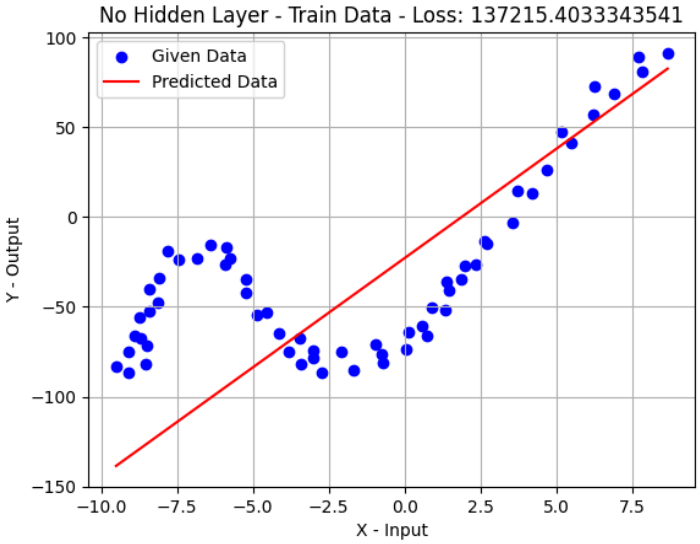
Bilkent University GE461 Introduction to Data Science | Project 3
Supervised Learning Report | Görkem Kadir Solun 22003214

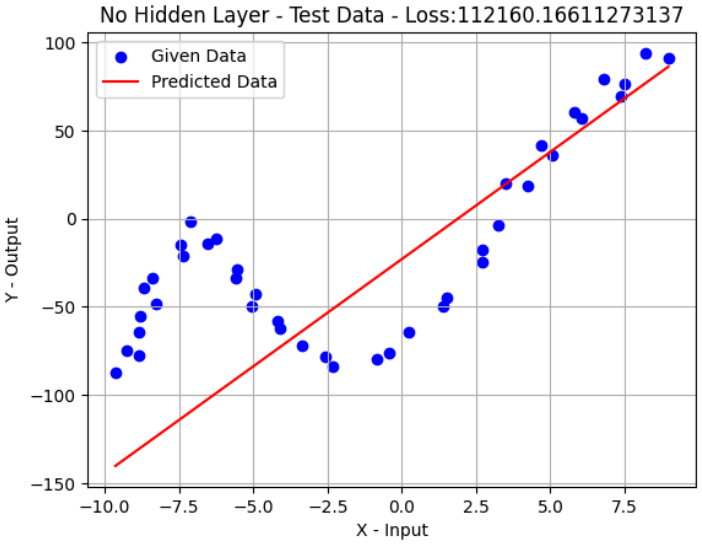
Part A

It's important to note that due to the characteristics of gradient descent and randomness, results may vary slightly between runs, but they should align closely with those reported.

Linear Regressor or ANN with Single Hidden Layer

Examining the data, it becomes apparent that a linear regressor is inadequate for this dataset, as the relationship between the input and output is not linear. Consequently, a linear regressor cannot encapsulate the non-linear relationship, and the losses are more significant than the model with a hidden layer.





ANN with Single Hidden Layer: How Hidden Unit Size Affects

To estimate the number of hidden units required, we can analyze both plots and the loss values for each model configuration using different numbers of hidden units. The values tested were 2, 4, 8, 16, 32, 64, and 128, each applied sequentially to the test dataset. The epoch was 10000, and the learning rate was 0.001. The test dataset is used. Generally, the higher the hidden unit sizes, the better results. However, I have encountered an outlier at 32. I would say the minimum is 16, but I would choose 128 as it has the lowest. You may see each result in the source code.

Hidden Units	The Sum of Squared Loss
2	26611
4	24504
8	24343
16	16288
32	18452
64	12598
128	5530

Finding a Good Value For the Learning Rate

I chose 32 as the hidden layer size and the epoch as 10000 as they are middle ground and ideal. I searched for the optimal learning rate using this configuration. As part of a standard approach, I tested common learning rates 0.01, 0.005, 0.001, 0.0005, and 0.0001, assessing their loss values and visual plots on the test dataset. The results show that the learning rate of 0.001 resulted in the lowest loss value and produced the most favorable plot. Consequently, the learning rate of 0.001 is optimal.

Learning Rate	0.01	0.005	0.001	0.0005	0.0001
The Sum of Squared Loss	54695	41597	13831	19943	30244

How to initialize the weights? A standard method for weight initialization involves randomly selecting values from a univariate Gaussian distribution with a mean of 0 and a variance of 1. For this initialization, I utilized the Numpy library.

Epochs	The sum of Squared Loss
1000	25955
5000	19660
10000	12913
50000	11894
100000	12344
500000	12053

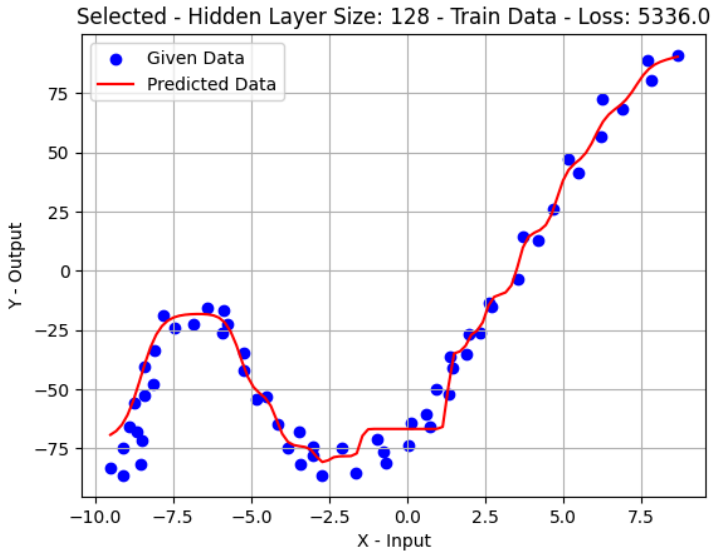
How many epochs?

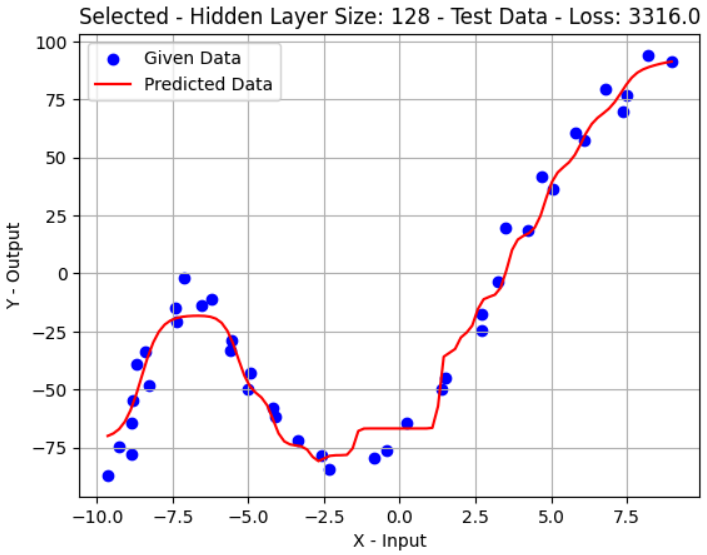
I tested epochs of 1000, 5000, 10000, 20000, 50000, and 100000. A minimum of 10000 epochs is required for practical training, with additional epochs like 50000 also a viable option. Nonetheless, given the slight improvement observed and the fast execution time, we can set the epoch count at 50000.

Does normalization affect the learning process?

Normalization is commonly employed to adjust the scale of various features, enhancing the accuracy of predictions. I applied both centering and min-max scaling exclusively to the input data. Normalization was unnecessary for this project as there may be information loss, or the results may not be impacted anyway.

Part B



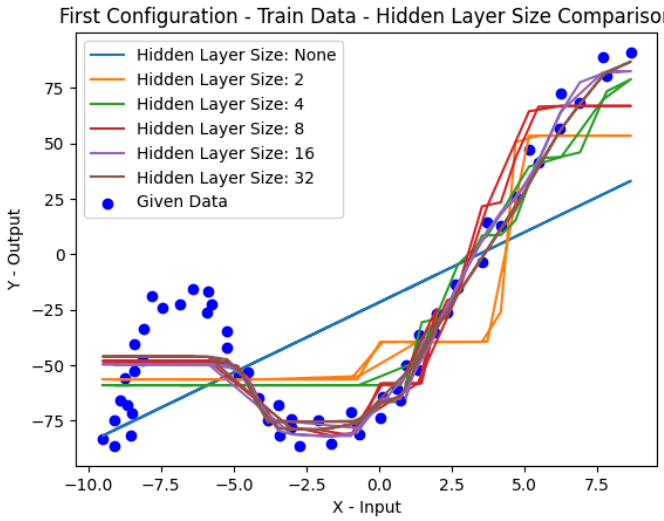


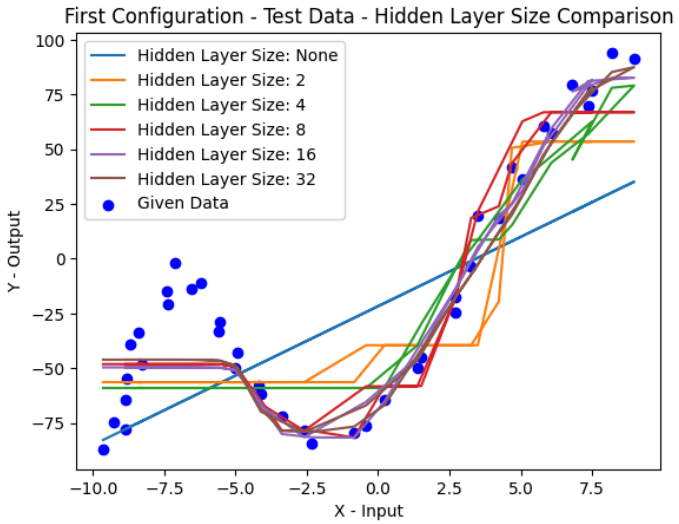
ANN used (specify the number of hidden units): **128**
Learning rate: **0.001**
Range of initial weights: **Gaussian Distribution with mean 0 and variance 1.**
Number of epochs: **50000**
When to stop: **When epochs are finished.**
Is normalization used: **No**
Training loss (averaged over training instances): **5336.431333924661**
Test loss (averaged over test instances): **3316.3432300664563**

PART C

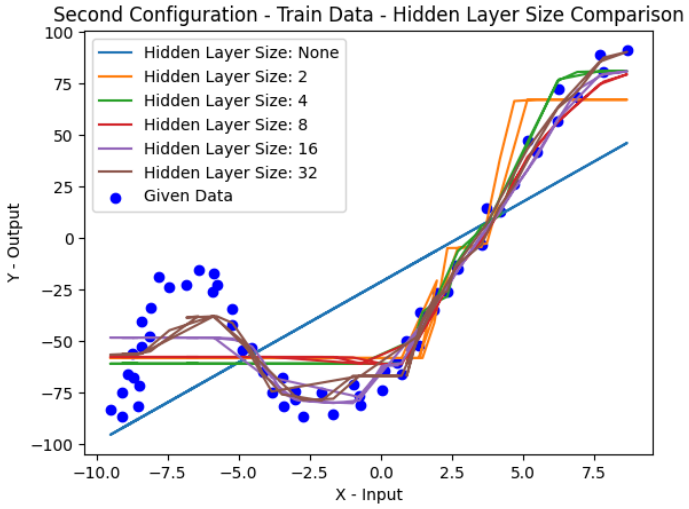
I have used two configurations to test different hidden layer sizes. The first configuration has 50000 epochs and 0.001 learning rate. The second configuration has 100000 epochs and a 0.0005 learning rate.

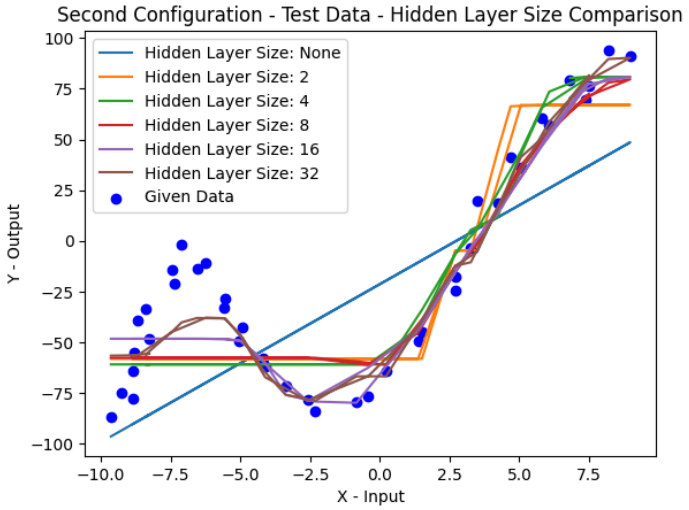
First Configuration 50000 epochs and 0.001 learning rate





Second Configuration 100000 epochs and 0.0005 learning rate





First Configuration Table

Hidden Layer Size	Train Loss Mean	Train Loss Std Dev	Test Loss Mean	Test Loss Std Dev
0	1207.292575	1157.400397	1157.400397	1348.119537
2	459.083512	532.607126	585.670112	722.005250
4	515.047736	591.629618	691.352341	794.569150
8	247.026024	354.500125	349.506325	496.218398
16	222.014327	366.947641	309.897423	477.433978
32	214.263218	370.475213	282.944298	478.789748

Second Configuration Table

Hidden Layer Size	Train Loss Mean	Traintheirss Std Dev	Test Loss Mean	Test Loss Std Dev
0	1190.304137	1026.823896	1361.391994	1173.413038
2	450.781133	491.750417	562.897014	671.358841
4	387.087287	409.108246	487.765324	641.339234
8	244.712939	356.813659	328.601070	491.883154
16	215.139058	375.122756	290.274399	473.366943
32	172.909798	307.429655	313.107402	575.586926

The models demonstrate improved accuracy as their complexity increases for both configurations (may also be for other configurations), indicating the necessity of a hidden layer and that more neurons enhance model performance. Additionally, there's a notable decrease in loss beyond a certain complexity threshold. However, the marginal gains between 16 and 32 units are minimal, suggesting that the model is nearing its saturation point. Thus, further increasing complexity might lead to diminished performance. Maintaining the complexity at an optimal level is essential to achieve the best results. Nevertheless, altering the configurations for each artificial neural network can result in more significant errors, even with a higher number of units, compared to networks with fewer hidden units. This is because complexity alone isn't the sole determining factor for neural network performance; the specific configurations of the network also play a significant role.