# AMORE++

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

Generated by Doxygen 1.7.4

Thu Jul 28 2011 01:19:52

# Contents

# 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:

# Chapter 3

# Class Index

## 3.1 Class List

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

# Chapter 4

# File Index

## 4.1 File List

Here is a list of all files with brief descriptions:

# Chapter 5

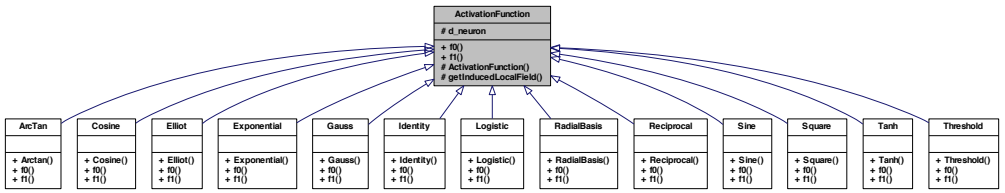
# Class Documentation

## 5.1 ActivationFunction Class Reference

class ActivationFunction -

`#include <ActivationFunction.h>`

Inheritance diagram for ActivationFunction:




### Public Member Functions

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

### Protected Member Functions

- ActivationFunction (NeuronPtr neuronPtr)
- double getInducedLocalField ()

### Protected Attributes

- NeuronWeakPtr d_neuron

### 5.1.1 Detailed Description

class ActivationFunction -

Definition at line 4 of file ActivationFunction.h.

### 5.1.2 Constructor & Destructor Documentation

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

Definition at line 12 of file ActivationFunction.cpp.

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

### 5.1.3 Member Function Documentation

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

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

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

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

#### 5.1.3.3 double ActivationFunction::getInducedLocalField ( ) [protected]

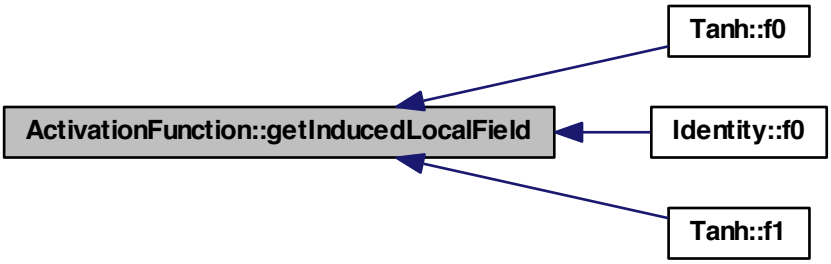
Definition at line 18 of file ActivationFunction.cpp.

References d_neuron.

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

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

Here is the caller graph for this function:




### 5.1.4 Member Data Documentation

#### 5.1.4.1 NeuronWeakPtr ActivationFunction::d_neuron [protected]

Definition at line 7 of file ActivationFunction.h.

Referenced by getInducedLocalField().

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

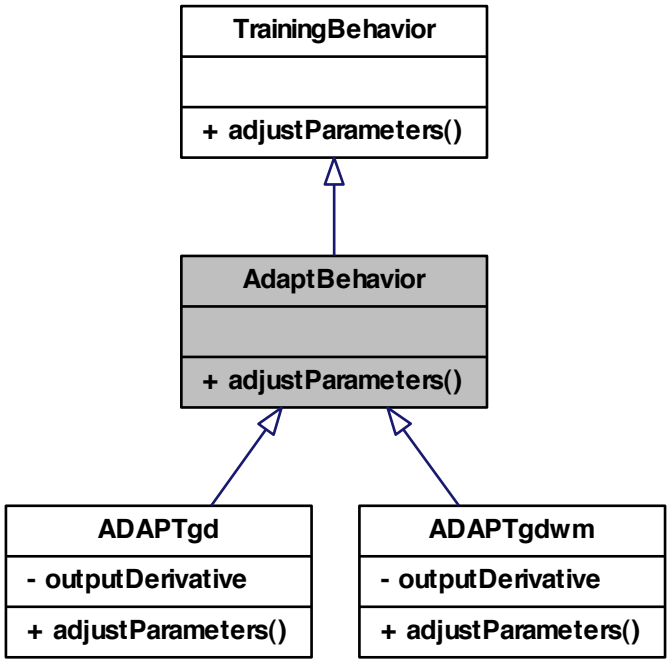
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Activation
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/ActivationFunction.cpp

## 5.2 AdaptBehavior Class Reference

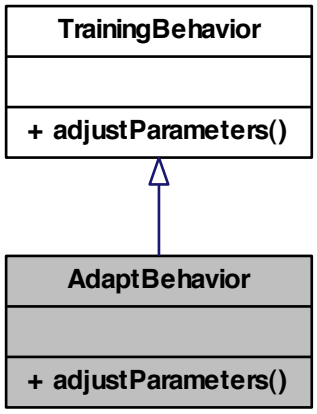
class AdaptBehavior -

`#include <AdaptBehavior.h>`

Inheritance diagram for AdaptBehavior:



Collaboration diagram for AdaptBehavior:




### Public Member Functions

- virtual void adjustParameters ()=0

### 5.2.1 Detailed Description

class AdaptBehavior -

Definition at line 5 of file AdaptBehavior.h.

### 5.2.2 Member Function Documentation

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

Reimplemented from TrainingBehavior.

Implemented in ADAPTgd, and ADAPTgdwm.

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

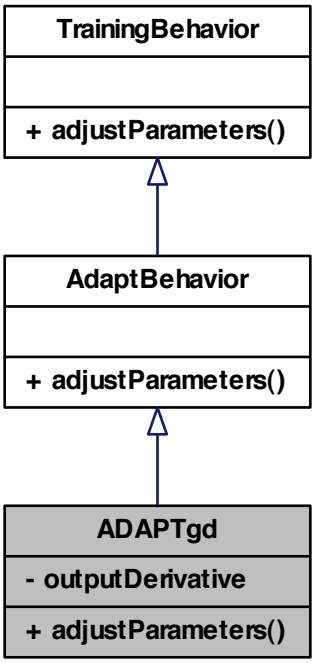
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptBeh

## 5.3 ADAPTgd Class Reference

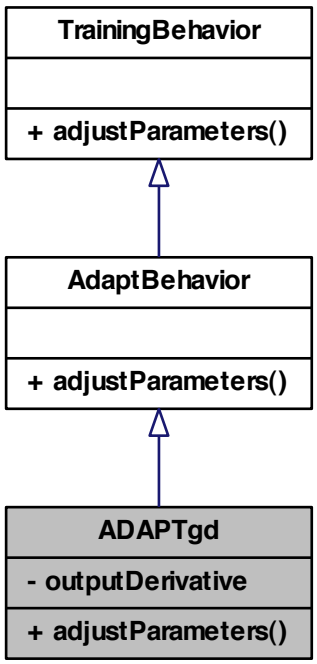
class ADAPTgd -

`#include <ADAPTgd.h>`

Inheritance diagram for ADAPTgd:



Collaboration diagram for ADAPTgd:




### Public Member Functions

- void adjustParameters ()

### Private Attributes

- double outputDerivative

### 5.3.1 Detailed Description

class ADAPTgd -

Definition at line 5 of file ADAPTgd.h.

### 5.3.2 Member Function Documentation

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

Implements AdaptBehavior.

### 5.3.3 Member Data Documentation

#### 5.3.3.1 double ADAPTgd::outputDerivative [private]

Definition at line 8 of file ADAPTgd.h.

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

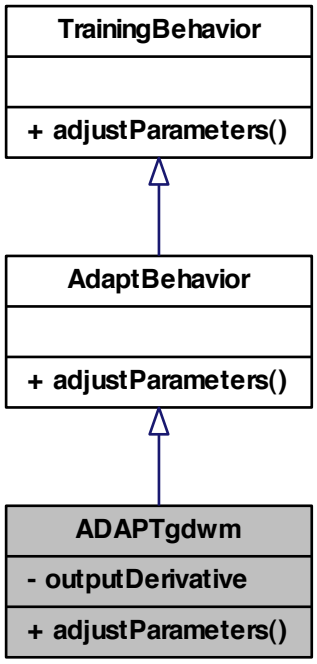
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade

## 5.4 ADAPTgdwm Class Reference

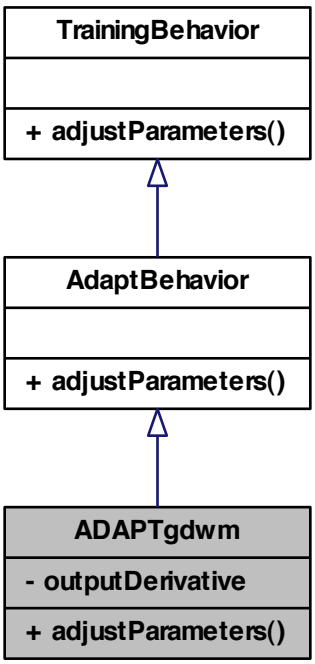
class ADAPTgdwm -

`#include <ADAPTgdwm.h>`

Inheritance diagram for ADAPTgdwm:



Collaboration diagram for ADAPTgdwm:




### Public Member Functions

- void adjustParameters ()

### Private Attributes

- double outputDerivative

### 5.4.1 Detailed Description

class ADAPTgdwm -

Definition at line 5 of file ADAPTgdwm.h.

### 5.4.2 Member Function Documentation

#### 5.4.2.1 void ADAPTgdwm::adjustParameters ( ) [virtual]

Implements AdaptBehavior.

### 5.4.3 Member Data Documentation

#### 5.4.3.1 double ADAPTgdwm::outputDerivative [private]

Definition at line 8 of file ADAPTgdwm.h.

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

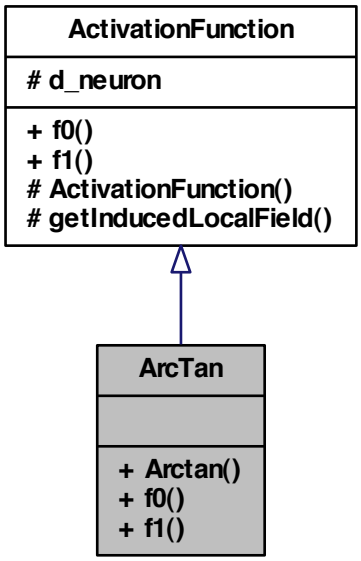
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ADAPTgd

## 5.5 ArcTan Class Reference

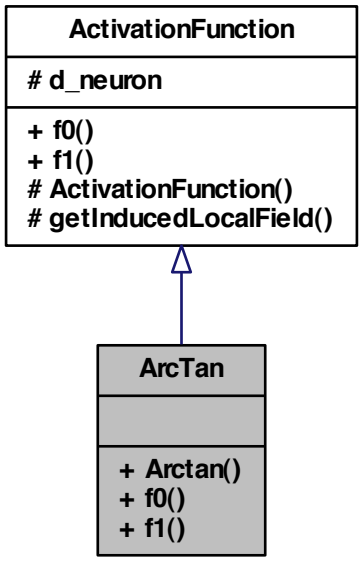
class ArcTan -

`#include <ArcTan.h>`

Inheritance diagram for ArcTan:



Collaboration diagram for ArcTan:




### Public Member Functions

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

### 5.5.1 Detailed Description

class ArcTan -

Definition at line 5 of file ArcTan.h.

### 5.5.2 Member Function Documentation

#### 5.5.2.1 ArcTan::Arctan ( NeuronPtr *neuronPtr* )

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

Implements ActivationFunction.

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

Implements ActivationFunction.

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

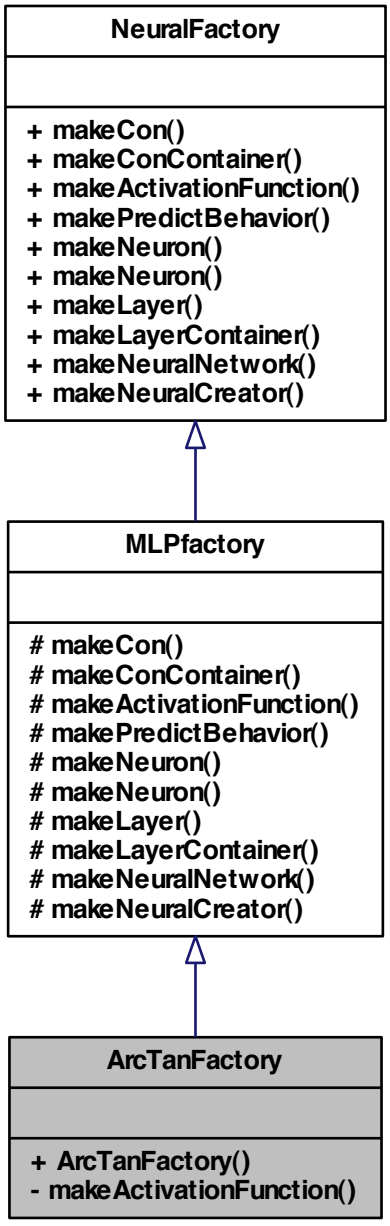
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ArcTan.h

## 5.6 ArcTanFactory Class Reference

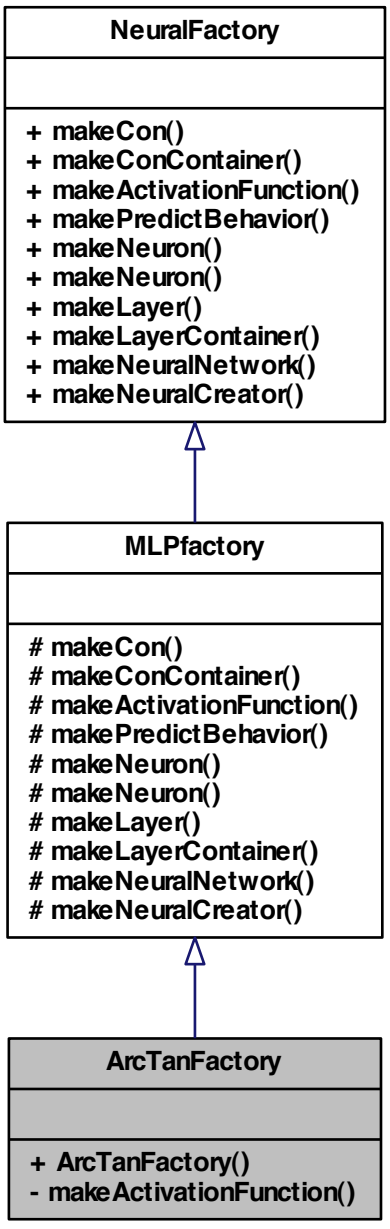
class ArcTanFactory -

`#include <ArcTanFactory.h>`

Inheritance diagram for ArcTanFactory:



Collaboration diagram for ArcTanFactory:



### Public Member Functions

- ArcTanFactory ()

### Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

### 5.6.1 Detailed Description

class ArcTanFactory -

Definition at line 5 of file ArcTanFactory.h.

### 5.6.2 Constructor & Destructor Documentation

#### 5.6.2.1 ArcTanFactory::ArcTanFactory ( )

### 5.6.3 Member Function Documentation

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

Implements MLPfactory.

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

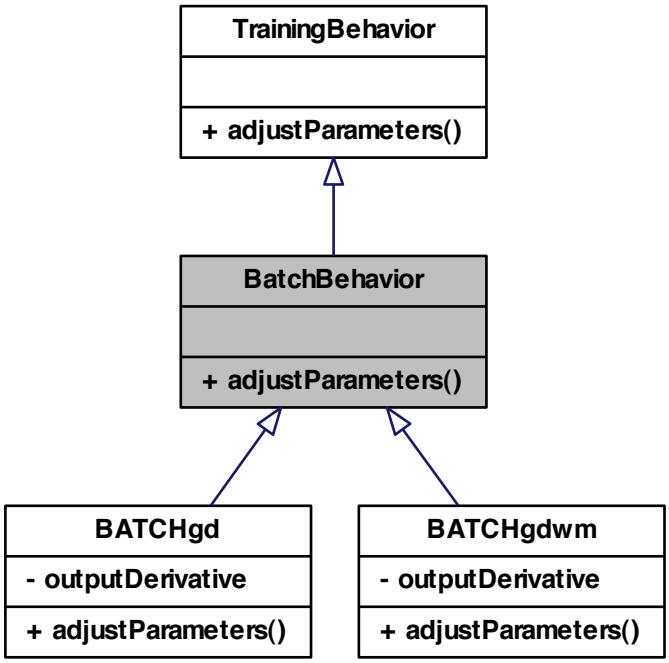
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade

## 5.7 BatchBehavior Class Reference

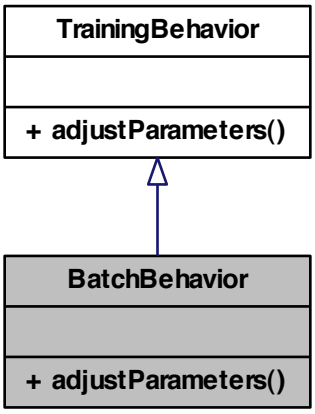
class BatchBehavior -

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

Inheritance diagram for BatchBehavior:



Collaboration diagram for BatchBehavior:




### Public Member Functions

- virtual void adjustParameters ()=0

## 5.7.1 Detailed Description

class BatchBehavior -

Definition at line 5 of file BatchBehavior.h.

## 5.7.2 Member Function Documentation

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

Reimplemented from TrainingBehavior.

Implemented in BATCHgd, and BATCHgdwm.

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

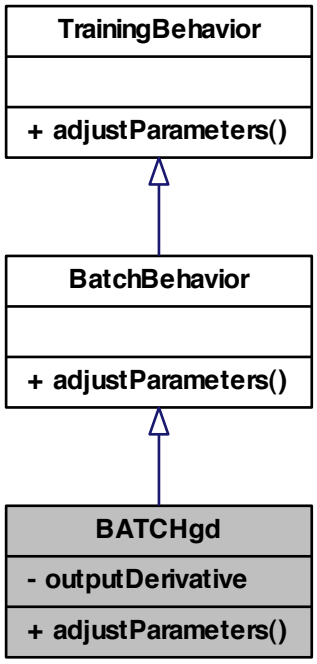
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade

## 5.8 BATCHgd Class Reference

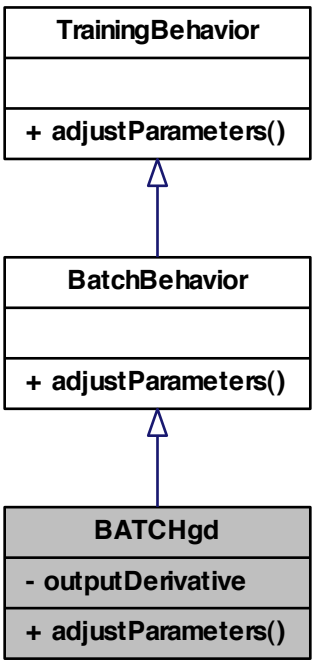
class BATCHgd -

`#include <BATCHgd.h>`

Inheritance diagram for BATCHgd:



Collaboration diagram for BATCHgd:




### Public Member Functions

- void adjustParameters ()

### Private Attributes

- double outputDerivative

### 5.8.1 Detailed Description

class BATCHgd -

Definition at line 5 of file BATCHgd.h.

### 5.8.2 Member Function Documentation

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

Implements BatchBehavior.

### 5.8.3 Member Data Documentation

#### 5.8.3.1 double BATCHgd::outputDerivative [private]

Definition at line 8 of file BATCHgd.h.

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

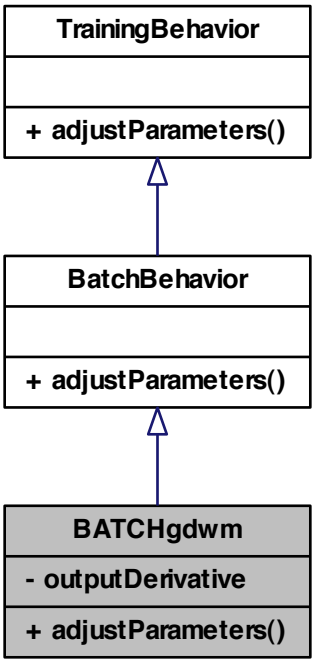
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BATCHgd

## 5.9 BATCHgdwm Class Reference

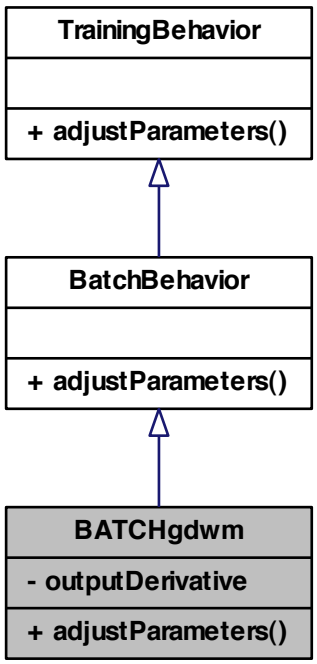
class BATCHgdwm -

`#include <BATCHgdwm.h>`

Inheritance diagram for BATCHgdwm:



Collaboration diagram for BATCHgdwm:




### Public Member Functions

- void adjustParameters ()

### Private Attributes

- double outputDerivative

### 5.9.1 Detailed Description

class BATCHgdwm -

Definition at line 5 of file BATCHgdwm.h.

### 5.9.2 Member Function Documentation

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

Implements BatchBehavior.

### 5.9.3 Member Data Documentation

#### 5.9.3.1 double BATCHgdwm::outputDerivative [private]

Definition at line 8 of file BATCHgdwm.h.

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

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

## 5.10 Con Class Reference

class Con -

`#include <Connection.h>`

### Public Member Functions

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

### Private Attributes

- NeuronRef d_neuron
- double d_weight

### 5.10.1 Detailed Description

class Con -

Definition at line 3 of file Connection.h.

### 5.10.2 Constructor & Destructor Documentation

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

Constructor.

Definition at line 20 of file Connection.cpp.

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

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

Constructor.

Definition at line 31 of file Connection.cpp.

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

### 5.10.3 Member Function Documentation

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

from field accessor.

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

**Returns**

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

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

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

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

**See also**

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

Definition at line 57 of file Connection.cpp.

References d_neuron.

```
{
  return d_neuron;
}
```

#### 5.10.3.2 double Con::getWeight ( )

weight field accessor.

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

**Returns**

The value of weight (double)

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

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

**See also**

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

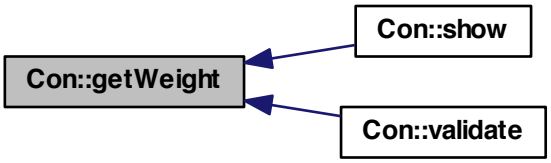
Definition at line 117 of file Connection.cpp.

References d_weight.

Referenced by show(), and validate().

```
{
  return d_weight;
}
```

Here is the caller graph for this function:




#### 5.10.3.3 int Con::Id ( )

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

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

**Returns**

The value of the Id (an integer).

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

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

**See also**

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

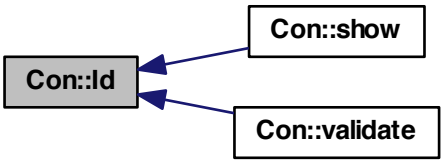
Definition at line 89 of file Connection.cpp.

References d_neuron.

Referenced by show(), and validate().

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

Here is the caller graph for this function:




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

Definition at line 64 of file Connection.cpp.

References d_neuron.

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

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

Definition at line 124 of file Connection.cpp.

References d_weight.

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

#### 5.10.3.6 void Con::show ( )

Pretty print of the Con information.

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

**Returns**

true in case everything works without throwing an exception

**See also**

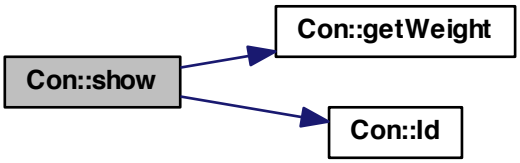
setWeight and the unit test files, e.g., runit.Cpp.Con.R, for usage examples.

Definition at line 136 of file Connection.cpp.

References getWeight(), and Id().

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

Here is the call graph for this function:




#### 5.10.3.7 bool Con::validate ( )

Object validator.

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

**Returns**

true in case the checks are Ok.

**Exceptions**

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

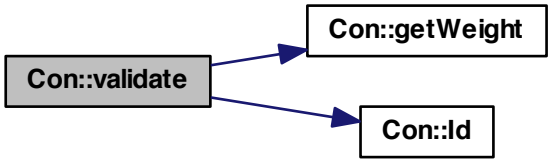
Definition at line 156 of file Connection.cpp.

References getWeight(), and Id().

```
{
```

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

Here is the call graph for this function:




### 5.10.4 Member Data Documentation

#### 5.10.4.1 NeuronRef Con::d_neuron [private]

Definition at line 6 of file Connection.h.

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

#### 5.10.4.2 double Con::d_weight [private]

Definition at line 7 of file Connection.h.

Referenced by getWeight(), and setWeight().

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

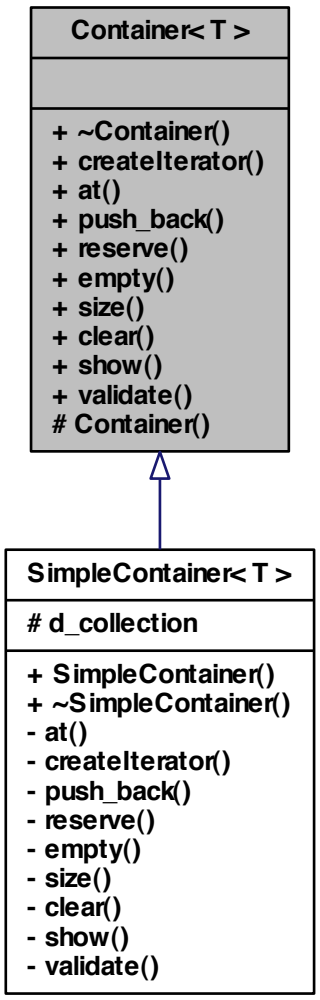
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Connection

## 5.11 Container< T > Class Template Reference

class Container -

`#include <Container.h>`

Inheritance diagram for Container< T >:




## Public Member Functions

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

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

### Protected Member Functions

- Container ()

## 5.11.1 Detailed Description

**template<typename T>class Container< T >**

class Container -

Definition at line 5 of file Container.h.

## 5.11.2 Constructor & Destructor Documentation

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

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

## 5.11.3 Member Function Documentation

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

Implemented in SimpleContainer< T >.

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

Implemented in SimpleContainer< T >.

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

Implemented in SimpleContainer< T >.

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

Implemented in SimpleContainer< T >.

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

Implemented in SimpleContainer< T >.

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

Implemented in SimpleContainer< T >.

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

Implemented in SimpleContainer< T >.

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

Implemented in SimpleContainer< T >.

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

Implemented in SimpleContainer< T >.

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

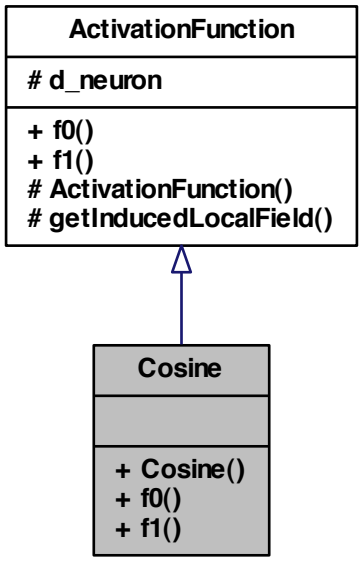
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Container

## 5.12 Cosine Class Reference

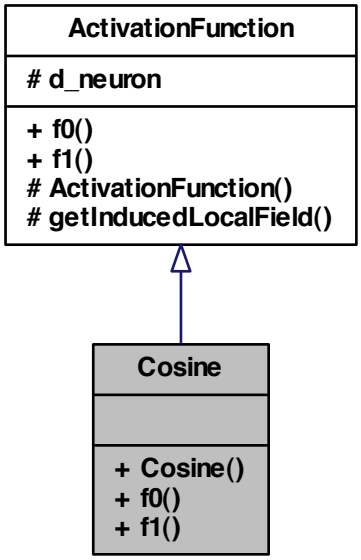
class Cosine -

`#include <Cosine.h>`

Inheritance diagram for Cosine:



Collaboration diagram for Cosine:




## Public Member Functions

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

### 5.12.1 Detailed Description

class Cosine -

Definition at line 5 of file Cosine.h.

### 5.12.2 Constructor & Destructor Documentation

#### 5.12.2.1 Cosine::Cosine ( NeuronPtr *neuronPtr* )

### 5.12.3 Member Function Documentation

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

Implements ActivationFunction.

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

Implements ActivationFunction.

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

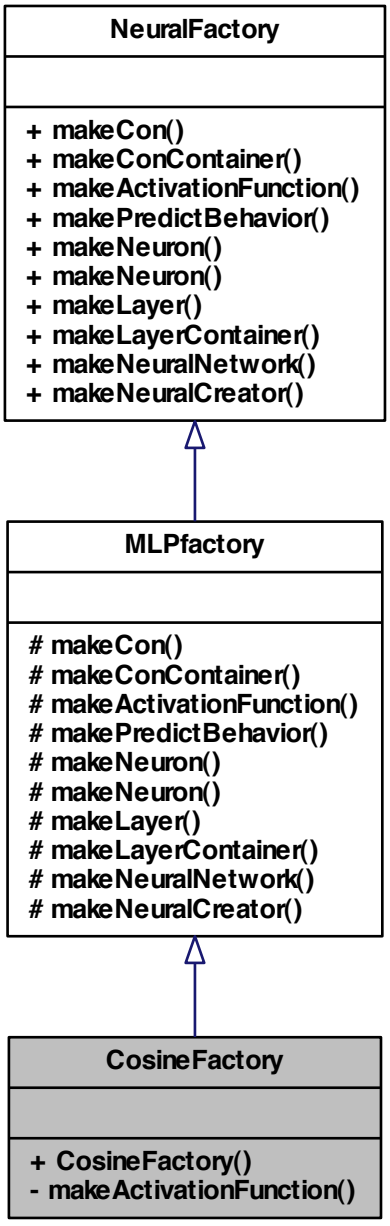
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade

## 5.13 CosineFactory Class Reference

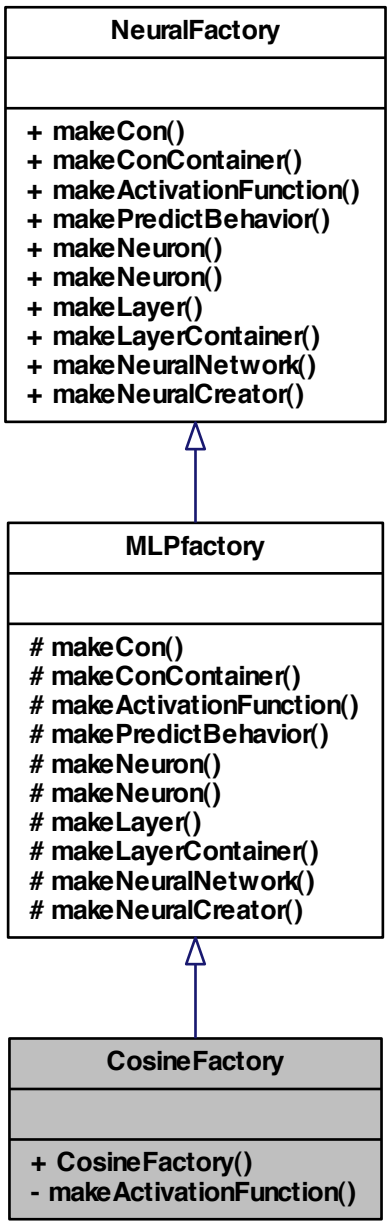
class CosineFactory -

`#include <CosineFactory.h>`

Inheritance diagram for CosineFactory:



Collaboration diagram for CosineFactory:



### Public Member Functions

- CosineFactory ()

### Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

#### 5.13.1 Detailed Description

class CosineFactory -

Definition at line 5 of file CosineFactory.h.

#### 5.13.2 Constructor & Destructor Documentation

##### 5.13.2.1 CosineFactory::CosineFactory ( )

#### 5.13.3 Member Function Documentation

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

Implements MLPfactory.

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

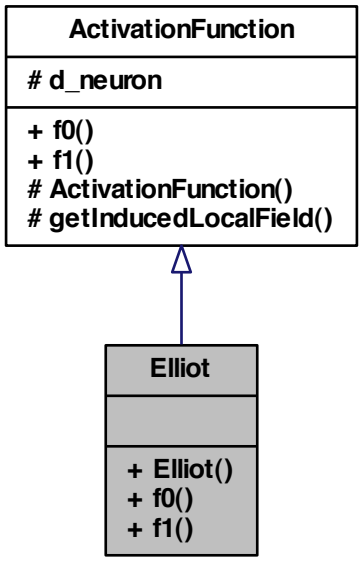
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/CosineFa

## 5.14 Elliot Class Reference

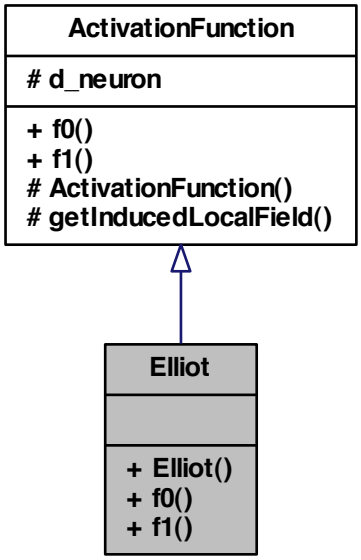
class Elliot -

`#include <Elliot.h>`

Inheritance diagram for Elliot:



Collaboration diagram for Elliot:




## Public Member Functions

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

### 5.14.1 Detailed Description

class Elliot -

Definition at line 5 of file Elliot.h.

### 5.14.2 Constructor & Destructor Documentation

#### 5.14.2.1 Elliot::Elliot ( NeuronPtr *neuronPtr* )

### 5.14.3 Member Function Documentation

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

Implements ActivationFunction.

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

Implements ActivationFunction.

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

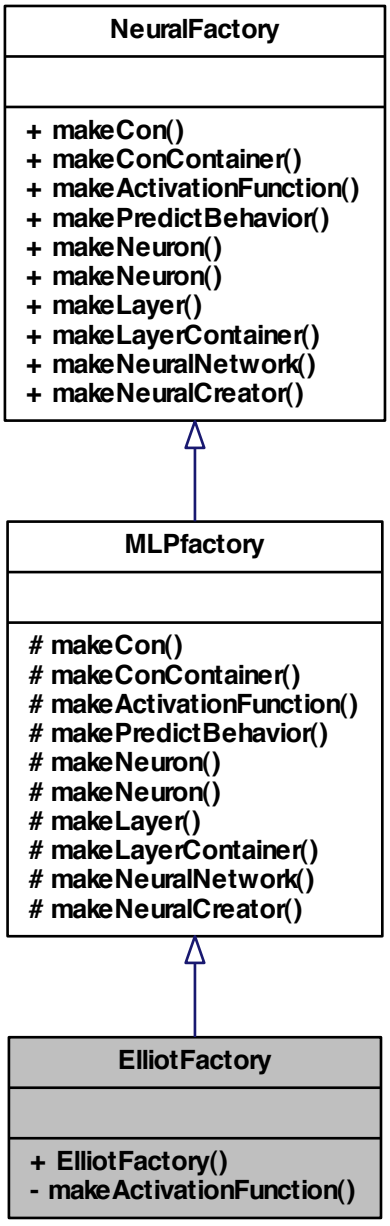
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade

## 5.15 ElliotFactory Class Reference

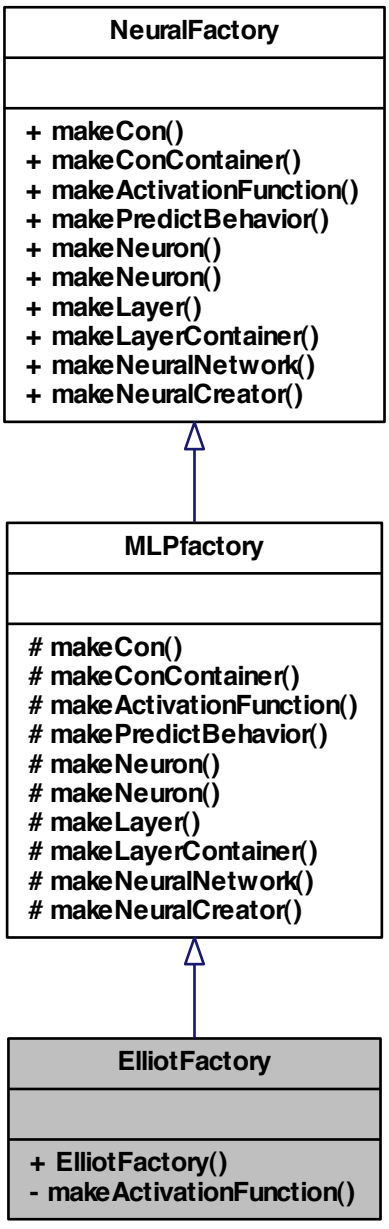
class ElliotFactory -

`#include <ElliotFactory.h>`

Inheritance diagram for ElliotFactory:



Collaboration diagram for ElliotFactory:



### Public Member Functions

- ElliotFactory ()

### Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

#### 5.15.1 Detailed Description

class ElliotFactory -

Definition at line 5 of file ElliotFactory.h.

#### 5.15.2 Constructor & Destructor Documentation

##### 5.15.2.1 ElliotFactory::ElliotFactory ( )

#### 5.15.3 Member Function Documentation

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

Implements MLPfactory.

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

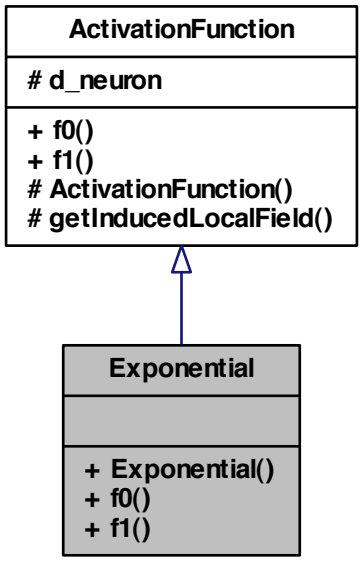
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/ElliotFacto

## 5.16 Exponential Class Reference

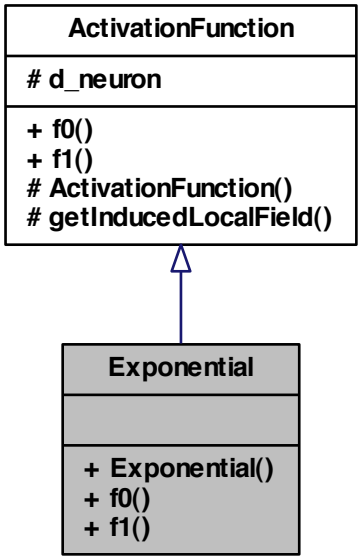
class Exponential -

`#include <Exponential.h>`

Inheritance diagram for Exponential:



Collaboration diagram for Exponential:




## Public Member Functions

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

### 5.16.1 Detailed Description

class Exponential -

Definition at line 5 of file Exponential.h.

### 5.16.2 Constructor & Destructor Documentation

#### 5.16.2.1 Exponential::Exponential ( NeuronPtr *neuronPtr* )

### 5.16.3 Member Function Documentation

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

Implements ActivationFunction.

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

Implements ActivationFunction.

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

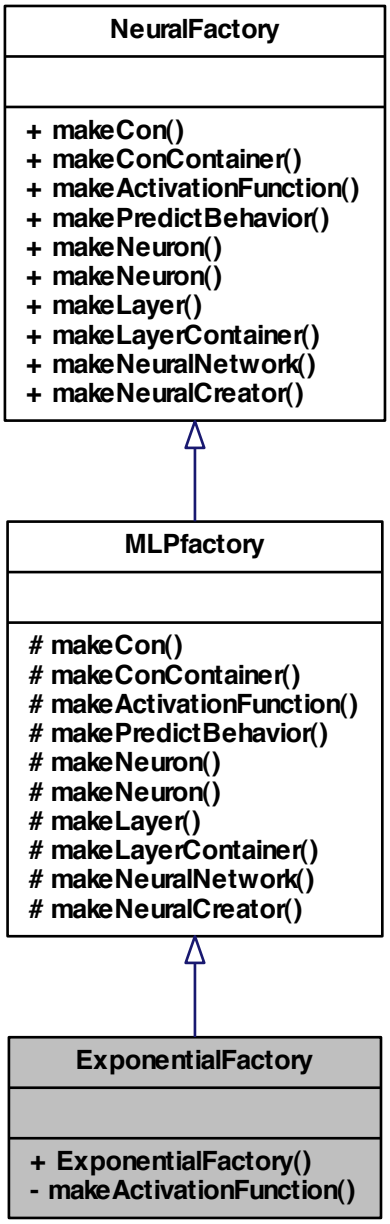
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade

## 5.17 ExponentialFactory Class Reference

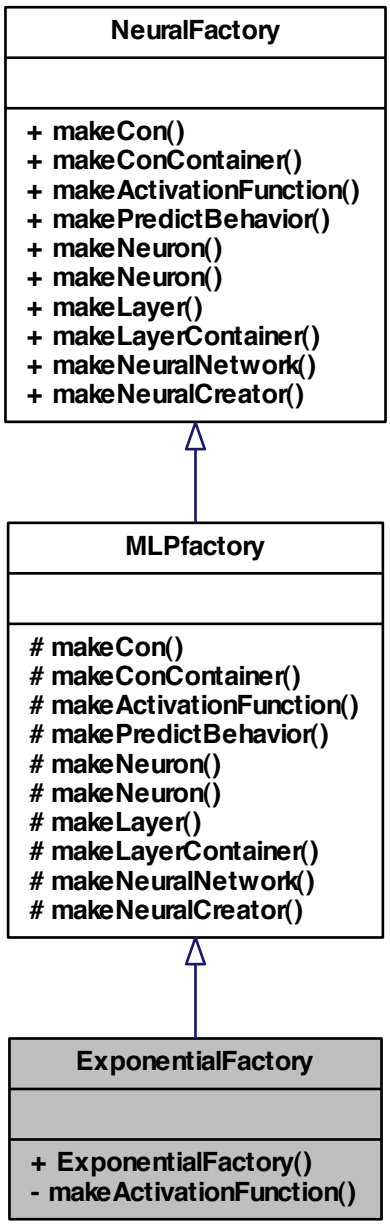
class ExponentialFactory -

`#include <ExponentialFactory.h>`

Inheritance diagram for ExponentialFactory:



Collaboration diagram for ExponentialFactory:



### Public Member Functions

- ExponentialFactory ()

### Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

### 5.17.1 Detailed Description

class ExponentialFactory -

Definition at line 5 of file ExponentialFactory.h.

### 5.17.2 Constructor & Destructor Documentation

#### 5.17.2.1 ExponentialFactory::ExponentialFactory ( )

### 5.17.3 Member Function Documentation

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

Implements MLPfactory.

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

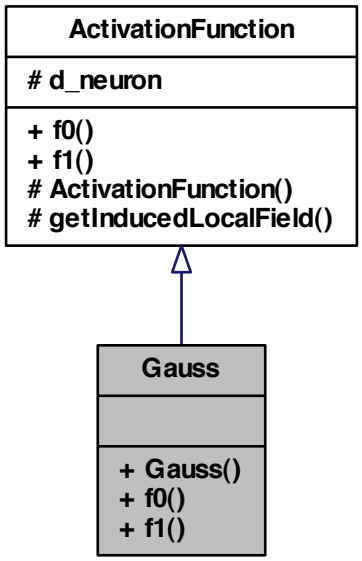
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Exponent

## 5.18 Gauss Class Reference

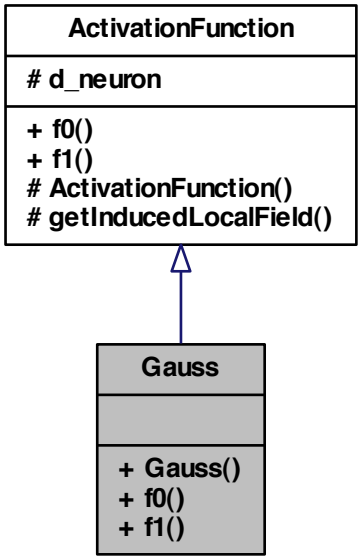
class Gauss -

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

Inheritance diagram for Gauss:



Collaboration diagram for Gauss:




## Public Member Functions

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

### 5.18.1 Detailed Description

class Gauss -

Definition at line 5 of file Gauss.h.

### 5.18.2 Constructor & Destructor Documentation

#### 5.18.2.1 Gauss::Gauss ( NeuronPtr *neuronPtr* )

### 5.18.3 Member Function Documentation

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

Implements ActivationFunction.

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

Implements ActivationFunction.

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

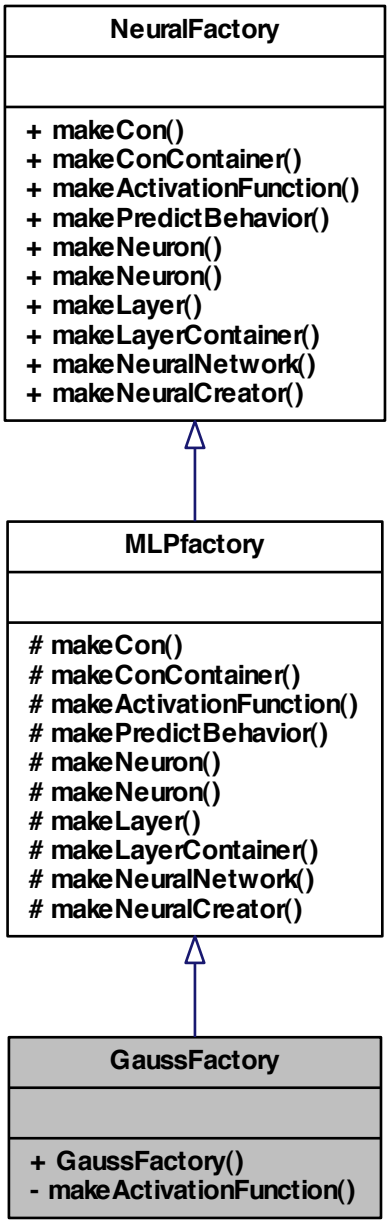
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade

## 5.19 GaussFactory Class Reference

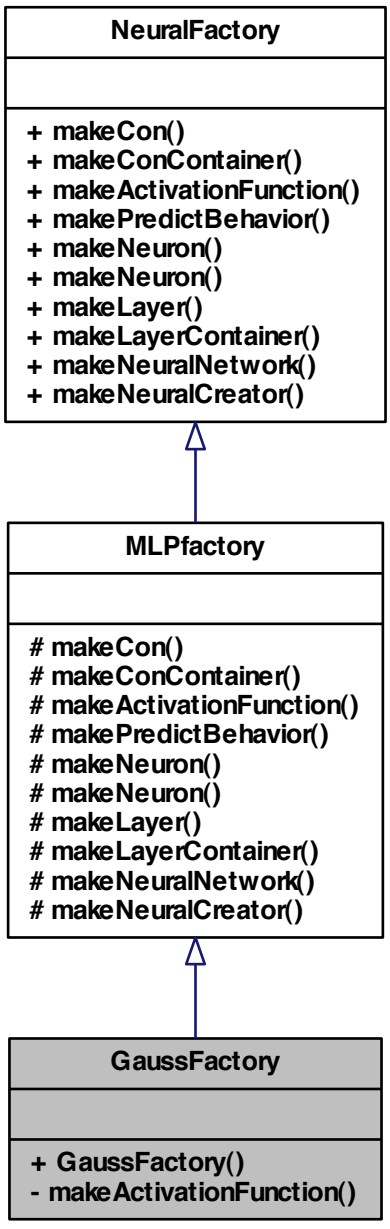
class GaussFactory -

`#include <GaussFactory.h>`

Inheritance diagram for GaussFactory:



Collaboration diagram for GaussFactory:



### Public Member Functions

- GaussFactory ()

### Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

#### 5.19.1 Detailed Description

class GaussFactory -

Definition at line 5 of file GaussFactory.h.

#### 5.19.2 Constructor & Destructor Documentation

##### 5.19.2.1 GaussFactory::GaussFactory ( )

#### 5.19.3 Member Function Documentation

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

Implements MLPfactory.

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

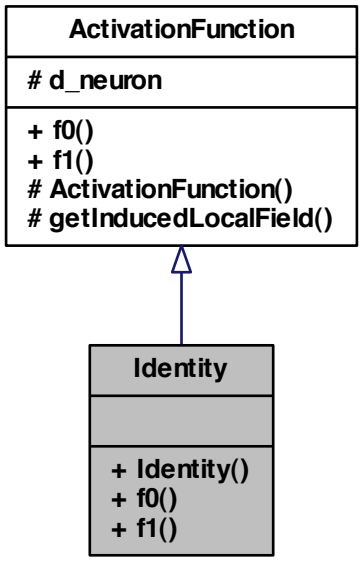
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/GaussFac

## 5.20 Identity Class Reference

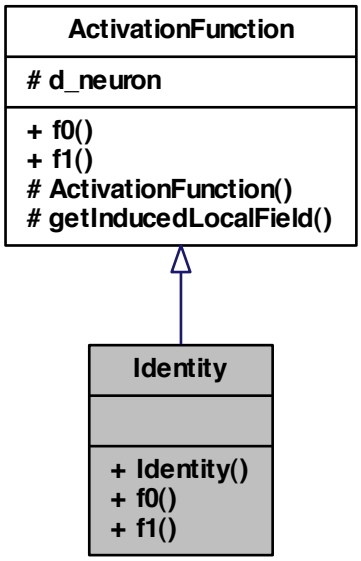
class Identity -

`#include <Identity.h>`

Inheritance diagram for Identity:



Collaboration diagram for Identity:




## Public Member Functions

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

### 5.20.1 Detailed Description

class Identity -

Definition at line 5 of file Identity.h.

### 5.20.2 Constructor & Destructor Documentation

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

Definition at line 13 of file Identity.cpp.

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

```
}
```

### 5.20.3 Member Function Documentation

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

Implements ActivationFunction.

Definition at line 17 of file Identity.cpp.

References ActivationFunction::getInducedLocalField().

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

Here is the call graph for this function:


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

Implements ActivationFunction.

Definition at line 21 of file Identity.cpp.

```
                        {
  return 1 ;
}
```

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

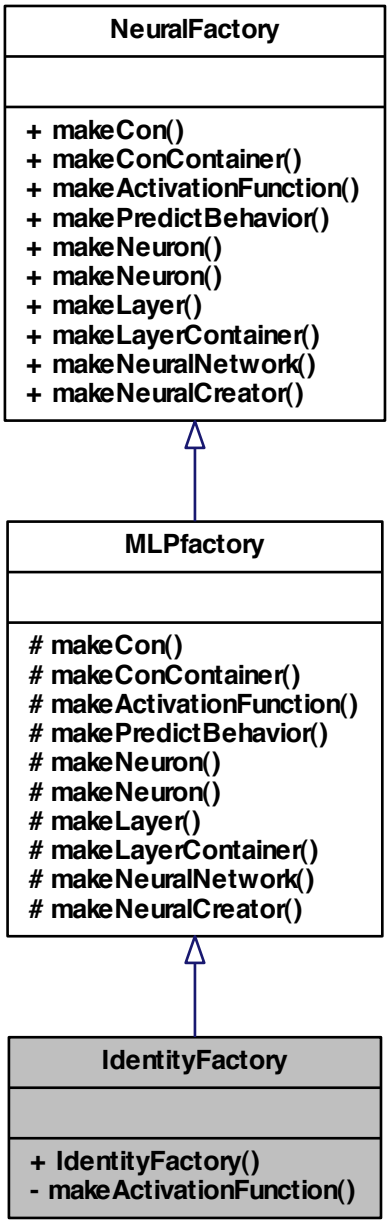
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Identity.cpp

## 5.21 IdentityFactory Class Reference

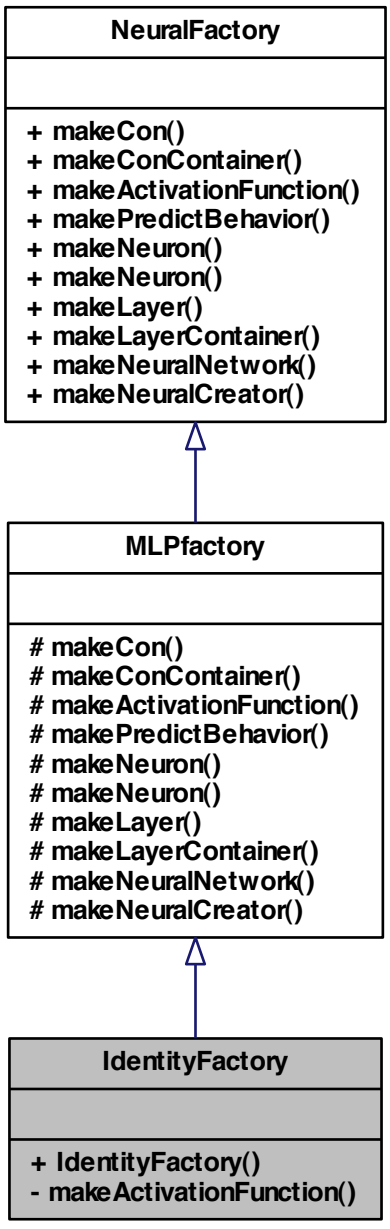
class IdentityFactory -

`#include <IdentityFactory.h>`

Inheritance diagram for IdentityFactory:



Collaboration diagram for IdentityFactory:



### Public Member Functions

- IdentityFactory ()

### Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

### 5.21.1 Detailed Description

class IdentityFactory -

Definition at line 5 of file IdentityFactory.h.

### 5.21.2 Constructor & Destructor Documentation

#### 5.21.2.1 IdentityFactory::IdentityFactory ( )

Definition at line 14 of file IdentityFactory.cpp.

```
{
}
```

### 5.21.3 Member Function Documentation

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

Implements MLPfactory.

Definition at line 20 of file IdentityFactory.cpp.

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

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

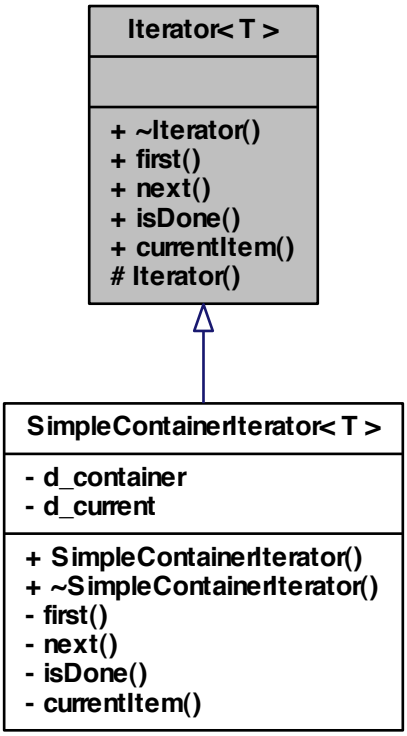
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/IdentityFa
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/IdentityFactory.cpp

## 5.22 Iterator< T > Class Template Reference

class Iterator -

`#include <Iterator.h>`

Inheritance diagram for Iterator< T >:




## Public Member Functions

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

## Protected Member Functions

- Iterator ()

### 5.22.1 Detailed Description

**template<typename T>class Iterator< T >**

class Iterator -

Definition at line 5 of file Iterator.h.

### 5.22.2 Constructor & Destructor Documentation

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

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

### 5.22.3 Member Function Documentation

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

Implemented in SimpleContainerIterator< T >.

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

Implemented in SimpleContainerIterator< T >.

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

Implemented in SimpleContainerIterator< T >.

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

Implemented in SimpleContainerIterator< T >.

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

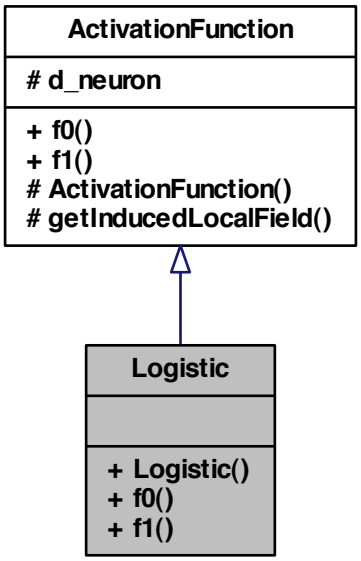
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Iterator.h

## 5.23 Logistic Class Reference

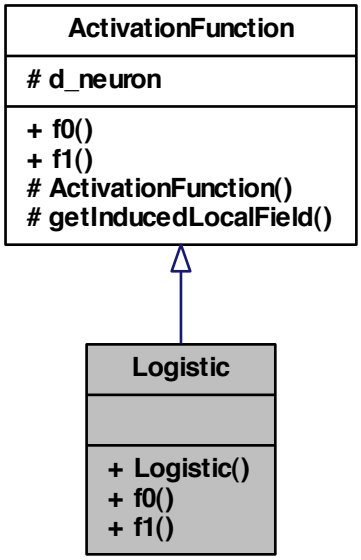
class Logistic -

`#include <Logistic.h>`

Inheritance diagram for Logistic:



Collaboration diagram for Logistic:




## Public Member Functions

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

### 5.23.1 Detailed Description

class Logistic -

Definition at line 5 of file Logistic.h.

### 5.23.2 Constructor & Destructor Documentation

#### 5.23.2.1 Logistic::Logistic ( NeuronPtr *neuronPtr* )

### 5.23.3 Member Function Documentation

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

Implements ActivationFunction.

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

Implements ActivationFunction.

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

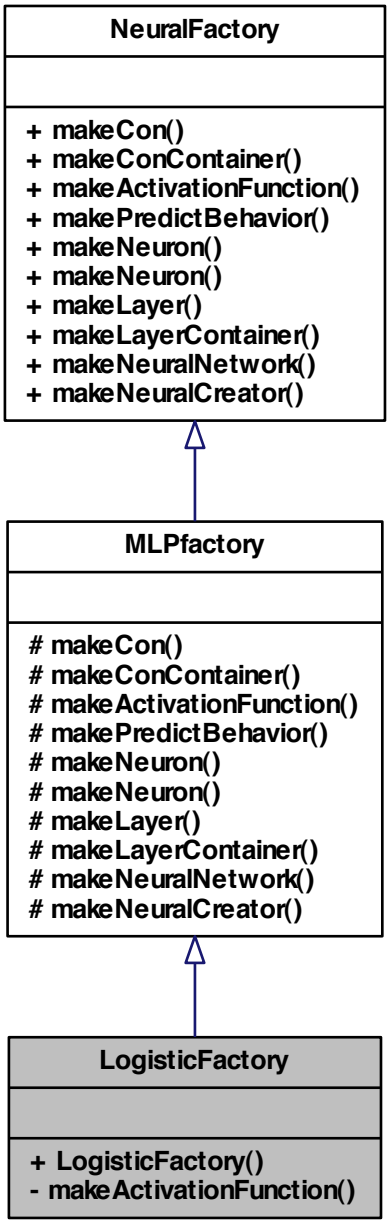
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade

## 5.24 LogisticFactory Class Reference

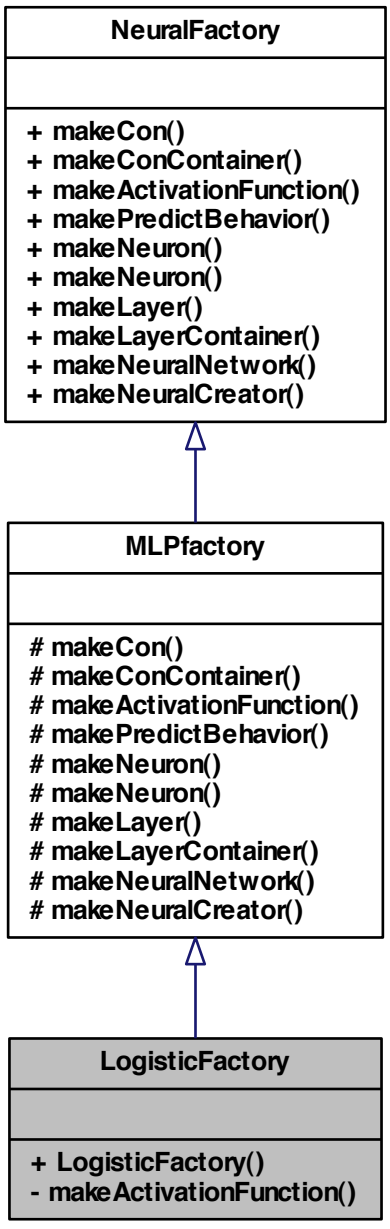
class LogisticFactory -

`#include <LogisticFactory.h>`

Inheritance diagram for LogisticFactory:



Collaboration diagram for LogisticFactory:



### Public Member Functions

- LogisticFactory ()

### Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

#### 5.24.1 Detailed Description

class LogisticFactory -

Definition at line 5 of file LogisticFactory.h.

#### 5.24.2 Constructor & Destructor Documentation

##### 5.24.2.1 LogisticFactory::LogisticFactory ( )

#### 5.24.3 Member Function Documentation

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

Implements MLPfactory.

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

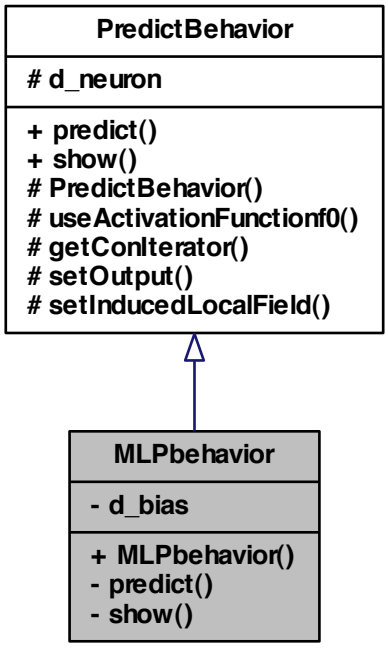
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/LogisticFa

## 5.25 MLPbehavior Class Reference

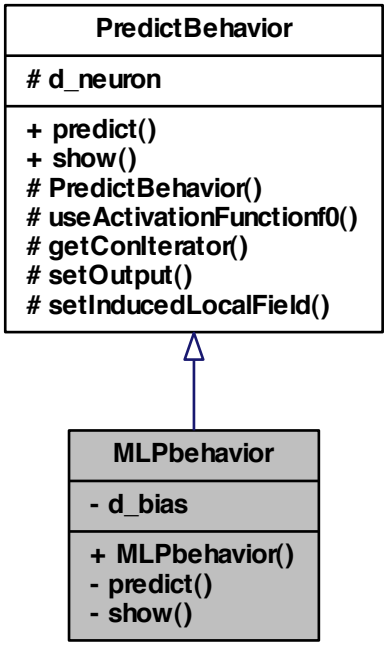
class MLPbehavior -

`#include <MLPbehavior.h>`

Inheritance diagram for MLPbehavior:



Collaboration diagram for MLPbehavior:




## Public Member Functions

- MLPbehavior (NeuronPtr neuronPtr)

## Private Member Functions

- void predict ()
- void show ()

## Private Attributes

- double d_bias

## Friends

- class MLPfactory

### 5.25.1 Detailed Description

class MLPbehavior -

Definition at line 5 of file MLPbehavior.h.

### 5.25.2 Constructor & Destructor Documentation

#### 5.25.2.1 MLPbehavior::MLPbehavior ( NeuronPtr *neuronPtr* )

Definition at line 17 of file MLPbehavior.cpp.

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

### 5.25.3 Member Function Documentation

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

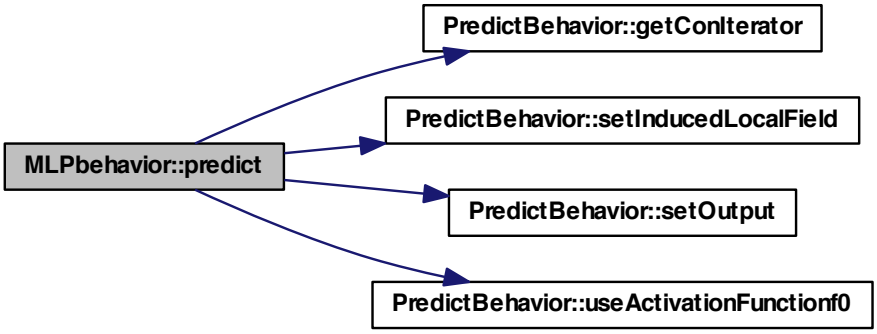
Implements PredictBehavior.

Definition at line 23 of file MLPbehavior.cpp.

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

```
{

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

Here is the call graph for this function:




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

Implements PredictBehavior.

Definition at line 42 of file MLPbehavior.cpp.

References d_bias.

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

### 5.25.4 Friends And Related Function Documentation

#### 5.25.4.1 friend class MLPfactory [friend]

Definition at line 11 of file MLPbehavior.h.

### 5.25.5 Member Data Documentation

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

Definition at line 8 of file MLPbehavior.h.

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

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

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

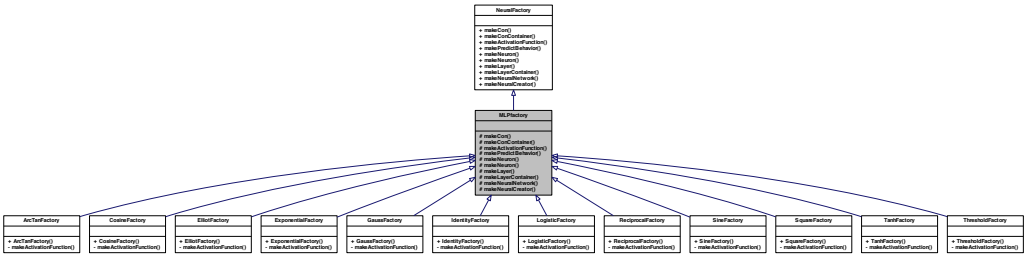
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPbehavi

## 5.26 MLPfactory Class Reference

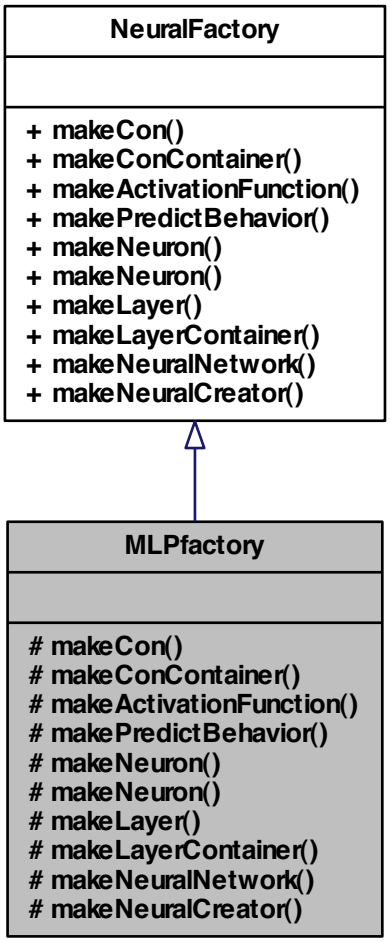
class MLPfactory -

`#include <MLPfactory.h>`

Inheritance diagram for MLPfactory:



Collaboration diagram for MLPfactory:




## Protected Member Functions

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

- LayerPtr makeLayer ()
- LayerContainerPtr makeLayerContainer ()
- NeuralNetworkPtr makeNeuralNetwork (NeuralFactory &neuralFactory)
- NeuralCreatorPtr makeNeuralCreator ()

### 5.26.1 Detailed Description

class MLPfactory -

Definition at line 5 of file MLPfactory.h.

### 5.26.2 Member Function Documentation

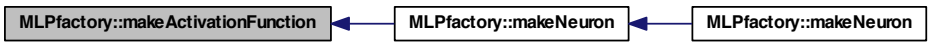
#### 5.26.2.1 virtual ActivationFunctionPtr MLPfactory::makeActivationFunction ( NeuronPtr *neuronPtr* ) [protected, pure virtual]

Implements NeuralFactory.

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

Referenced by makeNeuron().

Here is the caller graph for this function:




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

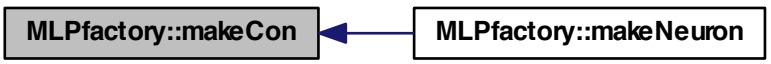
Implements NeuralFactory.

Definition at line 30 of file MLPfactory.cpp.

Referenced by makeNeuron().

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

Here is the caller graph for this function:




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

Implements NeuralFactory.

Definition at line 37 of file MLPfactory.cpp.

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

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

Implements NeuralFactory.

Definition at line 84 of file MLPfactory.cpp.

Referenced by makeLayerContainer().

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

Here is the caller graph for this function:

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

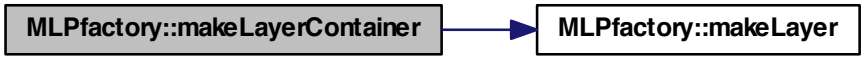
Implements NeuralFactory.

Definition at line 92 of file MLPfactory.cpp.

References makeLayer().

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

Here is the call graph for this function:




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

Implements NeuralFactory.

Definition at line 109 of file MLPfactory.cpp.

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

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

Implements NeuralFactory.

Definition at line 101 of file MLPfactory.cpp.

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

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

Implements NeuralFactory.

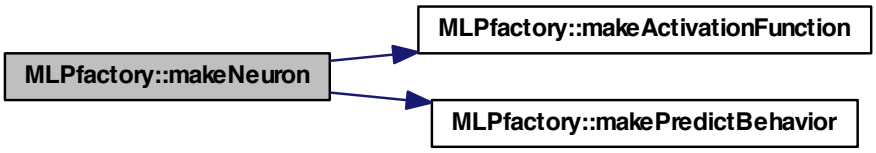
Definition at line 52 of file MLPfactory.cpp.

References makeActivationFunction(), and makePredictBehavior().

Referenced by makeNeuron().

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

Here is the call graph for this function:




Here is the caller graph for this function:


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

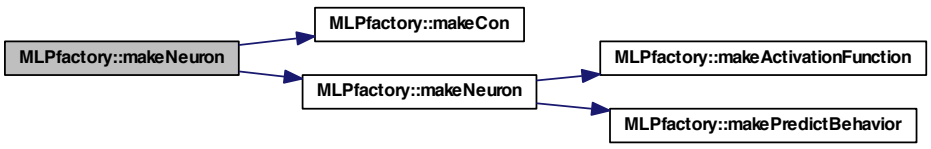
Implements NeuralFactory.

Definition at line 62 of file MLPfactory.cpp.

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

```
{
  RNGScope scope;

  NeuronPtr neuronPtr(makeNeuron(Id));

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

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

return neuronPtr;
}
```

Here is the call graph for this function:




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

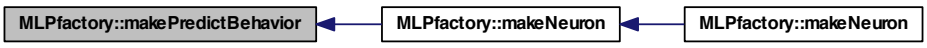
Implements NeuralFactory.

Definition at line 45 of file MLPfactory.cpp.

Referenced by makeNeuron().

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

Here is the caller graph for this function:




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

- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPfacto
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPfactory.cpp

## 5.27 NetworkRinterface Class Reference

class NetworkRinterface -

`#include <NetworkRinterface.h>`

### Public Member Functions

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

### Private Attributes

- NeuralNetworkPtr d_neuralNetwork

### 5.27.1 Detailed Description

class NetworkRinterface -

Definition at line 3 of file NetworkRinterface.h.

### 5.27.2 Constructor & Destructor Documentation

#### 5.27.2.1 NetworkRinterface::NetworkRinterface ( )

Definition at line 22 of file NetworkRinterface.cpp.

```
{

}
```

### 5.27.3 Member Function Documentation

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

Definition at line 28 of file NetworkRinterface.cpp.

References d_neuralNetwork.

Referenced by RCPP_MODULE().

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

Here is the caller graph for this function:


#### 5.27.3.2 size_type NetworkRinterface::inputSize ( )

Definition at line 70 of file NetworkRinterface.cpp.

References d_neuralNetwork.

Referenced by predict(), and RCPP_MODULE().

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

Here is the caller graph for this function:


#### 5.27.3.3 size_type NetworkRinterface::outputSize ( )

Definition at line 76 of file NetworkRinterface.cpp.

References d_neuralNetwork.

Referenced by predict(), and RCPP_MODULE().

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

Here is the caller graph for this function:


#### 5.27.3.4 Rcpp::NumericMatrix NetworkRinterface::predict ( Rcpp::NumericMatrix *numericMatrix* )

Definition at line 39 of file NetworkRinterface.cpp.

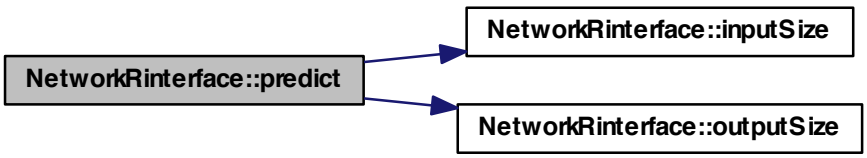
References d_neuralNetwork, inputSize(), and outputSize().

Referenced by RCPP_MODULE().

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

```
      reate methods available.\n");
    }

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

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

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

END_RCPP}
```

Here is the call graph for this function:




Here is the caller graph for this function:

#### 5.27.3.5 void NetworkRinterface::show ( )

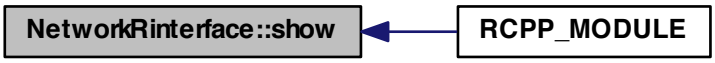
Definition at line 82 of file NetworkRinterface.cpp.

References d_neuralNetwork.

Referenced by RCPP_MODULE().

```
{

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

Here is the caller graph for this function:




#### 5.27.3.6 bool NetworkRinterface::validate ( )

Definition at line 97 of file NetworkRinterface.cpp.

References d_neuralNetwork.

Referenced by RCPP_MODULE().

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

```
END_RCPP
}
```

Here is the caller graph for this function:


### 5.27.4 Member Data Documentation

#### 5.27.4.1 NeuralNetworkPtr NetworkRinterface::d_neuralNetwork [private]

Definition at line 6 of file NetworkRinterface.h.

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

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

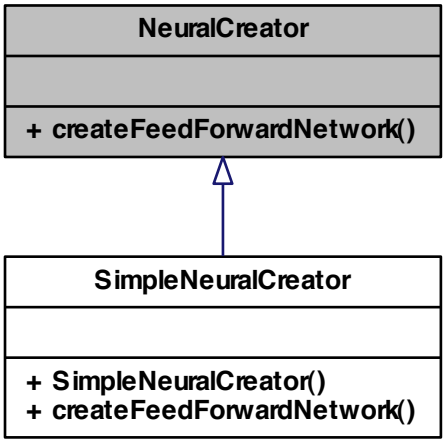
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NetworkRir

## 5.28 NeuralCreator Class Reference

class NeuralCreator -

`#include <NeuralCreator.h>`

Inheritance diagram for NeuralCreator:




### Public Member Functions

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

## 5.28.1 Detailed Description

class NeuralCreator -

Definition at line 4 of file NeuralCreator.h.

## 5.28.2 Member Function Documentation

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

Implemented in SimpleNeuralCreator.

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

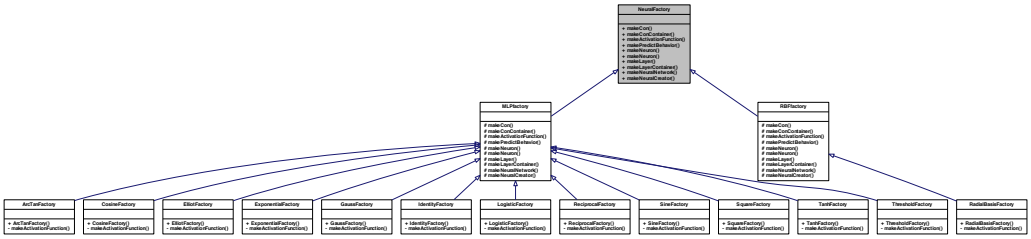
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralCre

## 5.29 NeuralFactory Class Reference

class NeuralFactory -

`#include <NeuralFactory.h>`

Inheritance diagram for NeuralFactory:



### Public Member Functions

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

### 5.29.1 Detailed Description

class NeuralFactory -

Definition at line 4 of file NeuralFactory.h.

### 5.29.2 Member Function Documentation

#### 5.29.2.1 virtual ActivationFunctionPtr NeuralFactory::makeActivationFunction ( NeuronPtr *neuronPtr* ) [pure virtual]

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

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

Implemented in MLPfactory.

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

Implemented in MLPfactory, and RBFfactory.

Referenced by Neuron::Neuron().

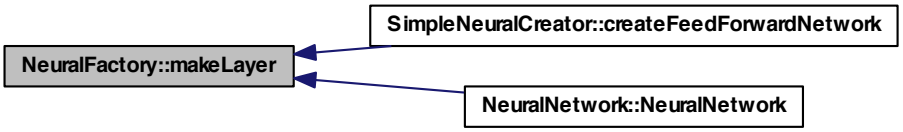
Here is the caller graph for this function:


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

Implemented in MLPfactory, and RBFfactory.

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

Here is the caller graph for this function:

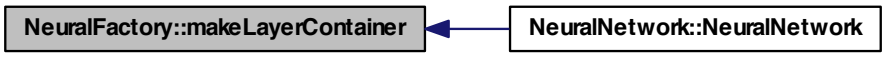



#### 5.29.2.5 virtual LayerContainerPtr NeuralFactory::makeLayerContainer ( ) [pure virtual]

Implemented in MLPfactory, and RBFfactory.

Referenced by NeuralNetwork::NeuralNetwork().

Here is the caller graph for this function:




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

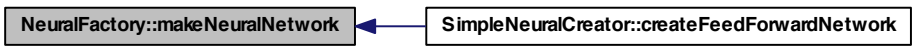
Implemented in MLPfactory, and RBFfactory.

#### 5.29.2.7 virtual NeuralNetworkPtr NeuralFactory::makeNeuralNetwork ( NeuralFactory & *neuralFactory* ) [pure virtual]

Implemented in MLPfactory, and RBFfactory.

Referenced by SimpleNeuralCreator::createFeedForwardNetwork().

Here is the caller graph for this function:




#### 5.29.2.8 virtual NeuronPtr NeuralFactory::makeNeuron ( Handler *id* ) [pure virtual]

Implemented in MLPfactory, and RBFfactory.

Referenced by SimpleNeuralCreator::createFeedForwardNetwork().

Here is the caller graph for this function:


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

Implemented in MLPfactory, and RBFfactory.

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

Implemented in MLPfactory.

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

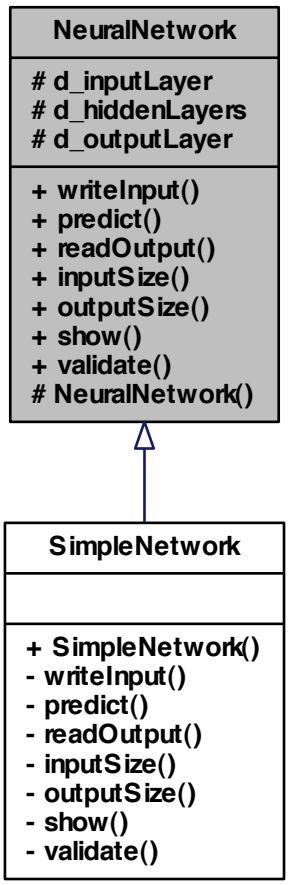
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralFac

## 5.30 NeuralNetwork Class Reference

class NeuralNetwork -

`#include <NeuralNetwork.h>`

Inheritance diagram for NeuralNetwork:




## Public Member Functions

- virtual void writeInput (std::vector< double >::iterator &iterator)=0
- virtual void predict ()=0
- virtual void readOutput (std::vector< double >::iterator &iterator)=0
- virtual size_type inputSize ()=0
- virtual size_type outputSize ()=0
- virtual void show ()=0
- virtual bool validate ()=0

### Protected Member Functions

- NeuralNetwork (NeuralFactory &neuralFactory)

### Protected Attributes

- LayerPtr d_inputLayer
- LayerContainerPtr d_hiddenLayers
- LayerPtr d_outputLayer

### Friends

- class SimpleNeuralCreator

## 5.30.1 Detailed Description

class NeuralNetwork -

Definition at line 3 of file NeuralNetwork.h.

## 5.30.2 Constructor & Destructor Documentation

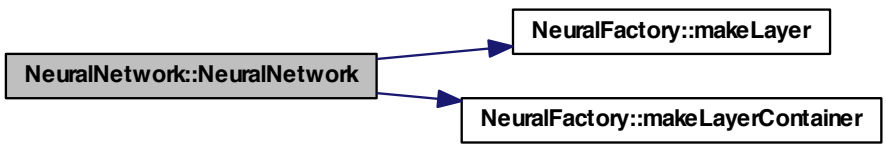
### 5.30.2.1 NeuralNetwork::NeuralNetwork ( NeuralFactory & *neuralFactory* ) [protected]

Definition at line 12 of file NeuralNetwork.cpp.

References d_hiddenLayers, d_inputLayer, d_outputLayer, NeuralFactory::makeLayer(), and NeuralFactory::makeLayerContainer().

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

Here is the call graph for this function:



### 5.30.3 Member Function Documentation

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

Implemented in SimpleNetwork.

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

Implemented in SimpleNetwork.

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

Implemented in SimpleNetwork.

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

Implemented in SimpleNetwork.

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

Implemented in SimpleNetwork.

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

Implemented in SimpleNetwork.

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

Implemented in SimpleNetwork.

### 5.30.4 Friends And Related Function Documentation

#### 5.30.4.1 friend class SimpleNeuralCreator [friend]

Definition at line 11 of file NeuralNetwork.h.

### 5.30.5 Member Data Documentation

#### 5.30.5.1 LayerContainerPtr NeuralNetwork::d_hiddenLayers [protected]

Definition at line 7 of file NeuralNetwork.h.

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

#### 5.30.5.2 LayerPtr NeuralNetwork::d_inputLayer [protected]

Definition at line 6 of file NeuralNetwork.h.

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

#### 5.30.5.3 LayerPtr NeuralNetwork::d_outputLayer [protected]

Definition at line 8 of file NeuralNetwork.h.

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

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

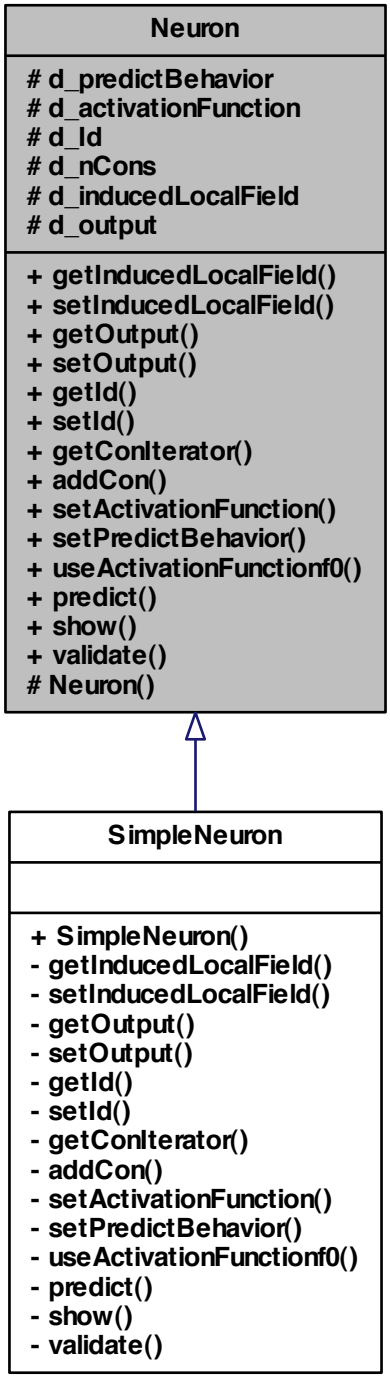
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/NeuralNe
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NeuralNetwork.cpp

## 5.31 Neuron Class Reference

class Neuron -

`#include <Neuron.h>`

Inheritance diagram for Neuron:



### Public Member Functions

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

### Protected Member Functions

- Neuron (NeuralFactory &neuralFactory)

### Protected Attributes

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

### Friends

- class MLPfactory

#### 5.31.1 Detailed Description

class Neuron -

Definition at line 3 of file Neuron.h.

### 5.31.2 Constructor & Destructor Documentation

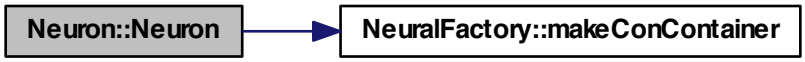
#### 5.31.2.1 Neuron::Neuron ( NeuralFactory & *neuralFactory* ) [protected]

Definition at line 12 of file Neuron.cpp.

References d_nCons, and NeuralFactory::makeConContainer().

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

Here is the call graph for this function:




### 5.31.3 Member Function Documentation

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

Implemented in SimpleNeuron.

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

Implemented in SimpleNeuron.

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

Implemented in SimpleNeuron.

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

Implemented in SimpleNeuron.

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

Implemented in SimpleNeuron.

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

Implemented in SimpleNeuron.

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

Implemented in SimpleNeuron.

#### 5.31.3.8 virtual void Neuron::setId ( Handler *id* ) [pure virtual]

Implemented in SimpleNeuron.

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

Implemented in SimpleNeuron.

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

Implemented in SimpleNeuron.

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

Implemented in SimpleNeuron.

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

Implemented in SimpleNeuron.

#### 5.31.3.13 virtual double Neuron::useActivationFunctionf0 ( ) [pure virtual]

Implemented in SimpleNeuron.

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

Implemented in SimpleNeuron.

### 5.31.4 Friends And Related Function Documentation

#### 5.31.4.1 friend class MLPfactory [friend]

Definition at line 15 of file Neuron.h.

### 5.31.5 Member Data Documentation

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

Definition at line 7 of file Neuron.h.

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

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

Definition at line 9 of file Neuron.h.

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

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

Definition at line 11 of file Neuron.h.

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

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

Definition at line 10 of file Neuron.h.

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

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

Definition at line 12 of file Neuron.h.

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

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

Definition at line 6 of file Neuron.h.

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

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

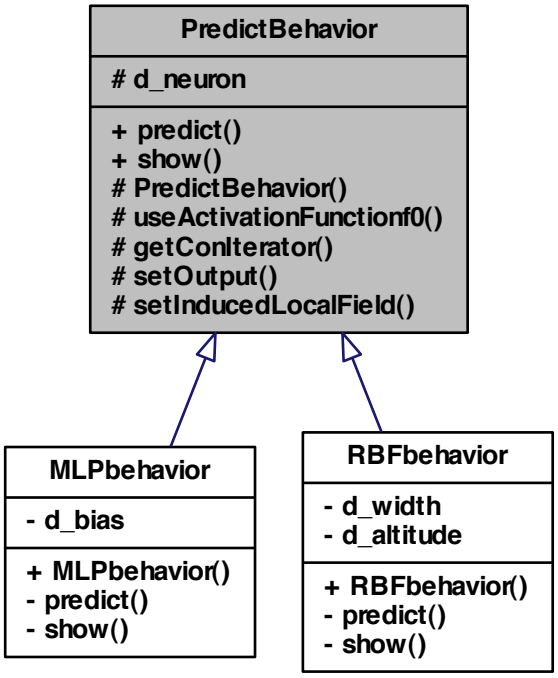
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Neuron.h
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Neuron.cpp

## 5.32 PredictBehavior Class Reference

class PredictBehavior -

`#include <PredictBehavior.h>`

Inheritance diagram for PredictBehavior:




### Public Member Functions

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

### Protected Member Functions

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

### Protected Attributes

- NeuronWeakPtr d_neuron

## 5.32.1 Detailed Description

class PredictBehavior -

Definition at line 4 of file PredictBehavior.h.

## 5.32.2 Constructor & Destructor Documentation

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

Definition at line 14 of file PredictBehavior.cpp.

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

## 5.32.3 Member Function Documentation

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

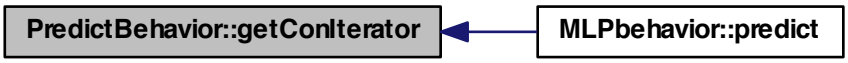
Definition at line 28 of file PredictBehavior.cpp.

References d_neuron.

Referenced by MLPbehavior::predict().

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

Here is the caller graph for this function:




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

Implemented in MLPbehavior, and RBFbehavior.

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

Definition at line 42 of file PredictBehavior.cpp.

References d_neuron.

Referenced by MLPbehavior::predict().

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

Here is the caller graph for this function:


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

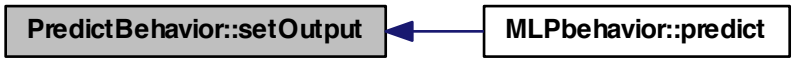
Definition at line 35 of file PredictBehavior.cpp.

References d_neuron.

Referenced by MLPbehavior::predict().

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

Here is the caller graph for this function:




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

Implemented in MLPbehavior, and RBFbehavior.

#### 5.32.3.6 double PredictBehavior::useActivationFunctionf0 ( ) [protected]

Definition at line 20 of file PredictBehavior.cpp.

References d_neuron.

Referenced by MLPbehavior::predict().

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

}
```

Here is the caller graph for this function:

### 5.32.4 Member Data Documentation

#### 5.32.4.1 NeuronWeakPtr PredictBehavior::d_neuron [protected]

Definition at line 7 of file PredictBehavior.h.

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

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

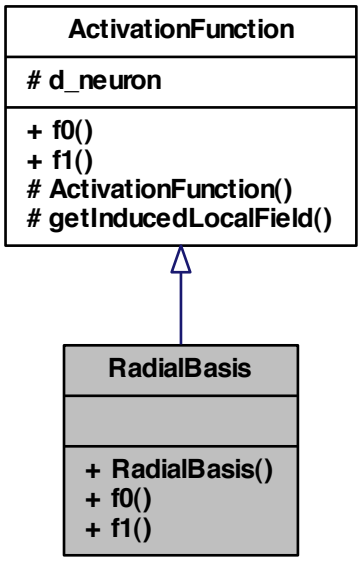
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/PredictBe
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/PredictBehavior.cpp

## 5.33 RadialBasis Class Reference

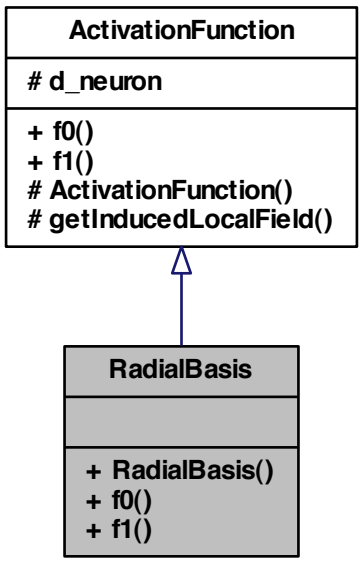
class RadialBasis -

`#include <RadialBasis.h>`

Inheritance diagram for RadialBasis:



Collaboration diagram for RadialBasis:




## Public Member Functions

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

### 5.33.1 Detailed Description

class RadialBasis -

Definition at line 5 of file RadialBasis.h.

### 5.33.2 Constructor & Destructor Documentation

#### 5.33.2.1 RadialBasis::RadialBasis ( NeuronPtr *neuronPtr* )

### 5.33.3 Member Function Documentation

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

Implements ActivationFunction.

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

Implements ActivationFunction.

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

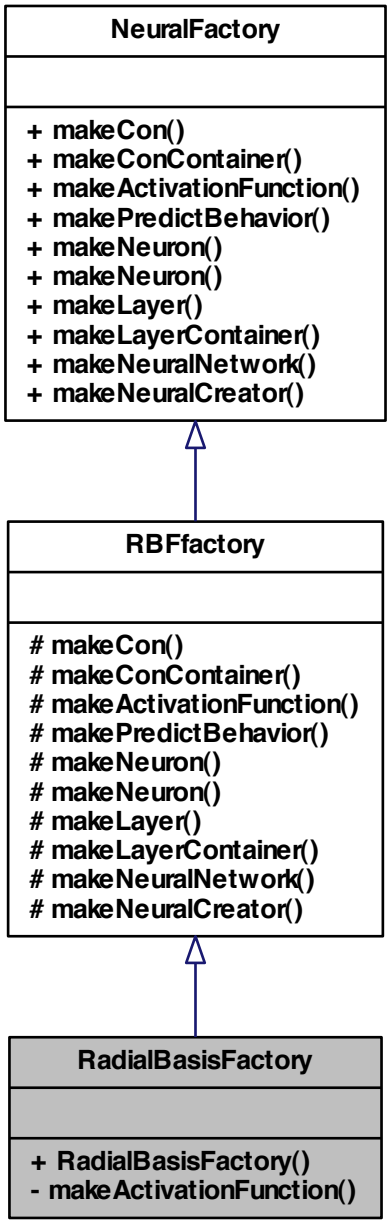
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RadialBas

## 5.34 RadialBasisFactory Class Reference

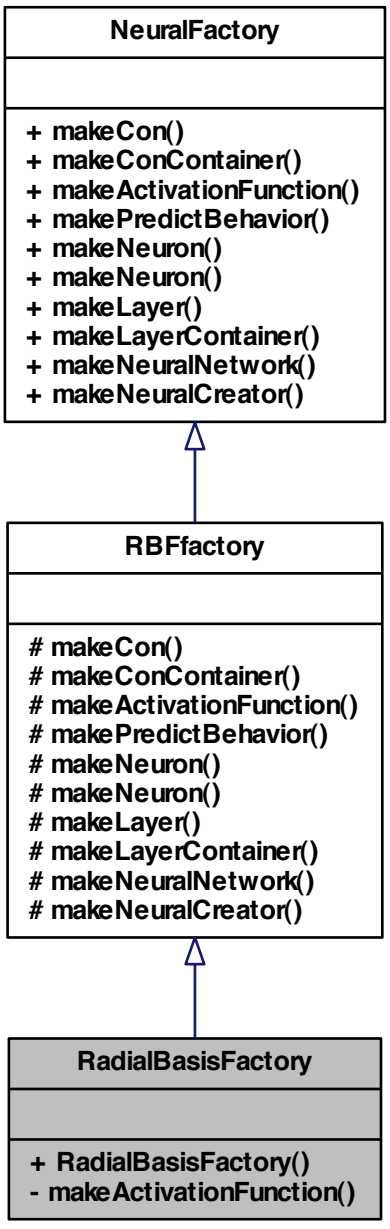
class RadialBasisFactory -

`#include <RadialBasisFactory.h>`

Inheritance diagram for RadialBasisFactory:



Collaboration diagram for RadialBasisFactory:



### Public Member Functions

- RadialBasisFactory ()

### Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

#### 5.34.1 Detailed Description

class RadialBasisFactory -

Definition at line 5 of file RadialBasisFactory.h.

#### 5.34.2 Constructor & Destructor Documentation

##### 5.34.2.1 RadialBasisFactory::RadialBasisFactory ( )

#### 5.34.3 Member Function Documentation

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

Implements RBFfactory.

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

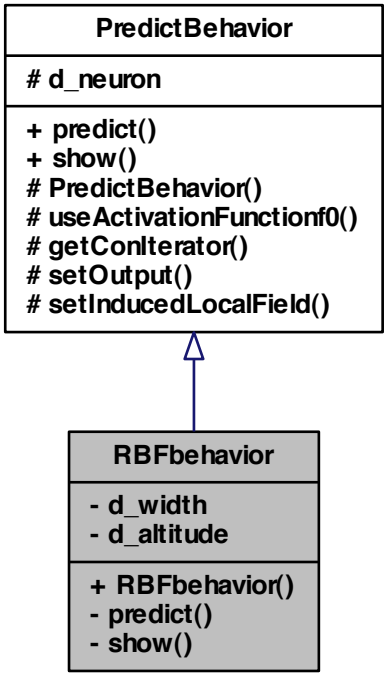
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade

## 5.35 RBFbehavior Class Reference

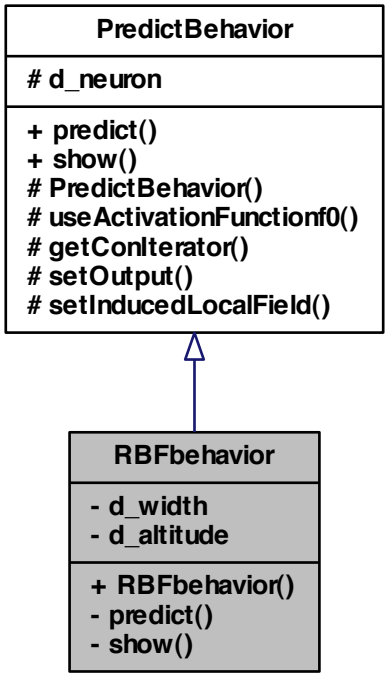
class RBFbehavior -

`#include <RBFbehavior.h>`

Inheritance diagram for RBFbehavior:



Collaboration diagram for RBFbehavior:




## Public Member Functions

- RBFbehavior (NeuronPtr neuronPtr)

## Private Member Functions

- void predict ()
- void show ()

## Private Attributes

- double d_width
- double d_altitude

### 5.35.1 Detailed Description

class RBFbehavior -

Definition at line 5 of file RBFbehavior.h.

### 5.35.2 Constructor & Destructor Documentation

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

### 5.35.3 Member Function Documentation

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

Implements PredictBehavior.

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

Implements PredictBehavior.

### 5.35.4 Member Data Documentation

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

Definition at line 9 of file RBFbehavior.h.

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

Definition at line 8 of file RBFbehavior.h.

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

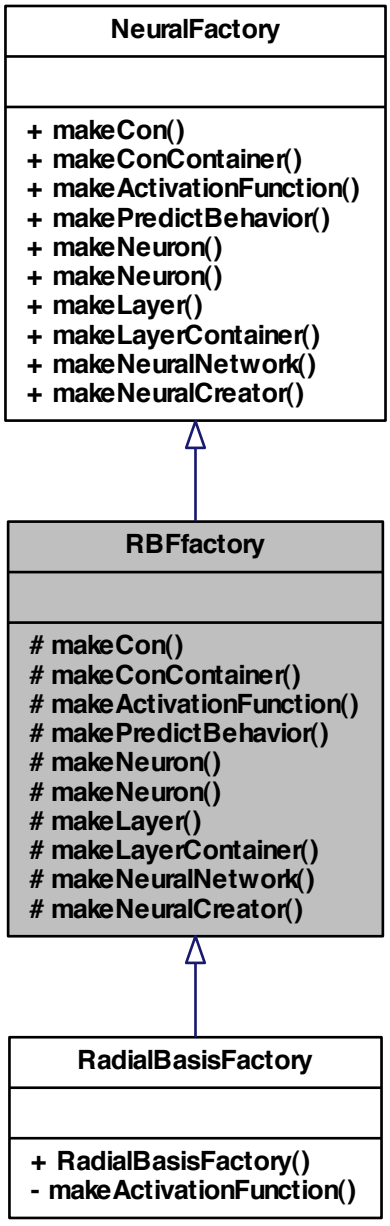
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFbeha

## 5.36 RBFfactory Class Reference

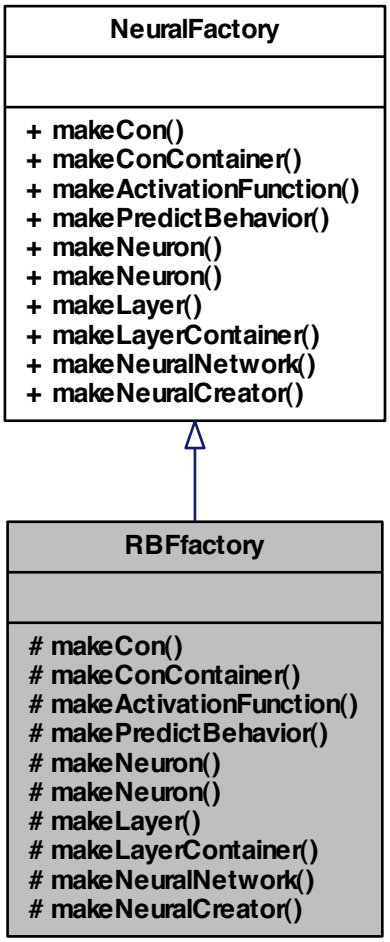
class RBFfactory -

`#include <RBFfactory.h>`

Inheritance diagram for RBFfactory:



Collaboration diagram for RBFfactory:




## Protected Member Functions

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

- LayerPtr makeLayer ()
- LayerContainerPtr makeLayerContainer ()
- NeuralNetworkPtr makeNeuralNetwork (NeuralFactory &neuralFactory)
- NeuralCreatorPtr makeNeuralCreator ()

### 5.36.1 Detailed Description

class RBFfactory -

Definition at line 5 of file RBFfactory.h.

### 5.36.2 Member Function Documentation

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

Implements NeuralFactory.

Implemented in RadialBasisFactory.

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

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

Implements NeuralFactory.

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

Implements NeuralFactory.

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

Implements NeuralFactory.

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

Implements NeuralFactory.

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

Implements NeuralFactory.

#### 5.36.2.8 NeuronPtr RBFfactory::makeNeuron ( Handler *id* ) [protected, virtual]

Implements NeuralFactory.

#### 5.36.2.9 NeuronPtr RBFfactory::makeNeuron ( Handler *id*, NeuronIteratorPtr *neuronIteratorPtr*, double *totalAmountOfParameters* ) [protected, virtual]

Implements NeuralFactory.

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

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

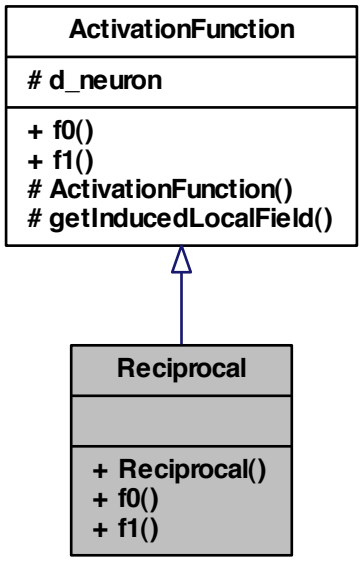
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFfactor

## 5.37 Reciprocal Class Reference

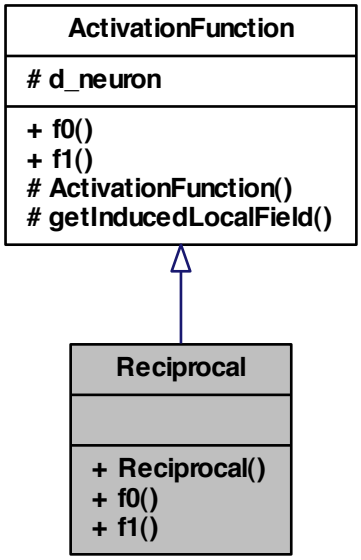
class Reciprocal -

`#include <Reciprocal.h>`

Inheritance diagram for Reciprocal:



Collaboration diagram for Reciprocal:




## Public Member Functions

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

### 5.37.1 Detailed Description

class Reciprocal -

Definition at line 5 of file Reciprocal.h.

### 5.37.2 Constructor & Destructor Documentation

#### 5.37.2.1 Reciprocal::Reciprocal ( NeuronPtr *neuronPtr* )

### 5.37.3 Member Function Documentation

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

Implements ActivationFunction.

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

Implements ActivationFunction.

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

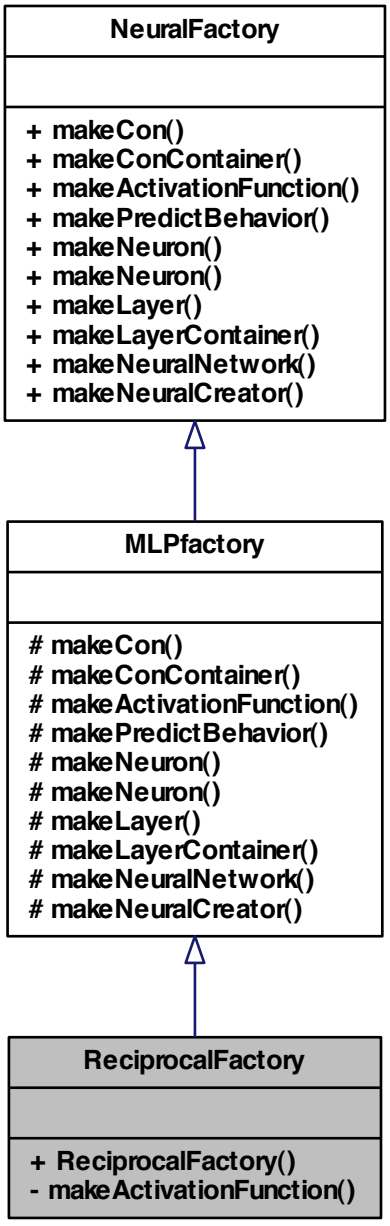
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade

## 5.38 ReciprocalFactory Class Reference

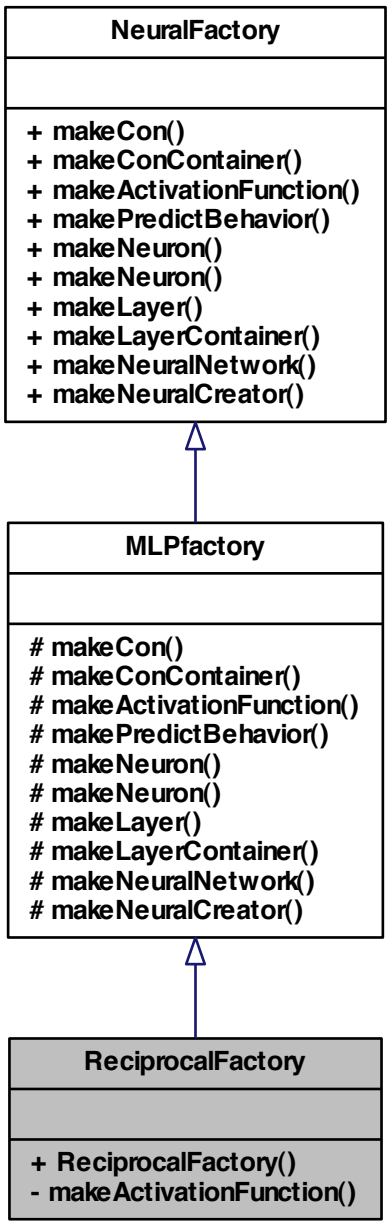
class ReciprocalFactory -

`#include <ReciprocalFactory.h>`

Inheritance diagram for ReciprocalFactory:



Collaboration diagram for ReciprocalFactory:



### Public Member Functions

- ReciprocalFactory ()

### Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

### 5.38.1 Detailed Description

class ReciprocalFactory -

Definition at line 5 of file ReciprocalFactory.h.

### 5.38.2 Constructor & Destructor Documentation

#### 5.38.2.1 ReciprocalFactory::ReciprocalFactory ( )

### 5.38.3 Member Function Documentation

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

Implements MLPfactory.

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

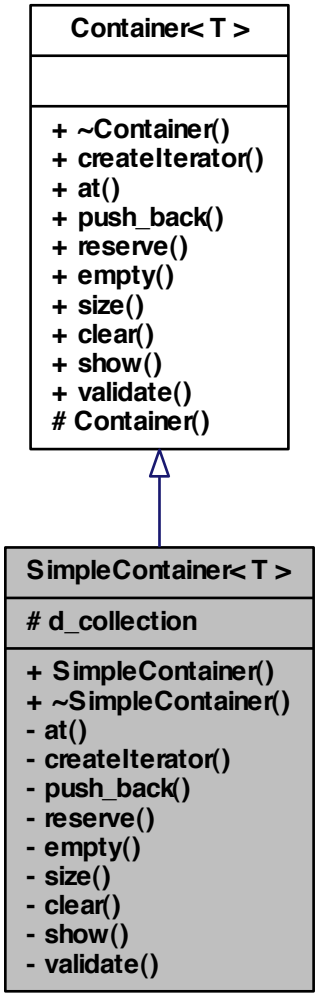
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Reciproca

## 5.39 SimpleContainer< T > Class Template Reference

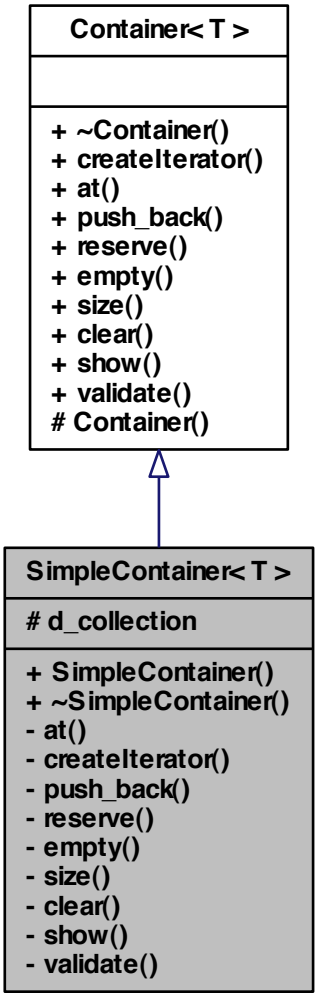
class SimpleContainer -

`#include <SimpleContainer.h>`

Inheritance diagram for SimpleContainer< T >:



Collaboration diagram for SimpleContainer< T >:




## Public Member Functions

- SimpleContainer ()
- ~SimpleContainer ()

### Protected Attributes

- std::vector< T > d_collection

### Private Member Functions

- T at (size_type element)
- boost::shared_ptr< Iterator< T > > createIterator ()
- void push_back (T const &const_reference)
- void reserve (int n)
- bool empty ()
- size_type size ()
- void clear ()
- void show ()
- bool validate ()

### Friends

- class SimpleContainerIterator< T >

## 5.39.1 Detailed Description

**template<typename T>class SimpleContainer< T >**

class SimpleContainer -

Definition at line 6 of file SimpleContainer.h.

## 5.39.2 Constructor & Destructor Documentation

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

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

## 5.39.3 Member Function Documentation

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

Implements Container< T >.

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

Implements Container< T >.

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

Implements Container< T >.

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

Implements Container< T >.

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

Implements Container< T >.

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

Implements Container< T >.

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

Implements Container< T >.

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

Implements Container< T >.

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

Implements Container< T >.

### 5.39.4 Friends And Related Function Documentation

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

Definition at line 12 of file SimpleContainer.h.

### 5.39.5 Member Data Documentation

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

Definition at line 9 of file SimpleContainer.h.

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

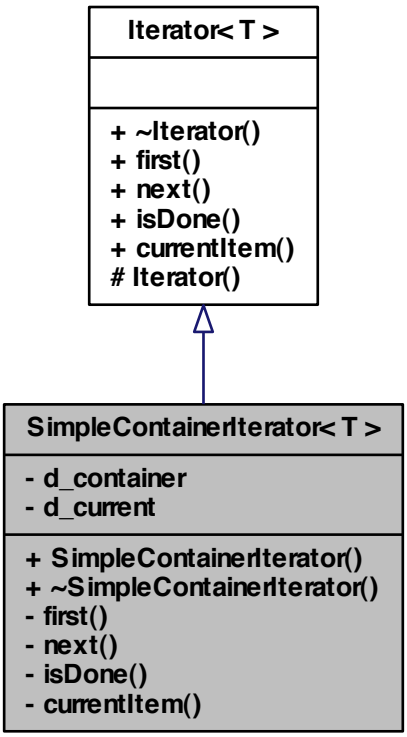
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade

## 5.40 SimpleContainerIterator< T > Class Template Reference

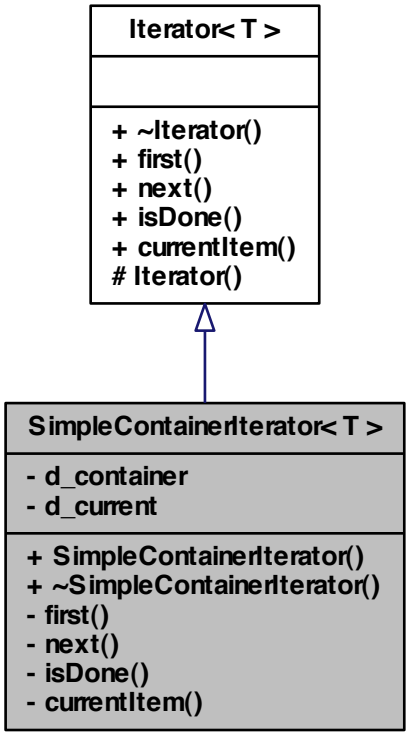
class SimpleContainerIterator -

`#include <SimpleContainerIterator.h>`

Inheritance diagram for SimpleContainerIterator< T >:



Collaboration diagram for SimpleContainerIterator< T >:




## Public Member Functions

- SimpleContainerIterator ()
- ~SimpleContainerIterator ()

## Private Member Functions

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

### Private Attributes

- Container< T > * d_container
- size_type d_current

### Friends

- class SimpleContainer< T >

## 5.40.1 Detailed Description

**template<typename T>class SimpleContainerIterator< T >**

class SimpleContainerIterator -

Definition at line 6 of file SimpleContainerIterator.h.

## 5.40.2 Constructor & Destructor Documentation

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

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

## 5.40.3 Member Function Documentation

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

Implements Iterator< T >.

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

Implements Iterator< T >.

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

Implements Iterator< T >.

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

Implements Iterator< T >.

### 5.40.4 Friends And Related Function Documentation

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

Definition at line 13 of file SimpleContainerIterator.h.

### 5.40.5 Member Data Documentation

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

Definition at line 9 of file SimpleContainerIterator.h.

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

Definition at line 10 of file SimpleContainerIterator.h.

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

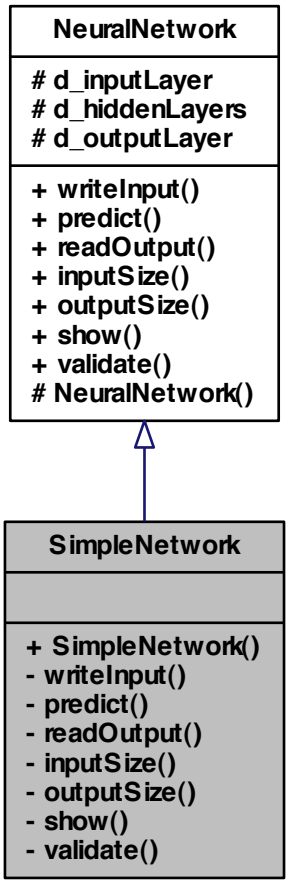
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleCo

## 5.41 SimpleNetwork Class Reference

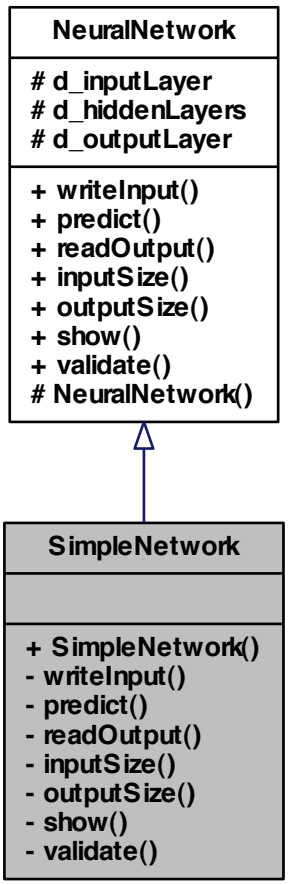
class SimpleNetwork -

`#include <SimpleNetwork.h>`

Inheritance diagram for SimpleNetwork:



Collaboration diagram for SimpleNetwork:




## Public Member Functions

- SimpleNetwork (NeuralFactory &neuralFactory)

## Private Member Functions

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

- size_type outputSize ()
- void show ()
- bool validate ()

### 5.41.1 Detailed Description

class SimpleNetwork -

Definition at line 5 of file SimpleNetwork.h.

### 5.41.2 Constructor & Destructor Documentation

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

Definition at line 16 of file SimpleNetwork.cpp.

```
                                                                    :
  NeuralNetwork(neuralFactory)
{

}
```

### 5.41.3 Member Function Documentation

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

Implements NeuralNetwork.

Definition at line 70 of file SimpleNetwork.cpp.

References NeuralNetwork::d_inputLayer.

Referenced by writeInput().

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

Here is the caller graph for this function:

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

Implements NeuralNetwork.

Definition at line 76 of file SimpleNetwork.cpp.

References NeuralNetwork::d_outputLayer.

Referenced by readOutput().

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

Here is the caller graph for this function:


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

Implements NeuralNetwork.

Definition at line 23 of file SimpleNetwork.cpp.

References NeuralNetwork::d_hiddenLayers, and NeuralNetwork::d_outputLayer.

```
{

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

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

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

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

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

Implements NeuralNetwork.

Definition at line 60 of file SimpleNetwork.cpp.

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

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

Here is the call graph for this function:


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

Implements NeuralNetwork.

Definition at line 82 of file SimpleNetwork.cpp.

References NeuralNetwork::d_hiddenLayers, NeuralNetwork::d_inputLayer, and NeuralNetwork::d_-outputLayer.

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

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

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

}
```

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

Implements NeuralNetwork.

Definition at line 103 of file SimpleNetwork.cpp.

References NeuralNetwork::d_hiddenLayers, NeuralNetwork::d_inputLayer, and NeuralNetwork::d_outputLayer.

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

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

Implements NeuralNetwork.

Definition at line 50 of file SimpleNetwork.cpp.

References NeuralNetwork::d_inputLayer, inputSize(), and size_type.

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

Here is the call graph for this function:


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

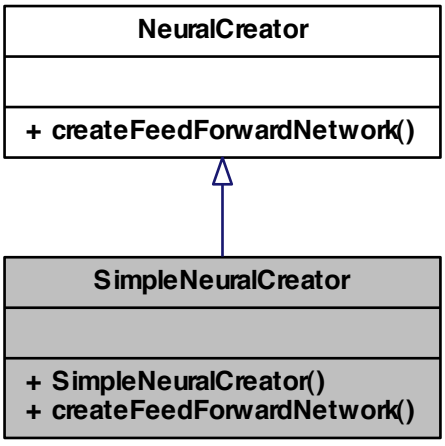
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNetw

## 5.42 SimpleNeuralCreator Class Reference

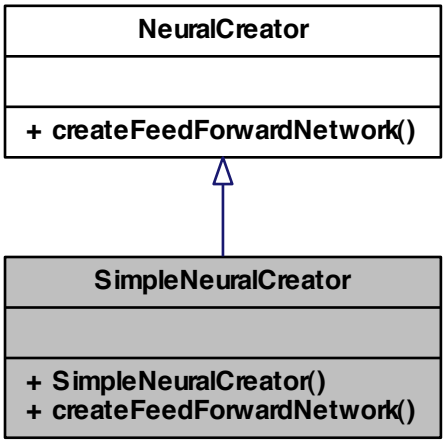
class SimpleNeuralCreator -

`#include <SimpleNeuralCreator.h>`

Inheritance diagram for SimpleNeuralCreator:



Collaboration diagram for SimpleNeuralCreator:




### Public Member Functions

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

## 5.42.1 Detailed Description

class SimpleNeuralCreator -

Definition at line 5 of file SimpleNeuralCreator.h.

## 5.42.2 Constructor & Destructor Documentation

### 5.42.2.1 SimpleNeuralCreator::SimpleNeuralCreator ( )

Definition at line 19 of file SimpleNeuralCreator.cpp.

```
{
}
```

### 5.42.3 Member Function Documentation

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

Implements NeuralCreator.

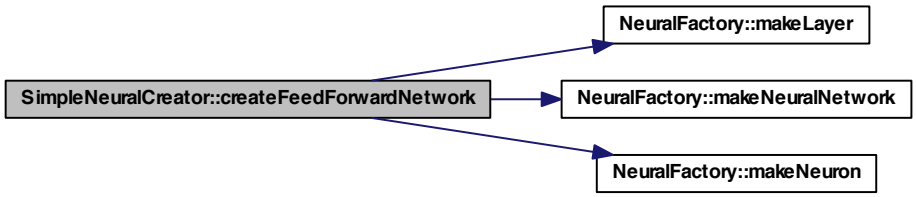
Definition at line 24 of file SimpleNeuralCreator.cpp.

References NeuralFactory::makeLayer(), NeuralFactory::makeNeuralNetwork(), and NeuralFactory::makeNeuron().

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

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

  Handler neuronId = 1;

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

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


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

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


  // Hidden layers

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

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

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

        }
    }


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

  return neuralNetworkPtr;
}
```

Here is the call graph for this function:




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

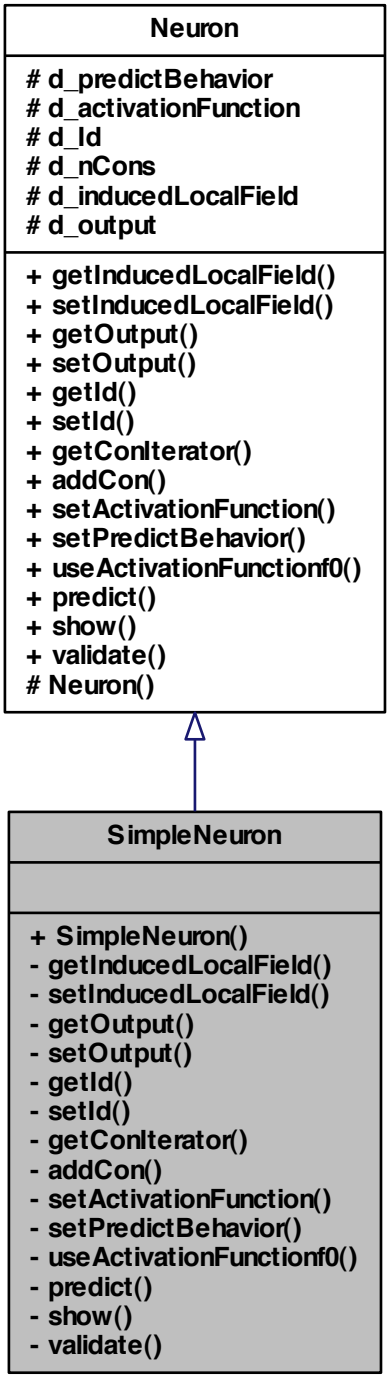
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNe
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuralCreator.cp

## 5.43 SimpleNeuron Class Reference

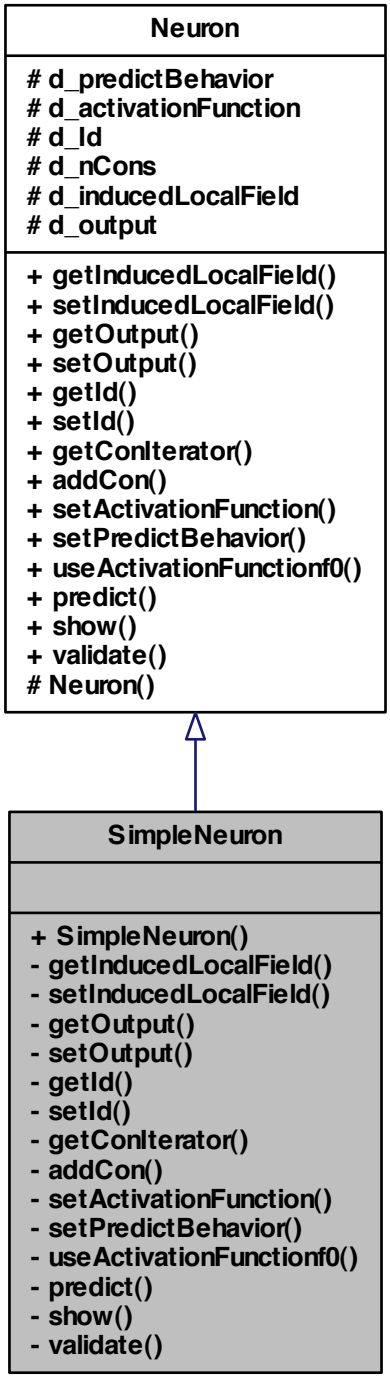
class SimpleNeuron -

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

Inheritance diagram for SimpleNeuron:



Collaboration diagram for SimpleNeuron:



### Public Member Functions

- SimpleNeuron (NeuralFactory &neuralFactory)

### Private Member Functions

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

## 5.43.1 Detailed Description

class SimpleNeuron -

Definition at line 5 of file SimpleNeuron.h.

## 5.43.2 Constructor & Destructor Documentation

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

Definition at line 18 of file SimpleNeuron.cpp.

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

## 5.43.3 Member Function Documentation

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

Implements Neuron.

Definition at line 66 of file SimpleNeuron.cpp.

References Neuron::d_nCons.

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

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

Implements Neuron.

Definition at line 60 of file SimpleNeuron.cpp.

References Neuron::d_nCons.

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

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

Implements Neuron.

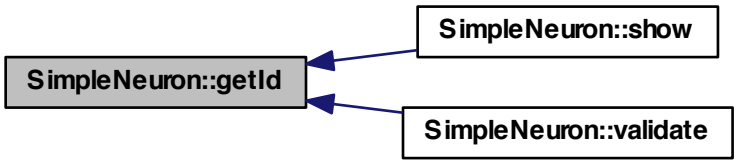
Definition at line 48 of file SimpleNeuron.cpp.

References Neuron::d_Id.

Referenced by show(), and validate().

```
{
  return d_Id;
}
```

Here is the caller graph for this function:




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

Implements Neuron.

Definition at line 24 of file SimpleNeuron.cpp.

References Neuron::d_inducedLocalField.

```
{
  return d_inducedLocalField;
}
```

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

Implements Neuron.

Definition at line 36 of file SimpleNeuron.cpp.

References Neuron::d_output.

```
{
  return d_output;
}
```

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

Implements Neuron.

Definition at line 90 of file SimpleNeuron.cpp.

References Neuron::d_predictBehavior.

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

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

Implements Neuron.

Definition at line 72 of file SimpleNeuron.cpp.

References Neuron::d_activationFunction.

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

#### 5.43.3.8 void SimpleNeuron::setId ( Handler *id* ) [private, virtual]

Implements Neuron.

Definition at line 54 of file SimpleNeuron.cpp.

References Neuron::d_Id.

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

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

Implements Neuron.

Definition at line 30 of file SimpleNeuron.cpp.

References Neuron::d_inducedLocalField.

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

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

Implements Neuron.

Definition at line 42 of file SimpleNeuron.cpp.

References Neuron::d_output.

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

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

Implements Neuron.

Definition at line 78 of file SimpleNeuron.cpp.

References Neuron::d_predictBehavior.

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

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

Implements Neuron.

Definition at line 96 of file SimpleNeuron.cpp.

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

```
{
  if (d_nCons->size() == 0)
    {

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

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

Here is the call graph for this function:


#### 5.43.3.13 double SimpleNeuron::useActivationFunctionf0 ( ) [private, virtual]

Implements Neuron.

Definition at line 84 of file SimpleNeuron.cpp.

References Neuron::d_activationFunction.

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

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

Implements Neuron.

Definition at line 138 of file SimpleNeuron.cpp.

References getId().

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

Here is the call graph for this function:


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

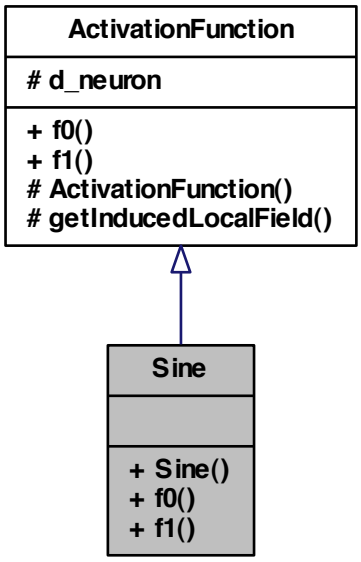
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeu

## 5.44 Sine Class Reference

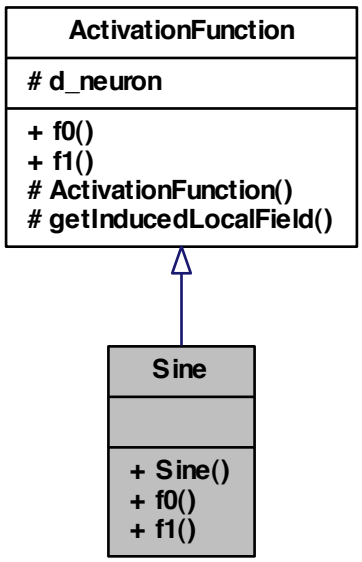
class Sine -

`#include <Sine.h>`

Inheritance diagram for Sine:



Collaboration diagram for Sine:




## Public Member Functions

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

### 5.44.1 Detailed Description

class Sine -

Definition at line 5 of file Sine.h.

### 5.44.2 Constructor & Destructor Documentation

#### 5.44.2.1 Sine::Sine ( NeuronPtr *neuronPtr* )

### 5.44.3 Member Function Documentation

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

Implements ActivationFunction.

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

Implements ActivationFunction.

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

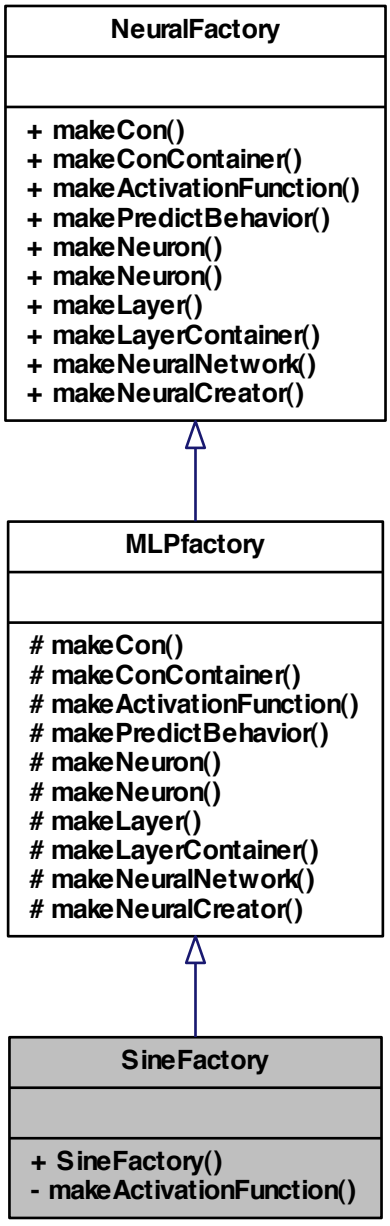
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Sine.h

## 5.45 SineFactory Class Reference

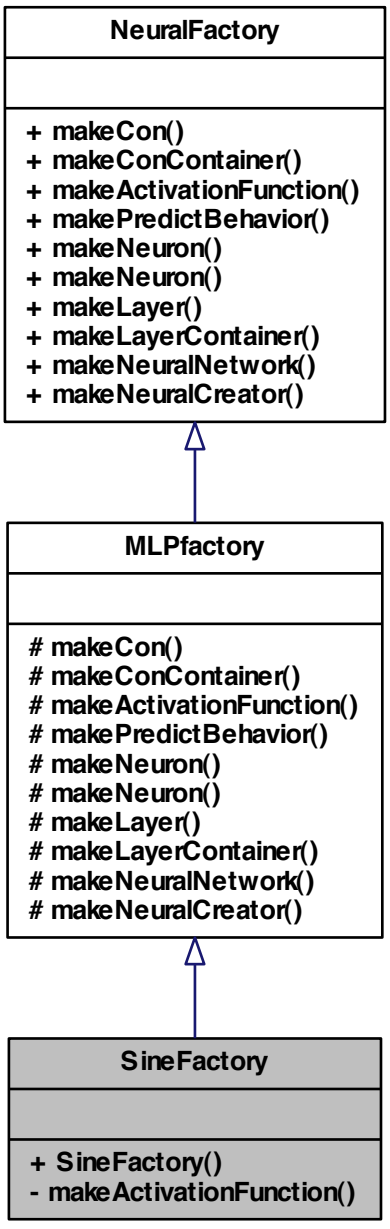
class SineFactory -

`#include <SineFactory.h>`

Inheritance diagram for SineFactory:



Collaboration diagram for SineFactory:



### Public Member Functions

- SineFactory ()

### Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

#### 5.45.1 Detailed Description

class SineFactory -

Definition at line 5 of file SineFactory.h.

#### 5.45.2 Constructor & Destructor Documentation

##### 5.45.2.1 SineFactory::SineFactory ( )

#### 5.45.3 Member Function Documentation

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

Implements MLPfactory.

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

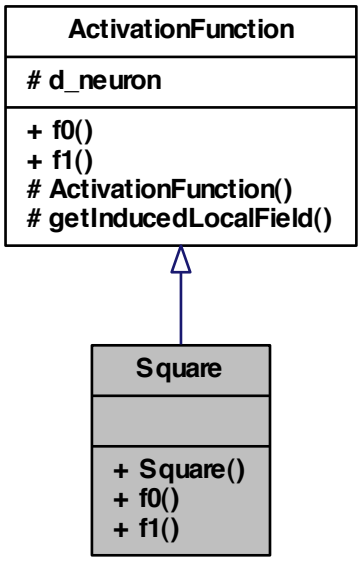
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade

## 5.46 Square Class Reference

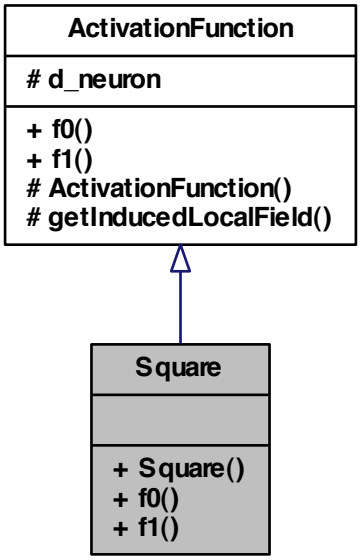
class Square -

`#include <Square.h>`

Inheritance diagram for Square:



Collaboration diagram for Square:




## Public Member Functions

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

### 5.46.1 Detailed Description

class Square -

Definition at line 5 of file Square.h.

### 5.46.2 Constructor & Destructor Documentation

#### 5.46.2.1 Square::Square ( NeuronPtr *neuronPtr* )

### 5.46.3 Member Function Documentation

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

Implements ActivationFunction.

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

Implements ActivationFunction.

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

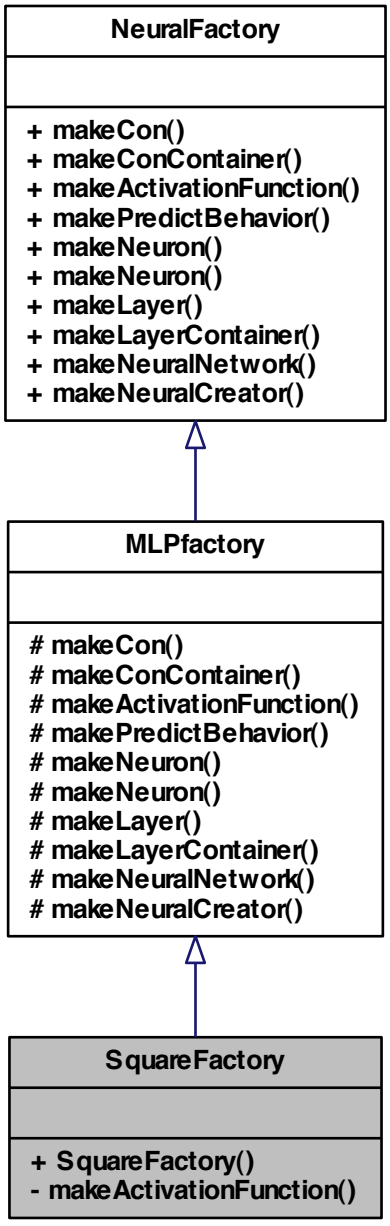
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Square.h

## 5.47 SquareFactory Class Reference

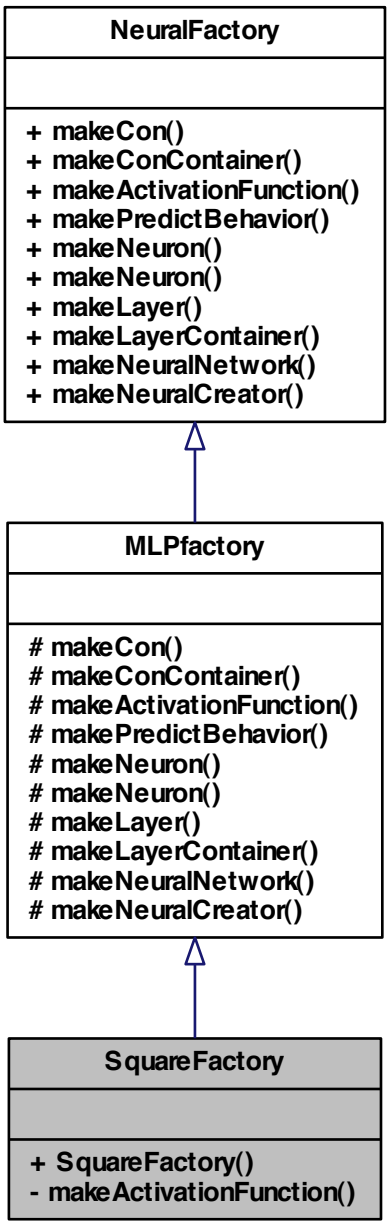
class SquareFactory -

`#include <SquareFactory.h>`

Inheritance diagram for SquareFactory:



Collaboration diagram for SquareFactory:



### Public Member Functions

- SquareFactory ()

### Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

#### 5.47.1 Detailed Description

class SquareFactory -

Definition at line 5 of file SquareFactory.h.

#### 5.47.2 Constructor & Destructor Documentation

##### 5.47.2.1 SquareFactory::SquareFactory ( )

#### 5.47.3 Member Function Documentation

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

Implements MLPfactory.

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

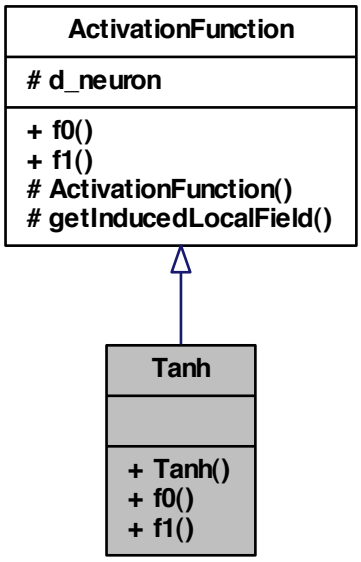
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade

## 5.48 Tanh Class Reference

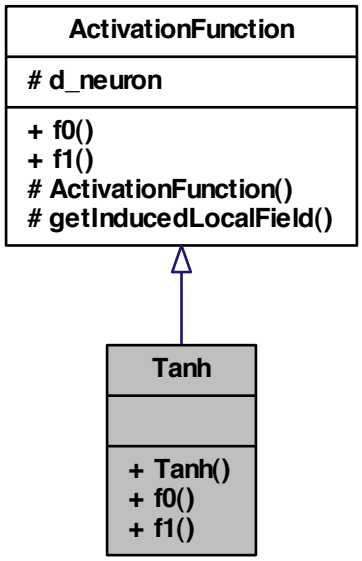
class Tanh -

`#include <Tanh.h>`

Inheritance diagram for Tanh:



Collaboration diagram for Tanh:




## Public Member Functions

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

### 5.48.1 Detailed Description

class Tanh -

Definition at line 5 of file Tanh.h.

### 5.48.2 Constructor & Destructor Documentation

#### 5.48.2.1 Tanh::Tanh ( NeuronPtr *neuronPtr* )

Definition at line 15 of file Tanh.cpp.

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

```
}
```

### 5.48.3 Member Function Documentation

#### 5.48.3.1 double Tanh::f0 ( ) [virtual]

Implements ActivationFunction.

Definition at line 19 of file Tanh.cpp.

References ActivationFunction::getInducedLocalField().

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

Here is the call graph for this function:


#### 5.48.3.2 double Tanh::f1 ( ) [virtual]

Implements ActivationFunction.

Definition at line 24 of file Tanh.cpp.

References ActivationFunction::getInducedLocalField().

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

Here is the call graph for this function:


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

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

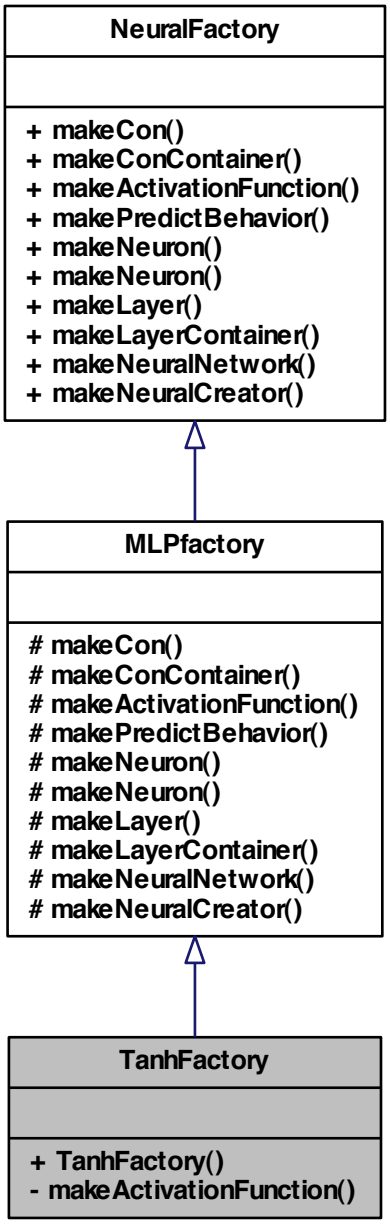
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Tanh.cpp

## 5.49 TanhFactory Class Reference

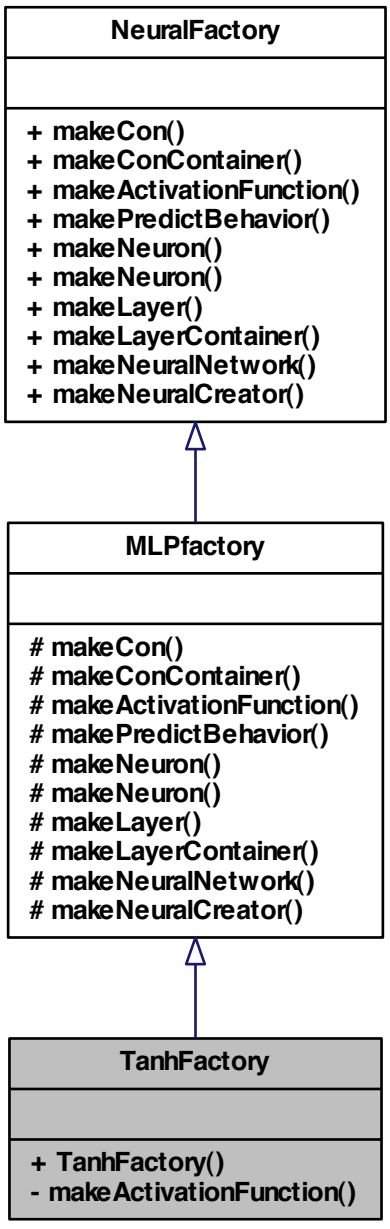
class TanhFactory -

`#include <TanhFactory.h>`

Inheritance diagram for TanhFactory:



Collaboration diagram for TanhFactory:



### Public Member Functions

- TanhFactory ()

### Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

### 5.49.1 Detailed Description

class TanhFactory -

Definition at line 5 of file TanhFactory.h.

### 5.49.2 Constructor & Destructor Documentation

#### 5.49.2.1 TanhFactory::TanhFactory ( )

Definition at line 17 of file TanhFactory.cpp.

```
{
}
```

### 5.49.3 Member Function Documentation

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

Implements MLPfactory.

Definition at line 22 of file TanhFactory.cpp.

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

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

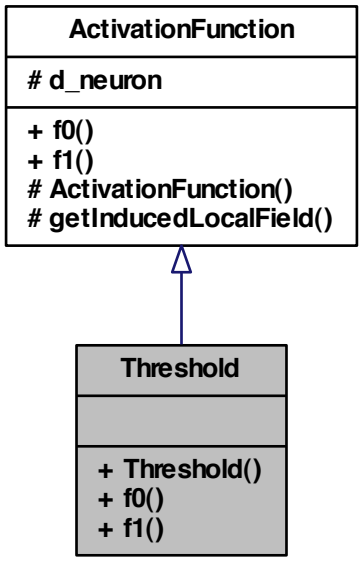
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/TanhFacto
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/TanhFactory.cpp

## 5.50 Threshold Class Reference

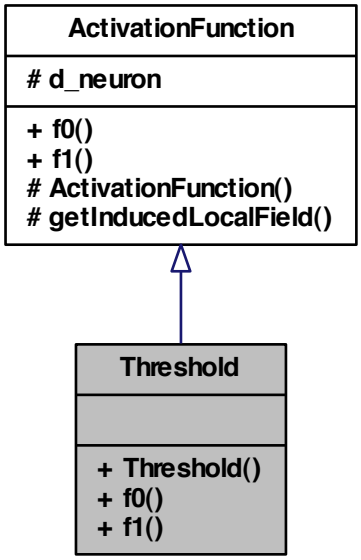
class Threshold -

`#include <Threshold.h>`

Inheritance diagram for Threshold:



Collaboration diagram for Threshold:




## Public Member Functions

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

### 5.50.1 Detailed Description

class Threshold -

Definition at line 5 of file Threshold.h.

### 5.50.2 Constructor & Destructor Documentation

#### 5.50.2.1 Threshold::Threshold ( NeuronPtr *neuronPtr* )

### 5.50.3 Member Function Documentation

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

Implements ActivationFunction.

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

Implements ActivationFunction.

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

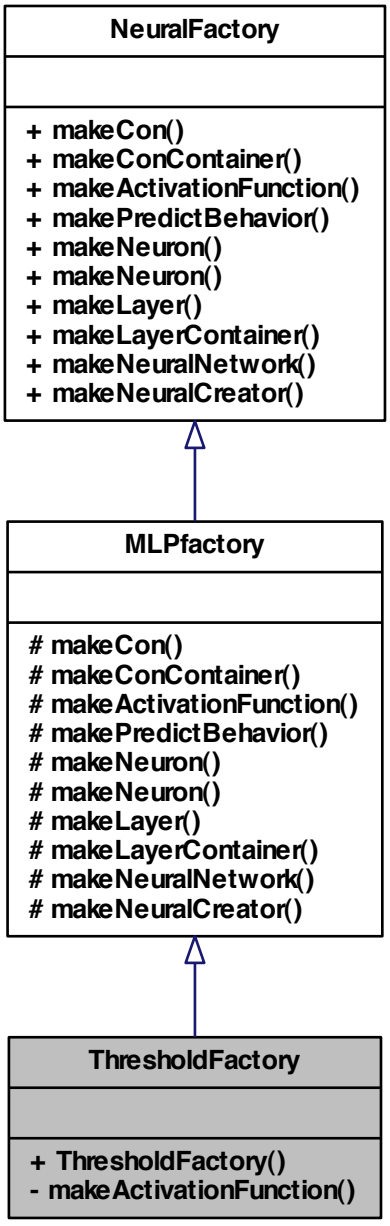
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeade

## 5.51 ThresholdFactory Class Reference

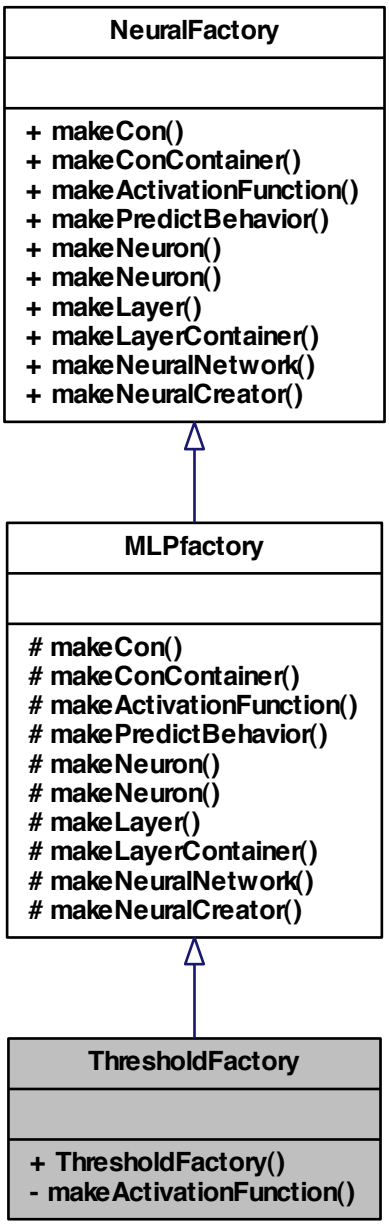
class ThresholdFactory -

`#include <ThresholdFactory.h>`

Inheritance diagram for ThresholdFactory:



Collaboration diagram for ThresholdFactory:



### Public Member Functions

- ThresholdFactory ()

### Private Member Functions

- ActivationFunctionPtr makeActivationFunction (NeuronPtr neuronPtr)

#### 5.51.1 Detailed Description

class ThresholdFactory -

Definition at line 5 of file ThresholdFactory.h.

#### 5.51.2 Constructor & Destructor Documentation

##### 5.51.2.1 ThresholdFactory::ThresholdFactory ( )

#### 5.51.3 Member Function Documentation

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

Implements MLPfactory.

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

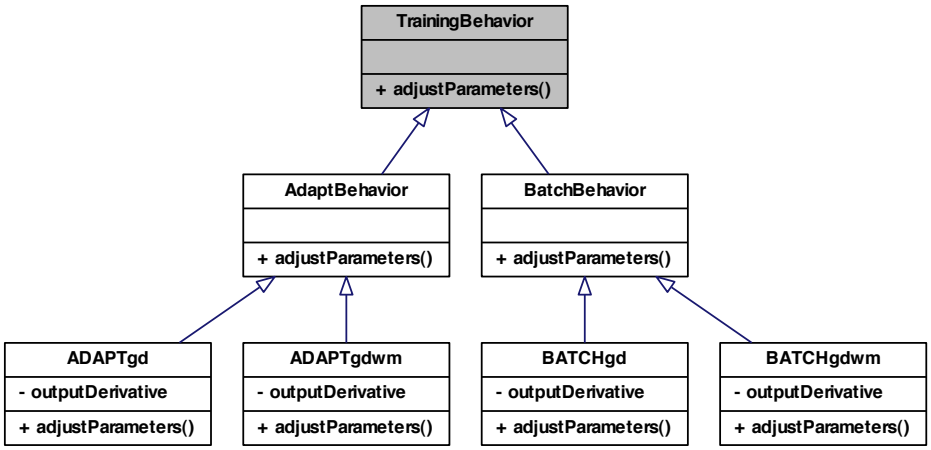
- /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Threshold

## 5.52 TrainingBehavior Class Reference

class TrainingBehavior -

`#include <TrainingBehavior.h>`

Inheritance diagram for TrainingBehavior:




## Public Member Functions

- void adjustParameters ()

### 5.52.1 Detailed Description

class TrainingBehavior -

Definition at line 4 of file TrainingBehavior.h.

### 5.52.2 Member Function Documentation

#### 5.52.2.1 void TrainingBehavior::adjustParameters ( )

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

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

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

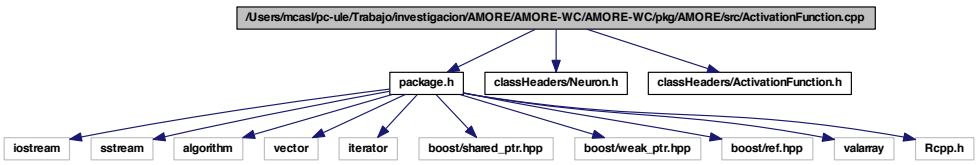
# Chapter 6

# File Documentation

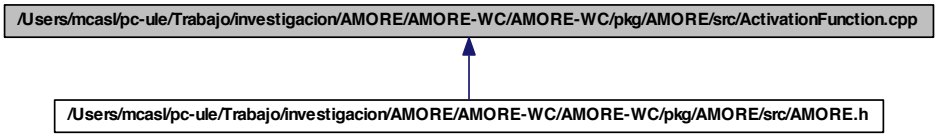
## 6.1 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/ActivationFunction.cpp File Reference

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

Include dependency graph for ActivationFunction.cpp:




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



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

```
#include <iostream>
#include <sstream>
#include <algorithm>
#include <vector>
#include <iterator>
#include <boost/shared_ptr.hpp>
#include <boost/weak_ptr.hpp>
#include <boost/ref.hpp>
#include <valarray>
#include <Rcpp.h>
#include "classHeaders/Connection.h"
#include "classHeaders/ActivationFunction.h"
#include "classHeaders/Tanh.h"
#include "classHeaders/Identity.h"
#include "classHeaders/PredictBehavior.h"
#include "classHeaders/MLPBehavior.h"
#include "classHeaders/Neuron.h"
#include "classHeaders/SimpleNeuron.h"
#include "classHeaders/NeuralFactory.h"
#include "classHeaders/MLPfactory.h"
#include "classHeaders/TanhFactory.h"
#include "classHeaders/IdentityFactory.h"
#include "classHeaders/NeuralNetwork.h"
#include "classHeaders/SimpleNetwork.h"
#include "classHeaders/NeuralCreator.h"
#include "classHeaders/SimpleNeuralCreator.h"
#include "classHeaders/NetworkRinterface.h"
#include "classHeaders/Container.h"
#include "classHeaders/SimpleContainer.h"
#include "classHeaders/Iterator.h"
#include "classHeaders/SimpleContainerIterator.h"
```

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

## Defines

- #define size_type unsigned int

## Typedefs

- typedef int Handler
- typedef boost::reference_wrapper< PredictBehavior > ActivationFunctionRef
- typedef boost::reference_wrapper< PredictBehavior > PredictBehaviorRef
- typedef boost::reference_wrapper< TrainingBehavior > TrainingBehaviorRef
- typedef boost::reference_wrapper< Neuron > NeuronRef
- typedef boost::shared_ptr< ActivationFunction > ActivationFunctionPtr
- typedef boost::shared_ptr< PredictBehavior > PredictBehaviorPtr
- typedef boost::shared_ptr< Neuron > NeuronPtr
- typedef boost::shared_ptr< Con > ConPtr
- typedef boost::shared_ptr< NeuralNetwork > NeuralNetworkPtr
- typedef boost::shared_ptr< Iterator< NeuronPtr > > NeuronIteratorPtr

- typedef boost::shared_ptr< Iterator< ConPtr > > ConIteratorPtr
- typedef boost::shared_ptr< Container< NeuronPtr > > LayerPtr
- typedef boost::shared_ptr< Container< LayerPtr > > LayerContainerPtr
- typedef boost::shared_ptr< Container< ConPtr > > ConContainerPtr
- typedef boost::shared_ptr< NeuralFactory > NeuralFactoryPtr
- typedef boost::shared_ptr< NeuralCreator > NeuralCreatorPtr
- typedef boost::weak_ptr< Neuron > NeuronWeakPtr

### 6.2.1 Define Documentation

#### 6.2.1.1 #define size_type unsigned int

Definition at line 75 of file AMORE.h.

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

### 6.2.2 Typedef Documentation

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

Definition at line 87 of file AMORE.h.

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

Definition at line 81 of file AMORE.h.

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

Definition at line 99 of file AMORE.h.

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

Definition at line 95 of file AMORE.h.

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

Definition at line 90 of file AMORE.h.

#### 6.2.2.6 typedef int Handler

Definition at line 78 of file AMORE.h.

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

Definition at line 98 of file AMORE.h.

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

Definition at line 97 of file AMORE.h.

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

Definition at line 102 of file AMORE.h.

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

Definition at line 101 of file AMORE.h.

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

Definition at line 91 of file AMORE.h.

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

Definition at line 94 of file AMORE.h.

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

Definition at line 89 of file AMORE.h.

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

Definition at line 84 of file AMORE.h.

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

Definition at line 104 of file AMORE.h.

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

Definition at line 88 of file AMORE.h.

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

Definition at line 82 of file AMORE.h.

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

Definition at line 83 of file AMORE.h.

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

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

### Classes

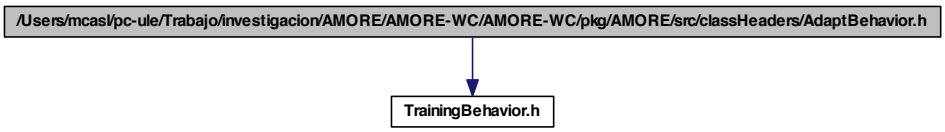
- class ActivationFunction

  *class ActivationFunction -*

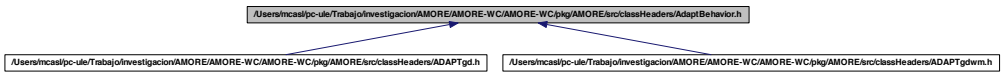
## 6.4 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/AdaptBehavior.h File Reference

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

Include dependency graph for AdaptBehavior.h:



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




### Classes

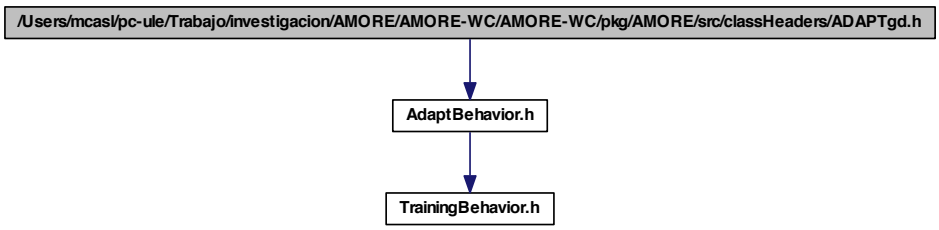
- class AdaptBehavior

  *class AdaptBehavior -*

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

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

Include dependency graph for ADAPTgd.h:




### Classes

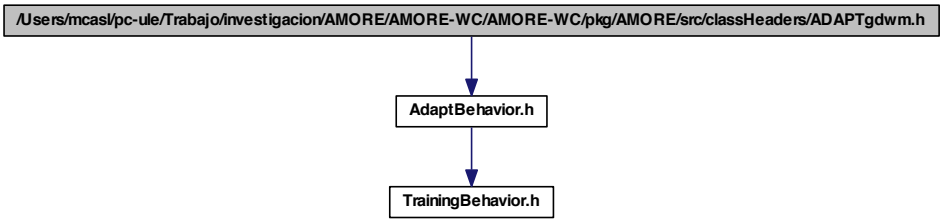
- class ADAPTgd

  *class ADAPTgd -*

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

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

Include dependency graph for ADAPTgdwm.h:




### Classes

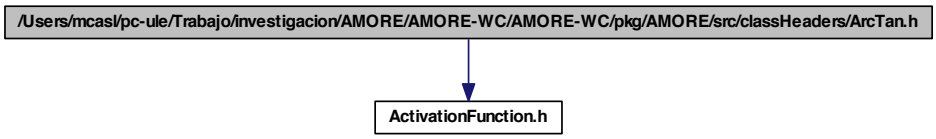
- class ADAPTgdwm

  *class ADAPTgdwm -*

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

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

Include dependency graph for ArcTan.h:




### Classes

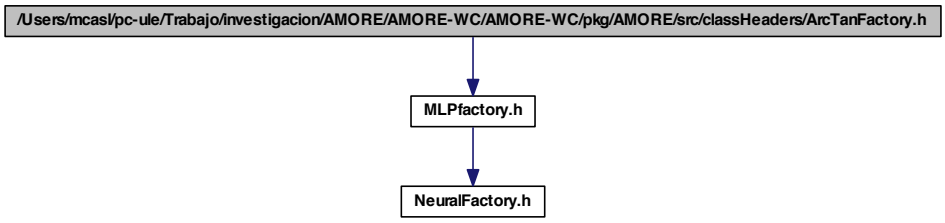
- class ArcTan

  *class ArcTan -*

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

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

Include dependency graph for ArcTanFactory.h:




### Classes

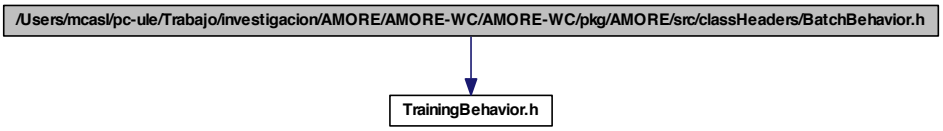
- class ArcTanFactory

  *class ArcTanFactory -*

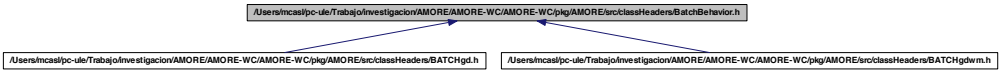
## 6.9 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/BatchBehavior.h File Reference

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

Include dependency graph for BatchBehavior.h:



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




### Classes

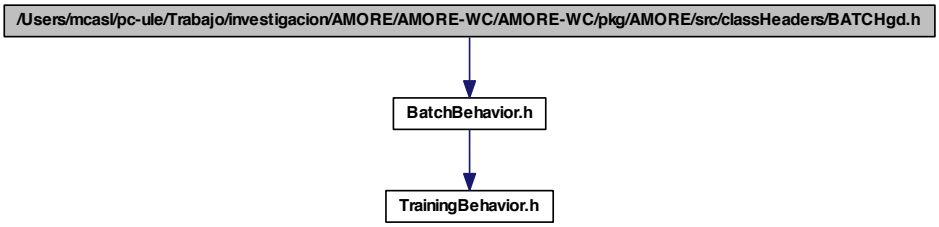
- class BatchBehavior

  *class BatchBehavior -*

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

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

Include dependency graph for BATCHgd.h:




### Classes

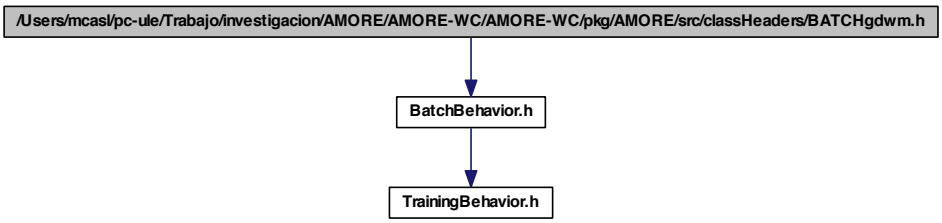
- class BATCHgd

  *class BATCHgd -*

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

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

Include dependency graph for BATCHgdwm.h:



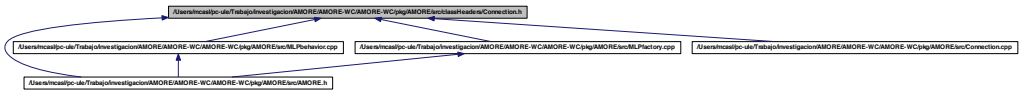

### Classes

- class BATCHgdwm

  *class BATCHgdwm -*

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

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




### Classes

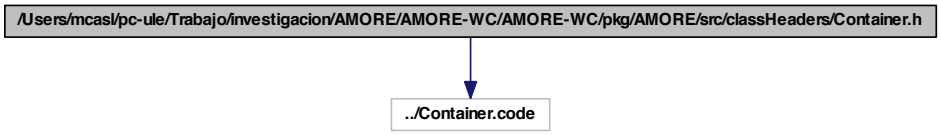
- class Con

  *class Con -*

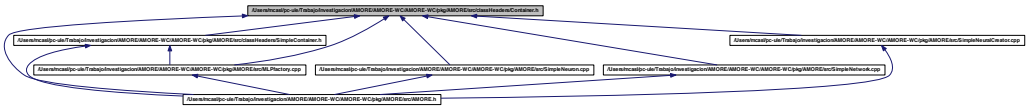
## 6.13 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Container.h File Reference

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

Include dependency graph for Container.h:




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



## Classes

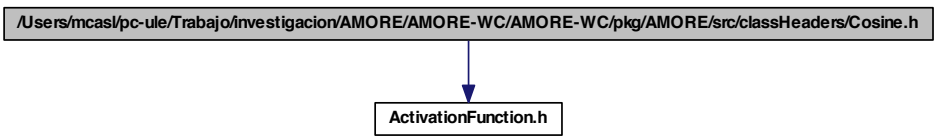
- class Container< T >

  *class Container -*

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

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

Include dependency graph for Cosine.h:




## Classes

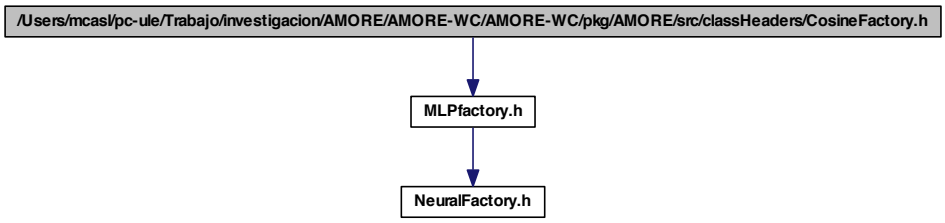
- class Cosine

*class Cosine -*

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

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

Include dependency graph for CosineFactory.h:




### Classes

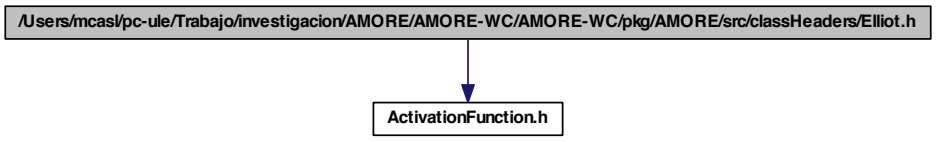
- class CosineFactory

  *class CosineFactory -*

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

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

Include dependency graph for Elliot.h:



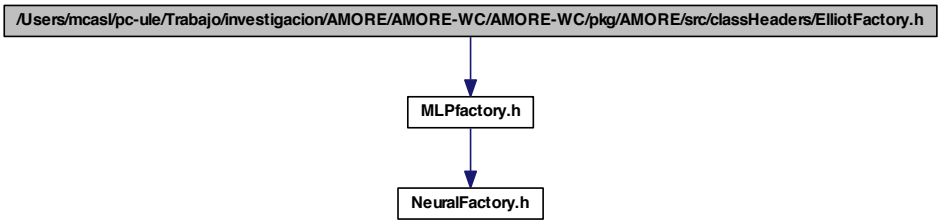
### Classes

- class Elliot
  *class Elliot -*

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

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

Include dependency graph for ElliotFactory.h:



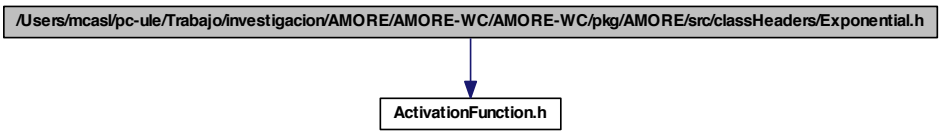

### Classes

- class ElliotFactory
  *class ElliotFactory -*

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

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

Include dependency graph for Exponential.h:



### Classes

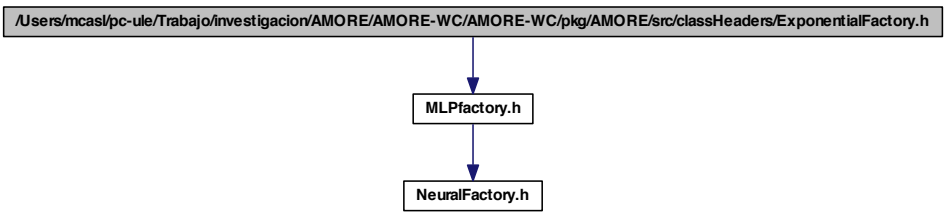
- class Exponential

  *class Exponential -*

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

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

Include dependency graph for ExponentialFactory.h:




### Classes

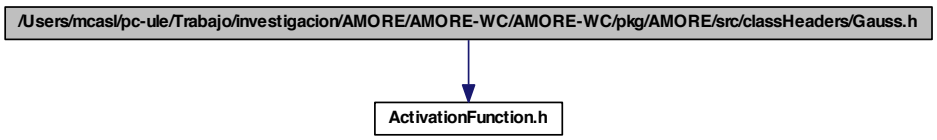
- class ExponentialFactory

  *class ExponentialFactory -*

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

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

Include dependency graph for Gauss.h:




### Classes

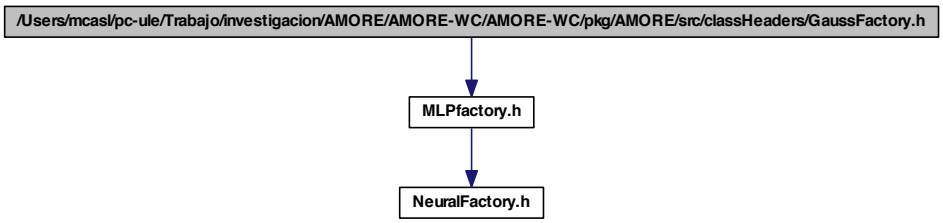
- class Gauss

  *class Gauss -*

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

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

Include dependency graph for GaussFactory.h:




### Classes

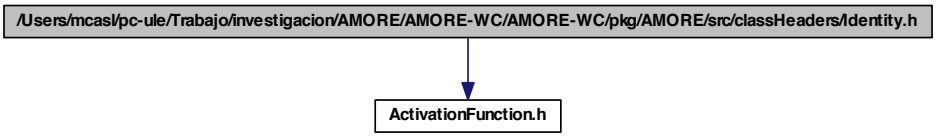
- class GaussFactory

  *class GaussFactory -*

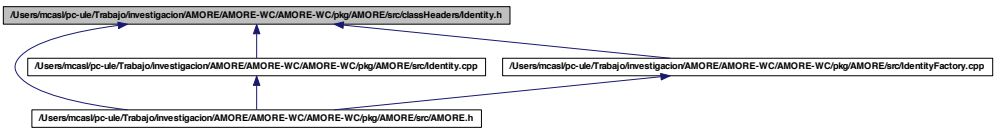
## 6.22 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Identity.h File Reference

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

Include dependency graph for Identity.h:




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




### Classes

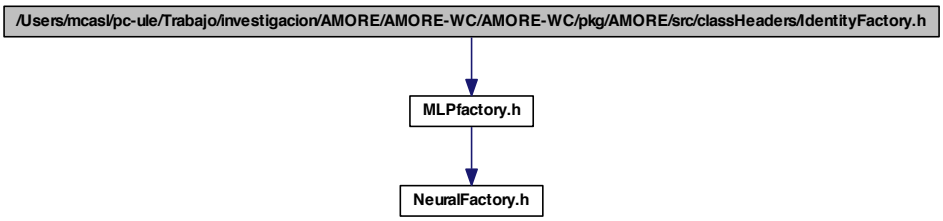
- class Identity

  *class Identity* -

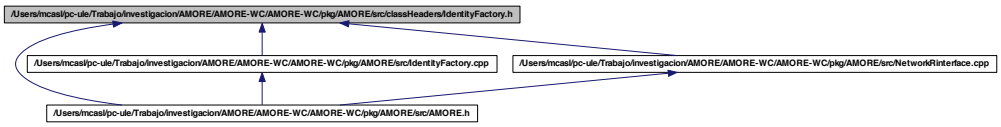
## 6.23 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/IdentityFactory.h File Reference

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

Include dependency graph for IdentityFactory.h:




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




### Classes

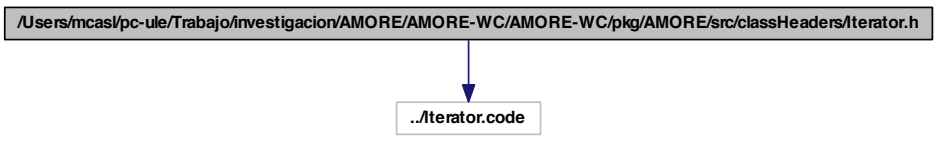
- class IdentityFactory

  *class IdentityFactory -*

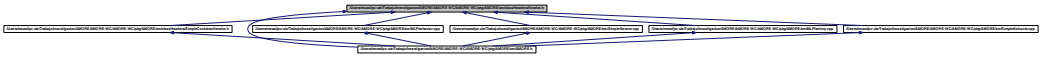
## 6.24 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Iterator.h File Reference

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

Include dependency graph for Iterator.h:



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



### Classes

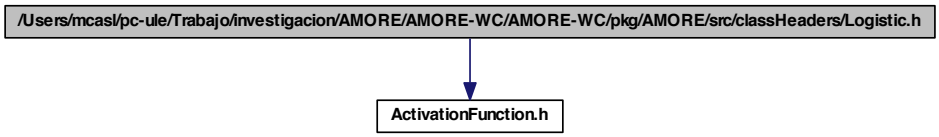
- class Iterator< T >

  *class Iterator -*

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

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

Include dependency graph for Logistic.h:




### Classes

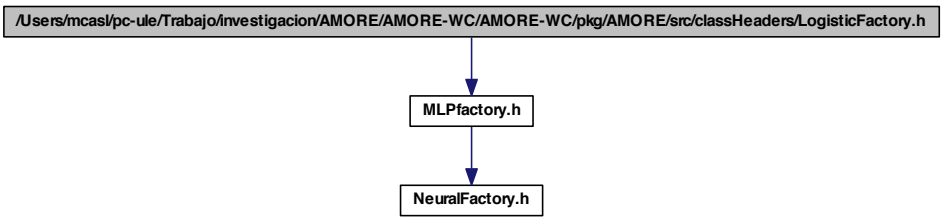
- class Logistic

  *class Logistic -*

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

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

Include dependency graph for LogisticFactory.h:




### Classes

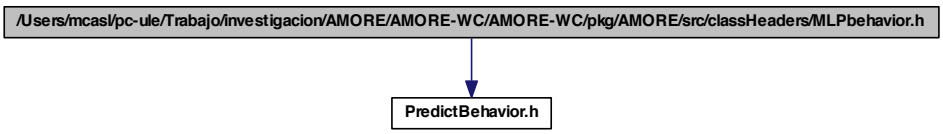
- class LogisticFactory

  *class LogisticFactory -*

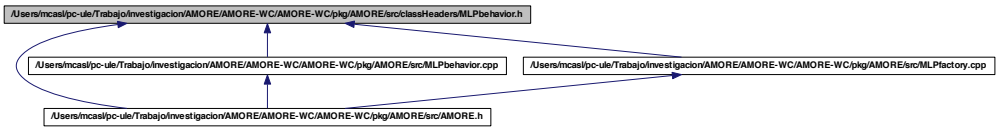
## 6.27 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/MLPbehavior.h File Reference

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

Include dependency graph for MLPbehavior.h:




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



Classes

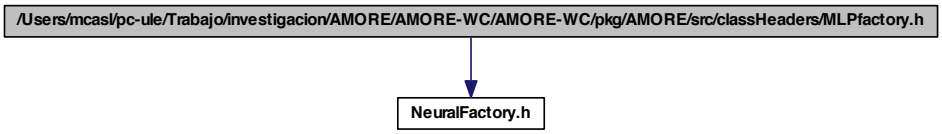
- class MLPbehavior

  *class MLPbehavior -*

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

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

Include dependency graph for MLPfactory.h:




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

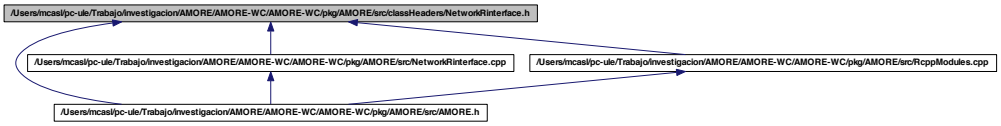
### Classes

- class MLPfactory

  *class MLPfactory -*

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

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



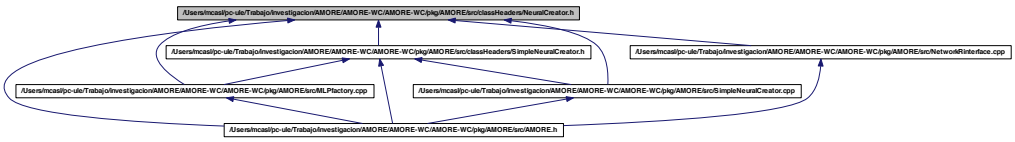

### Classes

- class NetworkRinterface

  *class NetworkRinterface -*

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

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




### Classes

- class NeuralCreator

  *class NeuralCreator -*

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

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

### Classes

- class NeuralFactory

  *class NeuralFactory -*

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

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

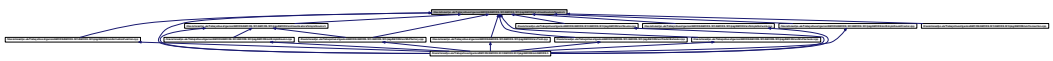
### Classes

- class NeuralNetwork

  *class NeuralNetwork -*

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

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



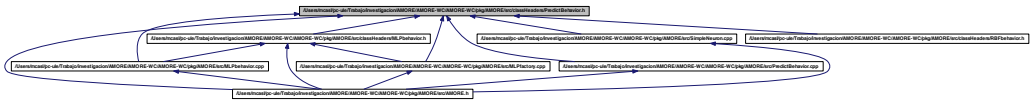
### Classes

- class Neuron

  *class Neuron -*

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

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



### Classes

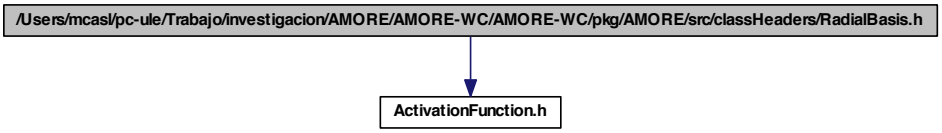
- class PredictBehavior

  *class PredictBehavior -*

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

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

Include dependency graph for RadialBasis.h:




### Classes

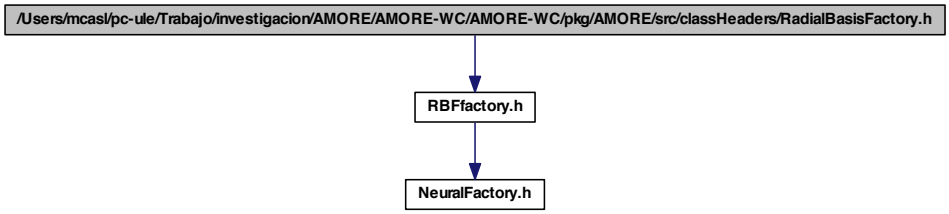
- class RadialBasis

  *class RadialBasis -*

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

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

Include dependency graph for RadialBasisFactory.h:




### Classes

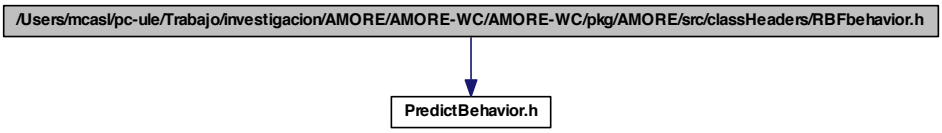
- class RadialBasisFactory

  *class RadialBasisFactory -*

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

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

Include dependency graph for RBFbehavior.h:




### Classes

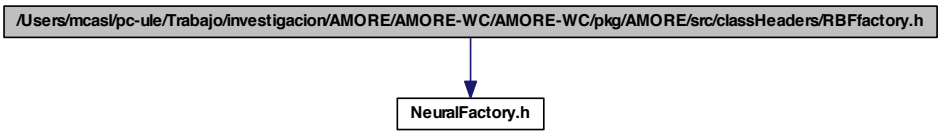
- class RBFbehavior

*class RBFbehavior -*

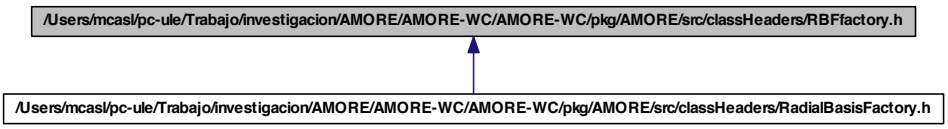
## 6.38 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/RBFfactory.h File Reference

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

Include dependency graph for RBFfactory.h:




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




### Classes

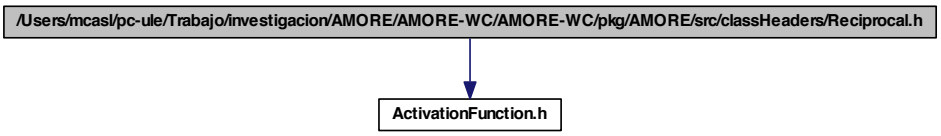
- class RBFfactory

  *class RBFfactory -*

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

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

Include dependency graph for Reciprocal.h:




### Classes

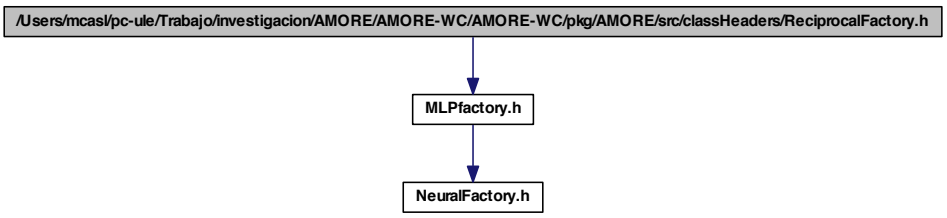
- class Reciprocal

  *class Reciprocal -*

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

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

Include dependency graph for ReciprocalFactory.h:




### Classes

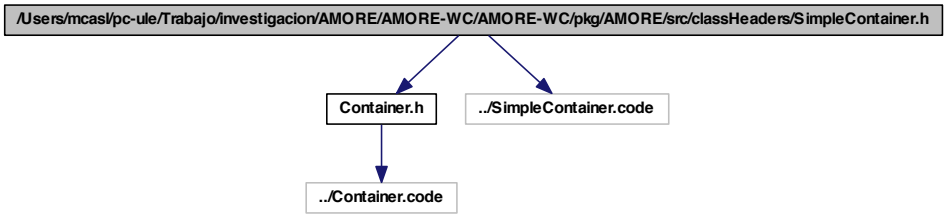
- class ReciprocalFactory

  *class ReciprocalFactory -*

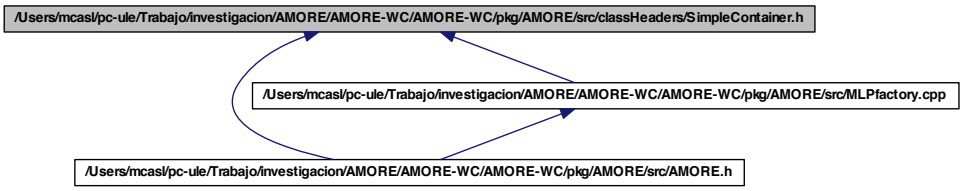
## 6.41 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleContainer.h File Reference

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

Include dependency graph for SimpleContainer.h:




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




### Classes

- class SimpleContainer< T >

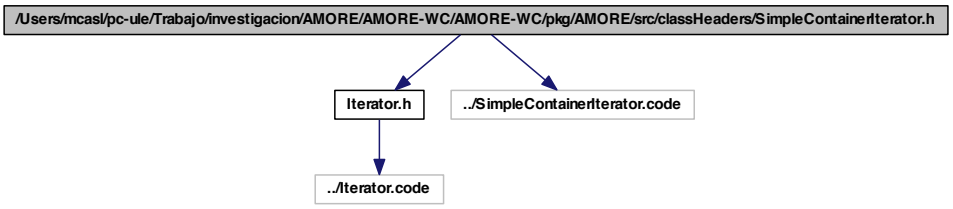
  *class SimpleContainer -*

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

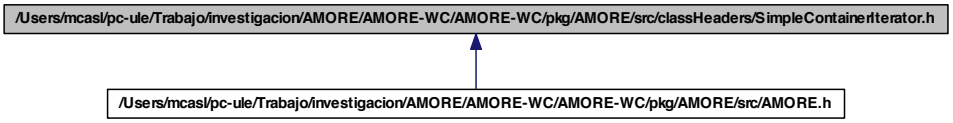
```
#include "Iterator.h"
```

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

Include dependency graph for SimpleContainerIterator.h:




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




### Classes

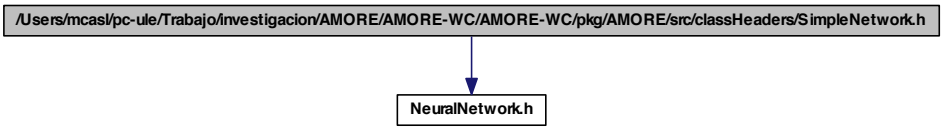
- class SimpleContainerIterator< T >

  *class SimpleContainerIterator -*

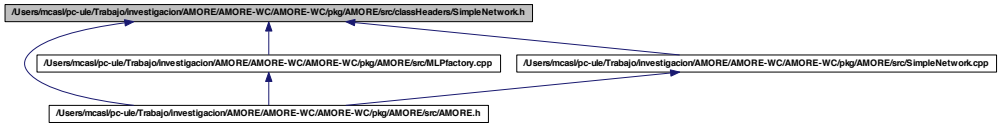
## 6.43 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNetwork.h File Reference

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

Include dependency graph for SimpleNetwork.h:




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




### Classes

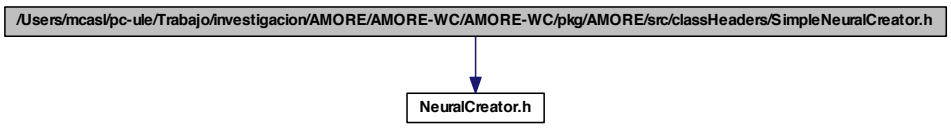
- class SimpleNetwork

  *class SimpleNetwork -*

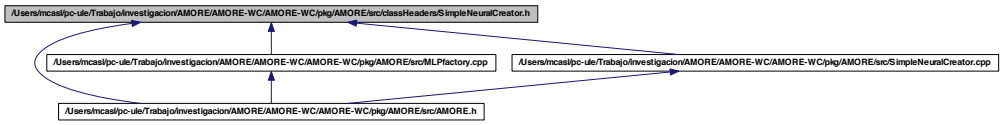
## 6.44 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuralCreator.h File Reference

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

Include dependency graph for SimpleNeuralCreator.h:



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




### Classes

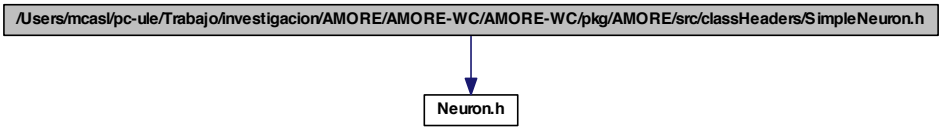
- class SimpleNeuralCreator

  *class SimpleNeuralCreator -*

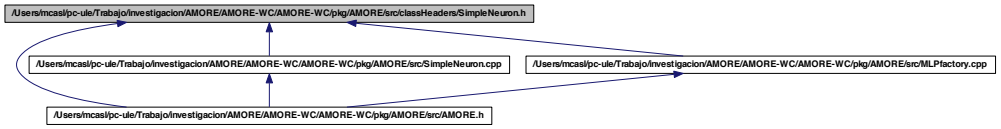
## 6.45 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/SimpleNeuron.h File Reference

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

Include dependency graph for SimpleNeuron.h:




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




### Classes

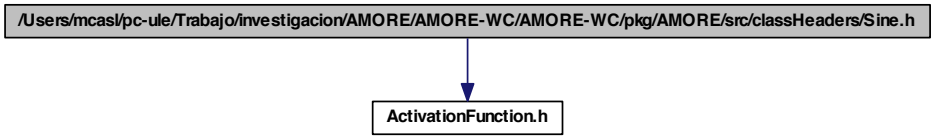
- class SimpleNeuron

*class SimpleNeuron -*

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

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

Include dependency graph for Sine.h:




### Classes

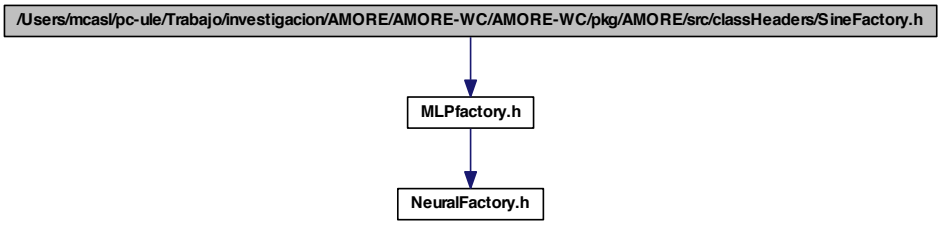
- class Sine

  *class Sine -*

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

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

Include dependency graph for SineFactory.h:



Classes

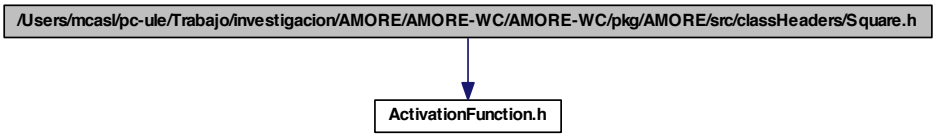
- class SineFactory

  *class SineFactory* -

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

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

Include dependency graph for Square.h:




### Classes

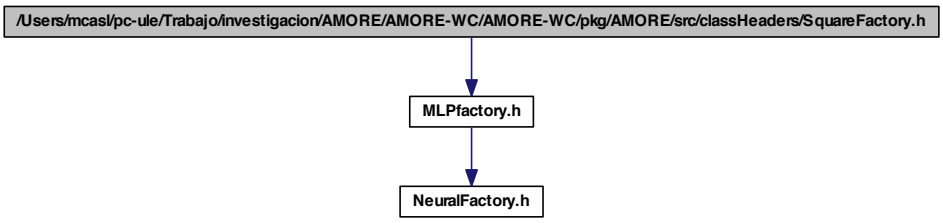
- class Square

  *class Square* -

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

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

Include dependency graph for SquareFactory.h:




### Classes

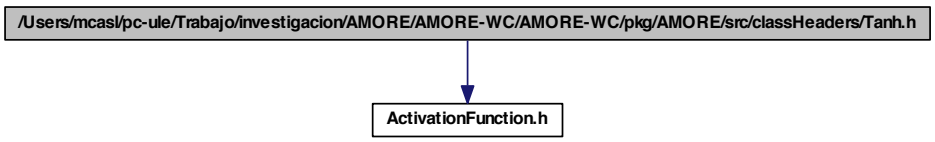
- class SquareFactory

  *class SquareFactory -*

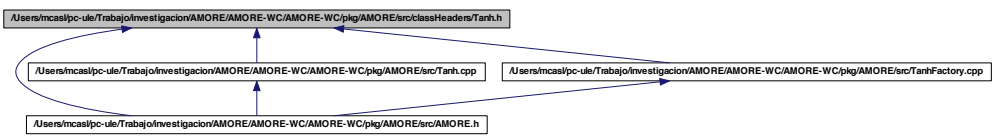
## 6.50 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/Tanh.h File Reference

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

Include dependency graph for Tanh.h:




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



### Classes

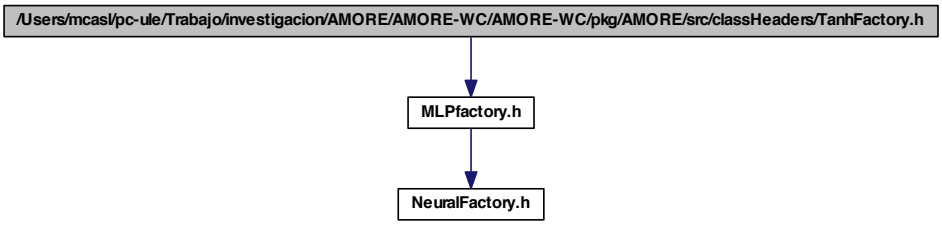
- class Tanh

  *class Tanh -*

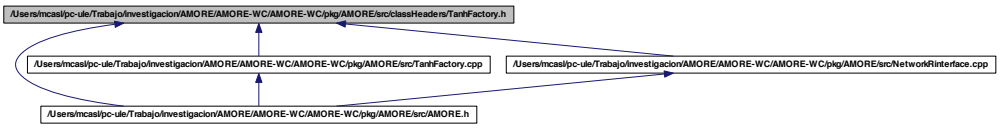
## 6.51 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/classHeaders/TanhFactory.h File Reference

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

Include dependency graph for TanhFactory.h:




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




### Classes

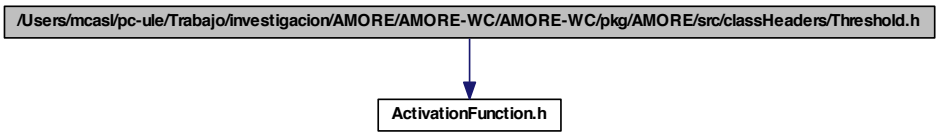
- class TanhFactory

  *class TanhFactory -*

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

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

Include dependency graph for Threshold.h:




### Classes

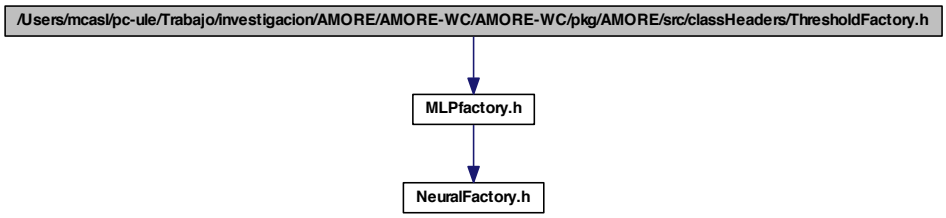
- class Threshold

  *class Threshold -*

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

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

Include dependency graph for ThresholdFactory.h:




### Classes

- class ThresholdFactory

  *class ThresholdFactory -*

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

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

### Classes

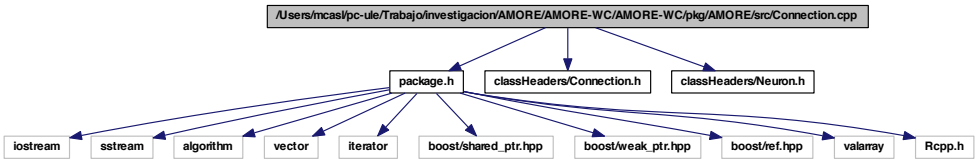
- class TrainingBehavior

  *class TrainingBehavior -*

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

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

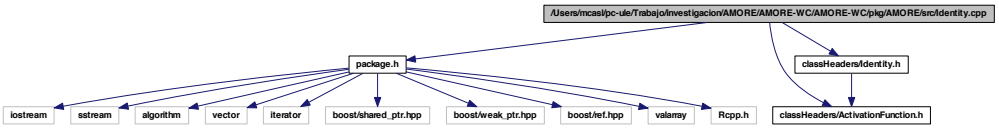
Include dependency graph for Connection.cpp:



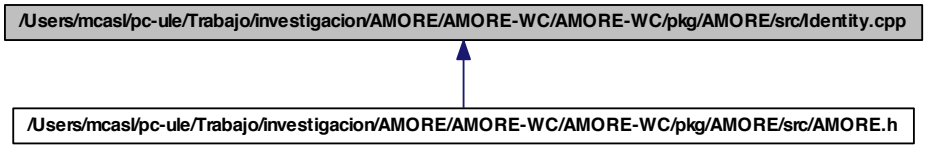

## 6.56 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Identity.cpp File Reference

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

Include dependency graph for Identity.cpp:



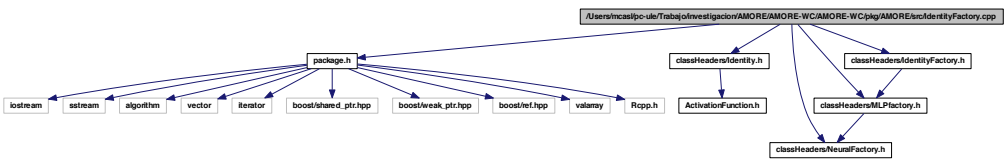

This graph shows which files directly or indirectly include this file:



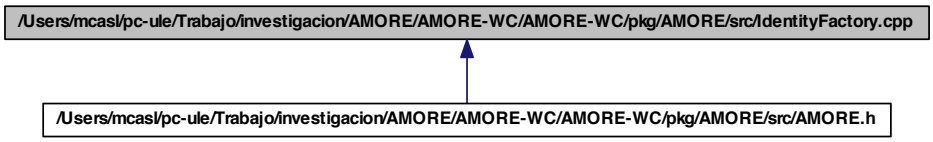

## 6.57 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/IdentityFactory.cpp File Reference

```
#include "package.h"
#include "classHeaders/Identity.h"
#include "classHeaders/NeuralFactory.h"
#include "classHeaders/MLPfactory.h"
#include "classHeaders/IdentityFactory.h"
```

Include dependency graph for IdentityFactory.cpp:



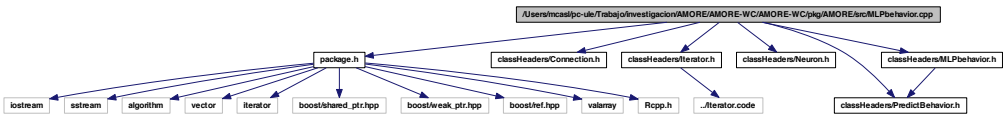
This graph shows which files directly or indirectly include this file:



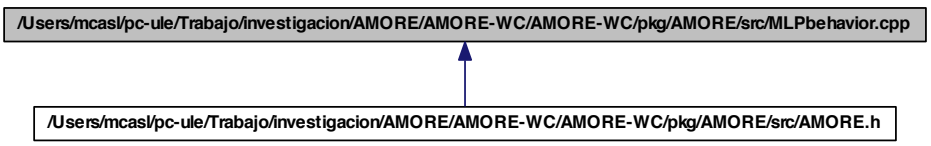

## 6.58 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPbehavior.cpp File Reference

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

Include dependency graph for MLPbehavior.cpp:




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

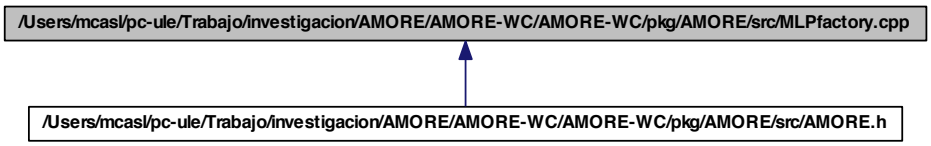


## 6.59 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/MLPfactory.cpp File Reference

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

Include dependency graph for MLPfactory.cpp:

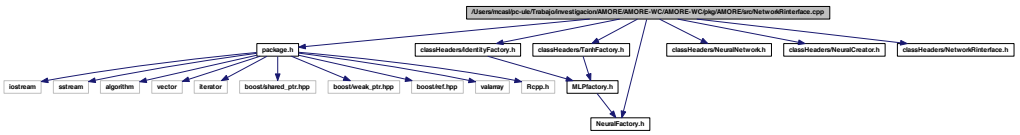
This graph shows which files directly or indirectly include this file:



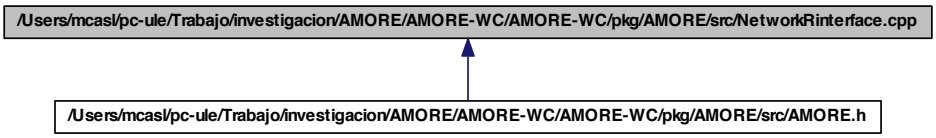
## 6.60 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NetworkRinterface.cpp File Reference

```
#include "package.h"
#include "classHeaders/IdentityFactory.h"
#include "classHeaders/TanhFactory.h"
#include "classHeaders/NeuralFactory.h"
#include "classHeaders/NeuralNetwork.h"
#include "classHeaders/NeuralCreator.h"
#include "classHeaders/NetworkRinterface.h"
```

Include dependency graph for NetworkRinterface.cpp:



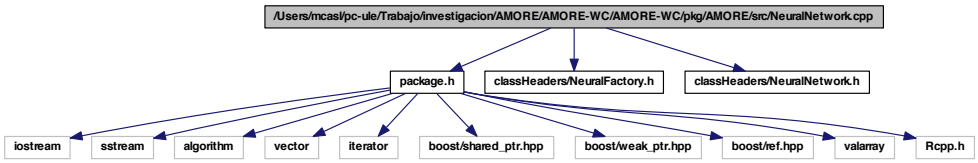

This graph shows which files directly or indirectly include this file:



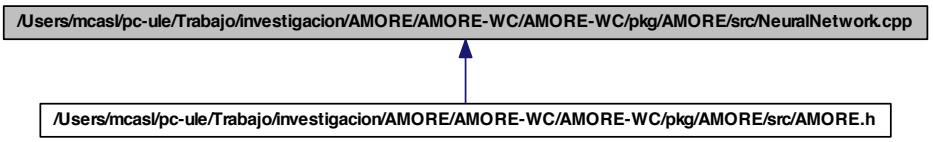

## 6.61 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/NeuralNetwork.cpp File Reference

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

Include dependency graph for NeuralNetwork.cpp:



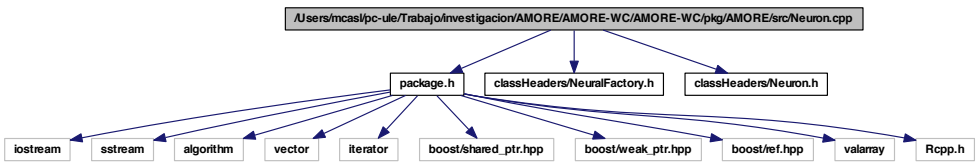

This graph shows which files directly or indirectly include this file:



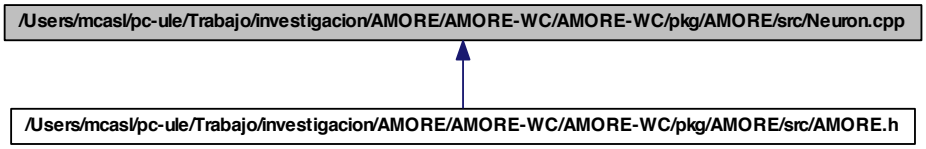

## 6.62 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Neuron.cpp File Reference

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

Include dependency graph for Neuron.cpp:



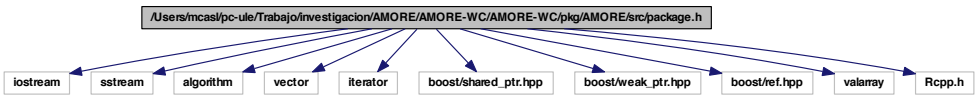
This graph shows which files directly or indirectly include this file:




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

```
#include <iostream>
#include <sstream>
#include <algorithm>
#include <vector>
#include <iterator>
#include <boost/shared_ptr.hpp>
#include <boost/weak_ptr.hpp>
#include <boost/ref.hpp>
#include <valarray>
#include <Rcpp.h>
```

Include dependency graph for package.h:




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

### Defines

- #define size_type unsigned int

### Typedefs

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

## 6.63.1 Define Documentation

### 6.63.1.1 #define size_type unsigned int

Definition at line 75 of file package.h.

## 6.63.2 Typedef Documentation

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

Definition at line 85 of file package.h.

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

Definition at line 80 of file package.h.

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

Definition at line 97 of file package.h.

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

Definition at line 92 of file package.h.

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

Definition at line 88 of file package.h.

#### 6.63.2.6 typedef int Handler

Definition at line 78 of file package.h.

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

Definition at line 95 of file package.h.

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

Definition at line 94 of file package.h.

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

Definition at line 100 of file package.h.

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

Definition at line 99 of file package.h.

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

Definition at line 89 of file package.h.

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

Definition at line 91 of file package.h.

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

Definition at line 87 of file package.h.

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

Definition at line 83 of file package.h.

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

Definition at line 102 of file package.h.

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

Definition at line 86 of file package.h.

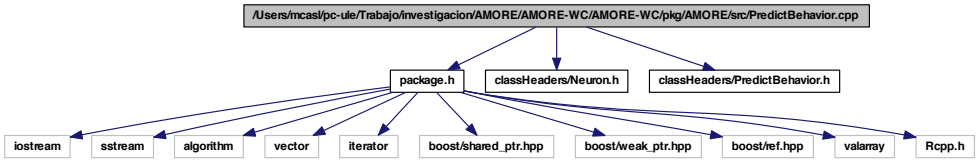
#### 6.63.2.17 typedef boost::reference_wrapper<PredictBehavior> PredictBehaviorRef

Definition at line 81 of file package.h.

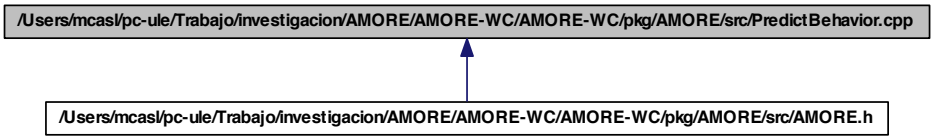
## 6.64 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/PredictBehavior.cpp File Reference

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

Include dependency graph for PredictBehavior.cpp:



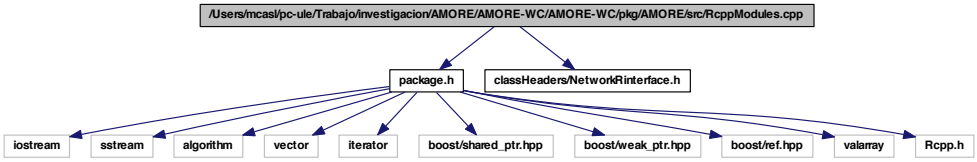
This graph shows which files directly or indirectly include this file:



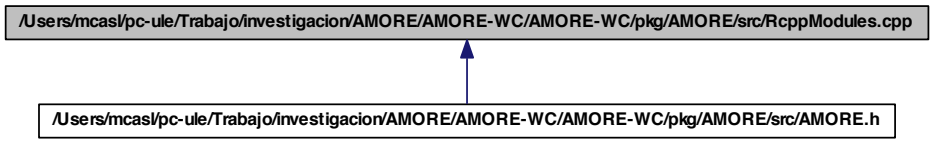

## 6.65 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/RcppModules.cpp File Reference

```
#include "package.h"
#include "classHeaders/NetworkRinterface.h"
```

Include dependency graph for RcppModules.cpp:




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




### Functions

- RCPP_MODULE (modAMORE)

### 6.65.1 Function Documentation

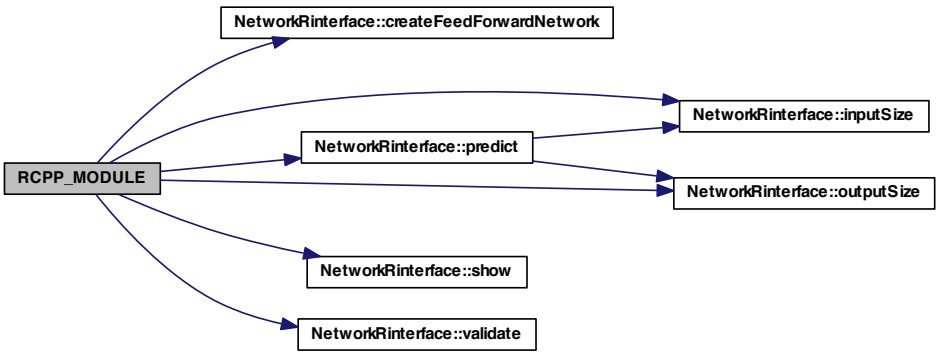
#### 6.65.1.1 RCPP_MODULE ( modAMORE )

Definition at line 5 of file RcppModules.cpp.

References NetworkRinterface::createFeedForwardNetwork(), NetworkRinterface::inputSize(), NetworkRinterface::outputSize(), NetworkRinterface::predict(), NetworkRinterface::show(), and NetworkRinterface::validate().

```
{
  class_<NetworkRinterface>( "NetworkRinterface" )
  .constructor()
  .method( "createFeedForwardNetwork", &
    NetworkRinterface::createFeedForwardNetwork )
  .method( "predict",                  &NetworkRinterface::predict )
  .method( "inputSize",                &NetworkRinterface::inputSize )
  .method( "outputSize",               &NetworkRinterface::outputSize )
  .method( "show",                     &NetworkRinterface::show )
  .method( "validate",                 &NetworkRinterface::validate )
  ;
}
```

Here is the call graph for this function:

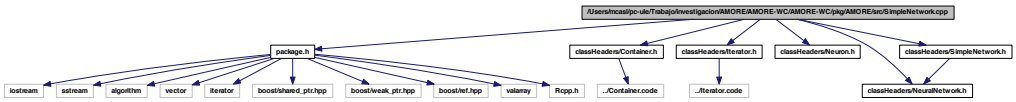



## 6.66 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNetwork.cpp File Reference

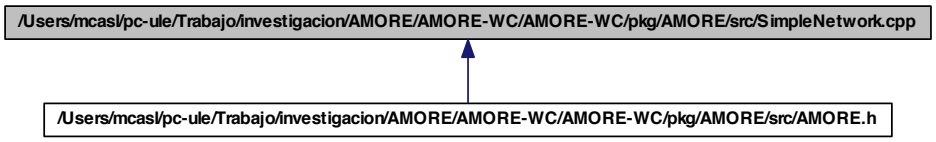
```
#include "package.h"
#include "classHeaders/Container.h"
#include "classHeaders/Iterator.h"
#include "classHeaders/Neuron.h"
```

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

Include dependency graph for SimpleNetwork.cpp:



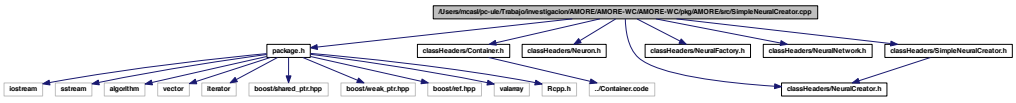

This graph shows which files directly or indirectly include this file:



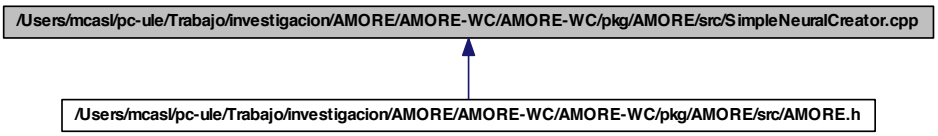

## 6.67 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuralCreator.cpp File Reference

```
#include "package.h"
#include "classHeaders/Container.h"
#include "classHeaders/Neuron.h"
#include "classHeaders/NeuralCreator.h"
#include "classHeaders/NeuralFactory.h"
#include "classHeaders/NeuralNetwork.h"
#include "classHeaders/SimpleNeuralCreator.h"
```

Include dependency graph for SimpleNeuralCreator.cpp:



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

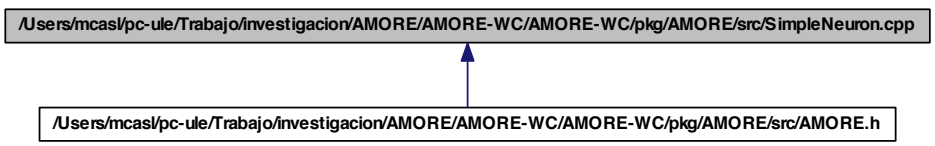



## 6.68 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/SimpleNeuron.cpp File Reference

```
#include "package.h"
#include "classHeaders/NeuralFactory.h"
#include "classHeaders/Container.h"
#include "classHeaders/Iterator.h"
#include "classHeaders/ActivationFunction.h"
#include "classHeaders/PredictBehavior.h"
#include "classHeaders/SimpleNeuron.h"
```

Include dependency graph for SimpleNeuron.cpp:

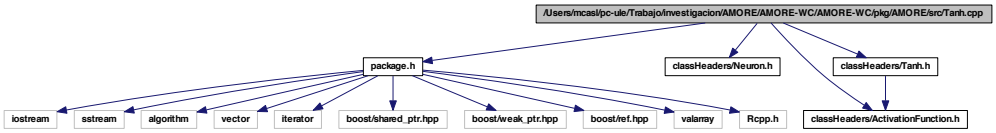

This graph shows which files directly or indirectly include this file:



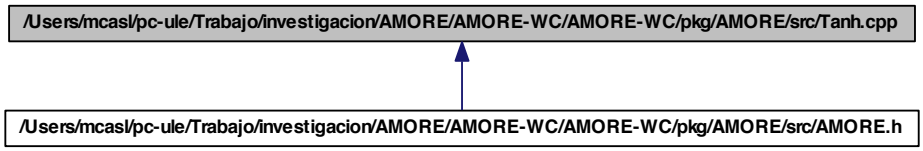
## 6.69 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/Tanh.cpp File Reference

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

Include dependency graph for Tanh.cpp:



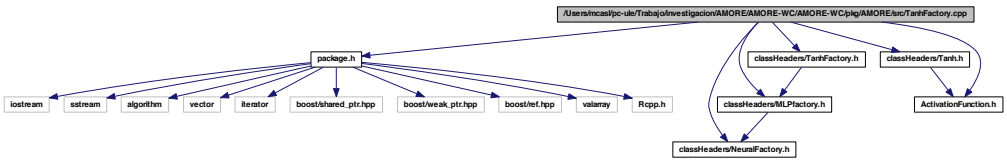

This graph shows which files directly or indirectly include this file:



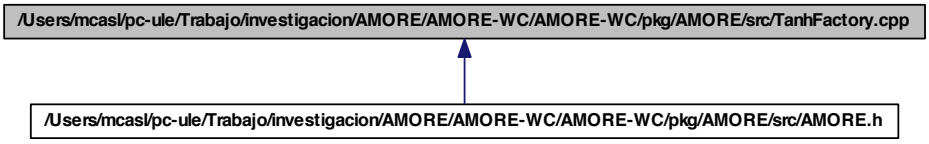

## 6.70 /Users/mcasl/pc-ule/Trabajo/investigacion/AMORE/AMORE-WC/AMORE-WC/pkg/AMORE/src/TanhFactory.cpp File Reference

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

Include dependency graph for TanhFactory.cpp:




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



# Index