Jiang
计算机视觉/户外运动爱好者

不请自来，**BN本质上解决的是反向传播过程中的梯度问题。**

详细点说，反向传播时经过该层的梯度是要乘以该层的参数的，即前向有：

$$h_l = w_l^T h_{l-1}$$

那么反向传播时便有：

$$\frac{\partial l}{\partial h_{l-1}} = \frac{\partial l}{\partial h_l} . \frac{\partial h_l}{\partial h_{l-1}} = \frac{\partial l}{\partial h_l} w_l$$

那么考虑从l层传到k层的情况，有：

$$\frac{\partial l}{\partial h_k} = \frac{\partial l}{\partial h_l} \prod_{i=k+1}^{l} w_i$$

上面这个 $\prod_{i=k+1}^{l} w_i$ 便是问题所在。因为网络层很深，如果 w_i 大多小于1，那么传到这里的时候梯度会变得很小比如 0.9^{100} ；而如果 w_i 又大多大于1，那么传到这里的时候又会有梯度爆炸问题 比如 1.1^{100} 。BN所做的就是解决这个梯度传播的问题，因为BN作用抹去了w的scale影响。

具体有：

$$h_l = BN(w_l h_{l-1}) = BN(\alpha w_l h_{l-1})$$

那么反向求导时便有了：

$$\frac{\partial h_l}{\partial h_{l-1}} = \frac{\partial BN w_l h_{l-1}}{\partial h_{l-1}} = \frac{\partial BN \alpha w_l h_{l-1}}{\partial h_{l-1}}$$

可以看到此时反向传播乘以的数不再和 w 的尺度相关，也就是说尽管我们在更新过程中改变了 w 的值，但是反向传播的梯度却不受影响。更进一步：

$$\frac{\partial h_l}{\partial w_l} = \frac{\partial BN w_l h_{l-1}}{\partial w_l} = \frac{1}{\alpha} . \frac{\partial BN \alpha w_l h_{l-1}}{\partial w_l}$$

即尺度较大的 w 将获得一个较小的梯度，在同等的学习速率下其获得的更新更少，这样使得整体 w 的更新更加稳健起来。

总结起来就是BN解决了反向传播过程中的梯度问题（梯度消失和爆炸），同时使得不同sc
w 整体更新步调更一致。

更详细的解释可以看我写的一篇BN的文章Batch Normalization详解

编辑于 2017-05-04