DD2424 Deep Learning in Data Science: Assignment 1

Romain Trost

April 2022

Exercise 1: *Training a multi-linear classifier*

The task for this assignment consisted of building and training a single layer network with multiple output labels. The network was trained using mini-batch gradient descent to classify images contained in the CIFAR-10 dataset.

Analytical and numerical gradients
To make sure that the analytically computed gradients were accurate, the results were compared to those computed numerically over small batches (20 first samples). More specifically, the absolute error difference between both computations was measured. Two tests were implemented to make sure this absolute error was low enough. Firstly, the percentage of gradients for which the absolute error was below the threshold of $1e-6$ was recorded. Secondly, the maximum absolute error was recorded. The results are shown below.

```
Percentage of weight errors < 1e-6: 100.0
Percentage of bias errors < 1e-6: 100.0
Max weight error: 6.406758017163394e-08
Max bias error: 4.752002508157105e-08
```

Figure 1: Error between analytically and numerically computed gradients.

Results of model learning under different configurations]
The training, validation and test data used for each configuration are "data_batch_1", "data_batch_2" and "test_batch, respectively. The resulting learning curves, weight matrices and test scores for the 4 different configurations are given below.

Configuration 1: *lambda=0, n epochs=40, n batch=100, eta=.1*

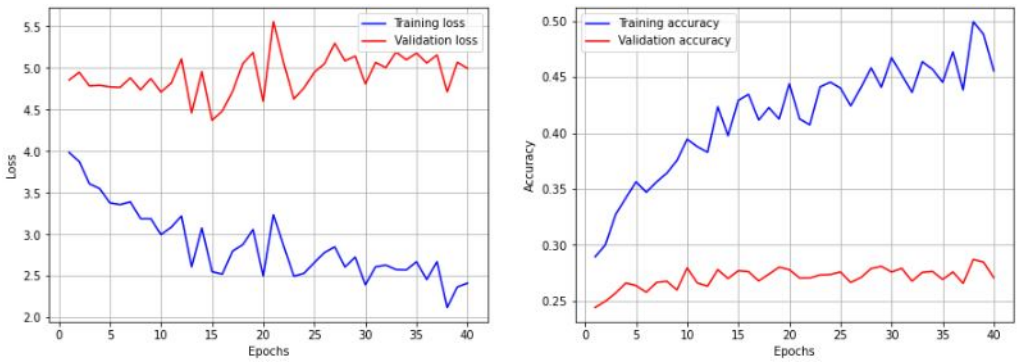

Figure 2: Loss and accuracy learning curves.

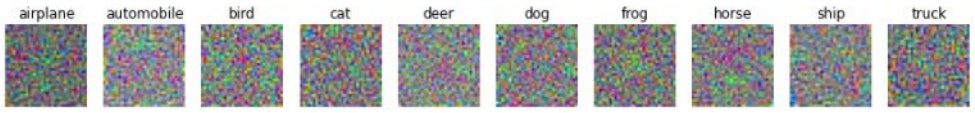

Figure 3: Learnt weight matrix after completion of training.

As shown, there are high fluctuations in the learning curves however the validation learning curves don't seem to improve throughout the learning process and hence, the model isn't able to classify new data well. This behaviour is most likely due to the big value assigned to the learning rate, causing the overshooting when trying to reach a minima during gradient descent. The weight matrices are shown to be blurry and the accuracy obtained on the test data after the training process is 27.27%.

Configuration 2: *lambda=0, n epochs=40, n batch=100, eta=.001*

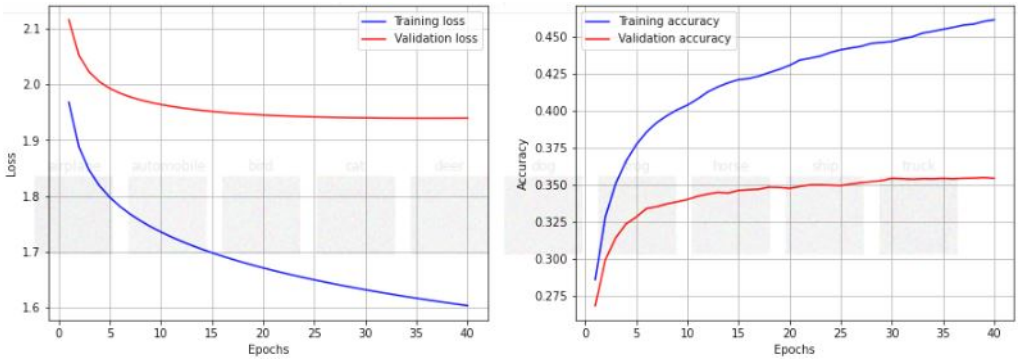

Figure 4: Loss and accuracy learning curves.

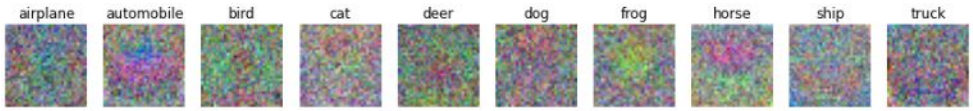

Figure 5: Learnt weight matrix after completion of training.

Here we can see that the model does a much better job at learning seeing as both the validation and training loss are decreasing and both the validation and training accuracies are increasing steadily. The weight matrices show more clear colours and less randomness. The accuracy obtained on the test data after the training process is 35.89%.

Configuration 3: *lambda=.1, n epochs=40, n batch=100, eta=.001*

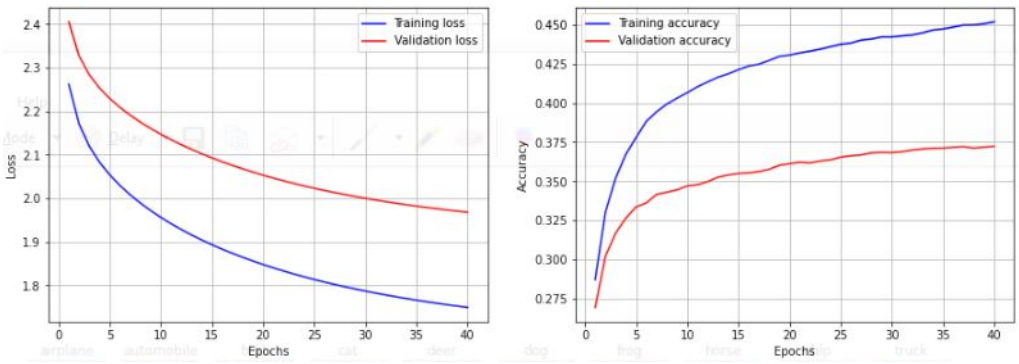

Figure 6: Loss and accuracy learning curves.

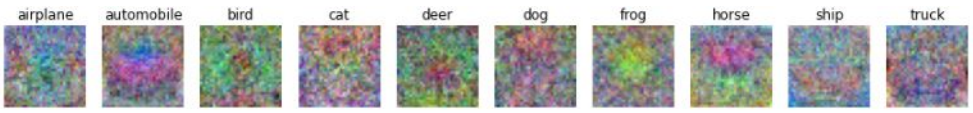

Figure 7: Learnt weight matrix after completion of training.

The learning is even better here than in the previous configuration, as evidenced by lower training and validation losses and higher training and validation accuracies. The weight matrix is also more clear than in the previous case. The accuracy obtained on the test data after the training process is also better, coming in at 37.63%.

Configuration 4: *lambda=1, n epochs=40, n batch=100, eta=.001*

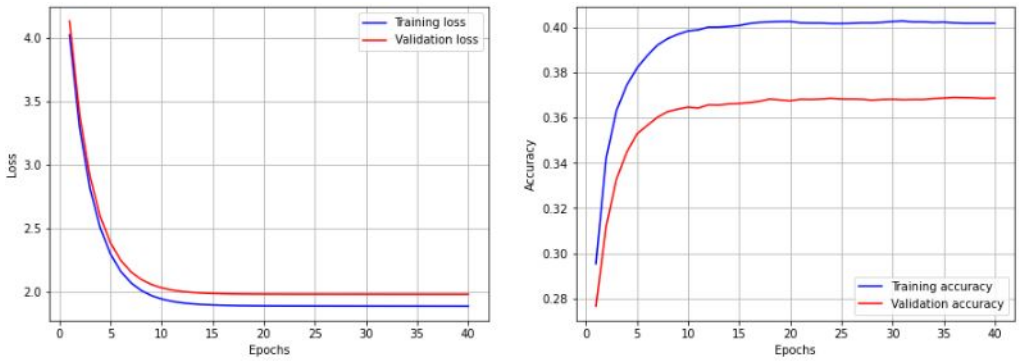

Figure 8: Loss and accuracy learning curves.

Figure 9: Learnt weight matrix after completion of training.

In this last case, we can observe that the loss is slightly higher than in the previous case however the convergence happens faster. This is due to the high regularization used, which makes the network focus only on the main features. The weight matrices seem quite clear now, probably due to the presence of convergence. The accuracy obtained on the test data after the training process is 36.7%.

Conclusion

The results obtained under the different configurations allow us to make conclusions on how both the learning rate and regularisation affects learning. A very high learning rate will cause big oscillations in the loss and accuracy curves. Furthermore, a high learning rate causes the model to not learn very well due to the step taken when updating weights being too large (minima overshoot). Reducing the learning rate gives smoother learning curves and better generalisation performance. Furthermore, the results show that having a small regularisation constant can improve generalisation performance. However, a too big regularisation constant can hurt the network's generalisation capabilities due to lack of complexity forced upon the network.