Question 1

Answer to Question 1

$x \rightarrow$ input $n \times f$

$y \rightarrow$ labels $n \times 1$

Hidden layer :

Output $z_1 = w_1 x + b_1$

after activation function $a_1 = g(z_1)$

Hidden layer activation function is sigmoid.

Output layer :

Output $z_2 = w_2 a_1 + b_2$

final prediction $\hat{y} = a_2 = g(z_2) = z_2$

Since this is a regression task, output layer has linear activation.

Here loss is mean squared error. $L = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

To learn the parameters

1) Start with initial guesses w_1, w_2, b_1, b_2

2) Update the weights using gradient descent by

$$w_j = w_j - \alpha \frac{\delta L}{\delta w_j}$$ Here α is learning rate.

$$b_j = b_j - \alpha \frac{\delta L}{\delta b_j}$$

3) Repeat until convergence.

Now $\frac{\delta L}{\delta w_2} = \frac{\delta L}{\delta a_2} \cdot \frac{\delta a_2}{\delta z_2} \cdot \frac{\delta z_2}{\delta w_2}$

$= \frac{\delta}{\delta a_2}\left[\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\right] \underbrace{\frac{\delta a_2}{\delta z_2}}_{\text{Identity}} \cdot \frac{\delta z_2}{\delta w_2}$

$= \frac{1}{n} \cdot 2 \cdot \sum_{i=1}^{n}(y_i - \hat{y}_i)(-1) \cdot a_2^T$ (When providing a vectorial solution)

$= \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)\, a_2^T$

Now. $\frac{\delta L}{\delta b_2} = \frac{\delta L}{\delta a_2} \cdot \frac{\delta a_2}{\delta z_2} \cdot \frac{\delta z_2}{\delta b_2}$

$= \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i) \cdot 1$ [since $z_2 = w_2 a_1 + b_2$]

∴ Update rule:

$$w_2 = w_2 - \alpha\left[\frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)\, a_2^T\right]$$

$$b_2 = b_2 - \alpha\left[\frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)\right]$$

Now, $\frac{\delta L}{\delta w_1} = \frac{\delta L}{\delta a_2} \cdot \frac{\delta a_2}{\delta z_2} \cdot \frac{\delta z_2}{\delta a_1} \cdot \frac{\delta a_1}{\delta z_1} \cdot \frac{\delta z_1}{\delta w_1}$

$= \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i) \cdot w_2\, g(z_1)(1 - g(z_1))\, x$

Rearranging for vectorization

$\frac{\delta L}{\delta w_1} = w_2^T \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)\, g(z_1)(1 - g(z_1))\, x$

$$\frac{\delta z_1}{\delta b_1} = 1$$

$$\text{So, } \frac{\delta L}{\delta b_1} = \omega_2^T \frac{2}{n} \sum_{i=1}^{n} (\hat{y_i} - y_i)\, g(z_1)(1 - g(z_1))$$

Update rule

$$\omega_1 = \omega_1 - \alpha \left[\omega_2^T \frac{2}{n} \sum_{i=1}^{n} (\hat{y_i} - y_i)\, g(z_1)(1 - g(z_1))x \right]$$

$$b_1 = b_1 - \alpha \left[\omega_2^T \frac{2}{n} \sum_{i=1}^{n} (\hat{y_i} - y_i)\, g(z_1)(1 - g(z_1)) \right]$$

For regression, output is a continuous value. MSE is suitable in this case as a loss function, and we use linear activation in the output layer. However, for binary classification output is either 0 or 1. Here, we use log loss as the loss function and sigmoid function as an activation function in the output layer. Log loss has some advantage over MSE when the task is binary classification. It penalizes misclassifications strongly compared to MSE. For example, when the actual output is 1 but predicted output is close to 0, from the MSE formula

$$MSE = \frac{1}{n} \Sigma \underbrace{\left(y - \hat{y} \right)^2}_{\text{The square of the difference between actual and predicted}}$$

MSE is close to 1

But from log loss formula

$$L = -\left[(1-y) \cdot \log(1-\hat{y}) + y \cdot \log(\hat{y}) \right]$$

Log loss tends to infinity. So, its clear that log loss penalizes misclassifications strongly compared to MSE.

[Question 2]

1. What is the activation function that you will choose for the output layer? Justify your answer briefly.

Ans: I will choose linear activation function for the output layer.

This is a regression task. For regression task we usually use linear activation function. Output of a regression is a continuous value. Linear activation doesn't change weighted sum of inputs and returns the value directly. So, it is suitable for regression.

2. How many neurons should there be in the output layer? Why?

Ans: There should be one neuron in the output layer.

Here, the output is a continuous value. So, only one neuron is sufficient in this case.

3. Report the average MSE loss and the accuracy.

For,

number of neurons in hidden layer = 3

iteration number = 100 (I report loss at this iteration because the network converged as can be seen from the plot in the answer to the next question)

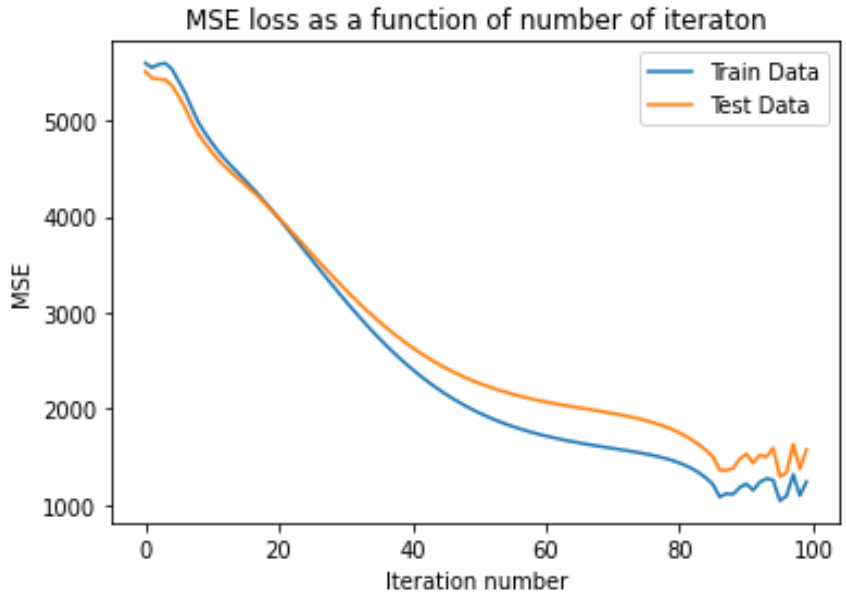
leaning rate = .001

Mean Squared Error on training data: 1241.29

Mean Squared Error on test data: 1574.40

4. Plot the loss and accuracy as a function of the number of iterations.

MSE loss is plotted as a function of number of iterations below:



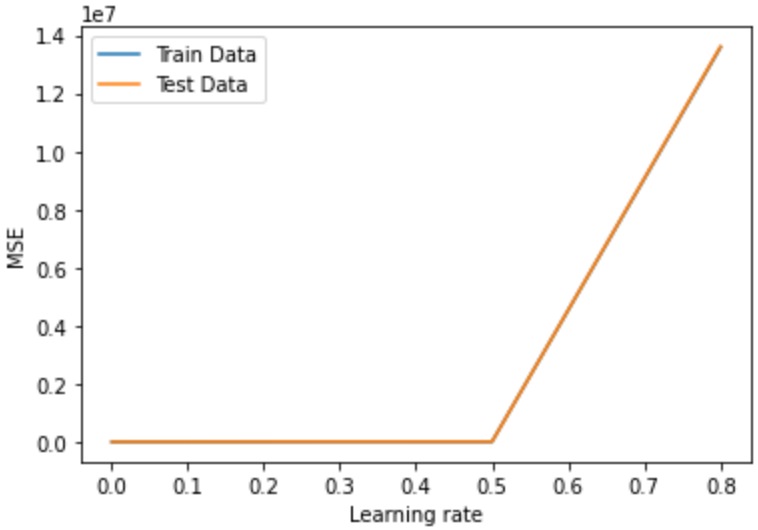
5. What is the effect of the learning rate on the training process? Vary the learning rate to be between 0.001 and 1.0 and plot the resulting accuracy as a function of learning rate.

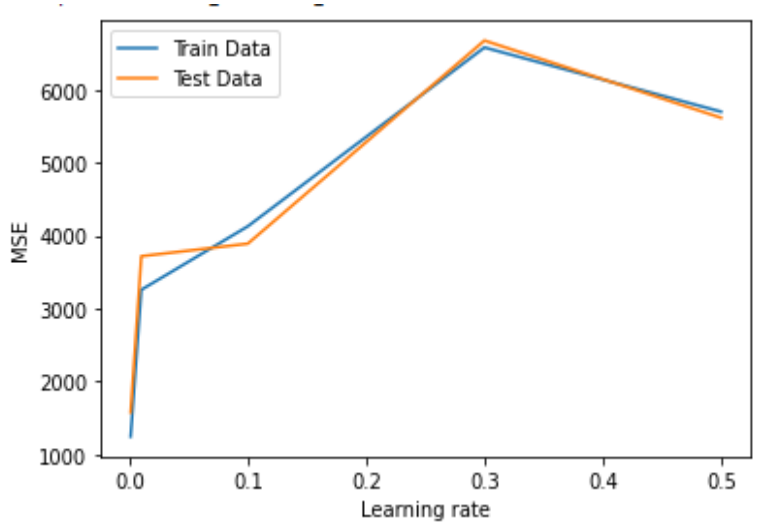
Answer: I have chosen the following learning rate to observe the effect of learning rate on the network: 0.001, 0.01, 0.1, 0.3, 0.5, 0.7, 0.9, 1

Here, I report loss values after 100 iterations, number of neurons in hidden layer = 3

Learning rate	MSE loss on training data	MSE loss on test data
.001	1241.29	1574.40
.01	3256.91	3718.22
.1	4128.52	3890.20
.3	6583.62	6678.61
.5	5702.20	5618.67
.8	13584143.65	13612060.59
1	Math range error	Math range error

MSE loss is lowest for both training and test data for learning rate = .001. As the learning rate increased, MSE loss also increased. Loss was very big for learning rate = .8 and for learning rate = 1, I got a math range error. For, high learning rates the learning process goes back and forth and doesn't find the minima.





6. What is the effect of the number of neurons in the hidden layer? To answer this question,

you will need to consider and answer the following:

a. You will need to vary the number of neurons from 1 to 10. Does the update rule need to be changed/derived again? Why or why not?

The update rule doesn't need to be changed.

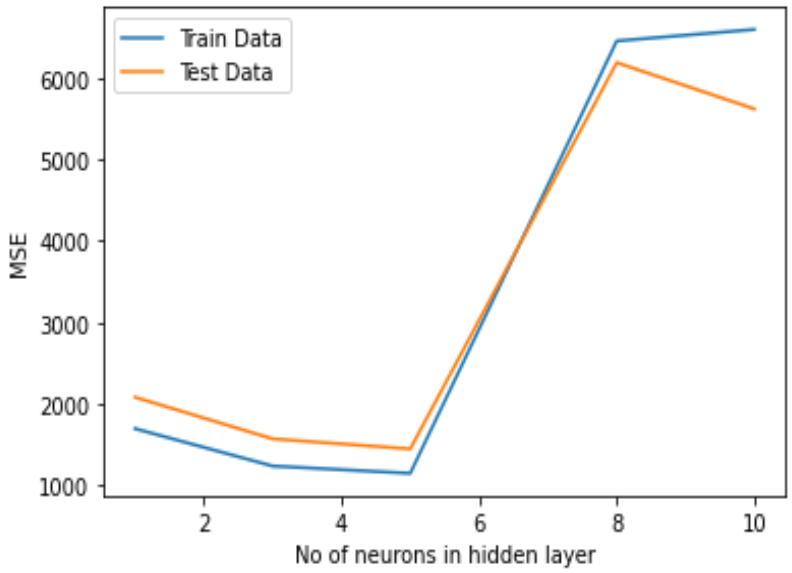
Update rule remains same for all the weights associated with a single layer. Varying number of neurons means change in number of weights, which doesn't have any influence on update rule.

b. Report your observations by reporting the final loss and plotting the true labels and

your predicted labels, along with a brief (2-3 lines) description.

Answer: Here, I report loss values after 100 iterations, learning rate = .001

No of neurons	MSE loss on training data	MSE loss on test data
1	1700	2085.01
3	1241.29	1574.4
5	1152.41	1451.24
8	6451.3	6185.53
10	6592.1	6301.8



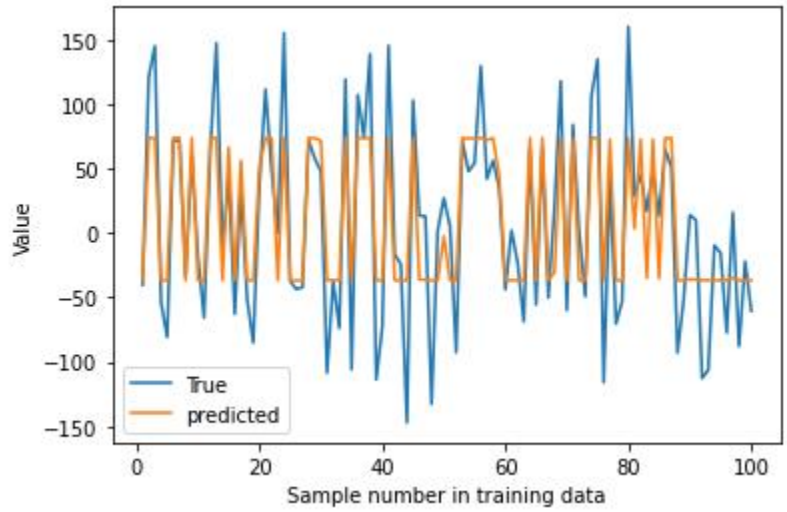
Brief 2-3 line description:

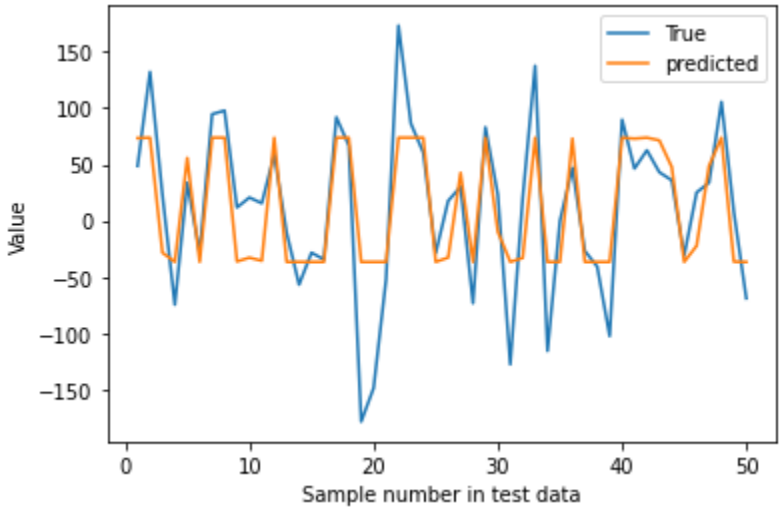
MSE decreased initially when no of neurons was increased. This happened from no of neurons 1 to 5. Then loss started to increase a lot. Increasing network parameters can be attributed to this increase.

true labels and predicted labels plots are in the next 5 pages

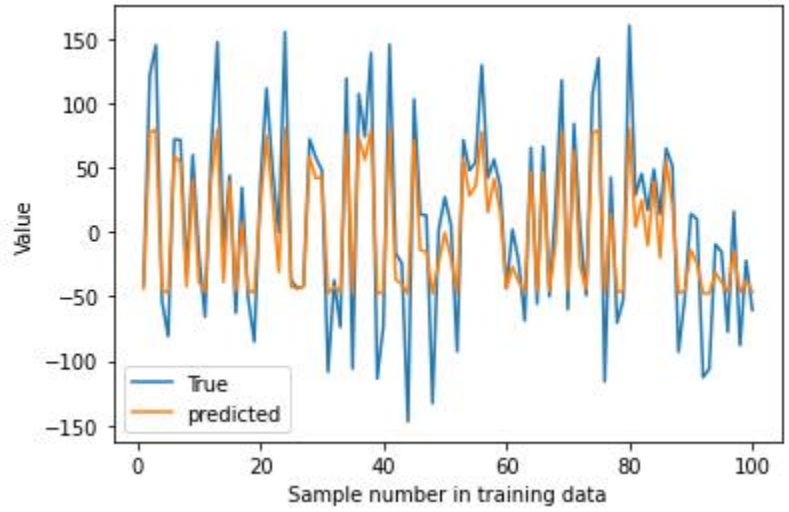
Plotting true labels and predicted labels :

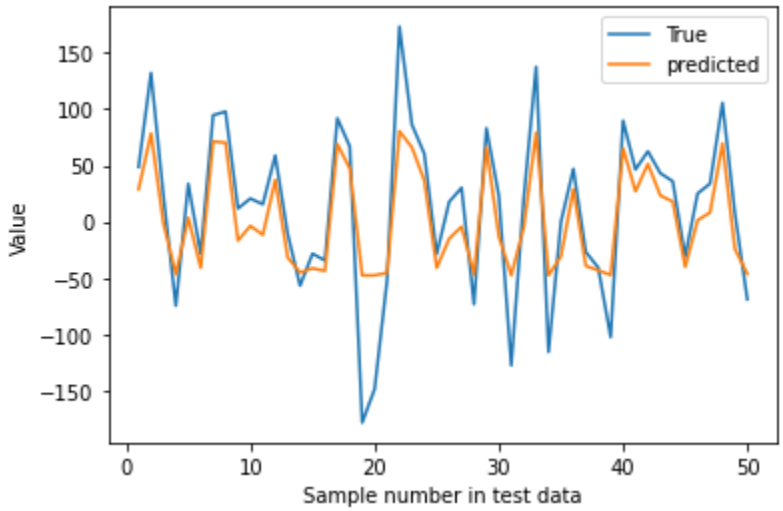
For no of neurons = 1



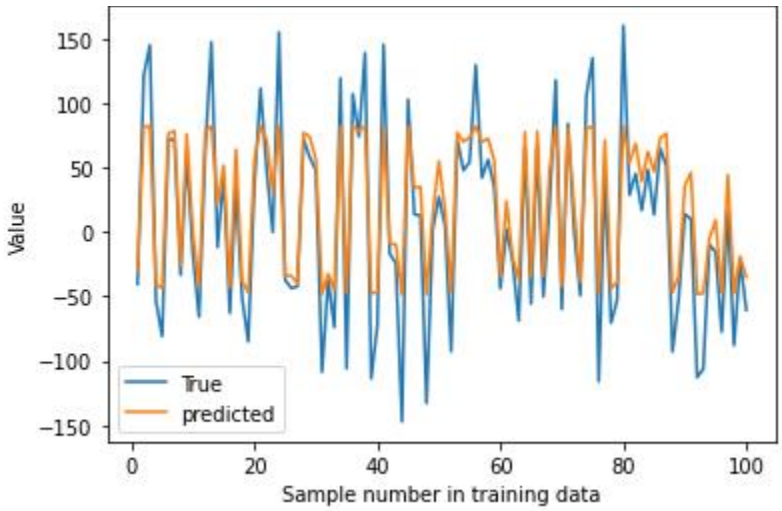


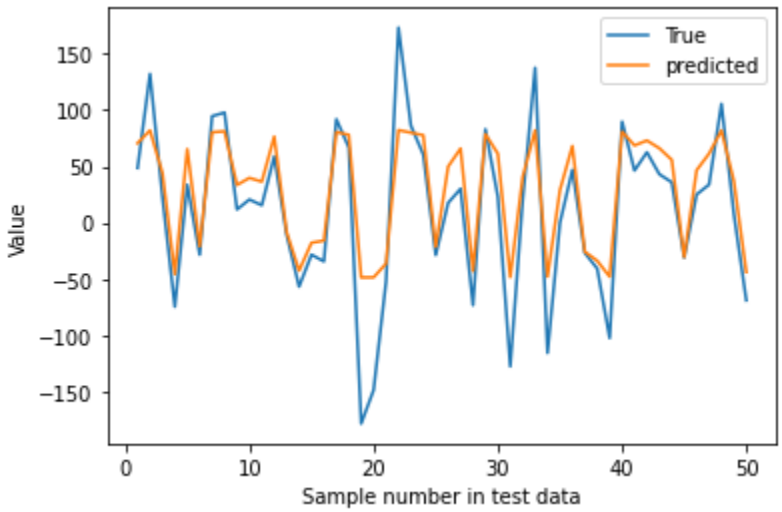
For no of neurons = 3



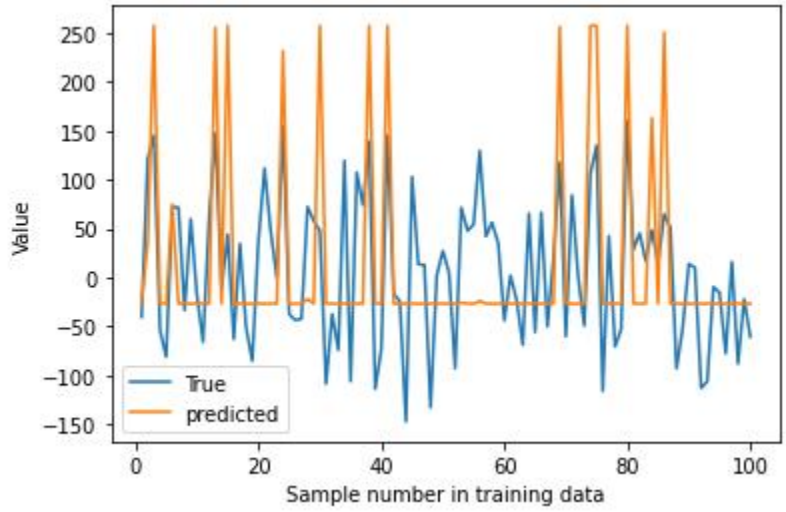


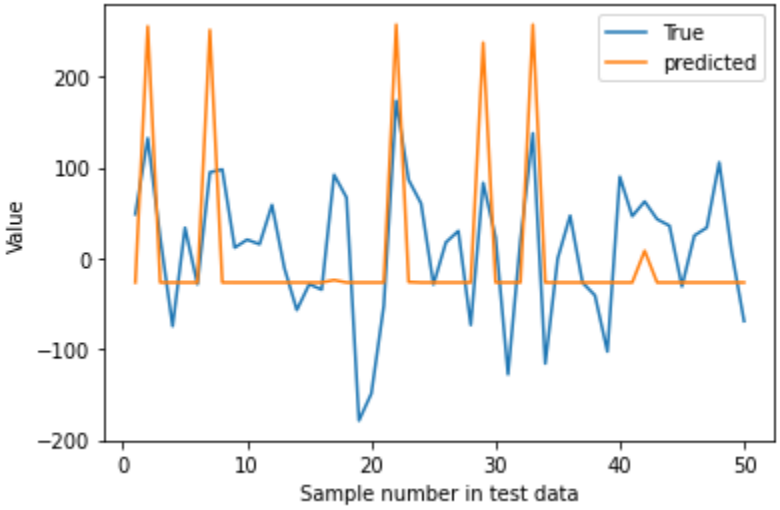
For no of neurons = 5



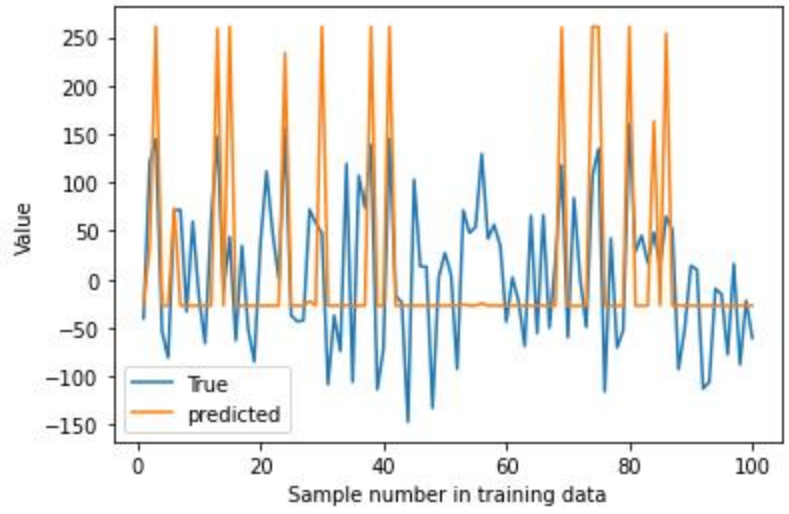


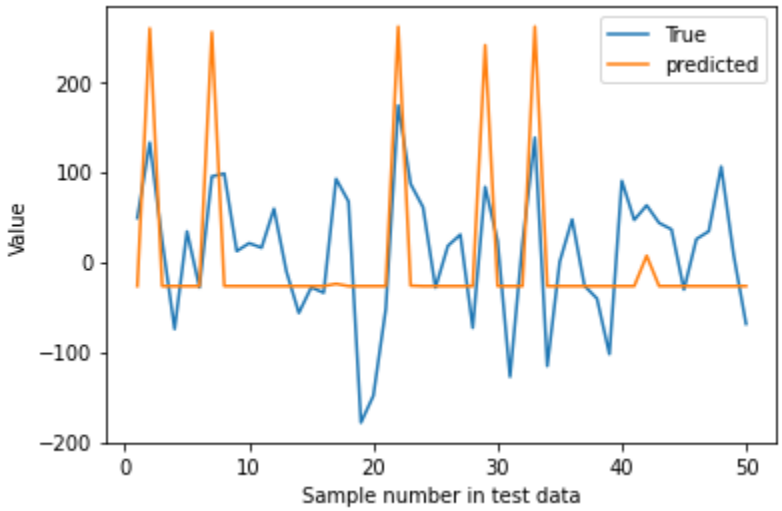
For no of neurons = 8





For no of neurons = 10





7. What is the effect of the activation functions in the network? Explore two different activation functions other than sigmoid such as tanh, linear, or ReLU.

a. Will you need to change the update rule?

Yes, I will need to change the update rule.

b. What is the change that you need to make to achieve this experiment?

I have tested tanh and linear activation function.

Update rule for tanh:

dLw1 = (2/n)*sum_a2_minus_y* g_dot * np.matmul(w2.T, x_train[i][:,None].T)

dLb1 = (2/n)*sum_a2_minus_y* g_dot * w2.T

dLw2 = (2/n)*sum_a2_minus_y*a1.T

dLb2 = (2/n)*sum_a2_minus_y

w1 = w1 - alpha*dLw1

b1 = b1 - alpha*dLb1

w2 = w2 - alpha*dLw2

b2 = b2 - alpha*dLb2

Update rule for linear:

dLw2 = (2/n)*sum_a2_minus_y*a1.T

dLb2 = (2/n)*sum_a2_minus_y

dLw1 = (2/n)*sum_a2_minus_y* g_dot * np.matmul(w2.T, x_train[i][:,None].T)

dLb1 = (2/n)*sum_a2_minus_y* g_dot * w2.T

w1 = w1 - alpha*dLw1

b1 = b1 - alpha*dLb1

w2 = w2 - alpha*dLw2

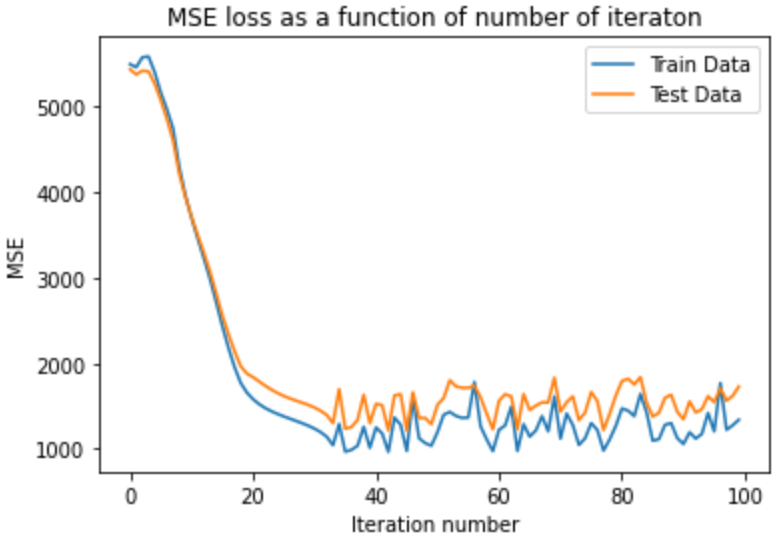
b2 = b2 - alpha*dLb2

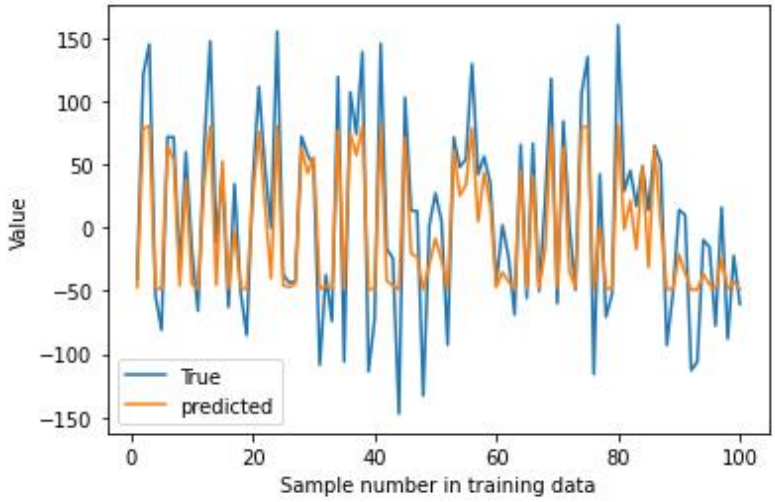
c. Report your observations by reporting the final loss and plotting the true labels and your predicted labels, along with a brief (2-3 lines) description.

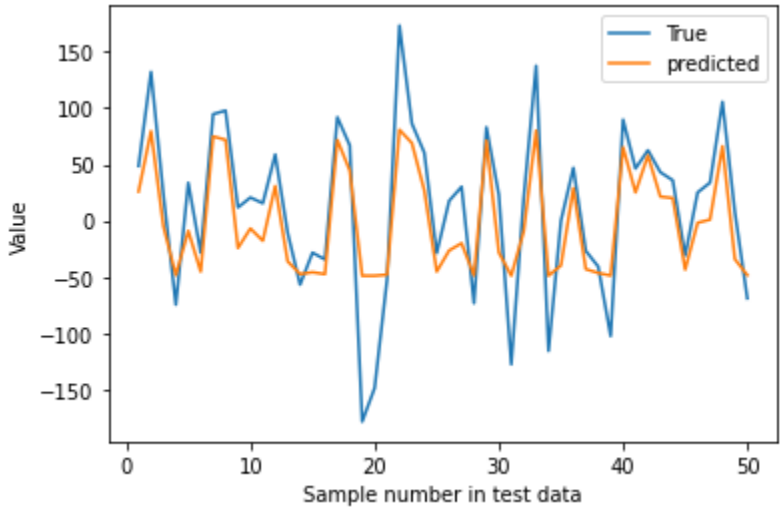
Tanh:

Mean Squared Error on training data: 1339.63

Mean Squared Error on test data: 1728.92


MSE loss as a function of number of iteraton
Train Data
Test Data
MSE
5000
4000
3000
2000
1000
0
20
40
60
80
100
Iteration number


Value
150
100
50
0
−50
−100
−150
True
predicted
0
20
40
60
80
100
Sample number in training data


Value
150
100
50
0
−50
−100
−150
True
predicted
0
10
20
30
40
50
Sample number in test data

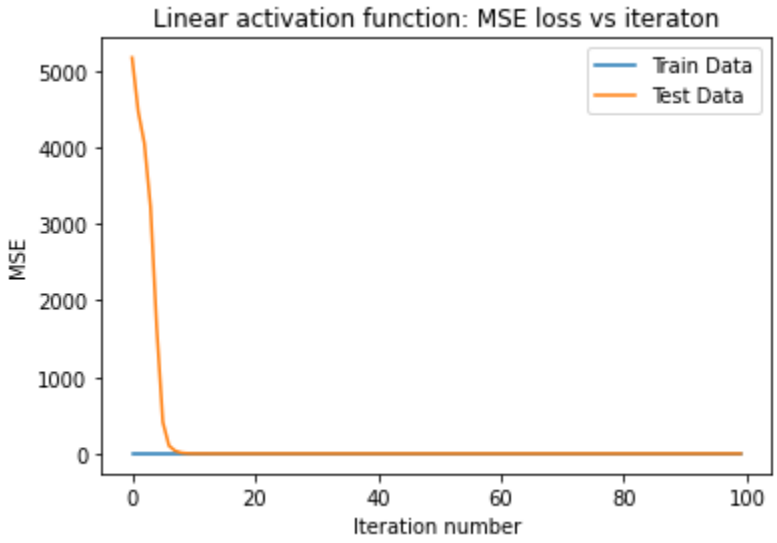
2-3 line observation:

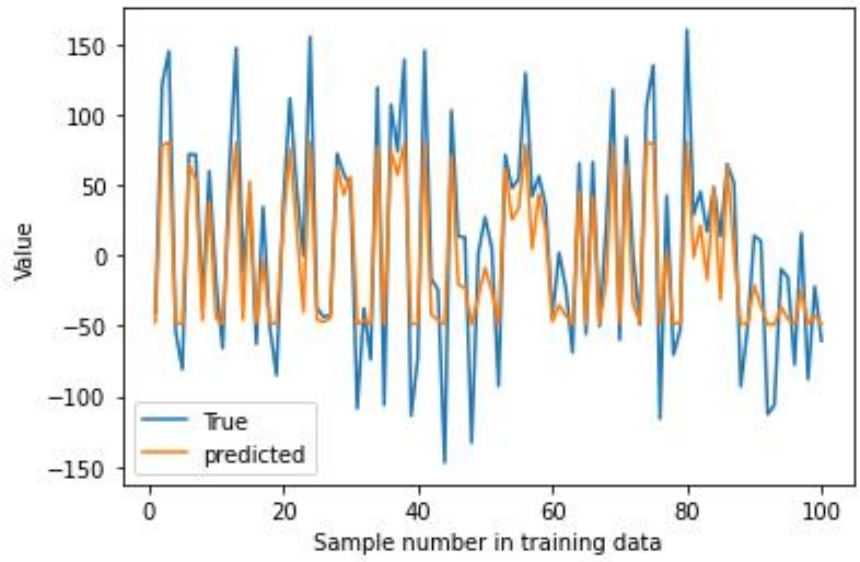
Compared to sigmoid, tanh converged much earlier. The MSE loss is also comparable to sigmoid.

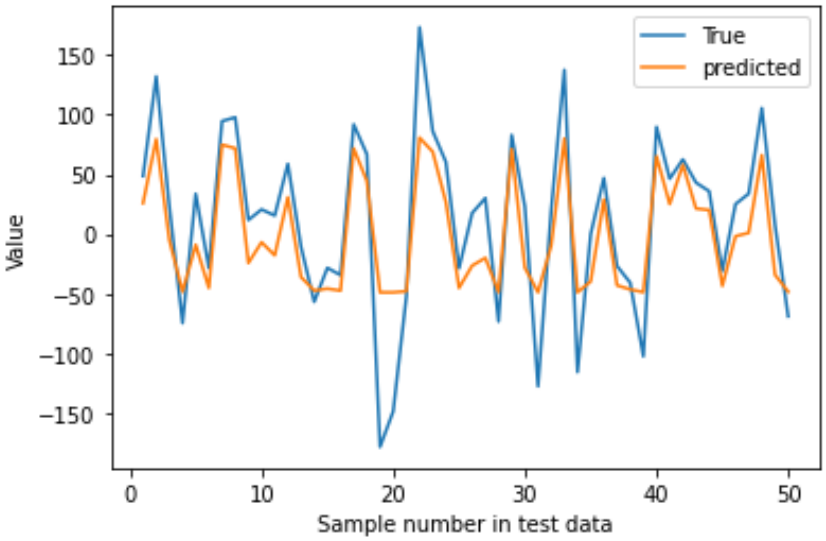
Linear:

Mean Squared Error on training data: 0.0

Mean Squared Error on test data: 7.11397765e-26







2-3 line observation:

MSE loss was 0 in both training and test data. However, from prediction plots we can see that prediction don’t match the true values which is contradictory