DD2424 Deep Learning in Data Science

Assignment 2 - Lab report, Valdemar Gezelius (vgez@kth.se)

March 9, 2021

1 About the lab

The lab focused on to what extent we can train a two layered network to be able to classify pictures. The data-set came in batches of `10000` data-points with `3072` features each, divided into `10` unique classes. A `softmax` classification using a mini-batch gradient descent model meant that for each data-point, the model makes a prediction stored in a `probability vector` and `argmax` of said vector is predicted to be the class of the data-point. For each mini-batch, the weights and biases are updated. A `coarse-to-fine search` for optimal hyperparameter values was conducted and a cyclic eta update was implemented.

2 Gradient examinations

For this lab the provided numerical gradient function was needed to be updated to be able to handle multiple layers, the value for `h` was set to `1e-5`. I then used the built-in Python library `Unittest` to compare my results against the numerically computed gradient. First comparing the two gradients directly using the built-in function `assertAlmostEqual(g_a, g_n)`, where I measure that the analytically computed gradient and the numerical does not differ down to the seventh decimal place. Secondly by implementing `Eq.1`, where the relative error between the numerical gradient `g_n` and the analytical gradient `g_a`, is computed (here `eps` is a very small positive number)

$$\frac{|g_a - g_n|}{max(eps, |g_a| + |g_n|)} \tag{1}$$

The function I then used was the built-in `assertLessEqual(g, num)`, that return `true iff g` is less than or equal to `num` (where `num` is a small number, for testing I set this to `1e-6`). Both tests were successful when, as stated in the lab, the first `20` features were selected.

When the tests seemed to work I performed the sanity check by trying to overfit the model on a small set (`100` samples) of the training data. I trained it for `200` epochs with λ=`0`, η=`0.003` and `batch_size=10`, which resulted in a training loss of ≈`0.0086`. This made me believe that my gradient calculations were correct and I could move on with the lab.

3 Introducing Cyclical Learning Rates

3.1 Replicating Figure 3

In `Figure 1` I replicate the conditions of `Figure 3` in the lab instructions. As the plots show the results does not differ significantly from the ones shown in the lab instructions, with minor

differences that could be explained by the random initialization of the weight layers and biases. The network yielded a test accuracy of ≈45.5%.

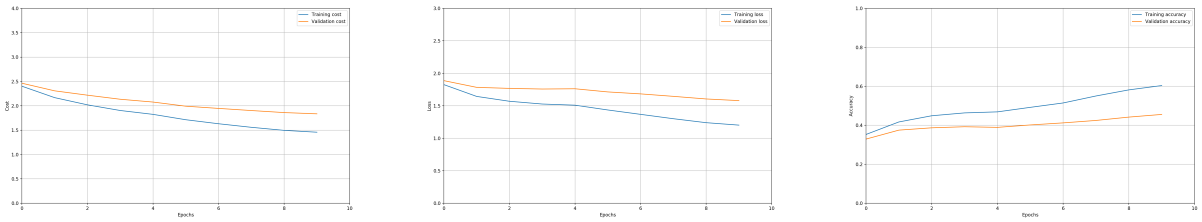

Figure 1: Replica of `Figure 3` from lab instructions, training- and validation plots. **Left:** cost plot. **Middle:** loss plot. **Right:** accuracy plot. One cycle (10 epochs) of training performed. Network settings: η_{min}=1e-5, η_{max}=1e-1, n_s=500, λ=0.01, batch_size=100. The network yielded a test accuracy of ≈45.5%.

3.2 Replicating Figure 4

In `Figure 2` I replicate the conditions of `Figure 4` in the lab instructions. As the plots show the results does not differ significantly from the ones shown in the lab instructions, with minor differences that can be explained, as stated before, by the random initialization of the weight layers and biases.

As we perform several cycles we can see more fluctuation in the graphs, this is what we expect when working with a cyclic η-value as we train the model and this help us from getting stuck in a local minima. The network yielded a test accuracy of ≈46.6%.

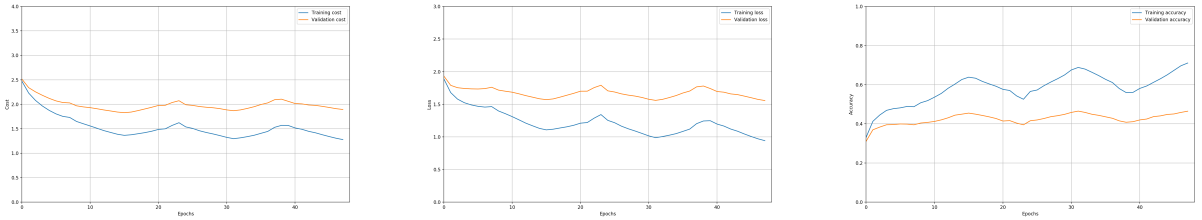

Figure 2: Replica of `Figure 4` from lab instructions, training- and validation plots. **Left:** cost plot. **Middle:** loss plot. **Right:** accuracy plot. Three cycle (48 epochs) of training performed. Network settings: η_{min}=1e-5, η_{max}=1e-1, n_s=800, λ=0.01, batch_size=100. The network yielded a test accuracy of ≈46.6%.

4 Hyperparameter optimization

A coarse-to-fine search for a optimized λ-value was conducted. Each network trained for 2 cycles with additional network settings (apart from λ-value): batch_size=100, n_s=900, η_{min}=1e-5, η_{max}=1e-1.

My approach was to run 20 iterations, training each new network with a different λ-value sampled uniformly from a set range of values. After 10 iterations I evaluated the results (compiled

Iteration	λ-value	Validation accuracy
3	**0.002548**	0.518504
5	**0.003113**	0.518304
6	0.002491	0.517704
8	0.004245	0.516903
9	0.001224	0.516703
2	0.003150	0.516103
4	0.004126	0.515703
7	0.003723	0.515103
10	0.001226	0.513503
1	0.004764	0.512102

Table 1: Coarse search for λ-value.

Iteration	λ-value	Validation accuracy
12	**0.003038**	0.533507
15	0.002787	0.528906
16	0.002607	0.528706
13	0.003066	0.527105
20	0.002996	0.526905
18	0.003102	0.525105
11	0.003084	0.524905
17	0.003087	0.523905
14	0.002671	0.523705
19	0.003020	0.521504

Table 2: Fine search for λ-value.

in Table 1) and updated the range for the λ-values. The start range for the λ-values was (l_max=0.005, l_min=0.001), and after the evaluation, the range was updated to (l_max=0.003113 +eps, l_min=0.002548 -eps). I decided to add/remove a small number (eps=0.0001) to prevent the range to be too narrow since I automated the search and there could be a case where the range became to narrow for the two networks that yielded the best validation accuracy. But as we see in Table 2 all the best performing networks had λ-values that were sampled from the refined search range. We can see that the network trained in the 12th iteration with λ=0.003038 yielded a validation accuracy of ≈53.4%, highest in the test and thus was chosen for the final network training.

5 Training best network

With the λ-value from the fine search a best network was trained. The preferences for the network and training was: batch_size=100, n_s=1200, η_{min}=1e-5, η_{max}=1e-1. The network was trained for 3 cycles and yielded a test accuracy of ≈52.4%. We see that the resulting accuracy is lower than the accuracy from the fine search, this could be explained by the initialization of the weights and biases and the fact that we use a larger (and different) set of data for testing the network. 49000 samples was used for training and 1000 samples for validation, testing was done on the test_batch.

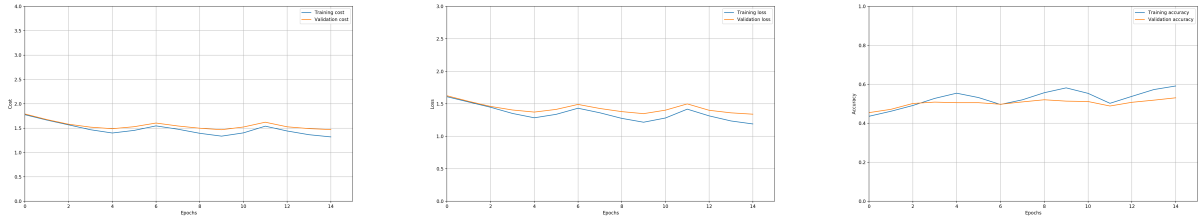

Figure 3: Best performing network, training- and validation plots. **Left:** cost plot. **Middle:** loss plot. **Right:** accuracy plot. Three cycle (`15` epochs) of training performed. Parameter settings: η_{min}`=1e-5,` η_{max}`=1e-1,` `n_s=1200,` λ`=0.003038,` `batch_size=100`. The network yielded a test accuracy of ≈`52.4%`.

6 Reflections

One key reflection is that the range used for the coarse search could be crucial to the results, so with a different starting range, a different (perhaps better) outcome could have been reached. Also, when the coarse-to-fine search was conducted, I fixated the weights and biases using a random seed to only have the λ`-value` differ between networks. This could of course lead to sub-optimal initialization of the weights and biases, but my thought was that the relation between different iterations would still hold. Another reflection is that when the best network is trained, the random initialization could also factor in on the results and for a truer result one would maybe have to average over a number of runs.