

AMORE++

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

Generated by Doxygen 1.7.4

Sat Jul 30 2011 04:41:16



# 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>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Class Documentation</b>	<b>11</b>
5.1	ActivationFunction Class Reference . . . . .	11
5.1.1	Detailed Description . . . . .	12
5.1.2	Constructor & Destructor Documentation . . . . .	12
5.1.2.1	ActivationFunction . . . . .	12
5.1.3	Member Function Documentation . . . . .	12
5.1.3.1	f0 . . . . .	12
5.1.3.2	f1 . . . . .	12
5.1.3.3	getInducedLocalField . . . . .	12
5.1.4	Member Data Documentation . . . . .	13
5.1.4.1	d_neuron . . . . .	13
5.2	ADAPTgdNetworkTrainBehavior Class Reference . . . . .	13
5.2.1	Detailed Description . . . . .	15

5.2.2	Member Function Documentation	15
5.2.2.1	train	15
5.3	ADAPTgdNeuronTrainBehavior Class Reference	16
5.3.1	Detailed Description	19
5.3.2	Member Function Documentation	19
5.3.2.1	endOfEpochAction	19
5.3.2.2	singlePatternBackwardAction	19
5.3.3	Member Data Documentation	19
5.3.3.1	delta	19
5.3.3.2	rate	19
5.4	ADAPTgdwmNetworkTrainBehavior Class Reference	19
5.4.1	Detailed Description	21
5.4.2	Member Function Documentation	21
5.4.2.1	train	21
5.5	ADAPTgdwmNeuronTrainBehavior Class Reference	22
5.5.1	Detailed Description	24
5.5.2	Member Function Documentation	24
5.5.2.1	endOfEpochAction	24
5.5.2.2	singlePatternBackwardAction	24
5.5.3	Member Data Documentation	24
5.5.3.1	change	24
5.5.3.2	change	24
5.5.3.3	delta	25
5.5.3.4	momentum	25
5.5.3.5	rate	25
5.6	AdaptNetworkTrainBehavior Class Reference	25
5.6.1	Detailed Description	27
5.6.2	Member Function Documentation	27
5.6.2.1	train	27
5.7	AdaptNeuronTrainBehavior Class Reference	28
5.7.1	Detailed Description	29
5.7.2	Member Function Documentation	29
5.7.2.1	endOfEpochAction	29
5.7.2.2	singlePatternBackwardAction	30

5.8	ArcTan Class Reference . . . . .	30
5.8.1	Detailed Description . . . . .	31
5.8.2	Member Function Documentation . . . . .	31
5.8.2.1	Arctan . . . . .	31
5.8.2.2	f0 . . . . .	31
5.8.2.3	f1 . . . . .	32
5.9	ArcTanFactory Class Reference . . . . .	32
5.9.1	Detailed Description . . . . .	35
5.9.2	Constructor & Destructor Documentation . . . . .	35
5.9.2.1	ArcTanFactory . . . . .	35
5.9.3	Member Function Documentation . . . . .	35
5.9.3.1	makeActivationFunction . . . . .	35
5.10	BATCHgdNetworkTrainBehavior Class Reference . . . . .	35
5.10.1	Detailed Description . . . . .	37
5.10.2	Member Function Documentation . . . . .	37
5.10.2.1	train . . . . .	37
5.11	BATCHgdNeuronTrainBehavior Class Reference . . . . .	38
5.11.1	Detailed Description . . . . .	40
5.11.2	Member Function Documentation . . . . .	40
5.11.2.1	endOfEpochAction . . . . .	40
5.11.2.2	singlePatternBackwardAction . . . . .	40
5.11.3	Member Data Documentation . . . . .	40
5.11.3.1	bias . . . . .	40
5.11.3.2	delta . . . . .	40
5.11.3.3	rate . . . . .	40
5.11.3.4	x . . . . .	41
5.12	BATCHgdwmNetworkTrainBehavior Class Reference . . . . .	41
5.12.1	Detailed Description . . . . .	42
5.12.2	Member Function Documentation . . . . .	42
5.12.2.1	train . . . . .	42
5.13	BATCHgdwmNeuronTrainBehavior Class Reference . . . . .	43
5.13.1	Detailed Description . . . . .	45
5.13.2	Member Function Documentation . . . . .	45
5.13.2.1	endOfEpochAction . . . . .	45

5.13.2.2	<a href="#">singlePatternBackwardAction</a>	45
5.13.3	<a href="#">Member Data Documentation</a>	45
5.13.3.1	<a href="#">bias</a>	45
5.13.3.2	<a href="#">change</a>	45
5.13.3.3	<a href="#">change</a>	46
5.13.3.4	<a href="#">delta</a>	46
5.13.3.5	<a href="#">momentum</a>	46
5.13.3.6	<a href="#">rate</a>	46
5.13.3.7	<a href="#">x</a>	46
5.14	<a href="#">BatchNetworkTrainBehavior Class Reference</a>	46
5.14.1	<a href="#">Detailed Description</a>	48
5.14.2	<a href="#">Member Function Documentation</a>	48
5.14.2.1	<a href="#">train</a>	48
5.15	<a href="#">BatchNeuronTrainBehavior Class Reference</a>	49
5.15.1	<a href="#">Detailed Description</a>	50
5.15.2	<a href="#">Member Function Documentation</a>	50
5.15.2.1	<a href="#">endOfEpochAction</a>	50
5.15.2.2	<a href="#">singlePatternBackwardAction</a>	51
5.16	<a href="#">Con Class Reference</a>	51
5.16.1	<a href="#">Detailed Description</a>	51
5.16.2	<a href="#">Constructor &amp; Destructor Documentation</a>	52
5.16.2.1	<a href="#">Con</a>	52
5.16.2.2	<a href="#">Con</a>	52
5.16.3	<a href="#">Member Function Documentation</a>	52
5.16.3.1	<a href="#">getNeuron</a>	52
5.16.3.2	<a href="#">getWeight</a>	53
5.16.3.3	<a href="#">ld</a>	54
5.16.3.4	<a href="#">setNeuron</a>	55
5.16.3.5	<a href="#">setWeight</a>	55
5.16.3.6	<a href="#">show</a>	55
5.16.3.7	<a href="#">validate</a>	56
5.16.4	<a href="#">Member Data Documentation</a>	57
5.16.4.1	<a href="#">d_neuron</a>	57
5.16.4.2	<a href="#">d_weight</a>	57

5.17 Container< T > Class Template Reference . . . . .	57
5.17.1 Detailed Description . . . . .	59
5.17.2 Constructor & Destructor Documentation . . . . .	59
5.17.2.1 ~Container . . . . .	59
5.17.2.2 Container . . . . .	59
5.17.3 Member Function Documentation . . . . .	59
5.17.3.1 at . . . . .	59
5.17.3.2 clear . . . . .	59
5.17.3.3 createliterator . . . . .	59
5.17.3.4 createReverseliterator . . . . .	60
5.17.3.5 empty . . . . .	60
5.17.3.6 push_back . . . . .	60
5.17.3.7 reserve . . . . .	60
5.17.3.8 show . . . . .	60
5.17.3.9 size . . . . .	60
5.17.3.10 validate . . . . .	60
5.18 Cosine Class Reference . . . . .	60
5.18.1 Detailed Description . . . . .	62
5.18.2 Constructor & Destructor Documentation . . . . .	62
5.18.2.1 Cosine . . . . .	62
5.18.3 Member Function Documentation . . . . .	62
5.18.3.1 f0 . . . . .	63
5.18.3.2 f1 . . . . .	63
5.19 CosineFactory Class Reference . . . . .	63
5.19.1 Detailed Description . . . . .	66
5.19.2 Constructor & Destructor Documentation . . . . .	66
5.19.2.1 CosineFactory . . . . .	66
5.19.3 Member Function Documentation . . . . .	66
5.19.3.1 makeActivationFunction . . . . .	66
5.20 Elliot Class Reference . . . . .	66
5.20.1 Detailed Description . . . . .	68
5.20.2 Constructor & Destructor Documentation . . . . .	68
5.20.2.1 Elliot . . . . .	68
5.20.3 Member Function Documentation . . . . .	68

5.20.3.1	f0	69
5.20.3.2	f1	69
5.21	ElliotFactory Class Reference	69
5.21.1	Detailed Description	72
5.21.2	Constructor & Destructor Documentation	72
5.21.2.1	ElliotFactory	72
5.21.3	Member Function Documentation	72
5.21.3.1	makeActivationFunction	72
5.22	Exponential Class Reference	72
5.22.1	Detailed Description	74
5.22.2	Constructor & Destructor Documentation	74
5.22.2.1	Exponential	74
5.22.3	Member Function Documentation	74
5.22.3.1	f0	75
5.22.3.2	f1	75
5.23	ExponentialFactory Class Reference	75
5.23.1	Detailed Description	78
5.23.2	Constructor & Destructor Documentation	78
5.23.2.1	ExponentialFactory	78
5.23.3	Member Function Documentation	78
5.23.3.1	makeActivationFunction	78
5.24	Gauss Class Reference	78
5.24.1	Detailed Description	80
5.24.2	Constructor & Destructor Documentation	80
5.24.2.1	Gauss	80
5.24.3	Member Function Documentation	80
5.24.3.1	f0	81
5.24.3.2	f1	81
5.25	GaussFactory Class Reference	81
5.25.1	Detailed Description	84
5.25.2	Constructor & Destructor Documentation	84
5.25.2.1	GaussFactory	84
5.25.3	Member Function Documentation	84
5.25.3.1	makeActivationFunction	84



5.26 Identity Class Reference . . . . .	84
5.26.1 Detailed Description . . . . .	86
5.26.2 Constructor & Destructor Documentation . . . . .	86
5.26.2.1 Identity . . . . .	86
5.26.3 Member Function Documentation . . . . .	87
5.26.3.1 f0 . . . . .	87
5.26.3.2 f1 . . . . .	87
5.27 IdentityFactory Class Reference . . . . .	87
5.27.1 Detailed Description . . . . .	90
5.27.2 Constructor & Destructor Documentation . . . . .	90
5.27.2.1 IdentityFactory . . . . .	90
5.27.3 Member Function Documentation . . . . .	90
5.27.3.1 makeActivationFunction . . . . .	90
5.28 Iterator< T > Class Template Reference . . . . .	90
5.28.1 Detailed Description . . . . .	91
5.28.2 Constructor & Destructor Documentation . . . . .	92
5.28.2.1 ~Iterator . . . . .	92
5.28.2.2 Iterator . . . . .	92
5.28.3 Member Function Documentation . . . . .	92
5.28.3.1 currentItem . . . . .	92
5.28.3.2 first . . . . .	92
5.28.3.3 isDone . . . . .	92
5.28.3.4 next . . . . .	92
5.29 Logistic Class Reference . . . . .	92
5.29.1 Detailed Description . . . . .	94
5.29.2 Constructor & Destructor Documentation . . . . .	94
5.29.2.1 Logistic . . . . .	94
5.29.3 Member Function Documentation . . . . .	94
5.29.3.1 f0 . . . . .	95
5.29.3.2 f1 . . . . .	95
5.30 LogisticFactory Class Reference . . . . .	95
5.30.1 Detailed Description . . . . .	98
5.30.2 Constructor & Destructor Documentation . . . . .	98
5.30.2.1 LogisticFactory . . . . .	98

5.30.3	Member Function Documentation	98
5.30.3.1	makeActivationFunction	98
5.31	MLPbehavior Class Reference	98
5.31.1	Detailed Description	101
5.31.2	Constructor & Destructor Documentation	101
5.31.2.1	MLPbehavior	101
5.31.3	Member Function Documentation	101
5.31.3.1	show	101
5.31.3.2	singlePatternForwardAction	101
5.31.4	Friends And Related Function Documentation	102
5.31.4.1	MLPfactory	102
5.31.5	Member Data Documentation	102
5.31.5.1	d_bias	102
5.32	MLPfactory Class Reference	103
5.32.1	Detailed Description	105
5.32.2	Member Function Documentation	105
5.32.2.1	makeActivationFunction	105
5.32.2.2	makeCon	105
5.32.2.3	makeConContainer	106
5.32.2.4	makeLayer	106
5.32.2.5	makeLayerContainer	107
5.32.2.6	makeNeuralCreator	107
5.32.2.7	makeNeuralNetwork	107
5.32.2.8	makeNeuron	108
5.32.2.9	makeNeuron	108
5.32.2.10	makePredictBehavior	109
5.33	NetworkRinterface Class Reference	110
5.33.1	Detailed Description	110
5.33.2	Constructor & Destructor Documentation	111
5.33.2.1	NetworkRinterface	111
5.33.3	Member Function Documentation	111
5.33.3.1	createFeedForwardNetwork	111
5.33.3.2	inputSize	111
5.33.3.3	outputSize	112

5.33.3.4	<a href="#">predict</a>	112
5.33.3.5	<a href="#">show</a>	114
5.33.3.6	<a href="#">train</a>	114
5.33.3.7	<a href="#">validate</a>	115
5.33.4	<a href="#">Member Data Documentation</a>	115
5.33.4.1	<a href="#">d_neuralNetwork</a>	115
5.34	<a href="#">NetworkTrainBehavior Class Reference</a>	116
5.34.1	<a href="#">Detailed Description</a>	116
5.34.2	<a href="#">Member Function Documentation</a>	116
5.34.2.1	<a href="#">train</a>	117
5.34.3	<a href="#">Member Data Documentation</a>	117
5.34.3.1	<a href="#">d_neuralNetwork</a>	117
5.35	<a href="#">NeuralCreator Class Reference</a>	117
5.35.1	<a href="#">Detailed Description</a>	118
5.35.2	<a href="#">Member Function Documentation</a>	118
5.35.2.1	<a href="#">createFeedForwardNetwork</a>	118
5.36	<a href="#">NeuralFactory Class Reference</a>	118
5.36.1	<a href="#">Detailed Description</a>	119
5.36.2	<a href="#">Member Function Documentation</a>	119
5.36.2.1	<a href="#">makeActivationFunction</a>	119
5.36.2.2	<a href="#">makeCon</a>	119
5.36.2.3	<a href="#">makeConContainer</a>	119
5.36.2.4	<a href="#">makeLayer</a>	120
5.36.2.5	<a href="#">makeLayerContainer</a>	120
5.36.2.6	<a href="#">makeNeuralCreator</a>	121
5.36.2.7	<a href="#">makeNeuralNetwork</a>	121
5.36.2.8	<a href="#">makeNeuron</a>	121
5.36.2.9	<a href="#">makeNeuron</a>	121
5.36.2.10	<a href="#">makePredictBehavior</a>	122
5.37	<a href="#">NeuralNetwork Class Reference</a>	122
5.37.1	<a href="#">Detailed Description</a>	124
5.37.2	<a href="#">Constructor &amp; Destructor Documentation</a>	124
5.37.2.1	<a href="#">NeuralNetwork</a>	124
5.37.3	<a href="#">Member Function Documentation</a>	125

5.37.3.1	<a href="#">inputSize</a>	125
5.37.3.2	<a href="#">outputSize</a>	125
5.37.3.3	<a href="#">readOutput</a>	125
5.37.3.4	<a href="#">show</a>	125
5.37.3.5	<a href="#">singlePatternBackwardAction</a>	125
5.37.3.6	<a href="#">singlePatternForwardAction</a>	125
5.37.3.7	<a href="#">train</a>	126
5.37.3.8	<a href="#">validate</a>	126
5.37.3.9	<a href="#">writeInput</a>	126
5.37.4	<a href="#">Friends And Related Function Documentation</a>	126
5.37.4.1	<a href="#">SimpleNeuralCreator</a>	126
5.37.5	<a href="#">Member Data Documentation</a>	126
5.37.5.1	<a href="#">d_hiddenLayers</a>	126
5.37.5.2	<a href="#">d_inputLayer</a>	126
5.37.5.3	<a href="#">d_networkTrainBehavior</a>	126
5.37.5.4	<a href="#">d_outputLayer</a>	127
5.38	<a href="#">Neuron Class Reference</a>	127
5.38.1	<a href="#">Detailed Description</a>	129
5.38.2	<a href="#">Constructor &amp; Destructor Documentation</a>	130
5.38.2.1	<a href="#">Neuron</a>	130
5.38.3	<a href="#">Member Function Documentation</a>	130
5.38.3.1	<a href="#">addCon</a>	130
5.38.3.2	<a href="#">getConIterator</a>	130
5.38.3.3	<a href="#">getId</a>	130
5.38.3.4	<a href="#">getInducedLocalField</a>	130
5.38.3.5	<a href="#">getOutput</a>	130
5.38.3.6	<a href="#">setActivationFunction</a>	131
5.38.3.7	<a href="#">setId</a>	131
5.38.3.8	<a href="#">setInducedLocalField</a>	131
5.38.3.9	<a href="#">setOutput</a>	131
5.38.3.10	<a href="#">setOutputDerivative</a>	131
5.38.3.11	<a href="#">setPredictBehavior</a>	131
5.38.3.12	<a href="#">show</a>	131
5.38.3.13	<a href="#">singlePatternBackwardAction</a>	131

5.38.3.14	singlePatternForwardAction	131
5.38.3.15	useActivationFunctionf0	131
5.38.3.16	useActivationFunctionf1	132
5.38.3.17	validate	132
5.38.4	Friends And Related Function Documentation	132
5.38.4.1	MLPfactory	132
5.38.5	Member Data Documentation	132
5.38.5.1	d_activationFunction	132
5.38.5.2	d_Id	132
5.38.5.3	d_inducedLocalField	132
5.38.5.4	d_nCons	132
5.38.5.5	d_neuronTrainBehavior	132
5.38.5.6	d_output	133
5.38.5.7	d_outputDerivative	133
5.38.5.8	d_predictBehavior	133
5.39	NeuronTrainBehavior Class Reference	133
5.39.1	Detailed Description	134
5.39.2	Member Function Documentation	134
5.39.2.1	endOfEpochAction	134
5.39.2.2	singlePatternBackwardAction	135
5.39.3	Member Data Documentation	135
5.39.3.1	d_neuron	135
5.40	PredictBehavior Class Reference	135
5.40.1	Detailed Description	137
5.40.2	Constructor & Destructor Documentation	137
5.40.2.1	PredictBehavior	137
5.40.3	Member Function Documentation	137
5.40.3.1	getConlterator	137
5.40.3.2	setInducedLocalField	138
5.40.3.3	setOutput	138
5.40.3.4	setOutputDerivative	139
5.40.3.5	show	139
5.40.3.6	singlePatternForwardAction	139
5.40.3.7	useActivationFunctionf0	139

5.40.3.8	<a href="#">useActivationFunctionf1</a>	140
5.40.4	<a href="#">Member Data Documentation</a>	140
5.40.4.1	<a href="#">d_neuron</a>	140
5.41	<a href="#">RadialBasis Class Reference</a>	141
5.41.1	<a href="#">Detailed Description</a>	142
5.41.2	<a href="#">Constructor &amp; Destructor Documentation</a>	142
5.41.2.1	<a href="#">RadialBasis</a>	142
5.41.3	<a href="#">Member Function Documentation</a>	142
5.41.3.1	<a href="#">f0</a>	143
5.41.3.2	<a href="#">f1</a>	143
5.42	<a href="#">RadialBasisFactory Class Reference</a>	143
5.42.1	<a href="#">Detailed Description</a>	146
5.42.2	<a href="#">Constructor &amp; Destructor Documentation</a>	146
5.42.2.1	<a href="#">RadialBasisFactory</a>	146
5.42.3	<a href="#">Member Function Documentation</a>	146
5.42.3.1	<a href="#">makeActivationFunction</a>	146
5.43	<a href="#">RBFbehavior Class Reference</a>	146
5.43.1	<a href="#">Detailed Description</a>	149
5.43.2	<a href="#">Constructor &amp; Destructor Documentation</a>	149
5.43.2.1	<a href="#">RBFbehavior</a>	149
5.43.3	<a href="#">Member Function Documentation</a>	149
5.43.3.1	<a href="#">show</a>	149
5.43.3.2	<a href="#">singlePatternForwardAction</a>	149
5.43.4	<a href="#">Member Data Documentation</a>	149
5.43.4.1	<a href="#">d_altitude</a>	149
5.43.4.2	<a href="#">d_width</a>	149
5.44	<a href="#">RBFfactory Class Reference</a>	149
5.44.1	<a href="#">Detailed Description</a>	152
5.44.2	<a href="#">Member Function Documentation</a>	152
5.44.2.1	<a href="#">makeActivationFunction</a>	152
5.44.2.2	<a href="#">makeCon</a>	152
5.44.2.3	<a href="#">makeConContainer</a>	152
5.44.2.4	<a href="#">makeLayer</a>	152
5.44.2.5	<a href="#">makeLayerContainer</a>	152

5.44.2.6	makeNeuralCreator	152
5.44.2.7	makeNeuralNetwork	153
5.44.2.8	makeNeuron	153
5.44.2.9	makeNeuron	153
5.44.2.10	makePredictBehavior	153
5.45	Reciprocal Class Reference	153
5.45.1	Detailed Description	155
5.45.2	Constructor & Destructor Documentation	155
5.45.2.1	Reciprocal	155
5.45.3	Member Function Documentation	155
5.45.3.1	f0	156
5.45.3.2	f1	156
5.46	ReciprocalFactory Class Reference	156
5.46.1	Detailed Description	159
5.46.2	Constructor & Destructor Documentation	159
5.46.2.1	ReciprocalFactory	159
5.46.3	Member Function Documentation	159
5.46.3.1	makeActivationFunction	159
5.47	SimpleContainer< T > Class Template Reference	159
5.47.1	Detailed Description	162
5.47.2	Constructor & Destructor Documentation	162
5.47.2.1	SimpleContainer	162
5.47.2.2	~SimpleContainer	162
5.47.3	Member Function Documentation	162
5.47.3.1	at	162
5.47.3.2	clear	163
5.47.3.3	createIterator	163
5.47.3.4	createReverseIterator	163
5.47.3.5	empty	163
5.47.3.6	push_back	163
5.47.3.7	reserve	163
5.47.3.8	show	163
5.47.3.9	size	163
5.47.3.10	validate	163

5.47.4	Friends And Related Function Documentation . . . . .	164
5.47.4.1	SimpleContainerIterator< T > . . . . .	164
5.47.4.2	SimpleContainerReverseliterator< T > . . . . .	164
5.47.5	Member Data Documentation . . . . .	164
5.47.5.1	d_collection . . . . .	164
5.48	SimpleContainerIterator< T > Class Template Reference . . . . .	164
5.48.1	Detailed Description . . . . .	167
5.48.2	Constructor & Destructor Documentation . . . . .	167
5.48.2.1	SimpleContainerIterator . . . . .	167
5.48.2.2	~SimpleContainerIterator . . . . .	167
5.48.3	Member Function Documentation . . . . .	167
5.48.3.1	currentItem . . . . .	167
5.48.3.2	first . . . . .	167
5.48.3.3	isDone . . . . .	167
5.48.3.4	next . . . . .	168
5.48.4	Friends And Related Function Documentation . . . . .	168
5.48.4.1	SimpleContainer< T > . . . . .	168
5.48.5	Member Data Documentation . . . . .	168
5.48.5.1	d_container . . . . .	168
5.48.5.2	d_current . . . . .	168
5.49	SimpleContainerReverseliterator< T > Class Template Reference . . . . .	168
5.49.1	Detailed Description . . . . .	171
5.49.2	Constructor & Destructor Documentation . . . . .	171
5.49.2.1	SimpleContainerReverseliterator . . . . .	171
5.49.2.2	~SimpleContainerReverseliterator . . . . .	171
5.49.3	Member Function Documentation . . . . .	171
5.49.3.1	currentItem . . . . .	171
5.49.3.2	first . . . . .	171
5.49.3.3	isDone . . . . .	171
5.49.3.4	next . . . . .	172
5.49.4	Friends And Related Function Documentation . . . . .	172
5.49.4.1	SimpleContainer< T > . . . . .	172
5.49.5	Member Data Documentation . . . . .	172
5.49.5.1	d_container . . . . .	172



5.49.5.2	d_current	172
5.50	SimpleNetwork Class Reference	172
5.50.1	Detailed Description	175
5.50.2	Constructor & Destructor Documentation	175
5.50.2.1	SimpleNetwork	175
5.50.3	Member Function Documentation	175
5.50.3.1	inputSize	175
5.50.3.2	outputSize	176
5.50.3.3	readOutput	176
5.50.3.4	show	177
5.50.3.5	singlePatternBackwardAction	177
5.50.3.6	singlePatternForwardAction	178
5.50.3.7	train	179
5.50.3.8	validate	179
5.50.3.9	writeInput	179
5.51	SimpleNeuralCreator Class Reference	180
5.51.1	Detailed Description	181
5.51.2	Constructor & Destructor Documentation	181
5.51.2.1	SimpleNeuralCreator	181
5.51.3	Member Function Documentation	182
5.51.3.1	createFeedForwardNetwork	182
5.52	SimpleNeuron Class Reference	183
5.52.1	Detailed Description	187
5.52.2	Constructor & Destructor Documentation	187
5.52.2.1	SimpleNeuron	187
5.52.3	Member Function Documentation	188
5.52.3.1	addCon	188
5.52.3.2	getConIterator	188
5.52.3.3	getId	188
5.52.3.4	getInducedLocalField	189
5.52.3.5	getOutput	189
5.52.3.6	setActivationFunction	189
5.52.3.7	setId	190
5.52.3.8	setInducedLocalField	190

5.52.3.9	setOutput	190
5.52.3.10	setOutputDerivative	190
5.52.3.11	setPredictBehavior	191
5.52.3.12	show	191
5.52.3.13	singlePatternBackwardAction	192
5.52.3.14	singlePatternForwardAction	192
5.52.3.15	useActivationFunction0	192
5.52.3.16	useActivationFunction1	193
5.52.3.17	validate	193
5.53	Sine Class Reference	194
5.53.1	Detailed Description	195
5.53.2	Constructor & Destructor Documentation	195
5.53.2.1	Sine	195
5.53.3	Member Function Documentation	195
5.53.3.1	f0	196
5.53.3.2	f1	196
5.54	SineFactory Class Reference	196
5.54.1	Detailed Description	199
5.54.2	Constructor & Destructor Documentation	199
5.54.2.1	SineFactory	199
5.54.3	Member Function Documentation	199
5.54.3.1	makeActivationFunction	199
5.55	Square Class Reference	199
5.55.1	Detailed Description	201
5.55.2	Constructor & Destructor Documentation	201
5.55.2.1	Square	201
5.55.3	Member Function Documentation	201
5.55.3.1	f0	202
5.55.3.2	f1	202
5.56	SquareFactory Class Reference	202
5.56.1	Detailed Description	205
5.56.2	Constructor & Destructor Documentation	205
5.56.2.1	SquareFactory	205
5.56.3	Member Function Documentation	205

5.56.3.1	<a href="#">makeActivationFunction</a>	205
5.57	<a href="#">Tanh Class Reference</a>	205
5.57.1	<a href="#">Detailed Description</a>	207
5.57.2	<a href="#">Constructor &amp; Destructor Documentation</a>	207
5.57.2.1	<a href="#">Tanh</a>	207
5.57.3	<a href="#">Member Function Documentation</a>	208
5.57.3.1	<a href="#">f0</a>	208
5.57.3.2	<a href="#">f1</a>	208
5.58	<a href="#">TanhFactory Class Reference</a>	209
5.58.1	<a href="#">Detailed Description</a>	212
5.58.2	<a href="#">Constructor &amp; Destructor Documentation</a>	212
5.58.2.1	<a href="#">TanhFactory</a>	212
5.58.3	<a href="#">Member Function Documentation</a>	212
5.58.3.1	<a href="#">makeActivationFunction</a>	212
5.59	<a href="#">Threshold Class Reference</a>	212
5.59.1	<a href="#">Detailed Description</a>	214
5.59.2	<a href="#">Constructor &amp; Destructor Documentation</a>	214
5.59.2.1	<a href="#">Threshold</a>	214
5.59.3	<a href="#">Member Function Documentation</a>	214
5.59.3.1	<a href="#">f0</a>	215
5.59.3.2	<a href="#">f1</a>	215
5.60	<a href="#">ThresholdFactory Class Reference</a>	215
5.60.1	<a href="#">Detailed Description</a>	218
5.60.2	<a href="#">Constructor &amp; Destructor Documentation</a>	218
5.60.2.1	<a href="#">ThresholdFactory</a>	218
5.60.3	<a href="#">Member Function Documentation</a>	218
5.60.3.1	<a href="#">makeActivationFunction</a>	218
<b>6</b>	<b><a href="#">File Documentation</a></b>	<b>219</b>
6.1	<a href="#">/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/ActivationFunction.cpp File Reference</a>	219
6.2	<a href="#">/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/ADAPTgdNetworkTrainBehavior.cpp File Reference</a>	220
6.3	<a href="#">/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/AMORE.h File Reference</a>	220

6.3.1	Define Documentation	222
6.3.1.1	size_type	222
6.3.2	Typedef Documentation	222
6.3.2.1	ActivationFunctionPtr	222
6.3.2.2	ActivationFunctionRef	222
6.3.2.3	ConContainerPtr	222
6.3.2.4	ConIteratorPtr	222
6.3.2.5	ConPtr	222
6.3.2.6	Handler	223
6.3.2.7	LayerContainerPtr	223
6.3.2.8	LayerPtr	223
6.3.2.9	NetworkTrainBehaviorPtr	223
6.3.2.10	NeuralCreatorPtr	223
6.3.2.11	NeuralFactoryPtr	223
6.3.2.12	NeuralNetworkPtr	223
6.3.2.13	NeuralNetworkWeakPtr	223
6.3.2.14	NeuronIteratorPtr	223
6.3.2.15	NeuronPtr	223
6.3.2.16	NeuronRef	224
6.3.2.17	NeuronTrainBehaviorPtr	224
6.3.2.18	NeuronWeakPtr	224
6.3.2.19	PredictBehaviorPtr	224
6.3.2.20	PredictBehaviorRef	224
6.3.2.21	TrainingBehaviorRef	224
6.4	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ActivationFunction.h File Reference	224
6.5	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdNetworkTrainBehavior.h File Reference	225
6.6	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdNeuronTrainBehavior.h File Reference	225
6.7	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwmNetworkTrainBehavior.h File Reference	226

6.8	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwmNeuronTrainBehavior.h File Reference . . . . .	226
6.9	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptNetworkTrainBehavior.h File Reference . . . . .	227
6.10	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptNeuronTrainBehavior.h File Reference . . . . .	228
6.11	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ArcTan.h File Reference . . . . .	228
6.12	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ArcTanFactory.h File Reference . . . . .	229
6.13	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdNetworkTrainBehavior.h File Reference . . . . .	230
6.14	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdNeuronTrainBehavior.h File Reference . . . . .	230
6.15	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwmNetworkTrainBehavior.h File Reference . . . . .	231
6.16	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwmNeuronTrainBehavior.h File Reference . . . . .	231
6.17	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BatchNetworkTrainBehavior.h File Reference . . . . .	232
6.18	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BatchNeuronTrainBehavior.h File Reference . . . . .	233
6.19	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Connection.h File Reference . . . . .	234
6.20	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Container.h File Reference . . . . .	234
6.21	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Cosine.h File Reference . . . . .	235
6.22	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/CosineFactory.h File Reference . . . . .	235
6.23	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Elliot.h File Reference . . . . .	236

6.24	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ElliotFactory.h File Reference . . . .	236
6.25	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Exponential.h File Reference . . . .	237
6.26	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ExponentialFactory.h File Reference . . . .	238
6.27	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Gauss.h File Reference . . . . .	238
6.28	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/GaussFactory.h File Reference . . . .	239
6.29	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Identity.h File Reference . . . . .	239
6.30	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/IdentityFactory.h File Reference . . . .	240
6.31	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Iterator.h File Reference . . . . .	241
6.32	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Logistic.h File Reference . . . . .	241
6.33	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticFactory.h File Reference . . . .	242
6.34	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPbehavior.h File Reference . . . .	243
6.35	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPfactory.h File Reference . . . . .	243
6.36	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkRinterface.h File Reference . . . .	244
6.37	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkTrainBehavior.h File Reference . . . . .	245
6.38	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralCreator.h File Reference . . . .	245
6.39	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralFactory.h File Reference . . . .	246
6.40	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralNetwork.h File Reference . . . .	246
6.41	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Neuron.h File Reference . . . . .	246
6.42	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuronTrainBehavior.h File Reference . . . .	247

6.43	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/PredictBehavior.h File Reference	247
6.44	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasis.h File Reference	248
6.45	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasisFactory.h File Reference	248
6.46	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFbehavior.h File Reference	249
6.47	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFfactory.h File Reference	249
6.48	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Reciprocal.h File Reference	250
6.49	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ReciprocalFactory.h File Reference	250
6.50	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainer.h File Reference	251
6.51	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainerIterator.h File Reference	252
6.52	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainerReverselIterator.h File Reference	253
6.53	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNetwork.h File Reference	254
6.54	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuralCreator.h File Reference	255
6.55	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuron.h File Reference	255
6.56	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Sine.h File Reference	256
6.57	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SineFactory.h File Reference	257
6.58	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Square.h File Reference	257
6.59	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SquareFactory.h File Reference	258
6.60	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Tanh.h File Reference	259
6.61	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/TanhFactory.h File Reference	259

6.62	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Threshold.h File Reference . . . . .	260
6.63	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ThresholdFactory.h File Reference . . . . .	261
6.64	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Connection.cpp File Reference . . . . .	261
6.65	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Identity.cpp File Reference . . . . .	262
6.66	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/IdentityFactory.cpp File Reference . . . . .	263
6.67	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPbehavior.cpp File Reference . . . . .	264
6.68	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPfactory.cpp File Reference . . . . .	264
6.69	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NetworkRinterface.cpp File Reference . . . . .	265
6.70	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NeuralNetwork.cpp File Reference . . . . .	266
6.71	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Neuron.cpp File Reference . . . . .	267
6.72	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/package.h File Reference . . . . .	268
6.72.1	Define Documentation . . . . .	269
6.72.1.1	size_type . . . . .	269
6.72.2	Typedef Documentation . . . . .	269
6.72.2.1	ActivationFunctionPtr . . . . .	269
6.72.2.2	ActivationFunctionRef . . . . .	269
6.72.2.3	ConContainerPtr . . . . .	269
6.72.2.4	ConIteratorPtr . . . . .	269
6.72.2.5	ConPtr . . . . .	269
6.72.2.6	Handler . . . . .	269
6.72.2.7	LayerContainerPtr . . . . .	270
6.72.2.8	LayerPtr . . . . .	270
6.72.2.9	NetworkTrainBehaviorPtr . . . . .	270
6.72.2.10	NeuralCreatorPtr . . . . .	270
6.72.2.11	NeuralFactoryPtr . . . . .	270
6.72.2.12	NeuralNetworkPtr . . . . .	270



6.72.2.13 NeuralNetworkWeakPtr . . . . .	270
6.72.2.14 NeuronIteratorPtr . . . . .	270
6.72.2.15 NeuronPtr . . . . .	270
6.72.2.16 NeuronRef . . . . .	270
6.72.2.17 NeuronTrainBehaviorPtr . . . . .	271
6.72.2.18 NeuronWeakPtr . . . . .	271
6.72.2.19 PredictBehaviorPtr . . . . .	271
6.72.2.20 PredictBehaviorRef . . . . .	271
6.73 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE- WC/pkg/AMORE/src/PredictBehavior.cpp File Reference . . . . .	271
6.74 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE- WC/pkg/AMORE/src/RcppModules.cpp File Reference . . . . .	272
6.74.1 Function Documentation . . . . .	273
6.74.1.1 RCPP_MODULE . . . . .	273
6.75 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE- WC/pkg/AMORE/src/SimpleNetwork.cpp File Reference . . . . .	273
6.76 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE- WC/pkg/AMORE/src/SimpleNeuralCreator.cpp File Reference . . . . .	274
6.77 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE- WC/pkg/AMORE/src/SimpleNeuron.cpp File Reference . . . . .	275
6.78 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE- WC/pkg/AMORE/src/Tanh.cpp File Reference . . . . .	276
6.79 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE- WC/pkg/AMORE/src/TanhFactory.cpp File Reference . . . . .	276



# 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 . . . . .	11
ArcTan . . . . .	30
Cosine . . . . .	60
Elliot . . . . .	66
Exponential . . . . .	72
Gauss . . . . .	78
Identity . . . . .	84
Logistic . . . . .	92
RadialBasis . . . . .	141
Reciprocal . . . . .	153
Sine . . . . .	194
Square . . . . .	199
Tanh . . . . .	205
Threshold . . . . .	212
Con . . . . .	51
Container< T > . . . . .	57
SimpleContainer< T > . . . . .	159
Iterator< T > . . . . .	90
SimpleContainerIterator< T > . . . . .	164
SimpleContainerReverselIterator< T > . . . . .	168
NetworkRinterface . . . . .	110
NetworkTrainBehavior . . . . .	116
AdaptNetworkTrainBehavior . . . . .	25
ADAPTgdNetworkTrainBehavior . . . . .	13
ADAPTgdwmNetworkTrainBehavior . . . . .	19
BatchNetworkTrainBehavior . . . . .	46
BATCHgdNetworkTrainBehavior . . . . .	35
BATCHgdwmNetworkTrainBehavior . . . . .	41

NeuralCreator . . . . .	117
SimpleNeuralCreator . . . . .	180
NeuralFactory . . . . .	118
MLPfactory . . . . .	103
ArcTanFactory . . . . .	32
CosineFactory . . . . .	63
ElliotFactory . . . . .	69
ExponentialFactory . . . . .	75
GaussFactory . . . . .	81
IdentityFactory . . . . .	87
LogisticFactory . . . . .	95
ReciprocalFactory . . . . .	156
SineFactory . . . . .	196
SquareFactory . . . . .	202
TanhFactory . . . . .	209
ThresholdFactory . . . . .	215
RBFfactory . . . . .	149
RadialBasisFactory . . . . .	143
NeuralNetwork . . . . .	122
SimpleNetwork . . . . .	172
Neuron . . . . .	127
SimpleNeuron . . . . .	183
NeuronTrainBehavior . . . . .	133
AdaptNeuronTrainBehavior . . . . .	28
ADAPTgdNeuronTrainBehavior . . . . .	16
ADAPTgdwmNeuronTrainBehavior . . . . .	22
BatchNeuronTrainBehavior . . . . .	49
BATCHgdNeuronTrainBehavior . . . . .	38
BATCHgdwmNeuronTrainBehavior . . . . .	43
PredictBehavior . . . . .	135
MLPbehavior . . . . .	98
RBFbehavior . . . . .	146

## 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> - ) . . . . .	11
<a href="#">ADAPTgdNetworkTrainBehavior</a> (Class <a href="#">ADAPTgdNetworkTrainBehavior</a> - ) . . .	13
<a href="#">ADAPTgdNeuronTrainBehavior</a> (Class <a href="#">ADAPTgdNeuronTrainBehavior</a> - ) . . .	16
<a href="#">ADAPTgdwmNetworkTrainBehavior</a> (Class <a href="#">ADAPTgdwmNetworkTrainBehavior</a> - ) . . . . .	19
<a href="#">ADAPTgdwmNeuronTrainBehavior</a> (Class <a href="#">ADAPTgdwmNeuronTrainBehavior</a> - ) . . . . .	22
<a href="#">AdaptNetworkTrainBehavior</a> (Class <a href="#">AdaptNetworkTrainBehavior</a> - ) . . . . .	25
<a href="#">AdaptNeuronTrainBehavior</a> (Class <a href="#">AdaptNeuronTrainBehavior</a> - ) . . . . .	28
<a href="#">ArcTan</a> (Class <a href="#">ArcTan</a> - ) . . . . .	30
<a href="#">ArcTanFactory</a> (Class <a href="#">ArcTanFactory</a> - ) . . . . .	32
<a href="#">BATCHgdNetworkTrainBehavior</a> (Class <a href="#">BATCHgdNetworkTrainBehavior</a> - ) . . .	35
<a href="#">BATCHgdNeuronTrainBehavior</a> (Class <a href="#">BATCHgdNeuronTrainBehavior</a> - ) . . .	38
<a href="#">BATCHgdwmNetworkTrainBehavior</a> (Class <a href="#">BATCHgdwmNetworkTrainBehavior</a> - ) . . . . .	41
<a href="#">BATCHgdwmNeuronTrainBehavior</a> (Class <a href="#">BATCHgdwmNeuronTrainBehavior</a> - ) . . . . .	43
<a href="#">BatchNetworkTrainBehavior</a> (Class <a href="#">BatchNetworkTrainBehavior</a> - ) . . . . .	46
<a href="#">BatchNeuronTrainBehavior</a> (Class <a href="#">BatchNeuronTrainBehavior</a> - ) . . . . .	49
<a href="#">Con</a> (Class <a href="#">Con</a> - ) . . . . .	51
<a href="#">Container&lt; T &gt;</a> (Class <a href="#">Container</a> - ) . . . . .	57
<a href="#">Cosine</a> (Class <a href="#">Cosine</a> - ) . . . . .	60
<a href="#">CosineFactory</a> (Class <a href="#">CosineFactory</a> - ) . . . . .	63
<a href="#">Elliot</a> (Class <a href="#">Elliot</a> - ) . . . . .	66
<a href="#">ElliotFactory</a> (Class <a href="#">ElliotFactory</a> - ) . . . . .	69
<a href="#">Exponential</a> (Class <a href="#">Exponential</a> - ) . . . . .	72
<a href="#">ExponentialFactory</a> (Class <a href="#">ExponentialFactory</a> - ) . . . . .	75
<a href="#">Gauss</a> (Class <a href="#">Gauss</a> - ) . . . . .	78
<a href="#">GaussFactory</a> (Class <a href="#">GaussFactory</a> - ) . . . . .	81

Identity (Class <a href="#">Identity</a> - ) . . . . .	84
IdentityFactory (Class <a href="#">IdentityFactory</a> - ) . . . . .	87
Iterator< T > (Class <a href="#">Iterator</a> - ) . . . . .	90
Logistic (Class <a href="#">Logistic</a> - ) . . . . .	92
LogisticFactory (Class <a href="#">LogisticFactory</a> - ) . . . . .	95
MLPbehavior (Class <a href="#">MLPbehavior</a> - ) . . . . .	98
MLPfactory (Class <a href="#">MLPfactory</a> - ) . . . . .	103
NetworkRinterface (Class <a href="#">NetworkRinterface</a> - ) . . . . .	110
NetworkTrainBehavior (Class <a href="#">NetworkTrainBehavior</a> - ) . . . . .	116
NeuralCreator (Class <a href="#">NeuralCreator</a> - ) . . . . .	117
NeuralFactory (Class <a href="#">NeuralFactory</a> - ) . . . . .	118
NeuralNetwork (Class <a href="#">NeuralNetwork</a> - ) . . . . .	122
Neuron (Class <a href="#">Neuron</a> - ) . . . . .	127
NeuronTrainBehavior (Class <a href="#">NeuronTrainBehavior</a> - ) . . . . .	133
PredictBehavior (Class <a href="#">PredictBehavior</a> - ) . . . . .	135
RadialBasis (Class <a href="#">RadialBasis</a> - ) . . . . .	141
RadialBasisFactory (Class <a href="#">RadialBasisFactory</a> - ) . . . . .	143
RBFbehavior (Class <a href="#">RBFbehavior</a> - ) . . . . .	146
RBFfactory (Class <a href="#">RBFfactory</a> - ) . . . . .	149
Reciprocal (Class <a href="#">Reciprocal</a> - ) . . . . .	153
ReciprocalFactory (Class <a href="#">ReciprocalFactory</a> - ) . . . . .	156
SimpleContainer< T > (Class <a href="#">SimpleContainer</a> - ) . . . . .	159
SimpleContainerIterator< T > (Class <a href="#">SimpleContainerIterator</a> - ) . . . . .	164
SimpleContainerReverselIterator< T > (Class <a href="#">SimpleContainerReverselIterator</a> - ) . . . . .	168
SimpleNetwork (Class <a href="#">SimpleNetwork</a> - ) . . . . .	172
SimpleNeuralCreator (Class <a href="#">SimpleNeuralCreator</a> - ) . . . . .	180
SimpleNeuron (Class <a href="#">SimpleNeuron</a> - ) . . . . .	183
Sine (Class <a href="#">Sine</a> - ) . . . . .	194
SineFactory (Class <a href="#">SineFactory</a> - ) . . . . .	196
Square (Class <a href="#">Square</a> - ) . . . . .	199
SquareFactory (Class <a href="#">SquareFactory</a> - ) . . . . .	202
Tanh (Class <a href="#">Tanh</a> - ) . . . . .	205
TanhFactory (Class <a href="#">TanhFactory</a> - ) . . . . .	209
Threshold (Class <a href="#">Threshold</a> - ) . . . . .	212
ThresholdFactory (Class <a href="#">ThresholdFactory</a> - ) . . . . .	215



## 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](#)  
219

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[ADAPTgdNetworkTrainBeh](#)  
220

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[AMORE.h](#)  
220

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[Connection.cpp](#)  
261

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[Identity.cpp](#)  
262

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[IdentityFactory.cpp](#)  
263

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[MLPbehavior.cpp](#)  
264

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[MLPfactory.cpp](#)  
264

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[NetworkRinterface.cpp](#)  
265

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[NeuralNetwork.cpp](#)  
266

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[Neuron.cpp](#)  
267

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[package.h](#)  
268

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[PredictBehavior.cpp](#)  
271

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[RcppModules.cpp](#)  
272

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNetwork  
273

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeural  
274

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuron  
275

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Tanh.cpp  
276

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/TanhFactory.c  
276

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
224

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
225

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
225

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
226

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
226

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
227

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
228

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
228

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
229

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
230

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
230

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
231

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
231

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
232

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
233

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
234

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
234

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
235

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
235

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
236

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ElliotFactory.h	236
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Exponential.h	237
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ExponentialFactory.h	238
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Gauss.h	238
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/GaussFactory.h	239
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Identity.h	239
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/IdentityFactory.h	240
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Iterator.h	241
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Logistic.h	241
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticFactory.h	242
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPbehavior.h	243
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPfactory.h	243
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkRint.h	244
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkTrain.h	245
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralCreat.h	245
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralFactory.h	246
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralNetwork.h	246
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Neuron.h	246
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuronTrain.h	247
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/PredictBehavior.h	247
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasis.h	248
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasisFactory.h	248
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFbehavior.h	249
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFfactory.h	249
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Reciprocal.h	250

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
250

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
251

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
252

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
253

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
254

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
255

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
255

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
256

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
257

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
257

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
258

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
259

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
259

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
260

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.  
261

## Chapter 5

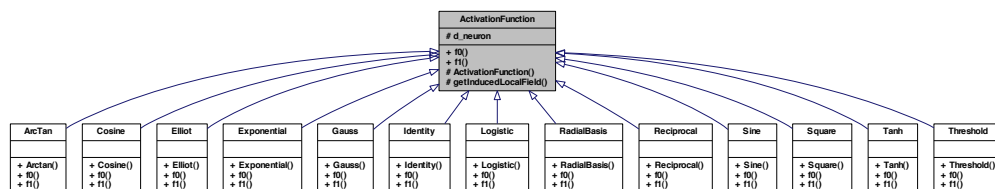
# Class Documentation

### 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`.

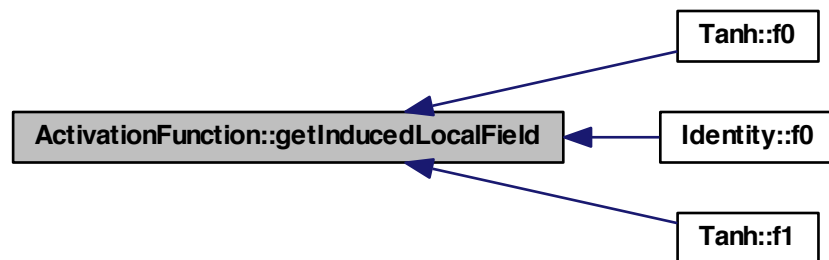
Referenced by `Tanh::f0()`, `Identity::f0()`, and `Tanh::f1()`.

```

{
    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:

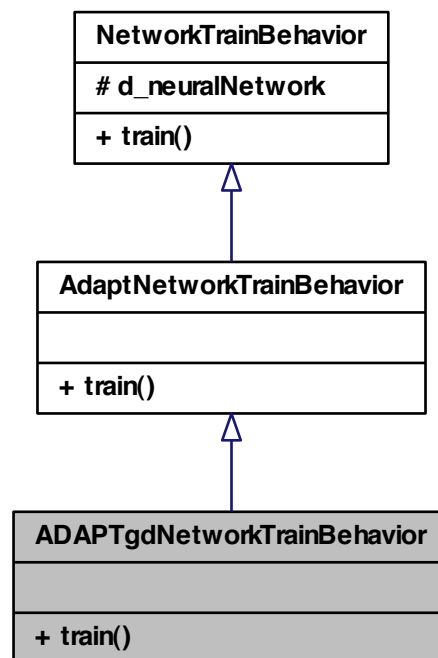
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ActivationFunction.h`
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/ActivationFunction.cpp`

## 5.2 ADAPTgdNetworkTrainBehavior Class Reference

class [ADAPTgdNetworkTrainBehavior](#) -

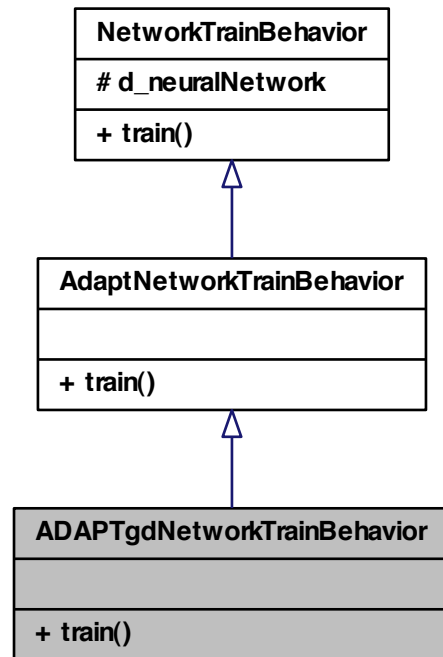
```
#include <ADAPTgdNetworkTrainBehavior.h>
```

Inheritance diagram for ADAPTgdNetworkTrainBehavior:





Collaboration diagram for ADAPTgdNetworkTrainBehavior:



### Public Member Functions

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

### 5.2.1 Detailed Description

class [ADAPTgdNetworkTrainBehavior](#) -

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

### 5.2.2 Member Function Documentation

5.2.2.1 `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:

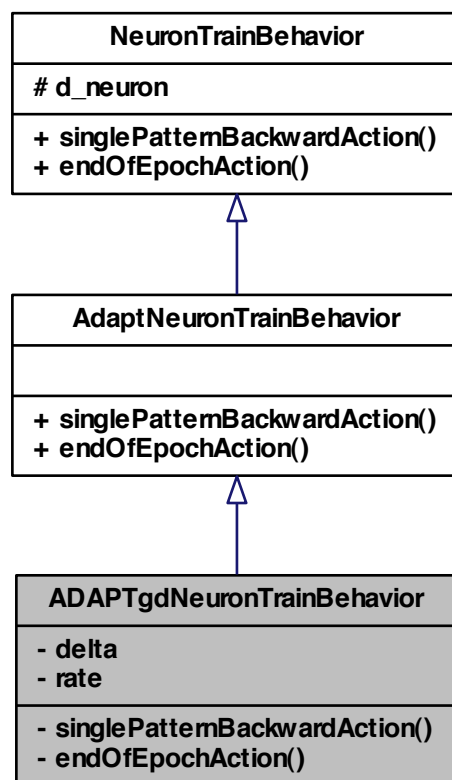
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[ADAPTgdNeuronTrainBehavior](#)

## 5.3 ADAPTgdNeuronTrainBehavior Class Reference

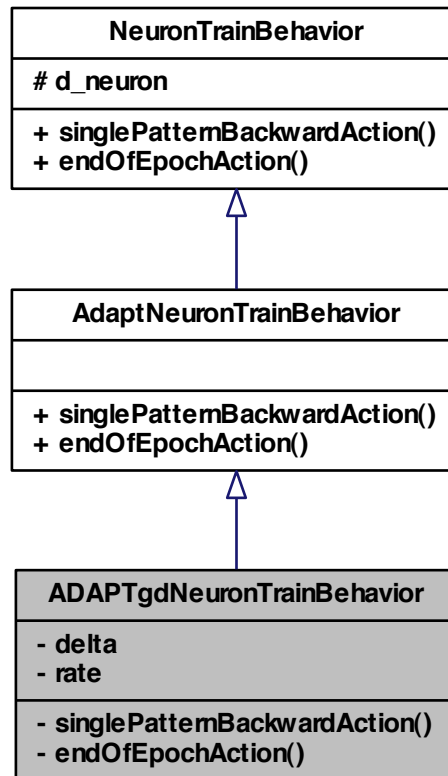
class [ADAPTgdNeuronTrainBehavior](#) -

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

Inheritance diagram for ADAPTgdNeuronTrainBehavior:



Collaboration diagram for ADAPTgdNeuronTrainBehavior:



### Private Member Functions

- void `singlePatternBackwardAction` ()
- void `endOfEpochAction` ()

### Private Attributes

- double `delta`
- double learning `rate`

### 5.3.1 Detailed Description

class [ADAPTgdNeuronTrainBehavior](#) -

Definition at line 5 of file ADAPTgdNeuronTrainBehavior.h.

### 5.3.2 Member Function Documentation

5.3.2.1 void ADAPTgdNeuronTrainBehavior::endOfEpochAction ( ) [private, virtual]

Implements [AdaptNeuronTrainBehavior](#).

5.3.2.2 void ADAPTgdNeuronTrainBehavior::singlePatternBackwardAction ( ) [private, virtual]

Implements [AdaptNeuronTrainBehavior](#).

### 5.3.3 Member Data Documentation

5.3.3.1 double ADAPTgdNeuronTrainBehavior::delta [private]

Definition at line 8 of file ADAPTgdNeuronTrainBehavior.h.

5.3.3.2 double learning ADAPTgdNeuronTrainBehavior::rate [private]

Definition at line 9 of file ADAPTgdNeuronTrainBehavior.h.

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

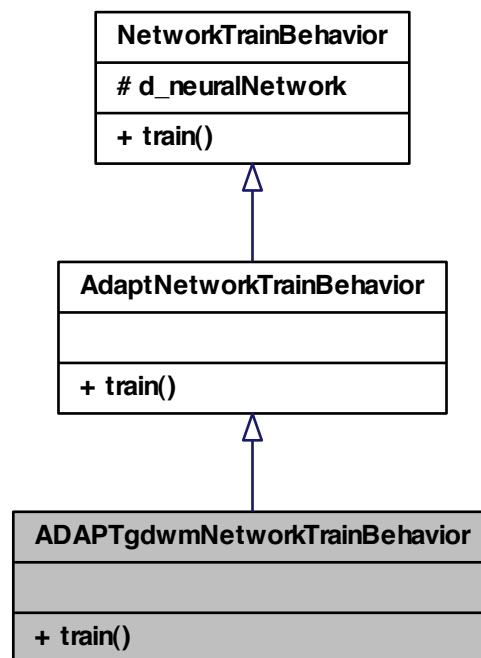
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[ADAPTgdNeuronTrainBehavior.h](#)

## 5.4 ADAPTgdwmNetworkTrainBehavior Class Reference

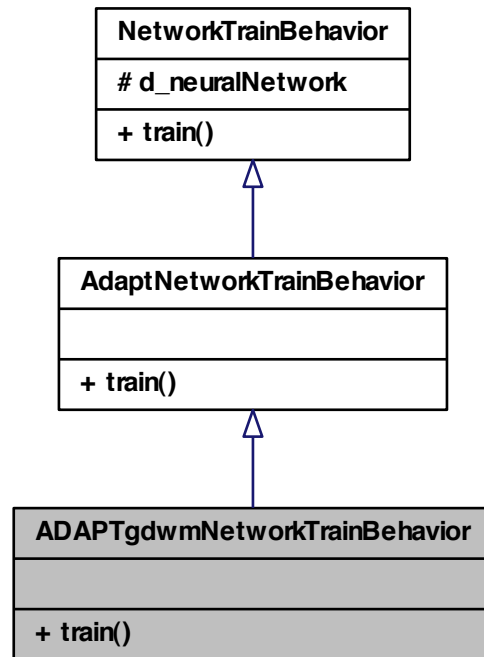
class [ADAPTgdwmNetworkTrainBehavior](#) -

#include <ADAPTgdwmNetworkTrainBehavior.h>

Inheritance diagram for ADAPTgdwmNetworkTrainBehavior:



Collaboration diagram for ADAPTgdwmNetworkTrainBehavior:



### Public Member Functions

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

#### 5.4.1 Detailed Description

class [ADAPTgdwmNetworkTrainBehavior](#) -

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

#### 5.4.2 Member Function Documentation

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

Implements [AdaptNetworkTrainBehavior](#).

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

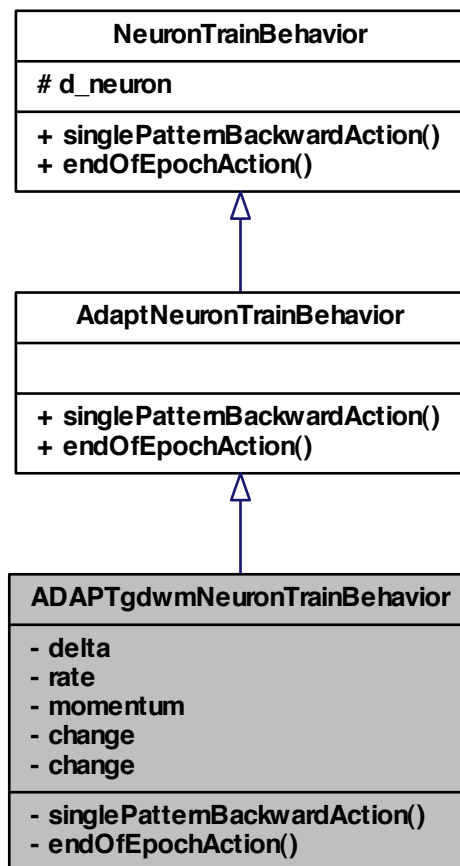
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.5 ADAPTgdwmNeuronTrainBehavior Class Reference

class [ADAPTgdwmNeuronTrainBehavior](#) -

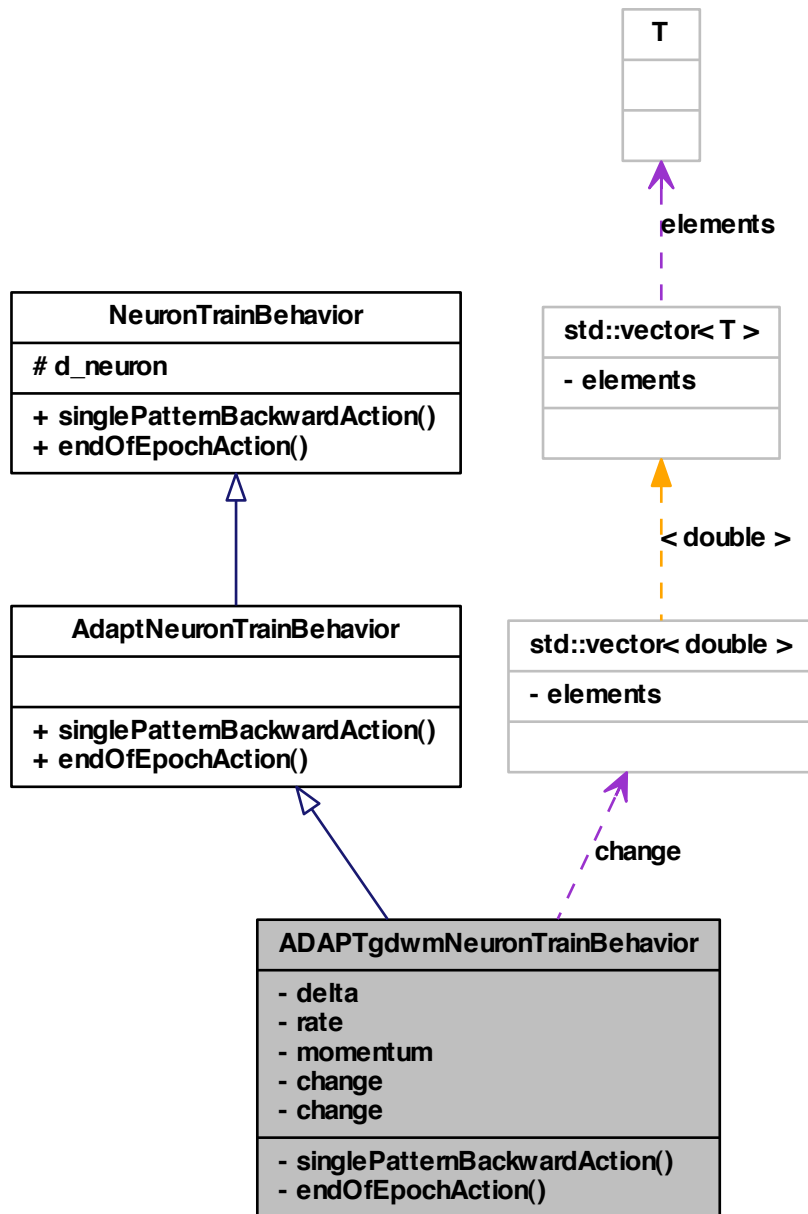
```
#include <ADAPTgdwmNeuronTrainBehavior.h>
```

Inheritance diagram for ADAPTgdwmNeuronTrainBehavior:





Collaboration diagram for ADAPTgdwmNeuronTrainBehavior:



### Private Member Functions

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

### Private Attributes

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

#### 5.5.1 Detailed Description

class [ADAPTgdwmNeuronTrainBehavior](#) -

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

#### 5.5.2 Member Function Documentation

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

Implements [AdaptNeuronTrainBehavior](#).

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

Implements [AdaptNeuronTrainBehavior](#).

#### 5.5.3 Member Data Documentation

**5.5.3.1** std::vector<double> former weight [ADAPTgdwmNeuronTrainBehavior::change](#) [[private](#)]

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

**5.5.3.2** double former bias [ADAPTgdwmNeuronTrainBehavior::change](#)  
[[private](#)]

Definition at line 12 of file [ADAPTgdwmNeuronTrainBehavior.h](#).

5.5.3.3 double **ADAPTgdwmNeuronTrainBehavior::delta** [private]

Definition at line 8 of file ADAPTgdwmNeuronTrainBehavior.h.

5.5.3.4 double **ADAPTgdwmNeuronTrainBehavior::momentum** [private]

Definition at line 10 of file ADAPTgdwmNeuronTrainBehavior.h.

5.5.3.5 double learning **ADAPTgdwmNeuronTrainBehavior::rate** [private]

Definition at line 9 of file ADAPTgdwmNeuronTrainBehavior.h.

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

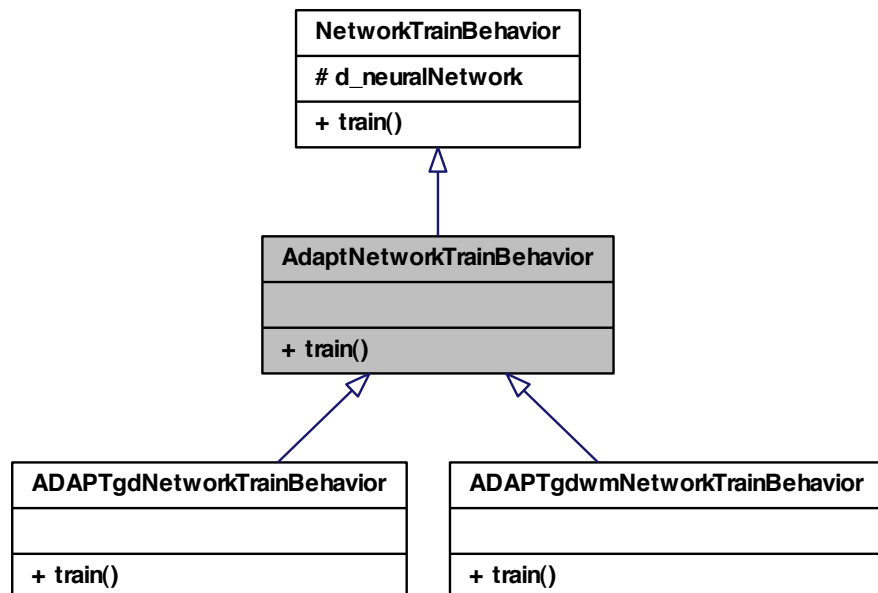
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[ADAPTgdwmNeuronTrainBehavior.h](#)

## 5.6 AdaptNetworkTrainBehavior Class Reference

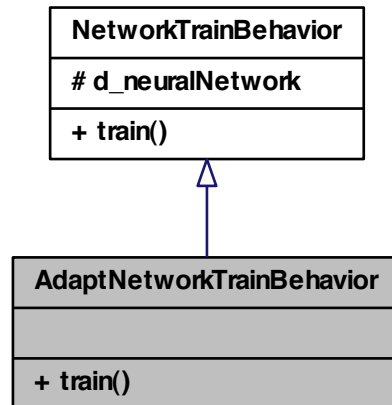
class [AdaptNetworkTrainBehavior](#) -

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

Inheritance diagram for AdaptNetworkTrainBehavior:



Collaboration diagram for AdaptNetworkTrainBehavior:



### Public Member Functions

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

#### 5.6.1 Detailed Description

class [AdaptNetworkTrainBehavior](#) -

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

#### 5.6.2 Member Function Documentation

5.6.2.1 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:

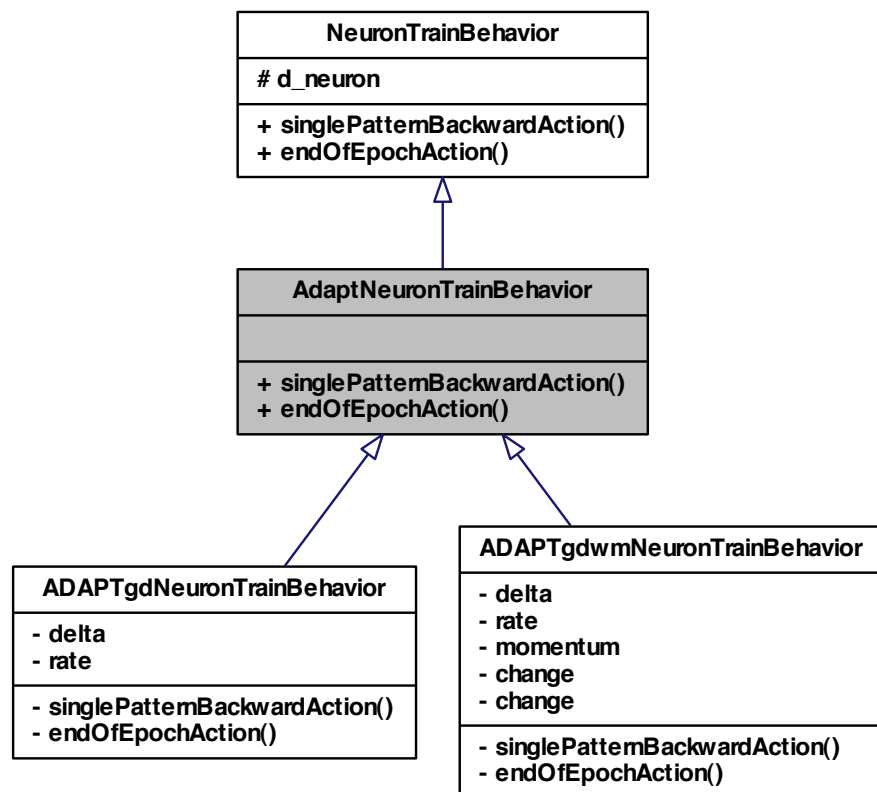
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptNet`

## 5.7 AdaptNeuronTrainBehavior Class Reference

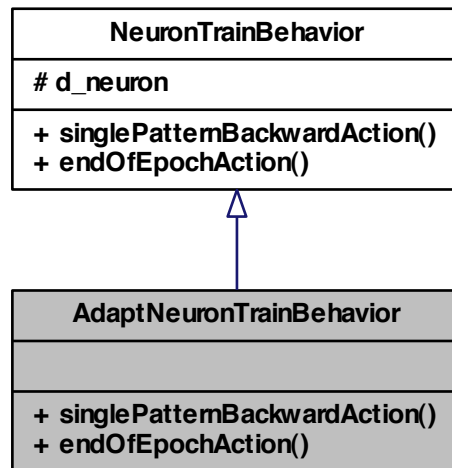
class [AdaptNeuronTrainBehavior](#) -

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

Inheritance diagram for AdaptNeuronTrainBehavior:



Collaboration diagram for AdaptNeuronTrainBehavior:



### Public Member Functions

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

#### 5.7.1 Detailed Description

class [AdaptNeuronTrainBehavior](#) -

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

#### 5.7.2 Member Function Documentation

**5.7.2.1** virtual void [AdaptNeuronTrainBehavior::endOfEpochAction](#) ( ) [pure virtual]

Implements [NeuronTrainBehavior](#).

Implemented in [ADAPTgdNeuronTrainBehavior](#), and [ADAPTgdwmNeuronTrainBehavior](#).

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

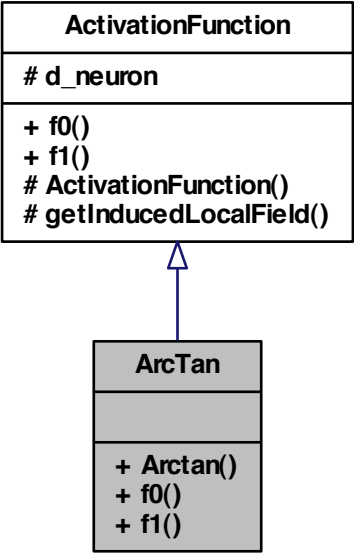
Implements [NeuronTrainBehavior](#).  
Implemented in [ADAPTgdNeuronTrainBehavior](#), and [ADAPTgdwmNeuronTrainBehavior](#).

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`

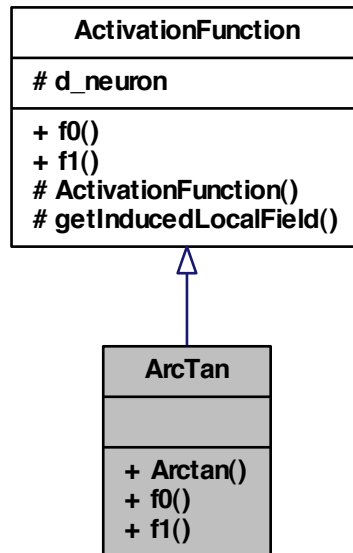
5.8 ArcTan Class Reference

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





Collaboration diagram for ArcTan:



### Public Member Functions

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

### 5.8.1 Detailed Description

class [ArcTan](#) -

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

### 5.8.2 Member Function Documentation

5.8.2.1 `ArcTan::ArcTan ( NeuronPtr neuronPtr )`

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

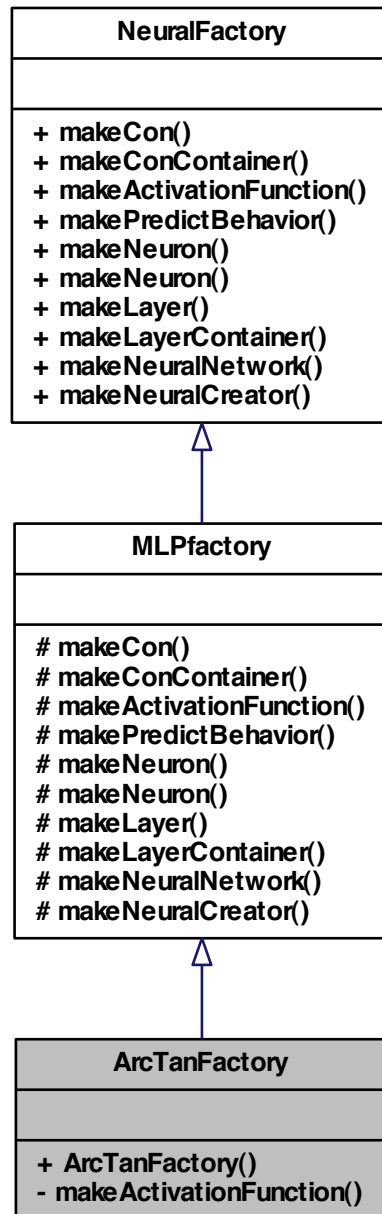
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead`

## 5.9 ArcTanFactory Class Reference

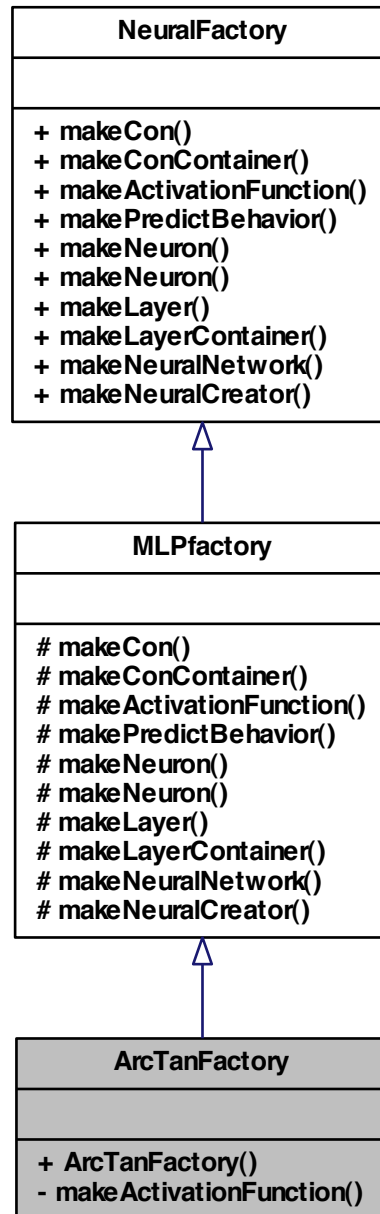
class [ArcTanFactory](#) -

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

Inheritance diagram for ArcTanFactory:



Collaboration diagram for ArcTanFactory:



## Public Member Functions

- [ArcTanFactory](#) ()

## Private Member Functions

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

### 5.9.1 Detailed Description

class [ArcTanFactory](#) -

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

### 5.9.2 Constructor & Destructor Documentation

5.9.2.1 [ArcTanFactory::ArcTanFactory](#) ( )

### 5.9.3 Member Function Documentation

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

Implements [MLPfactory](#).

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

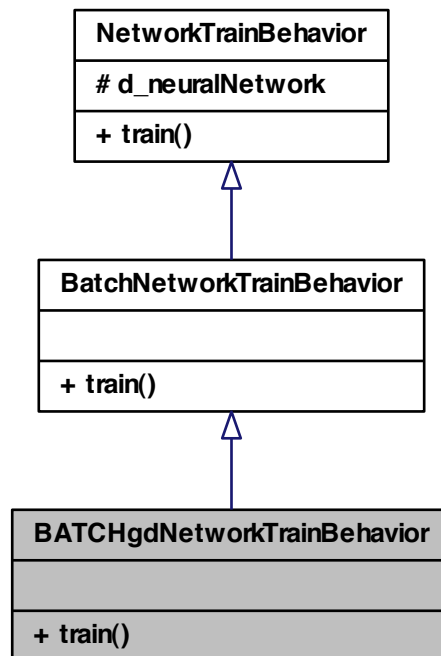
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/\[ArcTanFa\]\(#\)](#)

## 5.10 BATCHgdNetworkTrainBehavior Class Reference

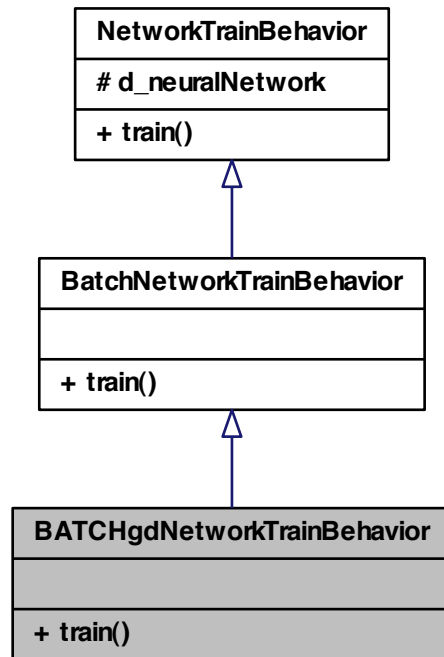
class [BATCHgdNetworkTrainBehavior](#) -

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

Inheritance diagram for BATCHgdNetworkTrainBehavior:



Collaboration diagram for BATCHgdNetworkTrainBehavior:



### Public Member Functions

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

### 5.10.1 Detailed Description

class [BATCHgdNetworkTrainBehavior](#) -

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

### 5.10.2 Member Function Documentation

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

Implements [BatchNetworkTrainBehavior](#).

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

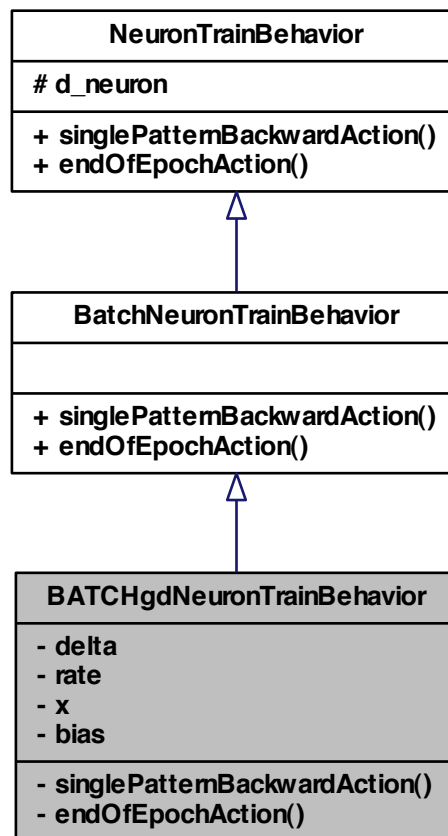
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.11 BATCHgdNeuronTrainBehavior Class Reference

class [BATCHgdNeuronTrainBehavior](#) -

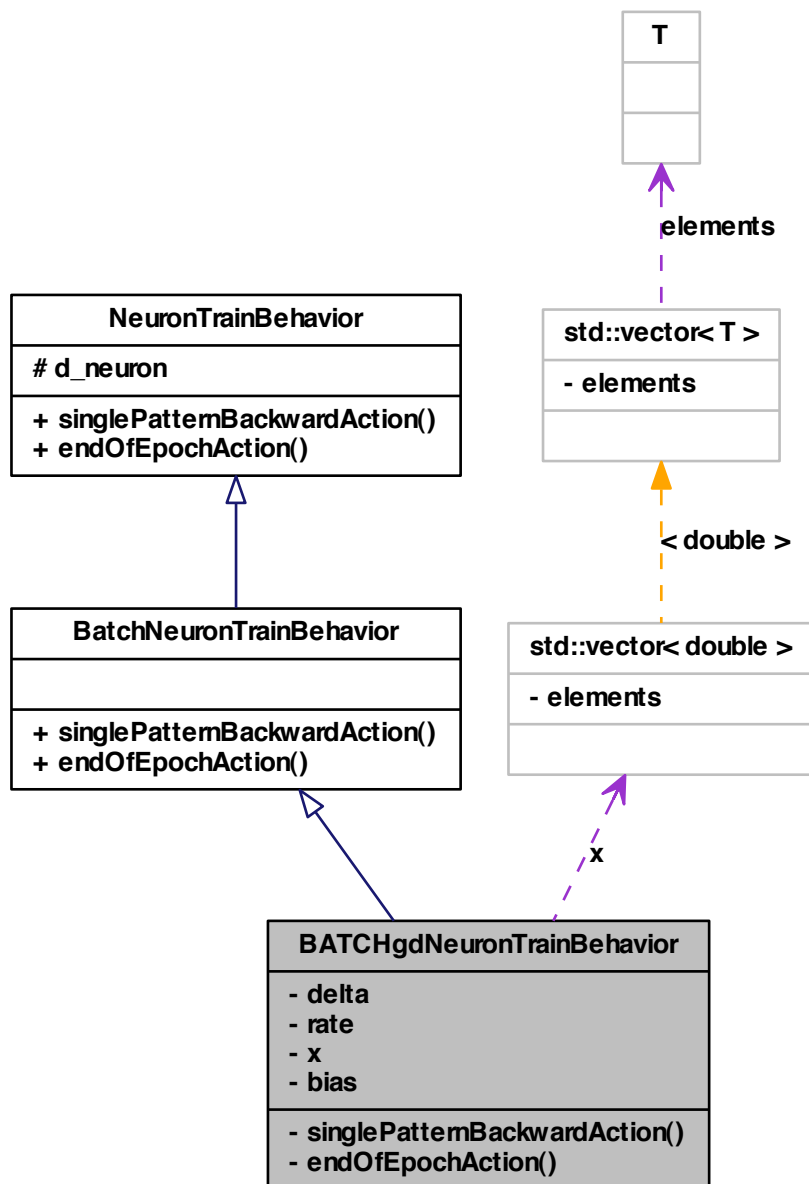
```
#include <BATCHgdNeuronTrainBehavior.h>
```

Inheritance diagram for BATCHgdNeuronTrainBehavior:





Collaboration diagram for BATCHgdNeuronTrainBehavior:



### Private Member Functions

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

### Private Attributes

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

#### 5.11.1 Detailed Description

class [BATCHgdNeuronTrainBehavior](#) -

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

#### 5.11.2 Member Function Documentation

**5.11.2.1** void [BATCHgdNeuronTrainBehavior::endOfEpochAction](#) ( ) [private, virtual]

Implements [BatchNeuronTrainBehavior](#).

**5.11.2.2** void [BATCHgdNeuronTrainBehavior::singlePatternBackwardAction](#) ( ) [private, virtual]

Implements [BatchNeuronTrainBehavior](#).

#### 5.11.3 Member Data Documentation

**5.11.3.1** double sum [delta](#) [BATCHgdNeuronTrainBehavior::bias](#) [private]

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

**5.11.3.2** double [BATCHgdNeuronTrainBehavior::delta](#) [private]

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

**5.11.3.3** double learning [BATCHgdNeuronTrainBehavior::rate](#) [private]

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

5.11.3.4 `std::vector<double> sum delta BATCHgdNeuronTrainBehavior::x`  
[private]

Definition at line 10 of file BATCHgdNeuronTrainBehavior.h.

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

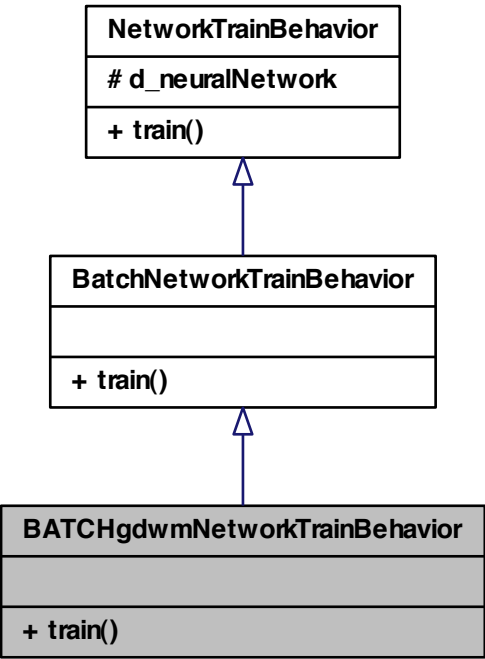
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgd

## 5.12 BATCHgdwmNetworkTrainBehavior Class Reference

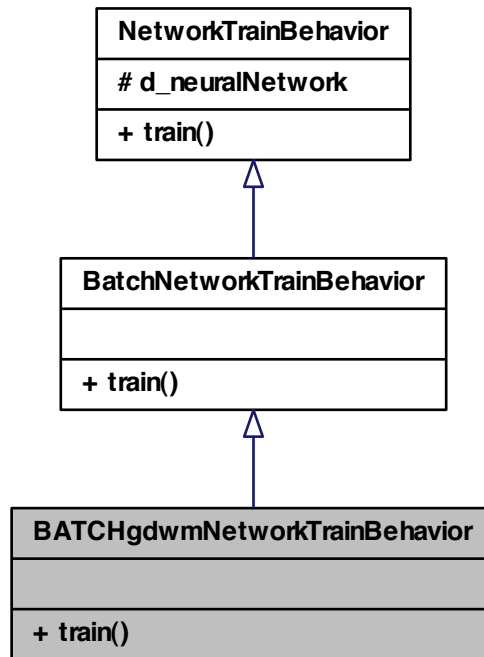
class [BATCHgdwmNetworkTrainBehavior](#) -

`#include <BATCHgdwmNetworkTrainBehavior.h>`

Inheritance diagram for BATCHgdwmNetworkTrainBehavior:



Collaboration diagram for BATCHgdwmNetworkTrainBehavior:



## Public Member Functions

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

### 5.12.1 Detailed Description

class [BATCHgdwmNetworkTrainBehavior](#) -

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

### 5.12.2 Member Function Documentation

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

Implements [BatchNetworkTrainBehavior](#).

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

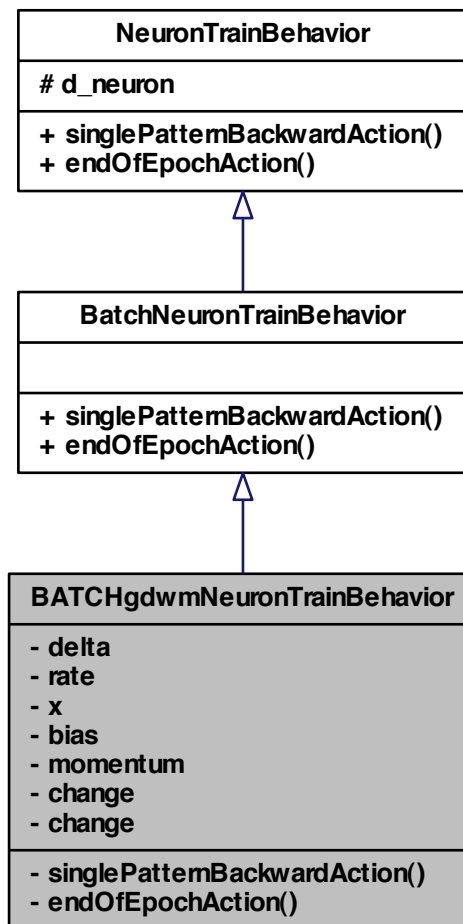
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwmNeuronTrainBehavior.h

## 5.13 BATCHgdwmNeuronTrainBehavior Class Reference

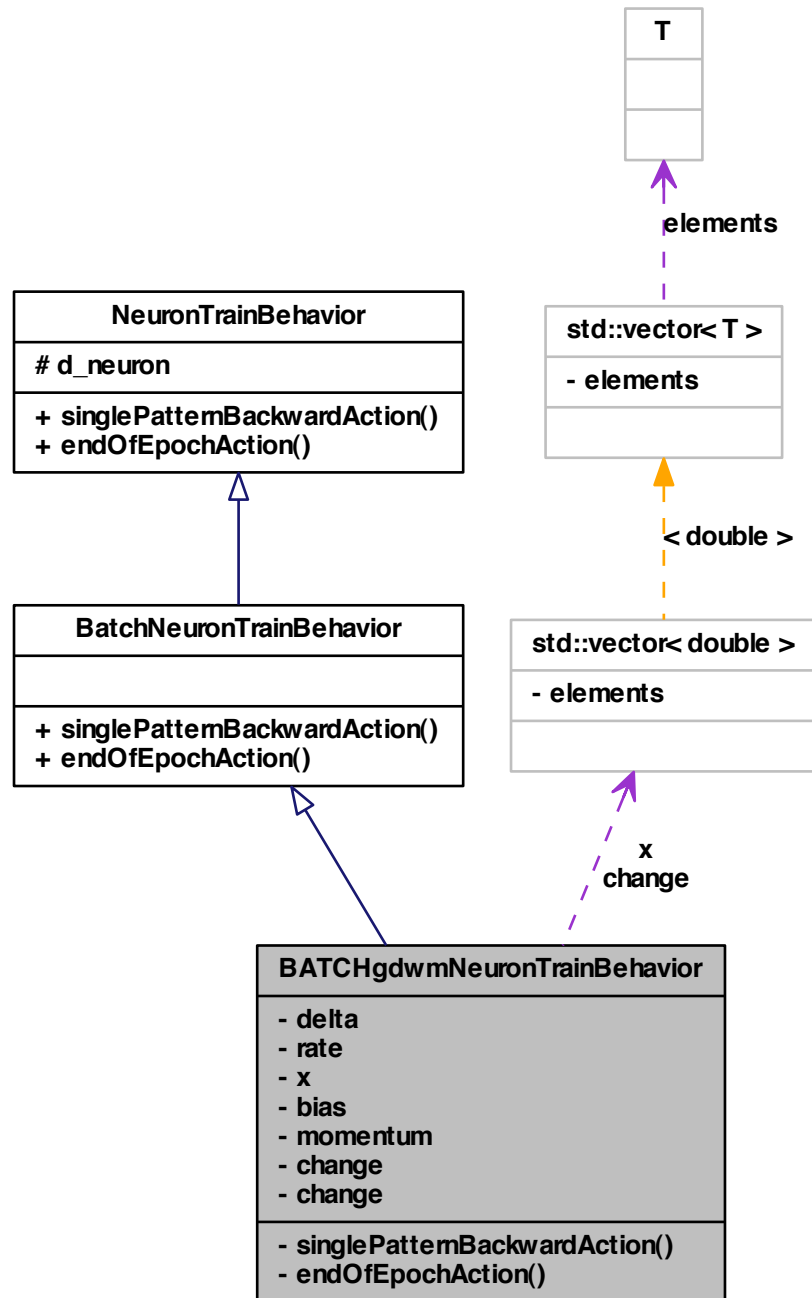
class [BATCHgdwmNeuronTrainBehavior](#) -

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

Inheritance diagram for BATCHgdwmNeuronTrainBehavior:



Collaboration diagram for BATCHgdwmNeuronTrainBehavior:



### Private Member Functions

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

### Private 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](#)

#### 5.13.1 Detailed Description

class [BATCHgdwmNeuronTrainBehavior](#) -

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

#### 5.13.2 Member Function Documentation

**5.13.2.1** void [BATCHgdwmNeuronTrainBehavior::endOfEpochAction](#) ( ) [private, virtual]

Implements [BatchNeuronTrainBehavior](#).

**5.13.2.2** void [BATCHgdwmNeuronTrainBehavior::singlePatternBackwardAction](#) ( ) [private, virtual]

Implements [BatchNeuronTrainBehavior](#).

#### 5.13.3 Member Data Documentation

**5.13.3.1** double sum [delta BATCHgdwmNeuronTrainBehavior::bias](#) [private]

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

**5.13.3.2** double former [bias BATCHgdwmNeuronTrainBehavior::change](#) [private]

Definition at line 14 of file [BATCHgdwmNeuronTrainBehavior.h](#).

5.13.3.3 `std::vector<double> former weight BATCHgdwmNeuronTrainBehavior::change` [private]

Definition at line 13 of file BATCHgdwmNeuronTrainBehavior.h.

5.13.3.4 `double BATCHgdwmNeuronTrainBehavior::delta` [private]

Definition at line 8 of file BATCHgdwmNeuronTrainBehavior.h.

5.13.3.5 `double BATCHgdwmNeuronTrainBehavior::momentum` [private]

Definition at line 12 of file BATCHgdwmNeuronTrainBehavior.h.

5.13.3.6 `double learning BATCHgdwmNeuronTrainBehavior::rate` [private]

Definition at line 9 of file BATCHgdwmNeuronTrainBehavior.h.

5.13.3.7 `std::vector<double> sum delta BATCHgdwmNeuronTrainBehavior::x` [private]

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/classHead

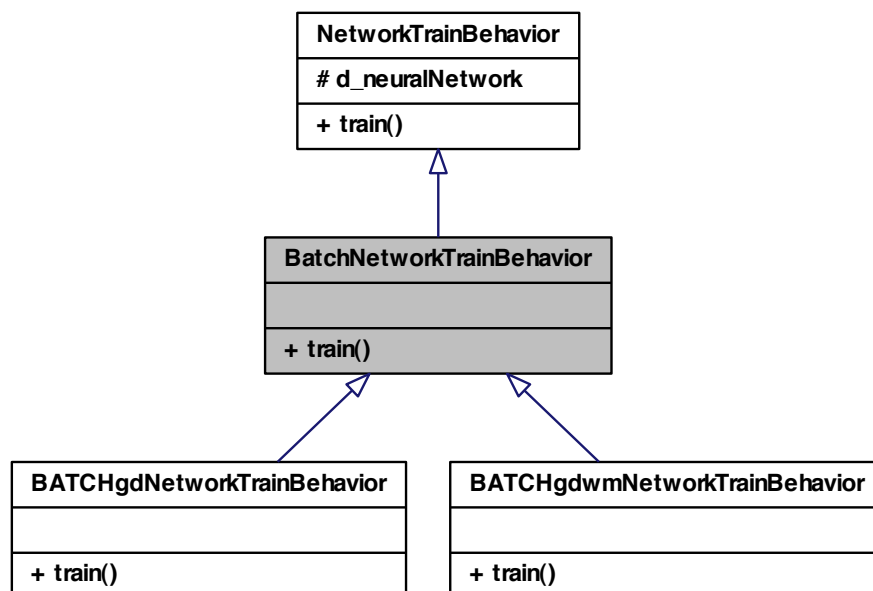
## 5.14 BatchNetworkTrainBehavior Class Reference

class [BatchNetworkTrainBehavior](#) -

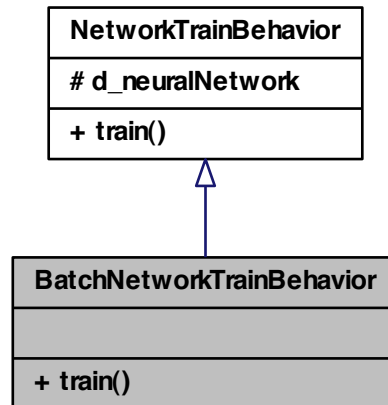
```
#include <BatchNetworkTrainBehavior.h>
```



Inheritance diagram for BatchNetworkTrainBehavior:



Collaboration diagram for BatchNetworkTrainBehavior:



## Public Member Functions

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

### 5.14.1 Detailed Description

class [BatchNetworkTrainBehavior](#) -

Definition at line 5 of file BatchNetworkTrainBehavior.h.

### 5.14.2 Member Function Documentation

5.14.2.1 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:

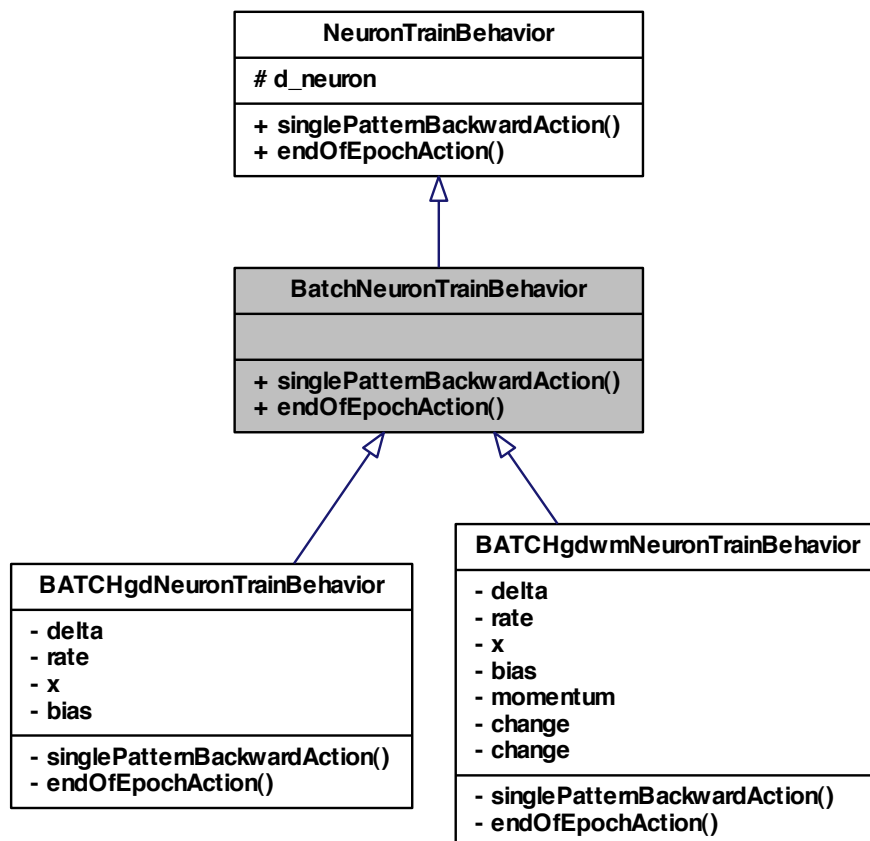
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.15 BatchNeuronTrainBehavior Class Reference

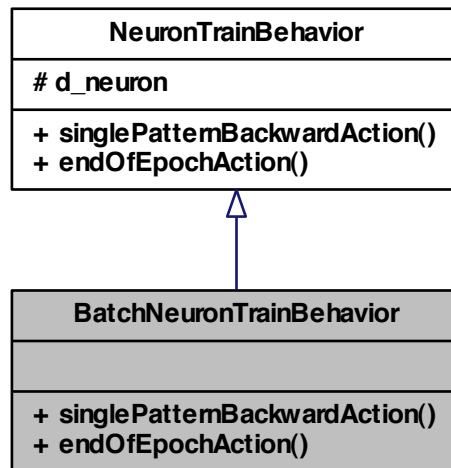
class [BatchNeuronTrainBehavior](#) -

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

Inheritance diagram for BatchNeuronTrainBehavior:



Collaboration diagram for BatchNeuronTrainBehavior:



## Public Member Functions

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

### 5.15.1 Detailed Description

class [BatchNeuronTrainBehavior](#) -

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

### 5.15.2 Member Function Documentation

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

Implements [NeuronTrainBehavior](#).

Implemented in [BATCHgdNeuronTrainBehavior](#), and [BATCHgdwmNeuronTrainBehavior](#).

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

Implements [NeuronTrainBehavior](#).

Implemented in [BATCHgdNeuronTrainBehavior](#), and [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/classHeaders/BatchNeu`

## 5.16 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

### 5.16.1 Detailed Description

class [Con](#) -

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

## 5.16.2 Constructor & Destructor Documentation

### 5.16.2.1 Con::Con ( Neuron & *neuron* )

Constructor.

Definition at line 20 of file Connection.cpp.

```

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

```

### 5.16.2.2 Con::Con ( Neuron & *neuron*, double *weight* )

Constructor.

Definition at line 31 of file Connection.cpp.

```

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

```

## 5.16.3 Member Function Documentation

### 5.16.3.1 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;  
}
```

### 5.16.3.2 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.

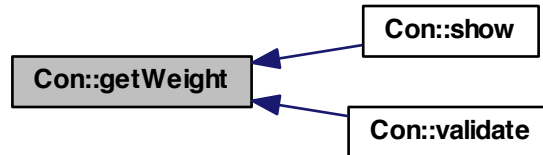
Definition at line 117 of file Connection.cpp.

References `d_weight`.

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

```
{  
    return d_weight;  
}
```

Here is the caller graph for this function:



#### 5.16.3.3 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`.

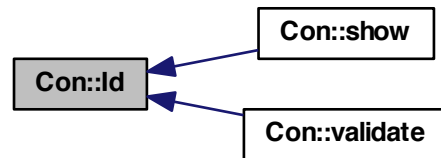
Referenced by `show()`, and `validate()`.

```

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



Here is the caller graph for this function:



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

Definition at line 64 of file Connection.cpp.

References `d_neuron`.

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

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

Definition at line 124 of file Connection.cpp.

References `d_weight`.

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

#### 5.16.3.6 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

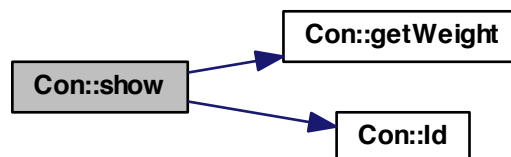
[setWeight](#) and the unit test files, e.g., `runit.Cpp.Con.R`, for usage examples.

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:



#### 5.16.3.7 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()`.

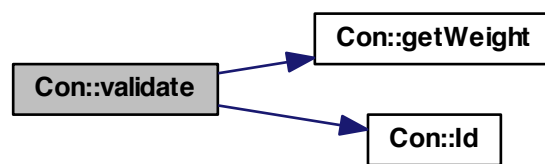
```
{
```

```

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:



#### 5.16.4 Member Data Documentation

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

Definition at line 6 of file Connection.h.

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

##### 5.16.4.2 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:

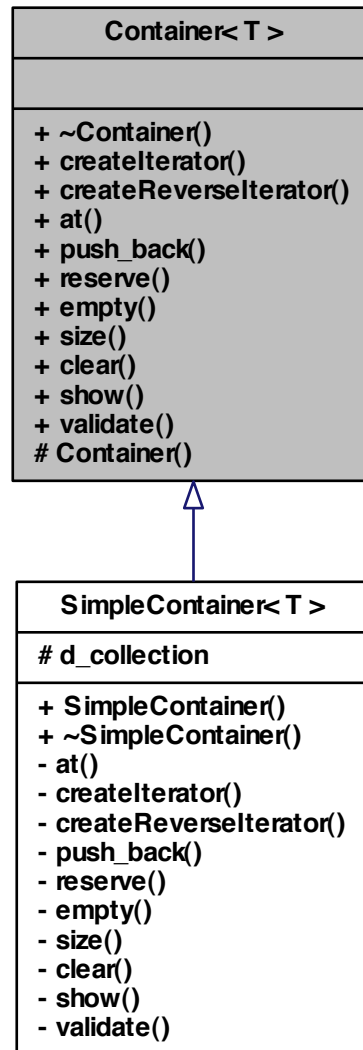
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Connection.h
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Connection.cpp

## 5.17 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 > > [createReverselIterator](#) ()=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](#) ()

#### 5.17.1 Detailed Description

template<typename T>class Container< T >

class [Container](#) -

Definition at line 5 of file Container.h.

#### 5.17.2 Constructor & Destructor Documentation

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

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

#### 5.17.3 Member Function Documentation

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.17.3.10 `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:

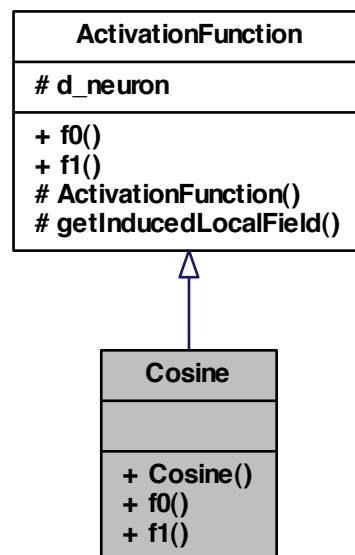
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.18 Cosine Class Reference

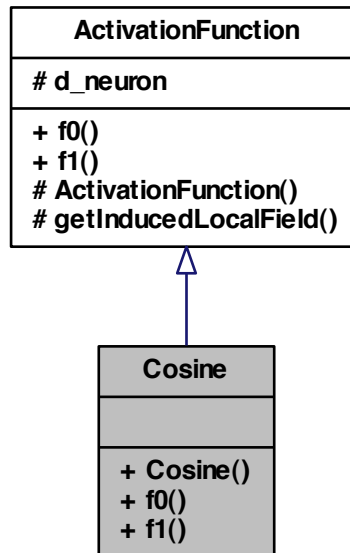
class [Cosine](#) -

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

Inheritance diagram for Cosine:



Collaboration diagram for Cosine:



## Public Member Functions

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

### 5.18.1 Detailed Description

class [Cosine](#) -

Definition at line 5 of file Cosine.h.

### 5.18.2 Constructor & Destructor Documentation

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

### 5.18.3 Member Function Documentation



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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

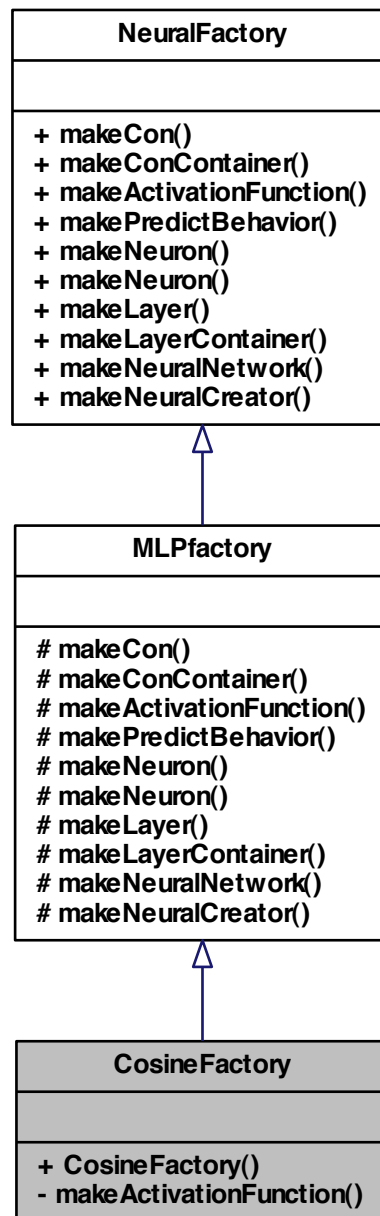
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Cosine.h`

## 5.19 CosineFactory Class Reference

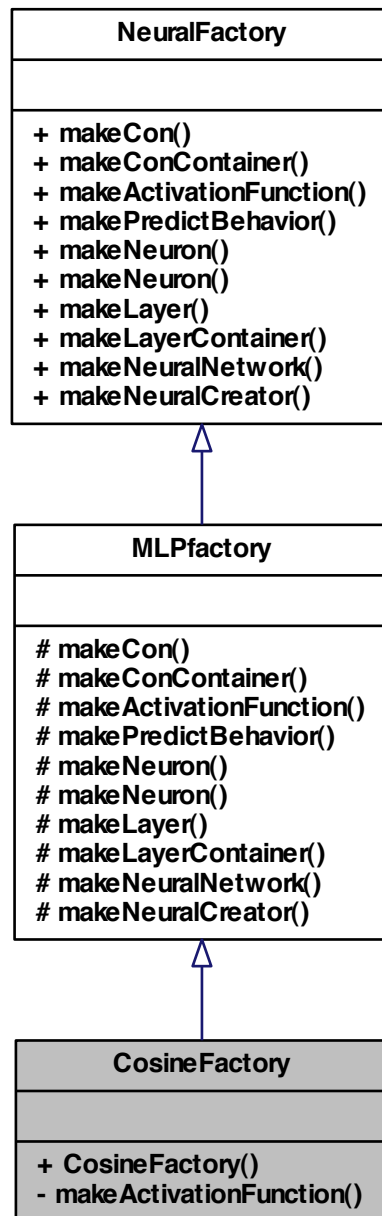
class [CosineFactory](#) -

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

Inheritance diagram for CosineFactory:



Collaboration diagram for CosineFactory:



## Public Member Functions

- [CosineFactory](#) ()

## Private Member Functions

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

### 5.19.1 Detailed Description

class [CosineFactory](#) -

Definition at line 5 of file CosineFactory.h.

### 5.19.2 Constructor & Destructor Documentation

#### 5.19.2.1 [CosineFactory::CosineFactory](#) ( )

### 5.19.3 Member Function Documentation

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

Implements [MLPfactory](#).

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

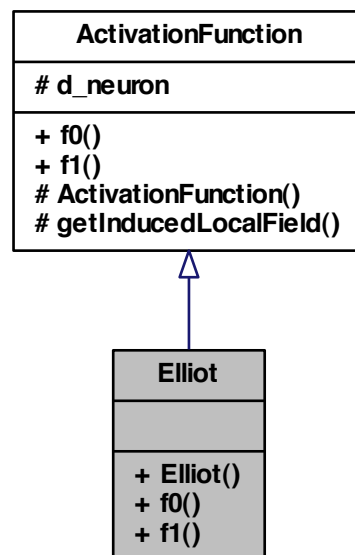
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.20 Elliot Class Reference

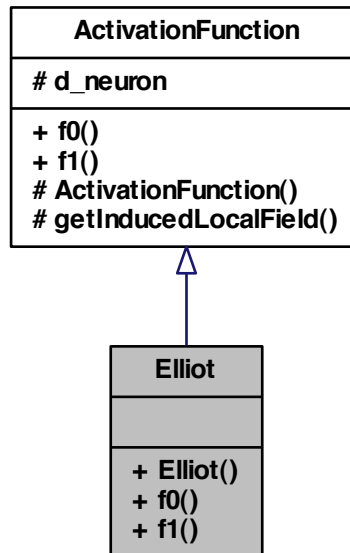
class [Elliot](#) -

```
#include <Elliot.h>
```

Inheritance diagram for Elliot:



Collaboration diagram for Elliot:



## Public Member Functions

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

### 5.20.1 Detailed Description

class [Elliot](#) -

Definition at line 5 of file Elliot.h.

### 5.20.2 Constructor & Destructor Documentation

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

### 5.20.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

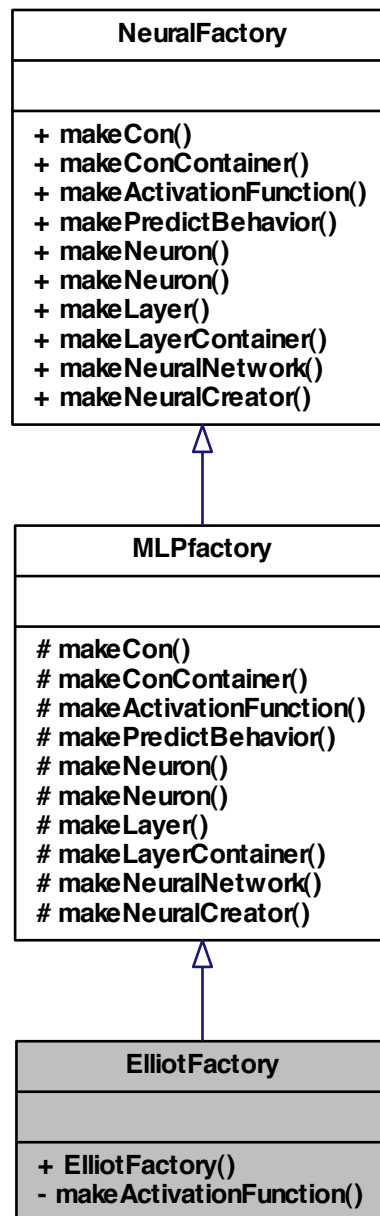
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Elliot.h`

## 5.21 ElliotFactory Class Reference

class [ElliotFactory](#) -

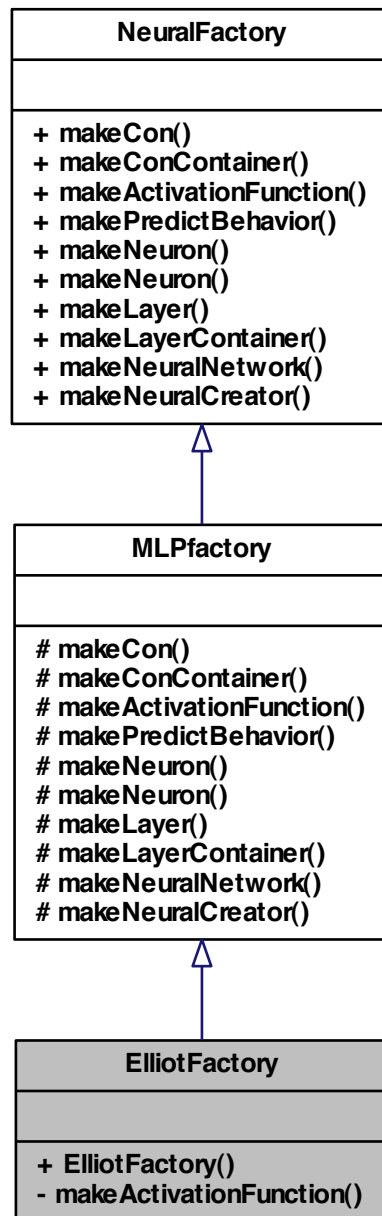
```
#include <ElliotFactory.h>
```

Inheritance diagram for ElliotFactory:





Collaboration diagram for ElliotFactory:



## Public Member Functions

- [ElliotFactory](#) ()

## Private Member Functions

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

### 5.21.1 Detailed Description

class [ElliotFactory](#) -

Definition at line 5 of file ElliotFactory.h.

### 5.21.2 Constructor & Destructor Documentation

5.21.2.1 [ElliotFactory::ElliotFactory](#) ( )

### 5.21.3 Member Function Documentation

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

Implements [MLPfactory](#).

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

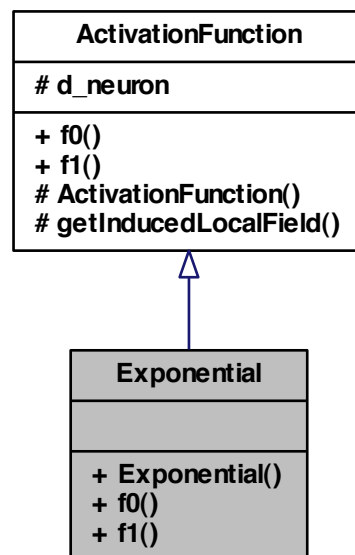
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.22 Exponential Class Reference

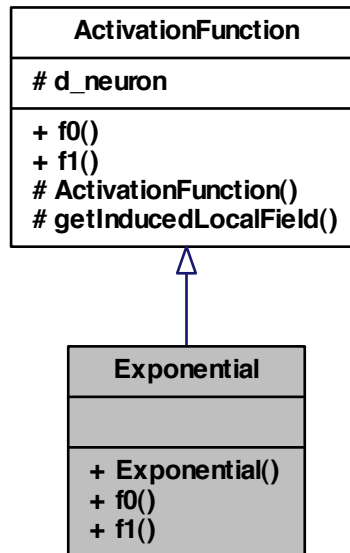
class [Exponential](#) -

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

Inheritance diagram for Exponential:



Collaboration diagram for Exponential:



## Public Member Functions

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

### 5.22.1 Detailed Description

class [Exponential](#) -

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

### 5.22.2 Constructor & Destructor Documentation

5.22.2.1 `Exponential::Exponential ( NeuronPtr neuronPtr )`

### 5.22.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

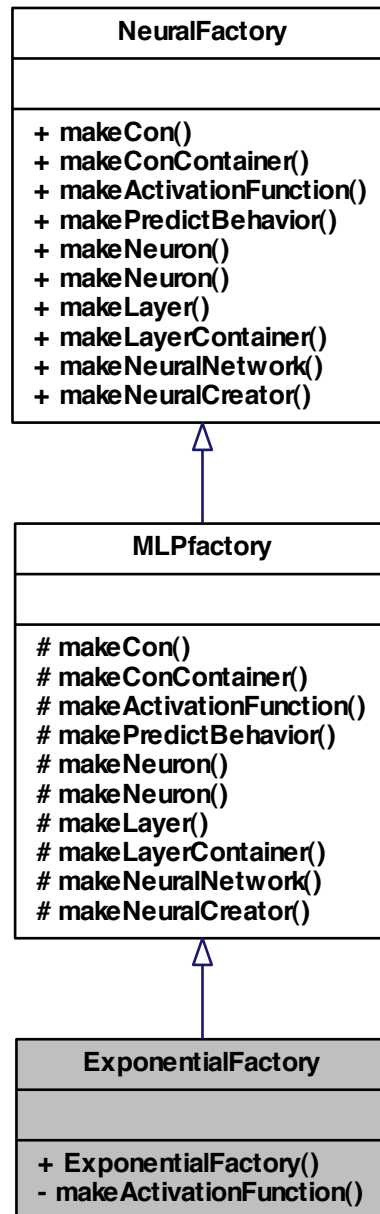
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ExponentialFactory.h`

## 5.23 ExponentialFactory Class Reference

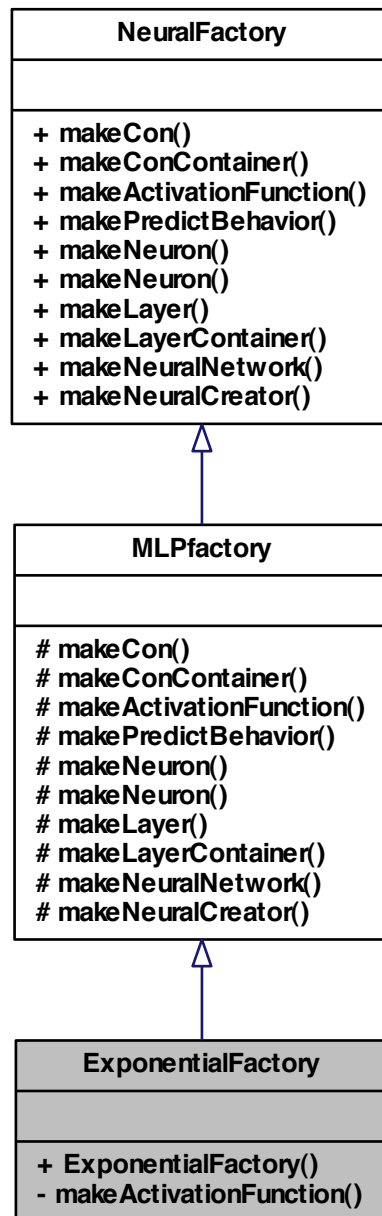
class [ExponentialFactory](#) -

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

Inheritance diagram for ExponentialFactory:



Collaboration diagram for ExponentialFactory:



## Public Member Functions

- [ExponentialFactory](#) ()

## Private Member Functions

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

### 5.23.1 Detailed Description

class [ExponentialFactory](#) -

Definition at line 5 of file ExponentialFactory.h.

### 5.23.2 Constructor & Destructor Documentation

#### 5.23.2.1 [ExponentialFactory::ExponentialFactory](#) ( )

### 5.23.3 Member Function Documentation

#### 5.23.3.1 [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/classHead

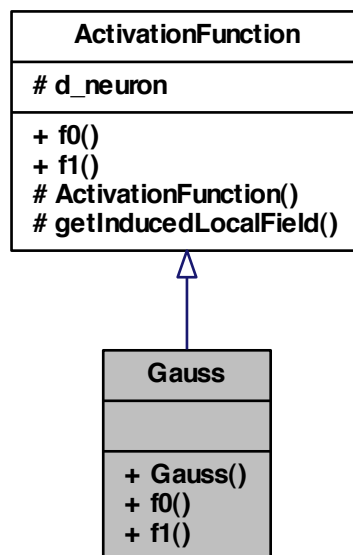
## 5.24 Gauss Class Reference

class [Gauss](#) -

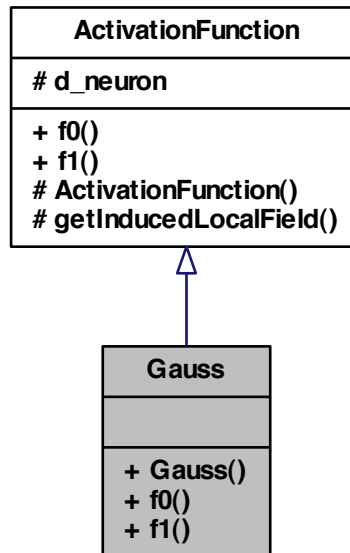
```
#include <Gauss.h>
```



Inheritance diagram for Gauss:



Collaboration diagram for Gauss:



## Public Member Functions

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

### 5.24.1 Detailed Description

class [Gauss](#) -

Definition at line 5 of file Gauss.h.

### 5.24.2 Constructor & Destructor Documentation

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

### 5.24.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

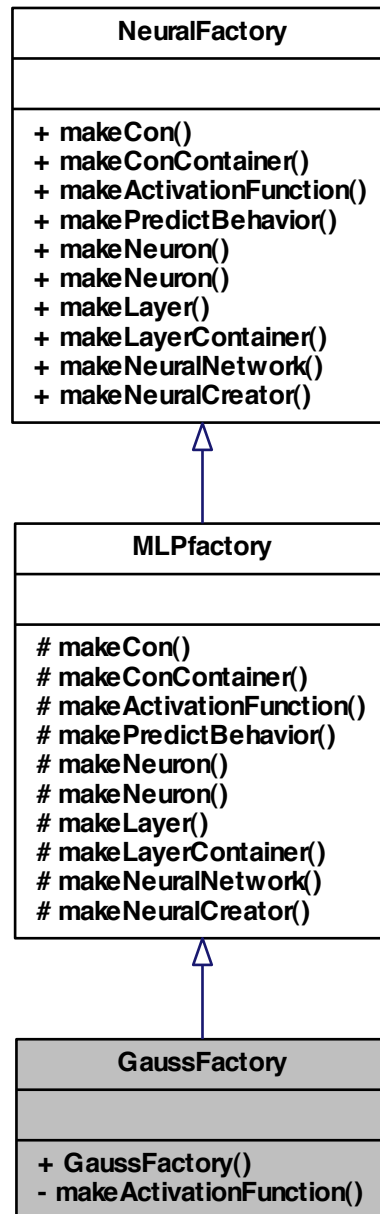
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Gauss.h`

## 5.25 GaussFactory Class Reference

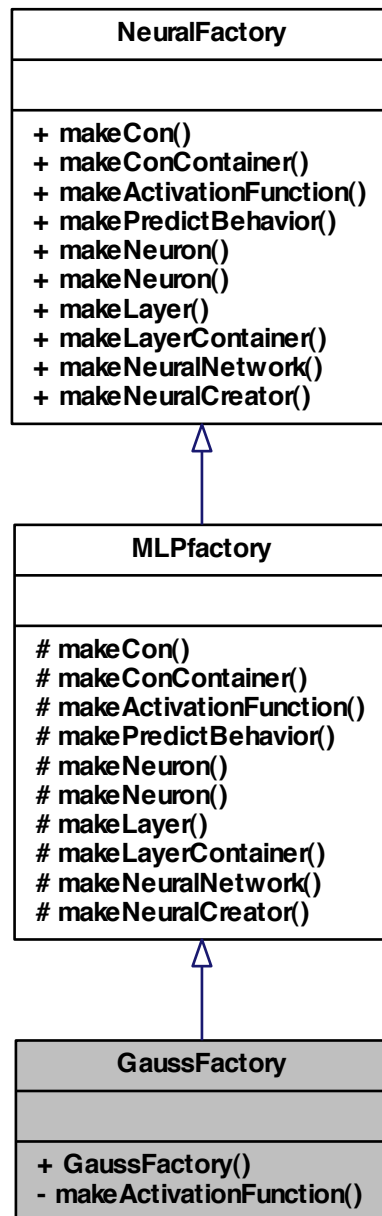
class [GaussFactory](#) -

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

Inheritance diagram for GaussFactory:



Collaboration diagram for GaussFactory:



## Public Member Functions

- [GaussFactory](#) ()

## Private Member Functions

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

### 5.25.1 Detailed Description

class [GaussFactory](#) -

Definition at line 5 of file GaussFactory.h.

### 5.25.2 Constructor & Destructor Documentation

#### 5.25.2.1 [GaussFactory::GaussFactory](#) ( )

### 5.25.3 Member Function Documentation

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

Implements [MLPfactory](#).

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

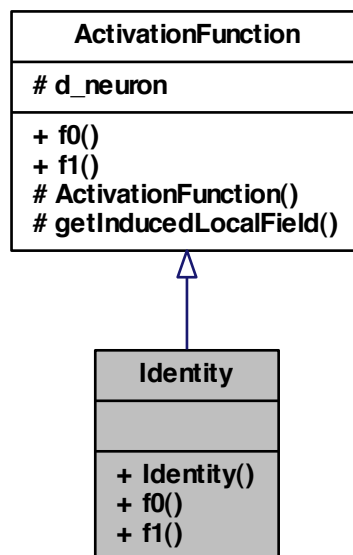
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.26 Identity Class Reference

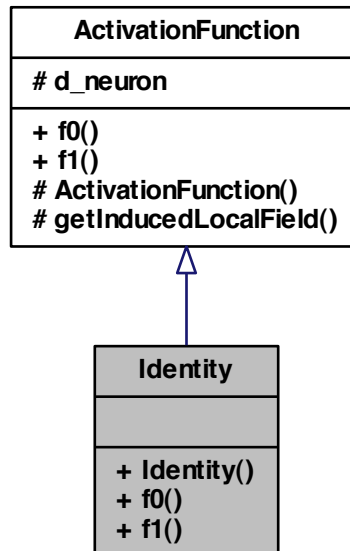
class [Identity](#) -

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

Inheritance diagram for Identity:



Collaboration diagram for Identity:



### Public Member Functions

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

#### 5.26.1 Detailed Description

class [Identity](#) -

Definition at line 5 of file Identity.h.

#### 5.26.2 Constructor & Destructor Documentation

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

Definition at line 13 of file Identity.cpp.

```

: ActivationFunction(neuronPtr) {

```



```
}
```

### 5.26.3 Member Function Documentation

#### 5.26.3.1 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:



#### 5.26.3.2 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:

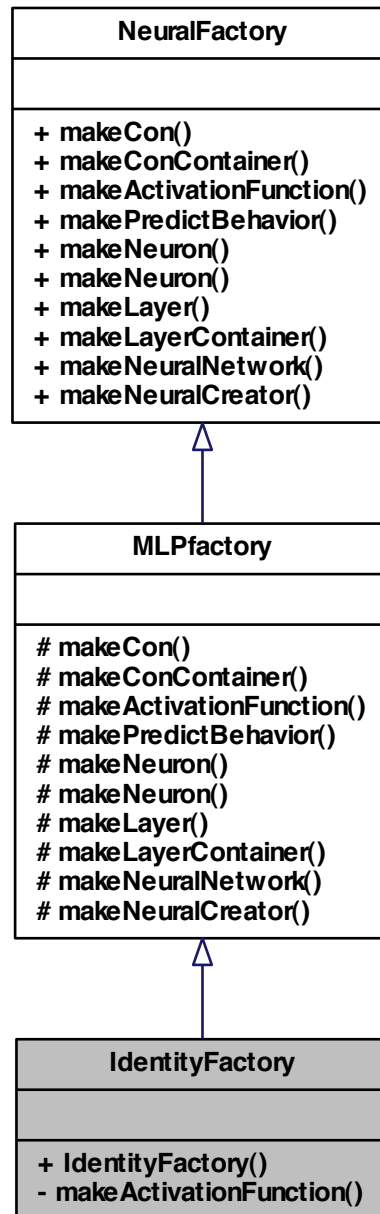
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Identity.h](#)
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Identity.cpp](#)

## 5.27 IdentityFactory Class Reference

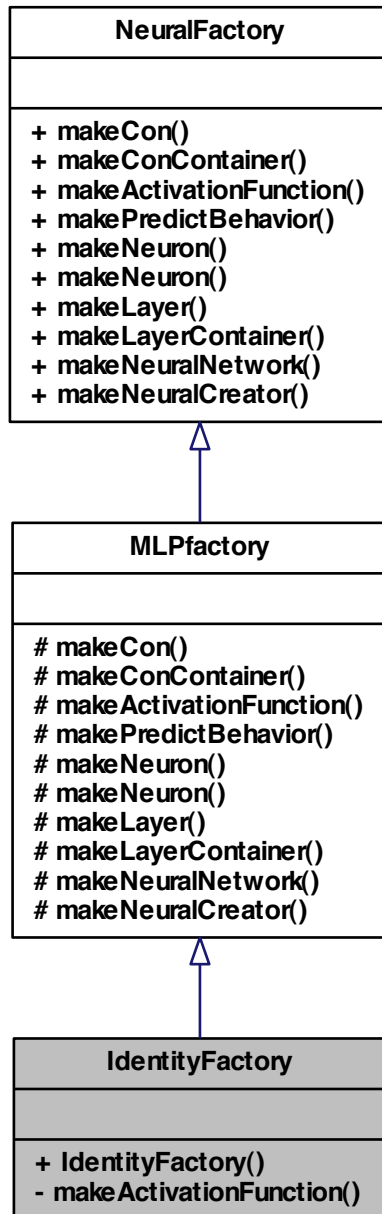
class [IdentityFactory](#) -

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

Inheritance diagram for IdentityFactory:



Collaboration diagram for IdentityFactory:



## Public Member Functions

- [IdentityFactory](#) ()

## Private Member Functions

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

### 5.27.1 Detailed Description

class [IdentityFactory](#) -

Definition at line 5 of file IdentityFactory.h.

### 5.27.2 Constructor & Destructor Documentation

#### 5.27.2.1 IdentityFactory::IdentityFactory ( )

Definition at line 14 of file IdentityFactory.cpp.

```
{
}
```

### 5.27.3 Member Function Documentation

#### 5.27.3.1 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:

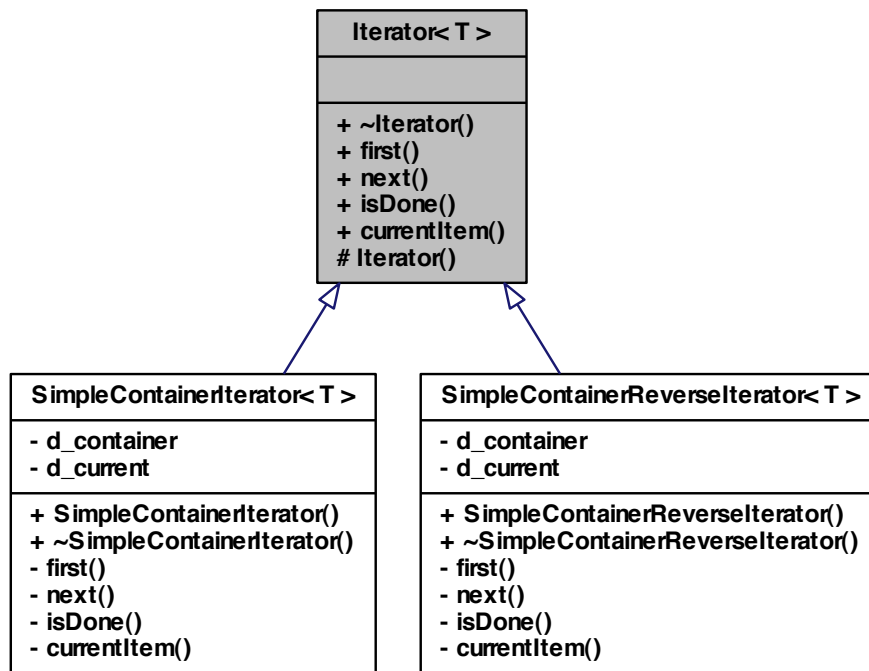
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[IdentityFac](#)

## 5.28 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()`

#### 5.28.1 Detailed Description

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

class [Iterator](#) -

Definition at line 5 of file Iterator.h.

## 5.28.2 Constructor & Destructor Documentation

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

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

## 5.28.3 Member Function Documentation

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

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

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

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

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

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

5.28.3.4 `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:

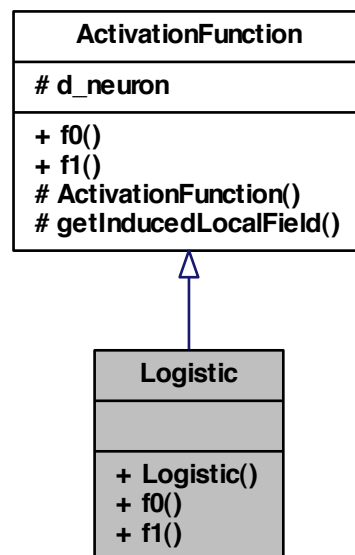
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.29 Logistic Class Reference

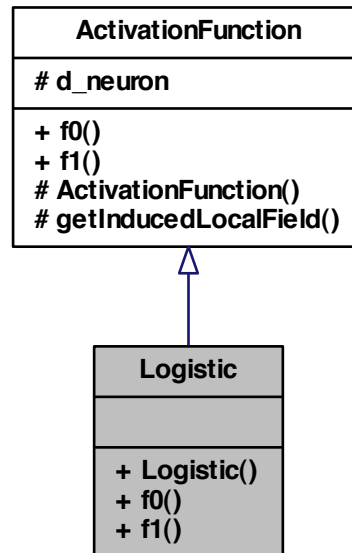
class [Logistic](#) -

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

Inheritance diagram for Logistic:



Collaboration diagram for Logistic:



## Public Member Functions

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

### 5.29.1 Detailed Description

class [Logistic](#) -

Definition at line 5 of file Logistic.h.

### 5.29.2 Constructor & Destructor Documentation

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

### 5.29.3 Member Function Documentation



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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

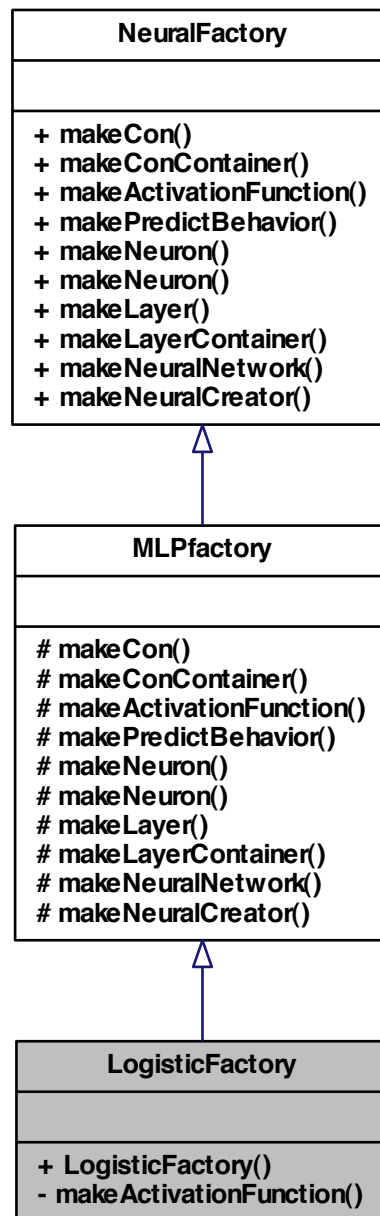
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Logistic.h`

## 5.30 LogisticFactory Class Reference

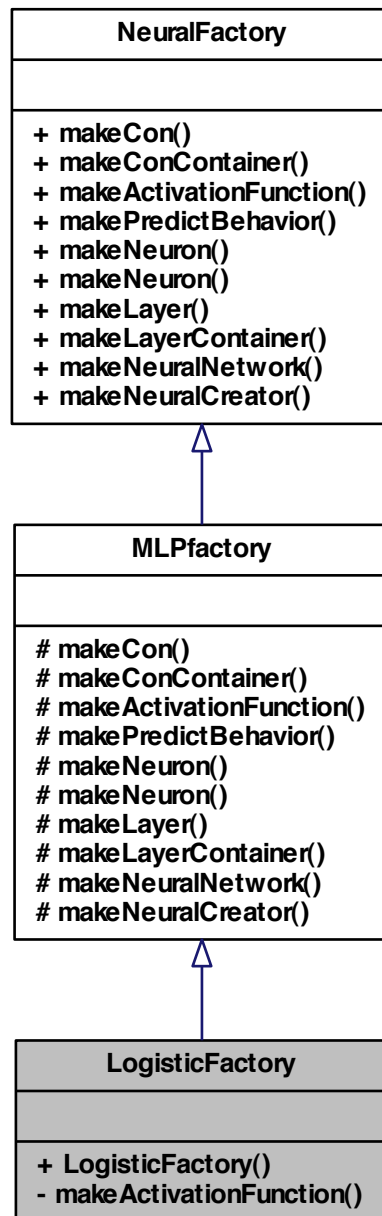
class [LogisticFactory](#) -

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

Inheritance diagram for LogisticFactory:



Collaboration diagram for LogisticFactory:



## Public Member Functions

- [LogisticFactory](#) ()

## Private Member Functions

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

### 5.30.1 Detailed Description

class [LogisticFactory](#) -

Definition at line 5 of file LogisticFactory.h.

### 5.30.2 Constructor & Destructor Documentation

#### 5.30.2.1 [LogisticFactory::LogisticFactory](#) ( )

### 5.30.3 Member Function Documentation

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

Implements [MLPfactory](#).

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

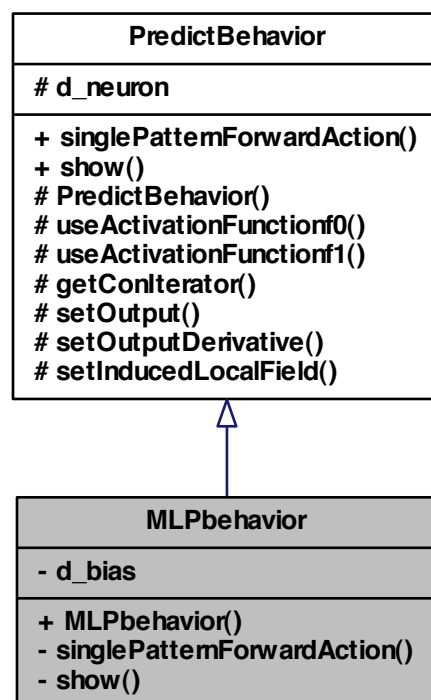
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.31 MLPbehavior Class Reference

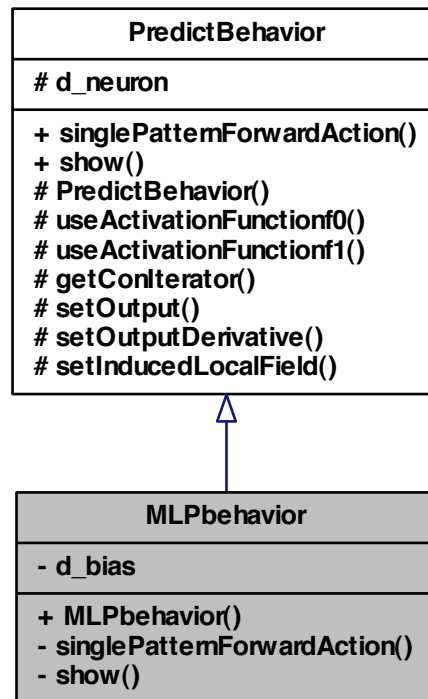
class [MLPbehavior](#) -

```
#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](#)

### 5.31.1 Detailed Description

class [MLPbehavior](#) -

Definition at line 5 of file MLPbehavior.h.

### 5.31.2 Constructor & Destructor Documentation

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

Definition at line 17 of file MLPbehavior.cpp.

```

        PredictBehavior(neuronPtr) , d_bias(0.0)
    {
    }

```

### 5.31.3 Member Function Documentation

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

Implements [PredictBehavior](#).

Definition at line 42 of file MLPbehavior.cpp.

References [d\\_bias](#).

```

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

```

#### 5.31.3.2 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::useActivationFunctionf0\(\)](#), and [PredictBehavior::useActivationFunctionf1\(\)](#).

```

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

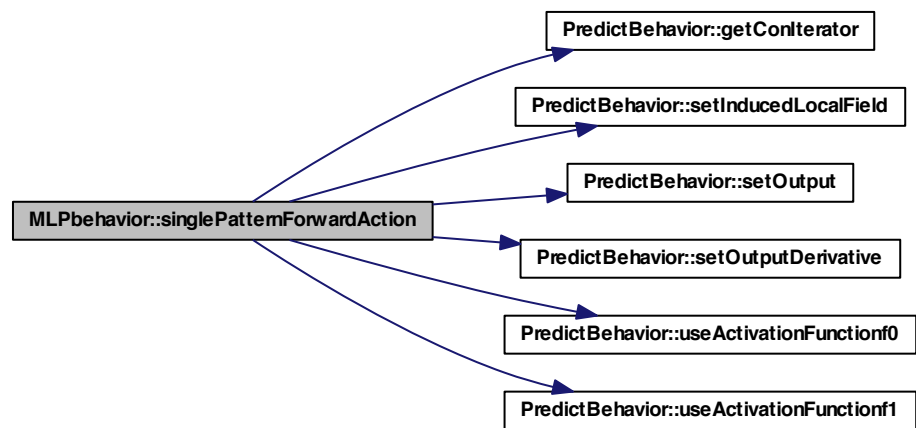
```

```

double weight;
double incomingSignalValue;
for (conIterator->first(); !conIterator->isDone(); conIterator->next())
{
    weight = conIterator->currentItem()->getWeight();
    incomingSignalValue = conIterator->currentItem()->getNeuron().getOutput();
    accumulator += weight * incomingSignalValue;
}
setInducedLocalField(accumulator);
setOutput (useActivationFunctionf0());
setOutputDerivative (useActivationFunctionf1());
}

```

Here is the call graph for this function:



### 5.31.4 Friends And Related Function Documentation

#### 5.31.4.1 friend class `MLPfactory` [friend]

Definition at line 11 of file `MLPbehavior.h`.

### 5.31.5 Member Data Documentation

#### 5.31.5.1 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:



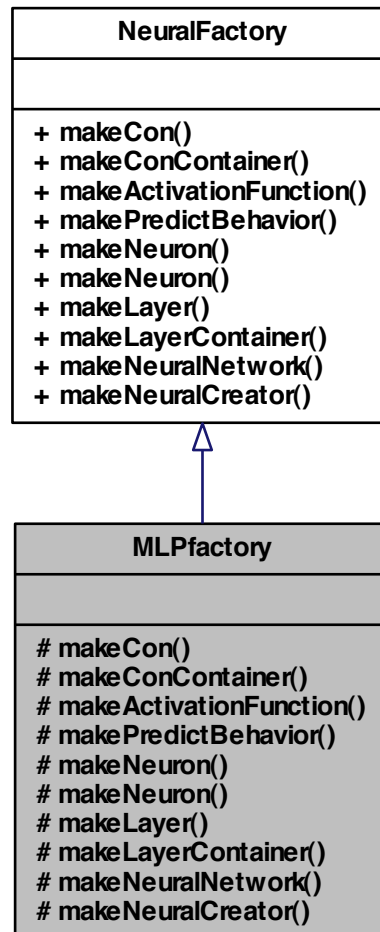
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPbehavior.cpp

class MLPfactory -

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 \(\)](#)

### 5.32.1 Detailed Description

class [MLPfactory](#) -

Definition at line 5 of file MLPfactory.h.

### 5.32.2 Member Function Documentation

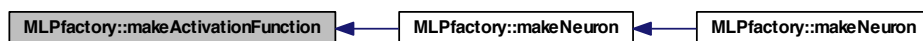
5.32.2.1 **virtual ActivationFunctionPtr MLPfactory::makeActivationFunction ( [NeuronPtr](#) *neuronPtr* )** [protected, pure virtual]

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:



5.32.2.2 **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:



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

Implements [NeuralFactory](#).

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

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

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

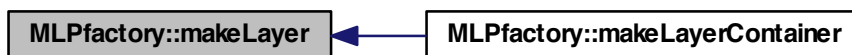
Implements [NeuralFactory](#).

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:



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

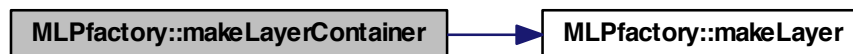
Implements [NeuralFactory](#).

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:



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

Implements [NeuralFactory](#).

Definition at line 109 of file MLPfactory.cpp.

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

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

Implements [NeuralFactory](#).

Definition at line 101 of file MLPfactory.cpp.

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

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

Implements [NeuralFactory](#).

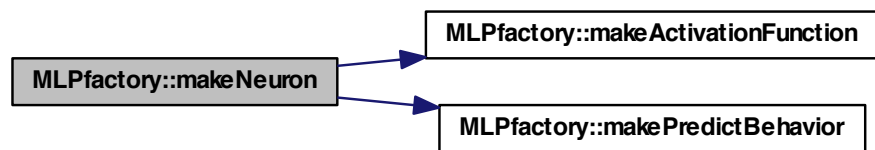
Definition at line 52 of file MLPfactory.cpp.

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:



### 5.32.2.9 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;

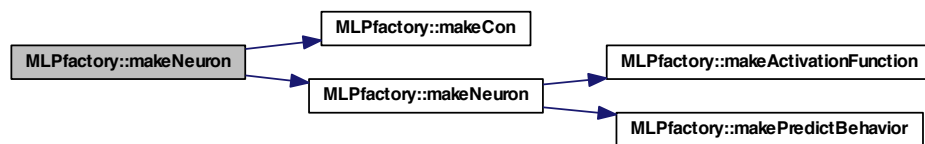
    NeuronPtr neuronPtr(makeNeuron(Id));

    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:



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

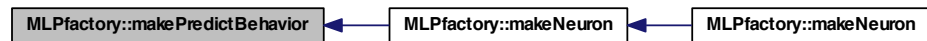
Implements [NeuralFactory](#).

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

### 5.33 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`

#### 5.33.1 Detailed Description

class [NetworkRinterface](#) -

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



### 5.33.2 Constructor & Destructor Documentation

#### 5.33.2.1 NetworkRinterface::NetworkRinterface ( )

Definition at line 22 of file NetworkRinterface.cpp.

```
{
}
```

### 5.33.3 Member Function Documentation

#### 5.33.3.1 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:



#### 5.33.3.2 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:



### 5.33.3.3 `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:



### 5.33.3.4 `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");
}

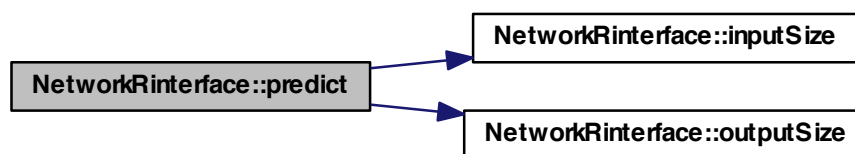
bool checkIncorrectNumberOfRows(
    inputSize() != static_cast<size_type> (numericMatrix.nrow()));
if (checkIncorrectNumberOfRows)
{
    throw std::runtime_error(
        "\nIncorrect number or rows. The number of input neurons must be equal
        to the number of rows of the input matrix.\n");
}

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:



#### 5.33.3.5 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:



#### 5.33.3.6 Rcpg::List NetworkRinterface::train ( Rcpg::List *parameterList* )

Definition at line 77 of file NetworkRinterface.cpp.

References `d_neuralNetwork`.

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

#### 5.33.3.7 `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:



### 5.33.4 Member Data Documentation

#### 5.33.4.1 `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:

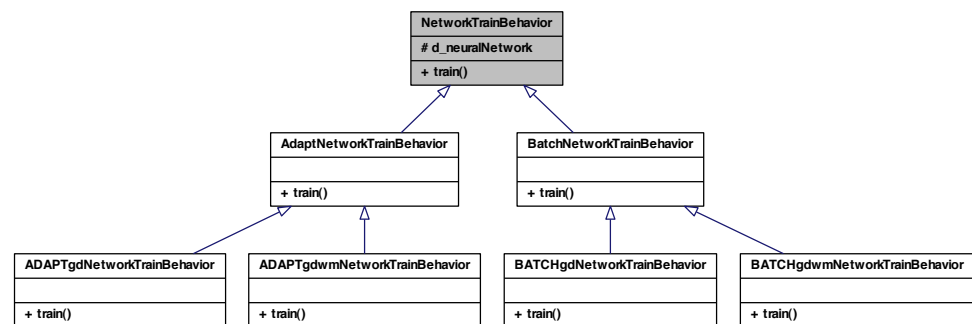
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeader
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[NetworkTrainBehavior.h](#)

## 5.34 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](#)

#### 5.34.1 Detailed Description

class [NetworkTrainBehavior](#) -

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

#### 5.34.2 Member Function Documentation

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

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

### 5.34.3 Member Data Documentation

5.34.3.1 `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:

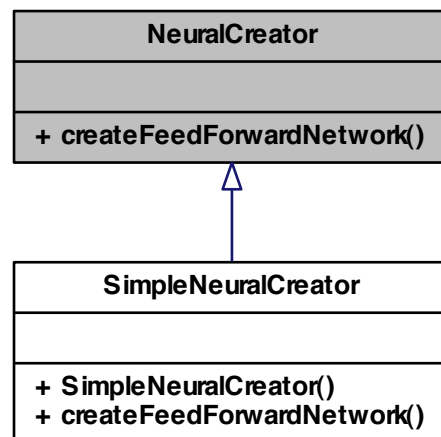
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkTrainBehavior.h`

## 5.35 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

5.35.1 Detailed Description

class [NeuralCreator](#) -  
Definition at line 4 of file NeuralCreator.h.

5.35.2 Member Function Documentation

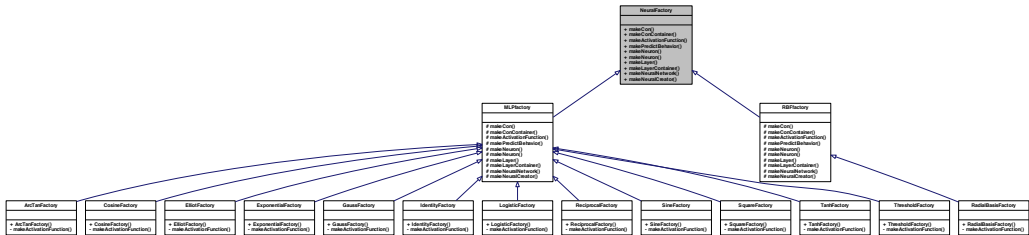
5.35.2.1 virtual [NeuralNetworkPtr](#) [NeuralCreator::createFeedForwardNetwork](#) ( std::vector< int > *numberOfNeurons*, [NeuralFactory](#) & *hiddenLayersFactory*, [NeuralFactory](#) & *outputLayerFactory* ) [pure virtual]

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/classHead

5.36 NeuralFactory Class Reference

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



Public Member Functions

- virtual [ConPtr](#) [makeCon](#) ([Neuron](#) &neuron, double weight)=0
- virtual [ConContainerPtr](#) [makeConContainer](#) ()=0



- virtual [ActivationFunctionPtr](#) `makeActivationFunction` ([NeuronPtr](#) neuronPtr)=0
- virtual [PredictBehaviorPtr](#) `makePredictBehavior` ([NeuronPtr](#) neuronPtr)=0
- virtual [NeuronPtr](#) `makeNeuron` ([Handler](#) Id)=0
- virtual [NeuronPtr](#) `makeNeuron` ([Handler](#) Id, [NeuronIteratorPtr](#) neuronIteratorPtr, double totalAmountOfParameters)=0
- virtual [LayerPtr](#) `makeLayer` ()=0
- virtual [LayerContainerPtr](#) `makeLayerContainer` ()=0
- virtual [NeuralNetworkPtr](#) `makeNeuralNetwork` ([NeuralFactory](#) &neuralFactory)=0
- virtual [NeuralCreatorPtr](#) `makeNeuralCreator` ()=0

### 5.36.1 Detailed Description

class [NeuralFactory](#) -

Definition at line 4 of file NeuralFactory.h.

### 5.36.2 Member Function Documentation

5.36.2.1 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](#).

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

Implemented in [MLPfactory](#).

5.36.2.3 virtual [ConContainerPtr](#) `NeuralFactory::makeConContainer` ( ) [pure virtual]

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

Referenced by `Neuron::Neuron()`.

Here is the caller graph for this function:

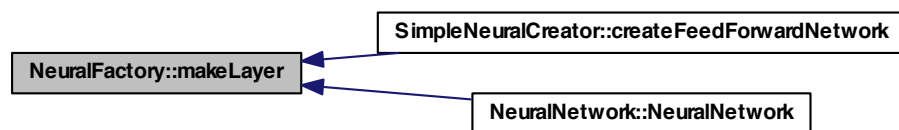


#### 5.36.2.4 `virtual LayerPtr NeuralFactory::makeLayer( ) [pure virtual]`

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

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

Here is the caller graph for this function:



#### 5.36.2.5 `virtual LayerContainerPtr NeuralFactory::makeLayerContainer( ) [pure virtual]`

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

Referenced by `NeuralNetwork::NeuralNetwork()`.

Here is the caller graph for this function:



5.36.2.6 `virtual NeuralCreatorPtr NeuralFactory::makeNeuralCreator ( ) [pure virtual]`

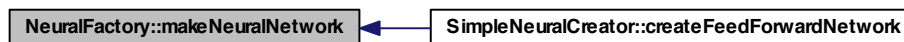
Implemented in [MLPfactory](#), and [RBFfactory](#).

5.36.2.7 `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:

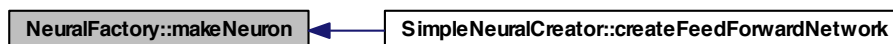


5.36.2.8 `virtual NeuronPtr NeuralFactory::makeNeuron ( Handler Id ) [pure virtual]`

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

Referenced by `SimpleNeuralCreator::createFeedForwardNetwork()`.

Here is the caller graph for this function:



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

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

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

Implemented in [MLPfactory](#).

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

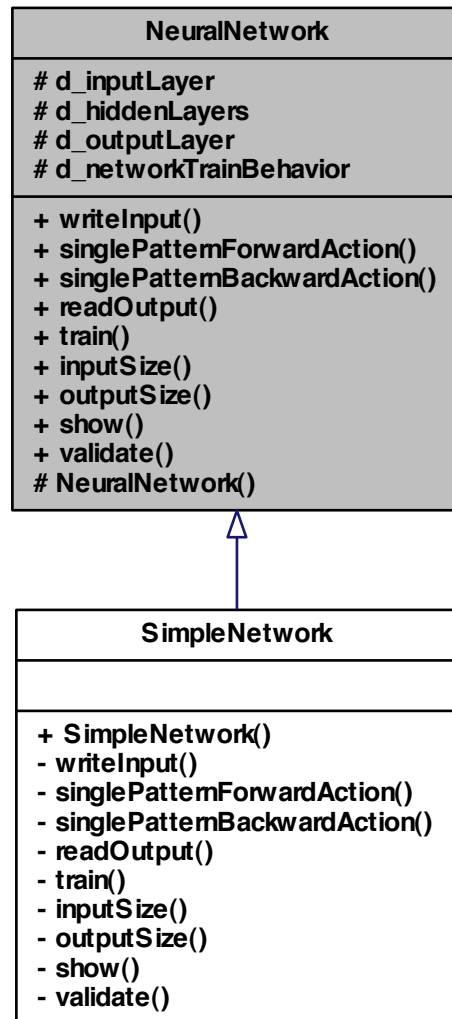
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead`

## 5.37 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](#)

## 5.37.1 Detailed Description

class [NeuralNetwork](#) -

Definition at line 3 of file NeuralNetwork.h.

## 5.37.2 Constructor & Destructor Documentation

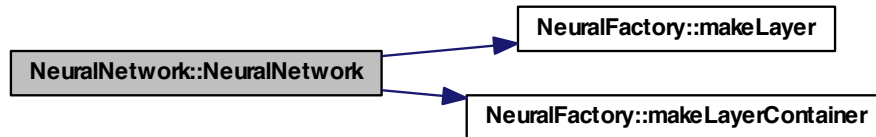
### 5.37.2.1 [NeuralNetwork::NeuralNetwork](#) ( [NeuralFactory](#) & *neuralFactory* ) [protected]

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:



### 5.37.3 Member Function Documentation

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

Implemented in [SimpleNetwork](#).

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

Implemented in [SimpleNetwork](#).

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

Implemented in [SimpleNetwork](#).

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

Implemented in [SimpleNetwork](#).

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

Implemented in [SimpleNetwork](#).

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

Implemented in [SimpleNetwork](#).

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

Implemented in [SimpleNetwork](#).

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

Implemented in [SimpleNetwork](#).

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

Implemented in [SimpleNetwork](#).

## 5.37.4 Friends And Related Function Documentation

**5.37.4.1** `friend class SimpleNeuralCreator` [friend]

Definition at line 12 of file NeuralNetwork.h.

## 5.37.5 Member Data Documentation

**5.37.5.1** `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().

**5.37.5.2** `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().

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

Definition at line 9 of file NeuralNetwork.h.

Referenced by SimpleNetwork::train().



#### 5.37.5.4 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:

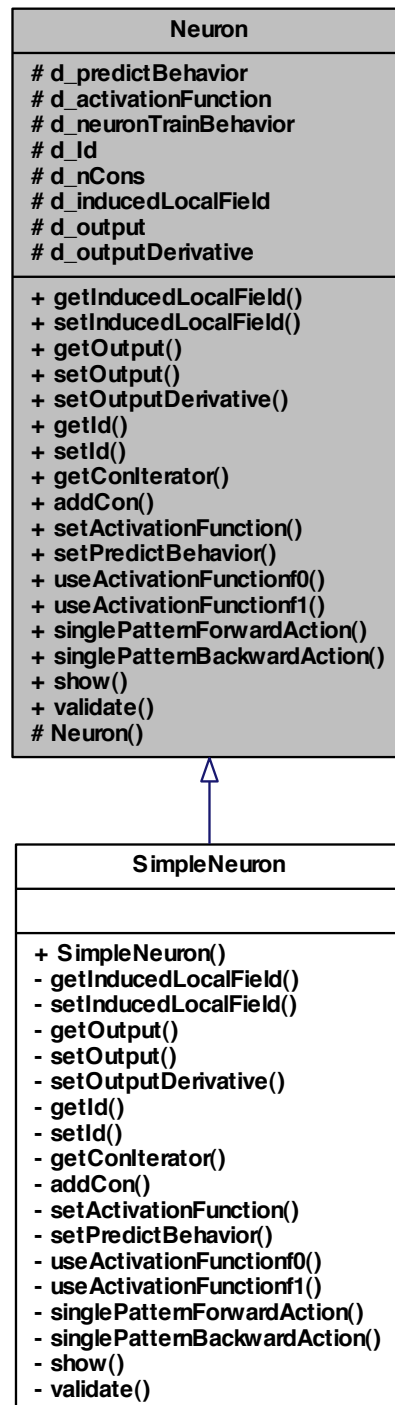
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[NeuralNe](#)
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[NeuralNetwork.cpp](#)

## 5.38 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](#)

#### 5.38.1 Detailed Description

class [Neuron](#) -

Definition at line 3 of file Neuron.h.

## 5.38.2 Constructor & Destructor Documentation

### 5.38.2.1 `Neuron::Neuron ( NeuralFactory & neuralFactory )` `[protected]`

Definition at line 12 of file `Neuron.cpp`.

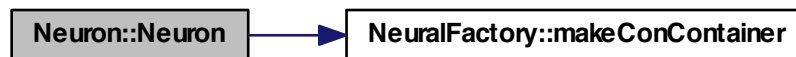
References `d_nCons`, and `NeuralFactory::makeConContainer()`.

```

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

```

Here is the call graph for this function:



## 5.38.3 Member Function Documentation

### 5.38.3.1 `virtual void Neuron::addCon ( ConPtr conPtr )` `[pure virtual]`

Implemented in [SimpleNeuron](#).

### 5.38.3.2 `virtual ConIteratorPtr Neuron::getConIterator ( )` `[pure virtual]`

Implemented in [SimpleNeuron](#).

### 5.38.3.3 `virtual Handler Neuron::getId ( )` `[pure virtual]`

Implemented in [SimpleNeuron](#).

### 5.38.3.4 `virtual double Neuron::getInducedLocalField ( )` `[pure virtual]`

Implemented in [SimpleNeuron](#).

### 5.38.3.5 `virtual double Neuron::getOutput ( )` `[pure virtual]`

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

**5.38.3.16** `virtual double Neuron::useActivationFunction1 ( )` [pure virtual]

Implemented in [SimpleNeuron](#).

**5.38.3.17** `virtual bool Neuron::validate ( )` [pure virtual]

Implemented in [SimpleNeuron](#).

## 5.38.4 Friends And Related Function Documentation

**5.38.4.1** `friend class MLPfactory` [friend]

Definition at line 16 of file Neuron.h.

## 5.38.5 Member Data Documentation

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

Definition at line 7 of file Neuron.h.

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

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

Definition at line 9 of file Neuron.h.

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

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

Definition at line 11 of file Neuron.h.

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

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

Definition at line 10 of file Neuron.h.

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

**5.38.5.5** `NeuronTrainBehaviorPtr Neuron::d_neuronTrainBehavior`  
[protected]

Definition at line 8 of file Neuron.h.

Referenced by SimpleNeuron::singlePatternBackwardAction().

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

Definition at line 12 of file Neuron.h.

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

#### 5.38.5.7 double Neuron::d\_outputDerivative [protected]

Definition at line 13 of file Neuron.h.

Referenced by SimpleNeuron::setOutputDerivative().

#### 5.38.5.8 PredictBehaviorPtr Neuron::d\_predictBehavior [protected]

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:

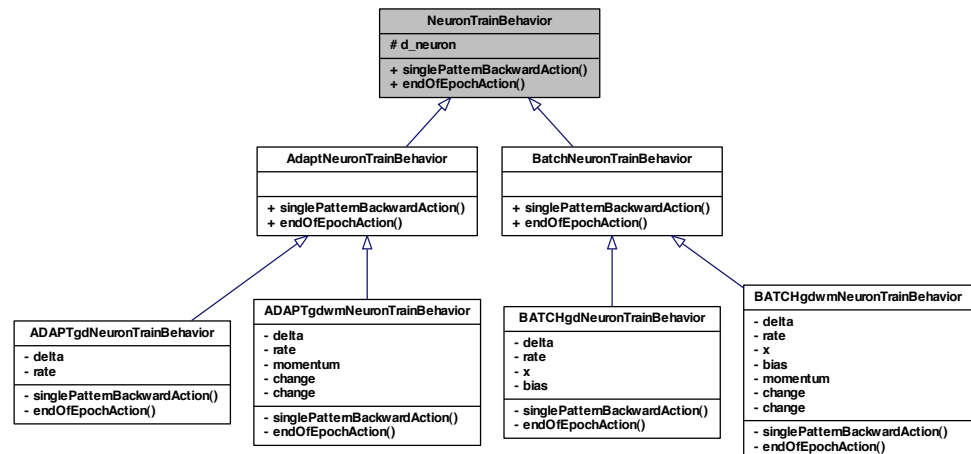
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[Neuron.h](#)
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[Neuron.cpp](#)

## 5.39 NeuronTrainBehavior Class Reference

class [NeuronTrainBehavior](#) -

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

Inheritance diagram for NeuronTrainBehavior:



## Public Member Functions

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

## Protected Attributes

- [NeuronWeakPtr d\\_neuron](#)

### 5.39.1 Detailed Description

class [NeuronTrainBehavior](#) -

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

### 5.39.2 Member Function Documentation

#### 5.39.2.1 virtual void NeuronTrainBehavior::endOfEpochAction ( ) [pure virtual]

Implemented in [ADAPTgdNeuronTrainBehavior](#), [ADAPTgdwmNeuronTrainBehavior](#), [AdaptNeuronTrainBehavior](#), [BATCHgdNeuronTrainBehavior](#), [BATCHgdwmNeuronTrainBehavior](#), and [BatchNeuronTrainBehavior](#).



5.39.2.2 `virtual void NeuronTrainBehavior::singlePatternBackwardAction ( )` [pure virtual]

Implemented in [ADAPTgdNeuronTrainBehavior](#), [ADAPTgdwmNeuronTrainBehavior](#), [AdaptNeuronTrainBehavior](#), [BATCHgdNeuronTrainBehavior](#), [BATCHgdwmNeuronTrainBehavior](#), and [BatchNeuronTrainBehavior](#).

### 5.39.3 Member Data Documentation

5.39.3.1 `NeuronWeakPtr NeuronTrainBehavior::d_neuron` [protected]

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

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

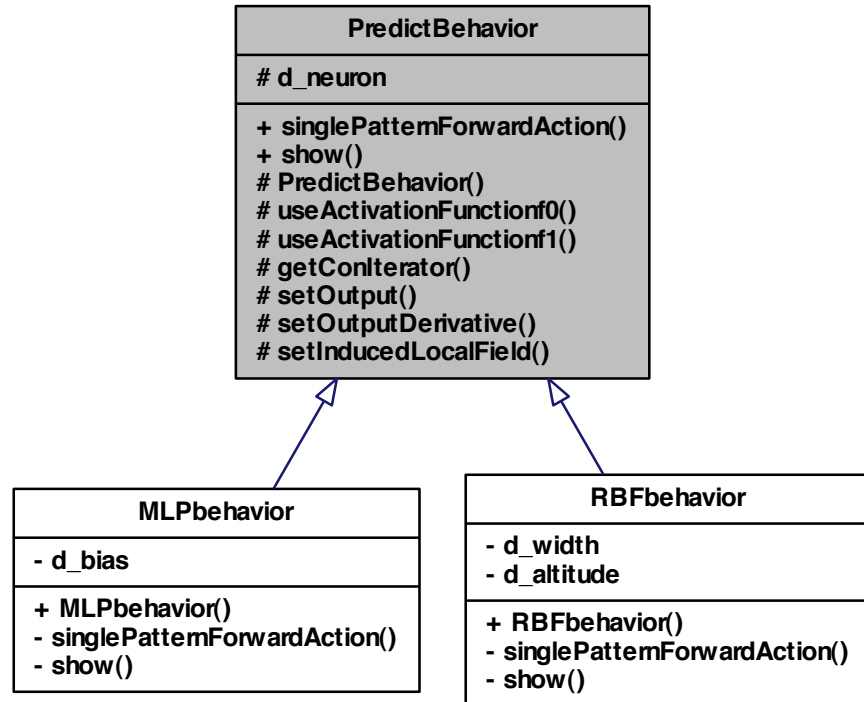
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuronTrainBehavior.h`

## 5.40 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](#) ()
- [ConlteratorPtr](#) [getConlterator](#) ()
- void [setOutput](#) (double output)
- void [setOutputDerivative](#) (double outputDerivative)
- void [setInducedLocalField](#) (double inducedLocalField)

## Protected Attributes

- [NeuronWeakPtr d\\_neuron](#)

### 5.40.1 Detailed Description

class [PredictBehavior](#) -

Definition at line 4 of file PredictBehavior.h.

### 5.40.2 Constructor & Destructor Documentation

#### 5.40.2.1 PredictBehavior::PredictBehavior ( [NeuronPtr neuronPtr](#) ) [protected]

Definition at line 14 of file PredictBehavior.cpp.

```

    d_neuron(neuronPtr)
{
}
:
```

### 5.40.3 Member Function Documentation

#### 5.40.3.1 ConlteratorPtr PredictBehavior::getConlterator ( ) [protected]

Definition at line 36 of file PredictBehavior.cpp.

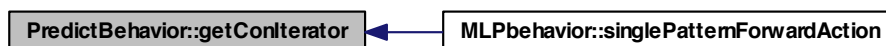
References [d\\_neuron](#).

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

```

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

Here is the caller graph for this function:



#### 5.40.3.2 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:



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

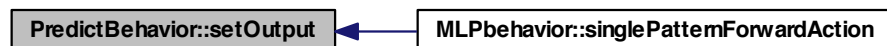
Definition at line 43 of file PredictBehavior.cpp.

References `d_neuron`.

Referenced by `MLPbehavior::singlePatternForwardAction()`.

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

Here is the caller graph for this function:



#### 5.40.3.4 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:



#### 5.40.3.5 virtual void PredictBehavior::show ( ) [pure virtual]

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

#### 5.40.3.6 virtual void PredictBehavior::singlePatternForwardAction ( ) [pure virtual]

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

#### 5.40.3.7 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:



#### 5.40.3.8 `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:



### 5.40.4 Member Data Documentation

#### 5.40.4.1 `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:

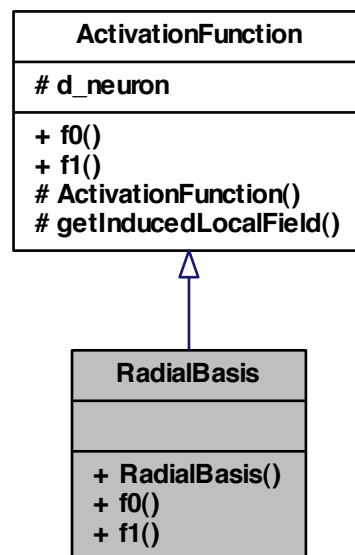
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead`
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/PredictBeh`

## 5.41 RadialBasis Class Reference

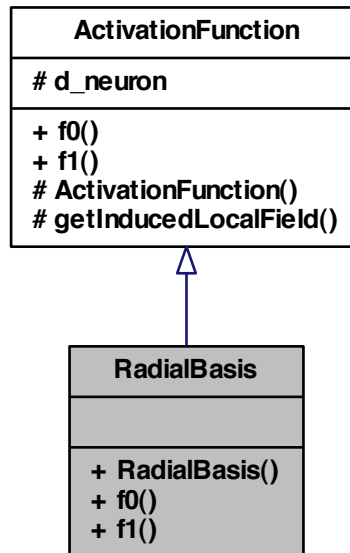
class [RadialBasis](#) -

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

Inheritance diagram for RadialBasis:



Collaboration diagram for RadialBasis:



## Public Member Functions

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

### 5.41.1 Detailed Description

class [RadialBasis](#) -

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

### 5.41.2 Constructor & Destructor Documentation

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

### 5.41.3 Member Function Documentation



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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

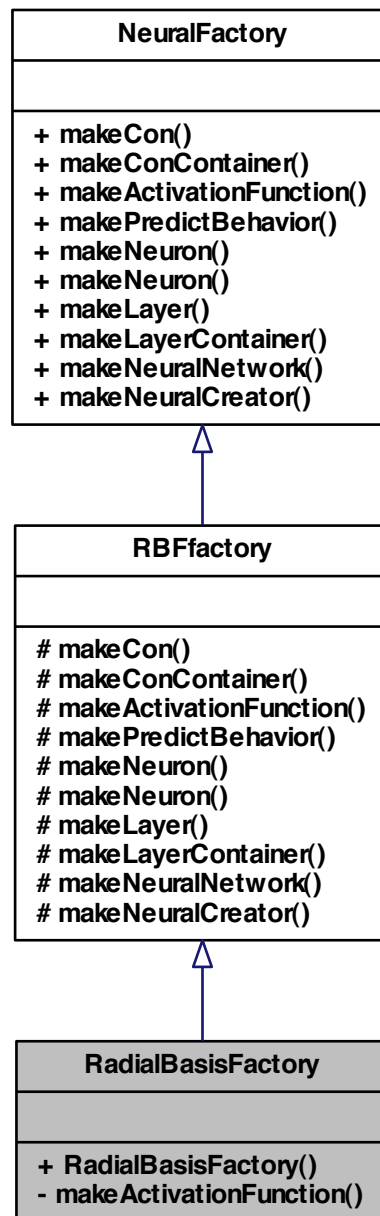
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasisFactory.h`

## 5.42 RadialBasisFactory Class Reference

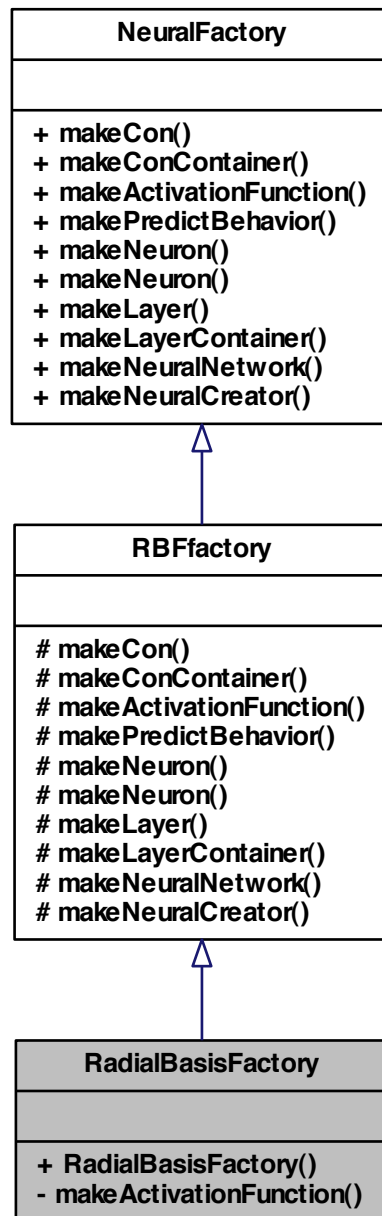
class [RadialBasisFactory](#) -

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

Inheritance diagram for RadialBasisFactory:



Collaboration diagram for RadialBasisFactory:



## Public Member Functions

- [RadialBasisFactory](#) ()

## Private Member Functions

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

### 5.42.1 Detailed Description

class [RadialBasisFactory](#) -

Definition at line 5 of file RadialBasisFactory.h.

### 5.42.2 Constructor & Destructor Documentation

#### 5.42.2.1 [RadialBasisFactory::RadialBasisFactory](#) ( )

### 5.42.3 Member Function Documentation

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

Implements [RBFfactory](#).

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

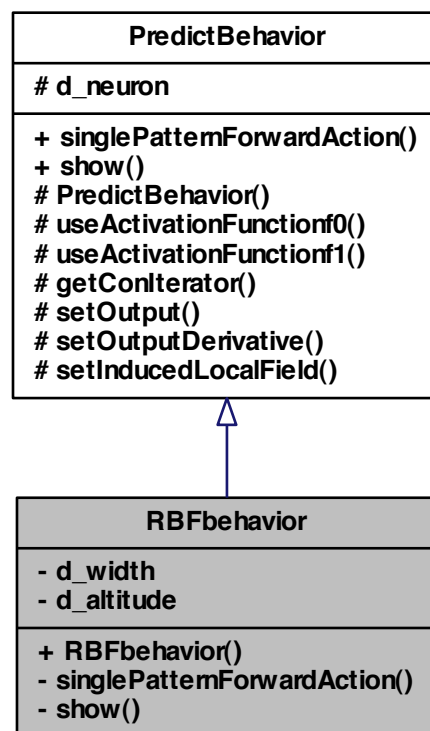
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.43 RBFbehavior Class Reference

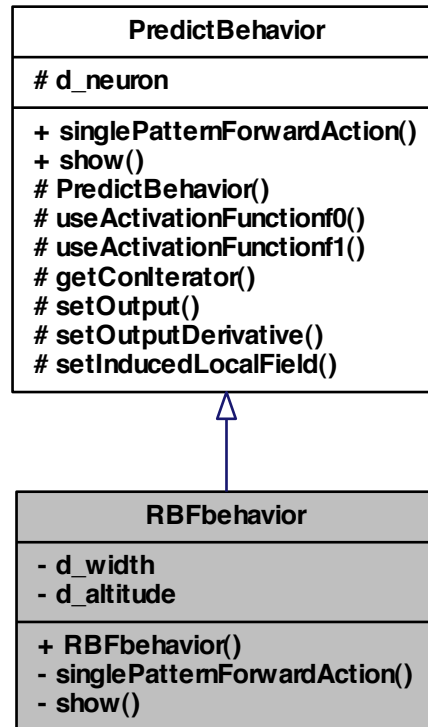
class [RBFbehavior](#) -

```
#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](#)

### 5.43.1 Detailed Description

class [RBFbehavior](#) -

Definition at line 5 of file RBFbehavior.h.

### 5.43.2 Constructor & Destructor Documentation

5.43.2.1 [RBFbehavior::RBFbehavior \( \*NeuronPtr neuronPtr\* \)](#)

### 5.43.3 Member Function Documentation

5.43.3.1 [void RBFbehavior::show \( \)](#) [*private*, *virtual*]

Implements [PredictBehavior](#).

5.43.3.2 [void RBFbehavior::singlePatternForwardAction \( \)](#) [*private*, *virtual*]

Implements [PredictBehavior](#).

### 5.43.4 Member Data Documentation

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

Definition at line 9 of file RBFbehavior.h.

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

Definition at line 8 of file RBFbehavior.h.

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

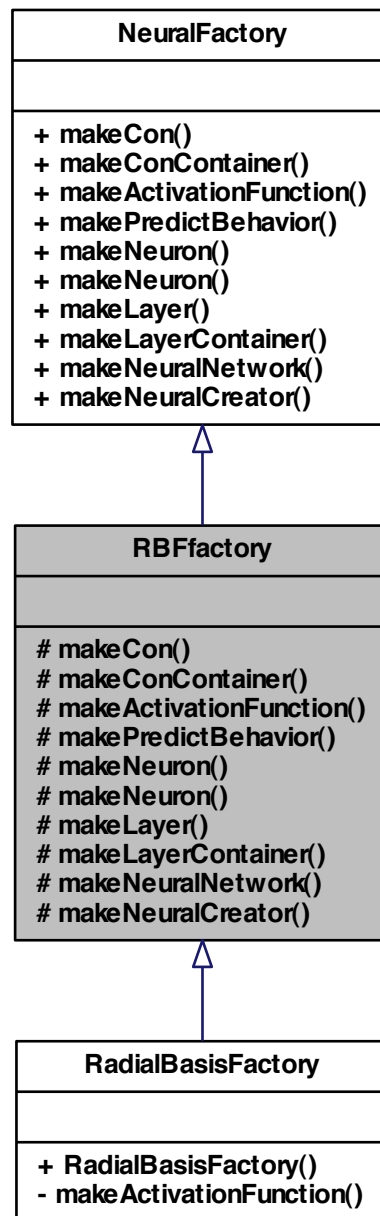
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFbehavior.h](#)

## 5.44 RBFfactory Class Reference

class [RBFfactory](#) -

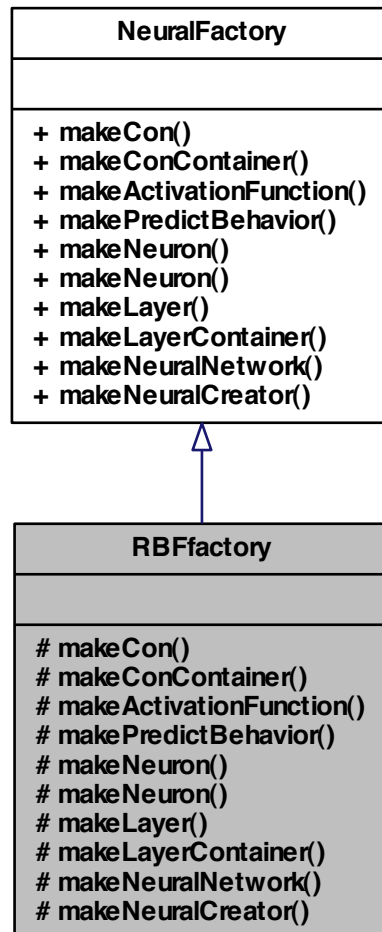
```
#include <RBFfactory.h>
```

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 ()`

### 5.44.1 Detailed Description

class [RBFfactory](#) -

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

### 5.44.2 Member Function Documentation

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

Implements [NeuralFactory](#).

Implemented in [RadialBasisFactory](#).

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

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

Implements [NeuralFactory](#).

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

Implements [NeuralFactory](#).

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

Implements [NeuralFactory](#).

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

Implements [NeuralFactory](#).

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

Implements [NeuralFactory](#).

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

Implements [NeuralFactory](#).

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

Implements [NeuralFactory](#).

5.44.2.10 **PredictBehaviorPtr** RBFfactory::makePredictBehavior ( ) [protected]

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

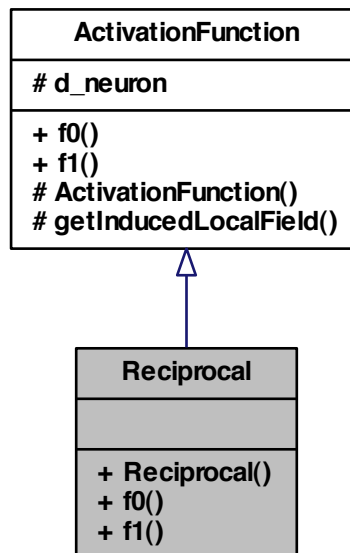
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[RBFfactory.h](#)

## 5.45 Reciprocal Class Reference

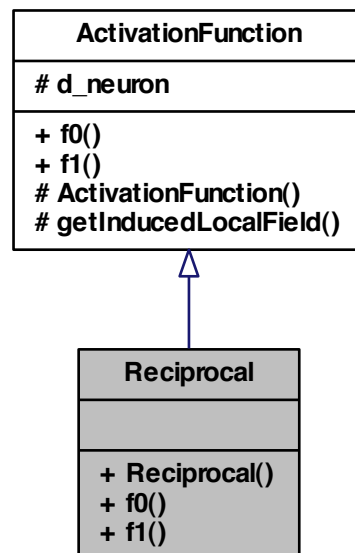
class [Reciprocal](#) -

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

Inheritance diagram for Reciprocal:



Collaboration diagram for Reciprocal:



### Public Member Functions

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

### 5.45.1 Detailed Description

class [Reciprocal](#) -

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

### 5.45.2 Constructor & Destructor Documentation

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

### 5.45.3 Member Function Documentation

5.45.3.1 void Reciprocal::f0 ( ) [virtual]

Implements [ActivationFunction](#).

5.45.3.2 void Reciprocal::f1 ( ) [virtual]

Implements [ActivationFunction](#).

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

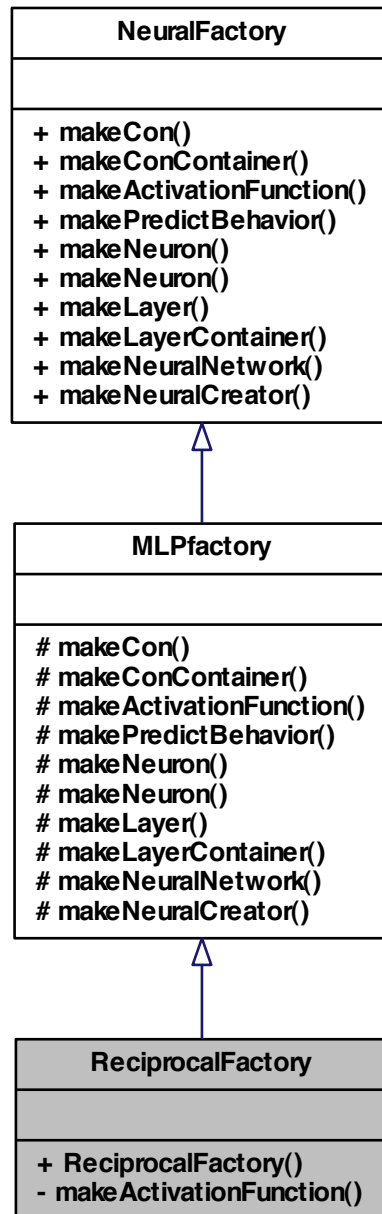
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.46 ReciprocalFactory Class Reference

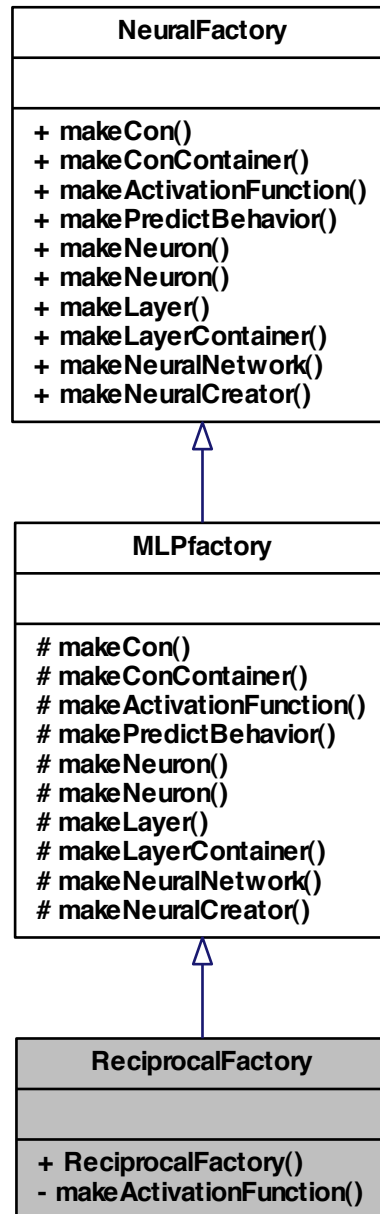
class [ReciprocalFactory](#) -

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

Inheritance diagram for ReciprocalFactory:



Collaboration diagram for ReciprocalFactory:





## Public Member Functions

- [ReciprocalFactory](#) ()

## Private Member Functions

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

### 5.46.1 Detailed Description

class [ReciprocalFactory](#) -

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

### 5.46.2 Constructor & Destructor Documentation

5.46.2.1 [ReciprocalFactory::ReciprocalFactory](#) ( )

### 5.46.3 Member Function Documentation

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

Implements [MLPfactory](#).

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

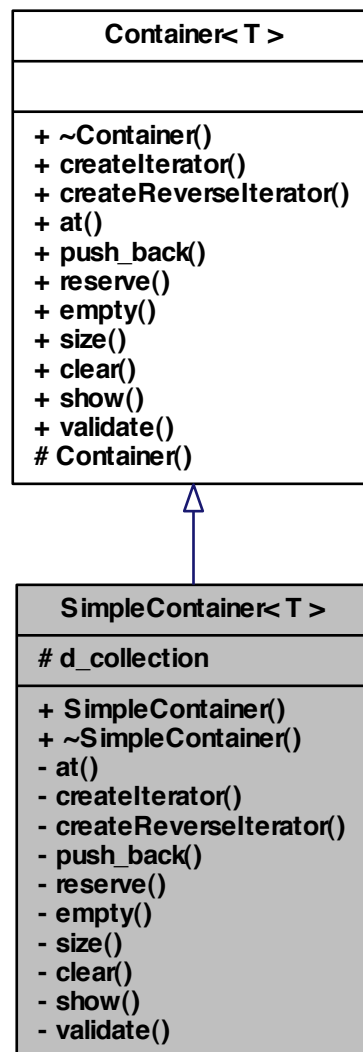
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ReciprocalFactory.h](#)

## 5.47 SimpleContainer< T > Class Template Reference

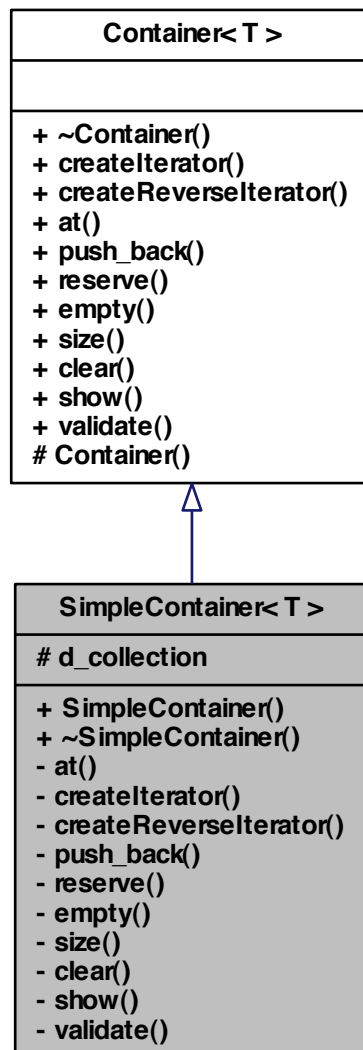
class [SimpleContainer](#) -

`#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 >](#)

#### 5.47.1 Detailed Description

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

`class` [SimpleContainer](#) -

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

#### 5.47.2 Constructor & Destructor Documentation

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

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

#### 5.47.3 Member Function Documentation

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

### 5.47.4 Friends And Related Function Documentation

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

Definition at line 13 of file SimpleContainer.h.

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

Definition at line 12 of file SimpleContainer.h.

### 5.47.5 Member Data Documentation

5.47.5.1 `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:

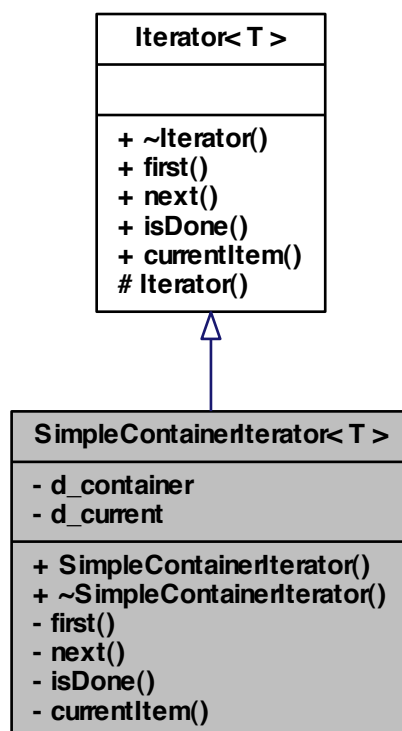
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.48 SimpleContainerIterator< T > Class Template Reference

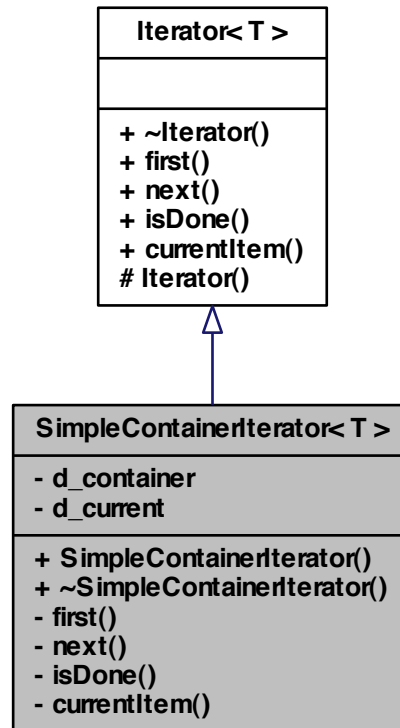
class [SimpleContainerIterator](#) -

```
#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 >](#)

### 5.48.1 Detailed Description

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

class [SimpleContainerIterator](#) -

Definition at line 6 of file SimpleContainerIterator.h.

### 5.48.2 Constructor & Destructor Documentation

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

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

### 5.48.3 Member Function Documentation

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

Implements [Iterator< T >](#).

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

Implements [Iterator< T >](#).

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

Implements [Iterator< T >](#).

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

Implements [Iterator< T>](#).

#### 5.48.4 Friends And Related Function Documentation

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

Definition at line 13 of file SimpleContainerIterator.h.

#### 5.48.5 Member Data Documentation

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

Definition at line 9 of file SimpleContainerIterator.h.

5.48.5.2 `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:

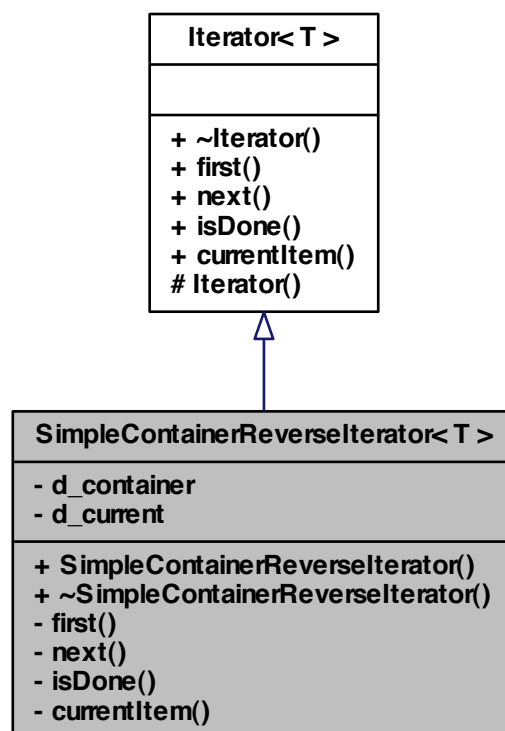
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

### 5.49 SimpleContainerReverselIterator< T> Class Template Reference

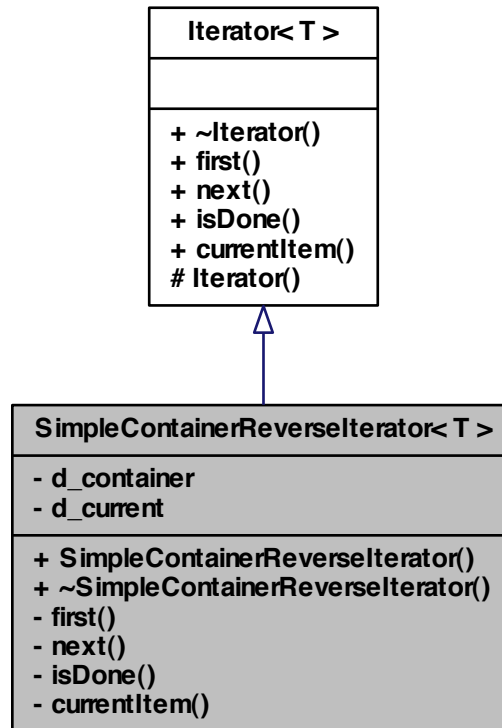
class [SimpleContainerReverselIterator](#) -

`#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 >](#)

### 5.49.1 Detailed Description

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

class [SimpleContainerReverseliterator](#) -

Definition at line 6 of file SimpleContainerReverseliterator.h.

### 5.49.2 Constructor & Destructor Documentation

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

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

### 5.49.3 Member Function Documentation

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

Implements [literator< T >](#).

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

Implements [literator< T >](#).

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

Implements [literator< T >](#).

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

Implements [Iterator< T >](#).

#### 5.49.4 Friends And Related Function Documentation

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

Definition at line 13 of file SimpleContainerReverseliterator.h.

#### 5.49.5 Member Data Documentation

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

Definition at line 9 of file SimpleContainerReverseliterator.h.

5.49.5.2 `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:

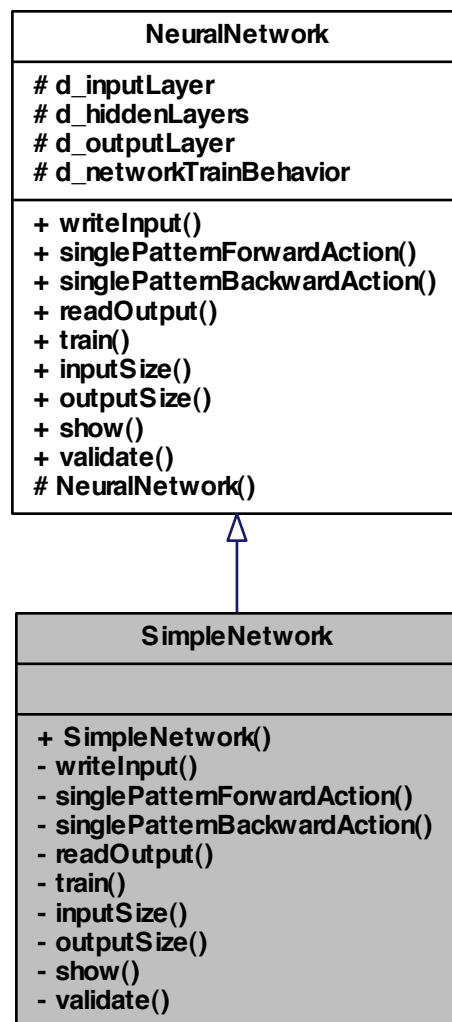
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.50 SimpleNetwork Class Reference

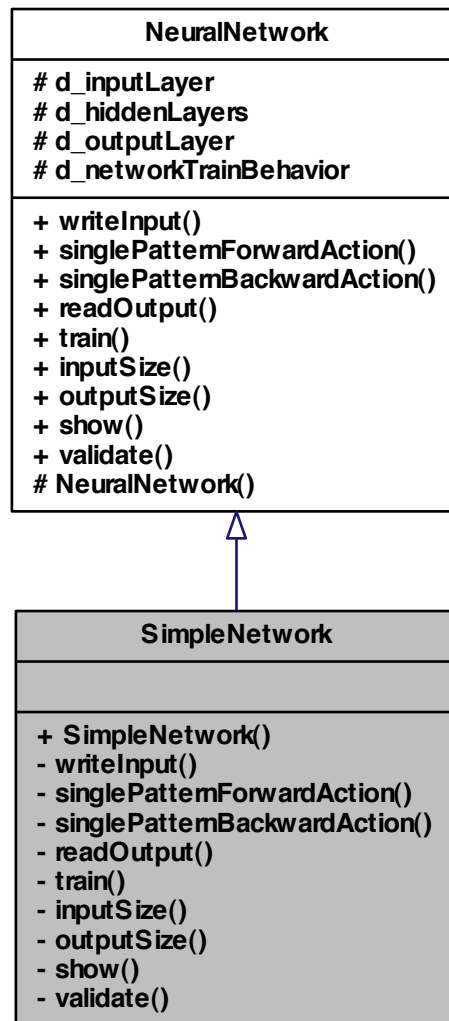
class [SimpleNetwork](#) -

```
#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](#) ()

#### 5.50.1 Detailed Description

class [SimpleNetwork](#) -

Definition at line 5 of file SimpleNetwork.h.

#### 5.50.2 Constructor & Destructor Documentation

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

Definition at line 16 of file SimpleNetwork.cpp.

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

#### 5.50.3 Member Function Documentation

##### 5.50.3.1 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:



#### 5.50.3.2 `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:



#### 5.50.3.3 `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:



#### 5.50.3.4 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=====\\n");
}

```

#### 5.50.3.5 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();
        }
    }
}
```

#### 5.50.3.6 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();
}
}
```

#### 5.50.3.7 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);
}
```

#### 5.50.3.8 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;
}
```

#### 5.50.3.9 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:

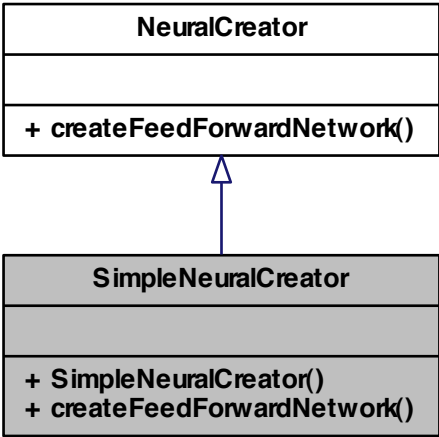


The documentation for this class was generated from the following files:

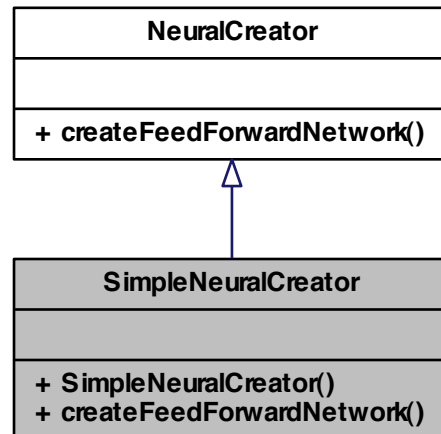
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[SimpleNetw](#)

5.51 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)

### 5.51.1 Detailed Description

class [SimpleNeuralCreator](#) -

Definition at line 5 of file SimpleNeuralCreator.h.

### 5.51.2 Constructor & Destructor Documentation

#### 5.51.2.1 SimpleNeuralCreator::SimpleNeuralCreator ( )

Definition at line 19 of file SimpleNeuralCreator.cpp.

```
{
}
```

### 5.51.3 Member Function Documentation

#### 5.51.3.1 `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);

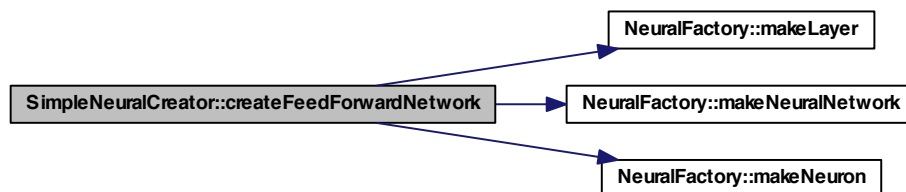
        }
    }

    //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:

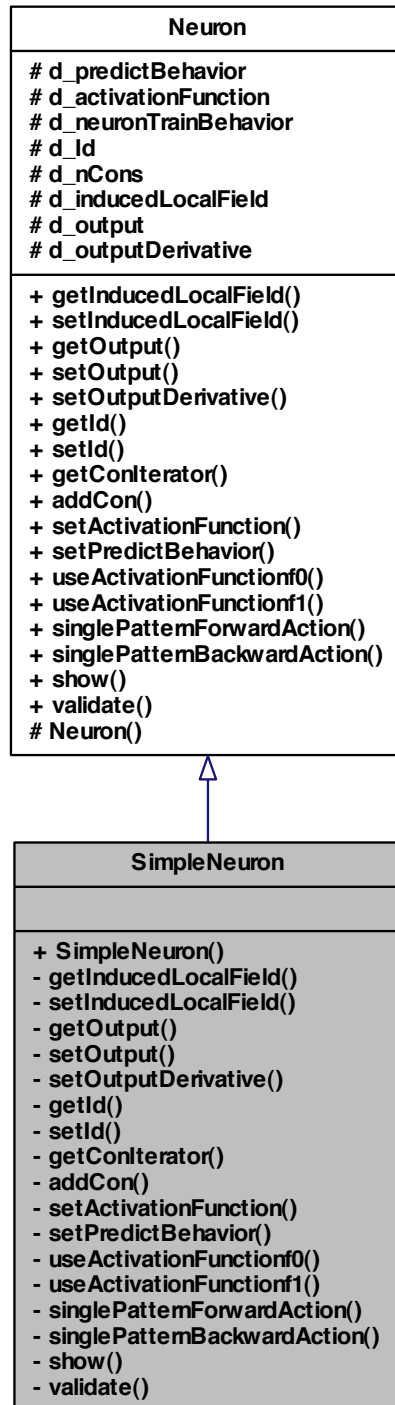
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNe`
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuralCreator.cp`

## 5.52 SimpleNeuron Class Reference

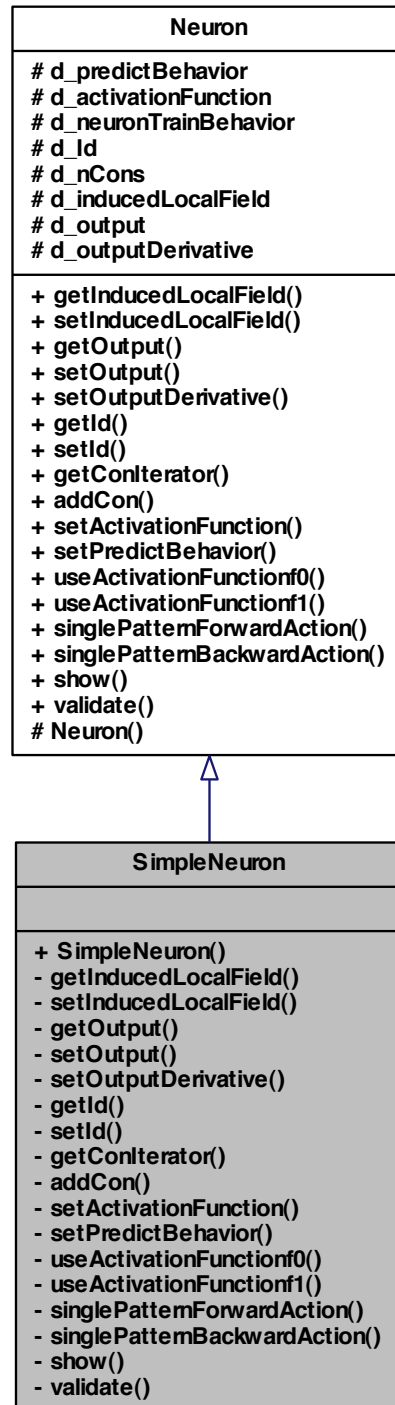
class [SimpleNeuron](#) -

```
#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](#) ()

### 5.52.1 Detailed Description

class [SimpleNeuron](#) -

Definition at line 5 of file SimpleNeuron.h.

### 5.52.2 Constructor & Destructor Documentation

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

Definition at line 18 of file SimpleNeuron.cpp.

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

### 5.52.3 Member Function Documentation

#### 5.52.3.1 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);  
}
```

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

Implements [Neuron](#).

Definition at line 68 of file SimpleNeuron.cpp.

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

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

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

Implements [Neuron](#).

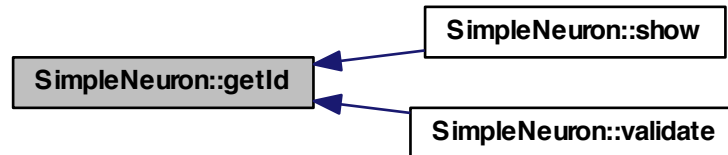
Definition at line 56 of file SimpleNeuron.cpp.

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

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

```
{  
    return d_Id;  
}
```

Here is the caller graph for this function:



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

Implements [Neuron](#).

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

References `Neuron::d_inducedLocalField`.

```
{  
    return d_inducedLocalField;  
}
```

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

Implements [Neuron](#).

Definition at line 36 of file `SimpleNeuron.cpp`.

References `Neuron::d_output`.

```
{  
    return d_output;  
}
```

#### 5.52.3.6 `void SimpleNeuron::setActivationFunction ( ActivationFunctionPtr activationFunctionPtr ) [private, virtual]`

Implements [Neuron](#).

Definition at line 80 of file `SimpleNeuron.cpp`.

References `Neuron::d_activationFunction`.

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

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

Implements [Neuron](#).

Definition at line 62 of file SimpleNeuron.cpp.

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

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

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

Implements [Neuron](#).

Definition at line 30 of file SimpleNeuron.cpp.

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

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

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

Implements [Neuron](#).

Definition at line 42 of file SimpleNeuron.cpp.

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

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

**5.52.3.10** void SimpleNeuron::setOutputDerivative ( double *outputDerivative* ) [private,  
virtual]

Implements [Neuron](#).

Definition at line 50 of file SimpleNeuron.cpp.

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



```
{
    d_outputDerivative = outputDerivative;
}
```

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

Implements [Neuron](#).

Definition at line 86 of file SimpleNeuron.cpp.

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

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

**5.52.3.12** `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:



**5.52.3.13** void SimpleNeuron::singlePatternBackwardAction ( ) [private, virtual]

Implements [Neuron](#).

Definition at line 114 of file SimpleNeuron.cpp.

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

```

{
    d_neuronTrainBehavior->singlePatternBackwardAction();
}

```

**5.52.3.14** void SimpleNeuron::singlePatternForwardAction ( ) [private, virtual]

Implements [Neuron](#).

Definition at line 108 of file SimpleNeuron.cpp.

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

```

{
    d_predictBehavior->singlePatternForwardAction();
}

```

**5.52.3.15** double SimpleNeuron::useActivationFunction0 ( ) [private, virtual]

Implements [Neuron](#).

Definition at line 92 of file SimpleNeuron.cpp.

References `Neuron::d_activationFunction`.

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

#### 5.52.3.16 `double SimpleNeuron::useActivationFunctionf1 ( )` [private, virtual]

Implements [Neuron](#).

Definition at line 100 of file SimpleNeuron.cpp.

References `Neuron::d_activationFunction`.

```
{  
    return d_activationFunction->f1();  
}
```

#### 5.52.3.17 `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:

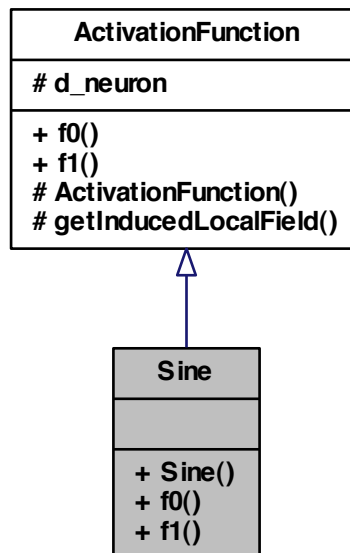
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuron.h`
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuron.cpp`

### 5.53 Sine Class Reference

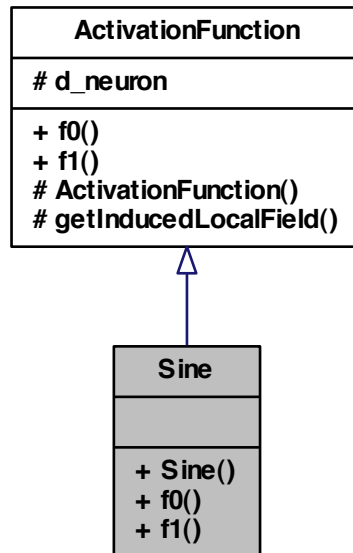
class [Sine](#) -

```
#include <Sine.h>
```

Inheritance diagram for Sine:



Collaboration diagram for Sine:



### Public Member Functions

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

### 5.53.1 Detailed Description

class [Sine](#) -

Definition at line 5 of file Sine.h.

### 5.53.2 Constructor & Destructor Documentation

5.53.2.1 [Sine::Sine](#) ( [NeuronPtr](#) neuronPtr )

### 5.53.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

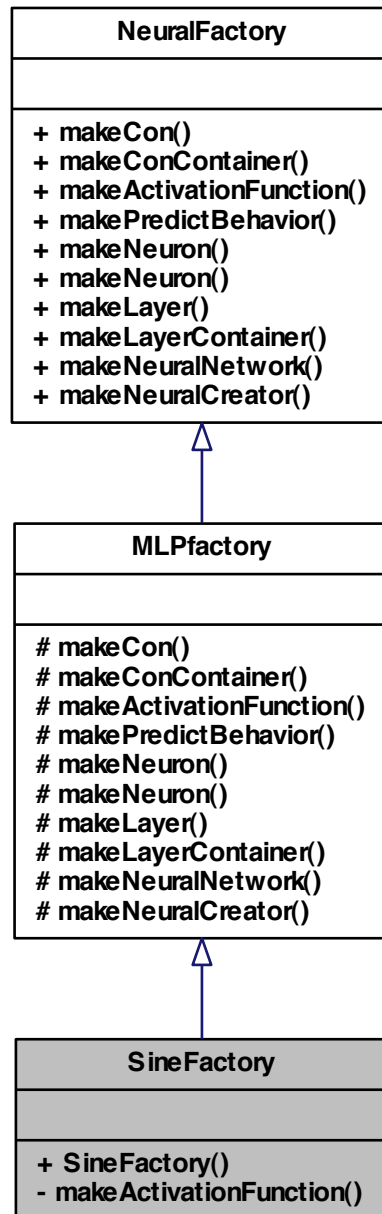
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.54 SineFactory Class Reference

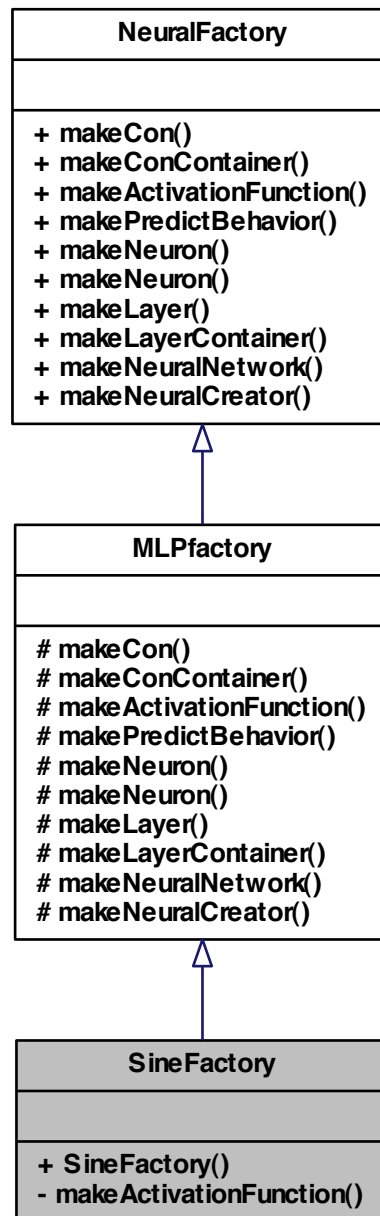
class [SineFactory](#) -

```
#include <SineFactory.h>
```

Inheritance diagram for SineFactory:



Collaboration diagram for SineFactory:





## Public Member Functions

- [SineFactory](#) ()

## Private Member Functions

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

### 5.54.1 Detailed Description

class [SineFactory](#) -

Definition at line 5 of file SineFactory.h.

### 5.54.2 Constructor & Destructor Documentation

5.54.2.1 [SineFactory::SineFactory](#) ( )

### 5.54.3 Member Function Documentation

5.54.3.1 [ActivationFunctionPtr](#) [SineFactory::makeActivationFunction](#) ( [NeuronPtr](#) *neuronPtr* ) [[private](#), [virtual](#)]

Implements [MLPfactory](#).

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

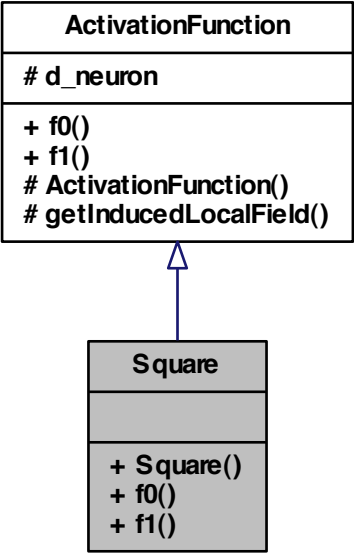
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[SineFactory.h](#)

## 5.55 Square Class Reference

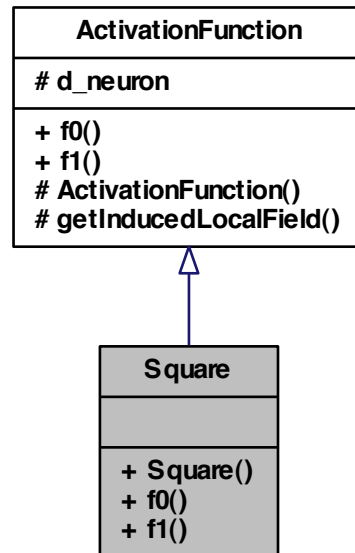
class [Square](#) -

```
#include <Square.h>
```

Inheritance diagram for Square:



Collaboration diagram for Square:



### Public Member Functions

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

### 5.55.1 Detailed Description

class [Square](#) -

Definition at line 5 of file Square.h.

### 5.55.2 Constructor & Destructor Documentation

5.55.2.1 [Square::Square](#) ( [NeuronPtr](#) neuronPtr )

### 5.55.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

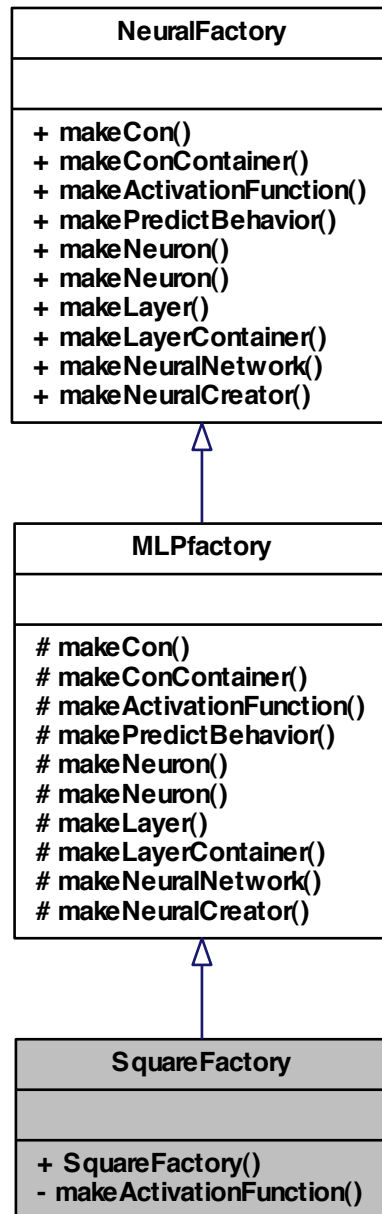
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

## 5.56 SquareFactory Class Reference

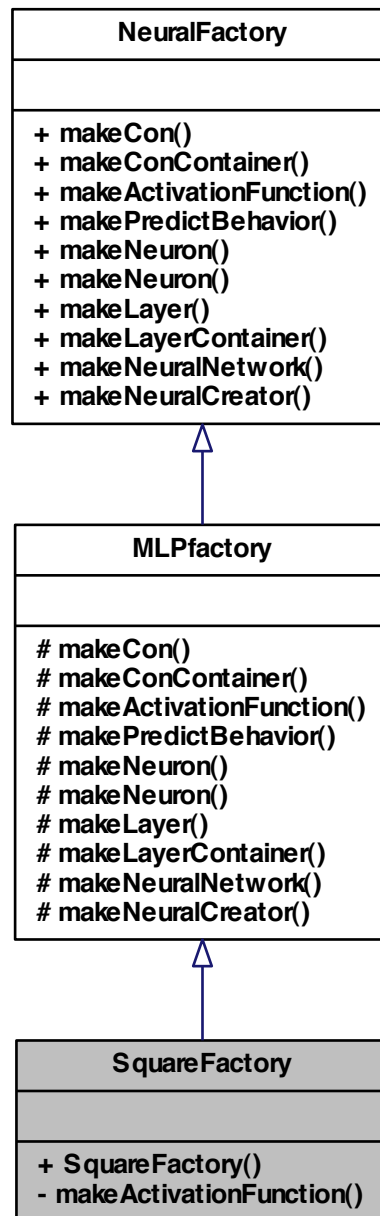
class [SquareFactory](#) -

```
#include <SquareFactory.h>
```

Inheritance diagram for SquareFactory:



Collaboration diagram for SquareFactory:



## Public Member Functions

- [SquareFactory](#) ()

## Private Member Functions

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

### 5.56.1 Detailed Description

class [SquareFactory](#) -

Definition at line 5 of file SquareFactory.h.

### 5.56.2 Constructor & Destructor Documentation

5.56.2.1 [SquareFactory::SquareFactory](#) ( )

### 5.56.3 Member Function Documentation

5.56.3.1 [ActivationFunctionPtr](#) [SquareFactory::makeActivationFunction](#) ( [NeuronPtr](#) *neuronPtr* ) [*private*, *virtual*]

Implements [MLPfactory](#).

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

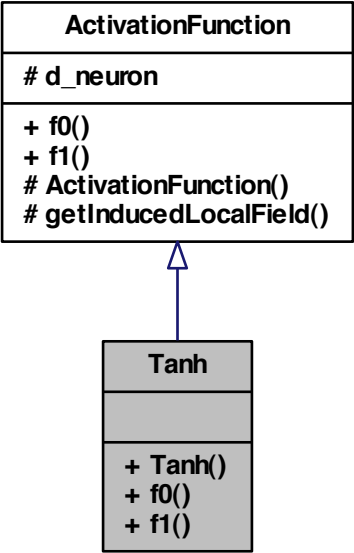
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[SquareFa](#)

## 5.57 Tanh Class Reference

class [Tanh](#) -

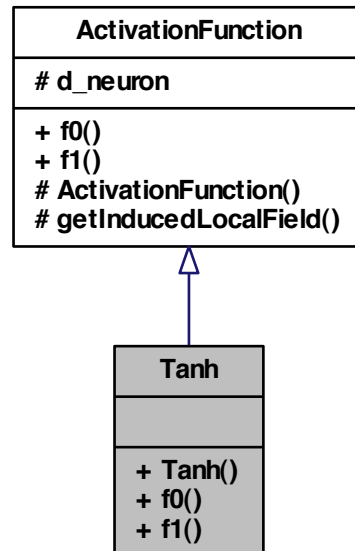
```
#include <Tanh.h>
```

Inheritance diagram for Tanh:





Collaboration diagram for Tanh:



### Public Member Functions

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

#### 5.57.1 Detailed Description

class [Tanh](#) -

Definition at line 5 of file Tanh.h.

#### 5.57.2 Constructor & Destructor Documentation

##### 5.57.2.1 Tanh::Tanh ( [NeuronPtr](#) neuronPtr )

Definition at line 15 of file Tanh.cpp.

```

: ActivationFunction(neuronPtr) {

```

```
}
```

### 5.57.3 Member Function Documentation

#### 5.57.3.1 `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:



#### 5.57.3.2 `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:

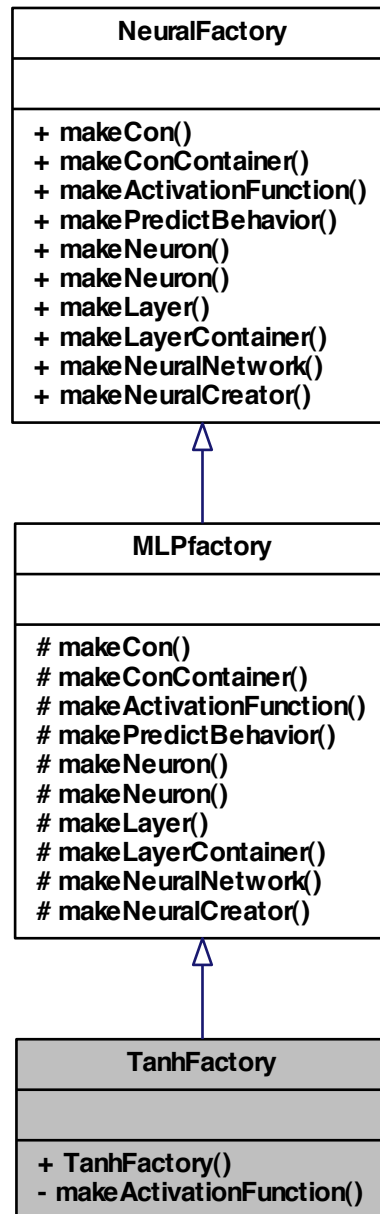
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[Tanh.h](#)
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[Tanh.cpp](#)

## 5.58 TanhFactory Class Reference

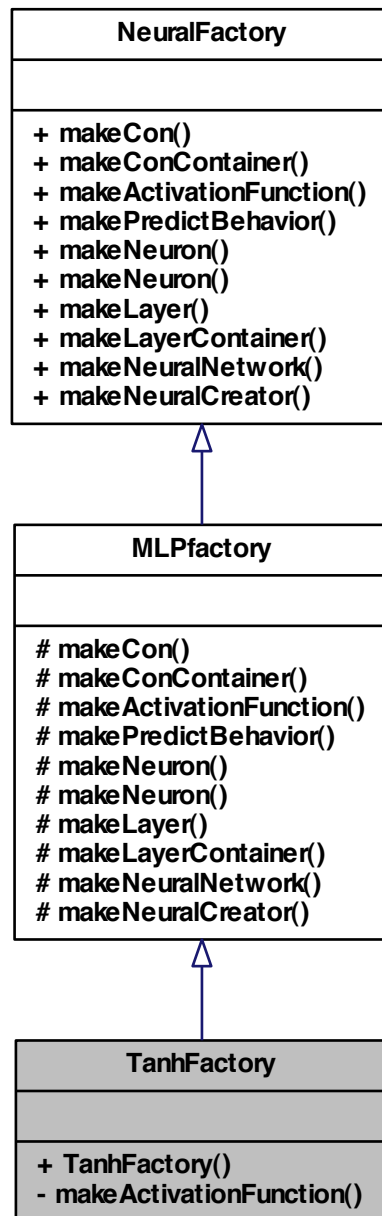
class [TanhFactory](#) -

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

Inheritance diagram for TanhFactory:



Collaboration diagram for TanhFactory:



## Public Member Functions

- [TanhFactory](#) ()

## Private Member Functions

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

### 5.58.1 Detailed Description

class [TanhFactory](#) -

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

### 5.58.2 Constructor & Destructor Documentation

#### 5.58.2.1 [TanhFactory::TanhFactory](#) ( )

Definition at line 17 of file [TanhFactory.cpp](#).

```
{
}
```

### 5.58.3 Member Function Documentation

#### 5.58.3.1 [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:

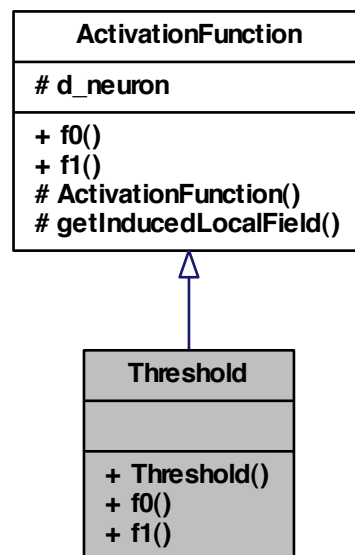
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead](#)
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/\[TanhFactor\]\(#\)](#)

## 5.59 Threshold Class Reference

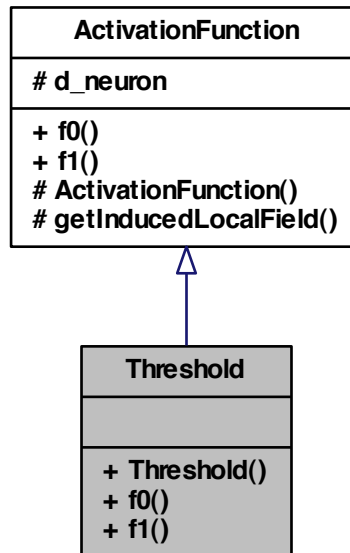
class [Threshold](#) -

```
#include <Threshold.h>
```

Inheritance diagram for Threshold:



Collaboration diagram for Threshold:



## Public Member Functions

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

### 5.59.1 Detailed Description

class [Threshold](#) -

Definition at line 5 of file Threshold.h.

### 5.59.2 Constructor & Destructor Documentation

5.59.2.1 [Threshold::Threshold](#) ( [NeuronPtr](#) neuronPtr )

### 5.59.3 Member Function Documentation



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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

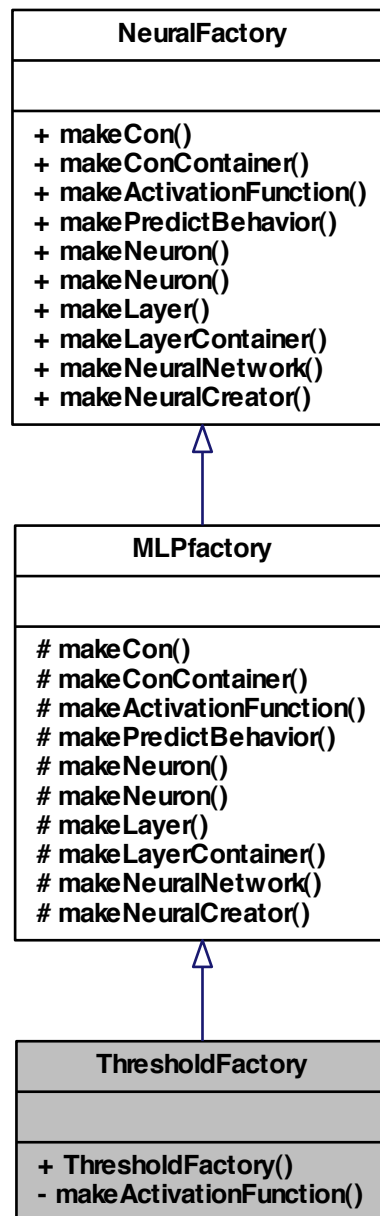
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Threshold`

## 5.60 ThresholdFactory Class Reference

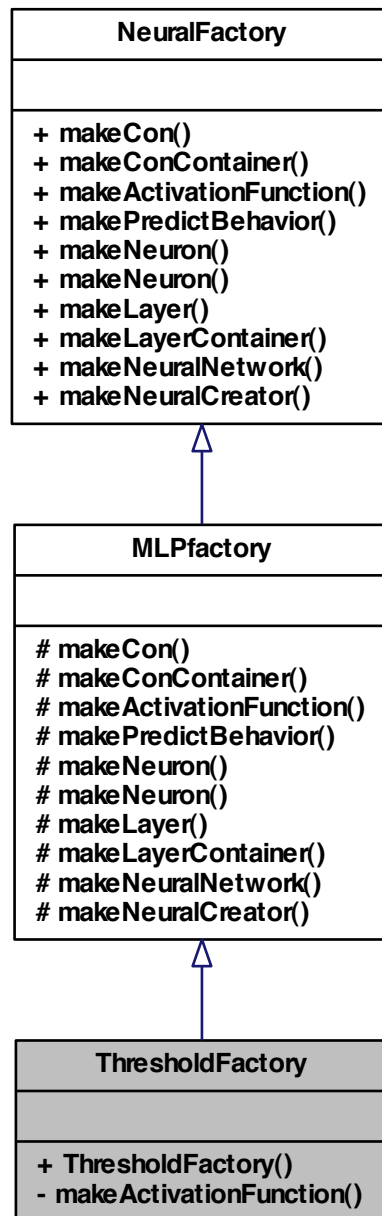
class [ThresholdFactory](#) -

```
#include <ThresholdFactory.h>
```

Inheritance diagram for ThresholdFactory:



Collaboration diagram for ThresholdFactory:



### Public Member Functions

- [ThresholdFactory](#) ()

### Private Member Functions

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

#### 5.60.1 Detailed Description

class [ThresholdFactory](#) -

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

#### 5.60.2 Constructor & Destructor Documentation

##### 5.60.2.1 [ThresholdFactory::ThresholdFactory](#) ( )

#### 5.60.3 Member Function Documentation

##### 5.60.3.1 [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](#)

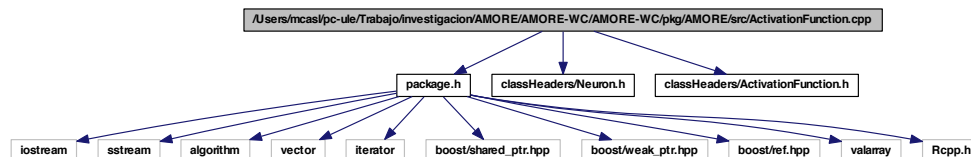
## Chapter 6

# File Documentation

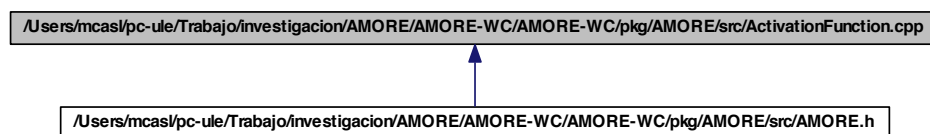
### 6.1 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/ActivationFunction.cpp File Reference

```
#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](#) -*

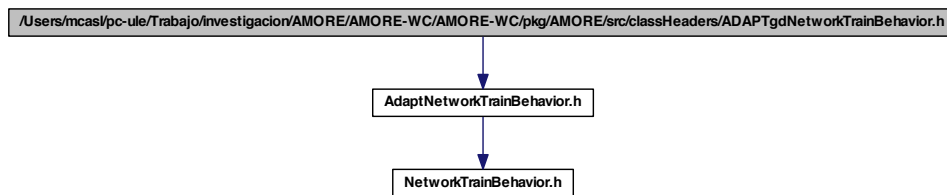
6.5 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdNetworkTrainBehavior.h File

Reference

6.5 — 225 /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

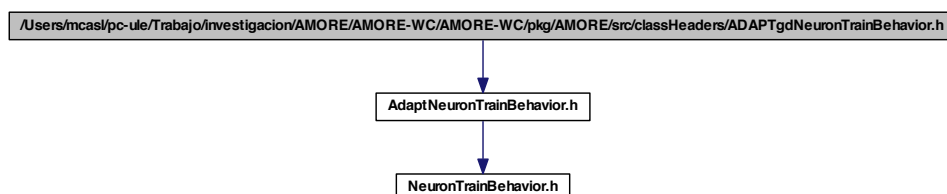
- class [ADAPTgdNetworkTrainBehavior](#)

*class [ADAPTgdNetworkTrainBehavior](#) -*

6.6 /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:



## Classes

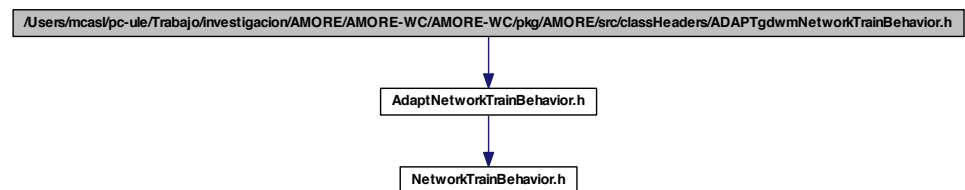
- class [ADAPTgdNeuronTrainBehavior](#)

*class [ADAPTgdNeuronTrainBehavior](#) -*

## 6.7 /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](#) -*

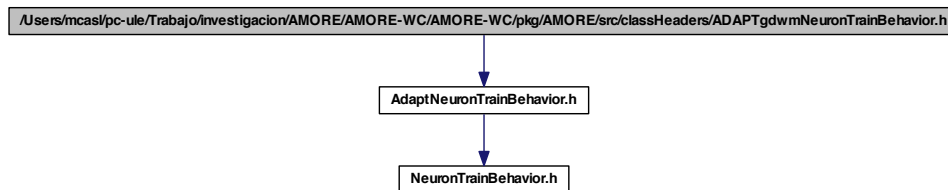
## 6.8 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwmNeuronTrainBehavior.h File Reference

```
#include "AdaptNeuronTrainBehavior.h"
```

## 6.9 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptNetworkTrainBehavior.h File Reference

227

Include dependency graph for ADAPTgdwmNeuronTrainBehavior.h:



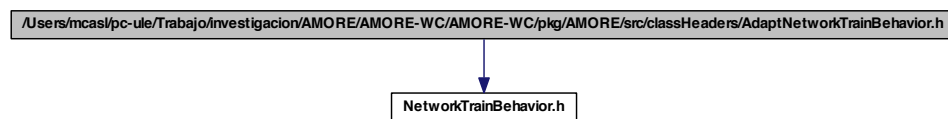
### Classes

- class [ADAPTgdwmNeuronTrainBehavior](#)  
*class ADAPTgdwmNeuronTrainBehavior -*

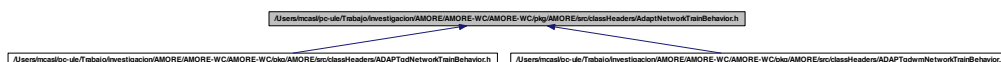
## 6.9 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptNetworkTrainBehavior.h File Reference

```
#include "NetworkTrainBehavior.h"
```

Include dependency graph for AdaptNetworkTrainBehavior.h:



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



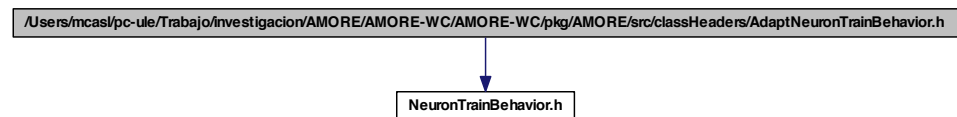
## Classes

- class [AdaptNetworkTrainBehavior](#)  
class [AdaptNetworkTrainBehavior](#) -

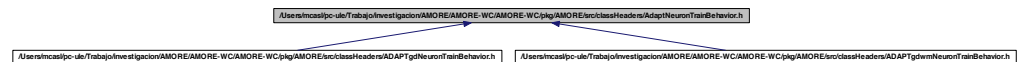
## 6.10 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptNeuronTrainBehavior.h File Reference

```
#include "NeuronTrainBehavior.h"
```

Include dependency graph for AdaptNeuronTrainBehavior.h:



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



## Classes

- class [AdaptNeuronTrainBehavior](#)  
class [AdaptNeuronTrainBehavior](#) -

## 6.11 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ArcTan.h File Reference

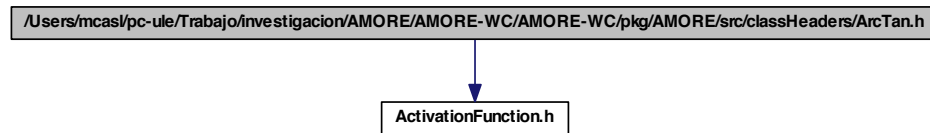
```
#include "ActivationFunction.h"
```

## 6.12 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ArcTanFactory.h File

### Reference

229

Include dependency graph for ArcTan.h:



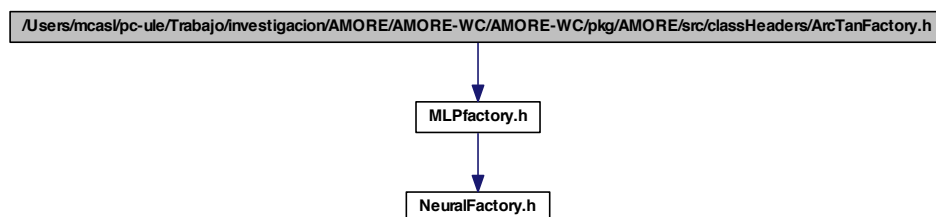
### Classes

- class [ArcTan](#)  
*class [ArcTan](#) -*

## 6.12 /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:



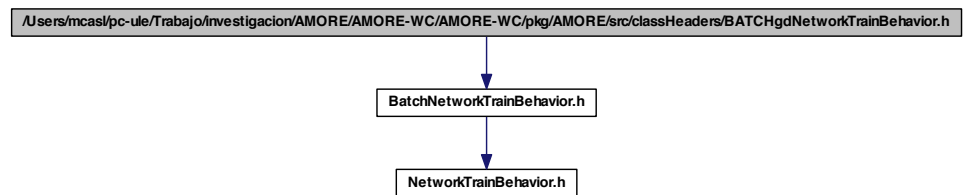
### Classes

- class [ArcTanFactory](#)  
*class [ArcTanFactory](#) -*

### 6.13 /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:



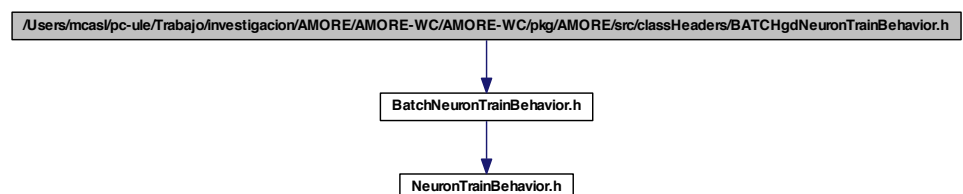
#### Classes

- class [BATCHgdNetworkTrainBehavior](#)  
*class [BATCHgdNetworkTrainBehavior](#) -*

### 6.14 /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:





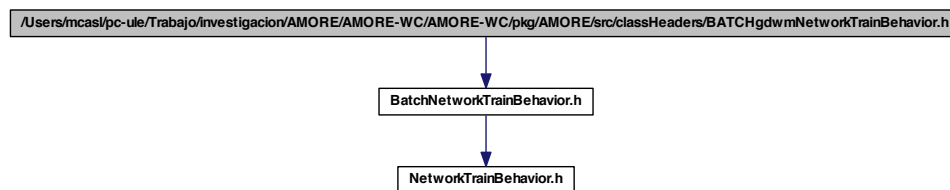
- class [BATCHgdNeuronTrainBehavior](#)

*class BATCHgdNeuronTrainBehavior -*

## 6.15 /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

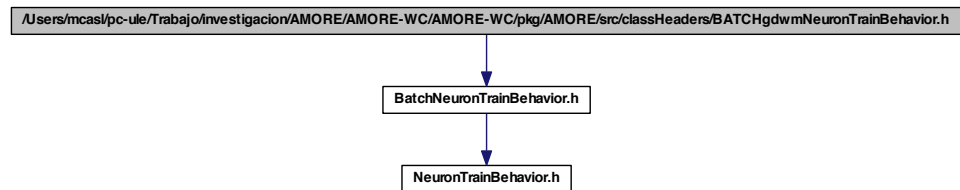
- class [BATCHgdwmNetworkTrainBehavior](#)

*class BATCHgdwmNetworkTrainBehavior -*

## 6.16 /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:



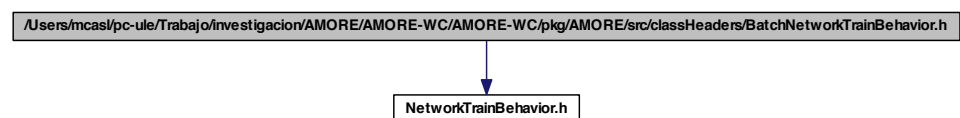
## Classes

- class [BATCHgdwmNeuronTrainBehavior](#)  
*class [BATCHgdwmNeuronTrainBehavior](#) -*

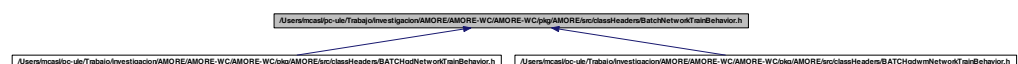
## 6.17 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BatchNetworkTrainBehavior.h File Reference

```
#include "NetworkTrainBehavior.h"
```

Include dependency graph for BatchNetworkTrainBehavior.h:



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



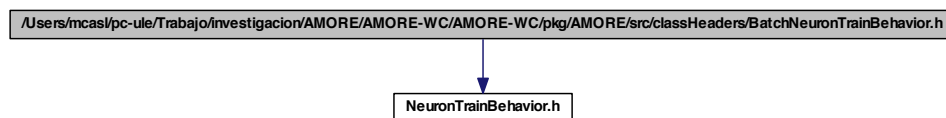
- class [BatchNetworkTrainBehavior](#)

*class [BatchNetworkTrainBehavior](#) -*

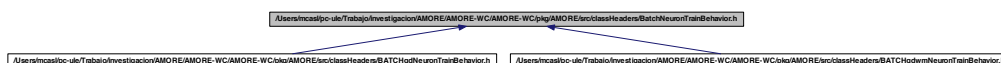
## 6.18 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BatchNeuronTrainBehavior.h File Reference

```
#include "NeuronTrainBehavior.h"
```

Include dependency graph for BatchNeuronTrainBehavior.h:



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



## Classes

- class [BatchNeuronTrainBehavior](#)

*class [BatchNeuronTrainBehavior](#) -*

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



- Include dependency graph for Container.h:

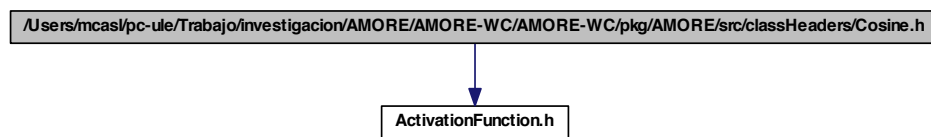


- Generated on Sat Jul 30 2011 04:41:16 for AMORE++ by Doxygen

## 6.21 /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:



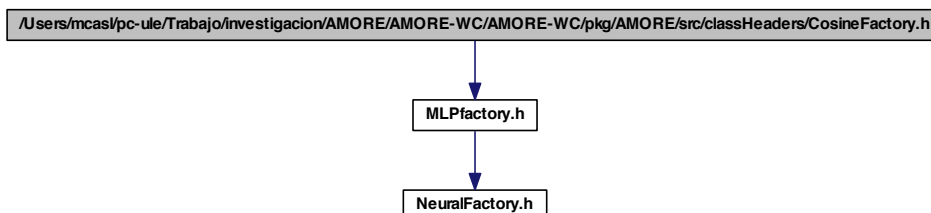
### Classes

- class [Cosine](#)  
class [Cosine](#) -

## 6.22 /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

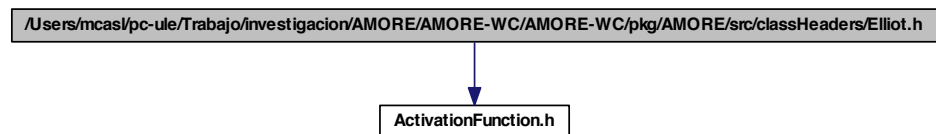
- class [CosineFactory](#)

*class [CosineFactory](#) -*

## 6.23 [/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](#) -*

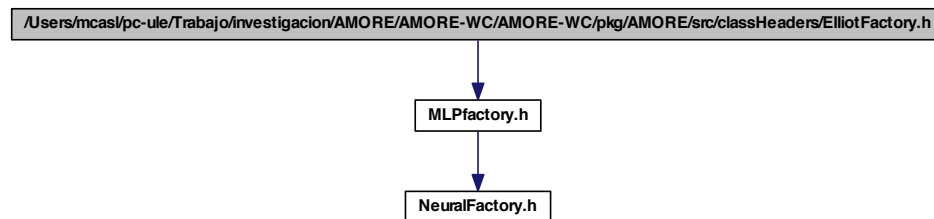
## 6.24 [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ElliotFactory.h](#) File Reference

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

## 6.25 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Exponential.h File Reference

237

Include dependency graph for ElliotFactory.h:



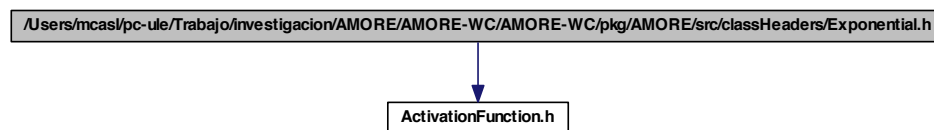
### Classes

- class [ElliotFactory](#)  
*class [ElliotFactory](#) -*

## 6.25 /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:



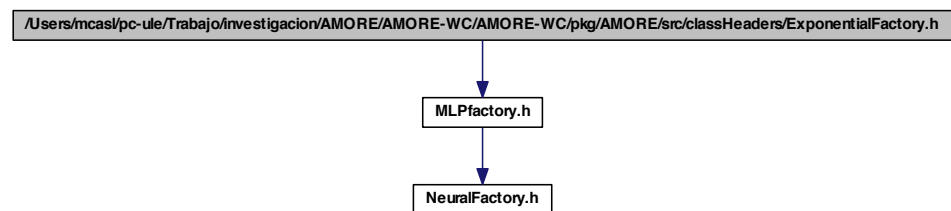
### Classes

- class [Exponential](#)  
*class [Exponential](#) -*

## 6.26 /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:



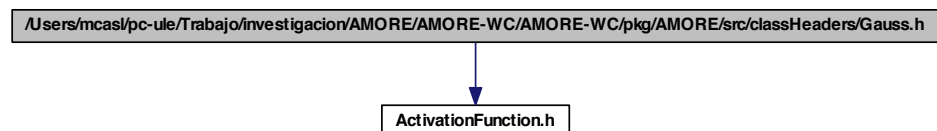
### Classes

- class [ExponentialFactory](#)  
class *ExponentialFactory* -

## 6.27 /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](#)

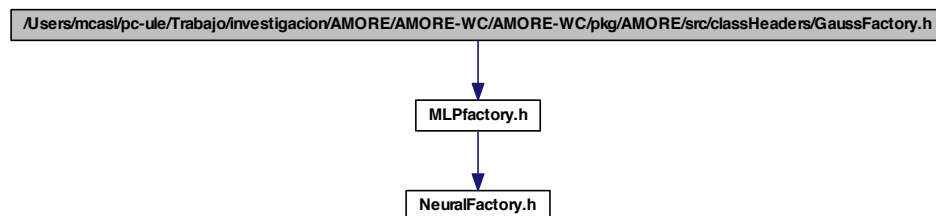


*class* [Gauss](#) -

## 6.28 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/GaussFactory.h File Reference

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

Include dependency graph for GaussFactory.h:



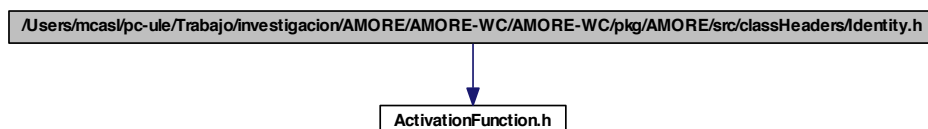
### Classes

- class [GaussFactory](#)  
*class* [GaussFactory](#) -

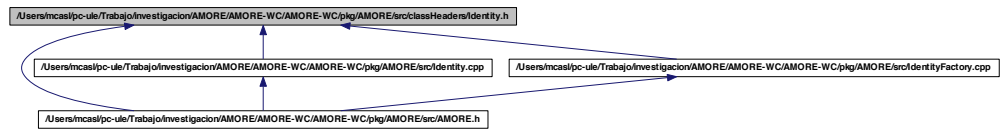
## 6.29 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Identity.h File Reference

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

Include dependency graph for Identity.h:



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



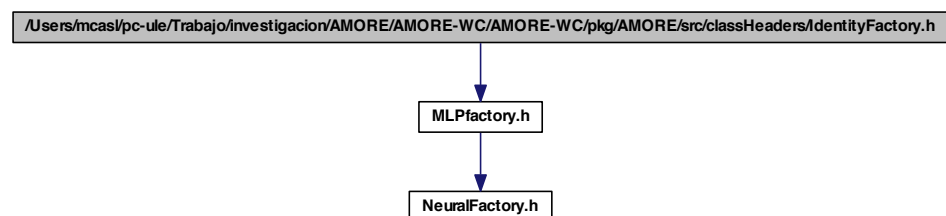
## Classes

- class [Identity](#)  
class *Identity* -

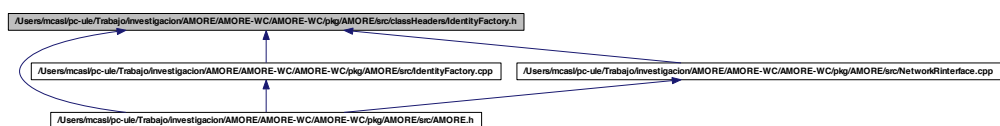
## 6.30 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/IdentityFactory.h File Reference

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

Include dependency graph for IdentityFactory.h:



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



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

#### Reference Classes

241

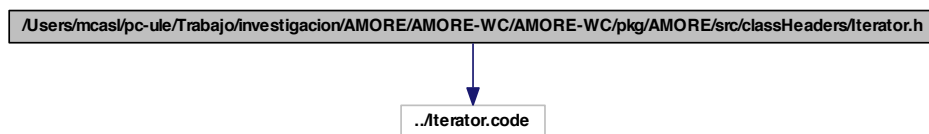
- class [IdentityFactory](#)

*class IdentityFactory -*

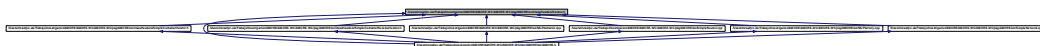
### 6.31 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Iterator.h File Reference

```
#include "../Iterator.code"
```

Include dependency graph for Iterator.h:



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



#### Classes

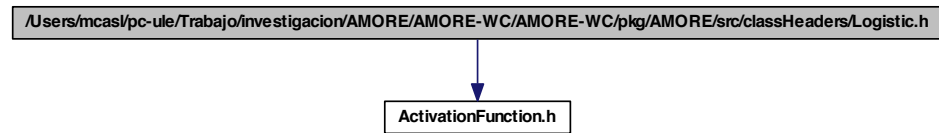
- class [Iterator< T >](#)

*class Iterator -*

### 6.32 /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:



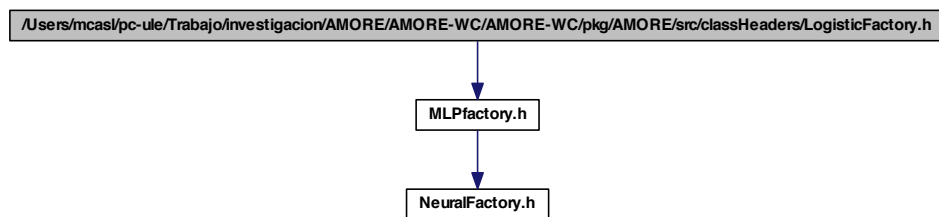
## Classes

- class [Logistic](#)  
*class [Logistic](#) -*

## 6.33 /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](#) -*

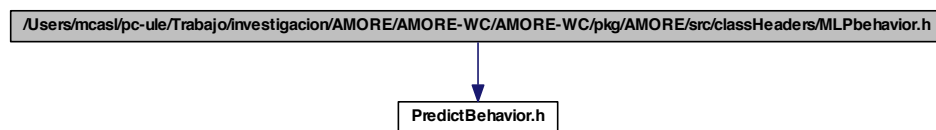
### 6.34 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPbehavior.h File Reference

Reference

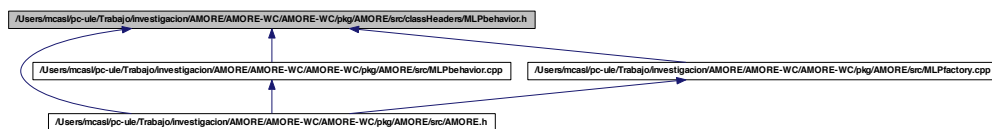
### 6.34 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPbehavior.h File Reference

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

Include dependency graph for MLPbehavior.h:



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



## Classes

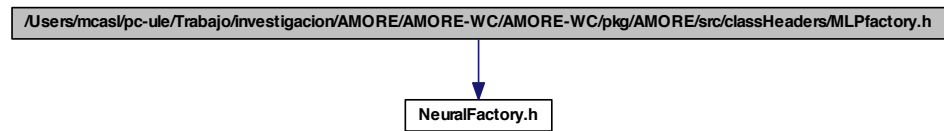
- class [MLPbehavior](#)

*class [MLPbehavior](#) -*

### 6.35 /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:

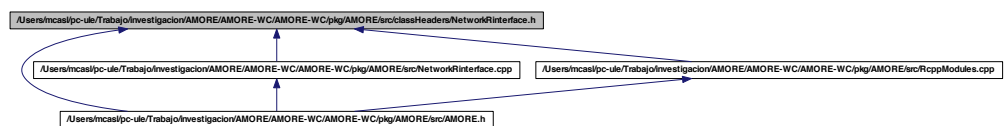


## Classes

- class [MLPfactory](#)  
class [MLPfactory](#) -

## 6.36 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkRinterface.h File Reference

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



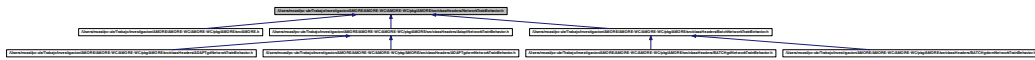
## Classes

- class [NetworkRinterface](#)  
class [NetworkRinterface](#) -

### 6.37 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkTrainBehavior.h File Reference

### 6.37 — 245 /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:

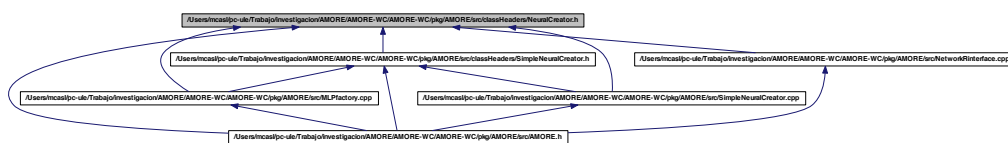


#### Classes

- class [NetworkTrainBehavior](#)  
*class [NetworkTrainBehavior](#) -*

### 6.38 /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](#) -*

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



- class `NeuralFactory`

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



- class `NeuralNetwork`

*class `NeuralNetwork` -*

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





## 6.42 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuronTrainBehavior.h File Reference

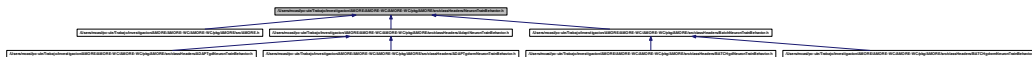
247

### Classes

- class [Neuron](#)  
*class [Neuron](#) -*

## 6.42 /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:

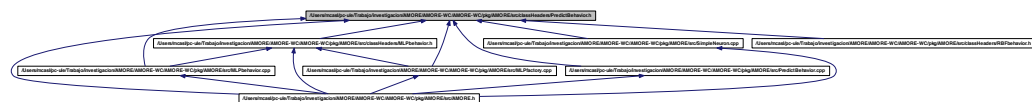


### Classes

- class [NeuronTrainBehavior](#)  
*class [NeuronTrainBehavior](#) -*

## 6.43 /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:



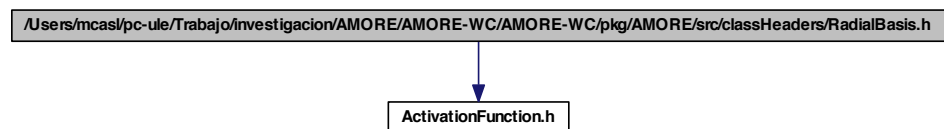
### Classes

- class [PredictBehavior](#)  
*class [PredictBehavior](#) -*

## 6.44 /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:



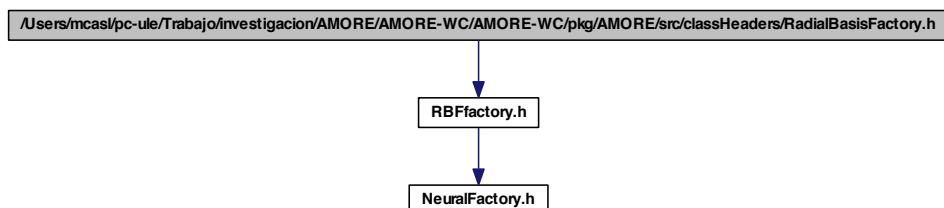
### Classes

- class [RadialBasis](#)  
*class [RadialBasis](#) -*

## 6.45 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasisFactory.h File Reference

```
#include "RBFFactory.h"
```

Include dependency graph for RadialBasisFactory.h:



### Classes

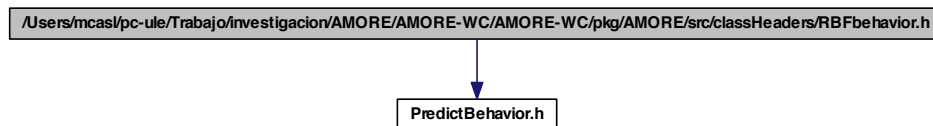
- class [RadialBasisFactory](#)

*class [RadialBasisFactory](#) -*

## 6.46 /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

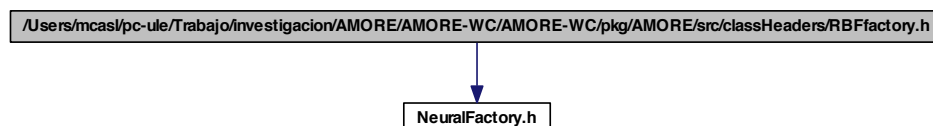
- class [RBFbehavior](#)

*class [RBFbehavior](#) -*

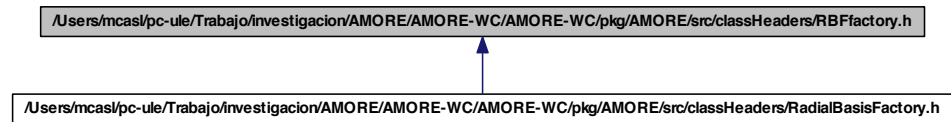
## 6.47 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFfactory.h File Reference

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

Include dependency graph for RBFfactory.h:



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



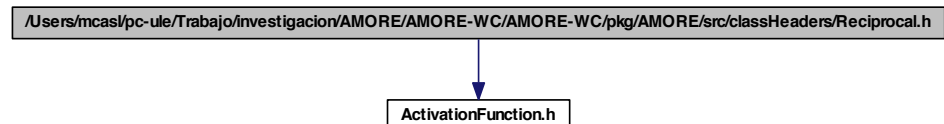
## Classes

- class [RBFfactory](#)  
*class [RBFfactory](#) -*

## 6.48 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Reciprocal.h File Reference

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

Include dependency graph for Reciprocal.h:



## Classes

- class [Reciprocal](#)  
*class [Reciprocal](#) -*

## 6.49 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ReciprocalFactory.h File Reference

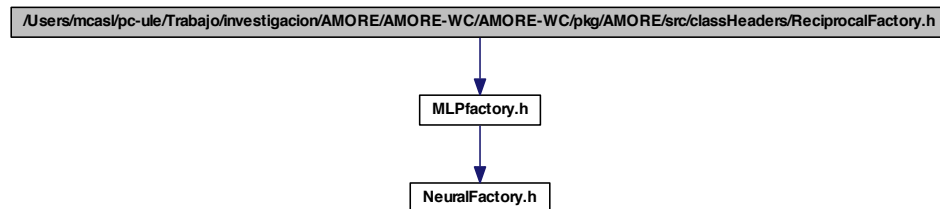
```
#include "MLPfactory.h"
```

## 6.50 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainer.h File

### Reference

251

Include dependency graph for ReciprocalFactory.h:



### Classes

- class [ReciprocalFactory](#)

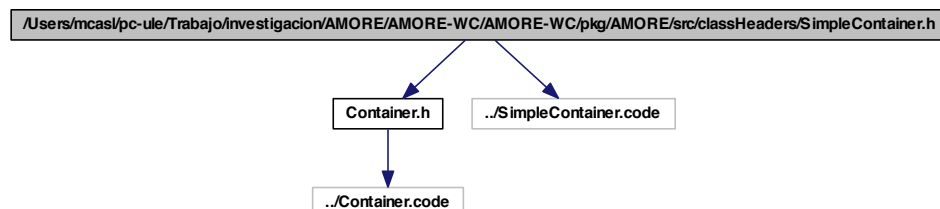
*class [ReciprocalFactory](#) -*

## 6.50 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainer.h File Reference

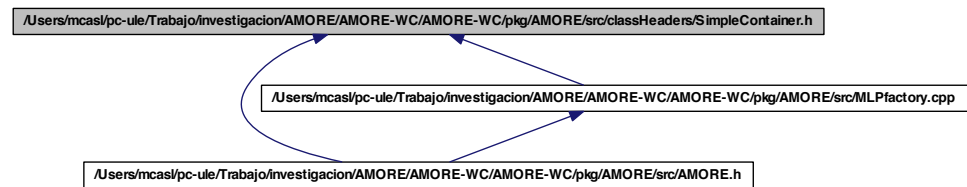
```
#include "Container.h"
```

```
#include "../SimpleContainer.code"
```

Include dependency graph for SimpleContainer.h:



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



## Classes

- class `SimpleContainer< T >`

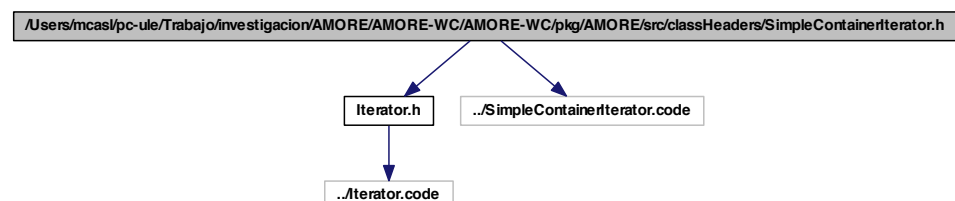
class `SimpleContainer` -

## 6.51 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainerIterator.h File Reference

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

```
#include "../SimpleContainerIterator.code"
```

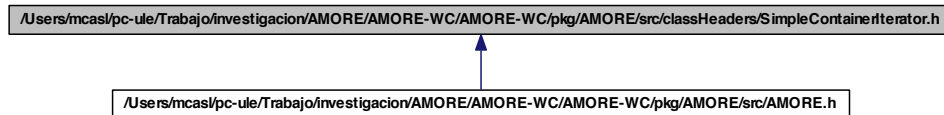
Include dependency graph for SimpleContainerIterator.h:



## 6.52 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainerReverseliterator.h File Reference

253

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



## Classes

- class `SimpleContainerIterator< T >`

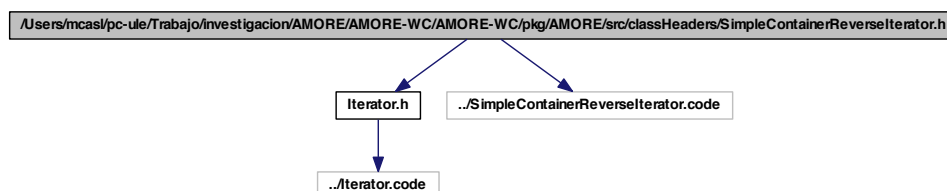
*class `SimpleContainerIterator` -*

## 6.52 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainerReverseliterator.h File Reference

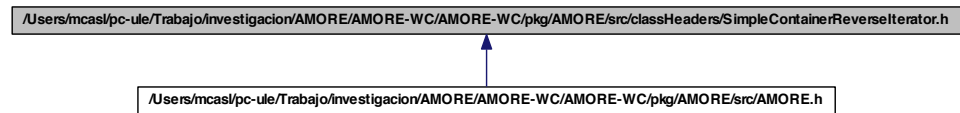
```
#include "Iterator.h"
```

```
#include "../SimpleContainerReverseIterator.code"
```

Include dependency graph for SimpleContainerReverseliterator.h:



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



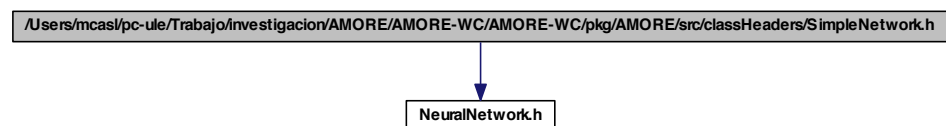
## Classes

- class `SimpleContainerReverseIterator< T >`  
*class `SimpleContainerReverseIterator` -*

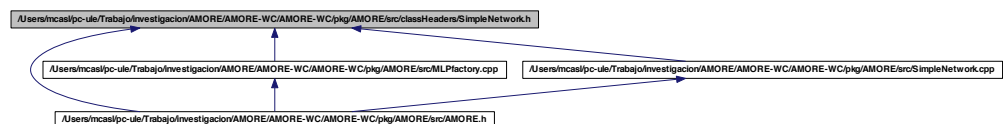
## 6.53 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNetwork.h File Reference

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

Include dependency graph for SimpleNetwork.h:



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





## 6.54 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuralCreator.h File Reference

255

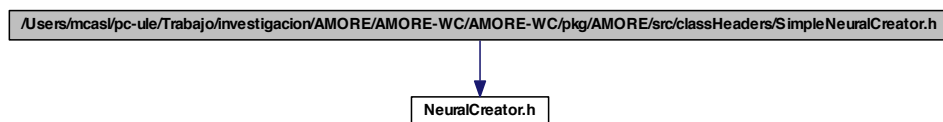
### Classes

- class [SimpleNetwork](#)  
*class SimpleNetwork -*

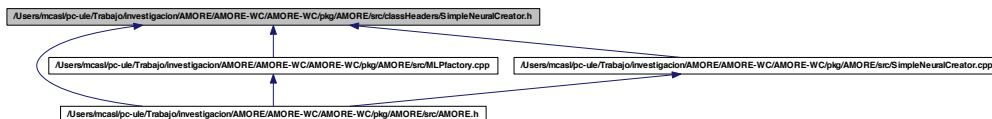
## 6.54 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuralCreator.h File Reference

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

Include dependency graph for SimpleNeuralCreator.h:



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



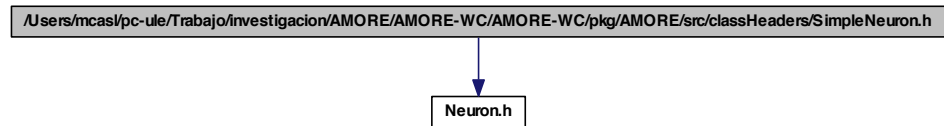
### Classes

- class [SimpleNeuralCreator](#)  
*class SimpleNeuralCreator -*

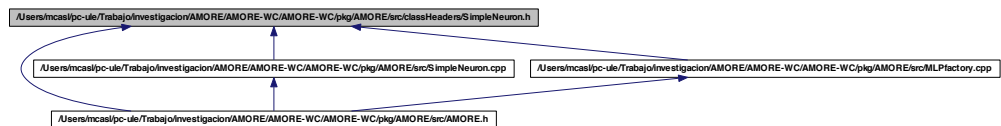
## 6.55 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuron.h File Reference

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

Include dependency graph for SimpleNeuron.h:



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



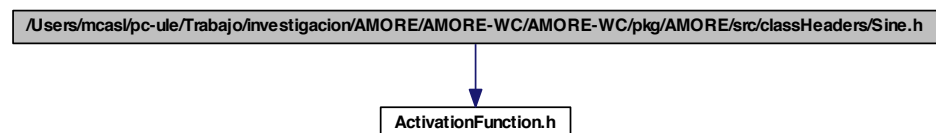
## Classes

- class [SimpleNeuron](#)  
*class [SimpleNeuron](#) -*

## 6.56 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Sine.h File Reference

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

Include dependency graph for Sine.h:



## 6.57 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SineFactory.h File Reference

257

### Reference Classes

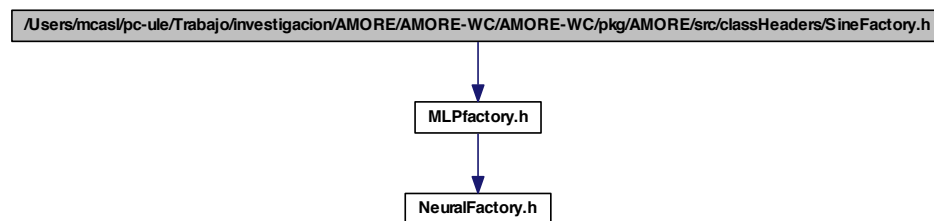
- class [Sine](#)

*class [Sine](#) -*

## 6.57 /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

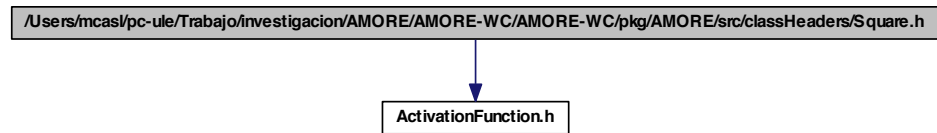
- class [SineFactory](#)

*class [SineFactory](#) -*

## 6.58 /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:



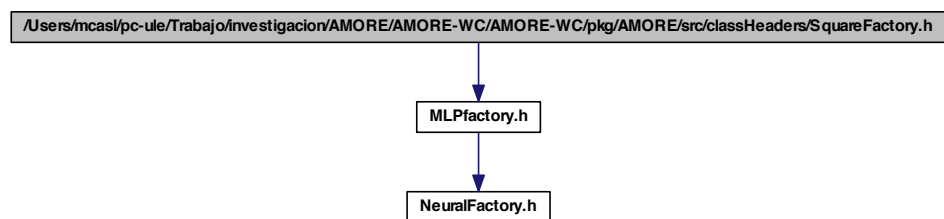
## Classes

- class [Square](#)  
*class [Square](#) -*

## 6.59 /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](#)  
*class [SquareFactory](#) -*

## 6.60 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Tanh.h File Reference

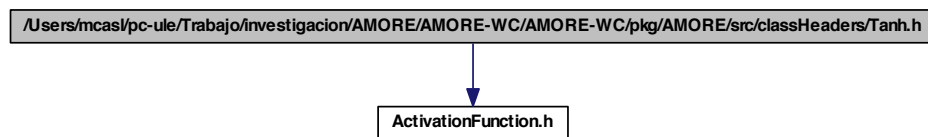
Reference

## 6.60 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Tanh.h File Reference

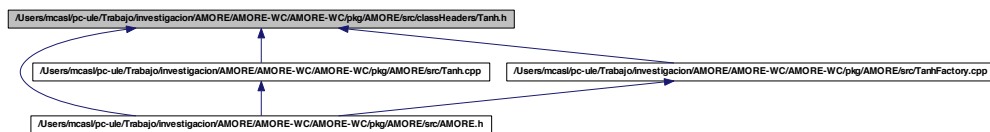
259

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

Include dependency graph for Tanh.h:



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



## Classes

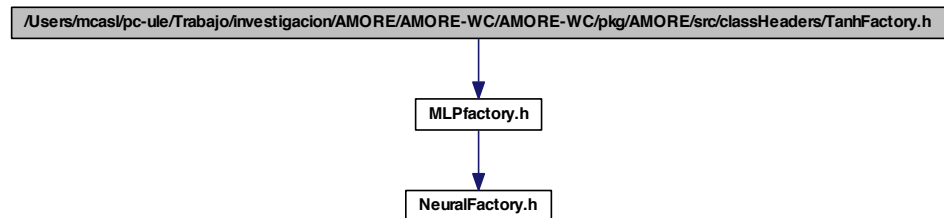
- class `Tanh`

*class `Tanh` -*

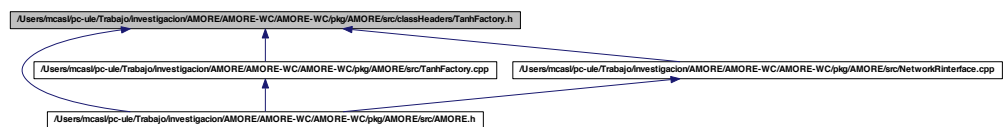
## 6.61 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/TanhFactory.h File Reference

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

Include dependency graph for TanhFactory.h:



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



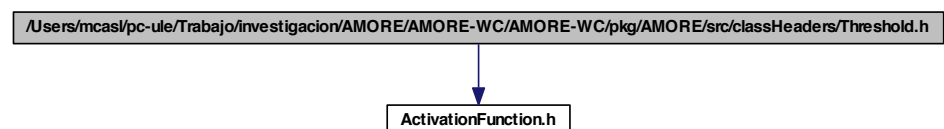
## Classes

- class [TanhFactory](#)  
class [TanhFactory](#) -

## 6.62 /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

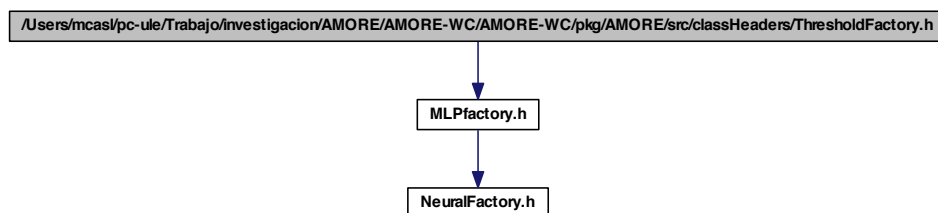
- class [Threshold](#)

*class [Threshold](#) -*

## 6.63 /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](#) -*

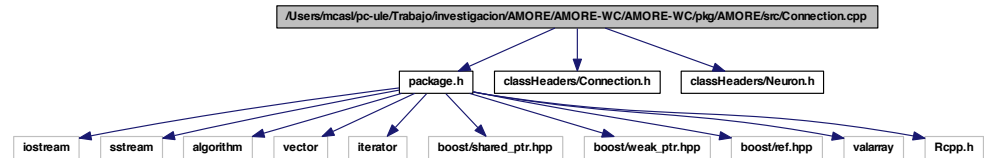
## 6.64 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Connection.cpp File Reference

```
#include "package.h"
```

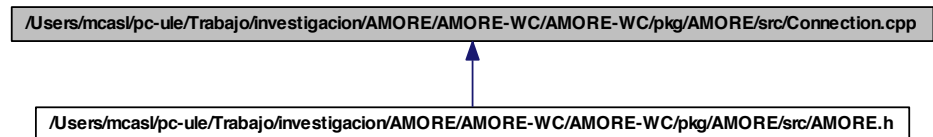
```
#include "classHeaders/Connection.h"
```

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

Include dependency graph for Connection.cpp:



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

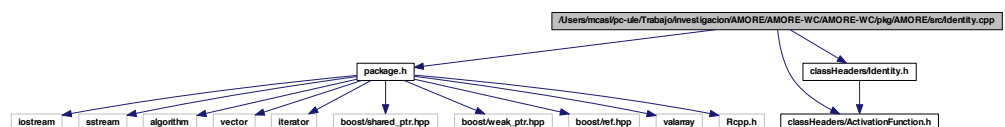


## 6.65 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Identity.cpp File Reference

```

#include "package.h"
#include "classHeaders/ActivationFunction.h"
#include "classHeaders/Identity.h"
  
```

Include dependency graph for Identity.cpp:

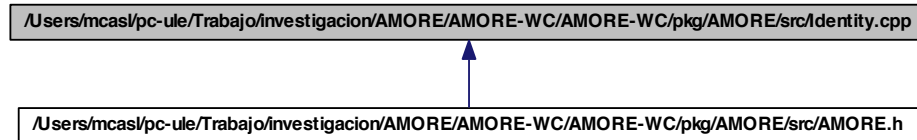




## 6.66 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/IdentityFactory.cpp File Reference

263

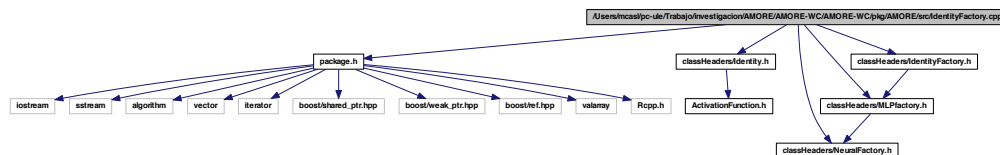
This graph shows which files directly or indirectly include this file:



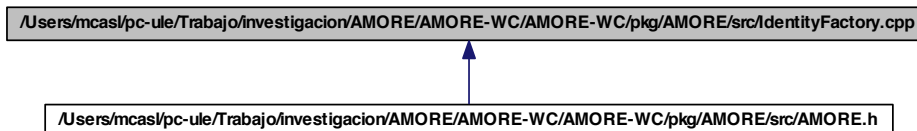
## 6.66 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/IdentityFactory.cpp File Reference

```
#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:



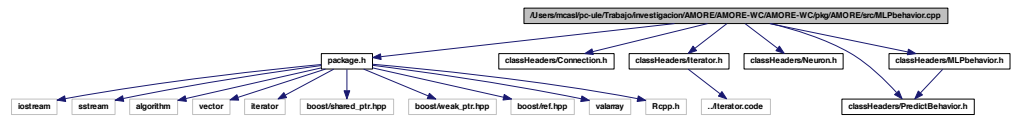
This graph shows which files directly or indirectly include this file:



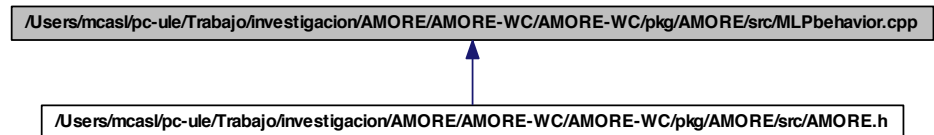
## 6.67 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPbehavior.cpp File Reference

```
#include "package.h"
#include "classHeaders/Connection.h"
#include "classHeaders/Iterator.h"
#include "classHeaders/Neuron.h"
#include "classHeaders/PredictBehavior.h"
#include "classHeaders/MLPbehavior.h"
```

Include dependency graph for MLPbehavior.cpp:



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



## 6.68 /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"
```

## 6.69 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NetworkRinterface.cpp File Reference

265

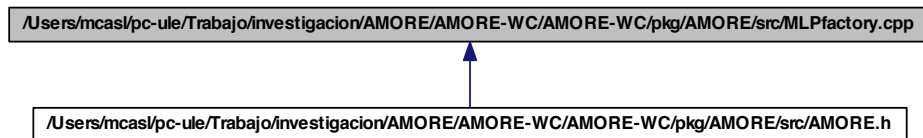
### Reference

```
#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"
```

Include dependency graph for MLPfactory.cpp:



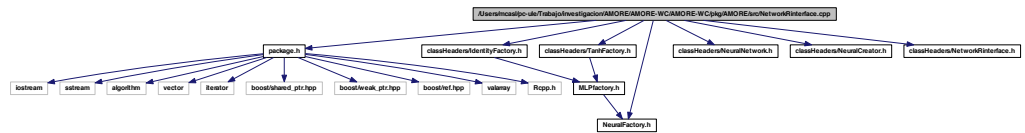
This graph shows which files directly or indirectly include this file:



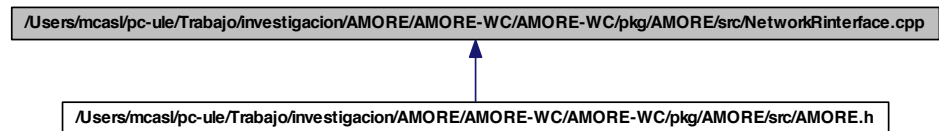
## 6.69 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NetworkRinterface.cpp File Reference

```
#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"
```

Include dependency graph for NetworkRinterface.cpp:



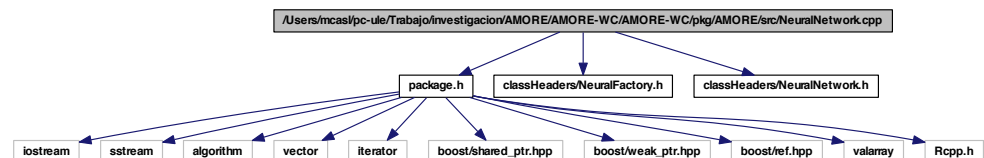
This graph shows which files directly or indirectly include this file:



## 6.70 /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"
```

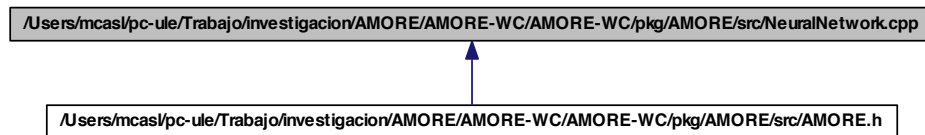
Include dependency graph for NeuralNetwork.cpp:



## 6.71 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Neuron.cpp File Reference

267

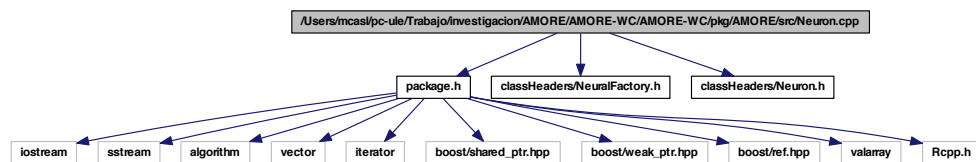
This graph shows which files directly or indirectly include this file:



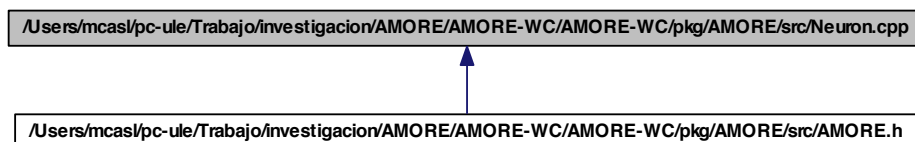
## 6.71 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Neuron.cpp File Reference

```
#include "package.h"
#include "classHeaders/NeuralFactory.h"
#include "classHeaders/Neuron.h"
```

Include dependency graph for Neuron.cpp:



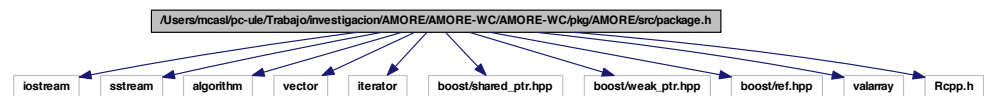
This graph shows which files directly or indirectly include this file:



## 6.72 /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](#)

Reference

- 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](#)

## 6.72.1 Define Documentation

### 6.72.1.1 #define size\_type unsigned int

Definition at line 81 of file package.h.

## 6.72.2 Typedef Documentation

### 6.72.2.1 typedef boost::shared\_ptr<ActivationFunction> ActivationFunctionPtr

Definition at line 91 of file package.h.

### 6.72.2.2 typedef boost::reference\_wrapper<PredictBehavior> ActivationFunctionRef

Definition at line 86 of file package.h.

### 6.72.2.3 typedef boost::shared\_ptr<Container<ConPtr> > ConContainerPtr

Definition at line 105 of file package.h.

### 6.72.2.4 typedef boost::shared\_ptr<Iterator<ConPtr> > ConIteratorPtr

Definition at line 100 of file package.h.

### 6.72.2.5 typedef boost::shared\_ptr<Con> ConPtr

Definition at line 96 of file package.h.

### 6.72.2.6 typedef int Handler

Definition at line 84 of file package.h.

**6.72.2.7** `typedef boost::shared_ptr<Container<LayerPtr> > LayerContainerPtr`

Definition at line 103 of file package.h.

**6.72.2.8** `typedef boost::shared_ptr<Container<NeuronPtr> > LayerPtr`

Definition at line 102 of file package.h.

**6.72.2.9** `typedef boost::shared_ptr<NetworkTrainBehavior>  
NetworkTrainBehaviorPtr`

Definition at line 93 of file package.h.

**6.72.2.10** `typedef boost::shared_ptr<NeuralCreator> NeuralCreatorPtr`

Definition at line 108 of file package.h.

**6.72.2.11** `typedef boost::shared_ptr<NeuralFactory> NeuralFactoryPtr`

Definition at line 107 of file package.h.

**6.72.2.12** `typedef boost::shared_ptr<NeuralNetwork> NeuralNetworkPtr`

Definition at line 97 of file package.h.

**6.72.2.13** `typedef boost::weak_ptr<NeuralNetwork> NeuralNetworkWeakPtr`

Definition at line 110 of file package.h.

**6.72.2.14** `typedef boost::shared_ptr<Iterator<NeuronPtr> > NeuronIteratorPtr`

Definition at line 99 of file package.h.

**6.72.2.15** `typedef boost::shared_ptr<Neuron> NeuronPtr`

Definition at line 95 of file package.h.

**6.72.2.16** `typedef boost::reference_wrapper<Neuron> NeuronRef`

Definition at line 89 of file package.h.



### 6.73 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/PredictBehavior.cpp File Reference

271

6.72.2.17 `typedef boost::shared_ptr<NeuronTrainBehavior> NeuronTrainBehaviorPtr`

Definition at line 94 of file package.h.

6.72.2.18 `typedef boost::weak_ptr<Neuron> NeuronWeakPtr`

Definition at line 111 of file package.h.

6.72.2.19 `typedef boost::shared_ptr<PredictBehavior> PredictBehaviorPtr`

Definition at line 92 of file package.h.

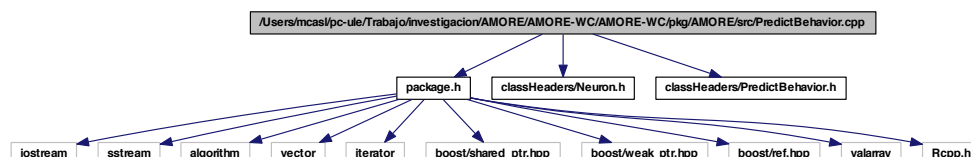
6.72.2.20 `typedef boost::reference_wrapper<PredictBehavior> PredictBehaviorRef`

Definition at line 87 of file package.h.

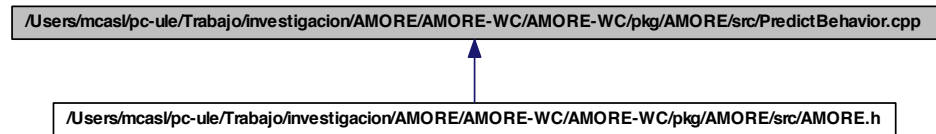
## 6.73 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/PredictBehavior.cpp File Reference

```
#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:

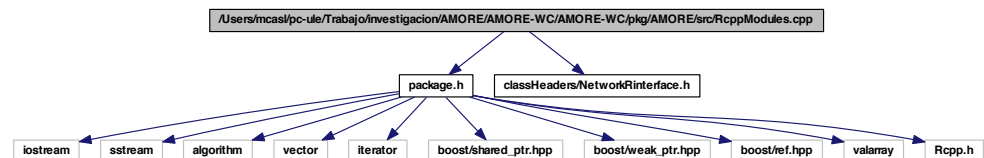


## 6.74 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/RcppModules.cpp File Reference

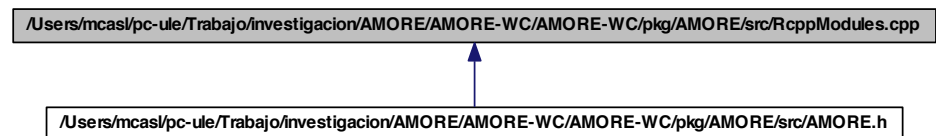
```
#include "package.h"
```

```
#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)

## 6.74.1 Function Documentation

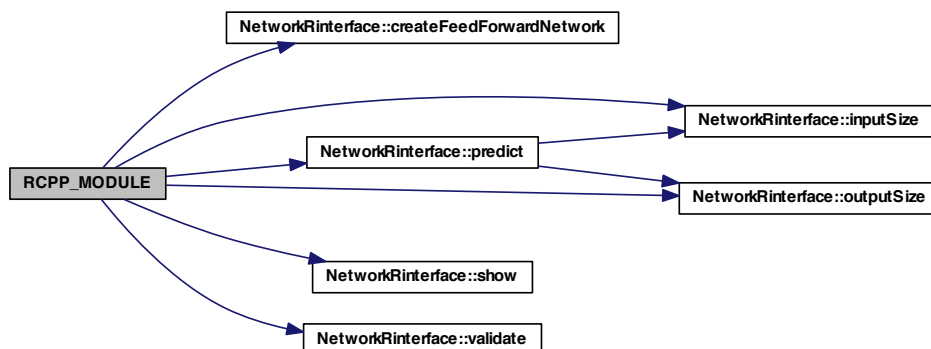
### 6.74.1.1 RCPP\_MODULE ( modAMORE )

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:



## 6.75 /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"
```

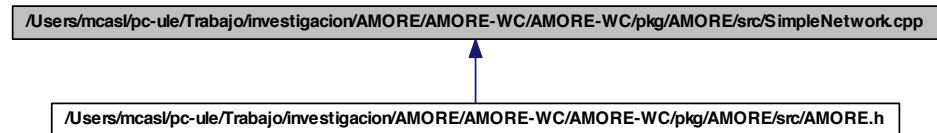
```
#include "classHeaders/NeuralNetwork.h"
```

```
#include "classHeaders/SimpleNetwork.h"
```

Include dependency graph for SimpleNetwork.cpp:



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



## 6.76 /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"
```

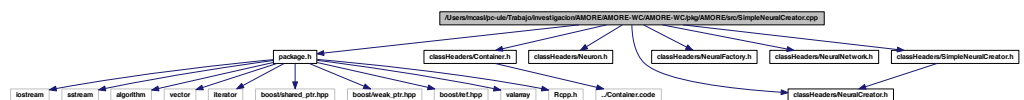
```
#include "classHeaders/NeuralCreator.h"
```

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

```
#include "classHeaders/NeuralNetwork.h"
```

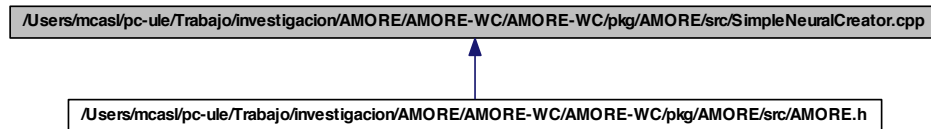
```
#include "classHeaders/SimpleNeuralCreator.h"
```

Include dependency graph for SimpleNeuralCreator.cpp:



## Reference

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



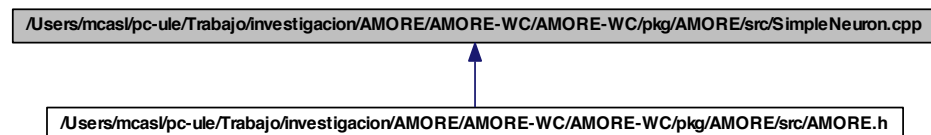
6.77 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuron.cpp File Reference

```
#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:



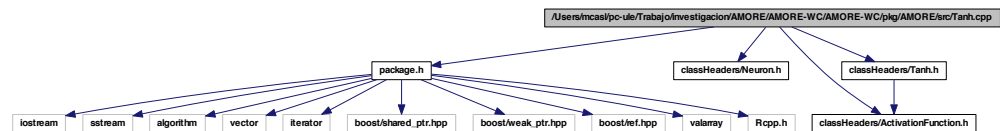
This graph shows which files directly or indirectly include this file:



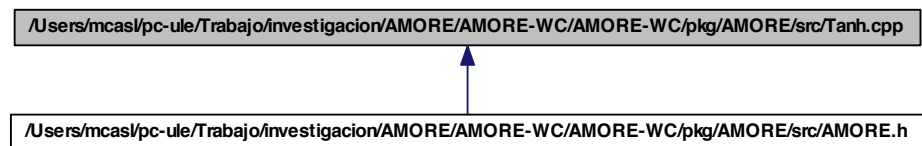
## 6.78 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Tanh.cpp File Reference

```
#include "package.h"
#include "classHeaders/Neuron.h"
#include "classHeaders/ActivationFunction.h"
#include "classHeaders/Tanh.h"
```

Include dependency graph for Tanh.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/TanhFactory.cpp File Reference

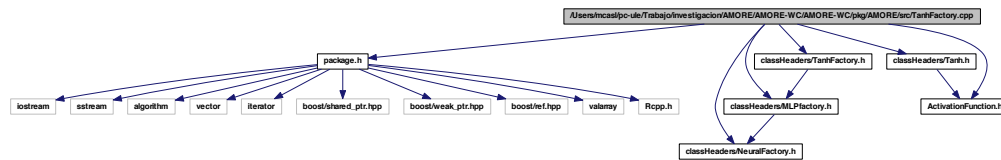
```
#include "package.h"
#include "classHeaders/NeuralFactory.h"
#include "classHeaders/MLPFactory.h"
#include "classHeaders/Tanh.h"
#include "classHeaders/TanhFactory.h"
#include "classHeaders/ActivationFunction.h"
```

## 6.79 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/TanhFactory.cpp File

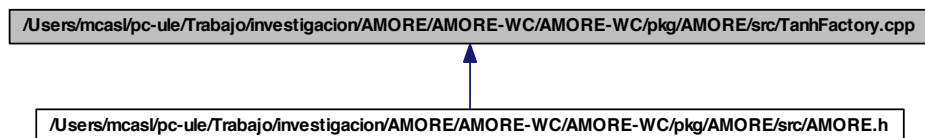
### Reference

277

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, 59	267
~Iterator	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
Iterator, 92	WC/AMORE-WC/pkg/AMORE/src/PredictBehavior.cpp,
~SimpleContainer	271
SimpleContainer, 162	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
~SimpleContainerIterator	WC/AMORE-WC/pkg/AMORE/src/RcppModules.cpp,
SimpleContainerIterator, 167	272
~SimpleContainerReverselIterator	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
SimpleContainerReverselIterator, 171	WC/AMORE-WC/pkg/AMORE/src/SimpleNetwork.cpp,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	220
WC/AMORE-WC/pkg/AMORE/src/ADAPTg	WC/AMORE-WC/pkg/AMORE/src/SimpleNeuralCreator.c
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	220
WC/AMORE-WC/pkg/AMORE/src/AMORE	WC/AMORE-WC/pkg/AMORE/src/SimpleNeuron.cpp,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	219
WC/AMORE-WC/pkg/AMORE/src/Activation	WC/AMORE-WC/pkg/AMORE/src/Tanh.cpp,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	261
WC/AMORE-WC/pkg/AMORE/src/Connections	WC/AMORE-WC/pkg/AMORE/src/TanhFactory.cpp,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	262
WC/AMORE-WC/pkg/AMORE/src/Utils	WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTg
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	263
WC/AMORE-WC/pkg/AMORE/src/Utils/Factory	WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTg
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	264
WC/AMORE-WC/pkg/AMORE/src/Utils/Factory	WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTg
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	265
WC/AMORE-WC/pkg/AMORE/src/Utils/Factory	WC/AMORE-WC/pkg/AMORE/src/classHeaders/Activatio
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	266
WC/AMORE-WC/pkg/AMORE/src/Utils/Factory	WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptNe
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	



[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
WC/AMORE-WC/pkg/AMORE/src/classHeaders/PerceptronTrainBehavior/AMORE/AMORE-  
228 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Gauss.h,](#)  
[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
WC/AMORE-WC/pkg/AMORE/src/classHeaders/PerceptronTrainBehavior/AMORE/AMORE-  
228 WC/AMORE-WC/pkg/AMORE/src/classHeaders/GaussFactory.h,](#)  
[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
WC/AMORE-WC/pkg/AMORE/src/classHeaders/PerceptronFactory/AMORE/AMORE-  
229 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Identity.h,](#)  
[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATOffHiddenInvestigBehav/AMORE/AMORE-  
230 WC/AMORE-WC/pkg/AMORE/src/classHeaders/IdentityFactory.h,](#)  
[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATOffHiddenInvestigBehav/AMORE/AMORE-  
230 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Iterator.h,](#)  
[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATOffHiddenInvestigBehav/AMORE/AMORE-  
231 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Logistic.h,](#)  
[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATOffHiddenInvestigBehav/AMORE/AMORE-  
231 WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticFactory.h,](#)  
[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
WC/AMORE-WC/pkg/AMORE/src/classHeaders/BasicNeuralNetworkTrainBehavior/AMORE/AMORE-  
232 WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPbehavior.h,](#)  
[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
WC/AMORE-WC/pkg/AMORE/src/classHeaders/BasicNeuralNetworkTrainBehavior/AMORE/AMORE-  
233 WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPfactory.h,](#)  
[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
WC/AMORE-WC/pkg/AMORE/src/classHeaders/Carroll/Trabajo/investigacion/AMORE/AMORE-  
234 WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkPinterface.h,](#)  
[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
WC/AMORE-WC/pkg/AMORE/src/classHeaders/Carroll/Trabajo/investigacion/AMORE/AMORE-  
234 WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkTrainBehavior](#)  
[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
WC/AMORE-WC/pkg/AMORE/src/classHeaders/Carroll/Trabajo/investigacion/AMORE/AMORE-  
235 WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralCreator.h,](#)  
[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
WC/AMORE-WC/pkg/AMORE/src/classHeaders/Carroll/Trabajo/investigacion/AMORE/AMORE-  
235 WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralFactory.h,](#)  
[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
WC/AMORE-WC/pkg/AMORE/src/classHeaders/Carroll/Trabajo/investigacion/AMORE/AMORE-  
236 WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralNetwork.h,](#)  
[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
WC/AMORE-WC/pkg/AMORE/src/classHeaders/Carroll/Trabajo/investigacion/AMORE/AMORE-  
236 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Neuron.h,](#)  
[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
WC/AMORE-WC/pkg/AMORE/src/classHeaders/ExplorTrabajo/investigacion/AMORE/AMORE-  
237 WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuronTrainBehavior](#)  
[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-  
WC/AMORE-WC/pkg/AMORE/src/classHeaders/ExplorTrabajo/investigacion/AMORE/AMORE-](#)

WC/AMORE-WC/pkg/AMORE/src/classHeaders/ProtectBehavior.h, 247  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Tanh.h, 249  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/TanhFactory.h, 249  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Threshold.h, 249  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ThresholdFactory.h, 248  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ThresholdFactory.h, 248  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/package.h, 250  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Reciprocal.h, 250  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ReciprocalFactory.h, 250  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ReciprocalFactory.h, 251  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainerIterator.h, 252  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainerReverselIterator.h, 253  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuron.h, 254  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuralCreator.h, 255  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuron.h, 255  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Sine.h, 256  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SineFactory.h, 257  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Square.h, 257  
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SquareFactory.h, 258

- Neuron, 130
- SimpleNeuron, 188
- AMORE.h
  - ActivationFunctionPtr, 222
  - ActivationFunctionRef, 222
  - ConContainerPtr, 222
  - ConIteratorPtr, 222
  - ConPtr, 222
  - Handler, 222
  - LayerContainerPtr, 223
  - LayerPtr, 223
  - NetworkTrainBehaviorPtr, 223
  - NeuralCreatorPtr, 223
  - NeuralFactoryPtr, 223
  - NeuralNetworkPtr, 223
  - NeuralNetworkWeakPtr, 223
  - NeuronIteratorPtr, 223
  - NeuronPtr, 223
  - NeuronRef, 223
  - NeuronTrainBehaviorPtr, 224
  - NeuronWeakPtr, 224
  - PredictBehaviorPtr, 224
  - PredictBehaviorRef, 224
  - size\_type, 222
  - TrainingBehaviorRef, 224
- ArcTan, 30
  - Arctan, 31
  - f0, 31
  - f1, 31
- Arctan
  - ArcTan, 31
- ArcTanFactory, 32
  - ArcTanFactory, 35
  - makeActivationFunction, 35
- at
  - Container, 59
  - SimpleContainer, 162
- BATCHgdNetworkTrainBehavior, 35
  - train, 37
- BATCHgdNeuronTrainBehavior, 38
  - bias, 40
  - delta, 40
  - endOfEpochAction, 40
  - rate, 40
  - singlePatternBackwardAction, 40
  - x, 40
- BATCHgdwmNetworkTrainBehavior, 41
  - train, 42
- BATCHgdwmNeuronTrainBehavior, 43
  - bias, 45
  - change, 45
  - delta, 46
  - endOfEpochAction, 45
  - momentum, 46
  - rate, 46
  - singlePatternBackwardAction, 45
  - x, 46
- BatchNetworkTrainBehavior, 46
  - train, 48
- BatchNeuronTrainBehavior, 49
  - endOfEpochAction, 50
  - singlePatternBackwardAction, 50
- bias
  - BATCHgdNeuronTrainBehavior, 40
  - BATCHgdwmNeuronTrainBehavior, 45
- change
  - ADAPTgdwmNeuronTrainBehavior, 24
  - BATCHgdwmNeuronTrainBehavior, 45
- clear
  - Container, 59
  - SimpleContainer, 162
- Con, 51
  - Con, 52
  - d\_neuron, 57
  - d\_weight, 57
  - getNeuron, 52
  - getWeight, 53
  - Id, 54
  - setNeuron, 55
  - setWeight, 55
  - show, 55
  - validate, 56
- ConContainerPtr
  - AMORE.h, 222
  - package.h, 269
- ConIteratorPtr
  - AMORE.h, 222
  - package.h, 269
- ConPtr
  - AMORE.h, 222
  - package.h, 269
- Container, 57
  - ~Container, 59
  - at, 59
  - clear, 59
  - Container, 59
  - createIterator, 59
  - createReverselIterator, 59

- empty, 60
- push\_back, 60
- reserve, 60
- show, 60
- size, 60
- validate, 60
- Cosine, 60
  - Cosine, 62
  - f0, 62
  - f1, 63
- CosineFactory, 63
  - CosineFactory, 66
  - makeActivationFunction, 66
- createFeedForwardNetwork
  - NetworkRinterface, 111
  - NeuralCreator, 118
  - SimpleNeuralCreator, 182
- createIterator
  - Container, 59
  - SimpleContainer, 163
- createReverselIterator
  - Container, 59
  - SimpleContainer, 163
- currentItem
  - Iterator, 92
  - SimpleContainerIterator, 167
  - SimpleContainerReverselIterator, 171
- d\_activationFunction
  - Neuron, 132
- d\_altitude
  - RBFbehavior, 149
- d\_bias
  - MLPbehavior, 102
- d\_collection
  - SimpleContainer, 164
- d\_container
  - SimpleContainerIterator, 168
  - SimpleContainerReverselIterator, 172
- d\_current
  - SimpleContainerIterator, 168
  - SimpleContainerReverselIterator, 172
- d\_hiddenLayers
  - NeuralNetwork, 126
- d\_id
  - Neuron, 132
- d\_inducedLocalField
  - Neuron, 132
- d\_inputLayer
  - NeuralNetwork, 126
- d\_nCons
  - Neuron, 132
- d\_networkTrainBehavior
  - NeuralNetwork, 126
- d\_neuralNetwork
  - NetworkRinterface, 115
  - NetworkTrainBehavior, 117
- d\_neuron
  - ActivationFunction, 13
  - Con, 57
  - NeuronTrainBehavior, 135
  - PredictBehavior, 140
- d\_neuronTrainBehavior
  - Neuron, 132
- d\_output
  - Neuron, 133
- d\_outputDerivative
  - Neuron, 133
- d\_outputLayer
  - NeuralNetwork, 126
- d\_predictBehavior
  - Neuron, 133
- d\_weight
  - Con, 57
- d\_width
  - RBFbehavior, 149
- delta
  - ADAPTgdNeuronTrainBehavior, 19
  - ADAPTgdwmNeuronTrainBehavior, 24
  - BATCHgdNeuronTrainBehavior, 40
  - BATCHgdwmNeuronTrainBehavior, 46
- Elliot, 66
  - Elliot, 68
  - f0, 68
  - f1, 69
- ElliotFactory, 69
  - ElliotFactory, 72
  - makeActivationFunction, 72
- empty
  - Container, 60
  - SimpleContainer, 163
- endOfEpochAction
  - ADAPTgdNeuronTrainBehavior, 19
  - ADAPTgdwmNeuronTrainBehavior, 24
  - AdaptNeuronTrainBehavior, 29
  - BATCHgdNeuronTrainBehavior, 40
  - BATCHgdwmNeuronTrainBehavior, 45
  - BatchNeuronTrainBehavior, 50
  - NeuronTrainBehavior, 134

- Exponential, [72](#)
  - Exponential, [74](#)
  - f0, [74](#)
  - f1, [75](#)
- ExponentialFactory, [75](#)
  - ExponentialFactory, [78](#)
  - makeActivationFunction, [78](#)
- f0
  - ActivationFunction, [12](#)
  - ArcTan, [31](#)
  - Cosine, [62](#)
  - Elliot, [68](#)
  - Exponential, [74](#)
  - Gauss, [80](#)
  - Identity, [87](#)
  - Logistic, [94](#)
  - RadialBasis, [142](#)
  - Reciprocal, [155](#)
  - Sine, [195](#)
  - Square, [201](#)
  - Tanh, [208](#)
  - Threshold, [214](#)
- f1
  - ActivationFunction, [12](#)
  - ArcTan, [31](#)
  - Cosine, [63](#)
  - Elliot, [69](#)
  - Exponential, [75](#)
  - Gauss, [81](#)
  - Identity, [87](#)
  - Logistic, [95](#)
  - RadialBasis, [143](#)
  - Reciprocal, [156](#)
  - Sine, [196](#)
  - Square, [202](#)
  - Tanh, [208](#)
  - Threshold, [215](#)
- first
  - Iterator, [92](#)
  - SimpleContainerIterator, [167](#)
  - SimpleContainerReverseliterator, [171](#)
- Gauss, [78](#)
  - f0, [80](#)
  - f1, [81](#)
  - Gauss, [80](#)
- GaussFactory, [81](#)
  - GaussFactory, [84](#)
  - makeActivationFunction, [84](#)
- getConliterator
  - Neuron, [130](#)
  - PredictBehavior, [137](#)
  - SimpleNeuron, [188](#)
- getId
  - Neuron, [130](#)
  - SimpleNeuron, [188](#)
- getInducedLocalField
  - ActivationFunction, [12](#)
  - Neuron, [130](#)
  - SimpleNeuron, [189](#)
- getNeuron
  - Con, [52](#)
- getOutput
  - Neuron, [130](#)
  - SimpleNeuron, [189](#)
- getWeight
  - Con, [53](#)
- Handler
  - AMORE.h, [222](#)
  - package.h, [269](#)
- Id
  - Con, [54](#)
- Identity, [84](#)
  - f0, [87](#)
  - f1, [87](#)
  - Identity, [86](#)
- IdentityFactory, [87](#)
  - IdentityFactory, [90](#)
  - makeActivationFunction, [90](#)
- inputSize
  - NetworkRinterface, [111](#)
  - NeuralNetwork, [125](#)
  - SimpleNetwork, [175](#)
- isDone
  - Iterator, [92](#)
  - SimpleContainerIterator, [167](#)
  - SimpleContainerReverseliterator, [171](#)
- Iterator, [90](#)
  - ~Iterator, [92](#)
  - currentItem, [92](#)
  - first, [92](#)
  - isDone, [92](#)
  - Iterator, [92](#)
  - next, [92](#)
- LayerContainerPtr
  - AMORE.h, [223](#)

- package.h, 269
- LayerPtr
  - AMORE.h, 223
  - package.h, 270
- Logistic, 92
  - f0, 94
  - f1, 95
  - Logistic, 94
- LogisticFactory, 95
  - LogisticFactory, 98
  - makeActivationFunction, 98
- makeActivationFunction
  - ArcTanFactory, 35
  - CosineFactory, 66
  - ElliotFactory, 72
  - ExponentialFactory, 78
  - GaussFactory, 84
  - IdentityFactory, 90
  - LogisticFactory, 98
  - MLPfactory, 105
  - NeuralFactory, 119
  - RadialBasisFactory, 146
  - RBFfactory, 152
  - ReciprocalFactory, 159
  - SineFactory, 199
  - SquareFactory, 205
  - TanhFactory, 212
  - ThresholdFactory, 218
- makeCon
  - MLPfactory, 105
  - NeuralFactory, 119
  - RBFfactory, 152
- makeConContainer
  - MLPfactory, 106
  - NeuralFactory, 119
  - RBFfactory, 152
- makeLayer
  - MLPfactory, 106
  - NeuralFactory, 120
  - RBFfactory, 152
- makeLayerContainer
  - MLPfactory, 106
  - NeuralFactory, 120
  - RBFfactory, 152
- makeNeuralCreator
  - MLPfactory, 107
  - NeuralFactory, 121
  - RBFfactory, 152
- makeNeuralNetwork
  - MLPfactory, 107
  - NeuralFactory, 121
  - RBFfactory, 152
- MLPbehavior, 98
  - d\_bias, 102
  - MLPbehavior, 101
  - MLPfactory, 102
  - show, 101
  - singlePatternForwardAction, 101
- MLPfactory, 103
  - makeActivationFunction, 105
  - makeCon, 105
  - makeConContainer, 106
  - makeLayer, 106
  - makeLayerContainer, 106
  - makeNeuralCreator, 107
  - makeNeuralNetwork, 107
  - makeNeuron, 107, 108
  - makePredictBehavior, 109
  - MLPbehavior, 102
  - Neuron, 132
- momentum
  - ADAPTgdwmNeuronTrainBehavior, 25
  - BATCHgdwmNeuronTrainBehavior, 46
- NetworkRinterface, 110
  - createFeedForwardNetwork, 111
  - d\_neuralNetwork, 115
  - inputSize, 111
  - NetworkRinterface, 111
  - outputSize, 112
  - predict, 112
  - show, 114
  - train, 114
  - validate, 115
- NetworkTrainBehavior, 116
  - d\_neuralNetwork, 117
  - train, 116
- NetworkTrainBehaviorPtr
  - AMORE.h, 223
  - package.h, 270
  - NeuralCreator, 117

- createFeedForwardNetwork, 118
- NeuralCreatorPtr
  - AMORE.h, 223
  - package.h, 270
- NeuralFactory, 118
  - makeActivationFunction, 119
  - makeCon, 119
  - makeConContainer, 119
  - makeLayer, 120
  - makeLayerContainer, 120
  - makeNeuralCreator, 121
  - makeNeuralNetwork, 121
  - makeNeuron, 121
  - makePredictBehavior, 121
- NeuralFactoryPtr
  - AMORE.h, 223
  - package.h, 270
- NeuralNetwork, 122
  - d\_hiddenLayers, 126
  - d\_inputLayer, 126
  - d\_networkTrainBehavior, 126
  - d\_outputLayer, 126
  - inputSize, 125
  - NeuralNetwork, 124
  - outputSize, 125
  - readOutput, 125
  - show, 125
  - SimpleNeuralCreator, 126
  - singlePatternBackwardAction, 125
  - singlePatternForwardAction, 125
  - train, 125
  - validate, 126
  - writeInput, 126
- NeuralNetworkPtr
  - AMORE.h, 223
  - package.h, 270
- NeuralNetworkWeakPtr
  - AMORE.h, 223
  - package.h, 270
- Neuron, 127
  - addCon, 130
  - d\_activationFunction, 132
  - d\_id, 132
  - d\_inducedLocalField, 132
  - d\_nCons, 132
  - d\_neuronTrainBehavior, 132
  - d\_output, 133
  - d\_outputDerivative, 133
  - d\_predictBehavior, 133
  - getConIterator, 130
  - getId, 130
  - getInducedLocalField, 130
  - getOutput, 130
  - MLPfactory, 132
  - Neuron, 130
  - setActivationFunction, 130
  - setId, 131
  - setInducedLocalField, 131
  - setOutput, 131
  - setOutputDerivative, 131
  - setPredictBehavior, 131
  - show, 131
  - singlePatternBackwardAction, 131
  - singlePatternForwardAction, 131
  - useActivationFunctionf0, 131
  - useActivationFunctionf1, 131
  - validate, 132
- NeuronIteratorPtr
  - AMORE.h, 223
  - package.h, 270
- NeuronPtr
  - AMORE.h, 223
  - package.h, 270
- NeuronRef
  - AMORE.h, 223
  - package.h, 270
- NeuronTrainBehavior, 133
  - d\_neuron, 135
  - endOfEpochAction, 134
  - singlePatternBackwardAction, 134
- NeuronTrainBehaviorPtr
  - AMORE.h, 224
  - package.h, 270
- NeuronWeakPtr
  - AMORE.h, 224
  - package.h, 271
- next
  - Iterator, 92
  - SimpleContainerIterator, 167
  - SimpleContainerReverseliterator, 171
- outputSize
  - NetworkRinterface, 112
  - NeuralNetwork, 125
  - SimpleNetwork, 176
- package.h
  - ActivationFunctionPtr, 269
  - ActivationFunctionRef, 269
  - ConContainerPtr, 269



- ConIteratorPtr, 269
- ConPtr, 269
- Handler, 269
- LayerContainerPtr, 269
- LayerPtr, 270
- NetworkTrainBehaviorPtr, 270
- NeuralCreatorPtr, 270
- NeuralFactoryPtr, 270
- NeuralNetworkPtr, 270
- NeuralNetworkWeakPtr, 270
- NeuronIteratorPtr, 270
- NeuronPtr, 270
- NeuronRef, 270
- NeuronTrainBehaviorPtr, 270
- NeuronWeakPtr, 271
- PredictBehaviorPtr, 271
- PredictBehaviorRef, 271
- size\_type, 269
- predict
  - NetworkRinterface, 112
- PredictBehavior, 135
  - d\_neuron, 140
  - getConIterator, 137
  - PredictBehavior, 137
  - setInducedLocalField, 137
  - setOutput, 138
  - setOutputDerivative, 138
  - show, 139
  - singlePatternForwardAction, 139
  - useActivationFunctionf0, 139
  - useActivationFunctionf1, 140
- PredictBehaviorPtr
  - AMORE.h, 224
  - package.h, 271
- PredictBehaviorRef
  - AMORE.h, 224
  - package.h, 271
- push\_back
  - Container, 60
  - SimpleContainer, 163
- RadialBasis, 141
  - f0, 142
  - f1, 143
  - RadialBasis, 142
- RadialBasisFactory, 143
  - makeActivationFunction, 146
  - RadialBasisFactory, 146
- rate
  - ADAPTgdNeuronTrainBehavior, 19
  - ADAPTgdwmNeuronTrainBehavior, 25
  - BATCHgdNeuronTrainBehavior, 40
  - BATCHgdwmNeuronTrainBehavior, 46
- RBFbehavior, 146
  - d\_altitude, 149
  - d\_width, 149
  - RBFbehavior, 149
  - show, 149
  - singlePatternForwardAction, 149
- RBFfactory, 149
  - makeActivationFunction, 152
  - makeCon, 152
  - makeConContainer, 152
  - makeLayer, 152
  - makeLayerContainer, 152
  - makeNeuralCreator, 152
  - makeNeuralNetwork, 152
  - makeNeuron, 153
  - makePredictBehavior, 153
- Rcpp\_MODULE
  - RcppModules.cpp, 273
- RcppModules.cpp
  - Rcpp\_MODULE, 273
- readOutput
  - NeuralNetwork, 125
  - SimpleNetwork, 176
- Reciprocal, 153
  - f0, 155
  - f1, 156
  - Reciprocal, 155
- ReciprocalFactory, 156
  - makeActivationFunction, 159
  - ReciprocalFactory, 159
- reserve
  - Container, 60
  - SimpleContainer, 163
- setActivationFunction
  - Neuron, 130
  - SimpleNeuron, 189
- setId
  - Neuron, 131
  - SimpleNeuron, 190
- setInducedLocalField
  - Neuron, 131
  - PredictBehavior, 137
  - SimpleNeuron, 190
- setNeuron
  - Con, 55
- setOutput



- Neuron, 131
- PredictBehavior, 138
- SimpleNeuron, 190
- setOutputDerivative
  - Neuron, 131
  - PredictBehavior, 138
  - SimpleNeuron, 190
- setPredictBehavior
  - Neuron, 131
  - SimpleNeuron, 191
- setWeight
  - Con, 55
- show
  - Con, 55
  - Container, 60
  - MLPbehavior, 101
  - NetworkRinterface, 114
  - NeuralNetwork, 125
  - Neuron, 131
  - PredictBehavior, 139
  - RBFbehavior, 149
  - SimpleContainer, 163
  - SimpleNetwork, 177
  - SimpleNeuron, 191
- SimpleContainer, 159
  - ~SimpleContainer, 162
  - at, 162
  - clear, 162
  - createIterator, 163
  - createReverselIterator, 163
  - d\_collection, 164
  - empty, 163
  - push\_back, 163
  - reserve, 163
  - show, 163
  - SimpleContainer, 162
  - SimpleContainerIterator< T >, 164
  - SimpleContainerReverselIterator< T >, 164
  - size, 163
  - validate, 163
- SimpleContainer< T >
  - SimpleContainerIterator, 168
  - SimpleContainerReverselIterator, 172
- SimpleContainerIterator, 164
  - ~SimpleContainerIterator, 167
  - currentItem, 167
  - d\_container, 168
  - d\_current, 168
  - first, 167
  - isDone, 167
  - next, 167
  - SimpleContainer< T >, 168
  - SimpleContainerIterator, 167
- SimpleContainerIterator< T >
  - SimpleContainer, 164
- SimpleContainerReverselIterator, 168
  - ~SimpleContainerReverselIterator, 171
  - currentItem, 171
  - d\_container, 172
  - d\_current, 172
  - first, 171
  - isDone, 171
  - next, 171
  - SimpleContainer< T >, 172
  - SimpleContainerReverselIterator, 171
- SimpleContainerReverselIterator< T >
  - SimpleContainer, 164
- SimpleNetwork, 172
  - inputSize, 175
  - outputSize, 176
  - readOutput, 176
  - show, 177
  - SimpleNetwork, 175
  - singlePatternBackwardAction, 177
  - singlePatternForwardAction, 178
  - train, 179
  - validate, 179
  - writelnInput, 179
- SimpleNeuralCreator, 180
  - createFeedForwardNetwork, 182
  - NeuralNetwork, 126
  - SimpleNeuralCreator, 181
- SimpleNeuron, 183
  - addCon, 188
  - getConIterator, 188
  - getId, 188
  - getInducedLocalField, 189
  - getOutput, 189
  - setActivationFunction, 189
  - setId, 190
  - setInducedLocalField, 190
  - setOutput, 190
  - setOutputDerivative, 190
  - setPredictBehavior, 191
  - show, 191
  - SimpleNeuron, 187
  - singlePatternBackwardAction, 192
  - singlePatternForwardAction, 192
  - useActivationFunction0, 192

- useActivationFunctionf1, 193
  - validate, 193
- Sine, 194
  - f0, 195
  - f1, 196
  - Sine, 195
- SineFactory, 196
  - makeActivationFunction, 199
  - SineFactory, 199
- singlePatternBackwardAction
  - ADAPTgdNeuronTrainBehavior, 19
  - ADAPTgdwmNeuronTrainBehavior, 24
  - AdaptNeuronTrainBehavior, 29
  - BATCHgdNeuronTrainBehavior, 40
  - BATCHgdwmNeuronTrainBehavior, 45
  - BatchNeuronTrainBehavior, 50
  - NeuralNetwork, 125
  - Neuron, 131
  - NeuronTrainBehavior, 134
  - SimpleNetwork, 177
  - SimpleNeuron, 192
- singlePatternForwardAction
  - MLPbehavior, 101
  - NeuralNetwork, 125
  - Neuron, 131
  - PredictBehavior, 139
  - RBFbehavior, 149
  - SimpleNetwork, 178
  - SimpleNeuron, 192
- size
  - Container, 60
  - SimpleContainer, 163
- size\_type
  - AMORE.h, 222
  - package.h, 269
- Square, 199
  - f0, 201
  - f1, 202
  - Square, 201
- SquareFactory, 202
  - makeActivationFunction, 205
  - SquareFactory, 205
- Tanh, 205
  - f0, 208
  - f1, 208
  - Tanh, 207
- TanhFactory, 209
  - makeActivationFunction, 212
  - TanhFactory, 212
- Threshold, 212
  - f0, 214
  - f1, 215
  - Threshold, 214
- ThresholdFactory, 215
  - makeActivationFunction, 218
  - ThresholdFactory, 218
- train
  - ADAPTgdNetworkTrainBehavior, 15
  - ADAPTgdwmNetworkTrainBehavior, 21
  - AdaptNetworkTrainBehavior, 27
  - BATCHgdNetworkTrainBehavior, 37
  - BATCHgdwmNetworkTrainBehavior, 42
  - BatchNetworkTrainBehavior, 48
  - NetworkRinterface, 114
  - NetworkTrainBehavior, 116
  - NeuralNetwork, 125
  - SimpleNetwork, 179
- TrainingBehaviorRef
  - AMORE.h, 224
- useActivationFunctionf0
  - Neuron, 131
  - PredictBehavior, 139
  - SimpleNeuron, 192
- useActivationFunctionf1
  - Neuron, 131
  - PredictBehavior, 140
  - SimpleNeuron, 193
- validate
  - Con, 56
  - Container, 60
  - NetworkRinterface, 115
  - NeuralNetwork, 126
  - Neuron, 132
  - SimpleContainer, 163
  - SimpleNetwork, 179
  - SimpleNeuron, 193
- writeInput
  - NeuralNetwork, 126
  - SimpleNetwork, 179
- x
  - BATCHgdNeuronTrainBehavior, 40
  - BATCHgdwmNeuronTrainBehavior, 46