AMORE++

pre-alpha (active development aiming to release a beta version this summer (2011))

Generated by Doxygen 1.7.4

Fri Jul 22 2011 04:19:41

Contents

1 The AMORE++ package **1**

1.1 Introduction 1

1.2 Motivation 1

1.3 Road Map 1

2 Class Index **3**

2.1 Class Hierarchy 3

3 Class Index **5**

3.1 Class List 5

4 File Index **7**

4.1 File List 7

5 Class Documentation **9**

5.1 ActivationFunction Class Reference 9

5.1.1 Detailed Description 10

5.1.2 Constructor & Destructor Documentation 10

5.1.2.1 ActivationFunction 10

5.1.3 Member Function Documentation 10

5.1.3.1 f0 10

5.1.3.2 f1 10

5.1.3.3 getInducedLocalField 10

5.1.4 Member Data Documentation 11

5.1.4.1 d_neuron 11

5.2 AdaptBehavior Class Reference 11

5.2.1 Detailed Description 13

5.2.2 Member Function Documentation 13
5.2.2.1 adjustParameters 13
5.3 ADAPTgd Class Reference 14
5.3.1 Detailed Description 15
5.3.2 Member Function Documentation 15
5.3.2.1 adjustParameters 16
5.3.3 Member Data Documentation 16
5.3.3.1 outputDerivative 16
5.4 ADAPTgdwm Class Reference 16
5.4.1 Detailed Description 18
5.4.2 Member Function Documentation 18
5.4.2.1 adjustParameters 19
5.4.3 Member Data Documentation 19
5.4.3.1 outputDerivative 19
5.5 ArcTan Class Reference 19
5.5.1 Detailed Description 20
5.5.2 Member Function Documentation 20
5.5.2.1 Arctan 20
5.5.2.2 f0 20
5.5.2.3 f1 21
5.6 ArcTanFactory Class Reference 21
5.6.1 Detailed Description 24
5.6.2 Member Function Documentation 24
5.6.2.1 makeActivationFunction 24
5.7 BatchBehavior Class Reference 24
5.7.1 Detailed Description 26
5.7.2 Member Function Documentation 26
5.7.2.1 adjustParameters 26
5.8 BATCHgd Class Reference 27
5.8.1 Detailed Description 28
5.8.2 Member Function Documentation 28
5.8.2.1 adjustParameters 29
5.8.3 Member Data Documentation 29
5.8.3.1 outputDerivative 29

5.9 BATCHgdwm Class Reference 29
5.9.1 Detailed Description 31
5.9.2 Member Function Documentation 31
5.9.2.1 adjustParameters 32
5.9.3 Member Data Documentation 32
5.9.3.1 outputDerivative 32
5.10 Con Class Reference 32
5.10.1 Detailed Description 33
5.10.2 Constructor & Destructor Documentation 33
5.10.2.1 Con 33
5.10.2.2 Con 33
5.10.3 Member Function Documentation 33
5.10.3.1 getNeuron 33
5.10.3.2 getWeight 34
5.10.3.3 Id 35
5.10.3.4 setNeuron 36
5.10.3.5 setWeight 36
5.10.3.6 show 36
5.10.3.7 validate 37
5.10.4 Member Data Documentation 38
5.10.4.1 d_neuron 38
5.10.4.2 d_weight 38
5.11 Container$<$ T $>$ Class Template Reference 38
5.11.1 Detailed Description 40
5.11.2 Constructor & Destructor Documentation 40
5.11.2.1 ∼Container 40
5.11.2.2 Container 40
5.11.3 Member Function Documentation 40
5.11.3.1 at 40
5.11.3.2 clear 40
5.11.3.3 createIterator 41
5.11.3.4 empty 41
5.11.3.5 push_back 41
5.11.3.6 reserve 41

5.11.3.7 show 41
5.11.3.8 size 41
5.11.3.9 validate 41
5.12 Cosine Class Reference 41
5.12.1 Detailed Description 43
5.12.2 Constructor & Destructor Documentation 43
5.12.2.1 Cosine 43
5.12.3 Member Function Documentation 43
5.12.3.1 f0 44
5.12.3.2 f1 44
5.13 CosineFactory Class Reference 44
5.13.1 Detailed Description 47
5.13.2 Member Function Documentation 47
5.13.2.1 makeActivationFunction 47
5.14 Elliot Class Reference 47
5.14.1 Detailed Description 49
5.14.2 Constructor & Destructor Documentation 49
5.14.2.1 Elliot 49
5.14.3 Member Function Documentation 49
5.14.3.1 f0 50
5.14.3.2 f1 50
5.15 ElliotFactory Class Reference 50
5.15.1 Detailed Description 53
5.15.2 Member Function Documentation 53
5.15.2.1 makeActivationFunction 53
5.16 Exponential Class Reference 53
5.16.1 Detailed Description 55
5.16.2 Constructor & Destructor Documentation 55
5.16.2.1 Exponential 55
5.16.3 Member Function Documentation 55
5.16.3.1 f0 56
5.16.3.2 f1 56
5.17 ExponentialFactory Class Reference 56
5.17.1 Detailed Description 59

5.17.2 Member Function Documentation 59
5.17.2.1 makeActivationFunction 59
5.18 Gauss Class Reference 59
5.18.1 Detailed Description 61
5.18.2 Constructor & Destructor Documentation 61
5.18.2.1 Gauss 61
5.18.3 Member Function Documentation 61
5.18.3.1 f0 62
5.18.3.2 f1 62
5.19 GaussFactory Class Reference 62
5.19.1 Detailed Description 65
5.19.2 Member Function Documentation 65
5.19.2.1 makeActivationFunction 65
5.20 Identity Class Reference 65
5.20.1 Detailed Description 67
5.20.2 Constructor & Destructor Documentation 67
5.20.2.1 Identity 67
5.20.3 Member Function Documentation 68
5.20.3.1 f0 68
5.20.3.2 f1 68
5.21 IdentityFactory Class Reference 68
5.21.1 Detailed Description 71
5.21.2 Member Function Documentation 71
5.21.2.1 makeActivationFunction 71
5.22 Iterator< T > Class Template Reference 71
5.22.1 Detailed Description 72
5.22.2 Constructor & Destructor Documentation 73
5.22.2.1 ∼Iterator 73
5.22.2.2 Iterator 73
5.22.3 Member Function Documentation 73
5.22.3.1 currentItem 73
5.22.3.2 first 73
5.22.3.3 isDone 73
5.22.3.4 next 73

5.23 Logistic Class Reference 74
5.23.1 Detailed Description 75
5.23.2 Constructor & Destructor Documentation 75
5.23.2.1 Logistic 75
5.23.3 Member Function Documentation 75
5.23.3.1 f0 76
5.23.3.2 f1 76
5.24 LogisticFactory Class Reference 76
5.24.1 Detailed Description 79
5.24.2 Member Function Documentation 79
5.24.2.1 makeActivationFunction 79
5.25 MLPbehavior Class Reference 79
5.25.1 Detailed Description 82
5.25.2 Constructor & Destructor Documentation 82
5.25.2.1 MLPbehavior 82
5.25.3 Member Function Documentation 82
5.25.3.1 predict 82
5.25.3.2 show 83
5.25.4 Friends And Related Function Documentation 83
5.25.4.1 MLPfactory 83
5.25.5 Member Data Documentation 83
5.25.5.1 d_bias 83
5.26 MLPfactory Class Reference 84
5.26.1 Detailed Description 86
5.26.2 Constructor & Destructor Documentation 86
5.26.2.1 MLPfactory 86
5.26.3 Member Function Documentation 86
5.26.3.1 makeActivationFunction 86
5.26.3.2 makeCon 86
5.26.3.3 makeConContainer 87
5.26.3.4 makeNeuron 87
5.26.3.5 makeNeuron 88
5.26.3.6 makeNeuronContainer 89
5.26.3.7 makePredictBehavior 89

5.27 NeuralCreator Class Reference 90
5.27.1 Detailed Description 91
5.27.2 Member Function Documentation 91
5.27.2.1 createFeedForwardFullyConnectedNetwork 91
5.28 NeuralFactory Class Reference 91
5.28.1 Detailed Description 92
5.28.2 Member Function Documentation 92
5.28.2.1 makeActivationFunction 92
5.28.2.2 makeCon 92
5.28.2.3 makeConContainer 92
5.28.2.4 makeNeuron 93
5.28.2.5 makeNeuron 93
5.28.2.6 makeNeuronContainer 93
5.28.2.7 makePredictBehavior 93
5.29 NeuralNetwork Class Reference 93
5.29.1 Detailed Description 94
5.29.2 Member Function Documentation 94
5.29.2.1 show 95
5.29.2.2 validate 95
5.29.3 Member Data Documentation 95
5.29.3.1 d_hiddenLayers 95
5.29.3.2 d_inputLayer 95
5.29.3.3 d_outputLayer 95
5.30 Neuron Class Reference 95
5.30.1 Detailed Description 97
5.30.2 Constructor & Destructor Documentation 98
5.30.2.1 Neuron 98
5.30.3 Member Function Documentation 98
5.30.3.1 addCon 98
5.30.3.2 getConIterator 98
5.30.3.3 getId 98
5.30.3.4 getInducedLocalField 98
5.30.3.5 getOutput 98
5.30.3.6 predict 99

5.30.3.7 setActivationFunction 99
5.30.3.8 setId 99
5.30.3.9 setInducedLocalField 99
5.30.3.10 setOutput 99
5.30.3.11 setPredictBehavior 99
5.30.3.12 show 99
5.30.3.13 useActivationFunctionf0 99
5.30.3.14 validate 99
5.30.4 Friends And Related Function Documentation 100
5.30.4.1 MLPfactory 100
5.30.5 Member Data Documentation 100
5.30.5.1 d_activationFunction 100
5.30.5.2 d_Id 100
5.30.5.3 d_inducedLocalField 100
5.30.5.4 d_nCons 100
5.30.5.5 d_output 100
5.30.5.6 d_predictBehavior 100
5.31 PredictBehavior Class Reference 101
5.31.1 Detailed Description 102
5.31.2 Constructor & Destructor Documentation 102
5.31.2.1 PredictBehavior 102
5.31.3 Member Function Documentation 102
5.31.3.1 getConIterator 102
5.31.3.2 predict 103
5.31.3.3 setInducedLocalField 103
5.31.3.4 setOutput 103
5.31.3.5 show 104
5.31.3.6 useActivationFunctionf0 104
5.31.4 Member Data Documentation 105
5.31.4.1 d_neuron 105
5.32 RadialBasis Class Reference 105
5.32.1 Detailed Description 106
5.32.2 Constructor & Destructor Documentation 106
5.32.2.1 RadialBasis 106

5.32.3 Member Function Documentation 106
5.32.3.1 f0 107
5.32.3.2 f1 107
5.33 RadialBasisFactory Class Reference 107
5.33.1 Detailed Description 110
5.33.2 Member Function Documentation 110
5.33.2.1 makeActivationFunction 110
5.34 RBFbehavior Class Reference 110
5.34.1 Detailed Description 113
5.34.2 Constructor & Destructor Documentation 113
5.34.2.1 RBFbehavior 113
5.34.3 Member Function Documentation 113
5.34.3.1 predict 113
5.34.3.2 show 113
5.34.4 Member Data Documentation 113
5.34.4.1 d_altitude 113
5.34.4.2 d_width 113
5.35 RBFfactory Class Reference 113
5.35.1 Detailed Description 116
5.35.2 Constructor & Destructor Documentation 116
5.35.2.1 RBFfactory 116
5.35.3 Member Function Documentation 116
5.35.3.1 makeActivationFunction 116
5.35.3.2 makeCon 116
5.35.3.3 makeConContainer 116
5.35.3.4 makeNeuron 116
5.35.3.5 makeNeuron 116
5.35.3.6 makeNeuronContainer 116
5.35.3.7 makePredictBehavior 117
5.36 Reciprocal Class Reference 117
5.36.1 Detailed Description 118
5.36.2 Constructor & Destructor Documentation 118
5.36.2.1 Reciprocal 118
5.36.3 Member Function Documentation 118

5.36.3.1 f0 119
5.36.3.2 f1 119
5.37 ReciprocalFactory Class Reference 119
5.37.1 Detailed Description 122
5.37.2 Member Function Documentation 122
5.37.2.1 makeActivationFunction 122
5.38 SimpleContainer$<$ T $>$ Class Template Reference 122
5.38.1 Detailed Description 125
5.38.2 Constructor & Destructor Documentation 125
5.38.2.1 SimpleContainer 125
5.38.2.2 ∼SimpleContainer 126
5.38.3 Member Function Documentation 126
5.38.3.1 at 126
5.38.3.2 clear 127
5.38.3.3 createIterator 127
5.38.3.4 empty 127
5.38.3.5 push_back 128
5.38.3.6 reserve 128
5.38.3.7 show 128
5.38.3.8 size 129
5.38.3.9 validate 129
5.38.4 Friends And Related Function Documentation 130
5.38.4.1 SimpleContainerIterator$<$ T $>$ 130
5.38.5 Member Data Documentation 130
5.38.5.1 d_collection 130
5.39 SimpleContainerIterator$<$ T $>$ Class Template Reference 130
5.39.1 Detailed Description 133
5.39.2 Constructor & Destructor Documentation 133
5.39.2.1 SimpleContainerIterator 133
5.39.2.2 ∼SimpleContainerIterator 133
5.39.3 Member Function Documentation 133
5.39.3.1 currentItem 133
5.39.3.2 first 134
5.39.3.3 isDone 134

5.39.3.4 next . . . 134
5.39.4 Friends And Related Function Documentation . . . 134
5.39.4.1 SimpleContainer< T > . . . 134
5.39.5 Member Data Documentation . . . 135
5.39.5.1 d_container . . . 135
5.39.5.2 d_current . . . 135
5.40 SimpleNetwork Class Reference . . . 135
5.40.1 Detailed Description . . . 136
5.40.2 Member Function Documentation . . . 136
5.40.2.1 show . . . 136
5.40.2.2 validate . . . 136
5.41 SimpleNeuralCreator Class Reference . . . 137
5.41.1 Detailed Description . . . 138
5.41.2 Constructor & Destructor Documentation . . . 138
5.41.2.1 SimpleNeuralCreator . . . 138
5.41.3 Member Function Documentation . . . 139
5.41.3.1 createFeedForwardFullyConnectedNetwork . . . 139
5.42 SimpleNeuron Class Reference . . . 139
5.42.1 Detailed Description . . . 142
5.42.2 Constructor & Destructor Documentation . . . 142
5.42.2.1 SimpleNeuron . . . 142
5.42.3 Member Function Documentation . . . 142
5.42.3.1 addCon . . . 142
5.42.3.2 getConIterator . . . 143
5.42.3.3 getId . . . 143
5.42.3.4 getInducedLocalField . . . 143
5.42.3.5 getOutput . . . 144
5.42.3.6 predict . . . 144
5.42.3.7 setActivationFunction . . . 144
5.42.3.8 setId . . . 144
5.42.3.9 setInducedLocalField . . . 145
5.42.3.10 setOutput . . . 145
5.42.3.11 setPredictBehavior . . . 145
5.42.3.12 show . . . 145

5.42.3.13 useActivationFunctionf0 . . . 146
5.42.3.14 validate . . . 146
5.43 Sine Class Reference . . . 147
5.43.1 Detailed Description . . . 149
5.43.2 Constructor & Destructor Documentation . . . 149
5.43.2.1 Sine . . . 149
5.43.3 Member Function Documentation . . . 149
5.43.3.1 f0 . . . 150
5.43.3.2 f1 . . . 150
5.44 SineFactory Class Reference . . . 150
5.44.1 Detailed Description . . . 153
5.44.2 Member Function Documentation . . . 153
5.44.2.1 makeActivationFunction . . . 153
5.45 Square Class Reference . . . 153
5.45.1 Detailed Description . . . 155
5.45.2 Constructor & Destructor Documentation . . . 155
5.45.2.1 Square . . . 155
5.45.3 Member Function Documentation . . . 155
5.45.3.1 f0 . . . 156
5.45.3.2 f1 . . . 156
5.46 SquareFactory Class Reference . . . 156
5.46.1 Detailed Description . . . 159
5.46.2 Member Function Documentation . . . 159
5.46.2.1 makeActivationFunction . . . 159
5.47 Tanh Class Reference . . . 159
5.47.1 Detailed Description . . . 161
5.47.2 Constructor & Destructor Documentation . . . 161
5.47.2.1 Tanh . . . 161
5.47.3 Member Function Documentation . . . 162
5.47.3.1 f0 . . . 162
5.47.3.2 f1 . . . 162
5.48 TanhFactory Class Reference . . . 163
5.48.1 Detailed Description . . . 166
5.48.2 Member Function Documentation . . . 166

5.48.2.1 makeActivationFunction 166
5.49 Threshold Class Reference 166
5.49.1 Detailed Description 168
5.49.2 Constructor & Destructor Documentation 168
5.49.2.1 Threshold 168
5.49.3 Member Function Documentation 168
5.49.3.1 f0 169
5.49.3.2 f1 169
5.50 ThresholdFactory Class Reference 169
5.50.1 Detailed Description 172
5.50.2 Member Function Documentation 172
5.50.2.1 makeActivationFunction 172
5.51 TrainingBehavior Class Reference 172
5.51.1 Detailed Description 173
5.51.2 Member Function Documentation 173
5.51.2.1 adjustParameters 173

6 File Documentation 175
6.1 pkg/AMORE/src/ActivationFunction.cpp File Reference 175
6.2 pkg/AMORE/src/AMORE.h File Reference 176
6.2.1 Define Documentation 178
6.2.1.1 foreach 178
6.2.1.2 size_type 178
6.2.2 Typedef Documentation 178
6.2.2.1 ActivationFunctionPtr 178
6.2.2.2 ActivationFunctionRef 178
6.2.2.3 ConContainerPtr 179
6.2.2.4 ConIteratorPtr 179
6.2.2.5 ConPtr 179
6.2.2.6 Handler 179
6.2.2.7 LayerPtr 179
6.2.2.8 NeuralCreatorPtr 179
6.2.2.9 NeuralFactoryPtr 179
6.2.2.10 NeuralNetworkPtr 179

6.2.2.11 NeuronContainerPtr 179
6.2.2.12 NeuronIteratorPtr 179
6.2.2.13 NeuronPtr 180
6.2.2.14 NeuronRef 180
6.2.2.15 NeuronWeakPtr 180
6.2.2.16 PredictBehaviorPtr 180
6.2.2.17 PredictBehaviorRef 180
6.2.2.18 TrainingBehaviorRef 180
6.3 pkg/AMORE/src/Con.cpp File Reference 180
6.4 pkg/AMORE/src/Container.cpp File Reference 181
6.5 pkg/AMORE/src/dia/ActivationFunction.h File Reference 182
6.6 pkg/AMORE/src/dia/AdaptBehavior.h File Reference 183
6.7 pkg/AMORE/src/dia/ADAPTgd.h File Reference 184
6.8 pkg/AMORE/src/dia/ADAPTgdwm.h File Reference 184
6.9 pkg/AMORE/src/dia/ArcTan.h File Reference 185
6.10 pkg/AMORE/src/dia/ArcTanFactory.h File Reference 186
6.11 pkg/AMORE/src/dia/BatchBehavior.h File Reference 186
6.12 pkg/AMORE/src/dia/BATCHgd.h File Reference 187
6.13 pkg/AMORE/src/dia/BATCHgdwm.h File Reference 188
6.14 pkg/AMORE/src/dia/Con.h File Reference 190
6.15 pkg/AMORE/src/dia/Container.h File Reference 190
6.16 pkg/AMORE/src/dia/Cosine.h File Reference 191
6.17 pkg/AMORE/src/dia/CosineFactory.h File Reference 191
6.18 pkg/AMORE/src/dia/Elliot.h File Reference 192
6.19 pkg/AMORE/src/dia/ElliotFactory.h File Reference 193
6.20 pkg/AMORE/src/dia/Exponential.h File Reference 193
6.21 pkg/AMORE/src/dia/ExponentialFactory.h File Reference 194
6.22 pkg/AMORE/src/dia/Gauss.h File Reference 195
6.23 pkg/AMORE/src/dia/GaussFactory.h File Reference 195
6.24 pkg/AMORE/src/dia/Identity.h File Reference 196
6.25 pkg/AMORE/src/dia/IdentityFactory.h File Reference 197
6.26 pkg/AMORE/src/dia/Iterator.h File Reference 199
6.27 pkg/AMORE/src/dia/Logistic.h File Reference 199
6.28 pkg/AMORE/src/dia/LogisticFactory.h File Reference 200

6.29 pkg/AMORE/src/dia/MLPbehavior.h File Reference 201
6.30 pkg/AMORE/src/dia/MLPfactory.h File Reference 202
6.31 pkg/AMORE/src/dia/NeuralCreator.h File Reference 203
6.32 pkg/AMORE/src/dia/NeuralFactory.h File Reference 203
6.33 pkg/AMORE/src/dia/NeuralNetwork.h File Reference 204
6.34 pkg/AMORE/src/dia/Neuron.h File Reference 205
6.35 pkg/AMORE/src/dia/PredictBehavior.h File Reference 206
6.36 pkg/AMORE/src/dia/RadialBasis.h File Reference 206
6.37 pkg/AMORE/src/dia/RadialBasisFactory.h File Reference 207
6.38 pkg/AMORE/src/dia/RBFbehavior.h File Reference 207
6.39 pkg/AMORE/src/dia/RBFfactory.h File Reference 208
6.40 pkg/AMORE/src/dia/Reciprocal.h File Reference 209
6.41 pkg/AMORE/src/dia/ReciprocalFactory.h File Reference 210
6.42 pkg/AMORE/src/dia/SimpleContainer.h File Reference 210
6.43 pkg/AMORE/src/dia/SimpleContainerIterator.h File Reference 211
6.44 pkg/AMORE/src/dia/SimpleNetwork.h File Reference 212
6.45 pkg/AMORE/src/dia/SimpleNeuralCreator.h File Reference 213
6.46 pkg/AMORE/src/dia/SimpleNeuron.h File Reference 214
6.47 pkg/AMORE/src/dia/Sine.h File Reference 215
6.48 pkg/AMORE/src/dia/SineFactory.h File Reference 216
6.49 pkg/AMORE/src/dia/Square.h File Reference 217
6.50 pkg/AMORE/src/dia/SquareFactory.h File Reference 217
6.51 pkg/AMORE/src/dia/Tanh.h File Reference 218
6.52 pkg/AMORE/src/dia/TanhFactory.h File Reference 219
6.53 pkg/AMORE/src/dia/Threshold.h File Reference 221
6.54 pkg/AMORE/src/dia/ThresholdFactory.h File Reference 221
6.55 pkg/AMORE/src/dia/TrainingBehavior.h File Reference 222
6.56 pkg/AMORE/src/Identity.cpp File Reference 223
6.57 pkg/AMORE/src/IdentityFactory.cpp File Reference 223
6.58 pkg/AMORE/src/Iterator.cpp File Reference 225
6.59 pkg/AMORE/src/IteratorInterface.cpp File Reference 225
6.60 pkg/AMORE/src/MLPbehavior.cpp File Reference 226
6.61 pkg/AMORE/src/MLPfactory.cpp File Reference 227
6.62 pkg/AMORE/src/NeuralNetwork.cpp File Reference 228

6.63 pkg/AMORE/src/Neuron.cpp File Reference 228
6.64 pkg/AMORE/src/PredictBehavior.cpp File Reference 229
6.65 pkg/AMORE/src/SimpleContainer.cpp File Reference 230
6.66 pkg/AMORE/src/SimpleContainerIterator.cpp File Reference 231
6.67 pkg/AMORE/src/SimpleNetwork.cpp File Reference 233
6.68 pkg/AMORE/src/SimpleNeuralCreator.cpp File Reference 233
6.69 pkg/AMORE/src/SimpleNeuron.cpp File Reference 234
6.70 pkg/AMORE/src/Tanh.cpp File Reference 235
6.71 pkg/AMORE/src/TanhFactory.cpp File Reference 236

Chapter 1

The AMORE++ package

1.1 Introduction

Here you will find the documentation of the C++ component of the AMORE++ R package.

The AMORE++ package is a new version of the publicly available AMORE package for neural network training and simulation under R

1.2 Motivation

Since the release of the previous version of the AMORE many things have changed in the R programming world.

The advent of the Reference Classes and of packages like Rcpp, inline and RUnit compel us to write a better version of the package in order to provide a more useful framework for neural network training and simulation.

1.3 Road Map

This project is currently very active and the development team intends to provide a beta version as soon as this summer (2011)

Chapter 2

Class Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

- ActivationFunction 9
  - ArcTan 19
  - Cosine 41
  - Elliot 47
  - Exponential 53
  - Gauss 59
  - Identity 65
  - Logistic 74
  - RadialBasis 105
  - Reciprocal 117
  - Sine 147
  - Square 153
  - Tanh 159
  - Threshold 166
- Con 32
- Container< T > 38
  - SimpleContainer< T > 122
- Iterator< T > 71
  - SimpleContainerIterator< T > 130
- NeuralCreator 90
  - SimpleNeuralCreator 137
- NeuralFactory 91
  - MLPfactory 84
    - ArcTanFactory 21
    - CosineFactory 44
    - ElliotFactory 50
    - ExponentialFactory 56
    - GaussFactory 62

        IdentityFactory 68
        LogisticFactory 76
        ReciprocalFactory 119
        SineFactory 150
        SquareFactory 156
        TanhFactory 163
        ThresholdFactory 169
    RBFfactory 113
        RadialBasisFactory 107
NeuralNetwork 93
    SimpleNetwork 135
Neuron 95
    SimpleNeuron 139
PredictBehavior 101
    MLPbehavior 79
    RBFbehavior 110
TrainingBehavior 172
    AdaptBehavior 11
        ADAPTgd 14
        ADAPTgdwm 16
    BatchBehavior 24
        BATCHgd 27
        BATCHgdwm 29

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ActivationFunction (Class ActivationFunction -) 9
AdaptBehavior (Class AdaptBehavior -) 11
ADAPTgd (Class ADAPTgd -) 14
ADAPTgdwm (Class ADAPTgdwm -) 16
ArcTan (Class ArcTan -) 19
ArcTanFactory (Class ArcTanFactory -) 21
BatchBehavior (Class BatchBehavior -) 24
BATCHgd (Class BATCHgd -) 27
BATCHgdwm (Class BATCHgdwm -) 29
Con (Class Con -) 32
Container$<$ T $>$ (Class Container -) 38
Cosine (Class Cosine -) 41
CosineFactory (Class CosineFactory -) 44
Elliot (Class Elliot -) 47
ElliotFactory (Class ElliotFactory -) 50
Exponential (Class Exponential -) 53
ExponentialFactory (Class ExponentialFactory -) 56
Gauss (Class Gauss -) 59
GaussFactory (Class GaussFactory -) 62
Identity (Class Identity -) 65
IdentityFactory (Class IdentityFactory -) 68
Iterator$<$ T $>$ (Class Iterator -) 71
Logistic (Class Logistic -) 74
LogisticFactory (Class LogisticFactory -) 76
MLPbehavior (Class MLPbehavior -) 79
MLPfactory (Class MLPfactory -) 84
NeuralCreator (Class NeuralCreator -) 90
NeuralFactory (Class NeuralFactory -) 91
NeuralNetwork (Class NeuralNetwork -) 93

Neuron (Class Neuron -) 95
PredictBehavior (Class PredictBehavior -) 101
RadialBasis (Class RadialBasis -) 105
RadialBasisFactory (Class RadialBasisFactory -) 107
RBFbehavior (Class RBFbehavior -) 110
RBFfactory (Class RBFfactory -) 113
Reciprocal (Class Reciprocal -) 117
ReciprocalFactory (Class ReciprocalFactory -) 119
SimpleContainer< T > (Class SimpleContainer -) 122
SimpleContainerIterator< T > (Class SimpleContainerIterator -) 130
SimpleNetwork (Class SimpleNetwork -) 135
SimpleNeuralCreator (Class SimpleNeuralCreator -) 137
SimpleNeuron (Class SimpleNeuron -) 139
Sine (Class Sine -) 147
SineFactory (Class SineFactory -) 150
Square (Class Square -) 153
SquareFactory (Class SquareFactory -) 156
Tanh (Class Tanh -) 159
TanhFactory (Class TanhFactory -) 163
Threshold (Class Threshold -) 166
ThresholdFactory (Class ThresholdFactory -) 169
TrainingBehavior (Class TrainingBehavior -) 172

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

pkg/AMORE/src/ActivationFunction.cpp 175
pkg/AMORE/src/AMORE.h 176
pkg/AMORE/src/Con.cpp 180
pkg/AMORE/src/Container.cpp 181
pkg/AMORE/src/Identity.cpp 223
pkg/AMORE/src/IdentityFactory.cpp 223
pkg/AMORE/src/Iterator.cpp 225
pkg/AMORE/src/IteratorInterface.cpp 225
pkg/AMORE/src/MLPbehavior.cpp 226
pkg/AMORE/src/MLPfactory.cpp 227
pkg/AMORE/src/NeuralNetwork.cpp 228
pkg/AMORE/src/Neuron.cpp 228
pkg/AMORE/src/PredictBehavior.cpp 229
pkg/AMORE/src/SimpleContainer.cpp 230
pkg/AMORE/src/SimpleContainerIterator.cpp 231
pkg/AMORE/src/SimpleNetwork.cpp 233
pkg/AMORE/src/SimpleNeuralCreator.cpp 233
pkg/AMORE/src/SimpleNeuron.cpp 234
pkg/AMORE/src/Tanh.cpp 235
pkg/AMORE/src/TanhFactory.cpp 236
pkg/AMORE/src/dia/ActivationFunction.h 182
pkg/AMORE/src/dia/AdaptBehavior.h 183
pkg/AMORE/src/dia/ADAPTgd.h 184
pkg/AMORE/src/dia/ADAPTgdwm.h 184
pkg/AMORE/src/dia/ArcTan.h 185
pkg/AMORE/src/dia/ArcTanFactory.h 186
pkg/AMORE/src/dia/BatchBehavior.h 186
pkg/AMORE/src/dia/BATCHgd.h 187
pkg/AMORE/src/dia/BATCHgdwm.h 188

pkg/AMORE/src/dia/Con.h 190
pkg/AMORE/src/dia/Container.h 190
pkg/AMORE/src/dia/Cosine.h 191
pkg/AMORE/src/dia/CosineFactory.h 191
pkg/AMORE/src/dia/Elliot.h 192
pkg/AMORE/src/dia/ElliotFactory.h 193
pkg/AMORE/src/dia/Exponential.h 193
pkg/AMORE/src/dia/ExponentialFactory.h 194
pkg/AMORE/src/dia/Gauss.h 195
pkg/AMORE/src/dia/GaussFactory.h 195
pkg/AMORE/src/dia/Identity.h 196
pkg/AMORE/src/dia/IdentityFactory.h 197
pkg/AMORE/src/dia/Iterator.h 199
pkg/AMORE/src/dia/Logistic.h 199
pkg/AMORE/src/dia/LogisticFactory.h 200
pkg/AMORE/src/dia/MLPbehavior.h 201
pkg/AMORE/src/dia/MLPfactory.h 202
pkg/AMORE/src/dia/NeuralCreator.h 203
pkg/AMORE/src/dia/NeuralFactory.h 203
pkg/AMORE/src/dia/NeuralNetwork.h 204
pkg/AMORE/src/dia/Neuron.h 205
pkg/AMORE/src/dia/PredictBehavior.h 206
pkg/AMORE/src/dia/RadialBasis.h 206
pkg/AMORE/src/dia/RadialBasisFactory.h 207
pkg/AMORE/src/dia/RBFbehavior.h 207
pkg/AMORE/src/dia/RBFfactory.h 208
pkg/AMORE/src/dia/Reciprocal.h 209
pkg/AMORE/src/dia/ReciprocalFactory.h 210
pkg/AMORE/src/dia/SimpleContainer.h 210
pkg/AMORE/src/dia/SimpleContainerIterator.h 211
pkg/AMORE/src/dia/SimpleNetwork.h 212
pkg/AMORE/src/dia/SimpleNeuralCreator.h 213
pkg/AMORE/src/dia/SimpleNeuron.h 214
pkg/AMORE/src/dia/Sine.h 215
pkg/AMORE/src/dia/SineFactory.h 216
pkg/AMORE/src/dia/Square.h 217
pkg/AMORE/src/dia/SquareFactory.h 217
pkg/AMORE/src/dia/Tanh.h 218
pkg/AMORE/src/dia/TanhFactory.h 219
pkg/AMORE/src/dia/Threshold.h 221
pkg/AMORE/src/dia/ThresholdFactory.h 221
pkg/AMORE/src/dia/TrainingBehavior.h 222

Chapter 5

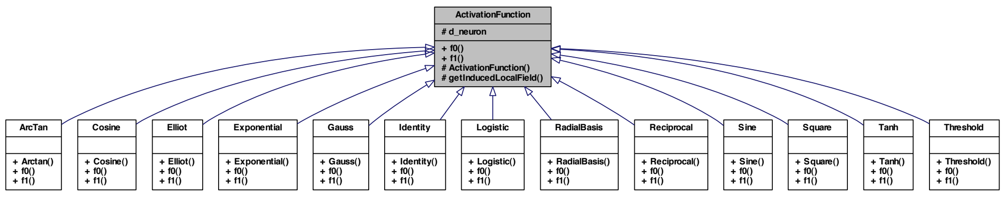
Class Documentation

5.1 ActivationFunction Class Reference

class ActivationFunction -

```
#include <ActivationFunction.h>
```

Inheritance diagram for ActivationFunction:



Public Member Functions

- virtual double f0 ()=0
- virtual double f1 ()=0

Protected Member Functions

- ActivationFunction (NeuronPtr neuronPtr)
- double getInducedLocalField ()

Protected Attributes

- NeuronWeakPtr d_neuron

5.1.1 Detailed Description

class ActivationFunction -

Definition at line 4 of file ActivationFunction.h.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 ActivationFunction::ActivationFunction (NeuronPtr *neuronPtr*) [protected]

Definition at line 11 of file ActivationFunction.cpp.

```
                                                              :
  d_neuron(neuronPtr)
{
}
```

5.1.3 Member Function Documentation

5.1.3.1 virtual double ActivationFunction::f0 ( ) [pure virtual]

Implemented in ArcTan, Cosine, Elliot, Exponential, Gauss, Identity, Logistic, RadialBasis, Reciprocal, Sine, Square, Tanh, and Threshold.

5.1.3.2 virtual double ActivationFunction::f1 ( ) [pure virtual]

Implemented in ArcTan, Cosine, Elliot, Exponential, Gauss, Identity, Logistic, RadialBasis, Reciprocal, Sine, Square, Tanh, and Threshold.

5.1.3.3 double ActivationFunction::getInducedLocalField ( ) [protected]

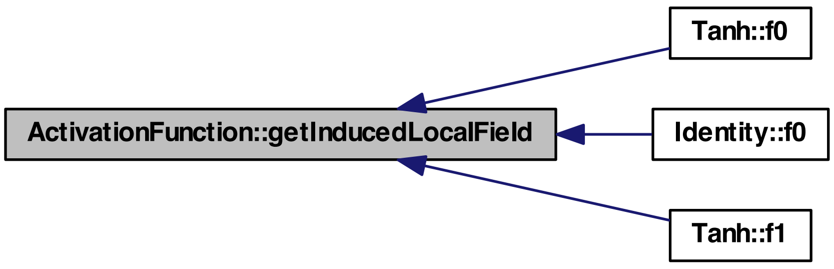
Definition at line 17 of file ActivationFunction.cpp.

References d_neuron.

Referenced by Tanh::f0(), Identity::f0(), and Tanh::f1().

```
{
  NeuronPtr neuronPtr(d_neuron.lock());
  return neuronPtr->getInducedLocalField();
}
```

Here is the caller graph for this function:



5.1.4 Member Data Documentation

5.1.4.1 NeuronWeakPtr ActivationFunction::d_neuron `[protected]`

Definition at line 7 of file ActivationFunction.h.

Referenced by getInducedLocalField().

The documentation for this class was generated from the following files:

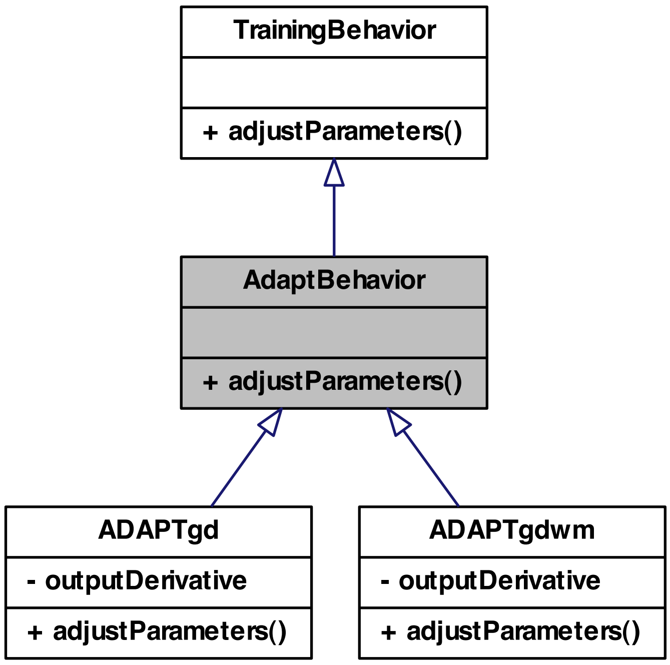
- pkg/AMORE/src/dia/ActivationFunction.h
- pkg/AMORE/src/ActivationFunction.cpp

5.2 AdaptBehavior Class Reference

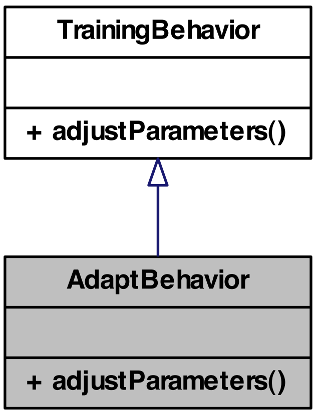
class AdaptBehavior -

`#include <AdaptBehavior.h>`

Inheritance diagram for AdaptBehavior:



Collaboration diagram for AdaptBehavior:



Public Member Functions

- virtual void adjustParameters ()=0

5.2.1 Detailed Description

class AdaptBehavior -

Definition at line 5 of file AdaptBehavior.h.

5.2.2 Member Function Documentation

5.2.2.1 virtual void AdaptBehavior::adjustParameters ( ) `[pure virtual]`

Reimplemented from TrainingBehavior.

Implemented in ADAPTgd, and ADAPTgdwm.

The documentation for this class was generated from the following file:

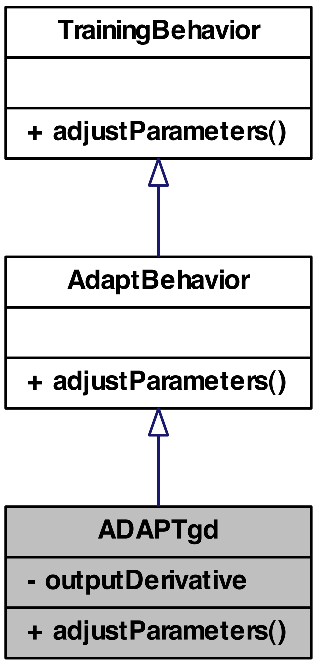
- pkg/AMORE/src/dia/AdaptBehavior.h

5.3 ADAPTgd Class Reference

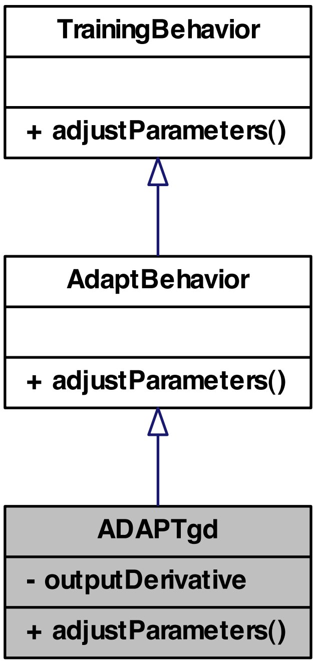
class ADAPTgd -

`#include <ADAPTgd.h>`

Inheritance diagram for ADAPTgd:



Collaboration diagram for ADAPTgd:



Public Member Functions

- void adjustParameters ()

Private Attributes

- double outputDerivative

5.3.1 Detailed Description

class ADAPTgd -

Definition at line 5 of file ADAPTgd.h.

5.3.2 Member Function Documentation

5.3.2.1 void ADAPTgd::adjustParameters ( ) `[virtual]`

Implements AdaptBehavior.

5.3.3 Member Data Documentation

5.3.3.1 double ADAPTgd::outputDerivative `[private]`

Definition at line 8 of file ADAPTgd.h.

The documentation for this class was generated from the following file:

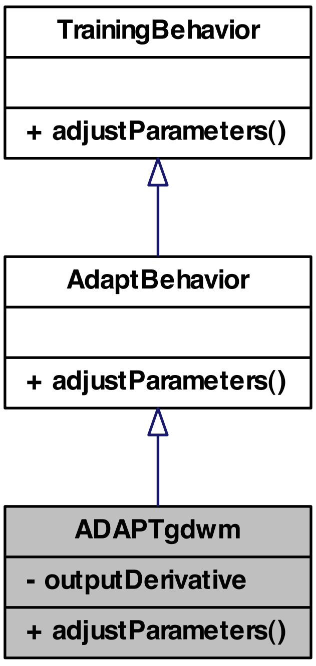
- pkg/AMORE/src/dia/ADAPTgd.h

5.4 ADAPTgdwm Class Reference

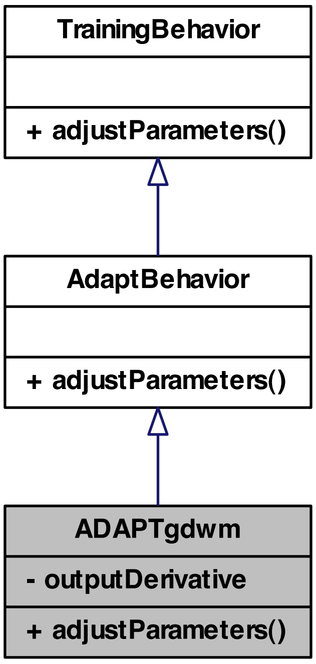
class ADAPTgdwm -

`#include <ADAPTgdwm.h>`

Inheritance diagram for ADAPTgdwm:



Collaboration diagram for ADAPTgdwm:



Public Member Functions

- void adjustParameters ()

Private Attributes

- double outputDerivative

5.4.1 Detailed Description

class ADAPTgdwm -

Definition at line 5 of file ADAPTgdwm.h.

5.4.2 Member Function Documentation

5.4.2.1 void ADAPTgdwm::adjustParameters ( ) [virtual]

Implements AdaptBehavior.

5.4.3 Member Data Documentation

5.4.3.1 double ADAPTgdwm::outputDerivative [private]

Definition at line 8 of file ADAPTgdwm.h.

The documentation for this class was generated from the following file:

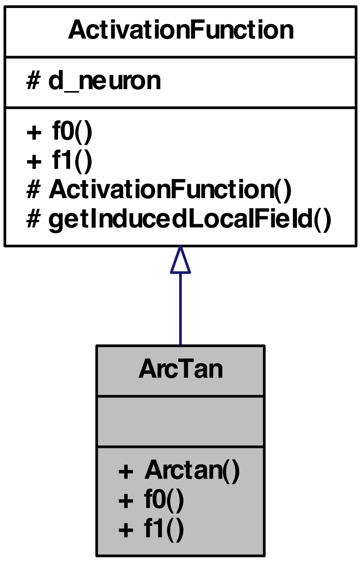
- pkg/AMORE/src/dia/ADAPTgdwm.h

5.5 ArcTan Class Reference

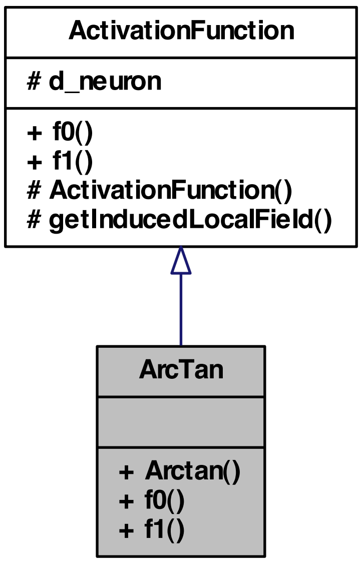
class ArcTan -

`#include <ArcTan.h>`

Inheritance diagram for ArcTan:



Collaboration diagram for ArcTan:



Public Member Functions

- Arctan (NeuronPtr neuronPtr)
- double f0 ()
- double f1 ()

5.5.1 Detailed Description

class ArcTan -

Definition at line 5 of file ArcTan.h.

5.5.2 Member Function Documentation

5.5.2.1 ArcTan::Arctan (NeuronPtr *neuronPtr*)

5.5.2.2 double ArcTan::f0 ( ) `[virtual]`

Implements ActivationFunction.

5.5.2.3 double ArcTan::f1 ( ) `[virtual]`

Implements ActivationFunction.

The documentation for this class was generated from the following file:

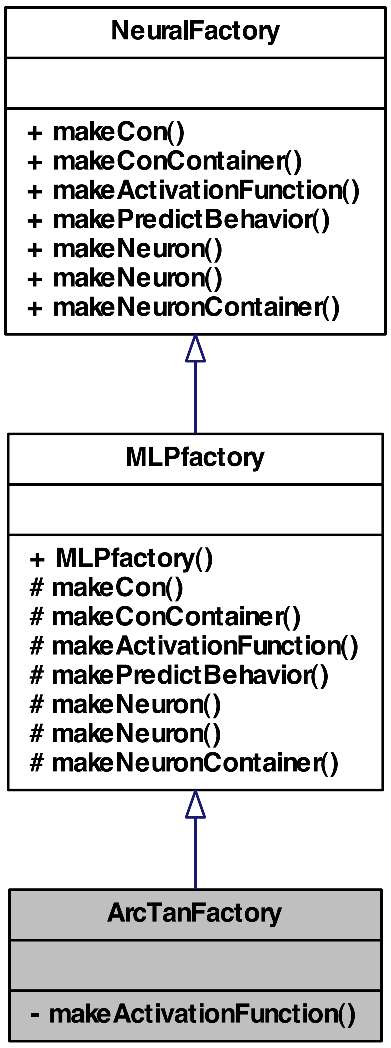
- pkg/AMORE/src/dia/ArcTan.h

5.6 ArcTanFactory Class Reference

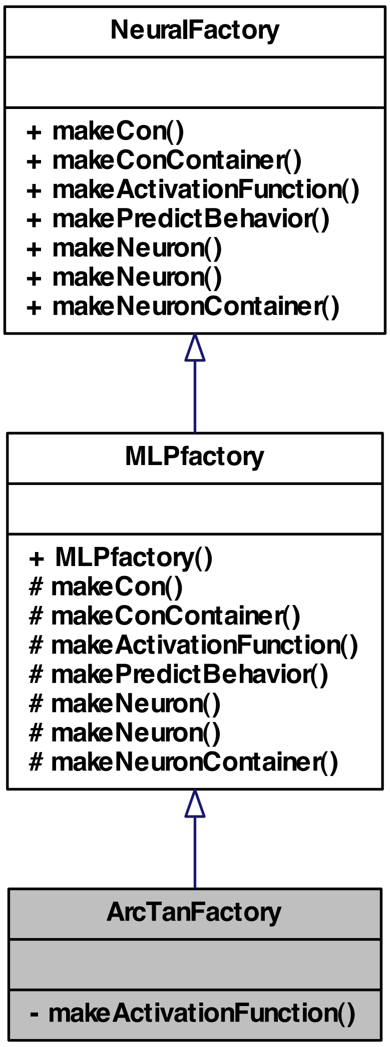
class ArcTanFactory -

`#include <ArcTanFactory.h>`

Inheritance diagram for ArcTanFactory:



Collaboration diagram for ArcTanFactory:



Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

5.6.1 Detailed Description

class ArcTanFactory -

Definition at line 5 of file ArcTanFactory.h.

5.6.2 Member Function Documentation

5.6.2.1 ActivationFunctionPtr ArcTanFactory::makeActivationFunction (NeuronPtr *neuronPtr*) `[private, virtual]`

Implements MLPfactory.

The documentation for this class was generated from the following file:

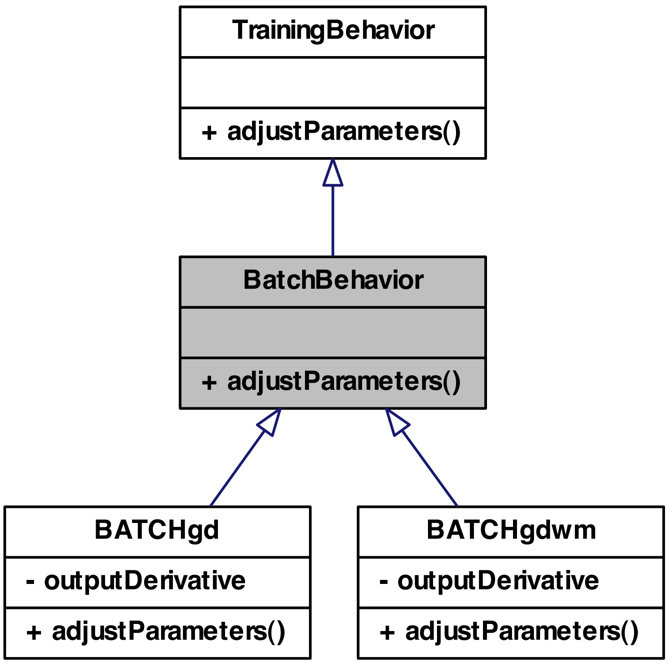
- pkg/AMORE/src/dia/ArcTanFactory.h

5.7 BatchBehavior Class Reference

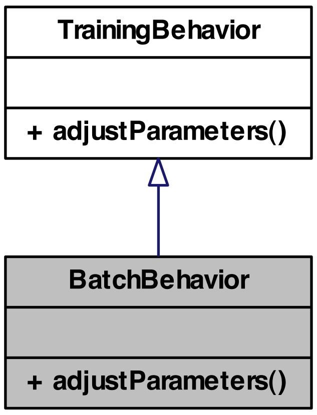
class BatchBehavior -

`#include <BatchBehavior.h>`

Inheritance diagram for BatchBehavior:



Collaboration diagram for BatchBehavior:



Public Member Functions

- virtual void adjustParameters ()=0

5.7.1 Detailed Description

class BatchBehavior -

Definition at line 5 of file BatchBehavior.h.

5.7.2 Member Function Documentation

5.7.2.1 virtual void BatchBehavior::adjustParameters ( ) `[pure virtual]`

Reimplemented from TrainingBehavior.

Implemented in BATCHgd, and BATCHgdwm.

The documentation for this class was generated from the following file:

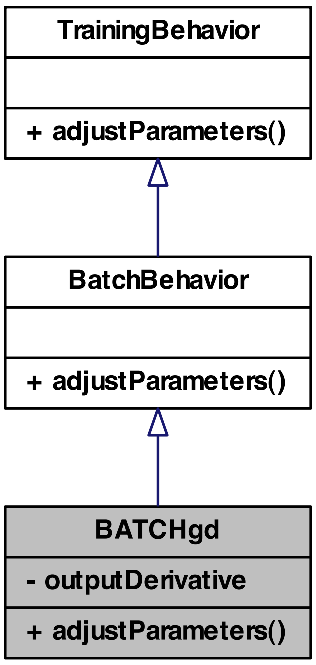
- pkg/AMORE/src/dia/BatchBehavior.h

5.8 BATCHgd Class Reference

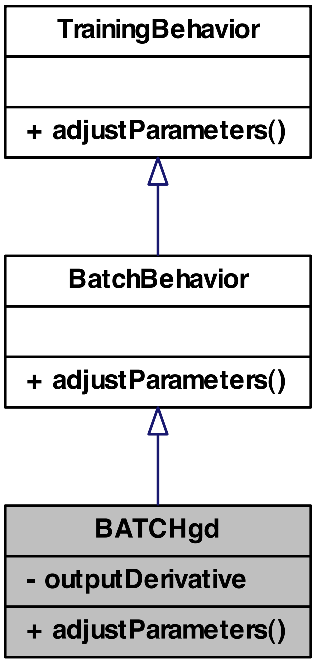
class BATCHgd -

`#include <BATCHgd.h>`

Inheritance diagram for BATCHgd:



Collaboration diagram for BATCHgd:



Public Member Functions

- void adjustParameters ()

Private Attributes

- double outputDerivative

5.8.1 Detailed Description

class BATCHgd -

Definition at line 5 of file BATCHgd.h.

5.8.2 Member Function Documentation

5.8.2.1 void BATCHgd::adjustParameters ( ) `[virtual]`

Implements BatchBehavior.

5.8.3 Member Data Documentation

5.8.3.1 double BATCHgd::outputDerivative `[private]`

Definition at line 8 of file BATCHgd.h.

The documentation for this class was generated from the following file:

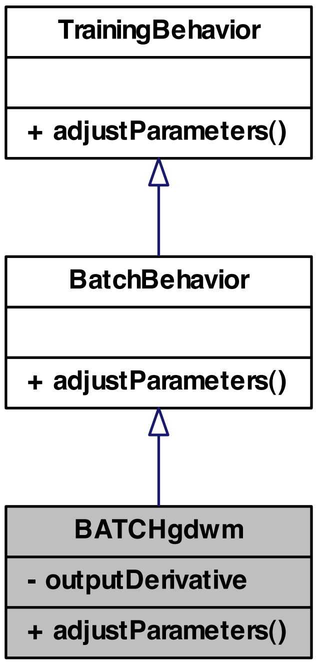
- pkg/AMORE/src/dia/BATCHgd.h

5.9 BATCHgdwm Class Reference

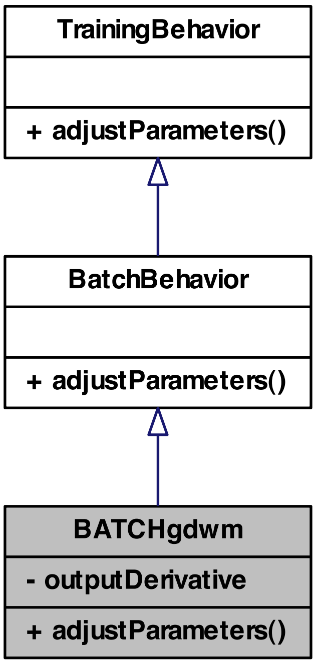
class BATCHgdwm -

`#include <BATCHgdwm.h>`

Inheritance diagram for BATCHgdwm:



Collaboration diagram for BATCHgdwm:



Public Member Functions

- void adjustParameters ()

Private Attributes

- double outputDerivative

5.9.1 Detailed Description

class BATCHgdwm -

Definition at line 5 of file BATCHgdwm.h.

5.9.2 Member Function Documentation

5.9.2.1 void BATCHgdwm::adjustParameters ( ) `[virtual]`

Implements BatchBehavior.

5.9.3 Member Data Documentation

5.9.3.1 double **BATCHgdwm::outputDerivative** `[private]`

Definition at line 8 of file BATCHgdwm.h.

The documentation for this class was generated from the following file:

- pkg/AMORE/src/dia/BATCHgdwm.h

5.10 Con Class Reference

class Con -

`#include <Con.h>`

Public Member Functions

- Con (Neuron &neuron)

  Constructor.
- Con (Neuron &neuron, double weight)

  Constructor.
- Handler Id ()

  A getter of the Id of the Neuron pointed by the from field.
- Neuron & getNeuron ()

  from field accessor.
- void setNeuron (Neuron &neuron)
- double getWeight ()

  weight field accessor.
- void setWeight (double weight)
- void show ()

  Pretty print of the Con information.
- bool validate ()

  Object validator.

Private Attributes

- NeuronRef d_neuron
- double d_weight

5.10.1 Detailed Description

class Con -

Definition at line 3 of file Con.h.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 Con::Con (**Neuron** & *neuron*)

Constructor.

Definition at line 19 of file Con.cpp.

```
                       :
  d_neuron( boost::ref(neuron) ), d_weight(0)
{
}
```

5.10.2.2 Con::Con (**Neuron** & *neuron,* double *weight*)

Constructor.

Definition at line 30 of file Con.cpp.

```
                                            :
  d_neuron(boost::ref(neuron)), d_weight(weight)
{
}
```

5.10.3 Member Function Documentation

5.10.3.1 **Neuron** & Con::getNeuron ( )

from field accessor.

This method allows access to the address stored in the private from field (a pointer to a Neuron object).∗

Returns

A pointer to the Neuron object referred to by the from field.

```
        //================
        //Usage example:
        //================
        // Data set up
                        NeuronPtr ptShNeuron ( new Neuron(1) );          // Neuron
       Id is set 1
                        ConPtr ptShCon( new Con(ptShNeuron) );           // from p
      oints to ptShNeuron and weight is set to 0
```

```
        // Test
                        ptShNeuron = ptShCon->getFrom() ;
                        int result = ptShNeuron->getId();

        // Now, result is equal to 1.
```

See also

getId and the unit test files, e.g., runit.Cpp.Con.R, for further examples.

Definition at line 56 of file Con.cpp.

References d_neuron.

```
{
  return d_neuron;
}
```

5.10.3.2 double Con::getWeight ( )

weight field accessor.

This method allows access to the value stored in the private field weight

Returns

The value of weight (double)

```
  //================
  //Usage example:
  //================
  // Data set up
                        std::vector<double> result;
                        NeuronPtr ptShNeuron ( new Neuron(16) );                /
      / Neuron Id is set to 16
                        ConPtr ptShCon( new Con(ptShNeuron, 12.4) );  // from poi
      nts to ptShNeuron and weight is set to 12.4
        // Test
                        result.push_back( ptShCon->getWeight() );
                        ptShCon->setWeight(2.2);
                        result.push_back( ptShCon->getWeight() );

        // Now, result is a numeric vector that contains the values 12.4 and 2.2
      .
```

See also

setWeight and the unit test files, e.g., runit.Cpp.Con.R, for further examples.

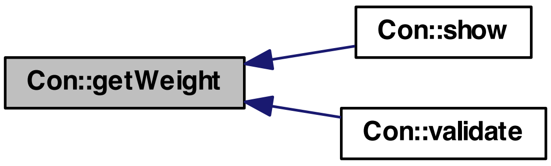
Definition at line 116 of file Con.cpp.

References d_weight.

Referenced by show(), and validate().

```
{
  return d_weight;
}
```

Here is the caller graph for this function:



5.10.3.3 int Con::Id ( )

A getter of the Id of the Neuron pointed by the from field.

This method gets the Id of the Neuron referred to by the from field

Returns

The value of the Id (an integer).

```
//================
//Usage example:
//================
// Data set up
                NeuronPtr ptShNeuron ( new Neuron(16) );        // Neuron I
d is set to 16
                ConPtr ptShCon( new Con(ptShNeuron) );          // from poi
nts to ptShNeuron and weight is set to 0
// Test
                int result = ptShCon->getId();

// Now, result is equal to 16.
```

See also

getFrom, setFrom and the unit test files, e.g., runit.Cpp.Con.R, for further examples.

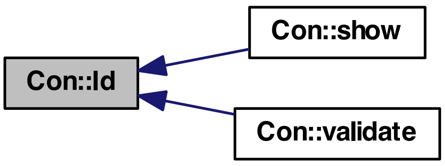
Definition at line 88 of file Con.cpp.

References d_neuron.

Referenced by show(), and validate().

```
{
  return d_neuron.get().getId();
}
```

Here is the caller graph for this function:



5.10.3.4 void Con::setNeuron (Neuron & *neuron*)

Definition at line 63 of file Con.cpp.

References d_neuron.

```
{
  d_neuron=boost::ref(neuron);
}
```

5.10.3.5 void Con::setWeight (double *weight*)

Definition at line 123 of file Con.cpp.

References d_weight.

```
{
  d_weight=weight;
}
```

5.10.3.6 void Con::show ( )

Pretty print of the Con information.

This method outputs in the R terminal the contents of the Con fields.

Returns

true in case everything works without throwing an exception

See also

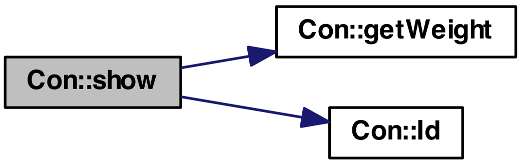
setWeight and the unit test files, e.g., runit.Cpp.Con.R, for usage examples.

Definition at line 135 of file Con.cpp.

References getWeight(), and Id().

```
{
  int id = Id();
  if (id == NA_INTEGER)
    {
      Rprintf("From: NA\t Invalid Connection \n");
    }
  else
    {
      Rprintf("From:\t %d \t Weight= \t %lf \n", id , getWeight() );
    }
}
```

Here is the call graph for this function:



5.10.3.7 bool Con::validate ( )

Object validator.

This method checks the object for internal coherence. A try / catch mechanism exits normal execution and returns control to the R terminal in case the contents of the Con object are identified as corrupted.

Returns

true in case the checks are Ok.

Exceptions

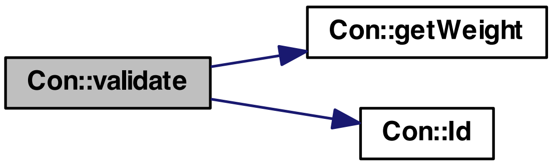
An	std::range error if weight or from are not finite.

Definition at line 155 of file Con.cpp.

References getWeight(), and Id().

```
{
  BEGIN_RCPP
  if (! R_FINITE(getWeight()) ) throw std::range_error("weight is not finite.");
  if (Id() == NA_INTEGER)
    throw std::range_error("fromId is not finite.");
  return (true);
END_RCPP}
```

Here is the call graph for this function:



5.10.4 Member Data Documentation

5.10.4.1 NeuronRef Con::d_neuron `[private]`

Definition at line 6 of file Con.h.

Referenced by getNeuron(), Id(), and setNeuron().

5.10.4.2 double Con::d_weight `[private]`

Definition at line 7 of file Con.h.

Referenced by getWeight(), and setWeight().

The documentation for this class was generated from the following files:

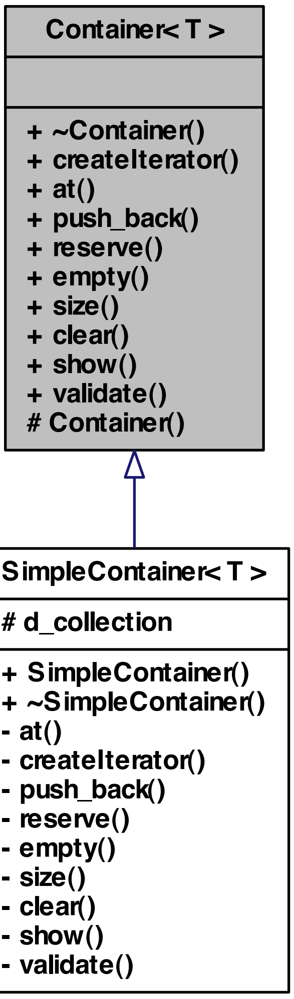
- pkg/AMORE/src/dia/Con.h
- pkg/AMORE/src/Con.cpp

5.11 Container< T > Class Template Reference

class Container -

`#include <Container.h>`

Inheritance diagram for Container< T >:



Public Member Functions

- virtual ~Container ()
- virtual boost::shared_ptr< Iterator< T > > createIterator ()=0
- virtual T at (size_type element)=0
- virtual void push_back (T const &const_reference)=0
- virtual void reserve (int n)=0

- virtual bool empty ()=0
- virtual size_type size ()=0
- virtual void clear ()=0
- virtual void show ()=0
- virtual bool validate ()=0

Protected Member Functions

- Container ()

5.11.1 Detailed Description

template<typename T>class Container< T >

class Container -

Definition at line 5 of file Container.h.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 template<typename T > Container< T >::~Container ( ) [virtual]

Definition at line 20 of file Container.cpp.

```
{
}
```

5.11.2.2 template<typename T > Container< T >::Container ( ) [protected]

Definition at line 14 of file Container.cpp.

```
{
}
```

5.11.3 Member Function Documentation

5.11.3.1 template<typename T > virtual T Container< T >::at (size_type *element*) [pure virtual]

Implemented in SimpleContainer< T >.

5.11.3.2 template<typename T > virtual void Container< T >::clear ( ) [pure virtual]

Implemented in SimpleContainer< T >.

5.11.3.3 template<typename T > virtual boost::shared_ptr< Iterator<T> > Container< T >::createIterator ( ) [pure virtual]

Implemented in SimpleContainer< T >.

5.11.3.4 template<typename T > virtual bool Container< T >::empty ( ) [pure virtual]

Implemented in SimpleContainer< T >.

5.11.3.5 template<typename T > virtual void Container< T >::push_back (T const & *const_reference*) [pure virtual]

Implemented in SimpleContainer< T >.

5.11.3.6 template<typename T > virtual void Container< T >::reserve (int *n*) [pure virtual]

Implemented in SimpleContainer< T >.

5.11.3.7 template<typename T > virtual void Container< T >::show ( ) [pure virtual]

Implemented in SimpleContainer< T >.

5.11.3.8 template<typename T > virtual size_type Container< T >::size ( ) [pure virtual]

Implemented in SimpleContainer< T >.

5.11.3.9 template<typename T > virtual bool Container< T >::validate ( ) [pure virtual]

Implemented in SimpleContainer< T >.

The documentation for this class was generated from the following files:

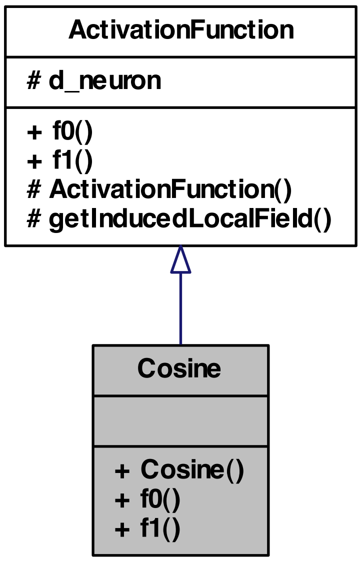
- pkg/AMORE/src/dia/Container.h
- pkg/AMORE/src/Container.cpp

5.12 Cosine Class Reference

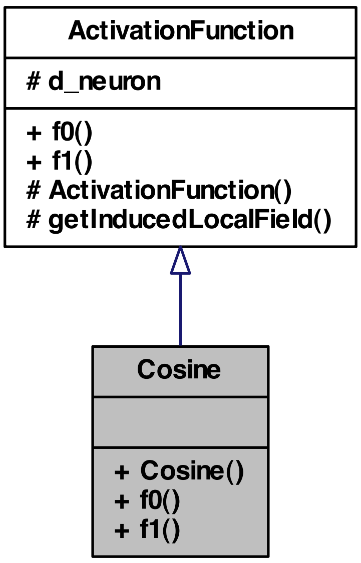
class Cosine -

```
#include <Cosine.h>
```

Inheritance diagram for Cosine:



Collaboration diagram for Cosine:



Public Member Functions

- Cosine (NeuronPtr neuronPtr)
- double f0 ()
- double f1 ()

5.12.1 Detailed Description

class Cosine -

Definition at line 5 of file Cosine.h.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 Cosine::Cosine (NeuronPtr *neuronPtr*)

5.12.3 Member Function Documentation

5.12.3.1 double Cosine::f0 ( ) [virtual]

Implements ActivationFunction.

5.12.3.2 double Cosine::f1 ( ) [virtual]

Implements ActivationFunction.

The documentation for this class was generated from the following file:

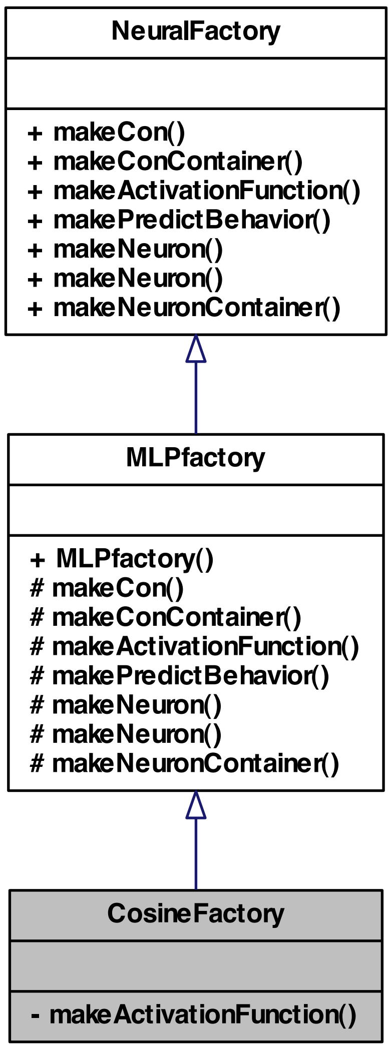
- pkg/AMORE/src/dia/Cosine.h

5.13 CosineFactory Class Reference

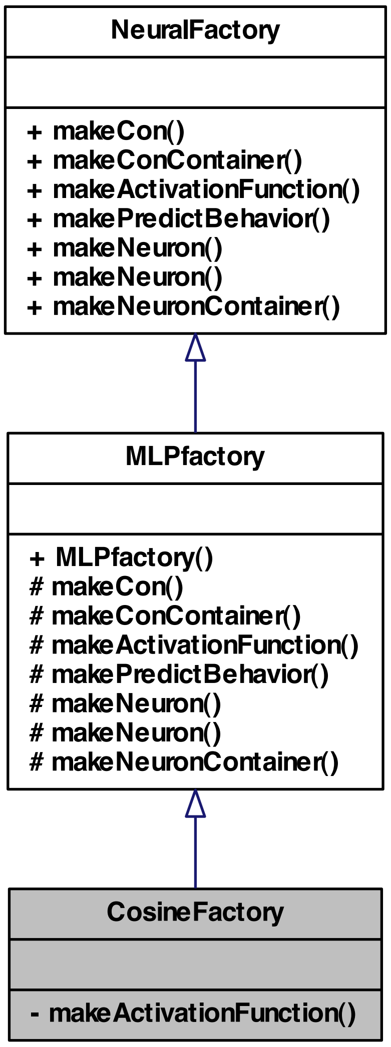
class CosineFactory -

```
#include <CosineFactory.h>
```

Inheritance diagram for CosineFactory:



Collaboration diagram for CosineFactory:



Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

5.13.1 Detailed Description

class CosineFactory -

Definition at line 5 of file CosineFactory.h.

5.13.2 Member Function Documentation

5.13.2.1 ActivationFunctionPtr CosineFactory::makeActivationFunction (NeuronPtr *neuronPtr*) `[private, virtual]`

Implements MLPfactory.

The documentation for this class was generated from the following file:

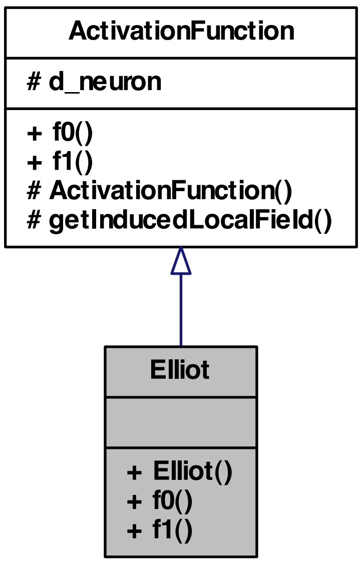
- pkg/AMORE/src/dia/CosineFactory.h

5.14 Elliot Class Reference

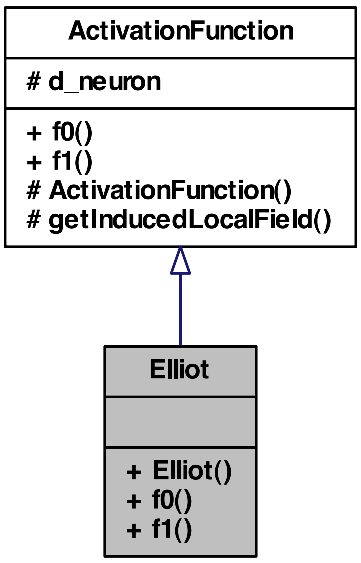
class Elliot -

`#include <Elliot.h>`

Inheritance diagram for Elliot:



Collaboration diagram for Elliot:



Public Member Functions

- Elliot (NeuronPtr neuronPtr)
- double f0 ()
- double f1 ()

5.14.1 Detailed Description

class Elliot -

Definition at line 5 of file Elliot.h.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 Elliot::Elliot (NeuronPtr *neuronPtr*)

5.14.3 Member Function Documentation

5.14.3.1 double Elliot::f0 ( ) [virtual]

Implements ActivationFunction.

5.14.3.2 double Elliot::f1 ( ) [virtual]

Implements ActivationFunction.

The documentation for this class was generated from the following file:

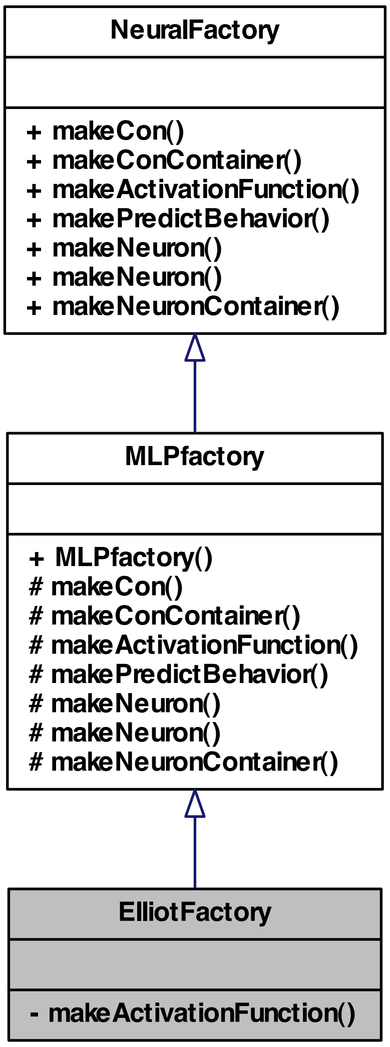
- pkg/AMORE/src/dia/Elliot.h

5.15 ElliotFactory Class Reference

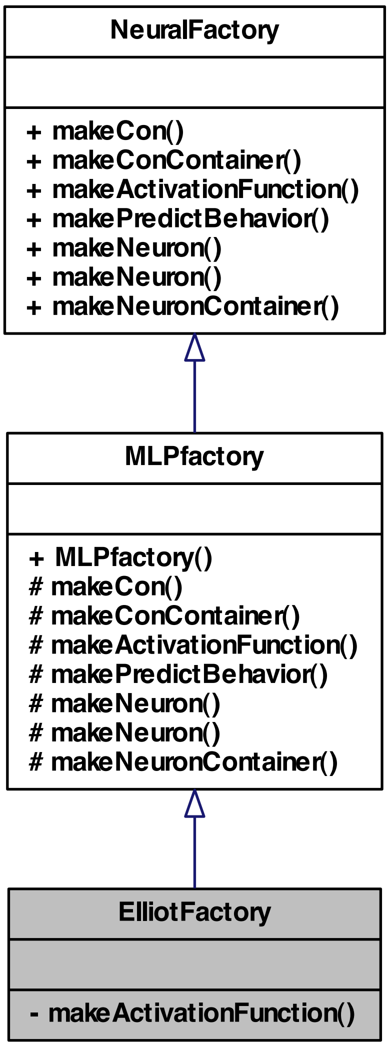
class ElliotFactory -

`#include <ElliotFactory.h>`

Inheritance diagram for ElliotFactory:



Collaboration diagram for ElliotFactory:



Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

5.15.1 Detailed Description

class ElliotFactory -

Definition at line 5 of file ElliotFactory.h.

5.15.2 Member Function Documentation

5.15.2.1 ActivationFunctionPtr ElliotFactory::makeActivationFunction (NeuronPtr *neuronPtr*) `[private, virtual]`

Implements MLPfactory.

The documentation for this class was generated from the following file:

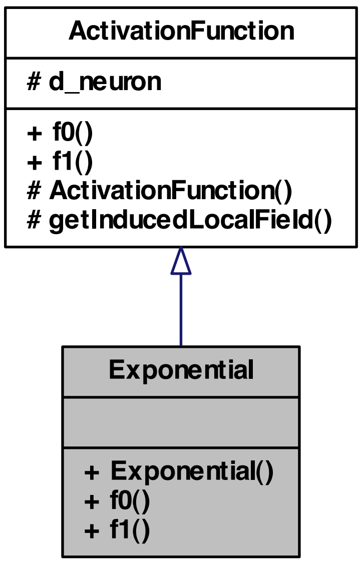
- pkg/AMORE/src/dia/ElliotFactory.h

5.16 Exponential Class Reference

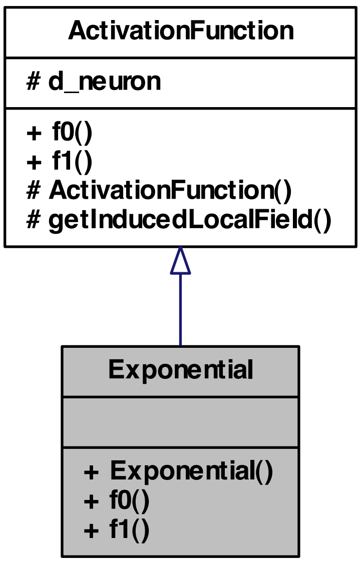
class Exponential -

`#include <Exponential.h>`

Inheritance diagram for Exponential:



Collaboration diagram for Exponential:



Public Member Functions

- Exponential (NeuronPtr neuronPtr)
- double f0 ()
- double f1 ()

5.16.1 Detailed Description

class Exponential -

Definition at line 5 of file Exponential.h.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 Exponential::Exponential (NeuronPtr *neuronPtr*)

5.16.3 Member Function Documentation

5.16.3.1 double Exponential::f0 ( ) `[virtual]`

Implements ActivationFunction.

5.16.3.2 double Exponential::f1 ( ) `[virtual]`

Implements ActivationFunction.

The documentation for this class was generated from the following file:

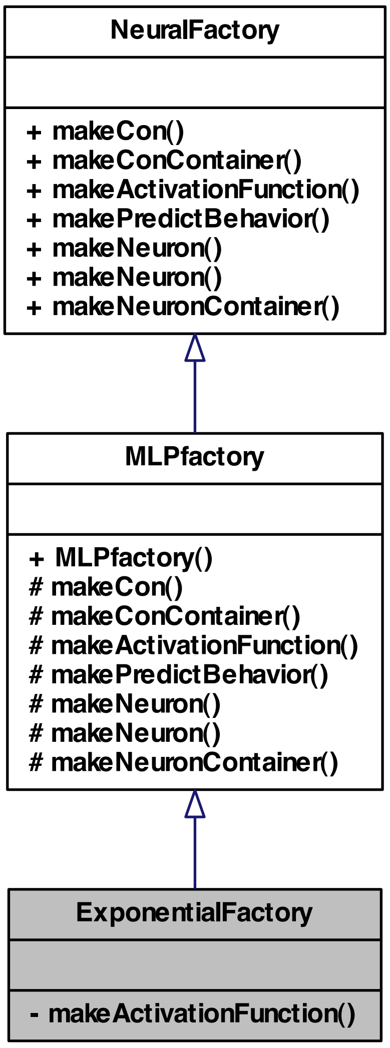
- pkg/AMORE/src/dia/Exponential.h

5.17 ExponentialFactory Class Reference

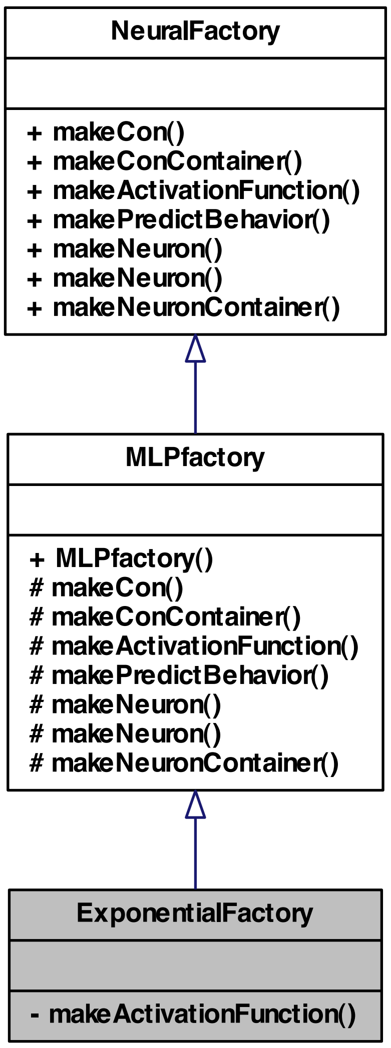
class ExponentialFactory -

`#include <ExponentialFactory.h>`

Inheritance diagram for ExponentialFactory:



Collaboration diagram for ExponentialFactory:



Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

5.17.1 Detailed Description

class ExponentialFactory -

Definition at line 5 of file ExponentialFactory.h.

5.17.2 Member Function Documentation

5.17.2.1 ActivationFunctionPtr ExponentialFactory::makeActivationFunction (NeuronPtr *neuronPtr*) `[private, virtual]`

Implements MLPfactory.

The documentation for this class was generated from the following file:

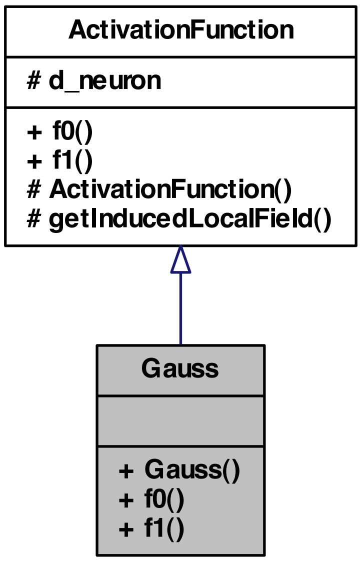
- pkg/AMORE/src/dia/ExponentialFactory.h

5.18 Gauss Class Reference

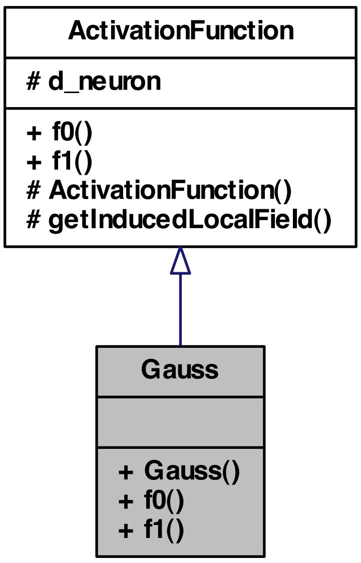
class Gauss -

`#include <Gauss.h>`

Inheritance diagram for Gauss:



Collaboration diagram for Gauss:



Public Member Functions

- Gauss (NeuronPtr neuronPtr)
- double f0 ()
- double f1 ()

5.18.1 Detailed Description

class Gauss -

Definition at line 5 of file Gauss.h.

5.18.2 Constructor & Destructor Documentation

5.18.2.1 Gauss::Gauss (NeuronPtr *neuronPtr*)

5.18.3 Member Function Documentation

5.18.3.1 double Gauss::f0 ( ) `[virtual]`

Implements ActivationFunction.

5.18.3.2 double Gauss::f1 ( ) `[virtual]`

Implements ActivationFunction.

The documentation for this class was generated from the following file:

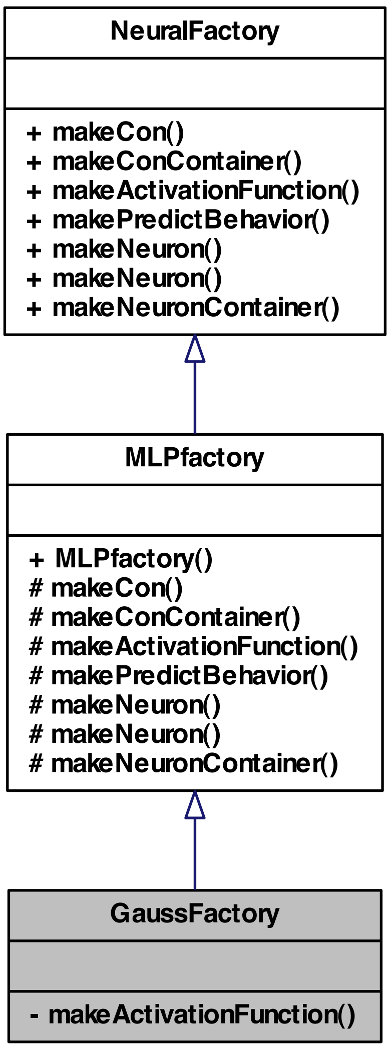
- pkg/AMORE/src/dia/Gauss.h

5.19 GaussFactory Class Reference

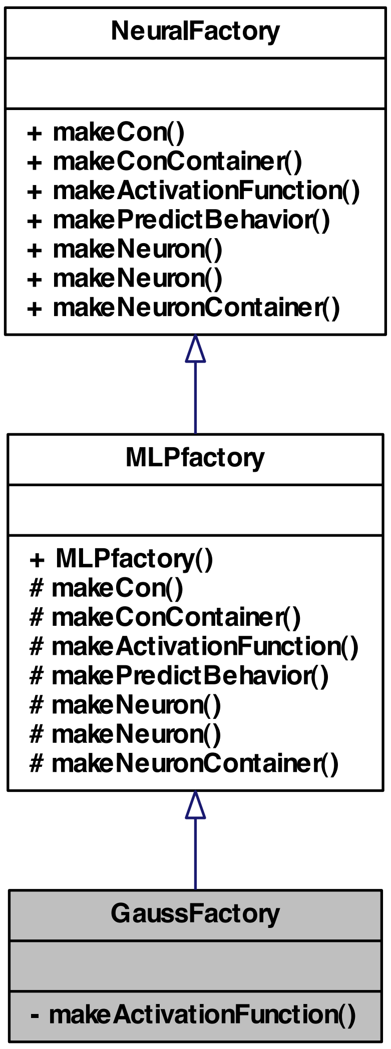
class GaussFactory -

`#include <GaussFactory.h>`

Inheritance diagram for GaussFactory:



Collaboration diagram for GaussFactory:



Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

5.19.1 Detailed Description

class GaussFactory -

Definition at line 5 of file GaussFactory.h.

5.19.2 Member Function Documentation

5.19.2.1 ActivationFunctionPtr GaussFactory::makeActivationFunction (NeuronPtr *neuronPtr*) `[private, virtual]`

Implements MLPfactory.

The documentation for this class was generated from the following file:

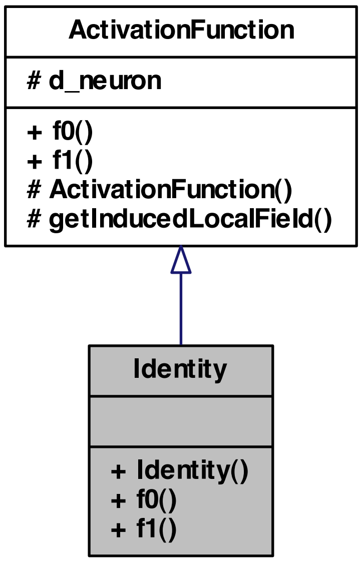
- pkg/AMORE/src/dia/GaussFactory.h

5.20 Identity Class Reference

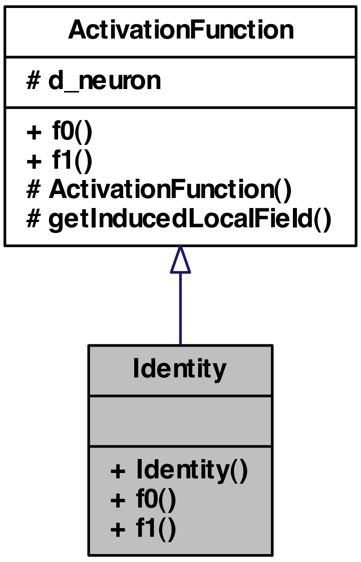
class Identity -

`#include <Identity.h>`

Inheritance diagram for Identity:



Collaboration diagram for Identity:



Public Member Functions

- Identity (NeuronPtr neuronPtr)
- double f0 ()
- double f1 ()

5.20.1 Detailed Description

class Identity -

Definition at line 5 of file Identity.h.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 Identity::Identity (**NeuronPtr** ***neuronPtr***)

Definition at line 12 of file Identity.cpp.

```
                                             : ActivationFunction(neuronPtr) {
}
```

5.20.3 Member Function Documentation

5.20.3.1 double Identity::f0 ( ) [virtual]

Implements ActivationFunction.

Definition at line 16 of file Identity.cpp.

References ActivationFunction::getInducedLocalField().

```
                     {
  return getInducedLocalField() ;
}
```

Here is the call graph for this function:

Identity::f0
ActivationFunction::getInducedLocalField

5.20.3.2 double Identity::f1 ( ) [virtual]

Implements ActivationFunction.

Definition at line 20 of file Identity.cpp.

```
                     {
  return 1 ;
}
```

The documentation for this class was generated from the following files:

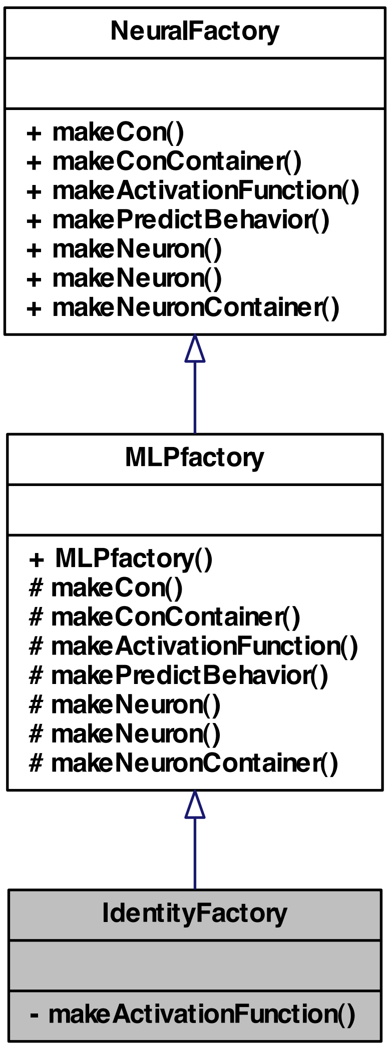
- pkg/AMORE/src/dia/Identity.h
- pkg/AMORE/src/Identity.cpp

5.21 IdentityFactory Class Reference

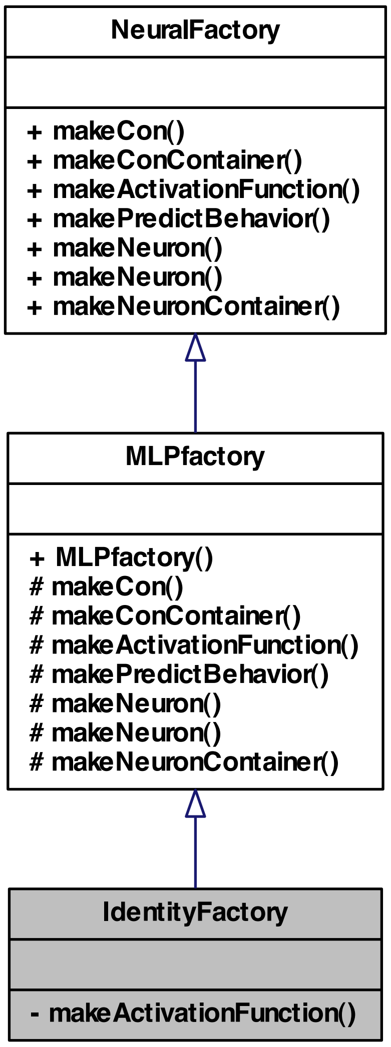
class IdentityFactory -

`#include <IdentityFactory.h>`

Inheritance diagram for IdentityFactory:



Collaboration diagram for IdentityFactory:



Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

5.21.1 Detailed Description

class IdentityFactory -

Definition at line 5 of file IdentityFactory.h.

5.21.2 Member Function Documentation

5.21.2.1 ActivationFunctionPtr IdentityFactory::makeActivationFunction (NeuronPtr *neuronPtr*) [private, virtual]

Implements MLPfactory.

Definition at line 17 of file IdentityFactory.cpp.

```
{
  ActivationFunctionPtr activationFunctionPtr(new Identity(neuronPtr));
  return activationFunctionPtr;
}
```

The documentation for this class was generated from the following files:

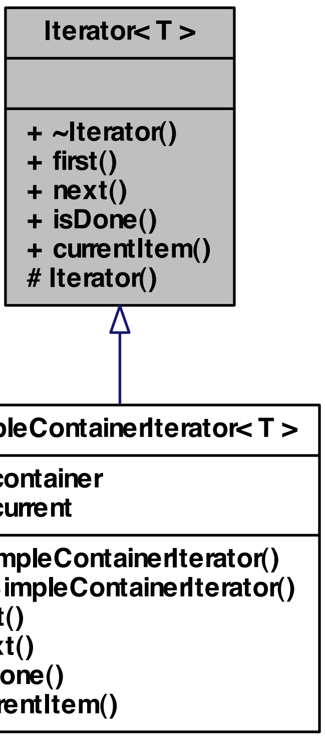
- pkg/AMORE/src/dia/IdentityFactory.h
- pkg/AMORE/src/IdentityFactory.cpp

5.22 Iterator< T > Class Template Reference

class Iterator -

`#include <Iterator.h>`

Inheritance diagram for Iterator< T >:



Public Member Functions

- virtual ~Iterator ()
- virtual void first ()=0
- virtual void next ()=0
- virtual bool isDone ()=0
- virtual T currentItem ()=0

Protected Member Functions

- Iterator ()

5.22.1 Detailed Description

template<typename T>class Iterator< T >

class Iterator -

Definition at line 5 of file Iterator.h.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 template<typename T > Iterator< T >::~Iterator ( ) [virtual]

Definition at line 20 of file Iterator.cpp.

```
{
}
```

5.22.2.2 template<typename T > Iterator< T >::Iterator ( ) [protected]

Definition at line 14 of file Iterator.cpp.

```
{
}
```

5.22.3 Member Function Documentation

5.22.3.1 template<typename T > virtual T Iterator< T >::currentItem ( ) [pure virtual]

Implemented in SimpleContainerIterator< T >.

5.22.3.2 template<typename T > virtual void Iterator< T >::first ( ) [pure virtual]

Implemented in SimpleContainerIterator< T >.

5.22.3.3 template<typename T > virtual bool Iterator< T >::isDone ( ) [pure virtual]

Implemented in SimpleContainerIterator< T >.

5.22.3.4 template<typename T > virtual void Iterator< T >::next ( ) [pure virtual]

Implemented in SimpleContainerIterator< T >.

The documentation for this class was generated from the following files:

- pkg/AMORE/src/dia/Iterator.h

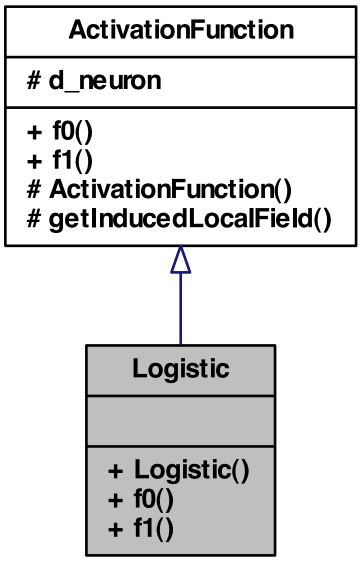
- pkg/AMORE/src/Iterator.cpp

5.23 Logistic Class Reference

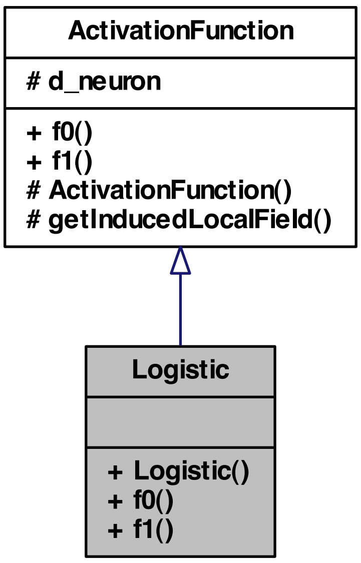
class Logistic -

```
#include <Logistic.h>
```

Inheritance diagram for Logistic:



Collaboration diagram for Logistic:



Public Member Functions

- Logistic (NeuronPtr neuronPtr)
- double f0 ()
- double f1 ()

5.23.1 Detailed Description

class Logistic -

Definition at line 5 of file Logistic.h.

5.23.2 Constructor & Destructor Documentation

5.23.2.1 Logistic::Logistic (NeuronPtr *neuronPtr*)

5.23.3 Member Function Documentation

5.23.3.1 double Logistic::f0 ( ) `[virtual]`

Implements ActivationFunction.

5.23.3.2 double Logistic::f1 ( ) `[virtual]`

Implements ActivationFunction.

The documentation for this class was generated from the following file:

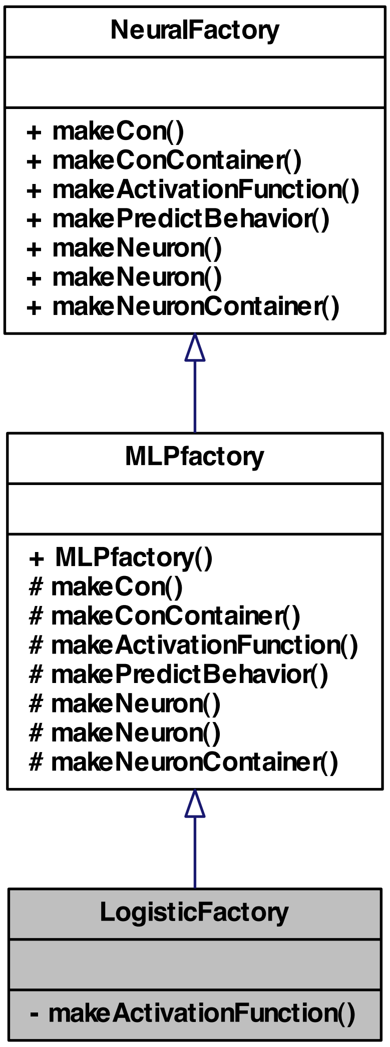
- pkg/AMORE/src/dia/Logistic.h

5.24 LogisticFactory Class Reference

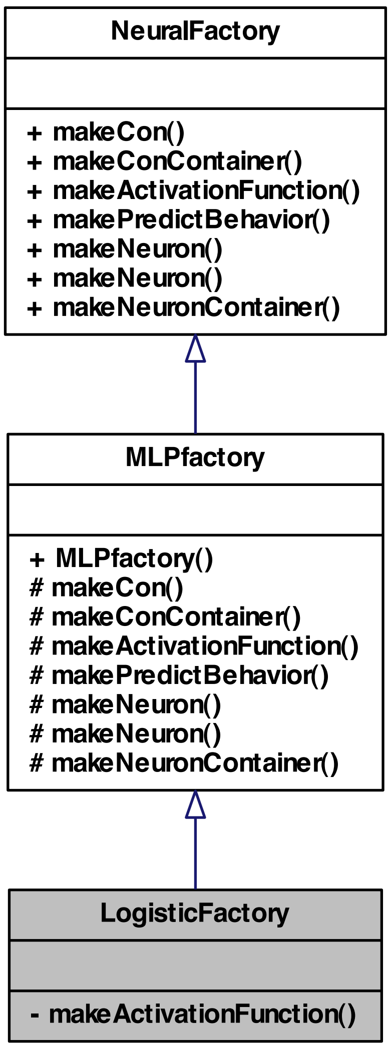
class LogisticFactory -

`#include <LogisticFactory.h>`

Inheritance diagram for LogisticFactory:



Collaboration diagram for LogisticFactory:



Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

5.24.1 Detailed Description

class LogisticFactory -

Definition at line 5 of file LogisticFactory.h.

5.24.2 Member Function Documentation

5.24.2.1 ActivationFunctionPtr LogisticFactory::makeActivationFunction (NeuronPtr *neuronPtr*) [private, virtual]

Implements MLPfactory.

The documentation for this class was generated from the following file:

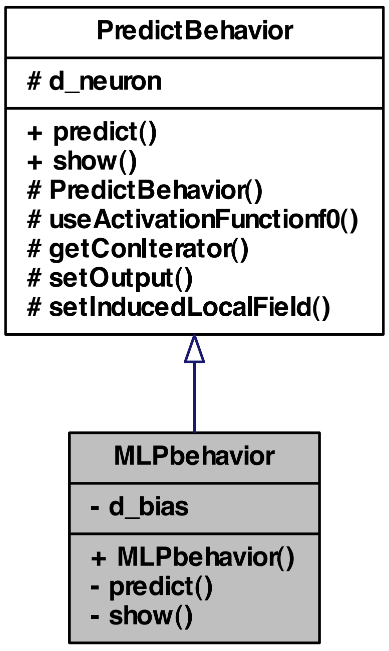
- pkg/AMORE/src/dia/LogisticFactory.h

5.25 MLPbehavior Class Reference

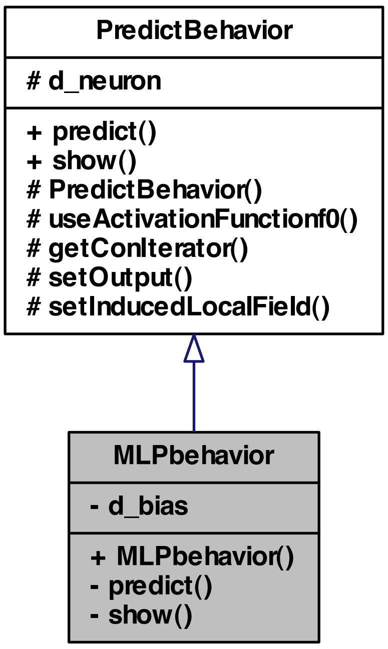
class MLPbehavior -

`#include <MLPbehavior.h>`

Inheritance diagram for MLPbehavior:



Collaboration diagram for MLPbehavior:



Public Member Functions

- MLPbehavior (NeuronPtr neuronPtr)

Private Member Functions

- void predict ()
- void show ()

Private Attributes

- double d_bias

Friends

- class MLPfactory

5.25.1 Detailed Description

class MLPbehavior -

Definition at line 5 of file MLPbehavior.h.

5.25.2 Constructor & Destructor Documentation

5.25.2.1 MLPbehavior::MLPbehavior (NeuronPtr *neuronPtr*)

Definition at line 13 of file MLPbehavior.cpp.

```
                                                           :
   PredictBehavior(neuronPtr) , d_bias(0.0)
{
}
```

5.25.3 Member Function Documentation

5.25.3.1 void MLPbehavior::predict ( ) [private, virtual]

Implements PredictBehavior.

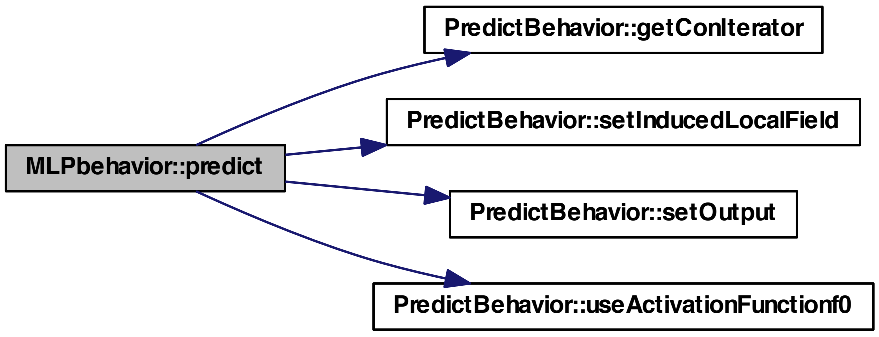
Definition at line 19 of file MLPbehavior.cpp.

References d_bias, PredictBehavior::getConIterator(), PredictBehavior::setInducedLocalField(), PredictBehavior::setOutput(), and PredictBehavior::useActivationFunctionf0().

```
{

  double accumulator(d_bias);
  ConIteratorPtr conIterator = getConIterator();
  double weight;
  double incomingSignalValue;
  for (conIterator->first(); !conIterator->isDone(); conIterator->next())
    {
      weight = conIterator->currentItem()->getWeight();
      incomingSignalValue = conIterator->currentItem()->getNeuron().getOutput();
      accumulator += weight * incomingSignalValue;
    }
  setInducedLocalField(accumulator);
  setOutput (
  useActivationFunctionf0());
}
```

Here is the call graph for this function:


MLPbehavior::predict
PredictBehavior::getConIterator
PredictBehavior::setInducedLocalField
PredictBehavior::setOutput
PredictBehavior::useActivationFunctionf0

5.25.3.2 void MLPbehavior::show ( ) [private, virtual]

Implements PredictBehavior.

Definition at line 38 of file MLPbehavior.cpp.

References d_bias.

```
{
  Rprintf("\n bias: %lf", d_bias);
}
```

5.25.4 Friends And Related Function Documentation

5.25.4.1 friend class **MLPfactory** [friend]

Definition at line 11 of file MLPbehavior.h.

5.25.5 Member Data Documentation

5.25.5.1 double **MLPbehavior::d_bias** [private]

Definition at line 8 of file MLPbehavior.h.

Referenced by MLPfactory::makeNeuron(), predict(), and show().

The documentation for this class was generated from the following files:

- pkg/AMORE/src/dia/MLPbehavior.h

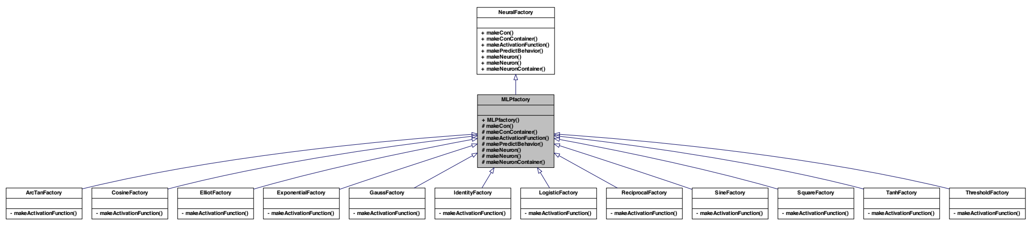
- pkg/AMORE/src/MLPbehavior.cpp

5.26 MLPfactory Class Reference

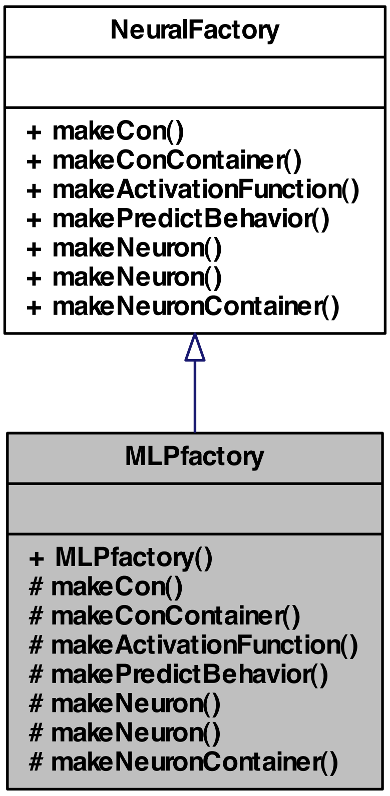
class MLPfactory -

`#include <MLPfactory.h>`

Inheritance diagram for MLPfactory:



Collaboration diagram for MLPfactory:



Public Member Functions

- MLPfactory ()

Protected Member Functions

- ConPtr makeCon (Neuron &neuron, double weight)
- ConContainerPtr makeConContainer ()
- virtual ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)=0
- PredictBehaviorPtr makePredictBehavior (NeuronPtr neuronPtr)
- NeuronPtr makeNeuron (Handler Id)
- NeuronPtr makeNeuron (Handler Id, NeuronIteratorPtr neuronIteratorPtr, double totalAmountOfParameters)

- NeuronContainerPtr makeNeuronContainer ()

5.26.1 Detailed Description

class MLPfactory -

Definition at line 5 of file MLPfactory.h.

5.26.2 Constructor & Destructor Documentation

5.26.2.1 MLPfactory::MLPfactory ( )

Definition at line 13 of file MLPfactory.cpp.

```
{
}
```

5.26.3 Member Function Documentation

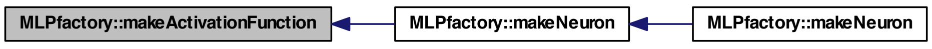
5.26.3.1 virtual **ActivationFunctionPtr** MLPfactory::makeActivationFunction (**NeuronPtr** *neuronPtr*) [protected, pure virtual]

Implements NeuralFactory.

Implemented in ArcTanFactory, CosineFactory, ElliotFactory, ExponentialFactory, GaussFactory, IdentityFactory, LogisticFactory, ReciprocalFactory, SineFactory, SquareFactory, TanhFactory, and ThresholdFactory.

Referenced by makeNeuron().

Here is the caller graph for this function:



5.26.3.2 **ConPtr** MLPfactory::makeCon (**Neuron** & *neuron,* double *weight*) [protected, virtual]

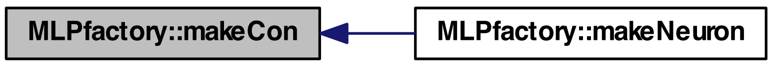
Implements NeuralFactory.

Definition at line 19 of file MLPfactory.cpp.

Referenced by makeNeuron().

```
{
  ConPtr conPtr(new Con(neuron, weight));
  return conPtr;
}
```

Here is the caller graph for this function:



5.26.3.3 ConContainerPtr MLPfactory::makeConContainer ( ) [protected, virtual]

Implements NeuralFactory.

Definition at line 26 of file MLPfactory.cpp.

```
{
  ConContainerPtr conContainerPtr(new SimpleContainer<ConPtr> );
  return conContainerPtr;
}
```

5.26.3.4 NeuronPtr MLPfactory::makeNeuron (Handler *Id*) [protected, virtual]

Implements NeuralFactory.

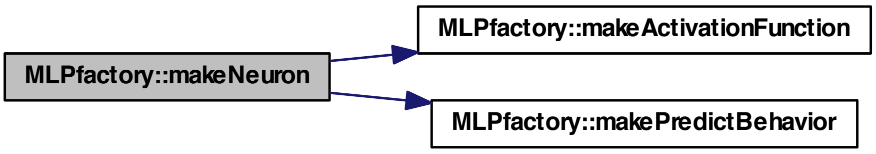
Definition at line 40 of file MLPfactory.cpp.

References makeActivationFunction(), and makePredictBehavior().

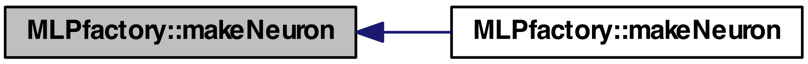
Referenced by makeNeuron().

```
{
  NeuronPtr neuronPtr(new SimpleNeuron(*this));
  neuronPtr->setId(Id);
  neuronPtr->setPredictBehavior(makePredictBehavior(neuronPtr));
  neuronPtr->setActivationFunction(makeActivationFunction(neuronPtr));
  return neuronPtr;
}
```

Here is the call graph for this function:


MLPfactory::makeNeuron
MLPfactory::makeActivationFunction
MLPfactory::makePredictBehavior

Here is the caller graph for this function:



5.26.3.5 NeuronPtr MLPfactory::makeNeuron (Handler *Id*, NeuronIteratorPtr *neuronIteratorPtr*, double *totalAmountOfParameters*) `[protected, virtual]`

Implements NeuralFactory.

Definition at line 50 of file MLPfactory.cpp.

References MLPbehavior::d_bias, makeCon(), and makeNeuron().

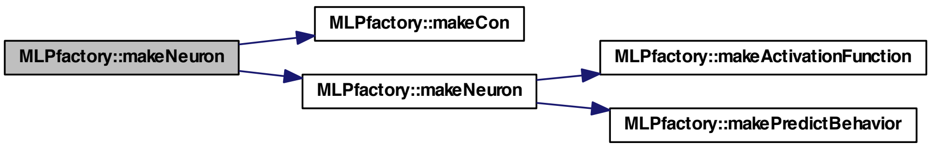
```
{
  RNGScope scope;

  NeuronPtr neuronPtr(makeNeuron(Id));

  double extreme = sqrt(3 / totalAmountOfParameters);
  double weight;
  for (neuronIteratorPtr->first(); !neuronIteratorPtr->isDone(); neuronIteratorPt
      r->next())
    {
      weight =as<double>(runif(1, -extreme, extreme));
      neuronPtr->addCon(makeCon(*neuronIteratorPtr->currentItem(), weight));
    }
  MLPbehavior* mlpBehavior = dynamic_cast<MLPbehavior*>(neuronPtr->d_predictBehav
      ior.get()) ;
  mlpBehavior->d_bias=as<double>(runif(1, -extreme, extreme));

return neuronPtr;
}
```

Here is the call graph for this function:


MLPfactory::makeNeuron
MLPfactory::makeCon
MLPfactory::makeNeuron
MLPfactory::makeActivationFunction
MLPfactory::makePredictBehavior

5.26.3.6 NeuronContainerPtr MLPfactory::makeNeuronContainer ( ) [protected, virtual]

Implements NeuralFactory.

Definition at line 72 of file MLPfactory.cpp.

```
{
  NeuronContainerPtr neuronContainerPtr(new SimpleContainer<NeuronPtr> );
  return neuronContainerPtr;
}
```

5.26.3.7 PredictBehaviorPtr MLPfactory::makePredictBehavior (NeuronPtr *neuronPtr*) [protected, virtual]

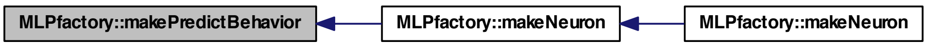
Implements NeuralFactory.

Definition at line 33 of file MLPfactory.cpp.

Referenced by makeNeuron().

```
{
  PredictBehaviorPtr predictBehaviorPtr(new MLPbehavior(neuronPtr));
  return predictBehaviorPtr;
}
```

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

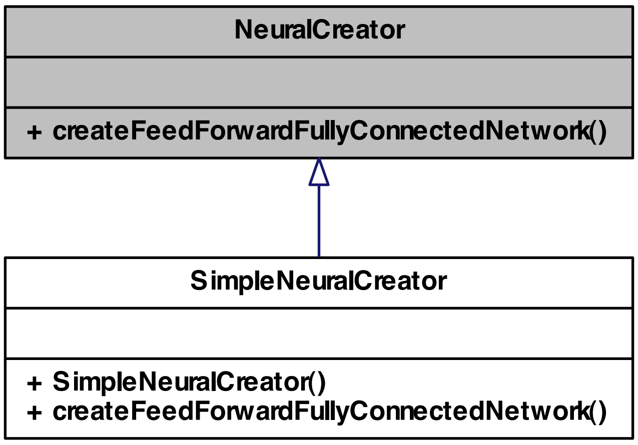
- pkg/AMORE/src/dia/MLPfactory.h
- pkg/AMORE/src/MLPfactory.cpp

5.27 NeuralCreator Class Reference

class NeuralCreator -

```
#include <NeuralCreator.h>
```

Inheritance diagram for NeuralCreator:


NeuralCreator
+ createFeedForwardFullyConnectedNetwork()
SimpleNeuralCreator
+ SimpleNeuralCreator()
+ createFeedForwardFullyConnectedNetwork()

Public Member Functions

- virtual NeuralNetworkPtr createFeedForwardFullyConnectedNetwork (NeuralFactoryPtr neuralFactoryPtr)=0

5.27.1 Detailed Description

class NeuralCreator -

Definition at line 4 of file NeuralCreator.h.

5.27.2 Member Function Documentation

5.27.2.1 virtual **NeuralNetworkPtr** NeuralCreator::createFeedForwardFullyConnectedNetwork (**NeuralFactoryPtr** *neuralFactoryPtr*) `[pure virtual]`

Implemented in SimpleNeuralCreator.

The documentation for this class was generated from the following file:

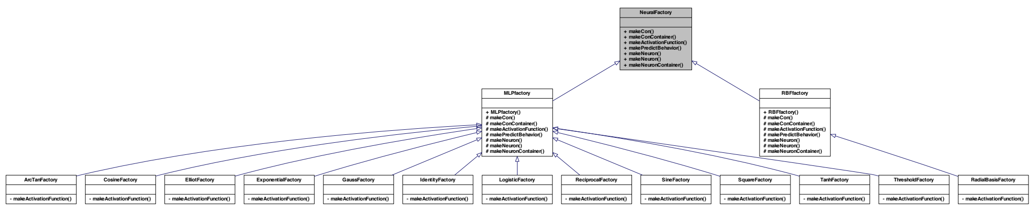
- pkg/AMORE/src/dia/NeuralCreator.h

5.28 NeuralFactory Class Reference

class NeuralFactory -

`#include <NeuralFactory.h>`

Inheritance diagram for NeuralFactory:



Public Member Functions

- virtual ConPtr makeCon (Neuron &neuron, double weight)=0
- virtual ConContainerPtr makeConContainer ()=0
- virtual ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)=0
- virtual PredictBehaviorPtr makePredictBehavior (NeuronPtr neuronPtr)=0
- virtual NeuronPtr makeNeuron (Handler Id)=0

- virtual NeuronPtr makeNeuron (Handler Id, NeuronIteratorPtr neuronIteratorPtr, double totalAmountOfParameters)=0
- virtual NeuronContainerPtr makeNeuronContainer ()=0

5.28.1 Detailed Description

class NeuralFactory -

Definition at line 4 of file NeuralFactory.h.

5.28.2 Member Function Documentation

5.28.2.1 virtual **ActivationFunctionPtr** NeuralFactory::makeActivationFunction (**NeuronPtr** *neuronPtr*) `[pure virtual]`

Implemented in ArcTanFactory, CosineFactory, ElliotFactory, ExponentialFactory, GaussFactory, IdentityFactory, LogisticFactory, MLPfactory, RadialBasisFactory, RBFfactory, ReciprocalFactory, SineFactory, SquareFactory, TanhFactory, and ThresholdFactory.

5.28.2.2 virtual **ConPtr** NeuralFactory::makeCon (**Neuron** & *neuron*, double *weight*) `[pure virtual]`

Implemented in MLPfactory.

5.28.2.3 virtual **ConContainerPtr** NeuralFactory::makeConContainer ( ) `[pure virtual]`

Implemented in MLPfactory, and RBFfactory.

Referenced by Neuron::Neuron().

Here is the caller graph for this function:

NeuralFactory::makeConContainer
Neuron::Neuron

5.28.2.4 virtual **NeuronPtr** NeuralFactory::makeNeuron (**Handler** *Id*) `[pure virtual]`

Implemented in MLPfactory, and RBFfactory.

5.28.2.5 virtual **NeuronPtr** NeuralFactory::makeNeuron (**Handler** *Id*, **NeuronIteratorPtr** *neuronIteratorPtr*, double *totalAmountOfParameters*) `[pure virtual]`

Implemented in MLPfactory, and RBFfactory.

5.28.2.6 virtual **NeuronContainerPtr** NeuralFactory::makeNeuronContainer ( ) `[pure virtual]`

Implemented in MLPfactory, and RBFfactory.

5.28.2.7 virtual **PredictBehaviorPtr** NeuralFactory::makePredictBehavior (**NeuronPtr** *neuronPtr*) `[pure virtual]`

Implemented in MLPfactory.

The documentation for this class was generated from the following file:

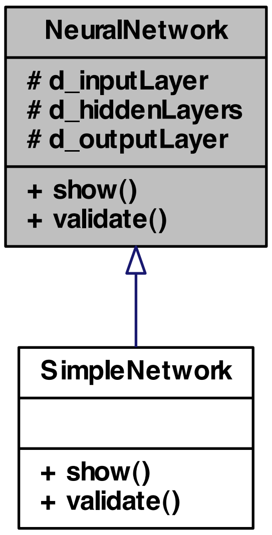
- pkg/AMORE/src/dia/NeuralFactory.h

5.29 NeuralNetwork Class Reference

class NeuralNetwork -

`#include <NeuralNetwork.h>`

Inheritance diagram for NeuralNetwork:



Public Member Functions

- void show ()
- bool validate ()

Protected Attributes

- LayerPtr d_inputLayer
- boost::shared_ptr< Container< LayerPtr > > d_hiddenLayers
- LayerPtr d_outputLayer

5.29.1 Detailed Description

class NeuralNetwork -

Definition at line 3 of file NeuralNetwork.h.

5.29.2 Member Function Documentation

5.29.2.1 void NeuralNetwork::show ( )

Reimplemented in SimpleNetwork.

5.29.2.2 bool NeuralNetwork::validate ( )

Reimplemented in SimpleNetwork.

5.29.3 Member Data Documentation

5.29.3.1 boost::shared_ptr< Container< LayerPtr > > NeuralNetwork::d_hiddenLayers `[protected]`

Definition at line 7 of file NeuralNetwork.h.

5.29.3.2 LayerPtr NeuralNetwork::d_inputLayer `[protected]`

Definition at line 6 of file NeuralNetwork.h.

5.29.3.3 LayerPtr NeuralNetwork::d_outputLayer `[protected]`

Definition at line 8 of file NeuralNetwork.h.

The documentation for this class was generated from the following file:

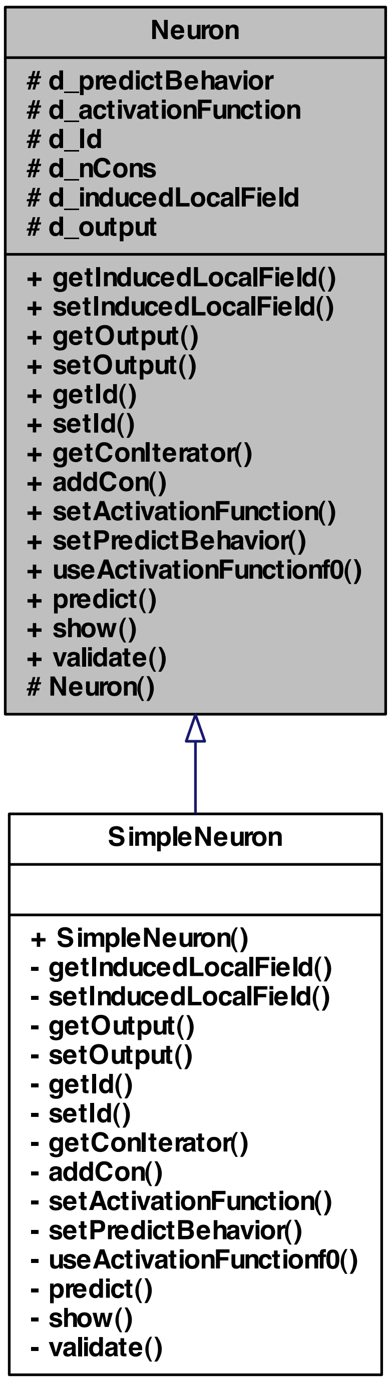
- pkg/AMORE/src/dia/NeuralNetwork.h

5.30 Neuron Class Reference

class Neuron -

`#include <Neuron.h>`

Inheritance diagram for Neuron:



Public Member Functions

- virtual double getInducedLocalField ()=0
- virtual void setInducedLocalField (double inducedLocalField)=0
- virtual double getOutput ()=0
- virtual void setOutput (double output)=0
- virtual Handler getId ()=0
- virtual void setId (Handler Id)=0
- virtual ConIteratorPtr getConIterator ()=0
- virtual void addCon (ConPtr conPtr)=0
- virtual void setActivationFunction (ActivationFunctionPtr activationFunctionPtr)=0
- virtual void setPredictBehavior (PredictBehaviorPtr predictBehaviorPtr)=0
- virtual double useActivationFunctionf0 ()=0
- virtual void predict ()=0
- virtual void show ()=0
- virtual bool validate ()=0

Protected Member Functions

- Neuron (NeuralFactory &neuralFactory)

Protected Attributes

- PredictBehaviorPtr d_predictBehavior
- ActivationFunctionPtr d_activationFunction
- Handler d_Id
- ConContainerPtr d_nCons
- double d_inducedLocalField
- double d_output

Friends

- class MLPfactory

5.30.1 Detailed Description

class Neuron -

Definition at line 3 of file Neuron.h.

5.30.2 Constructor & Destructor Documentation

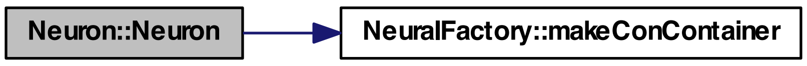
5.30.2.1 Neuron::Neuron (**NeuralFactory** & *neuralFactory*) [protected]

Definition at line 10 of file Neuron.cpp.

References d_nCons, and NeuralFactory::makeConContainer().

```
                                                            :
  d_Id(NA_INTEGER), d_inducedLocalField(0.0), d_output(0.0)
{
  d_nCons = neuralFactory.makeConContainer();
}
```

Here is the call graph for this function:



5.30.3 Member Function Documentation

5.30.3.1 virtual void Neuron::addCon (**ConPtr** *conPtr*) [pure virtual]

Implemented in SimpleNeuron.

5.30.3.2 virtual **ConIteratorPtr** Neuron::getConIterator ( ) [pure virtual]

Implemented in SimpleNeuron.

5.30.3.3 virtual **Handler** Neuron::getId ( ) [pure virtual]

Implemented in SimpleNeuron.

5.30.3.4 virtual double Neuron::getInducedLocalField ( ) [pure virtual]

Implemented in SimpleNeuron.

5.30.3.5 virtual double Neuron::getOutput ( ) [pure virtual]

Implemented in SimpleNeuron.

5.30.3.6 virtual void Neuron::predict ( ) [pure virtual]

Implemented in SimpleNeuron.

5.30.3.7 virtual void Neuron::setActivationFunction (ActivationFunctionPtr *activationFunctionPtr*) [pure virtual]

Implemented in SimpleNeuron.

5.30.3.8 virtual void Neuron::setId (Handler *Id*) [pure virtual]

Implemented in SimpleNeuron.

5.30.3.9 virtual void Neuron::setInducedLocalField (double *inducedLocalField*) [pure virtual]

Implemented in SimpleNeuron.

5.30.3.10 virtual void Neuron::setOutput (double *output*) [pure virtual]

Implemented in SimpleNeuron.

5.30.3.11 virtual void Neuron::setPredictBehavior (PredictBehaviorPtr *predictBehaviorPtr*) [pure virtual]

Implemented in SimpleNeuron.

5.30.3.12 virtual void Neuron::show ( ) [pure virtual]

Implemented in SimpleNeuron.

5.30.3.13 virtual double Neuron::useActivationFunctionf0 ( ) [pure virtual]

Implemented in SimpleNeuron.

5.30.3.14 virtual bool Neuron::validate ( ) [pure virtual]

Implemented in SimpleNeuron.

5.30.4 Friends And Related Function Documentation

5.30.4.1 friend class **MLPfactory** `[friend]`

Definition at line 15 of file Neuron.h.

5.30.5 Member Data Documentation

5.30.5.1 **ActivationFunctionPtr Neuron::d_activationFunction** `[protected]`

Definition at line 7 of file Neuron.h.

Referenced by SimpleNeuron::setActivationFunction(), and SimpleNeuron::useActivationFunctionf0().

5.30.5.2 **Handler Neuron::d_Id** `[protected]`

Definition at line 9 of file Neuron.h.

Referenced by SimpleNeuron::getId(), and SimpleNeuron::setId().

5.30.5.3 double **Neuron::d_inducedLocalField** `[protected]`

Definition at line 11 of file Neuron.h.

Referenced by SimpleNeuron::getInducedLocalField(), and SimpleNeuron::setInducedLocalField().

5.30.5.4 **ConContainerPtr Neuron::d_nCons** `[protected]`

Definition at line 10 of file Neuron.h.

Referenced by SimpleNeuron::addCon(), SimpleNeuron::getConIterator(), Neuron(), and SimpleNeuron::show().

5.30.5.5 double **Neuron::d_output** `[protected]`

Definition at line 12 of file Neuron.h.

Referenced by SimpleNeuron::getOutput(), SimpleNeuron::setOutput(), and SimpleNeuron::show().

5.30.5.6 **PredictBehaviorPtr Neuron::d_predictBehavior** `[protected]`

Definition at line 6 of file Neuron.h.

Referenced by SimpleNeuron::predict(), SimpleNeuron::setPredictBehavior(), and SimpleNeuron::show().

The documentation for this class was generated from the following files:

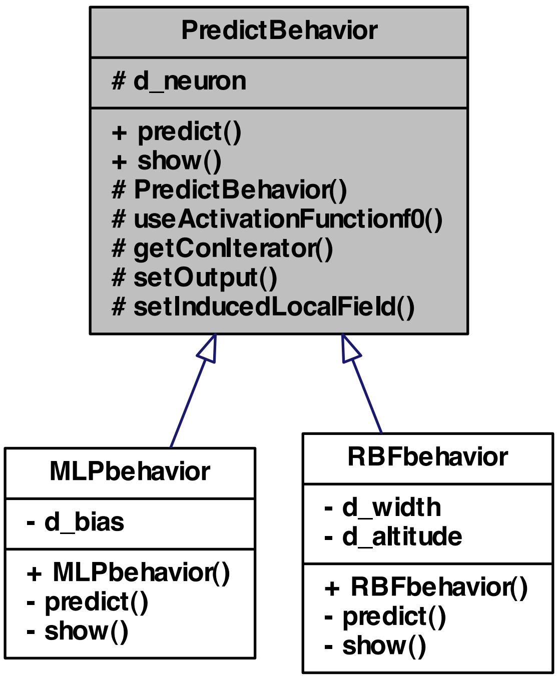
- pkg/AMORE/src/dia/Neuron.h
- pkg/AMORE/src/Neuron.cpp

5.31 PredictBehavior Class Reference

class PredictBehavior -

```
#include <PredictBehavior.h>
```

Inheritance diagram for PredictBehavior:



Public Member Functions

- virtual void predict ()=0
- virtual void show ()=0

Protected Member Functions

- PredictBehavior (NeuronPtr neuronPtr)
- double useActivationFunctionf0 ()
- ConIteratorPtr getConIterator ()
- void setOutput (double output)
- void setInducedLocalField (double inducedLocalField)

Protected Attributes

- NeuronWeakPtr d_neuron

5.31.1 Detailed Description

class PredictBehavior -

Definition at line 4 of file PredictBehavior.h.

5.31.2 Constructor & Destructor Documentation

5.31.2.1 PredictBehavior::PredictBehavior (**NeuronPtr** *neuronPtr*) [protected]

Definition at line 11 of file PredictBehavior.cpp.

```
                                                  :
  d_neuron(neuronPtr)
{
}
```

5.31.3 Member Function Documentation

5.31.3.1 ConIteratorPtr PredictBehavior::getConIterator ( ) [protected]

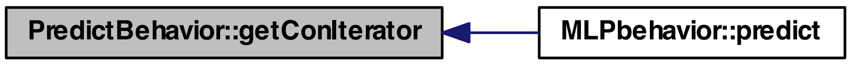
Definition at line 25 of file PredictBehavior.cpp.

References d_neuron.

Referenced by MLPbehavior::predict().

```
{
  NeuronPtr neuronPtr( d_neuron.lock() ) ;
  return neuronPtr->getConIterator();
}
```

Here is the caller graph for this function:



5.31.3.2 virtual void PredictBehavior::predict ( ) [pure virtual]

Implemented in MLPbehavior, and RBFbehavior.

5.31.3.3 void PredictBehavior::setInducedLocalField (double *inducedLocalField*) [protected]

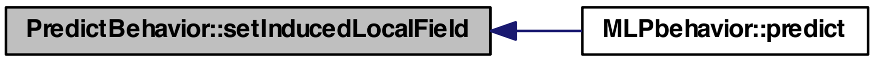
Definition at line 39 of file PredictBehavior.cpp.

References d_neuron.

Referenced by MLPbehavior::predict().

```
{
  NeuronPtr neuronPtr( d_neuron.lock() ) ;
  return neuronPtr->setInducedLocalField(inducedLocalField);
}
```

Here is the caller graph for this function:



5.31.3.4 void PredictBehavior::setOutput (double *output*) [protected]

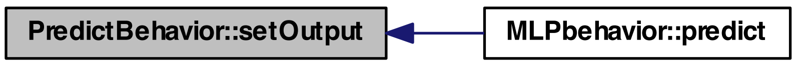
Definition at line 32 of file PredictBehavior.cpp.

References d_neuron.

Referenced by MLPbehavior::predict().

```
{
  NeuronPtr neuronPtr( d_neuron.lock() ) ;
  return neuronPtr->setOutput(output);
}
```

Here is the caller graph for this function:



5.31.3.5 virtual void PredictBehavior::show ( ) [pure virtual]

Implemented in MLPbehavior, and RBFbehavior.

5.31.3.6 double PredictBehavior::useActivationFunctionf0 ( ) [protected]

Definition at line 17 of file PredictBehavior.cpp.

References d_neuron.

Referenced by MLPbehavior::predict().

```
{
  NeuronPtr neuronPtr( d_neuron.lock() ) ;
  return neuronPtr->useActivationFunctionf0();

}
```

Here is the caller graph for this function:

PredictBehavior::useActivationFunctionf0
MLPbehavior::predict

5.31.4 Member Data Documentation

5.31.4.1 NeuronWeakPtr PredictBehavior::d_neuron [protected]

Definition at line 7 of file PredictBehavior.h.

Referenced by getConIterator(), setInducedLocalField(), setOutput(), and useActivationFunctionf0().

The documentation for this class was generated from the following files:

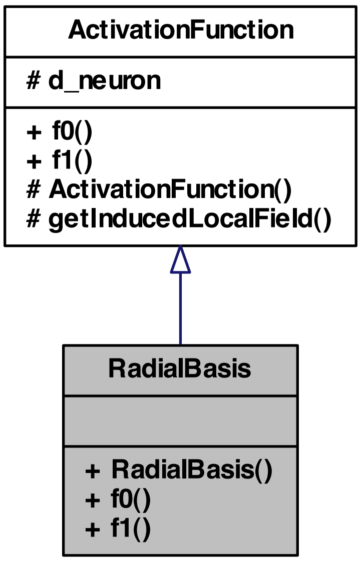
- pkg/AMORE/src/dia/PredictBehavior.h
- pkg/AMORE/src/PredictBehavior.cpp

5.32 RadialBasis Class Reference

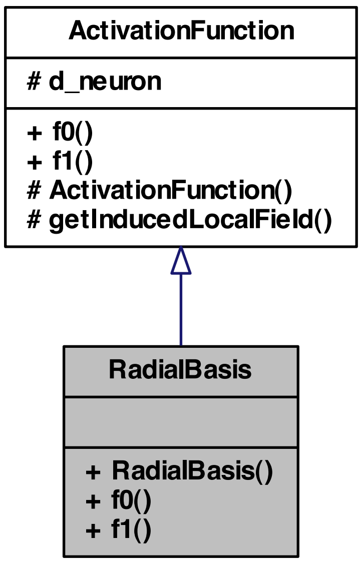
class RadialBasis -

`#include <RadialBasis.h>`

Inheritance diagram for RadialBasis:



Collaboration diagram for RadialBasis:



Public Member Functions

- RadialBasis (NeuronPtr neuronPtr)
- double f0 ()
- double f1 ()

5.32.1 Detailed Description

class RadialBasis -

Definition at line 5 of file RadialBasis.h.

5.32.2 Constructor & Destructor Documentation

5.32.2.1 RadialBasis::RadialBasis (NeuronPtr *neuronPtr*)

5.32.3 Member Function Documentation

5.32.3.1 double RadialBasis::f0 ( ) `[virtual]`

Implements ActivationFunction.

5.32.3.2 double RadialBasis::f1 ( ) `[virtual]`

Implements ActivationFunction.

The documentation for this class was generated from the following file:

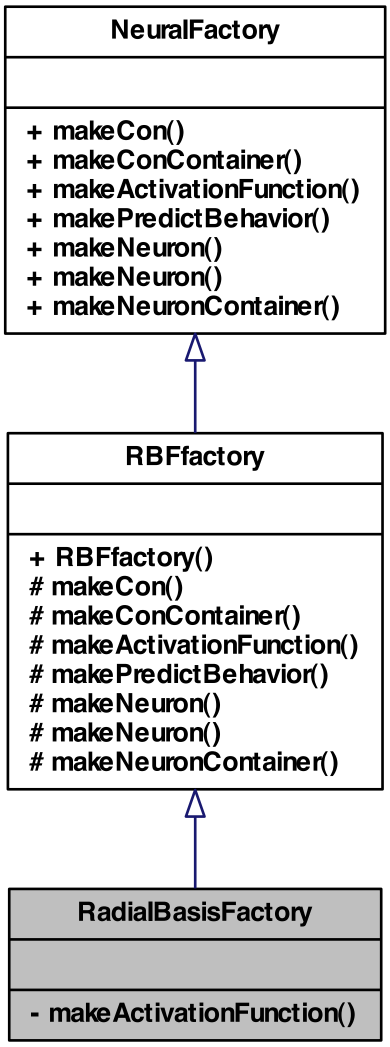
- pkg/AMORE/src/dia/RadialBasis.h

5.33 RadialBasisFactory Class Reference

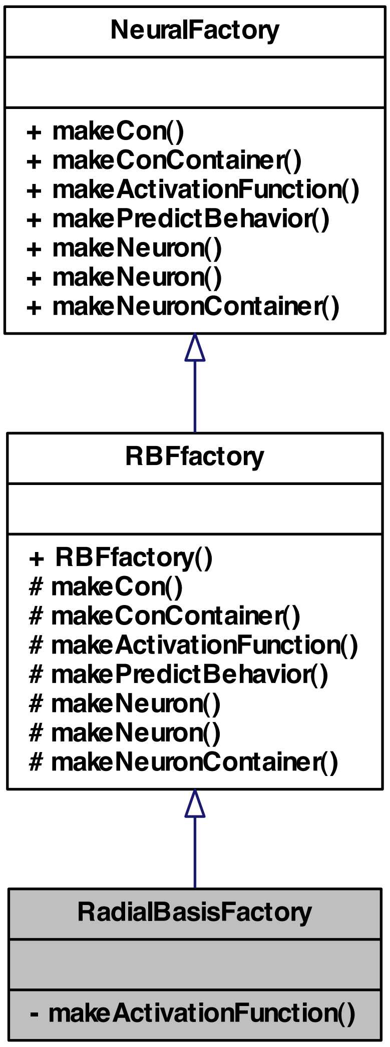
class RadialBasisFactory -

`#include <RadialBasisFactory.h>`

Inheritance diagram for RadialBasisFactory:



Collaboration diagram for RadialBasisFactory:



Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

5.33.1 Detailed Description

class RadialBasisFactory -

Definition at line 5 of file RadialBasisFactory.h.

5.33.2 Member Function Documentation

5.33.2.1 ActivationFunctionPtr RadialBasisFactory::makeActivationFunction (NeuronPtr *neuronPtr*) `[private, virtual]`

Implements RBFfactory.

The documentation for this class was generated from the following file:

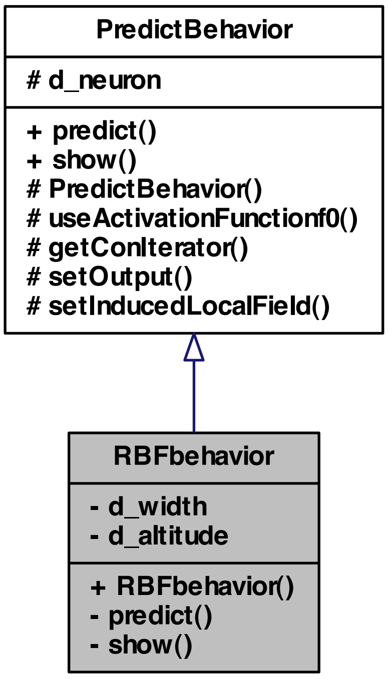
- pkg/AMORE/src/dia/RadialBasisFactory.h

5.34 RBFbehavior Class Reference

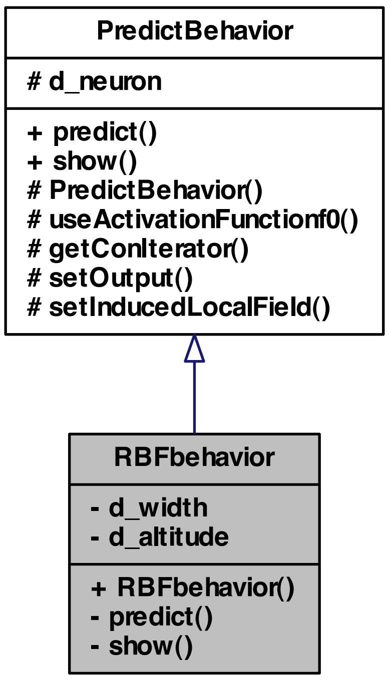
class RBFbehavior -

`#include <RBFbehavior.h>`

Inheritance diagram for RBFbehavior:



Collaboration diagram for RBFbehavior:



Public Member Functions

- RBFbehavior (NeuronPtr neuronPtr)

Private Member Functions

- void predict ()
- void show ()

Private Attributes

- double d_width
- double d_altitude

5.34.1 Detailed Description

class RBFbehavior -

Definition at line 5 of file RBFbehavior.h.

5.34.2 Constructor & Destructor Documentation

5.34.2.1 RBFbehavior::RBFbehavior (NeuronPtr *neuronPtr*)

5.34.3 Member Function Documentation

5.34.3.1 void RBFbehavior::predict ( ) `[private, virtual]`

Implements PredictBehavior.

5.34.3.2 void RBFbehavior::show ( ) `[private, virtual]`

Implements PredictBehavior.

5.34.4 Member Data Documentation

5.34.4.1 double RBFbehavior::d_altitude `[private]`

Definition at line 9 of file RBFbehavior.h.

5.34.4.2 double RBFbehavior::d_width `[private]`

Definition at line 8 of file RBFbehavior.h.

The documentation for this class was generated from the following file:

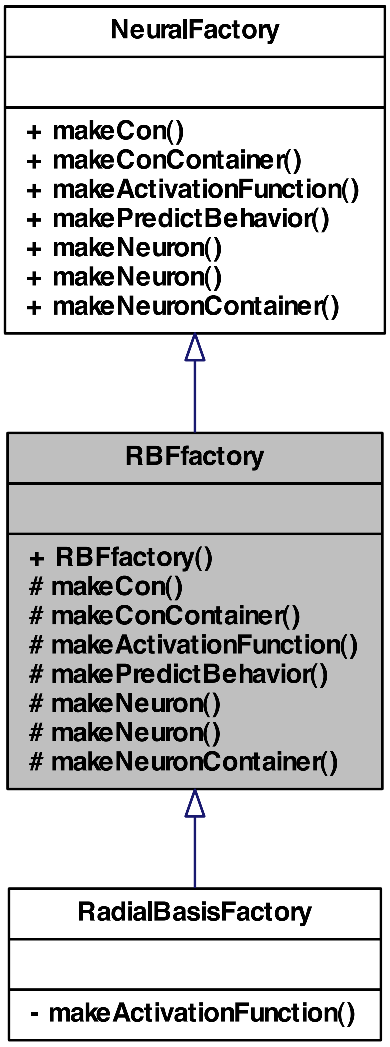
- pkg/AMORE/src/dia/RBFbehavior.h

5.35 RBFfactory Class Reference

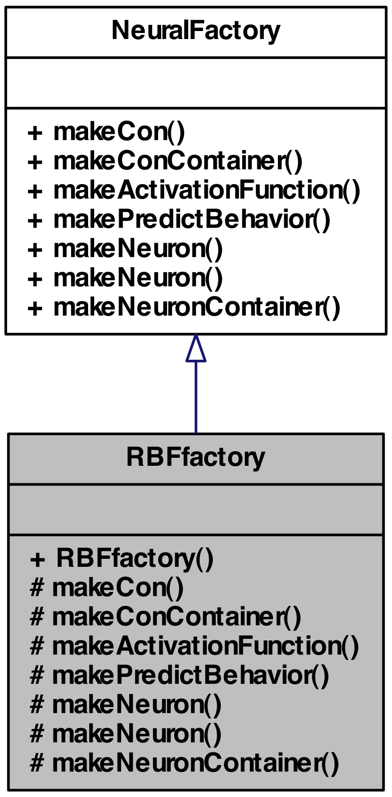
class RBFfactory -

`#include <RBFfactory.h>`

Inheritance diagram for RBFfactory:



Collaboration diagram for RBFfactory:



Public Member Functions

- RBFfactory ()

Protected Member Functions

- ConPtr makeCon (Neuron ∗neuron, double weight)
- ConContainerPtr makeConContainer ()
- virtual ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)=0
- PredictBehaviorPtr makePredictBehavior ()
- NeuronPtr makeNeuron (Handler Id)
- NeuronPtr makeNeuron (Handler Id, NeuronIteratorPtr neuronIteratorPtr, double totalAmountOfParameters)

- NeuronContainerPtr makeNeuronContainer ()

5.35.1 Detailed Description

class RBFfactory -

Definition at line 5 of file RBFfactory.h.

5.35.2 Constructor & Destructor Documentation

5.35.2.1 RBFfactory::RBFfactory ( )

5.35.3 Member Function Documentation

5.35.3.1 virtual **ActivationFunctionPtr** RBFfactory::makeActivationFunction (**NeuronPtr** *neuronPtr*) `[protected, pure virtual]`

Implements NeuralFactory.

Implemented in RadialBasisFactory.

5.35.3.2 **ConPtr** RBFfactory::makeCon (**Neuron** ∗ *neuron*, double *weight*) `[protected]`

5.35.3.3 **ConContainerPtr** RBFfactory::makeConContainer ( ) `[protected, virtual]`

Implements NeuralFactory.

5.35.3.4 **NeuronPtr** RBFfactory::makeNeuron (**Handler** *Id*) `[protected, virtual]`

Implements NeuralFactory.

5.35.3.5 **NeuronPtr** RBFfactory::makeNeuron (**Handler** *Id*, **NeuronIteratorPtr** *neuronIteratorPtr*, double *totalAmountOfParameters*) `[protected, virtual]`

Implements NeuralFactory.

5.35.3.6 **NeuronContainerPtr** RBFfactory::makeNeuronContainer ( ) `[protected, virtual]`

Implements NeuralFactory.

5.35.3.7 PredictBehaviorPtr RBFfactory::makePredictBehavior ( ) [protected]

The documentation for this class was generated from the following file:

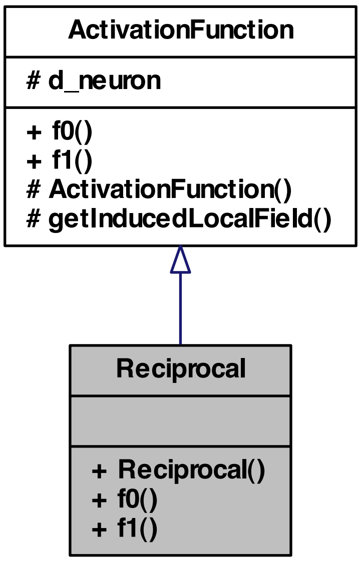
- pkg/AMORE/src/dia/RBFfactory.h

5.36 Reciprocal Class Reference

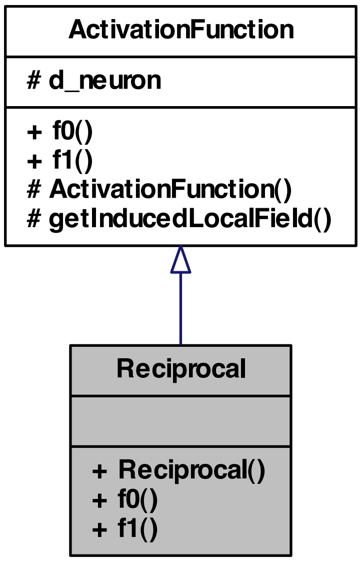
class Reciprocal -

`#include <Reciprocal.h>`

Inheritance diagram for Reciprocal:



Collaboration diagram for Reciprocal:



Public Member Functions

- Reciprocal (NeuronPtr neuronPtr)
- void f0 ()
- void f1 ()

5.36.1 Detailed Description

class Reciprocal -

Definition at line 5 of file Reciprocal.h.

5.36.2 Constructor & Destructor Documentation

5.36.2.1 Reciprocal::Reciprocal (**NeuronPtr** *neuronPtr*)

5.36.3 Member Function Documentation

5.36.3.1 void Reciprocal::f0 ( ) [virtual]

Implements ActivationFunction.

5.36.3.2 void Reciprocal::f1 ( ) [virtual]

Implements ActivationFunction.

The documentation for this class was generated from the following file:

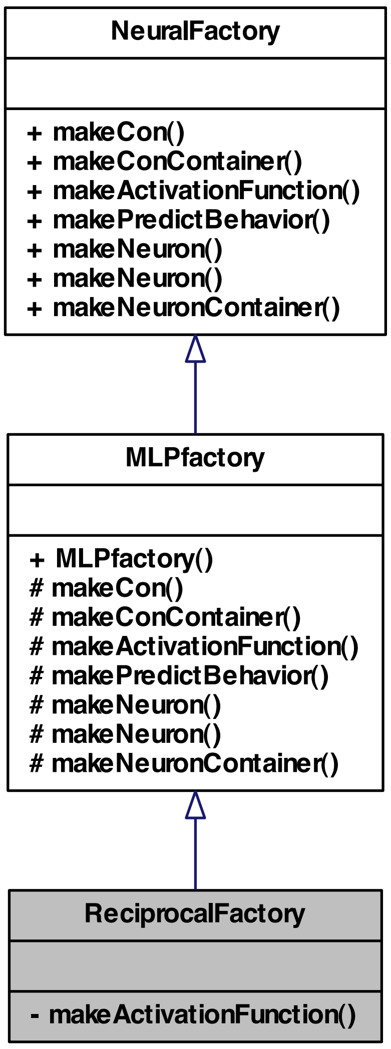
- pkg/AMORE/src/dia/Reciprocal.h

5.37 ReciprocalFactory Class Reference

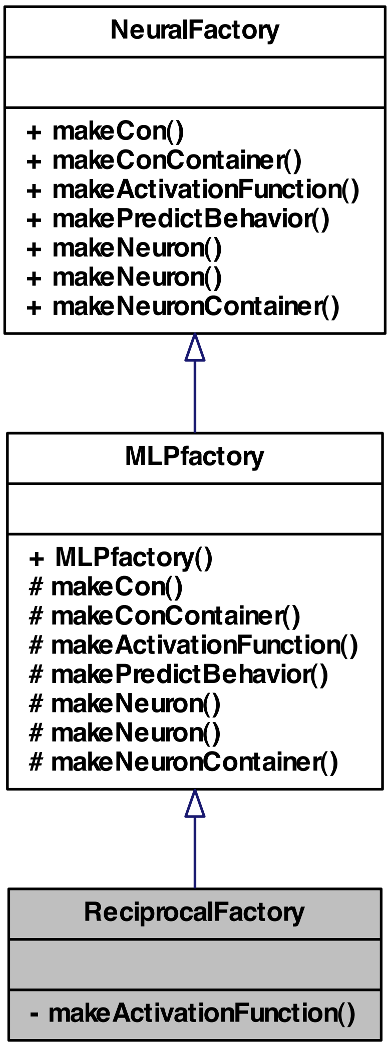
class ReciprocalFactory -

`#include <ReciprocalFactory.h>`

Inheritance diagram for ReciprocalFactory:



Collaboration diagram for ReciprocalFactory:



Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

5.37.1 Detailed Description

class ReciprocalFactory -

Definition at line 5 of file ReciprocalFactory.h.

5.37.2 Member Function Documentation

5.37.2.1 ActivationFunctionPtr ReciprocalFactory::makeActivationFunction (NeuronPtr *neuronPtr*) `[private, virtual]`

Implements MLPfactory.

The documentation for this class was generated from the following file:

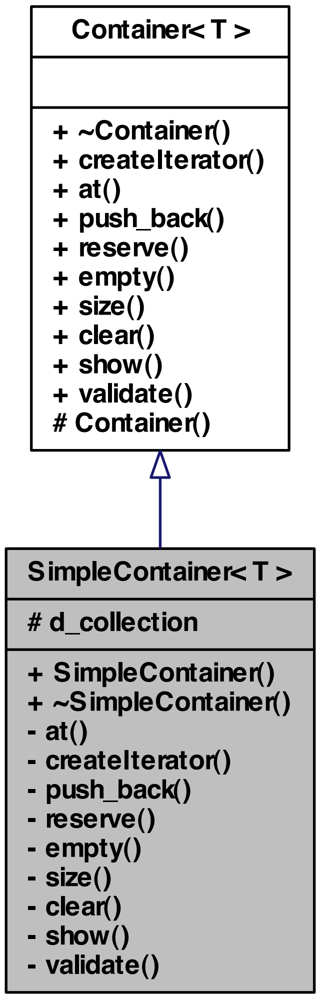
- pkg/AMORE/src/dia/ReciprocalFactory.h

5.38 SimpleContainer< T > Class Template Reference

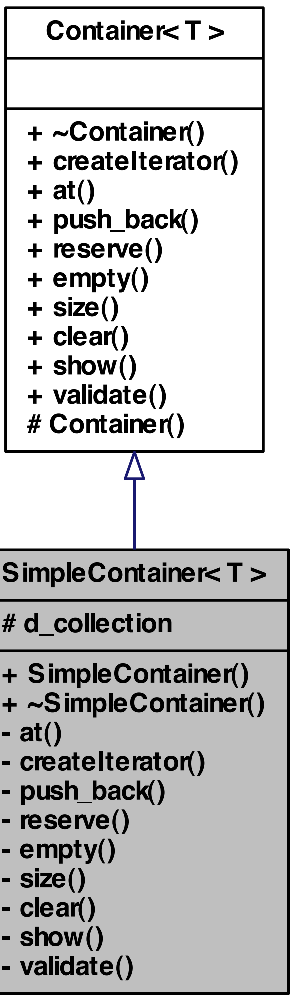
class SimpleContainer -

`#include <SimpleContainer.h>`

Inheritance diagram for SimpleContainer< T >:



Collaboration diagram for SimpleContainer< T >:



Public Member Functions

- SimpleContainer ()
- ∼SimpleContainer ()

Protected Attributes

- std::vector< T > d_collection

Private Member Functions

- T at (size_type element)

    Append a shared_ptr at the end of collection.
- boost::shared_ptr< Iterator< T > > createIterator ()
- void push_back (T const &const_reference)
- void reserve (int n)
- bool empty ()
- size_type size ()

    Returns the size or length of the vector.
- void clear ()
- void show ()

    Pretty print of the SimpleContainer<T>
- bool validate ()

    Object validator.

Friends

- class SimpleContainerIterator< T >

5.38.1 Detailed Description

template<typename T>class SimpleContainer< T >

class SimpleContainer -

Definition at line 6 of file SimpleContainer.h.

5.38.2 Constructor & Destructor Documentation

5.38.2.1 template<typename T > SimpleContainer< T >::SimpleContainer ( )

Definition at line 11 of file SimpleContainer.cpp.

```
{

}
```

5.38.2.2 template<typename T > SimpleContainer< T >::~SimpleContainer ( )

Definition at line 17 of file SimpleContainer.cpp.

```
{
}
```

5.38.3 Member Function Documentation

5.38.3.1 template<typename T > T SimpleContainer< T >::at (size_type *element*) [private, virtual]

Append a shared_ptr at the end of collection.

Implements push_back for the Container class

Parameters

TsharedPtr	A shared_ptr pointer to be inserted at the end of collection

```
                //================
                //Usage example:
                //================
                // Data set up
                        Neuron N1, N2, N3;
                        Container<Con> conContainer;
                        std::vector<ConPtr> vc;
                        std::vector<int> result;
                        N1.setId(10);
                        N2.setId(20);
                        N3.setId(30);
                // Test
                        ConPtr ptCon( new Con(&N1, 1.13) );     // Create new Con
 and initialize ptCon
                        conContainer.push_back(ptCon);                         /
/ push_back
                        ptCon.reset(  new Con(&N2, 2.22) );             // create
 new Con and assign to ptCon
                        conContainer.push_back(ptCon);                         /
/ push_back
                        ptCon.reset(  new Con(&N3, 3.33) );             // create
 new Con and assign to ptCon
                        conContainer.push_back(ptCon);                         /
/ push_back

                        vc = conContainer.load();

                        result.push_back(vc.at(0)->getId());
                        result.push_back(vc.at(1)->getId());
                        result.push_back(vc.at(2)->getId());
  // After execution of this code, result contains a numeric vector with va
lues 10, 20 and 30.
```

See also

C++ documentation for std::vector::push_back and the unit test files, e.g., runit.Cpp.Container.R, for usage examples.

Implements Container< T >.

Definition at line 69 of file SimpleContainer.cpp.

```
{
return d_collection.at(element);
}
```

5.38.3.2 template<typename T > void SimpleContainer< T >::clear ( ) [private, virtual]

Implements Container< T >.

Definition at line 182 of file SimpleContainer.cpp.

```
{
d_collection.clear();
}
```

5.38.3.3 template<typename T > boost::shared_ptr< Iterator< T > > SimpleContainer< T >::createIterator ( ) [private, virtual]

Implements Container< T >.

Definition at line 23 of file SimpleContainer.cpp.

```
{
  boost::shared_ptr < SimpleContainerIterator<T> > iteratorPtr(  new
    SimpleContainerIterator<T> ());
  iteratorPtr->d_container = this;
  iteratorPtr->d_current= 0;
  return iteratorPtr;
}
```

5.38.3.4 template<typename T > bool SimpleContainer< T >::empty ( ) [private, virtual]

Implements Container< T >.

Definition at line 168 of file SimpleContainer.cpp.

```
{
  return (d_collection.empty());
}
```

5.38.3.5 template<typename T > void SimpleContainer< T >::push_back (T const & *const_reference*) [private, virtual]

Implements Container< T >.

Definition at line 77 of file SimpleContainer.cpp.

```
{
d_collection.push_back(reference);
}
```

5.38.3.6 template<typename T > void SimpleContainer< T >::reserve (int *n*) [private, virtual]

Implements Container< T >.

Definition at line 175 of file SimpleContainer.cpp.

```
{
d_collection.reserve(n);
}
```

5.38.3.7 template<typename T > void SimpleContainer< T >::show ( ) [private, virtual]

Pretty print of the SimpleContainer<T>

This method outputs in the R terminal the contents of Container::collection.

Returns

true in case everything works without throwing an exception

*

```
                //================
                //Usage example:
                //================
                // Data set up
                        ContainerNeuronPtr      neuronContainerPtr( new
      Container<Neuron>() );
                        ContainerConPtr conContainerPtr( new Container<Con>() );
                        ConPtr  ptC;
                        NeuronPtr ptN;
                        int ids[]= {10, 20, 30};
                        double weights[] = {1.13, 2.22, 3.33 };

                        for (int i=0; i<=2 ; i++) {                            /
      / Let's create a vector with three neurons
                                ptN.reset( new Neuron( ids[i] ) );
                                neuronContainerPtr->push_back(ptN);
                        }
```

```
                    for (int i=0; i<=2 ; i++) {                              /
    / and a vector with three connections
                            ptC.reset( new Con( neuronContainerPtr->load().at
    (i), weights[i]) );
                            conContainerPtr->push_back(ptC);
                    }

            // Test
                    conContainerPtr->show() ;

            // The output at the R terminal would display:
            //
            //      # From:  10      Weight=         1.130000
            //      # From:  20      Weight=         2.220000
            //      # From:  30      Weight=         3.330000
            //
```

See also

The unit test files, e.g., runit.Cpp.Container.R, for usage examples.

Implements Container< T >.

Definition at line 127 of file SimpleContainer.cpp.

```
{
    boost::shared_ptr< Iterator <T> >  itr = createIterator();
    for ( itr->first(); !itr->isDone(); itr->next() ) {
       itr->currentItem()->show();
    }
}
```

5.38.3.8 template< typename T > size_type SimpleContainer< T >::size ( ) [private, virtual]

Returns the size or length of the vector.

This method returns the size of the vector. In the classes derived from SimpleContainer<T> this is aliased as numOfCons, numOfNeurons and numOfLayers. The unit test files, e.g., runit.Cpp.Container.R, for usage examples.

Implements Container< T >.

Definition at line 160 of file SimpleContainer.cpp.

```
{
  return d_collection.size();
}
```

5.38.3.9 template< typename T > bool SimpleContainer< T >::validate ( ) [private, virtual]

Object validator.

This method checks the object for internal coherence. This method calls the validate method for each element in collection,

See also

The unit test files, e.g., runit.Cpp.Container.R, for usage examples.

Implements Container< T >.

Definition at line 142 of file SimpleContainer.cpp.

```
{
   boost::shared_ptr< Iterator <T> >  itr = createIterator();
   for ( itr->first(); !itr->isDone(); itr->next() ) {
      itr->currentItem()->validate();
   }
return true;
}
```

5.38.4 Friends And Related Function Documentation

5.38.4.1 template<typename T > friend class SimpleContainerIterator< T > [friend]

Definition at line 12 of file SimpleContainer.h.

5.38.5 Member Data Documentation

5.38.5.1 template<typename T > std::vector< T > SimpleContainer< T >::d_collection [protected]

Definition at line 9 of file SimpleContainer.h.

The documentation for this class was generated from the following files:

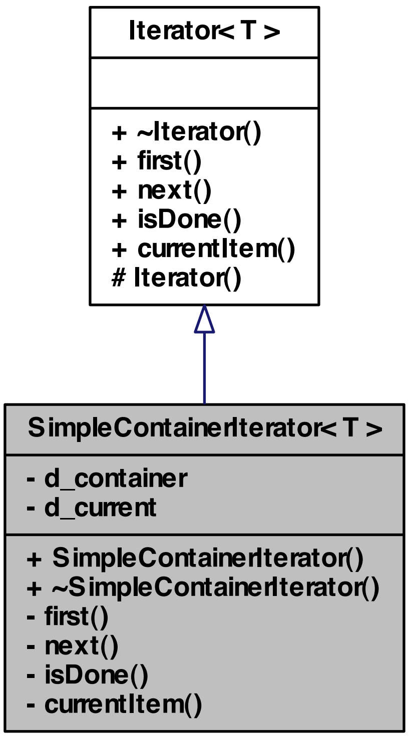
- pkg/AMORE/src/dia/SimpleContainer.h
- pkg/AMORE/src/SimpleContainer.cpp

5.39 SimpleContainerIterator< T > Class Template Reference

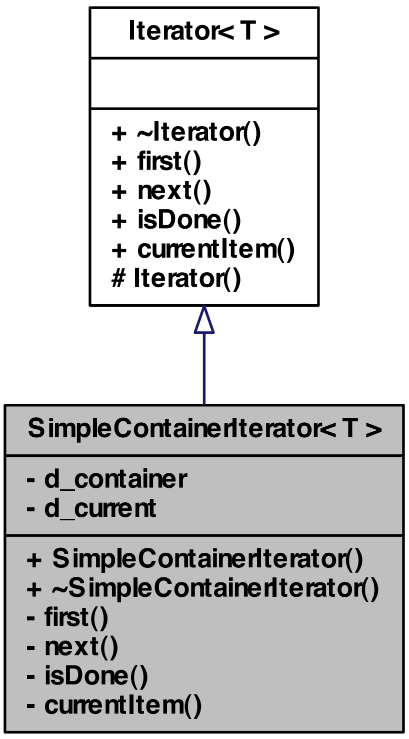
class SimpleContainerIterator -

`#include <SimpleContainerIterator.h>`

Inheritance diagram for SimpleContainerIterator< T >:



Collaboration diagram for SimpleContainerIterator< T >:



Public Member Functions

- SimpleContainerIterator ()
- ∼SimpleContainerIterator ()

Private Member Functions

- void first ()
- void next ()
- bool isDone ()
- T currentItem ()

Private Attributes

- Container< T > ∗ d_container
- size_type d_current

Friends

- class SimpleContainer< T >

5.39.1 Detailed Description

template<typename T>class SimpleContainerIterator< T >

class SimpleContainerIterator -

Definition at line 6 of file SimpleContainerIterator.h.

5.39.2 Constructor & Destructor Documentation

5.39.2.1 template<typename T > SimpleContainerIterator< T >::SimpleContainerIterator ( )

Definition at line 4 of file SimpleContainerIterator.cpp.

```
{
}
```

5.39.2.2 template<typename T > SimpleContainerIterator< T >::~SimpleContainerIterator ( )

Definition at line 9 of file SimpleContainerIterator.cpp.

```
{
}
```

5.39.3 Member Function Documentation

5.39.3.1 template<typename T > T SimpleContainerIterator< T >::currentItem ( ) [private, virtual]

Implements Iterator< T >.

Definition at line 37 of file SimpleContainerIterator.cpp.

```
{
    if (isDone()) throw std::range_error("SimpleContainerIterator::currentItem
     Error: IteratorOutOfBounds");
    return d_container->at(d_current);
}
```

5.39.3.2 template<typename T > void SimpleContainerIterator< T >::first ( ) [private, virtual]

Implements Iterator< T >.

Definition at line 15 of file SimpleContainerIterator.cpp.

```
{
  d_current = 0;
}
```

5.39.3.3 template<typename T > bool SimpleContainerIterator< T >::isDone ( ) [private, virtual]

Implements Iterator< T >.

Definition at line 29 of file SimpleContainerIterator.cpp.

```
{
  bool IteratorIsDone(d_current == d_container->size());
  return IteratorIsDone;
}
```

5.39.3.4 template<typename T > void SimpleContainerIterator< T >::next ( ) [private, virtual]

Implements Iterator< T >.

Definition at line 22 of file SimpleContainerIterator.cpp.

```
{
  ++d_current;
}
```

5.39.4 Friends And Related Function Documentation

5.39.4.1 template<typename T > friend class SimpleContainer< T > [friend]

Definition at line 13 of file SimpleContainerIterator.h.

5.39.5 Member Data Documentation

5.39.5.1 template<typename T > Container<T>* SimpleContainerIterator< T >::d_container [private]

Definition at line 9 of file SimpleContainerIterator.h.

5.39.5.2 template<typename T > size_type SimpleContainerIterator< T >::d_current [private]

Definition at line 10 of file SimpleContainerIterator.h.

The documentation for this class was generated from the following files:

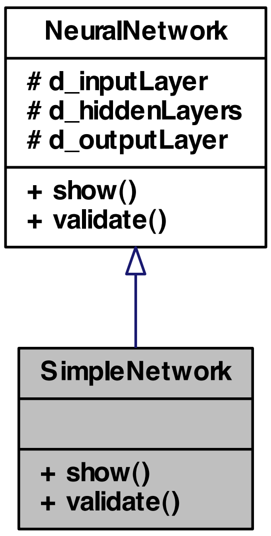
- pkg/AMORE/src/dia/SimpleContainerIterator.h
- pkg/AMORE/src/SimpleContainerIterator.cpp

5.40 SimpleNetwork Class Reference

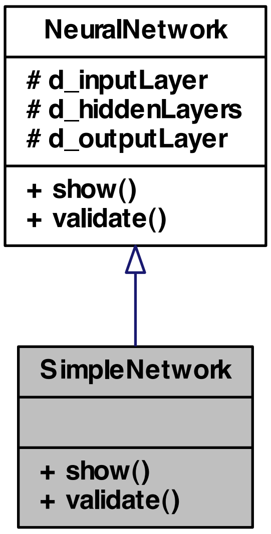
class SimpleNetwork -

`#include <SimpleNetwork.h>`

Inheritance diagram for SimpleNetwork:



Collaboration diagram for SimpleNetwork:



Public Member Functions

- void show ()
- bool validate ()

5.40.1 Detailed Description

class SimpleNetwork -

Definition at line 5 of file SimpleNetwork.h.

5.40.2 Member Function Documentation

5.40.2.1 void SimpleNetwork::show ( )

Reimplemented from NeuralNetwork.

5.40.2.2 bool SimpleNetwork::validate ( )

Reimplemented from NeuralNetwork.

The documentation for this class was generated from the following file:

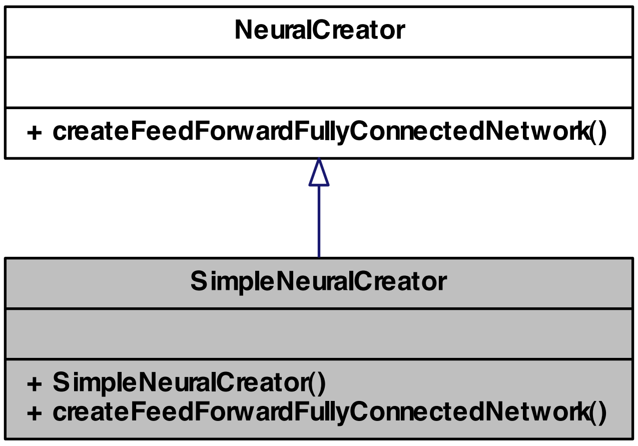
- pkg/AMORE/src/dia/SimpleNetwork.h

5.41 SimpleNeuralCreator Class Reference

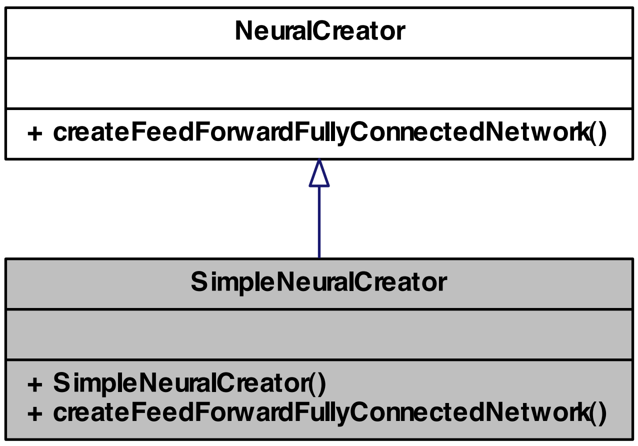
class SimpleNeuralCreator -

`#include <SimpleNeuralCreator.h>`

Inheritance diagram for SimpleNeuralCreator:


NeuralCreator
+ createFeedForwardFullyConnectedNetwork()
SimpleNeuralCreator
+ SimpleNeuralCreator()
+ createFeedForwardFullyConnectedNetwork()

Collaboration diagram for SimpleNeuralCreator:


NeuralCreator
+ createFeedForwardFullyConnectedNetwork()
SimpleNeuralCreator
+ SimpleNeuralCreator()
+ createFeedForwardFullyConnectedNetwork()

Public Member Functions

- SimpleNeuralCreator ()
- NeuralNetworkPtr createFeedForwardFullyConnectedNetwork (NeuralFactoryPtr neuralFactoryPtr)

5.41.1 Detailed Description

class SimpleNeuralCreator -

Definition at line 5 of file SimpleNeuralCreator.h.

5.41.2 Constructor & Destructor Documentation

5.41.2.1 SimpleNeuralCreator::SimpleNeuralCreator ( )

Definition at line 15 of file SimpleNeuralCreator.cpp.

```
{
}
```

5.41.3 Member Function Documentation

5.41.3.1 NeuralNetworkPtr SimpleNeuralCreator::createFeedForwardFullyConnectedNetwork (NeuralFactoryPtr *neuralFactoryPtr*) [virtual]

Implements NeuralCreator.

Definition at line 22 of file SimpleNeuralCreator.cpp.

```
{
  NeuralNetworkPtr n;
  return n ; // change this, does'n make sense
}
```

The documentation for this class was generated from the following files:

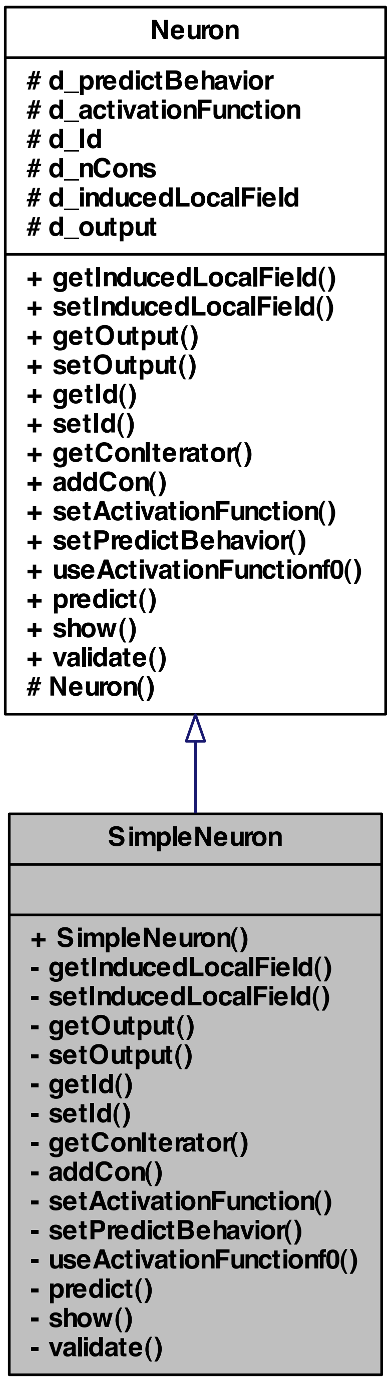
- pkg/AMORE/src/dia/SimpleNeuralCreator.h
- pkg/AMORE/src/SimpleNeuralCreator.cpp

5.42 SimpleNeuron Class Reference

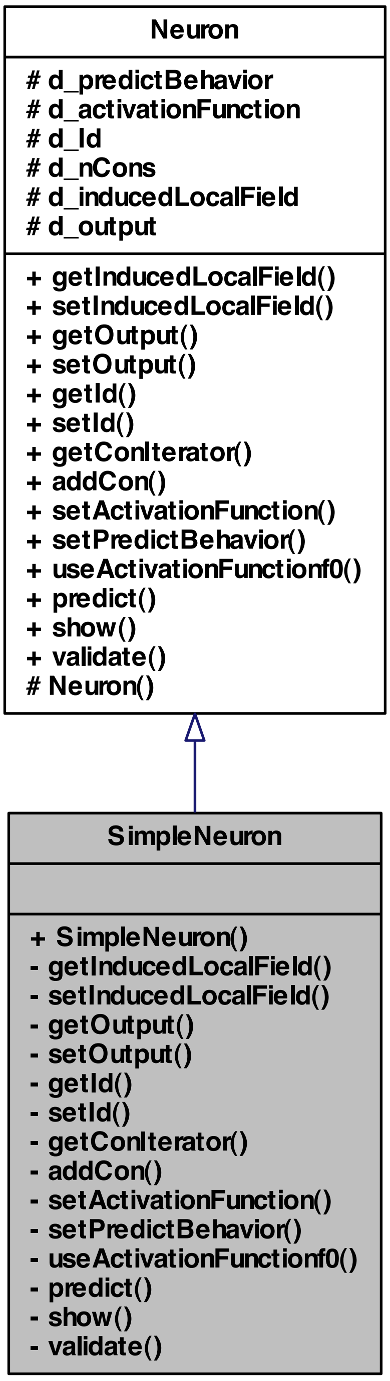
class SimpleNeuron -

`#include <SimpleNeuron.h>`

Inheritance diagram for SimpleNeuron:



Collaboration diagram for SimpleNeuron:



Public Member Functions

- SimpleNeuron (NeuralFactory &neuralFactory)

Private Member Functions

- double getInducedLocalField ()
- void setInducedLocalField (double inducedLocalField)
- double getOutput ()
- void setOutput (double output)
- Handler getId ()
- void setId (Handler Id)
- ConIteratorPtr getConIterator ()
- void addCon (ConPtr conPtr)
- void setActivationFunction (ActivationFunctionPtr activationFunctionPtr)
- void setPredictBehavior (PredictBehaviorPtr predictBehaviorPtr)
- double useActivationFunctionf0 ()
- void predict ()
- void show ()
- bool validate ()

5.42.1 Detailed Description

class SimpleNeuron -

Definition at line 5 of file SimpleNeuron.h.

5.42.2 Constructor & Destructor Documentation

5.42.2.1 SimpleNeuron::SimpleNeuron (**NeuralFactory** & *neuralFactory*)

Definition at line 10 of file SimpleNeuron.cpp.

```
                                                                    :
  Neuron(neuralFactory)
{
}
```

5.42.3 Member Function Documentation

5.42.3.1 void SimpleNeuron::addCon (**ConPtr** *conPtr*) `[private, virtual]`

Implements Neuron.

Definition at line 59 of file SimpleNeuron.cpp.

References Neuron::d_nCons.

```
{
  d_nCons->push_back( conPtr) ;
}
```

5.42.3.2 ConIteratorPtr SimpleNeuron::getConIterator ( ) [private, virtual]

Implements Neuron.

Definition at line 53 of file SimpleNeuron.cpp.

References Neuron::d_nCons.

```
{
  return d_nCons->createIterator();
}
```

5.42.3.3 Handler SimpleNeuron::getId ( ) [private, virtual]

Implements Neuron.

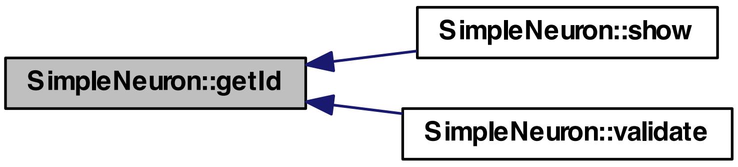
Definition at line 41 of file SimpleNeuron.cpp.

References Neuron::d_Id.

Referenced by show(), and validate().

```
{
  return d_Id;
}
```

Here is the caller graph for this function:



5.42.3.4 double SimpleNeuron::getInducedLocalField ( ) [private, virtual]

Implements Neuron.

Definition at line 17 of file SimpleNeuron.cpp.

References Neuron::d_inducedLocalField.

```
{
  return d_inducedLocalField;
}
```

5.42.3.5 double SimpleNeuron::getOutput ( ) [private, virtual]

Implements Neuron.

Definition at line 29 of file SimpleNeuron.cpp.

References Neuron::d_output.

```
{
  return d_output;
}
```

5.42.3.6 void SimpleNeuron::predict ( ) [private, virtual]

Implements Neuron.

Definition at line 83 of file SimpleNeuron.cpp.

References Neuron::d_predictBehavior.

```
{
  d_predictBehavior->predict();
}
```

5.42.3.7 void SimpleNeuron::setActivationFunction (ActivationFunctionPtr *activationFunctionPtr*) [private, virtual]

Implements Neuron.

Definition at line 65 of file SimpleNeuron.cpp.

References Neuron::d_activationFunction.

```
{
  d_activationFunction = activationFunctionPtr;
}
```

5.42.3.8 void SimpleNeuron::setId (Handler *Id*) [private, virtual]

Implements Neuron.

Definition at line 47 of file SimpleNeuron.cpp.

References Neuron::d_Id.

```
{
  d_Id = Id;
}
```

5.42.3.9 void SimpleNeuron::setInducedLocalField (double *inducedLocalField*) [private, virtual]

Implements Neuron.

Definition at line 23 of file SimpleNeuron.cpp.

References Neuron::d_inducedLocalField.

```
{
  d_inducedLocalField = inducedLocalField;
}
```

5.42.3.10 void SimpleNeuron::setOutput (double *output*) [private, virtual]

Implements Neuron.

Definition at line 35 of file SimpleNeuron.cpp.

References Neuron::d_output.

```
{
  d_output = output;
}
```

5.42.3.11 void SimpleNeuron::setPredictBehavior (PredictBehaviorPtr *predictBehaviorPtr*) [private, virtual]

Implements Neuron.

Definition at line 71 of file SimpleNeuron.cpp.

References Neuron::d_predictBehavior.

```
{
  d_predictBehavior = predictBehaviorPtr;
}
```

5.42.3.12 void SimpleNeuron::show ( ) [private, virtual]

Implements Neuron.

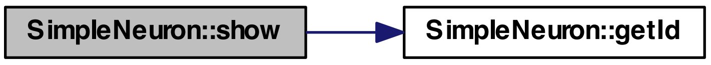
Definition at line 89 of file SimpleNeuron.cpp.

References Neuron::d_nCons, Neuron::d_output, Neuron::d_predictBehavior, and getId().

```
{
  int id = getId();
  Rprintf("\n------------------------\n");
  if (id == NA_INTEGER)
    {
      Rprintf("\n Id: NA, Invalid neuron Id");
    }
  else
    {
      Rprintf("\n Id: %d", id);
    }
  Rprintf("\n------------------------\n");
  d_predictBehavior->show();
  Rprintf("\n output: %lf", d_output);
  Rprintf("\n------------------------\n");
  if (d_nCons->size() == 0)
    {
      Rprintf("\n No connections defined");
    }
  else
    {
      d_nCons->show();
    }
  Rprintf("\n------------------------\n");

}
```

Here is the call graph for this function:



5.42.3.13 double SimpleNeuron::useActivationFunctionf0 ( ) [private, virtual]

Implements Neuron.

Definition at line 77 of file SimpleNeuron.cpp.

References Neuron::d_activationFunction.

```
{
  return d_activationFunction->f0();
}
```

5.42.3.14 bool SimpleNeuron::validate ( ) [private, virtual]

Implements Neuron.

Definition at line 118 of file SimpleNeuron.cpp.

References getId().

```
{
  BEGIN_RCPP
  if (getId() == NA_INTEGER ) throw std::range_error("[C++ SimpleNeuron::validate
      ]: Error, Id is NA.");
  // nCons.validate();
  return (TRUE);
END_RCPP}
```

Here is the call graph for this function:

SimpleNeuron::validate
SimpleNeuron::getId

The documentation for this class was generated from the following files:

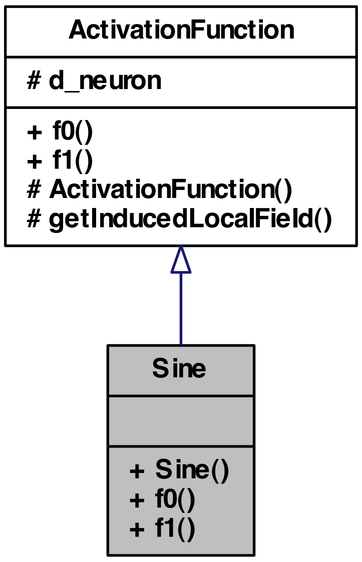
- pkg/AMORE/src/dia/SimpleNeuron.h
- pkg/AMORE/src/SimpleNeuron.cpp

5.43 Sine Class Reference

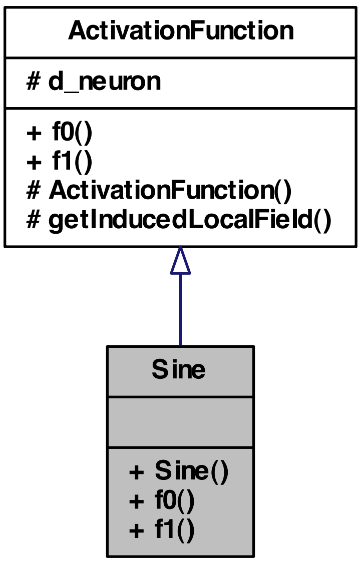
class Sine -

`#include <Sine.h>`

Inheritance diagram for Sine:



Collaboration diagram for Sine:



Public Member Functions

- Sine (NeuronPtr neuronPtr)
- double f0 ()
- double f1 ()

5.43.1 Detailed Description

class Sine -

Definition at line 5 of file Sine.h.

5.43.2 Constructor & Destructor Documentation

5.43.2.1 Sine::Sine (NeuronPtr *neuronPtr*)

5.43.3 Member Function Documentation

5.43.3.1 double Sine::f0 ( ) `[virtual]`

Implements ActivationFunction.

5.43.3.2 double Sine::f1 ( ) `[virtual]`

Implements ActivationFunction.

The documentation for this class was generated from the following file:

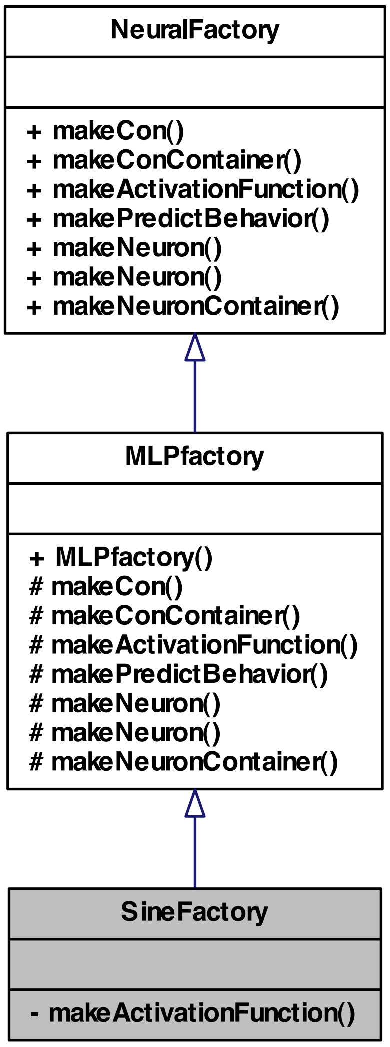
- pkg/AMORE/src/dia/Sine.h

5.44 SineFactory Class Reference

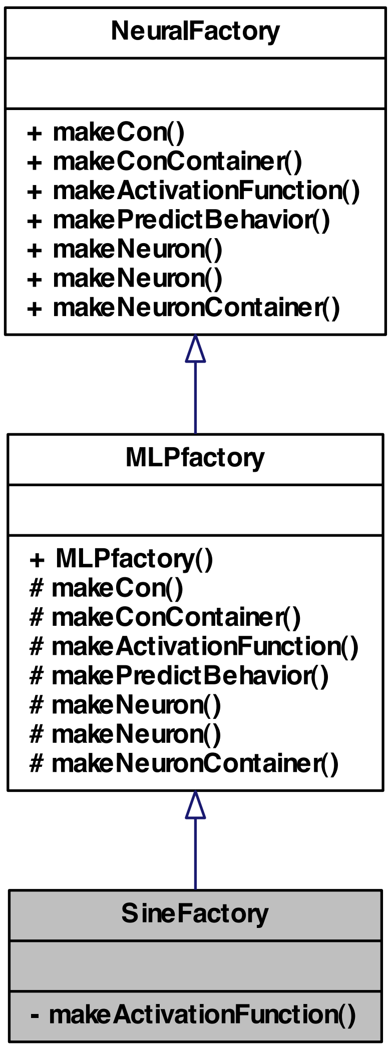
class SineFactory -

`#include <SineFactory.h>`

Inheritance diagram for SineFactory:



Collaboration diagram for SineFactory:



Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

5.44.1 Detailed Description

class SineFactory -

Definition at line 5 of file SineFactory.h.

5.44.2 Member Function Documentation

5.44.2.1 ActivationFunctionPtr SineFactory::makeActivationFunction (NeuronPtr *neuronPtr*) `[private, virtual]`

Implements MLPfactory.

The documentation for this class was generated from the following file:

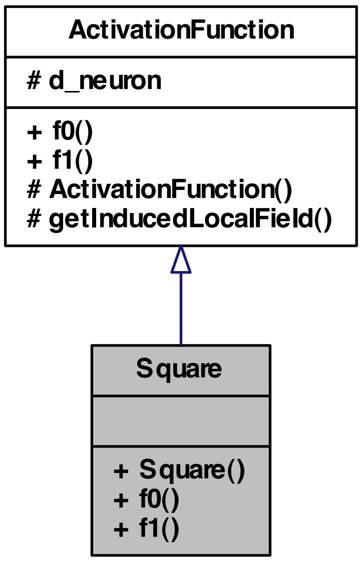
- pkg/AMORE/src/dia/SineFactory.h

5.45 Square Class Reference

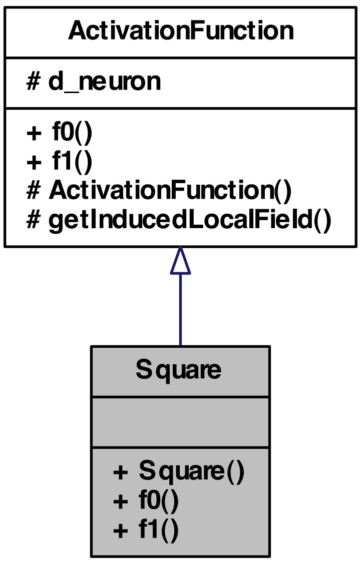
class Square -

`#include <Square.h>`

Inheritance diagram for Square:



Collaboration diagram for Square:



Public Member Functions

- Square (NeuronPtr neuronPtr)
- double f0 ()
- double f1 ()

5.45.1 Detailed Description

class Square -

Definition at line 5 of file Square.h.

5.45.2 Constructor & Destructor Documentation

5.45.2.1 Square::Square (NeuronPtr *neuronPtr*)

5.45.3 Member Function Documentation

5.45.3.1 double Square::f0 ( ) `[virtual]`

Implements ActivationFunction.

5.45.3.2 double Square::f1 ( ) `[virtual]`

Implements ActivationFunction.

The documentation for this class was generated from the following file:

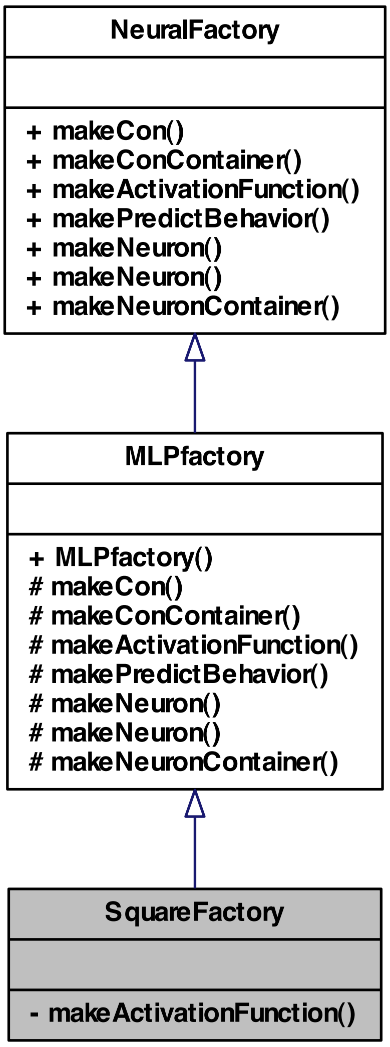
- pkg/AMORE/src/dia/Square.h

5.46 SquareFactory Class Reference

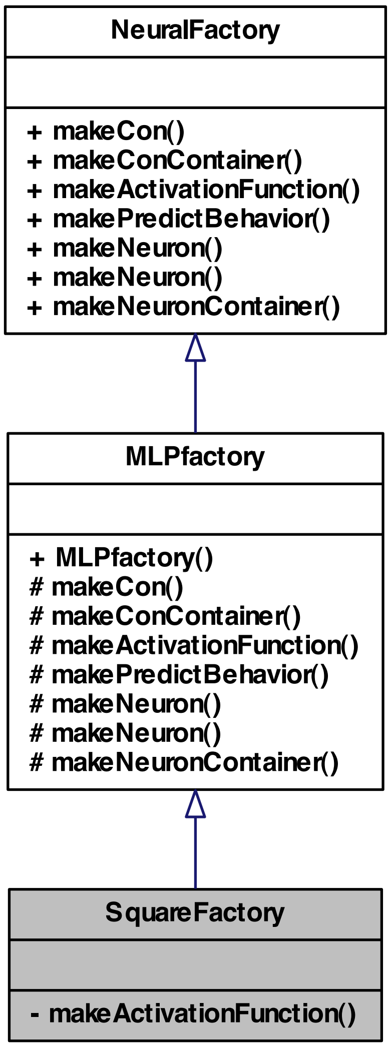
class SquareFactory -

`#include <SquareFactory.h>`

Inheritance diagram for SquareFactory:



Collaboration diagram for SquareFactory:



Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

5.46.1 Detailed Description

class SquareFactory -

Definition at line 5 of file SquareFactory.h.

5.46.2 Member Function Documentation

5.46.2.1 **ActivationFunctionPtr** SquareFactory::makeActivationFunction (**NeuronPtr** ***neuronPtr***) `[private, virtual]`

Implements MLPfactory.

The documentation for this class was generated from the following file:

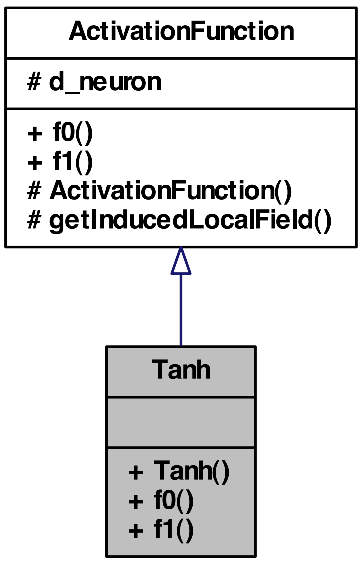
- pkg/AMORE/src/dia/SquareFactory.h

5.47 Tanh Class Reference

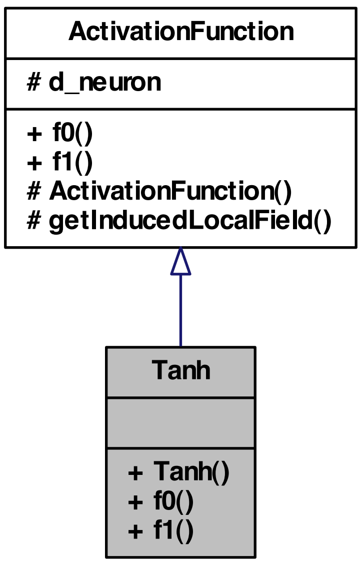
class Tanh -

`#include <Tanh.h>`

Inheritance diagram for Tanh:



Collaboration diagram for Tanh:



Public Member Functions

- Tanh (NeuronPtr neuronPtr)
- double f0 ()
- double f1 ()

5.47.1 Detailed Description

class Tanh -

Definition at line 5 of file Tanh.h.

5.47.2 Constructor & Destructor Documentation

5.47.2.1 Tanh::Tanh (**NeuronPtr** *neuronPtr*)

Definition at line 12 of file Tanh.cpp.

```
                              : ActivationFunction(neuronPtr) {
}
```

5.47.3 Member Function Documentation

5.47.3.1 double Tanh::f0 ( ) [virtual]

Implements ActivationFunction.

Definition at line 16 of file Tanh.cpp.

References ActivationFunction::getInducedLocalField().

```
                   {
  return tanh(getInducedLocalField());

}
```

Here is the call graph for this function:

Tanh::f0
ActivationFunction::getInducedLocalField

5.47.3.2 double Tanh::f1 ( ) [virtual]

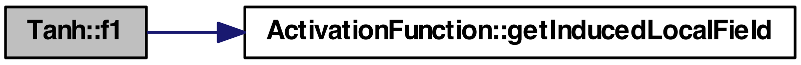
Implements ActivationFunction.

Definition at line 21 of file Tanh.cpp.

References ActivationFunction::getInducedLocalField().

```
                   {
  double tanhx ( tanh(getInducedLocalField()) );
  return (1-tanhx*tanhx) ; // TODO consider speeding up the calculation by using
      caller.d_output instead of tanhx
}
```

Here is the call graph for this function:



The documentation for this class was generated from the following files:

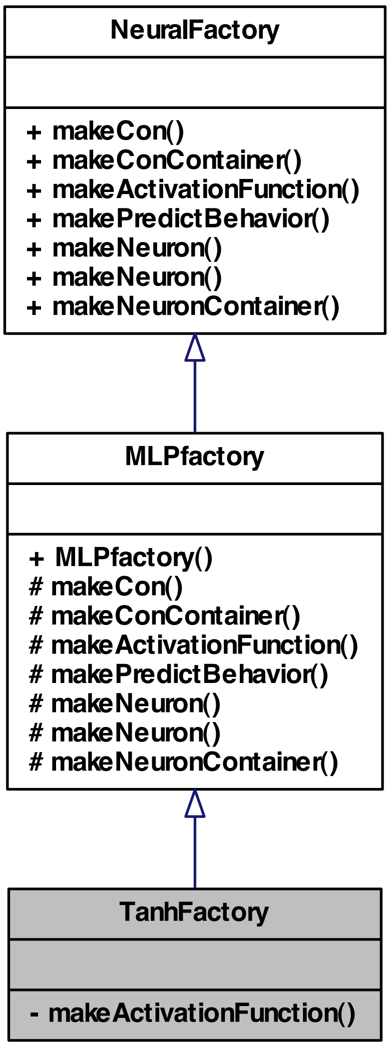
- pkg/AMORE/src/dia/Tanh.h
- pkg/AMORE/src/Tanh.cpp

5.48 TanhFactory Class Reference

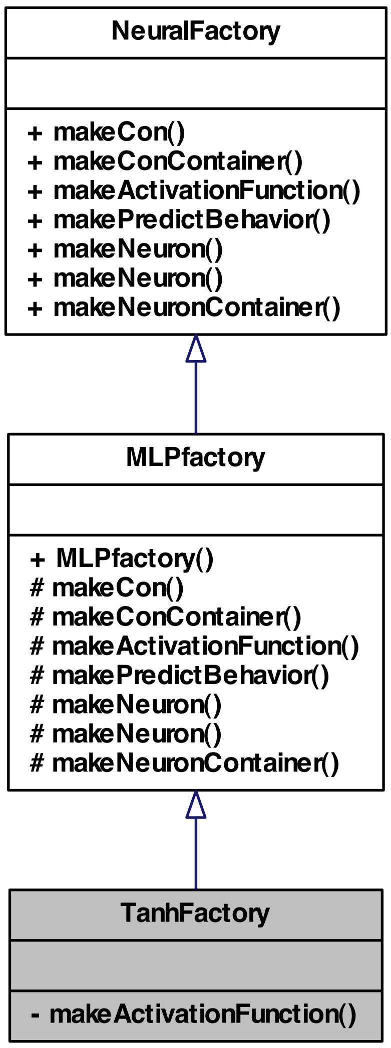
class TanhFactory -

```
#include <TanhFactory.h>
```

Inheritance diagram for TanhFactory:



Collaboration diagram for TanhFactory:



Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

5.48.1 Detailed Description

class TanhFactory -

Definition at line 5 of file TanhFactory.h.

5.48.2 Member Function Documentation

5.48.2.1 ActivationFunctionPtr TanhFactory::makeActivationFunction (NeuronPtr *neuronPtr*) [private, virtual]

Implements MLPfactory.

Definition at line 17 of file TanhFactory.cpp.

```
{
  ActivationFunctionPtr activationFunctionPtr(new Tanh(neuronPtr));
  return activationFunctionPtr;
}
```

The documentation for this class was generated from the following files:

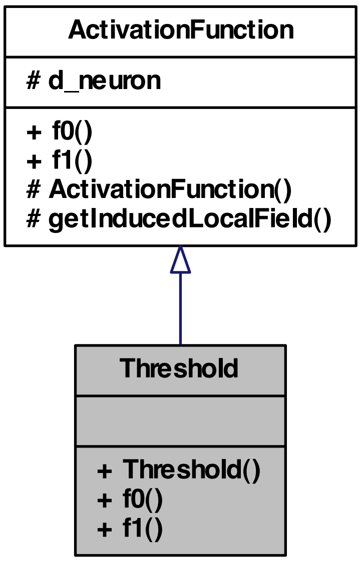
- pkg/AMORE/src/dia/TanhFactory.h
- pkg/AMORE/src/TanhFactory.cpp

5.49 Threshold Class Reference

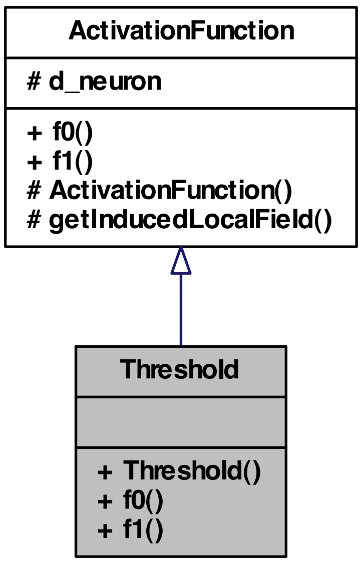
class Threshold -

`#include <Threshold.h>`

Inheritance diagram for Threshold:



Collaboration diagram for Threshold:



Public Member Functions

- Threshold (NeuronPtr neuronPtr)
- double f0 ()
- double f1 ()

5.49.1 Detailed Description

class Threshold -

Definition at line 5 of file Threshold.h.

5.49.2 Constructor & Destructor Documentation

5.49.2.1 Threshold::Threshold (NeuronPtr *neuronPtr*)

5.49.3 Member Function Documentation

5.49.3.1 double Threshold::f0 ( ) `[virtual]`

Implements ActivationFunction.

5.49.3.2 double Threshold::f1 ( ) `[virtual]`

Implements ActivationFunction.

The documentation for this class was generated from the following file:

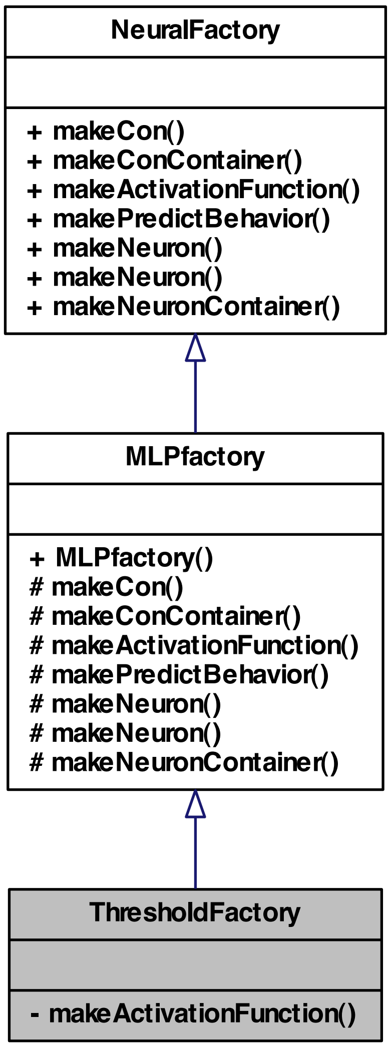
- pkg/AMORE/src/dia/Threshold.h

5.50 ThresholdFactory Class Reference

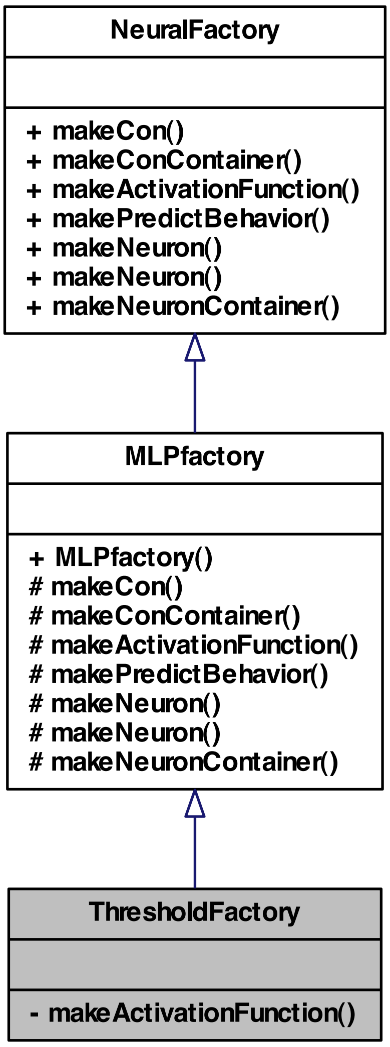
class ThresholdFactory -

`#include <ThresholdFactory.h>`

Inheritance diagram for ThresholdFactory:



Collaboration diagram for ThresholdFactory:



Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

5.50.1 Detailed Description

class ThresholdFactory -

Definition at line 5 of file ThresholdFactory.h.

5.50.2 Member Function Documentation

5.50.2.1 ActivationFunctionPtr ThresholdFactory::makeActivationFunction (NeuronPtr *neuronPtr*) `[private, virtual]`

Implements MLPfactory.

The documentation for this class was generated from the following file:

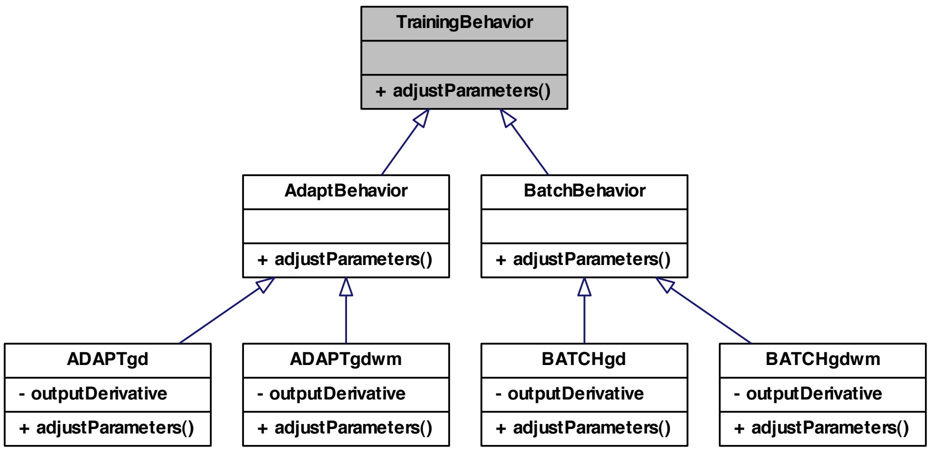
- pkg/AMORE/src/dia/ThresholdFactory.h

5.51 TrainingBehavior Class Reference

class TrainingBehavior -

`#include <TrainingBehavior.h>`

Inheritance diagram for TrainingBehavior:



Public Member Functions

- void adjustParameters ()

5.51.1 Detailed Description

class TrainingBehavior -

Definition at line 4 of file TrainingBehavior.h.

5.51.2 Member Function Documentation

5.51.2.1 void TrainingBehavior::adjustParameters ( )

Reimplemented in AdaptBehavior, ADAPTgd, ADAPTgdwm, BatchBehavior, BATCHgd, and BATCHgdwm.

The documentation for this class was generated from the following file:

- pkg/AMORE/src/dia/TrainingBehavior.h

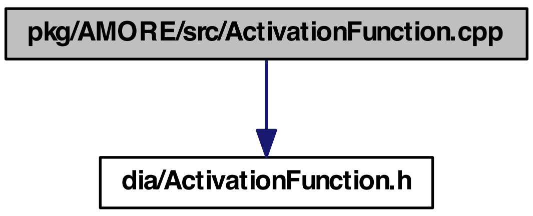
Chapter 6

File Documentation

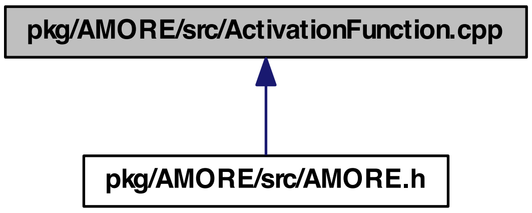
6.1 pkg/AMORE/src/ActivationFunction.cpp File Reference

```
#include "dia/ActivationFunction.h"
```

Include dependency graph for ActivationFunction.cpp:



This graph shows which files directly or indirectly include this file:

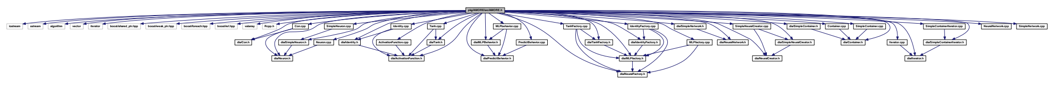


6.2 pkg/AMORE/src/AMORE.h File Reference

```
#include <iostream>
#include <sstream>
#include <algorithm>
#include <vector>
#include <iterator>
#include <boost/shared_ptr.hpp>
#include <boost/weak_ptr.hpp>
#include <boost/foreach.hpp>
#include <boost/ref.hpp>
#include <valarray>
#include <Rcpp.h>
#include "dia/Con.h"
#include "dia/ActivationFunction.h"
#include "dia/Tanh.h"
#include "dia/Identity.h"
#include "dia/PredictBehavior.h"
#include "dia/MLPBehavior.h"
#include "dia/Neuron.h"
#include "dia/SimpleNeuron.h"
#include "dia/NeuralFactory.h"
```

```
#include "dia/MLPfactory.h"
#include "dia/TanhFactory.h"
#include "dia/IdentityFactory.h"
#include "dia/NeuralNetwork.h"
#include "dia/SimpleNetwork.h"
#include "dia/NeuralCreator.h"
#include "dia/SimpleNeuralCreator.h"
#include "dia/Container.h"
#include "dia/SimpleContainer.h"
#include "dia/Iterator.h"
#include "dia/SimpleContainerIterator.h"
#include "Con.cpp"
#include "ActivationFunction.cpp"
#include "Tanh.cpp"
#include "Identity.cpp"
#include "PredictBehavior.cpp"
#include "MLPbehavior.cpp"
#include "Neuron.cpp"
#include "SimpleNeuron.cpp"
#include "MLPfactory.cpp"
#include "TanhFactory.cpp"
#include "IdentityFactory.cpp"
#include "NeuralNetwork.cpp"
#include "SimpleNetwork.cpp"
#include "SimpleNeuralCreator.cpp"
#include "Container.cpp"
#include "Iterator.cpp"
#include "SimpleContainer.cpp"
#include "SimpleContainerIterator.cpp"
```

Include dependency graph for AMORE.h:



Defines

- #define foreach BOOST_FOREACH
- #define size_type unsigned int

Typedefs

- typedef int Handler
- typedef boost::reference_wrapper< PredictBehavior > ActivationFunctionRef
- typedef boost::reference_wrapper< PredictBehavior > PredictBehaviorRef
- typedef boost::reference_wrapper< TrainingBehavior > TrainingBehaviorRef
- typedef boost::reference_wrapper< Neuron > NeuronRef
- typedef boost::shared_ptr< ActivationFunction > ActivationFunctionPtr
- typedef boost::shared_ptr< PredictBehavior > PredictBehaviorPtr
- typedef boost::shared_ptr< Neuron > NeuronPtr
- typedef boost::shared_ptr< Con > ConPtr
- typedef boost::shared_ptr< NeuralNetwork > NeuralNetworkPtr
- typedef boost::shared_ptr< Iterator< NeuronPtr > > NeuronIteratorPtr
- typedef boost::shared_ptr< Iterator< ConPtr > > ConIteratorPtr
- typedef boost::shared_ptr< Container< NeuronPtr > > LayerPtr
- typedef boost::shared_ptr< Container< NeuronPtr > > NeuronContainerPtr
- typedef boost::shared_ptr< Container< ConPtr > > ConContainerPtr
- typedef boost::shared_ptr< NeuralFactory > NeuralFactoryPtr
- typedef boost::shared_ptr< NeuralCreator > NeuralCreatorPtr
- typedef boost::weak_ptr< Neuron > NeuronWeakPtr

6.2.1 Define Documentation

6.2.1.1 #define foreach BOOST_FOREACH

Definition at line 68 of file AMORE.h.

6.2.1.2 #define size_type unsigned int

Definition at line 71 of file AMORE.h.

6.2.2 Typedef Documentation

6.2.2.1 typedef boost::shared_ptr<ActivationFunction> ActivationFunctionPtr

Definition at line 83 of file AMORE.h.

6.2.2.2 typedef boost::reference_wrapper<PredictBehavior> ActivationFunctionRef

Definition at line 77 of file AMORE.h.

6.2.2.3 typedef boost::shared_ptr< Container<ConPtr> > ConContainerPtr

Definition at line 95 of file AMORE.h.

6.2.2.4 typedef boost::shared_ptr< Iterator<ConPtr> > ConIteratorPtr

Definition at line 91 of file AMORE.h.

6.2.2.5 typedef boost::shared_ptr<Con> ConPtr

Definition at line 86 of file AMORE.h.

6.2.2.6 typedef int Handler

Definition at line 74 of file AMORE.h.

6.2.2.7 typedef boost::shared_ptr< Container<NeuronPtr> > LayerPtr

Definition at line 93 of file AMORE.h.

6.2.2.8 typedef boost::shared_ptr< NeuralCreator > NeuralCreatorPtr

Definition at line 98 of file AMORE.h.

6.2.2.9 typedef boost::shared_ptr< NeuralFactory > NeuralFactoryPtr

Definition at line 97 of file AMORE.h.

6.2.2.10 typedef boost::shared_ptr<NeuralNetwork> NeuralNetworkPtr

Definition at line 87 of file AMORE.h.

6.2.2.11 typedef boost::shared_ptr< Container<NeuronPtr> > NeuronContainerPtr

Definition at line 94 of file AMORE.h.

6.2.2.12 typedef boost::shared_ptr< Iterator<NeuronPtr> > NeuronIteratorPtr

Definition at line 90 of file AMORE.h.

6.2.2.13 typedef boost::shared_ptr<Neuron> NeuronPtr

Definition at line 85 of file AMORE.h.

6.2.2.14 typedef boost::reference_wrapper<Neuron> NeuronRef

Definition at line 80 of file AMORE.h.

6.2.2.15 typedef boost::weak_ptr<Neuron> NeuronWeakPtr

Definition at line 100 of file AMORE.h.

6.2.2.16 typedef boost::shared_ptr<PredictBehavior> PredictBehaviorPtr

Definition at line 84 of file AMORE.h.

6.2.2.17 typedef boost::reference_wrapper<PredictBehavior> PredictBehaviorRef

Definition at line 78 of file AMORE.h.

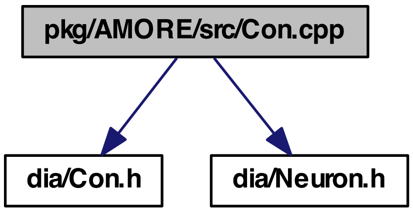
6.2.2.18 typedef boost::reference_wrapper<TrainingBehavior> TrainingBehaviorRef

Definition at line 79 of file AMORE.h.

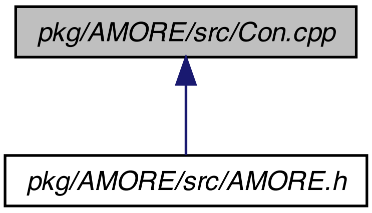
6.3 pkg/AMORE/src/Con.cpp File Reference

```
#include "dia/Con.h"
#include "dia/Neuron.h"
```

Include dependency graph for Con.cpp:



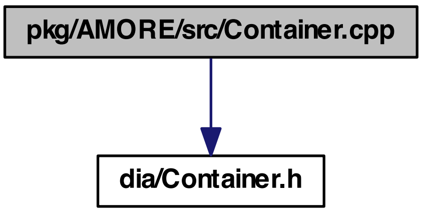
This graph shows which files directly or indirectly include this file:



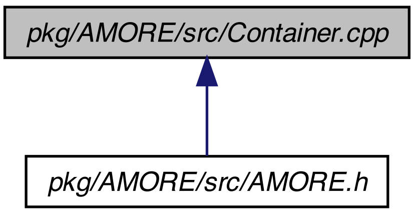
6.4 pkg/AMORE/src/Container.cpp File Reference

```
#include "dia/Container.h"
```

Include dependency graph for Container.cpp:

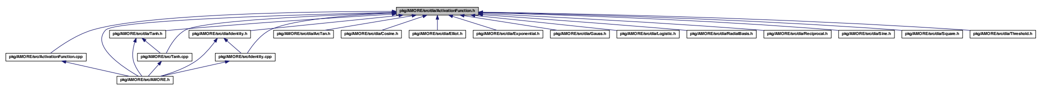


This graph shows which files directly or indirectly include this file:



6.5 pkg/AMORE/src/dia/ActivationFunction.h File Reference

This graph shows which files directly or indirectly include this file:


pkg/AMORE/src/dia/ActivationFunction.h
pkg/AMORE/src/dia/Tanh.h
pkg/AMORE/src/dia/Identity.h
pkg/AMORE/src/dia/ArcTan.h
pkg/AMORE/src/dia/Cosine.h
pkg/AMORE/src/dia/Elliot.h
pkg/AMORE/src/dia/Exponential.h
pkg/AMORE/src/dia/Gauss.h
pkg/AMORE/src/dia/Logistic.h
pkg/AMORE/src/dia/RadialBasis.h
pkg/AMORE/src/dia/Reciprocal.h
pkg/AMORE/src/dia/Sine.h
pkg/AMORE/src/dia/Square.h
pkg/AMORE/src/dia/Threshold.h
pkg/AMORE/src/ActivationFunction.cpp
pkg/AMORE/src/Tanh.cpp
pkg/AMORE/src/Identity.cpp
pkg/AMORE/src/AMORE.h

Classes

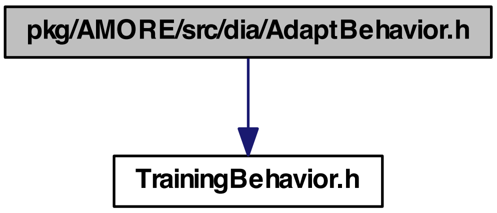
- class ActivationFunction

class ActivationFunction -

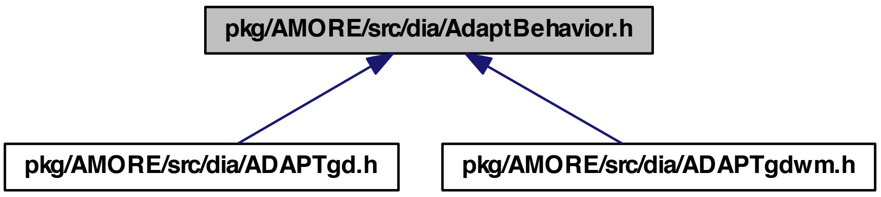
6.6 pkg/AMORE/src/dia/AdaptBehavior.h File Reference

`#include "TrainingBehavior.h"`

Include dependency graph for AdaptBehavior.h:



This graph shows which files directly or indirectly include this file:


pkg/AMORE/src/dia/AdaptBehavior.h
pkg/AMORE/src/dia/ADAPTgd.h
pkg/AMORE/src/dia/ADAPTgdwm.h

Classes

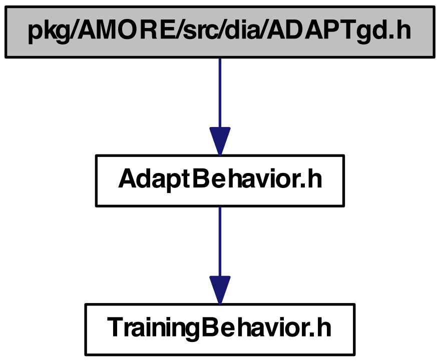
- class AdaptBehavior

  class AdaptBehavior -

6.7 pkg/AMORE/src/dia/ADAPTgd.h File Reference

```
#include "AdaptBehavior.h"
```

Include dependency graph for ADAPTgd.h:



Classes

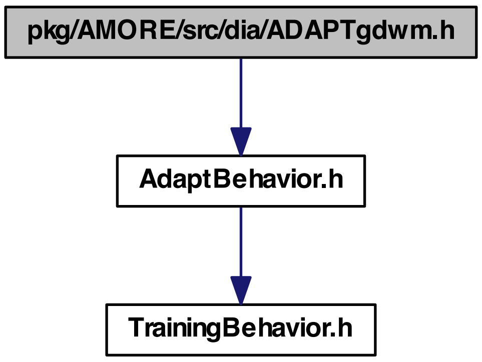
- class ADAPTgd

    class ADAPTgd -

6.8 pkg/AMORE/src/dia/ADAPTgdwm.h File Reference

```
#include "AdaptBehavior.h"
```

Include dependency graph for ADAPTgdwm.h:



Classes

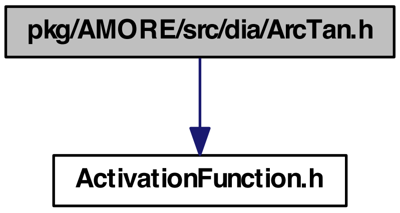
- class ADAPTgdwm

  class ADAPTgdwm -

6.9 pkg/AMORE/src/dia/ArcTan.h File Reference

```
#include "ActivationFunction.h"
```

Include dependency graph for ArcTan.h:



Classes

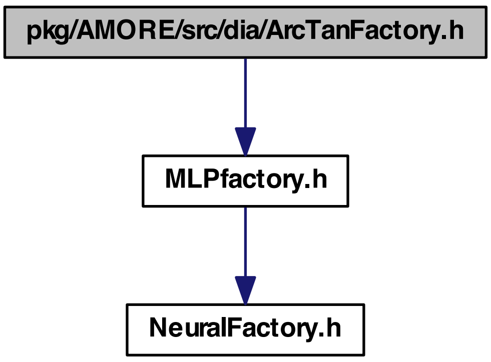
- class ArcTan

  class ArcTan -

6.10 pkg/AMORE/src/dia/ArcTanFactory.h File Reference

```
#include "MLPfactory.h"
```

Include dependency graph for ArcTanFactory.h:



Classes

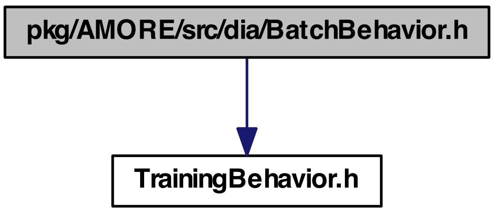
- class ArcTanFactory

  class ArcTanFactory -

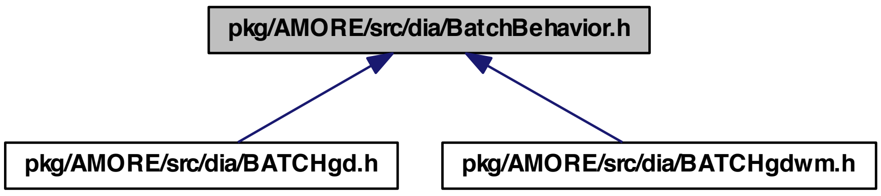
6.11 pkg/AMORE/src/dia/BatchBehavior.h File Reference

```
#include "TrainingBehavior.h"
```

Include dependency graph for BatchBehavior.h:



This graph shows which files directly or indirectly include this file:


pkg/AMORE/src/dia/BatchBehavior.h
pkg/AMORE/src/dia/BATCHgd.h
pkg/AMORE/src/dia/BATCHgdwm.h

Classes

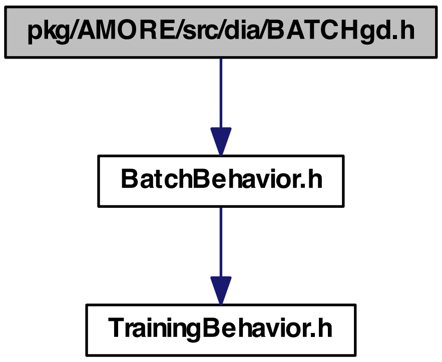
- class BatchBehavior

    class BatchBehavior -

6.12 pkg/AMORE/src/dia/BATCHgd.h File Reference

```
#include "BatchBehavior.h"
```

Include dependency graph for BATCHgd.h:



Classes

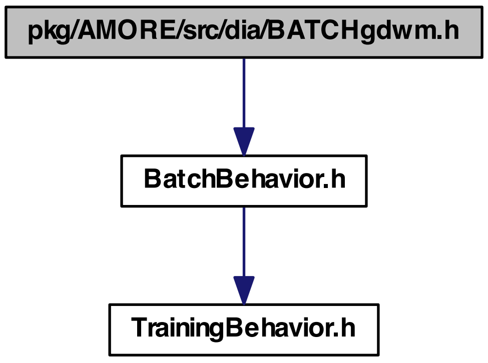
- class BATCHgd

    class BATCHgd -

6.13 pkg/AMORE/src/dia/BATCHgdwm.h File Reference

```
#include "BatchBehavior.h"
```

Include dependency graph for BATCHgdwm.h:



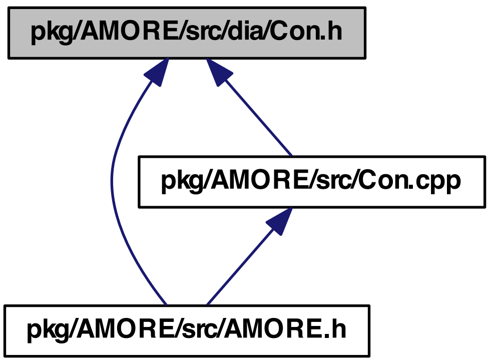
Classes

- class BATCHgdwm

class BATCHgdwm -

6.14 pkg/AMORE/src/dia/Con.h File Reference

This graph shows which files directly or indirectly include this file:



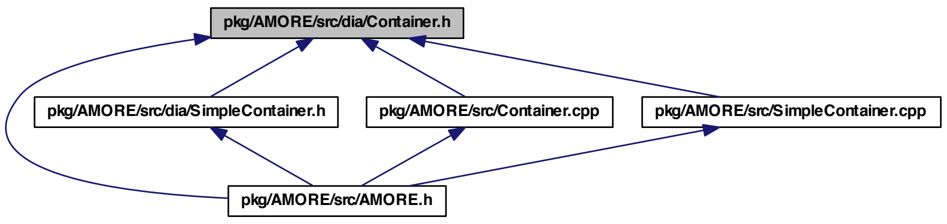
Classes

- class Con

    class Con -

6.15 pkg/AMORE/src/dia/Container.h File Reference

This graph shows which files directly or indirectly include this file:


pkg/AMORE/src/dia/Container.h
pkg/AMORE/src/dia/SimpleContainer.h
pkg/AMORE/src/Container.cpp
pkg/AMORE/src/SimpleContainer.cpp
pkg/AMORE/src/AMORE.h

Classes

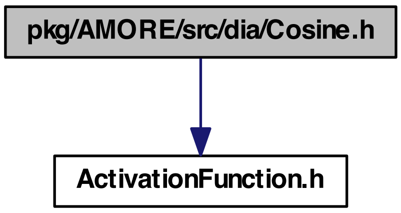
- class Container< T >

  class Container -

6.16 pkg/AMORE/src/dia/Cosine.h File Reference

```
#include "ActivationFunction.h"
```

Include dependency graph for Cosine.h:



Classes

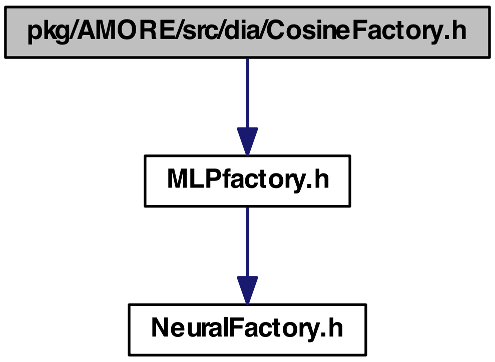
- class Cosine

  class Cosine -

6.17 pkg/AMORE/src/dia/CosineFactory.h File Reference

```
#include "MLPfactory.h"
```

Include dependency graph for CosineFactory.h:



Classes

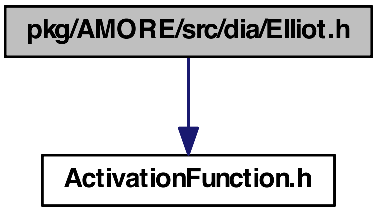
- class CosineFactory

  class CosineFactory -

6.18 pkg/AMORE/src/dia/Elliot.h File Reference

```
#include "ActivationFunction.h"
```

Include dependency graph for Elliot.h:



Classes

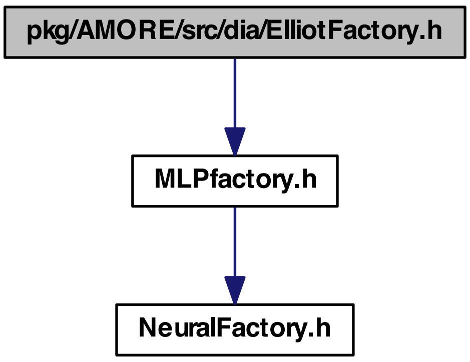
- class Elliot

  class Elliot -

6.19 pkg/AMORE/src/dia/ElliotFactory.h File Reference

```
#include "MLPfactory.h"
```

Include dependency graph for ElliotFactory.h:



Classes

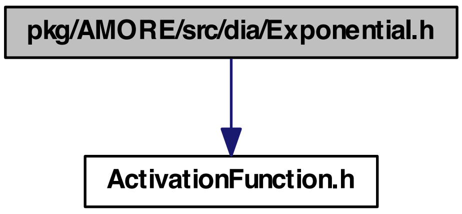
- class ElliotFactory

  class ElliotFactory -

6.20 pkg/AMORE/src/dia/Exponential.h File Reference

```
#include "ActivationFunction.h"
```

Include dependency graph for Exponential.h:



Classes

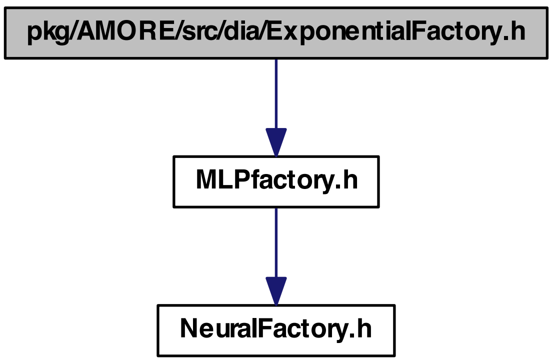
- class Exponential

  class Exponential -

6.21 pkg/AMORE/src/dia/ExponentialFactory.h File Reference

```
#include "MLPfactory.h"
```

Include dependency graph for ExponentialFactory.h:



Classes

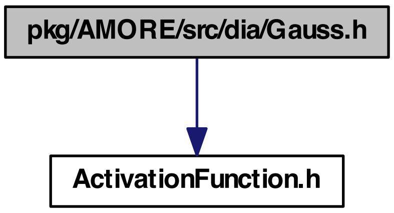
- class ExponentialFactory

    class ExponentialFactory -

6.22 pkg/AMORE/src/dia/Gauss.h File Reference

```
#include "ActivationFunction.h"
```

Include dependency graph for Gauss.h:



Classes

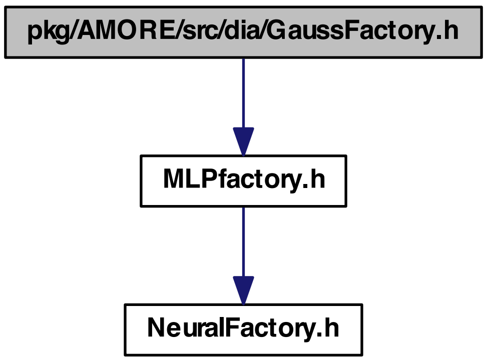
- class Gauss

    class Gauss -

6.23 pkg/AMORE/src/dia/GaussFactory.h File Reference

```
#include "MLPfactory.h"
```

Include dependency graph for GaussFactory.h:



Classes

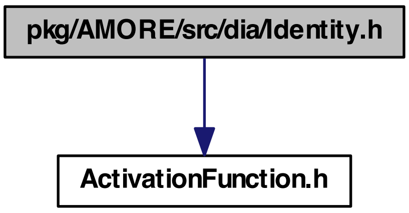
- class GaussFactory

    class GaussFactory -

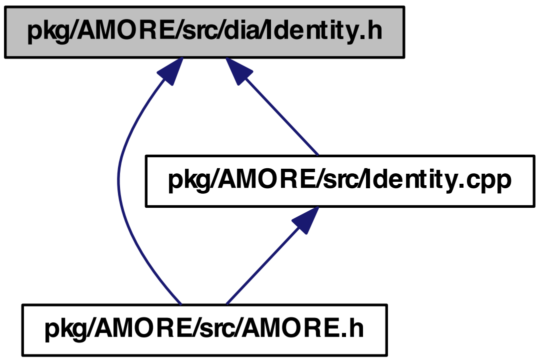
6.24 pkg/AMORE/src/dia/Identity.h File Reference

```
#include "ActivationFunction.h"
```

Include dependency graph for Identity.h:



This graph shows which files directly or indirectly include this file:



Classes

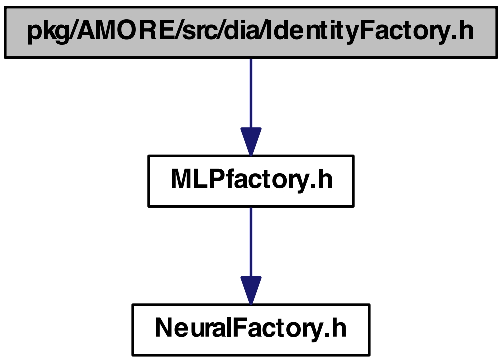
- class Identity

  class Identity -

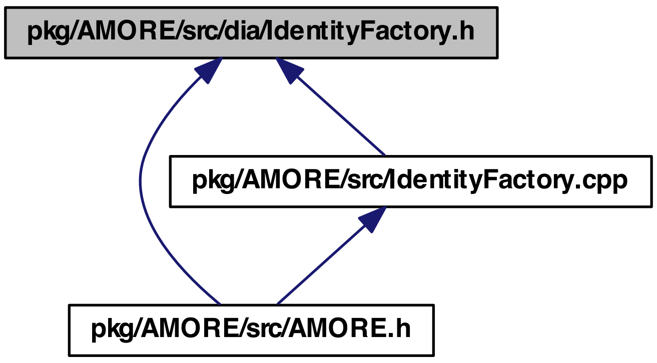
6.25 pkg/AMORE/src/dia/IdentityFactory.h File Reference

```
#include "MLPfactory.h"
```

Include dependency graph for IdentityFactory.h:



This graph shows which files directly or indirectly include this file:


pkg/AMORE/src/dia/IdentityFactory.h
pkg/AMORE/src/IdentityFactory.cpp
pkg/AMORE/src/AMORE.h

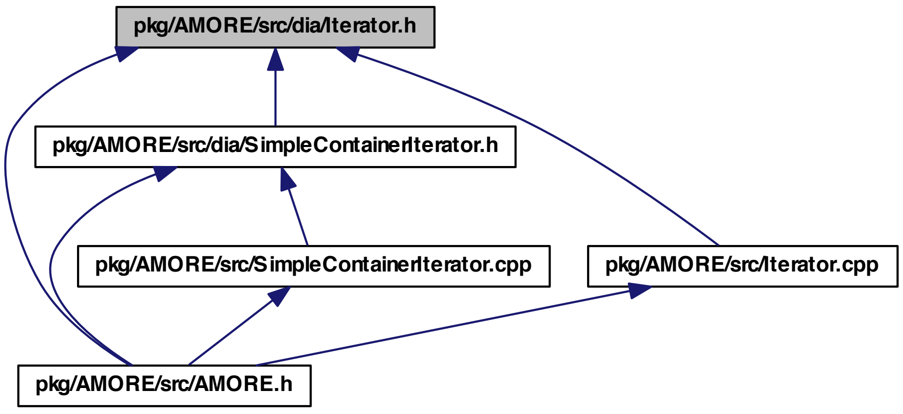
Classes

- class IdentityFactory

    class IdentityFactory -

6.26 pkg/AMORE/src/dia/Iterator.h File Reference

This graph shows which files directly or indirectly include this file:


pkg/AMORE/src/dia/Iterator.h
pkg/AMORE/src/dia/SimpleContainerIterator.h
pkg/AMORE/src/SimpleContainerIterator.cpp
pkg/AMORE/src/Iterator.cpp
pkg/AMORE/src/AMORE.h

Classes

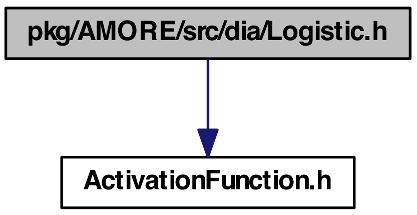
- class Iterator< T >

  class Iterator -

6.27 pkg/AMORE/src/dia/Logistic.h File Reference

```
#include "ActivationFunction.h"
```

Include dependency graph for Logistic.h:



Classes

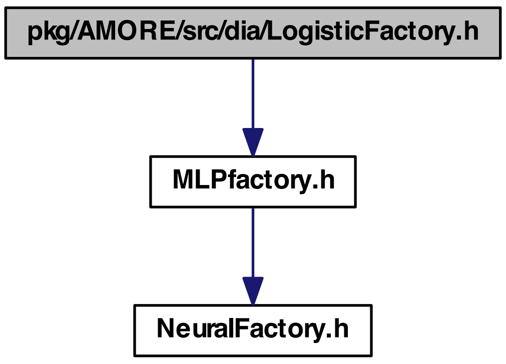
- class Logistic

    class Logistic -

6.28 pkg/AMORE/src/dia/LogisticFactory.h File Reference

```
#include "MLPfactory.h"
```

Include dependency graph for LogisticFactory.h:



Classes

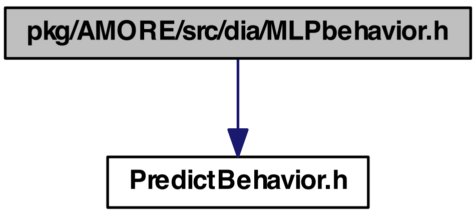
- class LogisticFactory

    class LogisticFactory -

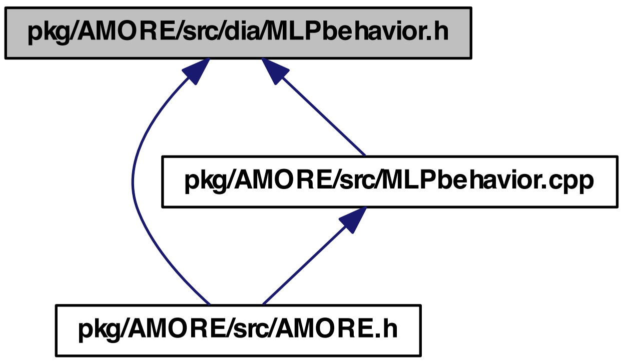
6.29 pkg/AMORE/src/dia/MLPbehavior.h File Reference

```
#include "PredictBehavior.h"
```

Include dependency graph for MLPbehavior.h:



This graph shows which files directly or indirectly include this file:


pkg/AMORE/src/dia/MLPbehavior.h
pkg/AMORE/src/MLPbehavior.cpp
pkg/AMORE/src/AMORE.h

Classes

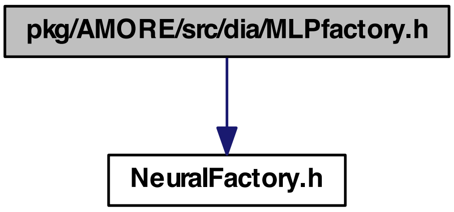
- class MLPbehavior

  class MLPbehavior -

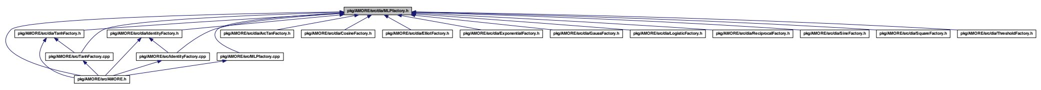
6.30 pkg/AMORE/src/dia/MLPfactory.h File Reference

`#include "NeuralFactory.h"`

Include dependency graph for MLPfactory.h:



This graph shows which files directly or indirectly include this file:



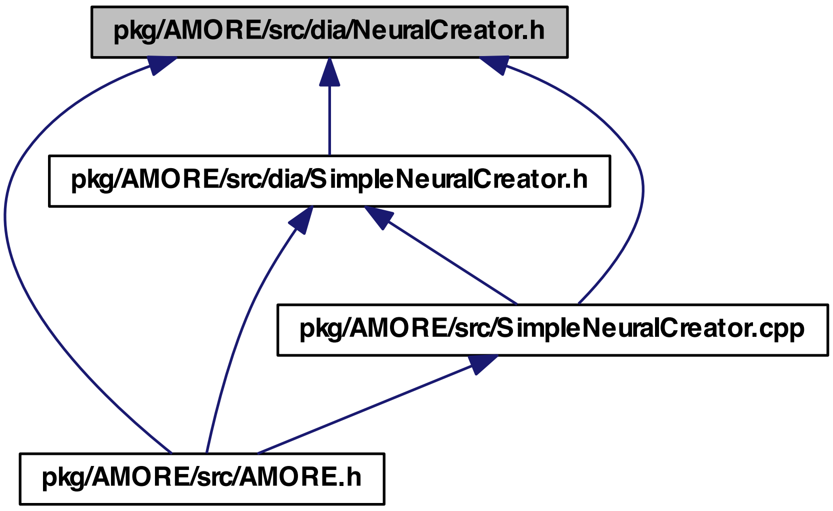
Classes

- class MLPfactory

  class MLPfactory -

6.31 pkg/AMORE/src/dia/NeuralCreator.h File Reference

This graph shows which files directly or indirectly include this file:


pkg/AMORE/src/dia/NeuralCreator.h
pkg/AMORE/src/dia/SimpleNeuralCreator.h
pkg/AMORE/src/SimpleNeuralCreator.cpp
pkg/AMORE/src/AMORE.h

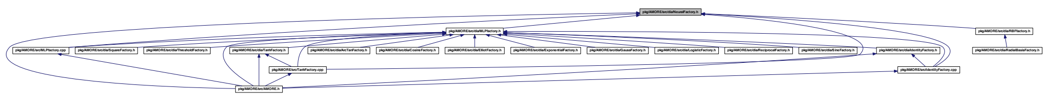
Classes

- class NeuralCreator

  class NeuralCreator -

6.32 pkg/AMORE/src/dia/NeuralFactory.h File Reference

This graph shows which files directly or indirectly include this file:



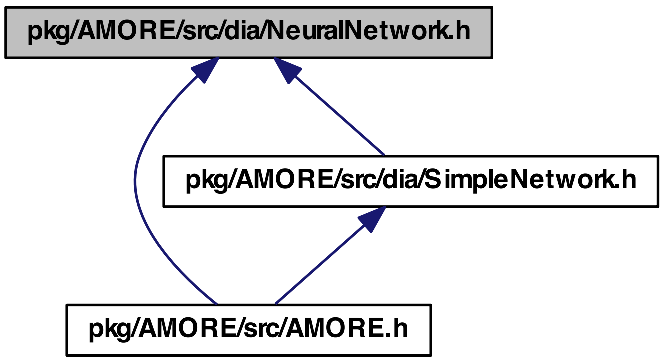
Classes

- class NeuralFactory

class NeuralFactory -

6.33 pkg/AMORE/src/dia/NeuralNetwork.h File Reference

This graph shows which files directly or indirectly include this file:


pkg/AMORE/src/dia/NeuralNetwork.h
pkg/AMORE/src/dia/SimpleNetwork.h
pkg/AMORE/src/AMORE.h

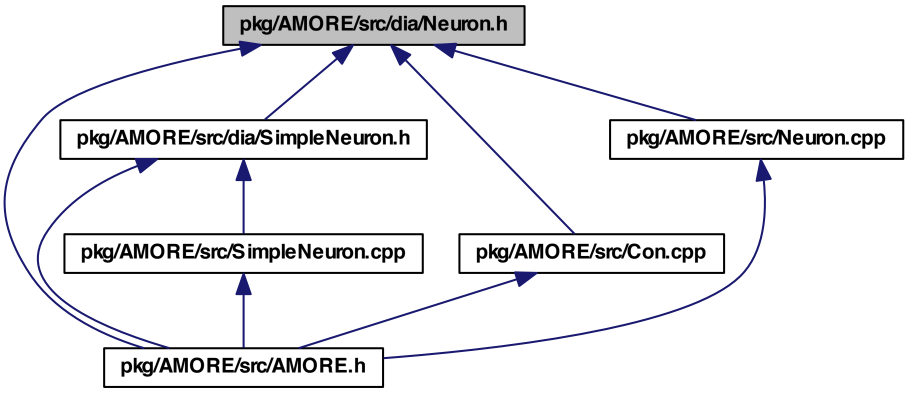
Classes

- class NeuralNetwork

  class NeuralNetwork -

6.34 pkg/AMORE/src/dia/Neuron.h File Reference

This graph shows which files directly or indirectly include this file:


pkg/AMORE/src/dia/Neuron.h
pkg/AMORE/src/dia/SimpleNeuron.h
pkg/AMORE/src/Neuron.cpp
pkg/AMORE/src/SimpleNeuron.cpp
pkg/AMORE/src/Con.cpp
pkg/AMORE/src/AMORE.h

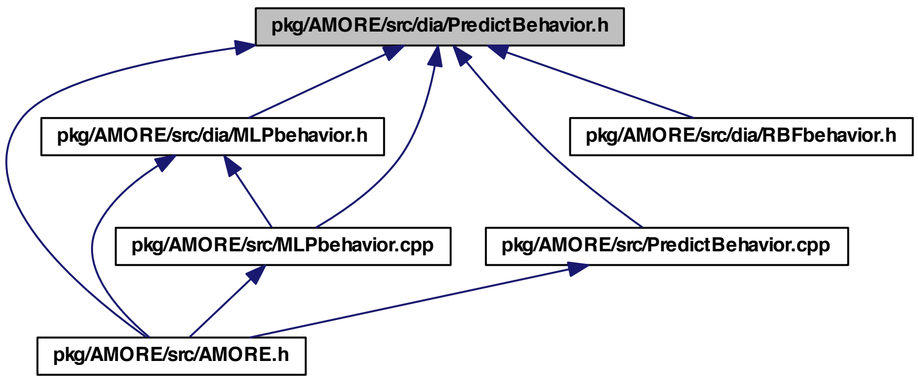
Classes

- class Neuron

  class Neuron -

6.35 pkg/AMORE/src/dia/PredictBehavior.h File Reference

This graph shows which files directly or indirectly include this file:


pkg/AMORE/src/dia/PredictBehavior.h
pkg/AMORE/src/dia/MLPbehavior.h
pkg/AMORE/src/dia/RBFbehavior.h
pkg/AMORE/src/MLPbehavior.cpp
pkg/AMORE/src/PredictBehavior.cpp
pkg/AMORE/src/AMORE.h

Classes

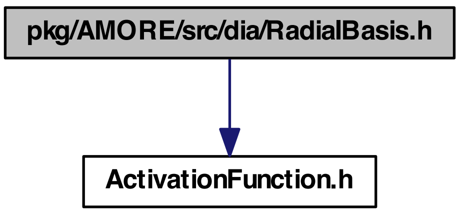
- class PredictBehavior

  class PredictBehavior -

6.36 pkg/AMORE/src/dia/RadialBasis.h File Reference

```
#include "ActivationFunction.h"
```

Include dependency graph for RadialBasis.h:



Classes

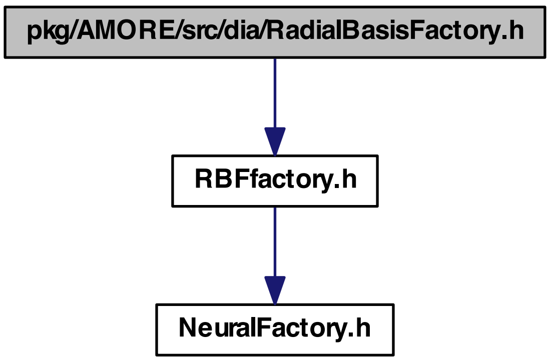
- class RadialBasis

  class RadialBasis -

6.37 pkg/AMORE/src/dia/RadialBasisFactory.h File Reference

```
#include "RBFfactory.h"
```

Include dependency graph for RadialBasisFactory.h:



Classes

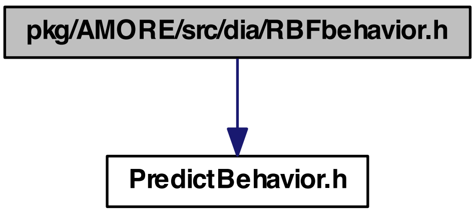
- class RadialBasisFactory

  class RadialBasisFactory -

6.38 pkg/AMORE/src/dia/RBFbehavior.h File Reference

```
#include "PredictBehavior.h"
```

Include dependency graph for RBFbehavior.h:



Classes

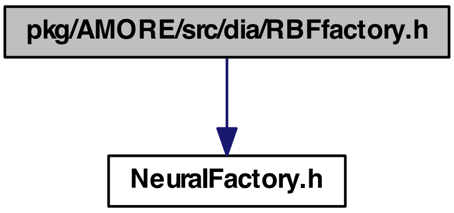
- class RBFbehavior

  class RBFbehavior -

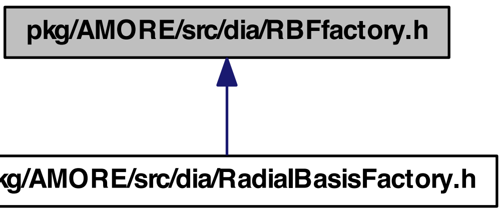
6.39 pkg/AMORE/src/dia/RBFfactory.h File Reference

```
#include "NeuralFactory.h"
```

Include dependency graph for RBFfactory.h:



This graph shows which files directly or indirectly include this file:


pkg/AMORE/src/dia/RBFfactory.h
pkg/AMORE/src/dia/RadialBasisFactory.h

Classes

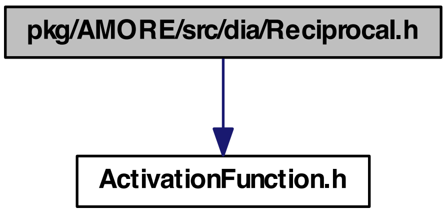
- class RBFfactory

  class RBFfactory -

6.40 pkg/AMORE/src/dia/Reciprocal.h File Reference

```
#include "ActivationFunction.h"
```

Include dependency graph for Reciprocal.h:



Classes

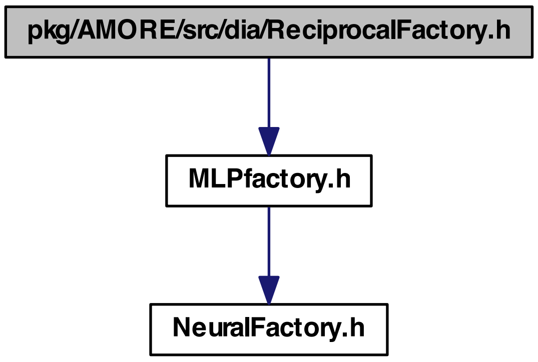
- class Reciprocal

  class Reciprocal -

6.41 pkg/AMORE/src/dia/ReciprocalFactory.h File Reference

```
#include "MLPfactory.h"
```

Include dependency graph for ReciprocalFactory.h:



Classes

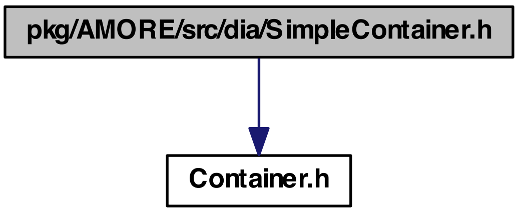
- class ReciprocalFactory

  class ReciprocalFactory -

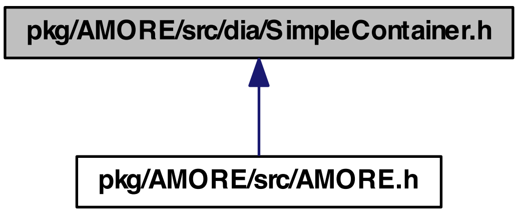
6.42 pkg/AMORE/src/dia/SimpleContainer.h File Reference

```
#include "Container.h"
```

Include dependency graph for SimpleContainer.h:



This graph shows which files directly or indirectly include this file:



Classes

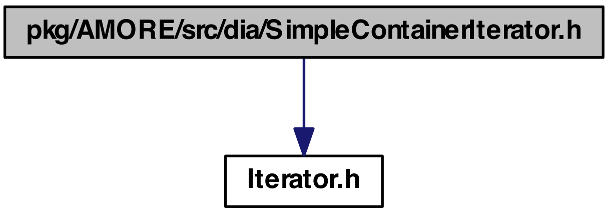
- class SimpleContainer< T >

    class SimpleContainer -

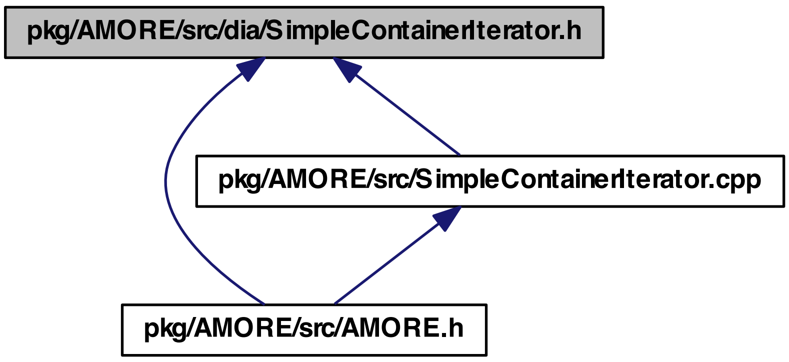
6.43 pkg/AMORE/src/dia/SimpleContainerIterator.h File Reference

```
#include "Iterator.h"
```

Include dependency graph for SimpleContainerIterator.h:



This graph shows which files directly or indirectly include this file:


pkg/AMORE/src/dia/SimpleContainerIterator.h
pkg/AMORE/src/SimpleContainerIterator.cpp
pkg/AMORE/src/AMORE.h

Classes

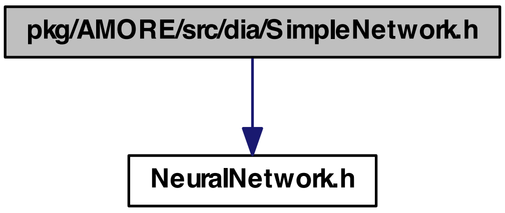
- class SimpleContainerIterator< T >

    class SimpleContainerIterator -

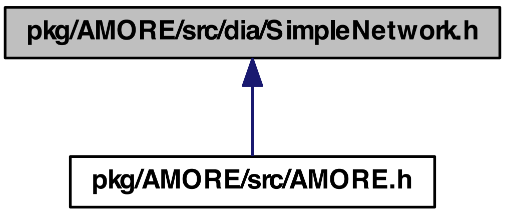
6.44 pkg/AMORE/src/dia/SimpleNetwork.h File Reference

```
#include "NeuralNetwork.h"
```

Include dependency graph for SimpleNetwork.h:



This graph shows which files directly or indirectly include this file:



Classes

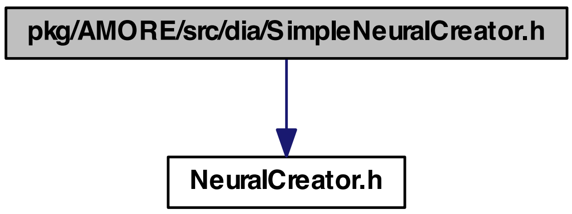
- class SimpleNetwork

    class SimpleNetwork -

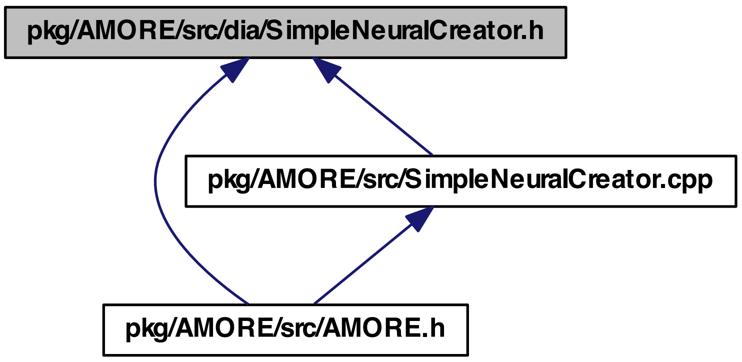
6.45 pkg/AMORE/src/dia/SimpleNeuralCreator.h File Reference

```
#include "NeuralCreator.h"
```

Include dependency graph for SimpleNeuralCreator.h:



This graph shows which files directly or indirectly include this file:


pkg/AMORE/src/dia/SimpleNeuralCreator.h
pkg/AMORE/src/SimpleNeuralCreator.cpp
pkg/AMORE/src/AMORE.h

Classes

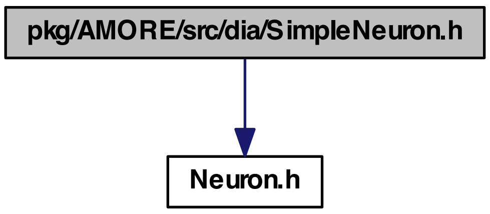
- class SimpleNeuralCreator

    class SimpleNeuralCreator -

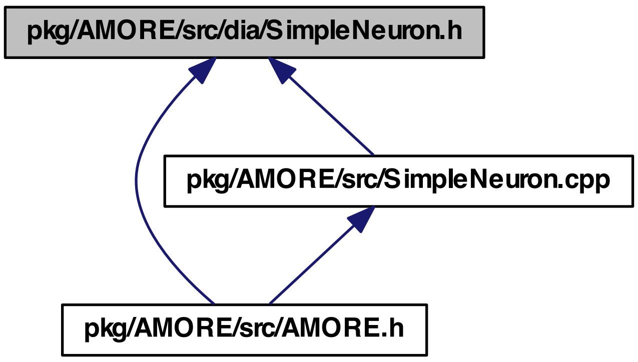
6.46 pkg/AMORE/src/dia/SimpleNeuron.h File Reference

```
#include "Neuron.h"
```

Include dependency graph for SimpleNeuron.h:



This graph shows which files directly or indirectly include this file:


pkg/AMORE/src/dia/SimpleNeuron.h
pkg/AMORE/src/SimpleNeuron.cpp
pkg/AMORE/src/AMORE.h

Classes

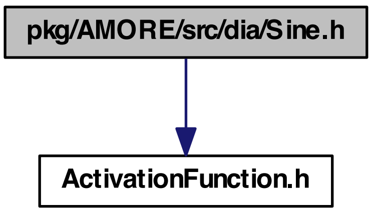
- class SimpleNeuron

  class SimpleNeuron -

6.47 pkg/AMORE/src/dia/Sine.h File Reference

```
#include "ActivationFunction.h"
```

Include dependency graph for Sine.h:



Classes

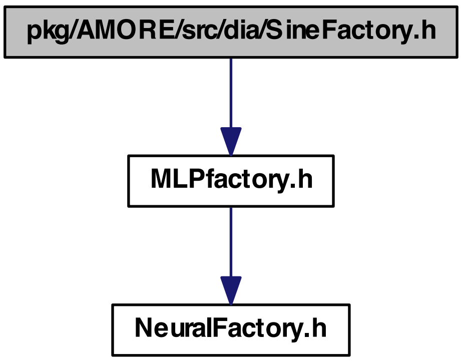
- class Sine

    class Sine -

6.48 pkg/AMORE/src/dia/SineFactory.h File Reference

```
#include "MLPfactory.h"
```

Include dependency graph for SineFactory.h:



Classes

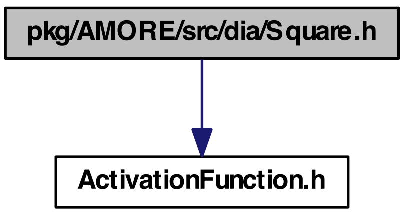
- class SineFactory

  class SineFactory -

6.49 pkg/AMORE/src/dia/Square.h File Reference

```
#include "ActivationFunction.h"
```

Include dependency graph for Square.h:



Classes

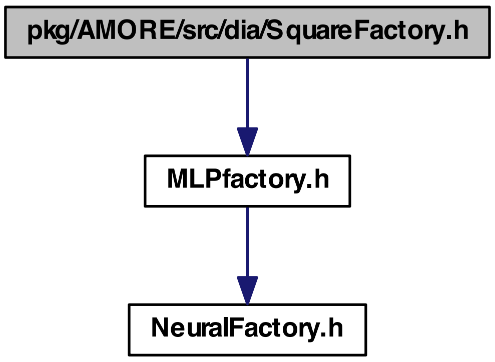
- class Square

  class Square -

6.50 pkg/AMORE/src/dia/SquareFactory.h File Reference

```
#include "MLPfactory.h"
```

Include dependency graph for SquareFactory.h:



Classes

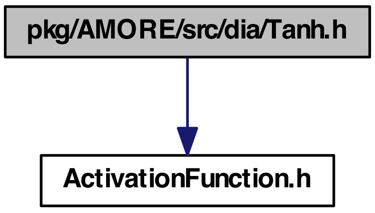
- class SquareFactory

    class SquareFactory -

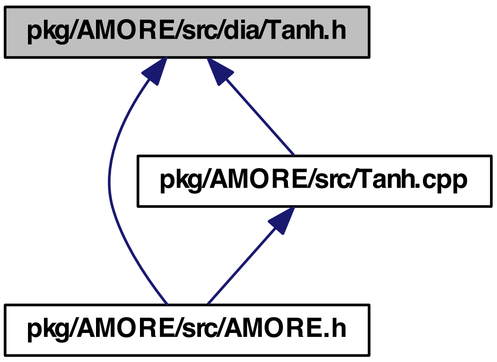
6.51 pkg/AMORE/src/dia/Tanh.h File Reference

```
#include "ActivationFunction.h"
```

Include dependency graph for Tanh.h:



This graph shows which files directly or indirectly include this file:



Classes

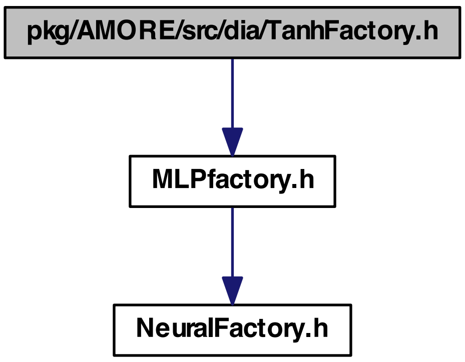
- class Tanh

  class Tanh -

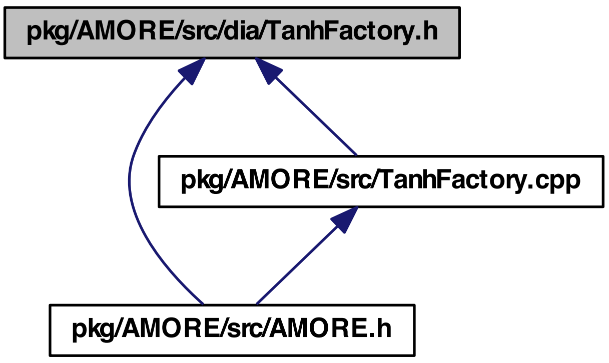
6.52 pkg/AMORE/src/dia/TanhFactory.h File Reference

```
#include "MLPfactory.h"
```

Include dependency graph for TanhFactory.h:



This graph shows which files directly or indirectly include this file:


pkg/AMORE/src/dia/TanhFactory.h
pkg/AMORE/src/TanhFactory.cpp
pkg/AMORE/src/AMORE.h

Classes

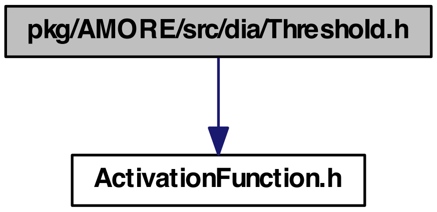
- class TanhFactory

    class TanhFactory -

6.53 pkg/AMORE/src/dia/Threshold.h File Reference

```
#include "ActivationFunction.h"
```

Include dependency graph for Threshold.h:



Classes

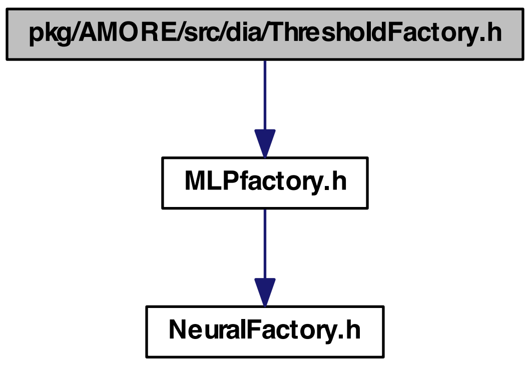
- class Threshold

  class Threshold -

6.54 pkg/AMORE/src/dia/ThresholdFactory.h File Reference

```
#include "MLPfactory.h"
```

Include dependency graph for ThresholdFactory.h:



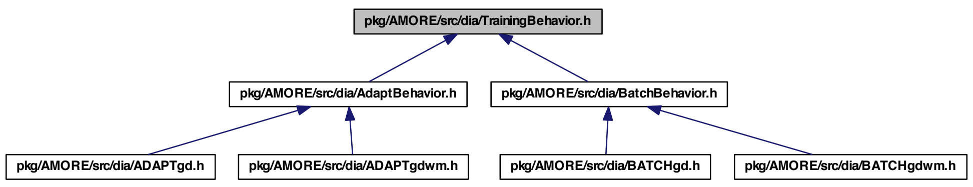
Classes

- class ThresholdFactory

  class ThresholdFactory -

6.55 pkg/AMORE/src/dia/TrainingBehavior.h File Reference

This graph shows which files directly or indirectly include this file:


pkg/AMORE/src/dia/TrainingBehavior.h
pkg/AMORE/src/dia/AdaptBehavior.h
pkg/AMORE/src/dia/BatchBehavior.h
pkg/AMORE/src/dia/ADAPTgd.h
pkg/AMORE/src/dia/ADAPTgdwm.h
pkg/AMORE/src/dia/BATCHgd.h
pkg/AMORE/src/dia/BATCHgdwm.h

Classes

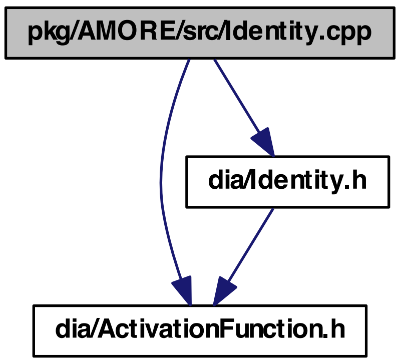
- class TrainingBehavior

  class TrainingBehavior -

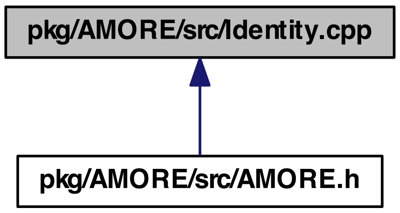
6.56 pkg/AMORE/src/Identity.cpp File Reference

```
#include "dia/ActivationFunction.h"
#include "dia/Identity.h"
```

Include dependency graph for Identity.cpp:



This graph shows which files directly or indirectly include this file:

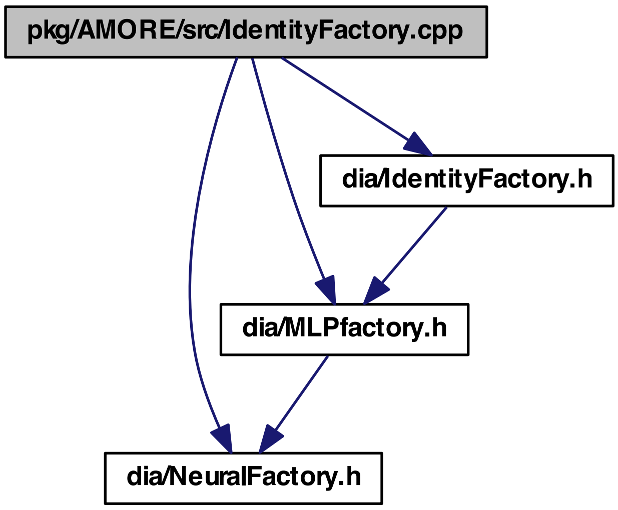


6.57 pkg/AMORE/src/IdentityFactory.cpp File Reference

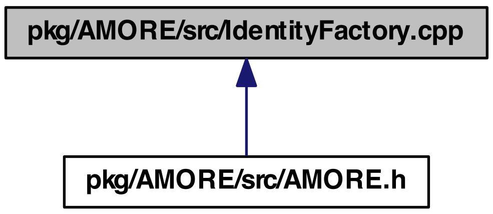
```
#include "dia/NeuralFactory.h"
```

```
#include "dia/MLPfactory.h"
#include "dia/IdentityFactory.h"
```

Include dependency graph for IdentityFactory.cpp:



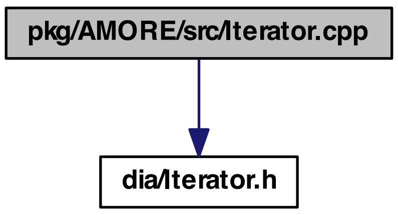
This graph shows which files directly or indirectly include this file:



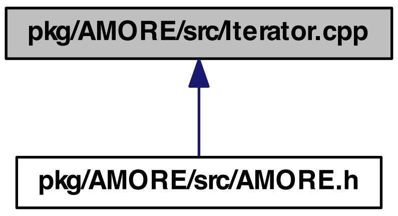
6.58 pkg/AMORE/src/Iterator.cpp File Reference

```
#include "dia/Iterator.h"
```

Include dependency graph for Iterator.cpp:



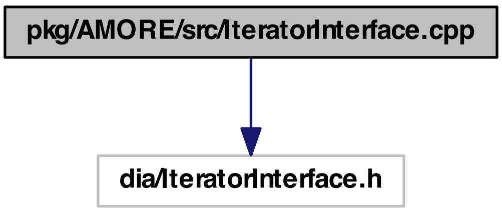
This graph shows which files directly or indirectly include this file:



6.59 pkg/AMORE/src/IteratorInterface.cpp File Reference

```
#include "dia/IteratorInterface.h"
```

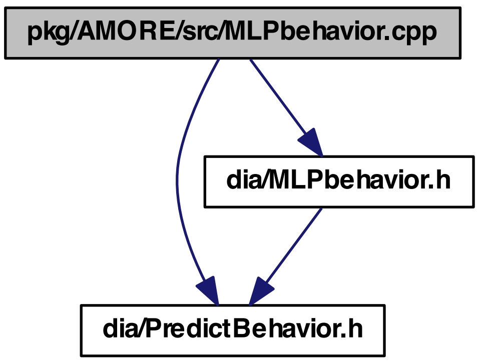
Include dependency graph for IteratorInterface.cpp:



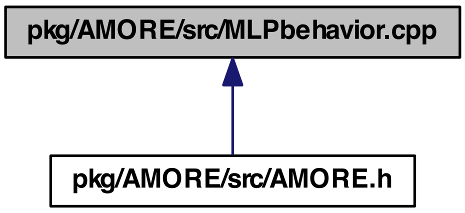
6.60 pkg/AMORE/src/MLPbehavior.cpp File Reference

```
#include "dia/PredictBehavior.h"
#include "dia/MLPbehavior.h"
```

Include dependency graph for MLPbehavior.cpp:



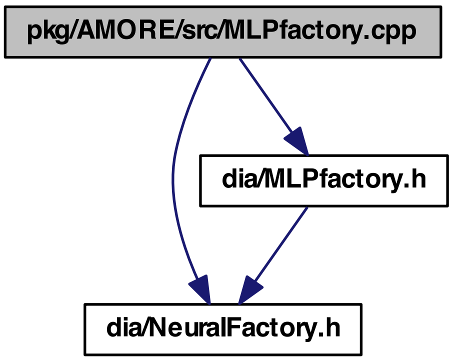
This graph shows which files directly or indirectly include this file:



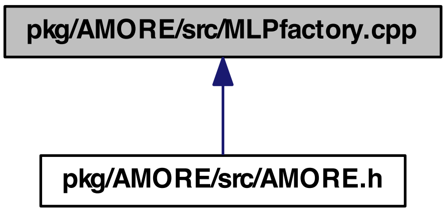
6.61 pkg/AMORE/src/MLPfactory.cpp File Reference

```
#include "dia/NeuralFactory.h"
#include "dia/MLPfactory.h"
```

Include dependency graph for MLPfactory.cpp:

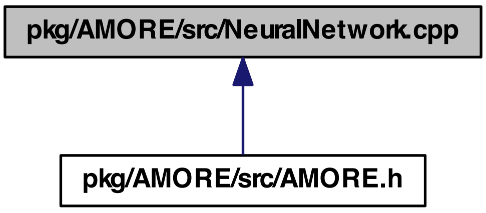


This graph shows which files directly or indirectly include this file:



6.62 pkg/AMORE/src/NeuralNetwork.cpp File Reference

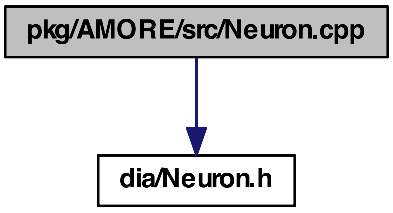
This graph shows which files directly or indirectly include this file:



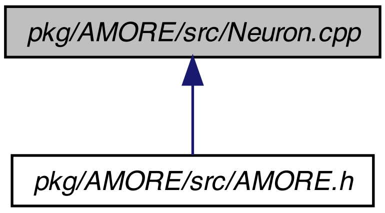
6.63 pkg/AMORE/src/Neuron.cpp File Reference

```
#include "dia/Neuron.h"
```

Include dependency graph for Neuron.cpp:



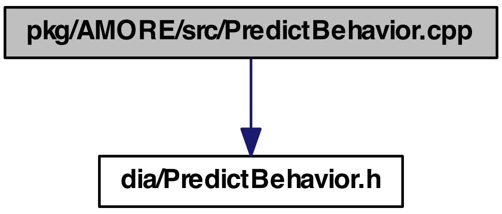
This graph shows which files directly or indirectly include this file:



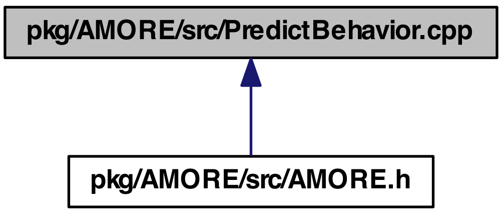
6.64 pkg/AMORE/src/PredictBehavior.cpp File Reference

```
#include "dia/PredictBehavior.h"
```

Include dependency graph for PredictBehavior.cpp:



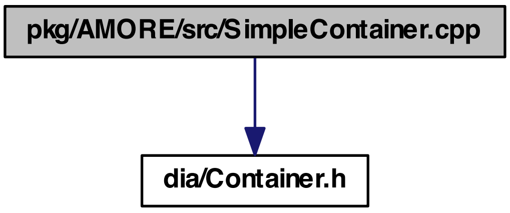
This graph shows which files directly or indirectly include this file:



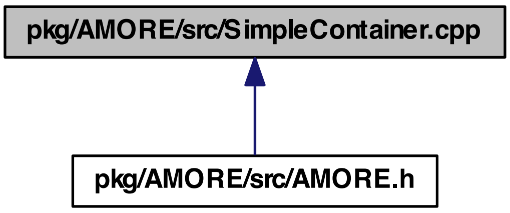
6.65 pkg/AMORE/src/SimpleContainer.cpp File Reference

```
#include "dia/Container.h"
```

Include dependency graph for SimpleContainer.cpp:



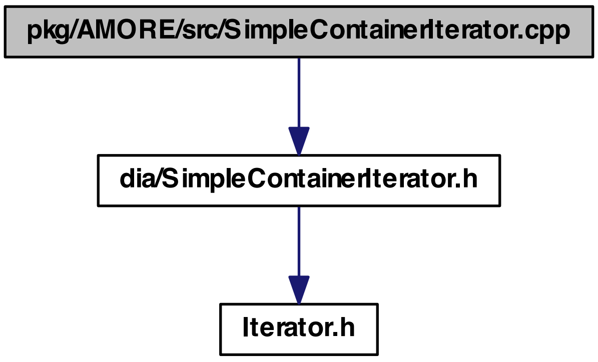
This graph shows which files directly or indirectly include this file:



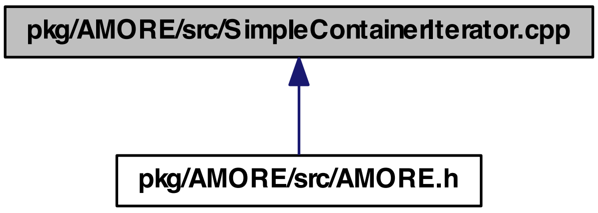
6.66 pkg/AMORE/src/SimpleContainerIterator.cpp File Reference

```
#include "dia/SimpleContainerIterator.h"
```

Include dependency graph for SimpleContainerIterator.cpp:

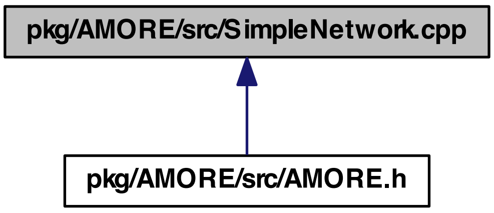

pkg/AMORE/src/SimpleContainerIterator.cpp
dia/SimpleContainerIterator.h
Iterator.h

This graph shows which files directly or indirectly include this file:



6.67 pkg/AMORE/src/SimpleNetwork.cpp File Reference

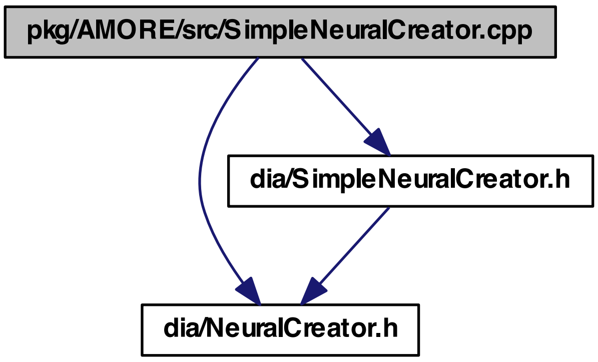
This graph shows which files directly or indirectly include this file:



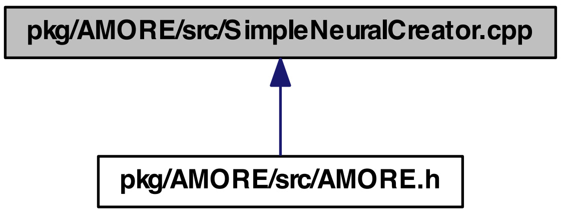
6.68 pkg/AMORE/src/SimpleNeuralCreator.cpp File Reference

```
#include "dia/NeuralCreator.h"
#include "dia/SimpleNeuralCreator.h"
```

Include dependency graph for SimpleNeuralCreator.cpp:


pkg/AMORE/src/SimpleNeuralCreator.cpp
dia/SimpleNeuralCreator.h
dia/NeuralCreator.h

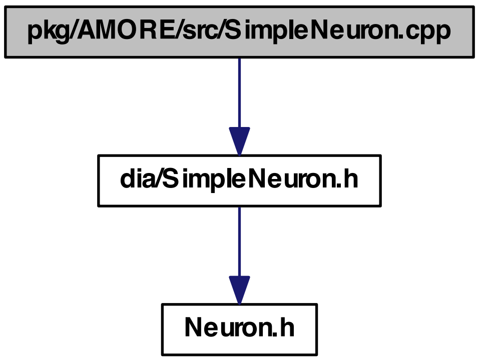
This graph shows which files directly or indirectly include this file:



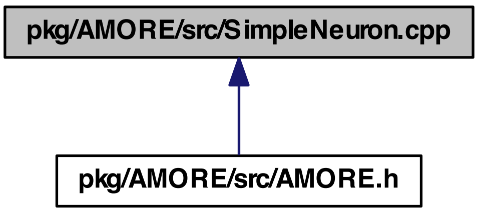
6.69 pkg/AMORE/src/SimpleNeuron.cpp File Reference

```
#include "dia/SimpleNeuron.h"
```

Include dependency graph for SimpleNeuron.cpp:



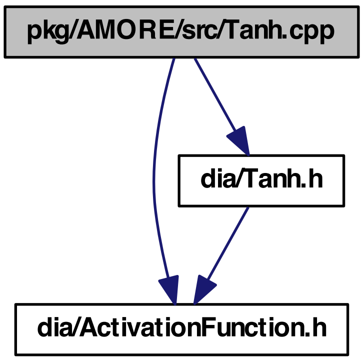
This graph shows which files directly or indirectly include this file:



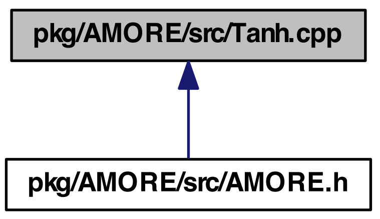
6.70 pkg/AMORE/src/Tanh.cpp File Reference

```
#include "dia/ActivationFunction.h"
#include "dia/Tanh.h"
```

Include dependency graph for Tanh.cpp:



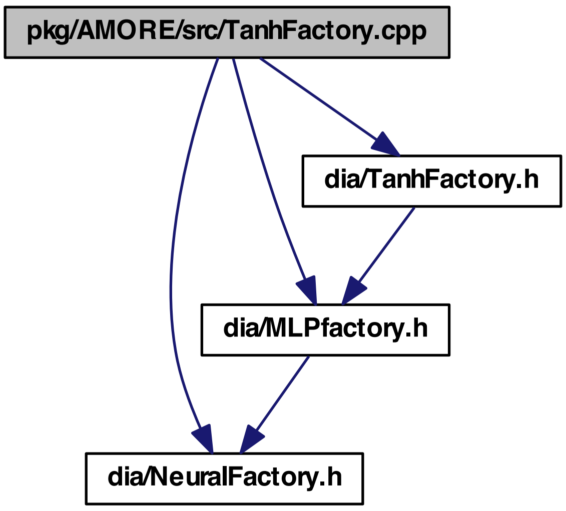
This graph shows which files directly or indirectly include this file:



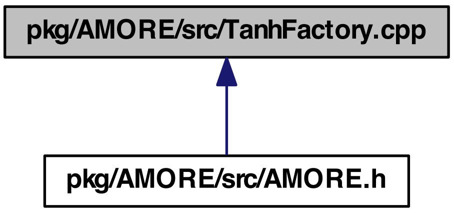
6.71 pkg/AMORE/src/TanhFactory.cpp File Reference

```
#include "dia/NeuralFactory.h"
#include "dia/MLPfactory.h"
#include "dia/TanhFactory.h"
```

Include dependency graph for TanhFactory.cpp:



This graph shows which files directly or indirectly include this file:



Index

∼Container
  Container, 40
∼Iterator
  Iterator, 73
∼SimpleContainer
  SimpleContainer, 125
∼SimpleContainerIterator
  SimpleContainerIterator, 133

ActivationFunction, 9
  ActivationFunction, 10
  d_neuron, 11
  f0, 10
  f1, 10
  getInducedLocalField, 10
ActivationFunctionPtr
  AMORE.h, 178
ActivationFunctionRef
  AMORE.h, 178
AdaptBehavior, 11
  adjustParameters, 13
ADAPTgd, 14
  adjustParameters, 15
  outputDerivative, 16
ADAPTgdwm, 16
  adjustParameters, 18
  outputDerivative, 19
addCon
  Neuron, 98
  SimpleNeuron, 142
adjustParameters
  AdaptBehavior, 13
  ADAPTgd, 15
  ADAPTgdwm, 18
  BatchBehavior, 26
  BATCHgd, 28
  BATCHgdwm, 31
  TrainingBehavior, 173
AMORE.h
  ActivationFunctionPtr, 178
  ActivationFunctionRef, 178
  ConContainerPtr, 178
  ConIteratorPtr, 179
  ConPtr, 179
  foreach, 178
  Handler, 179
  LayerPtr, 179
  NeuralCreatorPtr, 179
  NeuralFactoryPtr, 179
  NeuralNetworkPtr, 179
  NeuronContainerPtr, 179
  NeuronIteratorPtr, 179
  NeuronPtr, 179
  NeuronRef, 180
  NeuronWeakPtr, 180
  PredictBehaviorPtr, 180
  PredictBehaviorRef, 180
  size_type, 178
  TrainingBehaviorRef, 180
ArcTan, 19
  Arctan, 20
  f0, 20
  f1, 20
Arctan
  ArcTan, 20
ArcTanFactory, 21
  makeActivationFunction, 24
at
  Container, 40
  SimpleContainer, 126

BatchBehavior, 24
  adjustParameters, 26
BATCHgd, 27
  adjustParameters, 28
  outputDerivative, 29
BATCHgdwm, 29
  adjustParameters, 31
  outputDerivative, 32

clear
  Container, 40

SimpleContainer, 127
Con, 32
Con, 33
d_neuron, 38
d_weight, 38
getNeuron, 33
getWeight, 34
Id, 35
setNeuron, 36
setWeight, 36
show, 36
validate, 37
ConContainerPtr
AMORE.h, 178
ConIteratorPtr
AMORE.h, 179
ConPtr
AMORE.h, 179
Container, 38
~Container, 40
at, 40
clear, 40
Container, 40
createIterator, 40
empty, 41
push_back, 41
reserve, 41
show, 41
size, 41
validate, 41
Cosine, 41
Cosine, 43
f0, 43
f1, 44
CosineFactory, 44
makeActivationFunction, 47
createFeedForwardFullyConnectedNetwork
NeuralCreator, 91
SimpleNeuralCreator, 139
createIterator
Container, 40
SimpleContainer, 127
currentItem
Iterator, 73
SimpleContainerIterator, 133

d_activationFunction
Neuron, 100
d_altitude
RBFbehavior, 113
d_bias
MLPbehavior, 83
d_collection
SimpleContainer, 130
d_container
SimpleContainerIterator, 135
d_current
SimpleContainerIterator, 135
d_hiddenLayers
NeuralNetwork, 95
d_Id
Neuron, 100
d_inducedLocalField
Neuron, 100
d_inputLayer
NeuralNetwork, 95
d_nCons
Neuron, 100
d_neuron
ActivationFunction, 11
Con, 38
PredictBehavior, 105
d_output
Neuron, 100
d_outputLayer
NeuralNetwork, 95
d_predictBehavior
Neuron, 100
d_weight
Con, 38
d_width
RBFbehavior, 113

Elliot, 47
Elliot, 49
f0, 49
f1, 50
ElliotFactory, 50
makeActivationFunction, 53
empty
Container, 41
SimpleContainer, 127
Exponential, 53
Exponential, 55
f0, 55
f1, 56
ExponentialFactory, 56
makeActivationFunction, 59

f0

ActivationFunction, 10
ArcTan, 20
Cosine, 43
Elliot, 49
Exponential, 55
Gauss, 61
Identity, 68
Logistic, 75
RadialBasis, 106
Reciprocal, 118
Sine, 149
Square, 155
Tanh, 162
Threshold, 168

f1
ActivationFunction, 10
ArcTan, 20
Cosine, 44
Elliot, 50
Exponential, 56
Gauss, 62
Identity, 68
Logistic, 76
RadialBasis, 107
Reciprocal, 119
Sine, 150
Square, 156
Tanh, 162
Threshold, 169

first
Iterator, 73
SimpleContainerIterator, 134

foreach
AMORE.h, 178

Gauss, 59
f0, 61
f1, 62
Gauss, 61

GaussFactory, 62
makeActivationFunction, 65

getConIterator
Neuron, 98
PredictBehavior, 102
SimpleNeuron, 143

getId
Neuron, 98
SimpleNeuron, 143

getInducedLocalField
ActivationFunction, 10
Neuron, 98
SimpleNeuron, 143

getNeuron
Con, 33

getOutput
Neuron, 98
SimpleNeuron, 144

getWeight
Con, 34

Handler
AMORE.h, 179

Id
Con, 35

Identity, 65
f0, 68
f1, 68
Identity, 67

IdentityFactory, 68
makeActivationFunction, 71

isDone
Iterator, 73
SimpleContainerIterator, 134

Iterator, 71
∼Iterator, 73
currentItem, 73
first, 73
isDone, 73
Iterator, 73
next, 73

LayerPtr
AMORE.h, 179

Logistic, 74
f0, 75
f1, 76
Logistic, 75

LogisticFactory, 76
makeActivationFunction, 79

makeActivationFunction
ArcTanFactory, 24
CosineFactory, 47
ElliotFactory, 53
ExponentialFactory, 59
GaussFactory, 65
IdentityFactory, 71
LogisticFactory, 79
MLPfactory, 86

NeuralFactory, 92
RadialBasisFactory, 110
RBFfactory, 116
ReciprocalFactory, 122
SineFactory, 153
SquareFactory, 159
TanhFactory, 166
ThresholdFactory, 172
makeCon
MLPfactory, 86
NeuralFactory, 92
RBFfactory, 116
makeConContainer
MLPfactory, 87
NeuralFactory, 92
RBFfactory, 116
makeNeuron
MLPfactory, 87, 88
NeuralFactory, 92, 93
RBFfactory, 116
makeNeuronContainer
MLPfactory, 89
NeuralFactory, 93
RBFfactory, 116
makePredictBehavior
MLPfactory, 89
NeuralFactory, 93
RBFfactory, 116
MLPbehavior, 79
d_bias, 83
MLPbehavior, 82
MLPfactory, 83
predict, 82
show, 83
MLPfactory, 84
makeActivationFunction, 86
makeCon, 86
makeConContainer, 87
makeNeuron, 87, 88
makeNeuronContainer, 89
makePredictBehavior, 89
MLPbehavior, 83
MLPfactory, 86
Neuron, 100

NeuralCreator, 90
createFeedForwardFullyConnectedNet-work, 91
NeuralCreatorPtr
AMORE.h, 179
NeuralFactory, 91
makeActivationFunction, 92
makeCon, 92
makeConContainer, 92
makeNeuron, 92, 93
makeNeuronContainer, 93
makePredictBehavior, 93
NeuralFactoryPtr
AMORE.h, 179
NeuralNetwork, 93
d_hiddenLayers, 95
d_inputLayer, 95
d_outputLayer, 95
show, 94
validate, 95
NeuralNetworkPtr
AMORE.h, 179
Neuron, 95
addCon, 98
d_activationFunction, 100
d_Id, 100
d_inducedLocalField, 100
d_nCons, 100
d_output, 100
d_predictBehavior, 100
getConIterator, 98
getId, 98
getInducedLocalField, 98
getOutput, 98
MLPfactory, 100
Neuron, 98
predict, 98
setActivationFunction, 99
setId, 99
setInducedLocalField, 99
setOutput, 99
setPredictBehavior, 99
show, 99
useActivationFunctionf0, 99
validate, 99
NeuronContainerPtr
AMORE.h, 179
NeuronIteratorPtr
AMORE.h, 179
NeuronPtr
AMORE.h, 179
NeuronRef
AMORE.h, 180
NeuronWeakPtr
AMORE.h, 180

next
  Iterator, 73
  SimpleContainerIterator, 134

outputDerivative
  ADAPTgd, 16
  ADAPTgdwm, 19
  BATCHgd, 29
  BATCHgdwm, 32

pkg/AMORE/src/ActivationFunction.cpp, 175
pkg/AMORE/src/AMORE.h, 176
pkg/AMORE/src/Con.cpp, 180
pkg/AMORE/src/Container.cpp, 181
pkg/AMORE/src/dia/ActivationFunction.h, 182
pkg/AMORE/src/dia/AdaptBehavior.h, 183
pkg/AMORE/src/dia/ADAPTgd.h, 184
pkg/AMORE/src/dia/ADAPTgdwm.h, 184
pkg/AMORE/src/dia/ArcTan.h, 185
pkg/AMORE/src/dia/ArcTanFactory.h, 186
pkg/AMORE/src/dia/BatchBehavior.h, 186
pkg/AMORE/src/dia/BATCHgd.h, 187
pkg/AMORE/src/dia/BATCHgdwm.h, 188
pkg/AMORE/src/dia/Con.h, 190
pkg/AMORE/src/dia/Container.h, 190
pkg/AMORE/src/dia/Cosine.h, 191
pkg/AMORE/src/dia/CosineFactory.h, 191
pkg/AMORE/src/dia/Elliot.h, 192
pkg/AMORE/src/dia/ElliotFactory.h, 193
pkg/AMORE/src/dia/Exponential.h, 193
pkg/AMORE/src/dia/ExponentialFactory.h, 194
pkg/AMORE/src/dia/Gauss.h, 195
pkg/AMORE/src/dia/GaussFactory.h, 195
pkg/AMORE/src/dia/Identity.h, 196
pkg/AMORE/src/dia/IdentityFactory.h, 197
pkg/AMORE/src/dia/Iterator.h, 199
pkg/AMORE/src/dia/Logistic.h, 199
pkg/AMORE/src/dia/LogisticFactory.h, 200
pkg/AMORE/src/dia/MLPbehavior.h, 201
pkg/AMORE/src/dia/MLPfactory.h, 202
pkg/AMORE/src/dia/NeuralCreator.h, 203
pkg/AMORE/src/dia/NeuralFactory.h, 203
pkg/AMORE/src/dia/NeuralNetwork.h, 204
pkg/AMORE/src/dia/Neuron.h, 205
pkg/AMORE/src/dia/PredictBehavior.h, 206
pkg/AMORE/src/dia/RadialBasis.h, 206
pkg/AMORE/src/dia/RadialBasisFactory.h, 207
pkg/AMORE/src/dia/RBFbehavior.h, 207
pkg/AMORE/src/dia/RBFfactory.h, 208
pkg/AMORE/src/dia/Reciprocal.h, 209
pkg/AMORE/src/dia/ReciprocalFactory.h, 210
pkg/AMORE/src/dia/SimpleContainer.h, 210
pkg/AMORE/src/dia/SimpleContainerIterator.h, 211
pkg/AMORE/src/dia/SimpleNetwork.h, 212
pkg/AMORE/src/dia/SimpleNeuralCreator.h, 213
pkg/AMORE/src/dia/SimpleNeuron.h, 214
pkg/AMORE/src/dia/Sine.h, 215
pkg/AMORE/src/dia/SineFactory.h, 216
pkg/AMORE/src/dia/Square.h, 217
pkg/AMORE/src/dia/SquareFactory.h, 217
pkg/AMORE/src/dia/Tanh.h, 218
pkg/AMORE/src/dia/TanhFactory.h, 219
pkg/AMORE/src/dia/Threshold.h, 221
pkg/AMORE/src/dia/ThresholdFactory.h, 221
pkg/AMORE/src/dia/TrainingBehavior.h, 222
pkg/AMORE/src/Identity.cpp, 223
pkg/AMORE/src/IdentityFactory.cpp, 223
pkg/AMORE/src/Iterator.cpp, 225
pkg/AMORE/src/IteratorInterface.cpp, 225
pkg/AMORE/src/MLPbehavior.cpp, 226
pkg/AMORE/src/MLPfactory.cpp, 227
pkg/AMORE/src/NeuralNetwork.cpp, 228
pkg/AMORE/src/Neuron.cpp, 228
pkg/AMORE/src/PredictBehavior.cpp, 229
pkg/AMORE/src/SimpleContainer.cpp, 230
pkg/AMORE/src/SimpleContainerIterator.cpp, 231
pkg/AMORE/src/SimpleNetwork.cpp, 233
pkg/AMORE/src/SimpleNeuralCreator.cpp, 233
pkg/AMORE/src/SimpleNeuron.cpp, 234
pkg/AMORE/src/Tanh.cpp, 235
pkg/AMORE/src/TanhFactory.cpp, 236
predict
  MLPbehavior, 82
  Neuron, 98
  PredictBehavior, 103
  RBFbehavior, 113
  SimpleNeuron, 144
PredictBehavior, 101
  d_neuron, 105
  getConIterator, 102
  predict, 103
  PredictBehavior, 102
  setInducedLocalField, 103
  setOutput, 103
  show, 104

useActivationFunctionf0, 104
PredictBehaviorPtr
AMORE.h, 180
PredictBehaviorRef
AMORE.h, 180
push_back
Container, 41
SimpleContainer, 127

RadialBasis, 105
f0, 106
f1, 107
RadialBasis, 106
RadialBasisFactory, 107
makeActivationFunction, 110
RBFbehavior, 110
d_altitude, 113
d_width, 113
predict, 113
RBFbehavior, 113
show, 113
RBFfactory, 113
makeActivationFunction, 116
makeCon, 116
makeConContainer, 116
makeNeuron, 116
makeNeuronContainer, 116
makePredictBehavior, 116
RBFfactory, 116
Reciprocal, 117
f0, 118
f1, 119
Reciprocal, 118
ReciprocalFactory, 119
makeActivationFunction, 122
reserve
Container, 41
SimpleContainer, 128

setActivationFunction
Neuron, 99
SimpleNeuron, 144
setId
Neuron, 99
SimpleNeuron, 144
setInducedLocalField
Neuron, 99
PredictBehavior, 103
SimpleNeuron, 145
setNeuron
Con, 36
setOutput
Neuron, 99
PredictBehavior, 103
SimpleNeuron, 145
setPredictBehavior
Neuron, 99
SimpleNeuron, 145
setWeight
Con, 36
show
Con, 36
Container, 41
MLPbehavior, 83
NeuralNetwork, 94
Neuron, 99
PredictBehavior, 104
RBFbehavior, 113
SimpleContainer, 128
SimpleNetwork, 136
SimpleNeuron, 145
SimpleContainer, 122
∼SimpleContainer, 125
at, 126
clear, 127
createIterator, 127
d_collection, 130
empty, 127
push_back, 127
reserve, 128
show, 128
SimpleContainer, 125
SimpleContainerIterator< T >, 130
size, 129
validate, 129
SimpleContainer< T >
SimpleContainerIterator, 134
SimpleContainerIterator, 130
∼SimpleContainerIterator, 133
currentItem, 133
d_container, 135
d_current, 135
first, 134
isDone, 134
next, 134
SimpleContainer< T >, 134
SimpleContainerIterator, 133
SimpleContainerIterator< T >
SimpleContainer, 130
SimpleNetwork, 135

show, 136
validate, 136
SimpleNeuralCreator, 137
createFeedForwardFullyConnectedNetwork, 139
SimpleNeuralCreator, 138
SimpleNeuron, 139
addCon, 142
getConIterator, 143
getId, 143
getInducedLocalField, 143
getOutput, 144
predict, 144
setActivationFunction, 144
setId, 144
setInducedLocalField, 145
setOutput, 145
setPredictBehavior, 145
show, 145
SimpleNeuron, 142
useActivationFunctionf0, 146
validate, 146
Sine, 147
f0, 149
f1, 150
Sine, 149
SineFactory, 150
makeActivationFunction, 153
size
Container, 41
SimpleContainer, 129
size_type
AMORE.h, 178
Square, 153
f0, 155
f1, 156
Square, 155
SquareFactory, 156
makeActivationFunction, 159

Tanh, 159
f0, 162
f1, 162
Tanh, 161
TanhFactory, 163
makeActivationFunction, 166
Threshold, 166
f0, 168
f1, 169
Threshold, 168
ThresholdFactory, 169
makeActivationFunction, 172
TrainingBehavior, 172
adjustParameters, 173
TrainingBehaviorRef
AMORE.h, 180

useActivationFunctionf0
Neuron, 99
PredictBehavior, 104
SimpleNeuron, 146

validate
Con, 37
Container, 41
NeuralNetwork, 95
Neuron, 99
SimpleContainer, 129
SimpleNetwork, 136
SimpleNeuron, 146