Assignment 2: Two layer network

DD2424 - Deep Learning
Svenja Räther

March 2020

1 Introduction

This assignment 2 in the course DD2424 Deep Learning in Data Science trains and tests a two layer network with multiple outputs to classify images from the CIFAR-10 dataset. The network is trained using mini-batch gradient descent applied to a cost function that computes the cross-entropy loss of the classifier applied to the labelled training data and an L2 regularization term on the weight matrix.

2 Analytical gradient computation

2.1 Comparison of numerical and analytical gradients

The analytical gradients are computed using function *computeGradients*. Those follow the efficient implementation given in the lecture notes. To validate the analytical gradients. Those were compared against the slow but precise computation of the gradients. Due to speed issues, the test was called using a reduced number of images. The relative errors in in table 1 was given for the comparison. Those can be considered sufficiently small. Therefore, I continued with the assignment.

Gradient	**Maximum relative error**
W1	2.4707010908961613e-05
W2	2.614437764086639e-07
b1	2.038579480930604e-10
b2	1.573218141344005e-10

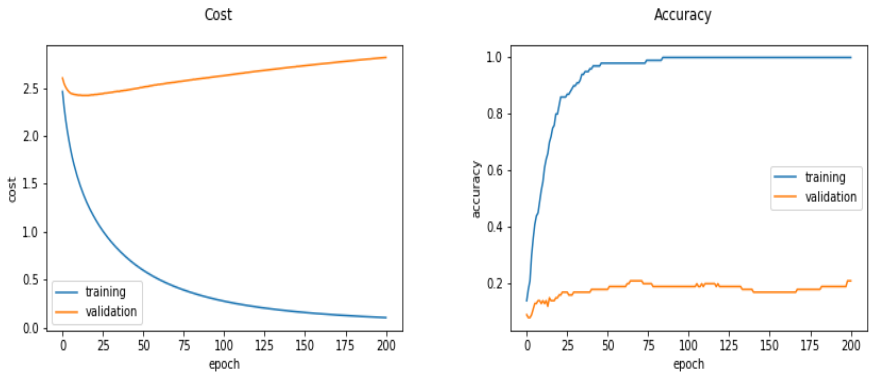
Table 1: Relative error for comparing numerical and analytical gradients for 0:1 entries

2.2 Sanity check

The following sanity check was executed using 100 examples of training data, turned off regularization, 200 epochs, and a eta of 0.01.

Try and train your network on a small amount of the training data (say 100 examples) with regularization turned off (lambda=0) and check if you can overfit to the training data and get a very low loss on the training data after training for a suffcient number of epochs (200) and with a reasonable eta.

Given figure 1a, 1b and table 2 it is clear that the network heavily overfits on the Training while the values for Validation and Test do not show a good result. The test states that: *Being able to achieve this indicates that your gradient computations and mini-batch gradient descent algorithm are okay.*



(a) Cost for the sanity check

(b) Accuracy for the sanity check

Measure	Training	Validation	Testing
Cost	0.1062	2.8174	-
Accuracy	1.0	0.21	0.2003

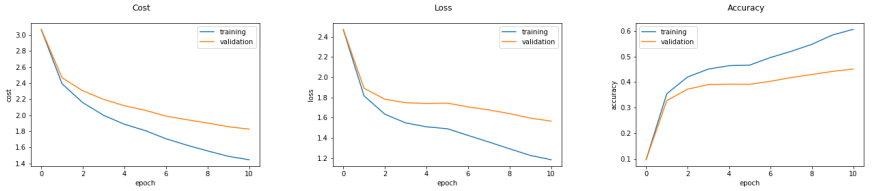
Table 2: Results for the sanity check

3 Cyclical learning rate

The next step implements cyclical learning rates for the update steps.

3.1 Setup 1: One cycle

The first setup uses the following parameter settings:
eta_min = 1e-5, eta_max = 1e-1 and ns=500 and the batch size to 100, lambda=.01. Figure 2 and 4 show the results plotted each epoch. A test accuracy of *46.04%* is achieved. Compared to the previous graphs, a good improvement can be noted by applying the CLR. Other than the overfitted example, now also the Validation and Test accurac is improving.

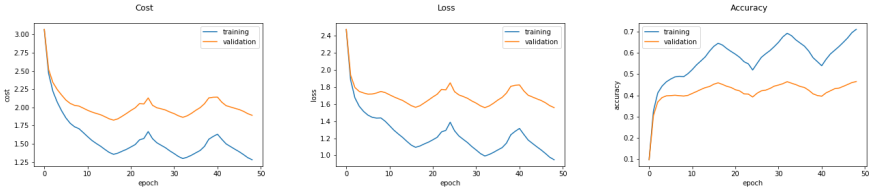


(a) Cost for setup 1 (b) Loss for setup 1 (c) Accuracy for setup 1

Figure 2: Plots for setup 1 printed each epoch

3.2 Setup 2: Three cycles

The first setup uses the following parameter settings:
eta_min = 1e-5, eta_max = 1e-1 and ns=800 and the batch size to 100, lambda=.01. A test accuracy of *46.85%* is achieved. This is slightly better than with only one cycle. Furthermore, the cycles are clearly visible in the plots.



(a) Cost for setup 2 (b) Loss for setup 2 (c) Accuracy for setup 2

Figure 3: Plots for setup 2 printed each epoch

4 Coarse search

The coarse search was done using all data for training except 5000 images for validation with those parameters: eta_min = 1e-5, eta_max = 1e-1, n_batches = 100, cycles = 2, lam_min = -5, lam_max = -1
First, a random search over different values was executed. Those are shown in table 3. The highest accuracy values on Validation are marked in bold font. Good values can be achieved for lambdas between 1e-3 and 1e-2 (0.00122, 0.00130). Also values above 1e-5 (1.9159e-05) show good results in this search. In the next step, lambda was increased by *10 in each round. The values given by this approach underline the findings from the random search. Those are shown in table 4.

Lambda	**Accuracy (Validate)**	**Accuracy (Test)**
0.43764952436411647	0.2532	0.2609
0.00020249682035463785	0.5172	0.5089
0.0012243872776428408	**0.5226**	0.5087
0.014679082188994916	0.5012	0.5034
0.016991071478333907	0.496	0.497
1.9159716518696227e-05	**0.5194**	0.509
1.1953362811452369e-05	0.5186	0.5107
0.004320670609064646	0.52	0.5087
0.0494629300563988	0.4372	0.4465
0.011752492551021617	0.5094	0.5061
2.4771636800671306	0.095	0.1
0.0023627756837847637	0.5152	0.506
0.020339092443524914	0.4884	0.4883
0.0013070594122249744	**0.5194**	0.5151

Table 3: Croase search done randomly

Lambda	**Accuracy (Validate)**	**Accuracy (Test)**
1e-05	**0.5204**	0.5079
1e-04	0.5126	0.5098
1e-03	**0.5202**	0.5107
1e-02	**0.5148**	0.5104
1e-01	0.3832	0.3928

Table 4: Croase search done stepwise

5 Fine search

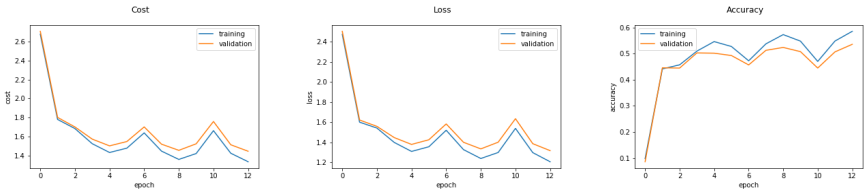
Given the information from the croase search, the fine search was executed with 10 different random lambdas between *1e-03* and *1e-02*. As before, 2 cycles were used. Table 5 shows the results. The best accuracy on Validation was achieved for lambda = *0.0033892965614371645*.

Lambda	Accuracy (Validate)	Accuracy (Test)
0.001	0.5202	0.5107
0.0027507882002820953	0.5216	0.5122
0.0025137229188259003	0.5156	0.5128
0.002381748988351755	0.522	0.5099
0.0033892965614371645	**0.524**	**0.5131**
0.008575230169932417	0.5182	0.5109
0.002408497884331884	0.5222	0.5124
0.0017561342138934421	0.5224	0.5148
0.0038386574892497805	0.516	0.5098
0.0038749553935311675	0.5178	0.5138

Table 5: Fine search between 1e-03 and 1e-02 done randomly

6 Best performing network

Given the previous searches, the best value for lambda = 0.00906. In this step, the network is trained for 3 cycles using the stated lambda. The result shows an accuracy of *51.92%* on Test and *53.6%* on Validation.



(a) Cost for the improved setup (b) Loss for the improved setup (c) Accuracy for the improved setup

Figure 4: Plots for the improved setup