DD 2424 - Assignment 3

Jia Fu (jiafu@kth.se)

May 14, 2021

Gradient check of k-layer network with batch normalization

Here I computed the relative errors between numerically computed gradients g_n and analytically computed gradients g_a:

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

I set `eps = 1e-5`. The numerically computed gradients were obtained by the given MATLAB function `ComputeGradsNumSlow`. The comparison was based on the mean values in the relative error matrices of W_l, b_l, γ_l, and β_l. I used two training samples and 10 dimensions for speed and numerical precision issues, $\lambda = 0.005$ and He initialization were applied.

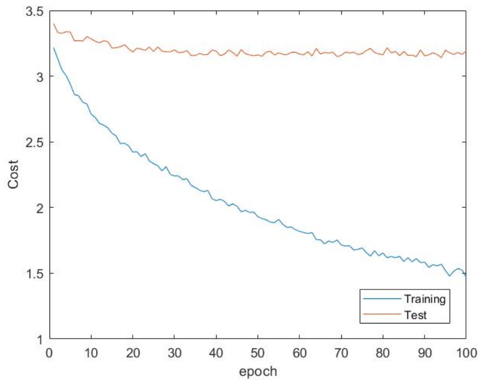
	W_l	b_l	γ_l	β_l
$l = 1$	8.2345e-08	8.8818e-12	3.0494e-10	3.0463e-10
$l = 2$	1.8325e-10	5.5403e-11	/	/

Table 1: Mean relative errors of W_l, b_l, γ_l, and β_l (2-layer network with 50 hidden nodes)

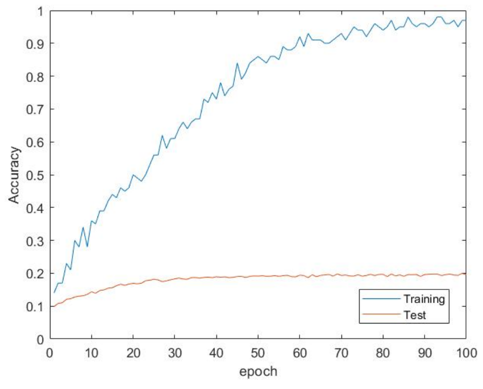
	W_l	b_l	γ_l	β_l
$l = 1$	7.0534e-08	8.8818e-07	6.2209e-06	5.7768e-06
$l = 2$	1.7443e-08	2.2209e-07	1.1730e-09	1.1730e-09
$l = 3$	1.1418e-10	2.9243e-11	/	/

Table 2: Mean relative errors of W_l, b_l, γ_l, and β_l (3-layer network with 50 and 50 hidden nodes respectively)

Mean relative errors in table 1 and 2 are all below 1e-5, indicating that the analytically computed gradients are fine for the following sanity check.



(a) Training and Test Cost



(b) Training and Test Accuracy

Figure 1: `eta = 0.001`, `alpha = 0.7`, `n_batch = 10`

I checked whether this 3-layer network could overfit 100 training samples and get a very low loss after training for 100 epochs. A good example of this can be seen in figure 1, which suggests that gradient computations are bug free for my k-layer network with batch normalization.

Comparison of results with and without batch normalization (BN)

In this part, 45000 data points were used for training and 5000 data points for validation. I used given default parameter settings: `n_batch = 100`, `eta_min = 1e-5`, `eta_max = 1e-1`, `lambda = 0.005`, and `n_s = 2250`. Two cycles of training and Xavier initialization were applied. In the condition of BN, `alpha` was set as 0.7 which is smaller than the typical number 0.9 since our training is shorter than usual. Figure 2 and 3 show the evolution of training and validation cost for the network trained with and without BN.

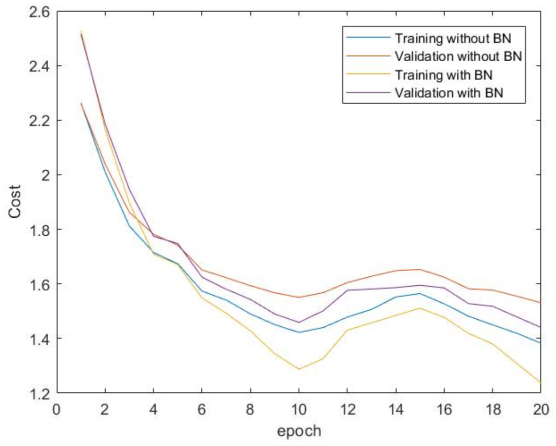


Figure 2: 3-layer network with hidden nodes [50, 50]

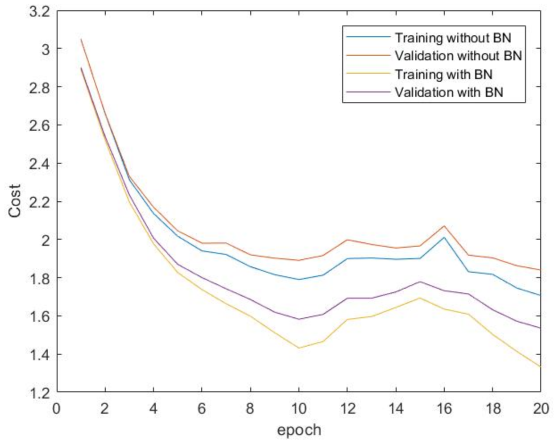


Figure 3: 9-layer network with hidden nodes [50, 30, 20, 20, 10, 10, 10, 10]

It can be seen that the direct effect of BN is to accelerate the convergence of the network. The test accuracy of 3-layer network is 53.69% without BN and 53.97% with BN, which doesn't show much improvement. But for 9-layer network, the accuracy without BN 40.28% can be raised to 51.73% with BN. BN alleviates the Internal Covariate Shift problem in the network and plays an effective role in training multi-layer network models.

Lambda search for 3-layer network model

In this section, I set `n_s = 900`, the other parameters were kept the same as last section. Coarse search was done on log scale, let $l = log_{10}\lambda$, then

$$l = l_{min} + (l_{max} - l_{min}) * random(0, 1)$$

In my experiments for the coarse search I set $l_{min} = -5$ and $l_{max} = -1$, a uniform grid with 20 different values were tested. Three best results are:

	λ	Validation Accuracy
1	0.001489	52.30%
2	0.001948	52.22%
3	0.006807	52.18%

Table 3: λ settings for the 3 best performing networks from the coarse search

Based on the best λ obtained in the coarse search, the range of λ was set as $0.001 - 0.007$ in the fine search. Other parameters were identical to the coarse search, a uniform grid with 10 different values were tested. Three best results are:

	λ	Validation Accuracy
1	0.003221	52.70%
2	0.006834	52.68%
3	0.006144	52.54%

Table 4: λ settings for the 3 best performing networks from the fine search

Set `n_s = 2250` and `lambda = 0.003221`, keep the other parameters unchanged. The test accuracy achieved by this network can reach 54.00% after 3 cycles of training. Using the same parameters settings, test accuracy can reach 52.63% for 9-layer network which is also praiseworthy.

Sensitivity to initialization

In this part, each weight parameter was initialized to be normally distributed with σ equal to the same value `sig` at each layer. Use best searched `lambda` and other parameter settings in the second section. Figure 4, 5, and 6 show the training and validation cost for the network trained with and without BN given different `sig`.

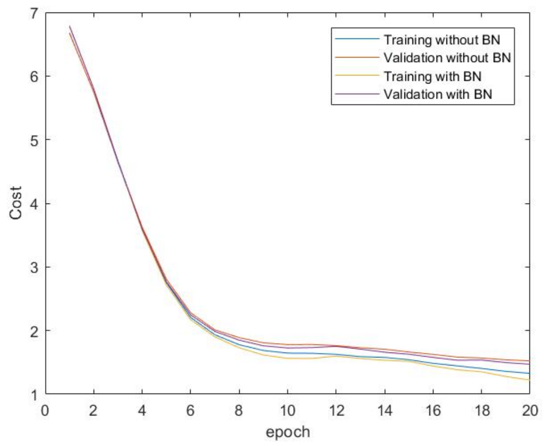


Figure 4: `sig = 1e-1`

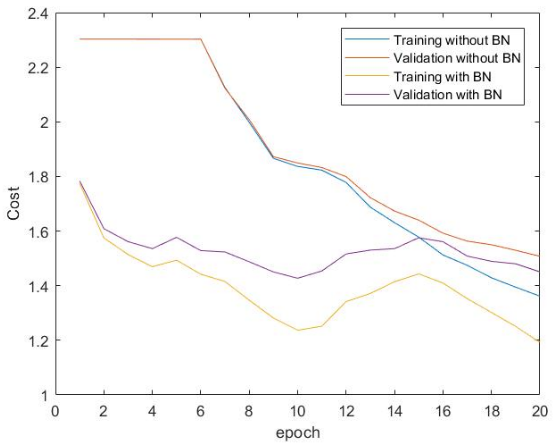


Figure 5: `sig = 1e-3`

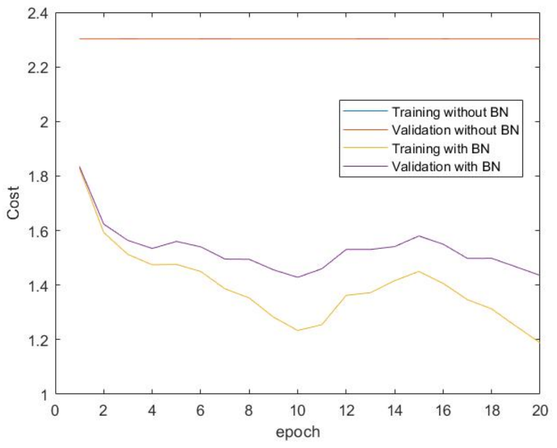


Figure 6: `sig = 1e-4`

σ	test accuracy without BN	test accuracy with BN
1e-1	53.11%	52.69%
1e-3	52.11%	52.87%
1e-4	10.00%	53.75%

Table 5: Test accuracy with and without BN using different `sig` to initialize weight parameters

When `sig = 1e-1`, whether use BN or not makes little difference to the gradient descent. But when `sig = 1e-3`, the cost of the network without BN remains constant in the first six epochs, even when `sig = 1e-4`, the weight parameters are failed to update and get only 10% of test accuracy. When the weight parameters are initialized very close to 0, the computation results of gradients will be 0. However, there are no such problems for the network with BN.
After the BN operation, the effect of the weight parameters' scale will be eliminated, so the distribution of the input data is adjusted within a certain range. Adaptability of networks to the parameters of different size and the stability of parameters' update process are guaranteed in this way.