Taylor Berger
Neural Networks
Homework 4: Perceptron and LMS Learning

Perceptron Learning

Note: Code can be viewed at https://github.com/teberger/neural_networks/tree/master/homework4

Initial Conditions

Unless otherwise stated, the initial conditions for the Perceptron learning algorithm were the follow:

Perceptron weights = (0, 1)

Eta = 0.1

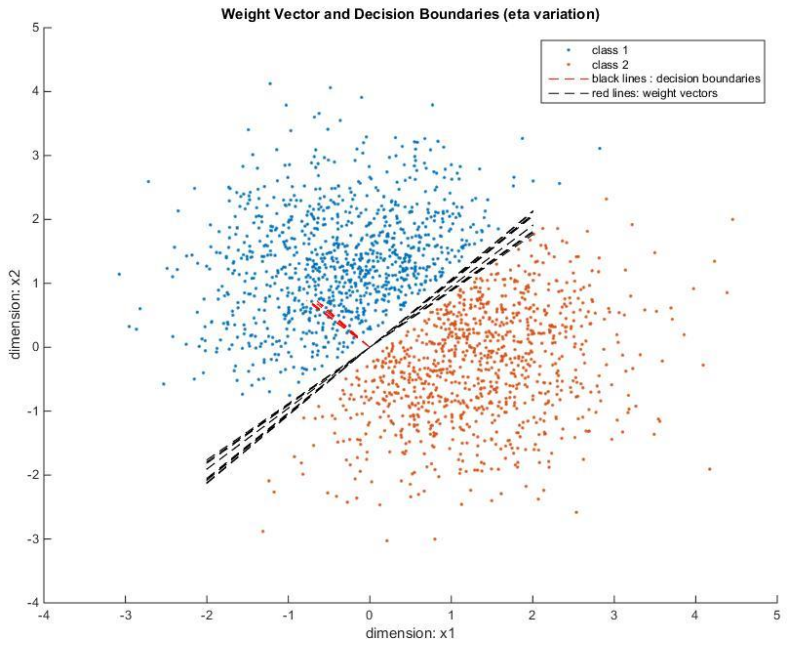
Presentation order: all of class 1 then all of class 2

All stopping criteria was based on the number of errors in the training set being equal to 0.

The Effect of η Variations

Eta affected the convergence of the Perceptron learning rule very little, but that could have been due to the initial perceptron weight selection. In the graph below, eta was varied from 0.001 to 0.101 in a total of 0.01 steps (for a total of 10 variations). In the end, most of the learning converged in one or two iterations of the algorithm. However, this could have been due to the initial value of the perceptron weights (as we see in section two).

The graph below shows the decision boundaries (black dashed lines) for all variations of eta that were tested. I would presume that lines closer to the bottom left of the class one data are the lines that correspond to the smaller values of eta since the correction was smaller during those iterations.

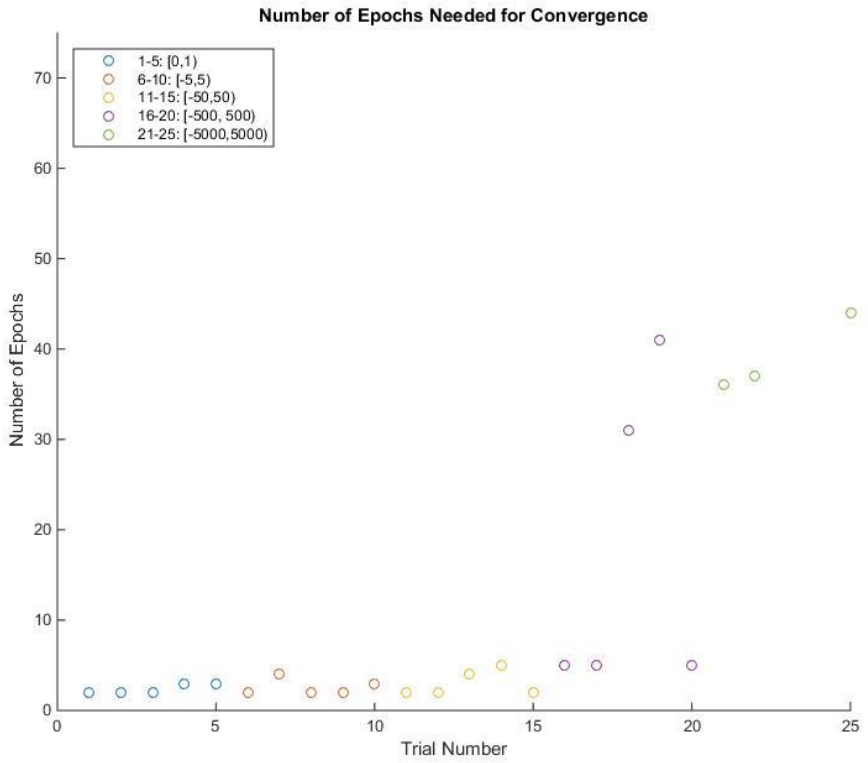


Taylor Berger
Neural Networks
Homework 4: Perceptron and LMS Learning

The Effect of Initial Weight Distributions

The initial weight distributions had a considerable effect on the number of epochs required for convergence. I found that as you increased the length of the initial weight vector, it becomes harder and harder move the vector into the right position thereby increasing the number of epochs needed for convergence.

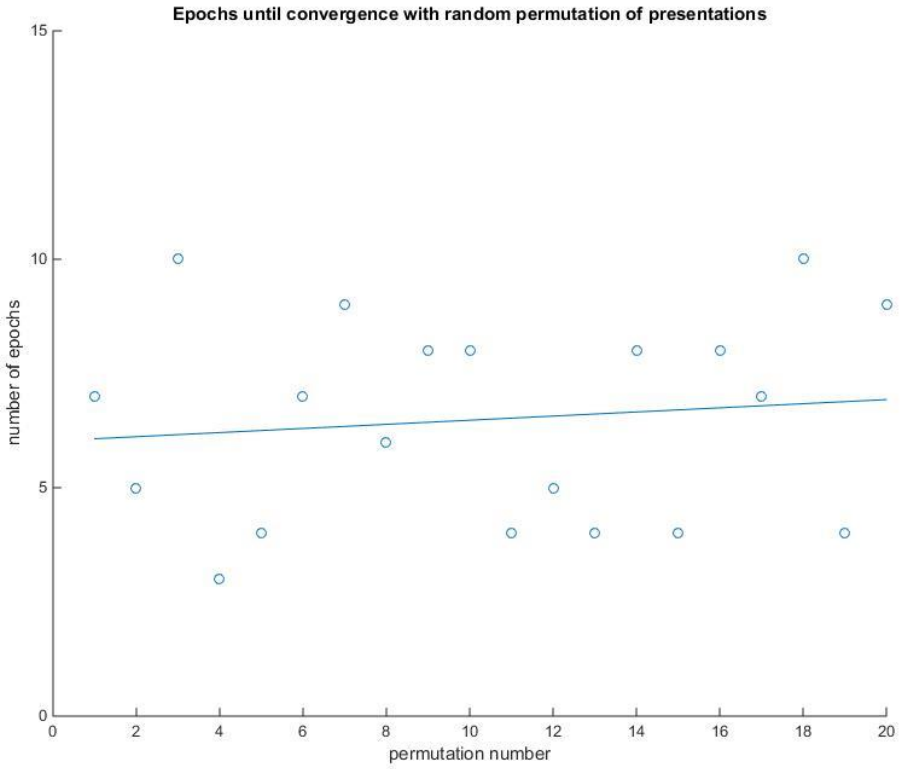
The graph below shows five samples of uniformly distributed random numbers over a specified range. There are two points excluded at 23 and 24 – their values are 291 and 406 respectively. We note that as the range increases, so does the number of epochs needed for convergence. This is because the angle between the old position of the weight vector and new position of the weight vector after correction becomes very small when the weight vector is large.



Order of Presentations

The order of presentations in the Perceptron learning algorithm turned out to not matter a whole lot. For this experiment, I set the perceptron's initial weights to be a uniformly random number in the range of [0, 100) so we had some variation in the number of epochs needed to converge instead of the average of two epochs for any of the lower weights. The random permutations of the presentations are graphed below, with each presentation graphed on the x axis and the number of epochs needed for convergence were plotted on the y axis.

We can note that the average across the graph is about the difference between the maximum (10) and minimum (3) divided by two. Since all presentation orderings show approximately the same number of epochs for convergence, we can say that this does not affect the number of epochs needed to converge – unlike the initial configuration of the weight vector.



Delta Rule

Initial Conditions

For the Delta Rule learning algorithm the following initial conditions were used unless otherwise stated:

P = (0, 0)

Eta = 0.0001

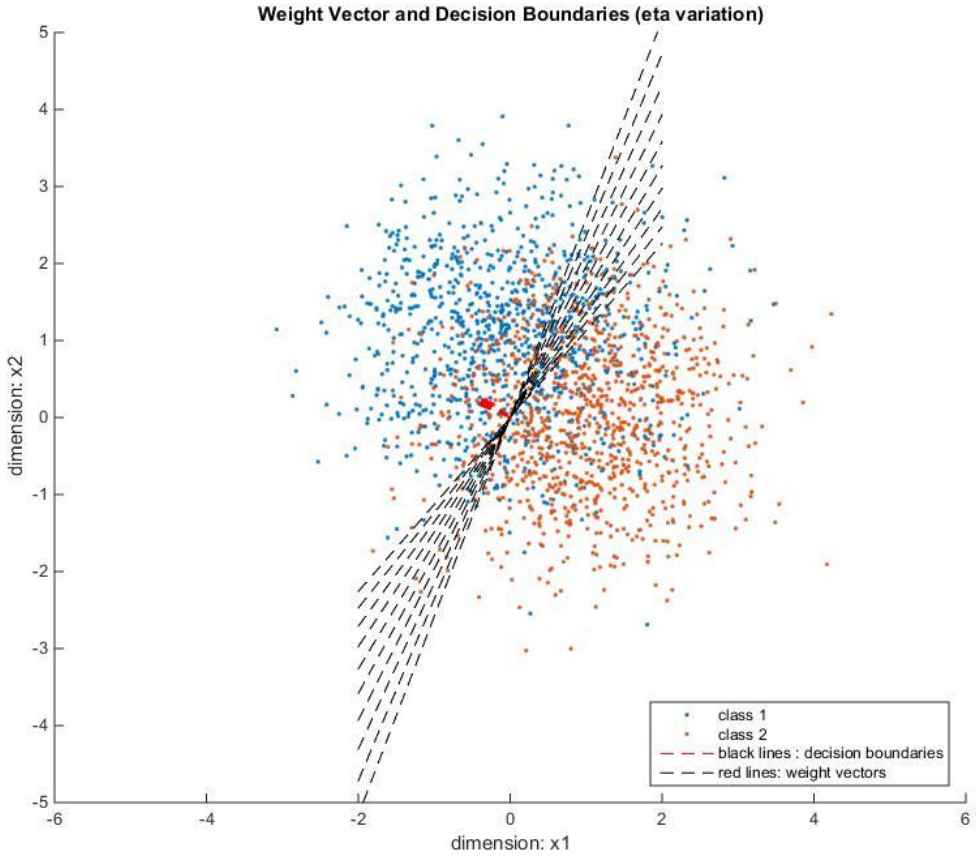
Presentation Order = all of class 1 then all of class 2

Stopping criteria = (sum of all squared errors at step n) – (sum of all squared errors at step n-1) < 0.005

The Effect of η Variations

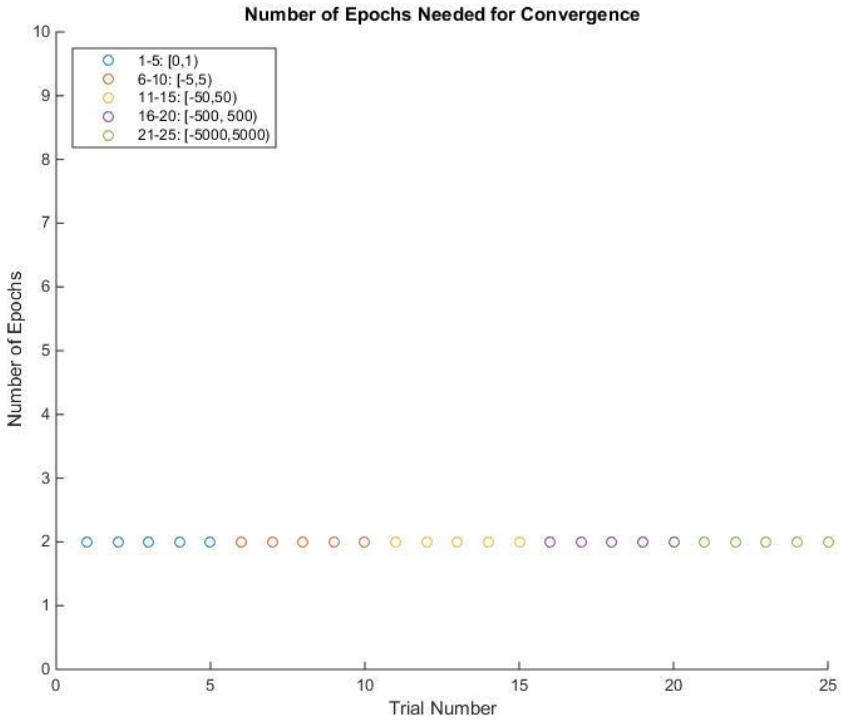
Eta variations mattered very little when using this algorithm. I varied eta from 0.0001 to 0.001 in steps of 0.0001 for a total of 10 variations. All variations converged in two epochs and yielded the following decision boundaries as seen in the graph below. The variation in the decision boundaries is more than

likely due to the presentation order since all errors in class 1 would pull it in one direction and then all errors in class 2 would pull it immediately back. However, they all seem to converge around the y = x line.



The Effect of Initial Weight Distributions

Again, with this algorithm, the initial weight distributions had no effect on the number of epochs needed to converge. The graph below shows that no matter the initial weight distribution, the algorithm converges in two epochs. This is due to the decision boundary changing with respect to the gradient of the squared error which would be large if the weight vector very far off from the desired value.



Order of Presentations

Just like the other analyses, the order of presentations revealed that they do not matter in terms of the number of epochs it takes to converge. The last figure show that no matter the presentation, the algorithm still converges at two epochs.