雄关漫道真如铁，而今迈步从头越。

朴素贝叶斯

深度学习中的逐层归一化

目录

概述

逐层归一化(Layer-wise Normalization)是将**传统机器学习中的数据归一化方法**应用到**深度**神经网络中，**对神经网络中隐藏层的输入进行归一化， 从而使得网络更容易训练**.

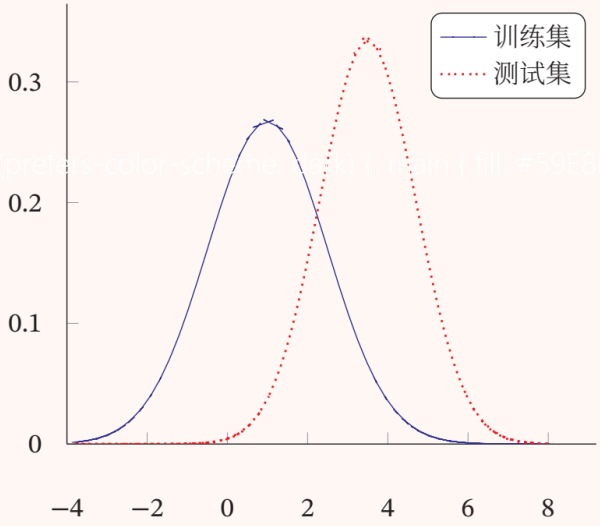
> **注**：这里的逐层归一化方法是指可以应用在深度神经网络中的任何一个中间层． **实际上并不需要对所有层进行归一化**。

逐层归一化**可以有效提高训练效率**的原因有以下几个方面：

- **更好的尺度不变性**：
  - 在深度神经网络中， 一个神经层的输入是之前神经层的输出.
  - 给定一个神经层 l， **它之前的**神经层$(1,\cdots,l-1)$**的参数变化**会导致**其输入的分布**发生较大的改变.
  - 当使用随机梯度下降来训练网络时，每次参数更新都会导致**该**神经层的输入分布发生改变.
  - **越高的层，其输入分布会改变得越明显**．就像一栋高楼，低楼层发生一个较小的**偏移**，可能会导致高楼层较大的偏移.
  - 从机器学习角度来看，**如果一个神经层的输入分布发生了改变，那么其参数需要重新学习，这种现象叫作内部协变量偏移(Internal Covariate Shift)**.
  - 为了缓解这个问题，我们可以**对每一个神经层的输入进行归一化操作，使其分布保持稳定**.

机器学习小知识 | 协变量偏移

协变量是一个统计学概念，是可能影响预测结果的统计变量. 在机器学习中，协变量可以看作输入. 一般的机器学习算法都要求输入在训练集和测试集上的分布是相似的. 协变量偏移（Covariate Shift）一般指输入在训练集和测试集上的分布不同. 这样，在训练集上学习到的模型在测试集上的表现会比较差.



- 把每个神经层的输入分布都归一化为标准正态分布，可以使得每个神经层**对其输入具有更好的尺度不变性**. **不论低层的参数如何变化，高层的输入保持相对稳定**.
- 另外，尺度不变性可以使得我们**更加高效地进行参数初始化以及超参选择**.

- **更平滑的优化地形**:
  - 逐层归一化一方面**可以使得大部分神经层的输入处于不饱和区域，从而让梯度变大,避免梯度消失问题**;
  - 另一方面**还可以使得神经网络的优化地形(Optimization Landscape) 更加平滑**，以及**使梯度变得更加稳定**，从而**允许我们使用更大的学习率， 并提高收敛速度** [Bjorck et al., 2018;Santurkar et al., 2018]

比较常用的逐层归一化方法有**批量归一化**、**层归一化**、**权重归一化**和**局部响应归一化**。本文中我们主要介绍一下批量归一化和层归一化。

批量归一化

批量归一化(Batch Normalization，BN）方法[Ioffe et al., 2015]是一种有效的逐层归一化方法，可以对神经网络中任意的中间层进行归一化操作.

- 批量归一化的**提出动机是为了解决内部协方差偏移问题**,
- 但**后来的研究者发现**其主要优点是**归一化会导致更平滑的优化地形** [Santurkar et al.,2018].

对于一个深度神经网络，令第l层的**净输入为**$z^{(l)}$， 神经元的**输出为**$a^{(l)}$，即

$$\boldsymbol{a}^{(l)} = f(\boldsymbol{z}^{(l)}) = f(\boldsymbol{W}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}),$$

其中$f(\cdot)$是激活函数， W和b是可学习的参数.

为了提高优化效率，就要使得净输入$z^{(l)}$的分布一致，比如都归一化到标准正态分布．**虽然归一化操作也可以应用在输入$a^{(l-1)}$上，但归一化$z^{(l)}$更加有利于优化**．因此，**在实践中归一化操作一般应用在仿射变换（Affine Transforma-tion）$Wa^{(l-1)}+b$之后、激活函数之前**.

注：个人看法，应该是一层的输入是上一层的输出，我们对一层的输入，也就是上一层的输出进行的归一化操作，不在这一层输入时进行，也不在紧接着上一层输出后进行，而是在上一层的即将得出输出的激活函数之前进行，这时上一层的权重参数和数据之间的计算已经完成了，即将得出输出，在激活函数之前进行归一化，可以使得处于激活函数的不饱和区域。

利用第7.4节中介绍的数据预处理方法对$z^{(l)}$进行归一化，相当于每一层都进行一次数据预处理，从而加速收敛速度．**但是逐层归一化需要在中间层进行操作，要求效率比较高，因此复杂度比较高的白化方法就不太合适．为了提高归一化效率，一般使用标准化将净输入$z^{(l)}$的每一维都归一到标准正态分布**.

$$\hat{\boldsymbol{z}}^{(l)} = \frac{\boldsymbol{z}^{(l)} - \mathbb{E}[\boldsymbol{z}^{(l)}]}{\sqrt{\text{var}(\boldsymbol{z}^{(l)}) + \epsilon}},$$

其中ϵ是为了保持数值稳定性而设置的非常小的常数.$\mathrm{E}[z^{(l)}]$和$\mathrm{var}(z^{(l)})$是指当前参数下，$z^{(l)}$的每一维在整个训练集上的期望和方差．**因为目前主要的优化算法是基于小批量的随机梯度下降法，所以准确地计算$z^{(l)}$的期望和方差是不可行的．因此，$z^{(l)}$的期望和方差通常用当前小批量样本集的均值和方差近似估计**.

给定一个包含 K个样本的小批量样本集合，第l层神经元的净输入$z^{(1,l)}, \cdots, z^{(K,l)}$的均值和方差为

$$\mu_{\mathcal{B}} = \frac{1}{K}\sum_{k=1}^{K} z^{(k,l)},$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{K}\sum_{k=1}^{K} (z^{(k,l)} - \mu_{\mathcal{B}}) \odot (z^{(k,l)} - \mu_{\mathcal{B}}).$$

对净输入$z^{(l)}$的标准归一化**会使得其取值集中到0附近**，如果使用Sigmoid型激活函数时，**这个取值区间刚好是接近线性变换的区间，减弱了神经网络的非线性性质**．因此，**为了使得归一化不对网络的表示能力造成负面影响，可以通过一个附加的缩放和平移变换改变取值区间**

$$\hat{z}^{(l)} = \frac{z^{(l)} - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \odot \gamma + \beta$$

$$\triangleq \mathrm{BN}_{\gamma,\beta}(z^{(l)}),$$

其中γ和β分别代表缩放和平移的参数向量．**从最保守的角度考虑，可以通过标准归一化的逆变换来使得归一化后的变量可以被还原为原来的值．当**$\gamma = \sqrt{\sigma_B^2}$，$\beta = \mu_B$ **时，**$\tilde{z}^{(l)} = z^{(l)}$.

批量归一化操作**可以看作一个特殊的神经层，加在每一层非线性激活函数之前**，即

$$a^{(l)} = f\left(\mathrm{BN}_{\gamma,\beta}(z^{(l)})\right) = f\left(\mathrm{BN}_{\gamma,\beta}(Wa^{(l-1)})\right),$$

其中因为批量归一化本身具有平移变换，所以仿射变换 $Wa^{(l-1)}$不再需要偏置参数.

这里要注意的是，

- **每次小批量样本的μ_B和方差σ_B^2是净输入$z^{(l)}$的函数，而不是常量．因此在计算参数梯度时需要考虑μ_B和σ_B^2的影响**(反向传播时需要考虑这个分支路径)．
- 当训练完成时，用整个数据集上的均值μ和方差σ^2来分别代替每次小批量样本的μ_B和方差σ_B^2． 在实践中，μ_B和σ_B^2也可以用移动平均来计算．**每个BN层最终都会保存一对最终的均值和方差，可以用于测试阶段。**

值得一提的是，

- 逐层归一化不但可以**提高优化效率**，
- 还可以作为一种**隐形的正则化方法**．**在训练时，神经网络对一个样本的预测不仅和该样本自身相关，也和同一批次中的其他样本相关．由于在选取批次时具有随机性， 因此使得神经网络不会“过拟合”到某个特定样本，从而提高网络的泛化能力**[Luo et al., 2018]．

层归一化

批量归一化是**对一个中间层的单个神经元**进行归一化操作，

- **因此要求小批量样本的数量不能太小，否则难以计算单个神经元的统计信息**．
- 此外， **如果一个神经元的净输入的分布在神经网络中是动态变化的(例如RNN，其净输入在不同的时刻的分布是不同的)，比如循环神经网络，那么就无法应用批量归一化操作**．

层归一化(Layer Normalization)[Ba et al., 2016]是和批量归一化非常类似的方法．和批量归一化不同的是，**层归一化是对一个中间层的所有神经元进行归一化**．

对于一个深度神经网络， 令第l层神经元的**净输入为**$z^{(l)}$，其均值和方差为

$$\mu^{(l)} = \frac{1}{M_l}\sum_{i=1}^{M_l} z_i^{(l)},$$

$$\sigma^{(l)^2} = \frac{1}{M_l}\sum_{i=1}^{M_l}(z_i^{(l)} - \mu^{(l)})^2,$$

其中M_l**为第l层神经元的数量**．

层归一化定义为

$$\hat{\boldsymbol{z}}^{(l)} = \frac{\boldsymbol{z}^{(l)} - \mu^{(l)}}{\sqrt{\sigma^{(l)2} + \epsilon}} \odot \boldsymbol{\gamma} + \boldsymbol{\beta}$$
$$\triangleq \mathrm{LN}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\boldsymbol{z}^{(l)}),$$

其中γ和β分别代表缩放和平移的参数向量，和$z^{(l)}$维数相同.

循环神经网络中的层归一化

层归一化可以应用在循环神经网络中,对循环神经层进行归一化操作. 假设在时刻t，**循环神经网络的隐藏层为**h_t， 其层归一化的更新为

$$\boldsymbol{z}_t = \boldsymbol{U}\boldsymbol{h}_{t-1} + \boldsymbol{W}\boldsymbol{x}_t,$$
$$\boldsymbol{h}_t = f(\mathrm{LN}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\boldsymbol{z}_t)),$$

其中输入为x_t 为第t时刻的输入，U 和W为网络参数

在标准循环神经网络中，循环神经层的**净输入**一般**会随着时间慢慢变大或变小，从而导致梯度爆炸或消失**. **而层归一化的循环神经网络可以有效地缓解这种状况**.

层归一化和批量归一化整体上是十分类似的，**差别在于归一化的方法不同**. 对于K个样本的一个小批量集合

$Z^{(l)} = [z^{(1,l)}; \cdots; z^{(K,l)}]$,

- **层归一化是对矩阵$Z^{(l)}$ 的每一列进行归一化**,
- **而批量归一化是对每一行进行归一化**.
- **一般而言，批量归一化是一种更好的选择. 当小批量样本数量比较小时， 可以选择层归一化.**.