ML perceptron (P1) v0.1

Generated by Doxygen 1.13.2

1 ML-perceptron	1
1.1 Student	1
1.2 Introduction	1
1.3 Documentation	1
1.4 Installing	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Perceptron Class Reference	7
4.1.1 Detailed Description	7
4.1.2 Constructor & Destructor Documentation	7
4.1.2.1 Perceptron()	7
4.1.3 Member Function Documentation	8
4.1.3.1str()	8
4.1.3.2 predict()	8
4.1.3.3 train()	8
4.2 PerceptronLayer Class Reference	9
4.2.1 Detailed Description	9
4.2.2 Constructor & Destructor Documentation	9
4.2.2.1 PerceptronLayer()	9
4.2.3 Member Function Documentation	10
4.2.3.1str()	10
4.2.3.2 feedForward()	11
4.3 PerceptronNetwork Class Reference	11
4.3.1 Detailed Description	11
4.3.2 Constructor & Destructor Documentation	11
4.3.2.1 PerceptronNetwork()	11
4.3.3 Member Function Documentation	12
4.3.3.1str()	12
4.3.3.2 feedForward()	12
5 File Documentation	13
5.1 /Users/stanislav/Github/MachineLearning/ML-Perceptron/src/header/perceptron.hpp File Reference .	13
5.1.1 Detailed Description	14
5.2 perceptron.hpp	15
5.3 /Users/stanislav/Github/MachineLearning/ML-Perceptron/src/header/perceptronLayer.hpp File Refer-	
ence	15
5.3.1 Detailed Description	16
5.4 perceptronLayer.hpp	17

5.14 test.cpp	 	. 29
5.13.4.1 inputs	 	. 29
5.13.4 Variable Documentation	 	. 29
<b>5.13.3.8 TEST_CASE()</b> [8/8]	 	. 28
<b>5.13.3.7 TEST_CASE()</b> [7/8]	 	. 28
5.13.3.6 TEST_CASE() [6/8]	 	. 28
<b>5.13.3.5 TEST_CASE()</b> [5/8]	 	. 28
5.13.3.4 TEST_CASE() [4/8]	 	. 28
<b>5.13.3.3 TEST_CASE()</b> [3/8]	 	. 27
<b>5.13.3.2 TEST_CASE()</b> [2/8]	 	. 27
5.13.3.1 TEST_CASE() [1/8]	 	. 27
5.13.3 Function Documentation	 	. 27
5.13.2.2 EPOCHS	 	. 27
5.13.2.1 CATCH_CONFIG_MAIN	 	. 27
5.13.2 Macro Definition Documentation	 	. 27
5.13.1 Detailed Description		
5.13 /Users/stanislav/Github/MachineLearning/ML-Perceptron/src/test/test.cpp File Refere		
5.12 perceptronNetwork.cpp		
5.11.1 Detailed Description		
5.11 /Users/stanislav/Github/MachineLearning/ML-Perceptron/src/perceptronNetwork.cpp		
5.10 perceptronLayer.cpp		
5.9.1 Detailed Description		
5.9 /Users/stanislav/Github/MachineLearning/ML-Perceptron/src/perceptronLayer.cpp File		
5.8 perceptron.cpp		
5.7.1 Detailed Description		
5.7 /Users/stanislav/Github/MachineLearning/ML-Perceptron/src/perceptron.cpp File Refer		
5.5.1 Detailed Description		
E E A Disable di Dispublication		

# **Chapter 1**

# **ML-perceptron**

# 1.1 Student

Name: Stan Merlijn

Student nummer: 1863967

# 1.2 Introduction

In this repo we are going to implement and test perceptrons, perceptron layers and a perceptron networks(neural network). Theset are going to be tested by creating AND, OR, INVERT, NAND, XOR and half adder logic gates. the reader can be found here

# 1.3 Documentation

For this assignment, documentation was generated using Doxygen. The LaTeX documentation can be found here and if you want to run the HTML local website, you can open the index.html.

# 1.4 Installing

Enter the test dir then

Generate build files:

cmake -S . -B build

Build the project:

cmake --build build

Run the executable:

./build/MLPerceptronTest

2 ML-perceptron

# **Chapter 2**

# **Class Index**

# 2.1 Class List

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

Perceptron	
A simple perceptron model for binary classification	7
PerceptronLayer	
Represents a layer of perceptrons in a neural network	ć
PerceptronNetwork	
Represents a multi-layer perceptron network	1

4 Class Index

# **Chapter 3**

# File Index

# 3.1 File List

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

/Users/stanislav/Github/MachineLearning/ML-Perceptron/src/perceptron.cpp	
Implementation of the Perceptron class	19
/Users/stanislav/Github/MachineLearning/ML-Perceptron/src/perceptronLayer.cpp	
Implementation of the PerceptronLayer class	21
/Users/stanislav/Github/MachineLearning/ML-Perceptron/src/perceptronNetwork.cpp	
Implementation of the PerceptronNetwork class	22
/Users/stanislav/Github/MachineLearning/ML-Perceptron/src/header/perceptron.hpp	
In this file the Perceptron class is defined	13
/Users/stanislav/Github/MachineLearning/ML-Perceptron/src/header/perceptronLayer.hpp	
In this file the PerceptronLayer class is defined	15
/Users/stanislav/Github/MachineLearning/ML-Perceptron/src/header/perceptronNetwork.hpp	
In this file the PerceptronNetwork class is defined	17
/Users/stanislav/Github/MachineLearning/ML-Perceptron/src/test/test.cpp	
In this file the test cases for the Perceptron, PerceptronLayer and PerceptronNetwork classes	
are defined	24

6 File Index

# **Chapter 4**

# **Class Documentation**

# 4.1 Perceptron Class Reference

A simple perceptron model for binary classification.

```
#include <perceptron.hpp>
```

#### **Public Member Functions**

- Perceptron (std::vector< double > weights, double bias, double learningRate)
  - Constructs a Perceptron with given weights, bias, and learning rate.
- int predict (const std::vector< int > &inputs) const

Predicts the output for a given input vector.

- void train (const std::vector< std::vector< int > & std::vector< int > & targets, int epochs)
  - Trains the perceptron using the given dataset. Using th learning rule to update the weights.
- · void str (int verbose) const

Prints perceptron details.

# 4.1.1 Detailed Description

A simple perceptron model for binary classification.

Definition at line 20 of file perceptron.hpp.

# 4.1.2 Constructor & Destructor Documentation

# 4.1.2.1 Perceptron()

```
Perceptron::Perceptron (
         std::vector< double > weights,
         double bias,
         double learningRate)
```

Constructs a Perceptron with given weights, bias, and learning rate.

8 Class Documentation

#### **Parameters**

weights	Initial weights.
bias	Initial bias.
learningRate	Learning rate for training.

Definition at line 13 of file perceptron.cpp.

# 4.1.3 Member Function Documentation

```
4.1.3.1 __str__()
```

Prints perceptron details.

#### **Parameters**

verbose	Verbosity level.
---------	------------------

Definition at line 49 of file perceptron.cpp.

# 4.1.3.2 predict()

Predicts the output for a given input vector.

#### **Parameters**

inputs	Input vector.

# Returns

1 if activated, otherwise 0.

Definition at line 16 of file perceptron.cpp.

# 4.1.3.3 train()

Trains the perceptron using the given dataset. Using th learning rule to update the weights.

#### **Parameters**

inputs	Input samples.
targets	Target outputs.
epochs	Number of training iterations.

Definition at line 27 of file perceptron.cpp.

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

- /Users/stanislav/Github/MachineLearning/ML-Perceptron/src/header/perceptron.hpp
- /Users/stanislav/Github/MachineLearning/ML-Perceptron/src/perceptron.cpp

# 4.2 PerceptronLayer Class Reference

Represents a layer of perceptrons in a neural network.

```
#include <perceptronLayer.hpp>
```

#### **Public Member Functions**

PerceptronLayer (const std::vector< Perceptron > &neurons)

Constructs a perceptron layer.

std::vector< int > feedForward (const std::vector< int > &input) const

Feeds input forward through the layer.

void <u>str</u> (int verbose) const

Prints layer details.

# 4.2.1 Detailed Description

Represents a layer of perceptrons in a neural network.

Definition at line 20 of file perceptronLayer.hpp.

## 4.2.2 Constructor & Destructor Documentation

# 4.2.2.1 PerceptronLayer()

Constructs a perceptron layer.

#### **Parameters**

neurons	List of perceptrons.
---------	----------------------

Definition at line 13 of file perceptronLayer.cpp.

10 Class Documentation

# 4.2.3 Member Function Documentation

```
4.2.3.1 __str__()
```

Prints layer details.

#### **Parameters**

verbose Verbosity level.
--------------------------

Definition at line 27 of file perceptronLayer.cpp.

#### 4.2.3.2 feedForward()

Feeds input forward through the layer.

#### **Parameters**

```
input Input vector.
```

#### Returns

Output vector after applying all perceptrons.

Definition at line 16 of file perceptronLayer.cpp.

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

- /Users/stanislav/Github/MachineLearning/ML-Perceptron/src/header/perceptronLayer.hpp
- /Users/stanislav/Github/MachineLearning/ML-Perceptron/src/perceptronLayer.cpp

# 4.3 PerceptronNetwork Class Reference

Represents a multi-layer perceptron network.

```
#include <perceptronNetwork.hpp>
```

#### **Public Member Functions**

- PerceptronNetwork (std::vector< PerceptronLayer > layers)
  - Constructs a perceptron network.
- std::vector< int > feedForward (const std::vector< int > &input) const

Feeds input forward through the network.

void <u>str</u> (int verbose) const

Prints network details.

# 4.3.1 Detailed Description

Represents a multi-layer perceptron network.

Definition at line 20 of file perceptronNetwork.hpp.

## 4.3.2 Constructor & Destructor Documentation

# 4.3.2.1 PerceptronNetwork()

Constructs a perceptron network.

12 Class Documentation

#### **Parameters**

layers	List of perceptron layers.
idyoro	List of perception layers.

Definition at line 15 of file perceptronNetwork.cpp.

# 4.3.3 Member Function Documentation

```
4.3.3.1 __str__()
```

Prints network details.

#### **Parameters**

verbose	Verbosity level.
---------	------------------

Definition at line 28 of file perceptronNetwork.cpp.

#### 4.3.3.2 feedForward()

Feeds input forward through the network.

#### **Parameters**

input	Input vector.
-------	---------------

#### Returns

Output vector after processing through all layers.

Definition at line 18 of file perceptronNetwork.cpp.

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

- /Users/stanislav/Github/MachineLearning/ML-Perceptron/src/header/perceptronNetwork.hpp
- /Users/stanislav/Github/MachineLearning/ML-Perceptron/src/perceptronNetwork.cpp

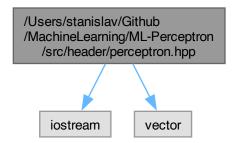
# **Chapter 5**

# **File Documentation**

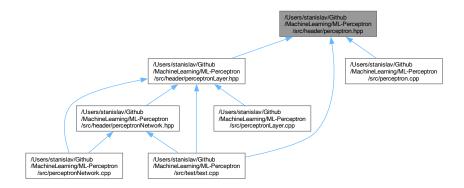
# 5.1 /Users/stanislav/Github/MachineLearning/ML-← Perceptron/src/header/perceptron.hpp File Reference

In this file the Perceptron class is defined.

#include <iostream>
#include <vector>
Include dependency graph for perceptron.hpp:



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



## Classes

• class Perceptron

A simple perceptron model for binary classification.

# 5.1.1 Detailed Description

In this file the Perceptron class is defined.

**Author** 

Stan Merlijn

Version

0.1

Date

2025-02-12

Copyright

Copyright (c) 2025

Definition in file perceptron.hpp.

5.2 perceptron.hpp 15

# 5.2 perceptron.hpp

Go to the documentation of this file.

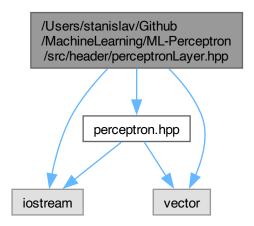
```
00011
00012 #pragma once
00013 #include <iostream>
00014 #include <vector>
00015
00020 class Perceptron
00021 {
00022 public:
00029
          Perceptron(std::vector<double> weights, double bias, double learningRate);
00030
00036
          int predict(const std::vector<int>& inputs) const;
         void train(const std::vector<std::vector<int>% inputs, const std::vector<int>% targets, int
void epochs);
00050
          void __str__(int verbose) const;
00051
00052 private:
00053 std::vector<double> weights;
00054 double bias;
00055
              double learningRate;
00056 };
```

# 5.3 /Users/stanislav/Github/MachineLearning/ML Perceptron/src/header/perceptronLayer.hpp File Reference

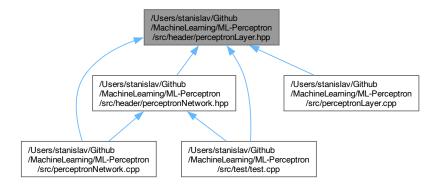
In this file the PerceptronLayer class is defined.

```
#include "perceptron.hpp"
#include <iostream>
#include <vector>
```

Include dependency graph for perceptronLayer.hpp:



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



#### Classes

· class PerceptronLayer

Represents a layer of perceptrons in a neural network.

# 5.3.1 Detailed Description

In this file the PerceptronLayer class is defined.

Author

Stan Merlijn

Version

0.1

Date

2025-02-12

Copyright

Copyright (c) 2025

Definition in file perceptronLayer.hpp.

# 5.4 perceptronLayer.hpp

# Go to the documentation of this file.

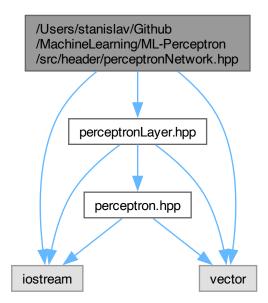
```
00011 #pragma once
00012 #include "perceptron.hpp"
00013 #include <iostream>
00014 #include <vector>
00015
00020 class PerceptronLayer
00021 {
00022 public:
00027
          PerceptronLayer(const std::vector<Perceptron>& neurons);
00028
00034
          std::vector<int> feedForward(const std::vector<int>& input) const;
00040
          void __str__(int verbose) const;
00041
00042 private:
00043
          std::vector<Perceptron> neurons;
00044 };
```

# 5.5 /Users/stanislav/Github/MachineLearning/ML-← Perceptron/src/header/perceptronNetwork.hpp File Reference

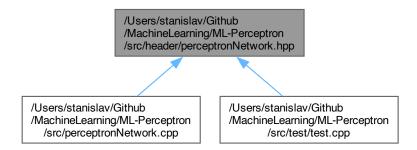
In this file the PerceptronNetwork class is defined.

```
#include "perceptronLayer.hpp"
#include <iostream>
#include <vector>
```

Include dependency graph for perceptronNetwork.hpp:



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



## Classes

• class PerceptronNetwork

Represents a multi-layer perceptron network.

# 5.5.1 Detailed Description

In this file the PerceptronNetwork class is defined.

Author

Stan Merlijn

Version

0.1

Date

2025-02-12

Copyright

Copyright (c) 2025

Definition in file perceptronNetwork.hpp.

# 5.6 perceptronNetwork.hpp

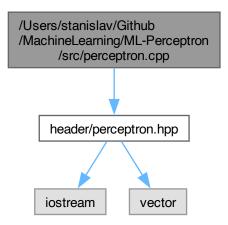
## Go to the documentation of this file.

```
00011 #pragma once
00012 #include "perceptronLayer.hpp"
00013 #include <iostream>
00014 #include <vector>
00015
00020 class PerceptronNetwork
00021 {
00022 public:
00027
          PerceptronNetwork(std::vector<PerceptronLayer> layers);
00028
00034
          std::vector<int> feedForward(const std::vector<int>& input) const;
00040
          void __str__(int verbose) const;
00041
00042 private:
00043
          std::vector<PerceptronLayer> layers;
00044
00045 };
```

# 5.7 /Users/stanislav/Github/MachineLearning/ML Perceptron/src/perceptron.cpp File Reference

Implementation of the Perceptron class.

#include "header/perceptron.hpp"
Include dependency graph for perceptron.cpp:



## 5.7.1 Detailed Description

Implementation of the Perceptron class.

Author

Stan Merlijn

Version

0.1

Date

2025-02-12

Copyright

Copyright (c) 2025

Definition in file perceptron.cpp.

# 5.8 perceptron.cpp

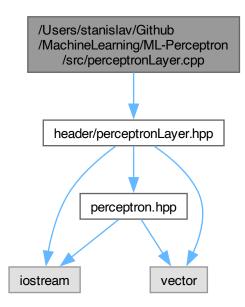
Go to the documentation of this file.

```
00001
00011 #include "header/perceptron.hpp"
00013 Perceptron::Perceptron(std::vector<double> weights, double bias, double learningRate)
00014
            : weights(weights), bias(bias), learningRate(learningRate) {}
00015
00016 int Perceptron::predict(const std::vector<int>& inputs) const
00017 {
00018
            // Dot prodcut for the perceptron
            double dot_product = bias;
for (int i = 0; i < weights.size(); i++) {
    dot_product += weights[i] * inputs[i];
00019
00020
00021
00022
00023
            // Threshold function
00024
            return dot_product >= 0 ? 1 : 0;
00025 }
00026
00027 void Perceptron::train(const std::vector<std::vector<int>& inputs, const std::vector<int>& targets,
       int epochs)
00028 {
00029
            // ensure both arrays are the same size
00030
            if (inputs.size() != targets.size()) return;
00031
00032
            // Train the perceptron
            for (int epoch = 0; epoch < epochs; epoch++) {
    for (int i = 0; i < inputs.size(); i++) {</pre>
00033
00034
00035
                     // get the prediction
00036
                      double pred = predict(inputs[i]);
00037
                      // Calculate the error based of the target value
00038
                      double error = targets[i] - pred;
                      // Update each weight based on the input value
for (int j = 0; j < weights.size(); j++) {
    weights[j] += learningRate * error * inputs[i][j];</pre>
00039
00040
00041
00042
00043
                       // Update bias
00044
                      bias += learningRate * error;
00045
                 }
            }
00046
00047 }
00048
00049 void Perceptron::__str__(int verbose) const
00050 {
            \ensuremath{//} Printing the weights
00051
            std::cout « "weights for perceptron:\n";
for (auto i : weights)
00052
00053
00054
                 std::cout « i « "
00055
00056
            // Other info
            if (verbose >= 1) {
    std::cout « "\nbias = " « bias « "\n";
    std::cout « "Learning rate = " « learningRate « std::endl;
00057
00058
00059
00060
            }
00061 }
```

# 5.9 /Users/stanislav/Github/MachineLearning/ML-← Perceptron/src/perceptronLayer.cpp File Reference

Implementation of the PerceptronLayer class.

#include "header/perceptronLayer.hpp"
Include dependency graph for perceptronLayer.cpp:



# 5.9.1 Detailed Description

Implementation of the PerceptronLayer class.

Author

Stan Merlijn

Version

0.1

Date

2025-02-12

Copyright

Copyright (c) 2025

Definition in file perceptronLayer.cpp.

# 5.10 perceptronLayer.cpp

#### Go to the documentation of this file.

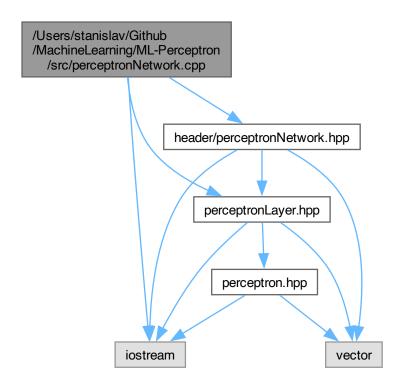
```
00011 #include "header/perceptronLayer.hpp"
00012
00013 PerceptronLayer::PerceptronLayer(const std::vector<Perceptron>& neurons)
00014
          : neurons (neurons) {}
00015
00016 std::vector<int> PerceptronLayer::feedForward(const std::vector<int>& input) const
00017 {
00018
          \ensuremath{//} Predict the output for each perceptron
          std::vector<int> outputs;
// Propagate the input through each layer sequentially. Also called feedforward.
00019
00020
00021
          for (const Perceptron& neuron : neurons) {
              outputs.push_back(neuron.predict(input));
00023
00024
          return outputs;
00025 }
00026
00027 void PerceptronLayer::__str__(int verbose) const
00028 {
           // For each neuron in the layer print the data
00030
          for (const Perceptron& neuron : neurons) {
00031
            neuron.__str__(verbose);
00032
00033 }
```

# 5.11 /Users/stanislav/Github/MachineLearning/MLPerceptron/src/perceptronNetwork.cpp File Reference

Implementation of the PerceptronNetwork class.

```
#include "header/perceptronNetwork.hpp"
#include "header/perceptronLayer.hpp"
#include <iostream>
```

Include dependency graph for perceptronNetwork.cpp:



# 5.11.1 Detailed Description

Implementation of the PerceptronNetwork class.

Author

Stan Merlijn

Version

0.1

Date

2025-02-12

Copyright

Copyright (c) 2025

Definition in file perceptronNetwork.cpp.

# 5.12 perceptronNetwork.cpp

#### Go to the documentation of this file.

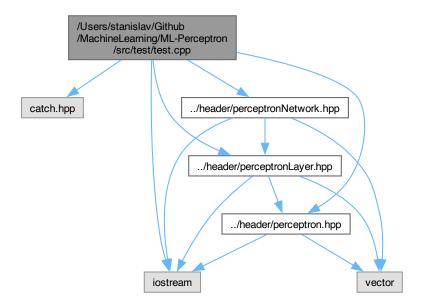
```
00011 #include "header/perceptronNetwork.hpp"
00012 #include "header/perceptronLayer.hpp"
00013 #include <iostream>
00014
00015 PerceptronNetwork::PerceptronNetwork(std::vector<PerceptronLayer> layers)
00016
          : layers(layers) {}
00017
00018 std::vector<int> PerceptronNetwork::feedForward(const std::vector<int>& input) const
00019 {
           std::vector<int> activation = input;
00020
00021
           // Propagate the input through each layer sequentially. Also called feedforward.
           for (const PerceptronLayer& layer: layers) {
00023
               activation = layer.feedForward(activation);
00024
           return activation;
00025
00026 }
00027
00028 void PerceptronNetwork::__str__(int verbose) const
00030
           // Print the network structure
          std::cout « "Perceptron Network Structure:" « std::endl;
std::cout « "Number of layers: " « layers.size() « std::endl;
00031
00032
          // For each layer in the network print the data for (int i = 0; i < layers.size(); ++i)
00033
00034
00035
00036
               std::cout « "Layer " « i + 1 « ": ";
00037
               layers[i].__str__(verbose);
           }
00038
00039 }
```

# 5.13 /Users/stanislav/Github/MachineLearning/ML-Perceptron/src/test/test.cpp File Reference

In this file the test cases for the Perceptron, PerceptronLayer and PerceptronNetwork classes are defined.

```
#include "catch.hpp"
#include "../header/perceptron.hpp"
#include "../header/perceptronLayer.hpp"
#include "../header/perceptronNetwork.hpp"
#include <iostream>
```

Include dependency graph for test.cpp:



#### **Macros**

- #define CATCH CONFIG MAIN
- #define EPOCHS 100

# **Functions**

TEST\_CASE ("Perceptron for INVERT Gate", "[perceptron]")

Perceptron for INVERT Gate: Tests the perceptron's ability to learn the INVERT gate.

• TEST\_CASE ("Perceptron for AND Gate", "[perceptron]")

Perceptron for AND Gate: Tests the perceptron's ability to learn the AND gate.

• TEST CASE ("Perceptron for OR Gate", "[perceptron]")

Perceptron for OR Gate: Tests the perceptron's ability to learn the OR gate.

• TEST\_CASE ("Perceptron for NOR Gate (3 inputs)", "[perceptron]")

Perceptron for NOR Gate (3 inputs): Tests the perceptron's ability to learn the NOR gate with 3 inputs. The NOR gate is a digital logic gate that implements logical NOR

TEST\_CASE ("Perceptron for 3-input Majority Gate", "[perceptron]")

Perceptron for 3-input Majority Gate: Tests the perceptron's ability to learn the 3-input Majority gate.

TEST CASE ("PerceptronLayer for AND and OR Gates", "[perceptronLayer]")

PerceptronLayer for AND and OR Gates: Tests the PerceptronLayer's ability to learn the AND and OR gates. It contains two perceptrons: one for the AND gate and one for the OR gate.

TEST\_CASE ("PerceptronNetwork for the XOR gate with 2 inputs", "[perceptronNetwork]")

PerceptronNetwork for the XOR gate with 2 inputs. This network contains two layers: inputLayer for the AND gate and one for the OR gate. outputLayer for the AND gate.

• TEST\_CASE ("PerceptronNetwork for half adder", "[perceptronNetwork]")

PerceptronNetwork for a half adder. This network contains two layers: hiddenLayer for the OR and AND gates. outputLayer for the XOR gate(sum) and the carry.

#### **Variables**

• std::vector< std::vector< int >> inputs = {{0, 0}, {0, 1}, {1, 0}, {1, 1}}

# 5.13.1 Detailed Description

In this file the test cases for the Perceptron, PerceptronLayer and PerceptronNetwork classes are defined.

Unit tests for the Perceptron, PerceptronLayer and PerceptronNetwork classes.

**Author** 

Stan Merlijn

Version

0.1

Date

2025-02-12

# Copyright

Copyright (c) 2025

This file contains a series of test cases to verify the functionality of the Perceptron and PerceptronLayer classes. The tests include training and prediction for various logic gates.

## Test Cases:

- Perceptron for INVERT Gate: Tests the perceptron's ability to learn the INVERT gate.
- Perceptron for AND Gate: Tests the perceptron's ability to learn the AND gate.
- Perceptron for OR Gate: Tests the perceptron's ability to learn the OR gate.
- Perceptron for NOR Gate (3 inputs): Tests the perceptron's ability to learn the NOR gate with 3 inputs.
- Perceptron for 3-input Majority Gate: Tests the perceptron's ability to learn the 3-input Majority gate.
- PerceptronLayer for AND and OR Gates: Tests the PerceptronLayer's ability to learn the AND and OR gates.
- PerceptronNetwork for the XOR gate with 2 inputs.
- PerceptronNetwork for a half adder.

Note

The tests use the Catch2 framework for unit testing.

Definition in file test.cpp.

## 5.13.2 Macro Definition Documentation

#### 5.13.2.1 CATCH CONFIG MAIN

```
#define CATCH_CONFIG_MAIN
```

Definition at line 11 of file test.cpp.

#### 5.13.2.2 EPOCHS

```
#define EPOCHS 100
```

Definition at line 12 of file test.cpp.

## 5.13.3 Function Documentation

#### 5.13.3.1 TEST\_CASE() [1/8]

Perceptron for 3-input Majority Gate: Tests the perceptron's ability to learn the 3-input Majority gate.

Definition at line 124 of file test.cpp.

# 5.13.3.2 TEST\_CASE() [2/8]

Perceptron for AND Gate: Tests the perceptron's ability to learn the AND gate.

Definition at line 66 of file test.cpp.

## 5.13.3.3 TEST\_CASE() [3/8]

Perceptron for INVERT Gate: Tests the perceptron's ability to learn the INVERT gate.

Definition at line 49 of file test.cpp.

## 5.13.3.4 TEST\_CASE() [4/8]

Perceptron for NOR Gate (3 inputs): Tests the perceptron's ability to learn the NOR gate with 3 inputs. The NOR gate is a digital logic gate that implements logical NOR

```
0, 0, 0, 0, 0, 0, 0, 1
```

Definition at line 98 of file test.cpp.

# 5.13.3.5 TEST\_CASE() [5/8]

Perceptron for OR Gate: Tests the perceptron's ability to learn the OR gate.

Definition at line 81 of file test.cpp.

## 5.13.3.6 TEST\_CASE() [6/8]

PerceptronLayer for AND and OR Gates: Tests the PerceptronLayer's ability to learn the AND and OR gates. It contains two perceptrons: one for the AND gate and one for the OR gate.

Definition at line 152 of file test.cpp.

# 5.13.3.7 TEST\_CASE() [7/8]

PerceptronNetwork for a half adder. This network contains two layers: hiddenLayer for the OR and AND gates. outputLayer for the XOR gate(sum) and the carry.

Definition at line 216 of file test.cpp.

## 5.13.3.8 TEST\_CASE() [8/8]

PerceptronNetwork for the XOR gate with 2 inputs. This network contains two layers: inputLayer for the AND gate and one for the OR gate. outputLayer for the AND gate.

Definition at line 181 of file test.cpp.

5.14 test.cpp 29

#### 5.13.4 Variable Documentation

#### 5.13.4.1 inputs

```
std::vector < std::vector < int > inputs = {{0, 0}, {0, 1}, {1, 0}, {1, 1}}
```

Definition at line 42 of file test.cpp.

# 5.14 test.cpp

#### Go to the documentation of this file.

```
00011 #define CATCH_CONFIG_MAIN
00012 #define EPOCHS 100
00013
00014 #include "catch.hpp"
00014 #Include catch.hpp"
00015 #include "../header/perceptronLayer.hpp"
00016 #include "../header/perceptronLayer.hpp"
00017 #include "../header/perceptronNetwork.hpp"
00018
00019 #include <iostream>
00020
00040
00041 // Define the input vectors for the logic gates
00042 std::vector<std::vector<int> inputs = \{\{0, 0\}, \{0, 1\}, \{1, 0\}, \{1, 1\}\}\};
00044
00049 TEST_CASE("Perceptron for INVERT Gate", "[perceptron]")
00050 {
           Perceptron invert_gate({0.1, 0.1}, 1, 0.1);
00051
00052
00053
           // Training data: for input 0 we expect output 1, and for input 1 we expect output 0.
00054
           // The second element in the input vector is always 0.
00055
           std::vector<std::vector<int> inputsInverter = {{0, 0}, {1, 0}};
00056
           std::vector<int> targets = \{1, 0\};
           invert_gate.train(inputsInverter, targets, EPOCHS);
00057
00058
00059
           REQUIRE(invert_gate.predict({1, 0}) == 0);
           REQUIRE(invert_gate.predict({0, 1}) == 1);
00060
00061 }
00062
00066 TEST_CASE("Perceptron for AND Gate", "[perceptron]")
00067 {
00068
           Perceptron p_and({0.1, 0.1}, 1, 0.1);
00069
           std::vector<int> targets = {0,0,0,1};
00070
           p_and.train(inputs, targets, EPOCHS);
00071
00072
           REQUIRE(p_and.predict({0, 0}) == 0);
00073
           REQUIRE(p_and.predict({0, 1}) == 0);
REQUIRE(p_and.predict({1, 0}) == 0);
00074
00075
           REQUIRE(p_and.predict({1, 1}) == 1);
00076 }
00077
00081 TEST_CASE("Perceptron for OR Gate", "[perceptron]")
00082 {
00083
           Perceptron p_or({0.1, 0.1}, 1, 0.1);
           std::vector<int> targets = {0,1,1,1};
00084
00085
           p_or.train(inputs, targets, EPOCHS);
00086
           REQUIRE(p_or.predict({0, 0}) == 0);
REQUIRE(p_or.predict({0, 1}) == 1);
00087
00088
           REQUIRE (p_or.predict({1, 0}) == 1);
00089
00090
           REQUIRE(p_or.predict({1, 1}) == 1);
00091 }
00092
00098 TEST_CASE("Perceptron for NOR Gate (3 inputs)", "[perceptron]") {
00099
           \ensuremath{//} Instantiate the perceptron with three weights.
00100
           Perceptron norGate({-0.1, -0.1, -0.1}, 1, 0.1);
00101
00102
           // Training data for a NOR gate with 3 inputs:
00103
           // Only (0,0,0) should yield 1; all others yield 0.
00104
           std::vector<std::vector<int> inputsNOR = {
               {0, 0, 0}, {0, 0, 1}, {0, 1, 0}, {1, 0, 0}, {1, 1, 0}, {1, 1, 1, 1}
00105
00106
00107
00108
           std::vector<int> targets = {1, 0, 0, 0, 0, 0, 0, 0};
```

```
norGate.train(inputsNOR, targets, EPOCHS);
00110
00111
           REQUIRE (norGate.predict(\{0, 0, 0\}) == 1);
00112
           REQUIRE(norGate.predict(\{0, 0, 1\}) == 0);
           REQUIRE(norGate.predict({0, 1, 0}) == 0);
00113
00114
           REQUIRE (norGate.predict ({1, 0, 0}) == 0);
00115
           REQUIRE (norGate.predict (\{0, 1, 1\}) == 0);
00116
           REQUIRE (norGate.predict(\{1, 0, 1\}) == 0);
00117
           REQUIRE (norGate.predict(\{1, 1, 0\}) == 0);
00118
           REQUIRE (norGate.predict(\{1, 1, 1\}) == 0);
00119 }
00120
00124 TEST_CASE("Perceptron for 3-input Majority Gate", "[perceptron]") {
00125
           // Instantiate the perceptron with three inputs. Here we choose small positive initial weights
00126
           // and a negative bias. Adjust these parameters if necessary to speed up convergence.
00127
           Perceptron majorityGate({0.1, 0.1, 0.1}, -0.2, 0.1);
00128
00129
           // Training data for a majority gate:
           // Output 1 if at least two inputs are 1, else output 0.
00131
           std::vector<std::vector<int> inputsMajority = {
               {0, 0, 0}, {0, 0, 1}, {0, 1, 0}, {1, 0, 0}, {0, 1, 1}, {1, 0, 1}, {1, 1, 1}
00132
00133
00134
          std::vector<int> y = {0, 0, 0, 0, 1, 1, 1, 1};
majorityGate.train(inputsMajority, y, EPOCHS);
00135
00136
00137
00138
           REQUIRE(majorityGate.predict({0, 0, 0}) == 0);
00139
           REQUIRE(majorityGate.predict({0, 0, 1}) == 0);
00140
           REQUIRE(majorityGate.predict({0, 1, 0}) == 0);
00141
           REQUIRE(majorityGate.predict({1, 0, 0}) == 0);
00142
           REQUIRE(majorityGate.predict({0, 1, 1}) == 1);
00143
           REQUIRE(majorityGate.predict({1, 0, 1}) == 1);
00144
           REQUIRE(majorityGate.predict({1, 1, 0}) == 1);
00145
           REQUIRE(majorityGate.predict({1, 1, 1}) == 1);
00146 }
00147
00152 TEST_CASE("PerceptronLayer for AND and OR Gates", "[perceptronLayer]") {
00153
           // Training data common to both gates:
00154
           Perceptron p_or({0.1, 0.1}, 1, 0.1);
00155
           Perceptron p_and({0.1, 0.1}, 1, 0.1);
00156
           // Training the OR and AND gates.
00157
          p_or.train(inputs, {0, 1, 1, 1}, EPOCHS);
p_and.train(inputs, {0, 0, 0, 1}, EPOCHS);
00158
00159
00160
00161
           // Create a layer with two neurons (2 inputs) for the AND gate and a learning rate of 0.1.
00162
           // Train the layer with the AND gate targets and OR gate targets.
00163
           PerceptronLayer andLayer({p_and, p_or});
00164
00165
           // Define expected outputs for the AND gate and OR gate.
          std::vector<int> out00 = {0, 0};
std::vector<int> out01 = {0, 1};
00166
00167
00168
           std::vector < int > out11 = {1, 1};
00169
00170
           REQUIRE(andLayer.feedForward({0, 0}) == out00);
00171
           REQUIRE(andLayer.feedForward({0, 1}) == out01);
00172
           REQUIRE(andLayer.feedForward({1, 0}) == out01);
00173
           REQUIRE(andLayer.feedForward({1, 1}) == out11);
00174 }
00175
00181 TEST_CASE("PerceptronNetwork for the XOR gate with 2 inputs", "[perceptronNetwork]") {
00182
          // Create a network with two layers: one for the AND gate and one for the OR gate.
           // OR and NAND gates for the input layer
00183
           Perceptron p_or({0.1, 0.1}, 1, 0.1);
00184
00185
           Perceptron p_nand({0.1, 0.1}, 1, 0.1);
00186
           Perceptron p_and({0.1, 0.1}, 1, 0.1);
00187
00188
           // Training The gates
          p_or.train(inputs, {0, 1, 1, 1}, EPOCHS);
p_nand.train(inputs, {1, 1, 1, 0}, EPOCHS);
p_and.train(inputs, {0, 0, 0, 1}, EPOCHS);
00189
00190
00191
00192
00193
           PerceptronLayer inputLayer({p_or, p_nand});
00194
           PerceptronLayer outputLayer({p_and});
00195
00196
           PerceptronNetwork xor_network({inputLayer, outputLayer});
00197
00198
           // Define expected outputs for the XOR gate.
           std::vector<int> out00 = {0};
std::vector<int> out01 = {1};
00199
00200
           std::vector<int> out10 = {1};
00201
           std::vector<int> out11 = {0};
00202
00203
00204
           // Verify network's predictions for the XOR gate.
00205
           REQUIRE(xor_network.feedForward({0, 0}) == out00);
00206
           REQUIRE (xor_network.feedForward({0, 1}) == out01);
REQUIRE (xor_network.feedForward({1, 0}) == out10);
00207
```

5.14 test.cpp 31

```
00208
              REQUIRE(xor_network.feedForward({1, 1}) == out11);
00209 }
00210
00216 TEST_CASE("PerceptronNetwork for half adder", "[perceptronNetwork]") 00217 {
              // Hidden layer: compute OR and AND
Perceptron n_or({0.1, 0.1}, 0.1, 0.1);
Perceptron n_and({0.1, 0.1}, 0.1, 0.1);
00218
00219
00220
00221
              n_or.train(inputs, {0, 1, 1, 1}, EPOCHS);
n_and.train(inputs, {0, 0, 0, 1}, EPOCHS);
00222
00223
00224
00225
              PerceptronLayer hiddenLayer({n_or, n_and});
00226
00227
              // Output layer: compute XOR (for sum) and carry \,
              // XOR = OR - AND; Perceptron with weights \{1, -1\} and bias -0.5 // x1 = 0, x2 = 0: 1 * 0 + -1 * 0 - 0.5 = -0.5 // x1 = 1, x2 = 0: 1 * 1 + -1 * 0 - 0.5 = 0.5 // x1 = 1, x2 = 1: 1 * 1 + -1 * 1 - 0.5 = -0.5
00228
00229
00230
00232
              Perceptron n_xor({0.1, 0.1}, 0.1, 0.1);
Perceptron n_carry({0.1, 0.1}, 0.1, 0.1);
00233
00234
00235
00236
              n_xor.train({{0, 0}, {1, 0}, {1, 1}}, {0, 1, 0}, EPOCHS);
n_carry.train(inputs, {0, 0, 0, 1}, EPOCHS);
00237
00238
00239
              PerceptronLayer outputLayer({n_xor, n_carry});
00240
00241
              PerceptronNetwork halfAdder({hiddenLayer, outputLayer});
00242
               // Test cases for half adder: {Sum, Carry}
00243
              REQUIRE (halfAdder.feedForward({0, 0}) == std::vector<int>{0, 0});
REQUIRE (halfAdder.feedForward({0, 1}) == std::vector<int>{1, 0});
00244
00245
00246
              REQUIRE(halfAdder.feedForward(\{1, 0\}) == std::vector<int>\{1, 0\});
              REQUIRE(halfAdder.feedForward({1, 1}) == std::vector<int>{0, 1});
00247
00248 }
```

# Index

```
/Users/stanislav/Github/MachineLearning/ML-Perceptron/src/he@dati@der@@ptwbfich.pdplAIN, 27
                                                              EPOCHS, 27
/Users/stanislav/Github/MachineLearning/ML-Perceptron/src/hearthander/sperceptronLayer.hpp,
          15, 17
                                                              TEST_CASE, 27, 28
/Users/stanislav/Github/MachineLearning/ML-Perceptron/sīd#$@ad@Af$@ceptronNetwork.hpp,
          17, 19
                                                              test.cpp, 27, 28
/Users/stanislav/Github/MachineLearning/ML-Perceptron/straiperceptron.cpp,
          19, 20
                                                              Perceptron, 8
/Users/stanislav/Github/MachineLearning/ML-Perceptron/src/perceptronLayer.cpp,
         21, 22
/Users/stanislav/Github/MachineLearning/ML-Perceptron/src/perceptronNetwork.cpp,
          22, 24
/Users/stanislav/Github/MachineLearning/ML-Perceptron/src/test/test.cpp,
         24, 29
 str
     Perceptron, 8
     PerceptronLayer, 10
     PerceptronNetwork, 12
CATCH_CONFIG_MAIN
     test.cpp, 27
EPOCHS
    test.cpp, 27
feedForward
     PerceptronLayer, 11
     PerceptronNetwork, 12
inputs
    test.cpp, 29
ML-perceptron, 1
Perceptron, 7
     __str__, <mark>8</mark>
    Perceptron, 7
    predict, 8
    train, 8
PerceptronLayer, 9
       _str__, 10
     feedForward, 11
     PerceptronLayer, 9
PerceptronNetwork, 11
      _str__, 12
    feedForward, 12
     PerceptronNetwork, 11
predict
     Perceptron, 8
```

test.cpp