Deep Learning in Data Science (DD2424)
Report to Assignment 3

Cong Gao

April 25, 2020

1 Introduction

In this assignment, I trained and tested k-layer networks with multiple outputs to classify images from the CIFAR-10 dataset. I updated the code from assignment 2 in two following ways:
1. Generalize the code so that I can train and test k-layer networks.
2. Incorporate batch normalization into the k-layer network both for training and testing.

2 Results and conclusions of assignment 3

2.1 Analytic gradient computations check

2.1.1 Without batch normalization

The numerical gradient should be compared to the analytical gradient to ensure that the analytical gradient is computed correctly. My way to check gradient computation was to compare the numerically and analytically computed gradient vectors (matrices) on reduced version of the input data with reduced dimensionality, by examining their absolute differences and declaring if all these absolute differences are small ($\leq 1e-6$).

$k=2$	W_1	b_1	W_2	b_2
Difference	$< 1e-8$	$< 1e-8$	$< 1e-8$	$< 1e-8$

Table 1: 2-layer network

$k=3$	W_1	b_1	W_2	b_2	W_3	b_3
Difference	$< 1e-8$	$< 1e-8$	$< 1e-8$	$< 1e-9$	$< 1e-8$	$< 1e-9$

Table 2: 3-layer network

$k=4$	W_1	b_1	W_2	b_2
Difference	$< 1e-9$	$< 1e-9$	$< 1e-8$	$< 1e-8$
$k=4$	W_3	b_3	W_4	b_4
Difference	$< 1e-8$	$< 1e-9$	$< 1e-8$	$< 1e-9$

Table 3: 4-layer network

Then I can draw the conclusion that my analytical gradient computation was correct.

2.1.2 Without batch normalization

I took the same way as before and the check results are shown as follows.

$k=2$	W_1	b_1	W_2	b_2
Difference	$< 1e-8$	$< 1e-9$	$< 1e-8$	$< 1e-8$
$k=2$	γ_1	β_1		
Difference	$< 1e-9$	$< 1e-9$		

Table 4: 2-layer network

$k=3$	W_1	b_1	W_2	b_2	W_3	b_3
Difference	$< 1e-8$	$< 1e-16$	$< 1e-8$	$< 1e-9$	$< 1e-8$	$< 1e-9$
$k=3$	γ_1	β_1	γ_2	β_2		
Difference	$< 1e-8$	$< 1e-9$	$< 1e-9$	$< 1e-9$		

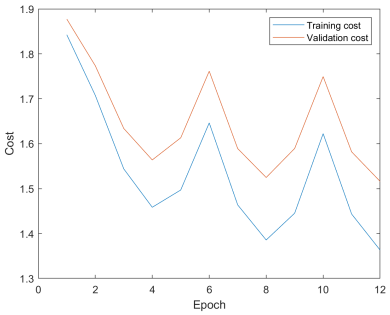
Table 5: 3-layer network

Then I can draw the conclusion that my analytical gradient computation was correct.

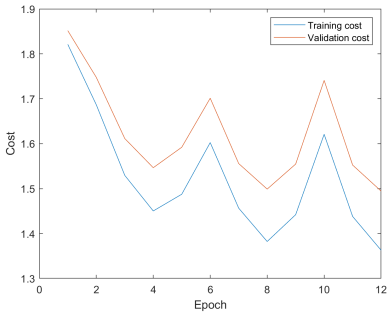
2.2 Graphs of the evolution of the loss function and classification accuracy

2.2.1 Two-layer networks

2-layer network has 50 nodes in the hidden layer and other parameters are the same as Assignment 2.



(a) Cost plot for three cycles training in Assignment 3



(b) Cost plot for three cycles training in Assignment 2

	Training data	Validation data	Test data
Accuracy	58.01%	52.32%	52.10%

Table 6: Classification accuracy in Assignment 3 with 2 layers

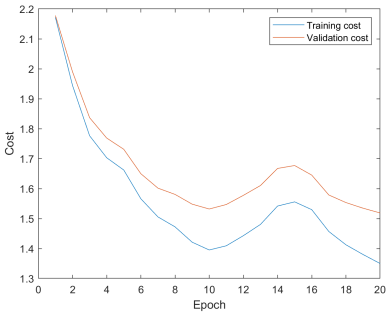
	Training data	Validation data	Test data
Accuracy	57.51%	52.30%	52.52%

Table 7: Classification accuracy in Assignment 2

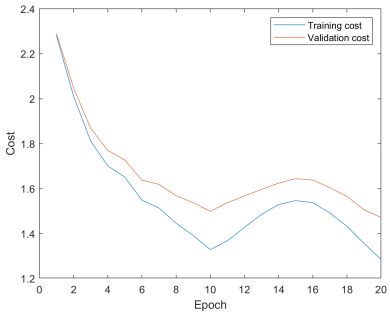
Results in Assignment 3 and Assignment 2 are the same, so the new version code is right and then I can continue to train 3 or more layers networks.

2.2.2 Three-layer networks

Three-layer network has 50 and 50 nodes in the first and second hidden layer respectively with the parameters setting: $n\_batch = 100$, $eta\_min = 1e-5$, $eta\_max = 0.1$, $\lambda = 5e-3$, $n\_s = 5 * 45000/n\_batch$, $n\_cycles = 2$. I used Xavier initialization and also randomly shuffled the order of the training data after each epoch.



(a) Cost plot for two cycles training (without BN)



(b) Cost plot for two cycles training (with BN)

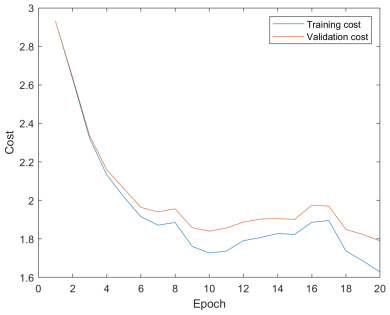
Accuracy	Training data	Validation data	Test data
Without BN	59.78%	53.68%	53.71%
With BN	61.51%	54.08%	53.26%

Table 8: Classification accuracy of three-layer network

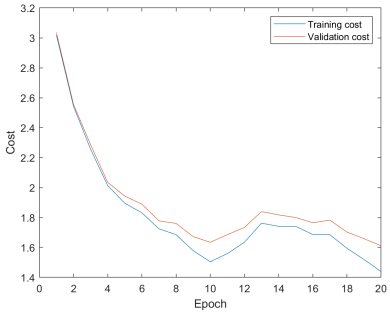
The classification accuracy on test set is improved a little with three-layer network compared to two-layer network. And the results of the network without batch normalization and the results of the network with batch normalization are similar.

2.2.3 Nine-layer networks

Then I trained a 9-layer network whose number of nodes at the hidden layers are [50, 30, 20, 20, 10, 10, 10, 10]. This network has the same number of weight parameters as the earlier network. And I trained the network with the same hyper-parameter settings as before.



(a) Cost plot for two cycles training (without BN)



(b) Cost plot for two cycles training (with BN)

Accuracy	Training data	Validation data	Test data
Without BN	47.59%	42.30%	42.51%
With BN	56.60%	49.68%	50.76%

Table 9: Classification accuracy of Nine-layer network

For the deeper network without batch normalization, its performance dropped by quite a bit. Generally as a network becomes deeper it becomes harder to train when training with variants of mini-batch gradient descent and using a more standard decay of the learning rate. For deeper network, The result of the network with batch normalization is much better than that of the network without batch normalization.

2.3 Search for λ for the 3-layer network with batch normalization

2.3.1 Coarse search

The range of the values I searched for λ is $[1e-5, 1e-1]$

λ	$1e-5$	$1e-4$	$1e-3$	$1e-2$	$1e-1$
Test accuracy	51.12%	51.89%	52.44%	53.12%	46.92%

Table 10: Classification accuracy of coarse search

2.3.2 Fine search

The range of the values I searched for λ is $[1e-3, 1e-2]$

λ	$1e-3$	$2e-3$	$3e-3$	$4e-3$	$5e-3$
Test accuracy	52.44%	53.14%	52.68%	53.68%	53.26%
λ	$6e-3$	$7e-3$	$8e-3$	$9e-3$	$1e-2$
Test accuracy	52.79%	53.26%	52.96%	53.14%	53.12%

Table 11: Classification accuracy for fine search

The best found λ is $4e-3$. And the graph of cost function and classification accuracy are shown as below.

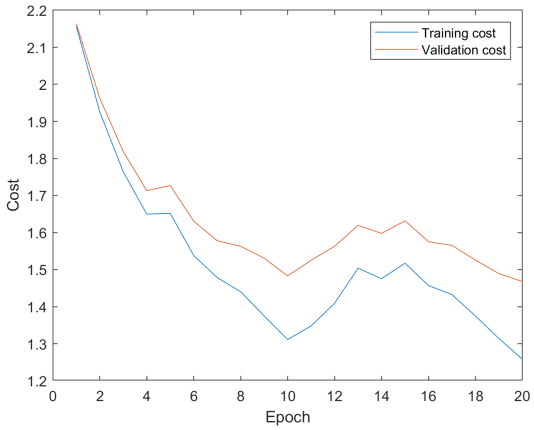


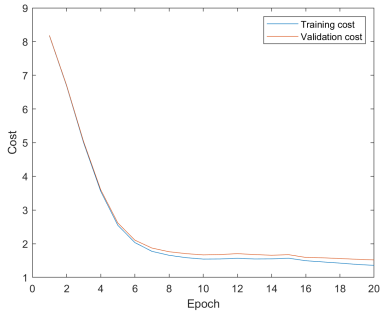
Figure 4: Cost plot for best found λ

	Training data	Validation data	Test data
Accuracy	62.35%	54.34%	53.68%

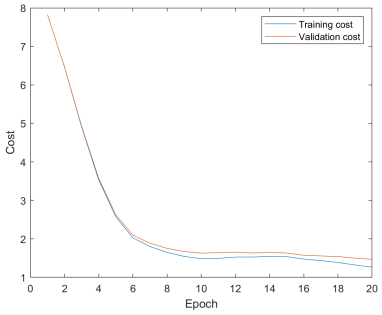
Table 12: Classification Accuracy for best found λ

2.4 Sensitivity to initialization

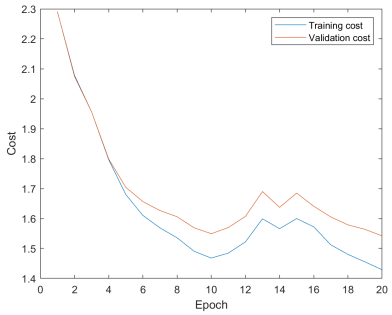
For each training regime instead of using Xavier initialization, I initialized each weight parameter to be normally distributed with sigmas equal to the same value σ at each layer (3-layer network).
Other parameters: $n\_batch = 100$, $eta\_min = 1e-5$, $eta\_max = 0.1$, $\lambda = 4e-3$, $n\_s = 5 * 45000/n\_batch$, $n\_cycles = 2$.



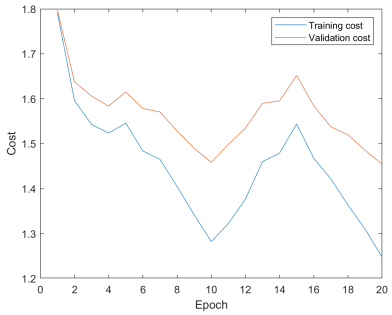
(a) Cost plot for $\sigma = 0.1$ without BN



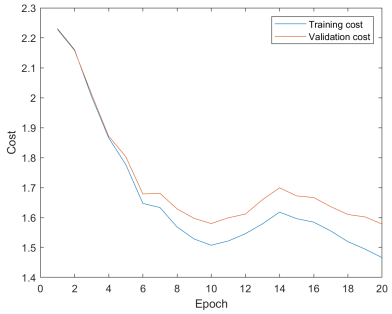
(b) Cost plot for $\sigma = 0.1$ with BN



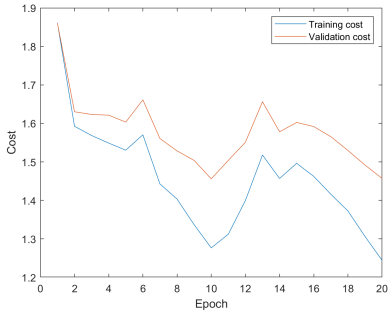
(a) Cost plot for $\sigma = 1e-3$ without BN



(b) Cost plot for $\sigma = 1e-3$ with BN



(a) Cost plot for $\sigma = 1e-4$ without BN



(b) Cost plot for $\sigma = 1e-4$ with BN

σ	$1e-1$	$1e-3$	$1e-4$
Without batch normalization	53.48%	50.14%	48.72%
With batch normalization	53.38%	53.73%	53.63%

Table 13: Test accuracy of three-layer networks with constant sigma value

While there was no batch normalization, the performance (classification accuracy) of the network dropped a lot. However, while I applied batch normalization, the performance of the network was almost not affected by the value of σ. Therefore, the network is more sensitive to initialization when without batch normalization.