批归一化: 通过减少内部协变量偏移来加速深度网络训练

Sergey Ioffe, Christian Szegedy

摘要

在训练中对于每一层，由于它之前层的参数会变化，所以它的输入的分布也会变化，这导致训练深度神经网络十分复杂。这个现象通过要求更低的学习率以及精心的参数初始化，降低了训练速度，并使得训练使用饱和非线性的模型非常困难。我们将这个现象成为*内部协变量偏移*，并通过归一化每一层的输入来解决它。我们的方法从将归一化作为模型结构的一部分汲取力量并对*每一个训练迷你批*进行归一化。批归一化使得我们可以使用更大的学习率并不用那么关心初始化。它也可以减少过拟合，在某些情况下消除了对于 Dropout 的需要。将批归一化应用在最好的图像分类模型上，模型达到与原来相同准确率的时间减少了 14 倍，并最终以很大幅度击败了原有模型。使用集成的进行了批归一化的网络，我们进一步提高了最好的 ImageNet 分类结果：到达了 4.9% 的 top-5 验证错误率（以及 4.8% 的测试错误率），超过了人类的准确率。

1 简介

深度学习极大地推动了视觉、语音和其他领域的发展。随机梯度下降（Stochastic gradient descent, SGD），已经被证明为是一种训练深度网络的有效方式。为了达到最好的性能，例如 momentum[**momentum**] 和 Adagrad[**adagrad**] 的 SGD 变种也会被使用。SGD 会优化网络的参数 Θ 来最小化损失

$$\Theta = \arg\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(\mathrm{x}_i, \Theta)$$

其中 $\mathrm{x}_{1\ldots N}$ 是训练数据集。使用 SGD，训练分步进行，在每一步中我们考虑大小为 m 的迷你批 $\mathrm{x}_{1\ldots m}$。我们使用通过迷你批计算的

$$\frac{1}{m}\frac{\partial \ell(\mathrm{x}_i, \Theta)}{\partial \Theta}$$

来近似损失函数对于参数的梯度。使用多个样例的迷你批，与每次仅使用一个样例相比，在以下几个方面会有帮助。首先，损失函数在迷你批的梯度是损失函数在训练集的梯度的估计，它的质量会随着批大小的增加而提高。其次，由于现代计算平台提供的并行，在一个批上的计算会比为单独的样例做 m 次计算更加高效。

虽然随机梯度是简单且有效的，但是它要求对模型超参数的精心调整，尤其是在优化中使用的学习率和模型参数的初始值。由于每一层的输入都会被前面所有层的参数影响，所以网络参数的微小变换会随着网络变深而被放大，这使训练变得复杂。

改变某一层的输入分布会使得这一层需要持续适应新的分布，这实际上是一个问题。当一个学习系统的输入分布变换时，我们称它经历了协变量偏移 [**covariateshift**]。这通常通过领域适应处理。然而，协变量偏移的概念可以扩展到整个学习系统之外，应用于其部分，例如子网络或某一层。考虑一个计算

$$\ell = F_2(F_1(\mathrm{u}, \Theta_1), \Theta_2)$$

的网络，其中 F_1 和 F_2 是任意的变换，通过优化参数 Θ_1, Θ_2 来最小化损失 ℓ。学习 Θ_2 可以被视为输入 $\mathrm{x} = \mathrm{F}_1(\mathrm{u}, \Theta_1)$ 被送入子网络

$$\ell = \mathrm{F}_2(\mathrm{x}, \Theta_2)$$

例如，一步梯度下降（批大小为 m，学习率为 α）

$$\Theta_2 \leftarrow \Theta_2 - \frac{\alpha}{m}\sum_{i=1}^{m}\frac{\partial \mathrm{F}_2\left(\mathrm{x}_i, \Theta_2\right)}{\Theta_2}$$

与为单独的 F_2 网络输入 x 是完全等价的。因此，使得训练更加高效的输入分布性质——例如训练数据和测试数据有相同的分布——对于训练子网络也是适用的。因此，随着时间的推移，保持 x 的分布固定是有利的。此后，Θ_2 不必为弥补 x 分布变换而从新调整。

固定子网络输入分布对于子网络之外的层也有积极影响。考虑某一层使用 $\mathrm{sigmoidz} = g(W\mathrm{u}+\mathrm{b})$ 作为激活函数，其中 u 是这一层的输入，参数矩阵

W 和偏差向量 b 是这一层将要学习的参数，$g(x) = \frac{1}{1+\exp(-x)}$。随着 $|x|$ 变大，$g'(x)$ 会趋向于 0。这意味这对于 $\mathrm{x} = W\mathrm{u} + \mathrm{b}$ 来说，除绝对值较小的维度之外的所有维度，u的梯度将会消失，进而导致模型训练缓慢。然而，由于 x 会被 W，b 和之前所有层的参数影响，所以在训练中改变这些参数将会很可能将 x 的很多维度移动到非线性的饱和区并减缓收敛。这个效应会随着网络深度的提高而被放大。在实际中，饱和问题以及它所导致的梯度消失通常通过使用修正线性单元（Rectified Linear Units）[**relu**]$\mathrm{ReLU}(x) = \max(x, 0)$，精心初始化和小学习率来解决。然而，如果我们可以在网络训练时保证非线性的输入分布保持稳定，那么优化器就更可能不被困在饱和区，进而训练也会更快。

我们将训练过程中深度网络内部节点分布的变换成为*协变量偏移*。消除它将会使得训练更快。我们提出了一种名为批归一化的机制，它朝减少内部协变量偏移迈出了一步，这样做显著加速了深度神经网络的训练。它通过归一化步骤来实现，归一化将会固定每一层输入的均值和方差。批归一化通过减少梯度对于参数尺度或者它们的初始值的依赖，对于梯度在网络流动也有好处。这使得我们可以使用更高的学习率，同时不会导致发散。不仅如此，批归一化减少了模型过拟合的风险并减少了对于 Dropout[**dropout**] 的需要。最后，批归一化通过防止网络被困在饱和区，使得使用饱和的非线性成为可能。

在 4.2 节中，我们为性能最好的 ImageNet 分类网络应用了批归一化，并显示了我们可以仅使用 7% 的训练步骤来达到它原有的性能，并最终显著超过了它原有的准确率。使用一个集成了以批归一化训练的这样的模型，我们达到了在 ImageNet 分类任务上的已知最好 top-5 错误率。

2 减少内部协变量偏移

我们将*内部协变量偏移*定义为训练中由于网络参数的变换导致的网络激活分布的变换。为了改善训练，我们想要减少内部协变量偏移。随着训练的进行，通过固定每一层输入 x 的分布，我们期待可以提高训练速度。众所周知，洗白网络的输入，也就是通过线性转化使它的均值为零方差为一并去相关，可以使得网络收敛更快。由于每一层观测到的输入是由之前层产生的，所以对每一层的输入进行相同的洗白应该是有利的。通过洗白每一层的输入，我们应该可以向实现输入分布固定更进一步，这可以去除内部协变量

偏移带来的负面影响。

我们可以考虑通过直接修改网络或者改变优化算法的参数来依赖网络激活值，来在每一步训练或某个间隔中来洗白激活。然而，如果这些修改被穿插在优化步骤中，这时梯度下降步骤可能会以要求更新归一化的方式进行，这会减少梯度步骤的作用。例如，考虑某一个输入为 u 的层，它添加了要被学习的偏差 b，并通过减去在训练数据上计算得到的激活的均值来进行归一化：$\hat{x} = x - E[x]$ 其中 $x = u + b$，$\chi = \{x_1 \dots N\}$ 是训练集中 x 的集合，$\mathrm{E}[x] = \frac{1}{N}\sum_{i=1}^{N} x_i$。如果梯度下降步骤忽略 $\mathrm{E}[x]$ 对 b 的依赖，那么它将会以 $b \leftarrow b + \Delta b$ 的方式更新，其中 $\Delta b \propto \partial \ell / \partial \hat{x}$。这时 $u + (b + \Delta b) - \mathrm{E}[u + (b + \Delta b)] = u + b - \mathrm{E}[u + b]$。因此，对 b 的更新和后续在归一化中改变的结合导致层的输出没有改变，于是，损失也没有改变。随着训练继续，b 将会在损失固定的情况下无限增长。如果归一化不仅中心化还缩放激活，这个问题会变得更糟。我们在最初的实验中经验性地观察到了这个情况，当归一化参数在梯度下降步骤之外计算时，模型会爆炸。

上述方法的问题在于梯度下降优化没有考虑发生了归一化。为了解决这个问题，我们想要确保对于任意的参数值，网络总是产生具有所需分布的激活。这样做将会允许损失相对于模型参数的梯度考虑归一化，以及它对于模型参数 Θ 的依赖。假设 x 是某一层的输入，被当做一个向量，χ 是训练集中这些输入的集合。这时归一化可以被写作变换

$$\hat{\mathrm{x}} = \mathrm{Norm}(\mathrm{x}, \chi)$$

，它不仅依赖于给定训练样本 x，还依赖于所有样本 χ——如果 x 是由别的层产生的，那么每个 x 都依赖于 Θ。对于反向传播，我们需要计算 Jacobians

$$\frac{\partial \mathrm{Norm}(\mathrm{x}, \chi)}{\partial \mathrm{x}} 和 \frac{\partial \mathrm{Norm}(\mathrm{x}, \chi)}{\partial \chi};$$

忽略后项将会导致上述的爆炸。在这个框架中，由于它要求计算协方差矩阵 $\mathrm{Conv}[\mathrm{x}] = \mathrm{E}_{\mathrm{x}\in\chi}\left[\mathrm{x}\mathrm{x}^{\top}\right] - \mathrm{E}[\mathrm{x}]\,\mathrm{E}[\mathrm{x}]^{\top}$ 和它的逆平方根，来产生洗白的激活 $\mathrm{Cov}[\mathrm{x}]^{-1/2}(\mathrm{x} - \mathrm{E}[\mathrm{x}])$，以及为了反向传播计算这些变换的导数，所以洗白层输入是昂贵的。这激励我们寻找一种替代品，它要通过一种可微同时在每次参数更新后不要求分析整个训练集的方式来进行输入归一化。

一些以前的方法使用在单个训练样例上或者，在图片网络的情况下，不同特征图中的给定位置计算得到的数据。然而，这会通过丢弃激活的绝对尺度改变网络的表征能力。我们希望通过相对于整个训练数据的数据对训练样例的激活进行归一化来保留网络中的信息。

3 通过迷你批数据进行归一化

由于每一层输入的完整洗白是大开销且不是处处可导的，所以我们做了两个必要的简化。第一点简化是我们独立归一化每一个标量特征，也就是使得它均值为零方差为一，而不是同时洗白层输入和输出的特征。对于有着 d 维输入 $\mathrm{x} = \left(x^{(1)} \dots x^{(d)}\right)$ 的层，我们会按照

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

归一化每个维度,其中期望和方差是在训练数据集上计算的。正如 [**backprop**] 中所讲的，这样的归一化可以加速收敛，即使特征并没有被去关联。

注意简单地归一化某一层的每一个输入可能会改变这一层可以表征的东西。例如，归一化 sigmoid 的输入将会使它们被限制在非线性的线性区域。为了解决这个问题，我们确保插入网络中的变换可以表示恒等变换。为了实现这个想法，我们为每一个激活 $x^{(k)}$ 引入了一对参数 $\gamma^{(k)}, \beta^{(k)}$，它们将会缩放和移动归一化后的值：

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

这些参数也会随着原有模型的参数被学习，并修复网络的表征能力。实际上，如果原有激活就是最优的，那么我们可以通过设置 $\gamma^{(k)} = \sqrt{\mathrm{Var}\left[x^{(k)}\right]}$ 和 $\beta^{(k)} = \mathrm{E}\left[x^{(k)}\right]$ 来恢复它。

如果每一步训练都使用整个训练集，那么我们可以使用整个集合来为激活进行归一化。然而，当使用随机优化时这是不切实际的。因此，我们做了第二个简化：由于我们在随机梯度下降中使用迷你批，所以每个迷你批将会产生激活的均值和方差的估计。这样，归一化中使用的数据可以全面参与到梯度反向传播中。注意迷你批的使用是通过计算每一个维度的变量而不是联合协变量使能的；在联合的情况下，由于迷你批的数量很可能比要洗白的激活数量小，导致奇异的协方差矩阵，所以需要正则化。

考虑大小为 m 的迷你批。由于正则化是独立应用在每个激活上的，所以让我们专注于特定激活 $x^{(k)}$ 并为了清晰去掉 k。我们在这个迷你批中有 m 个激活值，

$$\mathcal{B} = \{x_{1\dots m}\}.$$

记归一化后的值为 $\hat{x}_{1\dots m}$，记它们的线性变换为 $y_{1\dots m}$。我们将变换

$$\mathrm{BN}_{\gamma,\beta} : x_{1\dots m} \to y_{1\dots m}$$

称为 *Batch Normalizing Transform*。我们在算法1中展示了 BN 变换。在算法中，ϵ 是为了数值稳定性加在迷你批方差上的常量。

Algorithm 1: 应用在迷你批的激活 x 上的 Batch Normalizing 变换

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

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

$\hat{x}_i \leftarrow \frac{x_i-\mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2+\epsilon}}$ //nomalize

$y_i \leftarrow \gamma\hat{x}_i+\beta \equiv \mathrm{BN}_{\gamma,\beta}\left(x_i\right)$ // sclae and shift

BN 转化可以被加到网络中来对任意激活进行操作。在 $y=\mathrm{BN}_{\gamma,\beta}(x)$ 中，参数 γ 和 β 表示被学习的参数，但是应该被注意的是，BN 变换并不是独立处理每个训练样本的激活。与之相反，$\mathrm{BN}_{\gamma,\beta}(x)$ 依赖于这个训练样本和迷你批中的其他样本。缩放偏移后的值 y 将被传至网络的其他层。归一化后的激活 $\hat{x}$ 是变换的内部变量，但是它们的表示是重要的。只要每个迷你批的元素采样自相同的分布，如果我们忽略 ϵ，那么 $\hat{x}$ 的分布应该期望为 0 方差为 1。这可以通过观察 $\sum_{i=1}^{m}\hat{x}_i=0$ 以及 $\frac{1}{m}\sum_{i=1}^{m}\hat{x}_i^2=1$ 得到。每个正则化后的 $\hat{x}^k$ 可以被视为一个子网络的输入，这个子网络由一个线性变换 $y^{(k)}=\beta^{(k)}\hat{x}^{(k)}+\beta^{(k)}$ 以及原有网络完成的其他处理组成。这些子网络的输入都有固定的均值和方差，虽然这些归一化后的 $\hat{x}^{(k)}$ 的联合分布会在训练的过程中变化，但是我们期望引入对输入的正则化可以加速子网络的训练，进而，加速整个网络的训练。

在训练中我们需要将损失 ℓ 的梯度通过变换反向传播，同时也要计算对于 BN 变换中参数的梯度。我们使用如下的链式法则（简化前）：因此，BN 变换是一种将正则化激活引入网络的可微变换。这保证了随着模型的训练，层可以持续学习引入更少内部协变量偏移的输入分布，从而加速训练。不仅如此，习得的仿射变换允许 BN 变换表征恒等变换来保留网络的能力。

3.1 批归一化的网络上的训练和推理

为了对一个网络进行批归一化，我们指定激活的一个子集并根据算法1为其中的每一个激活插入 BN 变换。任何之前接收 x 作为输入的层现在接收 $\mathrm{BN}(x)$ 作为输入。应用了批归一化的模型可以使用批梯度下降或者迷你批大小 $m>1$ 的随机梯度下降或者它的任意变种，例如 Adagrad[**adagrad**] 进行训练。依赖迷你批的激活归一化使得训练更加高效，但

是在推理中，它既不是必要的，也不是想要的；我们希望输出仅仅依赖于输入，并且每次的结果都相同。出于这个想法，一旦网络完成训练，我们将使用总体而非小批量统计数据进行归一化：

$$\hat{x} = \frac{x - \mathrm{E}\left[x\right]}{\sqrt{\mathrm{Var}\left[x\right] + \epsilon}}.$$

忽略 ϵ，正如训练中，这些归一化后的激活有相同的零均值单位方差。我们使用无偏估计 $\mathrm{Var}\left[x\right] = \frac{m}{m-1} \cdot \mathrm{E}_{\mathcal{B}}\left[\sigma_{\mathcal{B}}^2\right]$，其中期望是训练中大小为 m 的迷你批上得到的，$\sigma_{\mathcal{B}}^2$ 则是它们的方差。使用移动平均值，我们可以随着模型的训练追踪它的准确率。由于均值和方差在推理中是固定的，所以归一化仅仅是应用在每个激活上的线性变换。为了得到替换 $\mathrm{BN}\left(x\right)$ 的单一线性变换，可能包括通过 γ 缩放以及通过 β 偏移。算法2总结了训练归一化网络的过程。

Algorithm 2: 应用在迷你批的激活 x 上的 Batch Normalizing 变换

Input: Network N with trainable parameters Θ; subset of activations $\left\{x^{(k)}\right\}_{k=1}^{K}$

Output: Batch-normalized network for inference, $N_{\mathrm{BN}}^{\mathrm{inf}}$

$N_{\mathrm{BN}}^{\mathrm{tr}} \leftarrow N$ // Training BN network

for $k = 1 \ldots K$ **do**

  Add transformation $y^{(k)} = \mathrm{BN}_{\gamma^{(k)},\beta^{(k)}}\left(x^{(k)}\right)$ to $N_{\mathrm{BN}}^{\mathrm{tr}}$

  Modify each layer in $N_{\mathrm{BN}}^{\mathrm{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instaed

Train $N_{\mathrm{BN}}^{\mathrm{tr}}$ to optimize the parameters $\Theta \cup \left\{\gamma^{(k)}, \beta^{(k)}\right\}_{k=1}^{K}$

$N_{\mathrm{BN}}^{\mathrm{inf}} \leftarrow N_{\mathrm{BN}}^{\mathrm{tr}}$ // Inference BN network with frozen parameters

for $k = 1 \ldots K$ **do**

  //For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu \equiv \mu^{(k)}$, etc.

  Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

$$\begin{aligned} \mathrm{E}[x] &\leftarrow \mathrm{E}_{\mathcal{B}}\left[\mu_{\mathcal{B}}\right] \\ \mathrm{Var}[x] &\leftarrow \frac{m}{m-1}\mathrm{E}_{\mathcal{B}}\left[\sigma_{\mathcal{B}}^{2}\right] \end{aligned}$$

In $N_{\mathrm{BN}}^{\mathrm{inf}}$, replace the transform $y = \mathrm{BN}_{\gamma,\beta}(x)$ with

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

3.2 批归一化的卷积网络

批归一化可以应用于网络的任意激活集合上。这里，我们专注于由仿射变换紧接一个元素非线性的变换，其中非线性为：

$$\mathrm{z} = g\left(W\mathrm{u} + \mathrm{b}\right)$$

其中 W 和 b 是模型习得的参数，$g(\cdot)$ 则是例如 sigmoid 或 ReLU 的非线性。这种提法既包括全连接层也包括卷积层。我们在非线性之前添加 BN 变换，也就是归一化 $\mathrm{x} = W\mathrm{u} + \mathrm{b}$。我们也可以归一化层输入 u，但是由于 u 很可能是其他的非线性的输出，在训练中它的分布的形状很可能改变，同时

约束其一阶和二阶矩不会消除协变量偏移。与之对比的是，$W\mathrm{u}+\mathrm{b}$ 更可能有一个对称的，不稀疏的分布，也就是“更加高斯”；归一化它更有可能产生一个有着稳定分布的激活。

注意，由于我们归一化 $W\mathrm{u}+\mathrm{b}$，偏移 b 的作用将会经由后续的减去均值而取消（偏移的作用将会归入算法1的 β 中）。因此，$\mathrm{z}=W\mathrm{u}+\mathrm{b}$ 被替换为

$$\mathrm{z}=g\left(\mathrm{BN}\left(W\mathrm{u}\right)\right)$$

其中 BN 变换被独立应用于 $\mathrm{x}=W\mathrm{u}$ 的每一个维度，每一个维度都有单独的习得的

对于卷积层，我们额外想要归一化遵循卷积的性质——这样同一特征图不同位置的不同元素通过同样的方式归一化。为了实现这一点，我们同时归一化迷你批中所有位置的所有激活。在算法1中，我们让 $\mathcal{B}$ 为跨越迷你批中所有元素以及所有空间位置的所有值的集合——所以对于大小为 m 的迷你批以及大小为 $p\times q$ 的特征图，我们使用有效的迷你批大小为 $m'=|\mathcal{B}|=m\cdot pq$。我们为每一个特征图学习一对参数 $\gamma^{(k)}$ 和 $\beta^{(k)}$，而不是为每一个激活。算法2也做类似的修改，这样在推理中 BN 变换会对给定特征图的每一个激活进行同样的线性变换。

3.3 批归一化使能更高的学习率

在传统的深度网络中，过高的学习率可能导致梯度爆炸或梯度消失，同时也会被困在局部最小值点。批归一化会帮助解决这个问题。通过归一化整个网络的激活，它防止了参数的微小变化被放大以及激活梯度的次优改变；例如，它会防止训练被困在非线性的饱和区。

批归一化也是的训练对于参数尺度更加灵活。通常，大的学习率可能增加参数的尺度，这会在反向传播中放大梯度并导致模型爆炸。然而，有了批归一化，反向传播通过一层时不会被它的参数尺度影响。实际上，对于某个标量 a，

$$\mathrm{BN}\left(W\mathrm{u}\right)=\mathrm{BN}\left(aW\mathrm{u}\right)$$

并且我们可以展示

$$\begin{aligned}\frac{\partial\mathrm{BN}\left(aW\mathrm{u}\right)}{\partial\mathrm{u}} &= \frac{\mathrm{BN}\left(W\mathrm{u}\right)}{\partial\mathrm{u}}\\ \frac{\partial\mathrm{BN}\left(aW\mathrm{u}\right)}{\partial aW} &= \frac{1}{a}\frac{\partial\mathrm{BN}\left(W\mathrm{u}\right)}{\partial W}\end{aligned}$$

尺寸不会影响层的 Jacobian 或者梯度的传播。不仅如此，更大的权重导致更小的梯度，批归一化将会平稳参数的增长。

我们进一步推测批归一化可能引导层的 Jacobians 的特征值接近 1，这对于训练是有好处的。

3.4 批归一化正则化了模型