Kungliga Tekniska Högskolan

DD2424 Deep Learning in Data Science
Assignment 1 Bonus

Quentin LEMAIRE

April 10, 2018

1 Optimization

In order to see the improvement of my optimizations, I put a seed for the randomness of the network. I started with no optimization and with the following parameters: eta = 0.01, lambda = 0, batch size = 100 and iteration number = 40. I had an accuracy of 0.3667. In order to test an optimization it is not good to put a precise seed for the randomness but it allows me to really compare the affect of one changement.

1.1 Xavier initialization

First, I tried to put the Xavier initialization of the weights without changing any other parameter. My accuracy went up to 0.3668. It did not changed a lot but it is still a bit better. Then, I tried the He initialization which gave me 0.3671.

The two initializations didn't change the accuracy a lot but I kept the He initialization as it was slightly better. To confirm it, we would need to try it over different random seeds.

1.2 Dataset augmentation

By using more of the available data, my score increasing to 0.3942. It is the bigger increase between all optimizations. With this optimization, the dataset was more than 4 times bigger.

1.3 More training

To make use of all this data, we might need to train longer. Therefore, I increased the number to iteration to 150 and to kept the best epoch by testing it with the validation set. I got an accuracy of 0.3966 which is a little increase.

1.4 Shuffling of the training set

In order to not see the dataset in the same order, I shuffle the dataset at every epoch. To have a good generalization, the training set should not follow a specific pattern. It increased my accuracy of about 1% and I got up to 0.4064.

1.5 Decaying of the learning rate

The last thing I implemented is the decaying of the learning rate. I was able to raise the initial learning rate up to 0.05 and I was reducing the learning rate of 5% at every epoch. I got an accuracy of 0.4103 this optimization.

1.6 Dataset centering

I also tried to make a modification to the dataset so it has a mean of 0. It improved the result and I got an accuracy of 0.4129.

1.7 Conclusion

The optimizations with the biggest effect was the augmentation of the dataset and the shuffling of the dataset. We can see here that maybe the most important task to train a network is to have a good and relevant dataset. By applying a grid search, I got better parameters and finally, I had a loss of 41.43%. I applied the following parameters:

- learning rate: 0.05
- learning rate decay factor: 0.95
- Regularization coefficient: 0
- Epoch number: 500
- Bach size: 100

I also reduced the the size of the validation set to 1000 and I got slightly better results.

2 SVM multi-class loss

Finally, I rewrote the loss and the computation of the gradients to have a SVM multi-class loss. One of the problems that I faced is that the SVM multi-class loss is more expensive to compute for me. I had to wait around 6.2 seconds for one epoch with the SVM multi-class loss when it is less than 2 seconds for the cross-entropy loss. Therefore, I have to reduce the numbers of epochs in order to compare the two losses.

For several combinations of parameters with the same parameters and same number of epochs, I got the following results:

- For SVM multi-class loss: 0.3782, for cross-entropy loss: 0.4083
- For SVM multi-class loss: 0.3761, for cross-entropy loss: 0.4134
- For SVM multi-class loss: 0.3976, for cross-entropy loss: 0.4097

We can see that for every example the cross-entropy loss is better. But it is hard to draw a real conclusion with this amount of test I made. And sometimes the difference is not too big.

It is possible that the SVM multi-class loss would work better on some parameters where the cross-entropy loss does not perform well.