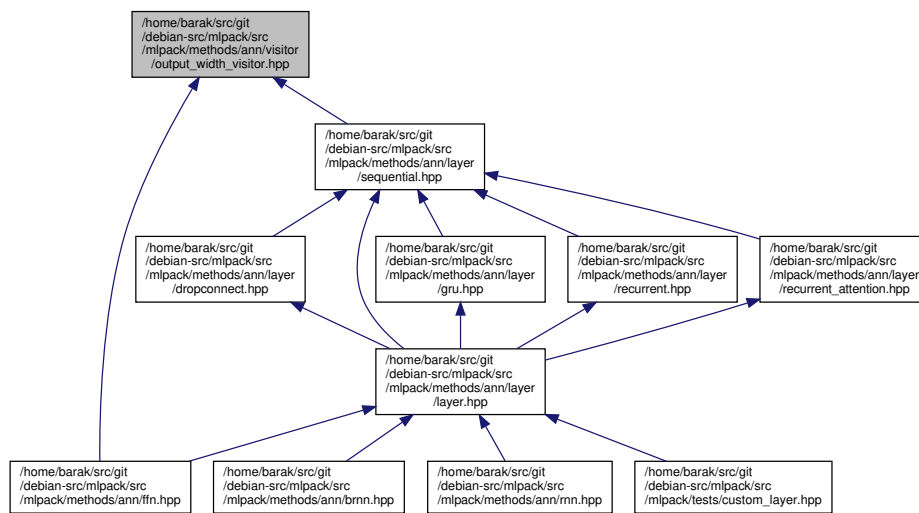
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.486 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/output_width_visitor.hpp File Reference

Include dependency graph for output_width_visitor.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **OutputWidthVisitor**

***OutputWidthVisitor** (p. 852) exposes the `OutputWidth()` method of the given module.*

Namespaces

- **mlpack**

.hpp

- **mlpack::ann**

Artificial Neural Network.

40.486.1 Detailed Description

Author

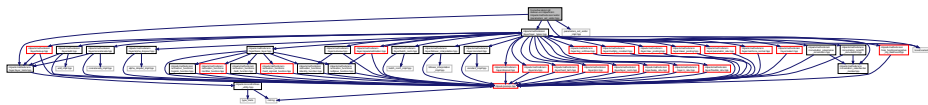
Marcus Edel

This file provides an abstraction for the `OutputWidth()` function for different layers and automatically directs any parameter to the right layer type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.487 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/parameters_set_visitor.hpp File Reference

Include dependency graph for parameters_set_visitor.hpp:



Classes

- class **ParametersSetVisitor**
ParametersSetVisitor (p. 853) update the parameters set using the given matrix.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.487.1 Detailed Description

Author

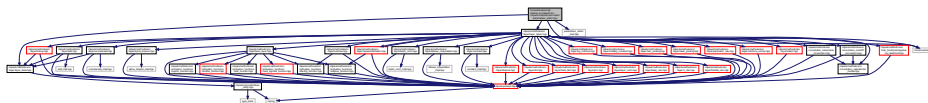
Marcus Edel

This file provides an abstraction for the Parameters() function for different layers and automatically directs any parameter to the right layer type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.488 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/parameters_visitor.hpp File Reference

Include dependency graph for parameters_visitor.hpp:



Classes

- class **ParametersVisitor**

ParametersVisitor (p. 854) exposes the parameters set of the given module and stores the parameters set into the given matrix.

Namespaces

- **mlpack**

.hpp

- **mlpack::ann**

Artificial Neural Network.

40.488.1 Detailed Description

Author

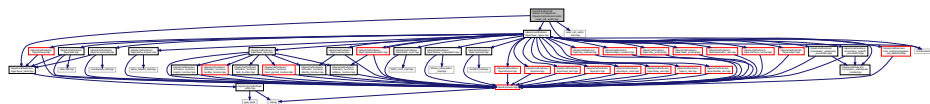
Marcus Edel

This file provides an abstraction for the Parameters() function for different layers and automatically directs any parameter to the right layer type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.489 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/reset_cell_visitor.hpp File Reference ↩

Include dependency graph for reset_cell_visitor.hpp:



Classes

- class **ResetCellVisitor**

ResetCellVisitor (p. 907) executes the ResetCell() function.

Namespaces

- **mlpack**
 .hpp
- **mlpack::ann**
 Artificial Neural Network.

40.489.1 Detailed Description

Author

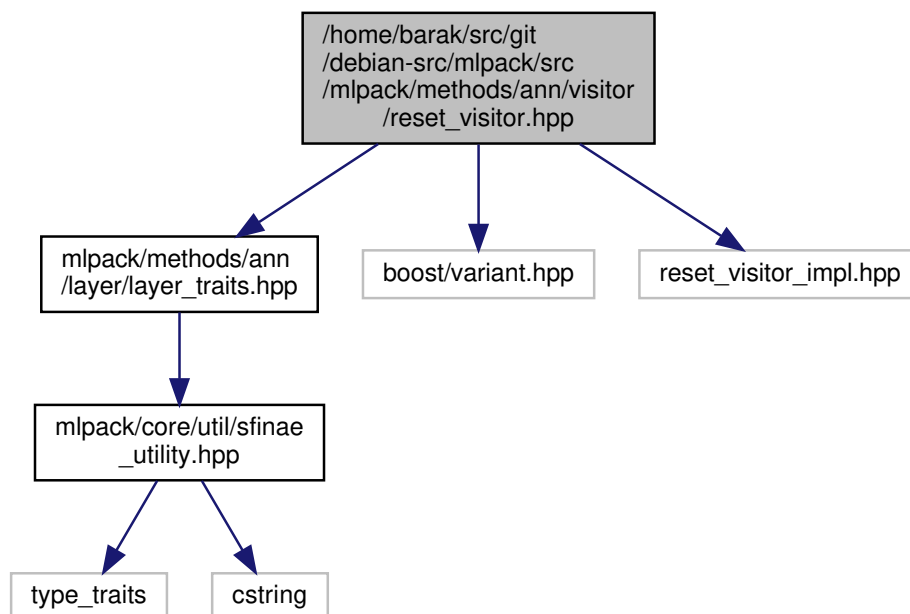
Sumedh Ghaisas

Boost static visitor abstraction for calling ResetCell function on RNN cells.

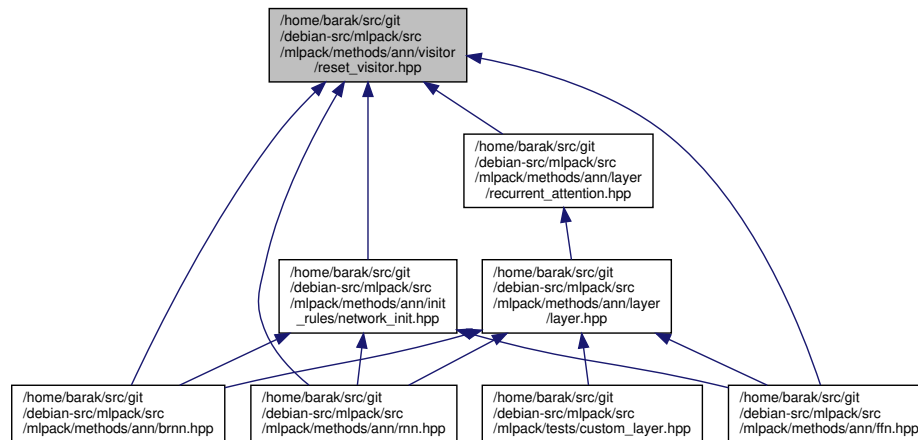
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.490 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/reset_visitor.hpp File Reference

Include dependency graph for reset_visitor.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **ResetVisitor**

***ResetVisitor** (p. 909) executes the `Reset()` function.*

Namespaces

- **mlpack**

.hpp

- **mlpack::ann**

Artificial Neural Network.

40.490.1 Detailed Description

Author

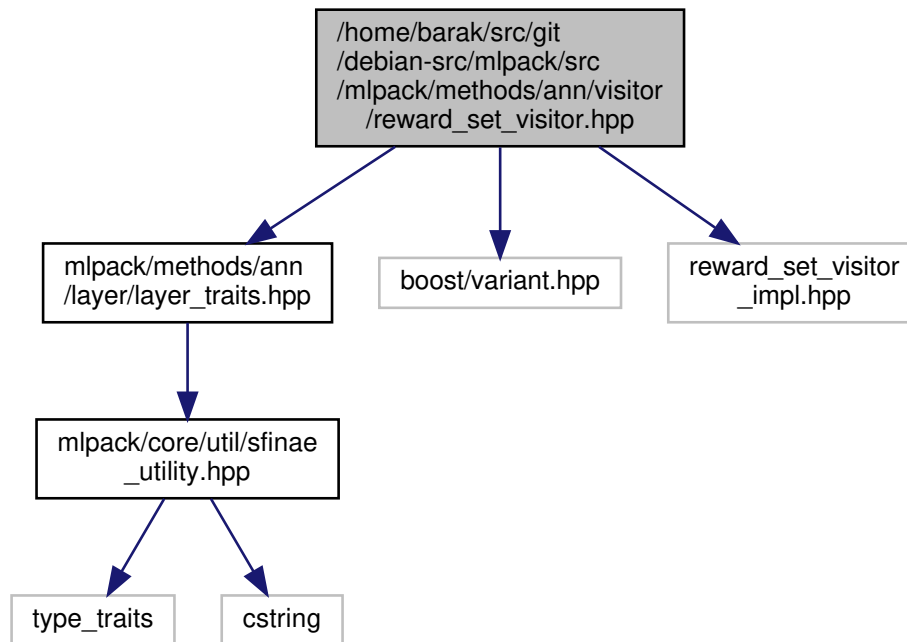
Marcus Edel

This file provides an abstraction for the `Reset()` function for different layers and automatically directs any parameter to the right layer type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.491 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/reward_set_visitor.hpp File Reference

Include dependency graph for reward_set_visitor.hpp:



Classes

- class **RewardSetVisitor**
RewardSetVisitor (p. 910) set the reward parameter given the reward value.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.491.1 Detailed Description

Author

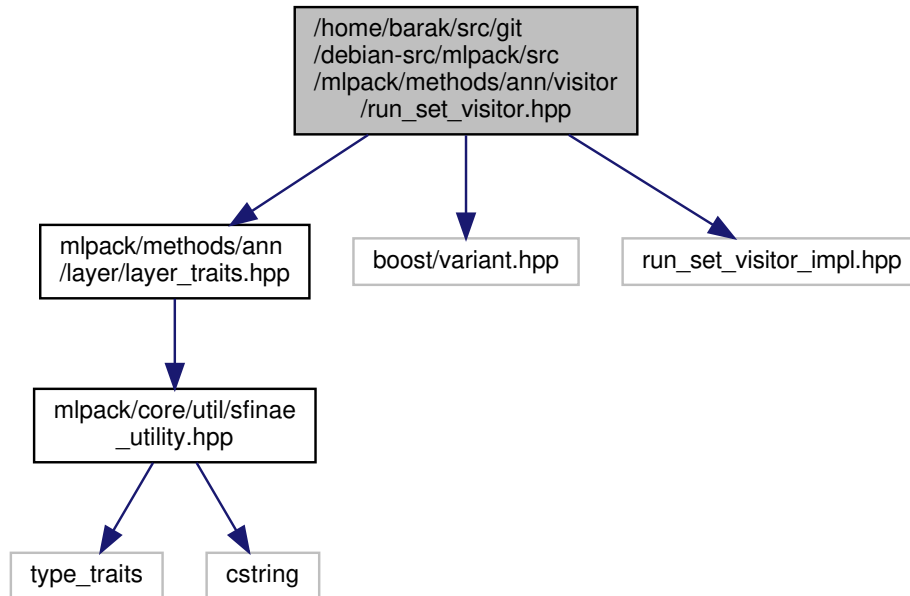
Marcus Edel

This file provides an abstraction for the `Reward()` function for different layers and automatically directs any parameter to the right layer type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.492 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/run_set_visitor.hpp File Reference

Include dependency graph for run_set_visitor.hpp:



Classes

- class **RunSetVisitor**
RunSetVisitor (p. 921) set the run parameter given the run value.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.492.1 Detailed Description

Author

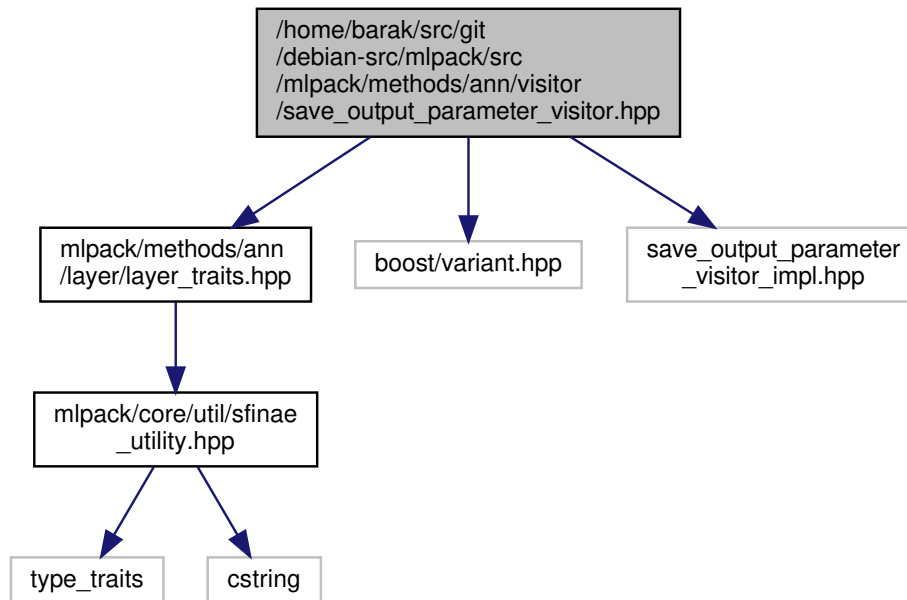
Saksham Bansal

This file provides an abstraction for the `Run()` function for different layers and automatically directs any parameter to the right layer type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.493 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/save_output_parameter_visitor.hpp File Reference

Include dependency graph for save_output_parameter_visitor.hpp:



Classes

- class **SaveOutputParameterVisitor**
SaveOutputParameterVisitor (p. 923) saves the output parameter into the given parameter set.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.493.1 Detailed Description

Author

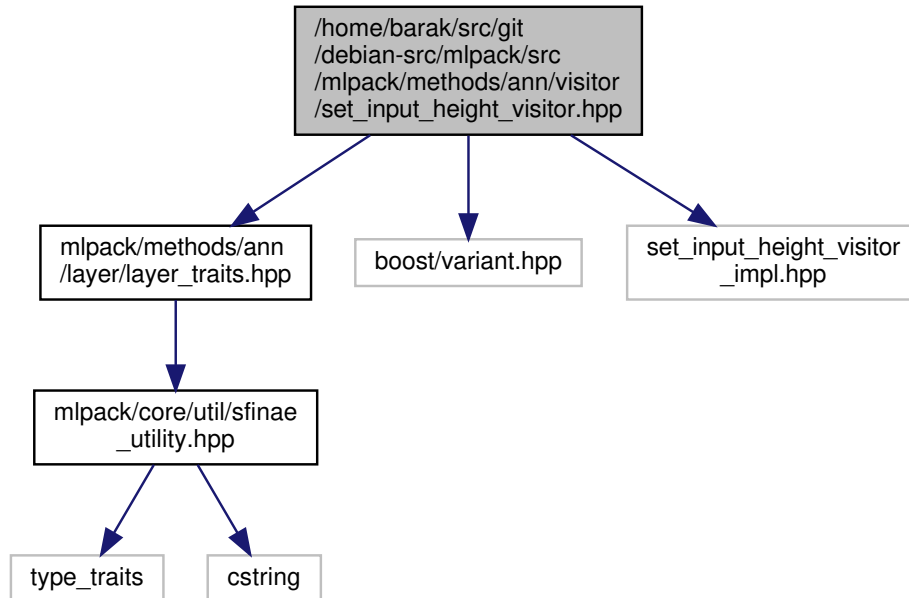
Marcus Edel

This file provides an abstraction for the `OutputParameter()` function for different layers and automatically directs any parameter to the right layer type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.494 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/set_input_height_visitor.hpp File Reference

Include dependency graph for set_input_height_visitor.hpp:



Classes

- class **SetInputHeightVisitor**
***SetInputHeightVisitor** (p. 934) updates the input height parameter with the given input height.*

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.494.1 Detailed Description

Author

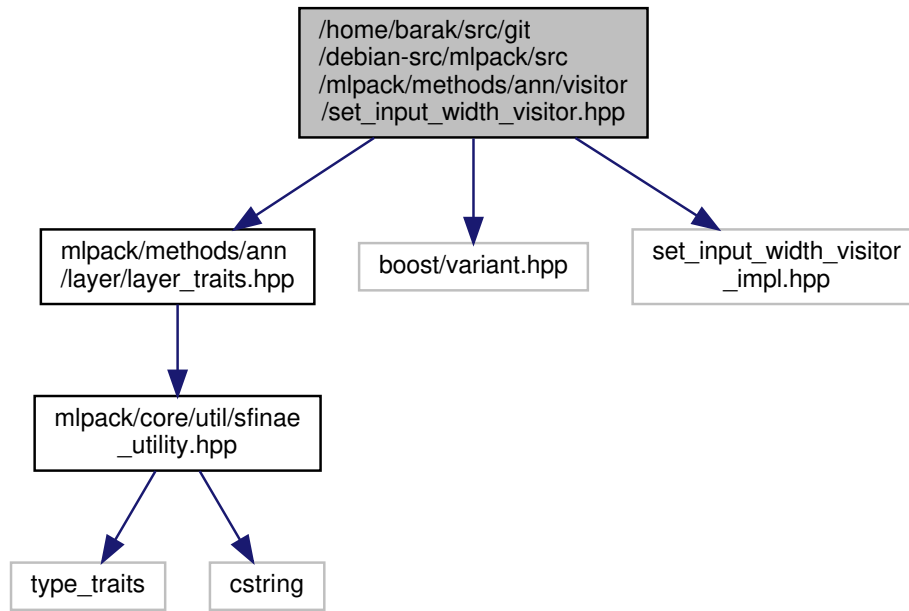
Marcus Edel

This file provides an abstraction for the `InputHeight()` function for different layers and automatically directs any parameter to the right layer type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.495 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/set_input_width_visitor.hpp File Reference

Include dependency graph for set_input_width_visitor.hpp:



Classes

- class **SetInputWidthVisitor**

SetInputWidthVisitor (p. 935) updates the input width parameter with the given input width.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.495.1 Detailed Description

Author

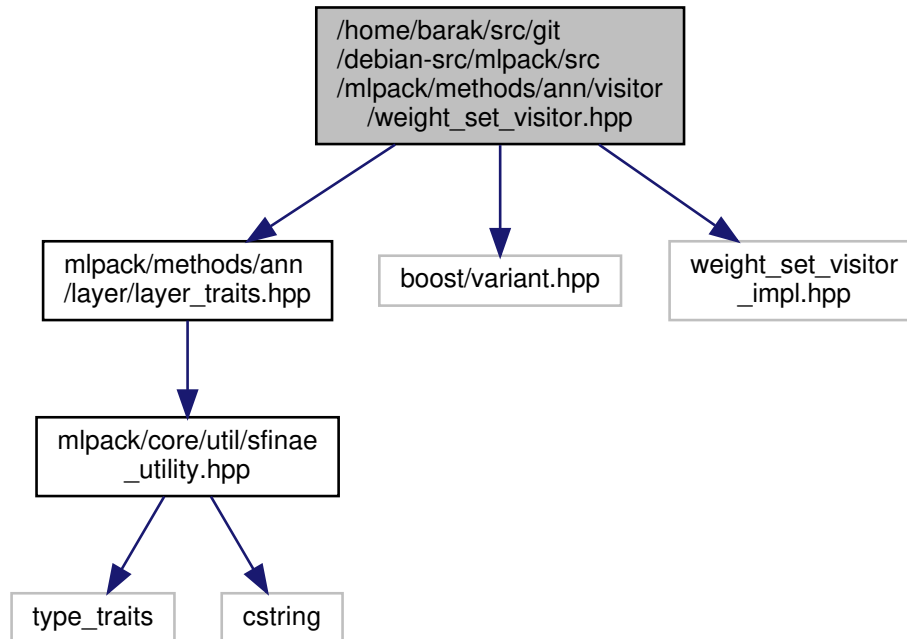
Marcus Edel

This file provides an abstraction for the `InputWidth()` function for different layers and automatically directs any parameter to the right layer type.

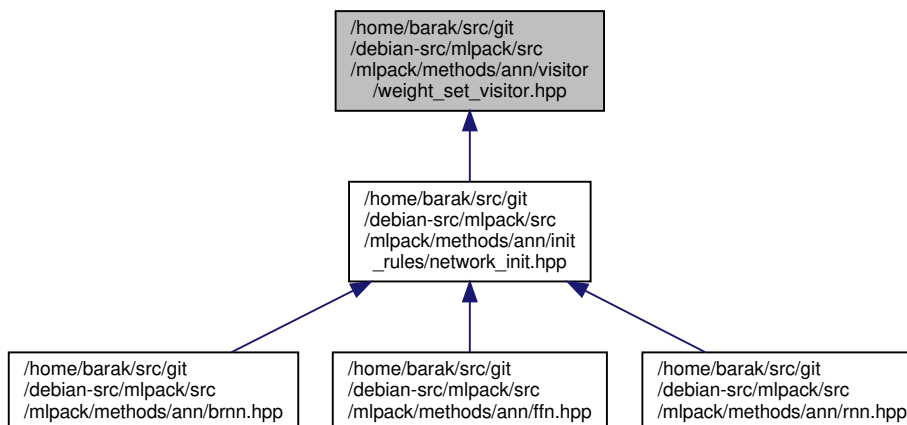
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.496 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/weight_set_visitor.hpp File Reference

Include dependency graph for weight_set_visitor.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **WeightSetVisitor**

WeightSetVisitor (p. 972) update the module parameters given the parameters set.

Namespaces

- **mlpack**
.hpp
- **mlpack::ann**
Artificial Neural Network.

40.496.1 Detailed Description

Author

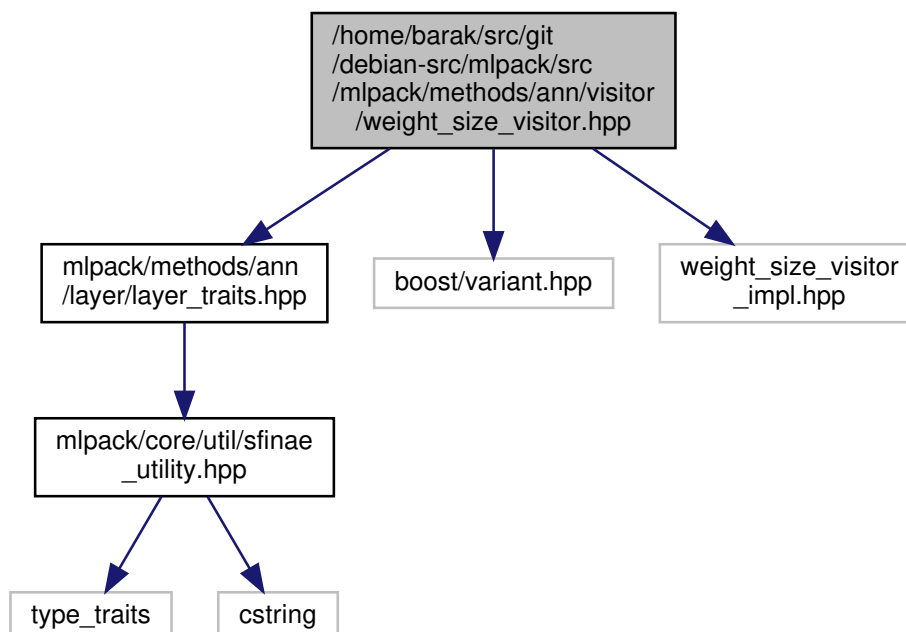
Marcus Edel

This file provides an abstraction for the `Weight()` function for different layers and automatically directs any parameter to the right layer type.

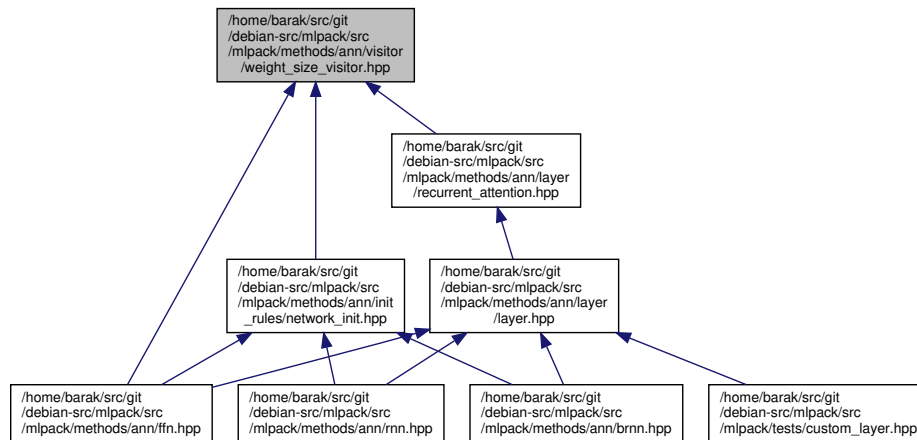
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.497 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/weight_size_visitor.hpp File Reference

Include dependency graph for `weight_size_visitor.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **WeightSizeVisitor**

***WeightSizeVisitor** (p. 973) returns the number of weights of the given module.*

Namespaces

- **mlpack**

.hpp

- **mlpack::ann**

Artificial Neural Network.

40.497.1 Detailed Description

Author

Marcus Edel

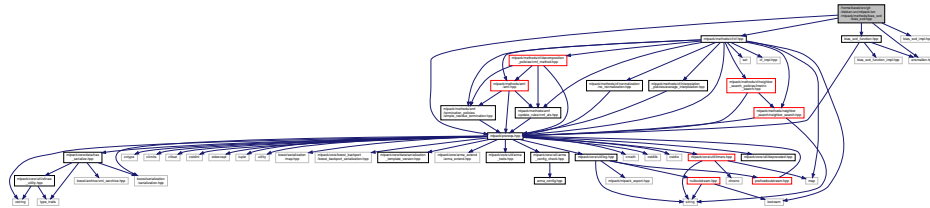
This file provides an abstraction for the `WeightSize()` function for different layers and automatically directs any parameter to the right layer type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

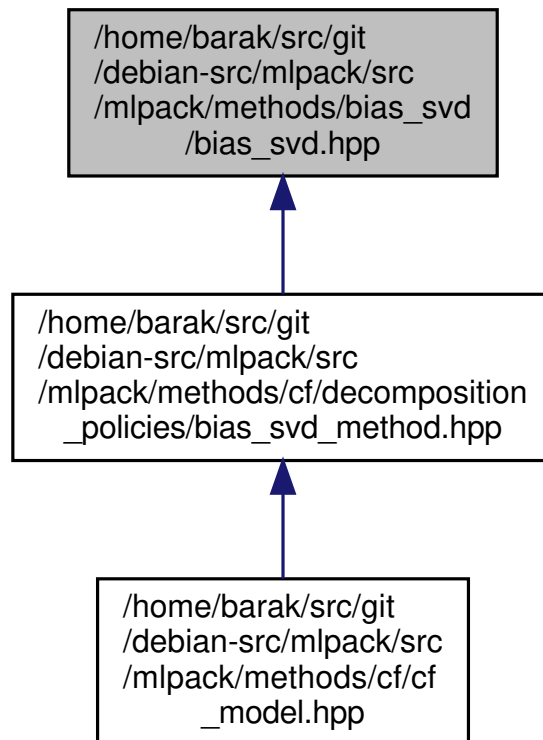
Author

40.500 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/bias_svd/bias_svd.hpp File Reference

Include dependency graph for bias_svd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **BiasSVD**< **OptimizerType** >

Bias SVD is an improvement on Regularized SVD which is a matrix factorization techniques.

Namespaces

- **mlpack**
 .hpp
- **mlpack::svd**

40.500.1 Detailed Description

Author

Siddharth Agrawal
Wenhao Huang

An implementation of Bias SVD.

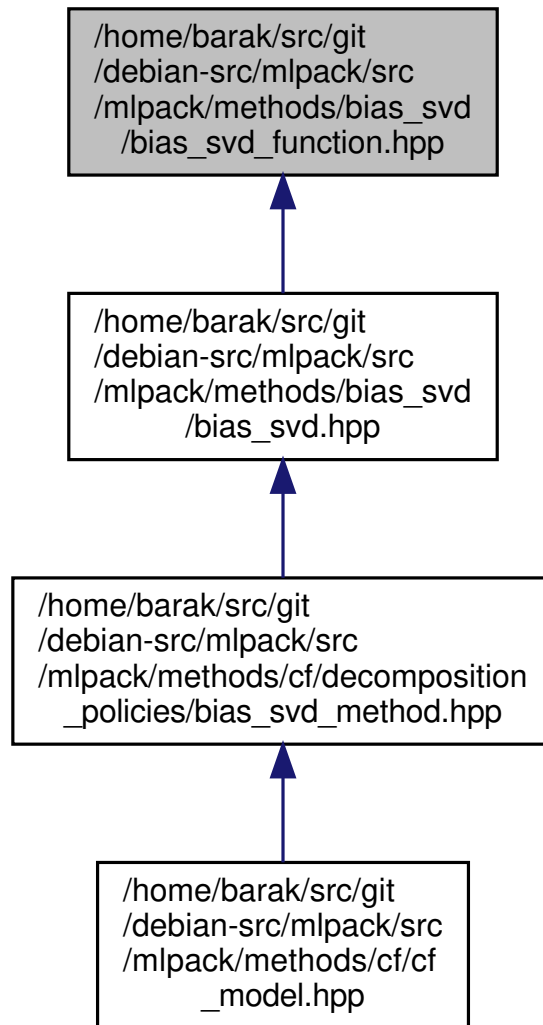
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpac. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.501 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/bias_svd/bias_svd_↵
function.hpp File Reference

Include dependency graph for bias_svd_function.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **BiasSVDFunction**< **MatType** >

*This class contains methods which are used to calculate the cost of **BiasSVD** (p. 1925)'s objective function, to calculate gradient of parameters with respect to the objective function, etc.*

Namespaces

- **ens**
- **mlpack**
 - **mlpack::svd**

Namespaces

- **mlpack**
 - .hpp
- **mlpack::svd**

40.502.1 Detailed Description

Author

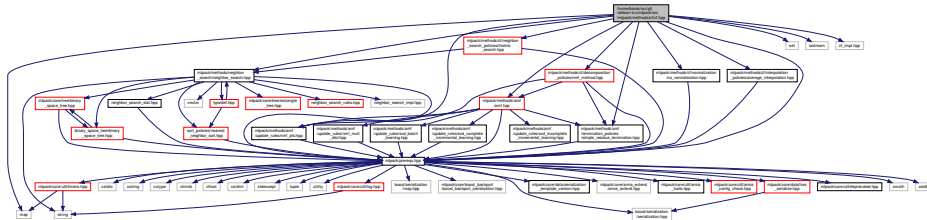
Marcus Edel

An implementation of the randomized block krylov SVD method.

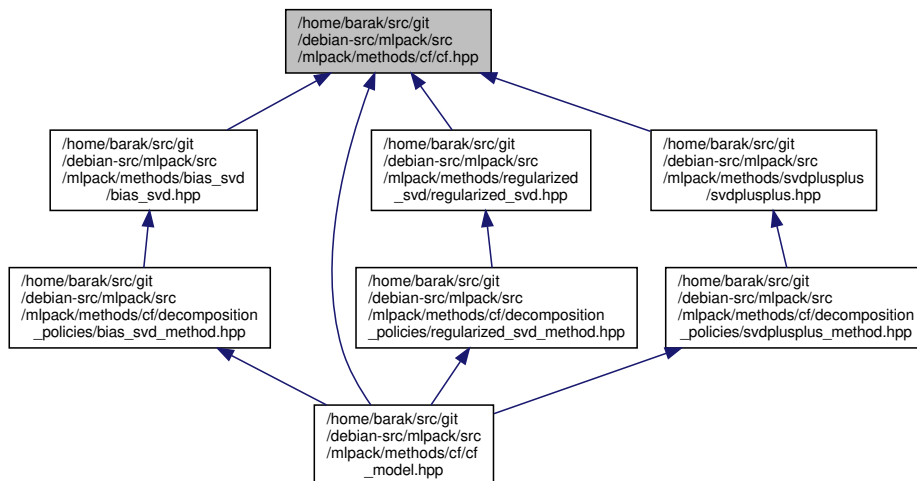
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.503 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/cf.hpp File Reference

Include dependency graph for cf.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **CFType**< **DecompositionPolicy**, **NormalizationType** >
This class implements Collaborative Filtering (CF).

Namespaces

- **mlpack**
.hpp
- **mlpack::cf**
Collaborative filtering.

40.503.1 Detailed Description

Author

Mudit Raj Gupta
Sumedh Ghaisas

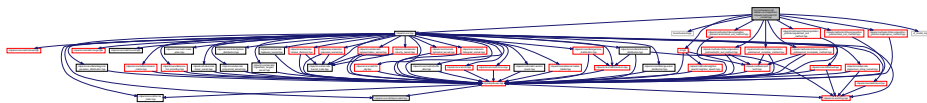
Collaborative filtering.

Defines the CFType class to perform collaborative filtering on the specified data set using alternating least squares (ALS).

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.504 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/cf_model.hpp File Reference

Include dependency graph for cf_model.hpp:



Classes

- class **CFModel**
The model to save to disk.
- class **DeleteVisitor**
DeleteVisitor (p. 1058) deletes the CFType<> object which is pointed to by the variable cf in class **CFModel** (p. 1042).
- class **GetValueVisitor**
GetValueVisitor (p. 1060) returns the pointer which points to the **CFType** (p. 1045) object.
- class **PredictVisitor**< **NeighborSearchPolicy**, **InterpolationPolicy** >
PredictVisitor (p. 1077) uses the **CFType** (p. 1045) object to make predictions on the given combinations of users and items.
- class **RecommendationVisitor**< **NeighborSearchPolicy**, **InterpolationPolicy** >
RecommendationVisitor (p. 1084) uses the **CFType** (p. 1045) object to get recommendations for the given users.

Namespaces

- **mlpack**
 .hpp
- **mlpack::cf**
 Collaborative filtering.

40.504.1 Detailed Description

Author

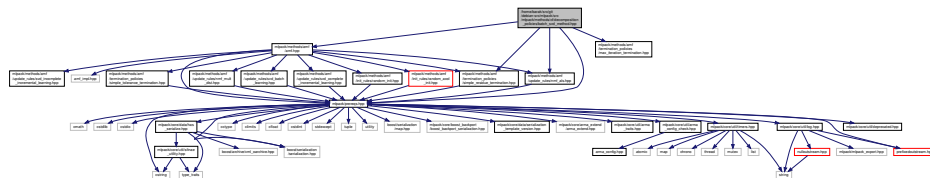
Wenhao Huang

A serializable CF model, used by the main program.

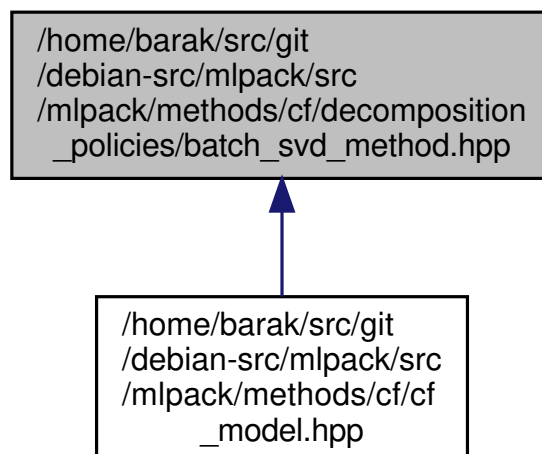
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.505 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/batch_svd_method.hpp File Reference

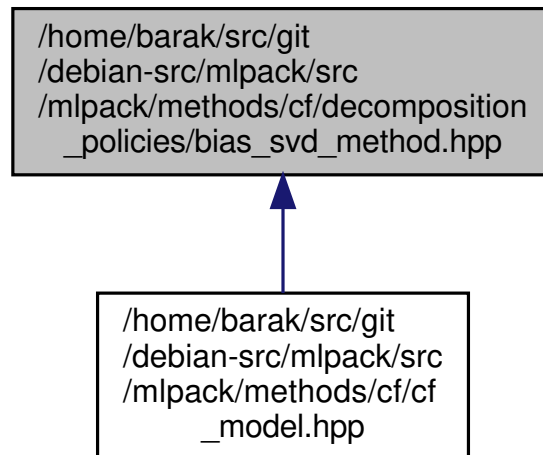
Include dependency graph for batch_svd_method.hpp:



This graph shows which files directly or indirectly include this file:



This graph shows which files directly or indirectly include this file:



Classes

- class **BiasSVDPolicy**

*Implementation of the Bias SVD policy to act as a wrapper when accessing Bias SVD from within **CFT** (p. 1045).*

Namespaces

- **mlpack**
.hpp
- **mlpack::cf**

Collaborative filtering.

40.506.1 Detailed Description

Author

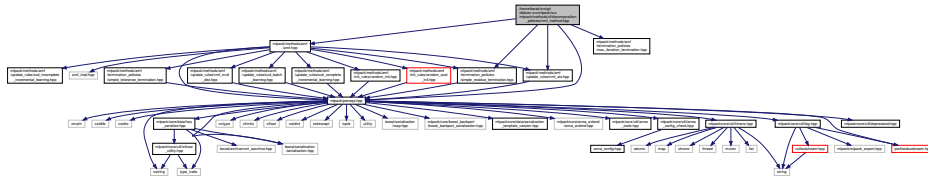
Wenhao Huang

Implementation of the bias svd method for use in the Collaborative Filtering.

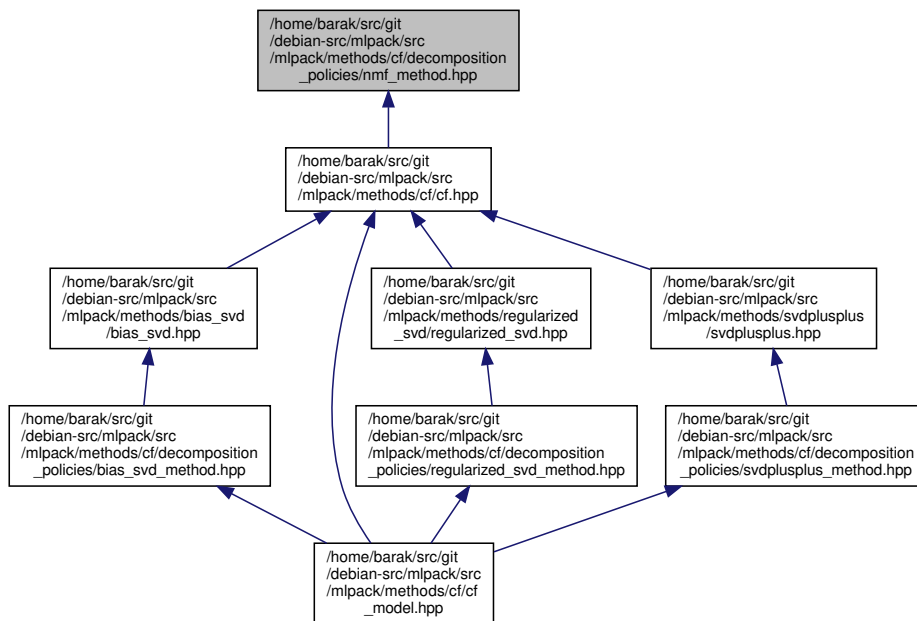
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.507 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/nmf_method.hpp File Reference

Include dependency graph for nmf_method.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **NMFPolicy**

Implementation of the NMF policy to act as a wrapper when accessing NMF from within **CFTType** (p. 1045).

Namespaces

- mlpack**
.hpp
- mlpack::cf**
Collaborative filtering.

40.507.1 Detailed Description

Author

Haritha Nair

Implementation of the exact svd method for use in Collaborative Filtering.

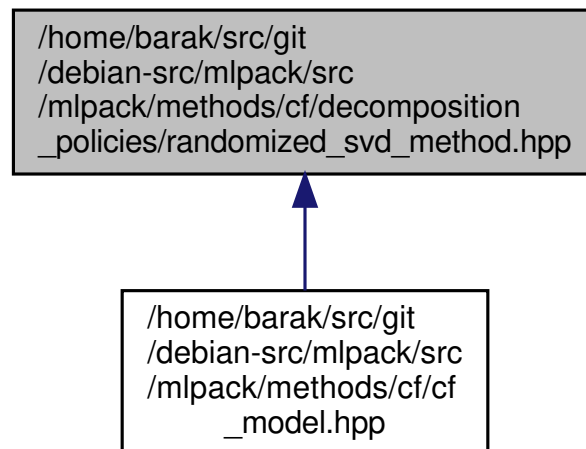
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.508 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/randomized_svd_method.hpp File Reference

Include dependency graph for randomized_svd_method.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RandomizedSVDPolicy**

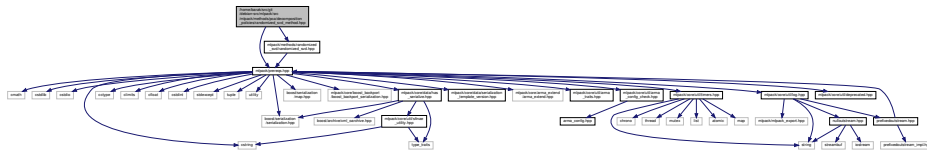
*Implementation of the Randomized SVD policy to act as a wrapper when accessing Randomized SVD from within **CFT**ype (p. 1045).*

Namespaces

- **mlpack**
 .hpp
- **mlpack::cf**
 Collaborative filtering.

40.509 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/decomposition_policies/randomized_svd_method.hpp File Reference

Include dependency graph for randomized_svd_method.hpp:



Classes

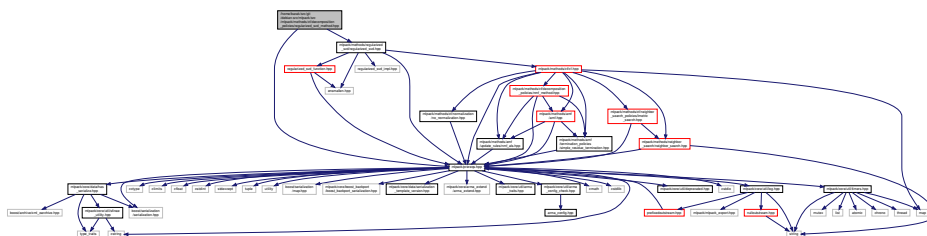
- class **RandomizedSVDPolicy**
 Implementation of the randomized SVD policy.

Namespaces

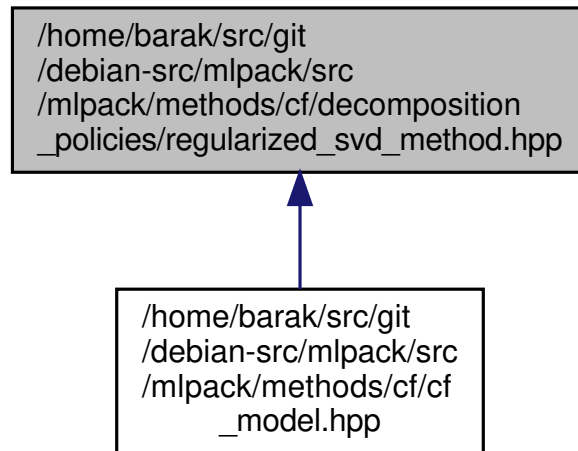
- **mlpack**
 .hpp
- **mlpack::pca**

40.510 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/regularized_svd_method.hpp File Reference

Include dependency graph for regularized_svd_method.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RegSVDPolicy**

*Implementation of the Regularized SVD policy to act as a wrapper when accessing Regularized SVD from within **CFTYPE** (p. 1045).*

Namespaces

- **mlpack**
.hpp
- **mlpack::cf**

Collaborative filtering.

40.510.1 Detailed Description

Author

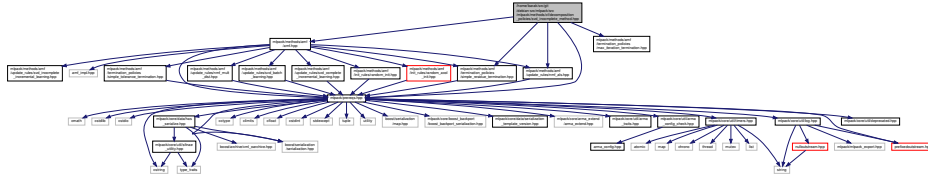
Haritha Nair

Implementation of the regularized svd method for use in the Collaborative Filtering.

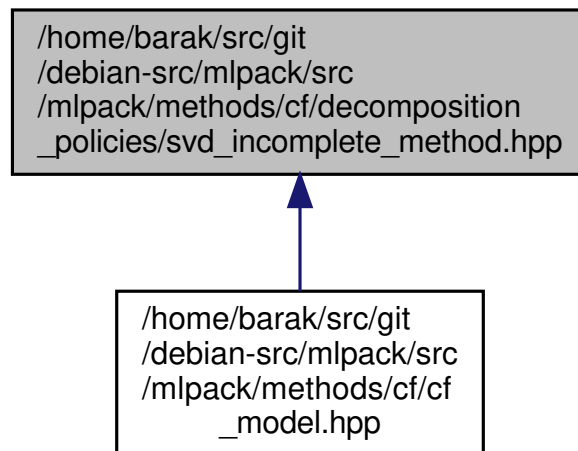
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information. ↩

40.512 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/svd_incomplete_method.hpp File Reference

Include dependency graph for svd_incomplete_method.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **SVDIncompletePolicy**

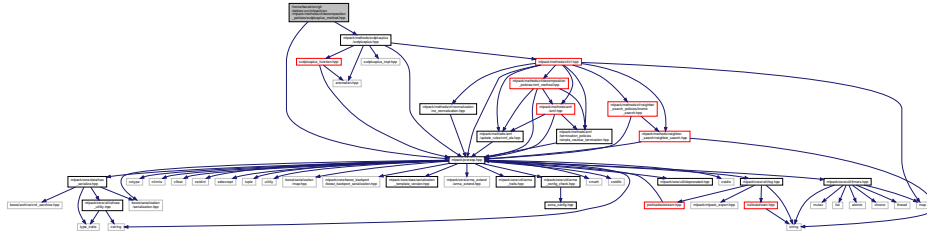
*Implementation of the SVD incomplete incremental to act as a wrapper when accessing SVD incomplete incremental from within **CFTYPE** (p. 1045).*

Namespaces

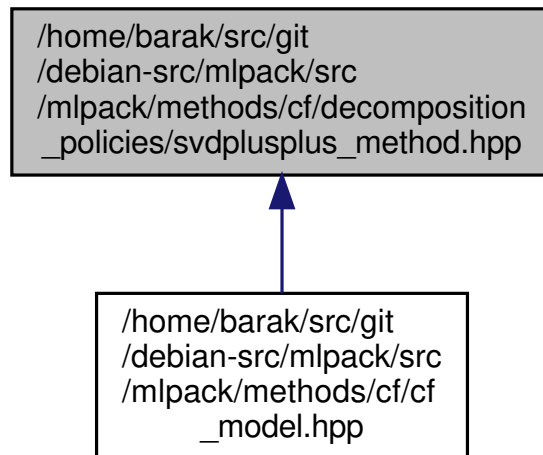
- **mlpack**
.hpp
- **mlpack::cf**
Collaborative filtering.

40.513 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition_policies/svdplusplus_method.hpp File Reference

Include dependency graph for svdplusplus_method.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **SVDPlusPlusPolicy**

*Implementation of the SVDPlusPlus policy to act as a wrapper when accessing SVDPlusPlus from within **CFTYPE** (p. 1045).*

Namespaces

- **mlpack**
.*hpp*
- **mlpack::cf**
Collaborative filtering.

40.513.1 Detailed Description

Author

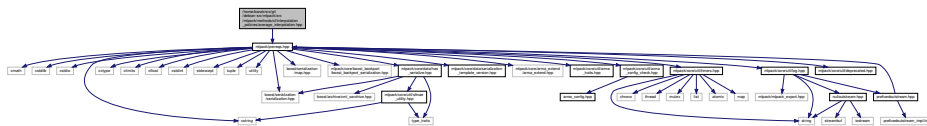
Wenhao Huang

Implementation of the svdplusplus method for use in the Collaborative Filtering.

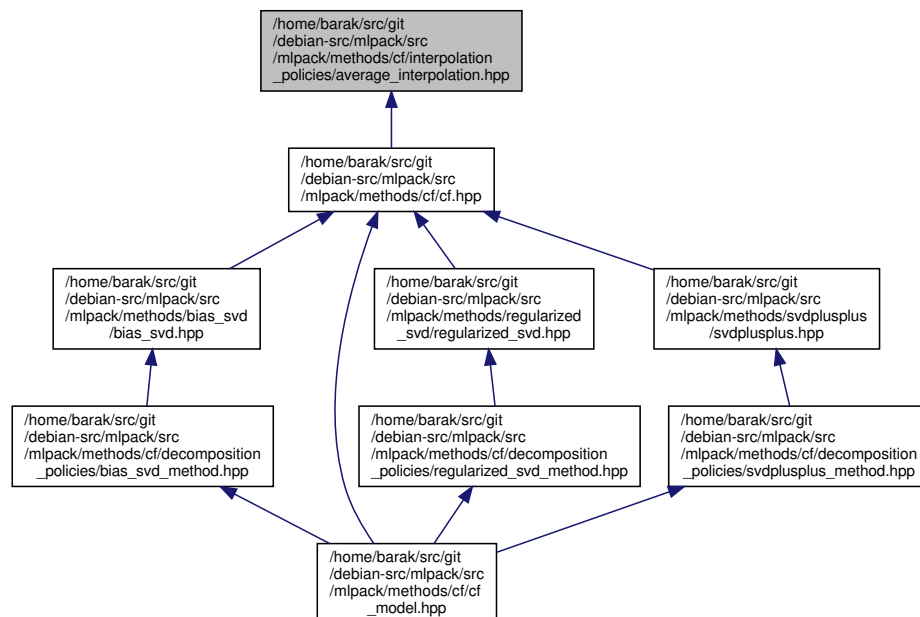
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.514 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/interpolation_policies/average_↔ _interpolation.hpp File Reference

Include dependency graph for average_interpolation.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **AverageInterpolation**

This class performs average interpolation to generate interpolation weights for neighborhood-based collaborative filtering.

Namespaces

- **mlpack**
 .hpp
- **mlpack::cf**
 Collaborative filtering.

40.514.1 Detailed Description

Author

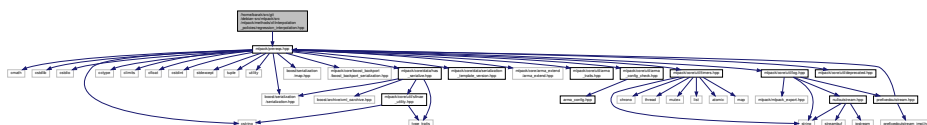
Wenhao Huang

Definition of AverageInterpolation class.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.515 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/interpolation_policies/regression_interpolation.hpp File Reference ↩↪

Include dependency graph for regression_interpolation.hpp:



Classes

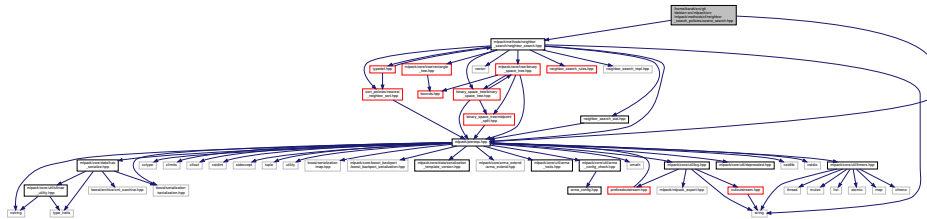
- class **RegressionInterpolation**
 Implementation of regression-based interpolation method.

Namespaces

- **mlpack**
 .hpp
- **mlpack::cf**
 Collaborative filtering.

40.517 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/neighbor_search_policies/cosine_search.hpp File Reference

Include dependency graph for cosine_search.hpp:



Classes

- class **CosineSearch**
Nearest neighbor search with cosine distance.

Namespaces

- **mlpack**
.hpp
- **mlpack::cf**
Collaborative filtering.

40.517.1 Detailed Description

Author

Wenhao Huang

Nearest neighbor search with cosine distance.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.519.1 Detailed Description

Author

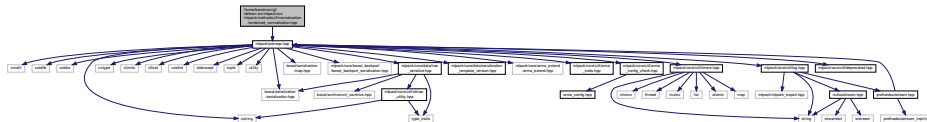
Wenhao Huang

Nearest neighbor search with pearson distance.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.520 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/combined_↵
_normalization.hpp File Reference

Include dependency graph for combined_normalization.hpp:



Classes

- class **CombinedNormalization**< **NormalizationTypes** >

This normalization class performs a sequence of normalization methods on raw ratings.

Namespaces

- **mlpack**
 .hpp
- **mlpack::cf**

Collaborative filtering.

40.520.1 Detailed Description

Author

Wenhao Huang

CombinedNormalization is a class template for performing a sequence of data normalization methods which are specified by template parameter.

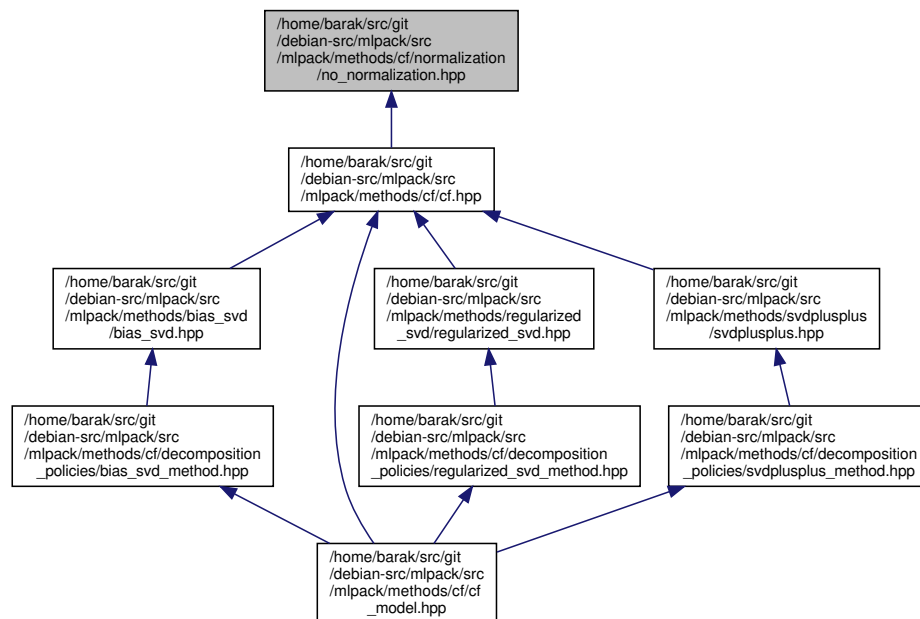
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.522 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/no_normalization.hpp File Reference

Include dependency graph for no_normalization.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **NoNormalization**
This normalization class doesn't perform any normalization.

Namespaces

- mlpack**
.hpp
- mlpack::cf**
Collaborative filtering.

- class **UserMeanNormalization**

- **mlpack**
 .hpp
- **mlpack::cf**
 Collaborative filtering.

Wenhao Huang

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.525 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/z_↵
score_normalization.hpp File Reference

Namespaces

- **mlpack**
 .hpp
- **mlpack::cf**
 Collaborative filtering.

Typedefs

- typedef SVDWrapper< DummyClass > **ArmaSVDFactorizer**
 add simple typedefs

40.526.1 Detailed Description

Author

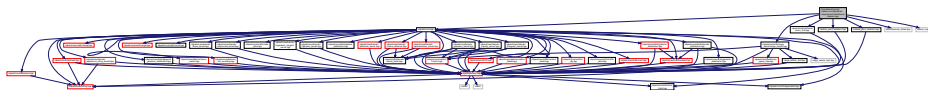
Sumedh Ghaisas

Wrapper class for SVD factorizers used for Collaborative Filtering.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.527 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/dbscan/dbscan.hpp File Reference

Include dependency graph for dbscan.hpp:



Classes

- class **DBSCAN**< RangeSearchType, PointSelectionPolicy >
 DBSCAN (p. 1197) (*Density-Based Spatial Clustering of Applications with Noise*) is a clustering technique described in the following paper:

Namespaces

- **mlpack**
 .hpp
- **mlpack::dbscan**

Namespaces

- **mlpack**
 .hpp
- **mlpack::dbscan**

40.528.1 Detailed Description

Author

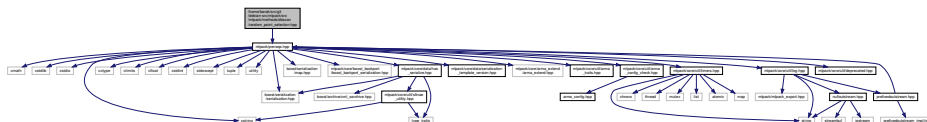
Kim SangYeon

Sequentially select the next point for DBSCAN.

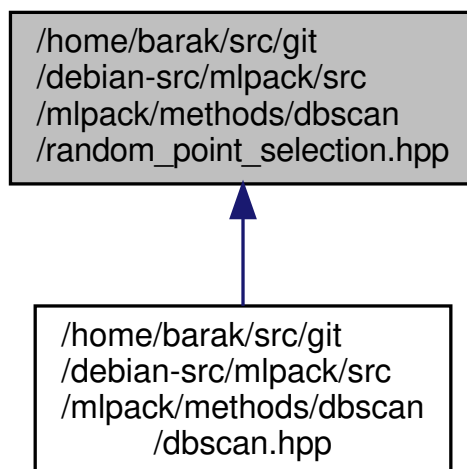
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.529 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/dbscan/random_point_selection.hpp File Reference

Include dependency graph for random_point_selection.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RandomPointSelection**

*This class can be used to randomly select the next point to use for **DBSCAN** (p. 1197).*

Namespaces

- **mlpack**
 .hpp
- **mlpack::dbscan**

40.529.1 Detailed Description

Author

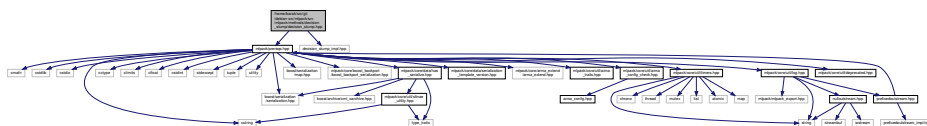
Ryan Curtin

Randomly select the next point for DBSCAN.

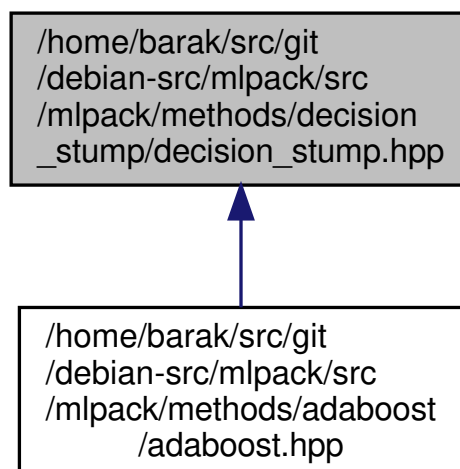
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.530 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_stump/decision_stump.hpp File Reference

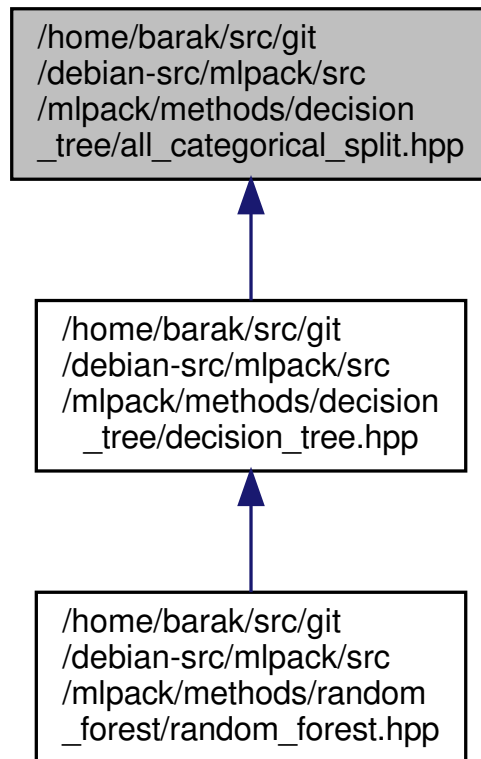
Include dependency graph for decision_stump.hpp:



This graph shows which files directly or indirectly include this file:



This graph shows which files directly or indirectly include this file:



Classes

- class **AllCategoricalSplit**< **FitnessFunction** >
*The **AllCategoricalSplit** (p. 1981) is a splitting function that will split categorical features into many children: one child for each category.*
- class **AllCategoricalSplit**< **FitnessFunction** >::**AuxiliarySplitInfo**< **ElemType** >

Namespaces

- **mlpack**
.hpp
- **mlpack::tree**
Trees and tree-building procedures.

40.531.1 Detailed Description

Author

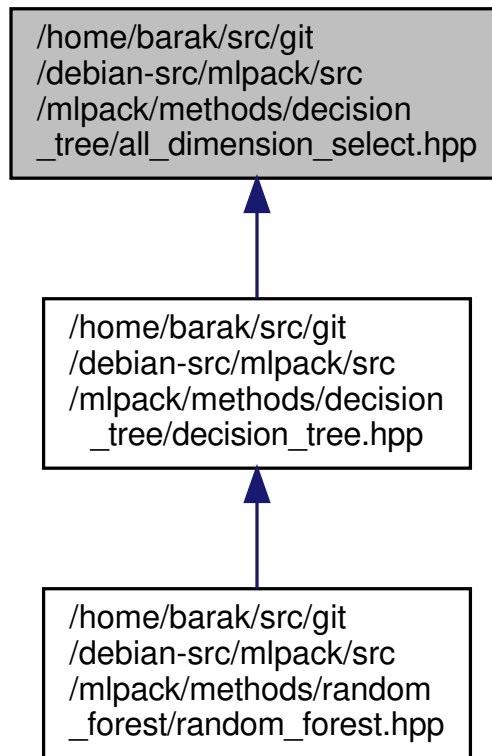
Ryan Curtin

This file defines a tree splitter that split a categorical feature into all of the possible categories.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.532 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/all_↵ dimension_select.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **AllDimensionSelect**

This dimension selection policy allows any dimension to be selected for splitting.

Namespaces

- **mlpack**
 .hpp
- **mlpack::tree**

Trees and tree-building procedures.

40.532.1 Detailed Description

Author

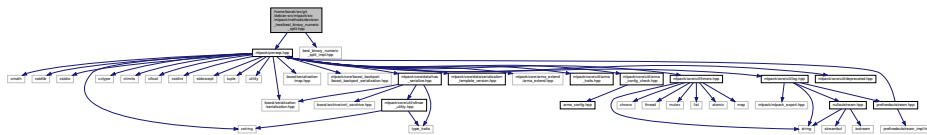
Ryan Curtin

Selects all dimensions for a split.

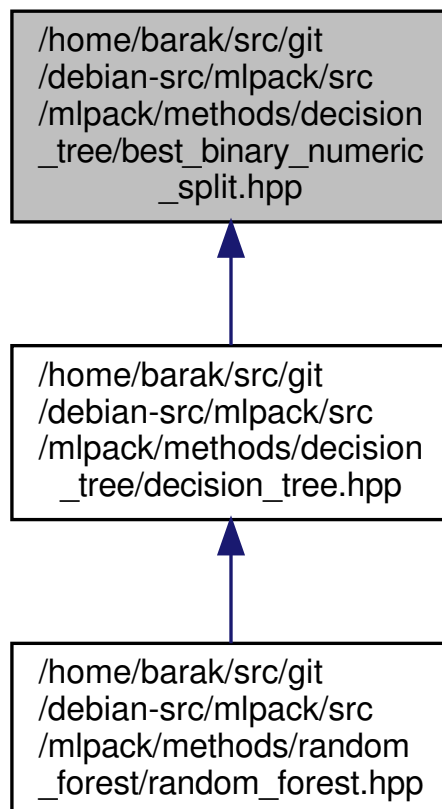
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.533 `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/best_binary_numeric_split.hpp` File Reference

Include dependency graph for `best_binary_numeric_split.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **BestBinaryNumericSplit**< **FitnessFunction** >

The **BestBinaryNumericSplit** (p. 1989) is a splitting function for decision trees that will exhaustively search a numeric dimension for the best binary split.

- class **BestBinaryNumericSplit**< **FitnessFunction** >::**AuxiliarySplitInfo**< **ElemType** >

Namespaces

- **mlpack**

.hpp

- **mlpack::tree**

Trees and tree-building procedures.

40.533.1 Detailed Description

Author

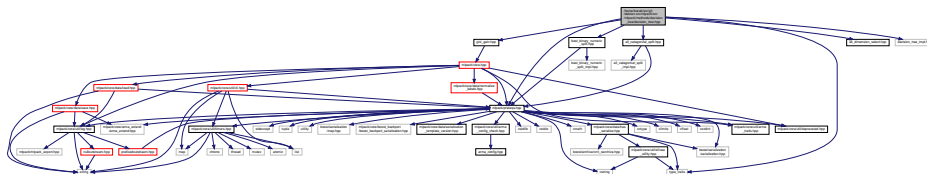
Ryan Curtin

A tree splitter that finds the best binary numeric split.

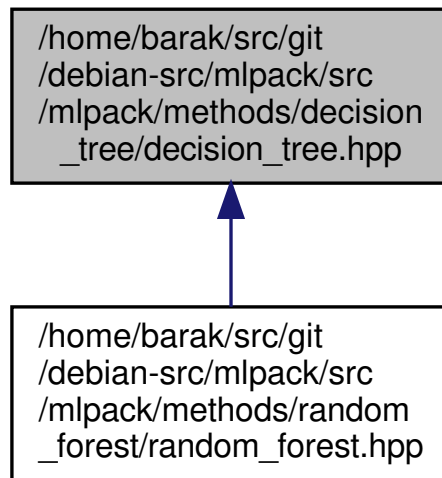
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.534 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/decision_tree.hpp File Reference

Include dependency graph for decision_tree.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **DecisionTree**< **FitnessFunction**, **NumericSplitType**, **CategoricalSplitType**, **DimensionSelectionType**, **ElemType**, **NoRecursion** >

This class implements a generic decision tree learner.

Namespaces

- **mlpack**
.hpp
- **mlpack::tree**

Trees and tree-building procedures.

Typedefs

- template<typename FitnessFunction = GiniGain, template< typename > class NumericSplitType = BestBinaryNumericSplit, template< typename > class CategoricalSplitType = AllCategoricalSplit, typename DimensionSelectType = AllDimensionSelect, typename ElemType = double>
using **DecisionStump** = DecisionTree< FitnessFunction, NumericSplitType, CategoricalSplitType, DimensionSelectType, ElemType, false >

Convenience typedef for decision stumps (single level decision trees).

40.534.1 Detailed Description

Author

Ryan Curtin

A generic decision tree learner. Its behavior can be controlled via template arguments.

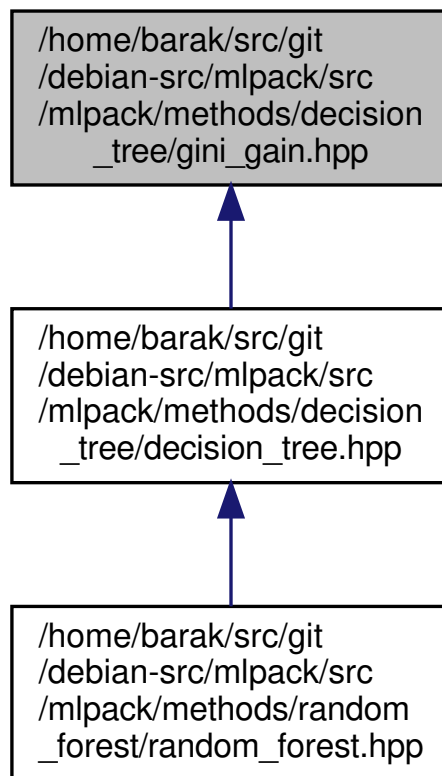
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.535 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/gini_gain.hpp File Reference

Include dependency graph for gini_gain.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **GiniGain**

The Gini gain, a measure of set purity usable as a fitness function (FitnessFunction) for decision trees.

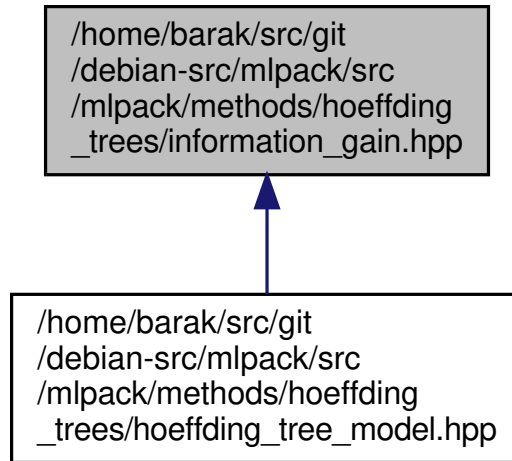
Namespaces

- **mlpack**
 .hpp
- **mlpack::tree**

Trees and tree-building procedures.

40.537 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/information_gain.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **InformationGain**

The standard information gain criterion, used for calculating gain in decision trees.

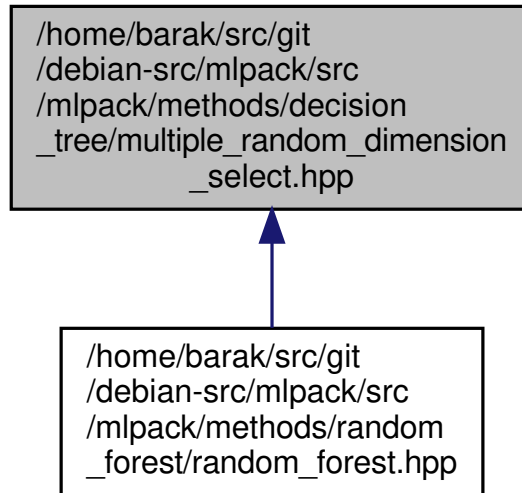
Namespaces

- **mlpack**
.hpp
- **mlpack::tree**

Trees and tree-building procedures.

40.538 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/decision_tree/multiple_random_dimension_select.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **MultipleRandomDimensionSelect**

This dimension selection policy allows the selection from a few random dimensions.

Namespaces

- **mlpack**
 .hpp
- **mlpack::tree**

Trees and tree-building procedures.

40.538.1 Detailed Description

Author

Ryan Curtin

Select a number of random dimensions to pick from.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

- class **RandomDimensionSelect**

- **mlpack**

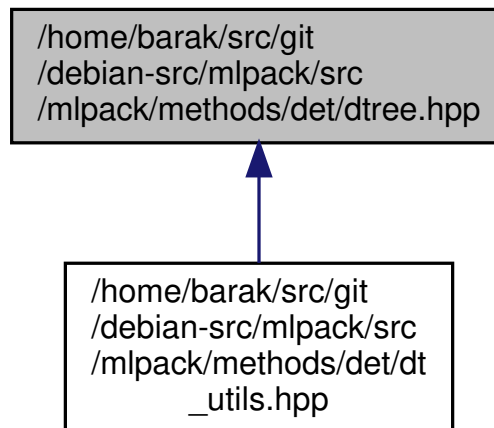
- `mlpack::tree`

40.540 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/det/dt_utils.hpp File Reference

- class **PathCacher**

Generated by Doxygen

This graph shows which files directly or indirectly include this file:



Classes

- class **DTree**< **MatType**, **TagType** >

A density estimation tree is similar to both a decision tree and a space partitioning tree (like a kd-tree).

Namespaces

- **mlpack**
 .hpp
- **mlpack::det**

Density Estimation Trees.

40.541.1 Detailed Description

Author

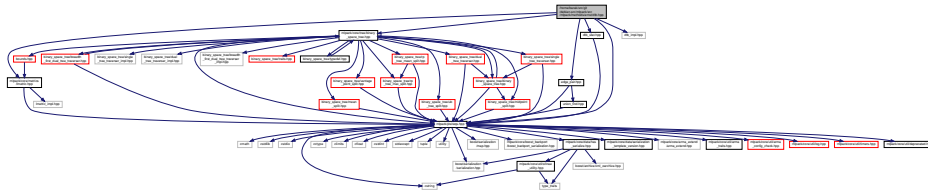
Parikshit Ram (pram@cc.gatech.edu)

Density Estimation Tree class

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.542 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/dtb.hpp File Reference

Include dependency graph for dtb.hpp:



Classes

- class **DualTreeBoruvka**< **MetricType**, **MatType**, **TreeType** >
Performs the MST calculation using the Dual-Tree Boruvka algorithm, using any type of tree.

Namespaces

- **mlpack**
.hpp
- **mlpack::emst**
Euclidean Minimum Spanning Trees.

40.542.1 Detailed Description

Author

Bill March (march@gatech.edu)

Contains an implementation of the DualTreeBoruvka algorithm for finding a Euclidean Minimum Spanning Tree using the kd-tree data structure.

```
@inproceedings{
  author = {March, W.B., Ram, P., and Gray, A.G.},
  title = {{Fast Euclidean Minimum Spanning Tree: Algorithm, Analysis,
    Applications.}},
  booktitle = {Proceedings of the 16th ACM SIGKDD International Conference
    on Knowledge Discovery and Data Mining}
  series = {KDD 2010},
  year = {2010}
}
```

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Author

Bill March (march@gatech.edu)

Tree traverser rules for the DualTreeBoruvka algorithm.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpac. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Author

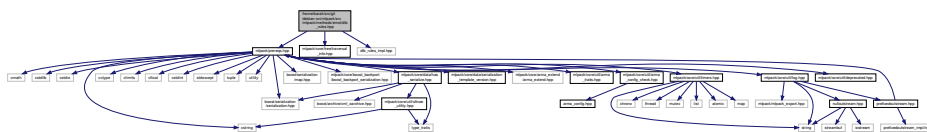
Bill March (march@gatech.edu)

DTBStat is the `StatisticType` used by trees when performing EMST.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.543 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/dtb_rules.hpp File Reference

Include dependency graph for dtb_rules.hpp:



Classes

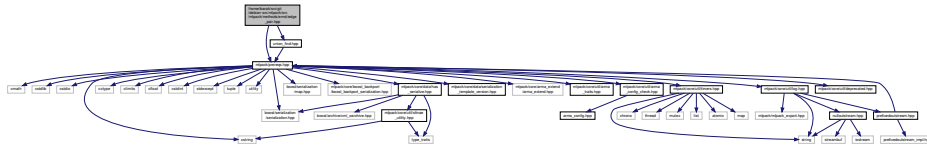
- class **DTBRules**< **MetricType**, **TreeType** >

Namespaces

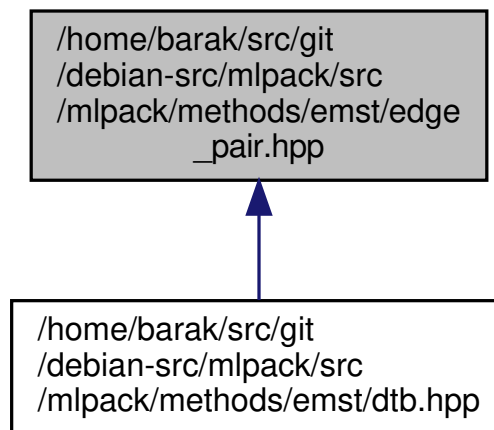
- **mlpack**
.hpp
- **mlpack::emst**
Euclidean Minimum Spanning Trees.

40.545 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/edge_pair.hpp File Reference

Include dependency graph for edge_pair.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **EdgePair**
An edge pair is simply two indices and a distance.

Namespaces

- **mlpack**
.hpp
- **mlpack::emst**
Euclidean Minimum Spanning Trees.

40.545.1 Detailed Description

Author

Bill March (march@gatech.edu)

This file contains utilities necessary for all of the minimum spanning tree algorithms.

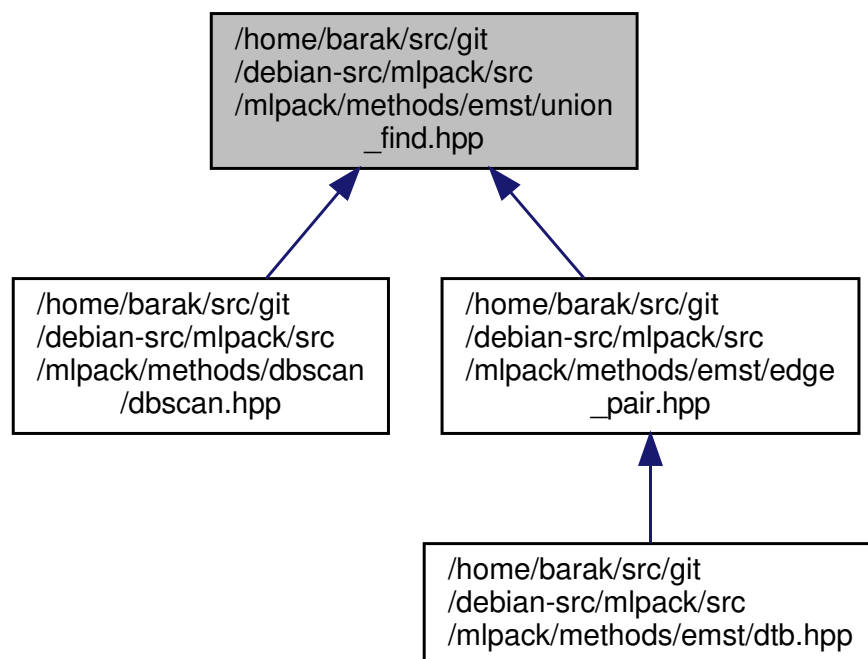
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.546 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/emst/union_find.hpp File Reference

Include dependency graph for union_find.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **UnionFind**
A Union-Find data structure.

Namespaces

- **mlpack**
.hpp
- **mlpack::emst**
Euclidean Minimum Spanning Trees.

40.546.1 Detailed Description

Author

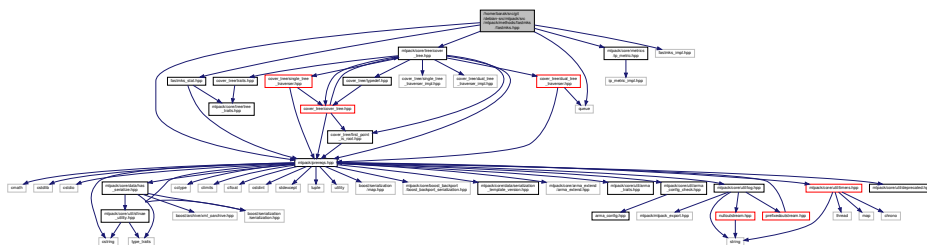
Bill March (march@gatech.edu)

Implements a union-find data structure. This structure tracks the components of a graph. Each point in the graph is initially in its own component. Calling `unionfind.Union(x, y)` unites the components indexed by `x` and `y`. `unionfind.Find(x)` returns the index of the component containing point `x`.

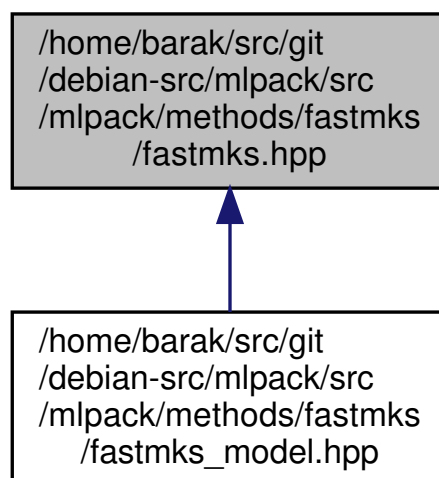
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.547 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/fastmks/fastmks.hpp File Reference

Include dependency graph for `fastmks.hpp`:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::fastmks**
 Fast max-kernel search.

40.548.1 Detailed Description

Author

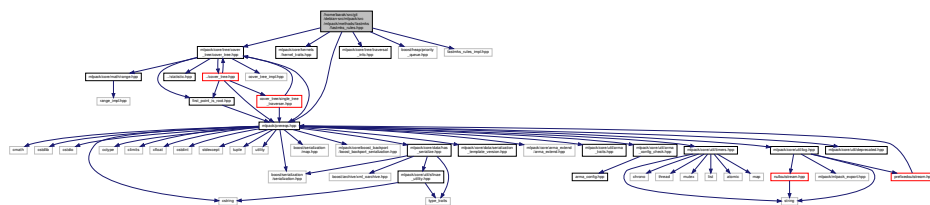
Ryan Curtin

A utility struct to contain all the possible FastMKS models.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.549 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/fastmks/fastmks_↵
rules.hpp File Reference

Include dependency graph for fastmks_rules.hpp:



Classes

- class **FastMKSRules**< **KernelType**, **TreeType** >
*The **FastMKSRules** (p. 1298) class is a template helper class used by **FastMKS** (p. 1280) class when performing exact max-kernel search.*

Namespaces

- **mlpack**
 .hpp
- **mlpack::fastmks**
 Fast max-kernel search.

40.549.1 Detailed Description

Author

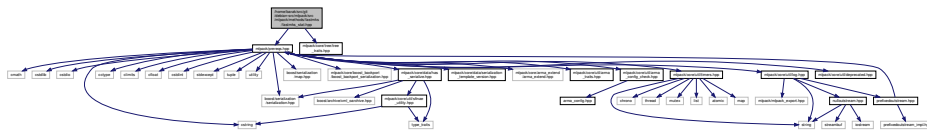
Ryan Curtin

Rules for the single or dual tree traversal for fast max-kernel search.

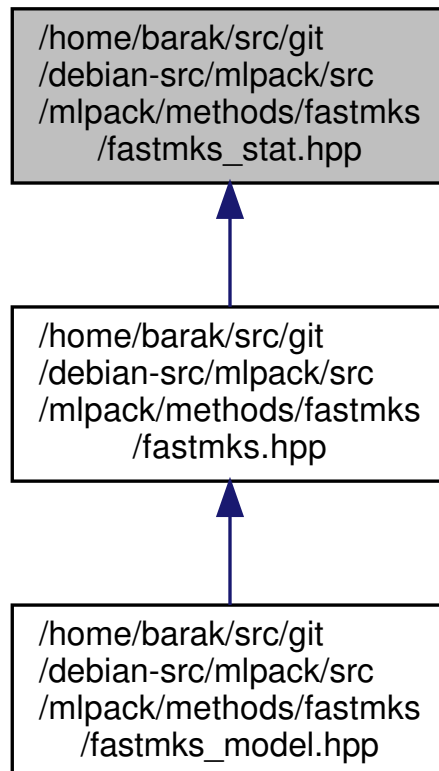
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.550 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/fastmks/fastmks_stat.hpp File Reference

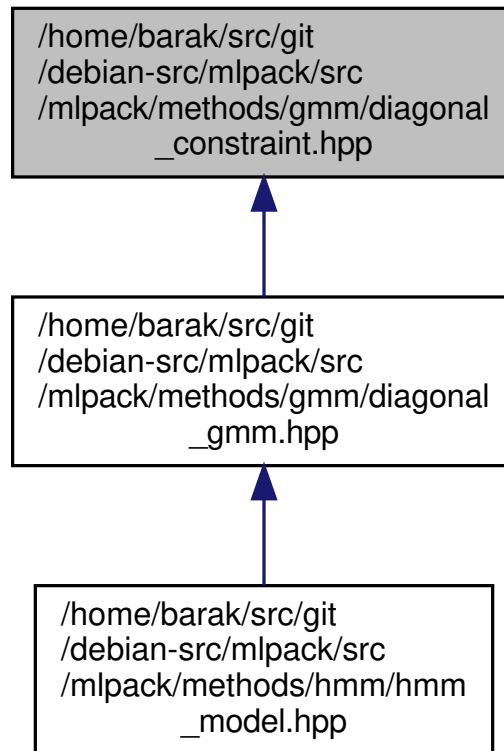
Include dependency graph for fastmks_stat.hpp:



This graph shows which files directly or indirectly include this file:



This graph shows which files directly or indirectly include this file:



Classes

- class **DiagonalConstraint**
Force a covariance matrix to be diagonal.

Namespaces

- **mlpack**
.hpp
- **mlpack::gmm**
Gaussian Mixture Models.

40.551.1 Detailed Description

Author

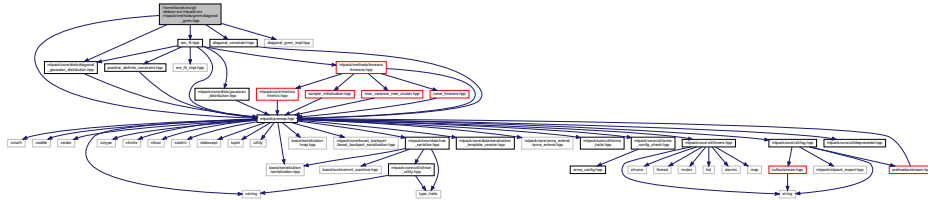
Ryan Curtin

Constrain a covariance matrix to be diagonal.

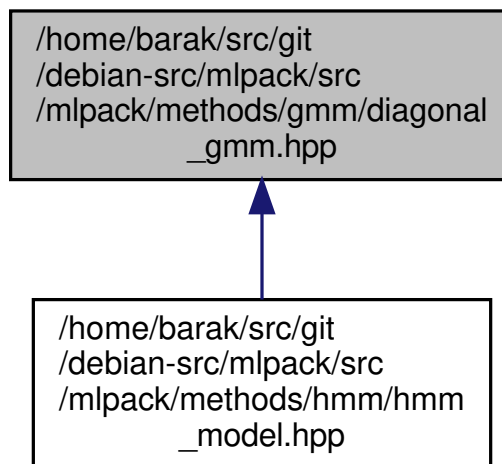
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.552 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/diagonal_gmm.hpp File Reference

Include dependency graph for diagonal_gmm.hpp:



This graph shows which files directly or indirectly include this file:



Classes

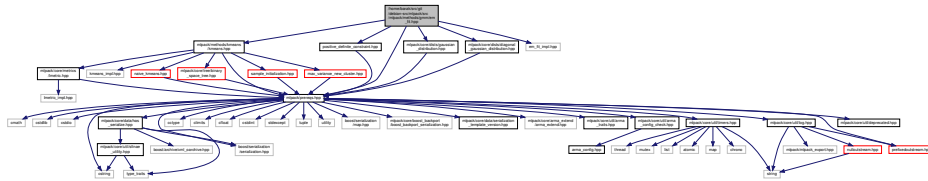
- class **DiagonalGMM**
A Diagonal Gaussian Mixture Model.

Namespaces

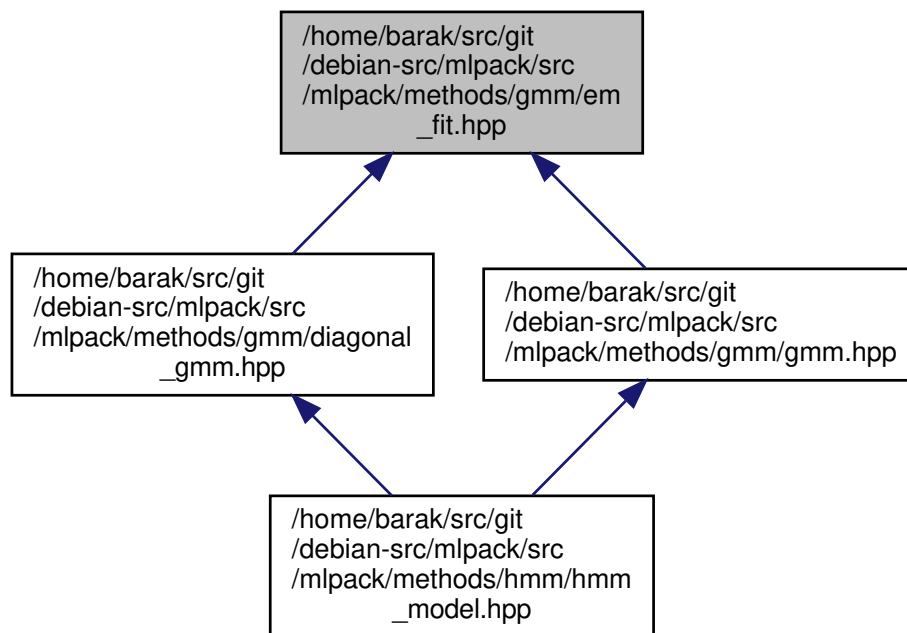
- **mlpack**
.hpp
- **mlpack::gmm**
Gaussian Mixture Models.

40.554 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/em_fit.hpp File Reference

Include dependency graph for em_fit.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **EMFit**< **InitialClusteringType**, **CovarianceConstraintPolicy**, **Distribution** >
*This class contains methods which can fit a **GMM** (p. 1323) to observations using the EM algorithm.*

Namespaces

- **mlpack**
.hpp
- **mlpack::gmm**
Gaussian Mixture Models.

40.556.1 Detailed Description

Author

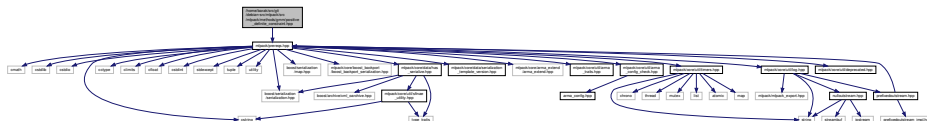
Ryan Curtin

No constraint on the covariance matrix.

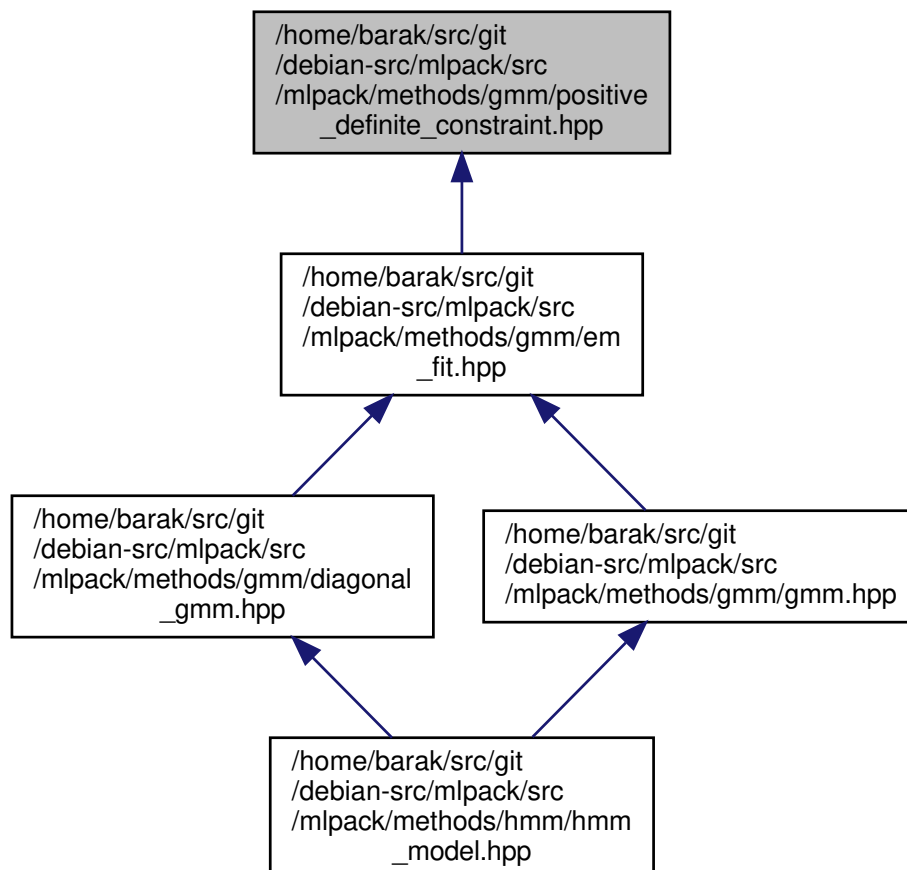
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.557 `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/gmm/positive_definite_constraint.hpp` File Reference

Include dependency graph for `positive_definite_constraint.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **PositiveDefiniteConstraint**
Given a covariance matrix, force the matrix to be positive definite.

Namespaces

- mlpack**
.hpp
- mlpack::gmm**
Gaussian Mixture Models.

40.557.1 Detailed Description

Author

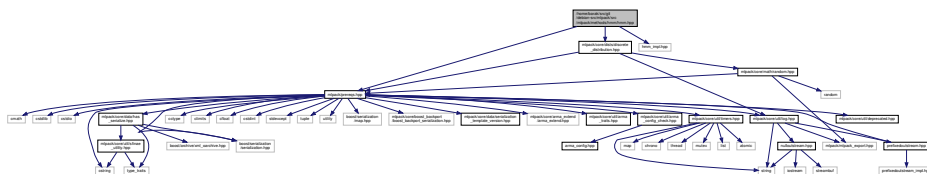
Ryan Curtin

Restricts a covariance matrix to being positive definite.

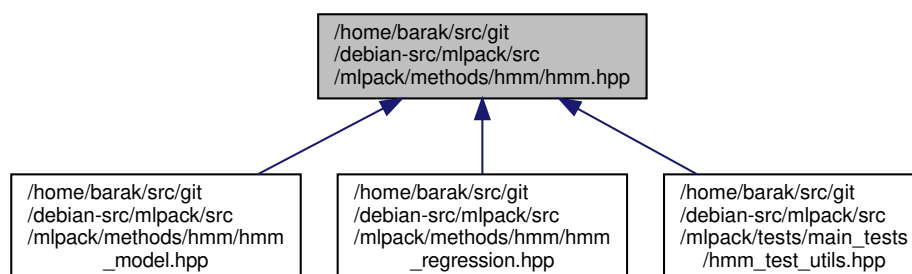
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.558 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hmm/hmm.hpp File Reference

Include dependency graph for hmm.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **HMM** < **Distribution** >

A class that represents a Hidden Markov Model with an arbitrary type of emission distribution.

Namespaces

- **mlpack**

.hpp

- **mlpack::hmm**

Hidden Markov Models.

40.558.1 Detailed Description

Author

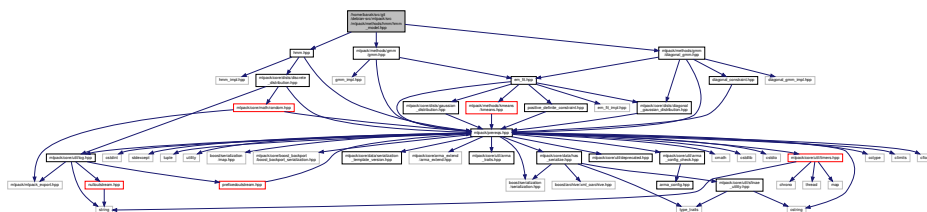
Ryan Curtin
Tran Quoc Long
Michael Fox

Definition of HMM class.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.559 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hmm/hmm_model.hpp File Reference

Include dependency graph for hmm_model.hpp:



Classes

- class **HMMModel**

*A serializable **HMM** (p. 1335) model that also stores the type.*

Namespaces

- **mlpack**
.hpp
- **mlpack::hmm**
Hidden Markov Models.

Enumerations

- enum **HMMType** : char {
 DiscreteHMM = 0,
 GaussianHMM,
 GaussianMixtureModelHMM,
 DiagonalGaussianMixtureModelHMM,
 DiscreteHMM = 0,
 GaussianHMM,
 GaussianMixtureModelHMM,
 DiagonalGaussianMixtureModelHMM }

Functions

- **BOOST_CLASS_VERSION** (**mlpack::hmm::HMMModel**, 1)
Set the serialization version of the HMMModel class.

40.559.1 Detailed Description

Author

Ryan Curtin

A serializable HMM model that also stores the type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.559.2 Function Documentation

40.559.2.1 BOOST_CLASS_VERSION()

```
BOOST_CLASS_VERSION (
    mlpack::hmm::HMMModel ,
    1 )
```

Set the serialization version of the HMMModel class.

Referenced by `HMMModel::DiagGMMHMM()`.

Namespaces

- **mlpack**
.hpp
- **mlpack::hmm**
Hidden Markov Models.

Enumerations

- enum **HMMType** : char {
 DiscreteHMM = 0,
 GaussianHMM,
 GaussianMixtureModelHMM,
 DiagonalGaussianMixtureModelHMM,
 DiscreteHMM = 0,
 GaussianHMM,
 GaussianMixtureModelHMM,
 DiagonalGaussianMixtureModelHMM }
HMMType, to be stored on disk.

Functions

- template<typename ActionType , typename ExtraInfoType = void>
 void **LoadHMMAndPerformAction** (const std::string &modelFile, ExtraInfoType *x=NULL)
 ActionType should implement static void Apply(HMMType&).
- template<typename HMMType >
 void **SaveHMM** (HMMType &hmm, const std::string &modelFile)
 *Save an **HMM** (p. 1335) to a file.*

40.561.1 Detailed Description

Author

Ryan Curtin

Utility to read HMM type from file.

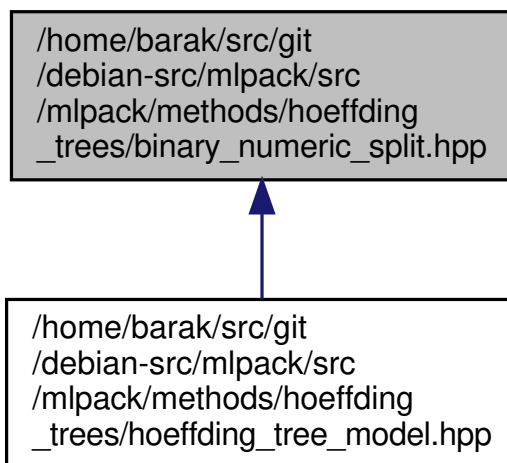
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.562 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/binary_numeric_split.hpp File Reference

Include dependency graph for binary_numeric_split.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **BinaryNumericSplit**< **FitnessFunction**, **ObservationType** >

The ***BinaryNumericSplit*** (p. 1992) class implements the numeric feature splitting strategy devised by Gama, Rocha, and Medas in the following paper:

Namespaces

- mlpack**
.*hpp*
- mlpack::tree**
Trees and tree-building procedures.

Typedefs

- template<typename **FitnessFunction** >
using **BinaryDoubleNumericSplit** = **BinaryNumericSplit**< **FitnessFunction**, double >

40.562.1 Detailed Description

Author

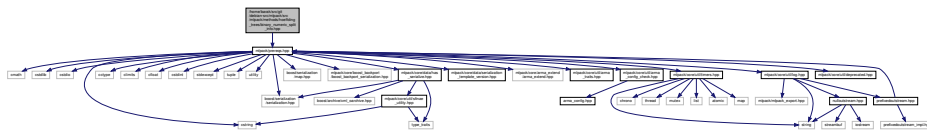
Ryan Curtin

An implementation of the binary-tree-based numeric splitting procedure described by Gama, Rocha, and Medas in their KDD 2003 paper.

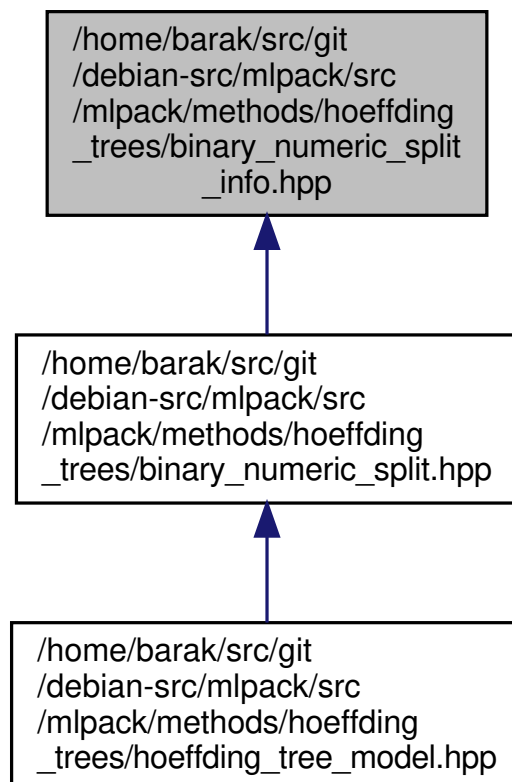
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.563 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/binary_numeric_split_info.hpp File Reference

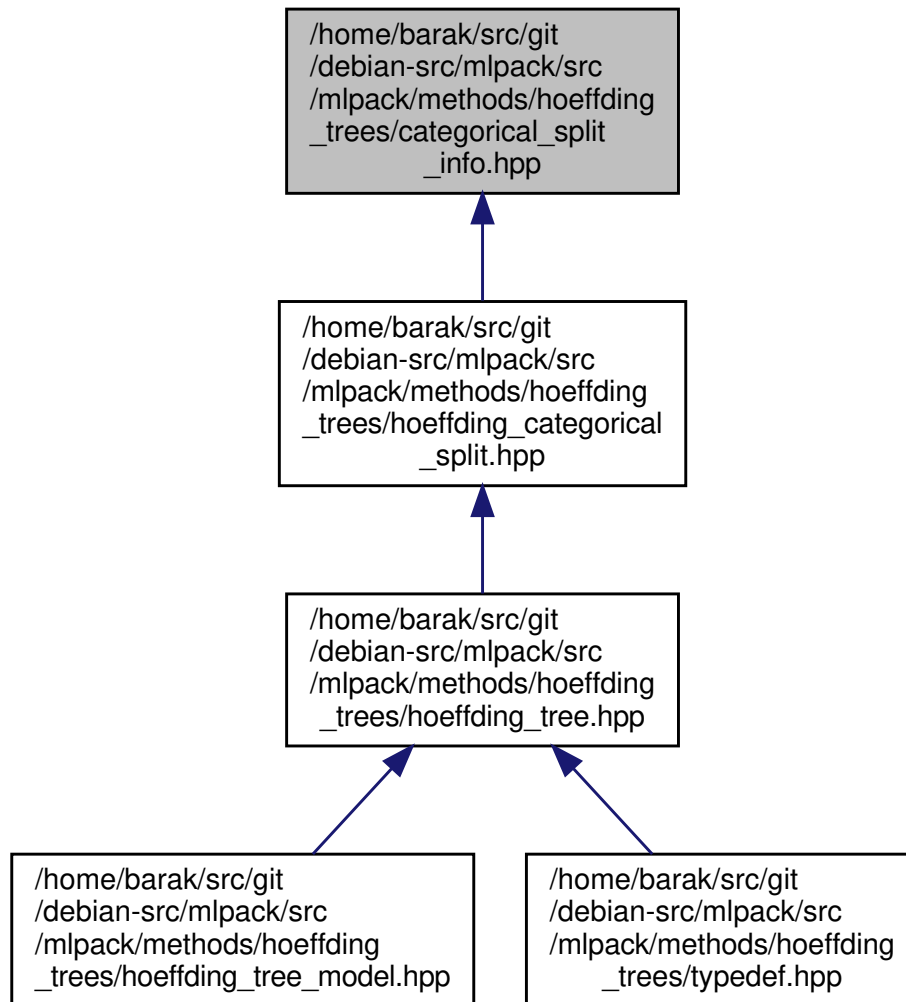
Include dependency graph for binary_numeric_split_info.hpp:



This graph shows which files directly or indirectly include this file:



This graph shows which files directly or indirectly include this file:



Classes

- class **CategoricalSplitInfo**

Namespaces

- **mlpack**
.hpp
- **mlpack::tree**
Trees and tree-building procedures.

40.564.1 Detailed Description

Author

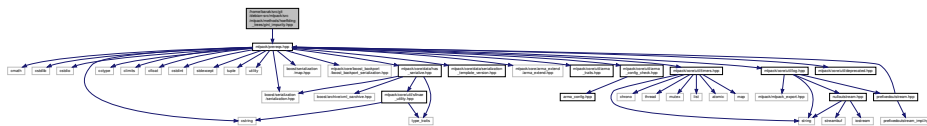
Ryan Curtin

After a categorical split has been made, this holds information on the split.

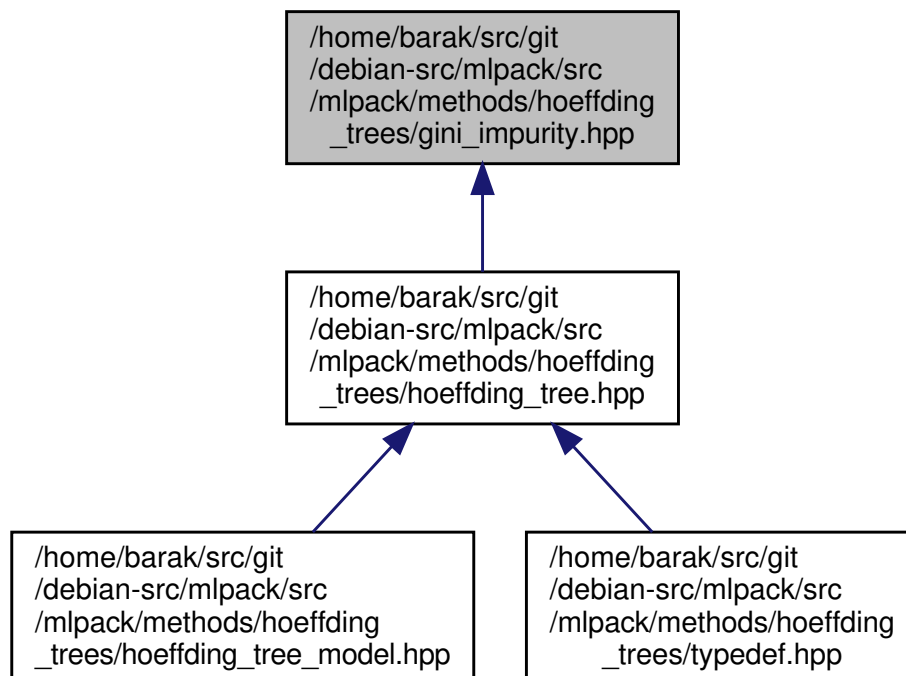
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.565 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/gini_impurity.hpp File Reference

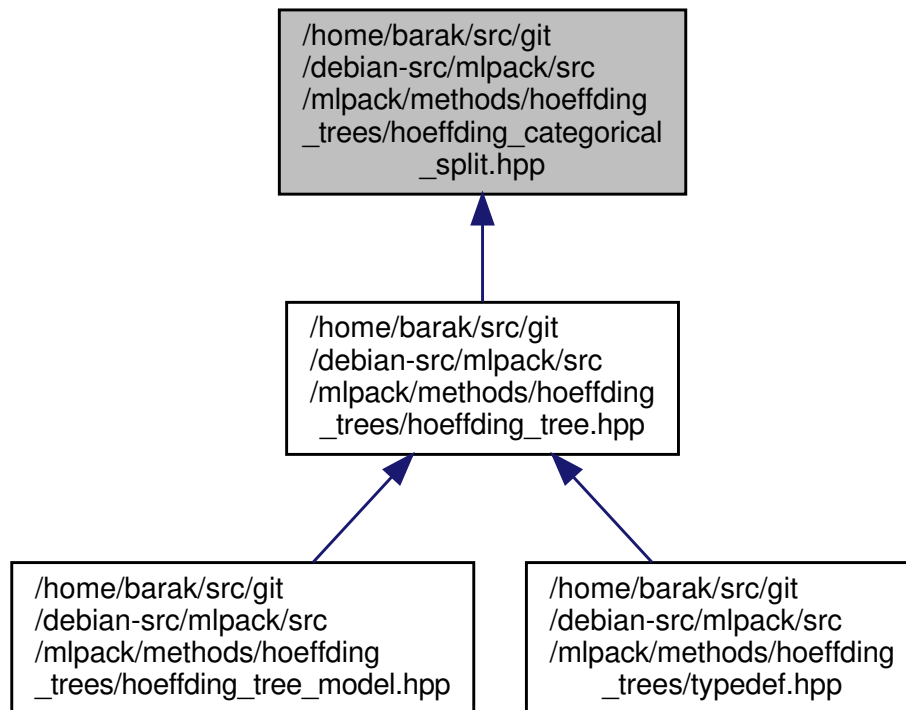
Include dependency graph for gini_impurity.hpp:



This graph shows which files directly or indirectly include this file:



This graph shows which files directly or indirectly include this file:



Classes

- class **HoeffdingCategoricalSplit**< **FitnessFunction** >

This is the standard Hoeffding-bound categorical feature proposed in the paper below:

Namespaces

- **mlpack**
 .hpp
- **mlpack::tree**

Trees and tree-building procedures.

40.566.1 Detailed Description

Author

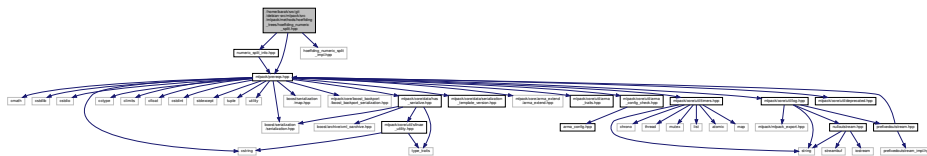
Ryan Curtin

A class that contains the information necessary to perform a categorical split for Hoeffding trees.

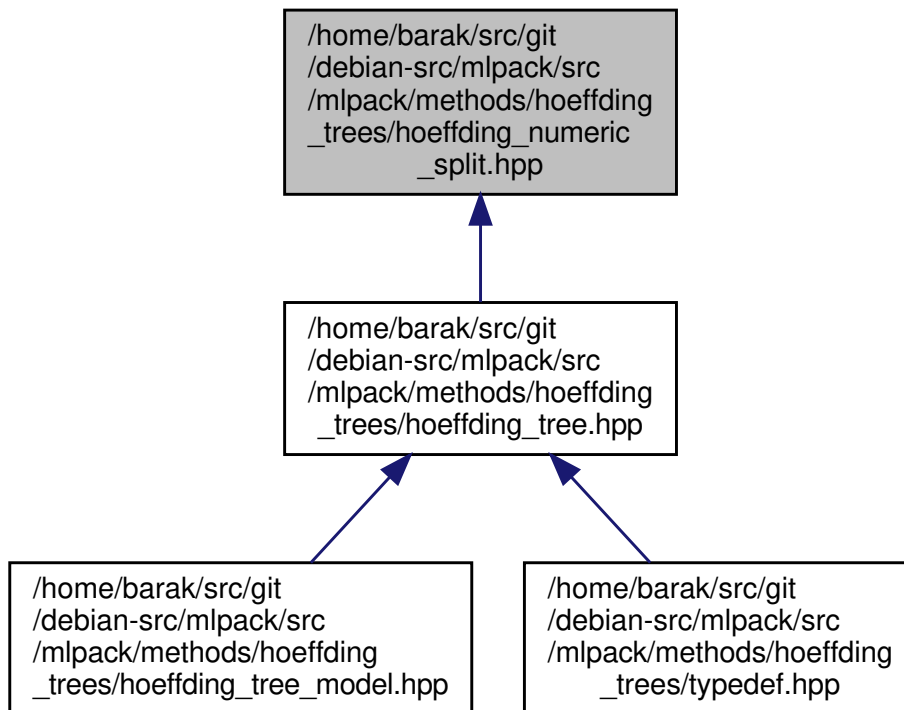
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.567 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/hoeffding_numeric_split.hpp File Reference

Include dependency graph for hoeffding_numeric_split.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **HoeffdingNumericSplit**< **FitnessFunction**, **ObservationType** >

The **HoeffdingNumericSplit** (p. 2108) class implements the numeric feature splitting strategy alluded to by Domingos and Hulten in the following paper:

Namespaces

- **mlpack**
 .hpp
- **mlpack::tree**

Trees and tree-building procedures.

Typedefs

- `template<typename FitnessFunction >`
using **HoeffdingDoubleNumericSplit** = `HoeffdingNumericSplit< FitnessFunction, double >`
Convenience typedef.

40.567.1 Detailed Description

Author

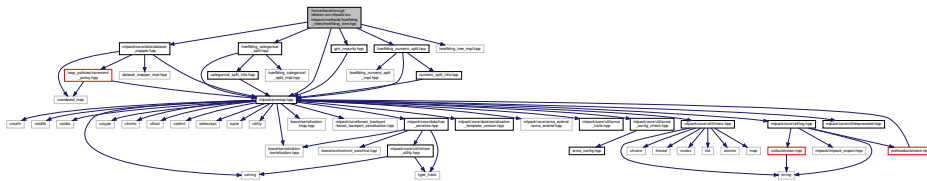
Ryan Curtin

A numeric feature split for Hoeffding trees. This is a very simple implementation based on a minor note in the paper "Mining High-Speed Data Streams" by Pedro Domingos and Geoff Hulten.

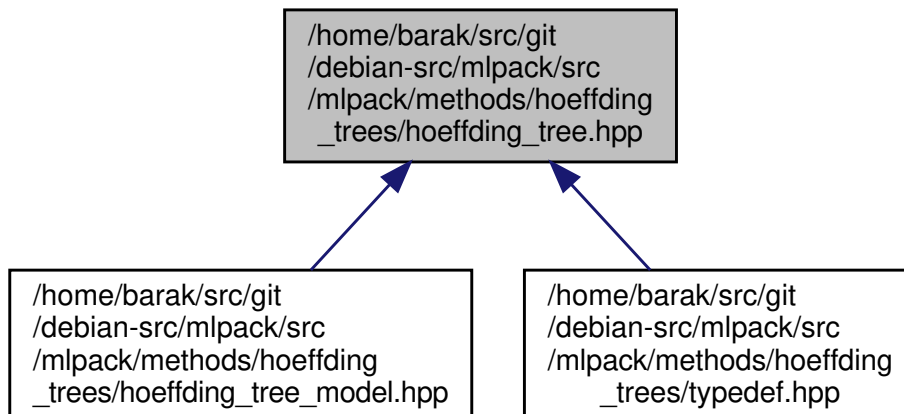
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.568 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/hoeffding_tree.hpp File Reference

Include dependency graph for `hoeffding_tree.hpp`:



This graph shows which files directly or indirectly include this file:

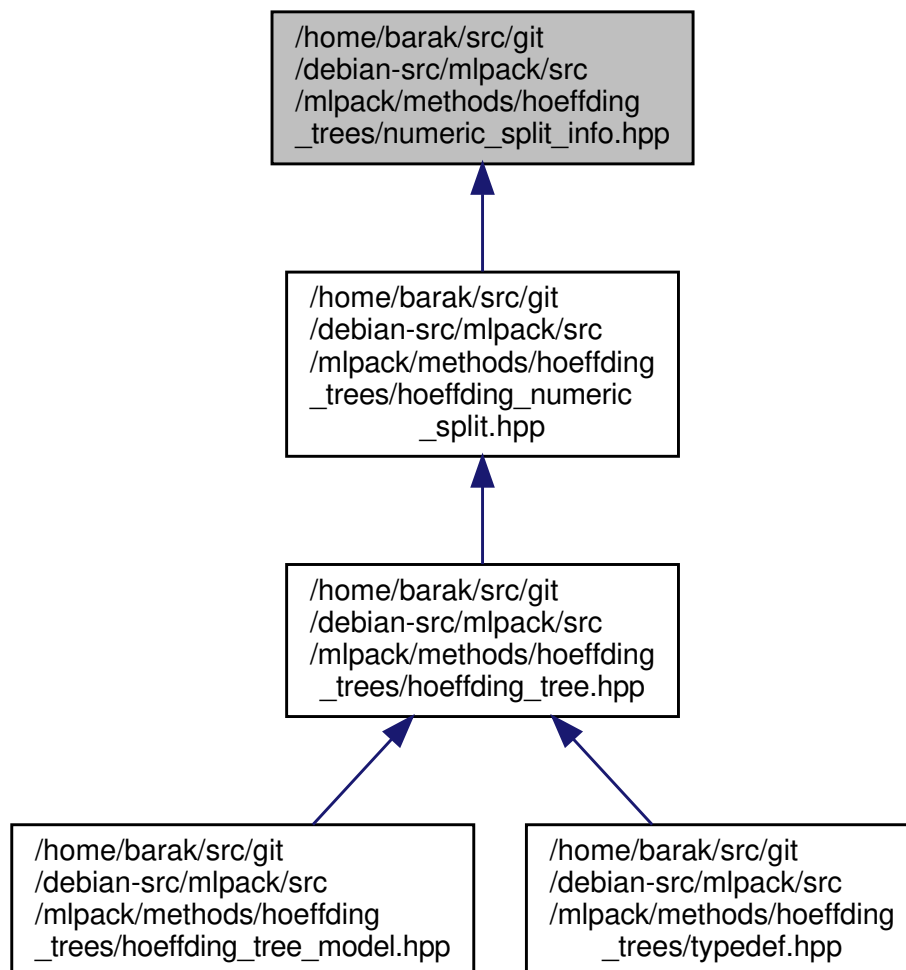


40.570 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding_trees/numeric_split_info.hpp File Reference

Include dependency graph for numeric_split_info.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **NumericSplitInfo**< **ObservationType** >

Namespaces

- **mlpack**
 .hpp
- **mlpack::tree**
 Trees and tree-building procedures.

40.570.1 Detailed Description

Author

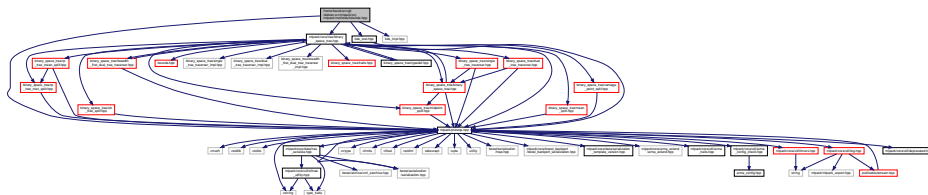
Ryan Curtin

After a numeric split has been made, this holds information on the split.

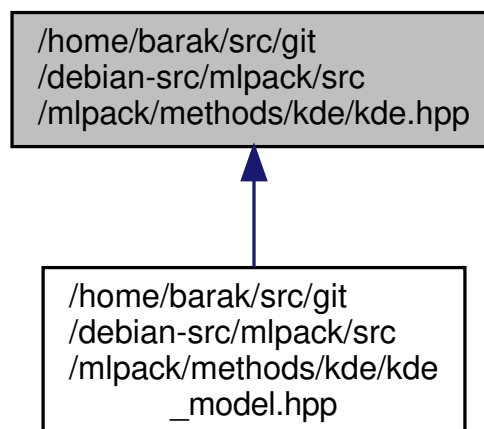
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.571 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/kde.hpp File Reference

Include dependency graph for kde.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **KDE**< **KernelType**, **MetricType**, **MatType**, **TreeType**, **DualTreeTraversalType**, **SingleTreeTraversalType** <→

The **KDE** (p. 1387) class is a template class for performing Kernel Density Estimations.

Namespaces

- **mlpack**

.hpp

- **mlpack::kde**

Kernel Density Estimation.

Enumerations

- enum **KDEMode** {
DUAL_TREE_MODE,
SINGLE_TREE_MODE }

KDEMode represents the ways in which KDE algorithm can be executed.

40.571.1 Detailed Description

Author

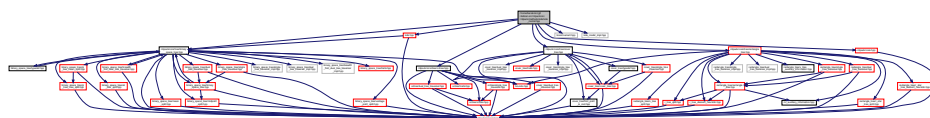
Roberto Hueso

Kernel Density Estimation.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.572 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/kde_model.hpp File Reference

Include dependency graph for kde_model.hpp:



Classes

- class **DeleteVisitor**
- class **DualBiKDE**

DualBiKDE (p. 1384) computes a Kernel Density Estimation on the given KDEType.
- class **DualMonoKDE**

DualMonoKDE (p. 1386) computes a Kernel Density Estimation on the given KDEType.
- class **KDEModel**
- class **KernelNormalizer**

KernelNormalizer (p. 1410) holds a set of methods to normalize estimations applying in each case the appropriate kernel normalizer function.
- class **ModeVisitor**

ModeVisitor (p. 1412) exposes the Mode() method of the KDEType.
- class **TrainVisitor**

TrainVisitor (p. 1413) trains a given KDEType using a reference set.

Namespaces

- **mlpack**

.hpp
- **mlpack::kde**

Kernel Density Estimation.

Typedefs

- `template<typename KernelType, template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType>`
`using KDEType = KDE< KernelType, metric::EuclideanDistance, arma::mat, TreeType, TreeType< metric::EuclideanDistance, kde::KDEStat, arma::mat >::template DualTreeTraverser, TreeType< metric::EuclideanDistance, kde::KDEStat, arma::mat >::template SingleTreeTraverser >`
Alias template.

40.572.1 Detailed Description

Author

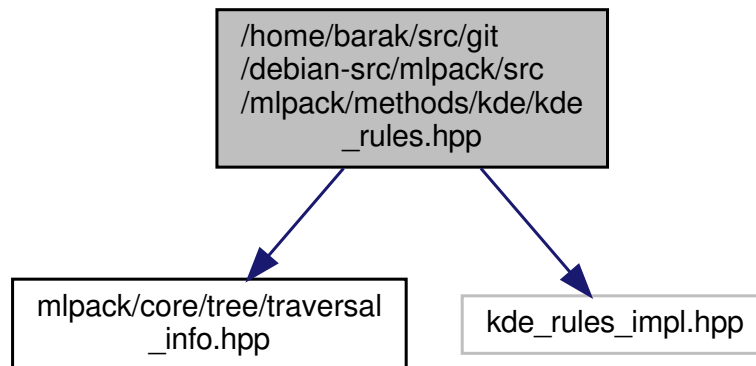
Roberto Hueso

Model for KDE. It abstracts different types of tree, kernels, etc.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.573 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/kde_rules.hpp File Reference

Include dependency graph for kde_rules.hpp:



Classes

- class **KDERules**< **MetricType**, **KernelType**, **TreeType** >
A dual-tree traversal Rules class for kernel density estimation.

Namespaces

- **mlpack**
.hpp
- **mlpack::kde**
Kernel Density Estimation.

40.573.1 Detailed Description

Author

Roberto Hueso

Rules for Kernel Density Estimation, so that it can be done with arbitrary tree types.

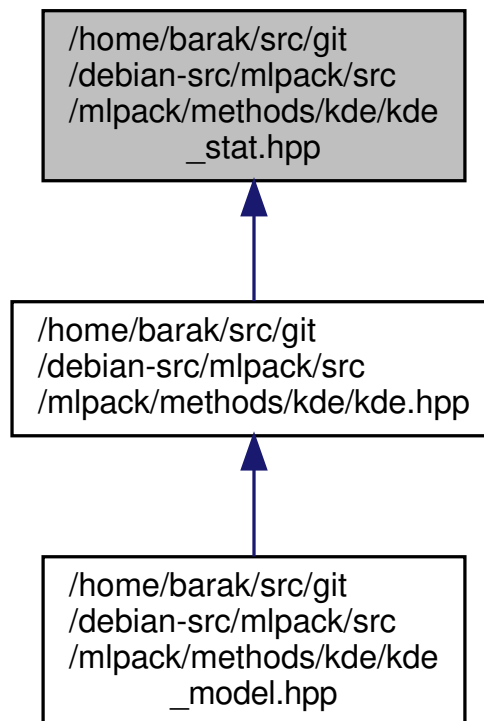
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.574 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kde/kde_stat.hpp File Reference

Include dependency graph for kde_stat.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **KDEStat**

Extra data for each node in the tree for the task of kernel density estimation.

Namespaces

- **mlpack**
 .hpp
- **mlpack::kde**

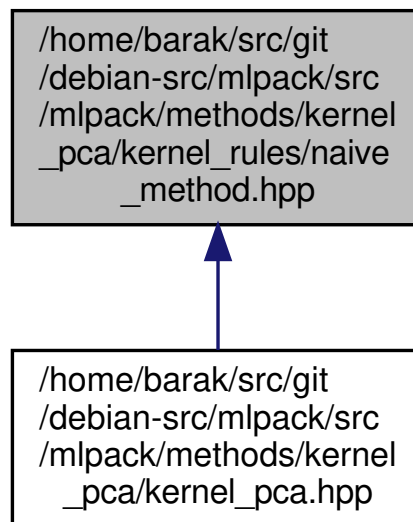
Kernel Density Estimation.

40.576 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kernel_pca/kernel_rules/naive_method.hpp File Reference

Include dependency graph for naive_method.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **NaiveKernelRule**< **KernelType** >

Namespaces

- **mlpack**
 .hpp
- **mlpack::kpca**

40.576.1 Detailed Description

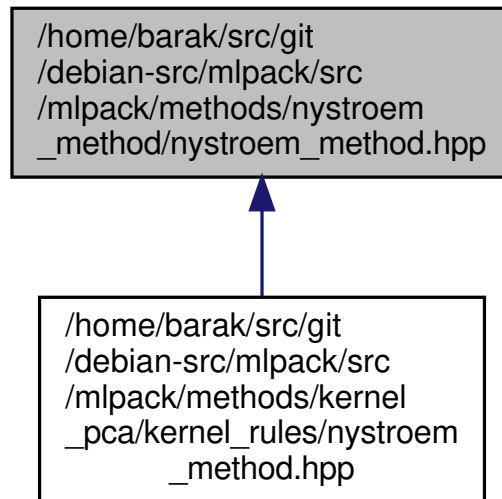
Author

Ajinkya Kale

Use the naive method to construct the kernel matrix.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

This graph shows which files directly or indirectly include this file:



Classes

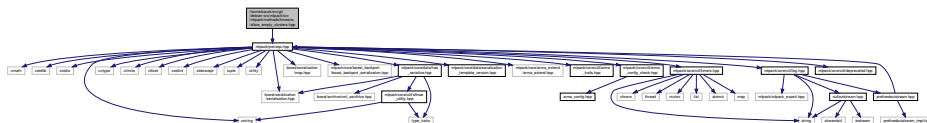
- class **NystroemMethod**< **KernelType**, **PointSelectionPolicy** >

Namespaces

- **mlpack**
 .hpp
- **mlpack::kernel**
 Kernel functions.

40.579 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/allow_empty_clusters.hpp File Reference

Include dependency graph for allow_empty_clusters.hpp:



Classes

- class **AllowEmptyClusters**
 Policy which allows K-Means to create empty clusters without any error being reported.

Typedefs

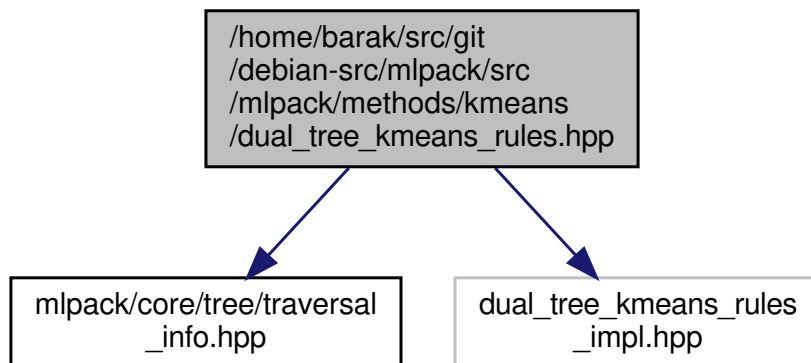
- `template<typename MetricType , typename MatType >`
`using CoverTreeDualTreeKMeans = DualTreeKMeans< MetricType, MatType, tree::StandardCoverTree >`
*A template typedef for the **DualTreeKMeans** (p. 1466) algorithm with the cover tree type.*
- `template<typename MetricType , typename MatType >`
`using DefaultDualTreeKMeans = DualTreeKMeans< MetricType, MatType >`
*A template typedef for the **DualTreeKMeans** (p. 1466) algorithm with the default tree type (a kd-tree).*

Functions

- `template<typename TreeType >`
`void HideChild (TreeType &node, const size_t child, const typename std::enable_if_t< !tree::TreeTraits< TreeType >::BinaryTree > *junk=0)`
Utility function for hiding children.
- `template<typename TreeType >`
`void HideChild (TreeType &node, const size_t child, const typename std::enable_if_t< tree::TreeTraits< TreeType >::BinaryTree > *junk=0)`
Utility function for hiding children.
- `template<typename TreeType >`
`void RestoreChildren (TreeType &node, const typename std::enable_if_t<!tree::TreeTraits< TreeType >::BinaryTree > *junk=0)`
Utility function for restoring children to a non-binary tree.
- `template<typename TreeType >`
`void RestoreChildren (TreeType &node, const typename std::enable_if_t< tree::TreeTraits< TreeType >::BinaryTree > *junk=0)`
Utility function for restoring children to a binary tree.

40.581 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/dual_tree_kmeans_rules.hpp File Reference

Include dependency graph for dual_tree_kmeans_rules.hpp:

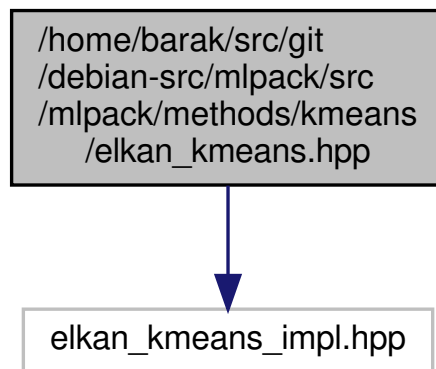


Namespaces

- **mlpack**
.hpp
- **mlpack::kmeans**
K-Means clustering.

40.583 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/elkan_kmeans.hpp File Reference

Include dependency graph for elkan_kmeans.hpp:



Classes

- class **ElkanKMeans**< **MetricType**, **MatType** >

Namespaces

- **mlpack**
.hpp
- **mlpack::kmeans**
K-Means clustering.

40.583.1 Detailed Description

Author

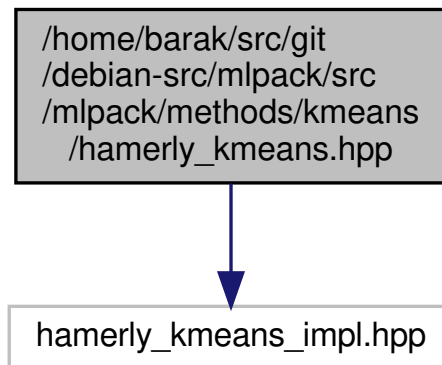
Ryan Curtin

An implementation of Elkan's algorithm for exact Lloyd iterations.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.584 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/hamerly_↵ kmeans.hpp File Reference

Include dependency graph for hamerly_kmeans.hpp:



Classes

- class **HamerlyKMeans**< **MetricType**, **MatType** >

Namespaces

- **mlpack**
 .hpp
- **mlpack::kmeans**
 K-Means clustering.

40.584.1 Detailed Description

Author

Ryan Curtin

An implementation of Greg Hamerly's algorithm for k-means clustering.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.585 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/kill_empty_clusters.hpp File Reference

Include dependency graph for kill_empty_clusters.hpp:



Classes

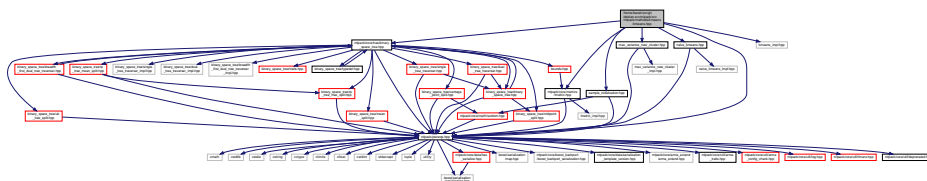
- class **KillEmptyClusters**
Policy which allows K-Means to "kill" empty clusters without any error being reported.

Namespaces

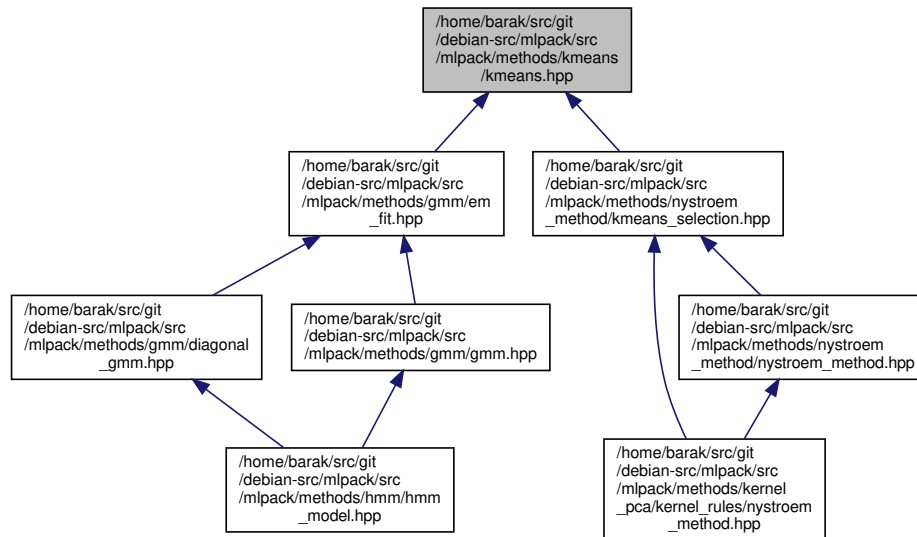
- **mlpack**
.hpp
- **mlpack::kmeans**
K-Means clustering.

40.586 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/kmeans.hpp File Reference

Include dependency graph for kmeans.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **KMeans**< **MetricType**, **InitialPartitionPolicy**, **EmptyClusterPolicy**, **LloydStepType**, **MatType** >

This class implements K-Means clustering, using a variety of possible implementations of Lloyd's algorithm.

Namespaces

- **mlpack**
.*hpp*
- **mlpack::kmeans**
K-Means clustering.

40.586.1 Detailed Description

Author

Parikshit Ram (pram@cc.gatech.edu)

K-Means clustering.

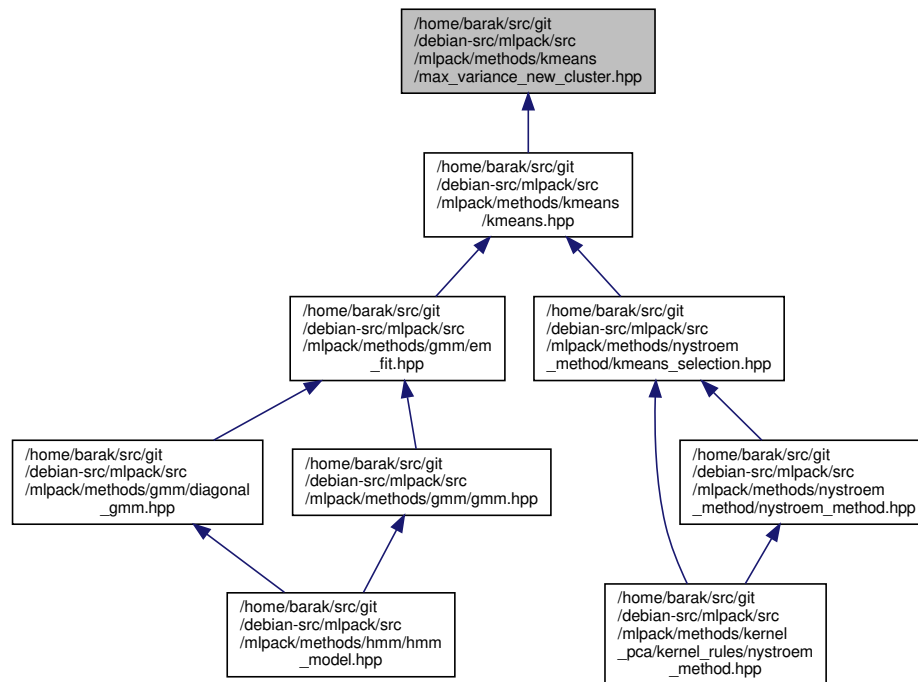
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.587 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/max_variance_new_cluster.hpp File Reference

Include dependency graph for max_variance_new_cluster.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **MaxVarianceNewCluster**

When an empty cluster is detected, this class takes the point furthest from the centroid of the cluster with maximum variance as a new cluster.

Namespaces

- mlpack**
.hpp
- mlpack::kmeans**
K-Means clustering.

40.587.1 Detailed Description

Author

Ryan Curtin

An implementation of the EmptyClusterPolicy policy class for K-Means. When an empty cluster is detected, the point furthest from the centroid of the cluster with maximum variance is taken to be a new cluster.

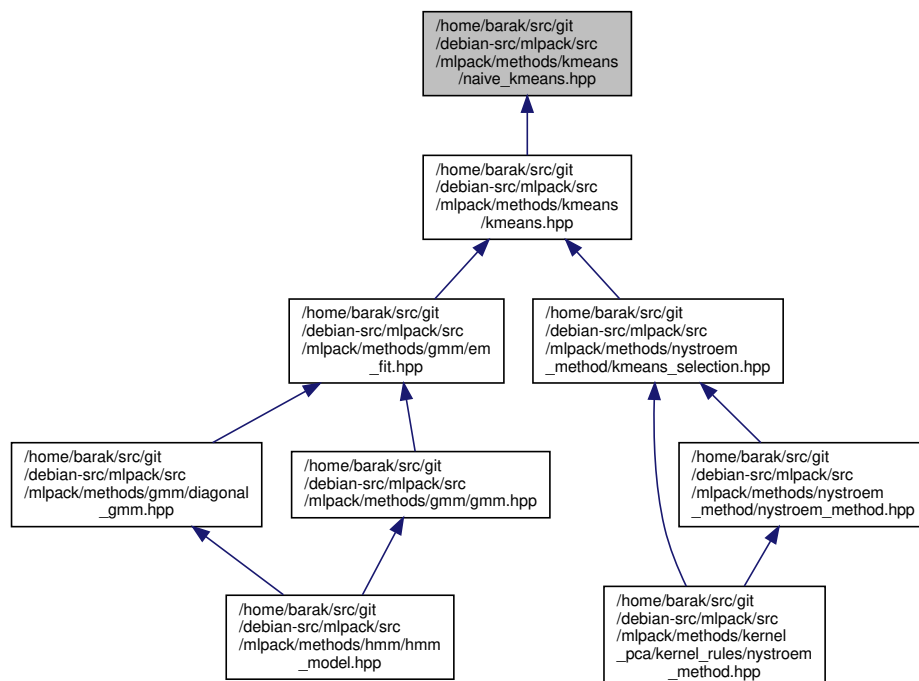
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.588 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/naive_kmeans.hpp File Reference

Include dependency graph for naive_kmeans.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **NaiveKMeans** < **MetricType**, **MatType** >

This is an implementation of a single iteration of Lloyd's algorithm for k-means.

Namespaces

- **mlpack**

.hpp

- **mlpack::kmeans**

K-Means clustering.

40.588.1 Detailed Description

Author

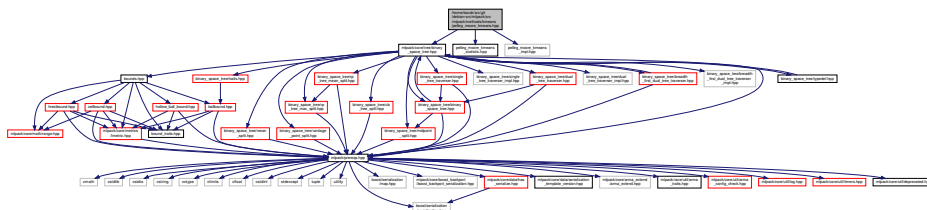
Ryan Curtin

An implementation of a naively-implemented step of the Lloyd algorithm for k-means clustering, using OpenMP for parallelization over multiple threads. This may still be the best choice for small datasets or datasets with very high dimensionality.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.589 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/pelleg_moore_↵
_kmeans.hpp File Reference

Include dependency graph for pelleg_moore_kmeans.hpp:



Classes

- class **PellegMooreKMeans**< **MetricType**, **MatType** >

An implementation of Pelleg-Moore's 'blacklist' algorithm for k-means clustering.

Namespaces

- **mlpack**
 .hpp
- **mlpack::kmeans**
 K-Means clustering.

40.589.1 Detailed Description

Author

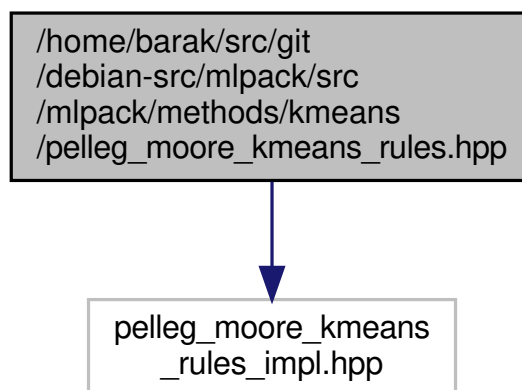
Ryan Curtin

An implementation of Pelleg-Moore's 'blacklist' algorithm for k-means clustering.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.590 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/pelleg_moore_kmeans_rules.hpp File Reference

Include dependency graph for pelleg_moore_kmeans_rules.hpp:



Classes

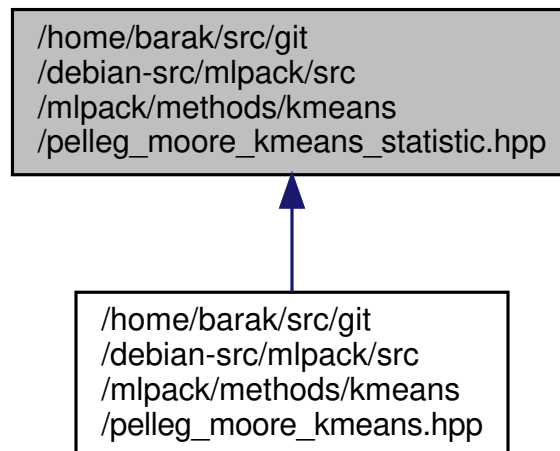
- class **PellegMooreKMeansRules**< **MetricType**, **TreeType** >
 The rules class for the single-tree Pelleg-Moore kd-tree traversal for k-means clustering.

Namespaces

- **mlpack**
 .hpp
- **mlpack::kmeans**
 K-Means clustering.

40.591 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/pelleg_moore_kmeans_statistic.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **PellegMooreKMeansStatistic**
 A statistic for trees which holds the blacklist for Pelleg-Moore k-means clustering (which represents the clusters that cannot possibly own any points in a node).

Namespaces

- **mlpack**
 .hpp
- **mlpack::kmeans**
 K-Means clustering.

40.593 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/refined_start.hpp File Reference

Include dependency graph for refined_start.hpp:



Classes

- class **RefinedStart**
A refined approach for choosing initial points for k-means clustering.

Namespaces

- **mlpack**
.hpp
- **mlpack::kmeans**
K-Means clustering.

40.593.1 Detailed Description

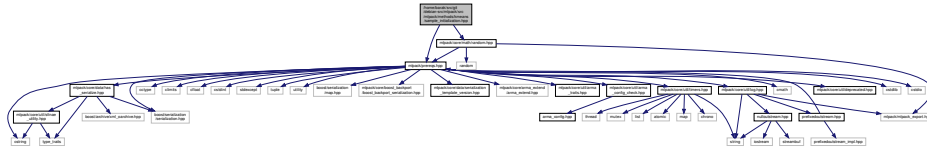
Author

Ryan Curtin

An implementation of Bradley and Fayyad's "Refining Initial Points for K-Means clustering". This class is meant to provide better initial points for the k-means algorithm.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Include dependency graph for sample_initialization.hpp:



```

graph TD
    A["/home/barak/src/git  
/debian-src/mlpack/src  
/mlpack/methods/kmeans  
/sample_initialization.hpp"]
    B["/home/barak/src/git  
/debian-src/mlpack/src  
/mlpack/methods/kmeans  
/kmeans.hpp"]
    C["/home/barak/src/git  
/debian-src/mlpack/src  
/mlpack/methods/gmm/em_fit.hpp"]
    D["/home/barak/src/git  
/debian-src/mlpack/src  
/mlpack/methods/nystroem  
_method/kmeans_selection.hpp"]
    E["/home/barak/src/git  
/debian-src/mlpack/src  
/mlpack/methods/diagonal_gmm.hpp"]
    F["/home/barak/src/git  
/debian-src/mlpack/src  
/mlpack/methods/gmm/gmm.hpp"]
    G["/home/barak/src/git  
/debian-src/mlpack/src  
/mlpack/methods/nystroem  
_method/nystroem_method.hpp"]
    H["/home/barak/src/git  
/debian-src/mlpack/src  
/mlpack/methods/hmm/hmm_model.hpp"]
    I["/home/barak/src/git  
/debian-src/mlpack/src  
/mlpack/methods/kernel_pca/kernel_rules/nystroem_method.hpp"]

    A --> B
    B --> C
    B --> D
    C --> E
    C --> F
    D --> G
    D --> I
    E --> H
    F --> H
    G --> I
    I --> H
  
```

- class **SampleInitialization**

- **mlpack**
.hpp
- **mlpack::kmeans**
K-Means clustering.

40.594.1 Detailed Description

Author

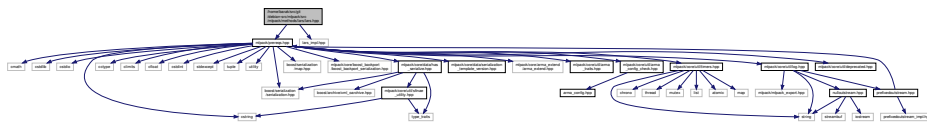
Ryan Curtin

In order to construct initial centroids, randomly sample points from the dataset. This tends to give better results than the RandomPartition strategy.

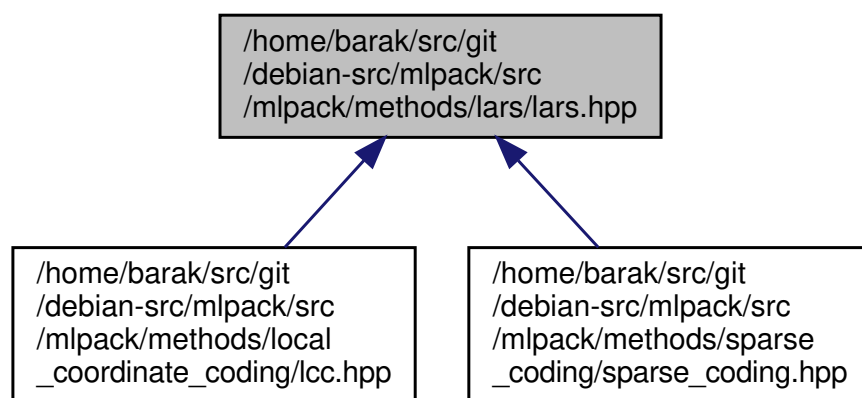
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.595 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lars/lars.hpp File Reference

Include dependency graph for lars.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **LARS**

An implementation of **LARS** (p. 1783), a stage-wise homotopy-based algorithm for l_1 -regularized linear regression (LASSO) and l_1+l_2 regularized linear regression (Elastic Net).

Namespaces

- **mlpack**
 .hpp
- **mlpack::regression**
 Regression methods.

40.595.1 Detailed Description

Author

Nishant Mehta (niche)

Definition of the LARS class, which performs Least Angle Regression and the LASSO.

Only minor modifications of LARS are necessary to handle the constrained version of the problem:

$$\min_{\beta} 0.5 \|X\beta - y\|_2^2 + 0.5\lambda_2 \|\beta\|_2^2$$

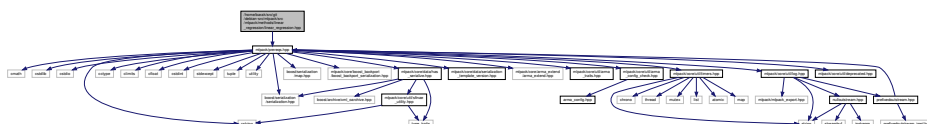
subject to $\|\beta\|_1 \leq \tau$

Although this option currently is not implemented, it will be implemented very soon.

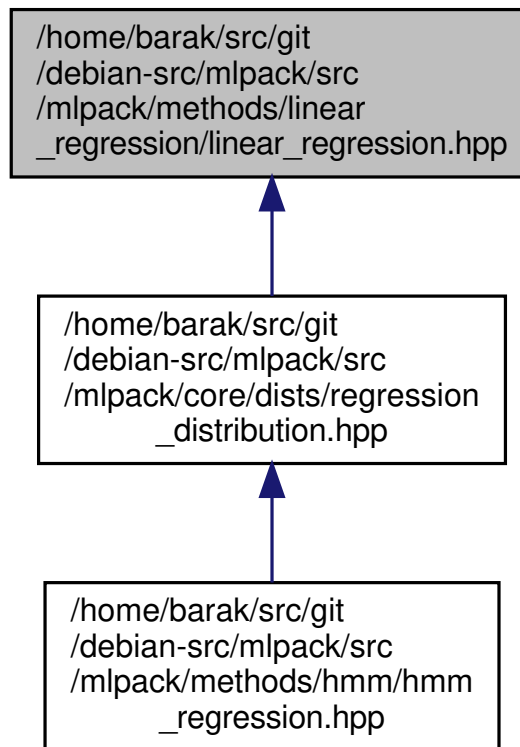
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.596 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear_regression/linear_regression.hpp File Reference

Include dependency graph for linear_regression.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **LinearRegression**
A simple linear regression algorithm using ordinary least squares.

Namespaces

- **mlpack**
.hpp
- **mlpack::regression**
Regression methods.

40.596.1 Detailed Description

Author

James Cline
Michael Fox

Simple least-squares linear regression.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.


```
graph BT; A["/home/barak/src/git  
/debian-src/mlpack/src  
/mlpack/methods/linear  
_svm/linear_svm.hpp"] --> B["/home/barak/src/git  
/debian-src/mlpack/src  
/mlpack/methods/linear  
_svm/linear_svm_function.hpp"]
```

/home/barak/src/git
/debian-src/mlpack/src
/mlpack/methods/linear
_svm/linear_svm_function.hpp

/home/barak/src/git
/debian-src/mlpack/src
/mlpack/methods/linear
_svm/linear_svm.hpp

- class **LinearSVMFunction**< **MatType** >
The hinge loss function for the linear SVM objective function.

- **mlpack**
 .hpp
- **mlpack::svm**

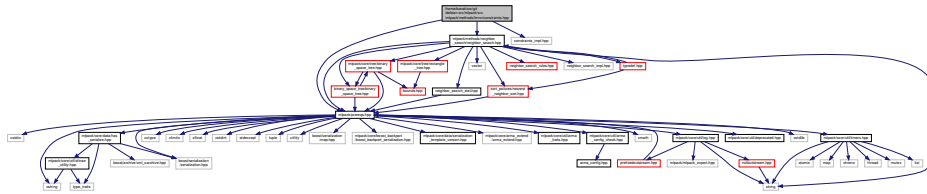
Shikhar Bhardwaj
Ayush Chamoli

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

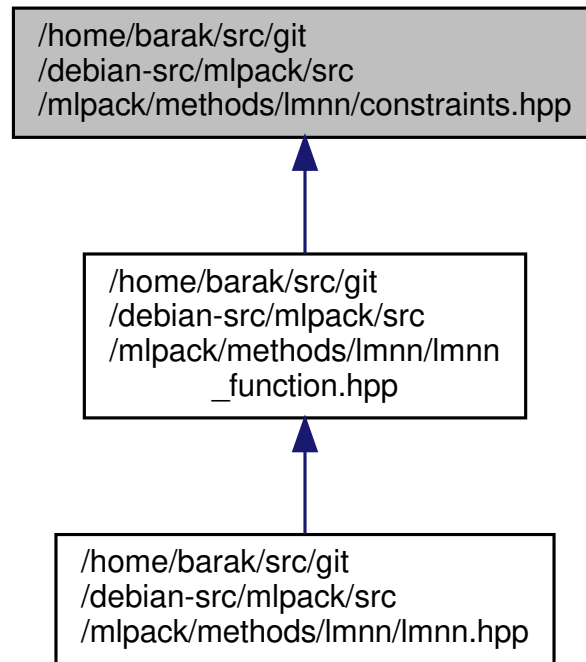
40.599 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lmnn/constraints.hpp

File Reference

Include dependency graph for constraints.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **Constraints**< **MetricType** >
Interface for generating distance based constraints on a given dataset, provided corresponding true labels and a quantity parameter (k) are specified.

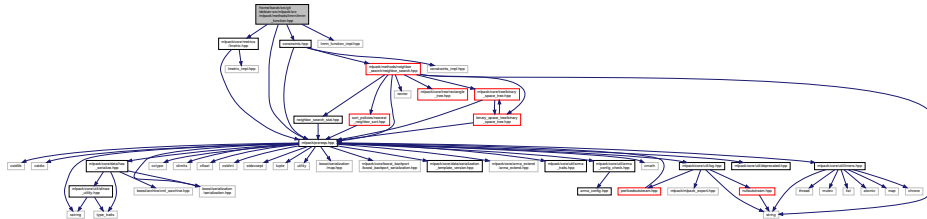
Namespaces

- **mlpack**
.hpp
- **mlpack::lmnn**
Large Margin Nearest Neighbor.

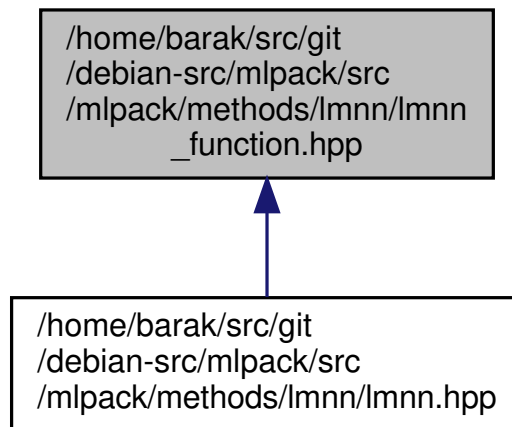
40.601 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lmnn/lmnn_function.hpp

File Reference

Include dependency graph for lmnn_function.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **LMNNFunction**< **MetricType** >
The Large Margin Nearest Neighbors function.

Namespaces

- **mlpack**
.hpp
- **mlpack::lmnn**
Large Margin Nearest Neighbor.

40.603.1 Detailed Description

Author

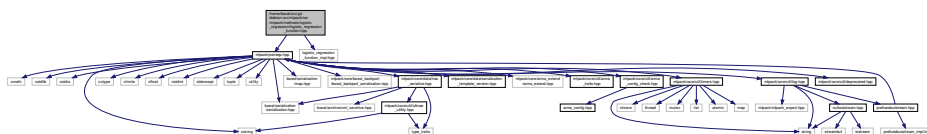
Sumedh Ghaisas
Arun Reddy

The LogisticRegression class, which implements logistic regression. This implements supports L2-regularization.

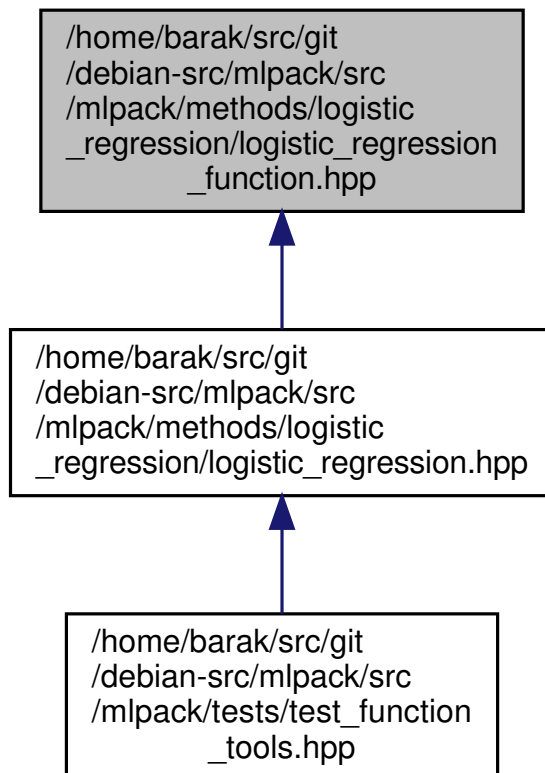
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.↵org/licenses/BSD-3-Clause> for more information.

40.604 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/logistic_regression/logistic_↵ _regression_function.hpp File Reference

Include dependency graph for logistic_regression_function.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::neighbor**

Functions

- **BOOST_TEMPLATE_CLASS_VERSION** (template< typename SortPolicy >, **mlpack::neighbor::LSHSearch**< SortPolicy >, 1)
 Set the serialization version of the LSHSearch class.

40.605.1 Detailed Description

Author

Parikshit Ram

Defines the LSHSearch class, which performs an approximate nearest neighbor search for a queries in a query set over a given dataset using Locality-sensitive hashing with 2-stable distributions.

The details of this method can be found in the following paper:

{datar2004locality, title={Locality-sensitive hashing scheme based on p-stable distributions}, author={Datar, M. and Immorlica, N. and Indyk, P. and Mirrokni, V.S.}, booktitle={Proceedings of the 12th Annual Symposium on Computational Geometry}, pages={253–262}, year={2004}, organization={ACM} }

Additionally, the class implements Multiprobe LSH, which improves approximation results during the search for approximate nearest neighbors. The Multiprobe LSH algorithm was presented in the paper:

{Lv2007multiprobe, title={Multi-probe LSH: efficient indexing for high-dimensional similarity search}, author={Lv, Qin and Josephson, William and Wang, Zhe and Charikar, Moses and Li, Kai}, booktitle={Proceedings of the 33rd international conference on Very large data bases}, year={2007}, pages={950–961} }

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.605.2 Function Documentation

40.605.2.1 BOOST_TEMPLATE_CLASS_VERSION()

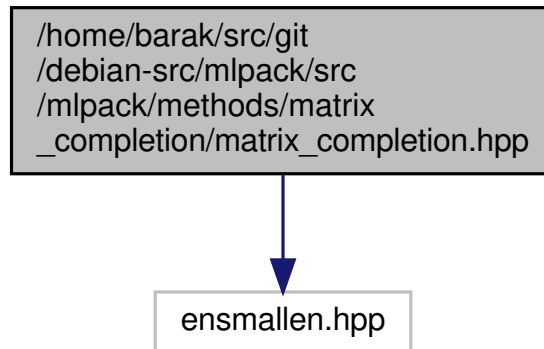
```
BOOST_TEMPLATE_CLASS_VERSION (
    template< typename SortPolicy > ,
    mlpack::neighbor::LSHSearch< SortPolicy > ,
    1 )
```

Set the serialization version of the LSHSearch class.

Referenced by LSHSearch< SortPolicy >::Projections().

40.606 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/matrix_completion/matrix_completion.hpp File Reference

Include dependency graph for matrix_completion.hpp:



Classes

- class **MatrixCompletion**

This class implements the popular nuclear norm minimization heuristic for matrix completion problems.

Namespaces

- **mlpack**
.hpp
- **mlpack::matrix_completion**

40.606.1 Detailed Description

Author

Stephen Tu

A thin wrapper around nuclear norm minimization to solve low rank matrix completion problems.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

[illegible]

- class **MeanShift**< **UseKernel**, **KernelType**, **MatType** >
This class implements mean shift clustering.

- **mlpack**
.hpp
- **mlpack::meanshift**
Mean shift clustering.

Shangtong Zhang

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

The diagram illustrates the hierarchical structure of the 'Introduction' section. The root node is 'Introduction' (1.0). It branches into four main categories: 'Background' (1.1), 'Methods' (1.2), 'Results' (1.3), and 'Discussion' (1.4). Each of these categories further branches into sub-sections, which then branch into more detailed sub-sections, creating a dense tree structure. The diagram uses blue lines and boxes to represent the hierarchy.

Namespaces

- **mlpack**
 .hpp
- **mlpack::nca**
 Neighborhood Components Analysis.

40.609.1 Detailed Description

Author

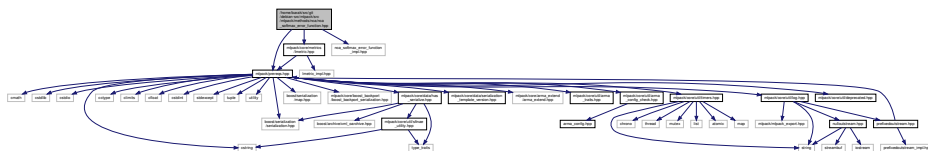
Ryan Curtin

Declaration of NCA class (Neighborhood Components Analysis).

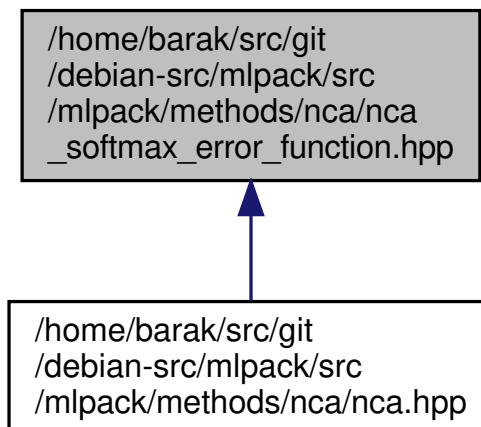
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.610 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nca/nca_softmax_error_function.hpp File Reference

Include dependency graph for nca_softmax_error_function.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **NeighborSearch**< **SortPolicy**, **MetricType**, **MatType**, **TreeType**, **DualTreeTraversalType**, **SingleTree**↔
TraversalType >

*The **NeighborSearch** (p. 1627) class is a template class for performing distance-based neighbor searches.*

- class **TrainVisitor**< **SortPolicy** >

***TrainVisitor** (p. 1707) sets the reference set to a new reference set on the given **NSType**.*

Namespaces

- **mlpack**
.hpp
- **mlpack::neighbor**

Enumerations

- enum **NeighborSearchMode** {
 NAIVE_MODE,
 SINGLE_TREE_MODE,
 DUAL_TREE_MODE,
 GREEDY_SINGLE_TREE_MODE }

NeighborSearchMode represents the different neighbor search modes available.

40.611.1 Detailed Description

Author

Ryan Curtin

Defines the NeighborSearch class, which performs an abstract nearest-neighbor-like query on two datasets.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.612.1 Detailed Description

Author

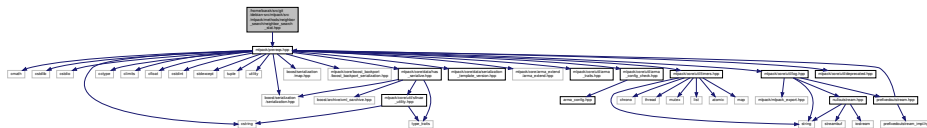
Ryan Curtin

Defines the pruning rules and base case rules necessary to perform a tree-based search (with an arbitrary tree) for the NeighborSearch class.

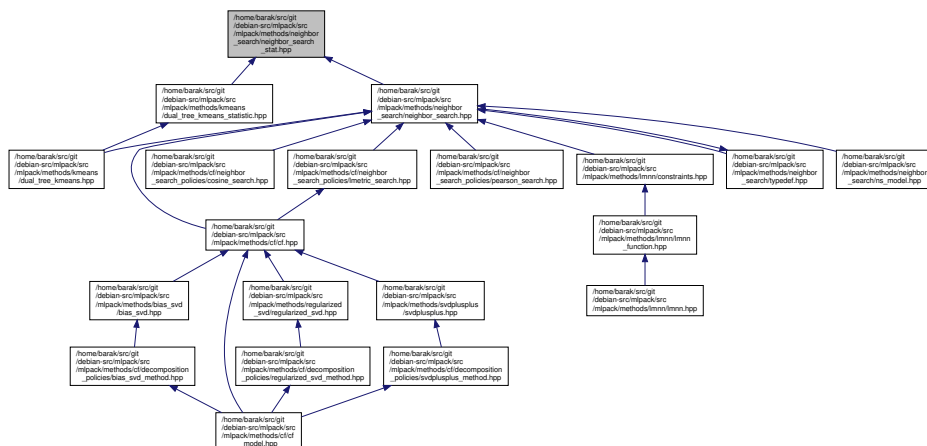
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.613 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/neighbor_↵
_search_stat.hpp File Reference

Include dependency graph for neighbor_search_stat.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **NeighborSearchStat** < **SortPolicy** >

Extra data for each node in the tree.

Namespaces

- **mlpack**
 .hpp
- **mlpack::neighbor**

40.614 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/ns_model.hpp File Reference

Include dependency graph for ns_model.hpp:



Classes

- class **BiSearchVisitor**< **SortPolicy** >
 BiSearchVisitor (p. 1591) executes a bichromatic neighbor search on the given *NSType*.
- class **DeleteVisitor**
 DeleteVisitor (p. 1595) deletes the given *NSType* instance.
- class **EpsilonVisitor**
 EpsilonVisitor (p. 1600) exposes the *Epsilon* method of the given *NSType*.
- class **MonoSearchVisitor**
 MonoSearchVisitor (p. 1618) executes a monochromatic neighbor search on the given *NSType*.
- class **NSModel**< **SortPolicy** >
 The *NSModel* (p. 1657) class provides an easy way to serialize a model, abstracts away the different types of trees, and also reflects the *NeighborSearch* (p. 1627) API.
- class **ReferenceSetVisitor**
 ReferenceSetVisitor (p. 1701) exposes the *referenceSet* of the given *NSType*.
- class **SearchModeVisitor**
 SearchModeVisitor (p. 1703) exposes the *SearchMode()* method of the given *NSType*.
- class **TrainVisitor**< **SortPolicy** >
 TrainVisitor (p. 1707) sets the reference set to a new reference set on the given *NSType*.

Namespaces

- **mlpack**
 .hpp
- **mlpack::neighbor**

Typedefs

- `template<typename SortPolicy , template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType> using NSType = NeighborSearch< SortPolicy, metric::EuclideanDistance, arma::mat, TreeType, TreeType< metric::EuclideanDistance, NeighborSearchStat< SortPolicy >, arma::mat >::template DualTreeTraverser >`
Alias template for euclidean neighbor search.

Functions

- **BOOST_TEMPLATE_CLASS_VERSION** (template< typename SortPolicy >, `mlpack::neighbor::NSModel< SortPolicy >`, 1)
Set the serialization version of the NSModel class.

40.614.1 Detailed Description

Author

Ryan Curtin

This is a model for nearest or furthest neighbor search. It is useful in that it provides an easy way to serialize a model, abstracts away the different types of trees, and also reflects the NeighborSearch API and automatically directs to the right tree type.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.614.2 Function Documentation

40.614.2.1 BOOST_TEMPLATE_CLASS_VERSION()

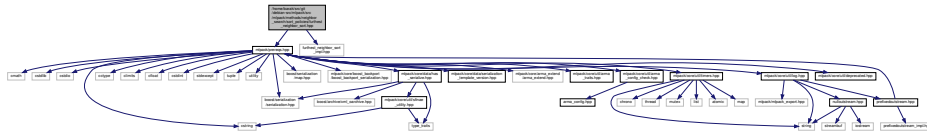
```
BOOST_TEMPLATE_CLASS_VERSION (
    template< typename SortPolicy > ,
    mlpack::neighbor::NSModel< SortPolicy > ,
    1 )
```

Set the serialization version of the NSModel class.

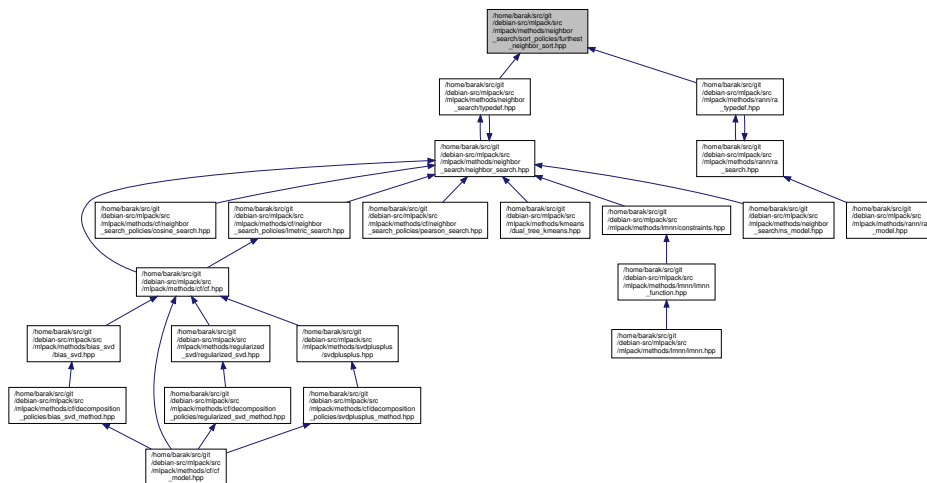
Referenced by `NSModel< SortPolicy >::RandomBasis()`.

40.615 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/neighbor_search/sort_
policies/furthest_neighbor_sort.hpp File Reference

Include dependency graph for furthest_neighbor_sort.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **FurthestNS**

*This class implements the necessary methods for the SortPolicy template parameter of the **NeighborSearch** (p. 1627) class.*

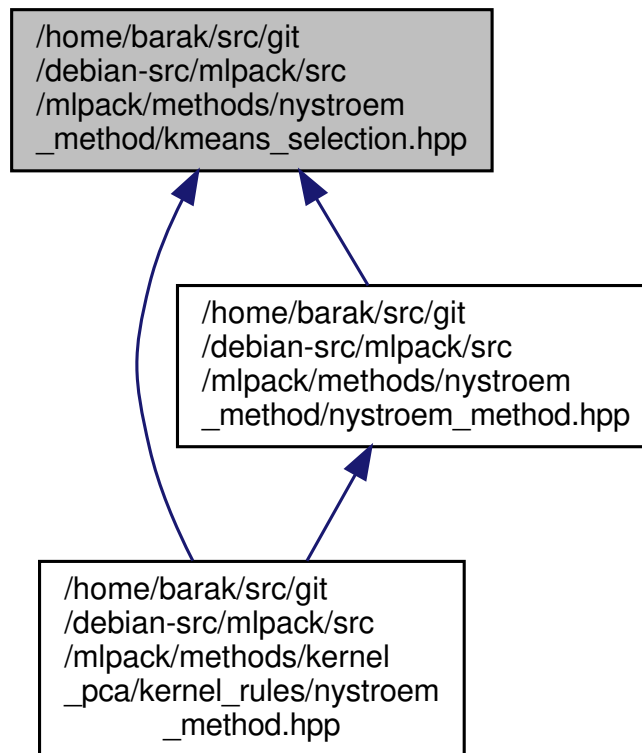
Namespaces

- **mlpack**
 .hpp
- **mlpack::neighbor**

Typedefs

- using **FurthestNeighborSort** = FurthestNS

This graph shows which files directly or indirectly include this file:



Classes

- class **KMeansSelection**< **ClusteringType**, **maxIterations** >
Implementation of the kmeans sampling scheme.

Namespaces

- **mlpack**
.hpp
- **mlpack::kernel**
Kernel functions.

40.618.1 Detailed Description

Author

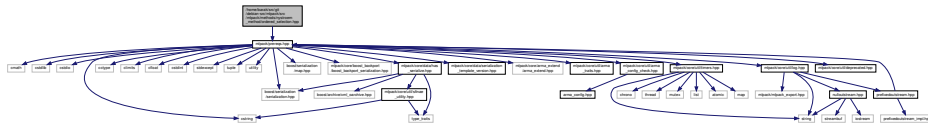
Marcus Edel

Use the centroids of the K-Means clustering method for use in the Nystroem method of kernel matrix approximation.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

_selection.hpp File Reference

Include dependency graph for ordered_selection.hpp:



Classes

- class **OrderedSelection**

Namespaces

- **mlpack**
.hpp
- **mlpack::kernel**
Kernel functions.

40.619.1 Detailed Description

Author

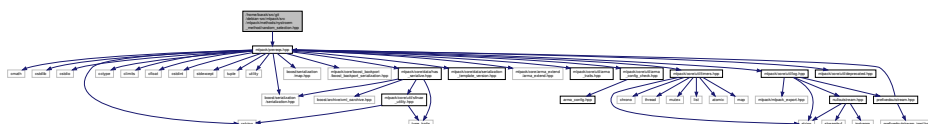
Ryan Curtin

Select the first points of the dataset for use in the Nystroem method of kernel matrix approximation. This is mostly for testing, but might have other uses.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

_selection.hpp File Reference

Include dependency graph for random_selection.hpp:



Classes

- class **RandomSelection**

Namespaces

- **mlpack**
.hpp
- **mlpack::kernel**
Kernel functions.

40.620.1 Detailed Description

Author

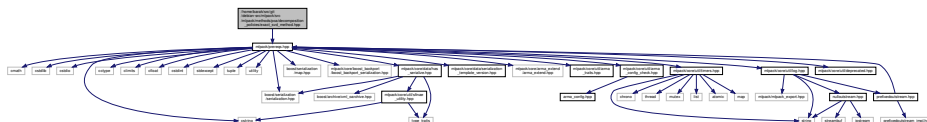
Ryan Curtin

Randomly select some points (with replacement) to use for the Nystroem method. Replacement is suboptimal, but for rank \ll number of points, this is unlikely.

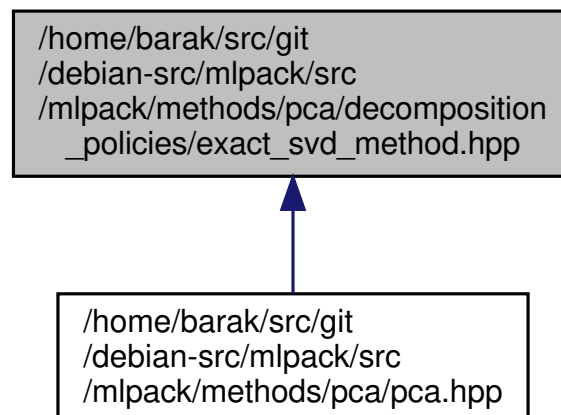
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.621 `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/pca/decomposition_policies/exact_svd_method.hpp` File Reference

Include dependency graph for `exact_svd_method.hpp`:

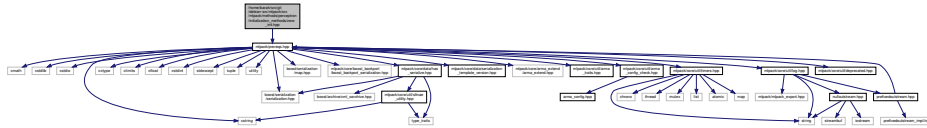


This graph shows which files directly or indirectly include this file:

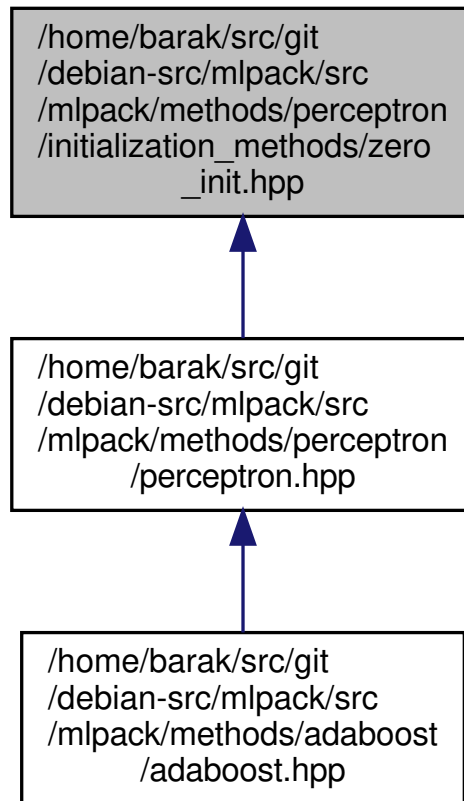


40.625 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/initialization_methods/zero_init.hpp File Reference [↔](#)

Include dependency graph for zero_init.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **ZeroInitialization**

This class is used to initialize the matrix weightVectors to zero.

Namespaces

- **mlpack**
 .hpp
- **mlpack::perceptron**

40.625.1 Detailed Description

Author

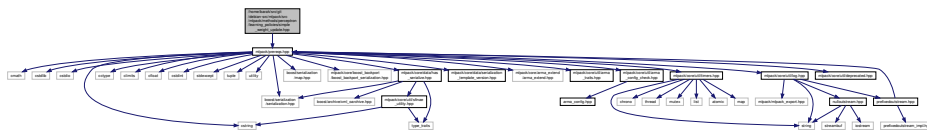
Udit Saxena

Implementation of ZeroInitialization policy for perceptrons.

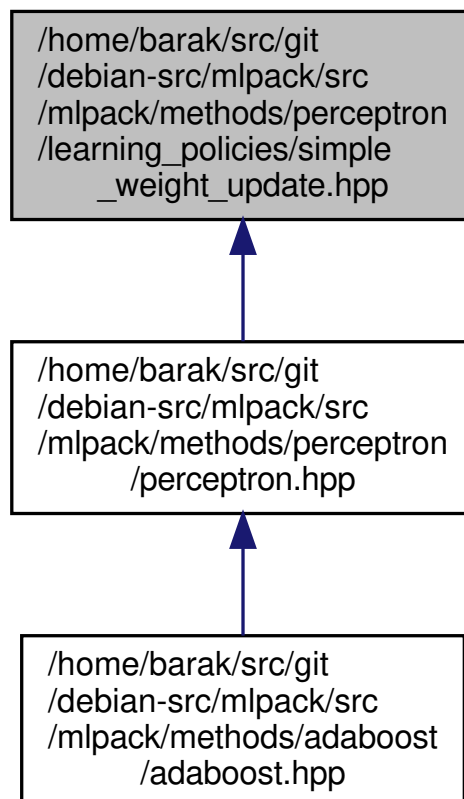
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.626 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/learning_policies/simple_weight_update.hpp File Reference

Include dependency graph for simple_weight_update.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **SimpleWeightUpdate**

Namespaces

- **mlpack**
 .hpp
- **mlpack::perceptron**

40.626.1 Detailed Description

Author

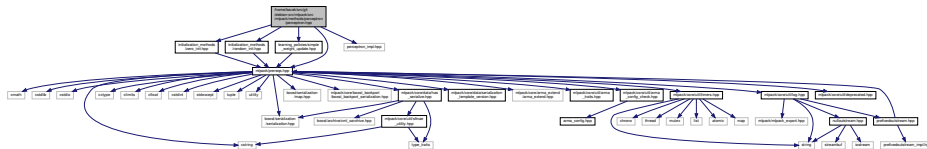
Udit Saxena

Simple weight update rule for the perceptron.

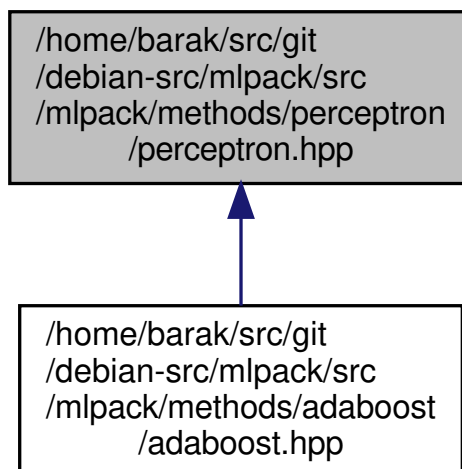
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.627 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/perceptron/perceptron.hpp File Reference

Include dependency graph for perceptron.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **Perceptron**< **LearnPolicy**, **WeightInitializationPolicy**, **MatType** >
This class implements a simple perceptron (i.e., a single layer neural network).

Namespaces

- mlpack**
 .hpp
- mlpack::perceptron**

40.627.1 Detailed Description

Author

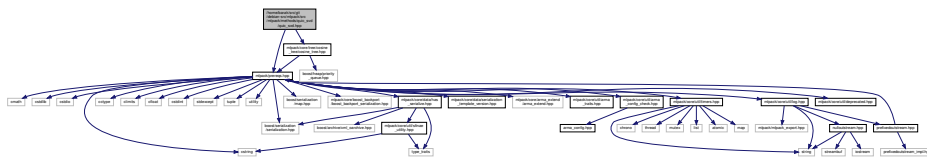
Udit Saxena

Definition of Perceptron class.

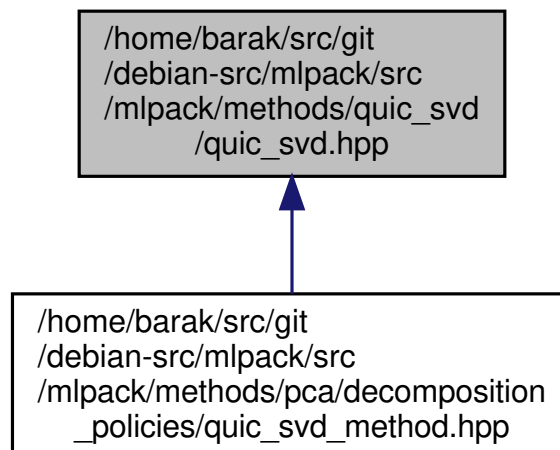
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.628 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/quic_svd/quic_svd.hpp File Reference

Include dependency graph for quic_svd.hpp:



This graph shows which files directly or indirectly include this file:



Functions

- void **WhitenFeatureMajorMatrix** (const arma::mat &matX, arma::mat &matXWhitened, arma::mat &matXWhitening)

40.629.1 Detailed Description

Author

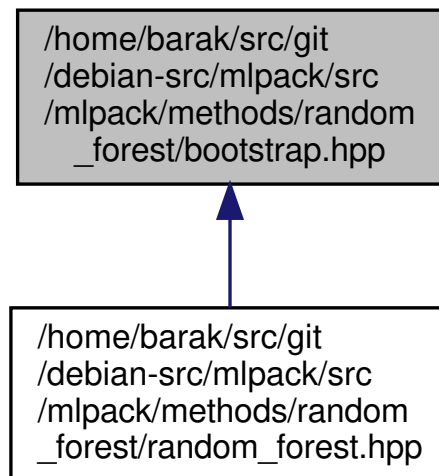
Nishant Mehta

Declaration of Radical class (RADICAL is Robust, Accurate, Direct ICA aLgorithm).

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.630 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/random_forest/bootstrap.hpp File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::tree**
 Trees and tree-building procedures.

Functions

- `template<bool UseWeights, typename MatType , typename LabelsType , typename WeightsType >`
`void Bootstrap (const MatType &dataset, const LabelsType &labels, const WeightsType &weights, MatType`
`&bootstrapDataset, LabelsType &bootstrapLabels, WeightsType &bootstrapWeights)`

Given a dataset, create another dataset via bootstrap sampling, with labels.

40.630.1 Detailed Description

Author

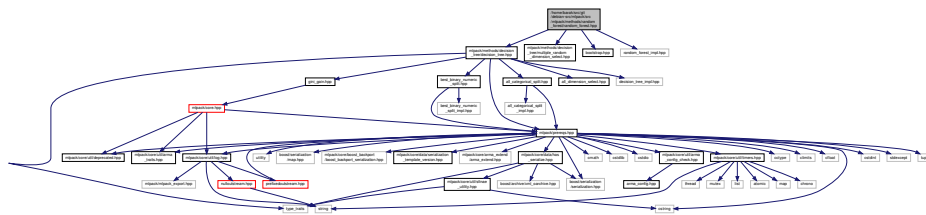
Ryan Curtin

Implementation of the **Bootstrap()** (p. 463) function, which creates a bootstrapped dataset from the given input dataset.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.631 `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/random_forest/random_↵` `_forest.hpp` File Reference

Include dependency graph for `random_forest.hpp`:



Classes

- class **RandomForest**< **FitnessFunction**, **DimensionSelectionType**, **NumericSplitType**, **CategoricalSplitType**, **ElemType** >

Namespaces

- **mlpack**
`.hpp`
- **mlpack::tree**
Trees and tree-building procedures.

40.631.1 Detailed Description

Author

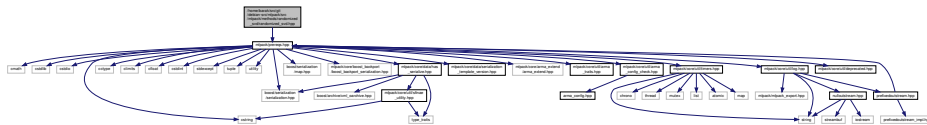
Ryan Curtin

Definition of the RandomForest class.

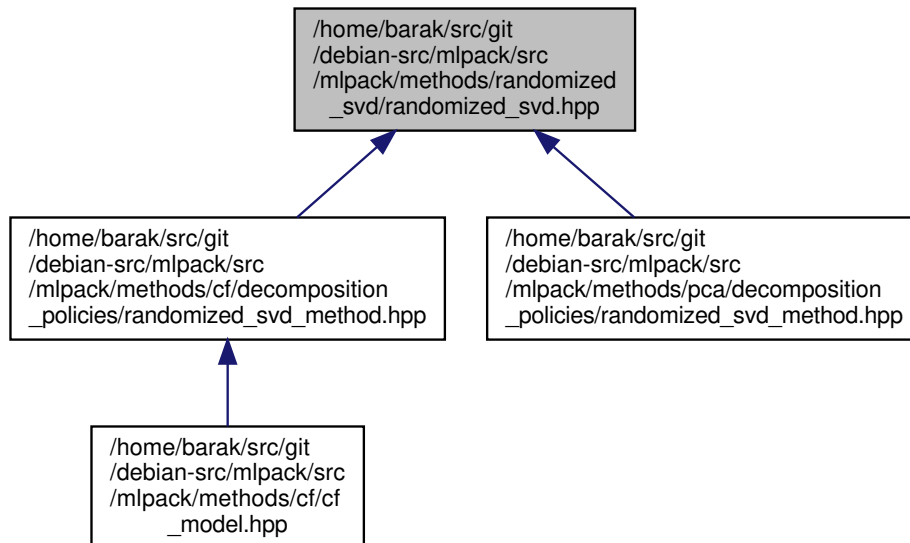
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.632 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/randomized_svd/randomized_svd.hpp File Reference

Include dependency graph for randomized_svd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RandomizedSVD**

Randomized SVD is a matrix factorization that is based on randomized matrix approximation techniques, developed in in "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions".

Namespaces

- **mlpack**
 - .hpp*
- **mlpack::svd**

40.632.1 Detailed Description

Author

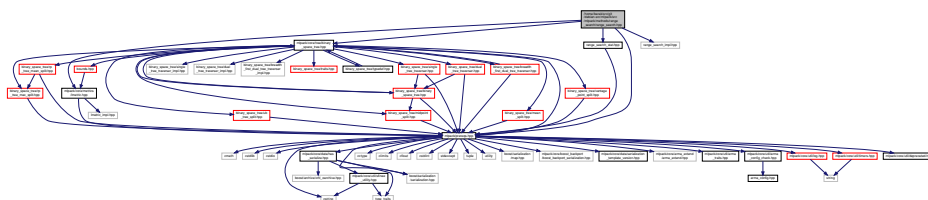
Marcus Edel

An implementation of the randomized SVD method.

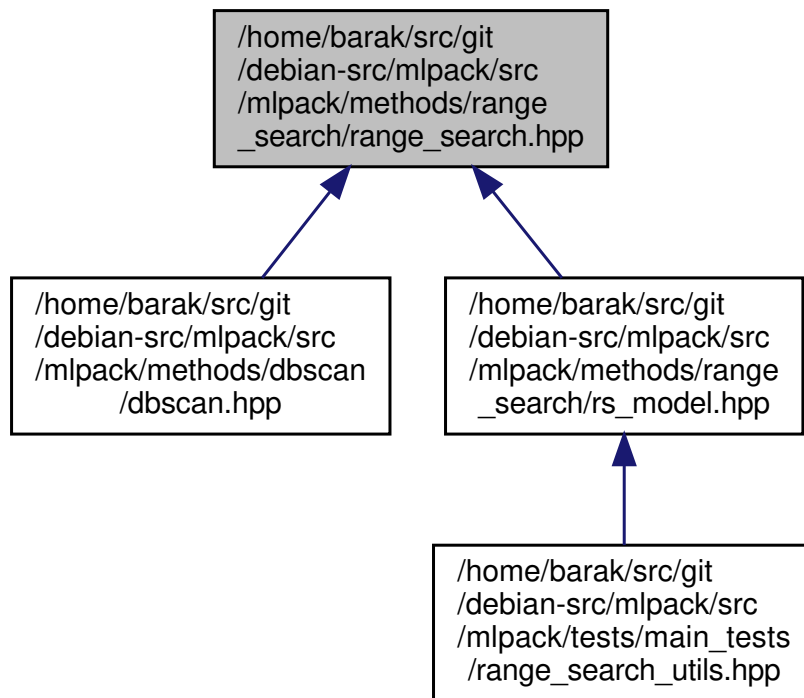
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.633 `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/range_search.hpp` File Reference

Include dependency graph for `range_search.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **RangeSearch**< **MetricType**, **MatType**, **TreeType** >

The *RangeSearch* (p. 1754) class is a template class for performing range searches.

Namespaces

- **mlpack**
.hpp
- **mlpack::range**

Range-search routines.

40.633.1 Detailed Description

Author

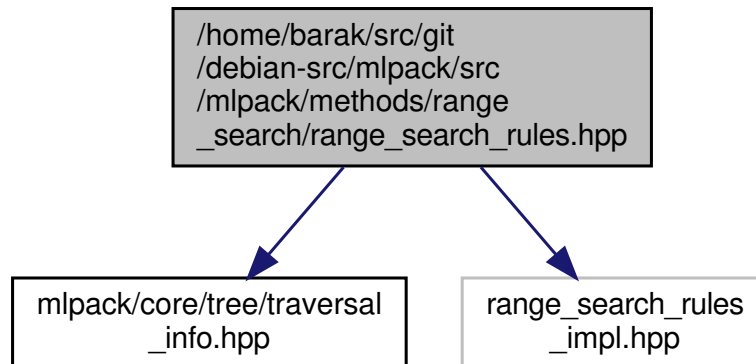
Ryan Curtin

Defines the RangeSearch class, which performs a generalized range search on points.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.634 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/range_search_rules.hpp File Reference

Include dependency graph for range_search_rules.hpp:



Classes

- class **RangeSearchRules**< **MetricType**, **TreeType** >

The **RangeSearchRules** (p. 1764) class is a template helper class used by **RangeSearch** (p. 1754) class when performing range searches.

Namespaces

- **mlpack**
.hpp
- **mlpack::range**

Range-search routines.

40.634.1 Detailed Description

Author

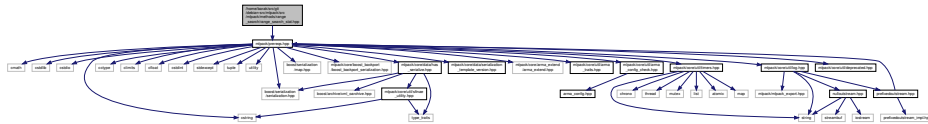
Ryan Curtin

Rules for range search, so that it can be done with arbitrary tree types.

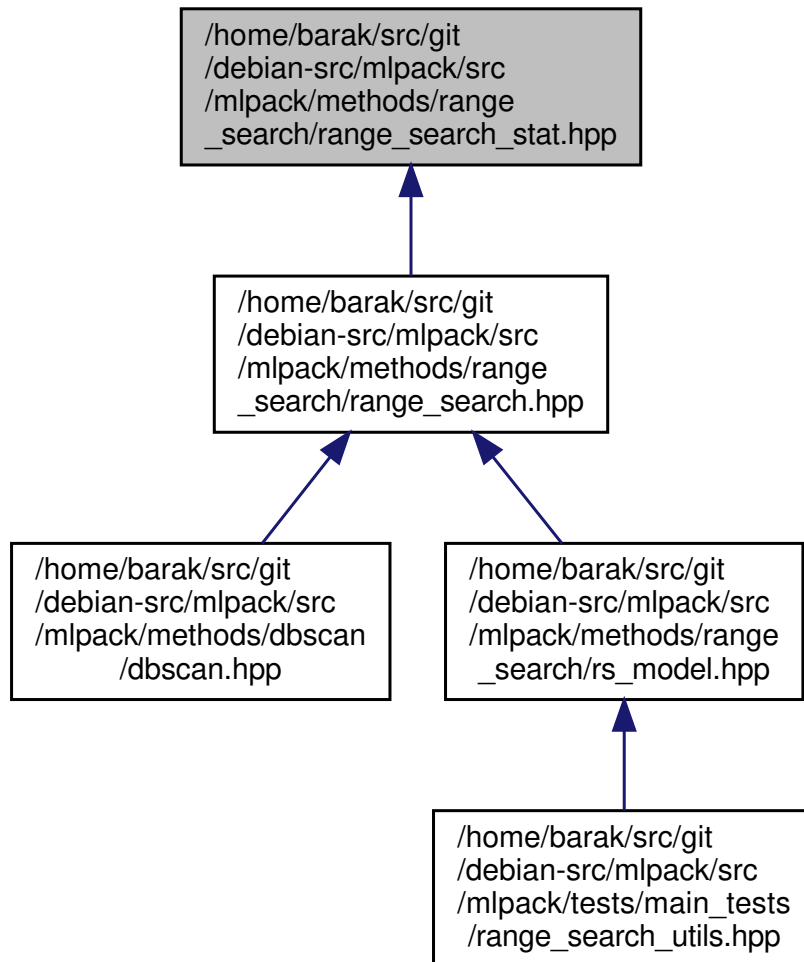
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.635 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/range_search_stat.hpp File Reference

Include dependency graph for range_search_stat.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RangeSearchStat**

Statistic class for **RangeSearch** (p. 1754), to be set to the *StatisticType* of the tree type that range search is being performed with.

Namespaces

- **mlpack**
 .hpp
- **mlpack::range**
 Range-search routines.

40.635.1 Detailed Description

Author

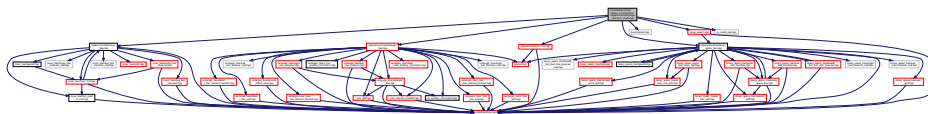
Ryan Curtin

Statistic class for RangeSearch, which just holds the last visited node and the corresponding base case result.

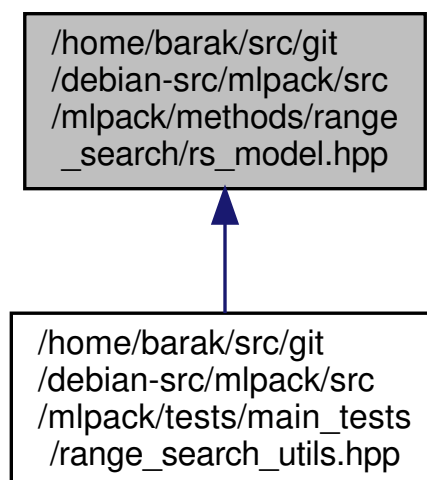
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.636 `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/range_search/rs_model.hpp` File Reference

Include dependency graph for `rs_model.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **BiSearchVisitor**
BiSearchVisitor (p. 1748) executes a bichromatic range search on the given *RSType*.
- class **DeleteVisitor**
DeleteVisitor (p. 1751) deletes the given *RSType* instance.
- class **MonoSearchVisitor**
MonoSearchVisitor (p. 1752) executes a monochromatic range search on the given *RSType*.
- class **NaiveVisitor**
NaiveVisitor (p. 1753) exposes the *Naive()* method of the given *RSType*.
- class **ReferenceSetVisitor**
ReferenceSetVisitor (p. 1771) exposes the *referenceSet* of the given *RSType*.
- class **RSType**
- class **SingleNodeVisitor**
SingleNodeVisitor (p. 1779) exposes the *SingleNode()* method of the given *RSType*.
- class **TrainVisitor**
TrainVisitor (p. 1780) sets the reference set to a new reference set on the given *RSType*.

Namespaces

- **mlpack**
.hpp
- **mlpack::range**
Range-search routines.

Typedefs

- `template<template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType>
using RSType = RangeSearch< metric::EuclideanDistance, arma::mat, TreeType >`
Alias template for Range Search.

40.636.1 Detailed Description

Author

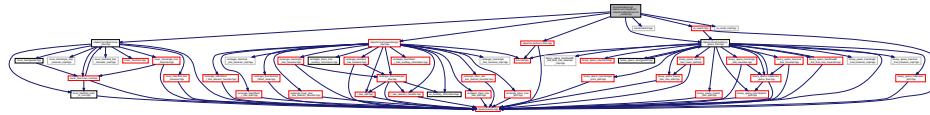
Ryan Curtin

This is a model for range search. It is useful in that it provides an easy way to serialize a model, abstracts away the different types of trees, and also reflects the RangeSearch API and automatically directs to the right tree types.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.637 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ra_model.hpp File Reference

Include dependency graph for ra_model.hpp:



Classes

- class **AlphaVisitor**
Exposes the Alpha() method of the given RType.
- class **BiSearchVisitor**< **SortPolicy** >
***BiSearchVisitor** (p. 1591) executes a bichromatic neighbor search on the given NType.*
- class **DeleteVisitor**
***DeleteVisitor** (p. 1595) deletes the given NType instance.*
- class **FirstLeafExactVisitor**
Exposes the FirstLeafExact() method of the given RType.
- class **MonoSearchVisitor**
***MonoSearchVisitor** (p. 1618) executes a monochromatic neighbor search on the given NType.*
- class **NaiveVisitor**
***NaiveVisitor** (p. 1620) exposes the Naive() method of the given RType.*
- class **RAModel**< **SortPolicy** >
*The **RAModel** (p. 1669) class provides an abstraction for the **RASearch** (p. 1681) class, abstracting away the TreeType parameter and allowing it to be specified at runtime in this class.*
- class **ReferenceSetVisitor**
***ReferenceSetVisitor** (p. 1701) exposes the referenceSet of the given NType.*
- class **SampleAtLeavesVisitor**
Exposes the SampleAtLeaves() method of the given RType.
- class **SingleModeVisitor**
Exposes the SingleMode() method of the given RType.
- class **SingleSampleLimitVisitor**
Exposes the SingleSampleLimit() method of the given RType.
- class **TauVisitor**
Exposes the Tau() method of the given RType.
- class **TrainVisitor**< **SortPolicy** >
***TrainVisitor** (p. 1707) sets the reference set to a new reference set on the given NType.*

Namespaces

- **mlpack**
 .hpp
- **mlpack::neighbor**

Typedefs

- template<typename SortPolicy , template< typename TreeMetricType, typename TreeStatType, typename TreeMatType > class TreeType> using **RAType** = RASearch< SortPolicy, metric::EuclideanDistance, arma::mat, TreeType >

Alias template for **RASearch** (p. 1681).

40.637.1 Detailed Description

Author

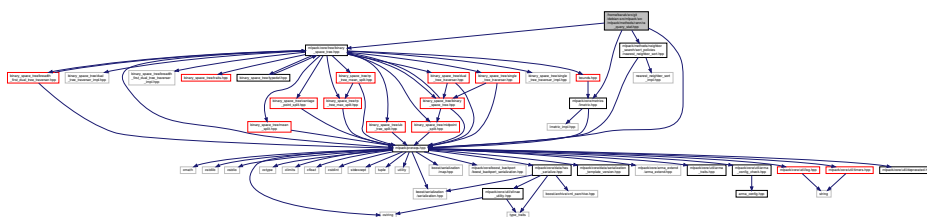
Ryan Curtin

This is a model for rank-approximate nearest neighbor search. It provides an easy way to serialize a rank-approximate neighbor search model by abstracting the types of trees and reflecting the RASearch API.

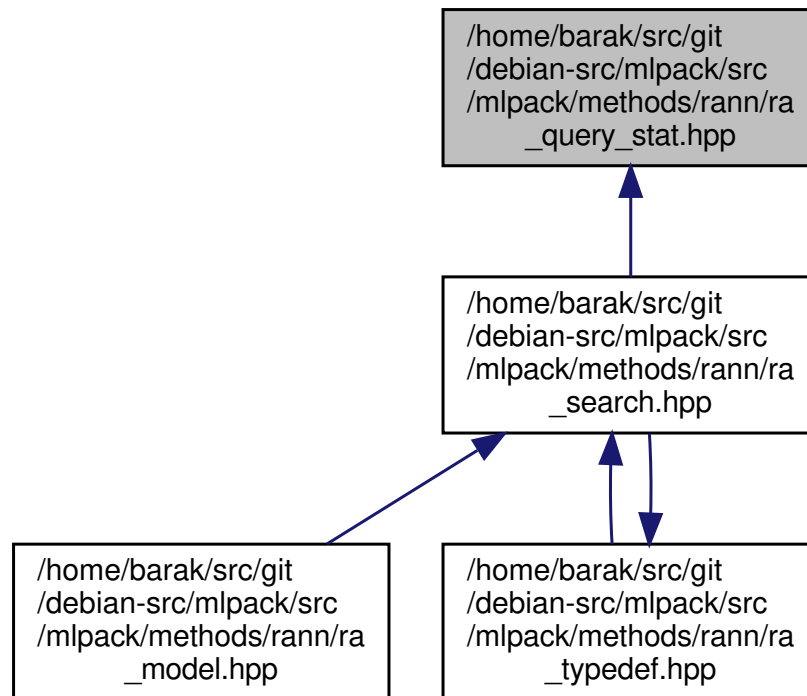
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.638 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ra_query_stat.hpp File Reference

Include dependency graph for ra_query_stat.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RAQueryStat**< **SortPolicy** >
Extra data for each node in the tree.

Namespaces

- **mlpack**
.hpp
- **mlpack::neighbor**

40.638.1 Detailed Description

Author

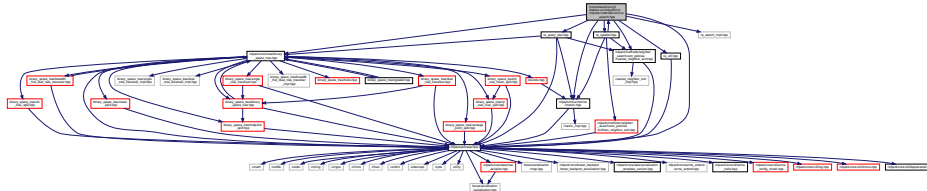
Parikshit Ram

Defines the RAQueryStat class, which is the statistic used for rank-approximate nearest neighbor search (RASearch).

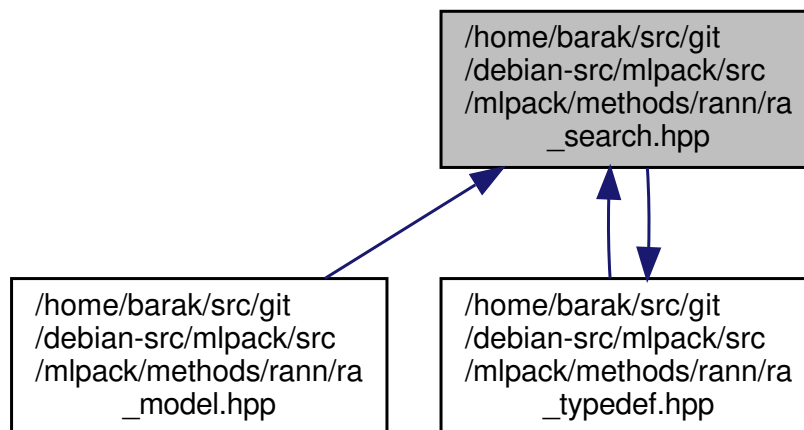
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.639 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ra_search.hpp File Reference

Include dependency graph for ra_search.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RASearch**< **SortPolicy**, **MetricType**, **MatType**, **TreeType** >

The **RASearch** (p. 1681) class: This class provides a generic manner to perform rank-approximate search via random-sampling.

- class **TrainVisitor**< **SortPolicy** >

TrainVisitor (p. 1707) sets the reference set to a new reference set on the given **NSType**.

Namespaces

- **mlpack**
 .hpp
- **mlpack::neighbor**

40.639.1 Detailed Description

Author

Parikshit Ram

Defines the `RASearch` class, which performs an abstract rank-approximate nearest/farthest neighbor query on two datasets.

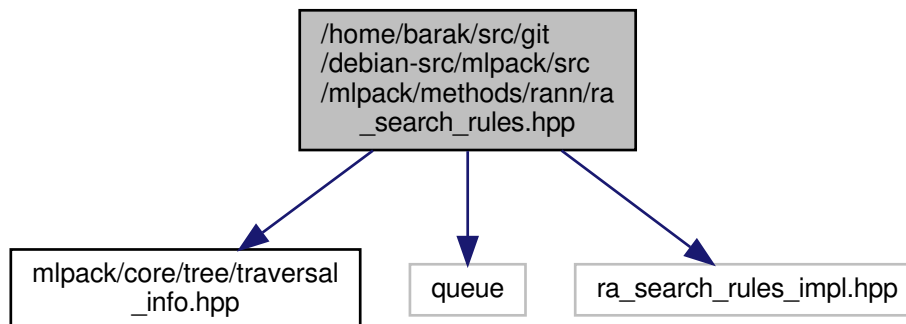
The details of this method can be found in the following paper:

{ram2009rank, title={{Rank-Approximate Nearest Neighbor Search: Retaining Meaning and Speed in High Dimensions}}, author={{Ram, P. and Lee, D. and Ouyang, H. and Gray, A. G.}}, booktitle={{Advances of Neural Information Processing Systems}}, year={2009} }

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.640 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ra_search_rules.hpp File Reference

Include dependency graph for `ra_search_rules.hpp`:



Classes

- class **`RASearchRules`**< `SortPolicy`, `MetricType`, `TreeType` >

The **`RASearchRules`** (p. 1692) class is a template helper class used by **`RASearch`** (p. 1681) class when performing rank-approximate search via random-sampling.

Namespaces

- **`mlpack`**
 `.hpp`
- **`mlpack::neighbor`**

40.640.1 Detailed Description

Author

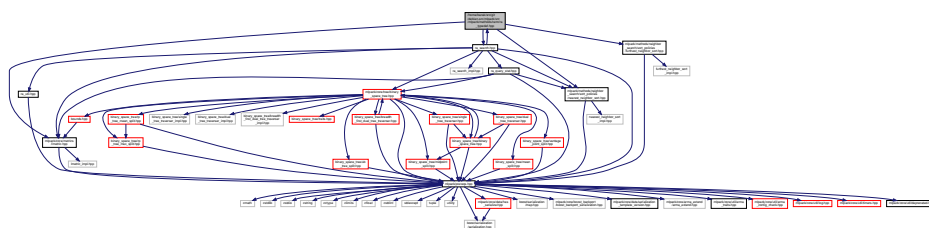
Parikshit Ram

Defines the pruning rules and base case rules necessary to perform a tree-based rank-approximate search (with an arbitrary tree) for the RASearch class.

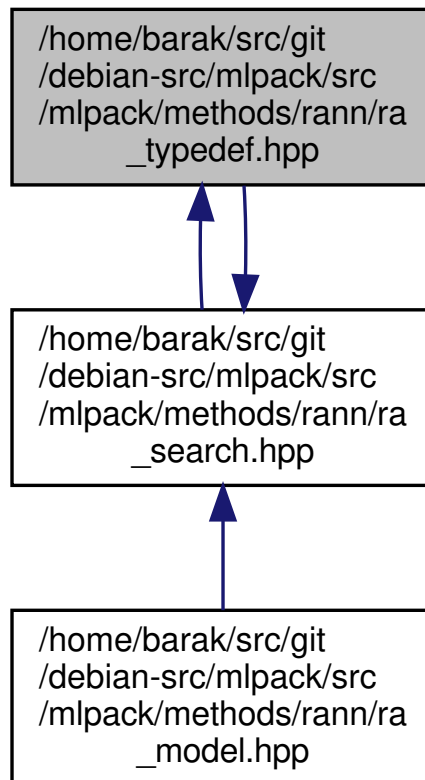
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.641 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/rann/ra_typedef.hpp File Reference

Include dependency graph for ra_typedef.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- **mlpack**
 .hpp
- **mlpack::neighbor**

Typedefs

- typedef RASearch< FurthestNeighborSort > **KRAFN**
 The KRAFN class is the k-rank-approximate-farthest-neighbors method.
- typedef RASearch **KRANN**
 The KRANN class is the k-rank-approximate-nearest-neighbors method.

40.641.1 Detailed Description

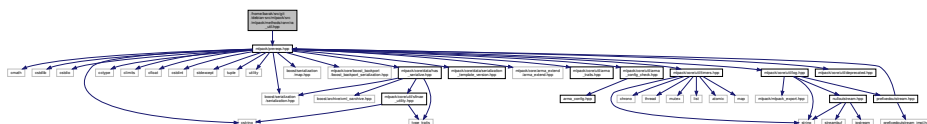
Author

Parikshit Ram

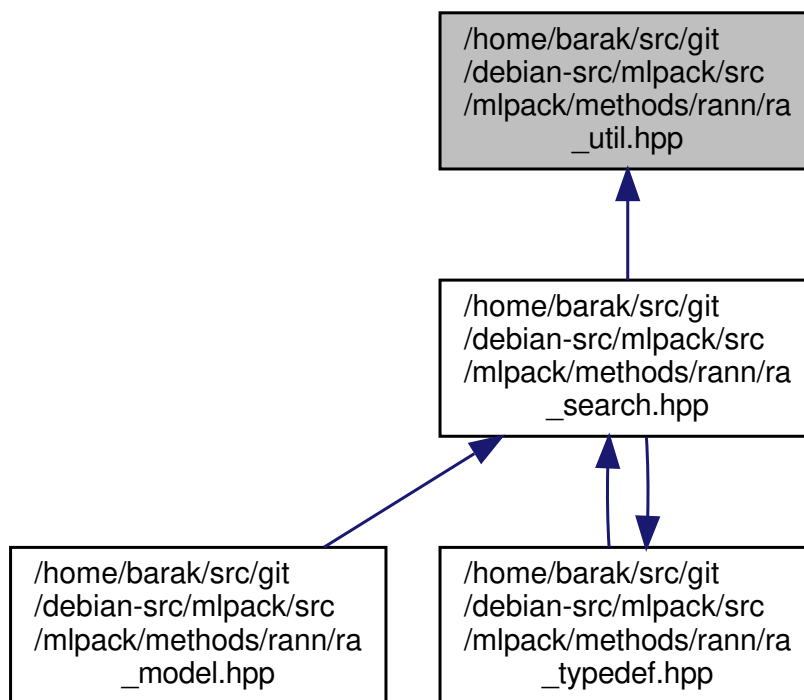
Simple typedefs describing template instantiations of the RASearch class which are commonly used.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Include dependency graph for ra_util.hpp:



This graph shows which files directly or indirectly include this file:



- class **RAUtil**

- **mlpack**
 .hpp
- **mlpack::neighbor**

40.642.1 Detailed Description

Author

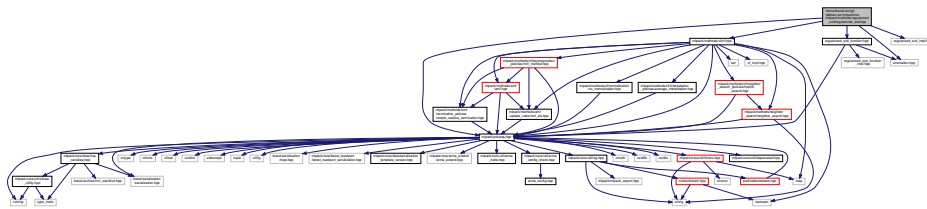
Parikshit Ram
Ryan Curtin

Utilities for rank-approximate neighbor search.

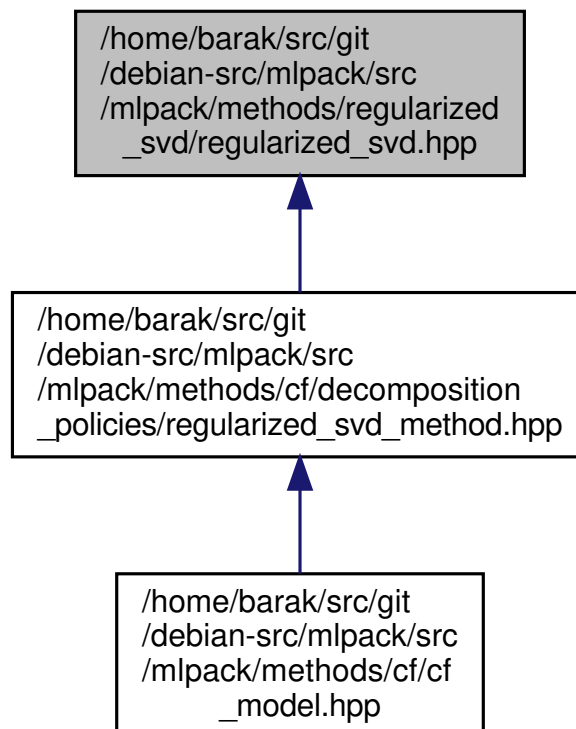
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.643 `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/regularized_svd/regularized_svd.hpp` File Reference

Include dependency graph for `regularized_svd.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **RegularizedSVD**< **OptimizerType** >

Regularized SVD is a matrix factorization technique that seeks to reduce the error on the training set, that is on the examples for which the ratings have been provided by the users.

Namespaces

- **mlpack**
.hpp
- **mlpack::svd**

40.643.1 Detailed Description

Author

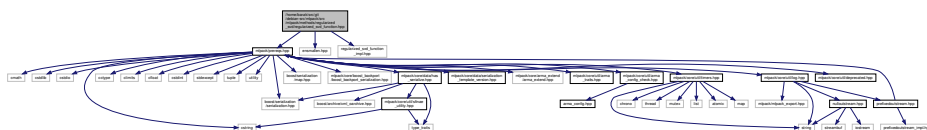
Siddharth Agrawal

An implementation of Regularized SVD.

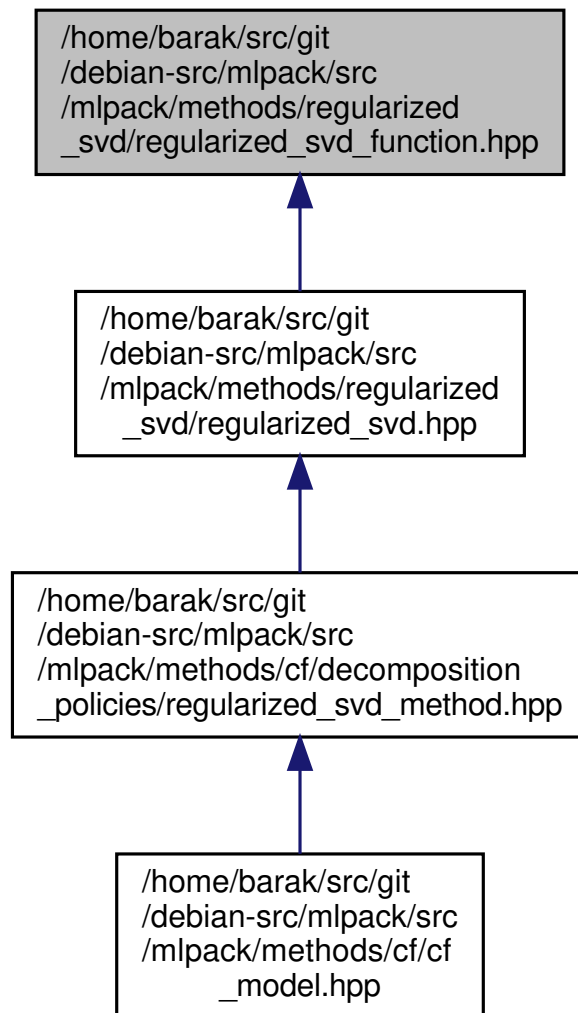
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.644 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/regularized_svd/regularized_svd_function.hpp File Reference

Include dependency graph for regularized_svd_function.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RegularizedSVDFunction**< **MatType** >

*The data is stored in a matrix of type **MatType**, so that this class can be used with both dense and sparse matrix types.*

Namespaces

- **ens**
- **mlpack**
 .hpp
- **mlpack::svd**

Typedefs

- `template<typename EnvironmentType , typename NetworkType , typename UpdaterType , typename PolicyType >`
`using NStepQLearning = AsyncLearning< NStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >, EnvironmentType, NetworkType, UpdaterType, PolicyType >`
Convenient typedef for async n step q-learning.
- `template<typename EnvironmentType , typename NetworkType , typename UpdaterType , typename PolicyType >`
`using OneStepQLearning = AsyncLearning< OneStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >, EnvironmentType, NetworkType, UpdaterType, PolicyType >`
Convenient typedef for async one step q-learning.
- `template<typename EnvironmentType , typename NetworkType , typename UpdaterType , typename PolicyType >`
`using OneStepSarsa = AsyncLearning< OneStepSarsaWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >, EnvironmentType, NetworkType, UpdaterType, PolicyType >`
Convenient typedef for async one step Sarsa.

40.645.1 Detailed Description

Author

Shangtong Zhang

This file is the definition of AsyncLearning class, which is wrapper for various asynchronous learning algorithms.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.646 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/environment/acrobot.hpp File Reference

Include dependency graph for acrobot.hpp:

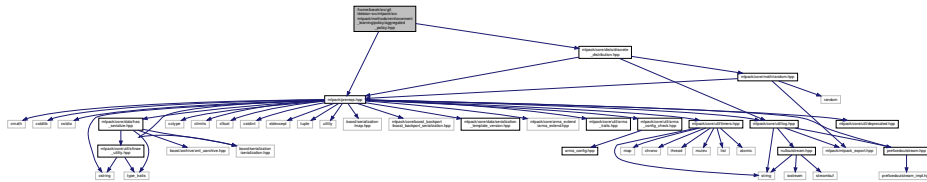


Classes

- class **Acrobot**
*Implementation of **Acrobot** (p. 1825) game.*
- class **Acrobot::State**

40.652 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/policy/aggregated_policy.hpp File Reference

Include dependency graph for aggregated_policy.hpp:



Classes

- class **AggregatedPolicy**< **PolicyType** >

Namespaces

- **mlpack**
 .hpp
- **mlpack::rl**

40.652.1 Detailed Description

Author

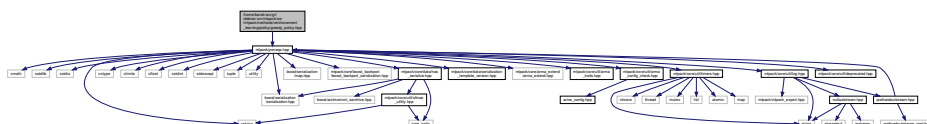
Shangtong Zhang

This file is the implementation of AggregatedPolicy class. An aggregated policy will randomly select a child policy under a given distribution at each time step.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.653 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/policy/greedy_policy.hpp File Reference

Include dependency graph for greedy_policy.hpp:



Classes

- class **GreedyPolicy**< **EnvironmentType** >
Implementation for epsilon greedy policy.

Namespaces

- mlpack**
.hpp
- mlpack::rl**

40.653.1 Detailed Description

Author

Shangtong Zhang
Abhinav Sagar

This file is an implementation of epsilon greedy policy.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.654 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/q_learning.hpp File Reference

Include dependency graph for q_learning.hpp:



Classes

- class **QLearning**< **EnvironmentType**, **NetworkType**, **UpdaterType**, **PolicyType**, **ReplayType** >
Implementation of various Q-Learning algorithms, such as DQN, double DQN.

Namespaces

- mlpack**
.hpp
- mlpack::rl**

40.654.1 Detailed Description

Author

Shangtong Zhang

This file is the definition of QLearning class, which implements Q-Learning algorithms.

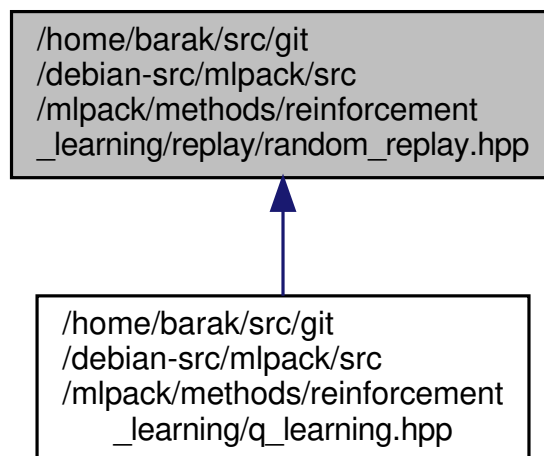
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.655 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/replay/random_replay.hpp File Reference

Include dependency graph for random_replay.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RandomReplay**< **EnvironmentType** >
Implementation of random experience replay.

Namespaces

- **mlpack**
.hpp
- **mlpack::rl**

40.655.1 Detailed Description

Author

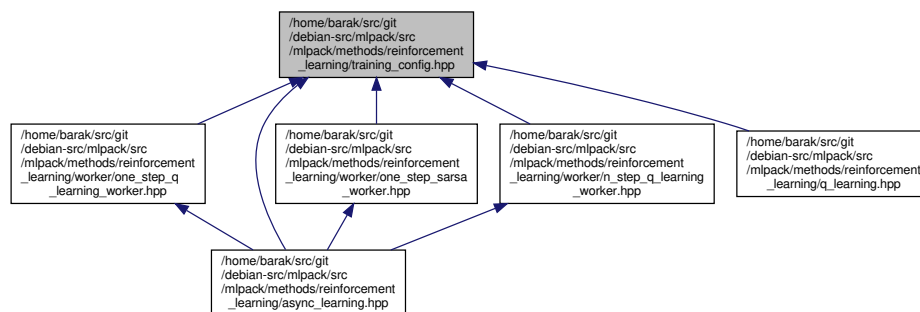
Shangtong Zhang

This file is an implementation of random experience replay.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.656 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/training_config.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **TrainingConfig**

Namespaces

- **mlpack**
.hpp
- **mlpack::rl**

40.656.1 Detailed Description

Author

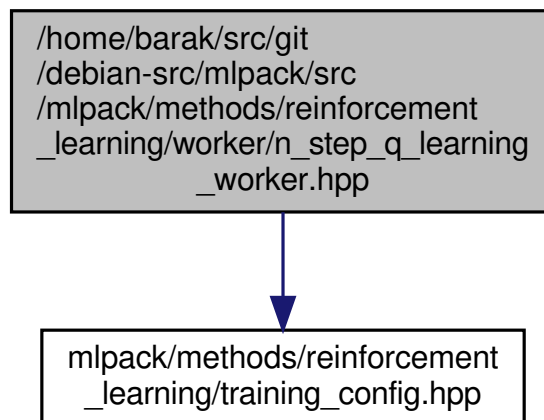
Shangtong Zhang

This file is the implementation of TrainingConfig class, which contains hyper-parameters for training RL agent.

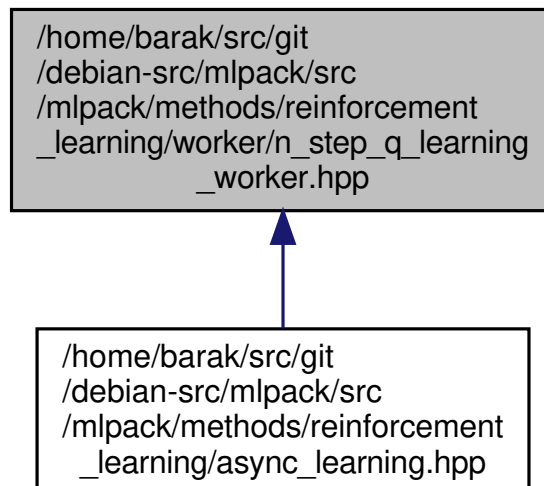
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.657 `/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/n_step_q_learning_worker.hpp` File Reference

Include dependency graph for `n_step_q_learning_worker.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class **NStepQLearningWorker**< **EnvironmentType**, **NetworkType**, **UpdaterType**, **PolicyType** >
*Forward declaration of **NStepQLearningWorker** (p. 1868).*

Namespaces

- **mlpack**
.hpp
- **mlpack::rl**

40.657.1 Detailed Description

Author

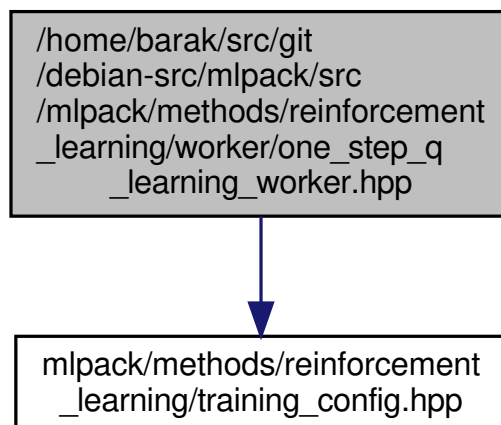
Shangtong Zhang

This file is the definition of NStepQLearningWorker class, which implements an episode for async n step Q-Learning algorithm.

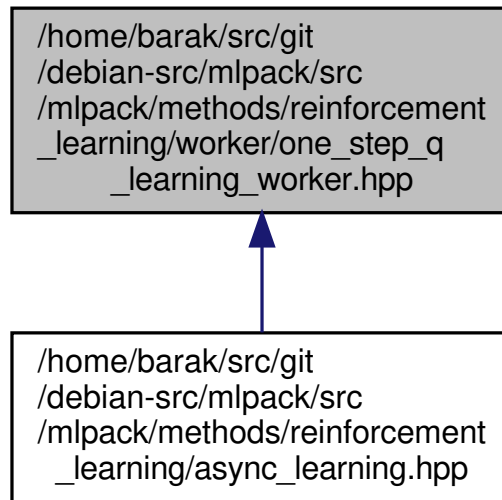
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.658 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/one_step_q_learning_worker.hpp File Reference ↩

Include dependency graph for one_step_q_learning_worker.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **OneStepQLearningWorker**< **EnvironmentType**, **NetworkType**, **UpdaterType**, **PolicyType** >
*Forward declaration of **OneStepQLearningWorker** (p. 1871).*

Namespaces

- **mlpack**
.hpp
- **mlpack::rl**

40.658.1 Detailed Description

Author

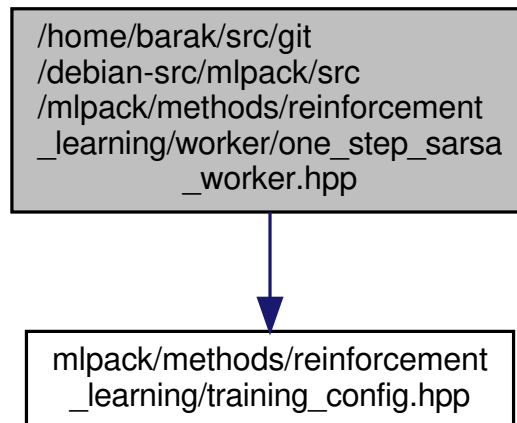
Shangtong Zhang

This file is the definition of OneStepQLearningWorker class, which implements an episode for async one step Q-
Learning algorithm.

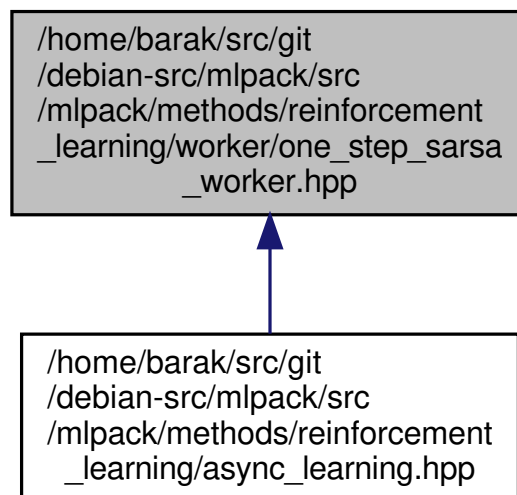
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.659 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/one_step_sarsa_worker.hpp File Reference

Include dependency graph for one_step_sarsa_worker.hpp:



This graph shows which files directly or indirectly include this file:



Classes

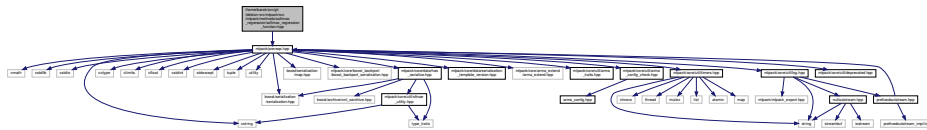
- class **OneStepSarsaWorker**< **EnvironmentType**, **NetworkType**, **UpdaterType**, **PolicyType** >
*Forward declaration of **OneStepSarsaWorker** (p. 1875).*

Namespaces

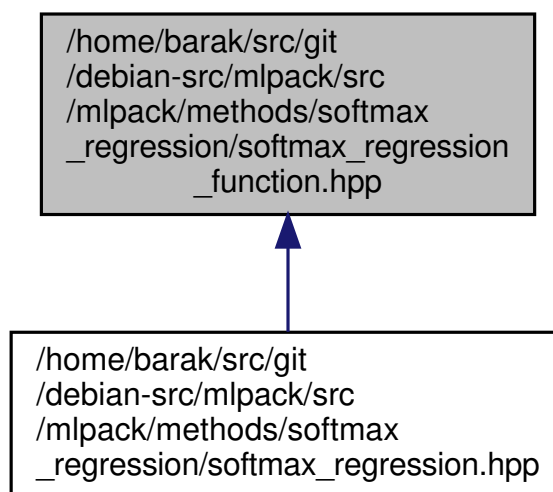
- **mlpack**
.hpp
- **mlpack::rl**

40.661 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/softmax_regression/softmax_↵ _regression_function.hpp File Reference

Include dependency graph for softmax_regression_function.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **SoftmaxRegressionFunction**

Namespaces

- **mlpack**
.hpp
- **mlpack::regression**
Regression methods.

40.661.1 Detailed Description

Author

Siddharth Agrawal

The function to be optimized for softmax regression. Any mlpack optimizer can be used.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

Classes

- class **SparseAutoencoder**

A sparse autoencoder is a neural network whose aim to learn compressed representations of the data, typically for dimensionality reduction, with a constraint on the activity of the neurons in the network.

Namespaces

- **mlpack**
 .hpp
- **mlpack::nn**

40.663.1 Detailed Description

Author

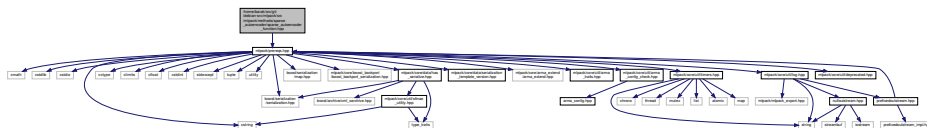
Siddharth Agrawal

An implementation of sparse autoencoders.

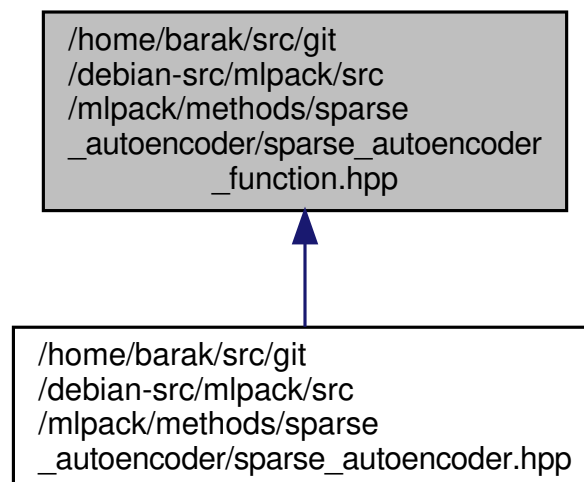
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.↵

40.664 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_autoencoder/sparse_↵ _autoencoder_function.hpp File Reference

Include dependency graph for sparse_autoencoder_function.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **DataDependentRandomInitializer**
A data-dependent random dictionary initializer for *SparseCoding* (p. 1916).

Namespaces

- mlpack**
.hpp
- mlpack::sparse_coding**

40.665.1 Detailed Description

Author

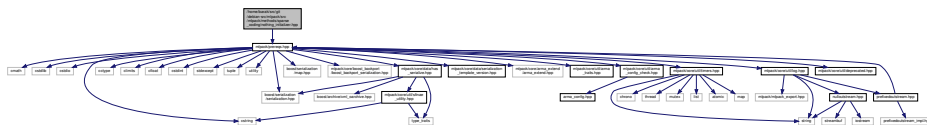
Nishant Mehta

A sensible heuristic for initializing dictionaries for sparse coding.

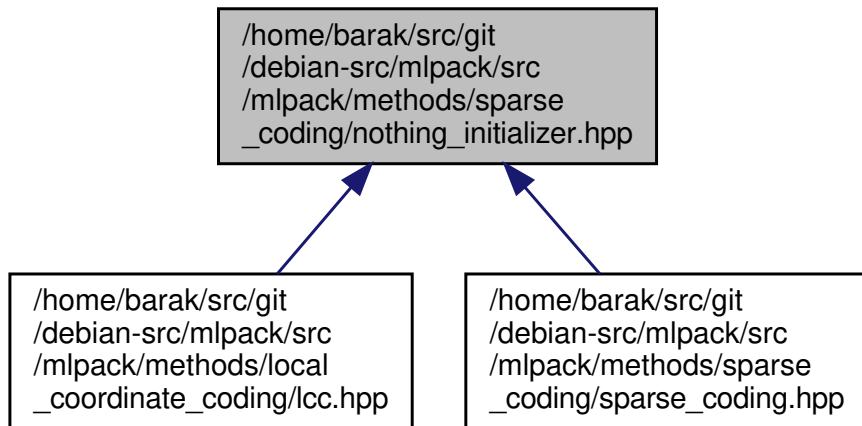
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.666 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_coding/nothing_initializer.hpp File Reference

Include dependency graph for nothing_initializer.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **NothingInitializer**

A DictionaryInitializer for **SparseCoding** (p. 1916) which does not initialize anything; it is useful for when the dictionary is already known and will be set with **SparseCoding::Dictionary()** (p. 1920).

Namespaces

- **mlpack**
 .hpp
- **mlpack::sparse_coding**

40.666.1 Detailed Description

Author

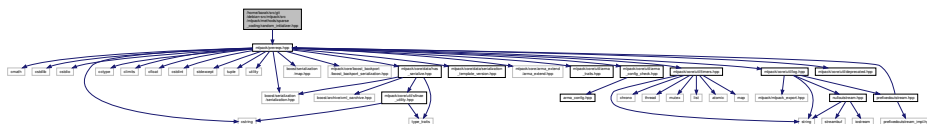
Ryan Curtin

An initializer for SparseCoding which does precisely nothing. It is useful for when you have an already defined dictionary and you plan on setting it with SparseCoding::Dictionary().

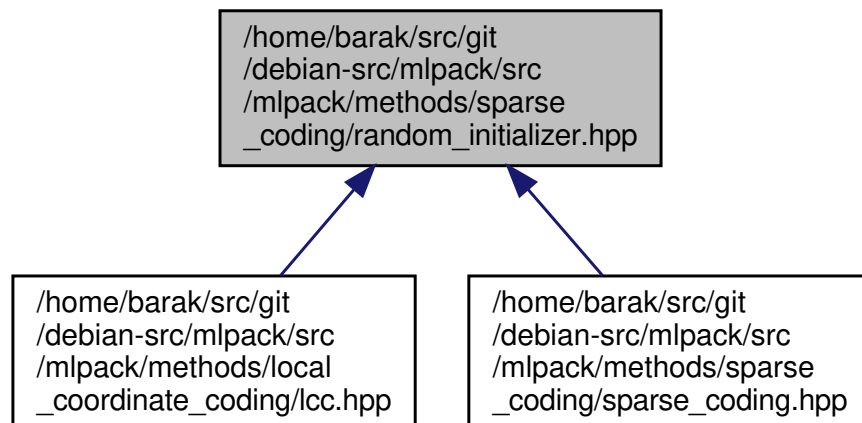
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.667 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_coding/random_← _initializer.hpp File Reference

Include dependency graph for random_initializer.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class **RandomInitializer**
A *DictionaryInitializer* for use with the **SparseCoding** (p. 1916) class.

Namespaces

- **mlpack**
.hpp
- **mlpack::sparse_coding**

40.667.1 Detailed Description

Author

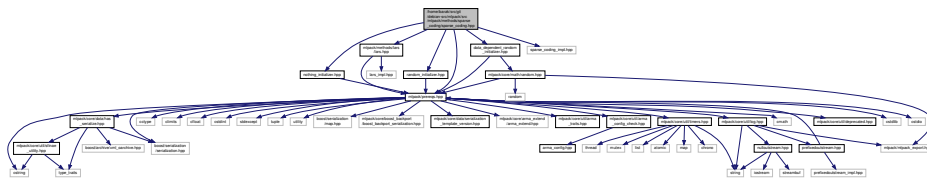
Nishant Mehta

A very simple random dictionary initializer for SparseCoding; it is probably not a very good choice.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.668 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_coding/sparse_coding.hpp File Reference

Include dependency graph for sparse_coding.hpp:



Classes

- class **SparseCoding**
An implementation of Sparse Coding with Dictionary Learning that achieves sparsity via an l_1 -norm regularizer on the codes (LASSO) or an (l_1+l_2) -norm regularizer on the codes (the Elastic Net).

Namespaces

- **mlpack**
.hpp
- **mlpack::sparse_coding**

40.668.1 Detailed Description

Author

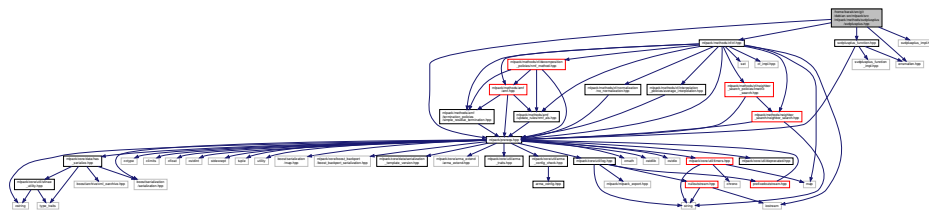
Nishant Mehta

Definition of the SparseCoding class, which performs L1 (LASSO) or L1+L2 (Elastic Net)-regularized sparse coding with dictionary learning

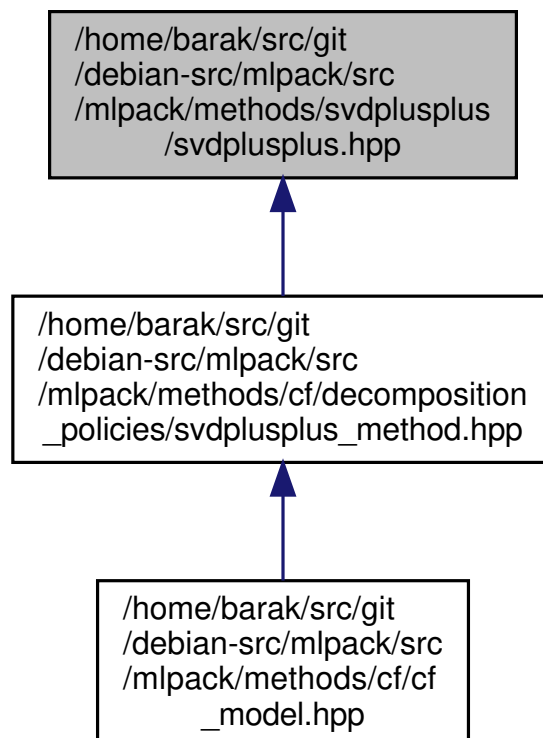
mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.669 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/svdplusplus/svdplusplus.hpp File Reference

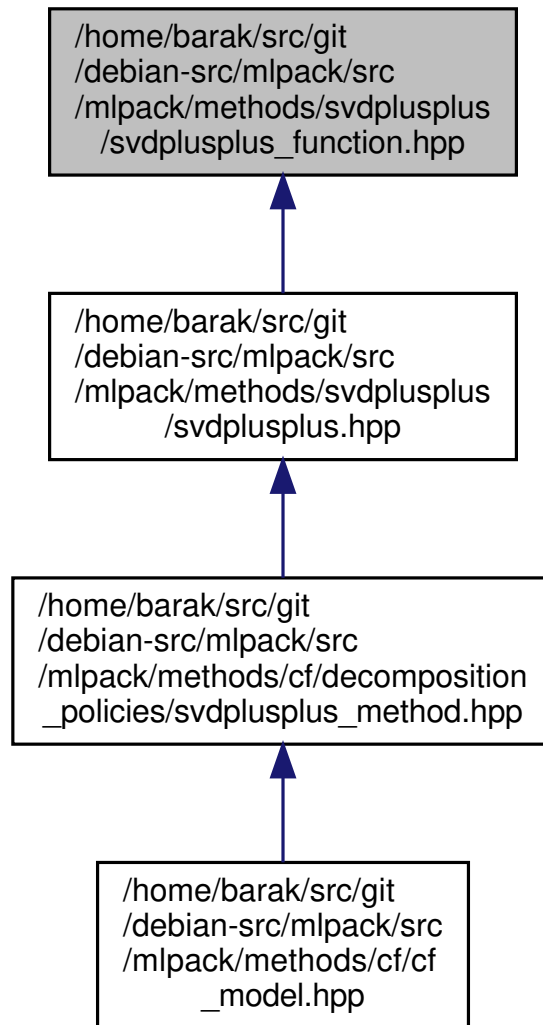
Include dependency graph for svdplusplus.hpp:



This graph shows which files directly or indirectly include this file:



This graph shows which files directly or indirectly include this file:



Classes

- class **SVDPlusPlusFunction**< **MatType** >

This class contains methods which are used to calculate the cost of SVD++'s objective function, to calculate gradient of parameters with respect to the objective function, etc.

Namespaces

- **ens**
- **mlpack**
 .hpp
- **mlpack::svd**

40.670.1 Detailed Description

Author

Siddharth Agrawal
Wenhao Huang

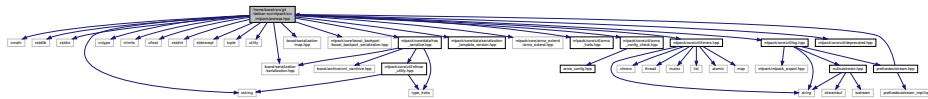
An implementation of the SVDPlusPlusFunction class.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.671 /home/barak/src/git/debian-src/mlpack/src/mlpack/prereqs.hpp File Reference

The core includes that mlpack expects; standard C++ includes and Armadillo.

Include dependency graph for prereqs.hpp:



Namespaces

- **std**

Macros

- `#define _USE_MATH_DEFINES`
- `#define BOOST_MPL_CFG_NO_PREPROCESSED_HEADERS`
- `#define BOOST_MPL_LIMIT_LIST_SIZE 50`
- `#define BOOST_PFTO`
- `#define force_inline`
- `#define M_PI 3.141592653589793238462643383279`
- `#define omp_size_t size_t`

Typedefs

- `template<bool B, class T = void>`
using **enable_if_t** = typename enable_if< B, T >::type

40.671.1 Detailed Description

The core includes that mpack expects; standard C++ includes and Armadillo.

mpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.671.2 Macro Definition Documentation

40.671.2.1 _USE_MATH_DEFINES

```
#define _USE_MATH_DEFINES
```

Definition at line 15 of file prereqs.hpp.

40.671.2.2 BOOST_MPL_CFG_NO_PREPROCESSED_HEADERS

```
#define BOOST_MPL_CFG_NO_PREPROCESSED_HEADERS
```

Definition at line 67 of file prereqs.hpp.

40.671.2.3 BOOST_MPL_LIMIT_LIST_SIZE

```
#define BOOST_MPL_LIMIT_LIST_SIZE 50
```

Definition at line 68 of file prereqs.hpp.

40.671.2.4 BOOST_PFTO

```
#define BOOST_PFTO
```

Definition at line 86 of file prereqs.hpp.

40.671.2.5 force_inline

```
#define force_inline
```

Definition at line 43 of file prereqs.hpp.

40.671.2.6 M_PI

```
#define M_PI 3.141592653589793238462643383279
```

Definition at line 39 of file prereqs.hpp.

Referenced by `Pendulum::AngleNormalize()`, `Acrobot::Dsdt()`, `Pendulum::InitialSample()`, `MockCategoricalData()`, `SphericalKernel::Normalizer()`, `GaussianKernel::Normalizer()`, `Pendulum::Sample()`, and `Acrobot::Sample()`.

40.671.2.7 omp_size_t

```
#define omp_size_t size_t
```

Definition at line 124 of file prereqs.hpp.

Referenced by `mlpack::data::Binarize()`.

40.672 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/ann_test_tools.hpp File Reference

Include dependency graph for `ann_test_tools.hpp`:



Functions

- `template<class FunctionType >`
`double CheckGradient (FunctionType &function, const double eps=1e-7)`
- `template<typename ModuleType >`
`double JacobianPerformanceTest (ModuleType &module, arma::mat &input, arma::mat &target, const double eps=1e-6)`
- `template<typename ModuleType >`
`double JacobianTest (ModuleType &module, arma::mat &input, const double minValue=-2, const double maxValue=-1, const double perturbation=1e-6)`
- `template<class T >`
`void ResetFunction (T &layer, typename std::enable_if< HasResetCheck< T, void(T::*)()>::value >::type ==0)`
- `template<class T >`
`void ResetFunction (T &, typename std::enable_if<!HasResetCheck< T, void(T::*)()>::value >::type ==0)`

40.672.1 Detailed Description

Author

Marcus Edel

This file includes some useful functions for ann tests.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.672.2 Function Documentation

40.672.2.1 CheckGradient()

```
double CheckGradient (
    FunctionType & function,
    const double eps = 1e-7 )
```

Definition at line 137 of file ann_test_tools.hpp.

40.672.2.2 JacobianPerformanceTest()

```
double JacobianPerformanceTest (
    ModuleType & module,
    arma::mat & input,
    arma::mat & target,
    const double eps = 1e-6 )
```

Definition at line 104 of file ann_test_tools.hpp.

40.672.2.3 JacobianTest()

```
double JacobianTest (
    ModuleType & module,
    arma::mat & input,
    const double minValue = -2,
    const double maxValue = -1,
    const double perturbation = 1e-6 )
```

Definition at line 40 of file ann_test_tools.hpp.

References `RandomInitialization::Initialize()`, and `ResetFunction()`.

40.673.1 Detailed Description

Author

Projyal Dev

A simple custom layer mimicing SigmoidLayer for testing if custom layers work.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

**40.674 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/braziltourism_labels.txt
File Reference**

40.675 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/corrupt-observations-1.txt File Reference

40.676 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/corrupt-observations-2.txt File Reference

**40.677 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/data_3d_ind.txt File
Reference**

**40.678 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/data_3d_mixed.txt File
Reference**

**40.679 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/iris_labels.txt File Ref-
erence**

**40.680 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/labels.txt File Refer-
ence**

**40.681 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/observations.txt File
Reference**

40.683 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/test_labels_nonlinsep.txt
File Reference

40.684 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/test_nonlinsep.txt File Reference

40.685 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/train_labels_nonlinsep.txt
File Reference

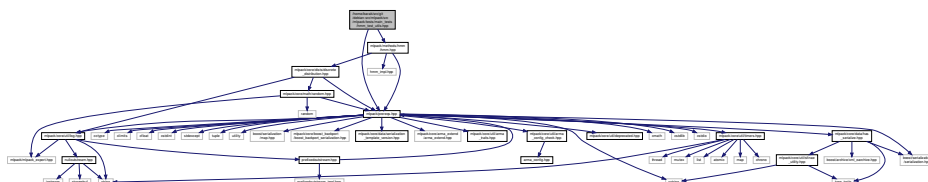
```
40.686 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/train_nonlinsep.txt File
Reference
```

40.687 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/vc2_labels.txt File Reference

40.688 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/vc2_test_labels.txt File Reference

40.689 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/main_tests/hmm_test_↵
utils.hpp File Reference

Include dependency graph for `hmm_test_utils.hpp`:



Classes

- struct **InitHMMModel**
- struct **TrainHMMModel**

40.689.1 Detailed Description

Author

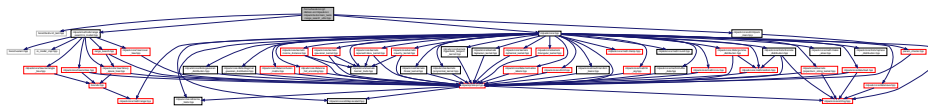
Daivik Nema

Structs for initializing and training HMMs (either of Discrete, Gaussian, GMM, or Diagonal GMM HMMs). These structs are passed as template parameters to the PerformAction function of an HMMModel object. These structs have been adapted from the structs in `mlpack/methods/hmm/hmm_train_main.cpp`.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.690 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/main_tests/range_search_utils.hpp File Reference

Include dependency graph for `range_search_utils.hpp`:



Functions

- void **CheckMatrices** (std::vector< std::vector< double >> &vec1, std::vector< std::vector< double >> &vec2, const double tolerance=1e-3)
Check that 2 matrices of type vector<vector<double>> are close to equal, using the given tolerance.
- void **CheckMatrices** (std::vector< std::vector< size_t >> &vec1, std::vector< std::vector< size_t >> &vec2)
Check that 2 matrices of type vector<vector<size_t>> are equal.
- std::string **ModelToString** (RSMModel *model)
*Convert a model to a string using the text_oarchive of **boost::serialization** (p. 251).*
- template<typename T >
 std::vector< std::vector< T >> **ReadData** (const std::string &filename)
Load a CSV file into a vector of vector with a templated datatype.

40.690.1 Detailed Description

Author

Niteya Shah

Helper functions used in the execution of the Range Search test.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.690.2 Function Documentation

40.690.2.1 CheckMatrices() [1/2]

```
void CheckMatrices (
    std::vector< std::vector< double >> & vec1,
    std::vector< std::vector< double >> & vec2,
    const double tolerance = 1e-3 ) [inline]
```

Check that 2 matrices of type `vector<vector<double>>` are close to equal, using the given tolerance.

Parameters

<i>vec1</i>	First vector to compare.
<i>vec2</i>	Second vector to compare.
<i>tolerance</i>	Allowed tolerance for values.

Definition at line 41 of file `range_search_utils.hpp`.

40.690.2.2 CheckMatrices() [2/2]

```
void CheckMatrices (
    std::vector< std::vector< size_t >> & vec1,
    std::vector< std::vector< size_t >> & vec2 ) [inline]
```

Check that 2 matrices of type `vector<vector<size_t>>` are equal.

Parameters

<i>vec1</i>	First vector to compare.
<i>vec2</i>	Second vector to compare.

Definition at line 64 of file `range_search_utils.hpp`.

40.690.2.3 ModelToString()

```
std::string ModelToString (
    RSModel * model ) [inline]
```

Convert a model to a string using the text_oarchive of **boost::serialization** (p. 251).

40.691.1 Detailed Description

Author

Eugene Freyman

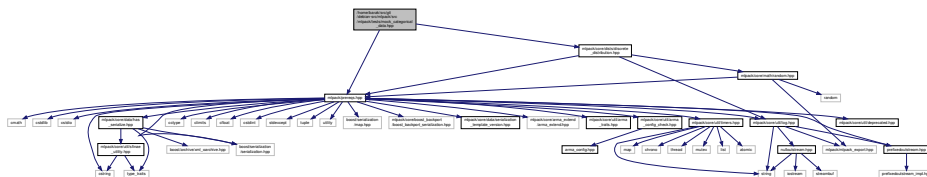
Helper functions for testing.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.692 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/mock_categorical_data.hpp File Reference

Generate categorical dataset for tests.

Include dependency graph for mock_categorical_data.hpp:



Functions

- void **MockCategoricalData** (arma::mat &d, arma::Row< size_t > &l, **mlpack::data::DatasetInfo** &datasetInfo)
Create a mock categorical dataset for testing.

40.692.1 Detailed Description

Generate categorical dataset for tests.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.692.2 Function Documentation

40.692.2.1 MockCategoricalData()

```
void MockCategoricalData (
    arma::mat & d,
    arma::Row< size_t > & l,
    mlpack::data::DatasetInfo & datasetInfo ) [inline]
```

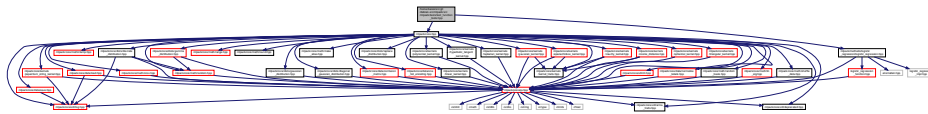
Create a mock categorical dataset for testing.

Definition at line 20 of file mock_categorical_data.hpp.

References `mlpack::data::categorical`, `M_PI`, `DatasetMapper< PolicyType, InputType >::MapString()`, `mlpack::math::Random()`, `DiscreteDistribution::Random()`, and `DatasetMapper< PolicyType, InputType >::Type()`.

40.693 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/test_function_tools.hpp File Reference

Include dependency graph for test_function_tools.hpp:



Functions

- void **LogisticRegressionTestData** (arma::mat &data, arma::mat &testData, arma::mat &shuffledData, arma::Row< size_t > &responses, arma::Row< size_t > &testResponses, arma::Row< size_t > &shuffledResponses)
Create the data for the a logistic regression test.

40.693.1 Detailed Description

Author

Marcus Edel

This file provides some useful test function methods.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.693.2 Function Documentation

40.693.2.1 LogisticRegressionTestData()

```

void LogisticRegressionTestData (
    arma::mat & data,
    arma::mat & testData,
    arma::mat & shuffledData,
    arma::Row< size_t > & responses,
    arma::Row< size_t > & testResponses,
    arma::Row< size_t > & shuffledResponses ) [inline]

```

Create the data for the a logistic regression test.

Parameters

<i>data</i>	Matrix object to store the data into.
<i>testData</i>	Matrix object to store the test data into.
<i>shuffledData</i>	Matrix object to store the shuffled data into.
<i>responses</i>	Matrix object to store the overall responses into.
<i>testResponses</i>	Matrix object to store the test responses into.
<i>shuffledResponses</i>	Matrix object to store the shuffled responses into.

Definition at line 33 of file test_function_tools.hpp.

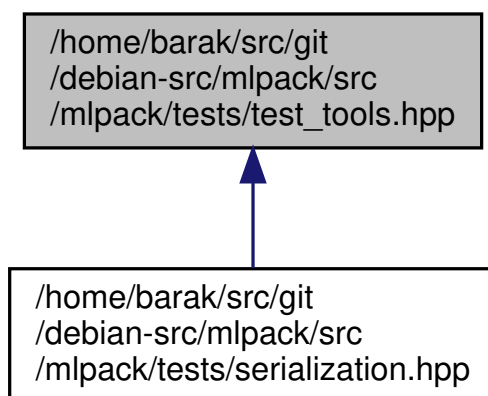
References GaussianDistribution::Random().

40.694 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/test_tools.hpp File Reference

Include dependency graph for test_tools.hpp:



This graph shows which files directly or indirectly include this file:



Macros

- `#define REQUIRE_RELATIVE_ERR(L, R, E) BOOST_REQUIRE_LE(std::abs((R) - (L)), (E) * std::abs(R))`

Functions

- void **CheckMatrices** (const arma::mat &a, const arma::mat &b, double tolerance=1e-5)
- void **CheckMatrices** (const arma::Mat< size_t > &a, const arma::Mat< size_t > &b)
- void **CheckMatrices** (const arma::cube &a, const arma::cube &b, double tolerance=1e-5)
- void **CheckMatricesNotEqual** (const arma::mat &a, const arma::mat &b, double tolerance=1e-5)
- void **CheckMatricesNotEqual** (const arma::Mat< size_t > &a, const arma::Mat< size_t > &b)
- void **CheckMatricesNotEqual** (const arma::cube &a, const arma::cube &b, double tolerance=1e-5)
- std::string **FilterFileName** (const std::string &inputString)

40.694.1 Detailed Description

Author

Ryan Curtin

This file includes some useful macros for tests.

mlpack is free software; you may redistribute it and/or modify it under the terms of the 3-clause BSD license. You should have received a copy of the 3-clause BSD license along with mlpack. If not, see <http://www.opensource.org/licenses/BSD-3-Clause> for more information.

40.694.2 Macro Definition Documentation

40.694.2.1 REQUIRE_RELATIVE_ERR

```
#define REQUIRE_RELATIVE_ERR(  
    L,  
    R,  
    E ) BOOST_REQUIRE_LE(std::abs((R) - (L)), (E) * std::abs(R))
```

Definition at line 20 of file test_tools.hpp.

40.694.3 Function Documentation

40.694.3.1 CheckMatrices() [1/3]

```
void CheckMatrices (
    const arma::mat & a,
    const arma::mat & b,
    double tolerance = 1e-5 ) [inline]
```

Definition at line 24 of file test_tools.hpp.

40.694.3.2 CheckMatrices() [2/3]

```
void CheckMatrices (
    const arma::Mat< size_t > & a,
    const arma::Mat< size_t > & b ) [inline]
```

Definition at line 41 of file test_tools.hpp.

40.694.3.3 CheckMatrices() [3/3]

```
void CheckMatrices (
    const arma::cube & a,
    const arma::cube & b,
    double tolerance = 1e-5 ) [inline]
```

Definition at line 52 of file test_tools.hpp.

40.694.3.4 CheckMatricesNotEqual() [1/3]

```
void CheckMatricesNotEqual (
    const arma::mat & a,
    const arma::mat & b,
    double tolerance = 1e-5 ) [inline]
```

Definition at line 70 of file test_tools.hpp.

40.694.3.5 CheckMatricesNotEqual() [2/3]

```
void CheckMatricesNotEqual (
    const arma::Mat< size_t > & a,
    const arma::Mat< size_t > & b ) [inline]
```

Definition at line 102 of file test_tools.hpp.

40.694.3.6 CheckMatricesNotEqual() [3/3]

```
void CheckMatricesNotEqual (
    const arma::cube & a,
    const arma::cube & b,
    double tolerance = 1e-5 ) [inline]
```

Definition at line 127 of file test_tools.hpp.

40.694.3.7 FilterFileName()

```
std::string FilterFileName (
    const std::string & inputString ) [inline]
```

Definition at line 161 of file test_tools.hpp.

Referenced by `mlpack::SerializeObject()`, `mlpack::SerializePointerObject()`, and `mlpack::TestArmadilloSerialization()`.

Index

/home/barak/src/git/debian-src/mlpack/doc/guide/bindings.↵
hpp, 2373

/home/barak/src/git/debian-src/mlpack/doc/guide/build.↵
hpp, 2373

/home/barak/src/git/debian-src/mlpack/doc/guide/build_↵
windows.hpp, 2373

/home/barak/src/git/debian-src/mlpack/doc/guide/cli_↵
quickstart.hpp, 2373

/home/barak/src/git/debian-src/mlpack/doc/guide/cv.hpp,
2373

/home/barak/src/git/debian-src/mlpack/doc/guide/formats.↵
hpp, 2374

/home/barak/src/git/debian-src/mlpack/doc/guide/hpt.hpp,
2577

/home/barak/src/git/debian-src/mlpack/doc/guide/iodoc.↵
hpp, 2374

/home/barak/src/git/debian-src/mlpack/doc/guide/matrices.↵
hpp, 2374

/home/barak/src/git/debian-src/mlpack/doc/guide/python↵
_quickstart.hpp, 2374

/home/barak/src/git/debian-src/mlpack/doc/guide/sample.↵
hpp, 2374

/home/barak/src/git/debian-src/mlpack/doc/guide/sample↵
_ml_app.hpp, 2374

/home/barak/src/git/debian-src/mlpack/doc/guide/timer.↵
hpp, 2375

/home/barak/src/git/debian-src/mlpack/doc/guide/version.↵
hpp, 2748

/home/barak/src/git/debian-src/mlpack/doc/policies/elemtyp.↵
hpp, 2375

/home/barak/src/git/debian-src/mlpack/doc/policies/functiontype.↵
hpp, 2375

/home/barak/src/git/debian-src/mlpack/doc/policies/kernels.↵
hpp, 2375

/home/barak/src/git/debian-src/mlpack/doc/policies/metrics.↵
hpp, 2375

/home/barak/src/git/debian-src/mlpack/doc/policies/trees.↵
hpp, 2375

/home/barak/src/git/debian-src/mlpack/doc/tutorials/amf/amf.↵
txt, 2375

/home/barak/src/git/debian-src/mlpack/doc/tutorials/ann/ann.↵
txt, 2375

/home/barak/src/git/debian-src/mlpack/doc/tutorials/approx↵
_kfn/approx_kfn.txt, 2376

/home/barak/src/git/debian-src/mlpack/doc/tutorials/cf/cf.↵
txt, 2376

/home/barak/src/git/debian-src/mlpack/doc/tutorials/det/det.↵
txt, 2376

/home/barak/src/git/debian-src/mlpack/doc/tutorials/emst/emst.↵
txt, 2378

/home/barak/src/git/debian-src/mlpack/doc/tutorials/fastmks/fastmks.↵
txt, 2378

/home/barak/src/git/debian-src/mlpack/doc/tutorials/kmeans/kmeans.↵
txt, 2379

/home/barak/src/git/debian-src/mlpack/doc/tutorials/linear↵
_regression/linear_regression.txt, 2379

/home/barak/src/git/debian-src/mlpack/doc/tutorials/neighbor↵
_search/neighbor_search.txt, 2379

/home/barak/src/git/debian-src/mlpack/doc/tutorials/range↵
_search/range_search.txt, 2380

/home/barak/src/git/debian-src/mlpack/doc/tutorials/tutorials.↵
txt, 2380

/home/barak/src/git/debian-src/mlpack/src/mlpack/C↵
MakeLists.txt, 2389

/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/↵
CMakeLists.txt, 2384

/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/↵
CMakeLists.txt, 2383

/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/add↵
_to_po.hpp, 2381

/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/cli↵
_option.hpp, 2382

/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/default↵
_param.hpp, 2456

/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/delete↵
_allocated_memory.hpp, 2460

/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/end↵
_program.hpp, 2462

/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get↵
_allocated_memory.hpp, 2463

/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get↵
_param.hpp, 2465

/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get↵
_printable_param.hpp, 2470

/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get↵
_printable_param_name.hpp, 2475

/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get↵
_printable_param_value.hpp, 2478

/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/cli/get↵
_printable_type.hpp, 2481

/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/c/arma_type.hpp, 2485
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/c/parameter_name.hpp, 2487
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/c/param.hpp, 2488
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/c/type.hpp, 2490
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/c/command_line.hpp, 2491
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/c/doc_functions.hpp, 2492
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/c/help.hpp, 2496
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/c/type_doc.hpp, 2497
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/c/param.hpp, 2501
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/c/string_type_param.hpp, 2503
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/mlpack/CMakeLists.txt, 2385
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/mlpack/arma_type.hpp, 2504
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/mlpack/param.hpp, 2458
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/mlpack/binding_name.hpp, 2505
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/mlpack/param.hpp, 2467
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/mlpack/printable_param.hpp, 2471
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/mlpack/printable_param_name.hpp, 2477
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/mlpack/printable_param_value.hpp, 2479
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/mlpack/printable_type.hpp, 2482
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/mlpack/serializable.hpp, 2506
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/mlpack/option.hpp, 2507
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/mlpack/printable.hpp, 2531
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/mlpack/tests/doc_functions.hpp, 2494
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/mlpack/tests/docs.hpp, 2508
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/mlpack/tests/type_doc.hpp, 2499
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/mlpack/tests/delete_doc_wrapper.hpp, 2509
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/CMakeLists.txt, 2386
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/get_arma_type.hpp, 2510
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/get_cython_type.hpp, 2512
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/get_numpy_type.hpp, 2513
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/get_numpy_type_char.hpp, 2515
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/param.hpp, 2468
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/printable_param.hpp, 2472
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/printable_type.hpp, 2483
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/import_decl.hpp, 2516
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/mlpack/arma_type.hpp, 2518
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/mlpack/string_type_param.hpp, 2519
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/mlpack/arma_type.hpp, 2521
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/printable_class_defn.hpp, 2522
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/printable_defn.hpp, 2524
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/printable_doc.hpp, 2525
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/printable_doc_functions.hpp, 2495
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/printable_input_processing.hpp, 2526
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/printable_output_processing.hpp, 2528
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/printable_pyx.hpp, 2530
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/printable_type_doc.hpp, 2500
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/py_option.hpp, 2530
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/strip_type.hpp, 2531
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/python/tests/CMakeLists.txt, 2388
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/CMakeLists.txt, 2388
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/clean_memory.hpp, 2532
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/delete_allocated_memory.hpp, 2461
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/get_allocated_memory.hpp, 2464
 /home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/tests/get_param.hpp, 2469

/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/test/get.cpp, 2547	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/listwise_deletion.hpp, 2551
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/test/get.cpp, 2547	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/mean_imputation.hpp, 2552
/home/barak/src/git/debian-src/mlpack/src/mlpack/bindings/test/get.cpp, 2547	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputation_methods/median_imputation.hpp, 2553
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/impl/arma.hpp, 2535	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/imputer.hpp, 2553
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/impl/arma/CMakeLists.txt, 2389	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/isnan.hpp, 2554
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/impl/arma/CMakeLists.txt, 2390	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/load.hpp, 2555
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/impl/arma/_base.hpp, 2535	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/load_arff.hpp, 2556
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/impl/arma/_fold_cv.hpp, 2536	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/load_csv.hpp, 2557
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/impl/arma/_info_extractor.hpp, 2537	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/map_policies/CMakeLists.txt, 2393
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/impl/arma/CMakeLists.txt, 2390	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/map_policies/datatype.hpp, 2558
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/impl/arma.hpp, 2539	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/map_policies/increment_policy.hpp, 2559
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/impl/arma/_strategy.hpp, 2539	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/map_policies/missing_policy.hpp, 2560
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/impl/arma.hpp, 2540	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/normalize_labels.hpp, 2561
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/impl/arma.hpp, 2541	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/one_hot_encoding.hpp, 2562
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/impl/arma.hpp, 2542	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/save.hpp, 2563
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/impl/arma.hpp, 2543	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/serialization_template_version.hpp, 2564
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/impl/arma.hpp, 2543	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/split_data.hpp, 2565
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/cv/similarity/_cv.hpp, 2544	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/CMakeLists.txt, 2393
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/arma/CMakeLists.txt, 2391	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/diagonal_gaussian_distribution.hpp, 2566
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/arma.hpp, 2545	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/discrete_distribution.hpp, 2567
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/arma/_matrix.hpp, 2546	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/gamma_distribution.hpp, 2568
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/arma/_mapper.hpp, 2547	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/gaussian_distribution.hpp, 2569
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/arma_ext.hpp, 2548	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/laplace_distribution.hpp, 2570
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/arma_for_mat.hpp, 2549	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/dists/regression_distribution.hpp, 2571
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/arma/_serialize.hpp, 2549	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/CMakeLists.txt, 2394
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/arma/_methods/CMakeLists.txt, 2392	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/cv_function.hpp, 2572
/home/barak/src/git/debian-src/mlpack/src/mlpack/core/data/arma/_methods/custom_imputation.hpp, 2550	/home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/deduce_hp_types.hpp, 2573

/home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/fix_data.hpp, 2574
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/hpt/hpt.hpp, 2576
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/CMakeLists.txt, 2395
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/_kernel.hpp, 2577
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/_distance.hpp, 2578
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/_kernel.hpp, 2579
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/_kernel.hpp, 2580
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/_kernel.hpp, 2581
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/_tangent_kernel.hpp, 2582
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/_traits.hpp, 2583
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/_kernel.hpp, 2583
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/linear_kernel.hpp, 2584
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/polynomial_kernel.hpp, 2585
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/pspectrum_string_kernel.hpp, 2586
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/spherical_kernel.hpp, 2587
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/kernels/triangular_kernel.hpp, 2588
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/CMakeLists.txt, 2396
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/ccov.hpp, 2589
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/clamp.hpp, 2590
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/columns_to_blocks.hpp, 2591
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/lin_alg.hpp, 2592
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/log_add.hpp, 2594
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/make_alias.hpp, 2594
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/random.hpp, 2596
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/random_basis.hpp, 2597
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/range.hpp, 2598
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/round.hpp, 2599
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/math/shuffle_data.hpp, 2600
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/metrics/CMakeLists.txt, 2396
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/metrics/ip_metric.hpp, 2601
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/metrics/lmetric.hpp, 2603
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/metrics/mahalanobis_distance.hpp, 2604
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/CMakeLists.txt, 2397
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/address.hpp, 2604
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/ballbound.hpp, 2606
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree.hpp, 2608
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/binary_space_tree.hpp, 2607
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/breadth_first_dual_tree_traverser.hpp, 2608
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/dual_tree_traverser.hpp, 2610
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/mean_split.hpp, 2613
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/midpoint_split.hpp, 2615
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/rp_tree_max_split.hpp, 2616
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/rp_tree_mean_split.hpp, 2617
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/single_tree_traverser.hpp, 2618
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/traits.hpp, 2622
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/typedef.hpp, 2627
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/ub_tree_split.hpp, 2634
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/binary_space_tree/vantage_point_split.hpp, 2636
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/bound_traits.hpp, 2637
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/bounds.hpp, 2638
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cellbound.hpp, 2638
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cosine_tree.hpp, 2640
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree.hpp, 2642
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_tree.hpp, 2642

_tree/cover_tree.hpp, 2641
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵
 _tree/dual_tree_traverser.hpp, 2611 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ _tree/r_plus_plus_tree_descent_heuristic.hpp,
 _tree/first_point_is_root.hpp, 2643 2662
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵
 _tree/single_tree_traverser.hpp, 2619 _tree/r_plus_plus_tree_split_policy.hpp, 2663
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵
 _tree/traits.hpp, 2623 _tree/r_plus_tree_descent_heuristic.hpp, 2664
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵
 _tree/typedef.hpp, 2629 _tree/r_plus_tree_split.hpp, 2665
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵
 _tree.hpp, 2644 _tree/r_plus_tree_split_policy.hpp, 2666
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵
 _tree.hpp, 2645 _tree/r_star_tree_descent_heuristic.hpp, 2667
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵
 _single_tree_traverser.hpp, 2645 _tree/r_star_tree_split.hpp, 2668
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵
 _ball_bound.hpp, 2646 _tree/r_tree_descent_heuristic.hpp, 2669
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵
 _tree.hpp, 2647 _tree/r_tree_split.hpp, 2670
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵
 _tree.hpp, 2650 _tree/rectangle_tree.hpp, 2671
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵
 _tree_traverser.hpp, 2611 _tree/single_tree_traverser.hpp, 2621
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵
 _tree.hpp, 2648 _tree/traits.hpp, 2625
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵
 _tree_traverser.hpp, 2620 _tree/typedef.hpp, 2630
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵
 _tree.hpp, 2624 _tree/x_tree_auxiliary_information.hpp, 2673
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵
 _split.hpp, 2650 _tree/x_tree_split.hpp, 2673
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_↵
 _tree.hpp, 2672 _split/hyperplane.hpp, 2674
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_↵
 _tree/discrete_hilbert_value.hpp, 2651 _split/mean_space_split.hpp, 2676
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_↵
 _tree/dual_tree_traverser.hpp, 2612 _split/midpoint_space_split.hpp, 2678
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/cover_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_↵
 _tree/hilbert_r_tree_auxiliary_information.hpp, _split/projection_vector.hpp, 2679
 2653 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/space_↵
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵ _split/space_split.hpp, 2680
 _tree/hilbert_r_tree_descent_heuristic.hpp, /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_↵
 2654 _tree.hpp, 2687
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_↵
 _tree/hilbert_r_tree_split.hpp, 2655 _tree/is_spill_tree.hpp, 2681
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_↵
 _tree/minimal_coverage_sweep.hpp, 2656 _tree/spill_dual_tree_traverser.hpp, 2683
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_↵
 _tree/minimal_splits_number_sweep.hpp, 2658 _tree/spill_single_tree_traverser.hpp, 2684
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_↵
 _tree/no_auxiliary_information.hpp, 2659 _tree/spill_tree.hpp, 2686
 /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/rectangle_↵ /home/barak/src/git/debian-src/mlpack/src/mlpack/core/tree/spill_↵

CMakeLists.txt, 2406
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation/_functions/CMakeLists.txt, 2404
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation/_functions/hard_sigmoid_function.hpp, 2770
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation/_functions/identity_function.hpp, 2771
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation/_functions/logistic_function.hpp, 2772
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation/_functions/rectifier_function.hpp, 2773
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation/_functions/softplus_function.hpp, 2774
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation/_functions/softsign_function.hpp, 2775
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation/_functions/swish_function.hpp, 2776
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/activation/_functions/tanh_function.hpp, 2777
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented_tasks/CMakeLists.txt, 2404
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented_tasks/CMakeLists.txt, 2405
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented_tasks.hpp, 2778
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented_tasks.hpp, 2779
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented_tasks.hpp, 2780
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/augmented_tasks.hpp, 2781
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/brnn_convolution.hpp, 2782
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution/_rules/CMakeLists.txt, 2406
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution/_rules/border_modes.hpp, 2783
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution/_rules/fft_convolution.hpp, 2784
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution/_rules/naive_convolution.hpp, 2785
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/convolution/_rules/svd_convolution.hpp, 2786
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/dists/CMakeLists.txt, 2407
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/dists/beta_distribution.hpp, 2787
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/ffn.hpp, 2788
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/CMakeLists.txt, 2408
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/const_init.hpp, 2789
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/gaussian_init.hpp, 2790
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/glorot_init.hpp, 2791
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/he_init.hpp, 2792
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/init_rules_traits.hpp, 2793
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/kathirvalavakumar_subavathi_init.hpp, 2794
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/lecun_normal_init.hpp, 2795
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/network_init.hpp, 2796
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/nguyen_widrow_init.hpp, 2797
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/oivs_init.hpp, 2798
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/orthogonal_init.hpp, 2799
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/init_rules/task_domain_init.hpp, 2755
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/CMakeLists.txt, 2409
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/add.hpp, 2799
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/add.hpp, 2799
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/add.hpp, 2800
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/alpha.hpp, 2801
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/atrous.hpp, 2802
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/base.hpp, 2803
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/batch_norm.hpp, 2805
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/bilinear_interpolation.hpp, 2806
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/cut.hpp, 2807
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/concat.hpp, 2807
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/concat.hpp, 2809
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/concat.hpp, 2810
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/constant.hpp, 2811
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/convolution.hpp, 2812
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/dropout.hpp, 2813
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/dropout.hpp, 2814

/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/sequence/sequence.hpp, 2815
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/sequence/sequence.hpp, 2844
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/subviewer/subviewer.hpp, 2816
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/subviewer/subviewer.hpp, 2846
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/transpose/transpose_relu.hpp, 2817
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/transpose/transpose_convolution.hpp, 2847
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/vr/vr_class_reward.hpp, 2818
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/vr/vr_class_reward.hpp, 2848
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/CMakeLists.txt, 2819
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/CMakeLists.txt, 2410
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/cross_entropy_error.hpp, 2820
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/cross_entropy_error.hpp, 2849
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/dice_loss.hpp, 2821
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/dice_loss.hpp, 2849
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/earth_mover_distance.hpp, 2822
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/earth_mover_distance.hpp, 2850
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/kl_divergence.hpp, 2823
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/kl_divergence.hpp, 2851
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/mean_squared_error.hpp, 2824
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/mean_squared_error.hpp, 2852
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/negative_log_likelihood.hpp, 2826
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/negative_log_likelihood.hpp, 2853
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/reconstruction_loss.hpp, 2828
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/reconstruction_loss.hpp, 2854
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/sigmoid_cross_entropy_error.hpp, 2829
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/loss/loss_functions/sigmoid_cross_entropy_error.hpp, 2854
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear/linear_no_bias.hpp, 2830
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/rbm/rbm_no_bias.hpp, 2854
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear/linear_softmax.hpp, 2831
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/rbm/rbm_softmax.hpp, 2411
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear/linear_softmax.hpp, 2832
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/rbm/rbm_softmax.hpp, 2855
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear/linear_policies.hpp, 2833
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/rnn/rnn_policies.hpp, 2856
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear/linear_pooling.hpp, 2834
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/visitor_pooling.hpp, 2857
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear/linear_pooling.hpp, 2835
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/visitor_pooling.hpp, 2411
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear/linear_multiplier.hpp, 2836
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/visitor_multiplier.hpp, 2858
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear/linear_merge.hpp, 2837
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/visitor_merge.hpp, 2859
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear/linear_relu.hpp, 2838
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/visitor_relu.hpp, 2860
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear/linear_recurrent.hpp, 2839
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/visitor_recurrent.hpp, 2861
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear/linear_attention.hpp, 2840
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/visitor_attention.hpp, 2862
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear/linear_recurrent.hpp, 2841
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/visitor_recurrent.hpp, 2863
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear/linear_normal.hpp, 2842
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/visitor_normal.hpp, 2864
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear/linear_normal.hpp, 2842
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/visitor_normal.hpp, 2864
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear/linear_normal.hpp, 2843
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/visitor_normal.hpp, 2865
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/layer/linear/linear_normal.hpp, 2843
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/visitor_normal.hpp, 2865

<code>_update_visitor.hpp</code> , 2866	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/↵</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/CMakeLists.txt</code> , 2414	
<code>_visitor.hpp</code> , 2867	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/cf.↵</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/bradi.hpp</code> , 2893	
<code>_zero_visitor.hpp</code> , 2867	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/cf.↵</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/lowmodel.hpp</code> , 2894	
<code>_output_parameter_visitor.hpp</code> , 2868	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/lowpolies/CMakeLists.txt</code> , 2415	
<code>_visitor.hpp</code> , 2869	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/optimization/policies/batch_svd_method.hpp</code> , 2895	
<code>_height_visitor.hpp</code> , 2871	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/optimization/policies/bias_svd_method.hpp</code> , 2896	
<code>_parameter_visitor.hpp</code> , 2872	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/optimization/policies/nmf_method.hpp</code> , 2898	
<code>_width_visitor.hpp</code> , 2873	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/optimization/policies/randomized_svd_method.hpp</code> , 2899	
<code>_set_visitor.hpp</code> , 2875	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/optimization/policies/regularized_svd_method.hpp</code> , 2900	
<code>_visitor.hpp</code> , 2875	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/optimization/policies/svd_complete_method.hpp</code> , 2902	
<code>_cell_visitor.hpp</code> , 2876	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/optimization/policies/svd_incomplete_method.hpp</code> , 2903	
<code>_visitor.hpp</code> , 2877	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/decomposition</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/optimization/policies/svdplusplus_method.hpp</code> , 2904	
<code>_set_visitor.hpp</code> , 2879	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/interpolation↵</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/optimization/policies/CMakeLists.txt</code> , 2416	
<code>_set_visitor.hpp</code> , 2880	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/interpolation↵</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/optimization/policies/average_interpolation.hpp</code> , 2905	
<code>_output_parameter_visitor.hpp</code> , 2881	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/interpolation↵</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/optimization/policies/regression_interpolation.hpp</code> , 2906	
<code>_input_height_visitor.hpp</code> , 2882	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/interpolation↵</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/optimization/policies/similarity_interpolation.hpp</code> , 2907	
<code>_input_width_visitor.hpp</code> , 2883	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/neighbor↵</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/optimization/policies/CMakeLists.txt</code> , 2416	
<code>_set_visitor.hpp</code> , 2884	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/neighbor↵</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/ann/visitor/optimization/policies/cosine_search.hpp</code> , 2908	
<code>_size_visitor.hpp</code> , 2885	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/neighbor↵</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/approx↵ _search_policies/lmetric_search.hpp</code> , 2909	
<code>_kfn/CMakeLists.txt</code> , 2412	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/neighbor↵</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/approx↵ _search_policies/pearson_search.hpp</code> , 2910	
<code>_kfn/drussila_select.hpp</code> , 2887	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/approx↵ CMakeLists.txt</code> , 2417	
<code>_kfn/qdafn.hpp</code> , 2888	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/bias↵ _normalization.hpp</code> , 2911	
<code>_svd/CMakeLists.txt</code> , 2413	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/bias↵ _mean_normalization.hpp</code> , 2912	
<code>_svd/bias_svd.hpp</code> , 2889	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/bias↵ _normalization.hpp</code> , 2913	
<code>_svd/bias_svd_function.hpp</code> , 2890	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/block↵ _mean_normalization.hpp</code> , 2914	
<code>_krylov_svd/CMakeLists.txt</code> , 2414	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/</code>
<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/block↵ _mean_normalization.hpp</code> , 2915	
<code>_krylov_svd/randomized_block_krylov_svd.hpp</code> , 2892	<code>/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/cf/normalization/</code>
	<code>score_normalization.hpp</code> , 2915

_trees/numeric_split_info.hpp, 2966
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/hoeffding/
 _trees/typedef.hpp, 2633
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 CMakeLists.txt, 2426
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 hpp, 2967
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _model.hpp, 2968
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _rules.hpp, 2970
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _stat.hpp, 2971
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _pca/CMakeLists.txt, 2426
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _pca/kernel_pca.hpp, 2972
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _pca/kernel_rules/CMakeLists.txt, 2427
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _pca/kernel_rules/naive_method.hpp, 2973
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _pca/kernel_rules/nystroem_method.hpp, 2974
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 CMakeLists.txt, 2428
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _empty_clusters.hpp, 2975
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _tree_kmeans.hpp, 2976
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _tree_kmeans_rules.hpp, 2977
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _tree_kmeans_statistic.hpp, 2978
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _kmeans.hpp, 2979
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _kmeans.hpp, 2980
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _empty_clusters.hpp, 2981
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 kmeans.hpp, 2981
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _variance_new_cluster.hpp, 2983
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _kmeans.hpp, 2984
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _moore_kmeans.hpp, 2985
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _moore_kmeans_rules.hpp, 2986
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _moore_kmeans_statistic.hpp, 2987
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _partition.hpp, 2988
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _start.hpp, 2989
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/kmeans/
 _initialization.hpp, 2990
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lars/
 CMakeLists.txt, 2429
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lars/
 hpp, 2991
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear/
 _regression/CMakeLists.txt, 2429
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear/
 _regression/linear_regression.hpp, 2992
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear/
 _svm/CMakeLists.txt, 2430
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear/
 _svm/linear_svm.hpp, 2994
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/linear/
 _svm/linear_svm_function.hpp, 2995
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lmnn/
 CMakeLists.txt, 2431
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lmnn/
 hpp, 2996
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lmnn/
 hpp, 2997
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lmnn/
 _function.hpp, 2998
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/local/
 _coordinate_coding/CMakeLists.txt, 2432
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/local/
 _coordinate_coding/lcc.hpp, 2999
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/logistic/
 _regression/CMakeLists.txt, 2432
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/logistic/
 _regression/logistic_regression.hpp, 3000
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/logistic/
 _regression/logistic_regression_function.hpp, 3001
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lsh/
 CMakeLists.txt, 2433
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/lsh/
 hpp, 3002
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/matrix/
 _completion/CMakeLists.txt, 2434
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/matrix/
 _completion/matrix_completion.hpp, 3004
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/mean/
 _shift/CMakeLists.txt, 2434
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/mean/
 _shift/mean_shift.hpp, 3005
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/naive/
 _bayes/CMakeLists.txt, 2435
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/naive/
 _bayes/naive_bayes_classifier.hpp, 3005
 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/nca/
 CMakeLists.txt, 2436


```
util.hpp, 3045 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/softmax_↵
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/regularized_regression/softmax_regression.hpp, 3064
_svd/CMakeLists.txt, 2447 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/softmax_↵
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/regularized_regression/softmax_regression_function.hpp,
_svd/regularized_svd.hpp, 3046 3065
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/regularized_regression/softmax_regression_function.hpp, 3047 _autoencoder/CMakeLists.txt, 2452
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/regularized_regression/softmax_regression_function.hpp, 3048 _autoencoder/maximal_inputs.hpp, 3066
_learning/CMakeLists.txt, 2448 _autoencoder/sparse_autoencoder.hpp, 3066
_learning/async_learning.hpp, 3049 _autoencoder/sparse_autoencoder_function.↵
_learning/environment/CMakeLists.txt, 2449 _autoencoder/sparse_autoencoder_function.↵
_learning/environment/acrobot.hpp, 3050 http, 3067
_learning/environment/cart_pole.hpp, 3051 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_↵
_learning/environment/continuous_mountain_↵ nothing/CMakeLists.txt, 2453
_car.hpp, 3052 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_↵
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/nothing_initializer.hpp, 3069
_learning/environment/mountain_car.hpp, 3053 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_↵
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/random_initializer.hpp, 3070
_learning/environment/pendulum.hpp, 3054 /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/sparse_↵
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/sparse_coding.hpp, 3071
_learning/environment/reward_clipping.hpp, /home/barak/src/git/debian-src/mlpack/src/mlpack/methods/svdplusplus/↵
3055 CMakeLists.txt, 2454
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/policy/CMakeLists.txt, 2449 http, 3072
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/policy/aggregated_policy.hpp, 3056 _function.hpp, 3073
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/policy/greedy_policy.hpp, 3056 /home/barak/src/git/debian-src/mlpack/src/mlpack/prereqs.↵
_learning/policy/greedy_policy.hpp, 3056 http, 3075
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/q_learning.hpp, 3057 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/↵
_learning/replay/CMakeLists.txt, 2450 CMakeLists.txt, 2454
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/replay/random_replay.hpp, 3058 _test_tools.hpp, 3077
_homebarak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/replay/random_replay.hpp, 3058 _layer.hpp, 3079
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/training_config.hpp, 3059 _labels.txt, 3080
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/CMakeLists.txt, 2451 observations-1.txt, 3080
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/n_step_q_learning_worker.↵ observations-2.txt, 3080
http, 3060 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/data_↵
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/one_step_q_learning_↵ 3d_ind.txt, 3080
worker.hpp, 3061 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/data_↵
3d_mixed.txt, 3080
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/reinforcement_learning/worker/one_step_sarsa_worker.hpp, _labels.txt, 3080
3063 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/labels.↵
/home/barak/src/git/debian-src/mlpack/src/mlpack/methods/softmax_↵ txt, 3080
_regression/CMakeLists.txt, 2451 /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/observations.↵
```

- txt, 3080
- /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/rf1
txt, 3081
- /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/test
_labels_nonlinsep.txt, 3081
- /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/test
_nonlinsep.txt, 3081
- /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/train
_labels_nonlinsep.txt, 3081
- /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/train
_nonlinsep.txt, 3081
- /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/vc2
_labels.txt, 3081
- /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/data/vc2
_test_labels.txt, 3081
- /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/main~
_tests/hmm_test_utils.hpp, 3081
- /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/main~
_tests/range_search_utils.hpp, 3082
- /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/main~
_tests/test_helper.hpp, 3084
- /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/mock~
_categorical_data.hpp, 3085
- /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/serializ~
hpp, 2521
- /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/test~
_function_tools.hpp, 3086
- /home/barak/src/git/debian-src/mlpack/src/mlpack/tests/test~
_tools.hpp, 3087
- \$V
- det.txt, 2377
- _USE_MATH_DEFINES
- prereqs.hpp, 3076
- ~AdaBoostModel
- mlpack::adaboost::AdaBoostModel, 496
- ~AddMerge
- mlpack::ann::AddMerge, 560
- ~BallBound
- mlpack::bound::BallBound, 995
- ~BinarySpaceTree
- mlpack::tree::BinarySpaceTree, 2007
- ~CFModel
- mlpack::cf::CFModel, 1043
- ~Concat
- mlpack::ann::Concat, 625
- ~CosineTree
- mlpack::tree::CosineTree, 2033
- ~CoverTree
- mlpack::tree::CoverTree, 2049
- ~DTree
- mlpack::det::DTree, 1213
- ~DecisionTree
- mlpack::tree::DecisionTree, 2072
- ~DropConnect
- mlpack::ann::DropConnect, 668
- ~DualTreeBoruvka
- mlpack::emst::DualTreeBoruvka, 1275
- ~DualTreeKMeans
- mlpack::kmeans::DualTreeKMeans, 1468
- ~EmptyStatistic
- mlpack::tree::EmptyStatistic, 2080
- ~FFN
- mlpack::ann::FFN, 695
- ~FastMKS
- mlpack::fastmks::FastMKS, 1286
- ~FastMKSMModel
- mlpack::fastmks::FastMKSMModel, 1294
- ~GRU
- mlpack::ann::GRU, 733
- ~GammaDistribution
- mlpack::distribution::GammaDistribution, 1241
- ~HMMModel
- mlpack::hmm::HMMModel, 1351
- ~HRectBound
- mlpack::bound::HRectBound, 1021
- ~HoeffdingTree
- mlpack::tree::HoeffdingTree, 2118
- ~HoeffdingTreeModel
- mlpack::tree::HoeffdingTreeModel, 2130
- ~HollowBallBound
- mlpack::bound::HollowBallBound, 1010
- ~IPMetric
- mlpack::metric::IPMetric, 1567
- ~KDE
- mlpack::kde::KDE, 1391
- ~KDEModel
- mlpack::kde::KDEModel, 1400
- ~MultiplyMerge
- mlpack::ann::MultiplyMerge, 830
- ~NSModel
- mlpack::neighbor::NSModel, 1661
- ~NeighborSearch
- mlpack::neighbor::NeighborSearch, 1633
- ~Octree
- mlpack::tree::Octree, 2175
- ~PellegMooreKMeans
- mlpack::kmeans::PellegMooreKMeans, 1494
- ~RAModel
- mlpack::neighbor::RAModel, 1672
- ~RASearch
- mlpack::neighbor::RASearch, 1685
- ~RNN
- mlpack::ann::RNN, 914
- ~RSModel
- mlpack::range::RSModel, 1775
- ~RangeSearch
- mlpack::range::RangeSearch, 1758
- ~RectangleTree

- mlpack::tree::RectangleTree, 2215
- ~Recurrent
 - mlpack::ann::Recurrent, 888
- ~SVDCompleteIncrementalLearning
 - mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >, 545
- ~Sequential
 - mlpack::ann::Sequential, 930
- ~SpillTree
 - mlpack::tree::SpillTree, 2283
- ~UnionFind
 - mlpack::emst::UnionFind, 1279
- A
 - mlpack::ann::AlphaDropout, 569
- AMF< TerminationPolicyType, InitializationRuleType, UpdateRuleType >, 499
- AMF
 - mlpack::amf::AMF, 500
- AbsoluteError
 - mlpack::kde::KDEModel, 1400
 - mlpack::kde::KDE, 1391
- AccuLog
 - mlpack::math, 410
- Accuracy, 1127
- Acrobat
 - mlpack::rl, 441
- Acrobot, 1825
 - mlpack::rl::Acrobot, 1827
- Acrobot::State, 1831
- Action
 - mlpack::rl::Acrobot, 1826
 - mlpack::rl::CartPole, 1843
 - mlpack::rl::MountainCar, 1862
 - mlpack::rl::RewardClipping, 1896
- action
 - mlpack::rl::ContinuousMountainCar::Action, 1854
 - mlpack::rl::Pendulum::Action, 1881
- ActionType
 - mlpack::rl::AggregatedPolicy, 1836
 - mlpack::rl::GreedyPolicy, 1859
 - mlpack::rl::NStepQLearningWorker, 1869
 - mlpack::rl::OneStepQLearningWorker, 1872
 - mlpack::rl::OneStepSarsaWorker, 1875
 - mlpack::rl::QLearning, 1886
 - mlpack::rl::RandomReplay, 1891
- ActiveSet
 - mlpack::regression::LARS, 1786
- AdaBoost
 - mlpack::adaboost::AdaBoost, 490, 491
- AdaBoost< WeakLearnerType, MatType >, 488
- AdaBoostModel, 494
 - mlpack::adaboost::AdaBoostModel, 495, 496
- Add
 - mlpack::CLI, 1121
 - mlpack::ann::Add, 555
 - mlpack::ann::AddMerge, 561
 - mlpack::ann::BRNN, 613
 - mlpack::ann::Concat, 625
 - mlpack::ann::FFN, 696
 - mlpack::ann::MultiplyMerge, 831
 - mlpack::ann::RNN, 914
 - mlpack::ann::Sequential, 930
 - mlpack::ann::VRClassReward, 969
- Add< InputDataType, OutputDataType >, 553
- add_cli_executable
 - methods/nmf/CMakeLists.txt, 2438
- add_executable
 - tests/CMakeLists.txt, 2455
- add_subdirectory
 - bindings/CMakeLists.txt, 2384
 - core/CMakeLists.txt, 2389
 - core/cv/CMakeLists.txt, 2390
 - methods/CMakeLists.txt, 2418
- AddMerge
 - mlpack::ann::AddMerge, 560
- AddMerge< InputDataType, OutputDataType, CustomLayers >, 558
- AddTask, 580
 - mlpack::ann::augmented::tasks::AddTask, 581
- AddToPO
 - mlpack::bindings::cli, 284, 285
- AddVisitor
 - mlpack::ann::AddVisitor, 566
- AddVisitor< CustomLayers >, 565
- AddressToPoint
 - mlpack::bound::addr, 365
- AggregatedPolicy
 - mlpack::rl::AggregatedPolicy, 1836
- AggregatedPolicy< PolicyType >, 1835
- alias
 - mlpack::util::ParamData, 2357
- Aliases
 - mlpack::CLI, 1122
- AllCategoricalSplit< FitnessFunction >, 1981
- AllCategoricalSplit< FitnessFunction >::AuxiliarySplit< Info< ElemType >, 1984
- AllDimensionSelect, 1984
 - mlpack::tree::AllDimensionSelect, 1985
- AllowEmptyClusters, 1464
 - mlpack::kmeans::AllowEmptyClusters, 1465
- Alpha
 - mlpack::adaboost::AdaBoost, 491
 - mlpack::ann::ELU, 683
 - mlpack::ann::FlexibleReLU, 709
 - mlpack::ann::LeakyReLU, 772
 - mlpack::ann::PReLU, 857
 - mlpack::cf::BiasSVDPolicy, 1037

- mlpack::cf::SVDPlusPlusPolicy, 1103, 1104
- mlpack::distribution::GammaDistribution, 1241
- mlpack::neighbor::RAModel, 1673
- mlpack::neighbor::RASearch, 1686
- alpha
 - det.txt, 2377
- AlphaDash
 - mlpack::ann::AlphaDropout, 569
- AlphaDropout
 - mlpack::ann::AlphaDropout, 568
- AlphaDropout< InputDataType, OutputDataType >, 567
- AlphaUpper
 - mlpack::det::DTree, 1213
- AlphaVisitor, 1590
- Angle
 - mlpack::rl::CartPole::State, 1848
- AngleNormalize
 - mlpack::rl::Pendulum, 1879
- Angles
 - mlpack::radical::Radical, 1745
- AngularVelocity
 - mlpack::rl::CartPole::State, 1848
 - mlpack::rl::Pendulum::State, 1883
- AngularVelocity1
 - mlpack::rl::Acrobot::State, 1833
- AngularVelocity2
 - mlpack::rl::Acrobot::State, 1833
- ann_test_tools.hpp
 - CheckGradient, 3078
 - JacobianPerformanceTest, 3078
 - JacobianTest, 3078
 - ResetFunction, 3078, 3079
- Anneal
 - mlpack::rl::AggregatedPolicy, 1836
 - mlpack::rl::GreedyPolicy, 1860
- Apply
 - InitHMMModel, 480
 - mlpack::amf::AMF, 500
 - mlpack::cf::BatchSVDPolicy, 1033
 - mlpack::cf::BiasSVDPolicy, 1038
 - mlpack::cf::NMFPolicy, 1067
 - mlpack::cf::RandomizedSVDPolicy, 1080
 - mlpack::cf::RegSVDPolicy, 1089
 - mlpack::cf::SVDCompletePolicy, 1095
 - mlpack::cf::SVDIncompletePolicy, 1099
 - mlpack::cf::SVDPlusPlusPolicy, 1104
 - mlpack::cf::SVDWrapper, 1109
 - mlpack::kernel::NystroemMethod, 1449
 - mlpack::kpca::KernelPCA, 1508–1510
 - mlpack::pca::ExactSVDPolicy, 1722
 - mlpack::pca::PCA, 1724–1726
 - mlpack::pca::QUICSVDPolicy, 1728
 - mlpack::pca::RandomizedBlockKrylovSVDPolicy, 1731
 - mlpack::pca::RandomizedSVDPolicy, 1733
 - mlpack::svd::BiasSVD, 1926
 - mlpack::svd::RandomizedBlockKrylovSVD, 1936
 - mlpack::svd::RandomizedSVD, 1940, 1941
 - mlpack::svd::RegularizedSVD, 1944
 - mlpack::svd::SVDPlusPlus, 1952, 1953
 - TrainHMMModel, 2371
- ApplyConstraint
 - mlpack::gmm::DiagonalConstraint, 1307
 - mlpack::gmm::EigenvalueRatioConstraint, 1318
 - mlpack::gmm::NoConstraint, 1333
 - mlpack::gmm::PositiveDefiniteConstraint, 1334
- ApplyKernelMatrix
 - mlpack::kpca::NaiveKernelRule, 1511
 - mlpack::kpca::NystroemKernelRule, 1512
- ApplyNormalizer
 - mlpack::kde::KernelNormalizer, 1411
- arma_config.hpp
 - MLPACK_ARMAS_NO64BIT_WORD, 2691
 - MLPACK_ARMAS_USE_OPENMP, 2691
- arma_util.hpp
 - GetMemState, 2519
 - GetMemory, 2519
 - SetMemState, 2519
- ArmaSVDFactorizer
 - mlpack::cf, 370
- Assert
 - mlpack::Log, 1539
- AssertDataConsistency
 - mlpack::cv::CVBase, 1131
- AssertSizes
 - mlpack::cv, 372
- AssertWeightsConsistency
 - mlpack::cv::CVBase, 1131
- AssignToFirstTree
 - mlpack::tree::RPlusPlusTreeSplitPolicy, 2249
 - mlpack::tree::RPlusTreeSplitPolicy, 2255
- AssignToLeftNode
 - mlpack::tree::MeanSplit, 2143
 - mlpack::tree::MidpointSplit, 2148
 - mlpack::tree::RPTreeMaxSplit, 2257
 - mlpack::tree::RPTreeMeanSplit, 2262
 - mlpack::tree::VantagePointSplit, 2333
- AssignToSecondTree
 - mlpack::tree::RPlusPlusTreeSplitPolicy, 2249
 - mlpack::tree::RPlusTreeSplitPolicy, 2255
- AsyncLearning
 - mlpack::rl::AsyncLearning, 1839
- AsyncLearning< WorkerType, EnvironmentType, NetworkType, UpdaterType, PolicyType >, 1837
- Atoms
 - mlpack::lcc::LocalCoordinateCoding, 1516
 - mlpack::sparse_coding::SparseCoding, 1920
- AtrousConvolution

- mlpack::ann::AtrousConvolution, 574
- AtrousConvolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, InputDataType, OutputDataType >, 572
- AuxBound
 - mlpack::neighbor::NeighborSearchStat, 1655
- AuxiliaryInfo
 - mlpack::tree::RectangleTree, 2216
- AuxiliaryInformation
 - mlpack::tree::RectangleTree, 2212, 2233
- AverageInitialization, 502
 - mlpack::amf::AverageInitialization, 503
- AverageInterpolation, 1030
 - mlpack::cf::AverageInterpolation, 1031
- AverageStrategy
 - mlpack::cv, 372
- AxisOrthogonalHyperplane
 - mlpack::tree, 450
- AxisParallelProjVector, 1986
 - mlpack::tree::AxisParallelProjVector, 1987
- B
 - mlpack::ann::AlphaDropout, 569
- BASH_CLEAR
 - cli_option.hpp, 2383
- BASH_RED
 - cli_option.hpp, 2383
- BINDING_TYPE_CLI
 - mlpack_main.hpp, 2700
- BINDING_TYPE_MARKDOWN
 - mlpack_main.hpp, 2700
- BINDING_TYPE_PYX
 - mlpack_main.hpp, 2701
- BINDING_TYPE_TEST
 - mlpack_main.hpp, 2701
- BINDING_TYPE_UNKNOWN
 - mlpack_main.hpp, 2701
- BINDING_TYPE
 - mlpack_main.hpp, 2700
- BOOST_CLASS_VERSION
 - hmm_model.hpp, 2953
- BOOST_MPL_CFG_NO_PREPROCESSED_HEADERS
 - prereqs.hpp, 3076
- BOOST_MPL_LIMIT_LIST_SIZE
 - prereqs.hpp, 3076
- BOOST_PFTO
 - prereqs.hpp, 3076
- BOOST_STATIC_CONSTANT
 - boost::serialization::version< mlpack::adaboost::AdaBoost< WeakLearnerType, MatType > >, 477
 - boost::serialization::version< mlpack::ann::BRNN< OutputLayerType, MergeLayerType, MergeOutputType, InitializationRuleType, CustomLayers >, 610
- OutputType, InitializationRuleType, CustomLayer... > >, 478
- boost::serialization::version< mlpack::ann::FFN< N< OutputLayerType, InitializationRuleType, CustomLayer... > >, 479
- boost::serialization::version< mlpack::ann::RN< N< OutputLayerType, InitializationRuleType, CustomLayer... > >, 479
- BOOST_TEMPLATE_CLASS_VERSION
 - lsh_search.hpp, 3003
 - ns_model.hpp, 3013
 - serialization_template_version.hpp, 2565
- BRNN< OutputLayerType, MergeLayerType, MergeOutputType, InitializationRuleType, CustomLayers >, 610
- BRNN
 - mlpack::ann::BRNN, 612
- Backtrace, 974
 - mlpack::Backtrace, 975
- backtrace
 - mlpack::util::PrefixedOutputStream, 2367
- Backward
 - mlpack::ann::Add, 555
 - mlpack::ann::AddMerge, 561
 - mlpack::ann::AlphaDropout, 569
 - mlpack::ann::AtrousConvolution, 575
 - mlpack::ann::BaseLayer, 590
 - mlpack::ann::BatchNorm, 594
 - mlpack::ann::BilinearInterpolation, 607
 - mlpack::ann::CReLU, 654
 - mlpack::ann::Concat, 625, 626
 - mlpack::ann::ConcatPerformance, 636
 - mlpack::ann::Concatenate, 631
 - mlpack::ann::Constant, 639
 - mlpack::ann::Convolution, 646
 - mlpack::ann::CrossEntropyError, 657
 - mlpack::ann::DiceLoss, 664
 - mlpack::ann::DropConnect, 668
 - mlpack::ann::Dropout, 675
 - mlpack::ann::ELU, 683
 - mlpack::ann::EarthMoverDistance, 679
 - mlpack::ann::FFN, 696
 - mlpack::ann::FastLSTM, 688
 - mlpack::ann::FlexibleReLU, 710
 - mlpack::ann::GRU, 733
 - mlpack::ann::Glimpse, 718
 - mlpack::ann::HardTanH, 742
 - mlpack::ann::Join, 754
 - mlpack::ann::KLDivergence, 760
 - mlpack::ann::LSTM, 803
 - mlpack::ann::LayerNorm, 764
 - mlpack::ann::LeakyReLU, 772
 - mlpack::ann::Linear, 778
 - mlpack::ann::LinearNoBias, 784

- mlpack::ann::LogSoftMax, 793
- mlpack::ann::Lookup, 797
- mlpack::ann::MaxPooling, 810
- mlpack::ann::MeanPooling, 817
- mlpack::ann::MeanSquaredError, 824
- mlpack::ann::MultiplyConstant, 827
- mlpack::ann::MultiplyMerge, 831
- mlpack::ann::NegativeLogLikelihood, 838
- mlpack::ann::PReLU, 858
- mlpack::ann::ReconstructionLoss, 882
- mlpack::ann::Recurrent, 889
- mlpack::ann::RecurrentAttention, 895
- mlpack::ann::ReinforceNormal, 900
- mlpack::ann::Reparametrization, 905
- mlpack::ann::Select, 925
- mlpack::ann::Sequential, 930
- mlpack::ann::SigmoidCrossEntropyError, 938
- mlpack::ann::Subview, 948
- mlpack::ann::TransposedConvolution, 962
- mlpack::ann::VRClassReward, 969
- mlpack::hmm::HMM, 1339
- BackwardVisitor, 587
 - mlpack::ann::BackwardVisitor, 587, 588
- BallBound
 - mlpack::bound::BallBound, 994, 995
- BallBound< MetricType, VecType >, 991
- BallTree
 - mlpack::tree, 451
- Bandwidth
 - mlpack::kde::KDEModel, 1400, 1401
 - mlpack::kernel::GaussianKernel, 1426
 - mlpack::kernel::LaplacianKernel, 1444
 - mlpack::kernel::TriangularKernel, 1462
- Base
 - mlpack::tree::CoverTree, 2049, 2050
- BaseCase
 - mlpack::emst::DTBRules, 1265
 - mlpack::fastmks::FastMKSRules, 1300
 - mlpack::kde::KDERules, 1406
 - mlpack::kmeans::DualTreeKMeansRules, 1471
 - mlpack::kmeans::PellegMooreKMeansRules, 1496
 - mlpack::neighbor::NeighborSearchRules, 1644
 - mlpack::neighbor::RASearchRules, 1694
 - mlpack::range::RangeSearchRules, 1766
- BaseCases
 - mlpack::emst::DTBRules, 1265, 1266
 - mlpack::fastmks::FastMKSRules, 1300
 - mlpack::kde::KDERules, 1406
 - mlpack::kmeans::DualTreeKMeansRules, 1471
 - mlpack::neighbor::NeighborSearch, 1633
 - mlpack::neighbor::NeighborSearchRules, 1644
 - mlpack::range::RangeSearch, 1758
 - mlpack::range::RangeSearchRules, 1766
- baseCases
- mlpack::neighbor::NeighborSearchRules, 1649
- BaseLayer
 - mlpack::ann::BaseLayer, 590
- BaseLayer< ActivationFunction, InputDataType, OutputDataType >, 588
- BasisVector
 - mlpack::tree::CosineTree, 2033
- BatchNorm
 - mlpack::ann::BatchNorm, 594
- BatchNorm< InputDataType, OutputDataType >, 592
- BatchSVDPolicy, 1032
- Begin
 - mlpack::tree::AllDimensionSelect, 1985
 - mlpack::tree::BinarySpaceTree, 2008
 - mlpack::tree::MultipleRandomDimensionSelect, 2159
 - mlpack::tree::RandomDimensionSelect, 2195
 - mlpack::tree::RectangleTree, 2216
- BernoulliDistribution
 - mlpack::ann::BernoulliDistribution, 600, 601
- BernoulliDistribution< DataType >, 599
- BestBinaryNumericSplit< FitnessFunction >, 1989
- BestBinaryNumericSplit< FitnessFunction >::Auxiliary< SplitInfo< ElemType >, 1991
- BestDistance
 - mlpack::neighbor::FurthestNS, 1603
 - mlpack::neighbor::NearestNS, 1622
- BestModel
 - mlpack::hpt::CVFunction, 1364
 - mlpack::hpt::HyperParameterTuner, 1377
- BestNodeToNodeDistance
 - mlpack::neighbor::FurthestNS, 1604
 - mlpack::neighbor::NearestNS, 1623
- BestObjective
 - mlpack::hpt::HyperParameterTuner, 1377
- BestPointToNodeDistance
 - mlpack::neighbor::FurthestNS, 1605
 - mlpack::neighbor::NearestNS, 1624
- Beta
 - mlpack::distribution::GammaDistribution, 1241, 1242
 - mlpack::nn::SparseAutoencoder, 1714
 - mlpack::nn::SparseAutoencoderFunction, 1718
 - mlpack::regression::LARS, 1787
- BetaPath
 - mlpack::regression::LARS, 1787
- BiSearchVisitor, 1748
 - mlpack::neighbor::BiSearchVisitor, 1593
 - mlpack::range::BiSearchVisitor, 1749
- BiSearchVisitor< SortPolicy >, 1591
- BiasSVD< OptimizerType >, 1925
- BiasSVDFunction
 - mlpack::svd::BiasSVDFunction, 1928
- BiasSVDFunction< MatType >, 1926
- BiasSVDPolicy, 1035
 - mlpack::cf::BiasSVDPolicy, 1037

- BiasSVD
 - mlpack::svd::BiasSVD, 1925
- Biases
 - mlpack::perceptron::Perceptron, 1737, 1738
- BilinearInterpolation
 - mlpack::ann::BilinearInterpolation, 606
- BilinearInterpolation< InputDataType, OutputDataType >, 605
- BinLabels
 - mlpack::decision_stump::DecisionStump, 1205
- Binarize
 - mlpack::data, 377
- BinaryDoubleNumericSplit
 - mlpack::tree, 451
- BinaryNumericSplit
 - mlpack::tree::BinaryNumericSplit, 1993, 1994
- BinaryNumericSplit< FitnessFunction, ObservationType >, 1992
- BinaryNumericSplitInfo
 - mlpack::tree::BinaryNumericSplitInfo, 1997
- BinaryNumericSplitInfo< ObservationType >, 1996
- BinaryRBM, 609
- BinarySearch
 - mlpack::tree::CosineTree, 2033
- BinarySpaceTree
 - mlpack::tree::BinarySpaceTree, 2002–2008
- BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >, 1998
- BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::BreadthFirstDualTreeTraverser< RuleType >, 2019
- BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::DualTreeTraverser< RuleType >, 2022
- BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >::SingleTreeTraverser< RuleType >, 2026
- BinaryTree
 - mlpack::tree::TreeTraits, 2303
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, boundType::BallBound, SplitType >, 2306
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, boundType::CellBound, SplitType >, 2308
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, boundType::HollowBallBound, SplitType >, 2310
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, RPTreeMaxSplit >, 2312
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, RPTreeMeanSplit >, 2314
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType >, 2317
 - mlpack::tree::TreeTraits< CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >, 2319
 - mlpack::tree::TreeTraits< Octree< MetricType, StatisticType, MatType >, 2322
 - mlpack::tree::TreeTraits< RectangleTree< MetricType, StatisticType, MatType, RPlusTreeSplit< SplitPolicyType, SweepType >, DescentType, AuxiliaryInformationType >, 2324
 - mlpack::tree::TreeTraits< RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >, 2327
 - mlpack::tree::TreeTraits< SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType >, 2329
- BindingInfo, 984
- bindings/CMakeLists.txt
 - add_subdirectory, 2384
 - set, 2385
- bindings/cli/CMakeLists.txt
 - set, 2384
- bindings/markdown/CMakeLists.txt
 - macro, 2385, 2386
 - set, 2386
- bindings/python/CMakeLists.txt
 - else, 2387
 - endif, 2387
 - find_python_module, 2387
 - macro, 2387
 - set, 2387
- bindings/tests/CMakeLists.txt
 - set, 2388
- Bins
 - mlpack::tree::HoeffdingNumericSplit, 2111
- Blacklist
 - mlpack::kmeans::PellegMooreKMeansStatistic, 1499, 1500
- BlockHeight
 - mlpack::math::ColumnsToBlocks, 1544
- BlockSize
 - mlpack::pca::RandomizedBlockKrylovSVDPolicy, 1731
 - mlpack::svd::RandomizedBlockKrylovSVD, 1936
- BlockWidth
 - mlpack::math::ColumnsToBlocks, 1545
- boost, 251
- boost::serialization, 251
- boost::serialization::version< mlpack::adaboost::AdaBoost< WeakLearnerType, MatType >, BOOST_STATIC_CONSTANT, 477

- boost::serialization::version< mlpack::ann::BRNN<
 - OutputLayerType, MergeLayerType, Merge↔
 - OutputType, InitializationRuleType, Custom↔
 - Layer... > >
 BOOST_STATIC_CONSTANT, 478
- boost::serialization::version< mlpack::ann::FFN< Output↔
 - LayerType, InitializationRuleType, Custom↔
 - Layer... > >
 BOOST_STATIC_CONSTANT, 479
- boost::serialization::version< mlpack::ann::RNN<
 - OutputLayerType, InitializationRuleType,
 - CustomLayer... > >
 BOOST_STATIC_CONSTANT, 479
- Bootstrap
 - mlpack::tree, 463
- Bound
 - mlpack::emst::DTBStat, 1270
 - mlpack::fastmks::FastMKStat, 1304, 1305
 - mlpack::neighbor::RAQueryStat, 1679, 1680
 - mlpack::tree::BinarySpaceTree, 2008, 2009
 - mlpack::tree::Octree, 2176
 - mlpack::tree::RPlusPlusTreeSplitPolicy, 2248
 - mlpack::tree::RPlusTreeSplitPolicy, 2254
 - mlpack::tree::RectangleTree, 2216, 2217
 - mlpack::tree::SpillTree, 2283
- BoundTraits< BallBound< MetricType, VecType > >, 1003
- BoundTraits< BoundType >, 1002
- BoundTraits< CellBound< MetricType, ElemType > >, 1003
- BoundTraits< HRectBound< MetricType, ElemType > >, 1005
- BoundTraits< HollowBallBound< MetricType, ElemType > >, 1004
- BoundType
 - mlpack::tree::HyperplaneBase, 2134
 - mlpack::tree::RPlusPlusTreeAuxiliaryInformation, 2240
 - mlpack::tree::SpillTree, 2278
- BreadthFirstDualTreeTraverser
 - mlpack::tree::BinarySpaceTree::BreadthFirstDual↔
 - TreeTraverser, 2020
 - mlpack::tree::CoverTree, 2044
- BucketSize
 - mlpack::neighbor::LSHSearch, 1612
- BucketTag
 - mlpack::det::DTree, 1214
- BufSize
 - mlpack::math::ColumnsToBlocks, 1545
- BufValue
 - mlpack::math::ColumnsToBlocks, 1546
- BuildModel
 - mlpack::fastmks::FastMKModel, 1295
 - mlpack::kde::KDEModel, 1401
 - mlpack::neighbor::NSModel, 1661
 - mlpack::neighbor::RAModel, 1673
 - mlpack::range::RModel, 1775
 - mlpack::tree::HoeffdingTreeModel, 2130
- BuildString
 - mlpack::det::PathCacher, 1224
- CFModel, 1042
 - mlpack::cf::CFModel, 1043
- CFPtr
 - mlpack::cf::CFModel, 1043
- CFTType
 - mlpack::cf::CFTType, 1047
- CFTType< DecompositionPolicy, NormalizationType >, 1045
- CLIOption
 - mlpack::bindings::cli::CLIOption, 976
- CLIOption< N >, 975
- CLI, 1117
- CMakeLists.txt
 - include_directories, 2389
- CReLU< InputDataType, OutputDataType >, 652
- CReLU
 - mlpack::ann::CReLU, 654
- CVBase
 - mlpack::cv::CVBase, 1130, 1131
- CVBase< MLAlgorithm, MatType, PredictionsType, WeightsType >, 1129
- CVFunction
 - mlpack::hpt::CVFunction, 1362
- CVFunction< CVType, MLAlgorithm, TotalArgs, Bound↔
- Args >, 1361
- CalculateBound
 - mlpack::neighbor::NeighborSearchRules, 1645
- CalculateCentroid
 - mlpack::tree::CosineTree, 2034
- CalculateCosines
 - mlpack::tree::CosineTree, 2034
- CalculateDirection
 - mlpack::tree::AllCategoricalSplit, 1982
 - mlpack::tree::BestBinaryNumericSplit, 1990
 - mlpack::tree::BinaryNumericSplitInfo, 1997
 - mlpack::tree::CategoricalSplitInfo, 2028
 - mlpack::tree::DecisionTree, 2072
 - mlpack::tree::HoeffdingTree, 2119
 - mlpack::tree::NumericSplitInfo, 2166
- Candidate
 - mlpack::neighbor::NeighborSearchRules, 1643
- CandidateIndices
 - mlpack::neighbor::DrusillaSelect, 1598
- CandidateList
 - mlpack::neighbor::NeighborSearchRules, 1643
- CandidateSet
 - mlpack::neighbor::DrusillaSelect, 1598, 1599

- mlpack::neighbor::QDAFN, 1667
- candidates
 - mlpack::neighbor::NeighborSearchRules, 1649
- CartPole, 1842
 - mlpack::rl::CartPole, 1843
- CartPole::State, 1846
- CategoricalSplit
 - mlpack::tree::DecisionTree, 2067
 - mlpack::tree::HoeffdingTree, 2116
- CategoricalSplitInfo, 2028
 - mlpack::tree::CategoricalSplitInfo, 2028
- CauchyKernel, 1414
 - mlpack::kernel::CauchyKernel, 1415
- CellBound< MetricType, ElemType >, 1006
- Center
 - mlpack::bound::BallBound, 995, 996
 - mlpack::bound::HRectBound, 1022
 - mlpack::bound::HollowBallBound, 1010, 1011
 - mlpack::math, 410
 - mlpack::tree::BinarySpaceTree, 2009
 - mlpack::tree::CoverTree, 2050
 - mlpack::tree::Octree, 2176
 - mlpack::tree::RectangleTree, 2217
 - mlpack::tree::SpillTree, 2284
- center
 - mlpack::tree::Octree::SplitType::SplitInfo, 2190
- CenterTransformedData
 - mlpack::kpca::KernelPCA, 1510
- Centroid
 - mlpack::kde::KDEStat, 1409
 - mlpack::kmeans::DualTreeKMeansStatistic, 1474
 - mlpack::kmeans::PellegMooreKMeansStatistic, 1500
 - mlpack::tree::CosineTree, 2034
 - mlpack::tree::ExampleTree, 2084
- ChebyshevDistance
 - mlpack::metric, 427
- Check
 - mlpack::hpt::DeduceHyperParameterTypes< Args... >::IsCollectionType, 1370
- CheckGradient
 - ann_test_tools.hpp, 3078
- CheckInterval
 - mlpack::tree::HoeffdingTree, 2119
- CheckLeafSweep
 - mlpack::tree::MinimalCoverageSweep, 2152
- CheckMatrices
 - mlpack, 254, 255
 - range_search_utils.hpp, 3083
 - test_tools.hpp, 3088, 3089
- CheckMatricesNotEqual
 - test_tools.hpp, 3089, 3090
- CheckNonLeafSweep
 - mlpack::tree::MinimalCoverageSweep, 2153
- Child
 - mlpack::det::DTree, 1214
 - mlpack::tree::BinarySpaceTree, 2009
 - mlpack::tree::CoverTree, 2050
 - mlpack::tree::DecisionTree, 2072, 2073
 - mlpack::tree::ExampleTree, 2084
 - mlpack::tree::HoeffdingTree, 2120
 - mlpack::tree::Octree, 2176
 - mlpack::tree::RectangleTree, 2217, 2218
 - mlpack::tree::SpillTree, 2284
- ChildPtr
 - mlpack::det::DTree, 1214
 - mlpack::tree::BinarySpaceTree, 2010
 - mlpack::tree::CoverTree, 2051
 - mlpack::tree::Octree, 2177
 - mlpack::tree::SpillTree, 2284
- Children
 - mlpack::tree::CoverTree, 2051
 - mlpack::tree::HilbertRTreeAuxiliaryInformation, 2097
- chk
 - mlpack::data::HasSerialize, 1179
- ChooseDescentNode
 - mlpack::tree::HilbertRTreeDescentHeuristic, 2101, 2102
 - mlpack::tree::RPlusPlusTreeDescentHeuristic, 2246, 2247
 - mlpack::tree::RPlusTreeDescentHeuristic, 2250, 2251
 - mlpack::tree::RStarTreeDescentHeuristic, 2267
 - mlpack::tree::RTreeDescentHeuristic, 2270
- ChooseRoot
 - mlpack::tree::FirstPointsRoot, 2090
- ClampNonNegative
 - mlpack::math, 411
- ClampNonPositive
 - mlpack::math, 411
- ClampRange
 - mlpack::math, 411
- T, Classify
 - mlpack::adaboost::AdaBoost, 491
 - mlpack::adaboost::AdaBoostModel, 496
 - mlpack::decision_stump::DecisionStump, 1205
 - mlpack::gmm::DiagonalGMM, 1311
 - mlpack::gmm::GMM, 1327
 - mlpack::naive_bayes::NaiveBayesClassifier, 1579, 1580
 - mlpack::perceptron::Perceptron, 1738
 - mlpack::regression::LogisticRegression, 1798, 1799
 - mlpack::regression::SoftmaxRegression, 1813–1815
 - mlpack::svm::LinearSVM, 1962, 1963
 - mlpack::tree::DecisionTree, 2073, 2074
 - mlpack::tree::HoeffdingTree, 2120, 2121
 - mlpack::tree::HoeffdingTreeModel, 2131
 - mlpack::tree::RandomForest, 2201–2203
- CleanData

- mlpack::cf::CFTYPE, 1048
- mlpack::svd::SVDPlusPlus, 1953
- CleanMemory
 - mlpack::bindings::tests, 359
- CleanedData
 - mlpack::cf::CFTYPE, 1048
- Clear
 - mlpack::bound::HRectBound, 1022
- ClearAlias
 - mlpack::math, 412
- ClearSettings
 - mlpack::CLI, 1122
- cli_option.hpp
 - BASH_CLEAR, 2383
 - BASH_RED, 2383
- Cluster
 - mlpack::dbscan::DBSCAN, 1199, 1200
 - mlpack::kmeans::KMeans, 1485, 1486
 - mlpack::kmeans::RandomPartition, 1501
 - mlpack::kmeans::RefinedStart, 1503, 1504
 - mlpack::kmeans::SampleInitialization, 1506
 - mlpack::meanshift::MeanShift, 1563
- Clusterer
 - mlpack::gmm::EMFit, 1320
- Cols
 - mlpack::math::ColumnsToBlocks, 1546
- ColumnCovariance
 - mlpack::math, 412, 413
- ColumnSampleLS
 - mlpack::tree::CosineTree, 2034
- ColumnSamplesLS
 - mlpack::tree::CosineTree, 2035
- ColumnsToBlocks, 1541
 - mlpack::math::ColumnsToBlocks, 1544
- CombineBest
 - mlpack::neighbor::FurthestNS, 1605
 - mlpack::neighbor::NearestNS, 1624
- CombineWorst
 - mlpack::neighbor::FurthestNS, 1605
 - mlpack::neighbor::NearestNS, 1624
- CombinedNormalization
 - mlpack::cf::CombinedNormalization, 1054
- CombinedNormalization< NormalizationTypes >, 1053
- CompareAddresses
 - mlpack::bound::addr, 367
- CompareCosineNode, 2029
- CompleteIncrementalTermination
 - mlpack::amf::CompleteIncrementalTermination, 505
- CompleteIncrementalTermination< TerminationPolicy >, 504
- Component
 - mlpack::gmm::DiagonalGMM, 1312
 - mlpack::gmm::GMM, 1327, 1328
- ComponentMembership
 - mlpack::emst::DTBStat, 1271
- ComputeAccuracy
 - mlpack::regression::LogisticRegression, 1800
 - mlpack::regression::SoftmaxRegression, 1815
 - mlpack::svm::LinearSVM, 1964
- ComputeError
 - mlpack::regression::LinearRegression, 1791
 - mlpack::regression::LogisticRegression, 1800
- ComputeMST
 - mlpack::emst::DualTreeBoruvka, 1275
- ComputeRecall
 - mlpack::neighbor::LSHSearch, 1613
- ComputeValue
 - mlpack::det::DTree, 1214
- ComputeVariableImportance
 - mlpack::det::DTree, 1215
- Concat
 - mlpack::ann::Concat, 623
 - mlpack::ann::Concatenate, 632
- Concat< InputDataType, OutputDataType, CustomLayers >, 621
- ConcatPerformance
 - mlpack::ann::ConcatPerformance, 635
- ConcatPerformance< OutputLayerType, InputDataType, OutputDataType >, 634
- Concatenate
 - mlpack::ann::Concatenate, 631
- Concatenate< InputDataType, OutputDataType >, 630
- CondenseTree
 - mlpack::tree::RectangleTree, 2218
- Config
 - mlpack::rl::AsyncLearning, 1839
- ConfusionMatrix
 - mlpack::data, 378
- ConstInitialization, 641
 - mlpack::ann::ConstInitialization, 641
- Constant
 - mlpack::ann::Constant, 638
- Constant< InputDataType, OutputDataType >, 638
- Constraint
 - mlpack::gmm::EMFit, 1321
- Constraints
 - mlpack::lmnn::Constraints, 1521
- Constraints< MetricType >, 1520
- ConstructBasis
 - mlpack::tree::CosineTree, 2035
- Contains
 - mlpack::bound::BallBound, 996
 - mlpack::bound::HRectBound, 1022
 - mlpack::bound::HollowBallBound, 1011, 1012
 - mlpack::bound::addr, 367
 - mlpack::math::RangeType, 1551, 1552
- ContinuousMountainCar, 1850
 - mlpack::rl::ContinuousMountainCar, 1851

- ContinuousMountainCar::Action, 1854
- ContinuousMountainCar::State, 1855
- ConvertToDistance
 - mlpack::neighbor::FurthestNS, 1606
 - mlpack::neighbor::NearestNS, 1625
- ConvertToScore
 - mlpack::neighbor::FurthestNS, 1606
 - mlpack::neighbor::NearestNS, 1625
- Convolution
 - mlpack::ann::Convolution, 645
 - mlpack::ann::FFTConvolution, 706, 707
 - mlpack::ann::NaiveConvolution, 835, 836
 - mlpack::ann::SVDConvolution, 951, 952
- Convolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, InputDataType, OutputDataType >, 643
- ConvolutionIntegral
 - mlpack::kernel::EpanechnikovKernel, 1418
 - mlpack::kernel::ExampleKernel, 1422
 - mlpack::kernel::GaussianKernel, 1426
 - mlpack::kernel::SphericalKernel, 1459
- CopyAndPerturb
 - mlpack::radical::Radical, 1745
- CopyTask, 582
 - mlpack::ann::augmented::tasks::CopyTask, 583
- CopyVisitor< CustomLayers >, 651
- core/CMakeLists.txt
 - add_subdirectory, 2389
 - set, 2389
- core/cv/CMakeLists.txt
 - add_subdirectory, 2390
 - set, 2390
- core/cv/metrics/CMakeLists.txt
 - set, 2391
- core/data/CMakeLists.txt
 - set, 2391, 2392
- core/data/imputation_methods/CMakeLists.txt
 - set, 2392
- core/data/map_policies/CMakeLists.txt
 - set, 2393
- core/dists/CMakeLists.txt
 - set, 2394
- core/hpt/CMakeLists.txt
 - set, 2394
- core/kernels/CMakeLists.txt
 - set, 2395
- core/math/CMakeLists.txt
 - set, 2396
- core/metrics/CMakeLists.txt
 - set, 2397
- core/tree/CMakeLists.txt
 - set, 2398, 2399
- core/util/CMakeLists.txt
 - set, 2400
- CosineDistance, 1416
- CosineNodeQueue
 - mlpack::tree, 451
- CosineNodeSplit
 - mlpack::tree::CosineTree, 2035
- CosineSearch, 1056
 - mlpack::cf::CosineSearch, 1057
- CosineTree, 2030
 - mlpack::tree::CosineTree, 2031, 2032
- Count
 - mlpack::tree::BinarySpaceTree, 2010
 - mlpack::tree::RectangleTree, 2218, 2219
- Counts
 - mlpack::kernel::PSpectrumStringKernel, 1456
- cout
 - mlpack::Log, 1540
- Covariance
 - mlpack::distribution::DiagonalGaussianDistribution, 1228
 - mlpack::distribution::GaussianDistribution, 1247, 1248
 - mlpack::metric::MahalanobisDistance, 1574, 1575
- CoverTree
 - mlpack::tree::CoverTree, 2045–2049
- CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >, 2040
- CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::DualTreeTraverser< RuleType >, 2060
- CoverTree< MetricType, StatisticType, MatType, RootPointPolicy >::SingleTreeTraverser< RuleType >, 2063
- CoverTreeDualTreeKMeans
 - mlpack::kmeans, 404
- cppType
 - mlpack::util::ParamData, 2358
- Create
 - InitHMMModel, 480, 481
- CreateChildren
 - mlpack::tree::HoeffdingTree, 2122
- CrossEntropyError
 - mlpack::ann::CrossEntropyError, 657
- CrossEntropyError< InputDataType, OutputDataType >, 656
- CustomImputation
 - mlpack::data::CustomImputation, 1171
- CustomImputation< T >, 1170
- CustomLayer
 - mlpack::ann, 268
- d
 - mlpack::tree::Octree::SplitType::SplitInfo, 2190
- DBSCAN< RangeSearchType, PointSelectionPolicy >, 1197

- DBSCAN
 - mlpack::dbscan::DBSCAN, 1198
- DTBRules
 - mlpack::emst::DTBRules, 1265
- DTBRules< MetricType, TreeType >, 1264
- DTBStat, 1269
 - mlpack::emst::DTBStat, 1269, 1270
- DTree
 - mlpack::det::DTree, 1211–1213
- DTree< MatType, TagType >, 1207
- Data
 - mlpack::rl::Acrobot::State, 1834
 - mlpack::rl::CartPole::State, 1848
 - mlpack::rl::ContinuousMountainCar::State, 1856
 - mlpack::rl::MountainCar::State, 1866
 - mlpack::rl::Pendulum::State, 1884
- DataDependentRandomInitializer, 1913
- Dataset
 - mlpack::lmnn::LMNNFunction, 1533
 - mlpack::lmnn::LMNN, 1528
 - mlpack::nca::NCA, 1585
 - mlpack::neighbor::NSModel, 1661
 - mlpack::neighbor::RAModel, 1673
 - mlpack::range::RSModel, 1775
 - mlpack::svd::BiasSVDFunction, 1928
 - mlpack::svd::RegularizedSVDFunction, 1946
 - mlpack::svd::SVDPlusPlusFunction, 1955
 - mlpack::svm::LinearSVMFunction, 1969
 - mlpack::tree::BinarySpaceTree, 2010, 2011
 - mlpack::tree::CoverTree, 2051
 - mlpack::tree::Octree, 2177
 - mlpack::tree::RectangleTree, 2219
 - mlpack::tree::SpillTree, 2285
- DatasetInfo
 - mlpack::data, 376
- DatasetMapper
 - mlpack::data::DatasetMapper, 1173, 1174
- DatasetMapper< PolicyType, InputType >, 1172
- Datatype
 - mlpack::data, 376
- Debug
 - mlpack::Log, 1540
- DecisionStump
 - mlpack::decision_stump::DecisionStump, 1203, 1204
 - mlpack::tree, 452
- DecisionStump< MatType >, 1202
- DecisionTree
 - mlpack::tree::DecisionTree, 2068–2071
- DecisionTree< FitnessFunction, NumericSplitType, CategoricalSplitType, DimensionSelectionType, ElemType, NoRecursion >, 2065
- DecisionTreeType
 - mlpack::tree::RandomForest, 2198
- Decomposition
 - mlpack::cf::CFTYPE, 1048
- DeduceHyperParameterTypes< Args >, 1365
- DeduceHyperParameterTypes< Args >::ResultHolder< HPTypes >, 1366
- DeduceHyperParameterTypes< PreFixedArg< T >, Args... >, 1366
- DeduceHyperParameterTypes< PreFixedArg< T >, Args... >::ResultHolder< HPTypes >, 1367
- DeduceHyperParameterTypes< T, Args... >, 1368
- DeduceHyperParameterTypes< T, Args... >::IsCollectionType< Type >, 1369
- DeduceHyperParameterTypes< T, Args... >::ResultHPType< ArgumentType, IsArithmetic >, 1372
- DeduceHyperParameterTypes< T, Args... >::ResultHPType< ArithmeticType, true >, 1372
- DeduceHyperParameterTypes< T, Args... >::ResultHPType< CollectionType, false >, 1373
- DeduceHyperParameterTypes< T, Args... >::ResultHolder< HPTypes >, 1371
- DefaultDualTreeKMeans
 - mlpack::kmeans, 404
- DefaultParam
 - mlpack::bindings::cli, 286
 - mlpack::bindings::markdown, 314
 - mlpack::bindings::python, 331
- DefaultParamImpl
 - mlpack::bindings::cli, 286, 287
 - mlpack::bindings::python, 331, 332
- DefeatistDualTreeTraverser
 - mlpack::tree::SpillTree, 2278
- DefeatistKNN
 - mlpack::neighbor, 431
- DefeatistSingleTreeTraverser
 - mlpack::tree::SpillTree, 2278
- Degree
 - mlpack::kernel::PolynomialKernel, 1452
- DeleteAllocatedMemory
 - mlpack::bindings::cli, 287
 - mlpack::bindings::tests, 359
- DeleteAllocatedMemoryImpl
 - mlpack::bindings::cli, 287, 288
 - mlpack::bindings::tests, 359, 360
- DeleteModules
 - mlpack::ann::Sequential, 931
- DeletePoint
 - mlpack::tree::RectangleTree, 2219
- DeleteVisitor, 659, 1058, 1383, 1595, 1751
- Delta
 - mlpack::ann::Add, 555, 556
 - mlpack::ann::AddMerge, 562
 - mlpack::ann::AlphaDropout, 570
 - mlpack::ann::AtrousConvolution, 576
 - mlpack::ann::BaseLayer, 590, 591
 - mlpack::ann::BatchNorm, 596

- mlpack::ann::BilinearInterpolation, 607
- mlpack::ann::CReLU, 654
- mlpack::ann::Concat, 626
- mlpack::ann::ConcatPerformance, 636
- mlpack::ann::Concatenate, 632
- mlpack::ann::Constant, 639
- mlpack::ann::Convolution, 646
- mlpack::ann::DropConnect, 669
- mlpack::ann::Dropout, 675
- mlpack::ann::ELU, 683, 684
- mlpack::ann::FastLSTM, 689
- mlpack::ann::FlexibleReLU, 710
- mlpack::ann::GRU, 734
- mlpack::ann::Glimpse, 719
- mlpack::ann::HardTanH, 742
- mlpack::ann::Join, 754, 755
- mlpack::ann::LSTM, 804
- mlpack::ann::LayerNorm, 765
- mlpack::ann::LeakyReLU, 773
- mlpack::ann::Linear, 779
- mlpack::ann::LinearNoBias, 784
- mlpack::ann::LogSoftMax, 794
- mlpack::ann::Lookup, 797, 798
- mlpack::ann::MaxPooling, 810
- mlpack::ann::MeanPooling, 818
- mlpack::ann::MultiplyConstant, 827
- mlpack::ann::MultiplyMerge, 831, 832
- mlpack::ann::NegativeLogLikelihood, 839
- mlpack::ann::PReLU, 858
- mlpack::ann::Recurrent, 889
- mlpack::ann::RecurrentAttention, 895
- mlpack::ann::ReinforceNormal, 900, 901
- mlpack::ann::Reparametrization, 905
- mlpack::ann::Select, 926
- mlpack::ann::Sequential, 931
- mlpack::ann::Subview, 949
- mlpack::ann::TransposedConvolution, 962
- mlpack::ann::VRClassReward, 970
- mlpack::pca::QUICSVDPolicy, 1729
- DeltaVisitor, 660
- Denormalize
 - mlpack::cf::CombinedNormalization, 1054, 1055
 - mlpack::cf::ItemMeanNormalization, 1062
 - mlpack::cf::NoNormalization, 1070, 1071
 - mlpack::cf::OverallMeanNormalization, 1073
 - mlpack::cf::UserMeanNormalization, 1112
 - mlpack::cf::ZScoreNormalization, 1115
- deprecated.hpp
 - mlpack_deprecated, 2695
- Deriv
 - mlpack::ann::HardSigmoidFunction, 738
 - mlpack::ann::IdentityFunction, 747, 748
 - mlpack::ann::LogisticFunction, 790
 - mlpack::ann::RectifierFunction, 884
 - mlpack::ann::SoftplusFunction, 940, 941
 - mlpack::ann::SoftsignFunction, 944
 - mlpack::ann::SwishFunction, 953, 954
 - mlpack::ann::TanhFunction, 956, 957
- desc
 - mlpack::util::ParamData, 2358
- Descendant
 - mlpack::tree::BinarySpaceTree, 2011
 - mlpack::tree::CoverTree, 2052
 - mlpack::tree::ExampleTree, 2084
 - mlpack::tree::Octree, 2177
 - mlpack::tree::RectangleTree, 2220
 - mlpack::tree::SpillTree, 2285
- DescentType
 - mlpack::tree::RectangleTree, 2233
- destination
 - mlpack::util::PrefixedOutputStream, 2368
- det.txt
 - \$V, 2377
 - alpha, 2377
 - estimation, 2377
 - now, 2377
 - regularization, 2378
 - Thus, 2378
- Deterministic
 - mlpack::ann::AlphaDropout, 570
 - mlpack::ann::BatchNorm, 596
 - mlpack::ann::DropConnect, 669
 - mlpack::ann::Dropout, 675
 - mlpack::ann::GRU, 734
 - mlpack::ann::Glimpse, 719
 - mlpack::ann::MaxPooling, 810, 811
 - mlpack::ann::MeanPooling, 818
 - mlpack::ann::Recurrent, 889, 890
 - mlpack::ann::RecurrentAttention, 895, 896
 - mlpack::ann::ReinforceNormal, 901
 - mlpack::ann::VRClassReward, 970
 - mlpack::rl::QLearning, 1888
- DeterministicSetVisitor, 661
 - mlpack::ann::DeterministicSetVisitor, 662
- DiagGMMHMM
 - mlpack::hmm::HMMModel, 1352
- DiagonalConstraint, 1306
- DiagonalGMM, 1308
 - mlpack::gmm::DiagonalGMM, 1310, 1311
- DiagonalGaussianDistribution, 1226
 - mlpack::distribution::DiagonalGaussianDistribution, 1227, 1228
- Diameter
 - mlpack::bound::BallBound, 997
 - mlpack::bound::HRectBound, 1023
 - mlpack::bound::HollowBallBound, 1012
- DiceLoss
 - mlpack::ann::DiceLoss, 663

- DiceLoss< InputDataType, OutputDataType >, 662
- Dictionary
 - mlpack::lcc::LocalCoordinateCoding, 1516
 - mlpack::sparse_coding::SparseCoding, 1920
- didParse
 - mlpack::CLI, 1126
- Dim
 - mlpack::bound::BallBound, 997
 - mlpack::bound::HRectBound, 1023
 - mlpack::bound::HollowBallBound, 1012
- dimension
 - mlpack::rl::Acrobot::State, 1835
 - mlpack::rl::CartPole::State, 1850
 - mlpack::rl::ContinuousMountainCar::State, 1858
 - mlpack::rl::MountainCar::State, 1868
 - mlpack::rl::Pendulum::State, 1885
- DimensionSelection
 - mlpack::tree::DecisionTree, 2068
- Dimensionality
 - mlpack::adaboost::AdaBoostModel, 496, 497
 - mlpack::data::DatasetMapper, 1174
 - mlpack::distribution::DiagonalGaussianDistribution, 1228
 - mlpack::distribution::DiscreteDistribution, 1234
 - mlpack::distribution::GammaDistribution, 1242
 - mlpack::distribution::GaussianDistribution, 1248
 - mlpack::distribution::LaplaceDistribution, 1254
 - mlpack::distribution::RegressionDistribution, 1260
 - mlpack::gmm::DiagonalGMM, 1312
 - mlpack::gmm::GMM, 1328
 - mlpack::hmm::HMM, 1341
- Dimensions
 - mlpack::tree::AllDimensionSelect, 1985
 - mlpack::tree::MultipleRandomDimensionSelect, 2159
 - mlpack::tree::RandomDimensionSelect, 2195
- direction
 - mlpack::tree::RPTreeMaxSplit::SplitInfo, 2260
 - mlpack::tree::RPTreeMeanSplit::SplitInfo, 2265
- DisableBacktrace
 - mlpack::util, 468
- DisableTiming
 - mlpack::Timer, 1975
- DisableVerbose
 - mlpack::util, 468
- Discount
 - mlpack::rl::TrainingConfig, 1902
- DiscreteDistribution, 1232
 - mlpack::distribution::DiscreteDistribution, 1233, 1234
- DiscreteHMM
 - mlpack::hmm::HMMModel, 1352
- DiscreteHilbertRTreeAuxiliaryInformation
 - mlpack::tree, 452
- DiscreteHilbertValue< TreeElemType >, 2079
- Distance
 - mlpack::emst::EdgePair, 1276, 1277
- DistanceCalculations
 - mlpack::kmeans::DualTreeKMeans, 1468
 - mlpack::kmeans::ElkanKMeans, 1479
 - mlpack::kmeans::HamerlyKMeans, 1480
 - mlpack::kmeans::NaiveKMeans, 1492
 - mlpack::kmeans::PellegMooreKMeans, 1494
 - mlpack::kmeans::PellegMooreKMeansRules, 1497
- DistanceComps
 - mlpack::tree::CoverTree, 2052
- DistanceEvaluations
 - mlpack::neighbor::LSHSearch, 1613
- DoRadical
 - mlpack::radical::Radical, 1746
- DoRadical2D
 - mlpack::radical::Radical, 1746
- doc
 - mlpack::CLI, 1126
- documentation
 - mlpack::bindings::cli::ProgramDoc, 983
 - mlpack::bindings::tests::ProgramDoc, 989
 - mlpack::util::ProgramDoc, 2370
- DoubleQLearning
 - mlpack::rl::TrainingConfig, 1903
- DropConnect
 - mlpack::ann::DropConnect, 668
- DropConnect< InputDataType, OutputDataType >, 666
- Dropout
 - mlpack::ann::Dropout, 674
- Dropout< InputDataType, OutputDataType >, 673
- DrusillaSelect
 - mlpack::neighbor::DrusillaSelect, 1597, 1598
- DrusillaSelect< MatType >, 1597
- Dsdt
 - mlpack::rl::Acrobot, 1827
- DualBiKDE, 1384
 - mlpack::kde::DualBiKDE, 1385
- DualMonoKDE, 1386
 - mlpack::kde::DualMonoKDE, 1387
- DualTreeBoruvka
 - mlpack::emst::DualTreeBoruvka, 1274
- DualTreeBoruvka< MetricType, MatType, TreeType >, 1272
- DualTreeKMeans
 - mlpack::kmeans::DualTreeKMeans, 1468
- DualTreeKMeans< MetricType, MatType, TreeType >, 1466
- DualTreeKMeansRules
 - mlpack::kmeans::DualTreeKMeansRules, 1470
- DualTreeKMeansRules< MetricType, TreeType >, 1469
- DualTreeKMeansStatistic, 1473
 - mlpack::kmeans::DualTreeKMeansStatistic, 1474
- DualTreeTraverser

- mlpack::tree::BinarySpaceTree::DualTreeTraverser, 2023
- mlpack::tree::CoverTree::DualTreeTraverser, 2061
- mlpack::tree::Octree::DualTreeTraverser, 2185
- mlpack::tree::RectangleTree::DualTreeTraverser, 2235
- mlpack::tree::SpillTree, 2278
- DummyClass, 1059
- ELU< InputDataType, OutputDataType >, 680
- ELU
 - mlpack::ann::ELU, 682
- EMFit
 - mlpack::gmm::EMFit, 1320
- EMFit< InitialClusteringType, CovarianceConstraintPolicy, Distribution >, 1318
- EarthMoverDistance
 - mlpack::ann::EarthMoverDistance, 678
- EarthMoverDistance< InputDataType, OutputDataType >, 677
- EdgePair, 1275
 - mlpack::emst::EdgePair, 1276
- EffectiveError
 - mlpack::neighbor::NeighborSearch, 1633
- EigenvalueRatioConstraint, 1317
 - mlpack::gmm::EigenvalueRatioConstraint, 1317
- ElemType
 - mlpack::ann::FastLSTM, 687
 - mlpack::ann::RBM, 868
 - mlpack::bound::BallBound, 993
 - mlpack::det::DTree, 1210
 - mlpack::naive_bayes::NaiveBayesClassifier, 1578
 - mlpack::tree::BinarySpaceTree, 2002
 - mlpack::tree::CoverTree, 2044
 - mlpack::tree::HilbertRTreeAuxiliaryInformation, 2096
 - mlpack::tree::Octree, 2170
 - mlpack::tree::RPTreeMaxSplit, 2256
 - mlpack::tree::RPTreeMeanSplit, 2262
 - mlpack::tree::RPlusPlusTreeAuxiliaryInformation, 2240
 - mlpack::tree::RectangleTree, 2212
 - mlpack::tree::SpillTree, 2279
 - mlpack::tree::VantagePointSplit, 2332
- ElkanKMeans
 - mlpack::kmeans::ElkanKMeans, 1478
- ElkanKMeans< MetricType, MatType >, 1478
- else
 - bindings/python/CMakeLists.txt, 2387
- Embedding
 - mlpack::ann, 269
- Emission
 - mlpack::hmm::HMM, 1341
- emission
 - mlpack::hmm::HMM, 1349
- EmptyCluster
 - mlpack::kmeans::AllowEmptyClusters, 1465
 - mlpack::kmeans::KillEmptyClusters, 1482
 - mlpack::kmeans::MaxVarianceNewCluster, 1490
- EmptyClusterAction
 - mlpack::kmeans::KMeans, 1487
- EmptyStatistic, 2080
 - mlpack::tree::EmptyStatistic, 2080, 2081
- enable_if_t
 - std, 476
- EnableTimers
 - mlpack::util, 468
- EnableTiming
 - mlpack::Timer, 1975
- EnableVerbose
 - mlpack::util, 468
- Enabled
 - mlpack::Timers, 1978
- Encode
 - mlpack::lcc::LocalCoordinateCoding, 1517
 - mlpack::rl::Acrobot::State, 1834
 - mlpack::rl::CartPole::State, 1849
 - mlpack::rl::ContinuousMountainCar::State, 1857
 - mlpack::rl::MountainCar::State, 1867
 - mlpack::rl::Pendulum::State, 1884
 - mlpack::sparse_coding::SparseCoding, 1921
- End
 - mlpack::det::DTree, 1215
 - mlpack::tree::AllDimensionSelect, 1986
 - mlpack::tree::MultipleRandomDimensionSelect, 2159
 - mlpack::tree::RandomDimensionSelect, 2195
- EndProgram
 - mlpack::bindings::cli, 288
- endif
 - bindings/python/CMakeLists.txt, 2387
- ens, 252
- Enter
 - mlpack::det::PathCacher, 1224
- EnumerateTree
 - mlpack::tree, 463
- EnumerateTreeImpl
 - mlpack::tree::enumerate, 464
- Environment
 - mlpack::rl::AsyncLearning, 1840
 - mlpack::rl::QLearning, 1888
 - mlpack::rl::RewardClipping, 1897
- EpanechnikovKernel, 1417
 - mlpack::kernel::EpanechnikovKernel, 1418
- Episode
 - mlpack::rl::QLearning, 1888
- Eps
 - mlpack::ann::CrossEntropyError, 657, 658
- Epsilon
 - mlpack::neighbor::NSModel, 1661

- mlpack::neighbor::NeighborSearch, 1634
- mlpack::pca::QUICSVDPolicy, 1729
- mlpack::rl::GreedyPolicy, 1860
- mlpack::svd::RandomizedSVD, 1941, 1942
- epsilon
 - mlpack::neighbor::NeighborSearchRules, 1649
- EpsilonVisitor, 1600
- Err
 - mlpack::distribution::RegressionDistribution, 1260
- Estimate
 - mlpack::distribution::LaplaceDistribution, 1254
 - mlpack::gmm::EMFit, 1321, 1322
 - mlpack::hmm::HMMRegression, 1357, 1358
 - mlpack::hmm::HMM, 1341, 1342
- EstimateRadius
 - mlpack::meanshift::MeanShift, 1563
- estimation
 - det.txt, 2377
- EuclideanDistance
 - mlpack::metric, 427
- EuclideanSearch
 - mlpack::cf, 370
- Evaluate
 - mlpack::ann::BRNN, 613
 - mlpack::ann::FFN, 697, 698
 - mlpack::ann::RBM, 869
 - mlpack::ann::RNN, 915
 - mlpack::cv::Accuracy, 1128
 - mlpack::cv::F1, 1133
 - mlpack::cv::KFoldCV, 1139
 - mlpack::cv::MSE, 1143
 - mlpack::cv::Precision, 1146
 - mlpack::cv::Recall, 1148
 - mlpack::cv::SimpleCV, 1157
 - mlpack::hpt::CVFunction, 1364
 - mlpack::kde::KDEModel, 1401, 1402
 - mlpack::kde::KDE, 1391, 1392
 - mlpack::kernel::CauchyKernel, 1415
 - mlpack::kernel::CosineDistance, 1416
 - mlpack::kernel::EpanechnikovKernel, 1419
 - mlpack::kernel::ExampleKernel, 1423
 - mlpack::kernel::GaussianKernel, 1427
 - mlpack::kernel::HyperbolicTangentKernel, 1431
 - mlpack::kernel::LaplacianKernel, 1444, 1445
 - mlpack::kernel::LinearKernel, 1447
 - mlpack::kernel::PSpectrumStringKernel, 1456
 - mlpack::kernel::PolynomialKernel, 1453
 - mlpack::kernel::SphericalKernel, 1459, 1460
 - mlpack::kernel::TriangularKernel, 1463
 - mlpack::lmnn::LMNNFunction, 1533
 - mlpack::metric::IPMetric, 1567
 - mlpack::metric::LMetric, 1571
 - mlpack::metric::MahalanobisDistance, 1575
 - mlpack::nca::SoftmaxErrorFunction, 1588
 - mlpack::nn::SparseAutoencoderFunction, 1719
 - mlpack::regression::LogisticRegressionFunction, 1806
 - mlpack::regression::SoftmaxRegressionFunction, 1820
 - mlpack::svd::BiasSVDFunction, 1928, 1929
 - mlpack::svd::RegularizedSVDFunction, 1946, 1948
 - mlpack::svd::SVDPlusPlusFunction, 1956
 - mlpack::svm::LinearSVMFunction, 1969, 1970
 - mlpack::tree::GiniGain, 2091
 - mlpack::tree::GiniImpurity, 2092
 - mlpack::tree::InformationGain, 2138, 2139
- EvaluateFitnessFunction
 - mlpack::tree::BinaryNumericSplit, 1994
 - mlpack::tree::HoeffdingCategoricalSplit, 2106
 - mlpack::tree::HoeffdingNumericSplit, 2111
- EvaluatePtr
 - mlpack::tree::GiniGain, 2091
 - mlpack::tree::InformationGain, 2139
- EvaluateWithGradient
 - mlpack::ann::BRNN, 614
 - mlpack::ann::FFN, 698, 699
 - mlpack::ann::RNN, 916
 - mlpack::lmnn::LMNNFunction, 1534
 - mlpack::regression::LogisticRegressionFunction, 1807
 - mlpack::svm::LinearSVMFunction, 1970, 1971
- ExactClone
 - mlpack::tree::RectangleTree, 2220
- ExactSVDPolicy, 1722
- ExampleKernel, 1421
 - mlpack::kernel::ExampleKernel, 1422
- ExampleTree
 - mlpack::tree::ExampleTree, 2083
- ExampleTree< MetricType, StatisticType, MatType >, 2081
- ExplorationSteps
 - mlpack::rl::TrainingConfig, 1903
- Extension
 - mlpack::data, 379
- ExtractSVD
 - mlpack::svd::QUIC_SVD, 1933
- F1< AS, PositiveClass >, 1132
- FFN< OutputLayerType, InitializationRuleType, CustomLayers >, 692
- FFTConvolution< BorderMode, padLastDim >, 705
- FFN
 - mlpack::ann::FFN, 695
- FastLSTM< InputDataType, OutputDataType >, 685
- FastLSTM
 - mlpack::ann::FastLSTM, 688
- FastMKS< KernelType, MatType, TreeType >, 1280
- FastMKSMModel, 1291

- mlpack::fastmks::FastMKModel, 1294
- FastMKSRules
 - mlpack::fastmks::FastMKSRules, 1299
- FastMKSRules< KernelType, TreeType >, 1298
- FastMKSStat, 1303
 - mlpack::fastmks::FastMKSStat, 1304
- FastMKS
 - mlpack::fastmks::FastMKS, 1283–1286
- Fatal
 - mlpack::Log, 1540
- FeatureSize
 - mlpack::regression::SoftmaxRegression, 1815
 - mlpack::svm::LinearSVM, 1964
- Filter
 - mlpack::hmm::HMMRegression, 1358
 - mlpack::hmm::HMM, 1343
- FilterFileName
 - test_tools.hpp, 3090
- Find
 - mlpack::emst::UnionFind, 1279
- find_python_module
 - bindings/python/CMakeLists.txt, 2387
- FindBucket
 - mlpack::det::DTree, 1215
- FindByBeginCount
 - mlpack::tree::RectangleTree, 2220, 2221
- FirstBound
 - mlpack::neighbor::NeighborSearchStat, 1655
- FirstLeafExact
 - mlpack::neighbor::RAModel, 1673
 - mlpack::neighbor::RASearch, 1686
- FirstLeafExactVisitor, 1601
- FirstPointIsCentroid
 - mlpack::tree::TreeTraits, 2304
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound←::BallBound, SplitType > >, 2306
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound←::CellBound, SplitType > >, 2308
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound←::HollowBallBound, SplitType > >, 2310
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, Bound←Type, RPTreeMaxSplit > >, 2312
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, Bound←Type, RPTreeMeanSplit > >, 2314
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, Bound←Type, SplitType > >, 2317
 - mlpack::tree::TreeTraits< CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > >, 2319
- mlpack::tree::TreeTraits< Octree< MetricType, StatisticType, MatType > >, 2322
- mlpack::tree::TreeTraits< RectangleTree< Metric←Type, StatisticType, MatType, RPlusTreeSplit< SplitPolicyType, SweepType >, DescentType, AuxiliaryInformationType > >, 2324
- mlpack::tree::TreeTraits< RectangleTree< Metric←Type, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType > >, 2327
- mlpack::tree::TreeTraits< SpillTree< MetricType, StatisticType, MatType, HyperplaneType, Split←Type > >, 2329
- FirstPointIsRoot, 2089
- FitIntercept
 - mlpack::regression::SoftmaxRegression, 1816
 - mlpack::regression::SoftmaxRegressionFunction, 1821
 - mlpack::svm::LinearSVMFunction, 1971
- Fixed
 - mlpack::hpt, 398
- FixedArg< T, I >, 1373
- FixedRandomSeed
 - mlpack::math, 413
- FlexibleReLU< InputDataType, OutputDataType >, 707
- FlexibleReLU
 - mlpack::ann::FlexibleReLU, 709
- Fn
 - mlpack::ann::HardSigmoidFunction, 739
 - mlpack::ann::IdentityFunction, 748, 750
 - mlpack::ann::LogisticFunction, 790, 791
 - mlpack::ann::RectifierFunction, 885
 - mlpack::ann::SoftplusFunction, 941, 942
 - mlpack::ann::SoftsignFunction, 945
 - mlpack::ann::SwishFunction, 954, 955
 - mlpack::ann::TanhFunction, 957
- force_inline
 - prereqs.hpp, 3076
- format
 - mlpack::data, 376
 - mlpack::det::PathCacher, 1225
- Forward
 - mlpack::ann::Add, 556
 - mlpack::ann::AddMerge, 562
 - mlpack::ann::AlphaDropout, 570
 - mlpack::ann::AtrousConvolution, 576
 - mlpack::ann::BaseLayer, 591
 - mlpack::ann::BatchNorm, 596
 - mlpack::ann::BilinearInterpolation, 607
 - mlpack::ann::CReLU, 655
 - mlpack::ann::Concat, 627
 - mlpack::ann::ConcatPerformance, 637
 - mlpack::ann::Concatenate, 633

- mlpack::ann::Constant, 640
- mlpack::ann::Convolution, 647
- mlpack::ann::CrossEntropyError, 658
- mlpack::ann::DiceLoss, 664
- mlpack::ann::DropConnect, 670
- mlpack::ann::Dropout, 676
- mlpack::ann::ELU, 684
- mlpack::ann::EarthMoverDistance, 679
- mlpack::ann::FFN, 699, 700
- mlpack::ann::FastLSTM, 689
- mlpack::ann::FlexibleReLU, 710
- mlpack::ann::GRU, 734
- mlpack::ann::Glimpse, 720
- mlpack::ann::HardTanH, 742
- mlpack::ann::Join, 755
- mlpack::ann::KLDivergence, 760
- mlpack::ann::LSTM, 804, 805
- mlpack::ann::LayerNorm, 765
- mlpack::ann::LeakyReLU, 773
- mlpack::ann::Linear, 779
- mlpack::ann::LinearNoBias, 785
- mlpack::ann::LogSoftMax, 794
- mlpack::ann::Lookup, 798
- mlpack::ann::MaxPooling, 811
- mlpack::ann::MeanPooling, 819
- mlpack::ann::MeanSquaredError, 824
- mlpack::ann::MultiplyConstant, 827
- mlpack::ann::MultiplyMerge, 832
- mlpack::ann::NegativeLogLikelihood, 839
- mlpack::ann::PReLU, 858
- mlpack::ann::ReconstructionLoss, 882
- mlpack::ann::Recurrent, 890
- mlpack::ann::RecurrentAttention, 896
- mlpack::ann::ReinforceNormal, 901
- mlpack::ann::Reparametrization, 905
- mlpack::ann::Select, 926
- mlpack::ann::Sequential, 931
- mlpack::ann::SigmoidCrossEntropyError, 938
- mlpack::ann::Subview, 949
- mlpack::ann::TransposedConvolution, 962
- mlpack::ann::VRClassReward, 970
- mlpack::hmm::HMM, 1343
- ForwardVisitor, 713
 - mlpack::ann::ForwardVisitor, 714
- FreeEnergy
 - mlpack::ann::RBM, 869, 870
- FrobNormSquared
 - mlpack::tree::CosineTree, 2035
- FullConvolution, 714
- functionMap
 - mlpack::CLI, 1126
- FunctionMapType
 - mlpack::CLI, 1121
- FurthestDescendantDistance
 - mlpack::tree::BinarySpaceTree, 2011
 - mlpack::tree::CoverTree, 2052
 - mlpack::tree::ExampleTree, 2085
 - mlpack::tree::Octree, 2177
 - mlpack::tree::RectangleTree, 2221
 - mlpack::tree::SpillTree, 2285
- FurthestNeighborSort
 - mlpack::neighbor, 431
- FurthestNS, 1602
- FurthestPointDistance
 - mlpack::tree::BinarySpaceTree, 2011
 - mlpack::tree::CoverTree, 2053
 - mlpack::tree::Octree, 2178
 - mlpack::tree::RectangleTree, 2221
 - mlpack::tree::SpillTree, 2285
- GMMHMM
 - mlpack::hmm::HMMModel, 1352
- GMM, 1323
 - mlpack::gmm::GMM, 1325–1327
- GRU< InputDataType, OutputDataType >, 731
- GRU
 - mlpack::ann::GRU, 732
- Gamma
 - mlpack::kernel::GaussianKernel, 1428
- GammaDistribution, 1238
 - mlpack::distribution::GammaDistribution, 1240
- GaussianDistribution, 1246
 - mlpack::distribution::GaussianDistribution, 1247
- GaussianHMM
 - mlpack::hmm::HMMModel, 1352
- GaussianInitialization, 715
 - mlpack::ann::GaussianInitialization, 715
- GaussianKernel, 1424
 - mlpack::kernel::GaussianKernel, 1425
- Gaussians
 - mlpack::gmm::DiagonalGMM, 1312
 - mlpack::gmm::GMM, 1328
- Generate
 - mlpack::ann::augmented::tasks::AddTask, 581, 582
 - mlpack::ann::augmented::tasks::CopyTask, 584
 - mlpack::ann::augmented::tasks::SortTask, 586
 - mlpack::hmm::HMM, 1343
- Get
 - mlpack::Timer, 1976
- GetAllTimers
 - mlpack::Timers, 1979
- GetAllocatedMemory
 - mlpack::bindings::cli, 288, 289
 - mlpack::bindings::tests, 360, 361
- GetArmaType
 - mlpack::bindings::python, 333
- GetBestChild
 - mlpack::neighbor::FurthestNS, 1606, 1607

- mlpack::neighbor::NearestNS, 1625
- mlpack::neighbor::NeighborSearchRules, 1645
- GetBindingName
 - mlpack::bindings::cli, 289
 - mlpack::bindings::markdown, 314
 - mlpack::bindings::python, 333
- GetCythonType
 - mlpack::bindings::python, 333, 334
- GetCythonType< bool >
 - mlpack::bindings::python, 334
- GetCythonType< double >
 - mlpack::bindings::python, 334
- GetCythonType< int >
 - mlpack::bindings::python, 334
- GetCythonType< size_t >
 - mlpack::bindings::python, 335
- GetCythonType< std::string >
 - mlpack::bindings::python, 335
- GetDataset
 - mlpack::tree::CosineTree, 2036
- GetFinalBasis
 - mlpack::tree::CosineTree, 2036
- GetFurthestChild
 - mlpack::tree::BinarySpaceTree, 2012
 - mlpack::tree::CoverTree, 2053
 - mlpack::tree::Octree, 2178
 - mlpack::tree::RectangleTree, 2222
 - mlpack::tree::SpillTree, 2286
- GetGroundTruthMatrix
 - mlpack::regression::SoftmaxRegressionFunction, 1821
 - mlpack::svm::LinearSVMFunction, 1971
- GetInitialPoint
 - mlpack::lmnn::LMNNFunction, 1535
 - mlpack::nca::SoftmaxErrorFunction, 1588
 - mlpack::nn::SparseAutoencoderFunction, 1719
 - mlpack::regression::LogisticRegressionFunction, 1807
 - mlpack::regression::SoftmaxRegressionFunction, 1821
 - mlpack::svd::BiasSVDFunction, 1929
 - mlpack::svd::RegularizedSVDFunction, 1948
 - mlpack::svd::SVDPlusPlusFunction, 1956
- GetKernelMatrix
 - mlpack::kernel::NystroemMethod, 1449, 1450
- GetMatrixSize
 - mlpack::data::LoadCSV, 1189
- GetMemState
 - arma_util.hpp, 2519
- GetMemory
 - arma_util.hpp, 2519
- GetNearestChild
 - mlpack::tree::BinarySpaceTree, 2012
 - mlpack::tree::CoverTree, 2053, 2054
 - mlpack::tree::Octree, 2178, 2179
 - mlpack::tree::RectangleTree, 2222
 - mlpack::tree::SpillTree, 2286
- GetNeighborhood
 - mlpack::cf::BatchSVDPolicy, 1033
 - mlpack::cf::BiasSVDPolicy, 1038
 - mlpack::cf::NMFPolicy, 1067
 - mlpack::cf::RandomizedSVDPolicy, 1081
 - mlpack::cf::RegSVDPolicy, 1090
 - mlpack::cf::SVDCompletePolicy, 1096
 - mlpack::cf::SVDIncompletePolicy, 1099
 - mlpack::cf::SVDPlusPlusPolicy, 1104
- GetNewFeatures
 - mlpack::nn::SparseAutoencoder, 1714
- GetNumpyType
 - mlpack::bindings::python, 335
- GetNumpyType< double >
 - mlpack::bindings::python, 335
- GetNumpyType< size_t >
 - mlpack::bindings::python, 336
- GetNumpyTypeChar
 - mlpack::bindings::python, 336
- GetNumpyTypeChar< arma::Col< size_t > >
 - mlpack::bindings::python, 336
- GetNumpyTypeChar< arma::Mat< size_t > >
 - mlpack::bindings::python, 336
- GetNumpyTypeChar< arma::Row< size_t > >
 - mlpack::bindings::python, 336
- GetNumpyTypeChar< arma::mat >
 - mlpack::bindings::python, 336
- GetNumpyTypeChar< arma::rowvec >
 - mlpack::bindings::python, 337
- GetNumpyTypeChar< arma::vec >
 - mlpack::bindings::python, 337
- GetParam
 - mlpack::CLI, 1122
 - mlpack::bindings::cli, 290, 291
 - mlpack::bindings::markdown, 315
 - mlpack::bindings::python, 337
 - mlpack::bindings::tests, 361
- GetParamPtr
 - mlpack::util, 469
- GetParamWithInfo
 - mlpack::util, 469
- GetPrintableParam
 - mlpack::CLI, 1122
 - mlpack::bindings::cli, 292, 293
 - mlpack::bindings::markdown, 315–317
 - mlpack::bindings::python, 337–339
 - mlpack::bindings::tests, 362, 363
- GetPrintableParamName
 - mlpack::bindings::cli, 293, 294
 - mlpack::bindings::markdown, 317, 318
- GetPrintableParamValue

- mlpack::bindings::cli, 294, 295
- mlpack::bindings::markdown, 318, 319
- GetPrintableType
 - mlpack::bindings::cli, 295–297
 - mlpack::bindings::markdown, 320
 - mlpack::bindings::python, 340, 341
- GetPrintableType< bool >
 - mlpack::bindings::python, 341
- GetPrintableType< double >
 - mlpack::bindings::python, 341
- GetPrintableType< int >
 - mlpack::bindings::python, 341
- GetPrintableType< size_t >
 - mlpack::bindings::python, 342
- GetPrintableType< std::string >
 - mlpack::bindings::python, 342
- GetProbabilitiesMatrix
 - mlpack::regression::SoftmaxRegressionFunction, 1821
- GetProgramDoc
 - mlpack::bindings::markdown::BindingInfo, 984
- GetProjVector
 - mlpack::tree::SpaceSplit, 2272, 2273
- GetRating
 - mlpack::cf::BatchSVDPolicy, 1034
 - mlpack::cf::BiasSVDPolicy, 1039
 - mlpack::cf::NMFPolicy, 1068
 - mlpack::cf::RandomizedSVDPolicy, 1081
 - mlpack::cf::RegSVDPolicy, 1090
 - mlpack::cf::SVDCompletePolicy, 1096
 - mlpack::cf::SVDIncompletePolicy, 1100
 - mlpack::cf::SVDPlusPlusPolicy, 1105
- GetRatingOfUser
 - mlpack::cf::BatchSVDPolicy, 1034
 - mlpack::cf::BiasSVDPolicy, 1039
 - mlpack::cf::NMFPolicy, 1068
 - mlpack::cf::RandomizedSVDPolicy, 1082
 - mlpack::cf::RegSVDPolicy, 1090
 - mlpack::cf::SVDCompletePolicy, 1097
 - mlpack::cf::SVDIncompletePolicy, 1100
 - mlpack::cf::SVDPlusPlusPolicy, 1105
- GetRawParam
 - mlpack::CLI, 1123
 - mlpack::bindings::cli, 297, 298
- GetRecommendations
 - mlpack::cf::CFModel, 1043, 1044
 - mlpack::cf::CFTYPE, 1048, 1049
- GetResults
 - mlpack::fastmks::FastMKSRules, 1300
 - mlpack::neighbor::NeighborSearchRules, 1646
 - mlpack::neighbor::RASearchRules, 1694
- GetSingleton
 - mlpack::CLI, 1123
- GetSplitPolicy
 - mlpack::tree::RPlusPlusTreeSplitPolicy, 2248
 - mlpack::tree::RPlusTreeSplitPolicy, 2254
- GetState
 - mlpack::Timers, 1979
- GetTimer
 - mlpack::Timers, 1979
- GetTransposeMatrixSize
 - mlpack::data::LoadCSV, 1190
- GetValueVisitor, 1060
- GetVersion
 - mlpack::util, 469
- GetWeights
 - mlpack::cf::AverageInterpolation, 1031
 - mlpack::cf::RegressionInterpolation, 1087
 - mlpack::cf::SimilarityInterpolation, 1093
- Gibbs
 - mlpack::ann::RBM, 870
- GiniBinaryTreeType
 - mlpack::tree::HoeffdingTreeModel, 2128
- GiniGain, 2090
- GiniHoeffdingTreeType
 - mlpack::tree::HoeffdingTreeModel, 2128
- GiniImpurity, 2092
- git
 - gitversion.hpp, 2696
- gitversion.hpp
 - git, 2696
- GivenInitialization, 508
 - mlpack::amf::GivenInitialization, 508, 509
- Glimpse
 - mlpack::ann::Glimpse, 718
- Glimpse< InputDataType, OutputDataType >, 717
- GlorotInitialization
 - mlpack::ann, 269
- GlorotInitializationType
 - mlpack::ann::GlorotInitializationType, 724
- GlorotInitializationType< Uniform >, 723
- Gradient
 - mlpack::ann::Add, 556, 557
 - mlpack::ann::AddMerge, 563
 - mlpack::ann::AtrousConvolution, 576, 577
 - mlpack::ann::BRNN, 614
 - mlpack::ann::BatchNorm, 597
 - mlpack::ann::Concat, 627, 628
 - mlpack::ann::Convolution, 647
 - mlpack::ann::DropConnect, 670, 671
 - mlpack::ann::FFN, 700
 - mlpack::ann::FastLSTM, 690
 - mlpack::ann::FlexibleReLU, 711
 - mlpack::ann::GRU, 735
 - mlpack::ann::LSTM, 805
 - mlpack::ann::LayerNorm, 765, 767
 - mlpack::ann::Linear, 780
 - mlpack::ann::LinearNoBias, 785

- mlpack::ann::Lookup, 798, 799
- mlpack::ann::MultiplyMerge, 832
- mlpack::ann::PReLU, 859
- mlpack::ann::RBM, 870
- mlpack::ann::RNN, 916
- mlpack::ann::Recurrent, 890, 891
- mlpack::ann::RecurrentAttention, 896, 897
- mlpack::ann::Sequential, 932
- mlpack::ann::TransposedConvolution, 963
- mlpack::hpt::CVFunction, 1364
- mlpack::kernel::EpanechnikovKernel, 1419
- mlpack::kernel::GaussianKernel, 1428
- mlpack::kernel::LaplacianKernel, 1445
- mlpack::kernel::SphericalKernel, 1460
- mlpack::kernel::TriangularKernel, 1464
- mlpack::lmnn::LMNNFunction, 1535, 1536
- mlpack::nca::SoftmaxErrorFunction, 1588, 1589
- mlpack::nn::SparseAutoencoderFunction, 1719
- mlpack::regression::LogisticRegressionFunction, 1807, 1808
- mlpack::regression::SoftmaxRegressionFunction, 1822
- mlpack::svd::BiasSVDFunction, 1929, 1930
- mlpack::svd::RegularizedSVDFunction, 1948, 1949
- mlpack::svd::SVDPlusPlusFunction, 1957
- mlpack::svm::LinearSVMFunction, 1972
- GradientForSquaredDistance
 - mlpack::kernel::EpanechnikovKernel, 1420
 - mlpack::kernel::GaussianKernel, 1428
- GradientLimit
 - mlpack::rl::TrainingConfig, 1903, 1904
- GradientSetVisitor, 725
 - mlpack::ann::GradientSetVisitor, 726
- GradientUpdateVisitor, 727
 - mlpack::ann::GradientUpdateVisitor, 727
- GradientVisitor, 728
 - mlpack::ann::GradientVisitor, 729
- GradientZeroVisitor, 730
 - mlpack::ann::GradientZeroVisitor, 730
- Greater
 - mlpack::emst::EdgePair, 1277
- GreedyPolicy
 - mlpack::rl::GreedyPolicy, 1859
- GreedyPolicy< EnvironmentType >, 1858
- GreedySingleTreeTraverser
 - mlpack::tree::GreedySingleTreeTraverser, 2093
- GreedySingleTreeTraverser< TreeType, RuleType >, 2093
- Grow
 - mlpack::det::DTree, 1216
- H
 - mlpack::cf::BatchSVDPolicy, 1034
 - mlpack::cf::BiasSVDPolicy, 1040
 - mlpack::cf::NMFPolicy, 1069
 - mlpack::cf::RandomizedSVDPolicy, 1082
 - mlpack::cf::RegSVDPolicy, 1091
 - mlpack::cf::SVDCompletePolicy, 1097
 - mlpack::cf::SVDIncompletePolicy, 1101
 - mlpack::cf::SVDPlusPlusPolicy, 1106
- HAS_ANY_METHOD_FORM
 - mlpack::ann, 272
 - sfinae_utility.hpp, 2742
- HAS_EXACT_METHOD_FORM
 - mlpack::data, 379
 - sfinae_utility.hpp, 2743
- HAS_MEM_FUNC
 - mlpack::ann, 272–274
 - sfinae_utility.hpp, 2744
- HAS_METHOD_FORM_BASE
 - sfinae_utility.hpp, 2745
- HAS_METHOD_FORM
 - sfinae_utility.hpp, 2744
- HMM< Distribution >, 1335
- HMMModel, 1350
 - mlpack::hmm::HMMModel, 1351
- HMMRegression, 1354
 - mlpack::hmm::HMMRegression, 1356
- HMMType
 - mlpack::hmm, 396
- HMM
 - mlpack::hmm::HMM, 1338, 1339
- HRectBound
 - mlpack::bound::HRectBound, 1020, 1021
- HRectBound< MetricType, ElemType >, 1018
- HUpdate
 - mlpack::amf::NMFALSUpdate, 517
 - mlpack::amf::NMFMultiplicativeDistanceUpdate, 520
 - mlpack::amf::NMFMultiplicativeDivergenceUpdate, 524
 - mlpack::amf::SVDBatchLearning, 540
 - mlpack::amf::SVDCompleteIncrementalLearning, 543
 - mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >, 545
 - mlpack::amf::SVDIncompleteIncrementalLearning, 548
- HamerlyKMeans
 - mlpack::kmeans::HamerlyKMeans, 1480
- HamerlyKMeans< MetricType, MatType >, 1479
- HandleNodeInsertion
 - mlpack::tree::HilbertRTreeAuxiliaryInformation, 2098
 - mlpack::tree::NoAuxiliaryInformation, 2162
 - mlpack::tree::RPlusPlusTreeAuxiliaryInformation, 2242
 - mlpack::tree::XTreeAuxiliaryInformation, 2341
- HandleNodeRemoval
 - mlpack::tree::HilbertRTreeAuxiliaryInformation, 2098
 - mlpack::tree::NoAuxiliaryInformation, 2162

- mlpack::tree::RPlusPlusTreeAuxiliaryInformation, 2243
- mlpack::tree::XTreeAuxiliaryInformation, 2341
- HandlePointDeletion
 - mlpack::tree::HilbertRTreeAuxiliaryInformation, 2098
 - mlpack::tree::NoAuxiliaryInformation, 2163
 - mlpack::tree::RPlusPlusTreeAuxiliaryInformation, 2243
 - mlpack::tree::XTreeAuxiliaryInformation, 2342
- HandlePointInsertion
 - mlpack::tree::HilbertRTreeAuxiliaryInformation, 2099
 - mlpack::tree::NoAuxiliaryInformation, 2163
 - mlpack::tree::RPlusPlusTreeAuxiliaryInformation, 2244
 - mlpack::tree::XTreeAuxiliaryInformation, 2342
- HardSigmoidFunction, 737
- HardSigmoidLayer
 - mlpack::ann, 269
- HardTanH< InputDataType, OutputDataType >, 740
- HardTanH
 - mlpack::ann::HardTanH, 741
- HasDuplicatedPoints
 - mlpack::tree::TreeTraits, 2304
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound← ::BallBound, SplitType > >, 2306
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound← ::CellBound, SplitType > >, 2308
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound← ::HollowBallBound, SplitType > >, 2310
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, Bound← Type, RPTreeMaxSplit > >, 2312
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, Bound← Type, RPTreeMeanSplit > >, 2315
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, Bound← Type, SplitType > >, 2317
 - mlpack::tree::TreeTraits< CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > >, 2320
 - mlpack::tree::TreeTraits< Octree< MetricType, StatisticType, MatType > >, 2322
 - mlpack::tree::TreeTraits< RectangleTree< Metric← Type, StatisticType, MatType, RPlusTreeSplit< SplitPolicyType, SweepType >, DescentType, AuxiliaryInformationType > >, 2325
 - mlpack::tree::TreeTraits< RectangleTree< Metric← Type, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType > >, 2327
 - mlpack::tree::TreeTraits< SpillTree< MetricType, StatisticType, MatType, HyperplaneType, Split← Type > >, 2330
- HasParam
 - mlpack::CLI, 1123
- HasSelfChildren
 - mlpack::tree::SpillTree, 2287
 - mlpack::tree::TreeTraits, 2304
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound← ::BallBound, SplitType > >, 2306
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound← ::CellBound, SplitType > >, 2308
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound← ::HollowBallBound, SplitType > >, 2310
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, Bound← Type, RPTreeMaxSplit > >, 2313
 - mlpack::tree::TreeTraits< BinarySpaceTree<
- HasOverlappingChildren
 - mlpack::tree::TreeTraits, 2304
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound← ::BallBound, SplitType > >, 2306
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound← ::CellBound, SplitType > >, 2308
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound← ::HollowBallBound, SplitType > >, 2310
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, Bound← Type, RPTreeMaxSplit > >, 2312
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, Bound← Type, RPTreeMeanSplit > >, 2315
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, Bound← Type, SplitType > >, 2317
 - mlpack::tree::TreeTraits< CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > >, 2320
 - mlpack::tree::TreeTraits< Octree< MetricType, StatisticType, MatType > >, 2322
 - mlpack::tree::TreeTraits< RectangleTree< Metric← Type, StatisticType, MatType, RPlusTreeSplit< SplitPolicyType, SweepType >, DescentType, AuxiliaryInformationType > >, 2325
 - mlpack::tree::TreeTraits< RectangleTree< Metric← Type, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType > >, 2327
 - mlpack::tree::TreeTraits< SpillTree< MetricType, StatisticType, MatType, HyperplaneType, Split← Type > >, 2330

- MetricType, StatisticType, MatType, Bound←
Type, RPTreeMeanSplit > >, 2315
- mlpack::tree::TreeTraits< BinarySpaceTree<
MetricType, StatisticType, MatType, Bound←
Type, SplitType > >, 2317
- mlpack::tree::TreeTraits< CoverTree< MetricType,
StatisticType, MatType, RootPointPolicy > >,
2320
- mlpack::tree::TreeTraits< Octree< MetricType,
StatisticType, MatType > >, 2322
- mlpack::tree::TreeTraits< RectangleTree< Metric←
Type, StatisticType, MatType, RPlusTreeSplit<
SplitPolicyType, SweepType >, DescentType,
AuxiliaryInformationType > >, 2325
- mlpack::tree::TreeTraits< RectangleTree< Metric←
Type, StatisticType, MatType, SplitType,
DescentType, AuxiliaryInformationType > >,
2327
- mlpack::tree::TreeTraits< SpillTree< MetricType,
StatisticType, MatType, HyperplaneType, Split←
Type > >, 2330
- HasSerialize< T >, 1178
- HasSerialize< T >::check< U, V, W >, 1180
- HasSerializeFunction< T >, 1180
- HasTightBounds
 - mlpack::bound::BoundTraits, 1002
 - mlpack::bound::BoundTraits< BallBound< Metric←
Type, VecType > >, 1003
 - mlpack::bound::BoundTraits< CellBound< Metric←
Type, ElemType > >, 1004
 - mlpack::bound::BoundTraits< HRectBound<
MetricType, ElemType > >, 1005
 - mlpack::bound::BoundTraits< HollowBallBound<
MetricType, ElemType > >, 1005
- HelInitialization, 744
 - mlpack::ann::HelInitialization, 745
- Hi
 - mlpack::math::RangeType, 1552
- HiddenBias
 - mlpack::ann::RBM, 871
- HiddenMean
 - mlpack::ann::RBM, 871, 872
- HiddenSize
 - mlpack::ann::RBM, 872
 - mlpack::nn::SparseAutoencoder, 1715
 - mlpack::nn::SparseAutoencoderFunction, 1720
- HideChild
 - mlpack::kmeans, 404
- HilbertRTree
 - mlpack::tree, 452
- HilbertRTreeAuxiliaryInformation
 - mlpack::tree::HilbertRTreeAuxiliaryInformation, 2096,
2097
- HilbertRTreeAuxiliaryInformation< TreeType, Hilbert←
ValueType >, 2095
- HilbertRTreeDescentHeuristic, 2101
- HilbertRTreeSplit< splitOrder >, 2102
- HilbertValue
 - mlpack::tree::HilbertRTreeAuxiliaryInformation, 2099
- history
 - mlpack::tree::XTreeAuxiliaryInformation::Split←
HistoryStruct, 2347
- hmm_model.hpp
 - BOOST_CLASS_VERSION, 2953
- HoeffdingCategoricalSplit
 - mlpack::tree::HoeffdingCategoricalSplit, 2105, 2106
- HoeffdingCategoricalSplit< FitnessFunction >, 2104
- HoeffdingDoubleNumericSplit
 - mlpack::tree, 453
- HoeffdingNumericSplit
 - mlpack::tree::HoeffdingNumericSplit, 2110
- HoeffdingNumericSplit< FitnessFunction, Observation←
Type >, 2108
- HoeffdingTree
 - mlpack::tree::HoeffdingTree, 2116–2118
- HoeffdingTree< FitnessFunction, NumericSplitType,
CategoricalSplitType >, 2113
- HoeffdingTreeModel, 2126
 - mlpack::tree::HoeffdingTreeModel, 2129
- HoeffdingTreeType
 - mlpack::tree, 453
- HollowBallBound
 - mlpack::bound::HollowBallBound, 1009, 1010
- HollowBallBound< TMetricType, ElemType >, 1006
- HollowCenter
 - mlpack::bound::HollowBallBound, 1012, 1013
- HyperParameterTuner
 - mlpack::hpt::HyperParameterTuner, 1377
- HyperParameterTuner< MLAlgorithm, Metric, C←
V, OptimizerType, MatType, PredictionsType,
WeightsType >, 1375
- HyperbolicTangentKernel, 1430
 - mlpack::kernel::HyperbolicTangentKernel, 1430,
1431
- Hyperplane
 - mlpack::tree, 453
 - mlpack::tree::SpillTree, 2287
- HyperplaneBase
 - mlpack::tree::HyperplaneBase, 2135
- HyperplaneBase< BoundT, ProjVectorT >, 2133
- HyphenateString
 - mlpack::util, 469
- IPMetric
 - mlpack::metric::IPMetric, 1566, 1567
- IPMetric< KernelType >, 1565
- IdentityFunction, 746
- IdentityLayer

- mlpack::ann, 269
- IgnoreCheck
 - mlpack::bindings::cli, 298
 - mlpack::bindings::markdown, 320
 - mlpack::bindings::python, 342, 343
 - mlpack::bindings::tests, 364
- ignoreInput
 - mlpack::util::PrefixedOutputStream, 2368
- ImplicitData
 - mlpack::cf::SVDPlusPlusPolicy, 1106
- ImplicitDataset
 - mlpack::svd::SVDPlusPlusFunction, 1957
- ImportDecl
 - mlpack::bindings::python, 343, 344
- Impostors
 - mlpack::lmnn::Constraints, 1522–1524
- Impute
 - mlpack::data::CustomImputation, 1171
 - mlpack::data::Imputer, 1183
 - mlpack::data::ListwiseDeletion, 1188
 - mlpack::data::MeanImputation, 1191
 - mlpack::data::MedianImputation, 1193
- Imputer
 - mlpack::data::Imputer, 1182, 1183
- Imputer< T, MapperType, StrategyType >, 1182
- include_directories
 - CMakeLists.txt, 2389
- IncompleteIncrementalTermination
 - mlpack::amf::IncompleteIncrementalTermination, 511
- IncompleteIncrementalTermination< TerminationPolicy >, 510
- IncrementPolicy, 1184
 - mlpack::data::IncrementPolicy, 1185
- Index
 - mlpack::amf::CompleteIncrementalTermination, 505
 - mlpack::amf::IncompleteIncrementalTermination, 511
 - mlpack::amf::MaxIterationTermination, 515
 - mlpack::amf::SimpleResidueTermination, 531
 - mlpack::amf::SimpleToleranceTermination, 536
 - mlpack::amf::ValidationRMSETermination, 551
- index
 - mlpack::hpt::FixedArg, 1374
- Info
 - mlpack::Log, 1540
- InfoBinaryTreeType
 - mlpack::tree::HoeffdingTreeModel, 2128
- InfoHoeffdingTreeType
 - mlpack::tree::HoeffdingTreeModel, 2128
- InformationGain, 2138
- InitHMMModel, 480
 - Apply, 480
 - Create, 480, 481
 - RandomInitialize, 481, 482
- InitTraits< InitRuleType >, 750
- InitTraits< KathirvalavakumarSubavathiInitialization >, 751
- InitTraits< NguyenWidrowInitialization >, 752
- InitValue
 - mlpack::ann::ConstInitialization, 643
- initValue
 - mlpack::ann::ConstInitialization, 643
- Initial
 - mlpack::hmm::HMM, 1344
- InitialPoint
 - mlpack::regression::LogisticRegressionFunction, 1808
 - mlpack::svm::LinearSVMFunction, 1973
- InitialSample
 - mlpack::rl::Acrobot, 1828
 - mlpack::rl::CartPole, 1844
 - mlpack::rl::ContinuousMountainCar, 1852
 - mlpack::rl::MountainCar, 1863
 - mlpack::rl::Pendulum, 1879
 - mlpack::rl::RewardClipping, 1897
- Initialize
 - mlpack::amf::AverageInitialization, 503
 - mlpack::amf::CompleteIncrementalTermination, 505, 506
 - mlpack::amf::GivenInitialization, 509
 - mlpack::amf::IncompleteIncrementalTermination, 511
 - mlpack::amf::MaxIterationTermination, 515
 - mlpack::amf::NMFALSUpdate, 518
 - mlpack::amf::NMFMultiplicativeDistanceUpdate, 522
 - mlpack::amf::NMFMultiplicativeDivergenceUpdate, 525
 - mlpack::amf::RandomAcolInitialization, 527
 - mlpack::amf::RandomInitialization, 528
 - mlpack::amf::SVDBatchLearning, 541
 - mlpack::amf::SVDCompleteIncrementalLearning, 543
 - mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >, 546
 - mlpack::amf::SVDIncompleteIncrementalLearning, 548
 - mlpack::amf::SimpleResidueTermination, 531
 - mlpack::amf::SimpleToleranceTermination, 536
 - mlpack::amf::ValidationRMSETermination, 551
 - mlpack::ann::ConstInitialization, 642
 - mlpack::ann::GaussianInitialization, 715, 716
 - mlpack::ann::GlorotInitializationType, 724, 725
 - mlpack::ann::HeInitialization, 745, 746
 - mlpack::ann::KathirvalavakumarSubavathiInitialization, 757, 758
 - mlpack::ann::LecunNormalInitialization, 775, 776
 - mlpack::ann::NetworkInitialization, 841
 - mlpack::ann::NguyenWidrowInitialization, 843, 844
 - mlpack::ann::OivsInitialization, 846
 - mlpack::ann::OrthogonalInitialization, 849

- mlpack::ann::RandomInitialization, 863, 864
- mlpack::perceptron::RandomInitialization, 1741
- mlpack::perceptron::ZeroInitialization, 1743
- mlpack::rl::NStepQLearningWorker, 1870
- mlpack::rl::OneStepQLearningWorker, 1873
- mlpack::rl::OneStepSarsaWorker, 1877
- mlpack::sparse_coding::DataDependentRandom←
Initializer, 1913
- mlpack::sparse_coding::NothingInitializer, 1914
- mlpack::sparse_coding::RandomInitializer, 1915
- InitializeRule
 - mlpack::amf::AMF, 501
- InitializeWeights
 - mlpack::nn::SparseAutoencoderFunction, 1720
 - mlpack::regression::SoftmaxRegressionFunction,
1823
 - mlpack::svm::LinearSVMFunction, 1973
- InnerRadius
 - mlpack::bound::HollowBallBound, 1013
- input
 - mlpack::util::ParamData, 2358
- InputElemType
 - mlpack::ann::FastLSTM, 687
- InputHeight
 - mlpack::ann::AtrousConvolution, 577
 - mlpack::ann::Convolution, 648
 - mlpack::ann::Glimpse, 720
 - mlpack::ann::MaxPooling, 811
 - mlpack::ann::MeanPooling, 819
 - mlpack::ann::TransposedConvolution, 963
- InputParameter
 - mlpack::ann::AddMerge, 563
 - mlpack::ann::Concat, 628
 - mlpack::ann::Convolution, 648
 - mlpack::ann::Linear, 780
 - mlpack::ann::LinearNoBias, 786
 - mlpack::ann::NegativeLogLikelihood, 839, 840
 - mlpack::ann::Sequential, 932
 - mlpack::ann::TransposedConvolution, 964
- InputWidth
 - mlpack::ann::AtrousConvolution, 577, 578
 - mlpack::ann::Convolution, 648, 649
 - mlpack::ann::Glimpse, 720, 721
 - mlpack::ann::MaxPooling, 812
 - mlpack::ann::MeanPooling, 819, 820
 - mlpack::ann::TransposedConvolution, 964
- InsertNeighbor
 - mlpack::neighbor::NeighborSearchRules, 1646
- InsertNode
 - mlpack::tree::RectangleTree, 2223
- InsertPoint
 - mlpack::tree::RectangleTree, 2223
- Intercept
 - mlpack::regression::LinearRegression, 1792
- Inv
 - mlpack::ann::LogisticFunction, 791
 - mlpack::ann::SoftplusFunction, 942
 - mlpack::ann::SoftsignFunction, 946
 - mlpack::ann::TanhFunction, 958
- IsBetter
 - mlpack::neighbor::FurthestNS, 1607
 - mlpack::neighbor::NearestNS, 1626
- IsBiasLayer
 - mlpack::ann::LayerTraits, 769
- IsBinary
 - mlpack::ann::LayerTraits, 770
- IsConnection
 - mlpack::ann::LayerTraits, 770
- IsConverged
 - mlpack::amf::CompleteIncrementalTermination, 506
 - mlpack::amf::IncompleteIncrementalTermination, 512
 - mlpack::amf::MaxIterationTermination, 515
 - mlpack::amf::SimpleResidueTermination, 531
 - mlpack::amf::SimpleToleranceTermination, 536
 - mlpack::amf::ValidationRMSETermination, 552
- IsLMetric< metric::LMetric< Power, TakeRoot > >, 1029
- IsLMetric< MetricType >, 1028
- IsLSTMLayer
 - mlpack::ann::LayerTraits, 770
- IsLeaf
 - mlpack::tree::BinarySpaceTree, 2013
 - mlpack::tree::CoverTree, 2054
 - mlpack::tree::Octree, 2179
 - mlpack::tree::RectangleTree, 2224
 - mlpack::tree::SpillTree, 2287
- IsNaNInf
 - mlpack::data, 379
- IsNormalized
 - mlpack::kernel::KernelTraits, 1434
 - mlpack::kernel::KernelTraits< CauchyKernel >, 1435
 - mlpack::kernel::KernelTraits< CosineDistance >,
1435
 - mlpack::kernel::KernelTraits< EpanechnikovKernel
>, 1437
 - mlpack::kernel::KernelTraits< GaussianKernel >,
1438
 - mlpack::kernel::KernelTraits< LaplacianKernel >,
1439
 - mlpack::kernel::KernelTraits< SphericalKernel >,
1440
 - mlpack::kernel::KernelTraits< TriangularKernel >,
1441
- IsOutputLayer
 - mlpack::ann::LayerTraits, 770
- IsPreFixedArg< T >, 1380
- IsSerializable
 - mlpack::bindings::markdown, 321

- IsSpillTree< tree::SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > >, 2141
- IsSpillTree< TreeType >, 2140
- IsStdVector< std::vector< T, A > >, 2349
- IsStdVector< T >, 2349
- IsSupported
 - mlpack::cv::MetalInfoExtractor, 1141
- IsTerminal
 - mlpack::rl::Acrobot, 1828
 - mlpack::rl::CartPole, 1844
 - mlpack::rl::ContinuousMountainCar, 1852
 - mlpack::rl::MountainCar, 1863
 - mlpack::rl::RewardClipping, 1897
- IsTrained
 - mlpack::kde::KDE, 1393
- IsVector
 - value, 483
- IsVector< arma::Col< eT > >, 483
 - value, 484
- IsVector< arma::Row< eT > >, 484
 - value, 484
- IsVector< arma::SpCol< eT > >, 485
 - value, 485
- IsVector< arma::SpRow< eT > >, 485
 - value, 486
- IsVector< arma::SpSubview< eT > >, 486
 - value, 486
- IsVector< arma::subview_col< eT > >, 487
 - value, 487
- IsVector< arma::subview_row< eT > >, 487
 - value, 488
- IsVector< VecType >, 482
- ItemMeanNormalization, 1061
 - mlpack::cf::ItemMeanNormalization, 1062
- Iterate
 - mlpack::kmeans::DualTreeKMeans, 1469
 - mlpack::kmeans::ElkanKMeans, 1479
 - mlpack::kmeans::HamerlyKMeans, 1480
 - mlpack::kmeans::NaiveKMeans, 1492
 - mlpack::kmeans::PellegMooreKMeans, 1495
- IteratedPower
 - mlpack::cf::RandomizedSVDPolicy, 1082
 - mlpack::pca::RandomizedSVDPolicy, 1734
 - mlpack::svd::RandomizedSVD, 1942
- Iteration
 - mlpack::amf::CompleteIncrementalTermination, 507
 - mlpack::amf::IncompleteIncrementalTermination, 512
 - mlpack::amf::MaxIterationTermination, 515
 - mlpack::amf::SimpleResidueTermination, 532
 - mlpack::amf::SimpleToleranceTermination, 538
 - mlpack::amf::ValidationRMSETermination, 552
- iteration
 - mlpack::amf::SimpleResidueTermination, 533
- JacobianPerformanceTest
 - ann_test_tools.hpp, 3078
- JacobianTest
 - ann_test_tools.hpp, 3078
- Join
 - mlpack::ann::Join, 754
- Join< InputDataType, OutputDataType >, 753
- K
 - mlpack::lmnn::Constraints, 1524
 - mlpack::lmnn::LMNNFunction, 1536
 - mlpack::lmnn::LMNN, 1528, 1529
- k
 - mlpack::neighbor::NeighborSearchRules, 1650
- KDE< KernelType, MetricType, MatType, TreeType, DualTreeTraversalType, SingleTreeTraversalType >, 1387
- KDEMode
 - mlpack::kde, 400
- KDEModel, 1397
 - mlpack::kde::KDEModel, 1399, 1400
- KDERules
 - mlpack::kde::KDERules, 1406
- KDERules< MetricType, KernelType, TreeType >, 1404
- KDEStat, 1408
 - mlpack::kde::KDEStat, 1409
- KDEType
 - mlpack::kde, 400
- KDETypeT
 - mlpack::kde::DualBiKDE, 1385
 - mlpack::kde::DualMonoKDE, 1387
- KDTree
 - mlpack::tree, 453
- KDE
 - mlpack::kde::KDE, 1390
- KFN
 - mlpack::neighbor, 432
- KFoldCV< MLAlgorithm, Metric, MatType, PredictionsType, WeightsType >, 1134
- KFoldCV
 - mlpack::cv::KFoldCV, 1136–1138
- KLDivergence
 - mlpack::ann::KLDivergence, 760
- KLDivergence< InputDataType, OutputDataType >, 759
- KMeans
 - mlpack::kmeans::KMeans, 1485
- KMeans< MetricType, InitialPartitionPolicy, EmptyClusterPolicy, LloydStepType, MatType >, 1483
- KMeansSelection< ClusteringType, maxIterations >, 1441
- KNN
 - mlpack::lmnn::Constraints, 1521
 - mlpack::neighbor, 432
- KRAFN

- mlpack::neighbor, 432
- KRANN
 - mlpack::neighbor, 432
- KathirvalavakumarSubavathiInitialization, 756
 - mlpack::ann::KathirvalavakumarSubavathiInitialization, 757
- Kernel
 - mlpack::kde::KDE, 1393
 - mlpack::kpca::KernelPCA, 1510, 1511
 - mlpack::meanshift::MeanShift, 1563, 1564
 - mlpack::metric::IPMetric, 1568
- KernelNormalizer, 1410
- KernelPCA< KernelType, KernelRule >, 1507
- KernelPCA
 - mlpack::kpca::KernelPCA, 1508
- KernelTraits< CauchyKernel >, 1434
- KernelTraits< CosineDistance >, 1435
- KernelTraits< EpanechnikovKernel >, 1436
- KernelTraits< GaussianKernel >, 1437
- KernelTraits< KernelType >, 1433
- KernelTraits< LaplacianKernel >, 1438
- KernelTraits< SphericalKernel >, 1439
- KernelTraits< TriangularKernel >, 1440
- KernelType
 - mlpack::fastmks::FastMKModel, 1295
 - mlpack::kde::KDEModel, 1402
- KernelTypes
 - mlpack::fastmks::FastMKModel, 1292
 - mlpack::kde::KDEModel, 1398
- KillEmptyClusters, 1481
 - mlpack::kmeans::KillEmptyClusters, 1481
- L2Error
 - mlpack::tree::CosineTree, 2036
- LARS, 1783
 - mlpack::regression::LARS, 1784–1786
- LMNN< MetricType, OptimizerType >, 1526
- LMNNFunction
 - mlpack::lmnn::LMNNFunction, 1532
- LMNNFunction< MetricType >, 1531
- LMNN
 - mlpack::lmnn::LMNN, 1528
- LMetric
 - mlpack::metric::LMetric, 1570
- LMetric< TPower, TTakeRoot >, 1569
- LMetricSearch
 - mlpack::cf::LMetricSearch, 1065
- LMetricSearch< TPower >, 1064
- LSHSearch
 - mlpack::neighbor::LSHSearch, 1610–1612
- LSHSearch< SortPolicy >, 1609
- LSTM< InputDataType, OutputDataType >, 801
- LSTM
 - mlpack::ann::LSTM, 803
- Labels
 - mlpack::lmnn::LMNN, 1529
 - mlpack::nca::NCA, 1585
- Lambda
 - mlpack::ann::ELU, 684
 - mlpack::cf::BiasSVDPolicy, 1040
 - mlpack::cf::SVDPlusPlusPolicy, 1106
 - mlpack::lcc::LocalCoordinateCoding, 1517
 - mlpack::nn::SparseAutoencoder, 1715
 - mlpack::nn::SparseAutoencoderFunction, 1720
 - mlpack::regression::LinearRegression, 1792
 - mlpack::regression::LogisticRegression, 1801
 - mlpack::regression::LogisticRegressionFunction, 1808, 1809
 - mlpack::regression::SoftmaxRegression, 1816
 - mlpack::regression::SoftmaxRegressionFunction, 1824
 - mlpack::svd::BiasSVDFunction, 1930
 - mlpack::svd::RegularizedSVDFunction, 1949
 - mlpack::svd::SVDPlusPlusFunction, 1958
 - mlpack::svm::LinearSVMFunction, 1974
 - mlpack::svm::LinearSVM, 1964, 1965
- Lambda1
 - mlpack::sparse_coding::SparseCoding, 1921
- Lambda2
 - mlpack::sparse_coding::SparseCoding, 1921, 1922
- LambdaPath
 - mlpack::regression::LARS, 1787
- Language
 - mlpack::bindings::markdown::BindingInfo, 985
- LaplaceDistribution, 1251
 - mlpack::distribution::LaplaceDistribution, 1252, 1253
- LaplacianKernel, 1442
 - mlpack::kernel::LaplacianKernel, 1443
- LastBaseCase
 - mlpack::tree::TraversallInfo, 2300, 2301
- lastBaseCase
 - mlpack::neighbor::NeighborSearchRules, 1650
- lastDimension
 - mlpack::tree::XTreeAuxiliaryInformation::Split←HistoryStruct, 2347
- LastDistance
 - mlpack::neighbor::NeighborSearchStat, 1655, 1656
 - mlpack::range::RangeSearchStat, 1770
- LastKernel
 - mlpack::fastmks::FastMKStat, 1305
- LastKernelNode
 - mlpack::fastmks::FastMKStat, 1305
- lastQueryIndex
 - mlpack::neighbor::NeighborSearchRules, 1650
- LastQueryNode
 - mlpack::tree::TraversallInfo, 2301
- lastReferenceIndex
 - mlpack::neighbor::NeighborSearchRules, 1650

- LastReferenceNode
 - mlpack::tree::TraversalInfo, 2301
- LastScore
 - mlpack::tree::TraversalInfo, 2302
- LayerNorm
 - mlpack::ann::LayerNorm, 764
- LayerNorm< InputDataType, OutputDataType >, 762
- LayerTraits< LayerType >, 769
- LayerTypes
 - mlpack::ann, 269
- LeafSize
 - mlpack::neighbor::NSModel, 1662
 - mlpack::neighbor::RAModel, 1674
 - mlpack::range::RSModel, 1775, 1776
- LeakyReLU< InputDataType, OutputDataType >, 771
- LeakyReLU
 - mlpack::ann::LeakyReLU, 772
- LearnDistance
 - mlpack::lmnn::LMNN, 1529
 - mlpack::nca::NCA, 1585
- Leave
 - mlpack::det::PathCacher, 1224
- LecunNormalInitialization, 774
 - mlpack::ann::LecunNormalInitialization, 775
- Left
 - mlpack::det::DTree, 1216
 - mlpack::tree::BinarySpaceTree, 2013
 - mlpack::tree::CosineTree, 2037
 - mlpack::tree::HyperplaneBase, 2136
 - mlpack::tree::SpillTree, 2287
- Lesser
 - mlpack::emst::EdgePair, 1277
- Linear
 - mlpack::ann::Linear, 778
- Linear< InputDataType, OutputDataType >, 776
- LinearKernel, 1446
 - mlpack::kernel::LinearKernel, 1447
- LinearNoBias
 - mlpack::ann::LinearNoBias, 783
- LinearNoBias< InputDataType, OutputDataType >, 782
- LinearRegression, 1789
 - mlpack::regression::LinearRegression, 1790, 1791
- LinearSVM< MatType >, 1959
- LinearSVMFunction
 - mlpack::svm::LinearSVMFunction, 1968
- LinearSVMFunction< MatType >, 1967
- LinearSVM
 - mlpack::svm::LinearSVM, 1961, 1962
- ListwiseDeletion< T >, 1187
- Lo
 - mlpack::math::RangeType, 1553
- Load
 - mlpack::data, 379–383
 - mlpack::data::LoadCSV, 1190
- LoadARFF
 - mlpack::data, 384
- LoadCSV, 1188
 - mlpack::data::LoadCSV, 1189
- LoadHMMAndPerformAction
 - mlpack::hmm, 397
- LoadOutputParameterVisitor, 788
 - mlpack::ann::LoadOutputParameterVisitor, 788
- loaded
 - mlpack::util::ParamData, 2358
- LocalCoordinateCoding, 1513
 - mlpack::lcc::LocalCoordinateCoding, 1515
- Location
 - mlpack::ann::Glimpse, 721
- Log, 1538
- LogAdd
 - mlpack::math, 413
- LogEstimate
 - mlpack::hmm::HMM, 1344
- LogLikelihood
 - mlpack::hmm::HMMRegression, 1359
 - mlpack::hmm::HMM, 1346
- LogNegError
 - mlpack::det::DTree, 1217
- LogNegativeError
 - mlpack::det::DTree, 1216
- LogProbBackward
 - mlpack::ann::BernoulliDistribution, 602
- LogProbability
 - mlpack::ann::BernoulliDistribution, 602
 - mlpack::distribution::DiagonalGaussianDistribution, 1229
 - mlpack::distribution::DiscreteDistribution, 1235
 - mlpack::distribution::GammaDistribution, 1242, 1243
 - mlpack::distribution::GaussianDistribution, 1248
 - mlpack::distribution::LaplaceDistribution, 1254, 1255
 - mlpack::distribution::RegressionDistribution, 1260
 - mlpack::gmm::DiagonalGMM, 1313
 - mlpack::gmm::GMM, 1328, 1329
- LogSoftMax
 - mlpack::ann::LogSoftMax, 793
- LogSoftMax< InputDataType, OutputDataType >, 792
- LogVolume
 - mlpack::det::DTree, 1217
- LogisticFunction, 789
- LogisticRegression
 - mlpack::regression::LogisticRegression, 1797, 1798
- LogisticRegression< MatType >, 1795
- LogisticRegressionFunction
 - mlpack::regression::LogisticRegressionFunction, 1805
- LogisticRegressionFunction< MatType >, 1803
- LogisticRegressionTestData
 - test_function_tools.hpp, 3086

- Logits
 - mlpack::ann::BernoulliDistribution, 601
- Lookup
 - mlpack::ann::Lookup, 797
- Lookup< InputDataType, OutputDataType >, 795
- Loss
 - mlpack::ann::Reparametrization, 906
- LossVisitor, 800
- LowerBound
 - mlpack::kmeans::DualTreeKMeansStatistic, 1474, 1475
- lsh_search.hpp
 - BOOST_TEMPLATE_CLASS_VERSION, 3003
- M_PI
 - prereqs.hpp, 3077
- MAX_OVERLAP
 - mlpack::tree, 464
- MDOption
 - mlpack::bindings::markdown::MDOption, 986
- MDOption< T >, 985
- MIE
 - mlpack::cv::CVBase, 1130
- MLPACK_ARMA_NO64BIT_WORD
 - arma_config.hpp, 2691
- MLPACK_ARMA_USE_OPENMP
 - arma_config.hpp, 2691
- MLPACK_VERSION_MAJOR
 - src/mlpack/core/util/version.hpp, 2747
- MLPACK_VERSION_MINOR
 - src/mlpack/core/util/version.hpp, 2747
- MLPACK_VERSION_PATCH
 - src/mlpack/core/util/version.hpp, 2747
- MSE, 1143
- macro
 - bindings/markdown/CMakeLists.txt, 2385, 2386
 - bindings/python/CMakeLists.txt, 2387
- MahalanobisDistance
 - mlpack::metric::MahalanobisDistance, 1573, 1574
- MahalanobisDistance< TakeRoot >, 1572
- MajorityClass
 - mlpack::tree::BinaryNumericSplit, 1994
 - mlpack::tree::HoeffdingCategoricalSplit, 2106
 - mlpack::tree::HoeffdingNumericSplit, 2111
 - mlpack::tree::HoeffdingTree, 2122
- MajorityProbability
 - mlpack::tree::BinaryNumericSplit, 1995
 - mlpack::tree::HoeffdingCategoricalSplit, 2107
 - mlpack::tree::HoeffdingNumericSplit, 2112
 - mlpack::tree::HoeffdingTree, 2122, 2123
- MakeAlias
 - mlpack::math, 414, 415
- ManhattanDistance
 - mlpack::metric, 427
- MapFirstPass
 - mlpack::data::DatasetMapper, 1174
 - mlpack::data::IncrementPolicy, 1186
 - mlpack::data::MissingPolicy, 1195
- MapParameterName
 - mlpack::bindings::cli, 298, 299
- MapString
 - mlpack::data::DatasetMapper, 1175
 - mlpack::data::IncrementPolicy, 1186
 - mlpack::data::MissingPolicy, 1195
- MappedType
 - mlpack::data::IncrementPolicy, 1185
 - mlpack::data::MissingPolicy, 1194
- Mapper
 - mlpack::data::Imputer, 1183, 1184
- Mappings
 - mlpack::adaboost::AdaBoostModel, 497
- Mask
 - mlpack::ann::AlphaDropout, 571
- Mat
 - mlpack::tree::BinarySpaceTree, 2002
 - mlpack::tree::CoverTree, 2045
 - mlpack::tree::Octree, 2170
 - mlpack::tree::RectangleTree, 2212
 - mlpack::tree::SpillTree, 2279
- MatUtriCholFactor
 - mlpack::regression::LARS, 1787
- MatrixCompletion, 1558
 - mlpack::matrix_completion::MatrixCompletion, 1559
- MaxDistance
 - mlpack::bound::BallBound, 997
 - mlpack::bound::HRectBound, 1023
 - mlpack::bound::HollowBallBound, 1013, 1014
 - mlpack::tree::BinarySpaceTree, 2013
 - mlpack::tree::CoverTree, 2054, 2055
 - mlpack::tree::ExampleTree, 2085
 - mlpack::tree::Octree, 2179
 - mlpack::tree::RectangleTree, 2224
 - mlpack::tree::SpillTree, 2288
- MaxIterationTermination, 513
 - mlpack::amf::MaxIterationTermination, 514
- MaxIterations
 - mlpack::amf::CompleteIncrementalTermination, 507
 - mlpack::amf::IncompleteIncrementalTermination, 512
 - mlpack::amf::MaxIterationTermination, 516
 - mlpack::amf::SimpleResidueTermination, 532
 - mlpack::amf::SimpleToleranceTermination, 538
 - mlpack::amf::ValidationRMSETermination, 552
 - mlpack::cf::BiasSVDPolicy, 1040
 - mlpack::cf::RandomizedSVDPolicy, 1082, 1083
 - mlpack::cf::RegSVDPolicy, 1091
 - mlpack::cf::SVDPlusPlusPolicy, 1106, 1107
 - mlpack::gmm::EMFit, 1322
 - mlpack::kmeans::KMeans, 1487, 1488

- mlpack::lcc::LocalCoordinateCoding, 1517, 1518
- mlpack::meanshift::MeanShift, 1564
- mlpack::pca::RandomizedBlockKrylovSVDPolicy, 1732
- mlpack::pca::RandomizedSVDPolicy, 1734
- mlpack::perceptron::Perceptron, 1738
- mlpack::sparse_coding::SparseCoding, 1922
- mlpack::svd::RandomizedBlockKrylovSVD, 1937
- mlpack::svd::RandomizedSVD, 1942
- maxIterations
 - mlpack::amf::SimpleResidueTermination, 533
- MaxLeafSize
 - mlpack::tree::RectangleTree, 2224, 2225
- MaxNeighborDistance
 - mlpack::emst::DTBStat, 1271
- MaxNumChildren
 - mlpack::tree::RectangleTree, 2225
- MaxPooling
 - mlpack::ann::MaxPooling, 809
- MaxPooling< InputDataType, OutputDataType >, 808
- MaxPoolingRule, 814
- MaxRPTree
 - mlpack::tree, 454
- MaxRange
 - mlpack::math::ColumnsToBlocks, 1546, 1547
- MaxReward
 - mlpack::rl::RewardClipping, 1898
- MaxSamples
 - mlpack::tree::HoeffdingTree, 2123
- MaxVals
 - mlpack::det::DTree, 1217
- MaxValue
 - mlpack::ann::HardTanH, 743
- MaxVarianceNewCluster, 1489
 - mlpack::kmeans::MaxVarianceNewCluster, 1490
- MaximalInputs
 - mlpack::nn, 436
- Mean
 - mlpack::ann::LayerNorm, 767
 - mlpack::cf::ItemMeanNormalization, 1063
 - mlpack::cf::OverallMeanNormalization, 1074
 - mlpack::cf::UserMeanNormalization, 1113
 - mlpack::cf::ZScoreNormalization, 1116
 - mlpack::distribution::DiagonalGaussianDistribution, 1229
 - mlpack::distribution::GaussianDistribution, 1249
 - mlpack::distribution::LaplaceDistribution, 1255
- mean
 - mlpack::tree::RPTreeMeanSplit::SplitInfo, 2265
- MeanImputation< T >, 1191
- MeanPooling
 - mlpack::ann::MeanPooling, 817
- MeanPooling< InputDataType, OutputDataType >, 814
- MeanPoolingRule, 822
- MeanSPTree
 - mlpack::tree, 456
- MeanShift
 - mlpack::meanshift::MeanShift, 1562
- MeanShift< UseKernel, KernelType, MatType >, 1561
- MeanSpaceSplit< MetricType, MatType >, 2141
- meanSplit
 - mlpack::tree::RPTreeMeanSplit::SplitInfo, 2265
- MeanSplit< BoundType, MatType >, 2142
- MeanSplit< BoundType, MatType >::SplitInfo, 2145
- MeanSplitBallTree
 - mlpack::tree, 455
- MeanSplitKDTree
 - mlpack::tree, 455
- MeanSquaredError
 - mlpack::ann::MeanSquaredError, 823
- MeanSquaredError< InputDataType, OutputDataType >, 823
- Means
 - mlpack::naive_bayes::NaiveBayesClassifier, 1581
- MedianImputation< T >, 1192
- MetalInfoExtractor< MLAlgorithm, MT, PT, WT >, 1140
- MethodFormDetector< Class, MethodForm, 0 >, 1907
- MethodFormDetector< Class, MethodForm, 1 >, 1908
- MethodFormDetector< Class, MethodForm, 2 >, 1908
- MethodFormDetector< Class, MethodForm, 3 >, 1909
- MethodFormDetector< Class, MethodForm, 4 >, 1910
- MethodFormDetector< Class, MethodForm, 5 >, 1910
- MethodFormDetector< Class, MethodForm, 6 >, 1911
- MethodFormDetector< Class, MethodForm, 7 >, 1912
- MethodFormDetector< Class, MethodForm, Additional← ArgsCount >, 1907
- methods/CMakeLists.txt
 - add_subdirectory, 2418
 - set, 2418
- methods/adaboost/CMakeLists.txt
 - set, 2400
- methods/amf/CMakeLists.txt
 - set, 2401
- methods/amf/init_rules/CMakeLists.txt
 - set, 2402
- methods/amf/termination_policies/CMakeLists.txt
 - set, 2403
- methods/amf/update_rules/CMakeLists.txt
 - set, 2403
- methods/ann/CMakeLists.txt
 - set, 2406
- methods/ann/activation_functions/CMakeLists.txt
 - set, 2404
- methods/ann/augmented/CMakeLists.txt
 - set, 2405
- methods/ann/augmented/tasks/CMakeLists.txt
 - set, 2405
- methods/ann/convolution_rules/CMakeLists.txt

- set, 2407
- methods/ann/dists/CMakeLists.txt
 - set, 2407
- methods/ann/init_rules/CMakeLists.txt
 - set, 2408
- methods/ann/layer/CMakeLists.txt
 - set, 2409
- methods/ann/loss_functions/CMakeLists.txt
 - set, 2410
- methods/ann/rbm/CMakeLists.txt
 - set, 2411
- methods/ann/visitor/CMakeLists.txt
 - set, 2412
- methods/approx_kfn/CMakeLists.txt
 - set, 2413
- methods/bias_svd/CMakeLists.txt
 - set, 2413
- methods/block_krylov_svd/CMakeLists.txt
 - set, 2414
- methods/cf/CMakeLists.txt
 - set, 2415
- methods/cf/decomposition_policies/CMakeLists.txt
 - set, 2415
- methods/cf/interpolation_policies/CMakeLists.txt
 - set, 2416
- methods/cf/neighbor_search_policies/CMakeLists.txt
 - set, 2417
- methods/cf/normalization/CMakeLists.txt
 - set, 2417
- methods/dbscan/CMakeLists.txt
 - set, 2419
- methods/decision_stump/CMakeLists.txt
 - set, 2420
- methods/decision_tree/CMakeLists.txt
 - set, 2420, 2421
- methods/det/CMakeLists.txt
 - set, 2421
- methods/emst/CMakeLists.txt
 - set, 2422
- methods/fastmks/CMakeLists.txt
 - set, 2423
- methods/gmm/CMakeLists.txt
 - set, 2424
- methods/hmm/CMakeLists.txt
 - set, 2424
- methods/hoeffding_trees/CMakeLists.txt
 - set, 2425
- methods/kde/CMakeLists.txt
 - set, 2426
- methods/kernel_pca/CMakeLists.txt
 - set, 2427
- methods/kernel_pca/kernel_rules/CMakeLists.txt
 - set, 2427
- methods/kmeans/CMakeLists.txt
 - set, 2428
- methods/lars/CMakeLists.txt
 - set, 2429
- methods/linear_regression/CMakeLists.txt
 - set, 2429, 2430
- methods/linear_svm/CMakeLists.txt
 - set, 2430
- methods/lmnn/CMakeLists.txt
 - set, 2431
- methods/local_coordinate_coding/CMakeLists.txt
 - set, 2432
- methods/logistic_regression/CMakeLists.txt
 - set, 2433
- methods/lsh/CMakeLists.txt
 - set, 2433
- methods/matrix_completion/CMakeLists.txt
 - set, 2434
- methods/mean_shift/CMakeLists.txt
 - set, 2435
- methods/naive_bayes/CMakeLists.txt
 - set, 2435
- methods/nca/CMakeLists.txt
 - set, 2436
- methods/neighbor_search/CMakeLists.txt
 - set, 2437
- methods/nmf/CMakeLists.txt
 - add_cli_executable, 2438
- methods/nystroem_method/CMakeLists.txt
 - set, 2438
- methods/pca/CMakeLists.txt
 - set, 2439
- methods/pca/decomposition_policies/CMakeLists.txt
 - set, 2440
- methods/perceptron/CMakeLists.txt
 - set, 2440
- methods/perceptron/initialization_methods/CMakeLists.txt
 - set, 2441
- methods/perceptron/learning_policies/CMakeLists.txt
 - set, 2442
- methods/preprocess/CMakeLists.txt
 - set, 2442
- methods/quick_svd/CMakeLists.txt
 - set, 2443
- methods/radical/CMakeLists.txt
 - set, 2444
- methods/random_forest/CMakeLists.txt
 - set, 2444
- methods/randomized_svd/CMakeLists.txt
 - set, 2445
- methods/range_search/CMakeLists.txt
 - set, 2446
- methods/rann/CMakeLists.txt
 - set, 2447
- methods/regularized_svd/CMakeLists.txt

- set, 2447
- methods/reinforcement_learning/CMakeLists.txt
 - set, 2448
- methods/reinforcement_learning/environment/CMakeLists.txt
 - set, 2449
- methods/reinforcement_learning/policy/CMakeLists.txt
 - set, 2449, 2450
- methods/reinforcement_learning/replay/CMakeLists.txt
 - set, 2450
- methods/reinforcement_learning/worker/CMakeLists.txt
 - set, 2451
- methods/softmax_regression/CMakeLists.txt
 - set, 2452
- methods/sparse_autoencoder/CMakeLists.txt
 - set, 2452
- methods/sparse_coding/CMakeLists.txt
 - set, 2453
- methods/svdplusplus/CMakeLists.txt
 - set, 2454
- Metric
 - mlpack::bound::BallBound, 998
 - mlpack::bound::HRectBound, 1024
 - mlpack::bound::HollowBallBound, 1014
 - mlpack::fastmks::FastMKS, 1286
 - mlpack::kde::KDE, 1393
 - mlpack::kmeans::KMeans, 1488
 - mlpack::tree::BinarySpaceTree, 2014
 - mlpack::tree::CoverTree, 2055
 - mlpack::tree::ExampleTree, 2086
 - mlpack::tree::Octree, 2180
 - mlpack::tree::RectangleTree, 2225
 - mlpack::tree::SpillTree, 2288
- metric
 - mlpack::neighbor::NeighborSearchRules, 1650
 - mlpack::tree::VantagePointSplit::SplitInfo, 2337
- MetricType
 - mlpack::bound::HollowBallBound, 1009
 - mlpack::tree::VantagePointSplit, 2332
- Mid
 - mlpack::math::RangeType, 1553
- MidpointSpaceSplit< MetricType, MatType >, 2146
- MidpointSplit< BoundType, MatType >, 2147
- MidpointSplit< BoundType, MatType >::SplitInfo, 2150
- MinBaseCases
 - mlpack::tree::GreedySingleTreeTraverser, 2094
- MinDelta
 - mlpack::hpt::HyperParameterTuner, 1378
- MinDistance
 - mlpack::bound::BallBound, 998, 999
 - mlpack::bound::HRectBound, 1024, 1025
 - mlpack::bound::HollowBallBound, 1014, 1015
 - mlpack::tree::BinarySpaceTree, 2014
 - mlpack::tree::CoverTree, 2055, 2056
 - mlpack::tree::ExampleTree, 2086
 - mlpack::tree::Octree, 2180
 - mlpack::tree::RectangleTree, 2225, 2226
 - mlpack::tree::SpillTree, 2288, 2289
- MinLeafSize
 - mlpack::tree::RectangleTree, 2226
- MinNeighborDistance
 - mlpack::emst::DTBStat, 1271, 1272
- MinNumChildren
 - mlpack::tree::RectangleTree, 2227
- MinNumberOfAdditionalArgs
 - mlpack::cv::TrainFormBase4, 1165
 - mlpack::cv::TrainFormBase5, 1167
 - mlpack::cv::TrainFormBase6, 1168
 - mlpack::cv::TrainFormBase7, 1170
- MinRange
 - mlpack::math::ColumnsToBlocks, 1547
- MinResidue
 - mlpack::amf::SimpleResidueTermination, 532, 533
- minResidue
 - mlpack::amf::SimpleResidueTermination, 533
- MinReward
 - mlpack::rl::RewardClipping, 1898
- MinSamples
 - mlpack::tree::HoeffdingTree, 2123
- MinVals
 - mlpack::det::DTree, 1217
- MinValue
 - mlpack::ann::HardTanH, 743
- MinWidth
 - mlpack::bound::BallBound, 999
 - mlpack::bound::HRectBound, 1025
 - mlpack::bound::HollowBallBound, 1015
- MinimalCoverageSweep< SplitPolicy >, 2151
- MinimalCoverageSweep< SplitPolicy >::SweepCost< TreeType >, 2154
- MinimalSplitsNumberSweep< SplitPolicy >, 2155
- MinimalSplitsNumberSweep< SplitPolicy >::SweepCost< typename >, 2157
- MinimumBoundDistance
 - mlpack::tree::BinarySpaceTree, 2014
 - mlpack::tree::CoverTree, 2056
 - mlpack::tree::Octree, 2180
 - mlpack::tree::RectangleTree, 2226
 - mlpack::tree::SpillTree, 2289
- MinimumSamplesReqd
 - mlpack::neighbor::RAUtil, 1699
- MissingPolicy, 1193
 - mlpack::data::MissingPolicy, 1194, 1195
- mlpack, 252
 - CheckMatrices, 254, 255
 - SerializeObject, 255
 - SerializeObjectAll, 255
 - SerializePointerObject, 256

- SerializePointerObjectAll, 256
- TestAllArmadilloSerialization, 256
- TestArmadilloSerialization, 257
- mlpack::Backtrace
 - Backtrace, 975
 - ToString, 975
- mlpack::CLI
 - Add, 1121
 - Aliases, 1122
 - ClearSettings, 1122
 - didParse, 1126
 - doc, 1126
 - functionMap, 1126
 - FunctionMapType, 1121
 - GetParam, 1122
 - GetPrintableParam, 1122
 - GetRawParam, 1123
 - GetSingleton, 1123
 - HasParam, 1123
 - Parameters, 1124
 - ProgramName, 1124
 - programName, 1126
 - RegisterProgramDoc, 1124
 - RestoreSettings, 1125
 - SetPassed, 1125
 - StoreSettings, 1125
 - timer, 1127
- mlpack::Log
 - Assert, 1539
 - cout, 1540
 - Debug, 1540
 - Fatal, 1540
 - Info, 1540
 - Warn, 1541
- mlpack::Timer
 - DisableTiming, 1975
 - EnableTiming, 1975
 - Get, 1976
 - ResetAll, 1976
 - Start, 1976
 - Stop, 1977
- mlpack::Timers
 - Enabled, 1978
 - GetAllTimers, 1979
 - GetState, 1979
 - GetTimer, 1979
 - PrintTimer, 1980
 - Reset, 1980
 - StartTimer, 1980
 - StopAllTimers, 1981
 - StopTimer, 1981
 - Timers, 1978
- mlpack::adaboost, 257
- mlpack::adaboost::AdaBoost
 - AdaBoost, 490, 491
 - Alpha, 491
 - Classify, 491
 - NumClasses, 492
 - serialize, 492
 - Tolerance, 492
 - Train, 492
 - WeakLearner, 493
 - WeakLearners, 493
- mlpack::adaboost::AdaBoostModel
 - ~AdaBoostModel, 496
 - AdaBoostModel, 495, 496
 - Classify, 496
 - Dimensionality, 496, 497
 - Mappings, 497
 - operator=, 497
 - serialize, 497
 - Train, 498
 - WeakLearnerType, 498
 - WeakLearnerTypes, 495
- mlpack::amf, 257
 - NMFALSFactorizer, 259
 - SVDBatchFactorizer, 259
 - SVDBatchLearning::HUpdate< arma::sp_mat >, 261
 - SVDBatchLearning::WUpdate< arma::sp_mat >, 261
 - SVDCompleteIncrementalFactorizer, 260
 - SVDIncompleteIncrementalFactorizer, 260
 - SVDIncompleteIncrementalLearning::HUpdate< arma::sp_mat >, 261
 - SVDIncompleteIncrementalLearning::WUpdate< arma::sp_mat >, 261
- mlpack::amf::AMF
 - AMF, 500
 - Apply, 500
 - InitializeRule, 501
 - TerminationPolicy, 501
 - Update, 502
- mlpack::amf::AverageInitialization
 - AverageInitialization, 503
 - Initialize, 503
 - serialize, 504
- mlpack::amf::CompleteIncrementalTermination
 - CompleteIncrementalTermination, 505
 - Index, 505
 - Initialize, 505, 506
 - IsConverged, 506
 - Iteration, 507
 - MaxIterations, 507
 - TPolicy, 507
- mlpack::amf::GivenInitialization
 - GivenInitialization, 508, 509
 - Initialize, 509

- serialize, 510
- mlpack::amf::IncompleteIncrementalTermination
 - IncompleteIncrementalTermination, 511
 - Index, 511
 - Initialize, 511
 - IsConverged, 512
 - Iteration, 512
 - MaxIterations, 512
 - TPolicy, 513
- mlpack::amf::MaxIterationTermination
 - Index, 515
 - Initialize, 515
 - IsConverged, 515
 - Iteration, 515
 - MaxIterationTermination, 514
 - MaxIterations, 516
- mlpack::amf::NMFALSUpdate
 - HUpdate, 517
 - Initialize, 518
 - NMFALSUpdate, 517
 - serialize, 518
 - WUpdate, 518
- mlpack::amf::NMFMultiplicativeDistanceUpdate
 - HUpdate, 520
 - Initialize, 522
 - NMFMultiplicativeDistanceUpdate, 520
 - serialize, 522
 - WUpdate, 522
- mlpack::amf::NMFMultiplicativeDivergenceUpdate
 - HUpdate, 524
 - Initialize, 525
 - NMFMultiplicativeDivergenceUpdate, 524
 - serialize, 525
 - WUpdate, 525
- mlpack::amf::RandomAcolInitialization
 - Initialize, 527
 - RandomAcolInitialization, 527
 - serialize, 527
- mlpack::amf::RandomInitialization
 - Initialize, 528
 - RandomInitialization, 528
 - serialize, 529
- mlpack::amf::SVDBatchLearning
 - HUpdate, 540
 - Initialize, 541
 - SVDBatchLearning, 540
 - serialize, 541
 - WUpdate, 541
- mlpack::amf::SVDCompleteIncrementalLearning
 - HUpdate, 543
 - Initialize, 543
 - SVDCompleteIncrementalLearning, 543
 - WUpdate, 544
- mlpack::amf::SVDCompleteIncrementalLearning< arma↵
::sp_mat >
 - ~SVDCompleteIncrementalLearning, 545
 - HUpdate, 545
 - Initialize, 546
 - SVDCompleteIncrementalLearning, 545
 - WUpdate, 546
- mlpack::amf::SVDIncompleteIncrementalLearning
 - HUpdate, 548
 - Initialize, 548
 - SVDIncompleteIncrementalLearning, 548
 - WUpdate, 549
- mlpack::amf::SimpleResidueTermination
 - Index, 531
 - Initialize, 531
 - IsConverged, 531
 - Iteration, 532
 - iteration, 533
 - MaxIterations, 532
 - maxIterations, 533
 - MinResidue, 532, 533
 - minResidue, 533
 - nm, 534
 - normOld, 534
 - residue, 534
 - SimpleResidueTermination, 530
- mlpack::amf::SimpleToleranceTermination
 - Index, 536
 - Initialize, 536
 - IsConverged, 536
 - Iteration, 538
 - MaxIterations, 538
 - SimpleToleranceTermination, 535
 - Tolerance, 538
- mlpack::amf::ValidationRMSETermination
 - Index, 551
 - Initialize, 551
 - IsConverged, 552
 - Iteration, 552
 - MaxIterations, 552
 - NumTestPoints, 553
 - Tolerance, 553
 - ValidationRMSETermination, 551
- mlpack::ann, 262
 - CustomLayer, 268
 - Embedding, 269
 - GlorotInitialization, 269
 - HAS_ANY_METHOD_FORM, 272
 - HAS_MEM_FUNC, 272–274
 - HardSigmoidLayer, 269
 - IdentityLayer, 269
 - LayerTypes, 269
 - ReLULayer, 270
 - Residual, 270

- SELU, 270
- SigmoidLayer, 271
- SoftPlusLayer, 271
- TanHLayer, 271
- XavierInitialization, 271
- mlpack::ann::Add
 - Add, 555
 - Backward, 555
 - Delta, 555, 556
 - Forward, 556
 - Gradient, 556, 557
 - OutputParameter, 557
 - Parameters, 557, 558
 - serialize, 558
- mlpack::ann::AddMerge
 - ~AddMerge, 560
 - Add, 561
 - AddMerge, 560
 - Backward, 561
 - Delta, 562
 - Forward, 562
 - Gradient, 563
 - InputParameter, 563
 - Model, 563
 - OutputParameter, 564
 - Parameters, 564
 - Run, 564, 565
 - serialize, 565
- mlpack::ann::AddVisitor
 - AddVisitor, 566
 - operator(), 566
- mlpack::ann::AlphaDropout
 - A, 569
 - AlphaDash, 569
 - AlphaDropout, 568
 - B, 569
 - Backward, 569
 - Delta, 570
 - Deterministic, 570
 - Forward, 570
 - Mask, 571
 - OutputParameter, 571
 - Ratio, 571
 - serialize, 572
- mlpack::ann::AtrousConvolution
 - AtrousConvolution, 574
 - Backward, 575
 - Delta, 576
 - Forward, 576
 - Gradient, 576, 577
 - InputHeight, 577
 - InputWidth, 577, 578
 - OutputHeight, 578
 - OutputParameter, 578, 579
 - OutputWidth, 579
 - Parameters, 579
 - Reset, 580
 - serialize, 580
- mlpack::ann::BRNN
 - Add, 613
 - BRNN, 612
 - Evaluate, 613
 - EvaluateWithGradient, 614
 - Gradient, 614
 - NetworkType, 612
 - NumFunctions, 616
 - Parameters, 616
 - Predict, 616
 - Predictors, 617
 - Reset, 617
 - ResetParameters, 618
 - Responses, 618
 - Rho, 618, 619
 - serialize, 619
 - Shuffle, 619
 - Train, 619, 620
- mlpack::ann::BackwardVisitor
 - BackwardVisitor, 587, 588
 - operator(), 588
- mlpack::ann::BaseLayer
 - Backward, 590
 - BaseLayer, 590
 - Delta, 590, 591
 - Forward, 591
 - OutputParameter, 591
 - serialize, 592
- mlpack::ann::BatchNorm
 - Backward, 594
 - BatchNorm, 594
 - Delta, 596
 - Deterministic, 596
 - Forward, 596
 - Gradient, 597
 - OutputParameter, 597, 598
 - Parameters, 598
 - Reset, 598
 - serialize, 598
 - TrainingMean, 599
 - TrainingVariance, 599
- mlpack::ann::BernoulliDistribution
 - BernoulliDistribution, 600, 601
 - LogProbBackward, 602
 - LogProbability, 602
 - Logits, 601
 - Probability, 602, 604
 - Sample, 604
 - serialize, 604
- mlpack::ann::BilinearInterpolation

- Backward, 607
- BilinearInterpolation, 606
- Delta, 607
- Forward, 607
- OutputParameter, 609
- serialize, 609
- mlpack::ann::CReLU
 - Backward, 654
 - CReLU, 654
 - Delta, 654
 - Forward, 655
 - OutputParameter, 655
 - serialize, 655
- mlpack::ann::Concat
 - ~Concat, 625
 - Add, 625
 - Backward, 625, 626
 - Concat, 623
 - Delta, 626
 - Forward, 627
 - Gradient, 627, 628
 - InputParameter, 628
 - Model, 628
 - OutputParameter, 628, 629
 - Parameters, 629
 - Run, 629
 - serialize, 630
- mlpack::ann::ConcatPerformance
 - Backward, 636
 - ConcatPerformance, 635
 - Delta, 636
 - Forward, 637
 - OutputParameter, 637
 - serialize, 637
- mlpack::ann::Concatenate
 - Backward, 631
 - Concat, 632
 - Concatenate, 631
 - Delta, 632
 - Forward, 633
 - OutputParameter, 633
 - Parameters, 633, 634
 - serialize, 634
- mlpack::ann::ConstInitialization
 - ConstInitialization, 641
 - InitValue, 643
 - initValue, 643
 - Initialize, 642
- mlpack::ann::Constant
 - Backward, 639
 - Constant, 638
 - Delta, 639
 - Forward, 640
 - OutputParameter, 640
 - serialize, 640
- mlpack::ann::Convolution
 - Backward, 646
 - Convolution, 645
 - Delta, 646
 - Forward, 647
 - Gradient, 647
 - InputHeight, 648
 - InputParameter, 648
 - InputWidth, 648, 649
 - OutputHeight, 649
 - OutputParameter, 649, 650
 - OutputWidth, 650
 - Parameters, 650
 - Reset, 651
 - serialize, 651
- mlpack::ann::CopyVisitor
 - operator(), 652
- mlpack::ann::CrossEntropyError
 - Backward, 657
 - CrossEntropyError, 657
 - Eps, 657, 658
 - Forward, 658
 - OutputParameter, 658
 - serialize, 659
- mlpack::ann::DeleteVisitor
 - operator(), 660
- mlpack::ann::DeltaVisitor
 - operator(), 661
- mlpack::ann::DeterministicSetVisitor
 - DeterministicSetVisitor, 662
 - operator(), 662
- mlpack::ann::DiceLoss
 - Backward, 664
 - DiceLoss, 663
 - Forward, 664
 - OutputParameter, 665
 - serialize, 665
 - Smooth, 665
- mlpack::ann::DropConnect
 - ~DropConnect, 668
 - Backward, 668
 - Delta, 669
 - Deterministic, 669
 - DropConnect, 668
 - Forward, 670
 - Gradient, 670, 671
 - Model, 671
 - OutputParameter, 671
 - Parameters, 671, 672
 - Ratio, 672
 - serialize, 672
- mlpack::ann::Dropout
 - Backward, 675

- Delta, 675
- Deterministic, 675
- Dropout, 674
- Forward, 676
- OutputParameter, 676
- Ratio, 676, 677
- serialize, 677
- mlpack::ann::ELU
 - Alpha, 683
 - Backward, 683
 - Delta, 683, 684
 - ELU, 682
 - Forward, 684
 - Lambda, 684
 - OutputParameter, 684, 685
 - serialize, 685
- mlpack::ann::EarthMoverDistance
 - Backward, 679
 - EarthMoverDistance, 678
 - Forward, 679
 - OutputParameter, 679
 - serialize, 680
- mlpack::ann::FFTConvolution
 - Convolution, 706, 707
- mlpack::ann::FFN
 - ~FFN, 695
 - Add, 696
 - Backward, 696
 - Evaluate, 697, 698
 - EvaluateWithGradient, 698, 699
 - FFN, 695
 - Forward, 699, 700
 - Gradient, 700
 - NetworkType, 694
 - NumFunctions, 700
 - operator=, 701
 - Parameters, 701
 - Predict, 701
 - Predictors, 702
 - ResetParameters, 702
 - Responses, 702, 703
 - serialize, 703
 - Shuffle, 703
 - Train, 703, 704
- mlpack::ann::FastLSTM
 - Backward, 688
 - Delta, 689
 - ElemType, 687
 - FastLSTM, 688
 - Forward, 689
 - Gradient, 690
 - InputElemType, 687
 - OutputParameter, 690
 - Parameters, 691
 - Reset, 691
 - ResetCell, 691
 - Rho, 691, 692
 - serialize, 692
- mlpack::ann::FlexibleReLU
 - Alpha, 709
 - Backward, 710
 - Delta, 710
 - FlexibleReLU, 709
 - Forward, 710
 - Gradient, 711
 - OutputParameter, 712
 - Parameters, 712
 - Reset, 712
 - serialize, 713
- mlpack::ann::ForwardVisitor
 - ForwardVisitor, 714
 - operator(), 714
- mlpack::ann::GRU
 - ~GRU, 733
 - Backward, 733
 - Delta, 734
 - Deterministic, 734
 - Forward, 734
 - GRU, 732
 - Gradient, 735
 - Model, 735
 - OutputParameter, 735, 736
 - Parameters, 736
 - ResetCell, 736
 - Rho, 736, 737
 - serialize, 737
- mlpack::ann::GaussianInitialization
 - GaussianInitialization, 715
 - Initialize, 715, 716
- mlpack::ann::Glimpse
 - Backward, 718
 - Delta, 719
 - Deterministic, 719
 - Forward, 720
 - Glimpse, 718
 - InputHeight, 720
 - InputWidth, 720, 721
 - Location, 721
 - OutputHeight, 721
 - OutputParameter, 721, 722
 - OutputWidth, 722
 - serialize, 722
- mlpack::ann::GlorotInitializationType
 - GlorotInitializationType, 724
 - Initialize, 724, 725
- mlpack::ann::GradientSetVisitor
 - GradientSetVisitor, 726
 - operator(), 726

- mlpack::ann::GradientUpdateVisitor
 - GradientUpdateVisitor, 727
 - operator(), 728
- mlpack::ann::GradientVisitor
 - GradientVisitor, 729
 - operator(), 729
- mlpack::ann::GradientZeroVisitor
 - GradientZeroVisitor, 730
 - operator(), 730
- mlpack::ann::HardSigmoidFunction
 - Deriv, 738
 - Fn, 739
- mlpack::ann::HardTanH
 - Backward, 742
 - Delta, 742
 - Forward, 742
 - HardTanH, 741
 - MaxValue, 743
 - MinValue, 743
 - OutputParameter, 743, 744
 - serialize, 744
- mlpack::ann::HelInitialization
 - HelInitialization, 745
 - Initialize, 745, 746
- mlpack::ann::IdentityFunction
 - Deriv, 747, 748
 - Fn, 748, 750
- mlpack::ann::InitTraits
 - UseLayer, 751
- mlpack::ann::InitTraits< KathirvalavakumarSubavathi↔ Initialization >
 - UseLayer, 752
- mlpack::ann::InitTraits< NguyenWidrowInitialization >
 - UseLayer, 752
- mlpack::ann::Join
 - Backward, 754
 - Delta, 754, 755
 - Forward, 755
 - Join, 754
 - OutputParameter, 755
 - serialize, 756
- mlpack::ann::KLDivergence
 - Backward, 760
 - Forward, 760
 - KLDivergence, 760
 - OutputParameter, 761
 - serialize, 761
 - TakeMean, 761
- mlpack::ann::KathirvalavakumarSubavathiInitialization
 - Initialize, 757, 758
 - KathirvalavakumarSubavathiInitialization, 757
- mlpack::ann::LSTM
 - Backward, 803
 - Delta, 804
 - Forward, 804, 805
 - Gradient, 805
 - LSTM, 803
 - OutputParameter, 806
 - Parameters, 806
 - Reset, 806
 - ResetCell, 807
 - Rho, 807
 - serialize, 807
- mlpack::ann::LayerNorm
 - Backward, 764
 - Delta, 765
 - Forward, 765
 - Gradient, 765, 767
 - LayerNorm, 764
 - Mean, 767
 - OutputParameter, 767
 - Parameters, 768
 - Reset, 768
 - serialize, 768
 - Variance, 768
- mlpack::ann::LayerTraits
 - IsBiasLayer, 769
 - IsBinary, 770
 - IsConnection, 770
 - IsLSTMLayer, 770
 - IsOutputLayer, 770
- mlpack::ann::LeakyReLU
 - Alpha, 772
 - Backward, 772
 - Delta, 773
 - Forward, 773
 - LeakyReLU, 772
 - OutputParameter, 774
 - serialize, 774
- mlpack::ann::LecunNormalInitialization
 - Initialize, 775, 776
 - LecunNormalInitialization, 775
- mlpack::ann::Linear
 - Backward, 778
 - Delta, 779
 - Forward, 779
 - Gradient, 780
 - InputParameter, 780
 - Linear, 778
 - OutputParameter, 781
 - Parameters, 781
 - Reset, 781
 - serialize, 782
- mlpack::ann::LinearNoBias
 - Backward, 784
 - Delta, 784
 - Forward, 785
 - Gradient, 785

- InputParameter, 786
- LinearNoBias, 783
- OutputParameter, 786
- Parameters, 786, 787
- Reset, 787
- serialize, 787
- mlpack::ann::LoadOutputParameterVisitor
 - LoadOutputParameterVisitor, 788
 - operator(), 788
- mlpack::ann::LogSoftMax
 - Backward, 793
 - Delta, 794
 - Forward, 794
 - LogSoftMax, 793
 - OutputParameter, 794, 795
 - serialize, 795
- mlpack::ann::LogisticFunction
 - Deriv, 790
 - Fn, 790, 791
 - Inv, 791
- mlpack::ann::Lookup
 - Backward, 797
 - Delta, 797, 798
 - Forward, 798
 - Gradient, 798, 799
 - Lookup, 797
 - OutputParameter, 799
 - Parameters, 799
 - serialize, 800
- mlpack::ann::LossVisitor
 - operator(), 801
- mlpack::ann::MaxPooling
 - Backward, 810
 - Delta, 810
 - Deterministic, 810, 811
 - Forward, 811
 - InputHeight, 811
 - InputWidth, 812
 - MaxPooling, 809
 - OutputHeight, 812
 - OutputParameter, 812, 813
 - OutputWidth, 813
 - serialize, 813
- mlpack::ann::MaxPoolingRule
 - Pooling, 814
- mlpack::ann::MeanPooling
 - Backward, 817
 - Delta, 818
 - Deterministic, 818
 - Forward, 819
 - InputHeight, 819
 - InputWidth, 819, 820
 - MeanPooling, 817
 - OutputHeight, 820
 - OutputParameter, 820
 - OutputWidth, 821
 - serialize, 821
- mlpack::ann::MeanPoolingRule
 - Pooling, 822
 - Unpooling, 822
- mlpack::ann::MeanSquaredError
 - Backward, 824
 - Forward, 824
 - MeanSquaredError, 823
 - OutputParameter, 824, 825
 - serialize, 825
- mlpack::ann::MultiplyConstant
 - Backward, 827
 - Delta, 827
 - Forward, 827
 - MultiplyConstant, 826
 - OutputParameter, 828
 - serialize, 828
- mlpack::ann::MultiplyMerge
 - ~MultiplyMerge, 830
 - Add, 831
 - Backward, 831
 - Delta, 831, 832
 - Forward, 832
 - Gradient, 832
 - Model, 833
 - MultiplyMerge, 830
 - OutputParameter, 833
 - Parameters, 833
 - serialize, 834
- mlpack::ann::NaiveConvolution
 - Convolution, 835, 836
- mlpack::ann::NegativeLogLikelihood
 - Backward, 838
 - Delta, 839
 - Forward, 839
 - InputParameter, 839, 840
 - NegativeLogLikelihood, 838
 - OutputParameter, 840
 - serialize, 840
- mlpack::ann::NetworkInitialization
 - Initialize, 841
 - NetworkInitialization, 841
- mlpack::ann::NguyenWidrowInitialization
 - Initialize, 843, 844
 - NguyenWidrowInitialization, 843
- mlpack::ann::OivsInitialization
 - Initialize, 846
 - OivsInitialization, 845
- mlpack::ann::OrthogonalInitialization
 - Initialize, 849
 - OrthogonalInitialization, 848
- mlpack::ann::OutputHeightVisitor

- operator(), 850
- mlpack::ann::OutputParameterVisitor
 - operator(), 851
- mlpack::ann::OutputWidthVisitor
 - operator(), 852
- mlpack::ann::PReLU
 - Alpha, 857
 - Backward, 858
 - Delta, 858
 - Forward, 858
 - Gradient, 859
 - OutputParameter, 860
 - PReLU, 857
 - Parameters, 860
 - Reset, 860
 - serialize, 861
- mlpack::ann::ParametersSetVisitor
 - operator(), 854
 - ParametersSetVisitor, 853
- mlpack::ann::ParametersVisitor
 - operator(), 855
 - ParametersVisitor, 855
- mlpack::ann::RBM
 - ElemType, 868
 - Evaluate, 869
 - FreeEnergy, 869, 870
 - Gibbs, 870
 - Gradient, 870
 - HiddenBias, 871
 - HiddenMean, 871, 872
 - HiddenSize, 872
 - NetworkType, 868
 - NumFunctions, 872
 - NumSteps, 872
 - Parameters, 872, 873
 - Phase, 873
 - PoolSize, 874
 - RBM, 868
 - Reset, 874
 - SampleHidden, 874
 - SampleSlab, 875
 - SampleSpike, 875
 - SampleVisible, 875, 876
 - serialize, 876
 - Shuffle, 876
 - SlabMean, 876
 - SlabPenalty, 877
 - SpikeBias, 877
 - SpikeMean, 877
 - Train, 878
 - VisibleBias, 878
 - VisibleMean, 878, 879
 - VisiblePenalty, 879
 - VisibleSize, 879
 - Weight, 880
- mlpack::ann::RNN
 - ~RNN, 914
 - Add, 914
 - Evaluate, 915
 - EvaluateWithGradient, 916
 - Gradient, 916
 - NetworkType, 913
 - NumFunctions, 916
 - Parameters, 917
 - Predict, 917
 - Predictors, 918
 - RNN, 913
 - Reset, 918
 - ResetParameters, 918
 - Responses, 918, 919
 - Rho, 919
 - serialize, 919
 - Shuffle, 919
 - Train, 920
- mlpack::ann::RandomInitialization
 - Initialize, 863, 864
 - RandomInitialization, 862, 863
- mlpack::ann::ReconstructionLoss
 - Backward, 882
 - Forward, 882
 - OutputParameter, 882
 - ReconstructionLoss, 881
 - serialize, 883
- mlpack::ann::RectifierFunction
 - Deriv, 884
 - Fn, 885
- mlpack::ann::Recurrent
 - ~Recurrent, 888
 - Backward, 889
 - Delta, 889
 - Deterministic, 889, 890
 - Forward, 890
 - Gradient, 890, 891
 - Model, 891
 - OutputParameter, 891
 - Parameters, 891, 892
 - Recurrent, 887, 888
 - serialize, 892
- mlpack::ann::RecurrentAttention
 - Backward, 895
 - Delta, 895
 - Deterministic, 895, 896
 - Forward, 896
 - Gradient, 896, 897
 - Model, 897
 - OutputParameter, 897
 - Parameters, 897, 898
 - RecurrentAttention, 894

- serialize, 898
- mlpack::ann::ReinforceNormal
 - Backward, 900
 - Delta, 900, 901
 - Deterministic, 901
 - Forward, 901
 - OutputParameter, 902
 - ReinforceNormal, 900
 - Reward, 902
 - serialize, 902
- mlpack::ann::Reparametrization
 - Backward, 905
 - Delta, 905
 - Forward, 905
 - Loss, 906
 - OutputParameter, 906
 - OutputSize, 906, 907
 - Reparametrization, 904
 - serialize, 907
- mlpack::ann::ResetCellVisitor
 - operator(), 908
 - ResetCellVisitor, 908
- mlpack::ann::ResetVisitor
 - operator(), 909
- mlpack::ann::RewardSetVisitor
 - operator(), 910
 - RewardSetVisitor, 910
- mlpack::ann::RunSetVisitor
 - operator(), 922
 - RunSetVisitor, 922
- mlpack::ann::SVDConvolution
 - Convolution, 951, 952
- mlpack::ann::SaveOutputParameterVisitor
 - operator(), 924
 - SaveOutputParameterVisitor, 923
- mlpack::ann::Select
 - Backward, 925
 - Delta, 926
 - Forward, 926
 - OutputParameter, 926, 927
 - Select, 925
 - serialize, 927
- mlpack::ann::Sequential
 - ~Sequential, 930
 - Add, 930
 - Backward, 930
 - DeleteModules, 931
 - Delta, 931
 - Forward, 931
 - Gradient, 932
 - InputParameter, 932
 - Model, 933
 - OutputParameter, 933
 - Parameters, 933
 - Sequential, 929
 - serialize, 934
- mlpack::ann::SetInputHeightVisitor
 - operator(), 935
 - SetInputHeightVisitor, 935
- mlpack::ann::SetInputWidthVisitor
 - operator(), 936
 - SetInputWidthVisitor, 936
- mlpack::ann::SigmoidCrossEntropyError
 - Backward, 938
 - Forward, 938
 - OutputParameter, 939
 - serialize, 939
 - SigmoidCrossEntropyError, 938
- mlpack::ann::SoftplusFunction
 - Deriv, 940, 941
 - Fn, 941, 942
 - Inv, 942
- mlpack::ann::SoftsignFunction
 - Deriv, 944
 - Fn, 945
 - Inv, 946
- mlpack::ann::Subview
 - Backward, 948
 - Delta, 949
 - Forward, 949
 - OutputParameter, 950
 - serialize, 950
 - Subview, 948
- mlpack::ann::SwishFunction
 - Deriv, 953, 954
 - Fn, 954, 955
- mlpack::ann::TanhFunction
 - Deriv, 956, 957
 - Fn, 957
 - Inv, 958
- mlpack::ann::TransposedConvolution
 - Backward, 962
 - Delta, 962
 - Forward, 962
 - Gradient, 963
 - InputHeight, 963
 - InputParameter, 964
 - InputWidth, 964
 - OutputHeight, 964, 965
 - OutputParameter, 965
 - OutputWidth, 965, 966
 - Parameters, 966
 - Reset, 966
 - serialize, 966
 - TransposedConvolution, 961
- mlpack::ann::VRClassReward
 - Add, 969
 - Backward, 969

- Delta, 970
- Deterministic, 970
- Forward, 970
- OutputParameter, 971
- serialize, 971
- VRClassReward, 968
- mlpack::ann::WeightSetVisitor
 - operator(), 973
 - WeightSetVisitor, 972
- mlpack::ann::WeightSizeVisitor
 - operator(), 974
- mlpack::ann::augmented, 274
- mlpack::ann::augmented::scorers, 274
 - SequencePrecision, 274
- mlpack::ann::augmented::tasks, 276
- mlpack::ann::augmented::tasks::AddTask
 - AddTask, 581
 - Generate, 581, 582
- mlpack::ann::augmented::tasks::CopyTask
 - CopyTask, 583
 - Generate, 584
- mlpack::ann::augmented::tasks::SortTask
 - Generate, 586
 - SortTask, 585
- mlpack::bindings, 276
- mlpack::bindings::cli, 276
 - AddToPO, 284, 285
 - DefaultParam, 286
 - DefaultParamImpl, 286, 287
 - DeleteAllocatedMemory, 287
 - DeleteAllocatedMemoryImpl, 287, 288
 - EndProgram, 288
 - GetAllocatedMemory, 288, 289
 - GetBindingName, 289
 - GetParam, 290, 291
 - GetPrintableParam, 292, 293
 - GetPrintableParamName, 293, 294
 - GetPrintableParamValue, 294, 295
 - GetPrintableType, 295–297
 - GetRawParam, 297, 298
 - IgnoreCheck, 298
 - MapParameterName, 298, 299
 - OutputParam, 300
 - OutputParamImpl, 300, 301
 - PARAM_FLAG, 301
 - PARAM_STRING_IN, 302
 - ParamString, 302
 - ParseCommandLine, 302
 - PrintDataset, 302
 - PrintDefault, 303
 - PrintHelp, 303
 - PrintImport, 303
 - PrintModel, 303
 - PrintOutputOptionInfo, 304
 - PrintType, 304
 - PrintTypeDoc, 304, 305
 - PrintTypeDocs, 305
 - PrintValue, 306
 - ProcessOptions, 306
 - ProgramCall, 306
 - SetParam, 307, 308
 - StringTypeParam, 309
 - StringTypeParam< bool >, 309
 - StringTypeParam< double >, 309
 - StringTypeParam< int >, 309
 - StringTypeParam< std::string >, 310
 - StringTypeParam< std::tuple< mlpack::data::↵
DatasetInfo, arma::mat > >, 310
 - StringTypeParamImpl, 310, 311
- mlpack::bindings::cli::CLIOption
 - CLIOption, 976
- mlpack::bindings::cli::ParameterType
 - type, 977
- mlpack::bindings::cli::ParameterType< arma::Col< eT >
>
 - type, 978
- mlpack::bindings::cli::ParameterType< arma::Mat< eT >
>
 - type, 979
- mlpack::bindings::cli::ParameterType< arma::Row< eT >
>
 - type, 980
- mlpack::bindings::cli::ParameterType< std::tuple<
mlpack::data::DatasetMapper< PolicyType,
std::string >, arma::Mat< eT > > >
 - type, 980
- mlpack::bindings::cli::ParameterTypeDeducer
 - type, 981
- mlpack::bindings::cli::ParameterTypeDeducer< true, T >
 - type, 982
- mlpack::bindings::cli::ProgramDoc
 - documentation, 983
 - ProgramDoc, 983
 - programName, 983
- mlpack::bindings::markdown, 311
 - DefaultParam, 314
 - GetBindingName, 314
 - GetParam, 315
 - GetPrintableParam, 315–317
 - GetPrintableParamName, 317, 318
 - GetPrintableParamValue, 318, 319
 - GetPrintableType, 320
 - IgnoreCheck, 320
 - IsSerializable, 321
 - ParamString, 322
 - ParamType, 322
 - PrintDataset, 322
 - PrintDefault, 322

- PrintImport, 322
- PrintLanguage, 323
- PrintModel, 323
- PrintOutputOptionInfo, 323
- PrintTypeDoc, 323
- PrintTypeDocs, 323
- PrintValue, 324
- ProgramCall, 324
- mlpack::bindings::markdown::BindingInfo
 - GetProgramDoc, 984
 - Language, 985
 - RegisterProgramDoc, 985
- mlpack::bindings::markdown::MDOption
 - MDOption, 986
- mlpack::bindings::markdown::ProgramDocWrapper
 - ProgramDocWrapper, 987
- mlpack::bindings::python, 324
 - DefaultParam, 331
 - DefaultParamImpl, 331, 332
 - GetArmaType, 333
 - GetBindingName, 333
 - GetCythonType, 333, 334
 - GetCythonType< bool >, 334
 - GetCythonType< double >, 334
 - GetCythonType< int >, 334
 - GetCythonType< size_t >, 335
 - GetCythonType< std::string >, 335
 - GetNumpyType, 335
 - GetNumpyType< double >, 335
 - GetNumpyType< size_t >, 336
 - GetNumpyTypeChar, 336
 - GetNumpyTypeChar< arma::Col< size_t > >, 336
 - GetNumpyTypeChar< arma::Mat< size_t > >, 336
 - GetNumpyTypeChar< arma::Row< size_t > >, 336
 - GetNumpyTypeChar< arma::mat >, 336
 - GetNumpyTypeChar< arma::rowvec >, 337
 - GetNumpyTypeChar< arma::vec >, 337
 - GetParam, 337
 - GetPrintableParam, 337–339
 - GetPrintableType, 340, 341
 - GetPrintableType< bool >, 341
 - GetPrintableType< double >, 341
 - GetPrintableType< int >, 341
 - GetPrintableType< size_t >, 342
 - GetPrintableType< std::string >, 342
 - IgnoreCheck, 342, 343
 - ImportDecl, 343, 344
 - ParamString, 344
 - PrintClassDefn, 344, 345
 - PrintDataset, 346
 - PrintDefault, 346
 - PrintDefn, 346
 - PrintDoc, 346
 - PrintImport, 347
 - PrintInputOptions, 347
 - PrintInputProcessing, 347–350
 - PrintModel, 350
 - PrintOutputOptionInfo, 351
 - PrintOutputOptions, 351
 - PrintOutputProcessing, 351–353
 - PrintPYX, 354
 - PrintTypeDoc, 354, 355
 - PrintValue, 355, 356
 - ProgramCall, 356
 - programName, 357
 - SerializeIn, 356
 - SerializeOut, 356
 - StripType, 357
- mlpack::bindings::python::PyOption
 - PyOption, 988
- mlpack::bindings::tests, 357
 - CleanMemory, 359
 - DeleteAllocatedMemory, 359
 - DeleteAllocatedMemoryImpl, 359, 360
 - GetAllocatedMemory, 360, 361
 - GetParam, 361
 - GetPrintableParam, 362, 363
 - IgnoreCheck, 364
 - programName, 364
- mlpack::bindings::tests::ProgramDoc
 - documentation, 989
 - ProgramDoc, 989
 - programName, 989
- mlpack::bindings::tests::TestOption
 - TestOption, 990
- mlpack::bound, 364
- mlpack::bound::BallBound
 - ~BallBound, 995
 - BallBound, 994, 995
 - Center, 995, 996
 - Contains, 996
 - Diameter, 997
 - Dim, 997
 - ElemType, 993
 - MaxDistance, 997
 - Metric, 998
 - MinDistance, 998, 999
 - MinWidth, 999
 - operator=, 999
 - operator[], 999
 - operator|=, 999, 1000
 - Radius, 1000
 - RangeDistance, 1000, 1001
 - serialize, 1001
 - Vec, 994
- mlpack::bound::BoundTraits
 - HasTightBounds, 1002

- mlpack::bound::BoundTraits< BallBound< MetricType, VecType > >
 - HasTightBounds, 1003
- mlpack::bound::BoundTraits< CellBound< MetricType, ElemType > >
 - HasTightBounds, 1004
- mlpack::bound::BoundTraits< HRectBound< MetricType, ElemType > >
 - HasTightBounds, 1005
- mlpack::bound::BoundTraits< HollowBallBound< MetricType, ElemType > >
 - HasTightBounds, 1005
- mlpack::bound::HRectBound
 - ~HRectBound, 1021
 - Center, 1022
 - Clear, 1022
 - Contains, 1022
 - Diameter, 1023
 - Dim, 1023
 - HRectBound, 1020, 1021
 - MaxDistance, 1023
 - Metric, 1024
 - MinDistance, 1024, 1025
 - MinWidth, 1025
 - operator &, 1025
 - operator &=, 1025
 - operator=, 1026
 - operator[], 1026
 - operator| =, 1026, 1027
 - Overlap, 1027
 - RangeDistance, 1027
 - serialize, 1028
 - Volume, 1028
- mlpack::bound::HollowBallBound
 - ~HollowBallBound, 1010
 - Center, 1010, 1011
 - Contains, 1011, 1012
 - Diameter, 1012
 - Dim, 1012
 - HollowBallBound, 1009, 1010
 - HollowCenter, 1012, 1013
 - InnerRadius, 1013
 - MaxDistance, 1013, 1014
 - Metric, 1014
 - MetricType, 1009
 - MinDistance, 1014, 1015
 - MinWidth, 1015
 - operator=, 1015
 - operator[], 1015
 - operator| =, 1016
 - OuterRadius, 1016, 1017
 - RangeDistance, 1017
 - serialize, 1018
- mlpack::bound::addr, 365
- AddressToPoint, 365
- CompareAddresses, 367
- Contains, 367
- PointToAddress, 367
- mlpack::bound::meta, 368
- mlpack::bound::meta::IsLMetric
 - Value, 1029
- mlpack::bound::meta::IsLMetric< metric::LMetric< Power, TakeRoot > >
 - Value, 1030
- mlpack::cf, 368
 - ArmaSVDFactorizer, 370
 - EuclideanSearch, 370
- mlpack::cf::AverageInterpolation
 - AverageInterpolation, 1031
 - GetWeights, 1031
- mlpack::cf::BatchSVDPolicy
 - Apply, 1033
 - GetNeighborhood, 1033
 - GetRating, 1034
 - GetRatingOfUser, 1034
 - H, 1034
 - serialize, 1035
 - W, 1035
- mlpack::cf::BiasSVDPolicy
 - Alpha, 1037
 - Apply, 1038
 - BiasSVDPolicy, 1037
 - GetNeighborhood, 1038
 - GetRating, 1039
 - GetRatingOfUser, 1039
 - H, 1040
 - Lambda, 1040
 - MaxIterations, 1040
 - P, 1041
 - Q, 1041
 - serialize, 1041
 - W, 1041
- mlpack::cf::CFModel
 - ~CFModel, 1043
 - CFModel, 1043
 - CFPtr, 1043
 - GetRecommendations, 1043, 1044
 - Predict, 1044
 - serialize, 1044
 - Train, 1044
- mlpack::cf::CFTYPE
 - CFTYPE, 1047
 - CleanData, 1048
 - CleanedData, 1048
 - Decomposition, 1048
 - GetRecommendations, 1048, 1049
 - Normalization, 1049
 - NumUsersForSimilarity, 1049, 1050

- Predict, 1050
- Rank, 1051
- serialize, 1051
- Train, 1052
- mlpack::cf::CombinedNormalization
 - CombinedNormalization, 1054
 - Denormalize, 1054, 1055
 - Normalizations, 1055
 - Normalize, 1056
 - serialize, 1056
 - TupleType, 1054
- mlpack::cf::CosineSearch
 - CosineSearch, 1057
 - Search, 1058
- mlpack::cf::DeleteVisitor
 - operator(), 1059
- mlpack::cf::GetValueVisitor
 - operator(), 1060
- mlpack::cf::ItemMeanNormalization
 - Denormalize, 1062
 - ItemMeanNormalization, 1062
 - Mean, 1063
 - Normalize, 1063
 - serialize, 1063
- mlpack::cf::LMetricSearch
 - LMetricSearch, 1065
 - NeighborSearchType, 1065
 - Search, 1065
- mlpack::cf::NMFPolicy
 - Apply, 1067
 - GetNeighborhood, 1067
 - GetRating, 1068
 - GetRatingOfUser, 1068
 - H, 1069
 - serialize, 1069
 - W, 1069
- mlpack::cf::NoNormalization
 - Denormalize, 1070, 1071
 - NoNormalization, 1070
 - Normalize, 1071
 - serialize, 1071
- mlpack::cf::OverallMeanNormalization
 - Denormalize, 1073
 - Mean, 1074
 - Normalize, 1074
 - OverallMeanNormalization, 1073
 - serialize, 1074
- mlpack::cf::PearsonSearch
 - PearsonSearch, 1076
 - Search, 1076
- mlpack::cf::PredictVisitor
 - operator(), 1078
 - PredictVisitor, 1078
- mlpack::cf::RandomizedSVDPolicy
 - Apply, 1080
 - GetNeighborhood, 1081
 - GetRating, 1081
 - GetRatingOfUser, 1082
 - H, 1082
 - IteratedPower, 1082
 - MaxIterations, 1082, 1083
 - RandomizedSVDPolicy, 1080
 - serialize, 1083
 - W, 1083
- mlpack::cf::RecommendationVisitor
 - operator(), 1085
 - RecommendationVisitor, 1084
- mlpack::cf::RegSVDPolicy
 - Apply, 1089
 - GetNeighborhood, 1090
 - GetRating, 1090
 - GetRatingOfUser, 1090
 - H, 1091
 - MaxIterations, 1091
 - RegSVDPolicy, 1089
 - serialize, 1091
 - W, 1092
- mlpack::cf::RegressionInterpolation
 - GetWeights, 1087
 - RegressionInterpolation, 1086
- mlpack::cf::SVDCompletePolicy
 - Apply, 1095
 - GetNeighborhood, 1096
 - GetRating, 1096
 - GetRatingOfUser, 1097
 - H, 1097
 - serialize, 1097
 - W, 1097
- mlpack::cf::SVDIncompletePolicy
 - Apply, 1099
 - GetNeighborhood, 1099
 - GetRating, 1100
 - GetRatingOfUser, 1100
 - H, 1101
 - serialize, 1101
 - W, 1101
- mlpack::cf::SVDPlusPlusPolicy
 - Alpha, 1103, 1104
 - Apply, 1104
 - GetNeighborhood, 1104
 - GetRating, 1105
 - GetRatingOfUser, 1105
 - H, 1106
 - ImplicitData, 1106
 - Lambda, 1106
 - MaxIterations, 1106, 1107
 - P, 1107
 - Q, 1107

- SVDPlusPlusPolicy, 1103
- serialize, 1107
- W, 1107
- Y, 1108
- mlpack::cf::SVDWrapper
 - Apply, 1109
 - SVDWrapper, 1109
- mlpack::cf::SimilarityInterpolation
 - GetWeights, 1093
 - SimilarityInterpolation, 1093
- mlpack::cf::UserMeanNormalization
 - Denormalize, 1112
 - Mean, 1113
 - Normalize, 1113
 - serialize, 1113
 - UserMeanNormalization, 1112
- mlpack::cf::ZScoreNormalization
 - Denormalize, 1115
 - Mean, 1116
 - Normalize, 1116
 - serialize, 1116
 - Stddev, 1117
 - ZScoreNormalization, 1115
- mlpack::cv, 371
 - AssertSizes, 372
 - AverageStrategy, 372
- mlpack::cv::Accuracy
 - Evaluate, 1128
 - NeedsMinimization, 1128
- mlpack::cv::CVBase
 - AssertDataConsistency, 1131
 - AssertWeightsConsistency, 1131
 - CVBase, 1130, 1131
 - MIE, 1130
 - Train, 1131, 1132
- mlpack::cv::F1
 - Evaluate, 1133
 - NeedsMinimization, 1134
- mlpack::cv::KFoldCV
 - Evaluate, 1139
 - KFoldCV, 1136–1138
 - Model, 1139
 - Shuffle, 1139
- mlpack::cv::MSE
 - Evaluate, 1143
 - NeedsMinimization, 1144
- mlpack::cv::MetaInfoExtractor
 - IsSupported, 1141
 - PredictionsType, 1141
 - SupportsWeights, 1142
 - TakesDatasetInfo, 1142
 - TakesNumClasses, 1142
 - WeightsType, 1141
- mlpack::cv::NotFoundMethodForm
 - PredictionsType, 1144
 - WeightsType, 1144
- mlpack::cv::Precision
 - Evaluate, 1146
 - NeedsMinimization, 1146
- mlpack::cv::Recall
 - Evaluate, 1148
 - NeedsMinimization, 1148
- mlpack::cv::SelectMethodForm< MLAlgorithm >::FromType, 1150
- mlpack::cv::SelectMethodForm< MLAlgorithm, HasMethodForm, HMFs... >::FromType, 1151
- mlpack::cv::SimpleCV
 - Evaluate, 1157
 - Model, 1157
 - SimpleCV, 1153–1156
- mlpack::cv::TrainFormBase4
 - MinNumberOfAdditionalArgs, 1165
 - PredictionsType, 1164
 - Type, 1165
 - WeightsType, 1165
- mlpack::cv::TrainFormBase5
 - MinNumberOfAdditionalArgs, 1167
 - PredictionsType, 1166
 - Type, 1166
 - WeightsType, 1166
- mlpack::cv::TrainFormBase6
 - MinNumberOfAdditionalArgs, 1168
 - PredictionsType, 1168
 - Type, 1168
 - WeightsType, 1168
- mlpack::cv::TrainFormBase7
 - MinNumberOfAdditionalArgs, 1170
 - PredictionsType, 1169
 - Type, 1169
 - WeightsType, 1170
- mlpack::data, 373
 - Binimize, 377
 - ConfusionMatrix, 378
 - DatasetInfo, 376
 - Datatype, 376
 - Extension, 379
 - format, 376
 - HAS_EXACT_METHOD_FORM, 379
 - IsNaNInf, 379
 - Load, 379–383
 - LoadARFF, 384
 - NormalizeLabels, 385
 - OneHotEncoding, 385
 - RevertLabels, 385
 - Save, 386, 387
 - Split, 387–389
- mlpack::data::CustomImputation

- CustomImputation, 1171
- Impute, 1171
- mlpack::data::DatasetMapper
 - DatasetMapper, 1173, 1174
 - Dimensionality, 1174
 - MapFirstPass, 1174
 - MapString, 1175
 - NumMappings, 1175
 - NumUnmappings, 1175
 - Policy, 1175, 1176
 - serialize, 1176
 - SetDimensionality, 1176
 - Type, 1177
 - UnmapString, 1177
 - UnmapValue, 1178
- mlpack::data::HasSerialize
 - chk, 1179
 - no, 1179
 - value, 1180
 - yes, 1179
- mlpack::data::HasSerializeFunction
 - NonStaticSerialize, 1181
 - StaticSerialize, 1181
 - value, 1181
- mlpack::data::Imputer
 - Impute, 1183
 - Imputer, 1182, 1183
 - Mapper, 1183, 1184
 - Strategy, 1184
- mlpack::data::IncrementPolicy
 - IncrementPolicy, 1185
 - MapFirstPass, 1186
 - MapString, 1186
 - MappedType, 1185
 - NeedsFirstPass, 1187
- mlpack::data::ListwiseDeletion
 - Impute, 1188
- mlpack::data::LoadCSV
 - GetMatrixSize, 1189
 - GetTransposeMatrixSize, 1190
 - Load, 1190
 - LoadCSV, 1189
- mlpack::data::MeanImputation
 - Impute, 1191
- mlpack::data::MedianImputation
 - Impute, 1193
- mlpack::data::MissingPolicy
 - MapFirstPass, 1195
 - MapString, 1195
 - MappedType, 1194
 - MissingPolicy, 1194, 1195
 - NeedsFirstPass, 1197
- mlpack::dbscan, 390
- mlpack::dbscan::DBSCAN
 - Cluster, 1199, 1200
 - DBSCAN, 1198
- mlpack::dbscan::OrderedPointSelection
 - Select, 1201
- mlpack::dbscan::RandomPointSelection
 - Select, 1202
- mlpack::decision_stump, 390
- mlpack::decision_stump::DecisionStump
 - BinLabels, 1205
 - Classify, 1205
 - DecisionStump, 1203, 1204
 - serialize, 1205
 - Split, 1206
 - SplitDimension, 1206
 - Train, 1206, 1207
- mlpack::det, 390
 - PrintLeafMembership, 391
 - PrintVariableImportance, 391
 - Trainer, 392
- mlpack::det::DTree
 - ~DTree, 1213
 - AlphaUpper, 1213
 - BucketTag, 1214
 - Child, 1214
 - ChildPtr, 1214
 - ComputeValue, 1214
 - ComputeVariableImportance, 1215
 - DTree, 1211–1213
 - ElemType, 1210
 - End, 1215
 - FindBucket, 1215
 - Grow, 1216
 - Left, 1216
 - LogNegError, 1217
 - LogNegativeError, 1216
 - LogVolume, 1217
 - MaxVals, 1217
 - MinVals, 1217
 - NumChildren, 1217
 - operator=, 1218
 - PruneAndUpdate, 1218
 - Ratio, 1219
 - Right, 1219
 - Root, 1219
 - serialize, 1219
 - SplitDim, 1220
 - SplitValue, 1220
 - Start, 1220
 - StatType, 1210
 - SubtreeLeaves, 1220
 - SubtreeLeavesLogNegError, 1220
 - TagTree, 1221
 - VecType, 1210
 - WithinRange, 1221

- mlpack::det::PathCacher
 - BuildString, 1224
 - Enter, 1224
 - format, 1225
 - Leave, 1224
 - NumNodes, 1224
 - ParentOf, 1225
 - path, 1225
 - pathCache, 1225
 - PathCacheType, 1223
 - PathCacher, 1223
 - PathFor, 1225
 - PathFormat, 1223
 - PathType, 1223
- mlpack::distribution, 392
- mlpack::distribution::DiagonalGaussianDistribution
 - Covariance, 1228
 - DiagonalGaussianDistribution, 1227, 1228
 - Dimensionality, 1228
 - LogProbability, 1229
 - Mean, 1229
 - Probability, 1230
 - Random, 1230
 - serialize, 1231
 - Train, 1231
- mlpack::distribution::DiscreteDistribution
 - Dimensionality, 1234
 - DiscreteDistribution, 1233, 1234
 - LogProbability, 1235
 - Probabilities, 1236
 - Probability, 1236, 1237
 - Random, 1237
 - serialize, 1237
 - Train, 1237, 1238
- mlpack::distribution::GammaDistribution
 - ~GammaDistribution, 1241
 - Alpha, 1241
 - Beta, 1241, 1242
 - Dimensionality, 1242
 - GammaDistribution, 1240
 - LogProbability, 1242, 1243
 - Probability, 1243, 1244
 - Random, 1244
 - Train, 1244, 1245
- mlpack::distribution::GaussianDistribution
 - Covariance, 1247, 1248
 - Dimensionality, 1248
 - GaussianDistribution, 1247
 - LogProbability, 1248
 - Mean, 1249
 - Probability, 1249
 - Random, 1250
 - serialize, 1250
 - Train, 1250, 1251
- mlpack::distribution::LaplaceDistribution
 - Dimensionality, 1254
 - Estimate, 1254
 - LaplaceDistribution, 1252, 1253
 - LogProbability, 1254, 1255
 - Mean, 1255
 - Probability, 1255, 1256
 - Random, 1256
 - Scale, 1256, 1257
 - serialize, 1257
- mlpack::distribution::RegressionDistribution
 - Dimensionality, 1260
 - Err, 1260
 - LogProbability, 1260
 - Parameters, 1261
 - Predict, 1261
 - Probability, 1262
 - RegressionDistribution, 1259
 - Rf, 1262
 - serialize, 1262
 - Train, 1263
- mlpack::emst, 393
- mlpack::emst::DTBRules
 - BaseCase, 1265
 - BaseCases, 1265, 1266
 - DTBRules, 1265
 - Rescore, 1266
 - Score, 1267
 - Scores, 1268
 - TraversalInfo, 1268
 - TraversalInfoType, 1265
- mlpack::emst::DTBStat
 - Bound, 1270
 - ComponentMembership, 1271
 - DTBStat, 1269, 1270
 - MaxNeighborDistance, 1271
 - MinNeighborDistance, 1271, 1272
- mlpack::emst::DualTreeBoruvka
 - ~DualTreeBoruvka, 1275
 - ComputeMST, 1275
 - DualTreeBoruvka, 1274
 - Tree, 1273
- mlpack::emst::EdgePair
 - Distance, 1276, 1277
 - EdgePair, 1276
 - Greater, 1277
 - Lesser, 1277
- mlpack::emst::UnionFind
 - ~UnionFind, 1279
 - Find, 1279
 - Union, 1279
 - UnionFind, 1278
- mlpack::fastmks, 394
- mlpack::fastmks::FastMKSMModel

- ~FastMKModel, 1294
- BuildModel, 1295
- FastMKModel, 1294
- KernelType, 1295
- KernelTypes, 1292
- Naive, 1295, 1296
- operator=, 1296
- Search, 1296, 1297
- serialize, 1297
- SingleMode, 1297
- mlpack::fastmks::FastMKRules
 - BaseCase, 1300
 - BaseCases, 1300
 - FastMKRules, 1299
 - GetResults, 1300
 - Rescore, 1301
 - Score, 1301, 1302
 - Scores, 1302
 - TraversalInfo, 1302, 1303
 - TraversalInfoType, 1299
- mlpack::fastmks::FastMKStat
 - Bound, 1304, 1305
 - FastMKStat, 1304
 - LastKernel, 1305
 - LastKernelNode, 1305
 - SelfKernel, 1306
 - serialize, 1306
- mlpack::fastmks::FastMKS
 - ~FastMKS, 1286
 - FastMKS, 1283–1286
 - Metric, 1286
 - Naive, 1286, 1287
 - operator=, 1287
 - Search, 1287, 1288
 - serialize, 1289
 - SingleMode, 1289
 - Train, 1289–1291
 - Tree, 1283
- mlpack::gmm, 394
- mlpack::gmm::DiagonalConstraint
 - ApplyConstraint, 1307
 - serialize, 1307
- mlpack::gmm::DiagonalGMM
 - Classify, 1311
 - Component, 1312
 - DiagonalGMM, 1310, 1311
 - Dimensionality, 1312
 - Gaussians, 1312
 - LogProbability, 1313
 - operator=, 1313
 - Probability, 1314
 - Random, 1314
 - serialize, 1314
 - Train, 1315
 - Weights, 1316
- mlpack::gmm::EMFit
 - Clusterer, 1320
 - Constraint, 1321
 - EMFit, 1320
 - Estimate, 1321, 1322
 - MaxIterations, 1322
 - serialize, 1322
 - Tolerance, 1323
- mlpack::gmm::EigenvalueRatioConstraint
 - ApplyConstraint, 1318
 - EigenvalueRatioConstraint, 1317
 - serialize, 1318
- mlpack::gmm::GMM
 - Classify, 1327
 - Component, 1327, 1328
 - Dimensionality, 1328
 - GMM, 1325–1327
 - Gaussians, 1328
 - LogProbability, 1328, 1329
 - operator=, 1329
 - Probability, 1329, 1330
 - Random, 1330
 - serialize, 1330
 - Train, 1330, 1331
 - Weights, 1332
- mlpack::gmm::NoConstraint
 - ApplyConstraint, 1333
 - serialize, 1333
- mlpack::gmm::PositiveDefiniteConstraint
 - ApplyConstraint, 1334
 - serialize, 1335
- mlpack::hmm, 395
 - HMMType, 396
 - LoadHMMAndPerformAction, 397
 - SaveHMM, 397
- mlpack::hmm::HMMModel
 - ~HMMModel, 1351
 - DiagGMMHMM, 1352
 - DiscreteHMM, 1352
 - GMMHMM, 1352
 - GaussianHMM, 1352
 - HMMModel, 1351
 - operator=, 1353
 - PerformAction, 1353
 - serialize, 1353
 - Type, 1353
- mlpack::hmm::HMMRegression
 - Estimate, 1357, 1358
 - Filter, 1358
 - HMMRegression, 1356
 - LogLikelihood, 1359
 - Predict, 1359
 - Smooth, 1360

- Train, 1360, 1361
- mlpack::hmm::HMM
 - Backward, 1339
 - Dimensionality, 1341
 - Emission, 1341
 - emission, 1349
 - Estimate, 1341, 1342
 - Filter, 1343
 - Forward, 1343
 - Generate, 1343
 - HMM, 1338, 1339
 - Initial, 1344
 - LogEstimate, 1344
 - LogLikelihood, 1346
 - Predict, 1346
 - serialize, 1347
 - Smooth, 1347
 - Tolerance, 1347
 - Train, 1348
 - Transition, 1349
 - transition, 1350
- mlpack::hpt, 397
 - Fixed, 398
 - TupleOfHyperParameters, 398
- mlpack::hpt::CVFunction
 - BestModel, 1364
 - CVFunction, 1362
 - Evaluate, 1364
 - Gradient, 1364
- mlpack::hpt::DeduceHyperParameterTypes< PreFixed←
Arg< T >, Args... >
 - TupleType, 1367
- mlpack::hpt::DeduceHyperParameterTypes< PreFixed←
Arg< T >, Args... >::ResultHolder
 - TupleType, 1368
- mlpack::hpt::DeduceHyperParameterTypes< T, Args... >
 - TupleType, 1369
- mlpack::hpt::DeduceHyperParameterTypes< T, Args...
>::IsCollectionType
 - Check, 1370
 - No, 1370
 - value, 1370
 - Yes, 1370
- mlpack::hpt::DeduceHyperParameterTypes< T, Args...
>::ResultHPTType< ArithmeticType, true >
 - Type, 1372
- mlpack::hpt::DeduceHyperParameterTypes< T, Args...
>::ResultHPTType< CollectionType, false >
 - Type, 1373
- mlpack::hpt::DeduceHyperParameterTypes< T, Args...
>::ResultHolder
 - TupleType, 1371
- mlpack::hpt::DeduceHyperParameterTypes::ResultHolder
 - TupleType, 1366
- mlpack::hpt::FixedArg
 - index, 1374
 - value, 1374
- mlpack::hpt::HyperParameterTuner
 - BestModel, 1377
 - BestObjective, 1377
 - HyperParameterTuner, 1377
 - MinDelta, 1378
 - Optimize, 1378
 - Optimizer, 1379
 - RelativeDelta, 1379
- mlpack::hpt::IsPreFixedArg
 - value, 1380
- mlpack::hpt::PreFixedArg
 - Type, 1381
 - value, 1382
- mlpack::hpt::PreFixedArg< T & >
 - Type, 1383
 - value, 1383
- mlpack::kde, 399
 - KDEMode, 400
 - KDEType, 400
- mlpack::kde::DeleteVisitor
 - operator(), 1384
- mlpack::kde::DualBiKDE
 - DualBiKDE, 1385
 - KDETypeT, 1385
 - operator(), 1385
- mlpack::kde::DualMonoKDE
 - DualMonoKDE, 1387
 - KDETypeT, 1387
 - operator(), 1387
- mlpack::kde::KDEModel
 - ~KDEModel, 1400
 - AbsoluteError, 1400
 - Bandwidth, 1400, 1401
 - BuildModel, 1401
 - Evaluate, 1401, 1402
 - KDEModel, 1399, 1400
 - KernelType, 1402
 - KernelTypes, 1398
 - Mode, 1402, 1403
 - operator=, 1403
 - RelativeError, 1403
 - serialize, 1403
 - TreeType, 1404
 - TreeTypes, 1399
- mlpack::kde::KDERules
 - BaseCase, 1406
 - BaseCases, 1406
 - KDERules, 1406
 - Rescore, 1407
 - Score, 1407
 - Scores, 1407

- TraversalInfo, 1408
- TraversalInfoType, 1405
- mlpack::kde::KDEStat
 - Centroid, 1409
 - KDEStat, 1409
 - serialize, 1409
 - SetCentroid, 1410
 - ValidCentroid, 1410
- mlpack::kde::KDE
 - ~KDE, 1391
 - AbsoluteError, 1391
 - Evaluate, 1391, 1392
 - IsTrained, 1393
 - KDE, 1390
 - Kernel, 1393
 - Metric, 1393
 - Mode, 1394
 - operator=, 1394
 - OwnsReferenceTree, 1395
 - ReferenceTree, 1395
 - RelativeError, 1395
 - serialize, 1395
 - Train, 1396
 - Tree, 1389
- mlpack::kde::KernelNormalizer
 - ApplyNormalizer, 1411
- mlpack::kde::ModeVisitor
 - operator(), 1412
- mlpack::kde::TrainVisitor
 - operator(), 1413
 - TrainVisitor, 1413
- mlpack::kernel, 401
- mlpack::kernel::CauchyKernel
 - CauchyKernel, 1415
 - Evaluate, 1415
 - serialize, 1415
- mlpack::kernel::CosineDistance
 - Evaluate, 1416
 - serialize, 1417
- mlpack::kernel::EpanechnikovKernel
 - ConvolutionIntegral, 1418
 - EpanechnikovKernel, 1418
 - Evaluate, 1419
 - Gradient, 1419
 - GradientForSquaredDistance, 1420
 - Normalizer, 1420
 - serialize, 1420
- mlpack::kernel::ExampleKernel
 - ConvolutionIntegral, 1422
 - Evaluate, 1423
 - ExampleKernel, 1422
 - Normalizer, 1423
 - serialize, 1424
- mlpack::kernel::GaussianKernel
 - Bandwidth, 1426
 - ConvolutionIntegral, 1426
 - Evaluate, 1427
 - Gamma, 1428
 - GaussianKernel, 1425
 - Gradient, 1428
 - GradientForSquaredDistance, 1428
 - Normalizer, 1429
 - serialize, 1429
- mlpack::kernel::HyperbolicTangentKernel
 - Evaluate, 1431
 - HyperbolicTangentKernel, 1430, 1431
 - Offset, 1432
 - Scale, 1432
 - serialize, 1433
- mlpack::kernel::KMeansSelection
 - Select, 1442
- mlpack::kernel::KernelTraits
 - IsNormalized, 1434
 - UsesSquaredDistance, 1434
- mlpack::kernel::KernelTraits< CauchyKernel >
 - IsNormalized, 1435
- mlpack::kernel::KernelTraits< CosineDistance >
 - IsNormalized, 1435
 - UsesSquaredDistance, 1436
- mlpack::kernel::KernelTraits< EpanechnikovKernel >
 - IsNormalized, 1437
 - UsesSquaredDistance, 1437
- mlpack::kernel::KernelTraits< GaussianKernel >
 - IsNormalized, 1438
 - UsesSquaredDistance, 1438
- mlpack::kernel::KernelTraits< LaplacianKernel >
 - IsNormalized, 1439
 - UsesSquaredDistance, 1439
- mlpack::kernel::KernelTraits< SphericalKernel >
 - IsNormalized, 1440
 - UsesSquaredDistance, 1440
- mlpack::kernel::KernelTraits< TriangularKernel >
 - IsNormalized, 1441
 - UsesSquaredDistance, 1441
- mlpack::kernel::LaplacianKernel
 - Bandwidth, 1444
 - Evaluate, 1444, 1445
 - Gradient, 1445
 - LaplacianKernel, 1443
 - serialize, 1446
- mlpack::kernel::LinearKernel
 - Evaluate, 1447
 - LinearKernel, 1447
 - serialize, 1448
- mlpack::kernel::NystroemMethod
 - Apply, 1449
 - GetKernelMatrix, 1449, 1450
 - NystroemMethod, 1449

- mlpack::kernel::OrderedSelection
 - Select, 1451
- mlpack::kernel::PSpectrumStringKernel
 - Counts, 1456
 - Evaluate, 1456
 - P, 1456, 1457
 - PSpectrumStringKernel, 1455
- mlpack::kernel::PolynomialKernel
 - Degree, 1452
 - Evaluate, 1453
 - Offset, 1453
 - PolynomialKernel, 1452
 - serialize, 1454
- mlpack::kernel::RandomSelection
 - Select, 1457
- mlpack::kernel::SphericalKernel
 - ConvolutionIntegral, 1459
 - Evaluate, 1459, 1460
 - Gradient, 1460
 - Normalizer, 1461
 - serialize, 1461
 - SphericalKernel, 1459
- mlpack::kernel::TriangularKernel
 - Bandwidth, 1462
 - Evaluate, 1463
 - Gradient, 1464
 - serialize, 1464
 - TriangularKernel, 1462
- mlpack::kmeans, 402
 - CoverTreeDualTreeKMeans, 404
 - DefaultDualTreeKMeans, 404
 - HideChild, 404
 - RestoreChildren, 405
- mlpack::kmeans::AllowEmptyClusters
 - AllowEmptyClusters, 1465
 - EmptyCluster, 1465
 - serialize, 1466
- mlpack::kmeans::DualTreeKMeans
 - ~DualTreeKMeans, 1468
 - DistanceCalculations, 1468
 - DualTreeKMeans, 1468
 - Iterate, 1469
 - NNSTreeType, 1467
 - Tree, 1467
- mlpack::kmeans::DualTreeKMeansRules
 - BaseCase, 1471
 - BaseCases, 1471
 - DualTreeKMeansRules, 1470
 - Rescore, 1471
 - Score, 1471, 1472
 - Scores, 1472
 - TraversalInfo, 1472
 - TraversalInfoType, 1470
- mlpack::kmeans::DualTreeKMeansStatistic
 - Centroid, 1474
 - DualTreeKMeansStatistic, 1474
 - LowerBound, 1474, 1475
 - NumTrueChildren, 1475
 - Owner, 1475
 - Pruned, 1475
 - StaticLowerBoundMovement, 1476
 - StaticPruned, 1476
 - StaticUpperBoundMovement, 1476
 - TrueChild, 1477
 - TrueParent, 1477
 - UpperBound, 1477, 1478
- mlpack::kmeans::ElkanKMeans
 - DistanceCalculations, 1479
 - ElkanKMeans, 1478
 - Iterate, 1479
- mlpack::kmeans::HamerlyKMeans
 - DistanceCalculations, 1480
 - HamerlyKMeans, 1480
 - Iterate, 1480
- mlpack::kmeans::KMeans
 - Cluster, 1485, 1486
 - EmptyClusterAction, 1487
 - KMeans, 1485
 - MaxIterations, 1487, 1488
 - Metric, 1488
 - Partitioner, 1488
 - serialize, 1489
- mlpack::kmeans::KillEmptyClusters
 - EmptyCluster, 1482
 - KillEmptyClusters, 1481
 - serialize, 1483
- mlpack::kmeans::MaxVarianceNewCluster
 - EmptyCluster, 1490
 - MaxVarianceNewCluster, 1490
 - serialize, 1491
- mlpack::kmeans::NaiveKMeans
 - DistanceCalculations, 1492
 - Iterate, 1492
 - NaiveKMeans, 1492
- mlpack::kmeans::PellegMooreKMeans
 - ~PellegMooreKMeans, 1494
 - DistanceCalculations, 1494
 - Iterate, 1495
 - PellegMooreKMeans, 1494
 - TreeType, 1494
- mlpack::kmeans::PellegMooreKMeansRules
 - BaseCase, 1496
 - DistanceCalculations, 1497
 - PellegMooreKMeansRules, 1496
 - Rescore, 1497
 - Score, 1498
- mlpack::kmeans::PellegMooreKMeansStatistic
 - Blacklist, 1499, 1500

- Centroid, 1500
- PellegMooreKMeansStatistic, 1499
- mlpack::kmeans::RandomPartition
 - Cluster, 1501
 - RandomPartition, 1501
 - serialize, 1502
- mlpack::kmeans::RefinedStart
 - Cluster, 1503, 1504
 - Percentage, 1504, 1505
 - RefinedStart, 1503
 - Samplings, 1505
 - serialize, 1505
- mlpack::kmeans::SampleInitialization
 - Cluster, 1506
 - SampleInitialization, 1506
- mlpack::kpca, 405
- mlpack::kpca::KernelPCA
 - Apply, 1508–1510
 - CenterTransformedData, 1510
 - Kernel, 1510, 1511
 - KernelPCA, 1508
- mlpack::kpca::NaiveKernelRule
 - ApplyKernelMatrix, 1511
- mlpack::kpca::NystroemKernelRule
 - ApplyKernelMatrix, 1512
- mlpack::lcc, 406
- mlpack::lcc::LocalCoordinateCoding
 - Atoms, 1516
 - Dictionary, 1516
 - Encode, 1517
 - Lambda, 1517
 - LocalCoordinateCoding, 1515
 - MaxIterations, 1517, 1518
 - Objective, 1518
 - OptimizeDictionary, 1518
 - serialize, 1518
 - Tolerance, 1519
 - Train, 1519
- mlpack::lmnn, 406
- mlpack::lmnn::Constraints
 - Constraints, 1521
 - Impostors, 1522–1524
 - K, 1524
 - KNN, 1521
 - PreCalulated, 1524, 1525
 - TargetNeighbors, 1525
 - Triplets, 1526
- mlpack::lmnn::LMNNFunction
 - Dataset, 1533
 - Evaluate, 1533
 - EvaluateWithGradient, 1534
 - GetInitialPoint, 1535
 - Gradient, 1535, 1536
 - K, 1536
 - LMNNFunction, 1532
 - NumFunctions, 1537
 - Range, 1537
 - Regularization, 1537
 - Shuffle, 1538
- mlpack::lmnn::LMNN
 - Dataset, 1528
 - K, 1528, 1529
 - LMNN, 1528
 - Labels, 1529
 - LearnDistance, 1529
 - Optimizer, 1529, 1530
 - Range, 1530
 - Regularization, 1530
- mlpack::math, 406
 - AccuLog, 410
 - Center, 410
 - ClampNonNegative, 411
 - ClampNonPositive, 411
 - ClampRange, 411
 - ClearAlias, 412
 - ColumnCovariance, 412, 413
 - FixedRandomSeed, 413
 - LogAdd, 413
 - MakeAlias, 414, 415
 - ObtainDistinctSamples, 415
 - Orthogonalize, 416
 - RandBernoulli, 416
 - randGen, 425
 - RandInt, 417
 - RandNormal, 417, 418
 - randNormalDist, 425
 - randUniformDist, 425
 - RandVector, 419
 - Random, 418
 - RandomBasis, 419
 - RandomSeed, 419
 - Range, 409
 - RemoveRows, 420
 - ShuffleData, 420–422
 - Sign, 422
 - Smat, 423
 - Svec, 423
 - SvecIndex, 424
 - SymKronId, 424
 - VectorPower, 424
 - WhitenUsingEig, 425
 - WhitenUsingSVD, 425
- mlpack::math::ColumnsToBlocks
 - BlockHeight, 1544
 - BlockWidth, 1545
 - BufSize, 1545
 - BufValue, 1546
 - Cols, 1546

- ColumnsToBlocks, 1544
- MaxRange, 1546, 1547
- MinRange, 1547
- Rows, 1547, 1548
- Scale, 1548
- Transform, 1548
- mlpack::math::RangeType
 - Contains, 1551, 1552
 - Hi, 1552
 - Lo, 1553
 - Mid, 1553
 - operator &, 1553
 - operator &=, 1554
 - operator !=, 1554
 - operator <, 1555
 - operator >, 1556
 - operator *, 1554, 1557
 - operator *=, 1555
 - operator ==, 1555
 - operator |, 1556
 - operator |=, 1556
 - RangeType, 1551
 - serialize, 1557
 - Width, 1557
- mlpack::matrix_completion, 426
- mlpack::matrix_completion::MatrixCompletion
 - MatrixCompletion, 1559
 - Recover, 1560
 - Sdp, 1560
- mlpack::meanshift, 426
- mlpack::meanshift::MeanShift
 - Cluster, 1563
 - EstimateRadius, 1563
 - Kernel, 1563, 1564
 - MaxIterations, 1564
 - MeanShift, 1562
 - Radius, 1564
- mlpack::metric, 426
 - ChebyshevDistance, 427
 - EuclideanDistance, 427
 - ManhattanDistance, 427
 - SquaredEuclideanDistance, 427
- mlpack::metric::IPMetric
 - ~IPMetric, 1567
 - Evaluate, 1567
 - IPMetric, 1566, 1567
 - Kernel, 1568
 - operator=, 1568
 - serialize, 1568
- mlpack::metric::LMetric
 - Evaluate, 1571
 - LMetric, 1570
 - Power, 1572
 - serialize, 1571
 - TakeRoot, 1572
- mlpack::metric::MahalanobisDistance
 - Covariance, 1574, 1575
 - Evaluate, 1575
 - MahalanobisDistance, 1573, 1574
 - serialize, 1576
- mlpack::naive_bayes, 428
- mlpack::naive_bayes::NaiveBayesClassifier
 - Classify, 1579, 1580
 - ElemType, 1578
 - Means, 1581
 - NaiveBayesClassifier, 1578, 1579
 - Probabilities, 1581
 - serialize, 1582
 - Train, 1582
 - Variances, 1583
- mlpack::nca, 428
- mlpack::nca::NCA
 - Dataset, 1585
 - Labels, 1585
 - LearnDistance, 1585
 - NCA, 1584
 - Optimizer, 1586
- mlpack::nca::SoftmaxErrorFunction
 - Evaluate, 1588
 - GetInitialPoint, 1588
 - Gradient, 1588, 1589
 - NumFunctions, 1589
 - Shuffle, 1590
 - SoftmaxErrorFunction, 1587
- mlpack::neighbor, 429
 - DefeatistKNN, 431
 - FurthestNeighborSort, 431
 - KFN, 432
 - KNN, 432
 - KRAFN, 432
 - KRANN, 432
 - NSType, 433
 - NearestNeighborSort, 433
 - NeighborSearchMode, 434
 - RAType, 433
 - SpillKNN, 433
 - Unmap, 434, 435
- mlpack::neighbor::AlphaVisitor
 - operator(), 1591
- mlpack::neighbor::BiSearchVisitor
 - BiSearchVisitor, 1593
 - NSTypeT, 1592
 - operator(), 1594, 1595
 - RATypeT, 1593
- mlpack::neighbor::DeleteVisitor
 - operator(), 1596
- mlpack::neighbor::DrusillaSelect
 - CandidateIndices, 1598

- CandidateSet, 1598, 1599
- DrusillaSelect, 1597, 1598
- Search, 1599
- serialize, 1599
- Train, 1600
- mlpack::neighbor::EpsilonVisitor
 - operator(), 1601
- mlpack::neighbor::FirstLeafExactVisitor
 - operator(), 1602
- mlpack::neighbor::FurthestNS
 - BestDistance, 1603
 - BestNodeToNodeDistance, 1604
 - BestPointToNodeDistance, 1605
 - CombineBest, 1605
 - CombineWorst, 1605
 - ConvertToDistance, 1606
 - ConvertToScore, 1606
 - GetBestChild, 1606, 1607
 - IsBetter, 1607
 - Relax, 1608
 - WorstDistance, 1608
- mlpack::neighbor::LSHSearch
 - BucketSize, 1612
 - ComputeRecall, 1613
 - DistanceEvaluations, 1613
 - LSHSearch, 1610–1612
 - NumProjections, 1613
 - Offsets, 1614
 - operator=, 1614
 - Projections, 1614, 1615
 - ReferenceSet, 1615
 - Search, 1615, 1616
 - SecondHashTable, 1616
 - SecondHashWeights, 1616
 - serialize, 1617
 - Train, 1617
- mlpack::neighbor::MonoSearchVisitor
 - MonoSearchVisitor, 1619
 - operator(), 1619, 1620
- mlpack::neighbor::NSModel
 - ~NSModel, 1661
 - BuildModel, 1661
 - Dataset, 1661
 - Epsilon, 1661
 - LeafSize, 1662
 - NSModel, 1660
 - operator=, 1662
 - RandomBasis, 1663
 - Rho, 1663
 - Search, 1663, 1664
 - SearchMode, 1664
 - serialize, 1664
 - Tau, 1664, 1665
 - TreeName, 1665
 - TreeType, 1665
 - TreeTypes, 1659
- mlpack::neighbor::NaiveVisitor
 - operator(), 1621
- mlpack::neighbor::NearestNS
 - BestDistance, 1622
 - BestNodeToNodeDistance, 1623
 - BestPointToNodeDistance, 1624
 - CombineBest, 1624
 - CombineWorst, 1624
 - ConvertToDistance, 1625
 - ConvertToScore, 1625
 - GetBestChild, 1625
 - IsBetter, 1626
 - Relax, 1626
 - WorstDistance, 1627
- mlpack::neighbor::NeighborSearch
 - ~NeighborSearch, 1633
 - BaseCases, 1633
 - EffectiveError, 1633
 - Epsilon, 1634
 - NeighborSearch, 1630–1632
 - operator=, 1634
 - Recall, 1635
 - ReferenceSet, 1635
 - ReferenceTree, 1635, 1636
 - Scores, 1636
 - Search, 1636, 1637
 - SearchMode, 1638
 - serialize, 1638
 - Train, 1638, 1640
 - Tree, 1630
- mlpack::neighbor::NeighborSearchRules
 - BaseCase, 1644
 - BaseCases, 1644
 - baseCases, 1649
 - CalculateBound, 1645
 - Candidate, 1643
 - CandidateList, 1643
 - candidates, 1649
 - epsilon, 1649
 - GetBestChild, 1645
 - GetResults, 1646
 - InsertNeighbor, 1646
 - k, 1650
 - lastBaseCase, 1650
 - lastQueryIndex, 1650
 - lastReferenceIndex, 1650
 - metric, 1650
 - NeighborSearchRules, 1643
 - querySet, 1651
 - referenceSet, 1651
 - Rescore, 1646, 1647
 - sameSet, 1651

- Score, 1647, 1648
- Scores, 1648
- scores, 1651
- TraversalInfo, 1648, 1649
- traversalInfo, 1651
- TraversalInfoType, 1643
- mlpack::neighbor::NeighborSearchRules::CandidateCmp
operator(), 1652
- mlpack::neighbor::NeighborSearchStat
 - AuxBound, 1655
 - FirstBound, 1655
 - LastDistance, 1655, 1656
 - NeighborSearchStat, 1654
 - Reset, 1656
 - SecondBound, 1656
 - serialize, 1656
- mlpack::neighbor::QDAFN
 - CandidateSet, 1667
 - NumProjections, 1667
 - QDAFN, 1666, 1667
 - Search, 1668
 - serialize, 1668
 - Train, 1668
- mlpack::neighbor::RAModel
 - ~RAModel, 1672
 - Alpha, 1673
 - BuildModel, 1673
 - Dataset, 1673
 - FirstLeafExact, 1673
 - LeafSize, 1674
 - Naive, 1674
 - operator=, 1674, 1675
 - RAModel, 1672
 - RandomBasis, 1675
 - SampleAtLeaves, 1675
 - Search, 1676
 - serialize, 1676
 - SingleMode, 1676
 - SingleSampleLimit, 1677
 - Tau, 1677
 - TreeName, 1677
 - TreeType, 1677, 1678
 - TreeTypes, 1671
- mlpack::neighbor::RAQueryStat
 - Bound, 1679, 1680
 - NumSamplesMade, 1680
 - RAQueryStat, 1679
 - serialize, 1680
- mlpack::neighbor::RASearch
 - ~RASearch, 1685
 - Alpha, 1686
 - FirstLeafExact, 1686
 - Naive, 1686, 1687
 - RASearch, 1683–1685
 - ReferenceSet, 1687
 - ResetQueryTree, 1687
 - SampleAtLeaves, 1688
 - Search, 1688, 1689
 - serialize, 1690
 - SingleMode, 1690
 - SingleSampleLimit, 1690
 - Tau, 1691
 - Train, 1691, 1692
 - Tree, 1683
- mlpack::neighbor::RASearchRules
 - BaseCase, 1694
 - GetResults, 1694
 - NumDistComputations, 1695
 - NumEffectiveSamples, 1695
 - RASearchRules, 1693
 - Rescore, 1695, 1696
 - Score, 1696–1698
 - TraversalInfo, 1698
 - TraversalInfoType, 1693
- mlpack::neighbor::RAUtil
 - MinimumSamplesReqd, 1699
 - ObtainDistinctSamples, 1700
 - SuccessProbability, 1700
- mlpack::neighbor::ReferenceSetVisitor
 - operator(), 1701, 1702
- mlpack::neighbor::SampleAtLeavesVisitor
 - operator(), 1703
- mlpack::neighbor::SearchModeVisitor
 - operator(), 1704
- mlpack::neighbor::SingleModeVisitor
 - operator(), 1705
- mlpack::neighbor::SingleSampleLimitVisitor
 - operator(), 1706
- mlpack::neighbor::TauVisitor
 - operator(), 1707
- mlpack::neighbor::TrainVisitor
 - NSTypeT, 1708
 - operator(), 1709–1711
 - RATypeT, 1709
 - TrainVisitor, 1709
- mlpack::nn, 435
 - MaximalInputs, 436
 - NormalizeColByMax, 437
- mlpack::nn::SparseAutoencoder
 - Beta, 1714
 - GetNewFeatures, 1714
 - HiddenSize, 1715
 - Lambda, 1715
 - Rho, 1715, 1716
 - Sigmoid, 1716
 - SparseAutoencoder, 1713
 - VisibleSize, 1716
- mlpack::nn::SparseAutoencoderFunction

- Beta, 1718
- Evaluate, 1719
- GetInitialPoint, 1719
- Gradient, 1719
- HiddenSize, 1720
- InitializeWeights, 1720
- Lambda, 1720
- Rho, 1721
- Sigmoid, 1721
- SparseAutoencoderFunction, 1718
- VisibleSize, 1721, 1722
- mlpack::pca, 437
- mlpack::pca::ExactSVDPolicy
 - Apply, 1722
- mlpack::pca::PCA
 - Apply, 1724–1726
 - PCA, 1724
 - ScaleData, 1726, 1727
- mlpack::pca::QUICSVDPolicy
 - Apply, 1728
 - Delta, 1729
 - Epsilon, 1729
 - QUICSVDPolicy, 1728
- mlpack::pca::RandomizedBlockKrylovSVDPolicy
 - Apply, 1731
 - BlockSize, 1731
 - MaxIterations, 1732
 - RandomizedBlockKrylovSVDPolicy, 1730
- mlpack::pca::RandomizedSVDPolicy
 - Apply, 1733
 - IteratedPower, 1734
 - MaxIterations, 1734
 - RandomizedSVDPolicy, 1733
- mlpack::perceptron, 438
- mlpack::perceptron::Perceptron
 - Biases, 1737, 1738
 - Classify, 1738
 - MaxIterations, 1738
 - NumClasses, 1739
 - Perceptron, 1736, 1737
 - serialize, 1739
 - Train, 1739
 - Weights, 1740
- mlpack::perceptron::RandomInitialization
 - Initialize, 1741
 - RandomInitialization, 1741
- mlpack::perceptron::SimpleWeightUpdate
 - UpdateWeights, 1742
- mlpack::perceptron::ZeroInitialization
 - Initialize, 1743
 - ZeroInitialization, 1743
- mlpack::radical, 438
 - WhitenFeatureMajorMatrix, 438
- mlpack::radical::Radical
 - Angles, 1745
 - CopyAndPerturb, 1745
 - DoRadical, 1746
 - DoRadical2D, 1746
 - NoiseStdDev, 1746
 - Radical, 1745
 - Replicates, 1747
 - Sweeps, 1747
 - Vasicek, 1747
- mlpack::range, 439
 - RSType, 439
- mlpack::range::BiSearchVisitor
 - BiSearchVisitor, 1749
 - operator(), 1749, 1750
 - RSTypeT, 1749
- mlpack::range::DeleteVisitor
 - operator(), 1751
- mlpack::range::MonoSearchVisitor
 - MonoSearchVisitor, 1752
 - operator(), 1753
- mlpack::range::NaiveVisitor
 - operator(), 1754
- mlpack::range::RSModel
 - ~RSModel, 1775
 - BuildModel, 1775
 - Dataset, 1775
 - LeafSize, 1775, 1776
 - Naive, 1776
 - operator=, 1776
 - RSModel, 1774
 - RandomBasis, 1777
 - Search, 1777, 1778
 - serialize, 1778
 - SingleMode, 1778
 - TreeType, 1778, 1779
 - TreeTypes, 1773
- mlpack::range::RangeSearch
 - ~RangeSearch, 1758
 - BaseCases, 1758
 - Naive, 1759
 - operator=, 1759
 - RangeSearch, 1756–1758
 - ReferenceSet, 1759
 - ReferenceTree, 1760
 - Scores, 1760
 - Search, 1760–1762
 - serialize, 1762
 - SingleMode, 1762, 1763
 - Train, 1763
 - Tree, 1756
- mlpack::range::RangeSearchRules
 - BaseCase, 1766
 - BaseCases, 1766
 - RangeSearchRules, 1765

- Rescore, 1766, 1767
- Score, 1767
- Scores, 1768
- TraversalInfo, 1768
- TraversalInfoType, 1765
- mlpack::range::RangeSearchStat
 - LastDistance, 1770
 - RangeSearchStat, 1769
 - serialize, 1770
- mlpack::range::ReferenceSetVisitor
 - operator(), 1771
- mlpack::range::SingleModeVisitor
 - operator(), 1780
- mlpack::range::TrainVisitor
 - operator(), 1782
 - RSTypeT, 1781
 - TrainVisitor, 1781
- mlpack::regression, 440
- mlpack::regression::LARS
 - ActiveSet, 1786
 - Beta, 1787
 - BetaPath, 1787
 - LARS, 1784–1786
 - LambdaPath, 1787
 - MatUtriCholFactor, 1787
 - Predict, 1787
 - serialize, 1788
 - Train, 1788
- mlpack::regression::LinearRegression
 - ComputeError, 1791
 - Intercept, 1792
 - Lambda, 1792
 - LinearRegression, 1790, 1791
 - Parameters, 1793
 - Predict, 1793
 - serialize, 1794
 - Train, 1794
- mlpack::regression::LogisticRegression
 - Classify, 1798, 1799
 - ComputeAccuracy, 1800
 - ComputeError, 1800
 - Lambda, 1801
 - LogisticRegression, 1797, 1798
 - Parameters, 1801
 - serialize, 1802
 - Train, 1802, 1803
- mlpack::regression::LogisticRegressionFunction
 - Evaluate, 1806
 - EvaluateWithGradient, 1807
 - GetInitialPoint, 1807
 - Gradient, 1807, 1808
 - InitialPoint, 1808
 - Lambda, 1808, 1809
 - LogisticRegressionFunction, 1805
 - NumFeatures, 1809
 - NumFunctions, 1809
 - PartialGradient, 1809
 - Predictors, 1810
 - Responses, 1810
 - Shuffle, 1810
- mlpack::regression::SoftmaxRegression
 - Classify, 1813–1815
 - ComputeAccuracy, 1815
 - FeatureSize, 1815
 - FitIntercept, 1816
 - Lambda, 1816
 - NumClasses, 1816
 - Parameters, 1817
 - serialize, 1817
 - SoftmaxRegression, 1812, 1813
 - Train, 1817
- mlpack::regression::SoftmaxRegressionFunction
 - Evaluate, 1820
 - FitIntercept, 1821
 - GetGroundTruthMatrix, 1821
 - GetInitialPoint, 1821
 - GetProbabilitiesMatrix, 1821
 - Gradient, 1822
 - InitializeWeights, 1823
 - Lambda, 1824
 - NumClasses, 1824
 - NumFeatures, 1824
 - PartialGradient, 1824
 - Shuffle, 1825
 - SoftmaxRegressionFunction, 1819
- mlpack::rl, 440
 - Acrobat, 441
 - NStepQLearning, 442
 - OneStepQLearning, 442
 - OneStepSarsa, 442
- mlpack::rl::Acrobot
 - Acrobot, 1827
 - Action, 1826
 - Dsdt, 1827
 - InitialSample, 1828
 - IsTerminal, 1828
 - Rk4, 1829
 - Sample, 1829, 1830
 - Torque, 1830
 - Wrap, 1831
- mlpack::rl::Acrobot::State
 - AngularVelocity1, 1833
 - AngularVelocity2, 1833
 - Data, 1834
 - dimension, 1835
 - Encode, 1834
 - State, 1832
 - Theta1, 1834

- Theta2, 1834, 1835
- mlpack::rl::AggregatedPolicy
 - ActionType, 1836
 - AggregatedPolicy, 1836
 - Anneal, 1836
 - Sample, 1837
- mlpack::rl::AsyncLearning
 - AsyncLearning, 1839
 - Config, 1839
 - Environment, 1840
 - Network, 1840
 - Policy, 1840, 1841
 - Train, 1841
 - Updater, 1841, 1842
- mlpack::rl::CartPole
 - Action, 1843
 - CartPole, 1843
 - InitialSample, 1844
 - IsTerminal, 1844
 - Sample, 1845
- mlpack::rl::CartPole::State
 - Angle, 1848
 - AngularVelocity, 1848
 - Data, 1848
 - dimension, 1850
 - Encode, 1849
 - Position, 1849
 - State, 1847
 - Velocity, 1849, 1850
- mlpack::rl::ContinuousMountainCar
 - ContinuousMountainCar, 1851
 - InitialSample, 1852
 - IsTerminal, 1852
 - Sample, 1853
- mlpack::rl::ContinuousMountainCar::Action
 - action, 1854
 - size, 1855
- mlpack::rl::ContinuousMountainCar::State
 - Data, 1856
 - dimension, 1858
 - Encode, 1857
 - Position, 1857
 - State, 1856
 - Velocity, 1857, 1858
- mlpack::rl::GreedyPolicy
 - ActionType, 1859
 - Anneal, 1860
 - Epsilon, 1860
 - GreedyPolicy, 1859
 - Sample, 1860
- mlpack::rl::MountainCar
 - Action, 1862
 - InitialSample, 1863
 - IsTerminal, 1863
 - MountainCar, 1862
 - Sample, 1864
- mlpack::rl::MountainCar::State
 - Data, 1866
 - dimension, 1868
 - Encode, 1867
 - Position, 1867
 - State, 1866
 - Velocity, 1867
- mlpack::rl::NStepQLearningWorker
 - ActionType, 1869
 - Initialize, 1870
 - NStepQLearningWorker, 1870
 - StateType, 1869
 - Step, 1870
 - TransitionType, 1869
- mlpack::rl::OneStepQLearningWorker
 - ActionType, 1872
 - Initialize, 1873
 - OneStepQLearningWorker, 1873
 - StateType, 1872
 - Step, 1874
 - TransitionType, 1873
- mlpack::rl::OneStepSarsaWorker
 - ActionType, 1875
 - Initialize, 1877
 - OneStepSarsaWorker, 1876
 - StateType, 1876
 - Step, 1877
 - TransitionType, 1876
- mlpack::rl::Pendulum
 - AngleNormalize, 1879
 - InitialSample, 1879
 - Pendulum, 1878
 - Sample, 1879, 1880
- mlpack::rl::Pendulum::Action
 - action, 1881
 - size, 1881
- mlpack::rl::Pendulum::State
 - AngularVelocity, 1883
 - Data, 1884
 - dimension, 1885
 - Encode, 1884
 - State, 1883
 - Theta, 1884
- mlpack::rl::QLearning
 - ActionType, 1886
 - Deterministic, 1888
 - Environment, 1888
 - Episode, 1888
 - Network, 1889
 - QLearning, 1887
 - State, 1889
 - StateType, 1887

- Step, 1890
- TotalSteps, 1890
- mlpack::rl::RandomReplay
 - ActionType, 1891
 - RandomReplay, 1892
 - Sample, 1892
 - Size, 1893
 - StateType, 1892
 - Store, 1893
- mlpack::rl::RewardClipping
 - Action, 1896
 - Environment, 1897
 - InitialSample, 1897
 - IsTerminal, 1897
 - MaxReward, 1898
 - MinReward, 1898
 - RewardClipping, 1896
 - Sample, 1899, 1900
 - State, 1896
- mlpack::rl::TrainingConfig
 - Discount, 1902
 - DoubleQLearning, 1903
 - ExplorationSteps, 1903
 - GradientLimit, 1903, 1904
 - NumWorkers, 1904
 - StepLimit, 1904, 1905
 - StepSize, 1905
 - TargetNetworkSyncInterval, 1905, 1906
 - TrainingConfig, 1902
 - UpdateInterval, 1906
- mlpack::sfinae, 443
- mlpack::sfinae::MethodFormDetector< Class, Method↵
 - Form, 0 >
 - operator(), 1907
- mlpack::sfinae::MethodFormDetector< Class, Method↵
 - Form, 1 >
 - operator(), 1908
- mlpack::sfinae::MethodFormDetector< Class, Method↵
 - Form, 2 >
 - operator(), 1909
- mlpack::sfinae::MethodFormDetector< Class, Method↵
 - Form, 3 >
 - operator(), 1909
- mlpack::sfinae::MethodFormDetector< Class, Method↵
 - Form, 4 >
 - operator(), 1910
- mlpack::sfinae::MethodFormDetector< Class, Method↵
 - Form, 5 >
 - operator(), 1911
- mlpack::sfinae::MethodFormDetector< Class, Method↵
 - Form, 6 >
 - operator(), 1911
- mlpack::sfinae::MethodFormDetector< Class, Method↵
 - Form, 7 >
- operator(), 1912
- mlpack::sparse_coding, 443
- mlpack::sparse_coding::DataDependentRandomInitializer
 - Initialize, 1913
- mlpack::sparse_coding::NothingInitializer
 - Initialize, 1914
- mlpack::sparse_coding::RandomInitializer
 - Initialize, 1915
- mlpack::sparse_coding::SparseCoding
 - Atoms, 1920
 - Dictionary, 1920
 - Encode, 1921
 - Lambda1, 1921
 - Lambda2, 1921, 1922
 - MaxIterations, 1922
 - NewtonTolerance, 1922
 - ObjTolerance, 1923
 - Objective, 1923
 - OptimizeDictionary, 1923
 - ProjectDictionary, 1924
 - serialize, 1924
 - SparseCoding, 1918, 1919
 - Train, 1924
- mlpack::svd, 444
- mlpack::svd::BiasSVDFunction
 - BiasSVDFunction, 1928
 - Dataset, 1928
 - Evaluate, 1928, 1929
 - GetInitialPoint, 1929
 - Gradient, 1929, 1930
 - Lambda, 1930
 - NumFunctions, 1930
 - NumItems, 1931
 - NumUsers, 1931
 - Rank, 1931
 - Shuffle, 1931
- mlpack::svd::BiasSVD
 - Apply, 1926
 - BiasSVD, 1925
- mlpack::svd::QUIC_SVD
 - ExtractSVD, 1933
 - QUIC_SVD, 1932
- mlpack::svd::RandomizedBlockKrylovSVD
 - Apply, 1936
 - BlockSize, 1936
 - MaxIterations, 1937
 - RandomizedBlockKrylovSVD, 1935
- mlpack::svd::RandomizedSVD
 - Apply, 1940, 1941
 - Epsilon, 1941, 1942
 - IteratedPower, 1942
 - MaxIterations, 1942
 - RandomizedSVD, 1939
- mlpack::svd::RegularizedSVDFunction

- Dataset, 1946
- Evaluate, 1946, 1948
- GetInitialPoint, 1948
- Gradient, 1948, 1949
- Lambda, 1949
- NumFunctions, 1949
- NumItems, 1950
- NumUsers, 1950
- Rank, 1950
- RegularizedSVDFunction, 1946
- Shuffle, 1950
- mlpack::svd::RegularizedSVD
 - Apply, 1944
 - RegularizedSVD, 1944
- mlpack::svd::SVDPlusPlus
 - Apply, 1952, 1953
 - CleanData, 1953
 - SVDPlusPlus, 1952
- mlpack::svd::SVDPlusPlusFunction
 - Dataset, 1955
 - Evaluate, 1956
 - GetInitialPoint, 1956
 - Gradient, 1957
 - ImplicitDataset, 1957
 - Lambda, 1958
 - NumFunctions, 1958
 - NumItems, 1958
 - NumUsers, 1958
 - Rank, 1958
 - SVDPlusPlusFunction, 1955
 - Shuffle, 1959
- mlpack::svm, 444
- mlpack::svm::LinearSVMFunction
 - Dataset, 1969
 - Evaluate, 1969, 1970
 - EvaluateWithGradient, 1970, 1971
 - FitIntercept, 1971
 - GetGroundTruthMatrix, 1971
 - Gradient, 1972
 - InitialPoint, 1973
 - InitializeWeights, 1973
 - Lambda, 1974
 - LinearSVMFunction, 1968
 - NumFunctions, 1974
 - Shuffle, 1974
- mlpack::svm::LinearSVM
 - Classify, 1962, 1963
 - ComputeAccuracy, 1964
 - FeatureSize, 1964
 - Lambda, 1964, 1965
 - LinearSVM, 1961, 1962
 - NumClasses, 1965
 - Parameters, 1965
 - serialize, 1966
 - Train, 1966
- mlpack::tree, 444
 - AxisOrthogonalHyperplane, 450
 - BallTree, 451
 - BinaryDoubleNumericSplit, 451
 - Bootstrap, 463
 - CosineNodeQueue, 451
 - DecisionStump, 452
 - DiscreteHilbertRTreeAuxiliaryInformation, 452
 - EnumerateTree, 463
 - HilbertRTree, 452
 - HoeffdingDoubleNumericSplit, 453
 - HoeffdingTreeType, 453
 - Hyperplane, 453
 - KDTree, 453
 - MAX_OVERLAP, 464
 - MaxRPTree, 454
 - MeanSPTree, 456
 - MeanSplitBallTree, 455
 - MeanSplitKDTree, 455
 - NonOrtMeanSPTree, 456
 - NonOrtSPTree, 457
 - RPTree, 458
 - RPlusPlusTree, 457
 - RPlusTree, 458
 - RStarTree, 459
 - RTree, 459
 - SPTree, 460
 - StandardCoverTree, 460
 - UBTree, 461
 - VPTree, 461
 - VPTreeSplit, 462
 - XTree, 462
- mlpack::tree::AllCategoricalSplit
 - CalculateDirection, 1982
 - NumChildren, 1983
 - SplitIfBetter, 1983
- mlpack::tree::AllDimensionSelect
 - AllDimensionSelect, 1985
 - Begin, 1985
 - Dimensions, 1985
 - End, 1986
 - Next, 1986
- mlpack::tree::AxisParallelProjVector
 - AxisParallelProjVector, 1987
 - Project, 1988
 - serialize, 1989
- mlpack::tree::BestBinaryNumericSplit
 - CalculateDirection, 1990
 - NumChildren, 1990
 - SplitIfBetter, 1991
- mlpack::tree::BinaryNumericSplit
 - BinaryNumericSplit, 1993, 1994
 - EvaluateFitnessFunction, 1994

- MajorityClass, 1994
- MajorityProbability, 1995
- NumChildren, 1995
- serialize, 1995
- Split, 1995
- SplitInfo, 1993
- Train, 1996
- mlpack::tree::BinaryNumericSplitInfo
 - BinaryNumericSplitInfo, 1997
 - CalculateDirection, 1997
 - serialize, 1997
- mlpack::tree::BinarySpaceTree
 - ~BinarySpaceTree, 2007
 - Begin, 2008
 - BinarySpaceTree, 2002–2008
 - Bound, 2008, 2009
 - Center, 2009
 - Child, 2009
 - ChildPtr, 2010
 - Count, 2010
 - Dataset, 2010, 2011
 - Descendant, 2011
 - ElemType, 2002
 - FurthestDescendantDistance, 2011
 - FurthestPointDistance, 2011
 - GetFurthestChild, 2012
 - GetNearestChild, 2012
 - IsLeaf, 2013
 - Left, 2013
 - Mat, 2002
 - MaxDistance, 2013
 - Metric, 2014
 - MinDistance, 2014
 - MinimumBoundDistance, 2014
 - NumChildren, 2015
 - NumDescendants, 2015
 - NumPoints, 2015
 - Parent, 2015, 2016
 - ParentDistance, 2016
 - Point, 2016
 - RangeDistance, 2017
 - Right, 2017
 - serialize, 2018
 - Split, 2002
 - Stat, 2018
- mlpack::tree::BinarySpaceTree::BreadthFirstDualTreeTraverser
 - BreadthFirstDualTreeTraverser, 2020
 - NumBaseCases, 2020
 - NumPrunes, 2020, 2021
 - NumScores, 2021
 - NumVisited, 2021
 - QueueFrameType, 2019
 - Traverse, 2022
- mlpack::tree::BinarySpaceTree::DualTreeTraverser
 - DualTreeTraverser, 2023
 - NumBaseCases, 2024
 - NumPrunes, 2024
 - NumScores, 2024, 2025
 - NumVisited, 2025
 - Traverse, 2025
- mlpack::tree::BinarySpaceTree::SingleTreeTraverser
 - NumPrunes, 2027
 - SingleTreeTraverser, 2026
 - Traverse, 2027
- mlpack::tree::CategoricalSplitInfo
 - CalculateDirection, 2028
 - CategoricalSplitInfo, 2028
 - serialize, 2029
- mlpack::tree::CompareCosineNode
 - operator(), 2029
- mlpack::tree::CosineTree
 - ~CosineTree, 2033
 - BasisVector, 2033
 - BinarySearch, 2033
 - CalculateCentroid, 2034
 - CalculateCosines, 2034
 - Centroid, 2034
 - ColumnSampleLS, 2034
 - ColumnSamplesLS, 2035
 - ConstructBasis, 2035
 - CosineNodeSplit, 2035
 - CosineTree, 2031, 2032
 - FrobNormSquared, 2035
 - GetDataset, 2036
 - GetFinalBasis, 2036
 - L2Error, 2036
 - Left, 2037
 - ModifiedGramSchmidt, 2037
 - MonteCarloError, 2038
 - NumColumns, 2038
 - Parent, 2038
 - Right, 2039
 - SplitPointIndex, 2039
 - VectorIndices, 2039
- mlpack::tree::CoverTree
 - ~CoverTree, 2049
 - Base, 2049, 2050
 - BreadthFirstDualTreeTraverser, 2044
 - Center, 2050
 - Child, 2050
 - ChildPtr, 2051
 - Children, 2051
 - CoverTree, 2045–2049
 - Dataset, 2051
 - Descendant, 2052
 - DistanceComps, 2052
 - ElemType, 2044

- FurthestDescendantDistance, 2052
- FurthestPointDistance, 2053
- GetFurthestChild, 2053
- GetNearestChild, 2053, 2054
- IsLeaf, 2054
- Mat, 2045
- MaxDistance, 2054, 2055
- Metric, 2055
- MinDistance, 2055, 2056
- MinimumBoundDistance, 2056
- NumChildren, 2056
- NumDescendants, 2057
- NumPoints, 2057
- Parent, 2057
- ParentDistance, 2057, 2058
- Point, 2058
- RangeDistance, 2058, 2059
- Scale, 2059
- serialize, 2059
- Stat, 2060
- mlpack::tree::CoverTree::DualTreeTraverser
 - DualTreeTraverser, 2061
 - NumBaseCases, 2061
 - NumPrunes, 2062
 - NumScores, 2062
 - NumVisited, 2062
 - Traverse, 2062
- mlpack::tree::CoverTree::SingleTreeTraverser
 - NumPrunes, 2064
 - SingleTreeTraverser, 2063
 - Traverse, 2064
- mlpack::tree::DecisionTree
 - ~DecisionTree, 2072
 - CalculateDirection, 2072
 - CategoricalSplit, 2067
 - Child, 2072, 2073
 - Classify, 2073, 2074
 - DecisionTree, 2068–2071
 - DimensionSelection, 2068
 - NumChildren, 2074
 - NumClasses, 2075
 - NumericSplit, 2068
 - operator=, 2075
 - serialize, 2076
 - SplitDimension, 2076
 - Train, 2076–2078
- mlpack::tree::EmptyStatistic
 - ~EmptyStatistic, 2080
 - EmptyStatistic, 2080, 2081
 - serialize, 2081
- mlpack::tree::ExampleTree
 - Centroid, 2084
 - Child, 2084
 - Descendant, 2084
 - ExampleTree, 2083
 - FurthestDescendantDistance, 2085
 - MaxDistance, 2085
 - Metric, 2086
 - MinDistance, 2086
 - NumChildren, 2087
 - NumDescendants, 2087
 - NumPoints, 2087
 - Parent, 2087
 - ParentDistance, 2087
 - Point, 2088
 - RangeDistance, 2088
 - Stat, 2089
- mlpack::tree::FirstPointIsRoot
 - ChooseRoot, 2090
- mlpack::tree::GiniGain
 - Evaluate, 2091
 - EvaluatePtr, 2091
 - Range, 2091
- mlpack::tree::GiniImpurity
 - Evaluate, 2092
 - Range, 2092
- mlpack::tree::GreedySingleTreeTraverser
 - GreedySingleTreeTraverser, 2093
 - MinBaseCases, 2094
 - NumPrunes, 2094
 - Traverse, 2094
- mlpack::tree::HilbertRTreeAuxiliaryInformation
 - Children, 2097
 - ElemType, 2096
 - HandleNodeInsertion, 2098
 - HandleNodeRemoval, 2098
 - HandlePointDeletion, 2098
 - HandlePointInsertion, 2099
 - HilbertRTreeAuxiliaryInformation, 2096, 2097
 - HilbertValue, 2099
 - NullifyData, 2099
 - operator=, 2100
 - serialize, 2100
 - UpdateAuxiliaryInfo, 2100
- mlpack::tree::HilbertRTreeDescentHeuristic
 - ChooseDescentNode, 2101, 2102
- mlpack::tree::HilbertRTreeSplit
 - SplitLeafNode, 2103
 - SplitNonLeafNode, 2103
- mlpack::tree::HoeffdingCategoricalSplit
 - EvaluateFitnessFunction, 2106
 - HoeffdingCategoricalSplit, 2105, 2106
 - MajorityClass, 2106
 - MajorityProbability, 2107
 - NumChildren, 2107
 - serialize, 2107
 - Split, 2107
 - SplitInfo, 2105

- Train, 2108
- mlpack::tree::HoeffdingNumericSplit
 - Bins, 2111
 - EvaluateFitnessFunction, 2111
 - HoeffdingNumericSplit, 2110
 - MajorityClass, 2111
 - MajorityProbability, 2112
 - NumChildren, 2112
 - serialize, 2112
 - Split, 2112
 - SplitInfo, 2110
 - Train, 2113
- mlpack::tree::HoeffdingTree
 - ~HoeffdingTree, 2118
 - CalculateDirection, 2119
 - CategoricalSplit, 2116
 - CheckInterval, 2119
 - Child, 2120
 - Classify, 2120, 2121
 - CreateChildren, 2122
 - HoeffdingTree, 2116–2118
 - MajorityClass, 2122
 - MajorityProbability, 2122, 2123
 - MaxSamples, 2123
 - MinSamples, 2123
 - NumChildren, 2124
 - NumDescendants, 2124
 - NumericSplit, 2116
 - serialize, 2124
 - SplitCheck, 2124
 - SplitDimension, 2124
 - SuccessProbability, 2125
 - Train, 2125, 2126
- mlpack::tree::HoeffdingTreeModel
 - ~HoeffdingTreeModel, 2130
 - BuildModel, 2130
 - Classify, 2131
 - GiniBinaryTreeType, 2128
 - GiniHoeffdingTreeType, 2128
 - HoeffdingTreeModel, 2129
 - InfoBinaryTreeType, 2128
 - InfoHoeffdingTreeType, 2128
 - NumNodes, 2132
 - operator=, 2132
 - serialize, 2132
 - Train, 2133
 - TreeType, 2128
- mlpack::tree::HyperplaneBase
 - BoundType, 2134
 - HyperplaneBase, 2135
 - Left, 2136
 - ProjVectorType, 2135
 - Project, 2136
 - Right, 2137
 - serialize, 2137
- mlpack::tree::InformationGain
 - Evaluate, 2138, 2139
 - EvaluatePtr, 2139
 - Range, 2139
- mlpack::tree::IsSpillTree
 - value, 2140
- mlpack::tree::IsSpillTree< tree::SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > >
 - value, 2141
- mlpack::tree::MeanSpaceSplit
 - SplitSpace, 2142
- mlpack::tree::MeanSplit
 - AssignToLeftNode, 2143
 - PerformSplit, 2144
 - SplitNode, 2145
- mlpack::tree::MeanSplit::SplitInfo
 - splitDimension, 2146
 - splitVal, 2146
- mlpack::tree::MidpointSpaceSplit
 - SplitSpace, 2147
- mlpack::tree::MidpointSplit
 - AssignToLeftNode, 2148
 - PerformSplit, 2149
 - SplitNode, 2150
- mlpack::tree::MidpointSplit::SplitInfo
 - splitDimension, 2151
 - splitVal, 2151
- mlpack::tree::MinimalCoverageSweep
 - CheckLeafSweep, 2152
 - CheckNonLeafSweep, 2153
 - SweepLeafNode, 2153
 - SweepNonLeafNode, 2154
- mlpack::tree::MinimalCoverageSweep::SweepCost
 - type, 2155
- mlpack::tree::MinimalSplitsNumberSweep
 - SweepLeafNode, 2156
 - SweepNonLeafNode, 2156
- mlpack::tree::MinimalSplitsNumberSweep::SweepCost
 - type, 2157
- mlpack::tree::MultipleRandomDimensionSelect
 - Begin, 2159
 - Dimensions, 2159
 - End, 2159
 - MultipleRandomDimensionSelect, 2158
 - Next, 2159
- mlpack::tree::NoAuxiliaryInformation
 - HandleNodeInsertion, 2162
 - HandleNodeRemoval, 2162
 - HandlePointDeletion, 2163
 - HandlePointInsertion, 2163
 - NoAuxiliaryInformation, 2161
 - NullifyData, 2164

- operator=, 2164
- serialize, 2164
- SplitAuxiliaryInfo, 2164
- UpdateAuxiliaryInfo, 2165
- mlpack::tree::NumericSplitInfo
 - CalculateDirection, 2166
 - NumericSplitInfo, 2166
 - serialize, 2166
- mlpack::tree::Octree
 - ~Octree, 2175
 - Bound, 2176
 - Center, 2176
 - Child, 2176
 - ChildPtr, 2177
 - Dataset, 2177
 - Descendant, 2177
 - ElemType, 2170
 - FurthestDescendantDistance, 2177
 - FurthestPointDistance, 2178
 - GetFurthestChild, 2178
 - GetNearestChild, 2178, 2179
 - IsLeaf, 2179
 - Mat, 2170
 - MaxDistance, 2179
 - Metric, 2180
 - MinDistance, 2180
 - MinimumBoundDistance, 2180
 - NumChildren, 2181
 - NumDescendants, 2181
 - NumPoints, 2181
 - Octree, 2170–2175
 - Parent, 2181
 - ParentDistance, 2182
 - Point, 2182
 - RangeDistance, 2182, 2183
 - serialize, 2183
 - Stat, 2183
- mlpack::tree::Octree::DualTreeTraverser
 - DualTreeTraverser, 2185
 - NumBaseCases, 2185
 - NumPrunes, 2185, 2186
 - NumScores, 2186
 - NumVisited, 2186
 - NumVistied, 2186
 - Traverse, 2187
- mlpack::tree::Octree::SingleTreeTraverser
 - NumPrunes, 2188
 - SingleTreeTraverser, 2188
 - Traverse, 2188
- mlpack::tree::Octree::SplitType::SplitInfo
 - center, 2190
 - d, 2190
 - SplitInfo, 2189
- mlpack::tree::ProjVector
 - ProjVector, 2191
 - Project, 2191, 2192
 - serialize, 2192
- mlpack::tree::QueueFrame
 - queryDepth, 2193
 - queryNode, 2193
 - referenceNode, 2193
 - score, 2193
 - traversalInfo, 2194
- mlpack::tree::RPTreeMaxSplit
 - AssignToLeftNode, 2257
 - ElemType, 2256
 - PerformSplit, 2257, 2258
 - SplitNode, 2258
- mlpack::tree::RPTreeMaxSplit::SplitInfo
 - direction, 2260
 - splitVal, 2260
- mlpack::tree::RPTreeMeanSplit
 - AssignToLeftNode, 2262
 - ElemType, 2262
 - PerformSplit, 2263
 - SplitNode, 2264
- mlpack::tree::RPTreeMeanSplit::SplitInfo
 - direction, 2265
 - mean, 2265
 - meanSplit, 2265
 - splitVal, 2266
- mlpack::tree::RPlusPlusTreeAuxiliaryInformation
 - BoundType, 2240
 - ElemType, 2240
 - HandleNodeInsertion, 2242
 - HandleNodeRemoval, 2243
 - HandlePointDeletion, 2243
 - HandlePointInsertion, 2244
 - NullifyData, 2244
 - OuterBound, 2244
 - RPlusPlusTreeAuxiliaryInformation, 2240–2242
 - serialize, 2245
 - SplitAuxiliaryInfo, 2245
 - UpdateAuxiliaryInfo, 2245
- mlpack::tree::RPlusPlusTreeDescentHeuristic
 - ChooseDescentNode, 2246, 2247
- mlpack::tree::RPlusPlusTreeSplitPolicy
 - AssignToFirstTree, 2249
 - AssignToSecondTree, 2249
 - Bound, 2248
 - GetSplitPolicy, 2248
 - SplitRequired, 2249
- mlpack::tree::RPlusTreeDescentHeuristic
 - ChooseDescentNode, 2250, 2251
- mlpack::tree::RPlusTreeSplit
 - SplitLeafNode, 2252
 - SplitNonLeafNode, 2253
 - SplitPolicy, 2252

- mlpack::tree::RPlusTreeSplitPolicy
 - AssignToFirstTree, 2255
 - AssignToSecondTree, 2255
 - Bound, 2254
 - GetSplitPolicy, 2254
 - SplitRequired, 2255
- mlpack::tree::RStarTreeDescentHeuristic
 - ChooseDescentNode, 2267
- mlpack::tree::RStarTreeSplit
 - PickLeafSplit, 2268
 - ReinsertPoints, 2268
 - SplitLeafNode, 2268
 - SplitNonLeafNode, 2269
- mlpack::tree::RTreeDescentHeuristic
 - ChooseDescentNode, 2270
- mlpack::tree::RTreeSplit
 - SplitLeafNode, 2271
 - SplitNonLeafNode, 2271
- mlpack::tree::RandomDimensionSelect
 - Begin, 2195
 - Dimensions, 2195
 - End, 2195
 - Next, 2196
 - RandomDimensionSelect, 2195
- mlpack::tree::RandomForest
 - Classify, 2201–2203
 - DecisionTreeType, 2198
 - NumTrees, 2203
 - RandomForest, 2199–2201
 - serialize, 2203
 - Train, 2203–2206
 - Tree, 2206, 2207
- mlpack::tree::RectangleTree
 - ~RectangleTree, 2215
 - AuxiliaryInfo, 2216
 - AuxiliaryInformation, 2212, 2233
 - Begin, 2216
 - Bound, 2216, 2217
 - Center, 2217
 - Child, 2217, 2218
 - CondenseTree, 2218
 - Count, 2218, 2219
 - Dataset, 2219
 - DeletePoint, 2219
 - Descendant, 2220
 - DescentType, 2233
 - ElemType, 2212
 - ExactClone, 2220
 - FindByBeginCount, 2220, 2221
 - FurthestDescendantDistance, 2221
 - FurthestPointDistance, 2221
 - GetFurthestChild, 2222
 - GetNearestChild, 2222
 - InsertNode, 2223
 - InsertPoint, 2223
 - IsLeaf, 2224
 - Mat, 2212
 - MaxDistance, 2224
 - MaxLeafSize, 2224, 2225
 - MaxNumChildren, 2225
 - Metric, 2225
 - MinDistance, 2225, 2226
 - MinLeafSize, 2226
 - MinNumChildren, 2227
 - MinimumBoundDistance, 2226
 - NullifyData, 2227
 - NumChildren, 2227
 - NumDescendants, 2228
 - NumPoints, 2228
 - Parent, 2228
 - ParentDistance, 2229
 - Point, 2229
 - RangeDistance, 2230
 - RectangleTree, 2213–2215
 - RemoveNode, 2230
 - serialize, 2231
 - ShrinkBoundForBound, 2231
 - ShrinkBoundForPoint, 2231
 - SoftDelete, 2232
 - SplitType, 2233
 - Stat, 2232
 - TreeDepth, 2232
 - TreeSize, 2232
- mlpack::tree::RectangleTree::DualTreeTraverser
 - DualTreeTraverser, 2235
 - NumBaseCases, 2235
 - NumPrunes, 2235
 - NumScores, 2236
 - NumVisited, 2236
 - Traverse, 2236
- mlpack::tree::RectangleTree::SingleTreeTraverser
 - NumPrunes, 2238
 - SingleTreeTraverser, 2238
 - Traverse, 2238
- mlpack::tree::SpaceSplit
 - GetProjVector, 2272, 2273
- mlpack::tree::SpillTree
 - ~SpillTree, 2283
 - Bound, 2283
 - BoundType, 2278
 - Center, 2284
 - Child, 2284
 - ChildPtr, 2284
 - Dataset, 2285
 - DefeatistDualTreeTraverser, 2278
 - DefeatistSingleTreeTraverser, 2278
 - Descendant, 2285
 - DualTreeTraverser, 2278

- ElemType, 2279
- FurthestDescendantDistance, 2285
- FurthestPointDistance, 2285
- GetFurthestChild, 2286
- GetNearestChild, 2286
- HasSelfChildren, 2287
- Hyperplane, 2287
- IsLeaf, 2287
- Left, 2287
- Mat, 2279
- MaxDistance, 2288
- Metric, 2288
- MinDistance, 2288, 2289
- MinimumBoundDistance, 2289
- NumChildren, 2289
- NumDescendants, 2289
- NumPoints, 2290
- Overlap, 2290
- Parent, 2290
- ParentDistance, 2291
- Point, 2291
- RangeDistance, 2292
- Right, 2292
- serialize, 2292
- SingleTreeTraverser, 2279
- SpillTree, 2279–2281, 2283
- Stat, 2293
- mlpack::tree::SpillTree::SpillDualTreeTraverser
 - NumBaseCases, 2295
 - NumPrunes, 2295
 - NumScores, 2296
 - NumVisited, 2296
 - SpillDualTreeTraverser, 2294
 - Traverse, 2296
- mlpack::tree::SpillTree::SpillSingleTreeTraverser
 - NumPrunes, 2298
 - SpillSingleTreeTraverser, 2298
 - Traverse, 2298
- mlpack::tree::TraversalInfo
 - LastBaseCase, 2300, 2301
 - LastQueryNode, 2301
 - LastReferenceNode, 2301
 - LastScore, 2302
 - TraversalInfo, 2300
- mlpack::tree::TreeTraits
 - BinaryTree, 2303
 - FirstPointIsCentroid, 2304
 - HasDuplicatedPoints, 2304
 - HasOverlappingChildren, 2304
 - HasSelfChildren, 2304
 - RearrangesDataset, 2304
 - UniqueNumDescendants, 2305
- mlpack::tree::TreeTraits< BinarySpaceTree< Metric←
Type, StatisticType, MatType, bound::BallBound,
SplitType > >
 - BinaryTree, 2306
 - FirstPointIsCentroid, 2306
 - HasDuplicatedPoints, 2306
 - HasOverlappingChildren, 2306
 - HasSelfChildren, 2306
 - RearrangesDataset, 2306
 - UniqueNumDescendants, 2307
- mlpack::tree::TreeTraits< BinarySpaceTree< Metric←
Type, StatisticType, MatType, bound::CellBound,
SplitType > >
 - BinaryTree, 2308
 - FirstPointIsCentroid, 2308
 - HasDuplicatedPoints, 2308
 - HasOverlappingChildren, 2308
 - HasSelfChildren, 2308
 - RearrangesDataset, 2308
 - UniqueNumDescendants, 2309
- mlpack::tree::TreeTraits< BinarySpaceTree< MetricType,
StatisticType, MatType, bound::HollowBall←
Bound, SplitType > >
 - BinaryTree, 2310
 - FirstPointIsCentroid, 2310
 - HasDuplicatedPoints, 2310
 - HasOverlappingChildren, 2310
 - HasSelfChildren, 2310
 - RearrangesDataset, 2310
 - UniqueNumDescendants, 2311
- mlpack::tree::TreeTraits< BinarySpaceTree< MetricType,
StatisticType, MatType, BoundType, RPTree←
MaxSplit > >
 - BinaryTree, 2312
 - FirstPointIsCentroid, 2312
 - HasDuplicatedPoints, 2312
 - HasOverlappingChildren, 2312
 - HasSelfChildren, 2313
 - RearrangesDataset, 2313
 - UniqueNumDescendants, 2313
- mlpack::tree::TreeTraits< BinarySpaceTree< MetricType,
StatisticType, MatType, BoundType, RPTree←
MeanSplit > >
 - BinaryTree, 2314
 - FirstPointIsCentroid, 2314
 - HasDuplicatedPoints, 2315
 - HasOverlappingChildren, 2315
 - HasSelfChildren, 2315
 - RearrangesDataset, 2315
 - UniqueNumDescendants, 2315
- mlpack::tree::TreeTraits< BinarySpaceTree< MetricType,
StatisticType, MatType, BoundType, SplitType >
>
 - BinaryTree, 2317
 - FirstPointIsCentroid, 2317
 - HasDuplicatedPoints, 2317

- HasOverlappingChildren, 2317
- HasSelfChildren, 2317
- RearrangesDataset, 2318
- UniqueNumDescendants, 2318
- mlpack::tree::TreeTraits< CoverTree< MetricType,
 - StatisticType, MatType, RootPointPolicy > >
 BinaryTree, 2319
- FirstPointIsCentroid, 2319
- HasDuplicatedPoints, 2320
- HasOverlappingChildren, 2320
- HasSelfChildren, 2320
- RearrangesDataset, 2320
- UniqueNumDescendants, 2320
- mlpack::tree::TreeTraits< Octree< MetricType, Statistic←
 - Type, MatType > >
 BinaryTree, 2322
- FirstPointIsCentroid, 2322
- HasDuplicatedPoints, 2322
- HasOverlappingChildren, 2322
- HasSelfChildren, 2322
- RearrangesDataset, 2323
- UniqueNumDescendants, 2323
- mlpack::tree::TreeTraits< RectangleTree< MetricType,
 - StatisticType, MatType, RPlusTreeSplit< Split←
 - PolicyType, SweepType >, DescentType,
 - AuxiliaryInformationType > >
 BinaryTree, 2324
- FirstPointIsCentroid, 2324
- HasDuplicatedPoints, 2325
- HasOverlappingChildren, 2325
- HasSelfChildren, 2325
- RearrangesDataset, 2325
- UniqueNumDescendants, 2325
- mlpack::tree::TreeTraits< RectangleTree< MetricType,
 - StatisticType, MatType, SplitType, DescentType,
 - AuxiliaryInformationType > >
 BinaryTree, 2327
- FirstPointIsCentroid, 2327
- HasDuplicatedPoints, 2327
- HasOverlappingChildren, 2327
- HasSelfChildren, 2327
- RearrangesDataset, 2328
- UniqueNumDescendants, 2328
- mlpack::tree::TreeTraits< SpillTree< MetricType, Statistic←
 - Type, MatType, HyperplaneType, SplitType > >
 BinaryTree, 2329
- FirstPointIsCentroid, 2329
- HasOverlappingChildren, 2330
- HasSelfChildren, 2330
- RearrangesDataset, 2330
- UniqueNumDescendants, 2330
- mlpack::tree::VantagePointSplit
 - AssignToLeftNode, 2333
 - ElemType, 2332
 - MetricType, 2332
 - PerformSplit, 2333, 2334
 - SplitNode, 2334
- mlpack::tree::VantagePointSplit::SplitInfo
 - metric, 2337
 - mu, 2337
 - SplitInfo, 2337
 - vantagePoint, 2337
- mlpack::tree::XTreeAuxiliaryInformation
 - HandleNodeInsertion, 2341
 - HandleNodeRemoval, 2341
 - HandlePointDeletion, 2342
 - HandlePointInsertion, 2342
 - NormalNodeMaxNumChildren, 2343
 - NullifyData, 2343
 - operator=, 2343
 - serialize, 2344
 - SplitHistory, 2344
 - SplitHistoryStruct, 2339
 - UpdateAuxiliaryInfo, 2344
 - XTreeAuxiliaryInformation, 2340, 2341
- mlpack::tree::XTreeAuxiliaryInformation::SplitHistoryStruct
 - history, 2347
 - lastDimension, 2347
 - operator=, 2346
 - serialize, 2346
 - SplitHistoryStruct, 2345, 2346
- mlpack::tree::XTreeSplit
 - SplitLeafNode, 2348
 - SplitNonLeafNode, 2348
- mlpack::tree::enumerate, 464
 - EnumerateTreeImpl, 464
- mlpack::tree::split, 465
 - PerformSplit, 465
- mlpack::util, 466
 - DisableBacktrace, 468
 - DisableVerbose, 468
 - EnableTimers, 468
 - EnableVerbose, 468
 - GetParamPtr, 469
 - GetParamWithInfo, 469
 - GetVersion, 469
 - HyphenateString, 469
 - ReportIgnoredParam, 470
 - RequireAtLeastOnePassed, 471
 - RequireNoneOrAllPassed, 471
 - RequireOnlyOnePassed, 472
 - RequireParamInSet, 473
 - RequireParamValue, 473
 - ResetTimers, 474
 - SetInputParam, 474
 - SetParam, 475
 - SetParamPtr, 475
 - SetParamWithInfo, 475

- mlpack::util::IsStdVector
 - value, 2349
- mlpack::util::IsStdVector< std::vector< T, A > >
 - value, 2350
- mlpack::util::NullOutputStream
 - NullOutputStream, 2351
 - operator<<, 2352–2356
- mlpack::util::ParamData
 - alias, 2357
 - cppType, 2358
 - desc, 2358
 - input, 2358
 - loaded, 2358
 - name, 2359
 - noTranspose, 2359
 - persistent, 2359
 - required, 2360
 - tname, 2360
 - value, 2360
 - wasPassed, 2361
- mlpack::util::PrefixedOutputStream
 - backtrace, 2367
 - destination, 2368
 - ignoreInput, 2368
 - operator<<, 2364–2367
 - PrefixedOutputStream, 2363
- mlpack::util::ProgramDoc
 - documentation, 2370
 - ProgramDoc, 2369, 2370
 - programName, 2370
 - seeAlso, 2370
 - shortDocumentation, 2370
- mlpack_deprecated
 - deprecated.hpp, 2695
- mlpack_main.hpp
 - BINDING_TYPE_CLI, 2700
 - BINDING_TYPE_MARKDOWN, 2700
 - BINDING_TYPE_PYX, 2701
 - BINDING_TYPE_TEST, 2701
 - BINDING_TYPE_UNKNOWN, 2701
 - BINDING_TYPE, 2700
- mock_categorical_data.hpp
 - MockCategoricalData, 3085
- MockCategoricalData
 - mock_categorical_data.hpp, 3085
- Mode
 - mlpack::kde::KDEModel, 1402, 1403
 - mlpack::kde::KDE, 1394
- ModeVisitor, 1412
- Model
 - mlpack::ann::AddMerge, 563
 - mlpack::ann::Concat, 628
 - mlpack::ann::DropConnect, 671
 - mlpack::ann::GRU, 735
 - mlpack::ann::MultiplyMerge, 833
 - mlpack::ann::Recurrent, 891
 - mlpack::ann::RecurrentAttention, 897
 - mlpack::ann::Sequential, 933
 - mlpack::cv::KFoldCV, 1139
 - mlpack::cv::SimpleCV, 1157
- ModelToString
 - range_search_utils.hpp, 3083
- ModifiedGramSchmidt
 - mlpack::tree::CosineTree, 2037
- MonoSearchVisitor, 1618, 1752
 - mlpack::neighbor::MonoSearchVisitor, 1619
 - mlpack::range::MonoSearchVisitor, 1752
- MonteCarloError
 - mlpack::tree::CosineTree, 2038
- MountainCar, 1861
 - mlpack::rl::MountainCar, 1862
- MountainCar::State, 1865
- mu
 - mlpack::tree::VantagePointSplit::SplitInfo, 2337
- MultipleRandomDimensionSelect, 2158
 - mlpack::tree::MultipleRandomDimensionSelect, 2158
- MultiplyConstant
 - mlpack::ann::MultiplyConstant, 826
- MultiplyConstant< InputDataType, OutputDataType >, 825
- MultiplyMerge
 - mlpack::ann::MultiplyMerge, 830
- MultiplyMerge< InputDataType, OutputDataType, CustomLayers >, 829
- NCA< MetricType, OptimizerType >, 1583
- NCA
 - mlpack::nca::NCA, 1584
- NMFALSFactorizer
 - mlpack::amf, 259
- NMFALSUpdate, 516
 - mlpack::amf::NMFALSUpdate, 517
- NMFMultiplicativeDistanceUpdate, 519
 - mlpack::amf::NMFMultiplicativeDistanceUpdate, 520
- NMFMultiplicativeDivergenceUpdate, 523
 - mlpack::amf::NMFMultiplicativeDivergenceUpdate, 524
- NMFPolicy, 1066
- NNSTreeType
 - mlpack::kmeans::DualTreeKMeans, 1467
- NSModel
 - mlpack::neighbor::NSModel, 1660
- NSModel< SortPolicy >, 1657
- NSType
 - mlpack::neighbor, 433
- NSTypeT
 - mlpack::neighbor::BiSearchVisitor, 1592
 - mlpack::neighbor::TrainVisitor, 1708

- NStepQLearning
 - mlpack::rl, 442
- NStepQLearningWorker
 - mlpack::rl::NStepQLearningWorker, 1870
- NStepQLearningWorker< EnvironmentType, Network↔
Type, UpdaterType, PolicyType >, 1868
- Naive
 - mlpack::fastmks::FastMKModel, 1295, 1296
 - mlpack::fastmks::FastMKS, 1286, 1287
 - mlpack::neighbor::RAModel, 1674
 - mlpack::neighbor::RASearch, 1686, 1687
 - mlpack::range::RModel, 1776
 - mlpack::range::RangeSearch, 1759
- NaiveBayesClassifier
 - mlpack::naive_bayes::NaiveBayesClassifier, 1578, 1579
- NaiveBayesClassifier< ModelMatType >, 1576
- NaiveConvolution< BorderMode >, 834
- NaiveKMeans
 - mlpack::kmeans::NaiveKMeans, 1492
- NaiveKMeans< MetricType, MatType >, 1491
- NaiveKernelRule< KernelType >, 1511
- NaiveVisitor, 1620, 1753
- name
 - mlpack::util::ParamData, 2359
- NearestNeighborSort
 - mlpack::neighbor, 433
- NearestNS, 1621
- NeedsFirstPass
 - mlpack::data::IncrementPolicy, 1187
 - mlpack::data::MissingPolicy, 1197
- NeedsMinimization
 - mlpack::cv::Accuracy, 1128
 - mlpack::cv::F1, 1134
 - mlpack::cv::MSE, 1144
 - mlpack::cv::Precision, 1146
 - mlpack::cv::Recall, 1148
- NegativeLogLikelihood
 - mlpack::ann::NegativeLogLikelihood, 838
- NegativeLogLikelihood< InputDataType, OutputDataType
>, 837
- NeighborSearch
 - mlpack::neighbor::NeighborSearch, 1630–1632
- NeighborSearch< SortPolicy, MetricType, MatType,
TreeType, DualTreeTraversalType, SingleTree↔
TraversalType >, 1627
- NeighborSearchMode
 - mlpack::neighbor, 434
- NeighborSearchRules
 - mlpack::neighbor::NeighborSearchRules, 1643
- NeighborSearchRules< SortPolicy, MetricType, TreeType
>, 1640
- NeighborSearchRules< SortPolicy, MetricType, TreeType
>::CandidateCmp, 1652
- NeighborSearchStat
 - mlpack::neighbor::NeighborSearchStat, 1654
- NeighborSearchStat< SortPolicy >, 1653
- NeighborSearchType
 - mlpack::cf::LMetricSearch, 1065
- Network
 - mlpack::rl::AsyncLearning, 1840
 - mlpack::rl::QLearning, 1889
- NetworkInitialization
 - mlpack::ann::NetworkInitialization, 841
- NetworkInitialization< InitializationRuleType, Custom↔
Layers >, 841
- NetworkType
 - mlpack::ann::BRNN, 612
 - mlpack::ann::FFN, 694
 - mlpack::ann::RBM, 868
 - mlpack::ann::RNN, 913
- NewtonTolerance
 - mlpack::sparse_coding::SparseCoding, 1922
- Next
 - mlpack::tree::AllDimensionSelect, 1986
 - mlpack::tree::MultipleRandomDimensionSelect, 2159
 - mlpack::tree::RandomDimensionSelect, 2196
- NguyenWidrowInitialization, 842
 - mlpack::ann::NguyenWidrowInitialization, 843
- nm
 - mlpack::amf::SimpleResidueTermination, 534
- No
 - mlpack::hpt::DeduceHyperParameterTypes< T,
Args... >::IsCollectionType, 1370
- no
 - mlpack::data::HasSerialize, 1179
- NoAuxiliaryInformation
 - mlpack::tree::NoAuxiliaryInformation, 2161
- NoAuxiliaryInformation< TreeType >, 2160
- NoConstraint, 1332
- NoNormalization, 1070
 - mlpack::cf::NoNormalization, 1070
- noTranspose
 - mlpack::util::ParamData, 2359
- NoiseStdDev
 - mlpack::radical::Radical, 1746
- NonOrtMeanSPTree
 - mlpack::tree, 456
- NonOrtSPTree
 - mlpack::tree, 457
- NonStaticSerialize
 - mlpack::data::HasSerializeFunction, 1181
- normOld
 - mlpack::amf::SimpleResidueTermination, 534
- NormalNodeMaxNumChildren
 - mlpack::tree::XTreeAuxiliaryInformation, 2343
- Normalization
 - mlpack::cf::CFTYPE, 1049

- Normalizations
 - mlpack::cf::CombinedNormalization, 1055
- Normalize
 - mlpack::cf::CombinedNormalization, 1056
 - mlpack::cf::ItemMeanNormalization, 1063
 - mlpack::cf::NoNormalization, 1071
 - mlpack::cf::OverallMeanNormalization, 1074
 - mlpack::cf::UserMeanNormalization, 1113
 - mlpack::cf::ZScoreNormalization, 1116
- NormalizeColByMax
 - mlpack::nn, 437
- NormalizeLabels
 - mlpack::data, 385
- Normalizer
 - mlpack::kernel::EpanechnikovKernel, 1420
 - mlpack::kernel::ExampleKernel, 1423
 - mlpack::kernel::GaussianKernel, 1429
 - mlpack::kernel::SphericalKernel, 1461
- NotFoundMethodForm, 1144
- NothingInitializer, 1914
- now
 - det.txt, 2377
- ns_model.hpp
 - BOOST_TEMPLATE_CLASS_VERSION, 3013
- NullOutputStream, 2350
 - mlpack::util::NullOutputStream, 2351
- NullifyData
 - mlpack::tree::HilbertRTreeAuxiliaryInformation, 2099
 - mlpack::tree::NoAuxiliaryInformation, 2164
 - mlpack::tree::RPlusPlusTreeAuxiliaryInformation, 2244
 - mlpack::tree::RectangleTree, 2227
 - mlpack::tree::XTreeAuxiliaryInformation, 2343
- NumBaseCases
 - mlpack::tree::BinarySpaceTree::BreadthFirstDualTreeTraverser, 2020
 - mlpack::tree::BinarySpaceTree::DualTreeTraverser, 2024
 - mlpack::tree::CoverTree::DualTreeTraverser, 2061
 - mlpack::tree::Octree::DualTreeTraverser, 2185
 - mlpack::tree::RectangleTree::DualTreeTraverser, 2235
 - mlpack::tree::SpillTree::SpillDualTreeTraverser, 2295
- NumChildren
 - mlpack::det::DTree, 1217
 - mlpack::tree::AllCategoricalSplit, 1983
 - mlpack::tree::BestBinaryNumericSplit, 1990
 - mlpack::tree::BinaryNumericSplit, 1995
 - mlpack::tree::BinarySpaceTree, 2015
 - mlpack::tree::CoverTree, 2056
 - mlpack::tree::DecisionTree, 2074
 - mlpack::tree::ExampleTree, 2087
 - mlpack::tree::HoeffdingCategoricalSplit, 2107
 - mlpack::tree::HoeffdingNumericSplit, 2112
 - mlpack::tree::HoeffdingTree, 2124
 - mlpack::tree::Octree, 2181
 - mlpack::tree::RectangleTree, 2227
 - mlpack::tree::SpillTree, 2289
- NumClasses
 - mlpack::adaboost::AdaBoost, 492
 - mlpack::perceptron::Perceptron, 1739
 - mlpack::regression::SoftmaxRegression, 1816
 - mlpack::regression::SoftmaxRegressionFunction, 1824
 - mlpack::svm::LinearSVM, 1965
 - mlpack::tree::DecisionTree, 2075
- NumColumns
 - mlpack::tree::CosineTree, 2038
- NumDescendants
 - mlpack::tree::BinarySpaceTree, 2015
 - mlpack::tree::CoverTree, 2057
 - mlpack::tree::ExampleTree, 2087
 - mlpack::tree::HoeffdingTree, 2124
 - mlpack::tree::Octree, 2181
 - mlpack::tree::RectangleTree, 2228
 - mlpack::tree::SpillTree, 2289
- NumDistComputations
 - mlpack::neighbor::RASearchRules, 1695
- NumEffectiveSamples
 - mlpack::neighbor::RASearchRules, 1695
- NumFeatures
 - mlpack::regression::LogisticRegressionFunction, 1809
 - mlpack::regression::SoftmaxRegressionFunction, 1824
- NumFunctions
 - mlpack::ann::BRNN, 616
 - mlpack::ann::FFN, 700
 - mlpack::ann::RBM, 872
 - mlpack::ann::RNN, 916
 - mlpack::lmnn::LMNNFunction, 1537
 - mlpack::nca::SoftmaxErrorFunction, 1589
 - mlpack::regression::LogisticRegressionFunction, 1809
 - mlpack::svd::BiasSVDFunction, 1930
 - mlpack::svd::RegularizedSVDFunction, 1949
 - mlpack::svd::SVDPlusPlusFunction, 1958
 - mlpack::svm::LinearSVMFunction, 1974
- NumItems
 - mlpack::svd::BiasSVDFunction, 1931
 - mlpack::svd::RegularizedSVDFunction, 1950
 - mlpack::svd::SVDPlusPlusFunction, 1958
- NumMappings
 - mlpack::data::DatasetMapper, 1175
- NumNodes
 - mlpack::det::PathCacher, 1224
 - mlpack::tree::HoeffdingTreeModel, 2132
- NumPoints

- mlpack::tree::BinarySpaceTree, 2015
- mlpack::tree::CoverTree, 2057
- mlpack::tree::ExampleTree, 2087
- mlpack::tree::Octree, 2181
- mlpack::tree::RectangleTree, 2228
- mlpack::tree::SpillTree, 2290
- NumProjections
 - mlpack::neighbor::LSHSearch, 1613
 - mlpack::neighbor::QDAFN, 1667
- NumPrunes
 - mlpack::tree::BinarySpaceTree::BreadthFirstDual↵TreeTraverser, 2020, 2021
 - mlpack::tree::BinarySpaceTree::DualTreeTraverser, 2024
 - mlpack::tree::BinarySpaceTree::SingleTreeTraverser, 2027
 - mlpack::tree::CoverTree::DualTreeTraverser, 2062
 - mlpack::tree::CoverTree::SingleTreeTraverser, 2064
 - mlpack::tree::GreedySingleTreeTraverser, 2094
 - mlpack::tree::Octree::DualTreeTraverser, 2185, 2186
 - mlpack::tree::Octree::SingleTreeTraverser, 2188
 - mlpack::tree::RectangleTree::DualTreeTraverser, 2235
 - mlpack::tree::RectangleTree::SingleTreeTraverser, 2238
 - mlpack::tree::SpillTree::SpillDualTreeTraverser, 2295
 - mlpack::tree::SpillTree::SpillSingleTreeTraverser, 2298
- NumSamplesMade
 - mlpack::neighbor::RAQueryStat, 1680
- NumScores
 - mlpack::tree::BinarySpaceTree::BreadthFirstDual↵TreeTraverser, 2021
 - mlpack::tree::BinarySpaceTree::DualTreeTraverser, 2024, 2025
 - mlpack::tree::CoverTree::DualTreeTraverser, 2062
 - mlpack::tree::Octree::DualTreeTraverser, 2186
 - mlpack::tree::RectangleTree::DualTreeTraverser, 2236
 - mlpack::tree::SpillTree::SpillDualTreeTraverser, 2296
- NumSteps
 - mlpack::ann::RBM, 872
- NumTestPoints
 - mlpack::amf::ValidationRMSETermination, 553
- NumTrees
 - mlpack::tree::RandomForest, 2203
- NumTrueChildren
 - mlpack::kmeans::DualTreeKMeansStatistic, 1475
- NumUnmappings
 - mlpack::data::DatasetMapper, 1175
- NumUsers
 - mlpack::svd::BiasSVDFFunction, 1931
 - mlpack::svd::RegularizedSVDFFunction, 1950
 - mlpack::svd::SVDPlusPlusFunction, 1958
- NumUsersForSimilarity
 - mlpack::cf::CFTYPE, 1049, 1050
- NumVisited
 - mlpack::tree::BinarySpaceTree::BreadthFirstDual↵TreeTraverser, 2021
 - mlpack::tree::BinarySpaceTree::DualTreeTraverser, 2025
 - mlpack::tree::CoverTree::DualTreeTraverser, 2062
 - mlpack::tree::Octree::DualTreeTraverser, 2186
 - mlpack::tree::RectangleTree::DualTreeTraverser, 2236
 - mlpack::tree::SpillTree::SpillDualTreeTraverser, 2296
- NumVisted
 - mlpack::tree::Octree::DualTreeTraverser, 2186
- NumWorkers
 - mlpack::rl::TrainingConfig, 1904
- NumericSplit
 - mlpack::tree::DecisionTree, 2068
 - mlpack::tree::HoeffdingTree, 2116
- NumericSplitInfo
 - mlpack::tree::NumericSplitInfo, 2166
- NumericSplitInfo< ObservationType >, 2165
- NystroemKernelRule< KernelType, PointSelectionPolicy >, 1512
- NystroemMethod
 - mlpack::kernel::NystroemMethod, 1449
- NystroemMethod< KernelType, PointSelectionPolicy >, 1448
- ObjTolerance
 - mlpack::sparse_coding::SparseCoding, 1923
- Objective
 - mlpack::lcc::LocalCoordinateCoding, 1518
 - mlpack::sparse_coding::SparseCoding, 1923
- ObtainDistinctSamples
 - mlpack::math, 415
 - mlpack::neighbor::RAUtil, 1700
- Octree
 - mlpack::tree::Octree, 2170–2175
- Octree< MetricType, StatisticType, MatType >, 2167
- Octree< MetricType, StatisticType, MatType >::Dual↵TreeTraverser< MetricType, StatisticType, Mat↵Type >, 2184
- Octree< MetricType, StatisticType, MatType >::Single↵TreeTraverser< RuleType >, 2187
- Octree< MetricType, StatisticType, MatType >::Split↵Type::SplitInfo, 2189
- Offset
 - mlpack::kernel::HyperbolicTangentKernel, 1432
 - mlpack::kernel::PolynomialKernel, 1453
- Offsets
 - mlpack::neighbor::LSHSearch, 1614
- OivsInitialization
 - mlpack::ann::OivsInitialization, 845

- OivsInitialization< ActivationFunction >, 844
- omp_size_t
 - prereqs.hpp, 3077
- OneHotEncoding
 - mlpack::data, 385
- OneStepQLearning
 - mlpack::rl, 442
- OneStepQLearningWorker
 - mlpack::rl::OneStepQLearningWorker, 1873
- OneStepQLearningWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >, 1871
- OneStepSarsa
 - mlpack::rl, 442
- OneStepSarsaWorker
 - mlpack::rl::OneStepSarsaWorker, 1876
- OneStepSarsaWorker< EnvironmentType, NetworkType, UpdaterType, PolicyType >, 1875
- operator &
 - mlpack::bound::HRectBound, 1025
 - mlpack::math::RangeType, 1553
- operator &=
 - mlpack::bound::HRectBound, 1025
 - mlpack::math::RangeType, 1554
- operator!=
 - mlpack::math::RangeType, 1554
- operator<
 - mlpack::math::RangeType, 1555
- operator<<
 - mlpack::util::NullOutputStream, 2352–2356
 - mlpack::util::PrefixedOutputStream, 2364–2367
- operator>
 - mlpack::math::RangeType, 1556
- operator*
 - mlpack::math::RangeType, 1554, 1557
- operator*=
 - mlpack::math::RangeType, 1555
- operator()
 - mlpack::ann::AddVisitor, 566
 - mlpack::ann::BackwardVisitor, 588
 - mlpack::ann::CopyVisitor, 652
 - mlpack::ann::DeleteVisitor, 660
 - mlpack::ann::DeltaVisitor, 661
 - mlpack::ann::DeterministicSetVisitor, 662
 - mlpack::ann::ForwardVisitor, 714
 - mlpack::ann::GradientSetVisitor, 726
 - mlpack::ann::GradientUpdateVisitor, 728
 - mlpack::ann::GradientVisitor, 729
 - mlpack::ann::GradientZeroVisitor, 730
 - mlpack::ann::LoadOutputParameterVisitor, 788
 - mlpack::ann::LossVisitor, 801
 - mlpack::ann::OutputHeightVisitor, 850
 - mlpack::ann::OutputParameterVisitor, 851
 - mlpack::ann::OutputWidthVisitor, 852
 - mlpack::ann::ParametersSetVisitor, 854
 - mlpack::ann::ParametersVisitor, 855
 - mlpack::ann::ResetCellVisitor, 908
 - mlpack::ann::ResetVisitor, 909
 - mlpack::ann::RewardSetVisitor, 910
 - mlpack::ann::RunSetVisitor, 922
 - mlpack::ann::SaveOutputParameterVisitor, 924
 - mlpack::ann::SetInputHeightVisitor, 935
 - mlpack::ann::SetInputWidthVisitor, 936
 - mlpack::ann::WeightSetVisitor, 973
 - mlpack::ann::WeightSizeVisitor, 974
 - mlpack::cf::DeleteVisitor, 1059
 - mlpack::cf::GetValueVisitor, 1060
 - mlpack::cf::PredictVisitor, 1078
 - mlpack::cf::RecommendationVisitor, 1085
 - mlpack::kde::DeleteVisitor, 1384
 - mlpack::kde::DualBiKDE, 1385
 - mlpack::kde::DualMonoKDE, 1387
 - mlpack::kde::ModeVisitor, 1412
 - mlpack::kde::TrainVisitor, 1413
 - mlpack::neighbor::AlphaVisitor, 1591
 - mlpack::neighbor::BiSearchVisitor, 1594, 1595
 - mlpack::neighbor::DeleteVisitor, 1596
 - mlpack::neighbor::EpsilonVisitor, 1601
 - mlpack::neighbor::FirstLeafExactVisitor, 1602
 - mlpack::neighbor::MonoSearchVisitor, 1619, 1620
 - mlpack::neighbor::NaiveVisitor, 1621
 - mlpack::neighbor::NeighborSearchRules::CandidateCmp, 1652
 - mlpack::neighbor::ReferenceSetVisitor, 1701, 1702
 - mlpack::neighbor::SampleAtLeavesVisitor, 1703
 - mlpack::neighbor::SearchModeVisitor, 1704
 - mlpack::neighbor::SingleModeVisitor, 1705
 - mlpack::neighbor::SingleSampleLimitVisitor, 1706
 - mlpack::neighbor::TauVisitor, 1707
 - mlpack::neighbor::TrainVisitor, 1709–1711
 - mlpack::range::BiSearchVisitor, 1749, 1750
 - mlpack::range::DeleteVisitor, 1751
 - mlpack::range::MonoSearchVisitor, 1753
 - mlpack::range::NaiveVisitor, 1754
 - mlpack::range::ReferenceSetVisitor, 1771
 - mlpack::range::SingleModeVisitor, 1780
 - mlpack::range::TrainVisitor, 1782
 - mlpack::sfinae::MethodFormDetector< MethodForm, 0 >, 1907
 - mlpack::sfinae::MethodFormDetector< MethodForm, 1 >, 1908
 - mlpack::sfinae::MethodFormDetector< MethodForm, 2 >, 1909
 - mlpack::sfinae::MethodFormDetector< MethodForm, 3 >, 1909
 - mlpack::sfinae::MethodFormDetector< MethodForm, 4 >, 1910
 - mlpack::sfinae::MethodFormDetector< MethodForm, 5 >, 1911

- mlpack::sfinae::MethodFormDetector< MethodForm, 6 >, 1911
- mlpack::sfinae::MethodFormDetector< MethodForm, 7 >, 1912
- mlpack::tree::CompareCosineNode, 2029
- operator=
 - mlpack::adaboost::AdaBoostModel, 497
 - mlpack::ann::FFN, 701
 - mlpack::bound::BallBound, 999
 - mlpack::bound::HRectBound, 1026
 - mlpack::bound::HollowBallBound, 1015
 - mlpack::det::DTree, 1218
 - mlpack::fastmks::FastMKModel, 1296
 - mlpack::fastmks::FastMKS, 1287
 - mlpack::gmm::DiagonalGMM, 1313
 - mlpack::gmm::GMM, 1329
 - mlpack::hmm::HMMModel, 1353
 - mlpack::kde::KDEModel, 1403
 - mlpack::kde::KDE, 1394
 - mlpack::metric::IPMetric, 1568
 - mlpack::neighbor::LSHSearch, 1614
 - mlpack::neighbor::NSModel, 1662
 - mlpack::neighbor::NeighborSearch, 1634
 - mlpack::neighbor::RAModel, 1674, 1675
 - mlpack::range::RSModel, 1776
 - mlpack::range::RangeSearch, 1759
 - mlpack::tree::DecisionTree, 2075
 - mlpack::tree::HilbertRTreeAuxiliaryInformation, 2100
 - mlpack::tree::HoeffdingTreeModel, 2132
 - mlpack::tree::NoAuxiliaryInformation, 2164
 - mlpack::tree::XTreeAuxiliaryInformation, 2343
 - mlpack::tree::XTreeAuxiliaryInformation::Split← HistoryStruct, 2346
- operator==
 - mlpack::math::RangeType, 1555
- operator[]
 - mlpack::bound::BallBound, 999
 - mlpack::bound::HRectBound, 1026
 - mlpack::bound::HollowBallBound, 1015
- operator |
 - mlpack::math::RangeType, 1556
- operator | =
 - mlpack::bound::BallBound, 999, 1000
 - mlpack::bound::HRectBound, 1026, 1027
 - mlpack::bound::HollowBallBound, 1016
 - mlpack::math::RangeType, 1556
- Optimize
 - mlpack::hpt::HyperParameterTuner, 1378
- OptimizeDictionary
 - mlpack::lcc::LocalCoordinateCoding, 1518
 - mlpack::sparse_coding::SparseCoding, 1923
- Optimizer
 - mlpack::hpt::HyperParameterTuner, 1379
 - mlpack::lmnn::LMNN, 1529, 1530
- Class,
 - mlpack::nca::NCA, 1586
- OrderedPointSelection, 1200
- Class,
 - OrderedSelection, 1450
- OrthogonalInitialization, 848
 - mlpack::ann::OrthogonalInitialization, 848
- Orthogonalize
 - mlpack::math, 416
- OuterBound
 - mlpack::tree::RPlusPlusTreeAuxiliaryInformation, 2244
- OuterRadius
 - mlpack::bound::HollowBallBound, 1016, 1017
- OutputHeight
 - mlpack::ann::AtrousConvolution, 578
 - mlpack::ann::Convolution, 649
 - mlpack::ann::Glimpse, 721
 - mlpack::ann::MaxPooling, 812
 - mlpack::ann::MeanPooling, 820
 - mlpack::ann::TransposedConvolution, 964, 965
- OutputHeightVisitor, 850
- OutputParam
 - mlpack::bindings::cli, 300
- OutputParamImpl
 - mlpack::bindings::cli, 300, 301
- OutputParameter
 - mlpack::ann::Add, 557
 - mlpack::ann::AddMerge, 564
 - mlpack::ann::AlphaDropout, 571
 - mlpack::ann::AtrousConvolution, 578, 579
 - mlpack::ann::BaseLayer, 591
 - mlpack::ann::BatchNorm, 597, 598
 - mlpack::ann::BilinearInterpolation, 609
 - mlpack::ann::CReLU, 655
 - mlpack::ann::Concat, 628, 629
 - mlpack::ann::ConcatPerformance, 637
 - mlpack::ann::Concatenate, 633
 - mlpack::ann::Constant, 640
 - mlpack::ann::Convolution, 649, 650
 - mlpack::ann::CrossEntropyError, 658
 - mlpack::ann::DiceLoss, 665
 - mlpack::ann::DropConnect, 671
 - mlpack::ann::Dropout, 676
 - mlpack::ann::ELU, 684, 685
 - mlpack::ann::EarthMoverDistance, 679
 - mlpack::ann::FastLSTM, 690
 - mlpack::ann::FlexibleReLU, 712
 - mlpack::ann::GRU, 735, 736
 - mlpack::ann::Glimpse, 721, 722
 - mlpack::ann::HardTanH, 743, 744
 - mlpack::ann::Join, 755
 - mlpack::ann::KLDivergence, 761
 - mlpack::ann::LSTM, 806
 - mlpack::ann::LayerNorm, 767
 - mlpack::ann::LeakyReLU, 774

- mlpack::ann::Linear, 781
- mlpack::ann::LinearNoBias, 786
- mlpack::ann::LogSoftMax, 794, 795
- mlpack::ann::Lookup, 799
- mlpack::ann::MaxPooling, 812, 813
- mlpack::ann::MeanPooling, 820
- mlpack::ann::MeanSquaredError, 824, 825
- mlpack::ann::MultiplyConstant, 828
- mlpack::ann::MultiplyMerge, 833
- mlpack::ann::NegativeLogLikelihood, 840
- mlpack::ann::PReLU, 860
- mlpack::ann::ReconstructionLoss, 882
- mlpack::ann::Recurrent, 891
- mlpack::ann::RecurrentAttention, 897
- mlpack::ann::ReinforceNormal, 902
- mlpack::ann::Reparametrization, 906
- mlpack::ann::Select, 926, 927
- mlpack::ann::Sequential, 933
- mlpack::ann::SigmoidCrossEntropyError, 939
- mlpack::ann::Subview, 950
- mlpack::ann::TransposedConvolution, 965
- mlpack::ann::VRClassReward, 971
- OutputParameterVisitor, 851
- OutputSize
 - mlpack::ann::Reparametrization, 906, 907
- OutputWidth
 - mlpack::ann::AtrousConvolution, 579
 - mlpack::ann::Convolution, 650
 - mlpack::ann::Glimpse, 722
 - mlpack::ann::MaxPooling, 813
 - mlpack::ann::MeanPooling, 821
 - mlpack::ann::TransposedConvolution, 965, 966
- OutputWidthVisitor, 852
- OverallMeanNormalization, 1072
 - mlpack::cf::OverallMeanNormalization, 1073
- Overlap
 - mlpack::bound::HRectBound, 1027
 - mlpack::tree::SpillTree, 2290
- Owner
 - mlpack::kmeans::DualTreeKMeansStatistic, 1475
- OwnsReferenceTree
 - mlpack::kde::KDE, 1395
- P
 - mlpack::cf::BiasSVDPolicy, 1041
 - mlpack::cf::SVDPlusPlusPolicy, 1107
 - mlpack::kernel::PSpectrumStringKernel, 1456, 1457
- PARAM_COL_IN_REQ
 - param.hpp, 2706
- PARAM_COL_IN
 - param.hpp, 2705
- PARAM_COL_OUT
 - param.hpp, 2707
- PARAM_COL
 - param.hpp, 2705
- PARAM_DOUBLE_IN_REQ
 - param.hpp, 2708
- PARAM_DOUBLE_IN
 - param.hpp, 2708
- PARAM_DOUBLE_OUT
 - param.hpp, 2709
- PARAM_FLAG
 - mlpack::bindings::cli, 301
 - param.hpp, 2710
- PARAM_INT_IN_REQ
 - param.hpp, 2712
- PARAM_INT_IN
 - param.hpp, 2711
- PARAM_INT_OUT
 - param.hpp, 2712
- PARAM_IN
 - param.hpp, 2710
- PARAM_MATRIX_AND_INFO_IN
 - param.hpp, 2714
- PARAM_MATRIX_IN_REQ
 - param.hpp, 2715
- PARAM_MATRIX_IN
 - param.hpp, 2714
- PARAM_MATRIX_OUT
 - param.hpp, 2716
- PARAM_MATRIX
 - param.hpp, 2714
- PARAM_MODEL_IN_REQ
 - param.hpp, 2718
- PARAM_MODEL_IN
 - param.hpp, 2717
- PARAM_MODEL_OUT
 - param.hpp, 2718
- PARAM_MODEL
 - param.hpp, 2717
- PARAM_OUT
 - param.hpp, 2719
- PARAM_ROW_IN
 - param.hpp, 2720
- PARAM_ROW_OUT
 - param.hpp, 2721
- PARAM_ROW
 - param.hpp, 2719
- PARAM_STRING_IN_REQ
 - param.hpp, 2722
- PARAM_STRING_IN
 - mlpack::bindings::cli, 302
 - param.hpp, 2721
- PARAM_STRING_OUT
 - param.hpp, 2723
- PARAM_TMATRIX_IN_REQ
 - param.hpp, 2724
- PARAM_TMATRIX_IN

- param.hpp, 2723
- PARAM_TMATRIX_OUT
 - param.hpp, 2725
- PARAM_UCOL_IN
 - param.hpp, 2726
- PARAM_UCOL_OUT
 - param.hpp, 2726
- PARAM_UCOL
 - param.hpp, 2725
- PARAM_UMATRIX_IN_REQ
 - param.hpp, 2728
- PARAM_UMATRIX_IN
 - param.hpp, 2727
- PARAM_UMATRIX_OUT
 - param.hpp, 2729
- PARAM_UMATRIX
 - param.hpp, 2727
- PARAM_UROW_IN
 - param.hpp, 2730
- PARAM_UROW_OUT
 - param.hpp, 2731
- PARAM_UROW
 - param.hpp, 2730
- PARAM_VECTOR_IN_REQ
 - param.hpp, 2732
- PARAM_VECTOR_IN
 - param.hpp, 2731
- PARAM_VECTOR_OUT
 - param.hpp, 2733
- PCA< DecompositionPolicy >, 1723
- PCA
 - mlpack::pca::PCA, 1724
- PROGRAM_INFO
 - param.hpp, 2733
- PReLU< InputDataType, OutputDataType >, 855
- PReLU
 - mlpack::ann::PReLU, 857
- PSpectrumStringKernel, 1454
 - mlpack::kernel::PSpectrumStringKernel, 1455
- param.hpp
 - PARAM_COL_IN_REQ, 2706
 - PARAM_COL_IN, 2705
 - PARAM_COL_OUT, 2707
 - PARAM_COL, 2705
 - PARAM_DOUBLE_IN_REQ, 2708
 - PARAM_DOUBLE_IN, 2708
 - PARAM_DOUBLE_OUT, 2709
 - PARAM_FLAG, 2710
 - PARAM_INT_IN_REQ, 2712
 - PARAM_INT_IN, 2711
 - PARAM_INT_OUT, 2712
 - PARAM_IN, 2710
 - PARAM_MATRIX_AND_INFO_IN, 2714
 - PARAM_MATRIX_IN_REQ, 2715

- PARAM_MATRIX_IN, 2714
- PARAM_MATRIX_OUT, 2716
- PARAM_MATRIX, 2714
- PARAM_MODEL_IN_REQ, 2718
- PARAM_MODEL_IN, 2717
- PARAM_MODEL_OUT, 2718
- PARAM_MODEL, 2717
- PARAM_OUT, 2719
- PARAM_ROW_IN, 2720
- PARAM_ROW_OUT, 2721
- PARAM_ROW, 2719
- PARAM_STRING_IN_REQ, 2722
- PARAM_STRING_IN, 2721
- PARAM_STRING_OUT, 2723
- PARAM_TMATRIX_IN_REQ, 2724
- PARAM_TMATRIX_IN, 2723
- PARAM_TMATRIX_OUT, 2725
- PARAM_UCOL_IN, 2726
- PARAM_UCOL_OUT, 2726
- PARAM_UCOL, 2725
- PARAM_UMATRIX_IN_REQ, 2728
- PARAM_UMATRIX_IN, 2727
- PARAM_UMATRIX_OUT, 2729
- PARAM_UMATRIX, 2727
- PARAM_UROW_IN, 2730
- PARAM_UROW_OUT, 2731
- PARAM_UROW, 2730
- PARAM_VECTOR_IN_REQ, 2732
- PARAM_VECTOR_IN, 2731
- PARAM_VECTOR_OUT, 2733
- PROGRAM_INFO, 2733
- SEE_ALSO, 2734
- TUPLE_TYPE, 2735
- param_data.hpp
 - TYPENAME, 2738
- ParamData, 2356
- ParamString
 - mlpack::bindings::cli, 302
 - mlpack::bindings::markdown, 322
 - mlpack::bindings::python, 344
- ParamType
 - mlpack::bindings::markdown, 322
- ParameterType< arma::Col< eT > >, 978
- ParameterType< arma::Mat< eT > >, 978
- ParameterType< arma::Row< eT > >, 979
- ParameterType< std::tuple< mlpack::data::Dataset←
Mapper< PolicyType, std::string >, arma::Mat<
eT > > >, 980
- ParameterType< T >, 977
- ParameterTypeDeducer< HasSerialize, T >, 981
- ParameterTypeDeducer< true, T >, 981
- Parameters
 - mlpack::CLI, 1124
 - mlpack::ann::Add, 557, 558

- mlpack::ann::AddMerge, 564
- mlpack::ann::AtrousConvolution, 579
- mlpack::ann::BRNN, 616
- mlpack::ann::BatchNorm, 598
- mlpack::ann::Concat, 629
- mlpack::ann::Concatenate, 633, 634
- mlpack::ann::Convolution, 650
- mlpack::ann::DropConnect, 671, 672
- mlpack::ann::FFN, 701
- mlpack::ann::FastLSTM, 691
- mlpack::ann::FlexibleReLU, 712
- mlpack::ann::GRU, 736
- mlpack::ann::LSTM, 806
- mlpack::ann::LayerNorm, 768
- mlpack::ann::Linear, 781
- mlpack::ann::LinearNoBias, 786, 787
- mlpack::ann::Lookup, 799
- mlpack::ann::MultiplyMerge, 833
- mlpack::ann::PReLU, 860
- mlpack::ann::RBM, 872, 873
- mlpack::ann::RNN, 917
- mlpack::ann::Recurrent, 891, 892
- mlpack::ann::RecurrentAttention, 897, 898
- mlpack::ann::Sequential, 933
- mlpack::ann::TransposedConvolution, 966
- mlpack::distribution::RegressionDistribution, 1261
- mlpack::regression::LinearRegression, 1793
- mlpack::regression::LogisticRegression, 1801
- mlpack::regression::SoftmaxRegression, 1817
- mlpack::svm::LinearSVM, 1965
- ParametersSetVisitor, 853
 - mlpack::ann::ParametersSetVisitor, 853
- ParametersVisitor, 854
 - mlpack::ann::ParametersVisitor, 855
- Parent
 - mlpack::tree::BinarySpaceTree, 2015, 2016
 - mlpack::tree::CosineTree, 2038
 - mlpack::tree::CoverTree, 2057
 - mlpack::tree::ExampleTree, 2087
 - mlpack::tree::Octree, 2181
 - mlpack::tree::RectangleTree, 2228
 - mlpack::tree::SpillTree, 2290
- ParentDistance
 - mlpack::tree::BinarySpaceTree, 2016
 - mlpack::tree::CoverTree, 2057, 2058
 - mlpack::tree::ExampleTree, 2087
 - mlpack::tree::Octree, 2182
 - mlpack::tree::RectangleTree, 2229
 - mlpack::tree::SpillTree, 2291
- ParentOf
 - mlpack::det::PathCacher, 1225
- ParseCommandLine
 - mlpack::bindings::cli, 302
- PartialGradient
 - mlpack::regression::LogisticRegressionFunction, 1809
 - mlpack::regression::SoftmaxRegressionFunction, 1824
- Partitioner
 - mlpack::kmeans::KMeans, 1488
- path
 - mlpack::det::PathCacher, 1225
- pathCache
 - mlpack::det::PathCacher, 1225
- PathCacheType
 - mlpack::det::PathCacher, 1223
- PathCacher, 1221
 - mlpack::det::PathCacher, 1223
- PathFor
 - mlpack::det::PathCacher, 1225
- PathFormat
 - mlpack::det::PathCacher, 1223
- PathType
 - mlpack::det::PathCacher, 1223
- PearsonSearch, 1075
 - mlpack::cf::PearsonSearch, 1076
- PellegMooreKMeans
 - mlpack::kmeans::PellegMooreKMeans, 1494
- PellegMooreKMeans< MetricType, MatType >, 1493
- PellegMooreKMeansRules
 - mlpack::kmeans::PellegMooreKMeansRules, 1496
- PellegMooreKMeansRules< MetricType, TreeType >, 1495
- PellegMooreKMeansStatistic, 1498
 - mlpack::kmeans::PellegMooreKMeansStatistic, 1499
- Pendulum, 1878
 - mlpack::rl::Pendulum, 1878
- Pendulum::Action, 1881
- Pendulum::State, 1882
- Percentage
 - mlpack::kmeans::RefinedStart, 1504, 1505
- Perceptron
 - mlpack::perceptron::Perceptron, 1736, 1737
- Perceptron< LearnPolicy, WeightInitializationPolicy, Mat↔Type >, 1735
- PerformAction
 - mlpack::hmm::HMMModel, 1353
- PerformSplit
 - mlpack::tree::MeanSplit, 2144
 - mlpack::tree::MidpointSplit, 2149
 - mlpack::tree::RPTreeMaxSplit, 2257, 2258
 - mlpack::tree::RPTreeMeanSplit, 2263
 - mlpack::tree::VantagePointSplit, 2333, 2334
 - mlpack::tree::split, 465
- persistent
 - mlpack::util::ParamData, 2359
- Phase
 - mlpack::ann::RBM, 873

- PickLeafSplit
 - mlpack::tree::RStarTreeSplit, 2268
- Point
 - mlpack::tree::BinarySpaceTree, 2016
 - mlpack::tree::CoverTree, 2058
 - mlpack::tree::ExampleTree, 2088
 - mlpack::tree::Octree, 2182
 - mlpack::tree::RectangleTree, 2229
 - mlpack::tree::SpillTree, 2291
- PointToAddress
 - mlpack::bound::addr, 367
- Policy
 - mlpack::data::DatasetMapper, 1175, 1176
 - mlpack::rl::AsyncLearning, 1840, 1841
- PolynomialKernel, 1451
 - mlpack::kernel::PolynomialKernel, 1452
- PoolSize
 - mlpack::ann::RBM, 874
- Pooling
 - mlpack::ann::MaxPoolingRule, 814
 - mlpack::ann::MeanPoolingRule, 822
- Position
 - mlpack::rl::CartPole::State, 1849
 - mlpack::rl::ContinuousMountainCar::State, 1857
 - mlpack::rl::MountainCar::State, 1867
- PositiveDefiniteConstraint, 1334
- Power
 - mlpack::metric::LMetric, 1572
- PreCalulated
 - mlpack::lmnn::Constraints, 1524, 1525
- PreFixedArg< T >, 1381
- PreFixedArg< T & >, 1382
- Precision< AS, PositiveClass >, 1145
- Predict
 - mlpack::ann::BRNN, 616
 - mlpack::ann::FFN, 701
 - mlpack::ann::RNN, 917
 - mlpack::cf::CFModel, 1044
 - mlpack::cf::CFTYPE, 1050
 - mlpack::distribution::RegressionDistribution, 1261
 - mlpack::hmm::HMMRegression, 1359
 - mlpack::hmm::HMM, 1346
 - mlpack::regression::LARS, 1787
 - mlpack::regression::LinearRegression, 1793
- PredictVisitor
 - mlpack::cf::PredictVisitor, 1078
- PredictVisitor< NeighborSearchPolicy, InterpolationPolicy >, 1077
- PredictionsType
 - mlpack::cv::MetaInfoExtractor, 1141
 - mlpack::cv::NotFoundMethodForm, 1144
 - mlpack::cv::TrainFormBase4, 1164
 - mlpack::cv::TrainFormBase5, 1166
 - mlpack::cv::TrainFormBase6, 1168
 - mlpack::cv::TrainFormBase7, 1169
- Predictors
 - mlpack::ann::BRNN, 617
 - mlpack::ann::FFN, 702
 - mlpack::ann::RNN, 918
 - mlpack::regression::LogisticRegressionFunction, 1810
- PrefixedOutputStream, 2361
 - mlpack::util::PrefixedOutputStream, 2363
- prereqs.hpp
 - _USE_MATH_DEFINES, 3076
 - BOOST_MPL_CFG_NO_PREPROCESSED_HEADERS, 3076
 - BOOST_MPL_LIMIT_LIST_SIZE, 3076
 - BOOST_PFTO, 3076
 - force_inline, 3076
 - M_PI, 3077
 - omp_size_t, 3077
- print_docs.hpp
 - PrintDocs, 2509
 - PrintHeaders, 2509
- PrintClassDefn
 - mlpack::bindings::python, 344, 345
- PrintDataset
 - mlpack::bindings::cli, 302
 - mlpack::bindings::markdown, 322
 - mlpack::bindings::python, 346
- PrintDefault
 - mlpack::bindings::cli, 303
 - mlpack::bindings::markdown, 322
 - mlpack::bindings::python, 346
- PrintDefn
 - mlpack::bindings::python, 346
- PrintDoc
 - mlpack::bindings::python, 346
- PrintDocs
 - print_docs.hpp, 2509
- PrintHeaders
 - print_docs.hpp, 2509
- PrintHelp
 - mlpack::bindings::cli, 303
- PrintImport
 - mlpack::bindings::cli, 303
 - mlpack::bindings::markdown, 322
 - mlpack::bindings::python, 347
- PrintInputOptions
 - mlpack::bindings::python, 347
- PrintInputProcessing
 - mlpack::bindings::python, 347–350
- PrintLanguage
 - mlpack::bindings::markdown, 323
- PrintLeafMembership
 - mlpack::det, 391
- PrintModel

- mlpack::bindings::cli, 303
- mlpack::bindings::markdown, 323
- mlpack::bindings::python, 350
- PrintOutputOptionInfo
 - mlpack::bindings::cli, 304
 - mlpack::bindings::markdown, 323
 - mlpack::bindings::python, 351
- PrintOutputOptions
 - mlpack::bindings::python, 351
- PrintOutputProcessing
 - mlpack::bindings::python, 351–353
- PrintPYX
 - mlpack::bindings::python, 354
- PrintTimer
 - mlpack::Timers, 1980
- PrintType
 - mlpack::bindings::cli, 304
- PrintTypeDoc
 - mlpack::bindings::cli, 304, 305
 - mlpack::bindings::markdown, 323
 - mlpack::bindings::python, 354, 355
- PrintTypeDocs
 - mlpack::bindings::cli, 305
 - mlpack::bindings::markdown, 323
- PrintValue
 - mlpack::bindings::cli, 306
 - mlpack::bindings::markdown, 324
 - mlpack::bindings::python, 355, 356
- PrintVariableImportance
 - mlpack::det, 391
- Probabilities
 - mlpack::distribution::DiscreteDistribution, 1236
 - mlpack::naive_bayes::NaiveBayesClassifier, 1581
- Probability
 - mlpack::ann::BernoulliDistribution, 602, 604
 - mlpack::distribution::DiagonalGaussianDistribution, 1230
 - mlpack::distribution::DiscreteDistribution, 1236, 1237
 - mlpack::distribution::GammaDistribution, 1243, 1244
 - mlpack::distribution::GaussianDistribution, 1249
 - mlpack::distribution::LaplaceDistribution, 1255, 1256
 - mlpack::distribution::RegressionDistribution, 1262
 - mlpack::gmm::DiagonalGMM, 1314
 - mlpack::gmm::GMM, 1329, 1330
- ProcessOptions
 - mlpack::bindings::cli, 306
- ProgramCall
 - mlpack::bindings::cli, 306
 - mlpack::bindings::markdown, 324
 - mlpack::bindings::python, 356
- ProgramDoc, 982, 988, 2368
 - mlpack::bindings::cli::ProgramDoc, 983
 - mlpack::bindings::tests::ProgramDoc, 989
 - mlpack::util::ProgramDoc, 2369, 2370
- ProgramDocWrapper, 986
 - mlpack::bindings::markdown::ProgramDocWrapper, 987
- ProgramName
 - mlpack::CLI, 1124
- programName
 - mlpack::CLI, 1126
 - mlpack::bindings::cli::ProgramDoc, 983
 - mlpack::bindings::python, 357
 - mlpack::bindings::tests, 364
 - mlpack::bindings::tests::ProgramDoc, 989
 - mlpack::util::ProgramDoc, 2370
- ProjVector, 2190
 - mlpack::tree::ProjVector, 2191
- ProjVectorType
 - mlpack::tree::HyperplaneBase, 2135
- Project
 - mlpack::tree::AxisParallelProjVector, 1988
 - mlpack::tree::HyperplaneBase, 2136
 - mlpack::tree::ProjVector, 2191, 2192
- ProjectDictionary
 - mlpack::sparse_coding::SparseCoding, 1924
- Projections
 - mlpack::neighbor::LSHSearch, 1614, 1615
- PruneAndUpdate
 - mlpack::det::DTree, 1218
- Pruned
 - mlpack::kmeans::DualTreeKMeansStatistic, 1475
- PyOption
 - mlpack::bindings::python::PyOption, 988
- PyOption< T >, 987
- Q
 - mlpack::cf::BiasSVDPolicy, 1041
 - mlpack::cf::SVDPlusPlusPolicy, 1107
- QDAFN< MatType >, 1666
- QDAFN
 - mlpack::neighbor::QDAFN, 1666, 1667
- QLearning
 - mlpack::rl::QLearning, 1887
- QLearning< EnvironmentType, NetworkType, Updater←Type, PolicyType, ReplayType >, 1885
- QUIC_SVD, 1932
 - mlpack::svd::QUIC_SVD, 1932
- QUICSVDPolicy, 1727
 - mlpack::pca::QUICSVDPolicy, 1728
- queryDepth
 - mlpack::tree::QueueFrame, 2193
- queryNode
 - mlpack::tree::QueueFrame, 2193
- querySet
 - mlpack::neighbor::NeighborSearchRules, 1651
- QueueFrame< TreeType, TraversalInfoType >, 2193
- QueueFrameType

- mlpack::tree::BinarySpaceTree::BreadthFirstDual↔
TreeTraverser, 2019
- RAModel
 - mlpack::neighbor::RAModel, 1672
- RAModel< SortPolicy >, 1669
- RAQueryStat
 - mlpack::neighbor::RAQueryStat, 1679
- RAQueryStat< SortPolicy >, 1678
- RASearch
 - mlpack::neighbor::RASearch, 1683–1685
- RASearch< SortPolicy, MetricType, MatType, TreeType >, 1681
- RASearchRules
 - mlpack::neighbor::RASearchRules, 1693
- RASearchRules< SortPolicy, MetricType, TreeType >, 1692
- RAType
 - mlpack::neighbor, 433
- RATypeT
 - mlpack::neighbor::BiSearchVisitor, 1593
 - mlpack::neighbor::TrainVisitor, 1709
- RAUtil, 1699
- RBM< InitializationRuleType, DataType, PolicyType >, 864
- RBM
 - mlpack::ann::RBM, 868
- REQUIRE_RELATIVE_ERR
 - test_tools.hpp, 3088
- RNN< OutputLayerType, InitializationRuleType, Custom↔
Layers >, 911
- RNN
 - mlpack::ann::RNN, 913
- RPTree
 - mlpack::tree, 458
- RPTreeMaxSplit< BoundType, MatType >, 2255
- RPTreeMaxSplit< BoundType, MatType >::SplitInfo, 2260
- RPTreeMeanSplit< BoundType, MatType >, 2261
- RPTreeMeanSplit< BoundType, MatType >::SplitInfo, 2264
- RPlusPlusTree
 - mlpack::tree, 457
- RPlusPlusTreeAuxiliaryInformation
 - mlpack::tree::RPlusPlusTreeAuxiliaryInformation, 2240–2242
- RPlusPlusTreeAuxiliaryInformation< TreeType >, 2239
- RPlusPlusTreeDescentHeuristic, 2246
- RPlusPlusTreeSplitPolicy, 2247
- RPlusTree
 - mlpack::tree, 458
- RPlusTreeDescentHeuristic, 2250
- RPlusTreeSplit< SplitPolicyType, SweepType >, 2251
- RPlusTreeSplitPolicy, 2253
- RSMModel, 1772
- mlpack::range::RSMModel, 1774
- RSType
 - mlpack::range, 439
- RSTypeT
 - mlpack::range::BiSearchVisitor, 1749
 - mlpack::range::TrainVisitor, 1781
- RStarTree
 - mlpack::tree, 459
- RStarTreeDescentHeuristic, 2266
- RStarTreeSplit, 2267
- RTree
 - mlpack::tree, 459
- RTreeDescentHeuristic, 2269
- RTreeSplit, 2271
- Radical, 1744
 - mlpack::radical::Radical, 1745
- Radius
 - mlpack::bound::BallBound, 1000
 - mlpack::meanshift::MeanShift, 1564
- RandBernoulli
 - mlpack::math, 416
- randGen
 - mlpack::math, 425
- RandInt
 - mlpack::math, 417
- RandNormal
 - mlpack::math, 417, 418
- randNormalDist
 - mlpack::math, 425
- randUniformDist
 - mlpack::math, 425
- RandVector
 - mlpack::math, 419
- Random
 - mlpack::distribution::DiagonalGaussianDistribution, 1230
 - mlpack::distribution::DiscreteDistribution, 1237
 - mlpack::distribution::GammaDistribution, 1244
 - mlpack::distribution::GaussianDistribution, 1250
 - mlpack::distribution::LaplaceDistribution, 1256
 - mlpack::gmm::DiagonalGMM, 1314
 - mlpack::gmm::GMM, 1330
 - mlpack::math, 418
- RandomAcolInitialization
 - mlpack::amf::RandomAcolInitialization, 527
- RandomAcolInitialization< columnsToAverage >, 526
- RandomBasis
 - mlpack::math, 419
 - mlpack::neighbor::NSModel, 1663
 - mlpack::neighbor::RAModel, 1675
 - mlpack::range::RSMModel, 1777
- RandomDimensionSelect, 2194
 - mlpack::tree::RandomDimensionSelect, 2195
- RandomForest

- mlpack::tree::RandomForest, 2199–2201
- RandomForest< FitnessFunction, DimensionSelection←
Type, NumericSplitType, CategoricalSplitType,
ElemType >, 2196
- RandomInitialization, 528, 861, 1740
 - mlpack::amf::RandomInitialization, 528
 - mlpack::ann::RandomInitialization, 862, 863
 - mlpack::perceptron::RandomInitialization, 1741
- RandomInitialize
 - InitHMMModel, 481, 482
- RandomInitializer, 1915
- RandomPartition, 1500
 - mlpack::kmeans::RandomPartition, 1501
- RandomPointSelection, 1201
- RandomReplay
 - mlpack::rl::RandomReplay, 1892
- RandomReplay< EnvironmentType >, 1890
- RandomSeed
 - mlpack::math, 419
- RandomSelection, 1457
- RandomizedBlockKrylovSVDPolicy, 1730
 - mlpack::pca::RandomizedBlockKrylovSVDPolicy,
1730
- RandomizedBlockKrylovSVD, 1934
 - mlpack::svd::RandomizedBlockKrylovSVD, 1935
- RandomizedSVDPolicy, 1078, 1732
 - mlpack::cf::RandomizedSVDPolicy, 1080
 - mlpack::pca::RandomizedSVDPolicy, 1733
- RandomizedSVD, 1937
 - mlpack::svd::RandomizedSVD, 1939
- Range
 - mlpack::lmnn::LMNNFunction, 1537
 - mlpack::lmnn::LMNN, 1530
 - mlpack::math, 409
 - mlpack::tree::GiniGain, 2091
 - mlpack::tree::GiniImpurity, 2092
 - mlpack::tree::InformationGain, 2139
- range_search_utils.hpp
 - CheckMatrices, 3083
 - ModelToString, 3083
 - ReadData, 3084
- RangeDistance
 - mlpack::bound::BallBound, 1000, 1001
 - mlpack::bound::HRectBound, 1027
 - mlpack::bound::HollowBallBound, 1017
 - mlpack::tree::BinarySpaceTree, 2017
 - mlpack::tree::CoverTree, 2058, 2059
 - mlpack::tree::ExampleTree, 2088
 - mlpack::tree::Octree, 2182, 2183
 - mlpack::tree::RectangleTree, 2230
 - mlpack::tree::SpillTree, 2292
- RangeSearch
 - mlpack::range::RangeSearch, 1756–1758
- RangeSearch< MetricType, MatType, TreeType >, 1754
- RangeSearchRules
 - mlpack::range::RangeSearchRules, 1765
- RangeSearchRules< MetricType, TreeType >, 1764
- RangeSearchStat, 1768
 - mlpack::range::RangeSearchStat, 1769
- RangeType
 - mlpack::math::RangeType, 1551
- RangeType< T >, 1549
- Rank
 - mlpack::cf::CFTYPE, 1051
 - mlpack::svd::BiasSVDFunction, 1931
 - mlpack::svd::RegularizedSVDFunction, 1950
 - mlpack::svd::SVDPlusPlusFunction, 1958
- Ratio
 - mlpack::ann::AlphaDropout, 571
 - mlpack::ann::DropConnect, 672
 - mlpack::ann::Dropout, 676, 677
 - mlpack::det::DTree, 1219
- ReLULayer
 - mlpack::ann, 270
- ReadData
 - range_search_utils.hpp, 3084
- RearrangesDataset
 - mlpack::tree::TreeTraits, 2304
 - mlpack::tree::TreeTraits< BinarySpaceTree<
MetricType, StatisticType, MatType, bound←
::BallBound, SplitType > >, 2306
 - mlpack::tree::TreeTraits< BinarySpaceTree<
MetricType, StatisticType, MatType, bound←
::CellBound, SplitType > >, 2308
 - mlpack::tree::TreeTraits< BinarySpaceTree<
MetricType, StatisticType, MatType, bound←
::HollowBallBound, SplitType > >, 2310
 - mlpack::tree::TreeTraits< BinarySpaceTree<
MetricType, StatisticType, MatType, Bound←
Type, RPTreeMaxSplit > >, 2313
 - mlpack::tree::TreeTraits< BinarySpaceTree<
MetricType, StatisticType, MatType, Bound←
Type, RPTreeMeanSplit > >, 2315
 - mlpack::tree::TreeTraits< BinarySpaceTree<
MetricType, StatisticType, MatType, Bound←
Type, SplitType > >, 2318
 - mlpack::tree::TreeTraits< CoverTree< MetricType,
StatisticType, MatType, RootPointPolicy > >,
2320
 - mlpack::tree::TreeTraits< Octree< MetricType,
StatisticType, MatType > >, 2323
 - mlpack::tree::TreeTraits< RectangleTree< Metric←
Type, StatisticType, MatType, RPlusTreeSplit<
SplitPolicyType, SweepType >, DescentType,
AuxiliaryInformationType > >, 2325
 - mlpack::tree::TreeTraits< RectangleTree< Metric←
Type, StatisticType, MatType, SplitType,
DescentType, AuxiliaryInformationType > >,
2325

- 2328
- mlpack::tree::TreeTraits< SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > >, 2330
- Recall
 - mlpack::neighbor::NeighborSearch, 1635
- Recall< AS, PositiveClass >, 1147
- RecommendationVisitor
 - mlpack::cf::RecommendationVisitor, 1084
- RecommendationVisitor< NeighborSearchPolicy, InterpolationPolicy >, 1084
- ReconstructionLoss
 - mlpack::ann::ReconstructionLoss, 881
- ReconstructionLoss< InputDataType, OutputDataType, DistType >, 880
- Recover
 - mlpack::matrix_completion::MatrixCompletion, 1560
- RectangleTree
 - mlpack::tree::RectangleTree, 2213–2215
- RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >, 2207
- RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::DualTreeTraverser< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >, 2234
- RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType >::SingleTreeTraverser< RuleType >, 2237
- RectifierFunction, 883
- Recurrent
 - mlpack::ann::Recurrent, 887, 888
- Recurrent< InputDataType, OutputDataType, CustomLayers >, 886
- RecurrentAttention
 - mlpack::ann::RecurrentAttention, 894
- RecurrentAttention< InputDataType, OutputDataType >, 892
- referenceNode
 - mlpack::tree::QueueFrame, 2193
- ReferenceSet
 - mlpack::neighbor::LSHSearch, 1615
 - mlpack::neighbor::NeighborSearch, 1635
 - mlpack::neighbor::RASearch, 1687
 - mlpack::range::RangeSearch, 1759
- referenceSet
 - mlpack::neighbor::NeighborSearchRules, 1651
- ReferenceSetVisitor, 1701, 1771
- ReferenceTree
 - mlpack::kde::KDE, 1395
 - mlpack::neighbor::NeighborSearch, 1635, 1636
 - mlpack::range::RangeSearch, 1760
- RefinedStart, 1502
 - mlpack::kmeans::RefinedStart, 1503
- RegSVDPolicy, 1088
 - mlpack::cf::RegSVDPolicy, 1089
- RegisterProgramDoc
 - mlpack::CLI, 1124
 - mlpack::bindings::markdown::BindingInfo, 985
- RegressionDistribution, 1257
 - mlpack::distribution::RegressionDistribution, 1259
- RegressionInterpolation, 1085
 - mlpack::cf::RegressionInterpolation, 1086
- Regularization
 - mlpack::lmnn::LMNNFunction, 1537
 - mlpack::lmnn::LMNN, 1530
- regularization
 - det.txt, 2378
- RegularizedSVD< OptimizerType >, 1943
- RegularizedSVDFunction
 - mlpack::svd::RegularizedSVDFunction, 1946
- RegularizedSVDFunction< MatType >, 1945
- RegularizedSVD
 - mlpack::svd::RegularizedSVD, 1944
- ReinforceNormal
 - mlpack::ann::ReinforceNormal, 900
- ReinforceNormal< InputDataType, OutputDataType >, 898
- ReinsertPoints
 - mlpack::tree::RStarTreeSplit, 2268
- RelativeDelta
 - mlpack::hpt::HyperParameterTuner, 1379
- RelativeError
 - mlpack::kde::KDEModel, 1403
 - mlpack::kde::KDE, 1395
- Relax
 - mlpack::neighbor::FurthestNS, 1608
 - mlpack::neighbor::NearestNS, 1626
- RemoveNode
 - mlpack::tree::RectangleTree, 2230
- RemoveRows
 - mlpack::math, 420
- Reparametrization
 - mlpack::ann::Reparametrization, 904
- Reparametrization< InputDataType, OutputDataType >, 903
- Replicates
 - mlpack::radical::Radical, 1747
- ReportIgnoredParam
 - mlpack::util, 470
- RequireAtLeastOnePassed
 - mlpack::util, 471
- RequireNoneOrAllPassed
 - mlpack::util, 471
- RequireOnlyOnePassed
 - mlpack::util, 472

- RequireParamInSet
 - mlpack::util, 473
- RequireParamValue
 - mlpack::util, 473
- required
 - mlpack::util::ParamData, 2360
- Rescore
 - mlpack::emst::DTBRules, 1266
 - mlpack::fastmks::FastMKSRules, 1301
 - mlpack::kde::KDERules, 1407
 - mlpack::kmeans::DualTreeKMeansRules, 1471
 - mlpack::kmeans::PellegMooreKMeansRules, 1497
 - mlpack::neighbor::NeighborSearchRules, 1646, 1647
 - mlpack::neighbor::RASearchRules, 1695, 1696
 - mlpack::range::RangeSearchRules, 1766, 1767
- Reset
 - mlpack::Timers, 1980
 - mlpack::ann::AtrousConvolution, 580
 - mlpack::ann::BRNN, 617
 - mlpack::ann::BatchNorm, 598
 - mlpack::ann::Convolution, 651
 - mlpack::ann::FastLSTM, 691
 - mlpack::ann::FlexibleReLU, 712
 - mlpack::ann::LSTM, 806
 - mlpack::ann::LayerNorm, 768
 - mlpack::ann::Linear, 781
 - mlpack::ann::LinearNoBias, 787
 - mlpack::ann::PReLU, 860
 - mlpack::ann::RBM, 874
 - mlpack::ann::RNN, 918
 - mlpack::ann::TransposedConvolution, 966
 - mlpack::neighbor::NeighborSearchStat, 1656
- ResetAll
 - mlpack::Timer, 1976
- ResetCell
 - mlpack::ann::FastLSTM, 691
 - mlpack::ann::GRU, 736
 - mlpack::ann::LSTM, 807
- ResetCellVisitor, 907
 - mlpack::ann::ResetCellVisitor, 908
- ResetFunction
 - ann_test_tools.hpp, 3078, 3079
- ResetParameters
 - mlpack::ann::BRNN, 618
 - mlpack::ann::FFN, 702
 - mlpack::ann::RNN, 918
- ResetQueryTree
 - mlpack::neighbor::RASearch, 1687
- ResetTimers
 - mlpack::util, 474
- ResetVisitor, 909
- Residual
 - mlpack::ann, 270
- residue
 - mlpack::amf::SimpleResidueTermination, 534
- Responses
 - mlpack::ann::BRNN, 618
 - mlpack::ann::FFN, 702, 703
 - mlpack::ann::RNN, 918, 919
 - mlpack::regression::LogisticRegressionFunction, 1810
- RestoreChildren
 - mlpack::kmeans, 405
- RestoreSettings
 - mlpack::CLI, 1125
- RevertLabels
 - mlpack::data, 385
- Reward
 - mlpack::ann::ReinforceNormal, 902
- RewardClipping
 - mlpack::rl::RewardClipping, 1896
- RewardClipping< EnvironmentType >, 1894
- RewardSetVisitor, 910
 - mlpack::ann::RewardSetVisitor, 910
- Rf
 - mlpack::distribution::RegressionDistribution, 1262
- Rho
 - mlpack::ann::BRNN, 618, 619
 - mlpack::ann::FastLSTM, 691, 692
 - mlpack::ann::GRU, 736, 737
 - mlpack::ann::LSTM, 807
 - mlpack::ann::RNN, 919
 - mlpack::neighbor::NSModel, 1663
 - mlpack::nn::SparseAutoencoder, 1715, 1716
 - mlpack::nn::SparseAutoencoderFunction, 1721
- Right
 - mlpack::det::DTree, 1219
 - mlpack::tree::BinarySpaceTree, 2017
 - mlpack::tree::CosineTree, 2039
 - mlpack::tree::HyperplaneBase, 2137
 - mlpack::tree::SpillTree, 2292
- Rk4
 - mlpack::rl::Acrobot, 1829
- Root
 - mlpack::det::DTree, 1219
- Rows
 - mlpack::math::ColumnsToBlocks, 1547, 1548
- Run
 - mlpack::ann::AddMerge, 564, 565
 - mlpack::ann::Concat, 629
- RunSetVisitor, 921
 - mlpack::ann::RunSetVisitor, 922
- SEE_ALSO
 - param.hpp, 2734
- SELU
 - mlpack::ann, 270
- SINGLE_ARG

- sfinae_utility.hpp, 2745
- SPTree
 - mlpack::tree, 460
- SVDBatchFactorizer
 - mlpack::amf, 259
- SVDBatchLearning, 539
 - mlpack::amf::SVDBatchLearning, 540
- SVDBatchLearning::HUpdate< arma::sp_mat >
 - mlpack::amf, 261
- SVDBatchLearning::WUpdate< arma::sp_mat >
 - mlpack::amf, 261
- SVDCompleteIncrementalFactorizer
 - mlpack::amf, 260
- SVDCompleteIncrementalLearning
 - mlpack::amf::SVDCompleteIncrementalLearning, 543
 - mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >, 545
- SVDCompleteIncrementalLearning< arma::sp_mat >, 544
- SVDCompleteIncrementalLearning< MatType >, 542
- SVDCompletePolicy, 1094
- SVDConvolution< BorderMode >, 950
- SVDIncompleteIncrementalFactorizer
 - mlpack::amf, 260
- SVDIncompleteIncrementalLearning, 547
 - mlpack::amf::SVDIncompleteIncrementalLearning, 548
- SVDIncompleteIncrementalLearning::HUpdate< arma::sp_mat >
 - mlpack::amf, 261
- SVDIncompleteIncrementalLearning::WUpdate< arma::sp_mat >
 - mlpack::amf, 261
- SVDIncompletePolicy, 1098
- SVDPlusPlus
 - mlpack::svd::SVDPlusPlus, 1952
- SVDPlusPlus< OptimizerType >, 1951
- SVDPlusPlusFunction
 - mlpack::svd::SVDPlusPlusFunction, 1955
- SVDPlusPlusFunction< MatType >, 1954
- SVDPlusPlusPolicy, 1102
 - mlpack::cf::SVDPlusPlusPolicy, 1103
- SVDWrapper
 - mlpack::cf::SVDWrapper, 1109
- SVDWrapper< Factorizer >, 1108
- sameSet
 - mlpack::neighbor::NeighborSearchRules, 1651
- Sample
 - mlpack::ann::BernoulliDistribution, 604
 - mlpack::rl::Acrobot, 1829, 1830
 - mlpack::rl::AggregatedPolicy, 1837
 - mlpack::rl::CartPole, 1845
 - mlpack::rl::ContinuousMountainCar, 1853
 - mlpack::rl::GreedyPolicy, 1860
 - mlpack::rl::MountainCar, 1864
 - mlpack::rl::Pendulum, 1879, 1880
 - mlpack::rl::RandomReplay, 1892
 - mlpack::rl::RewardClipping, 1899, 1900
- SampleAtLeaves
 - mlpack::neighbor::RAModel, 1675
 - mlpack::neighbor::RASearch, 1688
- SampleAtLeavesVisitor, 1702
- SampleHidden
 - mlpack::ann::RBM, 874
- SampleInitialization, 1506
 - mlpack::kmeans::SampleInitialization, 1506
- SampleSlab
 - mlpack::ann::RBM, 875
- SampleSpike
 - mlpack::ann::RBM, 875
- SampleVisible
 - mlpack::ann::RBM, 875, 876
- Samplings
 - mlpack::kmeans::RefinedStart, 1505
- Save
 - mlpack::data, 386, 387
- SaveHMM
 - mlpack::hmm, 397
- SaveOutputParameterVisitor, 923
 - mlpack::ann::SaveOutputParameterVisitor, 923
- Scale
 - mlpack::distribution::LaplaceDistribution, 1256, 1257
 - mlpack::kernel::HyperbolicTangentKernel, 1432
 - mlpack::math::ColumnsToBlocks, 1548
 - mlpack::tree::CoverTree, 2059
- ScaleData
 - mlpack::pca::PCA, 1726, 1727
- Score
 - mlpack::emst::DTBRules, 1267
 - mlpack::fastmks::FastMKSRules, 1301, 1302
 - mlpack::kde::KDERules, 1407
 - mlpack::kmeans::DualTreeKMeansRules, 1471, 1472
 - mlpack::kmeans::PellegMooreKMeansRules, 1498
 - mlpack::neighbor::NeighborSearchRules, 1647, 1648
 - mlpack::neighbor::RASearchRules, 1696–1698
 - mlpack::range::RangeSearchRules, 1767
- score
 - mlpack::tree::QueueFrame, 2193
- Scores
 - mlpack::emst::DTBRules, 1268
 - mlpack::fastmks::FastMKSRules, 1302
 - mlpack::kde::KDERules, 1407
 - mlpack::kmeans::DualTreeKMeansRules, 1472
 - mlpack::neighbor::NeighborSearch, 1636
 - mlpack::neighbor::NeighborSearchRules, 1648
 - mlpack::range::RangeSearch, 1760
 - mlpack::range::RangeSearchRules, 1768

- scores
 - mlpack::neighbor::NeighborSearchRules, 1651
- Sdp
 - mlpack::matrix_completion::MatrixCompletion, 1560
- Search
 - mlpack::cf::CosineSearch, 1058
 - mlpack::cf::LMetricSearch, 1065
 - mlpack::cf::PearsonSearch, 1076
 - mlpack::fastmks::FastMKModel, 1296, 1297
 - mlpack::fastmks::FastMKS, 1287, 1288
 - mlpack::neighbor::DrusillaSelect, 1599
 - mlpack::neighbor::LSHSearch, 1615, 1616
 - mlpack::neighbor::NSModel, 1663, 1664
 - mlpack::neighbor::NeighborSearch, 1636, 1637
 - mlpack::neighbor::QDAFN, 1668
 - mlpack::neighbor::RAModel, 1676
 - mlpack::neighbor::RASearch, 1688, 1689
 - mlpack::range::RModel, 1777, 1778
 - mlpack::range::RangeSearch, 1760–1762
- SearchMode
 - mlpack::neighbor::NSModel, 1664
 - mlpack::neighbor::NeighborSearch, 1638
- SearchModeVisitor, 1703
- SecondBound
 - mlpack::neighbor::NeighborSearchStat, 1656
- SecondHashTable
 - mlpack::neighbor::LSHSearch, 1616
- SecondHashWeights
 - mlpack::neighbor::LSHSearch, 1616
- seeAlso
 - mlpack::util::ProgramDoc, 2370
- Select
 - mlpack::ann::Select, 925
 - mlpack::dbscan::OrderedPointSelection, 1201
 - mlpack::dbscan::RandomPointSelection, 1202
 - mlpack::kernel::KMeansSelection, 1442
 - mlpack::kernel::OrderedSelection, 1451
 - mlpack::kernel::RandomSelection, 1457
- Select< InputDataType, OutputDataType >, 924
- SelectMethodForm< MLAlgorithm >, 1149
- SelectMethodForm< MLAlgorithm >::From< Forms >, 1149
- SelectMethodForm< MLAlgorithm, HMFs >, 1148
- SelectMethodForm< MLAlgorithm, HasMethodForm, HMFs... >, 1150
- SelectMethodForm< MLAlgorithm, HasMethodForm, HMFs... >::From< Forms >, 1151
- SelfKernel
 - mlpack::fastmks::FastMKStat, 1306
- SequencePrecision
 - mlpack::ann::augmented::scorers, 274
- Sequential
 - mlpack::ann::Sequential, 929
- Sequential< InputDataType, OutputDataType, Residual, CustomLayers >, 927
- serialization_template_version.hpp
 - BOOST_TEMPLATE_CLASS_VERSION, 2565
- serialize
 - mlpack::adaboost::AdaBoost, 492
 - mlpack::adaboost::AdaBoostModel, 497
 - mlpack::amf::AverageInitialization, 504
 - mlpack::amf::GivenInitialization, 510
 - mlpack::amf::NMFALSUpdate, 518
 - mlpack::amf::NMFMultiplicativeDistanceUpdate, 522
 - mlpack::amf::NMFMultiplicativeDivergenceUpdate, 525
 - mlpack::amf::RandomAcolInitialization, 527
 - mlpack::amf::RandomInitialization, 529
 - mlpack::amf::SVDBatchLearning, 541
 - mlpack::ann::Add, 558
 - mlpack::ann::AddMerge, 565
 - mlpack::ann::AlphaDropout, 572
 - mlpack::ann::AtrousConvolution, 580
 - mlpack::ann::BRNN, 619
 - mlpack::ann::BaseLayer, 592
 - mlpack::ann::BatchNorm, 598
 - mlpack::ann::BernoulliDistribution, 604
 - mlpack::ann::BilinearInterpolation, 609
 - mlpack::ann::CReLU, 655
 - mlpack::ann::Concat, 630
 - mlpack::ann::ConcatPerformance, 637
 - mlpack::ann::Concatenate, 634
 - mlpack::ann::Constant, 640
 - mlpack::ann::Convolution, 651
 - mlpack::ann::CrossEntropyError, 659
 - mlpack::ann::DiceLoss, 665
 - mlpack::ann::DropConnect, 672
 - mlpack::ann::Dropout, 677
 - mlpack::ann::ELU, 685
 - mlpack::ann::EarthMoverDistance, 680
 - mlpack::ann::FFN, 703
 - mlpack::ann::FastLSTM, 692
 - mlpack::ann::FlexibleReLU, 713
 - mlpack::ann::GRU, 737
 - mlpack::ann::Glimpse, 722
 - mlpack::ann::HardTanH, 744
 - mlpack::ann::Join, 756
 - mlpack::ann::KLDivergence, 761
 - mlpack::ann::LSTM, 807
 - mlpack::ann::LayerNorm, 768
 - mlpack::ann::LeakyReLU, 774
 - mlpack::ann::Linear, 782
 - mlpack::ann::LinearNoBias, 787
 - mlpack::ann::LogSoftMax, 795
 - mlpack::ann::Lookup, 800
 - mlpack::ann::MaxPooling, 813
 - mlpack::ann::MeanPooling, 821

- mlpack::ann::MeanSquaredError, 825
- mlpack::ann::MultiplyConstant, 828
- mlpack::ann::MultiplyMerge, 834
- mlpack::ann::NegativeLogLikelihood, 840
- mlpack::ann::PReLU, 861
- mlpack::ann::RBM, 876
- mlpack::ann::RNN, 919
- mlpack::ann::ReconstructionLoss, 883
- mlpack::ann::Recurrent, 892
- mlpack::ann::RecurrentAttention, 898
- mlpack::ann::ReinforceNormal, 902
- mlpack::ann::Reparametrization, 907
- mlpack::ann::Select, 927
- mlpack::ann::Sequential, 934
- mlpack::ann::SigmoidCrossEntropyError, 939
- mlpack::ann::Subview, 950
- mlpack::ann::TransposedConvolution, 966
- mlpack::ann::VRCClassReward, 971
- mlpack::bound::BallBound, 1001
- mlpack::bound::HRectBound, 1028
- mlpack::bound::HollowBallBound, 1018
- mlpack::cf::BatchSVDPolicy, 1035
- mlpack::cf::BiasSVDPolicy, 1041
- mlpack::cf::CFModel, 1044
- mlpack::cf::CFType, 1051
- mlpack::cf::CombinedNormalization, 1056
- mlpack::cf::ItemMeanNormalization, 1063
- mlpack::cf::NMFPolicy, 1069
- mlpack::cf::NoNormalization, 1071
- mlpack::cf::OverallMeanNormalization, 1074
- mlpack::cf::RandomizedSVDPolicy, 1083
- mlpack::cf::RegSVDPolicy, 1091
- mlpack::cf::SVDCompletePolicy, 1097
- mlpack::cf::SVDIncompletePolicy, 1101
- mlpack::cf::SVDPlusPlusPolicy, 1107
- mlpack::cf::UserMeanNormalization, 1113
- mlpack::cf::ZScoreNormalization, 1116
- mlpack::data::DatasetMapper, 1176
- mlpack::decision_stump::DecisionStump, 1205
- mlpack::det::DTree, 1219
- mlpack::distribution::DiagonalGaussianDistribution, 1231
- mlpack::distribution::DiscreteDistribution, 1237
- mlpack::distribution::GaussianDistribution, 1250
- mlpack::distribution::LaplaceDistribution, 1257
- mlpack::distribution::RegressionDistribution, 1262
- mlpack::fastmks::FastMKModel, 1297
- mlpack::fastmks::FastMKStat, 1306
- mlpack::fastmks::FastMKS, 1289
- mlpack::gmm::DiagonalConstraint, 1307
- mlpack::gmm::DiagonalGMM, 1314
- mlpack::gmm::EMFit, 1322
- mlpack::gmm::EigenvalueRatioConstraint, 1318
- mlpack::gmm::GMM, 1330
- mlpack::gmm::NoConstraint, 1333
- mlpack::gmm::PositiveDefiniteConstraint, 1335
- mlpack::hmm::HMMModel, 1353
- mlpack::hmm::HMM, 1347
- mlpack::kde::KDEModel, 1403
- mlpack::kde::KDEStat, 1409
- mlpack::kde::KDE, 1395
- mlpack::kernel::CauchyKernel, 1415
- mlpack::kernel::CosineDistance, 1417
- mlpack::kernel::EpanechnikovKernel, 1420
- mlpack::kernel::ExampleKernel, 1424
- mlpack::kernel::GaussianKernel, 1429
- mlpack::kernel::HyperbolicTangentKernel, 1433
- mlpack::kernel::LaplacianKernel, 1446
- mlpack::kernel::LinearKernel, 1448
- mlpack::kernel::PolynomialKernel, 1454
- mlpack::kernel::SphericalKernel, 1461
- mlpack::kernel::TriangularKernel, 1464
- mlpack::kmeans::AllowEmptyClusters, 1466
- mlpack::kmeans::KMeans, 1489
- mlpack::kmeans::KillEmptyClusters, 1483
- mlpack::kmeans::MaxVarianceNewCluster, 1491
- mlpack::kmeans::RandomPartition, 1502
- mlpack::kmeans::RefinedStart, 1505
- mlpack::lcc::LocalCoordinateCoding, 1518
- mlpack::math::RangeType, 1557
- mlpack::metric::IPMetric, 1568
- mlpack::metric::LMetric, 1571
- mlpack::metric::MahalanobisDistance, 1576
- mlpack::naive_bayes::NaiveBayesClassifier, 1582
- mlpack::neighbor::DrusillaSelect, 1599
- mlpack::neighbor::LSHSearch, 1617
- mlpack::neighbor::NSModel, 1664
- mlpack::neighbor::NeighborSearch, 1638
- mlpack::neighbor::NeighborSearchStat, 1656
- mlpack::neighbor::QDAFN, 1668
- mlpack::neighbor::RAModel, 1676
- mlpack::neighbor::RAQueryStat, 1680
- mlpack::neighbor::RASearch, 1690
- mlpack::perceptron::Perceptron, 1739
- mlpack::range::RSMModel, 1778
- mlpack::range::RangeSearch, 1762
- mlpack::range::RangeSearchStat, 1770
- mlpack::regression::LARS, 1788
- mlpack::regression::LinearRegression, 1794
- mlpack::regression::LogisticRegression, 1802
- mlpack::regression::SoftmaxRegression, 1817
- mlpack::sparse_coding::SparseCoding, 1924
- mlpack::svm::LinearSVM, 1966
- mlpack::tree::AxisParallelProjVector, 1989
- mlpack::tree::BinaryNumericSplit, 1995
- mlpack::tree::BinaryNumericSplitInfo, 1997
- mlpack::tree::BinarySpaceTree, 2018
- mlpack::tree::CategoricalSplitInfo, 2029

- mlpack::tree::CoverTree, 2059
- mlpack::tree::DecisionTree, 2076
- mlpack::tree::EmptyStatistic, 2081
- mlpack::tree::HilbertRTreeAuxiliaryInformation, 2100
- mlpack::tree::HoeffdingCategoricalSplit, 2107
- mlpack::tree::HoeffdingNumericSplit, 2112
- mlpack::tree::HoeffdingTree, 2124
- mlpack::tree::HoeffdingTreeModel, 2132
- mlpack::tree::HyperplaneBase, 2137
- mlpack::tree::NoAuxiliaryInformation, 2164
- mlpack::tree::NumericSplitInfo, 2166
- mlpack::tree::Octree, 2183
- mlpack::tree::ProjVector, 2192
- mlpack::tree::RPlusPlusTreeAuxiliaryInformation, 2245
- mlpack::tree::RandomForest, 2203
- mlpack::tree::RectangleTree, 2231
- mlpack::tree::SpillTree, 2292
- mlpack::tree::XTreeAuxiliaryInformation, 2344
- mlpack::tree::XTreeAuxiliaryInformation::Split←HistoryStruct, 2346
- SerializeIn
 - mlpack::bindings::python, 356
- SerializeObject
 - mlpack, 255
- SerializeObjectAll
 - mlpack, 255
- SerializeOut
 - mlpack::bindings::python, 356
- SerializePointerObject
 - mlpack, 256
- SerializePointerObjectAll
 - mlpack, 256
- set
 - bindings/CMakeLists.txt, 2385
 - bindings/cli/CMakeLists.txt, 2384
 - bindings/markdown/CMakeLists.txt, 2386
 - bindings/python/CMakeLists.txt, 2387
 - bindings/tests/CMakeLists.txt, 2388
 - core/CMakeLists.txt, 2389
 - core/cv/CMakeLists.txt, 2390
 - core/cv/metrics/CMakeLists.txt, 2391
 - core/data/CMakeLists.txt, 2391, 2392
 - core/data/imputation_methods/CMakeLists.txt, 2392
 - core/data/map_policies/CMakeLists.txt, 2393
 - core/dists/CMakeLists.txt, 2394
 - core/hpt/CMakeLists.txt, 2394
 - core/kernels/CMakeLists.txt, 2395
 - core/math/CMakeLists.txt, 2396
 - core/metrics/CMakeLists.txt, 2397
 - core/tree/CMakeLists.txt, 2398, 2399
 - core/util/CMakeLists.txt, 2400
 - methods/CMakeLists.txt, 2418
 - methods/adaboost/CMakeLists.txt, 2400
 - methods/amf/CMakeLists.txt, 2401
 - methods/amf/init_rules/CMakeLists.txt, 2402
 - methods/amf/termination_policies/CMakeLists.txt, 2403
 - methods/amf/update_rules/CMakeLists.txt, 2403
 - methods/ann/CMakeLists.txt, 2406
 - methods/ann/activation_functions/CMakeLists.txt, 2404
 - methods/ann/augmented/CMakeLists.txt, 2405
 - methods/ann/augmented/tasks/CMakeLists.txt, 2405
 - methods/ann/convolution_rules/CMakeLists.txt, 2407
 - methods/ann/dists/CMakeLists.txt, 2407
 - methods/ann/init_rules/CMakeLists.txt, 2408
 - methods/ann/layer/CMakeLists.txt, 2409
 - methods/ann/loss_functions/CMakeLists.txt, 2410
 - methods/ann/rbm/CMakeLists.txt, 2411
 - methods/ann/visitor/CMakeLists.txt, 2412
 - methods/approx_kfn/CMakeLists.txt, 2413
 - methods/bias_svd/CMakeLists.txt, 2413
 - methods/block_krylov_svd/CMakeLists.txt, 2414
 - methods/cf/CMakeLists.txt, 2415
 - methods/cf/decomposition_policies/CMakeLists.txt, 2415
 - methods/cf/interpolation_policies/CMakeLists.txt, 2416
 - methods/cf/neighbor_search_policies/CMakeLists.txt, 2417
 - methods/cf/normalization/CMakeLists.txt, 2417
 - methods/dbscan/CMakeLists.txt, 2419
 - methods/decision_stump/CMakeLists.txt, 2420
 - methods/decision_tree/CMakeLists.txt, 2420, 2421
 - methods/det/CMakeLists.txt, 2421
 - methods/emst/CMakeLists.txt, 2422
 - methods/fastmks/CMakeLists.txt, 2423
 - methods/gmm/CMakeLists.txt, 2424
 - methods/hmm/CMakeLists.txt, 2424
 - methods/hoeffding_trees/CMakeLists.txt, 2425
 - methods/kde/CMakeLists.txt, 2426
 - methods/kernel_pca/CMakeLists.txt, 2427
 - methods/kernel_pca/kernel_rules/CMakeLists.txt, 2427
 - methods/kmeans/CMakeLists.txt, 2428
 - methods/lars/CMakeLists.txt, 2429
 - methods/linear_regression/CMakeLists.txt, 2429, 2430
 - methods/linear_svm/CMakeLists.txt, 2430
 - methods/lmnn/CMakeLists.txt, 2431
 - methods/local_coordinate_coding/CMakeLists.txt, 2432
 - methods/logistic_regression/CMakeLists.txt, 2433
 - methods/lsh/CMakeLists.txt, 2433
 - methods/matrix_completion/CMakeLists.txt, 2434
 - methods/mean_shift/CMakeLists.txt, 2435
 - methods/naive_bayes/CMakeLists.txt, 2435

- methods/nca/CMakeLists.txt, 2436
- methods/neighbor_search/CMakeLists.txt, 2437
- methods/nystroem_method/CMakeLists.txt, 2438
- methods/pca/CMakeLists.txt, 2439
- methods/pca/decomposition_policies/CMakeLists.txt, 2440
- methods/perceptron/CMakeLists.txt, 2440
- methods/perceptron/initialization_methods/CMakeLists.txt, 2441
- methods/perceptron/learning_policies/CMakeLists.txt, 2442
- methods/preprocess/CMakeLists.txt, 2442
- methods/quic_svd/CMakeLists.txt, 2443
- methods/radical/CMakeLists.txt, 2444
- methods/random_forest/CMakeLists.txt, 2444
- methods/randomized_svd/CMakeLists.txt, 2445
- methods/range_search/CMakeLists.txt, 2446
- methods/rann/CMakeLists.txt, 2447
- methods/regularized_svd/CMakeLists.txt, 2447
- methods/reinforcement_learning/CMakeLists.txt, 2448
- methods/reinforcement_learning/environment/CMakeLists.txt, 2449
- methods/reinforcement_learning/policy/CMakeLists.txt, 2449, 2450
- methods/reinforcement_learning/replay/CMakeLists.txt, 2450
- methods/reinforcement_learning/worker/CMakeLists.txt, 2451
- methods/softmax_regression/CMakeLists.txt, 2452
- methods/sparse_autoencoder/CMakeLists.txt, 2452
- methods/sparse_coding/CMakeLists.txt, 2453
- methods/svdplusplus/CMakeLists.txt, 2454
- SetCentroid
 - mlpack::kde::KDEStat, 1410
- SetDimensionality
 - mlpack::data::DatasetMapper, 1176
- SetInputHeightVisitor, 934
 - mlpack::ann::SetInputHeightVisitor, 935
- SetInputParam
 - mlpack::util, 474
- SetInputWidthVisitor, 935
 - mlpack::ann::SetInputWidthVisitor, 936
- SetMemState
 - arma_util.hpp, 2519
- SetParam
 - mlpack::bindings::cli, 307, 308
 - mlpack::util, 475
- SetParamPtr
 - mlpack::util, 475
- SetParamWithInfo
 - mlpack::util, 475
- SetPassed
 - mlpack::CLI, 1125
- sfinae_utility.hpp
 - HAS_ANY_METHOD_FORM, 2742
 - HAS_EXACT_METHOD_FORM, 2743
 - HAS_MEM_FUNC, 2744
 - HAS_METHOD_FORM_BASE, 2745
 - HAS_METHOD_FORM, 2744
 - SINGLE_ARG, 2745
- shortDocumentation
 - mlpack::util::ProgramDoc, 2370
- ShrinkBoundForBound
 - mlpack::tree::RectangleTree, 2231
- ShrinkBoundForPoint
 - mlpack::tree::RectangleTree, 2231
- Shuffle
 - mlpack::ann::BRNN, 619
 - mlpack::ann::FFN, 703
 - mlpack::ann::RBM, 876
 - mlpack::ann::RNN, 919
 - mlpack::cv::KFoldCV, 1139
 - mlpack::lmnn::LMNNFunction, 1538
 - mlpack::nca::SoftmaxErrorFunction, 1590
 - mlpack::regression::LogisticRegressionFunction, 1810
 - mlpack::regression::SoftmaxRegressionFunction, 1825
 - mlpack::svd::BiasSVDFunction, 1931
 - mlpack::svd::RegularizedSVDFunction, 1950
 - mlpack::svd::SVDPlusPlusFunction, 1959
 - mlpack::svm::LinearSVMFunction, 1974
- ShuffleData
 - mlpack::math, 420–422
- SigCheck< U, U >, 1912
- Sigmoid
 - mlpack::nn::SparseAutoencoder, 1716
 - mlpack::nn::SparseAutoencoderFunction, 1721
- SigmoidCrossEntropyError
 - mlpack::ann::SigmoidCrossEntropyError, 938
- SigmoidCrossEntropyError< InputDataType, OutputDataType >, 937
- SigmoidLayer
 - mlpack::ann, 271
- Sign
 - mlpack::math, 422
- SimilarityInterpolation, 1092
 - mlpack::cf::SimilarityInterpolation, 1093
- SimpleCV< MLAlgorithm, Metric, MatType, PredictionsType, WeightsType >, 1151
- SimpleCV
 - mlpack::cv::SimpleCV, 1153–1156
- SimpleResidueTermination, 529
 - mlpack::amf::SimpleResidueTermination, 530
- SimpleToleranceTermination
 - mlpack::amf::SimpleToleranceTermination, 535
- SimpleToleranceTermination< MatType >, 535

- SimpleWeightUpdate, 1741
- SingleMode
 - mlpack::fastmks::FastMKModel, 1297
 - mlpack::fastmks::FastMKS, 1289
 - mlpack::neighbor::RAModel, 1676
 - mlpack::neighbor::RASearch, 1690
 - mlpack::range::RModel, 1778
 - mlpack::range::RangeSearch, 1762, 1763
- SingleModeVisitor, 1704, 1779
- SingleSampleLimit
 - mlpack::neighbor::RAModel, 1677
 - mlpack::neighbor::RASearch, 1690
- SingleSampleLimitVisitor, 1705
- SingleTreeTraverser
 - mlpack::tree::BinarySpaceTree::SingleTreeTraverser, 2026
 - mlpack::tree::CoverTree::SingleTreeTraverser, 2063
 - mlpack::tree::Octree::SingleTreeTraverser, 2188
 - mlpack::tree::RectangleTree::SingleTreeTraverser, 2238
 - mlpack::tree::SpillTree, 2279
- Size
 - mlpack::rl::RandomReplay, 1893
- size
 - mlpack::rl::ContinuousMountainCar::Action, 1855
 - mlpack::rl::Pendulum::Action, 1881
- SlabMean
 - mlpack::ann::RBM, 876
- SlabPenalty
 - mlpack::ann::RBM, 877
- Smat
 - mlpack::math, 423
- Smooth
 - mlpack::ann::DiceLoss, 665
 - mlpack::hmm::HMMRegression, 1360
 - mlpack::hmm::HMM, 1347
- SoftDelete
 - mlpack::tree::RectangleTree, 2232
- SoftPlusLayer
 - mlpack::ann, 271
- SoftmaxErrorFunction
 - mlpack::nca::SoftmaxErrorFunction, 1587
- SoftmaxErrorFunction< MetricType >, 1586
- SoftmaxRegression, 1811
 - mlpack::regression::SoftmaxRegression, 1812, 1813
- SoftmaxRegressionFunction, 1818
 - mlpack::regression::SoftmaxRegressionFunction, 1819
- SoftplusFunction, 940
- SoftsignFunction, 943
- SortTask, 584
 - mlpack::ann::augmented::tasks::SortTask, 585
- SpaceSplit< MetricType, MatType >, 2272
- SparseAutoencoder, 1711
 - mlpack::nn::SparseAutoencoder, 1713
- SparseAutoencoderFunction, 1717
 - mlpack::nn::SparseAutoencoderFunction, 1718
- SparseCoding, 1916
 - mlpack::sparse_coding::SparseCoding, 1918, 1919
- SphericalKernel, 1458
 - mlpack::kernel::SphericalKernel, 1459
- SpikeBias
 - mlpack::ann::RBM, 877
- SpikeMean
 - mlpack::ann::RBM, 877
- SpikeSlabRBM, 947
- SpillDualTreeTraverser
 - mlpack::tree::SpillTree::SpillDualTreeTraverser, 2294
- SpillKNN
 - mlpack::neighbor, 433
- SpillSingleTreeTraverser
 - mlpack::tree::SpillTree::SpillSingleTreeTraverser, 2298
- SpillTree
 - mlpack::tree::SpillTree, 2279–2281, 2283
- SpillTree< MetricType, StatisticType, MatType, Hyperplane↔Type, SplitType >, 2274
- SpillTree< MetricType, StatisticType, MatType, Hyperplane↔Type, SplitType >::SpillDualTreeTraverser< MetricType, StatisticType, MatType, Hyperplane↔Type, SplitType >, 2293
- SpillTree< MetricType, StatisticType, MatType, Hyperplane↔Type, SplitType >::SpillSingleTreeTraverser< MetricType, StatisticType, MatType, Hyperplane↔Type, SplitType >, 2297
- Split
 - mlpack::data, 387–389
 - mlpack::decision_stump::DecisionStump, 1206
 - mlpack::tree::BinaryNumericSplit, 1995
 - mlpack::tree::BinarySpaceTree, 2002
 - mlpack::tree::HoeffdingCategoricalSplit, 2107
 - mlpack::tree::HoeffdingNumericSplit, 2112
- SplitAuxiliaryInfo
 - mlpack::tree::NoAuxiliaryInformation, 2164
 - mlpack::tree::RPlusPlusTreeAuxiliaryInformation, 2245
- SplitCheck
 - mlpack::tree::HoeffdingTree, 2124
- SplitDim
 - mlpack::det::DTree, 1220
- SplitDimension
 - mlpack::decision_stump::DecisionStump, 1206
 - mlpack::tree::DecisionTree, 2076
 - mlpack::tree::HoeffdingTree, 2124
- splitDimension
 - mlpack::tree::MeanSplit::SplitInfo, 2146
 - mlpack::tree::MidpointSplit::SplitInfo, 2151
- SplitHistory

- mlpack::tree::XTreeAuxiliaryInformation, 2344
- SplitHistoryStruct
 - mlpack::tree::XTreeAuxiliaryInformation, 2339
 - mlpack::tree::XTreeAuxiliaryInformation::Split↔HistoryStruct, 2345, 2346
- SplitIfBetter
 - mlpack::tree::AllCategoricalSplit, 1983
 - mlpack::tree::BestBinaryNumericSplit, 1991
- SplitInfo
 - mlpack::tree::BinaryNumericSplit, 1993
 - mlpack::tree::HoeffdingCategoricalSplit, 2105
 - mlpack::tree::HoeffdingNumericSplit, 2110
 - mlpack::tree::Octree::SplitType::SplitInfo, 2189
 - mlpack::tree::VantagePointSplit::SplitInfo, 2337
- SplitLeafNode
 - mlpack::tree::HilbertRTreeSplit, 2103
 - mlpack::tree::RPlusTreeSplit, 2252
 - mlpack::tree::RStarTreeSplit, 2268
 - mlpack::tree::RTreeSplit, 2271
 - mlpack::tree::XTreeSplit, 2348
- SplitNode
 - mlpack::tree::MeanSplit, 2145
 - mlpack::tree::MidpointSplit, 2150
 - mlpack::tree::RPTreeMaxSplit, 2258
 - mlpack::tree::RPTreeMeanSplit, 2264
 - mlpack::tree::VantagePointSplit, 2334
- SplitNonLeafNode
 - mlpack::tree::HilbertRTreeSplit, 2103
 - mlpack::tree::RPlusTreeSplit, 2253
 - mlpack::tree::RStarTreeSplit, 2269
 - mlpack::tree::RTreeSplit, 2271
 - mlpack::tree::XTreeSplit, 2348
- SplitPointIndex
 - mlpack::tree::CosineTree, 2039
- SplitPolicy
 - mlpack::tree::RPlusTreeSplit, 2252
- SplitRequired
 - mlpack::tree::RPlusPlusTreeSplitPolicy, 2249
 - mlpack::tree::RPlusTreeSplitPolicy, 2255
- SplitSpace
 - mlpack::tree::MeanSpaceSplit, 2142
 - mlpack::tree::MidpointSpaceSplit, 2147
- SplitType
 - mlpack::tree::RectangleTree, 2233
- splitVal
 - mlpack::tree::MeanSplit::SplitInfo, 2146
 - mlpack::tree::MidpointSplit::SplitInfo, 2151
 - mlpack::tree::RPTreeMaxSplit::SplitInfo, 2260
 - mlpack::tree::RPTreeMeanSplit::SplitInfo, 2266
- SplitValue
 - mlpack::det::DTree, 1220
- SquaredEuclideanDistance
 - mlpack::metric, 427
- src/mlpack/core/util/version.hpp
- MLPACK_VERSION_MAJOR, 2747
- MLPACK_VERSION_MINOR, 2747
- MLPACK_VERSION_PATCH, 2747
- StandardCoverTree
 - mlpack::tree, 460
- Start
 - mlpack::Timer, 1976
 - mlpack::det::DTree, 1220
- StartTimer
 - mlpack::Timers, 1980
- Stat
 - mlpack::tree::BinarySpaceTree, 2018
 - mlpack::tree::CoverTree, 2060
 - mlpack::tree::ExampleTree, 2089
 - mlpack::tree::Octree, 2183
 - mlpack::tree::RectangleTree, 2232
 - mlpack::tree::SpillTree, 2293
- StatType
 - mlpack::det::DTree, 1210
- State
 - mlpack::rl::Acrobot::State, 1832
 - mlpack::rl::CartPole::State, 1847
 - mlpack::rl::ContinuousMountainCar::State, 1856
 - mlpack::rl::MountainCar::State, 1866
 - mlpack::rl::Pendulum::State, 1883
 - mlpack::rl::QLearning, 1889
 - mlpack::rl::RewardClipping, 1896
- StateType
 - mlpack::rl::NStepQLearningWorker, 1869
 - mlpack::rl::OneStepQLearningWorker, 1872
 - mlpack::rl::OneStepSarsaWorker, 1876
 - mlpack::rl::QLearning, 1887
 - mlpack::rl::RandomReplay, 1892
- StaticLowerBoundMovement
 - mlpack::kmeans::DualTreeKMeansStatistic, 1476
- StaticPruned
 - mlpack::kmeans::DualTreeKMeansStatistic, 1476
- StaticSerialize
 - mlpack::data::HasSerializeFunction, 1181
- StaticUpperBoundMovement
 - mlpack::kmeans::DualTreeKMeansStatistic, 1476
- std, 476
 - enable_if_t, 476
- Stddev
 - mlpack::cf::ZScoreNormalization, 1117
- Step
 - mlpack::rl::NStepQLearningWorker, 1870
 - mlpack::rl::OneStepQLearningWorker, 1874
 - mlpack::rl::OneStepSarsaWorker, 1877
 - mlpack::rl::QLearning, 1890
- StepLimit
 - mlpack::rl::TrainingConfig, 1904, 1905
- StepSize
 - mlpack::rl::TrainingConfig, 1905

- Stop
 - mlpack::Timer, 1977
- StopAllTimers
 - mlpack::Timers, 1981
- StopTimer
 - mlpack::Timers, 1981
- Store
 - mlpack::rl::RandomReplay, 1893
- StoreSettings
 - mlpack::CLI, 1125
- Strategy
 - mlpack::data::Imputer, 1184
- StringTypeParam
 - mlpack::bindings::cli, 309
- StringTypeParam< bool >
 - mlpack::bindings::cli, 309
- StringTypeParam< double >
 - mlpack::bindings::cli, 309
- StringTypeParam< int >
 - mlpack::bindings::cli, 309
- StringTypeParam< std::string >
 - mlpack::bindings::cli, 310
- StringTypeParam< std::tuple< mlpack::data::DatasetInfo, arma::mat > >
 - mlpack::bindings::cli, 310
- StringTypeParamImpl
 - mlpack::bindings::cli, 310, 311
- StripType
 - mlpack::bindings::python, 357
- SubtreeLeaves
 - mlpack::det::DTree, 1220
- SubtreeLeavesLogNegError
 - mlpack::det::DTree, 1220
- Subview
 - mlpack::ann::Subview, 948
- Subview< InputDataType, OutputDataType >, 947
- SuccessProbability
 - mlpack::neighbor::RAUtil, 1700
 - mlpack::tree::HoeffdingTree, 2125
- SupportsWeights
 - mlpack::cv::MetaInfoExtractor, 1142
- Svec
 - mlpack::math, 423
- SvecIndex
 - mlpack::math, 424
- SweepLeafNode
 - mlpack::tree::MinimalCoverageSweep, 2153
 - mlpack::tree::MinimalSplitsNumberSweep, 2156
- SweepNonLeafNode
 - mlpack::tree::MinimalCoverageSweep, 2154
 - mlpack::tree::MinimalSplitsNumberSweep, 2156
- Sweeps
 - mlpack::radical::Radical, 1747
- SwishFunction, 953
- SymKronId
 - mlpack::math, 424
- TPolicy
 - mlpack::amf::CompleteIncrementalTermination, 507
 - mlpack::amf::IncompleteIncrementalTermination, 513
- TUPLE_TYPE
 - param.hpp, 2735
- TYPENAME
 - param_data.hpp, 2738
- TagTree
 - mlpack::det::DTree, 1221
- TakeMean
 - mlpack::ann::KLDivergence, 761
- TakeRoot
 - mlpack::metric::LMetric, 1572
- TakesDatasetInfo
 - mlpack::cv::MetaInfoExtractor, 1142
- TakesNumClasses
 - mlpack::cv::MetaInfoExtractor, 1142
- TanHLayer
 - mlpack::ann, 271
- TanhFunction, 955
- TargetNeighbors
 - mlpack::lmnn::Constraints, 1525
- TargetNetworkSyncInterval
 - mlpack::rl::TrainingConfig, 1905, 1906
- Tau
 - mlpack::neighbor::NSModel, 1664, 1665
 - mlpack::neighbor::RAModel, 1677
 - mlpack::neighbor::RASearch, 1691
- TauVisitor, 1706
- TerminationPolicy
 - mlpack::amf::AMF, 501
- test_function_tools.hpp
 - LogisticRegressionTestData, 3086
- test_tools.hpp
 - CheckMatrices, 3088, 3089
 - CheckMatricesNotEqual, 3089, 3090
 - FilterFileName, 3090
 - REQUIRE_RELATIVE_ERR, 3088
- TestAllArmadilloSerialization
 - mlpack, 256
- TestArmadilloSerialization
 - mlpack, 257
- TestOption
 - mlpack::bindings::tests::TestOption, 990
- TestOption< N >, 990
- tests/CMakeLists.txt
 - add_executable, 2455
- Theta
 - mlpack::rl::Pendulum::State, 1884
- Theta1
 - mlpack::rl::Acrobot::State, 1834

- Theta2
 - mlpack::rl::Acrobot::State, 1834, 1835
- Thus
 - det.txt, 2378
- Timer, 1975
- timer
 - mlpack::CLI, 1127
- Timers, 1977
 - mlpack::Timers, 1978
- tname
 - mlpack::util::ParamData, 2360
- ToString
 - mlpack::Backtrace, 975
- Tolerance
 - mlpack::adaboost::AdaBoost, 492
 - mlpack::amf::SimpleToleranceTermination, 538
 - mlpack::amf::ValidationRMSETermination, 553
 - mlpack::gmm::EMFit, 1323
 - mlpack::hmm::HMM, 1347
 - mlpack::lcc::LocalCoordinateCoding, 1519
- Torque
 - mlpack::rl::Acrobot, 1830
- TotalSteps
 - mlpack::rl::QLearning, 1890
- Train
 - mlpack::adaboost::AdaBoost, 492
 - mlpack::adaboost::AdaBoostModel, 498
 - mlpack::ann::BRNN, 619, 620
 - mlpack::ann::FFN, 703, 704
 - mlpack::ann::RBM, 878
 - mlpack::ann::RNN, 920
 - mlpack::cf::CFModel, 1044
 - mlpack::cf::CFTYPE, 1052
 - mlpack::cv::CVBase, 1131, 1132
 - mlpack::decision_stump::DecisionStump, 1206, 1207
 - mlpack::distribution::DiagonalGaussianDistribution, 1231
 - mlpack::distribution::DiscreteDistribution, 1237, 1238
 - mlpack::distribution::GammaDistribution, 1244, 1245
 - mlpack::distribution::GaussianDistribution, 1250, 1251
 - mlpack::distribution::RegressionDistribution, 1263
 - mlpack::fastmks::FastMKS, 1289–1291
 - mlpack::gmm::DiagonalGMM, 1315
 - mlpack::gmm::GMM, 1330, 1331
 - mlpack::hmm::HMMRegression, 1360, 1361
 - mlpack::hmm::HMM, 1348
 - mlpack::kde::KDE, 1396
 - mlpack::lcc::LocalCoordinateCoding, 1519
 - mlpack::naive_bayes::NaiveBayesClassifier, 1582
 - mlpack::neighbor::DrusillaSelect, 1600
 - mlpack::neighbor::LSHSearch, 1617
 - mlpack::neighbor::NeighborSearch, 1638, 1640
 - mlpack::neighbor::QDAFN, 1668
 - mlpack::neighbor::RASearch, 1691, 1692
 - mlpack::perceptron::Perceptron, 1739
 - mlpack::range::RangeSearch, 1763
 - mlpack::regression::LARS, 1788
 - mlpack::regression::LinearRegression, 1794
 - mlpack::regression::LogisticRegression, 1802, 1803
 - mlpack::regression::SoftmaxRegression, 1817
 - mlpack::rl::AsyncLearning, 1841
 - mlpack::sparse_coding::SparseCoding, 1924
 - mlpack::svm::LinearSVM, 1966
 - mlpack::tree::BinaryNumericSplit, 1996
 - mlpack::tree::DecisionTree, 2076–2078
 - mlpack::tree::HoeffdingCategoricalSplit, 2108
 - mlpack::tree::HoeffdingNumericSplit, 2113
 - mlpack::tree::HoeffdingTree, 2125, 2126
 - mlpack::tree::HoeffdingTreeModel, 2133
 - mlpack::tree::RandomForest, 2203–2206
- TrainForm< MT, PT, void, false, false >, 1158
- TrainForm< MT, PT, void, false, true >, 1159
- TrainForm< MT, PT, void, true, false >, 1160
- TrainForm< MT, PT, void, true, true >, 1160
- TrainForm< MT, PT, WT, false, false >, 1161
- TrainForm< MT, PT, WT, false, true >, 1162
- TrainForm< MT, PT, WT, true, false >, 1162
- TrainForm< MT, PT, WT, true, true >, 1163
- TrainForm< MatType, PredictionsType, WeightsType, DatasetInfo, NumClasses >, 1157
- TrainFormBase4< PT, WT, T1, T2 >, 1164
- TrainFormBase5< PT, WT, T1, T2, T3 >, 1165
- TrainFormBase6< PT, WT, T1, T2, T3, T4 >, 1167
- TrainFormBase7< PT, WT, T1, T2, T3, T4, T5 >, 1169
- TrainHMMModel, 2371
 - Apply, 2371
- TrainVisitor, 1413, 1780
 - mlpack::kde::TrainVisitor, 1413
 - mlpack::neighbor::TrainVisitor, 1709
 - mlpack::range::TrainVisitor, 1781
- TrainVisitor< SortPolicy >, 1707
- Trainer
 - mlpack::det, 392
- TrainingConfig, 1901
 - mlpack::rl::TrainingConfig, 1902
- TrainingMean
 - mlpack::ann::BatchNorm, 599
- TrainingVariance
 - mlpack::ann::BatchNorm, 599
- Transform
 - mlpack::math::ColumnsToBlocks, 1548
- Transition
 - mlpack::hmm::HMM, 1349
- transition
 - mlpack::hmm::HMM, 1350
- TransitionType
 - mlpack::rl::NStepQLearningWorker, 1869

- mlpack::rl::OneStepQLearningWorker, 1873
- mlpack::rl::OneStepSarsaWorker, 1876
- TransposedConvolution
 - mlpack::ann::TransposedConvolution, 961
- TransposedConvolution< ForwardConvolutionRule, BackwardConvolutionRule, GradientConvolutionRule, InputDataType, OutputDataType >, 959
- TraversallInfo
 - mlpack::emst::DTBRules, 1268
 - mlpack::fastmks::FastMKSRules, 1302, 1303
 - mlpack::kde::KDERules, 1408
 - mlpack::kmeans::DualTreeKMeansRules, 1472
 - mlpack::neighbor::NeighborSearchRules, 1648, 1649
 - mlpack::neighbor::RASearchRules, 1698
 - mlpack::range::RangeSearchRules, 1768
 - mlpack::tree::TraversallInfo, 2300
- traversallInfo
 - mlpack::neighbor::NeighborSearchRules, 1651
 - mlpack::tree::QueueFrame, 2194
- TraversallInfo< TreeType >, 2299
- TraversallInfoType
 - mlpack::emst::DTBRules, 1265
 - mlpack::fastmks::FastMKSRules, 1299
 - mlpack::kde::KDERules, 1405
 - mlpack::kmeans::DualTreeKMeansRules, 1470
 - mlpack::neighbor::NeighborSearchRules, 1643
 - mlpack::neighbor::RASearchRules, 1693
 - mlpack::range::RangeSearchRules, 1765
- Traverse
 - mlpack::tree::BinarySpaceTree::BreadthFirstDualTreeTraverser, 2022
 - mlpack::tree::BinarySpaceTree::DualTreeTraverser, 2025
 - mlpack::tree::BinarySpaceTree::SingleTreeTraverser, 2027
 - mlpack::tree::CoverTree::DualTreeTraverser, 2062
 - mlpack::tree::CoverTree::SingleTreeTraverser, 2064
 - mlpack::tree::GreedySingleTreeTraverser, 2094
 - mlpack::tree::Octree::DualTreeTraverser, 2187
 - mlpack::tree::Octree::SingleTreeTraverser, 2188
 - mlpack::tree::RectangleTree::DualTreeTraverser, 2236
 - mlpack::tree::RectangleTree::SingleTreeTraverser, 2238
 - mlpack::tree::SpillTree::SpillDualTreeTraverser, 2296
 - mlpack::tree::SpillTree::SpillSingleTreeTraverser, 2298
- Tree
 - mlpack::emst::DualTreeBoruvka, 1273
 - mlpack::fastmks::FastMKS, 1283
 - mlpack::kde::KDE, 1389
 - mlpack::kmeans::DualTreeKMeans, 1467
 - mlpack::neighbor::NeighborSearch, 1630
 - mlpack::neighbor::RASearch, 1683
 - mlpack::range::RangeSearch, 1756
 - mlpack::tree::RandomForest, 2206, 2207
 - TreeDepth
 - mlpack::tree::RectangleTree, 2232
 - TreeName
 - mlpack::neighbor::NSModel, 1665
 - mlpack::neighbor::RAModel, 1677
 - TreeSize
 - mlpack::tree::RectangleTree, 2232
 - TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::BallBound, SplitType > >, 2305
 - TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::CellBound, SplitType > >, 2307
 - TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound::HollowBallBound, SplitType > >, 2309
 - TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, RPTreeMaxSplit > >, 2311
 - TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, RPTreeMeanSplit > >, 2313
 - TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, BoundType, SplitType > >, 2316
 - TreeTraits< CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > >, 2318
 - TreeTraits< Octree< MetricType, StatisticType, MatType > >, 2321
 - TreeTraits< RectangleTree< MetricType, StatisticType, MatType, RPlusTreeSplit< SplitPolicyType, SweepType >, DescentType, AuxiliaryInformationType > >, 2323
 - TreeTraits< RectangleTree< MetricType, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType > >, 2326
 - TreeTraits< SpillTree< MetricType, StatisticType, MatType, HyperplaneType, SplitType > >, 2328
 - TreeTraits< TreeType >, 2302
 - TreeType
 - mlpack::kde::KDEModel, 1404
 - mlpack::kmeans::PellegMooreKMeans, 1494
 - mlpack::neighbor::NSModel, 1665
 - mlpack::neighbor::RAModel, 1677, 1678
 - mlpack::range::RSMModel, 1778, 1779
 - mlpack::tree::HoeffdingTreeModel, 2128
 - TreeTypes
 - mlpack::kde::KDEModel, 1399
 - mlpack::neighbor::NSModel, 1659
 - mlpack::neighbor::RAModel, 1671
 - mlpack::range::RSMModel, 1773
 - TriangularKernel, 1461
 - mlpack::kernel::TriangularKernel, 1462

- Triplets
 - mlpack::lmnn::Constraints, 1526
- TrueChild
 - mlpack::kmeans::DualTreeKMeansStatistic, 1477
- TrueParent
 - mlpack::kmeans::DualTreeKMeansStatistic, 1477
- TupleOfHyperParameters
 - mlpack::hpt, 398
- TupleType
 - mlpack::cf::CombinedNormalization, 1054
 - mlpack::hpt::DeduceHyperParameterTypes< Pre↔ FixedArg< T >, Args... >, 1367
 - mlpack::hpt::DeduceHyperParameterTypes< Pre↔ FixedArg< T >, Args... >::ResultHolder, 1368
 - mlpack::hpt::DeduceHyperParameterTypes< T, Args... >, 1369
 - mlpack::hpt::DeduceHyperParameterTypes< T, Args... >::ResultHolder, 1371
 - mlpack::hpt::DeduceHyperParameterTypes::Result↔ Holder, 1366
- Type
 - mlpack::cv::SelectMethodForm< MLAlgorithm >::↔ From, 1150
 - mlpack::cv::SelectMethodForm< MLAlgorithm, HasMethodForm, HMFs... >::From, 1151
 - mlpack::cv::TrainFormBase4, 1165
 - mlpack::cv::TrainFormBase5, 1166
 - mlpack::cv::TrainFormBase6, 1168
 - mlpack::cv::TrainFormBase7, 1169
 - mlpack::data::DatasetMapper, 1177
 - mlpack::hmm::HMMModel, 1353
 - mlpack::hpt::DeduceHyperParameterTypes< T, Args... >::ResultHPTType< ArithmeticType, true >, 1372
 - mlpack::hpt::DeduceHyperParameterTypes< T, Args... >::ResultHPTType< CollectionType, false >, 1373
 - mlpack::hpt::PreFixedArg, 1381
 - mlpack::hpt::PreFixedArg< T & >, 1383
- type
 - mlpack::bindings::cli::ParameterType, 977
 - mlpack::bindings::cli::ParameterType< arma::Col< eT > >, 978
 - mlpack::bindings::cli::ParameterType< arma::Mat< eT > >, 979
 - mlpack::bindings::cli::ParameterType< arma::Row< eT > >, 980
 - mlpack::bindings::cli::ParameterType< std::tuple< mlpack::data::DatasetMapper< PolicyType, std::string >, arma::Mat< eT > > >, 980
 - mlpack::bindings::cli::ParameterTypeDeducer, 981
 - mlpack::bindings::cli::ParameterTypeDeducer< true, T >, 982
 - mlpack::tree::MinimalCoverageSweep::SweepCost, 2155
 - mlpack::tree::MinimalSplitsNumberSweep::Sweep↔ Cost, 2157
- UBTree
 - mlpack::tree, 461
- UBTreeSplit< BoundType, MatType >, 2331
- Union
 - mlpack::emst::UnionFind, 1279
- UnionFind, 1278
 - mlpack::emst::UnionFind, 1278
- UniqueNumDescendants
 - mlpack::tree::TreeTraits, 2305
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound↔ ::BallBound, SplitType > >, 2307
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound↔ ::CellBound, SplitType > >, 2309
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, bound↔ ::HollowBallBound, SplitType > >, 2311
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, Bound↔ Type, RPTreeMaxSplit > >, 2313
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, Bound↔ Type, RPTreeMeanSplit > >, 2315
 - mlpack::tree::TreeTraits< BinarySpaceTree< MetricType, StatisticType, MatType, Bound↔ Type, SplitType > >, 2318
 - mlpack::tree::TreeTraits< CoverTree< MetricType, StatisticType, MatType, RootPointPolicy > >, 2320
 - mlpack::tree::TreeTraits< Octree< MetricType, StatisticType, MatType > >, 2323
 - mlpack::tree::TreeTraits< RectangleTree< Metric↔ Type, StatisticType, MatType, RPlusTreeSplit< SplitPolicyType, SweepType >, DescentType, AuxiliaryInformationType > >, 2325
 - mlpack::tree::TreeTraits< RectangleTree< Metric↔ Type, StatisticType, MatType, SplitType, DescentType, AuxiliaryInformationType > >, 2328
 - mlpack::tree::TreeTraits< SpillTree< MetricType, StatisticType, MatType, HyperplaneType, Split↔ Type > >, 2330
- Unmap
 - mlpack::neighbor, 434, 435
- UnmapString
 - mlpack::data::DatasetMapper, 1177
- UnmapValue
 - mlpack::data::DatasetMapper, 1178
- Unpooling

- mlpack::ann::MeanPoolingRule, 822
- Update
 - mlpack::amf::AMF, 502
- UpdateAuxiliaryInfo
 - mlpack::tree::HilbertRTreeAuxiliaryInformation, 2100
 - mlpack::tree::NoAuxiliaryInformation, 2165
 - mlpack::tree::RPlusPlusTreeAuxiliaryInformation, 2245
 - mlpack::tree::XTreeAuxiliaryInformation, 2344
- UpdateInterval
 - mlpack::rl::TrainingConfig, 1906
- UpdateWeights
 - mlpack::perceptron::SimpleWeightUpdate, 1742
- Updater
 - mlpack::rl::AsyncLearning, 1841, 1842
- UpperBound
 - mlpack::kmeans::DualTreeKMeansStatistic, 1477, 1478
- UseLayer
 - mlpack::ann::InitTraits, 751
 - mlpack::ann::InitTraits< Kathirvalavakumar↔ SubavathiInitialization >, 752
 - mlpack::ann::InitTraits< NguyenWidrowInitialization >, 752
- UserMeanNormalization, 1111
 - mlpack::cf::UserMeanNormalization, 1112
- UsesSquaredDistance
 - mlpack::kernel::KernelTraits, 1434
 - mlpack::kernel::KernelTraits< CosineDistance >, 1436
 - mlpack::kernel::KernelTraits< EpanechnikovKernel >, 1437
 - mlpack::kernel::KernelTraits< GaussianKernel >, 1438
 - mlpack::kernel::KernelTraits< LaplacianKernel >, 1439
 - mlpack::kernel::KernelTraits< SphericalKernel >, 1440
 - mlpack::kernel::KernelTraits< TriangularKernel >, 1441
- VPTree
 - mlpack::tree, 461
- VPTreeSplit
 - mlpack::tree, 462
- VRClassReward
 - mlpack::ann::VRClassReward, 968
- VRClassReward< InputDataType, OutputDataType >, 967
- ValidCentroid
 - mlpack::kde::KDEStat, 1410
- ValidConvolution, 967
- ValidationRMSETermination
 - mlpack::amf::ValidationRMSETermination, 551
- ValidationRMSETermination< MatType >, 549
- Value
 - mlpack::bound::meta::IsLMetric, 1029
 - mlpack::bound::meta::IsLMetric< metric::LMetric< Power, TakeRoot > >, 1030
- value
 - IsVector, 483
 - IsVector< arma::Col< eT > >, 484
 - IsVector< arma::Row< eT > >, 484
 - IsVector< arma::SpCol< eT > >, 485
 - IsVector< arma::SpRow< eT > >, 486
 - IsVector< arma::SpSubview< eT > >, 486
 - IsVector< arma::subview_col< eT > >, 487
 - IsVector< arma::subview_row< eT > >, 488
 - mlpack::data::HasSerialize, 1180
 - mlpack::data::HasSerializeFunction, 1181
 - mlpack::hpt::DeduceHyperParameterTypes< T, Args... >::IsCollectionType, 1370
 - mlpack::hpt::FixedArg, 1374
 - mlpack::hpt::IsPreFixedArg, 1380
 - mlpack::hpt::PreFixedArg, 1382
 - mlpack::hpt::PreFixedArg< T & >, 1383
 - mlpack::tree::IsSpillTree, 2140
 - mlpack::tree::IsSpillTree< tree::SpillTree< Metric↔ Type, StatisticType, MatType, HyperplaneType, SplitType > >, 2141
 - mlpack::util::IsStdVector, 2349
 - mlpack::util::IsStdVector< std::vector< T, A > >, 2350
 - mlpack::util::ParamData, 2360
- vantagePoint
 - mlpack::tree::VantagePointSplit::SplitInfo, 2337
- VantagePointSplit< BoundType, MatType, MaxNum↔ Samples >, 2331
- VantagePointSplit< BoundType, MatType, MaxNum↔ Samples >::SplitInfo, 2336
- Variance
 - mlpack::ann::LayerNorm, 768
- Variances
 - mlpack::naive_bayes::NaiveBayesClassifier, 1583
- Vasicek
 - mlpack::radical::Radical, 1747
- Vec
 - mlpack::bound::BallBound, 994
- VecType
 - mlpack::det::DTree, 1210
- VectorIndices
 - mlpack::tree::CosineTree, 2039
- VectorPower
 - mlpack::math, 424
- Velocity
 - mlpack::rl::CartPole::State, 1849, 1850
 - mlpack::rl::ContinuousMountainCar::State, 1857, 1858

- mlpack::rl::MountainCar::State, 1867
- version< mlpack::adaboost::AdaBoost< WeakLearner< Type, MatType > >, 477
- version< mlpack::ann::BRNN< OutputLayerType, MergeLayerType, MergeOutputType, Initialization< RuleType, CustomLayer... > >, 478
- version< mlpack::ann::FFN< OutputLayerType, Initialization< RuleType, CustomLayer... > >, 478
- version< mlpack::ann::RNN< OutputLayerType, Initialization< RuleType, CustomLayer... > >, 479
- VisibleBias
 - mlpack::ann::RBM, 878
- VisibleMean
 - mlpack::ann::RBM, 878, 879
- VisiblePenalty
 - mlpack::ann::RBM, 879
- VisibleSize
 - mlpack::ann::RBM, 879
 - mlpack::nn::SparseAutoencoder, 1716
 - mlpack::nn::SparseAutoencoderFunction, 1721, 1722
- Volume
 - mlpack::bound::HRectBound, 1028
- W
 - mlpack::cf::BatchSVDPolicy, 1035
 - mlpack::cf::BiasSVDPolicy, 1041
 - mlpack::cf::NMFPolicy, 1069
 - mlpack::cf::RandomizedSVDPolicy, 1083
 - mlpack::cf::RegSVDPolicy, 1092
 - mlpack::cf::SVDCompletePolicy, 1097
 - mlpack::cf::SVDIncompletePolicy, 1101
 - mlpack::cf::SVDPlusPlusPolicy, 1107
- WUpdate
 - mlpack::amf::NMFALSUpdate, 518
 - mlpack::amf::NMFMultiplicativeDistanceUpdate, 522
 - mlpack::amf::NMFMultiplicativeDivergenceUpdate, 525
 - mlpack::amf::SVDBatchLearning, 541
 - mlpack::amf::SVDCompleteIncrementalLearning, 544
 - mlpack::amf::SVDCompleteIncrementalLearning< arma::sp_mat >, 546
 - mlpack::amf::SVDIncompleteIncrementalLearning, 549
- Warn
 - mlpack::Log, 1541
- wasPassed
 - mlpack::util::ParamData, 2361
- WeakLearner
 - mlpack::adaboost::AdaBoost, 493
- WeakLearnerType
 - mlpack::adaboost::AdaBoostModel, 498
- WeakLearnerTypes
 - mlpack::adaboost::AdaBoostModel, 495
 - WeakLearners
 - mlpack::adaboost::AdaBoost, 493
 - mlpack::ann::RBM, 880
 - WeightSetVisitor, 972
 - mlpack::ann::WeightSetVisitor, 972
 - WeightSizeVisitor, 973
 - Weights
 - mlpack::gmm::DiagonalGMM, 1316
 - mlpack::gmm::GMM, 1332
 - mlpack::perceptron::Perceptron, 1740
 - WeightsType
 - mlpack::cv::MetaInfoExtractor, 1141
 - mlpack::cv::NotFoundMethodForm, 1144
 - mlpack::cv::TrainFormBase4, 1165
 - mlpack::cv::TrainFormBase5, 1166
 - mlpack::cv::TrainFormBase6, 1168
 - mlpack::cv::TrainFormBase7, 1170
 - WhitenFeatureMajorMatrix
 - mlpack::radical, 438
 - WhitenUsingEig
 - mlpack::math, 425
 - WhitenUsingSVD
 - mlpack::math, 425
 - Width
 - mlpack::math::RangeType, 1557
 - WithinRange
 - mlpack::det::DTree, 1221
 - WorstDistance
 - mlpack::neighbor::FurthestNS, 1608
 - mlpack::neighbor::NearestNS, 1627
 - Wrap
 - mlpack::rl::Acrobot, 1831
- XTree
 - mlpack::tree, 462
- XTreeAuxiliaryInformation
 - mlpack::tree::XTreeAuxiliaryInformation, 2340, 2341
- XTreeAuxiliaryInformation< TreeType >, 2338
- XTreeAuxiliaryInformation< TreeType >::SplitHistory< Struct, 2345
- XTreeSplit, 2347
- XavierInitialization
 - mlpack::ann, 271
- Y
 - mlpack::cf::SVDPlusPlusPolicy, 1108
- Yes
 - mlpack::hpt::DeduceHyperParameterTypes< T, Args... >::IsCollectionType, 1370
- yes
 - mlpack::data::HasSerialize, 1179
- ZScoreNormalization, 1114

mlpack::cf::ZScoreNormalization, 1115
ZeroInitialization, 1742
mlpack::perceptron::ZeroInitialization, 1743