

AMORE++

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

Generated by Doxygen 1.7.4

Sat Jul 16 2011 10:53:54

Contents

1	The AMORE++ package	1
1.1	Introduction	1
1.2	Motivation	1
1.3	Road Map	1
2	Class Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Class Documentation	9
5.1	AdaptBehavior Class Reference	9
5.1.1	Detailed Description	11
5.1.2	Member Function Documentation	11
5.1.2.1	adjustParameters	11
5.2	ADAPTgd Class Reference	12
5.2.1	Detailed Description	13
5.2.2	Member Function Documentation	13
5.2.2.1	adjustParameters	14
5.2.3	Member Data Documentation	14
5.2.3.1	outputDerivative	14
5.3	ADAPTgdwm Class Reference	14
5.3.1	Detailed Description	16

5.3.2	Member Function Documentation	16
5.3.2.1	adjustParameters	17
5.3.3	Member Data Documentation	17
5.3.3.1	outputDerivative	17
5.4	BatchBehavior Class Reference	17
5.4.1	Detailed Description	19
5.4.2	Member Function Documentation	19
5.4.2.1	adjustParameters	19
5.5	BATCHgd Class Reference	20
5.5.1	Detailed Description	21
5.5.2	Member Function Documentation	21
5.5.2.1	adjustParameters	22
5.5.3	Member Data Documentation	22
5.5.3.1	outputDerivative	22
5.6	BATCHgdwm Class Reference	22
5.6.1	Detailed Description	24
5.6.2	Member Function Documentation	24
5.6.2.1	adjustParameters	25
5.6.3	Member Data Documentation	25
5.6.3.1	outputDerivative	25
5.7	Con Class Reference	25
5.7.1	Detailed Description	26
5.7.2	Constructor & Destructor Documentation	26
5.7.2.1	Con	26
5.7.2.2	Con	26
5.7.3	Member Function Documentation	26
5.7.3.1	getNeuron	26
5.7.3.2	getWeight	27
5.7.3.3	Id	28
5.7.3.4	setNeuron	29
5.7.3.5	setWeight	29
5.7.3.6	show	29
5.7.3.7	validate	30
5.7.4	Member Data Documentation	31

5.7.4.1	d_neuron	31
5.7.4.2	d_weight	31
5.8	Container< T > Class Template Reference	31
5.8.1	Detailed Description	33
5.8.2	Constructor & Destructor Documentation	33
5.8.2.1	~Container	33
5.8.2.2	Container	33
5.8.3	Member Function Documentation	33
5.8.3.1	at	33
5.8.3.2	clear	33
5.8.3.3	createIterator	34
5.8.3.4	empty	34
5.8.3.5	push_back	34
5.8.3.6	reserve	34
5.8.3.7	show	34
5.8.3.8	size	34
5.8.3.9	validate	34
5.9	Iterator< T > Class Template Reference	34
5.9.1	Detailed Description	36
5.9.2	Constructor & Destructor Documentation	36
5.9.2.1	~Iterator	36
5.9.2.2	Iterator	36
5.9.3	Member Function Documentation	36
5.9.3.1	currentItem	36
5.9.3.2	first	36
5.9.3.3	isDone	36
5.9.3.4	next	36
5.10	MLPbehavior Class Reference	37
5.10.1	Detailed Description	39
5.10.2	Member Function Documentation	39
5.10.2.1	predict	39
5.10.2.2	show	39
5.10.3	Friends And Related Function Documentation	39
5.10.3.1	MLPfactory	39

5.10.4	Member Data Documentation	39
5.10.4.1	d_accumulator	40
5.10.4.2	d_bias	40
5.10.4.3	d_nCons	40
5.10.4.4	d_output	40
5.11	MLPfactory Class Reference	40
5.11.1	Detailed Description	43
5.11.2	Constructor & Destructor Documentation	43
5.11.2.1	MLPfactory	43
5.11.3	Member Function Documentation	43
5.11.3.1	makeCon	43
5.11.3.2	makeCon	43
5.11.3.3	makeConContainer	43
5.11.3.4	makeNeuron	44
5.11.3.5	makeNeuronContainer	44
5.11.3.6	makePredictBehavior	45
5.12	NeuralCreator Class Reference	46
5.12.1	Detailed Description	46
5.12.2	Member Function Documentation	46
5.12.2.1	createCon	47
5.12.2.2	createNeuron	47
5.13	NeuralFactory Class Reference	47
5.13.1	Detailed Description	48
5.13.2	Member Function Documentation	49
5.13.2.1	makeCon	49
5.13.2.2	makeCon	49
5.13.2.3	makeConContainer	49
5.13.2.4	makeNeuron	49
5.13.2.5	makeNeuronContainer	50
5.13.2.6	makePredictBehavior	50
5.14	Neuron Class Reference	50
5.14.1	Detailed Description	51
5.14.2	Member Function Documentation	52
5.14.2.1	getId	52

5.14.2.2	setId	52
5.14.2.3	setPredictBehavior	52
5.14.2.4	show	52
5.14.2.5	validate	52
5.14.3	Member Data Documentation	52
5.14.3.1	d_predictBehavior	52
5.15	PredictBehavior Class Reference	53
5.15.1	Detailed Description	54
5.15.2	Member Function Documentation	54
5.15.2.1	predict	54
5.15.2.2	show	54
5.16	RBFbehavior Class Reference	54
5.16.1	Detailed Description	56
5.16.2	Member Function Documentation	57
5.16.2.1	predict	57
5.16.2.2	show	57
5.16.3	Member Data Documentation	57
5.16.3.1	d_accumulator	57
5.16.3.2	d_altitude	57
5.16.3.3	d_nCons	57
5.16.3.4	d_output	57
5.16.3.5	d_width	57
5.17	RBFfactory Class Reference	57
5.17.1	Detailed Description	60
5.17.2	Constructor & Destructor Documentation	60
5.17.2.1	RBFfactory	60
5.17.3	Member Function Documentation	60
5.17.3.1	makeCon	60
5.17.3.2	makeCon	60
5.17.3.3	makeConContainer	60
5.17.3.4	makeNeuron	60
5.17.3.5	makeNeuronContainer	60
5.17.3.6	makePredictBehavior	60
5.18	SimpleContainer< T > Class Template Reference	60

5.18.1 Detailed Description	63
5.18.2 Constructor & Destructor Documentation	63
5.18.2.1 SimpleContainer	63
5.18.2.2 ~SimpleContainer	64
5.18.3 Member Function Documentation	64
5.18.3.1 at	64
5.18.3.2 clear	65
5.18.3.3 createIterator	65
5.18.3.4 empty	65
5.18.3.5 push_back	66
5.18.3.6 reserve	66
5.18.3.7 show	66
5.18.3.8 size	67
5.18.3.9 validate	67
5.18.4 Friends And Related Function Documentation	68
5.18.4.1 SimpleContainerIterator< T >	68
5.18.5 Member Data Documentation	68
5.18.5.1 d_collection	68
5.19 SimpleContainerIterator< T > Class Template Reference	68
5.19.1 Detailed Description	71
5.19.2 Constructor & Destructor Documentation	71
5.19.2.1 SimpleContainerIterator	71
5.19.2.2 ~SimpleContainerIterator	71
5.19.3 Member Function Documentation	71
5.19.3.1 currentItem	71
5.19.3.2 first	72
5.19.3.3 isDone	72
5.19.3.4 next	72
5.19.4 Friends And Related Function Documentation	72
5.19.4.1 SimpleContainer< T >	72
5.19.5 Member Data Documentation	73
5.19.5.1 d_container	73
5.19.5.2 d_current	73
5.20 SimpleNeuralCreator Class Reference	73

5.20.1 Detailed Description	74
5.20.2 Constructor & Destructor Documentation	74
5.20.2.1 SimpleNeuralCreator	74
5.20.3 Member Function Documentation	75
5.20.3.1 createCon	75
5.20.3.2 createNeuron	75
5.21 SimpleNeuron Class Reference	76
5.21.1 Detailed Description	77
5.21.2 Constructor & Destructor Documentation	78
5.21.2.1 SimpleNeuron	78
5.21.3 Member Function Documentation	78
5.21.3.1 getId	78
5.21.3.2 setId	78
5.21.3.3 setPredictBehavior	79
5.21.3.4 show	79
5.21.3.5 validate	80
5.21.4 Member Data Documentation	80
5.21.4.1 d_Id	80
5.22 TrainingBehavior Class Reference	81
5.22.1 Detailed Description	81
5.22.2 Member Function Documentation	81
5.22.2.1 adjustParameters	81
6 File Documentation	83
6.1 pkg/AMORE/src/AMORE.h File Reference	83
6.1.1 Define Documentation	85
6.1.1.1 foreach	85
6.1.1.2 size_type	85
6.1.2 Typedef Documentation	85
6.1.2.1 ConContainerPtr	85
6.1.2.2 ConIteratorPtr	85
6.1.2.3 ConPtr	85
6.1.2.4 Handler	85
6.1.2.5 NeuralCreatorPtr	85

6.1.2.6	NeuralFactoryPtr	85
6.1.2.7	NeuronContainerPtr	85
6.1.2.8	NeuronIteratorPtr	85
6.1.2.9	NeuronPtr	86
6.1.2.10	NeuronRef	86
6.1.2.11	PredictBehaviorPtr	86
6.1.2.12	PredictBehaviorRef	86
6.1.2.13	TrainingBehaviorRef	86
6.2	pkg/AMORE/src/Con.cpp File Reference	86
6.3	pkg/AMORE/src/Container.cpp File Reference	87
6.4	pkg/AMORE/src/dia/AdaptBehavior.h File Reference	88
6.5	pkg/AMORE/src/dia/ADAPTgd.h File Reference	89
6.6	pkg/AMORE/src/dia/ADAPTgdwm.h File Reference	90
6.7	pkg/AMORE/src/dia/BatchBehavior.h File Reference	90
6.8	pkg/AMORE/src/dia/BATCHgd.h File Reference	91
6.9	pkg/AMORE/src/dia/BATCHgdwm.h File Reference	92
6.10	pkg/AMORE/src/dia/Con.h File Reference	94
6.11	pkg/AMORE/src/dia/Container.h File Reference	94
6.12	pkg/AMORE/src/dia/Iterator.h File Reference	95
6.13	pkg/AMORE/src/dia/MLPbehavior.h File Reference	95
6.14	pkg/AMORE/src/dia/MLPfactory.h File Reference	96
6.15	pkg/AMORE/src/dia/NeuralCreator.h File Reference	98
6.16	pkg/AMORE/src/dia/NeuralFactory.h File Reference	99
6.17	pkg/AMORE/src/dia/Neuron.h File Reference	100
6.18	pkg/AMORE/src/dia/PredictBehavior.h File Reference	101
6.19	pkg/AMORE/src/dia/RBFbehavior.h File Reference	101
6.20	pkg/AMORE/src/dia/RBFfactory.h File Reference	102
6.21	pkg/AMORE/src/dia/SimpleContainer.h File Reference	103
6.22	pkg/AMORE/src/dia/SimpleContainerIterator.h File Reference	103
6.23	pkg/AMORE/src/dia/SimpleNeuralCreator.h File Reference	104
6.24	pkg/AMORE/src/dia/SimpleNeuron.h File Reference	105
6.25	pkg/AMORE/src/dia/TrainingBehavior.h File Reference	107
6.26	pkg/AMORE/src/Iterator.cpp File Reference	107
6.27	pkg/AMORE/src/MLPbehavior.cpp File Reference	108

6.28	pkg/AMORE/src/MLPfactory.cpp File Reference	109
6.29	pkg/AMORE/src/SimpleContainer.cpp File Reference	110
6.30	pkg/AMORE/src/SimpleContainerIterator.cpp File Reference	111
6.31	pkg/AMORE/src/SimpleNeuralCreator.cpp File Reference	112
6.32	pkg/AMORE/src/SimpleNeuron.cpp File Reference	113

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:

Con	25
Container< T >	31
SimpleContainer< T >	60
Iterator< T >	34
SimpleContainerIterator< T >	68
NeuralCreator	46
SimpleNeuralCreator	73
NeuralFactory	47
MLPfactory	40
RBFfactory	57
Neuron	50
SimpleNeuron	76
PredictBehavior	53
MLPbehavior	37
RBFbehavior	54
TrainingBehavior	81
AdaptBehavior	9
ADAPTgd	12
ADAPTgdwm	14
BatchBehavior	17
BATCHgd	20
BATCHgdwm	22

Chapter 3

Class Index

3.1 Class List

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

AdaptBehavior (Class AdaptBehavior -)	9
ADAPTgd (Class ADAPTgd -)	12
ADAPTgdwm (Class ADAPTgdwm -)	14
BatchBehavior (Class BatchBehavior -)	17
BATCHgd (Class BATCHgd -)	20
BATCHgdwm (Class BATCHgdwm -)	22
Con (Class Con -)	25
Container< T > (Class Container -)	31
Iterator< T > (Class Iterator -)	34
MLPbehavior (Class MLPbehavior -)	37
MLPfactory (Class MLPfactory -)	40
NeuralCreator (Class NeuralCreator -)	46
NeuralFactory (Class NeuralFactory -)	47
Neuron (Class Neuron -)	50
PredictBehavior (Class PredictBehavior -)	53
RBFbehavior (Class RBFbehavior -)	54
RBFfactory (Class RBFfactory -)	57
SimpleContainer< T > (Class SimpleContainer -)	60
SimpleContainerIterator< T > (Class SimpleContainerIterator -)	68
SimpleNeuralCreator (Class SimpleNeuralCreator -)	73
SimpleNeuron (Class SimpleNeuron -)	76
TrainingBehavior (Class TrainingBehavior -)	81

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

pkg/AMORE/src/AMORE.h	83
pkg/AMORE/src/Con.cpp	86
pkg/AMORE/src/Container.cpp	87
pkg/AMORE/src/Iterator.cpp	107
pkg/AMORE/src/MLPbehavior.cpp	108
pkg/AMORE/src/MLPfactory.cpp	109
pkg/AMORE/src/SimpleContainer.cpp	110
pkg/AMORE/src/SimpleContainerIterator.cpp	111
pkg/AMORE/src/SimpleNeuralCreator.cpp	112
pkg/AMORE/src/SimpleNeuron.cpp	113
pkg/AMORE/src/dia/AdaptBehavior.h	88
pkg/AMORE/src/dia/ADAPTgd.h	89
pkg/AMORE/src/dia/ADAPTgdwm.h	90
pkg/AMORE/src/dia/BatchBehavior.h	90
pkg/AMORE/src/dia/BATCHgd.h	91
pkg/AMORE/src/dia/BATCHgdwm.h	92
pkg/AMORE/src/dia/Con.h	94
pkg/AMORE/src/dia/Container.h	94
pkg/AMORE/src/dia/Iterator.h	95
pkg/AMORE/src/dia/MLPbehavior.h	95
pkg/AMORE/src/dia/MLPfactory.h	96
pkg/AMORE/src/dia/NeuralCreator.h	98
pkg/AMORE/src/dia/NeuralFactory.h	99
pkg/AMORE/src/dia/Neuron.h	100
pkg/AMORE/src/dia/PredictBehavior.h	101
pkg/AMORE/src/dia/RBFbehavior.h	101
pkg/AMORE/src/dia/RBFfactory.h	102
pkg/AMORE/src/dia/SimpleContainer.h	103
pkg/AMORE/src/dia/SimpleContainerIterator.h	103

pkg/AMORE/src/dia/ SimpleNeuralCreator.h	104
pkg/AMORE/src/dia/ SimpleNeuron.h	105
pkg/AMORE/src/dia/ TrainingBehavior.h	107

Chapter 5

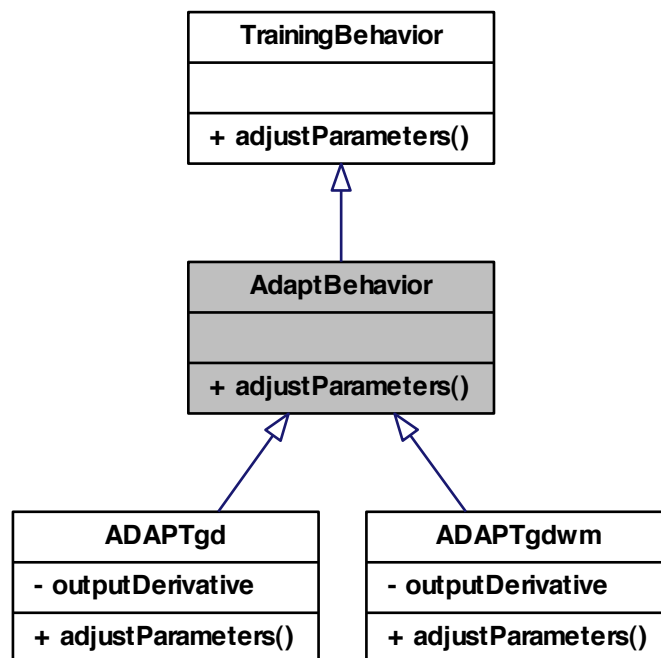
Class Documentation

5.1 `AdaptBehavior` Class Reference

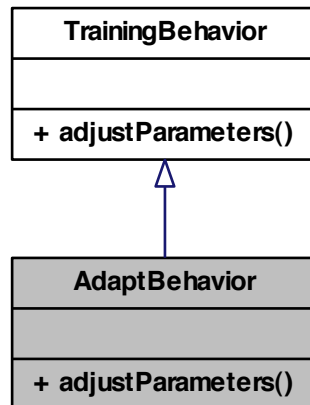
class `AdaptBehavior` -

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

Inheritance diagram for AdaptBehavior:



Collaboration diagram for AdaptBehavior:



Public Member Functions

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

5.1.1 Detailed Description

class [AdaptBehavior](#) -

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

5.1.2 Member Function Documentation

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

Reimplemented from [TrainingBehavior](#).

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

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

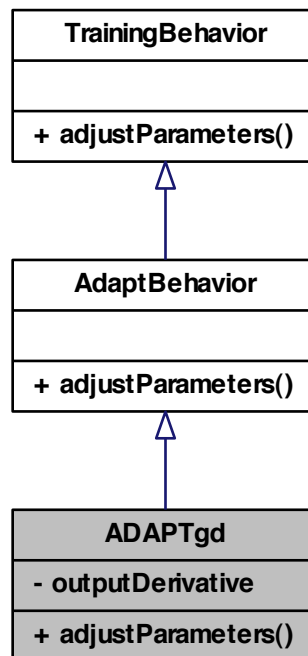
- [pkg/AMORE/src/dia/AdaptBehavior.h](#)

5.2 ADAPTgd Class Reference

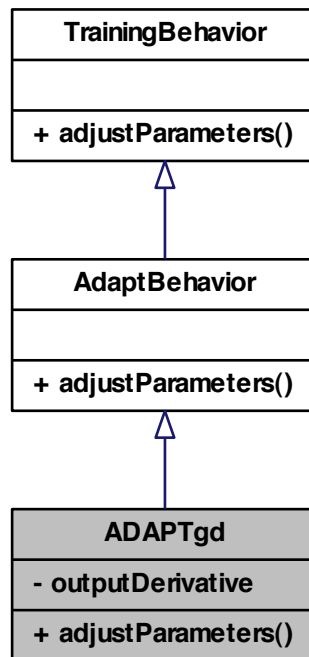
class [ADAPTgd](#) -

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

Inheritance diagram for ADAPTgd:



Collaboration diagram for ADAPTgd:



Public Member Functions

- void [adjustParameters](#) ()

Private Attributes

- double [outputDerivative](#)

5.2.1 Detailed Description

class [ADAPTgd](#) -

Definition at line 5 of file ADAPTgd.h.

5.2.2 Member Function Documentation

5.2.2.1 void ADAPTgd::adjustParameters () [virtual]

Implements [AdaptBehavior](#).

5.2.3 Member Data Documentation

5.2.3.1 double ADAPTgd::outputDerivative [private]

Definition at line 8 of file ADAPTgd.h.

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

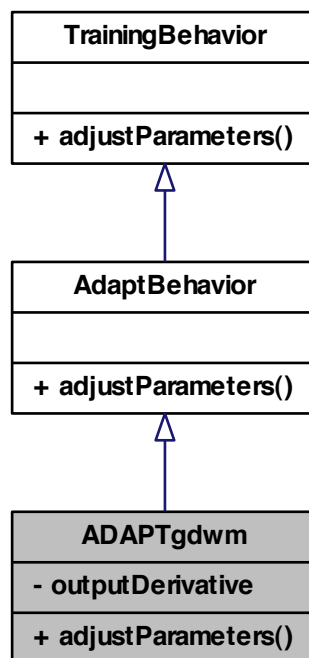
- pkg/AMORE/src/dia/[ADAPTgd.h](#)

5.3 ADAPTgdwm Class Reference

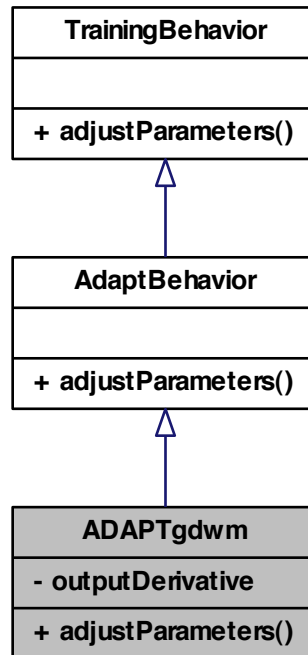
class [ADAPTgdwm](#) -

#include <ADAPTgdwm.h>

Inheritance diagram for ADAPTgdwm:



Collaboration diagram for ADAPTgdwm:



Public Member Functions

- void [adjustParameters](#) ()

Private Attributes

- double [outputDerivative](#)

5.3.1 Detailed Description

class [ADAPTgdwm](#) -

Definition at line 5 of file ADAPTgdwm.h.

5.3.2 Member Function Documentation

5.3.2.1 void ADAPTgdmw::adjustParameters () [virtual]

Implements [AdaptBehavior](#).

5.3.3 Member Data Documentation

5.3.3.1 double ADAPTgdmw::outputDerivative [private]

Definition at line 8 of file ADAPTgdmw.h.

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

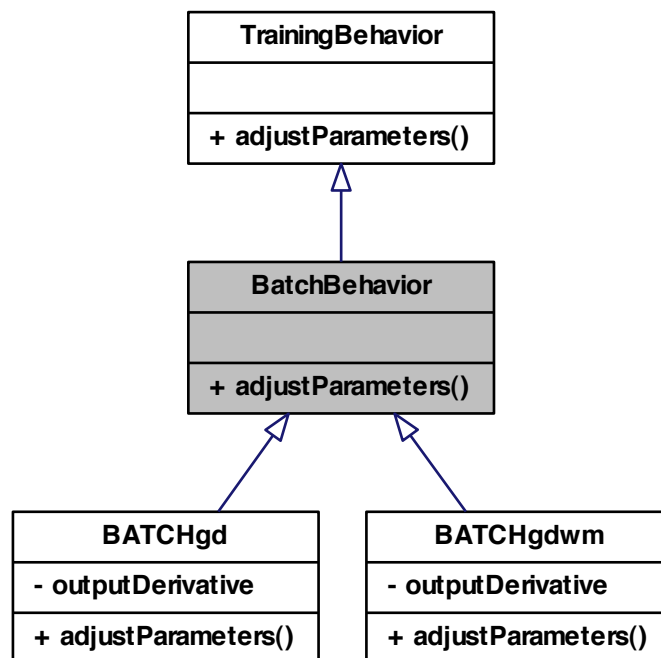
- pkg/AMORE/src/dia/[ADAPTgdmw.h](#)

5.4 BatchBehavior Class Reference

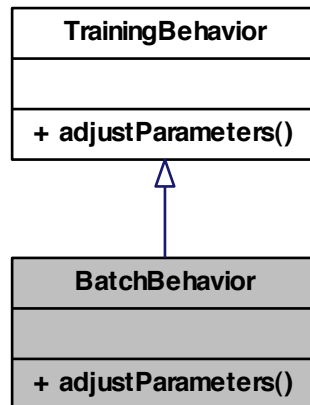
class [BatchBehavior](#) -

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

Inheritance diagram for BatchBehavior:



Collaboration diagram for BatchBehavior:



Public Member Functions

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

5.4.1 Detailed Description

class [BatchBehavior](#) -

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

5.4.2 Member Function Documentation

5.4.2.1 virtual void [BatchBehavior::adjustParameters](#) () `[pure virtual]`

Reimplemented from [TrainingBehavior](#).

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

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

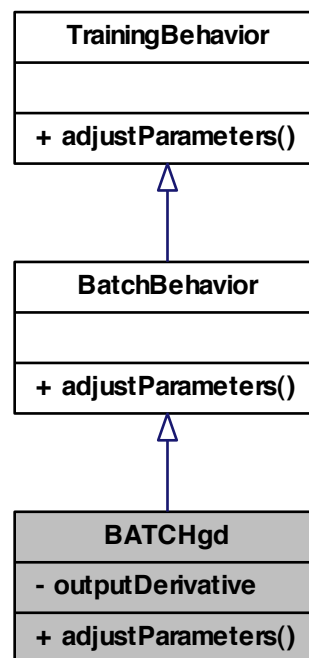
- [pkg/AMORE/src/dia/BatchBehavior.h](#)

5.5 BATCHgd Class Reference

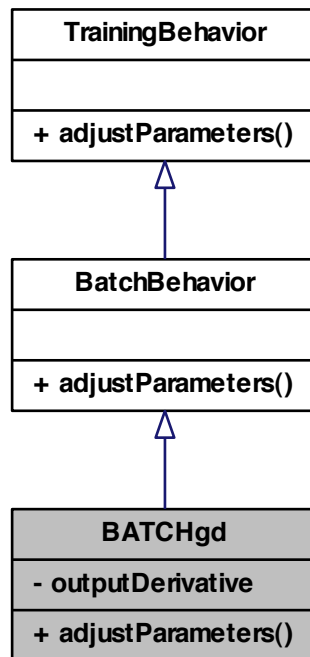
class [BATCHgd](#) -

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

Inheritance diagram for BATCHgd:



Collaboration diagram for BATCHgd:



Public Member Functions

- void [adjustParameters](#) ()

Private Attributes

- double [outputDerivative](#)

5.5.1 Detailed Description

class [BATCHgd](#) -

Definition at line 5 of file BATCHgd.h.

5.5.2 Member Function Documentation

5.5.2.1 void BATCHgd::adjustParameters () [virtual]

Implements [BatchBehavior](#).

5.5.3 Member Data Documentation

5.5.3.1 double BATCHgd::outputDerivative [private]

Definition at line 8 of file BATCHgd.h.

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

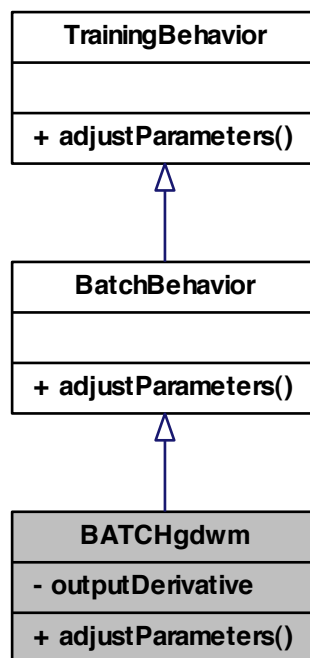
- pkg/AMORE/src/dia/[BATCHgd.h](#)

5.6 BATCHgdwm Class Reference

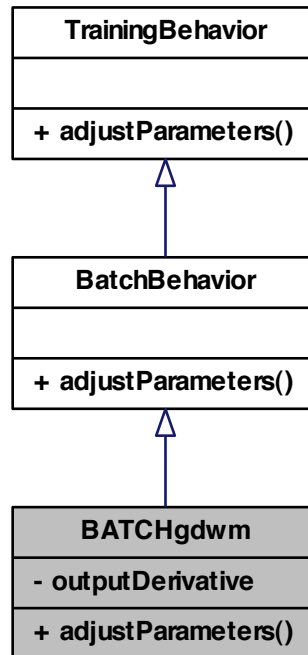
class [BATCHgdwm](#) -

#include <BATCHgdwm.h>

Inheritance diagram for BATCHgdwm:



Collaboration diagram for BATCHgdwm:



Public Member Functions

- void [adjustParameters](#) ()

Private Attributes

- double [outputDerivative](#)

5.6.1 Detailed Description

class [BATCHgdwm](#) -

Definition at line 5 of file BATCHgdwm.h.

5.6.2 Member Function Documentation

5.6.2.1 void BATCHgdwm::adjustParameters () [virtual]

Implements [BatchBehavior](#).

5.6.3 Member Data Documentation

5.6.3.1 double BATCHgdwm::outputDerivative [private]

Definition at line 8 of file BATCHgdwm.h.

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

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

5.7 Con Class Reference

class [Con](#) -

```
#include <Con.h>
```

Public Member Functions

- [Con](#) ([Neuron](#) &neuron)
Constructor.
- [Con](#) ([Neuron](#) &neuron, double weight)
Constructor.
- [Handler Id](#) ()
A getter of the Id of the [Neuron](#) pointed by the from field.
- [Neuron](#) & [getNeuron](#) ()
from field accessor.
- void [setNeuron](#) ([Neuron](#) &neuron)
- double [getWeight](#) ()
weight field accessor.
- void [setWeight](#) (double weight)
- void [show](#) ()
Pretty print of the [Con](#) information.
- bool [validate](#) ()
Object validator.

Private Attributes

- [NeuronRef](#) d_neuron
- double d_weight

5.7.1 Detailed Description

class [Con](#) -

Definition at line 3 of file Con.h.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 `Con::Con (Neuron & neuron)`

Constructor.

Definition at line 19 of file Con.cpp.

```

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

```

5.7.2.2 `Con::Con (Neuron & neuron, double weight)`

Constructor.

Definition at line 30 of file Con.cpp.

```

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

```

5.7.3 Member Function Documentation

5.7.3.1 `Neuron & Con::getNeuron ()`

from field accessor.

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

Returns

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

```

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

```

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

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

See also

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

Definition at line 56 of file `Con.cpp`.

References `d_neuron`.

```
{
    return d_neuron;
}
```

5.7.3.2 double Con::getWeight ()

weight field accessor.

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

Returns

The value of `weight` (double)

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

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

See also

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

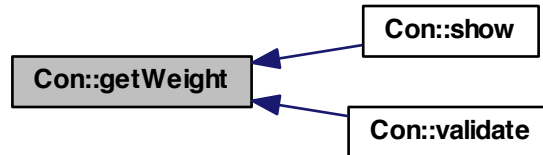
Definition at line 116 of file `Con.cpp`.

References `d_weight`.

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

```
{
    return d_weight;
}
```

Here is the caller graph for this function:



5.7.3.3 int Con::Id ()

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

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

Returns

The value of the Id (an integer).

```

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

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

See also

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

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

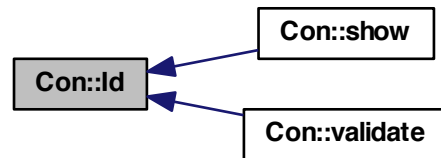
References `d_neuron`.

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

```

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


Here is the caller graph for this function:



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

Definition at line 63 of file Con.cpp.

References `d_neuron`.

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

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

Definition at line 123 of file Con.cpp.

References `d_weight`.

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

5.7.3.6 void Con::show ()

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

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

Returns

true in case everything works without throwing an exception

See also

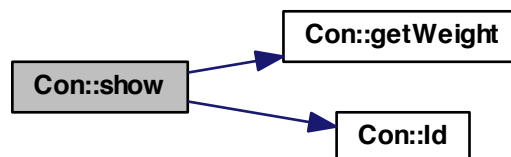
[setWeight](#) and the unit test files, e.g., `runit.Cpp.Con.R`, for usage examples.

Definition at line 135 of file Con.cpp.

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

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

Here is the call graph for this function:



5.7.3.7 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 155 of file Con.cpp.

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

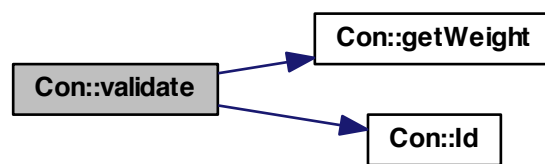
```
{
```

```

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

```

Here is the call graph for this function:



5.7.4 Member Data Documentation

5.7.4.1 NeuronRef Con::d_neuron [private]

Definition at line 6 of file Con.h.

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

5.7.4.2 double Con::d_weight [private]

Definition at line 7 of file Con.h.

Referenced by getWeight(), and setWeight().

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

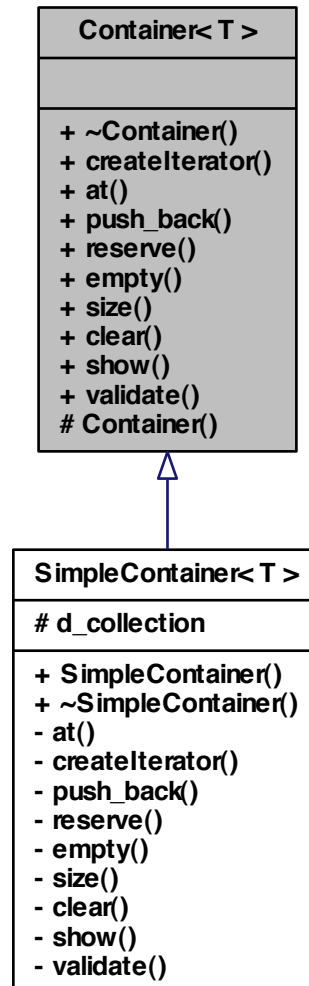
- pkg/AMORE/src/dia/[Con.h](#)
- pkg/AMORE/src/[Con.cpp](#)

5.8 Container< T > Class Template Reference

class [Container](#) -

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

Inheritance diagram for Container< T >:



Public Member Functions

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

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

Protected Member Functions

- [Container](#) ()

5.8.1 Detailed Description

template<typename T>class Container< T >

class [Container](#) -

Definition at line 5 of file Container.h.

5.8.2 Constructor & Destructor Documentation

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

Definition at line 20 of file Container.cpp.

```
{  
}
```

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

Definition at line 14 of file Container.cpp.

```
{  
}
```

5.8.3 Member Function Documentation

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

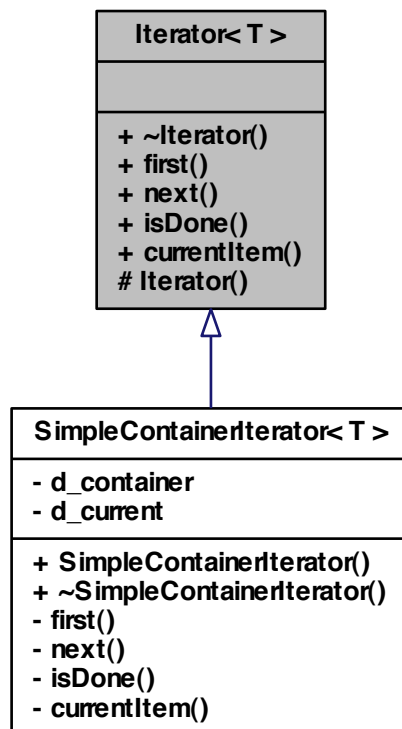
- pkg/AMORE/src/dia/[Container.h](#)
- pkg/AMORE/src/[Container.cpp](#)

5.9 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()`

5.9.1 Detailed Description

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

class [Iterator](#) -

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

5.9.2 Constructor & Destructor Documentation

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

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

```
{  
}
```

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

Definition at line 14 of file `Iterator.cpp`.

```
{  
}
```

5.9.3 Member Function Documentation

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

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

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

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

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

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

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

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

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

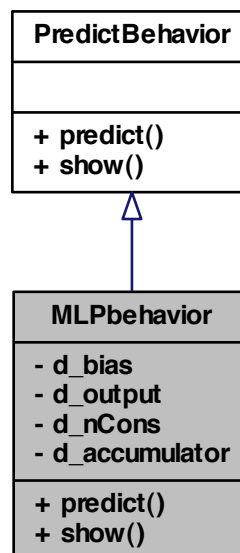
- [pkg/AMORE/src/dia/Iterator.h](#)
- [pkg/AMORE/src/Iterator.cpp](#)

5.10 MLPbehavior Class Reference

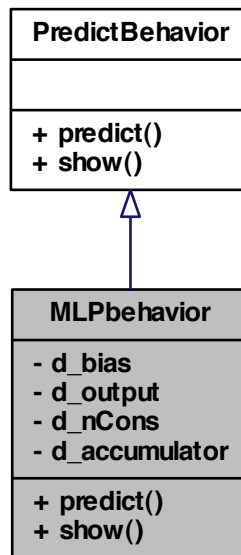
class [MLPbehavior](#) -

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

Inheritance diagram for MLPbehavior:



Collaboration diagram for MLPbehavior:



Public Member Functions

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

Private Attributes

- double [d_bias](#)
- double [d_output](#)
- [ConContainerPtr](#) [d_nCons](#)
- double [d_accumulator](#)

Friends

- class [MLPfactory](#)

5.10.1 Detailed Description

class [MLPbehavior](#) -

Definition at line 5 of file MLPbehavior.h.

5.10.2 Member Function Documentation

5.10.2.1 void MLPbehavior::predict() [virtual]

Implements [PredictBehavior](#).

Definition at line 15 of file MLPbehavior.cpp.

```
{  
}
```

5.10.2.2 void MLPbehavior::show() [virtual]

Implements [PredictBehavior](#).

Definition at line 22 of file MLPbehavior.cpp.

References [d_bias](#), [d_nCons](#), and [d_output](#).

```
{  
    Rprintf("\n bias: %lf", d_bias);  
    Rprintf("\n output: %lf", d_output);  
    Rprintf("\n-----\n");  
    if (d_nCons->size() == 0)  
    {  
        Rprintf("\n No connections defined");  
    }  
    else  
    {  
        d_nCons->show();  
    }  
    Rprintf("\n-----\n");  
}
```

5.10.3 Friends And Related Function Documentation

5.10.3.1 friend class MLPfactory [friend]

Definition at line 14 of file MLPbehavior.h.

5.10.4 Member Data Documentation

5.10.4.1 `double MLPbehavior::d_accumulator` [private]

Definition at line 11 of file MLPbehavior.h.

Referenced by MLPfactory::makePredictBehavior().

5.10.4.2 `double MLPbehavior::d_bias` [private]

Definition at line 8 of file MLPbehavior.h.

Referenced by MLPfactory::makePredictBehavior(), and show().

5.10.4.3 `ConContainerPtr MLPbehavior::d_nCons` [private]

Definition at line 10 of file MLPbehavior.h.

Referenced by MLPfactory::makePredictBehavior(), and show().

5.10.4.4 `double MLPbehavior::d_output` [private]

Definition at line 9 of file MLPbehavior.h.

Referenced by MLPfactory::makePredictBehavior(), and show().

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

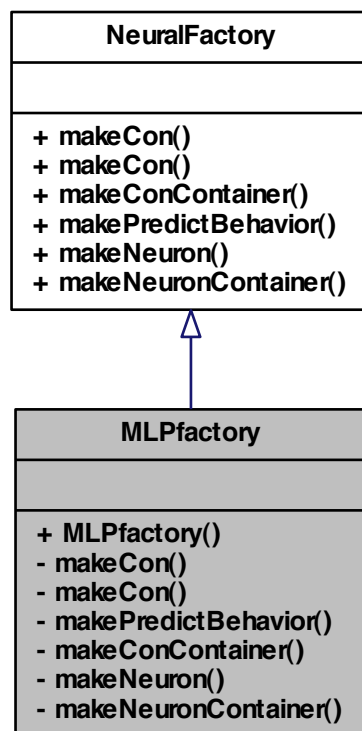
- pkg/AMORE/src/dia/[MLPbehavior.h](#)
- pkg/AMORE/src/[MLPbehavior.cpp](#)

5.11 MLPfactory Class Reference

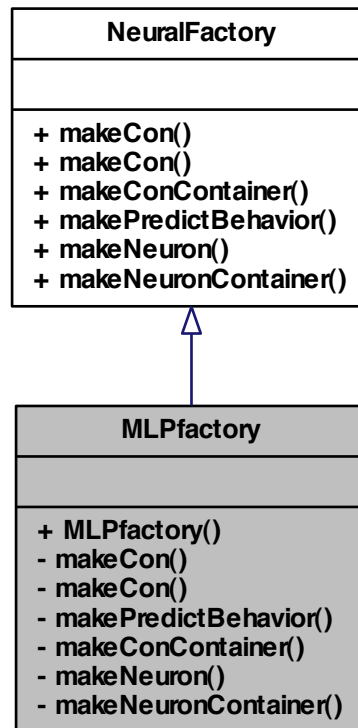
class [MLPfactory](#) -

```
#include <MLPfactory.h>
```

Inheritance diagram for MLPfactory:



Collaboration diagram for MLPfactory:



Public Member Functions

- [MLPfactory](#) ()

Private Member Functions

- [Con](#) * [makeCon](#) ([Neuron](#) &neuron)
- [Con](#) * [makeCon](#) ([Neuron](#) &neuron, double weight)
- [PredictBehavior](#) * [makePredictBehavior](#) ()
- [Container](#)< [ConPtr](#) > * [makeConContainer](#) ()
- [Neuron](#) * [makeNeuron](#) ()
- [Container](#)< [NeuronPtr](#) > * [makeNeuronContainer](#) ()

5.11.1 Detailed Description

class [MLPfactory](#) -

Definition at line 5 of file MLPfactory.h.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 MLPfactory::MLPfactory ()

Definition at line 13 of file MLPfactory.cpp.

```
{  
}
```

5.11.3 Member Function Documentation

5.11.3.1 Con * MLPfactory::makeCon (Neuron & *neuron*) [private, virtual]

Implements [NeuralFactory](#).

Definition at line 19 of file MLPfactory.cpp.

```
{  
    return new Con(neuron);  
}
```

5.11.3.2 Con * MLPfactory::makeCon (Neuron & *neuron*, double *weight*) [private, virtual]

Implements [NeuralFactory](#).

Definition at line 25 of file MLPfactory.cpp.

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

5.11.3.3 Container< ConPtr > * MLPfactory::makeConContainer () [private, virtual]

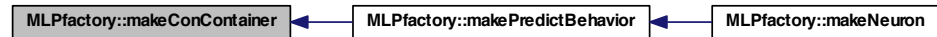
Implements [NeuralFactory](#).

Definition at line 31 of file MLPfactory.cpp.

Referenced by makePredictBehavior().

```
{
    return new SimpleContainer<ConPtr> ;
}
```

Here is the caller graph for this function:



5.11.3.4 `Neuron * MLPfactory::makeNeuron ()` [private, virtual]

Implements [NeuralFactory](#).

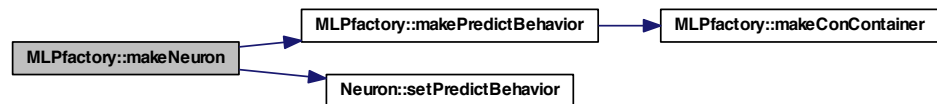
Definition at line 50 of file MLPfactory.cpp.

References `makePredictBehavior()`, and `Neuron::setPredictBehavior()`.

```
{
    Neuron* ptNeuron ( new SimpleNeuron() );
    ptNeuron->setPredictBehavior( makePredictBehavior() );

    return ptNeuron;
}
```

Here is the call graph for this function:



5.11.3.5 `Container< NeuronPtr > * MLPfactory::makeNeuronContainer ()` [private, virtual]

Implements [NeuralFactory](#).

Definition at line 62 of file MLPfactory.cpp.

```
{
    return new SimpleContainer<NeuronPtr> ;
}
```


5.11.3.6 PredictBehavior * MLPfactory::makePredictBehavior () [private, virtual]

Implements [NeuralFactory](#).

Definition at line 38 of file MLPfactory.cpp.

References `MLPbehavior::d_accumulator`, `MLPbehavior::d_bias`, `MLPbehavior::d_nCons`, `MLPbehavior::d_output`, and `makeConContainer()`.

Referenced by `makeNeuron()`.

```
{
    MLPbehavior* mlpBehavior( new MLPbehavior() );
    mlpBehavior->d_bias=0.0;
    mlpBehavior->d_output=0.0;
    mlpBehavior->d_accumulator=0.0;
    mlpBehavior->d_nCons.reset (makeConContainer());
    return mlpBehavior;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

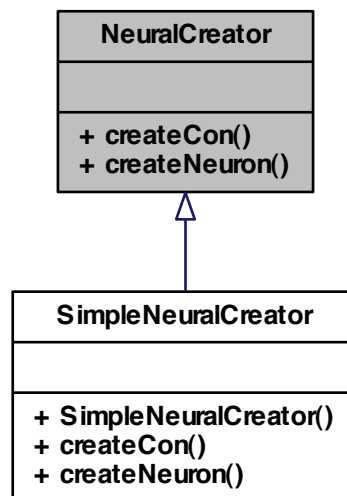
- `pkg/AMORE/src/dia/MLPfactory.h`
- `pkg/AMORE/src/MLPfactory.cpp`

5.12 NeuralCreator Class Reference

class [NeuralCreator](#) -

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

Inheritance diagram for NeuralCreator:



Public Member Functions

- virtual [Con](#) * `createCon` ([NeuralFactory](#) &neuralFactory, [Neuron](#) &neuron)=0
- virtual [Neuron](#) * `createNeuron` ([NeuralFactory](#) &neuralFactory)=0

5.12.1 Detailed Description

class [NeuralCreator](#) -

Definition at line 4 of file NeuralCreator.h.

5.12.2 Member Function Documentation

5.12.2.1 `virtual Con* NeuralCreator::createCon (NeuralFactory & neuralFactory, Neuron & neuron)` [pure virtual]

Implemented in [SimpleNeuralCreator](#).

5.12.2.2 `virtual Neuron* NeuralCreator::createNeuron (NeuralFactory & neuralFactory)`
[pure virtual]

Implemented in [SimpleNeuralCreator](#).

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

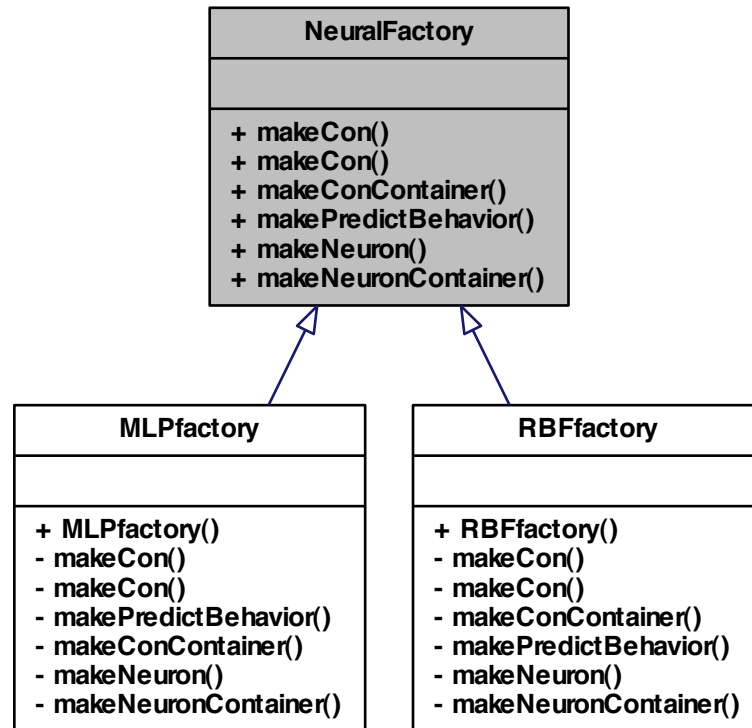
- `pkg/AMORE/src/dia/NeuralCreator.h`

5.13 NeuralFactory Class Reference

class [NeuralFactory](#) -

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

Inheritance diagram for NeuralFactory:



Public Member Functions

- virtual `Con * makeCon (Neuron &neuron)=0`
- virtual `Con * makeCon (Neuron &neuron, double weight)=0`
- virtual `Container< ConPtr > * makeConContainer ()=0`
- virtual `PredictBehavior * makePredictBehavior ()=0`
- virtual `Neuron * makeNeuron ()=0`
- virtual `Container< NeuronPtr > * makeNeuronContainer ()=0`

5.13.1 Detailed Description

class [NeuralFactory](#) -

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

5.13.2 Member Function Documentation

5.13.2.1 `virtual Con* NeuralFactory::makeCon (Neuron & neuron)` [pure virtual]

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

Referenced by `SimpleNeuralCreator::createCon()`.

Here is the caller graph for this function:



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

Implemented in [MLPfactory](#).

5.13.2.3 `virtual Container<ConPtr>* NeuralFactory::makeConContainer ()` [pure virtual]

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

5.13.2.4 `virtual Neuron* NeuralFactory::makeNeuron ()` [pure virtual]

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

Referenced by `SimpleNeuralCreator::createNeuron()`.

Here is the caller graph for this function:



5.13.2.5 `virtual Container<NeuronPtr>* NeuralFactory::makeNeuronContainer ()`
[pure virtual]

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

5.13.2.6 `virtual PredictBehavior* NeuralFactory::makePredictBehavior ()` [pure
virtual]

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

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

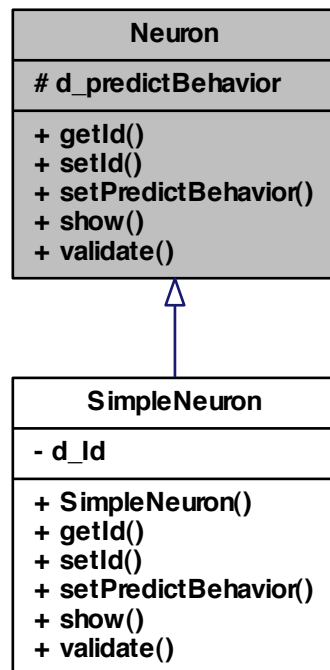
- `pkg/AMORE/src/dia/NeuralFactory.h`

5.14 Neuron Class Reference

class [Neuron](#) -

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

Inheritance diagram for Neuron:



Public Member Functions

- virtual `Handler getId ()=0`
- virtual void `setId (Handler Id)=0`
- virtual void `setPredictBehavior (PredictBehavior *predictBehavior)=0`
- virtual void `show ()=0`
- virtual bool `validate ()=0`

Protected Attributes

- `PredictBehaviorPtr d_predictBehavior`

5.14.1 Detailed Description

class `Neuron` -

Definition at line 3 of file Neuron.h.

5.14.2 Member Function Documentation

5.14.2.1 virtual Handler Neuron::getId () [pure virtual]

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

5.14.2.3 virtual void Neuron::setPredictBehavior (PredictBehavior * *predictBehavior*)
[pure virtual]

Implemented in [SimpleNeuron](#).

Referenced by MLPfactory::makeNeuron().

Here is the caller graph for this function:



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

Implemented in [SimpleNeuron](#).

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

Implemented in [SimpleNeuron](#).

5.14.3 Member Data Documentation

5.14.3.1 PredictBehaviorPtr Neuron::d_predictBehavior [protected]

Definition at line 6 of file Neuron.h.

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

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

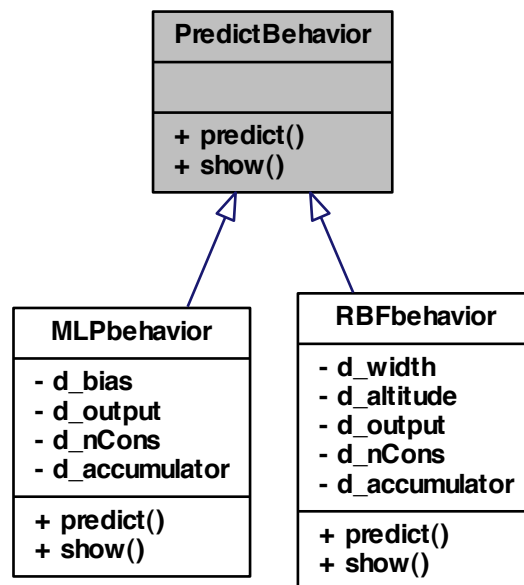
- pkg/AMORE/src/dia/Neuron.h

5.15 PredictBehavior Class Reference

class [PredictBehavior](#) -

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

Inheritance diagram for PredictBehavior:



Public Member Functions

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

5.15.1 Detailed Description

class [PredictBehavior](#) -

Definition at line 4 of file PredictBehavior.h.

5.15.2 Member Function Documentation

5.15.2.1 virtual void [PredictBehavior::predict](#) () [pure virtual]

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

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

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

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

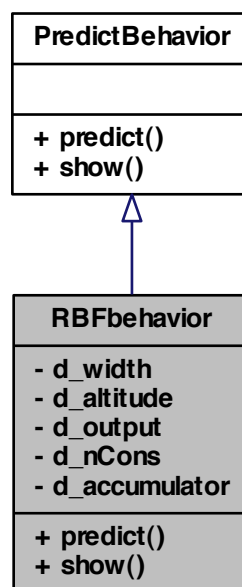
- pkg/AMORE/src/dia/[PredictBehavior.h](#)

5.16 RBFbehavior Class Reference

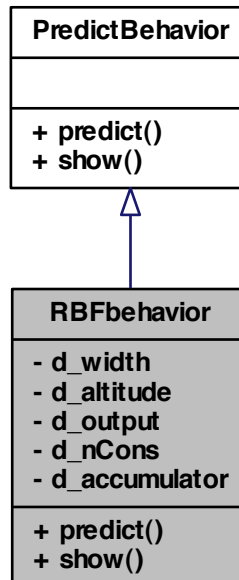
class [RBFbehavior](#) -

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

Inheritance diagram for RBFbehavior:



Collaboration diagram for RBFbehavior:



Public Member Functions

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

Private Attributes

- double [d_width](#)
- double [d_altitude](#)
- double [d_output](#)
- [ConContainerPtr](#) [d_nCons](#)
- double [d_accumulator](#)

5.16.1 Detailed Description

class [RBFbehavior](#) -

Definition at line 5 of file RBFbehavior.h.

5.16.2 Member Function Documentation

5.16.2.1 `void RBFbehavior::predict () [virtual]`

Implements [PredictBehavior](#).

5.16.2.2 `void RBFbehavior::show () [virtual]`

Implements [PredictBehavior](#).

5.16.3 Member Data Documentation

5.16.3.1 `double RBFbehavior::d_accumulator [private]`

Definition at line 12 of file RBFbehavior.h.

5.16.3.2 `double RBFbehavior::d_altitude [private]`

Definition at line 9 of file RBFbehavior.h.

5.16.3.3 `ConContainerPtr RBFbehavior::d_nCons [private]`

Definition at line 11 of file RBFbehavior.h.

5.16.3.4 `double RBFbehavior::d_output [private]`

Definition at line 10 of file RBFbehavior.h.

5.16.3.5 `double RBFbehavior::d_width [private]`

Definition at line 8 of file RBFbehavior.h.

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

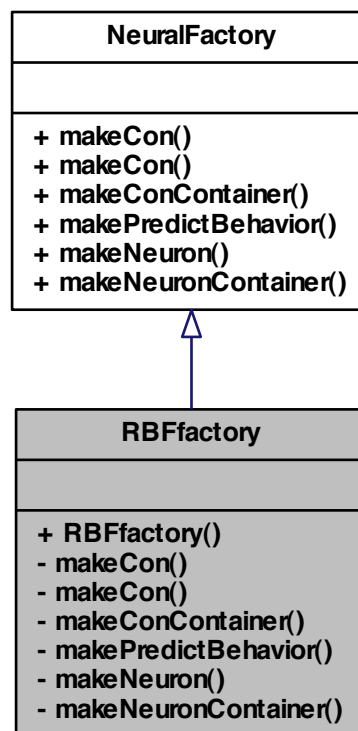
- `pkg/AMORE/src/dia/RBFbehavior.h`

5.17 RBFfactory Class Reference

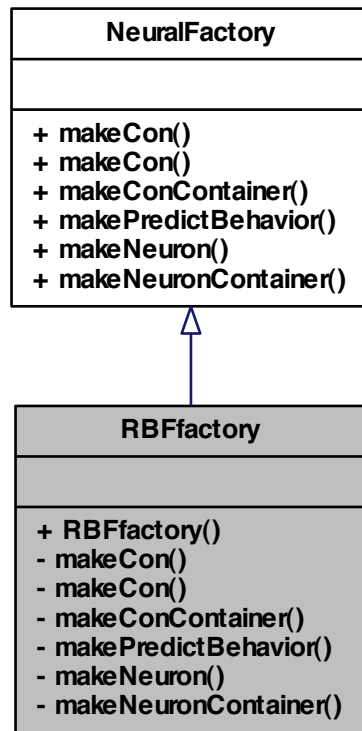
class [RBFfactory](#) -

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

Inheritance diagram for RBFactory:



Collaboration diagram for RBFactory:



Public Member Functions

- [RBFactory \(\)](#)

Private Member Functions

- [Con * makeCon \(Neuron *neuron, double weight\)](#)
- [Con * makeCon \(Neuron &neuron\)](#)
- [Container< ConPtr > * makeConContainer \(\)](#)
- [PredictBehavior * makePredictBehavior \(\)](#)
- [Neuron * makeNeuron \(\)](#)
- [Container< NeuronPtr > * makeNeuronContainer \(\)](#)

5.17.1 Detailed Description

class [RBFactory](#) -

Definition at line 5 of file RBFactory.h.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 RBFactory::RBFactory ()

5.17.3 Member Function Documentation

5.17.3.1 Con* RBFactory::makeCon (Neuron * *neuron*, double *weight*) [private]

5.17.3.2 Con* RBFactory::makeCon (Neuron & *neuron*) [private, virtual]

Implements [NeuralFactory](#).

5.17.3.3 Container<ConPtr>* RBFactory::makeConContainer () [private, virtual]

Implements [NeuralFactory](#).

5.17.3.4 Neuron* RBFactory::makeNeuron () [private, virtual]

Implements [NeuralFactory](#).

5.17.3.5 Container<NeuronPtr>* RBFactory::makeNeuronContainer () [private, virtual]

Implements [NeuralFactory](#).

5.17.3.6 PredictBehavior* RBFactory::makePredictBehavior () [private, virtual]

Implements [NeuralFactory](#).

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

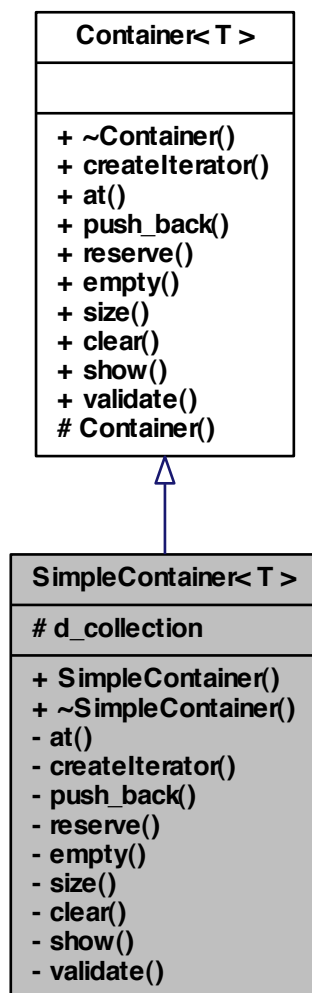
- pkg/AMORE/src/dia/[RBFactory.h](#)

5.18 SimpleContainer< T > Class Template Reference

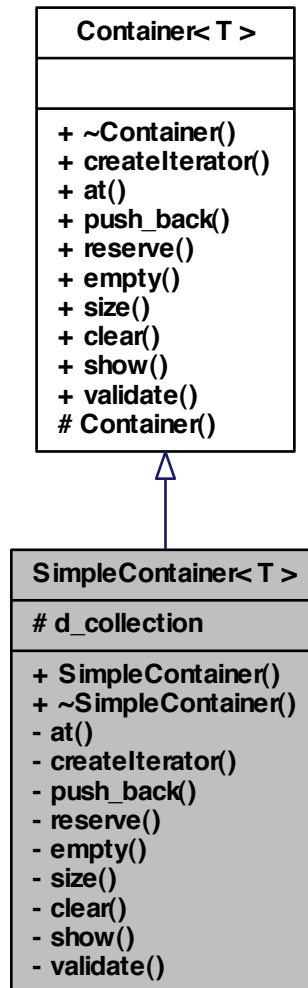
class [SimpleContainer](#) -


```
#include <SimpleContainer.h>
```

Inheritance diagram for SimpleContainer< T >:



Collaboration diagram for SimpleContainer< T >:



Public Member Functions

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

Protected Attributes

- `std::vector< T > d_collection`

Private Member Functions

- `T at (size_type element)`
Append a shared_ptr at the end of collection.
- `boost::shared_ptr< iterator< T > > createIterator ()`
- `void push_back (T const &const_reference)`
- `void reserve (int n)`
- `bool empty ()`
- `size_type size ()`
Returns the size or length of the vector.
- `void clear ()`
- `void show ()`
Pretty print of the SimpleContainer<T>
- `bool validate ()`
Object validator.

Friends

- class `SimpleContainerIterator< T >`

5.18.1 Detailed Description

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

class `SimpleContainer` -

Definition at line 6 of file SimpleContainer.h.

5.18.2 Constructor & Destructor Documentation

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

Definition at line 11 of file SimpleContainer.cpp.

```
{
}
```

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

Definition at line 17 of file SimpleContainer.cpp.

```
{
}
```

5.18.3 Member Function Documentation

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

Append a shared_ptr at the end of collection.

Implements push_back for the [Container](#) class

Parameters

<i>TsharedPtr</i>	A shared_ptr pointer to be inserted at the end of collection
-------------------	--

```
//=====
//Usage example:
//=====
// Data set up
Neuron N1, N2, N3;
Container<Con> conContainer;
std::vector<ConPtr> vc;
std::vector<int> result;
N1.setId(10);
N2.setId(20);
N3.setId(30);

// Test
ConPtr ptCon( new Con(&N1, 1.13) ); // Create new Con
and initialize ptCon
conContainer.push_back(ptCon); /
/ push_back
ptCon.reset( new Con(&N2, 2.22) ); // create
new Con and assign to ptCon
conContainer.push_back(ptCon); /
/ push_back
ptCon.reset( new Con(&N3, 3.33) ); // create
new Con and assign to ptCon
conContainer.push_back(ptCon); /
/ push_back

vc = conContainer.load();

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

See also

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

Implements [Container< T >](#).

Definition at line 69 of file SimpleContainer.cpp.

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

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

Implements [Container< T >](#).

Definition at line 182 of file SimpleContainer.cpp.

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

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

Implements [Container< T >](#).

Definition at line 23 of file SimpleContainer.cpp.

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

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

Implements [Container< T >](#).

Definition at line 168 of file SimpleContainer.cpp.

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

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

Implements [Container<T>](#).

Definition at line 77 of file SimpleContainer.cpp.

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

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

Implements [Container<T>](#).

Definition at line 175 of file SimpleContainer.cpp.

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

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

Pretty print of the SimpleContainer<T>

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

Returns

true in case everything works without throwing an exception

*

```

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

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

```

```

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

    // Test
    conContainerPtr->show() ;

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

```

See also

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

Implements [Container< T >](#).

Definition at line 127 of file `SimpleContainer.cpp`.

```

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

```

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

Returns the size or length of the vector.

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

Implements [Container< T >](#).

Definition at line 160 of file `SimpleContainer.cpp`.

```

{
    return d_collection.size();
}

```

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

Object validator.

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

See also

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

Implements [Container< T >](#).

Definition at line 142 of file `SimpleContainer.cpp`.

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

5.18.4 Friends And Related Function Documentation

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

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

5.18.5 Member Data Documentation

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

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

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

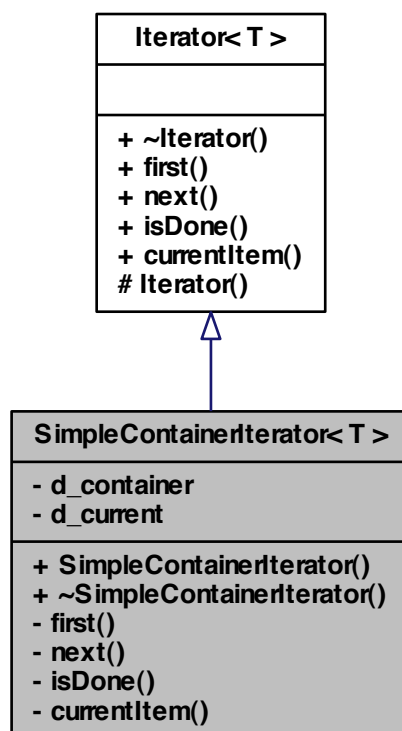
- `pkg/AMORE/src/dia/`[SimpleContainer.h](#)
- `pkg/AMORE/src/`[SimpleContainer.cpp](#)

5.19 SimpleContainerIterator< T > Class Template Reference

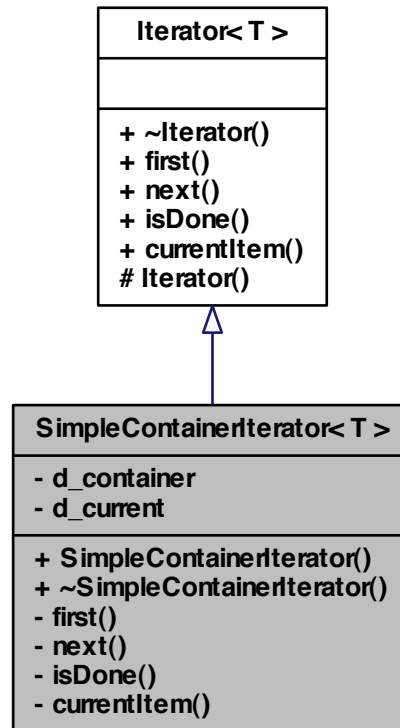
class [SimpleContainerIterator](#) -

`#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 >](#)

5.19.1 Detailed Description

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

class [SimpleContainerIterator](#) -

Definition at line 6 of file SimpleContainerIterator.h.

5.19.2 Constructor & Destructor Documentation

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

Definition at line 4 of file SimpleContainerIterator.cpp.

```
{  
}
```

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

Definition at line 9 of file SimpleContainerIterator.cpp.

```
{  
}
```

5.19.3 Member Function Documentation

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

Implements [Iterator< T >](#).

Definition at line 37 of file SimpleContainerIterator.cpp.

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

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

Implements [Iterator< T>](#).

Definition at line 15 of file SimpleContainerIterator.cpp.

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

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

Implements [Iterator< T>](#).

Definition at line 29 of file SimpleContainerIterator.cpp.

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

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

Implements [Iterator< T>](#).

Definition at line 22 of file SimpleContainerIterator.cpp.

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

5.19.4 Friends And Related Function Documentation

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

Definition at line 13 of file SimpleContainerIterator.h.

5.19.5 Member Data Documentation

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

Definition at line 9 of file SimpleContainerIterator.h.

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

Definition at line 10 of file SimpleContainerIterator.h.

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

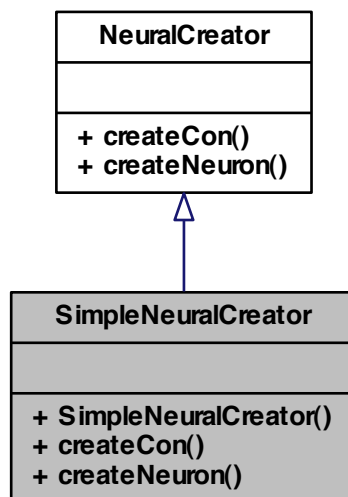
- pkg/AMORE/src/dia/[SimpleContainerIterator.h](#)
- pkg/AMORE/src/[SimpleContainerIterator.cpp](#)

5.20 SimpleNeuralCreator Class Reference

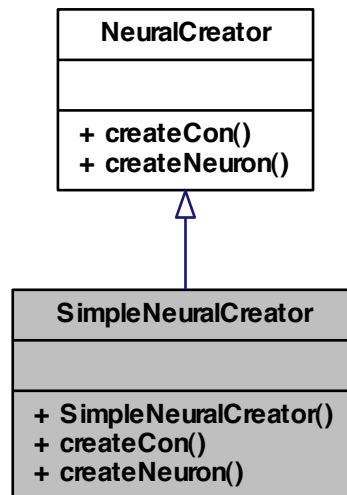
class [SimpleNeuralCreator](#) -

```
#include <SimpleNeuralCreator.h>
```

Inheritance diagram for SimpleNeuralCreator:



Collaboration diagram for SimpleNeuralCreator:



Public Member Functions

- [SimpleNeuralCreator](#) ()
- `Con * createCon` ([NeuralFactory](#) &neuralFactory, [Neuron](#) &neuron)
- `Neuron * createNeuron` ([NeuralFactory](#) &neuralFactory)

5.20.1 Detailed Description

class [SimpleNeuralCreator](#) -

Definition at line 5 of file SimpleNeuralCreator.h.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 SimpleNeuralCreator::SimpleNeuralCreator ()

Definition at line 15 of file SimpleNeuralCreator.cpp.

```
{  
}
```

5.20.3 Member Function Documentation

5.20.3.1 **Con** * SimpleNeuralCreator::createCon (NeuralFactory & *neuralFactory*, Neuron & *neuron*) [virtual]

Implements [NeuralCreator](#).

Definition at line 21 of file SimpleNeuralCreator.cpp.

References [NeuralFactory::makeCon\(\)](#).

```
{  
    return neuralFactory.makeCon(neuron);  
}
```

Here is the call graph for this function:



5.20.3.2 **Neuron** * SimpleNeuralCreator::createNeuron (NeuralFactory & *neuralFactory*) [virtual]

Implements [NeuralCreator](#).

Definition at line 28 of file SimpleNeuralCreator.cpp.

References [NeuralFactory::makeNeuron\(\)](#).

```
{  
    return neuralFactory.makeNeuron();  
}
```

Here is the call graph for this function:



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

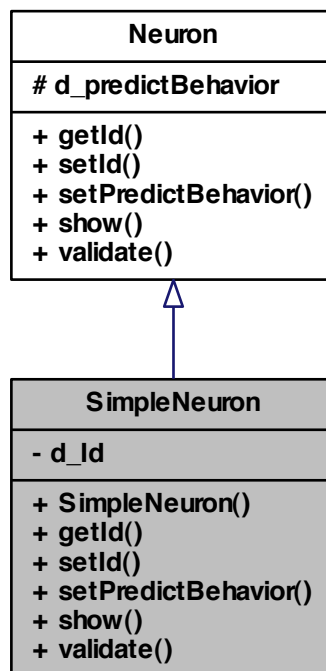
- pkg/AMORE/src/dia/[SimpleNeuralCreator.h](#)
- pkg/AMORE/src/[SimpleNeuralCreator.cpp](#)

5.21 SimpleNeuron Class Reference

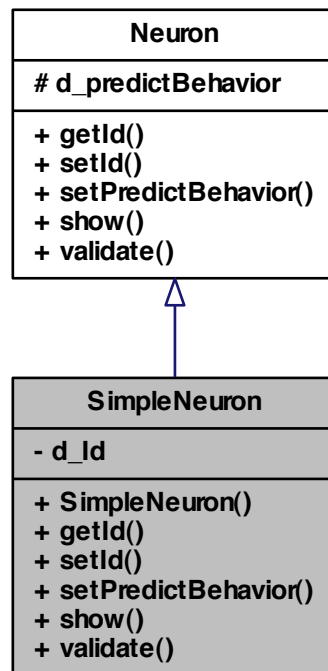
class [SimpleNeuron](#) -

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

Inheritance diagram for SimpleNeuron:



Collaboration diagram for SimpleNeuron:



Public Member Functions

- [SimpleNeuron \(\)](#)
- [Handler getId \(\)](#)
- void [setId \(Handler Id\)](#)
- void [setPredictBehavior \(PredictBehavior *predictBehavior\)](#)
- void [show \(\)](#)
- bool [validate \(\)](#)

Private Attributes

- int [d_Id](#)

5.21.1 Detailed Description

class [SimpleNeuron](#) -

Definition at line 5 of file SimpleNeuron.h.

5.21.2 Constructor & Destructor Documentation

5.21.2.1 SimpleNeuron::SimpleNeuron ()

Definition at line 10 of file SimpleNeuron.cpp.

```
        :  
        d_Id (NA_INTEGER) //, nCons ()  
    {  
    }  
}
```

5.21.3 Member Function Documentation

5.21.3.1 Handler SimpleNeuron::getId () [virtual]

Implements [Neuron](#).

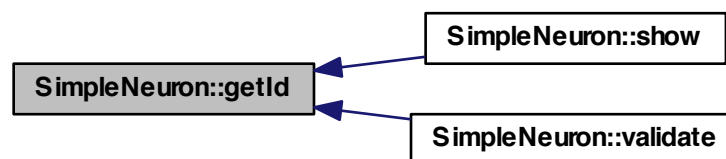
Definition at line 17 of file SimpleNeuron.cpp.

References `d_Id`.

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

```
{  
    return d_Id;  
}
```

Here is the caller graph for this function:



5.21.3.2 void SimpleNeuron::setId (Handler Id) [virtual]

Implements [Neuron](#).

Definition at line 25 of file SimpleNeuron.cpp.

References `d_Id`.

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

5.21.3.3 `void SimpleNeuron::setPredictBehavior (PredictBehavior * predictBehavior)`
[virtual]

Implements [Neuron](#).

Definition at line 33 of file SimpleNeuron.cpp.

References `Neuron::d_predictBehavior`.

```
{
    d_predictBehavior.reset(predictBehavior);
}
```

5.21.3.4 `void SimpleNeuron::show ()` [virtual]

Implements [Neuron](#).

Definition at line 40 of file SimpleNeuron.cpp.

References `Neuron::d_predictBehavior`, and `getId()`.

```
{
    int id = getId();
    Rprintf("\n-----\n");
    if (id == NA_INTEGER)
    {
        Rprintf("\n Id: NA, Invalid neuron Id");
    }
    else
    {
        Rprintf("\n Id: %d", id);
    }
    Rprintf("\n-----\n");
    d_predictBehavior->show();
}
```

Here is the call graph for this function:



5.21.3.5 `bool SimpleNeuron::validate ()` [virtual]

Implements [Neuron](#).

Definition at line 58 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:



5.21.4 Member Data Documentation

5.21.4.1 `int SimpleNeuron::d_Id` [private]

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

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

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

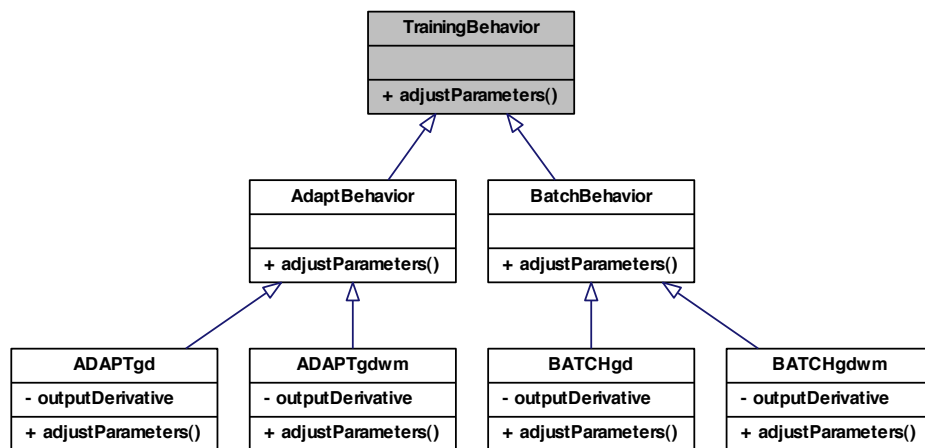
- [pkg/AMORE/src/dia/SimpleNeuron.h](#)
- [pkg/AMORE/src/SimpleNeuron.cpp](#)

5.22 TrainingBehavior Class Reference

class [TrainingBehavior](#) -

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

Inheritance diagram for TrainingBehavior:



Public Member Functions

- void [adjustParameters](#) ()

5.22.1 Detailed Description

class [TrainingBehavior](#) -

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

5.22.2 Member Function Documentation

5.22.2.1 void TrainingBehavior::adjustParameters ()

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

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

- pkg/AMORE/src/dia/[TrainingBehavior.h](#)

Chapter 6

File Documentation

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

```
#include <iostream>
#include <sstream>
#include <algorithm>
#include <vector>
#include <iterator>
#include <boost/shared_ptr.hpp>
#include <boost/weak_ptr.hpp>
#include <boost/foreach.hpp>
#include <boost/ref.hpp>
#include <Rcpp.h>
#include "dia/Con.h"
#include "dia/PredictBehavior.h"
#include "dia/MLPBehavior.h"
#include "dia/Neuron.h"
#include "dia/SimpleNeuron.h"
#include "dia/NeuralFactory.h"
#include "dia/MLPfactory.h"
#include "dia/NeuralCreator.h"
#include "dia/SimpleNeuralCreator.h"
#include "dia/Container.h"
#include "dia/SimpleContainer.h"
```

```

#include "dia/Iterator.h"
#include "dia/SimpleContainerIterator.h"
#include "Con.cpp"
#include "MLPbehavior.cpp"
#include "SimpleNeuron.cpp"
#include "MLPfactory.cpp"
#include "SimpleNeuralCreator.cpp"
#include "Container.cpp"
#include "Iterator.cpp"
#include "SimpleContainer.cpp"
#include "SimpleContainerIterator.cpp"

```

Include dependency graph for AMORE.h:



Defines

- #define [foreach](#) BOOST_FOREACH
- #define [size_type](#) unsigned int

Typedefs

- typedef int [Handler](#)
- typedef boost::reference_wrapper< [PredictBehavior](#) > [PredictBehaviorRef](#)
- typedef boost::reference_wrapper< [TrainingBehavior](#) > [TrainingBehaviorRef](#)
- typedef boost::reference_wrapper< [Neuron](#) > [NeuronRef](#)
- typedef boost::shared_ptr< [PredictBehavior](#) > [PredictBehaviorPtr](#)
- typedef boost::shared_ptr< [Neuron](#) > [NeuronPtr](#)
- typedef boost::shared_ptr< [Con](#) > [ConPtr](#)
- typedef boost::shared_ptr< [Iterator](#)< [NeuronPtr](#) > > [NeuronIteratorPtr](#)
- typedef boost::shared_ptr< [Iterator](#)< [ConPtr](#) > > [ConIteratorPtr](#)
- typedef boost::shared_ptr< [Container](#)< [NeuronPtr](#) > > [NeuronContainerPtr](#)
- typedef boost::shared_ptr< [Container](#)< [ConPtr](#) > > [ConContainerPtr](#)
- typedef boost::shared_ptr< [NeuralFactory](#) > [NeuralFactoryPtr](#)
- typedef boost::shared_ptr< [NeuralCreator](#) > [NeuralCreatorPtr](#)

6.1.1 Define Documentation

6.1.1.1 `#define` foreach BOOST_FOREACH

Definition at line 61 of file AMORE.h.

6.1.1.2 `#define` size_type unsigned int

Definition at line 64 of file AMORE.h.

6.1.2 Typedef Documentation

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

Definition at line 81 of file AMORE.h.

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

Definition at line 78 of file AMORE.h.

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

Definition at line 75 of file AMORE.h.

6.1.2.4 `typedef int Handler`

Definition at line 67 of file AMORE.h.

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

Definition at line 84 of file AMORE.h.

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

Definition at line 83 of file AMORE.h.

6.1.2.7 `typedef boost::shared_ptr< Container<NeuronPtr> > NeuronContainerPtr`

Definition at line 80 of file AMORE.h.

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

Definition at line 77 of file AMORE.h.

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

Definition at line 74 of file AMORE.h.

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

Definition at line 71 of file AMORE.h.

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

Definition at line 73 of file AMORE.h.

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

Definition at line 69 of file AMORE.h.

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

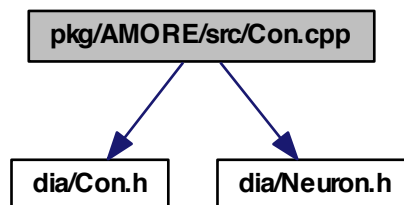
Definition at line 70 of file AMORE.h.

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

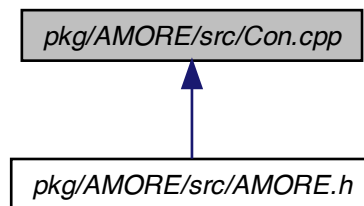
```
#include "dia/Con.h"
```

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

Include dependency graph for Con.cpp:



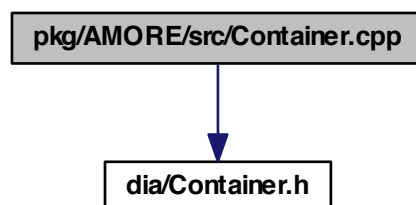
This graph shows which files directly or indirectly include this file:



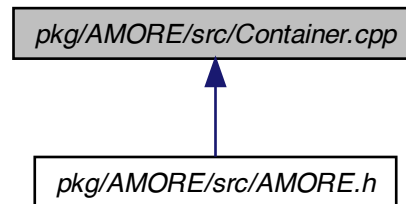
6.3 pkg/AMORE/src/Container.cpp File Reference

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

Include dependency graph for Container.cpp:



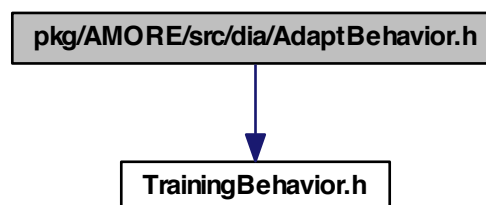
This graph shows which files directly or indirectly include this file:



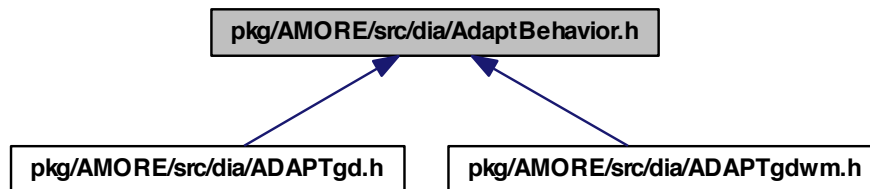
6.4 pkg/AMORE/src/dia/AdaptBehavior.h File Reference

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

Include dependency graph for AdaptBehavior.h:



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



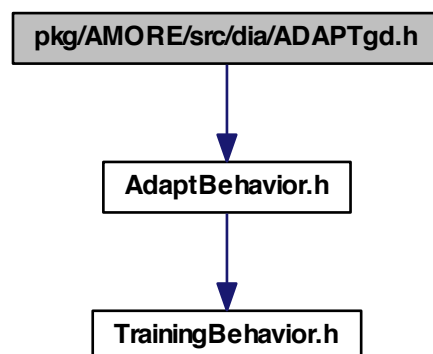
Classes

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

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

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

Include dependency graph for ADAPTgd.h:



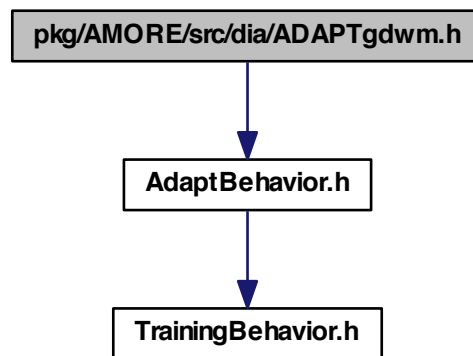
Classes

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

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

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

Include dependency graph for ADAPTgdwm.h:



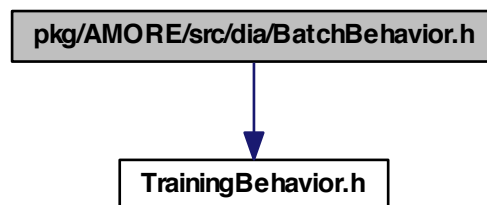
Classes

- class [ADAPTgdwm](#)
class [ADAPTgdwm](#) -

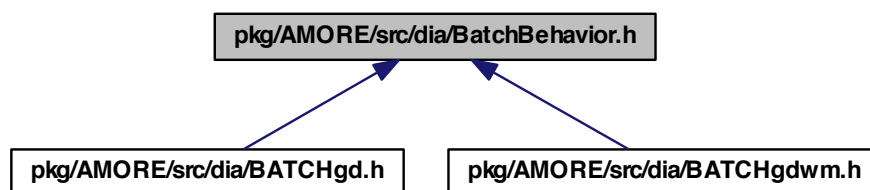
6.7 pkg/AMORE/src/dia/BatchBehavior.h File Reference

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

Include dependency graph for BatchBehavior.h:



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



Classes

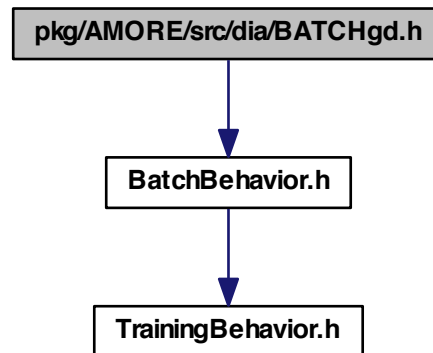
- class [BatchBehavior](#)

class [BatchBehavior](#) -

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

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

Include dependency graph for BATCHgd.h:



Classes

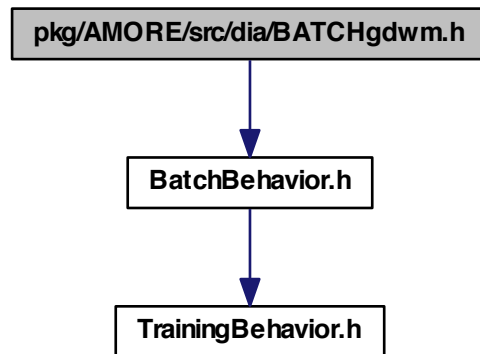
- class [BATCHgd](#)

class [BATCHgd](#) -

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

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


Include dependency graph for BATCHgdwm.h:



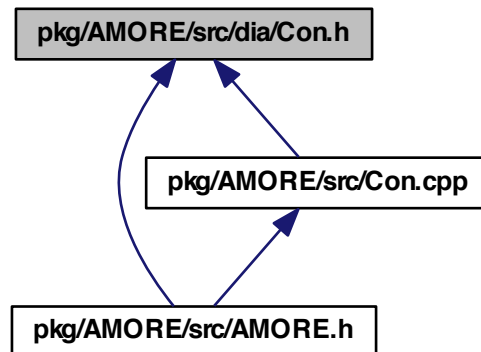
Classes

- class [BATCHgdwm](#)

class [BATCHgdwm](#) -

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

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



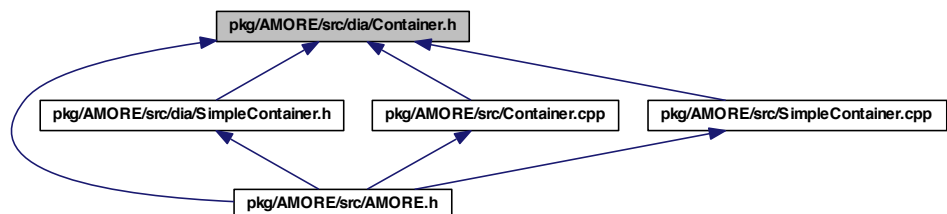
Classes

- class [Con](#)

class [Con](#) -

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

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



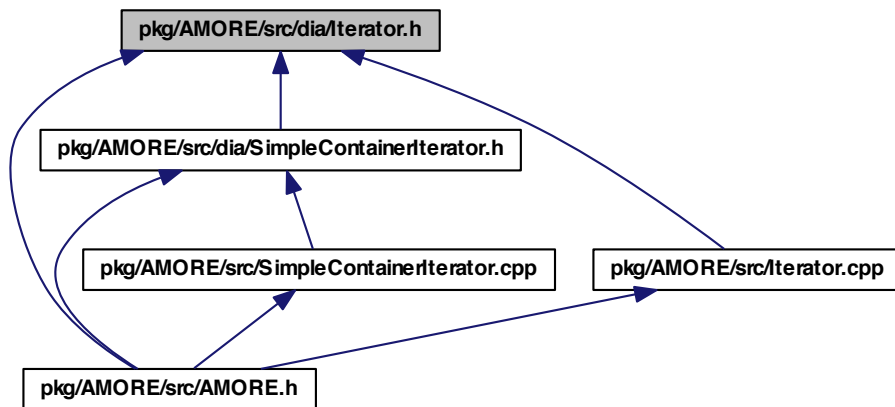
Classes

- class [Container< T >](#)

class [Container](#) -

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

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



Classes

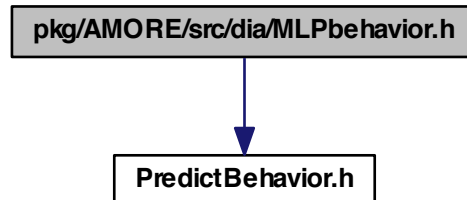
- class [Iterator< T >](#)

class [Iterator](#) -

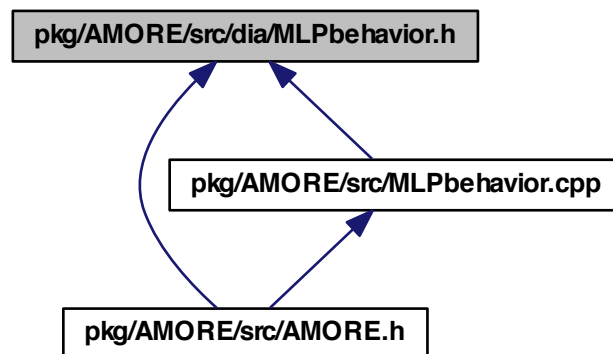
6.13 pkg/AMORE/src/dia/MLPbehavior.h File Reference

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

Include dependency graph for MLPbehavior.h:



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



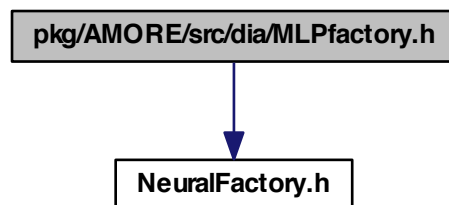
Classes

- class `MLPbehavior`
 class `MLPbehavior` -

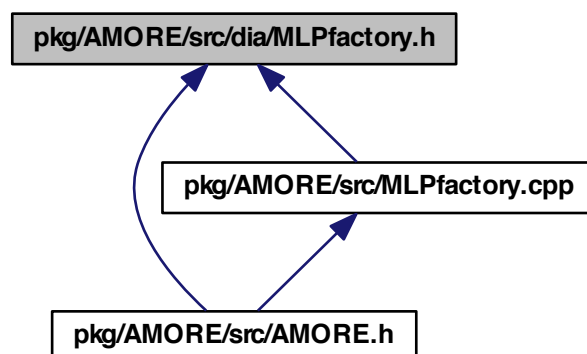
6.14 pkg/AMORE/src/dia/MLPfactory.h File Reference

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

Include dependency graph for MLPfactory.h:



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

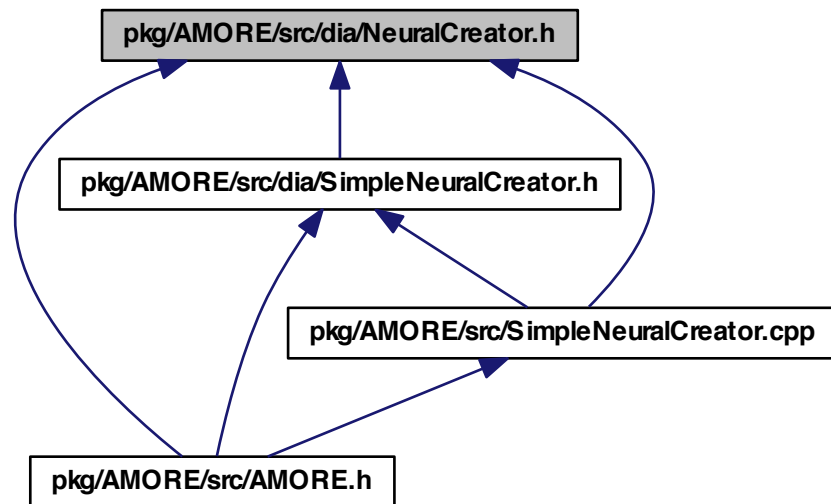


Classes

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

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

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



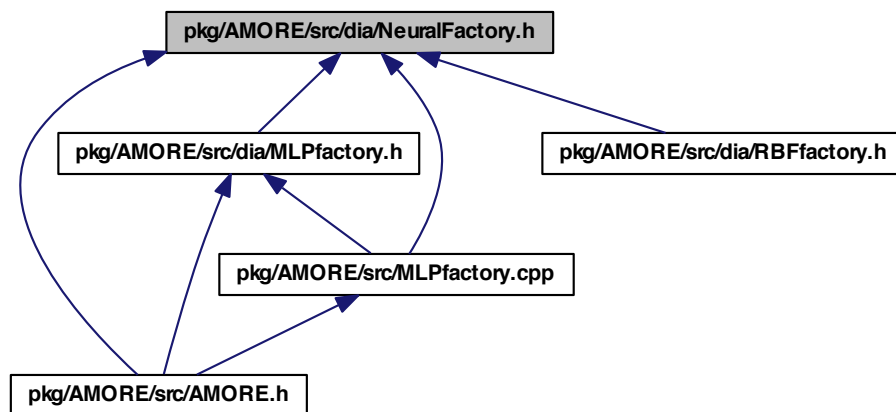
Classes

- class [NeuralCreator](#)

class [NeuralCreator](#) -

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

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



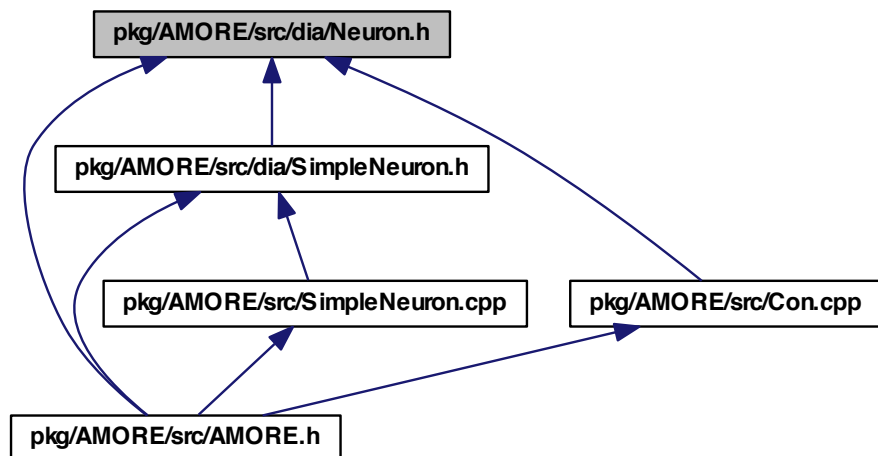
Classes

- class `NeuralFactory`

class `NeuralFactory` -

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

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



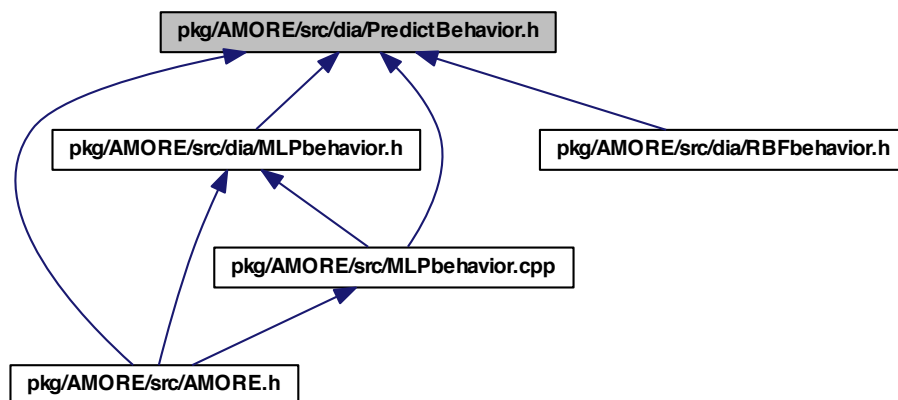
Classes

- class `Neuron`

class `Neuron` -

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

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



Classes

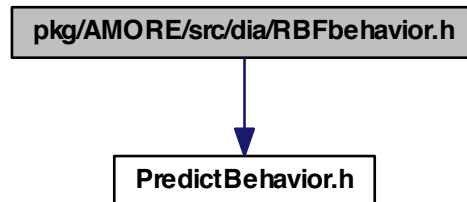
- class [PredictBehavior](#)

class [PredictBehavior](#) -

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

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

Include dependency graph for RBFbehavior.h:



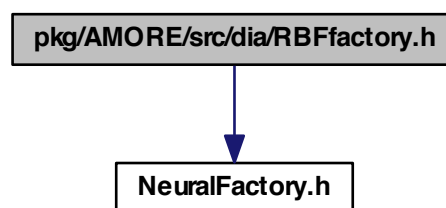
Classes

- class [RBFbehavior](#)
class [RBFbehavior](#) -

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

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

Include dependency graph for RBFfactory.h:



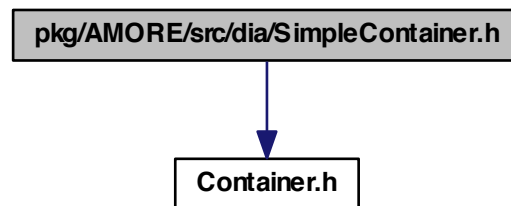
Classes

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

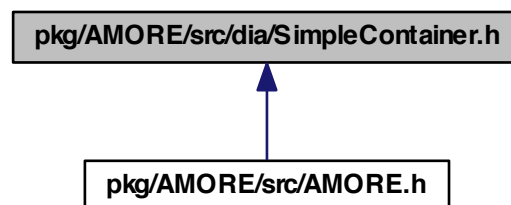
6.21 pkg/AMORE/src/dia/SimpleContainer.h File Reference

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

Include dependency graph for SimpleContainer.h:



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



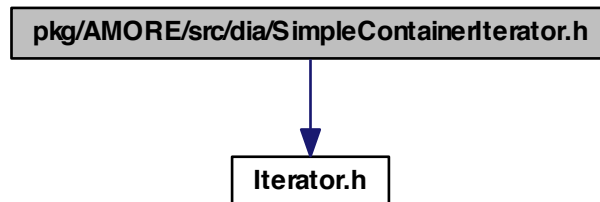
Classes

- class `SimpleContainer< T >`
 class `SimpleContainer` -

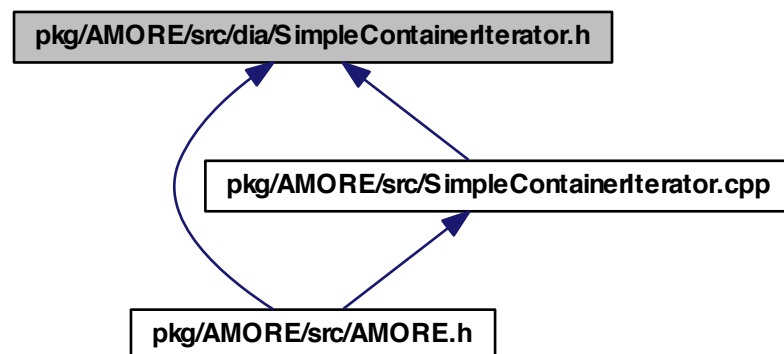
6.22 pkg/AMORE/src/dia/SimpleContainerIterator.h File Reference

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

Include dependency graph for SimpleContainerIterator.h:



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



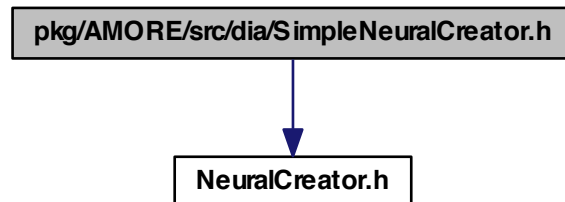
Classes

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

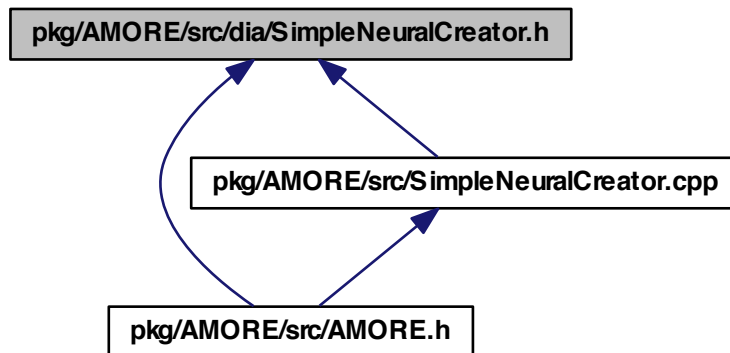
6.23 pkg/AMORE/src/dia/SimpleNeuralCreator.h File Reference

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

Include dependency graph for SimpleNeuralCreator.h:



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



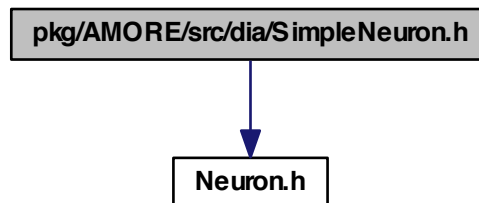
Classes

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

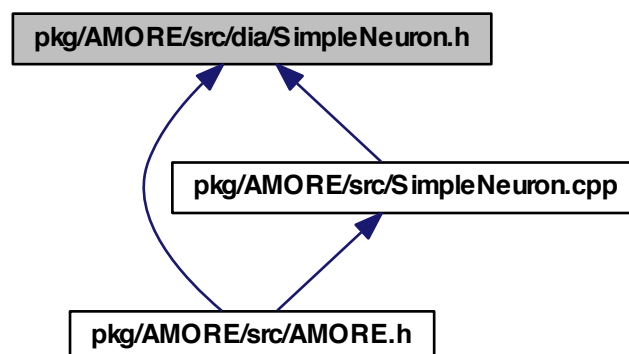
6.24 pkg/AMORE/src/dia/SimpleNeuron.h File Reference

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

Include dependency graph for SimpleNeuron.h:



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

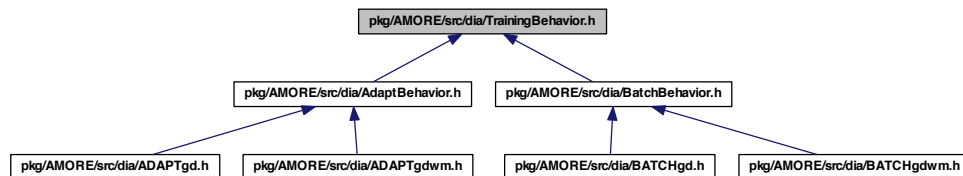


Classes

- class [SimpleNeuron](#)
class SimpleNeuron -

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

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



Classes

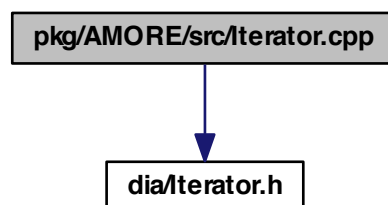
- class [TrainingBehavior](#)

class [TrainingBehavior](#) -

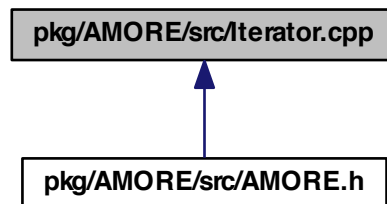
6.26 pkg/AMORE/src/Iterator.cpp File Reference

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

Include dependency graph for `Iterator.cpp`:



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

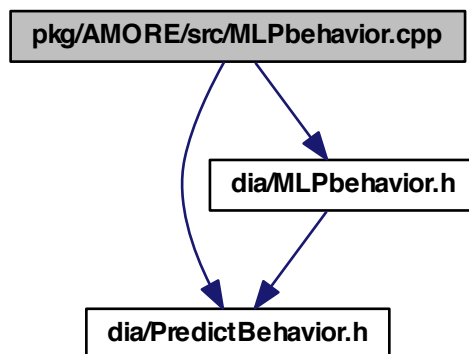


6.27 pkg/AMORE/src/MLPbehavior.cpp File Reference

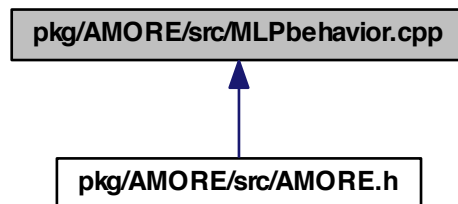
```
#include "dia/PredictBehavior.h"
```

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

Include dependency graph for MLPbehavior.cpp:



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

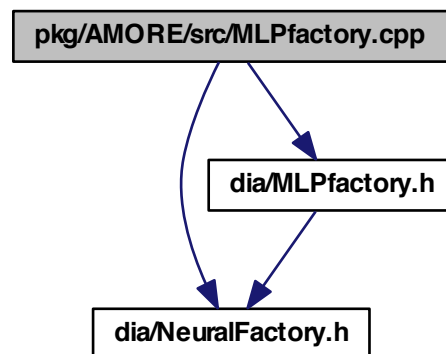


6.28 pkg/AMORE/src/MLPfactory.cpp File Reference

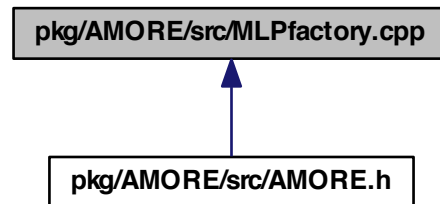
```
#include "dia/NeuralFactory.h"
```

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

Include dependency graph for MLPfactory.cpp:



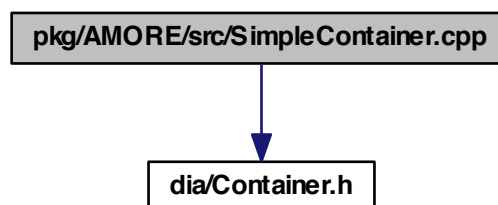
This graph shows which files directly or indirectly include this file:



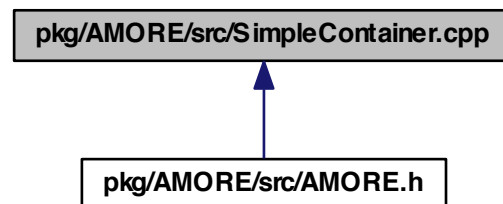
6.29 pkg/AMORE/src/SimpleContainer.cpp File Reference

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

Include dependency graph for SimpleContainer.cpp:



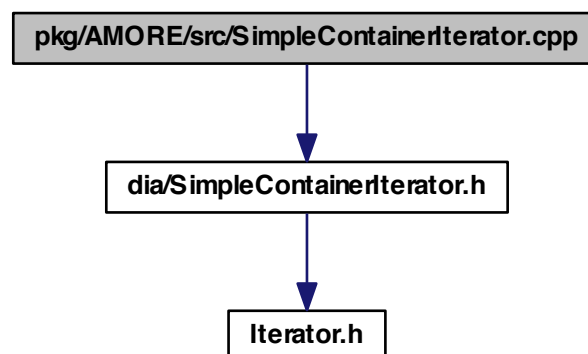
This graph shows which files directly or indirectly include this file:



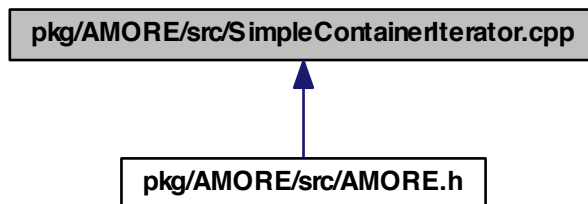
6.30 pkg/AMORE/src/SimpleContainerIterator.cpp File Reference

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

Include dependency graph for SimpleContainerIterator.cpp:



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

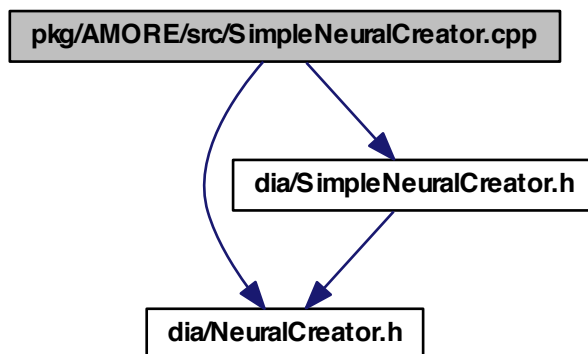


6.31 pkg/AMORE/src/SimpleNeuralCreator.cpp File Reference

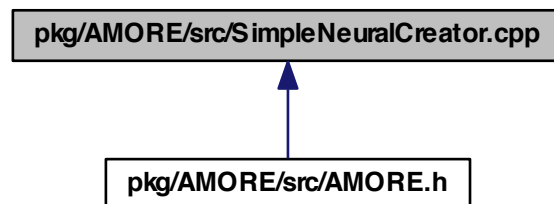
```
#include "dia/NeuralCreator.h"
```

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

Include dependency graph for SimpleNeuralCreator.cpp:



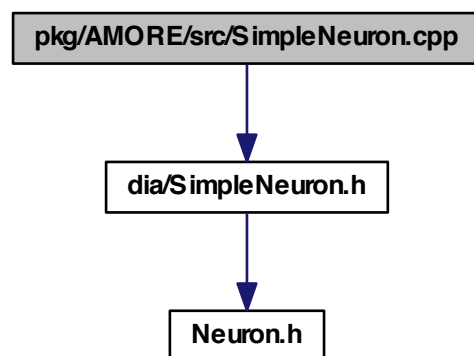
This graph shows which files directly or indirectly include this file:



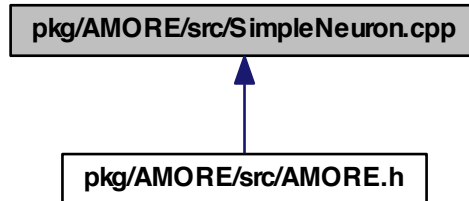
6.32 pkg/AMORE/src/SimpleNeuron.cpp File Reference

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

Include dependency graph for SimpleNeuron.cpp:



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



Index

- ~Container
 - Container, [33](#)
- ~Iterator
 - Iterator, [36](#)
- ~SimpleContainer
 - SimpleContainer, [63](#)
- ~SimpleContainerIterator
 - SimpleContainerIterator, [71](#)
- AdaptBehavior, [9](#)
 - adjustParameters, [11](#)
- ADAPTgd, [12](#)
 - adjustParameters, [13](#)
 - outputDerivative, [14](#)
- ADAPTgdwm, [14](#)
 - adjustParameters, [16](#)
 - outputDerivative, [17](#)
- adjustParameters
 - AdaptBehavior, [11](#)
 - ADAPTgd, [13](#)
 - ADAPTgdwm, [16](#)
 - BatchBehavior, [19](#)
 - BATCHgd, [21](#)
 - BATCHgdwm, [24](#)
 - TrainingBehavior, [81](#)
- AMORE.h
 - ConContainerPtr, [85](#)
 - ConIteratorPtr, [85](#)
 - ConPtr, [85](#)
 - foreach, [85](#)
 - Handler, [85](#)
 - NeuralCreatorPtr, [85](#)
 - NeuralFactoryPtr, [85](#)
 - NeuronContainerPtr, [85](#)
 - NeuronIteratorPtr, [85](#)
 - NeuronPtr, [85](#)
 - NeuronRef, [86](#)
 - PredictBehaviorPtr, [86](#)
 - PredictBehaviorRef, [86](#)
 - size_type, [85](#)
 - TrainingBehaviorRef, [86](#)
- at
 - Container, [33](#)
 - SimpleContainer, [64](#)
- BatchBehavior, [17](#)
 - adjustParameters, [19](#)
- BATCHgd, [20](#)
 - adjustParameters, [21](#)
 - outputDerivative, [22](#)
- BATCHgdwm, [22](#)
 - adjustParameters, [24](#)
 - outputDerivative, [25](#)
- clear
 - Container, [33](#)
 - SimpleContainer, [65](#)
- Con, [25](#)
 - Con, [26](#)
 - d_neuron, [31](#)
 - d_weight, [31](#)
 - getNeuron, [26](#)
 - getWeight, [27](#)
 - Id, [28](#)
 - setNeuron, [29](#)
 - setWeight, [29](#)
 - show, [29](#)
 - validate, [30](#)
- ConContainerPtr
 - AMORE.h, [85](#)
- ConIteratorPtr
 - AMORE.h, [85](#)
- ConPtr
 - AMORE.h, [85](#)
- Container, [31](#)
 - ~Container, [33](#)
 - at, [33](#)
 - clear, [33](#)
 - Container, [33](#)
 - createIterator, [33](#)
 - empty, [34](#)
 - push_back, [34](#)

- reserve, [34](#)
 - show, [34](#)
 - size, [34](#)
 - validate, [34](#)
- createCon
 - NeuralCreator, [46](#)
 - SimpleNeuralCreator, [75](#)
- createIterator
 - Container, [33](#)
 - SimpleContainer, [65](#)
- createNeuron
 - NeuralCreator, [47](#)
 - SimpleNeuralCreator, [75](#)
- currentItem
 - Iterator, [36](#)
 - SimpleContainerIterator, [71](#)
- d_accumulator
 - MLPbehavior, [39](#)
 - RBFbehavior, [57](#)
- d_altitude
 - RBFbehavior, [57](#)
- d_bias
 - MLPbehavior, [40](#)
- d_collection
 - SimpleContainer, [68](#)
- d_container
 - SimpleContainerIterator, [73](#)
- d_current
 - SimpleContainerIterator, [73](#)
- d_id
 - SimpleNeuron, [80](#)
- d_nCons
 - MLPbehavior, [40](#)
 - RBFbehavior, [57](#)
- d_neuron
 - Con, [31](#)
- d_output
 - MLPbehavior, [40](#)
 - RBFbehavior, [57](#)
- d_predictBehavior
 - Neuron, [52](#)
- d_weight
 - Con, [31](#)
- d_width
 - RBFbehavior, [57](#)
- empty
 - Container, [34](#)
 - SimpleContainer, [65](#)
- first
 - Iterator, [36](#)
 - SimpleContainerIterator, [72](#)
- foreach
 - AMORE.h, [85](#)
- getId
 - Neuron, [52](#)
 - SimpleNeuron, [78](#)
- getNeuron
 - Con, [26](#)
- getWeight
 - Con, [27](#)
- Handler
 - AMORE.h, [85](#)
- Id
 - Con, [28](#)
- isDone
 - Iterator, [36](#)
 - SimpleContainerIterator, [72](#)
- Iterator, [34](#)
 - ~Iterator, [36](#)
 - currentItem, [36](#)
 - first, [36](#)
 - isDone, [36](#)
 - Iterator, [36](#)
 - next, [36](#)
- makeCon
 - MLPfactory, [43](#)
 - NeuralFactory, [49](#)
 - RBFfactory, [60](#)
- makeConContainer
 - MLPfactory, [43](#)
 - NeuralFactory, [49](#)
 - RBFfactory, [60](#)
- makeNeuron
 - MLPfactory, [44](#)
 - NeuralFactory, [49](#)
 - RBFfactory, [60](#)
- makeNeuronContainer
 - MLPfactory, [44](#)
 - NeuralFactory, [49](#)
 - RBFfactory, [60](#)
- makePredictBehavior
 - MLPfactory, [44](#)
 - NeuralFactory, [50](#)
 - RBFfactory, [60](#)

- MLPbehavior, 37
 - d_accumulator, 39
 - d_bias, 40
 - d_nCons, 40
 - d_output, 40
 - MLPfactory, 39
 - predict, 39
 - show, 39
- MLPfactory, 40
 - makeCon, 43
 - makeConContainer, 43
 - makeNeuron, 44
 - makeNeuronContainer, 44
 - makePredictBehavior, 44
 - MLPbehavior, 39
 - MLPfactory, 43
- NeuralCreator, 46
 - createCon, 46
 - createNeuron, 47
- NeuralCreatorPtr
 - AMORE.h, 85
- NeuralFactory, 47
 - makeCon, 49
 - makeConContainer, 49
 - makeNeuron, 49
 - makeNeuronContainer, 49
 - makePredictBehavior, 50
- NeuralFactoryPtr
 - AMORE.h, 85
- Neuron, 50
 - d_predictBehavior, 52
 - getId, 52
 - setId, 52
 - setPredictBehavior, 52
 - show, 52
 - validate, 52
- NeuronContainerPtr
 - AMORE.h, 85
- NeuronIteratorPtr
 - AMORE.h, 85
- NeuronPtr
 - AMORE.h, 85
- NeuronRef
 - AMORE.h, 86
- next
 - Iterator, 36
 - SimpleContainerIterator, 72
- outputDerivative
- ADAPTgd, 14
- ADAPTgdwm, 17
- BATCHgd, 22
- BATCHgdwm, 25
- pkg/AMORE/src/AMORE.h, 83
- pkg/AMORE/src/Con.cpp, 86
- pkg/AMORE/src/Container.cpp, 87
- pkg/AMORE/src/dia/AdaptBehavior.h, 88
- pkg/AMORE/src/dia/ADAPTgd.h, 89
- pkg/AMORE/src/dia/ADAPTgdwm.h, 90
- pkg/AMORE/src/dia/BatchBehavior.h, 90
- pkg/AMORE/src/dia/BATCHgd.h, 91
- pkg/AMORE/src/dia/BATCHgdwm.h, 92
- pkg/AMORE/src/dia/Con.h, 94
- pkg/AMORE/src/dia/Container.h, 94
- pkg/AMORE/src/dia/Iterator.h, 95
- pkg/AMORE/src/dia/MLPbehavior.h, 95
- pkg/AMORE/src/dia/MLPfactory.h, 96
- pkg/AMORE/src/dia/NeuralCreator.h, 98
- pkg/AMORE/src/dia/NeuralFactory.h, 99
- pkg/AMORE/src/dia/Neuron.h, 100
- pkg/AMORE/src/dia/PredictBehavior.h, 101
- pkg/AMORE/src/dia/RBFbehavior.h, 101
- pkg/AMORE/src/dia/RBFfactory.h, 102
- pkg/AMORE/src/dia/SimpleContainer.h, 103
- pkg/AMORE/src/dia/SimpleContainerIterator.h, 103
- pkg/AMORE/src/dia/SimpleNeuralCreator.h, 104
- pkg/AMORE/src/dia/SimpleNeuron.h, 105
- pkg/AMORE/src/dia/TrainingBehavior.h, 107
- pkg/AMORE/src/Iterator.cpp, 107
- pkg/AMORE/src/MLPbehavior.cpp, 108
- pkg/AMORE/src/MLPfactory.cpp, 109
- pkg/AMORE/src/SimpleContainer.cpp, 110
- pkg/AMORE/src/SimpleContainerIterator.cpp, 111
- pkg/AMORE/src/SimpleNeuralCreator.cpp, 112
- pkg/AMORE/src/SimpleNeuron.cpp, 113
- predict
 - MLPbehavior, 39
 - PredictBehavior, 54
 - RBFbehavior, 57
- PredictBehavior, 53
 - predict, 54
 - show, 54
- PredictBehaviorPtr
 - AMORE.h, 86

- PredictBehaviorRef
 - AMORE.h, 86
- push_back
 - Container, 34
 - SimpleContainer, 65
- RBFbehavior, 54
 - d_accumulator, 57
 - d_altitude, 57
 - d_nCons, 57
 - d_output, 57
 - d_width, 57
 - predict, 57
 - show, 57
- RBFfactory, 57
 - makeCon, 60
 - makeConContainer, 60
 - makeNeuron, 60
 - makeNeuronContainer, 60
 - makePredictBehavior, 60
 - RBFfactory, 60
- reserve
 - Container, 34
 - SimpleContainer, 66
- setId
 - Neuron, 52
 - SimpleNeuron, 78
- setNeuron
 - Con, 29
- setPredictBehavior
 - Neuron, 52
 - SimpleNeuron, 79
- setWeight
 - Con, 29
- show
 - Con, 29
 - Container, 34
 - MLPbehavior, 39
 - Neuron, 52
 - PredictBehavior, 54
 - RBFbehavior, 57
 - SimpleContainer, 66
 - SimpleNeuron, 79
- SimpleContainer, 60
 - ~SimpleContainer, 63
 - at, 64
 - clear, 65
 - createIterator, 65
 - d_collection, 68
 - empty, 65
 - push_back, 65
 - reserve, 66
 - show, 66
 - SimpleContainer, 63
 - SimpleContainerIterator< T >, 68
 - size, 67
 - validate, 67
- SimpleContainer< T >
 - SimpleContainerIterator, 72
- SimpleContainerIterator, 68
 - ~SimpleContainerIterator, 71
 - currentItem, 71
 - d_container, 73
 - d_current, 73
 - first, 72
 - isDone, 72
 - next, 72
 - SimpleContainer< T >, 72
 - SimpleContainerIterator, 71
- SimpleContainerIterator< T >
 - SimpleContainer, 68
- SimpleNeuralCreator, 73
 - createCon, 75
 - createNeuron, 75
 - SimpleNeuralCreator, 74
- SimpleNeuron, 76
 - d_Id, 80
 - getId, 78
 - setId, 78
 - setPredictBehavior, 79
 - show, 79
 - SimpleNeuron, 78
 - validate, 80
- size
 - Container, 34
 - SimpleContainer, 67
- size_type
 - AMORE.h, 85
- TrainingBehavior, 81
 - adjustParameters, 81
- TrainingBehaviorRef
 - AMORE.h, 86
- validate
 - Con, 30
 - Container, 34
 - Neuron, 52
 - SimpleContainer, 67

SimpleNeuron, [80](#)