

AMORE++

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

Generated by Doxygen 1.7.4

Thu Jul 28 2011 01:19:52

Contents

1	The AMORE++ package	1
1.1	Introduction	1
1.2	Motivation	1
1.3	Road Map	1
2	Class Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Class Documentation	11
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	AdaptBehavior Class Reference	13
5.2.1	Detailed Description	15

5.2.2	Member Function Documentation	15
5.2.2.1	adjustParameters	15
5.3	ADAPTgd Class Reference	16
5.3.1	Detailed Description	17
5.3.2	Member Function Documentation	17
5.3.2.1	adjustParameters	18
5.3.3	Member Data Documentation	18
5.3.3.1	outputDerivative	18
5.4	ADAPTgdwm Class Reference	18
5.4.1	Detailed Description	20
5.4.2	Member Function Documentation	20
5.4.2.1	adjustParameters	21
5.4.3	Member Data Documentation	21
5.4.3.1	outputDerivative	21
5.5	ArcTan Class Reference	21
5.5.1	Detailed Description	22
5.5.2	Member Function Documentation	22
5.5.2.1	Arctan	22
5.5.2.2	f0	22
5.5.2.3	f1	23
5.6	ArcTanFactory Class Reference	23
5.6.1	Detailed Description	26
5.6.2	Constructor & Destructor Documentation	26
5.6.2.1	ArcTanFactory	26
5.6.3	Member Function Documentation	26
5.6.3.1	makeActivationFunction	26
5.7	BatchBehavior Class Reference	26
5.7.1	Detailed Description	28
5.7.2	Member Function Documentation	28
5.7.2.1	adjustParameters	28
5.8	BATCHgd Class Reference	29
5.8.1	Detailed Description	30
5.8.2	Member Function Documentation	30
5.8.2.1	adjustParameters	31

5.8.3	Member Data Documentation	31
5.8.3.1	outputDerivative	31
5.9	BATCHgdwm Class Reference	31
5.9.1	Detailed Description	33
5.9.2	Member Function Documentation	33
5.9.2.1	adjustParameters	34
5.9.3	Member Data Documentation	34
5.9.3.1	outputDerivative	34
5.10	Con Class Reference	34
5.10.1	Detailed Description	35
5.10.2	Constructor & Destructor Documentation	35
5.10.2.1	Con	35
5.10.2.2	Con	35
5.10.3	Member Function Documentation	35
5.10.3.1	getNeuron	35
5.10.3.2	getWeight	36
5.10.3.3	Id	37
5.10.3.4	setNeuron	38
5.10.3.5	setWeight	38
5.10.3.6	show	38
5.10.3.7	validate	39
5.10.4	Member Data Documentation	40
5.10.4.1	d_neuron	40
5.10.4.2	d_weight	40
5.11	Container< T > Class Template Reference	40
5.11.1	Detailed Description	42
5.11.2	Constructor & Destructor Documentation	42
5.11.2.1	~Container	42
5.11.2.2	Container	42
5.11.3	Member Function Documentation	42
5.11.3.1	at	42
5.11.3.2	clear	42
5.11.3.3	createIterator	42
5.11.3.4	empty	43

5.11.3.5	push_back	43
5.11.3.6	reserve	43
5.11.3.7	show	43
5.11.3.8	size	43
5.11.3.9	validate	43
5.12	Cosine Class Reference	43
5.12.1	Detailed Description	45
5.12.2	Constructor & Destructor Documentation	45
5.12.2.1	Cosine	45
5.12.3	Member Function Documentation	45
5.12.3.1	f0	46
5.12.3.2	f1	46
5.13	CosineFactory Class Reference	46
5.13.1	Detailed Description	49
5.13.2	Constructor & Destructor Documentation	49
5.13.2.1	CosineFactory	49
5.13.3	Member Function Documentation	49
5.13.3.1	makeActivationFunction	49
5.14	Elliot Class Reference	49
5.14.1	Detailed Description	51
5.14.2	Constructor & Destructor Documentation	51
5.14.2.1	Elliot	51
5.14.3	Member Function Documentation	51
5.14.3.1	f0	52
5.14.3.2	f1	52
5.15	ElliotFactory Class Reference	52
5.15.1	Detailed Description	55
5.15.2	Constructor & Destructor Documentation	55
5.15.2.1	ElliotFactory	55
5.15.3	Member Function Documentation	55
5.15.3.1	makeActivationFunction	55
5.16	Exponential Class Reference	55
5.16.1	Detailed Description	57
5.16.2	Constructor & Destructor Documentation	57

5.16.2.1	Exponential	57
5.16.3	Member Function Documentation	57
5.16.3.1	f0	58
5.16.3.2	f1	58
5.17	ExponentialFactory Class Reference	58
5.17.1	Detailed Description	61
5.17.2	Constructor & Destructor Documentation	61
5.17.2.1	ExponentialFactory	61
5.17.3	Member Function Documentation	61
5.17.3.1	makeActivationFunction	61
5.18	Gauss Class Reference	61
5.18.1	Detailed Description	63
5.18.2	Constructor & Destructor Documentation	63
5.18.2.1	Gauss	63
5.18.3	Member Function Documentation	63
5.18.3.1	f0	64
5.18.3.2	f1	64
5.19	GaussFactory Class Reference	64
5.19.1	Detailed Description	67
5.19.2	Constructor & Destructor Documentation	67
5.19.2.1	GaussFactory	67
5.19.3	Member Function Documentation	67
5.19.3.1	makeActivationFunction	67
5.20	Identity Class Reference	67
5.20.1	Detailed Description	69
5.20.2	Constructor & Destructor Documentation	69
5.20.2.1	Identity	69
5.20.3	Member Function Documentation	70
5.20.3.1	f0	70
5.20.3.2	f1	70
5.21	IdentityFactory Class Reference	70
5.21.1	Detailed Description	73
5.21.2	Constructor & Destructor Documentation	73
5.21.2.1	IdentityFactory	73

5.21.3	Member Function Documentation	73
5.21.3.1	makeActivationFunction	73
5.22	Iterator< T > Class Template Reference	73
5.22.1	Detailed Description	75
5.22.2	Constructor & Destructor Documentation	75
5.22.2.1	~Iterator	75
5.22.2.2	Iterator	75
5.22.3	Member Function Documentation	75
5.22.3.1	currentItem	75
5.22.3.2	first	75
5.22.3.3	isDone	75
5.22.3.4	next	75
5.23	Logistic Class Reference	75
5.23.1	Detailed Description	77
5.23.2	Constructor & Destructor Documentation	77
5.23.2.1	Logistic	77
5.23.3	Member Function Documentation	77
5.23.3.1	f0	78
5.23.3.2	f1	78
5.24	LogisticFactory Class Reference	78
5.24.1	Detailed Description	81
5.24.2	Constructor & Destructor Documentation	81
5.24.2.1	LogisticFactory	81
5.24.3	Member Function Documentation	81
5.24.3.1	makeActivationFunction	81
5.25	MLPbehavior Class Reference	81
5.25.1	Detailed Description	84
5.25.2	Constructor & Destructor Documentation	84
5.25.2.1	MLPbehavior	84
5.25.3	Member Function Documentation	84
5.25.3.1	predict	84
5.25.3.2	show	85
5.25.4	Friends And Related Function Documentation	85
5.25.4.1	MLPfactory	85

5.25.5	Member Data Documentation	85
5.25.5.1	d_bias	85
5.26	MLPfactory Class Reference	86
5.26.1	Detailed Description	88
5.26.2	Member Function Documentation	88
5.26.2.1	makeActivationFunction	88
5.26.2.2	makeCon	88
5.26.2.3	makeConContainer	89
5.26.2.4	makeLayer	89
5.26.2.5	makeLayerContainer	90
5.26.2.6	makeNeuralCreator	90
5.26.2.7	makeNeuralNetwork	90
5.26.2.8	makeNeuron	91
5.26.2.9	makeNeuron	91
5.26.2.10	makePredictBehavior	92
5.27	NetworkRinterface Class Reference	93
5.27.1	Detailed Description	93
5.27.2	Constructor & Destructor Documentation	93
5.27.2.1	NetworkRinterface	93
5.27.3	Member Function Documentation	94
5.27.3.1	createFeedForwardNetwork	94
5.27.3.2	inputSize	94
5.27.3.3	outputSize	95
5.27.3.4	predict	95
5.27.3.5	show	97
5.27.3.6	validate	97
5.27.4	Member Data Documentation	98
5.27.4.1	d_neuralNetwork	98
5.28	NeuralCreator Class Reference	98
5.28.1	Detailed Description	99
5.28.2	Member Function Documentation	99
5.28.2.1	createFeedForwardNetwork	99
5.29	NeuralFactory Class Reference	100
5.29.1	Detailed Description	100

5.29.2	Member Function Documentation	100
5.29.2.1	makeActivationFunction	100
5.29.2.2	makeCon	101
5.29.2.3	makeConContainer	101
5.29.2.4	makeLayer	101
5.29.2.5	makeLayerContainer	101
5.29.2.6	makeNeuralCreator	102
5.29.2.7	makeNeuralNetwork	102
5.29.2.8	makeNeuron	102
5.29.2.9	makeNeuron	103
5.29.2.10	makePredictBehavior	103
5.30	NeuralNetwork Class Reference	103
5.30.1	Detailed Description	105
5.30.2	Constructor & Destructor Documentation	105
5.30.2.1	NeuralNetwork	105
5.30.3	Member Function Documentation	106
5.30.3.1	inputSize	106
5.30.3.2	outputSize	106
5.30.3.3	predict	106
5.30.3.4	readOutput	106
5.30.3.5	show	106
5.30.3.6	validate	106
5.30.3.7	writeInput	106
5.30.4	Friends And Related Function Documentation	106
5.30.4.1	SimpleNeuralCreator	106
5.30.5	Member Data Documentation	106
5.30.5.1	d_hiddenLayers	106
5.30.5.2	d_inputLayer	107
5.30.5.3	d_outputLayer	107
5.31	Neuron Class Reference	107
5.31.1	Detailed Description	109
5.31.2	Constructor & Destructor Documentation	110
5.31.2.1	Neuron	110
5.31.3	Member Function Documentation	110

5.31.3.1	addCon	110
5.31.3.2	getConlterator	110
5.31.3.3	getId	110
5.31.3.4	getInducedLocalField	110
5.31.3.5	getOutput	110
5.31.3.6	predict	111
5.31.3.7	setActivationFunction	111
5.31.3.8	setId	111
5.31.3.9	setInducedLocalField	111
5.31.3.10	setOutput	111
5.31.3.11	setPredictBehavior	111
5.31.3.12	show	111
5.31.3.13	useActivationFunctionf0	111
5.31.3.14	validate	111
5.31.4	Friends And Related Function Documentation	112
5.31.4.1	MLPfactory	112
5.31.5	Member Data Documentation	112
5.31.5.1	d_activationFunction	112
5.31.5.2	d_Id	112
5.31.5.3	d_inducedLocalField	112
5.31.5.4	d_nCons	112
5.31.5.5	d_output	112
5.31.5.6	d_predictBehavior	112
5.32	PredictBehavior Class Reference	113
5.32.1	Detailed Description	114
5.32.2	Constructor & Destructor Documentation	114
5.32.2.1	PredictBehavior	114
5.32.3	Member Function Documentation	114
5.32.3.1	getConlterator	114
5.32.3.2	predict	115
5.32.3.3	setInducedLocalField	115
5.32.3.4	setOutput	115
5.32.3.5	show	116
5.32.3.6	useActivationFunctionf0	116

5.32.4	Member Data Documentation	117
5.32.4.1	d_neuron	117
5.33	RadialBasis Class Reference	117
5.33.1	Detailed Description	118
5.33.2	Constructor & Destructor Documentation	118
5.33.2.1	RadialBasis	118
5.33.3	Member Function Documentation	118
5.33.3.1	f0	119
5.33.3.2	f1	119
5.34	RadialBasisFactory Class Reference	119
5.34.1	Detailed Description	122
5.34.2	Constructor & Destructor Documentation	122
5.34.2.1	RadialBasisFactory	122
5.34.3	Member Function Documentation	122
5.34.3.1	makeActivationFunction	122
5.35	RBFbehavior Class Reference	122
5.35.1	Detailed Description	125
5.35.2	Constructor & Destructor Documentation	125
5.35.2.1	RBFbehavior	125
5.35.3	Member Function Documentation	125
5.35.3.1	predict	125
5.35.3.2	show	125
5.35.4	Member Data Documentation	125
5.35.4.1	d_altitude	125
5.35.4.2	d_width	125
5.36	RBFfactory Class Reference	125
5.36.1	Detailed Description	128
5.36.2	Member Function Documentation	128
5.36.2.1	makeActivationFunction	128
5.36.2.2	makeCon	128
5.36.2.3	makeConContainer	128
5.36.2.4	makeLayer	128
5.36.2.5	makeLayerContainer	128
5.36.2.6	makeNeuralCreator	128

5.36.2.7	makeNeuralNetwork	129
5.36.2.8	makeNeuron	129
5.36.2.9	makeNeuron	129
5.36.2.10	makePredictBehavior	129
5.37	Reciprocal Class Reference	129
5.37.1	Detailed Description	131
5.37.2	Constructor & Destructor Documentation	131
5.37.2.1	Reciprocal	131
5.37.3	Member Function Documentation	131
5.37.3.1	f0	132
5.37.3.2	f1	132
5.38	ReciprocalFactory Class Reference	132
5.38.1	Detailed Description	135
5.38.2	Constructor & Destructor Documentation	135
5.38.2.1	ReciprocalFactory	135
5.38.3	Member Function Documentation	135
5.38.3.1	makeActivationFunction	135
5.39	SimpleContainer< T > Class Template Reference	135
5.39.1	Detailed Description	138
5.39.2	Constructor & Destructor Documentation	138
5.39.2.1	SimpleContainer	138
5.39.2.2	~SimpleContainer	138
5.39.3	Member Function Documentation	138
5.39.3.1	at	138
5.39.3.2	clear	138
5.39.3.3	createIterator	139
5.39.3.4	empty	139
5.39.3.5	push_back	139
5.39.3.6	reserve	139
5.39.3.7	show	139
5.39.3.8	size	139
5.39.3.9	validate	139
5.39.4	Friends And Related Function Documentation	139
5.39.4.1	SimpleContainerIterator< T >	139

5.39.5	Member Data Documentation	140
5.39.5.1	d_collection	140
5.40	SimpleContainerIterator< T > Class Template Reference	140
5.40.1	Detailed Description	142
5.40.2	Constructor & Destructor Documentation	142
5.40.2.1	SimpleContainerIterator	142
5.40.2.2	~SimpleContainerIterator	142
5.40.3	Member Function Documentation	142
5.40.3.1	currentItem	142
5.40.3.2	first	142
5.40.3.3	isDone	142
5.40.3.4	next	143
5.40.4	Friends And Related Function Documentation	143
5.40.4.1	SimpleContainer< T >	143
5.40.5	Member Data Documentation	143
5.40.5.1	d_container	143
5.40.5.2	d_current	143
5.41	SimpleNetwork Class Reference	143
5.41.1	Detailed Description	146
5.41.2	Constructor & Destructor Documentation	146
5.41.2.1	SimpleNetwork	146
5.41.3	Member Function Documentation	146
5.41.3.1	inputSize	146
5.41.3.2	outputSize	147
5.41.3.3	predict	147
5.41.3.4	readOutput	148
5.41.3.5	show	148
5.41.3.6	validate	149
5.41.3.7	writelnInput	149
5.42	SimpleNeuralCreator Class Reference	150
5.42.1	Detailed Description	151
5.42.2	Constructor & Destructor Documentation	151
5.42.2.1	SimpleNeuralCreator	151
5.42.3	Member Function Documentation	152

5.42.3.1	createFeedForwardNetwork	152
5.43	SimpleNeuron Class Reference	153
5.43.1	Detailed Description	157
5.43.2	Constructor & Destructor Documentation	157
5.43.2.1	SimpleNeuron	157
5.43.3	Member Function Documentation	157
5.43.3.1	addCon	157
5.43.3.2	getConIterator	158
5.43.3.3	getId	158
5.43.3.4	getInducedLocalField	158
5.43.3.5	getOutput	159
5.43.3.6	predict	159
5.43.3.7	setActivationFunction	159
5.43.3.8	setId	159
5.43.3.9	setInducedLocalField	160
5.43.3.10	setOutput	160
5.43.3.11	setPredictBehavior	160
5.43.3.12	show	160
5.43.3.13	useActivationFunctionf0	161
5.43.3.14	validate	162
5.44	Sine Class Reference	162
5.44.1	Detailed Description	164
5.44.2	Constructor & Destructor Documentation	164
5.44.2.1	Sine	164
5.44.3	Member Function Documentation	164
5.44.3.1	f0	165
5.44.3.2	f1	165
5.45	SineFactory Class Reference	165
5.45.1	Detailed Description	168
5.45.2	Constructor & Destructor Documentation	168
5.45.2.1	SineFactory	168
5.45.3	Member Function Documentation	168
5.45.3.1	makeActivationFunction	168
5.46	Square Class Reference	168

5.46.1 Detailed Description	170
5.46.2 Constructor & Destructor Documentation	170
5.46.2.1 Square	170
5.46.3 Member Function Documentation	170
5.46.3.1 f0	171
5.46.3.2 f1	171
5.47 SquareFactory Class Reference	171
5.47.1 Detailed Description	174
5.47.2 Constructor & Destructor Documentation	174
5.47.2.1 SquareFactory	174
5.47.3 Member Function Documentation	174
5.47.3.1 makeActivationFunction	174
5.48 Tanh Class Reference	174
5.48.1 Detailed Description	176
5.48.2 Constructor & Destructor Documentation	176
5.48.2.1 Tanh	176
5.48.3 Member Function Documentation	177
5.48.3.1 f0	177
5.48.3.2 f1	177
5.49 TanhFactory Class Reference	178
5.49.1 Detailed Description	181
5.49.2 Constructor & Destructor Documentation	181
5.49.2.1 TanhFactory	181
5.49.3 Member Function Documentation	181
5.49.3.1 makeActivationFunction	181
5.50 Threshold Class Reference	181
5.50.1 Detailed Description	183
5.50.2 Constructor & Destructor Documentation	183
5.50.2.1 Threshold	183
5.50.3 Member Function Documentation	183
5.50.3.1 f0	184
5.50.3.2 f1	184
5.51 ThresholdFactory Class Reference	184
5.51.1 Detailed Description	187

5.51.2	Constructor & Destructor Documentation	187
5.51.2.1	ThresholdFactory	187
5.51.3	Member Function Documentation	187
5.51.3.1	makeActivationFunction	187
5.52	TrainingBehavior Class Reference	187
5.52.1	Detailed Description	188
5.52.2	Member Function Documentation	188
5.52.2.1	adjustParameters	188
6	File Documentation	189
6.1	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/ActivationFunction.cpp File Reference	189
6.2	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/AMORE.h File Reference	190
6.2.1	Define Documentation	192
6.2.1.1	size_type	192
6.2.2	Typedef Documentation	192
6.2.2.1	ActivationFunctionPtr	192
6.2.2.2	ActivationFunctionRef	192
6.2.2.3	ConContainerPtr	192
6.2.2.4	ConIteratorPtr	192
6.2.2.5	ConPtr	192
6.2.2.6	Handler	192
6.2.2.7	LayerContainerPtr	193
6.2.2.8	LayerPtr	193
6.2.2.9	NeuralCreatorPtr	193
6.2.2.10	NeuralFactoryPtr	193
6.2.2.11	NeuralNetworkPtr	193
6.2.2.12	NeuronIteratorPtr	193
6.2.2.13	NeuronPtr	193
6.2.2.14	NeuronRef	193
6.2.2.15	NeuronWeakPtr	193
6.2.2.16	PredictBehaviorPtr	193
6.2.2.17	PredictBehaviorRef	194
6.2.2.18	TrainingBehaviorRef	194

6.3	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ActivationFunction.h File Reference	194
6.4	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptBehavior.h File Reference	194
6.5	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgd.h File Reference	195
6.6	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwm.h File Reference	195
6.7	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ArcTan.h File Reference	196
6.8	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ArcTanFactory.h File Reference	197
6.9	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BatchBehavior.h File Reference	197
6.10	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgd.h File Reference	198
6.11	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwm.h File Reference	198
6.12	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Connection.h File Reference	199
6.13	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Container.h File Reference	199
6.14	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Cosine.h File Reference	200
6.15	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/CosineFactory.h File Reference	201
6.16	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Elliot.h File Reference	201
6.17	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ElliotFactory.h File Reference	202
6.18	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Exponential.h File Reference	202
6.19	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ExponentialFactory.h File Reference	203
6.20	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Gauss.h File Reference	203
6.21	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/GaussFactory.h File Reference	204
6.22	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Identity.h File Reference	205

6.23	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/IdentityFactory.h File Reference . . .	205
6.24	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Iterator.h File Reference	206
6.25	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Logistic.h File Reference	207
6.26	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticFactory.h File Reference . . .	207
6.27	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPbehavior.h File Reference . . .	208
6.28	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPfactory.h File Reference	209
6.29	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkRinterface.h File Reference .	210
6.30	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralCreator.h File Reference . . .	210
6.31	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralFactory.h File Reference . . .	211
6.32	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralNetwork.h File Reference . . .	211
6.33	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Neuron.h File Reference	211
6.34	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/PredictBehavior.h File Reference . .	212
6.35	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasis.h File Reference	212
6.36	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasisFactory.h File Reference	213
6.37	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFbehavior.h File Reference	213
6.38	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFfactory.h File Reference	214
6.39	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Reciprocal.h File Reference	214
6.40	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ReciprocalFactory.h File Reference .	215
6.41	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainer.h File Reference . .	216

6.42	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainerIterator.h File Reference	216
6.43	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNetwork.h File Reference	217
6.44	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuralCreator.h File Reference	218
6.45	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuron.h File Reference	219
6.46	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Sine.h File Reference	220
6.47	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SineFactory.h File Reference	220
6.48	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Square.h File Reference	221
6.49	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SquareFactory.h File Reference	221
6.50	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Tanh.h File Reference	222
6.51	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/TanhFactory.h File Reference	223
6.52	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Threshold.h File Reference	223
6.53	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ThresholdFactory.h File Reference	224
6.54	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/TrainingBehavior.h File Reference	225
6.55	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Connection.cpp File Reference	225
6.56	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Identity.cpp File Reference	225
6.57	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/IdentityFactory.cpp File Reference	226
6.58	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPbehavior.cpp File Reference	227
6.59	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPfactory.cpp File Reference	228
6.60	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NetworkRinterface.cpp File Reference	229

6.61	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NeuralNetwork.cpp File Reference	229
6.62	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Neuron.cpp File Reference	230
6.63	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/package.h File Reference	231
6.63.1	Define Documentation	232
6.63.1.1	size_type	232
6.63.2	Typedef Documentation	232
6.63.2.1	ActivationFunctionPtr	232
6.63.2.2	ActivationFunctionRef	232
6.63.2.3	ConContainerPtr	232
6.63.2.4	ConIteratorPtr	233
6.63.2.5	ConPtr	233
6.63.2.6	Handler	233
6.63.2.7	LayerContainerPtr	233
6.63.2.8	LayerPtr	233
6.63.2.9	NeuralCreatorPtr	233
6.63.2.10	NeuralFactoryPtr	233
6.63.2.11	NeuralNetworkPtr	233
6.63.2.12	NeuronIteratorPtr	233
6.63.2.13	NeuronPtr	233
6.63.2.14	NeuronRef	234
6.63.2.15	NeuronWeakPtr	234
6.63.2.16	PredictBehaviorPtr	234
6.63.2.17	PredictBehaviorRef	234
6.64	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/PredictBehavior.cpp File Reference	234
6.65	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/RcppModules.cpp File Reference	235
6.65.1	Function Documentation	236
6.65.1.1	RCPP_MODULE	236
6.66	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNetwork.cpp File Reference	236
6.67	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuralCreator.cpp File Reference	237

6.68	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuron.cpp File Reference	238
6.69	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Tanh.cpp File Reference	239
6.70	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/TanhFactory.cpp File Reference	239

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	21
Cosine	43
Elliot	49
Exponential	55
Gauss	61
Identity	67
Logistic	75
RadialBasis	117
Reciprocal	129
Sine	162
Square	168
Tanh	174
Threshold	181
Con	34
Container< T >	40
SimpleContainer< T >	135
Iterator< T >	73
SimpleContainerIterator< T >	140
NetworkRinterface	93
NeuralCreator	98
SimpleNeuralCreator	150
NeuralFactory	100
MLPfactory	86
ArcTanFactory	23
CosineFactory	46
ElliotFactory	52
ExponentialFactory	58

GaussFactory	64
IdentityFactory	70
LogisticFactory	78
ReciprocalFactory	132
SineFactory	165
SquareFactory	171
TanhFactory	178
ThresholdFactory	184
RBFfactory	125
RadialBasisFactory	119
NeuralNetwork	103
SimpleNetwork	143
Neuron	107
SimpleNeuron	153
PredictBehavior	113
MLPbehavior	81
RBFbehavior	122
TrainingBehavior	187
AdaptBehavior	13
ADAPTgd	16
ADAPTgdwm	18
BatchBehavior	26
BATCHgd	29
BATCHgdwm	31

Chapter 3

Class Index

3.1 Class List

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

ActivationFunction (Class ActivationFunction -)	11
AdaptBehavior (Class AdaptBehavior -)	13
ADAPTgd (Class ADAPTgd -)	16
ADAPTgdwm (Class ADAPTgdwm -)	18
ArcTan (Class ArcTan -)	21
ArcTanFactory (Class ArcTanFactory -)	23
BatchBehavior (Class BatchBehavior -)	26
BATCHgd (Class BATCHgd -)	29
BATCHgdwm (Class BATCHgdwm -)	31
Con (Class Con -)	34
Container< T > (Class Container -)	40
Cosine (Class Cosine -)	43
CosineFactory (Class CosineFactory -)	46
Elliot (Class Elliot -)	49
ElliotFactory (Class ElliotFactory -)	52
Exponential (Class Exponential -)	55
ExponentialFactory (Class ExponentialFactory -)	58
Gauss (Class Gauss -)	61
GaussFactory (Class GaussFactory -)	64
Identity (Class Identity -)	67
IdentityFactory (Class IdentityFactory -)	70
Iterator< T > (Class Iterator -)	73
Logistic (Class Logistic -)	75
LogisticFactory (Class LogisticFactory -)	78
MLPbehavior (Class MLPbehavior -)	81
MLPfactory (Class MLPfactory -)	86
NetworkRinterface (Class NetworkRinterface -)	93
NeuralCreator (Class NeuralCreator -)	98
NeuralFactory (Class NeuralFactory -)	100

NeuralNetwork (Class NeuralNetwork -)	103
Neuron (Class Neuron -)	107
PredictBehavior (Class PredictBehavior -)	113
RadialBasis (Class RadialBasis -)	117
RadialBasisFactory (Class RadialBasisFactory -)	119
RBFbehavior (Class RBFbehavior -)	122
RBFfactory (Class RBFfactory -)	125
Reciprocal (Class Reciprocal -)	129
ReciprocalFactory (Class ReciprocalFactory -)	132
SimpleContainer< T > (Class SimpleContainer -)	135
SimpleContainerIterator< T > (Class SimpleContainerIterator -)	140
SimpleNetwork (Class SimpleNetwork -)	143
SimpleNeuralCreator (Class SimpleNeuralCreator -)	150
SimpleNeuron (Class SimpleNeuron -)	153
Sine (Class Sine -)	162
SineFactory (Class SineFactory -)	165
Square (Class Square -)	168
SquareFactory (Class SquareFactory -)	171
Tanh (Class Tanh -)	174
TanhFactory (Class TanhFactory -)	178
Threshold (Class Threshold -)	181
ThresholdFactory (Class ThresholdFactory -)	184
TrainingBehavior (Class TrainingBehavior -)	187

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](#)
189

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[AMORE.h](#)
190

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[Connection.cpp](#)
225

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[Identity.cpp](#)
225

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[IdentityFactory.cpp](#)
226

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[MLPbehavior.cpp](#)
227

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[MLPfactory.cpp](#)
228

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[NetworkRinterface.cpp](#)
229

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[NeuralNetwork.cpp](#)
229

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[Neuron.cpp](#)
230

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[package.h](#)
231

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[PredictBehavior.cpp](#)
234

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[RcppModules.cpp](#)
235

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[SimpleNetwork.cpp](#)
236

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeural
237

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuron
238

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Tanh.cpp
239

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/TanhFactory.c
239

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
194

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
194

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
195

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
195

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
196

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
197

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
197

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
198

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
198

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
199

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
199

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
200

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
201

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
201

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
202

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
202

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
203

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
203

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
204

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
205

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
205

/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Iterator.h	206
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Logistic.h	207
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticFactory.h	207
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPbehavior.h	208
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPfactory.h	209
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NetworkPrint.h	210
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralCreat.h	210
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralFactory.h	211
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralNetwork.h	211
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Neuron.h	211
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/PredictBehavior.h	212
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasis.h	212
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasisFactory.h	213
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFbehavior.h	213
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFfactory.h	214
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Reciprocal.h	214
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ReciprocalFactory.h	215
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainer.h	216
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainerFactory.h	216
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNetwork.h	217
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuralNetwork.h	218
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuralNetworkFactory.h	219
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Sine.h	220
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SineFactory.h	220
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Square.h	221

[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
221](#)

[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
222](#)

[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
223](#)

[/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders.
223](#)

[/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](#)

Chapter 5

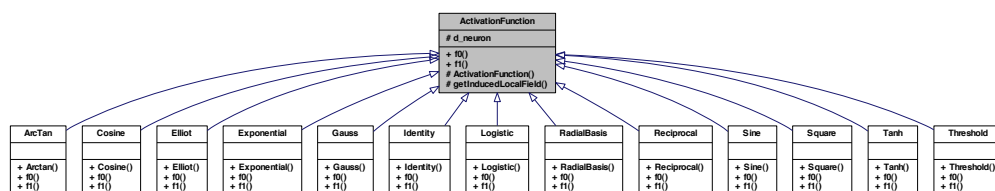
Class Documentation

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`.

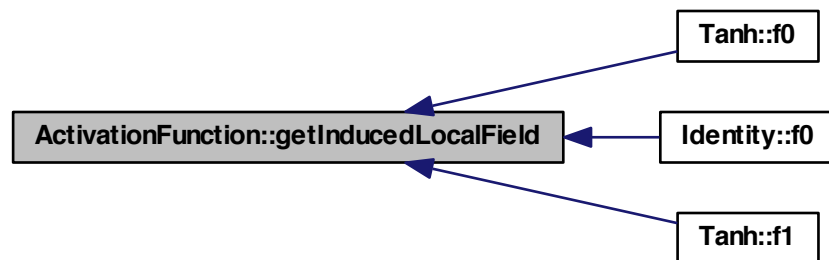
Referenced by `Tanh::f0()`, `Identity::f0()`, and `Tanh::f1()`.

```

{
    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:

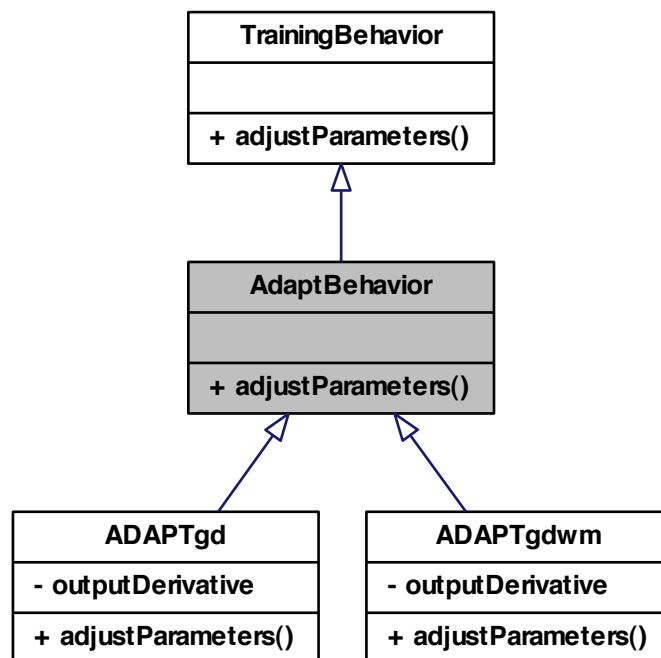
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ActivationFunction.h`
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/ActivationFunction.cpp`

5.2 AdaptBehavior Class Reference

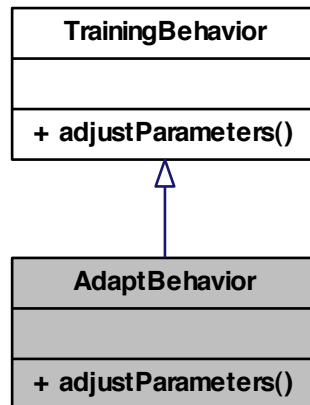
class `AdaptBehavior` -

```
#include <AdaptBehavior.h>
```

Inheritance diagram for AdaptBehavior:



Collaboration diagram for AdaptBehavior:



Public Member Functions

- virtual void [adjustParameters](#) ()=0

5.2.1 Detailed Description

class [AdaptBehavior](#) -

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

5.2.2 Member Function Documentation

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

Reimplemented from [TrainingBehavior](#).

Implemented in [ADAPTgd](#), and [ADAPTgdwm](#).

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

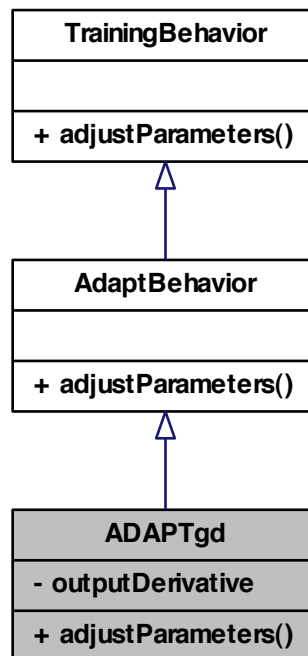
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptBeh](#)

5.3 ADAPTgd Class Reference

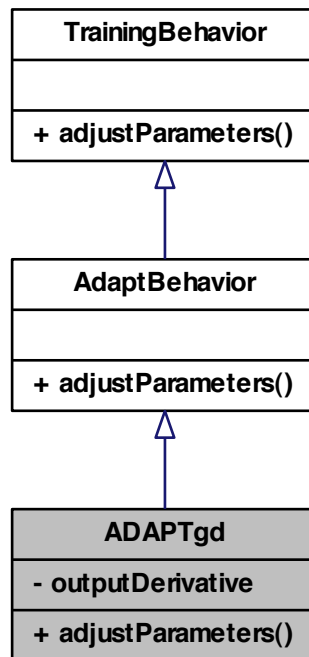
class [ADAPTgd](#) -

```
#include <ADAPTgd.h>
```

Inheritance diagram for ADAPTgd:



Collaboration diagram for ADAPTgd:



Public Member Functions

- void [adjustParameters](#) ()

Private Attributes

- double [outputDerivative](#)

5.3.1 Detailed Description

class [ADAPTgd](#) -

Definition at line 5 of file ADAPTgd.h.

5.3.2 Member Function Documentation

5.3.2.1 void ADAPTgd::adjustParameters () [virtual]

Implements [AdaptBehavior](#).

5.3.3 Member Data Documentation

5.3.3.1 double ADAPTgd::outputDerivative [private]

Definition at line 8 of file ADAPTgd.h.

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

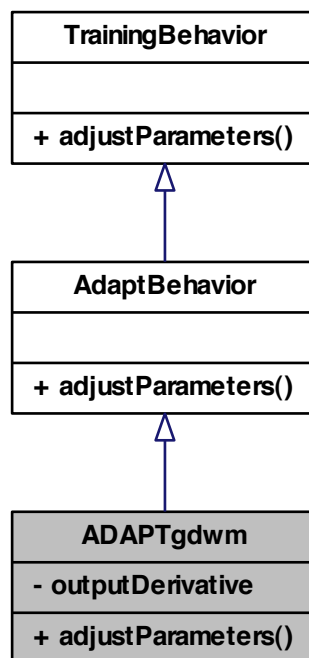
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

5.4 ADAPTgdwm Class Reference

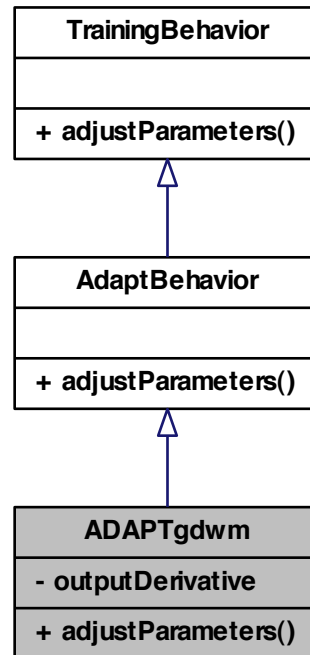
class [ADAPTgdwm](#) -

#include <ADAPTgdwm.h>

Inheritance diagram for ADAPTgdwm:



Collaboration diagram for ADAPTgdwm:



Public Member Functions

- void [adjustParameters](#) ()

Private Attributes

- double [outputDerivative](#)

5.4.1 Detailed Description

class [ADAPTgdwm](#) -

Definition at line 5 of file ADAPTgdwm.h.

5.4.2 Member Function Documentation

5.4.2.1 void ADAPTgdwm::adjustParameters () [virtual]

Implements [AdaptBehavior](#).

5.4.3 Member Data Documentation

5.4.3.1 double ADAPTgdwm::outputDerivative [private]

Definition at line 8 of file ADAPTgdwm.h.

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

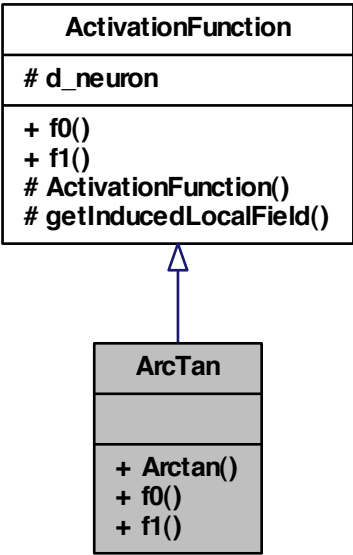
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[ADAPTgdwm.h](#)

5.5 ArcTan Class Reference

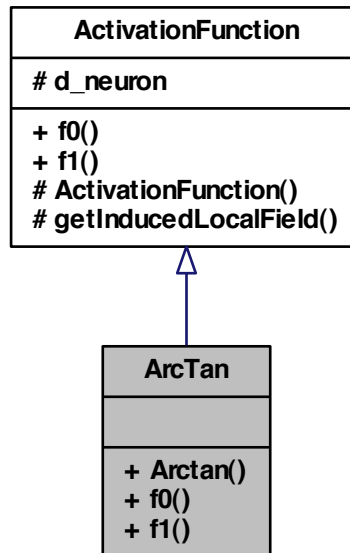
class [ArcTan](#) -

```
#include <ArcTan.h>
```

Inheritance diagram for ArcTan:



Collaboration diagram for ArcTan:



Public Member Functions

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

5.5.1 Detailed Description

class [ArcTan](#) -

Definition at line 5 of file ArcTan.h.

5.5.2 Member Function Documentation

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

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

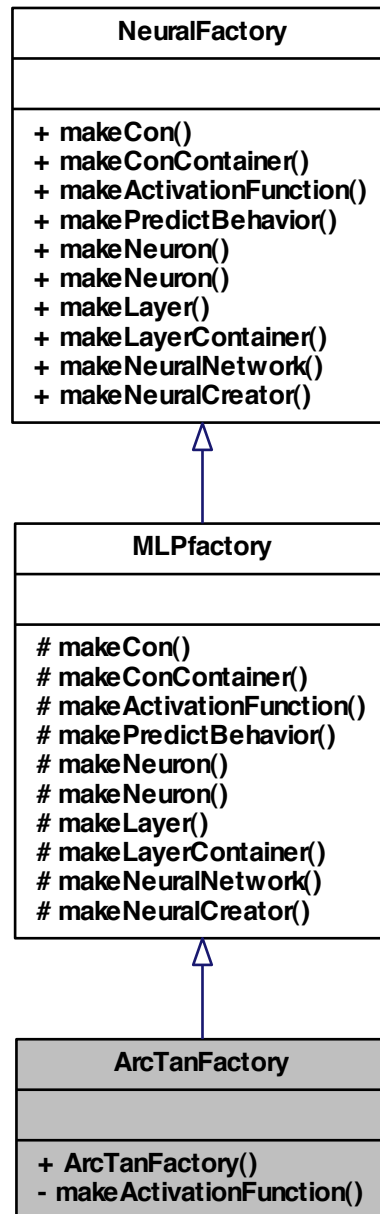
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ArcTan.h`

5.6 ArcTanFactory Class Reference

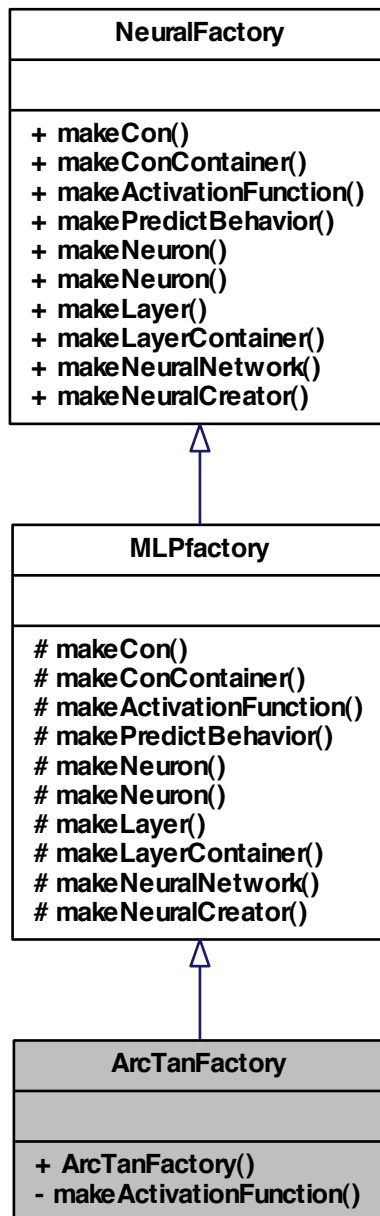
class [ArcTanFactory](#) -

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

Inheritance diagram for ArcTanFactory:



Collaboration diagram for ArcTanFactory:



Public Member Functions

- [ArcTanFactory](#) ()

Private Member Functions

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

5.6.1 Detailed Description

class [ArcTanFactory](#) -

Definition at line 5 of file ArcTanFactory.h.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 [ArcTanFactory::ArcTanFactory](#) ()

5.6.3 Member Function Documentation

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

Implements [MLPfactory](#).

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

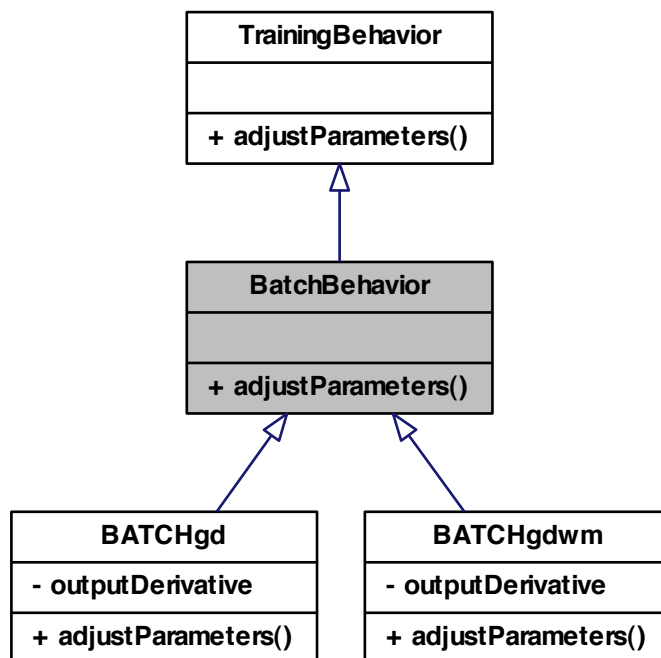
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

5.7 BatchBehavior Class Reference

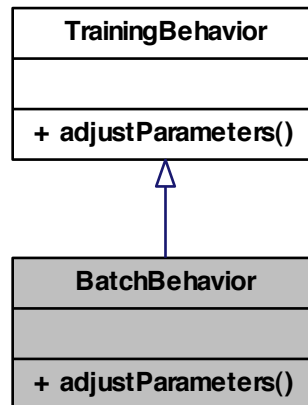
class [BatchBehavior](#) -

```
#include <BatchBehavior.h>
```


Inheritance diagram for BatchBehavior:



Collaboration diagram for BatchBehavior:



Public Member Functions

- virtual void [adjustParameters](#) ()=0

5.7.1 Detailed Description

class [BatchBehavior](#) -

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

5.7.2 Member Function Documentation

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

Reimplemented from [TrainingBehavior](#).

Implemented in [BATCHgd](#), and [BATCHgdwm](#).

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

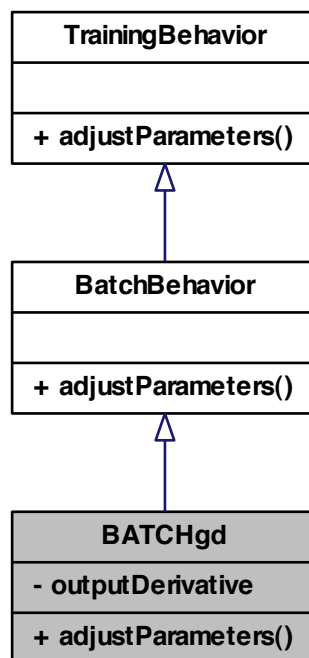
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead](#)

5.8 BATCHgd Class Reference

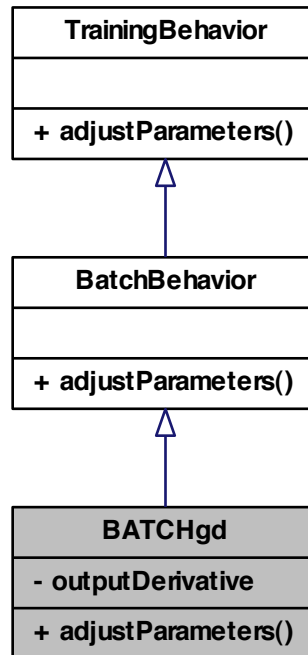
class [BATCHgd](#) -

```
#include <BATCHgd.h>
```

Inheritance diagram for BATCHgd:



Collaboration diagram for BATCHgd:



Public Member Functions

- void [adjustParameters](#) ()

Private Attributes

- double [outputDerivative](#)

5.8.1 Detailed Description

class [BATCHgd](#) -

Definition at line 5 of file BATCHgd.h.

5.8.2 Member Function Documentation

5.8.2.1 void BATCHgd::adjustParameters () [virtual]

Implements [BatchBehavior](#).

5.8.3 Member Data Documentation

5.8.3.1 double BATCHgd::outputDerivative [private]

Definition at line 8 of file BATCHgd.h.

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

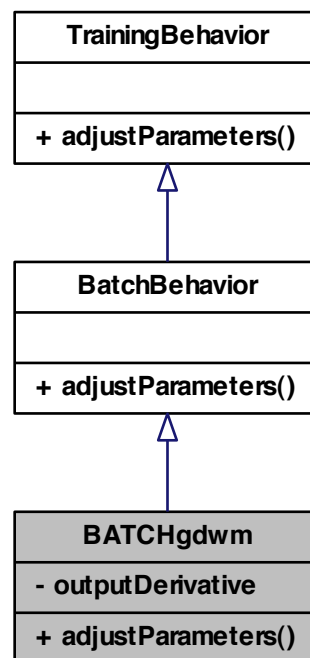
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[BATCHgd](#)

5.9 BATCHgdwm Class Reference

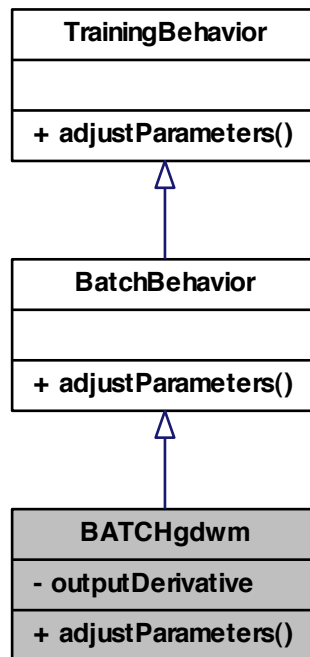
class [BATCHgdwm](#) -

```
#include <BATCHgdwm.h>
```

Inheritance diagram for BATCHgdwm:



Collaboration diagram for BATCHgdwm:



Public Member Functions

- void [adjustParameters](#) ()

Private Attributes

- double [outputDerivative](#)

5.9.1 Detailed Description

class [BATCHgdwm](#) -

Definition at line 5 of file BATCHgdwm.h.

5.9.2 Member Function Documentation

5.9.2.1 void BATCHgdwm::adjustParameters () [virtual]

Implements [BatchBehavior](#).

5.9.3 Member Data Documentation

5.9.3.1 double BATCHgdwm::outputDerivative [private]

Definition at line 8 of file BATCHgdwm.h.

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

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

5.10 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

5.10.1 Detailed Description

class [Con](#) -

Definition at line 3 of file Connection.h.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 Con::Con ([Neuron](#) & *neuron*)

Constructor.

Definition at line 20 of file Connection.cpp.

```

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

```

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

Constructor.

Definition at line 31 of file Connection.cpp.

```

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

```

5.10.3 Member Function Documentation

5.10.3.1 [Neuron](#) & Con::getNeuron ()

from field accessor.

This method allows access to the address stored in the private from field (a pointer to a [Neuron](#) object).*

Returns

A pointer to the [Neuron](#) object referred to by the from field.

```

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

```

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

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

See also

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

Definition at line 57 of file `Connection.cpp`.

References `d_neuron`.

```
{
    return d_neuron;
}
```

5.10.3.2 double Con::getWeight ()

weight field accessor.

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

Returns

The value of `weight` (double)

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

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

See also

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

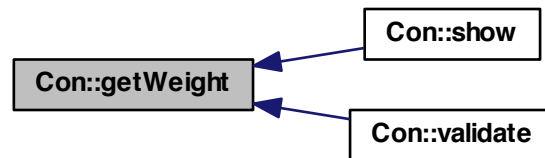
Definition at line 117 of file `Connection.cpp`.

References `d_weight`.

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

```
{
    return d_weight;
}
```

Here is the caller graph for this function:



5.10.3.3 int Con::Id ()

A getter of the Id of the [Neuron](#) pointed by the from field.

This method gets the Id of the [Neuron](#) referred to by the from field

Returns

The value of the Id (an integer).

```

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

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

See also

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

Definition at line 89 of file `Connection.cpp`.

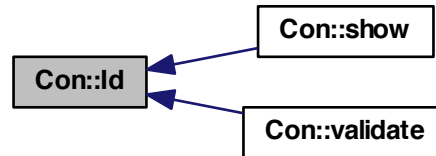
References `d_neuron`.

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

```

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

Here is the caller graph for this function:



5.10.3.4 void Con::setNeuron (Neuron & neuron)

Definition at line 64 of file Connection.cpp.

References `d_neuron`.

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

5.10.3.5 void Con::setWeight (double weight)

Definition at line 124 of file Connection.cpp.

References `d_weight`.

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

5.10.3.6 void Con::show ()

Pretty print of the [Con](#) information.

This method outputs in the R terminal the contents of the [Con](#) fields.

Returns

true in case everything works without throwing an exception

See also

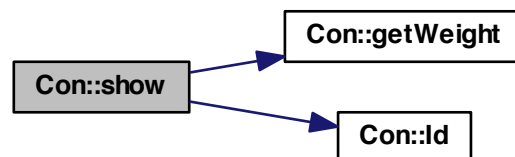
[setWeight](#) and the unit test files, e.g., `runit.Cpp.Con.R`, for usage examples.

Definition at line 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:



5.10.3.7 bool Con::validate ()

Object validator.

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

Returns

true in case the checks are Ok.

Exceptions

<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()`.

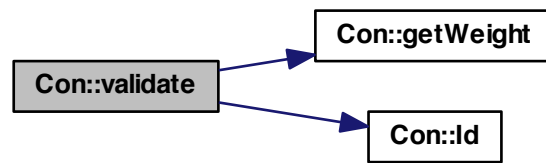
```
{
```

```

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:



5.10.4 Member Data Documentation

5.10.4.1 NeuronRef Con::d_neuron [private]

Definition at line 6 of file Connection.h.

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

5.10.4.2 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:

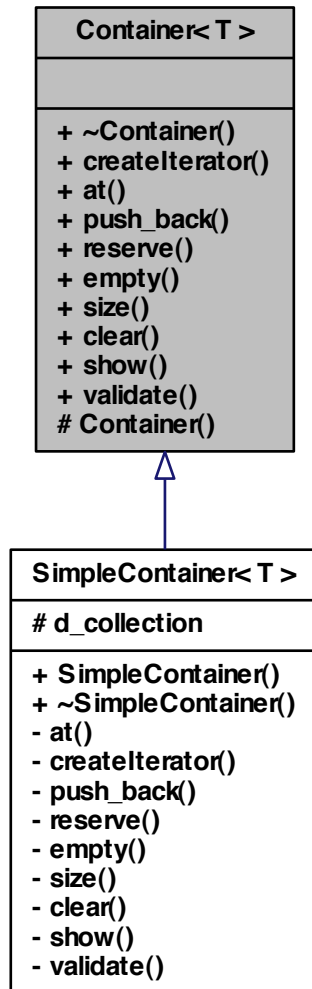
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[Connection](#)

5.11 Container< T > Class Template Reference

class [Container](#) -

```
#include <Container.h>
```

Inheritance diagram for Container< T >:



Public Member Functions

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

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

Protected Member Functions

- [Container](#) ()

5.11.1 Detailed Description

template<typename T>class Container< T >

class [Container](#) -

Definition at line 5 of file Container.h.

5.11.2 Constructor & Destructor Documentation

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

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

5.11.3 Member Function Documentation

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

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

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

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

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

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

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

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

5.11.3.5 `template<typename T> virtual void Container< T >::push_back (T const & const.reference)` [pure virtual]

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

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

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

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

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

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

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

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

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

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

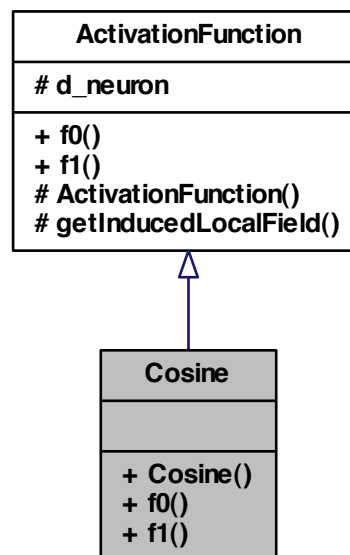
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Container`

5.12 Cosine Class Reference

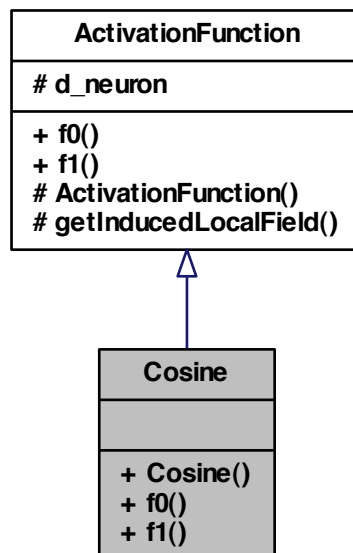
class [Cosine](#) -

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

Inheritance diagram for Cosine:



Collaboration diagram for Cosine:



Public Member Functions

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

5.12.1 Detailed Description

class [Cosine](#) -

Definition at line 5 of file Cosine.h.

5.12.2 Constructor & Destructor Documentation

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

5.12.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

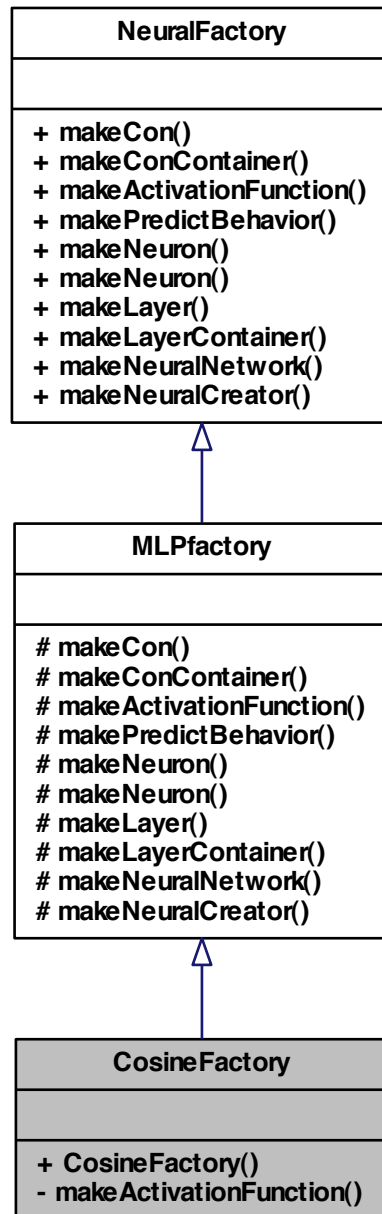
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

5.13 CosineFactory Class Reference

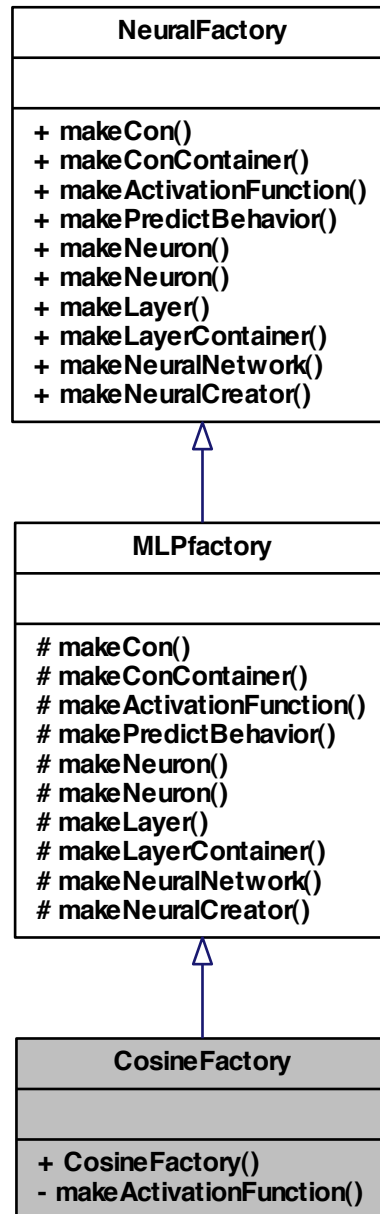
class [CosineFactory](#) -

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

Inheritance diagram for CosineFactory:



Collaboration diagram for CosineFactory:



Public Member Functions

- [CosineFactory](#) ()

Private Member Functions

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

5.13.1 Detailed Description

class [CosineFactory](#) -

Definition at line 5 of file CosineFactory.h.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 [CosineFactory::CosineFactory](#) ()

5.13.3 Member Function Documentation

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

Implements [MLPfactory](#).

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

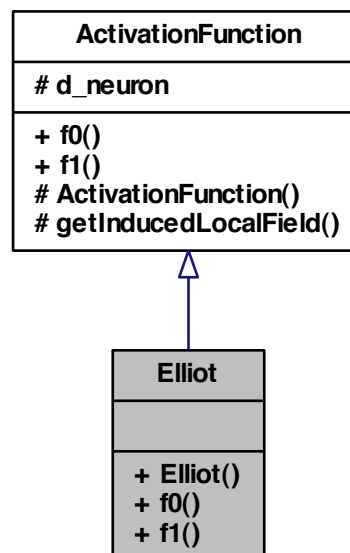
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[CosineFa](#)

5.14 Elliot Class Reference

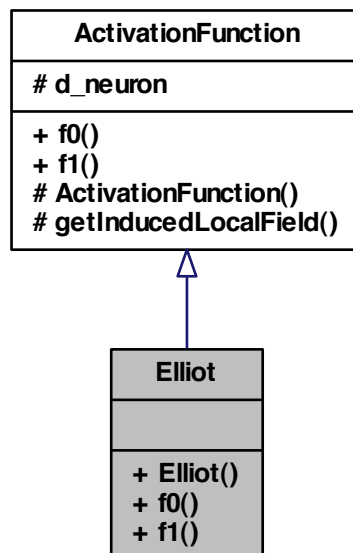
class [Elliot](#) -

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

Inheritance diagram for Elliot:



Collaboration diagram for Elliot:



Public Member Functions

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

5.14.1 Detailed Description

class [Elliot](#) -

Definition at line 5 of file Elliot.h.

5.14.2 Constructor & Destructor Documentation

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

5.14.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

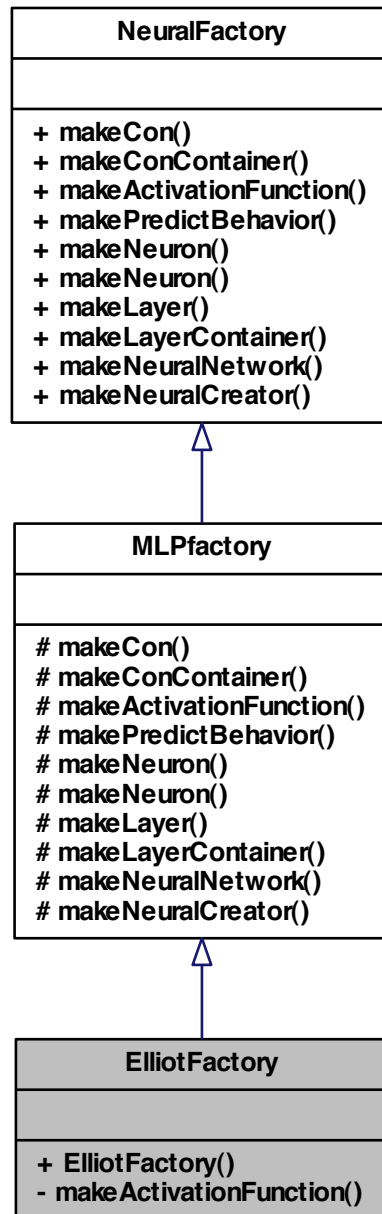
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

5.15 ElliotFactory Class Reference

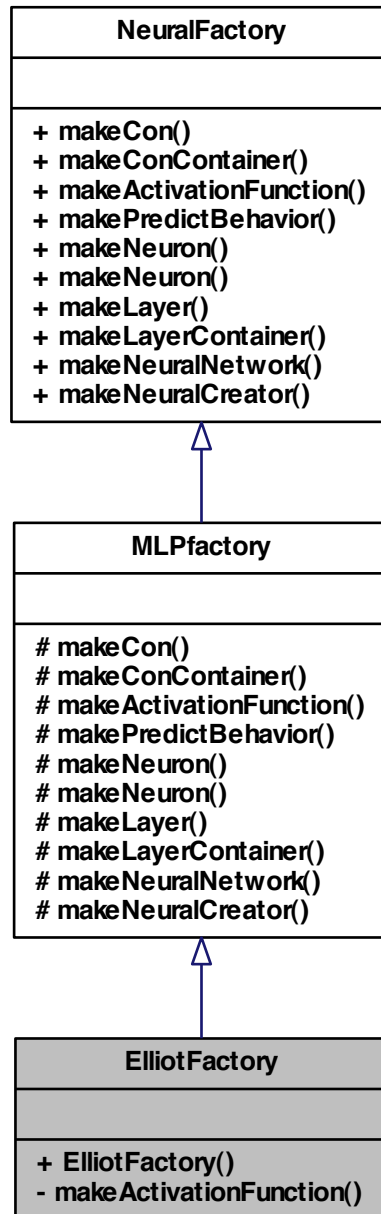
class [ElliotFactory](#) -

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

Inheritance diagram for ElliotFactory:



Collaboration diagram for ElliotFactory:



Public Member Functions

- [ElliotFactory](#) ()

Private Member Functions

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

5.15.1 Detailed Description

class [ElliotFactory](#) -

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

5.15.2 Constructor & Destructor Documentation

5.15.2.1 [ElliotFactory::ElliotFactory](#) ()

5.15.3 Member Function Documentation

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

Implements [MLPfactory](#).

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

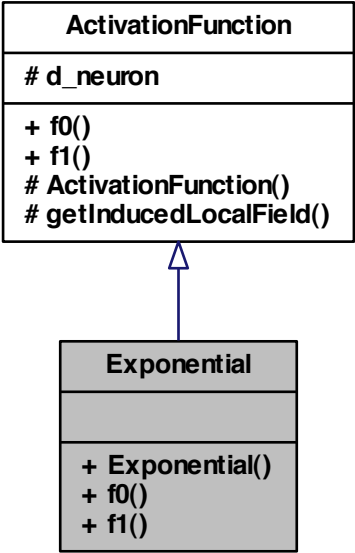
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/\[ElliotFactory.h\]\(#\)](#)

5.16 Exponential Class Reference

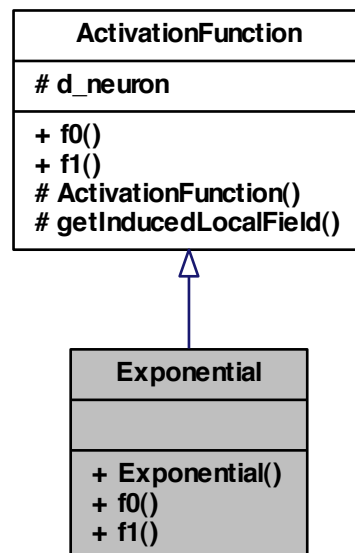
class [Exponential](#) -

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

Inheritance diagram for Exponential:



Collaboration diagram for Exponential:



Public Member Functions

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

5.16.1 Detailed Description

class [Exponential](#) -

Definition at line 5 of file Exponential.h.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 [Exponential::Exponential](#) ([NeuronPtr](#) neuronPtr)

5.16.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

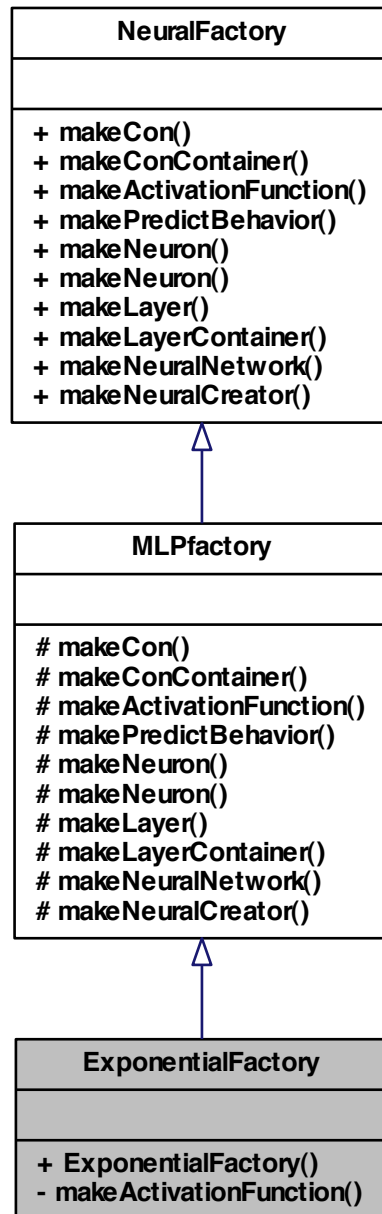
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

5.17 ExponentialFactory Class Reference

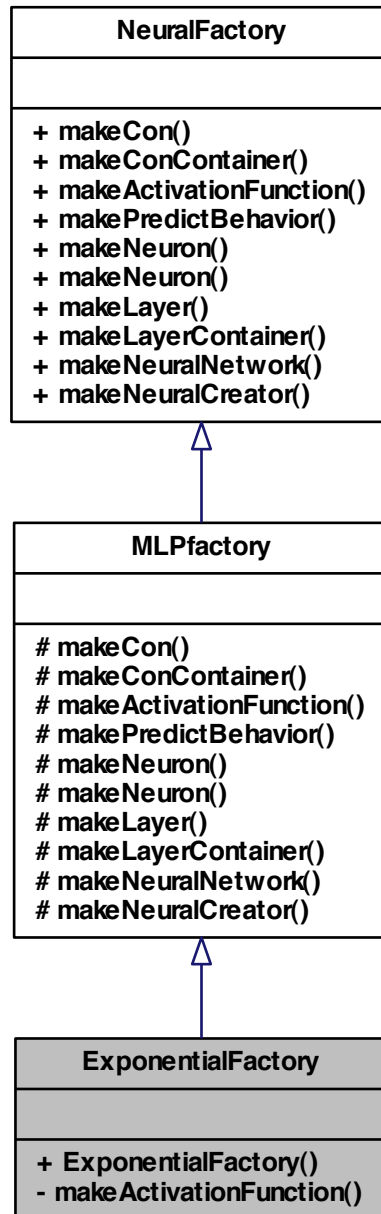
class [ExponentialFactory](#) -

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


Inheritance diagram for ExponentialFactory:



Collaboration diagram for ExponentialFactory:



Public Member Functions

- [ExponentialFactory](#) ()

Private Member Functions

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

5.17.1 Detailed Description

class [ExponentialFactory](#) -

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

5.17.2 Constructor & Destructor Documentation

5.17.2.1 [ExponentialFactory::ExponentialFactory](#) ()

5.17.3 Member Function Documentation

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

Implements [MLPfactory](#).

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

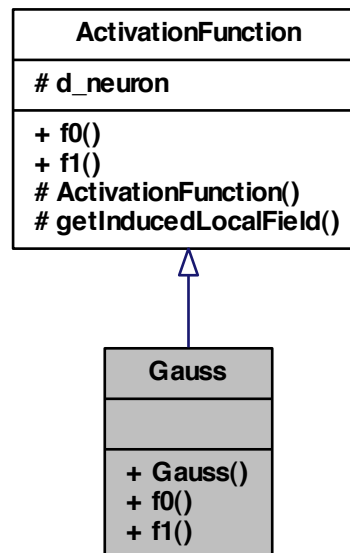
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Exponent](#)

5.18 Gauss Class Reference

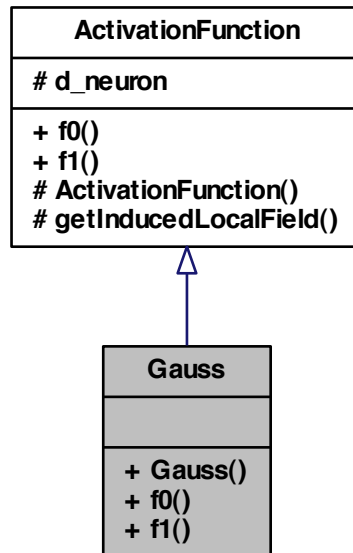
class [Gauss](#) -

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

Inheritance diagram for Gauss:



Collaboration diagram for Gauss:



Public Member Functions

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

5.18.1 Detailed Description

class [Gauss](#) -

Definition at line 5 of file Gauss.h.

5.18.2 Constructor & Destructor Documentation

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

5.18.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

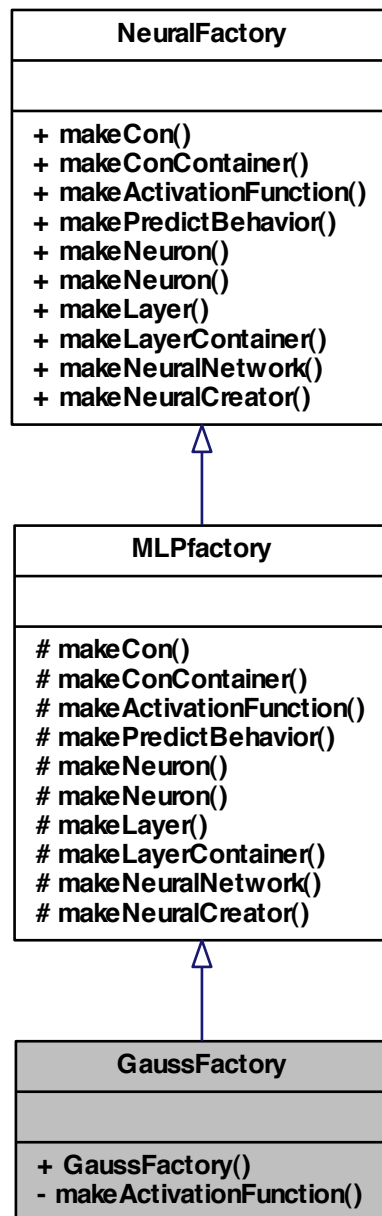
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

5.19 GaussFactory Class Reference

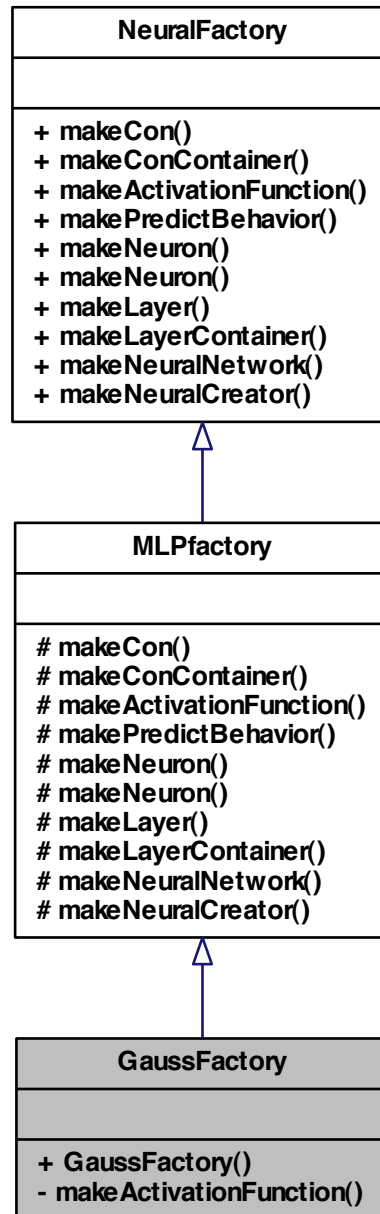
class [GaussFactory](#) -

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

Inheritance diagram for GaussFactory:



Collaboration diagram for GaussFactory:



Public Member Functions

- [GaussFactory](#) ()

Private Member Functions

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

5.19.1 Detailed Description

class [GaussFactory](#) -

Definition at line 5 of file GaussFactory.h.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 [GaussFactory::GaussFactory](#) ()

5.19.3 Member Function Documentation

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

Implements [MLPfactory](#).

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

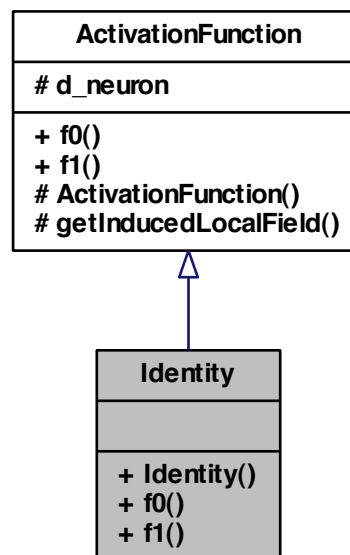
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[GaussFactory.h](#)

5.20 Identity Class Reference

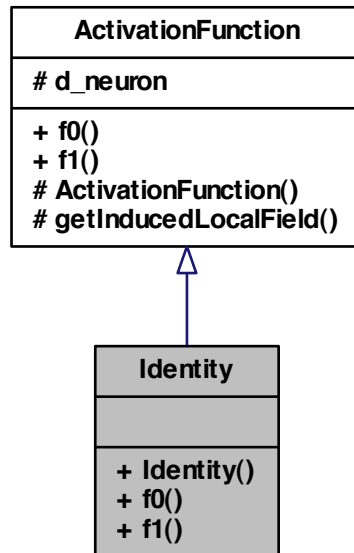
class [Identity](#) -

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

Inheritance diagram for Identity:



Collaboration diagram for Identity:



Public Member Functions

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

5.20.1 Detailed Description

class [Identity](#) -

Definition at line 5 of file Identity.h.

5.20.2 Constructor & Destructor Documentation

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

Definition at line 13 of file Identity.cpp.

```

: ActivationFunction(neuronPtr) {

```

```
}
```

5.20.3 Member Function Documentation

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

Implements [ActivationFunction](#).

Definition at line 17 of file Identity.cpp.

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

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

Here is the call graph for this function:



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

Implements [ActivationFunction](#).

Definition at line 21 of file Identity.cpp.

```
    {  
    return 1 ;  
    }
```

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

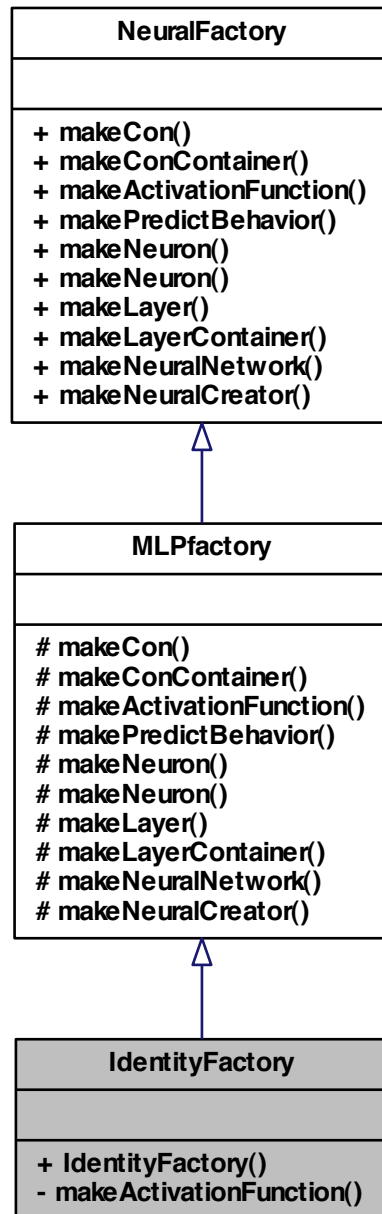
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead`
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Identity.cpp`

5.21 IdentityFactory Class Reference

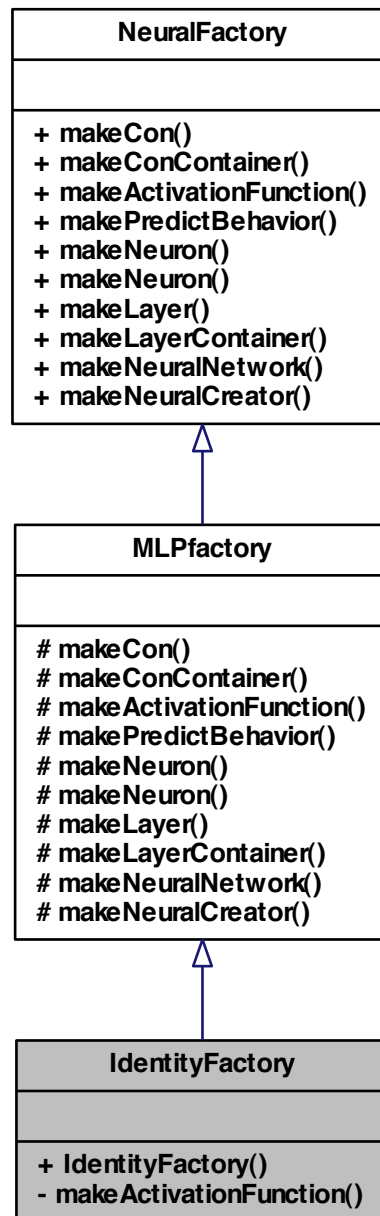
class [IdentityFactory](#) -

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

Inheritance diagram for IdentityFactory:



Collaboration diagram for IdentityFactory:



Public Member Functions

- [IdentityFactory](#) ()

Private Member Functions

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

5.21.1 Detailed Description

class [IdentityFactory](#) -

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

5.21.2 Constructor & Destructor Documentation

5.21.2.1 `IdentityFactory::IdentityFactory ()`

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

```
{  
}
```

5.21.3 Member Function Documentation

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

Implements [MLPfactory](#).

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

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

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

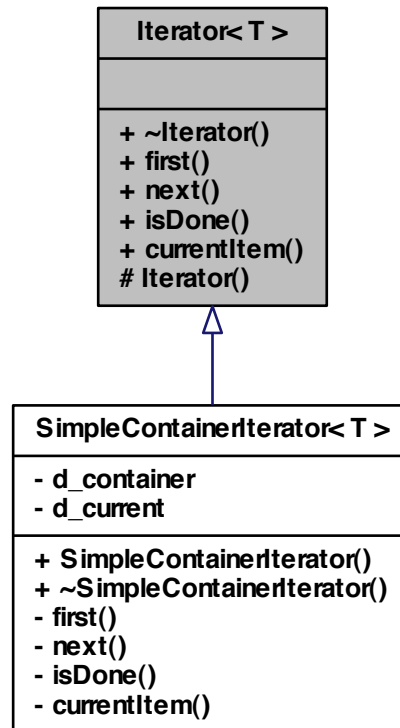
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/IdentityFa](#)
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/IdentityFactory.cpp](#)

5.22 `Iterator< T >` Class Template Reference

class [Iterator](#) -

```
#include <Iterator.h>
```

Inheritance diagram for `Iterator< T >`:



Public Member Functions

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

Protected Member Functions

- `Iterator()`

5.22.1 Detailed Description

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

class [Iterator](#) -

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

5.22.2 Constructor & Destructor Documentation

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

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

5.22.3 Member Function Documentation

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

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

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

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

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

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

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

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

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

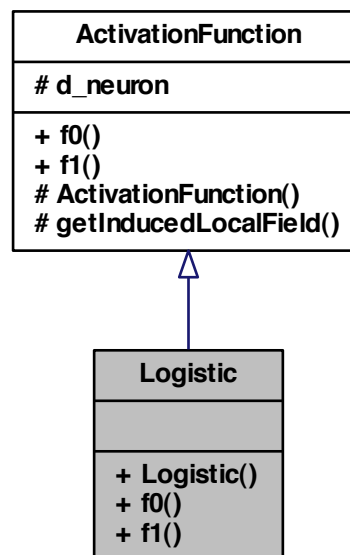
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Iterator.h`

5.23 Logistic Class Reference

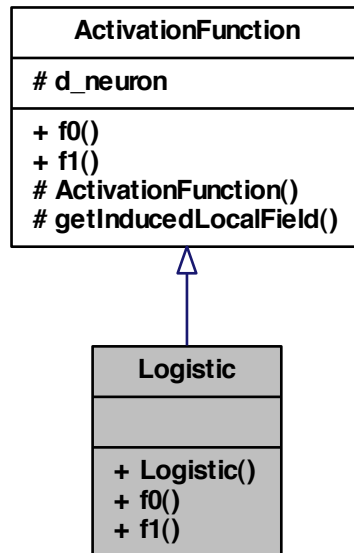
class [Logistic](#) -

`#include <Logistic.h>`

Inheritance diagram for Logistic:



Collaboration diagram for Logistic:



Public Member Functions

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

5.23.1 Detailed Description

class [Logistic](#) -

Definition at line 5 of file Logistic.h.

5.23.2 Constructor & Destructor Documentation

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

5.23.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

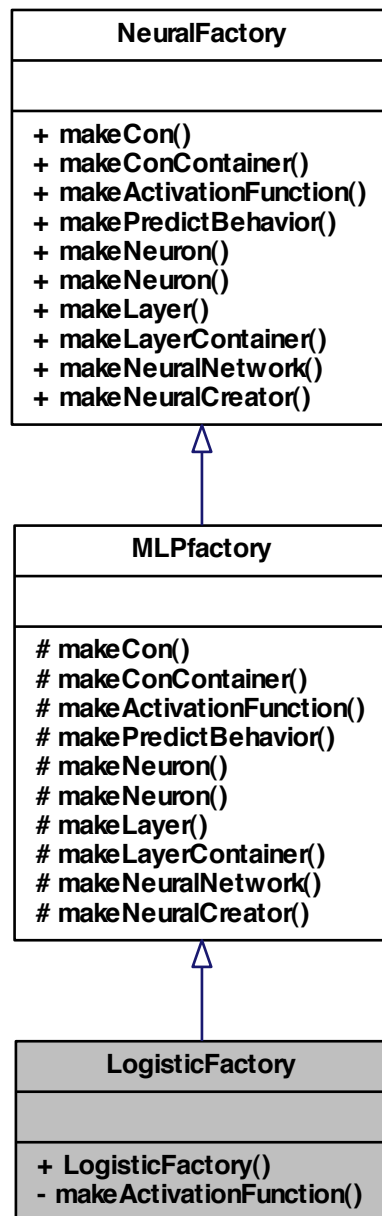
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead`

5.24 LogisticFactory Class Reference

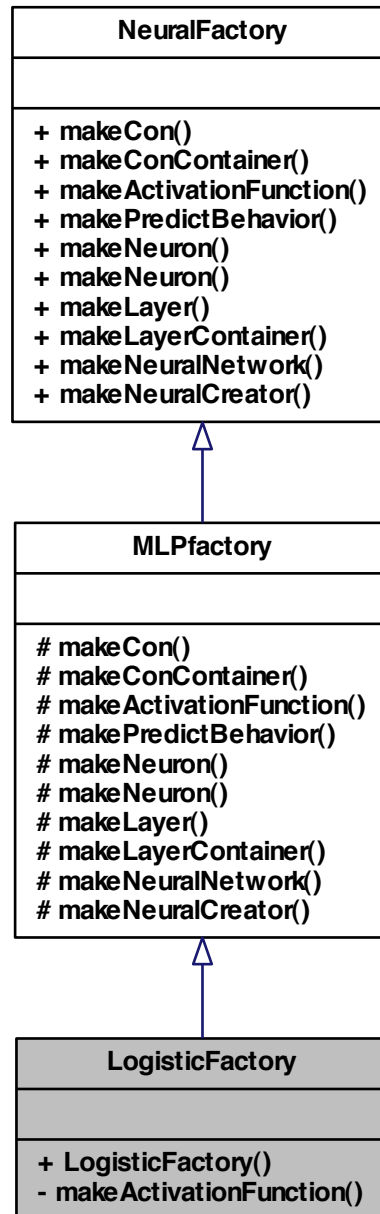
class [LogisticFactory](#) -

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

Inheritance diagram for LogisticFactory:



Collaboration diagram for LogisticFactory:



Public Member Functions

- [LogisticFactory](#) ()

Private Member Functions

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

5.24.1 Detailed Description

class [LogisticFactory](#) -

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

5.24.2 Constructor & Destructor Documentation

5.24.2.1 [LogisticFactory::LogisticFactory](#) ()

5.24.3 Member Function Documentation

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

Implements [MLPfactory](#).

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

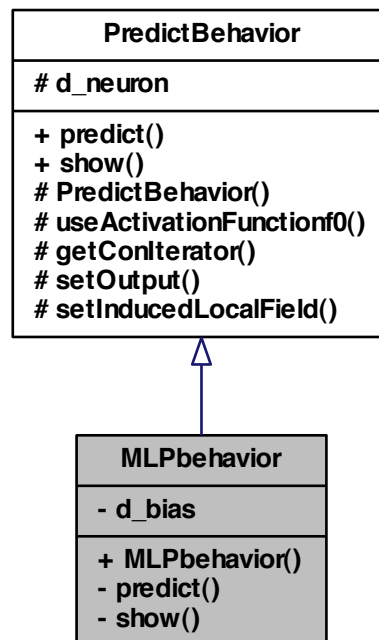
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticF](#)

5.25 MLPbehavior Class Reference

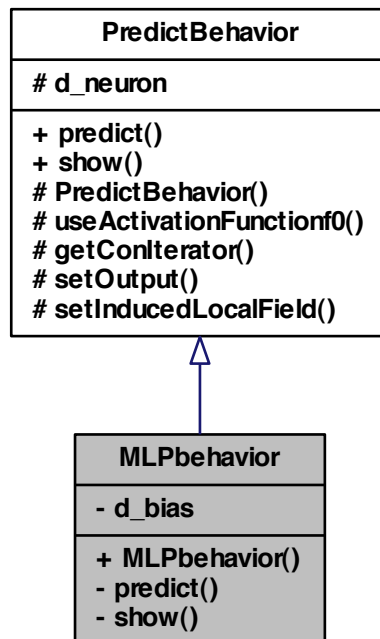
class [MLPbehavior](#) -

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

Inheritance diagram for MLPbehavior:



Collaboration diagram for MLPbehavior:



Public Member Functions

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

Private Member Functions

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

Private Attributes

- double [d_bias](#)

Friends

- class [MLPfactory](#)

5.25.1 Detailed Description

class [MLPbehavior](#) -

Definition at line 5 of file MLPbehavior.h.

5.25.2 Constructor & Destructor Documentation

5.25.2.1 MLPbehavior::MLPbehavior ([NeuronPtr neuronPtr](#))

Definition at line 17 of file MLPbehavior.cpp.

```

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

5.25.3 Member Function Documentation

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

Implements [PredictBehavior](#).

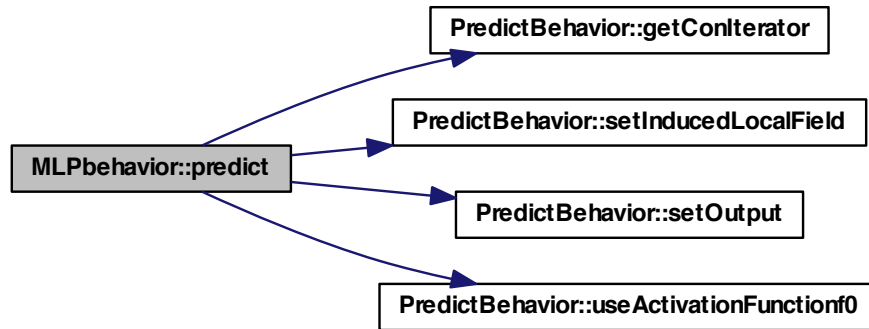
Definition at line 23 of file MLPbehavior.cpp.

References [d_bias](#), [PredictBehavior::getConIterator\(\)](#), [PredictBehavior::setInducedLocalField\(\)](#), [PredictBehavior::setOutput\(\)](#), and [PredictBehavior::useActivationFunction0\(\)](#).

```

{
    double accumulator(d_bias);
    ConIteratorPtr conIterator = getConIterator();
    double weight;
    double incomingSignalValue;
    for (conIterator->first(); !conIterator->isDone(); conIterator->next())
    {
        weight = conIterator->currentItem()->getWeight();
        incomingSignalValue = conIterator->currentItem()->getNeuron().getOutput();
        accumulator += weight * incomingSignalValue;
    }
    setInducedLocalField(accumulator);
    setOutput (
        useActivationFunction0());
}
```

Here is the call graph for this function:



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

Implements [PredictBehavior](#).

Definition at line 42 of file `MLPbehavior.cpp`.

References `d_bias`.

```

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

```

5.25.4 Friends And Related Function Documentation

5.25.4.1 friend class MLPfactory [friend]

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

5.25.5 Member Data Documentation

5.25.5.1 double MLPbehavior::d_bias [private]

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

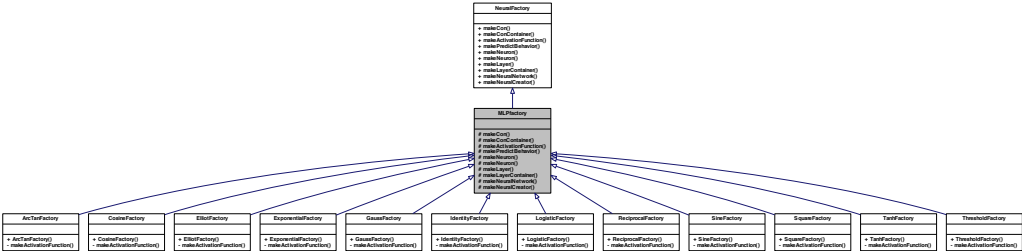
Referenced by `MLPfactory::makeNeuron()`, `predict()`, and `show()`.

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

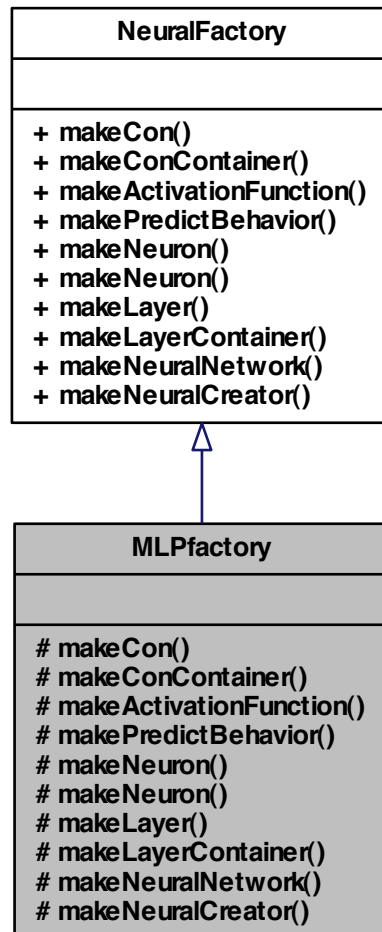
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPbehavi

5.26 MLPfactory Class Reference

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



Collaboration diagram for MLPfactory:



Protected Member Functions

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

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

5.26.1 Detailed Description

class [MLPfactory](#) -

Definition at line 5 of file MLPfactory.h.

5.26.2 Member Function Documentation

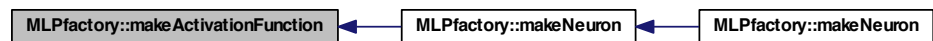
5.26.2.1 `virtual ActivationFunctionPtr MLPfactory::makeActivationFunction (NeuronPtr neuronPtr)` [protected, pure virtual]

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:



5.26.2.2 `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:



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

Implements [NeuralFactory](#).

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

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

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

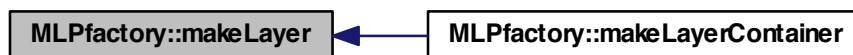
Implements [NeuralFactory](#).

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:



5.26.2.5 **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:



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

Implements [NeuralFactory](#).

Definition at line 109 of file MLPfactory.cpp.

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

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

Implements [NeuralFactory](#).

Definition at line 101 of file MLPfactory.cpp.

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


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

Implements [NeuralFactory](#).

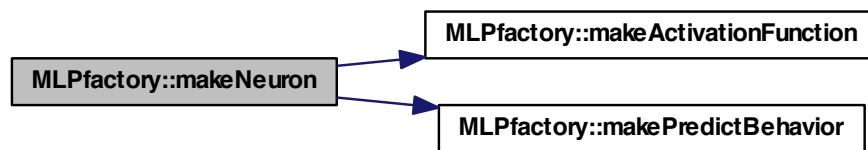
Definition at line 52 of file MLPfactory.cpp.

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:



5.26.2.9 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;

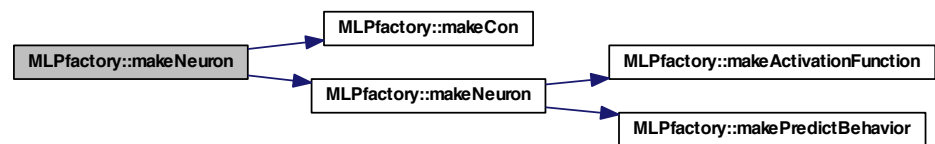
    NeuronPtr neuronPtr(makeNeuron(Id));

    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:



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

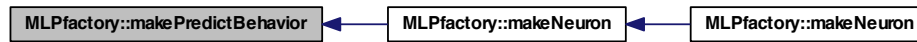
Implements [NeuralFactory](#).

Definition at line 45 of file MLPfactory.cpp.

Referenced by makeNeuron().

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

Here is the caller graph for this function:



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

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

5.27 NetworkRinterface Class Reference

class [NetworkRinterface](#) -

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

Public Member Functions

- [NetworkRinterface](#) ()
- void [createFeedForwardNetwork](#) (Rcpp::NumericVector numberOfNeurons)
- Rcpp::NumericMatrix [predict](#) (Rcpp::NumericMatrix numericMatrix)
- size_type [inputSize](#) ()
- size_type [outputSize](#) ()
- void [show](#) ()
- bool [validate](#) ()

Private Attributes

- [NeuralNetworkPtr](#) `d_neuralNetwork`

5.27.1 Detailed Description

class [NetworkRinterface](#) -

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

5.27.2 Constructor & Destructor Documentation

5.27.2.1 NetworkRinterface::NetworkRinterface ()

Definition at line 22 of file `NetworkRinterface.cpp`.

```
{
}
```

5.27.3 Member Function Documentation

5.27.3.1 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:



5.27.3.2 size_type NetworkRinterface::inputSize ()

Definition at line 70 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:



5.27.3.3 `size_type NetworkRinterface::outputSize ()`

Definition at line 76 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:



5.27.3.4 `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
    if (!d_neuralNetwork)
    {
        throw std::runtime_error( "\nUninitialized network. Please use any of the c
  
```

```

        reate methods available.\n");
    }

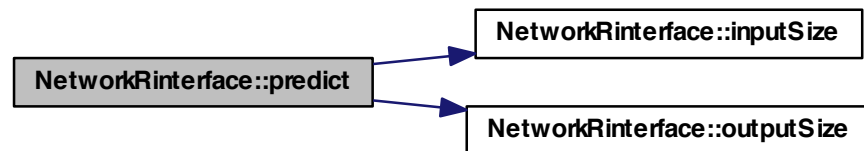
    bool checkIncorrectNumberOfRows(
        inputSize() != static_cast<size_type> (numericMatrix.nrow()));
    if (checkIncorrectNumberOfRows)
    {
        throw std::runtime_error(
            "\nIncorrect number or rows. The number of input neurons must be equal
            to the number of rows of the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

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

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

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.27.3.5 void NetworkRinterface::show ()

Definition at line 82 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:



5.27.3.6 bool NetworkRinterface::validate ()

Definition at line 97 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:



5.27.4 Member Data Documentation

5.27.4.1 NeuralNetworkPtr NetworkRinterface::d_neuralNetwork [private]

Definition at line 6 of file NetworkRinterface.h.

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

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

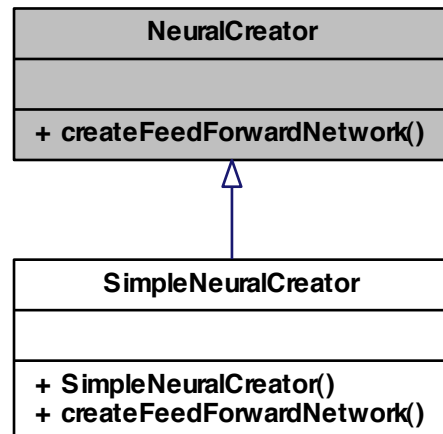
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[NetworkRin](#)

5.28 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

5.28.1 Detailed Description

class [NeuralCreator](#) -

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

5.28.2 Member Function Documentation

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

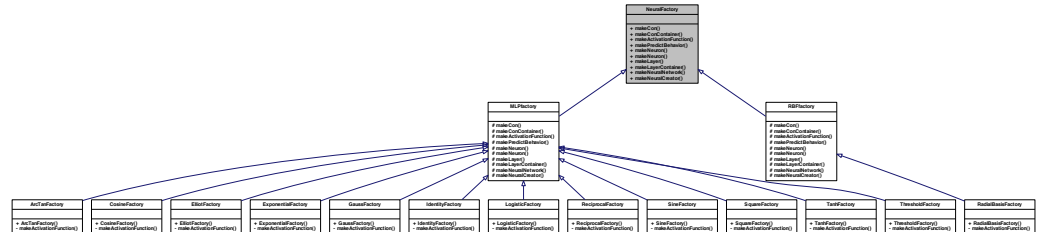
Implemented in [SimpleNeuralCreator](#).

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

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

class **NeuralFactory** -

Inheritance diagram for NeuralFactory:



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

class **NeuralFactory** -

5.29.2 Member Function Documentation

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

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

Implemented in [MLPfactory](#).

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

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

Referenced by `Neuron::Neuron()`.

Here is the caller graph for this function:

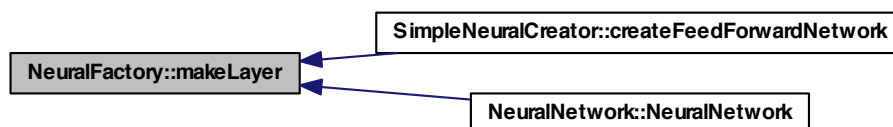


5.29.2.4 `virtual LayerPtr NeuralFactory::makeLayer ()` [pure virtual]

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

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

Here is the caller graph for this function:



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

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

Referenced by `NeuralNetwork::NeuralNetwork()`.

Here is the caller graph for this function:



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

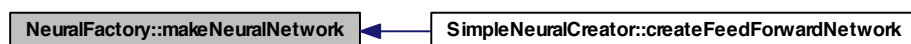
Implemented in [MLPfactory](#), and [RBFfactory](#).

5.29.2.7 `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:



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

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

Referenced by `SimpleNeuralCreator::createFeedForwardNetwork()`.

Here is the caller graph for this function:



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

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

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

Implemented in [MLPfactory](#).

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

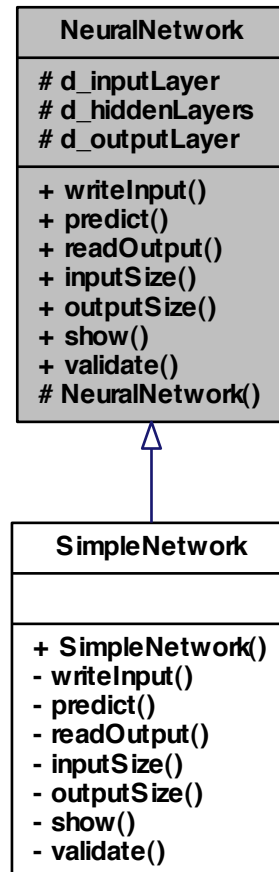
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralFactory.h`

5.30 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 [predict](#) ()=0
- virtual void [readOutput](#) (std::vector< double >::iterator &iterator)=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

Friends

- class [SimpleNeuralCreator](#)

5.30.1 Detailed Description

class [NeuralNetwork](#) -

Definition at line 3 of file NeuralNetwork.h.

5.30.2 Constructor & Destructor Documentation

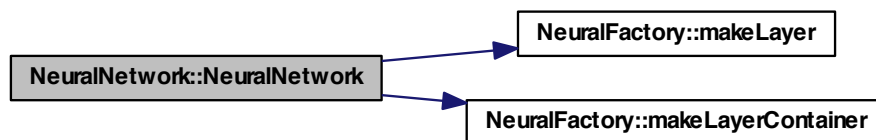
5.30.2.1 [NeuralNetwork::NeuralNetwork](#) ([NeuralFactory](#) & *neuralFactory*) [protected]

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:



5.30.3 Member Function Documentation

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

Implemented in [SimpleNetwork](#).

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

Implemented in [SimpleNetwork](#).

5.30.3.3 `virtual void NeuralNetwork::predict ()` [pure virtual]

Implemented in [SimpleNetwork](#).

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

Implemented in [SimpleNetwork](#).

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

Implemented in [SimpleNetwork](#).

5.30.3.6 `virtual bool NeuralNetwork::validate ()` [pure virtual]

Implemented in [SimpleNetwork](#).

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

Implemented in [SimpleNetwork](#).

5.30.4 Friends And Related Function Documentation

5.30.4.1 `friend class SimpleNeuralCreator` [friend]

Definition at line 11 of file NeuralNetwork.h.

5.30.5 Member Data Documentation

5.30.5.1 `LayerContainerPtr NeuralNetwork::d_hiddenLayers` [protected]

Definition at line 7 of file NeuralNetwork.h.

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

5.30.5.2 `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()`.

5.30.5.3 `LayerPtr NeuralNetwork::d_outputLayer` [protected]

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

Referenced by `NeuralNetwork()`, `SimpleNetwork::outputSize()`, `SimpleNetwork::predict()`, `SimpleNetwork::readOutput()`, `SimpleNetwork::show()`, and `SimpleNetwork::validate()`.

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

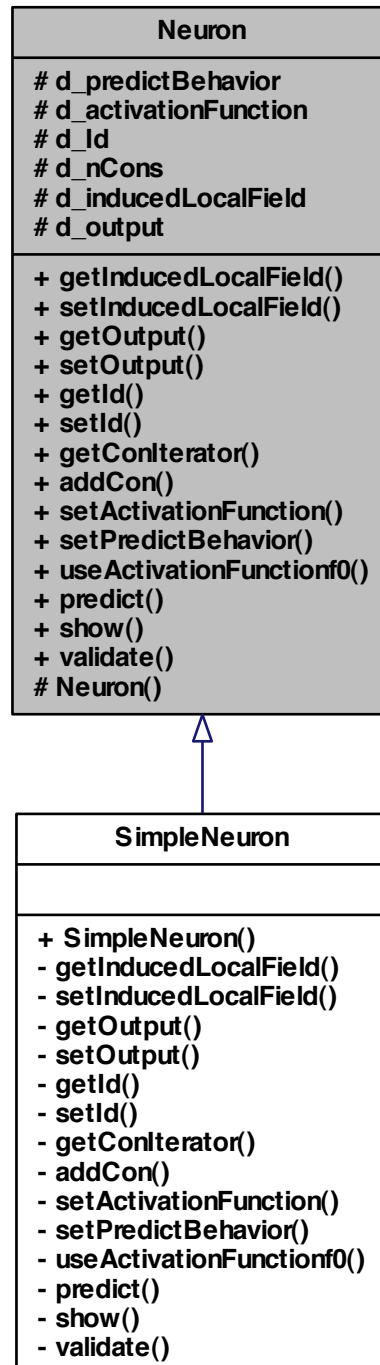
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralNe`
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NeuralNetwork.cpp`

5.31 Neuron Class Reference

class `Neuron` -

```
#include <Neuron.h>
```

Inheritance diagram for Neuron:



Public Member Functions

- virtual double [getInducedLocalField](#) ()=0
- virtual void [setInducedLocalField](#) (double inducedLocalField)=0
- virtual double [getOutput](#) ()=0
- virtual void [setOutput](#) (double output)=0
- virtual [Handler](#) [getId](#) ()=0
- virtual void [setId](#) ([Handler](#) Id)=0
- virtual [ConIteratorPtr](#) [getConIterator](#) ()=0
- virtual void [addCon](#) ([ConPtr](#) conPtr)=0
- virtual void [setActivationFunction](#) ([ActivationFunctionPtr](#) activationFunctionPtr)=0
- virtual void [setPredictBehavior](#) ([PredictBehaviorPtr](#) predictBehaviorPtr)=0
- virtual double [useActivationFunction0](#) ()=0
- virtual void [predict](#) ()=0
- virtual void [show](#) ()=0
- virtual bool [validate](#) ()=0

Protected Member Functions

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

Protected Attributes

- [PredictBehaviorPtr](#) d_predictBehavior
- [ActivationFunctionPtr](#) d_activationFunction
- [Handler](#) d_Id
- [ConContainerPtr](#) d_nCons
- double d_inducedLocalField
- double d_output

Friends

- class [MLPfactory](#)

5.31.1 Detailed Description

class [Neuron](#) -

Definition at line 3 of file Neuron.h.

5.31.2 Constructor & Destructor Documentation

5.31.2.1 `Neuron::Neuron (NeuralFactory & neuralFactory)` `[protected]`

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

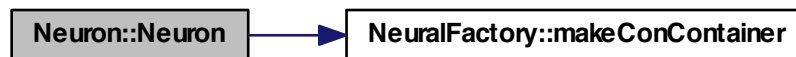
References `d_nCons`, and `NeuralFactory::makeConContainer()`.

```

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

```

Here is the call graph for this function:



5.31.3 Member Function Documentation

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

Implemented in [SimpleNeuron](#).

5.31.3.2 `virtual ConIteratorPtr Neuron::getConIterator ()` `[pure virtual]`

Implemented in [SimpleNeuron](#).

5.31.3.3 `virtual Handler Neuron::getId ()` `[pure virtual]`

Implemented in [SimpleNeuron](#).

5.31.3.4 `virtual double Neuron::getInducedLocalField ()` `[pure virtual]`

Implemented in [SimpleNeuron](#).

5.31.3.5 `virtual double Neuron::getOutput ()` `[pure virtual]`

Implemented in [SimpleNeuron](#).

5.31.3.6 `virtual void Neuron::predict ()` [pure virtual]

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

5.31.4 Friends And Related Function Documentation

5.31.4.1 friend class **MLPfactory** [friend]

Definition at line 15 of file Neuron.h.

5.31.5 Member Data Documentation

5.31.5.1 **ActivationFunctionPtr Neuron::d_activationFunction** [protected]

Definition at line 7 of file Neuron.h.

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

5.31.5.2 **Handler Neuron::d_Id** [protected]

Definition at line 9 of file Neuron.h.

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

5.31.5.3 **double Neuron::d_inducedLocalField** [protected]

Definition at line 11 of file Neuron.h.

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

5.31.5.4 **ConContainerPtr Neuron::d_nCons** [protected]

Definition at line 10 of file Neuron.h.

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

5.31.5.5 **double Neuron::d_output** [protected]

Definition at line 12 of file Neuron.h.

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

5.31.5.6 **PredictBehaviorPtr Neuron::d_predictBehavior** [protected]

Definition at line 6 of file Neuron.h.

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

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

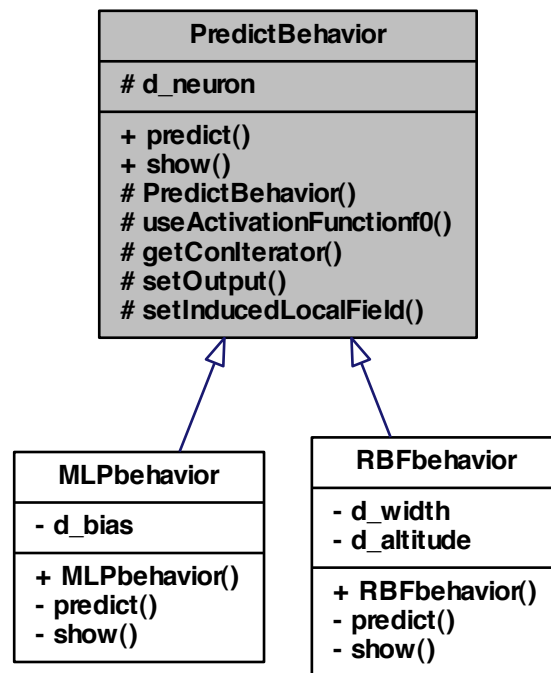
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Neuron.h](#)
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Neuron.cpp](#)

5.32 PredictBehavior Class Reference

class [PredictBehavior](#) -

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

Inheritance diagram for PredictBehavior:



Public Member Functions

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

Protected Member Functions

- [PredictBehavior](#) ([NeuronPtr](#) neuronPtr)
- double [useActivationFunction0](#) ()
- [ConlteratorPtr](#) [getConlterator](#) ()
- void [setOutput](#) (double output)
- void [setInducedLocalField](#) (double inducedLocalField)

Protected Attributes

- [NeuronWeakPtr](#) [d_neuron](#)

5.32.1 Detailed Description

class [PredictBehavior](#) -

Definition at line 4 of file PredictBehavior.h.

5.32.2 Constructor & Destructor Documentation

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

Definition at line 14 of file PredictBehavior.cpp.

```

                                :
{
    d_neuron(neuronPtr)
}

```

5.32.3 Member Function Documentation

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

Definition at line 28 of file PredictBehavior.cpp.

References [d_neuron](#).

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

```

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

```


Here is the caller graph for this function:



5.32.3.2 `virtual void PredictBehavior::predict ()` [pure virtual]

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

5.32.3.3 `void PredictBehavior::setInducedLocalField (double inducedLocalField)`
[protected]

Definition at line 42 of file `PredictBehavior.cpp`.

References `d_neuron`.

Referenced by `MLPbehavior::predict()`.

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

Here is the caller graph for this function:



5.32.3.4 `void PredictBehavior::setOutput (double output)` [protected]

Definition at line 35 of file `PredictBehavior.cpp`.

References `d_neuron`.

Referenced by MLPbehavior::predict().

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

Here is the caller graph for this function:



5.32.3.5 `virtual void PredictBehavior::show ()` [pure virtual]

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

5.32.3.6 `double PredictBehavior::useActivationFunction0 ()` [protected]

Definition at line 20 of file PredictBehavior.cpp.

References `d_neuron`.

Referenced by MLPbehavior::predict().

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

Here is the caller graph for this function:



5.32.4 Member Data Documentation

5.32.4.1 NeuronWeakPtr PredictBehavior::d_neuron [protected]

Definition at line 7 of file PredictBehavior.h.

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

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

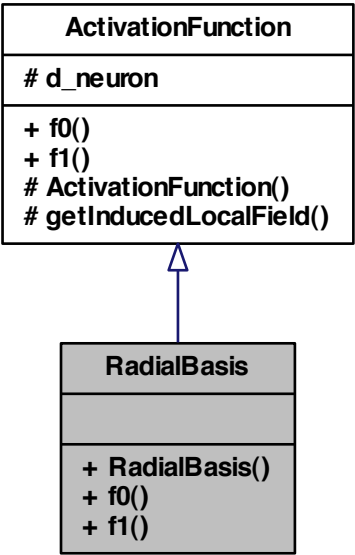
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[PredictBehavior.h](#)
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[PredictBehavior.cpp](#)

5.33 RadialBasis Class Reference

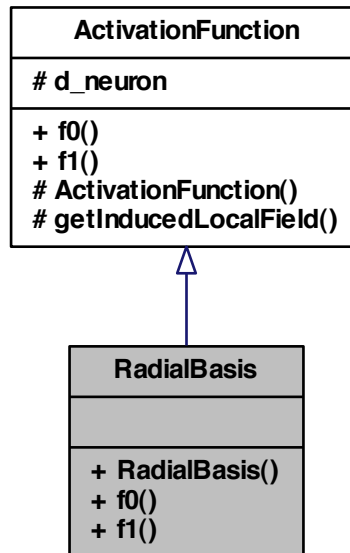
class [RadialBasis](#) -

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

Inheritance diagram for RadialBasis:



Collaboration diagram for RadialBasis:



Public Member Functions

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

5.33.1 Detailed Description

class [RadialBasis](#) -

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

5.33.2 Constructor & Destructor Documentation

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

5.33.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

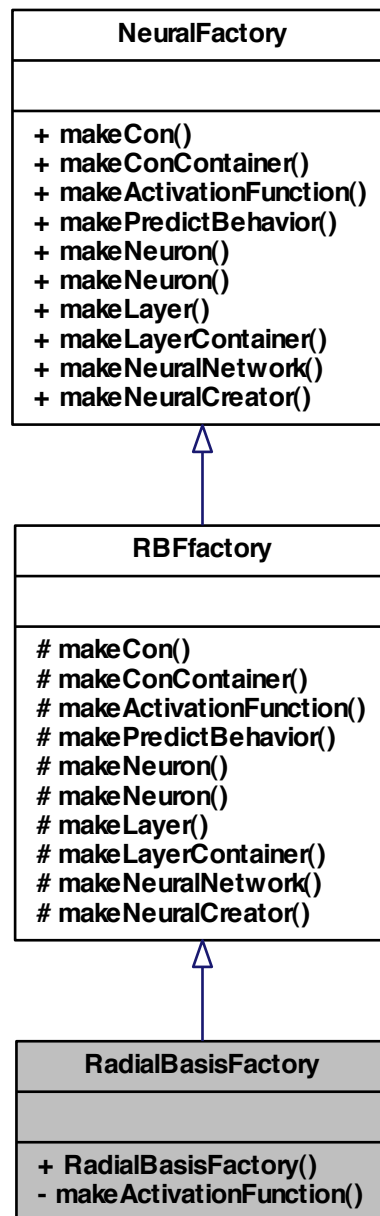
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasisFactory.h`

5.34 RadialBasisFactory Class Reference

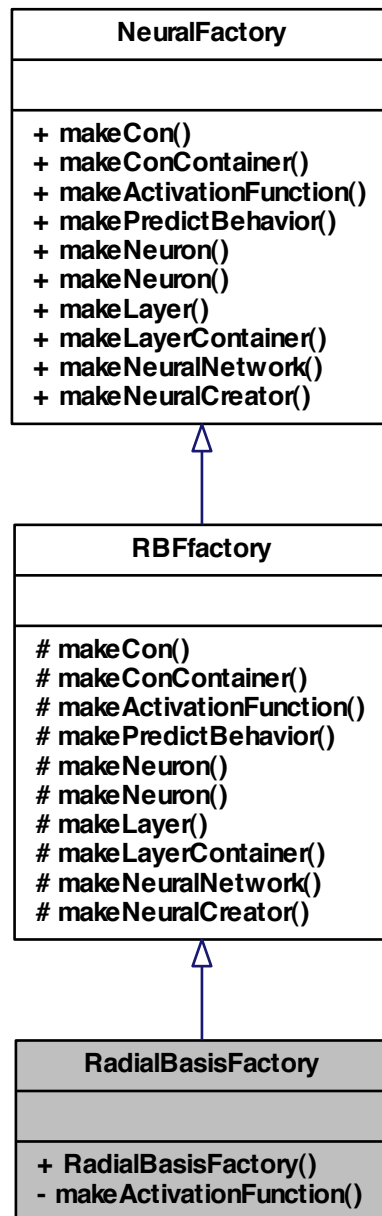
class [RadialBasisFactory](#) -

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

Inheritance diagram for RadialBasisFactory:



Collaboration diagram for RadialBasisFactory:



Public Member Functions

- [RadialBasisFactory](#) ()

Private Member Functions

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

5.34.1 Detailed Description

class [RadialBasisFactory](#) -

Definition at line 5 of file RadialBasisFactory.h.

5.34.2 Constructor & Destructor Documentation

5.34.2.1 [RadialBasisFactory::RadialBasisFactory](#) ()

5.34.3 Member Function Documentation

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

Implements [RBFfactory](#).

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

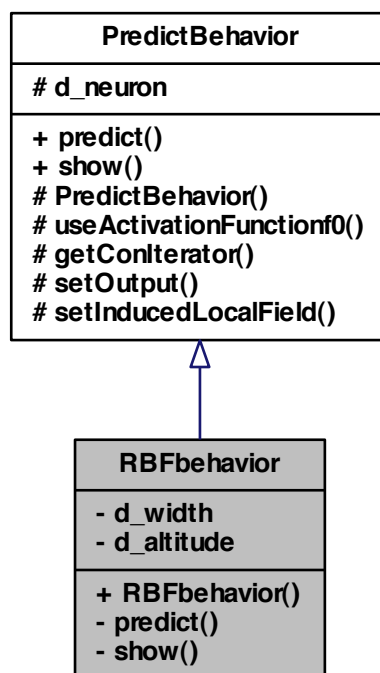
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

5.35 RBFbehavior Class Reference

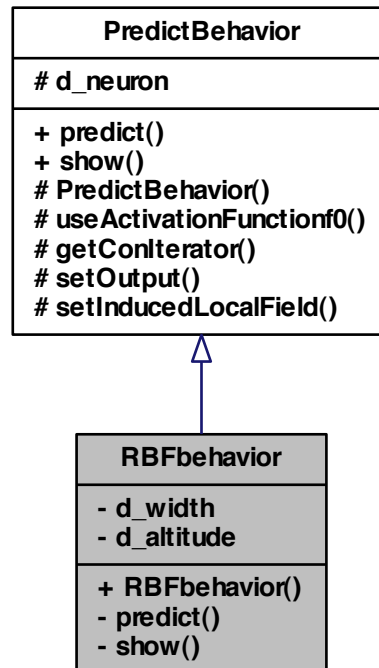
class [RBFbehavior](#) -

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


Inheritance diagram for RBFbehavior:



Collaboration diagram for RBFbehavior:



Public Member Functions

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

Private Member Functions

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

Private Attributes

- double [d_width](#)
- double [d_altitude](#)

5.35.1 Detailed Description

class [RBFbehavior](#) -

Definition at line 5 of file RBFbehavior.h.

5.35.2 Constructor & Destructor Documentation

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

5.35.3 Member Function Documentation

5.35.3.1 [void RBFbehavior::predict](#) () [private, virtual]

Implements [PredictBehavior](#).

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

Implements [PredictBehavior](#).

5.35.4 Member Data Documentation

5.35.4.1 [double RBFbehavior::d_altitude](#) [private]

Definition at line 9 of file RBFbehavior.h.

5.35.4.2 [double RBFbehavior::d_width](#) [private]

Definition at line 8 of file RBFbehavior.h.

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

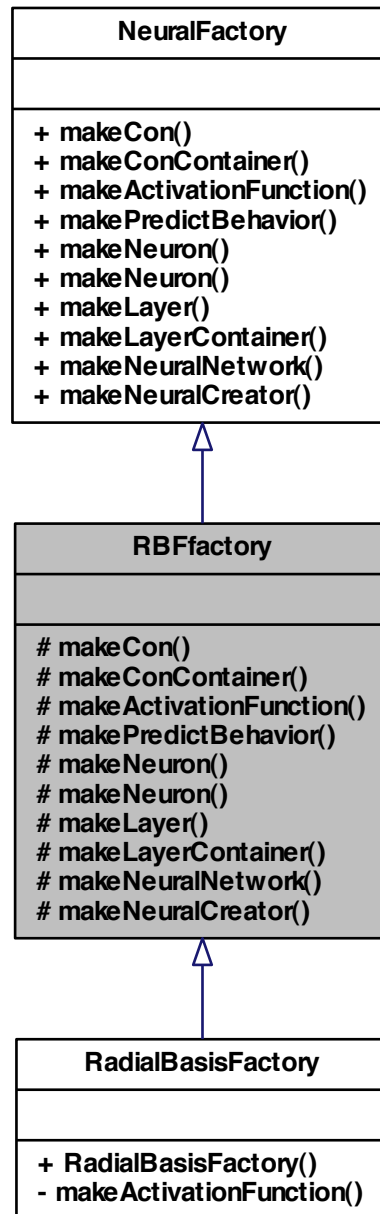
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFbehavior.h](#)

5.36 RBFfactory Class Reference

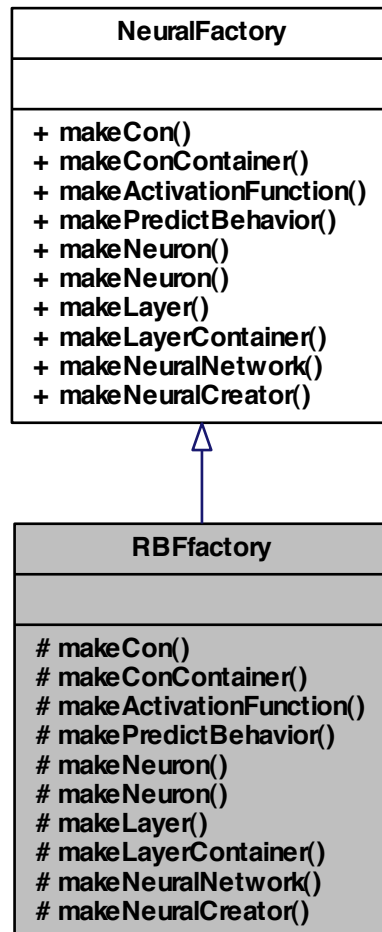
class [RBFfactory](#) -

```
#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 ()`

5.36.1 Detailed Description

class [RBFfactory](#) -

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

5.36.2 Member Function Documentation

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

Implements [NeuralFactory](#).

Implemented in [RadialBasisFactory](#).

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

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

Implements [NeuralFactory](#).

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

Implements [NeuralFactory](#).

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

Implements [NeuralFactory](#).

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

Implements [NeuralFactory](#).

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

Implements [NeuralFactory](#).

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

Implements [NeuralFactory](#).

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

Implements [NeuralFactory](#).

5.36.2.10 `PredictBehaviorPtr RBFfactory::makePredictBehavior ()` [protected]

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

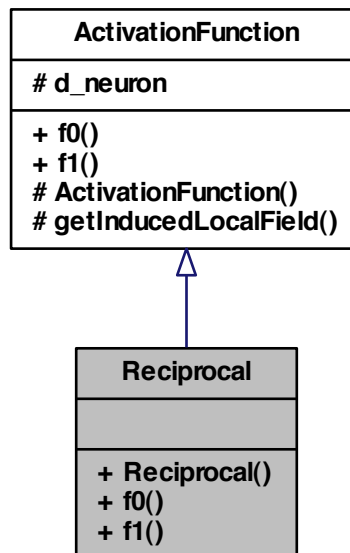
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFfactory.h`

5.37 Reciprocal Class Reference

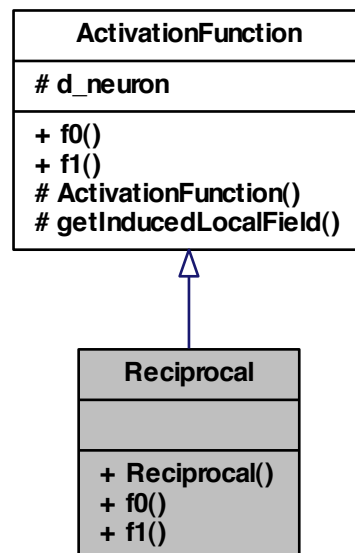
class [Reciprocal](#) -

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

Inheritance diagram for Reciprocal:



Collaboration diagram for Reciprocal:



Public Member Functions

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

5.37.1 Detailed Description

class [Reciprocal](#) -

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

5.37.2 Constructor & Destructor Documentation

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

5.37.3 Member Function Documentation

5.37.3.1 `void Reciprocal::f0 () [virtual]`

Implements [ActivationFunction](#).

5.37.3.2 `void Reciprocal::f1 () [virtual]`

Implements [ActivationFunction](#).

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

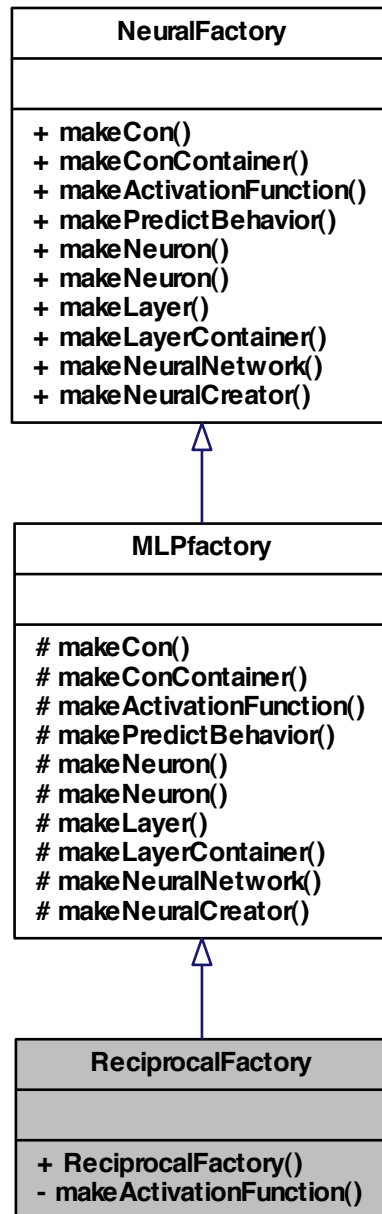
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead`

5.38 ReciprocalFactory Class Reference

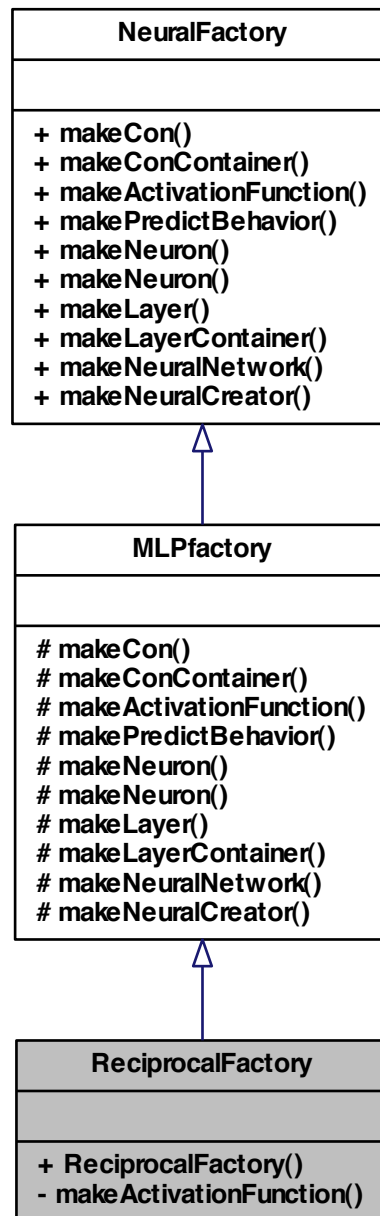
class [ReciprocalFactory](#) -

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

Inheritance diagram for ReciprocalFactory:



Collaboration diagram for ReciprocalFactory:



Public Member Functions

- [ReciprocalFactory](#) ()

Private Member Functions

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

5.38.1 Detailed Description

class [ReciprocalFactory](#) -

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

5.38.2 Constructor & Destructor Documentation

5.38.2.1 [ReciprocalFactory::ReciprocalFactory](#) ()

5.38.3 Member Function Documentation

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

Implements [MLPfactory](#).

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

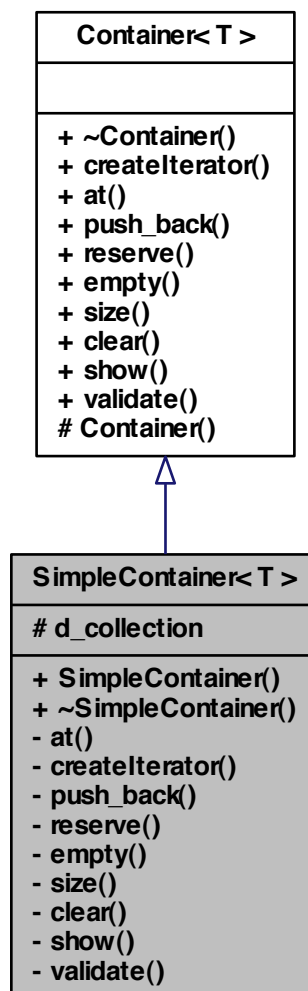
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ReciprocalFactory.h](#)

5.39 SimpleContainer< T > Class Template Reference

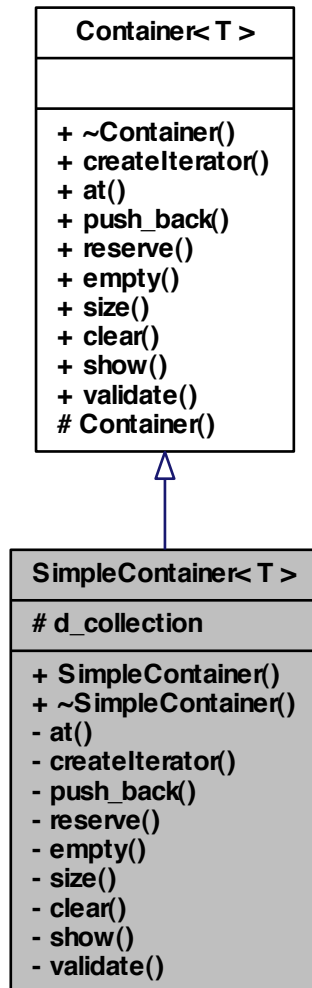
class [SimpleContainer](#) -

`#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](#) ()
- `void` [push_back](#) (`T` `const` &`const_reference`)
- `void` [reserve](#) (`int` `n`)
- `bool` [empty](#) ()
- `size_type` [size](#) ()
- `void` [clear](#) ()
- `void` [show](#) ()
- `bool` [validate](#) ()

Friends

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

5.39.1 Detailed Description

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

`class` [SimpleContainer](#) -

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

5.39.2 Constructor & Destructor Documentation

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

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

5.39.3 Member Function Documentation

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

Implements [Container](#)< `T` >.

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

Implements [Container](#)< `T` >.

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

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

Implements [Container< T >](#).

5.39.4 Friends And Related Function Documentation

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

Definition at line 12 of file SimpleContainer.h.

5.39.5 Member Data Documentation

5.39.5.1 `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:

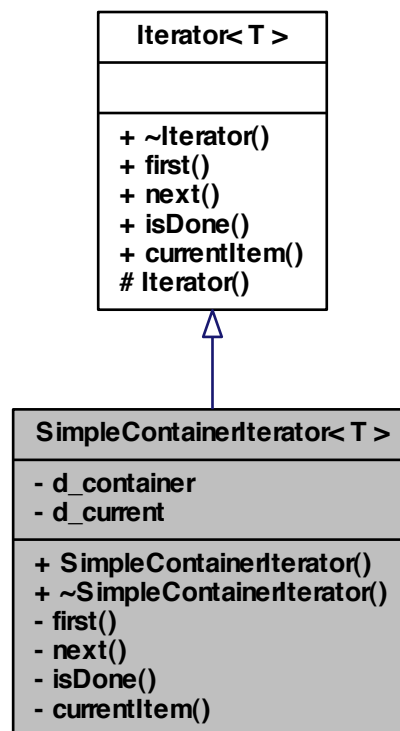
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

5.40 SimpleContainerIterator< T > Class Template Reference

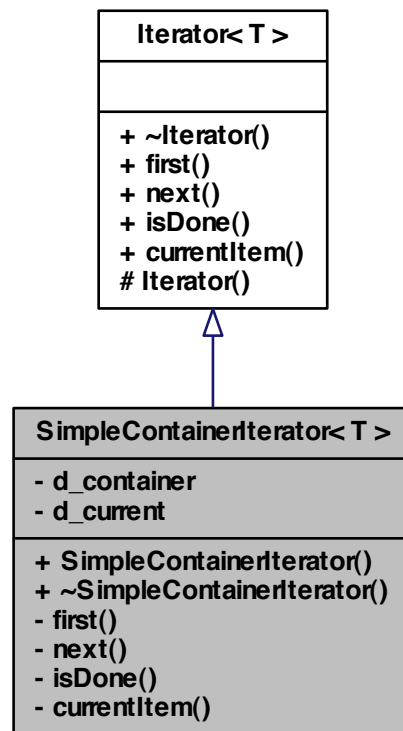
class [SimpleContainerIterator](#) -

```
#include <SimpleContainerIterator.h>
```

Inheritance diagram for SimpleContainerIterator< T >:



Collaboration diagram for SimpleContainerIterator< T >:



Public Member Functions

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

Private Member Functions

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

Private Attributes

- [Container< T > * d_container](#)
- size_type [d_current](#)

Friends

- class [SimpleContainer< T >](#)

5.40.1 Detailed Description

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

class [SimpleContainerIterator](#) -

Definition at line 6 of file SimpleContainerIterator.h.

5.40.2 Constructor & Destructor Documentation

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

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

5.40.3 Member Function Documentation

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

Implements [Iterator< T >](#).

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

Implements [Iterator< T >](#).

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

Implements [Iterator< T >](#).

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

Implements [Iterator< T >](#).

5.40.4 Friends And Related Function Documentation

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

Definition at line 13 of file SimpleContainerIterator.h.

5.40.5 Member Data Documentation

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

Definition at line 9 of file SimpleContainerIterator.h.

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

Definition at line 10 of file SimpleContainerIterator.h.

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

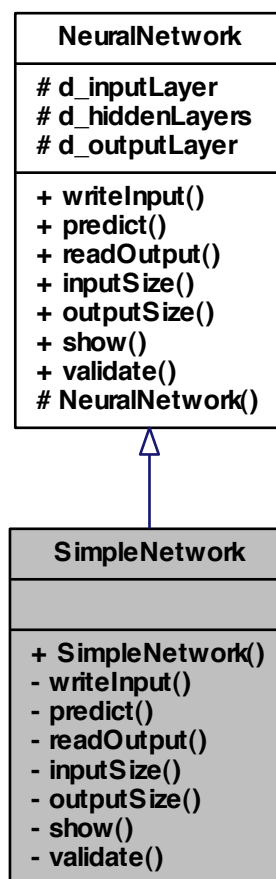
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[SimpleCo](#)

5.41 SimpleNetwork Class Reference

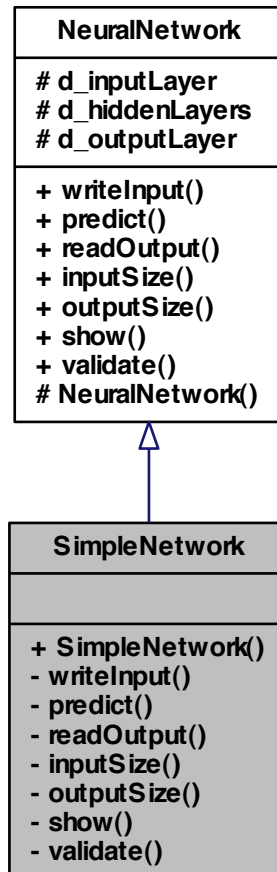
class [SimpleNetwork](#) -

#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 [predict](#) ()
- void [readOutput](#) (std::vector< double >::iterator &iterator)
- size_type [inputSize](#) ()

- size_type [outputSize](#) ()
- void [show](#) ()
- bool [validate](#) ()

5.41.1 Detailed Description

class [SimpleNetwork](#) -

Definition at line 5 of file SimpleNetwork.h.

5.41.2 Constructor & Destructor Documentation

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

Definition at line 16 of file SimpleNetwork.cpp.

```
SimpleNetwork (NeuralFactory &neuralFactory) :  
{  
    ...  
}
```

5.41.3 Member Function Documentation

5.41.3.1 size_type SimpleNetwork::inputSize () [private, virtual]

Implements [NeuralNetwork](#).

Definition at line 70 of file SimpleNetwork.cpp.

References [NeuralNetwork::d_inputLayer](#).

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

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

Here is the caller graph for this function:



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

Implements [NeuralNetwork](#).

Definition at line 76 of file SimpleNetwork.cpp.

References [NeuralNetwork::d_outputLayer](#).

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

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

Here is the caller graph for this function:

**5.41.3.3** `void SimpleNetwork::predict ()` [private, virtual]

Implements [NeuralNetwork](#).

Definition at line 23 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()->predict();
        }
    }

    // Output Layers
    boost::shared_ptr < Iterator<NeuronPtr> > neuronIterator(
        d_outputLayer->createIterator());
}
```

```

    for (neuronIterator->first(); !neuronIterator->isDone(); neuronIterator->next()
        )
    {
        neuronIterator->currentItem()->predict();
    }
}

```

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

Implements [NeuralNetwork](#).

Definition at line 60 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:



5.41.3.5 void SimpleNetwork::show () [private, virtual]

Implements [NeuralNetwork](#).

Definition at line 82 of file SimpleNetwork.cpp.

References [NeuralNetwork::d_hiddenLayers](#), [NeuralNetwork::d_inputLayer](#), and [NeuralNetwork::d_outputLayer](#).

```

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

```

```

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

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

```

5.41.3.6 bool SimpleNetwork::validate () [private, virtual]

Implements [NeuralNetwork](#).

Definition at line 103 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;
}

```

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

Implements [NeuralNetwork](#).

Definition at line 50 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:

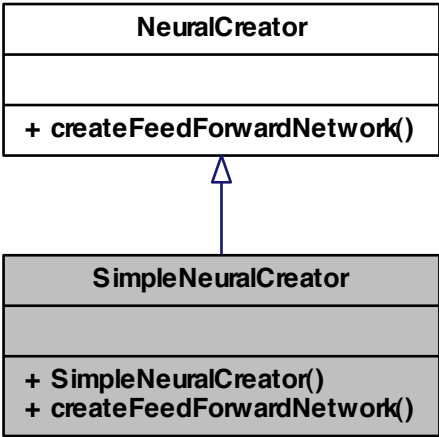


The documentation for this class was generated from the following files:

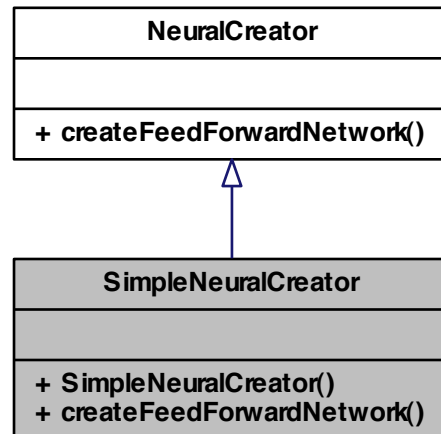
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[SimpleNetw](#)

5.42 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)

5.42.1 Detailed Description

class [SimpleNeuralCreator](#) -

Definition at line 5 of file SimpleNeuralCreator.h.

5.42.2 Constructor & Destructor Documentation

5.42.2.1 SimpleNeuralCreator::SimpleNeuralCreator ()

Definition at line 19 of file SimpleNeuralCreator.cpp.

```
{
}
```

5.42.3 Member Function Documentation

5.42.3.1 `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(outputLa
        yerFactory));
    NeuronPtr neuronPtr;

    if (numberOfNeurons.size() <= 2)
    {
        throw std::range_error(
            "[C++ CreateFeedForwardNetwork::validate]: Error, number of layers lowe
                r 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 w
            hether to use outputLayerFactory o 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);

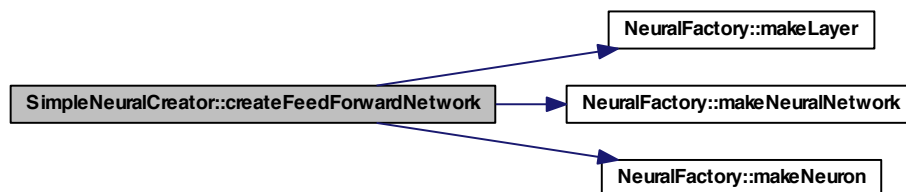
        }
    }

    //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:

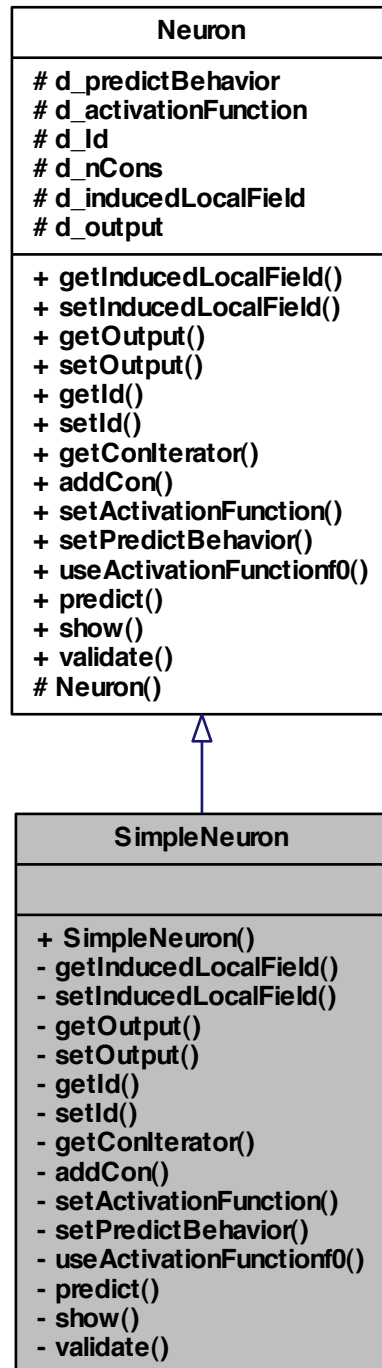
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/[SimpleNe](#)
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[SimpleNeuralCreator.cp](#)

5.43 SimpleNeuron Class Reference

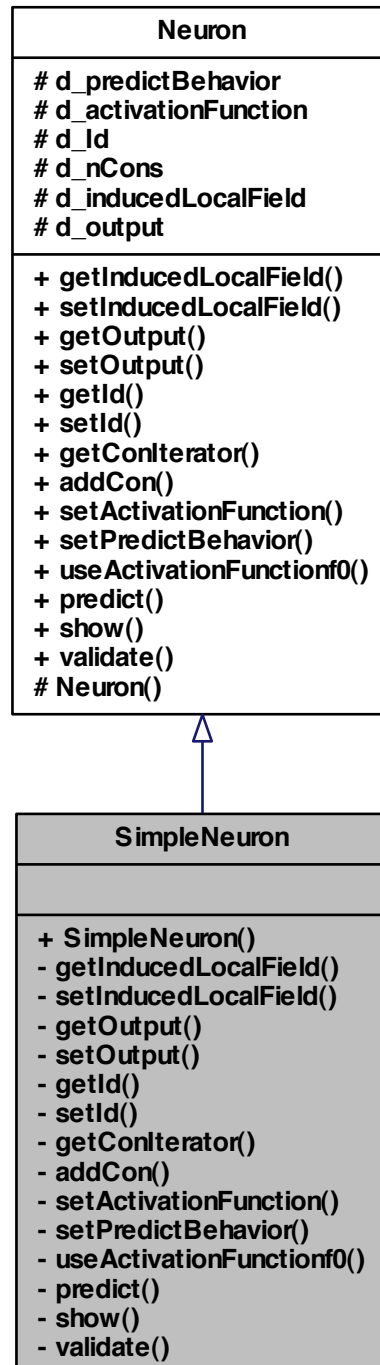
class [SimpleNeuron](#) -

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


Inheritance diagram for SimpleNeuron:



Collaboration diagram for SimpleNeuron:



Public Member Functions

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

Private Member Functions

- double [getInducedLocalField](#) ()
- void [setInducedLocalField](#) (double inducedLocalField)
- double [getOutput](#) ()
- void [setOutput](#) (double output)
- [Handler](#) [getId](#) ()
- void [setId](#) ([Handler](#) Id)
- [ConIteratorPtr](#) [getConIterator](#) ()
- void [addCon](#) ([ConPtr](#) conPtr)
- void [setActivationFunction](#) ([ActivationFunctionPtr](#) activationFunctionPtr)
- void [setPredictBehavior](#) ([PredictBehaviorPtr](#) predictBehaviorPtr)
- double [useActivationFunction0](#) ()
- void [predict](#) ()
- void [show](#) ()
- bool [validate](#) ()

5.43.1 Detailed Description

class [SimpleNeuron](#) -

Definition at line 5 of file SimpleNeuron.h.

5.43.2 Constructor & Destructor Documentation

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

Definition at line 18 of file SimpleNeuron.cpp.

```

SimpleNeuron (neuralFactory)
{
}

```

5.43.3 Member Function Documentation

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

Implements [Neuron](#).

Definition at line 66 of file SimpleNeuron.cpp.

References [Neuron::d_nCons](#).

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

5.43.3.2 ConlteratorPtr SimpleNeuron::getConlterator () [private, virtual]

Implements [Neuron](#).

Definition at line 60 of file SimpleNeuron.cpp.

References [Neuron::d_nCons](#).

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

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

Implements [Neuron](#).

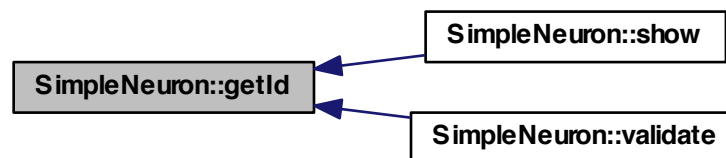
Definition at line 48 of file SimpleNeuron.cpp.

References [Neuron::d_Id](#).

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

```
{  
    return d_Id;  
}
```

Here is the caller graph for this function:



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

Implements [Neuron](#).

Definition at line 24 of file SimpleNeuron.cpp.

References `Neuron::d_inducedLocalField`.

```
{  
    return d_inducedLocalField;  
}
```

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

Implements [Neuron](#).

Definition at line 36 of file SimpleNeuron.cpp.

References `Neuron::d_output`.

```
{  
    return d_output;  
}
```

5.43.3.6 `void SimpleNeuron::predict ()` [private, virtual]

Implements [Neuron](#).

Definition at line 90 of file SimpleNeuron.cpp.

References `Neuron::d_predictBehavior`.

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

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

Implements [Neuron](#).

Definition at line 72 of file SimpleNeuron.cpp.

References `Neuron::d_activationFunction`.

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

5.43.3.8 `void SimpleNeuron::setId (Handler Id)` [private, virtual]

Implements [Neuron](#).

Definition at line 54 of file SimpleNeuron.cpp.

References `Neuron::d_Id`.

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

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

Implements [Neuron](#).

Definition at line 30 of file SimpleNeuron.cpp.

References [Neuron::d_inducedLocalField](#).

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

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

Implements [Neuron](#).

Definition at line 42 of file SimpleNeuron.cpp.

References [Neuron::d_output](#).

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

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

Implements [Neuron](#).

Definition at line 78 of file SimpleNeuron.cpp.

References [Neuron::d_predictBehavior](#).

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

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

Implements [Neuron](#).

Definition at line 96 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\n-----");
        Rprintf("\n output: %lf", d_output);
        Rprintf("\n\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\n-----");
        d_predictBehavior->show();

        Rprintf("\n output: %lf", d_output);
        Rprintf("\n\n-----");
        d_nCons->show();
        Rprintf("\n\n-----");
    }
}

```

Here is the call graph for this function:



5.43.3.13 `double SimpleNeuron::useActivationFunction0 ()` [private, virtual]

Implements [Neuron](#).

Definition at line 84 of file SimpleNeuron.cpp.

References [Neuron::d_activationFunction](#).

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

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

Implements [Neuron](#).

Definition at line 138 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:

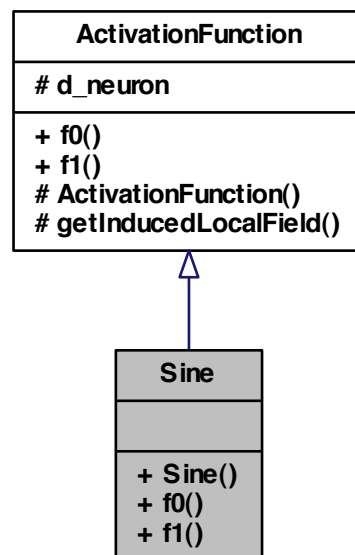
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/[SimpleNeu](#)

5.44 Sine Class Reference

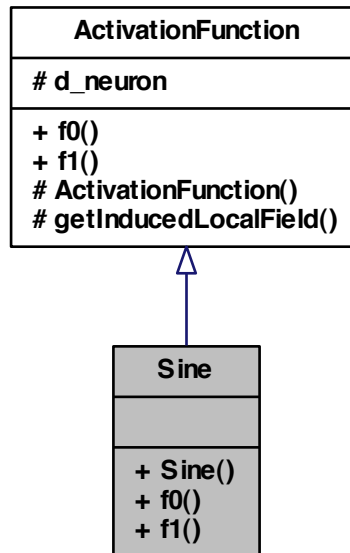
class [Sine](#) -

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


Inheritance diagram for Sine:



Collaboration diagram for Sine:



Public Member Functions

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

5.44.1 Detailed Description

class [Sine](#) -

Definition at line 5 of file Sine.h.

5.44.2 Constructor & Destructor Documentation

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

5.44.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

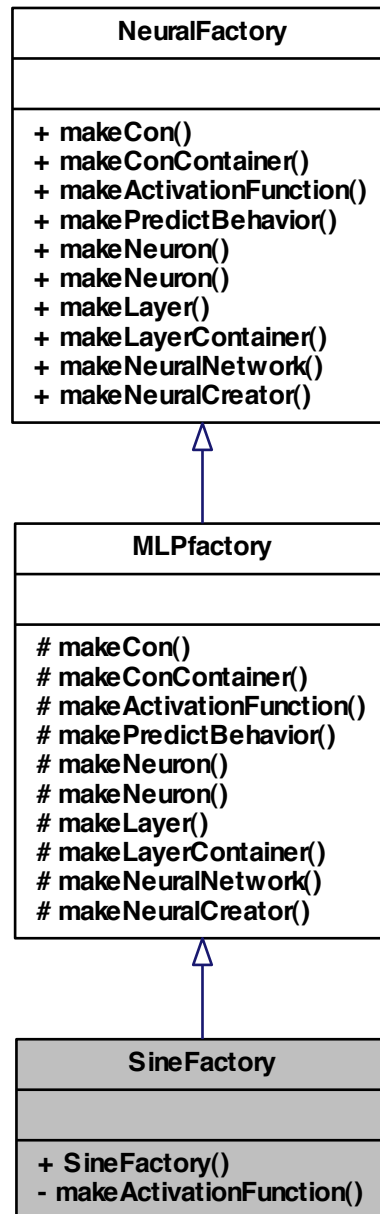
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Sine.h](#)

5.45 SineFactory Class Reference

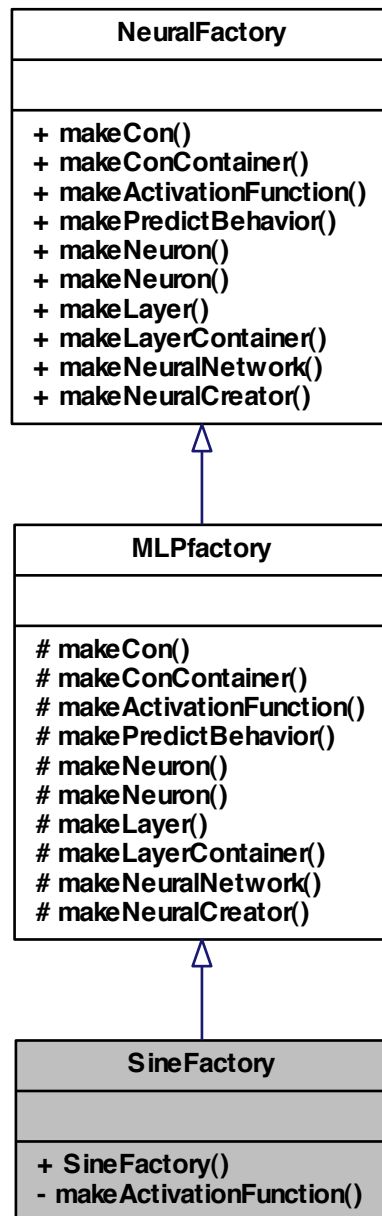
class [SineFactory](#) -

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

Inheritance diagram for SineFactory:



Collaboration diagram for SineFactory:



Public Member Functions

- [SineFactory](#) ()

Private Member Functions

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

5.45.1 Detailed Description

class [SineFactory](#) -

Definition at line 5 of file SineFactory.h.

5.45.2 Constructor & Destructor Documentation

5.45.2.1 [SineFactory::SineFactory](#) ()

5.45.3 Member Function Documentation

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

Implements [MLPfactory](#).

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

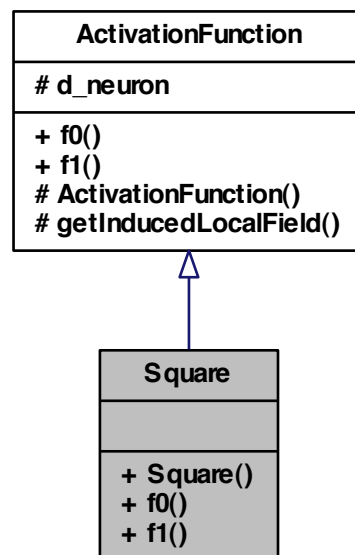
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

5.46 Square Class Reference

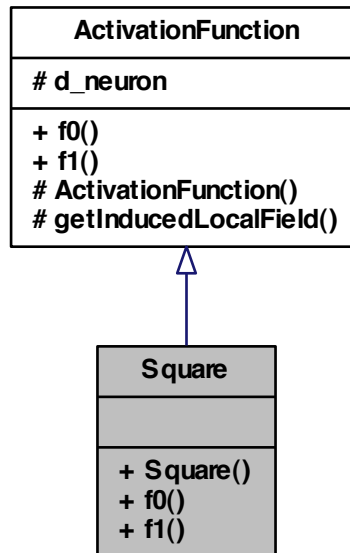
class [Square](#) -

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

Inheritance diagram for Square:



Collaboration diagram for Square:



Public Member Functions

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

5.46.1 Detailed Description

class [Square](#) -

Definition at line 5 of file Square.h.

5.46.2 Constructor & Destructor Documentation

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

5.46.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

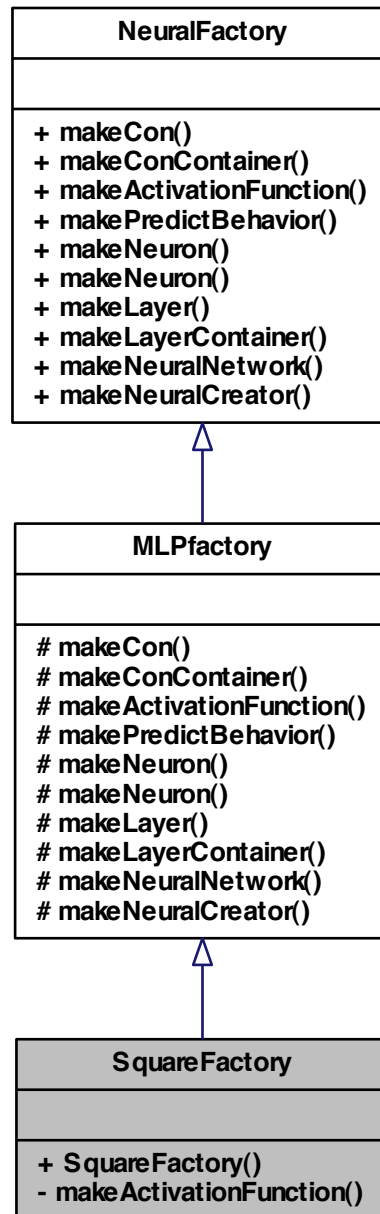
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Square.h`

5.47 SquareFactory Class Reference

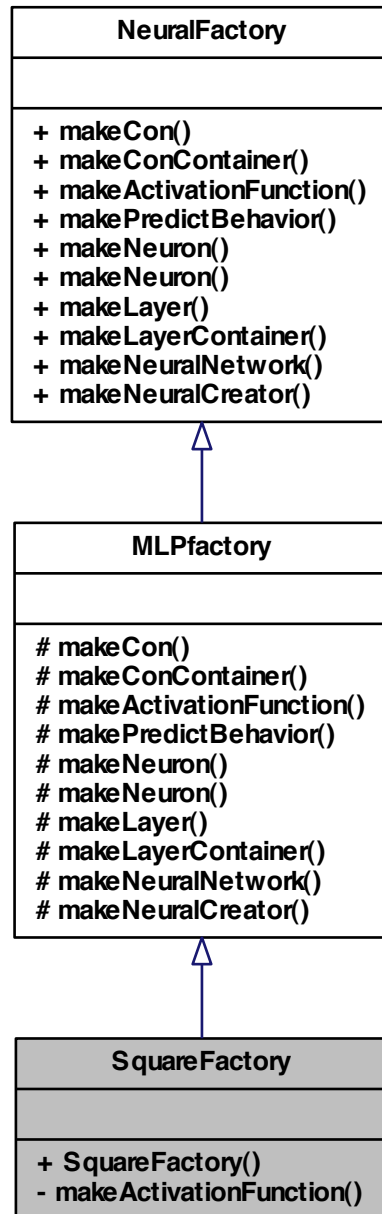
class [SquareFactory](#) -

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

Inheritance diagram for SquareFactory:



Collaboration diagram for SquareFactory:



Public Member Functions

- [SquareFactory](#) ()

Private Member Functions

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

5.47.1 Detailed Description

class [SquareFactory](#) -

Definition at line 5 of file SquareFactory.h.

5.47.2 Constructor & Destructor Documentation

5.47.2.1 [SquareFactory::SquareFactory](#) ()

5.47.3 Member Function Documentation

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

Implements [MLPfactory](#).

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

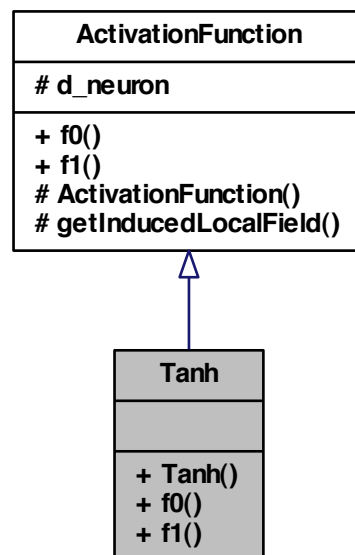
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

5.48 Tanh Class Reference

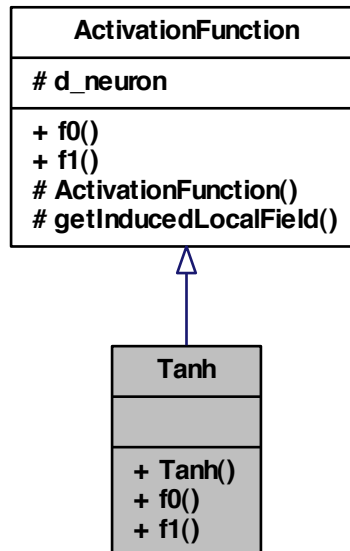
class [Tanh](#) -

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

Inheritance diagram for Tanh:



Collaboration diagram for Tanh:



Public Member Functions

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

5.48.1 Detailed Description

class [Tanh](#) -

Definition at line 5 of file Tanh.h.

5.48.2 Constructor & Destructor Documentation

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

Definition at line 15 of file Tanh.cpp.

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

```
}
```

5.48.3 Member Function Documentation

5.48.3.1 `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:



5.48.3.2 `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:

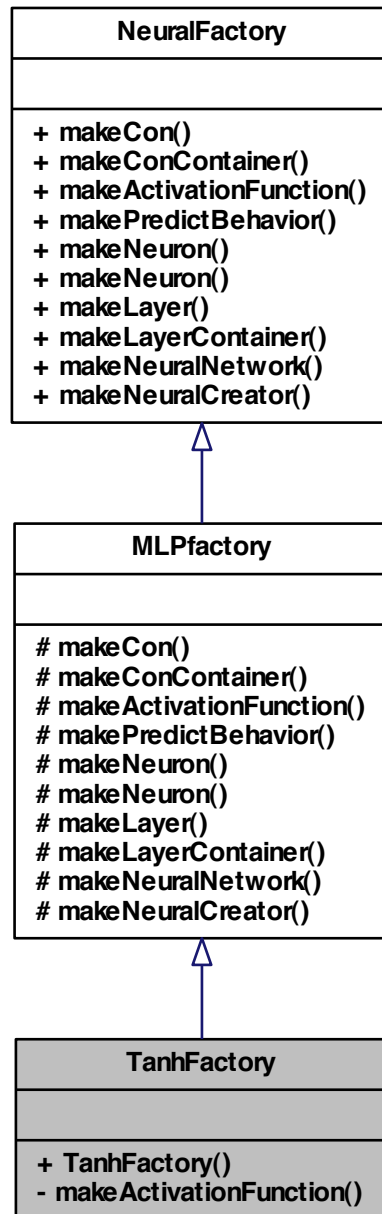
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead`
- `/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Tanh.cpp`

5.49 TanhFactory Class Reference

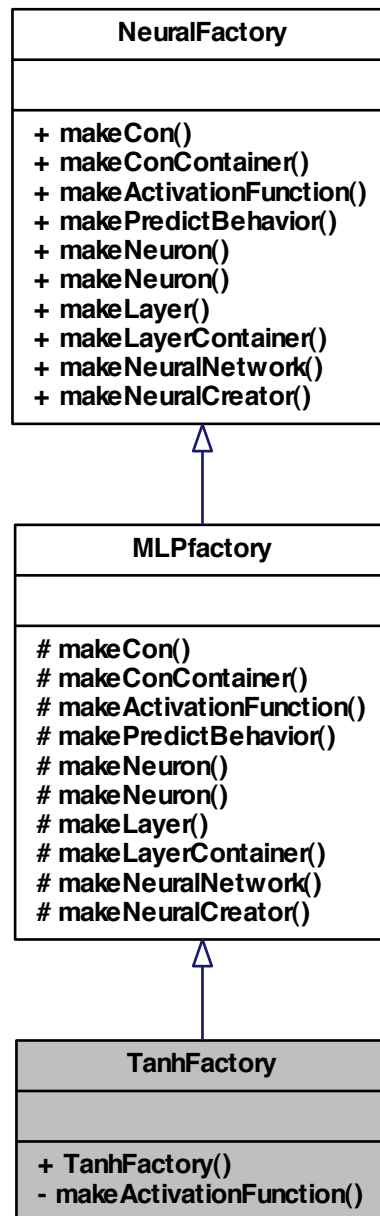
class [TanhFactory](#) -

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


Inheritance diagram for TanhFactory:



Collaboration diagram for TanhFactory:



Public Member Functions

- [TanhFactory](#) ()

Private Member Functions

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

5.49.1 Detailed Description

class [TanhFactory](#) -

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

5.49.2 Constructor & Destructor Documentation

5.49.2.1 [TanhFactory::TanhFactory](#) ()

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

```
{  
}
```

5.49.3 Member Function Documentation

5.49.3.1 [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:

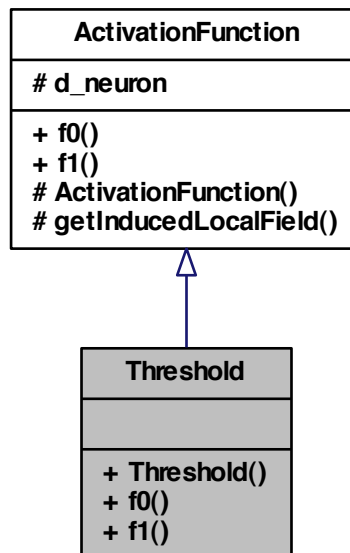
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/\[TanhFactory.h\]\(#\)](#)
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/\[TanhFactory.cpp\]\(#\)](#)

5.50 Threshold Class Reference

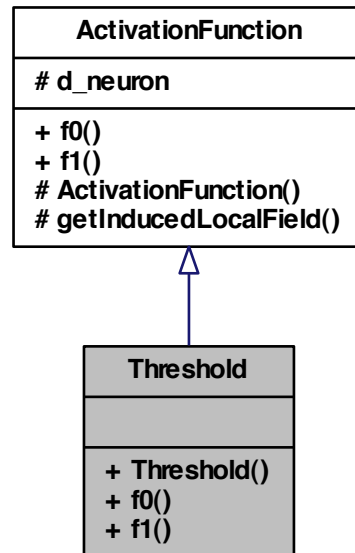
class [Threshold](#) -

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

Inheritance diagram for Threshold:



Collaboration diagram for Threshold:



Public Member Functions

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

5.50.1 Detailed Description

class [Threshold](#) -

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

5.50.2 Constructor & Destructor Documentation

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

5.50.3 Member Function Documentation

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

Implements [ActivationFunction](#).

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

Implements [ActivationFunction](#).

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

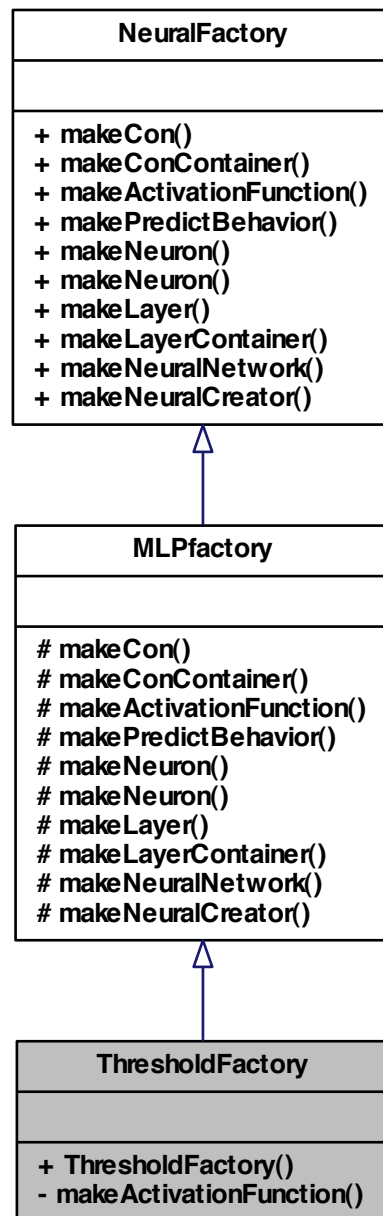
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHead

5.51 ThresholdFactory Class Reference

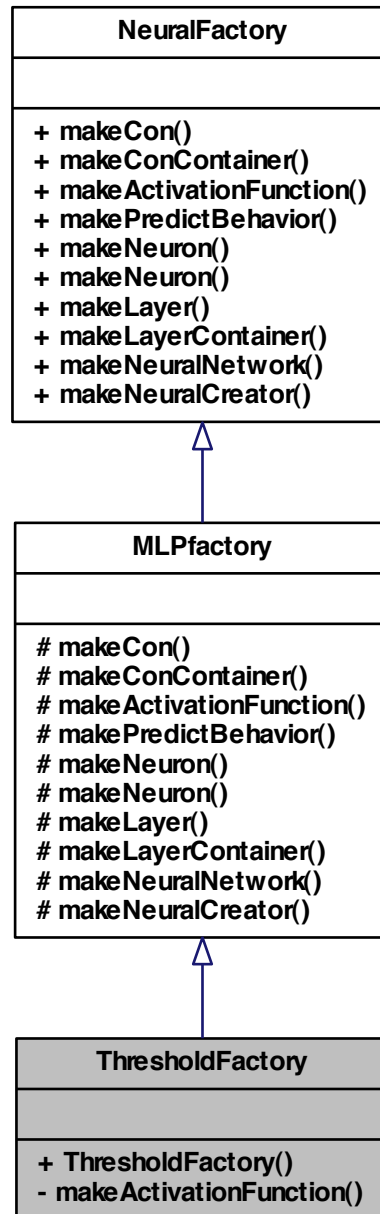
class [ThresholdFactory](#) -

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

Inheritance diagram for ThresholdFactory:



Collaboration diagram for ThresholdFactory:



Public Member Functions

- [ThresholdFactory](#) ()

Private Member Functions

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

5.51.1 Detailed Description

class [ThresholdFactory](#) -

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

5.51.2 Constructor & Destructor Documentation

5.51.2.1 [ThresholdFactory::ThresholdFactory](#) ()

5.51.3 Member Function Documentation

5.51.3.1 [ActivationFunctionPtr](#) [ThresholdFactory::makeActivationFunction](#) ([NeuronPtr](#) *neuronPtr*) [[private](#), [virtual](#)]

Implements [MLPfactory](#).

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

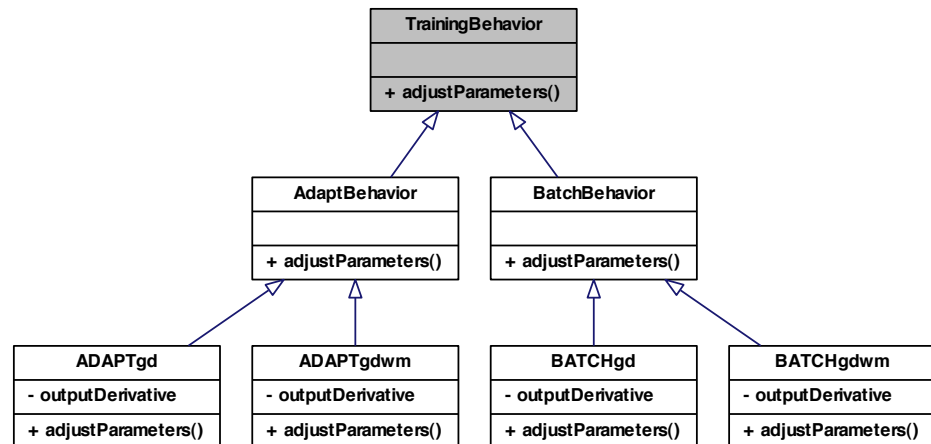
- [/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ThresholdFactory.h](#)

5.52 TrainingBehavior Class Reference

class [TrainingBehavior](#) -

```
#include <TrainingBehavior.h>
```

Inheritance diagram for TrainingBehavior:



Public Member Functions

- void [adjustParameters](#) ()

5.52.1 Detailed Description

class [TrainingBehavior](#) -

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

5.52.2 Member Function Documentation

5.52.2.1 void TrainingBehavior::adjustParameters ()

Reimplemented in [AdaptBehavior](#), [ADAPTgd](#), [ADAPTgdwm](#), [BatchBehavior](#), [BATCHgd](#), and [BATCHgdwm](#).

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

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

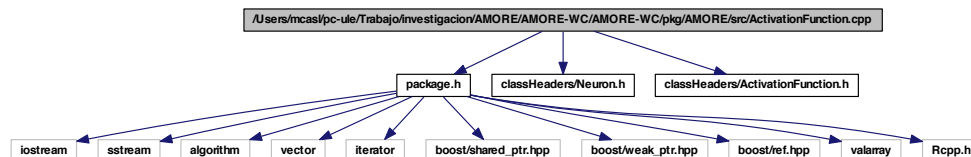
Chapter 6

File Documentation

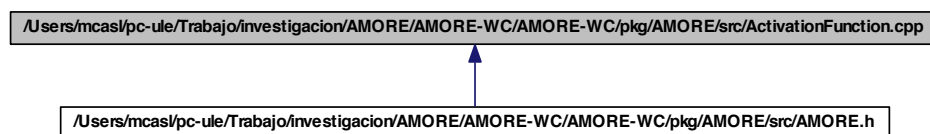
6.1 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/ActivationFunction.cpp File Reference

```
#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/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/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 "Con.cpp"

#include "ActivationFunction.cpp"
#include "Tanh.cpp"
#include "Identity.cpp"
#include "PredictBehavior.cpp"
#include "MLPbehavior.cpp"
#include "Neuron.cpp"
#include "SimpleNeuron.cpp"
#include "MLPfactory.cpp"
#include "TanhFactory.cpp"
#include "IdentityFactory.cpp"
#include "NeuralNetwork.cpp"
#include "SimpleNetwork.cpp"
#include "SimpleNeuralCreator.cpp"
#include "NetworkRinterface.cpp"
#include "Container.cpp"
#include "Iterator.cpp"
#include "SimpleContainer.cpp"
#include "SimpleContainerIterator.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< [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< Neuron > NeuronWeakPtr`

6.2.1 Define Documentation

6.2.1.1 `#define size_type unsigned int`

Definition at line 75 of file AMORE.h.

Referenced by `SimpleNetwork::readOutput()`, and `SimpleNetwork::writeInput()`.

6.2.2 Typedef Documentation

6.2.2.1 `typedef boost::shared_ptr<ActivationFunction> ActivationFunctionPtr`

Definition at line 87 of file AMORE.h.

6.2.2.2 `typedef boost::reference_wrapper<PredictBehavior> ActivationFunctionRef`

Definition at line 81 of file AMORE.h.

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

Definition at line 99 of file AMORE.h.

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

Definition at line 95 of file AMORE.h.

6.2.2.5 `typedef boost::shared_ptr<Con> ConPtr`

Definition at line 90 of file AMORE.h.

6.2.2.6 `typedef int Handler`

Definition at line 78 of file AMORE.h.

6.2.2.7 `typedef boost::shared_ptr< Container< LayerPtr > > LayerContainerPtr`

Definition at line 98 of file AMORE.h.

6.2.2.8 `typedef boost::shared_ptr< Container< NeuronPtr > > LayerPtr`

Definition at line 97 of file AMORE.h.

6.2.2.9 `typedef boost::shared_ptr< NeuralCreator > NeuralCreatorPtr`

Definition at line 102 of file AMORE.h.

6.2.2.10 `typedef boost::shared_ptr< NeuralFactory > NeuralFactoryPtr`

Definition at line 101 of file AMORE.h.

6.2.2.11 `typedef boost::shared_ptr< NeuralNetwork > NeuralNetworkPtr`

Definition at line 91 of file AMORE.h.

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

Definition at line 94 of file AMORE.h.

6.2.2.13 `typedef boost::shared_ptr< Neuron > NeuronPtr`

Definition at line 89 of file AMORE.h.

6.2.2.14 `typedef boost::reference_wrapper< Neuron > NeuronRef`

Definition at line 84 of file AMORE.h.

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

Definition at line 104 of file AMORE.h.

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

Definition at line 88 of file AMORE.h.

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

Definition at line 82 of file AMORE.h.

6.2.2.18 `typedef boost::reference_wrapper<TrainingBehavior> TrainingBehaviorRef`

Definition at line 83 of file AMORE.h.

6.3 /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:



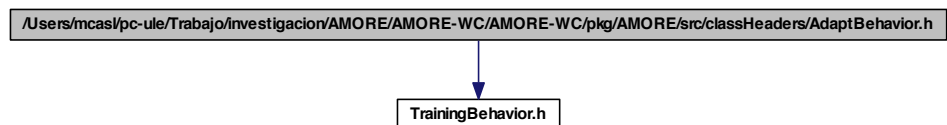
Classes

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

6.4 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptBehavior.h File Reference

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

Include dependency graph for AdaptBehavior.h:

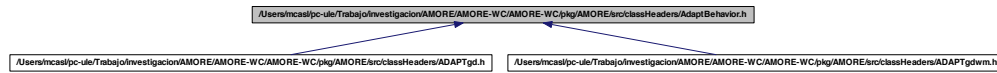


6.5 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgd.h File Reference

Reference

195

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



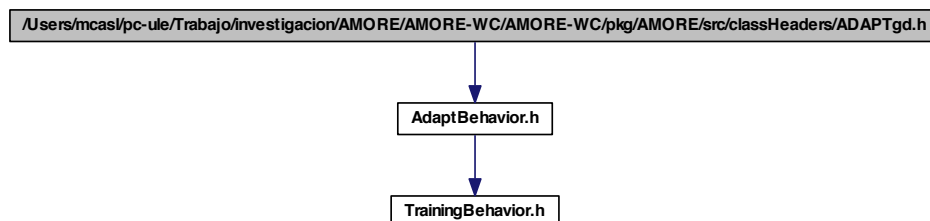
Classes

- class [AdaptBehavior](#)
class [AdaptBehavior](#) -

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

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

Include dependency graph for ADAPTgd.h:



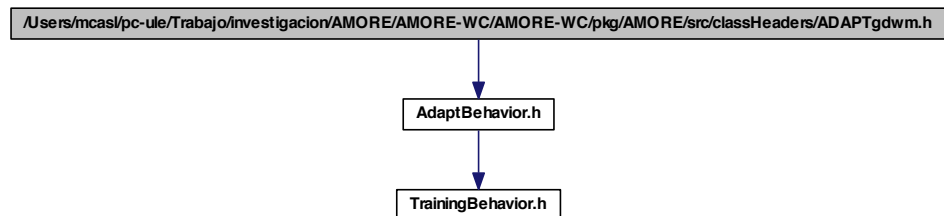
Classes

- class [ADAPTgd](#)
class [ADAPTgd](#) -

6.6 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwm.h File Reference

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

Include dependency graph for ADAPTgdm.h:



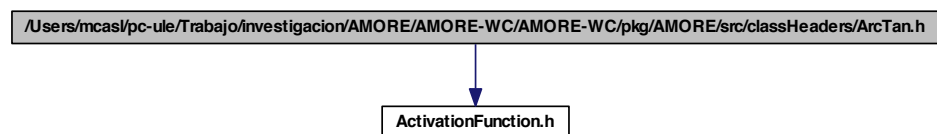
Classes

- class [ADAPTgdm](#)
class [ADAPTgdm](#) -

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

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

Include dependency graph for ArcTan.h:



Classes

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

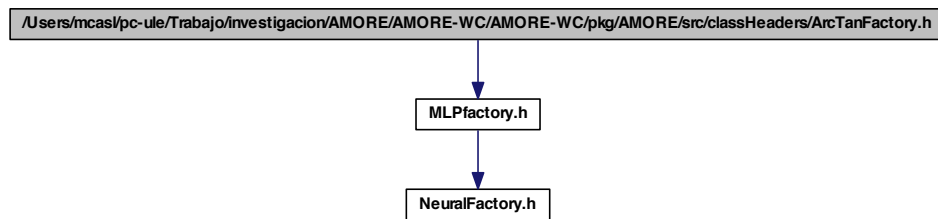
6.8 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ArcTanFactory.h File Reference

Reference

6.8 /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:



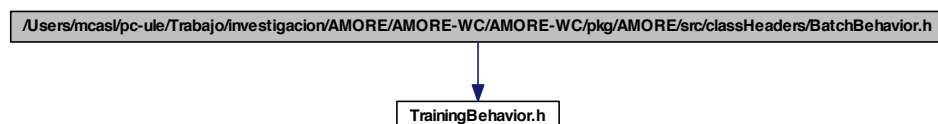
Classes

- class [ArcTanFactory](#)
class ArcTanFactory -

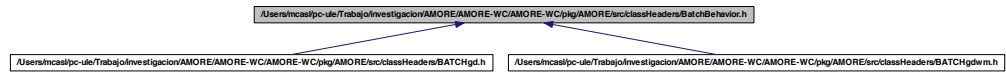
6.9 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BatchBehavior.h File Reference

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

Include dependency graph for BatchBehavior.h:



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



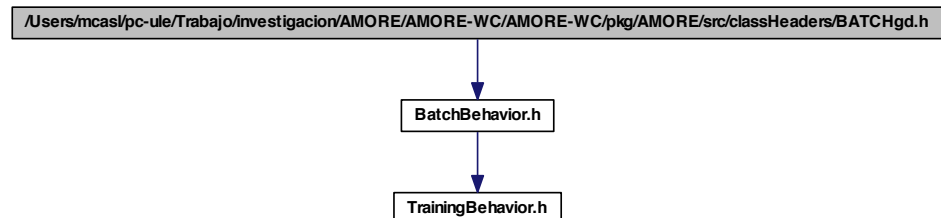
Classes

- class [BatchBehavior](#)
class [BatchBehavior](#) -

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

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

Include dependency graph for BATCHgd.h:



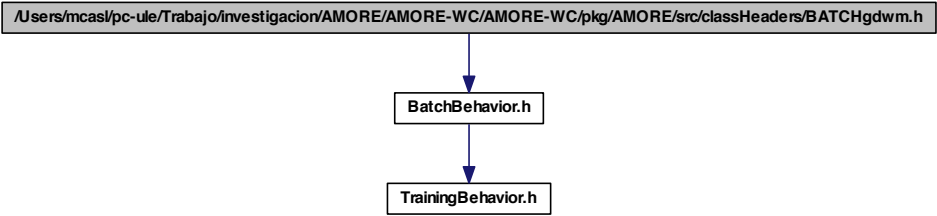
Classes

- class [BATCHgd](#)
class [BATCHgd](#) -

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

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

Include dependency graph for BATCHgdwm.h:



Classes

- class BATCHgdwm
class BATCHgdwm -

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

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



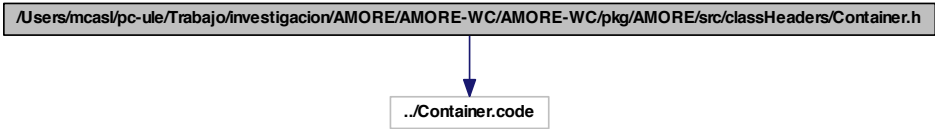
Classes

- class Con
class Con -

6.13 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Container.h File Reference

```
#include "../Container.code"
```

Include dependency graph for Container.h:



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



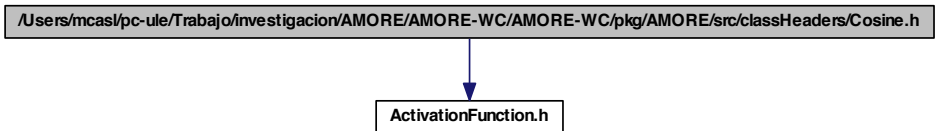
Classes

- class `Container< T >`
 class `Container` -

6.14 /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

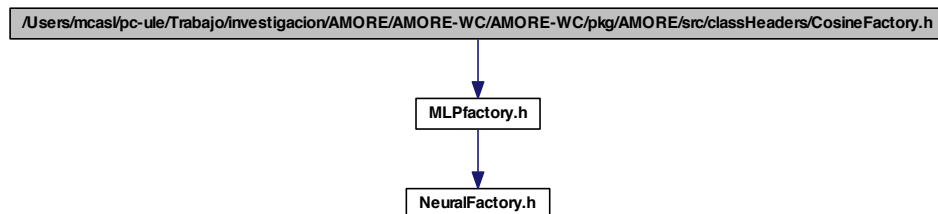
- class `Cosine`

class [Cosine](#) -

6.15 /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:



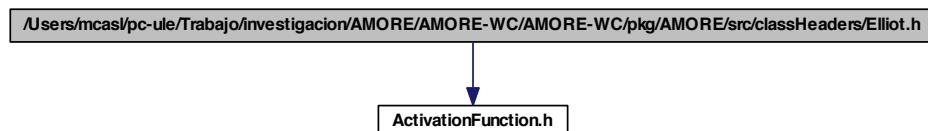
Classes

- class [CosineFactory](#)
class [CosineFactory](#) -

6.16 /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:



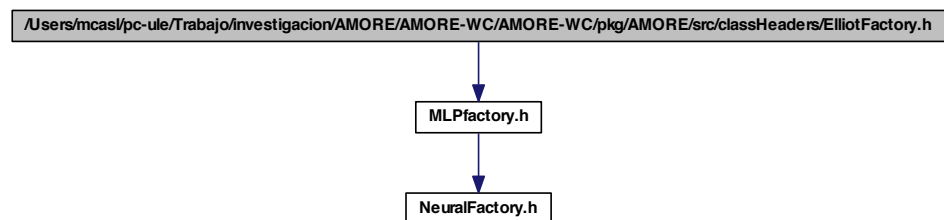
Classes

- class [Elliot](#)
class [Elliot](#) -

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

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

Include dependency graph for ElliotFactory.h:



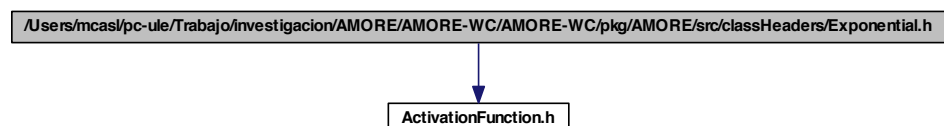
Classes

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

6.18 /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

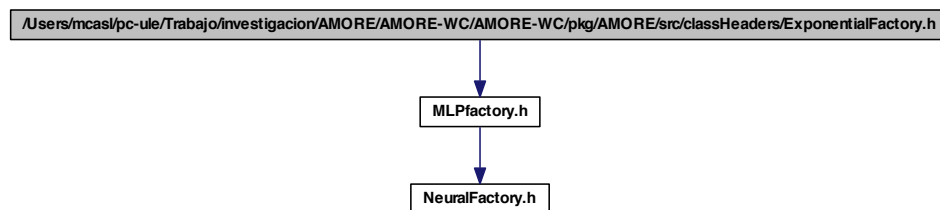
- class [Exponential](#)

class [Exponential](#) -

6.19 /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

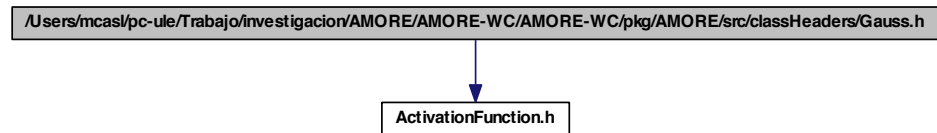
- class [ExponentialFactory](#)

class [ExponentialFactory](#) -

6.20 /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:



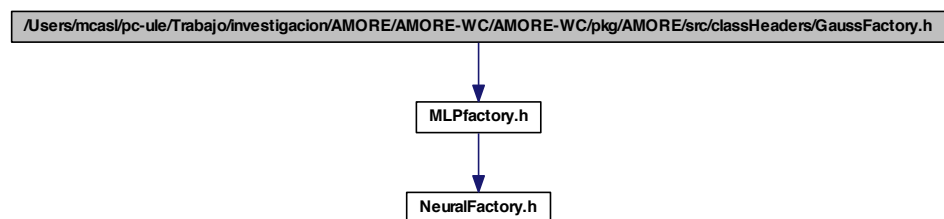
Classes

- class [Gauss](#)
class Gauss -

6.21 /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 -

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

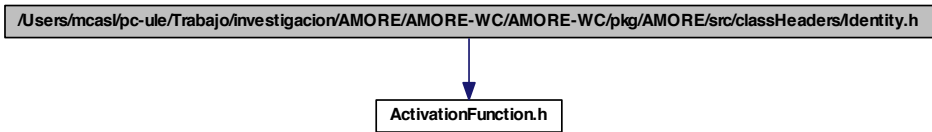
Reference

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

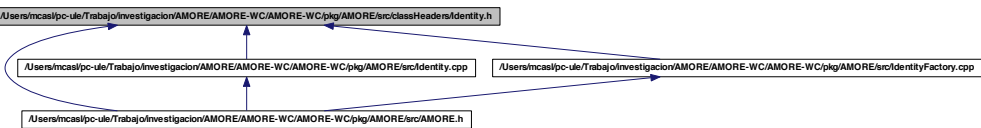
205

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

Include dependency graph for Identity.h:



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



Classes

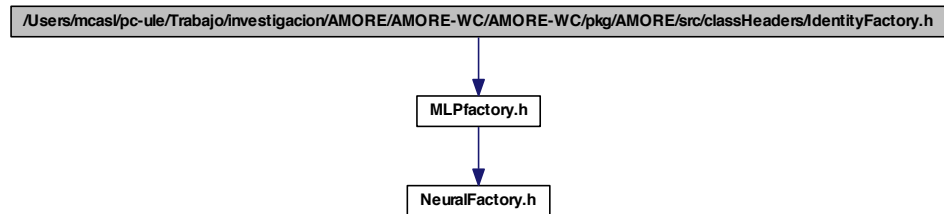
- class [Identity](#)
class Identity -

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

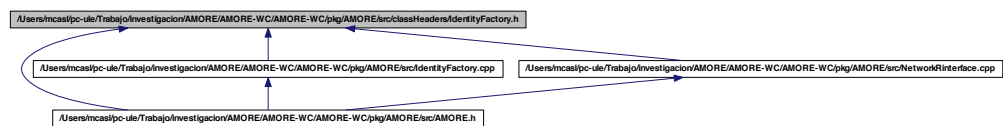
ence

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

Include dependency graph for IdentityFactory.h:



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



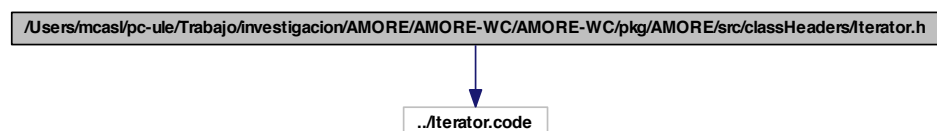
Classes

- class [IdentityFactory](#)
class IdentityFactory -

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

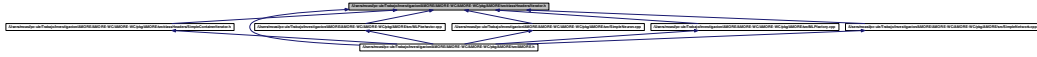
```
#include "../Iterator.code"
```

Include dependency graph for Iterator.h:



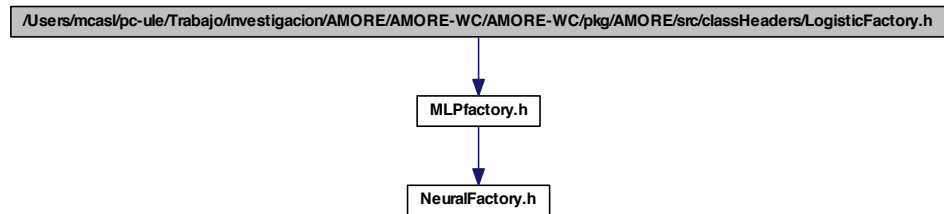
Reference

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



Generated on Thu Jul 28 2011 01:19:52 for AMORE++ by Doxygen

Include dependency graph for LogisticFactory.h:



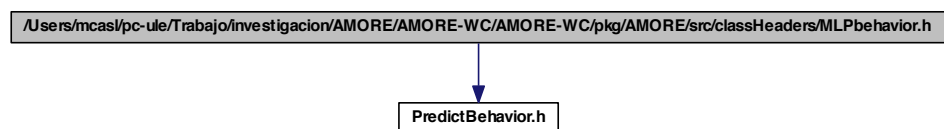
Classes

- class [LogisticFactory](#)
class [LogisticFactory](#) -

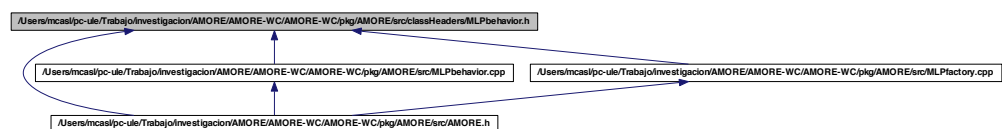
6.27 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPbehavior.h File Reference

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

Include dependency graph for MLPbehavior.h:



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



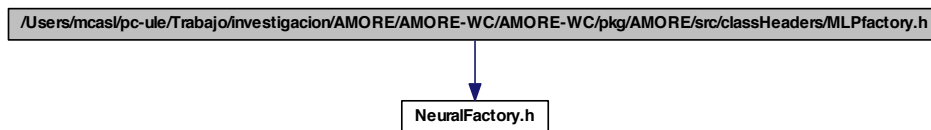
- class [MLPbehavior](#)

class [MLPbehavior](#) -

6.28 /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:



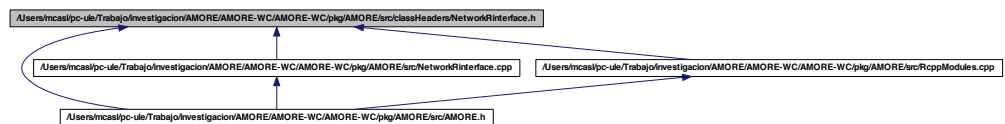
Classes

- class [MLPfactory](#)

class [MLPfactory](#) -

6.29 /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:

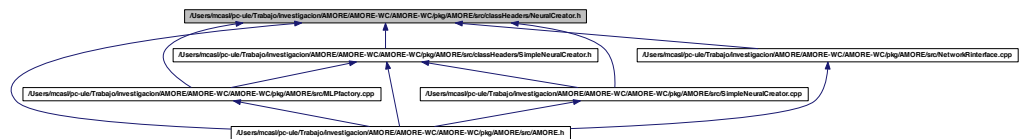


Classes

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

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

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



Classes

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

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

Reference

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

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

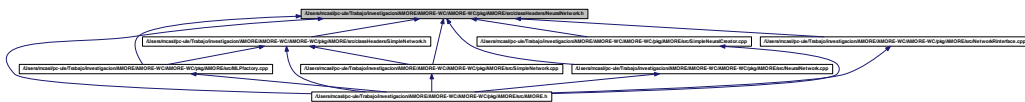


Classes

- class [NeuralFactory](#)
class [NeuralFactory](#) -

6.32 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralNetwork.h File Reference

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



Classes

- class [NeuralNetwork](#)
class [NeuralNetwork](#) -

6.33 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Neuron.h File Reference

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

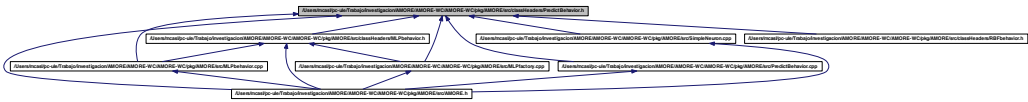


Classes

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

6.34 /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:



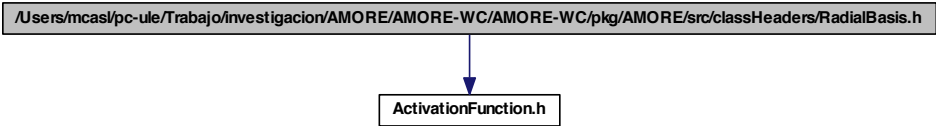
Classes

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

6.35 /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* -

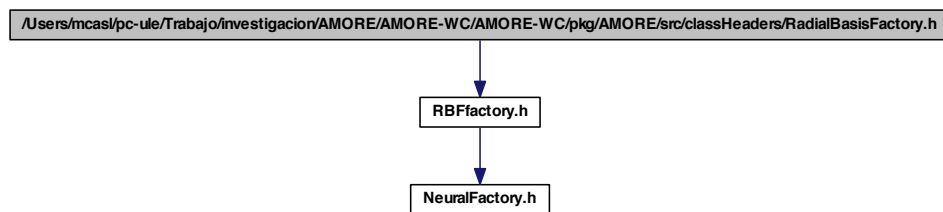
6.36 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasisFactory.h File

Reference

6.36 ²¹³ /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBasisFactory.h File Reference

```
#include "RBFactory.h"
```

Include dependency graph for RadialBasisFactory.h:



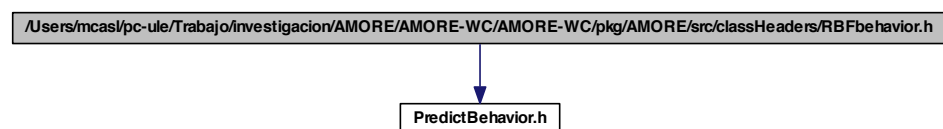
Classes

- class [RadialBasisFactory](#)
class [RadialBasisFactory](#) -

6.37 /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

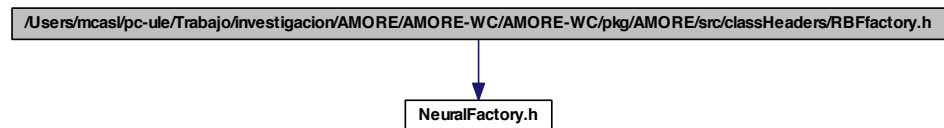
- class [RBFbehavior](#)

class [RBFbehavior](#) -

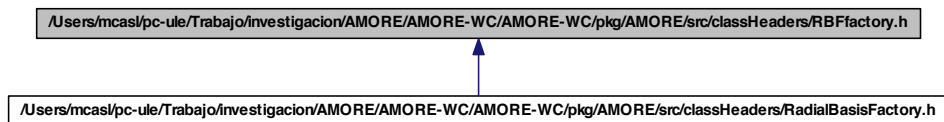
6.38 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFfactory.h File Reference

```
#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](#) -

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

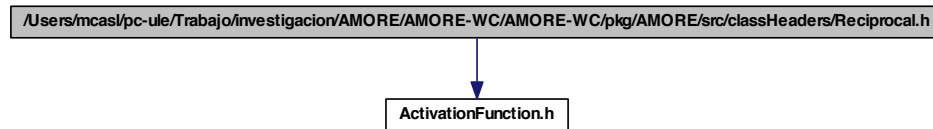
```
#include "ActivationFunction.h"
```

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

Reference

215

Include dependency graph for Reciprocal.h:



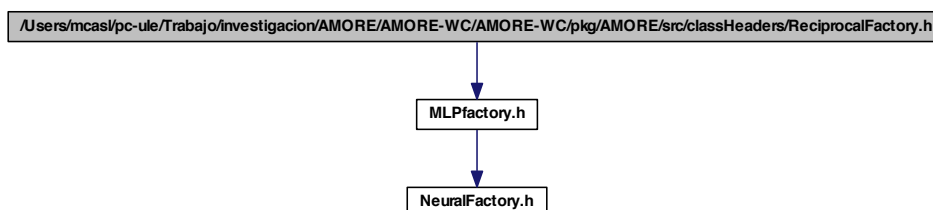
Classes

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

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

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

Include dependency graph for ReciprocalFactory.h:



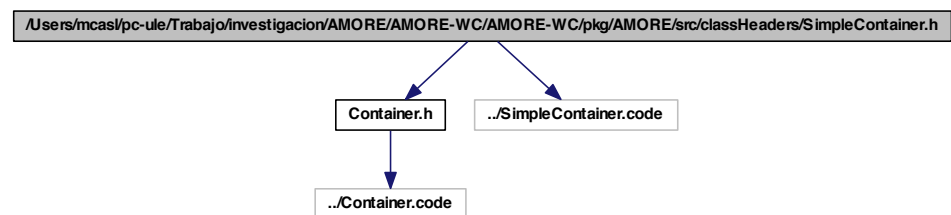
Classes

- class [ReciprocalFactory](#)
class [ReciprocalFactory](#) -

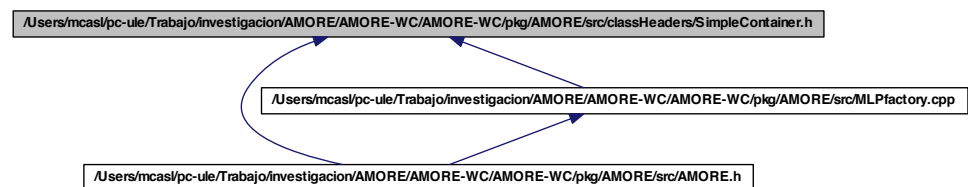
6.41 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainer.h File Reference

```
#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](#) -

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

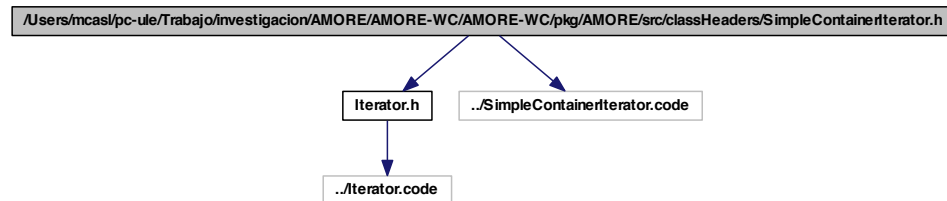
```
#include "Iterator.h"
```

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

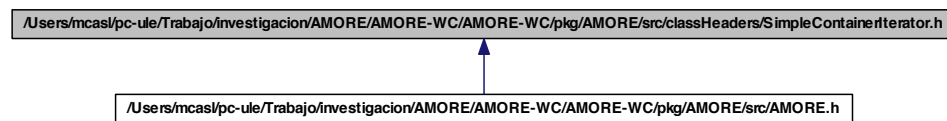
217

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

Include dependency graph for SimpleContainerIterator.h:



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



Classes

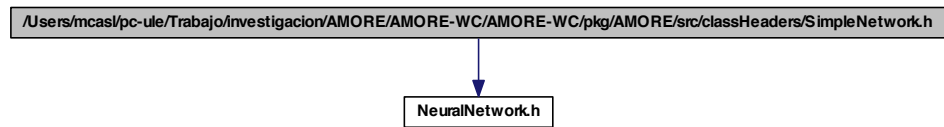
- class [SimpleContainerIterator< T >](#)

class [SimpleContainerIterator](#) -

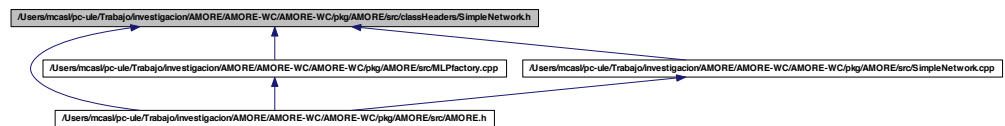
6.43 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNetwork.h File Reference

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

Include dependency graph for SimpleNetwork.h:



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



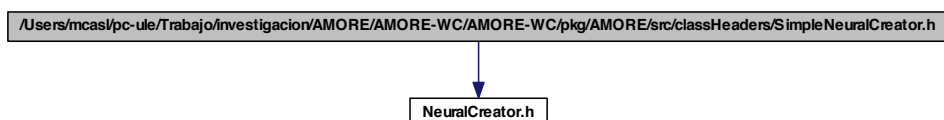
Classes

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

6.44 /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:

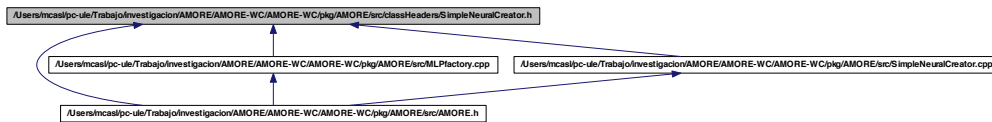


6.45 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuron.h File

Reference

219

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



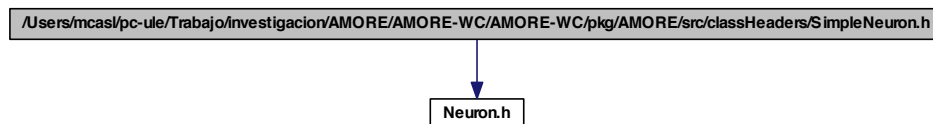
Classes

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

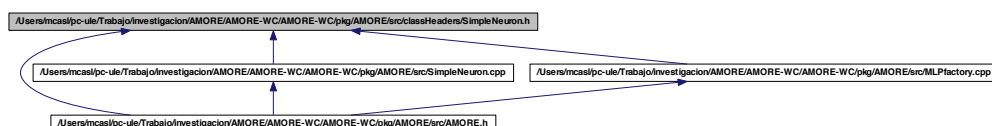
6.45 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuron.h File Reference

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

Include dependency graph for SimpleNeuron.h:



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



Classes

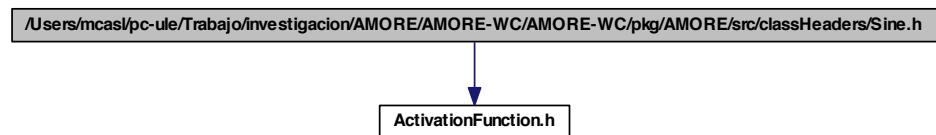
- class [SimpleNeuron](#)

class [SimpleNeuron](#) -

6.46 /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:



Classes

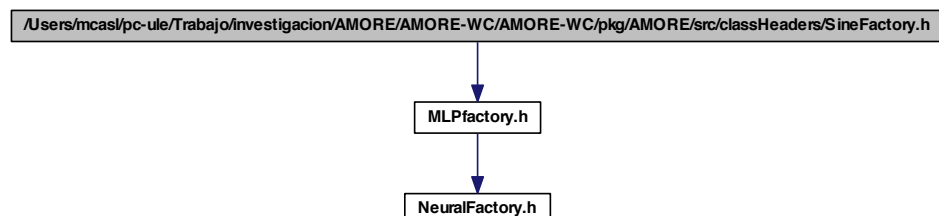
- class [Sine](#)

class [Sine](#) -

6.47 /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

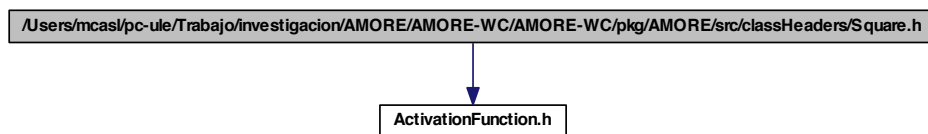
- class [SineFactory](#)

class [SineFactory](#) -

6.48 /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

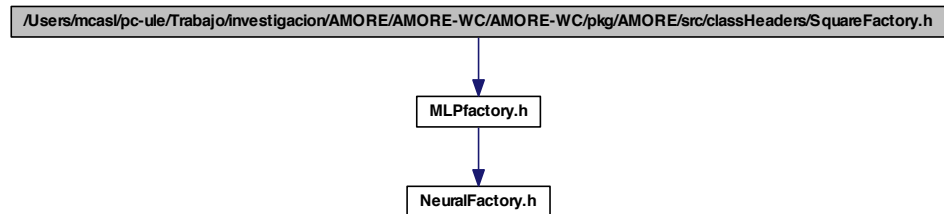
- class [Square](#)

class [Square](#) -

6.49 /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:



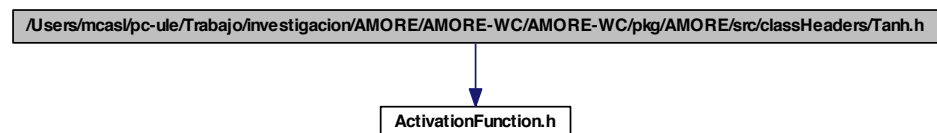
Classes

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

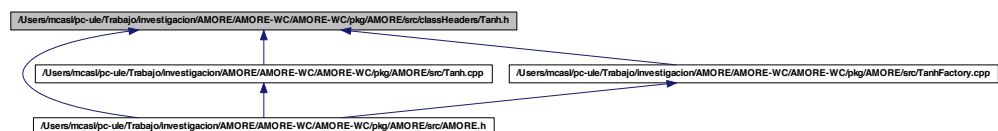
6.50 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Tanh.h File Reference

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

Include dependency graph for Tanh.h:



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



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

223

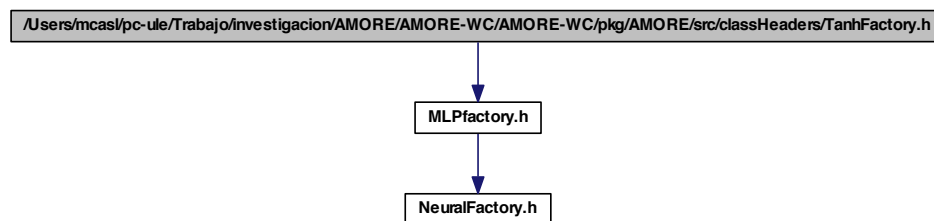
Reference Classes

- class [Tanh](#)
class [Tanh](#) -

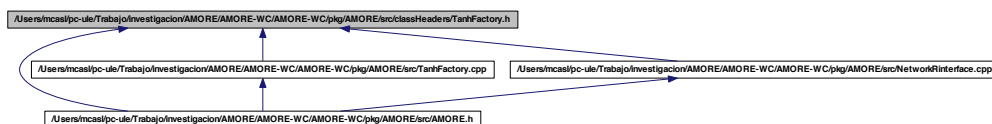
6.51 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/TanhFactory.h File Reference

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

Include dependency graph for TanhFactory.h:



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



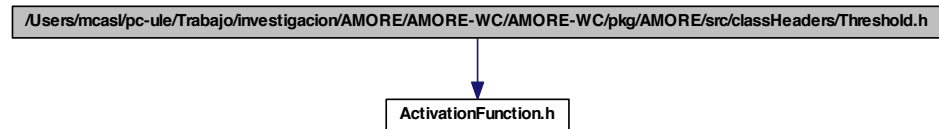
Classes

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

6.52 /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:



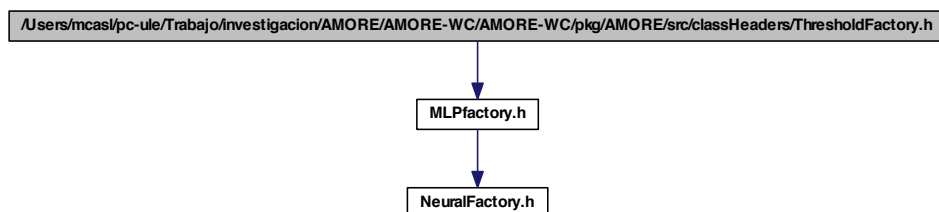
Classes

- class [Threshold](#)
class [Threshold](#) -

6.53 /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

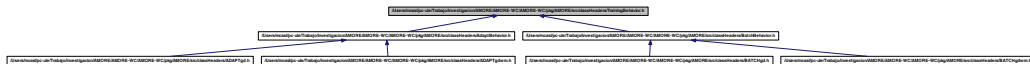
- class [ThresholdFactory](#)
class [ThresholdFactory](#) -

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

Reference

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

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



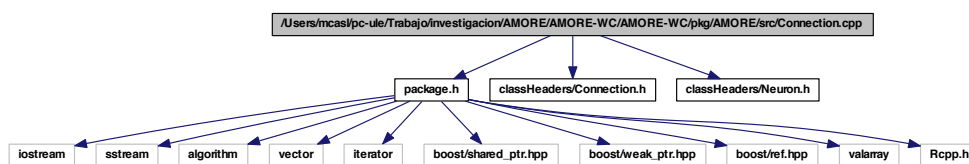
Classes

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

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

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

Include dependency graph for Connection.cpp:



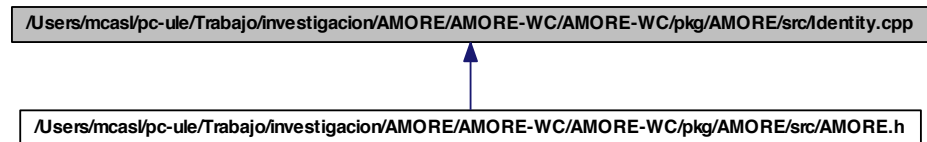
6.56 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Identity.cpp File Reference

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

Include dependency graph for Identity.cpp:



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



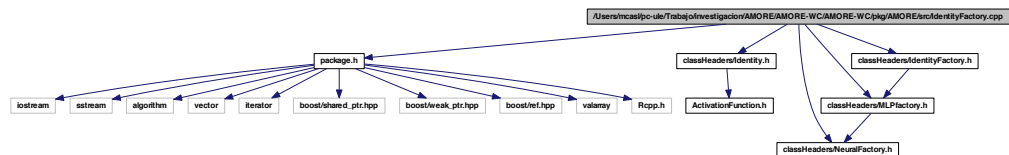
6.57 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/IdentityFactory.cpp File Reference

```

#include "package.h"
#include "classHeaders/Identity.h"
#include "classHeaders/NeuralFactory.h"
#include "classHeaders/MLPFactory.h"
#include "classHeaders/IdentityFactory.h"

```

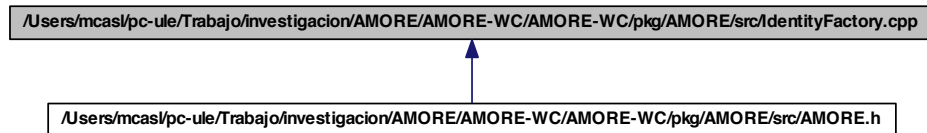
Include dependency graph for IdentityFactory.cpp:



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

227

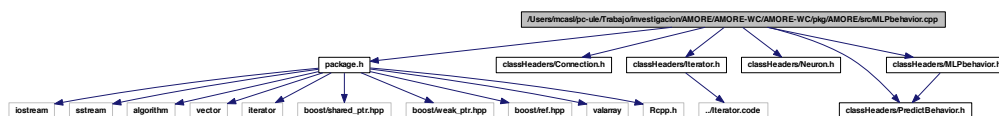
This graph shows which files directly or indirectly include this file:



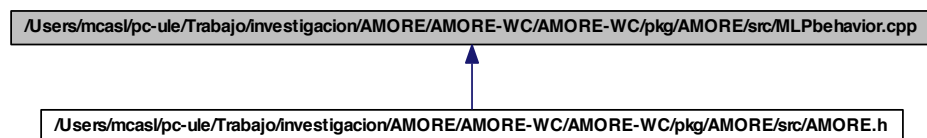
6.58 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPbehavior.cpp File Reference

```
#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:



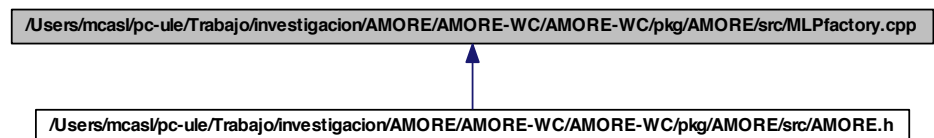
6.59 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPfactory.cpp File Reference

```
#include "package.h"
#include "classHeaders/Connection.h"
#include "classHeaders/Neuron.h"
#include "classHeaders/SimpleNeuron.h"
#include "classHeaders/Container.h"
#include "classHeaders/SimpleContainer.h"
#include "classHeaders/NeuralNetwork.h"
#include "classHeaders/SimpleNetwork.h"
#include "classHeaders/NeuralCreator.h"
#include "classHeaders/SimpleNeuralCreator.h"
#include "classHeaders/predictBehavior.h"
#include "classHeaders/MLPbehavior.h"
#include "classHeaders/Iterator.h"
#include "classHeaders/NeuralFactory.h"
#include "classHeaders/MLPfactory.h"
```

Include dependency graph for MLPfactory.cpp:



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



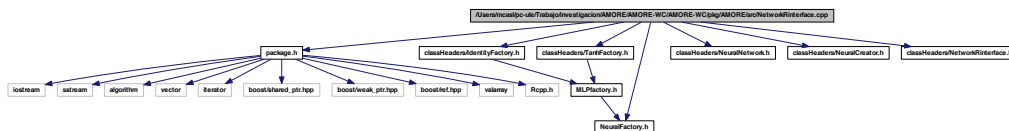
6.60 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NetworkRinterface.cpp File

Reference

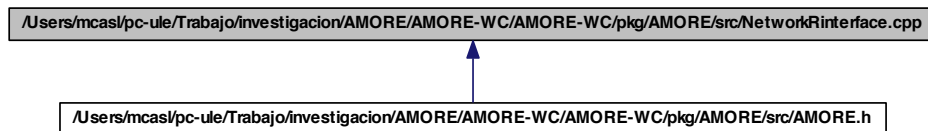
6.60 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NetworkRinterface.cpp File Reference

```
#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:



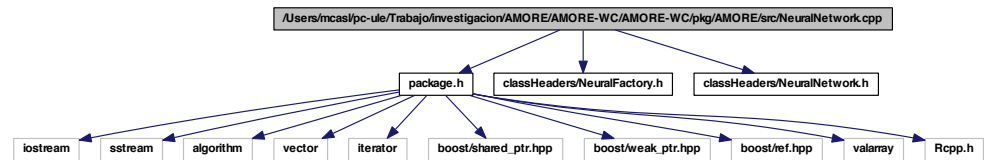
This graph shows which files directly or indirectly include this file:



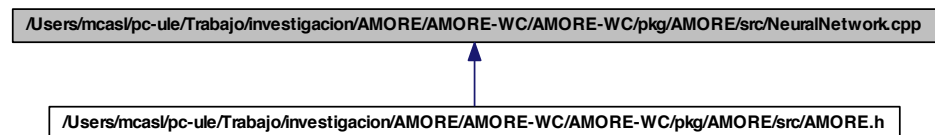
6.61 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NeuralNetwork.cpp File Reference

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

Include dependency graph for NeuralNetwork.cpp:



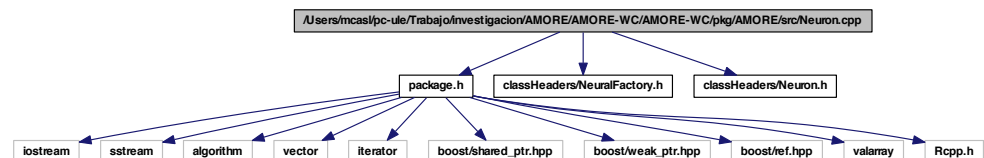
This graph shows which files directly or indirectly include this file:



6.62 /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:

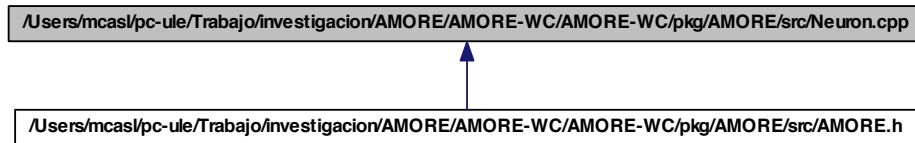


6.63 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/package.h File

Reference

231

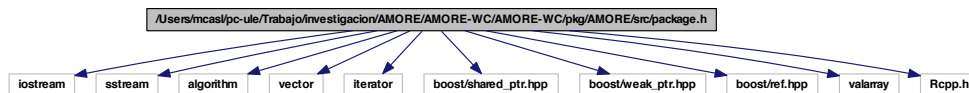
This graph shows which files directly or indirectly include this file:



6.63 /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< 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< Neuron > NeuronWeakPtr`

6.63.1 Define Documentation

6.63.1.1 `#define size_type unsigned int`

Definition at line 75 of file `package.h`.

6.63.2 Typedef Documentation

6.63.2.1 `typedef boost::shared_ptr<ActivationFunction> ActivationFunctionPtr`

Definition at line 85 of file `package.h`.

6.63.2.2 `typedef boost::reference_wrapper<PredictBehavior> ActivationFunctionRef`

Definition at line 80 of file `package.h`.

6.63.2.3 `typedef boost::shared_ptr<Container<ConPtr> > ConContainerPtr`

Definition at line 97 of file `package.h`.

6.63.2.4 `typedef boost::shared_ptr<Iterator<ConPtr>> ConIteratorPtr`

Definition at line 92 of file package.h.

6.63.2.5 `typedef boost::shared_ptr<Con> ConPtr`

Definition at line 88 of file package.h.

6.63.2.6 `typedef int Handler`

Definition at line 78 of file package.h.

6.63.2.7 `typedef boost::shared_ptr<Container<LayerPtr>> LayerContainerPtr`

Definition at line 95 of file package.h.

6.63.2.8 `typedef boost::shared_ptr<Container<NeuronPtr>> LayerPtr`

Definition at line 94 of file package.h.

6.63.2.9 `typedef boost::shared_ptr<NeuralCreator> NeuralCreatorPtr`

Definition at line 100 of file package.h.

6.63.2.10 `typedef boost::shared_ptr<NeuralFactory> NeuralFactoryPtr`

Definition at line 99 of file package.h.

6.63.2.11 `typedef boost::shared_ptr<NeuralNetwork> NeuralNetworkPtr`

Definition at line 89 of file package.h.

6.63.2.12 `typedef boost::shared_ptr<Iterator<NeuronPtr>> NeuronIteratorPtr`

Definition at line 91 of file package.h.

6.63.2.13 `typedef boost::shared_ptr<Neuron> NeuronPtr`

Definition at line 87 of file package.h.

6.63.2.14 `typedef boost::reference_wrapper<Neuron> NeuronRef`

Definition at line 83 of file package.h.

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

Definition at line 102 of file package.h.

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

Definition at line 86 of file package.h.

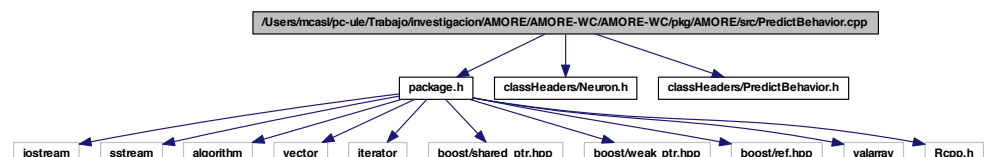
6.63.2.17 `typedef boost::reference_wrapper<PredictBehavior> PredictBehaviorRef`

Definition at line 81 of file package.h.

6.64 `/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:

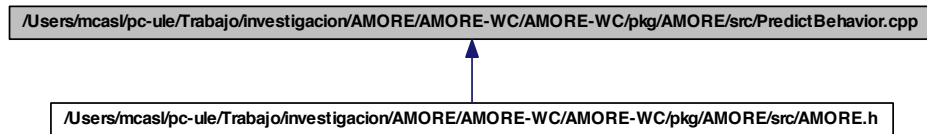


6.65 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/RcppModules.cpp File Reference

235

Reference

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

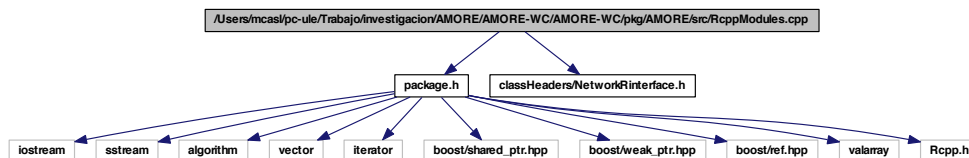


6.65 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/RcppModules.cpp File Reference

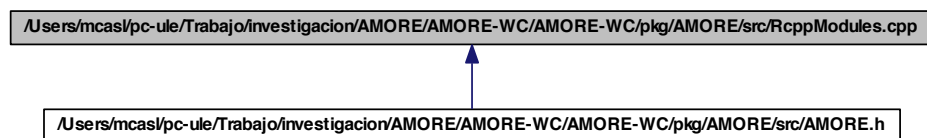
```
#include "package.h"
```

```
#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)

6.65.1 Function Documentation

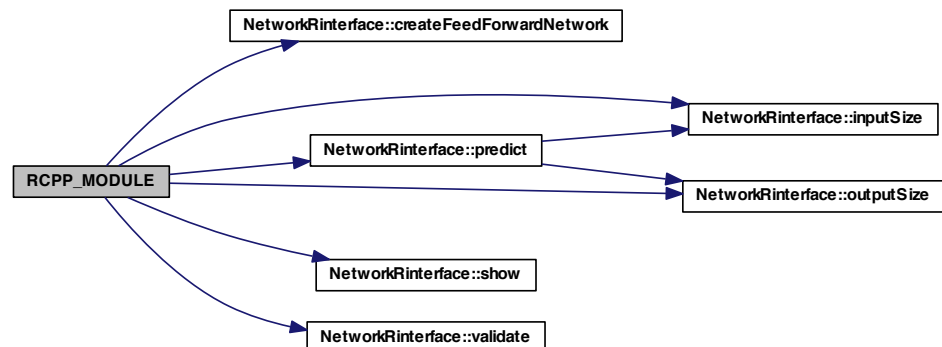
6.65.1.1 RCPP_MODULE (modAMORE)

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:



6.66 /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"
```

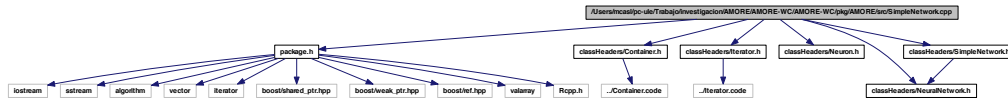
6.67 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuralCreator.cpp File Reference

237

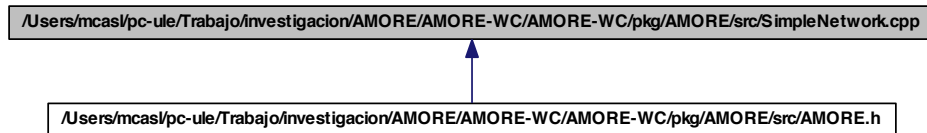
```
#include "classHeaders/NeuralNetwork.h"
```

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

Include dependency graph for SimpleNetwork.cpp:



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



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

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

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

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

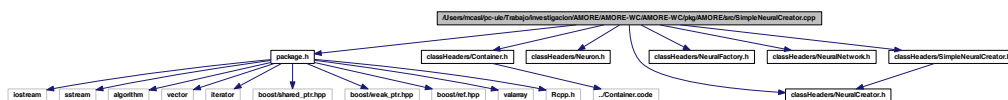
```
#include "classHeaders/NeuralCreator.h"
```

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

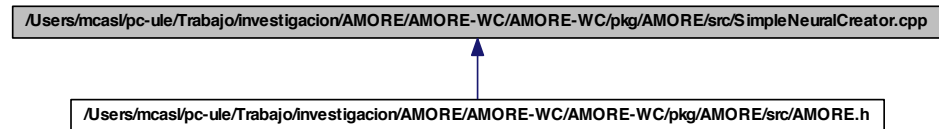
```
#include "classHeaders/NeuralNetwork.h"
```

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

Include dependency graph for SimpleNeuralCreator.cpp:



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



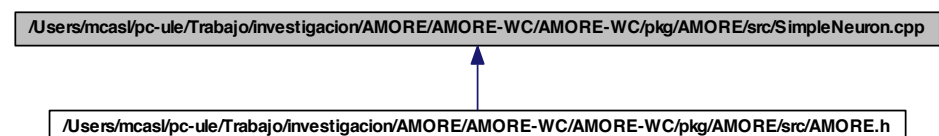
6.68 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuron.cpp File Reference

```
#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:



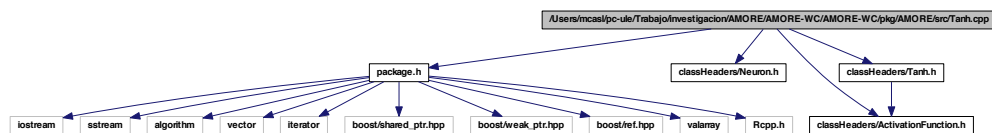
6.69 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Tanh.cpp File Reference

Reference

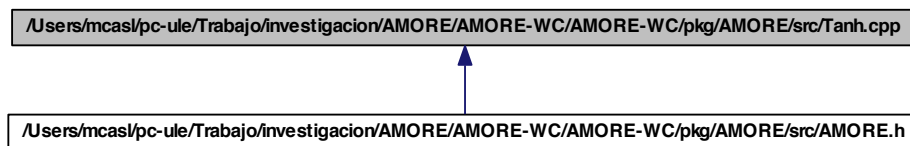
6.69 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Tanh.cpp File Reference

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

Include dependency graph for Tanh.cpp:



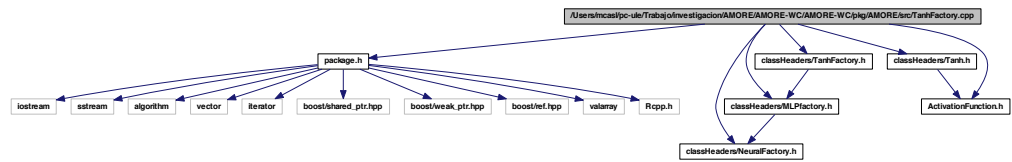
This graph shows which files directly or indirectly include this file:



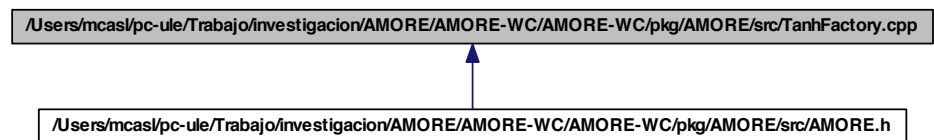
6.70 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/TanhFactory.cpp File Reference

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

Include dependency graph for TanhFactory.cpp:



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



Index

~Container	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
Container, 42	WC/AMORE-WC/pkg/AMORE/src/RcppModules.cpp,
~Iterator	235
Iterator, 75	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
~SimpleContainer	WC/AMORE-WC/pkg/AMORE/src/SimpleNetwork.cpp,
SimpleContainer, 138	236
~SimpleContainerIterator	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
SimpleContainerIterator, 142	WC/AMORE-WC/pkg/AMORE/src/SimpleNeuralCreator.cpp,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	
WC/AMORE-WC/pkg/AMORE/src/AMORE.h,	
190	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
	WC/AMORE-WC/pkg/AMORE/src/SimpleNeuron.cpp,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	
WC/AMORE-WC/pkg/AMORE/src/ActivationFunction.cpp,	
189	WC/AMORE-WC/pkg/AMORE/src/Tanh.cpp,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	
WC/AMORE-WC/pkg/AMORE/src/Connections.cpp,	
225	WC/AMORE-WC/pkg/AMORE/src/TanhFactory.cpp,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	
WC/AMORE-WC/pkg/AMORE/src/Utils.cpp,	
225	/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
	WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgd.h,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	
WC/AMORE-WC/pkg/AMORE/src/Utils/Factory.cpp,	
226	WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgdwm.h,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	
WC/AMORE-WC/pkg/AMORE/src/Utils/Model.cpp,	
227	WC/AMORE-WC/pkg/AMORE/src/classHeaders/ActivationFunction.h,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	
WC/AMORE-WC/pkg/AMORE/src/Utils/Primitives.cpp,	
228	WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptBehavior.h,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	
WC/AMORE-WC/pkg/AMORE/src/Utils/Random.cpp,	
229	WC/AMORE-WC/pkg/AMORE/src/classHeaders/ArcTan.h,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	
WC/AMORE-WC/pkg/AMORE/src/Utils/Neural.cpp,	
229	WC/AMORE-WC/pkg/AMORE/src/classHeaders/ArcTanFactory.h,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	
WC/AMORE-WC/pkg/AMORE/src/Utils/Nodes.cpp,	
230	WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgd.h,
/Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-	
WC/AMORE-WC/pkg/AMORE/src/Utils/Behavior.cpp,	
234	WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgdwm.h,

198 WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPbeh
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPbeh
 197 WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPfact
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPfact
 199 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Network
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Network
 199 WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralC
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralC
 200 WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralFa
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralFa
 201 WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralN
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralN
 201 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Neuron.f
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Neuron.f
 202 WC/AMORE-WC/pkg/AMORE/src/classHeaders/PredictB
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/PredictB
 202 WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFbeh
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFbeh
 203 WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFfact
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFfact
 203 WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBa
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBa
 204 WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBa
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Reciproco
 205 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Reciproco
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/Reciproco
 205 WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleC
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleC
 206 WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleC
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleC
 207 WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleC
 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-
 WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleC
 207 WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleN

- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE.h, 15
- WC/AMORE-WC/pkg/AMORE/src/AMORE.h, 15
- 218
- adjustParameters, 17
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE.h, 18
- WC/AMORE-WC/pkg/AMORE/src/AMORE.h, 18
- 219
- adjustParameters, 20
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE.h, 21
- WC/AMORE-WC/pkg/AMORE/src/AMORE.h, 21
- 220
- Neuron, 110
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE.h, 157
- WC/AMORE-WC/pkg/AMORE/src/AMORE.h, 157
- 220
- AdaptBehavior, 15
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE.h, 17
- WC/AMORE-WC/pkg/AMORE/src/AMORE.h, 17
- 221
- BatchBehavior, 28
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE.h, 30
- WC/AMORE-WC/pkg/AMORE/src/AMORE.h, 30
- 221
- TrainingBehavior, 188
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE.h, 192
- WC/AMORE-WC/pkg/AMORE/src/AMORE.h, 192
- 222
- ActivationFunctionRef, 192
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE.h, 192
- WC/AMORE-WC/pkg/AMORE/src/AMORE.h, 192
- 223
- Container, 42
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE.h, 192
- WC/AMORE-WC/pkg/AMORE/src/AMORE.h, 192
- 223
- LayerPtr, 193
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE.h, 193
- WC/AMORE-WC/pkg/AMORE/src/AMORE.h, 193
- 224
- NeuralNetworkPtr, 193
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE.h, 193
- WC/AMORE-WC/pkg/AMORE/src/AMORE.h, 193
- 225
- NeuronRef, 193
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE.h, 193
- WC/AMORE-WC/pkg/AMORE/src/AMORE.h, 193
- 231
- PredictBehaviorPtr, 193
- PredictBehaviorRef, 193
- size_type, 192
- TrainingBehaviorRef, 194
- ActivationFunction, 11
 - ActivationFunction, 12
 - d_neuron, 13
 - f0, 12
 - f1, 12
 - getInducedLocalField, 12
- ActivationFunctionPtr
 - AMORE.h, 192
 - package.h, 232
- ActivationFunctionRef
 - AMORE.h, 192
 - package.h, 232
- AdaptBehavior, 13
- ArcTan, 21
 - Arctan, 22
 - f0, 22
 - f1, 22
- Arctan
 - ArcTan, 22
- ArcTanFactory, 23
 - ArcTanFactory, 26
- makeActivationFunction, 26
- at
 - Container, 42
 - SimpleContainer, 138

- BatchBehavior, 26
 - adjustParameters, 28
- BATCHgd, 29
 - adjustParameters, 30
 - outputDerivative, 31
- BATCHgdwm, 31
 - adjustParameters, 33
 - outputDerivative, 34
- clear
 - Container, 42
 - SimpleContainer, 138
- Con, 34
 - Con, 35
 - d_neuron, 40
 - d_weight, 40
 - getNeuron, 35
 - getWeight, 36
 - Id, 37
 - setNeuron, 38
 - setWeight, 38
 - show, 38
 - validate, 39
- ConContainerPtr
 - AMORE.h, 192
 - package.h, 232
- ConIteratorPtr
 - AMORE.h, 192
 - package.h, 232
- ConPtr
 - AMORE.h, 192
 - package.h, 233
- Container, 40
 - ~Container, 42
 - at, 42
 - clear, 42
 - Container, 42
 - createIterator, 42
 - empty, 42
 - push_back, 43
 - reserve, 43
 - show, 43
 - size, 43
 - validate, 43
- Cosine, 43
 - Cosine, 45
 - f0, 45
 - f1, 46
- CosineFactory, 46
 - CosineFactory, 49
- makeActivationFunction, 49
- createFeedForwardNetwork
 - NetworkRinterface, 94
 - NeuralCreator, 99
 - SimpleNeuralCreator, 152
- createIterator
 - Container, 42
 - SimpleContainer, 138
- currentItem
 - Iterator, 75
 - SimpleContainerIterator, 142
- d_activationFunction
 - Neuron, 112
- d_altitude
 - RBFbehavior, 125
- d_bias
 - MLPbehavior, 85
- d_collection
 - SimpleContainer, 140
- d_container
 - SimpleContainerIterator, 143
- d_current
 - SimpleContainerIterator, 143
- d_hiddenLayers
 - NeuralNetwork, 106
- d_Id
 - Neuron, 112
- d_inducedLocalField
 - Neuron, 112
- d_inputLayer
 - NeuralNetwork, 107
- d_nCons
 - Neuron, 112
- d_neuralNetwork
 - NetworkRinterface, 98
- d_neuron
 - ActivationFunction, 13
 - Con, 40
 - PredictBehavior, 117
- d_output
 - Neuron, 112
- d_outputLayer
 - NeuralNetwork, 107
- d_predictBehavior
 - Neuron, 112
- d_weight
 - Con, 40
- d_width
 - RBFbehavior, 125

- Elliot, [49](#)
 - Elliot, [51](#)
 - f0, [51](#)
 - f1, [52](#)
- ElliotFactory, [52](#)
 - ElliotFactory, [55](#)
 - makeActivationFunction, [55](#)
- empty
 - Container, [42](#)
 - SimpleContainer, [139](#)
- Exponential, [55](#)
 - Exponential, [57](#)
 - f0, [57](#)
 - f1, [58](#)
- ExponentialFactory, [58](#)
 - ExponentialFactory, [61](#)
 - makeActivationFunction, [61](#)
- f0
 - ActivationFunction, [12](#)
 - ArcTan, [22](#)
 - Cosine, [45](#)
 - Elliot, [51](#)
 - Exponential, [57](#)
 - Gauss, [63](#)
 - Identity, [70](#)
 - Logistic, [77](#)
 - RadialBasis, [118](#)
 - Reciprocal, [131](#)
 - Sine, [164](#)
 - Square, [170](#)
 - Tanh, [177](#)
 - Threshold, [183](#)
- f1
 - ActivationFunction, [12](#)
 - ArcTan, [22](#)
 - Cosine, [46](#)
 - Elliot, [52](#)
 - Exponential, [58](#)
 - Gauss, [64](#)
 - Identity, [70](#)
 - Logistic, [78](#)
 - RadialBasis, [119](#)
 - Reciprocal, [132](#)
 - Sine, [165](#)
 - Square, [171](#)
 - Tanh, [177](#)
 - Threshold, [184](#)
- first
 - Iterator, [75](#)
 - SimpleContainerIterator, [142](#)
- Gauss, [61](#)
 - f0, [63](#)
 - f1, [64](#)
 - Gauss, [63](#)
- GaussFactory, [64](#)
 - GaussFactory, [67](#)
 - makeActivationFunction, [67](#)
- getConlterator
 - Neuron, [110](#)
 - PredictBehavior, [114](#)
 - SimpleNeuron, [158](#)
- getId
 - Neuron, [110](#)
 - SimpleNeuron, [158](#)
- getInducedLocalField
 - ActivationFunction, [12](#)
 - Neuron, [110](#)
 - SimpleNeuron, [158](#)
- getNeuron
 - Con, [35](#)
- getOutput
 - Neuron, [110](#)
 - SimpleNeuron, [159](#)
- getWeight
 - Con, [36](#)
- Handler
 - AMORE.h, [192](#)
 - package.h, [233](#)
- Id
 - Con, [37](#)
- Identity, [67](#)
 - f0, [70](#)
 - f1, [70](#)
 - Identity, [69](#)
- IdentityFactory, [70](#)
 - IdentityFactory, [73](#)
 - makeActivationFunction, [73](#)
- inputSize
 - NetworkRinterface, [94](#)
 - NeuralNetwork, [106](#)
 - SimpleNetwork, [146](#)
- isDone
 - Iterator, [75](#)
 - SimpleContainerIterator, [142](#)
- Iterator, [73](#)
 - ~Iterator, [75](#)

- currentItem, 75
- first, 75
- isDone, 75
- Iterator, 75
- next, 75
- LayerContainerPtr
 - AMORE.h, 192
 - package.h, 233
- LayerPtr
 - AMORE.h, 193
 - package.h, 233
- Logistic, 75
 - f0, 77
 - f1, 78
 - Logistic, 77
- LogisticFactory, 78
 - LogisticFactory, 81
 - makeActivationFunction, 81
- makeActivationFunction
 - ArcTanFactory, 26
 - CosineFactory, 49
 - ElliotFactory, 55
 - ExponentialFactory, 61
 - GaussFactory, 67
 - IdentityFactory, 73
 - LogisticFactory, 81
 - MLPfactory, 88
 - NeuralFactory, 100
 - RadialBasisFactory, 122
 - RBFfactory, 128
 - ReciprocalFactory, 135
 - SineFactory, 168
 - SquareFactory, 174
 - TanhFactory, 181
 - ThresholdFactory, 187
- makeCon
 - MLPfactory, 88
 - NeuralFactory, 100
 - RBFfactory, 128
- makeConContainer
 - MLPfactory, 89
 - NeuralFactory, 101
 - RBFfactory, 128
- makeLayer
 - MLPfactory, 89
 - NeuralFactory, 101
 - RBFfactory, 128
- makeLayerContainer
 - MLPfactory, 89
 - NeuralFactory, 101
 - RBFfactory, 128
- makeNeuralCreator
 - MLPfactory, 90
 - NeuralFactory, 102
 - RBFfactory, 128
- makeNeuralNetwork
 - MLPfactory, 90
 - NeuralFactory, 102
 - RBFfactory, 128
- makeNeuron
 - MLPfactory, 90, 91
 - NeuralFactory, 102, 103
 - RBFfactory, 129
- makePredictBehavior
 - MLPfactory, 92
 - NeuralFactory, 103
 - RBFfactory, 129
- MLPbehavior, 81
 - d_bias, 85
 - MLPbehavior, 84
 - MLPfactory, 85
 - predict, 84
 - show, 85
- MLPfactory, 86
 - makeActivationFunction, 88
 - makeCon, 88
 - makeConContainer, 89
 - makeLayer, 89
 - makeLayerContainer, 89
 - makeNeuralCreator, 90
 - makeNeuralNetwork, 90
 - makeNeuron, 90, 91
 - makePredictBehavior, 92
 - MLPbehavior, 85
 - Neuron, 112
- NetworkRinterface, 93
 - createFeedForwardNetwork, 94
 - d_neuralNetwork, 98
 - inputSize, 94
 - NetworkRinterface, 93
 - outputSize, 95
 - predict, 95
 - show, 96
 - validate, 97
- NeuralCreator, 98
 - createFeedForwardNetwork, 99
- NeuralCreatorPtr

- AMORE.h, 193
- package.h, 233
- NeuralFactory, 100
 - makeActivationFunction, 100
 - makeCon, 100
 - makeConContainer, 101
 - makeLayer, 101
 - makeLayerContainer, 101
 - makeNeuralCreator, 102
 - makeNeuralNetwork, 102
 - makeNeuron, 102, 103
 - makePredictBehavior, 103
- NeuralFactoryPtr
 - AMORE.h, 193
 - package.h, 233
- NeuralNetwork, 103
 - d_hiddenLayers, 106
 - d_inputLayer, 107
 - d_outputLayer, 107
 - inputSize, 106
 - NeuralNetwork, 105
 - outputSize, 106
 - predict, 106
 - readOutput, 106
 - show, 106
 - SimpleNeuralCreator, 106
 - validate, 106
 - writelnInput, 106
- NeuralNetworkPtr
 - AMORE.h, 193
 - package.h, 233
- Neuron, 107
 - addCon, 110
 - d_activationFunction, 112
 - d_id, 112
 - d_inducedLocalField, 112
 - d_nCons, 112
 - d_output, 112
 - d_predictBehavior, 112
 - getConIterator, 110
 - getId, 110
 - getInducedLocalField, 110
 - getOutput, 110
 - MLPfactory, 112
 - Neuron, 110
 - predict, 110
 - setActivationFunction, 111
 - setId, 111
 - setInducedLocalField, 111
 - setOutput, 111
 - setPredictBehavior, 111
 - show, 111
 - useActivationFunctionf0, 111
 - validate, 111
- NeuronIteratorPtr
 - AMORE.h, 193
 - package.h, 233
- NeuronPtr
 - AMORE.h, 193
 - package.h, 233
- NeuronRef
 - AMORE.h, 193
 - package.h, 233
- NeuronWeakPtr
 - AMORE.h, 193
 - package.h, 234
- next
 - Iterator, 75
 - SimpleContainerIterator, 142
- outputDerivative
 - ADAPTgd, 18
 - ADAPTgdwm, 21
 - BATCHgd, 31
 - BATCHgdwm, 34
- outputSize
 - NetworkRinterface, 95
 - NeuralNetwork, 106
 - SimpleNetwork, 146
- package.h
 - ActivationFunctionPtr, 232
 - ActivationFunctionRef, 232
 - ConContainerPtr, 232
 - ConIteratorPtr, 232
 - ConPtr, 233
 - Handler, 233
 - LayerContainerPtr, 233
 - LayerPtr, 233
 - NeuralCreatorPtr, 233
 - NeuralFactoryPtr, 233
 - NeuralNetworkPtr, 233
 - NeuronIteratorPtr, 233
 - NeuronPtr, 233
 - NeuronRef, 233
 - NeuronWeakPtr, 234
 - PredictBehaviorPtr, 234
 - PredictBehaviorRef, 234
 - size_type, 232
- predict

- MLPbehavior, 84
- NetworkRinterface, 95
- NeuralNetwork, 106
- Neuron, 110
- PredictBehavior, 115
- RBFbehavior, 125
- SimpleNetwork, 147
- SimpleNeuron, 159
- PredictBehavior, 113
 - d_neuron, 117
 - getConlterator, 114
 - predict, 115
 - PredictBehavior, 114
 - setInducedLocalField, 115
 - setOutput, 115
 - show, 116
 - useActivationFunctionf0, 116
- PredictBehaviorPtr
 - AMORE.h, 193
 - package.h, 234
- PredictBehaviorRef
 - AMORE.h, 193
 - package.h, 234
- push_back
 - Container, 43
 - SimpleContainer, 139
- RadialBasis, 117
 - f0, 118
 - f1, 119
 - RadialBasis, 118
- RadialBasisFactory, 119
 - makeActivationFunction, 122
 - RadialBasisFactory, 122
- RBFbehavior, 122
 - d_altitude, 125
 - d_width, 125
 - predict, 125
 - RBFbehavior, 125
 - show, 125
- RBFfactory, 125
 - makeActivationFunction, 128
 - makeCon, 128
 - makeConContainer, 128
 - makeLayer, 128
 - makeLayerContainer, 128
 - makeNeuralCreator, 128
 - makeNeuralNetwork, 128
 - makeNeuron, 129
 - makePredictBehavior, 129
- RCPP_MODULE
 - RcppModules.cpp, 236
- RcppModules.cpp
 - RCPP_MODULE, 236
- readOutput
 - NeuralNetwork, 106
 - SimpleNetwork, 148
- Reciprocal, 129
 - f0, 131
 - f1, 132
 - Reciprocal, 131
- ReciprocalFactory, 132
 - makeActivationFunction, 135
 - ReciprocalFactory, 135
- reserve
 - Container, 43
 - SimpleContainer, 139
- setActivationFunction
 - Neuron, 111
 - SimpleNeuron, 159
- setId
 - Neuron, 111
 - SimpleNeuron, 159
- setInducedLocalField
 - Neuron, 111
 - PredictBehavior, 115
 - SimpleNeuron, 160
- setNeuron
 - Con, 38
- setOutput
 - Neuron, 111
 - PredictBehavior, 115
 - SimpleNeuron, 160
- setPredictBehavior
 - Neuron, 111
 - SimpleNeuron, 160
- setWeight
 - Con, 38
- show
 - Con, 38
 - Container, 43
 - MLPbehavior, 85
 - NetworkRinterface, 96
 - NeuralNetwork, 106
 - Neuron, 111
 - PredictBehavior, 116
 - RBFbehavior, 125
 - SimpleContainer, 139
 - SimpleNetwork, 148

- SimpleNeuron, 160
- SimpleContainer, 135
 - ~SimpleContainer, 138
 - at, 138
 - clear, 138
 - createIterator, 138
 - d_collection, 140
 - empty, 139
 - push_back, 139
 - reserve, 139
 - show, 139
 - SimpleContainer, 138
 - SimpleContainerIterator< T >, 139
 - size, 139
 - validate, 139
- SimpleContainer< T >
 - SimpleContainerIterator, 143
- SimpleContainerIterator, 140
 - ~SimpleContainerIterator, 142
 - currentItem, 142
 - d_container, 143
 - d_current, 143
 - first, 142
 - isDone, 142
 - next, 142
 - SimpleContainer< T >, 143
 - SimpleContainerIterator, 142
- SimpleContainerIterator< T >
 - SimpleContainer, 139
- SimpleNetwork, 143
 - inputSize, 146
 - outputSize, 146
 - predict, 147
 - readOutput, 148
 - show, 148
 - SimpleNetwork, 146
 - validate, 149
 - writeInput, 149
- SimpleNeuralCreator, 150
 - createFeedForwardNetwork, 152
 - NeuralNetwork, 106
 - SimpleNeuralCreator, 151
- SimpleNeuron, 153
 - addCon, 157
 - getConIterator, 158
 - getId, 158
 - getInducedLocalField, 158
 - getOutput, 159
 - predict, 159
 - setActivationFunction, 159
 - setId, 159
 - setInducedLocalField, 160
 - setOutput, 160
 - setPredictBehavior, 160
 - show, 160
 - SimpleNeuron, 157
 - useActivationFunctionf0, 161
 - validate, 162
- Sine, 162
 - f0, 164
 - f1, 165
 - Sine, 164
- SineFactory, 165
 - makeActivationFunction, 168
 - SineFactory, 168
- size
 - Container, 43
 - SimpleContainer, 139
- size_type
 - AMORE.h, 192
 - package.h, 232
- Square, 168
 - f0, 170
 - f1, 171
 - Square, 170
- SquareFactory, 171
 - makeActivationFunction, 174
 - SquareFactory, 174
- Tanh, 174
 - f0, 177
 - f1, 177
 - Tanh, 176
- TanhFactory, 178
 - makeActivationFunction, 181
 - TanhFactory, 181
- Threshold, 181
 - f0, 183
 - f1, 184
 - Threshold, 183
- ThresholdFactory, 184
 - makeActivationFunction, 187
 - ThresholdFactory, 187
- TrainingBehavior, 187
 - adjustParameters, 188
- TrainingBehaviorRef
 - AMORE.h, 194
- useActivationFunctionf0
 - Neuron, 111

PredictBehavior, [116](#)

SimpleNeuron, [161](#)

validate

Con, [39](#)

Container, [43](#)

NetworkRinterface, [97](#)

NeuralNetwork, [106](#)

Neuron, [111](#)

SimpleContainer, [139](#)

SimpleNetwork, [149](#)

SimpleNeuron, [162](#)

writeInput

NeuralNetwork, [106](#)

SimpleNetwork, [149](#)