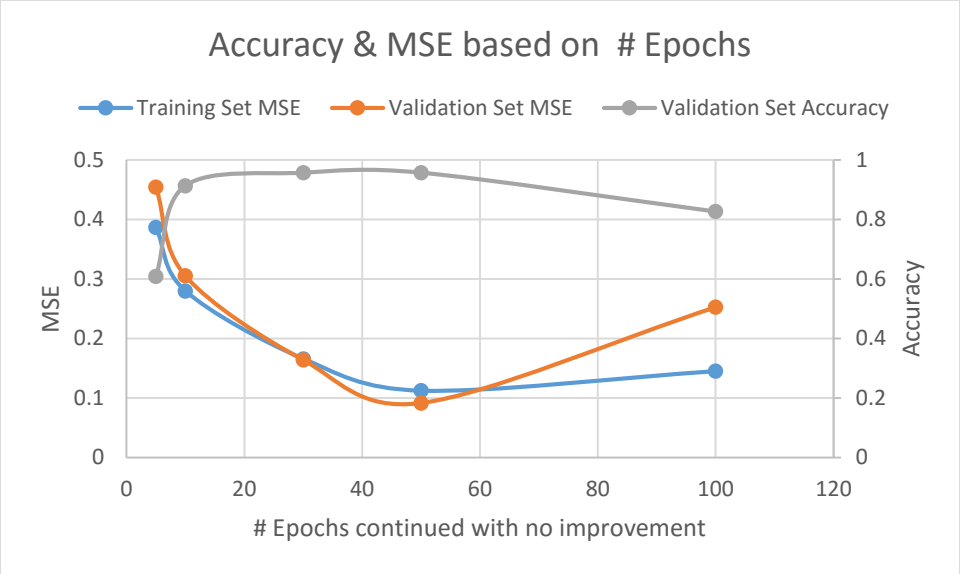
Backpropagation Report

Wesley Ackerman

Iris Dataset

I used backpropagation to teach a multi-layer perceptron the iris dataset. As instructed, a learning rate of .1 was used, with 75% of the data used for training. I experimented with the number of epochs to continue with no improved prediction accuracy on the validation set. The graph below illustrates mean squared error and accuracy as a function of the window of epochs.



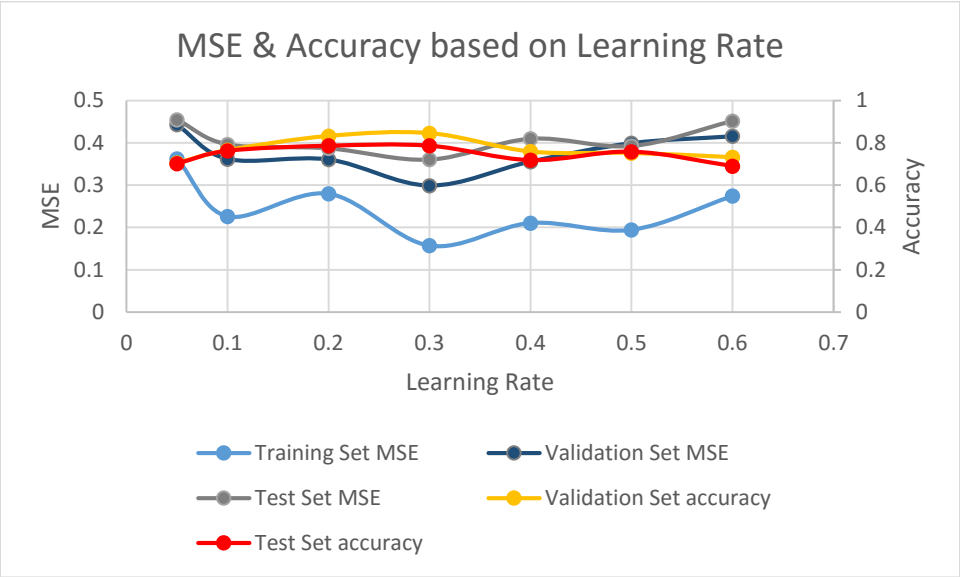
Notice that initially, continuing for more epochs has a large positive effect on accuracy. Jumping from 5 to 10 epochs brought accuracy from around 60% to around 90%. Continuing for 30 epochs brought a bit more accuracy, and continued to shrink mean squared error. Continuing for more epochs didn't bring much improvement. 100 epochs were run without improvement on the validation set, and this high number of epochs significantly decreased accuracy, and increased error. It appears that the optimal amount of epochs to continue, for my implementation, is around 30.

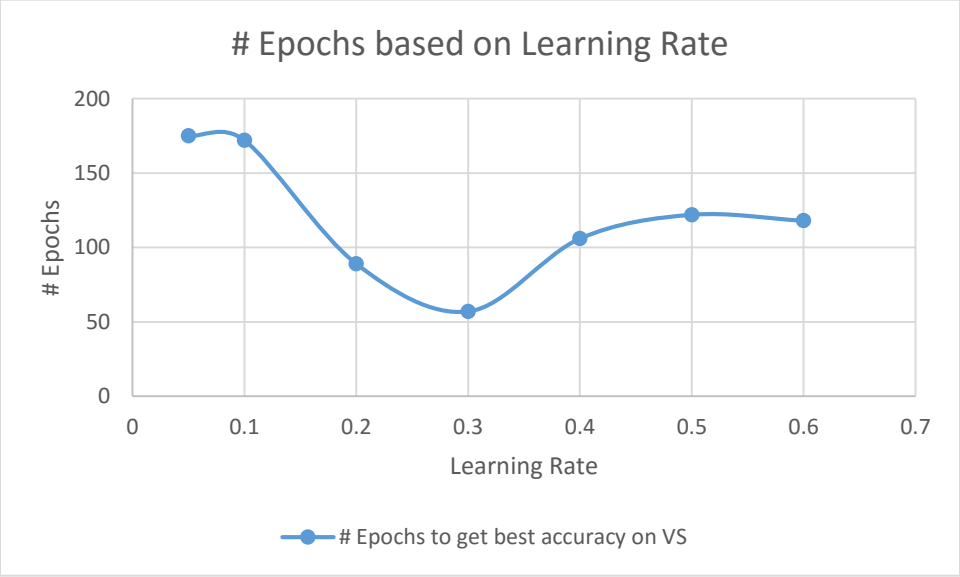
[*note: I was a bit unsure how to calculate mean squared error with nominal outputs. I settled for doing it in the following way: My implementation of the MLP has an output node for each possible output class (e.g. there were three output nodes for the iris dataset). So, to calculate MSE, I took the output from each node, and subtracted it from the expected for each node (zeros for all nodes but the one representing the expected output class). I squared each of those values, and added them together, representing the total squared distance from received output to expected output.]

Voting Dataset: Learning Rate

I ran backpropagation on the vowel dataset. I chose to remove train/test, speaker, and gender. Train/test and speaker were clearly not useful. The name of the person and the hypothetical dataset the instance belongs to have no relation to the way a vowel is pronounced. I chose to also remove gender, since it is unclear to me how closely related gender and pronunciation are. It is possible that gender would just confuse the MLP.

In experimenting with different learning rates, I found that a learning rate of 0.3 performed the best. It increased accuracy the most, to around 75%. It also decreased the number of epochs needed, to around 60. It makes sense that a good learning rate would improve accuracy and take less time to get results. Mean squared error for all sets was lowest with 0.3 as well. I attempted learning rates from 0.05 to 0.5, but at both ends of that spectrum error increased and accuracy decreased. My findings are graphed

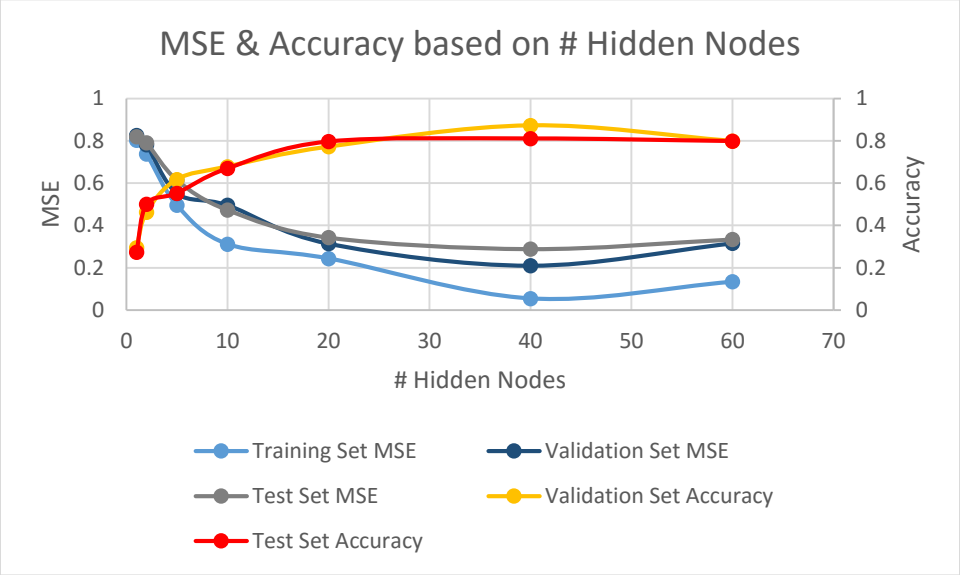


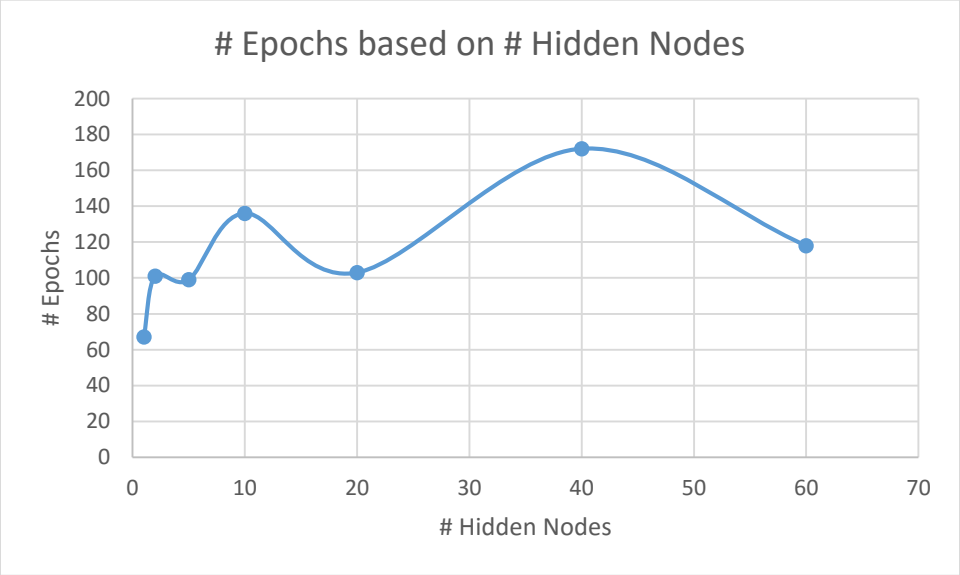


Voting Dataset: Hidden Nodes

In experimenting with hidden nodes, I found that 40 worked best of all the amounts I tried. At 40 hidden nodes, the mean squared errors were at their lowest, and accuracy was at its highest, above 80%. With more than 40 hidden nodes, I found that accuracy began to decrease slightly, and error increased more quickly.

In measuring number of epochs needed to reach best accuracy on the validation set, the data was less clear. There seems to be a fair amount of noise in the data, possibly because I didn't take the average of enough trials. However, it clearly took the most epochs when the MLP had 40 hidden nodes. It appears that with the best number of hidden nodes, the MLP takes longer, but gets more accuracy. It took around 170 epochs with 40 nodes, while trials with other amounts of nodes took significantly less.

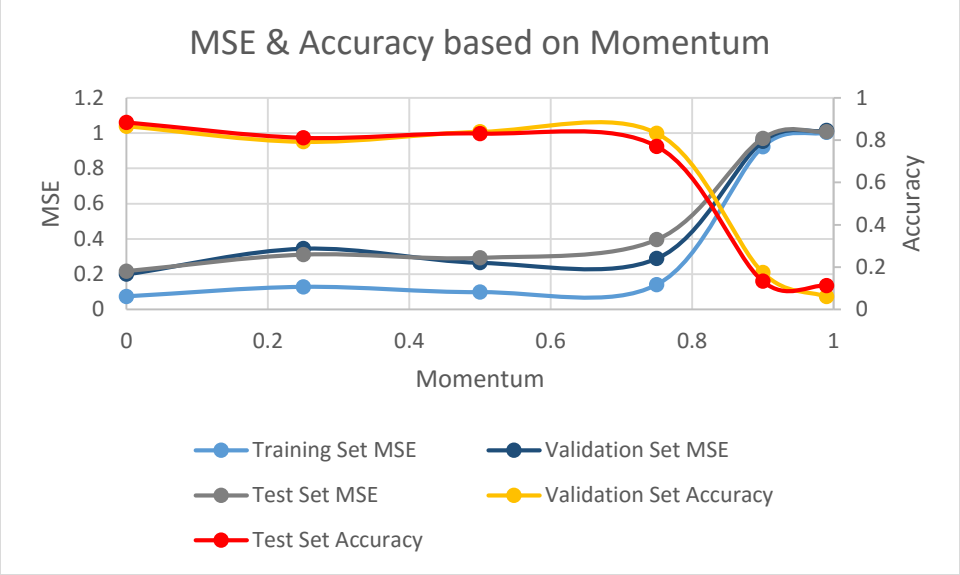


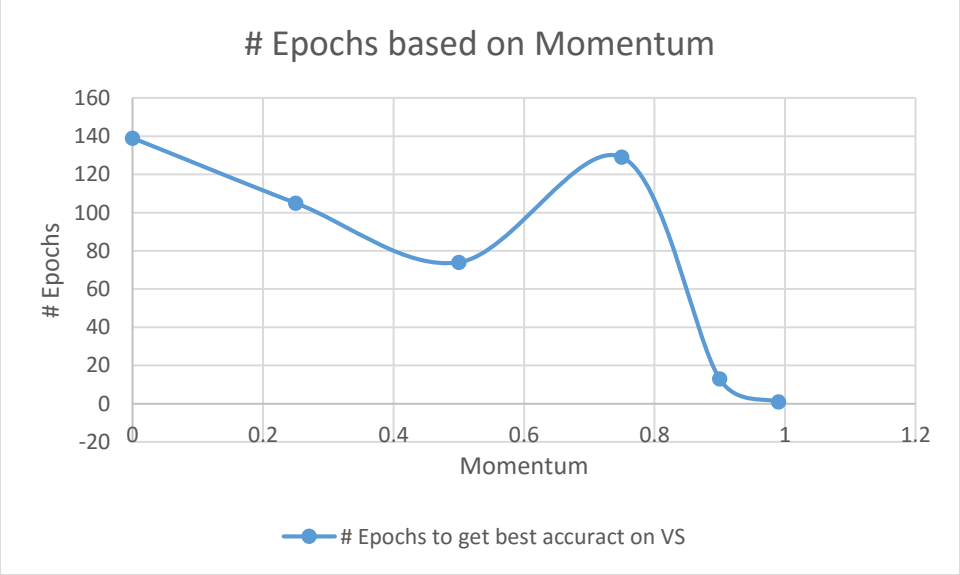


Voting Dataset: Momentum

In experimenting with momentum values, I found that a momentum of 0.5 performed best. It took a smaller amount of epochs for the algorithm to converge with momentum of 0.5. The accuracy wasn't negatively affected by this smaller number of iterations taken. Smaller momentum values took many more iterations, with no better accuracy. The large momentums (0.9 and 0.99) took very few epochs, but had terrible results. Their accuracy wasn't much better than the baseline.

Most momentums had little effect on accuracy and mean squared error. However, large momentums (0.9 and 0.99) had an extremely negative effect on the MLP. I think this is because the MLP keeps jumping around the minima, and because of the large momentum value it is unable to settle on a solid value. Error was much higher, and accuracy much lower, with these larger momentum values.





Experimentation

I was interested to see whether how a person pronounces vowels can be used to predict their gender. I modified the vowels dataset once again. I still removed the train/test and speaker features once again, since the prescribed data group and a small list of names would not generalize well. I modified the order of the dataset to place gender at the end, thus making it the target attribute.

The data was run using 10-fold cross validation. I ran the validation multiple times for each learning rate and number of hidden nodes that I tried. I found that 40 hidden nodes was best for this data, as with the

previous. Smaller numbers of hidden nodes were faster, but less accurate. I was interested to find that smaller learning rates worked much better this time around. A learning rate of 0.3 yielded around 85% accuracy, while 0.1 yielded around 90% and 0.05 around 95%.

It is truly amazing to me that the MLP was able to learn, with up to 95% accuracy, whether a speaker was male or female based on how they pronounced a word. It would be interesting to me to delve into what the vocal differences are, although that is probably hard to do with an MLP. I would be interested in studying this further, and learning about its implications in voice recognition. I'm sure study has been done on it in science and in industry.