不得不说的Batch Normalization

原创 空字符 月来客栈 2020-12-31

收录于话题
#《跟我一起深度学习》

28个

由于公众号改版不再按照作者的发布时间进行推送，为防止各位朋友错过月来客栈推送的最新文章，大家可以手动将公众号设置为“星标⭐”以第一时间获得推送内容，感谢各位~

1 前言

各位朋友大家好，欢迎来到月来客栈。在前面的几篇文章中，笔者陆续介绍了LeNet5、AlexNet、VGG、NiN和GoogLeNet这五个网络模型。这五个模型可以说各有各的特点也各有各的优点，还有一个共同点就是它们都是在网络结构上进行创新改进的。在接下来的这篇文章中，我们将看到另外一种形式的创新——优化创新。所谓优化创新指的就是对网络的训练策略进行改进，以此来提高网络的性能（训练速度或预测精度），并且这样的改进策略还能够迁移到其它的网络结构中，例如对输入数据进行标准化等。

在《跟我一起机器学习》中，笔者介绍了为什么我们需要对输入数据进行标准化，并且也介绍了一种使用最为广泛的标准化方法。既然在机器学习中我们都需要对输入数据进行标准化，那在深度学习中我们还需要吗？如果需要的话是只对原始输入进行标准化就够了吗？为了解开这些疑惑，就让我们通过下面的这篇论文来一探究竟。

今天要介绍的这篇论文名字有点长，叫做Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift[1]，简称叫做Batch Normalization（BN）。这是一篇谷歌公司发表在ICML2015上的论文，并且从题目来看，我们就知道这篇论文的目的就是用来提升网络的训练速度。下面就让我们一起来看看这篇到底是通过什么样的方法来提升网络训练速度的。公众号后台回复“论文”即可获得下载链接！

2 动机

在摘要部分，作者一开始就提到“在整个网络的训练过程中，由于上一层网络参数的变化将导致输出层结果分布的改变，这就使得网络中每一层输入的分布均会发生改变，从而加大的网络的训练难度”。鉴于这一问题，作者就开始细数了在网络训练中由于输入分布（input distribution）的改变进而所带来的一系列问题。

扫码回复“加群”即可进入月来客栈交流群！

2.1 面临的问题

在论文第一部分的开始，作者就开始谈论到基于mini-batch的Stochastic Gradient Descent(SGD)在网络训练过程中的好处。首先，在使用mini-batch时我们能够调节mini-batch的大小，并且理论上随着mini-batch的增大，计算出来的梯度也就更加准确；其次，基于mini-batch的SGD比原始的SGD（随机抽取一个样本进行梯度计算）在计算效率上更高。**在这里，作者其实是在暗示mini-batch的好处，因为作者后面也会继续用到。**

紧接着作者说到，虽然SGD既简单也高效，但是它最大的一个缺点就是极容易受到超参数（如学习率）或者是参数初始化的影响。同时，在网络训练的过程中，当前网络层的输入会受到前面层（参数变化）的影响，并且随着网络的加深这种影响将会得到放大。这一影响的罪魁祸首就是前一层参数的变化将会导致该层输出值的分布（distribution）发生变化，当这些值被输入到下一层后，下一层的网络就需要再来适应学习这些新的分布。什么意思呢？

假设现在有如下这么一个网络：

$$\mathcal{L} = F_2(F_1(u, \Theta_1), \Theta_2) \tag{1}$$

其中F_1, F_2为任意的两个变换，u为原始的网络输出，Θ_1, Θ_2分别为两个网络层的参数。

现在我们的目的就是通过最小化$\mathcal{L}$来求得参数Θ_1, Θ_2的取值。此时，我们也可以将F_2的输入看成是$x = F_1(u, \Theta_1)$，那么根据式子(1)我们就有：

$$\mathcal{L} = F_2(x, \Theta_2) \tag{2}$$

接着根据式子(3)就可以完成Θ_2的迭代求解：

$$\Theta_2 \leftarrow \Theta_2 - \frac{\alpha}{m}\sum_{i=1}^{m}\frac{\partial F_2(x_i, \Theta_2)}{\partial \Theta_2} \tag{3}$$

但一个不争的事实就是，原始输入u的分布在经过网络层F_1之后会发生改变，而这也就意味着网络层F_2中的参数Θ_2就需要再来学习输入值x的分布。也就是说，尽管你一开始对原始

的输入u进行了标准化，但是再经历过一个网络层后它的分布就发生了改变，那么下一层又需要重新学习另外一种分布，这就意味着每一层其实都是在学习不同的分布。因此，**作者将这种由于网络参数发生变化而引起分布发生改变的现象称为网络的Internal Covariate Shift(ICS)问题。**

同时作者继续说到，尽管先前由于ISC导致的梯度消失问题能够通过ReLU激活函数、较小的学习率或者是反复的初始化来解决。但是，如果我们能够确保每一层网络输入的分布更稳定，那么这将会极大的提高网络的训练速度。

到此，对于ICS所造成的问题作者就算是给介绍完了。但这该如何解决呢？

2.2 解决思路

根据前面作者的叙述，我们其实也不难猜出，对于解决该问题的方法就是：**以mini-batch的方式来对每一层网络的输入进行标准化。**

在论文的第二部分里作者说到，想要降低ICS对于网络训练的影响，那就需要找到一种能够缓解ICS的方法，而这个方法就是在网络的训练过程中固定每一层输入的分布。同时，在[2]中LeCun等人也已经证明，通过对网络的输入进行白化（whitened）处理能够有效的提升网络的收敛速度。所谓白化指的就是通过某种变换使得数据的均值为0方差为1，同时依赖性较低。

因此作者就想到，既然白化处理能够有效提升网络的收敛速度，那要是对网络的每一层输入都进行白化处理，不就能够使得每一层输入的分布固定住吗？这样一来，ICS的问题不就可以得到缓解了吗？虽然理论上来说可以这样做，但是作者发现如果对每一层的输入都进行白化处理将会带来两个问题：计算量太大（每次白化都需要用到整个数据集）以及该目标函数并不是处处可导。

鉴于上述存在的问题，作者在白化操作的基础上进行了两点必要的简化，得到了论文中所提出来的Batch Normalization标准化方法：

> Since the full whitening of each layer's inputs is costly and not everywhere differentiable, we make two necessary simplifications.

第一，通过独立地对输出向量的每个值进行标准化（均值为0，方差为1）来代替白化操作中同时对输入和输出向量进行标准化的方法；

第二，通过基于mini-batch的方式来对网络进行训练，将每个batch的输出值计算得到的均值和方差用于对输出值进行标准化。

到这里，整个BN的总体思想就算是弄清楚了，接下来让我们来看看它到底是如何进行标准化的。

3 技术手段

3.1 独立的对每个维度进行标准化

假设现在有一个d维的网络层，其输出为$x = (x^{(1)}, x^{(2)}, \ldots, x^{(d)})$，那么对于每一个维度，我们就可以通过如下公式来进行标准化：

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}} \qquad (4)$$

其中，期望$E[x^{(k)}]$和方差$Var[x^{(k)}]$都是在整个 数据集上计算得到的。

但是，作者又说到，如果仅仅只是简单通过公式(4)来对每个维度进行标准化，那么在某些情况下将会改变该维度原本的表示信息。

> Note that simply normalizing each input of a layer may change what the layer can represent.

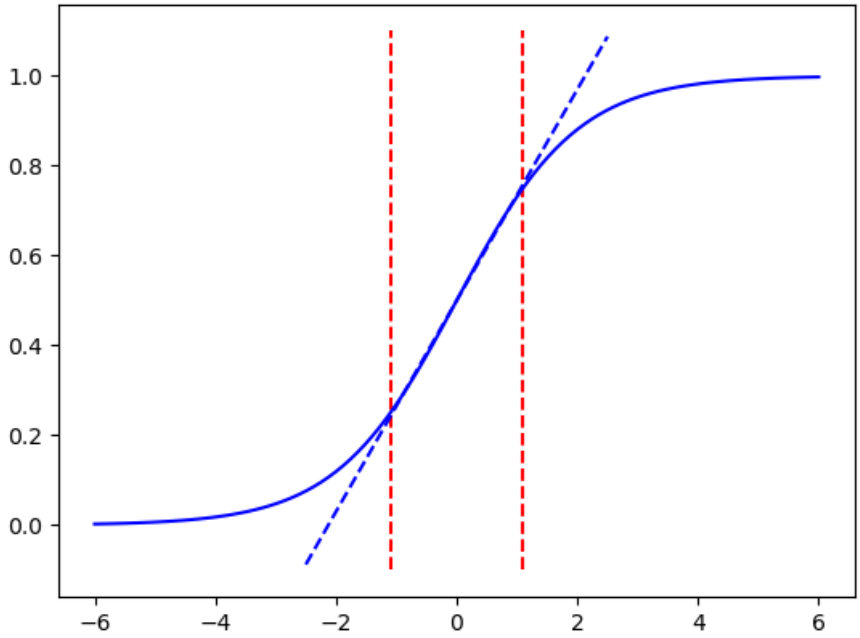

图 1. 非线性变成线性图

如图1所示，在对sigmoid激活函数的输入值进行标准化时，通过公式(4)标准化后的结果可能只会趋于0附近，从而把sigmoid变成了一个线性的激活函数。

为了解决这一问题，作者在公式(4)的基础上，加入了一组可学习的参数$\gamma^{(k)}$和$\beta^{(k)}$对$\hat{x}^{(k)}$进行了一次线性变换：

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)} \qquad (5)$$

其中$y^{(k)}$就是我们最后得到的标准化后的结果，而$\gamma^{(k)}$和$\beta^{(k)}$也会随着网络中的权重参数一起训练，当且仅当$\gamma^{(k)} = \sqrt{Var[x^{(k)}]}$，$\beta^{(k)} = E[x^{(k)}]$时，公式(5)就变成了恒等变换，也就相当于没有进行标准化（如果网络确实需要的话）。

3.2 标准化时采用mini-batch

由于计算机硬件的原因，我们不可能同时将所有的数据一次性输入到网络中进行训练，因此通常情况下都是喂入小批量的（mini-batch）的数据到网络中进行训练。所以，在这样的背景下，我们也不可能在使用BN的时候一次对所有的输出值进行标准化。因而，作者提出了第二个简化：以mini-batch的方式来进行标准化。

假设现在有一个大小为m的小批量数据$\mathcal{B}$，同时由于BN是**独立地对每个神经元的输出值进行标准化，这意味着每个神经元都有自己独立的参数**，因此我们这里以对第k个神经元$x^{(k)}$标准化为例进行介绍，并且进一步为了书写方便我们把k也暂时省略掉。此时，对于m个样本的输入，在第k个神经元就会有对应的m个输出：

$$\mathcal{B} = \{x_{1,2,...,m}\} \tag{6}$$

接着，我们将标准化后的结果记为$\hat{x}_{1,2,...,m}$，线性变换后的结果为$y_{1,2,...,m}$，我么就可以将整个BN的过程表示为：

$$BN_{\beta,\gamma} : x_{1,2,...,m} \rightarrow y_{1,2,...,m} \tag{7}$$

具体的，对于整个BN的详细过程如图2所示：

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
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.

图 2. BN算法流程图

其中$\mu_{\mathcal{B}}$为在小批量$\mathcal{B}$上对x_i期望的估计，$\sigma_{\mathcal{B}}^2$为对x_i方差的估计，而$\hat{x}_i$则表示标准化后的结果，y_i表示线性变换后的结果，也就是我们最后真正需要的结果。同时，为了防止方差为0的情况（numerical stability），在进行标准化时分母额外的加了一个很小的常数ϵ。这里需要说明的是，$\mu_{\mathcal{B}}$和$\sigma_{\mathcal{B}}^2$并不是整个数据集真实的期望与方差，而仅仅只是根据采样mini-batch估计得到的。

就这样，每一层的每个神经元的输出值都将会经历过图2所示的处理，使得均值为0方差为1，然后再输出到下一层网络中。尽管在这一个过程中可能会导致不同神经元之间的联合分布发生变换，但是这却使得每一层网络的输入具有了同样的均值与方差，进而加速了网络的训练过程。

同时，这里需要提到一点的是，作者在论文里特意提到，在用BN进行标准化时应该要使得$m > 1$，即mini-batch的大小要超过1，不然图2中对于均值和方差的估计就没有意义了。

3.3 训练与预测中的BN

根据上面内容的介绍，我们大致了解了BN在标准化时的算法流程。不过那只是最核心的部分，还有一些细节之处需要交代。根据图2的流程可以知道，BN中一共有五个参数：$\mu_{\mathcal{B}}, \sigma_{\mathcal{B}}^{w}, \epsilon, \gamma, \beta$，但是只有后两个参数才是随着网络一起训练（trainable），前两个参数是训练过程中用mini-batch中的样本估计得到的，用于对训练时的mini-bath进行标准化，而第三个参数则是自己预先设定的（例如1e-5)。那现在问题就来了，当网络训练完成时后三个参数算是有着落了，那前两个参数怎么办呢？采用什么值？

在论文的3.1小节中作者提到，虽然在训练过程使用mini-batch来估计均值和方差能够有效的提高网络的训练效率，但是这在预测推理时既不是必要的也并不是我们想要的。我们所想要的就是输出**仅仅只会完全**取决于输入。

> we want the output to depend only on the input, deterministically.

这句话什么意思呢？揣摩了很久，笔者认为作者想要表达的其实是“预测时均值和方差应该只取决于整个输入的训练集”。是不是还是有点朦胧的感觉？不着急，我们继续往下看。

接着作者说到，一旦整个网络训练完成后，我们就可以根据公式(8)来对输入数据进行标准化：

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}} \qquad (8)$$

此时式子(8)中使用到的$E[x^{(k)}]$和$Var[x^{(k)}]$是根据整个训练集（population statistics）而非mini-batch计算得到的真实期望与方差。进一步，我们就可以通过计算训练时mini-batch中对应统计量的无偏估计来得到$E[x^{(k)}]$和$Var[x^{(k)}]$。对这点不是特别清楚的朋友可以参考[3] [4] [5]这几篇回答。

下面作者继续说到，虽然通过无偏估计能够得到更加接近于数据集真实的期望与方差，但是这样得到的均值和期望可能会较大的偏离训练时通过mini-batch计算得到的均值与方差。换句话说就是，作者更希望的是用接近于训练过程中所产生的均值与方差来代替更接近于真实的结果，因为这样我们在预测过程中就能够最大程度上的达到训练时的精度。

> Using moving averages instead, we can track the accuracy of a model as it trains.

因此，可以采用以滑动平均的方式来计算预测过程中所需要用到的均值与方差：

$$\begin{aligned} moving\_mean &= momentum \times moving\_mean + (1.0 - momentum) \times mean \\ moving\_var &= momentum \times moving\_var + (1.0 - momentum) \times var \end{aligned} \tag{9}$$

其中$mean$和var指的就是在每个mini-batch上训练得到的均值与方差，通过调节系数momentum可以控制$moving_mean$和$moving_var$到底是想要接近于真实的值还是训练时的值。

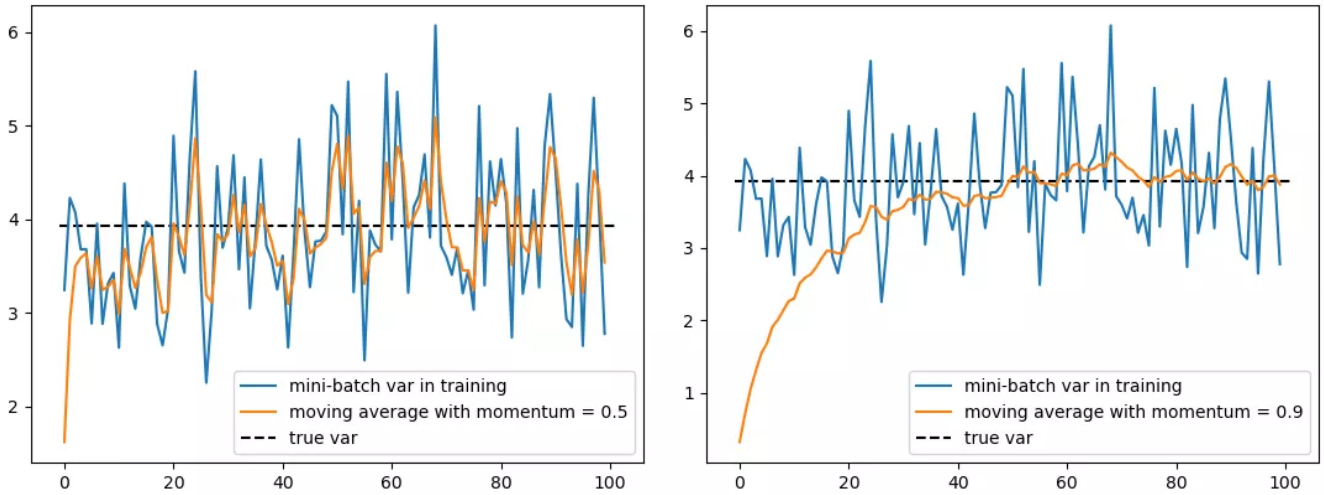

图 3. 真实方差与滑动平均方差图

如图3所示就是不同momentum取值下，训练中mini-batch计算得到的方差、滑动平均方差和真实方差的一个变化图。可以发现，momentum越小滑动平均计算得到的方差就会越靠近训练过程中的方差，momentum越大滑动平均计算得到的方差就会更靠近真正的方差。所以，一般情况下我们都会通过调节momentum来在两者间得到一个平衡。并且，笔者认为这可能也是论文作者选择滑动平均的理由之一。

在通过滑动平均的方法计算得到的均值和方差后，再结合训练得到γ和β就可以对预测过程中每一层的输出值进行标准化了。到此，我们就算是介绍完了整个BN的思想，以及在训练和测试过程中的计算流程。不过为了能够更加细微的了解这一过程，在后面我们还会通过实际的代码来进行介绍。下面，让我们接着继续来看BN在实际中的运用。

3.4 BN的实际运用

3.4.1 BN处处可导

在上面的内容中我们介绍到，在包含有白化操作的目标函数中，目标函数并不是处处可导的。但根据图2中BN的算法流程，我们可以得到如图4所示的在BN中各个参数的依赖关系图，从而求得损失$\mathcal{L}$关于各个参数的梯度。

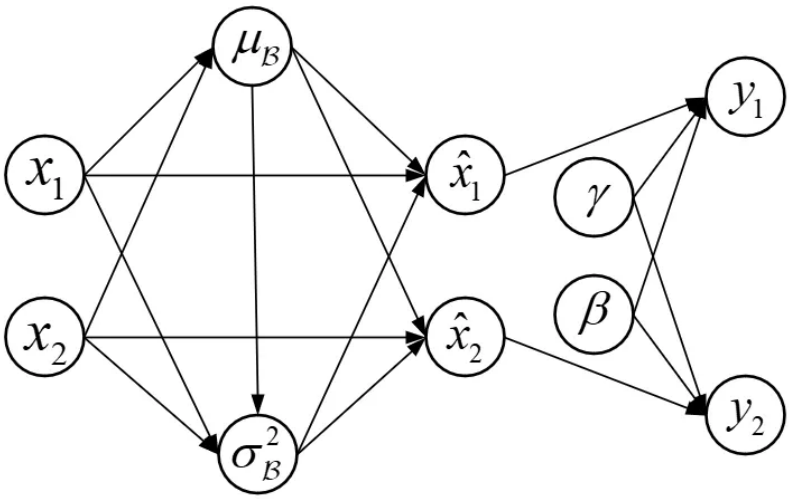

图 4. BN中各个参数的依赖关系图

有了这张清晰的关系依赖图，根据链式法则我们便能求得损失$\mathcal{L}$关于各个参数的梯度，其对应公式如下：

$$\frac{\partial \mathcal{L}}{\partial \hat{x}_i} = \frac{\partial \mathcal{L}}{\partial y_i} \cdot \gamma \tag{10}$$

$$\frac{\partial \mathcal{L}}{\partial \sigma_{\mathcal{B}}^2} = -\sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial \hat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{1}{2}(\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2} \tag{11}$$

$$\frac{\partial \mathcal{L}}{\partial \mu_{\mathcal{B}}} = \sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial \hat{x}_i} \cdot \frac{\partial \hat{x}_i}{\partial \mu_{\mathcal{B}}} = \sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial \hat{x}_i} \left[\frac{\partial \hat{x}_i}{\partial \mu_{\mathcal{B}}} + \frac{\partial \hat{x}_i}{\partial \sigma_{\mathcal{B}}^2} \frac{\partial \sigma_{\mathcal{B}}^2}{\partial \mu_{\mathcal{B}}} \right] \tag{12}$$

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

$$\frac{\partial \mathcal{L}}{\partial x_i} = \frac{\partial \mathcal{L}}{\partial \mu_B} \cdot \frac{\partial \mu_B}{\partial x_i} + \frac{\partial \mathcal{L}}{\partial \hat{x}_i} \cdot \frac{\partial \hat{x}_i}{\partial x_i} + \frac{\partial \mathcal{L}}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\partial \sigma_{\mathcal{B}}^2}{\partial x_i} \tag{13}$$

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

$$\frac{\partial \mathcal{L}}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial y_i} \cdot \hat{x}_i \, , \frac{\partial \mathcal{L}}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial y_i} \tag{14}$$

这里需要注意的是，由于方差和均值是通过在整个mini-batch上计算得到的，所以公式(11)(12)中会有一个求和符号，同时从图4的依赖关系也可以看出$\mu_{\mathcal{B}}$和$\sigma_{\mathcal{B}}^2$均依赖于$\hat{x}_1, \hat{x}_2$。

从以上公式可以看出，整个BN过程都是可导的，因此这也就保证了网络模型能够正常的按照设想进行学习，进而可以加快网络的训练速度。

3.4.2 BN的顺序

在论文的第3.2部分中作者继续提到，BN可以被用于网络中任意神经元的标准化，但是论文中暂时只对仿射变换（affine transformation）加非线性变换的过程进行讨论（说得直白点就是，在论文中作者只对在全连接网络和卷积网络中如何使用BN进行了说明，例如在LSTM中使用BN就没有介绍），即：

$$z = g(Wu + b) \tag{15}$$

其中$g(\cdot)$表示激活函数，如sigmoid或者是ReLU等。

作者继续说到，在这两种网络中将会把BN插入到非线性变换之前，即先对$x = Wu + b$进行标准化，然后再将其进行线性变换。此时作者说到，我们本应该对每一层的输入u进行标准化的，但由于u更像是上一层非线性变换后的输出，其分布的形状在训练过程中可能会发生变换，进而导致标准化无效的结果。

> We add the BN transform immediately before the nonlinearity, by normalizing $x = Wu + b$. We could have also normalized the layer inputs u, but since u is likely the output of another nonlinearity, the shape of its distribution is likely to change during training, and constraining its first and second moments would not eliminate the covariate shift.

这几句话什么意思呢？是不是没听明白到底作者想说什么？在这里，作者其实想要表达的是，非线性变换通常都会导致输入和输入之间分布的形状发生改变[6]（也就是产生不同的分布），如果是以BN+线性+非线性的顺序进行标准化，那么每次输入到激活函数中的分布就是不一样的，这样的话就达不到BN的初衷了（上面的first and second moments指的其实就是期望和方差）。

> In contrast, $Wu + b$ is more likely to have a symmetric, non-sparse distribution, that is "more Gaussian" (Hyv¨arinen & Oja, 2000); normalizing it is likely to produce activations with a stable distribution.

相反作者认为，$Wu + b$更接近于一个对称的高斯分布，如果是以线性+BN+非线性的顺序进行标准化，那么这就使得每次输入到激活函数中的值都具有同样的分布[7]，这样的话就更有利于网络的训练。

3.4.3 卷积中的BN

经过上述内容的介绍，我们大致清楚了如何在普通的前馈神经网络中运用BN，即以BN+线性+非线性的顺序对全连接层的中每一个神经元进行标准化。但如果是换到卷积操作中，还是这样来进行标准化吗？

对于卷积操作来说，作者为了能够使得BN遵循卷积特有的性质，所以将每个特征图中所有的神经元看作是一个整体来进行标准化。假设$\mathcal{B}$表示一个mini-batch中某一个特征图里所有的值，m, p, q分别表示mini-batch的大小，特征图的长和宽。那么此时我们将对这个小批量中的所有值（$m' = |\mathcal{B}| = m \cdot p \cdot q$个）以同一组均值、方差、$\gamma$和$\beta$进行标准化。也就是说，**在普通的前馈网络中BN是以每一个神经元为单位进行BN标准化，而在卷积中BN则是以每一个特征图为单位进行标准化。**

到这里，对于整个BN的思想与原理，以及具体的计算过程我们就算是介绍完了。由于文章篇幅有限，笔者将在下一篇文章中再来介绍BN的具体实现，以及通过对比实验来分析一下BN的优点。

4 总结

在这篇文章中，笔者首先在深度学习中训练网络时所面临的困难；然后介绍了一些传统的处理办法；接着介绍了BN算法的思想以及它的动机；最后详细的介绍了BN的原理、计算过程以及BN的实际运用等内容。

本次内容就到此结束，感谢您的阅读！如果你觉得上述内容对你有所帮助，**欢迎点赞或分享至一位你的朋友！**若有任何疑问与建议，请添加笔者微信'nulls8'或加群进行交流。**青山不改，绿水长流，我们月来客栈见！**

推荐阅读

[1]GoogLeNet介绍与实现

[2]厉害了，能把多尺度卷积说得这么高大上

[3]NiN一个放到现在也不过时的网络

[4]VGG一个可使用重复元素的网络

[5]LeNet5的继任者AlexNet网络模型

引用

[1] Sergey Ioffe, Christian Szegedy , Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. ICML, PMLR 37:448-456, 2015.

[2]LeCun et al, Neural Networks: Tricks of the trade. Springer, 1998b