ML perceptron sigmoid neuron (P3)

v0.1

Generated by Doxygen 1.13.2

Chapter 1

ML-sigmoid-neuron

1.1 Student

Name: Stan Merlijn

Student nummer: 1863967

1.2 Introduction

In this repository, we will implement and test a Neuron using the sigmoid function. This will be demonstrated by creating AND, OR, NOT, NOR gates aswell as an half adder. You can find the assignment here.

1.3 Documentation

For this assignment, the documentation was generated with Doxygen. The LaTeX documentation is available here and, to view the HTML documentation locally, open index.html in a browser.

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
```

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 Neuron Class Reference

Represents a single neuron in a neural network.

`#include <neuron.hpp>`

Public Member Functions

- Neuron (const std::vector< double > &weights, double bias)
  Constructs a Neuron with the given weights and bias.
- double sigmoid (double x)
  Computes the sigmoid activation function.
- double predict (const std::vector< int > &inputs)
  Performs a feedforward operation.
- void __str__ () const
  Prints the neuron details.

4.1.1 Detailed Description

Represents a single neuron in a neural network.

This class models a neuron with a set of weights and a bias. It provides methods to compute the sigmoid activation function and to perform a feedforward operation given a set of inputs.

Definition at line 24 of file neuron.hpp.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Neuron()

```
Neuron::Neuron (
            const std::vector< double > & weights,
            double bias)
```

Constructs a Neuron with the given weights and bias.

Parameters

weights	A vector of weights for the neuron.
bias	The bias term for the neuron.

Definition at line 13 of file neuron.cpp.

4.1.3 Member Function Documentation

4.1.3.1 __str__()

```
void Neuron::__str__ () const
```

Prints the neuron details.

Definition at line 38 of file neuron.cpp.

4.1.3.2 predict()

```
double Neuron::predict (
            const std::vector< int > & inputs)
```

Performs a feedforward operation.

Parameters

inputs	A vector of input values.

Returns

The output of the neuron after applying the weights, bias, and activation function.

Definition at line 22 of file neuron.cpp.

4.1.3.3 sigmoid()

```
double Neuron::sigmoid (
            double x)
```

Computes the sigmoid activation function.

Parameters

x	The input value.

Returns

The result of the sigmoid function applied to x.

Definition at line 16 of file neuron.cpp.

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

- /Users/stanislav/Github/MachineLearning/ML-sigmoid-neuron/src/header/neuron.hpp
- /Users/stanislav/Github/MachineLearning/ML-sigmoid-neuron/src/neuron.cpp

4.2 NeuronLayer Class Reference

Represents a layer of neurons in a neural network.

`#include <neuronLayer.hpp>`

Public Member Functions

- NeuronLayer (std::vector< Neuron > neurons)
  Constructs a NeuronLayer with the given neurons.
- std::vector< int > feedForward (const std::vector< int > &inputs)
  Performs a feedforward operation.
- void __str__ () const
  Prints the layer details.

4.2.1 Detailed Description

Represents a layer of neurons in a neural network.

The NeuronLayer class has a collection of neurons and provides methods to perform feedforward operations and to represent the layer as a string.

Definition at line 24 of file neuronLayer.hpp.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 NeuronLayer()

```
NeuronLayer::NeuronLayer (
            std::vector< Neuron > neurons)
```

Constructs a NeuronLayer with the given neurons.

Parameters

neurons	A vector of neurons for the layer.

Definition at line 13 of file neuronLayer.cpp.

4.2.3 Member Function Documentation

4.2.3.1 __str__()

```
void NeuronLayer::__str__ () const
```

Prints the layer details.

Definition at line 31 of file neuronLayer.cpp.

4.2.3.2 feedForward()

```
std::vector< int > NeuronLayer::feedForward (
            const std::vector< int > & inputs)
```

Performs a feedforward operation.

Parameters

inputs	A vector of input values.

Returns

The output of the layer.

Definition at line 17 of file neuronLayer.cpp.

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

- /Users/stanislav/Github/MachineLearning/ML-sigmoid-neuron/src/header/neuronLayer.hpp
- /Users/stanislav/Github/MachineLearning/ML-sigmoid-neuron/src/neuronLayer.cpp

4.3 NeuronNetwork Class Reference

Represents a neural network with multiple layers of neurons.

```
#include <neuronNetwork.hpp>
```

Public Member Functions

- NeuronNetwork (std::vector< NeuronLayer > layers)
  Constructs a NeuronNetwork with the given layers.
- std::vector< int > feedForward (const std::vector< int > &inputs)
  Performs a feedforward operation. On all the layers sequentially.
- void __str__ () const
  Prints the network details.

4.3.1 Detailed Description

Represents a neural network with multiple layers of neurons.

The NeuronNetwork class has a collection of neuron layers and provides methods to perform feedforward operations and to represent the network as a string.

Definition at line 25 of file neuronNetwork.hpp.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 NeuronNetwork()

```
NeuronNetwork::NeuronNetwork (
            std::vector< NeuronLayer > layers)
```

Constructs a NeuronNetwork with the given layers.

Parameters

layers	A vector of neuron layers for the network.

Definition at line 13 of file neuronNetwork.cpp.

4.3.3 Member Function Documentation

4.3.3.1 __str__()

```
void NeuronNetwork::__str__ () const
```

Prints the network details.

Definition at line 27 of file neuronNetwork.cpp.

4.3.3.2 feedForward()

```
std::vector< int > NeuronNetwork::feedForward (
            const std::vector< int > & inputs)
```

Performs a feedforward operation. On all the layers sequentially.

Parameters

inputs	A vector of input values.

Returns

The output of the network.

Definition at line 16 of file neuronNetwork.cpp.

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

- /Users/stanislav/Github/MachineLearning/ML-sigmoid-neuron/src/header/neuronNetwork.hpp
- /Users/stanislav/Github/MachineLearning/ML-sigmoid-neuron/src/neuronNetwork.cpp

Chapter 5

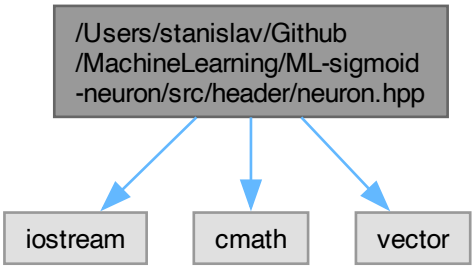
File Documentation

5.1 /Users/stanislav/Github/MachineLearning/ML-sigmoid-neuron/src/header/neuron.hpp File Reference

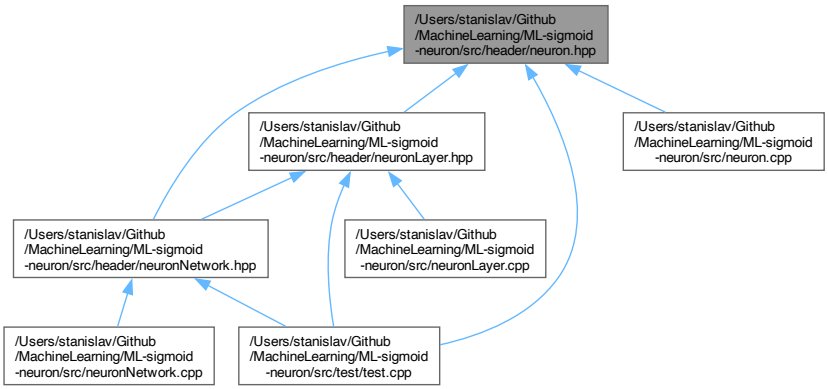
In this file the Neuron class is declared. This class represents a single neuron in a neural network.

```
#include <iostream>
#include <cmath>
#include <vector>
```

Include dependency graph for neuron.hpp:



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



Classes

- class Neuron

  Represents a single neuron in a neural network.

5.1.1 Detailed Description

In this file the Neuron class is declared. This class represents a single neuron in a neural network.

Author

Stan Merlijn

Version

0.1

Date

2025-02-14

Copyright

Copyright (c) 2025

Definition in file neuron.hpp.

5.2 neuron.hpp

Go to the documentation of this file.

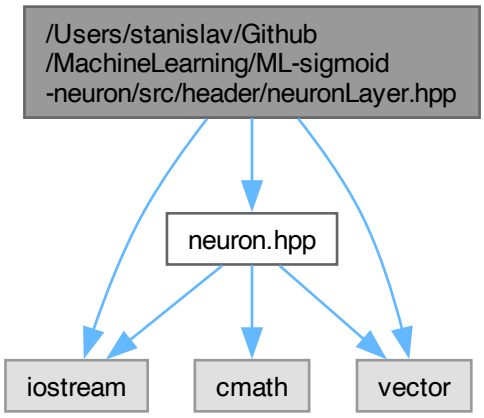
```
00001
00011 #pragma once
00012 #include <iostream>
00013 #include <cmath>
00014 #include <vector>
00015
00024 class Neuron {
00025 public:
00031     Neuron(const std::vector<double>& weights, double bias);
00032
00033
00039     double sigmoid(double x);
00040
00046     double predict(const std::vector<int>& inputs);
00047
00051     void __str__() const;
00052
00053 private:
00054     std::vector<double> weights;
00055     double bias;
00056 };
```

5.3 /Users/stanislav/Github/MachineLearning/ML-sigmoid-neuron/src/header/neuronLayer.hpp File Reference

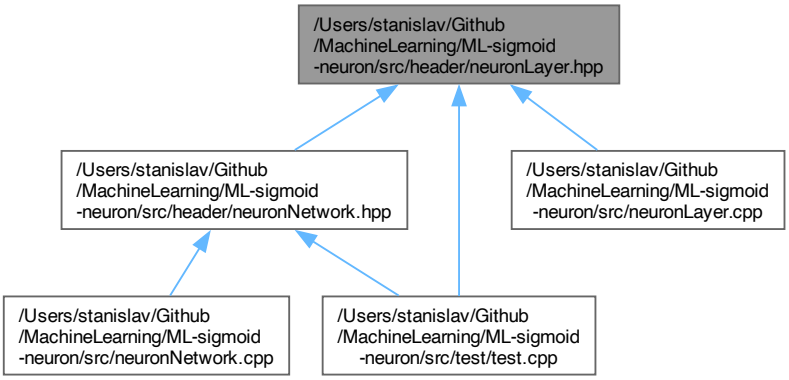
In this file the NeuronLayer class is declared. This class represents a layer of neurons in a neural network.

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

Include dependency graph for neuronLayer.hpp:



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



Classes

- class NeuronLayer

  Represents a layer of neurons in a neural network.

5.3.1 Detailed Description

In this file the NeuronLayer class is declared. This class represents a layer of neurons in a neural network.

Author

Stan Merlijn

Version

0.1

Date

2025-02-14

Copyright

Copyright (c) 2025

Definition in file neuronLayer.hpp.

5.4 neuronLayer.hpp

Go to the documentation of this file.

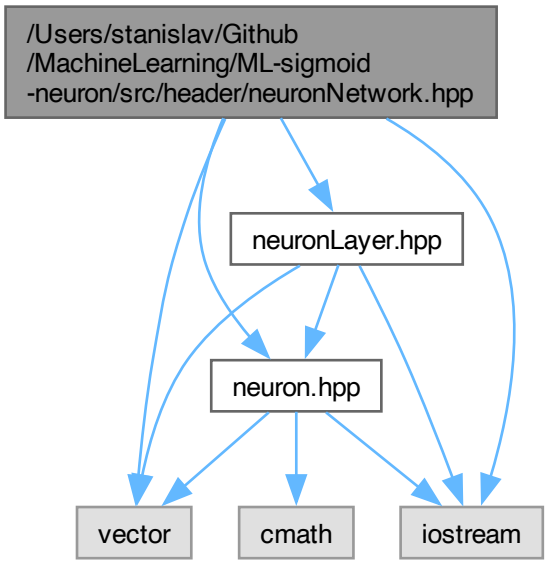
```
00001
00011 #pragma once
00012 #include <iostream>
00013 #include <vector>
00014
00015 #include "neuron.hpp"
00016
00024 class NeuronLayer {
00025 public:
00030     NeuronLayer(std::vector<Neuron> neurons);
00031
00037     std::vector<int> feedForward(const std::vector<int>& inputs);
00038
00042     void __str__() const;
00043
00044 private:
00045     std::vector<Neuron> neurons;
00046 };
```

5.5 /Users/stanislav/Github/MachineLearning/ML-sigmoid-neuron/src/header/neuronNetwork.hpp File Reference

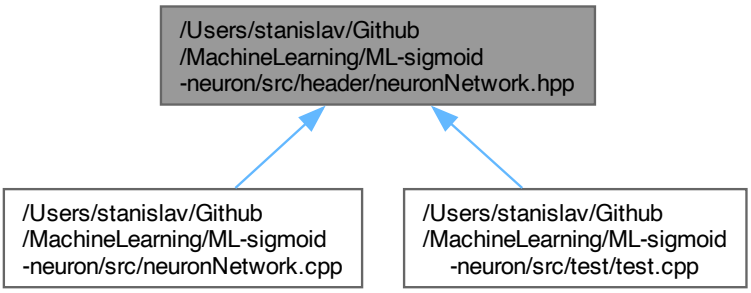
In this file the NeuronNetwork class is declared. This class represents a neural network with multiple layers of neurons.

```
#include <iostream>
#include <vector>
#include "neuron.hpp"
#include "neuronLayer.hpp"
```

Include dependency graph for neuronNetwork.hpp:



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



Classes

- class NeuronNetwork

  Represents a neural network with multiple layers of neurons.

5.5.1 Detailed Description

In this file the NeuronNetwork class is declared. This class represents a neural network with multiple layers of neurons.

Author

Stan Merlijn

Version

0.1

Date

2025-02-14

Copyright

Copyright (c) 2025

Definition in file neuronNetwork.hpp.

5.6 neuronNetwork.hpp

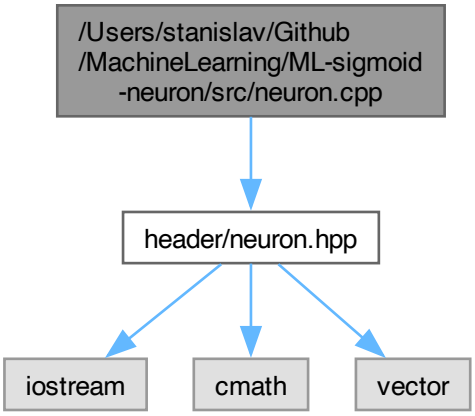
Go to the documentation of this file.

```
00001
00011 #pragma once
00012 #include <iostream>
00013 #include <vector>
00014
00015 #include "neuron.hpp"
00016 #include "neuronLayer.hpp"
00017
00025 class NeuronNetwork {
00026 public:
00031     NeuronNetwork(std::vector<NeuronLayer> layers);
00032
00038     std::vector<int> feedForward(const std::vector<int>& inputs);
00039
00043     void __str__() const;
00044
00045 private:
00046     std::vector<NeuronLayer> layers;
00047 };
```

5.7 /Users/stanislav/Github/MachineLearning/ML-sigmoid-neuron/src/neuron.cpp File Reference

In this file the Neuron class is implemented.

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



5.7.1 Detailed Description

In this file the Neuron class is implemented.

Author

Stan Merlijn

Version

0.1

Date

2025-02-14

Copyright

Copyright (c) 2025

Definition in file neuron.cpp.

5.8 neuron.cpp

Go to the documentation of this file.

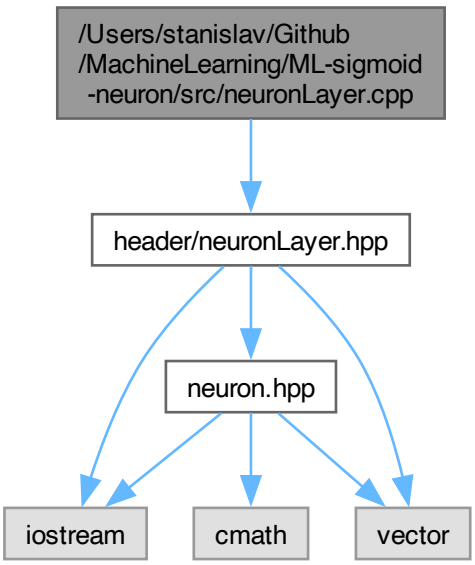
```
00001
00011 #include "header/neuron.hpp"
00012
00013 Neuron::Neuron(const std::vector<double>& weights, double bias)
00014     : weights(weights), bias(bias) {}
00015
00016 double Neuron::sigmoid(double x)
00017 {
00018     // Sigmoid activation function
00019     return 1 / (1 + exp(-x));
00020 }
00021
00022 double Neuron::predict(const std::vector<int>& inputs)
00023 {
00024     // Calculate the weighted sum of the inputs
00025     double weightedSum = bias;
00026     for (int i = 0; i < weights.size(); i++)
00027     {
00028         weightedSum += weights[i] * inputs[i];
00029     }
00030
00031     // Return the result of the sigmoid function
00032     double result = sigmoid(weightedSum);
00033
00034     // Return 1 if the result is greater than 0.5, otherwise return 0(threshold)
00035     return result > 0.5 ? 1 : 0;
00036 }
00037
00038 void Neuron::__str__() const
00039 {
00040     // Print the neuron details
00041     std::cout « "Neuron with weights: ";
00042     for (int i = 0; i < weights.size(); i++)
00043     {
00044         std::cout « weights[i] « " ";
00045     }
00046     std::cout « "and bias: " « bias « std::endl;
00047 }
```

5.9 /Users/stanislav/Github/MachineLearning/ML-sigmoid-neuron/src/neuronLayer.cpp File Reference

In this file the NeuronLayer class is implemented.

```
#include "header/neuronLayer.hpp"
```

Include dependency graph for neuronLayer.cpp:



5.9.1 Detailed Description

In this file the NeuronLayer class is implemented.

Author

Stan Merlijn

Version

0.1

Date

2025-02-14

Copyright

Copyright (c) 2025

Definition in file neuronLayer.cpp.

5.10 neuronLayer.cpp

Go to the documentation of this file.

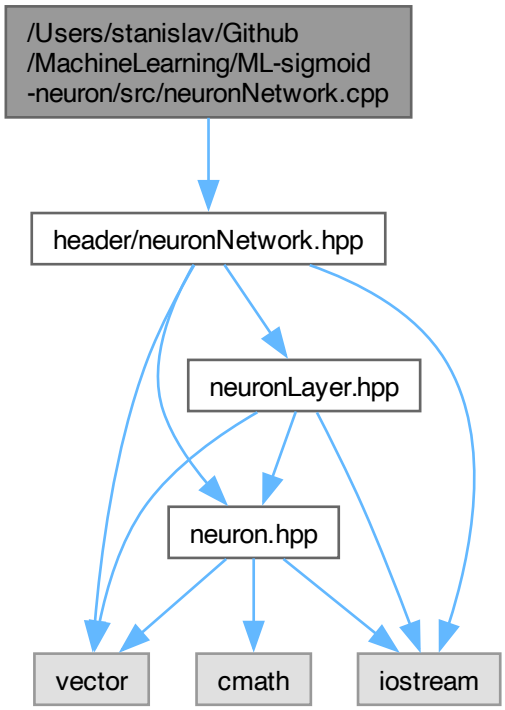
```
00001
00011 #include "header/neuronLayer.hpp"
00012
00013 NeuronLayer::NeuronLayer(std::vector<Neuron> neurons)
00014 : neurons(neurons) {}
00015
00016
00017 std::vector<int> NeuronLayer::feedForward(const std::vector<int>& inputs)
00018 {
00019     std::vector<int> outputs;
00020     // Reserve space for the outputs
00021     outputs.reserve(neurons.capacity());
00022     // Feed forward through each neuron in the layer
00023     for (int i = 0; i < neurons.size(); i++)
00024     {
00025         outputs.push_back(neurons[i].predict(inputs));
00026     }
00027
00028     return outputs;
00029 }
00030
00031 void NeuronLayer::__str__() const
00032 {
00033     // Print the layer details
00034     std::cout « "NeuronLayer with " « neurons.size() « " neurons" « std::endl;
00035     for (int i = 0; i < neurons.size(); i++)
00036     {
00037         neurons[i].__str__();
00038     }
00039 }
```

5.11 /Users/stanislav/Github/MachineLearning/ML-sigmoid-neuron/src/neuronNetwork.cpp File Reference

In this file the NeuronNetwork class is implemented.

```
#include "header/neuronNetwork.hpp"
```

Include dependency graph for neuronNetwork.cpp:



5.11.1 Detailed Description

In this file the NeuronNetwork class is implemented.

Author

Stan Merlijn

Version

0.1

Date

2025-02-14

Copyright

Copyright (c) 2025

Definition in file neuronNetwork.cpp.

5.12 neuronNetwork.cpp

Go to the documentation of this file.

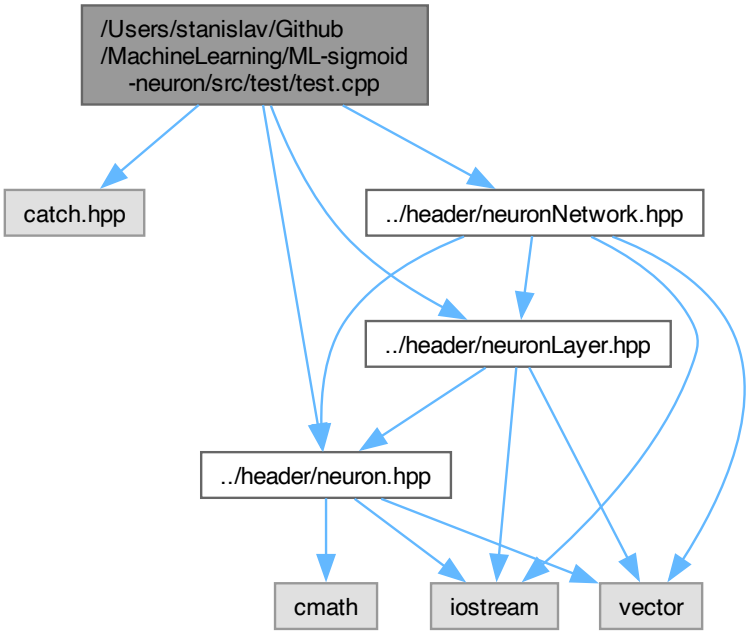
```
00001
00011 #include "header/neuronNetwork.hpp"
00012
00013 NeuronNetwork::NeuronNetwork(std::vector<NeuronLayer> layers)
00014     : layers(layers) {}
00015
00016 std::vector<int> NeuronNetwork::feedForward(const std::vector<int>& inputs)
00017 {
00018     std::vector<int> outputs = inputs;
00019     // Feed forward through each layer in the network
00020     for (int i = 0; i < layers.size(); i++)
00021     {
00022         outputs = layers[i].feedForward(outputs);
00023     }
00024     return outputs;
00025 }
00026
00027 void NeuronNetwork::__str__() const
00028 {
00029     // Print the network details
00030     std::cout « "NeuronNetwork with " « layers.size() « " layers" « std::endl;
00031     for (int i = 0; i < layers.size(); i++)
00032     {
00033         layers[i].__str__();
00034     }
00035 }
```

5.13 /Users/stanislav/Github/MachineLearning/ML-sigmoid-neuron/src/test/test.cpp File Reference

In this file the tests for the Neuron, NeuronLayer and NeuronNetwork classes are implemented.

```
#include "catch.hpp"
#include "../header/neuron.hpp"
#include "../header/neuronLayer.hpp"
#include "../header/neuronNetwork.hpp"
```

Include dependency graph for test.cpp:



Functions

- TEST_CASE ("Neuron AND gate", "[neuron]")

  In this test case we test the ability of a single neuron to learn the AND gate.
- TEST_CASE ("Neuron OR gate", "[neuron]")

  In this test case we test the ability of a single neuron to learn the OR gate.
- TEST_CASE ("Neuron NOT gate", "[neuron]")

  In this test case we test the ability of a single neuron to learn the NOT gate.
- TEST_CASE ("Neuron NOR gate (NOT OR)", "[neuron]")

  In this test case we test the ability of a single neuron to learn the NOR gate.
- TEST_CASE ("Half Adder using Two-Layer Neuron Network", "[half-adder]")

  In this test case we test the ability of a two-layer neural network to learn the XOR gate.

5.13.1 Detailed Description

In this file the tests for the Neuron, NeuronLayer and NeuronNetwork classes are implemented.

Unit tests for the Neuron, NeuronLayer and NeuronNetwork classes.

Author

Stan Merlijn

Version

0.1

Date

2025-02-14

Copyright

Copyright (c) 2025

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

Test Cases:

- Neuron for AND Gate: Tests the Neuron's ability to learn the AND gate.
- Neuron for OR Gate: Tests the Neuron's ability to learn the OR gate.
- Neuron for NOT Gate: Tests the Neuron's ability to learn the NOT gate.
- Neuron for NOR Gate (3 inputs): Tests the Neuron's ability to learn the NOR gate with 3 inputs.
- NeuronNetwork for the XOR gate with 2 inputs.

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.

5.13.3 Function Documentation

5.13.3.1 TEST_CASE() [1/5]

```
TEST_CASE (
            "Half Adder using Two-Layer Neuron Network" ,
            "" [half-adder])
```

In this test case we test the ability of a two-layer neural network to learn the XOR gate.

The XOR gate is a binary operation that returns true if the inputs are different, and false otherwise. The network consists of two layers: a hidden layer with an OR and an AND neuron, and an output layer with a *XOR* and a carry neuron.

The XOR in this case is not a traditional XOR gate, but a neuron that computes the XOR operation. It works because it can only take 3 inputs which is linearly separable. The inputs are the output of the OR and AND neurons. The are as follows:

x1	x2	OR	AND
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

So the only inputs for the XOR neuron are (0,0), (1,0) and (0,0).

XOR = OR - AND; neuron with weights {1, -1} and bias -0.5

- $x1 = 0, x2 = 0 : 1 * 0 + -1 * 0 - 0.5 = -0.5$
- $x1 = 1, x2 = 0 : 1 * 1 + -1 * 0 - 0.5 = 0.5$
- $x1 = 1, x2 = 1 : 1 * 1 + -1 * 1 - 0.5 = -0.5$

Definition at line 137 of file test.cpp.

5.13.3.2 TEST_CASE() [2/5]

```
TEST_CASE (
            "Neuron AND gate" ,
            "" [neuron])
```

In this test case we test the ability of a single neuron to learn the AND gate.

The AND gate is a binary operation that returns true if both inputs are true, and false otherwise. With a bias of -1.5 the dot product will only be greater than 0 if both inputs are 1

- $x1 = 0, x2 = 0 : 1 * 0 + 1 * 0 - 1.5 = -1.5$
- $x1 = 0, x2 = 1 : 1 * 0 + 1 * 1 - 1.5 = -0.5$
- $x1 = 1, x2 = 0 : 1 * 1 + 1 * 0 - 1.5 = -0.5$
- $x1 = 1, x2 = 1 : 1 * 1 + 1 * 1 - 1.5 = 0.5$

Definition at line 45 of file test.cpp.

5.13.3.3 TEST_CASE() [3/5]

```
TEST_CASE (
            "Neuron NOR gate (NOT OR)" ,
            "" [neuron])
```

In this test case we test the ability of a single neuron to learn the NOR gate.

The NOR gate is a binary operation that returns true if both inputs are false, and false otherwise. With a bias of 0.5 the dot product will be greater than 0 if both inputs are 0

- $x1 = 0, x2 = 0 : -1 * 0 + -1 * 0 + 0.5 = 0.5$
- $x1 = 0, x2 = 1 : -1 * 0 + -1 * 1 + 0.5 = -0.5$
- $x1 = 1, x2 = 0 : -1 * 1 + -1 * 0 + 0.5 = -0.5$
- $x1 = 1, x2 = 1 : -1 * 1 + -1 * 1 + 0.5 = -1.5$

Definition at line 103 of file test.cpp.

5.13.3.4 TEST_CASE() [4/5]

```
TEST_CASE (
             "Neuron NOT gate" ,
             "" [neuron])
```

In this test case we test the ability of a single neuron to learn the NOT gate.

The NOT gate is a unary operation that returns true if the input is false, and false otherwise. With a bias of 1 the dot product will be greater than 0 if the first input is 0

- $x1 = 1 : -2 * 1 + 0 * 1 + 1 = -1$
- $x1 = 0 : -2 * 0 + 0 * 0 + 1 = 1$

Definition at line 83 of file test.cpp.

5.13.3.5 TEST_CASE() [5/5]

```
TEST_CASE (
             "Neuron OR gate" ,
             "" [neuron])
```

In this test case we test the ability of a single neuron to learn the OR gate.

The OR gate is a binary operation that returns true if at least one of the inputs is true, and false otherwise. With a bias of -0.5 the dot product will be greater than 0 if any of the inputs are 1

- $x1 = 0, x2 = 0 : 1 * 0 + 1 * 0 - 0.5 = -0.5$
- $x1 = 0, x2 = 1 : 1 * 0 + 1 * 1 - 0.5 = 0.5$
- $x1 = 1, x2 = 0 : 1 * 1 + 1 * 0 - 0.5 = 0.5$
- $x1 = 1, x2 = 1 : 1 * 1 + 1 * 1 - 0.5 = 1.5$

Definition at line 65 of file test.cpp.

5.14 test.cpp

Go to the documentation of this file.

```
00001
00011 #define CATCH_CONFIG_MAIN
00012 #include "catch.hpp"
00013
00014 #include "../header/neuron.hpp"
00015 #include "../header/neuronLayer.hpp"
00016 #include "../header/neuronNetwork.hpp"
00017
00034
00045 TEST_CASE("Neuron AND gate", "[neuron]") {
00046     // Create a neuron with weights 1, 1 and bias -1.5
00047
00048     Neuron n({1, 1}, -1.5);
00049     REQUIRE(n.predict({0, 0}) == 0);
```

```
00050     REQUIRE(n.predict({0, 1}) == 0);
00051     REQUIRE(n.predict({1, 0}) == 0);
00052     REQUIRE(n.predict({1, 1}) == 1);
00053 }
00054
00065 TEST_CASE("Neuron OR gate", "[neuron]") {
00066     // Create a neuron with weights 1, 1 and bias -0.5
00067
00068     Neuron n({1, 1}, -0.5);
00069     REQUIRE(n.predict({0, 0}) == 0);
00070     REQUIRE(n.predict({0, 1}) == 1);
00071     REQUIRE(n.predict({1, 0}) == 1);
00072     REQUIRE(n.predict({1, 1}) == 1);
00073 }
00074
00083 TEST_CASE("Neuron NOT gate", "[neuron]") {
00084     // Create a neuron with weights -2 and 0 and bias 1
00085
00086     Neuron n({-2, 0}, 1);
00087     REQUIRE(n.predict({0, 0}) == 1);
00088     REQUIRE(n.predict({0, 0}) == 1);
00089     REQUIRE(n.predict({1, 0}) == 0);
00090     REQUIRE(n.predict({1, 0}) == 0);
00091 }
00092
00103 TEST_CASE("Neuron NOR gate (NOT OR)", "[neuron]") {
00104     // Create a neuron with weights -1, -1 and bias 0.5
00105
00106     Neuron n({-1, -1}, 0.5);
00107     REQUIRE(n.predict({0, 0}) == 1);
00108     REQUIRE(n.predict({0, 1}) == 0);
00109     REQUIRE(n.predict({1, 0}) == 0);
00110     REQUIRE(n.predict({1, 1}) == 0);
00111 }
00112
00137 TEST_CASE("Half Adder using Two-Layer Neuron Network", "[half-adder]") {
00138     // Hidden layer: compute OR and AND
00139     Neuron n_or({1, 1}, -0.5);   // OR gate
00140     Neuron n_and({1, 1}, -1.5);  // AND gate
00141     NeuronLayer hiddenLayer({n_or, n_and});
00142
00143     // Output layer: compute XOR (for sum) and carry
00144     Neuron n_xor({1, -1}, -0.5);
00145
00146     // Carry = AND; neuron with weights {1, 1} and bias -1.5
00147     Neuron n_carry({1, 1}, -1.5);
00148     NeuronLayer outputLayer({n_xor, n_carry});
00149
00150     // Two-layer network for half adder
00151     NeuronNetwork halfAdder({hiddenLayer, outputLayer});
00152
00153     // Test cases for half adder: {Sum, Carry}
00154     REQUIRE(halfAdder.feedForward({0, 0}) == std::vector<int>{0, 0});
00155     REQUIRE(halfAdder.feedForward({0, 1}) == std::vector<int>{1, 0});
00156     REQUIRE(halfAdder.feedForward({1, 0}) == std::vector<int>{1, 0});
00157     REQUIRE(halfAdder.feedForward({1, 1}) == std::vector<int>{0, 1});
00158 }
```

Index