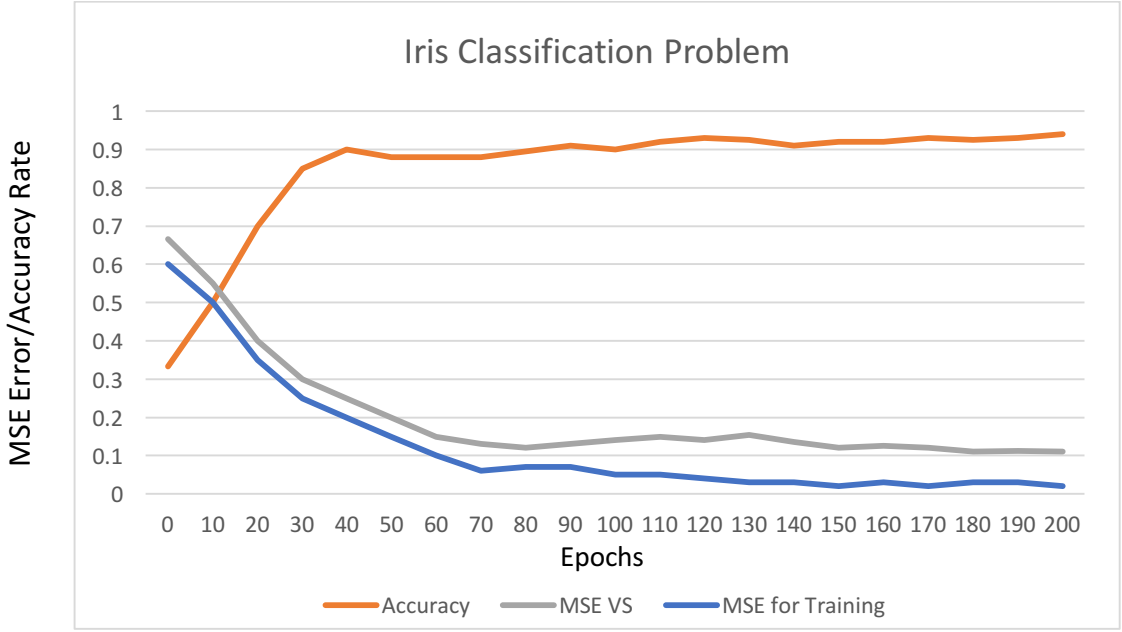
William Jones
478 Machine Learning

Iris Classification Problem

For the Iris classification problem, I initialized a neural network with 4 input nodes, one layer of 8 hidden units and 3 output units. The backpropagation algorithm was run with a learning rate of n = 0.1, a momentum variable of alpha = 0, and stochastic weight updates. Also, 75% of the iris data was randomly assigned for training and 25% for testing. I also did a check every five epochs to see if significant decrease was occurring in decreasing mean squared error and would stop if the difference was insignificant. The following graphs show the results in terms of the mean squared error (MSE) and classification rate during training:



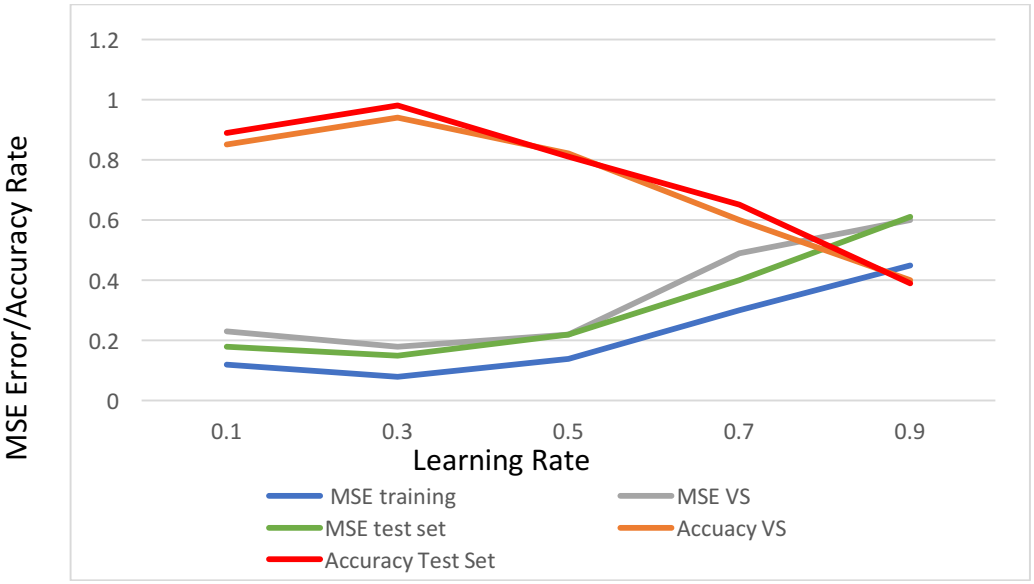
As you can see there definitely is a relationship with the mean squared error and the classification accuracy rate. By the end my accuracy was about 93-94% and didn't seem to really have any more improving to do.

Vowel Classification:

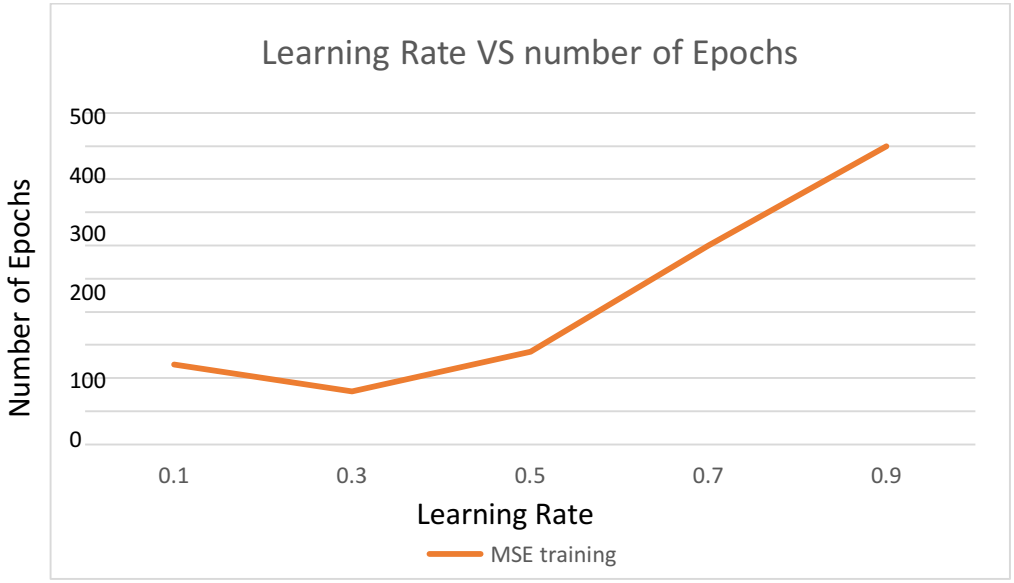
For the vowel classification problem, I used the same neural net configuration from above with the exception just using of single learning rate test. For the vowel classification problem, I ran the backpropagation algorithm with a neural net learning rate of n = 0.1, 0.3, 0.5, 0.7 and 0.9. When considering what data would be most useful for machine learning, I removed the first three columns in the data set. These columns contained information on the speaker's name and gender, I considered these features to not be as useful because there was not enough data to make them significant and in

my mind, I don't think that they should really make a difference as I fail to see what correlation they would have, unlike the other data values provided.

The stopping criteria I originally used was to compare each epoch's mean squared error with its previous mean squared error. If the difference was less than 0.01, then the stopping criterion was considered met. This stopping criterion allowed me to avoid overfitting but it did not guarantee backpropagation would leave a temporary local minima and thus I might miss out on finding a more accurate solution.

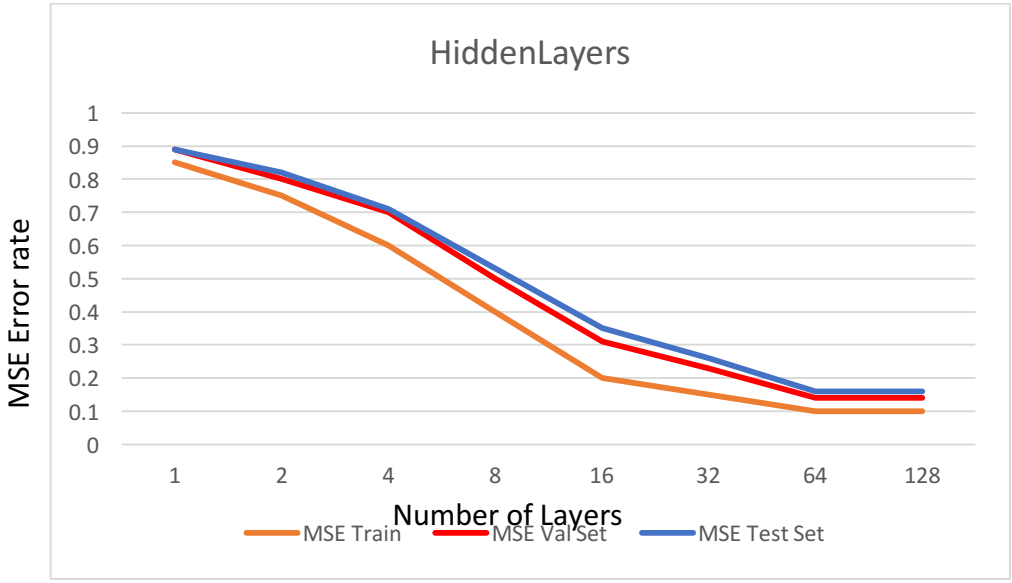


First off Learning rate 0.3 seemed to be the most accurate of the all the learning rates that I tried. Learning rate 0.1 seemed to take a lot more epochs to improve, but eventually got almost as accurate as 0.1 and may have become even better if more epochs were tried. This would make sense as it would be changing the rates a lot slower than 0.3 and 0.7 would. As expected 0.5, 0.7 and 0.9 made the most initial improvement the fastest, but it started to not really improve after 50 epochs and it didn't seem to learn anymore useful data. 0.3 seemed to be a happy balance between the two rates and seemed to improve faster than 0.1 but kept learning a lot longer than the 0.7 rate did. See the chart below



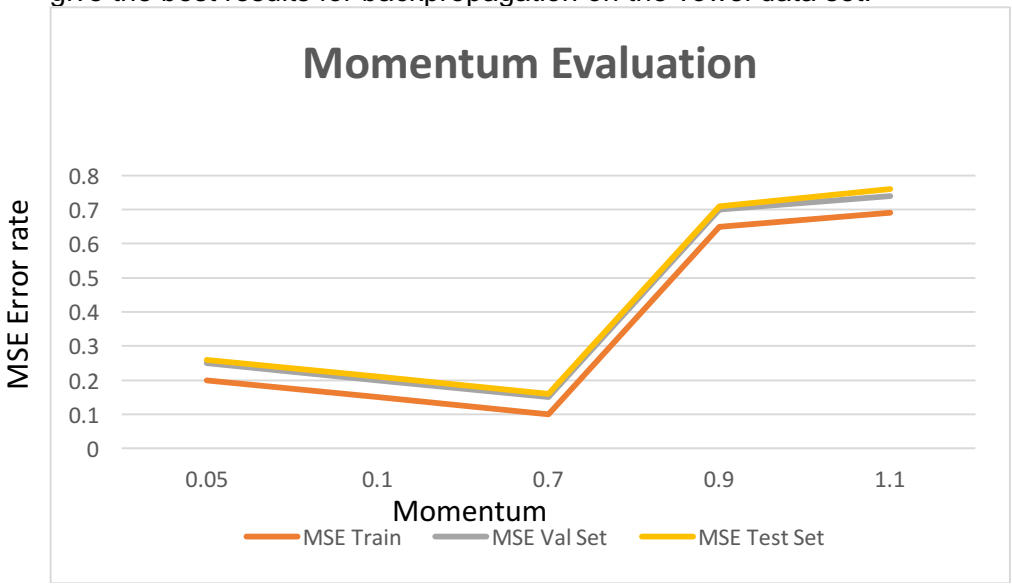
The Effects of Hidden Layers:

Backpropagation's performance rides heavily on whether or not it will escape local minim. This is because the Backpropagation algorithm employs a gradient descent search and so there is no guarantee that it will converge on the global minimum in the error space. Every weight in the network corresponds to a dimension in the search space and thus networks with many weights correspond to high dimensional surfaces. However, when there are more weights in the neural network it gives the gradient descent search a better chance to escape a local minimum that it may have entered from a single weight. Thus, the more weights in the network, the more dimensions that might provide 'escape routes' for gradient descent to fall away from the local minimum with respect to this single weight. Or in other words the more layers and thus the more weights we have the more likely we can learn more patterns and behaviors in the data that we are analyzing. The data definitely supports this idea as seen in the graphs below: I went up to 128 with 128 being the time that no improvement was made starting at 2 hidden layers.



Momentum Evaluation

I ran tests for momentum factors of values of 0.05, 0.1, 0.7, 0.9 and 1.1 with a learning rate of 0.2 and a neural network with one hidden layer of 32 hidden units which was the best that I got from the previous section. The average running time for the various factors of momentum was 76 seconds. As the momentum factor increased from 0.1 to 0.7, accuracy also increased. As can be seen in the data, the accuracy of the learning model with a momentum factor of 0.9 is drastically different than lesser momentum factors. A momentum factor allows gradient descent to get over local minima humps and accelerate learning in areas where no significant improvements have been made to weight updates. However, it also has the potential of causing gradient descent pass over the global minimum which can cause a decrease in validation set accuracy if not set to the correct frequency. A momentum factor of 0.7 seemed to give the best results for backpropagation on the vowel data set.



Experimenting:

I experimented with iris data set instead of using the Adam Classifier. This classifier is similar to the momentum classifier except it runs averages of both the gradients and the second moments of the gradients are used. I was curious to see if any significant progress or improvement was made by using this instead of just the regular momentum. It seems that it runs about the same as Momentum and doesn't to cause any significant improvement as you can see in the graph below compared to the one at the first page.