Deep Learning Assignment3

Mengdi Xue, mengdix@kth.se

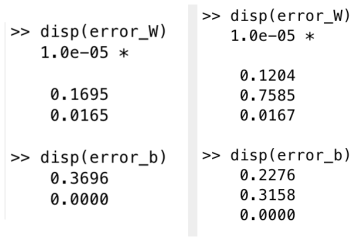
May 2018

1 Checking gradient

I checked my analytic gradient computations through computing the relative error using this formula:

$$\frac{|g_a - g_n|}{max(eps, |g_a + g_n|)}$$

The result is as following:



```
>> disp(error_W)
   1.0e-05 *

    0.1695
    0.0165

>> disp(error_b)
    0.3696
    0.0000
```

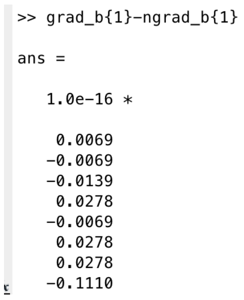
```
>> disp(error_W)
   1.0e-05 *

    0.1204
    0.7585
    0.0167

>> disp(error_b)
    0.2276
    0.3158
    0.0000
```

Figure 1: Relative error, Left: 2 layer, Right: 3 layers

We can see that the error of W is very small. Relative error of b seems large because the eps here is $2.2204e-16$ but $|g_a - g_n|$ here is all smaller than $1.0e-16$. We can see from Figure 2. In this way, I think my gradient computation is right.



```
>> grad_b{1}-ngrad_b{1}

ans =

   1.0e-16 *

    0.0069
   -0.0069
   -0.0139
    0.0278
   -0.0069
    0.0278
    0.0278
   -0.1110
```

Figure 2: Absolute differences

2 Loss with and without batch normalization

The graphs of the evolution of the loss function when you tried to train your 3-layer network without batch normalization and with batch normalization is as in Figure 4.

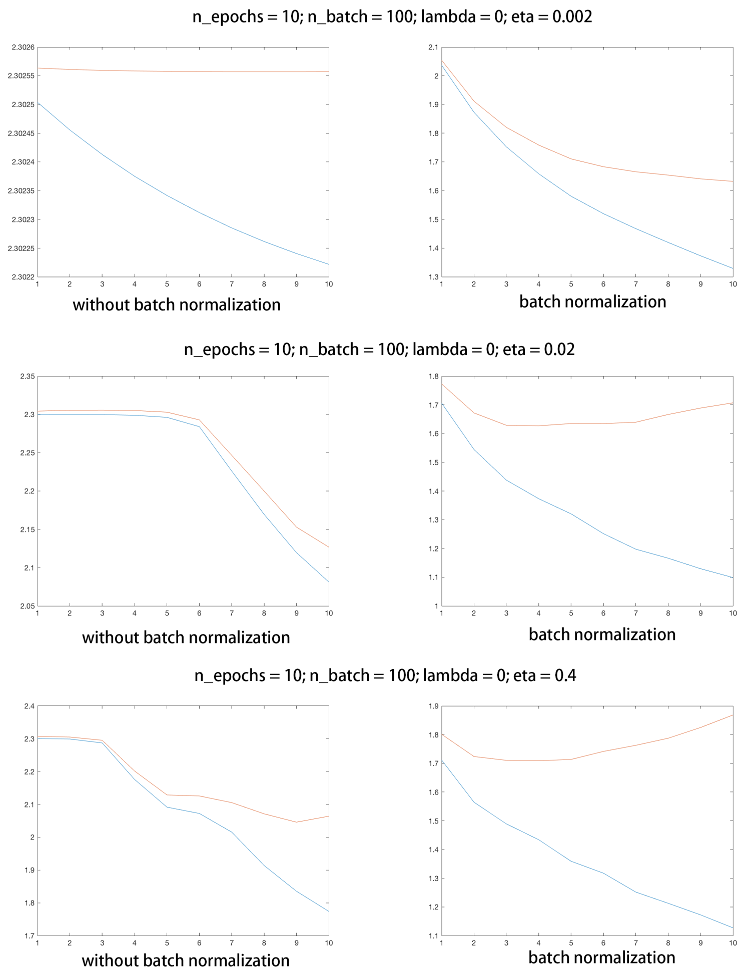


Figure 3: Loss with and without batch normalization

We can see that with batch normalization, the stability of the network is better and it helps to learn. For example, when eta = 0.002, after 10 epoches, the network without batch normalization can only reduce the training loss by about 0.0003, but the network with batch normalization can help to reduce the training loss by 0.7.

3 Coarse and fine search

The range of the values I searched for λ and η for coarse search under log domain is as following:

$$lmin = -6; lmax = -1; emin = log10(0.003); emax = log10(0.04)$$

The range of the values I searched for λ and η for fine search under log domain is as following:

$$lmin = log10(4e{-}6); lmax = log10(4e{-}1); emin = log10(0.001); emax = log10(0.003);$$

The number of epochs used for training during coarse and fine search are all 10 epochs.

The hyper-parameter settings for my best performing 3-layer network trained with batch normalization is as in Table 1.

Table 1: 3 Best parameters for fine search

	λ	η
1	5.91E-05	0.00266124
2	1.00E-05	0.00493538
3	0.000897411	0.00179585

The best test accuracy achieved by my network is 43.18%. The loss figure is as Figure 4.

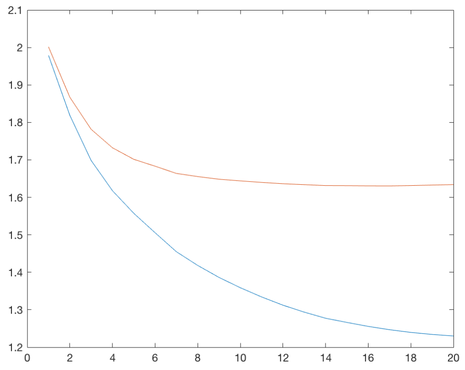


Figure 4: Loss with the best parameter

4 2-layer network with batch normalization

Here is the training and validation loss for my 2-layer network with batch normalization with 3 different learning rates for 10 epochs, and the same parameter settings with no batch normalization.

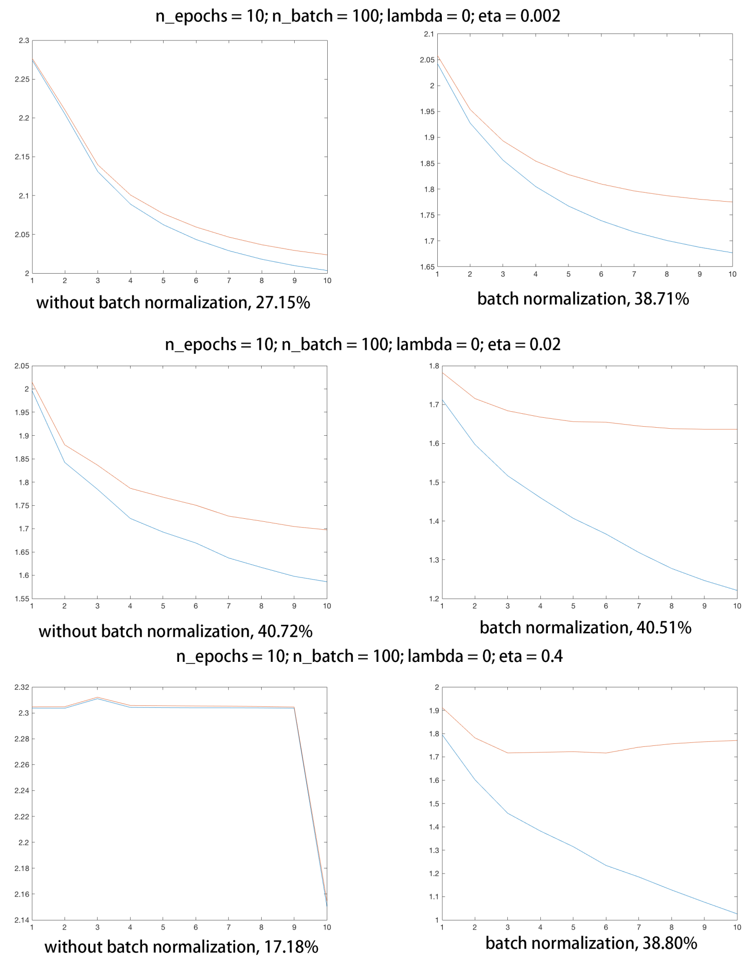


Figure 5: 2-layer network with and without batch normalization

We can see that in the second setting, which has the proper eta, batch normalization didn't help to improve the performance. But in the other two settings, in which the learning rate is either too high or too small, batch normalization helps to improve the performance a lot. So I think we can conclude that adding batch normalization can also help to improve the stability of 2 layer network.