DD2424 Deep Learning in Data Science

Assignment 2

Ali Shibli

May 2021

1 Introduction

In this part, I will report on the bonuses that I tackled from assignment 2 of the Deep Learning Course. I discuss the solution for the following 4 parts:

1. (1.b) You could also explore whether having more hidden nodes improves the final classification rate. One would expect that with more hidden nodes then the amount of regularization would have to increase.

2. (1.d) Apply dropout to your training if you have a high number of hidden nodes and you feel you need more regularization.

3. (1.e) Augment your training data by applying a small random geometric and photometric jitter to a training image each time you encounter it during training. You can do this on the fly by applying a random jitter to each image in the mini-batch before doing the forward and backward pass.

4. (2) Read reference [Smith, 2015] and set eta min and eta max to the guidelines in the paper. In the assignment I gave you values I had found for eta min and eta max. You should write code to find good values for these quantities for yourself and for networks with a different number of hidden nodes and regularization.

The network by default is the 2 layers network, with 50 units, unless specified. In addition in the experiments, unless specified as well, I will use the best parameters gotten from the main assignment 2 for regularization as well as the same parameters for the Mini-Batch function as follows:

- lambda = lambda_best = 0.004
- eta_min = 1e-5
- eta_max = 1e-1
- batch_size = 200

- number of cycles = 3
- number of sample per batch = 2*45000//batch_size

2 More hidden nodes

In this part, we investigate whether making the network bigger and changing the number of hidden nodes will affect the network. We try with 4 different numbers of hidden nodes, and report their corresponding test accuracies:

Number of hidden units	Test Accuracy
50	0.4911
100	0.5079
150	0.5156
150	0.5204

Table 1: Dropout rate vs test accuracy results

We notice that as the number of hidden nodes increases, the test accuracy increases as well. This is a little expected since as we increase the number of nodes, we are increasing the complexity of the model, thus being able to learn more from the data. However we should not forget that at some point we might start overfitting. The graphs for loss and accuracy on the training set were as follows:

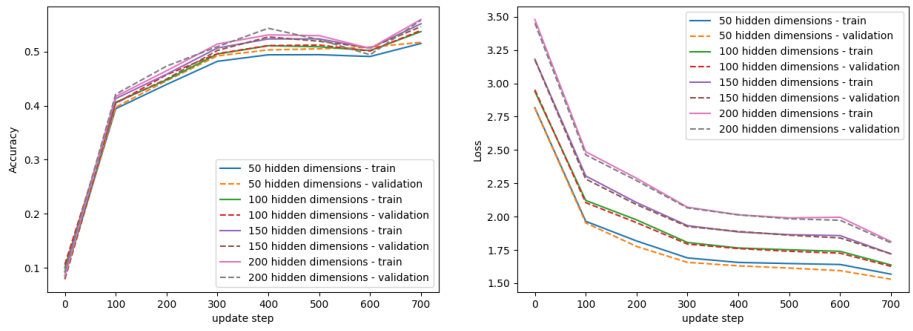

Figure 1: Loss-Accuracy for the different models

3 Adding Dropout

We try in this case to add more regularization to the network by applying dropout. We investigate with 3 different dropout rates, and report their test performances here:

Dropout Rate	Test Accuracy
0.3	0.4174
0.5	0.477
0.7	0.4936

Table 2: Dropout rate vs test accuracy results

We notice that as the dropout rate increases, the test accuracy increases, thus the performance of the model becomes better. In the following plots we show the evolution of the loss and accuracy graphs of this model with 0.7 dropout rate:

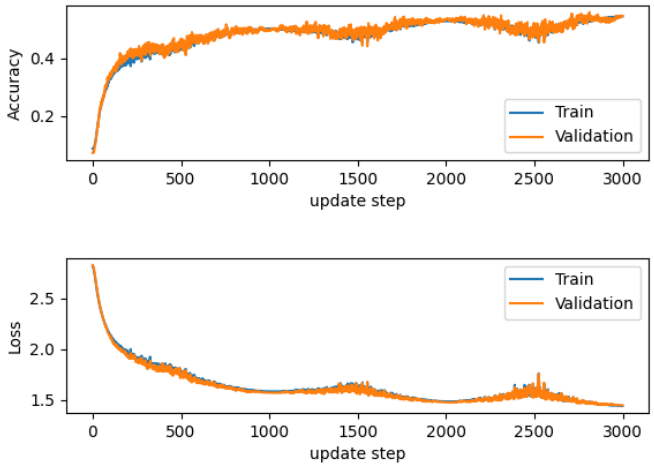

Figure 2: Loss-Accuracy when applying dropout with rate = 0.7

4 Augmenting with jitter

Finally, we experiment with adding jitter noise to the model dataset after each batch. We add normal gaussian noise with 0 mean and 0.1 standard deviation. However, we notice that test accuracy drops to 0.4934, a drop of about 2-3%. Thus we realize that in this case adding noise wasn't very helpful in this case. The plots for training loss and accuracy are shown as well:

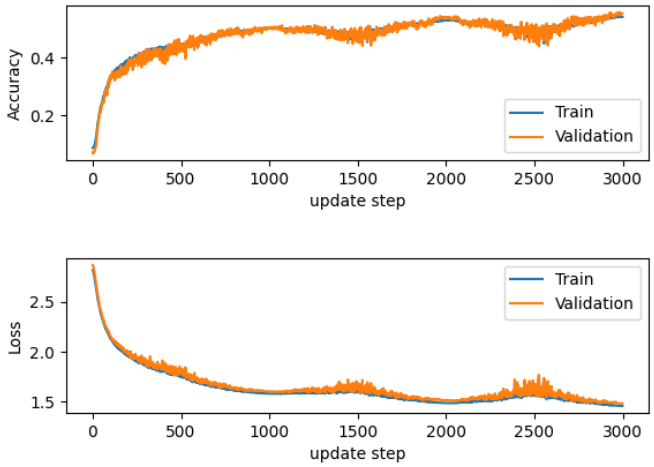

Figure 3: Loss-Accuracy curves when applying jitter noise

5 Smith Learning Rates

The author in [1] proposes the cyclical learning rate (CLR) mechanism for choosing best learning rate values during training. He justifies the reasoning behind using this cyclical learning rate from the topological structure of the loss function. As well, he mentions that he works particularly with triangular learning rate update, as we are doing in the assignment, in which the rates increase linearly from a min val to a max value, and then vice versa. He continues with triangular update since it seemed from his experiments that its similar to other geometrical functions in terms of performance results. To our case, we follow his method for finding the minimum and maximum learning rates for our triangular CLR updates. We plot the accuracy reported vs the corresponding learning rates, and find the points where the accuracy is increases to set minimum for the CLR, and then the point where the accuracy starts decreasing to set the maximum for the learning rate. Using this reasoning, we vary first the learning rate from 0.08 to 0.1, and make sure it finishes 1 half cycle in between (1half cycle = 1 n_s = 1 number of steps per cycle). The plot for this distribution of accuracy vs learning rates is shown below:

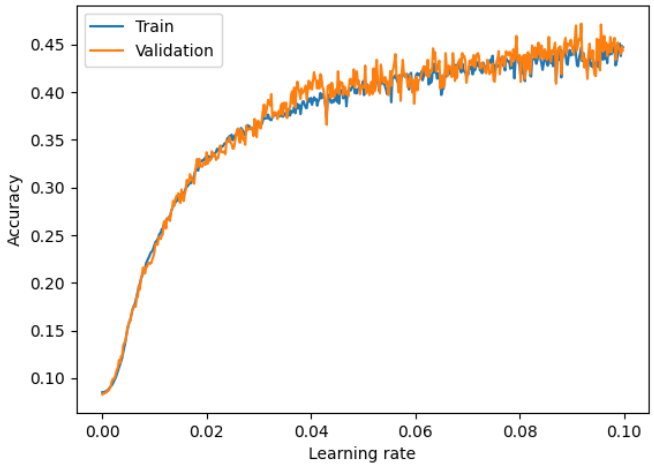

Figure 4: Accuracy vs learning rate to find best learning rate values

We notice how the accuracy starts increasing at learning rate = 0.04 and starts the process of converging. Thus, we can set $eta\_min = 0.04$ and $eta\_max = 0.1$. Setting these values results in the following plots for accuracy and loss over the training:

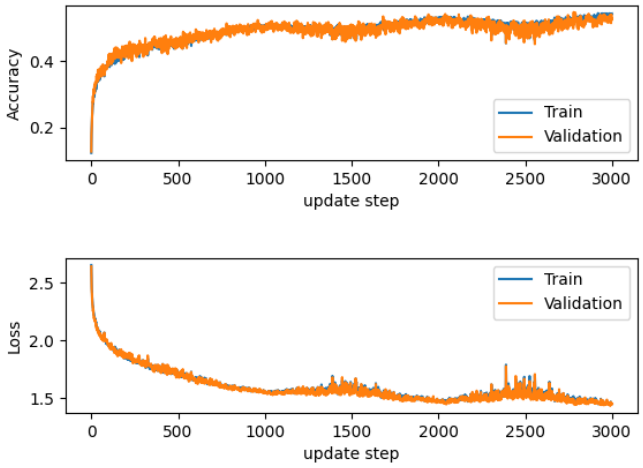

Figure 5: Accuracy vs learning rate to find best learning rate values

The reported test accuracy gotten is then 0.508.

References

[1] Leslie N. Smith. Cyclical learning rates for training neural networks, 2017.