DD2424 Deep Learning in Data Science - Assignment 1

Bas Straathof — btstr@kth.se

April 10, 2019

1 Introduction

The functions that were required for this assignment were all successfully implemented in Python 3, including the one to check the gradient numerically (`ComputeGradientNum`). A separate `TextMethods` class was established for testing several aspects of the data set and the methods of the `Classifier` class (with a random seed of `numpy.random.seed(0)`).

2 Gradient checking

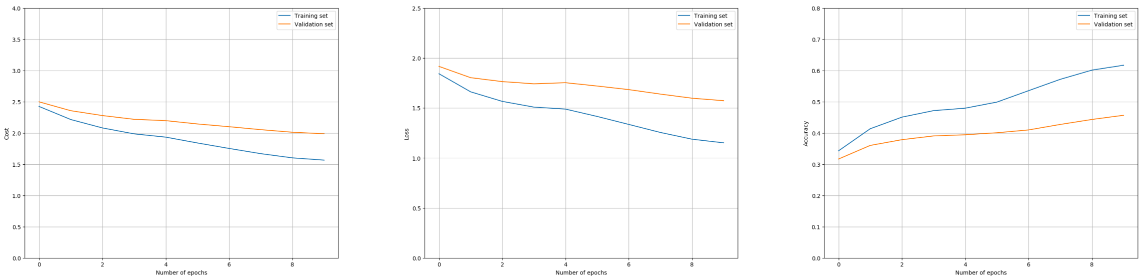
A custom test case was created to test the similarity of arrays resulting from computing the gradients analytically (`ComputeGradients`) analytically and numerically (`ComputeGradientsNum`) by using the finite difference method. Using the `assert_almost_equal` function from the `numpy.testing` library, it was found that the arrays were equal up to 5 decimals. If tested up to 6 decimals, there was a mismatch of ca. 1.60%.

3 Cost, loss and accuracy curves when using cyclical learning rates

See Figures 1 and 2 for the cost, loss and accuracy plots that correspond to Figure 3 and Figure 4 in the assignment sheet. As can be observed, these figures have been replicated up to a reasonable degree. Figures 1a and 1b, demonstrate that during a single cycle of the cyclic learning rate, both the cost and loss functions do not necessarily monotously decrease. This is a desirable feature, since using cyclic learning rates allow us to both close in on local optima and explore a wider search space than with simple learning rate decay. This behaviour is even more evident in Figures 2a and 2b. There is a slight discrepancy between the test accuracies reported in the assignment sheet and the ones reported here, which is due to randomness in the initialization of the network's weight and bias parameters.

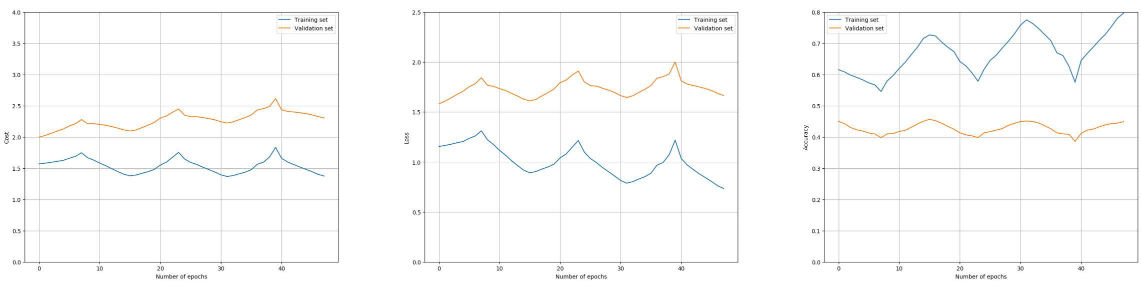
3.1 Coarse search

For a coarse search for `lambda`, 20 values were randomly sampled from the range $[10 \cdot 10^{-5}, 10 \cdot 10^{-1}]$. Using these values, in conjunction with the parameters `batch_s` $= 100$, `n_s` $= 900$ and `n_epochs` $=$ 8 the results displayed in Table 1 were obtained. Based on the accuracies obtained on the validation



(a) Costplot (b) Loss plot (c) Accuracy plot

Figure 1: Training curves (cost, loss, accuracy) for one cycle of training, where one batch of the training data is used. The hyper-parameter settings of the training algorithm are `eta_min = 1e-5, eta_max = 1e-1, lambda=.01, n_s=500, batch_s=100`. At the end of training a training accuracy of 61.67% and a test accuracy of 45.73% are achieved.



(a) Costplot (b) Loss plot (c) Accuracy plot

Figure 2: Training curves (cost, loss, accuracy) for three cycles of training, where one batch of the training data is used. The hyper-parameter settings of the training algorithm are `eta_min = 1e-5, eta_max = 1e-1, lambda=.01, n_s=800, batch_s=100`. At the end of training a training accuracy of 79.65% and a test accuracy of 45.59% are achieved.

set, the three best regularization values obtained were `lambda` $= \approx 1.82 \cdot 10^{-5}$, `lambda` $= \approx 0.0060$ and `lambda` $= \approx 0.0133$.

3.2 Fine search

For a fine search for `lambda`, the best values obtained during the coarse search were used as an indication for the fine search range. 15 values were uniformly sampled from the range $[0.006, 0.013]$ and another 15 values were uniformly sampled from $[1.7 \cdot 10^{-5}, 3.5 \cdot 10^{e-5}]$. Using these values, in conjunction with the parameters `batch_s` $= 100$, `n_s` $= 900$ and `n_epochs` $= 16$, the results displayed in Table 2 were obtained. Based on the accuracies obtained on the validation set, the three best regularization values obtained were `lambda` ≈ 0.00895, `lambda` ≈ 0.00949 and `lambda` ≈ 0.00828.

Lambda	Training accuracy	Validation accuracy
1.68608400995603E-05	**0.5940444444**	**0.5192**
1.82477332600308E-05	**0.6329111111**	**0.5232**
2.47164471534192E-05	**0.6551333333**	**0.5178**
3.50328394442253E-05	**0.6717111111**	**0.509**
3.63923504686741E-05	0.6854	0.5012
7.27225110298391E-05	0.6967111111	0.499
8.44837970248924E-05	0.7069333333	0.4924
0.000199594	0.7148	0.4876
0.0002440395	0.7218222222	0.4846
0.0002669687	0.7284666667	0.4802
0.0002796529	0.7337333333	0.48
0.0004849874	0.7356444444	0.4768
0.0006340283	0.7382888889	0.4758
0.0006419786	0.7406444444	0.4776
0.000660229	0.7399555556	0.4808
0.0009248035	0.7388	0.4828
0.0009864478	0.7352888889	0.4816
0.002210287	0.7161555556	0.4928
0.0033608031	0.6832888889	0.508
0.0060451793	**0.6334444444**	**0.5232**
0.0113158071	**0.5800888889**	**0.526**
0.0133930648	**0.5640666667**	**0.5228**
0.0293842626	0.5159777778	0.497
0.0321224588	0.5091777778	0.4966
0.0333943315	0.5078888889	0.4948

Table 1: Coarse search for the regularization parameter `lambda`.

Lambda	**Training accuracy**	**Validation accuracy**	**Test accuracy**
1.95245454779958E-05	0.6282222222	0.511	0.5069
2.10714883387606E-05	0.6706222222	0.5098	0.497
2.11704804101738E-05	0.6958666667	0.5016	0.4877
2.18287599271178E-05	0.7167333333	0.4908	0.4834
2.32435708682032E-05	0.7321555556	0.4858	0.4731
2.49333848215279E-05	0.7434	0.478	0.4685
2.51094888371022E-05	0.7544888889	0.4736	0.4601
2.51338425735977E-05	0.7627333333	0.4706	0.4599
2.51886257907754E-05	0.7714222222	0.464	0.4539
2.58472226847946E-05	0.7797555556	0.4608	0.4498
2.67246611679364E-05	0.7873555556	0.4582	0.4485
2.95163413455546E-05	0.7936888889	0.4582	0.446
3.26065165858308E-05	0.7966222222	0.4558	0.4416
3.28565273915834E-05	0.8026444444	0.456	0.4426
3.47938779507186E-05	0.807	0.4528	0.4393
0.0063970386	0.6255111111	0.5202	0.5081
0.0064571156	0.6030888889	0.5298	0.5196
0.0066877256	0.6004444444	0.5322	0.5212
0.0069528318	0.6015777778	0.5364	0.5245
0.0075079205	0.6001333333	0.5406	0.5266
0.0082752579	0.5954	0.5414	0.5239
0.0084963055	0.5947111111	0.5388	0.526
0.0085315839	0.5964444444	0.541	0.5282
0.0089500283	**0.5932**	**0.544**	**0.5265**
0.0094896297	0.5896888889	0.542	0.5264
0.0101027852	0.5873555556	0.5394	0.5262
0.0104212754	0.5855555556	0.5374	0.5215
0.0113439497	0.5816888889	0.5354	0.5223
0.0114720399	0.5795333333	0.535	0.5228
0.0121369154	0.5761111111	0.5322	0.5219

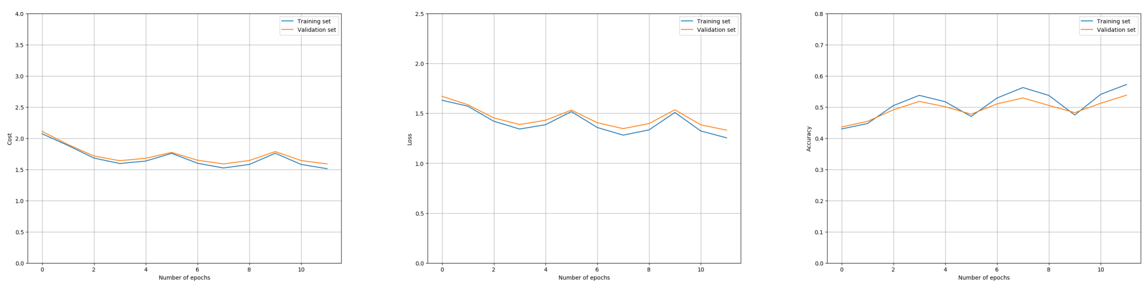
Table 2: Fine search for the regularization parameter `lambda`.

4 Best classifier

The best classifier was trained 10 times on a training set of 49,000 images using the following parameter settings: `lambda`=0.00895, `batch_s`=100, `n_s`=980 and `n_epochs`=12 (i.e. 3 cycles). The following results were obtained:

- The accuracy on the training set is: 0.5701 $\pm$ 0.0015
- The accuracy on the validation set is: 0.5228 ($\pm$ 0.0066
- The accuracy on the testing set is: 0.5207 $\pm$ 0.0026

Using `numpy.random.seed(0)`, for the best classifier the cost, loss and accuracy plots showed in Figure 3 were obtained. As can be observed, the classifier is not overfitting.



(a) Costplot (b) Loss plot (c) Accuracy plot

Figure 3: Training curves (cost, loss, accuracy) for three cycles of training, where 49,000 images are used as training data. The hyper-parameter settings of the training algorithm are `eta_min = 1e-5, eta_max = 1e-1, lambda=.00895, n_s=980, batch_s=100`. At the end of training a training accuracy of 57.22% and a test accuracy of 52.65% were achieved.