Batch Normalization Notes

1. 背景

我们知道在神经网络训练开始前，都要对输入数据做一个归一化处理，原因在于神经网络学习过程本质就是为了学习数据的分布，一旦训练数据与测试数据的分布不同，那么网络的泛化能力也将大大降低；另外一方面，一旦每批训练数据的分布各不相同，那么网络就要在每次迭代都去学习适应不同的分布，这样将会大大降低网络的训练速度，这也正是为什么需要对数据都要做一个归一化预处理的原因。

我们知道网络一旦 train 起来，参数就要发生更新，除了输入层的数据外(因为输入层数据，我们已经人为的为每个样本归一化)，后面网络每一层的输入数据分布是一直在发生变化的，因为在训练的时候，前面层训练参数的更新将导致后面层输入数据分布的变化。我们把网络中间层在训练过程中数据分布的改变称之为“Internal Covariate Shift”。Paper 所提出的算法，就是要解决在训练过程中，中间层数据分布发生改变的情况。

2. Batch Normalization

- **本质**

  对网络中送入激活函数之前的输入进行归一化处理，使它的均值为 0，方差为 1。

- **算法（训练阶段）**

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1 \ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

（1） 这可以看成是训练阶段应用 BN 的步骤

（2） ε是为了计算的稳定性（防止被 0 除）

- **算法（测试阶段）**

（1） 论文中的方法

$$y = \frac{\gamma}{\sqrt{\mathrm{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\mathrm{E}[x]}{\sqrt{\mathrm{Var}[x]+\epsilon}}\right)$$

其中 E[x]=E[μ_B]　Var[x]=(m/m-1) * E[σ^2_b]

（2） Momentum

使用滑动平均的方法，cs231 中使用的这种方法

```
running_mean = running_mean * momentum + (1-momentum) * mean
running_var = running_var * momentum + (1-momentum) * var
```

3. BN 为什么有效

如果是仅仅使用归一化公式，对网络某一层的输出数据做归一化，然后送入网络下一层，这样是会影响到本层网络所学习到的特征的。打个比方，比如网络中间某一层学习到的特征数据本身就分布在 S 型激活函数的两侧，你强制把它给我归一化处理、标准差也限制在了 1，把数据变换成分布于 S 函数的中间部分，这样就相当于我这一层网络所学习到的特征分布被你搞坏了，于是文献使出变换重构，引入了可学习参数 γ、β，γ 一般初始化为 1，β 初始化为 0，但是引入这两个参数后，归一化之后的 x 可以通过改变这两个参数还原到归一化之前的分布，如果归一化之前的分布是最优的，网络通过学习可以将 x 还原回去。

4. BN 在 CNN 中的应用

观察上面的 $x_i$hat，它是一个（1，D）维的向量，D 表示该**全连接**层有 D 个神经元，γ、β也是一个 D 维的向量。对于卷积层，如果我们把特征图上的每个 cell 都当成一个神经元来处理，那γ、β的参数将会非常多，因此论文中提出将每个特征图当成一个神经元来处理（每个特征图只对应一个参数），如 x.shape=[N,H,W,C]，处理时把它当成[N*H*W,C]的全连接层进行归一化。

5. BP

$$\frac{\partial \ell}{\partial \hat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^m \frac{\partial \ell}{\partial \hat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2}(\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \left(\sum_{i=1}^m \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2+\epsilon}}\right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^m -2(x_i-\mu_{\mathcal{B}})}{m}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2+\epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i-\mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^m \frac{\partial \ell}{\partial y_i} \cdot \hat{x}_i$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^m \frac{\partial \ell}{\partial y_i}$$

6. BN 的优点

（1） 可以使用更大的学习率

（2） 移除 Dropout

（3） 减小 L2 正则化的参数，论文里减小了 5 倍

（4） 加速学习率衰减

（5） 移除局部响应归一化

（6） 更彻底的随机打乱图片出现的顺序，避免一些图片总是同时出现在一个 batch 里，这使准确率提高了 1%

（7） 减少对图片本身的扰动，因为使用 BN 训练的更快，训练时同一张图片出现的次数更少，因此让模型专注于真实图片

7. Take-aways

- 评估模型准确率用的是验证集