NNTLib

Generated by Doxygen 1.8.9.1

Mon Jun 15 2015 10:25:20

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

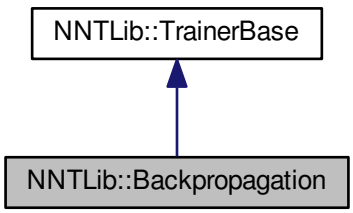
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

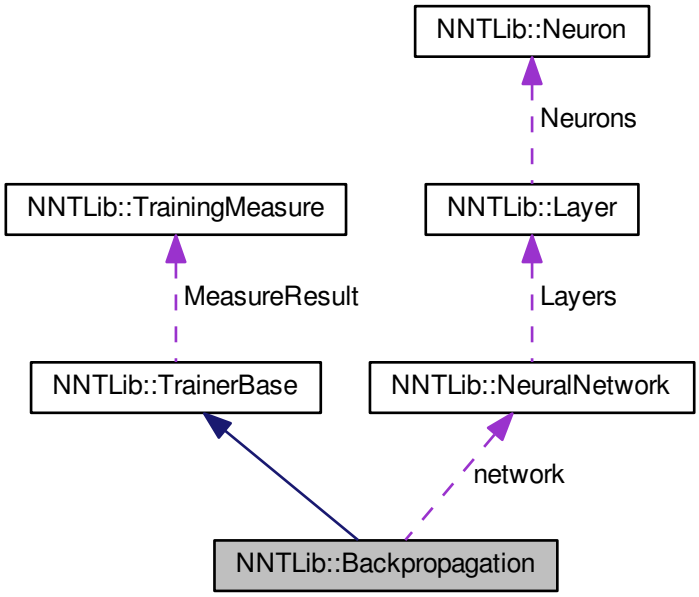
Class Documentation

3.1 NNTLib::Backpropagation Class Reference

Inheritance diagram for NNTLib::Backpropagation:



Collaboration diagram for NNTLib::Backpropagation:



Public Member Functions

- Backpropagation (NeuralNetwork &net)

  Initializes a new instance of the Backpropagation class.
- ~Backpropagation ()

  Finalizes an instance of the Backpropagation class.
- void Train (const DataContainer &container, const double learnRate, const int maxLoopCount, const double momentum=0, int minibatchSize=1, const double errorThreshold=0, const double decayRate=0)

  Trains the specified container.

Public Attributes

- NeuralNetwork * network

  The network

Additional Inherited Members

3.1.1 Constructor & Destructor Documentation

3.1.1.1 NNTLib::Backpropagation::Backpropagation (NeuralNetwork & *net*)

Initializes a new instance of the Backpropagation class.

Parameters

net	The net.

3.1.1.2 NNTLib::Backpropagation::~Backpropagation ( )

Finalizes an instance of the Backpropagation class.

3.1.2 Member Function Documentation

3.1.2.1 void NNTLib::Backpropagation::Train (const DataContainer & *container*, const double *learnRate*, const int *maxLoopCount*, const double *momentum* = 0, int *minibatchSize* = 1, const double *errorThreshold* = 0, const double *decayRate* = 0)

Trains the specified container.

Parameters

container	The container.
learnRate	The learn rate.
maxLoopCount	The maximum loop count.
momentum	The momentum.
minibatchSize	Size of the batch.
errorThreshold	The error threshold.
decayRate	The decay rate.

3.1.3 Member Data Documentation

3.1.3.1 NeuralNetwork* NNTLib::Backpropagation::network

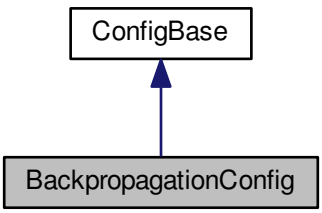
The network

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

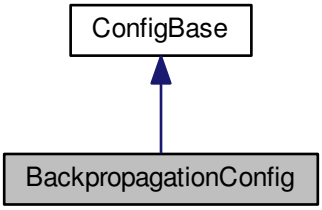
- NNTLib/Backpropagation.h
- NNTLib/Backpropagation.cpp

3.2 BackpropagationConfig Class Reference

Inheritance diagram for BackpropagationConfig:



Collaboration diagram for BackpropagationConfig:



Public Member Functions

- BackpropagationConfig ()

  Initializes a new instance of the BackpropagationConfig class.
- void PrintData ()

  Prints the data.
- bool IsConfigValid ()

  Determines whether [is configuration valid].

Public Attributes

- double ErrorThreshold

  The error threshold
- int MaxLoopCount

  The maximum loop count
- int BatchSize

  The batch size
- double Alpha

  The alpha
- double Momentum

  The momentum
- double DecayRate

  The decay rate

Protected Member Functions

- void HandleNameValue (std::string name, std::string value)

  Handles the name value.

3.2.1 Constructor & Destructor Documentation

3.2.1.1 BackpropagationConfig::BackpropagationConfig ( )

Initializes a new instance of the BackpropagationConfig class.

3.2.2 Member Function Documentation

3.2.2.1 void BackpropagationConfig::HandleNameValue (std::string *name,* std::string *value*) [protected], [virtual]

Handles the name value.

Parameters

name	The name.
value	The value.

Implements ConfigBase.

3.2.2.2 bool BackpropagationConfig::IsConfigValid ( ) [virtual]

Determines whether [is configuration valid].

Returns

Implements ConfigBase.

3.2.2.3 void BackpropagationConfig::PrintData ( ) [virtual]

Prints the data.

Implements ConfigBase.

3.2.3 Member Data Documentation

3.2.3.1 double BackpropagationConfig::Alpha

The alpha

3.2.3.2 int BackpropagationConfig::BatchSize

The batch size

3.2.3.3 double BackpropagationConfig::DecayRate

The decay rate

3.2.3.4 double BackpropagationConfig::ErrorThreshold

The error threshold

3.2.3.5 int BackpropagationConfig::MaxLoopCount

The maximum loop count

3.2.3.6 double BackpropagationConfig::Momentum

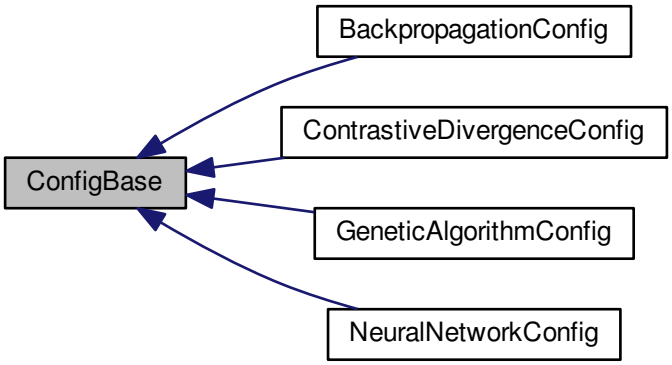
The momentum

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

- NNTLib/BackpropagationConfig.h
- NNTLib/BackpropagationConfig.cpp

3.3 ConfigBase Class Reference

Inheritance diagram for ConfigBase:



Public Member Functions

- void LoadFile (const char *file)

  Loads the file.
- virtual ~ConfigBase ()

  Finalizes an instance of the ConfigBase class.
- virtual void **PrintData** ()=0
- virtual bool **IsConfigValid** ()=0

Protected Member Functions

- virtual void **HandleNameValue** (std::string name, std::string value)=0

3.3.1 Constructor & Destructor Documentation

3.3.1.1 ConfigBase::~ConfigBase ( ) [virtual]

Finalizes an instance of the ConfigBase class.

3.3.2 Member Function Documentation

3.3.2.1 void ConfigBase::LoadFile (const char * *file*)

Loads the file.

Parameters

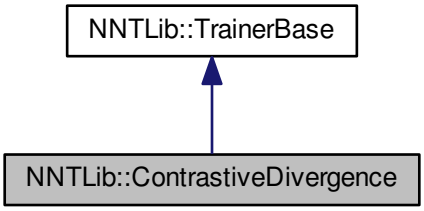
file	The file.

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

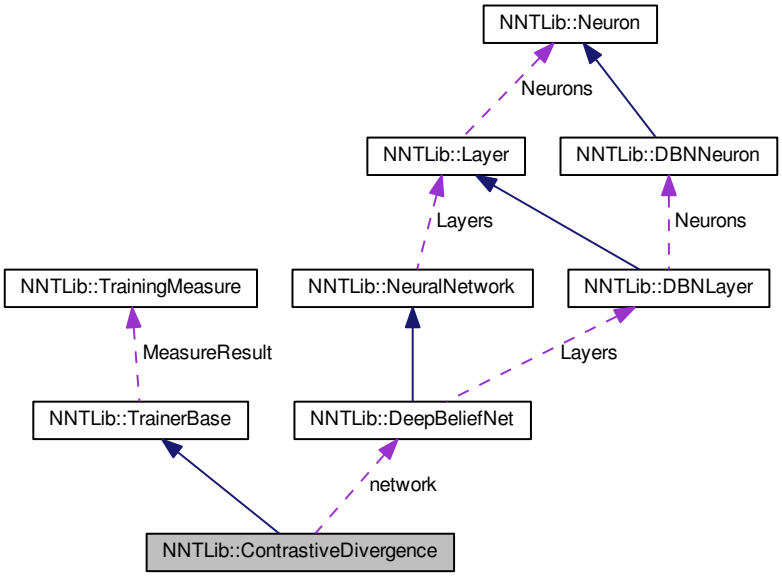
- NNTLib/ConfigBase.h
- NNTLib/ConfigBase.cpp

3.4 NNTLib::ContrastiveDivergence Class Reference

Inheritance diagram for NNTLib::ContrastiveDivergence:



Collaboration diagram for NNTLib::ContrastiveDivergence:



Public Member Functions

- void **GibbsSampling** (int gibbssteps, int d_i)
- void **UpdateHiddenUnits** ()
- void **UpdateVisibleUnits** ()
- **ContrastiveDivergence** (DeepBeliefNet &net)
- void **Train** (const DataContainer &container, const double learnRate, const int Epochs, int BatchSize=1, int gibbs=1)

Public Attributes

- DeepBeliefNet * network

  The network

Additional Inherited Members

3.4.1 Member Data Documentation

3.4.1.1 DeepBeliefNet* NNTLib::ContrastiveDivergence::network

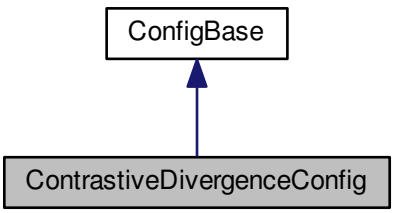
The network

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

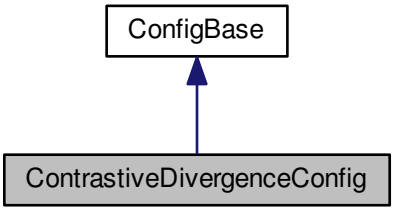
- NNTLib/ContrastiveDivergence.h
- NNTLib/ContrastiveDivergence.cpp

3.5 ContrastiveDivergenceConfig Class Reference

Inheritance diagram for ContrastiveDivergenceConfig:



Collaboration diagram for ContrastiveDivergenceConfig:



Public Member Functions

- ContrastiveDivergenceConfig ()
  Initializes a new instance of the ContrastiveDivergenceConfig class.
- void PrintData ()
  Prints the data.
- bool IsConfigValid ()
  Determines whether [is configuration valid].

Public Attributes

- int GibbsSteps
  The error threshold
- int BatchSize
  The batch size
- double LearnRate
  The alpha
- int **Epochs**

Protected Member Functions

- void HandleNameValue (std::string name, std::string value)
  Handles the name value.

3.5.1 Constructor & Destructor Documentation

3.5.1.1 ContrastiveDivergenceConfig::ContrastiveDivergenceConfig ( )

Initializes a new instance of the ContrastiveDivergenceConfig class.

3.5.2 Member Function Documentation

3.5.2.1 void ContrastiveDivergenceConfig::HandleNameValue (std::string *name*, std::string *value*) [protected], [virtual]

Handles the name value.

Parameters

name	The name.
value	The value.

Implements ConfigBase.

3.5.2.2 bool ContrastiveDivergenceConfig::IsConfigValid ( ) [virtual]

Determines whether [is configuration valid].

Returns

Implements ConfigBase.

3.5.2.3 void ContrastiveDivergenceConfig::PrintData ( ) [virtual]

Prints the data.

Implements ConfigBase.

3.5.3 Member Data Documentation

3.5.3.1 int ContrastiveDivergenceConfig::BatchSize

The batch size

3.5.3.2 int ContrastiveDivergenceConfig::GibbsSteps

The error threshold

The maximum loop count

3.5.3.3 double ContrastiveDivergenceConfig::LearnRate

The alpha

The momentum

The decay rate

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

- NNTLib/ContrastiveDivergenceConfig.h
- NNTLib/ContrastiveDivergenceConfig.cpp

3.6 NNTLib::DataContainer Class Reference

Public Member Functions

- void CopyData (const DataContainer &src, int startindexDst, int startindexSource, int lenght)

  Copies the data.
- DataContainer ()

Initializes a new instance of the DataContainer struct.

- void LoadFile (const char *file)

  Loads the file. Datei muss wie folgt ausgebaut sein 2 2 1 //[Anzahl Daten] [Anzahl Input] [Anzahl Output] 0 0 //Input Daten 1 {0,0} 0 //Output Daten 1 {0} 0 1 //Input Daten 2 {0,1} 1 //Output Daten 2 {1}
- void Init (int dataCount, int inputCount, int outputCount)

  Initializes the specified data count.
- ~DataContainer ()

  Finalizes an instance of the DataContainer class.
- DataContainer (const DataContainer &that)

  Initializes a new instance of the DataContainer class.
- DataContainer & operator= (const DataContainer &that)

  Operator=s the specified that.

Public Attributes

- int DataCount

  The data count
- int InputCount

  The input count
- int OutputCount

  The output count
- double ** DataInput

  The data input
- double ** DataOutput

  The data output

3.6.1 Constructor & Destructor Documentation

3.6.1.1 NNTLib::DataContainer::DataContainer ( )

Initializes a new instance of the DataContainer struct.

3.6.1.2 NNTLib::DataContainer::~DataContainer ( )

Finalizes an instance of the DataContainer class.

3.6.1.3 NNTLib::DataContainer::DataContainer (const DataContainer & *that*)

Initializes a new instance of the DataContainer class.

Parameters

that	The that.

3.6.2 Member Function Documentation

3.6.2.1 void NNTLib::DataContainer::CopyData (const DataContainer & *src*, int *startindexDst*, int *startindexSource*, int *lenght*)

Copies the data.

Parameters

src	The source.
startindexDst	The startindex DST.
startindexSource	The startindex source.
lenght	The lenght.

3.6.2.2 void NNTLib::DataContainer::Init (int *dataCount,* int *inputCount,* int *outputCount*)

Initializes the specified data count.

Parameters

dataCount	The data count.
inputCount	The input count.
outputCount	The output count.

3.6.2.3 void NNTLib::DataContainer::LoadFile (const char * *file*)

Loads the file. Datei muss wie folgt ausgebaut sein 2 2 1 //[Anzahl Daten] [Anzahl Input] [Anzahl Output] 0 0 //Input Daten 1 {0,0} 0 //Output Daten 1 {0} 0 1 //Input Daten 2 {0,1} 1 //Output Daten 2 {1}

Parameters

file	The file.

3.6.2.4 DataContainer & NNTLib::DataContainer::operator= (const DataContainer & *that*)

Operator=s the specified that.

Parameters

that	The that.

Returns

3.6.3 Member Data Documentation

3.6.3.1 int NNTLib::DataContainer::DataCount

The data count

3.6.3.2 double** NNTLib::DataContainer::DataInput

The data input

3.6.3.3 double** NNTLib::DataContainer::DataOutput

The data output

3.6.3.4 int NNTLib::DataContainer::InputCount

The input count

3.6.3.5 int NNTLib::DataContainer::OutputCount

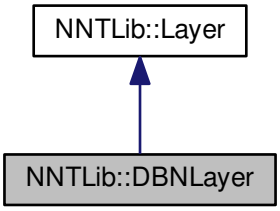
The output count

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

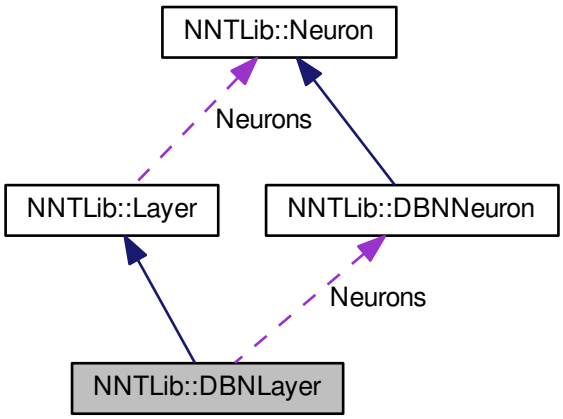
- NNTLib/DataContainer.h
- NNTLib/DataContainer.cpp

3.7 NNTLib::DBNLayer Class Reference

Inheritance diagram for NNTLib::DBNLayer:



Collaboration diagram for NNTLib::DBNLayer:



Public Member Functions

- void init ()

  Initiliases Layer.
- void Init (int inputsize, int neuronCount)

  Initialises Layer with parameters.
- void Forwardweightsinit (int inputsize, DBNLayer ∗Layerup)

  Creates Forwardweights.
- DBNLayer ()

  Initializes a new instance of the DBNLayer class.
- ∼DBNLayer ()

  Finalizes an instance of the DBNLayer class.
- DBNLayer (const DBNLayer &that)

  copy constructor
- DBNLayer & operator= (const DBNLayer &that)

  overloaded =

Public Attributes

- DBNNeuron ∗ Neurons

Additional Inherited Members

3.7.1 Constructor & Destructor Documentation

3.7.1.1 NNTLib::DBNLayer::DBNLayer ( )

Initializes a new instance of the DBNLayer class.

3.7.1.2 NNTLib::DBNLayer::∼DBNLayer ( )

Finalizes an instance of the DBNLayer class.

3.7.1.3 NNTLib::DBNLayer::DBNLayer (const DBNLayer & *that*)

copy constructor

copy constructor

Parameters

that	Layer to copy

3.7.2 Member Function Documentation

3.7.2.1 void NNTLib::DBNLayer::Forwardweightsinit (int *Neuronsdown,* DBNLayer ∗ *Layerup*)

Creates Forwardweights.

Sets Pointers to the weights from the layer above, making it easier to access them from the down Layer

Parameters

Neuronsdown	Number of Neurons on the down Layer
Layerup	Pointer to the above Layer

3.7.2.2 void NNTLib::DBNLayer::init ( )

Initiliases Layer.

Simple Initialisation without parameters sets everything to 0

3.7.2.3 void NNTLib::DBNLayer::Init (int *inputsize*, int *neuronCount*)

Initialises Layer with parameters.

Initialises the Layer with Neurons, the number of inputvalues, the Deltas for errors in the weighst. Takes Care of building the weights Between Layers and that the Bias has no input weights

Parameters

inputsize	Number of Inputs on the Layer
neuronCount	Number of Neurons on the Layer

3.7.2.4 DBNLayer & NNTLib::DBNLayer::operator= (const DBNLayer & *that*)

overloaded =

Copies one layer into another

Parameters

that	layer to copy

Returns

new layer with copied values

3.7.3 Member Data Documentation

3.7.3.1 DBNNeuron* NNTLib::DBNLayer::Neurons

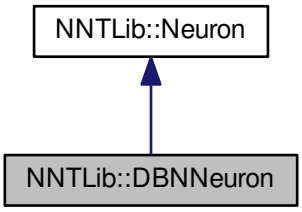
Deep Belief Neurons on the Layer

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

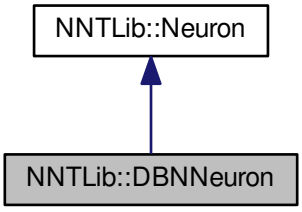
- NNTLib/DBNLayer.h
- NNTLib/DBNLayer.cpp

3.8 NNTLib::DBNNeuron Class Reference

Inheritance diagram for NNTLib::DBNNeuron:



Collaboration diagram for NNTLib::DBNNeuron:



Public Member Functions

- DBNNeuron ()
  standard constructor
- ∼DBNNeuron ()
  destructor
- DBNNeuron (const DBNNeuron &that)
  Copy Constructor.
- DBNNeuron & operator= (const DBNNeuron &that)
  Overloads =.
- void InitBias (const DataContainer ∗container)
  Initialises Biasneuron.
- void Init (int weightCount)
  Initialises Neuron.

Public Attributes

- int ForwardWeightCount

Number of Forwardweights.

- double p

  propability to turn on

- double ** ForwardWeights

  Pointer to weights in the layer above.

Protected Member Functions

- void copy (const DBNNeuron &that)

  Copies Neuron.

- void init ()

  Initialises Neuron with default Values.

- void freeMem ()

  Frees memory.

3.8.1 Constructor & Destructor Documentation

3.8.1.1 NNTLib::DBNNeuron::DBNNeuron (const DBNNeuron & *that*)

Copy Constructor.

Parameters

that	Neuron to copy

3.8.2 Member Function Documentation

3.8.2.1 void NNTLib::DBNNeuron::copy (const DBNNeuron & *that*) [protected]

Copies Neuron.

Copies Neuron into a new one

Parameters

that	Neuron to copy

3.8.2.2 void NNTLib::DBNNeuron::freeMem ( ) [protected]

Frees memory.

deletes Forwardweights

3.8.2.3 void NNTLib::DBNNeuron::init ( ) [protected]

Initialises Neuron with default Values.

Sets everything to 0 and the bias to 1

3.8.2.4 void NNTLib::DBNNeuron::Init (int *weightCount*)

Initialises Neuron.

Allocates weights, deltaeights and lastdeltaweights with weightcount and sets weightcount

Parameters

weightCount	Number of Backward weights on the neuron

3.8.2.5 void NNTLib::DBNNeuron::InitBias (const DataContainer * *container*)

Initialises Biasneuron.

Calculates $[p_i/(1-p_i)]$ with p_i as the average propability that a neuron turns on vor the visible layer. Hidden Layer gets Bias set to 0.

Parameters

container	Datacontainer with the probabilites for every training example

3.8.2.6 DBNNeuron & NNTLib::DBNNeuron::operator= (const DBNNeuron & *that*)

Overloads =.

Parameters

that	Neuron to copy

Returns

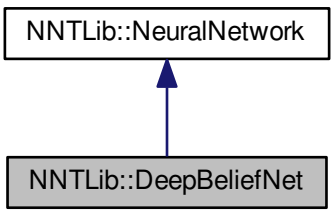
new Neuron with copied values

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

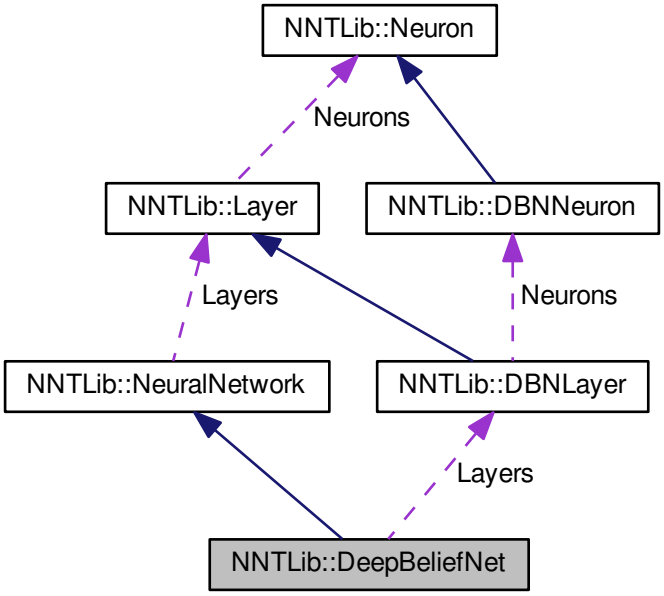
- NNTLib/DBNNeuron.h
- NNTLib/DBNNeuron.cpp

3.9 NNTLib::DeepBeliefNet Class Reference

Inheritance diagram for NNTLib::DeepBeliefNet:



Collaboration diagram for NNTLib::DeepBeliefNet:



Public Member Functions

- ~DeepBeliefNet ()

  destructor
- DeepBeliefNet (int *layers, int layercount, WeightInitEnum initType, FunctionEnum functionType)

  Constructor.
- DeepBeliefNet (const DeepBeliefNet &that)

  Copy Constructor.
- void InitWeights (WeightInitEnum initType)

  Initialises Weights.
- void SaveWeightsforNN (const std::string file)

  Saves Weights for neural network.

Public Attributes

- DBNLayer * **Layers**

Protected Member Functions

- void copy (const DeepBeliefNet &that)

  copies one Net into another
- void init ()

  Initializes this instance.
- void **freeMem** ()

Protected Attributes

- std::mt19937 **generator**

3.9.1 Constructor & Destructor Documentation

3.9.1.1 NNTLib::DeepBeliefNet::~DeepBeliefNet ( )

destructor

destructor

3.9.1.2 NNTLib::DeepBeliefNet::DeepBeliefNet (int * *neuronsCountPerLayer,* int *layercount,* WeightInitEnum *initType,* FunctionEnum *functionType*)

Constructor.

Initiliases the Deep Belief Net and builds it

Parameters

neuronsCount↩ PerLayer	Array with the number of Neurons for every layer
layercount	Number of Layers
initType	Which way to initialise Weights should be used
functionType	Which activation function is used in the neurons

3.9.1.3 NNTLib::DeepBeliefNet::DeepBeliefNet (const **DeepBeliefNet** & *that*)

Copy Constructor.

Parameters

that	Net to copy

3.9.2 Member Function Documentation

3.9.2.1 void NNTLib::DeepBeliefNet::copy (const **DeepBeliefNet** & *that*) [protected]

copies one Net into another

Parameters

that	Net to copy

3.9.2.2 void NNTLib::DeepBeliefNet::init ( ) [protected]

Initializes this instance.

3.9.2.3 void NNTLib::DeepBeliefNet::InitWeights (WeightInitEnum *initType*)

Initialises Weights.

Initialises backward Weights for every Neuron in the net

Parameters

initType	Which type of intialisation should be used

3.9.2.4 void NNTLib::DeepBeliefNet::SaveWeightsforNN (const std::string *file*)

Saves Weights for neural network.

Save weights in a way that a normal feedforward net can easily read them. The backward bias weights get lost in the process

Parameters

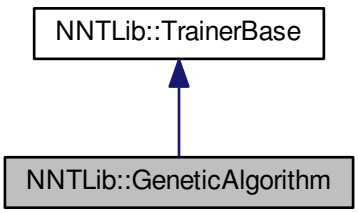
file	Path to file that gets the weights

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

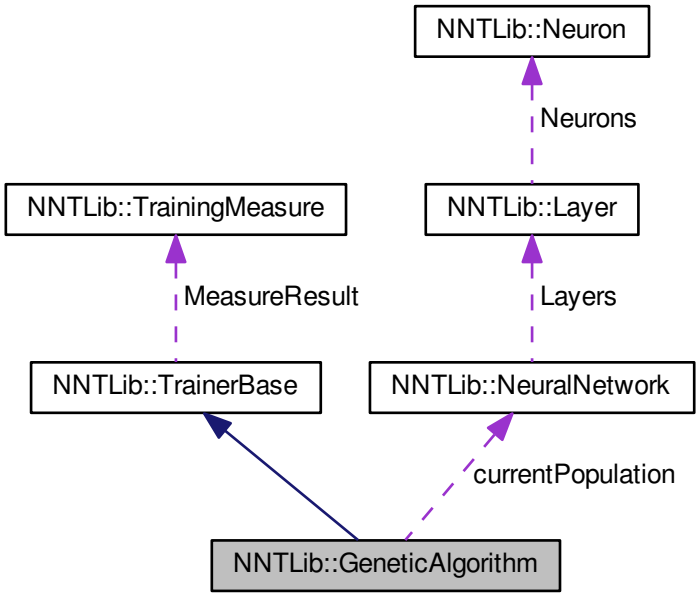
- NNTLib/DeepBeliefNet.h
- NNTLib/DeepBeliefNet.cpp

3.10 NNTLib::GeneticAlgorithm Class Reference

Inheritance diagram for NNTLib::GeneticAlgorithm:



Collaboration diagram for NNTLib::GeneticAlgorithm:



Public Member Functions

- GeneticAlgorithm (NeuralNetwork **currentPopulation, int populationsize)

  Initializes a new instance of the GeneticAlgorithm class.
- ~GeneticAlgorithm ()

  Finalizes an instance of the GeneticAlgorithm class.
- void Train (const DataContainer &dataContainer, int maxLoopCount, double errorThreshold, MutateEnum mutateType, CrossoverEnum crossoverType, RouletteEnum rouletteType, int eltismCount, double mutation↩ Probability=0.1, double crossoverProbability=0.8, int mutateNodeCount=0)

  Trains the specified data container.

Public Attributes

- NeuralNetwork ** currentPopulation

  The current population

Additional Inherited Members

3.10.1 Constructor & Destructor Documentation

3.10.1.1 NNTLib::GeneticAlgorithm::GeneticAlgorithm (NeuralNetwork ** *currentpopulation*, int *populationsize*)

Initializes a new instance of the GeneticAlgorithm class.

Parameters

currentPopulation	The current population.
populationSize	Size of the population.

Initializes a new instance of the GeneticAlgorithm class.

Parameters

currentPopulation	The current population.
populationsize	The populationsize.

3.10.1.2 NNTLib::GeneticAlgorithm::~GeneticAlgorithm ( )

Finalizes an instance of the GeneticAlgorithm class.

3.10.2 Member Function Documentation

3.10.2.1 void NNTLib::GeneticAlgorithm::Train (const DataContainer & *dataContainer,* int *maxLoopCount,* double *errorThreshold,* MutateEnum *mutateType,* CrossoverEnum *crossoverType,* RouletteEnum *rouletteType,* int *eltismCount,* double *mutationProbability* = 0.1, double *crossoverProbability* = 0.8, int *mutateNodeCount* = 0)

Trains the specified data container.

Parameters

dataContainer	The data container.
maxLoopCount	The maximum loop count.
errorThreshold	The error threshold.
mutateType	Type of the mutate.
crossoverType	Type of the crossover.
rouletteType	Type of the roulette.
eltismCount	The eltism count.
mutationProbability	The mutation probability.
crossoverProbability	The crossover probability.
mutateNodeCount	The mutate node count.

3.10.3 Member Data Documentation

3.10.3.1 NeuralNetwork** NNTLib::GeneticAlgorithm::currentPopulation

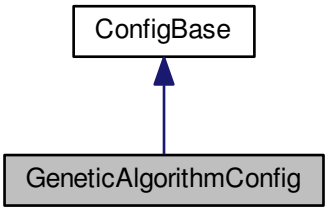
The current population

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

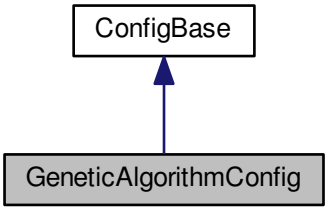
- NNTLib/GeneticAlgorithm.h
- NNTLib/GeneticAlgorithm.cpp

3.11 GeneticAlgorithmConfig Class Reference

Inheritance diagram for GeneticAlgorithmConfig:



Collaboration diagram for GeneticAlgorithmConfig:



Public Member Functions

- GeneticAlgorithmConfig ()

  Initializes a new instance of the GeneticAlgorithmConfig class.
- void PrintData ()

  Prints the data.
- bool IsConfigValid ()

  Determines whether [is configuration valid].

Public Attributes

- double ErrorThreshold

  The error threshold
- int MaxLoopCount

  The maximum loop count
- NNTLib::MutateEnum MutateType

  The mutate type
- NNTLib::CrossoverEnum CrossType

The cross type

- NNTLib::RouletteEnum RouletteType

  The roulette type

- int PopulationSize

  The population size

- int EltismCount

  The eltism count

- double MutationProbability

  The mutation probability

- double CrossoverProbability

  The crossover probability

- int MutateNodeCount

  The mutate node count

Protected Member Functions

- void HandleNameValue (std::string name, std::string value)

  Handles the name value.

3.11.1 Constructor & Destructor Documentation

3.11.1.1 GeneticAlgorithmConfig::GeneticAlgorithmConfig ( )

Initializes a new instance of the GeneticAlgorithmConfig class.

3.11.2 Member Function Documentation

3.11.2.1 void GeneticAlgorithmConfig::HandleNameValue (std::string *name*, std::string *value*) [protected], [virtual]

Handles the name value.

Parameters

name	The name.
value	The value.

Implements ConfigBase.

3.11.2.2 bool GeneticAlgorithmConfig::IsConfigValid ( ) [virtual]

Determines whether [is configuration valid].

Returns

Implements ConfigBase.

3.11.2.3 void GeneticAlgorithmConfig::PrintData ( ) [virtual]

Prints the data.

Implements ConfigBase.

3.11.3 Member Data Documentation

3.11.3.1 double GeneticAlgorithmConfig::CrossoverProbability

The crossover probability

3.11.3.2 NNTLib::CrossoverEnum GeneticAlgorithmConfig::CrossType

The cross type

3.11.3.3 int GeneticAlgorithmConfig::EltismCount

The eltism count

3.11.3.4 double GeneticAlgorithmConfig::ErrorThreshold

The error threshold

3.11.3.5 int GeneticAlgorithmConfig::MaxLoopCount

The maximum loop count

3.11.3.6 int GeneticAlgorithmConfig::MutateNodeCount

The mutate node count

3.11.3.7 NNTLib::MutateEnum GeneticAlgorithmConfig::MutateType

The mutate type

3.11.3.8 double GeneticAlgorithmConfig::MutationProbability

The mutation probability

3.11.3.9 int GeneticAlgorithmConfig::PopulationSize

The population size

3.11.3.10 NNTLib::RouletteEnum GeneticAlgorithmConfig::RouletteType

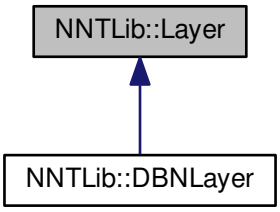
The roulette type

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

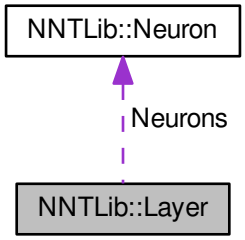
- NNTLib/GeneticAlgorithmConfig.h
- NNTLib/GeneticAlgorithmConfig.cpp

3.12 NNTLib::Layer Class Reference

Inheritance diagram for NNTLib::Layer:



Collaboration diagram for NNTLib::Layer:



Public Member Functions

- void Init (int inputsize, int neuronCount)

  Initializes the specified inputsize.
- Layer ()

  Initializes a new instance of the Layer class.
- ~Layer ()

  Finalizes an instance of the Layer class.
- Layer (const Layer &that)

  Initializes a new instance of the Layer class.
- Layer & operator= (const Layer &that)

  Operator=s the specified that.

Public Attributes

- int InputValuesCount

  The input values count

- int InputValuesCountWithBias

  The input values count with bias
- int NeuronCount

  The neuron count
- Neuron ∗ Neurons

  The neurons (Collection aller Neuronen auf dem Layer)
- double ∗ InputValues

  The input values (Collection mit Ausgaben des darunter liegenden Layers Li-1)
- double ∗ SumDeltaErrWeights

  The sum (delta ∗ error ∗ weights)

Protected Member Functions

- void copy (const Layer &that)

  Copies the specified that.
- void init ()

  Initializes this instance.
- void freeMem ()

  Frees the memory.

3.12.1 Constructor & Destructor Documentation

3.12.1.1 NNTLib::Layer::Layer ( )

Initializes a new instance of the Layer class.

3.12.1.2 NNTLib::Layer::∼Layer ( )

Finalizes an instance of the Layer class.

3.12.1.3 NNTLib::Layer::Layer (const Layer & *that*)

Initializes a new instance of the Layer class.

Parameters

that	The that.

3.12.2 Member Function Documentation

3.12.2.1 void NNTLib::Layer::copy (const Layer & *that*) [protected]

Copies the specified that.

Parameters

that	The that.

3.12.2.2 void NNTLib::Layer::freeMem ( ) [protected]

Frees the memory.

3.12.2.3 void NNTLib::Layer::init ( ) [protected]

Initializes this instance.

3.12.2.4 void NNTLib::Layer::Init (int *inputsize,* int *neuronCount*)

Initializes the specified inputsize.

Parameters

inputsize	The inputsize.
neuronCount	The neuron count.

3.12.2.5 Layer & NNTLib::Layer::operator= (const Layer & *that*)

Operator=s the specified that.

Parameters

that	The that.

Returns

3.12.3 Member Data Documentation

3.12.3.1 double* NNTLib::Layer::InputValues

The input values (Collection mit Ausgaben des darunter liegenden Layers Li-1)

3.12.3.2 int NNTLib::Layer::InputValuesCount

The input values count

3.12.3.3 int NNTLib::Layer::InputValuesCountWithBias

The input values count with bias

3.12.3.4 int NNTLib::Layer::NeuronCount

The neuron count

3.12.3.5 Neuron* NNTLib::Layer::Neurons

The neurons (Collection aller Neuronen auf dem Layer)

3.12.3.6 double* NNTLib::Layer::SumDeltaErrWeights

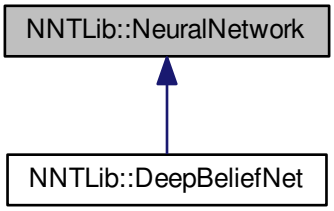
The sum (delta * error * weights)

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

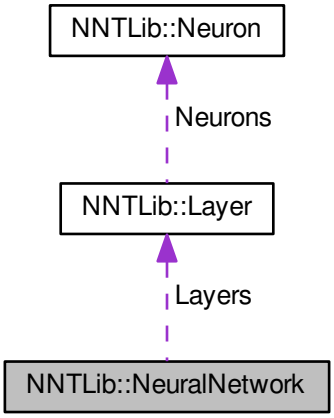
- NNTLib/Layer.h
- NNTLib/Layer.cpp

3.13 NNTLib::NeuralNetwork Class Reference

Inheritance diagram for NNTLib::NeuralNetwork:



Collaboration diagram for NNTLib::NeuralNetwork:



Public Member Functions

- NeuralNetwork (int *layers, int layercount, WeightInitEnum initType, FunctionEnum functionType)

  Initializes a new instance of the NeuralNetwork class.
- ~NeuralNetwork ()

  Finalizes an instance of the NeuralNetwork class.
- NeuralNetwork (const NeuralNetwork &that)

  Initializes a new instance of the NeuralNetwork class.
- NeuralNetwork & operator= (const NeuralNetwork &that)

Operator=s the specified that.

- bool operator< (const NeuralNetwork &net) const

  Operators the specified net.
- void InitWeights (WeightInitEnum initType)

  Initializes the weights.
- double GenerateRandomWeight (int weightCount)

  Generates the random weight.
- void SaveWeights (const std::string file)

  Saves the weights.
- void LoadWeights (const std::string file)

  Loads the weights.
- void Propagate (const double *input)

  Propagates the specified input.
- void CalculateMSE (const DataContainer &data)

  Calculates the mse.

Public Attributes

- int LayersCount

  The layers count
- double MeanSquareError

  The mean square error
- WeightInitEnum WeightInitType

  The weight initialize type
- FunctionEnum FunctionType

  The function type
- int TotalNeuronCount

  The total neuron count
- Layer * Layers

  The layers

Protected Member Functions

- void copy (const NeuralNetwork &that)

  Copies the specified that.
- void init ()

  Initializes this instance.
- void freeMem ()

  Frees the memory.

Protected Attributes

- std::mt19937 **generator**

3.13.1 Constructor & Destructor Documentation

3.13.1.1 NNTLib::NeuralNetwork::NeuralNetwork (int * *neuronsCountPerLayer*, int *layercount*, WeightInitEnum *initType*, FunctionEnum *functionType*)

Initializes a new instance of the NeuralNetwork class.

Parameters

neuronsCount↩ PerLayer	The hidden layers.
layercount	The hidden layercount.
initType	Type of the initialize.
functionType	Type of the function.

3.13.1.2 NNTLib::NeuralNetwork::~NeuralNetwork ( )

Finalizes an instance of the NeuralNetwork class.

3.13.1.3 NNTLib::NeuralNetwork::NeuralNetwork (const NeuralNetwork & *that*)

Initializes a new instance of the NeuralNetwork class.

Parameters

that	The that.

3.13.2 Member Function Documentation

3.13.2.1 void NNTLib::NeuralNetwork::CalculateMSE (const DataContainer & *data*)

Calculates the mse.

Parameters

data	The data.

3.13.2.2 void NNTLib::NeuralNetwork::copy (const NeuralNetwork & *that*) [protected]

Copies the specified that.

Parameters

that	The that.

3.13.2.3 void NNTLib::NeuralNetwork::freeMem ( ) [protected]

Frees the memory.

3.13.2.4 double NNTLib::NeuralNetwork::GenerateRandomWeight (int *weightCount*)

Generates the random weight.

Parameters

inputVector↩ CountWithBias	The input vector count with bias.

Returns

3.13.2.5 void NNTLib::NeuralNetwork::init ( ) [protected]

Initializes this instance.

3.13.2.6 void NNTLib::NeuralNetwork::InitWeights (WeightInitEnum *initType*)

Initializes the weights.

Parameters

initType	Type of the initialize.

3.13.2.7 void NNTLib::NeuralNetwork::LoadWeights (const std::string *file*)

Loads the weights.

Parameters

file	The file.

3.13.2.8 bool NNTLib::NeuralNetwork::operator< (const **NeuralNetwork** & *net*) const

Operators the specified net.

Parameters

net	The net.

Returns

3.13.2.9 **NeuralNetwork** & NNTLib::NeuralNetwork::operator= (const **NeuralNetwork** & *that*)

Operator=s the specified that.

Parameters

that	The that.

Returns

3.13.2.10 void NNTLib::NeuralNetwork::Propagate (const double * *input*)

Propagates the specified input.

Parameters

input	The input.

neuron->Bias

3.13.2.11 void NNTLib::NeuralNetwork::SaveWeights (const std::string *file*)

Saves the weights.

Parameters

file	The file.

3.13.3 Member Data Documentation

3.13.3.1 FunctionEnum NNTLib::NeuralNetwork::FunctionType

The function type

3.13.3.2 Layer* NNTLib::NeuralNetwork::Layers

The layers

3.13.3.3 int NNTLib::NeuralNetwork::LayersCount

The layers count

3.13.3.4 double NNTLib::NeuralNetwork::MeanSquareError

The mean square error

3.13.3.5 int NNTLib::NeuralNetwork::TotalNeuronCount

The total neuron count

3.13.3.6 WeightInitEnum NNTLib::NeuralNetwork::WeightInitType

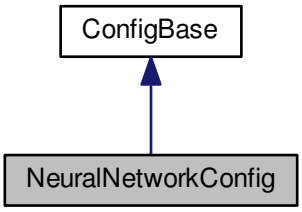
The weight initialize type

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

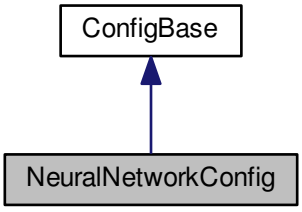
- NNTLib/NeuralNetwork.h
- NNTLib/NeuralNetwork.cpp

3.14 NeuralNetworkConfig Class Reference

Inheritance diagram for NeuralNetworkConfig:



Collaboration diagram for NeuralNetworkConfig:



Public Member Functions

- NeuralNetworkConfig ()

  Initializes a new instance of the NeuralNetworkConfig class.
- ~NeuralNetworkConfig ()

  Finalizes an instance of the NeuralNetworkConfig class.
- NeuralNetworkConfig (const NeuralNetworkConfig &that)

  Initializes a new instance of the NeuralNetworkConfig class.
- NeuralNetworkConfig & operator= (const NeuralNetworkConfig &that)

  Operator=s the specified that.
- void PrintData ()

  Prints the data.
- bool IsConfigValid ()

  Determines whether [is configuration valid].

Public Attributes

- int LayerCount

The layer count

- NNTLib::FunctionEnum FunctionType

  The function type

- NNTLib::WeightInitEnum WeightInitType

  The weight initialize type

- int * LayerNeuronCount

  The layer neuron count

Protected Member Functions

- void HandleNameValue (std::string name, std::string value)

  Handles the name value.

3.14.1 Constructor & Destructor Documentation

3.14.1.1 NeuralNetworkConfig::NeuralNetworkConfig ( )

Initializes a new instance of the NeuralNetworkConfig class.

3.14.1.2 NeuralNetworkConfig::~NeuralNetworkConfig ( )

Finalizes an instance of the NeuralNetworkConfig class.

3.14.1.3 NeuralNetworkConfig::NeuralNetworkConfig (const NeuralNetworkConfig & *that*)

Initializes a new instance of the NeuralNetworkConfig class.

Parameters

that	The that.

3.14.2 Member Function Documentation

3.14.2.1 void NeuralNetworkConfig::HandleNameValue (std::string *name*, std::string *value*) [protected], [virtual]

Handles the name value.

Parameters

name	The name.
value	The value.

Implements ConfigBase.

3.14.2.2 bool NeuralNetworkConfig::IsConfigValid ( ) [virtual]

Determines whether [is configuration valid].

Returns

Implements ConfigBase.

3.14.2.3 NeuralNetworkConfig & NeuralNetworkConfig::operator= (const NeuralNetworkConfig & *that*)

Operator=s the specified that.

Parameters

that	The that.

Returns

3.14.2.4 void NeuralNetworkConfig::PrintData ( ) [virtual]

Prints the data.

Implements ConfigBase.

3.14.3 Member Data Documentation

3.14.3.1 NNTLib::FunctionEnum NeuralNetworkConfig::FunctionType

The function type

3.14.3.2 int NeuralNetworkConfig::LayerCount

The layer count

3.14.3.3 int* NeuralNetworkConfig::LayerNeuronCount

The layer neuron count

3.14.3.4 NNTLib::WeightInitEnum NeuralNetworkConfig::WeightInitType

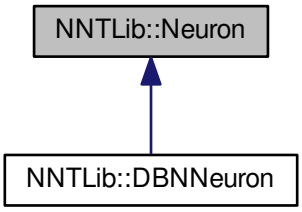
The weight initialize type

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

- NNTLib/NeuralNetworkConfig.h
- NNTLib/NeuralNetworkConfig.cpp

3.15 NNTLib::Neuron Class Reference

Inheritance diagram for NNTLib::Neuron:



Public Member Functions

- Neuron ()
  Initializes a new instance of the Neuron class.
- ~Neuron ()
  Finalizes an instance of the Neuron class.
- Neuron (const Neuron &that)
  Initializes a new instance of the Neuron class.
- Neuron & operator= (const Neuron &that)
  Operator=s the specified that.
- void Init (int weightCount)
  Initializes the specified input vector count.

Public Attributes

- int WeightCount
  The weight count (Anzahl eingehende Gewichte (mit Schwellenwert)
- double Output
  The output (Ausgabe des Neurons)
- double * Weights
  The weights (Collection aller Gewichte inklusive Schwellenwert Gewicht)
- double * LastDeltaWeights
  The last delta weights (letzte Gewichteurng wird fr Momentum Verfahren bentigt)
- double * DeltaWeights
  The delta weights (Summe aller Gewichtserungen innerhalb einer Batchsize, wird bei online training nicht verwendet)
- double **Bias**

Protected Member Functions

- void copy (const Neuron &that)
  Copies the specified that.
- void init ()
  Initializes this instance.
- void freeMem ()
  Frees the memory.

3.15.1 Constructor & Destructor Documentation

3.15.1.1 NNTLib::Neuron::Neuron ( )

Initializes a new instance of the Neuron class.

3.15.1.2 NNTLib::Neuron::~Neuron ( )

Finalizes an instance of the Neuron class.

3.15.1.3 NNTLib::Neuron::Neuron (const Neuron & *that*)

Initializes a new instance of the Neuron class.

Parameters

that	The that.

3.15.2 Member Function Documentation

3.15.2.1 void NNTLib::Neuron::copy (const Neuron & *that*) [protected]

Copies the specified that.

Parameters

that	The that.

3.15.2.2 void NNTLib::Neuron::freeMem ( ) [protected]

Frees the memory.

3.15.2.3 void NNTLib::Neuron::init ( ) [protected]

Initializes this instance.

3.15.2.4 void NNTLib::Neuron::Init (int *weightCount*)

Initializes the specified input vector count.

Parameters

inputVector↩ Count	The input vector count.

3.15.2.5 Neuron & NNTLib::Neuron::operator= (const Neuron & *that*)

Operator=s the specified that.

Parameters

that	The that.

Returns

3.15.3 Member Data Documentation

3.15.3.1 double* NNTLib::Neuron::DeltaWeights

The delta weights (Summe aller Gewichtserungen innerhalb einer Batchsize, wird bei online training nicht verwendet)

3.15.3.2 double* NNTLib::Neuron::LastDeltaWeights

The last delta weights (letzte Gewichteurng wird fr Momentum Verfahren bentigt)

3.15.3.3 double NNTLib::Neuron::Output

The output (Ausgabe des Neurons)

3.15.3.4 int NNTLib::Neuron::WeightCount

The weight count (Anzahl eingehende Gewichte (mit Schwellenwert)

3.15.3.5 double* NNTLib::Neuron::Weights

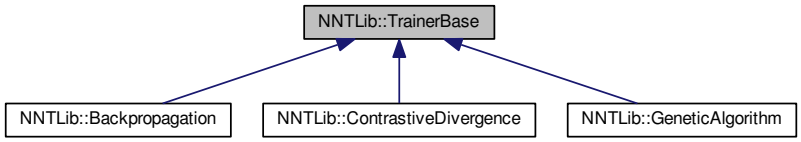
The weights (Collection aller Gewichte inklusive Schwellenwert Gewicht)

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

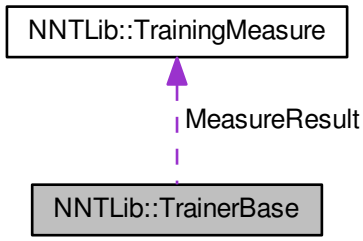
- NNTLib/Neuron.h
- NNTLib/Neuron.cpp

3.16 NNTLib::TrainerBase Class Reference

Inheritance diagram for NNTLib::TrainerBase:



Collaboration diagram for NNTLib::TrainerBase:



Public Member Functions

- void SaveResult (std::string file, unsigned long long timeOffsetInMs=0)

  Saves the mse result.
- TrainerBase ()

  Initializes a new instance of the TrainerBase class.
- ~TrainerBase ()

  Finalizes an instance of the TrainerBase class.

Public Attributes

- TrainingMeasure * MeasureResult

  The measure result
- int MeasureResultLenght

  The measure result lenght
- int MeasureFilledResultLenght

  The measure filled result lenght

Protected Member Functions

- void initMeasureResult (int lenght)

  Initializes the measure result.

3.16.1 Constructor & Destructor Documentation

3.16.1.1 NNTLib::TrainerBase::TrainerBase ( )

Initializes a new instance of the TrainerBase class.

3.16.1.2 NNTLib::TrainerBase::~TrainerBase ( )

Finalizes an instance of the TrainerBase class.

3.16.2 Member Function Documentation

3.16.2.1 void NNTLib::TrainerBase::initMeasureResult (int *lenght*) [protected]

Initializes the measure result.

Parameters

lenght	The lenght.

3.16.2.2 void NNTLib::TrainerBase::SaveResult (std::string *file*, unsigned long long *timeOffsetInMs* = 0)

Saves the mse result.

Parameters

file	The file.
timeOffsetInMs	The time offset in ms.

3.16.3 Member Data Documentation

3.16.3.1 int NNTLib::TrainerBase::MeasureFilledResultLenght

The measure filled result lenght

3.16.3.2 TrainingMeasure* NNTLib::TrainerBase::MeasureResult

The measure result

3.16.3.3 int NNTLib::TrainerBase::MeasureResultLenght

The measure result lenght

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

- NNTLib/TrainerBase.h
- NNTLib/TrainerBase.cpp

3.17 NNTLib::TrainingMeasure Struct Reference

Public Member Functions

- TrainingMeasure ()

  Initializes a new instance of the TrainingMeasure struct.
- ~TrainingMeasure ()

  Finalizes an instance of the TrainingMeasure class.

Public Attributes

- double MeanSquareError

  The mean square error
- unsigned long long ExecuteTime

  The execute time

3.17.1 Constructor & Destructor Documentation

3.17.1.1 NNTLib::TrainingMeasure::TrainingMeasure ( )

Initializes a new instance of the TrainingMeasure struct.

3.17.1.2 NNTLib::TrainingMeasure::~TrainingMeasure ( )

Finalizes an instance of the TrainingMeasure class.

3.17.2 Member Data Documentation

3.17.2.1 unsigned long long NNTLib::TrainingMeasure::ExecuteTime

The execute time

3.17.2.2 double NNTLib::TrainingMeasure::MeanSquareError

The mean square error

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

- NNTLib/TrainingMeasure.h
- NNTLib/TrainingMeasure.cpp