GE 461 PROJECT 3 REPORT (Supervised Learning)

Fahad Waseem Butt 21801356

INTRODUCTION

The goal of this project was to implement an Artificial Neural Network (ANN) regressor and learn its weights implementing the backpropagation algorithm. The implementation was required to be able to support an ANN for Linear Regression, i.e. without any hidden layers, as well as for an ANN with a single hidden layer. The loss function for this particular implementation was required to be the Sum of Squared Errors. The activation function, to define the hidden units, was to be the Sigmoid, and the learning algorithm followed was Stochastic. Train and Test datasets were provided for this project, to be used for the implementation of the ANN. As this is a new topic, the Python code, written on Google Colab, for this assignment has taken a main inspiration from the referenced source [1].

BACKGROUND

An ANN takes inspiration from the biological workings of neurons in the human brain. A basic ANN is made up of an input layer, hidden layer and output layer [2]. The input layer collects data from a dataset, and passes it to the hidden layer, where computations are then performed, and then the output layer has the final predictions formed from the computations on the data. Each layer can have one or more than one nodes, the number of which is determined by the size and type of data, as well as the level of computation required [3]. A node (or neuron) in a neural network takes inputs from all the nodes in the previous layer before it, with each connection having a different weight, and a unique bias. This may be demonstrated as follows, where Y is the output, W is the weight, X is the input and B is the bias.

$$Y_i = W_1X_1 + W_2X_2 + \ldots + W_nX_n + B_i$$

In the working of an ANN, the following steps are followed: a dataset is given to the input layer which then passes to the hidden layer(s); in the hidden layer(s), each hidden neuron is connected to each input before it, the inputs are multiplied by the weights and a bias is added; a second step that takes place in the hidden layer is that an activation function is applied to the results after the computations between the inputs, weights and bias; after passing through the hidden layer, the output layer is reached which has the final output [3]. This procedure can be called Forward Propagation. The activation function required for this project was the Sigmoid, which is as follows

$$S(x) = \frac{1}{1+e^{-x}}$$

In this project, a Backpropagation Algorithm was to be implemented. In backpropagation with gradient descent, the following steps are followed: initially, the weights are selected randomly; then the derivative, with respect to the current weights, is calculated for the error; finally the new weights are found using a learning rate, smaller learning rates tend to be better, and backpropagated; this process continues until the loss is minimized [3]. The new weights found in the backpropagation algorithm are found based on the following equation, where W represents the weight and L represents the learning rate.

$$^{new}W_x = {}^{old}W_x + L\left(\frac{\partial Error}{\partial {}^{old}W_x}\right) \qquad [3]$$

The loss function chosen for this particular project was the Sum of Squared Errors, which is as follows.

$$\Sigma_{i=1}^{n}(Y_i - \widehat{Y})^2$$

A requirement of this project was to have the capability of using the ANN as a Linear Regressor, this can simply be achieved by having an implementation with no hidden layers or backpropagation [4].

RESULTS AND DISCUSSION

Part (a):

No, a Linear Regressor is not sufficient as it gives a loss (Sum of Squared Errors) of 6.640506, whereas, while using an ANN with a single hidden layer, the resulting loss is significantly reduced. The minimum number of hidden units would be 2 hidden units. With 2 hidden units, a loss of 2.740185 is observed. It is observed that the loss for the ANN with a single hidden layer (2 hidden units) has an improvement by 58.7% when compared with the loss for just the Linear Regressor.

After testing with different numbers of hidden units (1, 2, 4, etc.), it was found that 0.05 may be a good value for the learning rate.

The weights are initialized as randomly found small values, this is because they will be updated as they go through backpropagation.

It was found that having 1000 iterations was a good generalization for different combinations of learning rates and number of hidden units. After a point, there is no significant decrease in the loss, and the value seems to start to saturate, and when such a phenomenon is observed, it is best to stop the iterations.

Normalization is done as with the following equation, with an adjustment by adding 1 to the numerator to avoid having zero values and losing meaningful data.

$$Z_i = \frac{X_i - min(X) + 1}{max(X) - min(X)} \quad [5]$$

Because the dataset is distributed across a large range of values including both positive and negative numbers, normalization becomes important. When a dataset is normalized, the lower and upper bounds of the data are changed to 0 and 1 respectively, i.e. the data is normalized between 0 and 1. Each point holds the same meaning relative to other points as it did when it was in the raw data. A big effect of normalization is that the weight held by each point is also normalized, and the significance of a point with a very large magnitude and a point with a low magnitude become relative to each other.

Part (b):

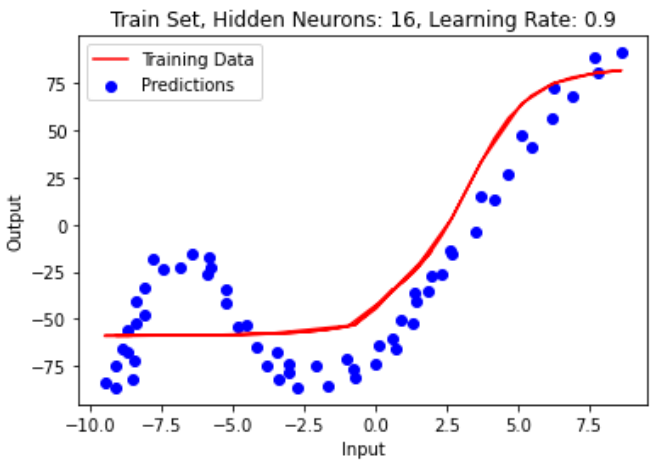

Figure 1a: Plot for Training Set

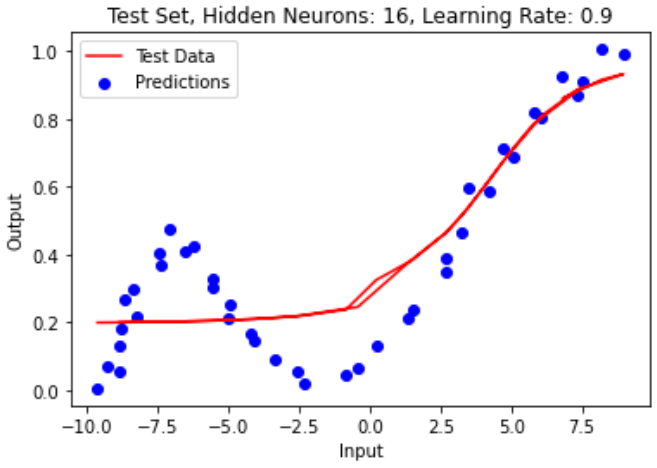

Figure 1b: Plot for Test Set

ANN Used (specify the number of hidden units): 16
Learning Rate: 0.9
Range of Initial Weights: [−0.15, 0.15]
Number of Epochs: 1000
When to Stop: 1000
Is Normalization Used: Yes
Testing Loss (averaged over training instances): 1.567824774565305
Test Loss (averaged over test instances): 1.181781810719858

Part (c):

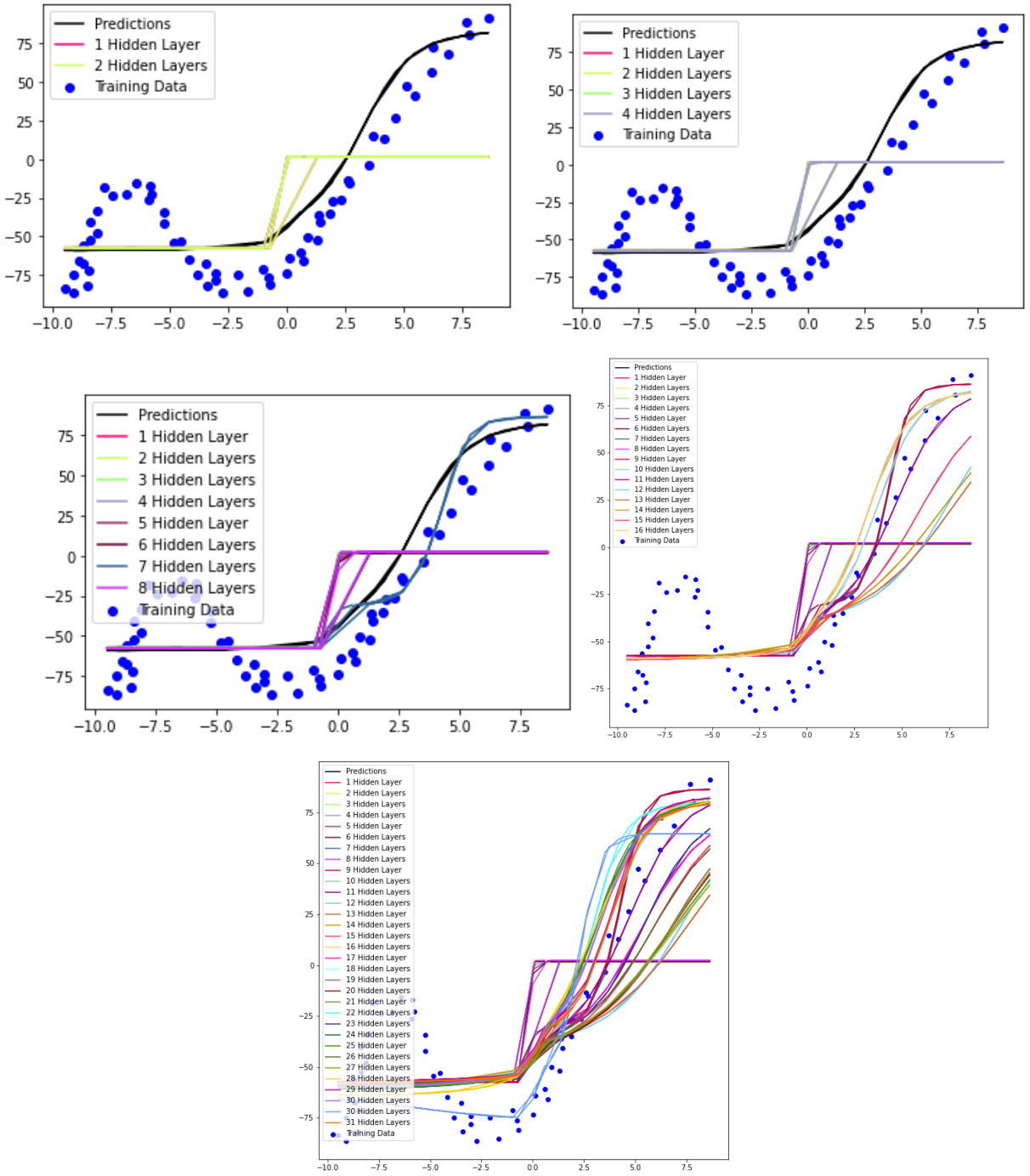

Figure 2: Plots for Each ANN with Different Hidden Layers

ANN with # Hidden Layers	2	4	8	16	32
Standard Deviation in Training Loss	0.1673145858 6032424	0.1711584822 942323	0.1834104607 6266976	0.5010113739 223707	0.5440829613 703048
Loss Averaged Over Training Instances	2.7928334358 64634	2.7478528156 97313	2.6865246510 733174	1.5678247745 65305	1.6597223775 400258
Standard Deviation in Test Loss	0.1431449542 50071	0.3589139308 044587	0.3713252790 74256	0.2931997517 8743773	0.3348993539 020798
Loss Averaged Over Test Instances	1.8035437100 258893	0.9544538842 854158	0.9899011196 158755	1.1817818107 19858	1.4574841284 097098

Table 1: Table Reporting Training and Test Set Losses

From the Plots in Figure 2, it can be noted that as the number of hidden layers (complexity) increases, the predictions begin to represent the dataset better and begin to match the trend. There are especially good results for when the number of hidden layers is any number that can be found using 2^k, where k is such that k=1, 2, 3, and so on. The improvement in the results by increasing the number of hidden layers is not linear, and such an inference can be made using the observation that the results for 16 hidden layers were slightly better than the results for 32 hidden layers, so were chosen in part(b). However, continuing the same example, from 17 hidden layers up to 31 hidden layers, an improvement is noted. When referring to the Training and Test Set Losses in Table 1, it is noted that the average loss in both the training and test is decreasing as the number of hidden layers increases. The standard deviation in the loss values for both the training and test do not follow any trend. Based on these observations, it can be understood that increasing the complexity (number of hidden layers) of an ANN tends to improve its ability to learn.

REFERENCES

[1] Paudel, R., 2020. Building a Neural Network with a Single Hidden Layer using Numpy. [online] Medium. Available at: https://towardsdatascience.com/building-a-neural-network-with-a-single-hidden-layer-using-numpy-923be1180dbf

[2] Park, Y. and Lek, S., 2016. Artificial Neural Networks. Developments in Environmental Modelling, pp.123-140.

[3] Analytics Vidhya. Artificial Neural Network | Beginners Guide to ANN. [online] Available at: https://www.analyticsvidhya.com/blog/2021/05/beginners-guide-to-artificial-neural-network/.

[4] Paolucci, R., 2020. Linear Regression v.s. Neural Networks. [online] Medium. Available at: https://towardsdatascience.com/linear-regression-v-s-neural-networks-cd03b29386d4

[5] Statology. 2021. How to Normalize Data Between 0 and 1 - Statology. [online] Available at: https://www.statology.org/normalize-data-between-0-and-1/