Ahmad Shalaby

CS 688

Homework 3

Effect of Different Training Data on Results of Perceptron Linear Discrimination On Testing Set

Introduction

The perceptron algorithm involves computing a line based on given training data points, under the assumption that the data is linearly separable. The function passes two parameters from the data and assigns them weights, as well as a bias constant. This forms the linear equation w1x1 + w2x2 + b, where w1, and w2 represent the weights and x1, x2 represent the training parameters. The b represents constant bias. There is a step function which takes in the data, plugs it into the equation above, and assigns the result to a class. The boundary of this class is chosen by whether the above equation is greater than or equal to zero. If so, it is class 1, else it is of class 0. The true class of the data is given in the data set and is used to compare the accuracy of the perception in classifying the data.

The algorithm is passed the testing data, at every point in testing data, if the result of the perception is not equivalent to the actual label, the perceptron "updates" the respective weights in the equation as shown.

```
if (stepFunction(w1*x1 + w2*x2 + b) > y): # aka step gave 1 and y is zero
    w1 = w1 - x1
    w2 = w2 - x2
    b = b - 1
# update weights properly

if (stepFunction(w1*x1 + w2*x2 + b) < y): # aka step gave 0 and y is one
    w1 = w1 + x1
    w2 = w2 + x2
    b = b + 1
#update weights properly
```

Figure1: Step function updating weights of perceptron based on true label.

Methods

In this experiment, the weights were initially and arbitrarily set to 5 for w1, and w2, and 1 for b. Initially the weights were going to be random, but this had a large effect on the perception since the training data only contains 40 data points. Because of this and to allow for better comparison, the initial weights were kept constant for the different data sets. The perception was run **twice** per data set, which meant that the same 40 data points where used twice for updating the algorithm. Conceptually, it could be said that the first 40 data points constituted the "random initialization" of a good perceptron initial weight, with the next 40 dialing the weights more effectively.

This experiment was run for 10 different test sets, with the weights initialized back to 5 and 1 for each new training set. After the training set was run, the updated perception was used to run the testing data, and the error rate was recorded. This error rate was calculated by the number of wrong guesses divided by total points in the testing data. Keep note that the perception weights did not update while running the testing data.

Results

	W1	W2	b	Error rate
Set 1	1.9968810000000006	0.23736300000000066	-5	0.05
Set 2	3.6017989999999998	-1.362230999999999	-9	0.0775
Set 3	4.974427	-4.311726000000002	-14	0.162
Set 4	2.8587940000000023	-3.2671390000000002	-11	0.2115
Set 5	5.585748999999998	0.7321900000000001	-16	0.0335
Set 6	6.649275	-0.7428509999999986	-14	0.0705
Set 7	6.135642000000002	1.7512840000000005	-10	0.198
Set 8	10.742928999999998	3.044209999999998	-9	0.311
Set 9	1.8350240000000024	0.778023999999998	-10	0.0185
Set 10	3.0564909999999994	0.14637199999999861	-8	0.033

Figure 2: W1, W2, b, and Error rate values across the 10 different training sets used on the same testing set. Note that the large decimal numbers are due to floating point errors in python

To help with visualization, the perceptron boundary equation $w1x1 + w2x2 + b = 0$ was graphed by transforming it into the form $x2 = (-b/w2) – (w1/w2)x1$, with x2 being on the y axis and x1 being on the x axis. This boundary line was drawn across the training data to visualize how well the boundary line separated the data. The data was given labels showing which true

class it belongs to. Finally, the boundary lines where also drawn on top of the real testing data to better understand how the lines and their separation correspond to the error rates.

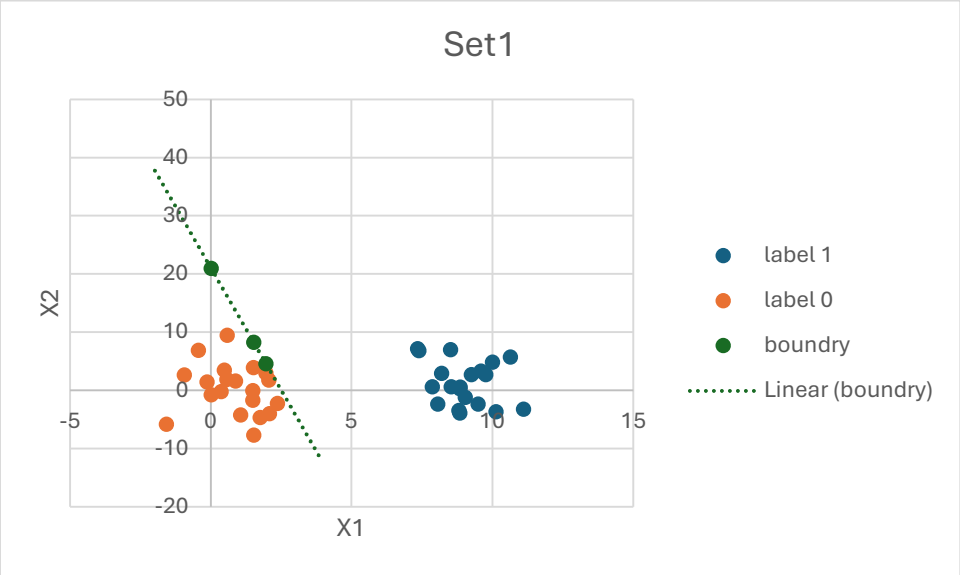

Figure 3: set 1 training data with boundary

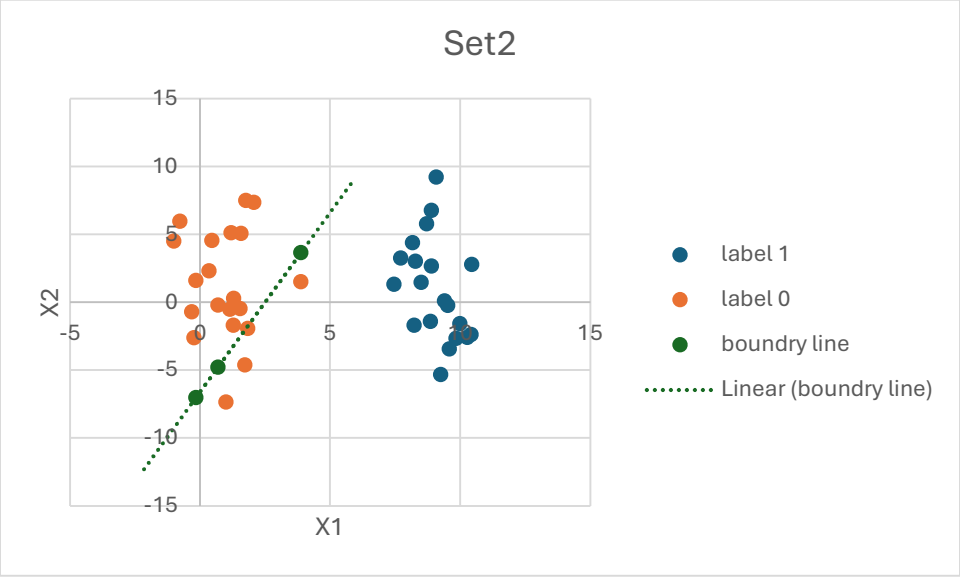

Figure 4: set 2 training data with boundary

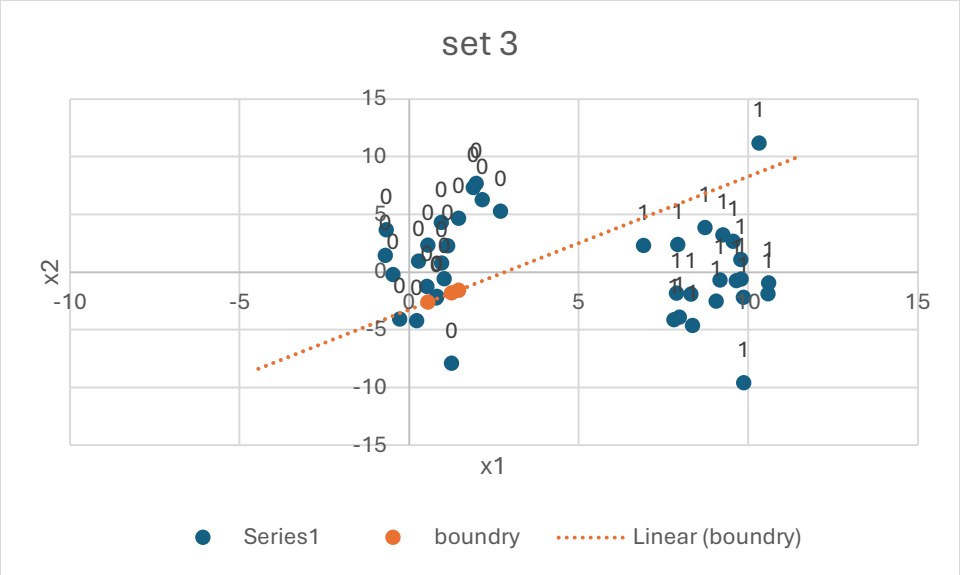

figure 5: set 3 training data with boundary

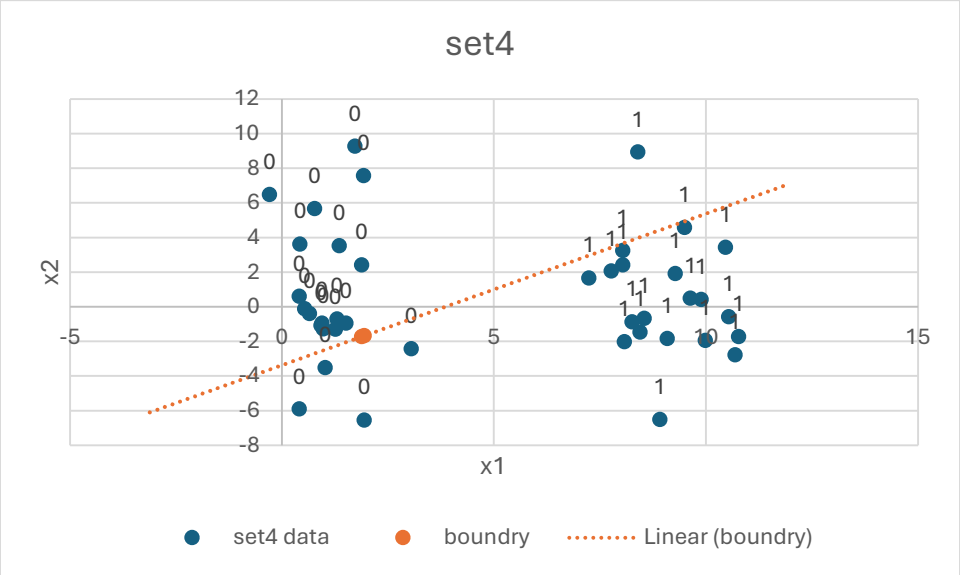

figure 6: set 4 training data with boundary

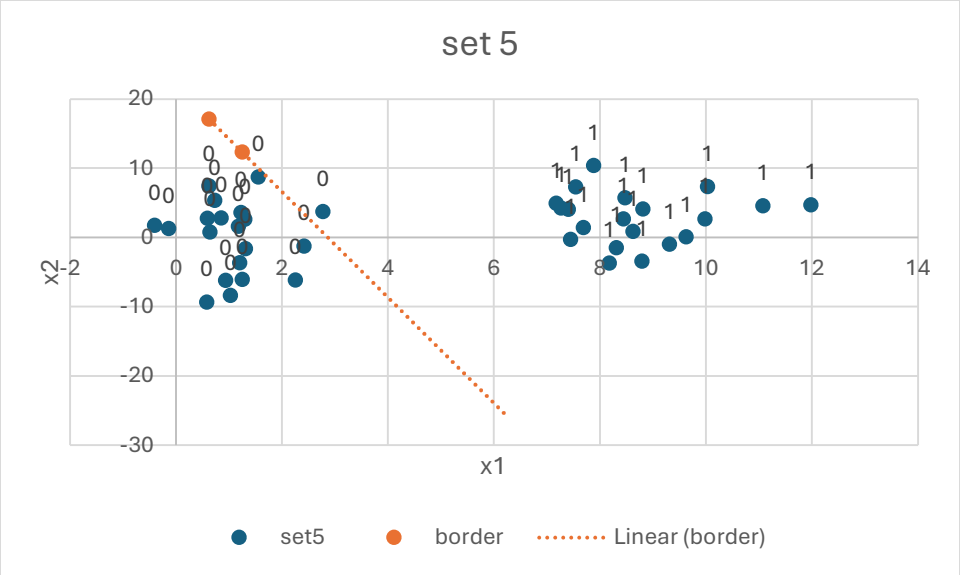

figure 7: set 5 training data with boundary

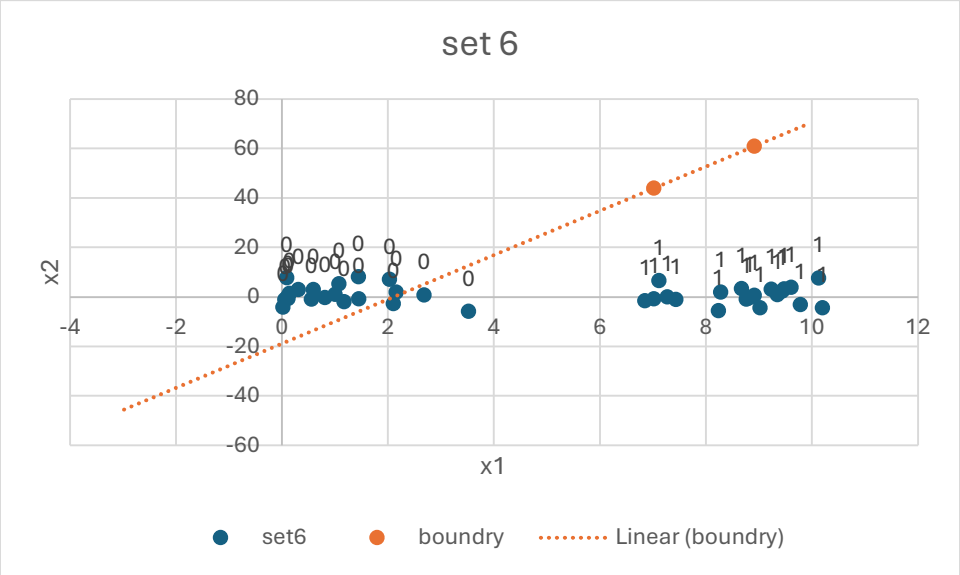

figure 8: set 6 training data with boundary

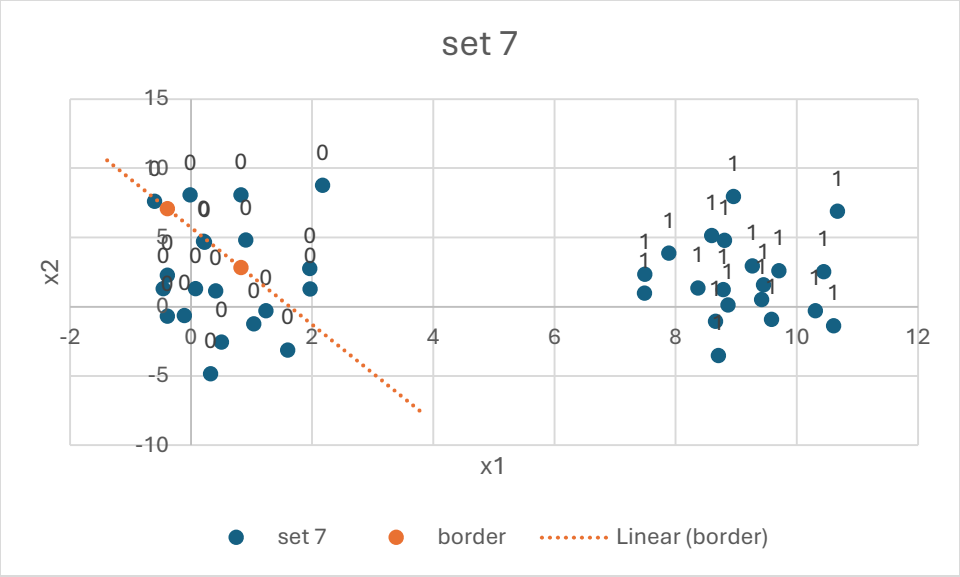

figure 9: set 7 training data with boundary

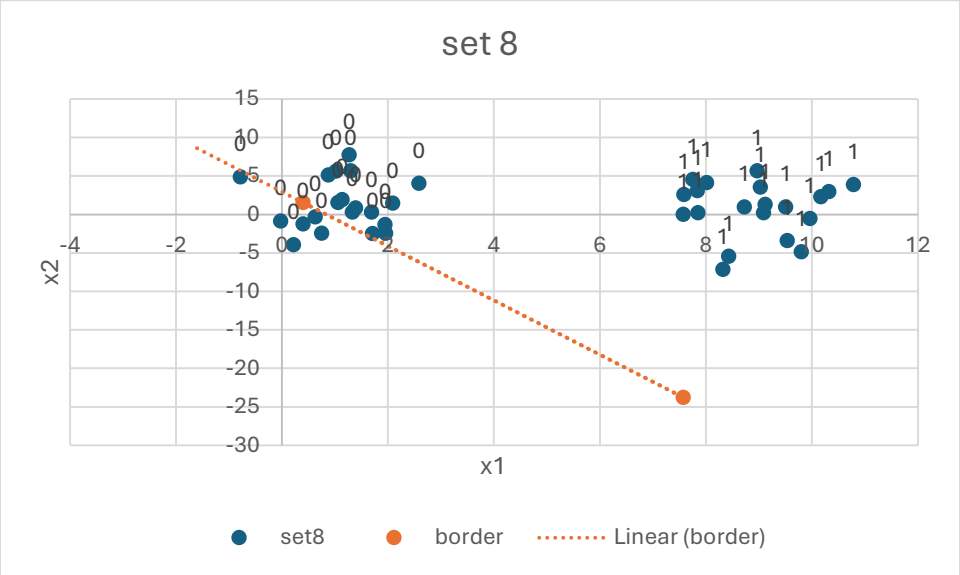

figure 10: set 8 training data with boundary

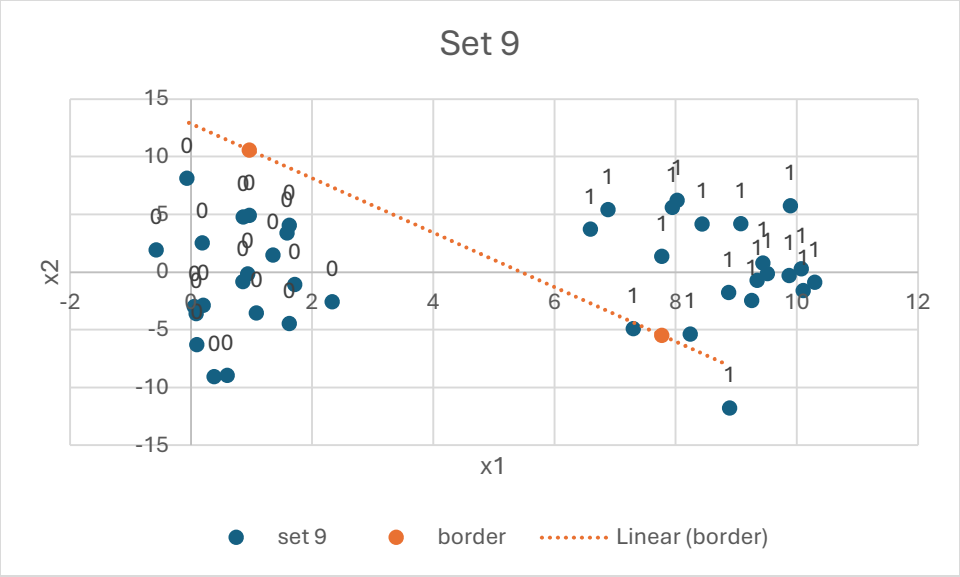

figure 11: set 9 training data with boundary

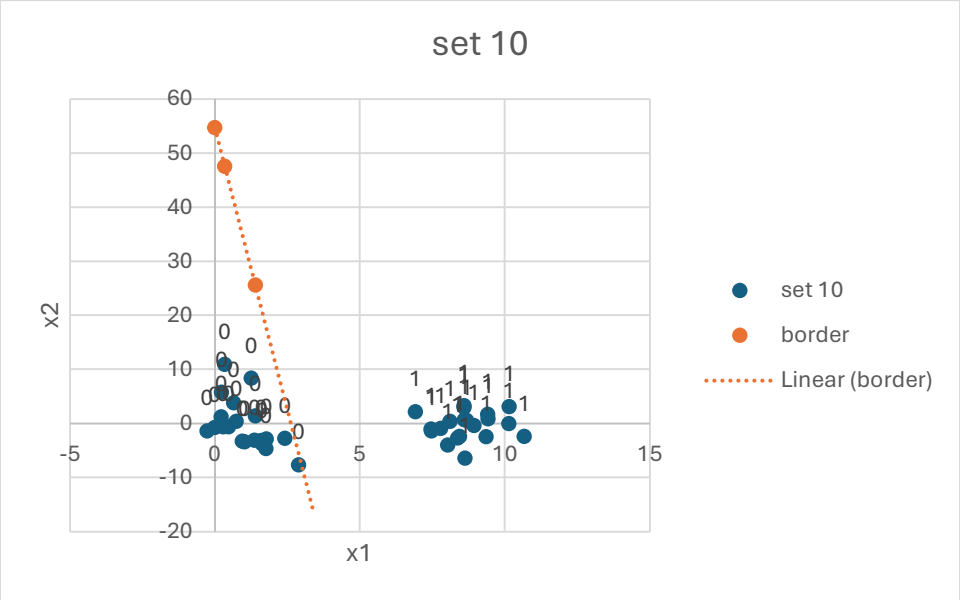

figure 12: set 10 training data with boundary

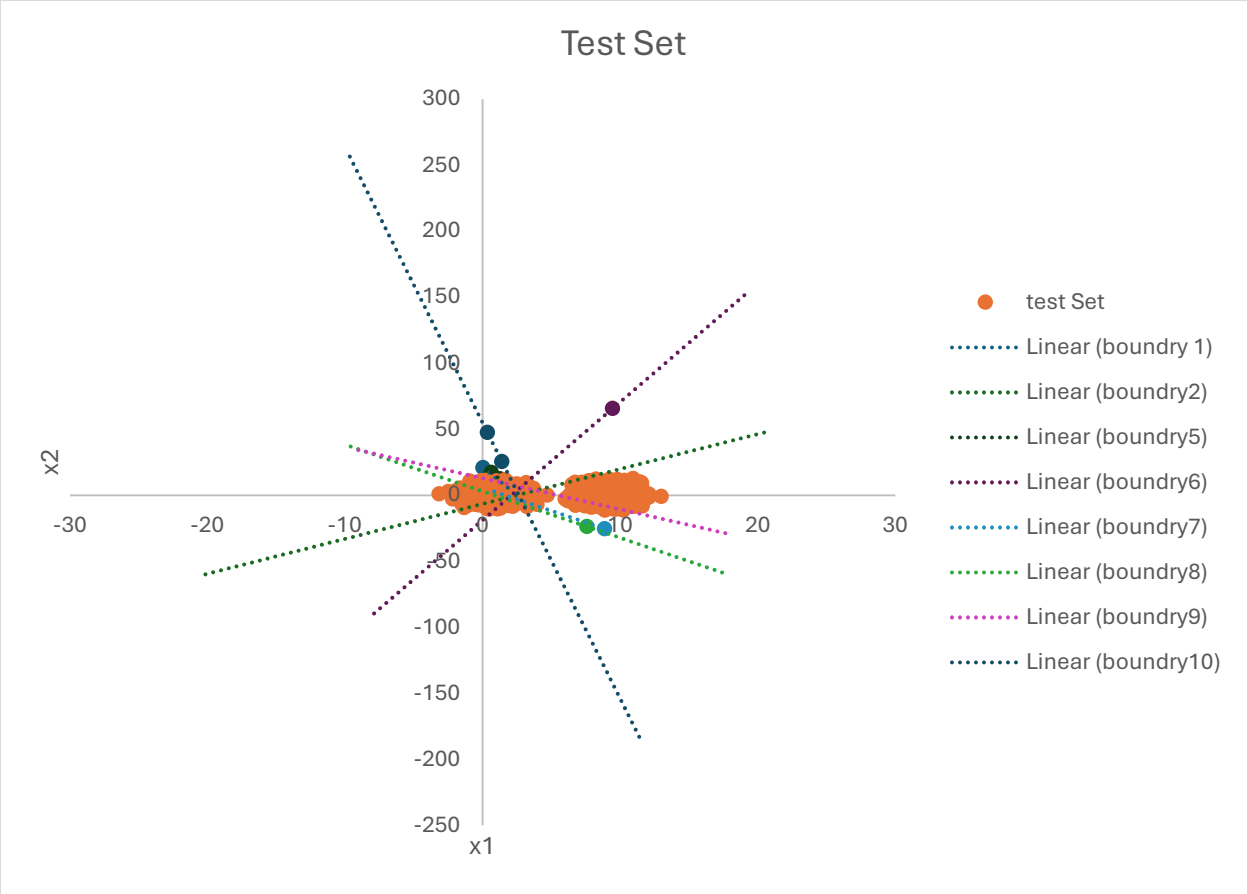

Figure 13: different linear boundaries for sets 1-10 on the testing data

Conclusions

As shown in figure2, the training data used has a large effect on the quality of the perception on the testing data, with the highest quality perceptron being training set 9 with an error rate of 0.0185 and the worse being training set 8 with an error rate of 0.311.

This is not obvious, as the perceptron equation should converge with a solution if the training data used matches the testing data properly. A reason for this error could be the presence of the W2 weight. As shown conceptually in figures 3-13, the data used in this training seems much more linearly separable across the x1 axis, with the x2 axis seemingly irrelevant to the label of the data point. Since the perception weights has w1, and w2 being equivalent, this clouded the perceptron's judgment to being able to separate the data. This is also shown in the data, with training 9 having a w2 value of 0.779 and training set 8 having a w2 value of 3.04. As the perceptron places less weight on the x2 values the quality of the prediction increased. As a result, when excluding x2 values from the perceptron entirely, the quality of the prediction increased.

It is shown through the data that different testing data impacts the quality of the perceptron, even if the data is all generated through the same equation. This suggests that engineers looking to utilize the perceptron algorithm to classify data should be work ahead of time to ensure that the data they have is of high quality and to prepare the perceptron with the data they have in mind. Future work can be done to see if this pattern still emerges when the training data set has a larger sample size, or if eventually all perceptron's converge to the optimal solution given enough data.