Feature Normalization

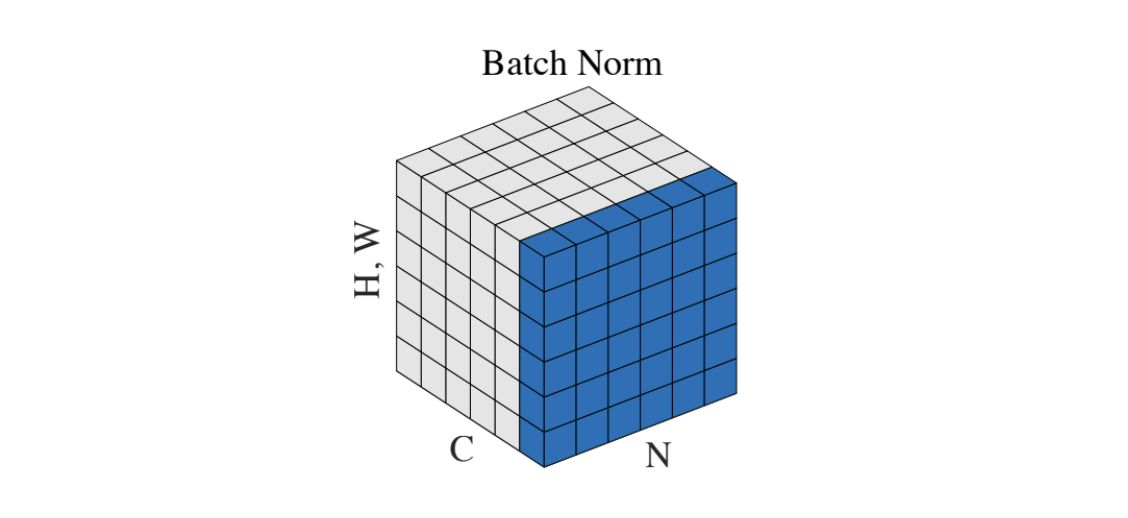
在统计机器学习中的一个经典假设是“源空间（source domain）和目标空间（target domain）的数据分布（distribution）是一致的”， 如果不一致，那么就出现了新的机器学习问题，如，transfer learning/domain adaptation等。而covariate shift就是分布不一致假设之下的一个分支问题，它是指源空间和目标空间的条件概率是一致的，但是其边缘概率不同。对于神经网络的各层输出，由于它们经过了层内操作作用，其分布显然与各层对应的输入信号分布不同，而且差异会随着网络深度增大而增大，可是它们所能“指示”的样本标签（label）仍然是不变的，这便符合了covariate shift的定义。使用normalization方法可以缓解covariate shift。

深度学习中，常见的的normalization方法有四种，分别为Batch Normalization, Layer Normalization, Instance Normalization, Group Normalization。四种方法均为计算均值和方差进行标准化，然后进行一定的缩放和位移。而只是对归一化的数据不一样。Batch Normalization是N\*H\*W，Layer Normalization是C\*H\*W，Instance Normalization是H\*W，Group Normalization是G\*H\*W。归一化的公式所示：

其中是一个很小的数，作用是防止数值计算不稳定。是仿射参数，将归一化后的数据再次放缩得到新的数据，可以理解为标准差，可以理解为均值，它们两个一般是可学习的。

1. Batch Normalization

Batch Normalization的归一化数据在Batch 维度，其计算归一化数据为Batch、W、H三个维度，即对每一个Channel中的所有数据进行归一化，共操作Channel次归一化。即如下图所示：

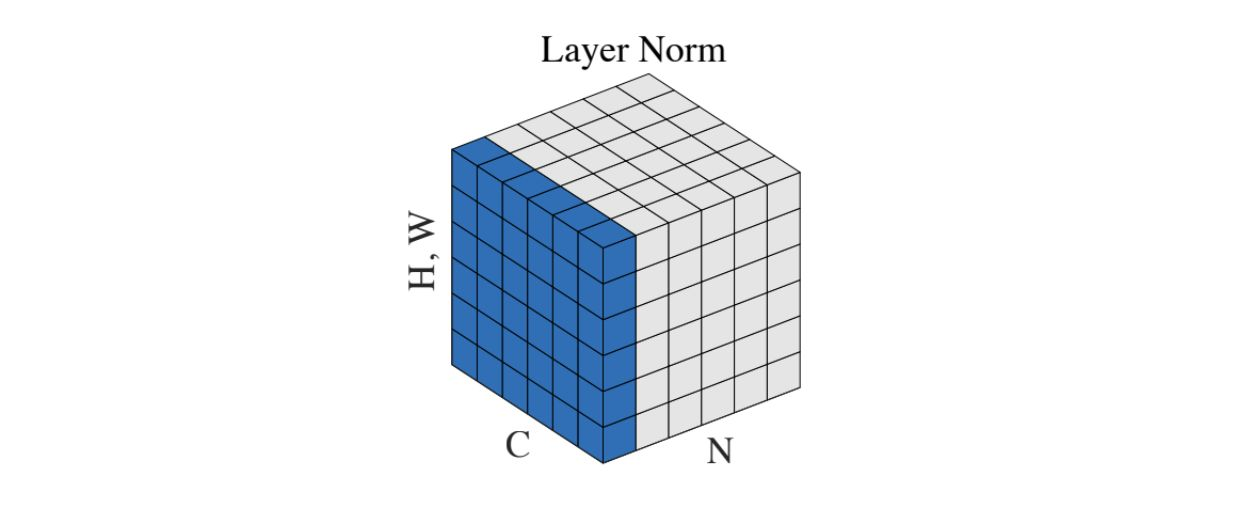


同时BN的存在会引入mini-batch内其他样本的信息，就会导致预测一个独立样本时，其他样本信息相当于正则项，使得loss曲面变得更加平滑，更容易找到最优解。相当于一次独立样本预测可以看多个样本，学到的特征泛化性更强，更加general。

但当batch-size较小时（batch-size<16），其计算的均值和方差不稳定。且如果在对比学习等任务时，会导致预测一个独立样本时，会引入其他mini-batch样本的信息，造成信息泄露，骗过模型。

1. Layer Normalization

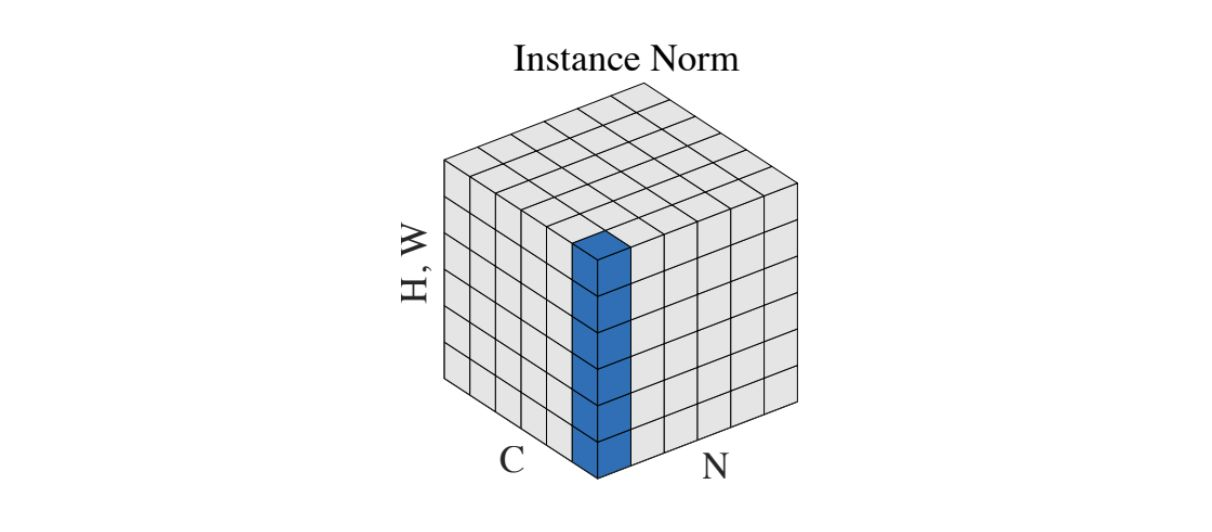
Layer Normalization计算归一化数据为Channel、W、H三个维度，即对mini-batch内的每一个样本进行归一化，共操作Batch-size次归一化。即如下图所示：



Layer Normalization提出的主要原因是无法找到将BN应用于递归神经网络，需要找到一个替代的方法。对于更一般的情况，Layer Normalization是对输入的后几维（除去N维）进行归一化。其常用于递归神经网络和transformer中。在实际的CNN或Transformer网络中，也有对同一通道中的各元素单独做归一化，即操作Batch-size×H×W次归一化。

1. Instance Normalization

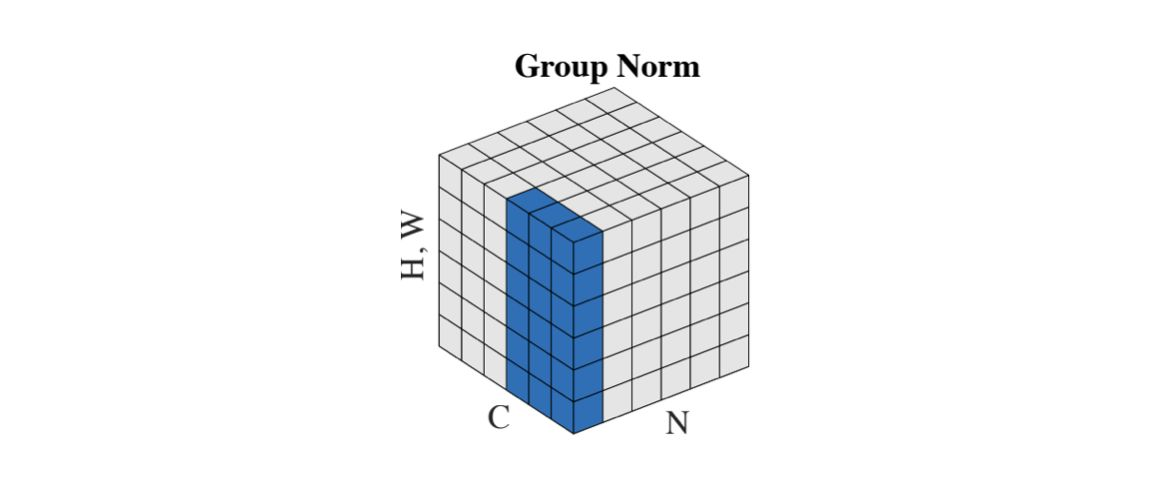
Instance Normalization计算归一化数据为W、H两个维度，即对每一个features map进行归一化,共操作Batch-size×Channels次归一化。即如下图所示：



跨批次和通道的标准化允许从图像中删除特定的对比度信息，这有助于泛化。这种方法在Pix2Pix或CycleGAN等生成模型中广受欢迎，并成为著名的 StyleGAN2中使用的自适应实例归一化的先驱。对于更一般的情况，Instance Normalization是对输入的后几维（除去N维、C维）进行归一化。其默认不进行仿射变换。

1. Group Normalization

Group Normalization在一个样本的多个Channels内分组,分成多组Channels,在各组内进行归一化。若分为group组，那么共操作Channels×Batch-size÷Group次归一化。GN 可以理解为LN 和IN的混合,或者说LN 和IN是GN的特例。即如果分为Channel组，即每组一个Channel，则等价于Instance Normalization；如果只分为1组，即所有Channel为一组，则等价于Layer Normalization。其操作如下图所示：



在深度学习没有火起来之前，提取特征通常是使用SIFT，HOG和GIST特征，这些特征有一个共性，都具有按group表示的特性.因此尝试在group内进行归一化。同时GN不同于BN的一点在于,GN不再需要维护归一化时的均值和方差。在batch-size较小（batch-size<16），group-size较大（group-size≥16）时，GN优于BN，而batch-size较大时则无法击败BN。

1. Weight Normalization

Weight Normalization是对模型中的权重（不包括偏置）进行归一化，并对其乘以一个可学习的参数，在训练时与均会进行学习。

1. AdaIN (Adaptive Instance Normalization)

AdaIN是Instance Normalization的变形，其没有可学习的仿射参数。AdaIN接收两个信息源：内容输入和风格输入，目的是将的通道级（Channel-wise）均值和标准差匹配到的通道级均值和标准差上：

在风格迁移类应用中效果很不错。

1. AdaGN (Adaptive Group Normalization)

AdaGN是Group Normalization的变形，其在Group Normalization的基础上进一步对其进行缩放和位移（类似于AdaIN和FiLM）。如下公式所示：

其中表示对输入的特征图进行正常的组归一化，对其之后的结果进行缩放和位移。在diffusion models beat GANs on image synthesis的论文中，是对timestep和class embedding的线性映射。这样做比直接将和加到中效果会好。

1. RMS Norm（Root Mean Square Layer Normalization）

对于正常的归一化而言，其都是减去均值，乘以标准差，再进行一个缩放和平移，即为最开始提到的。而RMS Norm则是对其的一个变体，其取消了减去均值和平移这两个步骤，即如下面公式：

一个直观的猜测是，center操作，类似于全连接层的bias项，储存到的是关于预训练任务的一种先验分布信息，而把这种先验分布信息直接储存在模型中，反而可能会导致模型的迁移能力下降。

**对于Normalization方法的使用建议：**

(1) 正常的处理图片的CNN模型都应该使用Batch Normalization。只要保证batch size较大(不低于32)，并且打乱了输入样本的顺序。如果batch太小，则优先用Group Normalization替代。

(2) 对于RNN等时序模型，有时候同一个batch内部的训练实例长度不一(不同长度的句子)，则不同的时态下需要保存不同的统计量，无法正确使用BN层，只能使用Layer Normalization。

(3) 对于图像生成以及风格迁移类应用，使用Instance Normalization更加合适。

Data Normalization

1. 线性归一化

线性归一化方法有个非常致命的缺陷，当的最大值或者最小值为孤立的极值点，会影响性能。

2. 零均值归一化/Z-score Normalization

零均值归一化称为标准化方法，即每一变量值与其平均值之差除以该变量的标准差。

经过处理后的数据符合均值为0，标准差为1的分布，如果原始的分布是正态分布，那么z-score标准化就将原始的正态分布转换为标准正态分布，机器学习中的很多问题都是基于正态分布的假设，这是更加常用的归一化方法。但是线性归一化和零均值归一化不会改变分布本身的形状。

3. 正态分布Box-Cox变换

box-cox变换可以将一个非正态分布转换为正态分布，使得分布具有对称性，变换公式如下：

式中为经Box-Cox变换后得到的新变量，为原始连续因变量，为变换参数，需要经过对数据样本的计算得到。且以上变换要求原始变量取值为正，若取值为负时，可先对所有原始数据同加一个常数使其为正值，然后再进行以上的变换。

4. 直方图均衡化

直方图均衡也可以将某一个分布归一化到另一个分布，它通过图像的灰度值分布，即图像直方图来对图像进行对比度进调整，可以增强局部的对比度。它的变换步骤如下：(1)计算概率密度和累积概率密度。

(2)创建累积概率到灰度分布范围的单调线性映射T。

(3)根据T进行原始灰度值到新灰度值的映射。

5. 白化(Whitening)

白化的目的是去除输入数据的冗余信息。假设训练数据是图像，由于图像中相邻像素之间具有很强的相关性，所以用于训练时输入是冗余的；白化的目的就是降低输入的冗余性。输入数据集X，经过白化处理后，新的数据X'满足两个性质：

(1)特征之间相关性较低；

(2)所有特征具有相同的方差。

白化算法的实现过程，第一步操作就是PCA(不进行降维，只是降低特征之间的相关性)，求出新特征空间中X的新坐标，然后再对新的坐标进行方差归一化操作。该方法又称为PCA白化（Principal Component Analysis Whitening），之后各维度变得不相关，且每个维度方差都是1。

其计算公式如下：

其中维X的奇异值，维X的奇异向量，是防止数值计算不稳定。

6. ZCA白化(Zero-phase Component Analysis Whitening)

ZCA白化则是在PCA白化基础上，将PCA白化后的数据旋转回到原来的特征空间，这样可以使得变换后的数据更加接近原始输入数据。 ZCA白化的计算公式：

数据在经过ZCA白化以后，其协方差矩阵依然是一个单位矩阵。

Weight initialization

神经网络参数初始化的几个基本要求：

1. 参数不能全部初始化为0，也不能全部初始化同一个值；
2. 参数初始化的均值为0，正负交错，正负参数大致上数量相等；
3. 初始化参数不能太大或者太小，参数太小会导致特征在每层间逐渐缩小而难以产生作用，参数太大会导致数据在逐层间传递时逐渐放大而导致梯度消失发散，不能训练；
4. 各个层的激活值h（输出值）的方差要保持一致；
5. 各个层对状态Z的梯度的方差要保持一致

以上第（4）、（5）点又成为Glorot条件，前三条是硬性条件，如果能满足Glorot条件是最好的。下面将介绍常见的权重初始化方法。

关于增益gain的计算可以使用torch.nn.init.calculate\_gain(nonlinearity, param=None)。其中param在nonlinearity= 'leaky\_relu'时使用，且值为leaky\_relu的负斜率大小。总的增益表如下：

|  |
| --- |
|  |
| nonlinearity | gain |
| Linear / Identity | 1 |
| Conv{1,2,3}D | 1 |
| Sigmoid | 1 |
| Tanh | 5/3 |
| ReLU |  |
| Leaky Relu | ​​ |
| SELU | 3/4 |

1. 常数初始化

torch.nn.init.constant\_(tensor, val)。对Normalization常用的初始化方法，即均值初始化为0，方差初始化为1。

1. 均匀分布初始化

torch.nn.init.uniform\_(tensor, a=0, b=1)。从均匀分布U(a, b)中生成值。

1. 正态分布初始化

torch.nn.init.normal\_(tensor, mean=0, std=1)。从给定均值和标准差的正态分布N(mean, std2)中生成值。

1. Xavier 均匀分布（Glorot initialisation）

torch.nn.init.xavier\_uniform\_(tensor, gain=1)。Xavier初始化的基本思想是，若对于一层网络的输出和输出可以保持正态分布且方差相近，这样就可以避免输出趋向于0，从而避免梯度弥散情况。

该方法推导基于以下两个假设：一是激活函数是线性的，二是激活值关于0对称。因此该方法不适用于非线性激活函数或激活后的值不是以0为中心对称的。其是从均匀分布U(-a, a)中生成值。

其中和分别是输入和输出神经元数量：

1. Xavier 正态分布（Glorot initialisation）

torch.nn.init.xavier\_normal\_(tensor, gain=1)。从给定均值和标准差的正态分布N(0, std2)中生成值。

1. kaiming均匀分布（He initialization）

torch.nn.init. kaiming\_uniform\_ (tensor, a=0, mode='fan\_in', nonlinearity='leaky \_relu')。其中a指nonlinearity='leaky\_relu'时leaky\_relu的负斜率大小。Nonlinearity可取'relu'或'leaky\_relu'。该方法不再要求激活函数是线性且激活值关于0对称。因此为最常用的神经网络初始化方法。其是从均匀分布U(-bound, bound)中生成值。

其中可以为也可以为。

1. kaiming正态分布（He initialization）

torch.nn.init.kaiming\_uniform\_(tensor, a=0, mode='fan\_in', nonlinearity='leaky\_ relu') 。该方法同样不再要求激活函数是线性且激活值关于0对称。因此为最常用的神经网络初始化方法。其是从给定均值和标准差的正态分布N(0, std2)中生成值。

1. NTK参数化

该初始化方法是用“均值为0、方差为1的随机分布”来初始化，但是将输出结果除以，即模型变为：

这在高斯过程中被称为“NTK参数化”。 很显然，利用NTK参数化，我们可以将所有参数都用标准方差初始化，但依然保持二阶矩不变。一个很自然的问题是：NTK参数化跟直接用Xavier初始化相比，有什么好处吗？

理论上，是有一点好处的。利用NTK参数化后，所有参数都可以用方差为1的分布初始化，这意味着每个参数的量级大致都是相同的O(1)级别，于是我们可以设置较大的学习率，比如10−2，并且如果使用自适应优化器，其更新量大致是，那么我们就知道10−2的学习率每一步对参数的调整程度大致是1%。总的来说，NTK参数化能让我们更平等地处理每一个参数，并且比较形象地了解到训练的更新幅度，以便我们更好地调整参数。