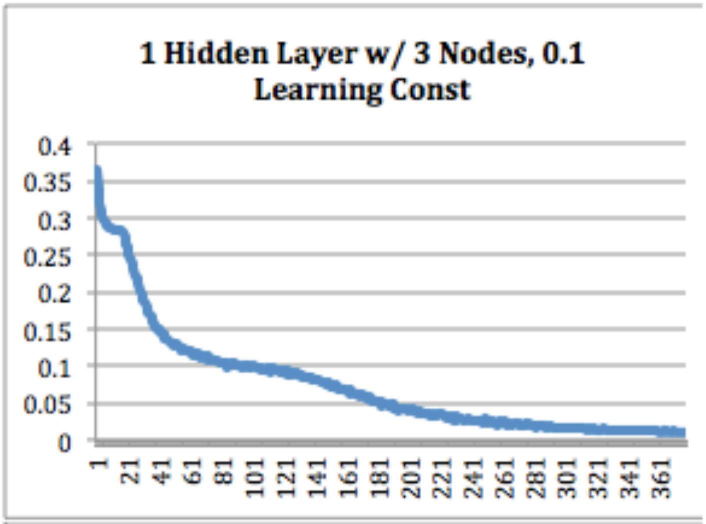
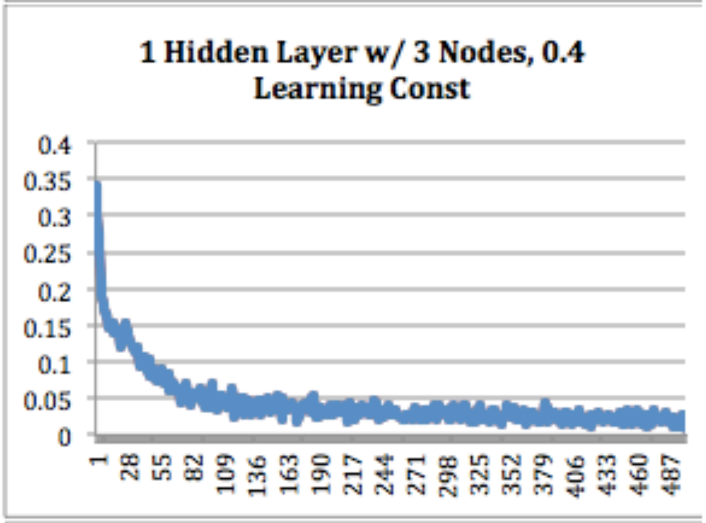
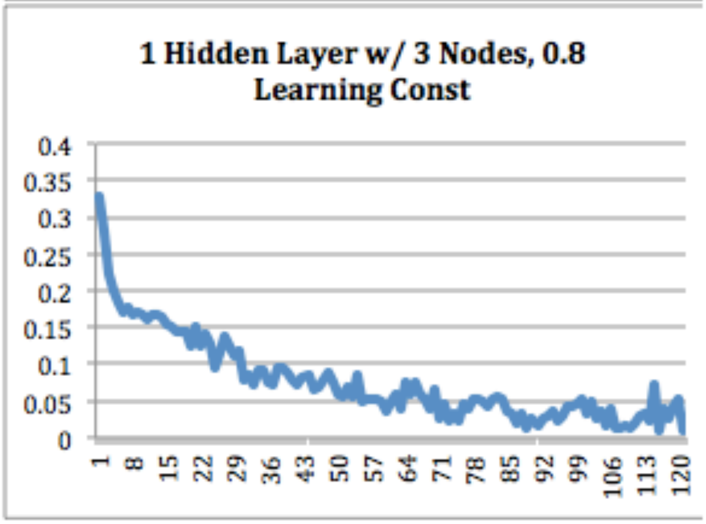
Christopher LaJon Morgan
Brother Christophe
Project 2: NN / CS 478
Oct 22, 2013

- **Discuss the effect of different learning rates on the algorithm's performance.**

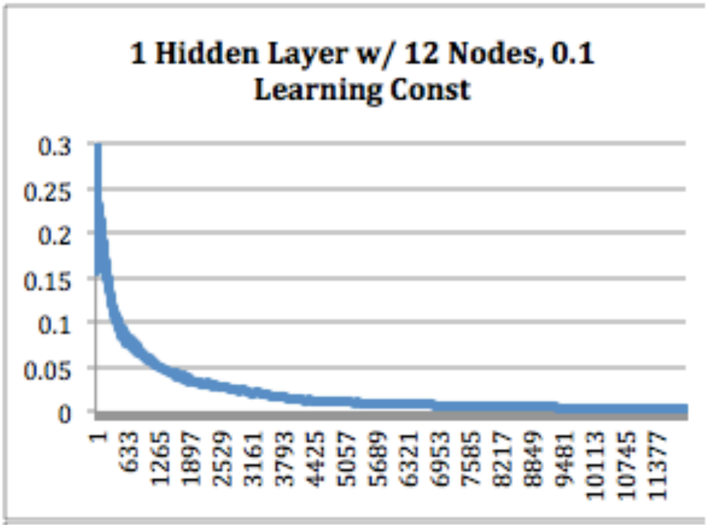
Iris

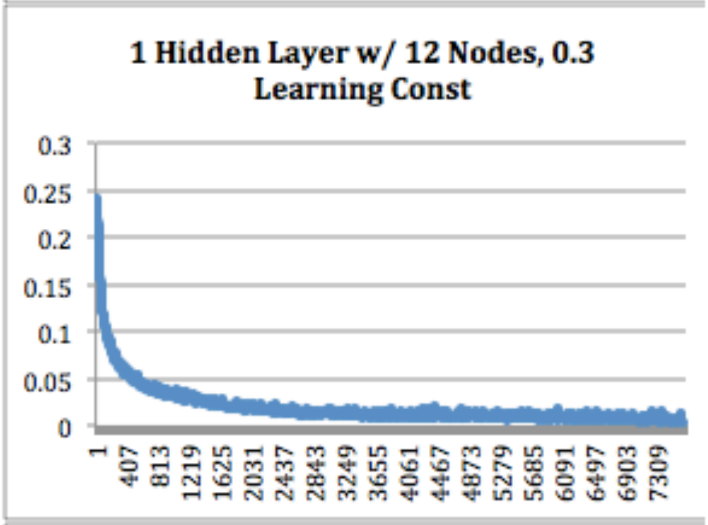


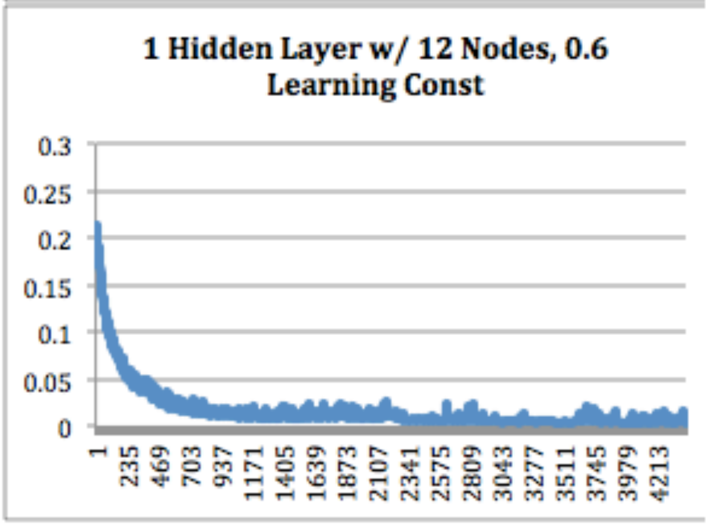




Vowel







The learning rate had an effect on at least three different behaviors of the learning algorithm. The effects of increasing the learning rate were an increase in jitter, an increase in the rate of decent, and a decrease in the algorithm's run time.

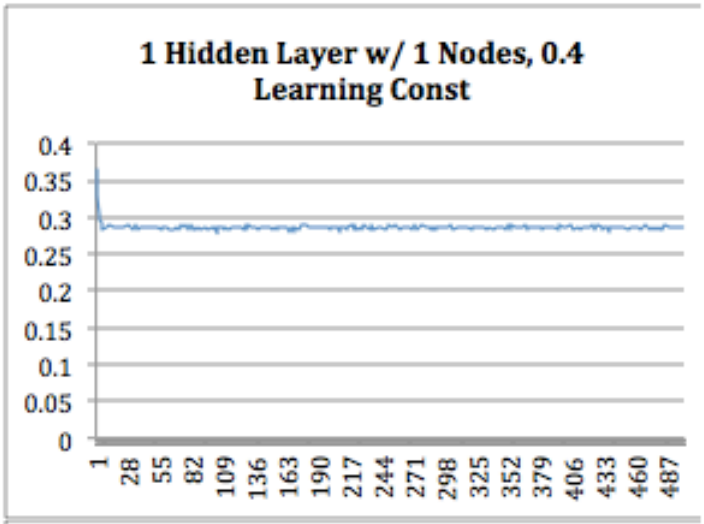
The increase in jitter of the algorithm on the Iris data was slightly more pronounced than the Vowels data. In the Iris data, as the learning rate increased from 0.1, to 0.4, 0.8, the jitter appeared to increase from approximately .01, to .02, to .04 respectively. However, the jitter in the algorithms error while representing the Vowels data appeared to be more incremental. For example, as the learning rate increased from 0.1, to 0.3, to 0.6, the jitter appeared to increase from approximately .01, to .02, to .03 respectively. Thus, it appears that the error fluctuates a little more forcefully as a higher learning rate is applied. This stands to reason, given a higher learning rate causes the algorithm to react more forcefully to error in the system. In other words, the system will appear more volatile given that a higher learning rate causes the algorithm to react more to a given degree of error.

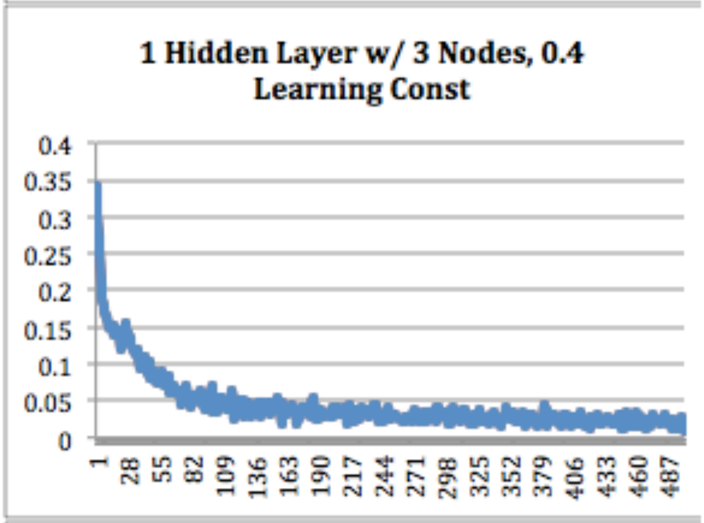
The data also suggests that increasing the learning rate also increases the rate of decent of the algorithm as it searches for a minimum error. In the Iris data, as the learning rate increased from 0.1, to 0.4, 0.8, the point at which the error began to level off decreased from approximately 341 iterations, to 217, to 106 respectively. In the Vowel data, as the learning rate increased from 0.1, to 0.3, 0.6, the point at which the error began to level off decreased from approximately 6953 iterations, to 4467, to 2575 respectively. Thus it appears that the error begins to level off quicker as a higher learning rate is applied.

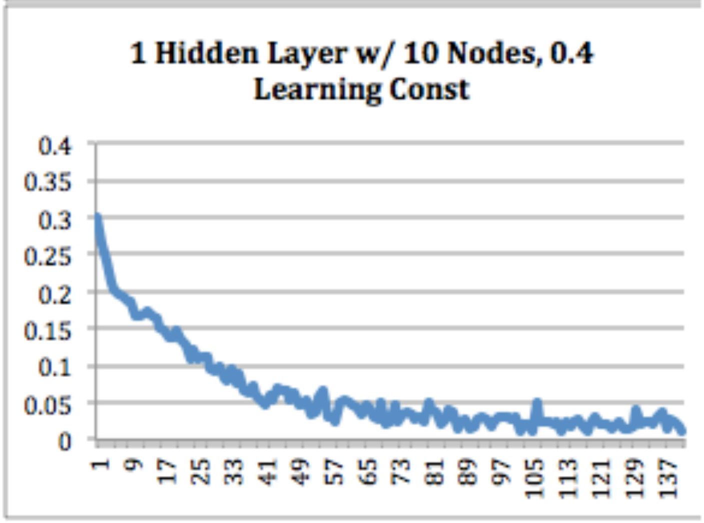
Finally, increasing the learning rate also showed on average a decrease in overall convergence time with both data sets. With the Iris data set, the algorithm decreased from approximately 361 iterations to convergence down to 120 iterations. With the Vowel data set, the algorithm decreased from approximately 11377 iterations to convergence to 4213 iterations, as the learning rate was increased. Thus it appears that the overall time to convergence may be decreased with an appropriate increase in an algorithms learning rate, to a certain extent of course.

- **Discuss the effect of different numbers of hidden units on the algorithm's performance (1-hidden layer case).**

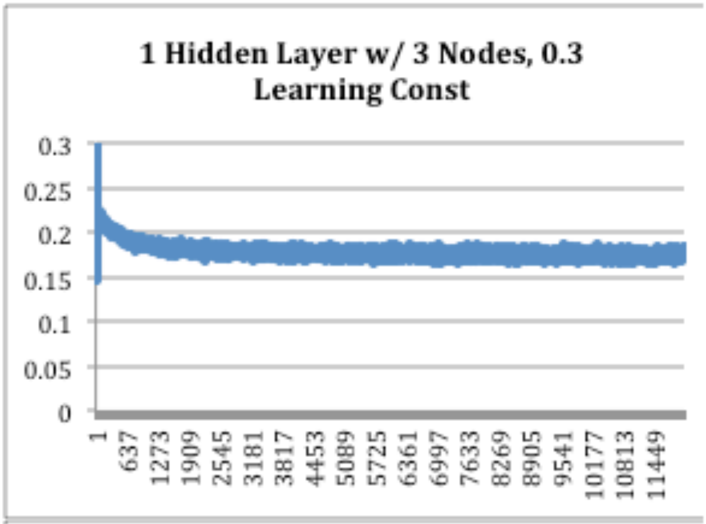
Iris

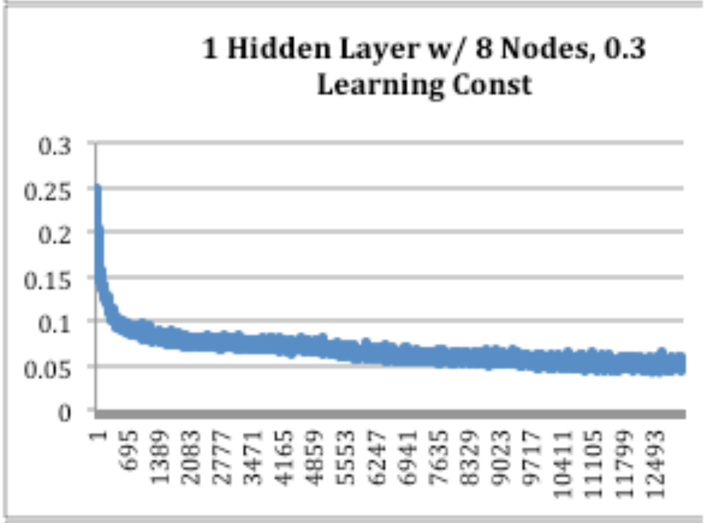


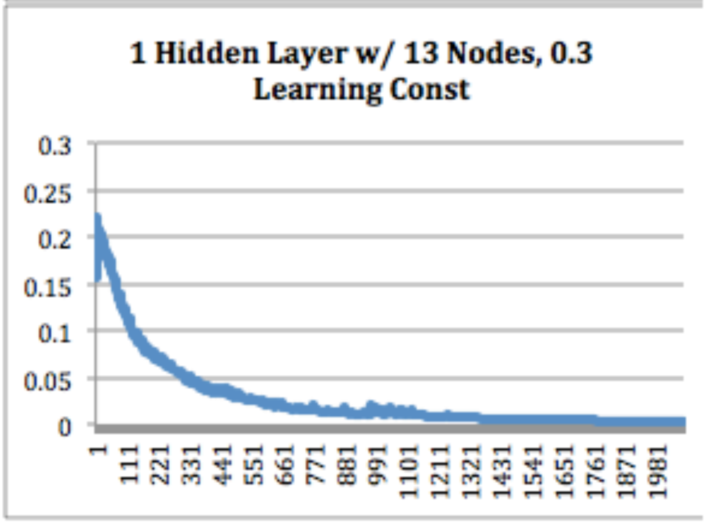


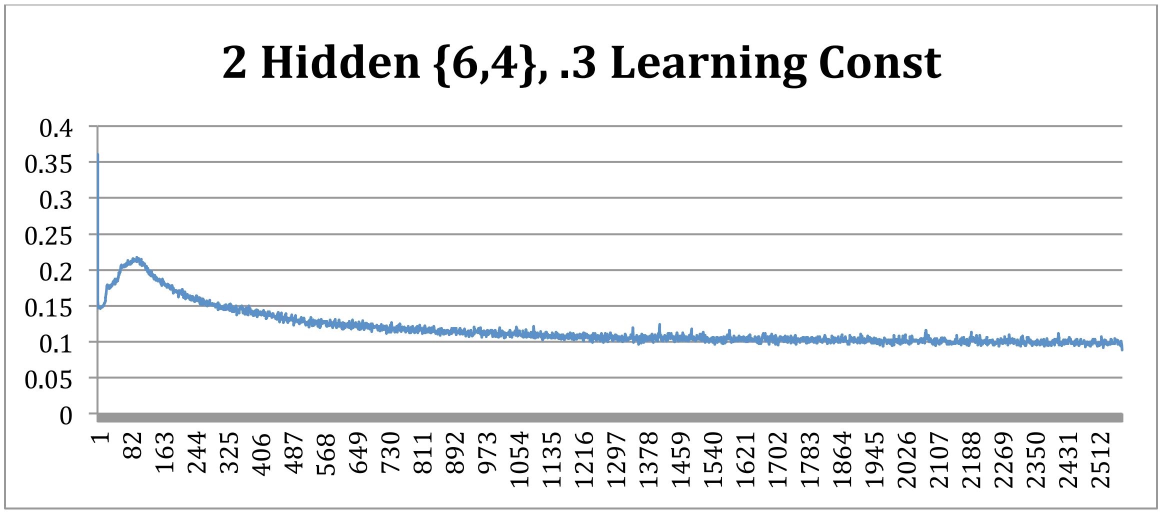


Vowel









The data suggests that as the number of hidden nodes increases the algorithm decreases in convergence time, converges to a lower critical error value, and increases in rate of decent to convergence.

The algorithm on both the Iris and the Vowel data did not converge on a low error rate when given a small number of hidden nodes. For example, the algorithm on the Iris data set with one hidden node did not manage to converge to a low critical error value, and it was terminated early. Also, the algorithm on the Vowel data set with three and then eight hidden nodes did not manage to converge to a low critical error value, and it was also terminated early. However, as the number of hidden nodes increased to three and then ten for Iris and to 13 hidden nodes with the Vowel data set, the algorithm was able to converge to a low critical error value and neither algorithm was terminated early. Thus, the data suggests that increasing the number of hidden nodes increases the algorithms ability to model a particular data set. Therefore, increasing the number of hidden nodes might be a good way to increase the accuracy of the algorithm and decrease the minimum error the algorithm is able to reach.

Increasing the number of hidden nodes, along with decreasing the minimum error, also caused an increase in the rate at which the algorithm arrived at its minimum error. With the Iris data, as the number of hidden nodes was increased from 1, to 3, to 10 the point at which the algorithm's error began to level off decreased from 487 iterations, to 217, to 105 respectively. This means the algorithm leveled off faster. The Vowel data followed a similar pattern, as the number of hidden nodes increased from 3, to 8, to 13 the point at which the algorithm's error began to level decreased from 1909 (at .17 critical error), to 6941 (at .05 critical error), to 1321 (at .004 critical error) respectively. Given the assumption that because the critical error dropped from iteration to iteration, we can reasonably justify the fact that the second instance is higher that the first. Given this assumption, the data suggest that increasing the number of hidden nodes helps to increase the rate at which the algorithm finds a lower minimum error and therefore decreases the number of iterations that are required to seek a lower minimum error.

Overall, it appears that increasing the number of hidden nodes, to some extent, might allow the algorithm to generate a more representative model of the data. With a more representative model the algorithm is able to find a lower minimum error and increase its predictive accuracy.

On a side note, the effect of adding an additional layer to my NN algorithm decreased the accuracy of the algorithm and increased the minimum convergence error from .004 with one layer and 13 nodes, to .09 with the two hidden layer version.

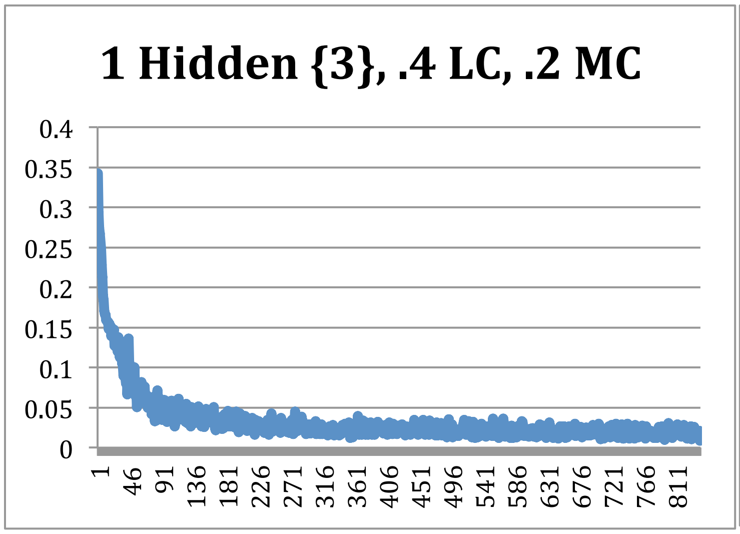
- **Compare your recorded best numbers of hidden nodes for each problem with the following heuristic value: *H*=*N*/(10(*I*+*O*)), where *N* is the size of the (training) data set, *I* is the number of network inputs and *O* is the number of outputs.**

Iris	Vowel
H = 45 / (10 + (4 + 3))	H = 248 / (10 + (12 + 11))
H = **2.647**	H = **7.5**
My Experimental Best == 3	My Experimental Best == 13

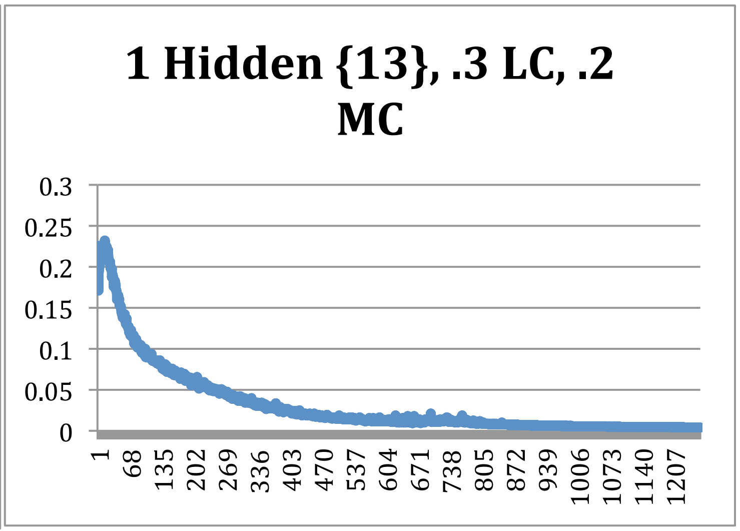
If one rounds both of the heuristics to there nearest whole (3 and 8) then the Iris hidden node measurement is right on with this heuristic. However, the Vowel hidden node measurement for the number of hidden nodes differs from the heuristic by 5 hidden nodes. 5 is 63% of the suggested heuristic, in other words, this is a pretty big difference. I'm not sure why there would be such a large discrepancy if this is an accurate heuristic.

- **How did the momentum term affect the learner's behavior (number of epochs to convergence, final accuracy, etc.)?**

Iris



Vowel



The momentum term changed the behavior of the learner in different ways with respect to the Iris data set and the Vowel data set. With the Iris data set, the moment term didn't seem to affect the learner very much in terms of overall accuracy, epochs to convergence, and epochs to error level out. The overall accuracy of the learner, with the Iris data, stayed about the same, with both performing at about 100% accuracy on the training set and 97% accuracy on the test set. There was a small difference however in epochs to convergence and error level out. Without the momentum term the learner converged after 487 epochs and its error leveled out at 271 epochs, however, with the momentum term the learner converged after 811 epochs and its error leveled out at 316 epochs. This suggests that the momentum term didn't affect the learner on the Iris data very much.

The momentum term showed similar effects on the algorithm with the Vowel data. Meaning, that with the Vowel data and the momentum term, the accuracy of the algorithm stayed close to the 90% accuracy on the training data and 60% accuracy on the test data, which was the same seen without the momentum term. However, with the Vowel data and using the momentum term, the algorithm did converge faster and level out faster than its none momentum counter part. With the momentum term the algorithm converged after 1207 epochs versus 1981 epochs. The momentum algorithm's error also leveled out after about 939 epochs versus the 1431 epochs. Thus, it appears that the momentum term can help the learner come to convergence faster and avoids some of the jitter seen by just increasing the learning rate.