Analysis of the Perceptron and the LMS Neurons

Matthew A. Letter

October 7th 2014

Neural Networks 547

Perceptron

The perceptron is an algorithm for supervised classification of an input into one of several possible non-binary outputs. It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector. The algorithm allows for online learning, in that it processes elements in the training set one at a time (“Perceptron”, 2014).

Java was used to code the perceptron and the matplot java API was used to plot the data into a presentable format automatically at the end of each test. The perceptron was given a set of learning data from two linearly separable classes. The testing data was made up of two non-linearly separable classes. The stop criterion for learning was when an epoch occurred without encountering any errors. The RMS error calculated for the testing class was 0.9596462478022414. Below are three plots: before learning w/ learning data(Figure 1), after learning w/ learning data(Figure 2), and one with the PDR testing data(Figure 3).

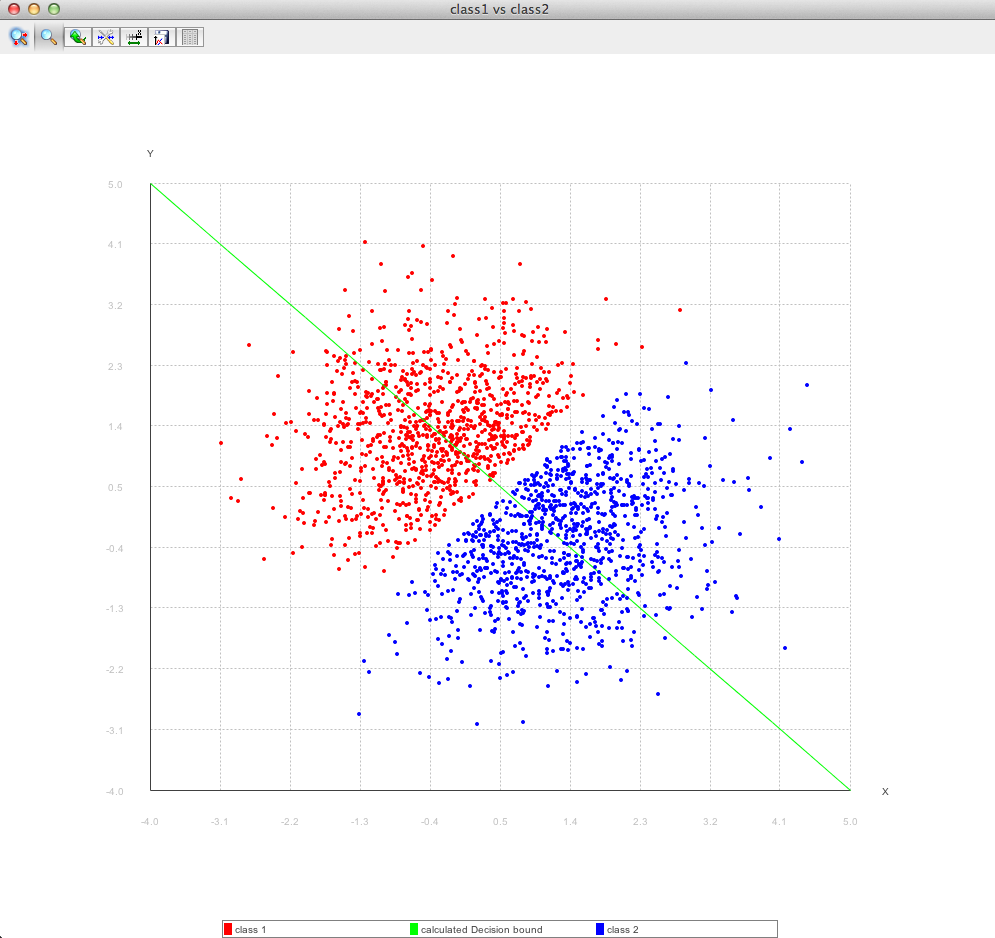


Figure 1

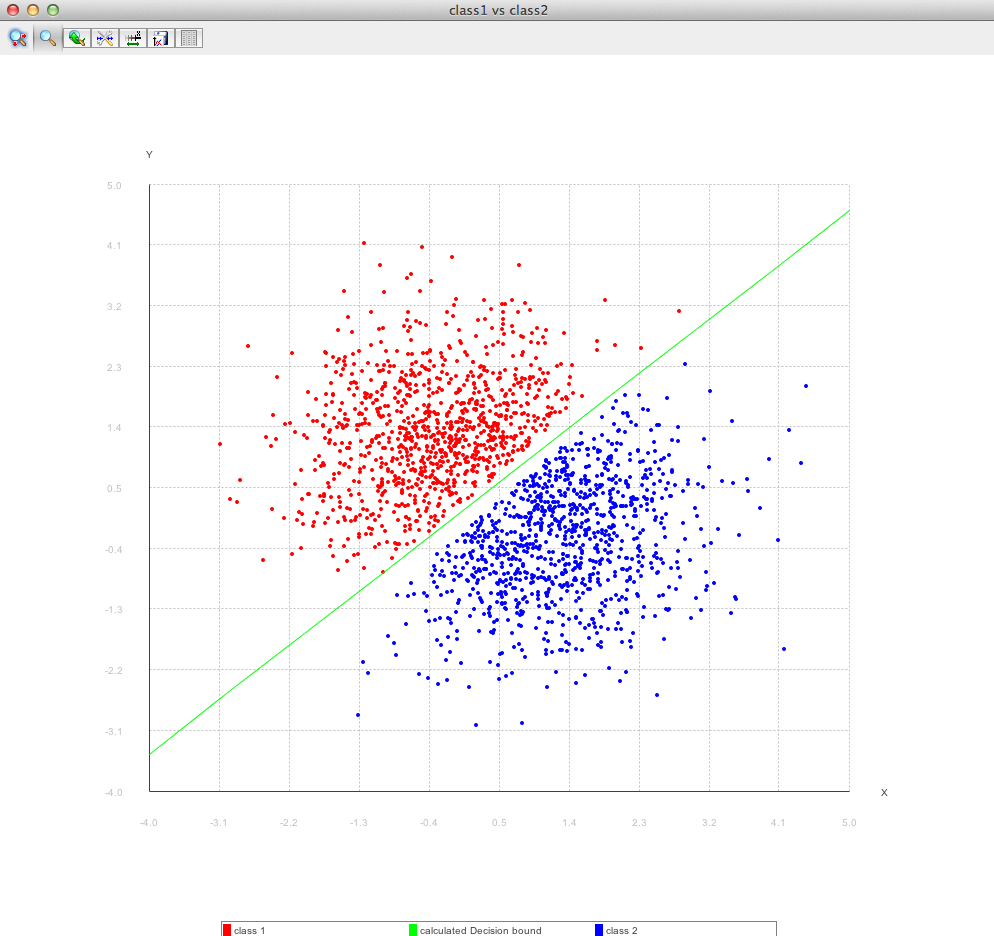


Figure 2

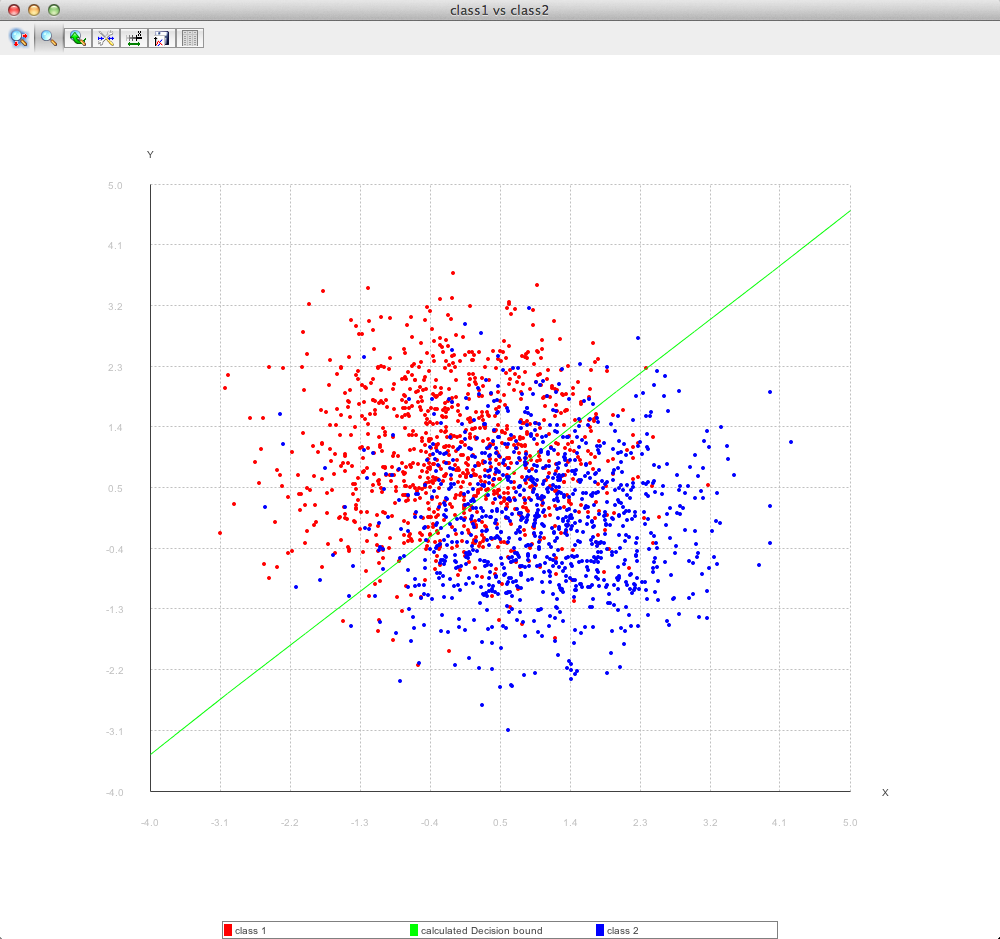


Figure 3

Below are tables with changing starting values on weights, learning rate, and shuffling of the data respectively.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **W0** | **W1** | **W2** | **# Of Epochs** | **Learning rate** |
| -100 | -100 | -100 | 17191 | 0.25 |
| -1 | -1 | -1 | 173 | 0.25 |
| 0 | 0 | 0 | 4 | 0.25 |
| 1 | 1 | 1 | 255 | 0.25 |
| 100 | 100 | 100 | 25460 | 0.25 |

Table 1: The learning rate is low to exaggerate the effects of the weight changes. The weights converge much faster when they are initially set to 0. All data is taken at the time error reached 0.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **W0** | **W1** | **W2** | **# Of Epochs** | **Learning rate** |
| 100 | 100 | 100 | 6365 | 1 |
| 100 | 100 | 100 | 637 | 10 |
| 100 | 100 | 100 | 64 | 100 |

Table 2: the weights are set to 100 based off figure one to exaggerate changes. As the learning rate gets large the weight converge much faster. All data is taken at the time error reached 0.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **W0** | **W1** | **W2** | **# Of Epochs** | **Learning rate** | **Was Shuffled** |
| 0 | 0 | 0 | 4 | 0.25 | False |
| 0 | 0 | 0 | 2 | 0.25 | True |

Table 3: halved the number of epochs by shuffling the input learning data.

A generalization graph was produced for 100 random weight sets leaving all other variables to a constant.

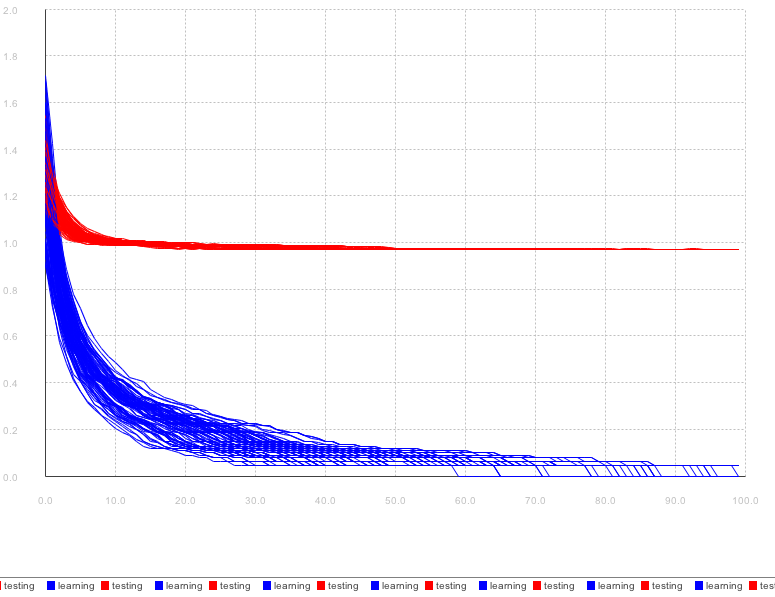


Figure 4: y-axis is the RMS error, x-axis are the number of epochs. Red is the error for testing data and blue is the error for the learning data.

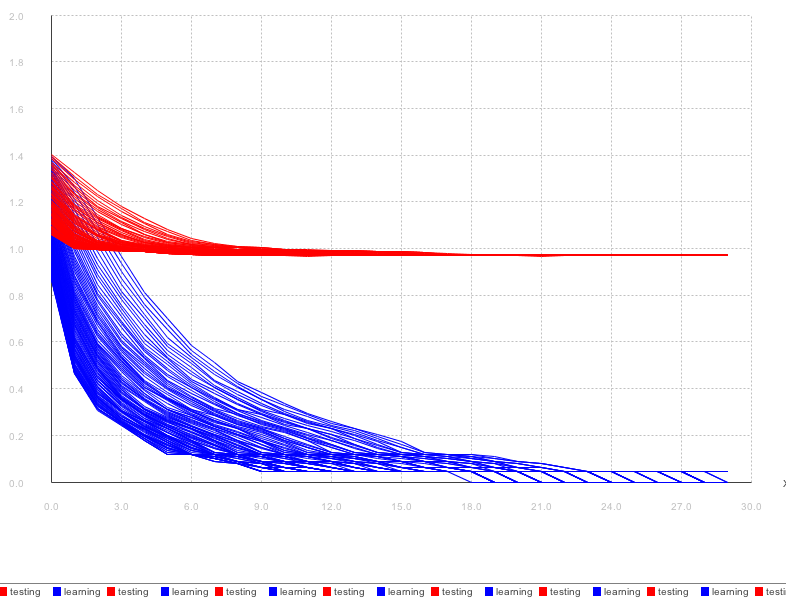


Figure 5: this increments the learning rate by 0.25 over 100 different runs. The lowest learning rate converges the slowest where as the highest learning rate (farthest left) converges much faster.

**Results and conclusions**

There are many things to be gleaned from the above data. From above: A learning rate increase also increases the convergence rate; starting the weights close to 0 makes the error rate converge faster; randomly shuffling the input learning data make the error converge faster.

Based on the results, when choosing start parameters for a fast converging perceptron, a high learning rate with a initial weight set close to 0 along with random shuffling of the inputs is the correct way to make the perceptron’s error rate to converge the fastest.

LMS

Citations

"Perceptron." *Wikipedia*. Wikimedia Foundation, 30 Sept. 2014. Web. 03 Oct. 2014. <http://en.wikipedia.org/wiki/Perceptron>.