Assignment 1 - DD2424

August Regnell 970712-9491

6 April, 2021

Contents

1 Introduction

In this assignment the task was to train and test a one layer neural network with multiple outputs to classify images from the CIFAR-10 dataset. The network was trained using mini-batch gradient descent applied to a cost function. The cost function was defined as sum of the cross-entropy loss of the classifier applied to the labelled training data and an L_2 regularization on the weight matrix.

2 Tests of analytical gradient

I managed to successfully write the functions to correctly compute the gradient analytically. I made sure of this by comparing the analytically computed gradient with the numerically computed gradient (code taken from canvas). More specifically, I compared with the function *ComputeGradsNumSlow*.

To make sure the computed gradients were correct, I checked that the relative error was small (`<1e-6`) using the following expression

$$r_e = \frac{|g_a - g_n|}{\max(\text{eps}, |g_a| - |g_n|)}$$

where g_a is the analytical gradient and g_n the numerical.

I started of by reducing the dimension of the data (to $d = 20$) and only using on datapoint while setting $\lambda = 0$. After the having found no non-small relative eror (they were around `<1e-9`) I increased the number of datapoints to 100 and the dimension to 200. Finding no non-small errors I used these alternatives as the final test: all dimensions (and 100 datapoints) and all datapoints (and 20 dimensions). None of them found any non-small errors.

3 Experiments

3.1 Initial experiment

For the initial experiment the following settings were used:

`n_batch=100, eta=.001, n_epochs=20, lambda=0`

These were the results:

`Training accuracy = 43.23%, Test accuracy = 38.47%`

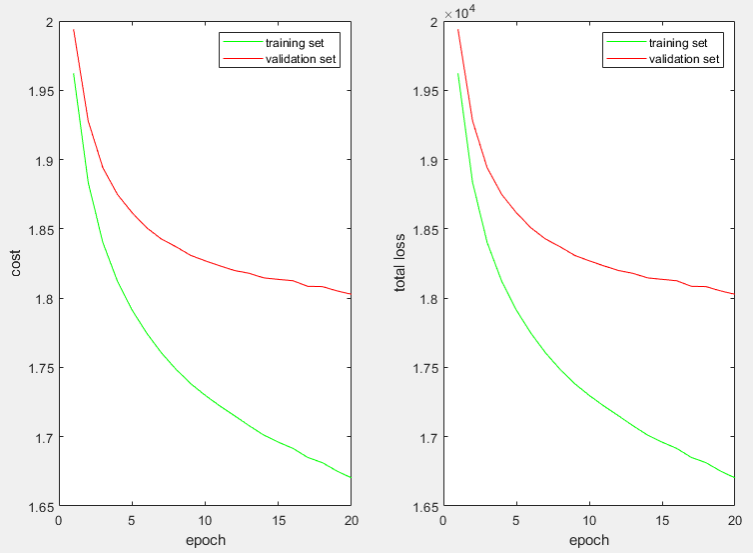

Figure 1: Total loss and the cost function of the initial experiment

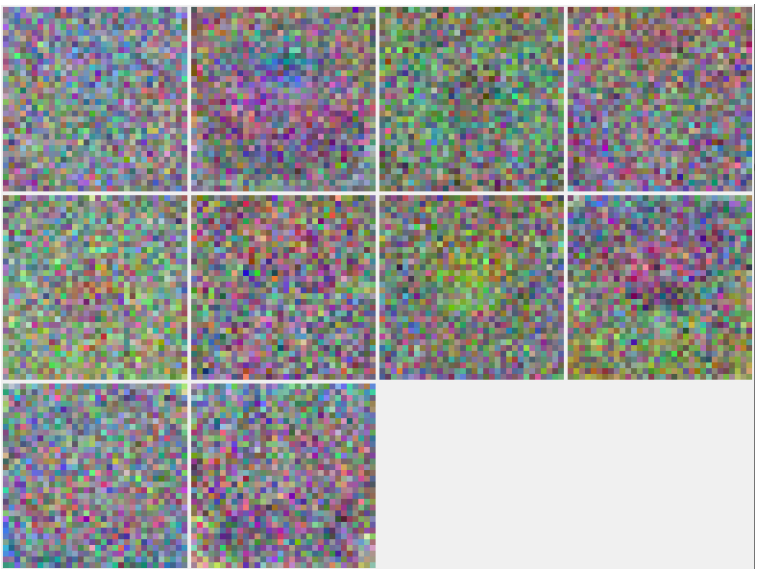

Figure 2: Images representing the learnt weight matrix after the completion of training of the initial experiment

3.2 Subsequent experiments

In this section follows the four experiments listed in the instructions

3.2.1 Experiment 1

For experiment 1 the following settings were used:

```
n_batch=100, eta=.1, n_epochs=40, lambda=0
```

These were the results:

```
Training accuracy = 39.04%, Test accuracy = 27.88%
```

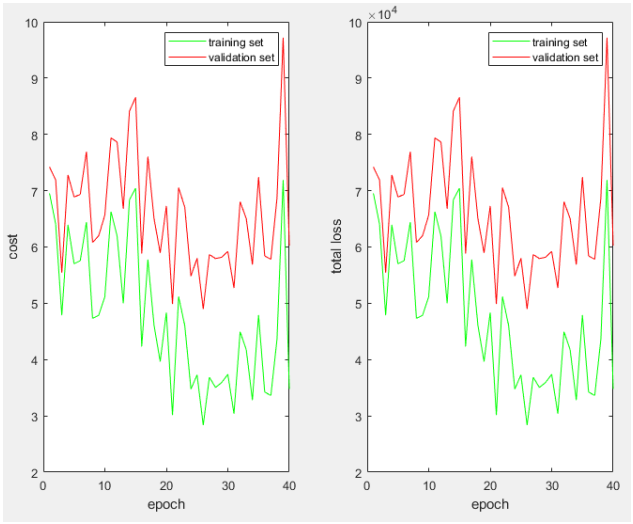

Figure 3: Total loss and the cost function of the first experiment

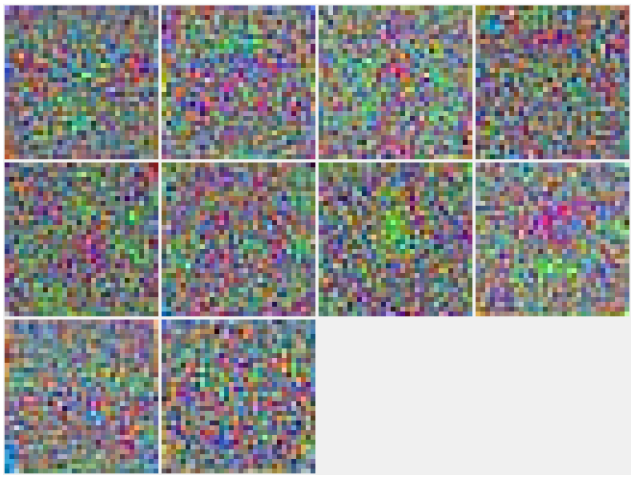

Figure 4: Images representing the learnt weight matrix after the completion of training of the first experiment

3.2.2 Experiment 2

For experiment 2 the following settings were used:

`n_batch=100, eta=.001, n_epochs=40, lambda=0`

These were the results:

`Training accuracy = 45.75%, Test accuracy = 38.53%`

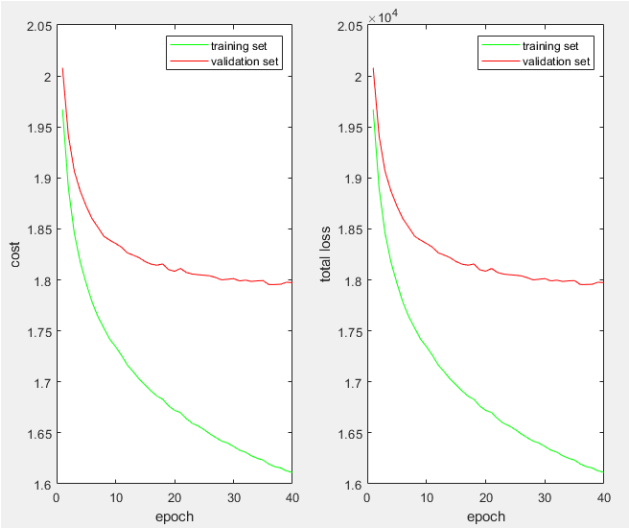

Figure 5: Total loss and the cost function of the second experiment

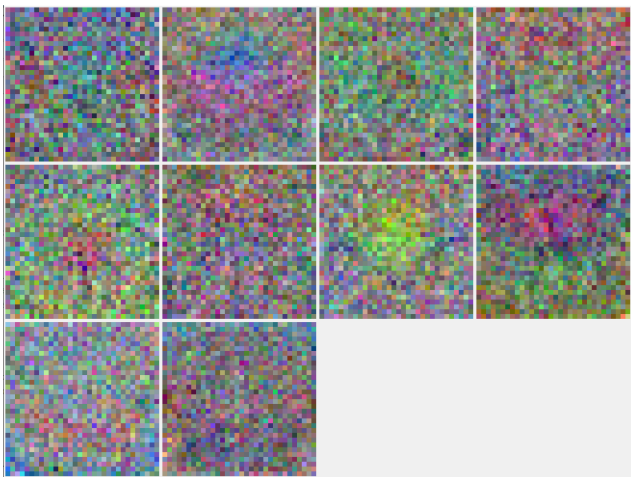

Figure 6: Images representing the learnt weight matrix after the completion of training of the second experiment

3.2.3 Experiment 3

For experiment 3 the following settings were used:

`n_batch=100, eta=.001, n_epochs=40, lambda=0.1`

These were the results:

`Training accuracy = 44.47%, Test accuracy = 39.10%`

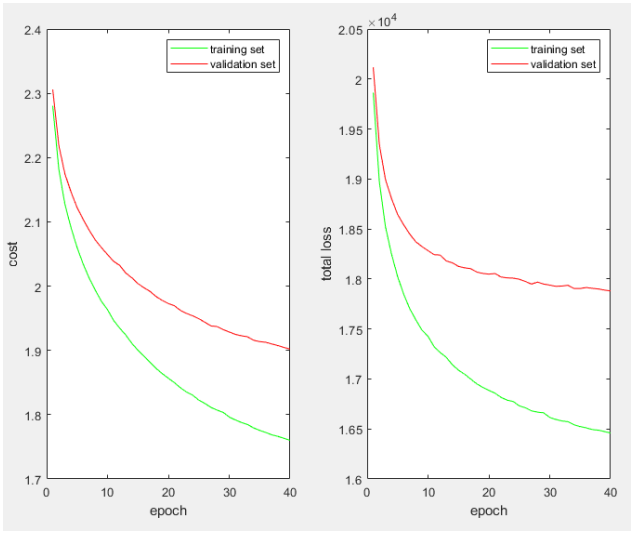

Figure 7: Total loss and the cost function of the third experiment

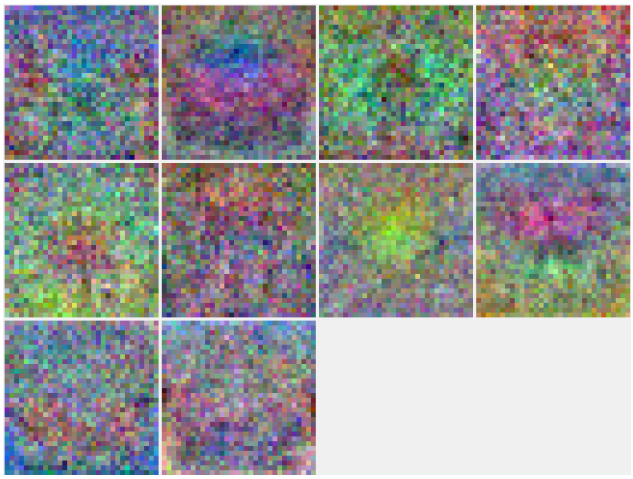

Figure 8: Images representing the learnt weight matrix after the completion of training of the third experiment

3.2.4 Experiment 4

For experiment 4 the following settings were used:

`n_batch=100, eta=.001, n_epochs=40, lambda=1`

These were the results:

`Training accuracy = 39.84%, Test accuracy = 37.54%`

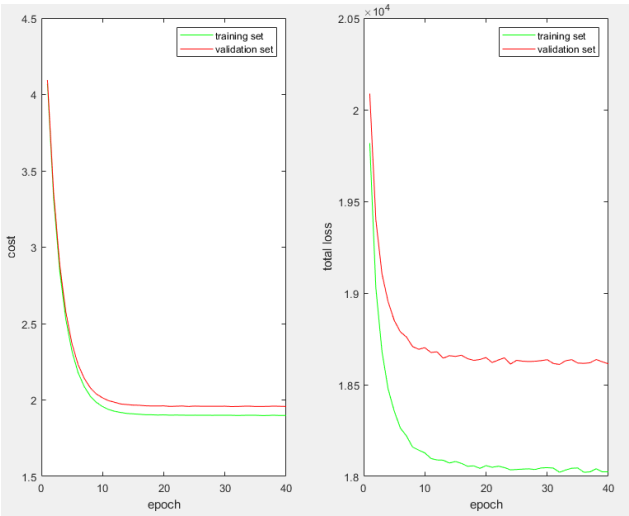

Figure 9: Total loss and the cost function of the fourth experiment

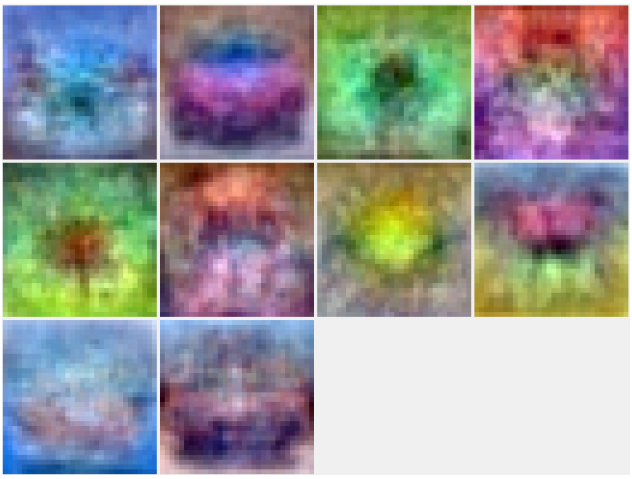

Figure 10: Images representing the learnt weight matrix after the completion of training of the fourth experiment

4 The effects of the hyper parameters

Here follows a short discussion about how the choice of hyper parameters affects the results.

4.1 The learning rate - η

The learning rate affects how fast the algorithm moves on the iso-contour of the loss function. If the iso-contour has steep and thin "valleys" the algorithm will have a hard time entering them, and thus finding the local minimum. This can be seen in the first experiment, where the cost and the total loss are fluctuating between lower and higher values, see figure 3. Comparing this with the later experiments, we see that the lower learning rate enables the algorithm to find the enter the valleys.

4.2 The regularization parameter - λ

In this assignment we were to use an L_2 regularization on the weight matrix, which intuitively could be understood as penalizing large absolute values of the weight matrix. No regularization means $\lambda = 0$, and for $\lambda > 0$ the regularization increases the larger the λ.

This regularization counteracts the over-fitting tendency. Thus a higher λ should *increase* the test accuracy but *decrease* the training accuracy. This can be seen in experiment 3. However, increasing it too much may lead to under-fitting, which we see in experiment 4. This can also be seen by observing that the lines for the cost in figure 9 are very close for the training and the validation set.

We also observe another effect - the visualization of the weight matrix becomes clearer when the regularization is added. This could be explained by that the network now more easily is able to capture the general pattern of the image classes. Examples of this are in figure 8 and 10 where the second and last pictures (left to right and top to bottom) can be seen as their classes, *car* and *truck* respectively. We also see a lot of blue in the first and second last picture, which represent *airplane* and *boat* respectively. However, I can not give a good explanation of why the objects seem clearer in figure 10 than 8.