Assignment 1

DD2424 Deep Learning in Data Science

Anna Canal Garcia
annacg@kth.se

April 10, 2018

1 Introduction

This assignment aims at training and testing a multi-linear one-layer network using mini-batch gradient descent method to classify images from the CIFAR-10 dataset.

File data_batch_1.mat is used for training, the file data_batch_2.mat for validation and the file test_batch.mat for testing.

2 Method

The classification function consists of a Linear scoring function (equation 1) and SoftMax operation (equation 2):

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

$$\boldsymbol{p} = SOFTMAX(\mathbf{s}) = \frac{exp(\mathbf{s})}{\mathbf{1}^T exp(\mathbf{s})} \tag{2}$$

The parameters **W** and **b** of our classifier are what we have to learn by exploiting labelled training data. The parameters are set by minimizing the cross-entropy loss plus a regularization term on **W**. So the cost function is the following:

$$J(D, \lambda, \boldsymbol{W}, \boldsymbol{b}) = \frac{1}{D} \sum_{x,y \in D} l_{cross}(\mathbf{x}_i, y_i, \mathbf{W}, \mathbf{b}) + \lambda \sum_{i,j} \mathbf{W}_{ij}^2 \tag{3}$$

In this assignment this optimization problem is solved via mini-batch gradient descent where the update of both parameters is described in equations 4 and 5:

$$\mathbf{W}^{t+1} = \mathbf{W}^t - \eta \frac{\partial J(\mathbf{B}^{t+1}, \lambda, \mathbf{W}, \mathbf{b})}{\partial \mathbf{W}} \tag{4}$$

$$\mathbf{b}^{t+1} = \mathbf{b}^t - \eta \frac{\partial J(\mathbf{B}^{t+1}, \lambda, \mathbf{W}, \mathbf{b})}{\partial \mathbf{b}} \tag{5}$$

We compute the gradient with regularization terms at each mini-batch with equations 6 and 7:

$$\frac{\partial J}{\partial \mathbf{W}} = \frac{1}{|D|} \sum \mathbf{g}^T \mathbf{x}^T + 2\lambda \mathbf{W} \tag{6}$$

$$\frac{\partial J}{\partial \mathbf{b}} = \frac{1}{|D|} \sum \mathbf{g} \tag{7}$$

where

$$\mathbf{g} = -\frac{\mathbf{y}^T}{\mathbf{y}^T \mathbf{p}} (diag(\mathbf{p}) - \mathbf{p}\mathbf{p}^T) = -(\mathbf{y} - \mathbf{p})^T \tag{8}$$

3 Results

First of all the analytical gradients calculation is tested compared with the numerical gradients provided in the course. The numerical and analytical computed gradients are compared by computing the relative error between them and check if this error is small. The error for the W gradient and b gradient are lower than 1e-6 for $\lambda = 0$ and batch size= 1 and for batch size= $\{50, 100\}$ W gradient error is a bit higher, of the order of 1e-4.

Secondly, 4 tests were ran to see the effect of the parameters lambda and eta. Graphs of the total loss and the cost function on the training data and the validation data after each epoch of the mini-batch gradient descent algorithm are shown and also figures with the learnt weight matrix after the completion of training. for the following parameter:

1. **Configuration 1: $\lambda = 0$, n_epochs = 40, n_batch = 100, $\eta = 0.1$**

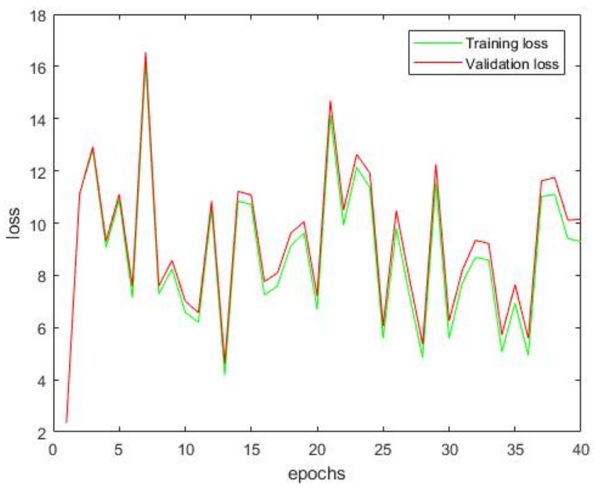

Figure 1: Loss function with $\lambda = 0$, $\eta = 0.1$, epochs = 40 and n_batch = 100.

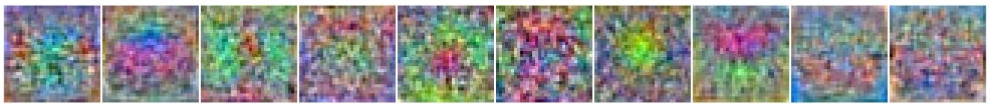

Figure 2: Weight matrix with $\lambda = 0$, $\eta = 0.1$, epochs = 40 and n_batch = 100.

Loss function of this configuration is presented in Fig.1 and the learnt weight matrix in Fig.2. After the training the network achieves a Test Accuracy of 20.93%.

2. **Configuration 2: λ = 0, n_epochs = 40, n_batch = 100, η = 0.01**

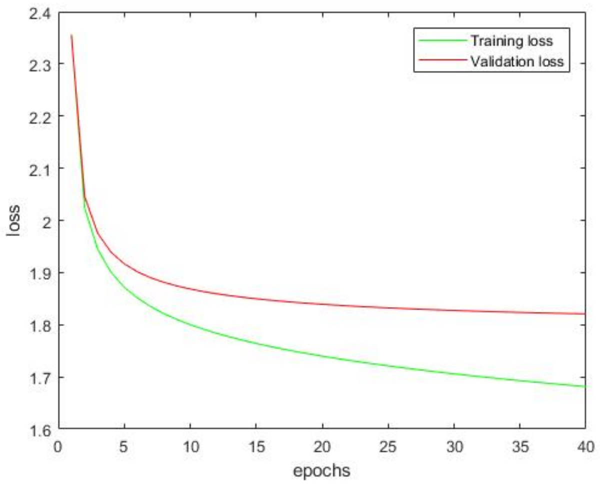

Figure 3: Loss function with $\lambda = 0$, $\eta = 0.01$, epochs = 40 and n_batch = 100.

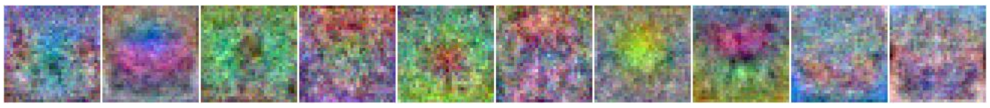

Figure 4: Weight matrix with $\lambda = 0$, $\eta = 0.01$, epochs = 40 and n_batch = 100.

Loss function of this configuration is presented in Fig.3 and the learnt weight matrix in Fig.4. After the trainingthe network achieves a Test Accuracy of 36.83%.

3. **Configuration 3: λ = 0.1, n_epochs = 40, n_batch = 100, η = 0.01**
   Loss function of this configuration is presented in Fig.5 and the learnt weight matrix in Fig.6. After the training the network achieves a Test Accuracy of 33.39%.

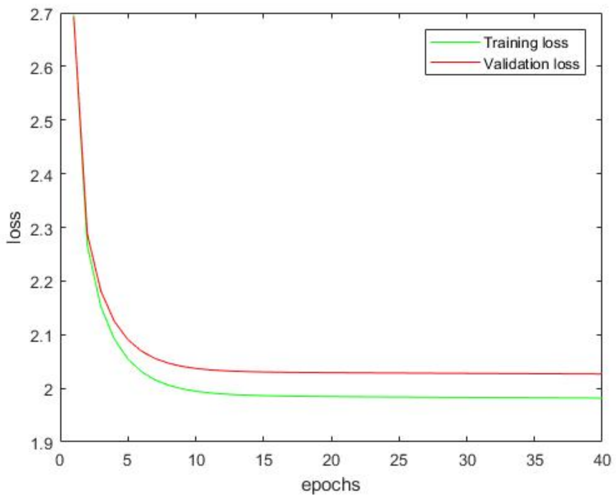

Figure 5: Loss function with $\lambda = 0.1$, $\eta = 0.01$, epochs = 40 and n_batch = 100.

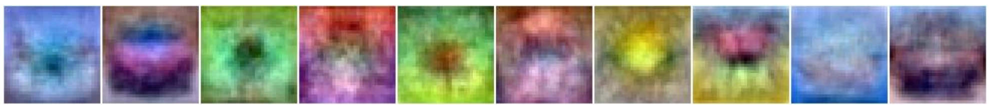

Figure 6: Weight matrix with $\lambda = 0.1$, $\eta = 0.01$, epochs = 40 and n_batch = 100.

4. **Configuration 4: $\lambda = 1$, n_epochs = 40, n_batch = 100, $\eta = 0.01$**

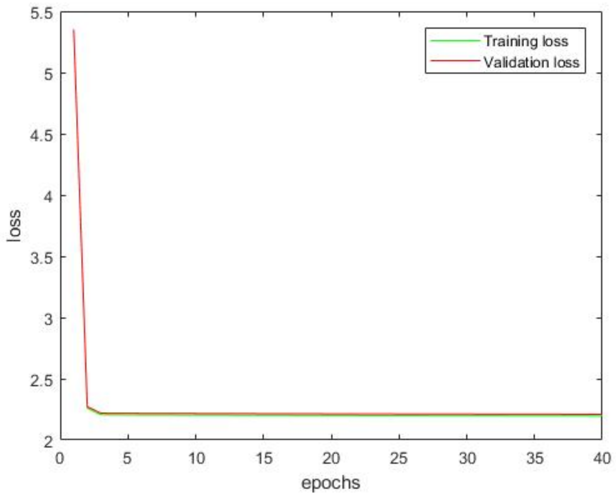

Figure 7: Loss function with $\lambda = 1$, $\eta = 0.01$, epochs = 40 and n_batch = 100.

Loss function of this configuration is presented in Fig.7 and the learnt

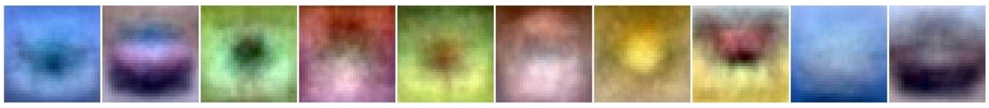

Figure 8: Weight matrix with $\lambda = 1$, $\eta = 0.01$, epochs = 40 and n_batch = 100.

weight matrix in Fig.8. After the training the network achieves a Test Accuracy of 21.94%.

4 Conclusion

A high learning rate means that the loss cost function will have a faster convergence but it will be unstable, with an inefficient zig-zag path. With a low learning rate the convergence will be slower but the update smoother.

The regularization terms could effectively avoid overfitting as we can check in Fig.5 and Fig.7 where the gap between training and validation accuracy is lower. The generalization of the network is improved and the test accuracy increases. However if it is too high it may decrease the size of the weight W, which may increase the bias.