hw2

Keith G. Williams 800690755

DSBA 6156 Spring 2016

Description of Algorithms and Design Decisions

`generateData(N)`

This function is used to create 2-D training data that are linearly seperable. A uniformly distributed sample of $2N$ points in $[-1,1]$ is created and reshaped to an $N \times 2$ matrix. Two points from this matrix are randomly selected to create the target line that seperates the N points. From these points, the slope and intercept are calculated, and each point in the training data is evaluated to be above the line. Those points above the line are labeled as 1 and -1 otherwise. This evaluation of each point relative to the target line is vectorized for efficiency.

`pla(X, Y, w0)`

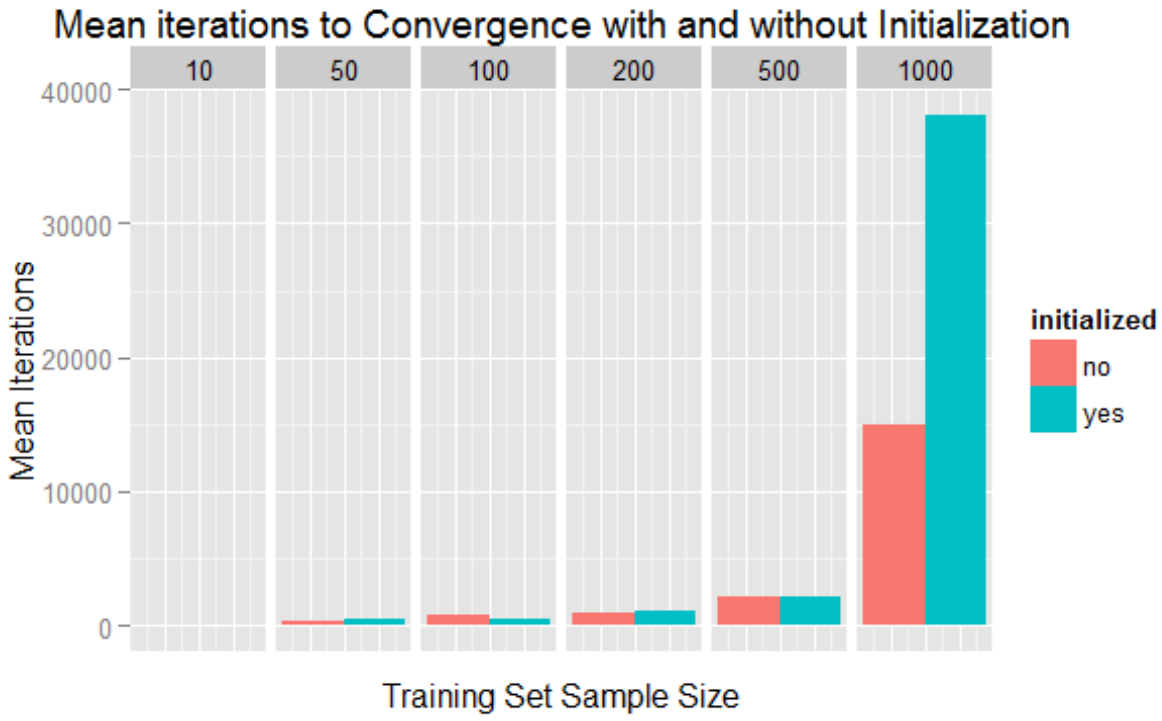
This function takes the randomly generated $m \times n$ training data and labels (X, Y), and learns the approximate target function. The learned function linearly seperates the training data X according to the labels in Y. By default, this algorithm initializes the weight vector w to the zero vector of length $n + 1$, though it can accept an initial weight vector output by `pseudoinverse`. The algorithm also preappends X with the bias term, a vector of ones of length m. The algorithm uses a boolean flag to check if the learned weight vector w has convereged. While the boolean flag is `False`, the algorithm checks if $w^T \cdot X$ is equivalent with Y. If they are equivalent, the boolean flag flips to `True`, and the algorithm returns the learned weight vector w and the number of iterations needed to converge. If the hypothesis and label vectors are not equal, a misclassified point X_i is randomly chosen, and w is updated to $w + X_i Y_i$. When the weight vector is updated, the number of iterations is incremented. A random point is chosen, so that the algorithm doesn't get stuck trying to correctly classify the same point repeatedly, when that point might get classified correctly in one iteration when a different point is used to move the weight vector.

`pseudoinverse(X, Y)`

This function takes the randomly generated $m \times n$ training data X and its continuous label vector Y, and returns the line of best fit that minimizes the squared error. First the algorithm preappends X with the bias term, a vector of ones of length m. Then the dot product of the pseudoinverse of X and X^T is returned. The pseudoinverse is the inverse of the dot product X and X^T.

Experiments

`pla` was tested on training samples of various sizes with and without an initialized weight vector w_0 from `pseudoinverse`. For each N in {10,50,100,200,500,1000} data was randomly generated 100 times, and the same random data was fed to `pla` with and without the initialized weight vector. The average number of iterations to convergence was calculated for each value of N with and without w_0. The summary results are tabulated and plotted below.



sample size	initialized	mean iterations	standard deviation	min	max
10	no	81.12	377.3641	1	3605
10	yes	57.95	288.0400	0	2735
50	no	399.13	1536.6634	3	11673
50	yes	512.25	2023.4614	0	15092
100	no	738.68	1851.4688	6	14249
100	yes	543.34	1173.0969	0	6975
200	no	949.67	2573.7885	16	17781
200	yes	1056.37	2752.5755	11	18915
500	no	2217.75	4523.7724	42	34561
500	yes	2083.09	3817.3186	75	28181
1000	no	14926.96	116690.4289	61	1167537
1000	yes	38122.36	346499.0311	115	3467864

From the graph, it is clear that the value of N increases the mean number of iterations exponentially, but it is less clear if the initialization has an effect on iterations. A clue that there might not be an effect is that the variance of iterations to convergence increases with N. To make a rational conclusion about the within group difference in means, a T-test is performed on the two groups of 100 trials for each value of N. The results are tabulated below.

sample size	uninitialized mean	initialized mean	lower bound estimated difference	upper bound estimated difference	p value
10	81.12	57.95	-70.48808	116.8281	0.6260817
50	399.13	512.25	-614.39489	388.1549	0.6566880
100	738.68	543.34	-237.37689	628.0569	0.3740893
200	949.67	1056.37	-849.86139	636.4614	0.7773652
500	2217.75	2083.09	-1032.81148	1302.1315	0.8202775
1000	14926.96	38122.36	-95578.56008	49187.7601	0.5270078

The table gives a lower and upperbound on the 95% confidence interval of the difference in means between the two groups as well as the p value. There does not appear to be a statistically significant difference in the mean iterations to convergence with or without initializing the weight vector with linear regression. All of the confidence intervals include 0, and the p values are all much greater than 0.05, so the null hypothesis - that the true difference in means is zero - cannot be rejected.

It is likely that initializing the weight vector had no effect on the number of iterations, because the `pla` algorithm was guaranteed to converge. So, even a small deviation from linear separation might take several iterations to correct, because small errors will lead to small adjustments in the decision boundary. (See appendix for visual). The initialized weight vector might be more useful for the pocket algorithm, in the case where linear separation is not possible, but the algorithm might get closer to convergence if it starts closer.

Appendix

Below is a visualization of the randomly generated data for $N = 100$. The points are colored by their true labels. The green line is the decision boundary learned by `pla`, and the black line is the decision boundary initialized by `pseudoinverse`. One can see that the initial decision boundary is close to the decision boundary, but this does not guarantee fewer iterations, since small errors will lead to small adjustments.

1 Decision Boundaries pla vs pseudoinverse, N=100