

Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Computer Programming

«Neural Network»

author	Olivier Halupczok
instructor	dr inż. Piotr Fabian
year	021-2022
lab group	even Tuesday, 10:15 – 11:45
deadline	2022-06-24

1 Project's topic

Simple Neural Network defined as a class that supports Neurons., configuring a user-defined connection structure, selected learning method.

2 Analysis of the task

The task focuses on the analyzing data given as an input dataset and predicting the feature of the object. It requires implementation of the learning algorithm and minimizing loss of the entire neural network.

2.1 Data structures

Program has used a lot of vectors and objects of custom classes. Vectors were used to store serial data and other structures. I implemented Neuron as a class as well to make calculations easier and to order them to the proper objects.

2.2 Algorithms

The program loads input, create an instance of Neural Network Class. While calling a constructor of the Neural Network, it creates Neurons of the specified amount in the constructor parameter. When the train method is called then Network is iterating throughout the input dataset and call the loss of every iteration(epoch) and on its basis adjust the weights and biases of every Neuron.

3 External specification

This is a command line program. You can execute a program by using compiled .exe file or by using make in the e.g. Bash terminal. The program requires input datasets specified in the main.cpp

Program call

```
program
program -h
```

prints a short manual. Program called with incorrect parameters prints an error message and prints help.

After execution you can find output file in the following direction:

/logs/output.csv

You have to ensure yourself '/logs' directory exists, otherwise output file won't be created.

4 Internal specification

The program is implemented with object-oriented, structural and functional paradigm. User interface is separated from program's logic.

4.1 Program overview

The `main` function checks number of arguments passed to the program and creates a new `Options` object, which constructor check parameters of the program. If the verification is negative, an appropriate message is printed. In case of positive verification, data are read and saved into vector, which is passed, simultaneously with the options object, to the constructor of an `GenerationsFactory` object. Then `GenerationsFactory::makeGenerations` method performs genetic algorithm by creating first generation using objects of classes: `Generation`, `Individual`, and then it creates every another generation of individuals. Finally the program prints data into an output file.

4.2 Description of types and functions

Description of types and functions is moved to the appendix.

5 Testing

The program has been tested with various types of files. Incorrect files (with no numbers, numbers in incorrect format, strings with some invalid whitespaces, ...) are detected and an error message is printed. An empty input file does not cause failure – an empty output file is created. Maximal number value(**double**) in an input file is approximately $1.8e+308$. Maximal input file size handled by the program is 1.57 GB. Larger files result in a bad allocation error. The program has no memory leaks.

6 Conclusions

The program implements a solution to a knapsack problem using genetic algorithm. The most challenging task is creating a crossover system where a

lot of data can be processed iteratively.

For some parameters the program elaborates incorrect results on some machines. This is caused by specification of the algorithm which does not includes an evolution coeeficient. Algorithm also base on a small dose of randomness, and it can be succesfull if generation/individuals number is insufficient.

Appendix

Description of types and functions

knapsackProblem

0.1

Generated by Doxygen 1.9.1

1 Neural Network	1
1.1 Olivier Halupczok	1
1.1.1 To build and execute the whole main and compile all necessary libraries you can simply execute make command in the terminal.	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Class Documentation	9
5.1 CSV_Logger Class Reference	9
5.1.1 Constructor & Destructor Documentation	10
5.1.1.1 CSV_Logger()	10
5.2 Logger Class Reference	10
5.2.1 Member Function Documentation	11
5.2.1.1 operator<<() [1/2]	11
5.2.1.2 operator<<() [2/2]	11
5.2.1.3 setOutputStream()	12
5.3 NetworkResult Class Reference	12
5.3.1 Constructor & Destructor Documentation	12
5.3.1.1 NetworkResult()	12
5.3.2 Member Function Documentation	13
5.3.2.1 getLoss()	13
5.3.2.2 getPredictions()	13
5.3.3 Friends And Related Function Documentation	13
5.3.3.1 operator<<	13
5.4 NeuralNetwork Class Reference	14
5.4.1 Constructor & Destructor Documentation	14
5.4.1.1 NeuralNetwork()	14
5.4.2 Member Function Documentation	15
5.4.2.1 feedforward()	15
5.4.2.2 train()	15
5.5 Neuron Class Reference	15
5.5.1 Constructor & Destructor Documentation	16
5.5.1.1 Neuron()	16
5.5.2 Member Function Documentation	16
5.5.2.1 adjustBias()	16
5.5.2.2 adjustWeight()	17
5.5.2.3 feedforward()	17

5.5.2.4 <code>getBias()</code>	17
5.5.2.5 <code>getOutput()</code>	18
5.5.2.6 <code>getTotal()</code>	18
5.5.2.7 <code>getWeights()</code>	18
6 File Documentation	19
6.1 <code>logger.cpp</code> File Reference	19
6.1.1 Detailed Description	20
6.2 <code>logger.h</code> File Reference	20
6.2.1 Detailed Description	21
6.3 <code>main.cpp</code> File Reference	21
6.3.1 Detailed Description	22
6.4 <code>mathFuncs.cpp</code> File Reference	22
6.4.1 Detailed Description	23
6.4.2 Function Documentation	23
6.4.2.1 <code>calc_mse_loss()</code>	23
6.4.2.2 <code>deriv_sigmoid()</code>	24
6.4.2.3 <code>dotProductOf2Vectors()</code>	24
6.4.2.4 <code>lossDeriv_outDeriv()</code>	25
6.4.2.5 <code>sigmoid()</code>	25
6.5 <code>mathFuncs.h</code> File Reference	25
6.5.1 Detailed Description	26
6.5.2 Function Documentation	27
6.5.2.1 <code>calc_mse_loss()</code>	27
6.5.2.2 <code>deriv_sigmoid()</code>	27
6.5.2.3 <code>dotProductOf2Vectors()</code>	28
6.5.2.4 <code>lossDeriv_outDeriv()</code>	28
6.5.2.5 <code>sigmoid()</code>	28
6.6 <code>networkResult.cpp</code> File Reference	29
6.6.1 Detailed Description	29
6.6.2 Function Documentation	30
6.6.2.1 <code>operator<<()</code>	30
6.7 <code>networkResult.h</code> File Reference	30
6.7.1 Detailed Description	31
6.8 <code>neuralNetwork.cpp</code> File Reference	31
6.8.1 Detailed Description	32
6.9 <code>neuralNetwork.h</code> File Reference	32
6.9.1 Detailed Description	33
6.10 <code>neuron.cpp</code> File Reference	34
6.10.1 Detailed Description	34
6.11 <code>neuron.h</code> File Reference	35
6.11.1 Detailed Description	35

Chapter 1

Neural Network

1.1 Olivier Halupczok

1.1.1 To build and execute the whole main and compile all necessary libraries you can simply execute make command in the terminal.

TO DO:

Use custom implementation of vector inheriting from `std::vector` with `*` operator leading to dot product of vectors
Use Model class to shape Net in a custom way Add Layer class Use passing arg by reference in dot product in `mathsFUNcs`

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Logger	10
CSV_Logger	9
NetworkResult	12
NeuralNetwork	14
Neuron	15

Chapter 3

Class Index

3.1 Class List

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

CSV_Logger	9
Logger	10
NetworkResult	12
NeuralNetwork	14
Neuron	15

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

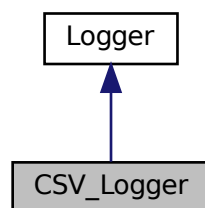
logger.cpp	Cpp file with the definitions of the logger library	19
logger.h	Declarations of the logger library	20
main.cpp	21
mathFuncs.cpp	Library with math functions used in the program	22
mathFuncs.h	Library with math functions used in the program	25
networkResult.cpp	29
networkResult.h	30
neuralNetwork.cpp	31
neuralNetwork.h	32
neuron.cpp	34
neuron.h	35

Chapter 5

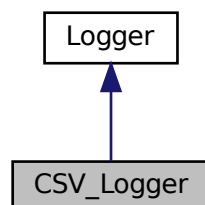
Class Documentation

5.1 CSV_Logger Class Reference

Inheritance diagram for CSV_Logger:



Collaboration diagram for CSV_Logger:



Public Member Functions

- [CSV_Logger](#) (std::string path)
Construct a new csv logger object.
- [~CSV_Logger](#) ()
Destroy the csv logger object and close opened file.

5.1.1 Constructor & Destructor Documentation

5.1.1.1 CSV_Logger()

```
CSV_Logger::CSV_Logger (
    std::string path )
```

Construct a new csv logger object.

Parameters

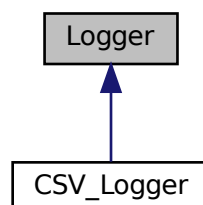
<i>path</i>	to the file
-------------	-------------

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

- [logger.h](#)
- [logger.cpp](#)

5.2 Logger Class Reference

Inheritance diagram for Logger:



Public Member Functions

- [Logger](#) ()
Construct a new [Logger](#) object.
- void [setOutputStream](#) (std::ostream &stream)
Set the Output Stream property.
- template<class T >
[Logger](#) & [operator<<](#) (T &&dataToLog)
Log data with << operator.
- [Logger](#) & [operator<<](#) (std::ostream &(*manip)(std::ostream &))
operator with manip definition to let [Logger](#) handle io manipulators

5.2.1 Member Function Documentation

5.2.1.1 [operator<<\(\)](#) [1/2]

```
Logger & Logger::operator<< (
    std::ostream &(*) (std::ostream &) manip )
```

operator with manip definition to let [Logger](#) handle io manipulators

Parameters

<i>manip</i>	io manipulators like 'std::endl'
--------------	----------------------------------

Returns

[Logger](#)& It returns instance of currently using object

5.2.1.2 [operator<<\(\)](#) [2/2]

```
template<class T >
Logger & Logger::operator<< (
    T && dataToLog )
```

Log data with << operator.

Template Parameters

<i>T</i>	template to let data of many types to be logged
----------	---

Parameters

<i>dataToLog</i>	Data to be printed with the logger
------------------	------------------------------------

Returns

[Logger](#) & It returns the whole instance of the object

5.2.1.3 setOutputStream()

```
void Logger::setOutputStream (
    std::ostream & stream )
```

Set the Output Stream property.

Parameters

<i>stream</i>	which the logs will be forwarded to
---------------	-------------------------------------

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

- [logger.h](#)
- [logger.cpp](#)

5.3 NetworkResult Class Reference

Public Member Functions

- [NetworkResult](#) (double _loss, std::vector< double > &_predictions)
Construct a new Network Result object.
- double [getLoss](#) ()
Get the Loss property.
- std::vector< double > [getPredictions](#) ()
Get the Predictions property.

Friends

- std::ostream & [operator<<](#) (std::ostream &stream, [NetworkResult](#) &result)
print loss to the stream

5.3.1 Constructor & Destructor Documentation

5.3.1.1 NetworkResult()

```
NetworkResult::NetworkResult (
    double _loss,
    std::vector< double > & _predictions )
```

Construct a new Network Result object.

Parameters

<code>_loss</code>	loss of the given epoch
<code>_predictions</code>	predictions of the given epoch

5.3.2 Member Function Documentation

5.3.2.1 getLoss()

```
double NetworkResult::getLoss ( )
```

Get the Loss property.

Returns

double

5.3.2.2 getPredictions()

```
std::vector< double > NetworkResult::getPredictions ( )
```

Get the Predictions property.

Returns

std::vector<double>

5.3.3 Friends And Related Function Documentation

5.3.3.1 operator<<

```
std::ostream& operator<< (
    std::ostream & stream,
    NetworkResult & result ) [friend]
```

print loss to the stream

Parameters

<code><i>stream</i></code>	stream which the data is printed into
<code><i>result</i></code>	result of the given epoch

Returns

std::ostream&

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

- [networkResult.h](#)
- [networkResult.cpp](#)

5.4 NeuralNetwork Class Reference

Public Member Functions

- [NeuralNetwork](#) (size_t numOfNeurons, std::function< double(double)> activationFunc, std::function< double(double)> activationFuncDeriv, double learningRate)
Construct a new Neural Network object.
- double [feedforward](#) (std::vector< double > inputs)
feedforward every neuron from hidden layer
- void [train](#) (long long int epochs, std::vector< std::vector< double >> inputData, std::vector< double > labels, std::function< void([NetworkResult](#))> callback)
train neural net

5.4.1 Constructor & Destructor Documentation

5.4.1.1 NeuralNetwork()

```
NeuralNetwork::NeuralNetwork (
    size_t numOfNeurons,
    std::function< double(double)> activationFunc,
    std::function< double(double)> activationFuncDeriv,
    double learningRate )
```

Construct a new Neural Network object.

random number engine

Parameters

<i>numOfNeurons</i>	num of Neurons to create
<i>activationFunc</i>	function to activate neuron
<i>activationFuncDeriv</i>	derivative of activation function
<i>learningRate</i>	learning rate of the neurons

5.4.2 Member Function Documentation

5.4.2.1 feedforward()

```
double NeuralNetwork::feedforward (
    std::vector< double > inputs )
```

feedforward every neuron from hidden layer

Parameters

<i>inputs</i>	from input layer
---------------	------------------

Returns

double output of the output neuron

5.4.2.2 train()

```
void NeuralNetwork::train (
    long long int epochs,
    std::vector< std::vector< double >> inputData,
    std::vector< double > labels,
    std::function< void(NetworkResult)> callback )
```

train neural net

Parameters

<i>epochs</i>	determines the number of iterations through the whole dataset
<i>inputData</i>	dataset to train on
<i>labels</i>	evaluate training process
<i>callback</i>	callback after training

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

- [neuralNetwork.h](#)
- [neuralNetwork.cpp](#)

5.5 Neuron Class Reference

Public Member Functions

- [Neuron](#) (std::vector< double > weightsToInit, double biasToInit, std::function< double(double)> activationFuncToInit, std::function< double(double)> activationFuncDeriv, double learningRateToInit)

- Construct a new [Neuron](#) object.
- double [feedforward](#) (std::vector< double > inputsToFeed)
 - Feedforward with neurons from previous layers.
- double [getTotal](#) ()
 - Get the Total property.
- std::vector< double > [getWeights](#) ()
 - Get the Weights property.
- double [getBias](#) ()
 - Get the Bias property.
- void [adjustWeight](#) (size_t index, [Neuron](#) outputNeuron, double lossDeriv_outDeriv_calced, size_t iterator)
 - use backprop to adjust weight of specified index and to train network by doing so
- void [adjustBias](#) ([Neuron](#) outputNeuron, double lossDeriv_outDeriv_calced, size_t iterator)
 - use backprop to adjust bias and to train network
- double [getOutput](#) ()
 - Get the Output calculated during feedforward's execution.

5.5.1 Constructor & Destructor Documentation

5.5.1.1 Neuron()

```
Neuron::Neuron (
    std::vector< double > weightsToInit,
    double biasToInit,
    std::function< double(double)> activationFuncToInit,
    std::function< double(double)> activationFuncDeriv,
    double learningRateToInit = LEARNING_DEFAULT_RATE )
```

Construct a new [Neuron](#) object.

Parameters

<i>weightsToInit</i>	Weigths of inputs
<i>biasToInit</i>	Bias to calculate feedforward's total
<i>activationFuncToInit</i>	

5.5.2 Member Function Documentation

5.5.2.1 adjustBias()

```
void Neuron::adjustBias (
    Neuron outputNeuron,
    double lossDeriv_outDeriv_calced,
    size_t iterator )
```

use backprop to adjust bias and to train network

Parameters

<i>outputNeuron</i>	
<i>lossDeriv_outDeriv</i>	
<i>iterator</i>	

5.5.2.2 adjustWeight()

```
void Neuron::adjustWeight (
    size_t index,
    Neuron outputNeuron,
    double lossDeriv_outDeriv_calced,
    size_t iterator )
```

use backprop to adjust weight of specified index and to train network by doing so

Parameters

<i>index</i>	
<i>outputNeuron</i>	
<i>lossDeriv_outDeriv</i>	
<i>iterator</i>	

5.5.2.3 feedforward()

```
double Neuron::feedforward (
    std::vector< double > inputsToFeed )
```

Feedforward with neurons from previous layers.

Parameters

<i>inputs</i>	values of previous neurons
---------------	----------------------------

Returns

double total value of neuron

5.5.2.4 getBias()

```
double Neuron::getBias ( )
```

Get the Bias property.

Returns

double bias of the neuron

5.5.2.5 getOutput()

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

Get the Output calculated during feedforward's execution.

Returns

double - output value

5.5.2.6 getTotal()

```
double Neuron::getTotal ( )
```

Get the Total property.

Returns

double - sum of: dot product of inputs and weights, and bias

5.5.2.7 getWeights()

```
std::vector< double > Neuron::getWeights ( )
```

Get the Weights property.

Returns

std::vector<double> of weights

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

- [neuron.h](#)
- [neuron.cpp](#)

Chapter 6

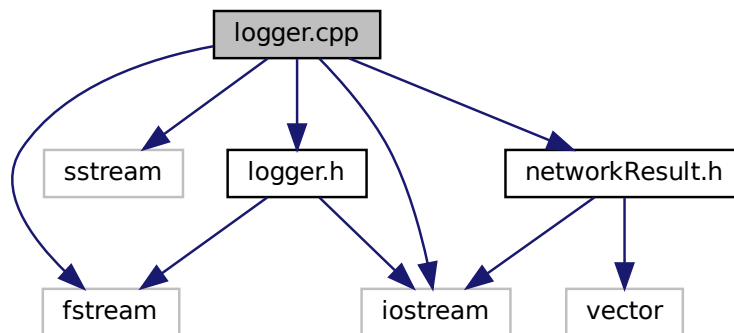
File Documentation

6.1 logger.cpp File Reference

cpp file with the definitions of the logger library

```
#include <iostream>
#include <sstream>
#include <fstream>
#include "networkResult.h"
#include "logger.h"
```

Include dependency graph for logger.cpp:



Functions

- template `Logger` & `Logger::operator<<< std::string >` (`std::string &&dataToLog`)

6.1.1 Detailed Description

cpp file with the definitions of the logger library

Author

Olivier Halupczok

Version

0.1

Date

2022-06-18

Copyright

Copyright (c) 2022

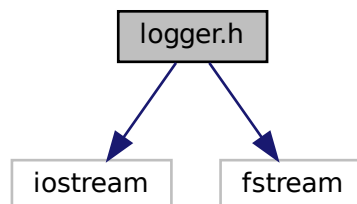
6.2 logger.h File Reference

declarations of the logger library

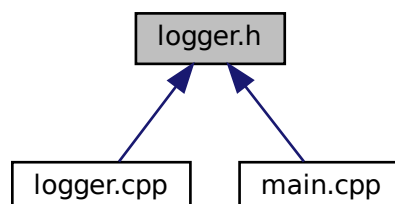
```
#include <iostream>
```

```
#include <fstream>
```

Include dependency graph for logger.h:



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



Classes

- class [Logger](#)
- class [CSV_Logger](#)

6.2.1 Detailed Description

declarations of the logger library

Author

Olivier Halupczok

Version

0.1

Date

2022-06-18

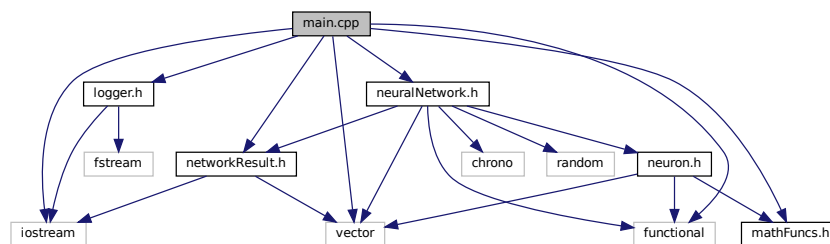
Copyright

Copyright (c) 2022

6.3 main.cpp File Reference

```
#include <iostream>
#include <vector>
#include <functional>
#include "mathFuncs.h"
#include "networkResult.h"
#include "neuralNetwork.h"
#include "logger.h"
```

Include dependency graph for main.cpp:



Functions

- void **logResults** ([NetworkResult](#) res)
- int **main** (int argc, char const *argv[])

Variables

- size_t **epochCount** = 1
- [CSV_Logger](#) **logger** ("logs/output.csv")

6.3.1 Detailed Description

Author

Olivier Halupczok

Version

0.1

Date

2022-06-11

Copyright

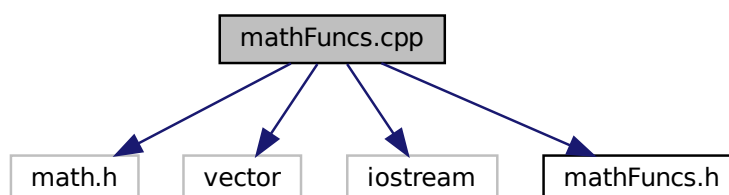
Copyright (c) 2022

6.4 mathFuncs.cpp File Reference

Library with math functions used in the program.

```
#include <math.h>
#include <vector>
#include <iostream>
#include "mathFuncs.h"
```

Include dependency graph for mathFuncs.cpp:



Functions

- `std::string invArgVectorsMsg` (`std::string nameOfFunc`)
- `double sigmoid` (`double arg`)
return value of sigmoid function ($1/(1 + \exp(-arg))$)
- `double deriv_sigmoid` (`double arg`)
return value of derivative of sigmoid
- `double dotProductOf2Vectors` (`std::vector< double > vector1`, `std::vector< double > vector2`)
return dot product of two two components vectors
- `double calc_mse_loss` (`std::vector< double > outputTrue`, `std::vector< double > outputPredicted`)
calculate mean squarred error of the neural network
- `double lossDeriv_outDeriv` (`double label`, `double output`)
it calculates value of the derivative of the Loss' function of the whole network divided by the derivative of output value's(of the entire network) function

Variables

- `const std::string INV_ARG_MSG` = "Invalid argument: "
- `const std::string INV_VECTORS_MSG` = " has to be executed with 2 vectors of the same length"
- `const std::string DOT_PRODUCT` = "dot product"
- `const std::string MSE` = "MSE loss calculation"

6.4.1 Detailed Description

Library with math functions used in the program.

Author

Olivier Halupczok

Version

0.1

Date

2022-06-18

Copyright

Copyright (c) 2022

6.4.2 Function Documentation

6.4.2.1 `calc_mse_loss()`

```
double calc_mse_loss (
    std::vector< double > outputTrue,
    std::vector< double > outputPredicted )
```

calculate mean squarred error of the neural network

Parameters

<i>outputTrue</i>	labels of data
<i>outputPredicted</i>	guesses of network

Returns

double return mean squarred error

6.4.2.2 deriv_sigmoid()

```
double deriv_sigmoid (  
    double arg )
```

return value of derivative of sigmoid

Parameters

<i>arg</i>	
------------	--

Returns

double

6.4.2.3 dotProductOf2Vectors()

```
double dotProductOf2Vectors (  
    std::vector< double > vector1,  
    std::vector< double > vector2 )
```

return dot product of two two components vectors

Parameters

<i>vector1</i>	
<i>vector2</i>	

Returns

double dot product

6.4.2.4 lossDeriv_outDeriv()

```
double lossDeriv_outDeriv (
    double label,
    double output )
```

it calculates value of the derivative of the Loss' function of the whole network divided by the derivative of output value's(of the entire network) function

Parameters

<i>label</i>	labels of the learning dataset
<i>output</i>	output value of net

Returns

double calculated derivative

6.4.2.5 sigmoid()

```
double sigmoid (
    double arg )
```

return value of sigmoid function ($1/(1 + \exp(-arg))$)

Parameters

<i>arg</i>	
------------	--

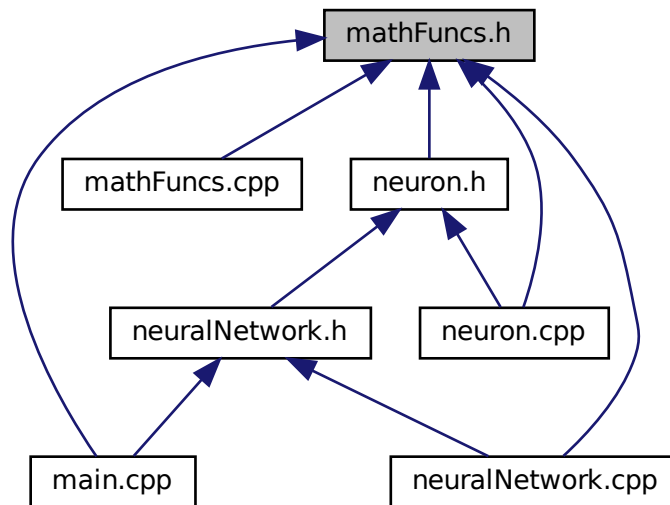
Returns

double

6.5 mathFuncs.h File Reference

library with math functions used in the program

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



Functions

- double [sigmoid](#) (double arg)
return value of sigmoid function ($1/(1 + \exp(-arg))$)
- double [deriv_sigmoid](#) (double arg)
return value of derivative of sigmoid
- double [dotProductOf2Vectors](#) (std::vector< double > vector1, std::vector< double > vector2)
return dot product of two two components vectors
- double [calc_mse_loss](#) (std::vector< double > outputTrue, std::vector< double > outputPredicted)
calculate mean squarred error of the neural network
- double [lossDeriv_outDeriv](#) (double label, double output)
it calculates value of the derivative of the Loss' function of the whole network divided by the derivative of output value's(of the entire network) function

6.5.1 Detailed Description

library with math functions used in the program

Author

Olivier Halupczok

Version

0.1

Date

2022-06-12

Copyright

Copyright (c) 2022

6.5.2 Function Documentation**6.5.2.1 calc_mse_loss()**

```
double calc_mse_loss (
    std::vector< double > outputTrue,
    std::vector< double > outputPredicted )
```

calculate mean squarred error of the neural network

Parameters

<i>outputTrue</i>	labels of data
<i>outputPredicted</i>	guesses of network

Returns

double return mean squarred error

6.5.2.2 deriv_sigmoid()

```
double deriv_sigmoid (
    double arg )
```

return value of derivative of sigmoid

Parameters

<i>arg</i>	
------------	--

Returns

double

6.5.2.3 dotProductOf2Vectors()

```
double dotProductOf2Vectors (
    std::vector< double > vector1,
    std::vector< double > vector2 )
```

return dot product of two two components vectors

Parameters

<i>vector1</i>	
<i>vector2</i>	

Returns

double dot product

6.5.2.4 lossDeriv_outDeriv()

```
double lossDeriv_outDeriv (
    double label,
    double output )
```

it calculates value of the derivative of the Loss' function of the whole network divided by the derivative of output value's(of the entire network) function

Parameters

<i>label</i>	labels of the learning dataset
<i>output</i>	output value of net

Returns

double calculated derivative

6.5.2.5 sigmoid()

```
double sigmoid (
    double arg )
```

return value of sigmoid function ($1/(1 + \exp(-arg))$)

Parameters

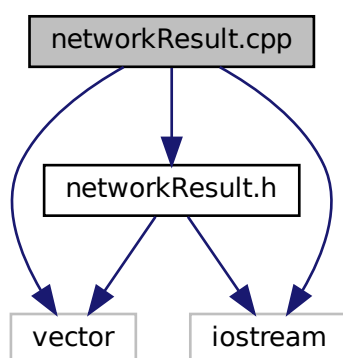
<i>arg</i>	
------------	--

Returns

double

6.6 networkResult.cpp File Reference

```
#include <vector>
#include <iostream>
#include "networkResult.h"
Include dependency graph for networkResult.cpp:
```



Functions

- std::ostream & [operator<<](#) (std::ostream &stream, [NetworkResult](#) &result)

6.6.1 Detailed Description

Author

Olivier Halupczok

Version

0.1

Date

2022-06-18

Copyright

Copyright (c) 2022

6.6.2 Function Documentation

6.6.2.1 `operator<<()`

```
std::ostream& operator<< (
    std::ostream & stream,
    NetworkResult & result )
```

Parameters

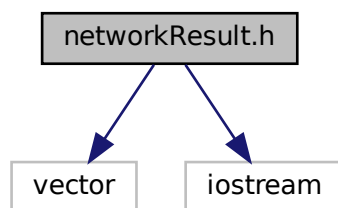
<i>stream</i>	stream which the data is printed into
<i>result</i>	result of the given epoch

Returns

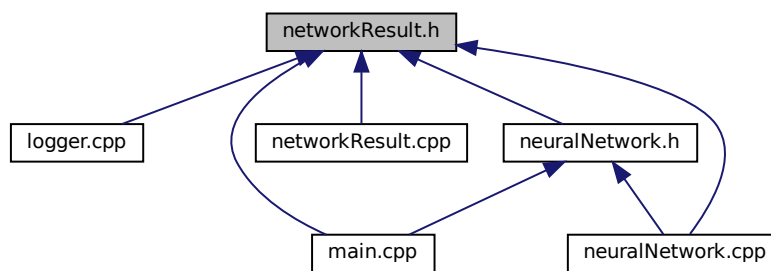
`std::ostream&`

6.7 `networkResult.h` File Reference

```
#include <vector>
#include <iostream>
Include dependency graph for networkResult.h:
```



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



Classes

- class [NetworkResult](#)

6.7.1 Detailed Description

Author

Olivier Halupczok

Version

0.1

Date

2022-06-18

Copyright

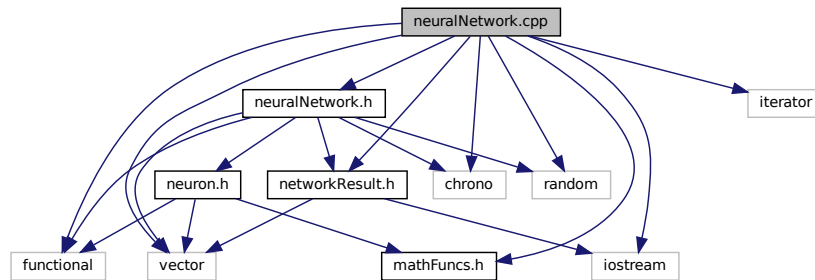
Copyright (c) 2022

6.8 neuralNetwork.cpp File Reference

```
#include <functional>
#include <vector>
#include <iterator>
#include <iostream>
#include <chrono>
#include <random>
#include "mathFuncs.h"
#include "networkResult.h"
```

```
#include "neuralNetwork.h"
```

Include dependency graph for neuralNetwork.cpp:



6.8.1 Detailed Description

Author

Olivier Halupczok

Version

0.1

Date

2022-06-13

Copyright

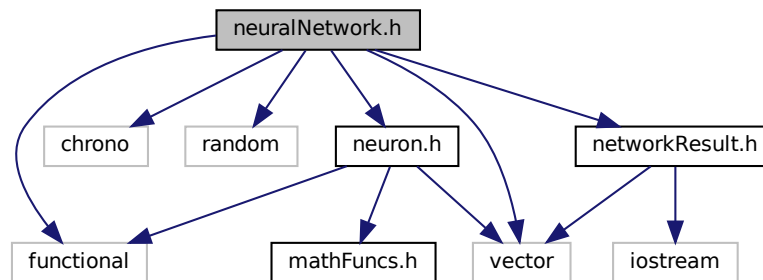
Copyright (c) 2022

6.9 neuralNetwork.h File Reference

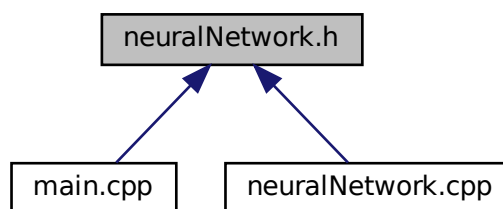
```
#include <vector>
#include <chrono>
#include <random>
#include <functional>
#include "neuron.h"
```

```
#include "networkResult.h"
```

Include dependency graph for neuralNetwork.h:



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



Classes

- class [NeuralNetwork](#)

6.9.1 Detailed Description

Author

Olivier Halupczok

Version

0.1

Date

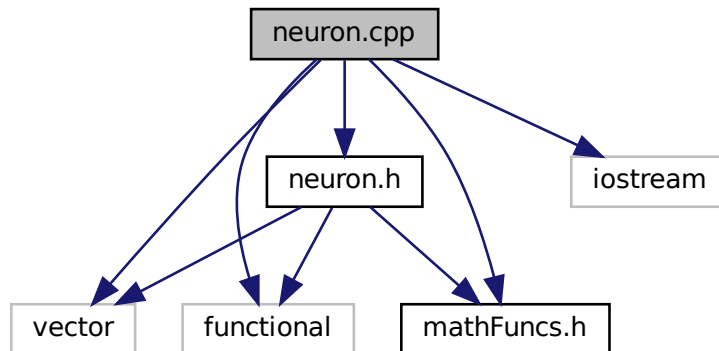
2022-06-13

Copyright

Copyright (c) 2022

6.10 neuron.cpp File Reference

```
#include <vector>
#include <functional>
#include <iostream>
#include "mathFuncs.h"
#include "neuron.h"
Include dependency graph for neuron.cpp:
```



Functions

- `std::string exceptionMsg` (`std::string` propertyName)

Variables

- `constexpr double LEARNING_DEFAULT_RATE` = 0.01
- `const std::string OUTPUT` = "output"
- `const std::string TOTAL` = "total"

6.10.1 Detailed Description

Author

Olivier Halupczok

Version

0.1

Date

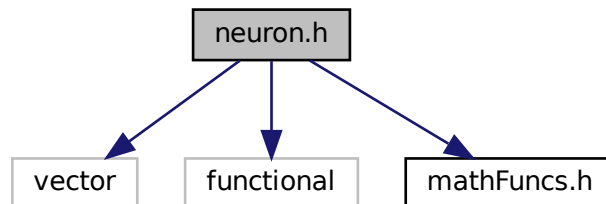
2022-06-12

Copyright

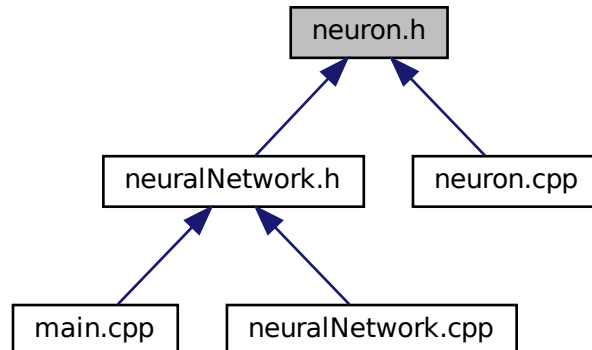
Copyright (c) 2022

6.11 neuron.h File Reference

```
#include <vector>
#include <functional>
#include "mathFuncs.h"
Include dependency graph for neuron.h:
```



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



Classes

- class [Neuron](#)

6.11.1 Detailed Description

Author

Olivier Halupczok

Version

0.1

Date

2022-06-12

Copyright

Copyright (c) 2022

Index

- adjustBias
 - Neuron, 16
- adjustWeight
 - Neuron, 17
- calc_mse_loss
 - mathFuncs.cpp, 23
 - mathFuncs.h, 27
- CSV_Logger, 9
 - CSV_Logger, 10
- deriv_sigmoid
 - mathFuncs.cpp, 24
 - mathFuncs.h, 27
- dotProductOf2Vectors
 - mathFuncs.cpp, 24
 - mathFuncs.h, 27
- feedforward
 - NeuralNetwork, 15
 - Neuron, 17
- getBias
 - Neuron, 17
- getLoss
 - NetworkResult, 13
- getOutput
 - Neuron, 18
- getPredictions
 - NetworkResult, 13
- getTotal
 - Neuron, 18
- getWeights
 - Neuron, 18
- Logger, 10
 - operator<<, 11
 - setOutputStream, 12
- logger.cpp, 19
- logger.h, 20
- lossDeriv_outDeriv
 - mathFuncs.cpp, 24
 - mathFuncs.h, 28
- main.cpp, 21
- mathFuncs.cpp, 22
 - calc_mse_loss, 23
 - deriv_sigmoid, 24
 - dotProductOf2Vectors, 24
 - lossDeriv_outDeriv, 24
 - sigmoid, 25
- mathFuncs.h, 25
 - calc_mse_loss, 27
 - deriv_sigmoid, 27
 - dotProductOf2Vectors, 27
 - lossDeriv_outDeriv, 28
 - sigmoid, 28
- NetworkResult, 12
 - getLoss, 13
 - getPredictions, 13
 - NetworkResult, 12
 - operator<<, 13
- networkResult.cpp, 29
 - operator<<, 30
- networkResult.h, 30
- NeuralNetwork, 14
 - feedforward, 15
 - NeuralNetwork, 14
 - train, 15
- neuralNetwork.cpp, 31
- neuralNetwork.h, 32
- Neuron, 15
 - adjustBias, 16
 - adjustWeight, 17
 - feedforward, 17
 - getBias, 17
 - getOutput, 18
 - getTotal, 18
 - getWeights, 18
 - Neuron, 16
- neuron.cpp, 34
- neuron.h, 35
- operator<<
 - Logger, 11
 - NetworkResult, 13
 - networkResult.cpp, 30
- setOutputStream
 - Logger, 12
- sigmoid
 - mathFuncs.cpp, 25
 - mathFuncs.h, 28
- train
 - NeuralNetwork, 15