KTH Royal Institute of Technology

DD2424 Deep Learning in Data Science

Assignment 2

Name: Polyxeni Ioannidou
E-mail: ppio@kth.se

October 15, 2019

---

1 Introduction

In this assignment it has been built a two-layer network to classify images from the CIFAR-10 dataset. The model has been trained with the Mini-Batch Gradient Descent with cyclical learning rates. We used the CIFAR-10 dataset, which consists of 60000 32x32 colour images in 10 classes. The dataset is divided into five training batches and one test batch, each with 10000 images. We trained our model with two different combinations of datasets. The first simple way is that we used data from data_batch_1 as our training set and data from data_batch_2 as validation set. The second way was that we created a larger dataset where the training set consisted of 5 folders data_batch_1 - data_batch_5 , while we also split the training set to create the validation set. Moreover, a cost function that computes the cross-entropy loss of the classifier is implemented and regularization is also being applied. For computing and testing my gradients it has been used the function assert_almost_equal from numpy library, where it was found that the arrays were equal up to 5 decimal. Last thing that should be mentioned is that all functions are implemented in Python 3.7, as well as there has been no use of libraries with already build-in networks.

2 Exercises 1,2: Build Model and Checking Gradients

The model is a two-layers neural network, where the first layer (the hidden layer) has 50 nodes and a bias term for each node. The second layer has 10 nodes and again a bias term for each node. The algorithm of Mini-Batch Gradient Descent with cyclical learning rates has been implemented to train the model. In mini-batch with GD, we computed and checked gradients analytically and numerically. For computing and testing my gradients it has been used the function assert_almost_equal from numpy library, where it was found that the arrays were equal up to 5 decimal. More specifically, when I tested for 6 decimals I got the following result:

AssertionError: Arrays are not almost equal to 6 decimals
Mismatch: 13%
Max absolute difference: 5.69965034e-06
Max relative difference: 0.00015417

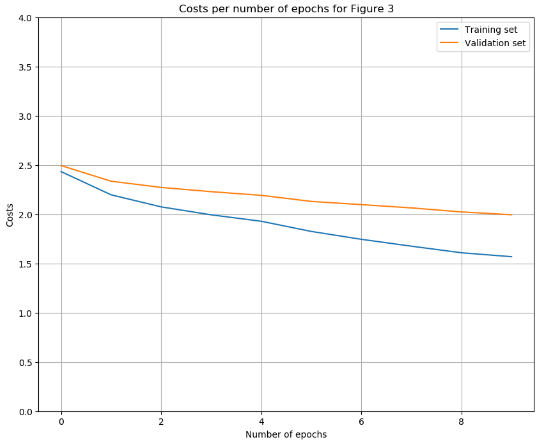
3 Exercise 3a: Training the network with CLR

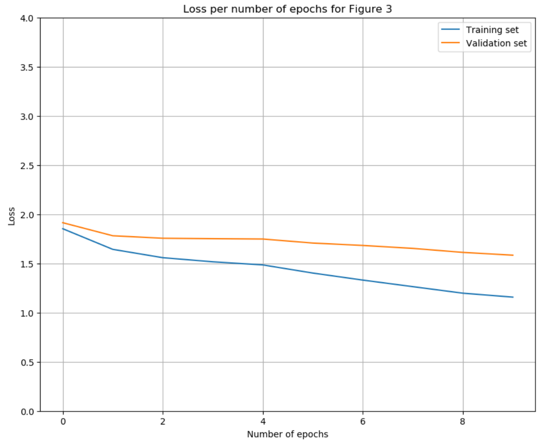
In this part, there have been performed two different tests. In the 1st one we tried to replicate Figure 3 as it has been described in the assignment's instructions. For this replication, we ran the training algorithm for 1 cycle (10 epochs) and one batch of the training data is used: we used data from data_batch_1 as our training set and data from data_batch_2 as validation set. The parameters settings are the followings: lamda=0.01, batch_size=100, learning_rate_min=1e-5, learning_rate_max=1e-1, stepsize=500 and n_epochs=10.

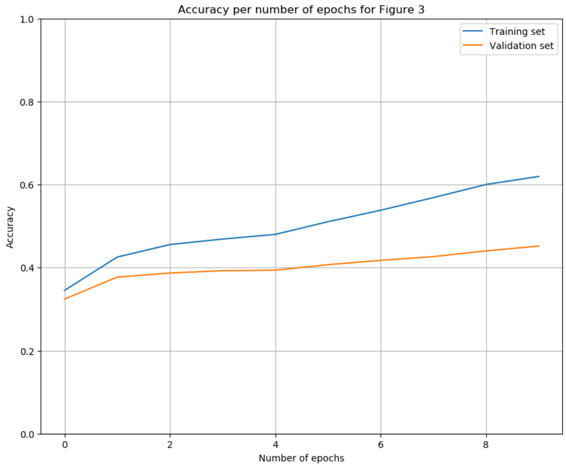
Below are the results for accuracy in training,validation and testing sets (Figure 3):

The accuracy on the training set is: 0.6195
The accuracy on the validation set is: 0.4519
The accuracy on the testing set is: 0.4589

Plots for cost, lost and accuracy in both validation and training sets are the followings:



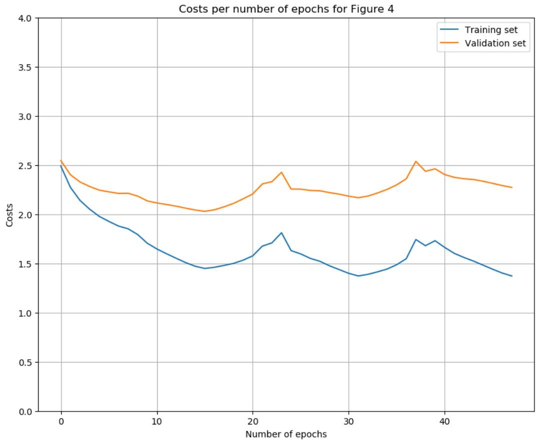


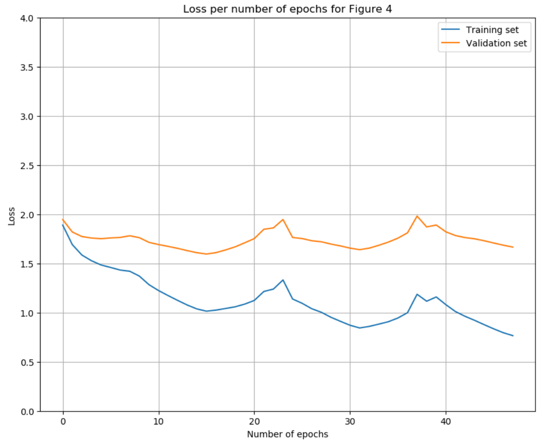


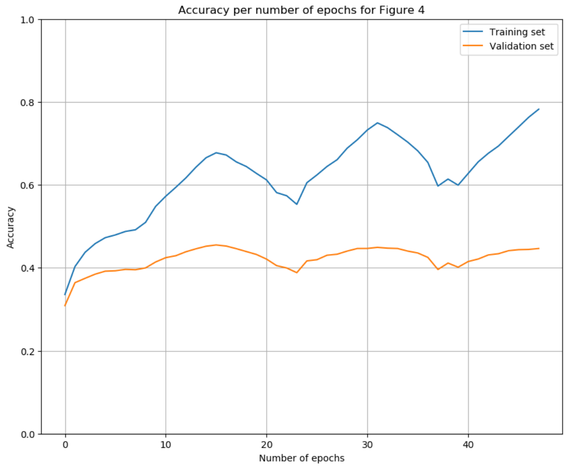
In the 2nd test we tried to replicate Figure 4, as it has been described in the assignment's instructions. For this replication, we ran the training algorithm for around 3 cycles (38 epochs) and one batch of the training data is used: we used data from data_batch_1 as our training set and data from data_batch_2 as validation set. The parameters settings are the followings: lamda=0.01, batch_size=100, learning_rate_min=1e-5, learning_rate_max=1e-1, stepsize=700 and n_epochs=38.

Below are the results for accuracy in training,validation and testing sets (Figure 4):

The accuracy on the training set is: 0.7825
The accuracy on the validation set is: 0.4465
The accuracy on the testing set is: 0.4665







Conclusions for using CLR: The simple vanilla Gradient Descent converges really slow and training is taking a lot of time. Using CLR we actually update the learning rate periodically, which means that we vary it between an upper and a lower bound. In other words, having an adaptive learning rate means that we change this term to match the the cost surface at a local point at the current estimate of the model's parameters.

It is obvious from the results that by increasing the number of cycles (epochs) the training and testing accuracy differ pretty much comparing with the difference that they had when training the model with one cycle:

- One cycle: training accuracy = 61%, testing accuracy = 45.8%, diff = 15.2%
- Three cycles: training accuracy = 78%, testing accuracy = 46.6%, diff = 31.4%

In the 2nd case we can see the change in the learning process as the model cycles three times over the learning rates.

4 Exercise 3b: Coarse-to-fine random search to set λ

For this part, we sampled 30 λ values from a uniform in between $(0.001, 1e-9)$ and then we ran the training algorithm for 2 cycles (20 epochs). Moreover, one batch of the training data is used. The rest of the parameters similar to the ones in exercise 3a for the replication of Figure 4. Below are the results:

Lambda = 0.0006776934919477309
Train accuracy: 0.6815 , Validation accuracy: 0.4093, Test accuracy: 0.4094

Lambda = 0.0008451467803311274
Train accuracy: 0.7832, Validation accuracy: 0.3999, Test accuracy: 0.3966

Lambda = 0.000970924758372925
Train accuracy: 0.8493, Validation accuracy: 0.3952, Test accuracy: 0.3942

Lambda = 0.0006282981057587514
Train accuracy: 0.8958, Validation accuracy: 0.3917, Test accuracy: 0.3915

Lambda = 0.00021200890185896974
Train accuracy: 0.9302, Validation accuracy: 0.3925, Test accuracy: 0.3911

Lambda = 0.0009387692769619314
Train accuracy: 0.9523, Validation accuracy: 0.3901, Test accuracy: 0.3909

Lambda = 0.0005832786948924888
Train accuracy: 0.9672, Validation accuracy: 0.3901, Test accuracy: 0.3908

Lambda = 0.0007492181257690207
Train accuracy: 0.9776, Validation accuracy: 0.3898, Test accuracy: 0.3909

Lambda = 0.0009734884031846111
Train accuracy: 0.9818, Validation accuracy: 0.3909, Test accuracy: 0.3912

Lambda = 0.00014028740525326043
Train accuracy: 0.9872, Validation accuracy: 0.3905, Test accuracy: 0.3929

Lambda = 7.611832815945861e-05
Train accuracy: 0.9914, Validation accuracy: 0.3909, Test accuracy: 0.3905

Lambda = 0.0007157367281307463
Train accuracy: 0.9929, Validation accuracy: 0.391, Test accuracy: 0.3915

Lambda = 0.00026080378727479255
Train accuracy: 0.9969, Validation accuracy: 0.3924, Test accuracy: 0.3869

Lambda = 5.418438547771505e-05
Train accuracy: 0.9988, Validation accuracy: 0.3911, Test accuracy: 0.3873

Lambda = 7.934996431579631e-05
Train accuracy: 0.9998, Validation accuracy: 0.3896, Test accuracy: 0.3872

Lambda = 0.0008064863617567868
Train accuracy: 0.9997, Validation accuracy: 0.3922, Test accuracy: 0.3876

Lambda = 0.0004016583347810842
Train accuracy: 0.9997, Validation accuracy: 0.3914, Test accuracy: 0.3882

Lambda = 0.0008426843571118819
Train accuracy: 0.9993, Validation accuracy: 0.3925, Test accuracy: 0.3894

Lambda = 0.0005192559140511307
Train accuracy: 0.9964, Validation accuracy: 0.3975, Test accuracy: 0.3939

Lambda = 0.0008272795875800633
Train accuracy: 0.9949, Validation accuracy: 0.3958, Test accuracy: 0.3958

Lambda = 0.00019444742366967762
Train accuracy: 0.9957, Validation accuracy: 0.3934, Test accuracy: 0.3904

Lambda = 0.0008116822137911484
Train accuracy: 0.9979, Validation accuracy: 0.3944, Test accuracy: 0.3916

Lambda = 6.0230062336062945e-05
Train accuracy: 0.9995, Validation accuracy: 0.3927, Test accuracy: 0.3905

Lambda = 0.00015992399773589582
Train accuracy: 0.9999, Validation accuracy: 0.3918, Test accuracy: 0.3894

Lambda = 0.00041635425670729525
Train accuracy: 0.9999, Validation accuracy: 0.392, Test accuracy: 0.3898

Lambda = 0.0002509711114535171
Train accuracy: 0.9999, Validation accuracy: 0.3933, Test accuracy: 0.3902

Lambda = 0.00012244316388740838
Train accuracy: 0.9999, Validation accuracy: 0.393, Test accuracy: 0.39

Lambda = 0.0008452512277932983
Train accuracy: 0.9999, Validation accuracy: 0.3942, Test accuracy: 0.3916

Lambda = 0.0003643822378200393
Train accuracy: 0.9999, Validation accuracy: 0.3947, Test accuracy: 0.3922

Lambda = 0.0002592807309570905
Train accuracy: 0.9999, Validation accuracy: 0.3942, Test accuracy: 0.3916

The best results based on validation accuracy were for lambda 0.00041635425670729525 and 0.0007492181257690207. So based on these two values, now we repeat the search to a more narrower range of $(0.0001, 1e-6)$. We also sample 30 λ values again.

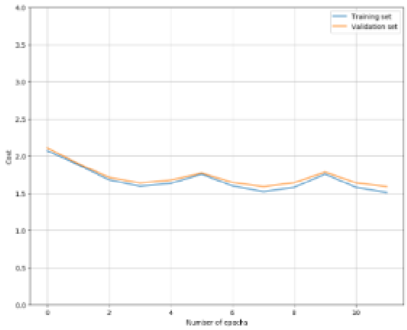
Lambda	Train accuracy	Validation accuracy	Test accuracy
6.861844223554064e-05	0.6903	0.4137	0.412
6.929466357926098e-05	0.8081	0.4078	0.4071
6.902760197234175e-05	0.8793	0.4027	0.4003
8.516726135527142e-05	0.9169	0.4031	0.3934
4.604988657028206e-05	0.9464	0.3999	0.3897
4.495908722634612e-05	0.9705	0.3975	0.3883
5.6315858711057074e-05	0.9792	0.3966	0.3887
8.104811855284822e-05	0.9878	0.3961	0.3882
9.734417892451165e-05	0.9917	0.3972	0.3871
6.388117894105882e-05	0.9938	0.3988	0.3887
5.703462131528679e-05	0.9969	0.3995	0.3898
5.084829561075981e-06	0.9991	0.4001	0.3907
5.5783639573794314e-05	0.9996	0.4004	0.3897
5.438393228502804e-06	1.0	0.3991	0.3893
4.714088248979307e-05	1.0	0.3998	0.3894
2.4182082167135827e-05	1.0	0.4001	0.3897
1.2234588672757396e-05	1.0	0.4001	0.3901
3.1194131236510333e-06	1.0	0.4	0.3903
5.162726959789089e-05	1.0	0.4001	0.3905
6.499395632477166e-05	1.0	0.3993	0.3906
5.82776309025767e-05	1.0	0.3985	0.3905
2.8594516396776093e-05	1.0	0.3987	0.3905
3.755076727030109e-05	1.0	0.3986	0.3905
9.748931724749742e-05	1.0	0.3987	0.3909
9.403875805288233e-05	1.0	0.3978	0.3911
3.8584309096083535e-05	1.0	0.3981	0.3916
4.5529454562390784e-05	1.0	0.3984	0.3915
4.8023050035156435e-05	1.0	0.3983	0.3919
9.678576739290062e-05	1.0	0.3982	0.3918
9.801629619476043e-05	1.0	0.399	0.3921

Fine search for the regularization parameter λ

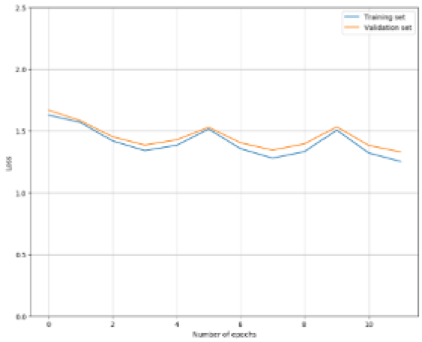
From the results above, we choose the first row as our λ, because it gives us the highest value for accuracy in the validation set. Moreover, it gives the best results for test accuracy. There are two interesting things here. Firstly, the training accuracy for this value of λ was the lowest. Secondly, in a more specific (narrow) area in this range we can observe almost the same performance for our model, as accuracies do not differ a lot, with training accuracy equal to 1.0 !!!

5 Exercise 4: Best Classifier

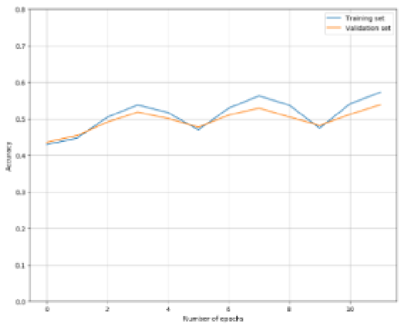
Using the best $\lambda = 0.4137$ we trained the model 10 times and then we achieved a test accuracy at 56%. The plots for cost, loss and accuracy on training and validation set are the followings:



Costs



Loss



Accuracy