DD2424 – Assignment 2

Deep Learning

April 21, 2017

Addi Djikic – TSCRM1
addi@kth.se
941028-5473

Abstract

For this assignment a two-layer network were trained with multiple outputs to classify the images from the given CIFAR-10 dataset. This was done mostly analogy as in Assignment 1 from DD2424 and was done successfully. No seed was used durring the assignment this time.

Two Layer Network Classifier

The steps in the process

Firstly the CIFAR-10 data was loaded and later the a weight cell-vector W_{cell} and bias cell-vector b_{cell} where randomly initialized as a Gaussian distribution $G(0, 0.001^2)$ with two metricises each, W_1, W_2 and b_1, b_2 respectively. Later a function that evaluates the network was built based on (1) and (2) from [1] and the softmax-function according to Lecture 4.

Later a cost function was introduced from [2]. Then the gradients were computed as in slide 29 in [2] from Lecture 4.

Now the final step was to implement the mini-batch gradient decent, and that was done analogy as in Assignment 1. Function to test the accuracy was also implemented, which gave a result of around 10% when running it only at the training data, which was reasonable because the dataset is 10 000 images and around 10% if a fair probability if you would only make guesses. The cost function gave around 2.30 with no regularization initialized.

Testing the Computed Gradients

To test that the gradient is correctly computed we can compare it with function that use numerical estimation, therefor we will use `ComputeGradNumSlow` and `ComputeGradNum` from [1] to compare it. The absolute difference between the maximum error values of the slow center difference method, and the faster finite difference method can be seen in the table below. Where all the errors are lower than 10^{-5} as we wanted.

Compared Errors with our Grad. Function	W_1	W_2	b_1	b_2
Central Difference Method	4.5704e-11	2.4997e-11	2.3508e-11	2.0270e-11
Finite Difference Method	6.9312e-11	6.5810e-10	5.1882e-11	4.5007e-07

This was tested with reduced input data, which can be seen in the Matlab code for this report. I believe the gradient computations were bug-free at this point due to similar computations as in Assignment 1 but we instead use cell-arrays for this assignment. The low values of the errors indicate a good gradient estimation.

Adding the Momentum Term

The momentum term can easily be regulated by setting the value of ρ(rho). If ρ is zero then we step as we did in the previous assignment without extra momentum, as it can be seen in the equations. The speed was checked by looking for how ling it took for the cost to land at a specific value with and without the momentum. It was about 10 to 15 times faster with momentum. An overfit check can be seen in fig. 1 where no momentum was used, compared to fig. 2 where momentum was used. (Where there were no regularization, `eta` = 0.01 and 200 epochs).

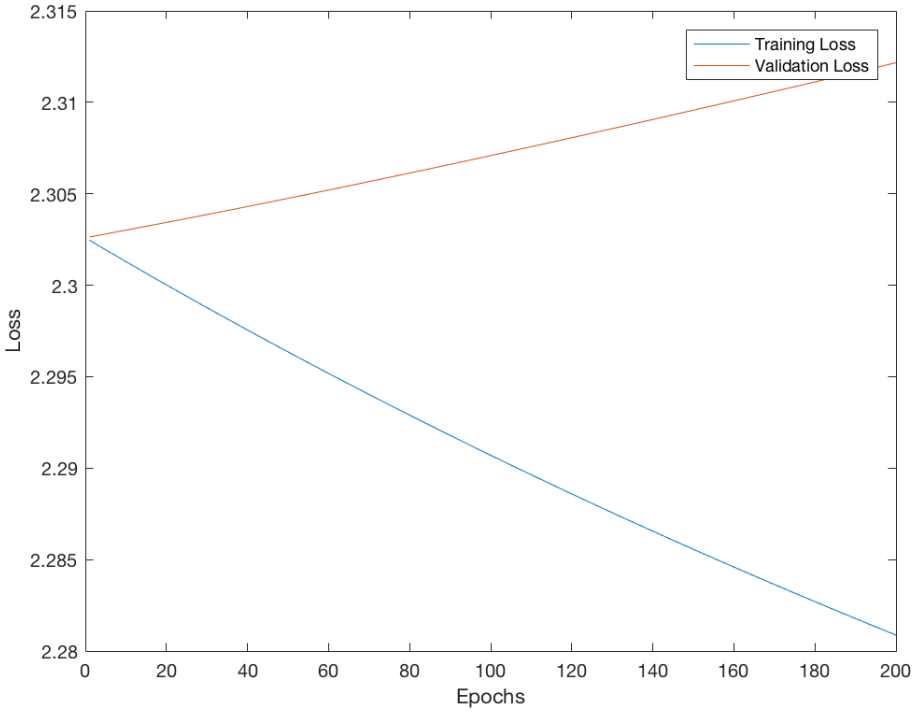

Figure 1: The training and validation cost with no regularization and no momentum

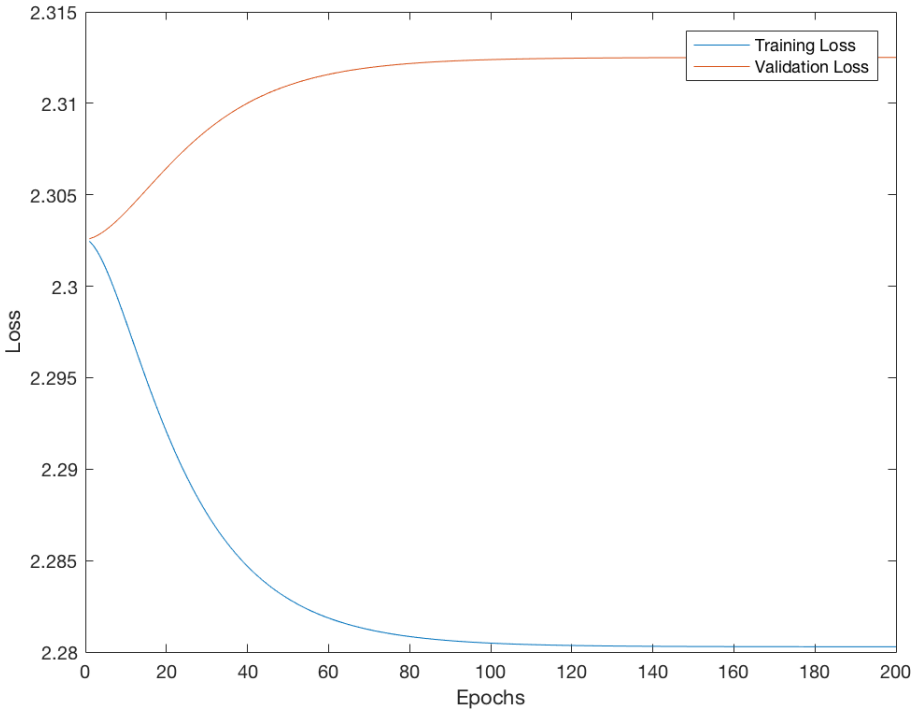

Figure 2: The training and validation cost with no regularization and with momentum

Coarse Search

When starting to search for the optimal values of `lambda` and `eta` the initialized values for the first run was accoring to the table below. Where the training was done for 10 epochs and for 50 runs with each run initializing random values between the range for `lambda` and `eta` along with W and b.

LambdaMax	LambdaMin	EtaMax	EtaMin	Epochs	Nbr of Runs
0.000001	0.1	0.035	0.008	10	50

Which gave values between 39% and at its best 43%, which is not what we wanted.
After just 2 to 3 runs while the values were still reduced quite a lot each time these values in the table below was found:

Lambda	Eta	Accuracy
9.8195e-4	0.01828	0.4351
9.7327e-4	0.02099	0.4413
9.6317e-4	0.01866	0.4379

Where we got a value over 44% which we are looking for, and now a fine search were performed as below to find better stable values of grater than 44% accuracy.

Fine Search

With fine search the best values from the coarse search were used where the most upper and most lower limits were used as a start for lambda, which can be sin in the table below. But the Eta parameter was increased a little to a it were a too small range at first.

LambdaMax	LambdaMin	EtaMax	EtaMin	Epochs	Nbr of Runs
9.8195e-4	9.6317e-4	0.0300	0.01828	10	50

Where the best parameters after a couple of runs were found as seen in the table below:

Lambda	Eta	Accuracy
9.7318e-4	0.02274	0.4461
9.7382e-4	0.02337	0.4422
9.7342e-4	0.02299	0.4427

Result with all training data

When all the data were used and the best parameters from the fine search, i.e. `lambda` = 9.7318e-4 and `eta` = 0.02274 on a validation of 1000 images from each batch, an accuracy of 51.80% was achieved. The plot for the training and validation cost can be seen below in figure fig. 3. Though it was trained for 10 epochs at a time it was expected to overfit a little a few more epochs were needed for it to overfit.

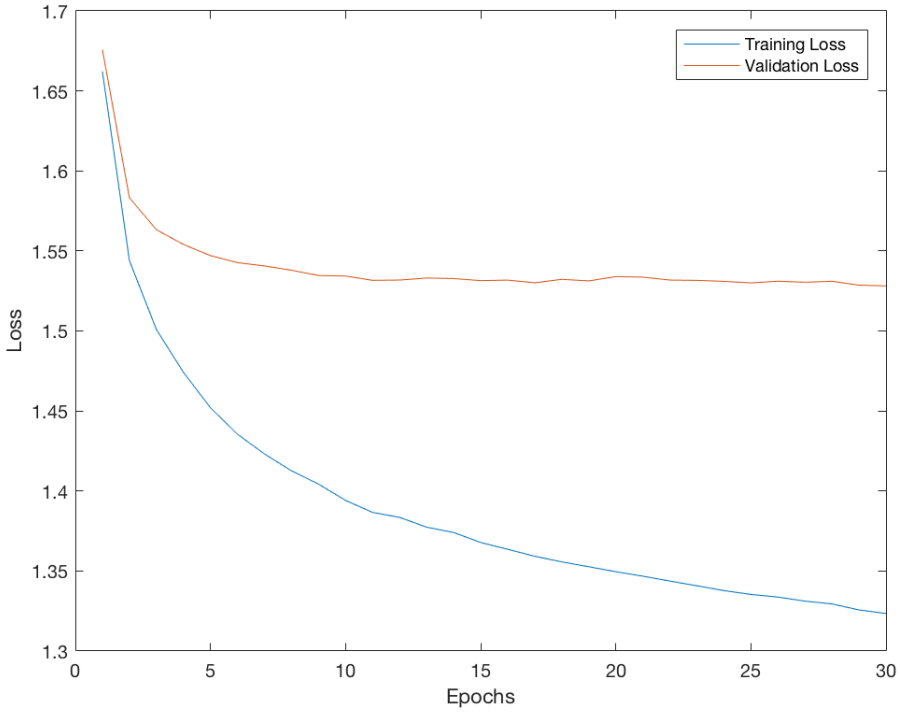

Figure 3: The validation cost and training cost plotted with a run of the best parameters and 30 epochs. Where the accuracy of the classifier was 51.80%

References

[1] Assignment2 DD2424 Instructions paper:
https://www.kth.se/social/files/58eb87d4f2765449ce1ffd1d/Assignment2.pdf

[2] Lecture notes by Josephine Sullivan on DD2424 KTH: https://www.kth.se/social/course/DD2424/subgroup/vt-2017-958/page/lectures-164/