# QHopfield

1.0.0

Generated by Doxygen 1.5.8

Fri May 13 11:25:25 2011

# Contents

# Chapter 1

# Class Index

## 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

# Chapter 2

# Class Index

## 2.1 Class List

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

# Chapter 3

# Class Documentation

## 3.1 ActivationFunction Class Reference

Activation function abstract base class.

`#include <ActivationFunction.h>`

Inherited by SaturatedLinearActivationFunction, and SignumActivationFunction.

### Public Member Functions

- double operator() (double x)

  *Apply the activation function on a scalar.*

- VectorXd operator() (VectorXd &X)

  *Apply the activation function on a vector, element-wise.*

### Protected Member Functions

- virtual double **func** (double x)=0

### 3.1.1 Detailed Description

Activation function abstract base class.

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

- /home/brcha/Projects/qhopfield/src/ActivationFunction.h
- /home/brcha/Projects/qhopfield/src/ActivationFunction.cpp

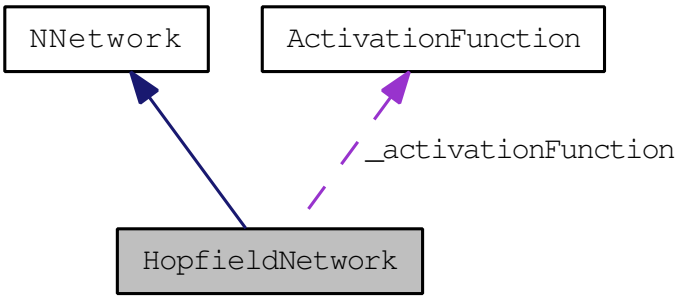
## 3.2 HopfieldNetwork Class Reference

A hopfield network.

`#include <HopfieldNetwork.h>`

Inherits NNetwork.

Collaboration diagram for HopfieldNetwork:




### Public Member Functions

- HopfieldNetwork (uint32_t size)
  *Default constructor.*
- double getWeight (uint32_t from, uint32_t to) const
  *Get the weight between two neurons.*
- void setWeight (uint32_t from, uint32_t to, double weight)
  *Set the weight between two neurons.*
- const MatrixXd & getWeightMatrix () const
  *Get the weight matrix.*
- double getEnergy () const
  *Get the energy of the current network state.*
- double getEnergy (VectorXi &pattern) const
  *Get the energy of the given pattern.*
- uint32_t getSize () const
  *Get the size of the input patterns.*
- void step ()
  *Step to the next network state.*
- uint32_t getTime () const
  *Get time.*
- void randomize ()

*Set random activations.*

- double getThreshold (uint32_t i) const

  *Get threshold for the given neuron.*

- void setThreshold (uint32_t i, double threshold)

  *Set threshold for the given neuron.*

- void setInput (const VectorXi &pattern)

  *Set the input of the neurons.*

- const VectorXd & getOutput () const

  *Get the output of the neurons.*

- VectorXi getOutputInt () const

  *Get the output of the neurons as ints in {-1,0,1}.*

- const VectorXd & getNextOutput ()

  *Step one unit of time and get output of the neurons.*

- const VectorXd & getOutput (const VectorXd &input)

  *Get output for the given input.*

- bool propagate (const VectorXi &pattern, uint32_t timeout=1000)

  *Propagate the input through the network.*

- void addPattern (const VectorXi &pattern)

  *Add a pattern to be memorized.*

- uint32_t getPatternCount () const

  *Get number of stored patterns.*

- void setActivationFunction (ActivationFunction *function)

  *Set activation function.*

## Protected Member Functions

- void _resize (uint32_t size)

  *Resize the network.*

- double _getNeuronOutput (uint32_t i) const

  *Get the output of the i-th neuron.*

- void _setNeuronOutput (uint32_t i, double output)

  *Set the output of the i-th neuron.*

## Protected Attributes

- uint32_t _nPatterns

  *Number of storred patterns.*

- uint32_t _time

  *Discrete time tracking.*

- VectorXd _thresholds

  *Neuron thresholds.*

- VectorXd _outputs

  *Network outputs.*

- ActivationFunction * _activationFunction

  *Activation function.*

- MatrixXd _weights

  *The weight matrix of the network.*

### 3.2.1 Detailed Description

A hopfield network.

Hopfield network is a single-layer content addressable associative memory.

### 3.2.2 Constructor & Destructor Documentation

#### 3.2.2.1 HopfieldNetwork::HopfieldNetwork (uint32_t *size*)

Default constructor.

**Parameters:**

***size*** The size of the network (= size of input patterns)

### 3.2.3 Member Function Documentation

#### 3.2.3.1 void HopfieldNetwork::addPattern (const VectorXi & *pattern*)

Add a pattern to be memorized.

This method adds the pattern and adjusts the weight matrix so that the pattern is memorized.

**Parameters:**

***pattern*** The pattern to be stored

Here is the call graph for this function:


#### 3.2.3.2 double HopfieldNetwork::getEnergy (VectorXi & *pattern*) const

Get the energy of the given pattern.

**Parameters:**

***pattern*** the pattern

**Returns:**

the energy of the pattern

#### 3.2.3.3 double HopfieldNetwork::getEnergy () const

Get the energy of the current network state.

**Returns:**

network state energy

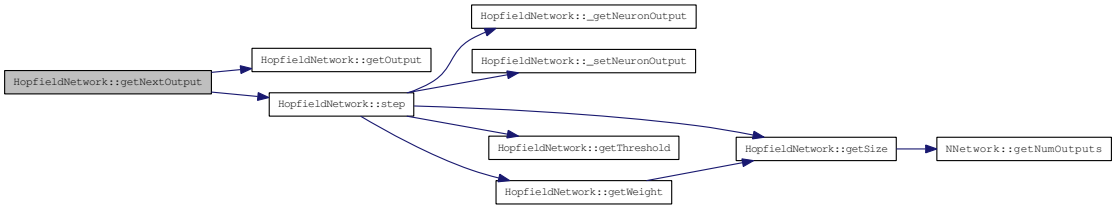
#### 3.2.3.4 const VectorXd & HopfieldNetwork::getNextOutput ()

Step one unit of time and get output of the neurons.

**Returns:**

the output of the neurons

Here is the call graph for this function:




#### 3.2.3.5 const VectorXd & HopfieldNetwork::getOutput (const VectorXd & *input*)

Get output for the given input.

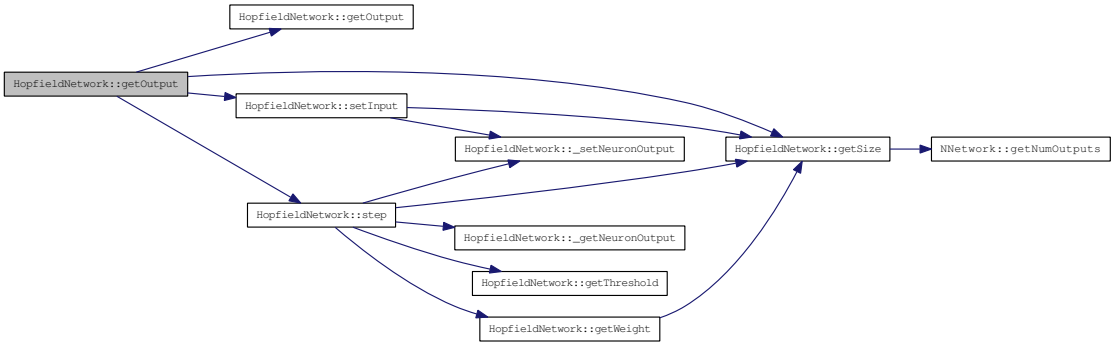
**Parameters:**

***input*** Input pattern feed to the network

**Returns:**

output of the network

Here is the call graph for this function:



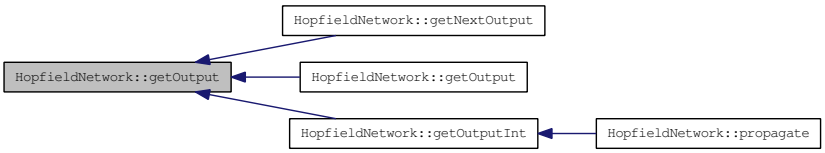

### 3.2.3.6 const VectorXd & HopfieldNetwork::getOutput () const

Get the output of the neurons.

**Returns:**

the current output of the neurons

Here is the caller graph for this function:



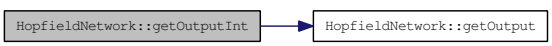

### 3.2.3.7 VectorXi HopfieldNetwork::getOutputInt () const

Get the output of the neurons as ints in {-1,0,1}.

**Returns:**

the current output of the neurons

Here is the call graph for this function:



Here is the caller graph for this function:


#### 3.2.3.8 uint32_t HopfieldNetwork::getPatternCount () const

Get number of stored patterns.

**Returns:**

the number of patterns stored in the network

#### 3.2.3.9 uint32_t HopfieldNetwork::getSize () const

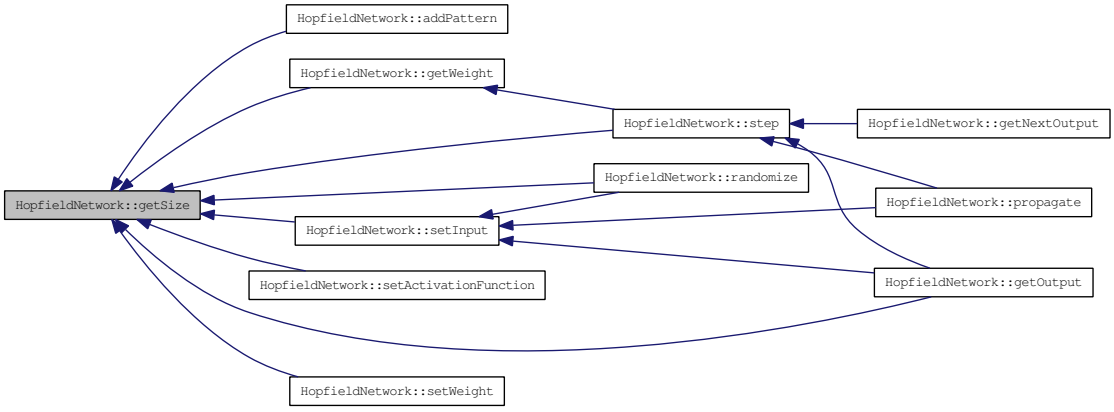
Get the size of the input patterns.

**Returns:**

the size of the patterns

Here is the call graph for this function:


Here is the caller graph for this function:




#### 3.2.3.10 double HopfieldNetwork::getThreshold (uint32_t *i*) const

Get threshold for the given neuron.

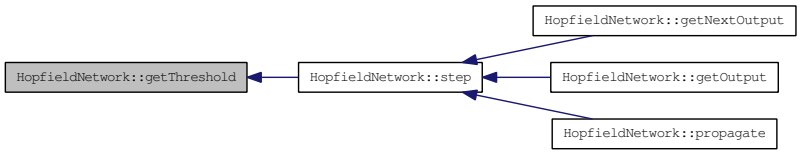
**Parameters:**

***i*** the index of the neuron

**Returns:**

threshold of the neuron

Here is the caller graph for this function:




#### 3.2.3.11 uint32_t HopfieldNetwork::getTime () const

Get time.

**Returns:**

current discrete time

#### 3.2.3.12 double HopfieldNetwork::getWeight (uint32_t *from*, uint32_t *to*) const

Get the weight between two neurons.

**Parameters:**

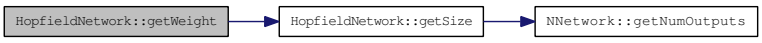
***from*** starting neuron

***to*** ending neuron

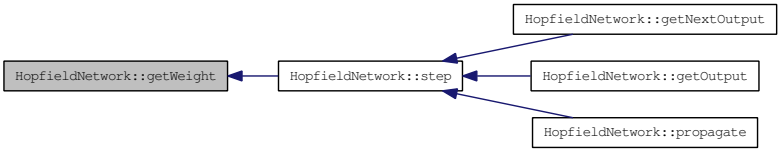
**Returns:**

weight from i-th to j-th neuron

Here is the call graph for this function:




Here is the caller graph for this function:



#### 3.2.3.13 const MatrixXd & HopfieldNetwork::getWeightMatrix () const

Get the weight matrix.

**Returns:**

the weight matrix

#### 3.2.3.14 bool HopfieldNetwork::propagate (const VectorXi & *pattern*, uint32_t *timeout* = 1000)

Propagate the input through the network.

This method propagates the input until the network converges or until the timeout is reached.

**Parameters:**

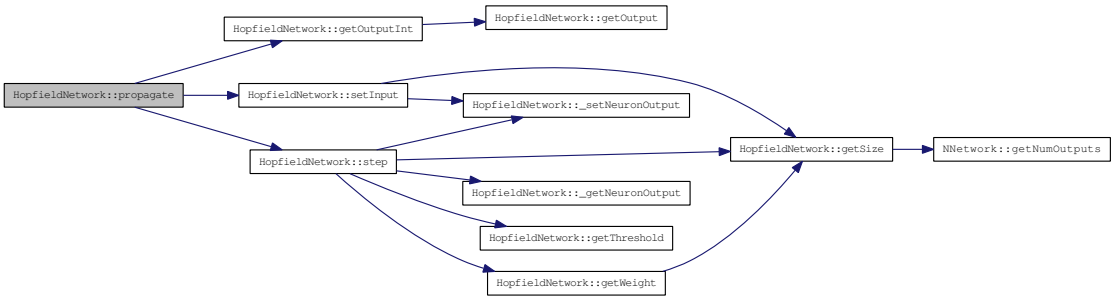
***pattern*** initial pattern for the neurons

***timeout*** maximum number of steps before returning the result

**Returns:**

true if the network converges, false otherwise

Here is the call graph for this function:




#### 3.2.3.15 void HopfieldNetwork::setActivationFunction (ActivationFunction * *function*)

Set activation function.

**Parameters:**

***function*** the new activation function

**Note:**

This method resets the network

Here is the call graph for this function:

#### 3.2.3.16 void HopfieldNetwork::setInput (const VectorXi & *pattern*)

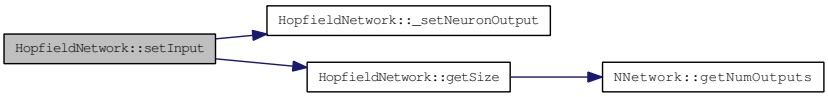
Set the input of the neurons.

This method sets the input of the neurons to the values given by the pattern and resets discrete time to zero.

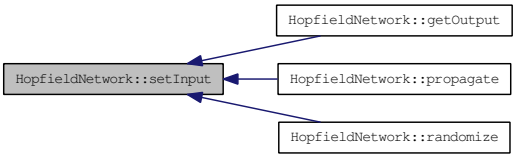
**Parameters:**

***pattern*** A vector containing initial outputs of the neurons.

Here is the call graph for this function:




Here is the caller graph for this function:




#### 3.2.3.17 void HopfieldNetwork::setThreshold (uint32_t *i*, double *threshold*)

Set threshold for the given neuron.

**Parameters:**

***i*** the index of the neuron

***threshold*** threshold of the neuron

#### 3.2.3.18 void HopfieldNetwork::setWeight (uint32_t *from*, uint32_t *to*, double *weight*)

Set the weight between two neurons.

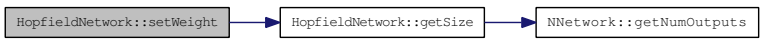
**Parameters:**

***from*** starting neuron

***to*** ending neuron

***w*** weight

Here is the call graph for this function:




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

- /home/brcha/Projects/qhopfield/src/HopfieldNetwork.h
- /home/brcha/Projects/qhopfield/src/HopfieldNetwork.cpp

## 3.3 NNetwork Class Reference

Neural-network abstract base class.

`#include <NNetwork.h>`

Inherited by HopfieldNetwork.

### Public Member Functions

- NNetwork (uint32_t inputs, uint32_t outputs)
  *Default neural-network constructor.*
- NNetwork (NNetwork &net)
  *Copy constructor.*
- uint32_t getNumInputs () const
  *Get number of network inputs.*
- uint32_t getNumOutputs () const
  *Get number of network outputs.*

### Protected Attributes

- uint32_t _nInputs
  *Number of network inputs.*
- uint32_t _nOutputs
  *Number of network outputs.*

### 3.3.1 Detailed Description

Neural-network abstract base class.

This class is a base class for all neural-network classes.

### 3.3.2 Constructor & Destructor Documentation

#### 3.3.2.1 NNetwork::NNetwork (uint32_t *inputs*, uint32_t *outputs*)

Default neural-network constructor.

**Parameters:**

***inputs*** Number of network inputs

***outputs*** Number of network outputs

#### 3.3.2.2 NNetwork::NNetwork (NNetwork & *net*)

Copy constructor.

**Parameters:**

***net*** Network to be copied

### 3.3.3 Member Function Documentation

#### 3.3.3.1 uint32_t NNetwork::getNumInputs () const

Get number of network inputs.

**Returns:**

the number of network inputs

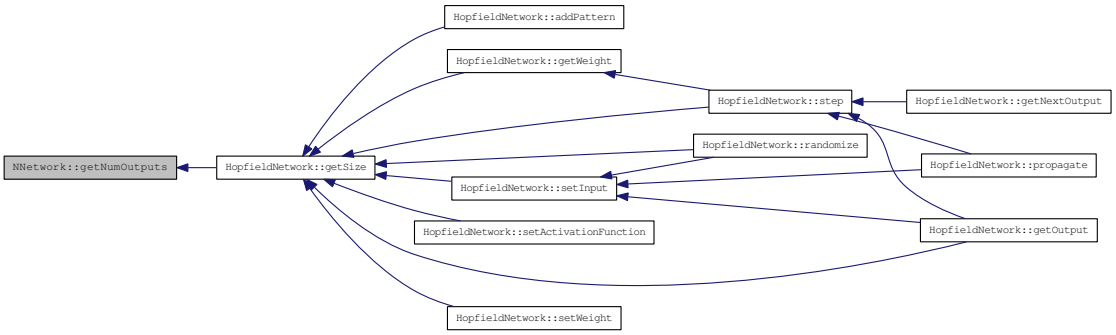
#### 3.3.3.2 uint32_t NNetwork::getNumOutputs () const

Get number of network outputs.

**Returns:**

the number of network outputs

Here is the caller graph for this function:




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

- /home/brcha/Projects/qhopfield/src/NNetwork.h
- /home/brcha/Projects/qhopfield/src/NNetwork.cpp

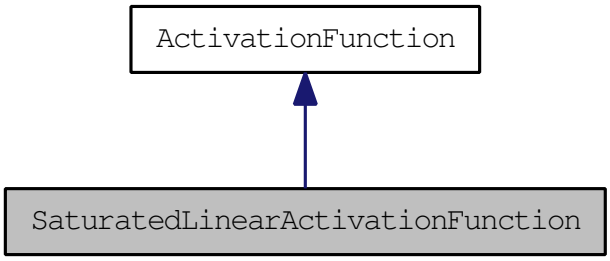
## 3.4 SaturatedLinearActivationFunction Class Reference

Saturated linear activation function.

`#include <SaturatedLinearActivationFunction.h>`

Inherits ActivationFunction.

Collaboration diagram for SaturatedLinearActivationFunction:




### Protected Member Functions

- virtual double **func** (double x)

### 3.4.1 Detailed Description

Saturated linear activation function.

This function returns ±1 if input is outside of (-1,1) region or it copies the input if it is inside (-1,1) region.

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

- /home/brcha/Projects/qhopfield/src/SaturatedLinearActivationFunction.h
- /home/brcha/Projects/qhopfield/src/SaturatedLinearActivationFunction.cpp

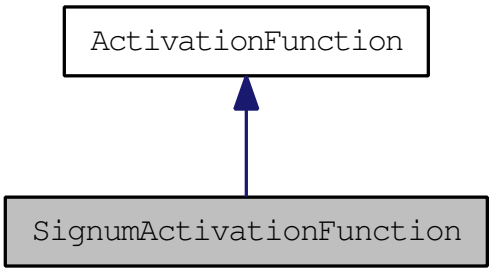
## 3.5 SignumActivationFunction Class Reference

Activation function based on the signum function.

`#include <SignumActivationFunction.h>`

Inherits ActivationFunction.

Collaboration diagram for SignumActivationFunction:




### Protected Member Functions

- virtual double **func** (double x)

### 3.5.1 Detailed Description

Activation function based on the signum function.

The activation function returns the sign of the input value or zero if input is zero.

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

- /home/brcha/Projects/qhopfield/src/SignumActivationFunction.h
- /home/brcha/Projects/qhopfield/src/SignumActivationFunction.cpp

# Index