Perceptron Project

John Yap

1 Introduction

The purpose of this project was to use the Perceptron Algorithm to segregate a binary data set and to evaluate the accuracy and the factors that affect accuracy. The algorithm is based on the vector math shown in Figure 1 that can linearly separate binary data.

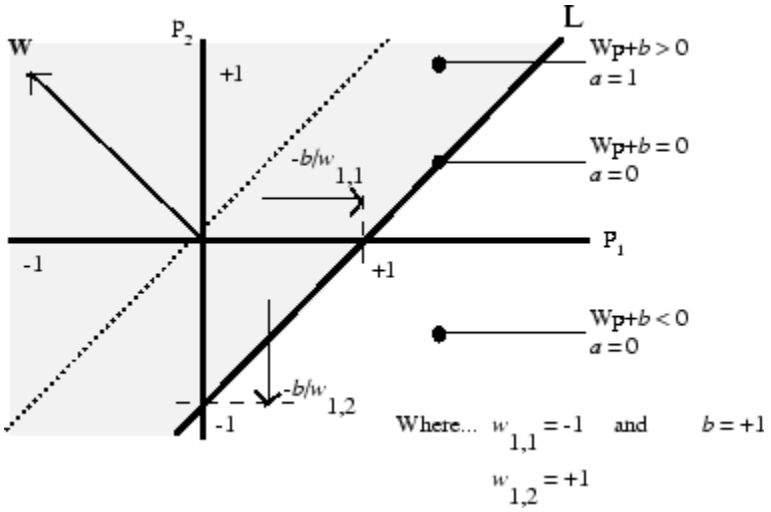

Figure 1: Vector Math (mathworks.com)

The weight vector, W, and the bias, b, create a line that when multiplied by a vector, p, will be either positive or negative depending on which side of the line the vector falls. The Perceptron then uses the Sign function to generate a binary label for the data. For this project, we ignored the bias since it can be integrated into the weight vector.

2 Training

The Perceptron algorithm generates a weight vector randomly and trains itself on a set of training data to find a weight vector that classifies all the points in the data set. For this project, our training data was generated by generating random points (x,y) such that x and y were between -10 and 10 and labeling the data using the line x = y.

For each training data point, the algorithm checks to see if the data is classified correctly and if not it modifies the weight vector based on the classified training vector and a learning rate.

3 Accuracy

The first way we determined accuracy is by comparing the weight generated by the Perceptron to the known line that we used to generate the training data, x = y. In Figure 2, we graph the error based on the dot product between the known weight vector and the trained weight vector vs the number of training data points in the training data set.

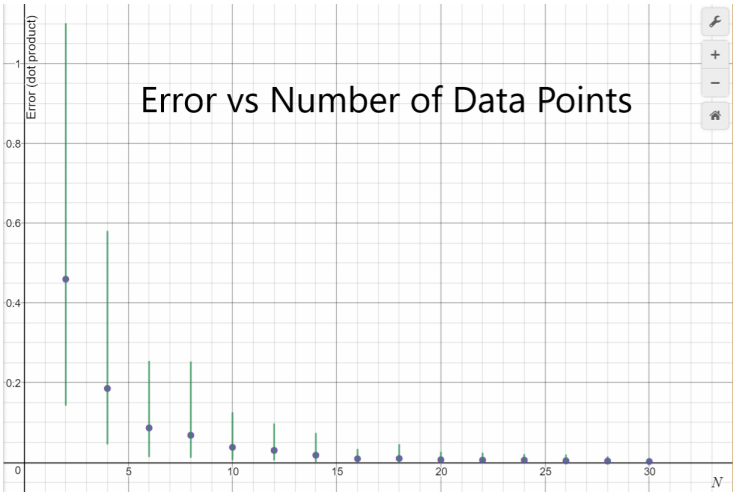

Figure 2: 100 Runs, Error is the dot product of the given W and the trained W.

This graph is in line with what we expect theoretically as the number of data points goes to infinity the Perceptron becomes more accurate.

The second way we determined accuracy was to test the trained weight vector against a single set of data generated with 500 data points.

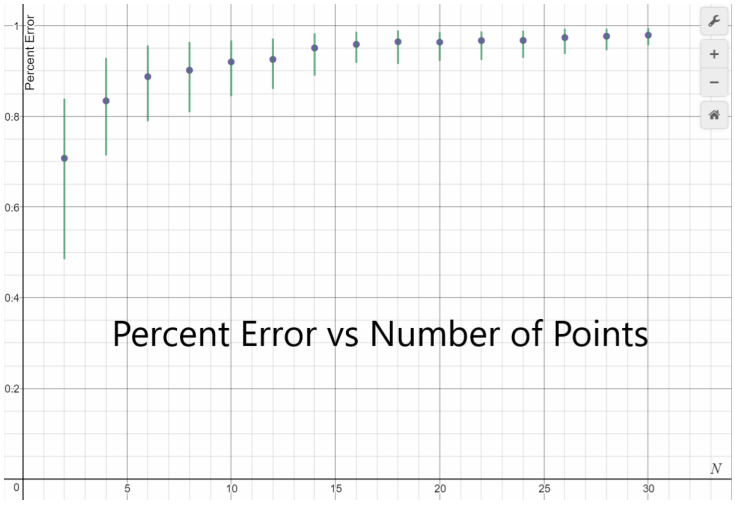

Figure 3: 100 Runs, trained W accuracy at predicting 500 data points.

As expected, the percent prediction accuracy of the Perceptron increases with the more data used to train it.

For both figures 1 and 2, we had finding a good measure of the deviation in the data from run to run. Using standard deviation didn't work because the error bars went past the possible range of data, which suggests that the error histogram would not be a normal curve. Figure 3 shows that the deviation is a half normal curve.

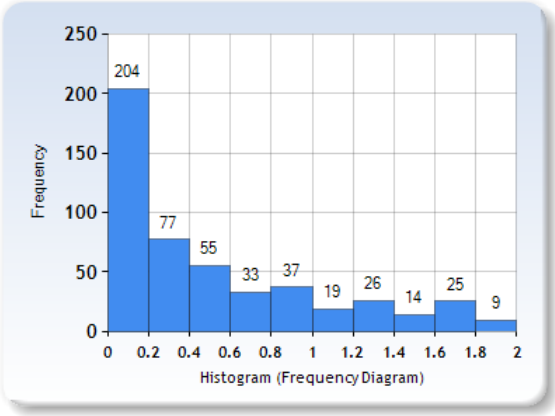

Figure 4: Error for 1000 Runs at 2 data points

At first, I tried to use the min and max as the error bars, but I felt that it was too susceptible to outliers. Instead, I took the average of all the data greater than the average for the positive error bar and the average of all the data lower than the average for the negative error bar, so that is how the error bars for figures 1 and 2 were generated.

Lastly, we explored the relationship between the distance from the closest point to the dividing line and the error.

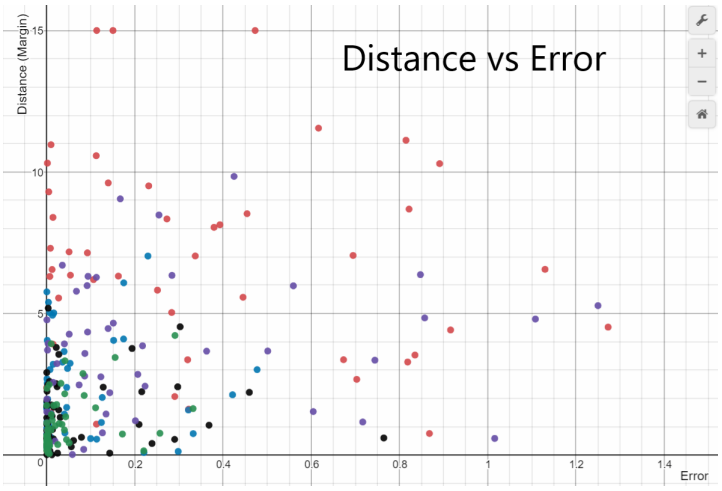

Figure 5: 100 Runs, closest distance, Red(N=2), Purple(4), Blue(6), Black(8), Green(N¿8)

To me this data shows that there is not a direct relationship between distance and error. The error tends to display the same distribution as Figure 3 and the distance. As the number of points increases, the problem just becomes more constrained and the same pattern just gets compressed toward the origin of the graph.