Algorithm

Input:
X: N*2 vector of points in coordinate plane
Y: N*1 vector of labels +1 or -1
w0(Optional): 3*1 vector of initial weights for pla generated by Linear Regression
Output:
w: Weight vector of dimension 3*1
iters: Number of iterations for pla to converge
Algorithm pla(X, Y, w0)

1. Check if w0 exists
   1.1 If exists w <- w0
   1.2 Else w <- $\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$
2. Iter <- 0
3. Add a column at 1st index of X. Set all the values of that column to 1. This acts as the bias term. This makes X a N*3 vector
4. Repeat infinitely
   4.1 Loop through all rows in X and check if sign(X(i)*w) != Y(i). If its equal set result(i) <- 0 else result(i) <- 1 i.e. set 1 if point is misclassified according to current weights.
   4.2 Set sampleIndices <- index of non-zero elements in result. sampleIndices now contains indices of all misclassified points in X.
   4.3 If sampleIndices is empty break the loop at line 3.
   4.4 sampleIndex <- randomly picked up number from sampleIndices.
   4.5 w <- w + transpose(Y(sampleIndex) * X(sampleIndex))
   4.6 Iter <- Iter + 1

Input:
X: N*2 vector of points in coordinate plane
Y: N*1 vector of labels +1 or -1
Output:
w: Weight vector of dimension 3*1
Algorithm pseudoinverse(X, Y)

1. Xdagger <- pseudoinverse(X)
2. w = Xdagger*Y

Input:
N: Number of points in coordinate plane to generate
Output:
X: N*2 vector of points in coordinate plane $[-1, 1] \times [-1, 1]$
Y: N*1 vector of labels +1 or -1
Algorithm generateData(N)

1. Generate X by picking 2N points in the interval $[-1, 1]$
2. Pick up two points A & B in the plane $[-1, 1] \times [-1, 1]$ randomly
3. Y(i) <- sign((B(1,1) - A(1,1)) * (X(i,2) - A(1,2)) - (B(1,2) - A(1,2)) * (X(i,1) - A(1,1)));

Algorithm main()

1. For N <- [10, 50, 100, 200, 500, 1000]
    1.1. For trial <- 1 to 100
        1.1.1. X, Y <- generateData(N)
        1.1.2. w, iters <- pla(X,Y)
        1.1.3. w <- pseudoinverse(X, Y)
        1.1.4. w, iters <- pla(X, Y, w)

Discussion

1. In the case where linear regression gives the initial weights of the line pla, generally converges in lesser number of iterations than running pla with weights initialized to zero since it approximately gives the orientation of classification line. But since we are randomly picking up points that are not correctly classified and adjusting the weights the line can be adjusted in such a manner that it is also probable that pla with weight initialization takes more iterations than pure pla.
2. The number of iterations generally increases with the sample size in both cases. In this case as well the results may vary since the sequence of adjusting the weights varies per iteration and the algorithm can converge in much lesser or much more iterations than expected. The number of iterations for same sample size also vary considerable.
3. The time taken for execution is directly proportional to the number of iterations taken by pla in both cases.
4. Since in our case the points are linearly separable pla is guaranteed to converge.
5. The line generated by the weights returned by linear regression correctly classifies majority of the points but does not guarantee to classify all since it tries to minimize the squared error. Pla guarantees to classify all points if they are linearly separable.

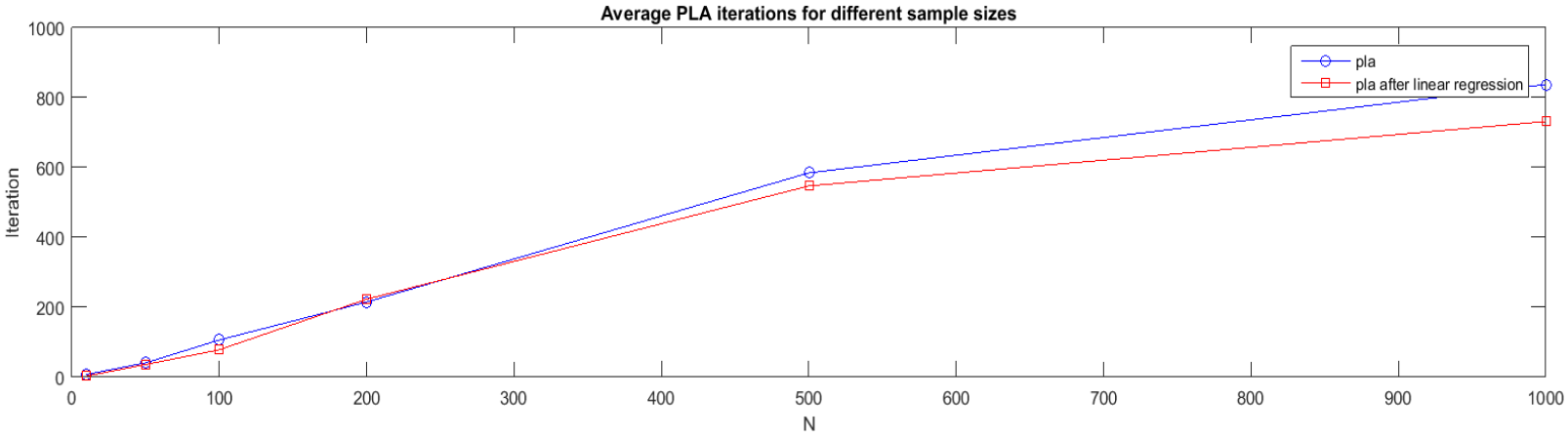
Plots

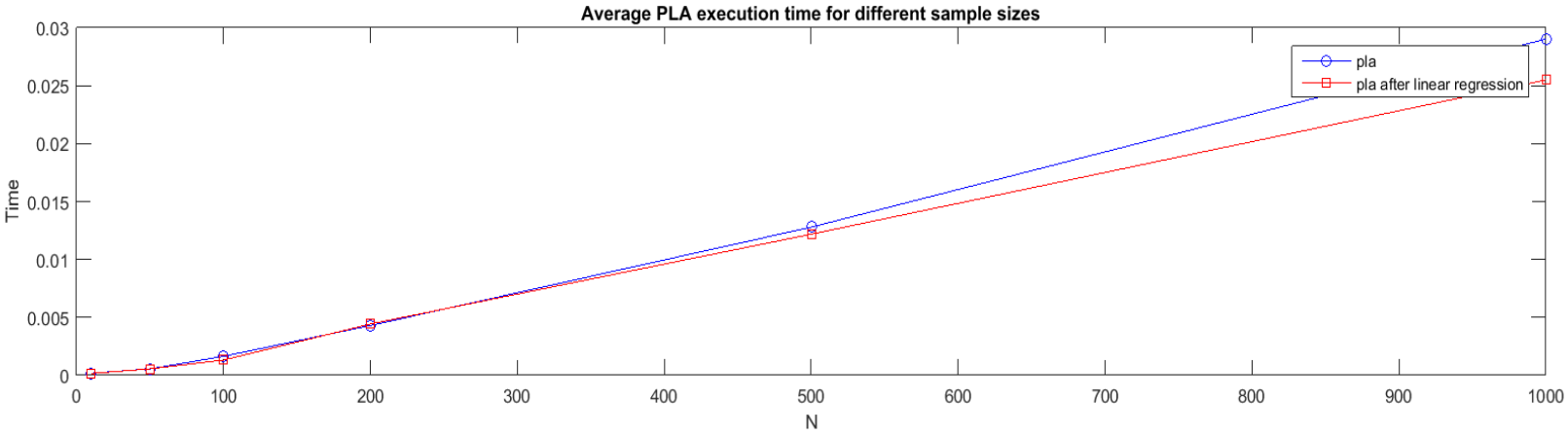
Following graphs are for this output:

N	PLA iterations	PLA iterations after Linear Regression
10	8	3.14
50	41.04	36.26
100	106.44	78.51
200	215.1	223.16
500	584.59	547.28
1000	837.49	731.5

Total time taken for execution = 9.265

Iterations vs N and time vs N for PLA and PLA after weight initialization by Linear Regression



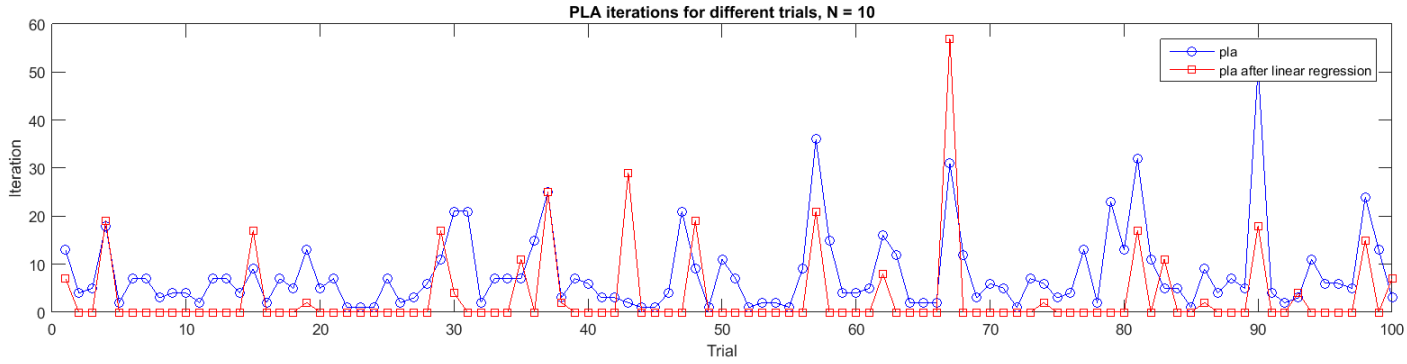


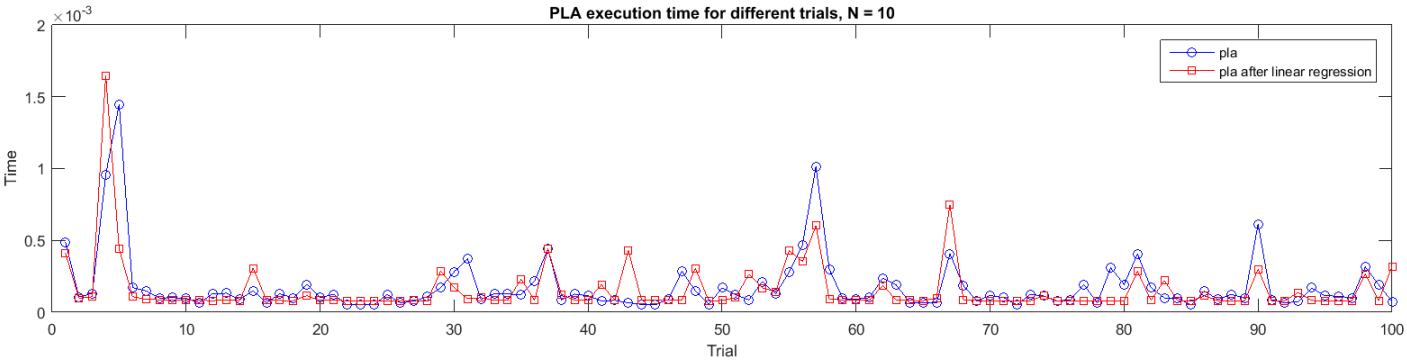
As we can see from the graph the number of iterations and the execution time generally reduces after weight initialization with Linear Regression. But sometimes it also gives a higher value.

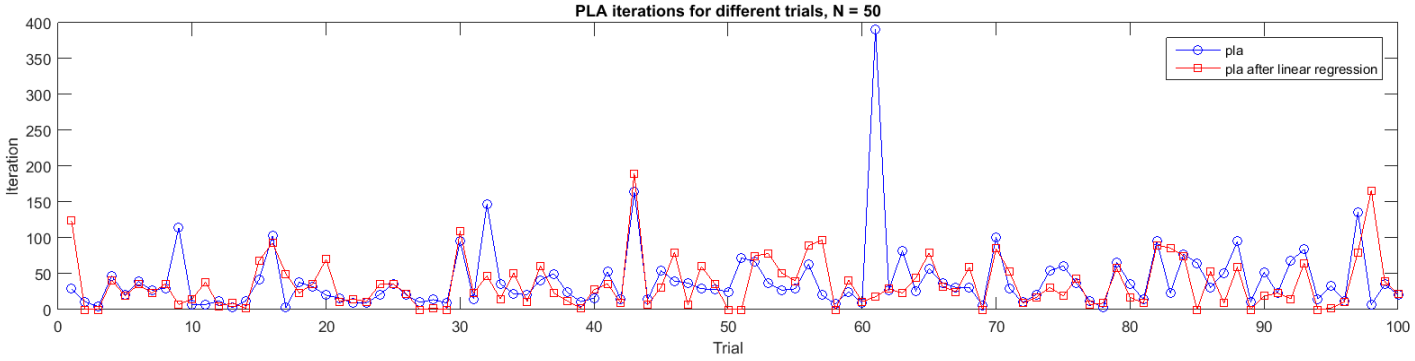
Also the number of iterations generally increases with sample size. But during different runs of the program I saw the iterations drop sometimes with increase in N.

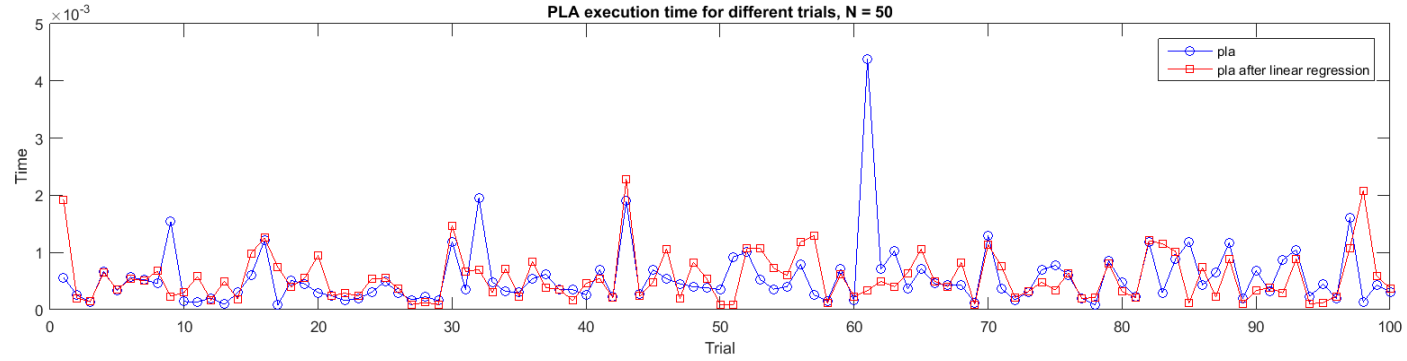
The time taken for execution is directly proportional to the number of iterations taken by pla.

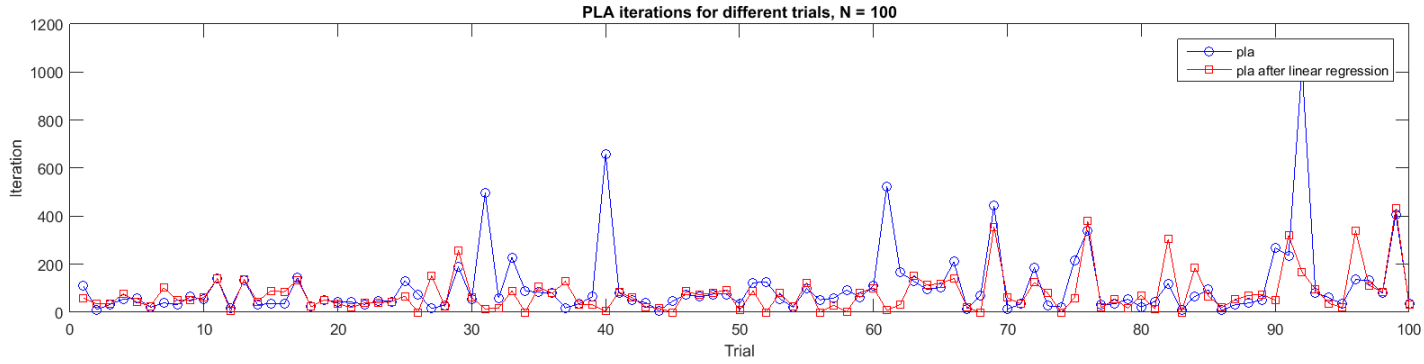
Iterations vs trial and time vs trial for PLA and PLA after weight initialization by Linear Regression

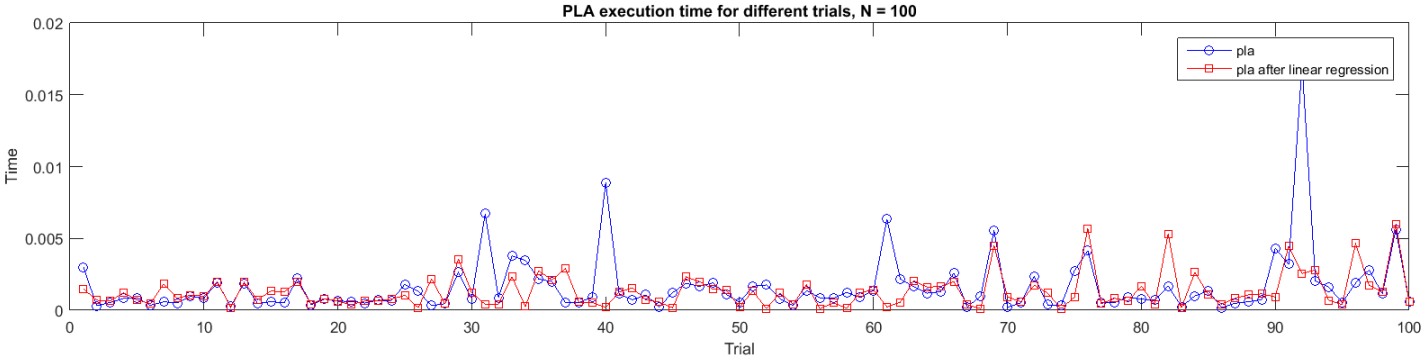


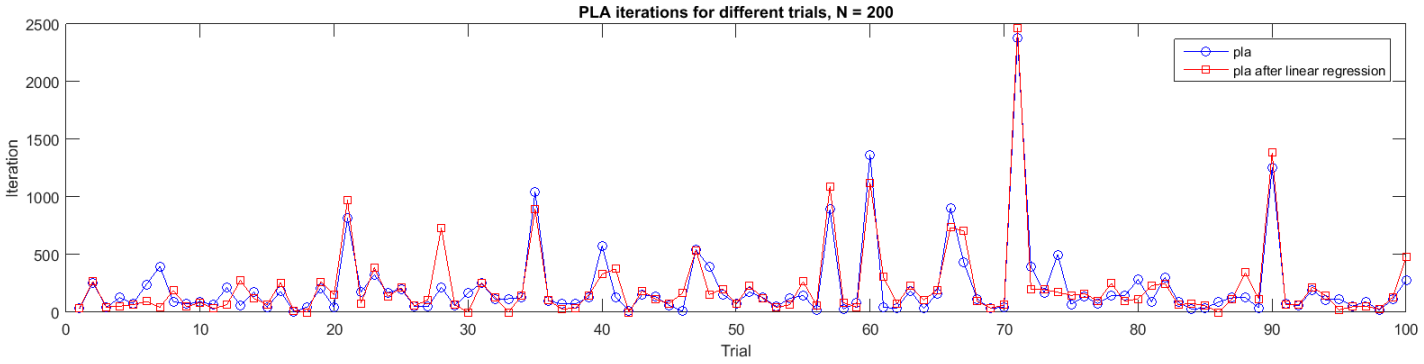


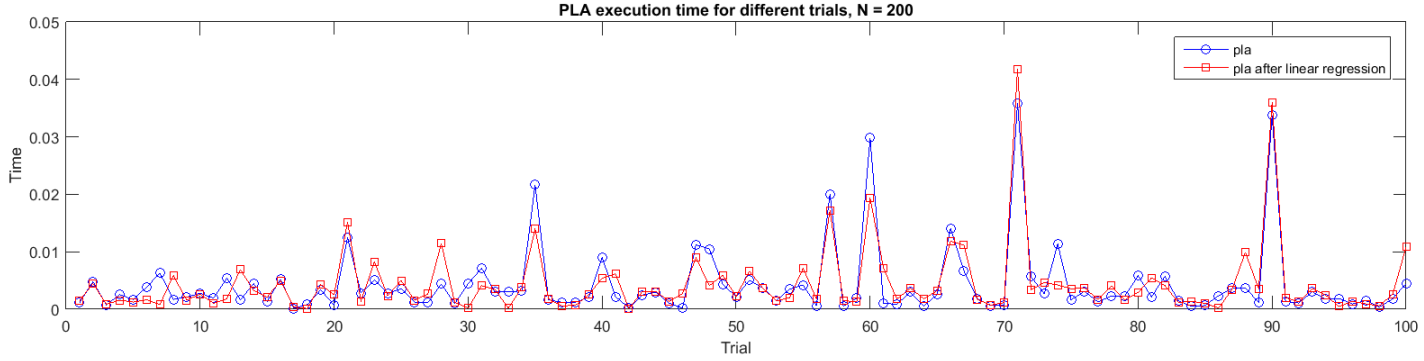


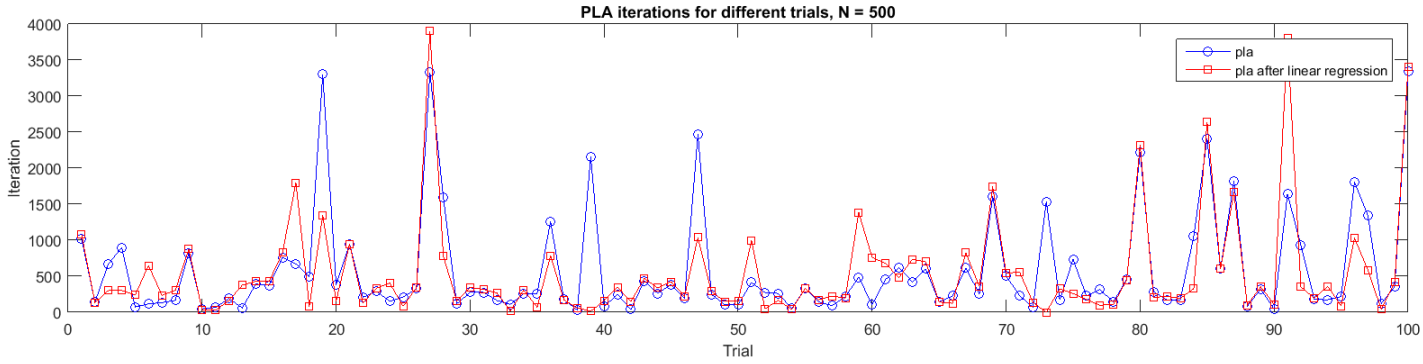


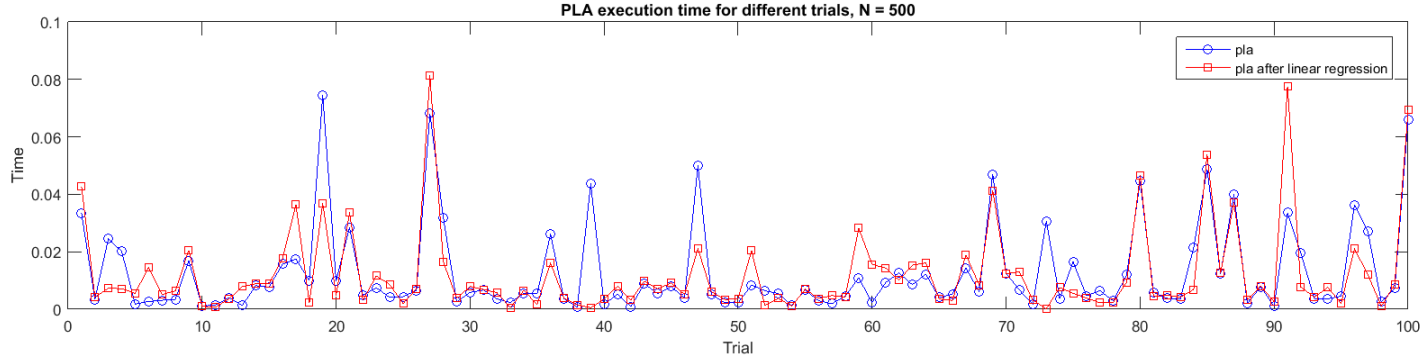

PLA iterations for different trials, N = 100
Iteration
Trial
pla
pla after linear regression

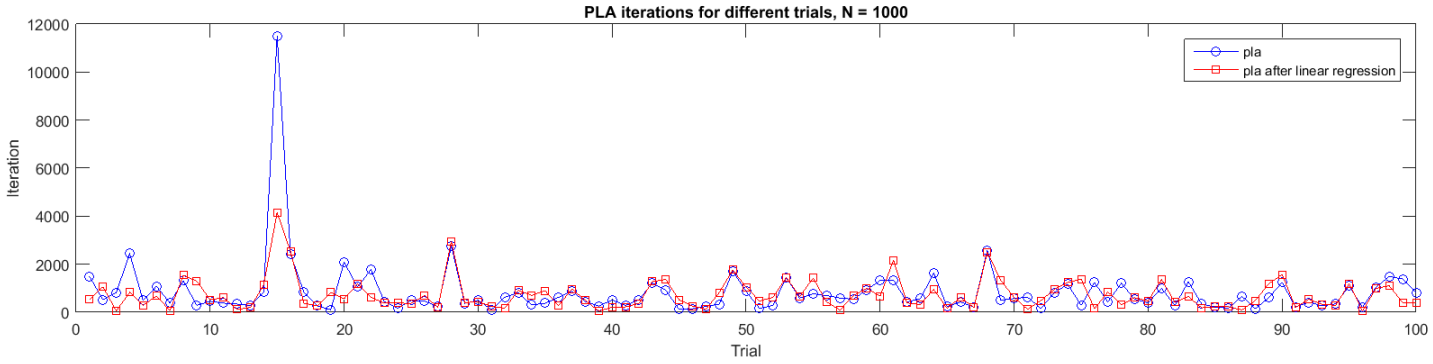

PLA execution time for different trials, N = 100
Time
Trial
pla
pla after linear regression

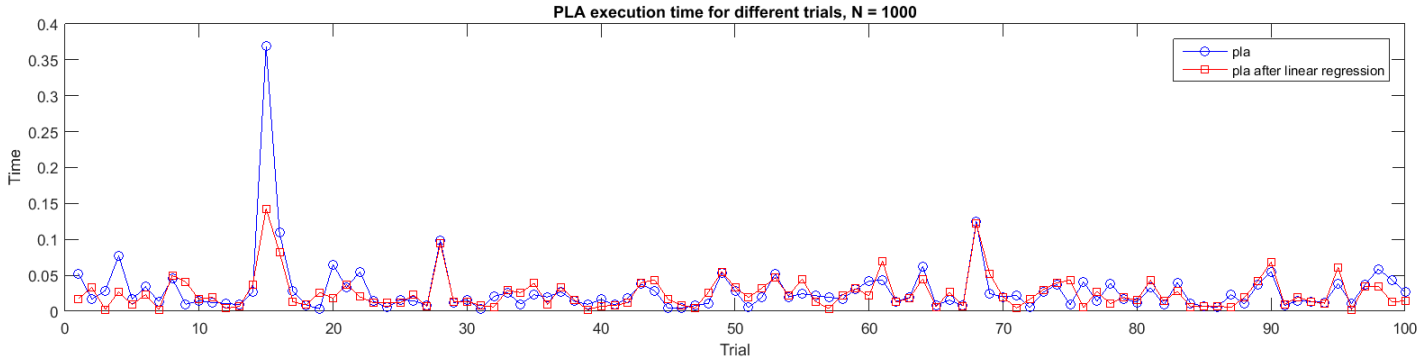

PLA iterations for different trials, N = 200
Iteration
Trial
pla
pla after linear regression


PLA execution time for different trials, N = 200
Time
Trial
pla
pla after linear regression

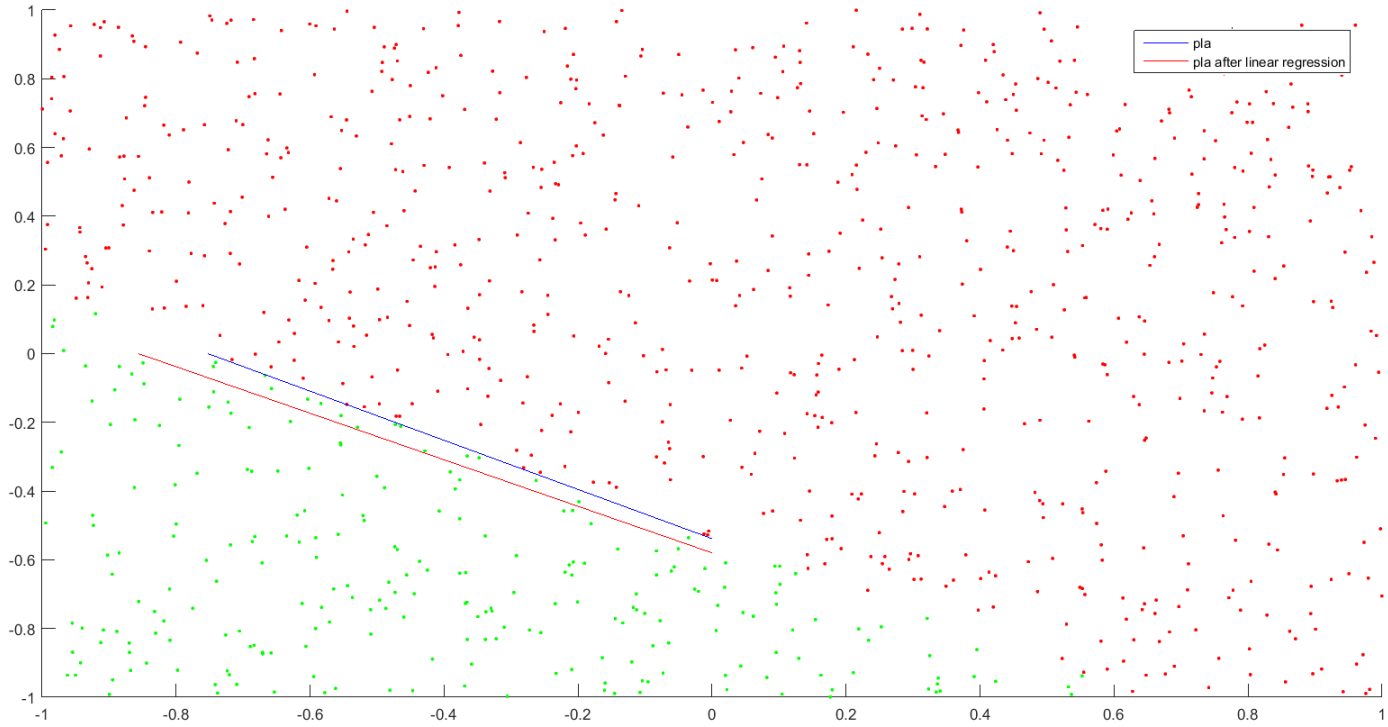

PLA iterations for different trials, N = 500
Iteration
Trial
pla
pla after linear regression


PLA execution time for different trials, N = 500
Time
Trial
pla
pla after linear regression


PLA iterations for different trials, N = 1000
Iteration
Trial
pla
pla after linear regression


PLA execution time for different trials, N = 1000
Time
Trial
pla
pla after linear regression

Linear Regression Line vs PLA Line



The line generated by the weights returned by linear regression correctly classifies majority of the points but does not guarantee to classify all since it tries to minimize the squared error. Pla classify all points correctly since they are linearly separable.