Lab 1 - Deep learning

Victor Stenmark (vstenm)

March 29, 2024

Contents

1 Implementation of functions and way of checking gradients

I managed to implement the functions to compute the gradient analytically correctly. To test this I calculated the gradient using both the analytical and the numerical method (with h = 10^{-6}) for the first 100 columns. I then compared the analytical and numerical gradient using the formula

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

where eps = 10^{-12} and checked that all values were less than 10^{-6}.

2 Training with different values of hyperparameters

The final accuracy of the test data set for the different models can be found in table 1. The graphs for the loss and the cost functions can be found in Figures 2-9.

Lambda	Number of epochs	Number of batches	η	Final accuracy
0	40	100	0.1	29.56%
0	40	100	0.001	39.14%
0.1	40	100	0.001	39.59%
1	40	100	0.001	37.53%

Figure 1: Table showing the model parameters and the final accuracy on the test dataset

3 Comment on regularisation and learning rate

Looking at the cost and loss curves in figures 2-9, it's clear that choosing an appropriate learning rate is very important. In figure 2, we see that the high learning rate causes the model to make adjustments that are too big. This causes the curves to jump around and results in the model not learning anything. With a learning rate of 0.001, on the other hand, we see a much smoother graph and a gradual decrease of both the loss and the cost.

Increasing the regularisation should penalize the model variants that have high weights. Ideally, this should reduce overfitting and make the model perform better on unseen data. Then if the regularisation term becomes too large, then the model can start to underfit since complicated models that accurately model the data get penalized.

This also appears to play out in the model. Without any regularisation, the accuracy is 39.14%, with a little regularisation it goes up to 39.59% and with a lot of regularisation the accuracy goes down to 37.53%. These differences could nevertheless be due to chance. It's unclear since I haven't run k-fold cross-validation or any statistical tests. What is clear, however, is that as the regularisation increases, the cost curves for the training data and the validation data start to converge. This can be seen most clearly in figure 8. Another interesting effect of increasing regularisation is that the weight matrix starts to look a lot less like random noise. This can be seen most evidently in figure 13.

4 Graphs of cost and loss functions

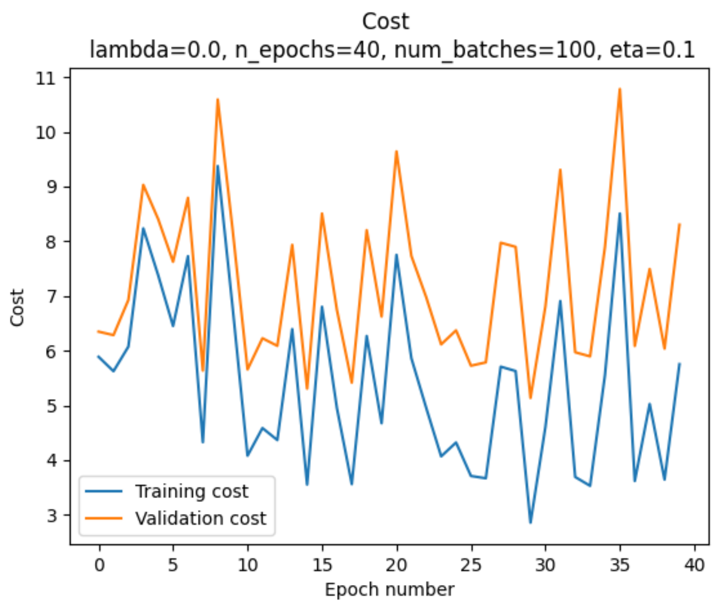

Figure 2: Curve of the training and validation cost for the first set of parameters

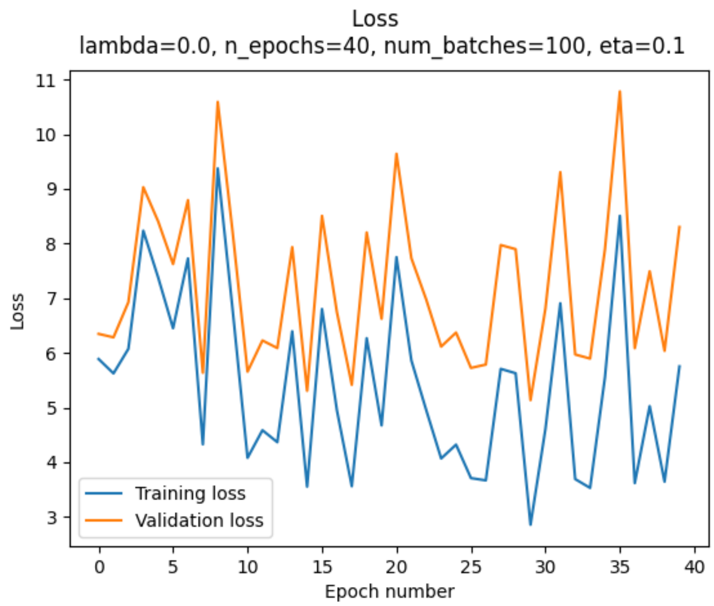

Figure 3: Curve of the training and validation loss for the first set of parameters

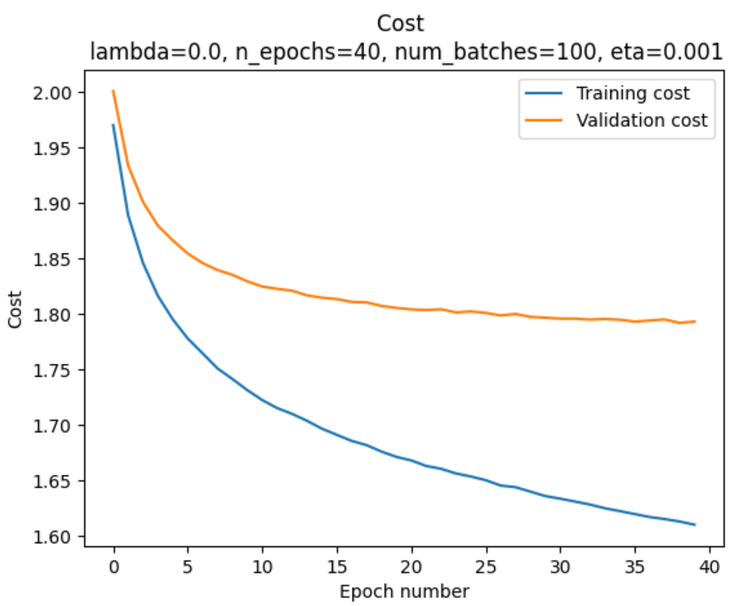

Figure 4: Curve of the training and validation cost for the second set of parameters

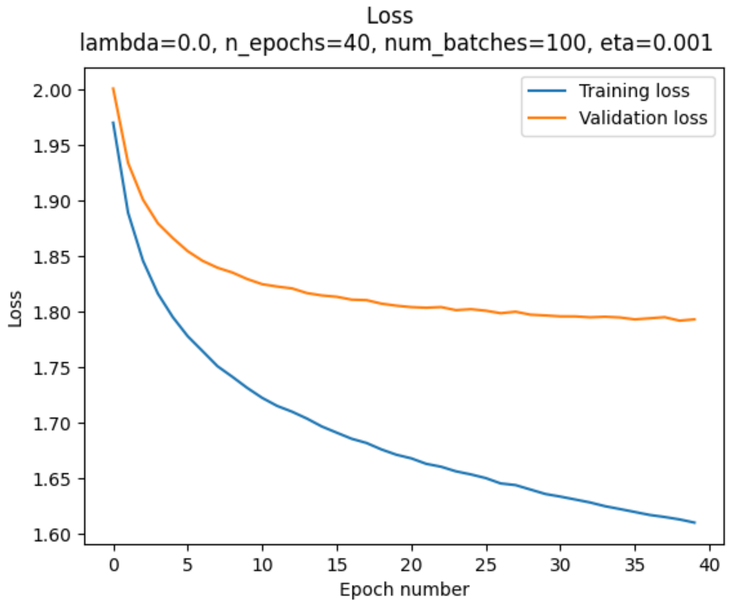

Figure 5: Curve of the training and validation loss for the second set of parameters

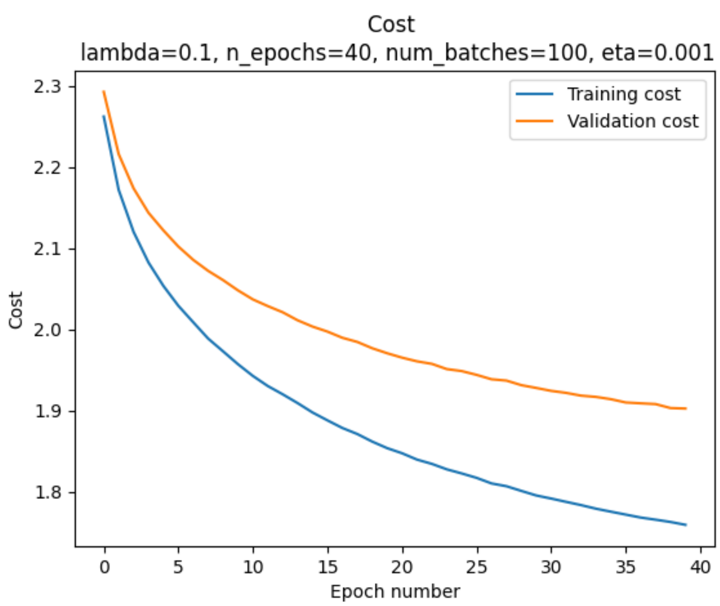

Figure 6: Curve of the training and validation cost for the third set of parameters

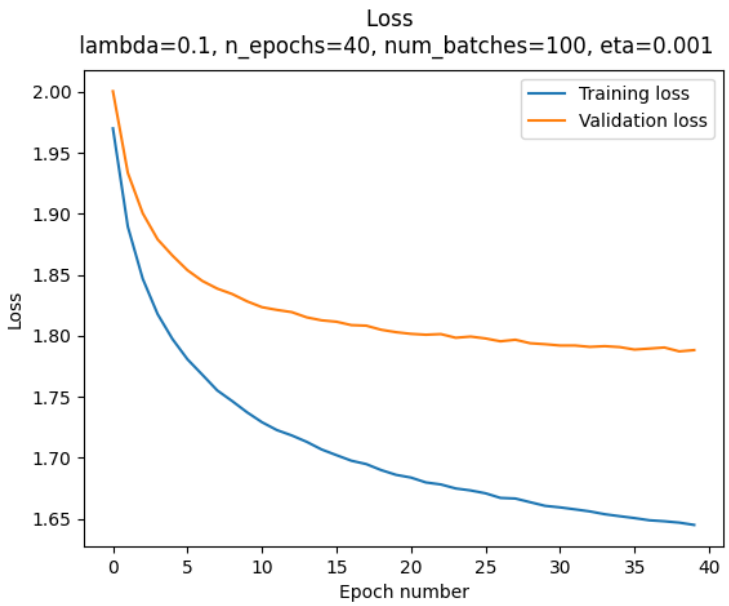

Figure 7: Curve of the training and validation loss for the third set of parameters

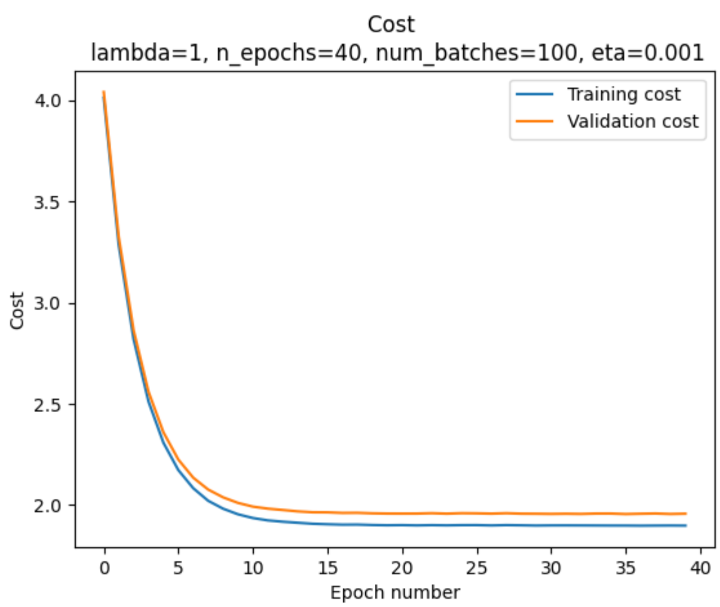

Figure 8: Curve of the training and validation cost for the fourth set of parameters

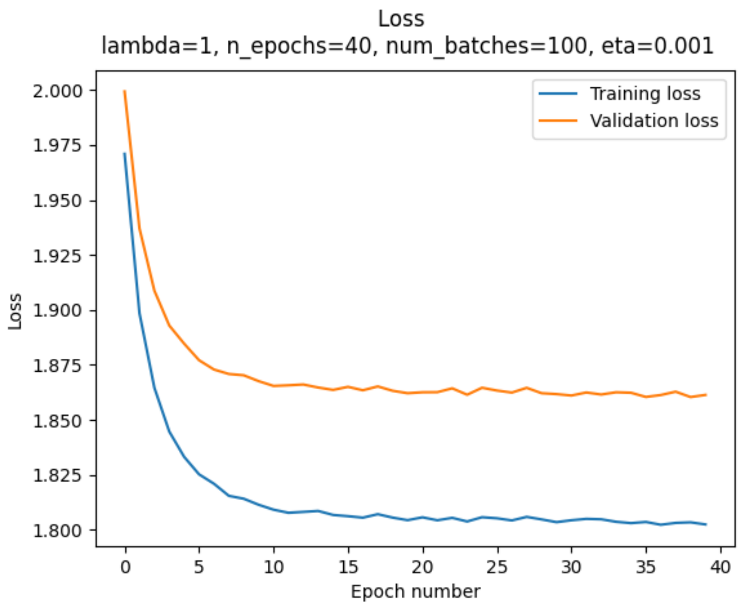

Figure 9: Curve of the training and validation loss for the fourth set of parameters

5 Weight matrices

In figures 10 to 13, it's possible to see the weight matrix for each of the parameter sets in figure 1.

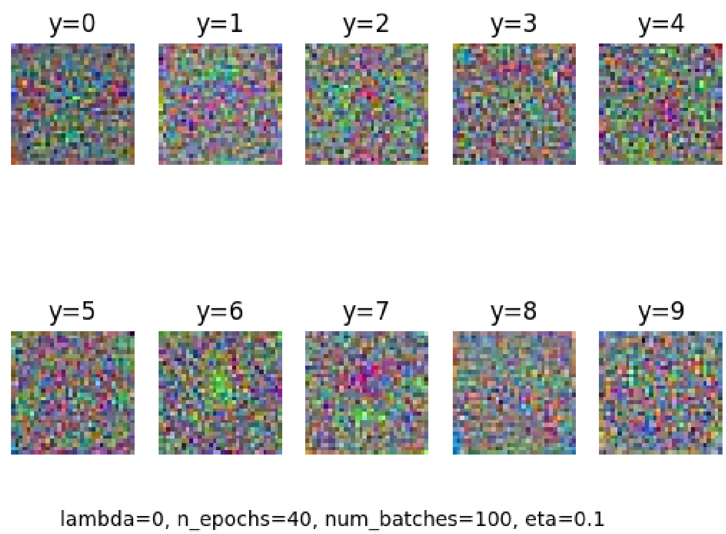

Figure 10: Visualisation of the weight matrix for the model with the parameters specified at the bottom of the picture.

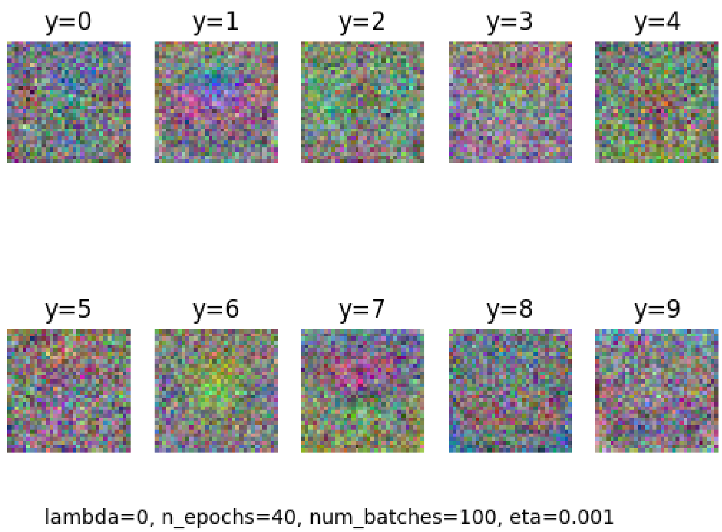

Figure 11: Visualisation of the weight matrix for the model with the parameters specified at the bottom of the picture.

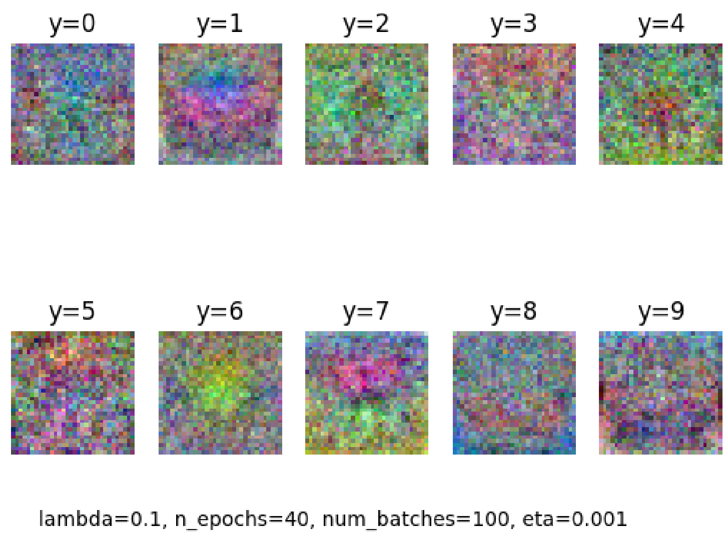

Figure 12: Visualisation of the weight matrix for the model with the parameters specified at the bottom of the picture.

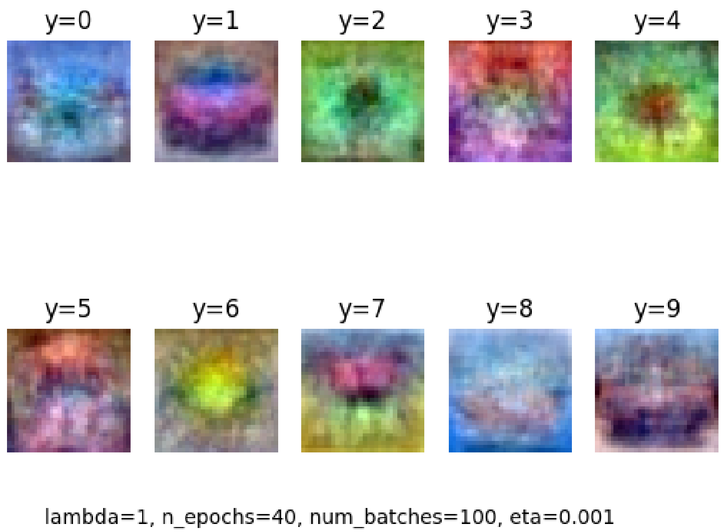

Figure 13: Visualisation of the weight matrix for the model with the parameters specified at the bottom of the picture.