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.1

Java was used to code the perceptron and the jmathplot 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 (root mean squared) error calculated for the testing class was 0.9596462478022414 and the learning rate was (learning rate/ 1000). Below are three plots: before learning with learning data (Figure 1), after learning with learning data (Figure 2), and one with the PDR testing data (Figure 3).

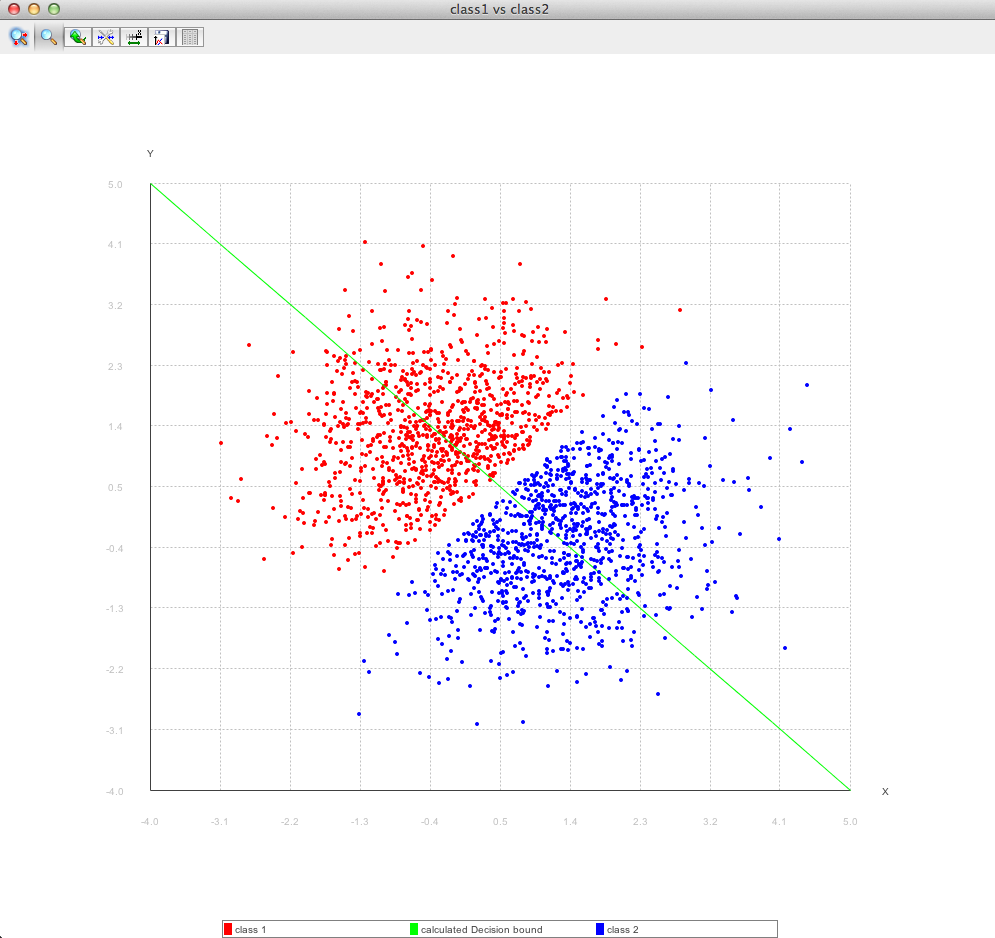


Figure 1: RMS error before learning with learning data.

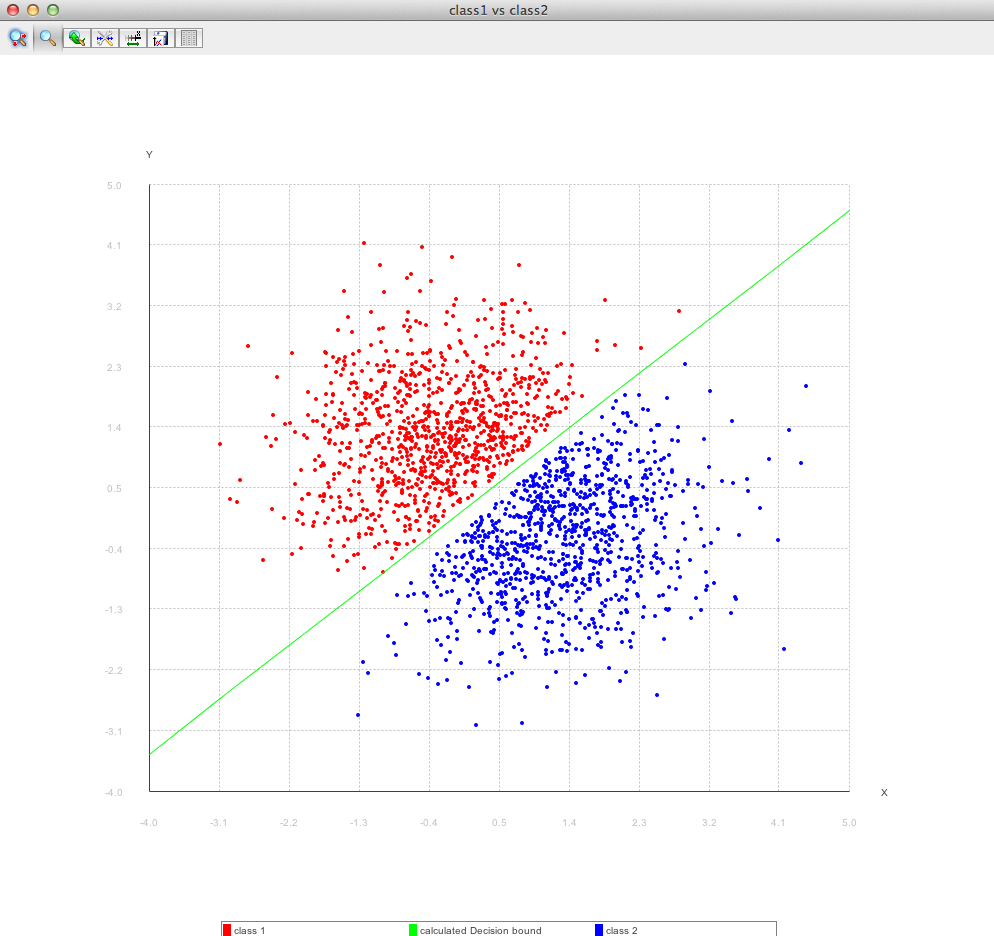


Figure 2: RMS error after learning with learning data.

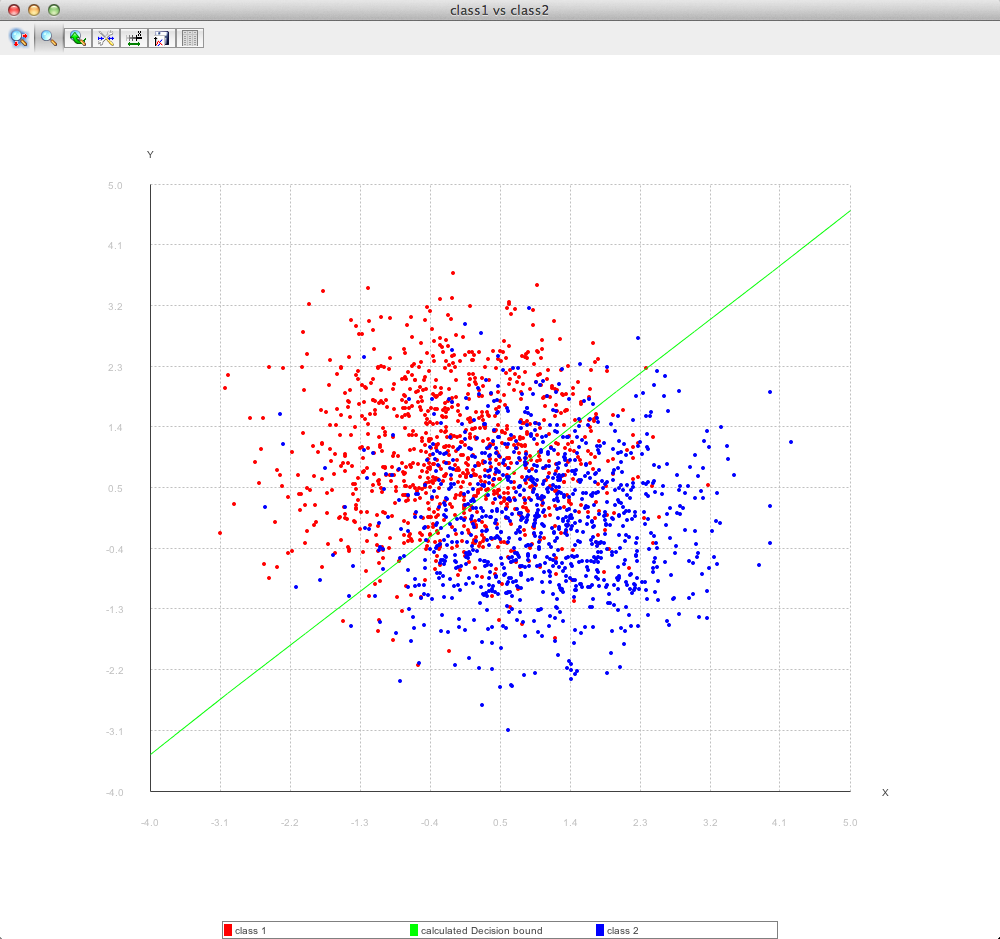


Figure 3: RMS error with PDR testing data.

Below are Tables 1-3 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.

In Graphs 4 and 5, a generalization graph was produced for 100 random weight sets leaving all other variables to a constant.

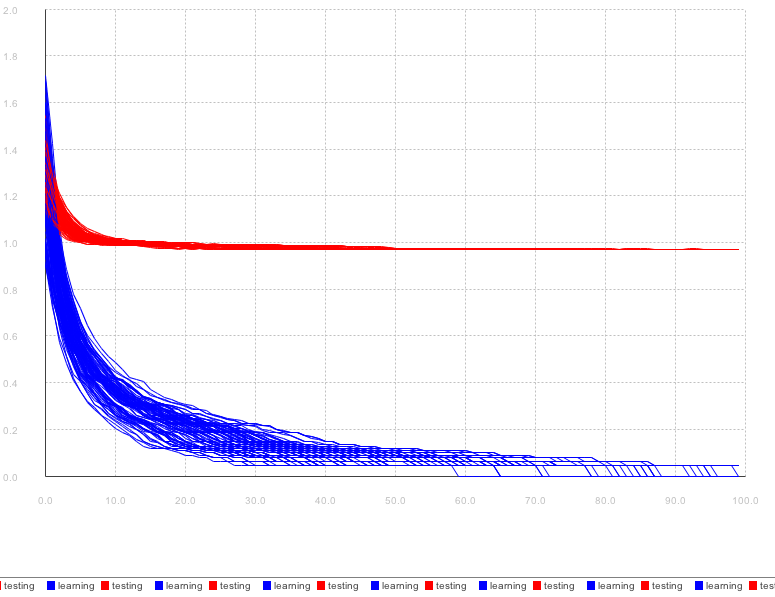


Figure 4: Y-axis is the RMS error, x-axis are the number of epochs. Red is the error for testing data and blue is the error for the learning data.

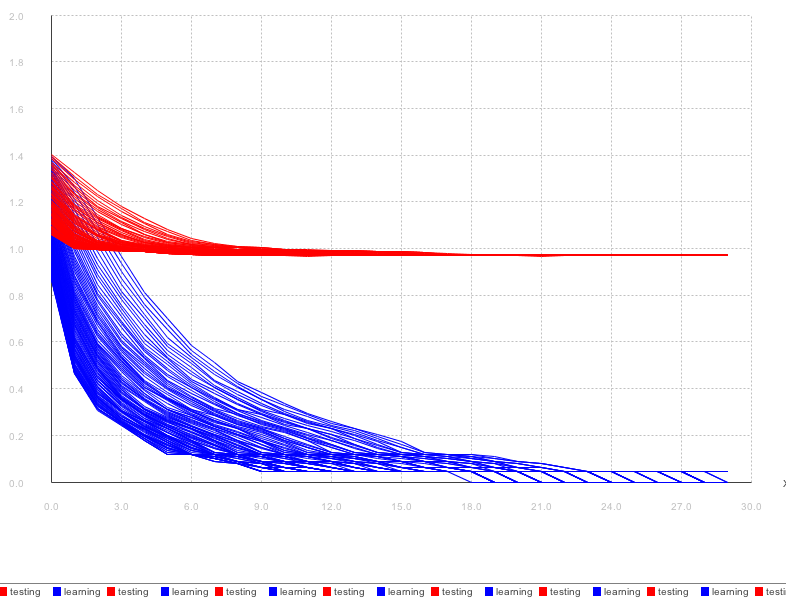


Figure 5: This increments the learning rate by 0.25 over 100 different runs. The lowest learning rate converges the slowest where as the highest learning rate (closest to y-axis) converges much faster.

**Perceptron Results and Conclusions**

There are many things to be gleaned from the above data. From the results presented above it is observed that a learning rate increase is positively correlated with the convergence rate. Starting the weights close to 0 makes the error rate converge faster and randomly shuffling the input learning data makes the error converge faster.

Based on the results, when choosing start parameters for a fast converging perceptron, a high learning rate with an 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

The delta rule is a gradient descent-learning rule for updating the weights of the inputs to artificial neurons in single-layer neural network. It is a special case of the more general backpropagation algorithm. The delta rule is derived by attempting to minimize the error in the output of the neural network through gradient descent. Therefore it is expected that the RMS error would approach an asymptote, and not 0 like the case of the perceptron.2

Java was used to code the LMS and the jmathplot java API (a common graphing java tool) was used to plot the data into a presentable format automatically at the end of each test. The LMS was given a set of learning data from two non-linearly separable classes. The testing data was made up of two non-linearly separable classes, as well. The stop criterion for learning was when an epoch occurred with an error difference from the previous of less then 0.0001. The RMS error calculated for the testing class was 0.9974943583775273 and the learning rate was (learning rate/ 1000). Below are three plots: LMS equation line before training with associated learning data (Figure 6), after learning with learning data (Figure 7), and one with the PDR testing data (Figure 8).

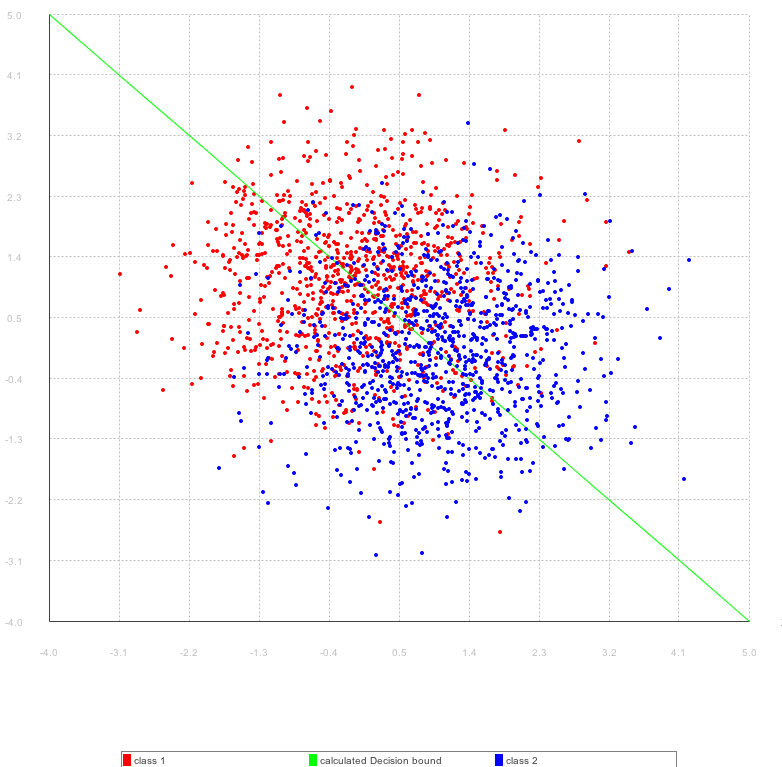


Figure 6: LMS equation line before training with associated learning data.

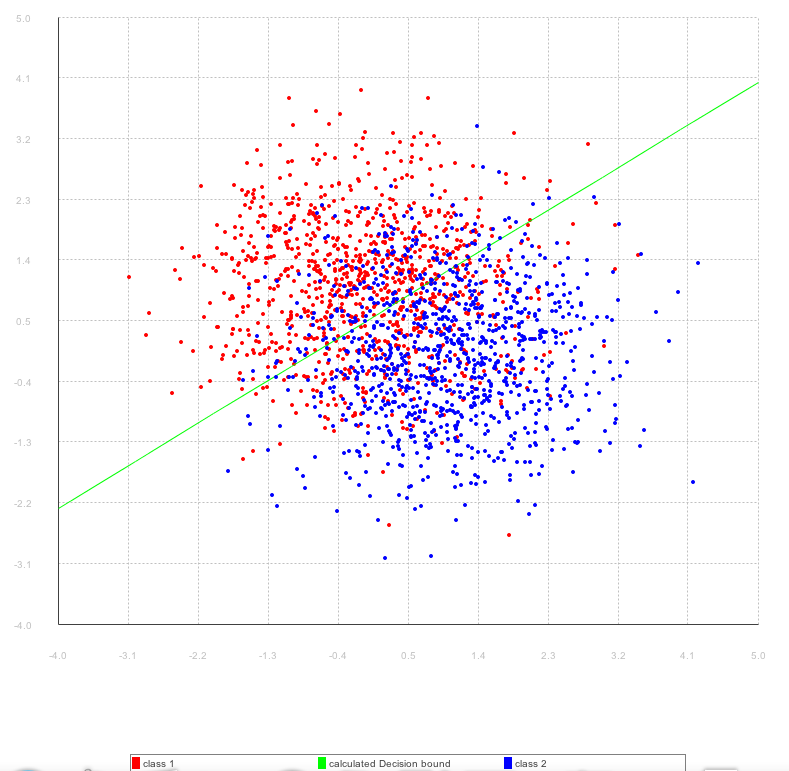


Figure 7: LMS equation line after training with associated learning data.

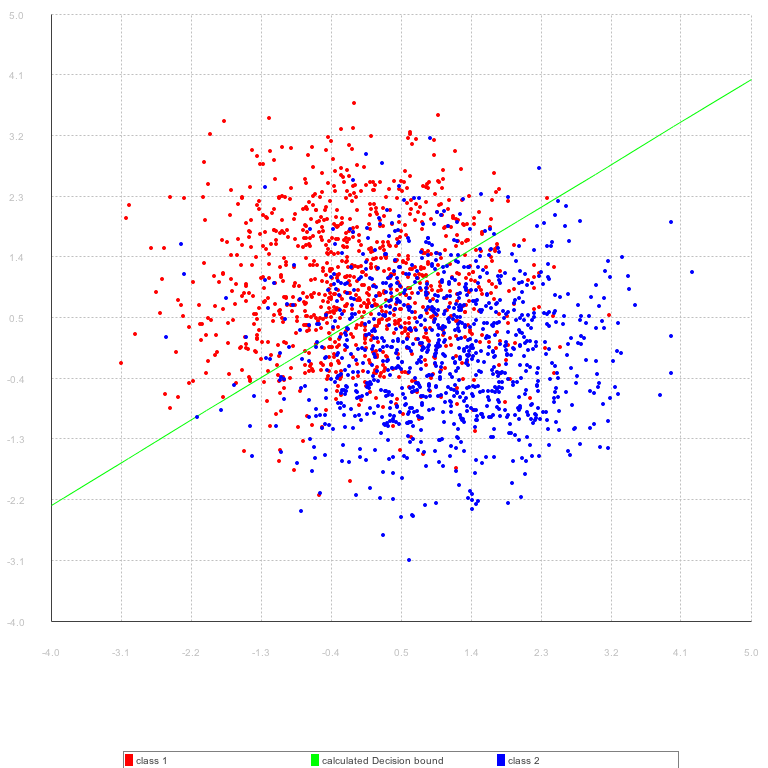


Figure 8: LMS equation line after training with testing data.

Tables 4-6 have varying stating weights (Table 4), learning rate (Table 5), and shuffling of the data (Table 6).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **W0** | **W1** | **W2** | **# Of Epochs** | **Learning rate** |
| -100 | -100 | -100 | broken | 0.25 |
| -1 | -1 | -1 | 18 | 0.25 |
| 0 | 0 | 0 | 2 | 0.25 |
| 1 | 1 | 1 | 17 | 0.25 |
| 100 | 100 | 100 | broken | 0.25 |

Table 4: The weights converge much faster when they are initially set to 0 and they generally converge must faster than the perceptron. All data is taken when the error change was minimized to 0.0001. Large start weights appear to have broken the LMS neuron.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **W0** | **W1** | **W2** | **# Of Epochs** | **Learning rate** |
| 1 | 1 | 1 | 17 | .25 |
| 1 | 1 | 1 | 11 | .5 |
| 1 | 1 | 1 | 9 | 1 |

Table 5: The weights are set to 1 (based on Table 4) to emphasize the changes. As the learning rate gets large the weight converges faster. All data is taken at the time the error reached 0.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **W0** | **W1** | **W2** | **# Of Epochs** | **Learning rate** | **Was Shuffled** |
| 1 | 1 | 1 | 17 | 0.25 | False |
| 1 | 1 | 1 | 18 | 0.25 | True |

Table 6: Shuffling does not appear to change the number of epochs to reach the convergence criteria.

In Graphs 9 and 10, a generalization graph was produced for 100 random weight sets leaving all other variables to a constant.

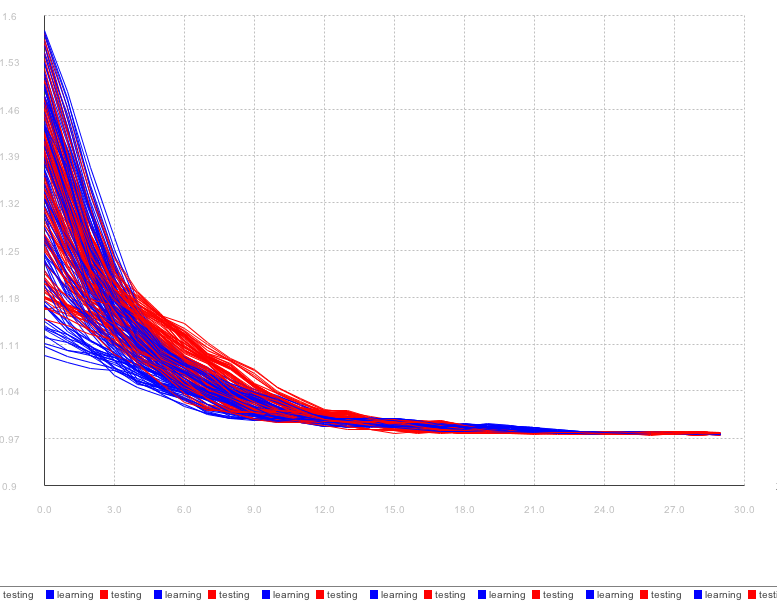


Figure 9: RMS error vs. number of epochs. The error for testing data (red) and the error for the learning data (blue) were taken after every epoch. The start weights were randomly chosen from 0-1.

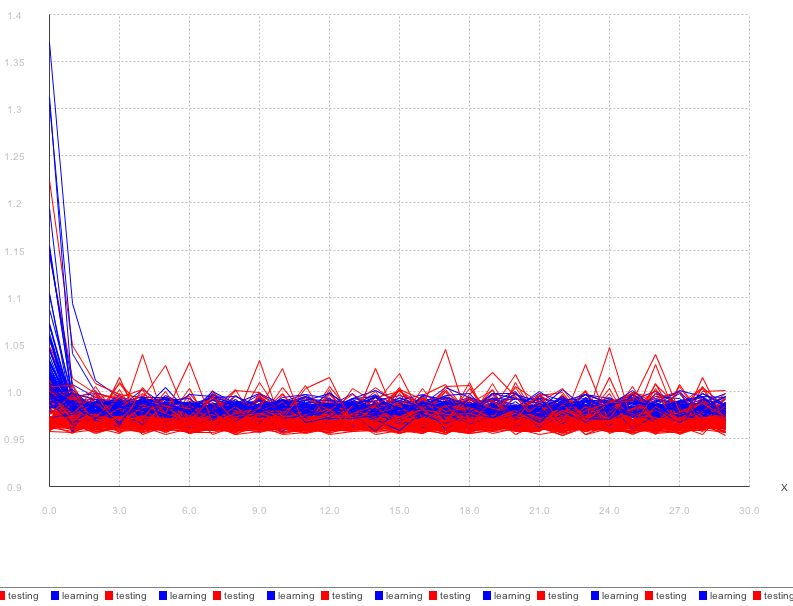


Figure 10: RMS error vs. number of epochs. The error for testing data (red) and the error for the learning data (blue) were taken after every epoch. The learning rate was incremented for each new LMS by 0.25 over 100 different neuron runs, all other variables stated with the same initial constants. The lowest learning rate converges the slowest and the highest learning rate (closest to y-axis (RMS axis)) converges much faster. The lower the learning rate, the less negative change when approaching the asymptote.

**LMS Results and Conclusions**

There are many things to be gleaned from the above data. From the LMS results presented above it is observed that learning rate increase positively correlates with the convergence rate increase but will cause the error to vary wildly once it has approached the asymptote. Starting the weights close to 0 makes the error rate converge faster and randomly shuffling the input learning data has very little effect.

Based on the results, the correct way to make the LMS rate reach converge the fastest is to choose starting parameters with a learning rate that is high enough to converge fast yet low enough to remain close to the asymptote and initial weight set close to 0.

Citations

1. "Perceptron." *Wikipedia*. Wikimedia Foundation, 30 Sept. 2014. Web. 03 Oct. 2014. <http://en.wikipedia.org/wiki/Perceptron>.
2. "Delta Rule." *Wikipedia*. Wikimedia Foundation, 24 Aug. 2014. Web. 05 Oct. 2014. <http://en.wikipedia.org/wiki/Delta\_rule>.