Project III

Supervised learning

Caner Canlıer 21702121

Bilkent University

GE 461 Introduction to Data Science

1st of May, 2022

(a)

1. Linear regression models are also known as single-layer neural networks because they are made up of only one artificial neuron. However, while regression models deal with linear dependencies; neural networks can deal with nonlinear dependencies. As a result, if a data set contains nonlinear dependencies, neural networks should outperform regression. Since our data set is not a linear data set, linear regressor is not sufficient. Also, for the problem, I started with one hidden units when epochs and learning rate are fixed to 10000 and 0.002, respectively. However, sum of squared error value become 167386.63 which is very large number. Therefore, I tried 2 and 3 hidden units and come up with 165643.80 and 39059.79 errors, respectively. There is a sharp drop when we have 3 hidden units, so I think minimum number of hidden units to come up with a good model would be 3.

2. While deciding our learning rate, if we choose very small number, it may take infinite time to achieve an optimal solution, on the other hand, if we choose very big number, we may miss the optimal solution. Unfortunately, we cannot analytically calculate the optimal learning rate for a given model on a given dataset. Instead, by using empirical optimization procedure we can determine a good (or good enough) learning rate. The range of values to consider for the learning rate is less than 1.0 and greater than 10^-6. For this project, I tried lots of learning rates with fixed minimum hidden units 3 and epochs 10000.

Learning Rates for Fixed values	Train Error Average	Test Error Average
0.01	840.97	868.82
0.005	572.38	802.69
0.001	371.39	575.80
0.0001	1593.82	1745.92

From this table, we can interpret that our learning rate should be between the range of 0.01 and 0.001 depending on the epochs and hidden units.

3. We have several options to initialize the weights such as zero initialization, random initialization or heuristic for weight initialization. I both tried zero initialization and random initialization, however, I found out that random way gave better solution so I choose random initialization by using np.random.rand() function due to easy implementation and adequately accurate solution.

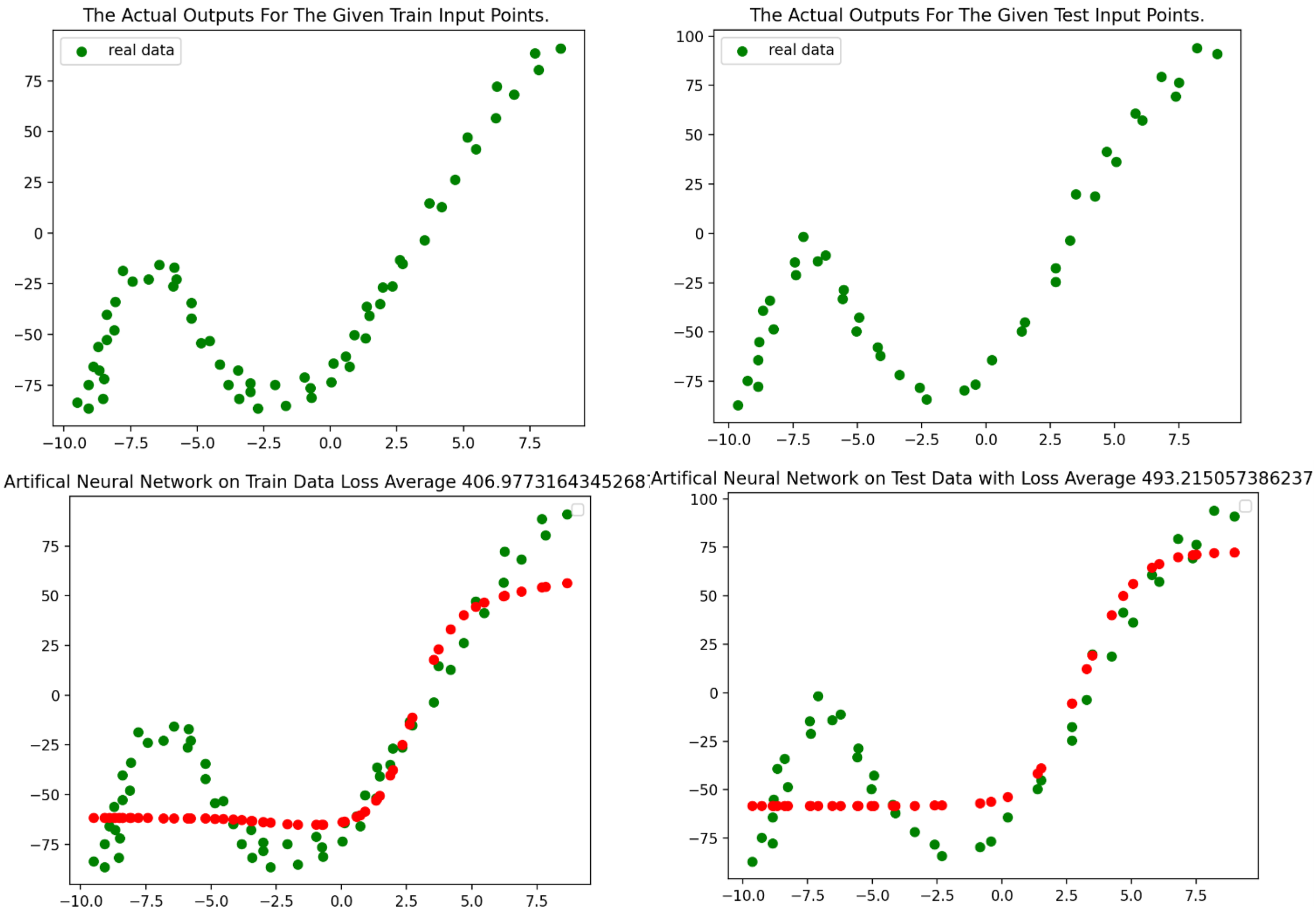
4. The proper number of epochs is determined by the dataset's intrinsic perplexity (or complexity). As rule of thumb, I started with a value that is three times the number of columns in the data. When the model continues to improve after all epochs have been completed, I tried again with a higher value. So basically, I used empirical method. If loss value starts to increase which means accuracy begins to decrease, we should stop. First, I tried stop right after finding the first increased loss value, however, in this case I stopped immediately. Therefore, I decided to monitor the performance of the model during training. Performance of the model is evaluated on the validation set at the end of each epoch, which adds an additional computational cost during training. This can be reduced by evaluating the model less frequently. For this purpose, I check model every 3 training epochs. I tried lots of epochs with fixed minimum hidden units 3 and learning rate 0.001.

Epochs for Fixed values	Train Error Average	Test Error Average
100	3390.28	3073.95
1000	1081.87	1382.63
10000	443.78	607.10
30000 (break at 25785)	410.36	465.97

From this table, range between 10000 and 30000 epochs works well for our fixed variables.

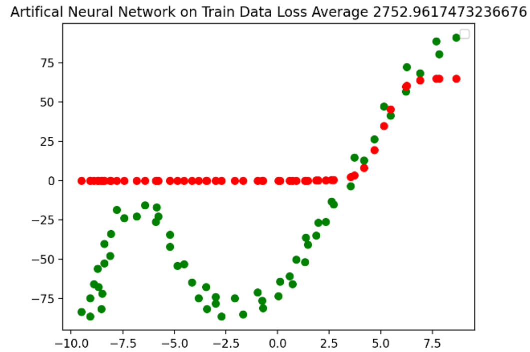
5. There is no need for normalization since data set is already normal.

(b)

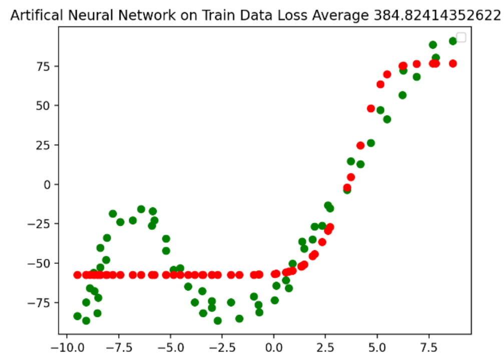


- ANN used (specify the number of hidden units): 32
- Learning rate: 0.005
- Range of initial weights: [0,1]
- Number of epochs: 1000
- When to stop: For train data set: 723, for test data set: 679
- Is normalization used: No
- Training loss (averaged over training instances): 406.97
    - Test loss (averaged over test instances): 493.21

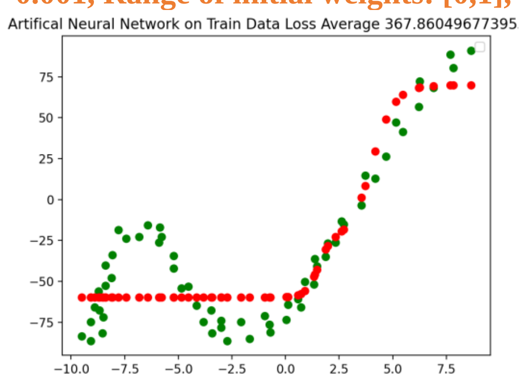
(c) **ANN = 2, Learning rate = 0.0005, Range of initial weights: [0,1], Number of epochs: 20000**



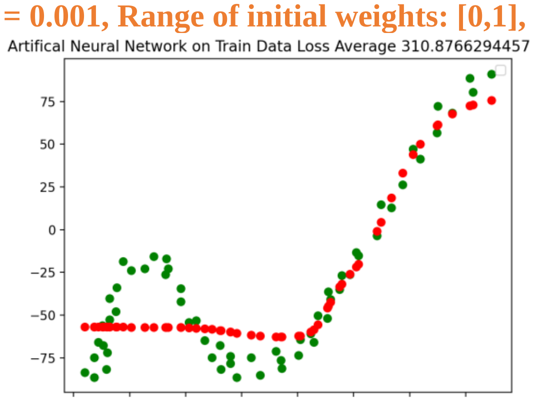
ANN = 4, Learning rate = 0.001, Range of initial weights: [0,1], Number of epochs: 20000



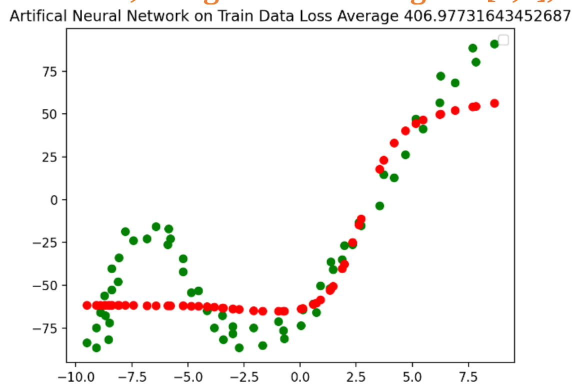
ANN = 8, Learning rate = 0.001, Range of initial weights: [0,1], Number of epochs: 10000



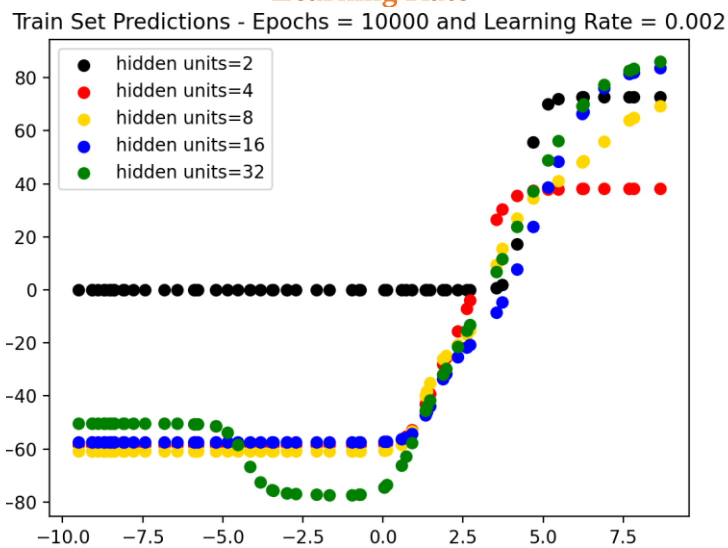
ANN = 16, Learning rate = 0.001, Range of initial weights: [0,1], Number of epochs: 10000



ANN = 32, Learning rate = 0.005, Range of initial weights: [0,1], Number of epochs: 1000



Train set Prediction Distribution with Different Hidden Units with Fixed Epochs and Learning Rate



Hidden units (with same inputs)	Train Loss Averaged	Train standard deviation	Test Lost Averaged	Test standard deviation
2	2752.96	24.98	1517.49	26.63
4	384.82	42.17	606.16	40.62
8	404.86	41.54	527.32	52.84
16	345.87	41.98	469.99	53.88
32	226.97	43.68	341.54	52.69

According to my findings, when we have bigger learning rate, more hidden units work better. On the other hand, if we have less hidden units, then we should have smaller learning rate to optimize the model. Also, we observed that having too much epochs might cause over-fitting so we need to do some empirical study to decide it. Also, when we fix all the inputs and change the only hidden unit number, we can say that our model gets more complex as it can be seen in the above graph. In our case, with the fix value that I decided, increasing the number of hidden units improved the accuracy.