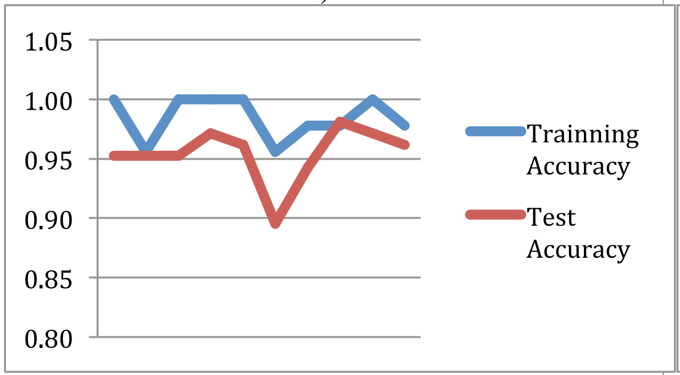
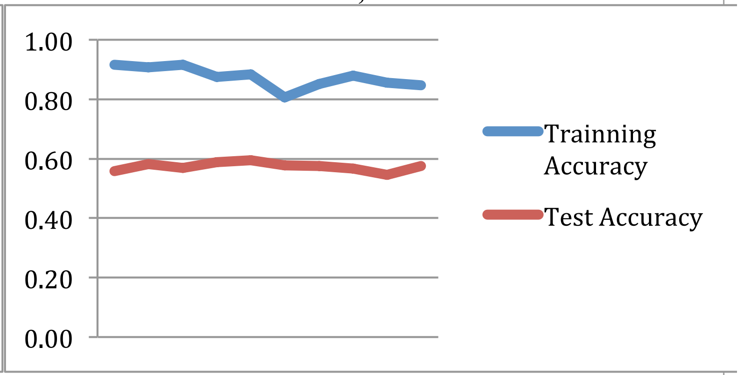
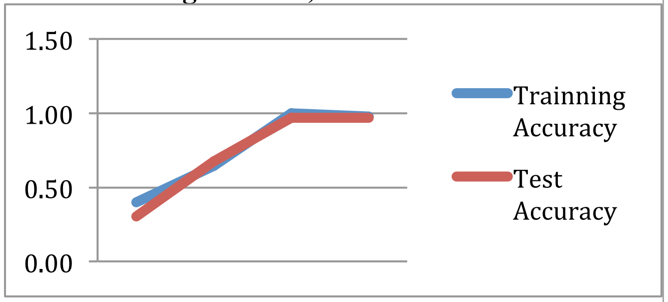
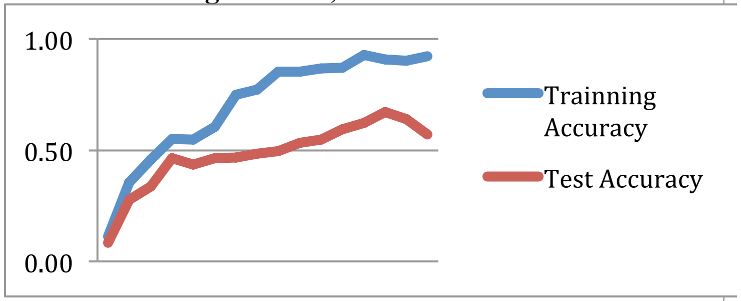
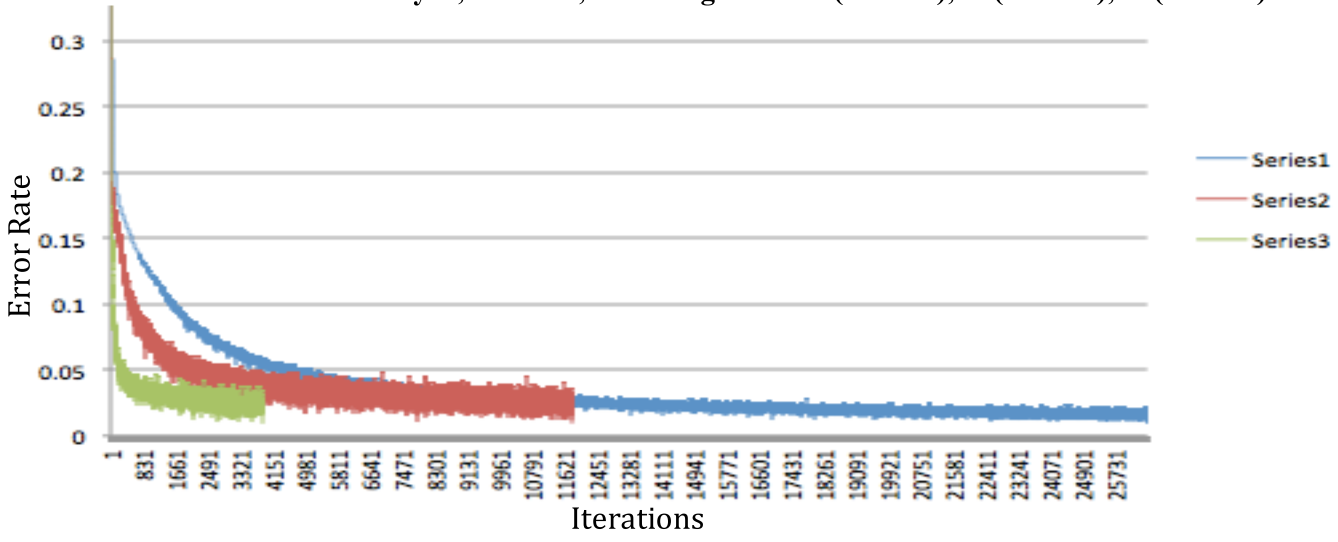
Christopher LaJon Morgan
Brother Christophe
Project 2: NN / CS 478
Oct 22, 2013

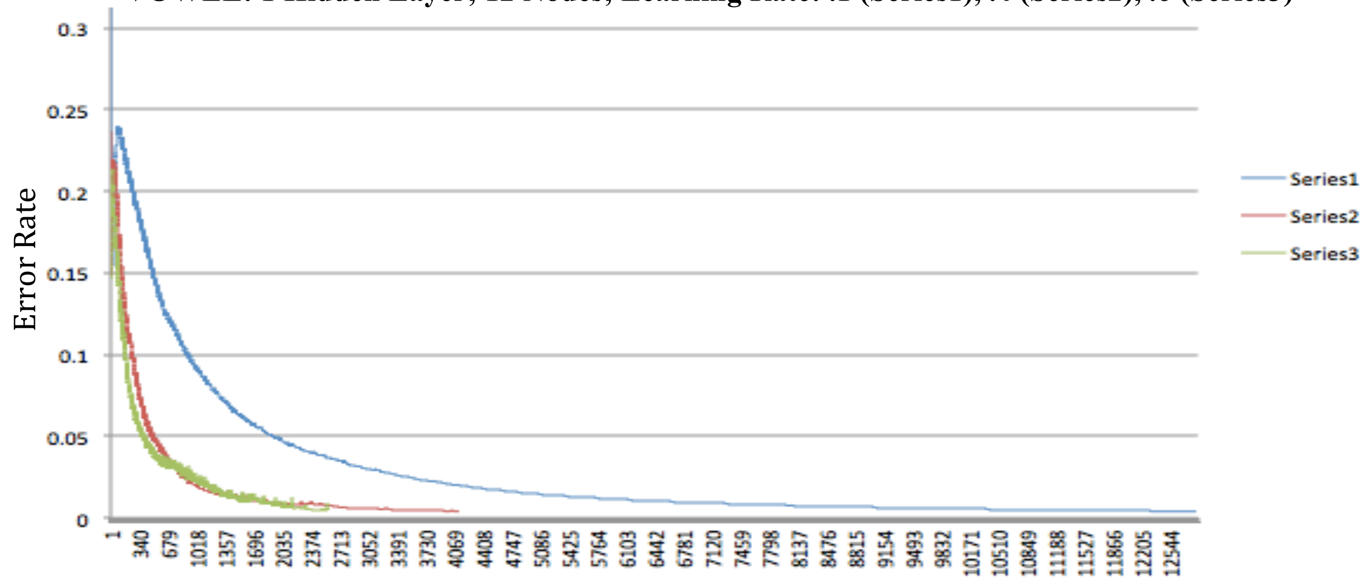
- **Discuss the effect of different learning rates on the algorithm's performance.**

Iris	**Vowel**
Hidden Nodes: 3, BEST L.R. == .4	**Hidden Nodes: 12, BEST L.R. == .3**
	
X-Axis: Learning Rate (.1 to 1) in .1 increments	**X-Axis: Learning Rate (.1 to 1) in .1 increments.**
Learning Rate: .4, BEST # Nodes == 3	**Learning Rate: .4, BEST # Nodes == 13**
	
X-Axis: Number of Hidden Nodes: {1-4,10}.	**X-Axis: Number of Hidden Nodes: {1-16,30}**

IRIS: 1 Hidden Layer; 3 Nodes; Learning Rate: .1 (Series1), .4 (Series2), .8 (Series3)



VOWEL: 1 Hidden Layer; 12 Nodes; Learning Rate: .1 (Series1), .4 (Series2), .6 (Series3)



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.

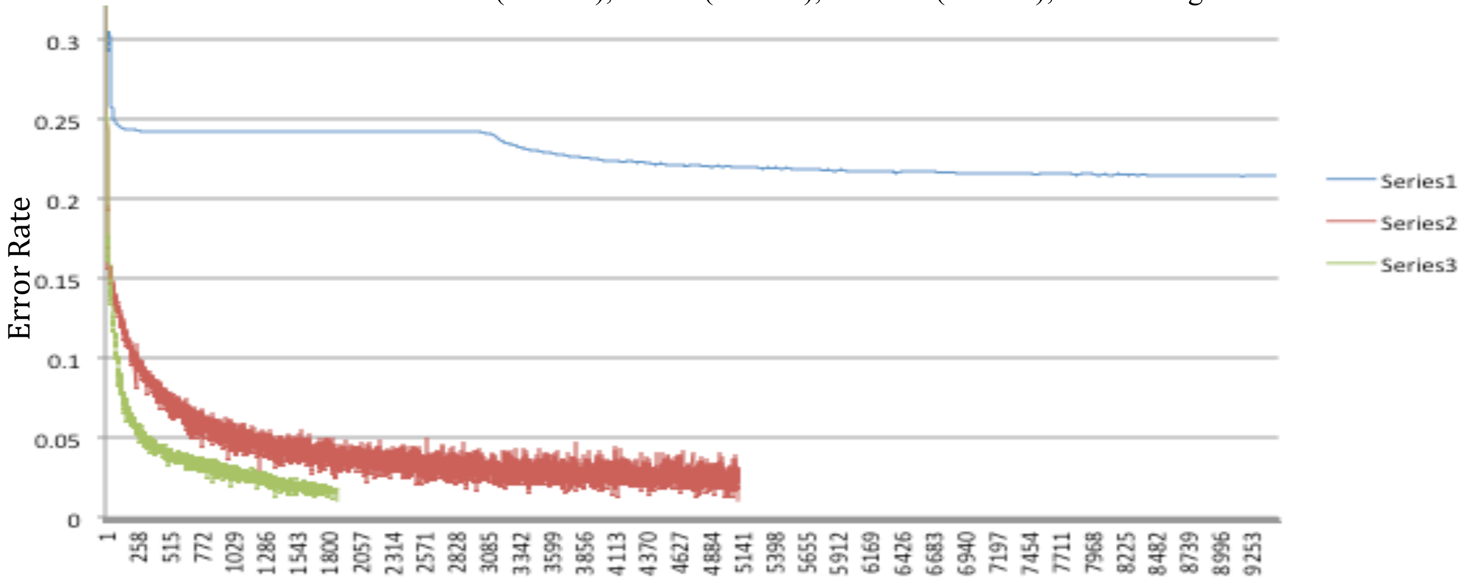
The increase in jitter of the error as the learning rate was increased appeared to have the same affect on both data sets. In the Iris data, as the learning rate increased from 0.1, to 0.4, 0.8, the jitter appeared to increase from approximately .01, to .02, to .03 respectively. On the Vowels data as the learning rate increased from 0.1, to 0.3, to 0.6, the jitter appeared to increase from approximately .006, to .008, to .01 respectively. Thus, it appears that the error fluctuates a little more forcefully as a higher learning rate is applied. WHY: 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 because a higher learning rate causes the algorithm to react (change its weights) more in response to a given degree of error.

The data also suggests that as the learning rate is increased the rate at which the error descends also increases. 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 (the decent stopped) decreased from approximately 9961 iterations, to 2491, to 1661 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 6103 iterations, to 2713, to 2374 respectively. Thus it appears that the error begins to level off quicker as a higher learning rate is applied. WHY: This also stands to reason because a higher learning rate will cause the algorithm to take larger steps at each iteration, which would cause it to descend faster.

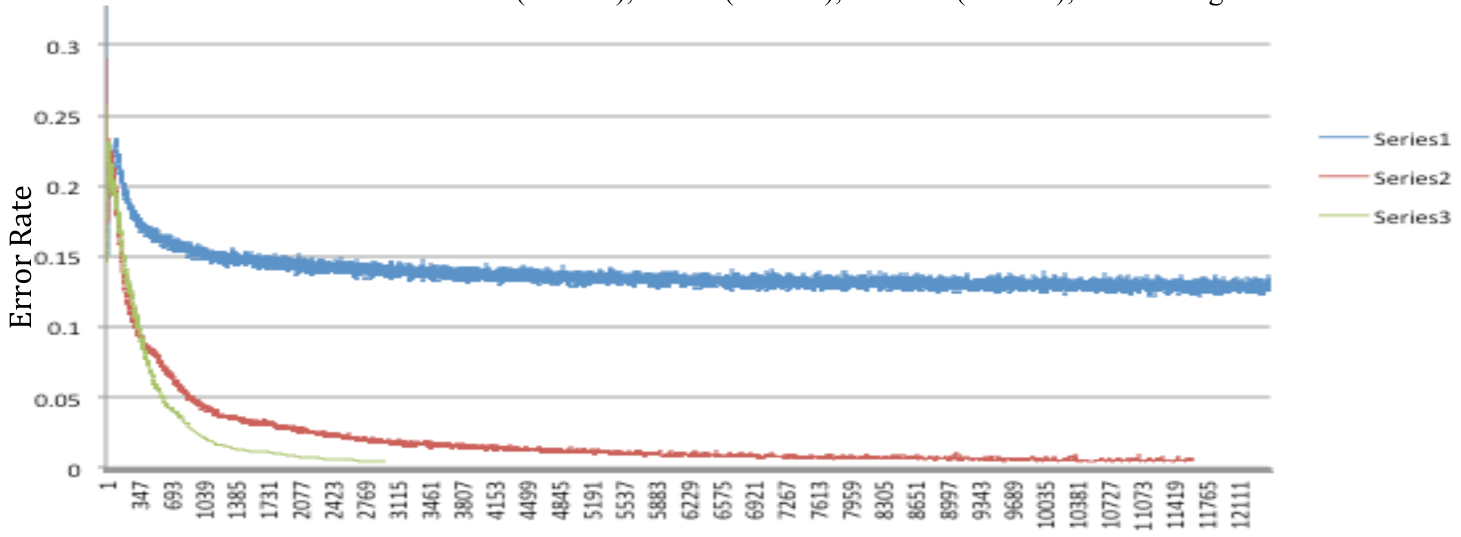
Finally, increasing the learning rate also showed a decrease in overall convergence time with both data sets. With the Iris data set, the algorithm decreased from approximately 25731 iterations to convergence down to 4151 iterations. With the Vowel data set, the algorithm decreased from approximately 12544 iterations to convergence to 2713 iterations, as the learning rate was increased. Thus it appears that the overall time to convergence might also be decreased with an appropriate increase in an algorithms learning rate, to a certain extent of course. WHY: This also stands to reason given that the algorithm is simply taking larger steps as the learning rate increases. I would also assume this could have the opposite effect if the algorithm were to simply overshoot the minimum each time.

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

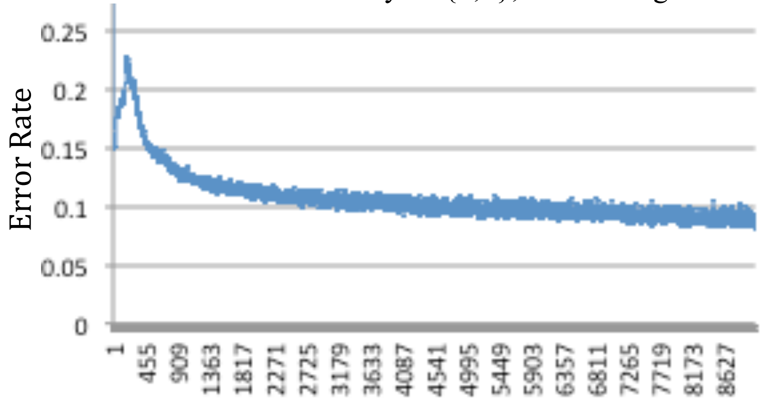
IRIS: 1 Hidden Node (Series1), 3 H.N.(Series2), 10 H.N. (Series3); .4 Learning Rate



VOWEL: 3 Hidden Node (Series1), 8 H.N.(Series2), 13 H.N. (Series3); .3 Learning Rate



VOWEL: 2 Hidden Layers {6,4}; .3 Learning Rate



The data suggests that as the number of hidden nodes increases the algorithm: decreases in convergence time, can converge 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. The algorithm didn't converge evaluating Iris with 1 hidden node and evaluating Vowel with 8 hidden nodes. However, as the number of hidden nodes increased to 3 and then 10 for Iris and to 8 and 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. WHY: The data suggests that increasing the number of hidden nodes increases the algorithms ability to model a particular data set. There exists a minimum number of hidden nodes required to model the complexity of a data set.

Increasing the number of hidden nodes, along with decreasing the minimum error, also caused an increase in the rate of decent of the model's error. With the Iris data, as the number of hidden nodes increased from 1, to 3, to 10 the error at which the algorithm began to level off decreased from .25, to .05, to .03 respectively. The Vowel data followed a similar pattern, as the number of hidden nodes increased from 3, to 8, to 13 the level off point decreased from .15, to .03, to .01 respectively.

In similar fashion, on both data sets, the algorithm also terminated in fewer iterations as more hidden nodes were added. 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 also helps to decrease the number of iterations that are required to seek a lower minimum error. WHY: This occurs because the algorithm can build a more effective model of the data that it receives and therefore doesn't bounce around in error as much. More nodes allow easier representation of the data; therefore, a faster decrease in error, and therefore, a shorter run time.

On a side note, the effect of adding an additional layer to my NN algorithm decreased the accuracy of the algorithm on the training set by ~6% while stay about the same at 64% accuracy on the test set. It also increased the minimum convergence error from .004 with one layer and 13 nodes, to .08 with the two hidden layer version. WHY: Increasing the number of layers doesn't guarantee an increase in accuracy. There are potentially many more local minima, which would explain the higher minimum error. There are also more weights to adjust which might cause the algorithm to run much longer to find a lower minimum error.

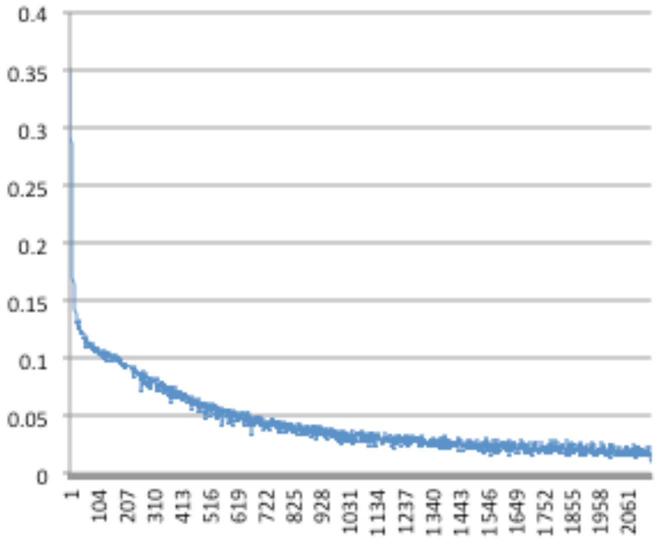
- **Compare your recorded best # of H.N. for each problem with: *H*=*N*/(10 (*I*+*O*))**

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

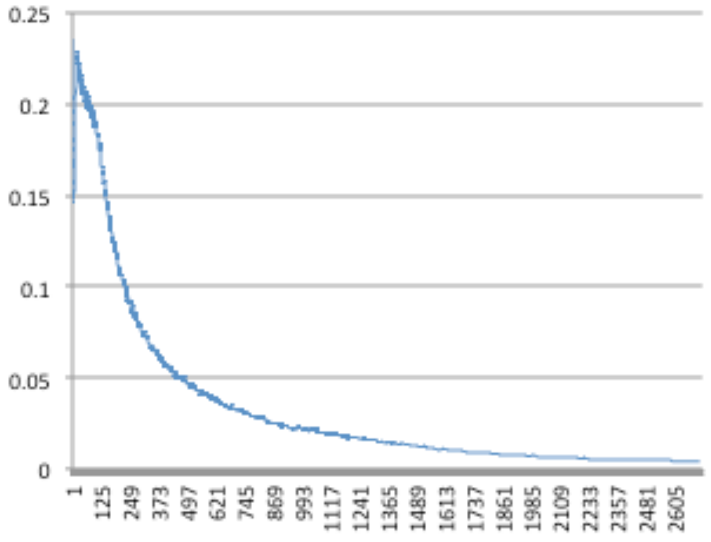
If one rounds both of the heuristics to there nearest whole (3 and 8) then the Iris hidden node measurement is right on with this heuristic. However, the Vowel hidden node measurement for the number of hidden nodes differs from the heuristic by 5 hidden nodes. 5 is 63% of the suggested heuristic, in other words, this is a pretty big difference. One possible explanation is my number of nodes doesn't take into account node configuration. If I were to try all possible combinations of nodes (meaning how they are layered) I might have found a smaller number of nodes, closer to 8, would have given me a similar accuracy as my one layer 13 configuration.

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

Iris
1 Hidden Layer {3Nodes}, .4 LC, .2 MC



Vowel
1 Hidden Layer {13Nodes}, .3 LC, .2 MC



Adding the momentum term affected the overall accuracy of the algorithm, iterations to convergence, and the decent rate of the algorithms error. While evaluating the Iris data the MC term allowed the algorithm to become more accurate. It increased the accuracy of the algorithm on the training and test sets by 3%. However, on the Vowel data it decreased accuracy in both sets by 2%. In one case, MC could help the algorithm over come a local minima but on the other hand it might have caused the algorithm to pass a lower minimum by.

Adding the momentum term also decreased the drop error (the error at which the rate of error drop starts to level out) and the number of iterations on both data sets. On the iris data adding the MC term kept the drop error about the same at 0.05, however, on the Vowel data it dropped from 0.04 to 0.03. Also the number of iterations the algorithm ran on the Iris data dropped from 5141 to 2061, and the with Vowel data the number of iterations dropped from 11419 to 2605. This is because adding the MC term acts similar to increasing the learning rate. But instead of a direct weight change the MC term takes into account the weight change made on the algorithm's last iteration. This is why adding the MC term helped to increase the rate at which the error dropped and helped to decrease the number of iterations required. It gives the algorithm a little boost down hill and on the plains. Therefore, one would expect it to decrease the number of iterations and the rate at which the error drops.