schuler_ece_527_report_02

September 12, 2024

1 GMU ECE 527 - Computer Exercise #2 - Report

Stewart Schuler - G01395779
20240912

1.0.1 Exercise 2.1

Poacket Algorithm I modified the provided SDG alogrithm to implement the pocket alogirthm. *Figures 1* and *2* demonstrate the resulting decision boundary and accuracy scores. As expected the pocket alogirthm performed equal to, or better than SGD on both datasets.
In order to make the algorithms work for the second dataset with minimal overlap, because that dataset has an offset from the origin the data sets needed to be modified to included an extra feature of all ones in order to learn the ω_0 bias.

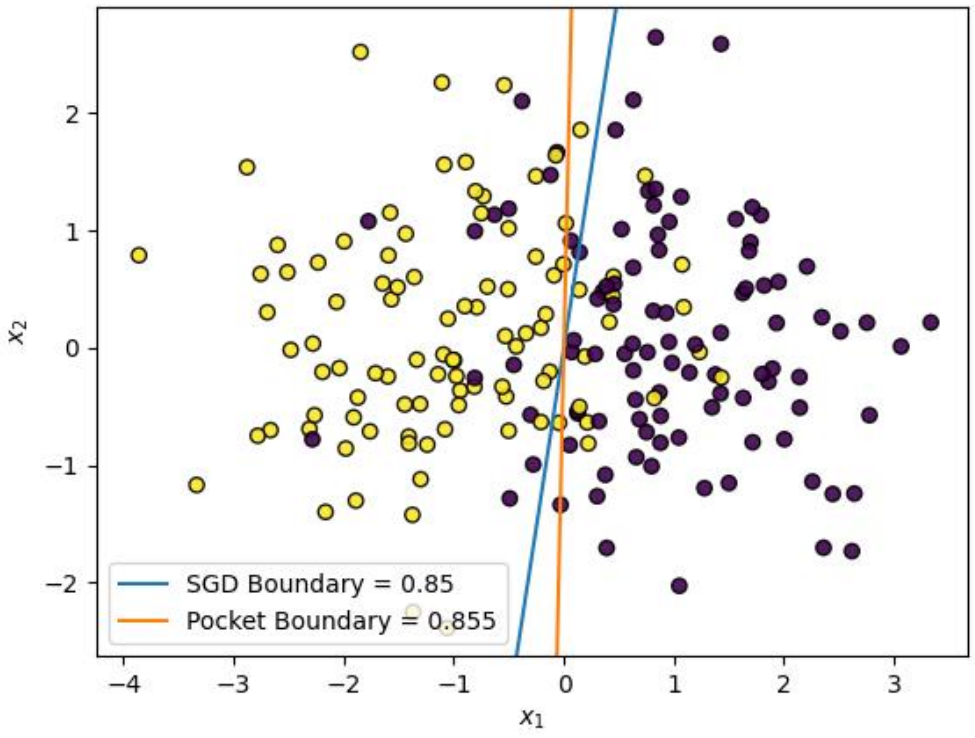

Figure 1. SGD vs Pocket with Overlaping dataset

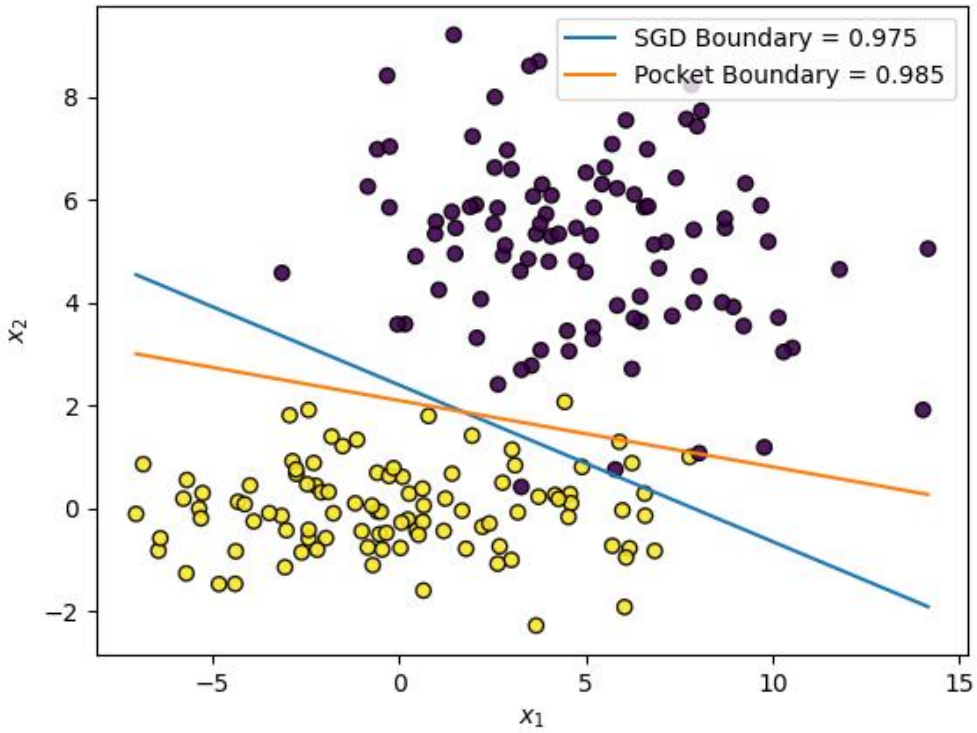

Figure 2. SGD vs Pocket with Overlaping dataset

Q: Will PLA work with target values of y = 0 and 1, or is it necessary to convert them to y = ±1? Does the Perceptron class care what the target values are?

A: No PLA will not work with y values of 0/1. This is because when an updated to the ω vector is triggered, it should update by $eta * X_i * y_i$. If $y_{i]}$ is zero the the update step will also be zero in size, that is no change in ω.
The *scikit-learn* implementation of Perception works equivilantly for y values of 0/1 or -1/1. *Figures 3-6* show the outputs of PLA and the Perceptron algorithm on the same dataset for both cases of y labels.

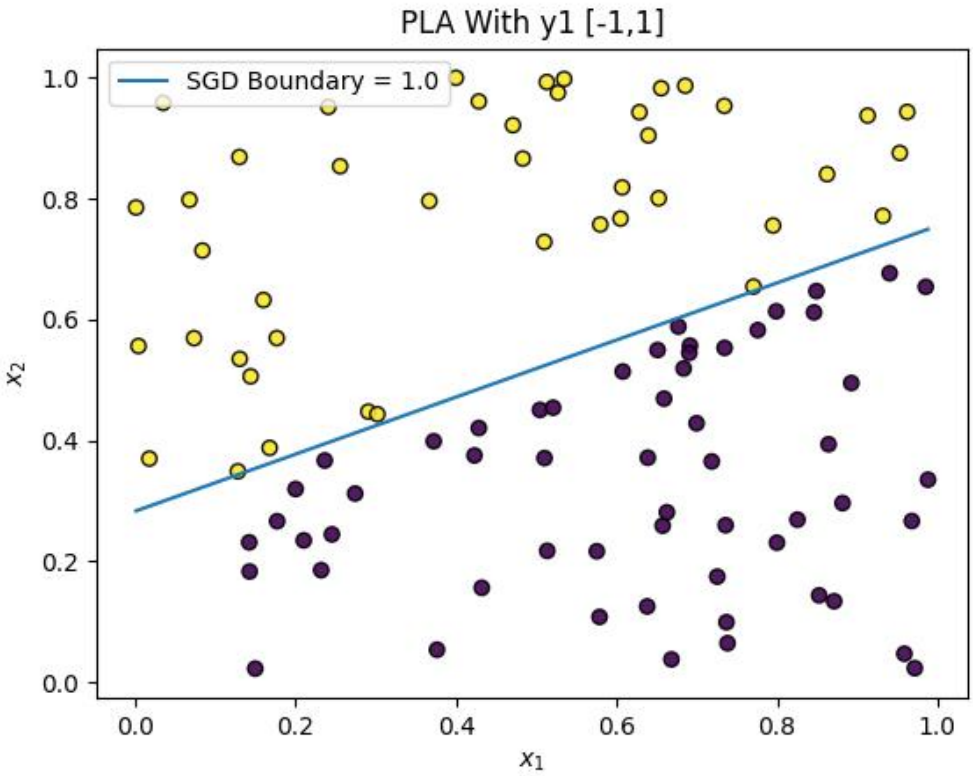

Figure 3. PLA with y=-1/1

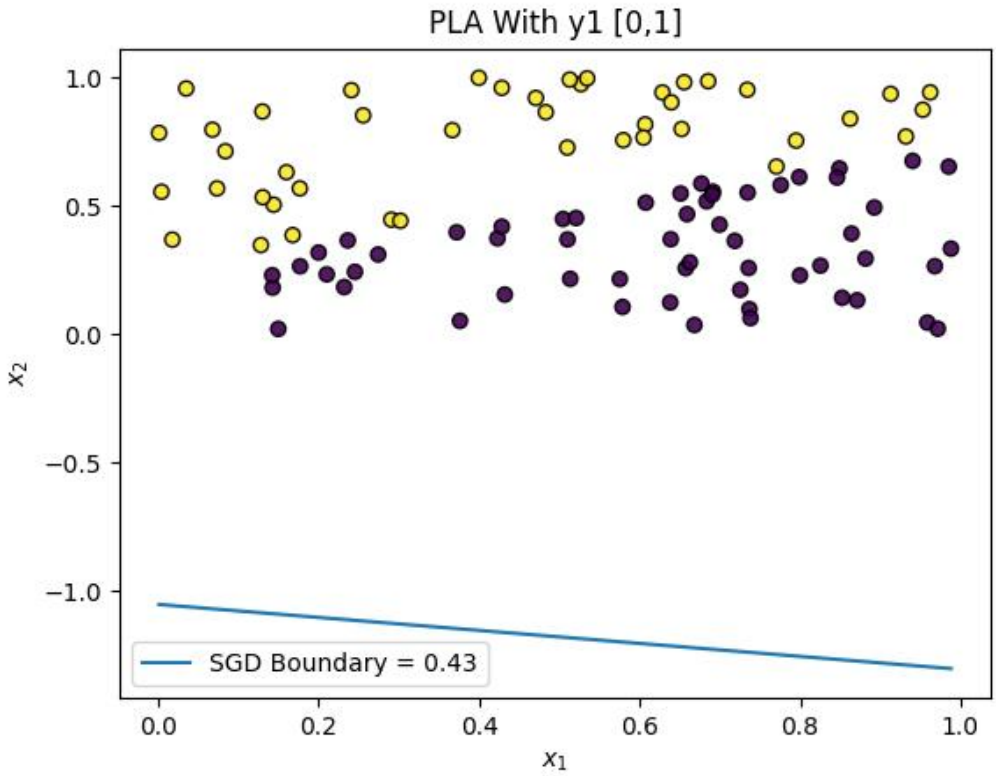

Figure 4. PLA with y=-1/1

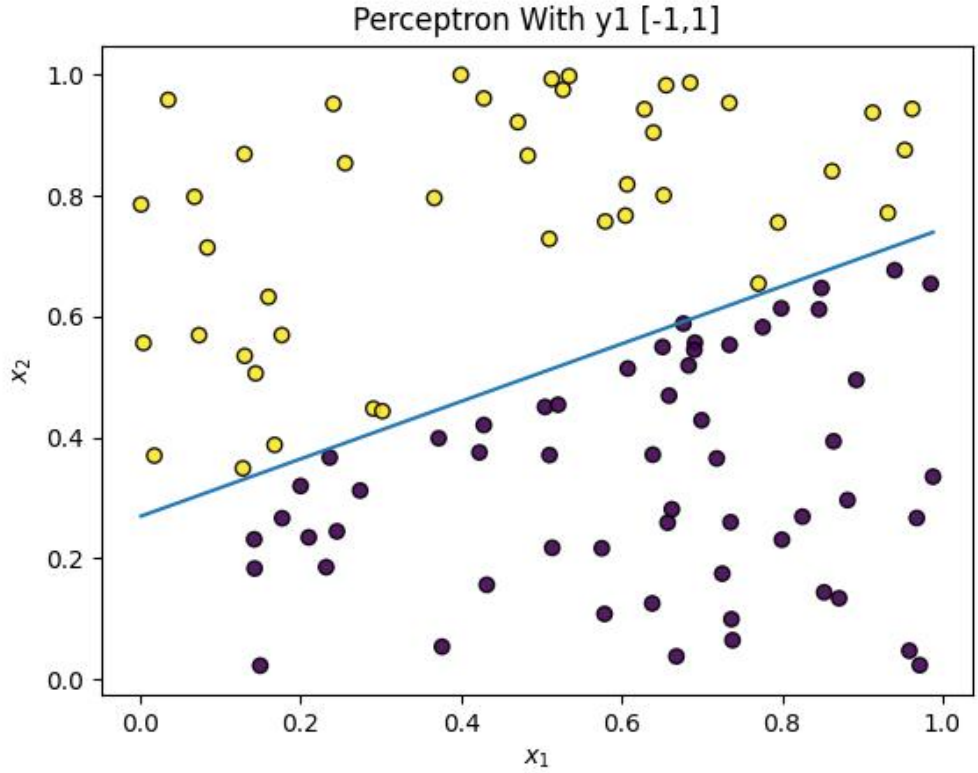

Figure 5. Perceptron with y=-1/1

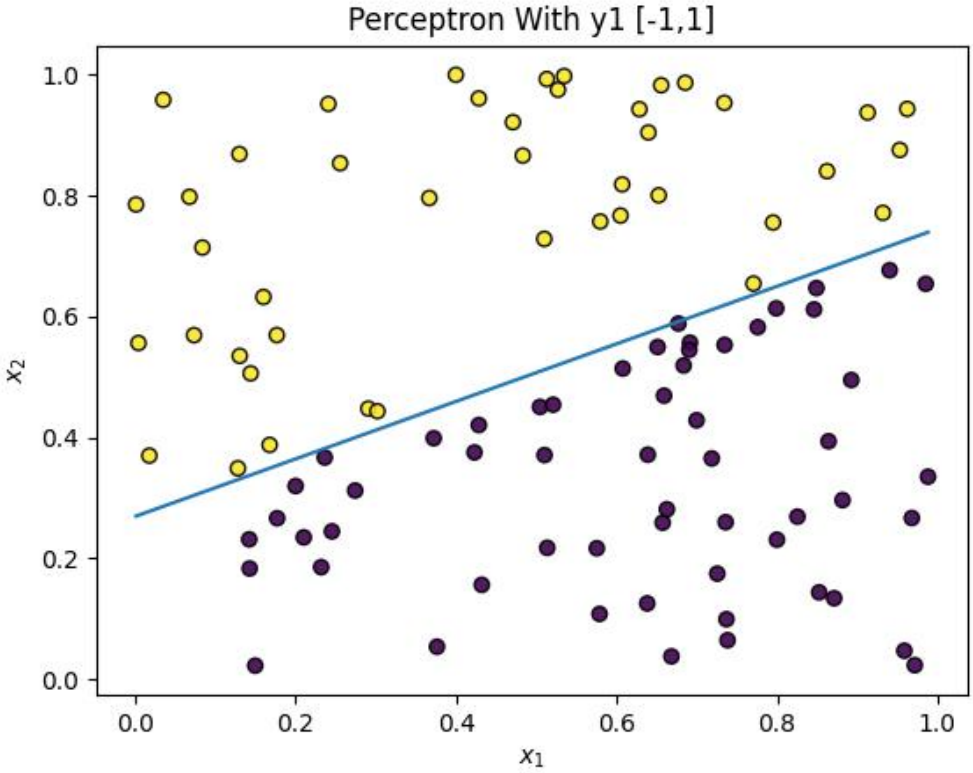

Figure 6. Perceptron with y=-1/1

1.1 PLA Experiments

1.1.1 Dataset 1

For dataset 1 because it is linearly separable PLA does converge. This can be seen by looking at the learning curve in *Figure 7*. When plotting the learning curve it was forced to run for 50 epochs, when letting the Perceptron class auto stop on convergence I found it to converge after 8 iterations. This was with $tol = 5E-6$ but I found similar fast convergence for other tested values of tol. Since for the learner to auto stop the loss moss be stable for some number of epochs auto stopping at 8 means that the learner converges quickly and stays stable. Which the plotted learning curve agrees with.

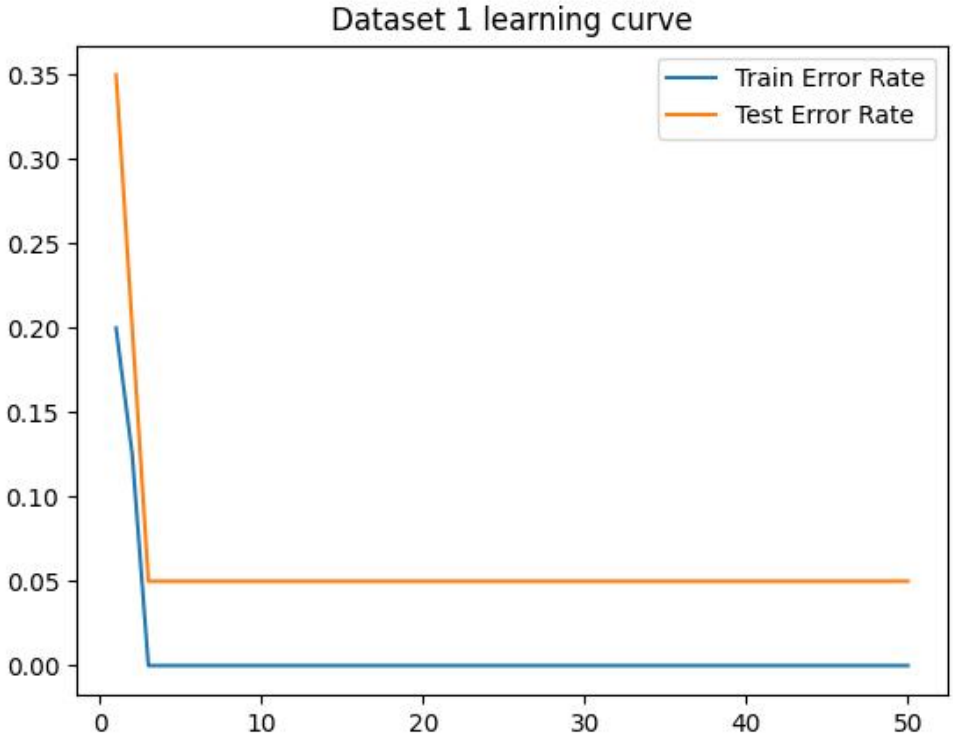

Figure 7. Dataset 1 PLA learning curve

We see that the learned decision boundary performs quite well on the held out test set.

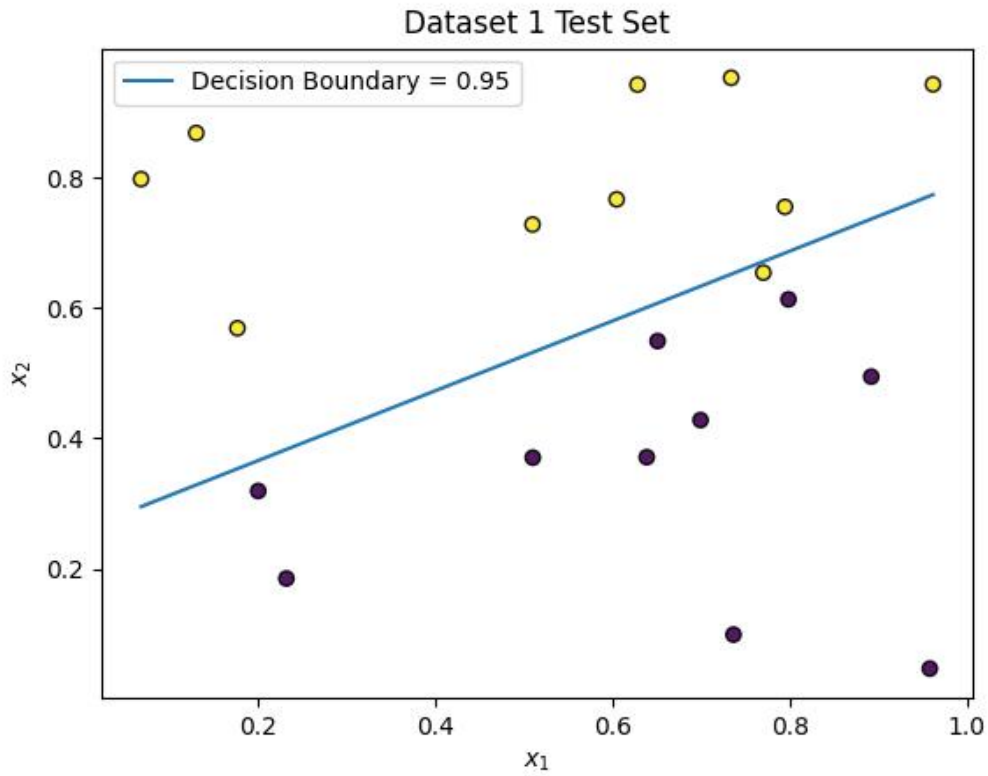

Figure 8. Dataset 1 Test Set Result

Figure 9 shows the affect of modifying the random seed for the class. It is demonstrated by the plot that the resulting converged upon decision boundary depends on the order of the data.

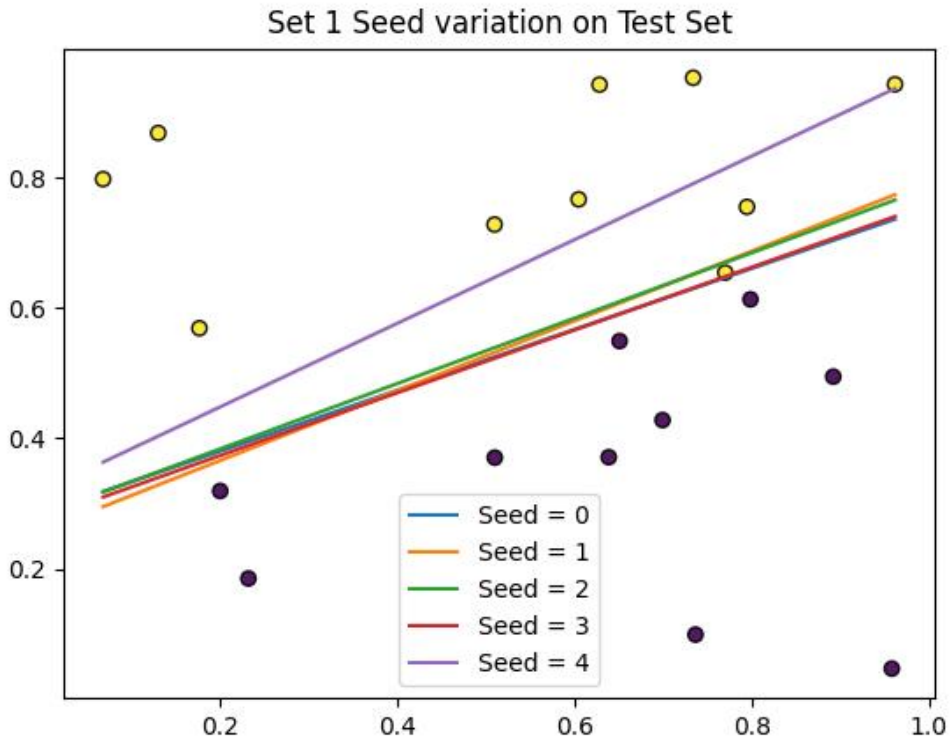

Figure 9. Dataset 1 Seed Variation

1.1.2 Dataset 2

With again a tol value of $5E-6$ dataset 2 converged after 10 iteration. Like dataset 1 this happens because the loss is now longer being improved by subsequent epochs. However unlike dataset 1, because this dataset is **not** linearly separable it doesn't converge to a perfect accuracy. But rather some error. We can see that by chance when plotting the learning curve it happened to be stable from the first epoch onward. Testing with different random seeds produced other learning curved that were unstable, similar to the ones presented for datasets 3/4.

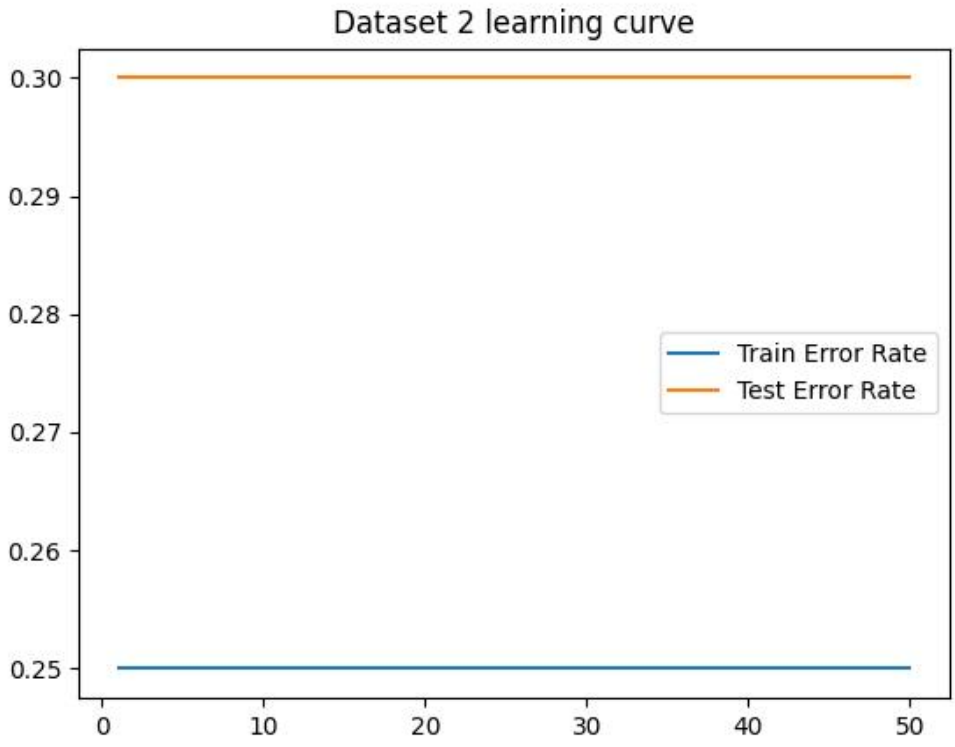

Figure 10. Dataset 2 PLA learning curve

The linear classifier performs okay on the held out test set. This is likely because the randomly chosen test set is not quite as non-linear as it could've potentially been randomly chosen to be.

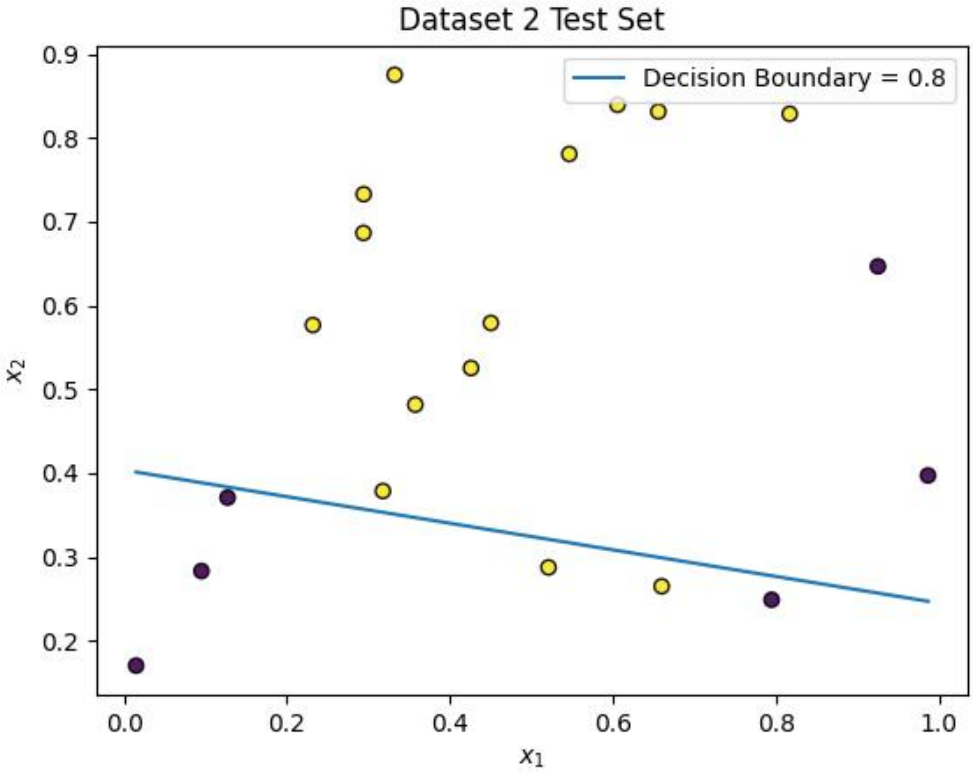

Figure 11. Dataset 1 Test Set Result

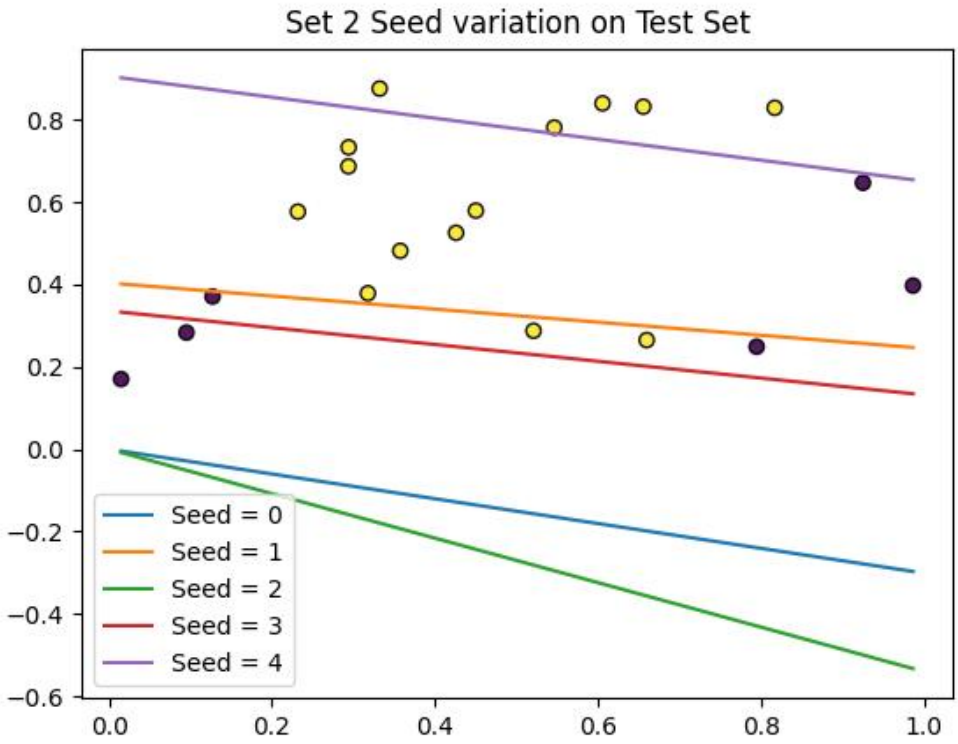

Figure 12. Dataset 2 Seed Variation

1.1.3 Dataset 3

With again a tol value of $5E-6$ dataset 2 auto stopped after 12 iteration. Because this dataset is highly non-linear it can be seen by the learning curve that it is never able to converge and fluctuates.

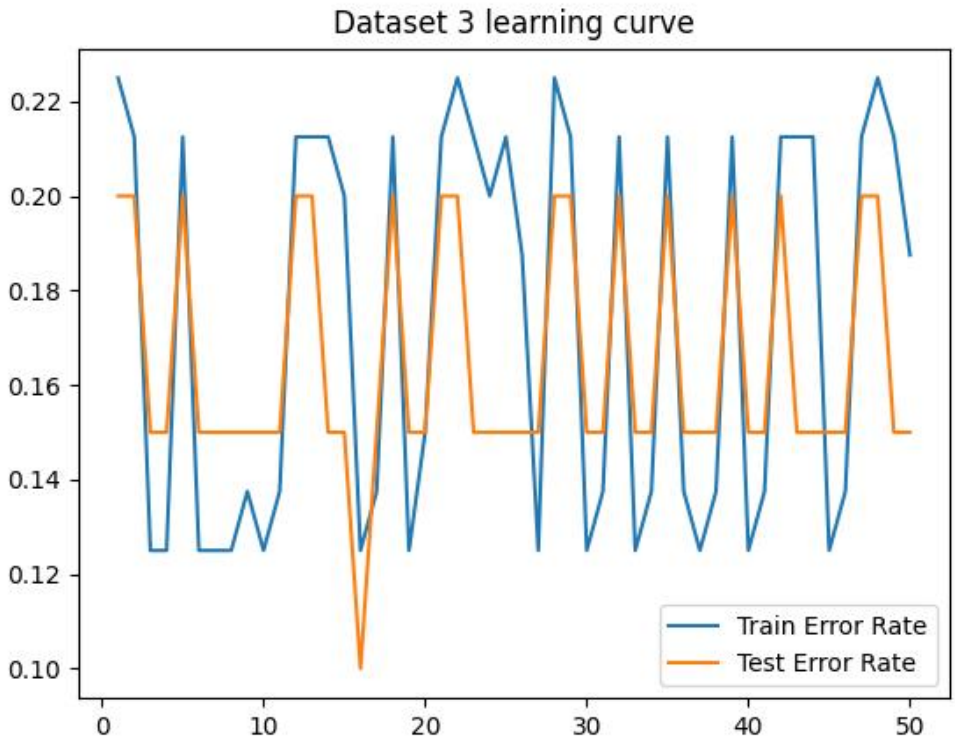

Figure 13. Dataset 3 PLA learning curve

Like with the non-linear dataset 2, the linear classifier does better than I would've expected. I believe this again happens because of the sparsity of the test dataset. If this testing set had a larger number of samples I would expect the accuracy to be lower and better represent the error of using the linear apporach on a non-linear dataset.

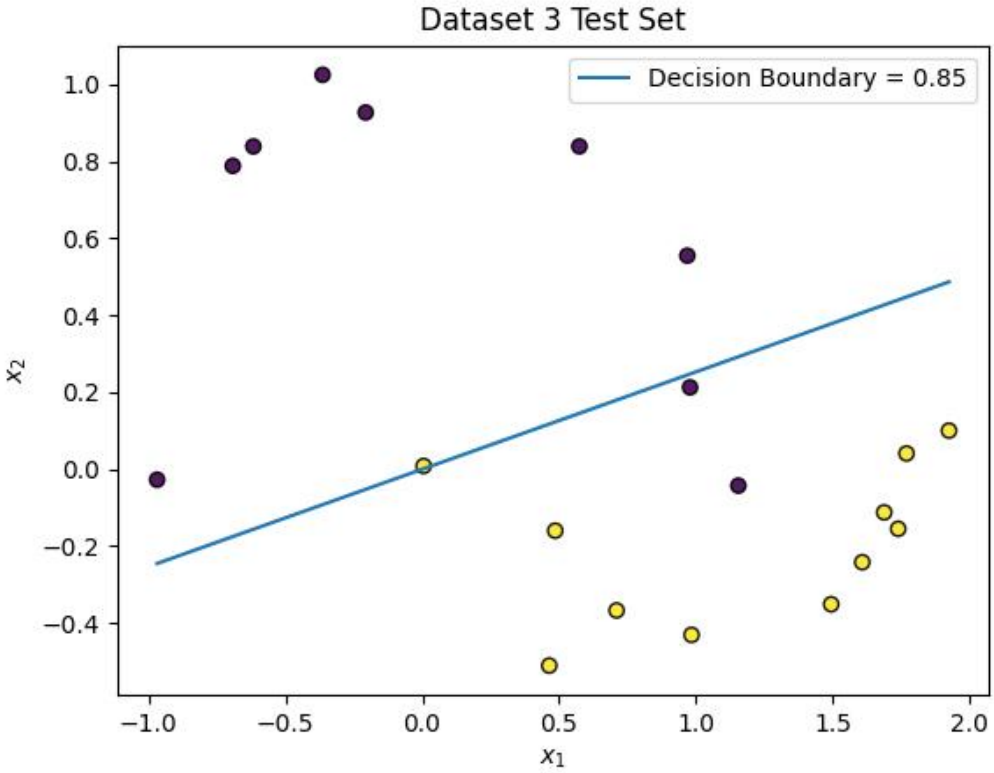

Figure 14. Dataset 3 Test Set Result

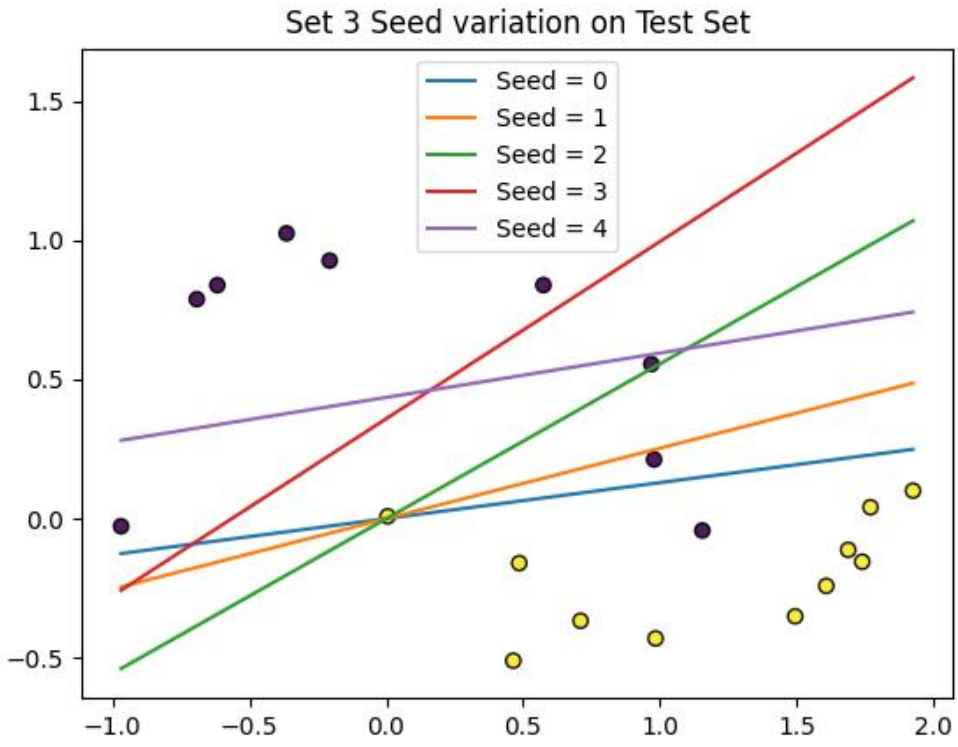

Figure 15. Dataset 3 Seed Variation

1.1.4 Dataset 4

With again a tol value of $5E-6$ dataset 2 auto stopped after 6 iteration. Because this dataset is highly non-linear it can be seen by the learning curve that it is never able to converge and fluctuates.

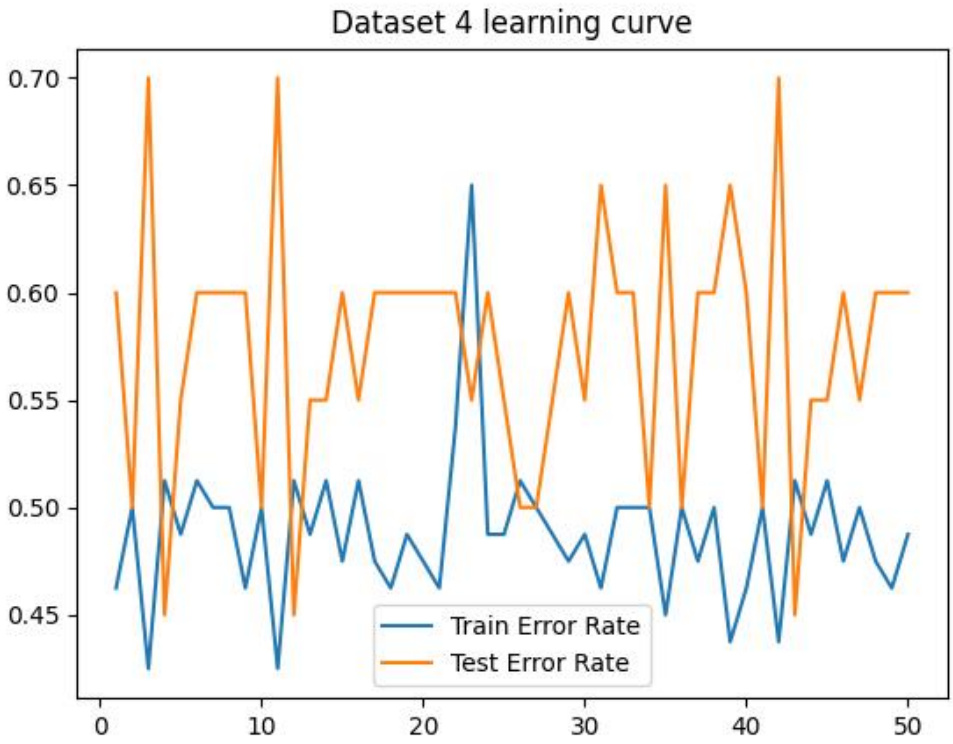

Figure 16. Dataset 4 PLA learning curve

As expected the learned decision boundary doesn't represent the underlaying dataset at all. As such it has very poor accuracy.

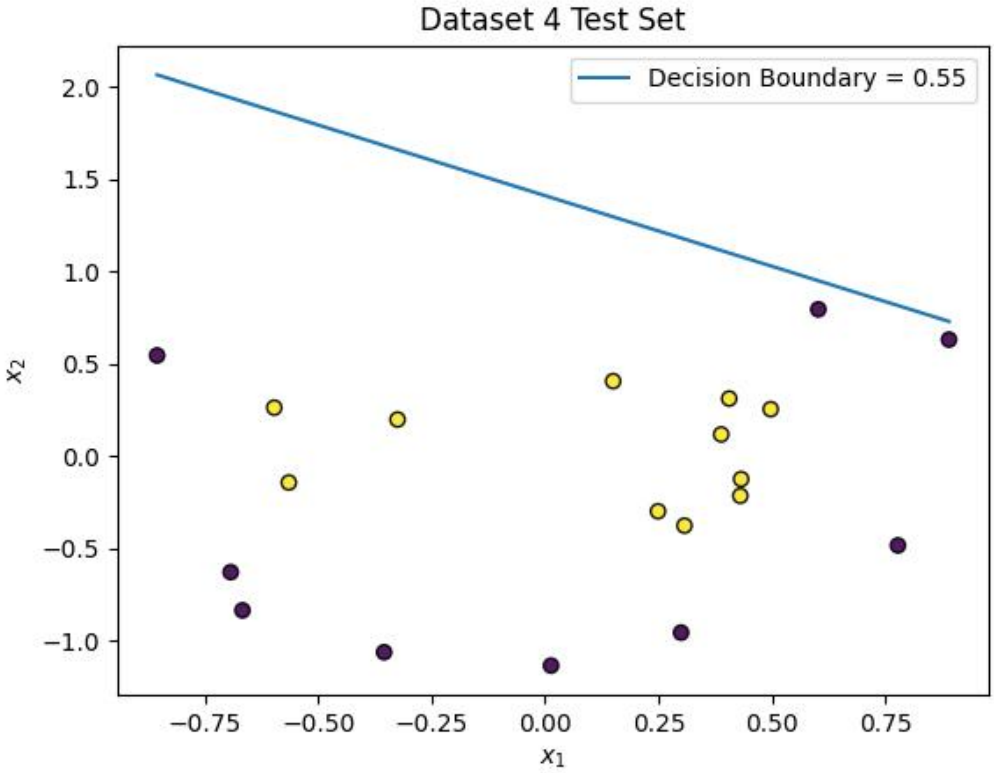

Figure 17. Dataset 4 Test Set Result

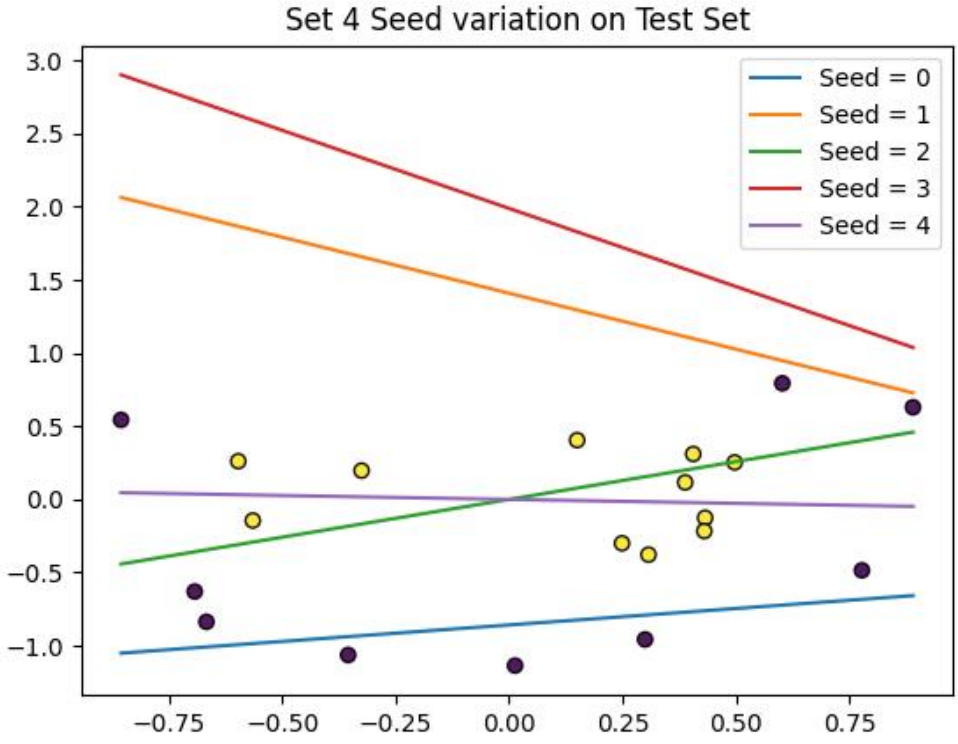

Figure 18. Dataset 4 Seed Variation

1.2 Exercise 2.2

1.2.1 Dataset 2

For dataset 2, because it is know that the underlaying dataset is defined by a second order polynomial, the features were cast into this produces a dataset with 6 features. The resulting classifier scored a 0.9 accuracy. The learned decision boundary is shown in *Figure 19*.

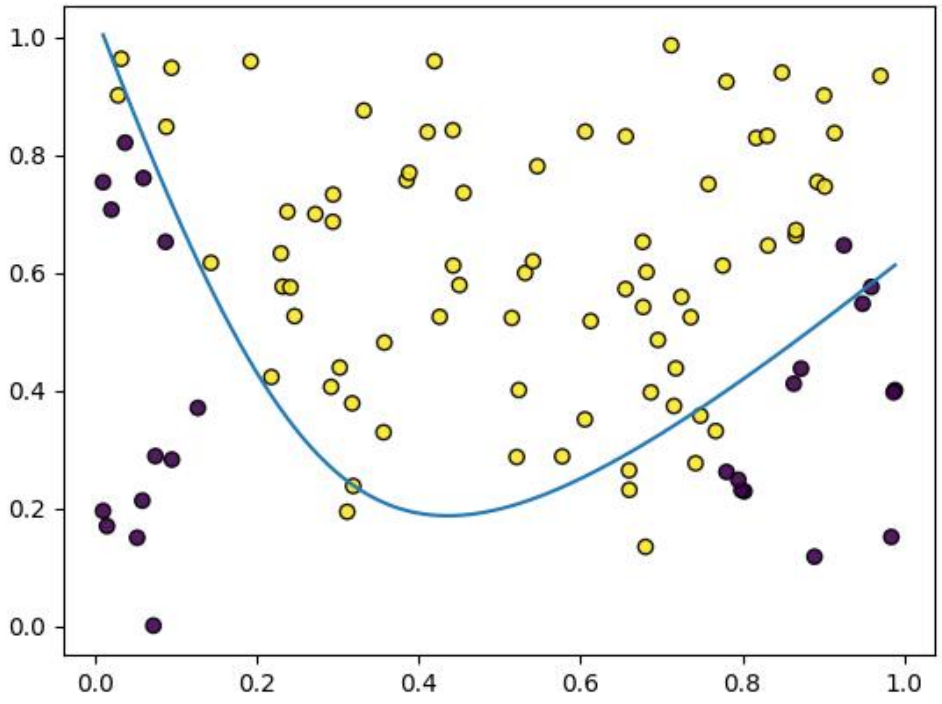

Figure 19. Dataset 2 2nd order polynomial

1.2.2 Dataset 3

For dataset 3 I applied a 3rd order poly nomial feature space resulting in 10 features. Once trained it was found to have an accuracy of 0.99.

1.2.3 Dataset 4

For dataset 4, because from prior knowledge we know the dataset is to circle with the same center point and different radii the ideal boundary would a circle centered at zero with a radius half way between the two class radii.

For such a circular desired boundary the only three features needed would be

$$z = [x_0, x_1^2, x_2^2]$$

When applying these features the learned classifier has an accuracy of 0.99.

1.2.4 Non-linear decision to linear

Applying non-linear features to the linear dataset I found that when using high order polynomials they would perform well, this is happening because the higher order polynomials can represent linear function by suppressing the high order portions. This is demonstrated by *Figure 20* which shows the learned decision boundary using second order polynomial features. It can be seen that

the resulting boundary is very nearly linear. While unplotted the high accuracy for the 3rd order polynomial boundary sugguests the same the same thing is happening.

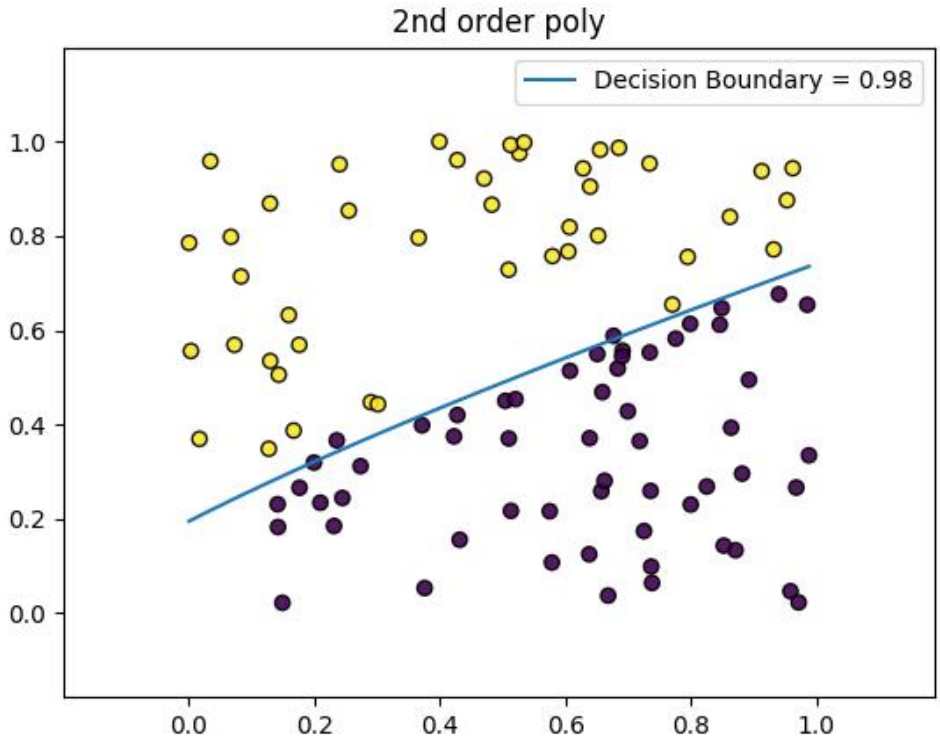

Figure 20. Dataset 1 2nd order polynomail

Other tested non-linear feature spaces was the circle boundary used by Dataset 4 and a sin/cos feature space, which shouldn't be able to learn a linear classifier. The z for the sin/cos feature spaced was

$$z = [x_0, \sin(x_0), \cos(x_1)]$$

The result of the 4 classifiers on the linearly separable dataset 1 were as follows | Feature Space | Accuracy | | ——— | ——- | | 2nd Poly | 0.98 | | 3rd Poly | 1.00 | | Circle | 0.98 | | Sin/Cos | 0.49 |