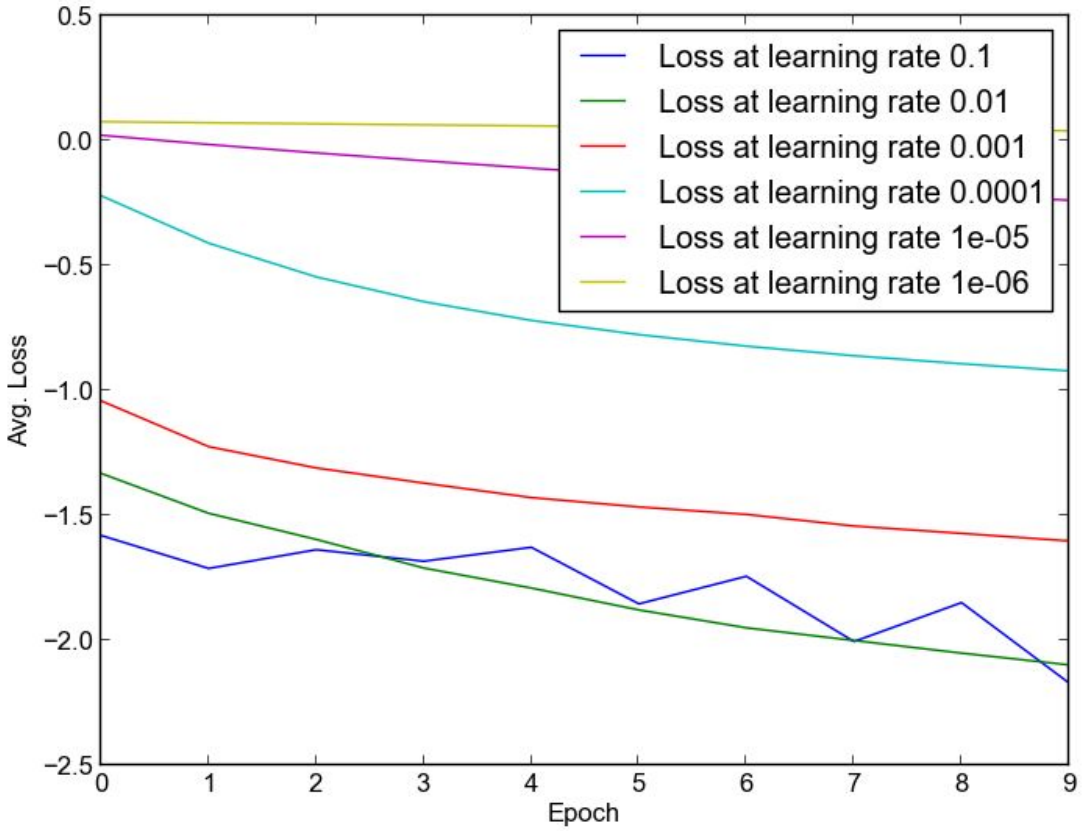
CS 434
Trevor Swope
William Selbie
May 15, 2018

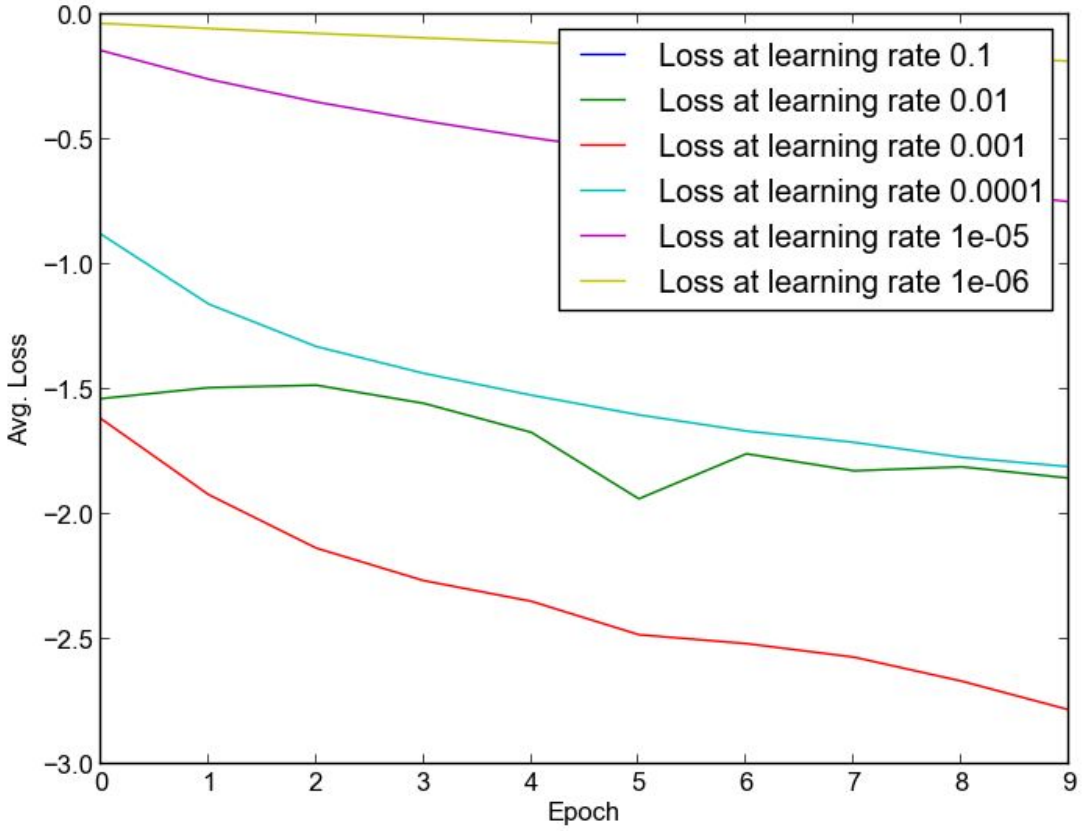
Assignment 3

1. Our best learning rates were 0.1 and 0.01, both resulted in reasonably high accuracy over a period of time, although clearly 0.1 oscillated more greatly. While the loss increases in magnitude as the learning rate increases, using learning rates of 0.1 and 0.01 resulted in the best validation and test set accuracy rates in the fewest epochs.



We stopped training after 10 epochs, as the CS434 server had very high load averages (70+) so we did not want to overload it more. After 10 epochs, the learning rate of 0.01 achieved an accuracy of 45.6% on the testing set.

2. When using Relu over sigmoid, we found the results to be slightly better. The accuracy on the validation and testing sets were continuously 2-4% higher after each epoch. In fact, after 10 epochs, with a learning rate of 0.001, Relu achieved 50% accuracy on the testing set; it is interesting that the ideal learning rate with Relu as an activation function was different than that for the Sigmoid function. We also found Relu to be slightly faster than Sigmoid. The overall trend given the same learning rate and number of epochs was similar to sigmoid, just slightly better.



3.

Momentum - The higher the momentum is, the less oscillation that the accuracy and loss had, we found that a momentum of 0.6 was consistently the best choice, given less

loss and higher accuracy. It makes convergence much smoother to have a high momentum, although past a certain point its effects were less noticeable.

Dropout - Changing the dropout rate did not significantly affect the results, although typically speaking lower dropout rates (0.1-0.3) resulted in higher accuracies. Going below 0.1 or above 0.3 tended to reduce accuracy more and more as well as increase average loss. We were unable to get plots of the data we saw for dropout as the python script did not finish running before the deadline.

Weight decay - We were unable to test weight decay before the deadline due to load on the servers, but we suspect that with a more significant weight decay, there will be less overfitting, as weight decay is a form of regularization. This will result in slower convergence, but there is likely a sweet spot that doesn't slow down convergence too much but keeps weights at desirably low values.

4.

For this section, we chose to use a learning rate of 0.01, and the sigmoid activation function. Running for 4 epochs, it quickly converged to 42.5% accuracy on the testing set. In fact, after only 1 epoch, it achieved 35% accuracy on the validation set. This is a significantly faster convergence than that we achieved for parts 1 and 2. This is interesting, and I would predict that with more hidden layers, you may converge to a higher accuracy faster. You can observe the convergence in the following two figures: