HP In-Semester Project Report

Members:
1) Rajeev Veeraraghavan (170905252)
2) Arjun Goel (170905284)

Introduction

We have attempted to build a neural network capable of solving quadratic equations with a single variable x. The expected speed up from the Neural network will make it superior to the traditional Newton Raphson method used to solve such equations.

The neural network gets the zeros of an equation of the form : $a(x^2) + b(x) + c = 0$.

The uploaded ipynb notebook can be referred to for the code.

Architecture

The input consists of 3 nodes (values of a,b and c) and the output contains 1 node (value of the solution) .We tested with 2 different sets of architectures.

Architecture - 1

- *Input Layer : 3 nodes* -> *Hidden Layer : 5 nodes* -> *Hidden Layer : 3 nodes* -> *Dense Layer : 1 node*

- *Activation : Relu*

Architecture – 2

- *Input Layer : 3 nodes* -> *Hidden Layer : 11 nodes* -> *Hidden Layer : 8 nodes* -> *Hiddem Layer : 5 node* -> *Dense Layer : 1 node*

- *Activation : Relu*

Dataset

The data was generated for training. The following steps were followed for doing so.

Values of a,b,c each in the range of -20 to 20 are generated and the corresponding value of the roots of x are found using numpy.
The values of a,b,c are normalised such that they lie between -1 and 1. Only those values of a,b,c are considered which produce a real value of x between -1 and 1. This is done to prevent the error at each step of training from getting too large.

The total size of the training set was : 20292 examples.

Training

The data was split into train and test sets and stored it in csv files. We trained the model on both the architectures for 100 epochs

Results

Architecture 1

When we fit the network to training data, we observe that loss decreases sharply in the beginning, and then the decrease is very minimal in later epochs. As for the accuracy, it is very low to start with, and only marginally increases throughout the iterations.

When we evaluate our model on the test set, we get similar results to that of the training set. The loss is slightly higher and the accuracy is more or less the same.

In either case, the highest accuracy we achived (0.10) was nowhere close to the 93% reported in the paper. This was despite trying out several combinations of activation layer functions.

Architecture – 2

When we fit the second network to the training data, we observe that loss decreases sharply in the beginning, and then the decrease is very minimal in later epochs, similar to the first neural network. As for the accuracy, it is slighlty higher than the previous 4 layered network, but is still extremely low.

When we evaluate our model on the test set, we get similar results to that of the training set. The loss and accuracy are identical to that in the training set indicating that the data generalizes well (no overfitting).

The highest accuracy we achived (0.11) was far lesser than the 95% reported in the paper. This was despite trying out several combinations of activation layers functions.