Introduction to Neural Networks (EE456)

Mini Project 1: SLNN for Linearly Separable Classes and Non-Linearly Separable Classes

By:

Syamsulbakhri Bin Ismail Hisamudin

October 2, 2024

Overview

1. This project is designed to analyze the use of Single-Layer Neural Network (SLNN) in separating data sets using linearly separable and non-separable datasets. It is designed as such to determine if SLNN will be able to find the most optimal solution. The data sets tested are labeled with:
    a. Set A (linearly separable dataset)
    b. Set B for (non-separable dataset).
2. Throughout this analysis, each training set is set to 35 epochs to maintain consistency across the perceptron.

Prerequisites

1. Matplotlib (Plotting and Graphing Modules)
2. Scipy (Loading .mat files)
3. Numpy (Calculations)

Implementation

1. Perceptron
2. Architecture:
    a. The perceptron design has two input layers (X1, X2)
    b. Single output layer (Y).
3. Activation function:
    a. signum (bipolar Heaviside)
    $f(v) = w_1 x_1 + w_2 x_2 + b$
4. Weight and bias:
    a. $w_{new} = w_{old} + \eta \times y[i] \times x[i]$ (η = learning rate) (Formula 1)
    b. $b_{new} = b_{old} + \eta \times y[i]$ (Formula 2)

5. Data Set A (Linearly Separable)
    a. Initialization:
        i. Weights, $w = 0$
        ii. Bias, $b = 0$
        iii. Threshold, $\theta = 0$
        iv. Learning rate, $\eta = 0.2$
    b. Training:
        i. The perceptron adjusts its weights until there are no errors. If there is an error, it will adjust the weights (Formula 1) and bias (Formula 2) with respect to the correct prediction.
6. Data Set B (Non-linearly Separable)
    a. Initialization
        i. Weights, $w = 0$
        ii. Bias, $b = 0$
        iii. Lower Threshold, $\theta_{lower} = -0.1$
        iv. Upper Threshold, $\theta_{upper} = 0.1$
        v. Learning rate, $\eta = 0.4$

b. Training

i. During training, the activation function for the region of threshold for the perceptron is as follows:

$$f(v(n)) = 1\,;\ v(n) > \theta_{upper}$$
$$f(v(n)) = -1\,;\ v(n) < \theta_{lower}$$
$$f(v(n)) = 0\,;\ \theta_{lower} < v(n) < \theta_{upper}$$

(Formula 3)

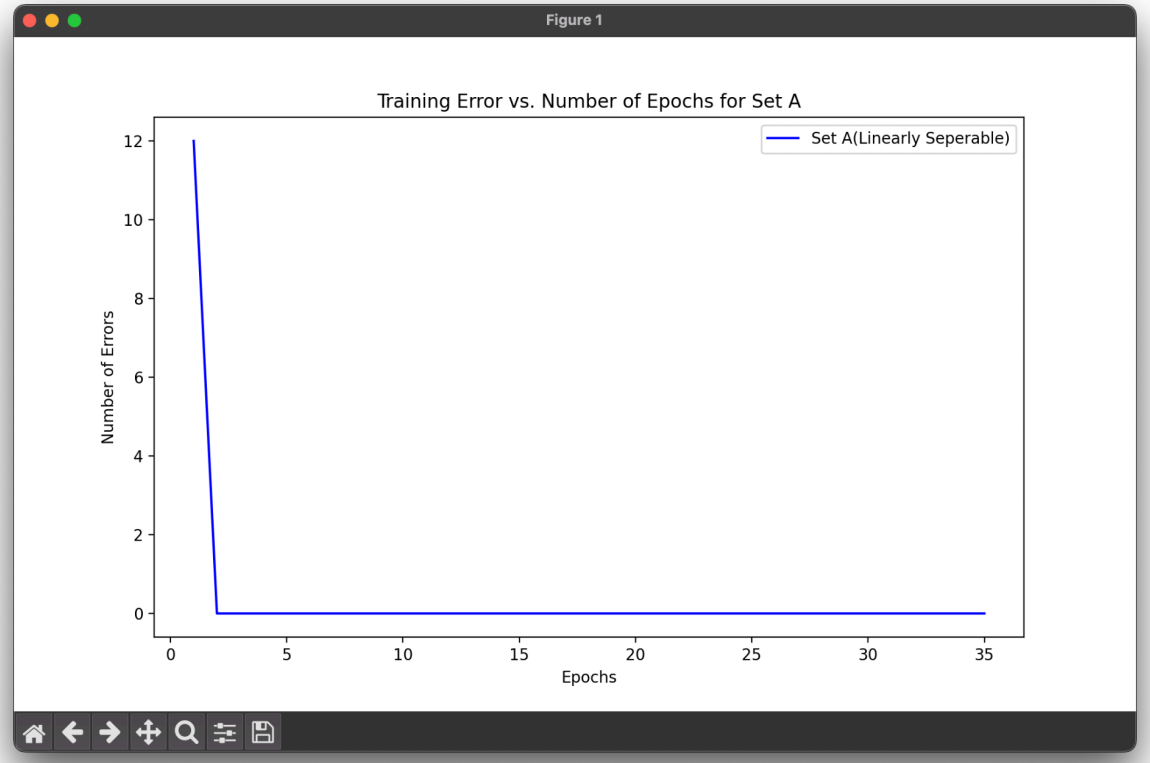
ii. If an error or multiple errors are detected, the weights and bias are modified using the Formula (1) and Formula (2) respectively.

Results and Observations

This section will provide the results, observations and insights gained from the analysis described in the implementation section. Each section is separated for dataset A and dataset B.

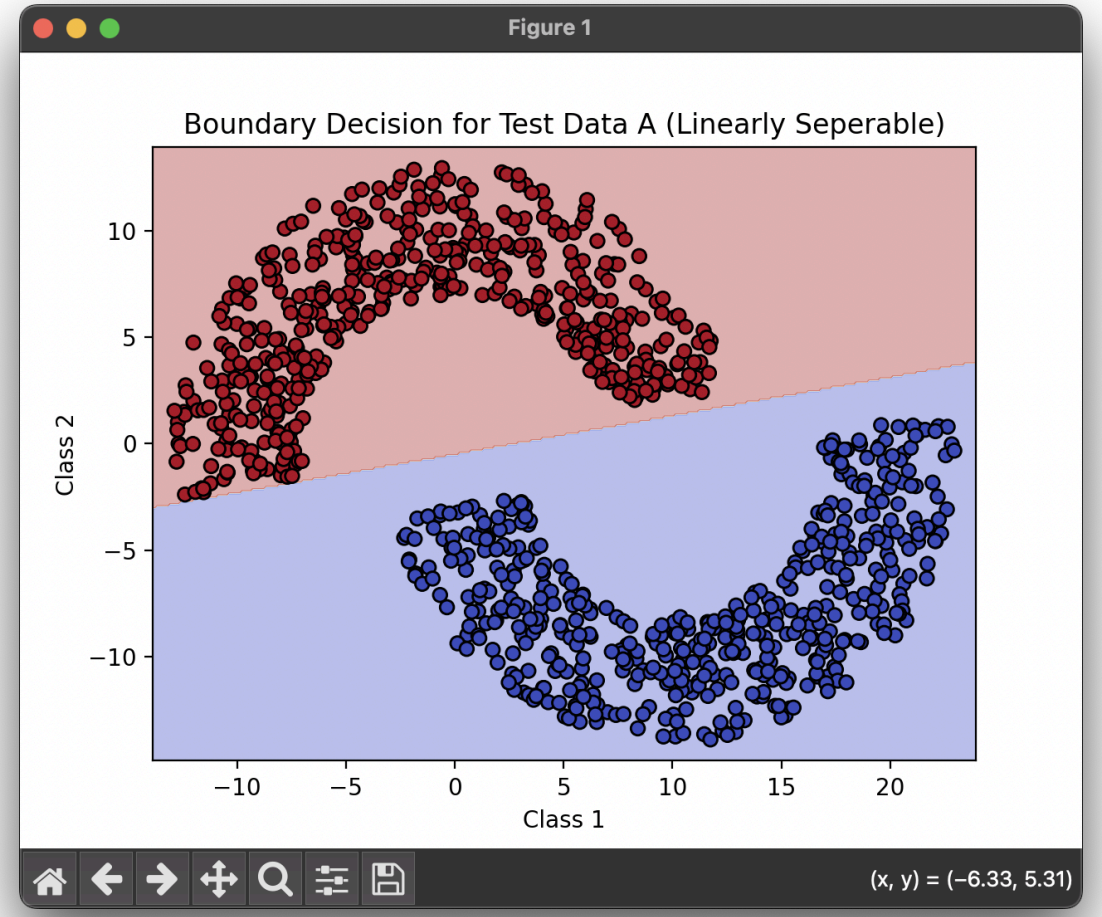
1. Data Set A (Linearly Separable)

a. Training Error



This shows the graphs of Number of Training Errors vs Number of Epochs for Set A. As shown in the graph, the perceptron has 0 errors after passing 2 epochs.

b. Test Data Plot and Decision Boundary



From the figure above, the data plots are completely separated from one another and there is a clear decision boundary separating them. The blue plots indicate Class 1 plot and the red plots indicate Class 2 plot.

c. Overall Error Rate

i. **0.00%**

This figure indicates that the perceptron is able to perfectly separate the different datasets.

d. Observations across different learning-rate, η

i. There are no obvious changes to the performance of the perceptron when the learning rate is between 0.2-0.8. The graph for Training Error and Test Data Plot remains essentially the same.

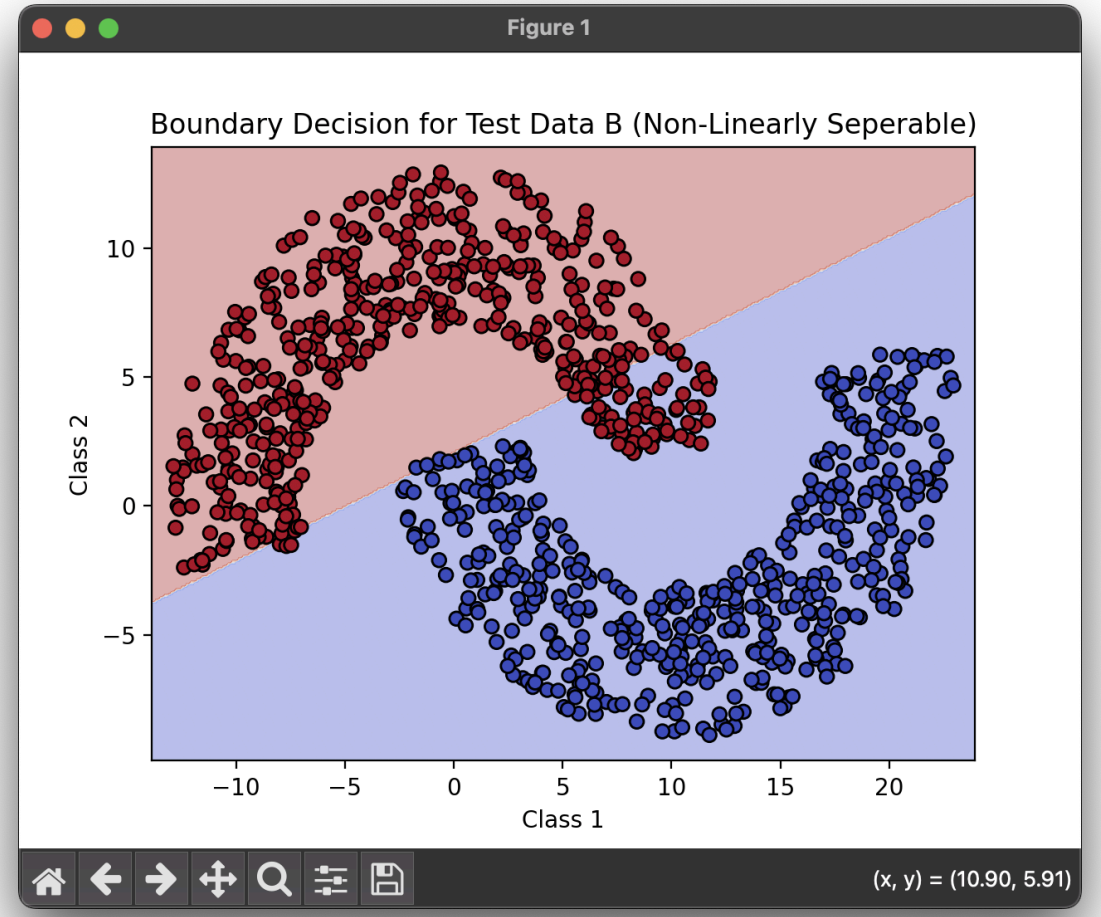
ii. Learning rate outside of the specified range is not chosen in order to avoid over-dampening and under-dampening of the perceptron. This occurrence happens due to the changes made to the weights and bias are too significant (under-dampening) or too little (over-dampening).

Data Set B (Non-linearly Separable)

a. Training Error

This figure shows the graph of Number of Errors vs Number of Epochs for dataset B. As shown in the graph, the number of errors is not constant but fluctuates between the range of 30-60 after several epochs.

b. Test Data plot



From the figure above, some of the data plots are not fully separable by the perceptron. This can be seen that the red plots and blue plots are crossing through the decision boundary identified by the weights and thresholds.

c. Overall Error Rate

7.70%

This figure described that the perceptron are not capable of perfectly separating the datasets into their target classes but still managed to increase the readability between them.

d. Observations across different learning-rate, η

i. There are no obvious changes to the performance of the perceptron when the learning rate is between 0.2-0.4. The graph for Training Error and Test Data Plot remains essentially the same.

ii. There are significant changes to the error rate when it is set between greater than 0.1. The error rate will be in the range of between 8.00-10.00%. This increase in error rate is due to not reaching the minimum number of errors during training. Although there is a reduction in performance, the graph for Training Error and Test Data Plot remains essentially the same.

iii. Learning rate outside of the specified range is not chosen in order to avoid over-dampening and under-dampening of the perceptron. This occurrence happens due to the changes made to the weights and bias are too significant (under-dampening) or too little (over-dampening).

Design Choices

1. The learning rate, η, for dataset A is set to 0.2 as there are no significant changes to the performance for the perceptron.
2. The learning rate, η, for dataset A is set to 0.4 as the perceptron is observed to reach the lowest number of errors most often.
3. 35 Epochs are standardized for both perceptrons for consistencies across training.

Conclusion

From the data gathered, the SLNN perceptron can only find the most optimal decision boundary and weights for linearly separable datasets (Dataset A) in which each data is distinct against one another. In contrast, non-linearly separable datasets (Dataset B) caused the SLNN perceptron to be unable to separate each data plot into their classes. Nevertheless, the SLNN perceptron is still able to distinguish some of the data plots with slight errors. Thus, SLNN perceptron can be optimally used for linearly separable datasets but suboptimally for non-linearly separable datasets.