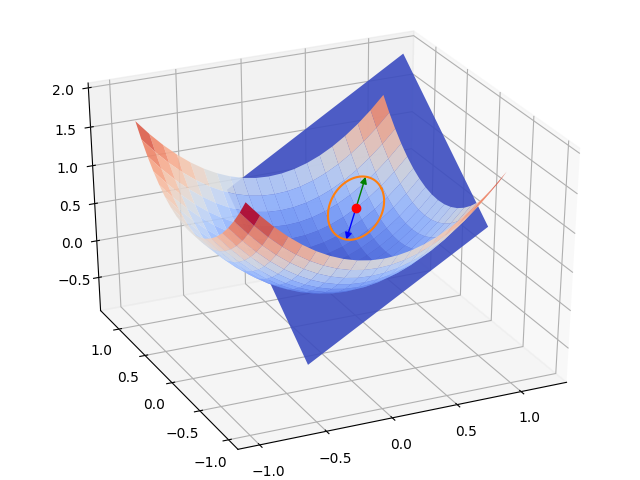
# Building A Neural Network From Scratch C++



Name of Student: Niranjan

Class: XII

Year: 2019 - 2020

# Certificate

This is to certify that Niranjan, a student of class XII has successfully completed the research under the guidance of

Mrs. Rachna during the year 2019 - 2020 regarding his project titled “*Building a Neural Network From Scratch C++*”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Internal Examiner’s Principal’s

Signature Signature

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

External Examiner’s

Signature

# Acknowledgement

I would like to thank my school for giving me the opportunity and for providing all facilities to meet my project requirements.

I wish to express my sincere gratitude to Mrs. Reshma Ganesh,

Principal of SSRVM Bangalore East, for the successful outcome of this project.

I would like to thank my computer science teacher Mrs. Rachna for her guidance and support.

I would also like to thank my parents for providing me with the necessary resources to complete this project.

# Index

# Objective of the Project

The objective of this project is to build a good understanding behind neural networks, by making one from scratch in C++ using no external libraries and a bare minimum of libraries or methods which has not been thought to us in the XI and XII standard of CBSE.

I have used the MNIST handwritten digits data-set to train the neural network.

I have shared all that I have learned by doing this project and hope I do it justice.

# Introduction

The project consists of five total files, whose functions have been described:

1. MNISTRead.cpp: Reads the MNIST data and provides it to the rest of the files.
2. DataHandle.cpp: All the File I/O takes place in this file, used to save and load a trained neural network.
3. Network.cpp: The backbone of the project, it has classes that make and train the network.
4. Train.cpp: Uses the Network class from Network.cpp to create, train and save a neural network.
5. runFromModel.cpp: Uses the saved model of a network created in Train.cpp to predict the handwritten digit.

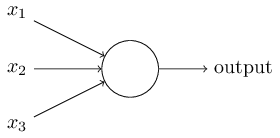
I would like to take this section to describe what the network actually does. I feel it belongs in the introduction as it is not any C++ code

# What is a Neural Network?

# The Sigmoid

A neural network is a network of “Neurons” that are connected to one another to simulate complex logic and abstract decisions that would be hard to create a hard and fast algorithm for.

At the cornet stone of neural networks, we have the *sigmoid neuron*:



This neuron is the building block of the network. It has many inputs and one output, all of them range from [0 - 1]. Each input is associated with a weight ‘*Wi’* is the weight associated to the ‘*ith’* input of the neuron. Each neuron also has another value, called bias or ‘*B’*. The output of the neuron is calculated as follows:

\*Here, *ai* refers to the *ith* input to the neuron



# What’s Really Happening Here? Neural Network as a function

I lied, the sigmoid neuron isn’t that simple, as you may have noticed I mentioned the output lies between [0 - 1], but you might say what if the sum of all the inputs crossed 1?

We fix that by passing the output of the calculation through a special function called.... ‘*sigmoid*’ function! It is defined as

\*graph on the left 

The function outputs values close to 1 for large values of *x* and close to 0 for small values of *x*.

The network is built by many of these neurons connected to each other, each taking input from the previous ones to finally reduce to desired number of outputs, this can hence be treated as a function with many inputs, such as:



However, since it is so heavily abstracted, we rarely know what the function is doing and just optimize the function as much as we can, but how do we optimize a function whose definition is unknown to us?

# The Heart of ‘Machine Learning’ Gradient Descent

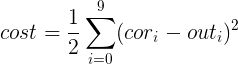
This is where gradient descent comes into action. It is a powerful tool that allows us to optimize any function.

This is where it gets interesting. The inputs to out neural network or function are the binary (0, 1) values of the pixels of the handwritten digits, each image being 28 x 28, meaning our function has 784 inputs, but we only want how likely the number we gave as inputs is a digit from 0 - 9, which means we have 10 outputs(0 - 9). Solving the problem of identifying numbers is as good as optimizing our function to a point where it can classify digits its never seen before.

Now that we know what our function is, what value are we aiming to optimize?

For each set of inputs, we want the output to tell us exactly what number it is. This means if we give it an input of the handwritten number ‘2’ the output ‘*out3*’ should be maximum(1) and everything else minimum(0). This gives as a value to optimize which is called as the cost function, which grades how good the network has identified the current set of inputs.

\*Here cori stands for the correct output(0, 1)



This gives us a way to judge how good a network is by passing all the weights and biases to the *cost* function and making the value of *cost* minimum will give us the result we need.

To minimize the *cost*, we start by assuming the network to have random values and calculate the slope or gradient of the cost function, and update all the weights in the negative direction of the slope.

That might be hard to understand, but it has a very good analogy. If we visualize the network having only two weights, then our input would be any value on the  plane then the output of the *cost* function would be a single value, which is mapped onto the *z-axis* therefore we can visualize the current setup as some function in 3-Dimensions, initializing our network with random values is as good as just picking a random value on the function curve. Now we can imagine a ball at the current position which is rolling down the slopes to find a nice minimum. We can simulate this by calculating the slope at the current position and changing our position by a small value in the negative direction of the slope. The small value will be called *learning rate* from now on, also represented by  . This is the heart of ‘machine learning’. The actual calculus involved in finding the slope is simple, but will require a much longer introduction still. I will try my best to explain what is happening as and when it happens int he code.