Assignment 2 - DD2424

August Regnell 970712-9491

14 April, 2021

Contents

1 Introduction

In this assignment the task was to train and test a two 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. Furthermore, there was a search for good parameter settings for the network's regularixation and the learning rate.

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*, and the faster but less accurate *ComputeGradsNum*.

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 one datapoint ($n = 1$) while setting $\lambda = 0$. After the having found no non-small relative error I increased the number of datapoints to 100 and the dimension to 200. The results can be seen in Figure 1.

ComputeGradsNumSlow	$\mathbf{W}_1$	0%
	$\mathbf{b}_1$	0%
	$\mathbf{W}_2$	0%
	$\mathbf{b}_2$	0%
ComputeGradsNum	$\mathbf{W}_1$	2.9%
	$\mathbf{b}_1$	10%
	$\mathbf{W}_2$	0%
	$\mathbf{b}_2$	90%

(a) d = 20, n = 1, $\lambda = 0$

ComputeGradsNumSlow	$\mathbf{W}_1$	0.08%
	$\mathbf{b}_1$	0%
	$\mathbf{W}_2$	0.2%
	$\mathbf{b}_2$	0%
ComputeGradsNum	$\mathbf{W}_1$	88.21%
	$\mathbf{b}_1$	92%
	$\mathbf{W}_2$	60.6%
	$\mathbf{b}_2$	100%

(b) d = 200, n = 100, $\lambda = 0$

Figure 1: Number of elements with a relative error larger than 1e-6.

Finding very few non-small errors with the more accurate *ComputeGradsNumSlow* in the previous tests I used these alternatives as the final test: even more dimensions ($d = 500$ and 10 datapoints) and even more datapoints with a high number of dimensions ($d = 500$ and 100 datapoints) with a non-zero lambda. The results can be seen in Figure 2, where we see that there were very few errors compared to *ComputeGradsNumSlow*. Interestingly, we notice that the faster and less accurate numerical calculation is in fact very inaccurate, often giving errors of 100% when the more accurate version gives no errors.

ComputeGradsNumSlow	$\mathbf{W}_1$	0.02%
	$\mathbf{b}_1$	0%
	$\mathbf{W}_2$	0%
	$\mathbf{b}_2$	0%
ComputeGradsNum	$\mathbf{W}_1$	100%
	$\mathbf{b}_1$	50%
	$\mathbf{W}_2$	100%
	$\mathbf{b}_2$	100%

(a) d = 500, n = 10, $\lambda = 0.1$

ComputeGradsNumSlow	$\mathbf{W}_1$	1.384%
	$\mathbf{b}_1$	0%
	$\mathbf{W}_2$	0%
	$\mathbf{b}_2$	0%
ComputeGradsNum	$\mathbf{W}_1$	100%
	$\mathbf{b}_1$	22%
	$\mathbf{W}_2$	93.2%
	$\mathbf{b}_2$	20%

(b) d = 500, n = 100, $\lambda = 0.1$

Figure 2: Number of elements with a relative error larger than 1e-6.

3 Replicating figures in instructions

In this section *Figure 3* and *4* found in the assignment are replicated. The data is recorded once every epoch, which results in slightly unsmooth curves.

3.1 Figure 3

To replicate *Figure 3* in the instructions the following settings were used:

```
n_batch=100, eta_min=1e-5, eta_max = 1e-1, n_s = 500, n_cycles=1, lambda=0.01
```

These were the results:

```
Training accuracy = 60.55%, Test accuracy = 46.07%
```

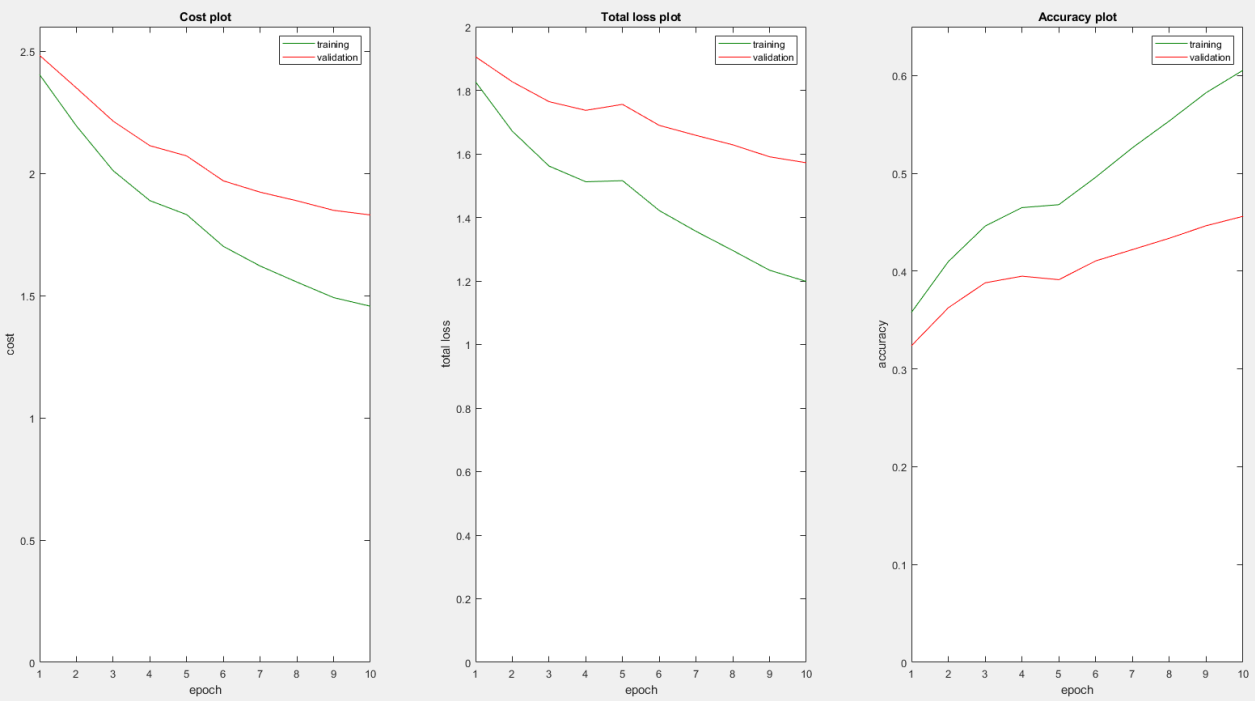

Figure 3: Cost, total loss and accuracy plots of the replicated experiment.

Comparing the replicated graphs to the original figure we notice that they are very similar. The small differences are probably due to the random initialization. However, we notice one significant difference, the replicated graphs are missing the "kink" between the zero:th and first recorded point. This is simply due to that I did not compute the cost, loss or accuracy before at least one training step was completed.

We also notice that this network probably needs more training, as the curves don't seem to flatten out.

3.2 Figure 4

To replicate *Figure 4* in the instructions the following settings were used:

`n_batch=100, eta_min=1e-5, eta_max = 1e-1, n_s = 800, n_cycles=3, lambda=0.01`

These were the results:

`Training accuracy = 70.95%, Test accuracy = 46.49%`

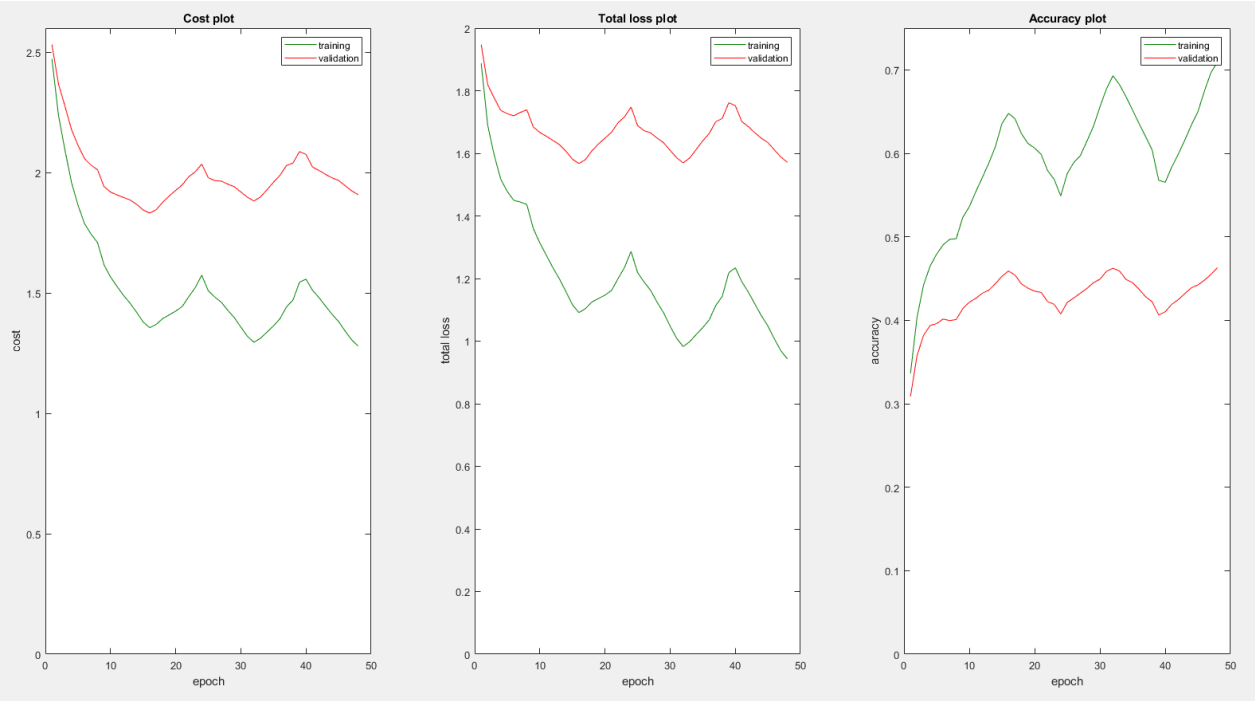

Figure 4: Cost, total loss and accuracy plots of the replicated experiment.

Comparing the replicated graphs to the original figure we notice that they are very similar. The small differences are probably due to the random initialization. However, we notice one significant difference, the replicated graphs are missing the "kink" between the zero:th and first recorded point. This is simply due to that I did not compute the cost, loss or accuracy before at least one training step was completed.

We also notice that this network probably doesn't need much more training, as the peaks/valleys of the lines for the validation data have flattened out.

4 Coarse search for λ

For the coarse search all training data was used, 5000 images for validation and the rest (45000) for training. The following parameters were used:

```
n_batch=100, eta_min=1e-5, eta_max = 1e-1, n_cycles=2
```

`n_s` was set to `n_s = 2 floor(n / n_batch)`.

λ was searched for by taking 8 random samples on the log-scale in the following way:

```
n_lambdas = 8;
l = l_min + (l_max - l_min) * rand(1, n_lambdas);
lambdas = 10 .^ l;
```

with `l_min = -1` and `l_min = -5`. This was then repeated two more times after adjusting `l_min` and `l_min` to make the range narrower in accordance to the accuracy achieved on the validation set. The results of the three searches can be seen in table 5.

Search range		**Sampled λ and validation accuracy**							
$\lambda_{min} = -5$		3.91e-04	1.22e-05	2.74e-02	7.55e-03	**7.70e-04**	2.23e-05	1.23e-05	1.55e-04
$\lambda_{max} = -1$		51.32%	51.14%	47.52%	51.24%	**51.98%**	51.54%	51.46%	51.16%
$\lambda_{min} = -4.7$		9.48e-05	2.17e-05	**5.76e-04**	3.33e-04	1.26e-04	2.80e-05	2.17e-05	6.40e-05
$\lambda_{max} = -3$		51.32%	51.22%	**52.02%**	51.46%	51.84%	51.52%	50.92%	51.48%
$\lambda_{min} = -3.89$		2.38e-04	1.33e-04	4.85e-04	3.90e-04	**2.66e-04**	1.47e-04	1.33e-04	2.03e-04
$\lambda_{max} = -3.22$		51.10%	51.48%	51.44%	51.52%	**52.00%**	51.46%	51.58%	51.13%

Figure 5: Results of coarse search for a good λ. The three highest validation accuracies and their corresponding λ is highlighted in bold text.

5 Training the network with the optimal λ

Using the best λ found in the previous section (5.7667486e-04) the network was now trained with all data, 1000 for validation and 49000 for training. The following settings were used:

```
n_batch=100, eta_min=1e-5, eta_max = 1e-1, n_s = 500, n_cycles=5, lambda=5.767e-04
```

These were the results:

```
Training accuracy = 58.36%, Test accuracy = 51.10%
```

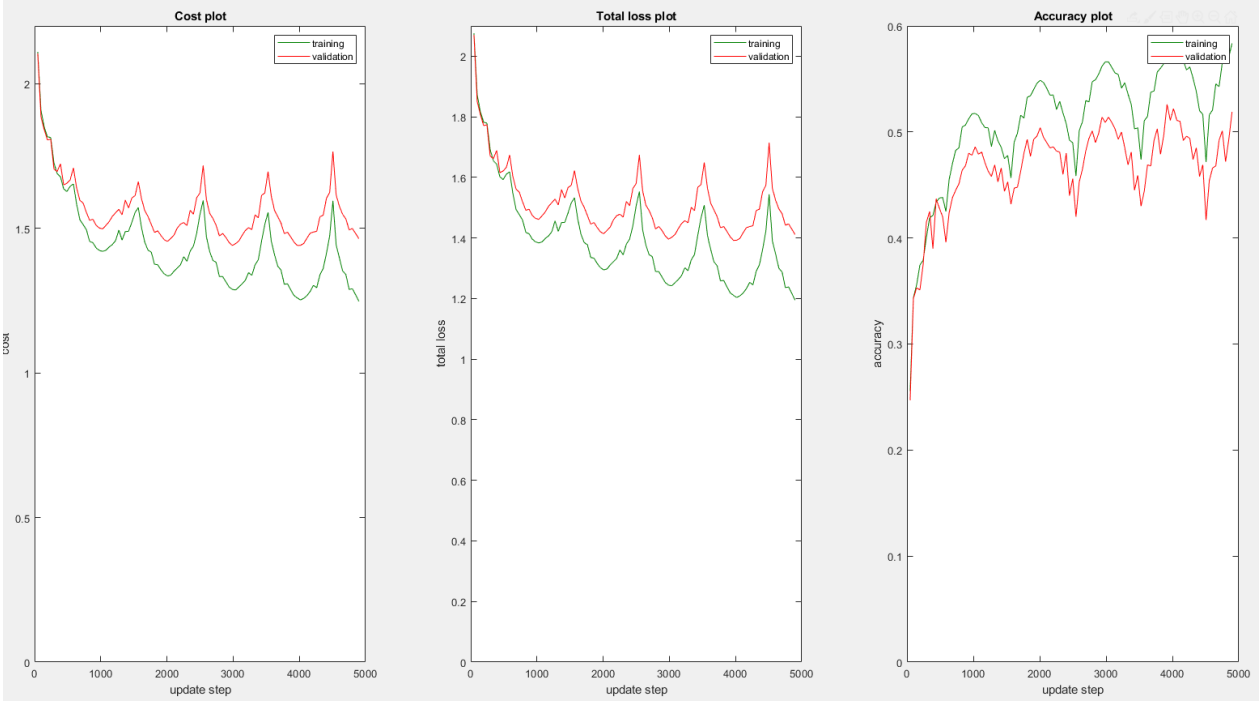

Figure 6: Cost, total loss and accuracy plots of the final network.

Note that for this experiment the data was collected ten times every epoch, which resulted in smoother graphs.