Flexible Closed-Loop Deep Learning (CLDL) platform

Generated by Doxygen 1.8.19

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 Layer Class Reference

```
#include <Layer.h>
```

Public Types

- enum whichGradient { **exploding** = 0, **average** = 1, **vanishing** = 2 }

Public Member Functions

- Layer (int _nNeurons, int _nInputs)
- ∼Layer ()
- void initLayer (int _layerIndex, Neuron::weightInitMethod _wim, Neuron::biasInitMethod _bim, Neuron::actMethod _am)
- void setlearningRate (double _learningRate)
- void setInputs (const double ∗_inputs)
- void propInputs (int _index, double _value)
- void calcOutputs ()
- void setForwardError (double _leadForwardError)
- void propErrorForward (int _index, double _value)
- void calcForwardError ()
- double getForwardError (int _neuronIndex)
- void setBackwardError (double _leadError)
- void propErrorBackward (int _neuronIndex, double _nextSum)
- double getBackwardError (int _neuronIndex)
- void setMidError (double _leadMidError)
- void calcMidError ()
- double getMidError (int _neuronIndex)
- void propMidErrorForward (int _index, double _value)
- void propMidErrorBackward (int _neuronIndex, double _nextSum)
- double getGradient (Neuron::whichError _whichError, whichGradient _whichGradient)
- void setErrorCoeff (double _globalCoeff, double _backwardsCoeff, double _midCoeff, double _forwardCoeff, double _localCoeff, double _echoCoeff)
- void updateWeights ()
- void setGlobalError (double _globalError)

- void setLocalError (double _leadLocalError)
- void propGlobalErrorBackwardLocally (int _neuronIndex, double _nextSum)
- double getLocalError (int _neuronIndex)
- void setEchoError (double _clError)
- void echoErrorBackward (int _neuronIndex, double _nextSum)
- double getEchoError (int _neuronIndex)
- void echoErrorForward (int _index, double _value)
- void calcEchoError ()
- Neuron ∗ getNeuron (int _neuronIndex)
- int getnNeurons ()
- double getOutput (int _neuronIndex)
- double getSumOutput (int _neuronIndex)
- double getWeights (int _neuronIndex, int _weightIndex)
- double getWeightChange ()
- double getWeightDistance ()
- double getGlobalError (int _neuronIndex)
- double getInitWeight (int _neuronIndex, int _weightIndex)
- void saveWeights ()
- void snapWeights ()
- void printLayer ()

2.1.1 Detailed Description

This is the class for creating layers that are contained inside the Net class. The Layer instances in turn contain neurons.

2.1.2 Member Enumeration Documentation

2.1.2.1 whichGradient

```
enum Layer::whichGradient
```

Options for what gradient of a chosen error to monitor

2.1.3 Constructor & Destructor Documentation

2.1.3.1 Layer()

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

Constructor for Layer: it initialises the neurons internally.

Parameters

_nNeurons	Total number of neurons in the layer
_nInputs	Total number of inputs to that layer

2.1.3.2 ∼Layer()

```
Layer::∼Layer ( )
```

Destructor De-allocated any memory

2.1.4 Member Function Documentation

2.1.4.1 calcEchoError()

```
void Layer::calcEchoError ( )
```

Demands that all neurons calculate their resonating error

2.1.4.2 calcForwardError()

```
void Layer::calcForwardError ( )
```

calculates the forward error by doing a weighed sum of forward errors and the weights

2.1.4.3 calcMidError()

```
void Layer::calcMidError ( )
```

calculates the error to be propagated bilaterally

2.1.4.4 calcOutputs()

```
void Layer::calcOutputs ( )
```

Demands that all neurons in this layer calculate their output

2.1.4.5 echoErrorBackward()

```
void Layer::echoErrorBackward (
             int _neuronIndex,
             double _nextSum )
```

Sets the resonating error for a specific neuron

Parameters

_neuronIndex	Index of the neurons receiving the error
_nextSum	The weighted sum of propagating errors

2.1.4.6 echoErrorForward()

```
void Layer::echoErrorForward (
             int _index,
             double _value )
```

Sets the resonating error for a specific neuron

Parameters

_index	the index of the incoming error
_value	The value of the incoming error

2.1.4.7 getBackwardError()

```
double Layer::getBackwardError (
             int _neuronIndex )
```

Allows for accessing the error that propagates backward in the network

Parameters

_neuronIndex	The index from which the error is requested

Returns

Returns the error of the chosen neuron

2.1.4.8 getEchoError()

```
double Layer::getEchoError (
             int _neuronIndex )
```

Allows for accessing the resonating error of a specific neuron

Parameters

_neuronIndex	The index of the neuron to reuquest the error form.

Returns

Returns the resonating error of the neuron

2.1.4.9 getForwardError()

```
double Layer::getForwardError (
             int _neuronIndex )
```

Allows for accessing the forward error of a specific neuron.

Parameters

_neuronIndex	Index of the neuron to request the error from

Returns

Returns the forward error from the chosen neuron

2.1.4.10 getGlobalError()

```
double Layer::getGlobalError (
             int _neuronIndex )
```

Reports the global error that is assigned to a specific neuron in this layer

Parameters

_neuronIndex	the neuron index

Returns

the value of the global error

2.1.4.11 getGradient()

```
double Layer::getGradient (
             Neuron::whichError _whichError,
             whichGradient _whichGradient )
```

It provides a measure of the magnitude of the error in this layer to alarm for vanishing or exploding gradients.

Parameters

_whichError	choose what error to monitor, for more information see Neuron::whichError
_whichGradient	choose what gradient of the chosen error to monitor, for more information see Layer::whichGradient

Returns

Returns the chosen gradient in this layer

2.1.4.12 getInitWeight()

```
double Layer::getInitWeight (
            int _neuronIndex,
            int _weightIndex )
```

Reports the initial value that was assigned to a specific weight at the initialisatin of the network

Parameters

_neuronIndex	Index of the neuron containing the weight
_weightIndex	Index of the weight

Returns

2.1.4.13 getLocalError()

```
double Layer::getLocalError (
            int _neuronIndex )
```

Allows for accessing the local error of a specific neuron

Parameters

_neuronIndex	The index of the neuron to request the local error from

Returns

Returns the local error

2.1.4.14 getMidError()

```
double Layer::getMidError (
            int _neuronIndex )
```

Allows for accessing the error that propagates bilaterally

Parameters

_neuronIndex	The index of the neuron that the error is requested from

Returns

Returns the mid error

2.1.4.15 getNeuron()

```
Neuron* Layer::getNeuron (
            int _neuronIndex )
```

Allows access to a specific neuron

Parameters

_neuronIndex	The index of the neuron to access

Returns

A pointer to that neuron

2.1.4.16 getnNeurons()

```
int Layer::getnNeurons ( )
```

Reports the number of neurons in this layer

Returns

The total number of neurons in this layer

2.1.4.17 getOutput()

```
double Layer::getOutput (
            int _neuronIndex )
```

Allows for accessing the activation of a specific neuron

Parameters

_neuronIndex	The index of the neuron

Returns

the activation of that neuron

2.1.4.18 getSumOutput()

```
double Layer::getSumOutput (
            int _neuronIndex )
```

Allows for accessing the sum output of any specific neuron

Parameters

_neuronIndex	The index of the neuron to access

Returns

Returns the wighted sum of the inputs to that neuron

2.1.4.19 getWeightChange()

```
double Layer::getWeightChange ( )
```

Accesses the total sum of weight changes of all the neurons in this layer

Returns

sum of weight changes all neurons

2.1.4.20 getWeightDistance()

```
double Layer::getWeightDistance ( )
```

Performs squared root on the weight change

Returns

The sqr of the weight changes

2.1.4.21 getWeights()

```
double Layer::getWeights (
             int _neuronIndex,
             int _weightIndex )
```

Allows for accessing any specific weights in the layer

Parameters

_neuronIndex	The index of the neuron containing that weight
_weightIndex	The index of the input to which that weight is assigned

Returns

Returns the chosen weight

2.1.4.22 initLayer()

```
void Layer::initLayer (
             int _layerIndex,
             Neuron::weightInitMethod _wim,
             Neuron::biasInitMethod _bim,
             Neuron::actMethod _am )
```

Initialises each layer with specific methods for weight/bias initialisation and activation function of neurons

Parameters

_layerIndex	The index that is assigned to this layer by the Net class
_wim	weights initialisation method, see Neuron::weightInitMethod for different options
_bim	biases initialisation method, see Neuron::biasInitMethod for different options
_am	activation method, see Neuron::actMethod for different options

2.1.4.23 printLayer()

```
void Layer::printLayer ( )
```

Prints on the console a full tree of this layer with the values of all weights and outputs for all neurons

2.1.4.24 propErrorBackward()

```
void Layer::propErrorBackward (
             int _neuronIndex,
             double _nextSum )
```

Sets the error to be propagated backward at all neurons, except those in the output layer.

Parameters

_neuronIndex	The index of the neuron receiving the weighted sum of errors
_nextSum	The weighted sum of propagating error

2.1.4.25 propErrorForward()

```
void Layer::propErrorForward (
            int _index,
            double _value )
```

Sets the error to be propagated forwards to all neurons in deeper layers

Parameters

_index	Index of input where the error originates form
_value	The value of the error

2.1.4.26 propGlobalErrorBackwardLocally()

```
void Layer::propGlobalErrorBackwardLocally (
            int _neuronIndex,
            double _nextSum )
```

sets the error that propagates backwards and locally (for one layer only) for all neurons

2.1.4.27 propInputs()

```
void Layer::propInputs (
            int _index,
            double _value )
```

Sets the inputs to all neurons in the deeper layers (excluding the first hidden layer)

Parameters

_index	The index of the input
_value	The value of the input

2.1.4.28 propMidErrorBackward()

```
void Layer::propMidErrorBackward (
             int _neuronIndex,
             double _nextSum )
```

Sets the mid error in all neurons of a specific layer chosen by Net

Parameters

_neuronIndex	The index of the neuron to receive the error
_nextSum	The weighted sum of errors

2.1.4.29 propMidErrorForward()

```
void Layer::propMidErrorForward (
             int _index,
             double _value )
```

Sets the mid error in all neurons of a chosen layer by Net

Parameters

_index	Index of the mid error
_value	Value of the mid error

2.1.4.30 saveWeights()

```
void Layer::saveWeights ( )
```

Saves the temporal weight change of all weights in all neurons into files

2.1.4.31 setBackwardError()

```
void Layer::setBackwardError (
             double _leadError )
```

Sets the error to be propagated backward at all neurons in the output layer only.

Parameters

_leadError	the error to be propagated backward

2.1.4.32 setEchoError()

```
void Layer::setEchoError (
             double _clError )
```

Sets the error to be resonated back and forth at all neurons

Parameters

_echoError	the resonating error

2.1.4.33 setErrorCoeff()

```
void Layer::setErrorCoeff (
             double _globalCoeff,
             double _backwardsCoeff,
             double _midCoeff,
             double _forwardCoeff,
             double _localCoeff,
             double _echoCoeff )
```

Sets the coefficient of the errors used for learning

Parameters

_globalCoeff	coefficient of the global error
_backwardsCoeff	coefficient of the error propagating backward
_midCoeff	coefficient of the error propagating bilaterally
_forwardCoeff	coefficient of the error propagating forward
_localCoeff	coefficient of the error propagating locally
_echoCoeff	coefficient of the error resonating back and forth

2.1.4.34 setForwardError()

```
void Layer::setForwardError (
             double _leadForwardError )
```

Sets the error to be propagated forward to all neurons in the first hidden layer only

Parameters

_leadForwardError	the error to be propagated forward

2.1.4.35 setGlobalError()

```
void Layer::setGlobalError (
             double _globalError )
```

Sets the global error, all neurons will have access to this error

Parameters

_globalError	The global error

2.1.4.36 setInputs()

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

Sets the inputs to all neurons in the first hidden layer only

Parameters

_inputs	A pointer to an array of inputs

2.1.4.37 setlearningRate()

```
void Layer::setlearningRate (
             double _learningRate )
```

Sets the learning rate.

Parameters

_learningRate	Sets the learning rate for all neurons.

2.1.4.38 setLocalError()

```
void Layer::setLocalError (
             double _leadLocalError )
```

Sets the local error at all neurons

Parameters

_leadLocalError	The error to be propagated locally only

2.1.4.39 setMidError()

```
void Layer::setMidError (
             double _leadMidError )
```

Sets the middle error in all neurons in the chosen layer by Net

Parameters

_leadMidError	The error to be propagated bilaterally

2.1.4.40 snapWeights()

```
void Layer::snapWeights ( )
```

Snaps the final distribution of weights in a specific layer, this is overwritten every time the function is called

2.1.4.41 updateWeights()

```
void Layer::updateWeights ( )
```

Requests that all neurons perform one iteration of learning

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

- Layer.h

2.2 Net Class Reference

```
#include <Net.h>
```

Public Member Functions

- Net (int _nLayers, int ∗_nNeurons, int _nInputs)
- ∼Net ()
- void initNetwork (Neuron::weightInitMethod _wim, Neuron::biasInitMethod _bim, Neuron::actMethod _am)
- void setLearningRate (double _learningRate)
- void setInputs (const double ∗_inputs)
- void propInputs ()
- void setForwardError (double _leadForwardError)
- void propErrorForward ()
- void setBackwardError (double _leadError)
- void propErrorBackward ()
- void setMidError (int _layerIndex, double _leadMidError)
- void propMidErrorForward ()
- void propMidErrorBackward ()
- double getGradient (Neuron::whichError _whichError, Layer::whichGradient _whichGradient)
- void setErrorCoeff (double _globalCoeff, double _backwardsCoeff, double _midCoeff, double _forwardCoeff, double _localCoeff, double _echoCoeff)
- void updateWeights ()
- void setGlobalError (double _globalError)
- void setEchoError (double _echoError)
- void echoErrorBackward ()
- void echoErrorForward ()
- void doEchoError (double _theError)
- void setLocalError (double _leadLocalError)
- void propGlobalErrorBackwardLocally ()
- Layer ∗ getLayer (int _layerIndex)
- double getOutput (int _neuronIndex)
- double getSumOutput (int _neuronIndex)
- int getnLayers ()
- int getnInputs ()
- double getWeightDistance ()
- double getLayerWeightDistance (int _layerIndex)
- double getWeights (int _layerIndex, int _neuronIndex, int _weightIndex)
- int getnNeurons ()
- void saveWeights ()
- void snapWeights ()
- void printNetwork ()

2.2.1 Detailed Description

Net is the main class used to set up a neural network used for closed-loop Deep Learning. It initialises all the layers and the neurons internally.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 Net()

```
Net::Net (
            int _nLayers,
            int * _nNeurons,
            int _nInputs )
```

Constructor: The neural network that performs the learning.

Parameters

_nLayers	Total number of hidden layers, excluding the input layer
_nNeurons	A pointer to an int array with number of neurons for all layers need to have the length of _nLayers.
_nInputs	Number of Inputs to the network

2.2.2.2 ∼Net()

```
Net::∼Net ( )
```

Destructor De-allocated any memory

2.2.3 Member Function Documentation

2.2.3.1 doEchoError()

```
void Net::doEchoError (
            double _theError )
```

It propagates the resonating error back and forth through the network using the echoErrorBackward and echo↩ ErrorForward until the residue error is zero

Parameters

_theError	The error used for resonating

2.2.3.2 echoErrorBackward()

```
void Net::echoErrorBackward ( )
```

Propagates the resonating error backward through the network

2.2.3.3 echoErrorForward()

```
void Net::echoErrorForward ( )
```

propagates the resonating error forward through the network

2.2.3.4 getGradient()

```
double Net::getGradient (
             Neuron::whichError _whichError,
             Layer::whichGradient _whichGradient )
```

It provides a measure of how the magnitude of the error changes through the layers to alarm for vanishing or exploding gradients.

Parameters

_whichError	choose what error to monitor, for more information see Neuron::whichError
_whichGradient	choose what gradient of the chosen error to monitor, for more information see Layer::whichGradient

Returns

Returns the ratio of the chosen gradient in the last layer to the the first layer

2.2.3.5 getLayer()

```
Layer* Net::getLayer (
             int _layerIndex )
```

Allows Net to access each layer

Parameters

_layerIndex	the index of the chosen layer

Returns

A pointer to the chosen Layer

2.2.3.6 getLayerWeightDistance()

```
double Net::getLayerWeightDistance (
             int _layerIndex )
```

Allows for monitoring the weight change in a specific layer of the network.

Parameters

_layerIndex	The index of the chosen layer

Returns

returns the Euclidean wight distance of neurons in the chosen layer from their initial value

2.2.3.7 getnInputs()

```
int Net::getnInputs ( )
```

Informs on the total number of inputs to the network

Returns

Total number of inputs

2.2.3.8 getnLayers()

```
int Net::getnLayers ( )
```

Informs on the total number of hidden layers (excluding the input layer)

Returns

Total number of hidden layers in the network

2.2.3.9 getnNeurons()

```
int Net::getnNeurons ( )
```

Informs on the total number of neurons in the network

Returns

The total number of neurons

2.2.3.10 getOutput()

```
double Net::getOutput (
             int _neuronIndex )
```

Allows the user to access the activation output of a specific neuron in the output layer only

Parameters

_neuronIndex	The index of the chosen neuron

Returns

The value at the output of the chosen neuron

2.2.3.11 getSumOutput()

```
double Net::getSumOutput (
             int _neuronIndex )
```

Allows the user to access the weighted sum output of a specific neuron in output layer only

Parameters

_neuronIndex	The index of the chosen neuron

Returns

The value at the sum output of the chosen neuron

2.2.3.12 getWeightDistance()

```
double Net::getWeightDistance ( )
```

Allows for monitoring the overall weight change of the network.

Returns

returns the Euclidean wight distance of all neurons in the network from their initial value

2.2.3.13 getWeights()

```
double Net::getWeights (
             int _layerIndex,
             int _neuronIndex,
             int _weightIndex )
```

Grants access to a specific weight in the network

Parameters

_layerIndex	Index of the layer that contains the chosen weight
_neuronIndex	Index of the neuron in the chosen layer that contains the chosen weight
_weightIndex	Index of the input to which the chosen weight is assigned

Returns

returns the value of the chosen weight

2.2.3.14 initNetwork()

```
void Net::initNetwork (
            Neuron::weightInitMethod _wim,
            Neuron::biasInitMethod _bim,
            Neuron::actMethod _am )
```

Dictates the initialisation of the weights and biases and determines the activation function of the neurons.

Parameters

_wim	weights initialisation method, see Neuron::weightInitMethod for different options
_bim	biases initialisation method, see Neuron::biasInitMethod for different options
_am	activation method, see Neuron::actMethod for different options

2.2.3.15 printNetwork()

```
void Net::printNetwork ( )
```

Prints on the console a full tree of the network with the values of all weights and outputs for all neurons

2.2.3.16 propErrorBackward()

```
void Net::propErrorBackward ( )
```

Propagates the _leadError backward through the network.

2.2.3.17 propErrorForward()

```
void Net::propErrorForward ( )
```

Propagates the _leadForwardError forward through the network.

2.2.3.18 propGlobalErrorBackwardLocally()

```
void Net::propGlobalErrorBackwardLocally ( )
```

propagates the local error backwards and locally (for one layer only)

2.2.3.19 propInputs()

```
void Net::propInputs ( )
```

It propagates the inputs forward through the network.

2.2.3.20 propMidErrorBackward()

```
void Net::propMidErrorBackward ( )
```

Propagates the _leadMidError from the chosen layer backward to the input layer.

2.2.3.21 propMidErrorForward()

```
void Net::propMidErrorForward ( )
```

Propagates the _leadMidError from the chosen layer forward to the output layer.

2.2.3.22 saveWeights()

```
void Net::saveWeights ( )
```

Saves the temporal changes of all weights in all neurons into files

2.2.3.23 setBackwardError()

```
void Net::setBackwardError (
             double _leadError )
```

Sets the error at the output layer to be propagated backward.

Parameters

_leadError	The closed-loop error for learning

2.2.3.24 setEchoError()

```
void Net::setEchoError (
            double _echoError )
```

Sets the error to be resonated back and forth in the network

Parameters

_echoError	the resonating error

2.2.3.25 setErrorCoeff()

```
void Net::setErrorCoeff (
            double _globalCoeff,
            double _backwardsCoeff,
            double _midCoeff,
            double _forwardCoeff,
            double _localCoeff,
            double _echoCoeff )
```

Sets the coefficient of the errors used for learning

Parameters

_globalCoeff	coefficient of the global error
_backwardsCoeff	coefficient of the error propagating backward
_midCoeff	coefficient of the error propagating bilaterally
_forwardCoeff	coefficient of the error propagating forward
_localCoeff	coefficient of the error propagating locally
_echoCoeff	coefficient of the error resonating back and forth

2.2.3.26 setForwardError()

```
void Net::setForwardError (
            double _leadForwardError )
```

Sets the error at the input layer to be propagated forward.

Parameters

_leadForwardError	The closed-loop error for learning

2.2.3.27 setGlobalError()

```
void Net::setGlobalError (
            double _globalError )
```

Sets the global error, all layers and neurons will have access to this error

Parameters

_globalError	The global error

2.2.3.28 setInputs()

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

Sets the inputs to the network in each iteration of learning, needs to be placed in an infinite loop.

Parameters

_inputs	A pointer to the array of inputs

2.2.3.29 setLearningRate()

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

Sets the learning rate.

Parameters

_learningRate	Sets the learning rate for all layers and neurons.

2.2.3.30 setLocalError()

```
void Net::setLocalError (
            double _leadLocalError )
```

Sets the local error at every layer

Parameters

_leadLocalError	The error to be propagated locally only

2.2.3.31 setMidError()

```
void Net::setMidError (
            int _layerIndex,
            double _leadMidError )
```

Sets the close-loop error to the a chosen layer to be propagated bilaterally.

Parameters

_layerIndex	The index of the layer at which to inject the error
_leadMidError	The closed-loop error for learning

2.2.3.32 snapWeights()

```
void Net::snapWeights ( )
```

Snaps the final distribution of all weights in a specific layer, this is overwritten every time the function is called

2.2.3.33 updateWeights()

```
void Net::updateWeights ( )
```

Requests that all layers perform one iteration of learning

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

- Net.h

2.3 Neuron Class Reference

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

Public Types

- enum biasInitMethod { **B_NONE** = 0, **B_RANDOM** = 1 }
- enum weightInitMethod { **W_ZEROS** = 0, **W_ONES** = 1, **W_RANDOM** = 2 }
- enum actMethod { **Act_Sigmoid** = 0, **Act_Tanh** = 1, **Act_NONE** = 2 }
- enum whichError { **onBackwardError** = 0, **onMidError** = 1, **onForwardError** = 2 }

Public Member Functions

- Neuron (int _nInputs)
- ∼Neuron ()
- void initNeuron (int _neuronIndex, int _layerIndex, weightInitMethod _wim, biasInitMethod _bim, actMethod _am)
- void setLearningRate (double _learningRate)
- void setInput (int _index, double _value)
- void propInputs (int _index, double _value)
- int calcOutput (int _layerHasReported)
- void setForwardError (double _value)
- void propErrorForward (int _index, double _value)
- void calcForwardError ()
- void setBackwardError (double _leadError)
- void propErrorBackward (double _nextSum)
- double getBackwardError ()
- void setMidError (double _leadMidError)
- void calcMidError ()
- double getMidError ()
- void propMidErrorForward (int _index, double _value)
- void propMidErrorBackward (double _nextSum)
- double getError (whichError _whichError)
- void setErrorCoeff (double _globalCoeff, double _backwardsCoeff, double _midCoeff, double _forwardCoeff, double _localCoeff, double _echoCoeff)
- void updateWeights ()
- double doActivation (double _sum)
- double doActivationPrime (double _input)
- void setGlobalError (double _globalError)
- double getGlobalError ()
- void setEchoError (double _echoError)
- double getEchoError ()
- void echoErrorBackward (double _nextSum)
- void echoErrorForward (int _index, double _value)
- void calcEchoError ()
- void setLocalError (double _leadLocalError)
- void propGlobalErrorBackwardLocally (double _nextSum)
- double getLocalError ()
- double getOutput ()
- double getForwardError ()
- double getSumOutput ()
- double getWeights (int _inputIndex)
- double getInitWeights (int _inputIndex)
- double getWeightChange ()
- double getMaxWeight ()
- double getMinWeight ()
- double getSumWeight ()
- double getWeightDistance ()
- int getnInputs ()
- void saveWeights ()
- void printNeuron ()
- void setWeight (int _index, double _weight)

2.3.1 Detailed Description

This is the class for creating neurons inside the Layer class. This is the building block class of the network.

2.3.2 Member Enumeration Documentation

2.3.2.1 actMethod

```
enum Neuron::actMethod
```

Options for activation functions of the neuron 0 for using the logistic function 1 for using the hyperbolic tan function 2 for unity function (no activation)

2.3.2.2 biasInitMethod

```
enum Neuron::biasInitMethod
```

Options for method of initialising biases 0 for initialising all weights to zero 1 for initialising all weights to one 2 for initialising all weights to a random value between 0 and 1

2.3.2.3 weightInitMethod

```
enum Neuron::weightInitMethod
```

Options for method of initialising weights 0 for initialising all weights to zero 1 for initialising all weights to one 2 for initialising all weights to a random value between 0 and 1

2.3.2.4 whichError

```
enum Neuron::whichError
```

Options for choosing an error to monitor the gradient of 0 for monitoring the error that propagates backward 1 for monitoring the error that propagates from the middle and bilaterally 2 for monitoring the error that propagates forward

2.3.3 Constructor & Destructor Documentation

2.3.3.1 Neuron()

```
Neuron::Neuron (
            int _nInputs )
```

Constructor for the Neuron class: it initialises a neuron with specific number fo inputs to that neuron

Parameters

_nInputs	

2.3.3.2 ∼Neuron()

```
Neuron::∼Neuron ( )
```

Destructor De-allocated any memory

2.3.4 Member Function Documentation

2.3.4.1 calcEchoError()

```
void Neuron::calcEchoError ( )
```

calculated the resonating error to be propagates to adjacent layers

2.3.4.2 calcForwardError()

```
void Neuron::calcForwardError ( )
```

Calculates the error to be propagated forward by doing a weighted sum of forward errors

2.3.4.3 calcMidError()

```
void Neuron::calcMidError ( )
```

calculates the mid error

2.3.4.4 calcOutput()

```
int Neuron::calcOutput (
             int _layerHasReported )
```

Calculates the output of the neuron by performing a weighed sum of all inputs to this neuron and activating the sum

Parameters

_layerHasReported	boolean variable to indicate whether or not any neuron in this layer has reported exploding output

Returns

Returns a boolean to report whether or not this neuron has exploding output

2.3.4.5 doActivation()

```
double Neuron::doActivation (
             double _sum )
```

Performs the activation of the sum output of the neuron

Parameters

_sum	the weighted sum of all inputs

Returns

activation of the sum

2.3.4.6 doActivationPrime()

```
double Neuron::doActivationPrime (
             double _input )
```

Performs inverse activation on any input that is passed to this function

Parameters

_input	the input value

Returns

the inverse activation of the input

2.3.4.7 echoErrorBackward()

```
void Neuron::echoErrorBackward (
             double _nextSum )
```

Sets the forward travelling resonating error for this neuron

Returns

the resonating error

2.3.4.8 echoErrorForward()

```
void Neuron::echoErrorForward (
             int _index,
             double _value )
```

Sets the backward travelling resonating error for this neuron

Returns

the resonating error

2.3.4.9 getBackwardError()

```
double Neuron::getBackwardError ( )
```

Allows accessing the backward error

Returns

The back propagating error fo this neuron

2.3.4.10 getEchoError()

```
double Neuron::getEchoError ( )
```

Requests for the resonating error

Returns

Returns the resonating error

2.3.4.11 getError()

```
double Neuron::getError (
             whichError _whichError )
```

Allows for accessing any specific error of this neuron

Parameters

_whichError	specifies the error, for more information see whichError

Returns

returns the value of the chosen error

2.3.4.12 getForwardError()

```
double Neuron::getForwardError ( )
```

Requests the forward propagating error

Returns

the forward error

2.3.4.13 getGlobalError()

```
double Neuron::getGlobalError ( )
```

Allows for accessing the global error

Returns

Returns the global error

2.3.4.14 getInitWeights()

```
double Neuron::getInitWeights (
             int _inputIndex )
```

Requests a inital value of a specific weight

Parameters

_inputIndex	index of the input to which the weight is assigned

Returns

teh inital value of the weight

2.3.4.15 getLocalError()

```
double Neuron::getLocalError ( )
```

Requests the local error fo this neuron

Returns

Returns the local error

2.3.4.16 getMaxWeight()

```
double Neuron::getMaxWeight ( )
```

Requests for the maximum weights located in this neuron

Returns

Returns the max weight

2.3.4.17 getMidError()

```
double Neuron::getMidError ( )
```

Allows accessing the mid error of this neuron

Returns

the value of the mid error

2.3.4.18 getMinWeight()

```
double Neuron::getMinWeight ( )
```

Requests for the minimum weights located in this neuron

Returns

Returns the min weight

2.3.4.19 getnInputs()

```
int Neuron::getnInputs ( )
```

Requests the total number of inputs to this neuron

Returns

total number of inputs

2.3.4.20 getOutput()

```
double Neuron::getOutput ( )
```

Requests the output of this neuron

Returns

the output of the neuron after activation

2.3.4.21 getSumOutput()

```
double Neuron::getSumOutput ( )
```

Requests the sum output of the neuron

Returns

returns the sum output of the neuron before activaiton

2.3.4.22 getSumWeight()

```
double Neuron::getSumWeight ( )
```

Requests for the total sum of weights located in this neuron

Returns

Returns the sum of weights

2.3.4.23 getWeightChange()

```
double Neuron::getWeightChange ( )
```

Requests for overall change of all weights contained in this neuron

Returns

the overal weight change

2.3.4.24 getWeightDistance()

```
double Neuron::getWeightDistance ( )
```

Requests the weight distance of all weighs in this neuron

Returns

returns the sqr of the total weight change in this neuron

2.3.4.25 getWeights()

```
double Neuron::getWeights (
             int _inputIndex )
```

Requests a specific weight

Parameters

_inputIndex	index of the input to which the chosen weight is assigned

Returns

Returns the chosen weight

2.3.4.26 initNeuron()

```
void Neuron::initNeuron (
             int _neuronIndex,
             int _layerIndex,
             weightInitMethod _wim,
             biasInitMethod _bim,
             actMethod _am )
```

Initialises the neuron with the given methods for weight/bias initialisation and for activation function. It also specifies the index of the neuron and the index of the layer that contains this neuron.

Parameters

_neuronIndex	The index of this neuron
_layerIndex	The index of the layer that contains this neuron
_wim	The method of initialising the weights, refer to weightInitMethod for more information
_bim	The method of initialising the biases, refer to biasInitMethod for more information
_am	The function used for activation of neurons, refer to actMethod for more information

2.3.4.27 printNeuron()

```
void Neuron::printNeuron ( )
```

Prints on the console a full description of all weights, inputs and outputs for this neuron

2.3.4.28 propErrorBackward()

```
void Neuron::propErrorBackward (
             double _nextSum )
```

Sets the error to be propagated backward for neurons in all layers except for the output layer

Parameters

_nextSum	the weighted sum of propagating errors

2.3.4.29 propErrorForward()

```
void Neuron::propErrorForward (
             int _index,
             double _value )
```

Sets the forward propagating error of the neuron in layers other than the first hidden layer

Parameters

_index	index of the error
_value	value of the error

2.3.4.30 propGlobalErrorBackwardLocally()

```
void Neuron::propGlobalErrorBackwardLocally (
            double _nextSum )
```

Sets the error that propagates backward but only locally (for one layer)

Parameters

_nextSum	the sum of errors to be propagated

2.3.4.31 propInputs()

```
void Neuron::propInputs (
            int _index,
            double _value )
```

Sets the inputs to this neuron that can be located in any layer other than the first hidden layer

Parameters

_index	index of the input
_value	value of the input

2.3.4.32 propMidErrorBackward()

```
void Neuron::propMidErrorBackward (
            double _nextSum )
```

Sets the backward propagating mid error for this neuron

Parameters

_nextSum	the value of weighted sum of mid errors in neurons of the adjacent layer

2.3.4.33 propMidErrorForward()

```
void Neuron::propMidErrorForward (
            int _index,
            double _value )
```

Sets the forward propagating mid errors for this neuron

Parameters

_index	index of the error
_value	value of the error

2.3.4.34 saveWeights()

```
void Neuron::saveWeights ( )
```

Saves the temporal weight change of all weights in this neuron into a file

2.3.4.35 setBackwardError()

```
void Neuron::setBackwardError (
             double _leadError )
```

Sets the backward propagating error in neuron in the output layer

Parameters

_leadError	the value of the error

2.3.4.36 setEchoError()

```
void Neuron::setEchoError (
             double _echoError )
```

Sets the resonating error for this neuron called from the output layer only

Parameters

_echoError	The resonating error

2.3.4.37 setErrorCoeff()

```
void Neuron::setErrorCoeff (
             double _globalCoeff,
             double _backwardsCoeff,
             double _midCoeff,
             double _forwardCoeff,
             double _localCoeff,
             double _echoCoeff )
```

Sets the coefficient of the errors used for learning

Parameters

_globalCoeff	coefficient of the global error
_backwardsCoeff	coefficient of the error propagating backward
_midCoeff	coefficient of the error propagating bilaterally
_forwardCoeff	coefficient of the error propagating forward
_localCoeff	coefficient of the error propagating locally
_echoCoeff	coefficient of the error resonating back and forth

2.3.4.38 setForwardError()

```
void Neuron::setForwardError (
            double _value )
```

Sets the error of the neuron in the first hidden layer that is to be propagated forward

Parameters

_value	value of the error

2.3.4.39 setGlobalError()

```
void Neuron::setGlobalError (
            double _globalError )
```

Sets the global error for this neuron

Parameters

_globalError	the global error

2.3.4.40 setInput()

```
void Neuron::setInput (
            int _index,
            double _value )
```

Sets the inputs to this neuron that is located in the first hidden layer

Parameters

_index	Index of the input
_value	Value of the input

2.3.4.41 setLearningRate()

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

Sets the learning rate.

Parameters

_learningRate	Sets the learning rate for this neuron.

2.3.4.42 setLocalError()

```
void Neuron::setLocalError (
            double _leadLocalError )
```

Sets the error to be propagated locally

Parameters

_leadLocalError	the local error

2.3.4.43 setMidError()

```
void Neuron::setMidError (
            double _leadMidError )
```

Sets the mid error of neuron that is on the chosen layer for bilateral propagation

Parameters

_leadMidError	the error to be propagated bilaterally

2.3.4.44 setWeight()

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

Sets the weights of the neuron

Parameters

_index	index of the weight
_weight	value of the weight

2.3.4.45 updateWeights()

```
void Neuron::updateWeights ( )
```

Performs one iteration of learning, that is: it updates all the weights assigned to each input to this neuron

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

- Neuron.h