Assignment 2 Report

Adriana Ixba

DD2424 Deep Learning

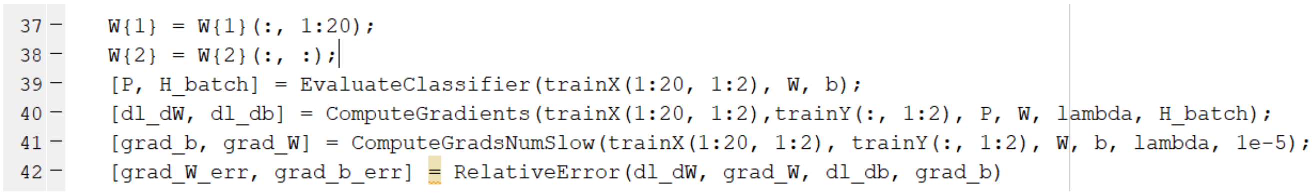
Josephine Sullivan

I. Implementation of Gradients

I believe that my implementation of gradients was correct. What I did to ensure its correctness, is calculate the Relative Error between the gradients generated from the provided, ComputeGradsNumSlow.m and the function I created, ComputeGradients.

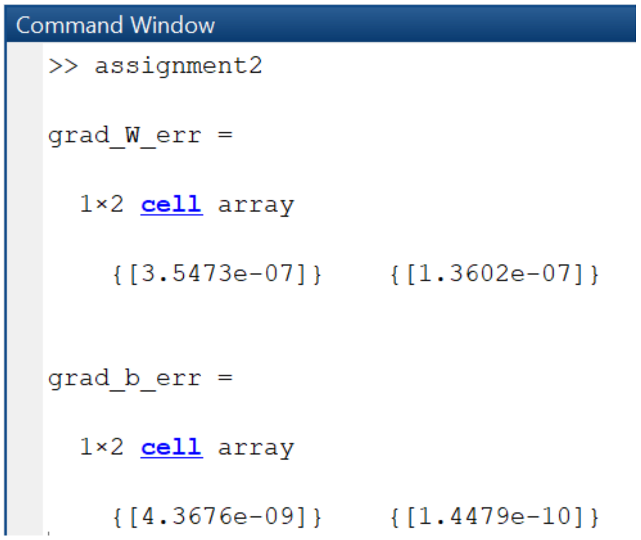
$$|g_a - g_n| \div max(eps, \ |g_a| \ + \ |g_n|) \text{, where the error must be } < 1e-6$$

This is the code that I ran:



```
W{1} = W{1}(:, 1:20);
W{2} = W{2}(:, :);
[P, H_batch] = EvaluateClassifier(trainX(1:20, 1:2), W, b);
[dl_dW, dl_db] = ComputeGradients(trainX(1:20, 1:2),trainY(:, 1:2), P, W, lambda, H_batch);
[grad_b, grad_W] = ComputeGradsNumSlow(trainX(1:20, 1:2), trainY(:, 1:2), W, b, lambda, 1e-5);
[grad_W_err, grad_b_err] = RelativeError(dl_dW, grad_W, dl_db, grad_b)
```

I was able to produce the following error results:



```
>> assignment2

grad_W_err =

  1×2 cell array

    {[3.5473e-07]}    {[1.3602e-07]}


grad_b_err =

  1×2 cell array

    {[4.3676e-09]}    {[1.4479e-10]}
```

This shows that for my W cell ({W1, W2}) and b cell ({b1, b2}), each component results in a less than 1e-6 error. This means that my computations for the gradients should be about right.

II. Experiments

Cyclical Learning Rates Curves (Figures 3 and 4 on Assignment Page)

Experiment 1

Now, we will train our classifier using cyclical learning rates with the following values/settings:

eta_min = 1e-5

eta_max = 1e-1

lambda = 0.01

n_s = 500

cycles = 1

With these settings we are able to achieve an accuracy of 46.05%. We can see the increase and change of this accuracy throughout the cycles in Figure 1.

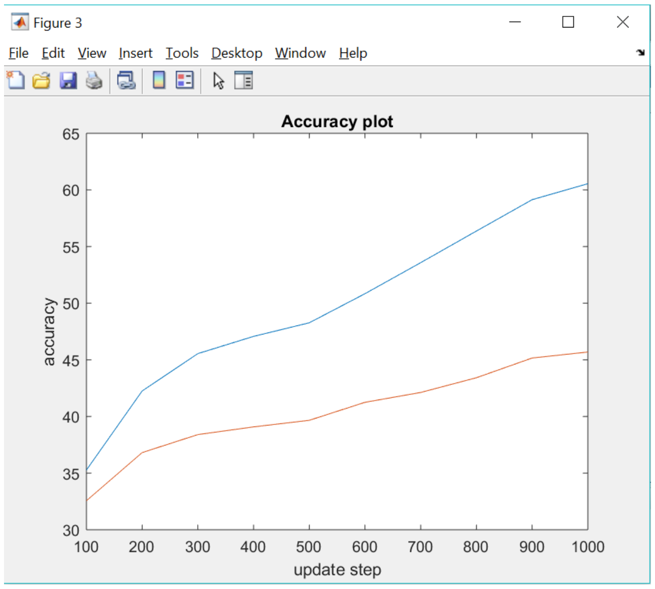


Figure 1. Accuracy when training using Cyclical Learning rates with an eta range between 1e-5 and 1e-1. With an n_s of 500, and a lambda value of 0.01. Where the training is represented by the blue, and the validation is represented by the red.

Finally, we also see the production of the following Loss and Costs plots:

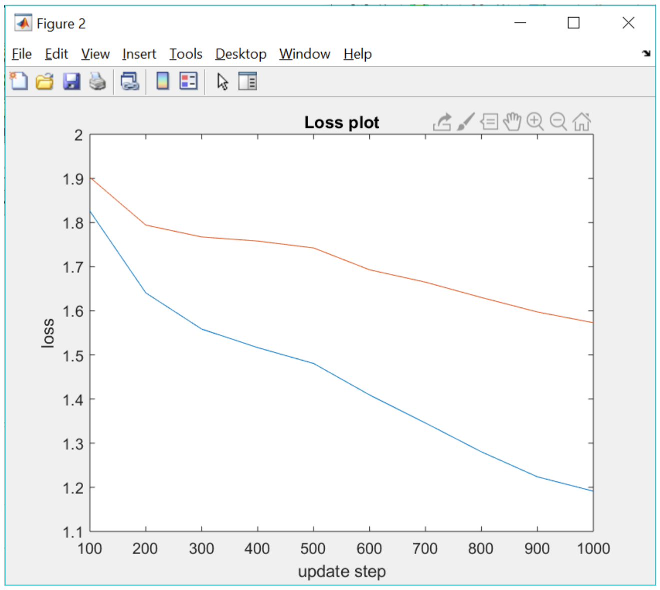


Figure 2. Losses when training using Cyclical Learning rates with an eta range between 1e-5 and 1e-1. With an n_s of 500, and a lambda value of 0.01. Where the training is represented by the blue, and the validation is represented by the red.

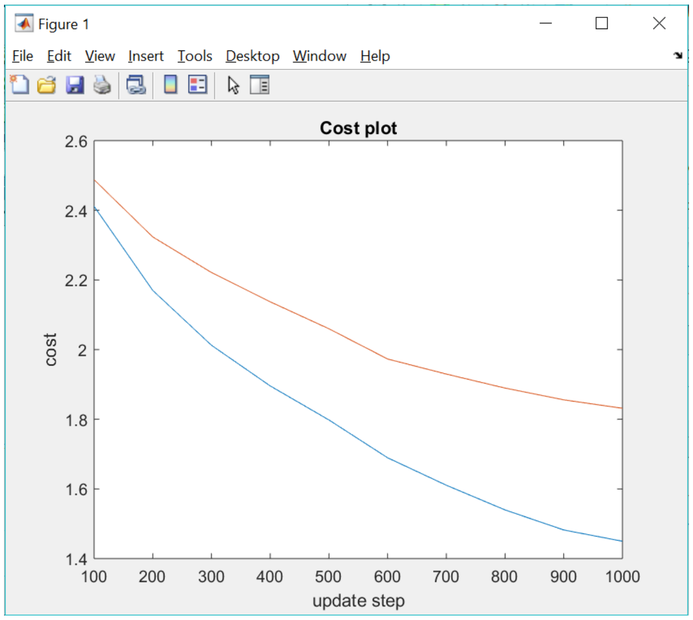


Figure 3. Costs when training using Cyclical Learning rates with an eta range between 1e-5 and 1e-1. With an n_s of 500, and a lambda value of 0.01. Where the training is represented by the blue, and the validation is represented by the red.

Experiment 2

Secondly, we will train our classifier using cyclical learning rates with the following values/settings:

eta_min = 1e-5

eta_max = 1e-1

lambda = 0.01

n_s = 800

cycles = 3

With these settings we are able to produce the following graphs:

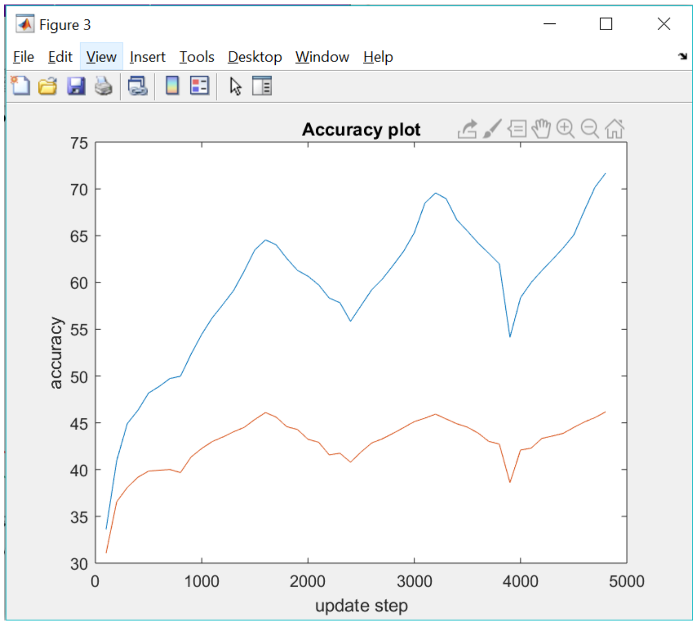


Figure 4. Accuracy when training using Cyclical Learning rates with an eta range between 1e-5 and 1e-1. With an n_s of 800, and a lambda value of 0.01. Where the training is represented by the blue, and the validation is represented by the red.

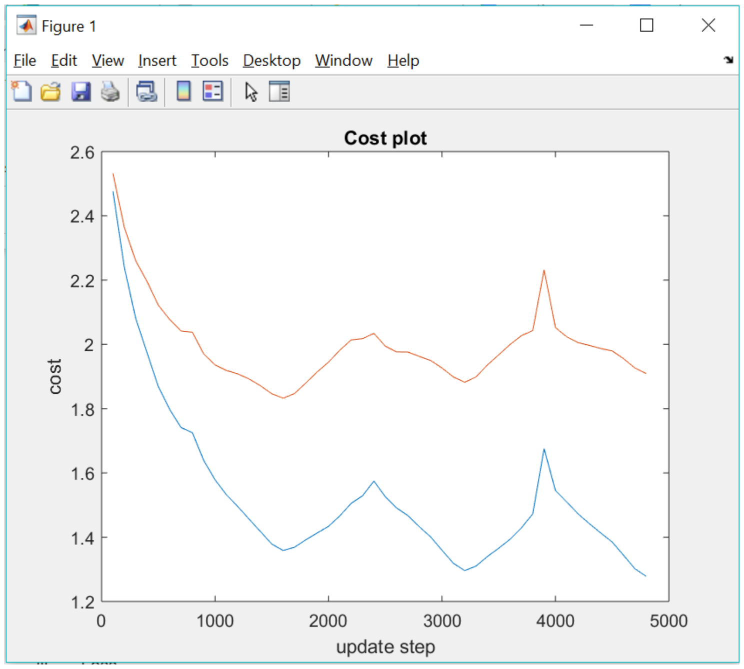


Figure 5. Costs when training using Cyclical Learning rates with an eta range between 1e-5 and 1e-1. With an n_s of 800, and a lambda value of 0.01. Where the training is represented by the blue, and the validation is represented by the red.

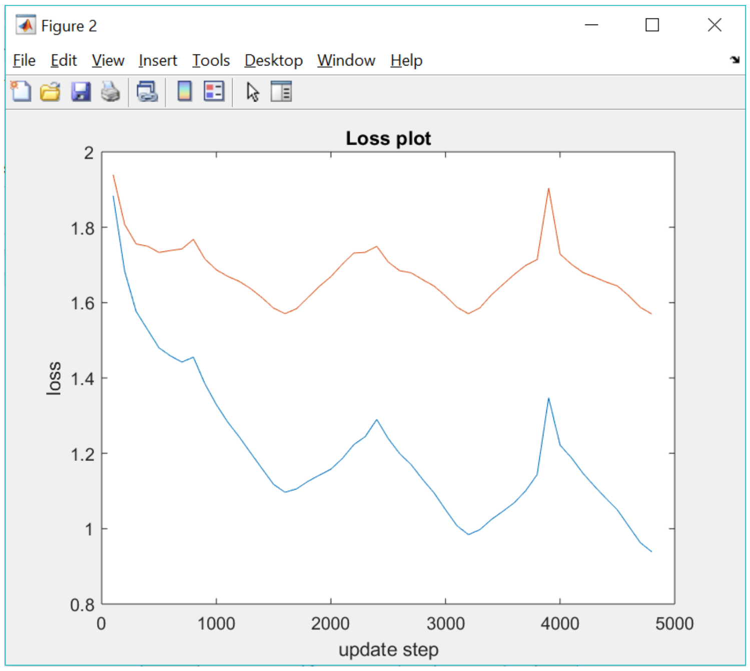


Figure 6. Losses when training using Cyclical Learning rates with an eta range between 1e-5 and 1e-1. With an n_s of 800, and a lambda value of 0.01. Where the training is represented by the blue, and the validation is represented by the red.

In these figures (4, 5, 6), we can see a difference in line compared to figures 1, 2, and 3. I believe the cause of this may be the constantly changing eta combined with the increase in stepsize (n_s) and cycles.

Finding Lambda: Coarse-Search

After training using Cyclical Learning rates, we are able to look for an optimal lambda that maximizes our accuracies in training and testing/validating. We do a "Coarse-Search" to do this using the following hyper-parameter settings:

eta_min = 1e-5

eta_max = 1e-1

cycles = 3

n_s = 800

I do this Coarse-Search for a total of 20 iterations. I then compiled the data, and saved the lambda, and its respective accuracy onto a graph.

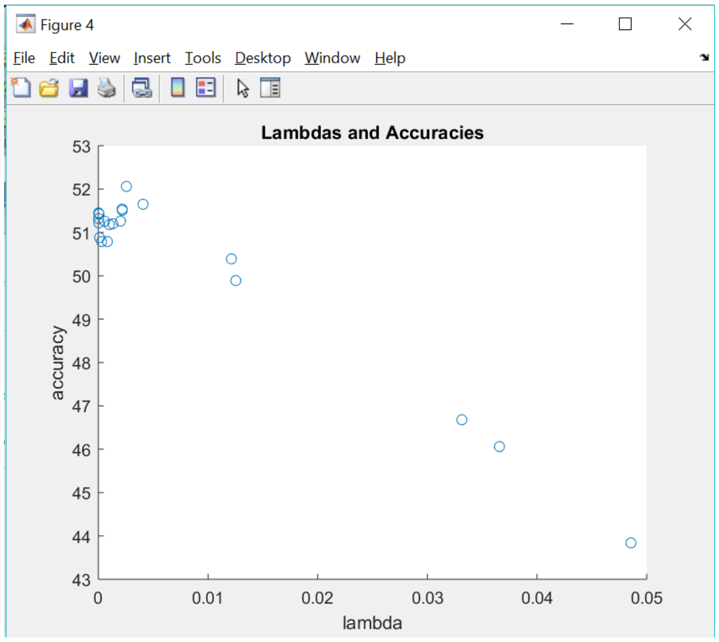


Figure 7. A graph plot where the X-axis represents the lambda that is tested in the Lambda Coarse-Search, and the Y-axis represents the respective accuracy computed from the network.

We save the top 3 accuracies:

Lambda	Accuracy
0.0025585	52.06%
0.0040738	51.65%
0.0021674	51.54%

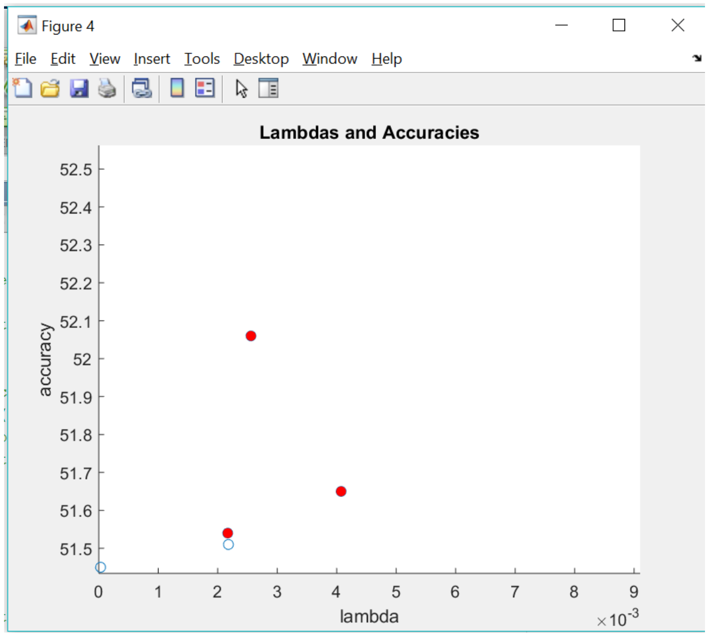


Figure 8. A graph plot showing the top 3 accuracies from Figure 7.

Finding Lambda: Fine-Search

After conducting a Coarse-Search, I then perform a Fine-Search, by constricting the range of the search and increasing both the cycles run. The following are the hyper-parameter settings:

eta_min = 1e-3

eta_max = 1e-1

cycles = 4

n_s = 800

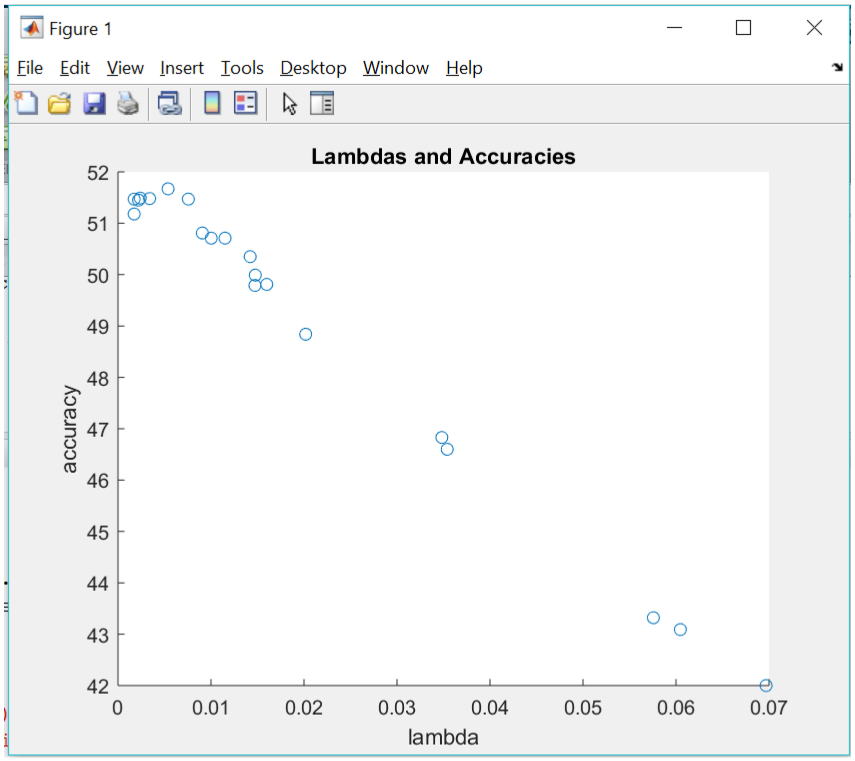


Figure 9. A graph plot where the X-axis represents the lambda that is tested in the Lambda Coarse-Search, and the Y-axis represents the respective accuracy computed from the network.

We save the top 3 accuracies:

Lambda	Accuracy
0.0053838	51.67%
0.0023901	51.49%
0.0033975	51.48%

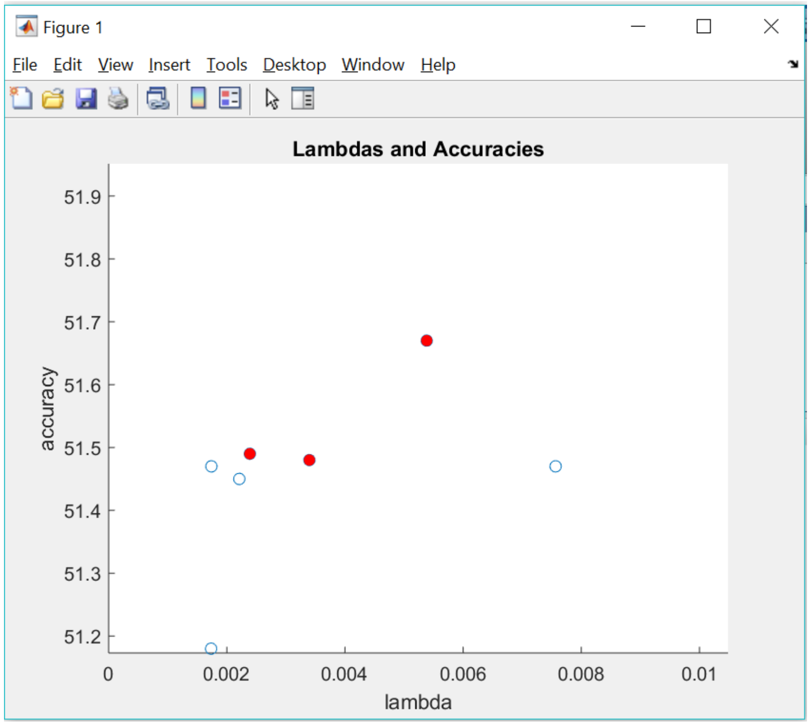


Figure 10. A graph plot showing the top 3 accuracies from Figure 9.

Best Lambda: Final Test

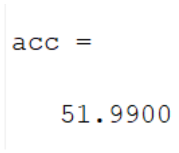
From the results in the previous section, we can see that the best found lambda is: 0.0025585. So now, we use this lambda to run our tests again. We train the network on all the training data, except for 1000 examples in a validation set for 3 cycles. With this lambda, I found that an n_s of 1250 heeds the best results for 3 cycles. From previous tests, I use an n_s of 800 and was only able to achieve a percentage of 51.53%.

So, I increased the n_s to 1250, and achieved an accuracy of 51.85%.

```
acc =

   51.8500
```

Then, with these same parameter settings, I wondered if it was possible to increase the accuracy with the number of cycles. So, I increased the number of cycles to 5, and was able to achieve the highest accuracy of 51.99%.



```
acc =

   51.9900
```

We were able to successfully find the hyper-parameters that give us an accuracy similar to that presented in the assignment page (~52%). I found that making sure to level off the correct stepsize, amount of data and learning rates are essential, as different settings can cause over/underfitting to data, thus reducing accuracies.