Assignment 2

DD2424 Deep Learning in Data Science

Anna Canal Garcia
annacg@kth.se

April 19, 2018

1 Introduction

This assignment aims at training and testing a multi-linear two-layer network using mini-batch gradient descent method applied to a cost function that computes the cross-entropy loss to classify images from the CIFAR-10 dataset. L2 regularization term on the weight matrix is also computed in the cost function.

In order to speed up the training we will add a momentum term in the update step.

Moreover, we will also be paying more attention to how to search for good parameter settings for the network's regularization term and the learning rate.

2 Method

The mathematical details of the 2-layer network are as follows:

$$s_1 = \boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1 \tag{1}$$

$$\boldsymbol{h} = max(0, \boldsymbol{s}_1) \tag{2}$$

$$s_2 = \boldsymbol{W}_2 \boldsymbol{h} + \boldsymbol{b}_2 \tag{3}$$

$$\boldsymbol{p} = SOFTMAX(\mathbf{s}) = \frac{exp(\mathbf{s})}{\mathbf{1}^T exp(\mathbf{s})} \tag{4}$$

And in order to speed up training momentum is added in the update step:

$$\boldsymbol{v}_t = \rho \boldsymbol{v}_{t-1} + \eta g \tag{5}$$

$$\theta_t = \theta_{t-1} - \boldsymbol{v}_t \tag{6}$$

3 Results

1. **Analytic gradient computations check.**

   In order to test that the gradients computations are correct, the difference between them and the numerical is calculated:

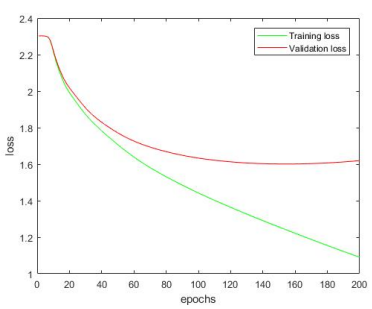
Error grad_W_1	**Error grad_b_1**	**Error grad_W_2**	**Error grad_b_2**
3.04e-04	1e-03	5.26e-08	6.43e-10

Table 1: Relative error of gradients.

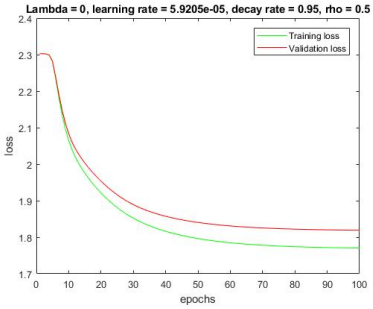
   Since the relative errors on all the parameters are very small number, I assume that my gradient computations are bug free.

2. **Effect on the training when adding the momentum term**

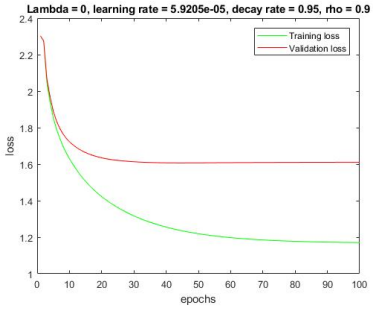
   Here a comparison between a training without and with the momentum term is done. The parameters are the following: n_epochs = 200, n_batch = 100, decay_factor = 0.95 and ρ = [0.5,0.9,0.99]



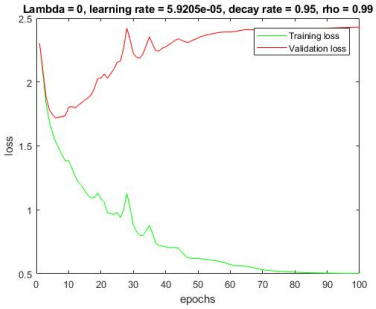
(a) Without momentum



(b) With momentum $\rho = 0.5$



(c) With momentum $\rho = 0.9$



(d) With momentum $\rho = 0.99$

Figure 1: Loss function with $\lambda = 0$ and decay_factor=0.95

Fig.1 shows that momentum term helps to learn faster so that we need less epochs to have convergence. One can see that momentum with ρ=0.9 gives better performance.

3. **Coarse random search**

In this search I have used n_epochs = 10, n_batch = 100, decay_factor = 0.95 and $\rho = 0.9$. 50 different pairings for lambda and eta are generated and tested on validation set. The coarse search range for eta is from 0.001 to 0.3 and for lambda is from 1e-7 to 0.1. The following are the hyper-parameter settings for the 3 best performing networks trained:

- Hyper-parameter setting 1: $\eta = 0.01916$ and $\lambda = 0.00033$

  With this setting the validation accuracy is 44.08%. On Fig.2 its loss function is shown.

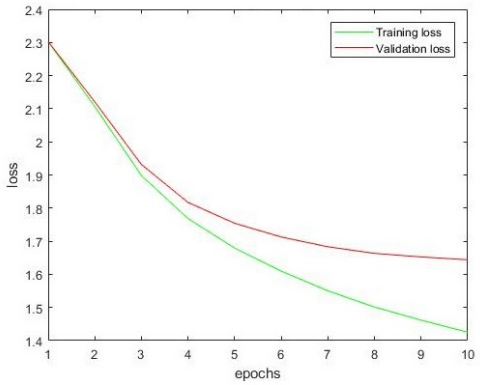


Figure 2: Loss function with $\lambda = 0.00033$, $\eta = 0.01916$, epochs = 10 and n_batch = 100.

- Hyper-parameter setting 2: $\eta = 0.02350$ and $\lambda = 0.00192$

  With this setting the validation accuracy is 43.8%. On Fig.3 its loss function is shown.

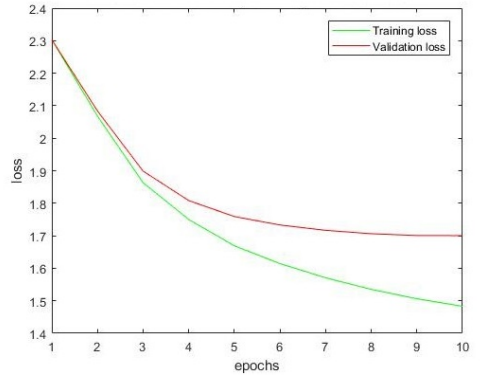


Figure 3: Loss function with $\lambda = 0.00192$, $\eta = 0.02350$, epochs = 10 and n_batch = 100.

- Hyper-parameter setting 3: $\eta = 0.03406$ and $\lambda = 0.00572$

  With this setting the validation accuracy is 43.7%. On Fig.4 its loss function is shown.

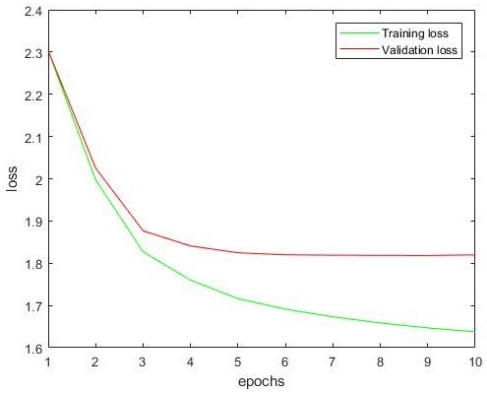


Figure 4: Loss function with $\lambda = 0.00572$, $\eta = 0.03406$, epochs = 10 and n_batch = 100.

4. **Fine random search**

   In this search I have used n_epochs = 10, n_batch = 100, decay_factor = 0.95 and $\rho = 0.9$. 50 different pairings for lambda and eta are generated and tested on validation set. The coarse search range for eta is from 0.01 to 0.06 and for lambda is from 1e-6 to 0.01. The following are the hyper-parameter settings for the 3 best performing networks trained:

   - Hyper-parameter setting 1: $\eta = 0.02589$ and $\lambda = 0.00261$

     With this setting the validation accuracy is 44.19%. On Fig.5 its loss function is shown.

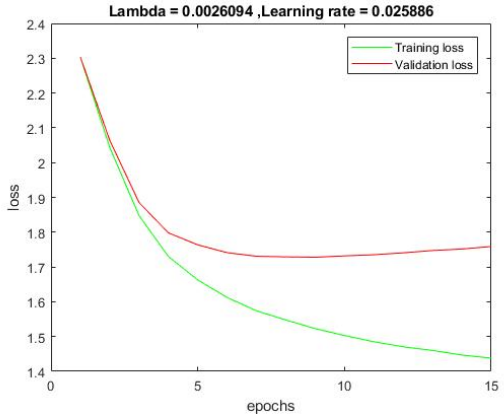


Figure 5: Loss function with epochs = 15 and n_batch = 100.

- Hyper-parameter setting 2: $\eta = 0.04305$ and $\lambda = 0.00420$

  With this setting the validation accuracy is 44.15%. On Fig.6 its loss function is shown.

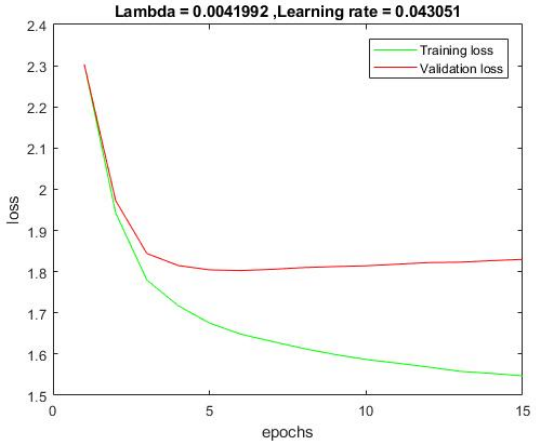


Figure 6: Loss function with epochs = 15 and n_batch = 100.

- Hyper-parameter setting 3: $\eta = 0.04488$ and $\lambda = 0.00472$

  With this setting the validation accuracy is 44.15%. On Fig.7 its loss function is shown.

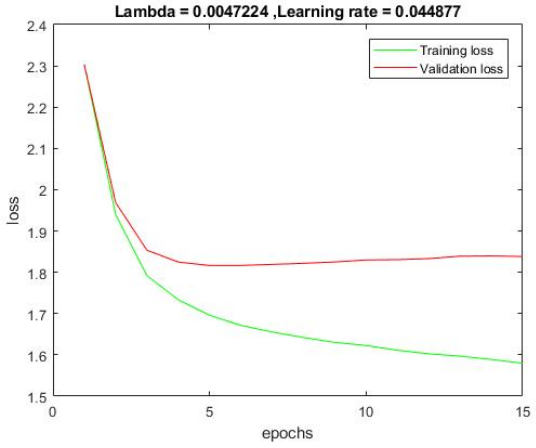


Figure 7: Loss function with epochs = 15 and n_batch = 100.

After that first fine random search, a second fine search is performed restricting η between 0.02 and 0.04, and λ between 1e-4 and 0.01. The best result is achieved with the setting: $\lambda = 0.00561$ and $\eta = 0.03546$ which gives a validation accuracy of 44.62%.

5. **Best found hyper-parameter setting**

Finally, the results when using the best hyper-parameters: $\lambda = 0.00561$ and $\eta = 0.03546$, n_batch = 100, m = 50, n_epochs = 30, decay_rate = 0.95 and rho = 0.9. With this configuration the training Accuracy is 52.94% and the test Accuracy is 47.62%. Fig.8 shows the loss function of this network.

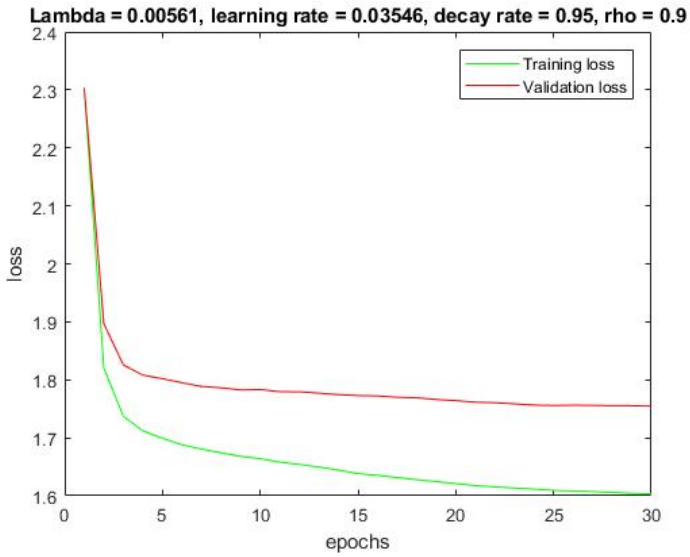


Figure 8: Loss function with epochs = 30 and n_batch = 100.

The data used for training is just 2 batch data files (data_batch_1.mat and data_batch_3.mat) and then 1000 points from data_batch_2.mat for validation. I cannot use more data on training in Matlab.