# Feedforward Closedloop Learning

Generated by Doxygen 1.9.1

# Chapter 1

# Feedforward Closedloop Learning (FCL)

Forward propagation closed loop learning Bernd Porr, Paul Miller. Adaptive Behaviour 2019.

Submission version

For an autonomous agent, the inputs are the sensory data that inform the agent of the state of the world, and the outputs are their actions, which act on the world and consequently produce new sensory inputs. The agent only knows of its own actions via their effect on future inputs; therefore desired states, and error signals, are most naturally defined in terms of the inputs. Most machine learning algorithms, however, operate in terms of desired outputs. For example, backpropagation takes target output values and propagates the corresponding error backwards through the network in order to change the weights. In closed loop settings, it is far more obvious how to define desired sensory inputs than desired actions, however. To train a deep network using errors defined in the input space would call for an algorithm that can propagate those errors forwards through the network, from input layer to output layer, in much the same way that activations are propagated.

Github project page

# Chapter 2

# Hierarchical Index

## 2.1 Class Hierarchy

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

# Chapter 3

# Class Index

## 3.1 Class List

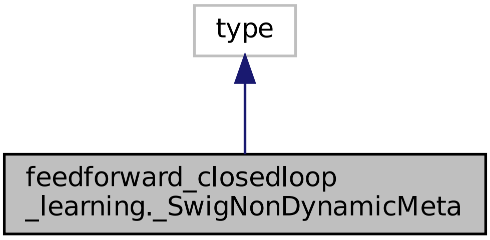
Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 4

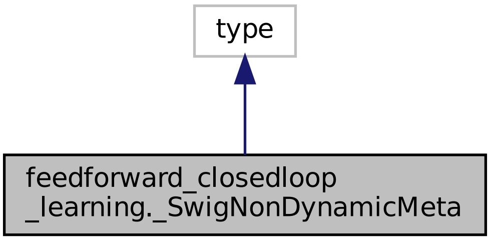
# Class Documentation

## 4.1 feedforward_closedloop_learning._SwigNonDynamicMeta Class Reference

Inheritance diagram for feedforward_closedloop_learning._SwigNonDynamicMeta:



Collaboration diagram for feedforward_closedloop_learning._SwigNonDynamicMeta:



### 4.1.1 Detailed Description

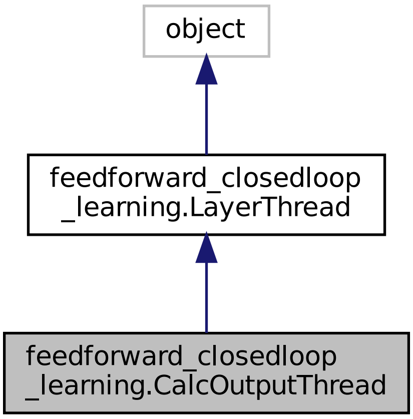
```
Meta class to enforce nondynamic attributes (no new attributes) for a class
```

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

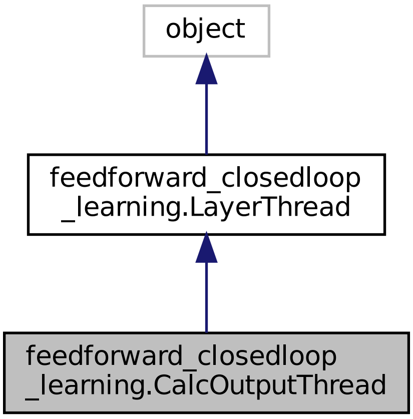
- feedforward_closedloop_learning.py

## 4.2 feedforward_closedloop_learning.CalcOutputThread Class Reference

Inheritance diagram for feedforward_closedloop_learning.CalcOutputThread:


object
feedforward_closedloop _learning.LayerThread
feedforward_closedloop _learning.CalcOutputThread

Collaboration diagram for feedforward_closedloop_learning.CalcOutputThread:


object
feedforward_closedloop _learning.LayerThread
feedforward_closedloop _learning.CalcOutputThread

### Public Member Functions

- def **__init__** (self, ∗args, ∗∗kwargs)

### Properties

- **thisown** = property(lambda x: x.this.own(), lambda x, v: x.this.own(v), doc="The membership flag")

### 4.2.1 Detailed Description

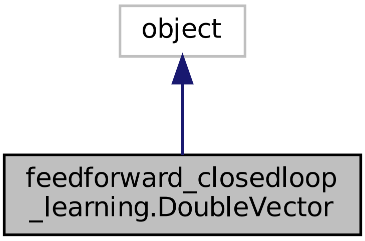
```
Proxy of C++ CalcOutputThread class.
```

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

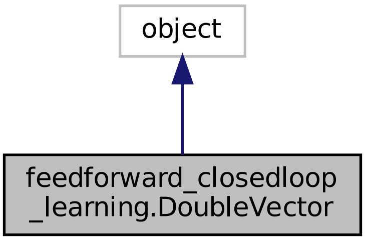
- feedforward_closedloop_learning.py

## 4.3 feedforward_closedloop_learning.DoubleVector Class Reference

Inheritance diagram for feedforward_closedloop_learning.DoubleVector:



Collaboration diagram for feedforward_closedloop_learning.DoubleVector:



## Public Member Functions

- "swig::SwigPyIterator *" **iterator** (self)
- def **__iter__** (self)
- "bool" **__nonzero__** (self)
- "bool" **__bool__** (self)
- "std::vector< double >::size_type" **__len__** (self)
- "std::vector< double,std::allocator< double > > *" **__getslice__** (self, "std::vector< double >::difference↩ _type" i, "std::vector< double >::difference_type" j)
- "void" **__setslice__** (self, *args)
- "void" **__delslice__** (self, "std::vector< double >::difference_type" i, "std::vector< double >::difference_↩ type" j)
- "void" **__delitem__** (self, *args)
- "std::vector< double >::value_type const &" **__getitem__** (self, *args)
- "void" **__setitem__** (self, *args)
- "std::vector< double >::value_type" **pop** (self)
- "void" **append** (self, "std::vector< double >::value_type const &" x)
- "bool" **empty** (self)
- "std::vector< double >::size_type" **size** (self)
- "void" **swap** (self, "DoubleVector" v)
- "std::vector< double >::iterator" **begin** (self)
- "std::vector< double >::iterator" **end** (self)
- "std::vector< double >::reverse_iterator" **rbegin** (self)
- "std::vector< double >::reverse_iterator" **rend** (self)
- "void" **clear** (self)
- "std::vector< double >::allocator_type" **get_allocator** (self)
- "void" **pop_back** (self)
- "std::vector< double >::iterator" **erase** (self, *args)
- def **__init__** (self, *args)
- "void" **push_back** (self, "std::vector< double >::value_type const &" x)
- "std::vector< double >::value_type const &" **front** (self)
- "std::vector< double >::value_type const &" **back** (self)
- "void" **assign** (self, "std::vector< double >::size_type" n, "std::vector< double >::value_type const &" x)
- "void" **resize** (self, *args)
- "void" **insert** (self, *args)
- "void" **reserve** (self, "std::vector< double >::size_type" n)
- "std::vector< double >::size_type" **capacity** (self)

## Properties

- **thisown** = property(lambda x: x.this.own(), lambda x, v: x.this.own(v), doc="The membership flag")

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

- feedforward_closedloop_learning.py

# 4.4 FCLBandpass Class Reference

Creates memory traces at specified length.

```
#include <bandpass.h>
```

## Public Member Functions

- FCLBandpass ()

  *Constructor.*
- double filter (double v)

  *Filter.*
- void calcPolesZeros (double f, double r)

  *Calculates the coefficients The frequency is the normalized frequency in the range [0..0.5].*
- void setParameters (double frequency, double Qfactor)

  *sets the filter parameters*
- void impulse (char ∗name)

  *Generates an acsii file with the impulse response of the filter.*
- void calcNorm (double f)

  *Normalises the output with f.*
- void transfer (char ∗name)

  *Generates an ASCII file with the transfer function.*
- double getOutput ()

  *Gets the output of the filter.*
- void reset ()

  *Sets the output to zero again.*

### 4.4.1 Detailed Description

Creates memory traces at specified length.

It's a 2nd order IIR filter.

### 4.4.2 Member Function Documentation

#### 4.4.2.1 getOutput()

```
double FCLBandpass::getOutput ( )  [inline]
```

Gets the output of the filter.

Same as the return value of the function "filter()".

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

- bandpass.h
- bandpass.cpp

## 4.5 FCLLayer Class Reference

Layer which contains the neurons of one layer.

```
#include <layer.h>
```

## Public Types

- enum WeightNormalisation {
**WEIGHT_NORM_NONE** = 0 , **WEIGHT_NORM_LAYER_EUCLEDIAN** = 1 , **WEIGHT_NORM_NEURON_EUCLEDIAN** = 2 , **WEIGHT_NORM_LAYER_MANHATTAN** = 3 ,
**WEIGHT_NORM_NEURON_MANHATTAN** = 4 , **WEIGHT_NORM_LAYER_INFINITY** = 5 , **WEIGHT_NORM_NEURON_INFINITY** = 6 }

  *Weight normalisation constants Defines if weights are normalised layer-wide or for every neuron separately.*

## Public Member Functions

- FCLLayer (int _nNeurons, int _nInputs)

  *Constructor.*
- ∼FCLLayer ()

  *Destructor Frees all memory.*
- void calcOutputs ()

  *Calculates the output values in all neurons.*
- void doLearning ()

  *Adjusts the weights.*
- void setError (double _error)

  *Sets the global error for all neurons.*
- void setError (int i, double _error)

  *sets the error individually*
- void setErrors (double ∗_errors)

  *Sets all errors from an input array.*
- double getError (int i)

  *Retrieves the error.*
- void setBias (double _bias)

  *Sets the global bias for all neurons.*
- void setInput (int inputIndex, double input)

  *Set the input value of one input.*
- void setInputs (const double ∗_inputs)

  *Sets all inputs from an input array.*
- void setLearningRate (double _learningRate)

  *Sets the learning rate of all neurons.*
- void setActivationFunction (FCLNeuron::ActivationFunction _activationFunction)

  *Set the activation function.*
- void setMomentum (double _momentum)

  *Set the momentum of all neurons in this layer.*
- void setDecay (double _decay)

  *Sets the weight decay scaled by the learning rate.*
- void initWeights (double _max=1, int initBiasWeight=1, FCLNeuron::WeightInitMethod weightInitMethod=FCLNeuron::MAX_OUTPUT_RANDOM)

  *Inits the weights.*
- double getOutput (int index)

  *Gets the outpuut of one neuron.*
- FCLNeuron ∗ getNeuron (int index)

  *Gets a pointer to one neuron.*
- int getNneurons ()

  *Gets the number of neurons.*
- int getNinputs ()

*Number of inputs.*

- void setConvolution (int width, int height)

  *Defines a 2D geometry for the input layer of widthxheight.*
- void setMaxDetLayer (int _m)

  *Maxium detection layer.*
- void setNormaliseWeights (WeightNormalisation _normaliseWeights)

  *Normalise the weights.*
- void setDebugInfo (int layerIndex)

  *Sets the layer index within the whole network.*
- void setStep (long int step)

  *Sets the simulation step in the layer for debug purposes.*
- double getWeightDistanceFromInitialWeights ()

  *Get weight distance from the start of the simulation.*
- void doNormaliseWeights ()

  *Performs the weight normalisation.*
- void setUseThreads (int _useThreads)

  *Sets if threads should be used.*
- int saveWeightMatrix (char ∗filename)

  *Save weight matrix for documentation and debugging.*

## 4.5.1 Detailed Description

Layer which contains the neurons of one layer.

It performs all computations possible in a layer. In particular it calls all neurons in separate threads and triggers the compuations there. These functions are all called from the parent class.

## 4.5.2 Constructor & Destructor Documentation

### 4.5.2.1 FCLLayer()

```
FCLLayer::FCLLayer (
            int _nNeurons,
            int _nInputs )
```

Constructor.

GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007.

**Parameters**

| | |
|---|---|
| *_nNeurons* | Number of neurons in the layer. |
| *_nInputs* | Number of inputs to the Layer. |
| *_nFilters* | Number of lowpass filters at each input. |
| *_minT* | Minimum time of the lowpass filter. |
| *_maxT* | Maximum time of the lowpass filter. |

## 4.5.3 Member Function Documentation

### 4.5.3.1 getError()

```
double FCLLayer::getError (
             int i )
```

Retrieves the error.

**Parameters**

| | |
|---|---|
| *i* | Index of the neuron |

### 4.5.3.2 getNeuron()

```
FCLNeuron* FCLLayer::getNeuron (
             int index )   [inline]
```

Gets a pointer to one neuron.

**Parameters**

| | |
|---|---|
| *index* | The index number of the neuron. |

**Returns**

A pointer to a Layer class.

### 4.5.3.3 getNinputs()

```
int FCLLayer::getNinputs ( )   [inline]
```

Number of inputs.

**Returns**

The number of inputs

#### 4.5.3.4 getNneurons()

```
int FCLLayer::getNneurons ( )  [inline]
```

Gets the number of neurons.

**Returns**

The number of neurons.

#### 4.5.3.5 getOutput()

```
double FCLLayer::getOutput (
             int index )  [inline]
```

Gets the outpuut of one neuron.

**Parameters**

| | |
|---|---|
| *index* | The index number of the neuron. |

**Returns**

Retuns the double valye of the output.

#### 4.5.3.6 getWeightDistanceFromInitialWeights()

```
double FCLLayer::getWeightDistanceFromInitialWeights ( )
```

Get weight distance from the start of the simulation.

**Returns**

The distance from the initial (random) weight setup.

#### 4.5.3.7 initWeights()

```
void FCLLayer::initWeights (
             double _max = 1,
             int initBiasWeight = 1,
             FCLNeuron::WeightInitMethod weightInitMethod = FCLNeuron::MAX_OUTPUT_RANDOM )
```

Inits the weights.

**Parameters**

| | |
|---|---|
| *_max* | Maximum value if using random init. |
| *initBiasWeight* | if one also the bias weight is initialised. |
| *weightInitMethod* | The methid employed to init the weights. |

#### 4.5.3.8 saveWeightMatrix()

```
int FCLLayer::saveWeightMatrix (
             char * filename )
```

Save weight matrix for documentation and debugging.

**Parameters**

| | |
|---|---|
| *filename* | The filename it should be saved to. |

#### 4.5.3.9 setActivationFunction()

```
void FCLLayer::setActivationFunction (
             FCLNeuron::ActivationFunction _activationFunction )
```

Set the activation function.

**Parameters**

| | |
|---|---|
| *_activationFunction* | The activation function. See: Neuron::ActivationFunction |

#### 4.5.3.10 setBias()

```
void FCLLayer::setBias (
             double _bias )
```

Sets the global bias for all neurons.

**Parameters**

| | |
|---|---|
| *_bias* | The bias for all neurons |

#### 4.5.3.11 setConvolution()

```
void FCLLayer::setConvolution (
             int width,
             int height )
```

Defines a 2D geometry for the input layer of widthxheight.

**Parameters**

| | |
|---|---|
| *width* | The width of the convolutional window. |
| *height* | The height of the convolution window. |

#### 4.5.3.12 setDebugInfo()

```
void FCLLayer::setDebugInfo (
             int layerIndex )
```

Sets the layer index within the whole network.

**Parameters**

| | |
|---|---|
| *layerIndex* | The layer index in the whole network. |

#### 4.5.3.13 setDecay()

```
void FCLLayer::setDecay (
             double _decay )
```

Sets the weight decay scaled by the learning rate.

**Parameters**

| | |
|---|---|
| *_decay* | The decay rate of the weights |

#### 4.5.3.14 setError() [1/2]

```
void FCLLayer::setError (
             double _error )
```

Sets the global error for all neurons.

**Parameters**

| | |
|---|---|
| *_error* | Sets the error in the whole layer |

### 4.5.3.15 setError() [2/2]

```
void FCLLayer::setError (
             int i,
             double _error )
```

sets the error individually

**Parameters**

| | |
|---|---|
| *i* | Index of the neuron |
| *_error* | The error to be set |

### 4.5.3.16 setErrors()

```
void FCLLayer::setErrors (
             double * _errors )
```

Sets all errors from an input array.

**Parameters**

| | |
|---|---|
| *_errors* | is an array of errors |

### 4.5.3.17 setInput()

```
void FCLLayer::setInput (
             int inputIndex,
             double input )
```

Set the input value of one input.

**Parameters**

| | |
|---|---|
| *inputIndex* | The index number of the input. |
| *input* | The value of the input |

#### 4.5.3.18 setInputs()

```
void FCLLayer::setInputs (
             const double * _inputs )
```

Sets all inputs from an input array.

**Parameters**

| | |
|---|---|
| *_inputs* | array of all inputs |

#### 4.5.3.19 setLearningRate()

```
void FCLLayer::setLearningRate (
             double _learningRate )
```

Sets the learning rate of all neurons.

**Parameters**

| | |
|---|---|
| *_learningRate* | The learning rate |

#### 4.5.3.20 setMaxDetLayer()

```
void FCLLayer::setMaxDetLayer (
             int _m )  [inline]
```

Maxium detection layer.

Experimental. This hasn't been implemented.

#### 4.5.3.21 setMomentum()

```
void FCLLayer::setMomentum (
             double _momentum )
```

Set the momentum of all neurons in this layer.

**Parameters**

| | |
|---|---|
| *_momentum* | The momentum for all neurons in this layer. |

#### 4.5.3.22 setNormaliseWeights()

```
void FCLLayer::setNormaliseWeights (
            WeightNormalisation _normaliseWeights )
```

Normalise the weights.

**Parameters**

| | |
|---|---|
| *_normaliseWeights* | Metod of normalisation. |

#### 4.5.3.23 setStep()

```
void FCLLayer::setStep (
            long int step )
```

Sets the simulation step in the layer for debug purposes.

**Parameters**

| | |
|---|---|
| *step* | Step number. |

#### 4.5.3.24 setUseThreads()

```
void FCLLayer::setUseThreads (
            int _useThreads )  [inline]
```

Sets if threads should be used.

**Parameters**

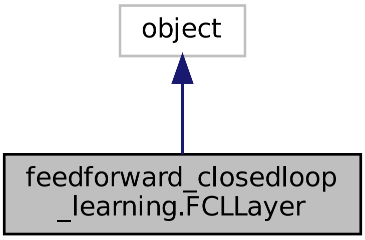
| | |
|---|---|
| *_useThreads* | 0 = no Threads, 1 = Threads |

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

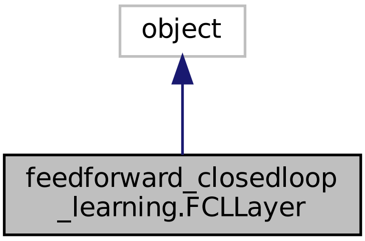
- layer.h
- layer.cpp

## 4.6 feedforward_closedloop_learning.FCLLayer Class Reference

Inheritance diagram for feedforward_closedloop_learning.FCLLayer:



Collaboration diagram for feedforward_closedloop_learning.FCLLayer:



### Public Member Functions

- def __init__ (self, "int" _nNeurons, "int" _nInputs)
- "void" calcOutputs (self)
- "void" doLearning (self)
- "void" setError (self, ∗args)
- "void" setErrors (self, "double ∗" _errors)
- "double" getError (self, "int" i)
- "void" setBias (self, "double" _bias)
- "void" setInput (self, "int" inputIndex, "double" input)
- "void" setInputs (self, "double const ∗" _inputs)
- "void" setLearningRate (self, "double" _learningRate)
- "void" setActivationFunction (self, "FCLNeuron::ActivationFunction" _activationFunction)
- "void" setMomentum (self, "double" _momentum)
- "void" setDecay (self, "double" _decay)
- "void" initWeights (self, ∗args)

- "double" getOutput (self, "int" index)
- "FCLNeuron ∗" getNeuron (self, "int" index)
- "int" getNneurons (self)
- "int" getNinputs (self)
- "void" setConvolution (self, "int" width, "int" height)
- "void" setMaxDetLayer (self, "int" _m)
- "void" setNormaliseWeights (self, "FCLLayer::WeightNormalisation" _normaliseWeights)
- "void" setDebugInfo (self, "int" layerIndex)
- "void" setStep (self, "long" step)
- "double" getWeightDistanceFromInitialWeights (self)
- "void" doNormaliseWeights (self)
- "void" setUseThreads (self, "int" _useThreads)
- "int" saveWeightMatrix (self, "char ∗" filename)

### Static Public Attributes

- **WEIGHT_NORM_NONE** = _feedforward_closedloop_learning.FCLLayer_WEIGHT_NORM_NONE
- **WEIGHT_NORM_LAYER_EUCLEDIAN** = _feedforward_closedloop_learning.FCLLayer_WEIGHT_↩
NORM_LAYER_EUCLEDIAN
- **WEIGHT_NORM_NEURON_EUCLEDIAN** = _feedforward_closedloop_learning.FCLLayer_WEIGHT_↩
NORM_NEURON_EUCLEDIAN
- **WEIGHT_NORM_LAYER_MANHATTAN** = _feedforward_closedloop_learning.FCLLayer_WEIGHT_↩
NORM_LAYER_MANHATTAN
- **WEIGHT_NORM_NEURON_MANHATTAN** = _feedforward_closedloop_learning.FCLLayer_WEIGHT_↩
NORM_NEURON_MANHATTAN
- **WEIGHT_NORM_LAYER_INFINITY** = _feedforward_closedloop_learning.FCLLayer_WEIGHT_NORM_↩
LAYER_INFINITY
- **WEIGHT_NORM_NEURON_INFINITY** = _feedforward_closedloop_learning.FCLLayer_WEIGHT_NORM↩
_NEURON_INFINITY

### Properties

- **thisown** = property(lambda x: x.this.own(), lambda x, v: x.this.own(v), doc="The membership flag")

## 4.6.1 Detailed Description

```
Proxy of C++ FCLLayer class.
```

## 4.6.2 Constructor & Destructor Documentation

### 4.6.2.1 __init__()

```
def feedforward_closedloop_learning.FCLLayer.__init__ (
             self,
            "int" _nNeurons,
            "int" _nInputs )
```

```
__init__(FCLLayer self, int _nNeurons, int _nInputs) -> FCLLayer

Parameters
----------
_nNeurons: int
_nInputs: int
```

### 4.6.3 Member Function Documentation

#### 4.6.3.1 calcOutputs()

```
"void" feedforward_closedloop_learning.FCLLayer.calcOutputs (
              self )
```

```
calcOutputs(FCLLayer self)
```

#### 4.6.3.2 doLearning()

```
"void" feedforward_closedloop_learning.FCLLayer.doLearning (
              self )
```

```
doLearning(FCLLayer self)
```

#### 4.6.3.3 doNormaliseWeights()

```
"void" feedforward_closedloop_learning.FCLLayer.doNormaliseWeights (
              self )
```

```
doNormaliseWeights(FCLLayer self)
```

#### 4.6.3.4 getError()

```
"double" feedforward_closedloop_learning.FCLLayer.getError (
              self,
             "int" i )
```

```
getError(FCLLayer self, int i) -> double

Parameters
----------
i: int
```

#### 4.6.3.5 getNeuron()

```
"FCLNeuron *" feedforward_closedloop_learning.FCLLayer.getNeuron (
              self,
             "int" index )
```

```
getNeuron(FCLLayer self, int index) -> FCLNeuron

Parameters
----------
index: int
```

#### 4.6.3.6 getNinputs()

```
"int" feedforward_closedloop_learning.FCLLayer.getNinputs (
              self )
```

```
getNinputs(FCLLayer self) -> int
```

#### 4.6.3.7 getNneurons()

```
"int" feedforward_closedloop_learning.FCLLayer.getNneurons (
              self )
```

```
getNneurons(FCLLayer self) -> int
```

#### 4.6.3.8 getOutput()

```
"double" feedforward_closedloop_learning.FCLLayer.getOutput (
              self,
             "int" index )
```

```
getOutput(FCLLayer self, int index) -> double

Parameters
----------
index: int
```

#### 4.6.3.9 getWeightDistanceFromInitialWeights()

```
"double" feedforward_closedloop_learning.FCLLayer.getWeightDistanceFromInitialWeights (
              self )
```

```
getWeightDistanceFromInitialWeights(FCLLayer self) -> double
```

#### 4.6.3.10 initWeights()

```
"void" feedforward_closedloop_learning.FCLLayer.initWeights (
              self,
             * args )
```

```
initWeights(FCLLayer self, double _max=1, int initBiasWeight=1, FCLNeuron::WeightInitMethod weightInitMethod=M

Parameters
----------
_max: double
initBiasWeight: int
weightInitMethod: enum FCLNeuron::WeightInitMethod
```

#### 4.6.3.11 saveWeightMatrix()

```
"int" feedforward_closedloop_learning.FCLLayer.saveWeightMatrix (
              self,
             "char *" filename )
```

```
saveWeightMatrix(FCLLayer self, char * filename) -> int

Parameters
----------
filename: char *
```

#### 4.6.3.12 setActivationFunction()

```
"void" feedforward_closedloop_learning.FCLLayer.setActivationFunction (
              self,
             "FCLNeuron::ActivationFunction" _activationFunction )
```

```
setActivationFunction(FCLLayer self, FCLNeuron::ActivationFunction _activationFunction)

Parameters
----------
_activationFunction: enum FCLNeuron::ActivationFunction
```

#### 4.6.3.13 setBias()

```
"void" feedforward_closedloop_learning.FCLLayer.setBias (
              self,
             "double" _bias )
```

```
setBias(FCLLayer self, double _bias)

Parameters
----------
_bias: double
```

#### 4.6.3.14 setConvolution()

```
"void" feedforward_closedloop_learning.FCLLayer.setConvolution (
              self,
             "int" width,
             "int" height )
```

```
setConvolution(FCLLayer self, int width, int height)

Parameters
----------
width: int
height: int
```

#### 4.6.3.15 setDebugInfo()

```
"void" feedforward_closedloop_learning.FCLLayer.setDebugInfo (
              self,
             "int" layerIndex )
```

```
setDebugInfo(FCLLayer self, int layerIndex)

Parameters
----------
layerIndex: int
```

#### 4.6.3.16 setDecay()

```
"void" feedforward_closedloop_learning.FCLLayer.setDecay (
              self,
             "double" _decay )
```

```
setDecay(FCLLayer self, double _decay)

Parameters
----------
_decay: double
```

#### 4.6.3.17 setError()

```
"void" feedforward_closedloop_learning.FCLLayer.setError (
              self,
             * args )
```

```
setError(FCLLayer self, double _error)

Parameters
----------
_error: double

setError(FCLLayer self, int i, double _error)

Parameters
----------
i: int
_error: double
```

#### 4.6.3.18 setErrors()

```
"void" feedforward_closedloop_learning.FCLLayer.setErrors (
              self,
             "double *" _errors )
```

```
setErrors(FCLLayer self, double * _errors)

Parameters
----------
_errors: double *
```

#### 4.6.3.19 setInput()

```
"void" feedforward_closedloop_learning.FCLLayer.setInput (
              self,
             "int" inputIndex,
             "double" input )
```

```
setInput(FCLLayer self, int inputIndex, double input)

Parameters
----------
inputIndex: int
input: double
```

#### 4.6.3.20 setInputs()

```
"void" feedforward_closedloop_learning.FCLLayer.setInputs (
              self,
             "double const *" _inputs )
```

```
setInputs(FCLLayer self, double const * _inputs)

Parameters
----------
_inputs: double const *
```

#### 4.6.3.21 setLearningRate()

```
"void" feedforward_closedloop_learning.FCLLayer.setLearningRate (
              self,
             "double" _learningRate )
```

```
setLearningRate(FCLLayer self, double _learningRate)

Parameters
----------
_learningRate: double
```

#### 4.6.3.22 setMaxDetLayer()

```
"void" feedforward_closedloop_learning.FCLLayer.setMaxDetLayer (
              self,
             "int" _m )
```

```
setMaxDetLayer(FCLLayer self, int _m)

Parameters
----------
_m: int
```

#### 4.6.3.23 setMomentum()

```
"void" feedforward_closedloop_learning.FCLLayer.setMomentum (
              self,
             "double" _momentum )
```

```
setMomentum(FCLLayer self, double _momentum)

Parameters
----------
_momentum: double
```

#### 4.6.3.24 setNormaliseWeights()

```
"void" feedforward_closedloop_learning.FCLLayer.setNormaliseWeights (
              self,
             "FCLLayer::WeightNormalisation" _normaliseWeights )
```

```
setNormaliseWeights(FCLLayer self, FCLLayer::WeightNormalisation _normaliseWeights)

Parameters
----------
_normaliseWeights: enum FCLLayer::WeightNormalisation
```

#### 4.6.3.25 setStep()

```
"void" feedforward_closedloop_learning.FCLLayer.setStep (
              self,
             "long" step )
```

```
setStep(FCLLayer self, long step)

Parameters
----------
step: long
```

#### 4.6.3.26 setUseThreads()

```
"void" feedforward_closedloop_learning.FCLLayer.setUseThreads (
              self,
             "int" _useThreads )
```

```
setUseThreads(FCLLayer self, int _useThreads)

Parameters
----------
_useThreads: int
```

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

- feedforward_closedloop_learning.py

## 4.7 FCLNeuron Class Reference

Neuron which calculates the output and performs learning.

```
#include <neuron.h>
```

## Public Types

- enum WeightInitMethod { **MAX_OUTPUT_RANDOM** = 0 , **MAX_WEIGHT_RANDOM** = 1 , **MAX_OUTPUT↩ _CONST** = 2 , **CONST_WEIGHTS** = 3 }

  *Constants how to init the weights in the neuron.*
- enum ActivationFunction {
  **LINEAR** = 0 , **TANH** = 1 , **RELU** = 2 , **REMAXLU** = 3 ,
  **TANHLIMIT** = 4 }

  *Activation functions on offer LINEAR: linear unit, TANH: tangens hyperbolicus, RELU: linear rectifier, REMAXLU: as RELU but limits to one.*

## Public Member Functions

- FCLNeuron (int _nInputs)

  *Constructor.*
- ∼FCLNeuron ()

  *Destructor Tidies up any memory allocations.*
- void calcOutput ()

  *Calculate the output of the neuron This runs the filters, activation functions, sum it all up.*
- void doLearning ()

  *Performs the learning Performs ICO learning in the neuron: pre ∗ error.*
- void doMaxDet ()

  *Detects max of an input Switches the highest weight to 1 and the others to 0.*
- void initWeights (double _max=1, int initBias=1, WeightInitMethod _wm=MAX_OUTPUT_RANDOM)

  *Inits the weights in the neuron.*
- void setActivationFunction (ActivationFunction _activationFunction)

  *Sets the activation function.*
- double dActivation ()

  *Returns the output of the neuron fed through the derivative of the activation.*
- double getMinWeightValue ()

  *Minimum weight value.*
- double getMaxWeightValue ()

  *Maximum weight value.*
- double getWeightDistanceFromInitialWeights ()

  *Weight development.*
- double getOutput ()

  *Gets the output of the neuron.*
- double getSum ()

  *Gets the weighted sum of all inputs pre-activation function.*
- double getWeight (int _index)

  *Gets one weight.*
- void setWeight (int _index, double _weight)

  *Sets one weight.*
- void setError (double _error)

  *Sets the error in the neuron If the derivative is activated then the derivative of the error is calculated.*
- double getError ()

  *Gets the error as set by setError.*
- void setInput (int _index, double _value)

  *Sets one input.*
- double getInput (int _index)

  *Get the value at one input.*

- double getBiasWeight ()

    *Gets the bias weight.*
- void setBiasWeight (double _biasweight)

    *Sets the bias weight.*
- void setBias (double _bias)

    *Sets the bias input value.*
- void setLearningRate (double _learningrate)

    *Sets the learning rate.*
- void setMomentum (double _momentum)

    *Sets the momentum.*
- void setDecay (double _decay)

    *Sets the weight decay over time.*
- double getDecay ()

    *Gets the weight decay over time.*
- int getNinputs ()

    *Get the number of inputs to the neuron.*
- void setGeometry (int _width, int _height)

    *Tells the layer that it's been a 2D array originally to be a convolutional layer.*
- void setMask (int x, int y, unsigned char c)

    *Boundary safe manipulation of the convolution mask.*
- void setMask (unsigned char c)

    *Init the whole mask with a single value.*
- unsigned char getMask (int x, int y)

    *Boundary safe return of the mask in (x,y) coordinates.*
- unsigned char getMask (int index)

    *Boundary safe return of the mask in flat form.*
- double getSumOfSquaredWeightVector ()

    *Calculates the sum of the squared weight vector values.*
- double getEuclideanNormOfWeightVector ()

    *Calculates the Eucledian length of the weight vector.*
- double getManhattanNormOfWeightVector ()

    *Calculates the Manhattan length of the weight vector /return Manhattan length of the weight vector.*
- double getInfinityNormOfWeightVector ()

    *Calculates the Infinity norm of the vector.*
- double getAverageOfWeightVector ()

    *Calculates the average of the weight values.*
- void normaliseWeights (double norm)

    *Normalises the weights with a divisor.*
- void saveInitialWeights ()

    *Save the initial weights.*
- void setDebugInfo (int _layerIndex, int _neuronIndex)

    *Sets debug info populated from Layer.*
- void setStep (long int _step)

    *Sets the simulation step for debugging and logging.*

## Static Public Member Functions

- static void ∗ calcOutputThread (void ∗object)

    *Wrapper for thread callback for output calc.*
- static void ∗ doLearningThread (void ∗object)

    *Wrapper for thread callback for learning.*
- static void ∗ doMaxDetThread (void ∗object)

    *Wrapper for thread callback for maxdet.*

### 4.7.1 Detailed Description

Neuron which calculates the output and performs learning.

### 4.7.2 Constructor & Destructor Documentation

#### 4.7.2.1 FCLNeuron()

```
FCLNeuron::FCLNeuron (
            int _nInputs )
```

Constructor.

**Parameters**

| | |
|---|---|
| *_nInputs* | Number of inputs to the Neuron |

### 4.7.3 Member Function Documentation

#### 4.7.3.1 dActivation()

```
double FCLNeuron::dActivation ( )
```

Returns the output of the neuron fed through the derivative of the activation.

**Returns**

Result

#### 4.7.3.2 getAverageOfWeightVector()

```
double FCLNeuron::getAverageOfWeightVector ( )
```

Calculates the average of the weight values.

**Returns**

average of the weight values.

#### 4.7.3.3 getBiasWeight()

```
double FCLNeuron::getBiasWeight ( )  [inline]
```

Gets the bias weight.

**Returns**

Bias weight value

#### 4.7.3.4 getDecay()

```
double FCLNeuron::getDecay ( )  [inline]
```

Gets the weight decay over time.

**Returns**

The weight decay value. The larger the faster the weight decay.

#### 4.7.3.5 getError()

```
double FCLNeuron::getError ( )  [inline]
```

Gets the error as set by setError.

**Returns**

The error value stored in the neuron

#### 4.7.3.6 getEuclideanNormOfWeightVector()

```
double FCLNeuron::getEuclideanNormOfWeightVector ( )  [inline]
```

Calculates the Eucledian length of the weight vector.

**Returns**

Eucledian length of the weight vector.

#### 4.7.3.7 getInfinityNormOfWeightVector()

```
double FCLNeuron::getInfinityNormOfWeightVector ( )
```

Calculates the Infinity norm of the vector.

/return Infinity norm of the vector.

#### 4.7.3.8 getInput()

```
double FCLNeuron::getInput (
             int _index )  [inline]
```

Get the value at one input.

**Parameters**

| | |
|---|---|
| *_index* | Index of the input |

**Returns**

Returns the input value

#### 4.7.3.9 getMask() [1/2]

```
unsigned char FCLNeuron::getMask (
            int index )  [inline]
```

Boundary safe return of the mask in flat form.

**Parameters**

| | |
|---|---|
| *index* | Mask index. |

**Returns**

The mask at the index: 0 = ignore underlying value, 1 = process underlying value.

#### 4.7.3.10 getMask() [2/2]

```
unsigned char FCLNeuron::getMask (
            int x,
            int y )
```

Boundary safe return of the mask in (x,y) coordinates.

**Parameters**

| | |
|---|---|
| *x* | Sets the mask value at coordinate x (0 .. width). |
| *y* | Sets the mask value at coordinate y (0 .. height). |

**Returns**

The mask at x,y: 0 = ignore underlying value, 1 = process underlying value.

#### 4.7.3.11 getMaxWeightValue()

```
double FCLNeuron::getMaxWeightValue ( )
```

Maximum weight value.

**Returns**

The maximum weight value in this neuron

#### 4.7.3.12 getMinWeightValue()

```
double FCLNeuron::getMinWeightValue ( )
```

Minimum weight value.

**Returns**

The minimum weight value in this neuron

#### 4.7.3.13 getNinputs()

```
int FCLNeuron::getNinputs ( )  [inline]
```

Get the number of inputs to the neuron.

**Returns**

The numer of inputs

#### 4.7.3.14 getOutput()

```
double FCLNeuron::getOutput ( )  [inline]
```

Gets the output of the neuron.

**Returns**

The overall output of the neuron after the activation function

#### 4.7.3.15 getSum()

```
double FCLNeuron::getSum ( ) [inline]
```

Gets the weighted sum of all inputs pre-activation function.

**Returns**

Weighted sum (linear)

#### 4.7.3.16 getSumOfSquaredWeightVector()

```
double FCLNeuron::getSumOfSquaredWeightVector ( )
```

Calculates the sum of the squared weight vector values.

**Returns**

The squared weight vector values.

#### 4.7.3.17 getWeight()

```
double FCLNeuron::getWeight (
            int _index ) [inline]
```

Gets one weight.

**Parameters**

| | |
|---|---|
| *_index* | The input index |

**Returns**

The weight value at one input and one filter

#### 4.7.3.18 getWeightDistanceFromInitialWeights()

```
double FCLNeuron::getWeightDistanceFromInitialWeights ( )
```

Weight development.

**Returns**

Returns the Euclidean distance of the weights from their starting position

#### 4.7.3.19 initWeights()

```
void FCLNeuron::initWeights (
             double _max = 1,
             int initBias = 1,
             WeightInitMethod _wm = MAX_OUTPUT_RANDOM )
```

Inits the weights in the neuron.

**Parameters**

| | |
|---|---|
| *_max* | Maximum value of the weights. |
| *initBias* | If one also the bias weight is initialised. |
| *_wm* | Method how to init the weights as defined by WeightInitMethod. |

#### 4.7.3.20 normaliseWeights()

```
void FCLNeuron::normaliseWeights (
             double norm )
```

Normalises the weights with a divisor.

**Parameters**

| | |
|---|---|
| *norm* | Divisor which normalises the weights. |

#### 4.7.3.21 saveInitialWeights()

```
void FCLNeuron::saveInitialWeights ( )
```

Save the initial weights.

This saves the initial weights for later comparisons. For internal use.

#### 4.7.3.22 setActivationFunction()

```
void FCLNeuron::setActivationFunction (
             ActivationFunction _activationFunction ) [inline]
```

Sets the activation function.

**Parameters**

| | |
|---|---|
| *_activationFunction* | Sets the activiation function according to ActivationFunction. |

### 4.7.3.23 setBias()

```
void FCLNeuron::setBias (
            double _bias )  [inline]
```

Sets the bias input value.

**Parameters**

| | |
|---|---|
| *_bias* | Bias value. |

### 4.7.3.24 setBiasWeight()

```
void FCLNeuron::setBiasWeight (
            double _biasweight )  [inline]
```

Sets the bias weight.

**Parameters**

| | |
|---|---|
| *_biasweight* | Bias value. |

### 4.7.3.25 setDebugInfo()

```
void FCLNeuron::setDebugInfo (
            int _layerIndex,
            int _neuronIndex )  [inline]
```

Sets debug info populated from Layer.

**Parameters**

| | |
|---|---|
| *_layerIndex* | The layer the neuron is in. |
| *_neuronIndex* | The index of the neuron in the layer. |

### 4.7.3.26 setDecay()

```
void FCLNeuron::setDecay (
            double _decay )  [inline]
```

Sets the weight decay over time.

**Parameters**

| | |
|---|---|
| *_decay* | The larger the faster the weight decay. |

#### 4.7.3.27 setError()

```
void FCLNeuron::setError (
            double _error )
```

Sets the error in the neuron If the derivative is activated then the derivative of the error is calculated.

**Parameters**

| | |
|---|---|
| *_error* | Sets the error of the neuron. |

#### 4.7.3.28 setGeometry()

```
void FCLNeuron::setGeometry (
            int _width,
            int _height )  [inline]
```

Tells the layer that it's been a 2D array originally to be a convolutional layer.

_width $*$ _height == nInputs. Otherwise an exception is triggered. The geometry entered here is then used in the mask operations so that every neuron is able to process a subset of the input space, for example an image and thus becomes a localised receptive field.

**Parameters**

| | |
|---|---|
| *_width* | The width of the layer |
| *_height* | of the layer |

#### 4.7.3.29 setInput()

```
void FCLNeuron::setInput (
            int _index,
            double _value )  [inline]
```

Sets one input.

**Parameters**

| | |
|---|---|
| *_index* | Index of the input. |
| *_value* | of the input. |

#### 4.7.3.30 setLearningRate()

```
void FCLNeuron::setLearningRate (
            double _learningrate )  [inline]
```

Sets the learning rate.

**Parameters**

| | |
|---|---|
| *_learningrate* | The learning rate |

#### 4.7.3.31 setMask() [1/2]

```
void FCLNeuron::setMask (
            int x,
            int y,
            unsigned char c )
```

Boundary safe manipulation of the convolution mask.

Sets the convolution mask using the geometry defined by setGeometry.

**Parameters**

| | |
|---|---|
| *x* | Sets the mask value at coordinate x (0 .. width). |
| *y* | Sets the mask value at coordinate y (0 .. height). |
| *c* | Sets the mask: 0 = ignore underlying value, 1 = process underlying value. |

#### 4.7.3.32 setMask() [2/2]

```
void FCLNeuron::setMask (
            unsigned char c )
```

Init the whole mask with a single value.

**Parameters**

| | |
|---|---|
| *c* | Sets the mask for the whole array. 0 = ignore the entire input, 1 = process every input. |

#### 4.7.3.33 setMomentum()

```
void FCLNeuron::setMomentum (
            double _momentum )  [inline]
```

Sets the momentum.

Sets the inertia of the learning.

**Parameters**

| | |
|---|---|
| *_momentum* | The new momentum |

#### 4.7.3.34 setStep()

```
void FCLNeuron::setStep (
            long int _step )  [inline]
```

Sets the simulation step for debugging and logging.

**Parameters**

| | |
|---|---|
| *_step* | Current simulation step. |

#### 4.7.3.35 setWeight()

```
void FCLNeuron::setWeight (
            int _index,
            double _weight )  [inline]
```

Sets one weight.

**Parameters**

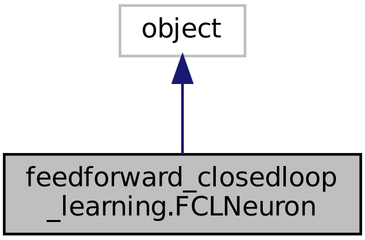
| | |
|---|---|
| *_index* | The input index |
| *_weight* | The weight value |

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

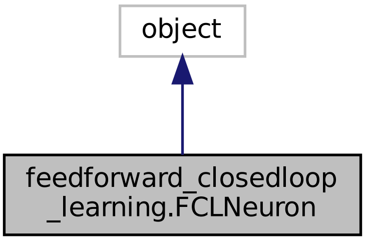
- neuron.h
- neuron.cpp

## 4.8 feedforward_closedloop_learning.FCLNeuron Class Reference

Inheritance diagram for feedforward_closedloop_learning.FCLNeuron:



Collaboration diagram for feedforward_closedloop_learning.FCLNeuron:



### Public Member Functions

- def __init__ (self, "int" _nInputs)
- "void" calcOutput (self)
- "void" doLearning (self)
- "void" doMaxDet (self)
- "void" initWeights (self, ∗args)
- "void" setActivationFunction (self, "FCLNeuron::ActivationFunction" _activationFunction)
- "double" dActivation (self)
- "double" getMinWeightValue (self)
- "double" getMaxWeightValue (self)
- "double" getWeightDistanceFromInitialWeights (self)
- "double" getOutput (self)
- "double" getSum (self)
- "double" getWeight (self, "int" _index)
- "void" setWeight (self, "int" _index, "double" _weight)

- "void" setError (self, "double" _error)
- "double" getError (self)
- "void" setInput (self, "int" _index, "double" _value)
- "double" getInput (self, "int" _index)
- "double" getBiasWeight (self)
- "void" setBiasWeight (self, "double" _biasweight)
- "void" setBias (self, "double" _bias)
- "void" setLearningRate (self, "double" _learningrate)
- "void" setMomentum (self, "double" _momentum)
- "void" setDecay (self, "double" _decay)
- "double" getDecay (self)
- "int" getNinputs (self)
- "void" setGeometry (self, "int" _width, "int" _height)
- "void" setMask (self, ∗args)
- "unsigned char" getMask (self, ∗args)
- "double" getSumOfSquaredWeightVector (self)
- "double" getEuclideanNormOfWeightVector (self)
- "double" getManhattanNormOfWeightVector (self)
- "double" getInfinityNormOfWeightVector (self)
- "double" getAverageOfWeightVector (self)
- "void" normaliseWeights (self, "double" norm)
- "void" saveInitialWeights (self)
- "void" setDebugInfo (self, "int" _layerIndex, "int" _neuronIndex)
- "void" setStep (self, "long" _step)

## Static Public Member Functions

- "void ∗" calcOutputThread ("void ∗" object)
- "void ∗" doLearningThread ("void ∗" object)
- "void ∗" doMaxDetThread ("void ∗" object)

## Static Public Attributes

- **MAX_OUTPUT_RANDOM** = _feedforward_closedloop_learning.FCLNeuron_MAX_OUTPUT_RANDOM
- **MAX_WEIGHT_RANDOM** = _feedforward_closedloop_learning.FCLNeuron_MAX_WEIGHT_RANDOM
- **MAX_OUTPUT_CONST** = _feedforward_closedloop_learning.FCLNeuron_MAX_OUTPUT_CONST
- **CONST_WEIGHTS** = _feedforward_closedloop_learning.FCLNeuron_CONST_WEIGHTS
- **LINEAR** = _feedforward_closedloop_learning.FCLNeuron_LINEAR
- **TANH** = _feedforward_closedloop_learning.FCLNeuron_TANH
- **RELU** = _feedforward_closedloop_learning.FCLNeuron_RELU
- **REMAXLU** = _feedforward_closedloop_learning.FCLNeuron_REMAXLU
- **TANHLIMIT** = _feedforward_closedloop_learning.FCLNeuron_TANHLIMIT

## Properties

- **thisown** = property(lambda x: x.this.own(), lambda x, v: x.this.own(v), doc="The membership flag")

### 4.8.1 Detailed Description

```
Proxy of C++ FCLNeuron class.
```

## 4.8.2 Constructor & Destructor Documentation

### 4.8.2.1 __init__()

```
def feedforward_closedloop_learning.FCLNeuron.__init__ (
              self,
             "int" _nInputs )
```

```
__init__(FCLNeuron self, int _nInputs) -> FCLNeuron

Parameters
----------
_nInputs: int
```

## 4.8.3 Member Function Documentation

### 4.8.3.1 calcOutput()

```
"void" feedforward_closedloop_learning.FCLNeuron.calcOutput (
              self )
```

```
calcOutput(FCLNeuron self)
```

### 4.8.3.2 calcOutputThread()

```
"void *" feedforward_closedloop_learning.FCLNeuron.calcOutputThread (
             "void *" object )  [static]
```

```
calcOutputThread(void * object) -> void *

Parameters
----------
object: void *
```

### 4.8.3.3 dActivation()

```
"double" feedforward_closedloop_learning.FCLNeuron.dActivation (
              self )
```

```
dActivation(FCLNeuron self) -> double
```

#### 4.8.3.4 doLearning()

```
"void" feedforward_closedloop_learning.FCLNeuron.doLearning (
             self )
```

```
doLearning(FCLNeuron self)
```

#### 4.8.3.5 doLearningThread()

```
"void *" feedforward_closedloop_learning.FCLNeuron.doLearningThread (
             "void *" object )  [static]
```

```
doLearningThread(void * object) -> void *

Parameters
----------
object: void *
```

#### 4.8.3.6 doMaxDet()

```
"void" feedforward_closedloop_learning.FCLNeuron.doMaxDet (
             self )
```

```
doMaxDet(FCLNeuron self)
```

#### 4.8.3.7 doMaxDetThread()

```
"void *" feedforward_closedloop_learning.FCLNeuron.doMaxDetThread (
             "void *" object )  [static]
```

```
doMaxDetThread(void * object) -> void *

Parameters
----------
object: void *
```

#### 4.8.3.8 getAverageOfWeightVector()

```
"double" feedforward_closedloop_learning.FCLNeuron.getAverageOfWeightVector (
             self )
```

```
getAverageOfWeightVector(FCLNeuron self) -> double
```

#### 4.8.3.9 getBiasWeight()

```
"double" feedforward_closedloop_learning.FCLNeuron.getBiasWeight (
             self )
```

```
getBiasWeight(FCLNeuron self) -> double
```

#### 4.8.3.10 getDecay()

```
"double" feedforward_closedloop_learning.FCLNeuron.getDecay (
             self )
```

```
getDecay(FCLNeuron self) -> double
```

#### 4.8.3.11 getError()

```
"double" feedforward_closedloop_learning.FCLNeuron.getError (
             self )
```

```
getError(FCLNeuron self) -> double
```

#### 4.8.3.12 getEuclideanNormOfWeightVector()

```
"double" feedforward_closedloop_learning.FCLNeuron.getEuclideanNormOfWeightVector (
             self )
```

```
getEuclideanNormOfWeightVector(FCLNeuron self) -> double
```

#### 4.8.3.13 getInfinityNormOfWeightVector()

```
"double" feedforward_closedloop_learning.FCLNeuron.getInfinityNormOfWeightVector (
              self )
```

```
getInfinityNormOfWeightVector(FCLNeuron self) -> double
```

#### 4.8.3.14 getInput()

```
"double" feedforward_closedloop_learning.FCLNeuron.getInput (
              self,
             "int" _index )
```

```
getInput(FCLNeuron self, int _index) -> double

Parameters
----------
_index: int
```

#### 4.8.3.15 getManhattanNormOfWeightVector()

```
"double" feedforward_closedloop_learning.FCLNeuron.getManhattanNormOfWeightVector (
              self )
```

```
getManhattanNormOfWeightVector(FCLNeuron self) -> double
```

#### 4.8.3.16 getMask()

```
"unsigned char" feedforward_closedloop_learning.FCLNeuron.getMask (
              self,
             * args )
```

```
getMask(FCLNeuron self, int x, int y) -> unsigned char

Parameters
----------
x: int
y: int

getMask(FCLNeuron self, int index) -> unsigned char

Parameters
----------
index: int
```

#### 4.8.3.17 getMaxWeightValue()

```
"double" feedforward_closedloop_learning.FCLNeuron.getMaxWeightValue (
              self )
```

```
getMaxWeightValue(FCLNeuron self) -> double
```

#### 4.8.3.18 getMinWeightValue()

```
"double" feedforward_closedloop_learning.FCLNeuron.getMinWeightValue (
              self )
```

```
getMinWeightValue(FCLNeuron self) -> double
```

#### 4.8.3.19 getNinputs()

```
"int" feedforward_closedloop_learning.FCLNeuron.getNinputs (
              self )
```

```
getNinputs(FCLNeuron self) -> int
```

#### 4.8.3.20 getOutput()

```
"double" feedforward_closedloop_learning.FCLNeuron.getOutput (
              self )
```

```
getOutput(FCLNeuron self) -> double
```

#### 4.8.3.21 getSum()

```
"double" feedforward_closedloop_learning.FCLNeuron.getSum (
              self )
```

```
getSum(FCLNeuron self) -> double
```

#### 4.8.3.22 getSumOfSquaredWeightVector()

```
"double" feedforward_closedloop_learning.FCLNeuron.getSumOfSquaredWeightVector (
              self )
```

```
getSumOfSquaredWeightVector(FCLNeuron self) -> double
```

#### 4.8.3.23 getWeight()

```
"double" feedforward_closedloop_learning.FCLNeuron.getWeight (
              self,
             "int" _index )
```

```
getWeight(FCLNeuron self, int _index) -> double

Parameters
----------
_index: int
```

#### 4.8.3.24 getWeightDistanceFromInitialWeights()

```
"double" feedforward_closedloop_learning.FCLNeuron.getWeightDistanceFromInitialWeights (
              self )
```

```
getWeightDistanceFromInitialWeights(FCLNeuron self) -> double
```

#### 4.8.3.25 initWeights()

```
"void" feedforward_closedloop_learning.FCLNeuron.initWeights (
              self,
             * args )
```

```
initWeights(FCLNeuron self, double _max=1, int initBias=1, FCLNeuron::WeightInitMethod _wm=MAX_OUTPUT_RANDOM)

Parameters
----------
_max: double
initBias: int
_wm: enum FCLNeuron::WeightInitMethod
```

#### 4.8.3.26 normaliseWeights()

```
"void" feedforward_closedloop_learning.FCLNeuron.normaliseWeights (
              self,
             "double" norm )
```

```
normaliseWeights(FCLNeuron self, double norm)

Parameters
----------
norm: double
```

#### 4.8.3.27 saveInitialWeights()

```
"void" feedforward_closedloop_learning.FCLNeuron.saveInitialWeights (
              self )
```

```
saveInitialWeights(FCLNeuron self)
```

#### 4.8.3.28 setActivationFunction()

```
"void" feedforward_closedloop_learning.FCLNeuron.setActivationFunction (
              self,
             "FCLNeuron::ActivationFunction" _activationFunction )
```

```
setActivationFunction(FCLNeuron self, FCLNeuron::ActivationFunction _activationFunction)

Parameters
----------
_activationFunction: enum FCLNeuron::ActivationFunction
```

#### 4.8.3.29 setBias()

```
"void" feedforward_closedloop_learning.FCLNeuron.setBias (
              self,
             "double" _bias )
```

```
setBias(FCLNeuron self, double _bias)

Parameters
----------
_bias: double
```

#### 4.8.3.30 setBiasWeight()

```
"void" feedforward_closedloop_learning.FCLNeuron.setBiasWeight (
              self,
             "double" _biasweight )
```

```
setBiasWeight(FCLNeuron self, double _biasweight)

Parameters
----------
_biasweight: double
```

#### 4.8.3.31 setDebugInfo()

```
"void" feedforward_closedloop_learning.FCLNeuron.setDebugInfo (
              self,
             "int" _layerIndex,
             "int" _neuronIndex )
```

```
setDebugInfo(FCLNeuron self, int _layerIndex, int _neuronIndex)

Parameters
----------
_layerIndex: int
_neuronIndex: int
```

#### 4.8.3.32 setDecay()

```
"void" feedforward_closedloop_learning.FCLNeuron.setDecay (
              self,
             "double" _decay )
```

```
setDecay(FCLNeuron self, double _decay)

Parameters
----------
_decay: double
```

#### 4.8.3.33 setError()

```
"void" feedforward_closedloop_learning.FCLNeuron.setError (
              self,
             "double" _error )
```

```
setError(FCLNeuron self, double _error)

Parameters
----------
_error: double
```

#### 4.8.3.34 setGeometry()

```
"void" feedforward_closedloop_learning.FCLNeuron.setGeometry (
              self,
             "int" _width,
             "int" _height )
```

```
setGeometry(FCLNeuron self, int _width, int _height)

Parameters
----------
_width: int
_height: int
```

#### 4.8.3.35 setInput()

```
"void" feedforward_closedloop_learning.FCLNeuron.setInput (
              self,
             "int" _index,
             "double" _value )
```

```
setInput(FCLNeuron self, int _index, double _value)

Parameters
----------
_index: int
_value: double
```

#### 4.8.3.36 setLearningRate()

```
"void" feedforward_closedloop_learning.FCLNeuron.setLearningRate (
              self,
             "double" _learningrate )
```

```
setLearningRate(FCLNeuron self, double _learningrate)

Parameters
----------
_learningrate: double
```

#### 4.8.3.37 setMask()

```
"void" feedforward_closedloop_learning.FCLNeuron.setMask (
             self,
            * args )
```

```
setMask(FCLNeuron self, int x, int y, unsigned char c)

Parameters
----------
x: int
y: int
c: unsigned char

setMask(FCLNeuron self, unsigned char c)

Parameters
----------
c: unsigned char
```

#### 4.8.3.38 setMomentum()

```
"void" feedforward_closedloop_learning.FCLNeuron.setMomentum (
             self,
            "double" _momentum )
```

```
setMomentum(FCLNeuron self, double _momentum)

Parameters
----------
_momentum: double
```

#### 4.8.3.39 setStep()

```
"void" feedforward_closedloop_learning.FCLNeuron.setStep (
             self,
            "long" _step )
```

```
setStep(FCLNeuron self, long _step)

Parameters
----------
_step: long
```

#### 4.8.3.40 setWeight()

```
"void" feedforward_closedloop_learning.FCLNeuron.setWeight (
              self,
             "int" _index,
             "double" _weight )
```

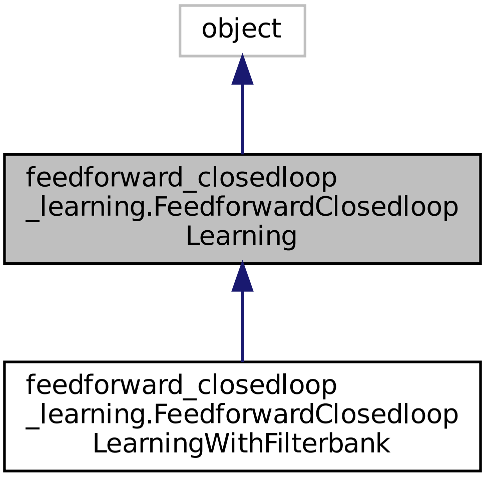
```
setWeight(FCLNeuron self, int _index, double _weight)

Parameters
----------
_index: int
_weight: double
```

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

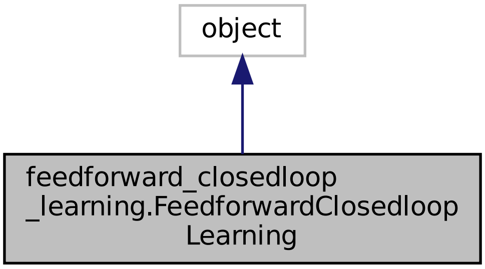
- feedforward_closedloop_learning.py

## 4.9 feedforward_closedloop_learning.FeedforwardClosedloopLearning Class Reference

Inheritance diagram for feedforward_closedloop_learning.FeedforwardClosedloopLearning:



Collaboration diagram for feedforward_closedloop_learning.FeedforwardClosedloopLearning:



## Public Member Functions

- def __init__ (self, "int const" num_input, "IntVector" num_of_neurons_per_layer)
- "void" doStep (self, "DoubleVector" input, "DoubleVector" error)
- "double" getOutput (self, "int" index)
- "void" setLearningRate (self, "double" learningRate)
- "void" setLearningRateDiscountFactor (self, "double" _learningRateDiscountFactor)
- "void" setDecay (self, "double" decay)
- "void" setMomentum (self, "double" momentum)
- "void" setActivationFunction (self, "FCLNeuron::ActivationFunction" _activationFunction)
- "void" initWeights (self, ∗args)
- "void" seedRandom (self, "int" s)
- "void" setBias (self, "double" _bias)
- "int" getNumLayers (self)
- "FCLLayer ∗" getLayer (self, "unsigned int" i)
- "FCLLayer ∗" getOutputLayer (self)
- "int" getNumInputs (self)
- "FCLLayer ∗∗" getLayers (self)
- "bool" saveModel (self, "char const ∗" name)
- "bool" loadModel (self, "char const ∗" name)

## Properties

- **thisown** = property(lambda x: x.this.own(), lambda x, v: x.this.own(v), doc="The membership flag")

### 4.9.1 Detailed Description

```
Proxy of C++ FeedforwardClosedloopLearning class.
```

### 4.9.2 Constructor & Destructor Documentation

#### 4.9.2.1 __init__()

```
def feedforward_closedloop_learning.FeedforwardClosedloopLearning.__init__ (
              self,
             "int const" num_input,
             "IntVector" num_of_neurons_per_layer )
```

```
__init__(FeedforwardClosedloopLearning self, int const num_input, IntVector num_of_neurons_per_layer) -> Feedf

Parameters
----------
num_input: int const
num_of_neurons_per_layer: std::vector< int,std::allocator< int > > const &
```

### 4.9.3 Member Function Documentation

#### 4.9.3.1 doStep()

```
"void" feedforward_closedloop_learning.FeedforwardClosedloopLearning.doStep (
              self,
             "DoubleVector" input,
             "DoubleVector" error )
```

```
doStep(FeedforwardClosedloopLearning self, DoubleVector input, DoubleVector error)

Parameters
----------
input: std::vector< double,std::allocator< double > > const &
error: std::vector< double,std::allocator< double > > const &
```

Reimplemented in feedforward_closedloop_learning.FeedforwardClosedloopLearningWithFilterbank.

#### 4.9.3.2 getLayer()

```
"FCLLayer *" feedforward_closedloop_learning.FeedforwardClosedloopLearning.getLayer (
              self,
             "unsigned int" i )
```

```
getLayer(FeedforwardClosedloopLearning self, unsigned int i) -> FCLLayer

Parameters
----------
i: unsigned int
```

#### 4.9.3.3 getLayers()

```
"FCLLayer **" feedforward_closedloop_learning.FeedforwardClosedloopLearning.getLayers (
              self )
```

```
getLayers(FeedforwardClosedloopLearning self) -> FCLLayer **
```

#### 4.9.3.4 getNumInputs()

```
"int" feedforward_closedloop_learning.FeedforwardClosedloopLearning.getNumInputs (
              self )
```

```
getNumInputs(FeedforwardClosedloopLearning self) -> int
```

#### 4.9.3.5 getNumLayers()

```
"int" feedforward_closedloop_learning.FeedforwardClosedloopLearning.getNumLayers (
              self )
```

```
getNumLayers(FeedforwardClosedloopLearning self) -> int
```

#### 4.9.3.6 getOutput()

```
"double" feedforward_closedloop_learning.FeedforwardClosedloopLearning.getOutput (
              self,
             "int" index )
```

```
getOutput(FeedforwardClosedloopLearning self, int index) -> double

Parameters
----------
index: int
```

#### 4.9.3.7 getOutputLayer()

```
"FCLLayer *" feedforward_closedloop_learning.FeedforwardClosedloopLearning.getOutputLayer (
              self )
```

```
getOutputLayer(FeedforwardClosedloopLearning self) -> FCLLayer
```

#### 4.9.3.8 initWeights()

```
"void" feedforward_closedloop_learning.FeedforwardClosedloopLearning.initWeights (
              self,
             * args )
```

```
initWeights(FeedforwardClosedloopLearning self, double max=0.001, int initBias=1, FCLNeuron::WeightInitMethod

Parameters
----------
max: double
initBias: int
weightInitMethod: enum FCLNeuron::WeightInitMethod
```

#### 4.9.3.9 loadModel()

```
"bool" feedforward_closedloop_learning.FeedforwardClosedloopLearning.loadModel (
              self,
             "char const *" name )
```

```
loadModel(FeedforwardClosedloopLearning self, char const * name) -> bool

Parameters
----------
name: char const *
```

#### 4.9.3.10 saveModel()

```
"bool" feedforward_closedloop_learning.FeedforwardClosedloopLearning.saveModel (
              self,
             "char const *" name )
```

```
saveModel(FeedforwardClosedloopLearning self, char const * name) -> bool

Parameters
----------
name: char const *
```

#### 4.9.3.11 seedRandom()

```
"void" feedforward_closedloop_learning.FeedforwardClosedloopLearning.seedRandom (
              self,
             "int" s )
```

```
seedRandom(FeedforwardClosedloopLearning self, int s)

Parameters
----------
s: int
```

#### 4.9.3.12 setActivationFunction()

```
"void" feedforward_closedloop_learning.FeedforwardClosedloopLearning.setActivationFunction (
              self,
             "FCLNeuron::ActivationFunction" _activationFunction )
```

```
setActivationFunction(FeedforwardClosedloopLearning self, FCLNeuron::ActivationFunction _activationFunction)

Parameters
----------
_activationFunction: enum FCLNeuron::ActivationFunction
```

#### 4.9.3.13 setBias()

```
"void" feedforward_closedloop_learning.FeedforwardClosedloopLearning.setBias (
              self,
             "double" _bias )
```

```
setBias(FeedforwardClosedloopLearning self, double _bias)

Parameters
----------
_bias: double
```

#### 4.9.3.14 setDecay()

```
"void" feedforward_closedloop_learning.FeedforwardClosedloopLearning.setDecay (
              self,
             "double" decay )
```

```
setDecay(FeedforwardClosedloopLearning self, double decay)

Parameters
----------
decay: double
```

#### 4.9.3.15 setLearningRate()

```
"void" feedforward_closedloop_learning.FeedforwardClosedloopLearning.setLearningRate (
              self,
             "double" learningRate )
```

```
setLearningRate(FeedforwardClosedloopLearning self, double learningRate)

Parameters
----------
learningRate: double
```

#### 4.9.3.16 setLearningRateDiscountFactor()

```
"void" feedforward_closedloop_learning.FeedforwardClosedloopLearning.setLearningRateDiscount←
Factor (
              self,
             "double" _learningRateDiscountFactor )
```

```
setLearningRateDiscountFactor(FeedforwardClosedloopLearning self, double _learningRateDiscountFactor)

Parameters
----------
_learningRateDiscountFactor: double
```

#### 4.9.3.17 setMomentum()

```
"void" feedforward_closedloop_learning.FeedforwardClosedloopLearning.setMomentum (
              self,
             "double" momentum )
```

```
setMomentum(FeedforwardClosedloopLearning self, double momentum)

Parameters
----------
momentum: double
```

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

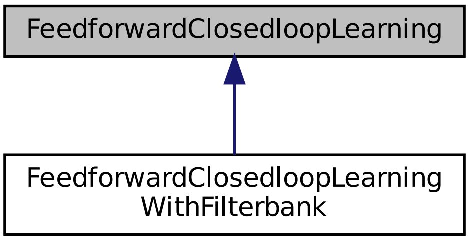
- feedforward_closedloop_learning.py

## 4.10 FeedforwardClosedloopLearning Class Reference

Main class of Feedforward Closed Loop Learning.

`#include <fcl.h>`

Inheritance diagram for FeedforwardClosedloopLearning:


FeedforwardClosedloopLearning
FeedforwardClosedloopLearning WithFilterbank

## Public Member Functions

- FeedforwardClosedloopLearning (const int num_input, const std::vector< int > &num_of_neurons_per_↩ layer)

  *Constructor: FCL without any filters.*
- ∼FeedforwardClosedloopLearning ()

  *Destructor De-allocated any memory.*
- void doStep (const std::vector< double > &input, const std::vector< double > &error)

  *Performs the simulation step.*
- double getOutput (int index)

  *Gets the output from one of the output neurons.*
- void setLearningRate (double learningRate)

  *Sets globally the learning rate.*
- void setLearningRateDiscountFactor (double _learningRateDiscountFactor)

  *Sets how the learnign rate increases or decreases from layer to layer.*
- void setDecay (double decay)

  *Sets a typical weight decay scaled with the learning rate.*
- void setMomentum (double momentum)

  *Sets the global momentum for all layers.*
- void setActivationFunction (FCLNeuron::ActivationFunction _activationFunction)

  *Sets the activation function of the Neuron.*
- void initWeights (double max=0.001, int initBias=1, FCLNeuron::WeightInitMethod weightInitMethod=FCLNeuron↩ ::MAX_OUTPUT_RANDOM)

  *Inits the weights in all layers.*
- void seedRandom (int s)

  *Seeds the random number generator.*
- void setBias (double _bias)

  *Sets globally the bias.*
- int getNumLayers ()

  *Gets the total number of layers.*
- FCLLayer ∗ getLayer (unsigned i)

  *Gets a pointer to a layer.*
- FCLLayer ∗ getOutputLayer ()

  *Gets the output layer.*
- int getNumInputs ()

  *Gets the number of inputs.*
- FCLLayer ∗∗ getLayers ()

  *Returns all Layers.*
- bool saveModel (const char ∗name)

  *Saves the whole network.*
- bool loadModel (const char ∗name)

  *Loads the while network.*

## 4.10.1 Detailed Description

Main class of Feedforward Closed Loop Learning.

Create an instance of this class to do the learning. It will create the whole network with an input layer, layers and an output layer. Learning is done iterative by first setting the input values and errors and then calling doStep().

(C) 2017,2018-2022, Bernd Porr `bernd@glasgowneuro.tech` (C) 2017,2018, Paul Miller `paul@glasgowneuro.↩ tech`

GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

## 4.10.2 Constructor & Destructor Documentation

### 4.10.2.1 FeedforwardClosedloopLearning()

```
FeedforwardClosedloopLearning::FeedforwardClosedloopLearning (
             const int num_input,
             const std::vector< int > & num_of_neurons_per_layer )
```

Constructor: FCL without any filters.

**Parameters**

| | |
|---|---|
| *num_of_inputs* | Number of inputs in the input layer |
| *num_of_neurons_per_layer_array* | Number of neurons in each layer |
| *_num_layers* | Number of layer (needs to match with array above) |

## 4.10.3 Member Function Documentation

### 4.10.3.1 doStep()

```
void FeedforwardClosedloopLearning::doStep (
             const std::vector< double > & input,
             const std::vector< double > & error )
```

Performs the simulation step.

**Parameters**

| | |
|---|---|
| *input* | Array with the input values |
| *error* | Array of the error signals |

### 4.10.3.2 getLayer()

```
FCLLayer* FeedforwardClosedloopLearning::getLayer (
             unsigned i ) [inline]
```

Gets a pointer to a layer.

**Parameters**

| | |
|---|---|
| *i* | Index of the layer. |

**Returns**

A pointer to a layer class.

#### 4.10.3.3 getLayers()

```
FCLLayer** FeedforwardClosedloopLearning::getLayers ( )  [inline]
```

Returns all Layers.

**Returns**

Returns a two dimensional array of all layers.

#### 4.10.3.4 getNumInputs()

```
int FeedforwardClosedloopLearning::getNumInputs ( )  [inline]
```

Gets the number of inputs.

**Returns**

The number of inputs

#### 4.10.3.5 getNumLayers()

```
int FeedforwardClosedloopLearning::getNumLayers ( )  [inline]
```

Gets the total number of layers.

**Returns**

The total number of all layers.

#### 4.10.3.6 getOutput()

```
double FeedforwardClosedloopLearning::getOutput (
             int index )  [inline]
```

Gets the output from one of the output neurons.

**Parameters**

| | |
|---|---|
| *index* | The index number of the output neuron. |

**Returns**

The output value of the output neuron.

### 4.10.3.7 getOutputLayer()

```
FCLLayer* FeedforwardClosedloopLearning::getOutputLayer ( )  [inline]
```

Gets the output layer.

**Returns**

A pointer to the output layer which is also a Layer class.

### 4.10.3.8 initWeights()

```
void FeedforwardClosedloopLearning::initWeights (
             double max = 0.001,
             int initBias = 1,
             FCLNeuron::WeightInitMethod weightInitMethod = FCLNeuron::MAX_OUTPUT_RANDOM )
```

Inits the weights in all layers.

**Parameters**

| | |
|---|---|
| *max* | Maximum value of the weights. |
| *initBias* | If the bias also should be initialised. |
| *weightInitMethod* | See Neuron::WeightInitMethod for the options. |

### 4.10.3.9 loadModel()

```
bool FeedforwardClosedloopLearning::loadModel (
             const char * name )
```

Loads the while network.

**Parameters**

| | |
|---|---|
| *name* | filename |

#### 4.10.3.10 saveModel()

```
bool FeedforwardClosedloopLearning::saveModel (
             const char * name )
```

Saves the whole network.

**Parameters**

| | |
|---|---|
| *name* | filename |

#### 4.10.3.11 seedRandom()

```
void FeedforwardClosedloopLearning::seedRandom (
             int s )  [inline]
```

Seeds the random number generator.

**Parameters**

| | |
|---|---|
| *s* | An arbitratry number. |

#### 4.10.3.12 setBias()

```
void FeedforwardClosedloopLearning::setBias (
             double _bias )
```

Sets globally the bias.

**Parameters**

| | |
|---|---|
| *_bias* | Sets globally the bias input to all neurons. |

#### 4.10.3.13 setDecay()

```
void FeedforwardClosedloopLearning::setDecay (
             double decay )
```

Sets a typical weight decay scaled with the learning rate.

**Parameters**

| | |
|---|---|
| *decay* | The larger the faster the decay. |

#### 4.10.3.14 setLearningRate()

```
void FeedforwardClosedloopLearning::setLearningRate (
             double learningRate )
```

Sets globally the learning rate.

**Parameters**

| | |
|---|---|
| *learningRate* | Sets the learning rate for all layers and neurons. |

#### 4.10.3.15 setLearningRateDiscountFactor()

```
void FeedforwardClosedloopLearning::setLearningRateDiscountFactor (
             double _learningRateDiscountFactor )  [inline]
```

Sets how the learnign rate increases or decreases from layer to layer.

**Parameters**

| | |
|---|---|
| *_learningRateDiscountFactor* | A factor of >1 means higher learning rate in deeper layers. |

#### 4.10.3.16 setMomentum()

```
void FeedforwardClosedloopLearning::setMomentum (
             double momentum )
```

Sets the global momentum for all layers.

**Parameters**

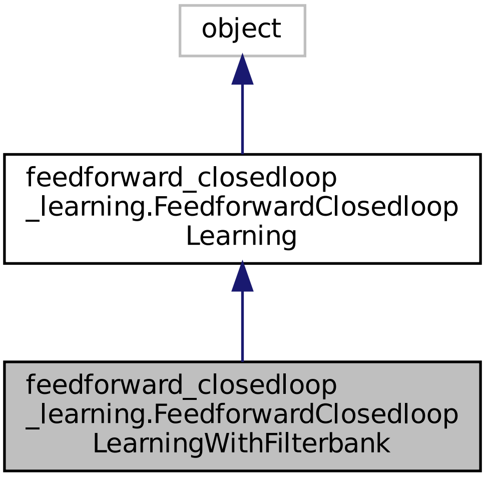
| | |
|---|---|
| *momentum* | Defines the intertia of the weight change over time. |

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

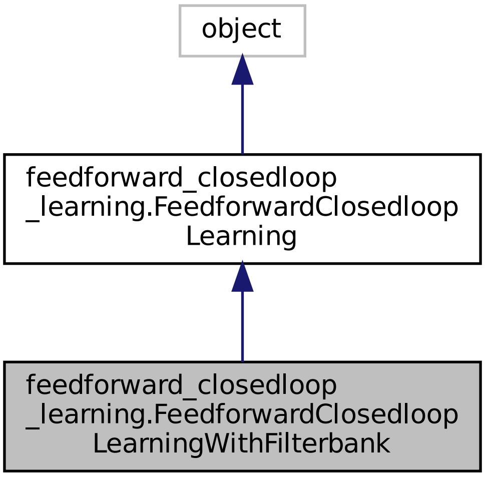
- fcl.h

## 4.11 feedforward_closedloop_learning.FeedforwardClosedloop↩ LearningWithFilterbank Class Reference

Inheritance diagram for feedforward_closedloop_learning.FeedforwardClosedloopLearningWithFilterbank:


object
feedforward_closedloop _learning.FeedforwardClosedloop Learning
feedforward_closedloop _learning.FeedforwardClosedloop LearningWithFilterbank

Collaboration diagram for feedforward_closedloop_learning.FeedforwardClosedloopLearningWithFilterbank:


object
feedforward_closedloop _learning.FeedforwardClosedloop Learning
feedforward_closedloop _learning.FeedforwardClosedloop LearningWithFilterbank

## Public Member Functions

- def __init__ (self, "int const" num_of_inputs, "IntVector" num_of_neurons_per_layer, "int const" num_filters↩ Input, "double const" minT, "double const" maxT)
- "void" doStep (self, "DoubleVector" input, "DoubleVector" error)
- "double" getFilterOutput (self, "int" inputIdx, "int" filterIdx)
- "int" getNFiltersPerInput (self)

## Properties

- **thisown** = property(lambda x: x.this.own(), lambda x, v: x.this.own(v), doc="The membership flag")

### 4.11.1 Detailed Description

```
Proxy of C++ FeedforwardClosedloopLearningWithFilterbank class.
```

### 4.11.2 Constructor & Destructor Documentation

#### 4.11.2.1 __init__()

```
def feedforward_closedloop_learning.FeedforwardClosedloopLearningWithFilterbank.__init__ (
              self,
             "int const" num_of_inputs,
             "IntVector" num_of_neurons_per_layer,
             "int const" num_filtersInput,
             "double const" minT,
             "double const" maxT )
```

```
__init__(FeedforwardClosedloopLearningWithFilterbank self, int const num_of_inputs, IntVector num_of_neurons_p

Parameters
----------
num_of_inputs: int const
num_of_neurons_per_layer: std::vector< int,std::allocator< int > > const &
num_filtersInput: int const
minT: double const
maxT: double const
```

### 4.11.3 Member Function Documentation

#### 4.11.3.1 doStep()

```
"void" feedforward_closedloop_learning.FeedforwardClosedloopLearningWithFilterbank.doStep (
              self,
             "DoubleVector" input,
             "DoubleVector" error )
```

```
doStep(FeedforwardClosedloopLearningWithFilterbank self, DoubleVector input, DoubleVector error)

Parameters
----------
input: std::vector< double,std::allocator< double > > const &
error: std::vector< double,std::allocator< double > > const &
```

Reimplemented from feedforward_closedloop_learning.FeedforwardClosedloopLearning.

#### 4.11.3.2 getFilterOutput()

```
"double" feedforward_closedloop_learning.FeedforwardClosedloopLearningWithFilterbank.get↩
FilterOutput (
              self,
             "int" inputIdx,
             "int" filterIdx )
```

```
getFilterOutput(FeedforwardClosedloopLearningWithFilterbank self, int inputIdx, int filterIdx) -> double

Parameters
----------
inputIdx: int
filterIdx: int
```

#### 4.11.3.3 getNFiltersPerInput()

```
"int" feedforward_closedloop_learning.FeedforwardClosedloopLearningWithFilterbank.getNFilters↩
PerInput (
              self )
```

```
getNFiltersPerInput(FeedforwardClosedloopLearningWithFilterbank self) -> int
```

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

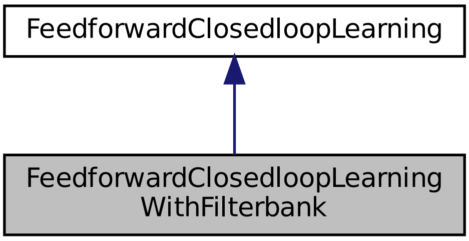
- feedforward_closedloop_learning.py

## 4.12 FeedforwardClosedloopLearningWithFilterbank Class Reference

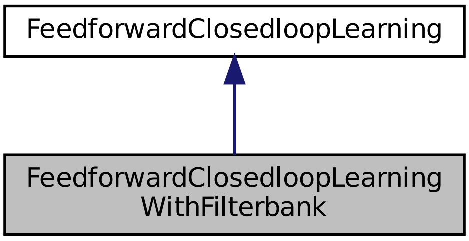
Derived classes of the FeedforwardClosedloopLearning class for special functionality.

`#include <fcl_util.h>`

Inheritance diagram for FeedforwardClosedloopLearningWithFilterbank:


FeedforwardClosedloopLearning
FeedforwardClosedloopLearning WithFilterbank

Collaboration diagram for FeedforwardClosedloopLearningWithFilterbank:


FeedforwardClosedloopLearning
FeedforwardClosedloopLearning WithFilterbank

## Public Member Functions

- FeedforwardClosedloopLearningWithFilterbank (const int num_of_inputs, const std::vector< int > &num_↩ of_neurons_per_layer, const int num_filtersInput, const double minT, const double maxT)

  *FeedforwardClosedloopLearning with Filterbank at each input.*
- ∼FeedforwardClosedloopLearningWithFilterbank ()

  *Destructor.*
- void doStep (const std::vector< double > &input, const std::vector< double > &error)

  *Performs the simulation step.*
- double **getFilterOutput** (int inputIdx, int filterIdx)
- int **getNFiltersPerInput** ()

## 4.12.1 Detailed Description

Derived classes of the FeedforwardClosedloopLearning class for special functionality.

## 4.12.2 Constructor & Destructor Documentation

### 4.12.2.1 FeedforwardClosedloopLearningWithFilterbank()

```
FeedforwardClosedloopLearningWithFilterbank::FeedforwardClosedloopLearningWithFilterbank (
            const int num_of_inputs,
            const std::vector< int > & num_of_neurons_per_layer,
            const int num_filtersInput,
            const double minT,
            const double maxT )
```

FeedforwardClosedloopLearning with Filterbank at each input.

Constructor: FCL with a filter bank at the input Every input feeds internally into a has a filter bank of num_filtersInput filters. This allows for a temporal distribution of the inputs.

**Parameters**

| | |
|---|---|
| *num_of_inputs* | Number of inputs in the input layer |
| *num_of_neurons_per_layer_array* | Number of neurons in each layer |
| *num_filtersInput* | Number of filters at the input layer, 0 = no filterbank |
| *num_filters* | Number of filters in the hiddel layers (usually zero) |
| *_minT* | Minimum/first temporal duration of the 1st filter |
| *_maxT* | Maximum/last temporal duration of the last filter |

## 4.12.3 Member Function Documentation

#### 4.12.3.1 doStep()

```
void FeedforwardClosedloopLearningWithFilterbank::doStep (
            const std::vector< double > & input,
            const std::vector< double > & error )
```

Performs the simulation step.

**Parameters**

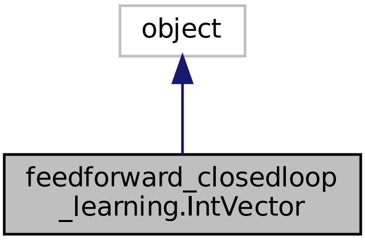
| | |
|---|---|
| *input* | Array with the input values |
| *error* | Array of the error signals |

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

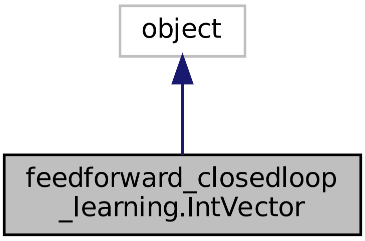
- fcl_util.h

## 4.13 feedforward_closedloop_learning.IntVector Class Reference

Inheritance diagram for feedforward_closedloop_learning.IntVector:



Collaboration diagram for feedforward_closedloop_learning.IntVector:



## Public Member Functions

- "swig::SwigPyIterator ∗" **iterator** (self)
- def **__iter__** (self)
- "bool" **__nonzero__** (self)
- "bool" **__bool__** (self)
- "std::vector< int >::size_type" **__len__** (self)
- "std::vector< int,std::allocator< int > > ∗" **__getslice__** (self, "std::vector< int >::difference_type" i, "std↩ ::vector< int >::difference_type" j)
- "void" **__setslice__** (self, ∗args)
- "void" **__delslice__** (self, "std::vector< int >::difference_type" i, "std::vector< int >::difference_type" j)
- "void" **__delitem__** (self, ∗args)
- "std::vector< int >::value_type const &" **__getitem__** (self, ∗args)
- "void" **__setitem__** (self, ∗args)
- "std::vector< int >::value_type" **pop** (self)
- "void" **append** (self, "std::vector< int >::value_type const &" x)
- "bool" **empty** (self)
- "std::vector< int >::size_type" **size** (self)
- "void" **swap** (self, "IntVector" v)
- "std::vector< int >::iterator" **begin** (self)
- "std::vector< int >::iterator" **end** (self)
- "std::vector< int >::reverse_iterator" **rbegin** (self)
- "std::vector< int >::reverse_iterator" **rend** (self)
- "void" **clear** (self)
- "std::vector< int >::allocator_type" **get_allocator** (self)
- "void" **pop_back** (self)
- "std::vector< int >::iterator" **erase** (self, ∗args)
- def **__init__** (self, ∗args)
- "void" **push_back** (self, "std::vector< int >::value_type const &" x)
- "std::vector< int >::value_type const &" **front** (self)
- "std::vector< int >::value_type const &" **back** (self)
- "void" **assign** (self, "std::vector< int >::size_type" n, "std::vector< int >::value_type const &" x)
- "void" **resize** (self, ∗args)
- "void" **insert** (self, ∗args)
- "void" **reserve** (self, "std::vector< int >::size_type" n)
- "std::vector< int >::size_type" **capacity** (self)

## Properties

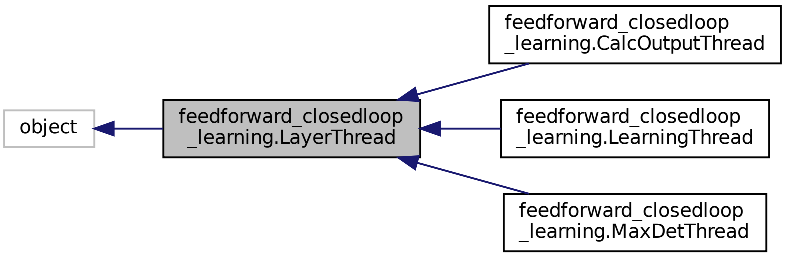
- **thisown** = property(lambda x: x.this.own(), lambda x, v: x.this.own(v), doc="The membership flag")

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

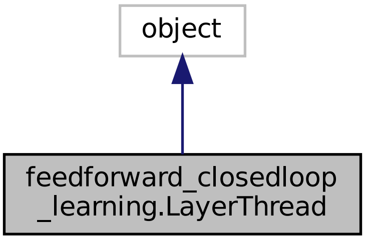
- feedforward_closedloop_learning.py

## 4.14 feedforward_closedloop_learning.LayerThread Class Reference

Inheritance diagram for feedforward_closedloop_learning.LayerThread:


object
feedforward_closedloop _learning.LayerThread
feedforward_closedloop _learning.CalcOutputThread
feedforward_closedloop _learning.LearningThread
feedforward_closedloop _learning.MaxDetThread

Collaboration diagram for feedforward_closedloop_learning.LayerThread:



### Public Member Functions

- def **__init__** (self, ∗args, ∗∗kwargs)
- "void" addNeuron (self, "FCLNeuron" neuron)
- "void" start (self)
- "void" join (self)
- "void" run (self)

### Properties

- **thisown** = property(lambda x: x.this.own(), lambda x, v: x.this.own(v), doc="The membership flag")

### 4.14.1 Detailed Description

```
Proxy of C++ LayerThread class.
```

## 4.14.2 Member Function Documentation

### 4.14.2.1 addNeuron()

```
"void" feedforward_closedloop_learning.LayerThread.addNeuron (
              self,
             "FCLNeuron" neuron )
```

```
addNeuron(LayerThread self, FCLNeuron neuron)

Parameters
----------
neuron: FCLNeuron *
```

### 4.14.2.2 join()

```
"void" feedforward_closedloop_learning.LayerThread.join (
              self )
```

```
join(LayerThread self)
```

### 4.14.2.3 run()

```
"void" feedforward_closedloop_learning.LayerThread.run (
              self )
```

```
run(LayerThread self)
```

### 4.14.2.4 start()

```
"void" feedforward_closedloop_learning.LayerThread.start (
              self )
```

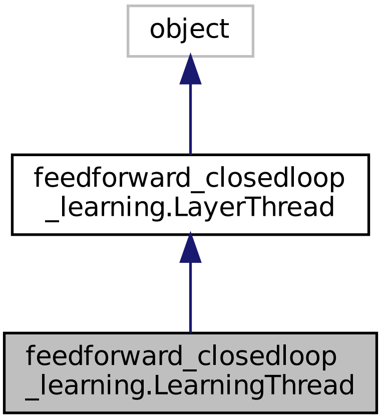
```
start(LayerThread self)
```

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

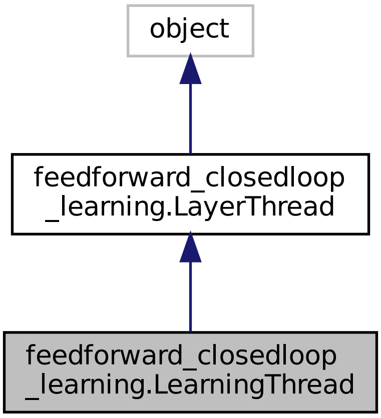
- feedforward_closedloop_learning.py

## 4.15 feedforward_closedloop_learning.LearningThread Class Reference

Inheritance diagram for feedforward_closedloop_learning.LearningThread:


object
feedforward_closedloop _learning.LayerThread
feedforward_closedloop _learning.LearningThread

Collaboration diagram for feedforward_closedloop_learning.LearningThread:


object
feedforward_closedloop _learning.LayerThread
feedforward_closedloop _learning.LearningThread

### Public Member Functions

- def **__init__** (self, ∗args, ∗∗kwargs)

### Properties

- **thisown** = property(lambda x: x.this.own(), lambda x, v: x.this.own(v), doc="The membership flag")

### 4.15.1 Detailed Description

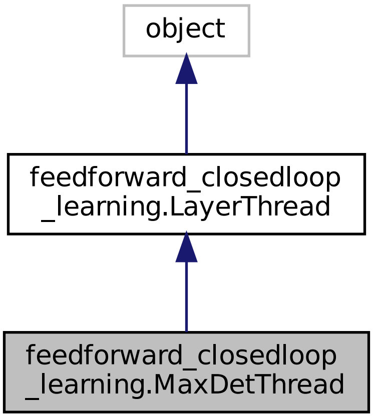
```
Proxy of C++ LearningThread class.
```

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

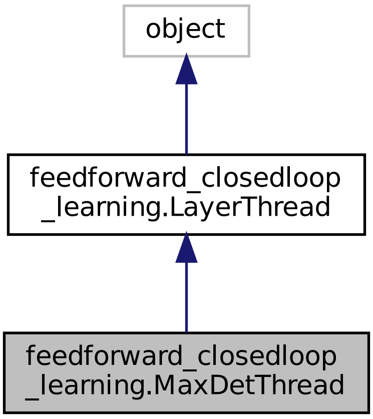
- feedforward_closedloop_learning.py

## 4.16 feedforward_closedloop_learning.MaxDetThread Class Reference

Inheritance diagram for feedforward_closedloop_learning.MaxDetThread:


object
feedforward_closedloop_learning.LayerThread
feedforward_closedloop_learning.MaxDetThread

Collaboration diagram for feedforward_closedloop_learning.MaxDetThread:


object
feedforward_closedloop_learning.LayerThread
feedforward_closedloop_learning.MaxDetThread

## Public Member Functions

- def **__init__** (self, ∗args, ∗∗kwargs)

## Properties

- **thisown** = property(lambda x: x.this.own(), lambda x, v: x.this.own(v), doc="The membership flag")

### 4.16.1 Detailed Description

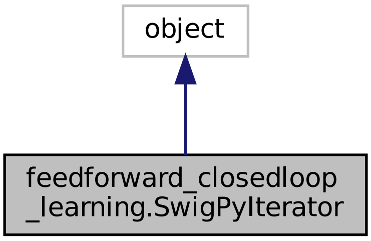
```
Proxy of C++ MaxDetThread class.
```

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

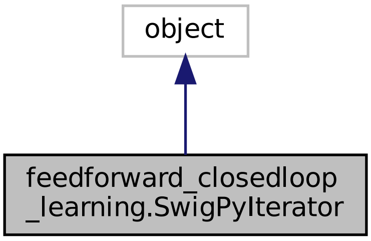
- feedforward_closedloop_learning.py

## 4.17 feedforward_closedloop_learning.SwigPyIterator Class Reference

Inheritance diagram for feedforward_closedloop_learning.SwigPyIterator:



Collaboration diagram for feedforward_closedloop_learning.SwigPyIterator:



## Public Member Functions

- def **__init__** (self, ∗args, ∗∗kwargs)
- "PyObject ∗" **value** (self)
- "swig::SwigPyIterator ∗" **incr** (self, "size_t" n=1)
- "swig::SwigPyIterator ∗" **decr** (self, "size_t" n=1)
- "ptrdiff_t" **distance** (self, "SwigPyIterator" x)
- "bool" **equal** (self, "SwigPyIterator" x)
- "swig::SwigPyIterator ∗" **copy** (self)
- "PyObject ∗" **next** (self)
- "PyObject ∗" **__next__** (self)
- "PyObject ∗" **previous** (self)
- "swig::SwigPyIterator ∗" **advance** (self, "ptrdiff_t" n)
- "bool" **__eq__** (self, "SwigPyIterator" x)
- "bool" **__ne__** (self, "SwigPyIterator" x)
- "swig::SwigPyIterator &" **__iadd__** (self, "ptrdiff_t" n)
- "swig::SwigPyIterator &" **__isub__** (self, "ptrdiff_t" n)
- "swig::SwigPyIterator ∗" **__add__** (self, "ptrdiff_t" n)
- "ptrdiff_t" **__sub__** (self, ∗args)
- def **__iter__** (self)

## Properties

- **thisown** = property(lambda x: x.this.own(), lambda x, v: x.this.own(v), doc="The membership flag")

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

- feedforward_closedloop_learning.py

# Index