DD2424 Assignment3 Mandatory Part

Shuyuan Zhang shuyuanz@kth.se

April 2019

1 Brief Introduction

In this assignment, we were asked to implement a multi-layer network with batch normalization to classify the cifar-10 data set.

2 Main Content

2.1 Gradient Check

To check whether the gradient was computed correctly, I compared my gradients with results of the provided function *ComputeGradsNumSlow* when $h = 1e-5$. The discrepancy between results (W, Gammas, Betas) given by different functions is measured by:

$$\frac{|g_a - g_n|}{max(1e-15, |g_a| + |g_n|)}$$

Because the network's bias parameters b are superfluous when using batch normalization as you will subtract away these biases when you normalize. These bias parameters will be estimated as effectively zero vectors when you train. So I didn't check gradients for them.

Results for a 2-layer network with hidden nodes [50]:

Layer No.	W difference	Gamma difference	Beta difference
1	6.7424e-10	3.2765e-10	3.6712e-10
2	1.1698e-10	/	/

Table 1: Gradient difference for a 2-layer network

Results for a 3-layer network with hidden nodes [50, 50]:

Layer No.	W difference	Gamma difference	Beta difference
1	8.1016e-10	4.7789e-10	6.9653e-10
2	5.2594e-10	2.3735e-10	2.1849e-10
3	1.1211e-10	/	/

Table 2: Gradient difference for a 3-layer network

Results for a 4-layer network with hidden nodes [50, 30, 20]:

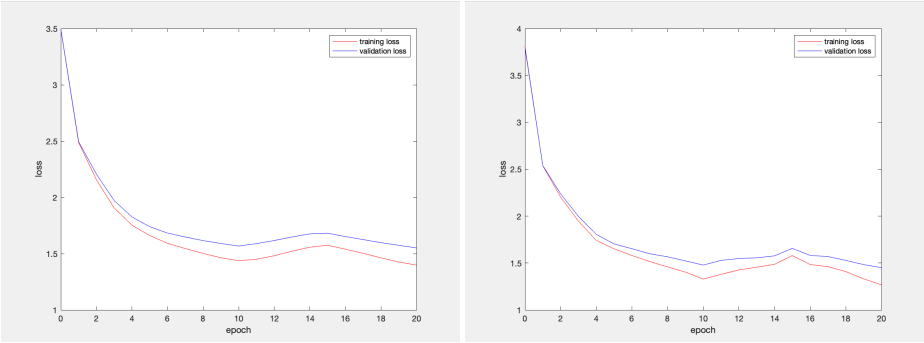
Layer No.	W difference	Gamma difference	Beta difference
1	1.0140e-09	6.0226e-10	1.2399e-09
2	5.6572e-10	3.4439e-10	4.8118e-10
3	3.6141e-10	2.7365e-10	1.7387e-10
4	1.1358e-10	/	/

Table 3: Gradient difference for a 4-layer network

From the table we can notice that discrepancies are rather small. So we may safely say that the gradients were calculated correctly and my gradient computations were bug free.

2.2 3-layer Network with/without BN

The graphs of the evolution of the loss function are shown below:



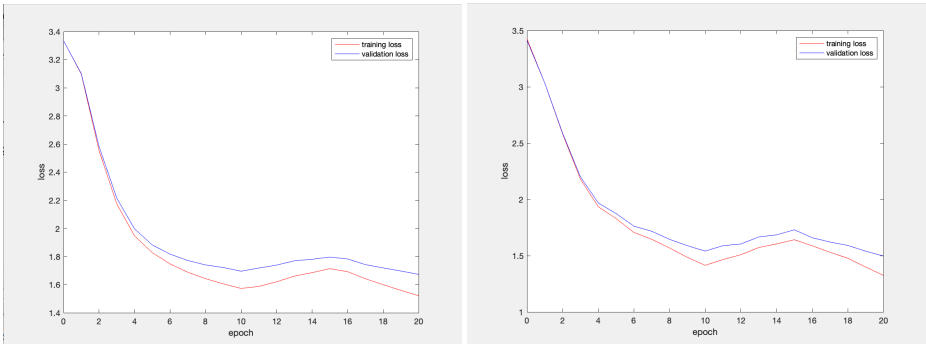
(a) Loss plot without BN (b) Loss plot with BN

Figure 1: Training curves for 2 cycles of 3-layer network

The given default parameter setting is:
$train\_size = 45000$, $n\_batch = 100$, $eta\_min = 1e-5$, $eta\_max = 1e-1$, $lambda = .005$, $n\_cycles = 2$, $n\_s = 5 * 45,000/n\_batch$, $num\_layers = 3$, $hidden\_node = [50,\ 50]$ with He initialization.

2.3 6-layer and 9-layer Network with/without BN

The graphs of the evolution of the loss function are shown below:

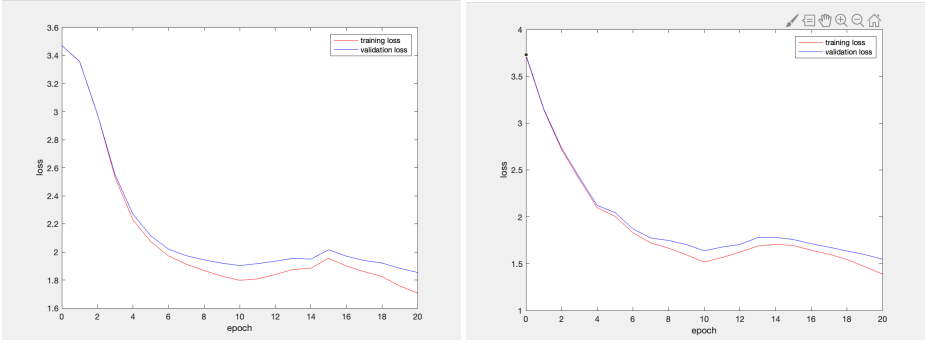


(a) Loss plot without BN (b) Loss plot with BN

Figure 2: Training curves for 2 cycles of 6-layer network

The given default parameter setting is:
$train\_size = 45000$, $n\_batch = 100$, $eta\_min = 1e-5$, $eta\_max = 1e-1$, $lambda = .005$, $n\_cycles = 2$, $n\_s = 5 * 45,000/n\_batch$, $num\_layers = 6$, $hidden\_node = [50,\ 30,\ 20,\ 20,\ 10]$ with He initialization.

Graphs of 9-layer network:



(a) Loss plot without BN (b) Loss plot with BN

Figure 3: Training curves for 2 cycles of 9-layer network

The given default parameter setting is:
$train\_size = 45000$, $n\_batch = 100$, $eta\_min = 1e-5$, $eta\_max = 1e-1$, $lambda = .005$, $n\_cycles = 2$, $n\_s = 5 * 45,000/n\_batch$, $num\_layers = 9$, $hidden\_node = [50,\ 30,\ 20,\ 20,\ 10,\ 10,\ 10,\ 10]$ with He initialization.

2.4 Accuracy of previous tests

Network	Validation accuracy
3-layer without BN	0.5313
3-layer with BN	0.5358
6-layer without BN	0.5103
6-layer with BN	0.5161
9-layer without BN	0.4072
9-layer with BN	0.4938

Table 4: Network performance on the validation set

We can notice that using BN may have a better effect when the network is deep. Also, with BN, loss function can be lower when training was done in deep networks.

2.5 Search for lambda in a 3-layer network

The range of lambda is from $1e-05$ to $1e-01$ with all other parameters remain the same as the 3-layer network in previous sections.

Test No.	lambda	validation accuracy
1	0.0142	0.5171
2	7.7117e-05	0.5244
3	0.0028	0.5368
4	7.5874e-05	0.513
5	0.0404	0.4982
6	6.0795e-05	0.518
7	2.0871e-05	0.5192
8	0.0973	0.4742
9	0.0211	0.509
10	0.0027	0.5376
11	1.9105e-04	0.515
12	0.0212	0.4894
13	0.0038	0.5308
14	0.0055	0.5255
15	7.2606e-04	0.5238
16	4.2473e-04	0.515
17	7.3978e-04	0.5174
18	1.2321e-05	0.5008
19	0.0072	0.5294
20	0.0280	0.4776

Table 5: Coarse Search Results

We may notice that when lambda is between $1e-03$ and $1e-02$, the validation

accuracy is relatively big(above 0.53). So in my fine search I will set the range of lambda from $1e-03$ to $1e-02$ with all other parameters remain the same.

Test No.	lambda	validation accuracy
1	0.0035	0.5358
2	0.0031	0.5378
3	0.0021	0.5336
4	0.0088	0.5186
5	0.0051	0.5316
6	0.0026	0.5262
7	0.0067	0.5258
8	0.0017	0.5334
9	0.0029	0.5392
10	0.0060	0.5214
11	0.0049	0.5246
12	0.0045	0.5334
13	0.0093	0.5248
14	0.0055	0.5255
15	0.0026	0.5336
16	0.0047	0.5352
17	0.0054	0.5284
18	0.0069	0.5292
19	0.0033	0.528
20	0.0086	0.5258

Table 6: Fine Search Results

We can see from the table that the network performs best when lambda is around 0.0030. The lambda setting for my best performing 3-layer network is 0.0029 with a validation accuracy of 0.5392.
After I have found a good setting for lambda, I trained this network for 3 cycles and see what test accuracy this network can achieve.

The test accuracy achieved by this network is 0.5316 after 3 cycles.

2.6 Sensitivity to Initialization

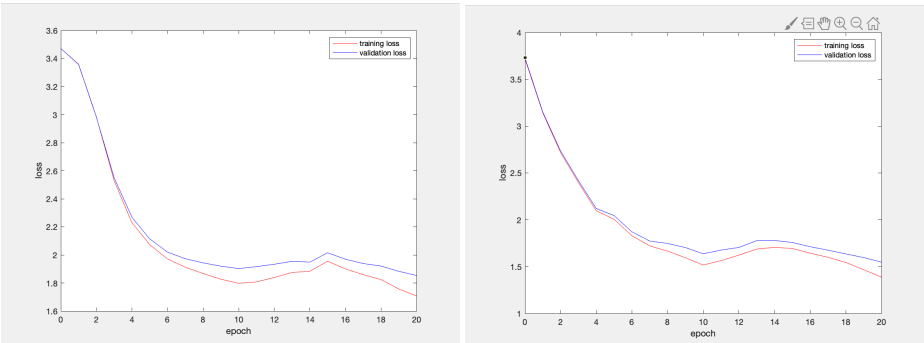
In this part I will still use the 3-layer network with lambda = 0.005.

The parameter setting is:
$train\_size = 45000,\ n\_batch = 100,\ eta\_min = 1e-5,\ eta\_max = 1e-1,\ lambda = .005,\ n\_cycles = 2,\ n\_s = 5 * 45,000/n\_batch,\ num\_layers = 3,\ hidden\_node = [50,\ 50]$

Instead of using He initialization, I initialized each weight parameter to be normally distributed with *sigmas* equal to the same value *sig* at each layer.

For each training regime I set $sig = 1e-1$, $1e-3$ and $1e-4$ respectively and train the network with/without BN and see the effect on the final test accuracy.

$sig = 1e-1$:

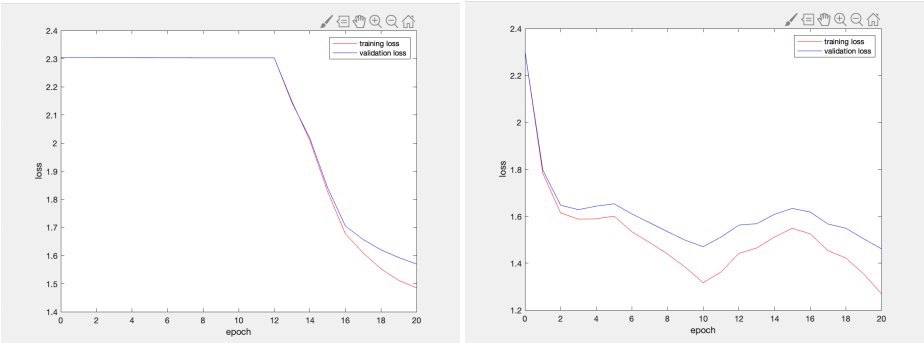


(a) Loss plot without BN (b) Loss plot with BN

Figure 4: Training curves for $sig = 1e-1$

No BN test accuracy: 0.5271
BN test accuracy: 0.5233

$sig = 1e-3$: 5284 5257 4886 4921

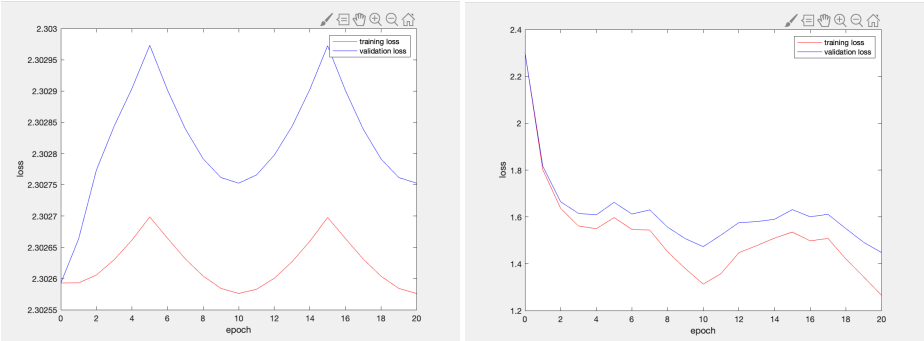


(a) Loss plot without BN (b) Loss plot with BN

Figure 5: Training curves for $sig = 1e-3$

No BN test accuracy: 0.4921
BN test accuracy: 0.5257

$sig = 1e-4$:



(a) Loss plot without BN (b) Loss plot with BN

Figure 6: Training curves for $sig = 1e-4$

No BN test accuracy: 0.1
BN test accuracy: 0.5276

In general, networks with BN are less sensitive to different initialization methods and parameters. When all the weight parameters are initialized under the same Gaussian distribution, networks with BN can maintain their performance as std changes.
While networks without BN is highly sensitive to initialization, especially when std is small. They may encounter serious gradient issues when std is very small thus the training process cannot be done normally.