Tianxiao Zhao

tzh@kth.se

[redacted]

Assignment 3

(i) Gradient Check

1) 2-layer network

First, we check the gradient of 2-layer networks using the error criteria: $\frac{\max|g_a \quad g_n|}{\max(eps, |g_a|+|g_n|)}$. The number of hidden nodes is set to 50, and delta is set to 1e-6, which is used for the numerical gradient calculation. Relative errors between analytical and numerical gradients with different batch size are shown below:

Tab. 1: Relative errors with respect to different parameters

Batch size	1	5	10	20
ε_W	1.4874e-05	1.6918e-05	1.9358e-05	1.7643e-05
ε_{W_2}	1.013e-11	1.2482e-11	1.0166e-11	1.1613e-11
ε_b	1.5179e-16	2.3852e-16	2.3852e-16	3.0358e-16
ε_{b_2}	4.2562e-10	5.4062e-10	2.4751e-10	2.3441e-10

2) 3-layer network

Afterwards, we will check the gradient of 3-layer networks with the same process mentioned above. The numbers of hidden nodes are 50 and 30 respectively. Other parameters are kept the same. Results are shown below:

Tab. 2: Relative errors with respect to different parameters

Batch size	1	5	10	20
ε_W	6.4849e-09	1.0253e-09	1.5993e-07	3.2125e-08
ε_{W_2}	1.8686e-07	1.0505e-07	1.597e-07	1.7859e-07
ε_{W_3}	1.8554e-07	2.0321e-07	1.5807e-07	1.7675e-07
ε_b	0	0	0	0
ε_{b_2}	0	0	0	0

ε_{b_3}	3.907e-11	3.907e-11	3.907e-11	3.907e-11

Since the relative errors above are very small, I believe that my analytical gradients are bug-free.

(ii) Loss Function Plots

In this section, the performances of 3-layer networks with and without batch normalization are briefly investigated (eta = 0.3423, lambda = 7.4862e-06). Parameter settings are kept the same for both cases. It seems quite hard to train a 3-layer network without batch normalization, and the test accuracy is not so satisfactory. And with batch normalization, the loss function will decay more quickly and smoothly.

1) Without batch normalization

Training accuracy: 45.78%; test accuracy: 31.66%.

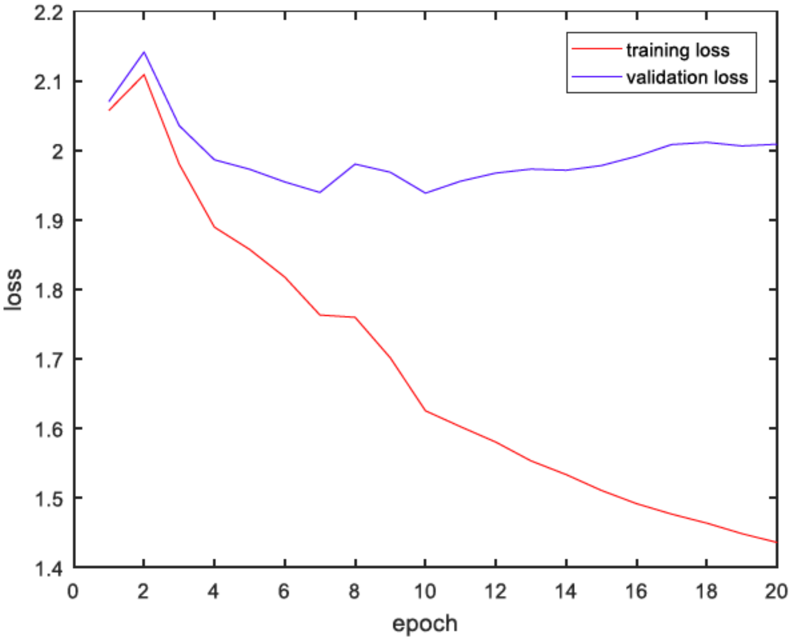

Fig. 1: Evolution of loss function as epoch increases

2) With batch normalization

Training accuracy: 63.52%; test accuracy: 45.31%.

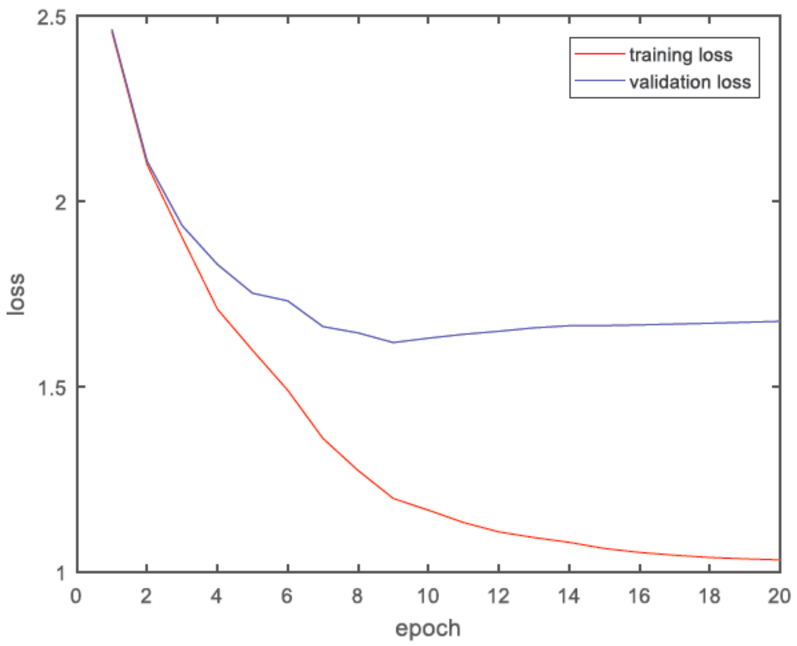

Fig. 2: Evolution of loss function as epoch increases

(iii) Fine Search

Before fine search, a round of coarse search is conducted. The coarse search range of eta is from 0.1 to 1, and of lambda is from 1e-7 to 0.1. The 3D plot of validation accuracy with respect to eta and lambda is shown below.

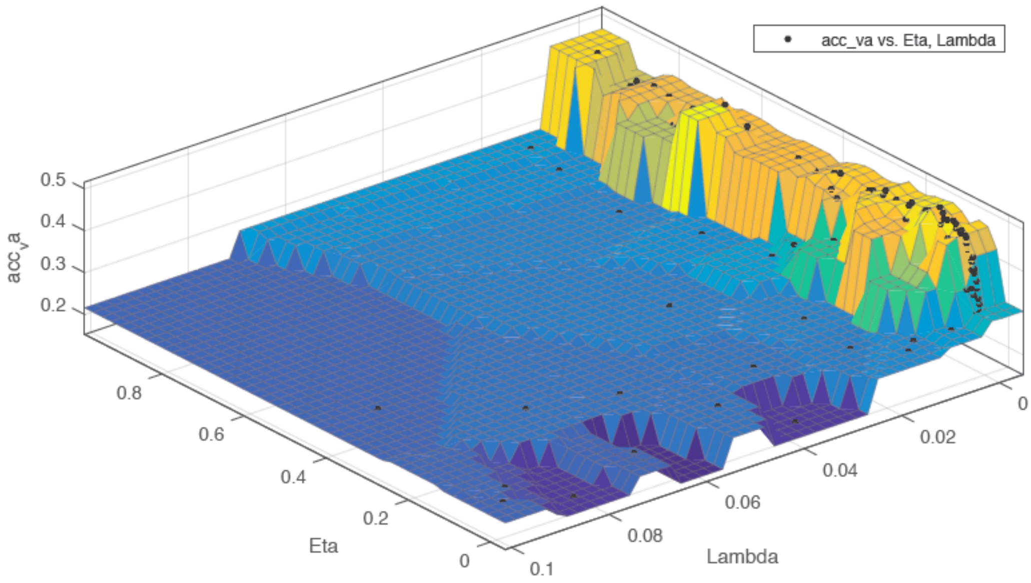

Fig. 3: The relation between validation accuracy and different parameter settings (lambda and eta) for coarse search. Fitting done with nearest neighbor.

Based on the results above, the search range of parameters can be further narrowed down. The fine search range of eta is from 0.12 to 0.9, and of lambda is from 1e-7 to 1.8e-3. The network is trained for 20 epochs, and 100 pairs of parameters are tried out. Also, batch normalization is used in this case. The decay rate is set to 0.8.

I use the data in data_batch_1.mat as training data and pick out hyper-parameter settings based on how the network performs over the validation data (data_batch_2.mat). Then, the trained network is further evaluated using the test data. The hyper-parameter settings for the top-3 best performing 3-layer networks are:

1) eta = 0.3423, lambda = 7.4862e-06 (accuracy over validation data: 45.38%).

   Training accuracy: 63.52%; test accuracy: 45.31%.

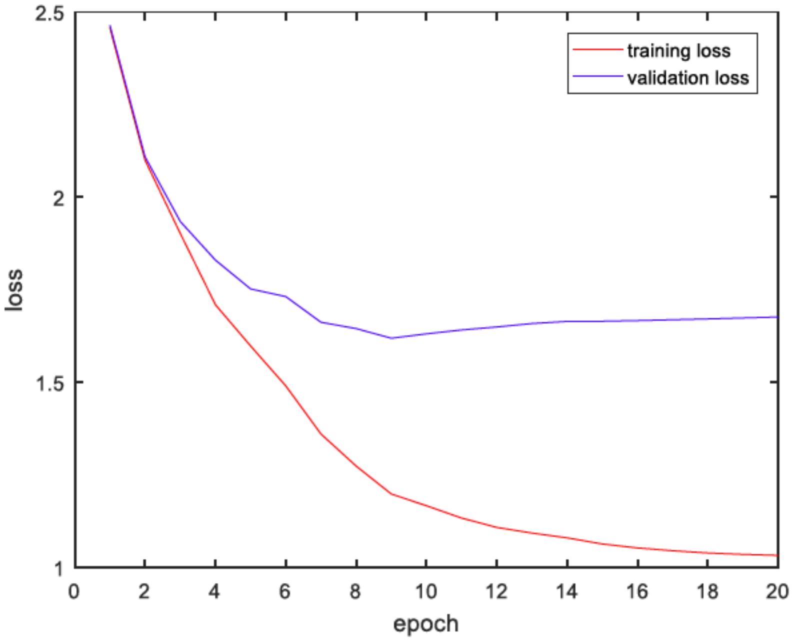

Fig. 4: Evolution of loss function as epoch increases (fine search)

2) eta = 0.4138, lambda = 2.8193e-05 (accuracy over validation data: 45.65%).

   Training accuracy: 66.04%; test accuracy: 45.24%.

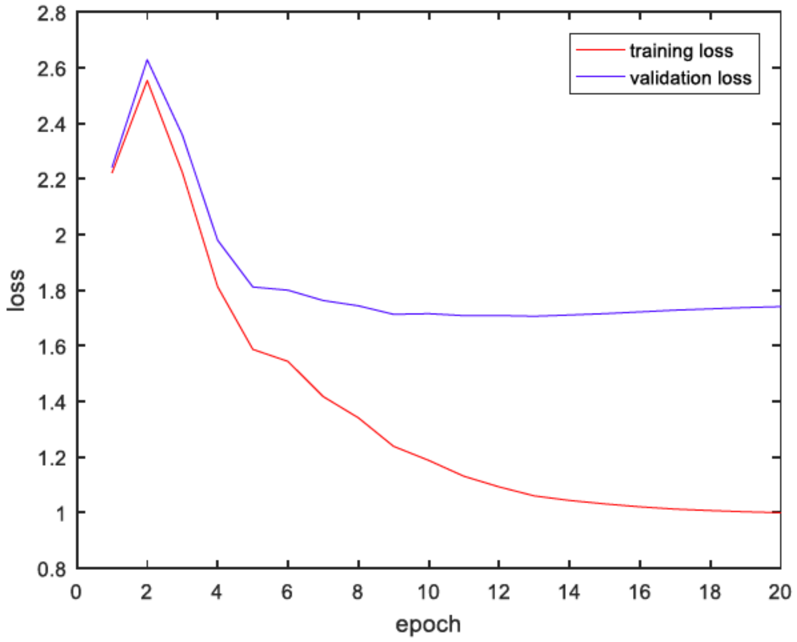

Fig. 5: Evolution of loss function as epoch increases (fine search)

3) eta = 0.4380, lambda = 1.0907e-04 (accuracy over validation data: 45.55%).

Training accuracy: 65.75%; test accuracy: 45.04%.

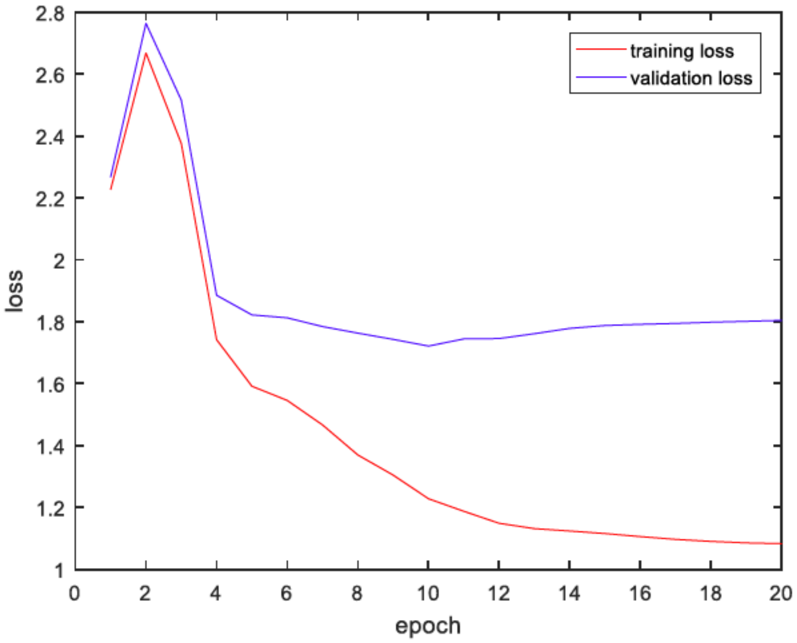

Fig. 6: Evolution of loss function as epoch increases (fine search)

(iv) Batch Normalization

To investigate the effect of batch normalization in depth, three groups of experiments with different learning rates (small, medium, high) are conducted. The 2-layer networks are trained for 10 epochs, and the only difference is whether batch normalization is applied during the training process.

Based on the results below, it turns out that batch normalization could improve the performance of the 2-layer network and reduce the strong dependence on initialization. When the learning rate is set too high, the training and validation loss will overshoot at once and then decay to the level of convergence when batch normalization is turned off. Compared to this, the loss will change more smoothly and no significant overshoot will be observed when batch normalization is used. And when the learning rate is very small, batch normalization accelerates the loss's convergence and makes it drop to a lower value.

Tab. 3: Performances with and without batch normalization

networks		Training accuracy	Test accuracy
eta = 0.3	No BN	59.12%	39.36%
	With BN	63.01%	44.89%
eta = 0.1	No BN	62.31%	43.98%
	With BN	56.26%	44.99%
eta = 0.01	No BN	29.64%	29.32%
	With BN	34.69%	33.83%

1) eta = 0.3

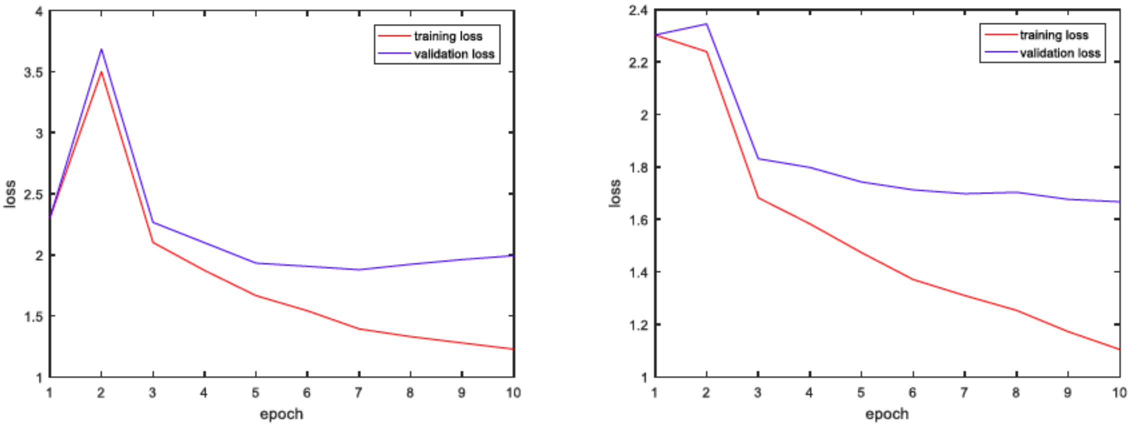

Fig. 7: Evolution of loss function
(Left: without batch normalization; right: with batch normalization)

2) eta = 0.1

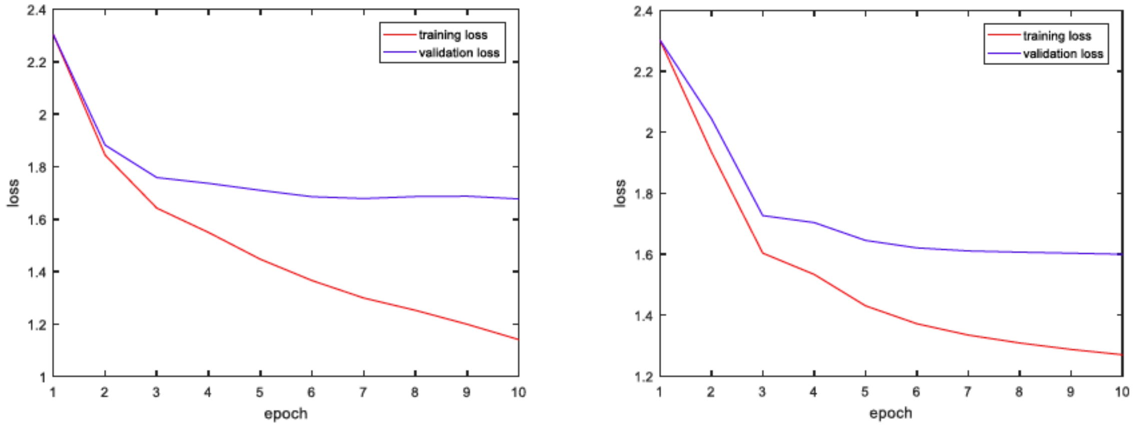

Fig. 8: Evolution of loss function
(Left: without batch normalization; right: with batch normalization)

3) eta = 0.01

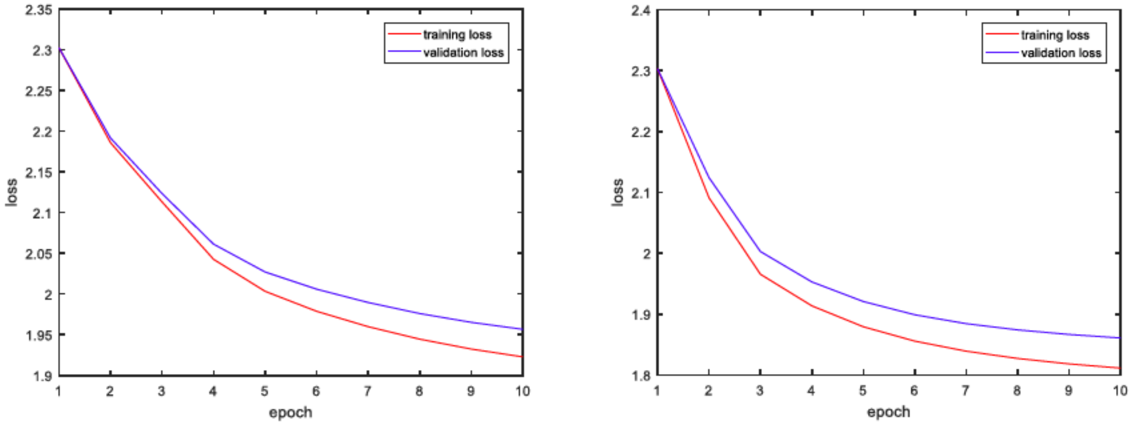

Fig. 9: Evolution of loss function
(Left: without batch normalization; right: with batch normalization)