# Assignment 1 Bonus Points - DD2424

August Regnell 970712-9491

16 April, 2021

## Contents

**1 Exercise 2.1** **2**
1.1 Improvement 1 - Exhaustive random search for good values . . . 3
1.1.1 $n_s$ . . . 3
1.1.2 n_cycles . . . 3
1.1.3 $\lambda$ . . . 3
1.2 Improvement 2 - More hidden nodes . . . 5
1.3 Improvement 3 - Applying jitter . . . 6
1.4 Summary . . . 7

**2 Exercise 2.2** **8**
2.1 Subsequent tests . . . 10

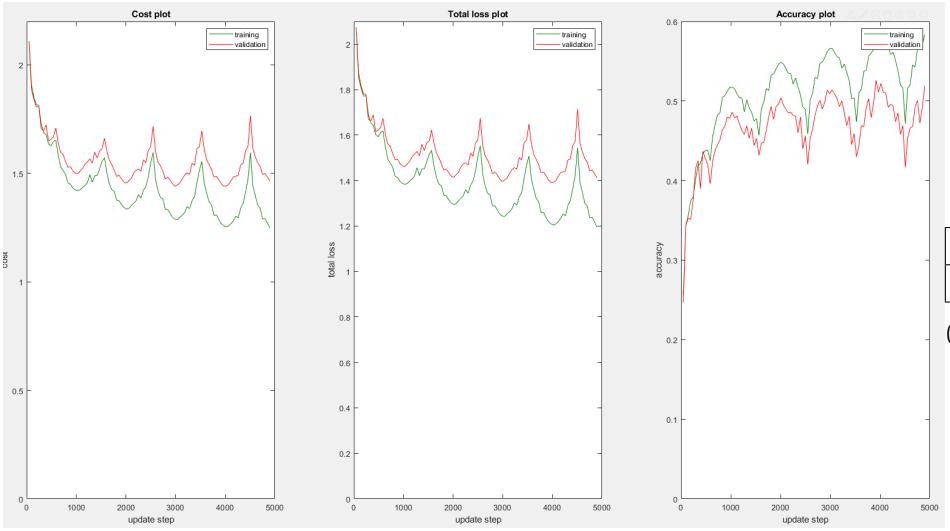
# 1 Exercise 2.1

I will use the best performing settings from the original assignment as a benchmark. That is,

| n_batch | $\eta_{min}$ | $\eta_{max}$ | n_s | n_cycles | $\lambda$ |
|---|---|---|---|---|---|
| 100 | 1e-5 | 1e-1 | 500 | 5 | 5.767e-04 |

Figure 1: Training parameters for the benchmark

which gave the following results




(a) Cost, total loss and accuracy plots of the final network.

| | |
|---|---|
| **Training accuracy** | 58.36% |
| **Test accuracy** | 51.10% |

(b) Accuracies of benchmark

## 1.1 Improvement 1 - Exhaustive random search for good values

### 1.1.1 $n_s$

First, an exhaustive search on $n_s$ was performed, of which the results can be seen in Figure 4. The following settings were used:

| n_batch | $\eta_{min}$ | $\eta_{max}$ | n_cycles | $\lambda$ |
|---|---|---|---|---|
| 100 | 1e-5 | 1e-1 | 3 | 5.767e-04 |

Figure 3: Training parameters for the exhaustive search for $n_s$

We notice a steady increase in the validation accuracy until the optimal values is reached.

| **Sampled $n_s$ and validation accuracy** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $n_s$ | 225 | 450 | 675 | 900 | 1125 | 1350 | **1575** | 1800 |
| Validation accuracy | 49.00% | 50.90% | 51.74% | 51.80% | 51.68% | 51.90% | **52.42%** | 51.40% |

Figure 4: Exhaustive search for a good $n_s$. The tested values are all multiples of half of the smallest $n_s$ which would make one cycle one full epoch. The highest validation accuracy and the corresponding $n_s$ is highlighted in bold text.

### 1.1.2 n_cycles

Second, an exhaustive search on `n_cycles` was performed, of which the results can be seen in Figure 6. The following settings were used:

| n_batch | $\eta_{min}$ | $\eta_{max}$ | n_s | $\lambda$ |
|---|---|---|---|---|
| 100 | 1e-5 | 1e-1 | 1575 | 5.767e-04 |

Figure 5: Training parameters for the exhaustive search for `n_cycles`

Surprisingly we notice that decreasing the number of cycles (compared to the previous search) to two yields the highest accuracy.

| **Sampled `n_cycles` and validation accuracy** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| `n_cycles` | 1 | **2** | 3 | 4 | 5 | 6 | 7 | 8 |
| Validation accuracy | 51.72% | **52.52%** | 52.42% | 52.18% | 51.58% | 51.58% | 51.02% | 51.22% |

Figure 6: Exhaustive search for a good `n_cycles`. The highest validation accuracy and the corresponding `n_cycles` is highlighted in bold text.

### 1.1.3 $\lambda$

So far we have not adjusted the regularization to the increased $n_s$ and changed `n_cycles`. We will thus perform another coarse search, but now with the optimal values obtained from the previous searches. That is:

| n_batch | $\eta_{min}$ | $\eta_{max}$ | n_s | n_cycles |
|---|---|---|---|---|
| 100 | 1e-5 | 1e-1 | 1575 | 2 |

Figure 7: Training parameters for the coarse search for $\lambda$

| **Search range** | | **Sampled $\lambda$ and validation accuracy** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $\lambda_{min} = -5$ | | 4.97e-02 | 6.77e-02 | 2.37e-02 | 7.60e-03 | 6.73e-02 | 2.20e-03 | 5.43e-03 | 8.06e-03 |
| $\lambda_{max} = -1$ | | 44.02% | 41.84% | 48.16% | 51.38% | 41.90% | 52.10% | 52.22% | 52.06% |
| $\lambda_{min} = -4.2$ | | 1.24e-03 | 1.38e-03 | **9.58e-04** | 6.43e-04 | **1.38e-03** | 4.17e-04 | 5.72e-04 | 6.56e-04 |
| $\lambda_{max} = -2.8$ | | 52.20% | 52.08% | **52.42%** | 51.58% | **52.78%** | 51.74% | 52.34% | 52.36% |
| $\lambda_{min} = -4$ | | 6.78e-04 | 7.27e-04 | 5.74e-04 | **4.44e-04** | 7.26e-04 | 3.36e-04 | 4.12e-04 | 4.50e-04 |
| $\lambda_{max} = -3.1$ | | 52.22% | 51.68% | 52.22% | **52.40%** | 51.92% | 51.58% | 51.70% | 52.38% |

Figure 8: Results of coarse search for a good $\lambda$. The three highest validation accuracies and their corresponding $\lambda$ is highlighted in bold text.

### 1.2 Improvement 2 - More hidden nodes

We shall now see if an increased number of hidden nodes, $m$, will yield an even better accuracy. Using the optimal values from the previous improvements,

| n_batch | $\eta_{min}$ | $\eta_{max}$ | n_s | n_cycles | $\lambda$ |
|---|---|---|---|---|---|
| 100 | 1e-5 | 1e-1 | 1575 | 2 | 1.3803717e-03 |

Figure 9: Training parameters for the coarse search for $\lambda$

which gave the results seen in Figure 10.

| $m$ | Test accuracy | $\Delta_{testaccuracy}$ |
|---|---|---|
| 50 | 51.87% | NA |
| 100 | 53.14% | +1.27% |
| 150 | 53.99% | +2.12% |
| **200** | **54.59%** | **+2.72%** |

Figure 10: Accuracies for different $m$ vs the benchmark ($m = 50$).

We clearly see that the accuracy increases when we increase the number of hidden nodes. However, increasing the number of hidden nodes should also result in a need of higher regularization. Thus, a coarse search for $m = 200$ was performed.

Note that tests with even higher $m$ could be performed, which probably would have given even better results, but these networks would take too long time to train.

| **Search range** | | **Sampled $\lambda$ and validation accuracy** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $\lambda_{min} = -5$ | | 6.48e-05 | 4.32e-04 | 4.83e-04 | **2.10e-03** | 9.42e-03 | **5.82e-03** | **4.62e-03** | 1.03e-05 |
| $\lambda_{max} = -1$ | | 55.08% | 55.72% | 55.20% | **55.80%** | 54.42% | **55.91%** | **55.92%** | 55.06% |
| $\lambda_{min} = -4$ | | 2.26e-04 | 5.19e-04 | 5.45e-04 | 1.03e-03 | 2.00e-03 | 5.92e-04 | 1.46e-03 | 1.01e-04 |
| $\lambda_{max} = -2.25$ | | 55.30% | 55.50% | 55.68% | 55.24% | 55.52% | 55.62% | 55.58% | 55.10% |

Figure 11: Results of coarse search for a good $\lambda$. The three highest validation accuracies and their corresponding $\lambda$ is highlighted in bold text.

The best $\lambda$ gave an accuracy of 54.68% on the test set.

### 1.3 Improvement 3 - Applying jitter

For this improvement the training data was augmented by applying a small normally distributed jitter. This was done by applying the jitter to each image in the mini-batch before doing the forward and backward pass. This was tested for different values of the jitter's standard deviation $\sigma$ and two different number of hidden nodes (with their hitherto optimal settings).

| | n_batch | $\eta_{min}$ | $\eta_{max}$ | n_s | n_cycles | $\lambda$ |
|---|---|---|---|---|---|---|
| $m = 50$ | 100 | 1e-5 | 1e-1 | 1575 | 2 | 1.3803717e-03 |
| $m = 200$ | 100 | 1e-5 | 1e-1 | 1575 | 2 | 4.6264704e-03 |

Figure 12: Training parameters for the coarse search for $\lambda$

The results can be seen in Figure 13.

| $\sigma$ | $m = 50$ | $m = 200$ |
|---|---|---|
| 0 | 51.87% | 54.68% |
| 0.001 | 51.81% | **54.70%** |
| 0.01 | 51.82% | 54.64% |
| 0.05 | **51.96%** | 54.67% |
| 0.1 | 51.94% | 54.66% |
| 0.2 | 51.94% | 54.48% |

Figure 13: Accuracies for different $\sigma$.

We notice that for $m = 200$ we find a tiny improvement, but is hard to say if it is significant.

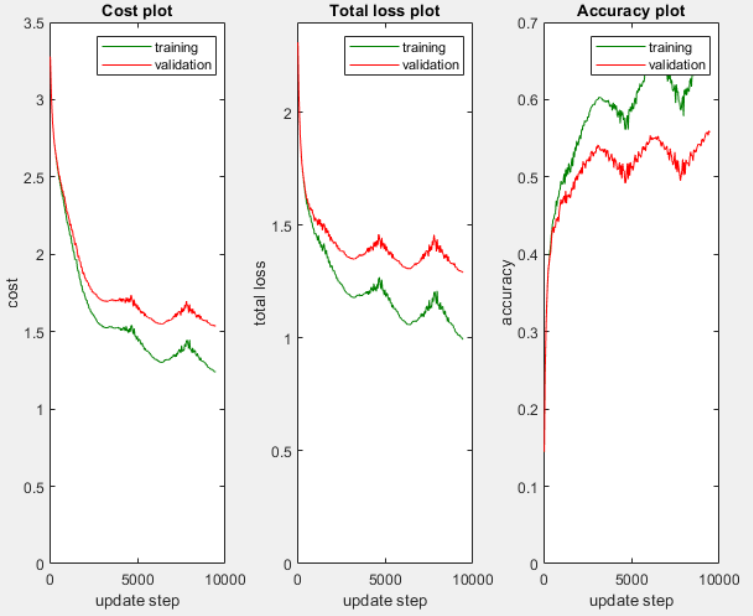
### 1.4 Summary

As a final test all previous improvements was combined with the best `eta_min` and `eta_min` found in the next section and running for one more cycle. Thus the following settings were used:

| m | n_batch | $\eta_{min}$ | $\eta_{max}$ | n_s | n_cycles | $\lambda$ | $\sigma$ (jitter) |
|---|---|---|---|---|---|---|---|
| 200 | 100 | 1e-5 | 5e-2 | 1575 | 3 | 4.6264704e-03 | 0.001 |

Figure 14: Training parameters for the final training run.

which gave the following results




(a) Cost, loss and validation accuracy for the final test.

| | Training | Test |
|---|---|---|
| Accuracy | 67.92% | 54.87% |
| $\Delta_{accuracy}$ | +9.56% | +3.77% |

(b) Accuracies for the final test compared to no improvements (Figure 2b).

We see that combining all the improvements from the previous section with the updated `eta_min` and `eta_min` from the next section we manage to improve the accuracy with 3.77%. However, from the graphs we notice that we probably would get a higher accuracy if we let the training run for more cycles.

## 2 Exercise 2.2

In this section we will explore the how to find good values for `eta_min` and `eta_min`, more specifically using the the methods proposed by Smith[1].

Smith proposes the following method: Run the network with `n_s` equal to `max_iter`, which is equivalent to running it for half a cycle. The learning rate will then increase linearly from the minimum to the maximum learning rate you are searching between. One should then choose the `eta_min` for which `eta` one first sees an increase in accuracy on the validation set. The `eta_max` should then be chosen when accuracy slows, becomes ragged, or starts to fall.

This was implemented by training a network for half a cycle with `eta_min = 0` and `eta_min = 2e-2` and recording the accuracy on the validation set for after every batch. The slope was of the accuracy was then calculated, and `eta_min` and `eta_min` were chosen where one first saw a significant increase and decrease respectively (slope $> 1$ and $< -3$ respectively).

For a first test all training data was used, with 5000 used for validation, and the following settings:

```
n_batch=100, eta_min=0, eta_max = 2e-2, n_s = 41, lambda=5.7667486e-04
```

The resulting graph can be seen in Figure 16, and the calculated parameters where `eta_min = 0.0112` and `eta_min = 9.7561e-04`.

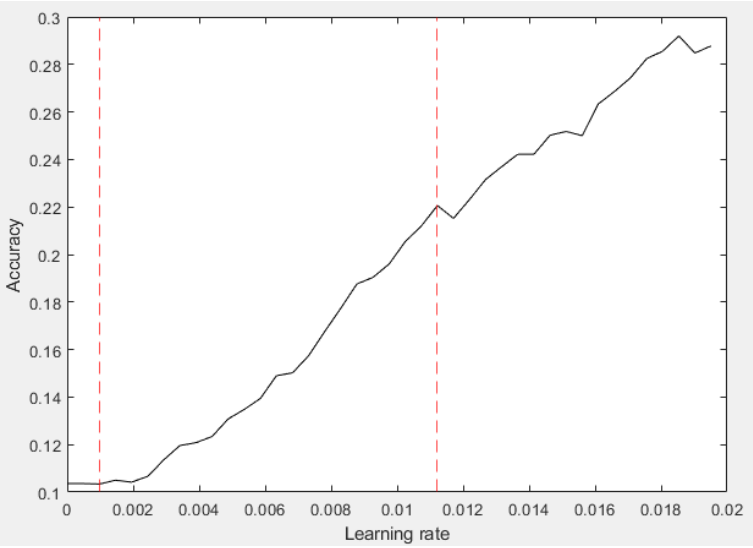



Figure 16: Classification accuracy on validation set as a function of increasing learning rate for 1 cycle

I also wanted to try a second method: training the network for a whole epoch for every value tested for `eta`. This should give a smoother curve. Using the same data but changing the settings to:

```
n_batch=100, eta_min=1e-5, eta_max = 1e-1, n_s = 500, n_epochs = 20
```

The resulting graph can be seen in Figure 17, and the calculated parameters where `eta_max = 0.04` and `eta_min = 1.00e-05`. The second method thus gave a much wider interval for `eta`, which is probably due to it running several times per tested `eta` (one full epoch), which would give a more stable outcome.

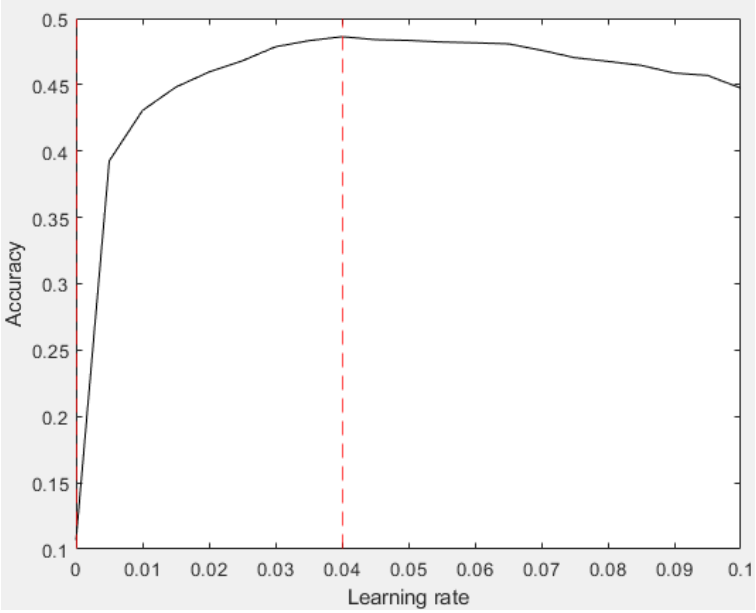


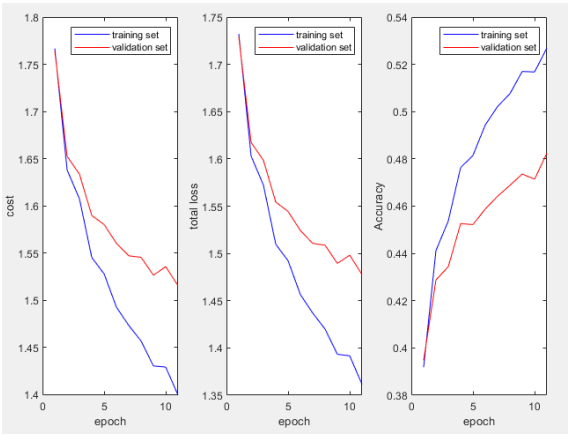

Figure 17: Classification accuracy on validation set as a function of increasing learning rate for 20 epochs

Comparing the methods on accuracy we get the following results when using the following settings:

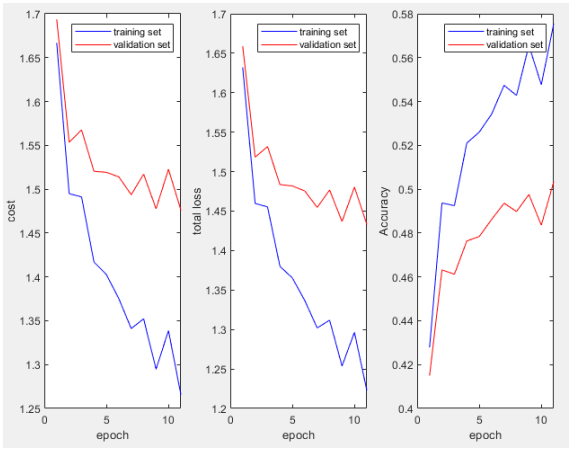
```
n_batch=100, n_s = 500, n_cycles = 5
```

| | Training accuracy | Test accuracy |
|---|---|---|
| Method 1 | 52.69% | 47.59% |
| Method 2 | 57.59% | 50.09% |
| $\Delta_{accuracy}$ | -4.90% | -2.50% |

Figure 18: Results of first test with method 1 and 2




(a) Method 1




(b) Method 2

Figure 19: Cost, loss and accuracy for the two methods.

As one sees in the results, the second method seems to give better results, which is why both methods will be used in the coming tests. One also notices that this difference probably can be mitigated by

running for more cycles since the first method seems to give a too small `eta_max`. However, doubling the number of cycles to ten only increases the test accuracy of the first method to 48.78% (+1.19%).

### 2.1 Subsequent tests

In this section the two methods were tested on several combinations of parameter settings. More specifically, the number of hidden nodes, $m$, and the regularization, $\lambda$, was varied. The results can be seen in Figure 20 and 21.

| | $m = 50$ | $m = 20$ | $m = 100$ |
|---|---|---|---|
| $\lambda = 5.77e-04$ | 2.4691e-04<br>0.0052<br>47.80% | 2.4691e-04<br>0.0077<br>44.28% | 4.9383e-04<br>0.0030<br>48.36% |
| $\lambda = 1.00e-03$ | 2.4691e-04<br>0.0052<br>47.87% | 2.4691e-04<br>0.0077<br>44.24% | 4.9383e-04<br>0.0030<br>48.24% |
| $\lambda = 1.00e-05$ | 2.4691e-04<br>0.0052<br>47.94% | 2.4691e-04<br>0.0077<br>44.16% | 4.9383e-04<br>0.0030<br>48.42% |

Figure 20: `eta_min`, `eta_max` and test accuracy for different parameter settings for **Method 1**.

| | $m = 50$ | $m = 20$ | $m = 100$ |
|---|---|---|---|
| $\lambda = 5.77e-04$ | 1.00e-05<br>0.0350<br>50.28% | 1.00e-05<br>0.0350<br>46.11% | 1.00e-05<br>0.0450<br>51.86% |
| $\lambda = 1.00e-03$ | 1.00e-05<br>0.0200<br>49.74% | 1.00e-05<br>0.0300<br>47.09% | 1.00e-05<br>0.0500<br>52.52% |
| $\lambda = 1.00e-05$ | 1.00e-05<br>0.0350<br>50.07% | 1.00e-05<br>0.0500<br>47.33% | 1.00e-05<br>0.0450<br>52.11% |

Figure 21: `eta_min`, `eta_max` and test accuracy for different parameter settings for **Method 2**.

Once again we see that Method 2 outperforms Method 1 in all cases. This indicates it may be the superior way to implement Smith's method for this two-layer network.

For the first method there seems to be a clear pattern of an increased accuracy with a larger $m$. We also see a pattern of `eta_max` increasing when $m$ decreases. However, there seems to be no clear pattern regarding the regularization.

Looking instead at the second method, we once again see the pattern of an increasing number of hidden nodes leading to a higher accuracy. However, the pattern of `eta_max` seems to be missing. It is in fact exchanged by a different, arguably as reasonable pattern, the best accuracy achieved by using a regularization proportional to the number of hidden nodes. This can be seen by observing that $m = 100$ performs best with the highest regularization, $m = 50$ the next highest and $m = 20$ the lowest.

## References

[1] Smith, L. N. (2015). Cyclical learning rates for training neural networks. arXiv:1506.01186 [cs.CV].