GE 461: Introduction to Data Science

Spring 2024

Project: Supervised Learning

The Dataset

The dataset consists of a training and a test set. The training set has **60** samples and the test set has **41** samples.

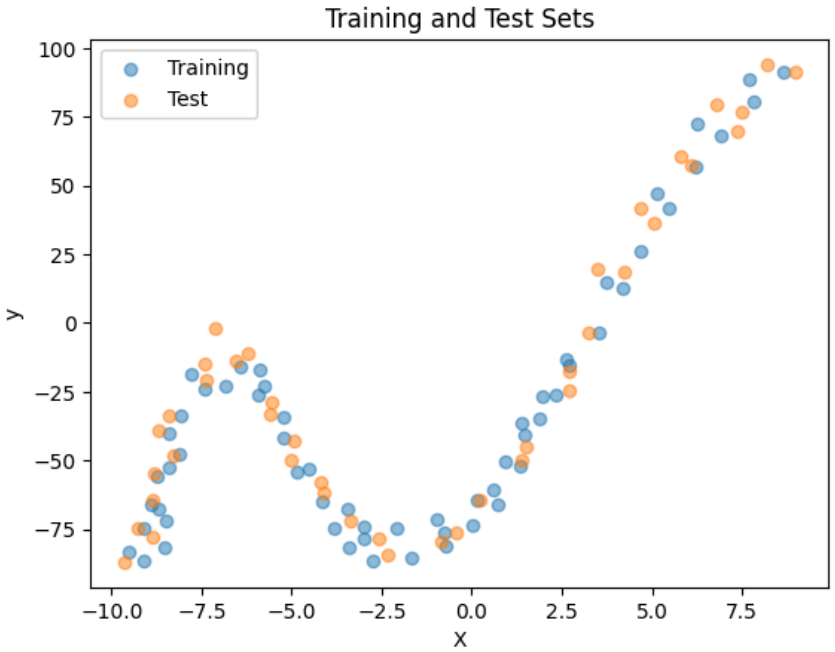

Figure 1: The training and test sets.

Figure 1 shows us that the test and training sets are very similar.

(a)

Question 1: Is it sufficient to use a linear regressor or is it necessary to use an ANN with a single hidden layer? If it is the latter, what will be the minimum number of hidden units?
Answer 1: Figure 1 indicates that a linear regressor will not be appropriate for the job. An ANN with a single hidden layer that contains 3 units at minimum will be necessary. This is because the shape of the data roughly contains three distinct linear regions.

Question 2: What is a good value for the learning rate?
Answer 2: A good value for the learning rate is a value that is as large as possible without disrupting the smooth decrease of the loss over the epochs.

Question 3: How to initialize the weights?

Answer 3: There are many different ways of initializing weights. Some can be listed as random weight initialization with normal distribution, random weight initialization with uniform distribution, and Xavier weight initialization. In this project, I used Xavier weight initialization. The worst way is to initialize all weights to zero which inhibits learning.

Question 4: How many epochs should you use? How to decide when to stop?
Answer 4: As epochs increase, the model fits the training data better and better. After some number of epochs have passed, the training loss does not decrease further. In this project, overfitting is not much of a concern. This is both because of the visualizability of the results and the simplicity of the models used (except for networks with large numbers of hidden units). For this reason, I trained all models at least until the training loss plot has largely stopped decreasing. In real world problems, training for more epochs than necessary may result in overfitting the training set. To prevent this phenomenon, the loss of the model can be checked on a validation set after every epoch and training can be stopped when the loss on the validation set stops decreasing.

Question 5: Does normalization affect the learning process **for this application**?
Answer 5: For this application, normalization seems to affect the optimal learning rate. I applied scaling to the interval [0, 1] for the normalization process. The optimal learning rate seems to be orders of magnitude larger for the normalized dataset compared to the non-normalized dataset. The network seems to be able to learn on both non-normalized and normalized datasets. However, normalization in general is desirable for training neural networks.

(b)

The configuration I found in part (a) is as follows: 1 input, 3 hidden, 1 output **layers**, 70,000 **epochs**, 0.01 **learning rate**, training and test sets **scaled** to the interval [0, 1] using the boundary information from the training set for both sets to avoid data leakage.

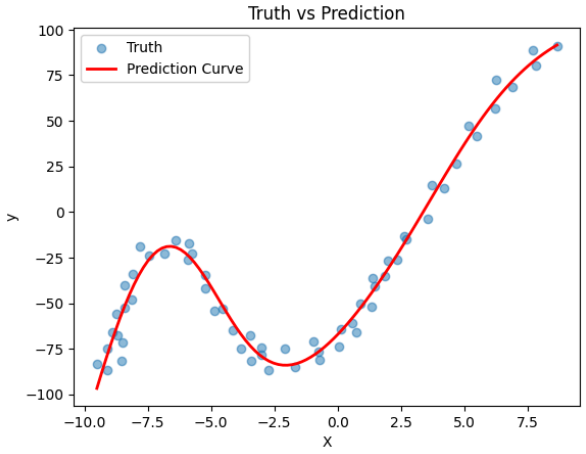

Figure 2: True data and prediction curve for the **training** set.

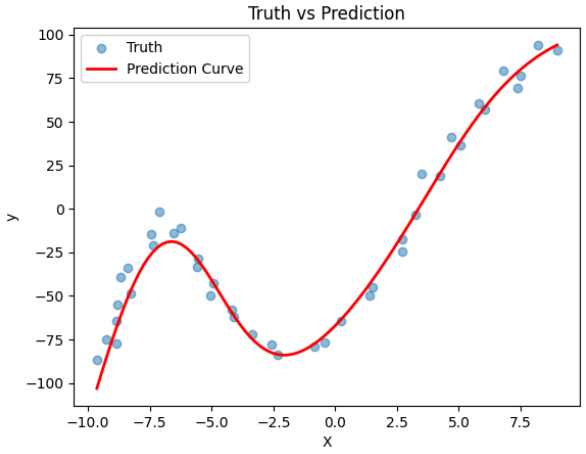

Figure 3: True data and prediction curve for the **test** set.

- **ANN used (specify the number of hidden units):** 3 hidden units.
- **Learning rate:** 0.01.
- **Range of initial weights:**[-(1 / sqrt(number of units in the previous layer)), (1 / sqrt(number of units in the previous layer))] (Xavier weight initialization)
- **Number of epochs:** 70,000.
- **When to stop:** When 70,000 epochs are complete.

- **Is normalization used:** Yes, training and test sets are scaled to the interval [0, 1] using the boundary information from the training set for both sets to avoid data leakage.
- **Training loss (averaged over training instances):** 52.26
- **Test loss (averaged over test instances):** 74.43

(c)

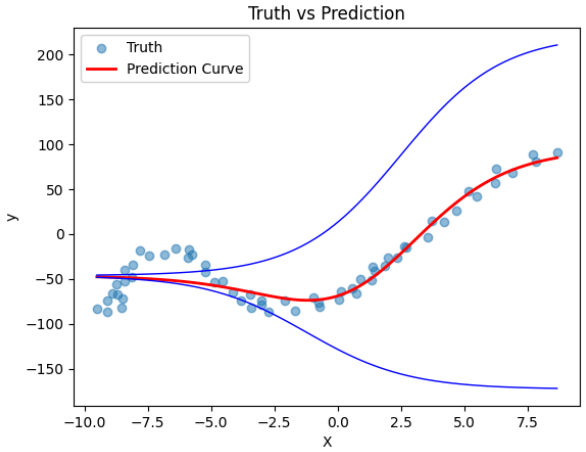

Figure 4: True data, prediction curve, and individual hidden unit curves for the training set. Network layers: [1, 2, 1]

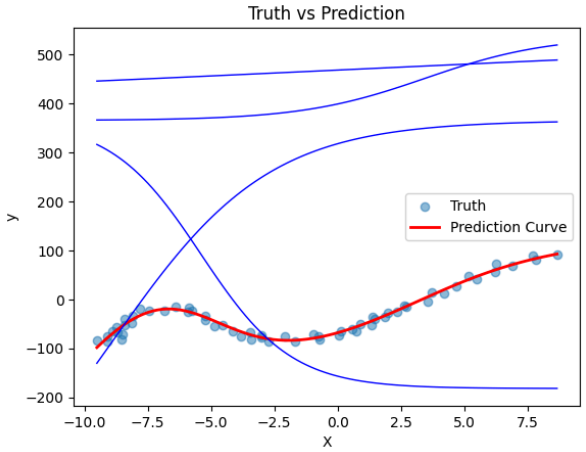

Figure 5: True data, prediction curve, and individual hidden unit curves for the training set. Network layers: [1, 4, 1]

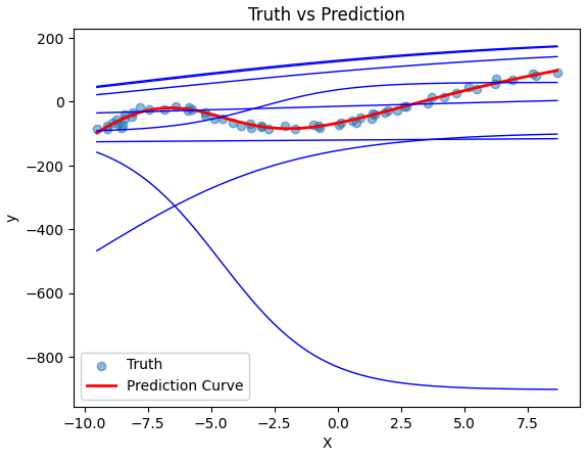

Figure 6: True data, prediction curve, and individual hidden unit curves for the training set. Network layers: [1, 8, 1]

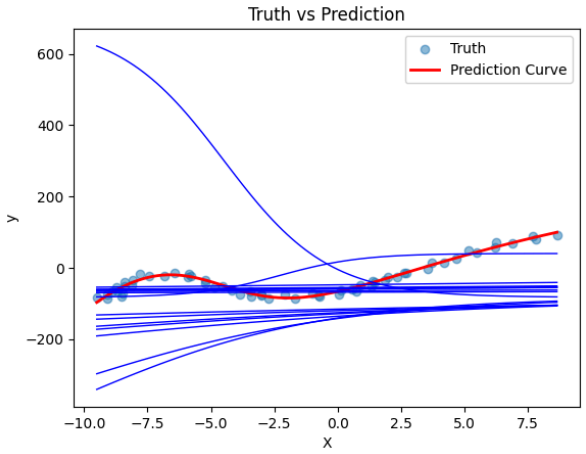

Figure 7: True data, prediction curve, and individual hidden unit curves for the training set. Network layers: [1, 16, 1]

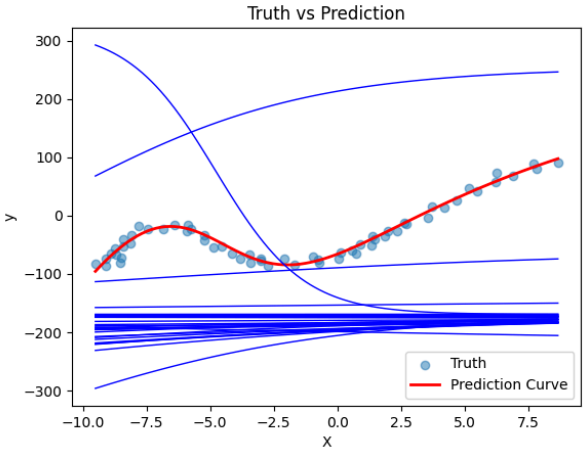

Figure 8: True data, prediction curve, and individual hidden unit curves for the training set. Network layers: [1, 32, 1]

	2 HU	4 HU	8 HU	16 HU	32 HU
Average of Training Loss	265.63	54.14	59.35	59.51	62.04
Standard Deviation of Training Loss	399.09	100.05	104.00	104.22	103.82
Average of Test Loss	338.36	76.07	83.41	86.67	82.95
Standard Deviation of Test Loss	561.35	112.90	109.70	114.47	105.34

Table 1: Set instance wise average and standard deviations of training and test sets.

In figures 4 - 8, the blue curves are calculated by setting all hidden layer weights to zero except one.

In general, in an ANN with one hidden layer, as the number of hidden units increases, the ability of the model to learn also increases. This aspect, and the ability of hidden layers to stack, are what makes the ANN architecture great for complex datasets. However, in this case, the data is not very complex and an ANN with 3 hidden units is enough to learn the distribution. As can be especially seen in figure 8, additional hidden units do not contribute much and form horizontal blue lines in contrast to the useful hidden units which form blue curves. Additionally, as observed from table 1, the performance of the ANN plateaus after 4 hidden units.

Another aspect of increasing model complexity is the increase in training difficulty. In general, the amount of optimization necessary when selecting the hyperparameters of the model increases with model complexity. Furthermore, one has to be more careful to avoid overfitting with more complex models.

Finally, another disadvantage of complex models is increased computing requirements.

To conclude, one should pick the simplest model that fits the data they are working with.