Batch Norm

Rahul Deshmukh

April 22, 2020

1 Question posted on Piazza

In today's class for Batch Normalization, we said that there are no learnable parameters. However, pytorch's documentation says that the γ, β are learnable parameters of size 'C' (number of input channels). Also, we compute the constants μ_B, σ_B^2 per channel. Thus we would need to back propagate to update the learnable parameters.

From my calculations, I get the following gradients:

$$
\begin{aligned}
g_\gamma &= \hat{z}^T g_y \\
g_\beta &= g_y \\
g_z &= \frac{\gamma}{\sqrt{\sigma_B^2 + \epsilon}} \odot g_y
\end{aligned}
$$

can anyone verify if they get the same expressions? Also why does this help in making the training fast? Is it because of the multiplicative factor to g_z that the gradients will not get muted?

2 My Asnwer:

A quick background read on batch normalization made me understand why batch normalization makes training faster. The key factor is the multiplicative factor to the gradient.

When training a DNN we can encounter two issues with gradients, (1) Vanishing gradients (2) Exploding gradients. While in case (1) we stop learning whereas in case (2) the gradient direction no longer ensures descent which causes problems in convergence. Batch Normalization handles both these cases effectively by rescaling the gradient by $\frac{1}{\sqrt{\sigma_B^2+\epsilon}}$ to ensure descent for case (2) and scaling by γ as a solution to case (1).

To see why the gradient direction is not a descent direction we can write the second order taylor's expansion of loss:

$$
\begin{aligned}
d^{(k)} &= -\nabla_\theta L(\theta^{(k)}) \\
\theta^{(k+1)} &= \theta^{(k)} + \alpha d^{(k)} \\
L(\theta^{(k+1)}) &= L(\theta^{(k)}) + (\nabla_\theta L(\theta^{(k)}))^T(\theta^{(k+1)} - \theta^{(k)}) + \frac{1}{2!}(\theta^{(k+1)} - \theta^{(k)})^T(\nabla_\theta^2 L(\theta^{(k)}))(\theta^{(k+1)} - \theta^{(k)}) \\
\Rightarrow L(\theta^{(k+1)}) &= L(\theta^{(k)}) - \alpha||d^{(k)}||_2^2 + \frac{\alpha^2}{2}(d^{(k)})^T(\nabla_\theta^2 L(\theta^{(k)}))(d^{(k)})
\end{aligned}
$$

Thererefore if $\frac{\alpha^2}{2}(d^{(k)})^T(\nabla_\theta^2 L(\theta^{(k)}))(d^{(k)}) \geq \alpha||d^{(k)}||_2^2$ then $d^{(k)}$ is not a descent direction. To ensure that it is a direction of descent we can either decrease α or rescale the gradients and

hessian. BN takes the second approach which is better because then we can choose step size without worrying about descent.

BN recenters and scales the input (y) to $z = \frac{y-\mu}{\sigma}$. Due to such a transformation the Loss gradient and Hessian also get scaled by $\frac{1}{\sigma}$. And the new taylors series is given by:

$$L(\theta^{(k+1)}) = L(\theta^{(k)}) - \frac{\alpha}{\sigma^2}||d^{(k)}||_2^2 + \frac{\alpha^2}{2\sigma^3}(d^{(k)})^T(\nabla_\theta^2 L(\theta^{(k)}))(d^{(k)})$$

Therefore the quadratic term can no longer be greater than the first order term and hence we have a descent direction.

BN also rescales the gradient by γ which helps in case of vanishing gradient and the effect of BN can be understood as a method of increasing the gradient magnitude through a learnable parameter.