CS 460

Assignment 2

Jeffrey Talada

**Building the code**

I wrote the client first so I’d know what methods I’d like to call. Then I started writing the code for a nueral network, beginning with the constructor and testing to make sure the network was constructed as intended. Then I streamed the training file one line at a time to set the input layer. I proceeded to write the code to train the network, including predicting the output then backpropagating the errors. Implementing w0 gave me lots of trouble as it was the exception to every rule. Finally, I implemented the testing code. Realizing it was half of the training code made me refactor a bit. I considered making it multi-threaded but, I didn’t want to lose the use of my computer, maybe in the future. It’s currently capable of doing multiple layers if you’re interested in testing it, but I didn’t include any cases on the graph.

**Experimentation**

I used a seeded random to generate the random starting weights. I first experimented with number of hidden units with a timer. I found that 16 HU takes about 7.5s per epoch, 32 HU about 10.8s per epoch, 64 HU about 21s per epoch, and 128 HU about 32.9s per epoch. More hidden units correlated with higher accuracy in general. Then I played with learning rate. I took samples for 16, 32, and 64 HUs at learning rates of 0.1, 0.05, and 0.01. The learning rate correlated with a faster rate of accuracy gain up to a point. After seeing the results for the various runs, I started a run with 128 HUs starting at a rate of 0.1, changing to 0.05 at epoch 20, then to 0.01 at epoch 35. This produced the best accuracy, peaking at 98.16%. Until this point all runs were at 60,000 training examples and up to 50 epochs.

I finally experimented with number of training examples doing runs at 10, 100, 1000, 10000, 20000, and 40000 examples with a learning rate of 0.1 and 16 HUs. Fewer training examples appear to limit the amount the network can learn.