DD2424 Deep Learning in Data Science

Assignment 3 - 23rd July, 2019

Musab Tayyib Gelisgen - gelisgen@kth.se

In this assignment, a k-layer neural network with a cyclical learning rate and batch normalization is implemented with experiments. Back-propagation and mini-batch gradient descent is used while training the neural network. The tests done are provided below.

1. Gradients

I have used centered difference method to calculate the gradient values and checked them against the values obtained by the formula. When tried with a bunch of different parameters, the resulting relative errors are so small (less than 10^{-6}) that led me to think that the calculations are correct. Then I concluded that I managed to write the code without any bugs and correctly.

2. Evolution of the loss

1. 3-layer

In this experiment, a 3-layer neural network is used with 50 nodes in the hidden layers. While training, batch normalization is applied to one of the models. He initialization is used while setting the parameters. I trained the models for 40 epochs.

The final losses are 1.002 when batch-normalization is applied and 1.192 when batch-normalization is not applied.

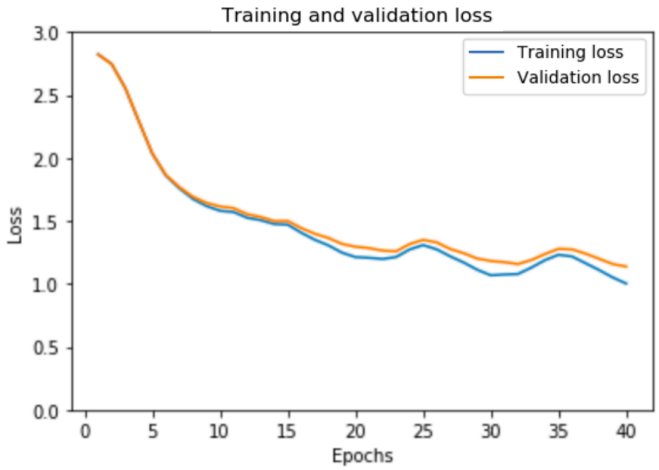

Figure 1: Loss when batch normalization is applied on 3-layer

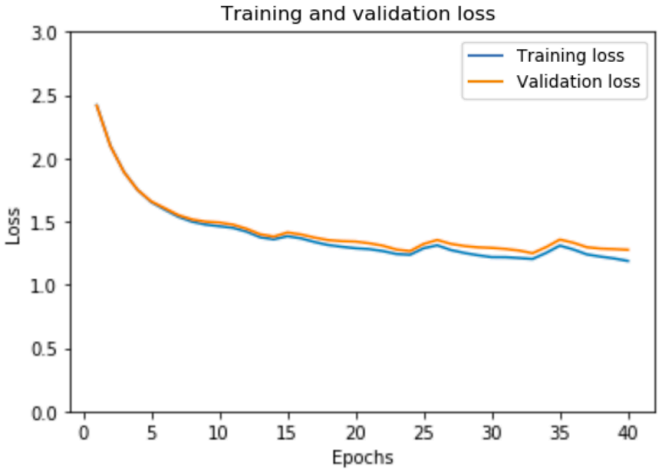

Figure 2: Loss when batch normalization is not applied on 3-layer

2. 9-layer

As a result from the figures, batch normalization has the affect of decreasing the loss in the later epochs. Its affects are greater with the wider model (9-layer) but the positive affects on the shallow (3-layer) are still visible. The final losses are 1.108 when batch-normalization is applied and 1.512 when batch-normalization is not applied.

Figure 3: Loss when batch normalization is applied on 9-layer

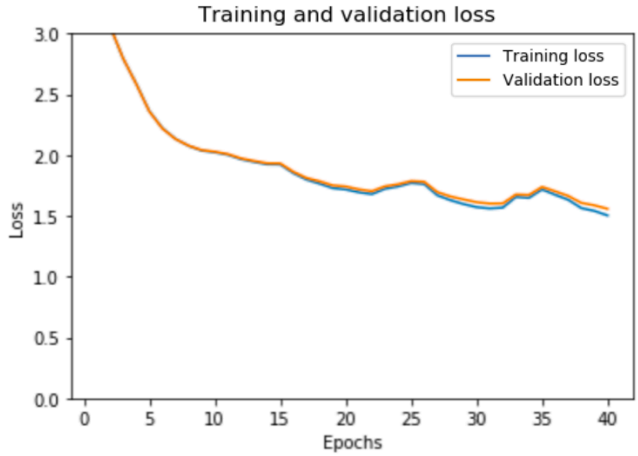

Figure 4: Loss when batch normalization is not applied on 9-layer

3. Setting lambda

Coarse to fine random search method is used in this assignment when trying to optimize the lambda. First, I began with a rather large range of random lambda values and then I tested the accuracy in the validation set with these random lambda values. The first lambda values used are 1e0, 1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7, 1e-8, 1e-9 with 20 epochs.

The best accuracy is obtained when lambda is set to 1e-2. This helped me look at a narrower range much closer so I continued with looking around the 1e-2 range closer. I have chosen my new interval as 5e-3 and 5e-2 and divided that interval into 10 pieces. Then I have continued my experiments by using these new lambda values.

Figure 5: Accuracy when lambda is set to 7e-3

Figure 6: Loss when lambda is set to 7e-3

The best accuracy I have obtained is **0.5401** when lambda is set to 7e-3. The plots regarding the best experiment are shown above. This model is run for 30 epochs and the α is set to 0.7.

4. Sensitivity to initialization

In the last experiment, I have 6 models where I performed 3 of them with batch normalization and 3 without batch normalization for 3 different values of sigma. Models are trained with 20 epochs. From the results, it can be easily seen that batch normalization provides a stability against the initialization of parameters. In the models where there is no batch normalization, the resulting accuracy is extremely dependent of the first initialization of the parameters. As the sigma becomes

smaller and smaller, the models start not to learn anymore. In addition, the batch-normalized models' accuracies are good and very similar to each other which shows that batch normalization balances the affect of parameters. Figures for the experiments are provided below:

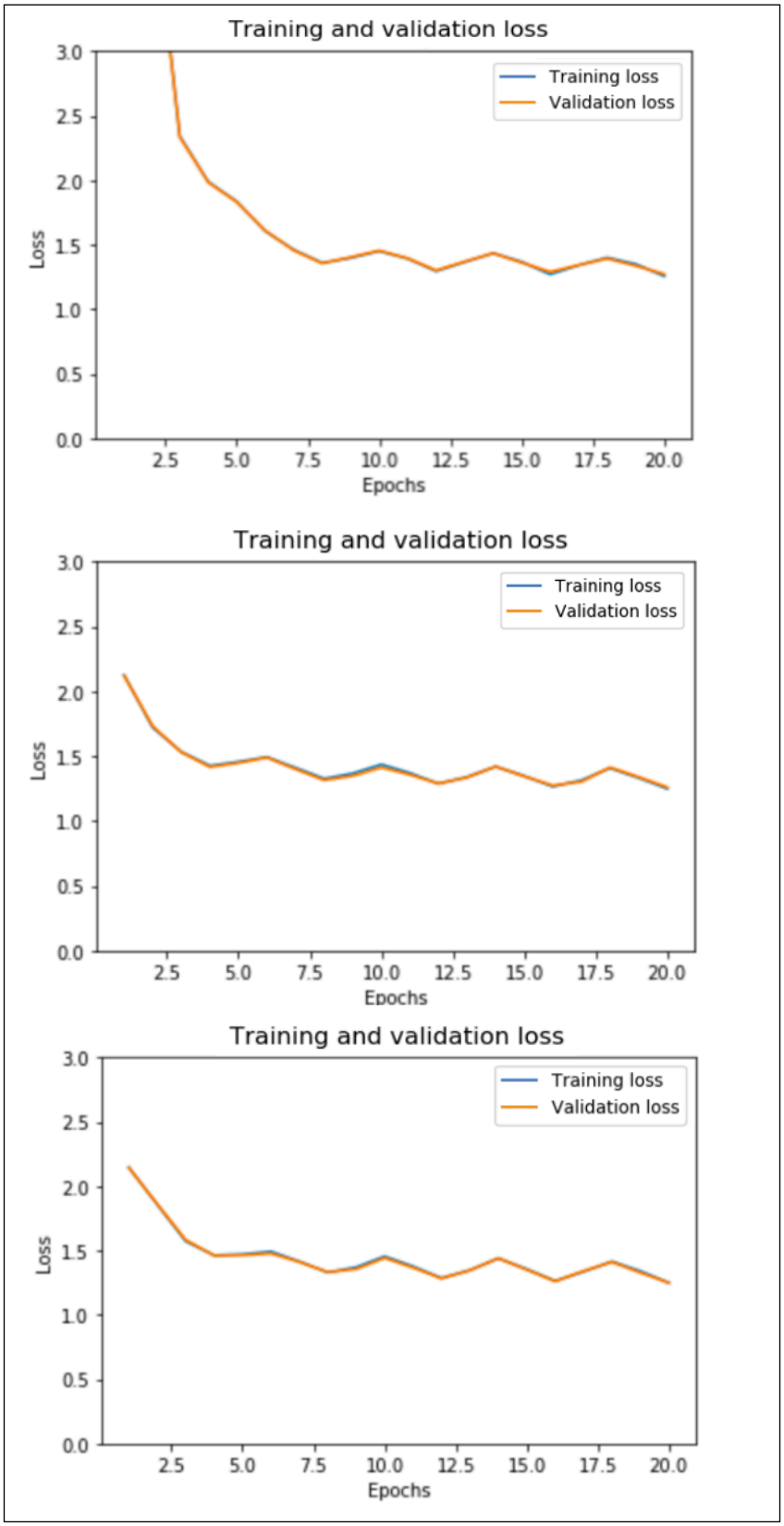

Figure 7: Loss with batch normalization for sigma=1e-1, 1e-3, 1e-4

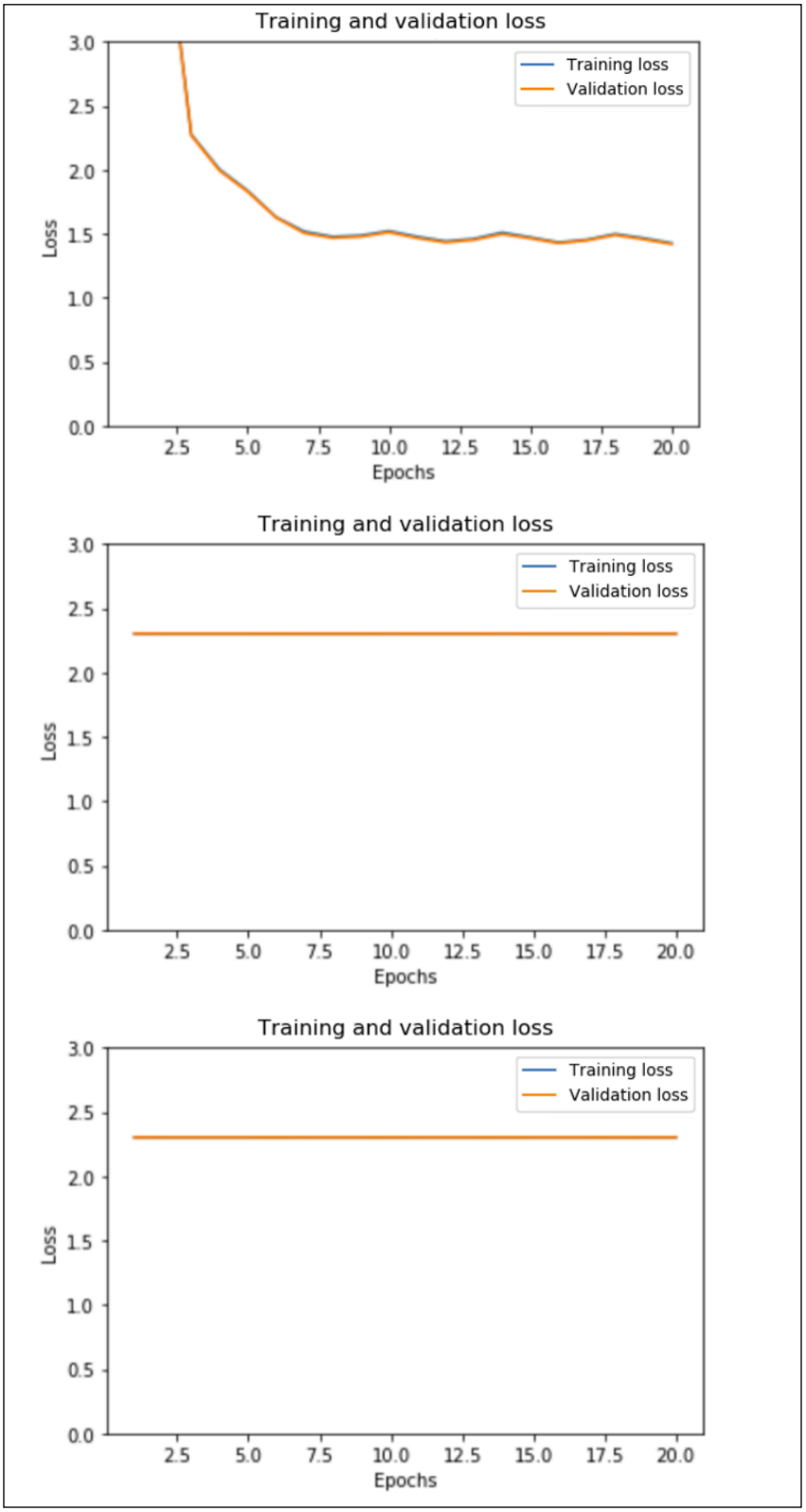

Figure 8: Loss without batch normalization for sigma=1e-1, 1e-3, 1e-4

Sigma	Accuracy with batch normalization	Accuracy without batch normalization
1e-1	0.5289	0.1974
1e-3	0.5284	0.1
1e-4	0.5298	0.1

Table 1: Accuracies of models with different sigmas for batch-normalization and no batch-normalization