DD2424 - Assignment 2

Marcus Jirwe

March 2020

Introduction

The purpose of the assignment was to train a multi-linear classifier with one hidden layer to perform classifications on images from the CIFAR-10 dataset. Additionally the impact of the introduction of cyclical learning rates were studied. Practically, this report relies on code programmed in python and primarily using the packages "numpy" and "matplotlib", with a function to compute gradients numerically being adapted from matlab.

Numerical vs Analytical gradient

More specifically, the relative difference between the gradients is calculated by the equation:

$$diff = \frac{||grad_{analytical} - grad_{numerical}||}{||grad_{analytical}|| + ||grad_{numerical}||}$$

The numerical difference between the two were:
For the W1-gradients: 1.6179915671735513e-07
For the b1-gradients: 9.169187706865806e-11
For the W2-gradients: 1.175090486419785e-06
For the b2-gradients: 1.7750236551523402e-07
Which seems small enough to assume the calculations to be correct. The results below seem to lend credit to this conclusion.

1 Practical and implementation

The code relies heavily on the use of the "numpy" library with "matplotlib" being used for plotting the results. Care was taken so as to ensure the dimensions of the matrices were the same as the one in the assignment description, which made it easier and more straightforward to follow the instructions. The testing was performed in the code's main function. The assignment necessitated the implementation of the analytical form of the cross-entropy loss and its gradients as opposed to relying on the numerical gradient. To do this, a derivation shown in the lecture notes were followed and the equations were extended to matrix form. To test that the results of the analytical gradient function was correct, it was tested against the values from the more accurate numerical gradient function using the centered difference. The testing was performed by calculating the Frobenius norm of the difference of the two gradients and dividing it with the sum of the individual norms. This gives a measure of a relative difference between the two. It seems that the implementations would be bug-free as the results to follow seem logical.

2 Results

Below are the cost, loss and accuracy curves for one cycle and three cycles of training. This was performed on a smaller subset of the total available data. Furthermore the three best λ-values for the coarse grid search as well as the three best λ-values in an uniform interval around each of these are shown in a table. The networks trained on these sampled λ-values were trained for two cycles of training each to make sure the network performance reflected a properly trained network. Finally, for the value of λ for which a model performed the best a network was trained for three cycles of training and evaluated. To ensure that a fair measure of the performance was captured a mean and standard deviation of the performance over 30 runs was calculated.

One cycle of training

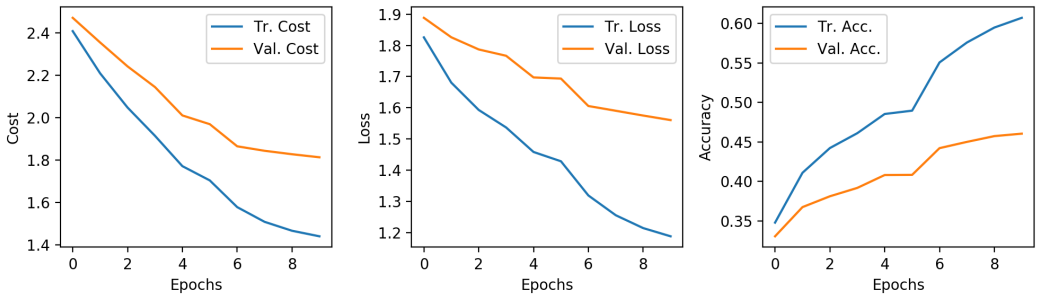

Figure 1: Cost, Loss and Accuracy plots for the hyper-parameter settings: $\eta_{min} = 1e-5, \eta_{max} = 1e-1, \lambda = 0.01, n_s = 500.$ and a batch size of 100 with 10 epochs.

Accuracy

Training Accuracy: 0.6069
Validation Accuracy: 0.4604
Test Accuracy: 0.4641

The effects of the cyclic learning rate is not very obvious in figure 1, except for some bumpiness. It will be more clear as more cycles are used. However, we can see a clear improvement in accuracy now that a hidden layer is employed in comparison to the classifier of the previous assignment.

Three cycles of training

Accuracy

Training Accuracy: 0.7126
Validation Accuracy: 0.4694
Test Accuracy: 0.4775

Here the effect of the cyclic learning rate is much more apparent in figure 2. Clear jumps in the cost and accuracy are apparent when the greatest learning rate is achieved. Furthermore, this clearly has had a positive impact on the performance of the model, although the magnitude of the improvement is somewhat small.

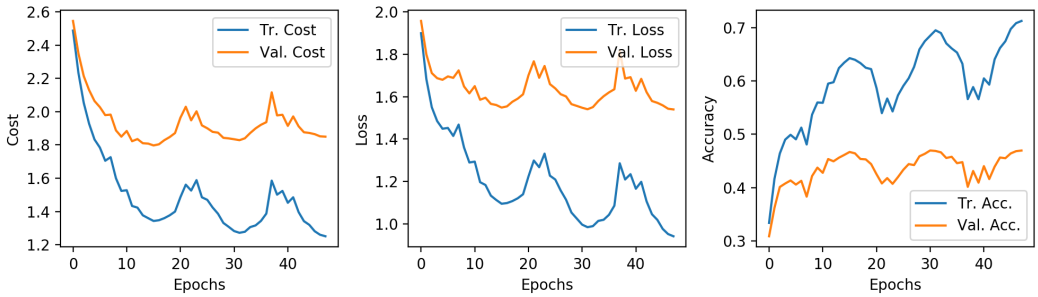

Figure 2: Cost, Loss and Accuracy plots for the hyper-parameter settings: $\eta_{min} = 1e-5, \eta_{max} = 1e-1, \lambda = 0.01, n_s = 800$. and a batch size of 100 with 48 epochs.

Coarse Grid Search

The coarse grid search was performed on a logarithmic scale. 50 values were sampled in the interval $[-5, -1]$ and were translated into λ-values by being used as exponents. In effect, the interval for λ was $[10^{-5}, 10^{-1}]$. The three best λ-values and their corresponding accuracies are presented in the table below.

λ	Tr. Accuracy	Val. Accuracy
0.002792580468572581	0.6196	0.5426
0.003190465556971795	0.6153	0.5406
0.004552705350104431	0.5955	0.5401

Fine Grid Search

Since all the best λ-values were of the same magnitude, a fine grid search was performed around each of them by using the values as the center of an uniform distribution with a width of $0.001/4$ on each side of the coarse value. In each interval around one of the coarse values 50 values were sampled and the three best of those and their accuracies are shown in the below table. The best out of all these λ-values is bolded.

lambda	Tr. Accuracy	Val. Accuracy
0.0028178198828960445	0.6160	0.5444
0.002953728862782196	0.6157	0.5446
0.0029711527547020434	0.6126	0.5448
0.003177893708046988	0.6139	0.5458
0.0032507833272718127	0.613	0.545
0.003330827626680218	**0.6095**	**0.5474**
0.004546239691765983	0.5950	0.5424
0.004589459127313282	0.5971	0.5418
0.004615642753355967	0.5937	0.5462

The best model

Using the best λ-value found in the fine grid search a model is trained on all of the available data for three cycles of the cyclic learning rate. Statistics of the performance of the model are calculated from 30 runs of 3 training cycles each.

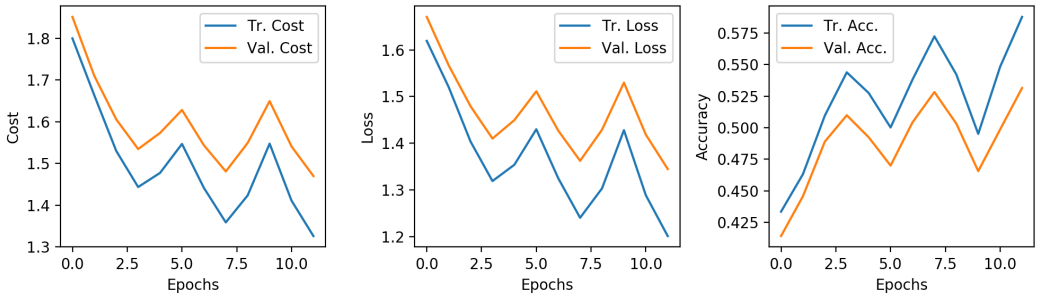

Figure 3: Cost, Loss and Accuracy plots for the hyper-parameter settings: $\eta_{min} = 1e-5, \eta_{max} = 1e-1, \lambda = 0.003330827626680218, n_s = 980.$ and a batch size of 100 with 12 epochs.

Accuracy

Training accuracy: 0.5871 ± 0.002
Validation accuracy: 0.5282 ± 0.004
Test accuracy: 0.5201 ± 0.0032

And we can see that the performance of this model is greater than the initial models by a significant margin.

3 Conclusion

As a conclusion, the addition of a hidden layer has significantly increased the performance, further enhanced by a cyclic learning rate scheme and a more systematic search for an optimal regularisation parameter. The best performance of a model in this assignment achieves an accuracy of around 52 % which is very good, but not excellent. Further improvements could be made like adding additional hidden layers. The introduction of an idea like the cyclic learning rate was a very interesting idea and strikes an intuitive balance between making large sweeping changes with the calculated gradient and making precise adjustments near a minimum.