Deep Learning in Data Science (DD2424) Report to Assignment 1

Cong Gao

March 24, 2020

1 Introduction

In this assignment I mainly trained and tested a one-layer network with multiple outputs to classify images from the CIFAR-10 dataset and using mini-batch gradient descent algorithm.

2 Methods

2.1 Dataset

I used the data in the file *data_batch_1.mat* for terainning, the file *data_batch_2.mat* for validation and the file *test_batch.mat* for test.

2.2 Mathematical details of the network

Given an input vector x, of size $d \times 1$ the classifier outputs a vector of probabilities, p $(K \times 1)$:

$$s = Wx + b$$

$$p = softmax(s)$$

$$softmax(s) = \frac{exp(s)}{1^T exp(s)}$$

The predicted class corresponds to the label with the highest probability:

$$k^* = arg \max_{1 \leq k \leq K} \{p_1, p_2, \cdots, p_K\}$$

The cost function containing regularization term is as follows:

$$J(D, \lambda, W, b) = \frac{1}{D} \sum_{(x,y) \in D} l_{cross}(x, y, W, b) + \lambda \sum_{i,j} W_{i,j}^2$$

$$l_{cross}(x, y, W, b) = -log(y^T p)$$

Then I solved this optimization problem via mini-batch gradient descent:

$$W^{t+1} = W^t - \eta \frac{\partial J}{\partial W^t}$$

$$b^{t+1} = b^t - \eta \frac{\partial J}{\partial b^t}$$

Where η is the learning rate.

$$\frac{\partial J}{\partial W} = \frac{1}{n} G_{batch} X_{batch}^T + 2\lambda W, \quad \frac{\partial J}{\partial b} = \frac{1}{n} G_{batch} 1_n$$

Where n is the size of each mini batch.

$$G_{batch} = P_{batch} - X_{batch}$$

3 Results

3.1 Gradient computation check

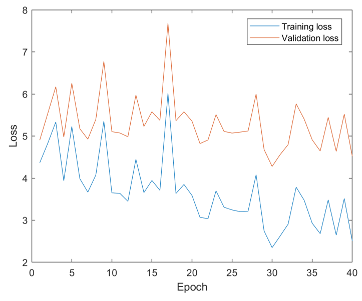
The numerical gradient should be compared to the analytical gradient to ensure that the analytical gradient is computed correctly. One way to check gradient computation is to compare the numerically and analytically computed gradient vectors (matrices) by examining their absolute differences and declaring, if all these absolutes difference are small ($\leq 1e - 6$). A more reliable method is to compute the relative error between a numerically computed gradient value g_n and an analytically computed gradient value g_a, which is shown as below.

$$\frac{|g_a - g_n|}{max(eps, |g_a| + |g_n|)}$$

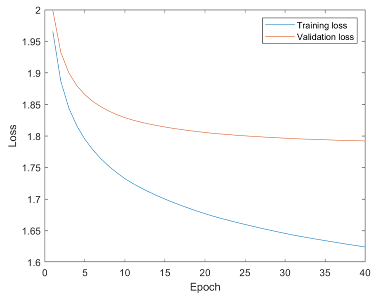
where eps is a very small positive number.
To check my gradient computation algorithm, I chose the first method and computed the analytically computed and numerically computed gradients on reduced version of the input data with reduced dimensionality. The mean error was: w: 2.33e-6, b: 1.58e-6. Then I can draw the conclusion that the analytical gradient computation was correct.

3.2 Results of different parameter settings

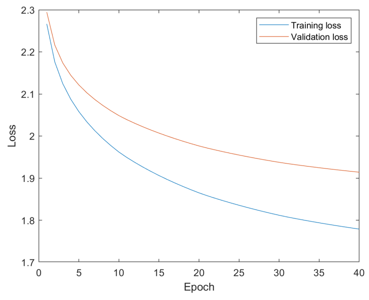
–Setting 1: $\lambda = 0, epochs = 40, batch = 100, \eta = 0.1$
–Setting 2: $\lambda = 0, epochs = 40, batch = 100, \eta = 0.001$
–Setting 3: $\lambda = 0.1, epochs = 40, batch = 100, \eta = 0.001$
–Setting 4: $\lambda = 1, epochs = 40, batch = 100, \eta = 0.001$
The graphs of the total loss and the cost function on the training data and the validation data after each epoch of the mini-batch gradient descent algorithm are as follow.



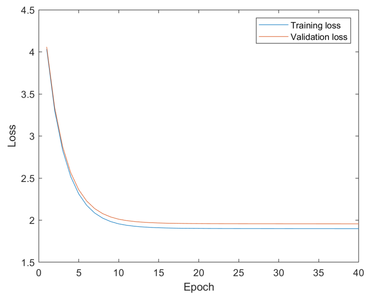
(a) Graph of total loss for setting 1



(b) Graph of total loss for setting 2



(c) Graph of total loss for setting 3



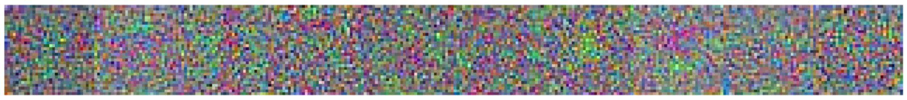
(d) Graph of total loss for setting 4

The classification accuracy on training set, validation set, and test set is as follows.

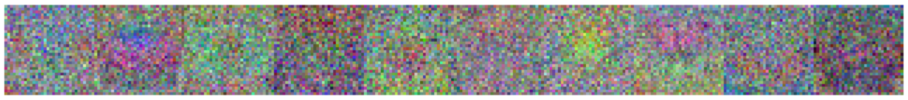
	Training data	Validation data	Test data
Setting 1	47.20%	28.94%	29.08%
Setting 2	45.16%	38.56%	38.82%
Setting 3	44.32%	38.66%	39.11%
Setting 4	40.21%	36.47%	37.48%

Table 1: Classification Accuracy

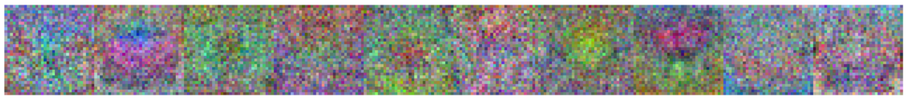
Images representing the learnt weight matrix after the completion of training are as follow.



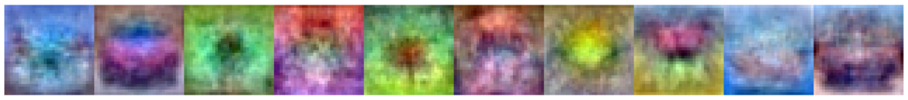
(a) The learnt weight matrix for setting 1



(b) The learnt weight matrix for setting 2



(c) The learnt weight matrix for setting 3



(d) The learnt weight matrix for setting 4

As we can see from the results above, regularization can prevent the network from overfitting and thus make the network get better generalization performance. The amount of regularization, that is, λ controls the trade-off between decreasing loss and decreasing the complexity of the network. The generalization performance increases with the amount of regularization but too large amount of regularization can make the network fail to work.
Also we need to select correct learning rate, otherwise if we select large learning rate, it is likely that the learning process will be unstable and even the network cannot converge and if we select too small learning rate, it will cost much more steps, thus much more time, for the network to converge.