BNC: Homework 2

Zayid Oyelami

July 25, 2019

1 Problem 1: Linear Classifier with Perceptron

1.1 Background

The perceptron serves the purpose of linearly discriminating data as a means of classifying the data into groups.This is a basic building block of Neural Networks.

1.2 Method

Linear and non-linear datasets were randomly generated with Matlab to test the efficacy of the perceptron algorithm of distinguishing data. The perceptron algorithm was created by a linear combination of d feature vectors(x_d) and d+1 weights (w_d). The dot product of the weight vector and feature vector determine the class a data point belongs to, and is mathematically represented as the following:

$$y(x_1, ..., x_d) = \begin{cases} 1 & (w_0x_0 + ... + w_dx_d > 0) \\ 0 & (otherwise) \end{cases}$$

The weight vector was initially estimated, but was updated after several iterations. The relationship between error (or the misclassification) of the perceptron and the number of iterations was also established by the by identifying the number of times outputs of the feature vector matched with the targeted.

1.3 Results

1.3.1 Linear Data

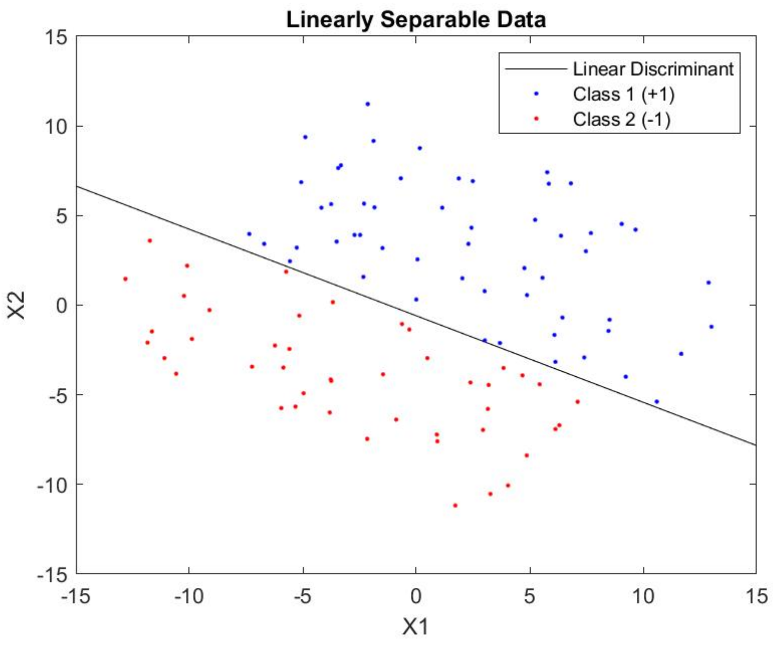

Figure 1: Linearly separable data.

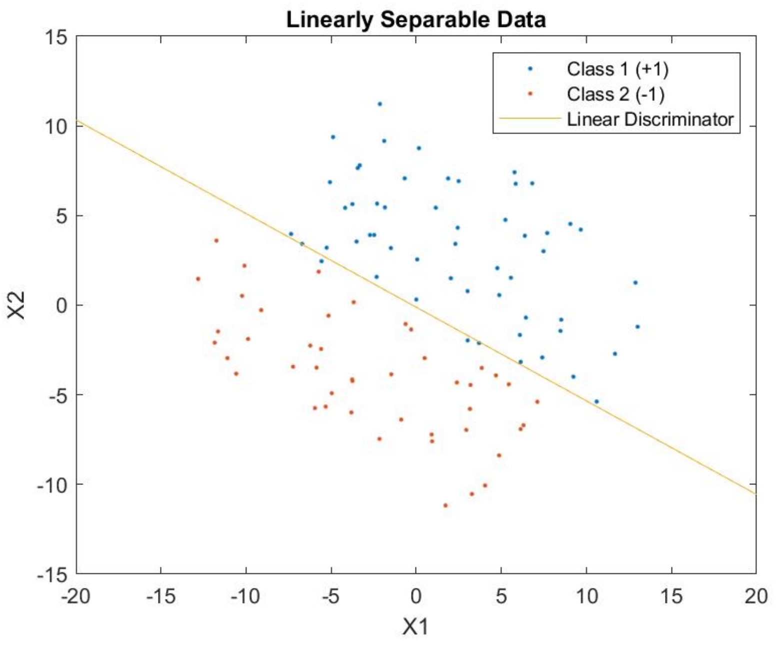

Figure 2: The application of a perceptron on a separable data set after just a 10 iterations.

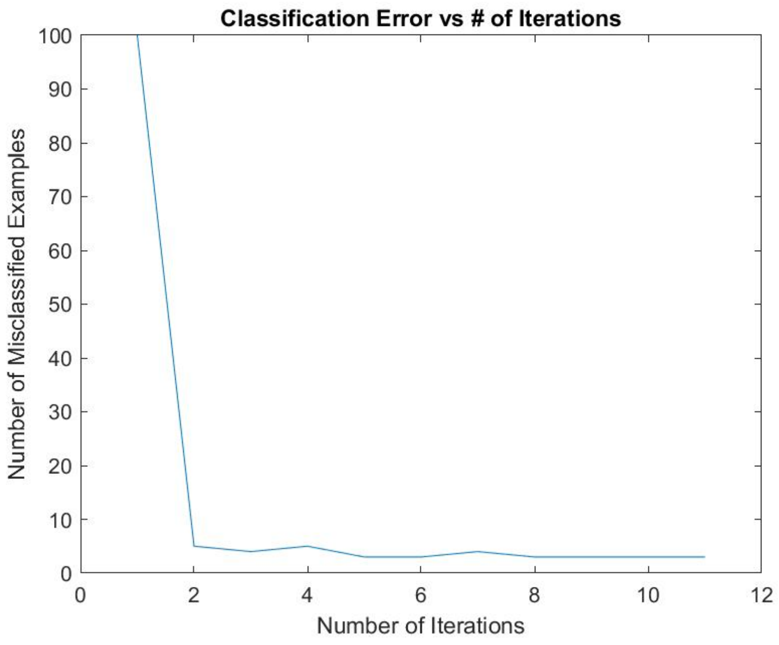

Figure 3: Mis-classification error after 10 iterations.

Error = 30%

$$w = \begin{bmatrix} 0.43 \\ 1.9437 \\ 3.7214 \end{bmatrix} \tag{1}$$

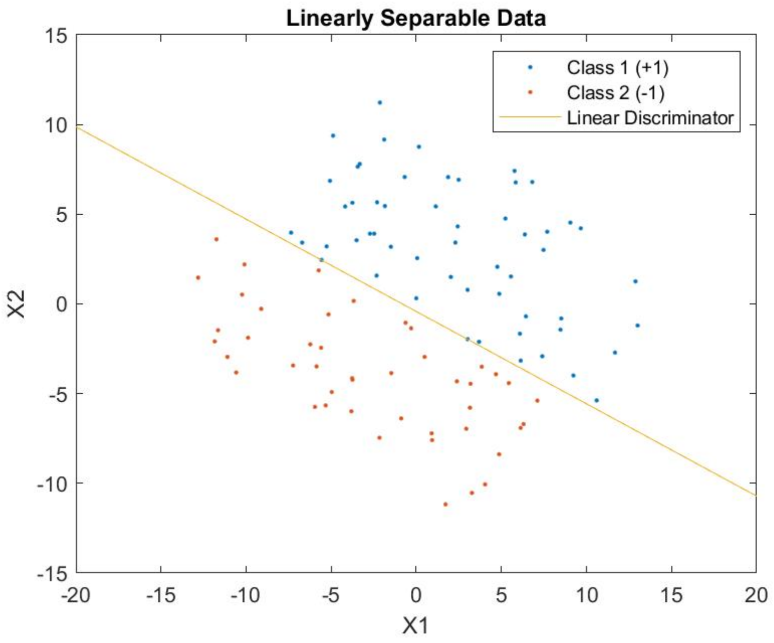

Figure 4: The application of a perceptron on a separable data set after 100 iterations.

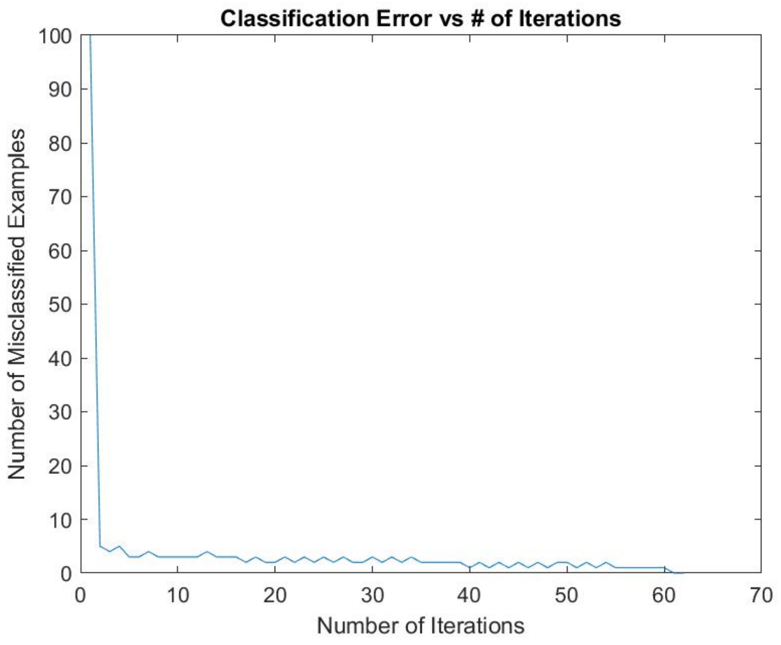

Figure 5: Mis-classification error after 100 iterations.

Error = 0%

$$w = \begin{bmatrix} 1.45 \\ 1.7626 \\ 3.4255 \end{bmatrix} \tag{2}$$

1.3.2 Non-linear Data

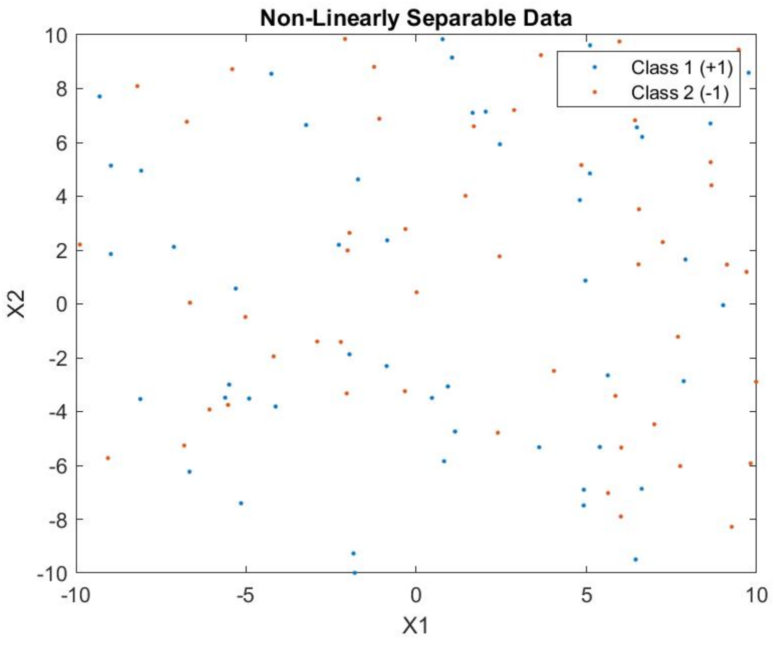

Figure 6: Non-linearly separable data.

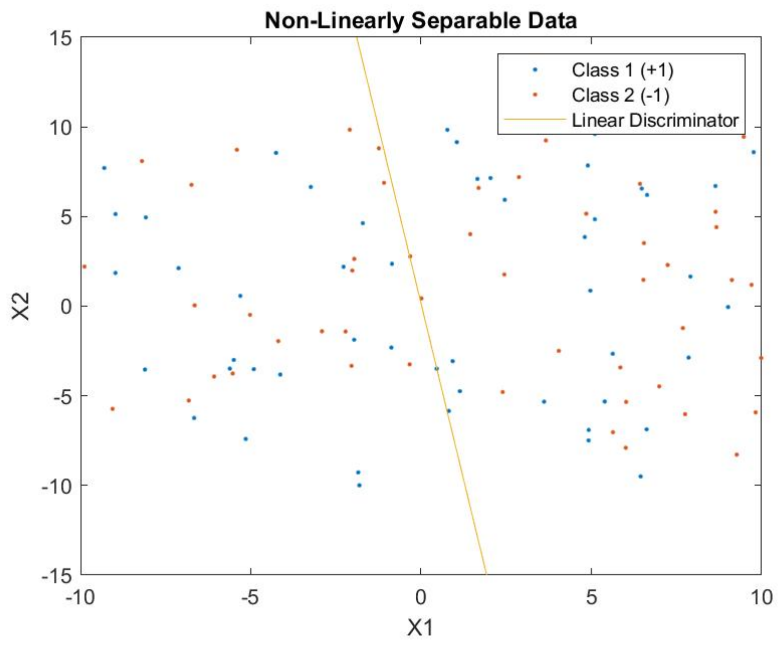

Figure 7: The application of a perceptron on a separable data set after 10 iterations.

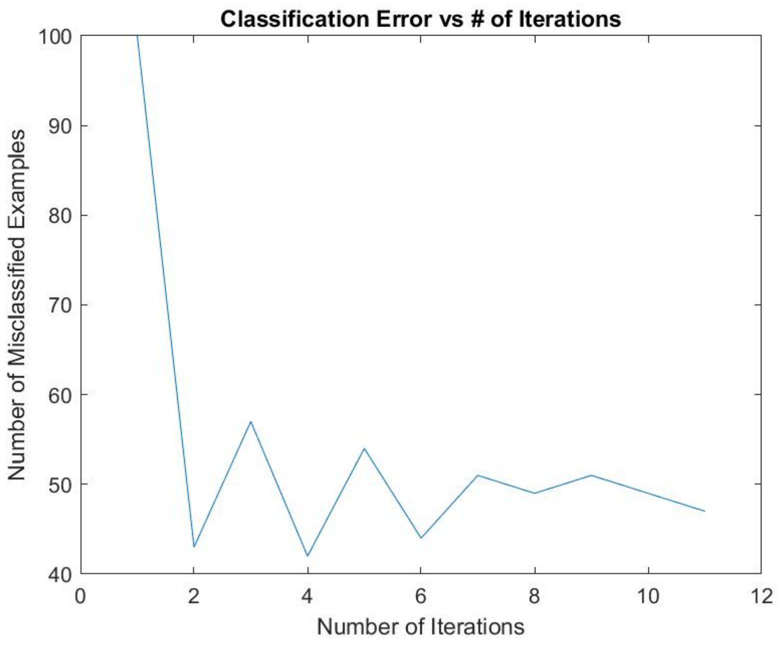

Figure 8: Mis-classification error after 10 iterations.

Error = 47%

$$w = \begin{bmatrix} 0.2 \\ -1.0652 \\ -0.1355 \end{bmatrix} \tag{3}$$

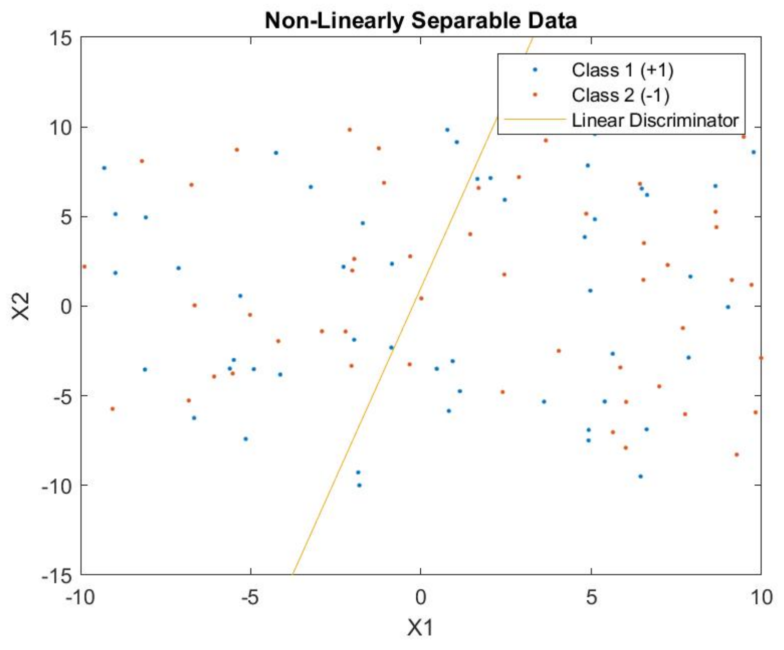

Figure 9: The application of a perceptron on a separable data set after 100 iterations.

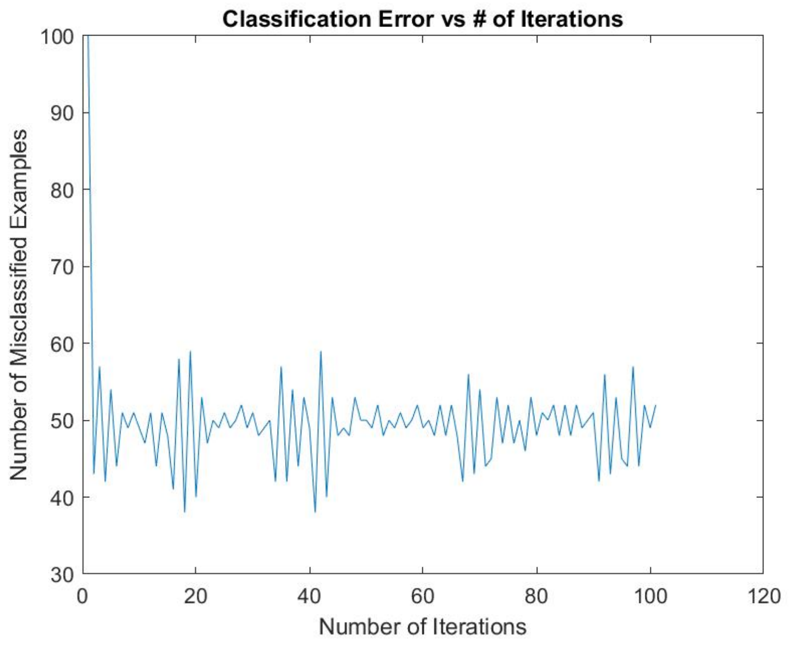

Figure 10: Mis-classification error after 100 iterations.

Error = 52%

$$w = \begin{bmatrix} 0.15 \\ 0.6132 \\ -0.1445 \end{bmatrix} \tag{4}$$

1.4 Discussions

The perceptron works perfectly on the linear separable data as can be imagined as it is a linear separator when the number of iterations is sufficient. This is because with the increasing number of training examples, the error within the discriminator is being minimized. It does not work as well on data that is non-linear as the amount of iterations does not lead to a better result and actually oscillates as seen in both mis-classification plots.

2 Problem 2: Linear Classifier with Least Mean Squares

2.1 Background

Similar to previous problem except using the method of Least Mean Squares squared difference between the target and the output. Least Mean Squares is another linear discriminating technique used to classify data.

2.2 Method

Kept the same linear and non-linear datasets from Problem 1, but now attempted to minimize the squared difference between the target and the output unlike the perceptron from Problem 1. Superimposed the resulting weight vector on the same axes as the Batch perceptron. Furthermore, plotted the number of mis-classified examples using the same technique of matching targets to predicted outputs as last time.

2.3 Results

2.3.1 Linear Data

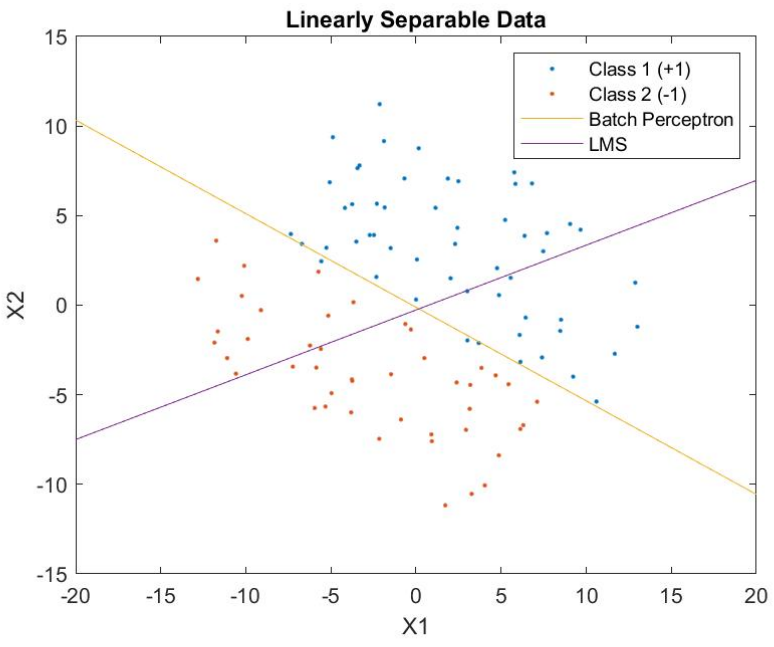

Figure 11: Comparison of Batch Perceptron and LMS on Linearly separable data after 100 iterations.

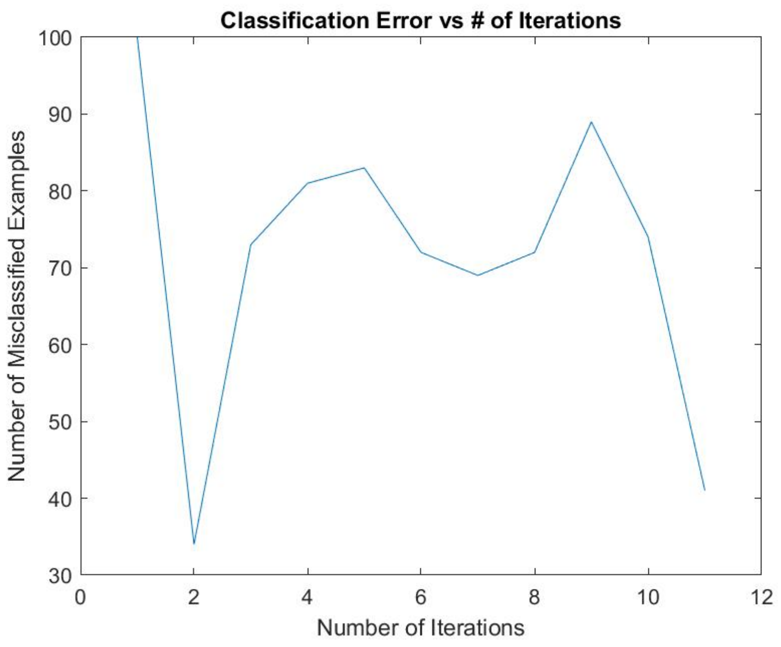

Figure 12: Mis-classification error after 10 iterations.

Error = 30%

$$w = \begin{bmatrix} 0.048 \\ -0.0655 \\ 0.181 \end{bmatrix} \tag{5}$$

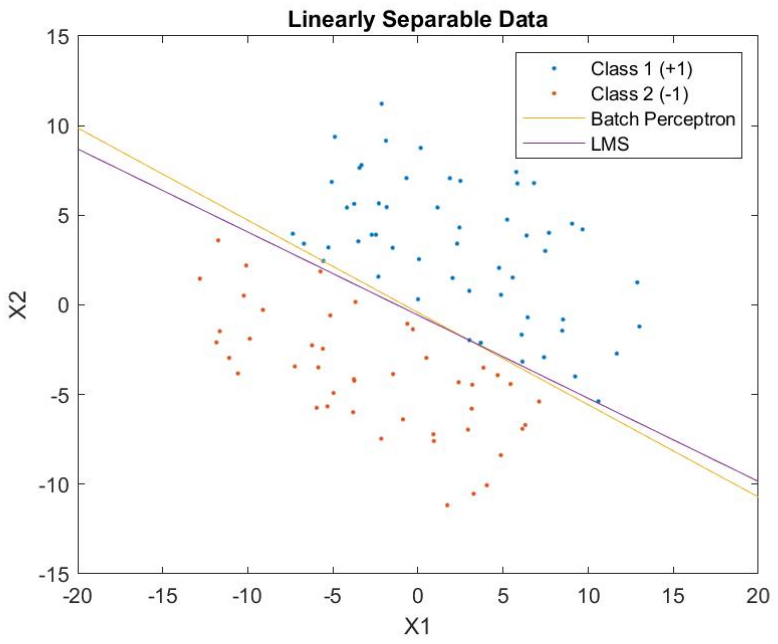

Figure 13: Comparison of Batch Perceptron and LMS on Linearly separable data after 100 iterations.

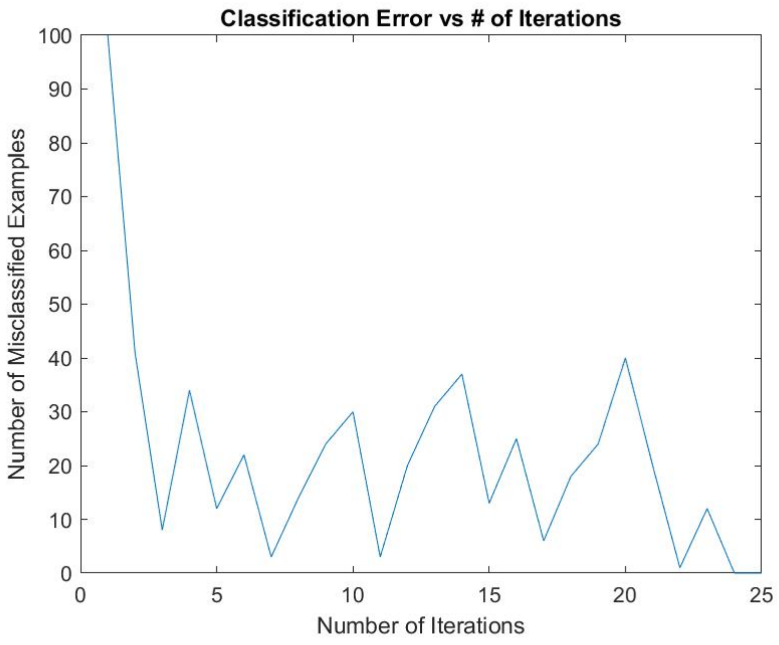

Figure 14: Mis-classification error after 100 iterations.

Error = 0%

$$w = \begin{bmatrix} 0.0899 \\ 0.0728 \\ 0.157 \end{bmatrix} \tag{6}$$

2.3.2 Non-linear Data

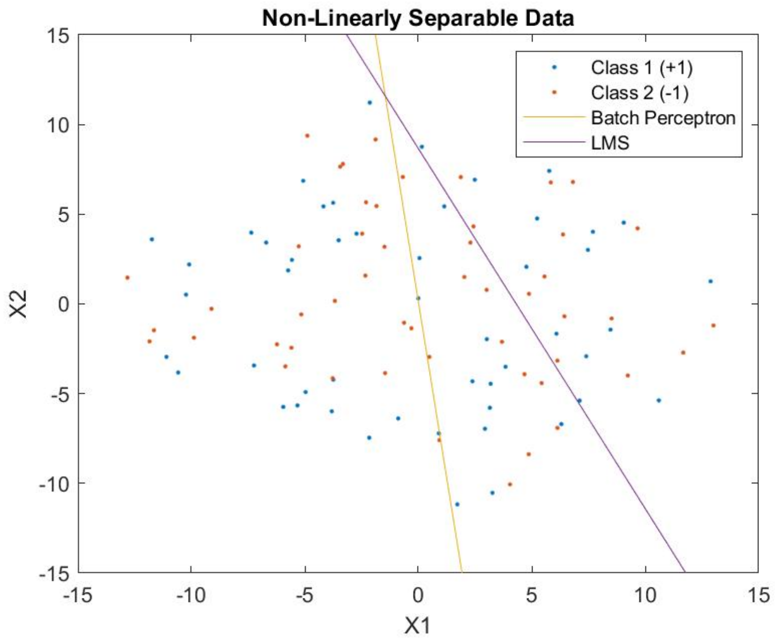

Figure 15: Comparison of Batch Perceptron and LMS on Non-linearly separable data.

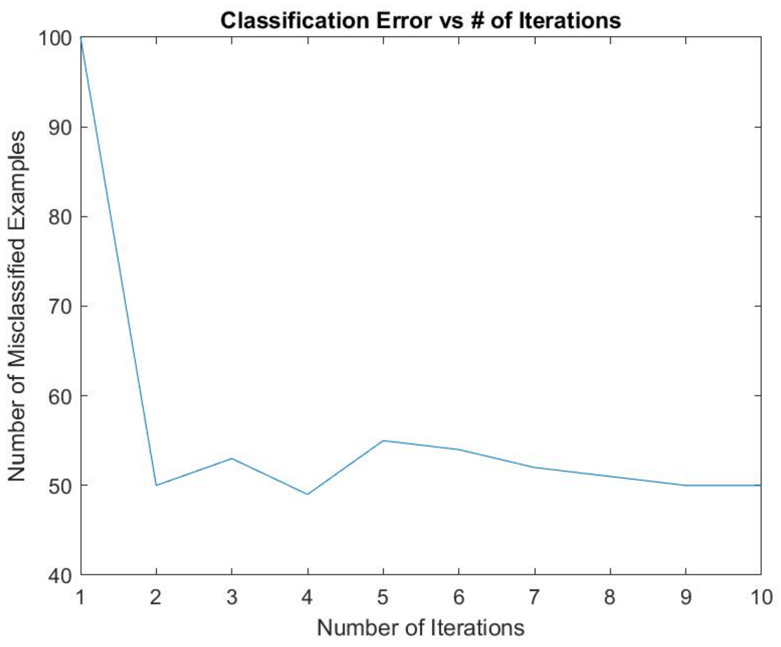

Figure 16: Mis-classification error after 10 iterations.

Error = 50%

$$w = \begin{bmatrix} 0.1309 \\ -0.0303 \\ -0.0151 \end{bmatrix} \tag{7}$$

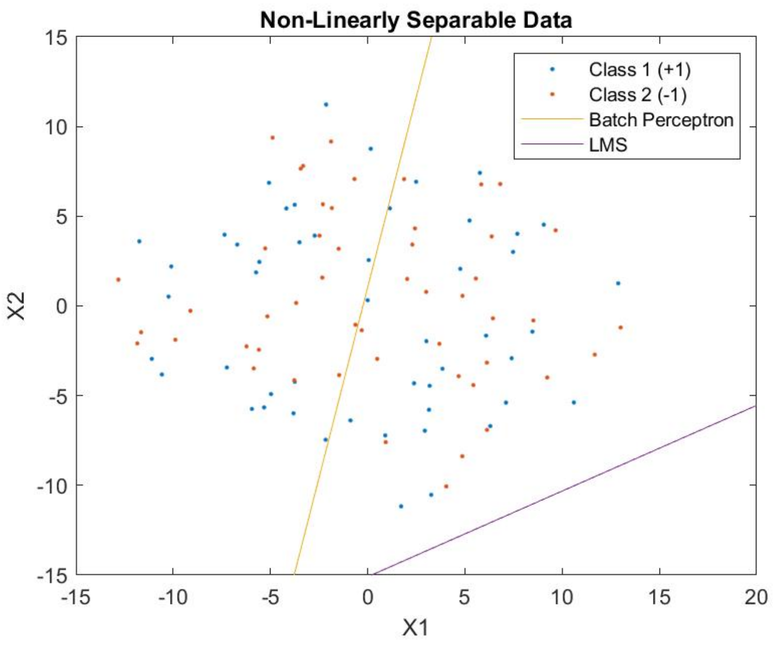

Figure 17: Comparison of Batch Perceptron and LMS on Non-linearly separable data.

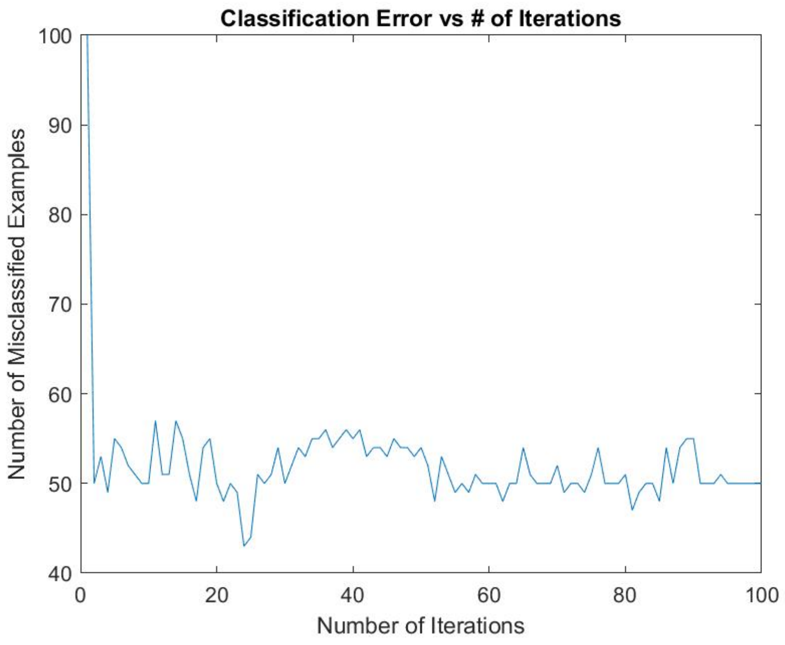

Figure 18: Mis-classification error after 100 iterations.

Error = 50%

$$w = \begin{bmatrix} 0.6130 \\ -0.0194 \\ 0.0406 \end{bmatrix} \tag{8}$$

2.4 Discussions

LMS performs very similarly to the Batch Perceptron on the linear dataset with a slight increase in error. The difference may be attributed to the fact the LMS rule is using a gradient descent based on real values as opposed to the binary 0's and 1's of the Batch Perceptron, so it may take longer to converge with an infinitely larger domain of values.

3 Problem 4: Backward Propagation Algorithm

3.1 Background

The backward-propagation algorithm is effectively a more complex version of training a perceptron using a feedback loop to correct errors that were made in a prior layer until error is overall minimized. With the added complexity comes the ability to learn more complex sets of data such as the MNIST handwriting samples.

3.2 Method

Set up a multi-layer perecptron with an input layer of 784 nodes, a hidden layer of 10 hidden output nodes, and 10 output nodes. Need to a create a desired vector by using a bit-wise interpretation for example of the number, 6 was denoted by $d = [0000010000]$ whereby the 7th position indicated its presence. Then, used a logistic activation function to go from input layer to hidden layer to output layer. Trained the neural network on each digit, then tested on the corresponding digits. Finally, calculated error via Mean-Squared-Error (MSE).

3.3 Results

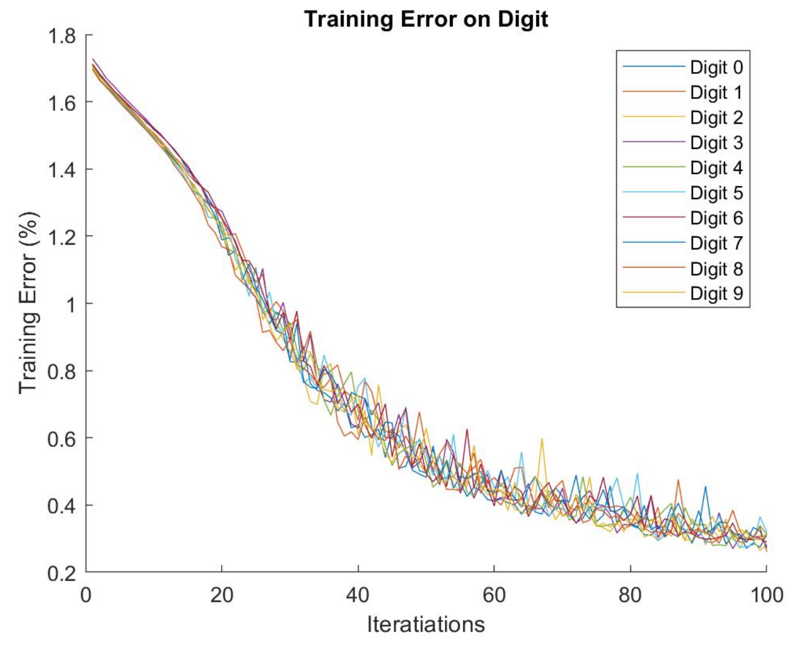

Figure 19: Training error for all the digits in the MNSIT 0-9.

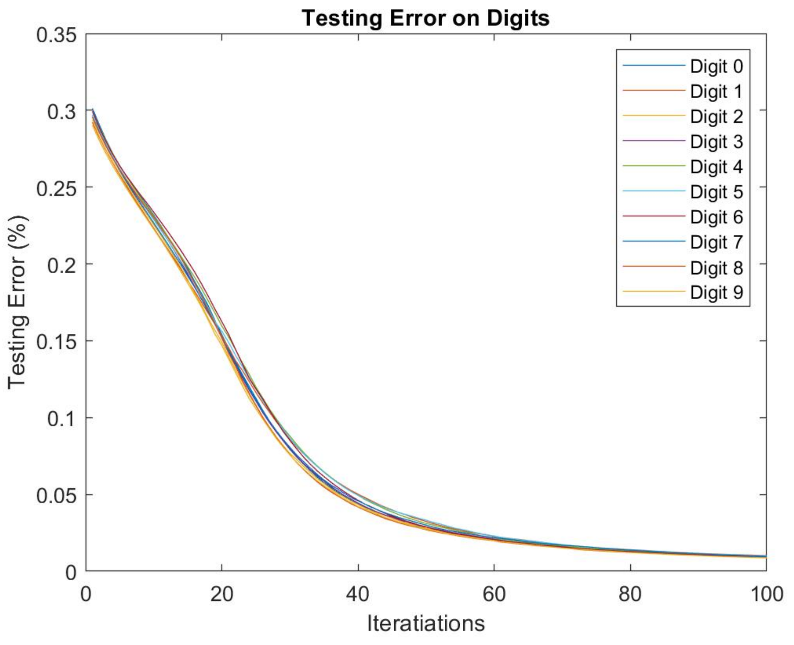

Figure 20: Testing error for all the digits in the MNSIT 0-9.

Digit	**Testing Error (%)**
Digit 0	0.90%
Digit 1	1.01%
Digit 2	0.93%
Digit 3	0.92%
Digit 4	0.94%
Digit 5	0.99%
Digit 6	0.92%
Digit 7	0.95%
Digit 8	0.88%
Digit 9	0.87%
Overall Error	0.93%

Table 1: Testing errors obtained from using MSE on Neural Network.

3.4 Discussions

The two-layer neural network performed better as the number of iterations increased leading to decreased error. This is similar to what was seen in the linear perceptron where the number of iterations pushed the classifier towards convergence as seen in both the training and testing error.