ross erskine

[Company name] | [Company address]

Neural network specification

MNIST dataset

Contents

[Artificial neural networks technical documentation 2](#_Toc62923157)

[Class diagrams 2](#_Toc62923158)

[Conceptual Class diagrams 2](#_Toc62923159)

[Logical Class diagrams 3](#_Toc62923160)

[Program flow 4](#_Toc62923161)

[Sequence diagram 5](#_Toc62923162)

[Data dictionary 6](#_Toc62923163)

[Calculating Neural networks gradient 9](#_Toc62923164)

[Math notation and description 9](#_Toc62923165)

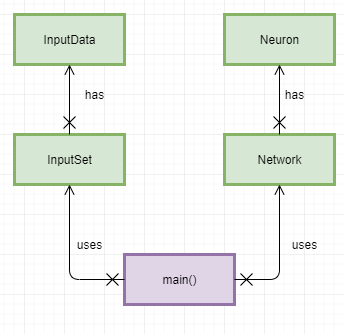
[Tests 12](#_Toc62923166)

## Artificial neural networks technical documentation

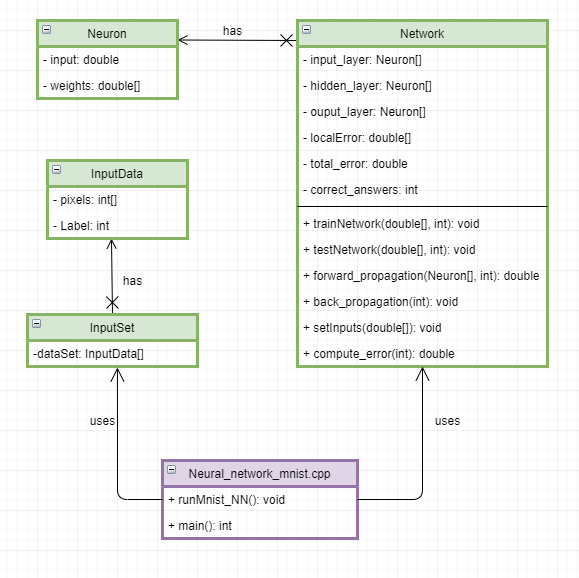
Technical documentation for the code *Artificial neural networks,* consisting of class diagram, program flow, sequence diagrams, design, data-dictionary test-table and results.

## Class diagrams

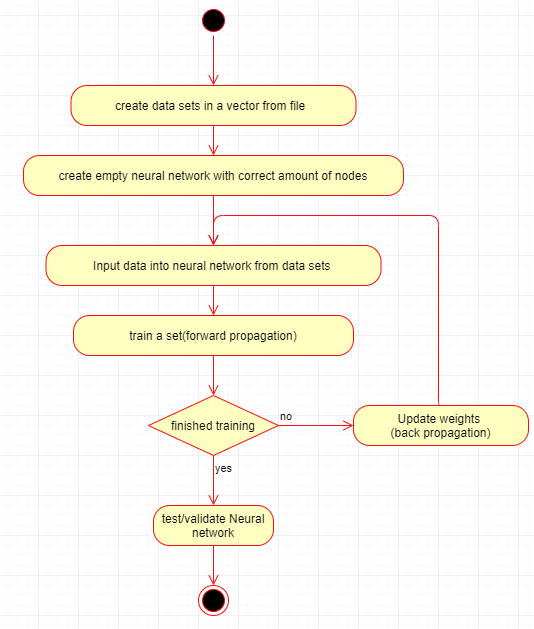
### Conceptual Class diagrams



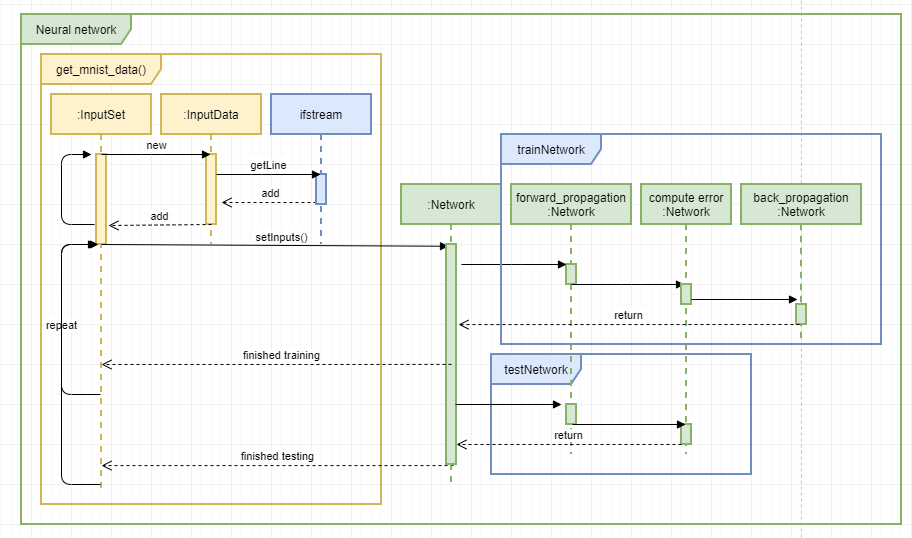
### Logical Class diagrams



## Program flow



## Sequence diagram



## Data dictionary

InputData is a single data set in a vector with attached label

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class: InputData** |  |  |  |  |
| **Field** | **Datatype** | **Validation** | **Example** | **description** |
| Label | Int | Private member | 5,7,9 | Target input |
| Pixels | Double[] | Private member | 0,145,255 | A single picture 28x28 , Pixels of data 784 of them |

InputSet is a vector of InputData or collection of datasets.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class: InputSet** |  |  |  |  |
| **Field** | **Datatype** | **Validation** | **Example** | **description** |
| dataset | InputData[] | Private member |  | A data set of 28x28 size pictures with attached labels |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class: Neuron** |  |  |  |  |
| **Field** | **Datatype** | **Validation** | **Example** | **description** |
| Input | double | Private member |  | One Input of one neuron |
| Weights | Double | Private member,  Size equals next layer size. | 30,15,10 | Number of weights depends on next layer, all weights start with random number between 0 and 1. |

Network class is a collection of layer Neuron's. the constructor can create the size network such as (3,2,1) and random size weights added to that input. Data to inputs are added at the start of train or test network.IL = input layer, HL = Hidden layer, OL = output layer, size\_of = number of nodes in layer

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Class: Network** |  | |  | |  | |  |
| **Field** | **Datatype** | | **Validation** | | **Example** | | **description** |
| Input\_layer | Neuron[] | | Private member | | 784 | | First layer of network size depends on amount of inputs example 28 x 28 = 784 input neurons |
| Hidden\_layer | Neuron[] | | Private member | | 100,30,15 | | Hidden layer is layer between input and output layer, can bea any size. |
| Ouput\_layer | Neuron[] | | Private member, size 10 | | (0,0,0,0,0,0,0,1,0,0). | | Output layer should be size of 10, to indicate which number being guessed example: if label is 7 output would be trying to give this output (0,0,0,0,0,0,0,1,0,0). |
| localError | Double[] | | Private member, size 10 | | (1,1,1,1,1,1,1,0,1,1) | | Error given by for each output layer with loss function  = target output  = actual output |
| Total\_error | double | | Private member | |  | | function mean Squared error:  MSE = |
| Correct\_answers | Int | | Private member | | 1,10 | | Used in validation phase to see how many correct guesses the trained network makes. |
| **Function** | | **Return type** | | **Parameters** | | **description** | |
| trainNetwork | | Void | | Double[], int | | trainNetwork consists of first initial data added to inputs of input layer, forward propagation sends the data to each layer, the total and local error is calculated, information is sent to csv file then finally ends with back propagation, needs to be in loop for multiple sets | |
| testNetwork | | Void | | Double[], int | | test network is same as train except uses final guess and does not upload to file or back propagation | |
| Forward\_propagation | | Double | | Neuron[], int | | forward propagation uses summation and activation functions then sets input of next layer | |
| Back\_propagation | | Void | | Int | | starts by updating hidden to output layer by calculating the individual weight partial derivative using the chain rule, then the same is done updating weights on input to hidden layers weights | |
| setInputs | | Void | | Double[] | | sets first data into network | |
| Compute\_error | | Double | | Int | | function mean Squared error:  MSE = | |
| Compute\_final\_error | | Double | |  | | creates final error from output of output layer then averaged and places local errors in local\_error. | |
| Final\_guess | | Int | |  | | final guess, number closest to 1 wins. | |
| randDouble | | Double | | Double,double | | "random number engine is a function that generates uniformly distributed sequence of integer values. Distribution is a function that generates a sequence of values according to a mathematical formaula given a sequence of value from an engine." Stroustrup (2014: p914) | |
| randInt | | Int | | Int, int | |
| Summation\_operator | | Double | | Neuron[], int | | Summation operator the sum of all inputs multiplied by weights that corresponds to the next layers input.  summation = bias +  = each input in layer  = each weight that corresponds to each input  Bias = extra input that; input is set to 1 | |
| Activation | | Double | | Double | | Activation function squashes the number and signifies how much a neuron has fired.  Next neurons input =  X = summation  e = Euler’s constant = 2.71828 | |
| Is\_target\_output | | Bool | | Int, int | | represents the target output if the label is the same as the iteration it will send back one. example label 4 (0,0,0,0,1,0,0,0,0,0) | |
| NN\_data | | Void | | Double, int | | uploads error and label to csv file | |

## Calculating Neural networks gradient

Based on formulas from Taylor (2017).

### Math notation and description

Table : artificial neural network math notation and description

|  |  |
| --- | --- |
| Notation | description |
| 𝜕E | partial derivative of total error |
|  | partial derivative of a specific weight |
| 𝜕 | partial derivative of output from specific Neuron in output layer |
| 𝜕Z | partial derivative of sum of all outputs |
|  | partial derivative of output from specific neuron in input layer |
| 𝜕 | partial derivative of output in specific neuron in hidden layer |
| 𝜕B | partial derivative of sum of all inputs to hidden layer |
| δ | Delta notation to represent part of an equation |

Always start with hidden to output layer.

#### Between hidden and output:

Partial derivative broken down using chain-rule.

= ,

= ( – ), = ( 1 – ), = ,

),

Delta\_z = = ( 1 – ),

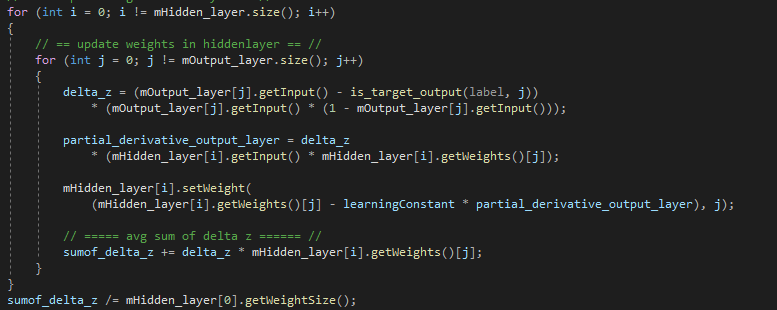
= )

Equation : artificial neural network between hidden and output partial derivative.

#### Output Bias weights

=

Equation : : artificial neural network output bias weights partial derivative



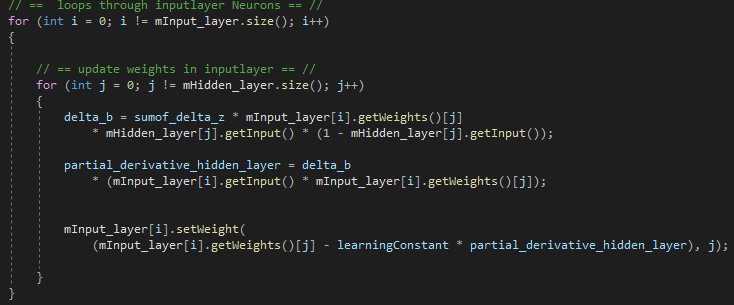
#### Between input and hidden:

=,

= , = , = , =

,

Delta\_b = = ,



## Tests

Table : Artificial neural network test-table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test: | Learning rate | Hidden layer neurons | Epochs | Validation | filename | results |
| 01 | 0.001 | 30 | 10000 | 131/1000 | NN\_data\_file\_001 | Figure 30 |
| 02 | 0.01 | 30 | 10000 | 120/1000 | NN\_data\_file\_002 | Figure 31 |
| 03 | 0.1 | 30 | 10000 | 82/1000 | NN\_data\_file\_003 | Figure 32 |
| 04 | 0.001 | 15 | 10000 | 79/1000 | NN\_data\_file\_004 | Figure 33 |
| 05 | 0.001 | 60 | 10000 | 78/1000 | NN\_data\_file\_005 | Figure 34 |
| 06 | 0.001 | 100 | 10000 | 101/1000 | NN\_data\_file\_006 | Figure 35 |

