

AMORE++

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

Generated by Doxygen 1.7.4

Sat Jul 30 2011 22:48:57



# Contents

<b>1</b>	<b>The AMORE++ package</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Motivation . . . . .	1
1.3	Road Map . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>File Index</b>	<b>9</b>
4.1	File List . . . . .	9
<b>5</b>	<b>Class Documentation</b>	<b>15</b>
5.1	ActivationFunction Class Reference . . . . .	15
5.1.1	Detailed Description . . . . .	16
5.1.2	Constructor & Destructor Documentation . . . . .	16
5.1.2.1	ActivationFunction . . . . .	16
5.1.3	Member Function Documentation . . . . .	16
5.1.3.1	f0 . . . . .	16
5.1.3.2	f1 . . . . .	16
5.1.3.3	getInducedLocalField . . . . .	16
5.1.4	Member Data Documentation . . . . .	17
5.1.4.1	d_neuron . . . . .	17
5.2	ADAPTgdHiddenNeuronTrainBehavior Class Reference . . . . .	17
5.2.1	Detailed Description . . . . .	20

5.2.2	Member Function Documentation	20
5.2.2.1	endOfEpochAction	20
5.2.2.2	singlePatternBackwardAction	20
5.3	ADAPTgdNetworkTrainBehavior Class Reference	20
5.3.1	Detailed Description	22
5.3.2	Member Function Documentation	22
5.3.2.1	train	22
5.4	ADAPTgdNeuronTrainBehavior Class Reference	23
5.4.1	Detailed Description	26
5.4.2	Member Function Documentation	26
5.4.2.1	endOfEpochAction	26
5.4.2.2	singlePatternBackwardAction	26
5.4.3	Member Data Documentation	26
5.4.3.1	d_delta	26
5.4.3.2	d_learningRate	26
5.5	ADAPTgdOutputNeuronTrainBehavior Class Reference	26
5.5.1	Detailed Description	29
5.5.2	Member Function Documentation	29
5.5.2.1	endOfEpochAction	29
5.5.2.2	singlePatternBackwardAction	29
5.5.3	Member Data Documentation	29
5.5.3.1	d_costFunction	29
5.6	ADAPTgdwmHiddenNeuronTrainBehavior Class Reference	29
5.6.1	Detailed Description	32
5.6.2	Member Function Documentation	32
5.6.2.1	endOfEpochAction	32
5.6.2.2	singlePatternBackwardAction	32
5.7	ADAPTgdwmNetworkTrainBehavior Class Reference	32
5.7.1	Detailed Description	34
5.7.2	Member Function Documentation	34
5.7.2.1	train	35
5.8	ADAPTgdwmNeuronTrainBehavior Class Reference	35
5.8.1	Detailed Description	38
5.8.2	Member Function Documentation	38

5.8.2.1	endOfEpochAction . . . . .	38
5.8.2.2	singlePatternBackwardAction . . . . .	38
5.8.3	Member Data Documentation . . . . .	38
5.8.3.1	d_delta . . . . .	38
5.8.3.2	d_formerBiasChange . . . . .	39
5.8.3.3	d_formerWeightChange . . . . .	39
5.8.3.4	d_learningRate . . . . .	39
5.8.3.5	d_momentum . . . . .	39
5.9	ADAPTgdwmOutputNeuronTrainBehavior Class Reference . . . . .	39
5.9.1	Detailed Description . . . . .	42
5.9.2	Member Function Documentation . . . . .	42
5.9.2.1	endOfEpochAction . . . . .	42
5.9.2.2	singlePatternBackwardAction . . . . .	42
5.9.3	Member Data Documentation . . . . .	42
5.9.3.1	d_costFunction . . . . .	42
5.10	AdaptNetworkTrainBehavior Class Reference . . . . .	42
5.10.1	Detailed Description . . . . .	44
5.10.2	Member Function Documentation . . . . .	44
5.10.2.1	train . . . . .	44
5.11	AdaptNeuronTrainBehavior Class Reference . . . . .	45
5.11.1	Detailed Description . . . . .	46
5.11.2	Member Function Documentation . . . . .	46
5.11.2.1	endOfEpochAction . . . . .	46
5.11.2.2	singlePatternBackwardAction . . . . .	47
5.12	ArcTan Class Reference . . . . .	47
5.12.1	Detailed Description . . . . .	48
5.12.2	Member Function Documentation . . . . .	48
5.12.2.1	Arctan . . . . .	48
5.12.2.2	f0 . . . . .	48
5.12.2.3	f1 . . . . .	49
5.13	ArcTanFactory Class Reference . . . . .	49
5.13.1	Detailed Description . . . . .	52
5.13.2	Constructor & Destructor Documentation . . . . .	52
5.13.2.1	ArcTanFactory . . . . .	52

5.13.3	Member Function Documentation	52
5.13.3.1	makeActivationFunction	52
5.14	BATCHgdHiddenNeuronTrainBehavior Class Reference	52
5.14.1	Detailed Description	55
5.14.2	Member Function Documentation	55
5.14.2.1	endOfEpochAction	55
5.14.2.2	singlePatternBackwardAction	55
5.15	BATCHgdNetworkTrainBehavior Class Reference	55
5.15.1	Detailed Description	57
5.15.2	Member Function Documentation	57
5.15.2.1	train	58
5.16	BATCHgdNeuronTrainBehavior Class Reference	58
5.16.1	Detailed Description	61
5.16.2	Member Function Documentation	61
5.16.2.1	endOfEpochAction	61
5.16.2.2	singlePatternBackwardAction	61
5.16.3	Member Data Documentation	61
5.16.3.1	d_delta	61
5.16.3.2	d_learningRate	61
5.16.3.3	d_sum_delta_bias	62
5.16.3.4	d_sum_delta_x	62
5.17	BATCHgdOutputNeuronTrainBehavior Class Reference	62
5.17.1	Detailed Description	65
5.17.2	Member Function Documentation	65
5.17.2.1	endOfEpochAction	65
5.17.2.2	singlePatternBackwardAction	65
5.17.3	Member Data Documentation	65
5.17.3.1	d_costFunction	65
5.18	BATCHgdwmHiddenNeuronTrainBehavior Class Reference	65
5.18.1	Detailed Description	68
5.18.2	Member Function Documentation	68
5.18.2.1	endOfEpochAction	68
5.18.2.2	singlePatternBackwardAction	68
5.19	BATCHgdwmNetworkTrainBehavior Class Reference	68

5.19.1 Detailed Description . . . . .	70
5.19.2 Member Function Documentation . . . . .	70
5.19.2.1 train . . . . .	71
5.20 BATCHGdwmNeuronTrainBehavior Class Reference . . . . .	71
5.20.1 Detailed Description . . . . .	74
5.20.2 Member Function Documentation . . . . .	74
5.20.2.1 endOfEpochAction . . . . .	74
5.20.2.2 singlePatternBackwardAction . . . . .	74
5.20.3 Member Data Documentation . . . . .	74
5.20.3.1 bias . . . . .	74
5.20.3.2 change . . . . .	75
5.20.3.3 change . . . . .	75
5.20.3.4 delta . . . . .	75
5.20.3.5 momentum . . . . .	75
5.20.3.6 rate . . . . .	75
5.20.3.7 x . . . . .	75
5.21 BATCHGdwmOutputNeuronTrainBehavior Class Reference . . . . .	75
5.21.1 Detailed Description . . . . .	78
5.21.2 Member Function Documentation . . . . .	78
5.21.2.1 endOfEpochAction . . . . .	78
5.21.2.2 singlePatternBackwardAction . . . . .	78
5.21.3 Member Data Documentation . . . . .	78
5.21.3.1 d_costFunction . . . . .	78
5.22 BatchNetworkTrainBehavior Class Reference . . . . .	78
5.22.1 Detailed Description . . . . .	80
5.22.2 Member Function Documentation . . . . .	80
5.22.2.1 train . . . . .	80
5.23 BatchNeuronTrainBehavior Class Reference . . . . .	81
5.23.1 Detailed Description . . . . .	82
5.23.2 Member Function Documentation . . . . .	82
5.23.2.1 endOfEpochAction . . . . .	82
5.23.2.2 singlePatternBackwardAction . . . . .	83
5.24 Con Class Reference . . . . .	83
5.24.1 Detailed Description . . . . .	84

5.24.2	Constructor & Destructor Documentation . . . . .	84
5.24.2.1	Con . . . . .	84
5.24.2.2	Con . . . . .	84
5.24.3	Member Function Documentation . . . . .	84
5.24.3.1	getNeuron . . . . .	84
5.24.3.2	getWeight . . . . .	85
5.24.3.3	ld . . . . .	86
5.24.3.4	setNeuron . . . . .	87
5.24.3.5	setWeight . . . . .	87
5.24.3.6	show . . . . .	87
5.24.3.7	validate . . . . .	88
5.24.4	Member Data Documentation . . . . .	89
5.24.4.1	d_neuron . . . . .	89
5.24.4.2	d_weight . . . . .	89
5.25	Container< T > Class Template Reference . . . . .	89
5.25.1	Detailed Description . . . . .	91
5.25.2	Constructor & Destructor Documentation . . . . .	91
5.25.2.1	~Container . . . . .	91
5.25.2.2	Container . . . . .	91
5.25.3	Member Function Documentation . . . . .	91
5.25.3.1	at . . . . .	91
5.25.3.2	clear . . . . .	91
5.25.3.3	createIterator . . . . .	91
5.25.3.4	createReverseIterator . . . . .	92
5.25.3.5	empty . . . . .	92
5.25.3.6	push_back . . . . .	92
5.25.3.7	reserve . . . . .	92
5.25.3.8	show . . . . .	92
5.25.3.9	size . . . . .	92
5.25.3.10	validate . . . . .	92
5.26	Cosine Class Reference . . . . .	92
5.26.1	Detailed Description . . . . .	94
5.26.2	Constructor & Destructor Documentation . . . . .	94
5.26.2.1	Cosine . . . . .	94



5.26.3	Member Function Documentation	94
5.26.3.1	f0	95
5.26.3.2	f1	95
5.27	CosineFactory Class Reference	95
5.27.1	Detailed Description	98
5.27.2	Constructor & Destructor Documentation	98
5.27.2.1	CosineFactory	98
5.27.3	Member Function Documentation	98
5.27.3.1	makeActivationFunction	98
5.28	CostFunction Class Reference	98
5.28.1	Detailed Description	99
5.28.2	Member Function Documentation	99
5.28.2.1	f0	99
5.28.2.2	f1	99
5.29	Elliot Class Reference	100
5.29.1	Detailed Description	101
5.29.2	Constructor & Destructor Documentation	101
5.29.2.1	Elliot	101
5.29.3	Member Function Documentation	101
5.29.3.1	f0	102
5.29.3.2	f1	102
5.30	ElliotFactory Class Reference	102
5.30.1	Detailed Description	105
5.30.2	Constructor & Destructor Documentation	105
5.30.2.1	ElliotFactory	105
5.30.3	Member Function Documentation	105
5.30.3.1	makeActivationFunction	105
5.31	Exponential Class Reference	105
5.31.1	Detailed Description	107
5.31.2	Constructor & Destructor Documentation	107
5.31.2.1	Exponential	107
5.31.3	Member Function Documentation	107
5.31.3.1	f0	108
5.31.3.2	f1	108

5.32 ExponentialFactory Class Reference . . . . .	108
5.32.1 Detailed Description . . . . .	111
5.32.2 Constructor & Destructor Documentation . . . . .	111
5.32.2.1 ExponentialFactory . . . . .	111
5.32.3 Member Function Documentation . . . . .	111
5.32.3.1 makeActivationFunction . . . . .	111
5.33 Gauss Class Reference . . . . .	111
5.33.1 Detailed Description . . . . .	113
5.33.2 Constructor & Destructor Documentation . . . . .	113
5.33.2.1 Gauss . . . . .	113
5.33.3 Member Function Documentation . . . . .	113
5.33.3.1 f0 . . . . .	114
5.33.3.2 f1 . . . . .	114
5.34 GaussFactory Class Reference . . . . .	114
5.34.1 Detailed Description . . . . .	117
5.34.2 Constructor & Destructor Documentation . . . . .	117
5.34.2.1 GaussFactory . . . . .	117
5.34.3 Member Function Documentation . . . . .	117
5.34.3.1 makeActivationFunction . . . . .	117
5.35 Identity Class Reference . . . . .	117
5.35.1 Detailed Description . . . . .	119
5.35.2 Constructor & Destructor Documentation . . . . .	119
5.35.2.1 Identity . . . . .	119
5.35.3 Member Function Documentation . . . . .	120
5.35.3.1 f0 . . . . .	120
5.35.3.2 f1 . . . . .	120
5.36 IdentityFactory Class Reference . . . . .	120
5.36.1 Detailed Description . . . . .	123
5.36.2 Constructor & Destructor Documentation . . . . .	123
5.36.2.1 IdentityFactory . . . . .	123
5.36.3 Member Function Documentation . . . . .	123
5.36.3.1 makeActivationFunction . . . . .	123
5.37 Iterator< T > Class Template Reference . . . . .	123
5.37.1 Detailed Description . . . . .	124

5.37.2	Constructor & Destructor Documentation	125
5.37.2.1	~Iterator	125
5.37.2.2	Iterator	125
5.37.3	Member Function Documentation	125
5.37.3.1	currentItem	125
5.37.3.2	first	125
5.37.3.3	isDone	125
5.37.3.4	next	125
5.38	LMLS Class Reference	125
5.38.1	Detailed Description	127
5.38.2	Member Function Documentation	127
5.38.2.1	f0	127
5.38.2.2	f1	127
5.39	LMS Class Reference	128
5.39.1	Detailed Description	129
5.39.2	Member Function Documentation	129
5.39.2.1	f0	129
5.39.2.2	f1	129
5.40	Logistic Class Reference	130
5.40.1	Detailed Description	131
5.40.2	Constructor & Destructor Documentation	131
5.40.2.1	Logistic	131
5.40.3	Member Function Documentation	131
5.40.3.1	f0	132
5.40.3.2	f1	132
5.41	LogisticFactory Class Reference	132
5.41.1	Detailed Description	135
5.41.2	Constructor & Destructor Documentation	135
5.41.2.1	LogisticFactory	135
5.41.3	Member Function Documentation	135
5.41.3.1	makeActivationFunction	135
5.42	MLPbehavior Class Reference	135
5.42.1	Detailed Description	138
5.42.2	Constructor & Destructor Documentation	138

5.42.2.1	MLPbehavior	138
5.42.3	Member Function Documentation	138
5.42.3.1	show	138
5.42.3.2	singlePatternForwardAction	138
5.42.4	Friends And Related Function Documentation	139
5.42.4.1	MLPfactory	139
5.42.5	Member Data Documentation	139
5.42.5.1	d_bias	139
5.43	MLPfactory Class Reference	140
5.43.1	Detailed Description	142
5.43.2	Member Function Documentation	142
5.43.2.1	makeActivationFunction	142
5.43.2.2	makeCon	142
5.43.2.3	makeConContainer	143
5.43.2.4	makeLayer	143
5.43.2.5	makeLayerContainer	144
5.43.2.6	makeNeuralCreator	144
5.43.2.7	makeNeuralNetwork	144
5.43.2.8	makeNeuron	145
5.43.2.9	makeNeuron	145
5.43.2.10	makePredictBehavior	146
5.44	NetworkRinterface Class Reference	147
5.44.1	Detailed Description	147
5.44.2	Constructor & Destructor Documentation	148
5.44.2.1	NetworkRinterface	148
5.44.3	Member Function Documentation	148
5.44.3.1	createFeedForwardNetwork	148
5.44.3.2	inputSize	148
5.44.3.3	outputSize	149
5.44.3.4	predict	149
5.44.3.5	show	151
5.44.3.6	train	151
5.44.3.7	validate	152
5.44.4	Member Data Documentation	152

5.44.4.1	d_neuralNetwork	152
5.45	NetworkTrainBehavior Class Reference	153
5.45.1	Detailed Description	153
5.45.2	Member Function Documentation	153
5.45.2.1	train	154
5.45.3	Member Data Documentation	154
5.45.3.1	d_costFunction	154
5.45.3.2	d_neuralNetwork	154
5.46	NeuralCreator Class Reference	154
5.46.1	Detailed Description	155
5.46.2	Member Function Documentation	155
5.46.2.1	createFeedForwardNetwork	155
5.47	NeuralFactory Class Reference	156
5.47.1	Detailed Description	156
5.47.2	Member Function Documentation	156
5.47.2.1	makeActivationFunction	156
5.47.2.2	makeCon	157
5.47.2.3	makeConContainer	157
5.47.2.4	makeLayer	157
5.47.2.5	makeLayerContainer	157
5.47.2.6	makeNeuralCreator	158
5.47.2.7	makeNeuralNetwork	158
5.47.2.8	makeNeuron	158
5.47.2.9	makeNeuron	159
5.47.2.10	makePredictBehavior	159
5.48	NeuralNetwork Class Reference	159
5.48.1	Detailed Description	161
5.48.2	Constructor & Destructor Documentation	161
5.48.2.1	NeuralNetwork	161
5.48.3	Member Function Documentation	162
5.48.3.1	inputSize	162
5.48.3.2	outputSize	162
5.48.3.3	readOutput	162
5.48.3.4	show	162

5.48.3.5	<a href="#">singlePatternBackwardAction</a>	162
5.48.3.6	<a href="#">singlePatternForwardAction</a>	162
5.48.3.7	<a href="#">train</a>	163
5.48.3.8	<a href="#">validate</a>	163
5.48.3.9	<a href="#">writeInput</a>	163
5.48.4	<a href="#">Friends And Related Function Documentation</a>	163
5.48.4.1	<a href="#">SimpleNeuralCreator</a>	163
5.48.5	<a href="#">Member Data Documentation</a>	163
5.48.5.1	<a href="#">d_hiddenLayers</a>	163
5.48.5.2	<a href="#">d_inputLayer</a>	163
5.48.5.3	<a href="#">d_networkTrainBehavior</a>	163
5.48.5.4	<a href="#">d_outputLayer</a>	164
5.49	<a href="#">Neuron Class Reference</a>	164
5.49.1	<a href="#">Detailed Description</a>	166
5.49.2	<a href="#">Constructor &amp; Destructor Documentation</a>	167
5.49.2.1	<a href="#">Neuron</a>	167
5.49.3	<a href="#">Member Function Documentation</a>	167
5.49.3.1	<a href="#">addCon</a>	167
5.49.3.2	<a href="#">getConIterator</a>	167
5.49.3.3	<a href="#">getId</a>	167
5.49.3.4	<a href="#">getInducedLocalField</a>	167
5.49.3.5	<a href="#">getOutput</a>	167
5.49.3.6	<a href="#">setActivationFunction</a>	168
5.49.3.7	<a href="#">setId</a>	168
5.49.3.8	<a href="#">setInducedLocalField</a>	168
5.49.3.9	<a href="#">setOutput</a>	168
5.49.3.10	<a href="#">setOutputDerivative</a>	168
5.49.3.11	<a href="#">setPredictBehavior</a>	168
5.49.3.12	<a href="#">show</a>	168
5.49.3.13	<a href="#">singlePatternBackwardAction</a>	168
5.49.3.14	<a href="#">singlePatternForwardAction</a>	168
5.49.3.15	<a href="#">useActivationFunction0</a>	168
5.49.3.16	<a href="#">useActivationFunction1</a>	169
5.49.3.17	<a href="#">validate</a>	169

5.49.4	Friends And Related Function Documentation . . . . .	169
5.49.4.1	MLPfactory . . . . .	169
5.49.5	Member Data Documentation . . . . .	169
5.49.5.1	d_activationFunction . . . . .	169
5.49.5.2	d_Id . . . . .	169
5.49.5.3	d_inducedLocalField . . . . .	169
5.49.5.4	d_nCons . . . . .	169
5.49.5.5	d_neuronTrainBehavior . . . . .	169
5.49.5.6	d_output . . . . .	170
5.49.5.7	d_outputDerivative . . . . .	170
5.49.5.8	d_predictBehavior . . . . .	170
5.50	NeuronTrainBehavior Class Reference . . . . .	170
5.50.1	Detailed Description . . . . .	171
5.50.2	Member Function Documentation . . . . .	171
5.50.2.1	endOfEpochAction . . . . .	171
5.50.2.2	singlePatternBackwardAction . . . . .	171
5.50.3	Member Data Documentation . . . . .	171
5.50.3.1	d_neuron . . . . .	171
5.51	PredictBehavior Class Reference . . . . .	172
5.51.1	Detailed Description . . . . .	173
5.51.2	Constructor & Destructor Documentation . . . . .	173
5.51.2.1	PredictBehavior . . . . .	173
5.51.3	Member Function Documentation . . . . .	173
5.51.3.1	getConlterator . . . . .	173
5.51.3.2	setInducedLocalField . . . . .	174
5.51.3.3	setOutput . . . . .	174
5.51.3.4	setOutputDerivative . . . . .	175
5.51.3.5	show . . . . .	175
5.51.3.6	singlePatternForwardAction . . . . .	175
5.51.3.7	useActivationFunctionf0 . . . . .	175
5.51.3.8	useActivationFunctionf1 . . . . .	176
5.51.4	Member Data Documentation . . . . .	176
5.51.4.1	d_neuron . . . . .	176
5.52	RadialBasis Class Reference . . . . .	177

5.52.1 Detailed Description . . . . .	178
5.52.2 Constructor & Destructor Documentation . . . . .	178
5.52.2.1 RadialBasis . . . . .	178
5.52.3 Member Function Documentation . . . . .	178
5.52.3.1 f0 . . . . .	179
5.52.3.2 f1 . . . . .	179
5.53 RadialBasisFactory Class Reference . . . . .	179
5.53.1 Detailed Description . . . . .	182
5.53.2 Constructor & Destructor Documentation . . . . .	182
5.53.2.1 RadialBasisFactory . . . . .	182
5.53.3 Member Function Documentation . . . . .	182
5.53.3.1 makeActivationFunction . . . . .	182
5.54 RBFbehavior Class Reference . . . . .	182
5.54.1 Detailed Description . . . . .	185
5.54.2 Constructor & Destructor Documentation . . . . .	185
5.54.2.1 RBFbehavior . . . . .	185
5.54.3 Member Function Documentation . . . . .	185
5.54.3.1 show . . . . .	185
5.54.3.2 singlePatternForwardAction . . . . .	185
5.54.4 Member Data Documentation . . . . .	185
5.54.4.1 d_altitude . . . . .	185
5.54.4.2 d_width . . . . .	185
5.55 RBFfactory Class Reference . . . . .	185
5.55.1 Detailed Description . . . . .	188
5.55.2 Member Function Documentation . . . . .	188
5.55.2.1 makeActivationFunction . . . . .	188
5.55.2.2 makeCon . . . . .	188
5.55.2.3 makeConContainer . . . . .	188
5.55.2.4 makeLayer . . . . .	188
5.55.2.5 makeLayerContainer . . . . .	188
5.55.2.6 makeNeuralCreator . . . . .	188
5.55.2.7 makeNeuralNetwork . . . . .	189
5.55.2.8 makeNeuron . . . . .	189
5.55.2.9 makeNeuron . . . . .	189



5.55.2.10	makePredictBehavior	189
5.56	Reciprocal Class Reference	189
5.56.1	Detailed Description	191
5.56.2	Constructor & Destructor Documentation	191
5.56.2.1	Reciprocal	191
5.56.3	Member Function Documentation	191
5.56.3.1	f0	192
5.56.3.2	f1	192
5.57	ReciprocalFactory Class Reference	192
5.57.1	Detailed Description	195
5.57.2	Constructor & Destructor Documentation	195
5.57.2.1	ReciprocalFactory	195
5.57.3	Member Function Documentation	195
5.57.3.1	makeActivationFunction	195
5.58	SimpleContainer< T > Class Template Reference	195
5.58.1	Detailed Description	198
5.58.2	Constructor & Destructor Documentation	198
5.58.2.1	SimpleContainer	198
5.58.2.2	~SimpleContainer	198
5.58.3	Member Function Documentation	198
5.58.3.1	at	198
5.58.3.2	clear	199
5.58.3.3	createIterator	199
5.58.3.4	createReverseIterator	199
5.58.3.5	empty	199
5.58.3.6	push_back	199
5.58.3.7	reserve	199
5.58.3.8	show	199
5.58.3.9	size	199
5.58.3.10	validate	199
5.58.4	Friends And Related Function Documentation	200
5.58.4.1	SimpleContainerIterator< T >	200
5.58.4.2	SimpleContainerReverseIterator< T >	200
5.58.5	Member Data Documentation	200

5.58.5.1	d_collection	200
5.59	SimpleContainerIterator< T > Class Template Reference	200
5.59.1	Detailed Description	203
5.59.2	Constructor & Destructor Documentation	203
5.59.2.1	SimpleContainerIterator	203
5.59.2.2	~SimpleContainerIterator	203
5.59.3	Member Function Documentation	203
5.59.3.1	currentItem	203
5.59.3.2	first	203
5.59.3.3	isDone	203
5.59.3.4	next	204
5.59.4	Friends And Related Function Documentation	204
5.59.4.1	SimpleContainer< T >	204
5.59.5	Member Data Documentation	204
5.59.5.1	d_container	204
5.59.5.2	d_current	204
5.60	SimpleContainerReverselIterator< T > Class Template Reference	204
5.60.1	Detailed Description	207
5.60.2	Constructor & Destructor Documentation	207
5.60.2.1	SimpleContainerReverselIterator	207
5.60.2.2	~SimpleContainerReverselIterator	207
5.60.3	Member Function Documentation	207
5.60.3.1	currentItem	207
5.60.3.2	first	207
5.60.3.3	isDone	207
5.60.3.4	next	208
5.60.4	Friends And Related Function Documentation	208
5.60.4.1	SimpleContainer< T >	208
5.60.5	Member Data Documentation	208
5.60.5.1	d_container	208
5.60.5.2	d_current	208
5.61	SimpleNetwork Class Reference	208
5.61.1	Detailed Description	211
5.61.2	Constructor & Destructor Documentation	211

5.61.2.1	SimpleNetwork	211
5.61.3	Member Function Documentation	211
5.61.3.1	inputSize	211
5.61.3.2	outputSize	212
5.61.3.3	readOutput	212
5.61.3.4	show	213
5.61.3.5	singlePatternBackwardAction	213
5.61.3.6	singlePatternForwardAction	214
5.61.3.7	train	215
5.61.3.8	validate	215
5.61.3.9	writelnInput	215
5.62	SimpleNeuralCreator Class Reference	216
5.62.1	Detailed Description	217
5.62.2	Constructor & Destructor Documentation	217
5.62.2.1	SimpleNeuralCreator	217
5.62.3	Member Function Documentation	218
5.62.3.1	createFeedForwardNetwork	218
5.63	SimpleNeuron Class Reference	219
5.63.1	Detailed Description	223
5.63.2	Constructor & Destructor Documentation	223
5.63.2.1	SimpleNeuron	223
5.63.3	Member Function Documentation	224
5.63.3.1	addCon	224
5.63.3.2	getConIterator	224
5.63.3.3	getId	224
5.63.3.4	getInducedLocalField	225
5.63.3.5	getOutput	225
5.63.3.6	setActivationFunction	225
5.63.3.7	setId	226
5.63.3.8	setInducedLocalField	226
5.63.3.9	setOutput	226
5.63.3.10	setOutputDerivative	226
5.63.3.11	setPredictBehavior	227
5.63.3.12	show	227

5.63.3.13 singlePatternBackwardAction . . . . .	228
5.63.3.14 singlePatternForwardAction . . . . .	228
5.63.3.15 useActivationFunction0 . . . . .	228
5.63.3.16 useActivationFunction1 . . . . .	229
5.63.3.17 validate . . . . .	229
5.64 Sine Class Reference . . . . .	230
5.64.1 Detailed Description . . . . .	231
5.64.2 Constructor & Destructor Documentation . . . . .	231
5.64.2.1 Sine . . . . .	231
5.64.3 Member Function Documentation . . . . .	231
5.64.3.1 f0 . . . . .	232
5.64.3.2 f1 . . . . .	232
5.65 SineFactory Class Reference . . . . .	232
5.65.1 Detailed Description . . . . .	235
5.65.2 Constructor & Destructor Documentation . . . . .	235
5.65.2.1 SineFactory . . . . .	235
5.65.3 Member Function Documentation . . . . .	235
5.65.3.1 makeActivationFunction . . . . .	235
5.66 Square Class Reference . . . . .	235
5.66.1 Detailed Description . . . . .	237
5.66.2 Constructor & Destructor Documentation . . . . .	237
5.66.2.1 Square . . . . .	237
5.66.3 Member Function Documentation . . . . .	237
5.66.3.1 f0 . . . . .	238
5.66.3.2 f1 . . . . .	238
5.67 SquareFactory Class Reference . . . . .	238
5.67.1 Detailed Description . . . . .	241
5.67.2 Constructor & Destructor Documentation . . . . .	241
5.67.2.1 SquareFactory . . . . .	241
5.67.3 Member Function Documentation . . . . .	241
5.67.3.1 makeActivationFunction . . . . .	241
5.68 Tanh Class Reference . . . . .	241
5.68.1 Detailed Description . . . . .	243
5.68.2 Constructor & Destructor Documentation . . . . .	243

5.68.2.1	Tanh	243
5.68.3	Member Function Documentation	244
5.68.3.1	f0	244
5.68.3.2	f1	244
5.69	TanhFactory Class Reference	245
5.69.1	Detailed Description	248
5.69.2	Constructor & Destructor Documentation	248
5.69.2.1	TanhFactory	248
5.69.3	Member Function Documentation	248
5.69.3.1	makeActivationFunction	248
5.70	Tao Class Reference	248
5.70.1	Detailed Description	250
5.70.2	Member Function Documentation	250
5.70.2.1	f0	250
5.70.2.2	f1	251
5.70.3	Member Data Documentation	251
5.70.3.1	d_STao	251
5.71	Threshold Class Reference	251
5.71.1	Detailed Description	252
5.71.2	Constructor & Destructor Documentation	252
5.71.2.1	Threshold	252
5.71.3	Member Function Documentation	252
5.71.3.1	f0	253
5.71.3.2	f1	253
5.72	ThresholdFactory Class Reference	253
5.72.1	Detailed Description	256
5.72.2	Constructor & Destructor Documentation	256
5.72.2.1	ThresholdFactory	256
5.72.3	Member Function Documentation	256
5.72.3.1	makeActivationFunction	256
<b>6</b>	<b>File Documentation</b>	<b>257</b>
6.1	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/ActivationFunction.cpp File Reference	257

6.2	<a href="#">/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/ADAPTgdNetworkTrainBehavior.cpp File Reference</a>	258
6.3	<a href="#">/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/AMORE.h File Reference</a>	258
6.3.1	Define Documentation	260
6.3.1.1	size_type	260
6.3.2	Typedef Documentation	260
6.3.2.1	ActivationFunctionPtr	260
6.3.2.2	ActivationFunctionRef	260
6.3.2.3	ConContainerPtr	260
6.3.2.4	ConIteratorPtr	260
6.3.2.5	ConPtr	260
6.3.2.6	Handler	261
6.3.2.7	LayerContainerPtr	261
6.3.2.8	LayerPtr	261
6.3.2.9	NetworkTrainBehaviorPtr	261
6.3.2.10	NeuralCreatorPtr	261
6.3.2.11	NeuralFactoryPtr	261
6.3.2.12	NeuralNetworkPtr	261
6.3.2.13	NeuralNetworkWeakPtr	261
6.3.2.14	NeuronIteratorPtr	261
6.3.2.15	NeuronPtr	261
6.3.2.16	NeuronRef	262
6.3.2.17	NeuronTrainBehaviorPtr	262
6.3.2.18	NeuronWeakPtr	262
6.3.2.19	PredictBehaviorPtr	262
6.3.2.20	PredictBehaviorRef	262
6.3.2.21	TrainingBehaviorRef	262
6.4	<a href="#">/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ActivationFunction.h File Reference</a>	262
6.5	<a href="#">/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdHiddenNeuronTrainBehavior.h File Reference</a>	263
6.6	<a href="#">/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdNetworkTrainBehavior.h File Reference</a>	263

6.7	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdNeuronTrainBehavior.h File Reference . . . . .	264
6.8	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdOutputNeuronTrainBehavior.h File Reference . . . . .	264
6.9	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwmHiddenNeuronTrainBehavior.h File Reference . . . . .	265
6.10	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwmNetworkTrainBehavior.h File Reference . . . . .	266
6.11	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwmNeuronTrainBehavior.h File Reference . . . . .	266
6.12	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwmOutputNeuronTrainBehavior.h File Reference . . . . .	267
6.13	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptNetworkTrainBehavior.h File Reference . . . . .	268
6.14	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptNeuronTrainBehavior.h File Reference . . . . .	268
6.15	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ArcTan.h File Reference . . . . .	269
6.16	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ArcTanFactory.h File Reference . . . . .	270
6.17	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdHiddenNeuronTrainBehavior.h File Reference . . . . .	270
6.18	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdNetworkTrainBehavior.h File Reference . . . . .	271
6.19	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdNeuronTrainBehavior.h File Reference . . . . .	272
6.20	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdOutputNeuronTrainBehavior.h File Reference . . . . .	272
6.21	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwmHiddenNeuronTrainBehavior.h File Reference . . . . .	273

6.22	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwmNetworkTrainBehavior.h File Reference . . . . .	274
6.23	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwmNeuronTrainBehavior.h File Reference . . . . .	274
6.24	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwmOutputNeuronTrainBehavior.h File Reference . . . . .	275
6.25	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BatchNetworkTrainBehavior.h File Reference . . . . .	276
6.26	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BatchNeuronTrainBehavior.h File Reference . . . . .	276
6.27	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Connection.h File Reference . . . . .	277
6.28	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Container.h File Reference . . . . .	277
6.29	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Cosine.h File Reference . . . . .	278
6.30	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/CosineFactory.h File Reference . . . . .	279
6.31	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/CostFunction.h File Reference . . . . .	279
6.32	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Elliot.h File Reference . . . . .	280
6.33	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ElliotFactory.h File Reference . . . . .	280
6.34	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Exponential.h File Reference . . . . .	281
6.35	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ExponentialFactory.h File Reference . . . . .	281
6.36	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Gauss.h File Reference . . . . .	282
6.37	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/GaussFactory.h File Reference . . . . .	282
6.38	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Identity.h File Reference . . . . .	283
6.39	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/IdentityFactory.h File Reference . . . . .	284



6.40	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Iterator.h File Reference . . . . .	284
6.41	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/LMLS.h File Reference . . . . .	285
6.42	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/LMS.h File Reference . . . . .	286
6.43	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Logistic.h File Reference . . . . .	286
6.44	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticFactory.h File Reference . . . . .	287
6.45	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPbehavior.h File Reference . . . . .	287
6.46	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPfactory.h File Reference . . . . .	288
6.47	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkRinterface.h File Reference . . . . .	289
6.48	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkTrainBehavior.h File Reference . . . . .	289
6.49	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralCreator.h File Reference . . . . .	290
6.50	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralFactory.h File Reference . . . . .	290
6.51	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralNetwork.h File Reference . . . . .	291
6.52	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Neuron.h File Reference . . . . .	291
6.53	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuronTrainBehavior.h File Reference . . . . .	291
6.54	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/PredictBehavior.h File Reference . . . . .	292
6.55	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasis.h File Reference . . . . .	292
6.56	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasisFactory.h File Reference . . . . .	293
6.57	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFbehavior.h File Reference . . . . .	293
6.58	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFfactory.h File Reference . . . . .	294

6.59	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Reciprocal.h File Reference . . . . .	294
6.60	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ReciprocalFactory.h File Reference . . . . .	295
6.61	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainer.h File Reference . . . . .	296
6.62	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainerIterator.h File Reference . . . . .	296
6.63	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainerReverseliterator.h File Reference . . . . .	297
6.64	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNetwork.h File Reference . . . . .	298
6.65	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuralCreator.h File Reference . . . . .	299
6.66	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuron.h File Reference . . . . .	300
6.67	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Sine.h File Reference . . . . .	300
6.68	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SineFactory.h File Reference . . . . .	301
6.69	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Square.h File Reference . . . . .	302
6.70	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SquareFactory.h File Reference . . . . .	302
6.71	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Tanh.h File Reference . . . . .	303
6.72	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/TanhFactory.h File Reference . . . . .	303
6.73	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Tao.h File Reference . . . . .	304
6.74	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Threshold.h File Reference . . . . .	305
6.75	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ThresholdFactory.h File Reference . . . . .	305
6.76	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Connection.cpp File Reference . . . . .	306
6.77	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Identity.cpp File Reference . . . . .	307

6.78	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/IdentityFactory.cpp File Reference . . . . .	308
6.79	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPbehavior.cpp File Reference . . . . .	308
6.80	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPfactory.cpp File Reference . . . . .	309
6.81	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NetworkRinterface.cpp File Reference . . . . .	310
6.82	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NeuralNetwork.cpp File Reference . . . . .	311
6.83	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Neuron.cpp File Reference . . . . .	312
6.84	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/package.h File Reference . . . . .	312
6.84.1	Define Documentation . . . . .	314
6.84.1.1	size_type . . . . .	314
6.84.2	Typedef Documentation . . . . .	314
6.84.2.1	ActivationFunctionPtr . . . . .	314
6.84.2.2	ActivationFunctionRef . . . . .	314
6.84.2.3	ConContainerPtr . . . . .	314
6.84.2.4	ConIteratorPtr . . . . .	314
6.84.2.5	ConPtr . . . . .	314
6.84.2.6	Handler . . . . .	314
6.84.2.7	LayerContainerPtr . . . . .	314
6.84.2.8	LayerPtr . . . . .	314
6.84.2.9	NetworkTrainBehaviorPtr . . . . .	315
6.84.2.10	NeuralCreatorPtr . . . . .	315
6.84.2.11	NeuralFactoryPtr . . . . .	315
6.84.2.12	NeuralNetworkPtr . . . . .	315
6.84.2.13	NeuralNetworkWeakPtr . . . . .	315
6.84.2.14	NeuronIteratorPtr . . . . .	315
6.84.2.15	NeuronPtr . . . . .	315
6.84.2.16	NeuronRef . . . . .	315
6.84.2.17	NeuronTrainBehaviorPtr . . . . .	315
6.84.2.18	NeuronWeakPtr . . . . .	315
6.84.2.19	PredictBehaviorPtr . . . . .	316

6.84.2.20 PredictBehaviorRef . . . . .	316
6.85 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/PredictBehavior.cpp File Reference . . . . .	316
6.86 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/RcppModules.cpp File Reference . . . . .	316
6.86.1 Function Documentation . . . . .	317
6.86.1.1 RCPP_MODULE . . . . .	317
6.87 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNetwork.cpp File Reference . . . . .	318
6.88 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuralCreator.cpp File Reference . . . . .	319
6.89 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuron.cpp File Reference . . . . .	320
6.90 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Tanh.cpp File Reference . . . . .	320
6.91 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/TanhFactory.cpp File Reference . . . . .	321

# 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 . . . . .	15
ArcTan . . . . .	47
Cosine . . . . .	92
Elliot . . . . .	100
Exponential . . . . .	105
Gauss . . . . .	111
Identity . . . . .	117
Logistic . . . . .	130
RadialBasis . . . . .	177
Reciprocal . . . . .	189
Sine . . . . .	230
Square . . . . .	235
Tanh . . . . .	241
Threshold . . . . .	251
Con . . . . .	83
Container< T > . . . . .	89
SimpleContainer< T > . . . . .	195
CostFunction . . . . .	98
LMLS . . . . .	125
LMS . . . . .	128
Tao . . . . .	248
Iterator< T > . . . . .	123
SimpleContainerIterator< T > . . . . .	200
SimpleContainerReverseIterator< T > . . . . .	204
NetworkRinterface . . . . .	147
NetworkTrainBehavior . . . . .	153
AdaptNetworkTrainBehavior . . . . .	42
ADAPTgdNetworkTrainBehavior . . . . .	20

ADAPTgdwmNetworkTrainBehavior . . . . .	32
BatchNetworkTrainBehavior . . . . .	78
BATCHgdNetworkTrainBehavior . . . . .	55
BATCHgdwmNetworkTrainBehavior . . . . .	68
NeuralCreator . . . . .	154
SimpleNeuralCreator . . . . .	216
NeuralFactory . . . . .	156
MLPfactory . . . . .	140
ArcTanFactory . . . . .	49
CosineFactory . . . . .	95
ElliotFactory . . . . .	102
ExponentialFactory . . . . .	108
GaussFactory . . . . .	114
IdentityFactory . . . . .	120
LogisticFactory . . . . .	132
ReciprocalFactory . . . . .	192
SineFactory . . . . .	232
SquareFactory . . . . .	238
TanhFactory . . . . .	245
ThresholdFactory . . . . .	253
RBFfactory . . . . .	185
RadialBasisFactory . . . . .	179
NeuralNetwork . . . . .	159
SimpleNetwork . . . . .	208
Neuron . . . . .	164
SimpleNeuron . . . . .	219
NeuronTrainBehavior . . . . .	170
AdaptNeuronTrainBehavior . . . . .	45
ADAPTgdNeuronTrainBehavior . . . . .	23
ADAPTgdHiddenNeuronTrainBehavior . . . . .	17
ADAPTgdOutputNeuronTrainBehavior . . . . .	26
ADAPTgdwmNeuronTrainBehavior . . . . .	35
ADAPTgdwmHiddenNeuronTrainBehavior . . . . .	29
ADAPTgdwmOutputNeuronTrainBehavior . . . . .	39
BatchNeuronTrainBehavior . . . . .	81
BATCHgdNeuronTrainBehavior . . . . .	58
BATCHgdHiddenNeuronTrainBehavior . . . . .	52
BATCHgdOutputNeuronTrainBehavior . . . . .	62
BATCHgdwmNeuronTrainBehavior . . . . .	71
BATCHgdwmHiddenNeuronTrainBehavior . . . . .	65
BATCHgdwmOutputNeuronTrainBehavior . . . . .	75
PredictBehavior . . . . .	172
MLPbehavior . . . . .	135
RBFbehavior . . . . .	182



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">ActivationFunction</a> (Class <a href="#">ActivationFunction</a> - ) . . . . .	15
<a href="#">ADAPTgdHiddenNeuronTrainBehavior</a> (Class <a href="#">ADAPTgdHiddenNeuronTrainBehavior</a> - ) . . . . .	17
<a href="#">ADAPTgdNetworkTrainBehavior</a> (Class <a href="#">ADAPTgdNetworkTrainBehavior</a> - ) . . . . .	20
<a href="#">ADAPTgdNeuronTrainBehavior</a> (Class <a href="#">ADAPTgdNeuronTrainBehavior</a> - ) . . . . .	23
<a href="#">ADAPTgdOutputNeuronTrainBehavior</a> (Class <a href="#">ADAPTgdOutputNeuronTrainBehavior</a> - ) . . . . .	26
<a href="#">ADAPTgdwmHiddenNeuronTrainBehavior</a> (Class <a href="#">ADAPTgdwmHiddenNeuronTrainBehavior</a> - ) . . . . .	29
<a href="#">ADAPTgdwmNetworkTrainBehavior</a> (Class <a href="#">ADAPTgdwmNetworkTrainBehavior</a> - ) . . . . .	32
<a href="#">ADAPTgdwmNeuronTrainBehavior</a> (Class <a href="#">ADAPTgdwmNeuronTrainBehavior</a> - ) . . . . .	35
<a href="#">ADAPTgdwmOutputNeuronTrainBehavior</a> (Class <a href="#">ADAPTgdwmOutputNeuronTrainBehavior</a> - ) . . . . .	39
<a href="#">AdaptNetworkTrainBehavior</a> (Class <a href="#">AdaptNetworkTrainBehavior</a> - ) . . . . .	42
<a href="#">AdaptNeuronTrainBehavior</a> (Class <a href="#">AdaptNeuronTrainBehavior</a> - ) . . . . .	45
<a href="#">ArcTan</a> (Class <a href="#">ArcTan</a> - ) . . . . .	47
<a href="#">ArcTanFactory</a> (Class <a href="#">ArcTanFactory</a> - ) . . . . .	49
<a href="#">BATCHgdHiddenNeuronTrainBehavior</a> (Class <a href="#">BATCHgdHiddenNeuronTrainBehavior</a> - ) . . . . .	52
<a href="#">BATCHgdNetworkTrainBehavior</a> (Class <a href="#">BATCHgdNetworkTrainBehavior</a> - ) . . . . .	55
<a href="#">BATCHgdNeuronTrainBehavior</a> (Class <a href="#">BATCHgdNeuronTrainBehavior</a> - ) . . . . .	58
<a href="#">BATCHgdOutputNeuronTrainBehavior</a> (Class <a href="#">BATCHgdOutputNeuronTrainBehavior</a> - ) . . . . .	62
<a href="#">BATCHgdwmHiddenNeuronTrainBehavior</a> (Class <a href="#">BATCHgdwmHiddenNeuronTrainBehavior</a> - ) . . . . .	65
<a href="#">BATCHgdwmNetworkTrainBehavior</a> (Class <a href="#">BATCHgdwmNetworkTrainBehavior</a> - ) . . . . .	68

BATCHgdwmNeuronTrainBehavior (Class BATCHgdwmNeuronTrainBehavior - ) . . . . .	71
BATCHgdwmOutputNeuronTrainBehavior (Class BATCHgdwmOutputNeuronTrainBehavior - ) . . . . .	75
BatchNetworkTrainBehavior (Class BatchNetworkTrainBehavior - ) . . . . .	78
BatchNeuronTrainBehavior (Class BatchNeuronTrainBehavior - ) . . . . .	81
Con (Class Con - ) . . . . .	83
Container< T > (Class Container - ) . . . . .	89
Cosine (Class Cosine - ) . . . . .	92
CosineFactory (Class CosineFactory - ) . . . . .	95
CostFunction (Class CostFunction - ) . . . . .	98
Elliot (Class Elliot - ) . . . . .	100
ElliotFactory (Class ElliotFactory - ) . . . . .	102
Exponential (Class Exponential - ) . . . . .	105
ExponentialFactory (Class ExponentialFactory - ) . . . . .	108
Gauss (Class Gauss - ) . . . . .	111
GaussFactory (Class GaussFactory - ) . . . . .	114
Identity (Class Identity - ) . . . . .	117
IdentityFactory (Class IdentityFactory - ) . . . . .	120
Iterator< T > (Class Iterator - ) . . . . .	123
LMLS (Class LMLS - ) . . . . .	125
LMS (Class LMS - ) . . . . .	128
Logistic (Class Logistic - ) . . . . .	130
LogisticFactory (Class LogisticFactory - ) . . . . .	132
MLPbehavior (Class MLPbehavior - ) . . . . .	135
MLPfactory (Class MLPfactory - ) . . . . .	140
NetworkRinterface (Class NetworkRinterface - ) . . . . .	147
NetworkTrainBehavior (Class NetworkTrainBehavior - ) . . . . .	153
NeuralCreator (Class NeuralCreator - ) . . . . .	154
NeuralFactory (Class NeuralFactory - ) . . . . .	156
NeuralNetwork (Class NeuralNetwork - ) . . . . .	159
Neuron (Class Neuron - ) . . . . .	164
NeuronTrainBehavior (Class NeuronTrainBehavior - ) . . . . .	170
PredictBehavior (Class PredictBehavior - ) . . . . .	172
RadialBasis (Class RadialBasis - ) . . . . .	177
RadialBasisFactory (Class RadialBasisFactory - ) . . . . .	179
RBFbehavior (Class RBFbehavior - ) . . . . .	182
RBFfactory (Class RBFfactory - ) . . . . .	185
Reciprocal (Class Reciprocal - ) . . . . .	189
ReciprocalFactory (Class ReciprocalFactory - ) . . . . .	192
SimpleContainer< T > (Class SimpleContainer - ) . . . . .	195
SimpleContainerIterator< T > (Class SimpleContainerIterator - ) . . . . .	200
SimpleContainerReverselIterator< T > (Class SimpleContainerReverselIterator - ) . . . . .	204
SimpleNetwork (Class SimpleNetwork - ) . . . . .	208
SimpleNeuralCreator (Class SimpleNeuralCreator - ) . . . . .	216
SimpleNeuron (Class SimpleNeuron - ) . . . . .	219
Sine (Class Sine - ) . . . . .	230
SineFactory (Class SineFactory - ) . . . . .	232
Square (Class Square - ) . . . . .	235

---

<a href="#">SquareFactory</a> (Class <a href="#">SquareFactory</a> - ) . . . . .	238
<a href="#">Tanh</a> (Class <a href="#">Tanh</a> - ) . . . . .	241
<a href="#">TanhFactory</a> (Class <a href="#">TanhFactory</a> - ) . . . . .	245
<a href="#">Tao</a> (Class <a href="#">Tao</a> - ) . . . . .	248
<a href="#">Threshold</a> (Class <a href="#">Threshold</a> - ) . . . . .	251
<a href="#">ThresholdFactory</a> (Class <a href="#">ThresholdFactory</a> - ) . . . . .	253



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[ActivationFunction.cpp](#)  
257

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[ADAPTgdNetworkTrainBeh](#)  
258

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[AMORE.h](#)  
258

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[Connection.cpp](#)  
306

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[Identity.cpp](#)  
307

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[IdentityFactory.cpp](#)  
308

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[MLPbehavior.cpp](#)  
308

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[MLPfactory.cpp](#)  
309

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[NetworkRinterface.cpp](#)  
310

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[NeuralNetwork.cpp](#)  
311

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[Neuron.cpp](#)  
312

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[package.h](#)  
312

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[PredictBehavior.cpp](#)  
316

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[RcppModules.cpp](#)  
316

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNetwork  
318  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeural  
319  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuron  
320  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Tanh.cpp  
320  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/TanhFactory.c  
321  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
262  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
263  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
263  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
264  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
264  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
265  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
266  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
266  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
267  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
268  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
268  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
269  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
270  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
270  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
271  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
272  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
272  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
273  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
274  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
274

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwn	275
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BatchNetwo	276
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BatchNeuron	276
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Connection.	277
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Container.h	277
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Cosine.h	278
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/CosineFactor	279
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/CostFunction	279
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Elliot.h	280
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ElliotFactory	280
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Exponential.	281
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ExponentialI	281
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Gauss.h	282
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/GaussFactor	282
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Identity.h	283
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/IdentityFacto	284
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Iterator.h	284
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/LMLS.h	285
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/LMS.h	286
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Logistic.h	286
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticFacto	287
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPbehavio	287
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPfactory.h	288
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkRint	289
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkTrain	289

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
290  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
290  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
291  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
291  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
291  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
292  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
292  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
293  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
293  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
294  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
294  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
295  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
296  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
296  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
297  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
298  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
299  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
300  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
300  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
301  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
302  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
302  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
303  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
303  
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
304



---

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[Threshold.h](#)  
305

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[ThresholdFa](#)  
305



## Chapter 5

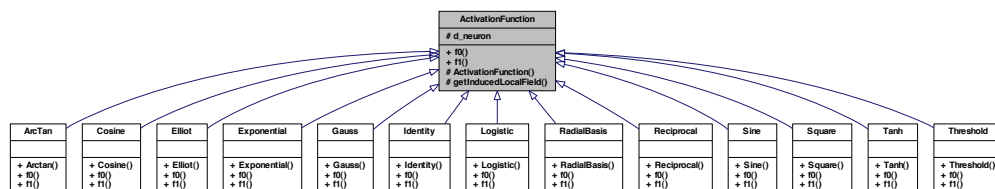
# Class Documentation

### 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 12 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 18 of file ActivationFunction.cpp.

References `d_neuron`.

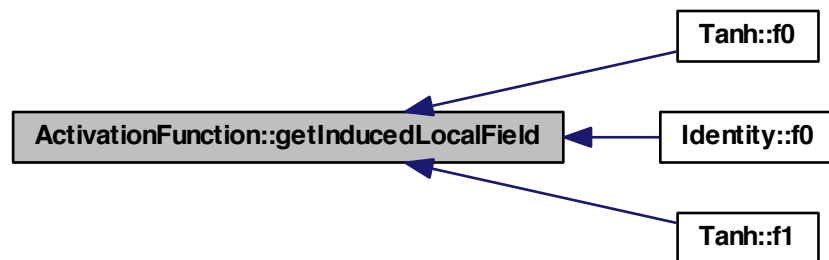
Referenced by `Tanh::f0()`, `Identity::f0()`, and `Tanh::f1()`.

```

{
    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:

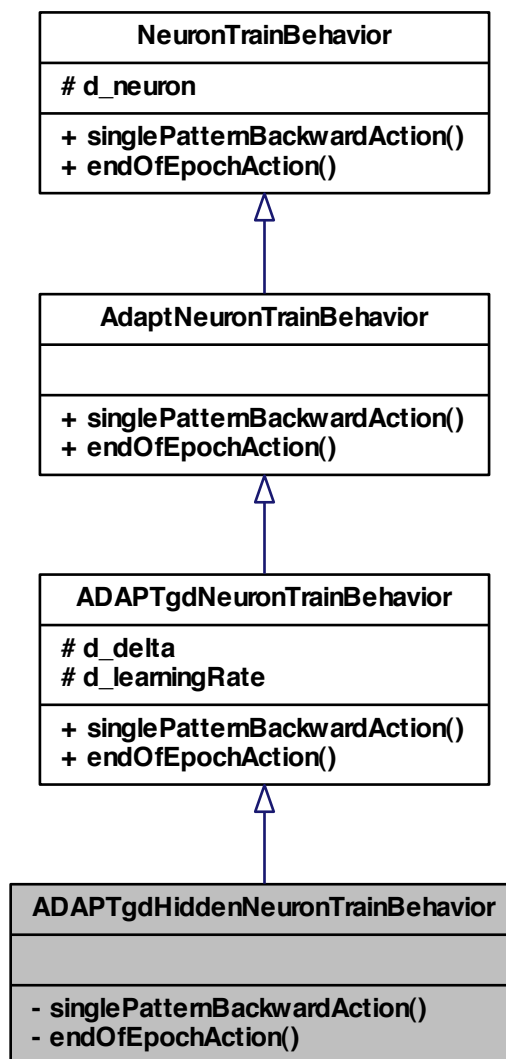
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ActivationFunction.h`
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/ActivationFunction.cpp`

## 5.2 ADAPTgdHiddenNeuronTrainBehavior Class Reference

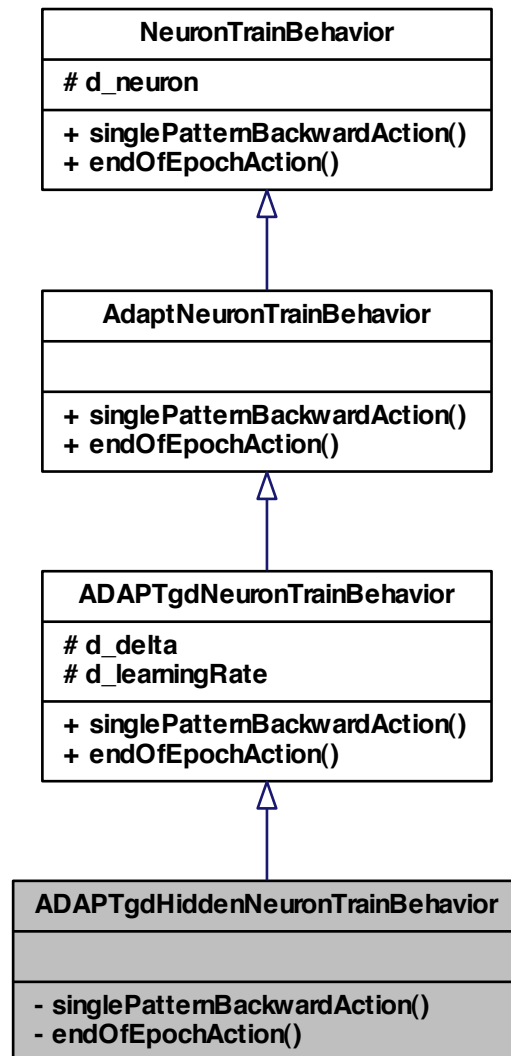
class [ADAPTgdHiddenNeuronTrainBehavior](#) -

```
#include <ADAPTgdHiddenNeuronTrainBehavior.h>
```

Inheritance diagram for ADAPTgdHiddenNeuronTrainBehavior:



Collaboration diagram for ADAPTgdHiddenNeuronTrainBehavior:



### Private Member Functions

- void [singlePatternBackwardAction](#) ()
- void [endOfEpochAction](#) ()

### 5.2.1 Detailed Description

class [ADAPTgdHiddenNeuronTrainBehavior](#) -

Definition at line 5 of file ADAPTgdHiddenNeuronTrainBehavior.h.

### 5.2.2 Member Function Documentation

**5.2.2.1** void ADAPTgdHiddenNeuronTrainBehavior::endOfEpochAction ( ) [private, virtual]

Implements [ADAPTgdNeuronTrainBehavior](#).

**5.2.2.2** void ADAPTgdHiddenNeuronTrainBehavior::singlePatternBackwardAction ( ) [private, virtual]

Implements [ADAPTgdNeuronTrainBehavior](#).

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

- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

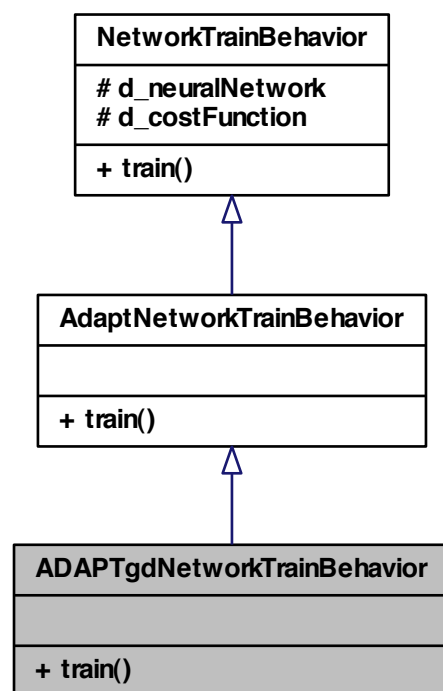
## 5.3 ADAPTgdNetworkTrainBehavior Class Reference

class [ADAPTgdNetworkTrainBehavior](#) -

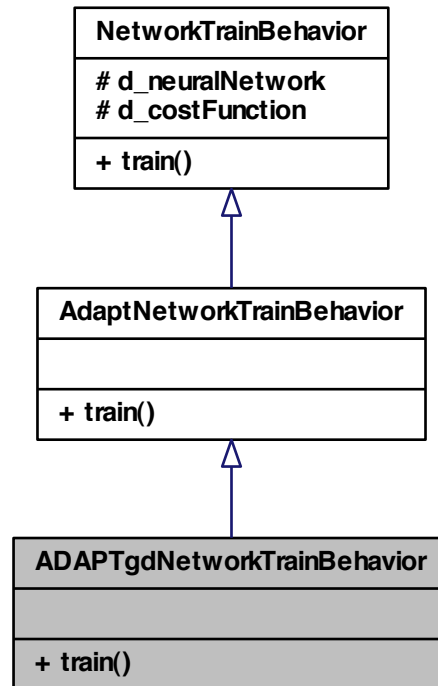
#include <ADAPTgdNetworkTrainBehavior.h>



Inheritance diagram for ADAPTgdNetworkTrainBehavior:



Collaboration diagram for ADAPTgdNetworkTrainBehavior:



## Public Member Functions

- `Rcpp::List` [train](#) (`Rcpp::List` parameterList)

### 5.3.1 Detailed Description

class [ADAPTgdNetworkTrainBehavior](#) -

Definition at line 5 of file `ADAPTgdNetworkTrainBehavior.h`.

### 5.3.2 Member Function Documentation

5.3.2.1 `ADAPTgdNetworkTrainBehavior::train ( Rcpp::List parameterList )` `[virtual]`

Implements [AdaptNetworkTrainBehavior](#).

Definition at line 8 of file ADAPTgdNetworkTrainBehavior.cpp.

References NetworkTrainBehavior::d\_neuralNetwork.

```
{
    int numberOfEpochs = as<int> (parameterList["numberOfEpochs"]);
    Rcpp::NumericMatrix inputMatrix = as<Rcpp::NumericMatrix> (
        parameterList["inputMatrix"]);
    Rcpp::NumericMatrix targetMatrix = as<Rcpp::NumericMatrix> (
        parameterList["targetMatrix"]);
    int numberOfEpochs = as<int> (parameterList["numberOfEpochs"]);
    int showStep = as<int> (parameterList["showStep"]);

    // Rcpp::NumericMatrix outputMatrix(outputSize(), numericMatrix.ncol());
    std::vector<double>::iterator inputIterator(inputMatrix.begin());
    std::vector<double>::iterator targetIterator(targetMatrix.begin());

    int maxShows = (numberOfEpochs > showStep) ? numberOfEpochs / showStep : 1;
    for (int idShow = 0; idShow < maxShows; ++idShow)
    {
        for (int step = 0; step < showStep; ++step)
        {
            for (int idRow = 0; idRow < inputMatrix.ncol(); idRow++)
            {
                d_neuralNetwork->writeInput(inputIterator);
                d_neuralNetwork->singlePatternForwardAction();

                d_neuralNetwork->singlePatternBackwardAction();
            }
        }
    }
}
```

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

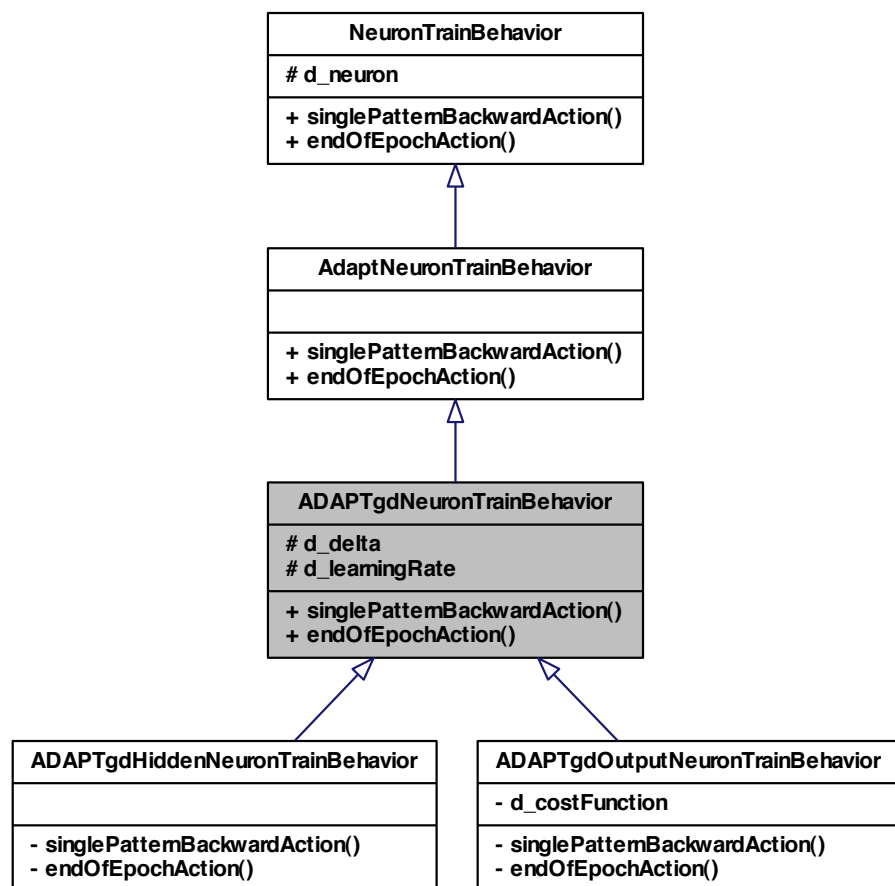
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdNeuronTrainBehavior.h
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/ADAPTgdNetworkTrainBehavior.cpp

## 5.4 ADAPTgdNeuronTrainBehavior Class Reference

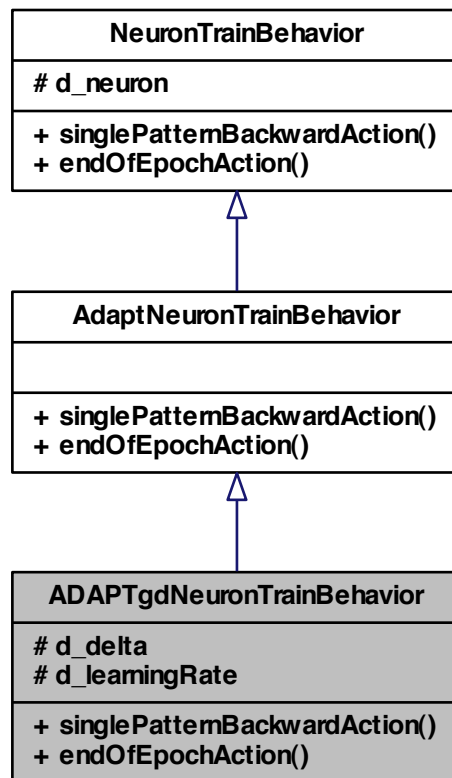
class [ADAPTgdNeuronTrainBehavior](#) -

```
#include <ADAPTgdNeuronTrainBehavior.h>
```

Inheritance diagram for ADAPTgdNeuronTrainBehavior:



Collaboration diagram for ADAPTgdNeuronTrainBehavior:



#### Public Member Functions

- virtual void `singlePatternBackwardAction` ()=0
- virtual void `endOfEpochAction` ()=0

#### Protected Attributes

- double `d_delta`
- double `d_learningRate`

### 5.4.1 Detailed Description

class [ADAPTgdNeuronTrainBehavior](#) -

Definition at line 5 of file ADAPTgdNeuronTrainBehavior.h.

### 5.4.2 Member Function Documentation

5.4.2.1 `virtual void ADAPTgdNeuronTrainBehavior::endOfEpochAction ( ) [pure virtual]`

Implements [AdaptNeuronTrainBehavior](#).

Implemented in [ADAPTgdHiddenNeuronTrainBehavior](#), and [ADAPTgdOutputNeuronTrainBehavior](#).

5.4.2.2 `virtual void ADAPTgdNeuronTrainBehavior::singlePatternBackwardAction ( ) [pure virtual]`

Implements [AdaptNeuronTrainBehavior](#).

Implemented in [ADAPTgdHiddenNeuronTrainBehavior](#), and [ADAPTgdOutputNeuronTrainBehavior](#).

### 5.4.3 Member Data Documentation

5.4.3.1 `double ADAPTgdNeuronTrainBehavior::d_delta [protected]`

Definition at line 8 of file ADAPTgdNeuronTrainBehavior.h.

5.4.3.2 `double ADAPTgdNeuronTrainBehavior::d_learningRate [protected]`

Definition at line 9 of file ADAPTgdNeuronTrainBehavior.h.

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

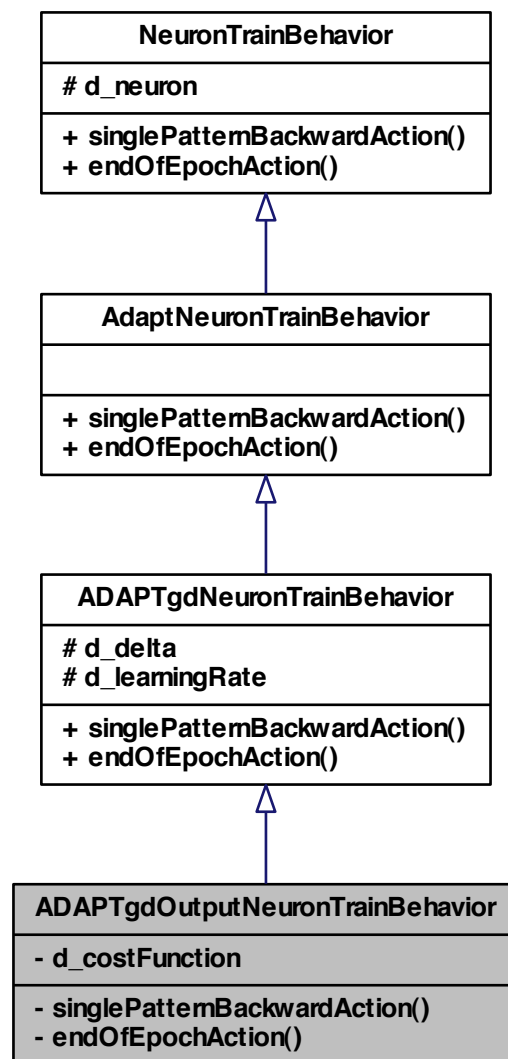
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.5 ADAPTgdOutputNeuronTrainBehavior Class Reference

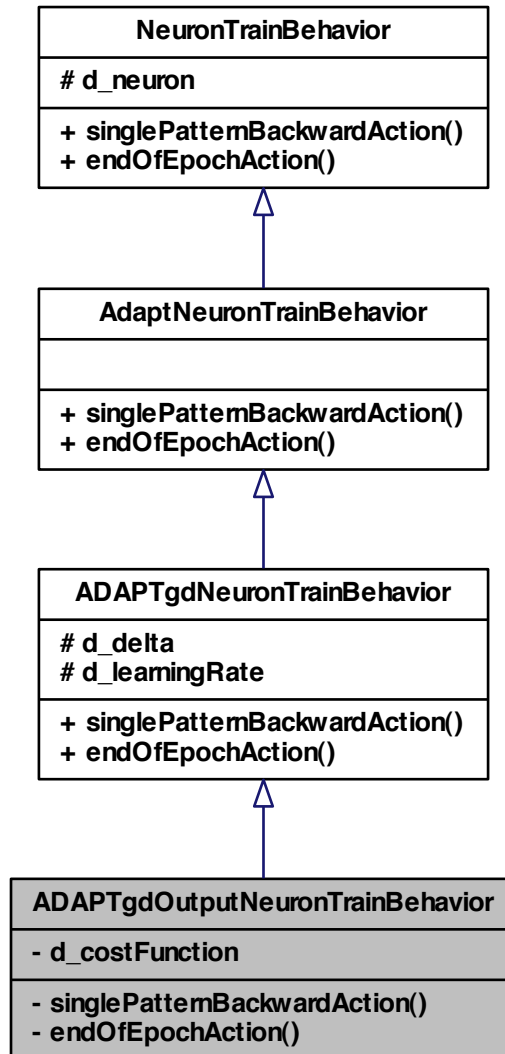
class [ADAPTgdOutputNeuronTrainBehavior](#) -

```
#include <ADAPTgdOutputNeuronTrainBehavior.h>
```

Inheritance diagram for ADAPTgdOutputNeuronTrainBehavior:



Collaboration diagram for ADAPTgdOutputNeuronTrainBehavior:



### Private Member Functions

- void [singlePatternBackwardAction](#) ()
- void [endOfEpochAction](#) ()



### Private Attributes

- CostFunctionWeakPtr [d\\_costFunction](#)

### 5.5.1 Detailed Description

class [ADAPTgdOutputNeuronTrainBehavior](#) -

Definition at line 5 of file [ADAPTgdOutputNeuronTrainBehavior.h](#).

### 5.5.2 Member Function Documentation

**5.5.2.1** void [ADAPTgdOutputNeuronTrainBehavior::endOfEpochAction](#) ( ) [private, virtual]

Implements [ADAPTgdNeuronTrainBehavior](#).

**5.5.2.2** void [ADAPTgdOutputNeuronTrainBehavior::singlePatternBackwardAction](#) ( ) [private, virtual]

Implements [ADAPTgdNeuronTrainBehavior](#).

### 5.5.3 Member Data Documentation

**5.5.3.1** CostFunctionWeakPtr [ADAPTgdOutputNeuronTrainBehavior::d\\_costFunction](#) [private]

Definition at line 8 of file [ADAPTgdOutputNeuronTrainBehavior.h](#).

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

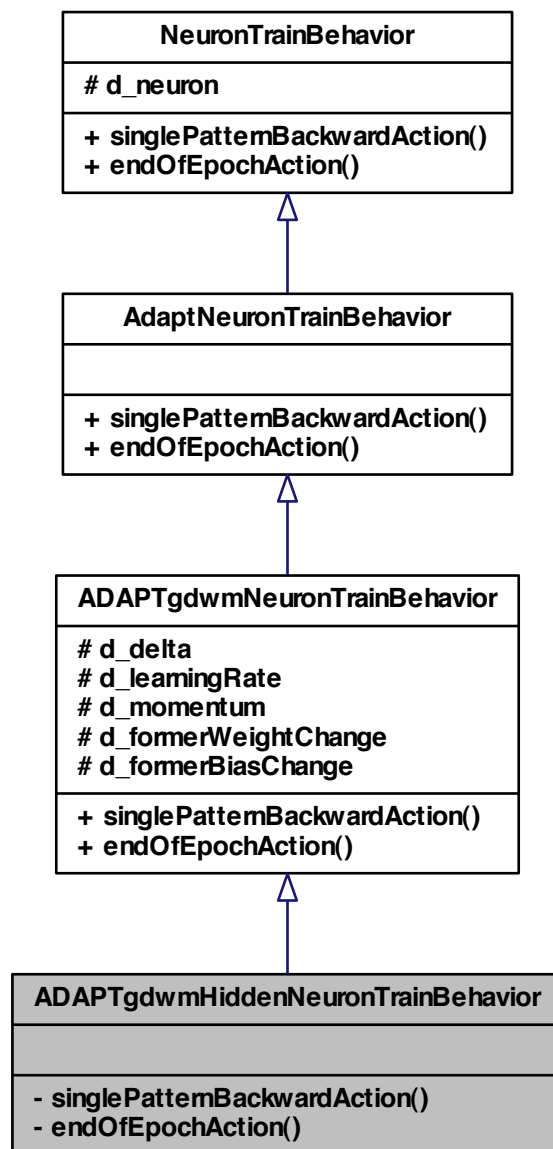
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwmHiddenNeuronTrainBehavior.h](#)

## 5.6 ADAPTgdwmHiddenNeuronTrainBehavior Class Reference

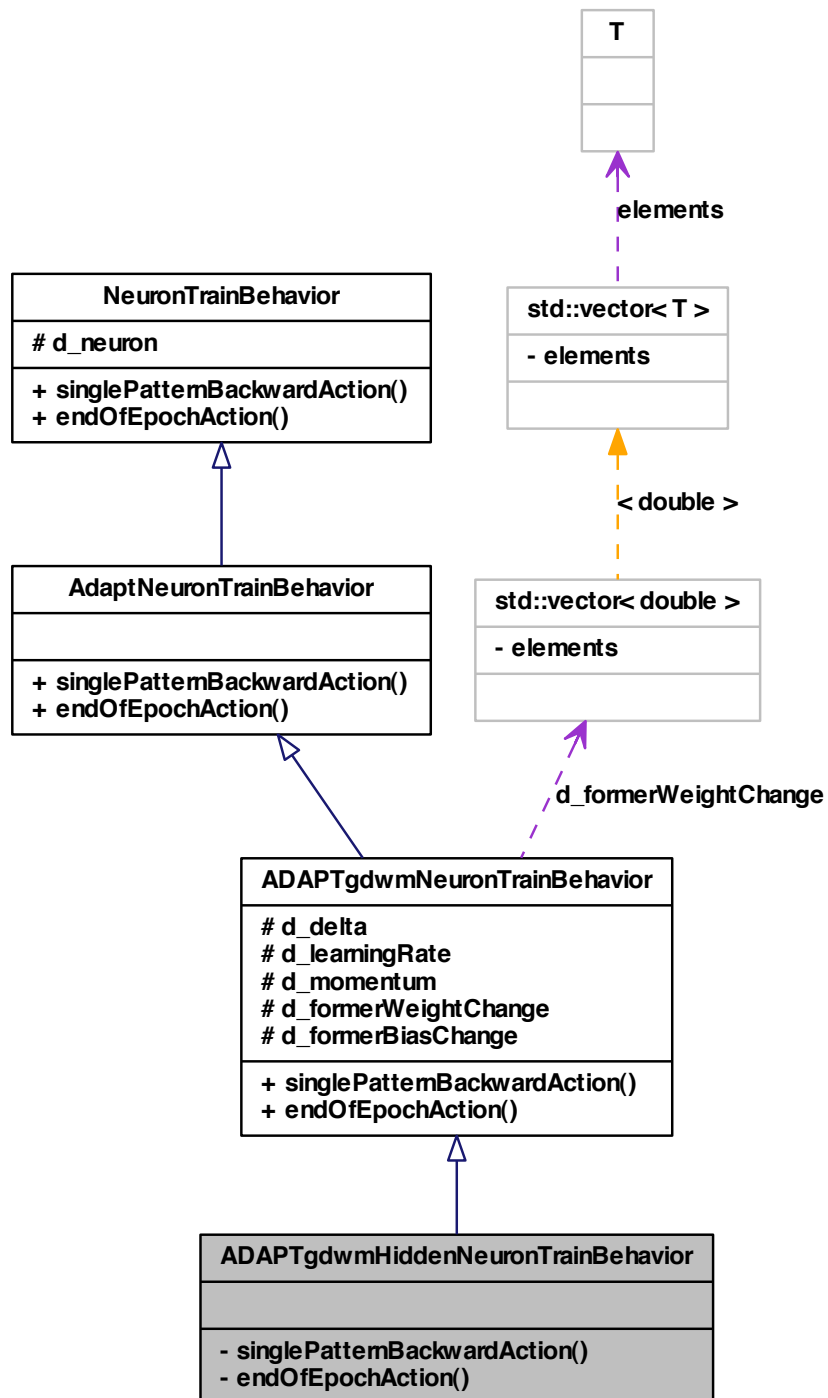
class [ADAPTgdwmHiddenNeuronTrainBehavior](#) -

```
#include <ADAPTgdwmHiddenNeuronTrainBehavior.h>
```

Inheritance diagram for ADAPTgdwmHiddenNeuronTrainBehavior:



Collaboration diagram for ADAPTgdwmHiddenNeuronTrainBehavior:



## Private Member Functions

- void [singlePatternBackwardAction](#) ()
- void [endOfEpochAction](#) ()

### 5.6.1 Detailed Description

class [ADAPTgdwmHiddenNeuronTrainBehavior](#) -

Definition at line 5 of file [ADAPTgdwmHiddenNeuronTrainBehavior.h](#).

### 5.6.2 Member Function Documentation

5.6.2.1 void [ADAPTgdwmHiddenNeuronTrainBehavior::endOfEpochAction](#) ( ) [private, virtual]

Implements [ADAPTgdwmNeuronTrainBehavior](#).

5.6.2.2 void [ADAPTgdwmHiddenNeuronTrainBehavior::singlePatternBackwardAction](#) ( ) [private, virtual]

Implements [ADAPTgdwmNeuronTrainBehavior](#).

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

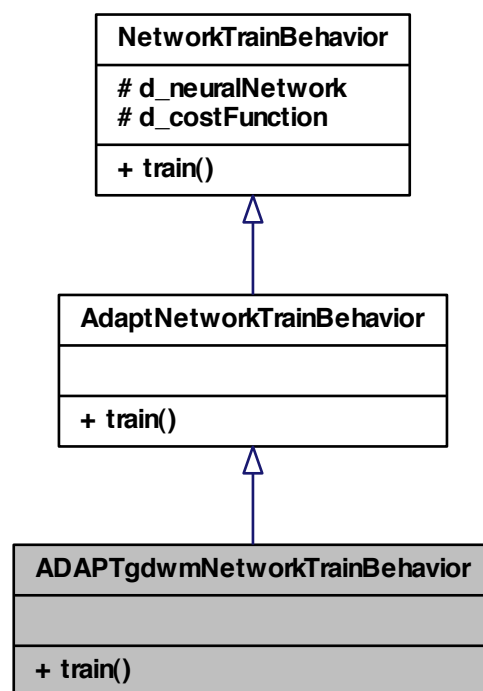
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead](#)

## 5.7 ADAPTgdwmNetworkTrainBehavior Class Reference

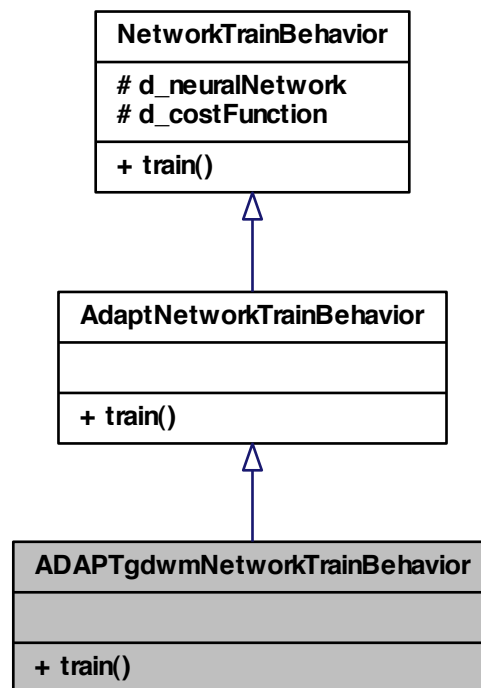
class [ADAPTgdwmNetworkTrainBehavior](#) -

```
#include <ADAPTgdwmNetworkTrainBehavior.h>
```

Inheritance diagram for ADAPTgdwmNetworkTrainBehavior:



Collaboration diagram for ADAPTgdwmNetworkTrainBehavior:



## Public Member Functions

- `Rcpp::List` [train](#) (`Rcpp::List` parameterList)

### 5.7.1 Detailed Description

class [ADAPTgdwmNetworkTrainBehavior](#) -

Definition at line 5 of file `ADAPTgdwmNetworkTrainBehavior.h`.

### 5.7.2 Member Function Documentation

5.7.2.1 Rcpp::List ADAPTgdwmNetworkTrainBehavior::train ( Rcpp::List *parameterList* )  
[virtual]

Implements [AdaptNetworkTrainBehavior](#).

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

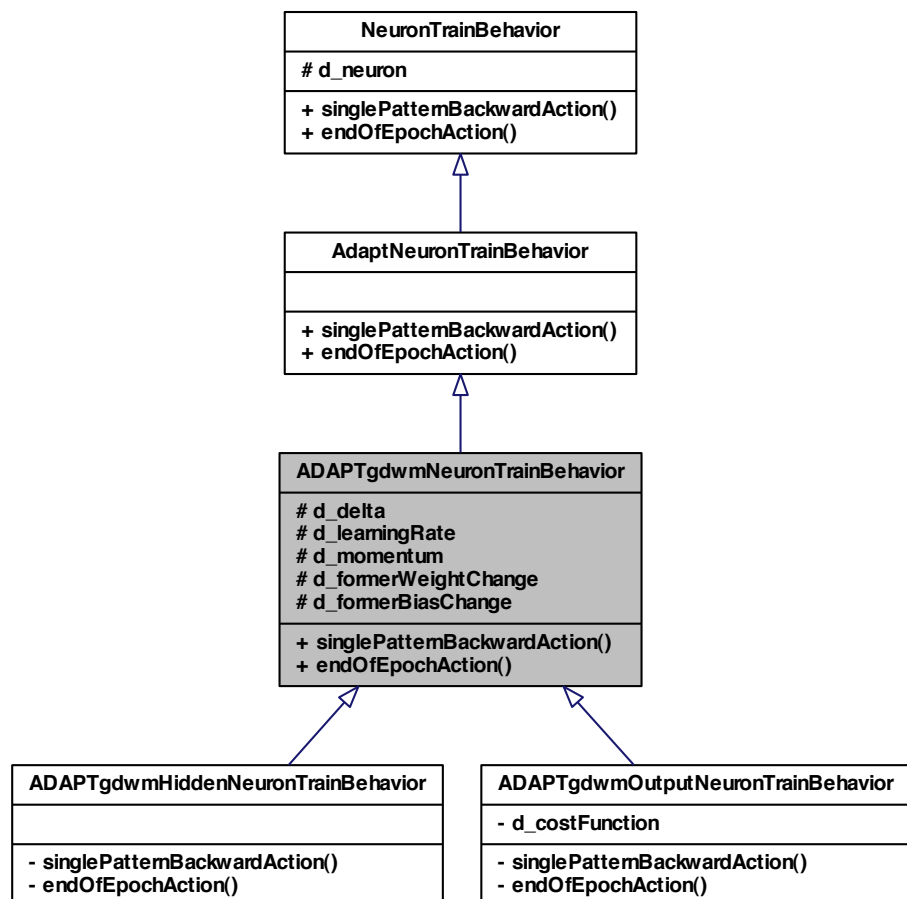
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[ADAPTgdwmNeuronTrainBehavior.h](#)

## 5.8 ADAPTgdwmNeuronTrainBehavior Class Reference

class [ADAPTgdwmNeuronTrainBehavior](#) -

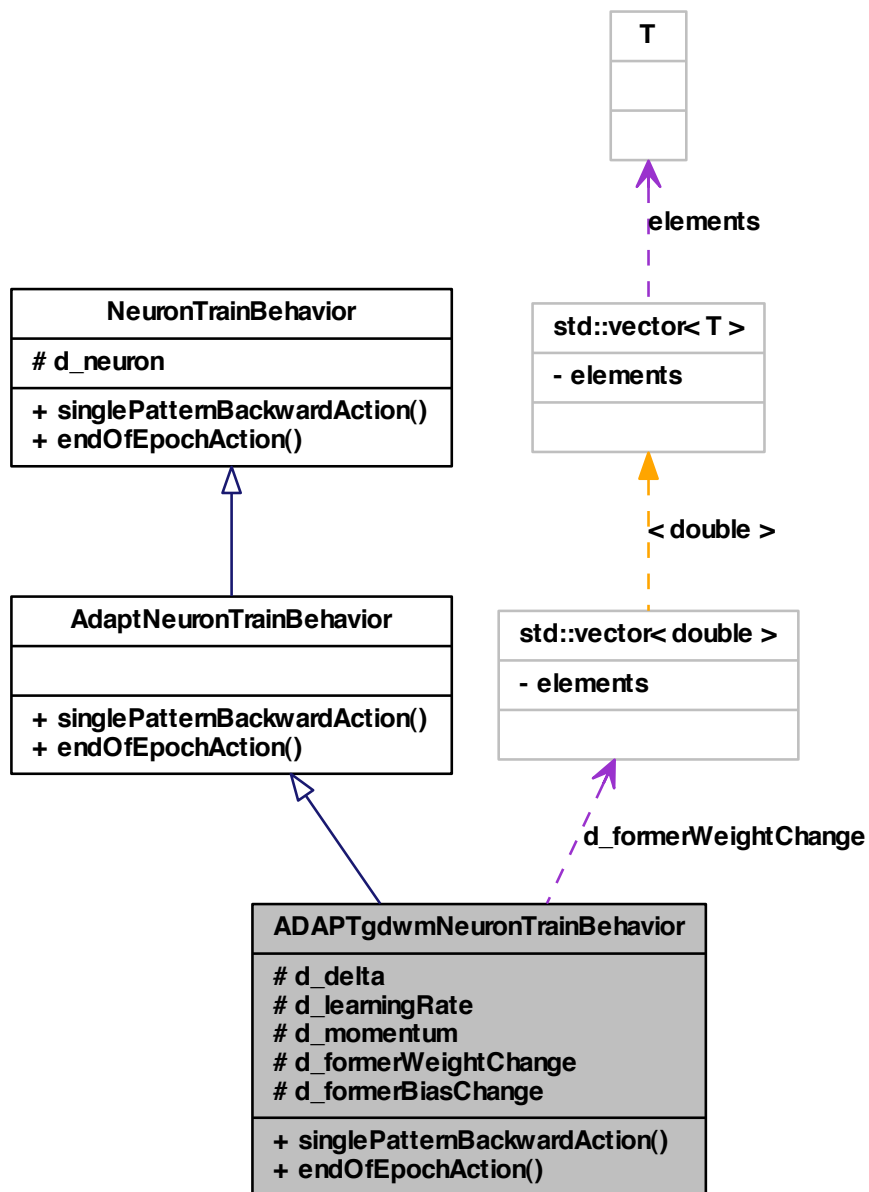
```
#include <ADAPTgdwmNeuronTrainBehavior.h>
```

Inheritance diagram for ADAPTgdmwNeuronTrainBehavior:





Collaboration diagram for ADAPTgdwmNeuronTrainBehavior:



## Public Member Functions

- virtual void [singlePatternBackwardAction](#) ()=0
- virtual void [endOfEpochAction](#) ()=0

## Protected Attributes

- double [d\\_delta](#)
- double [d\\_learningRate](#)
- double [d\\_momentum](#)
- std::vector< double > [d\\_formerWeightChange](#)
- double [d\\_formerBiasChange](#)

### 5.8.1 Detailed Description

class [ADAPTgdwmNeuronTrainBehavior](#) -

Definition at line 5 of file [ADAPTgdwmNeuronTrainBehavior.h](#).

### 5.8.2 Member Function Documentation

5.8.2.1 virtual void [ADAPTgdwmNeuronTrainBehavior::endOfEpochAction](#) ( ) [pure virtual]

Implements [AdaptNeuronTrainBehavior](#).

Implemented in [ADAPTgdwmHiddenNeuronTrainBehavior](#), and [ADAPTgdwmOutputNeuronTrainBehavior](#).

5.8.2.2 virtual void [ADAPTgdwmNeuronTrainBehavior::singlePatternBackwardAction](#) ( ) [pure virtual]

Implements [AdaptNeuronTrainBehavior](#).

Implemented in [ADAPTgdwmHiddenNeuronTrainBehavior](#), and [ADAPTgdwmOutputNeuronTrainBehavior](#).

### 5.8.3 Member Data Documentation

5.8.3.1 double [ADAPTgdwmNeuronTrainBehavior::d\\_delta](#) [protected]

Definition at line 8 of file [ADAPTgdwmNeuronTrainBehavior.h](#).

5.8.3.2 `double ADAPTgdwmNeuronTrainBehavior::d_formerBiasChange`  
[protected]

Definition at line 12 of file ADAPTgdwmNeuronTrainBehavior.h.

5.8.3.3 `std::vector<double> ADAPTgdwmNeuronTrainBehavior::d_formerWeightChange` [protected]

Definition at line 11 of file ADAPTgdwmNeuronTrainBehavior.h.

5.8.3.4 `double ADAPTgdwmNeuronTrainBehavior::d_learningRate`  
[protected]

Definition at line 9 of file ADAPTgdwmNeuronTrainBehavior.h.

5.8.3.5 `double ADAPTgdwmNeuronTrainBehavior::d_momentum` [protected]

Definition at line 10 of file ADAPTgdwmNeuronTrainBehavior.h.

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

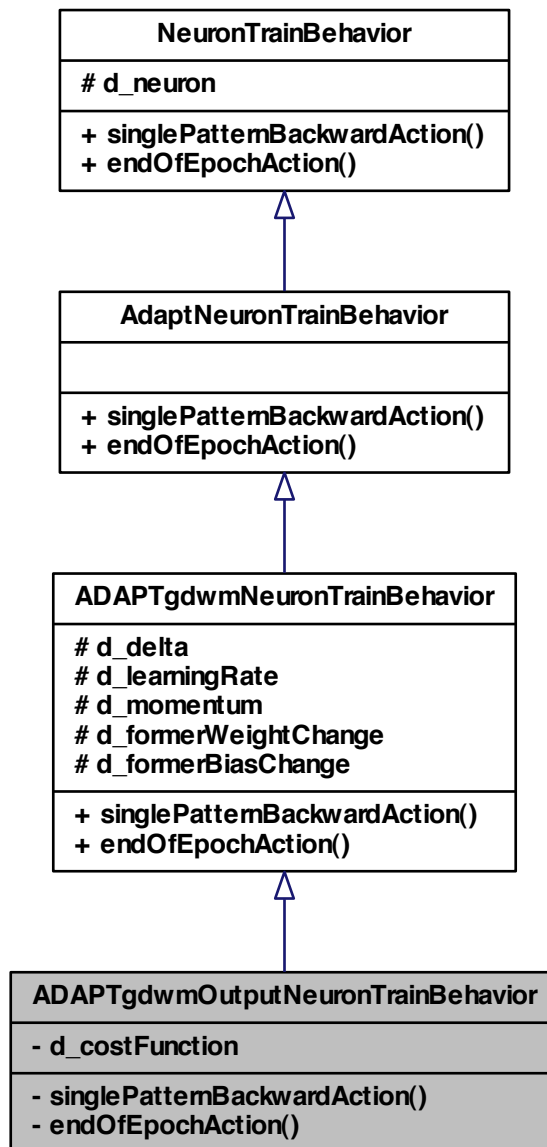
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[ADAPTgdwmOutputNeuronTrainBehavior.h](#)

## 5.9 ADAPTgdwmOutputNeuronTrainBehavior Class Reference

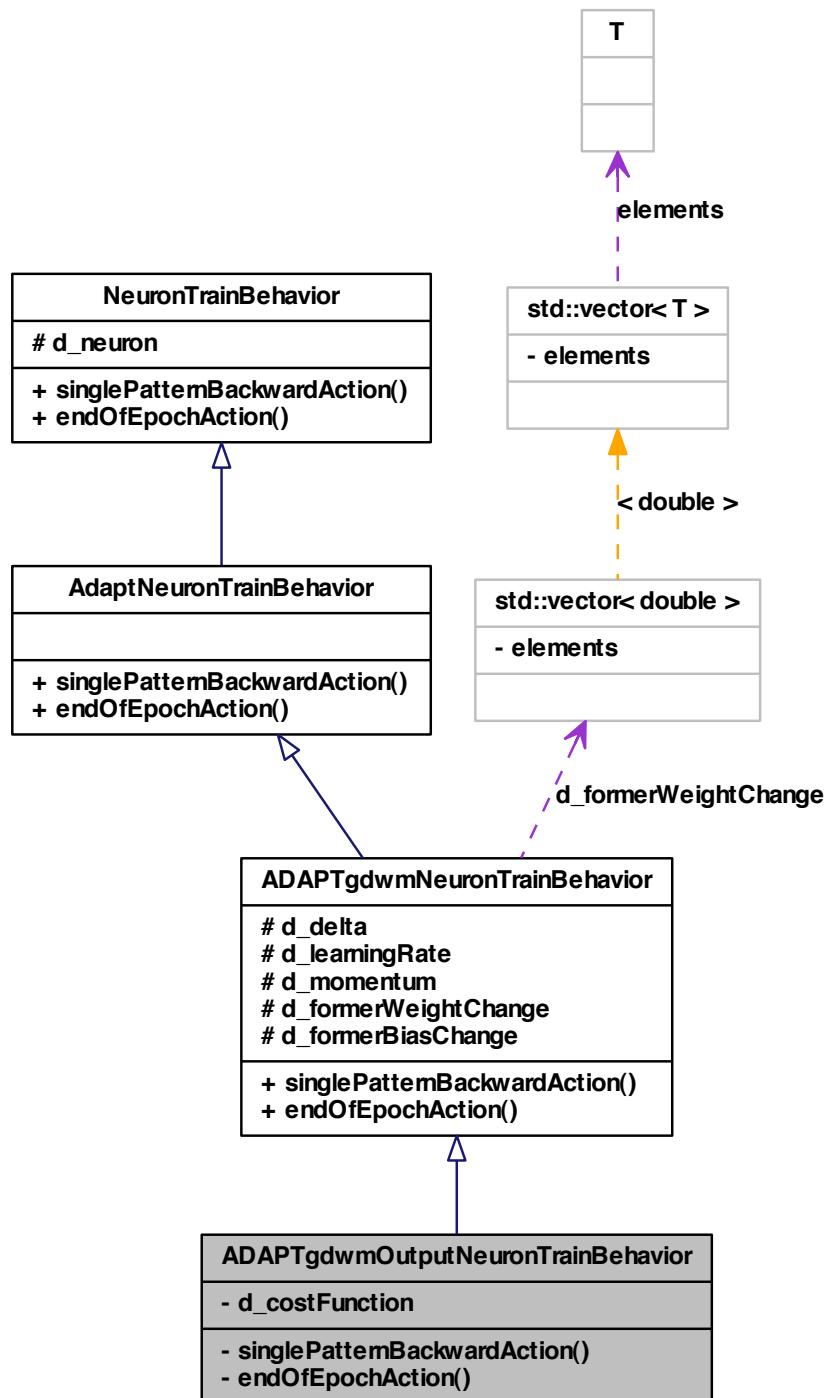
class [ADAPTgdwmOutputNeuronTrainBehavior](#) -

```
#include <ADAPTgdwmOutputNeuronTrainBehavior.h>
```

Inheritance diagram for ADAPTgdwmsOutputNeuronTrainBehavior:



Collaboration diagram for ADAPTgdwmOutputNeuronTrainBehavior:



### Private Member Functions

- void [singlePatternBackwardAction](#) ()
- void [endOfEpochAction](#) ()

### Private Attributes

- CostFuntionWeakPtr [d\\_costFunction](#)

### 5.9.1 Detailed Description

class [ADAPTgdwmOutputNeuronTrainBehavior](#) -

Definition at line 5 of file [ADAPTgdwmOutputNeuronTrainBehavior.h](#).

### 5.9.2 Member Function Documentation

5.9.2.1 void [ADAPTgdwmOutputNeuronTrainBehavior::endOfEpochAction](#) ( ) [private, virtual]

Implements [ADAPTgdwmNeuronTrainBehavior](#).

5.9.2.2 void [ADAPTgdwmOutputNeuronTrainBehavior::singlePatternBackwardAction](#) ( ) [private, virtual]

Implements [ADAPTgdwmNeuronTrainBehavior](#).

### 5.9.3 Member Data Documentation

5.9.3.1 CostFuntionWeakPtr [ADAPTgdwmOutputNeuronTrainBehavior::d\\_costFunction](#) [private]

Definition at line 8 of file [ADAPTgdwmOutputNeuronTrainBehavior.h](#).

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

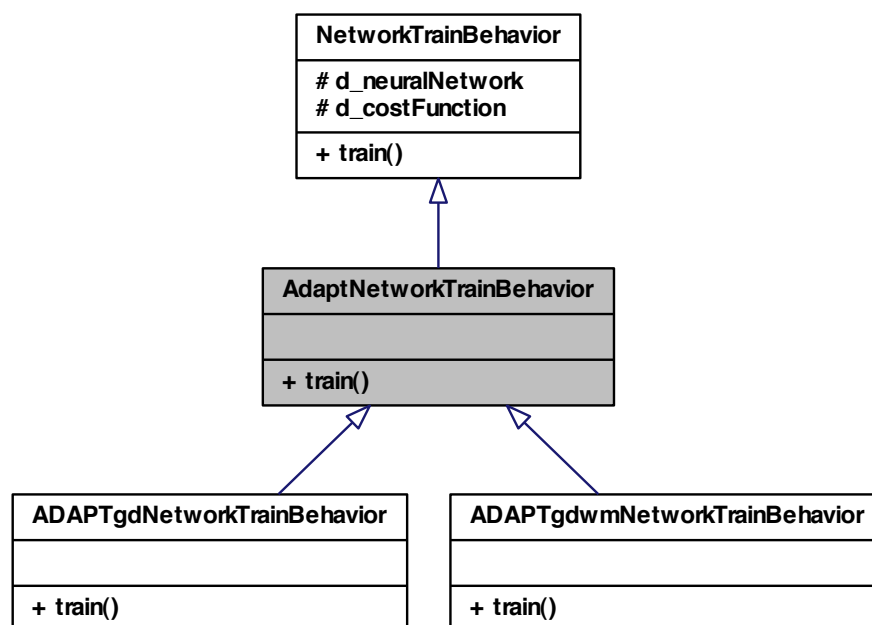
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead](#)

## 5.10 AdaptNetworkTrainBehavior Class Reference

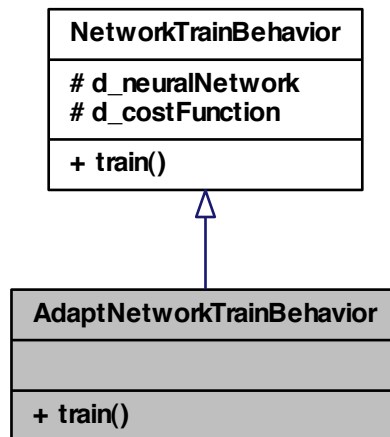
class [AdaptNetworkTrainBehavior](#) -

```
#include <AdaptNetworkTrainBehavior.h>
```

Inheritance diagram for AdaptNetworkTrainBehavior:



Collaboration diagram for AdaptNetworkTrainBehavior:



## Public Member Functions

- virtual Rcpp::List [train](#) (Rcpp::List parameterList)=0

### 5.10.1 Detailed Description

class [AdaptNetworkTrainBehavior](#) -

Definition at line 5 of file `AdaptNetworkTrainBehavior.h`.

### 5.10.2 Member Function Documentation

**5.10.2.1** virtual Rcpp::List `AdaptNetworkTrainBehavior::train` ( Rcpp::List *parameterList* )  
[pure virtual]

Implements [NetworkTrainBehavior](#).

Implemented in [ADAPTgdNetworkTrainBehavior](#), and [ADAPTgdwmNetworkTrainBehavior](#).

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

- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead`

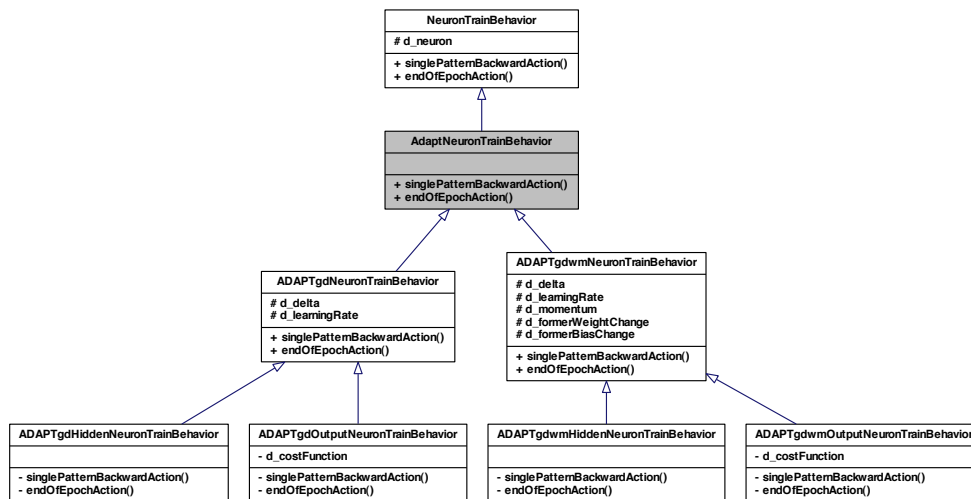


## 5.11 AdaptNeuronTrainBehavior Class Reference

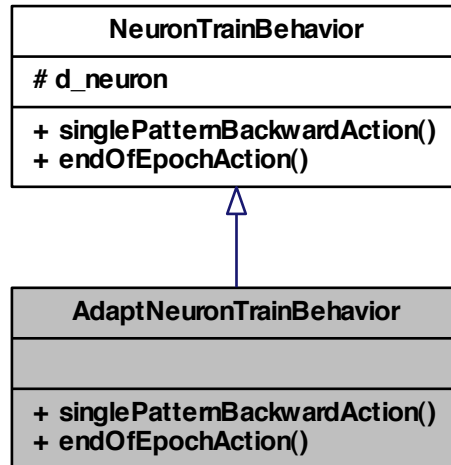
class [AdaptNeuronTrainBehavior](#) -

```
#include <AdaptNeuronTrainBehavior.h>
```

Inheritance diagram for AdaptNeuronTrainBehavior:



Collaboration diagram for `AdaptNeuronTrainBehavior`:



### Public Member Functions

- virtual void `singlePatternBackwardAction` ()=0
- virtual void `endOfEpochAction` ()=0

#### 5.11.1 Detailed Description

class `AdaptNeuronTrainBehavior` -

Definition at line 5 of file `AdaptNeuronTrainBehavior.h`.

#### 5.11.2 Member Function Documentation

5.11.2.1 virtual void `AdaptNeuronTrainBehavior::endOfEpochAction` ( ) [pure virtual]

Implements `NeuronTrainBehavior`.

Implemented in `ADAPTgdHiddenNeuronTrainBehavior`, `ADAPTgdNeuronTrainBehavior`, `ADAPTgdOutputNeuronTrainBehavior`, `ADAPTgdwmHiddenNeuronTrainBehavior`, `ADAPTgdwmNeuronTrainBehavior`, and `ADAPTgdwmOutputNeuronTrainBehavior`.

5.11.2.2 `virtual void AdaptNeuronTrainBehavior::singlePatternBackwardAction ( ) [pure virtual]`

Implements [NeuronTrainBehavior](#).

Implemented in [ADAPTgdHiddenNeuronTrainBehavior](#), [ADAPTgdNeuronTrainBehavior](#), [ADAPTgdOutputNeuronTrainBehavior](#), [ADAPTgdwmHiddenNeuronTrainBehavior](#), [ADAPTgdwmNeuronTrainBehavior](#), and [ADAPTgdwmOutputNeuronTrainBehavior](#).

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

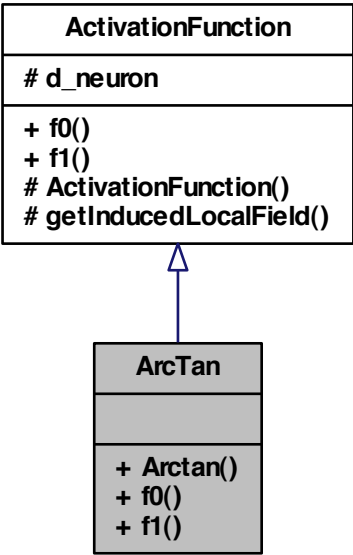
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptNeu`

5.12 ArcTan Class Reference

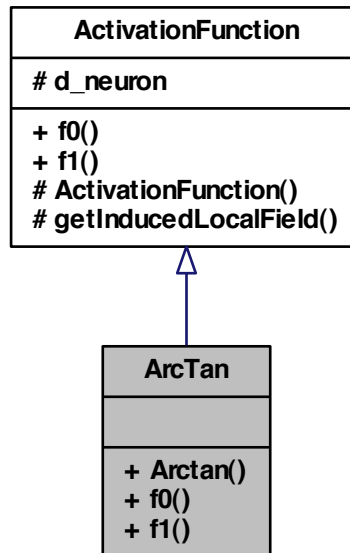
class [ArcTan](#) -

`#include <ArcTan.h>`

Inheritance diagram for ArcTan:



Collaboration diagram for ArcTan:



## Public Member Functions

- [Arctan](#) ([NeuronPtr](#) neuronPtr)
- double [f0](#) ()
- double [f1](#) ()

### 5.12.1 Detailed Description

class [ArcTan](#) -

Definition at line 5 of file [ArcTan.h](#).

### 5.12.2 Member Function Documentation

5.12.2.1 [ArcTan::Arctan](#) ( [NeuronPtr](#) neuronPtr )

5.12.2.2 double [ArcTan::f0](#) ( ) [virtual]

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

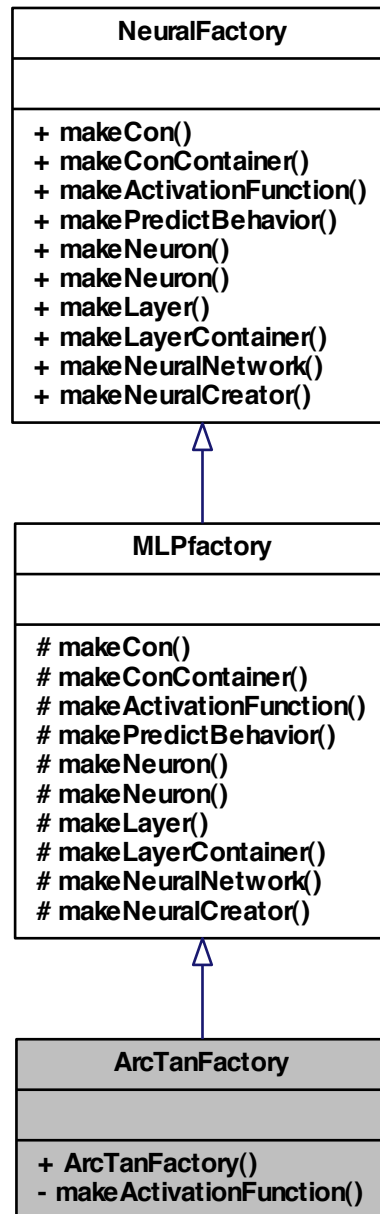
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ArcTan.h`

## 5.13 ArcTanFactory Class Reference

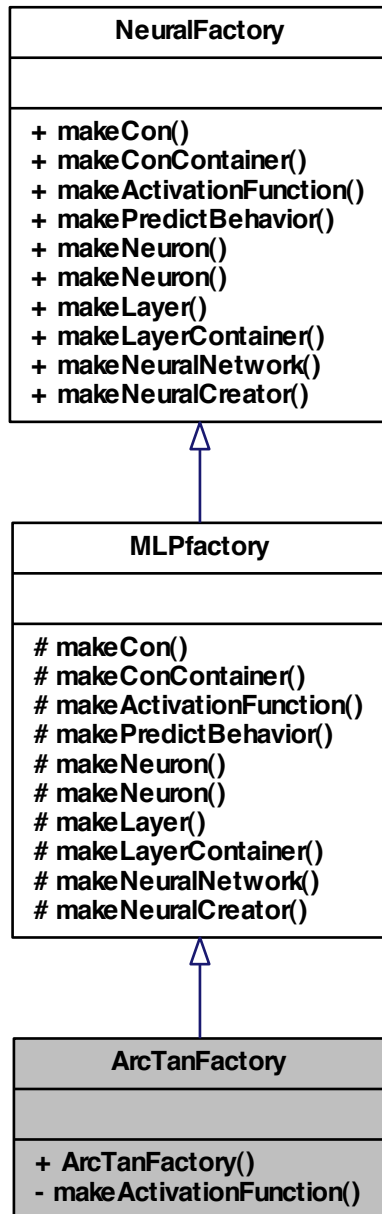
class [ArcTanFactory](#) -

```
#include <ArcTanFactory.h>
```

Inheritance diagram for ArcTanFactory:



Collaboration diagram for ArcTanFactory:



## Public Member Functions

- [ArcTanFactory](#) ()

## Private Member Functions

- [ActivationFunctionPtr](#) [makeActivationFunction](#) ([NeuronPtr](#) neuronPtr)

### 5.13.1 Detailed Description

class [ArcTanFactory](#) -

Definition at line 5 of file ArcTanFactory.h.

### 5.13.2 Constructor & Destructor Documentation

5.13.2.1 [ArcTanFactory::ArcTanFactory](#) ( )

### 5.13.3 Member Function Documentation

5.13.3.1 [ActivationFunctionPtr](#) [ArcTanFactory::makeActivationFunction](#) ( [NeuronPtr](#) *neuronPtr* ) [private, virtual]

Implements [MLPfactory](#).

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

- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

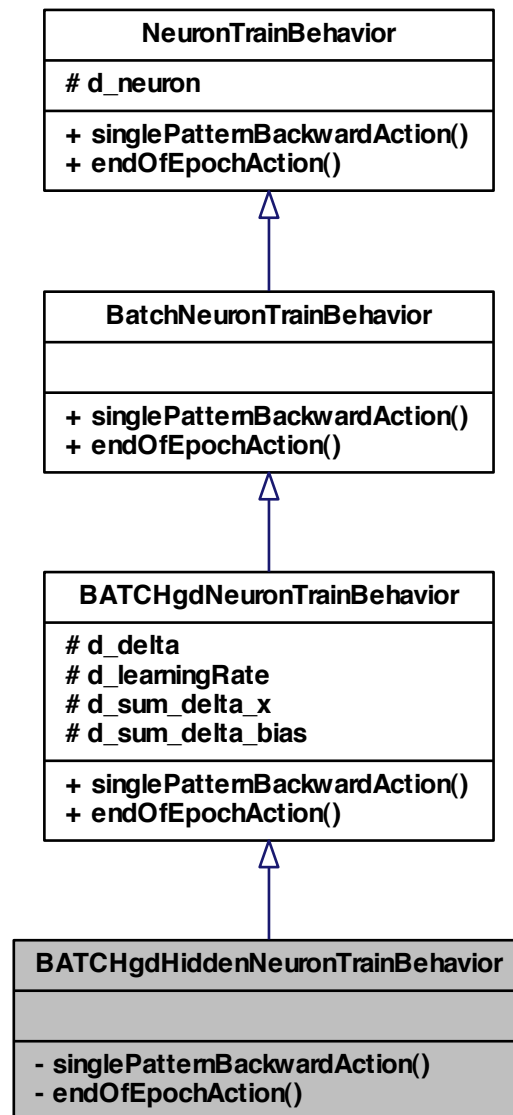
## 5.14 BATCHgdHiddenNeuronTrainBehavior Class Reference

class [BATCHgdHiddenNeuronTrainBehavior](#) -

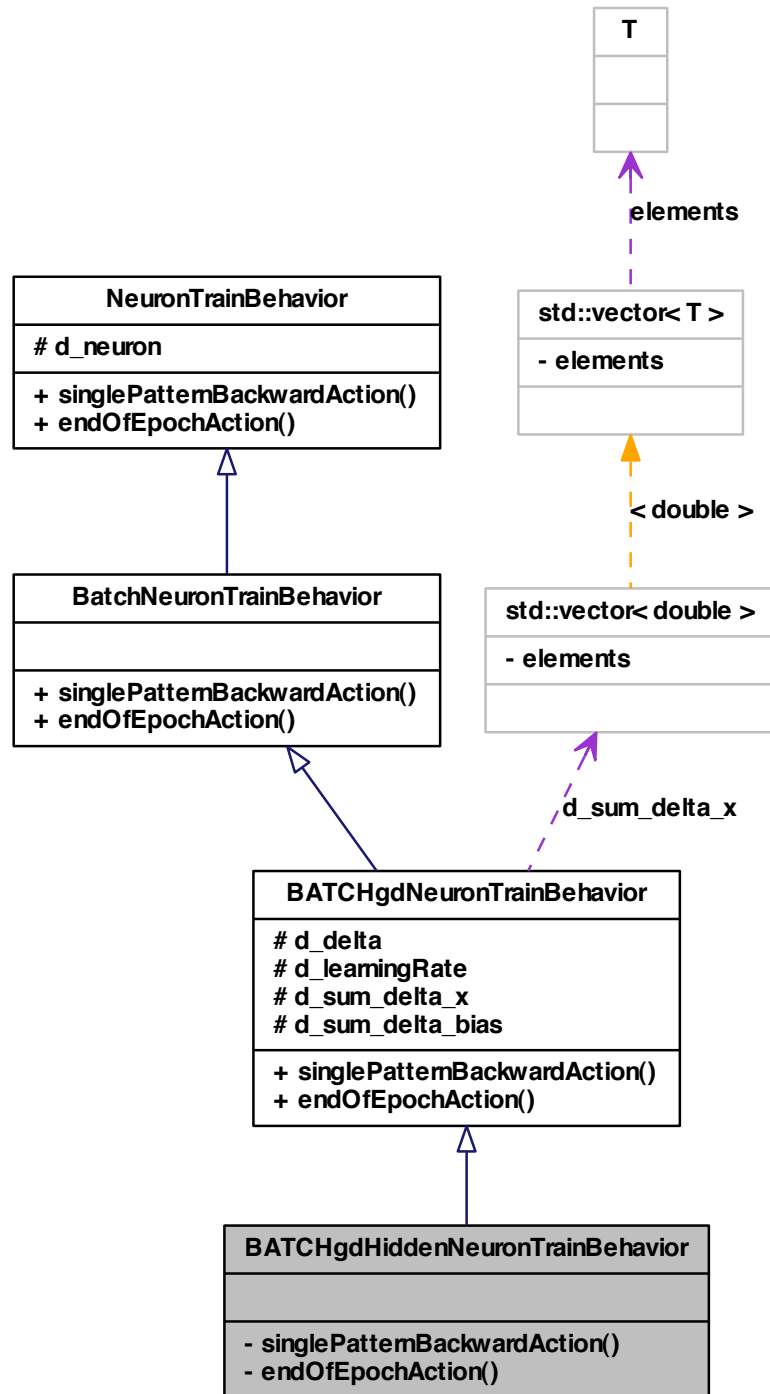
```
#include <BATCHgdHiddenNeuronTrainBehavior.h>
```



Inheritance diagram for BATCHgdHiddenNeuronTrainBehavior:



Collaboration diagram for BATCHgdHiddenNeuronTrainBehavior:



### Private Member Functions

- void [singlePatternBackwardAction](#) ()
- void [endOfEpochAction](#) ()

#### 5.14.1 Detailed Description

class [BATCHgdHiddenNeuronTrainBehavior](#) -

Definition at line 5 of file [BATCHgdHiddenNeuronTrainBehavior.h](#).

#### 5.14.2 Member Function Documentation

5.14.2.1 void [BATCHgdHiddenNeuronTrainBehavior::endOfEpochAction](#) ( ) [private, virtual]

Implements [BATCHgdNeuronTrainBehavior](#).

5.14.2.2 void [BATCHgdHiddenNeuronTrainBehavior::singlePatternBackwardAction](#) ( ) [private, virtual]

Implements [BATCHgdNeuronTrainBehavior](#).

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

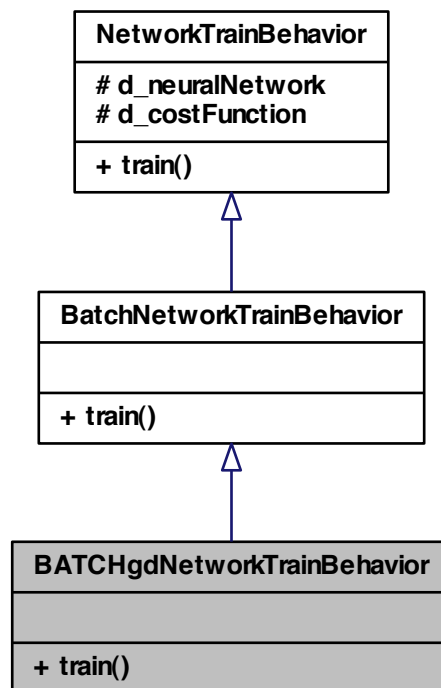
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgd](#)

## 5.15 BATCHgdNetworkTrainBehavior Class Reference

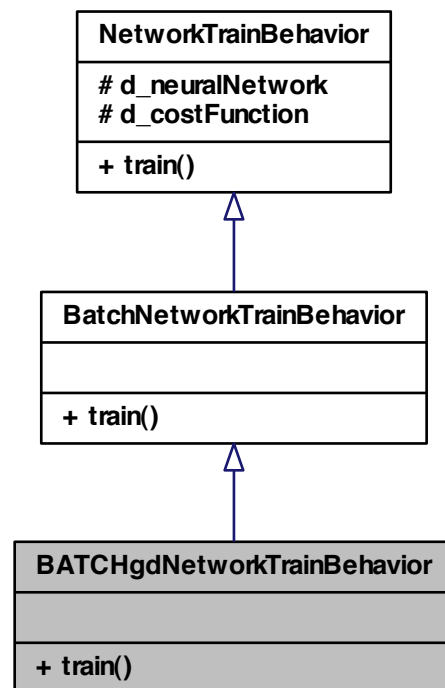
class [BATCHgdNetworkTrainBehavior](#) -

```
#include <BATCHgdNetworkTrainBehavior.h>
```

Inheritance diagram for BATCHgdNetworkTrainBehavior:



Collaboration diagram for BATCHgdNetworkTrainBehavior:



## Public Member Functions

- `Rcpp::List` [train](#) (`Rcpp::List` parameterList)

### 5.15.1 Detailed Description

class [BATCHgdNetworkTrainBehavior](#) -

Definition at line 5 of file `BATCHgdNetworkTrainBehavior.h`.

### 5.15.2 Member Function Documentation

5.15.2.1 `Rcpp::List BATCHgdNetworkTrainBehavior::train ( Rcpp::List parameterList )`  
[virtual]

Implements [BatchNetworkTrainBehavior](#).

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

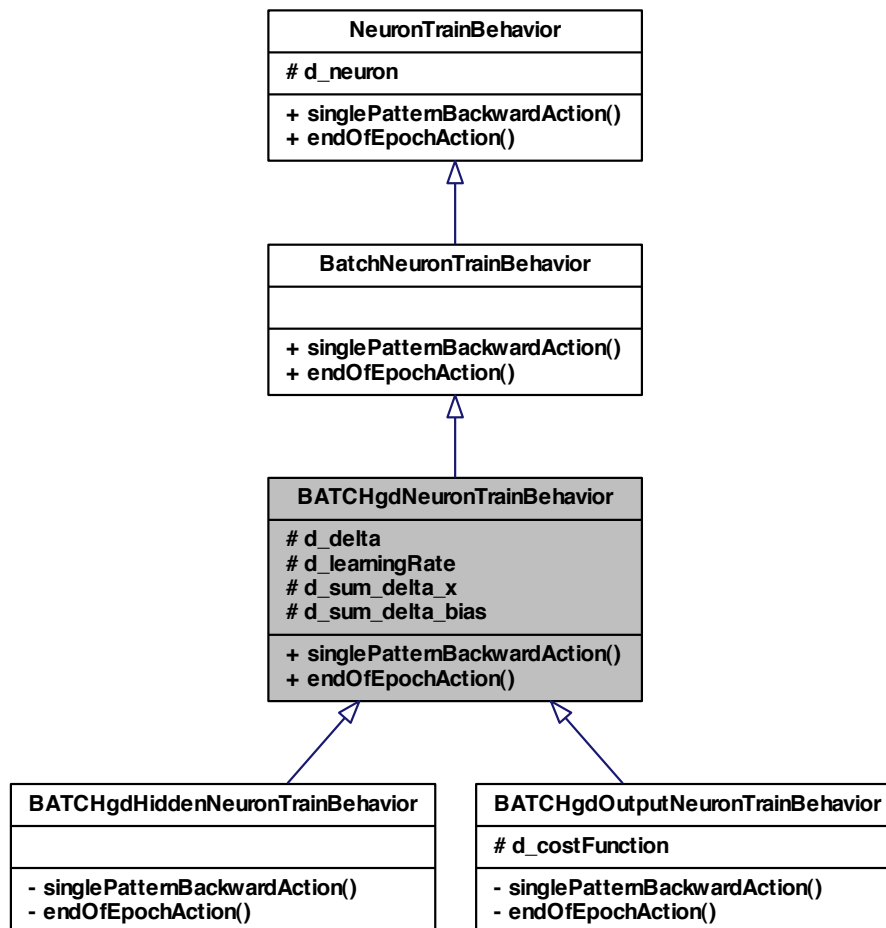
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.16 BATCHgdNeuronTrainBehavior Class Reference

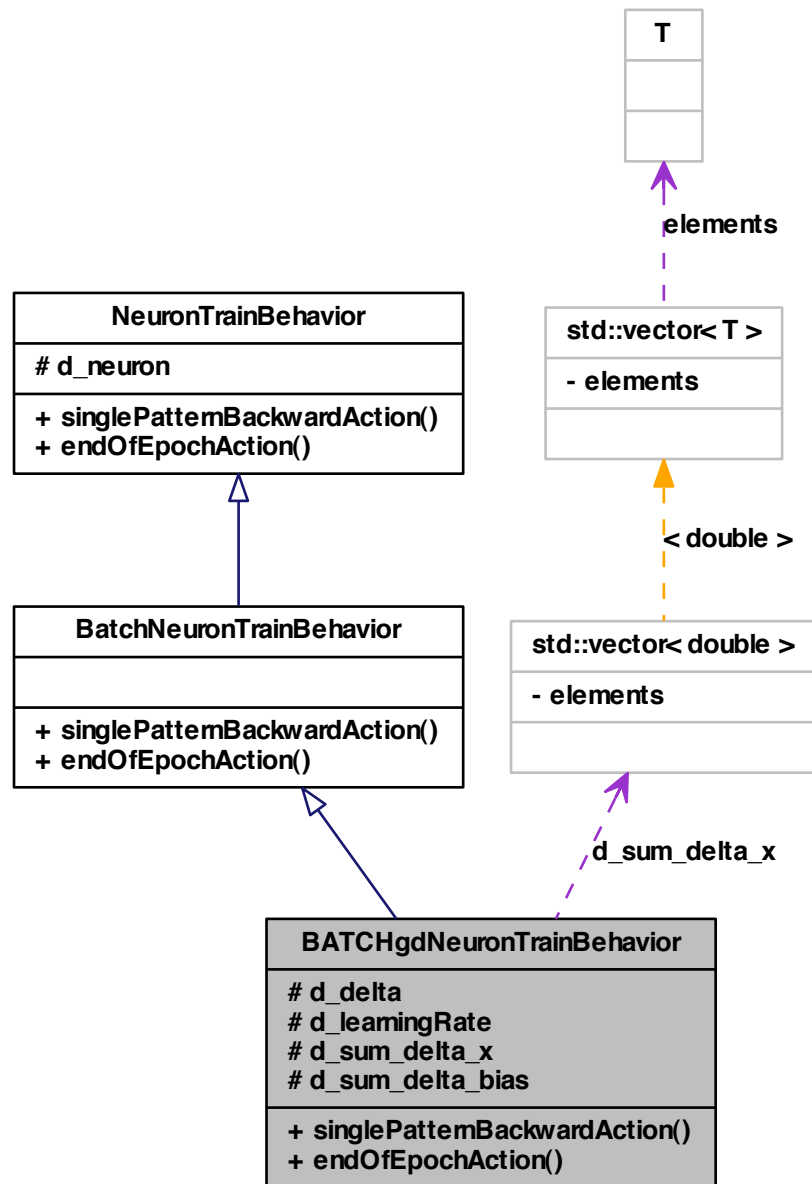
class [BATCHgdNeuronTrainBehavior](#) -

```
#include <BATCHgdNeuronTrainBehavior.h>
```

Inheritance diagram for BATCHgdNeuronTrainBehavior:



Collaboration diagram for BATCHgdNeuronTrainBehavior:





## Public Member Functions

- virtual void [singlePatternBackwardAction](#) ()=0
- virtual void [endOfEpochAction](#) ()=0

## Protected Attributes

- double [d\\_delta](#)
- double [d\\_learningRate](#)
- std::vector< double > [d\\_sum\\_delta\\_x](#)
- double [d\\_sum\\_delta\\_bias](#)

### 5.16.1 Detailed Description

class [BATCHgdNeuronTrainBehavior](#) -

Definition at line 5 of file [BATCHgdNeuronTrainBehavior.h](#).

### 5.16.2 Member Function Documentation

**5.16.2.1** virtual void [BATCHgdNeuronTrainBehavior::endOfEpochAction](#) ( ) [pure virtual]

Implements [BatchNeuronTrainBehavior](#).

Implemented in [BATCHgdHiddenNeuronTrainBehavior](#), and [BATCHgdOutputNeuronTrainBehavior](#).

**5.16.2.2** virtual void [BATCHgdNeuronTrainBehavior::singlePatternBackwardAction](#) ( ) [pure virtual]

Implements [BatchNeuronTrainBehavior](#).

Implemented in [BATCHgdHiddenNeuronTrainBehavior](#), and [BATCHgdOutputNeuronTrainBehavior](#).

### 5.16.3 Member Data Documentation

**5.16.3.1** double [BATCHgdNeuronTrainBehavior::d\\_delta](#) [protected]

Definition at line 8 of file [BATCHgdNeuronTrainBehavior.h](#).

**5.16.3.2** double [BATCHgdNeuronTrainBehavior::d\\_learningRate](#) [protected]

Definition at line 9 of file [BATCHgdNeuronTrainBehavior.h](#).

5.16.3.3 `double BATCHgdNeuronTrainBehavior::d_sum_delta_bias`  
[protected]

Definition at line 11 of file BATCHgdNeuronTrainBehavior.h.

5.16.3.4 `std::vector<double> BATCHgdNeuronTrainBehavior::d_sum_delta_x`  
[protected]

Definition at line 10 of file BATCHgdNeuronTrainBehavior.h.

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

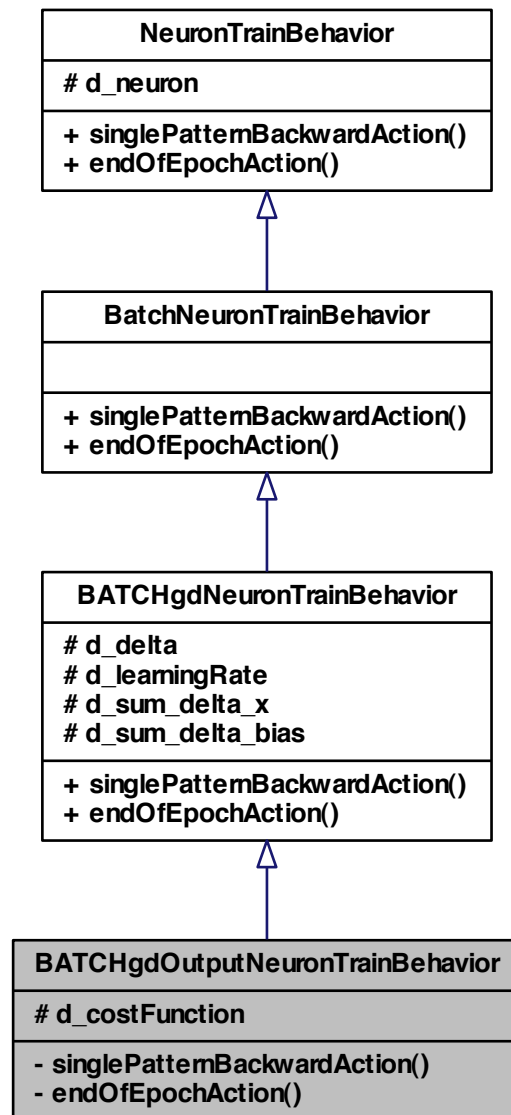
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.17 BATCHgdOutputNeuronTrainBehavior Class Reference

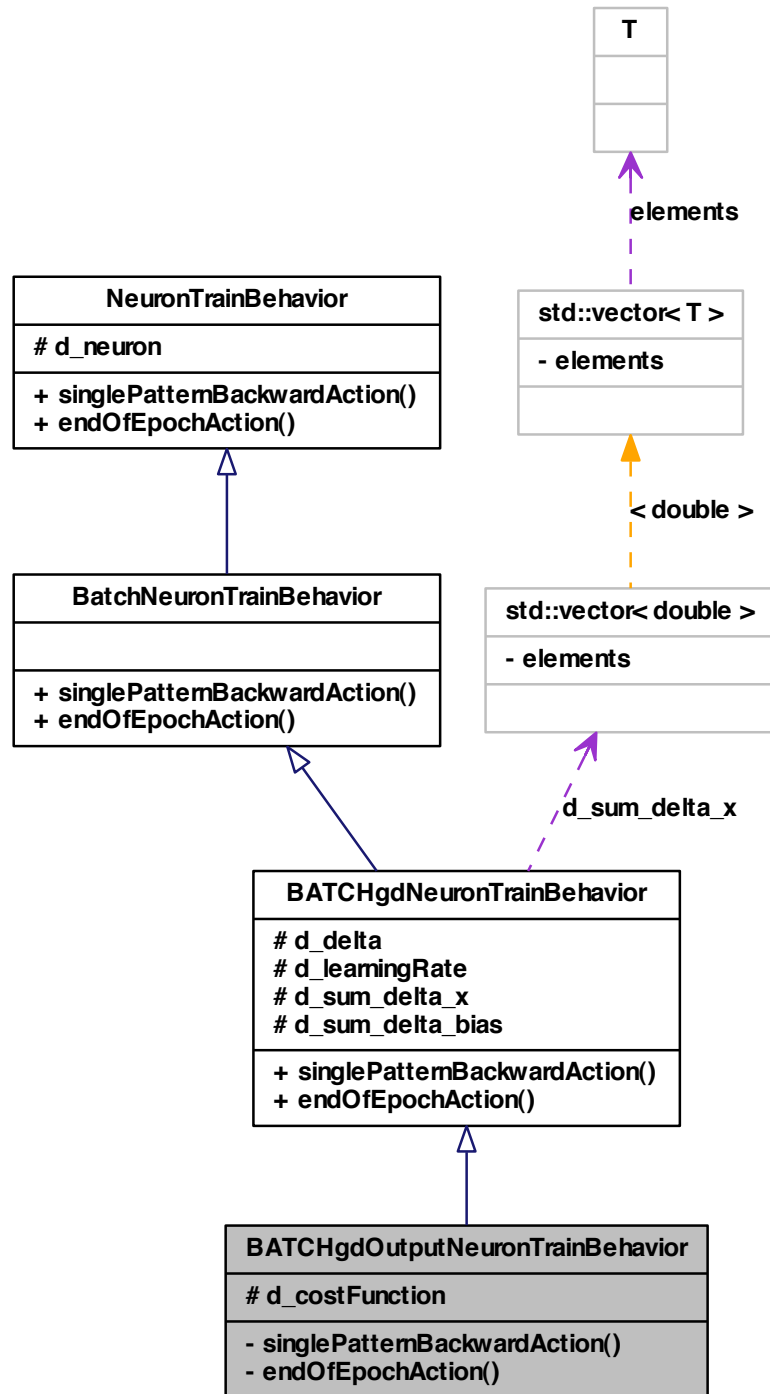
class [BATCHgdOutputNeuronTrainBehavior](#) -

```
#include <BATCHgdOutputNeuronTrainBehavior.h>
```

Inheritance diagram for BATCHgdOutputNeuronTrainBehavior:



Collaboration diagram for BATCHgdOutputNeuronTrainBehavior:



### Protected Attributes

- CostFunctionWeakPtr [d\\_costFunction](#)

### Private Member Functions

- void [singlePatternBackwardAction](#) ()
- void [endOfEpochAction](#) ()

#### 5.17.1 Detailed Description

class [BATCHgdOutputNeuronTrainBehavior](#) -

Definition at line 5 of file [BATCHgdOutputNeuronTrainBehavior.h](#).

#### 5.17.2 Member Function Documentation

5.17.2.1 void [BATCHgdOutputNeuronTrainBehavior::endOfEpochAction](#) ( ) [private, virtual]

Implements [BATCHgdNeuronTrainBehavior](#).

5.17.2.2 void [BATCHgdOutputNeuronTrainBehavior::singlePatternBackwardAction](#) ( ) [private, virtual]

Implements [BATCHgdNeuronTrainBehavior](#).

#### 5.17.3 Member Data Documentation

5.17.3.1 CostFunctionWeakPtr [BATCHgdOutputNeuronTrainBehavior::d\\_costFunction](#) [protected]

Definition at line 8 of file [BATCHgdOutputNeuronTrainBehavior.h](#).

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

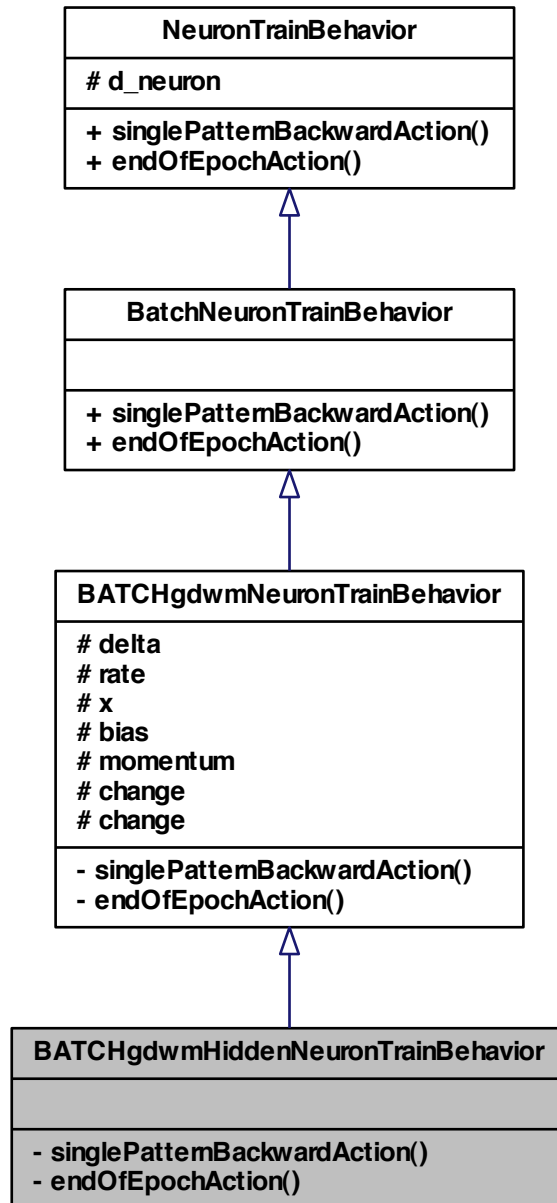
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdOutputNeuronTrainBehavior.h](#)

## 5.18 BATCHgdwmHiddenNeuronTrainBehavior Class Reference

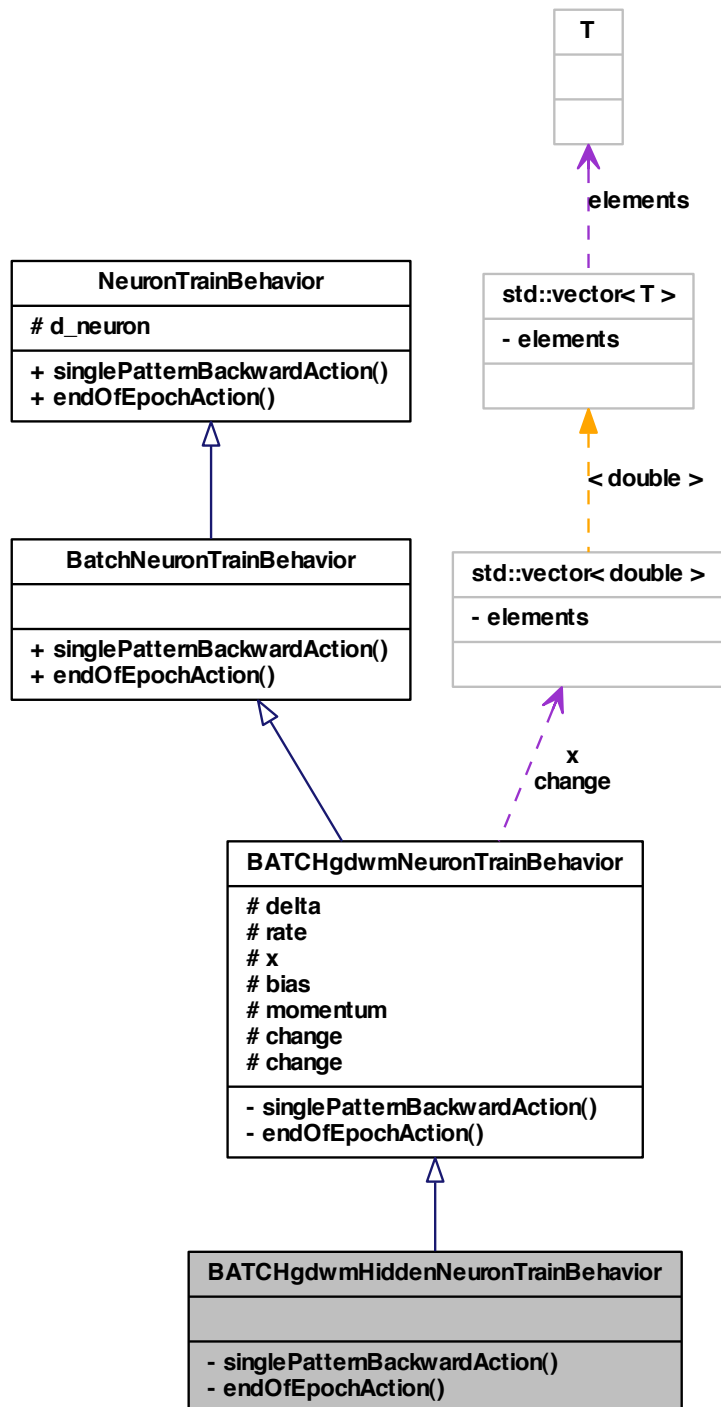
class [BATCHgdwmHiddenNeuronTrainBehavior](#) -

```
#include <BATCHgdwmHiddenNeuronTrainBehavior.h>
```

Inheritance diagram for BATCHgdwmHiddenNeuronTrainBehavior:



Collaboration diagram for BATCHgdwmHiddenNeuronTrainBehavior:



## Private Member Functions

- void [singlePatternBackwardAction](#) ()
- void [endOfEpochAction](#) ()

### 5.18.1 Detailed Description

class [BATCHgdwmHiddenNeuronTrainBehavior](#) -

Definition at line 5 of file [BATCHgdwmHiddenNeuronTrainBehavior.h](#).

### 5.18.2 Member Function Documentation

5.18.2.1 void [BATCHgdwmHiddenNeuronTrainBehavior::endOfEpochAction](#) ( )  
[private, virtual]

Implements [BATCHgdwmNeuronTrainBehavior](#).

5.18.2.2 void [BATCHgdwmHiddenNeuronTrainBehavior::singlePatternBackwardAction](#) ( )  
[private, virtual]

Implements [BATCHgdwmNeuronTrainBehavior](#).

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

- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead](#)

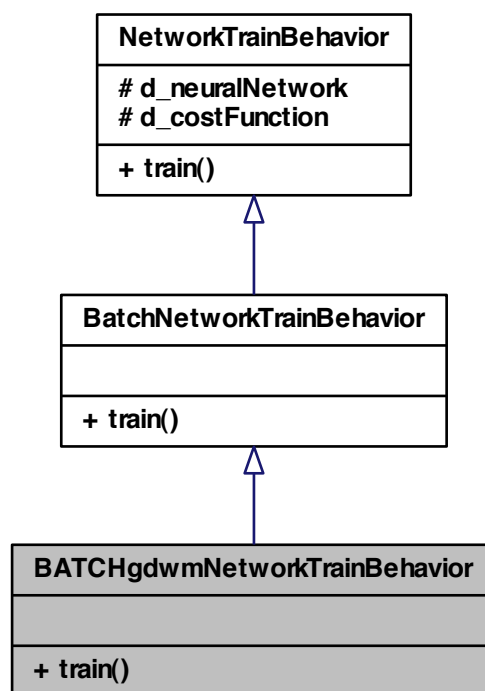
## 5.19 BATCHgdwmNetworkTrainBehavior Class Reference

class [BATCHgdwmNetworkTrainBehavior](#) -

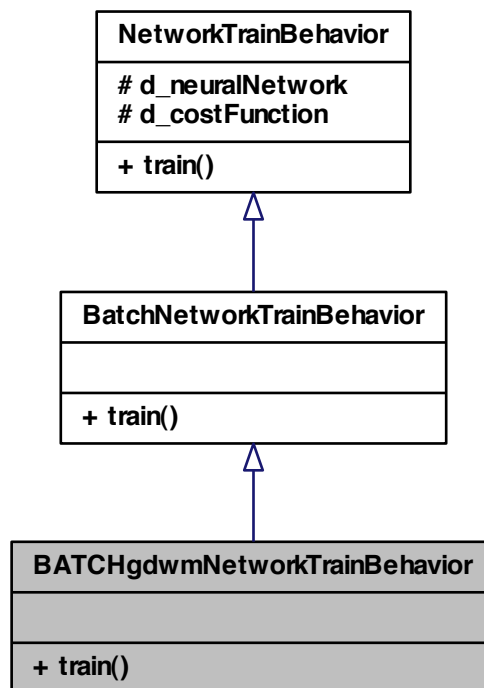
```
#include <BATCHgdwmNetworkTrainBehavior.h>
```



Inheritance diagram for BATCHgdwmNetworkTrainBehavior:



Collaboration diagram for BATCHgdwmNetworkTrainBehavior:



## Public Member Functions

- `Rcpp::List` [train](#) (`Rcpp::List` parameterList)

### 5.19.1 Detailed Description

class [BATCHgdwmNetworkTrainBehavior](#) -

Definition at line 5 of file `BATCHgdwmNetworkTrainBehavior.h`.

### 5.19.2 Member Function Documentation

5.19.2.1 `Rcpp::List BATCHgdwmNetworkTrainBehavior::train ( Rcpp::List parameterList )`  
[virtual]

Implements [BatchNetworkTrainBehavior](#).

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

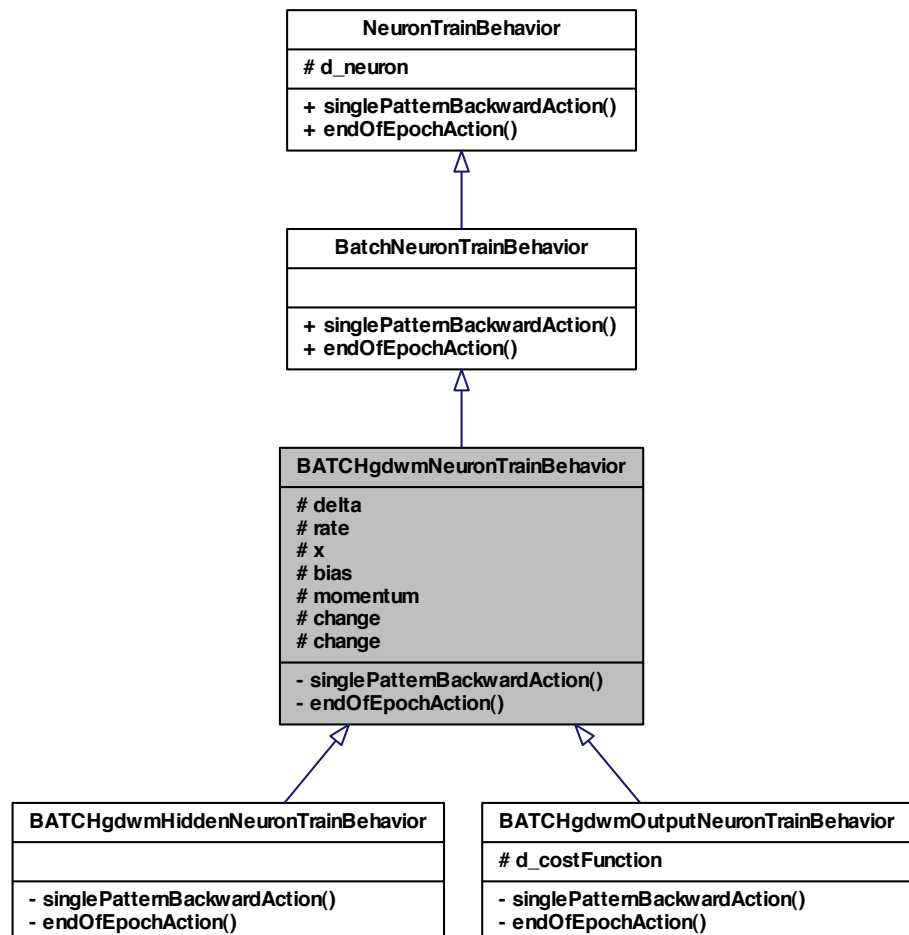
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwmNeuronTrainBehavior.h`

## 5.20 BATCHgdwmNeuronTrainBehavior Class Reference

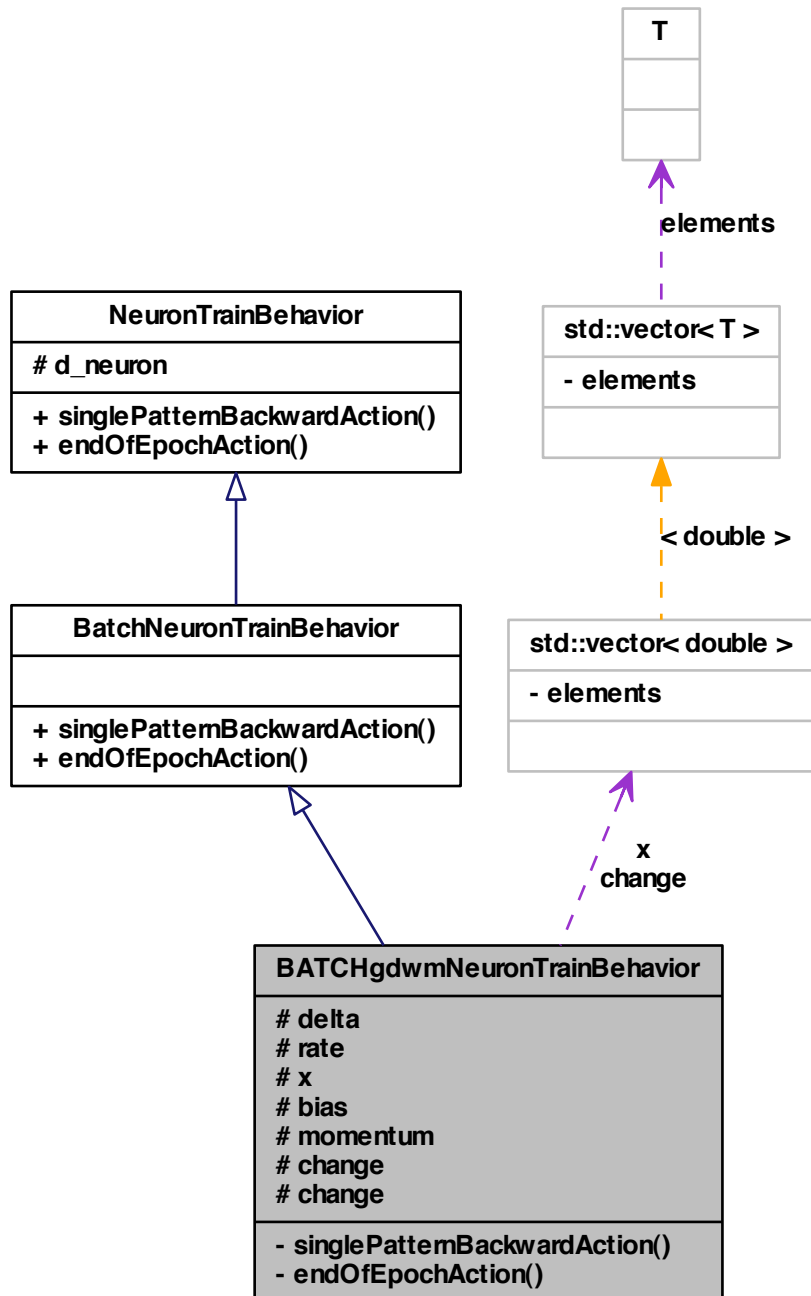
class [BATCHgdwmNeuronTrainBehavior](#) -

```
#include <BATCHgdwmNeuronTrainBehavior.h>
```

Inheritance diagram for BATCHgdwmNeuronTrainBehavior:



Collaboration diagram for BATCHgdwmNeuronTrainBehavior:



### Protected Attributes

- double [delta](#)
- double learning [rate](#)
- std::vector< double > sum [delta x](#)
- double sum [delta bias](#)
- double [momentum](#)
- std::vector< double > former weight [change](#)
- double former [bias change](#)

### Private Member Functions

- virtual void [singlePatternBackwardAction](#) ()=0
- virtual void [endOfEpochAction](#) ()=0

#### 5.20.1 Detailed Description

class [BATCHgdwmNeuronTrainBehavior](#) -

Definition at line 5 of file [BATCHgdwmNeuronTrainBehavior.h](#).

#### 5.20.2 Member Function Documentation

5.20.2.1 virtual void [BATCHgdwmNeuronTrainBehavior::endOfEpochAction](#) ( )  
[private, pure virtual]

Implements [BatchNeuronTrainBehavior](#).

Implemented in [BATCHgdwmHiddenNeuronTrainBehavior](#), and [BATCHgdwmOutputNeuronTrainBehavior](#).

5.20.2.2 virtual void [BATCHgdwmNeuronTrainBehavior::singlePatternBackwardAction](#) ( )  
[private, pure virtual]

Implements [BatchNeuronTrainBehavior](#).

Implemented in [BATCHgdwmHiddenNeuronTrainBehavior](#), and [BATCHgdwmOutputNeuronTrainBehavior](#).

#### 5.20.3 Member Data Documentation

5.20.3.1 double sum [delta](#) [BATCHgdwmNeuronTrainBehavior::bias](#)  
[protected]

Definition at line 11 of file [BATCHgdwmNeuronTrainBehavior.h](#).

5.20.3.2 `double` former bias `BATCHgdwmNeuronTrainBehavior::change`  
[protected]

Definition at line 14 of file `BATCHgdwmNeuronTrainBehavior.h`.

5.20.3.3 `std::vector<double>` former weight `BATCHgdwmNeuronTrainBehavior::change`  
[protected]

Definition at line 13 of file `BATCHgdwmNeuronTrainBehavior.h`.

5.20.3.4 `double` `BATCHgdwmNeuronTrainBehavior::delta` [protected]

Definition at line 8 of file `BATCHgdwmNeuronTrainBehavior.h`.

5.20.3.5 `double` `BATCHgdwmNeuronTrainBehavior::momentum` [protected]

Definition at line 12 of file `BATCHgdwmNeuronTrainBehavior.h`.

5.20.3.6 `double` learning `BATCHgdwmNeuronTrainBehavior::rate` [protected]

Definition at line 9 of file `BATCHgdwmNeuronTrainBehavior.h`.

5.20.3.7 `std::vector<double>` sum delta `BATCHgdwmNeuronTrainBehavior::x`  
[protected]

Definition at line 10 of file `BATCHgdwmNeuronTrainBehavior.h`.

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

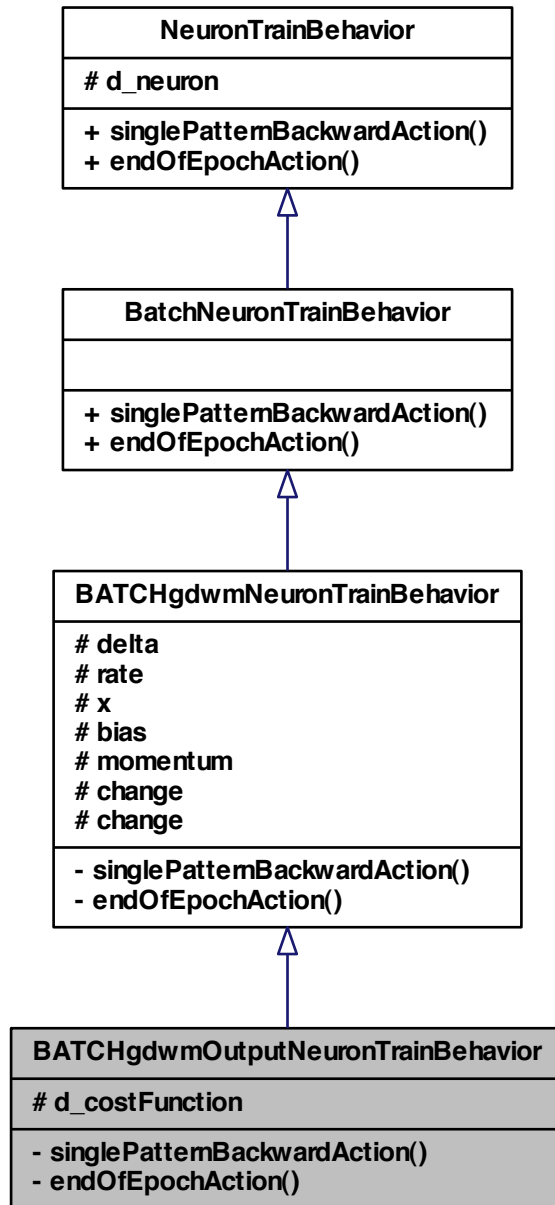
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwmOutputNeuronTrainBehavior.h`

## 5.21 BATCHgdwmOutputNeuronTrainBehavior Class Reference

class `BATCHgdwmOutputNeuronTrainBehavior` -

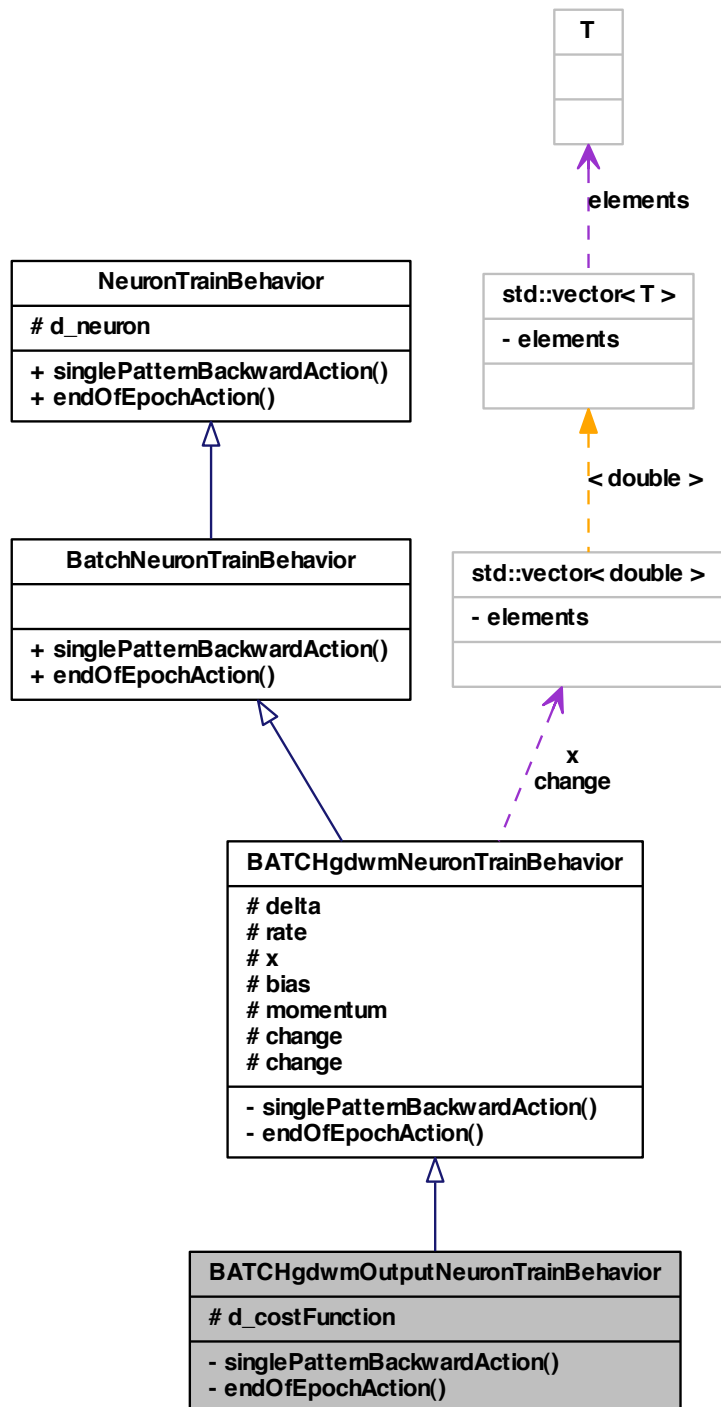
```
#include <BATCHgdwmOutputNeuronTrainBehavior.h>
```

Inheritance diagram for BATCHgdwmOutputNeuronTrainBehavior:





Collaboration diagram for BATCHgdwmOutputNeuronTrainBehavior:



### Protected Attributes

- CostFunctionWeakPtr [d\\_costFunction](#)

### Private Member Functions

- void [singlePatternBackwardAction](#) ()
- void [endOfEpochAction](#) ()

#### 5.21.1 Detailed Description

class [BATCHgdwmOutputNeuronTrainBehavior](#) -

Definition at line 5 of file BATCHgdwmOutputNeuronTrainBehavior.h.

#### 5.21.2 Member Function Documentation

5.21.2.1 void [BATCHgdwmOutputNeuronTrainBehavior::endOfEpochAction](#) ( )  
[private, virtual]

Implements [BATCHgdwmNeuronTrainBehavior](#).

5.21.2.2 void [BATCHgdwmOutputNeuronTrainBehavior::singlePatternBackwardAction](#) ( )  
[private, virtual]

Implements [BATCHgdwmNeuronTrainBehavior](#).

#### 5.21.3 Member Data Documentation

5.21.3.1 CostFunctionWeakPtr [BATCHgdwmOutputNeuronTrainBehavior::d\\_costFunction](#) [protected]

Definition at line 8 of file BATCHgdwmOutputNeuronTrainBehavior.h.

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

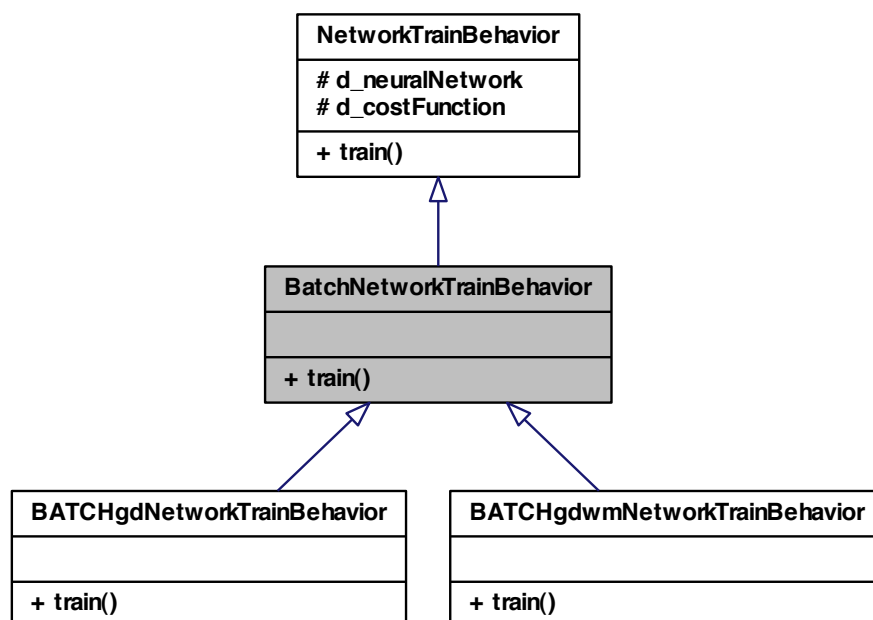
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.22 BatchNetworkTrainBehavior Class Reference

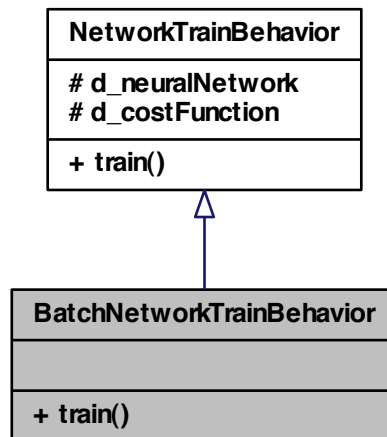
class [BatchNetworkTrainBehavior](#) -

```
#include <BatchNetworkTrainBehavior.h>
```

Inheritance diagram for BatchNetworkTrainBehavior:



Collaboration diagram for BatchNetworkTrainBehavior:



## Public Member Functions

- virtual `Rcpp::List` [train](#) (`Rcpp::List` parameterList)=0

### 5.22.1 Detailed Description

class [BatchNetworkTrainBehavior](#) -

Definition at line 5 of file `BatchNetworkTrainBehavior.h`.

### 5.22.2 Member Function Documentation

5.22.2.1 `virtual Rcpp::List BatchNetworkTrainBehavior::train ( Rcpp::List parameterList )`  
`[pure virtual]`

Implements [NetworkTrainBehavior](#).

Implemented in [BATCHgdNetworkTrainBehavior](#), and [BATCHgdwmNetworkTrainBehavior](#).

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

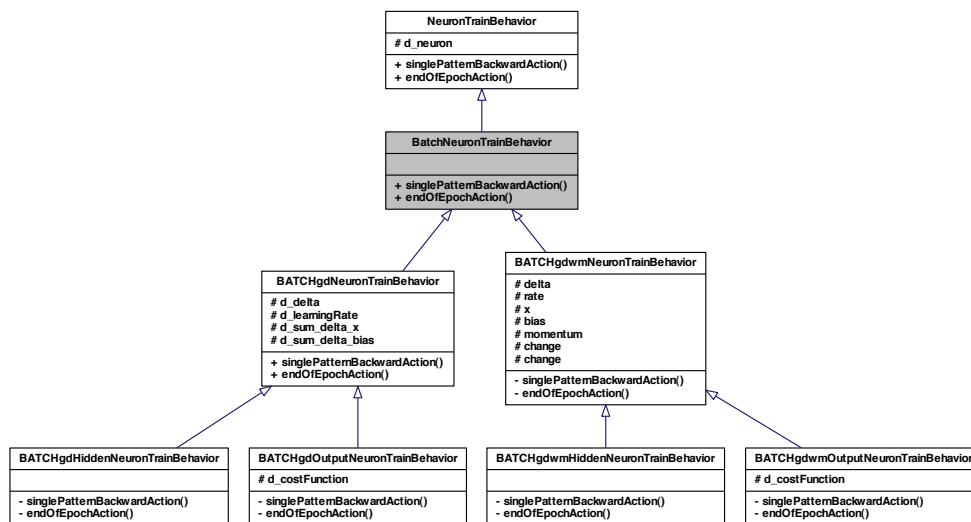
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeader.h`

## 5.23 BatchNeuronTrainBehavior Class Reference

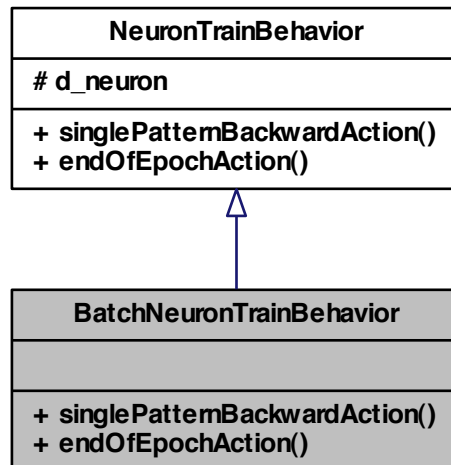
class [BatchNeuronTrainBehavior](#) -

```
#include <BatchNeuronTrainBehavior.h>
```

Inheritance diagram for BatchNeuronTrainBehavior:



Collaboration diagram for BatchNeuronTrainBehavior:



## Public Member Functions

- virtual void [singlePatternBackwardAction](#) ()=0
- virtual void [endOfEpochAction](#) ()=0

### 5.23.1 Detailed Description

class [BatchNeuronTrainBehavior](#) -

Definition at line 5 of file BatchNeuronTrainBehavior.h.

### 5.23.2 Member Function Documentation

5.23.2.1 virtual void [BatchNeuronTrainBehavior::endOfEpochAction](#) ( ) [pure virtual]

Implements [NeuronTrainBehavior](#).

Implemented in [BATCHgdHiddenNeuronTrainBehavior](#), [BATCHgdNeuronTrainBehavior](#), [BATCHgdOutputNeuronTrainBehavior](#), [BATCHgdwmHiddenNeuronTrainBehavior](#), [BATCHgdwmNeuronTrainBehavior](#), and [BATCHgdwmOutputNeuronTrainBehavior](#).

5.23.2.2 `virtual void BatchNeuronTrainBehavior::singlePatternBackwardAction ( ) [pure virtual]`

Implements [NeuronTrainBehavior](#).

Implemented in [BATCHgdHiddenNeuronTrainBehavior](#), [BATCHgdNeuronTrainBehavior](#), [BATCHgdOutputNeuronTrainBehavior](#), [BATCHgdwmHiddenNeuronTrainBehavior](#), [BATCHgdwmNeuronTrainBehavior](#), and [BATCHgdwmOutputNeuronTrainBehavior](#).

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

- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BatchNeu`

## 5.24 Con Class Reference

class [Con](#) -

```
#include <Connection.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](#)

### 5.24.1 Detailed Description

class [Con](#) -

Definition at line 3 of file Connection.h.

### 5.24.2 Constructor & Destructor Documentation

#### 5.24.2.1 `Con::Con ( Neuron & neuron )`

Constructor.

Definition at line 20 of file Connection.cpp.

```

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

```

#### 5.24.2.2 `Con::Con ( Neuron & neuron, double weight )`

Constructor.

Definition at line 31 of file Connection.cpp.

```

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

```

### 5.24.3 Member Function Documentation

#### 5.24.3.1 `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 57 of file `Connection.cpp`.

References `d_neuron`.

```
{
    return d_neuron;
}
```

**5.24.3.2 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.

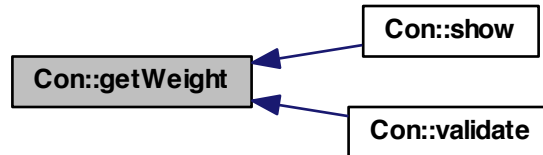
Definition at line 117 of file `Connection.cpp`.

References `d_weight`.

Referenced by `show()`, and `validate()`.

```
{
    return d_weight;
}
```

Here is the caller graph for this function:



#### 5.24.3.3 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 89 of file `Connection.cpp`.

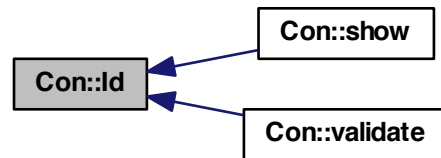
References `d_neuron`.

Referenced by `show()`, and `validate()`.

```

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

Here is the caller graph for this function:



#### 5.24.3.4 void Con::setNeuron ( Neuron & neuron )

Definition at line 64 of file Connection.cpp.

References `d_neuron`.

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

#### 5.24.3.5 void Con::setWeight ( double weight )

Definition at line 124 of file Connection.cpp.

References `d_weight`.

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

#### 5.24.3.6 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

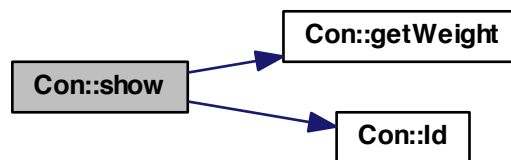
[setWeight](#) and the unit test files, e.g., `runit.Cpp.Con.R`, for usage examples.

Definition at line 136 of file Connection.cpp.

References `getWeight()`, and `Id()`.

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

Here is the call graph for this function:



#### 5.24.3.7 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

<i>An</i> std::range error if weight or from are not finite.
--

Definition at line 156 of file Connection.cpp.

References `getWeight()`, and `Id()`.

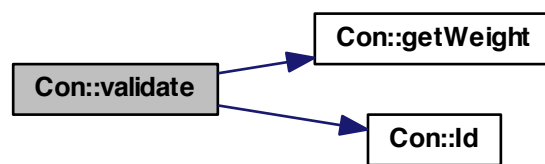
```
{
```

```

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:



#### 5.24.4 Member Data Documentation

##### 5.24.4.1 NeuronRef Con::d\_neuron [private]

Definition at line 6 of file Connection.h.

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

##### 5.24.4.2 double Con::d\_weight [private]

Definition at line 7 of file Connection.h.

Referenced by getWeight(), and setWeight().

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

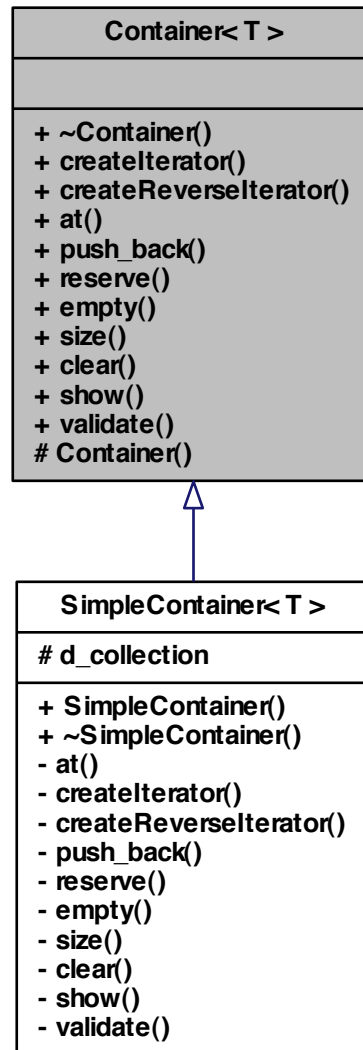
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Connection.h
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Connection.cpp

## 5.25 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 boost::shared\_ptr< [Iterator](#)< T > > [createReverseIterator](#) ()=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](#) ()

#### 5.25.1 Detailed Description

template<typename T>class Container< T >

class [Container](#) -

Definition at line 5 of file Container.h.

#### 5.25.2 Constructor & Destructor Documentation

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

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

#### 5.25.3 Member Function Documentation

5.25.3.1 template<typename T > virtual T Container< T >::at ( size\_type *element* )  
[pure virtual]

Implemented in [SimpleContainer< T >](#).

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

Implemented in [SimpleContainer< T >](#).

5.25.3.3 template<typename T > virtual boost::shared\_ptr< Iterator<T> > Container< T  
>::createIterator ( ) [pure virtual]

Implemented in [SimpleContainer< T >](#).

5.25.3.4 `template<typename T> virtual boost::shared_ptr< Iterator<T>> Container< T>::createReverseliterator ( )` [pure virtual]

Implemented in [SimpleContainer< T >](#).

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

Implemented in [SimpleContainer< T >](#).

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

Implemented in [SimpleContainer< T >](#).

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

Implemented in [SimpleContainer< T >](#).

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

Implemented in [SimpleContainer< T >](#).

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

Implemented in [SimpleContainer< T >](#).

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

Implemented in [SimpleContainer< T >](#).

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

- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

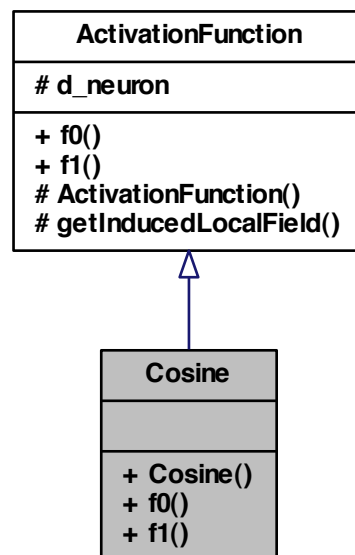
## 5.26 Cosine Class Reference

class [Cosine](#) -

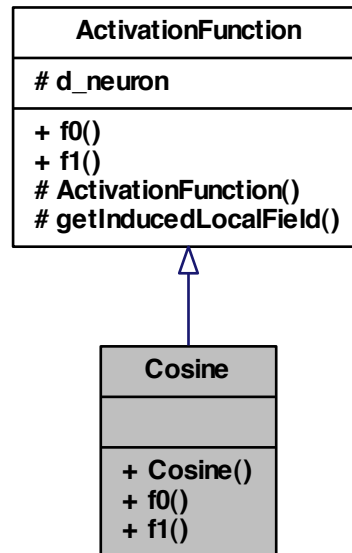
```
#include <Cosine.h>
```



Inheritance diagram for Cosine:



Collaboration diagram for Cosine:



### Public Member Functions

- [Cosine](#) ([NeuronPtr](#) neuronPtr)
- double [f0](#) ()
- double [f1](#) ()

### 5.26.1 Detailed Description

class [Cosine](#) -

Definition at line 5 of file Cosine.h.

### 5.26.2 Constructor & Destructor Documentation

5.26.2.1 [Cosine::Cosine](#) ( [NeuronPtr](#) neuronPtr )

### 5.26.3 Member Function Documentation

5.26.3.1 `double Cosine::f0 ( ) [virtual]`

Implements [ActivationFunction](#).

5.26.3.2 `double Cosine::f1 ( ) [virtual]`

Implements [ActivationFunction](#).

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

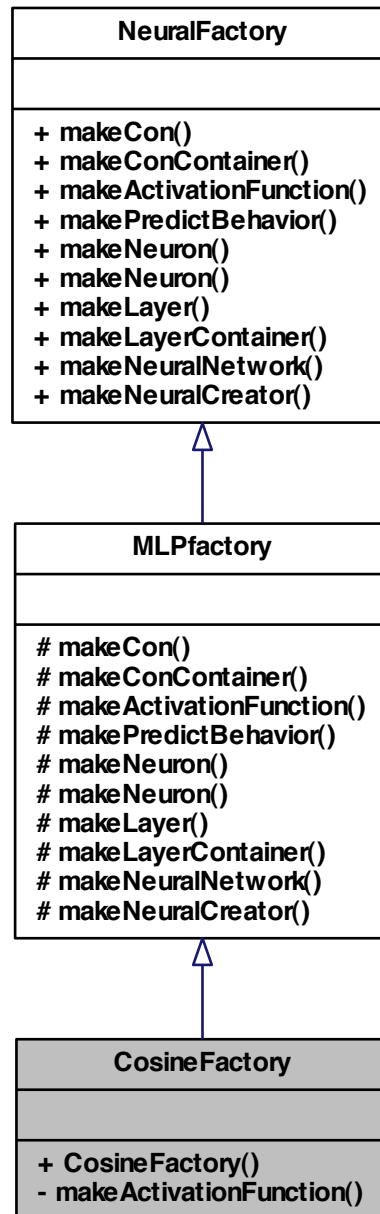
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Cosine.h`

## 5.27 CosineFactory Class Reference

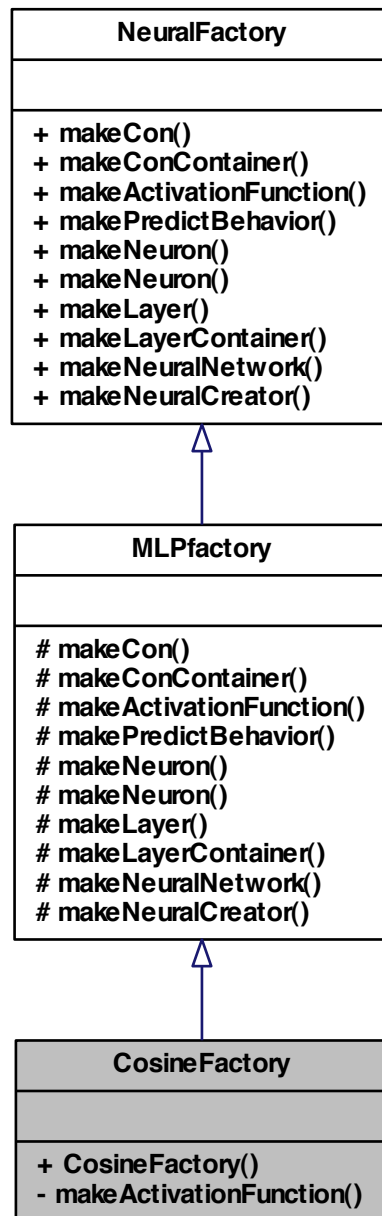
class [CosineFactory](#) -

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

Inheritance diagram for CosineFactory:



Collaboration diagram for CosineFactory:



## Public Member Functions

- [CosineFactory](#) ()

## Private Member Functions

- [ActivationFunctionPtr](#) [makeActivationFunction](#) ([NeuronPtr](#) neuronPtr)

### 5.27.1 Detailed Description

class [CosineFactory](#) -

Definition at line 5 of file CosineFactory.h.

### 5.27.2 Constructor & Destructor Documentation

#### 5.27.2.1 [CosineFactory::CosineFactory](#) ( )

### 5.27.3 Member Function Documentation

#### 5.27.3.1 [ActivationFunctionPtr](#) [CosineFactory::makeActivationFunction](#) ( [NeuronPtr](#) *neuronPtr* ) [private, virtual]

Implements [MLPfactory](#).

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

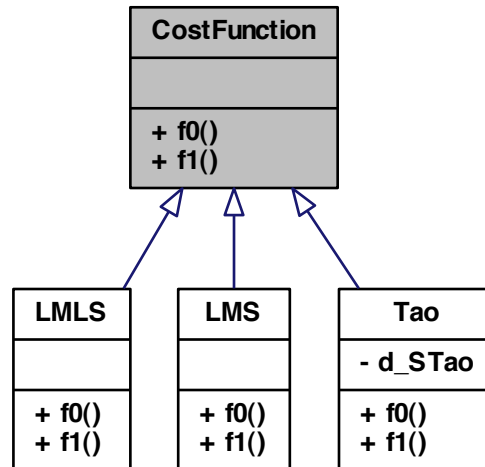
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.28 CostFunction Class Reference

class [CostFunction](#) -

```
#include <CostFunction.h>
```

Inheritance diagram for CostFunction:



### Public Member Functions

- virtual double [f0](#) (double output, double target)=0
- virtual double [f1](#) (double output, double target)=0

#### 5.28.1 Detailed Description

class [CostFunction](#) -

Definition at line 4 of file [CostFunction.h](#).

#### 5.28.2 Member Function Documentation

**5.28.2.1** virtual double [CostFunction::f0](#) ( double *output*, double *target* ) [pure virtual]

Implemented in [LMLS](#), [LMS](#), and [Tao](#).

**5.28.2.2** virtual double [CostFunction::f1](#) ( double *output*, double *target* ) [pure virtual]

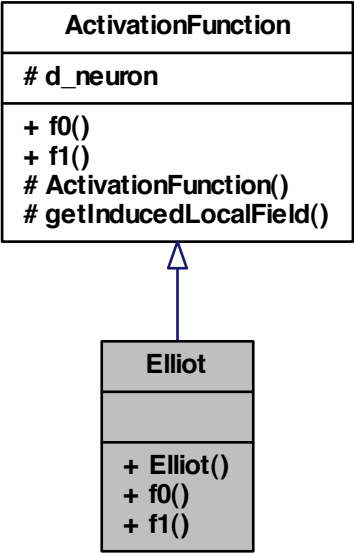
Implemented in [LMLS](#), [LMS](#), and [Tao](#).

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

- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

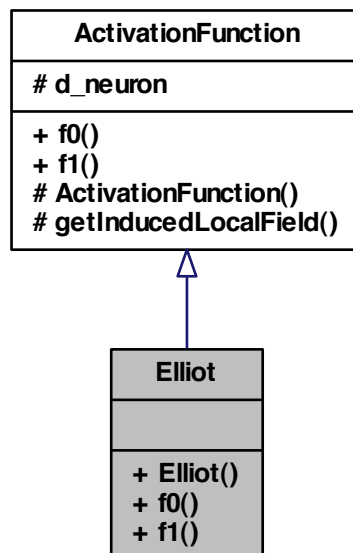
5.29 Elliot Class Reference

```
class Elliot -  
#include <Elliot.h>  
Inheritance diagram for Elliot:
```





Collaboration diagram for Elliot:



### Public Member Functions

- [Elliot](#) ([NeuronPtr](#) neuronPtr)
- double [f0](#) ()
- double [f1](#) ()

### 5.29.1 Detailed Description

class [Elliot](#) -

Definition at line 5 of file Elliot.h.

### 5.29.2 Constructor & Destructor Documentation

5.29.2.1 [Elliot::Elliot](#) ( [NeuronPtr](#) neuronPtr )

### 5.29.3 Member Function Documentation

5.29.3.1 `double Elliot::f0 ( ) [virtual]`

Implements [ActivationFunction](#).

5.29.3.2 `double Elliot::f1 ( ) [virtual]`

Implements [ActivationFunction](#).

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

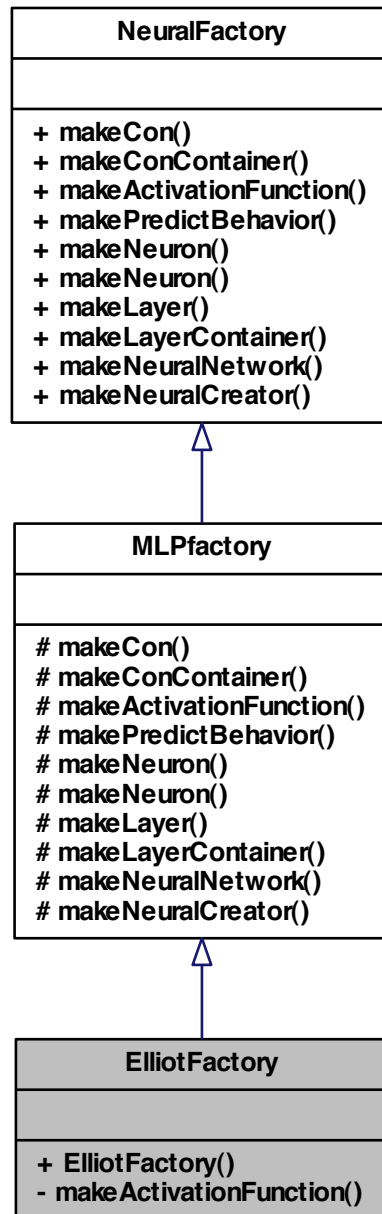
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.30 ElliotFactory Class Reference

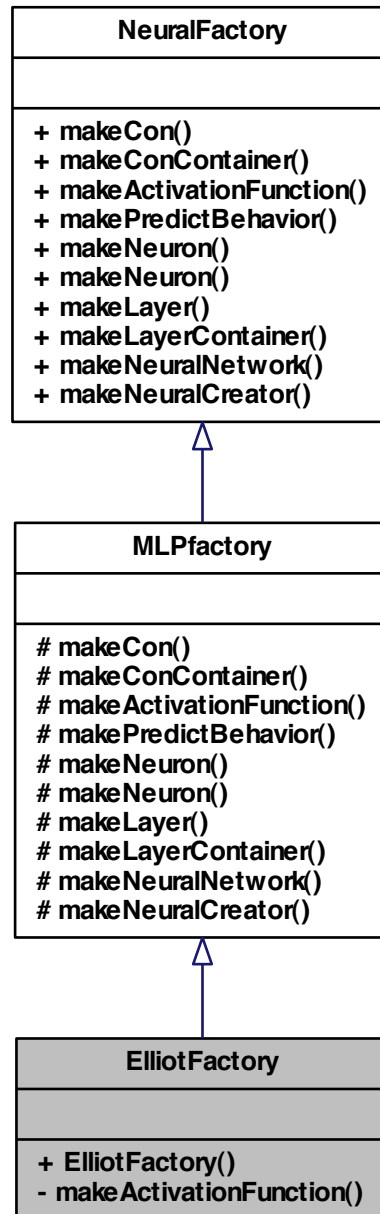
class [ElliotFactory](#) -

```
#include <ElliotFactory.h>
```

Inheritance diagram for ElliotFactory:



Collaboration diagram for ElliotFactory:



## Public Member Functions

- [ElliotFactory](#) ()

## Private Member Functions

- [ActivationFunctionPtr](#) [makeActivationFunction](#) ([NeuronPtr](#) neuronPtr)

### 5.30.1 Detailed Description

class [ElliotFactory](#) -

Definition at line 5 of file [ElliotFactory.h](#).

### 5.30.2 Constructor & Destructor Documentation

5.30.2.1 [ElliotFactory::ElliotFactory](#) ( )

### 5.30.3 Member Function Documentation

5.30.3.1 [ActivationFunctionPtr](#) [ElliotFactory::makeActivationFunction](#) ( [NeuronPtr](#) *neuronPtr* ) [[private](#), [virtual](#)]

Implements [MLPfactory](#).

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

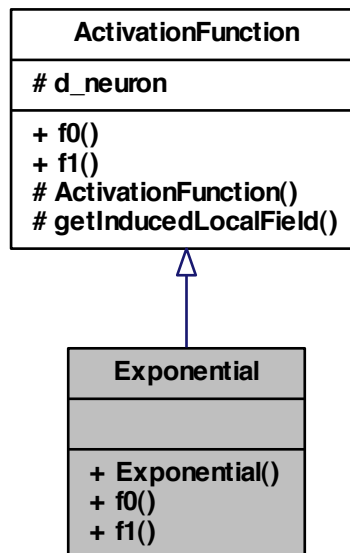
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/\[ElliotFactory.h\]\(#\)](#)

## 5.31 Exponential Class Reference

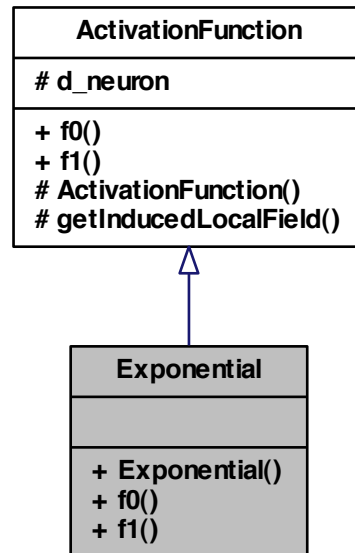
class [Exponential](#) -

```
#include <Exponential.h>
```

Inheritance diagram for Exponential:



Collaboration diagram for Exponential:



### Public Member Functions

- [Exponential](#) ([NeuronPtr](#) neuronPtr)
- double [f0](#) ()
- double [f1](#) ()

#### 5.31.1 Detailed Description

class [Exponential](#) -

Definition at line 5 of file `Exponential.h`.

#### 5.31.2 Constructor & Destructor Documentation

5.31.2.1 `Exponential::Exponential ( NeuronPtr neuronPtr )`

#### 5.31.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

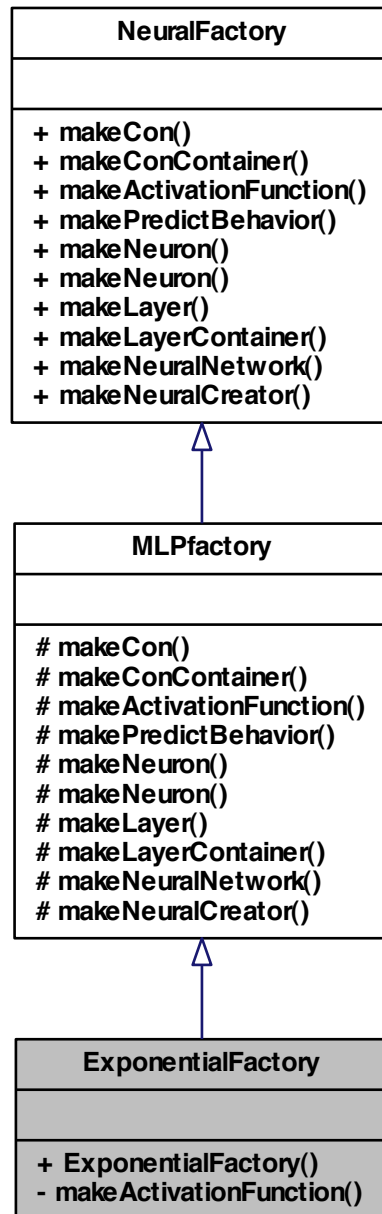
## 5.32 ExponentialFactory Class Reference

class [ExponentialFactory](#) -

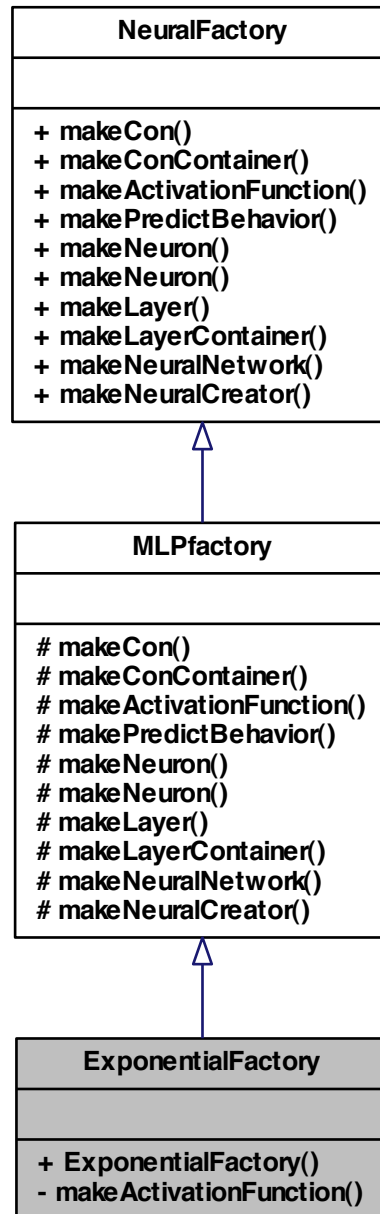
```
#include <ExponentialFactory.h>
```



Inheritance diagram for ExponentialFactory:



Collaboration diagram for ExponentialFactory:



## Public Member Functions

- [ExponentialFactory](#) ()

## Private Member Functions

- [ActivationFunctionPtr](#) [makeActivationFunction](#) ([NeuronPtr](#) neuronPtr)

### 5.32.1 Detailed Description

class [ExponentialFactory](#) -

Definition at line 5 of file [ExponentialFactory.h](#).

### 5.32.2 Constructor & Destructor Documentation

5.32.2.1 [ExponentialFactory::ExponentialFactory](#) ( )

### 5.32.3 Member Function Documentation

5.32.3.1 [ActivationFunctionPtr](#) [ExponentialFactory::makeActivationFunction](#) ( [NeuronPtr](#) *neuronPtr* ) [[private](#), [virtual](#)]

Implements [MLPfactory](#).

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

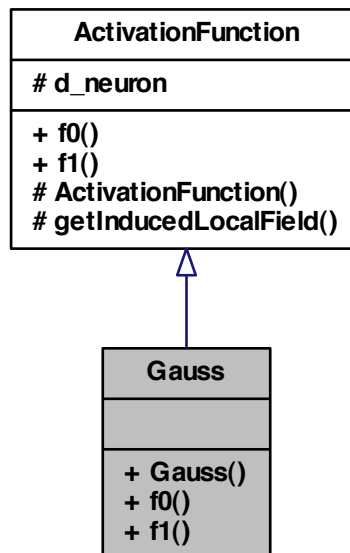
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Exponent](#)

## 5.33 Gauss Class Reference

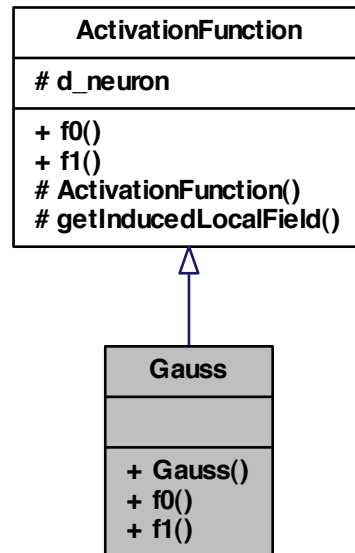
class [Gauss](#) -

```
#include <Gauss.h>
```

Inheritance diagram for Gauss:



Collaboration diagram for Gauss:



### Public Member Functions

- [Gauss](#) ([NeuronPtr](#) neuronPtr)
- double [f0](#) ()
- double [f1](#) ()

#### 5.33.1 Detailed Description

class [Gauss](#) -

Definition at line 5 of file Gauss.h.

#### 5.33.2 Constructor & Destructor Documentation

5.33.2.1 [Gauss::Gauss](#) ( [NeuronPtr](#) neuronPtr )

#### 5.33.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

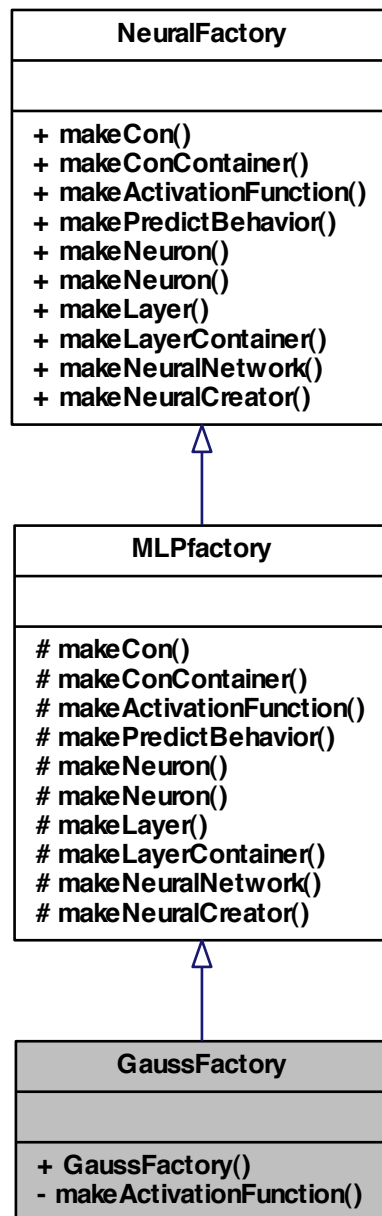
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.34 GaussFactory Class Reference

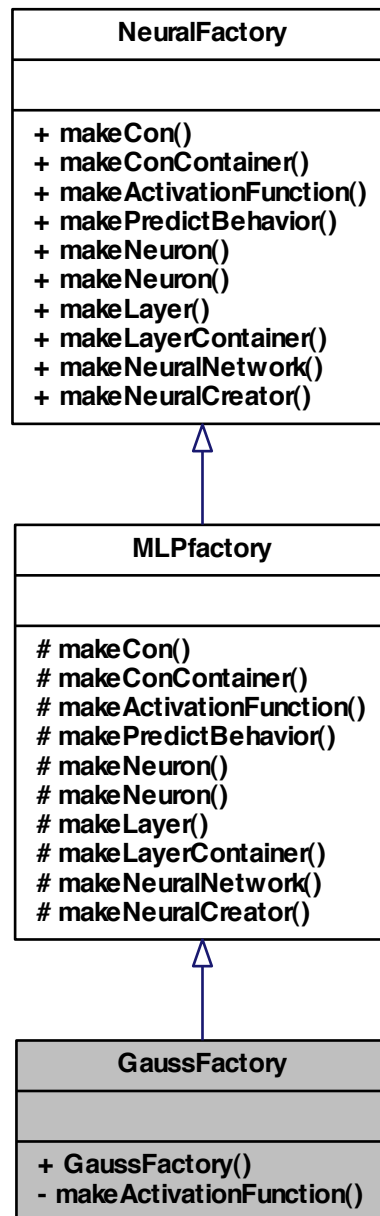
class [GaussFactory](#) -

```
#include <GaussFactory.h>
```

Inheritance diagram for GaussFactory:



Collaboration diagram for GaussFactory:





## Public Member Functions

- [GaussFactory](#) ()

## Private Member Functions

- [ActivationFunctionPtr](#) [makeActivationFunction](#) ([NeuronPtr](#) neuronPtr)

### 5.34.1 Detailed Description

class [GaussFactory](#) -

Definition at line 5 of file GaussFactory.h.

### 5.34.2 Constructor & Destructor Documentation

5.34.2.1 [GaussFactory::GaussFactory](#) ( )

### 5.34.3 Member Function Documentation

5.34.3.1 [ActivationFunctionPtr](#) [GaussFactory::makeActivationFunction](#) ( [NeuronPtr](#) *neuronPtr* ) [*private*, *virtual*]

Implements [MLPfactory](#).

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

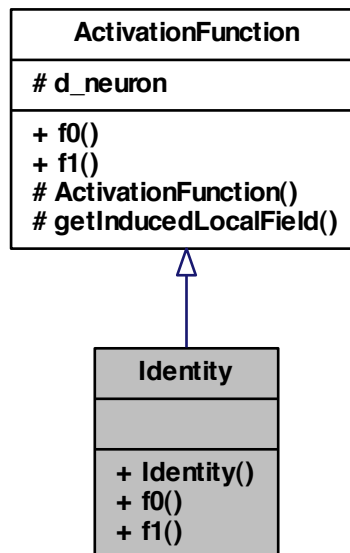
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[GaussFactory.h](#)

## 5.35 Identity Class Reference

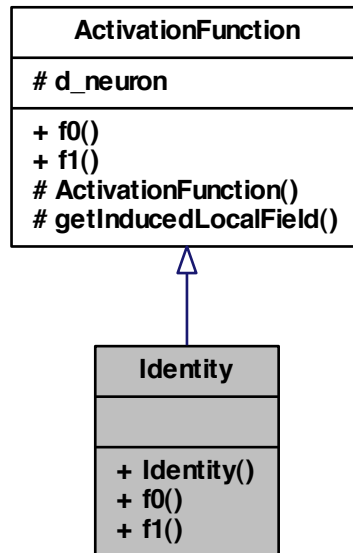
class [Identity](#) -

```
#include <Identity.h>
```

Inheritance diagram for Identity:



Collaboration diagram for Identity:



### Public Member Functions

- [Identity](#) ([NeuronPtr](#) neuronPtr)
- double [f0](#) ()
- double [f1](#) ()

#### 5.35.1 Detailed Description

class [Identity](#) -

Definition at line 5 of file Identity.h.

#### 5.35.2 Constructor & Destructor Documentation

##### 5.35.2.1 Identity::Identity ( [NeuronPtr](#) neuronPtr )

Definition at line 13 of file Identity.cpp.

```

: ActivationFunction(neuronPtr) {

```

```
}
```

### 5.35.3 Member Function Documentation

#### 5.35.3.1 `double Identity::f0 ( ) [virtual]`

Implements [ActivationFunction](#).

Definition at line 17 of file Identity.cpp.

References [ActivationFunction::getInducedLocalField\(\)](#).

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

Here is the call graph for this function:



#### 5.35.3.2 `double Identity::f1 ( ) [virtual]`

Implements [ActivationFunction](#).

Definition at line 21 of file Identity.cpp.

```
    {  
    return 1 ;  
    }
```

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

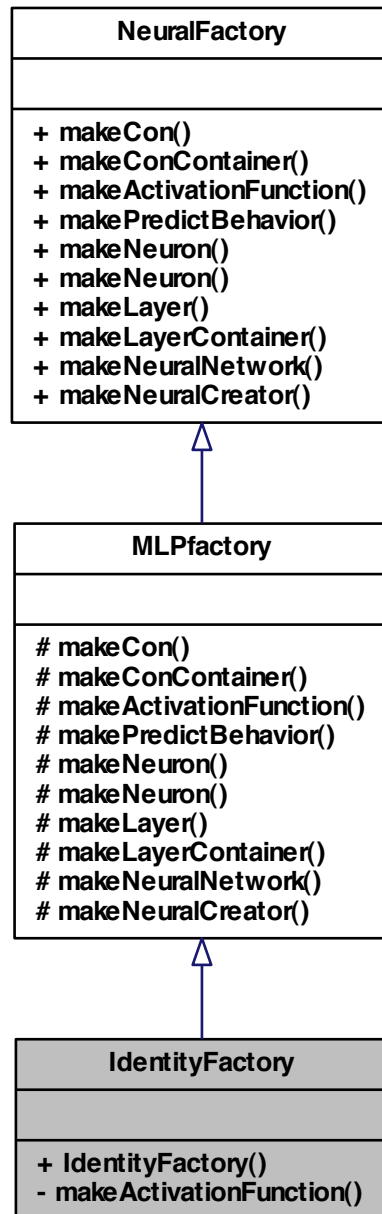
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead`
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Identity.cpp`

## 5.36 IdentityFactory Class Reference

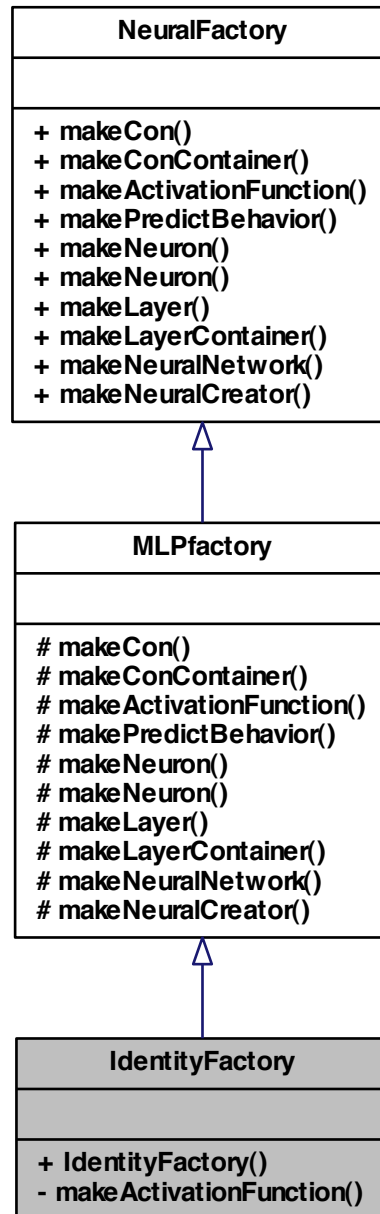
class [IdentityFactory](#) -

```
#include <IdentityFactory.h>
```

Inheritance diagram for IdentityFactory:



Collaboration diagram for IdentityFactory:



### Public Member Functions

- [IdentityFactory](#) ()

### Private Member Functions

- [ActivationFunctionPtr](#) [makeActivationFunction](#) ([NeuronPtr](#) neuronPtr)

#### 5.36.1 Detailed Description

class [IdentityFactory](#) -

Definition at line 5 of file [IdentityFactory.h](#).

#### 5.36.2 Constructor & Destructor Documentation

##### 5.36.2.1 `IdentityFactory::IdentityFactory ( )`

Definition at line 14 of file [IdentityFactory.cpp](#).

```
{  
}
```

#### 5.36.3 Member Function Documentation

##### 5.36.3.1 `ActivationFunctionPtr IdentityFactory::makeActivationFunction ( NeuronPtr neuronPtr )` [`private`, `virtual`]

Implements [MLPfactory](#).

Definition at line 20 of file [IdentityFactory.cpp](#).

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

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

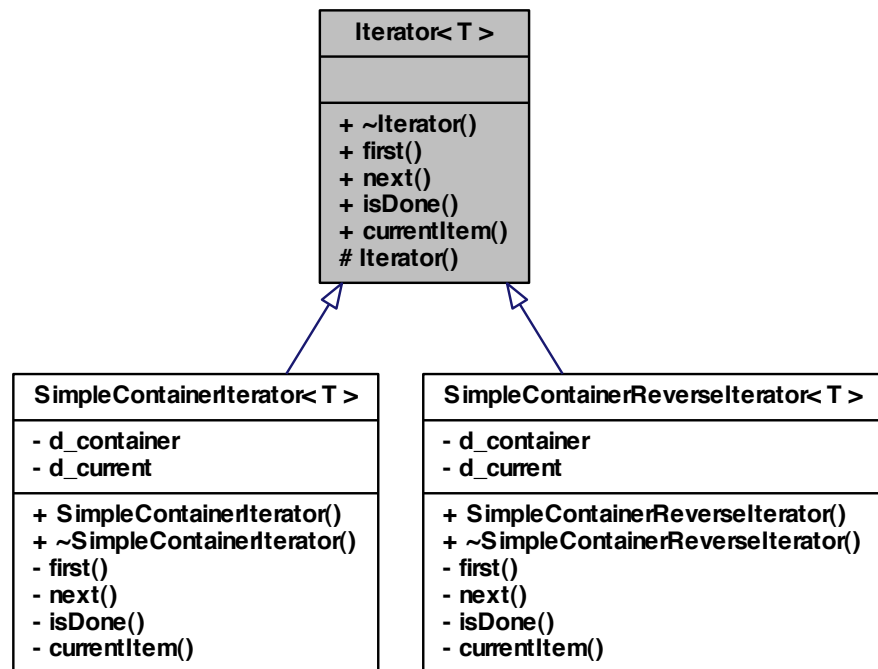
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/IdentityFa](#)
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/IdentityFactory.cpp](#)

## 5.37 `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()`

### 5.37.1 Detailed Description



```
template<typename T>class Iterator< T >
```

class [Iterator](#) -

Definition at line 5 of file Iterator.h.

### 5.37.2 Constructor & Destructor Documentation

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

5.37.2.2 `template<typename T> Iterator< T >::~Iterator ( )` [protected]

### 5.37.3 Member Function Documentation

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

Implemented in [SimpleContainerIterator< T >](#), and [SimpleContainerReverselIterator< T >](#).

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

Implemented in [SimpleContainerIterator< T >](#), and [SimpleContainerReverselIterator< T >](#).

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

Implemented in [SimpleContainerIterator< T >](#), and [SimpleContainerReverselIterator< T >](#).

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

Implemented in [SimpleContainerIterator< T >](#), and [SimpleContainerReverselIterator< T >](#).

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

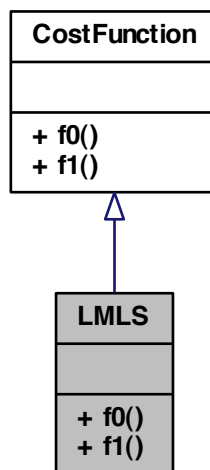
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Iterator.h`

## 5.38 LMLS Class Reference

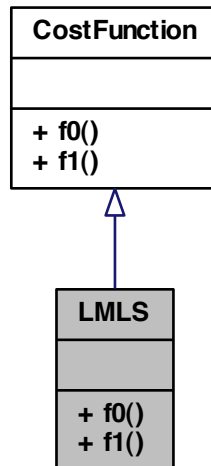
class [LMLS](#) -

```
#include <LMLS.h>
```

Inheritance diagram for LMLS:



Collaboration diagram for LMLS:



### Public Member Functions

- double [f0](#) (double output, double target)
- double [f1](#) (double output, double target)

### 5.38.1 Detailed Description

class [LMLS](#) -

Definition at line 5 of file LMLS.h.

### 5.38.2 Member Function Documentation

5.38.2.1 double [LMLS::f0](#) ( double *output*, double *target* ) `[virtual]`

Implements [CostFunction](#).

5.38.2.2 double [LMLS::f1](#) ( double *output*, double *target* ) `[virtual]`

Implements [CostFunction](#).

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

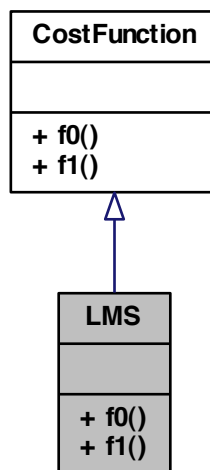
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

### 5.39 LMS Class Reference

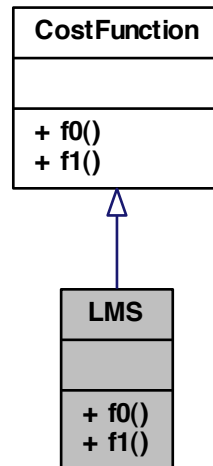
class [LMS](#) -

```
#include <LMS.h>
```

Inheritance diagram for LMS:



Collaboration diagram for LMS:



### Public Member Functions

- double `f0` (double output, double target)
- double `f1` (double output, double target)

#### 5.39.1 Detailed Description

class [LMS](#) -

Definition at line 5 of file `LMS.h`.

#### 5.39.2 Member Function Documentation

5.39.2.1 `double LMS::f0 ( double output, double target )` `[virtual]`

Implements [CostFunction](#).

5.39.2.2 `double LMS::f1 ( double output, double target )` `[virtual]`

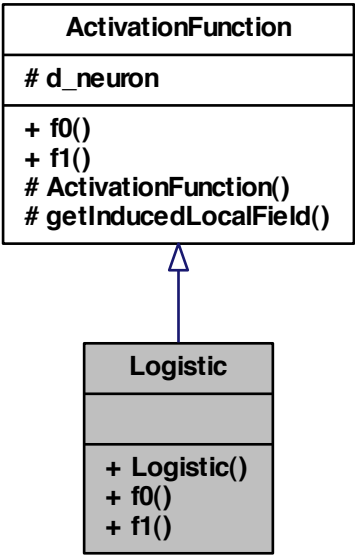
Implements [CostFunction](#).

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

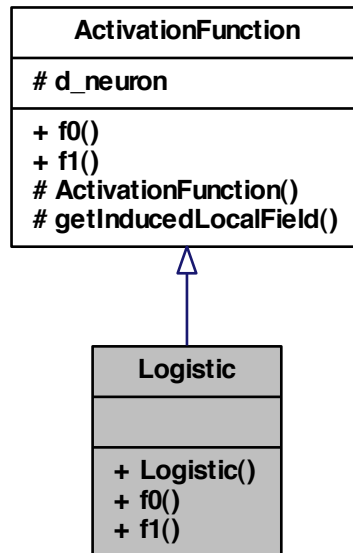
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

5.40 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](#) ()

#### 5.40.1 Detailed Description

class [Logistic](#) -

Definition at line 5 of file Logistic.h.

#### 5.40.2 Constructor & Destructor Documentation

5.40.2.1 [Logistic::Logistic](#) ( [NeuronPtr](#) neuronPtr )

#### 5.40.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead`

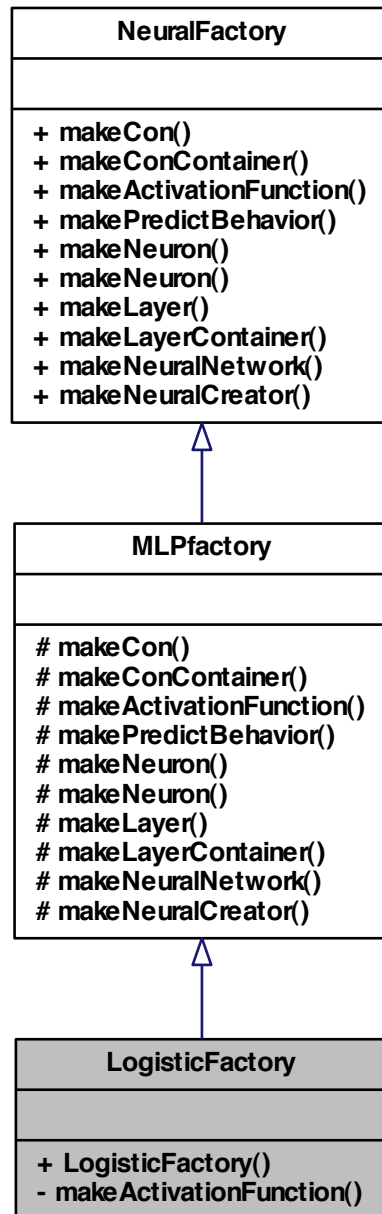
## 5.41 LogisticFactory Class Reference

class [LogisticFactory](#) -

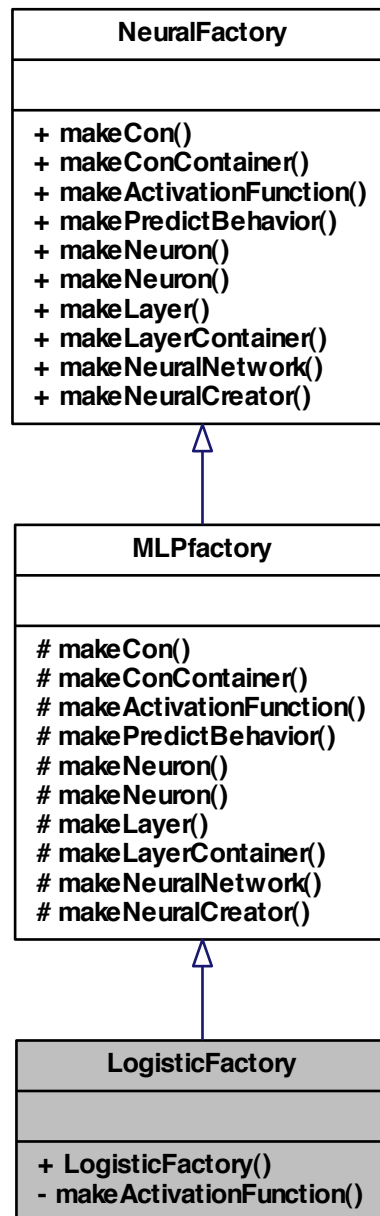
```
#include <LogisticFactory.h>
```



Inheritance diagram for LogisticFactory:



Collaboration diagram for LogisticFactory:



## Public Member Functions

- [LogisticFactory](#) ()

## Private Member Functions

- [ActivationFunctionPtr](#) [makeActivationFunction](#) ([NeuronPtr](#) neuronPtr)

### 5.41.1 Detailed Description

class [LogisticFactory](#) -

Definition at line 5 of file LogisticFactory.h.

### 5.41.2 Constructor & Destructor Documentation

5.41.2.1 [LogisticFactory::LogisticFactory](#) ( )

### 5.41.3 Member Function Documentation

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

Implements [MLPfactory](#).

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

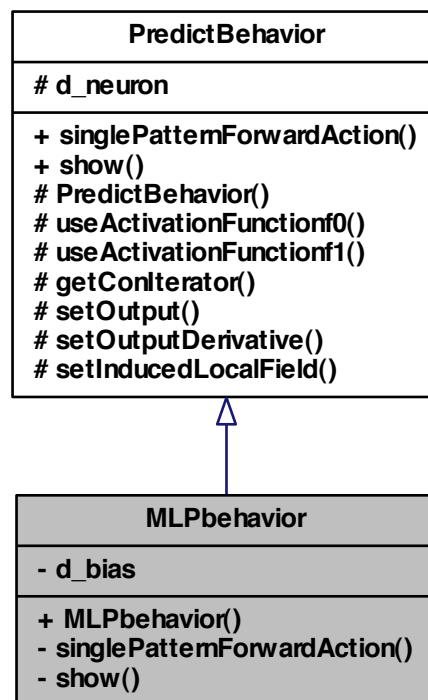
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[LogisticFactory.h](#)

## 5.42 MLPbehavior Class Reference

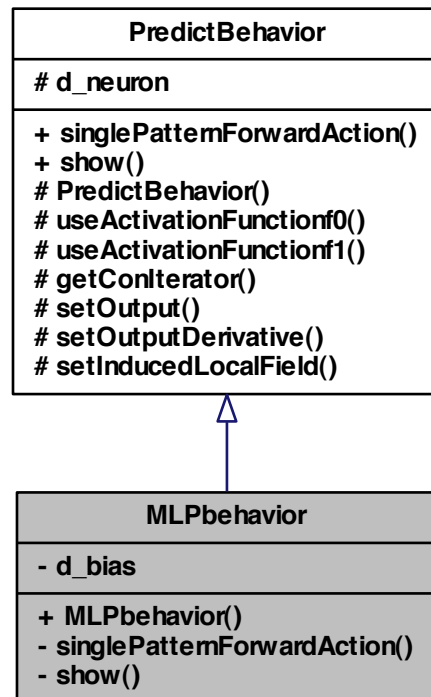
class [MLPbehavior](#) -

```
#include <MLPbehavior.h>
```

Inheritance diagram for MLPbehavior:



Collaboration diagram for MLPbehavior:



### Public Member Functions

- [MLPbehavior](#) ([NeuronPtr](#) neuronPtr)

### Private Member Functions

- void [singlePatternForwardAction](#) ()
- void [show](#) ()

### Private Attributes

- double [d\\_bias](#)

## Friends

- class [MLPfactory](#)

### 5.42.1 Detailed Description

class [MLPbehavior](#) -

Definition at line 5 of file MLPbehavior.h.

### 5.42.2 Constructor & Destructor Documentation

#### 5.42.2.1 MLPbehavior::MLPbehavior ( [NeuronPtr](#) *neuronPtr* )

Definition at line 17 of file MLPbehavior.cpp.

```

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

### 5.42.3 Member Function Documentation

#### 5.42.3.1 void MLPbehavior::show ( ) [private, virtual]

Implements [PredictBehavior](#).

Definition at line 42 of file MLPbehavior.cpp.

References [d\\_bias](#).

```

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

#### 5.42.3.2 void MLPbehavior::singlePatternForwardAction ( ) [private, virtual]

Implements [PredictBehavior](#).

Definition at line 23 of file MLPbehavior.cpp.

References [d\\_bias](#), [PredictBehavior::getConIterator\(\)](#), [PredictBehavior::setInducedLocalField\(\)](#), [PredictBehavior::setOutput\(\)](#), [PredictBehavior::setOutputDerivative\(\)](#), [PredictBehavior::useActivationFunction0\(\)](#) and [PredictBehavior::useActivationFunction1\(\)](#).

```

{

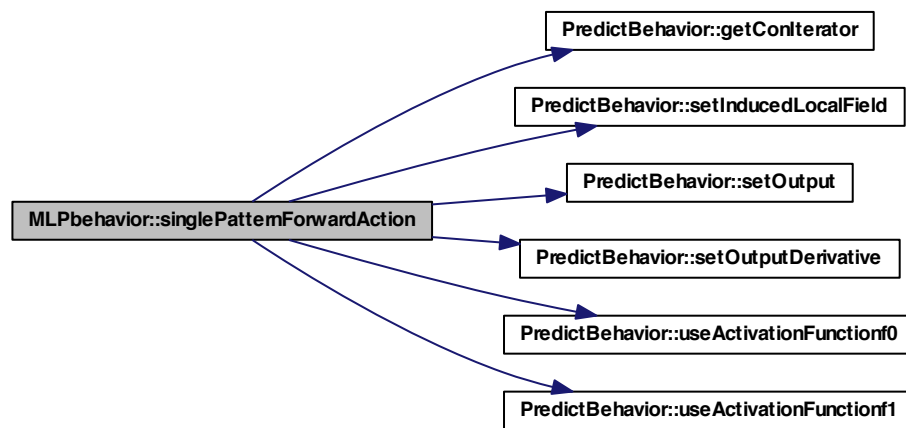
    double accumulator(d_bias);
    ConIteratorPtr conIterator = getConIterator();
```

```

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 (useActivationFunction0());
setOutputDerivative (useActivationFunction1());
}

```

Here is the call graph for this function:



## 5.42.4 Friends And Related Function Documentation

### 5.42.4.1 friend class MLPfactory [friend]

Definition at line 11 of file MLPbehavior.h.

## 5.42.5 Member Data Documentation

### 5.42.5.1 double MLPbehavior::d\_bias [private]

Definition at line 8 of file MLPbehavior.h.

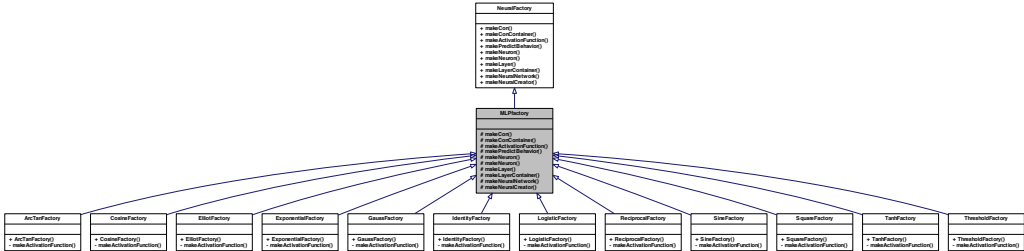
Referenced by `MLPfactory::makeNeuron()`, `show()`, and `singlePatternForwardAction()`.

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

- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPbehavi

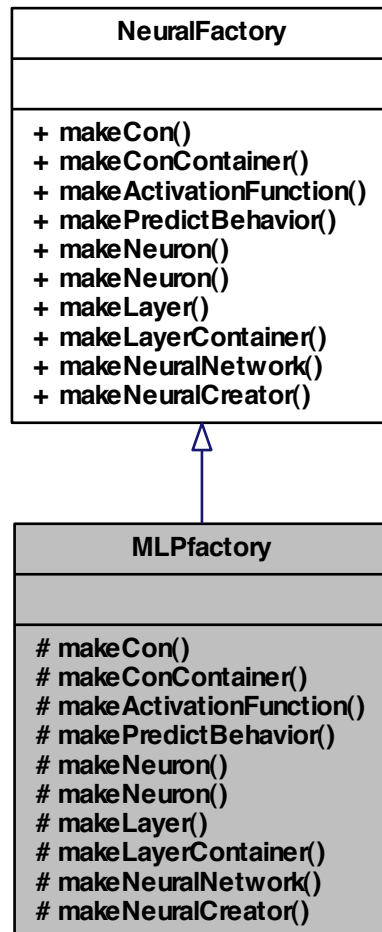
5.43 MLPfactory Class Reference

class [MLPfactory](#) -  
`#include <MLPfactory.h>`  
Inheritance diagram for MLPfactory:





Collaboration diagram for 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)

- [LayerPtr makeLayer \(\)](#)
- [LayerContainerPtr makeLayerContainer \(\)](#)
- [NeuralNetworkPtr makeNeuralNetwork \(NeuralFactory &neuralFactory\)](#)
- [NeuralCreatorPtr makeNeuralCreator \(\)](#)

### 5.43.1 Detailed Description

class [MLPfactory](#) -

Definition at line 5 of file MLPfactory.h.

### 5.43.2 Member Function Documentation

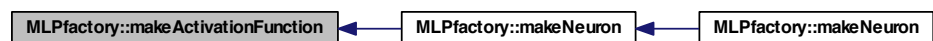
**5.43.2.1** `virtual ActivationFunctionPtr MLPfactory::makeActivationFunction ( NeuronPtr neuronPtr )` [protected, pure virtual]

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:



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

Implements [NeuralFactory](#).

Definition at line 30 of file MLPfactory.cpp.

Referenced by [makeNeuron\(\)](#).

```

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

Here is the caller graph for this function:



**5.43.2.3** `ConContainerPtr MLPfactory::makeConContainer( )` [protected, virtual]

Implements [NeuralFactory](#).

Definition at line 37 of file `MLPfactory.cpp`.

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

**5.43.2.4** `LayerPtr MLPfactory::makeLayer( )` [protected, virtual]

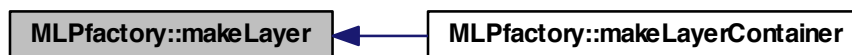
Implements [NeuralFactory](#).

Definition at line 84 of file `MLPfactory.cpp`.

Referenced by `makeLayerContainer()`.

```
{  
    LayerPtr layerPtr( new SimpleContainer<NeuronPtr> );  
    return layerPtr;  
}
```

Here is the caller graph for this function:



#### 5.43.2.5 **LayerContainerPtr** MLPfactory::makeLayerContainer ( ) [protected, virtual]

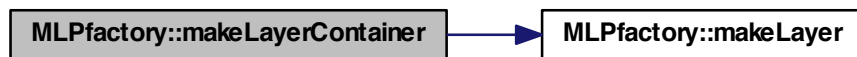
Implements [NeuralFactory](#).

Definition at line 92 of file MLPfactory.cpp.

References [makeLayer\(\)](#).

```
{
    LayerContainerPtr layerContainerPtr( new SimpleContainer<LayerPtr> );
    layerContainerPtr->push_back( makeLayer() );
    return layerContainerPtr;
}
```

Here is the call graph for this function:



#### 5.43.2.6 **NeuralCreatorPtr** MLPfactory::makeNeuralCreator ( ) [protected, virtual]

Implements [NeuralFactory](#).

Definition at line 109 of file MLPfactory.cpp.

```
{
    NeuralCreatorPtr neuralCreatorPtr(new SimpleNeuralCreator );
    return neuralCreatorPtr;
}
```

#### 5.43.2.7 **NeuralNetworkPtr** MLPfactory::makeNeuralNetwork ( **NeuralFactory & neuralFactory** ) [protected, virtual]

Implements [NeuralFactory](#).

Definition at line 101 of file MLPfactory.cpp.

```
{
    NeuralNetworkPtr neuralNetworkPtr(new SimpleNetwork(neuralFactory ) );
    return neuralNetworkPtr;
}
```

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

Implements [NeuralFactory](#).

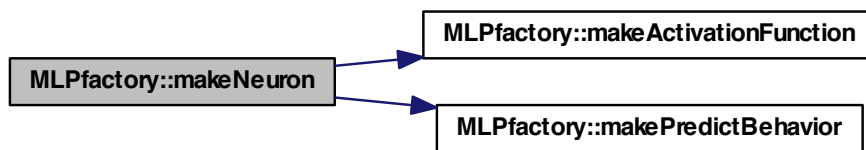
Definition at line 52 of file MLPfactory.cpp.

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:



Here is the caller graph for this function:



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

Implements [NeuralFactory](#).

Definition at line 62 of file MLPfactory.cpp.

References MLPbehavior::d\_bias, makeCon(), and makeNeuron().

```
{
    RNGScope scope;

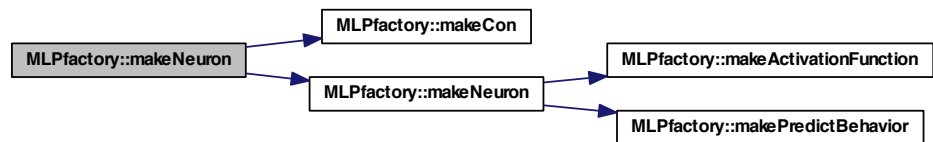
    NeuronPtr neuronPtr (makeNeuron (Id));

    double extreme = sqrt(3 / totalAmountOfParameters);
    double weight;
    for (neuronIteratorPtr->first(); !neuronIteratorPtr->isDone(); neuronIteratorPtr->next())
    {
        weight =as<double>(runif(1, -extreme, extreme));
        neuronPtr->addCon (makeCon (*neuronIteratorPtr->currentItem(), weight));
    }

    MLPbehavior* mlpBehavior = dynamic_cast<MLPbehavior*>(neuronPtr->d_predictBehavior.get());
    mlpBehavior->d_bias=as<double>(runif(1, -extreme, extreme));

    return neuronPtr;
}
```

Here is the call graph for this function:



#### 5.43.2.10 PredictBehaviorPtr MLPfactory::makePredictBehavior ( NeuronPtr neuronPtr ) [protected, virtual]

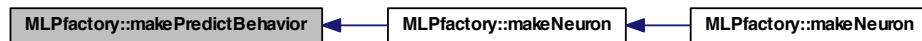
Implements [NeuralFactory](#).

Definition at line 45 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:

- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPfactory.h](#)
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPfactory.cpp](#)

## 5.44 NetworkRinterface Class Reference

class [NetworkRinterface](#) -

```
#include <NetworkRinterface.h>
```

### Public Member Functions

- [NetworkRinterface](#) ()
- void [createFeedForwardNetwork](#) (Rcpp::NumericVector numberOfNeurons)
- Rcpp::NumericMatrix [predict](#) (Rcpp::NumericMatrix numericMatrix)
- Rcpp::List [train](#) (Rcpp::List parameterList)
- size\_type [inputSize](#) ()
- size\_type [outputSize](#) ()
- void [show](#) ()
- bool [validate](#) ()

### Private Attributes

- [NeuralNetworkPtr](#) `d_neuralNetwork`

### 5.44.1 Detailed Description

class [NetworkRinterface](#) -

Definition at line 3 of file `NetworkRinterface.h`.

## 5.44.2 Constructor & Destructor Documentation

### 5.44.2.1 NetworkRinterface::NetworkRinterface ( )

Definition at line 22 of file NetworkRinterface.cpp.

```
{
}
```

## 5.44.3 Member Function Documentation

### 5.44.3.1 void NetworkRinterface::createFeedForwardNetwork ( Rcpp::NumericVector *numberOfNeurons* )

Definition at line 28 of file NetworkRinterface.cpp.

References `d_neuralNetwork`.

Referenced by `RCPP_MODULE()`.

```
{
    NeuralFactoryPtr hiddenLayersFactoryPtr(new TanhFactory());
    NeuralFactoryPtr outputFactoryPtr(new IdentityFactory());
    NeuralCreatorPtr neuralCreator(outputFactoryPtr->makeNeuralCreator());
    d_neuralNetwork = neuralCreator->createFeedForwardNetwork(
        as<std::vector<int>> > (numberOfNeurons), *hiddenLayersFactoryPtr,
        *outputFactoryPtr);
}
```

Here is the caller graph for this function:



### 5.44.3.2 size\_type NetworkRinterface::inputSize ( )

Definition at line 102 of file NetworkRinterface.cpp.

References `d_neuralNetwork`.

Referenced by `predict()`, and `RCPP_MODULE()`.

```
{
    return d_neuralNetwork->inputSize();
}
```



Here is the caller graph for this function:



#### 5.44.3.3 `size_type NetworkRinterface::outputSize ( )`

Definition at line 108 of file `NetworkRinterface.cpp`.

References `d_neuralNetwork`.

Referenced by `predict()`, and `RCPP_MODULE()`.

```

{
    return d_neuralNetwork->outputSize();
}

```

Here is the caller graph for this function:



#### 5.44.3.4 `Rcpp::NumericMatrix NetworkRinterface::predict ( Rcpp::NumericMatrix numericMatrix )`

Definition at line 39 of file `NetworkRinterface.cpp`.

References `d_neuralNetwork`, `inputSize()`, and `outputSize()`.

Referenced by `RCPP_MODULE()`.

```

{
    BEGIN_RCPP

    // VALIDATION

```

```

if (!d_neuralNetwork)
{
    throw std::runtime_error( "\nUninitialized network. Please use any of the c
    reate methods available.\n");
}

bool checkIncorrectNumberOfRows(
    inputSize() != static_cast<size_type> (numericMatrix.nrow()));
if (checkIncorrectNumberOfRows)
{
    throw std::runtime_error(
        "\nIncorrect number or rows. The number of input neurons must be equal
        to the number of rows of the input matrix.\n");
}

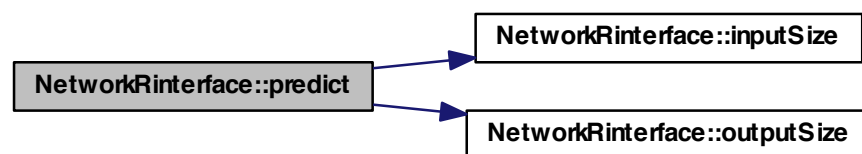
Rcpp::NumericMatrix outputMatrix(outputSize(), numericMatrix.ncol());
std::vector<double>::iterator inputIterator(numericMatrix.begin());
std::vector<double>::iterator outputIterator(outputMatrix.begin());

// PREDICT LOOP
for (int i = 0; i < numericMatrix.ncol(); i++)
{
    d_neuralNetwork->writeInput(inputIterator);
    d_neuralNetwork->singlePatternForwardAction();
    d_neuralNetwork->readOutput(outputIterator);
}
return outputMatrix;

END_RCPP}

```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.44.3.5 void NetworkRinterface::show ( )

Definition at line 114 of file NetworkRinterface.cpp.

References `d_neuralNetwork`.

Referenced by `RCPP_MODULE()`.

```
{  
    if (!d_neuralNetwork)  
    {  
        Rprintf(  
            "\nUninitialized network. Please use any of the create methods availabl  
e.\n");  
    }  
    else  
    {  
        d_neuralNetwork->show();  
    }  
}
```

Here is the caller graph for this function:



#### 5.44.3.6 Rcpp::List NetworkRinterface::train ( Rcpp::List *parameterList* )

Definition at line 77 of file NetworkRinterface.cpp.

References `d_neuralNetwork`.

```
{
    BEGIN_RCPP
        return d_neuralNetwork->train(parameterList);
    END_RCPP
}
```

#### 5.44.3.7 `bool NetworkRinterface::validate ( )`

Definition at line 129 of file `NetworkRinterface.cpp`.

References `d_neuralNetwork`.

Referenced by `RCPP_MODULE()`.

```
{
    BEGIN_RCPP if (d_neuralNetwork)
    {
        return d_neuralNetwork->validate();
    }
    else
    {
        throw std::runtime_error(
            "\nUninitialized network. Please use any of the create methods available.
            \n");
        return false;
    }
    END_RCPP
}
```

Here is the caller graph for this function:



### 5.44.4 Member Data Documentation

#### 5.44.4.1 `NeuralNetworkPtr NetworkRinterface::d_neuralNetwork` `[private]`

Definition at line 6 of file `NetworkRinterface.h`.

Referenced by `createFeedForwardNetwork()`, `inputSize()`, `outputSize()`, `predict()`, `show()`, `train()`, and `validate()`.

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

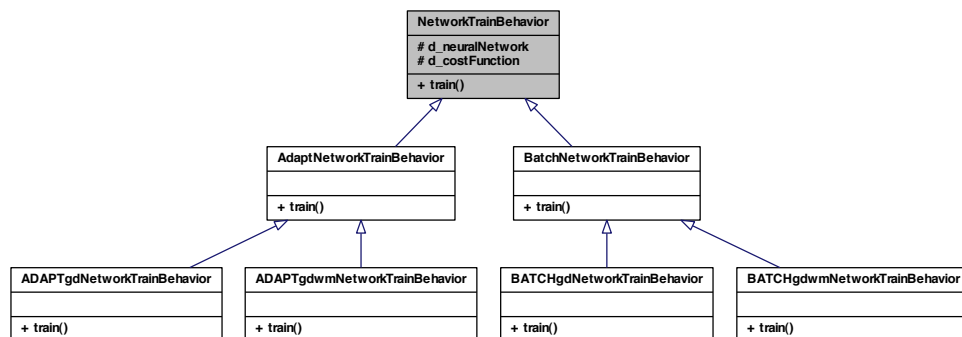
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkP](#)
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NetworkRinterface.cpp](#)

## 5.45 NetworkTrainBehavior Class Reference

class [NetworkTrainBehavior](#) -

```
#include <NetworkTrainBehavior.h>
```

Inheritance diagram for NetworkTrainBehavior:



### Public Member Functions

- virtual `Rcpp::List` [train](#) (`Rcpp::List` parameterList)=0

### Protected Attributes

- `NeuralNetworkWeakPtr` [d\\_neuralNetwork](#)
- `CostFunctionPtr` [d\\_costFunction](#)

#### 5.45.1 Detailed Description

class [NetworkTrainBehavior](#) -

Definition at line 4 of file `NetworkTrainBehavior.h`.

#### 5.45.2 Member Function Documentation

5.45.2.1 `virtual Rcpp::List NetworkTrainBehavior::train ( Rcpp::List parameterList )` [pure virtual]

Implemented in [ADAPTgdNetworkTrainBehavior](#), [ADAPTgdwmNetworkTrainBehavior](#), [AdaptNetworkTrainBehavior](#), [BATCHgdNetworkTrainBehavior](#), [BATCHgdwmNetworkTrainBehavior](#), and [BatchNetworkTrainBehavior](#).

### 5.45.3 Member Data Documentation

5.45.3.1 `CostFunctionPtr NetworkTrainBehavior::d_costFunction` [protected]

Definition at line 8 of file `NetworkTrainBehavior.h`.

5.45.3.2 `NeuralNetworkWeakPtr NetworkTrainBehavior::d_neuralNetwork` [protected]

Definition at line 7 of file `NetworkTrainBehavior.h`.

Referenced by `ADAPTgdNetworkTrainBehavior::train()`.

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

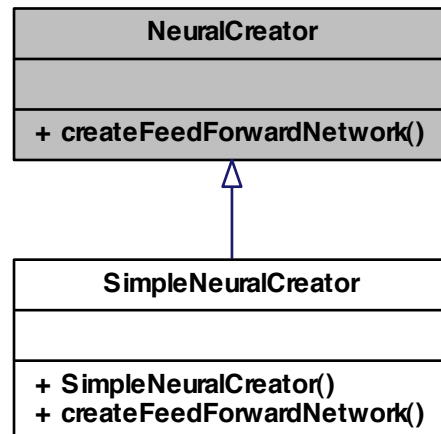
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead`

## 5.46 NeuralCreator Class Reference

class [NeuralCreator](#) -

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

Inheritance diagram for NeuralCreator:



### Public Member Functions

- virtual [NeuralNetworkPtr](#) `createFeedForwardNetwork` (std::vector< int > numberOfNeurons, [NeuralFactory](#) &hiddenLayersFactory, [NeuralFactory](#) &outputLayerFactory)=0

#### 5.46.1 Detailed Description

class [NeuralCreator](#) -

Definition at line 4 of file NeuralCreator.h.

#### 5.46.2 Member Function Documentation

- 5.46.2.1 virtual [NeuralNetworkPtr](#) `NeuralCreator::createFeedForwardNetwork` ( std::vector< int > *numberOfNeurons*, [NeuralFactory](#) & *hiddenLayersFactory*, [NeuralFactory](#) & *outputLayerFactory* ) [pure virtual]

Implemented in [SimpleNeuralCreator](#).

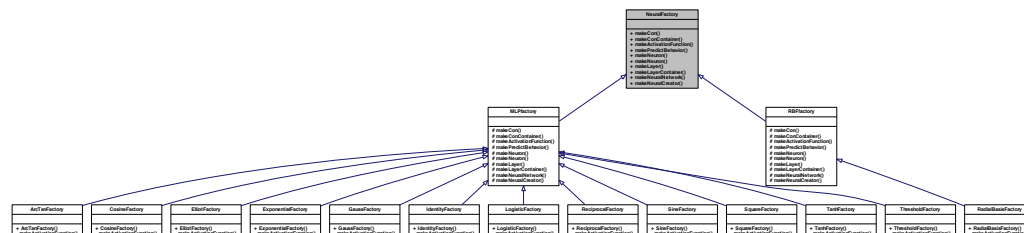
The documentation for this class was generated from the following file:

- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralCreator.h`

class NeuralFactory -

```
#include <NeuralFactory.h>
```

Inheritance diagram for NeuralFactory:



- 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 `LayerPtr makeLayer ()=0`
- virtual `LayerContainerPtr makeLayerContainer ()=0`
- virtual `NeuralNetworkPtr makeNeuralNetwork (NeuralFactory &neuralFactory)=0`
- virtual `NeuralCreatorPtr makeNeuralCreator ()=0`

class NeuralFactory -

Definition at line 4 of file NeuralFactory.h.

```
5.47.2.1 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](#).



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

Implemented in [MLPfactory](#).

5.47.2.3 `virtual ConContainerPtr NeuralFactory::makeConContainer ( )` [pure virtual]

Implemented in [MLPfactory](#), and [RBFfactory](#).

Referenced by `Neuron::Neuron()`.

Here is the caller graph for this function:

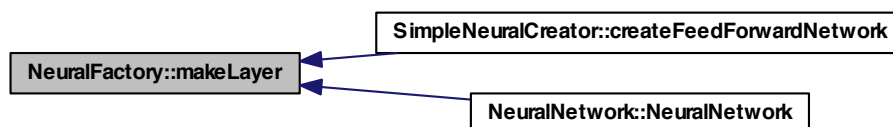


5.47.2.4 `virtual LayerPtr NeuralFactory::makeLayer ( )` [pure virtual]

Implemented in [MLPfactory](#), and [RBFfactory](#).

Referenced by `SimpleNeuralCreator::createFeedForwardNetwork()`, and `NeuralNetwork::NeuralNetwork()`.

Here is the caller graph for this function:



5.47.2.5 `virtual LayerContainerPtr NeuralFactory::makeLayerContainer ( )` [pure virtual]

Implemented in [MLPfactory](#), and [RBFfactory](#).

Referenced by `NeuralNetwork::NeuralNetwork()`.

Here is the caller graph for this function:



**5.47.2.6** `virtual NeuralCreatorPtr NeuralFactory::makeNeuralCreator ( ) [pure virtual]`

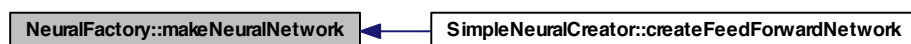
Implemented in [MLPfactory](#), and [RBFfactory](#).

**5.47.2.7** `virtual NeuralNetworkPtr NeuralFactory::makeNeuralNetwork ( NeuralFactory & neuralFactory ) [pure virtual]`

Implemented in [MLPfactory](#), and [RBFfactory](#).

Referenced by `SimpleNeuralCreator::createFeedForwardNetwork()`.

Here is the caller graph for this function:



**5.47.2.8** `virtual NeuronPtr NeuralFactory::makeNeuron ( Handler ld ) [pure virtual]`

Implemented in [MLPfactory](#), and [RBFfactory](#).

Referenced by `SimpleNeuralCreator::createFeedForwardNetwork()`.

Here is the caller graph for this function:



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

Implemented in [MLPfactory](#), and [RBFfactory](#).

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

Implemented in [MLPfactory](#).

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

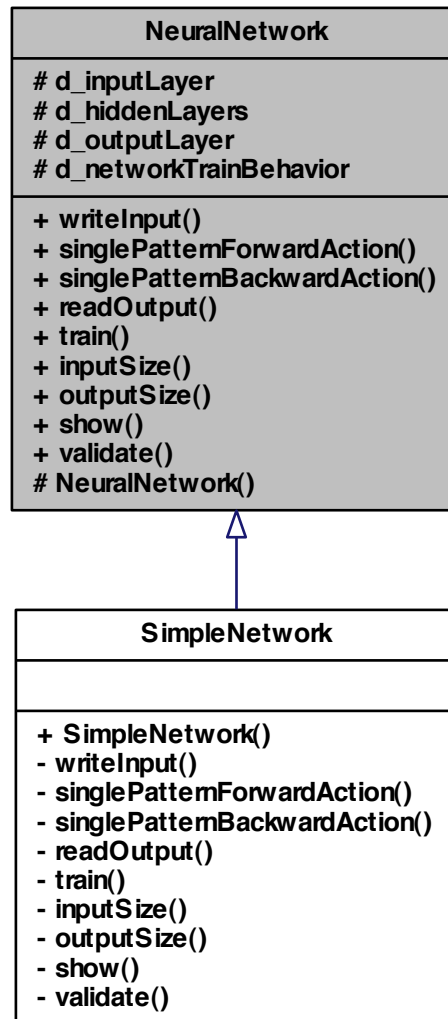
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralFactory.h`

## 5.48 NeuralNetwork Class Reference

class [NeuralNetwork](#) -

```
#include <NeuralNetwork.h>
```

Inheritance diagram for NeuralNetwork:



### Public Member Functions

- virtual void [writeInput](#) (std::vector< double >::iterator &iterator)=0
- virtual void [singlePatternForwardAction](#) ()=0
- virtual void [singlePatternBackwardAction](#) ()=0
- virtual void [readOutput](#) (std::vector< double >::iterator &iterator)=0

- virtual Rcpp::List [train](#) (Rcpp::List parameterList)=0
- virtual size\_type [inputSize](#) ()=0
- virtual size\_type [outputSize](#) ()=0
- virtual void [show](#) ()=0
- virtual bool [validate](#) ()=0

### Protected Member Functions

- [NeuralNetwork](#) ([NeuralFactory](#) &neuralFactory)

### Protected Attributes

- [LayerPtr](#) d\_inputLayer
- [LayerContainerPtr](#) d\_hiddenLayers
- [LayerPtr](#) d\_outputLayer
- [NetworkTrainBehaviorPtr](#) d\_networkTrainBehavior

### Friends

- class [SimpleNeuralCreator](#)

#### 5.48.1 Detailed Description

class [NeuralNetwork](#) -

Definition at line 3 of file NeuralNetwork.h.

#### 5.48.2 Constructor & Destructor Documentation

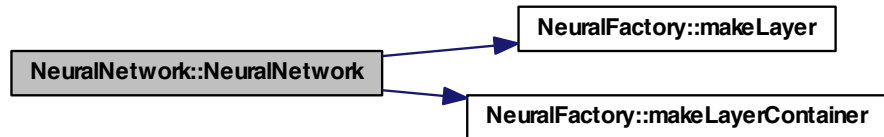
5.48.2.1 [NeuralNetwork::NeuralNetwork](#) ( [NeuralFactory](#) & *neuralFactory* )  
[protected]

Definition at line 12 of file NeuralNetwork.cpp.

References [d\\_hiddenLayers](#), [d\\_inputLayer](#), [d\\_outputLayer](#), [NeuralFactory::makeLayer\(\)](#), and [NeuralFactory::makeLayerContainer\(\)](#).

```
{  
    d_inputLayer = neuralFactory.makeLayer();  
    d_hiddenLayers = neuralFactory.makeLayerContainer();  
    d_outputLayer = neuralFactory.makeLayer();  
}
```

Here is the call graph for this function:



### 5.48.3 Member Function Documentation

5.48.3.1 `virtual size_type NeuralNetwork::inputSize ( )` [pure virtual]

Implemented in [SimpleNetwork](#).

5.48.3.2 `virtual size_type NeuralNetwork::outputSize ( )` [pure virtual]

Implemented in [SimpleNetwork](#).

5.48.3.3 `virtual void NeuralNetwork::readOutput ( std::vector< double >::iterator & iterator )`  
[pure virtual]

Implemented in [SimpleNetwork](#).

5.48.3.4 `virtual void NeuralNetwork::show ( )` [pure virtual]

Implemented in [SimpleNetwork](#).

5.48.3.5 `virtual void NeuralNetwork::singlePatternBackwardAction ( )` [pure virtual]

Implemented in [SimpleNetwork](#).

5.48.3.6 `virtual void NeuralNetwork::singlePatternForwardAction ( )` [pure virtual]

Implemented in [SimpleNetwork](#).

**5.48.3.7** `virtual Rcpp::List NeuralNetwork::train ( Rcpp::List parameterList )` [pure virtual]

Implemented in [SimpleNetwork](#).

**5.48.3.8** `virtual bool NeuralNetwork::validate ( )` [pure virtual]

Implemented in [SimpleNetwork](#).

**5.48.3.9** `virtual void NeuralNetwork::writeInput ( std::vector< double >::iterator & iterator )` [pure virtual]

Implemented in [SimpleNetwork](#).

## 5.48.4 Friends And Related Function Documentation

**5.48.4.1** `friend class SimpleNeuralCreator` [friend]

Definition at line 12 of file NeuralNetwork.h.

## 5.48.5 Member Data Documentation

**5.48.5.1** `LayerContainerPtr NeuralNetwork::d_hiddenLayers` [protected]

Definition at line 7 of file NeuralNetwork.h.

Referenced by `NeuralNetwork()`, `SimpleNetwork::show()`, `SimpleNetwork::singlePatternBackwardAction()`, `SimpleNetwork::singlePatternForwardAction()`, and `SimpleNetwork::validate()`.

**5.48.5.2** `LayerPtr NeuralNetwork::d_inputLayer` [protected]

Definition at line 6 of file NeuralNetwork.h.

Referenced by `SimpleNetwork::inputSize()`, `NeuralNetwork()`, `SimpleNetwork::show()`, `SimpleNetwork::validate()`, and `SimpleNetwork::writeInput()`.

**5.48.5.3** `NetworkTrainBehaviorPtr NeuralNetwork::d_networkTrainBehavior` [protected]

Definition at line 9 of file NeuralNetwork.h.

Referenced by `SimpleNetwork::train()`.

#### 5.48.5.4 LayerPtr NeuralNetwork::d\_outputLayer [protected]

Definition at line 8 of file NeuralNetwork.h.

Referenced by NeuralNetwork(), SimpleNetwork::outputSize(), SimpleNetwork::readOutput(), SimpleNetwork::show(), SimpleNetwork::singlePatternBackwardAction(), SimpleNetwork::singlePatternForwardAction() and SimpleNetwork::validate().

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

- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeader/NeuralNetwork.h
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NeuralNetwork.cpp

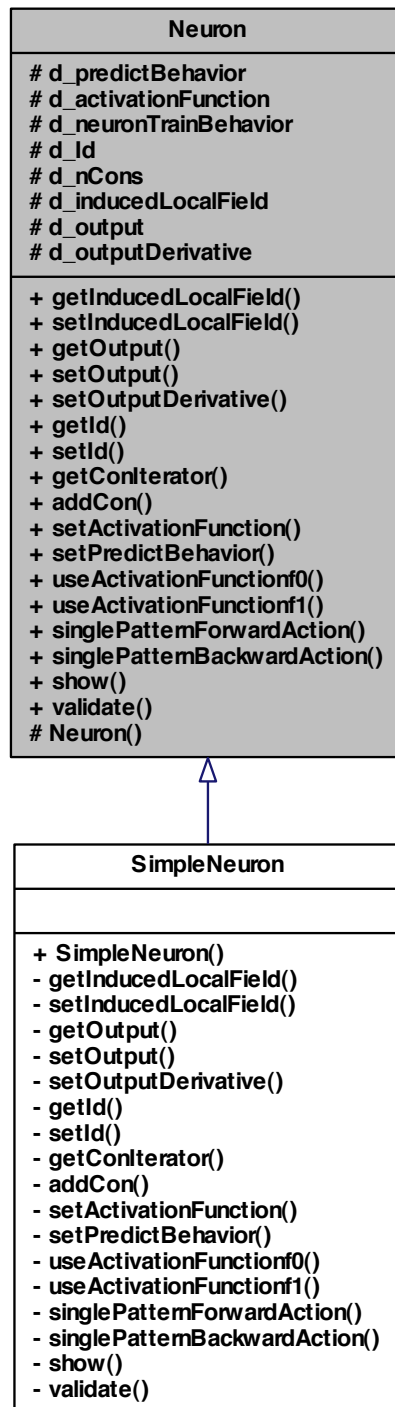
## 5.49 Neuron Class Reference

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 void [setOutputDerivative](#) (double outputDerivative)=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 [useActivationFunction0](#) ()=0
- virtual double [useActivationFunction1](#) ()=0
- virtual void [singlePatternForwardAction](#) ()=0
- virtual void [singlePatternBackwardAction](#) ()=0
- virtual void [show](#) ()=0
- virtual bool [validate](#) ()=0

### Protected Member Functions

- [Neuron](#) ([NeuralFactory](#) &neuralFactory)

### Protected Attributes

- [PredictBehaviorPtr](#) d\_predictBehavior
- [ActivationFunctionPtr](#) d\_activationFunction
- [NeuronTrainBehaviorPtr](#) d\_neuronTrainBehavior
- [Handler](#) d\_Id
- [ConContainerPtr](#) d\_nCons
- double d\_inducedLocalField
- double d\_output
- double d\_outputDerivative

### Friends

- class [MLPfactory](#)

### 5.49.1 Detailed Description

class [Neuron](#) -

Definition at line 3 of file [Neuron.h](#).

## 5.49.2 Constructor & Destructor Documentation

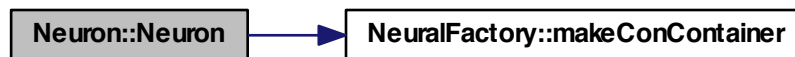
### 5.49.2.1 Neuron::Neuron ( NeuralFactory & *neuralFactory* ) [protected]

Definition at line 12 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:



## 5.49.3 Member Function Documentation

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

Implemented in [SimpleNeuron](#).

### 5.49.3.2 virtual ConIteratorPtr Neuron::getConIterator ( ) [pure virtual]

Implemented in [SimpleNeuron](#).

### 5.49.3.3 virtual Handler Neuron::getId ( ) [pure virtual]

Implemented in [SimpleNeuron](#).

### 5.49.3.4 virtual double Neuron::getInducedLocalField ( ) [pure virtual]

Implemented in [SimpleNeuron](#).

### 5.49.3.5 virtual double Neuron::getOutput ( ) [pure virtual]

Implemented in [SimpleNeuron](#).

5.49.3.6 `virtual void Neuron::setActivationFunction ( ActivationFunctionPtr  
activationFunctionPtr ) [pure virtual]`

Implemented in [SimpleNeuron](#).

5.49.3.7 `virtual void Neuron::setId ( Handler Id ) [pure virtual]`

Implemented in [SimpleNeuron](#).

5.49.3.8 `virtual void Neuron::setInducedLocalField ( double inducedLocalField ) [pure  
virtual]`

Implemented in [SimpleNeuron](#).

5.49.3.9 `virtual void Neuron::setOutput ( double output ) [pure virtual]`

Implemented in [SimpleNeuron](#).

5.49.3.10 `virtual void Neuron::setOutputDerivative ( double outputDerivative ) [pure  
virtual]`

Implemented in [SimpleNeuron](#).

5.49.3.11 `virtual void Neuron::setPredictBehavior ( PredictBehaviorPtr predictBehaviorPtr )  
[pure virtual]`

Implemented in [SimpleNeuron](#).

5.49.3.12 `virtual void Neuron::show ( ) [pure virtual]`

Implemented in [SimpleNeuron](#).

5.49.3.13 `virtual void Neuron::singlePatternBackwardAction ( ) [pure virtual]`

Implemented in [SimpleNeuron](#).

5.49.3.14 `virtual void Neuron::singlePatternForwardAction ( ) [pure virtual]`

Implemented in [SimpleNeuron](#).

5.49.3.15 `virtual double Neuron::useActivationFunction0 ( ) [pure virtual]`

Implemented in [SimpleNeuron](#).

5.49.3.16 `virtual double Neuron::useActivationFunction1 ( ) [pure virtual]`

Implemented in [SimpleNeuron](#).

5.49.3.17 `virtual bool Neuron::validate ( ) [pure virtual]`

Implemented in [SimpleNeuron](#).

#### 5.49.4 Friends And Related Function Documentation

5.49.4.1 `friend class MLPfactory [friend]`

Definition at line 16 of file Neuron.h.

#### 5.49.5 Member Data Documentation

5.49.5.1 `ActivationFunctionPtr Neuron::d_activationFunction [protected]`

Definition at line 7 of file Neuron.h.

Referenced by `SimpleNeuron::setActivationFunction()`, `SimpleNeuron::useActivationFunction0()`, and `SimpleNeuron::useActivationFunction1()`.

5.49.5.2 `Handler Neuron::d_Id [protected]`

Definition at line 9 of file Neuron.h.

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

5.49.5.3 `double Neuron::d_inducedLocalField [protected]`

Definition at line 11 of file Neuron.h.

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

5.49.5.4 `ConContainerPtr Neuron::d_nCons [protected]`

Definition at line 10 of file Neuron.h.

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

5.49.5.5 `NeuronTrainBehaviorPtr Neuron::d_neuronTrainBehavior [protected]`

Definition at line 8 of file Neuron.h.

#### 5.49.5.6 double Neuron::d\_output [protected]

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

Definition at line 13 of file Neuron.h.

Referenced by SimpleNeuron::setOutputDerivative().

Definition at line 6 of file Neuron.h.

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

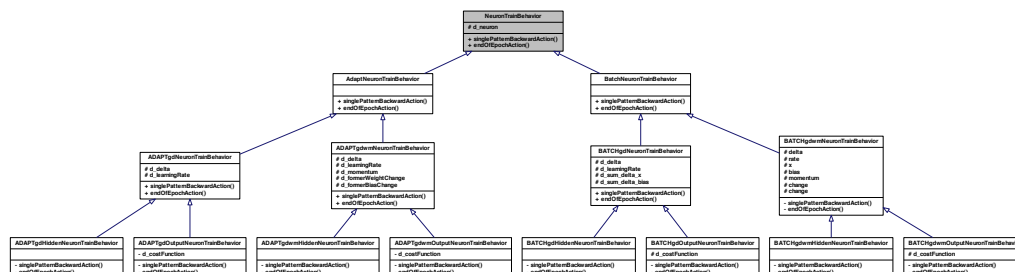
The documentation for this class was generated from the following files:

- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Neuron.cpp

## 5.50 NeuronTrainBehavior Class Reference

```
#include <NeuronTrainBehavior.h>
```

Inheritance diagram for NeuronTrainBehavior:



## Public Member Functions

- virtual void [singlePatternBackwardAction](#) ()=0
- virtual void [endOfEpochAction](#) ()=0

## Protected Attributes

- [NeuronWeakPtr d\\_neuron](#)

### 5.50.1 Detailed Description

class [NeuronTrainBehavior](#) -

Definition at line 4 of file [NeuronTrainBehavior.h](#).

### 5.50.2 Member Function Documentation

**5.50.2.1** virtual void [NeuronTrainBehavior::endOfEpochAction](#) ( ) [pure virtual]

Implemented in [ADAPTgdHiddenNeuronTrainBehavior](#), [ADAPTgdNeuronTrainBehavior](#), [ADAPTgdOutputNeuronTrainBehavior](#), [ADAPTgdwmHiddenNeuronTrainBehavior](#), [ADAPTgdwmNeuronTrainBehavior](#), [ADAPTgdwmOutputNeuronTrainBehavior](#), [AdaptNeuronTrainBehavior](#), [BATCHgdHiddenNeuronTrainBehavior](#), [BATCHgdNeuronTrainBehavior](#), [BATCHgdOutputNeuronTrainBehavior](#), [BATCHgdwmHiddenNeuronTrainBehavior](#), [BATCHgdwmNeuronTrainBehavior](#), [BATCHgdwmOutputNeuronTrainBehavior](#), and [BatchNeuronTrainBehavior](#).

**5.50.2.2** virtual void [NeuronTrainBehavior::singlePatternBackwardAction](#) ( ) [pure virtual]

Implemented in [ADAPTgdHiddenNeuronTrainBehavior](#), [ADAPTgdNeuronTrainBehavior](#), [ADAPTgdOutputNeuronTrainBehavior](#), [ADAPTgdwmHiddenNeuronTrainBehavior](#), [ADAPTgdwmNeuronTrainBehavior](#), [ADAPTgdwmOutputNeuronTrainBehavior](#), [AdaptNeuronTrainBehavior](#), [BATCHgdHiddenNeuronTrainBehavior](#), [BATCHgdNeuronTrainBehavior](#), [BATCHgdOutputNeuronTrainBehavior](#), [BATCHgdwmHiddenNeuronTrainBehavior](#), [BATCHgdwmNeuronTrainBehavior](#), [BATCHgdwmOutputNeuronTrainBehavior](#), and [BatchNeuronTrainBehavior](#).

### 5.50.3 Member Data Documentation

**5.50.3.1** [NeuronWeakPtr](#) [NeuronTrainBehavior::d\\_neuron](#) [protected]

Definition at line 7 of file [NeuronTrainBehavior.h](#).

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

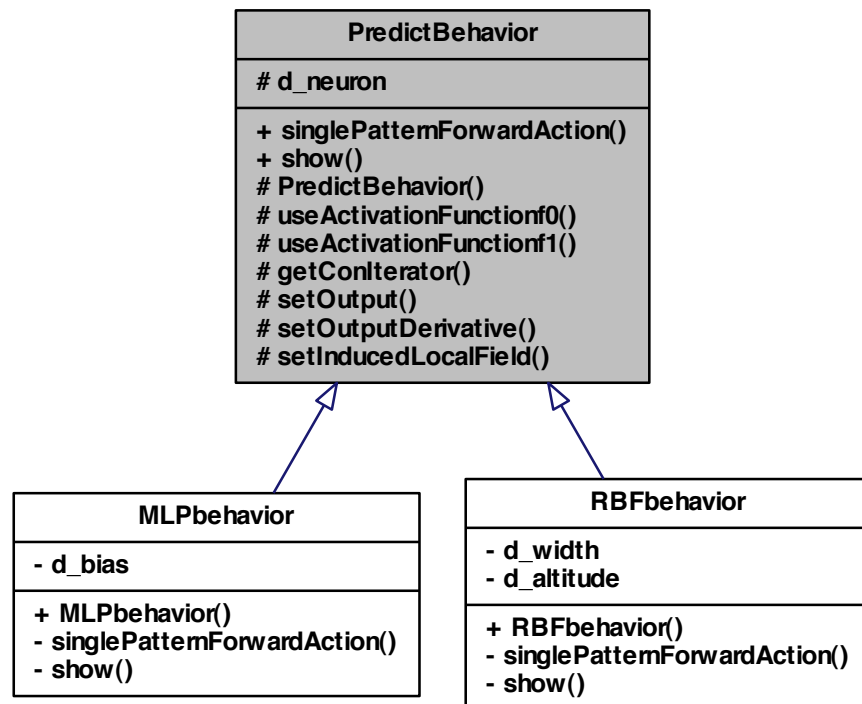
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuronTrainBehavior.h](#)

## 5.51 PredictBehavior Class Reference

class [PredictBehavior](#) -

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

Inheritance diagram for PredictBehavior:



### Public Member Functions

- virtual void [singlePatternForwardAction](#) ()=0
- virtual void [show](#) ()=0

### Protected Member Functions

- [PredictBehavior](#) ([NeuronPtr](#) neuronPtr)
- double [useActivationFunction0](#) ()
- double [useActivationFunction1](#) ()



- [ConIteratorPtr](#) [getConIterator](#) ()
- void [setOutput](#) (double output)
- void [setOutputDerivative](#) (double outputDerivative)
- void [setInducedLocalField](#) (double inducedLocalField)

### Protected Attributes

- [NeuronWeakPtr](#) [d\\_neuron](#)

#### 5.51.1 Detailed Description

class [PredictBehavior](#) -

Definition at line 4 of file PredictBehavior.h.

#### 5.51.2 Constructor & Destructor Documentation

##### 5.51.2.1 [PredictBehavior::PredictBehavior](#) ( [NeuronPtr](#) *neuronPtr* ) [protected]

Definition at line 14 of file PredictBehavior.cpp.

```

    d\_neuron(neuronPtr)
{
}
:
```

#### 5.51.3 Member Function Documentation

##### 5.51.3.1 [ConIteratorPtr](#) [PredictBehavior::getConIterator](#) ( ) [protected]

Definition at line 36 of file PredictBehavior.cpp.

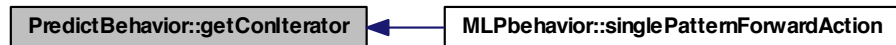
References [d\\_neuron](#).

Referenced by [MLPbehavior::singlePatternForwardAction](#)().

```

{
    NeuronPtr neuronPtr( d\_neuron.lock() );
    return neuronPtr->getConIterator();
}
```

Here is the caller graph for this function:



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

Definition at line 59 of file PredictBehavior.cpp.

References `d_neuron`.

Referenced by `MLPbehavior::singlePatternForwardAction()`.

```

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

Here is the caller graph for this function:



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

Definition at line 43 of file PredictBehavior.cpp.

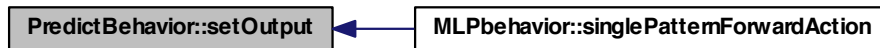
References `d_neuron`.

Referenced by `MLPbehavior::singlePatternForwardAction()`.

```

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

Here is the caller graph for this function:



**5.51.3.4** `void PredictBehavior::setOutputDerivative ( double outputDerivative )`  
 [protected]

Definition at line 51 of file `PredictBehavior.cpp`.

References `d_neuron`.

Referenced by `MLPbehavior::singlePatternForwardAction()`.

```

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

Here is the caller graph for this function:



**5.51.3.5** `virtual void PredictBehavior::show ( )` [pure virtual]

Implemented in [MLPbehavior](#), and [RBFbehavior](#).

**5.51.3.6** `virtual void PredictBehavior::singlePatternForwardAction ( )` [pure virtual]

Implemented in [MLPbehavior](#), and [RBFbehavior](#).

**5.51.3.7** `double PredictBehavior::useActivationFunction0 ( )` [protected]

Definition at line 20 of file `PredictBehavior.cpp`.

References `d_neuron`.

Referenced by `MLPbehavior::singlePatternForwardAction()`.

```
{
    NeuronPtr neuronPtr( d_neuron.lock() ) ;
    return neuronPtr->useActivationFunction0();
}
```

Here is the caller graph for this function:



#### 5.51.3.8 `double PredictBehavior::useActivationFunction1 ( )` [protected]

Definition at line 28 of file `PredictBehavior.cpp`.

References `d_neuron`.

Referenced by `MLPbehavior::singlePatternForwardAction()`.

```
{
    NeuronPtr neuronPtr( d_neuron.lock() ) ;
    return neuronPtr->useActivationFunction1();
}
```

Here is the caller graph for this function:



### 5.51.4 Member Data Documentation

#### 5.51.4.1 `NeuronWeakPtr PredictBehavior::d_neuron` [protected]

Definition at line 7 of file `PredictBehavior.h`.

Referenced by `getConlterator()`, `setInducedLocalField()`, `setOutput()`, `setOutputDerivative()`, `useActivationFunction0()`, and `useActivationFunction1()`.

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

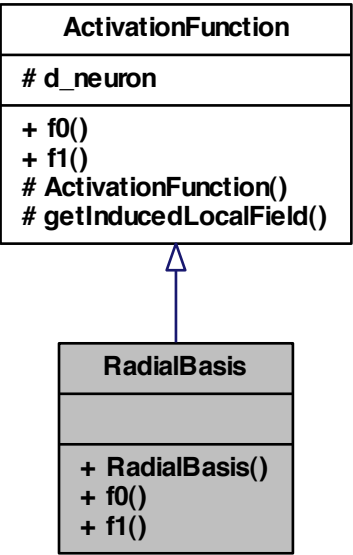
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/PredictBe](#)
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/PredictBehavior.cpp](#)

## 5.52 RadialBasis Class Reference

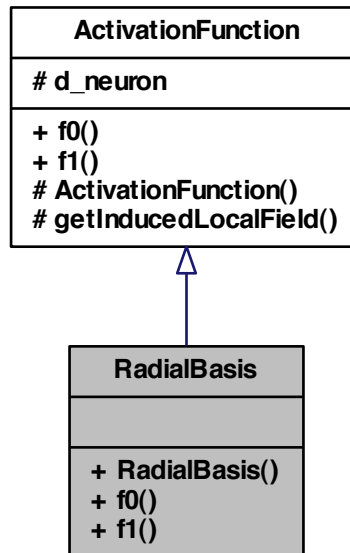
class [RadialBasis](#) -

```
#include <RadialBasis.h>
```

Inheritance diagram for RadialBasis:



Collaboration diagram for RadialBasis:



### Public Member Functions

- [RadialBasis](#) ([NeuronPtr](#) neuronPtr)
- double [f0](#) ()
- double [f1](#) ()

#### 5.52.1 Detailed Description

class [RadialBasis](#) -

Definition at line 5 of file RadialBasis.h.

#### 5.52.2 Constructor & Destructor Documentation

5.52.2.1 [RadialBasis::RadialBasis](#) ( [NeuronPtr](#) neuronPtr )

#### 5.52.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

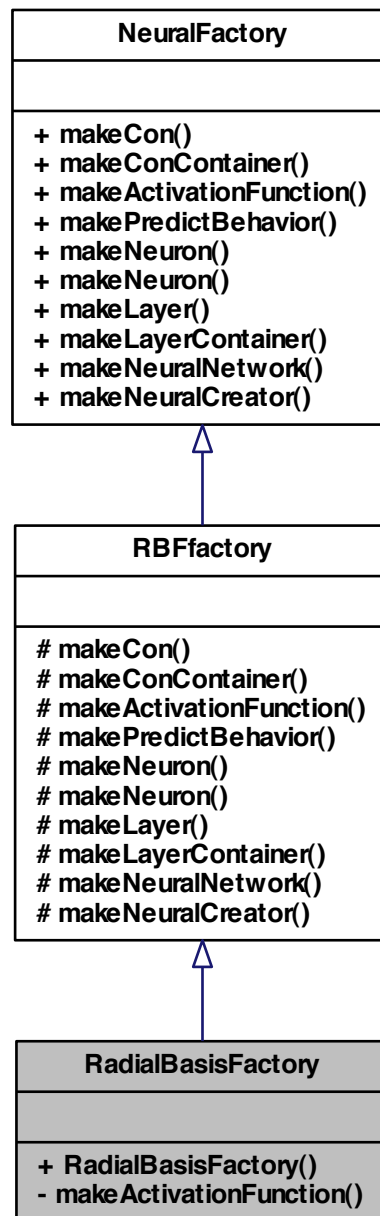
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasisFactory.h`

## 5.53 RadialBasisFactory Class Reference

class [RadialBasisFactory](#) -

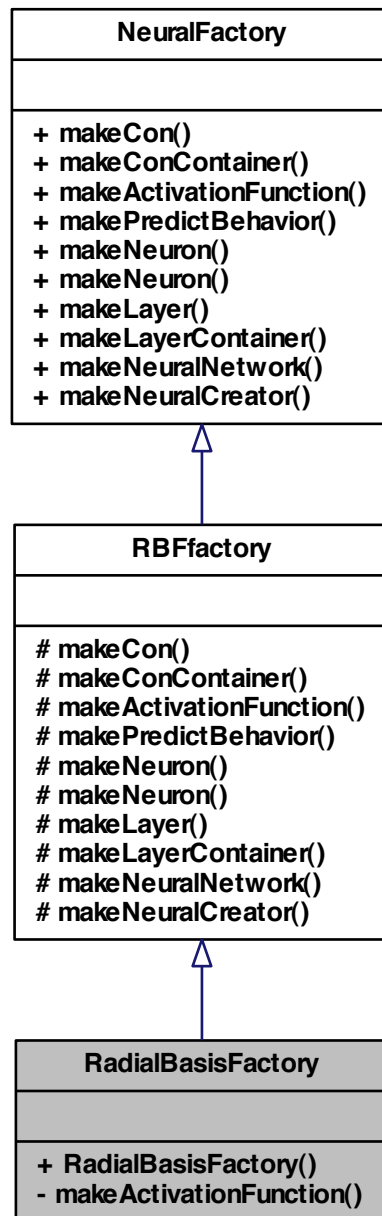
```
#include <RadialBasisFactory.h>
```

Inheritance diagram for RadialBasisFactory:





Collaboration diagram for RadialBasisFactory:



## Public Member Functions

- [RadialBasisFactory](#) ()

## Private Member Functions

- [ActivationFunctionPtr](#) [makeActivationFunction](#) ([NeuronPtr](#) neuronPtr)

### 5.53.1 Detailed Description

class [RadialBasisFactory](#) -

Definition at line 5 of file RadialBasisFactory.h.

### 5.53.2 Constructor & Destructor Documentation

#### 5.53.2.1 [RadialBasisFactory::RadialBasisFactory](#) ( )

### 5.53.3 Member Function Documentation

#### 5.53.3.1 [ActivationFunctionPtr](#) [RadialBasisFactory::makeActivationFunction](#) ( [NeuronPtr](#) *neuronPtr* ) [private, virtual]

Implements [RBFfactory](#).

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

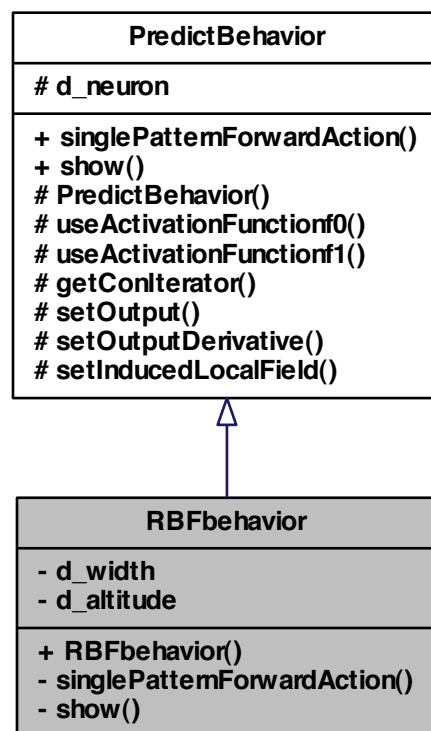
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.54 RBFbehavior Class Reference

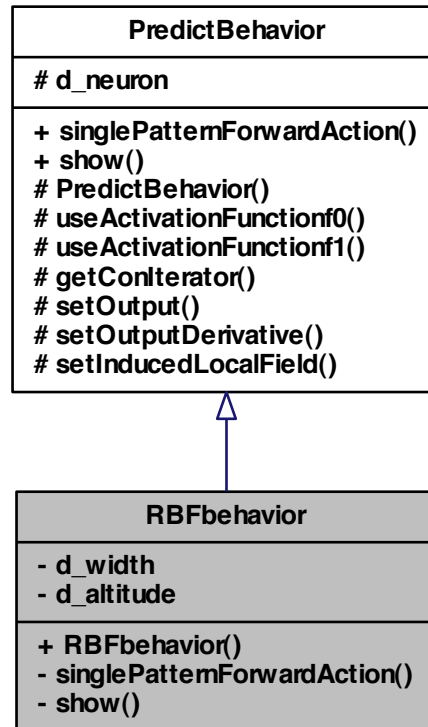
class [RBFbehavior](#) -

```
#include <RBFbehavior.h>
```

Inheritance diagram for RBFbehavior:



Collaboration diagram for RBFbehavior:



### Public Member Functions

- [RBFbehavior](#) ([NeuronPtr](#) neuronPtr)

### Private Member Functions

- void [singlePatternForwardAction](#) ()
- void [show](#) ()

### Private Attributes

- double [d\\_width](#)
- double [d\\_altitude](#)

### 5.54.1 Detailed Description

class [RBFbehavior](#) -

Definition at line 5 of file RBFbehavior.h.

### 5.54.2 Constructor & Destructor Documentation

5.54.2.1 [RBFbehavior::RBFbehavior](#) ( [NeuronPtr](#) *neuronPtr* )

### 5.54.3 Member Function Documentation

5.54.3.1 [void RBFbehavior::show](#) ( ) [private, virtual]

Implements [PredictBehavior](#).

5.54.3.2 [void RBFbehavior::singlePatternForwardAction](#) ( ) [private, virtual]

Implements [PredictBehavior](#).

### 5.54.4 Member Data Documentation

5.54.4.1 [double RBFbehavior::d\\_altitude](#) [private]

Definition at line 9 of file RBFbehavior.h.

5.54.4.2 [double RBFbehavior::d\\_width](#) [private]

Definition at line 8 of file RBFbehavior.h.

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

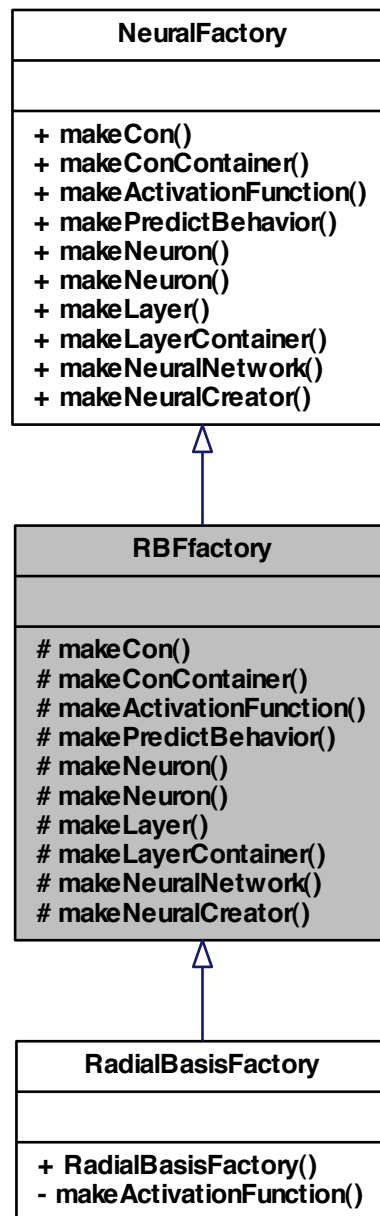
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFbehavior.h](#)

## 5.55 RBFfactory Class Reference

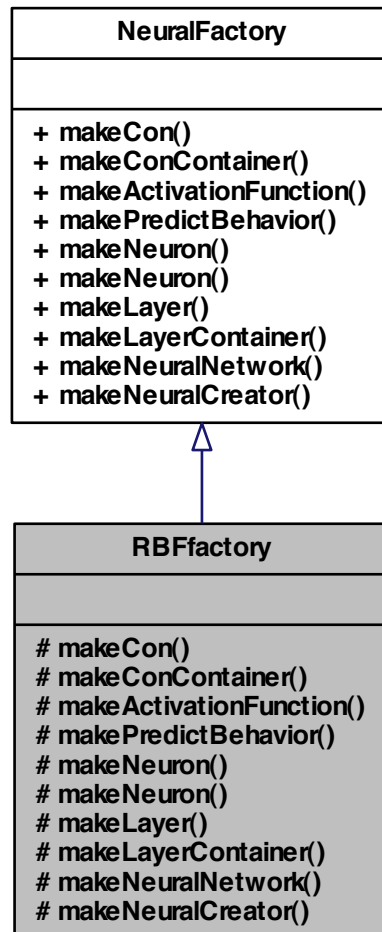
class [RBFfactory](#) -

```
#include <RBFfactory.h>
```

Inheritance diagram for RBFfactory:



Collaboration diagram for RBFactory:



### 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)

- [LayerPtr](#) `makeLayer ()`
- [LayerContainerPtr](#) `makeLayerContainer ()`
- [NeuralNetworkPtr](#) `makeNeuralNetwork (NeuralFactory &neuralFactory)`
- [NeuralCreatorPtr](#) `makeNeuralCreator ()`

### 5.55.1 Detailed Description

class [RBFfactory](#) -

Definition at line 5 of file [RBFfactory.h](#).

### 5.55.2 Member Function Documentation

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

Implements [NeuralFactory](#).

Implemented in [RadialBasisFactory](#).

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

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

Implements [NeuralFactory](#).

**5.55.2.4** `LayerPtr RBFfactory::makeLayer ( )` [protected, virtual]

Implements [NeuralFactory](#).

**5.55.2.5** `LayerContainerPtr RBFfactory::makeLayerContainer ( )` [protected, virtual]

Implements [NeuralFactory](#).

**5.55.2.6** `NeuralCreatorPtr RBFfactory::makeNeuralCreator ( )` [protected, virtual]

Implements [NeuralFactory](#).



**5.55.2.7** `NeuralNetworkPtr RBFfactory::makeNeuralNetwork ( NeuralFactory & neuralFactory )` [protected, virtual]

Implements [NeuralFactory](#).

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

Implements [NeuralFactory](#).

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

Implements [NeuralFactory](#).

**5.55.2.10** `PredictBehaviorPtr RBFfactory::makePredictBehavior ( )` [protected]

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

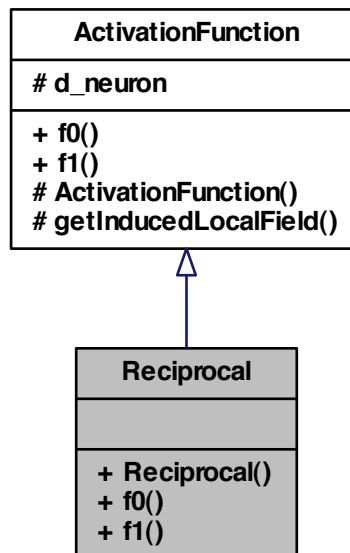
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFfactory.h`

## 5.56 Reciprocal Class Reference

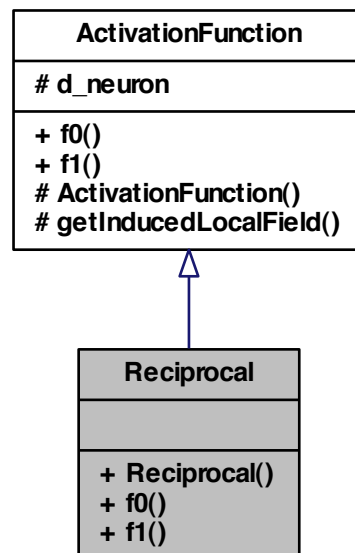
class [Reciprocal](#) -

```
#include <Reciprocal.h>
```

Inheritance diagram for Reciprocal:



Collaboration diagram for Reciprocal:



### Public Member Functions

- [Reciprocal](#) ([NeuronPtr](#) neuronPtr)
- void [f0](#) ()
- void [f1](#) ()

### 5.56.1 Detailed Description

class [Reciprocal](#) -

Definition at line 5 of file [Reciprocal.h](#).

### 5.56.2 Constructor & Destructor Documentation

5.56.2.1 [Reciprocal::Reciprocal](#) ( [NeuronPtr](#) neuronPtr )

### 5.56.3 Member Function Documentation

5.56.3.1 void Reciprocal::f0 ( ) [virtual]

Implements [ActivationFunction](#).

5.56.3.2 void Reciprocal::f1 ( ) [virtual]

Implements [ActivationFunction](#).

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

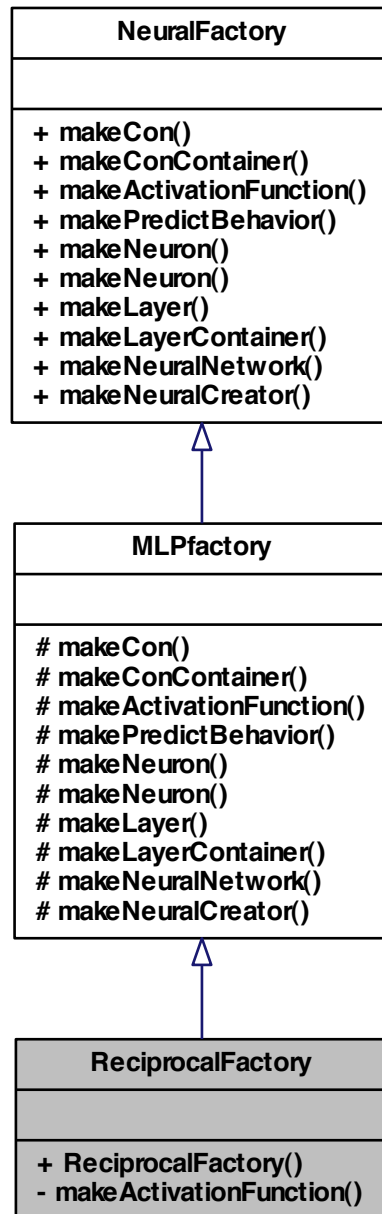
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.57 ReciprocalFactory Class Reference

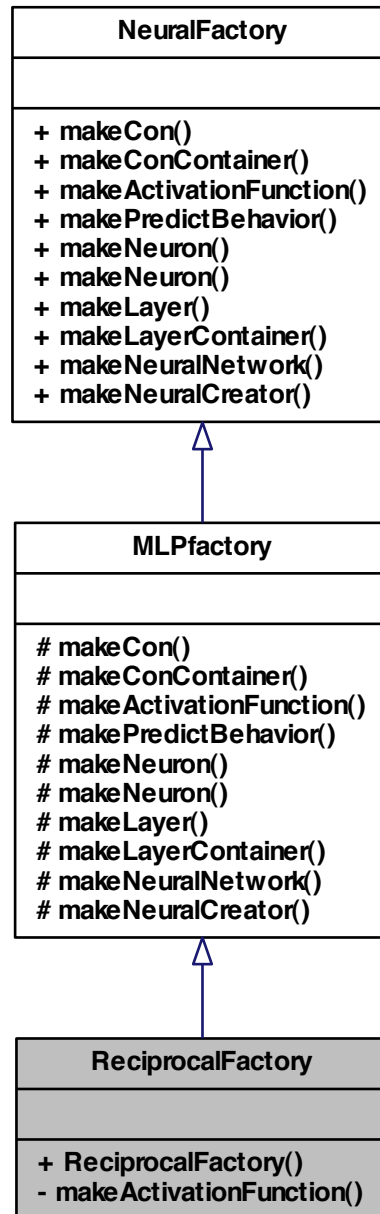
class [ReciprocalFactory](#) -

```
#include <ReciprocalFactory.h>
```

Inheritance diagram for ReciprocalFactory:



Collaboration diagram for ReciprocalFactory:



## Public Member Functions

- [ReciprocalFactory](#) ()

## Private Member Functions

- [ActivationFunctionPtr](#) [makeActivationFunction](#) ([NeuronPtr](#) neuronPtr)

### 5.57.1 Detailed Description

class [ReciprocalFactory](#) -

Definition at line 5 of file [ReciprocalFactory.h](#).

### 5.57.2 Constructor & Destructor Documentation

5.57.2.1 [ReciprocalFactory::ReciprocalFactory](#) ( )

### 5.57.3 Member Function Documentation

5.57.3.1 [ActivationFunctionPtr](#) [ReciprocalFactory::makeActivationFunction](#) ( [NeuronPtr](#) *neuronPtr* ) [[private](#), [virtual](#)]

Implements [MLPfactory](#).

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

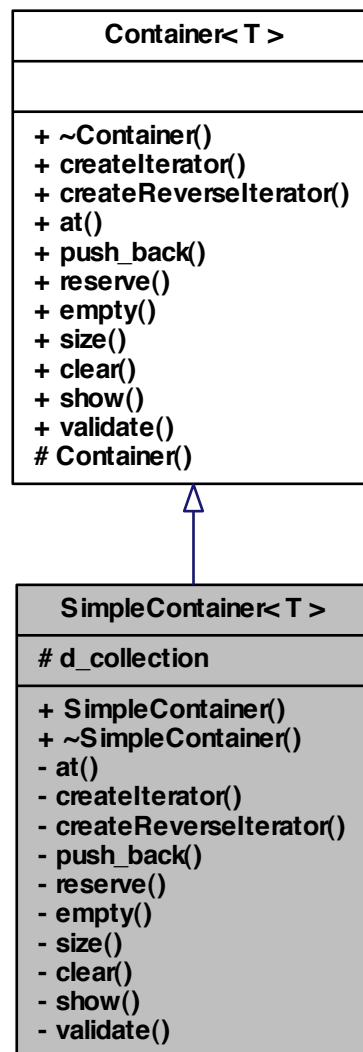
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ReciprocalFactory.h](#)

## 5.58 SimpleContainer< T > Class Template Reference

class [SimpleContainer](#) -

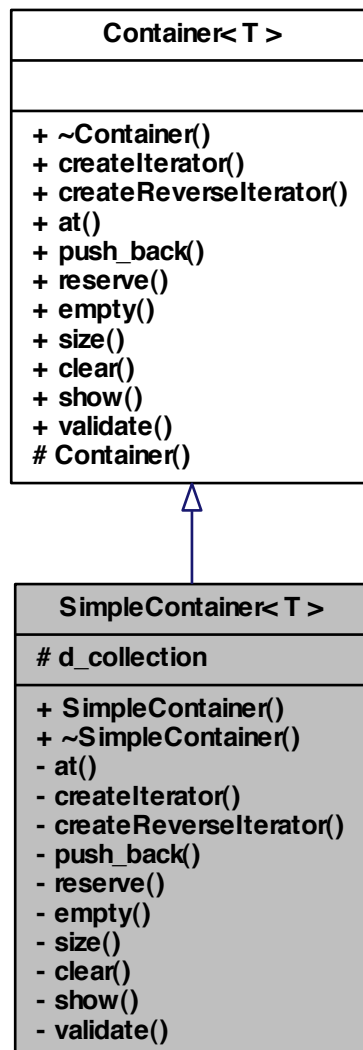
`#include <SimpleContainer.h>`

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`)
- `boost::shared_ptr< Iterator< T > >` [createIterator](#) ()
- `boost::shared_ptr< Iterator< T > >` [createReverserIterator](#) ()
- `void` [push\\_back](#) (`T` `const` &`const_reference`)
- `void` [reserve](#) (`int` `n`)
- `bool` [empty](#) ()
- `size_type` [size](#) ()
- `void` [clear](#) ()
- `void` [show](#) ()
- `bool` [validate](#) ()

## Friends

- `class` [SimpleContainerReverserIterator< T >](#)
- `class` [SimpleContainerIterator< T >](#)

### 5.58.1 Detailed Description

`template<typename T>class SimpleContainer< T >`

`class` [SimpleContainer](#) -

Definition at line 6 of file `SimpleContainer.h`.

### 5.58.2 Constructor & Destructor Documentation

5.58.2.1 `template<typename T > SimpleContainer< T >::SimpleContainer ( )`

5.58.2.2 `template<typename T > SimpleContainer< T >::~~SimpleContainer ( )`

### 5.58.3 Member Function Documentation

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

5.58.3.4 `template<typename T> boost::shared_ptr< Iterator<T> > SimpleContainer< T >::createReverseIterator ( )` [private, virtual]

Implements [Container< T >](#).

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

Implements [Container< T >](#).

5.58.3.6 `template<typename T> void SimpleContainer< T >::push.back ( T const & const.reference )` [private, virtual]

Implements [Container< T >](#).

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

### 5.58.4 Friends And Related Function Documentation

5.58.4.1 `template<typename T > friend class SimpleContainerIterator< T >`  
`[friend]`

Definition at line 13 of file SimpleContainer.h.

5.58.4.2 `template<typename T > friend class SimpleContainerReverselIterator< T >`  
`[friend]`

Definition at line 12 of file SimpleContainer.h.

### 5.58.5 Member Data Documentation

5.58.5.1 `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 file:

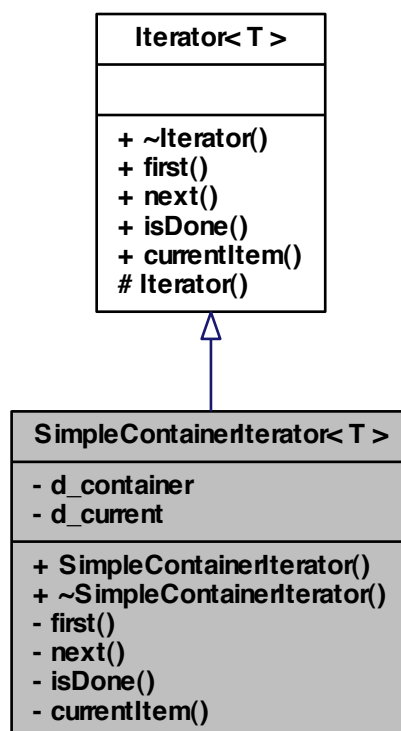
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.59 SimpleContainerIterator< T > Class Template Reference

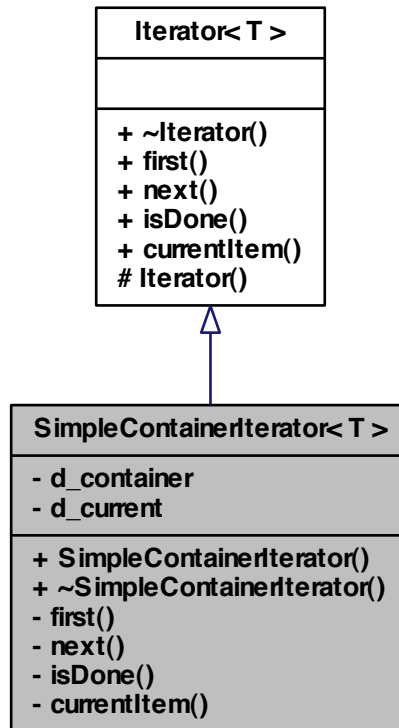
class [SimpleContainerIterator](#) -

```
#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](#)
- [int d\\_current](#)

### Friends

- [class SimpleContainer< T >](#)

### 5.59.1 Detailed Description

`template<typename T>class SimpleContainerIterator< T >`

class [SimpleContainerIterator](#) -

Definition at line 6 of file SimpleContainerIterator.h.

### 5.59.2 Constructor & Destructor Documentation

5.59.2.1 `template<typename T > SimpleContainerIterator< T >::SimpleContainerIterator ( )`

5.59.2.2 `template<typename T > SimpleContainerIterator< T >::~~SimpleContainerIterator ( )`

### 5.59.3 Member Function Documentation

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

Implements [Iterator< T >](#).

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

Implements [Iterator< T >](#).

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

Implements [Iterator< T >](#).

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

Implements [Iterator< T>](#).

#### 5.59.4 Friends And Related Function Documentation

5.59.4.1 `template<typename T> friend class SimpleContainer< T> [friend]`

Definition at line 13 of file SimpleContainerIterator.h.

#### 5.59.5 Member Data Documentation

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

Definition at line 9 of file SimpleContainerIterator.h.

5.59.5.2 `template<typename T> int SimpleContainerIterator< T>::d_current [private]`

Definition at line 10 of file SimpleContainerIterator.h.

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

- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

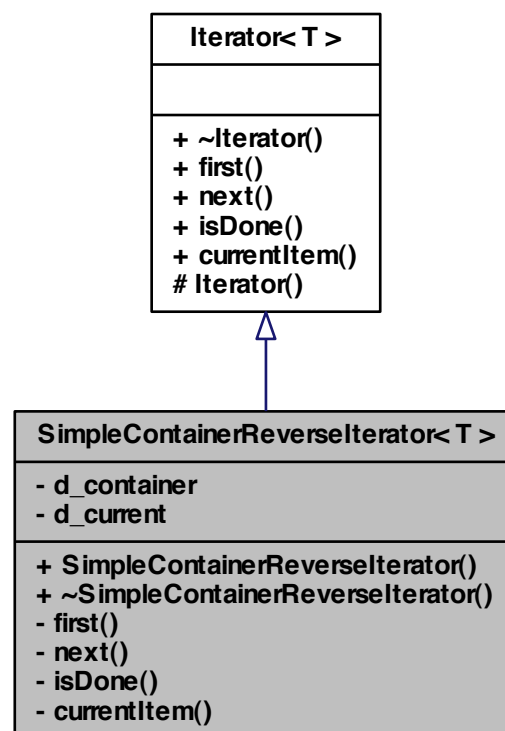
## 5.60 SimpleContainerReverselIterator< T> Class Template Reference

class [SimpleContainerReverselIterator](#) -

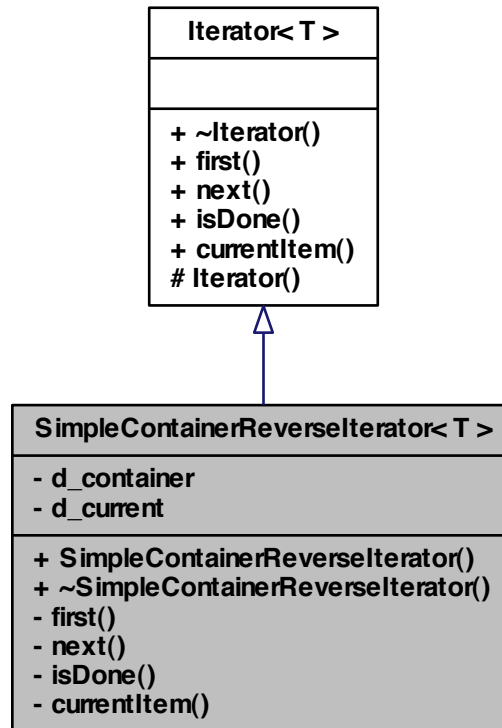
`#include <SimpleContainerReverseIterator.h>`



Inheritance diagram for SimpleContainerReverseliterator< T >:



Collaboration diagram for SimpleContainerReverselIterator< T >:



### Public Member Functions

- [SimpleContainerReverselIterator \(\)](#)
- [~SimpleContainerReverselIterator \(\)](#)

### Private Member Functions

- void [first \(\)](#)
- void [next \(\)](#)
- bool [isDone \(\)](#)
- T [currentItem \(\)](#)

### Private Attributes

- [Container< T > \\* d\\_container](#)
- [int d\\_current](#)

### Friends

- [class SimpleContainer< T >](#)

### 5.60.1 Detailed Description

`template<typename T>class SimpleContainerReverseliterator< T >`

class [SimpleContainerReverseliterator](#) -

Definition at line 6 of file SimpleContainerReverseliterator.h.

### 5.60.2 Constructor & Destructor Documentation

5.60.2.1 `template<typename T > SimpleContainerReverseliterator< T >::SimpleContainerReverseliterator ( )`

5.60.2.2 `template<typename T > SimpleContainerReverseliterator< T >::~~SimpleContainerReverseliterator ( )`

### 5.60.3 Member Function Documentation

5.60.3.1 `template<typename T > T SimpleContainerReverseliterator< T >::currentItem ( )` [private, virtual]

Implements [literator< T >](#).

5.60.3.2 `template<typename T > void SimpleContainerReverseliterator< T >::first ( )` [private, virtual]

Implements [literator< T >](#).

5.60.3.3 `template<typename T > bool SimpleContainerReverseliterator< T >::isDone ( )` [private, virtual]

Implements [literator< T >](#).

5.60.3.4 `template<typename T > void SimpleContainerReverseliterator< T >::next ( )`  
`[private, virtual]`

Implements [Iterator< T >](#).

## 5.60.4 Friends And Related Function Documentation

5.60.4.1 `template<typename T > friend class SimpleContainer< T > [friend]`

Definition at line 13 of file SimpleContainerReverseliterator.h.

## 5.60.5 Member Data Documentation

5.60.5.1 `template<typename T > Container<T>* SimpleContainerReverseliterator<`  
`T>::d_container [private]`

Definition at line 9 of file SimpleContainerReverseliterator.h.

5.60.5.2 `template<typename T > int SimpleContainerReverseliterator< T >::d_current`  
`[private]`

Definition at line 10 of file SimpleContainerReverseliterator.h.

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

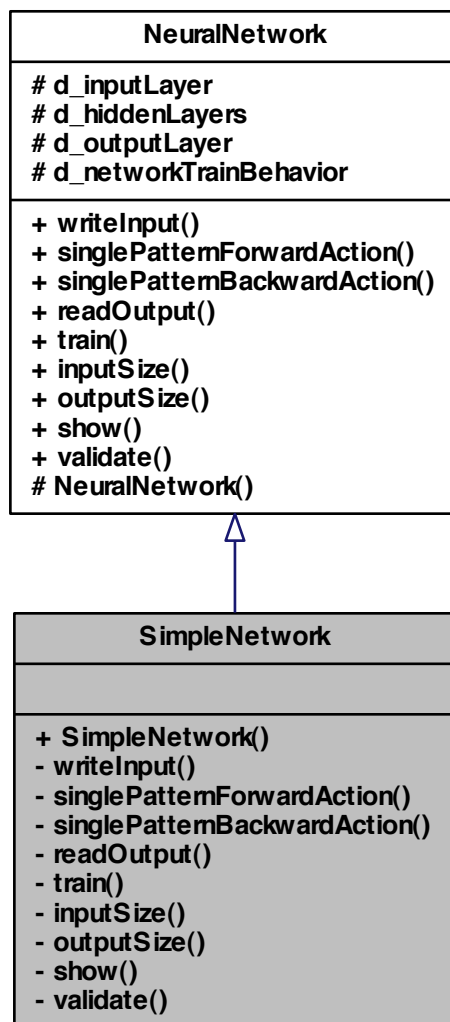
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.61 SimpleNetwork Class Reference

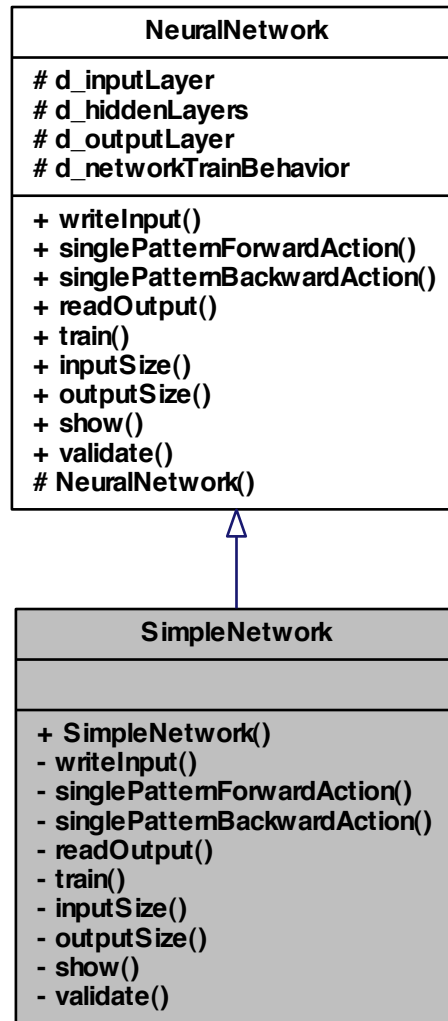
class [SimpleNetwork](#) -

```
#include <SimpleNetwork.h>
```

Inheritance diagram for SimpleNetwork:



Collaboration diagram for SimpleNetwork:



## Public Member Functions

- [SimpleNetwork](#) ([NeuralFactory](#) &neuralFactory)

## Private Member Functions

- void [writeInput](#) (std::vector< double >::iterator &iterator)
- void [singlePatternForwardAction](#) ()
- void [singlePatternBackwardAction](#) ()
- void [readOutput](#) (std::vector< double >::iterator &iterator)
- Rcpp::List [train](#) (Rcpp::List parameterList)
- size\_type [inputSize](#) ()
- size\_type [outputSize](#) ()
- void [show](#) ()
- bool [validate](#) ()

### 5.61.1 Detailed Description

class [SimpleNetwork](#) -

Definition at line 5 of file SimpleNetwork.h.

### 5.61.2 Constructor & Destructor Documentation

#### 5.61.2.1 SimpleNetwork::SimpleNetwork ( [NeuralFactory](#) & *neuralFactory* )

Definition at line 16 of file SimpleNetwork.cpp.

```
SimpleNetwork (neuralFactory) :  
{  
}  
}
```

### 5.61.3 Member Function Documentation

#### 5.61.3.1 size\_type SimpleNetwork::inputSize ( ) [private, virtual]

Implements [NeuralNetwork](#).

Definition at line 108 of file SimpleNetwork.cpp.

References [NeuralNetwork::d\\_inputLayer](#).

Referenced by [writeInput\(\)](#).

```
{  
    return d_inputLayer->size();  
}
```

Here is the caller graph for this function:



#### 5.61.3.2 `size_type SimpleNetwork::outputSize ( )` [private, virtual]

Implements [NeuralNetwork](#).

Definition at line 114 of file `SimpleNetwork.cpp`.

References `NeuralNetwork::d_outputLayer`.

Referenced by `readOutput()`.

```
{  
    return d_outputLayer->size();  
}
```

Here is the caller graph for this function:



#### 5.61.3.3 `void SimpleNetwork::readOutput ( std::vector< double >::iterator & iterator )` [private, virtual]

Implements [NeuralNetwork](#).

Definition at line 88 of file `SimpleNetwork.cpp`.

References `NeuralNetwork::d_outputLayer`, `outputSize()`, and `size_type`.

```
{
```



```

size_type nOutputs(outputSize());
for (size_type i = 0; i < nOutputs; i++)
{
    *iterator++ = d_outputLayer->at(i)->getOutput();
}

```

Here is the call graph for this function:



#### 5.61.3.4 void SimpleNetwork::show( ) [private, virtual]

Implements [NeuralNetwork](#).

Definition at line 120 of file SimpleNetwork.cpp.

References [NeuralNetwork::d\\_hiddenLayers](#), [NeuralNetwork::d\\_inputLayer](#), and [NeuralNetwork::d\\_outputLayer](#).

```

{
    Rprintf("\n\n=====\\n");
    Rprintf("      Input Layer");
    Rprintf("\n=====\\n");
    d_inputLayer->show();

    Rprintf("\n\n=====\\n");
    Rprintf("      Hidden Layers");
    Rprintf("\n=====\\n");
    d_hiddenLayers->show();

    Rprintf("\n\n=====\\n");
    Rprintf("      Output Layer");
    Rprintf("\n=====\\n");
    d_outputLayer->show();
    Rprintf("\n=====\\n");
}

```

#### 5.61.3.5 void SimpleNetwork::singlePatternBackwardAction( ) [private, virtual]

Implements [NeuralNetwork](#).

Definition at line 64 of file SimpleNetwork.cpp.

References [NeuralNetwork::d\\_hiddenLayers](#), and [NeuralNetwork::d\\_outputLayer](#).

```
{
    // Output Layers
    boost::shared_ptr < Iterator<NeuronPtr> > neuronIterator(d_outputLayer->createReverseIterator());
    for (neuronIterator->first(); !neuronIterator->isDone(); neuronIterator->next())
    {
        neuronIterator->currentItem()->singlePatternBackwardAction();
    }

    // Hidden Layers
    boost::shared_ptr < Iterator<LayerPtr> > layerIterator(d_hiddenLayers->createReverseIterator());
    for (layerIterator->first(); !layerIterator->isDone(); layerIterator->next())
    {
        boost::shared_ptr < Iterator<NeuronPtr> > neuronIterator( layerIterator->currentItem()->createReverseIterator());
        for (neuronIterator->first(); !neuronIterator->isDone(); neuronIterator->next())
        {
            neuronIterator->currentItem()->singlePatternBackwardAction();
        }
    }
}
```

#### 5.61.3.6 void SimpleNetwork::singlePatternForwardAction ( ) [private, virtual]

Implements [NeuralNetwork](#).

Definition at line 35 of file SimpleNetwork.cpp.

References [NeuralNetwork::d\\_hiddenLayers](#), and [NeuralNetwork::d\\_outputLayer](#).

```
{
    // Hidden Layers
    boost::shared_ptr < Iterator<LayerPtr> > layerIterator(d_hiddenLayers->createIterator());

    for (layerIterator->first(); !layerIterator->isDone(); layerIterator->next())
    {
        boost::shared_ptr < Iterator<NeuronPtr> > neuronIterator(
            layerIterator->currentItem()->createIterator());
        for (neuronIterator->first(); !neuronIterator->isDone(); neuronIterator->next())
        {
            neuronIterator->currentItem()->singlePatternForwardAction();
        }
    }

    // Output Layers
    boost::shared_ptr < Iterator<NeuronPtr> > neuronIterator(
        d_outputLayer->createIterator());
    for (neuronIterator->first(); !neuronIterator->isDone(); neuronIterator->next())
    {
    }
```

```
{
    neuronIterator->currentItem()->singlePatternForwardAction();
}
}
```

#### 5.61.3.7 Rcpp::List SimpleNetwork::train ( Rcpp::List *parameterList* ) [private, virtual]

Implements [NeuralNetwork](#).

Definition at line 98 of file SimpleNetwork.cpp.

References [NeuralNetwork::d\\_networkTrainBehavior](#).

```
{
    // TODO check train behavior and change it if need be
    // TODO check cost function and change it if need be

    return d_networkTrainBehavior->train(parameterList);
}
```

#### 5.61.3.8 bool SimpleNetwork::validate ( ) [private, virtual]

Implements [NeuralNetwork](#).

Definition at line 141 of file SimpleNetwork.cpp.

References [NeuralNetwork::d\\_hiddenLayers](#), [NeuralNetwork::d\\_inputLayer](#), and [NeuralNetwork::d\\_outputLayer](#).

```
{
    d_inputLayer->validate();
    d_hiddenLayers->validate();
    d_outputLayer->validate();
    return true;
}
```

#### 5.61.3.9 void SimpleNetwork::writeInput ( std::vector< double >::iterator & *iterator* ) [private, virtual]

Implements [NeuralNetwork](#).

Definition at line 23 of file SimpleNetwork.cpp.

References [NeuralNetwork::d\\_inputLayer](#), [inputSize\(\)](#), and [size\\_type](#).

```
{
    size_type nInputs(inputSize());
    for (size_type i = 0; i < nInputs; i++)
    {
        d_inputLayer->at(i)->setOutput(*iterator++);
    }
}
```

Here is the call graph for this function:

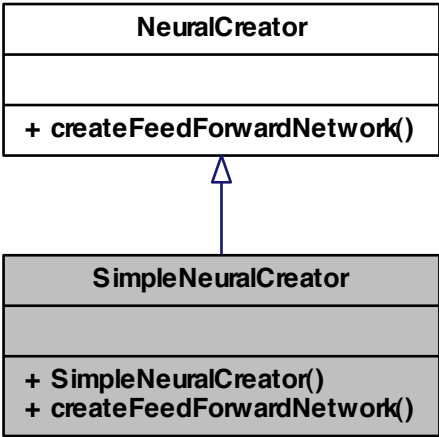


The documentation for this class was generated from the following files:

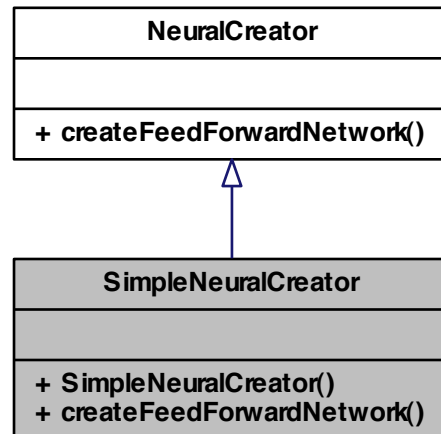
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[SimpleNetw](#)

5.62 SimpleNeuralCreator Class Reference

class [SimpleNeuralCreator](#) -  
`#include <SimpleNeuralCreator.h>`  
Inheritance diagram for SimpleNeuralCreator:



Collaboration diagram for SimpleNeuralCreator:



### Public Member Functions

- [SimpleNeuralCreator](#) ()
- [NeuralNetworkPtr createFeedForwardNetwork](#) (std::vector< int > numberOfNeurons, [NeuralFactory](#) &hiddenLayersFactory, [NeuralFactory](#) &outputLayerFactory)

#### 5.62.1 Detailed Description

class [SimpleNeuralCreator](#) -

Definition at line 5 of file `SimpleNeuralCreator.h`.

#### 5.62.2 Constructor & Destructor Documentation

##### 5.62.2.1 SimpleNeuralCreator::SimpleNeuralCreator ( )

Definition at line 19 of file `SimpleNeuralCreator.cpp`.

```
{  
}
```

### 5.62.3 Member Function Documentation

#### 5.62.3.1 `NeuralNetworkPtr SimpleNeuralCreator::createFeedForwardNetwork ( std::vector<int> > numberOfNeurons, NeuralFactory & hiddenLayersFactory, NeuralFactory & outputLayerFactory )` [virtual]

Implements [NeuralCreator](#).

Definition at line 24 of file SimpleNeuralCreator.cpp.

References [NeuralFactory::makeLayer\(\)](#), [NeuralFactory::makeNeuralNetwork\(\)](#), and [NeuralFactory::makeNeuron\(\)](#).

```
{
    NeuralNetworkPtr neuralNetworkPtr(outputLayerFactory.makeNeuralNetwork(outputLayerFactory));
    NeuronPtr neuronPtr;

    if (numberOfNeurons.size() <= 2)
    {
        throw std::range_error(
            "[C++ CreateFeedForwardNetwork::validate]: Error, number of layers lower than 3.");
    }

    Handler neuronId = 1;

    //=====
    // Calculation of the total amount of parameters
    //=====
    int totalAmountOfParameters = 0;

    std::vector<int>::iterator itr1 = numberOfNeurons.begin();
    int totalNumberOfNeurons = *itr1;
    for (std::vector<int>::iterator itr2 = 1+itr1; itr2 != numberOfNeurons.end(); ++itr2, ++itr1)
    {
        totalNumberOfNeurons += *itr2;
        totalAmountOfParameters += (*itr2) * (*itr1); //integer multiplication
    }
    totalAmountOfParameters += totalNumberOfNeurons;

    //=====
    // Neuron insertion
    //=====

    //Input Layer
    for (int i = 0; i < numberOfNeurons.at(0); ++i)
    {
        neuronPtr = outputLayerFactory.makeNeuron(neuronId++); // It's irrelevant whether to use outputLayerFactory or hiddenLayersFactory as inputFactory
        neuralNetworkPtr->d_inputLayer->push_back(neuronPtr);
    }

    // Hidden layers

    for (int i = 0; i < numberOfNeurons.at(1); ++i)
    {
        neuronPtr = hiddenLayersFactory.makeNeuron(neuronId++, neuralNetworkPtr->d
```

```

        _inputLayer->createIterator(), totalAmountOfParameters);
        neuralNetworkPtr->d_hiddenLayers->at(0)->push_back(neuronPtr);
    }

    unsigned int layerItr = 2 ;
    for (; layerItr < (-1 + numberOfNeurons.size()); ++layerItr)
    {
        neuralNetworkPtr->d_hiddenLayers->push_back( hiddenLayersFactory.makeLayer(
        ) ) ;
        for (int i = 0; i < numberOfNeurons.at(layerItr); ++i)
        {
            neuronPtr = hiddenLayersFactory.makeNeuron(neuronId++, neuralNetworkPtr
            ->d_hiddenLayers->at(layerItr-2)->createIterator(), totalAmountOfParameters);
            neuralNetworkPtr->d_hiddenLayers->at(layerItr-1)->push_back(neuronPtr);

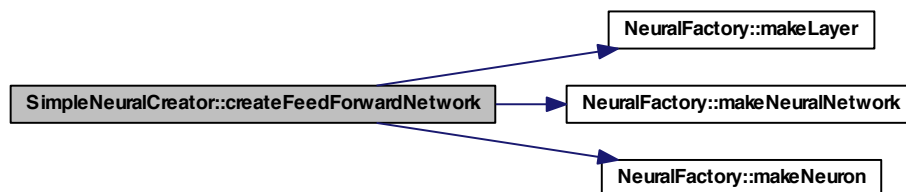
        }
    }

    //Output Layer
    for (int i = 0; i < numberOfNeurons.back(); ++i)
    {
        neuronPtr = outputLayerFactory.makeNeuron(neuronId++, neuralNetworkPtr->d_h
        iddenLayers->at(layerItr-2)->createIterator() , totalAmountOfParameters);
        neuralNetworkPtr->d_outputLayer->push_back(neuronPtr);
    }

    return neuralNetworkPtr;
}

```

Here is the call graph for this function:



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

- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[SimpleNe](#)
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[SimpleNeuralCreator.cp](#)

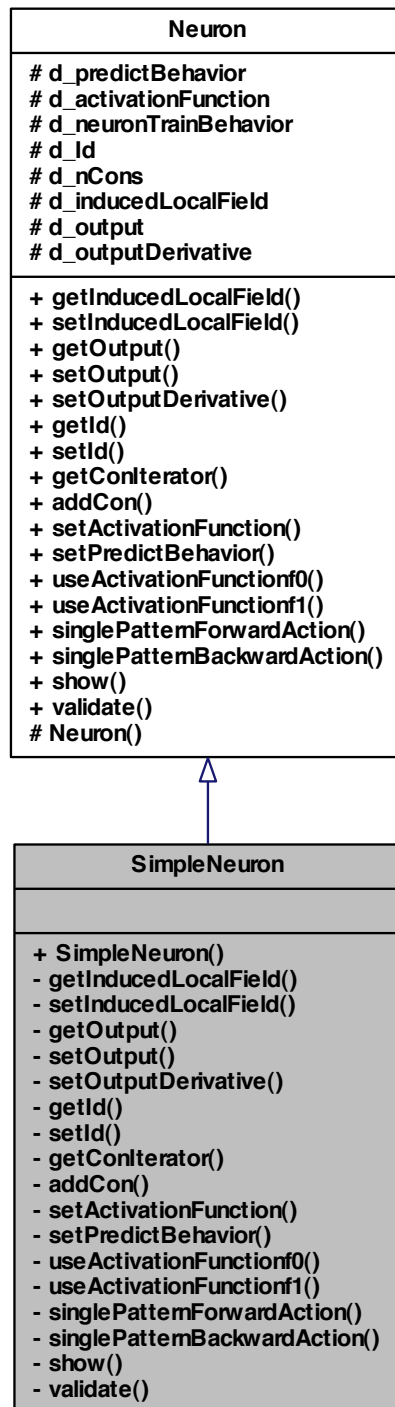
## 5.63 SimpleNeuron Class Reference

class [SimpleNeuron](#) -

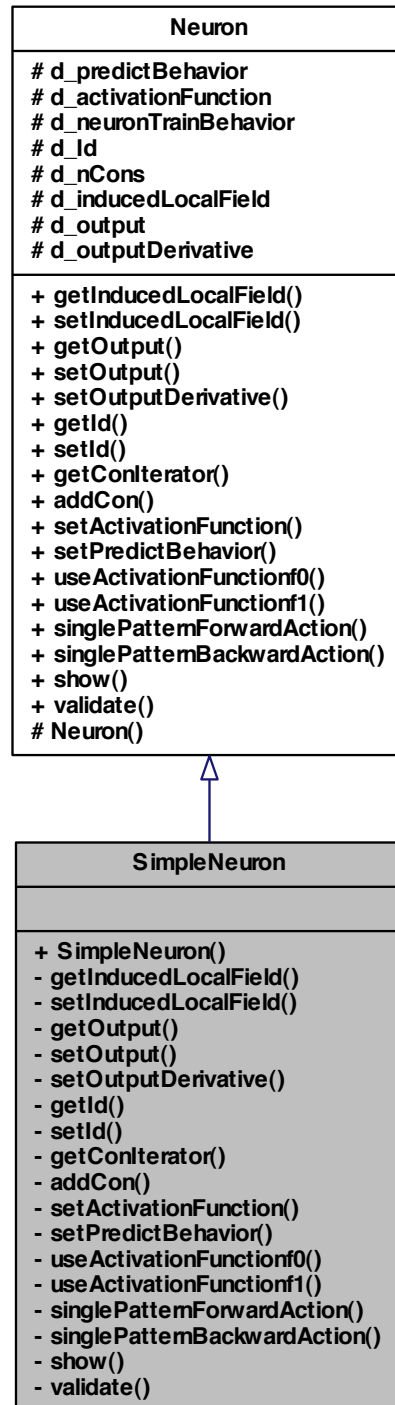
```
#include <SimpleNeuron.h>
```



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)
- void [setOutputDerivative](#) (double outputDerivative)
- [Handler](#) [getId](#) ()
- void [setId](#) ([Handler](#) Id)
- [ConIteratorPtr](#) [getConIterator](#) ()
- void [addCon](#) ([ConPtr](#) conPtr)
- void [setActivationFunction](#) ([ActivationFunctionPtr](#) activationFunctionPtr)
- void [setPredictBehavior](#) ([PredictBehaviorPtr](#) predictBehaviorPtr)
- double [useActivationFunction0](#) ()
- double [useActivationFunction1](#) ()
- void [singlePatternForwardAction](#) ()
- void [singlePatternBackwardAction](#) ()
- void [show](#) ()
- bool [validate](#) ()

### 5.63.1 Detailed Description

class [SimpleNeuron](#) -

Definition at line 5 of file SimpleNeuron.h.

### 5.63.2 Constructor & Destructor Documentation

#### 5.63.2.1 SimpleNeuron::SimpleNeuron ( [NeuralFactory](#) & *neuralFactory* )

Definition at line 18 of file SimpleNeuron.cpp.

```
SimpleNeuron (NeuralFactory &neuralFactory) :  
{  
}  
}
```

### 5.63.3 Member Function Documentation

#### 5.63.3.1 void SimpleNeuron::addCon ( ConPtr conPtr ) [private, virtual]

Implements [Neuron](#).

Definition at line 74 of file SimpleNeuron.cpp.

References [Neuron::d\\_nCons](#).

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

#### 5.63.3.2 ConIteratorPtr SimpleNeuron::getConIterator ( ) [private, virtual]

Implements [Neuron](#).

Definition at line 68 of file SimpleNeuron.cpp.

References [Neuron::d\\_nCons](#).

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

#### 5.63.3.3 Handler SimpleNeuron::getId ( ) [private, virtual]

Implements [Neuron](#).

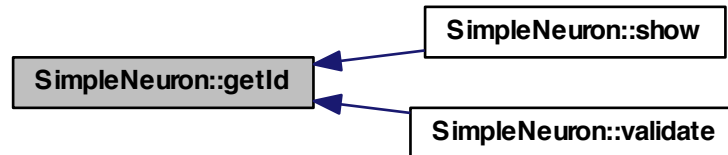
Definition at line 56 of file SimpleNeuron.cpp.

References [Neuron::d\\_Id](#).

Referenced by [show\(\)](#), and [validate\(\)](#).

```
{
    return d_Id;
}
```

Here is the caller graph for this function:



#### 5.63.3.4 `double SimpleNeuron::getInducedLocalField ( ) [private, virtual]`

Implements [Neuron](#).

Definition at line 24 of file `SimpleNeuron.cpp`.

References `Neuron::d_inducedLocalField`.

```
{  
    return d_inducedLocalField;  
}
```

#### 5.63.3.5 `double SimpleNeuron::getOutput ( ) [private, virtual]`

Implements [Neuron](#).

Definition at line 36 of file `SimpleNeuron.cpp`.

References `Neuron::d_output`.

```
{  
    return d_output;  
}
```

#### 5.63.3.6 `void SimpleNeuron::setActivationFunction ( ActivationFunctionPtr activationFunctionPtr ) [private, virtual]`

Implements [Neuron](#).

Definition at line 80 of file `SimpleNeuron.cpp`.

References `Neuron::d_activationFunction`.

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

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

Implements [Neuron](#).

Definition at line 62 of file SimpleNeuron.cpp.

References [Neuron::d\\_Id](#).

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

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

Implements [Neuron](#).

Definition at line 30 of file SimpleNeuron.cpp.

References [Neuron::d\\_inducedLocalField](#).

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

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

Implements [Neuron](#).

Definition at line 42 of file SimpleNeuron.cpp.

References [Neuron::d\\_output](#).

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

**5.63.3.10** void SimpleNeuron::setOutputDerivative ( double *outputDerivative* ) [private,  
virtual]

Implements [Neuron](#).

Definition at line 50 of file SimpleNeuron.cpp.

References [Neuron::d\\_outputDerivative](#).

```
{
    d_outputDerivative = outputDerivative;
}
```

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

Implements [Neuron](#).

Definition at line 86 of file SimpleNeuron.cpp.

References [Neuron::d\\_predictBehavior](#).

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

**5.63.3.12** `void SimpleNeuron::show ( )` `[private, virtual]`

Implements [Neuron](#).

Definition at line 122 of file SimpleNeuron.cpp.

References [Neuron::d\\_nCons](#), [Neuron::d\\_output](#), [Neuron::d\\_predictBehavior](#), and [getId\(\)](#).

```
{
    if (d_nCons->size() == 0)
    {
        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-----");
        Rprintf("\n output: %lf", d_output);
        Rprintf("\n-----");
    }
    else
    {
        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-----");
    }
}
```

```

        d_predictBehavior->show();

        Rprintf("\n output: %lf", d_output);
        Rprintf("\n-----");
        d_nCons->show();
        Rprintf("\n-----");
    }
}

```

Here is the call graph for this function:



**5.63.3.13** void SimpleNeuron::singlePatternBackwardAction ( ) [private, virtual]

Implements [Neuron](#).

Definition at line 114 of file SimpleNeuron.cpp.

References [Neuron::d\\_neuronTrainBehavior](#).

```

{
    d_neuronTrainBehavior->singlePatternBackwardAction();
}

```

**5.63.3.14** void SimpleNeuron::singlePatternForwardAction ( ) [private, virtual]

Implements [Neuron](#).

Definition at line 108 of file SimpleNeuron.cpp.

References [Neuron::d\\_predictBehavior](#).

```

{
    d_predictBehavior->singlePatternForwardAction();
}

```

**5.63.3.15** double SimpleNeuron::useActivationFunction0 ( ) [private, virtual]

Implements [Neuron](#).



Definition at line 92 of file SimpleNeuron.cpp.

References `Neuron::d_activationFunction`.

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

#### 5.63.3.16 `double SimpleNeuron::useActivationFunctionf1 ( )` [private, virtual]

Implements [Neuron](#).

Definition at line 100 of file SimpleNeuron.cpp.

References `Neuron::d_activationFunction`.

```
{  
    return d_activationFunction->f1();  
}
```

#### 5.63.3.17 `bool SimpleNeuron::validate ( )` [private, virtual]

Implements [Neuron](#).

Definition at line 164 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:



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

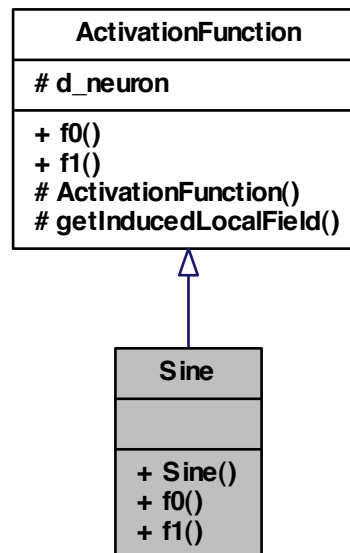
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuron.h`
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuron.cpp`

## 5.64 Sine Class Reference

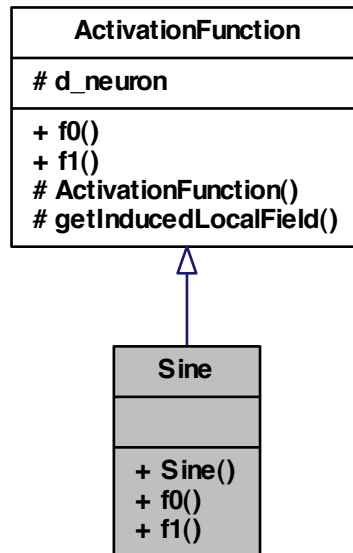
class [Sine](#) -

```
#include <Sine.h>
```

Inheritance diagram for Sine:



Collaboration diagram for Sine:



### Public Member Functions

- [Sine](#) ([NeuronPtr](#) neuronPtr)
- double [f0](#) ()
- double [f1](#) ()

#### 5.64.1 Detailed Description

class [Sine](#) -

Definition at line 5 of file Sine.h.

#### 5.64.2 Constructor & Destructor Documentation

5.64.2.1 [Sine::Sine](#) ( [NeuronPtr](#) neuronPtr )

#### 5.64.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

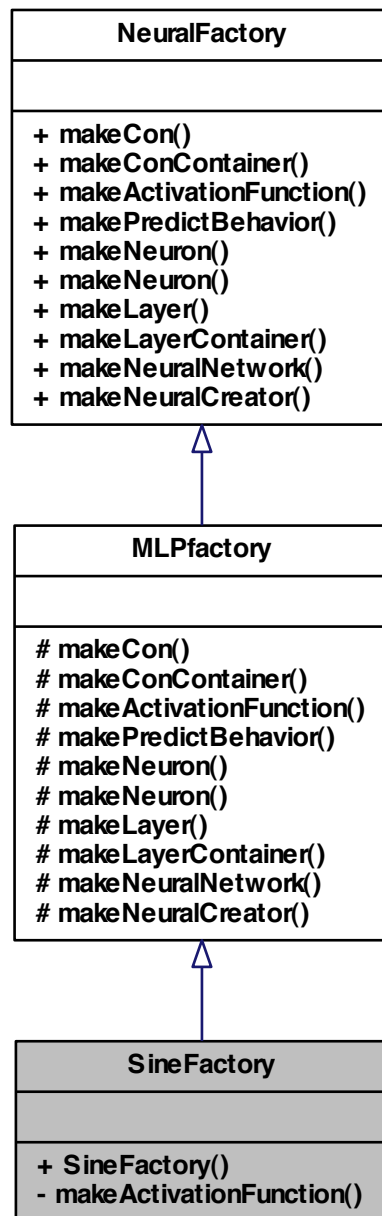
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.65 SineFactory Class Reference

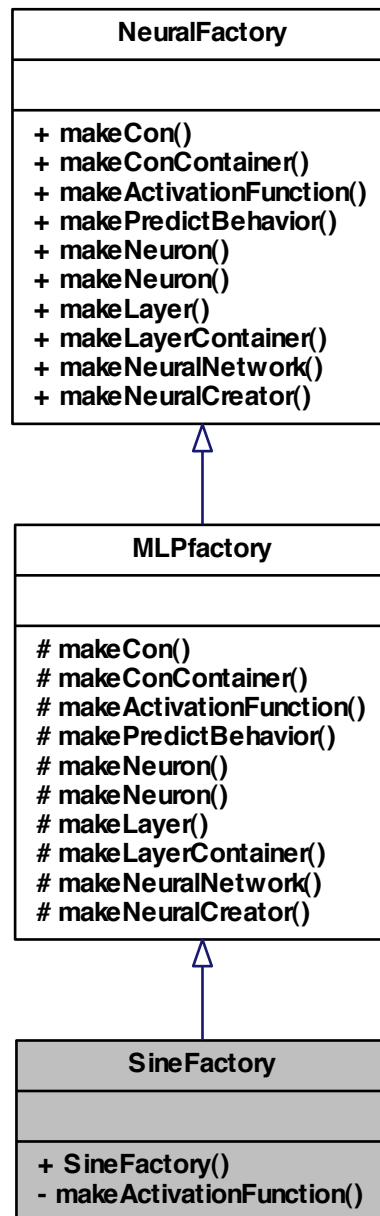
class [SineFactory](#) -

```
#include <SineFactory.h>
```

Inheritance diagram for SineFactory:



Collaboration diagram for SineFactory:



## Public Member Functions

- [SineFactory](#) ()

## Private Member Functions

- [ActivationFunctionPtr](#) [makeActivationFunction](#) ([NeuronPtr](#) neuronPtr)

### 5.65.1 Detailed Description

class [SineFactory](#) -

Definition at line 5 of file SineFactory.h.

### 5.65.2 Constructor & Destructor Documentation

5.65.2.1 [SineFactory::SineFactory](#) ( )

### 5.65.3 Member Function Documentation

5.65.3.1 [ActivationFunctionPtr](#) [SineFactory::makeActivationFunction](#) ( [NeuronPtr](#) *neuronPtr* ) [[private](#), [virtual](#)]

Implements [MLPfactory](#).

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

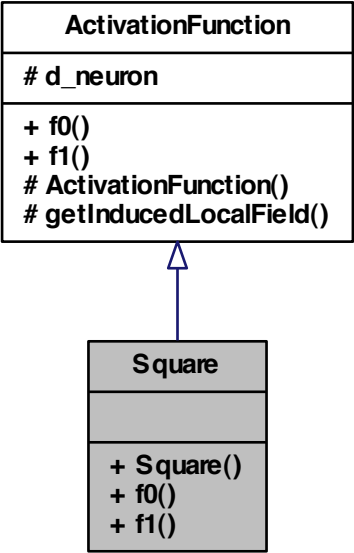
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[SineFactory.h](#)

## 5.66 Square Class Reference

class [Square](#) -

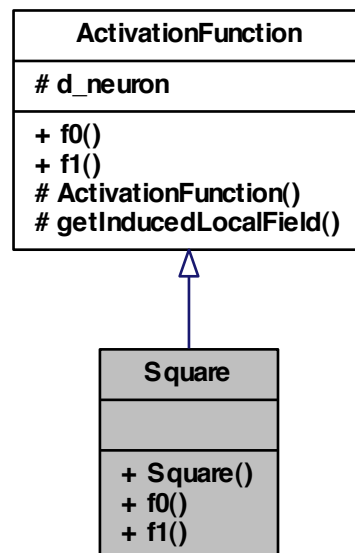
```
#include <Square.h>
```

Inheritance diagram for Square:





Collaboration diagram for Square:



### Public Member Functions

- [Square](#) ([NeuronPtr](#) neuronPtr)
- double [f0](#) ()
- double [f1](#) ()

### 5.66.1 Detailed Description

class [Square](#) -

Definition at line 5 of file Square.h.

### 5.66.2 Constructor & Destructor Documentation

5.66.2.1 [Square::Square](#) ( [NeuronPtr](#) neuronPtr )

### 5.66.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

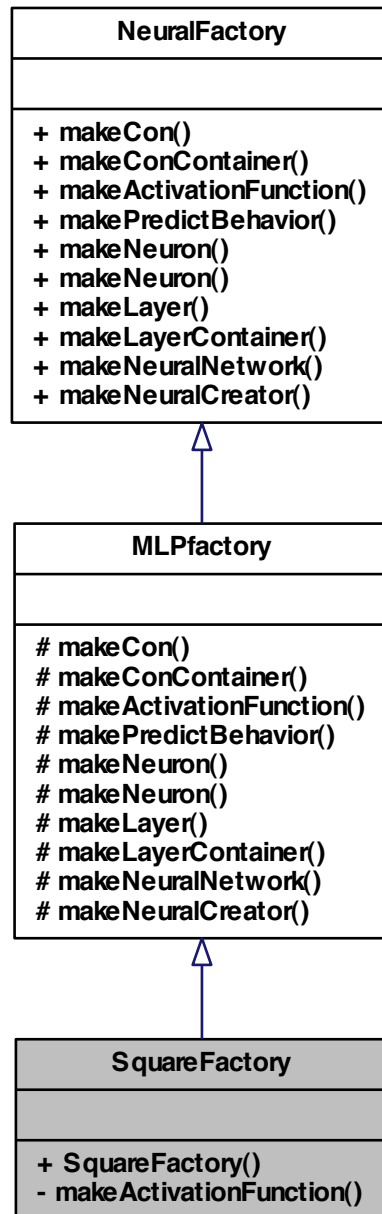
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.67 SquareFactory Class Reference

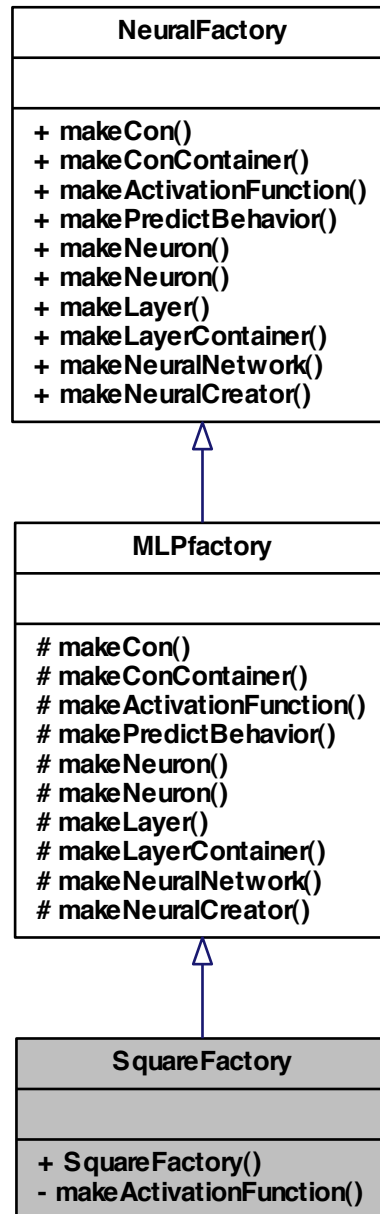
class [SquareFactory](#) -

```
#include <SquareFactory.h>
```

Inheritance diagram for SquareFactory:



Collaboration diagram for SquareFactory:



## Public Member Functions

- [SquareFactory](#) ()

## Private Member Functions

- [ActivationFunctionPtr](#) [makeActivationFunction](#) ([NeuronPtr](#) neuronPtr)

### 5.67.1 Detailed Description

class [SquareFactory](#) -

Definition at line 5 of file [SquareFactory.h](#).

### 5.67.2 Constructor & Destructor Documentation

5.67.2.1 [SquareFactory::SquareFactory](#) ( )

### 5.67.3 Member Function Documentation

5.67.3.1 [ActivationFunctionPtr](#) [SquareFactory::makeActivationFunction](#) ( [NeuronPtr](#) *neuronPtr* ) [[private](#), [virtual](#)]

Implements [MLPfactory](#).

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

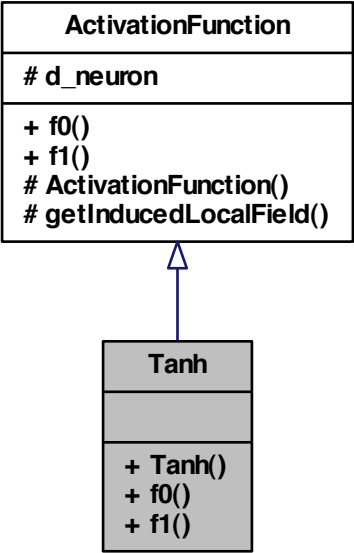
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SquareFa](#)

## 5.68 Tanh Class Reference

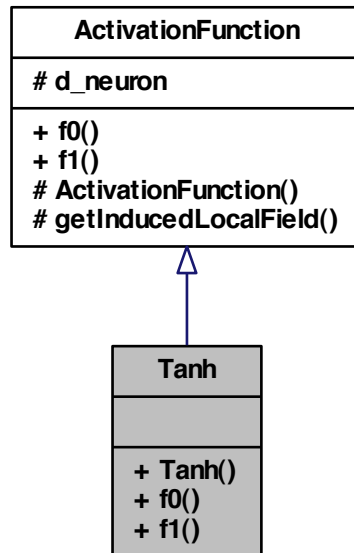
class [Tanh](#) -

```
#include <Tanh.h>
```

Inheritance diagram for Tanh:



Collaboration diagram for Tanh:



### Public Member Functions

- [Tanh](#) ([NeuronPtr](#) neuronPtr)
- double [f0](#) ()
- double [f1](#) ()

### 5.68.1 Detailed Description

class [Tanh](#) -

Definition at line 5 of file Tanh.h.

### 5.68.2 Constructor & Destructor Documentation

#### 5.68.2.1 Tanh::Tanh ( [NeuronPtr](#) neuronPtr )

Definition at line 15 of file Tanh.cpp.

```

: ActivationFunction(neuronPtr) {

```

```
}
```

### 5.68.3 Member Function Documentation

#### 5.68.3.1 `double Tanh::f0 ( ) [virtual]`

Implements [ActivationFunction](#).

Definition at line 19 of file Tanh.cpp.

References [ActivationFunction::getInducedLocalField\(\)](#).

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

Here is the call graph for this function:



#### 5.68.3.2 `double Tanh::f1 ( ) [virtual]`

Implements [ActivationFunction](#).

Definition at line 24 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:

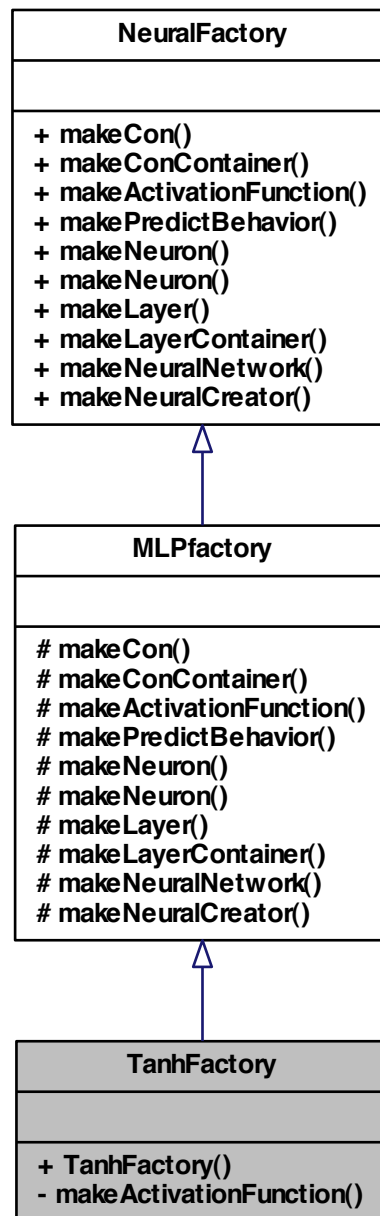
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[Tanh.h](#)
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[Tanh.cpp](#)

## 5.69 TanhFactory Class Reference

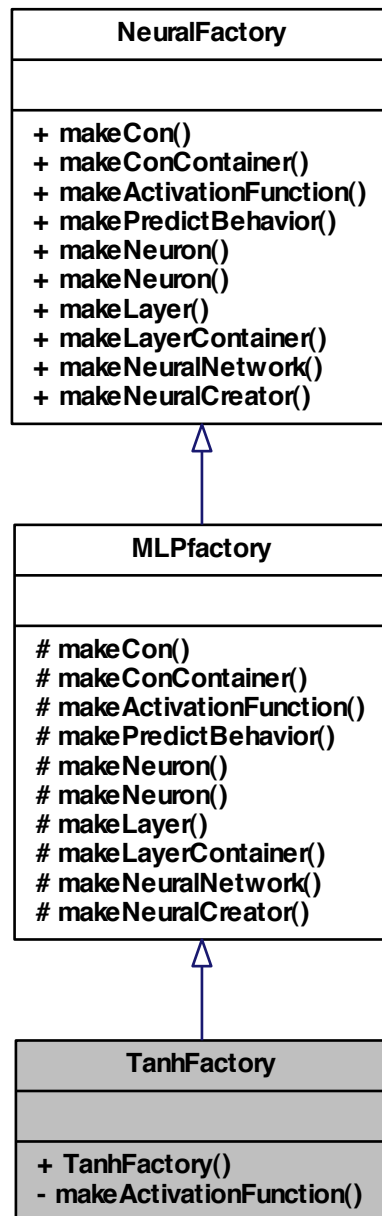
class [TanhFactory](#) -

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

Inheritance diagram for TanhFactory:



Collaboration diagram for TanhFactory:



## Public Member Functions

- [TanhFactory](#) ()

## Private Member Functions

- [ActivationFunctionPtr](#) [makeActivationFunction](#) ([NeuronPtr](#) neuronPtr)

### 5.69.1 Detailed Description

class [TanhFactory](#) -

Definition at line 5 of file [TanhFactory.h](#).

### 5.69.2 Constructor & Destructor Documentation

#### 5.69.2.1 [TanhFactory::TanhFactory](#) ( )

Definition at line 17 of file [TanhFactory.cpp](#).

```
{  
}
```

### 5.69.3 Member Function Documentation

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

Implements [MLPfactory](#).

Definition at line 22 of file [TanhFactory.cpp](#).

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

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

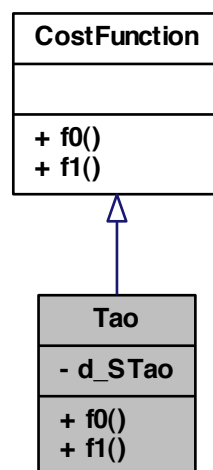
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead](#)
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/\[TanhFactor\]\(#\)](#)

## 5.70 Tao Class Reference

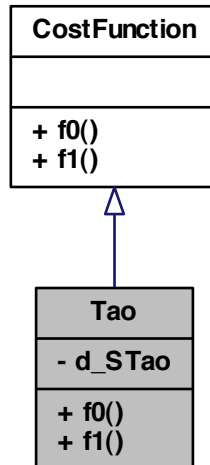
class [Tao](#) -

```
#include <Tao.h>
```

Inheritance diagram for Tao:



Collaboration diagram for Tao:



### Public Member Functions

- double `f0` (double output, double target)
- double `f1` (double output, double target)

### Private Attributes

- double `d_STao`

### 5.70.1 Detailed Description

class `Tao` -

Definition at line 5 of file `Tao.h`.

### 5.70.2 Member Function Documentation

5.70.2.1 `double Tao::f0 ( double output, double target )` `[virtual]`

Implements `CostFunction`.

5.70.2.2 double Tao::f1 ( double *output*, double *target* ) [virtual]

Implements [CostFunction](#).

5.70.3 Member Data Documentation

5.70.3.1 double Tao::d\_STao [private]

Definition at line 8 of file Tao.h.

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

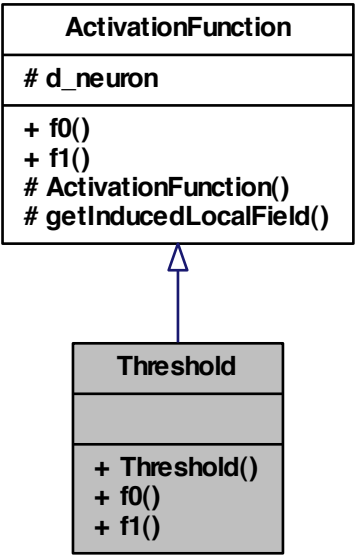
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Tao.h](#)

5.71 Threshold Class Reference

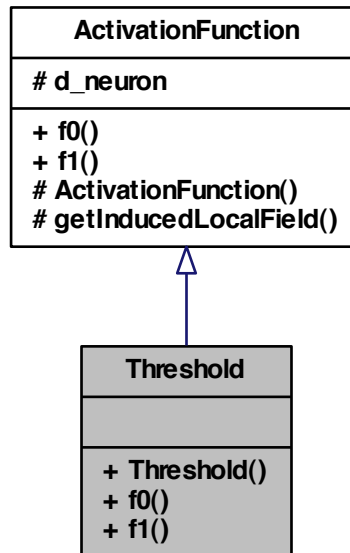
class [Threshold](#) -

```
#include <Threshold.h>
```

Inheritance diagram for Threshold:



Collaboration diagram for Threshold:



## Public Member Functions

- [Threshold](#) ([NeuronPtr](#) neuronPtr)
- double [f0](#) ()
- double [f1](#) ()

### 5.71.1 Detailed Description

class [Threshold](#) -

Definition at line 5 of file Threshold.h.

### 5.71.2 Constructor & Destructor Documentation

5.71.2.1 [Threshold::Threshold](#) ( [NeuronPtr](#) neuronPtr )

### 5.71.3 Member Function Documentation



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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

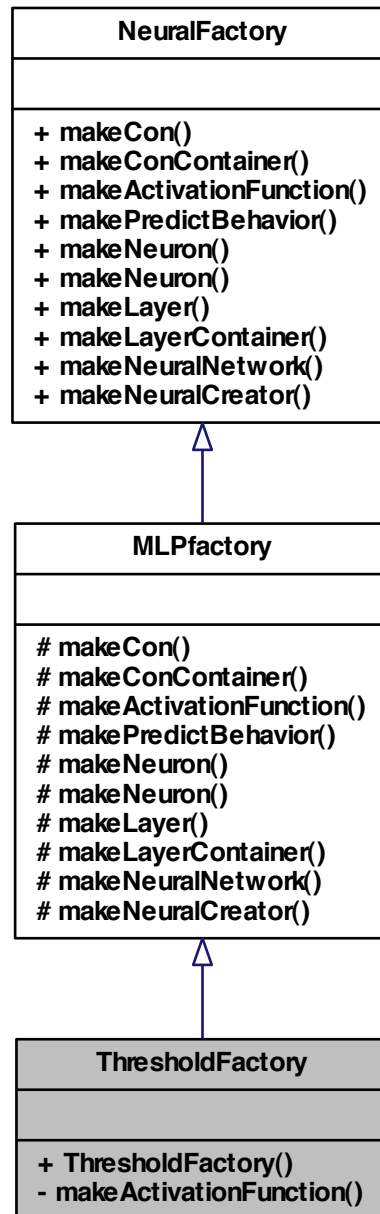
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Threshold`

## 5.72 ThresholdFactory Class Reference

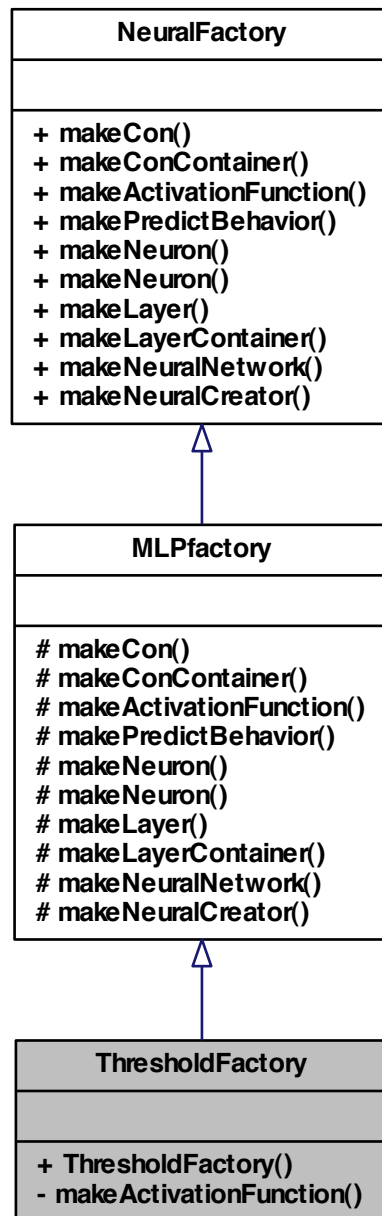
class [ThresholdFactory](#) -

```
#include <ThresholdFactory.h>
```

Inheritance diagram for ThresholdFactory:



Collaboration diagram for ThresholdFactory:



## Public Member Functions

- [ThresholdFactory](#) ()

## Private Member Functions

- [ActivationFunctionPtr](#) [makeActivationFunction](#) ([NeuronPtr](#) neuronPtr)

### 5.72.1 Detailed Description

class [ThresholdFactory](#) -

Definition at line 5 of file ThresholdFactory.h.

### 5.72.2 Constructor & Destructor Documentation

#### 5.72.2.1 [ThresholdFactory::ThresholdFactory](#) ( )

### 5.72.3 Member Function Documentation

#### 5.72.3.1 [ActivationFunctionPtr](#) [ThresholdFactory::makeActivationFunction](#) ( [NeuronPtr](#) *neuronPtr* ) [[private](#), [virtual](#)]

Implements [MLPfactory](#).

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

- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

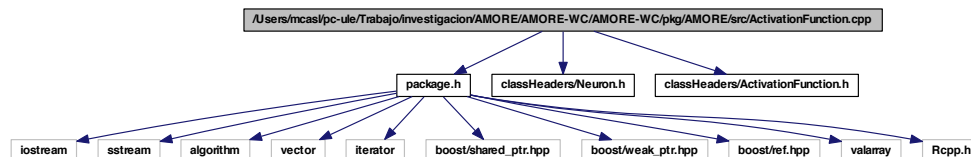
## Chapter 6

# File Documentation

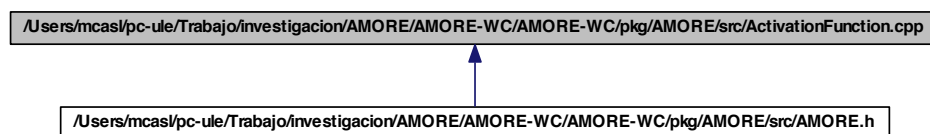
### 6.1 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/ActivationFunction.cpp File Reference

```
#include "package.h"  
#include "classHeaders/Neuron.h"  
#include "classHeaders/ActivationFunction.h"
```

Include dependency graph for ActivationFunction.cpp:



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



## 6.2 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/ADAPTgdNetworkTrainBehavior.cpp File Reference

## 6.3 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/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/ref.hpp>
#include <valarray>
#include <Rcpp.h>
#include "classHeaders/Connection.h"
#include "classHeaders/ActivationFunction.h"
#include "classHeaders/Tanh.h"
#include "classHeaders/Identity.h"
#include "classHeaders/PredictBehavior.h"
#include "classHeaders/MLPBehavior.h"
#include "classHeaders/NeuronTrainBehavior.h"
#include "classHeaders/NetworkTrainBehavior.h"
#include "classHeaders/Neuron.h"
#include "classHeaders/SimpleNeuron.h"
#include "classHeaders/NeuralFactory.h"
#include "classHeaders/MLPfactory.h"
#include "classHeaders/TanhFactory.h"
#include "classHeaders/IdentityFactory.h"
#include "classHeaders/NeuralNetwork.h"
#include "classHeaders/SimpleNetwork.h"
#include "classHeaders/NeuralCreator.h"
```

---

```
#include "classHeaders/SimpleNeuralCreator.h"

#include "classHeaders/NetworkRinterface.h"

#include "classHeaders/Container.h"

#include "classHeaders/SimpleContainer.h"

#include "classHeaders/Iterator.h"

#include "classHeaders/SimpleContainerIterator.h"

#include "classHeaders/SimpleContainerReverseIterator.h"

#include "Connection.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 "NetworkRinterface.cpp"

#include "RcppModules.cpp"
```

## Defines

- #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< NetworkTrainBehavior > NetworkTrainBehaviorPtr`
- `typedef boost::shared_ptr< NeuronTrainBehavior > NeuronTrainBehaviorPtr`
- `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< LayerPtr > > LayerContainerPtr`
- `typedef boost::shared_ptr< Container< ConPtr > > ConContainerPtr`
- `typedef boost::shared_ptr< NeuralFactory > NeuralFactoryPtr`
- `typedef boost::shared_ptr< NeuralCreator > NeuralCreatorPtr`
- `typedef boost::weak_ptr< NeuralNetwork > NeuralNetworkWeakPtr`
- `typedef boost::weak_ptr< Neuron > NeuronWeakPtr`

### 6.3.1 Define Documentation

#### 6.3.1.1 `#define size_type unsigned int`

Definition at line 86 of file `AMORE.h`.

Referenced by `SimpleNetwork::readOutput()`, and `SimpleNetwork::writeInput()`.

### 6.3.2 Typedef Documentation

#### 6.3.2.1 `typedef boost::shared_ptr<ActivationFunction> ActivationFunctionPtr`

Definition at line 98 of file `AMORE.h`.

#### 6.3.2.2 `typedef boost::reference_wrapper<PredictBehavior> ActivationFunctionRef`

Definition at line 92 of file `AMORE.h`.

#### 6.3.2.3 `typedef boost::shared_ptr< Container<ConPtr> > ConContainerPtr`

Definition at line 112 of file `AMORE.h`.

#### 6.3.2.4 `typedef boost::shared_ptr< Iterator<ConPtr> > ConIteratorPtr`

Definition at line 108 of file `AMORE.h`.

#### 6.3.2.5 `typedef boost::shared_ptr<Con> ConPtr`

Definition at line 103 of file `AMORE.h`.



---

6.3.2.6 `typedef int Handler`

Definition at line 89 of file AMORE.h.

6.3.2.7 `typedef boost::shared_ptr< Container< LayerPtr > > LayerContainerPtr`

Definition at line 111 of file AMORE.h.

6.3.2.8 `typedef boost::shared_ptr< Container< NeuronPtr > > LayerPtr`

Definition at line 110 of file AMORE.h.

6.3.2.9 `typedef boost::shared_ptr< NetworkTrainBehavior> NetworkTrainBehaviorPtr`

Definition at line 100 of file AMORE.h.

6.3.2.10 `typedef boost::shared_ptr< NeuralCreator > NeuralCreatorPtr`

Definition at line 115 of file AMORE.h.

6.3.2.11 `typedef boost::shared_ptr< NeuralFactory > NeuralFactoryPtr`

Definition at line 114 of file AMORE.h.

6.3.2.12 `typedef boost::shared_ptr< NeuralNetwork> NeuralNetworkPtr`

Definition at line 104 of file AMORE.h.

6.3.2.13 `typedef boost::weak_ptr< NeuralNetwork> NeuralNetworkWeakPtr`

Definition at line 117 of file AMORE.h.

6.3.2.14 `typedef boost::shared_ptr< Iterator< NeuronPtr> > NeuronIteratorPtr`

Definition at line 107 of file AMORE.h.

6.3.2.15 `typedef boost::shared_ptr< Neuron> NeuronPtr`

Definition at line 102 of file AMORE.h.

6.3.2.16 `typedef boost::reference_wrapper<Neuron> NeuronRef`

Definition at line 95 of file AMORE.h.

6.3.2.17 `typedef boost::shared_ptr<NeuronTrainBehavior> NeuronTrainBehaviorPtr`

Definition at line 101 of file AMORE.h.

6.3.2.18 `typedef boost::weak_ptr<Neuron> NeuronWeakPtr`

Definition at line 118 of file AMORE.h.

6.3.2.19 `typedef boost::shared_ptr<PredictBehavior> PredictBehaviorPtr`

Definition at line 99 of file AMORE.h.

6.3.2.20 `typedef boost::reference_wrapper<PredictBehavior> PredictBehaviorRef`

Definition at line 93 of file AMORE.h.

6.3.2.21 `typedef boost::reference_wrapper<TrainingBehavior> TrainingBehaviorRef`

Definition at line 94 of file AMORE.h.

## 6.4 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ActivationFunction.h File Reference

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



### Classes

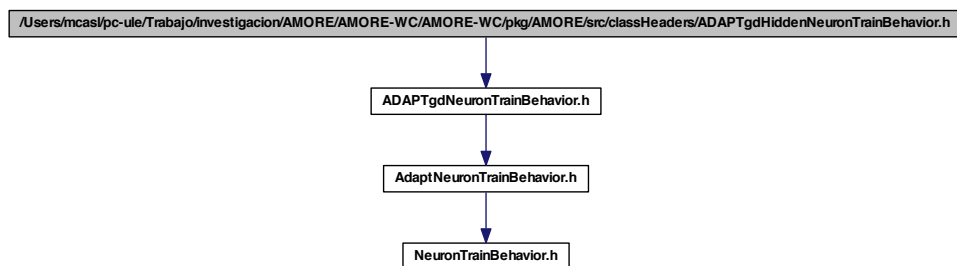
- class [ActivationFunction](#)  
*class [ActivationFunction](#) -*

6.5 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdHiddenNeuronTrainBehavior.h File Reference

263  
6.5 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdHiddenNeuronTrainBehavior.h  
File Reference

```
#include "ADAPTgdNeuronTrainBehavior.h"
```

Include dependency graph for ADAPTgdHiddenNeuronTrainBehavior.h:



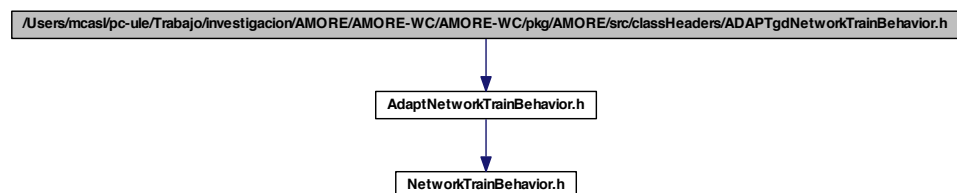
## Classes

- class [ADAPTgdHiddenNeuronTrainBehavior](#)  
class [ADAPTgdHiddenNeuronTrainBehavior](#) -

6.6 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdNetworkTrainBehavior.h  
File Reference

```
#include "AdaptNetworkTrainBehavior.h"
```

Include dependency graph for ADAPTgdNetworkTrainBehavior.h:



## Classes

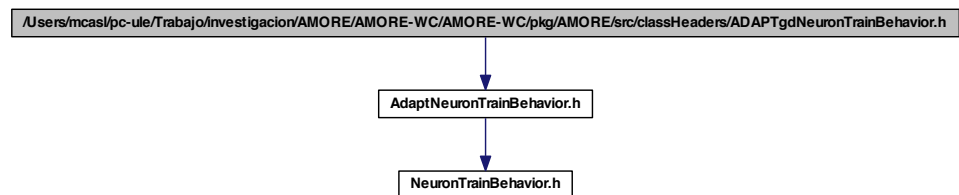
- class [ADAPTgdNetworkTrainBehavior](#)  
class [ADAPTgdNetworkTrainBehavior](#) -

## 6.7 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdNeuronTrainBehavior.h

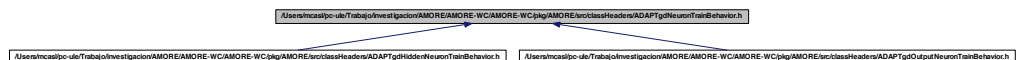
### File Reference

```
#include "AdaptNeuronTrainBehavior.h"
```

Include dependency graph for ADAPTgdNeuronTrainBehavior.h:



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



## Classes

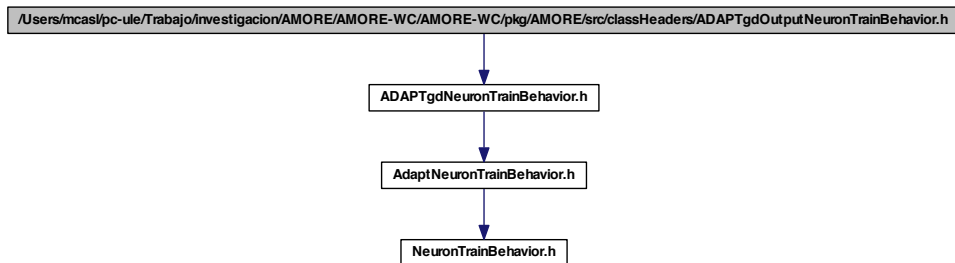
- class [ADAPTgdNeuronTrainBehavior](#)  
class [ADAPTgdNeuronTrainBehavior](#) -

## 6.8 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdOutputNeuronTrainBehavior.h

### File Reference

```
#include "ADAPTgdNeuronTrainBehavior.h"
```

6.9 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwmHiddenNeuronTrainBehavior.h  
File Reference 265  
Include dependency graph for ADAPTgdOutputNeuronTrainBehavior.h:



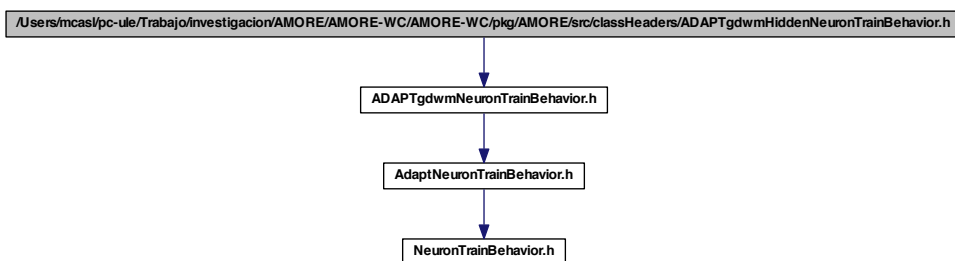
Classes

- class [ADAPTgdOutputNeuronTrainBehavior](#)  
class [ADAPTgdOutputNeuronTrainBehavior](#) -

6.9 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwmHiddenNeuronTrainBehavior.h  
File Reference

```
#include "ADAPTgdwmNeuronTrainBehavior.h"
```

Include dependency graph for ADAPTgdwmHiddenNeuronTrainBehavior.h:



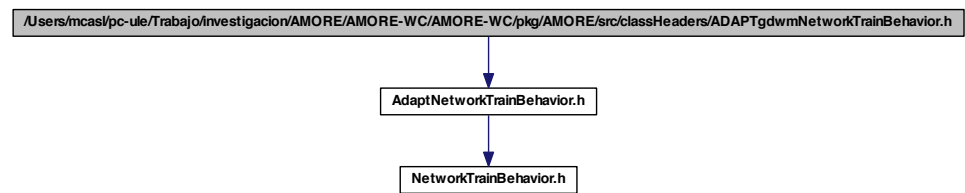
Classes

- class [ADAPTgdwmHiddenNeuronTrainBehavior](#)  
class [ADAPTgdwmHiddenNeuronTrainBehavior](#) -

## 6.10 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwmNetworkTrainBehavior.h File Reference

```
#include "AdaptNetworkTrainBehavior.h"
```

Include dependency graph for ADAPTgdwmNetworkTrainBehavior.h:



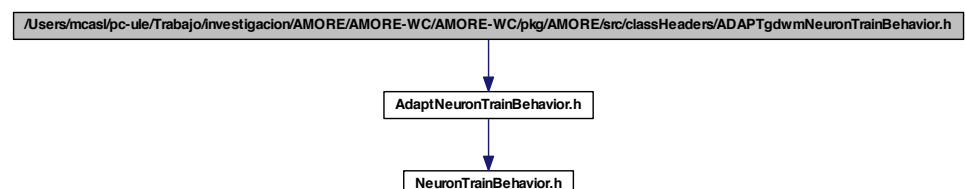
### Classes

- class [ADAPTgdwmNetworkTrainBehavior](#)  
*class ADAPTgdwmNetworkTrainBehavior -*

## 6.11 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwmNeuronTrainBehavior.h File Reference

```
#include "AdaptNeuronTrainBehavior.h"
```

Include dependency graph for ADAPTgdwmNeuronTrainBehavior.h:

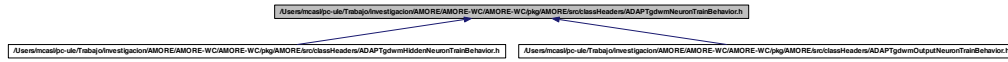


## 6.12 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwmOutputNeuronTrainBehavior.h

### File Reference

267

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



## Classes

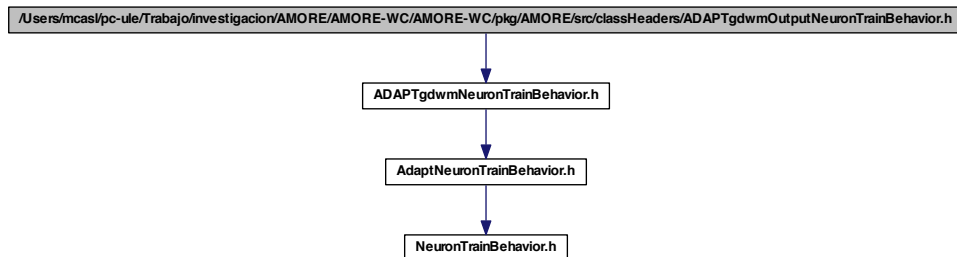
- class [ADAPTgdwmNeuronTrainBehavior](#)  
*class ADAPTgdwmNeuronTrainBehavior -*

## 6.12 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwmOutputNeuronTrainBehavior.h

### File Reference

```
#include "ADAPTgdwmNeuronTrainBehavior.h"
```

Include dependency graph for ADAPTgdwmOutputNeuronTrainBehavior.h:



## Classes

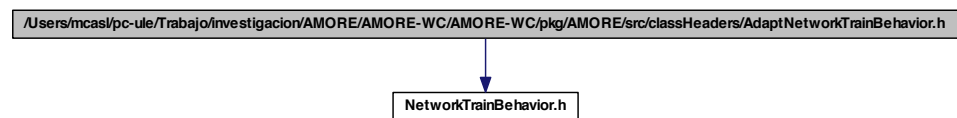
- class [ADAPTgdwmOutputNeuronTrainBehavior](#)  
*class ADAPTgdwmOutputNeuronTrainBehavior -*

### 6.13 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptNetworkTrainBehavior.h

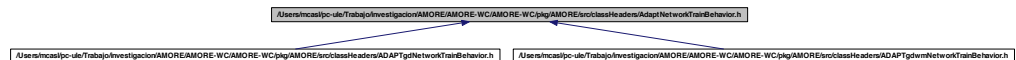
#### File Reference

```
#include "NetworkTrainBehavior.h"
```

Include dependency graph for AdaptNetworkTrainBehavior.h:



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



## Classes

- class [AdaptNetworkTrainBehavior](#)

*class [AdaptNetworkTrainBehavior](#) -*

### 6.14 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptNeuronTrainBehavior.h

#### File Reference

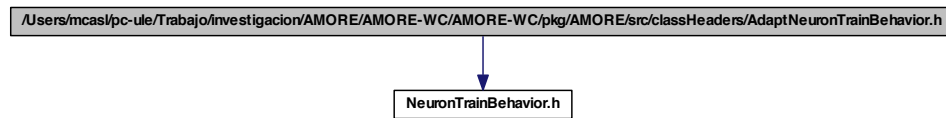
```
#include "NeuronTrainBehavior.h"
```



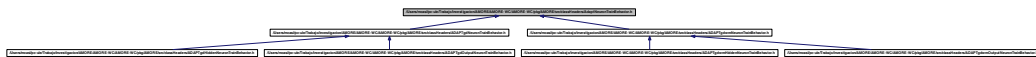
## 6.15 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ArcTan.h File Reference

269

Reference  
Include dependency graph for AdaptNeuronTrainBehavior.h:



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



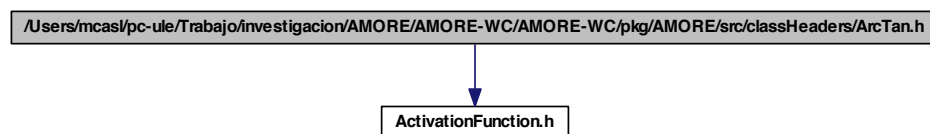
### Classes

- class [AdaptNeuronTrainBehavior](#)  
*class [AdaptNeuronTrainBehavior](#) -*

## 6.15 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ArcTan.h File Reference

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

Include dependency graph for ArcTan.h:



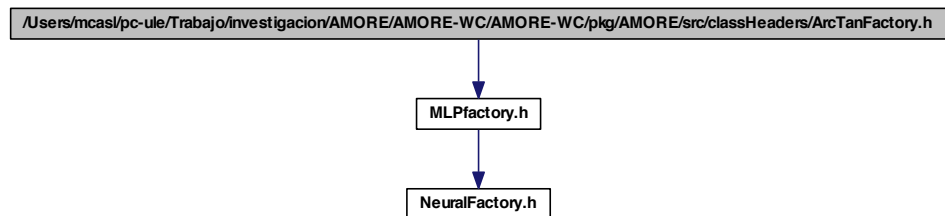
### Classes

- class [ArcTan](#)  
*class [ArcTan](#) -*

## 6.16 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ArcTanFactory.h File Reference

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

Include dependency graph for ArcTanFactory.h:



### Classes

- class [ArcTanFactory](#)

*class [ArcTanFactory](#) -*

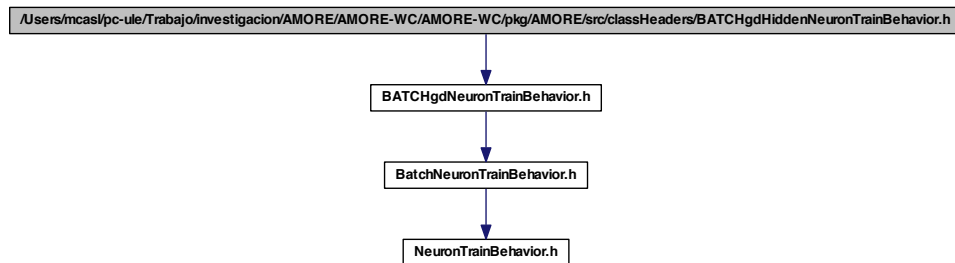
## 6.17 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdHiddenNeuronTrainBehavior.h File Reference

```
#include "BATCHgdNeuronTrainBehavior.h"
```

## 6.18 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdNetworkTrainBehavior.h File Reference

271

Include dependency graph for BATCHgdHiddenNeuronTrainBehavior.h:



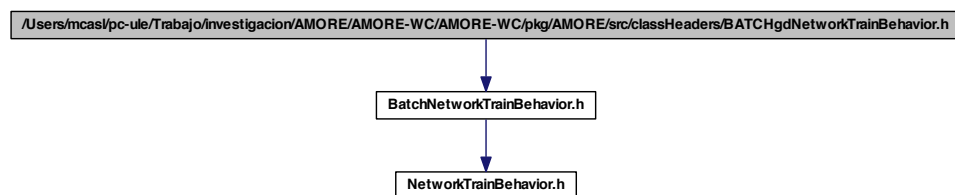
### Classes

- class [BATCHgdHiddenNeuronTrainBehavior](#)  
*class [BATCHgdHiddenNeuronTrainBehavior](#) -*

## 6.18 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdNetworkTrainBehavior.h File Reference

```
#include "BatchNetworkTrainBehavior.h"
```

Include dependency graph for BATCHgdNetworkTrainBehavior.h:



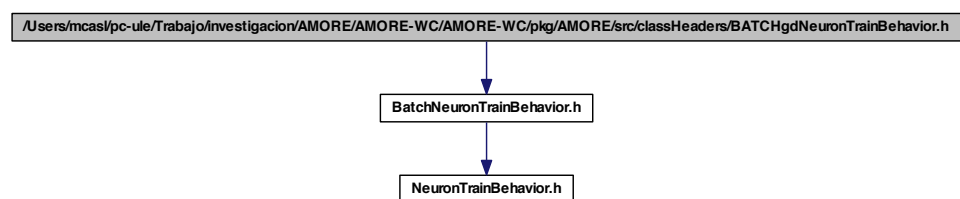
### Classes

- class [BATCHgdNetworkTrainBehavior](#)  
*class [BATCHgdNetworkTrainBehavior](#) -*

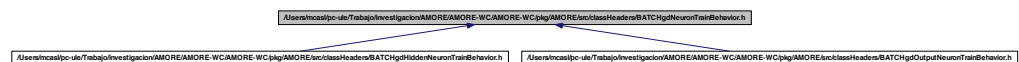
## 6.19 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdNeuronTrainBehavior.h File Reference

```
#include "BatchNeuronTrainBehavior.h"
```

Include dependency graph for BATCHgdNeuronTrainBehavior.h:



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



## Classes

- class [BATCHgdNeuronTrainBehavior](#)

*class [BATCHgdNeuronTrainBehavior](#) -*

## 6.20 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdOutputNeuronTrainBehavior.h File Reference

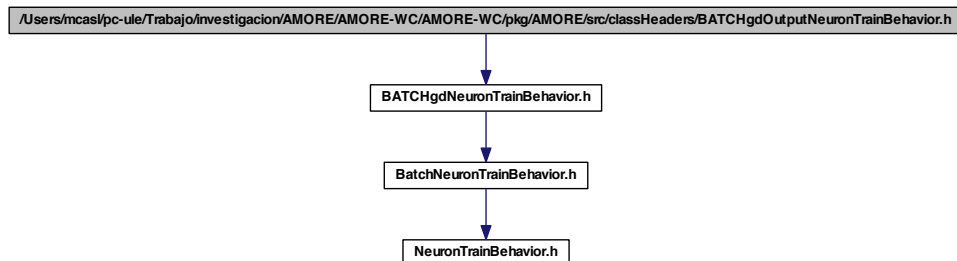
```
#include "BATCHgdNeuronTrainBehavior.h"
```

## 6.21 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwmHiddenNeuronTrainBehavior.h

### File Reference

273

Include dependency graph for BATCHgdOutputNeuronTrainBehavior.h:



### Classes

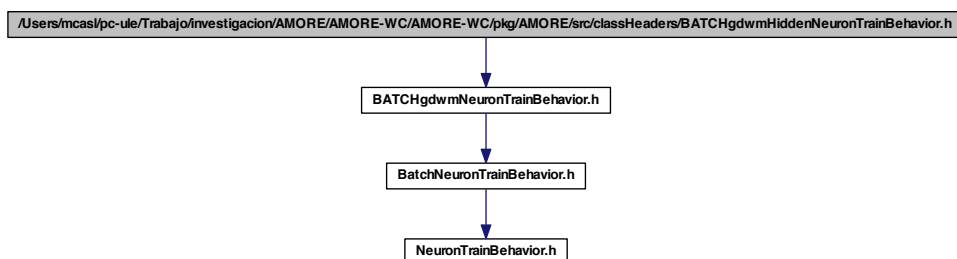
- class [BATCHgdOutputNeuronTrainBehavior](#)  
class [BATCHgdOutputNeuronTrainBehavior](#) -

## 6.21 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwmHiddenNeuronTrainBehavior.h

### File Reference

```
#include "BATCHgdwmNeuronTrainBehavior.h"
```

Include dependency graph for BATCHgdwmHiddenNeuronTrainBehavior.h:



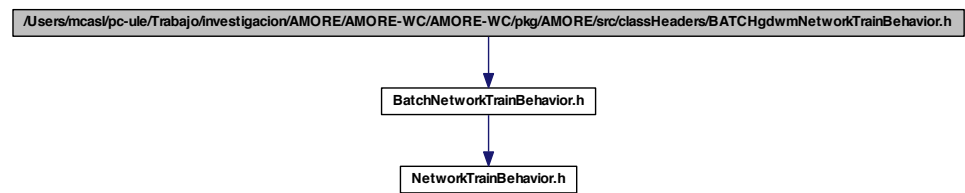
### Classes

- class [BATCHgdwmHiddenNeuronTrainBehavior](#)  
class [BATCHgdwmHiddenNeuronTrainBehavior](#) -

## 6.22 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwmNetworkTrainBehavior.h File Reference

```
#include "BatchNetworkTrainBehavior.h"
```

Include dependency graph for BATCHgdwmNetworkTrainBehavior.h:



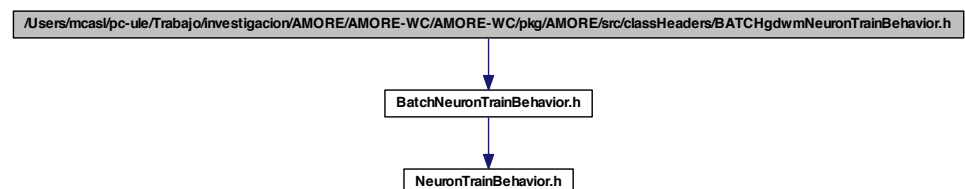
### Classes

- class [BATCHgdwmNetworkTrainBehavior](#)  
*class [BATCHgdwmNetworkTrainBehavior](#) -*

## 6.23 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwmNeuronTrainBehavior.h File Reference

```
#include "BatchNeuronTrainBehavior.h"
```

Include dependency graph for BATCHgdwmNeuronTrainBehavior.h:

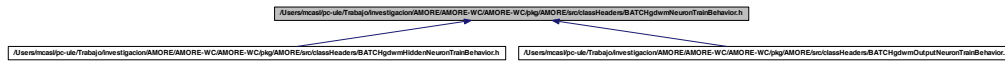


## 6.24 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwmOutputNeuronTrainBehavior.h

### File Reference

275

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



### Classes

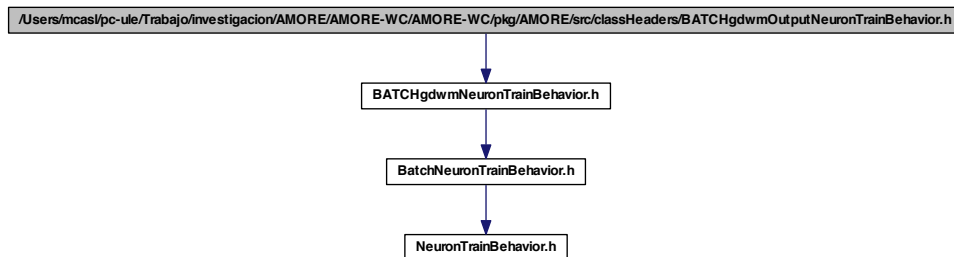
- class [BATCHgdwmNeuronTrainBehavior](#)  
*class [BATCHgdwmNeuronTrainBehavior](#) -*

## 6.24 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwmOutputNeuronTrainBehavior.h

### File Reference

```
#include "BATCHgdwmNeuronTrainBehavior.h"
```

Include dependency graph for BATCHgdwmOutputNeuronTrainBehavior.h:



### Classes

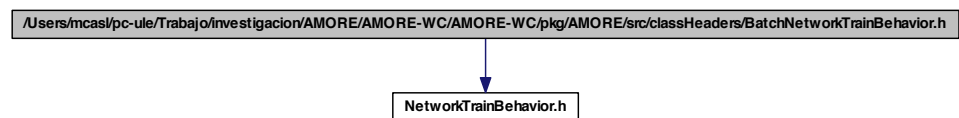
- class [BATCHgdwmOutputNeuronTrainBehavior](#)  
*class [BATCHgdwmOutputNeuronTrainBehavior](#) -*

## 6.25 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BatchNetworkTrainBehavior.h

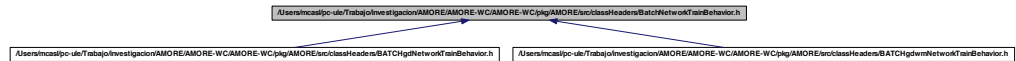
### File Reference

```
#include "NetworkTrainBehavior.h"
```

Include dependency graph for BatchNetworkTrainBehavior.h:



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



## Classes

- class [BatchNetworkTrainBehavior](#)

*class [BatchNetworkTrainBehavior](#) -*

## 6.26 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BatchNeuronTrainBehavior.h

### File Reference

```
#include "NeuronTrainBehavior.h"
```

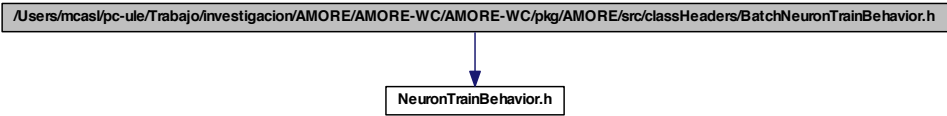


6.27 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Connection.h File

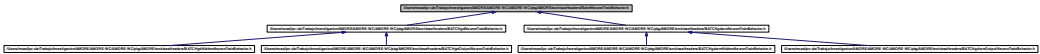
Reference

277

Include dependency graph for BatchNeuronTrainBehavior.h:



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



Classes

- class `BatchNeuronTrainBehavior`  
class `BatchNeuronTrainBehavior` -

6.27 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Connection.h File Reference

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



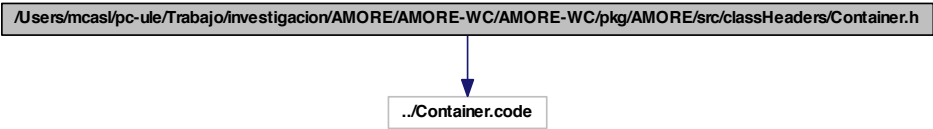
Classes

- class `Con`  
class `Con` -

6.28 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Container.h File Reference

```
#include "../Container.code"
```

Include dependency graph for Container.h:



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



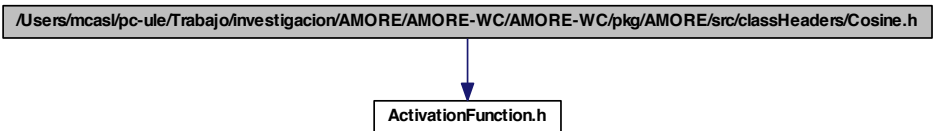
Classes

- class `Container< T >`  
class `Container` -

6.29 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Cosine.h File Reference

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

Include dependency graph for Cosine.h:



Classes

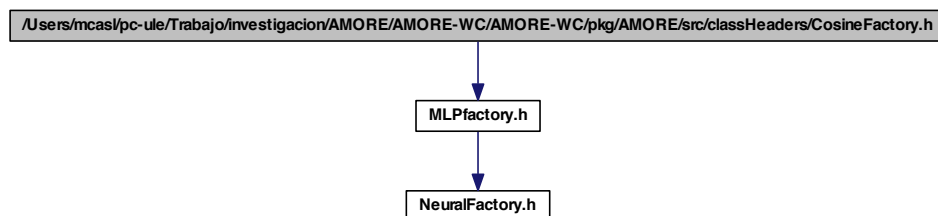
- class `Cosine`

class [Cosine](#) -

## 6.30 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/CosineFactory.h File Reference

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

Include dependency graph for CosineFactory.h:



### Classes

- class [CosineFactory](#)  
class [CosineFactory](#) -

## 6.31 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/CostFunction.h File Reference

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



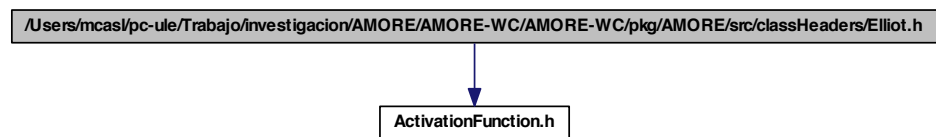
### Classes

- class [CostFunction](#)  
class [CostFunction](#) -

### 6.32 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Elliot.h File Reference

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

Include dependency graph for Elliot.h:



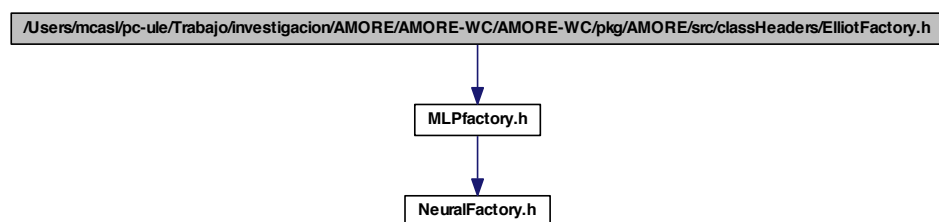
#### Classes

- class [Elliot](#)  
*class [Elliot](#) -*

### 6.33 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ElliotFactory.h File Reference

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

Include dependency graph for ElliotFactory.h:



#### Classes

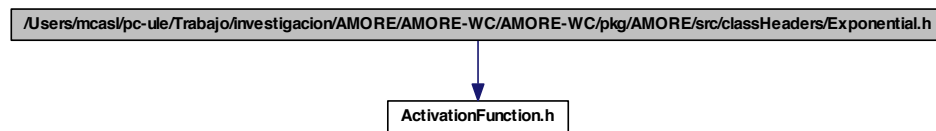
- class [ElliotFactory](#)

*class [ElliotFactory](#) -*

## 6.34 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Exponential.h File Reference

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

Include dependency graph for Exponential.h:



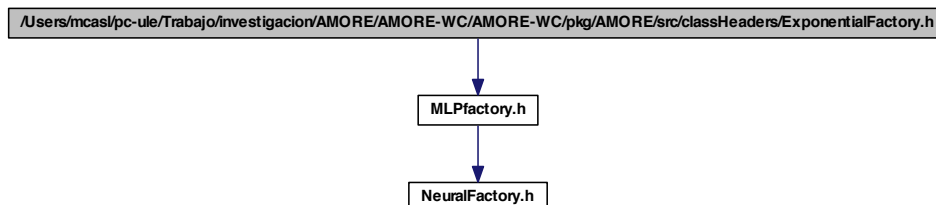
### Classes

- class [Exponential](#)  
*class [Exponential](#) -*

## 6.35 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ExponentialFactory.h File Reference

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

Include dependency graph for ExponentialFactory.h:



## Classes

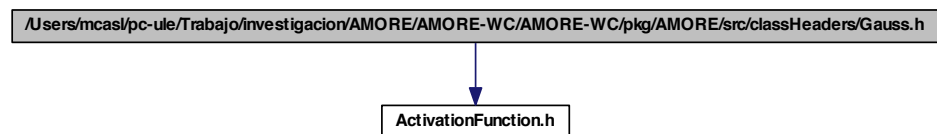
- class [ExponentialFactory](#)

*class [ExponentialFactory](#) -*

### 6.36 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Gauss.h File Reference

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

Include dependency graph for Gauss.h:



## Classes

- class [Gauss](#)

*class [Gauss](#) -*

### 6.37 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/GaussFactory.h File Reference

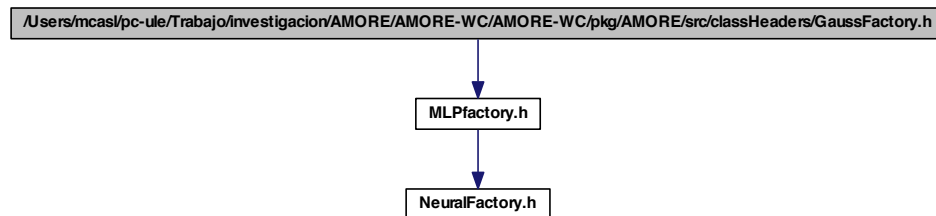
```
#include "MLPfactory.h"
```

### 6.38 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Identity.h File Reference

#### Reference

283

Include dependency graph for GaussFactory.h:



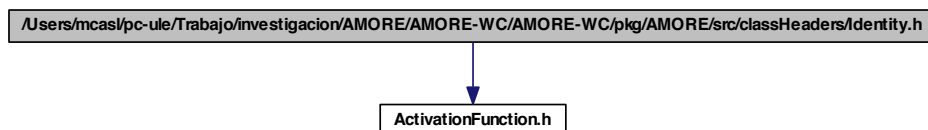
#### Classes

- class [GaussFactory](#)  
class [GaussFactory](#) -

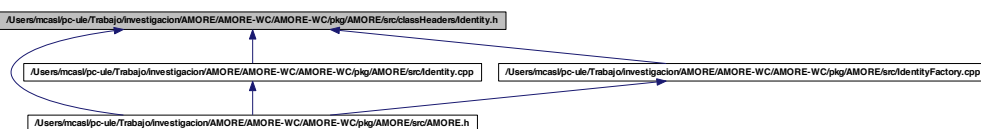
### 6.38 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Identity.h File Reference

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

Include dependency graph for Identity.h:



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



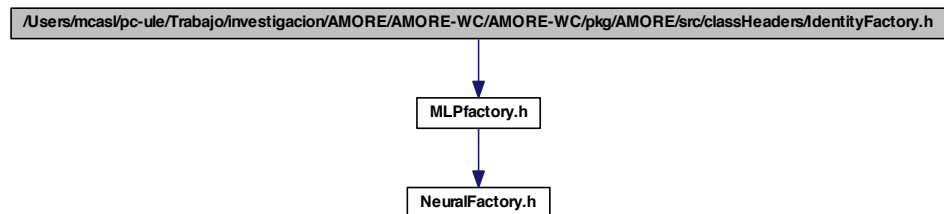
## Classes

- class [Identity](#)  
class [Identity](#) -

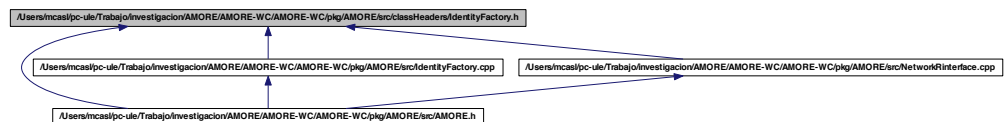
### 6.39 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/IdentityFactory.h File Reference

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

Include dependency graph for IdentityFactory.h:



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



## Classes

- class [IdentityFactory](#)  
class [IdentityFactory](#) -

### 6.40 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Iterator.h File Reference

```
#include "../Iterator.code"
```

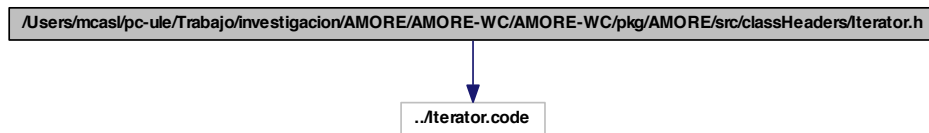


#### 6.41 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/LMLS.h File

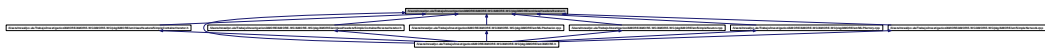
##### Reference

285

Include dependency graph for literator.h:



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



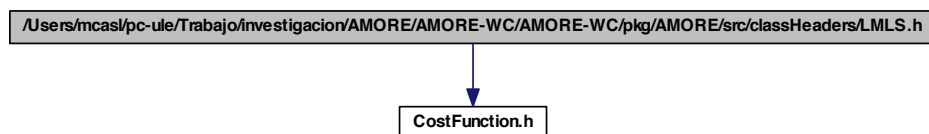
##### Classes

- class `literator< T >`  
*class `literator` -*

#### 6.41 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/LMLS.h File Reference

```
#include "CostFunction.h"
```

Include dependency graph for LMLS.h:



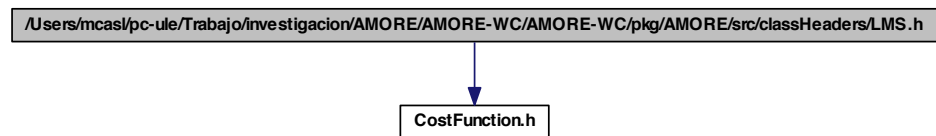
##### Classes

- class `LMLS`  
*class `LMLS` -*

## 6.42 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/LMS.h File Reference

```
#include "CostFunction.h"
```

Include dependency graph for LMS.h:



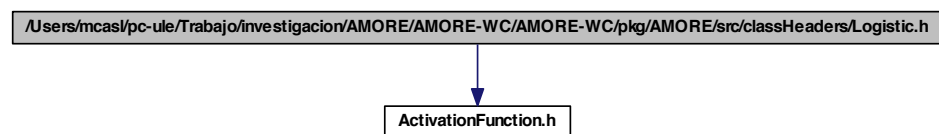
### Classes

- class [LMS](#)  
*class [LMS](#) -*

## 6.43 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Logistic.h File Reference

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

Include dependency graph for Logistic.h:



### Classes

- class [Logistic](#)  
*class [Logistic](#) -*

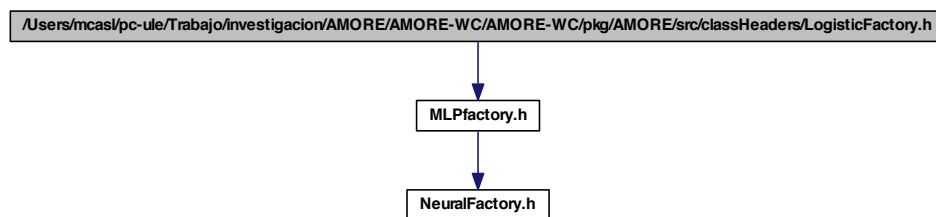
#### 6.44 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticFactory.h File

Reference

#### 6.44 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticFactory.h File Reference

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

Include dependency graph for LogisticFactory.h:



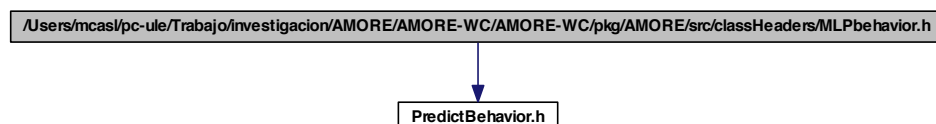
#### Classes

- class [LogisticFactory](#)  
*class [LogisticFactory](#) -*

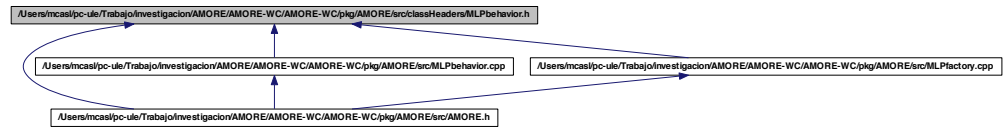
#### 6.45 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPbehavior.h File Reference

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

Include dependency graph for MLPbehavior.h:



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



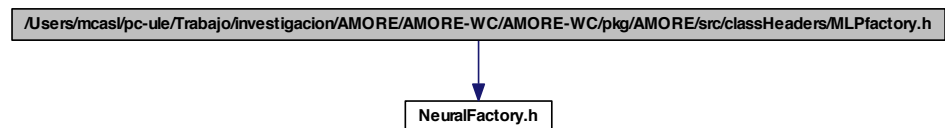
## Classes

- class [MLPbehavior](#)  
*class MLPbehavior -*

## 6.46 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/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 -*

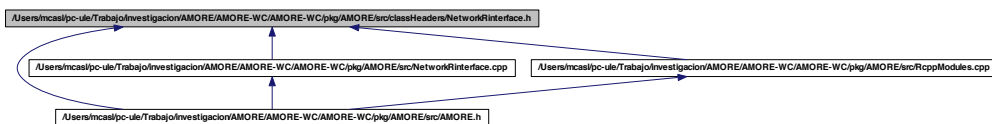
#### 6.47 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkRinterface.h File

Reference

#### 6.47 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkRinterface.h File Reference

289

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



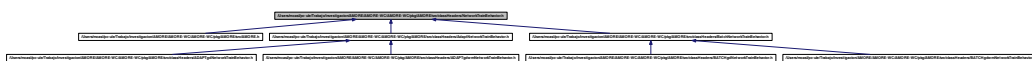
#### Classes

- class [NetworkRinterface](#)

class [NetworkRinterface](#) -

#### 6.48 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkTrainBehavior.h File Reference

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



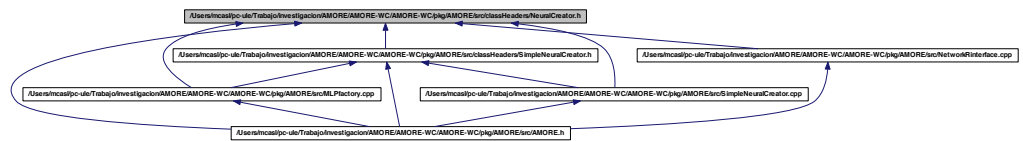
#### Classes

- class [NetworkTrainBehavior](#)

class [NetworkTrainBehavior](#) -

## 6.49 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralCreator.h File Reference

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



### Classes

- class [NeuralCreator](#)
- class [NeuralCreator](#) -

## 6.50 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralFactory.h File Reference

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



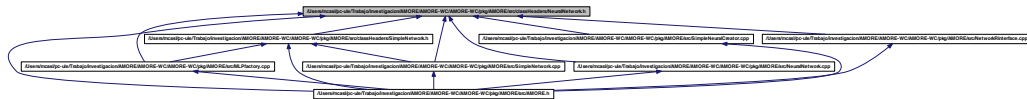
### Classes

- class [NeuralFactory](#)
- class [NeuralFactory](#) -

## 6.51 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralNetwork.h File Reference

## 6.51 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralNetwork.h File Reference

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

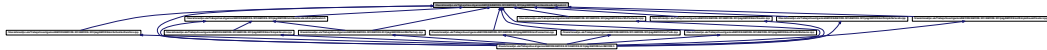


### Classes

- class [NeuralNetwork](#)  
class [NeuralNetwork](#) -

## 6.52 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Neuron.h File Reference

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



### Classes

- class [Neuron](#)  
class [Neuron](#) -

## 6.53 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuronTrainBehavior.h File Reference

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

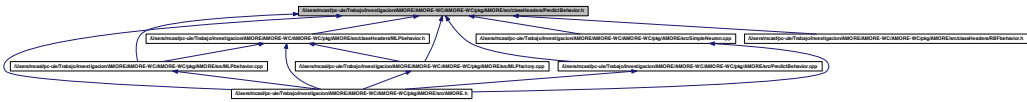


Classes

- class [NeuronTrainBehavior](#)  
class *NeuronTrainBehavior* -

6.54 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/PredictBehavior.h File Reference

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



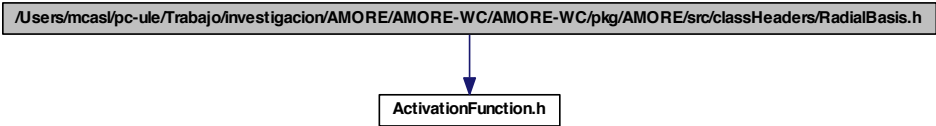
Classes

- class [PredictBehavior](#)  
class *PredictBehavior* -

6.55 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasis.h File Reference

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

Include dependency graph for RadialBasis.h:



Classes

- class [RadialBasis](#)  
class *RadialBasis* -



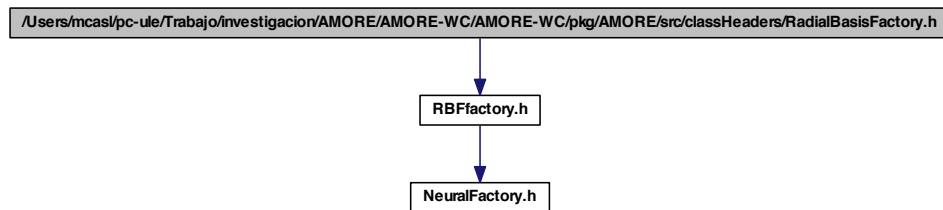
## 6.56 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasisFactory.h File Reference

Reference

## 6.56 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasisFactory.h File Reference

```
#include "RBFactory.h"
```

Include dependency graph for RadialBasisFactory.h:



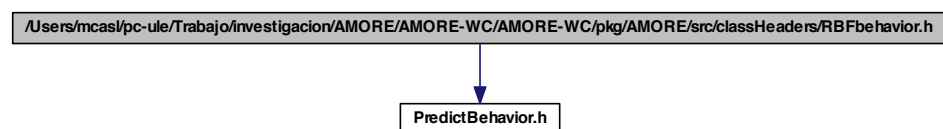
### Classes

- class [RadialBasisFactory](#)  
class [RadialBasisFactory](#) -

## 6.57 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFbehavior.h File Reference

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

Include dependency graph for RBFbehavior.h:



### Classes

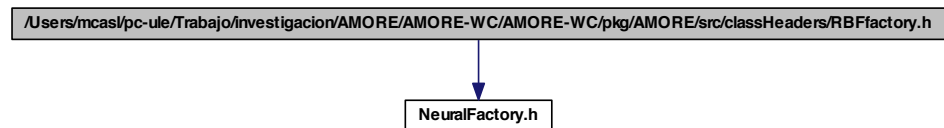
- class [RBFbehavior](#)

class *RBFBbehavior* -

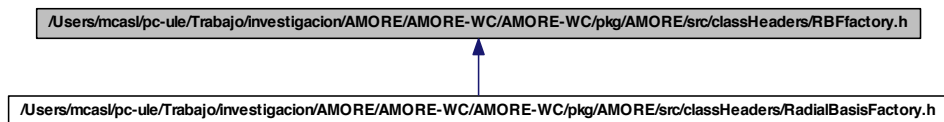
## 6.58 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFfactory.h File Reference

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

Include dependency graph for RBFfactory.h:



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



## Classes

- class *RBFBfactory*

class *RBFBfactory* -

## 6.59 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Reciprocal.h File Reference

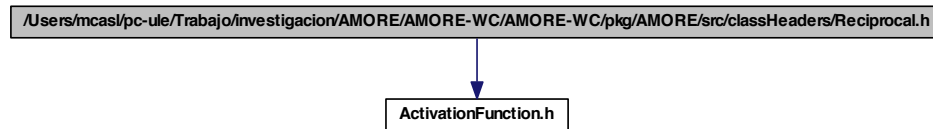
```
#include "ActivationFunction.h"
```

## 6.60 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ReciprocalFactory.h File

### Reference

295

Include dependency graph for Reciprocal.h:



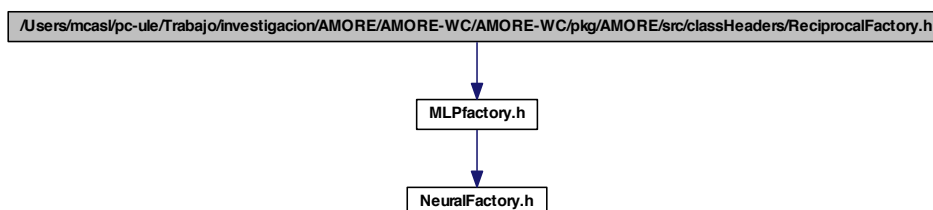
### Classes

- class [Reciprocal](#)  
*class [Reciprocal](#) -*

## 6.60 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ReciprocalFactory.h File Reference

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

Include dependency graph for ReciprocalFactory.h:



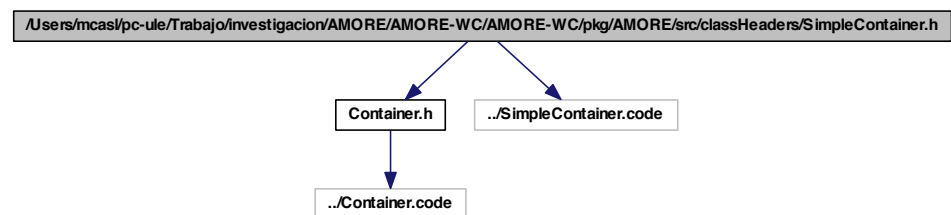
### Classes

- class [ReciprocalFactory](#)  
*class [ReciprocalFactory](#) -*

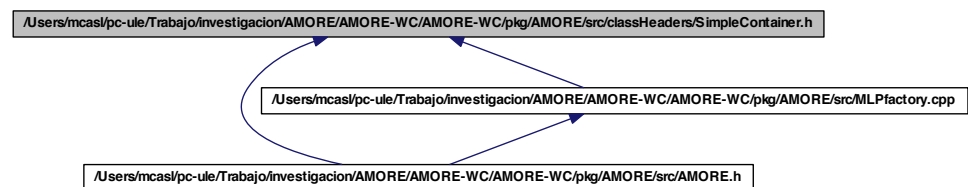
## 6.61 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainer.h File Reference

```
#include "Container.h"
#include "../SimpleContainer.code"
```

Include dependency graph for SimpleContainer.h:



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



### Classes

- class [SimpleContainer< T >](#)  
     class [SimpleContainer](#) -

## 6.62 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainerIterator.h File Reference

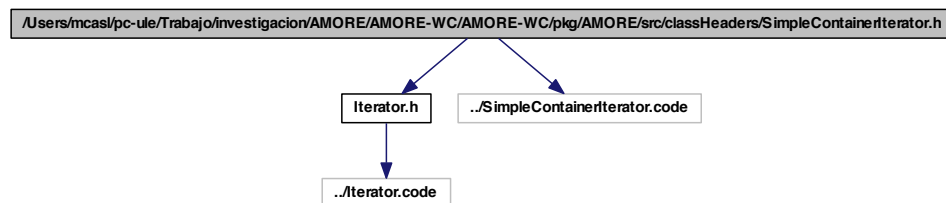
```
#include "Iterator.h"
```

### 6.63 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainerReverseliterator.h File Reference

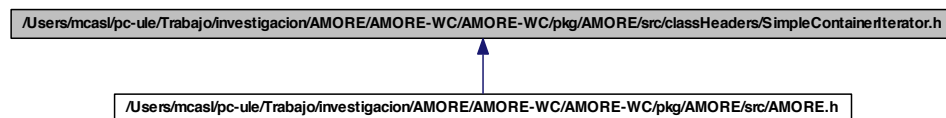
297

```
#include "../SimpleContainerIterator.code"
```

Include dependency graph for SimpleContainerIterator.h:



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



## Classes

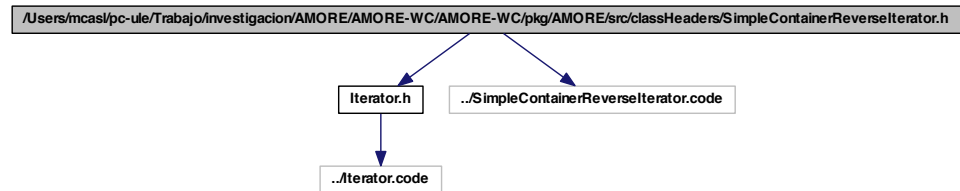
- class `SimpleContainerIterator< T >`  
*class `SimpleContainerIterator` -*

### 6.63 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainerReverseliterator.h File Reference

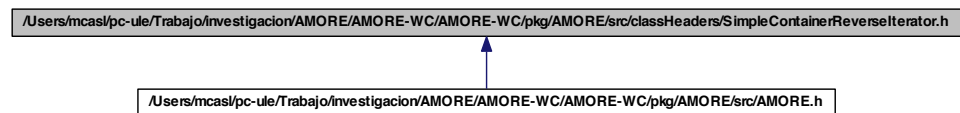
```
#include "Iterator.h"
```

```
#include "../SimpleContainerReverseIterator.code"
```

Include dependency graph for SimpleContainerReverseliterator.h:



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



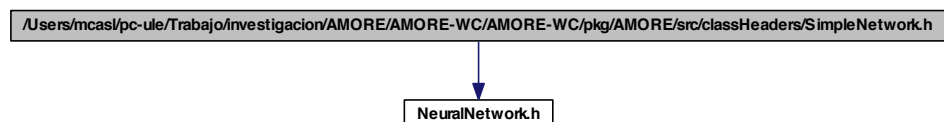
## Classes

- class `SimpleContainerReverseliterator< T >`  
     class `SimpleContainerReverseliterator` -

## 6.64 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNetwork.h File Reference

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

Include dependency graph for SimpleNetwork.h:

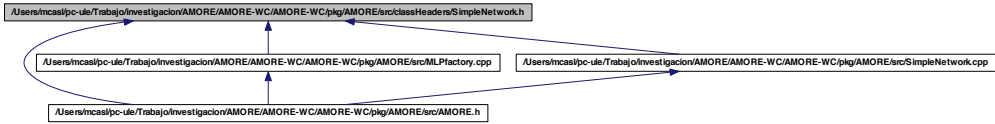


6.65 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuralCreator.h File

Reference

299

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



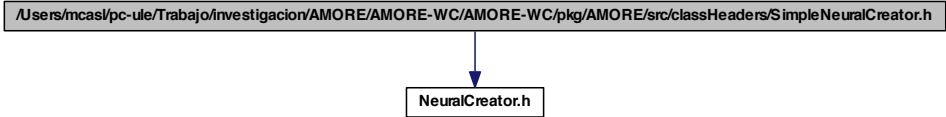
Classes

- class [SimpleNetwork](#)  
class [SimpleNetwork](#) -

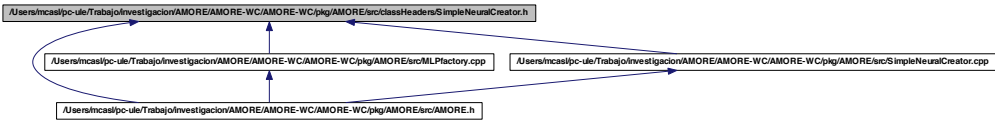
6.65 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuralCreator.h File  
Reference

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

Include dependency graph for SimpleNeuralCreator.h:



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



Classes

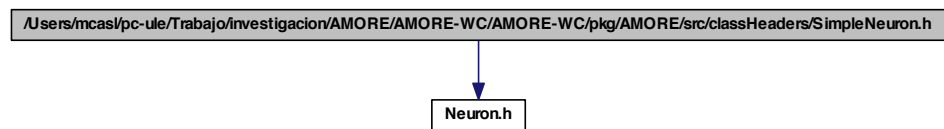
- class [SimpleNeuralCreator](#)

class [SimpleNeuralCreator](#) -

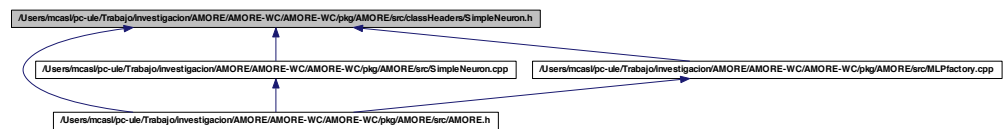
## 6.66 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuron.h File Reference

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

Include dependency graph for SimpleNeuron.h:



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



## Classes

- class [SimpleNeuron](#)

class [SimpleNeuron](#) -

## 6.67 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Sine.h File Reference

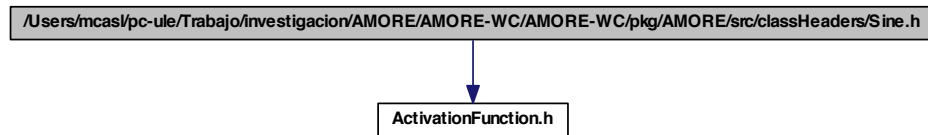
```
#include "ActivationFunction.h"
```



## 6.68 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SineFactory.h File Reference

301

Reference  
Include dependency graph for Sine.h:



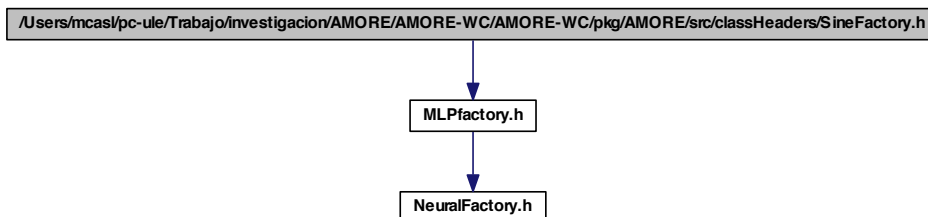
### Classes

- class [Sine](#)  
*class [Sine](#) -*

## 6.68 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SineFactory.h File Reference

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

Include dependency graph for SineFactory.h:



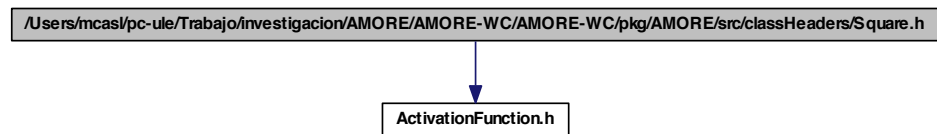
### Classes

- class [SineFactory](#)  
*class [SineFactory](#) -*

### 6.69 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Square.h File Reference

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

Include dependency graph for Square.h:



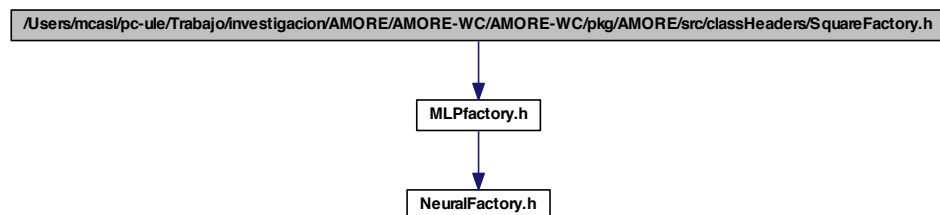
#### Classes

- class [Square](#)  
*class [Square](#) -*

### 6.70 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SquareFactory.h File Reference

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

Include dependency graph for SquareFactory.h:



#### Classes

- class [SquareFactory](#)

## 6.71 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Tanh.h File Reference

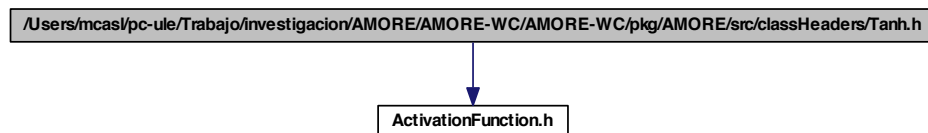
303

class [SquareFactory](#) -

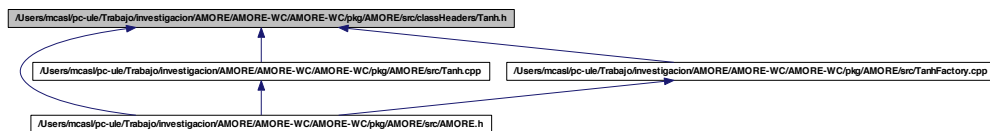
## 6.71 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Tanh.h File Reference

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

Include dependency graph for Tanh.h:



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



## Classes

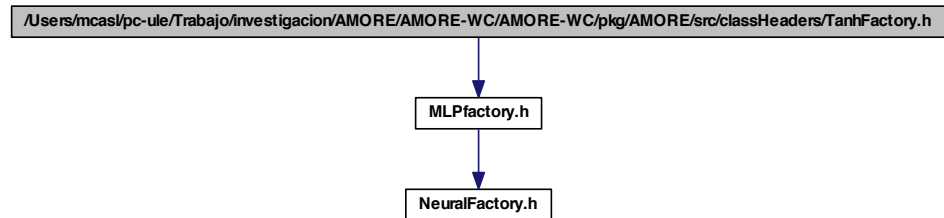
- class [Tanh](#)

class [Tanh](#) -

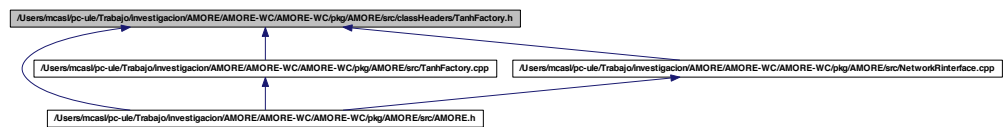
## 6.72 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/TanhFactory.h File Reference

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

Include dependency graph for TanhFactory.h:



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



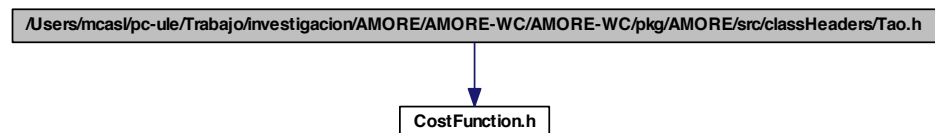
## Classes

- class [TanhFactory](#)  
class [TanhFactory](#) -

## 6.73 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Tao.h File Reference

```
#include "CostFunction.h"
```

Include dependency graph for Tao.h:



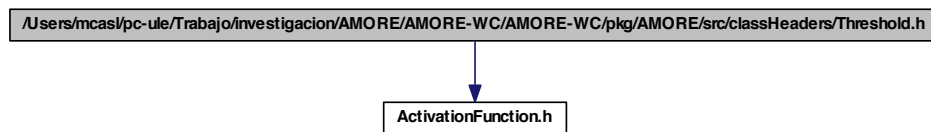
- class [Tao](#)

*class [Tao](#) -*

## 6.74 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Threshold.h File Reference

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

Include dependency graph for Threshold.h:



### Classes

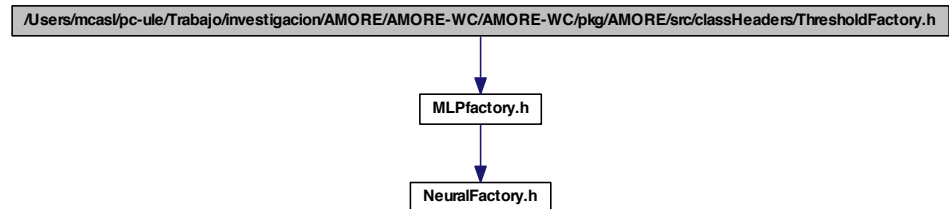
- class [Threshold](#)

*class [Threshold](#) -*

## 6.75 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ThresholdFactory.h File Reference

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

Include dependency graph for ThresholdFactory.h:



## Classes

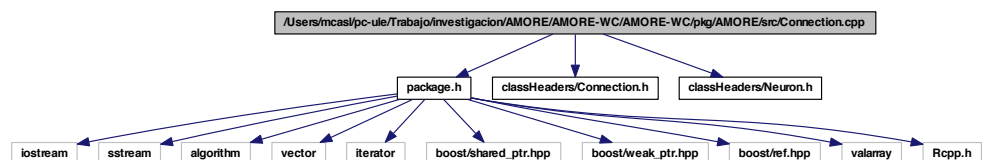
- class [ThresholdFactory](#)

class [ThresholdFactory](#) -

## 6.76 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Connection.cpp File Reference

```
#include "package.h"
#include "classHeaders/Connection.h"
#include "classHeaders/Neuron.h"
```

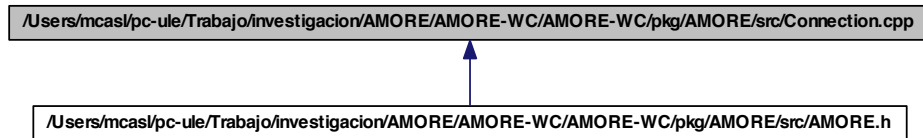
Include dependency graph for Connection.cpp:



## 6.77 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Identity.cpp File Reference

307

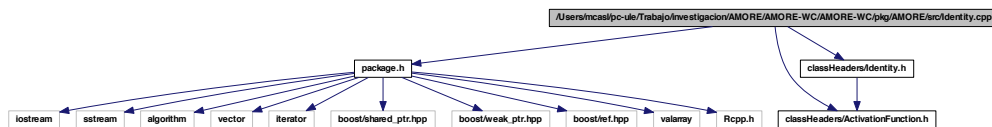
This graph shows which files directly or indirectly include this file:



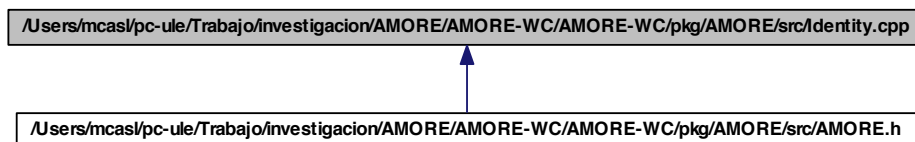
## 6.77 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Identity.cpp File Reference

```
#include "package.h"  
#include "classHeaders/ActivationFunction.h"  
#include "classHeaders/Identity.h"
```

Include dependency graph for Identity.cpp:



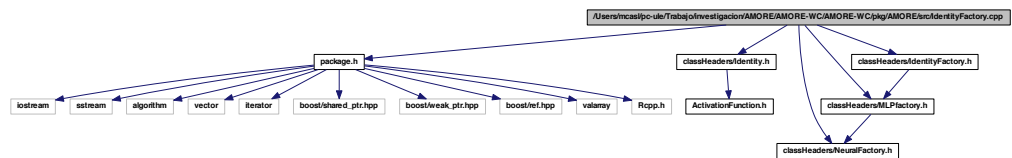
This graph shows which files directly or indirectly include this file:



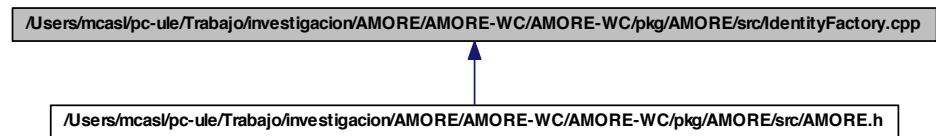
## 6.78 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/IdentityFactory.cpp File Reference

```
#include "package.h"
#include "classHeaders/Identity.h"
#include "classHeaders/NeuralFactory.h"
#include "classHeaders/MLPfactory.h"
#include "classHeaders/IdentityFactory.h"
```

Include dependency graph for IdentityFactory.cpp:



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



## 6.79 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPbehavior.cpp File Reference

```
#include "package.h"
#include "classHeaders/Connection.h"
#include "classHeaders/Iterator.h"
#include "classHeaders/Neuron.h"
#include "classHeaders/PredictBehavior.h"
#include "classHeaders/MLPbehavior.h"
```

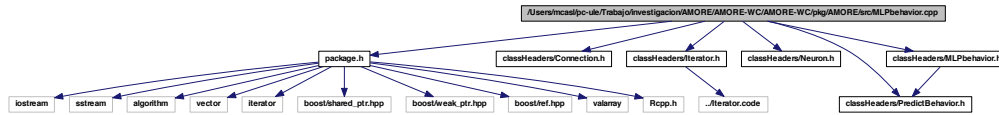


## 6.80 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPfactory.cpp File

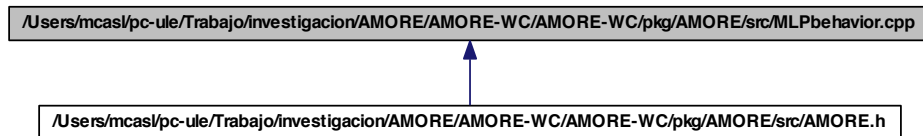
### Reference

309

Include dependency graph for MLPbehavior.cpp:



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



## 6.80 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPfactory.cpp File Reference

```
#include "package.h"
#include "classHeaders/Connection.h"
#include "classHeaders/Neuron.h"
#include "classHeaders/SimpleNeuron.h"
#include "classHeaders/Container.h"
#include "classHeaders/SimpleContainer.h"
#include "classHeaders/NeuralNetwork.h"
#include "classHeaders/SimpleNetwork.h"
#include "classHeaders/NeuralCreator.h"
#include "classHeaders/SimpleNeuralCreator.h"
#include "classHeaders/predictBehavior.h"
#include "classHeaders/MLPbehavior.h"
#include "classHeaders/Iterator.h"
#include "classHeaders/NeuralFactory.h"
#include "classHeaders/MLPfactory.h"
```

[illegible]

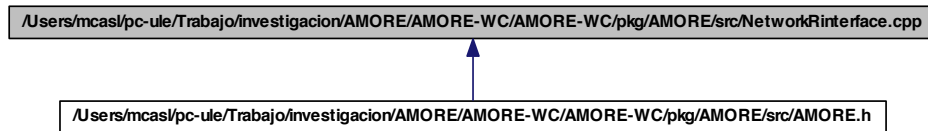
```
graph BT; A["/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPfactory.cpp"] --> B["/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/AMORE.h"]
```

```
#include "package.h"
#include "classHeaders/IdentityFactory.h"
#include "classHeaders/TanhFactory.h"
#include "classHeaders/NeuralFactory.h"
#include "classHeaders/NeuralNetwork.h"
#include "classHeaders/NeuralCreator.h"
#include "classHeaders/NetworkRinterface.h"
```

## 6.82 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NeuralNetwork.cpp File Reference

311

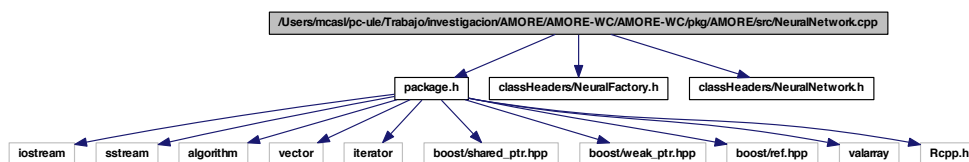
This graph shows which files directly or indirectly include this file:



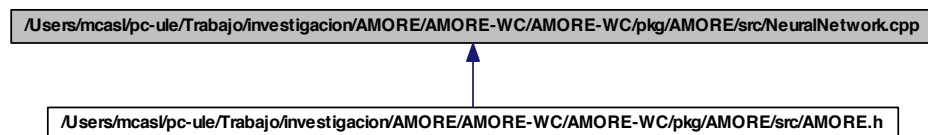
## 6.82 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NeuralNetwork.cpp File Reference

```
#include "package.h"  
#include "classHeaders/NeuralFactory.h"  
#include "classHeaders/NeuralNetwork.h"
```

Include dependency graph for NeuralNetwork.cpp:



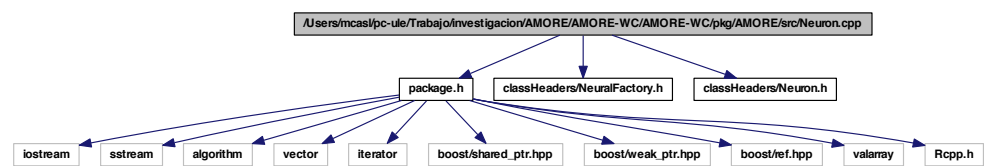
This graph shows which files directly or indirectly include this file:



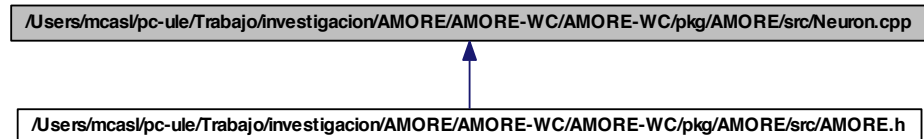
### 6.83 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Neuron.cpp File Reference

```
#include "package.h"
#include "classHeaders/NeuralFactory.h"
#include "classHeaders/Neuron.h"
```

Include dependency graph for Neuron.cpp:



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



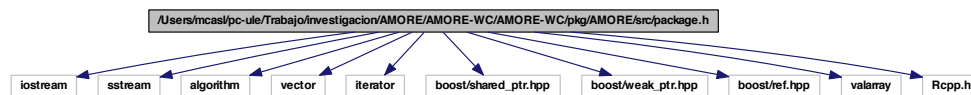
### 6.84 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/package.h File Reference

```
#include <iostream>
#include <sstream>
#include <algorithm>
#include <vector>
#include <iterator>
#include <boost/shared_ptr.hpp>
#include <boost/weak_ptr.hpp>
#include <boost/ref.hpp>
```

```
#include <valarray>
```

```
#include <Rcpp.h>
```

Include dependency graph for package.h:



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



## Defines

- #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< [Neuron](#) > [NeuronRef](#)
- typedef boost::shared\_ptr< [ActivationFunction](#) > [ActivationFunctionPtr](#)
- typedef boost::shared\_ptr< [PredictBehavior](#) > [PredictBehaviorPtr](#)
- typedef boost::shared\_ptr< [NetworkTrainBehavior](#) > [NetworkTrainBehaviorPtr](#)
- typedef boost::shared\_ptr< [NeuronTrainBehavior](#) > [NeuronTrainBehaviorPtr](#)
- 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](#)< [LayerPtr](#) > > [LayerContainerPtr](#)
- typedef boost::shared\_ptr< [Container](#)< [ConPtr](#) > > [ConContainerPtr](#)
- typedef boost::shared\_ptr< [NeuralFactory](#) > [NeuralFactoryPtr](#)
- typedef boost::shared\_ptr< [NeuralCreator](#) > [NeuralCreatorPtr](#)
- typedef boost::weak\_ptr< [NeuralNetwork](#) > [NeuralNetworkWeakPtr](#)
- typedef boost::weak\_ptr< [Neuron](#) > [NeuronWeakPtr](#)

### 6.84.1 Define Documentation

#### 6.84.1.1 `#define size_type unsigned int`

Definition at line 81 of file package.h.

### 6.84.2 Typedef Documentation

#### 6.84.2.1 `typedef boost::shared_ptr<ActivationFunction> ActivationFunctionPtr`

Definition at line 91 of file package.h.

#### 6.84.2.2 `typedef boost::reference_wrapper<PredictBehavior> ActivationFunctionRef`

Definition at line 86 of file package.h.

#### 6.84.2.3 `typedef boost::shared_ptr<Container<ConPtr> > ConContainerPtr`

Definition at line 105 of file package.h.

#### 6.84.2.4 `typedef boost::shared_ptr<Iterator<ConPtr> > ConIteratorPtr`

Definition at line 100 of file package.h.

#### 6.84.2.5 `typedef boost::shared_ptr<Con> ConPtr`

Definition at line 96 of file package.h.

#### 6.84.2.6 `typedef int Handler`

Definition at line 84 of file package.h.

#### 6.84.2.7 `typedef boost::shared_ptr<Container<LayerPtr> > LayerContainerPtr`

Definition at line 103 of file package.h.

#### 6.84.2.8 `typedef boost::shared_ptr<Container<NeuronPtr> > LayerPtr`

Definition at line 102 of file package.h.

6.84.2.9 `typedef boost::shared_ptr<NetworkTrainBehavior>  
NetworkTrainBehaviorPtr`

Definition at line 93 of file package.h.

6.84.2.10 `typedef boost::shared_ptr<NeuralCreator> NeuralCreatorPtr`

Definition at line 108 of file package.h.

6.84.2.11 `typedef boost::shared_ptr<NeuralFactory> NeuralFactoryPtr`

Definition at line 107 of file package.h.

6.84.2.12 `typedef boost::shared_ptr<NeuralNetwork> NeuralNetworkPtr`

Definition at line 97 of file package.h.

6.84.2.13 `typedef boost::weak_ptr<NeuralNetwork> NeuralNetworkWeakPtr`

Definition at line 110 of file package.h.

6.84.2.14 `typedef boost::shared_ptr<Iterator<NeuronPtr> > NeuronIteratorPtr`

Definition at line 99 of file package.h.

6.84.2.15 `typedef boost::shared_ptr<Neuron> NeuronPtr`

Definition at line 95 of file package.h.

6.84.2.16 `typedef boost::reference_wrapper<Neuron> NeuronRef`

Definition at line 89 of file package.h.

6.84.2.17 `typedef boost::shared_ptr<NeuronTrainBehavior> NeuronTrainBehaviorPtr`

Definition at line 94 of file package.h.

6.84.2.18 `typedef boost::weak_ptr<Neuron> NeuronWeakPtr`

Definition at line 111 of file package.h.

6.84.2.19 `typedef boost::shared_ptr<PredictBehavior> PredictBehaviorPtr`

Definition at line 92 of file package.h.

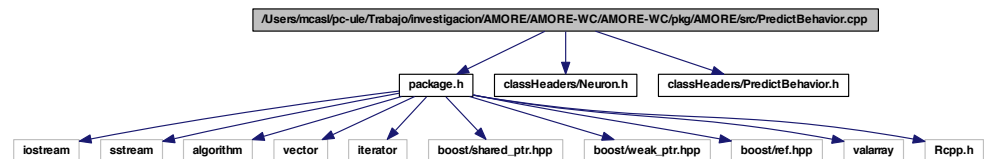
6.84.2.20 `typedef boost::reference_wrapper<PredictBehavior> PredictBehaviorRef`

Definition at line 87 of file package.h.

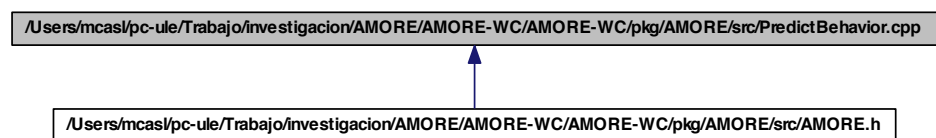
## 6.85 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/PredictBehavior.cpp File Reference

```
#include "package.h"
#include "classHeaders/Neuron.h"
#include "classHeaders/PredictBehavior.h"
```

Include dependency graph for PredictBehavior.cpp:



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



## 6.86 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/RcppModules.cpp File Reference

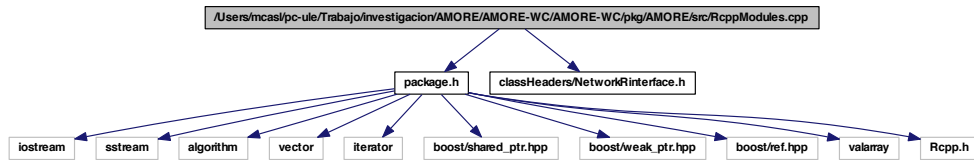
```
#include "package.h"
```



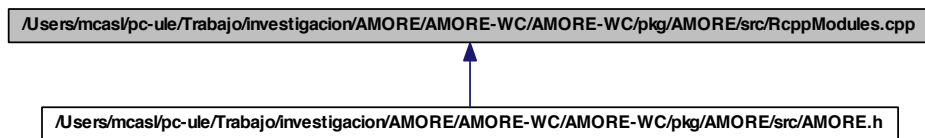
## Reference

#include "classHeaders/NetworkRinterface.h"

Include dependency graph for RcppModules.cpp:



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



## Functions

- [RCPP\\_MODULE](#) (modAMORE)

### 6.86.1 Function Documentation

#### 6.86.1.1 RCPP\_MODULE ( modAMORE )

Definition at line 5 of file RcppModules.cpp.

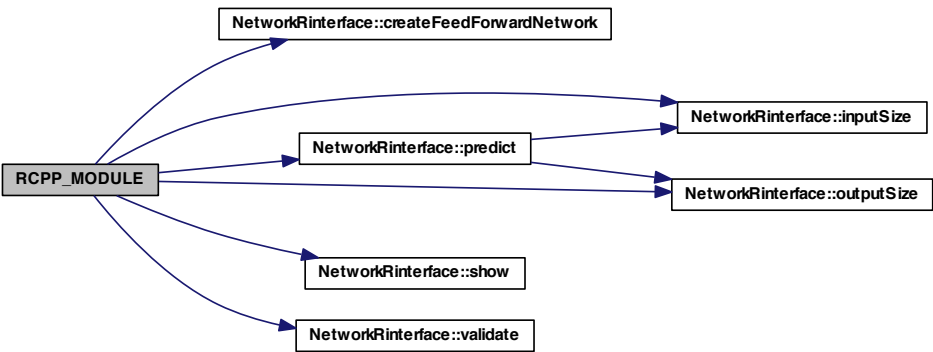
References `NetworkRinterface::createFeedForwardNetwork()`, `NetworkRinterface::inputSize()`, `NetworkRinterface::outputSize()`, `NetworkRinterface::predict()`, `NetworkRinterface::show()`, and `NetworkRinterface::validate()`.

```

{
  class_<NetworkRinterface>( "NetworkRinterface" )
    .constructor()
    .method( "createFeedForwardNetwork", &
      NetworkRinterface::createFeedForwardNetwork )
    .method( "predict", &NetworkRinterface::predict )
    .method( "inputSize", &NetworkRinterface::inputSize )
    .method( "outputSize", &NetworkRinterface::outputSize )
    .method( "show", &NetworkRinterface::show )
    .method( "validate", &NetworkRinterface::validate )
}
  
```

```
    ;  
}
```

Here is the call graph for this function:



6.87 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNetwork.cpp File Reference

```
#include "package.h"  
#include "classHeaders/Container.h"  
#include "classHeaders/Iterator.h"  
#include "classHeaders/Neuron.h"  
#include "classHeaders/NeuralNetwork.h"  
#include "classHeaders/SimpleNetwork.h"
```

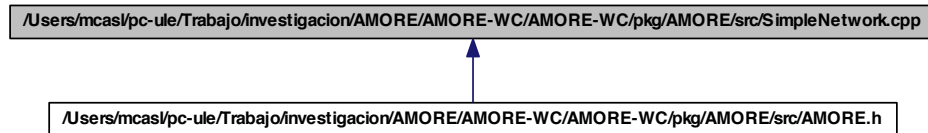
Include dependency graph for SimpleNetwork.cpp:



## 6.88 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuralCreator.cpp File Reference

319

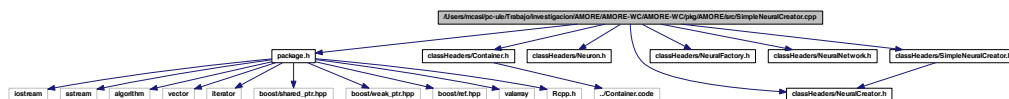
This graph shows which files directly or indirectly include this file:



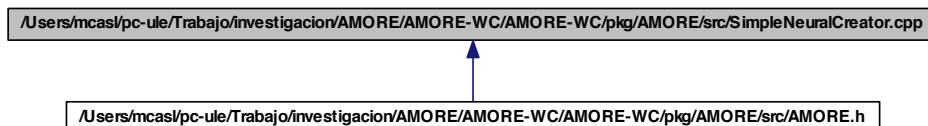
## 6.88 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuralCreator.cpp File Reference

```
#include "package.h"
#include "classHeaders/Container.h"
#include "classHeaders/Neuron.h"
#include "classHeaders/NeuralCreator.h"
#include "classHeaders/NeuralFactory.h"
#include "classHeaders/NeuralNetwork.h"
#include "classHeaders/SimpleNeuralCreator.h"
```

Include dependency graph for SimpleNeuralCreator.cpp:



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



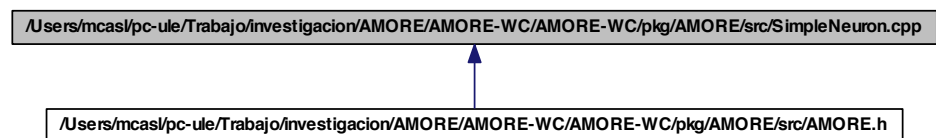
## 6.89 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuron.cpp File Reference

```
#include "package.h"
#include "classHeaders/NeuralFactory.h"
#include "classHeaders/Container.h"
#include "classHeaders/Iterator.h"
#include "classHeaders/ActivationFunction.h"
#include "classHeaders/PredictBehavior.h"
#include "classHeaders/SimpleNeuron.h"
```

Include dependency graph for SimpleNeuron.cpp:



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



## 6.90 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Tanh.cpp File Reference

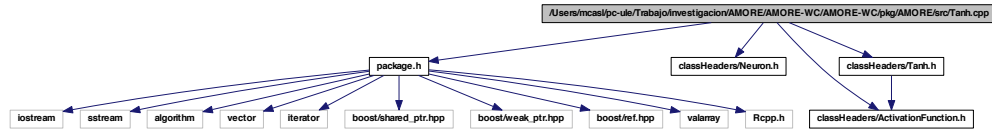
```
#include "package.h"
#include "classHeaders/Neuron.h"
#include "classHeaders/ActivationFunction.h"
#include "classHeaders/Tanh.h"
```

## 6.91 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/TanhFactory.cpp File Reference

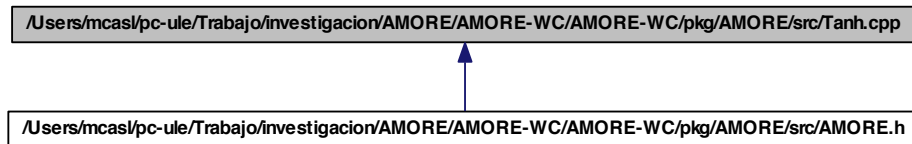
321

### Reference

Include dependency graph for Tanh.cpp:



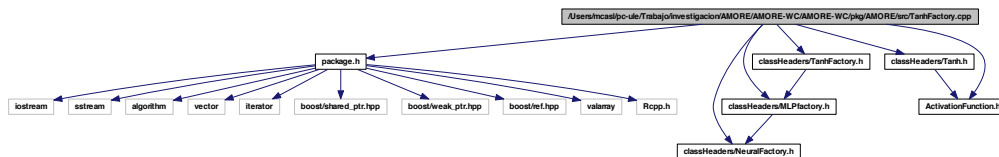
This graph shows which files directly or indirectly include this file:



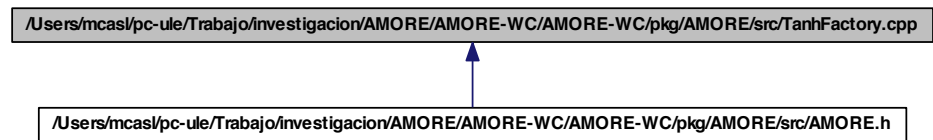
## 6.91 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/TanhFactory.cpp File Reference

```
#include "package.h"
#include "classHeaders/NeuralFactory.h"
#include "classHeaders/MLPFactory.h"
#include "classHeaders/Tanh.h"
#include "classHeaders/TanhFactory.h"
#include "classHeaders/ActivationFunction.h"
```

Include dependency graph for TanhFactory.cpp:



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



# Index

~Container	WC/AMORE-WC/pkg/AMORE/src/Neuron.cpp,
Container, 91	312
~Iterator	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
Iterator, 125	WC/AMORE-WC/pkg/AMORE/src/PredictBehavior.cpp,
~SimpleContainer	316
SimpleContainer, 198	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
~SimpleContainerIterator	WC/AMORE-WC/pkg/AMORE/src/RcppModules.cpp,
SimpleContainerIterator, 203	316
~SimpleContainerReverselIterator	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
SimpleContainerReverselIterator, 207	WC/AMORE-WC/pkg/AMORE/src/SimpleNetwork.cpp,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	316
WC/AMORE-WC/pkg/AMORE/src/ADAPTgdNetworkTrabajaInvestigacion/AMORE/AMORE-	316
258	WC/AMORE-WC/pkg/AMORE/src/SimpleNeuralCreator.cpp,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	316
WC/AMORE-WC/pkg/AMORE/src/AMORE/pc-ule/Trabajo/investigacion/AMORE/AMORE-	316
258	WC/AMORE-WC/pkg/AMORE/src/SimpleNeuron.cpp,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	316
WC/AMORE-WC/pkg/AMORE/src/ActivationFunctionTrabajo/investigacion/AMORE/AMORE-	316
257	WC/AMORE-WC/pkg/AMORE/src/Tanh.cpp,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	316
WC/AMORE-WC/pkg/AMORE/src/Connections.cpp,	316
306	WC/AMORE-WC/pkg/AMORE/src/TanhFactory.cpp,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	316
WC/AMORE-WC/pkg/AMORE/src/Utils.cpp,	316
307	WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdHiddenNeu
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	316
WC/AMORE-WC/pkg/AMORE/src/Utils/Factory.cpp,	316
308	WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdNetworkTra
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	316
WC/AMORE-WC/pkg/AMORE/src/Utils/Neuron.cpp,	316
308	WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdNeuronTrain
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	316
WC/AMORE-WC/pkg/AMORE/src/Utils/Factory.cpp,	316
309	WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdOutputNeu
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	316
WC/AMORE-WC/pkg/AMORE/src/Utils/Factory.cpp,	316
310	WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwmHiddenN
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	316
WC/AMORE-WC/pkg/AMORE/src/Utils/Factory.cpp,	316
311	WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwmNetwork
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	316

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/DAE/Trabajo/Investigacion/AMORE-  
 266 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Connect  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/DAE/Trabajo/Investigacion/AMORE/AMORE-  
 267 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Contain  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/DAE/Trabajo/Investigacion/AMORE/AMORE-  
 262 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Cosine.h  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/DAE/Trabajo/Investigacion/AMORE/AMORE-  
 268 WC/AMORE-WC/pkg/AMORE/src/classHeaders/CosineF  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/DAE/Trabajo/Investigacion/AMORE/AMORE-  
 268 WC/AMORE-WC/pkg/AMORE/src/classHeaders/CostFun  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/DAE/Trabajo/investigacion/AMORE/AMORE-  
 269 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Elliot.h,  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/DAE/Trabajo/investigacion/AMORE/AMORE-  
 270 WC/AMORE-WC/pkg/AMORE/src/classHeaders/ElliotFac  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/DAE/Trabajo/investigacion/AMORE/AMORE-  
 270 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Exponer  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/DAE/Trabajo/investigacion/AMORE/AMORE-  
 271 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Exponer  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/DAE/Trabajo/investigacion/AMORE/AMORE-  
 272 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Gauss.h  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/DAE/Trabajo/investigacion/AMORE/AMORE-  
 272 WC/AMORE-WC/pkg/AMORE/src/classHeaders/GaussFa  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/DAE/Trabajo/investigacion/AMORE/AMORE-  
 273 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Identity.h  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/DAE/Trabajo/Investigacion/AMORE/AMORE-  
 274 WC/AMORE-WC/pkg/AMORE/src/classHeaders/IdentityF  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/DAE/Trabajo/Investigacion/AMORE/AMORE-  
 274 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Iterator.h  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/DAE/Trabajo/Investigacion/AMORE/AMORE-  
 275 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LMLS.h,  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/DAE/Trabajo/Investigacion/AMORE/AMORE-  
 276 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LMS.h,  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/DAE/Trabajo/Investigacion/AMORE/AMORE-



WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticTrabajo/investigacion/AMORE/AMORE-  
 286 WC/AMORE-WC/pkg/AMORE/src/classHeaders/ReciprocalFactory.h  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticTrabajo/investigacion/AMORE/AMORE-  
 287 WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainer.h,  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticTrabajo/investigacion/AMORE/AMORE-  
 287 WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainerItera  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticTrabajo/investigacion/AMORE/AMORE-  
 288 WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainerReve  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticTrabajo/investigacion/AMORE/AMORE-  
 289 WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNetwork.h,  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticTrabajo/investigacion/AMORE/AMORE-  
 289 WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuralCreator  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticTrabajo/investigacion/AMORE/AMORE-  
 290 WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuron.h,  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticTrabajo/investigacion/AMORE/AMORE-  
 290 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Sine.h,  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticTrabajo/investigacion/AMORE/AMORE-  
 291 WC/AMORE-WC/pkg/AMORE/src/classHeaders/SineFactory.h,  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticTrabajo/investigacion/AMORE/AMORE-  
 291 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Square.h,  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticTrabajo/investigacion/AMORE/AMORE-  
 291 WC/AMORE-WC/pkg/AMORE/src/classHeaders/SquareFactory.h,  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticTrabajo/investigacion/AMORE/AMORE-  
 292 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Tanh.h,  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticTrabajo/investigacion/AMORE/AMORE-  
 293 WC/AMORE-WC/pkg/AMORE/src/classHeaders/TanhFactory.h,  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticTrabajo/investigacion/AMORE/AMORE-  
 294 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Tao.h,  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticTrabajo/investigacion/AMORE/AMORE-  
 292 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Threshold.h,  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticTrabajo/investigacion/AMORE/AMORE-  
 293 WC/AMORE-WC/pkg/AMORE/src/classHeaders/ThresholdFactory.h,  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticTrabajo/investigacion/AMORE/AMORE-  
 294 WC/AMORE-WC/pkg/AMORE/src/package.h,

- 312
- ActivationFunction, 15
  - ActivationFunction, 16
  - d\_neuron, 17
  - f0, 16
  - f1, 16
  - getInducedLocalField, 16
- ActivationFunctionPtr
  - AMORE.h, 260
  - package.h, 314
- ActivationFunctionRef
  - AMORE.h, 260
  - package.h, 314
- ADAPTgdHiddenNeuronTrainBehavior, 17
  - endOfEpochAction, 20
  - singlePatternBackwardAction, 20
- ADAPTgdNetworkTrainBehavior, 20
  - train, 22
- ADAPTgdNeuronTrainBehavior, 23
  - d\_delta, 26
  - d\_learningRate, 26
  - endOfEpochAction, 26
  - singlePatternBackwardAction, 26
- ADAPTgdOutputNeuronTrainBehavior, 26
  - d\_costFunction, 29
  - endOfEpochAction, 29
  - singlePatternBackwardAction, 29
- ADAPTgdwmHiddenNeuronTrainBehavior, 29
  - endOfEpochAction, 32
  - singlePatternBackwardAction, 32
- ADAPTgdwmNetworkTrainBehavior, 32
  - train, 34
- ADAPTgdwmNeuronTrainBehavior, 35
  - d\_delta, 38
  - d\_formerBiasChange, 38
  - d\_formerWeightChange, 39
  - d\_learningRate, 39
  - d\_momentum, 39
  - endOfEpochAction, 38
  - singlePatternBackwardAction, 38
- ADAPTgdwmOutputNeuronTrainBehavior, 39
  - d\_costFunction, 42
  - endOfEpochAction, 42
  - singlePatternBackwardAction, 42
- AdaptNetworkTrainBehavior, 42
  - train, 44
- AdaptNeuronTrainBehavior, 45
  - endOfEpochAction, 46
  - singlePatternBackwardAction, 46
- addCon
  - Neuron, 167
  - SimpleNeuron, 224
- AMORE.h
  - ActivationFunctionPtr, 260
  - ActivationFunctionRef, 260
  - ConContainerPtr, 260
  - ConIteratorPtr, 260
  - ConPtr, 260
  - Handler, 260
  - LayerContainerPtr, 261
  - LayerPtr, 261
  - NetworkTrainBehaviorPtr, 261
  - NeuralCreatorPtr, 261
  - NeuralFactoryPtr, 261
  - NeuralNetworkPtr, 261
  - NeuralNetworkWeakPtr, 261
  - NeuronIteratorPtr, 261
  - NeuronPtr, 261
  - NeuronRef, 261
  - NeuronTrainBehaviorPtr, 262
  - NeuronWeakPtr, 262
  - PredictBehaviorPtr, 262
  - PredictBehaviorRef, 262
  - size\_type, 260
  - TrainingBehaviorRef, 262
- ArcTan, 47
  - Arctan, 48
  - f0, 48
  - f1, 48
- Arctan
  - ArcTan, 48
- ArcTanFactory, 49
  - ArcTanFactory, 52
  - makeActivationFunction, 52
- at
  - Container, 91
  - SimpleContainer, 198
- BATCHgdHiddenNeuronTrainBehavior, 52
  - endOfEpochAction, 55
  - singlePatternBackwardAction, 55
- BATCHgdNetworkTrainBehavior, 55
  - train, 57
- BATCHgdNeuronTrainBehavior, 58
  - d\_delta, 61
  - d\_learningRate, 61
  - d\_sum\_delta\_bias, 61
  - d\_sum\_delta\_x, 62

- endOfEpochAction, [61](#)
  - singlePatternBackwardAction, [61](#)
- BATCHgdOutputNeuronTrainBehavior, [62](#)
  - d\_costFunction, [65](#)
  - endOfEpochAction, [65](#)
  - singlePatternBackwardAction, [65](#)
- BATCHgdwmHiddenNeuronTrainBehavior, [65](#)
  - endOfEpochAction, [68](#)
  - singlePatternBackwardAction, [68](#)
- BATCHgdwmNetworkTrainBehavior, [68](#)
  - train, [70](#)
- BATCHgdwmNeuronTrainBehavior, [71](#)
  - bias, [74](#)
  - change, [74](#), [75](#)
  - delta, [75](#)
  - endOfEpochAction, [74](#)
  - momentum, [75](#)
  - rate, [75](#)
  - singlePatternBackwardAction, [74](#)
  - x, [75](#)
- BATCHgdwmOutputNeuronTrainBehavior, [75](#)
  - d\_costFunction, [78](#)
  - endOfEpochAction, [78](#)
  - singlePatternBackwardAction, [78](#)
- BatchNetworkTrainBehavior, [78](#)
  - train, [80](#)
- BatchNeuronTrainBehavior, [81](#)
  - endOfEpochAction, [82](#)
  - singlePatternBackwardAction, [82](#)
- bias
  - BATCHgdwmNeuronTrainBehavior, [74](#)
- change
  - BATCHgdwmNeuronTrainBehavior, [74](#), [75](#)
- clear
  - Container, [91](#)
  - SimpleContainer, [198](#)
- Con, [83](#)
  - Con, [84](#)
  - d\_neuron, [89](#)
  - d\_weight, [89](#)
  - getNeuron, [84](#)
  - getWeight, [85](#)
  - Id, [86](#)
  - setNeuron, [87](#)
  - setWeight, [87](#)
  - show, [87](#)
  - validate, [88](#)
- ConContainerPtr
  - AMORE.h, [260](#)
  - package.h, [314](#)
- ConIteratorPtr
  - AMORE.h, [260](#)
  - package.h, [314](#)
- ConPtr
  - AMORE.h, [260](#)
  - package.h, [314](#)
- Container, [89](#)
  - ~Container, [91](#)
  - at, [91](#)
  - clear, [91](#)
  - Container, [91](#)
  - createIterator, [91](#)
  - createReverselIterator, [91](#)
  - empty, [92](#)
  - push\_back, [92](#)
  - reserve, [92](#)
  - show, [92](#)
  - size, [92](#)
  - validate, [92](#)
- Cosine, [92](#)
  - Cosine, [94](#)
  - f0, [94](#)
  - f1, [95](#)
- CosineFactory, [95](#)
  - CosineFactory, [98](#)
  - makeActivationFunction, [98](#)
- CostFunction, [98](#)
  - f0, [99](#)
  - f1, [99](#)
- createFeedForwardNetwork
  - NetworkRinterface, [148](#)
  - NeuralCreator, [155](#)
  - SimpleNeuralCreator, [218](#)
- createIterator
  - Container, [91](#)
  - SimpleContainer, [199](#)
- createReverselIterator
  - Container, [91](#)
  - SimpleContainer, [199](#)
- currentItem
  - Iterator, [125](#)
  - SimpleContainerIterator, [203](#)
  - SimpleContainerReverselIterator, [207](#)
- d\_activationFunction
  - Neuron, [169](#)
- d\_altitude

- RBFbehavior, 185
- d\_bias
  - MLPbehavior, 139
- d\_collection
  - SimpleContainer, 200
- d\_container
  - SimpleContainerIterator, 204
  - SimpleContainerReverseliterator, 208
- d\_costFunction
  - ADAPTgdOutputNeuronTrainBehavior, 29
  - ADAPTgdwmOutputNeuronTrainBehavior, 42
  - BATCHgdOutputNeuronTrainBehavior, 65
  - BATCHgdwmOutputNeuronTrainBehavior, 78
  - NetworkTrainBehavior, 154
- d\_current
  - SimpleContainerIterator, 204
  - SimpleContainerReverseliterator, 208
- d\_delta
  - ADAPTgdNeuronTrainBehavior, 26
  - ADAPTgdwmNeuronTrainBehavior, 38
  - BATCHgdNeuronTrainBehavior, 61
- d\_formerBiasChange
  - ADAPTgdwmNeuronTrainBehavior, 38
- d\_formerWeightChange
  - ADAPTgdwmNeuronTrainBehavior, 39
- d\_hiddenLayers
  - NeuralNetwork, 163
- d\_id
  - Neuron, 169
- d\_inducedLocalField
  - Neuron, 169
- d\_inputLayer
  - NeuralNetwork, 163
- d\_learningRate
  - ADAPTgdNeuronTrainBehavior, 26
  - ADAPTgdwmNeuronTrainBehavior, 39
  - BATCHgdNeuronTrainBehavior, 61
- d\_momentum
  - ADAPTgdwmNeuronTrainBehavior, 39
- d\_nCons
  - Neuron, 169
- d\_networkTrainBehavior
  - NeuralNetwork, 163
- d\_neuralNetwork
  - NetworkRinterface, 152
  - NetworkTrainBehavior, 154
- d\_neuron
  - ActivationFunction, 17
  - Con, 89
  - NeuronTrainBehavior, 171
  - PredictBehavior, 176
- d\_neuronTrainBehavior
  - Neuron, 169
- d\_output
  - Neuron, 170
- d\_outputDerivative
  - Neuron, 170
- d\_outputLayer
  - NeuralNetwork, 163
- d\_predictBehavior
  - Neuron, 170
- d\_STao
  - Tao, 251
- d\_sum\_delta\_bias
  - BATCHgdNeuronTrainBehavior, 61
- d\_sum\_delta\_x
  - BATCHgdNeuronTrainBehavior, 62
- d\_weight
  - Con, 89
- d\_width
  - RBFbehavior, 185
- delta
  - BATCHgdwmNeuronTrainBehavior, 75
- Elliot, 100
  - Elliot, 101
  - f0, 101
  - f1, 102
- ElliotFactory, 102
  - ElliotFactory, 105
  - makeActivationFunction, 105
- empty
  - Container, 92
  - SimpleContainer, 199
- endOfEpochAction
  - ADAPTgdHiddenNeuronTrainBehavior, 20
  - ADAPTgdNeuronTrainBehavior, 26
  - ADAPTgdOutputNeuronTrainBehavior, 29
  - ADAPTgdwmHiddenNeuronTrainBehavior, 32
  - ADAPTgdwmNeuronTrainBehavior, 38
  - ADAPTgdwmOutputNeuronTrainBehavior, 42
  - AdaptNeuronTrainBehavior, 46

- BATCHgdHiddenNeuronTrainBehavior, 55
- BATCHgdNeuronTrainBehavior, 61
- BATCHgdOutputNeuronTrainBehavior, 65
- BATCHgdwHiddenNeuronTrainBehavior, 68
- BATCHgdwNeuronTrainBehavior, 74
- BATCHgdwOutputNeuronTrainBehavior-first, 78
- BatchNeuronTrainBehavior, 82
- NeuronTrainBehavior, 171
- Exponential, 105
  - Exponential, 107
  - f0, 107
  - f1, 108
- ExponentialFactory, 108
  - ExponentialFactory, 111
  - makeActivationFunction, 111
- f0
  - ActivationFunction, 16
  - ArcTan, 48
  - Cosine, 94
  - CostFunction, 99
  - Elliot, 101
  - Exponential, 107
  - Gauss, 113
  - Identity, 120
  - LMLS, 127
  - LMS, 129
  - Logistic, 131
  - RadialBasis, 178
  - Reciprocal, 191
  - Sine, 231
  - Square, 237
  - Tanh, 244
  - Tao, 250
  - Threshold, 252
- f1
  - ActivationFunction, 16
  - ArcTan, 48
  - Cosine, 95
  - CostFunction, 99
  - Elliot, 102
  - Exponential, 108
  - Gauss, 114
  - Identity, 120
  - LMLS, 127
  - LMS, 129
- Logistic, 132
- RadialBasis, 179
- Reciprocal, 192
- Sine, 232
- Square, 238
- Tanh, 244
- Tao, 250
- Threshold, 253
- Iterator, 125
- SimpleContainerIterator, 203
- SimpleContainerReverselIterator, 207
- Gauss, 111
  - f0, 113
  - f1, 114
- Gauss, 113
- GaussFactory, 114
  - GaussFactory, 117
  - makeActivationFunction, 117
- getConlIterator
  - Neuron, 167
  - PredictBehavior, 173
  - SimpleNeuron, 224
- getId
  - Neuron, 167
  - SimpleNeuron, 224
- getInducedLocalField
  - ActivationFunction, 16
  - Neuron, 167
  - SimpleNeuron, 225
- getNeuron
  - Con, 84
- getOutput
  - Neuron, 167
  - SimpleNeuron, 225
- getWeight
  - Con, 85
- Handler
  - AMORE.h, 260
  - package.h, 314
- Id
  - Con, 86
- Identity, 117
  - f0, 120
  - f1, 120
- Identity, 119
- IdentityFactory, 120

- IdentityFactory, 123
- makeActivationFunction, 123
- inputSize
  - NetworkRinterface, 148
  - NeuralNetwork, 162
  - SimpleNetwork, 211
- isDone
  - Iterator, 125
  - SimpleContainerIterator, 203
  - SimpleContainerReverselIterator, 207
- Iterator, 123
  - ~Iterator, 125
  - currentItem, 125
  - first, 125
  - isDone, 125
  - Iterator, 125
  - next, 125
- LayerContainerPtr
  - AMORE.h, 261
  - package.h, 314
- LayerPtr
  - AMORE.h, 261
  - package.h, 314
- LMLS, 125
  - f0, 127
  - f1, 127
- LMS, 128
  - f0, 129
  - f1, 129
- Logistic, 130
  - f0, 131
  - f1, 132
  - Logistic, 131
- LogisticFactory, 132
  - LogisticFactory, 135
  - makeActivationFunction, 135
- makeActivationFunction
  - ArcTanFactory, 52
  - CosineFactory, 98
  - ElliotFactory, 105
  - ExponentialFactory, 111
  - GaussFactory, 117
  - IdentityFactory, 123
  - LogisticFactory, 135
  - MLPfactory, 142
  - NeuralFactory, 156
  - RadialBasisFactory, 182
  - RBFfactory, 188
  - ReciprocalFactory, 195
  - SineFactory, 235
  - SquareFactory, 241
  - TanhFactory, 248
  - ThresholdFactory, 256
- makeCon
  - MLPfactory, 142
  - NeuralFactory, 156
  - RBFfactory, 188
- makeConContainer
  - MLPfactory, 143
  - NeuralFactory, 157
  - RBFfactory, 188
- makeLayer
  - MLPfactory, 143
  - NeuralFactory, 157
  - RBFfactory, 188
- makeLayerContainer
  - MLPfactory, 143
  - NeuralFactory, 157
  - RBFfactory, 188
- makeNeuralCreator
  - MLPfactory, 144
  - NeuralFactory, 158
  - RBFfactory, 188
- makeNeuralNetwork
  - MLPfactory, 144
  - NeuralFactory, 158
  - RBFfactory, 188
- makeNeuron
  - MLPfactory, 144, 145
  - NeuralFactory, 158, 159
  - RBFfactory, 189
- makePredictBehavior
  - MLPfactory, 146
  - NeuralFactory, 159
  - RBFfactory, 189
- MLPbehavior, 135
  - d\_bias, 139
  - MLPbehavior, 138
  - MLPfactory, 139
  - show, 138
  - singlePatternForwardAction, 138
- MLPfactory, 140
  - makeActivationFunction, 142
  - makeCon, 142
  - makeConContainer, 143
  - makeLayer, 143
  - makeLayerContainer, 143
  - makeNeuralCreator, 144

- makeNeuralNetwork, 144
- makeNeuron, 144, 145
- makePredictBehavior, 146
- MLPbehavior, 139
- Neuron, 169
- momentum
  - BATCHGdwmNeuronTrainBehavior, 75
- NetworkRinterface, 147
  - createFeedForwardNetwork, 148
  - d\_neuralNetwork, 152
  - inputSize, 148
  - NetworkRinterface, 148
  - outputSize, 149
  - predict, 149
  - show, 151
  - train, 151
  - validate, 152
- NetworkTrainBehavior, 153
  - d\_costFunction, 154
  - d\_neuralNetwork, 154
  - train, 153
- NetworkTrainBehaviorPtr
  - AMORE.h, 261
  - package.h, 314
- NeuralCreator, 154
  - createFeedForwardNetwork, 155
- NeuralCreatorPtr
  - AMORE.h, 261
  - package.h, 315
- NeuralFactory, 156
  - makeActivationFunction, 156
  - makeCon, 156
  - makeConContainer, 157
  - makeLayer, 157
  - makeLayerContainer, 157
  - makeNeuralCreator, 158
  - makeNeuralNetwork, 158
  - makeNeuron, 158, 159
  - makePredictBehavior, 159
- NeuralFactoryPtr
  - AMORE.h, 261
  - package.h, 315
- NeuralNetwork, 159
  - d\_hiddenLayers, 163
  - d\_inputLayer, 163
  - d\_networkTrainBehavior, 163
  - d\_outputLayer, 163
  - inputSize, 162
  - NeuralNetwork, 161
  - outputSize, 162
  - readOutput, 162
  - show, 162
  - SimpleNeuralCreator, 163
  - singlePatternBackwardAction, 162
  - singlePatternForwardAction, 162
  - train, 162
  - validate, 163
  - writeInput, 163
- NeuralNetworkPtr
  - AMORE.h, 261
  - package.h, 315
- NeuralNetworkWeakPtr
  - AMORE.h, 261
  - package.h, 315
- Neuron, 164
  - addCon, 167
  - d\_activationFunction, 169
  - d\_Id, 169
  - d\_inducedLocalField, 169
  - d\_nCons, 169
  - d\_neuronTrainBehavior, 169
  - d\_output, 170
  - d\_outputDerivative, 170
  - d\_predictBehavior, 170
  - getConIterator, 167
  - getId, 167
  - getInducedLocalField, 167
  - getOutput, 167
  - MLPfactory, 169
  - Neuron, 167
  - setActivationFunction, 167
  - setId, 168
  - setInducedLocalField, 168
  - setOutput, 168
  - setOutputDerivative, 168
  - setPredictBehavior, 168
  - show, 168
  - singlePatternBackwardAction, 168
  - singlePatternForwardAction, 168
  - useActivationFunctionf0, 168
  - useActivationFunctionf1, 168
  - validate, 169
- NeuronIteratorPtr
  - AMORE.h, 261
  - package.h, 315
- NeuronPtr
  - AMORE.h, 261
  - package.h, 315
- NeuronRef

- AMORE.h, 261
  - package.h, 315
- NeuronTrainBehavior, 170
  - d\_neuron, 171
  - endOfEpochAction, 171
  - singlePatternBackwardAction, 171
- NeuronTrainBehaviorPtr
  - AMORE.h, 262
  - package.h, 315
- NeuronWeakPtr
  - AMORE.h, 262
  - package.h, 315
- next
  - Iterator, 125
  - SimpleContainerIterator, 203
  - SimpleContainerReverselIterator, 207
- outputSize
  - NetworkRinterface, 149
  - NeuralNetwork, 162
  - SimpleNetwork, 212
- package.h
  - ActivationFunctionPtr, 314
  - ActivationFunctionRef, 314
  - ConContainerPtr, 314
  - ConIteratorPtr, 314
  - ConPtr, 314
  - Handler, 314
  - LayerContainerPtr, 314
  - LayerPtr, 314
  - NetworkTrainBehaviorPtr, 314
  - NeuralCreatorPtr, 315
  - NeuralFactoryPtr, 315
  - NeuralNetworkPtr, 315
  - NeuralNetworkWeakPtr, 315
  - NeuronIteratorPtr, 315
  - NeuronPtr, 315
  - NeuronRef, 315
  - NeuronTrainBehaviorPtr, 315
  - NeuronWeakPtr, 315
  - PredictBehaviorPtr, 315
  - PredictBehaviorRef, 316
  - size\_type, 314
- predict
  - NetworkRinterface, 149
- PredictBehavior, 172
  - d\_neuron, 176
  - getConIterator, 173
  - PredictBehavior, 173
  - setInducedLocalField, 174
  - setOutput, 174
  - setOutputDerivative, 175
  - show, 175
  - singlePatternForwardAction, 175
  - useActivationFunctionf0, 175
  - useActivationFunctionf1, 176
- PredictBehaviorPtr
  - AMORE.h, 262
  - package.h, 315
- PredictBehaviorRef
  - AMORE.h, 262
  - package.h, 316
- push\_back
  - Container, 92
  - SimpleContainer, 199
- RadialBasis, 177
  - f0, 178
  - f1, 179
  - RadialBasis, 178
- RadialBasisFactory, 179
  - makeActivationFunction, 182
  - RadialBasisFactory, 182
- rate
  - BATCHgdwmNeuronTrainBehavior, 75
- RBFbehavior, 182
  - d\_altitude, 185
  - d\_width, 185
  - RBFbehavior, 185
  - show, 185
  - singlePatternForwardAction, 185
- RBFfactory, 185
  - makeActivationFunction, 188
  - makeCon, 188
  - makeConContainer, 188
  - makeLayer, 188
  - makeLayerContainer, 188
  - makeNeuralCreator, 188
  - makeNeuralNetwork, 188
  - makeNeuron, 189
  - makePredictBehavior, 189
- Rcpp\_MODULE
  - RcppModules.cpp, 317
- RcppModules.cpp
  - Rcpp\_MODULE, 317
- readOutput
  - NeuralNetwork, 162
  - SimpleNetwork, 212
- Reciprocal, 189



- f0, [191](#)
- f1, [192](#)
- Reciprocal, [191](#)
- ReciprocalFactory, [192](#)
  - makeActivationFunction, [195](#)
  - ReciprocalFactory, [195](#)
- reserve
  - Container, [92](#)
  - SimpleContainer, [199](#)
- setActivationFunction
  - Neuron, [167](#)
  - SimpleNeuron, [225](#)
- setId
  - Neuron, [168](#)
  - SimpleNeuron, [226](#)
- setInducedLocalField
  - Neuron, [168](#)
  - PredictBehavior, [174](#)
  - SimpleNeuron, [226](#)
- setNeuron
  - Con, [87](#)
- setOutput
  - Neuron, [168](#)
  - PredictBehavior, [174](#)
  - SimpleNeuron, [226](#)
- setOutputDerivative
  - Neuron, [168](#)
  - PredictBehavior, [175](#)
  - SimpleNeuron, [226](#)
- setPredictBehavior
  - Neuron, [168](#)
  - SimpleNeuron, [227](#)
- setWeight
  - Con, [87](#)
- show
  - Con, [87](#)
  - Container, [92](#)
  - MLPbehavior, [138](#)
  - NetworkRinterface, [151](#)
  - NeuralNetwork, [162](#)
  - Neuron, [168](#)
  - PredictBehavior, [175](#)
  - RBFbehavior, [185](#)
  - SimpleContainer, [199](#)
  - SimpleNetwork, [213](#)
  - SimpleNeuron, [227](#)
- SimpleContainer, [195](#)
  - ~SimpleContainer, [198](#)
  - at, [198](#)
  - clear, [198](#)
  - createIterator, [199](#)
  - createReverselIterator, [199](#)
  - d\_collection, [200](#)
  - empty, [199](#)
  - push\_back, [199](#)
  - reserve, [199](#)
  - show, [199](#)
  - SimpleContainer, [198](#)
  - SimpleContainerIterator< T >, [200](#)
  - SimpleContainerReverselIterator< T >, [200](#)
  - size, [199](#)
  - validate, [199](#)
- SimpleContainer< T >
  - SimpleContainerIterator, [204](#)
  - SimpleContainerReverselIterator, [208](#)
- SimpleContainerIterator, [200](#)
  - ~SimpleContainerIterator, [203](#)
  - currentItem, [203](#)
  - d\_container, [204](#)
  - d\_current, [204](#)
  - first, [203](#)
  - isDone, [203](#)
  - next, [203](#)
  - SimpleContainer< T >, [204](#)
  - SimpleContainerIterator, [203](#)
- SimpleContainerIterator< T >
  - SimpleContainer, [200](#)
- SimpleContainerReverselIterator, [204](#)
  - ~SimpleContainerReverselIterator, [207](#)
  - currentItem, [207](#)
  - d\_container, [208](#)
  - d\_current, [208](#)
  - first, [207](#)
  - isDone, [207](#)
  - next, [207](#)
  - SimpleContainer< T >, [208](#)
  - SimpleContainerReverselIterator, [207](#)
- SimpleContainerReverselIterator< T >
  - SimpleContainer, [200](#)
- SimpleNetwork, [208](#)
  - inputSize, [211](#)
  - outputSize, [212](#)
  - readOutput, [212](#)
  - show, [213](#)
  - SimpleNetwork, [211](#)
  - singlePatternBackwardAction, [213](#)
  - singlePatternForwardAction, [214](#)
  - train, [215](#)

- validate, [215](#)
- writelnInput, [215](#)
- SimpleNeuralCreator, [216](#)
  - createFeedForwardNetwork, [218](#)
  - NeuralNetwork, [163](#)
  - SimpleNeuralCreator, [217](#)
- SimpleNeuron, [219](#)
  - addCon, [224](#)
  - getConlterator, [224](#)
  - getId, [224](#)
  - getInducedLocalField, [225](#)
  - getOutput, [225](#)
  - setActivationFunction, [225](#)
  - setId, [226](#)
  - setInducedLocalField, [226](#)
  - setOutput, [226](#)
  - setOutputDerivative, [226](#)
  - setPredictBehavior, [227](#)
  - show, [227](#)
  - SimpleNeuron, [223](#)
  - singlePatternBackwardAction, [228](#)
  - singlePatternForwardAction, [228](#)
  - useActivationFunctionf0, [228](#)
  - useActivationFunctionf1, [229](#)
  - validate, [229](#)
- Sine, [230](#)
  - f0, [231](#)
  - f1, [232](#)
  - Sine, [231](#)
- SineFactory, [232](#)
  - makeActivationFunction, [235](#)
  - SineFactory, [235](#)
- singlePatternBackwardAction
  - ADAPTgdHiddenNeuronTrainBehavior, [20](#)
  - ADAPTgdNeuronTrainBehavior, [26](#)
  - ADAPTgdOutputNeuronTrainBehavior, [29](#)
  - ADAPTgdwmHiddenNeuronTrainBehavior, [32](#)
  - ADAPTgdwmNeuronTrainBehavior, [38](#)
  - ADAPTgdwmOutputNeuronTrainBehavior, [42](#)
  - AdaptNeuronTrainBehavior, [46](#)
  - BATCHgdHiddenNeuronTrainBehavior, [55](#)
  - BATCHgdNeuronTrainBehavior, [61](#)
  - BATCHgdOutputNeuronTrainBehavior, [65](#)
  - BATCHgdwmHiddenNeuronTrainBehavior, [68](#)
  - BATCHgdwmNeuronTrainBehavior, [74](#)
  - BATCHgdwmOutputNeuronTrainBehavior, [78](#)
  - BatchNeuronTrainBehavior, [82](#)
  - NeuralNetwork, [162](#)
  - Neuron, [168](#)
  - NeuronTrainBehavior, [171](#)
  - SimpleNetwork, [213](#)
  - SimpleNeuron, [228](#)
- singlePatternForwardAction
  - MLPbehavior, [138](#)
  - NeuralNetwork, [162](#)
  - Neuron, [168](#)
  - PredictBehavior, [175](#)
  - RBFbehavior, [185](#)
  - SimpleNetwork, [214](#)
  - SimpleNeuron, [228](#)
- size
  - Container, [92](#)
  - SimpleContainer, [199](#)
- size\_type
  - AMORE.h, [260](#)
  - package.h, [314](#)
- Square, [235](#)
  - f0, [237](#)
  - f1, [238](#)
  - Square, [237](#)
- SquareFactory, [238](#)
  - makeActivationFunction, [241](#)
  - SquareFactory, [241](#)
- Tanh, [241](#)
  - f0, [244](#)
  - f1, [244](#)
  - Tanh, [243](#)
  - TanhFactory, [245](#)
  - makeActivationFunction, [248](#)
  - TanhFactory, [248](#)
- Tao, [248](#)
  - d\_STao, [251](#)
  - f0, [250](#)
  - f1, [250](#)
- Threshold, [251](#)
  - f0, [252](#)
  - f1, [253](#)
  - Threshold, [252](#)
  - ThresholdFactory, [253](#)
  - makeActivationFunction, [256](#)

- ThresholdFactory, [256](#)
- train
  - ADAPTgdNetworkTrainBehavior, [22](#)
  - ADAPTgdwmNetworkTrainBehavior, [34](#)
  - AdaptNetworkTrainBehavior, [44](#)
  - BATCHgdNetworkTrainBehavior, [57](#)
  - BATCHgdwmNetworkTrainBehavior, [70](#)
  - BatchNetworkTrainBehavior, [80](#)
  - NetworkRinterface, [151](#)
  - NetworkTrainBehavior, [153](#)
  - NeuralNetwork, [162](#)
  - SimpleNetwork, [215](#)
- TrainingBehaviorRef
  - AMORE.h, [262](#)
- useActivationFunction0
  - Neuron, [168](#)
  - PredictBehavior, [175](#)
  - SimpleNeuron, [228](#)
- useActivationFunction1
  - Neuron, [168](#)
  - PredictBehavior, [176](#)
  - SimpleNeuron, [229](#)
- validate
  - Con, [88](#)
  - Container, [92](#)
  - NetworkRinterface, [152](#)
  - NeuralNetwork, [163](#)
  - Neuron, [169](#)
  - SimpleContainer, [199](#)
  - SimpleNetwork, [215](#)
  - SimpleNeuron, [229](#)
- writelnInput
  - NeuralNetwork, [163](#)
  - SimpleNetwork, [215](#)
- x
  - BATCHgdwmNeuronTrainBehavior, [75](#)