Assignment 2

Finn Womack

February 2020

1 Write a function that evaluates the trained network, as well as computes all the sub gradients of W_1 and W_2 using backpropagation

Implemented in mlp.py

2 Write a function that performs stochastic mini-batch gradient descent training

Implemented in mlp.py

3 Training

3.1 Learning Rate Tuning

I initialized my weight matrices/vectors randomly according the a normal distribution, N(0,0.1). Keeping them close to zero but not on zeros as to avoid getting stuck there off the bat.

I first tried a learning rate of 0.1 but found that was too large as it caused the output to explode and eventually overflowed the loss function returning NaN values. I then kept reducing by a factor of 10 until It stopped blowing up. After trying a few learning rate I found that the learning rate of 0.00001 gave me the best performance after 50 epochs.

3.2 Hidden units tuning

I started with 300 hidden units but found that it caused to learning process to go a bit slow so I decided to try the other end of the spectrum. I then tried 5 hidden units and that gave me poor performance likely because it was too simple so I started increasing the number and after trying a few different amounts of hidden units until I settled on 40 which seemed to give me good performance while not taking too long to train. Beyond that there seemed to be diminishing return.

3.3 Batch Size tuning

I initially started with 1000 batches and found that it seemed to work well. I tried a few different batch sizes and found that if I increased the batch size it seemed to preform a bit worse and if I decreased the batch size it preformed a bit better. In the end a batch size of 5000 worked best for me.

3.4 Momentum tuning

I initially started with a momentum value of 0.8 and played around with a few different values. Though I found that momentum didn't have as pronounced of an effect of the performance as some of the other parameters. In the end I used 0.85 as that gave me the best results.

3.5 Best Accuracy

I was able to achieve the best accuracy with a learning rate of 0.00001, 40 hidden units, 5000 batches, and a momentum of 0.85. With this setup I was able to achieve an accuracy of 80.35%.

4 Epoch monitoring

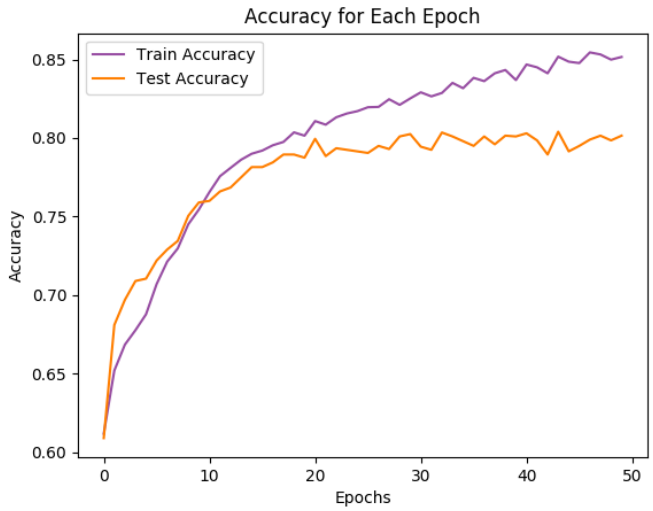

Figure 1: While testing accuracy out-preformed training at first it started to level off while training kept increasing. However since testing accuracy stayed relatively constant it does not appear to have started over-fitting yet

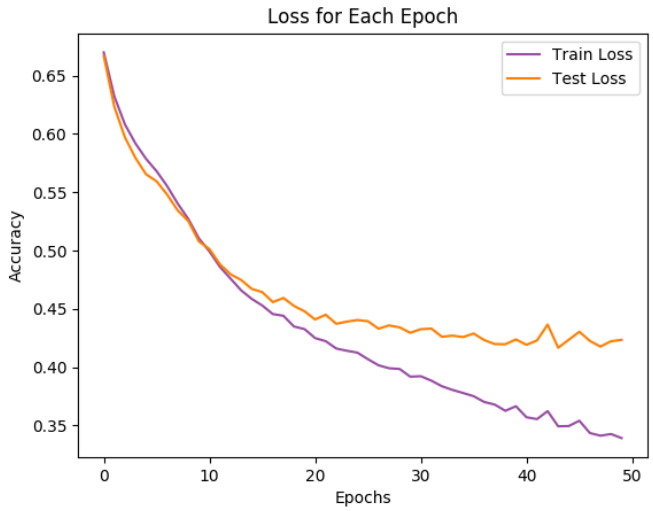

Figure 2

Interestingly enough my testing accuracy out-preformed my training accuracy for the first several epochs though eventually dipping below after about 10 epochs. After about 20 epochs the accuracy seems to level out converging on around 80%. When look at the loss it does seem to jump around a but but this is to be expected with the smaller batch size I used for stochastic mini-batch gradient decent.

5 Tuning plots

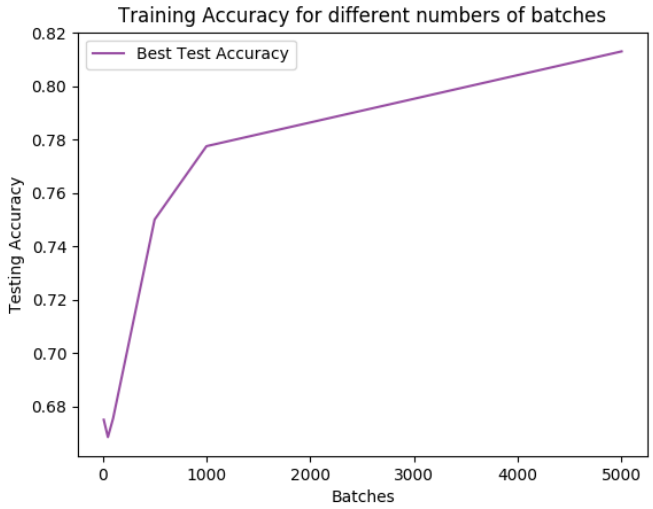

Figure 3: Smaller batches seemed to preform better.

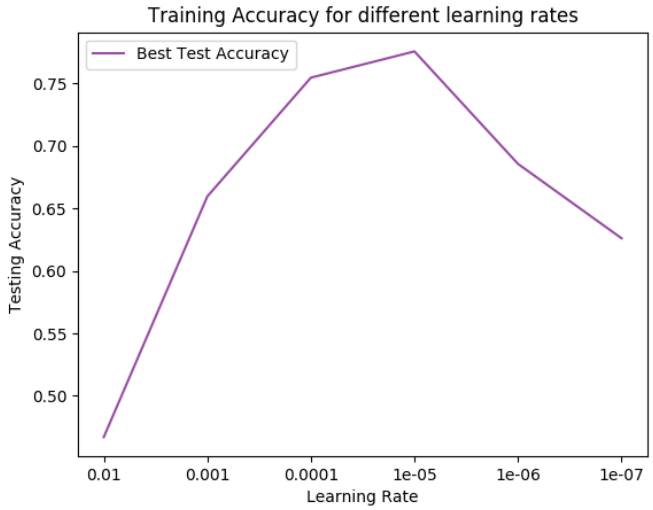

Figure 4: The learning rate preformed best at 0.00001

Figure 5: While more hidden layers did better training slowed down and the minimal increase in accuracy past a certain point didn't seem worth the slow down.

6 Discussion

Overall I found the tuning process to be quite finicky and brittle. Getting the learning rate and the initialization weights just right took a bit of time. Once I was able to get the parameters right it seemed to work pretty well albeit with the accuracy jumping around a bit due to the smaller batch size. I know it was mentioned in the assignment but the learning rate was definitely the most important part in my experience. Too small and everything would explode too large and it would converge way too slowly. A couple of times I thought my code was broken but it was really just the learning rate that need to be adjusted.