归一化的含义

归一化的具体作用是归纳统一样本的统计分布性。归一化在 0 – 1之间是统计的概率分布， 归一化在-1~+1 之间是统计的坐标分布。

归一化有同一、统一和合一的意思。无论是为了建模还是为了计算，首先基本度量单位要同一，神经网络是以样本在事件中的统计分别几率来进行训练（概率计算）和预测的，且 sigmoid 函数的取值是 0 到 1 之间的，网络最后一个节点的输出也是如此，所以经常要对样本的输出归一化处理。

归一化是统一在 0 – 1之间的统计概率分布， 当所有样本的输入信号都为正值时，与第一隐含层神经元相连的权值只能同时增加或减小，从 而导致学习速度很慢。另外在数据中常存在奇异样本数据，奇异样本数据存在所引起的网络训练时间增加，并可能引起网络无法收敛。为了避免出现这种情况及后面数据处理的方便，加快网络学习速度，可以对输入信号进行归一化，使得所有样本的输入信号其均值接近于 0 或与其均方差相比很小。

归一化的作用

1. 为了后面数据处理的方便，归一化的确可以避免一些不必要的数值问题。
2. 为了程序运行时收敛加快。
3. 统一量纲。样本数据的评价标准不一样，需要对其量纲化，统一评价标准。这算是应用层面的需求。
4. 避免神经元饱和。什么意思？就是当神经元的激活在接近 0 或者 1 时会饱和，在这些区域，梯度几乎为 0，这样，在反向传播过程中，局部梯度就会接近 0，这会有效地“杀死” 梯度。
5. 保证输出数据中数值小的不被吞食。

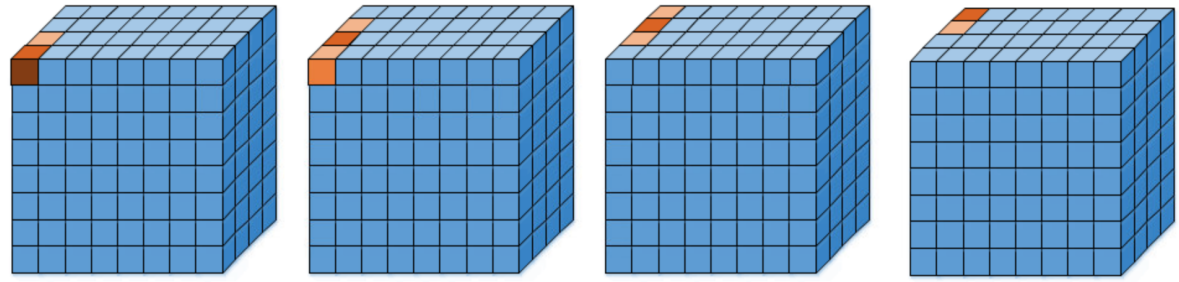
常见归一化方法

局部响应归一化（Local Response Normalization）

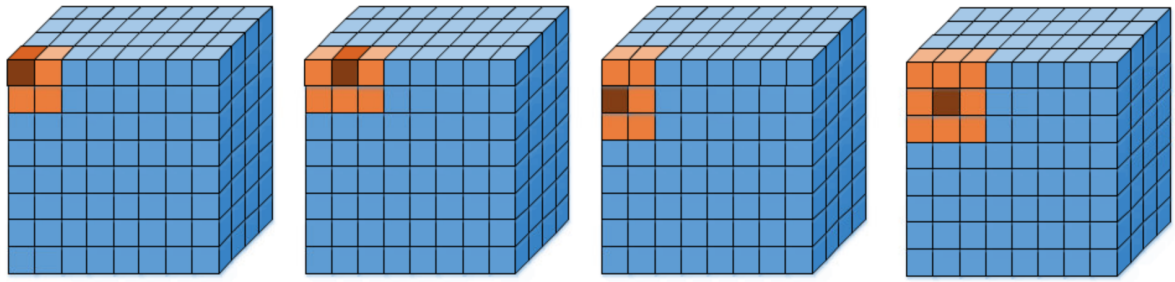
简介

局部响应归一化（LRN）最初是在AlexNet架构中引入的，其中使用的激活函数是ReLU，而不是当时更常见的tanh和sigmoid。除了上述原因，使用LRN的原因是为了鼓励**横向抑制（lateral inhibition）**。这是神经生物学中的一个概念，是指神经元减少其邻居活动的能力[1]（***注：阻止兴奋神经元向邻近神经元传播其动作趋势，从而减少兴奋神经元的邻近神经元的激活程度***）。在深度神经网络（Deep Neural Networks，DNN）中，这种横向抑制的目的是进行局部对比度增强，以便使局部最大像素值用作下一层的激励。

LRN是一个不可训练的层，用于对局部邻域内的特征图中的像素值进行平方归一化。根据所定义的邻域，有两种类型的LRN，如下图所示：



a) Inter-Channel LRN (n=2)



b) Intra-Channel LRN (n=2)

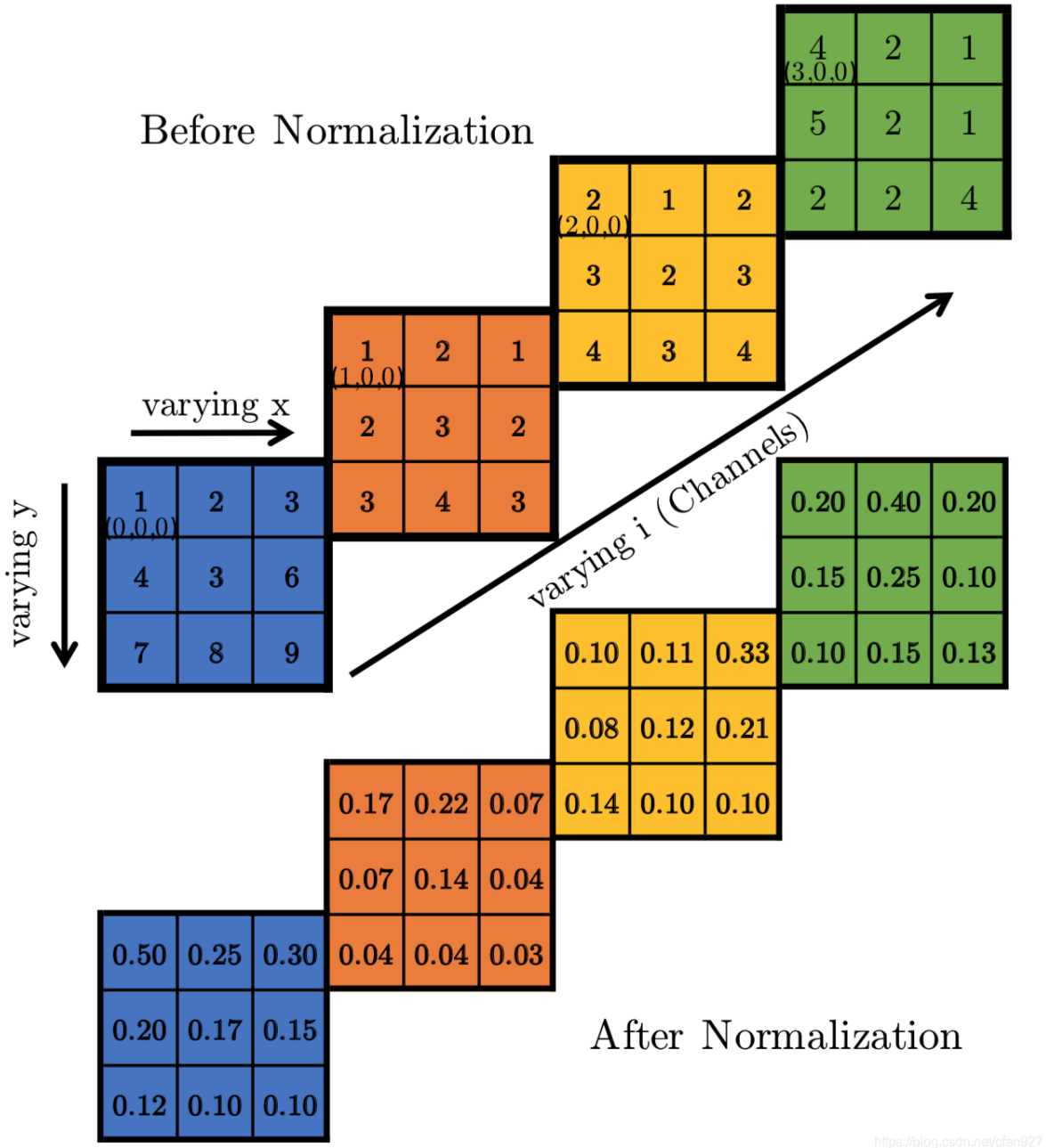
通道间的LRN（Inter-Channel LRN）

这最早是在AlexNet论文中使用的。定义的邻域在整个通道上，对于每个(x,y)位置，在深度维度上进行归一化，其公式如下：

$$b_{x,y}^{i}=a_{x,y}^{i}/\left(k+\alpha\sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)}\left(a_{x,y}^{j}\right)^{2}\right)^{\beta}$$

其中，i表示滤波器i的输出，$a(x,y)$、$b(x,y)$分别为归一化前后(x,y)处的像素值，N为通道总数。常数(k,α,β,n)为超参数。k用于避免奇异点(分母为零的情况)，α为归一化常数，β是对比度常数。常数n用于定义邻域长度，即在进行归一化时需要考虑多少个连续像素值。$(k,\alpha,\beta,n)=(0,1,1,n)$的情况是标准归一化。在上图中，N=4，n=2。

来看一个通道间LRN的例子。如下图：



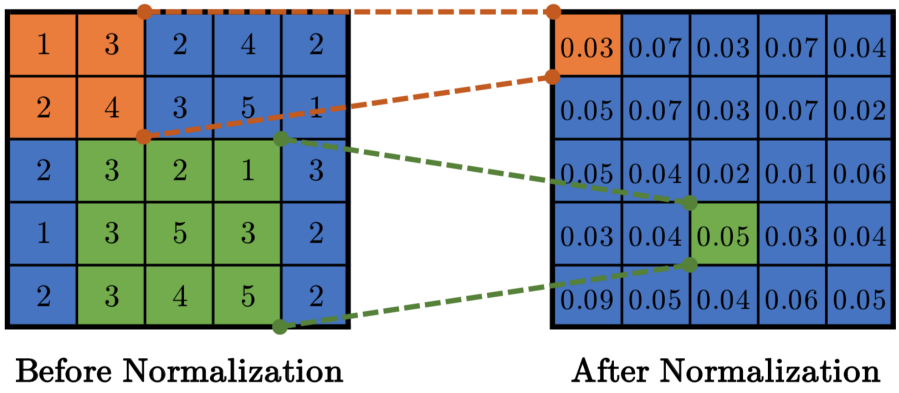
不同的颜色表示不同的通道，因此N=4。设置超参数为$(k, \alpha, \beta, n) = (0, 1, 1, 2)$。n=2的值表示在计算位置$(i, x, y)$处的归一化值时，我们考虑上一个滤波器和下一个滤波器在相同位置的值，即$(i-1, x, y)$和$(i+1, x, y)$。对于$(i, x, y) = (0, 0, 0)$（此处表示位置），我们有值$(i, x, y) = 1$，值$(i-1, x, y)$不存在，值$(i+1, x, y) = 1$。因此 $normalized\_value(i(x, y) = \frac{1}{1^2+1^2} = 0.5$,可以在上图的下部看到。其余归一化值的计算方法类似。

通道内的LRN（Intra-Channel LRN）

由上面的图可以看到，在通道内LRN中，邻域仅在同一通道中扩展。公式为：

$$b^k_{x,y} = a^k_{x,y} / \left(k + \alpha \sum_{i=\max(0,x-n/2)}^{\min(W,x+n/2)} \sum_{j=\max(0,y-n/2)}^{\min(H,y+n/2)} \left(a^k_{i,j}\right)^2 \right)^{\beta}$$

其中(W,H)为特征图的宽和高（如上面第一幅图中$(W, H) = (8, 8)$。**通道间LRN和通道内LRN之间的惟一区别是归一化的邻域。**在通道内LRN中，一个二维邻域（相对于通道间的一维邻域）是在考虑的像素周围定义的。例如，下图显示了在$n = 2$的5x5特征图上的通道内归一化（即以为中心的大小为$(n+1) \times (n+1)$的2D邻域)。



在paddlepaddle中使用

```
paddle.nn.LocalResponseNorm(size, alpha=0.0001, beta=0.75, k=1.0,
data_format='NCHW', name=None)
```

1. 参数:

- **size** (int) - 累加的通道数。
- **alpha** （float，可选）- 缩放参数，正数。默认值为1e-4。
- **beta** （float，可选）- 指数，正数。默认值为0.75。
- **k** （float，可选）- 位移，正数。默认值为1.0。
- **data_format** (str，可选) - 指定输入的数据格式，输出的数据格式将与输入保持一致。如果输入是三维 Tensor ,该参数可以是"NCL"或"NLC"，其中N是批尺寸，C是通道数，L是特征长度。如果输入是四维 Tensor ,该参数可以是"NCHW"或"NHWC"，其中N是批尺寸，C是通道数，H是特征高度，W是特征宽度。如果输入是五维 Tensor ,该参数可以是"NCDHW"或"NDHWC"，其中N是批尺寸，C是通道数，D是特征深度，H是特征高度，W是特征宽度。默认值："NCHW"。
- **name** (str，可选): 操作的名称(可选，默认值为None）。更多信息请参见 Name 。

2. 形状:

- **输入** - 三维/四维/五维 Tensor 。
- **输出** - 数据类型及维度和输入相同的 Tensor 。

3. 代码示例:

```
import paddle

x = paddle.rand(shape=(3, 3, 112, 112), dtype="float32")
m = paddle.nn.LocalResponseNorm(size=5)
y = m(x)
print(y.shape)  # [3, 3, 112, 112]
```

群组归一化（Group Normalization）

简介

FAIR 团队的吴育昕和何恺明提出了**组归一化**（Group Normalization，简称 GN）的方法，GN 将信号通道分成一个个组别，并在每个组别内计算归一化的均值和方差，以进行归一化处理。**GN 的计算与批量大小无关，而且在批次大小大幅变化时，精度依然稳定。**

通常来说，在使用 Batch Normalization（以下将简称 BN）时，采用小批次很难训练一个网络，而对于不使用批次的优化方法来说，效果很难媲美采用大批次BN时的训练结果。当使用 Group Normalization（以下将简称 GN），且 batch size 大小为 1 时，仅需要多写两行代码加入权重标准化方法，就能比肩甚至超越大批次BN时的训练效果。

Group Normalization是**对每个训练实例的通道组进行归一化。**我们可以说，Group Norm介于Instance Norm和Layer Norm之间。**当把所有的通道放到一个组中时，组归一化就变成了层归一化，而当把每个通道放到不同的组中时，就变成了实例归一化。**

在paddlepaddle中使用

```
class paddle.nn.GroupNorm(num_groups, num_channels, epsilon=1e-05, weight_attr=None, bias_attr=None, data_layout='NCHW, 'name=None)
```

1. 参数：

- **num_groups** (int) - 从通道中分离出来的 `group` 的数目。
- **num_channels** (int) - 输入的通道数。
- **epsilon** (float, 可选) - 为防止方差除零，增加一个很小的值。默认值：1e-05。
- **weight_attr** (ParamAttr|bool, 可选) - 指定权重参数属性的对象。如果为False, 表示参数不学习。默认值为None，表示使用默认的权重参数属性。具体用法请参见 ParamAttr 。
- **bias_attr** (ParamAttr|bool, 可选) - 指定偏置参数属性的对象。如果为False, 表示参数不学习。默认值为None，表示使用默认的偏置参数属性。具体用法请参见 ParamAttr 。
- **data_format** (string, 可选) - 只支持“NCHW”(num_batches，channels，height，width)格式。默认值：“NCHW”。
- **name** (string, 可选) – GroupNorm的名称, 默认值为None。更多信息请参见 Name

2. 形状：

- input: 形状为（批大小，通道数, 高度，宽度）的4-D Tensor。
- output: 和输入形状一样。

3. 代码示例：

```
import paddle
import numpy as np

np.random.seed(123)
x_data = np.random.random(size=(2, 6, 2, 2)).astype('float32')
x = paddle.to_tensor(x_data)
group_norm = paddle.nn.GroupNorm(num_channels=6, num_groups=6)
group_norm_out = group_norm(x)

print(group_norm_out)
```

实例归一化（Instance Normalization）

简介

IN针对图像像素做normalization，最初用于图像的风格化迁移。在图像风格化中，生成结果主要依赖于某个图像实例，feature map的各个channel的均值和方差会影响到最终生成的图像风格。所以对整个batch归一化不适合图像风格化中，因此对 H,W坐标归一化。可以加速模型的收敛，并且保持每个图像实例之间的独立。

对于特征图 $x \in \mathbb{R}^{N\times C\times H\times W}$ ，IN对每个样本的 H,W 维度的数据求均值和标准差，保留 N,C维度，也就是说，它只在channel内部求均值和标准差，计算公式如下：

$$y_{tijk} = \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \varepsilon}}$$

$$\mu_{ti} = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm}$$

$$\sigma_{ti}^2 = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tilm} = mu_{ti})$$

在paddlepaddle中使用

```
paddle.nn.InstanceNorm1D(num_features, epsilon=1e-05, momentum=0.9,
weight_attr=None, bias_attr=None, data_format="NCL", name=None)
```

1. 参数：

- **num_features** (int) - 指明输入 `Tensor` 的通道数量。
- **epsilon** (float, 可选) - 为了数值稳定加在分母上的值。默认值：1e-05。
- **momentum** (float, 可选) - 此值用于计算 `moving_mean` 和 `moving_var` 。默认值：0.9。更新公式如上所示。
- **weight_attr** (ParamAttr|bool, 可选) - 指定权重参数属性的对象。如果为False, 则表示每个通道的伸缩固定为1，不可改变。默认值为None，表示使用默认的权重参数属性。具体用法请参见 cn_api_ParamAttr 。
- **bias_attr** (ParamAttr, 可选) - 指定偏置参数属性的对象。如果为False, 则表示每一个通道的偏移固定为0，不可改变。默认值为None，表示使用默认的偏置参数属性。具体用法请参见 cn_api_ParamAttr 。
- **data_format** (string, 可选) - 指定输入数据格式，数据格式可以为“NC"或者"NCL"。默认值：“NCL”。
- **name** (string, 可选) – InstanceNorm的名称, 默认值为None。更多信息请参见 Name 。

2. 形状：

- input: 形状为（批大小，通道数）的2-D Tensor 或（批大小, 通道数，长度）的3-D Tensor。
- output: 和输入形状一样。

3. 代码示例：

```
import paddle
import numpy as np

np.random.seed(123)
x_data = np.random.random(size=(2, 2, 3)).astype('float32')
x = paddle.to_tensor(x_data)
instance_norm = paddle.nn.InstanceNorm1D(2)
instance_norm_out = instance_norm(x)

print(instance_norm_out)
```

层归一化（Layer Normalization）

简介

针对BN不适用于深度不固定的网络（sequence长度不一致，如RNN），LN对深度网络某一层的所有神经元的输入按照以下公式进行normalization操作：

$$\mu^l = \frac{l}{H}\sum_{i=1}^{H} a_i^l$$

$$\sigma^l = \sqrt{\frac{l}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2}$$

LN中同层神经元的输入拥有相同的均值和方差，不同的输入样本有不同的均值和方差。

对于特征图 $x \in \mathbb{R}^{N\times C\times H\times W}$ ，LN对每个样本的 C,G,W 维度上的数据求均值和标准差，保留维度 N。其均值和标准差的计算公式如下：

$$\mu_n(x) = \frac{1}{CHW}\sum_{c=1}^{C}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{nchw}$$

$$\sigma_n(x) = \sqrt{\frac{1}{CHW}\sum_{c=1}^{C}\sum_{h=1}^{H}\sum_{w=1}^{W}\left(x_{nchw} - \mu_{n(x)^2+\varepsilon}\right)}$$

LN的优势是不需要批训练，在单条数据内部就能归一化。LN不依赖batch size和输入sequence的长度，因此可以用在batch size为1的网络和RNN中。LN用于RNN效果比较明显。

在paddlepaddle中使用

```
class paddle.nn.LayerNorm(normalized_shape, epsilon=1e-05, weight_attr=None, bias_attr=None, name=None)
```

1. 参数：

- **normalized_shape** (int 或 list 或 tuple) – 需规范化的shape，期望的输入shape为 `[*, normalized_shape[0], normalized_shape[1], ..., normalized_shape[-1]]` 。如果是单个整数，则此模块将在最后一个维度上规范化（此时最后一维的维度需与该参数相同）。
- **epsilon** (float, 可选) - 指明在计算过程中是否添加较小的值到方差中以防止除零。默认值：1e-05。
- **weight_attr** (ParamAttr|bool, 可选) - 指定权重参数属性的对象。如果为False固定为1，不进行学习。默认值为None, 表示使用默认的权重参数属性。具体用法请参见 ParamAttr 。
- **bias_attr** (ParamAttr, 可选) - 指定偏置参数属性的对象。如果为False固定为0，不进行学习。默认值为None，表示使用默认的偏置参数属性。具体用法请参见 ParamAttr 。
- **name** (string, 可选) – LayerNorm的名称, 默认值为None。更多信息请参见 Name 。

2. 形状：

- input: 2-D, 3-D, 4-D或5D 的Tensor。
- output: 和输入形状一样。

3. 代码示例：

```python
import paddle
import numpy as np

np.random.seed(123)
x_data = np.random.random(size=(2, 2, 2, 3)).astype('float32')
x = paddle.to_tensor(x_data)
layer_norm = paddle.nn.LayerNorm(x_data.shape[1:])
layer_norm_out = layer_norm(x)

print(layer_norm_out)
```