ASSIHMENT 1

Diego Argüello Ron

Deep Learning in Data Science (DD2424)

11 April 2018

1 Introduction

The objective of this assignment has been training and testing a one layer network with multiple outputs to classify images from the CIFAR-10 dataset.
It must be pointed out that the final (and cleaned up) code (Assignemt 1.m) and all the functions implemented have been added together in a folder and uploaded to Canvas.

2 Computing Gradients Analytically

As it can be seen in the code uploaded in Canvas, there was no problem while creating the functions required to carry out the assignment. Nevertheless, it is worth noticing how the gradients have been calculated analytically.
In order to do it, a function of the next form has been created:

```
function [grad W, grad b] = ComputeGradients(X, Y, P, W, lambda)
```

Listing 1: Calculating Gradients Analitically

In it, the steps that can be found in the slices of the Lecture 3 have been followed. Thus, g has been defined as:

$$g = \frac{y^T}{y^T p}(diag(p) - pp^T) \tag{1}$$

The gradient of J w.r.t. the bias vector and the gradient of J w.r.t. the weight matrix W will be respectively:

$$\frac{\partial J}{\partial b} = g \tag{2}$$

$$\frac{\partial J}{\partial b} = g^T x^T + 2\lambda W \tag{3}$$

In order to ensure that no mistakes have been made when computing gradients, the analytic gradient computations exposed above were check against numerical estimations of the gradients. There were two methods at our disposal for this: the based on the centered difference and the finite difference one. The next results have been obtained using the last of these methods (more accurate).
Thus, the relative error between a numerically computed gradient value g_n and an analytically computed gradient value g_a was calculated:

$$\frac{|g_a - g_n|}{max(esp, |g_a| + |g_n|)} \tag{4}$$

where esp is a very small positive number, taken as 0.
Initially, it was checked in mini-batches of size 1 and with no regularization. Afterwards the size of mini-batch was increased (until 100) and regularization included ($\lambda = 0.1$).
It must be pointed out that the result could d give the next intervals: $relativeerror > 1e{-}1$ usually means the gradient is probably wrong, $1e{-}2 > relativeerror > 1e{-}4$ means that the gradient may be wrong, $1e-4 > relativeerror$ is usually okay for objectives

with kinks. But if there are no kinks (e.g. use of tanh nonlinearities and softmax), then 1e-4 is too high. For 1e-7 and less should be enough.
Finally, the values obtained were:

Batch size	λ	Relative Error b	Relative Error W
1	0	$1.4428 \cdot 10^{-9}$	$7.9907 \cdot 10^{-8}$
50	0	$5.897 \cdot 10^{-7}$	$3.6129 \cdot 10^{-4}$
100	0	$1.0409 \cdot 10^{-6}$	$4.6841 \cdot 10^{-4}$
1	0.1	$1.1246 \cdot 10^{-9}$	$4.6211 \cdot 10^{-6}$
50	0.1	$6.2245 \cdot 10^{-9}$	$15.0757 \cdot 10^{-4}$
100	0.1	$8.1013 \cdot 10^{-8}$	$8.5392 \cdot 10^{-5}$

3 Main results

Finally, the main results will be exposed and analyzed.

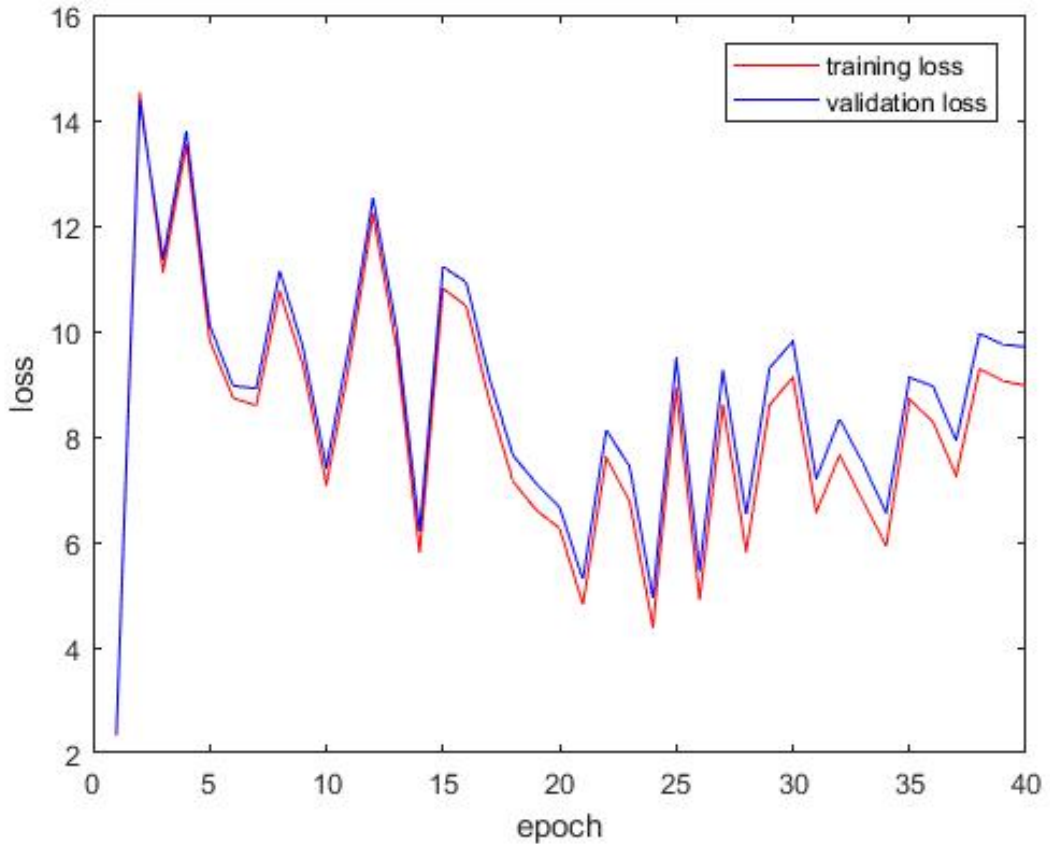

Figure 1: Loss for $\lambda = 0, n_epochs = 40, n_batch = 100, \eta = 0.1$

Figure 2: Weight Matrix for $\lambda = 0, n_epochs = 40, n_batch = 100, \eta = 0.1$

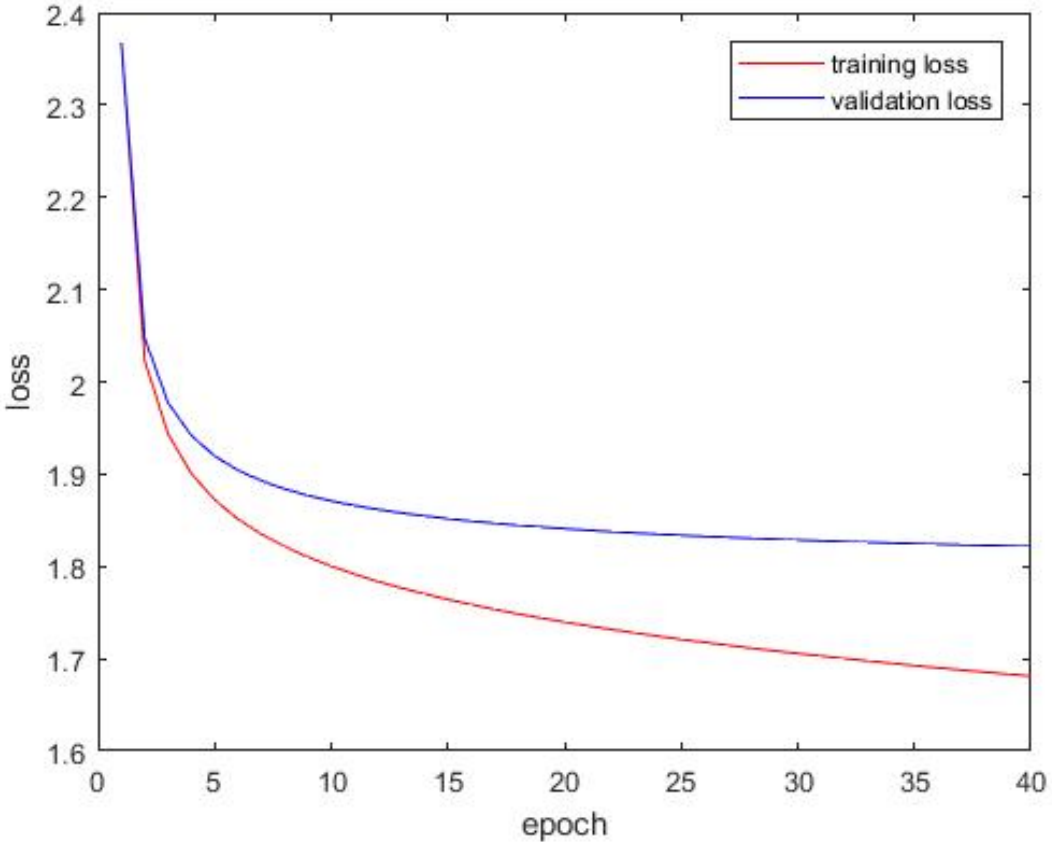

Figure 3: Loss for $\lambda = 0, n_epochs = 40, n_batch = 100, \eta = 0.01$

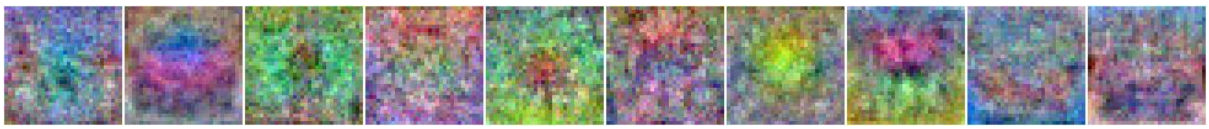

Figure 4: Weight Matrix for $\lambda = 0, n_epochs = 40, n_batch = 100, \eta = 0.01$

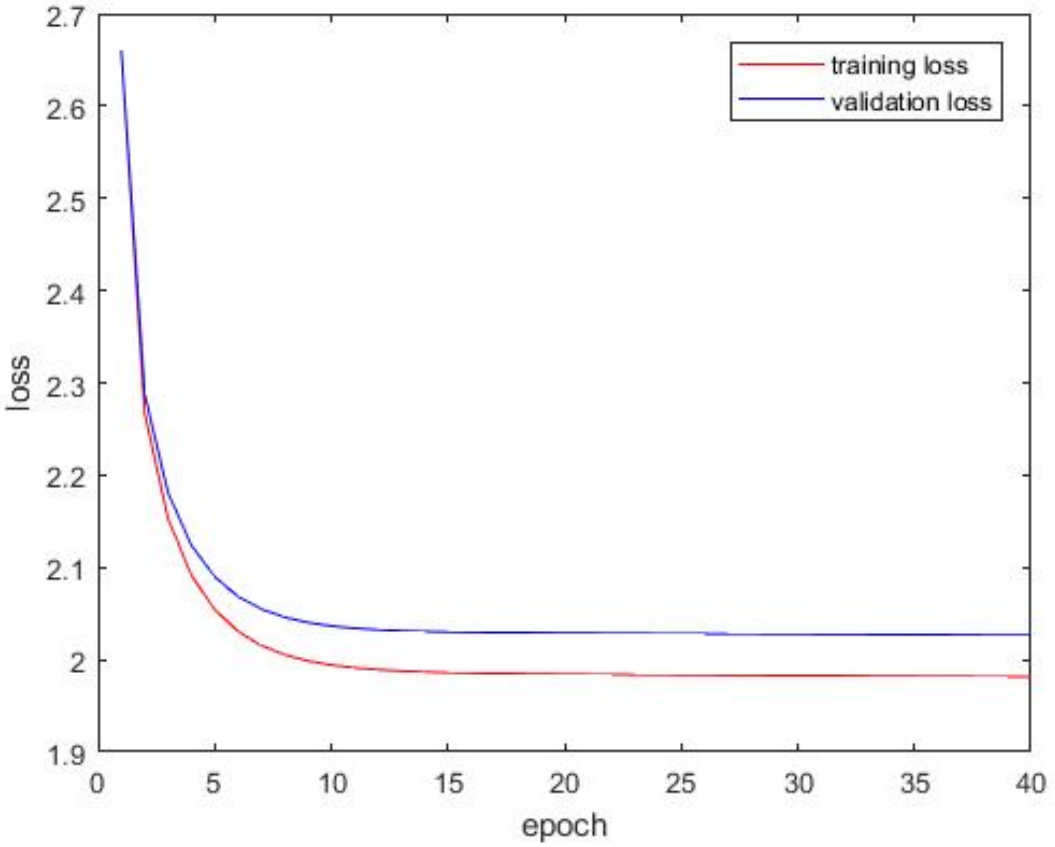

Figure 5: Loss for $\lambda = 0.1, n_epochs = 40, n_batch = 100, \eta = 0.01$

Figure 6: Weight Matrix for $\lambda = 0.1, n_epochs = 40, n_batch = 100, \eta = 0.01$

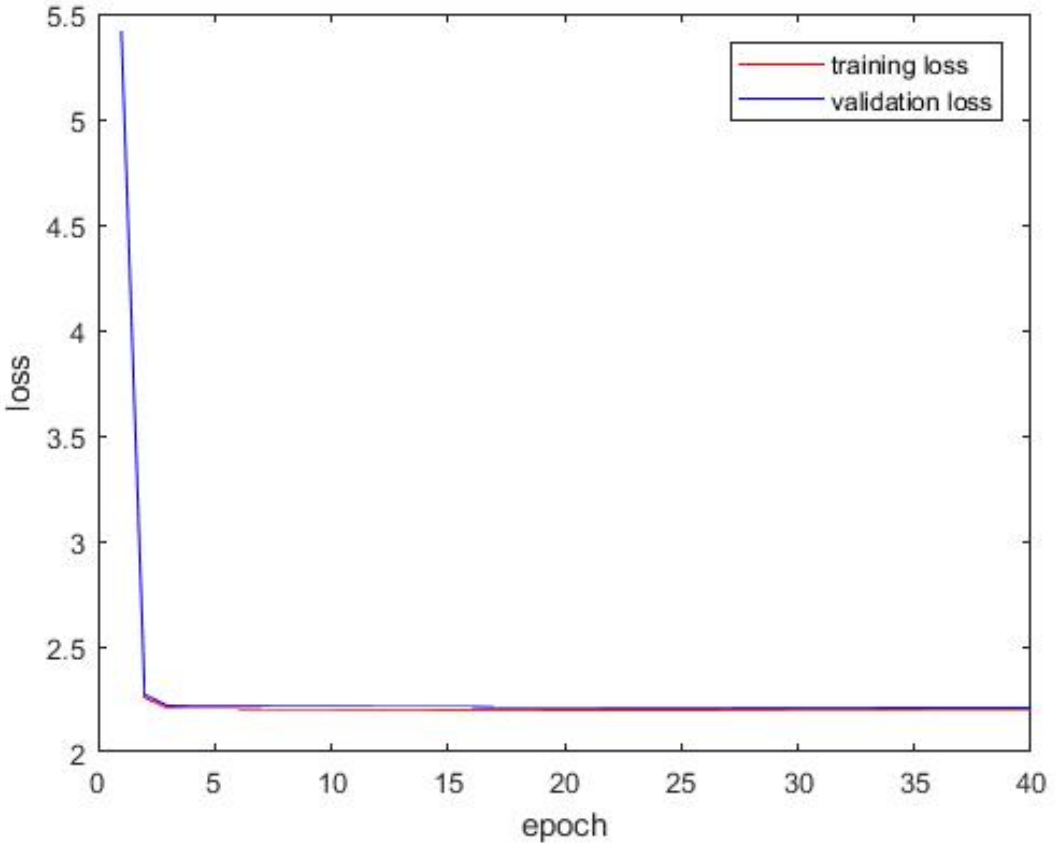

Figure 7: Loss for $\lambda = 1, n_epochs = 40, n_batch = 100, \eta = 0.01$

Figure 8: Weight Matrix for $\lambda = 1, n_epochs = 40, n_batch = 100, \eta = 0.01$

Number of epochs	λ	Batch size b	Accuracy training data	Accuracy test data
40	0	100	32	27.2
40	0	100	41.5	36.89
40	0.1	100	34.8	33.5
40	1	100	22.2	22

After this the main conclusions will be:
- Learning rate:the larger η, the faster the convergence. Nevertheless, it will lead to a more erratic behavior of the loss function and a worse test data set accuracy.

- Regularization: The regularization term tries to avoid overfitting due to having a function too complex. Thus, increasing regularization reduces the variance as it can be seen with the graphs representing the loss functions. On the other hand, if the complexity of the function is limited too much, then it will not be able to represent the data (very low accuracy).