Deep Learning DD2424 - Assignment 3

Anton Stagge

April 2019

1 Introduction

In this assignment I have created a kl-ayer dense network with batch normalization, and successfully trained it to classify CIFAR-10.

2 Test against numerical gradients

To test my new function that calculated gradients, I had to modify the Matlab code provided to calculate the gradient with respect to both the weights and the gamma and betas in the batch normalization. Then to test my analytically calculated gradients, I calculated the relative error between the two. Doing this, I managed to find a bug in my BatchNormBackPass function.

At first I started with just 3 image in the batch, lambda set to 0, a network with 1 hidden layer with 50 nodes and reduced the dimension of a picture to only 10 pixels. I then got the results:

Layer 0
gradient W0 relative error: 1.39e-05
gradient b0 relative error: 1.28e-10
gradient gamma0 relative error: 6.70e-07
gradient beta0 relative error: 5.24e-07

Layer 1
gradient W1 relative error: 9.57e-07
gradient b1 relative error: 1.13e-06

I then raised the *n_batch* to 10 and *lambda* to 0.005 and got the results:

Layer 0
gradient W0 relative error: 6.16e-06
gradient b0 relative error: 1.17e-10
gradient gamma0 relative error: 1.19e-06
gradient beta0 relative error: 1.19e-06

Layer 1
gradient W1 relative error: 2.32e-06
gradient b1 relative error: 2.30e-06

Feeling confident that my implementation was correct, I raised the dimension to 1000 and added a second hidden layer with 50 hidden nodes. This got me the result:

Layer 0
gradient W0 relative error: 5.01e-4
gradient b0 relative error: 4.69e-11
gradient gamma0 relative error: 4.88e-4
gradient beta0 relative error: 3.47e-4

Layer 1
gradient W1 relative error: 2.64e-4
gradient b1 relative error: 4.23e-09
gradient gamma1 relative error: 1.41e-06
gradient beta1 relative error: 1.90e-06

Layer 2
gradient W2 relative error: 3.61e-06
gradient b2 relative error: 4.38e-06

Finally I set $n\_batch = 100$ and the dimension back to 3072 and got the result:

Layer 0
gradient W0 relative error: 1.8513234424289293e-05
gradient b0 relative error: 1.5695317224714844e-11
gradient gamma0 relative error: 6.745933442443742e-05
gradient beta0 relative error: 3.2042016751691054e-05

Layer 1
gradient W1 relative error: 1.3002614292274095e-05
gradient b1 relative error: 7.581530889200218e-09
gradient gamma1 relative error: 3.2997157614408097e-06
gradient beta1 relative error: 3.6261996187091413e-06

Layer 2
gradient W2 relative error: 5.830062700931717e-06
gradient b2 relative error: 6.469638164345612e-06

This was calculated using forward difference and took about 8 minutes to calculate on my computer. That's why a decided that this was enough evidence that my analytically calculated gradients were correct and bug free.

3 3-layer network with and without batch normalization

The hyperparameters for both the models below were set as per instructions to: 3 layers with 50 and 50 hidden nodes in the first two layers respectively, $lambda = 0.005$, $eta\_min = 10^{-5}$, $eta\_max = 10^{-1}$, 2 cycles of training and $n\_s = 5 * 45000/n\_batch$. The best test and validation accuracy refers to the ensamble model, where a new model is considered at each time eta is equal to eta_min.

3.1 Without batch norm

Best test accuracy: 0.533700
Best valid accuracy: 0.537800

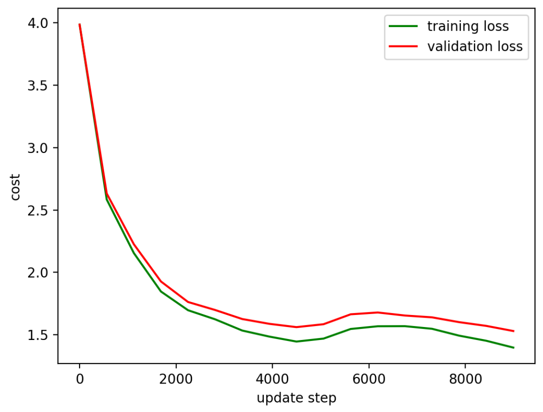

Figure 1: The graph of the training and validation cost for 2 cycles of training for the 3 layer model **without** batch normalization

3.2 With batch norm

Best test accuracy: 0.497100
Best valid accuracy: 0.504400

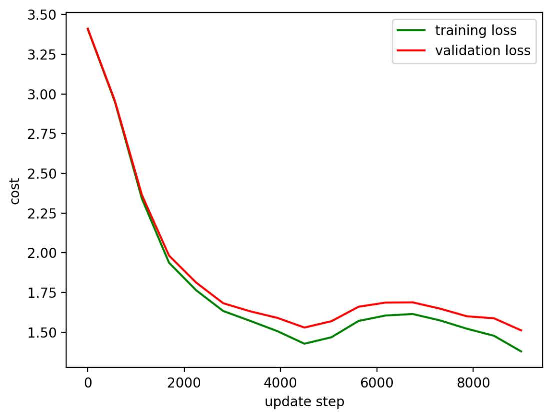

Figure 2: The graph of the training and validation cost for 2 cycles of training for the 3 layer model **with** batch normalization

4 9-layer network with and without batch normalization

The hyperparameters for both the models below were set as per instructions to: 9 layers with the number of hidden nodes 50, 30, 20, 20, 10, 10, 10, 10 in the first 8 layers respectively, $lambda = 0.005$, $eta\_min = 10^{-5}$, $eta\_max = 10^{-1}$, 2 cycles of training and $n\_s = 5 * 45000/n\_batch$. The best test and validation accuracy refers to the ensamble model, where a new model is considered at each time eta is equal to eta_min.

4.1 Without batch norm

Best test accuracy: 0.459200
Best valid accuracy: 0.463800

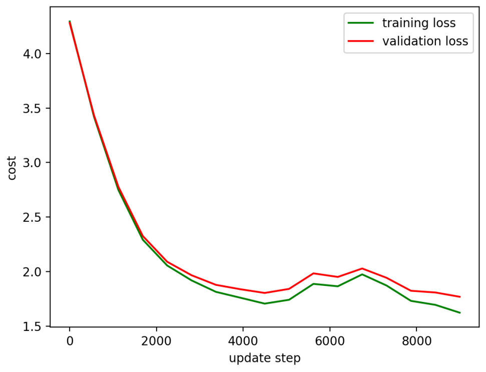

Figure 3: The graph of the training and validation cost for 2 cycles of training for the 9 layer model **without** batch normalization

4.2 With batch norm

Best test accuracy: 0.497700
Best valid accuracy: 0.491600

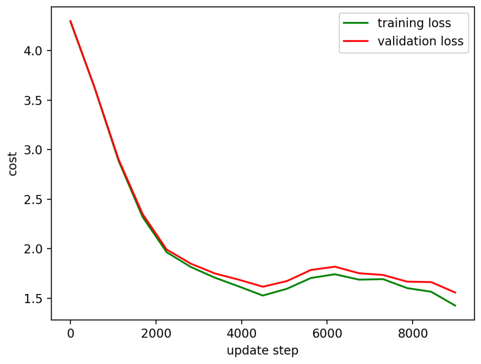

Figure 4: The graph of the training and validation cost for 2 cycles of training for the 9 layer model **with** batch normalization

5 Grid search for lambda

First I did a search with 10 different values of lambdas uniformly random between 10^{-1} and 10^{-5}. I Trained the models with all training data but 5000 that was used as validation data. I trained for 2 full cycles and recorded the best validation accuracy. The best accuracy i got was 50.17 percent for $lambda = 0.000028 = 2.8$. Then I did a search with 10 value between 10^{-3} and 10^{-4} and got the best result of 50.49 percent for $lambda = 0.000591$. Finally i did a search between $10^{-3.9}$ and $10^{-4.1}$ and got the best result 50.81 for $lambda = 0.000715$.

5.1 Final model

The best test accuracy achieved by the final model with $lambda = 0.000715$ was 50.37 percent. Figure 5 shows the evolution of the cost function for the final 3 layer model trained for 3 cycles.

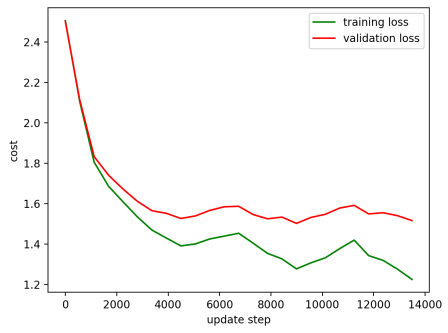

Figure 5: The graph of the training and validation cost for the final 3 layer model.

6 Sensitivity to initialization

6.1 Sigma = 1e-1

Accuracy **without**: 0.526000
Accuracy **with**: 0.516600

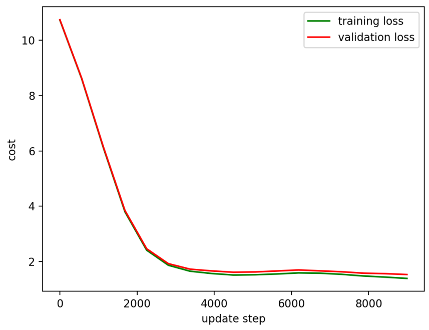

Figure 6: The graph of the training and validation cost **without** batch norm and sigma = 1e-1

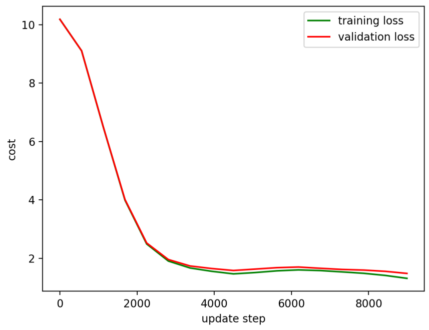

Figure 7: The graph of the training and validation cost **with** batch norm and sigma = 1e-1

6.2 Sigma = 1e-3

Accuracy **without**: 0.100000
Accuracy **with**: 0.506300

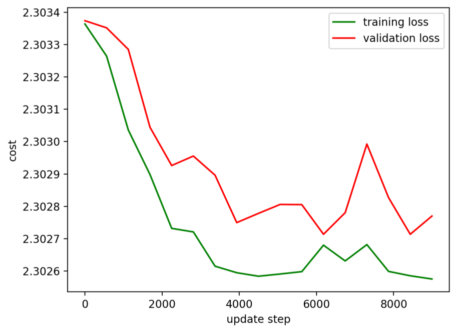

Figure 8: The graph of the training and validation cost **without** batch norm and sigma = 1e-3

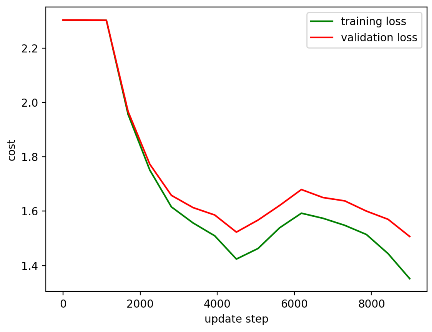

Figure 9: The graph of the training and validation cost **with** batch norm and sigma = 1e-3

6.3 Sigma = 1e-4

Accuracy **without**: 0.100000
Accuracy **with**: 0.480300

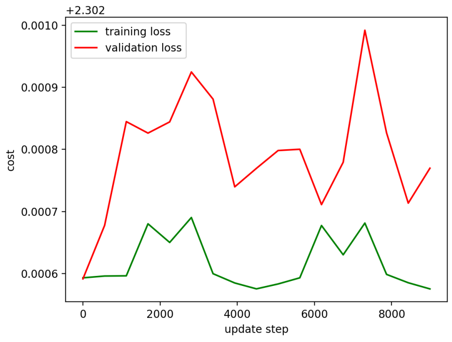

Figure 10: The graph of the training and validation cost **without** batch norm and sigma = 1e-4

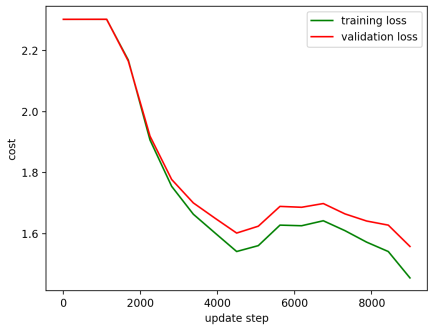

Figure 11: The graph of the training and validation cost **with** batch norm and sigma = 1e-4

6.4 Comments

As on can see when comparing Figure 8 with Figure 9, training with batch normalization seems to bee much more stable. With sigma set to 1e-3 for all layers the model without batch norm seems to struggle to reduce the cost. And when sigma = 1e-4 it fails completely as can be seen in Figure 10. Whereas the model with batch normalization seems to follow the same cost reduction pattern in Figure 11 as in Figure 9 no matter the initialization. Thus, the experiment seems to strengthen the fact that batch norm makes training more stable. As for the accuracy, one can see that the model without batch norm is as good as random, while the model with batch norm keeps hovering around 50 percent accuracy as before.