GE-461

Introduction to Data Science
Spring 2024

Prepared by
Bahadır Yüzlü

November 25, 2023

1- Data

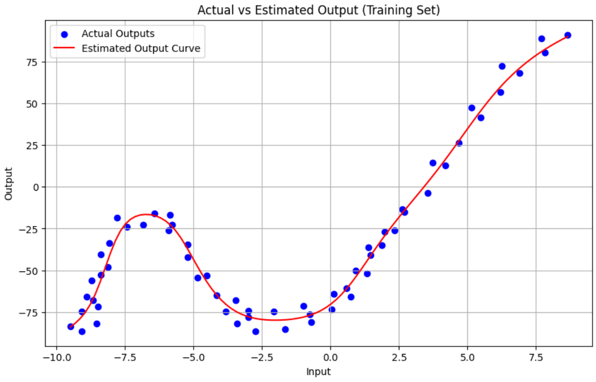
The dataset contains examples where each instance has a single input value and a corresponding single output value. The dataset is provided in two text files: train1.txt for training and test1.txt for testing. In these files, each line represents a specific example, with the first number representing the input and the next number representing the output. The training dataset contains 60 examples, and the test dataset contains 41 examples. This setup allows us to train the model on the training data and then see how well it performs on new, unseen data from the test set. By plotting this dataset, we can visually analyze how the model predicts outputs based on different input values and understand the patterns it learns.

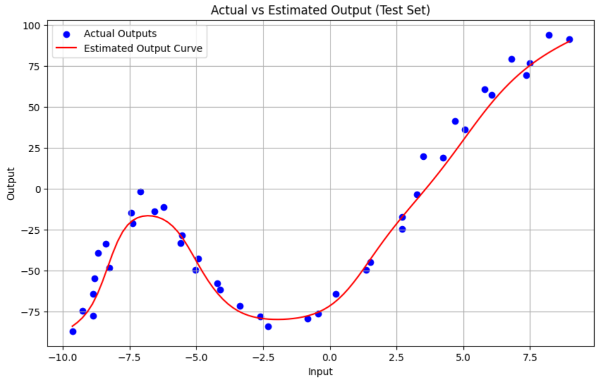
2- Model

The model is designed with specific configurations and techniques tailored for regression tasks. It utilizes the square error (mean squared error) as the loss function, which measures the average squared difference between predicted and actual values, guiding the model's learning process. The activation function employed is sigmoid, suitable for mapping outputs to a range between 0 and 1. Training is conducted using a stochastic learning algorithm, updating model parameters with small batches of randomly selected data samples to enhance efficiency and adaptability during training. The manual artificial neural network (ANN) function supports two distinct modes: linear regression for continuous output prediction and a customizable architecture featuring one hidden layer with variable node (neuron) counts. This flexibility enables experimentation with different network structures to optimize performance for specific regression tasks.

3- Configuration of the model

The model's setup was refined through multiple manual trials using gradient search. Parameters like epochs (training iterations), learning rate (rate of updating model weights), hidden node number (number of neurons in the hidden layer), and preprocessing techniques such as normalization were adjusted and tested. After this experimentation, a specific set of parameters emerged as the most effective in consistently capturing the data patterns observed during each trial. This is important given the random nature of the data and the iterative approach used, providing a stable reference point for understanding the data's behavior. The optimized parameter set reflects a thorough understanding of the data's characteristics, enabling the model to generalize well and learn effectively from different inputs and conditions.

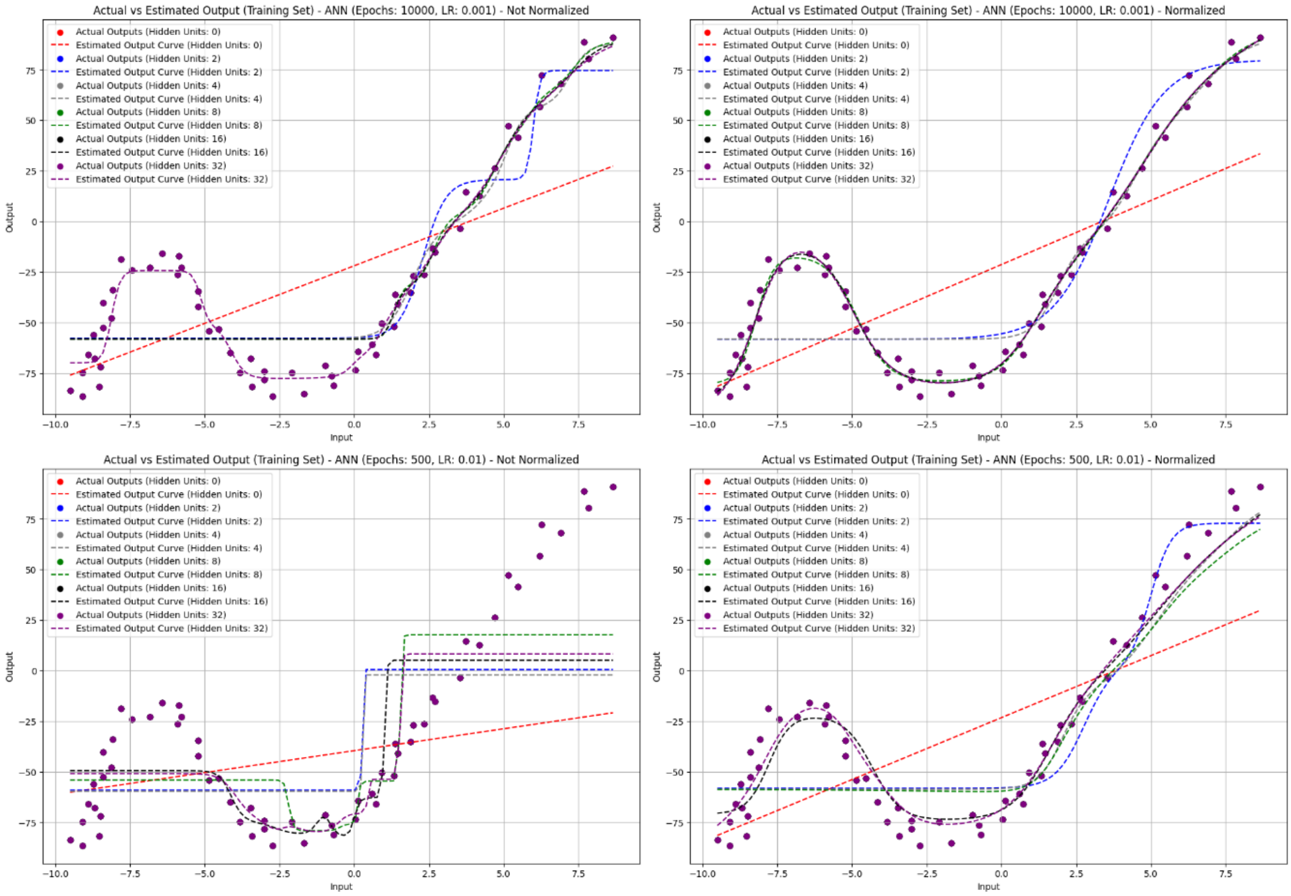




Based on the trials conducted, the linear regression model consistently underperformed and exhibited signs of underfitting in all cases. This observation is attributed to our dataset having a non-linear profile, meaning that a simple linear relationship is insufficient to adequately explain the data's complexity.

To optimize the learning rate and number of epochs, I employed a strategic approach where lower learning rates were paired with higher epochs, and higher learning rates with lower epochs. This method simplified the experimentation process by reducing the number of model configurations to evaluate. This assumption that lower learning rates are effective with larger epochs proved successful in optimizing the model's training, achieving satisfactory results while conserving computational resources.

When deciding how to normalize the data, it's important to look at how the model performs during training and testing. Without normalization, there's a risk of underfitting, especially with fewer training iterations (epochs), because the sigmoid function used in the model might not capture the full range of the data. On the other hand, using more epochs can lead to overfitting, where the model focuses too much on small details in the data instead of understanding the overall trends. Overfitting shows up as small, inconsistent changes in the model's predictions, which means it's not learning the general patterns well. Normalization is key to balancing this process, helping the model learn effectively from the data without getting too caught up in minor variations. By making sure the data is standardized, normalization ensures that the sigmoid function works well across different data ranges, improving the model's performance and ability to understand the bigger picture. All of the comments above are derived from these plots.



After visualizing the performance of different models, it's important to quantify their effectiveness by measuring the average square losses. This analysis is based on the data collected and organized in a table. The observations made from the fit plot are supported by this quantitative evaluation. Based on the analysis, the optimal configuration seems to be using 10,000 epochs, a learning rate of 0.001, normalized inputs and 32 hidden nodes. This combination of parameters yielded the best results. The following tables makes it easier to understand this approach.

Configuration	*Epoch*	*Learning Rate*	*Normalization (binary)*	*Hidden Node Number*
1	500	0.01	0	0
2	500	0.01	0	2
3	500	0.01	0	4
4	500	0.01	0	8
5	500	0.01	0	16
6	500	0.01	0	32
7	500	0.01	1	0
8	500	0.01	1	2
9	500	0.01	1	4

10	500	0.01	1	8
11	500	0.01	1	16
12	500	0.01	1	32
13	10000	0.001	0	0
14	10000	0.001	0	2
15	10000	0.001	0	4
16	10000	0.001	0	8
17	10000	0.001	0	16
18	10000	0.001	0	32
19	10000	0.001	1	0
20	10000	0.001	1	2
21	10000	0.001	1	4
22	10000	0.001	1	8
23	10000	0.001	1	16
24	10000	0.001	1	32

Configuration	*Average of Train Loss Set*	*Standard Deviation of Train Loss Set*	*Average of Test Loss Set*	*Standard Deviation of Test Loss Set*
1	1792.7588	3030.7485	2559.6488	3658.8125
2	1316.4384	1851.1410	1641.8741	2187.9536
3	1334.1657	1944.5995	1711.6964	2330.7580
4	817.2055	1162.3776	1068.7310	1407.2481
5	976.5021	1634.8326	1384.9647	1988.6248
6	834.3827	1477.0552	1188.2941	1825.1079
7	1194.0690	1064.7963	1447.9521	1295.5276
8	377.3140	432.2039	507.8182	686.8947
9	375.6623	449.4985	560.3273	692.8555
10	393.0184	459.3773	592.8751	703.6037
11	127.1337	168.9595	213.4593	241.9832
12	103.5388	141.9111	170.5670	210.7622
13	1202.2152	1117.5499	1404.9950	1279.3413
14	373.3069	431.4617	507.5428	677.5108
15	336.2459	449.8007	459.6897	694.1132
16	328.7151	453.5970	445.8829	697.7639
17	329.5334	453.2161	445.0731	698.4948
18	53.3836	92.6016	119.3703	211.7457
19	1189.8345	1023.1487	1394.1938	1214.8347
20	370.2558	443.0569	456.2470	700.5293
21	331.7748	453.3459	460.3972	696.5983
22	42.1445	62.6597	88.2807	148.1143
23	42.5484	65.8683	79.9321	133.2003
24	43.3321	68.4447	76.0493	127.8026