Introduction:

I think this assignment is about seeing which of the three algorithms, linear regression, logistic regression, or stochastic gradient descent, has the smallest error rate. In finding the smallest error rate, there are also some dependent factors in the logistic regression and stochastic gradient descent algorithms that may cause a higher or lower rate. For both algorithms, the learning rate, alpha, can be changed to find a suitable number for the data. Stochastic gradient descent can also change how many data points in the data set are used to learn and data points selected are completely randomized. Therefore, with both of these factors in mind, this homework is meant to find the smallest error rate using the most appropriate numbers for each element when finding the error rate of the data set and to see which algorithm is the most useful.

Method:

For this assignment, there were multiple methods used. The first part utilizes the linear regression algorithm, which takes in the data set X and the ground truths Y to create a 1x3 matrix by doing $(X^TX)^{-1}X^TY$. After that matrix is created, w_0 is set to the 1x3 matrix created from the linear regression algorithm to then be put into the perceptron learning algorithm. That algorithm takes in weights, this time initialized as the linear regression algorithm output instead of 0s, and goes through each data point in X, trying to find the mistakes and correcting them. Since the data is meant to be linear separable, we used the PLA algorithm so the loop will stop once the boundary line created from the weights has no errors. This algorithm is supposed to be more efficient and will have a lower error rate than initializing the weights to 0s in the beginning. The second algorithm used was for logistic regression. This algorithm used the sigmoid function, which took in a variable x and computed $e^x/(e^x + 1)$. In this algorithm, x was $(-y_n w_t^T x_n)$. The sig function was then multiplied by $(y_n x_n)$ where x_n represents the current vector in dataset X and y_n represents the current ground truth value in Y. After summing up the values of this function for each n, it was then averaged out by dividing the whole sum by the total number of data. The weights for logistic regression then added this new set of weights to it multiplied by alpha, which represented the learning rate. The algorithm repeated until the function was less than a certain value, which was found by using the algorithm $(1/N * \text{sum}(\ln(1 + \exp(-y_n w_t^T x_n))))$. For the stochastic gradient descent, the same algorithm as logistic regression was used, but instead of using N, it used K, which represented any number between 1 and N.

Experiments:

	Trial 1 ($\alpha = 0.3$, $K = 10$, $\varepsilon = 0.01$)	Trial 2 ($\alpha = 0.7$, $K = 10$, $\varepsilon = 0.01$)	Trial 3 ($\alpha = 0.7$, $K = 20$, $\varepsilon = 0.01$)	Trial 4 ($\alpha = 1$, $K = 20$, $\varepsilon = 0.01$)	Trial 5 ($\alpha = 1$, $K = 25$, $\varepsilon = 0.01$)	Trial 6 ($\alpha = 2$, $K = 25$, $\varepsilon = 0.01$)
Linear Regression Error Rate	0.025	0.025	0.025	0.025	0.025	0.025
$W_0 = 0$ PLA	0.1125 Total Iterations: 160	0.1 Total Iterations: 120	0.125 Total Iterations: 160	0.116667 Total Iterations: 120	0.13 Total Iterations: 240	0.05 Total Iterations: 80
$W_0 = w_{LinearRegression}$ PLA	0.0625 Total Iterations: 160	0.075 Total Iterations: 80	0.066667 Total Iterations: 120	0.07 Total Iterations: 80	0.075 Total Iterations: 160	0.05 Total Iterations: 160
$W_0 = 0$ Logistic Regression Error Rate	0.1	0.1	0.1	0.1	0.1	0.075
$W_0 = w_{LinearRegression}$ Logistic Regression Error Rate	0.05	0.05	0.05	0.025	0.025	0.0
Stochastic Gradient Descent Error Rate	0.125	0.075	0.125	0.1	0.075	0.0

$W_0 = w_{LinearRegression}$ - Black Line
$W_0 = 0$ - Green Line

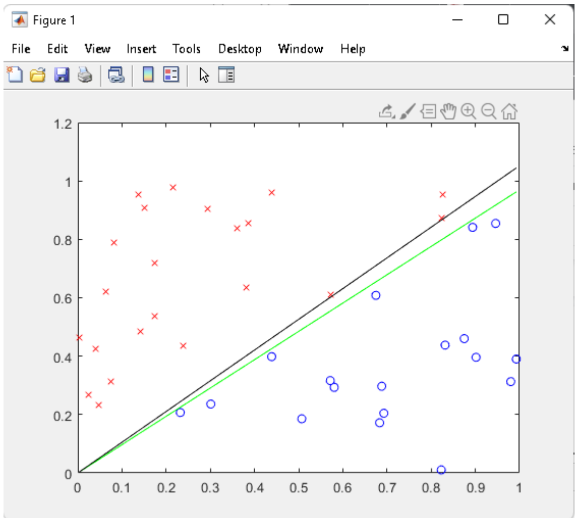
Figure 1
File Edit View Insert Tools Desktop Window Help
1.2 1 0.8 0.6 0.4 0.2 0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

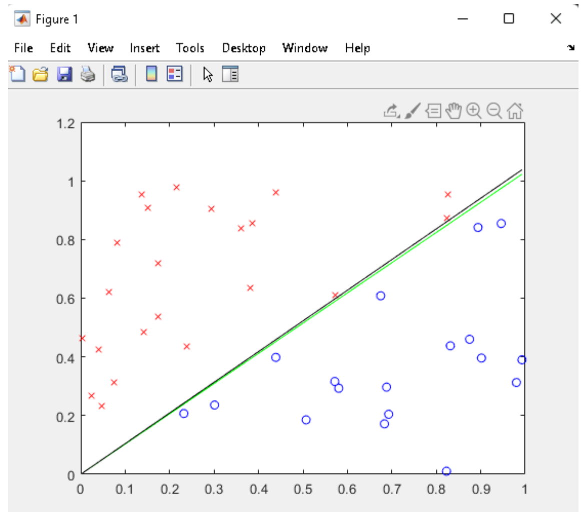
Figure 1
File Edit View Insert Tools Desktop Window Help
1.2 1 0.8 0.6 0.4 0.2 0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

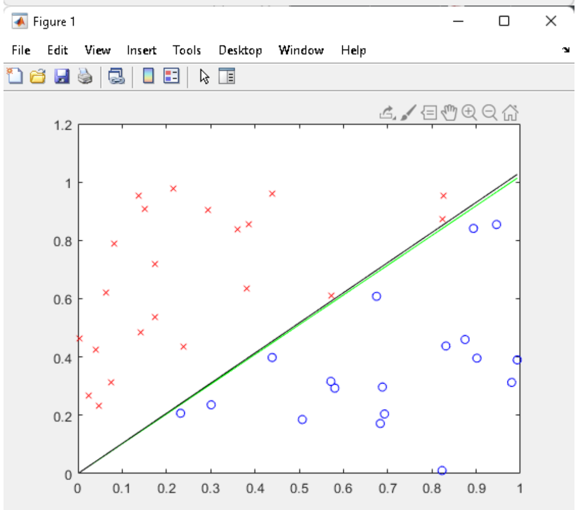
Figure 1
File Edit View Insert Tools Desktop Window Help
1.2 1 0.8 0.6 0.4 0.2 0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

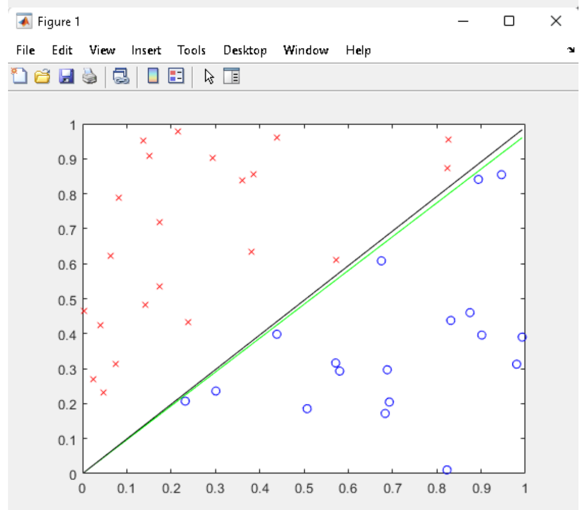
Figure 1
File Edit View Insert Tools Desktop Window Help
1 0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

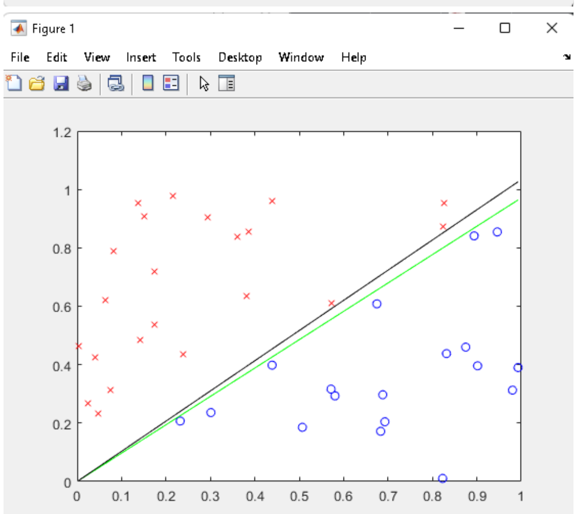
Figure 1
File Edit View Insert Tools Desktop Window Help
1.2 1 0.8 0.6 0.4 0.2 0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

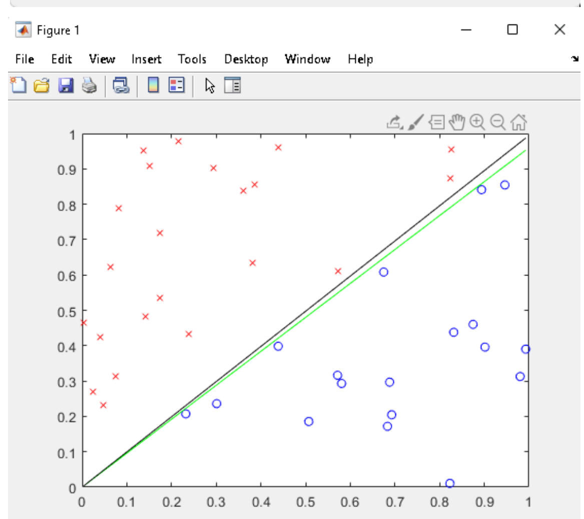
Figure 1
File Edit View Insert Tools Desktop Window Help
1 0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

Discussion:

As we can see in the tables above, w_0 initialized as $w_{LinearRegression}$ appears to have a lower error rate than $w_0 = 0$ consistently. However, in trial 6, both initializations had an error rate of 0.05, but that is mainly because it has to do with the random permutation that the for loop goes through each time. Because of that, there is a slight chance that the initialization of $w_0 = 0$ will have a really low error rate if the permutation works in its favor, which did seem to happen in that trial. The error rate after initializing $w_{LinearRegression}$ was only 0.025, which means that the Perceptron Learning Algorithm should be able to identify the errors quickly, but that completely depends on the random permutation that the for loop has to go through, which is why some error rates for PLA appear higher than they should be. Now for the logistic regression error rate, it looks like it is more effective than PLA when initializing w_0 to 0, to $w_{LinearRegression}$ or using Stochastic Gradient Descent. Starting with logistic regression when w_0 is initialized as 0 and $w_{LinearRegression}$, it appears that $w_{LinearRegression}$ has a lower error rate than 0 consistently. This makes sense since $w_{LinearRegression}$ had a very low error rate to begin with and it was just a matter of correcting a few small mistakes. When changing the learning rates (alpha), neither logistic regression error rates changed when going from 0.3 to 1. The error rate changed at 1 for $w_{LinearRegression}$, but it did not change for 0 until alpha became 2. However, it seemed that $w_{LinearRegression}$ was way more efficient since it reached an error rate of 0 when alpha became 2. Now for the Stochastic Gradient Descent, the error rates were a little erratic. From the information I have, it seems that a high K and high learning rate will produce optimal results. It makes sense since having a lot of information is better when wanting to be more accurate and having a high learning rate will have a greater effect when each error fixes its mistake.