Batch Normalization （BN 批量规范化）

> 批量规范化（Batch Normalization）方法，能大幅加速模型训练，同时保持预测准确率不降，因而被一些优秀模型采纳为标准模型层。

一、数据标准化的必要性

由于神经网络的表达能力大多来自激活函数的非线性，所以让输入数据“适应于”激活函数是很重要的，具体而言：

- 对于 sigmoid 系列的激活函数而言，由于函数两端过于平坦，所以为了不陷入梯度消失的窘境，我们希望神经元的输入集中在函数中央，而不希望神经元的输入（绝对值）过大。
- 对于 Relu 激活函数而言，由于函数在输入小于 0 的区域恒等于 0，那么从直观上来说，如果一个很大的梯度把某个神经元的输入拉到了小于 0 很多的区域，该神经元从此以后的输出将永远是 0 了，因为他基本不可能回到大于 0 的区域。这就是著名的“Dying Relu 问题”，这通常会导致最后很多神经元处于“死亡”状态。

总结一下，可以把激活函数对输入数据的要求归结为如下两点：

- 输入数据不要过大
- 输入数据带来的梯度不要太大

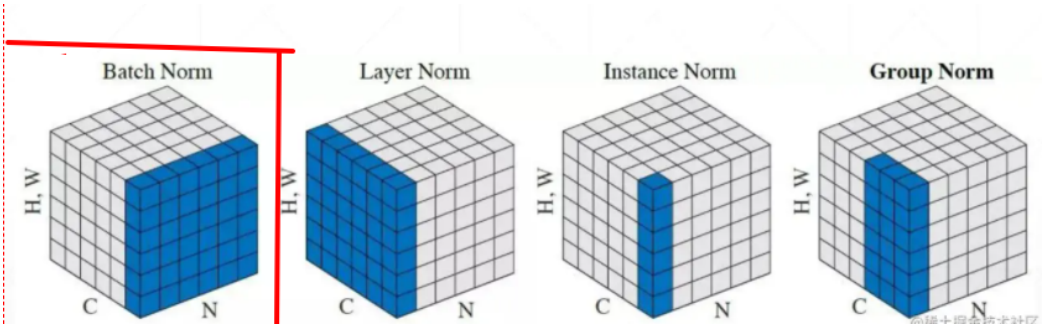
一个合理且行之有效的做法就是（即：**数据标准化**）：

- 将输入数据的均值控制在 0，从而意味着数据的“中心”是 0
- 将输入数据的方差控制在 1，从而意味着数据的“波动”不会太大

二、 批量规范化（Batch Normalization，BN）

> 输入数据为 N,C,(H*W) N 代表 tensor 数量，C 代表通道，H 代表高，W 代表宽

Batch Norm:对每一个批次（N 个 tensor）的每个通道分别计算均值 mean 和方差 var,如[10,4,9] 最终输出是[0,1,2,3]这样的 1*4 的 tensor



1.介绍

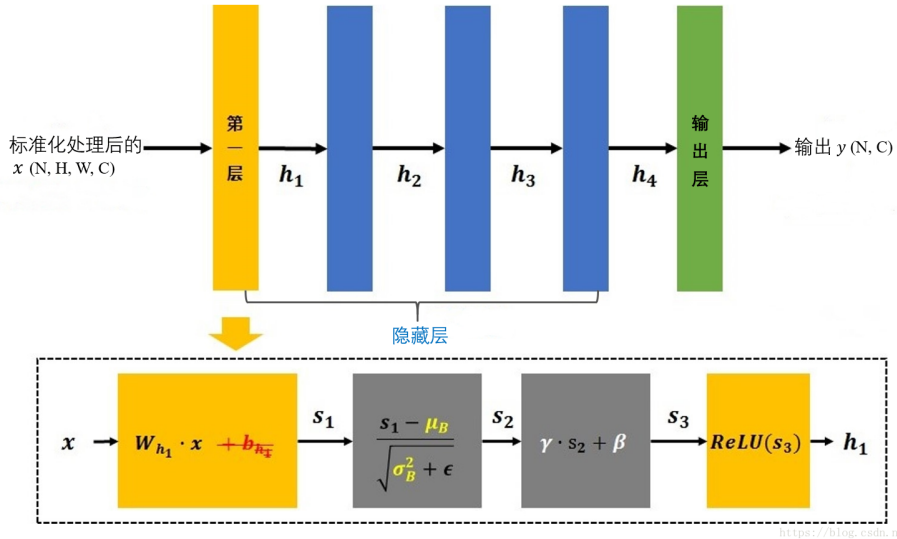
我们知道，在统计机器学习中算法中，一个常见的问题是 **协变量偏移**(Covariate Shift)，**协变量可以看作是输入变量。**一般的深度神经网络都要求输入变量在训练数据和测试数据上的分布是相似的，这是通过训练数据获得的模型能够在测试集获得好的效果的一个基本保障。

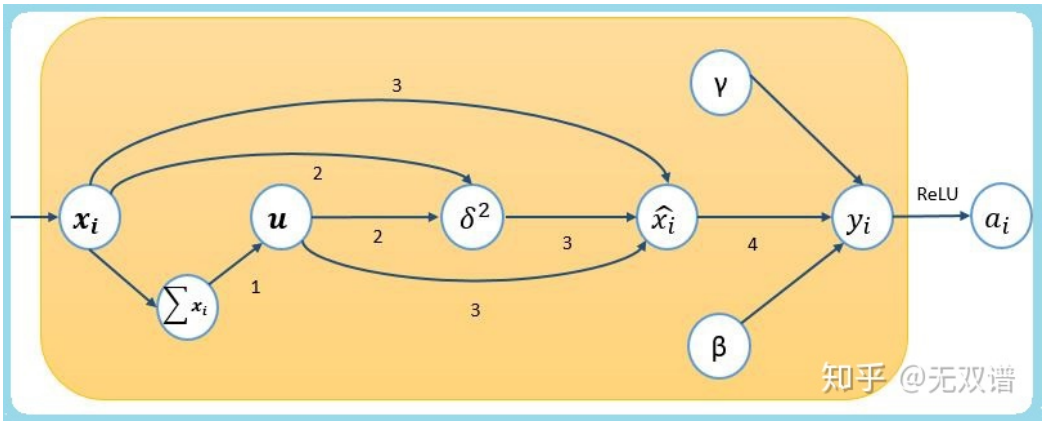
传统的深度神经网络在训练时，随着参数的不算更新，中间每一层输入的数据分布往往会和参数更新之前有较大的差异，导致网络要去不断的适应新的数据分布，进而使得训练变得异常困难，我们只能使用一个很小的学习速率和精调的初始化参数来解决这个问题。 仅仅对原始输入数据进行标准化是不充分的，因为虽然这种做法可以保证原始输入数据的质量，但它却无法保证隐藏层输入数据的质量。浅层参数的微弱变化经过多层线性变换与激活函数后被放大，改变了每一层的输入分布，造成深层的网络需要不断调整以适应这些分布变化，最终导致模型难以训练收敛。而且这个中间层的深度越大时，这种现象就越明显。由于是对层间数据的分析，也即是内部（internal），因此这种因网络中参数变化导致的内部节点数据分布发生变化的现象叫做**内部协变量偏移(internal Covariate Shift)，**称做 **ICS**。ICS 现象容易使训练过程陷入饱和区，减慢网络的收敛。Relu 从激活函数的角度出发，在一定程度上解决了梯度饱和的现象。而 2015 年提出的 BN 层，则从改变数据分布的角度避免了参数陷入饱和区。由于 BN 层优越的性能，其已经是当前卷积网络中的标配。

2.解决

为了解决这个问题，Sergey Ioffe’s 和 Christian Szegedy’s 在 2015 年首次提出了批量标准化（Batch Normalization，BN）的想法。该想法是：不仅仅对输入层做标准化处理，还要对**每一中间层的输入(激活函数前)** 做标准化处理，使得输出服从均值为 0，方差为 1 的正态分布，从而避免内部协变量偏移的问题。之所以称之为 批 标准化：是因为在训练期间，我们仅通过计算 当前层一小批数据 的均值和方差来标准化每一层的输入。

BN 的具体流程如下图所示：首先，它将隐藏层的输出结果（如，第一层： $W_{h1}.x$ 状态值）在 batch 上进行标准化；然后，经过缩放(scale)和平移(shift)处理，最后经过 RELU 激活函数得到 h_1 送入下一层（**就像我们在数据预处理中将 x 进行标准化后送入神经网络的第一层一样**）。





Input :$\mathcal{B}=\{x_1,\cdots,x_m\}$，为$m$个样本组成的一个$batch$数据。

Output :需要学习到的是γ和β，在框架中一般表述成weight和bias。

更新过程：

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \quad //得到batch中的统计特性之一：均值$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i-\mu_{\mathcal{B}})^2 \quad //得到batch中的另一个统计特性：方差$$

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

$//规范化，其中\epsilon是一个很小的数，防止计算出现数值问题。$

$$y_i \leftarrow \gamma\hat{x}_i+\beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \quad //这一步是输出尺寸伸缩和偏移。$$

γ **和** β **的解释：** γ 和 β 都是可学习参数，分别用作对标准化后的值进行 缩放(scale)和平移(shift)，提高网络的表达能力(**不加此参数的话，中间层的输出将被限制在标准正态分布下**)。 当参数分别初始化为 σ_B 和 μ_B 时，标准化的变量被还原为原来的值。随着网络的训练，网络会学到最合适的 γ 和 β ,最终中间层的输出将服从均值为 β 和 γ^2 的正态分布。

ϵ **的解释：**为了避免方差为 0 而加入的微小正数(eg：0.001).

b_{h1} **的解释：** 减去第一层 batch 个数据的均值 μ_B 后，偏置顶 b 的作用会被抵消掉， 所以在使用 BN 的层中没必要加入偏置项 b （红色删除线），其平移的功能将由 β 来代替

简单的将每层得到的数据进行直接的标准化操作显然是不可行的，因为这样会破坏每层自身学到的数据特征。为了使“规范化”之后不破坏层结构本身学到的特征，BN 引入了两个可以学习的**“重构参数”**以期望能够从规范化的数据中重构出层本身学到的特征。 算法步骤如下：其中批处理输入：$x:B=\{x_{1,\ldots,m}\}$，输出：规范化后的网络响应 $\{y_i=BN_{\gamma,\beta}(x_i)\}$。

算法 2 批规范化算法 BN

输入： 批处理（mini-batch）输入 x：$\mathcal{B}=\{x_{1,\dots,m}\}$

输出： 规范化后的网络响应 $\{y_i=\mathrm{BN}_{\gamma,\beta}(x_i)\}$

1: $\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$ // 计算批处理数据均值

2: $\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i-\mu_{\mathcal{B}})^2$ // 计算批处理数据方差

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

4: $y_i \leftarrow \gamma\hat{x}_i+\beta=\mathrm{BN}_{\gamma,\beta}(x_i)$ // 尺度变换和偏移

5: **return** 学习的参数 γ 和 β.

最后的**尺度变换和偏移操作是为了让因训练所需而加入的 BN 能够有可能还原最初的输入**（即当$\gamma=\sigma_B$和$\beta=\mu_B$时），从而保证了整个网络的容量（capacity）。 这里的最后一步也称之为仿射(affine)，引入这一步的目的主要是设计一个通道，使得输出 output 至少能够回到输入 input 的状态（当γ=1,β=0时）使得BN的引入至少不至于降低模型的表现，这是深度网络设计的一个套路。

BN 来规范化某些层或所有层的输入，从而可以**固定每层输入信号的均值与方差**。这样一来，即使网络模型较深层的响应或梯度很小，也可通过 BN 的规范化作用将其的尺度变大，以此便可解决深层网络训练很可能带来的“梯度弥散”问题。

值得注意的是，输入的标准化处理Normalizing inputs和隐藏层的标准化处理Batch Normalization是有区别的。Normalizing inputs使所有输入的均值为0，方差为1。而Batch Normalization可使各隐藏层输入的均值和方差为任意值。实际上，从激活函数的角度来说，如果各隐藏层的输入均值在靠近0的区域即处于激活函数的线性区域，这样不利于训练好的非线性神经网络，得到的模型效果也不会太好。这也解释了为什么需要用 γ 和 β 来对 $z^{[l](i)}$ 作进一步处理。

同时，**通过调整γ和β的值，既可以改变某层原来的输入，又可以保持原输入，这样就提高了模型的容纳能力。**在实验中， 研究人员发现可通过BN来规范化某些层或所有层的输入，从而可以固定每层输入信号的均值与方差。这样一来，即使网络模型较深层的响应或梯度很小，也可通过BN的规范化作用将其的尺度变大，以此便可解决深层网络训练很可能带来的“梯度弥散”问题。

Exmaple 1

一个直观的例子：**对一组很小的随机数做$\ell 2$ 规范化操作：**

如下：

$$v=[0.0066,0.0004,0.0085,0.0093,0.0068,0.0076,0.0074,0.0039,0.0066,0.0017]^{\top}.$$

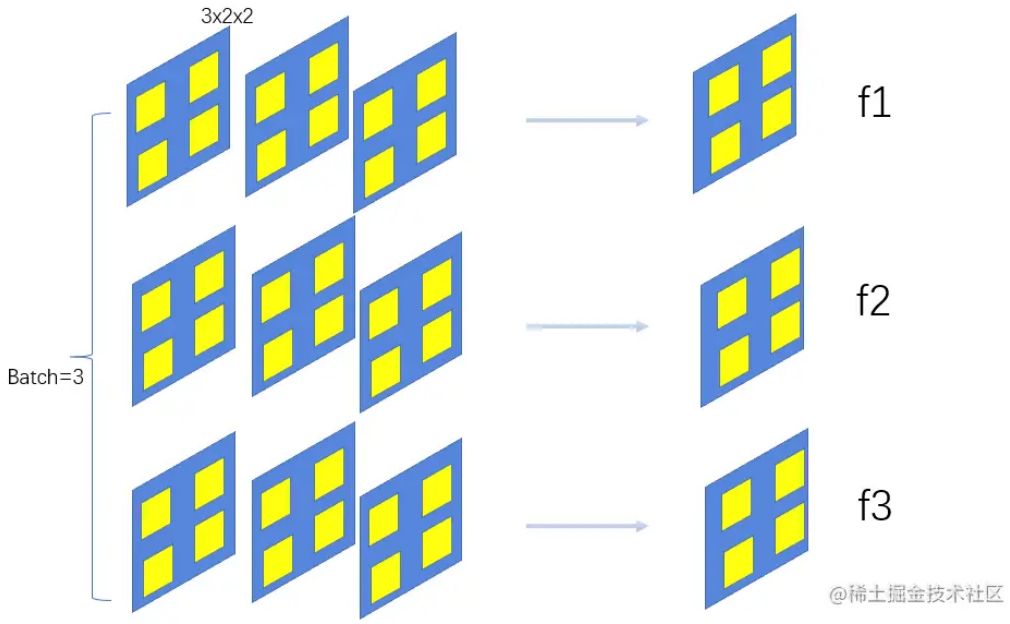
在 ℓ_2 规范化后，这组随机数变为：

$$v'=[0.3190,0.0174,0.4131,0.4544,0.3302,0.3687,0.3616,0.1908,0.3189,0.0833]^{\top}.$$

显然，经过规范化作用后，原本微小的数值其尺度被“拉大”了，试想如果未做规范化的那组随机数 v 就是反向传播的梯度信息，规范化自然可起到缓解“梯度弥散”效应的作用。

Exmaple 2

我们假设在网络中间经过某些卷积操作之后的输出的 feature map 的尺寸为 3×3×2×2
3 为 batch 的大小，3 为 channel 的数目，2×2 为 feature map 的长宽



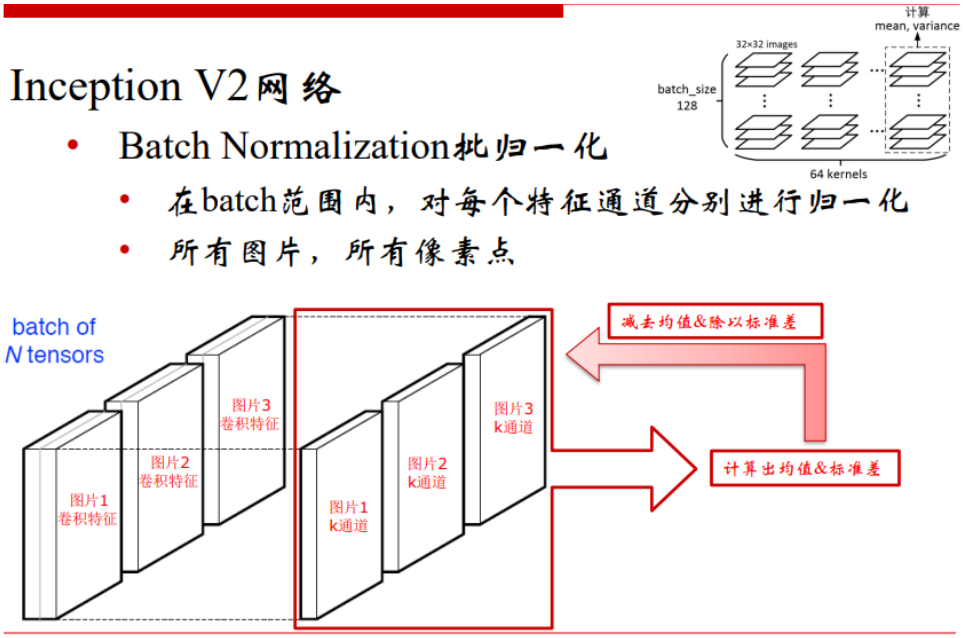
上图中，batch size 一共是 3, 对于每一个 batch 的 feature map 的 size 是 3×2×2
对于所有 batch 中的同一个 channel 的元素进行求均值与方差，比如上图，对于所有的 batch，都拿出来最后一个 channel，一共有 3×4=12 个元素.

$$mean = （f1 + f2 + f3）/12 \ (f1 : x_1 \dots x_4; f2 : x_5 \dots x_8; f3 : x_9 \dots)$$

$$\sigma^2 = \sum_{i=1}^{12}(x_i - mean)$$

3. 规则

- BN 通常应用于输入层或任意中间层，且最好在**激活函数前**使用使用 BN 的层不需要加 bias 项了，因为减去 batch 个数据的均值 μ_B 后， 偏置项 b 的作用会被抵消掉，其平移的功能将由 β 来代替。
- BN 的训练阶段均值和方差的求法:
  - **全连接层**：对每个节点(node) 对应的 batch 个数据（所有像素）分别求均值和方差.
  - **二维卷积层**：对每一个 channel 所对应的 batch 个特征图（所有像素）求均值和方差，输出 C 维.
  - **BN 层参数**：mean（C 维）、variance（C 维）、moving average factor（1 维，caffe 中默认为 0.999）.



- **BN 的测试阶段均值和方差的求法**：测试时，使用的是训练集的每一个 BN 层累积下来的 均值和方差(C 维)；训练阶段使用指数衰减滑动平均（EMA） 将每个 batch 的均值和方差不断的累积下来，从而近似得到整个样本集的均值和方差.
- **指数衰减滑动平均**（EMA:介均值的计算公式如下（标准差的计算同理）

$$\mu_n = \alpha\mu_{n-1} + (1-\alpha)\cdot\frac{1}{N}\sum_{i}^{N} x_{i,n}$$

三、BN 层的优缺点总结

优点:

1、 缓解梯度消失，加速网络收敛。

2、 简化调参，网络更稳定。BN 层抑制了参数微小变化随网络加深而被放大的问题，对参数变化的适应能力更强，更容易调参。

3、 防止过拟合。BN 层将每一个 batch 的均值和方差引入到网络中，由于每个 batch 的这俩个值都不相同，可看做为训练过程增 加了随机噪声，可以起到一定的正则效果，防止过拟合。

缺点:

1、由于是在 batch 的维度进行归一化，BN 层要求较大的 batch 才能有效的工作，而物体检测等任务由于占用内存过高，限制了 batch 的大小，这回限制 BN 层有效的发挥归一化的功能。

2、数据的 batch 大小在训练和测试时往往不一样。在训练时一般采用滑动来计算平均值与方差，在测试时直接拿训练集的平均值与方差来使用。这种方式会导致测试集依赖与训练集，然而有时测试集与训练集的数据分布并不一致。

四、好文

- [ ] 批量标准化（BN）、实例标准化(IN)、特征标准化(FN)
- [ ] 批量归一化（batch normalization）层
- [ ] 详解深度学习中的Normalization，BN/LN/WN
- [ ] 《Batch Normalization Accelerating Deep Network Training by Reducing Internal Covariate Shift》阅读笔记与实现
- [ ] Pytorch的BatchNorm层使用中容易出现的问题