**Tianxiao Zhao**

**tzh@kth.se**

[redacted]

# Assignment 2

## (i) Gradient check:

Parameter settings: lambda = 0, h = 1e-06, eps = 0.001.

Here, I use the same error criteria in assignment 1, which is $\frac{|g_a - g_n|}{\max(eps, |g_a| + |g_n|)}$. The error performance is listed in the table below. Since the relative errors on all the parameters are very small number, I can assume that the analytical and numeric gradients are closely the same, with a minor difference. In this sense, I believe that my gradient computations are bug free.

Tab. 1: Relative errors on different parameters with different batch size

| Batch size | 1 | 10 | 50 | 100 |
|---|---|---|---|---|
| $\varepsilon_{W1}$ | 5.8840e-06 | 4.7699e-06 | 5.0601e-06 | 5.8920e-06 |
| $\varepsilon_{W2}$ | 5.4142e-07 | 4.5548e-07 | 5.1133e-07 | 4.9726e-07 |
| $\varepsilon_{b1}$ | 1.0765e-06 | 1.3014e-06 | 1.3502e-06 | 1.6920e-06 |
| $\varepsilon_{b2}$ | 2.7774e-09 | 3.6746e-09 | 6.0315e-09 | 3.3970e-09 |

## (ii) Momentum term

In this case, momentum term is introduced to accelerate convergence by dampening the oscillations of the hyper-parameters and loss function. Here, I train the network using the same parameter settings except for different rho: rho = 0.9 indicates the case with momentum term and rho = 0 means no momentum term is applied. The result is shown below. It appears that momentum term could cause a more significant decay of the loss plot and fewer epochs are needed for convergence.

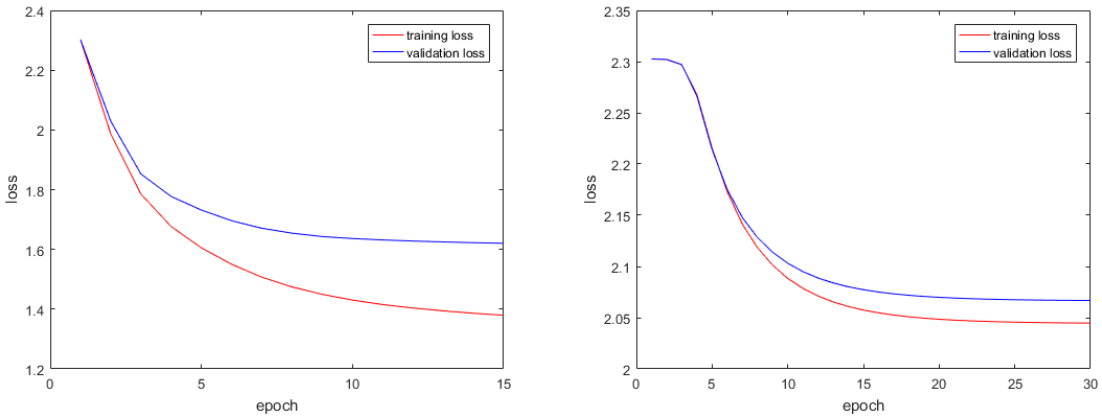


Fig. 1: Loss changes as epoch increases (with and without momentum term)

## (iii) Coarse search

The coarse search range for eta is from 0.01 to 0.3. The coarse search range for lambda is from 1e-7 to 0.1. Number of epochs for training during searching is set to 10. 50 different pairings for lambda and eta are generated and tried out. The plot below shows the relation of the validation accuracy with respect to lambda and eta. Decay rate is set to 0.8 and rho is set to 0.9.

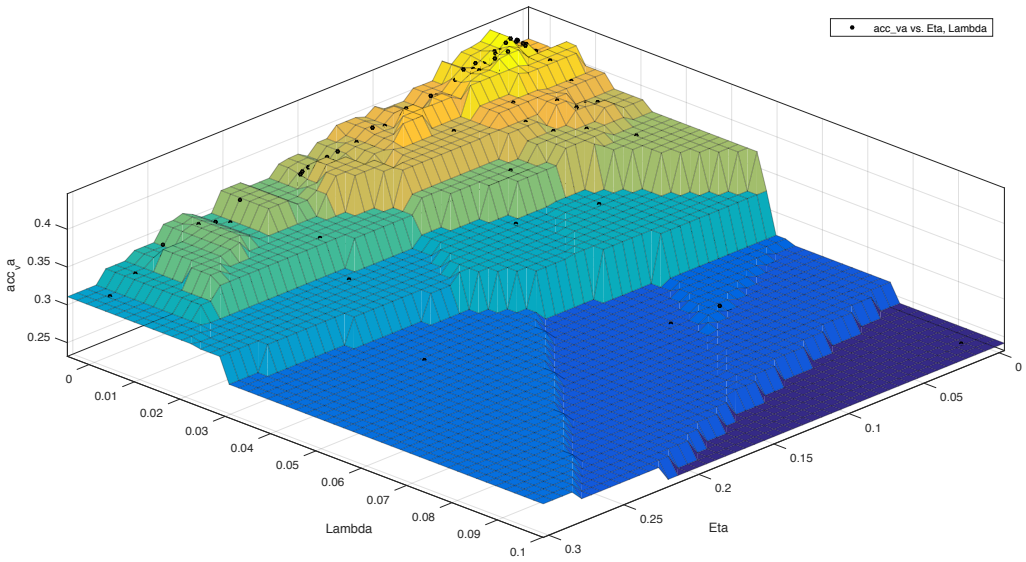


Fig. 2: The relation between validation accuracy and different parameter settings (lambda and eta) for coarse search. Fitting done with nearest neighbor.

From Fig. 2, we can pick out the hyper-parameter settings for the 3 best networks:

1) eta = 0.02279, lambda = 2.768e-07
   validation accuracy = 43.73%
2) eta = 0.03308, lambda = 2.831e-03
   validation accuracy = 43.50%
3) eta = 0.02634, lambda = 1.352e-04
   validation accuracy = 43.49%

## (iv) Fine search

From Fig. 2, we can narrow down the search range and start fine searching. The fine search range for eta is from 0.01 to 0.075. The fine search range for lambda is from 1e-7 to 1e-4. Number of epochs for training during searching is set to 12. 50 different pairings for lambda and eta are generated and tried out. The plot below shows the relation of the validation accuracy with respect to lambda and eta. Decay rate is set to 0.8 and rho is set to 0.9.

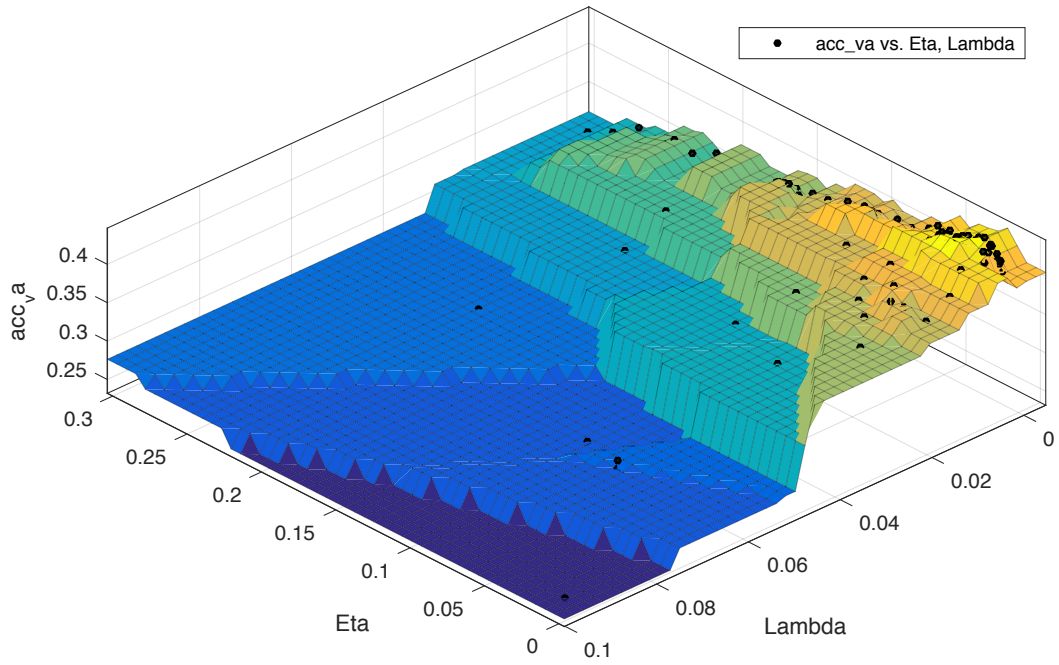


Fig. 3: The relation between validation accuracy and different parameter settings (lambda and eta) for fine search. Fitting done with nearest neighbor.

The hyper-parameter settings for the 3 best networks:

1) eta = 0.0622, lambda = 1.6424e-06
   test accuracy = 45.61%

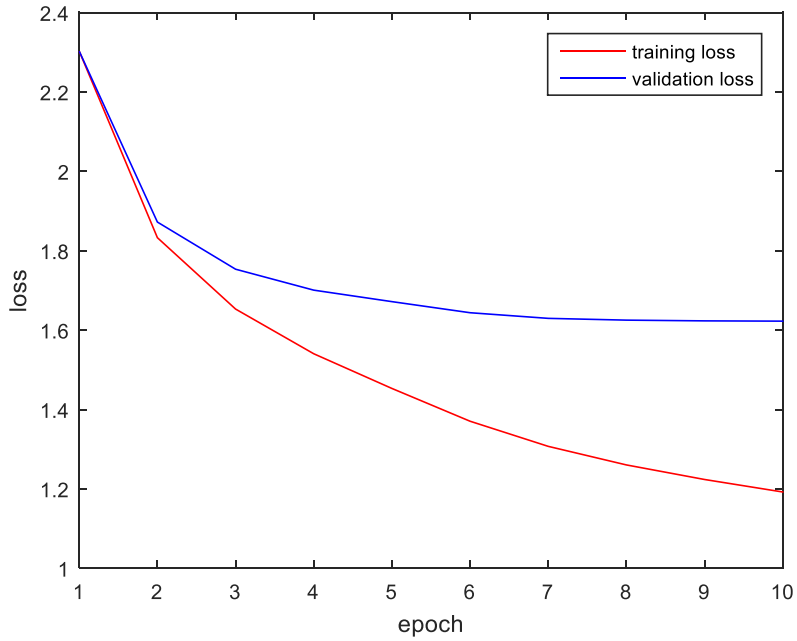


Fig. 4: Training and validation loss with respect to epochs

2) eta = 0.0516, lambda = 5.2165e-05
   test accuracy = 44.55%

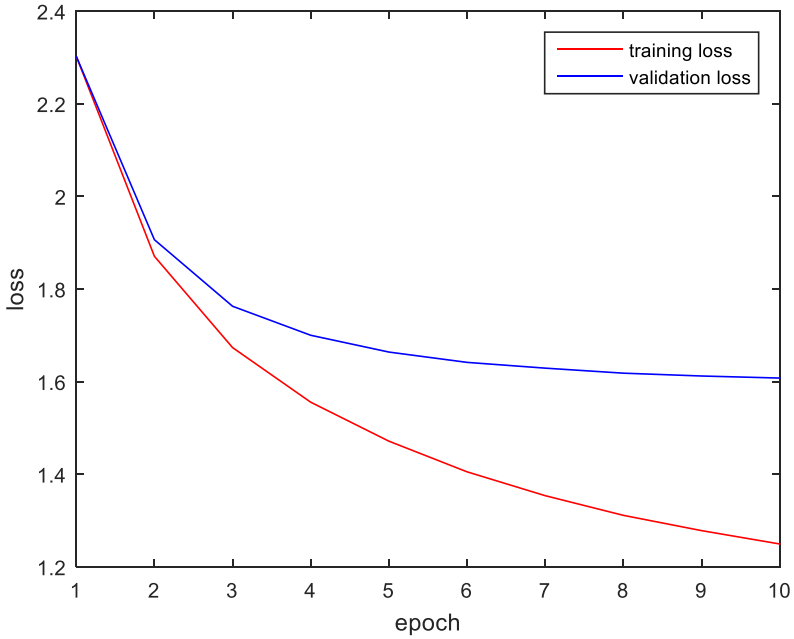

Fig. 5: Training and validation loss with respect to epochs

3) eta = 0.0306, lambda = 8.2136e-05
test accuracy = 44.04%

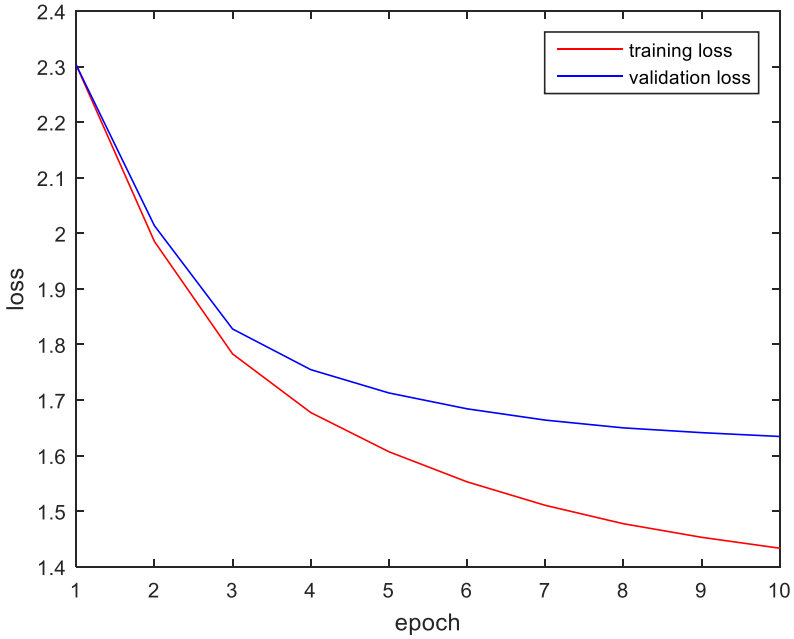

Fig. 6: Training and validation loss with respect to epochs

## (v) Best results

Here, I use the best hyper-parameters: eta = 0.0622, lambda = 1.6424e-06, n_batchs = 100, m = 50, n_epochs = 30, decay_rate = 0.8, rho = 0.9. After training the networks, I evaluate it using the test data and obtain a 45.23% test accuracy.

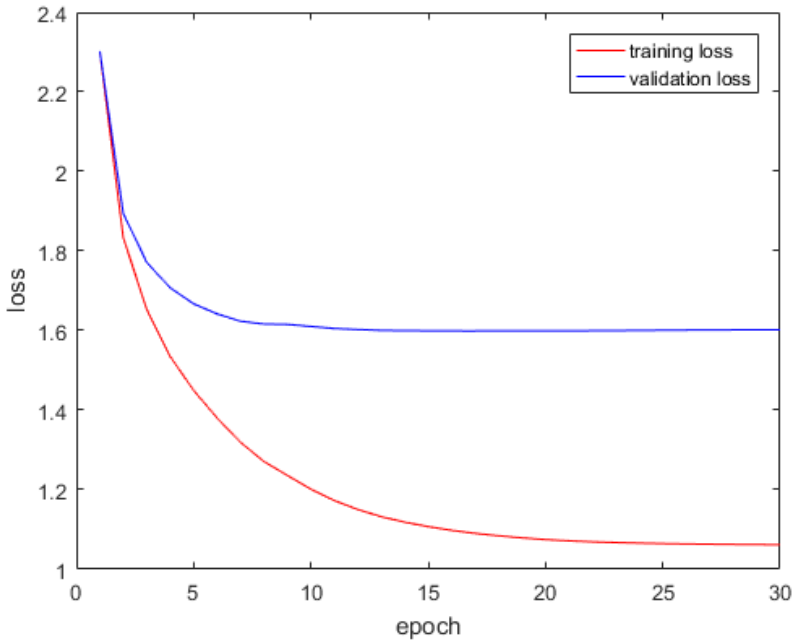


Fig. 7: Training and validation loss with respect to epochs