Deep Learning in Data Science, DD2424

Short report on lab assignment 3

Classifier k-layers

Anna Sánchez Espunyes

May 3, 2021

1 Main objectives and scope of the assignment

In this assignment you will train and test k-layer networks with multiple outputs to classify images (once again) from the CIFAR-10 dataset. You will upgrade your code from Assignment 2 in two significant ways. Firstly, generalizing your code so that you can train and test k-layer networks. Secondly, incorporating batch normalization into the k-layer network both for training and testing.

2 Methods

In this assignment we worked using Python and some of its well-known libraries, including matplotlib, numpy. We have also used the PyCharm IDE and Github.

3 Results and discussion

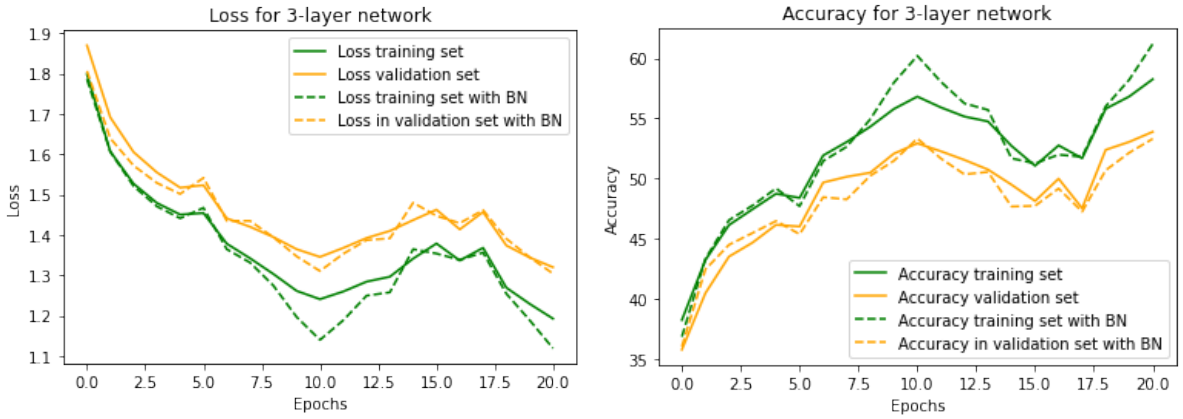
3.1 Gradient calculation

Gradient calculation is a key step when updating the weights of the k-layers using batch normalization. In the assignment the gradients are calculated analytically but an extra numerical function is used to check if the calculations are precise. After checking whether the calculations are correct, it can be stated that the computations are accurate. I have compared the numerically and analytically computed gradient by examining their absolute differences. The absolute differences is in the order of 1e-10, which is a threshold small enough to consider the calculations valid. For instance, the maximum absolute difference for each weight matrix, bias vector, gamma vector and beta vector in a 3-layer network is the following

Gradient W1	8.53e-10
Gradient W2	4.61e-10
Gradient W3	4.46e-10
Gradient B1	2.221e-10
Gradient B2	4.446e-17
Gradient B3	5.13e-10
Gradient Gamma1	3.03e-10
Gradient Gamma2	1.61e-10
Gradient Beta1	3.43e-10
Gradient Beta2	3.49e-10

3.2 Training a 3-layer network

Once the gradients are well-calculated it is possible to train a 3 layer network with 50 and 50 hidden nodes in the first and second hidden layers respectively. I will train the network with and without using batch normalization, in order to be able to extract conclusions. For the following experiment, I have used 45000 training examples, He initialization and shuffling the training examples after each epoch. The results can be seen in Figure 1.



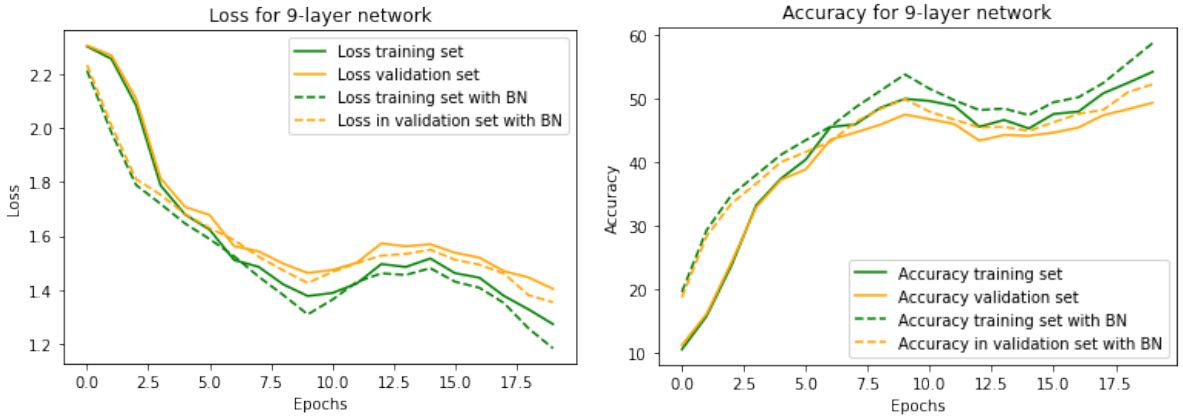
(a) Loss with and without BN

(b) Accuracy with and without BN

Figure 1: 3-layer neural network

In Figure 1a and in Figure 1b it can be seen that by using Batch Normalization we obtain better results. A lower loss and a higher accuracy is obtained both in the training set and in the validation set when BN is used. Another important metric to compare the results is the test accuracy. **The test accuracy obtained when training a 3-layer network without BN is 52.7 whereas when using BN the test accuracy is 52.82**.

3.3 Training a 9-layer network



(a) Loss with and without BN

(b) Accuracy with and without BN

Figure 2: 9-layer neural network

Now that a 3-layer network has been trained, it is possible to extend the network to a 9-layer network whose number of nodes at the hidden layers are [50, 30, 20, 20, 10, 10, 10, 10]. The network will be also trained with and without BN, to test the effect of BN in deeper networks. The parameters on which the network has been trained are the same ones as in the 3-layer network. The results obtained can be seen in Figure 2.

In Figure 2a and in Figure 2b it can be seen that by using BN we obtain notably better results than without using BN. A lower loss and a higher accuracy is obtained both in the training set and in the validation set when BN is used. **The test accuracy achieved without using BN is 49.43 whereas the test accuracy achieved with BN is 51.21**. It can be seen that compared to the test accuracies achieved in the 3-layer network, the performance has dropped quite a bit when not using BN. As the network becomes deeper it is more hard to train and BN is a great technique to overcome this problem.

3.4 Lambda search

A wide exploration of the regularization term will be performed in order to find a narrower range. I have chosen 6 values between 10e-6 and 10e-1 drawn from a uniform distribution. When training each network, I have used 2 cycles of training. The validation accuracy for the 6 different lambda values can be found in Table 1

Lambda	Validation accuracy
1e-06	52.02
1e-05	52.7
0.0001	51.68
0.001	52.86
0.01	53.3
0.1	43.78

Table 1: Validation accuracy for different regularization terms

Now that we have detected the most promising regularization terms, we can do a more narrow search. The range will now be between 10e-2.5 and 10e-1.5, using 2 cycles for training. The validation accuracy for the 10 different lambda values can be found in Table

Lambda	Validation accuracy
0.0031	53.24
0.0041	**53.58**
0.0053	**53.28**
0.0068	**53.92**
0.0088	53.02
0.0114	52.82
0.0147	51.72
0.0189	50.78
0.0245	50.36
0.0316	49.9

Table 2: Validation accuracy for different regularization terms

The final step will be to train the top-3 networks with 3 cycles of training. The test accuracies obtained with the top-3 networks can be seen in Table 3.

Lambda	Test accuracy
0.0041	**54.06**
0.0053	**53.88**
0.0068	**53.3**

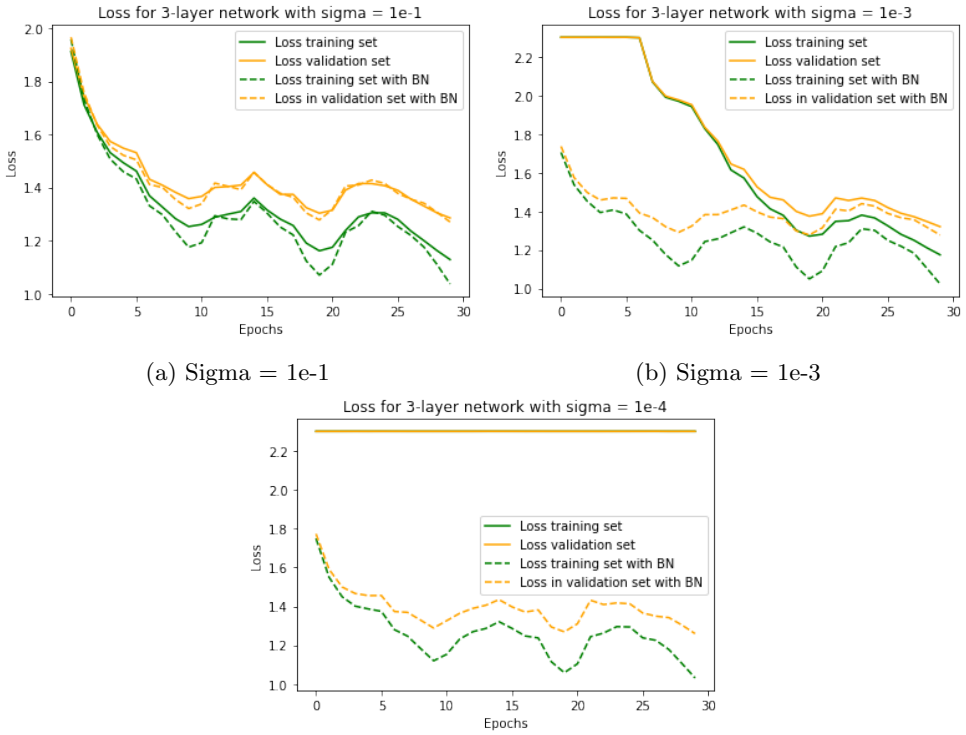
Table 3: Validation accuracy for different regularization terms

The parameters of the network used to achieved this result are the following: number of batch = 100, cycles = 3, ns = 5*45000/nbatch, etamin = 1e-5, etamax = 1e-1.

3.5 Sensitivity to initialization

The next experiment consists of initializing the network with a normal distribution with an equal value of sigma in all the layers. This test will be performed in three different values of sigma: 1e-1, 1e-3 and 1e-4 and with and without Batch Normalization (see results in Figure 3).

It can be seen that using Batch Normalization leads to better results for all the different initialization (the dashed lines have always lower loss). An interesting experiment is when using sigma = 1e-4 (see Figure 3c). For this case, there is no convergence without using BN whereas when using BN the results are satisfactory. Batch Normalization is a technique that helps to rely less in initialization and makes the training more stable. This method is very useful when training very deep networks.



(a) Sigma = 1e-1

(b) Sigma = 1e-3

(c) Sigma = 1e-4

Figure 3: Loss with and without Batch Normalization for different initialization