Batch Normalization:2014

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift[paper]

Google

摘要：

每层数据的分布变化加剧了神经网络在训练时的困难程度，进而在调参的时候要使用更小的学习率和更注重参数的初始化，同时也使得饱和非线性模型更加难以训练。我们将这种现象称为内部协变量的偏移，通过对输入进行归一化操作来解决该问题。在使用Batch Normalization的情况下，网络可以使用更大的学习率，降低参数初始化在训练中的作用，也可以作为一种正则化的方式，来代替Drop out。通过将BN运用到图像识别的最优模型上，发现达到相同精度时，BN可以将训练时间缩短十四倍，并且有着极少的开销。通过使用BN，我们在ImageNet数据集上面最好的模型进行优化，最终达到了验证集4.9%top-5错误率测试集4.8%的错误率

论文出发点和背景

随机梯度下降算法和其变体对于网络训练时一种十分有效的方式，但是在调整超参数的时候需要十分小心，尤其是学习率和参数初始化。训练变难的原因就是网络中的每个隐藏层都会受到前层的参数影响，因而随着网络的加深，即使是很小的变化也会因为每层的参数传递过去而变得特别大。

层的输入数据分布不同是难训练的一个原因，因为网络需要不断去适应一个新的数据分布。当输入学习系统的分布产生变化时，存在协变量偏移问题，一般可以通过域自适应来解决。但是协变量偏移的概念也可以迁移到学习系统的各个子层。类比于训练集和测试集如果有着相同的分布，网络训练会变得更有效率，这一观点应用到自网络或者网络的子层上面也是成立的。这样需要学习的参数θ就不需要通过变化来补偿输入的分布变化。

以饱和的非线性激活函数sigmoid而言，当输入值增大时，其导数就会趋近于0。这种特性意味着网络只有对输入比较小的时候训练比较块，输入绝对只比较大的时候的梯度将会趋近于0，模型就会缓慢训练。由于某一层的输入决定于其权重矩和偏置向量以及之前所有层的一些参数，这些参数很容易使得加权后的x绝对值变大，进而网络进入了非线性的饱和状态，收敛速度减慢。随着网络深度增加，这种效应也就被放大了。

在此之前提出的解决方案：1.使用ReLU激活函数；2,较小的学习率，3，参数初始化方式的选择。

论文创新思路

通过批归一化的方式来消除网络中的内部协变量偏移问题，进而加速神经网络的训练。通过固定层输入的均值和方差来实现，同时也可以在反向传播中产生更大的梯度。直观上，我们可以通过固定输入的均值和方差以及去相关来加快网络的训练。

fect of the gradient step. For example, consider a layer with the input u that adds the learned bias b, and normalizes the result by subtracting the mean of the activation computed over the training data: $\widehat{x} = x - E[x]$ where $x = u + b$, $\mathcal{X} = \{x_{1...N}\}$ is the set of values of x over the training set, and $\mathrm{E}[x] = \frac{1}{N}\sum_{i=1}^{N} x_i$. If a gradient descent step ignores the dependence of $\mathrm{E}[x]$ on b, then it will update $b \leftarrow b + \Delta b$, where $\Delta b \propto -\partial\ell/\partial\widehat{x}$. Then $u + (b + \Delta b) - \mathrm{E}[u + (b + \Delta b)] = u + b - \mathrm{E}[u + b]$.

通过推导和实验证明，如果归一化参数在梯度下降步骤之外计算时，模型会爆炸。随着训练的继续，b将无期限地增长，但是损失总是保持不变

因此归一化方法应该对于任何参数值都会产生所需分布的激活值，而且应该寻找一个可微的方式对参数进行归一化，而不是每经过一次参数更新以后对整个训练集进行分析。

论文方法的大概介绍

如果对每层的输入进行full whitening 的操作，在计算上有着极大的消耗，同时并不是所有都可微。因而我们做了两个方面的简化。

1.不对层的输入输出进行处理，仅仅对层的特征图进行归一化处理，使其满足零均值，方差为1。计算使用的方差和均值由在训练集计算中得出。之前Lecun的研究表明，即使这种方法没有减小特征的相关性，但是也加速了收敛的速度。

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

简单的归一化方法可能会改变网络中每层的表示，为了让网络经过归一化操作后还能保持本来的信息，我们对每个激活值引入了两个参数，γ和β，其中γ是激活值的缩放因子，β是激活值的偏移因子。这些因子参数可以随着网络的训练而学习到，可以恢复网络的表示能力。

$$y^{(k)} = \gamma^{(k)}\widehat{x}^{(k)} + \beta^{(k)}.$$

整个流程是基于训练集的，但是在随机优化时，这样的方式是不切实际的。因此我们进行了第二次的方法优化，通过使用小批量的方式来进行训练，每个mini-batch都会对每一个激活值产生均值和方差的估计。这样在反向传播的时候归一化的统计值就能充分参与。小批量的时候使用的是每一维上的方差，而不是联合协方差。

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.

参数学习的链式法则如下所示：

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

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

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

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

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

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

对激活值的归一化依赖于网络训练时的mini-natch，但是在网络执行推理的时候是没有必要的，我们只希望输出只确定性地取决于输入。我们使用方差的无偏估计 $\mathrm{Var}[x] = \frac{m}{m-1} \cdot \mathrm{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$，

来作为推理过程的方差，其中m代表mini-batch的大小，σ_B^2 来自于mini batch中样本的统计，整个流程如下图所示：

Input: Network N with trainable parameters Θ;
subset of activations $\{x^{(k)}\}_{k=1}^{K}$
Output: Batch-normalized network for inference, $N_{\text{BN}}^{\text{inf}}$
1: $N_{\text{BN}}^{\text{tr}} \leftarrow N$ // Training BN network
2: **for** $k = 1 \ldots K$ **do**
3: Add transformation $y^{(k)} = \text{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to $N_{\text{BN}}^{\text{tr}}$ (Alg. 1)
4: Modify each layer in $N_{\text{BN}}^{\text{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead
5: **end for**
6: Train $N_{\text{BN}}^{\text{tr}}$ to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$
7: $N_{\text{BN}}^{\text{inf}} \leftarrow N_{\text{BN}}^{\text{tr}}$ // Inference BN network with frozen // parameters
8: **for** $k = 1 \ldots K$ **do**
9: // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$, etc.
10: Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

$$\text{E}[x] \leftarrow \text{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$
$$\text{Var}[x] \leftarrow \frac{m}{m-1}\text{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

11: In $N_{\text{BN}}^{\text{inf}}$, replace the transform $y = \text{BN}_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{\text{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\text{E}[x]}{\sqrt{\text{Var}[x]+\epsilon}}\right)$
12: **end for**

Algorithm 2: Training a Batch-Normalized Network

BN在卷积神经网络中的使用

对于神经网络中的每一层我们都可以表达为$z = g(Wu + b)$,W,b是可学习到的参数，g()是非线性的激活函数，考虑到对比归一化u,归一化$Wu + b$更可能产生稳定的期望的分布，因为对比之下，$Wu + b$更对称、稀疏。由于偏置在后期会被归一化以及会引入β因子进行调节，所以经过BN层的网络仿射变换可以记作$z = g(BN(Wu))$

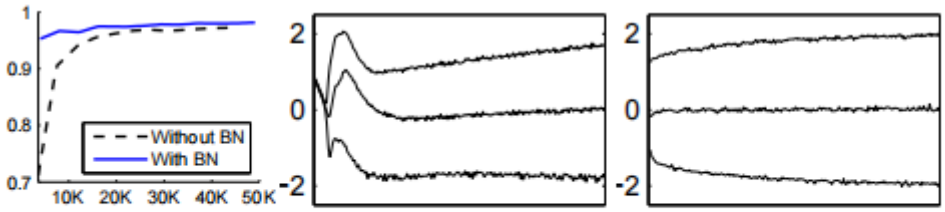
在卷积神经网络中，我们学习到的γ和β是针对每个特征图的，而不是针对每个激活值

实际效果

1.（摘要）：通过使用BN，我们在ImageNet数据集上面最好的模型进行优化，最终达到了验证集4.9%top-5错误率测试集4.8%的错误率

2.使用BN之后网络可以使用更大的学习率，且批归一化还有正则化的作用

3.在MNIST数据集上的效果：以28x28的二值图像作为输入，3个全连接的隐藏层，每个激活100个。每个隐层采用Sigmoid非线性计算y = g (W u + b)，权值W初始化为小随机高斯值。最后一个隐藏层是一个完全连接的层，有10个激活(每类一个)和交叉熵损失。我们对网络进行了50000步的训练，每小批60个例子
发现：带有BN的网络训练收敛更快，同时数据的分布也更加稳定



(a) (b) Without BN (c) With BN

Figure 1: (a) *The test accuracy of the MNIST network trained with and without Batch Normalization, vs. the number of training steps. Batch Normalization helps the network train faster and achieve higher accuracy.* (b, c) *The evolution of input distributions to a typical sigmoid, over the course of training, shown as* $\{15, 50, 85\}$*th percentiles. Batch Normalization makes the distribution more stable and reduces the internal covariate shift.*

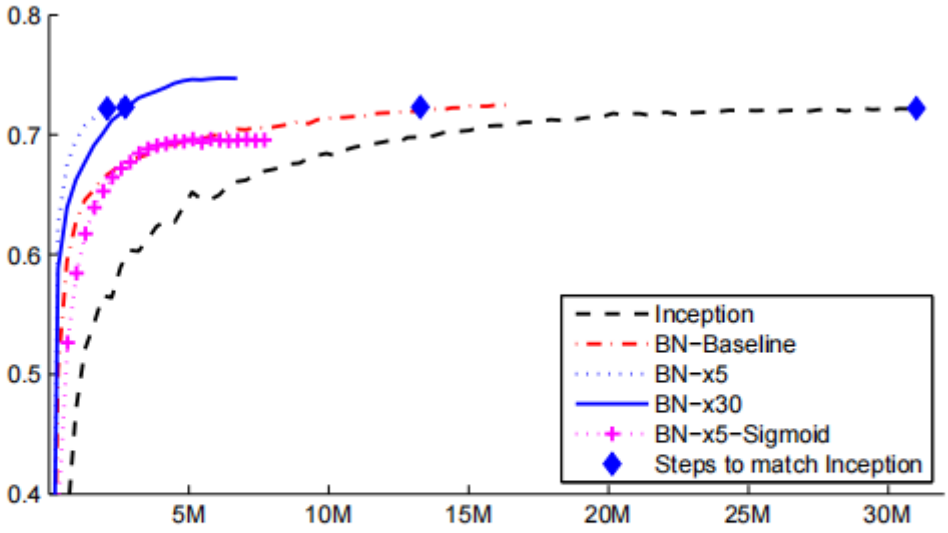


Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*

Model	Steps to 72.2%	Max accuracy
Inception	$31.0 \cdot 10^6$	72.2%
BN-Baseline	$13.3 \cdot 10^6$	72.7%
BN-x5	$2.1 \cdot 10^6$	73.0%
BN-x30	$2.7 \cdot 10^6$	74.8%
BN-x5-Sigmoid		69.8%

Figure 3: *For Inception and the batch-normalized variants, the number of training steps required to reach the maximum accuracy of Inception (72.2%), and the maximum accuracy achieved by the network.*

Model	Resolution	Crops	Models	Top-1 error	Top-5 error
GoogLeNet ensemble	224	144	7	-	6.67%
Deep Image low-res	256	-	1	-	7.96%
Deep Image high-res	512	-	1	24.88	7.42%
Deep Image ensemble	variable	-	-	-	5.98%
BN-Inception single crop	224	1	1	25.2%	7.82%
BN-Inception multicrop	224	144	1	21.99%	5.82%
BN-Inception ensemble	224	144	6	20.1%	**4.9%***

Figure 4: *Batch-Normalized Inception comparison with previous state of the art on the provided validation set comprising 50000 images. *BN-Inception ensemble has reached 4.82% top-5 error on the 100000 images of the test set of the ImageNet as reported by the test server.*

个人理解

1.对于饱和的非线性函数例如sigmoid，只有[-5,5]区间内的梯度很大，一旦超过这个区间范围内，其导数值偏近于0，导致网络学习变慢，收敛变难，也会导致一些梯度弥散的情况发生，一般来讲在神经网络和机器学习中，数据是决定模型的重要因素，比如今年的数学建模国赛C题，就要求应该先对数据进行对数归一化处理，而数据保持相同的分布就是神经网络所需要的，这样可以使得参数在学习时不必花费时间去弥补分布不同导致的误差，这样也就使得网络参数的学习变得比之前容易。

2.对于γ和β的存在理解：仅仅对数据进行归一化成均值为0，方差为1的数据不足以表征真实数据的分布，比如归一化前为500，归一化后为0.5，为了让归一化后的数据能够具有表征归一化前数据的能力，我们通过引入缩放因子γ和偏移因子β，通过梯度下降学习参数，来达到更好的还原效果。

3.BN在训练和推理时的差异，记得高中数学老师说统计学的核心思想就是样本估计总体，神经网络就是通过学习样本中的特征，然后对其他的样本进行特征上的匹配，然后做出判断。BN也是类似的道理，在训练过程中是通过梯度下降学习γ和β的值，在推理时就需要用到训练样本中方差和均值的无偏估计通过 $y = \frac{\gamma}{\sqrt{\mathrm{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x]+\epsilon}}\right)$ 得到推理网络中的BN结果。