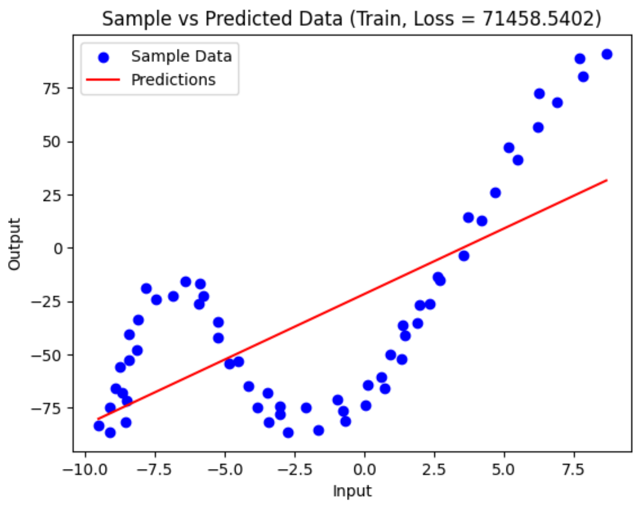
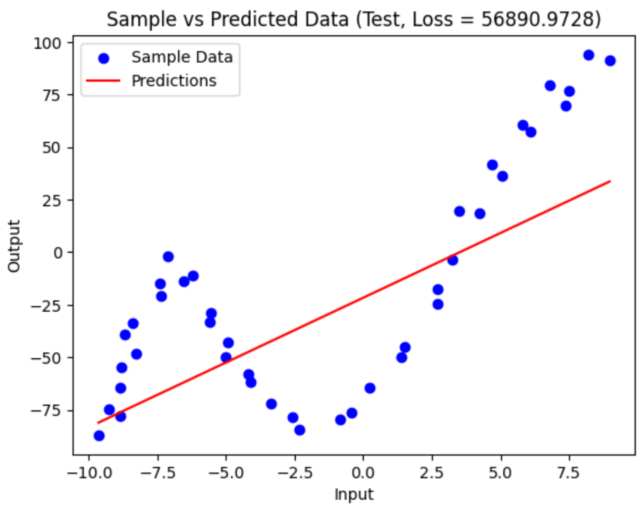
Bilkent University / GE461 Introduction to Data Science / Project 3 Report

Kerem Şahin / 21901724

PART A)

Linear Regressor or Single Layer?: When we visualize the data, we can quickly realise that a linear regressor is not sufficient for this dataset. The relationship between the input and the output is not linear. Therefore, a linear regressor cannot capture the non-linear relation. Also, the losses are higher when compared to the model with hidden layer. We can see its





visualization as follows:

Minimum number of hidden units required: In order to give an estimate of the minimum number of hidden units, we can look at both the plots and the loss values for each model with different hidden units. Since the suggested values for these values in part C are 2,4,8,16 and 32, I have tried these values out one by one on the test dataset. (*Note: Due to the nature of gradient descent, on different runs, one might get slightly different results but they will be parallel to what I gathered on the report*). The loss values are shown as follows:

Hidden Unit Size	Loss Value (SSE)
2	33730.6318
4	28825.2878
8	20298.6005
16	19522.6099
32	17405.5504

Higher hidden unit sizes yield better results, but the difference in losses before and after the hidden unit size of 8 is too large. On top of that, the plots of the hidden units with sizes 8 and higher had better plots than lower-sized hidden units. Therefore, a bare minimum value would be 8, but I would recommend 32 units as its graph are better. Due to page constraint, I will already display the plots in Part C, so plots for Part A will be visible in Part C.

Optimal Learning Rate: Since I saw that a hidden layer size of 32 is optimal, I will select my learning rate by tuning it on 32 layers. Since it is common practice, I will check for common values like 0.01, 0.005, 0.001, and 0.0005 and check their loss value and plots on the test set. The results are as follows:

Learning Rate	Loss Value (SSE)
0.01	35558.9259
0.005	27236.2268
0.001	13239.1557
0.0005	13482.0086

As we can see, the lowest loss value was gathered for a learning rate of 0.001. On top of that, its plot is also the most optimal plot. Therefore, we can conclude that a learning rate of 0.001 is optimal.

Weight Initialization: A usual technique for initializing the weights is to choose randomly selected numbers from a univariate Gaussian distribution of mean 0 and variance 1. I used the numpy library in order to make this initialization.

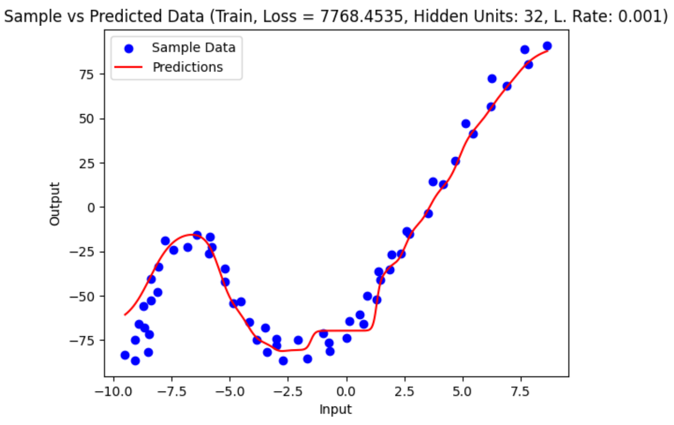
How many Epochs?: I tested epochs of 1000, 5000, 10000, 20000, 50000, and 100000. The algorithm will stop after The results are as follows:

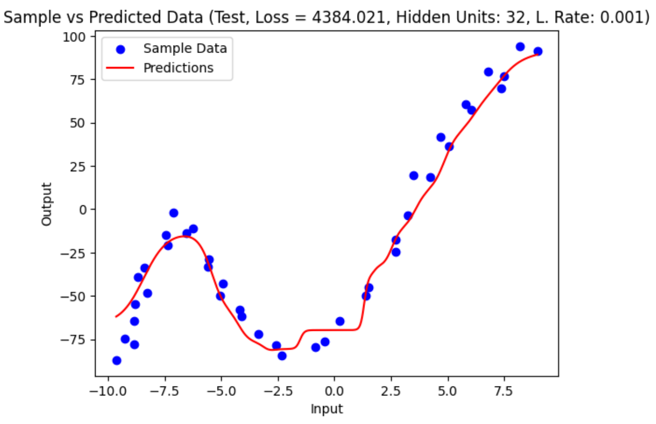
Epochs	Loss Value (SSE)
1000	31964.1798
5000	16860.8725
10000	12446.7908
20000	11957.8192
50000	11856.1658
100000	12609.2552

We can conclude that an epoch value of at least 10000 is necessary, but any value after that does not contribute too much. However, since we got a slightly better result and since the project already runs very fast, we can select 50000 as our epoch.

The effect of normalization: Normalization is typically used for adjusting the relative scale of different features in order to make better predictions. Applying normalization is crucial to get reasonable data. I will apply centering & min-max scaling only on the input data. Without normalization, I got the same result around 128 hidden units. With normalization, it was 32.

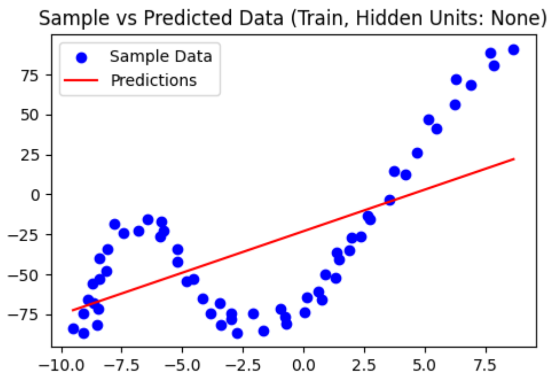
PART B) (Loss = SSE)

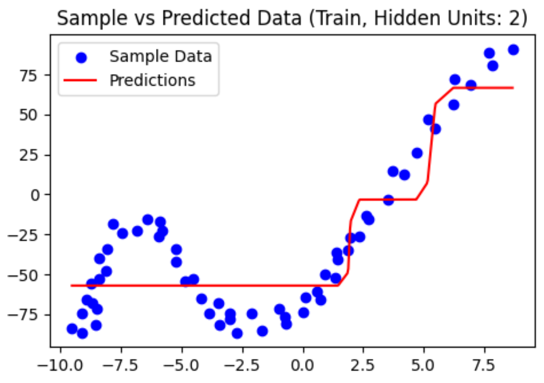


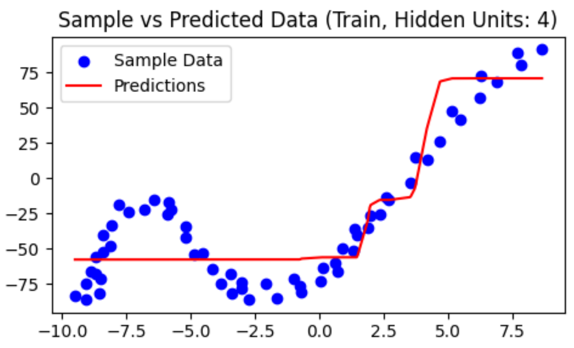


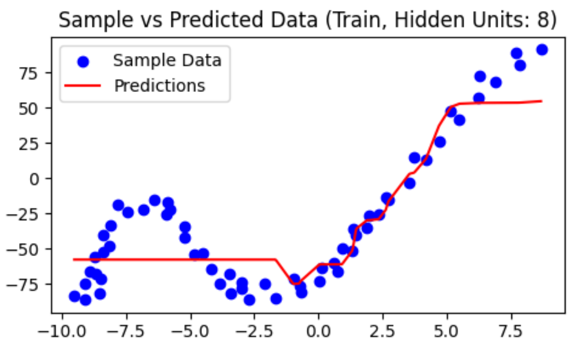
ANN used (specify the number of hidden units): **32**
Learning rate: **0.001**
Range of initial weights: **Gaussian Distribution with mean 0 and variance 1. No explicit range, but highly likely to be between -1 and 1.**
Number of epochs: **50000**
When to stop: **After epochs are finished, won't check improvement.**
Is normalization used: **Yes**
Training loss (averaged over training instances): **178.1714**
Test loss (averaged over test instances): **211.0813**

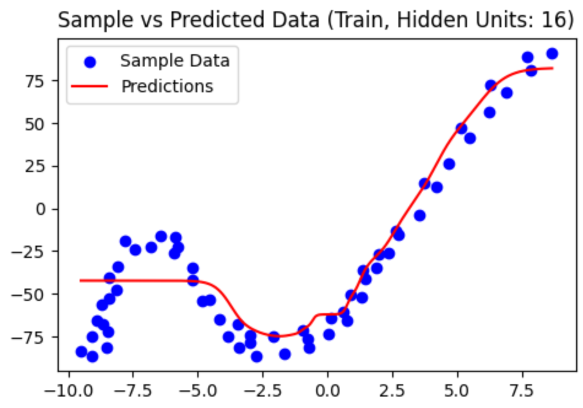
PART C)

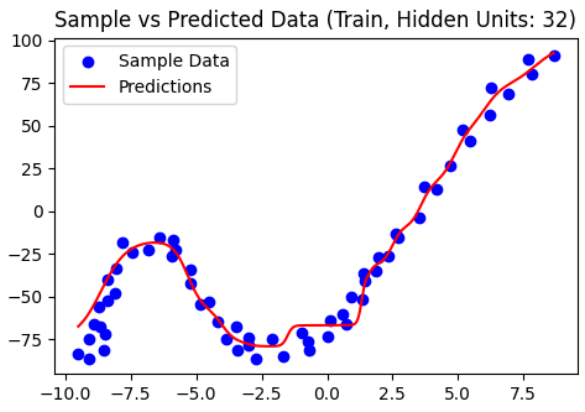












Hidden Unit Size	Averaged Train Loss	Averaged Test Loss	Standard Deviation Train	Standard Deviation Test
None	1295.2431	1492.2266	1148.8387	1280.4029
2	436.1424	566.4617	590.8646	806.7387
4	402.5433	509.2668	454.0454	705.3733
8	342.0825	386.2681	432.8899	701.5790
16	229.1732	298.9013	356.9483	460.2871
32	178.1714	211.0813	220.5303	259.9468

As it can be seen, the model is getting better at estimating when we increase the complexity. Therefore, we can conclude that a hidden layer is definitely necessary, and increasing the number of neurons actually increase the accuracy of our model. Our loss decreases after a certain point as well. However, we can also observe that the difference between 16 and 32 units is not much and therefore the model is probably coming close to a saturation point. Therefore, if we were to increase the complexity even further, we may get a worse result. Therefore, the complexity is crucial and it should be left at an optimum point in order to get the optimum result.