KTH Royal Institute of Technology

DD2424 Deep Learning in Data Science

Assignment 3 Report

k layer network with multiple outputs basic part

Written by

Yutong Jiang

yutongj@kth.se

Date: Mar 29, 2022

1 introduction

In this assignment, I have trained and test k-layer networks. Besides, batch normalization is also implemented in this assignment both for training and testing process.

2 Analytic gradient computation check

In order to test whether the gradient is calculated correctly or not, the gradient is compared to the numerical results provided by ComputeGradsNumSlow. In this part, h in ComputeGradsNumSlow.m is set to 1e-5. lambda is set to 0.00358.

The relative error could be computed as the absolute differences.

First, I check the gradients for networks without batch normalization, which indicates that HyperParams.use-bn should be set to 0. The initial dimension of input data has been reduced to 10 as described in instruction for a faster speed.

For a two-layer network with m = 50, the relative error between analytic gradient computation and numerical gradient computation is 9.5696e-11. For a three-layer network with m = [50, 30], the relative error between analytic gradient computation and numerical gradient computation is 8.6688e-11. For a four-layer network with m = [50 30 20 10], the relative error between analytic gradient computation and numerical gradient computation is 9.6444e-11.

Next, I check the gradients for networks with batch normalization, which indicates that the HyperParams.use-bn should be set to 1. The initial dimension of input data has been reduced to 10 as described in instruction for a faster speed.

For a two-layer network with m = 50, the relative error between analytic gradient computation and numerical gradient computation is 8.8747e-11. For a three-layer network with m = [50, 30], the relative error between analytic gradient computation and numerical gradient computation is 9.9645e-11. For a four-layer network with m = [50 30 20], the relative error between analytic gradient computation and numerical gradient computation is 1.4702e-10.

Based on the above numerical proof, I could draw the conclusion that gradient computation is bug free for k-layer network with batch normalization.

3 Evaluation of loss function for three-layer network

In this part, I have set the parameter m = [50 30], alpha = 0.9, lambda = 0.005, n_{batch} = 100, the size of training dataset is 45000 and the size of validation dataset is 5000, n_s equals to $\frac{5*45000}{n_{batch}}$,

eta_{min} = 1e-5 and $eta_{max} = 1e - 1$.

The loss and cost curve of 3 layer network without batch normalization is shown below

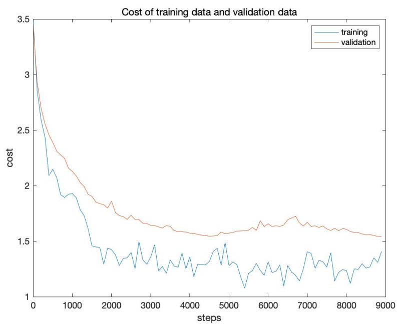

Figure 1: Cost of 3 layer network without batch normalization

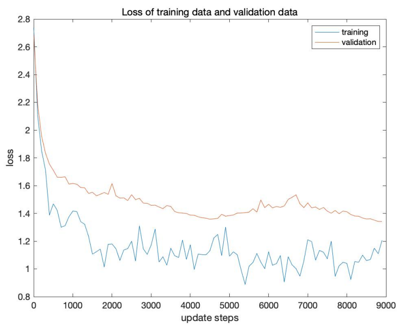

Figure 2: Loss of 3 layer network without batch normalization

The test accuracy of 3 layer network without batch normalization is 0.5268.

The loss and cost curve of 3 layer network with batch normalization is shown below.

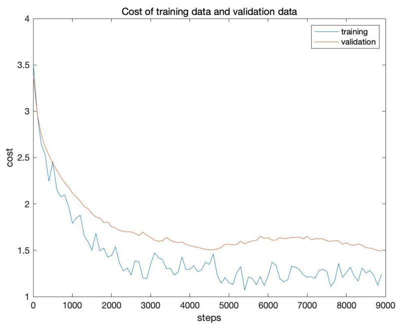

Figure 3: Cost of 3 layer network with batch normalization

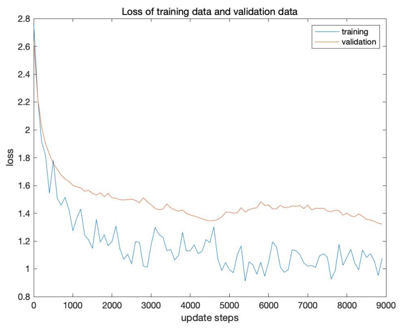

Figure 4: Loss of 3 layer network with batch normalization

The test accuracy of 3 layer network with batch normalization is 0.5276.

For three layer network, it is not obvious to notice the improvement on test accuracy when using batch normalization.

4 Evaluation of loss function of 9 layer network

The parameters are set the same as previous part except m is set to [50 30 20 20 10 10 10 10].

The loss and cost curve of 9 layer network without batch normalization is shown below

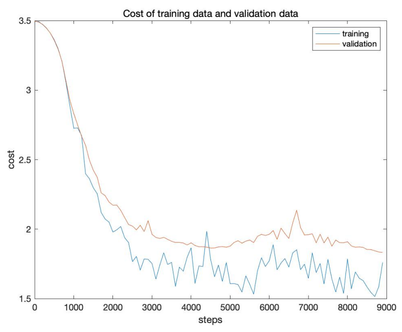

Figure 5: Cost of 9 layer network without batch normalization

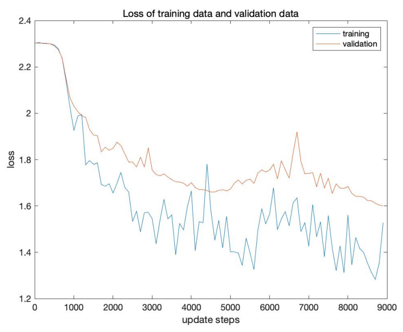

Figure 6: Loss of 3 layer network without batch normalization

The test accuracy of 9 layer network without batch normalization is 0.4274.

The loss and cost curve of 9 layer network with batch normalization is shown below.

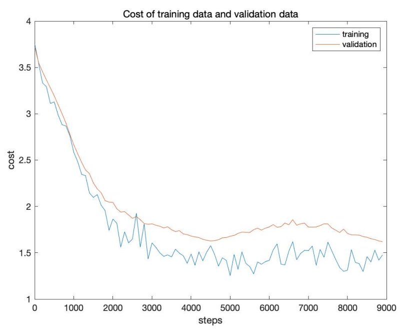

Figure 7: Cost of 9 layer network with batch normalization

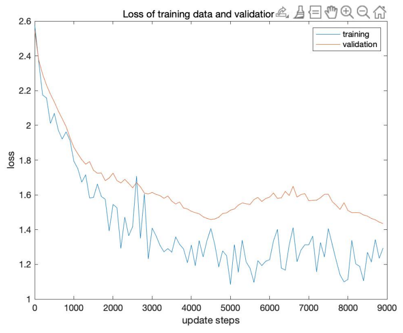

Figure 8: Loss of 9 layer network with batch normalization

The test accuracy of 9 layer network with batch normalization is 0.5047.

It is obvious to notice that with batch normalization, there is an obvious improvement on test accuracy. The results show that batch normalization make the training process more stable and it acts as a form of regularization.

5 lambda search for 3 layer network

In this part, I have finished the coarse-to-fine random search to set lambda.

I first perform a coarse search for lambda ranging from 0.01 to 0.001. The lambda is generated by the code shown below.

$$l = l_{min} + (l_{max} - l_{min}) * rand(1,1) \tag{1}$$

In my experiment, a uniform grid of 10 values are tested. And the best 3 values is shown below, which are 0.00428 with accuracy 0.5293, 0.00569 with accuracy 0.5287 and 0.00319 with accuracy 0.5307.

Based on the optimal lambda obtained in the above process, we could implement fine search and the range of lambda is set from 0.003 to 0.006. And lambda with best accuracy is 0.00358, the accuracy is 0.5325.

Keep the other parameter unchanged and set lambda to 0.00358 for 9-layer network, the final test accuracy I could achieve is 0.5158, which is greater than the accuracy I achieved in section 4.

6 Sensitivity to Initialization

In this section , each weight parameter is initialized to be normally distributed with sigmas equal to the same value sig at each layer.

The parameters are the same as previous section.

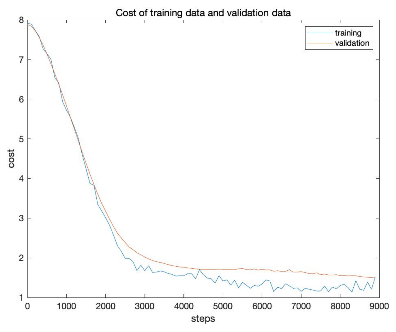

Figure 9: Sigma = 1e-1, Cost with batch normalization

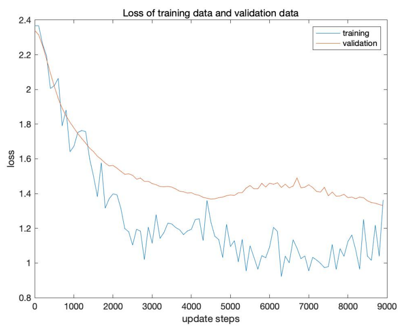

Figure 10: Sigma = 1e-1, Loss with batch normalization

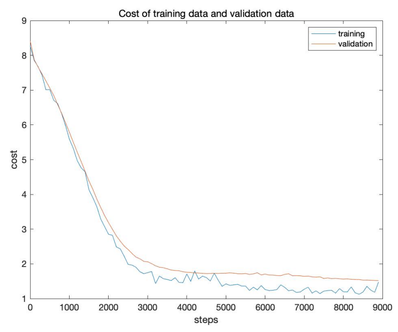

Figure 11: Sigma = 1e-1, Cost without batch normalization

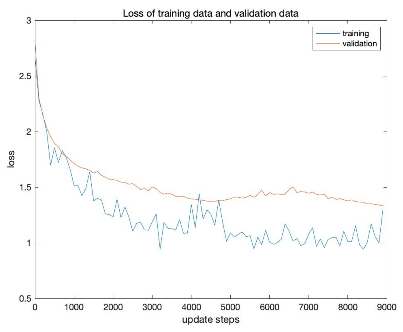

Figure 12: Sigma = 1e-1, Loss without batch normalization

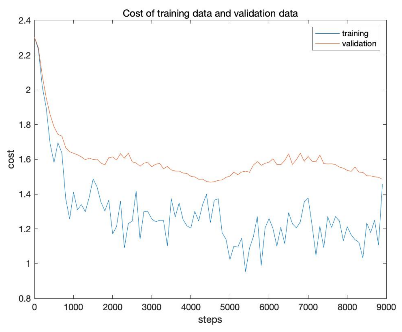

Figure 13: Sigma = 1e-3, Cost with batch normalization

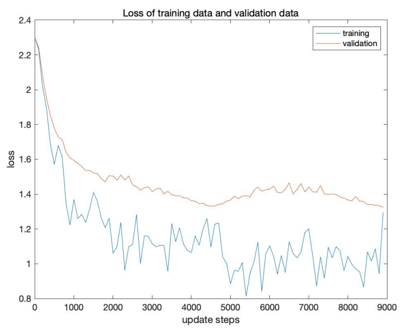

Figure 14: Sigma = 1e-3, Loss with batch normalization

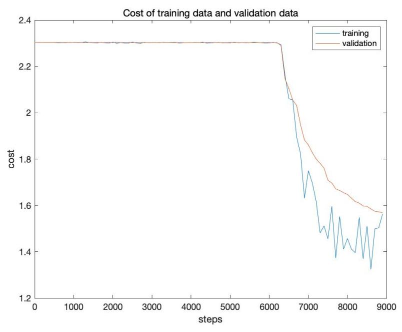

Figure 15: Sigma = 1e-3, Cost without batch normalization

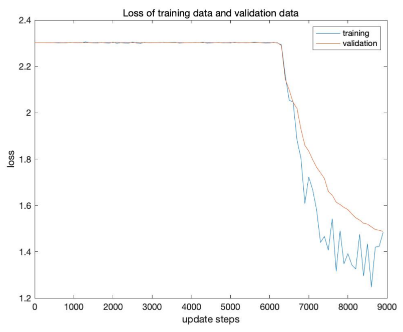

Figure 16: Sigma = 1e-3, Loss without batch normalization

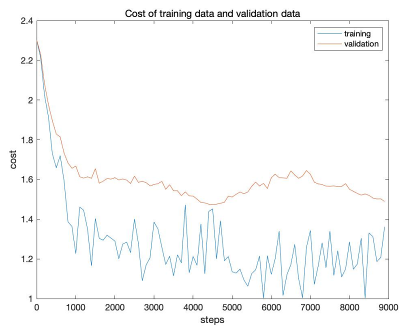

Figure 17: Sigma = 1e-4, Cost with batch normalization

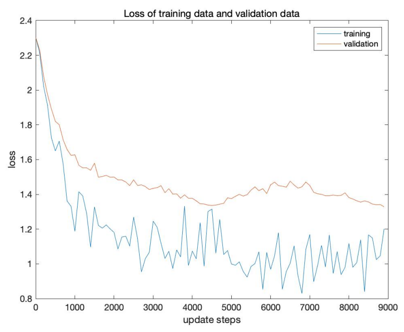

Figure 18: Sigma = 1e-4, Loss with batch normalization

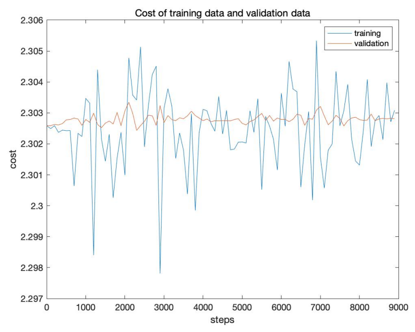

Figure 19: Sigma = 1e-4, Cost without batch normalization

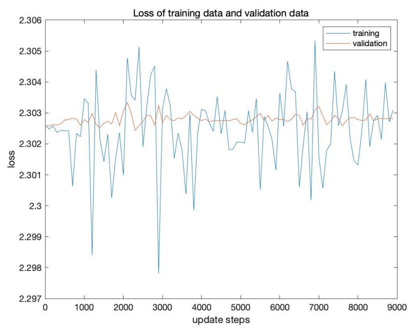

Figure 20: Sigma = 1e-4, Loss without batch normalization

From the above figure, we could find that when sigma equals 1e-1, whether using batch normalization or not has little influence on the gradient descent. However, for the network without batch normalization, with sigma equals to 1e-3, the cost without batch normalization remains constant initially. For sigma equals to 1e-4, the weight parameters could not be updated. However, for network with batch normalization, this problem would not happen.