深度学习理论常见概念

熵(Entropy)

熵是用来衡量一组数据中的不确定性或信息量的指标。熵这个概念在信息表示中就是“信息含量”。对于一个事件来说，熵越大，也就是信息含量越大，其能实现的可能性越小，反之则亦然。

信息量

假设是一个离散随机变量，则概率分布可表示为 ，其中所含的信息量可表示为：

信息熵

当一个事件发生的概率为，那么根据公式(1)可以得知其信息量为，那么如果我们把这个事件的所有可能性罗列出来，就可以求得该事件信息量的期望。

假设事件  有  种可能性，其中的发生概率为 ，那么该事件的熵 为：

KL散度(相对熵)

KL散度是衡量两个事件/分布之间的不同。假设一个随机变量 有两个单独的概率分布和 ，那么如何计算这两个分布的区别，我们一般使用KL散度用于计算两个分布之间的不同。看名字似乎跟计算两个点之间的距离也很像，但实则不然，因为**KL散度不具备有对称性**。在距离上的不对称性就是。

在我们的实际模型应用中，经常就是一个表示样本的真实分布，另一个表示模型的预测分布。我们常采用的就是这两个东西计算loss值，也就是衡量预测的分布与真实分布之间的差异性，然后利用梯度下降的思想来最小化二者间的差异。具体可以查看生成模型理解专题。

对于一个随机变量  有两个单独的概率分布和 ，其 KL散度 为：

观察公式可以看出， 越接近，KL散度就越小，即loss越小。由于对数函数是凸函数，所以KL散度的值为非负数。

交叉熵(Cross Entropy)

对于KL散度的公式，我们可以进行以下的变形：

观察上式，可以看出，前半式表示的是的熵，那么可得:

对于中括号里的其实就是交叉熵，即：

因此当的熵为常数的时候，**可以得到： KL散度＝交叉熵➖信息熵。**

区别

* 概念上：交叉熵衡量的是在一个分布下编码另一个分布所需的信息量；而KL散度衡量的是两个概率分布之间的信息损失或差异。
* 计算公式：交叉熵的计算公式中不直接涉及真实分布P 相对于预测分布Q 的比率，而KL散度的公式直接涉及到这个比率P ( x )/ Q ( x )。
* 对称性：交叉熵是对称的，没有明确的方向性；KL散度是非对称的，明确区分了哪个是“真实”分布，哪个是“预测”或“近似”分布。
* 关系：二者之间有密切关系，实际上，KL散度可以被视为两个分布之间交叉熵和真实分布熵的差值：

KL散度和交叉熵

KL散度和交叉熵都可以用来作为模型的loss函数，但二者的使用场景不一样。在这里引申一下模型loss的含义：“**通过样本来计算模型分布与目标分布间的差异。**”，这就是KL散度的作用。但有时候我们的**目标分布会是常数**，也就是这个分布是已知且不变的，例如分类任务，这个时候我们就会使用交叉熵来衡量模型的预测分布与实际分布之间的差异。这是因为对应目标分布是参数的情况下，将KL散度作为loss等价于将交叉熵作为loss。

经过前面的推导，我们得到了：

我们将其作为损失函数的时候，通常令表示目标分布，表示神经网络的分布，这是因为目标分布通常为one-hot类型的分布，那么计算的时候，如果为0，那么我们根据极限可以算出为0，因此是可以计算的。反之，如果我们令是神经网络的分布，那么在我们的优化中，其值是不可能为0的，那么当为0时，是无法计算的。这也是没有意义的。

极大似然估计等价于交叉熵

在生成式模型理解里我们提到极大似然估计其实最后等价于最小化两个分布的KL散度，而在上面我们也提到当我们的目标分布不变时，KL散度等价于交叉熵，因此我们一个合理的猜测就是：对于分类任务而言，极大似然估计等价于交叉熵。下面我们给出证明：

对于二分类任务而言，我们使用极大似然估计的目标为：

那么取对数后：

那么我们求最小化的时候就对两边都取负号即可，这样也转换为了二分类的交叉熵损失。

对于多分类任务而言，我们使用极大似然估计的目标为：

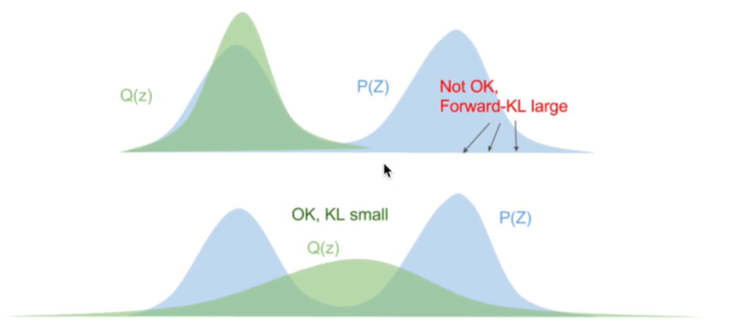
其中,为所有的样本数目，那么我们取对数后：

我们可以左右同时除以，同时令，那么：

这样，我们将多分类问题的极大似然估计也转换为了交叉熵。

forward KL散度和reverse KL散度

我们经常提到的KL散度都是指的forward KL散度，对于forward KL散度作为损失函数，其训练得到的分布通常是一个原始分布的一个平均分布，即如下图下面的图一样。即的分布会在波峰之间。



对于reverse KL散度则其训练得到的分布通常是拟合了其中的一个峰值。即 如上图的的上面的图一样。

理论分析：对于forward KL散度，其公式如下：

从公式直观来看，这个公式是对随机变量的每个取值上，这个值的加权平均，正向 KL 散度中的权值是 (实际上是算了期望)。

当随机变量的某个取值的概率 时，因为是权值，后面这一项多大多小已经不重要了，对 KL 散度不会产生影响，此时 KL 散度最小值是 O (也就意味着两个分布没有“距离,完全匹配),而是多少都无所谓。

相应地，当随机变量某个取值的概率时，此时后面的这一项就会对整体的 KL 散度大小产生影响，可以发现，作为分母，要尽量大一些才能保证 KL 散度小。

总体而言，对于正向 KL 散度，在大的地方，想让 KL 散度小，就需要 的值也尽量大； 在小的地方，对整体 KL 影响并不大 (因为 log 项本身分子很小，又乘了一个非常小的。换一种说法，要想使正向 KL 散度最小，则要求在p不为 0 的地方，q也尽量不为 0,所以正向 KL 散度被称为是 zero avoiding。此时得到的分布q是一个比较“宽”的分布。

而对于reverse KL散度，其公式如下：

公式与正向 KL 散度类似，不过这回权值成了。当随机变量的某个取值概率时，对的地方，我们完全可以忽略，因为对整体 KL 散度并没有什么影响。 而当随机变量的某个取值概率时，必须得保证在 小的地方也尽量小，才能使整体 KL 散度变小。 总体而言，对于反向 KL 散度，在小的地方，想让 KL 散度小，就需要的值也尽量小； 在大的地方，可以适当的忽略。换一种说法，要想使反向 KL 散度最小，则要求在 p为 0 的地方，q也尽量为 0, 所以反向 KL 散度被称为是 zero forcing。此时得到分布 q 是一个比较“窄” 的分布。最后的结果就很可能只是拟合了的一个峰。

MSELoss

对于MSELoss，我们可以对其做一个等效变换：

证明如下：

将Y带入等式左端得：

式中第二项为0，这是因为Y与其给定X的条件期望的偏差平均意义上是为0的。

即根据条件期望的线性，可以展开。

根据推导的式子，我们可以理解为式子本质上在说明MSE可以理解为两个部分的error。等式左边为MSE，等式右边为MSE的分解。其中分解的第一项于有关，说明是一个模型相关的error；第二项与无关，说明是数据集本身带来的error。

第二项很好解释，就是在数据集内相同情况下不同带来的错误，是数据集内禀的错误。其中，是最优预测（因为数据集存在噪声，相同可以对应不同  ，这个条件期望将这种方差平滑了）。

对于第一项，取决于模型的预测结果最优情况下也只能与 相等，因此这一项明显是模型引入的错误。换句话说，在带噪数据集中，最好的模型只能做到将第一项归零，但无法将第二项归零。

回归分析中MSE与MAE

在回归分析中，我们常用的评估函数有两个，分布是MSE与MAE，而对于MSE而言，其倾向于回归均值，MAE倾向于回归中位数。下面是其理论分析：

对于MSE而言，其计算方法为：

那当其达到最优时，也就是导数为0，即：

因此对于MSE而言，其倾向于回归均值。

对于MAE而言，其计算方法为：

那当其达到最优时，也就是导数为0，那么当时，其导数为1，当，其导数为-1，当时，其导数为0。因此我们想要导数为0，那么就需要是大于它的占一半，小于它的占一半，这样就可以使导数为0，达到最优了。

正则化：

正则化(Regularization)是机器学习中非常重要的一个技术点，因为它能够简单有效的减少泛化误差，在机器学习的应用实践中出现频率很高，尤其在深度学习模型中，由于其模型参数非常多很容易产生过拟合，使用正则化技术是一个行之有效的方法。

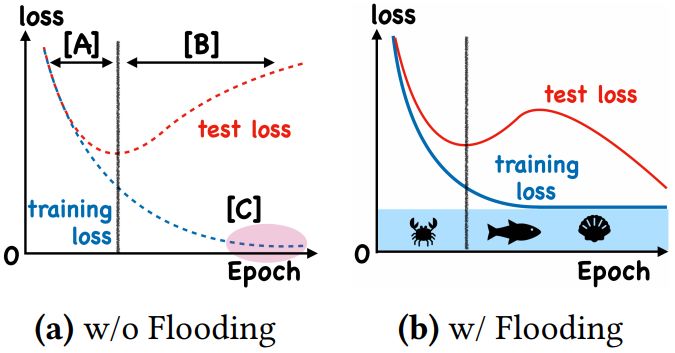
常规的正则化是通过在模型的损失函数中引入一个正则化项，来惩罚模型的复杂度。正则化项通常是模型参数的函数，它使得模型倾向于选择较简单的参数配置，从而减少过拟合的风险。不过，随着机器学习的发展，正则化技术的定义范围也有扩充，可以分为狭义正则化和广义正则化两个概念。

1.狭义正则化:在狭义上，正则化通常指的是通过在模型的目标函数中引入附加项来控制模型复杂度的技术。常见的狭义正则化方法包括L1正则化、L2 正则化(Ridge 正则化)、弹性网(Elastic Net)等。这些方法通过在目标函数中添加正则化项，对模型的参数进行约束，以减少过拟合现象，提高模型的泛化能力。

2.广义正则化:在广义上，正则化可以被理解为任何一种用于控制模型复杂度或增加模型鲁棒性的技术或策略。除了狭义正则化方法外，广义正则化还包括其他形式的技术，如数据增强（data augmentations）、噪声注入、特征选择、模型集成、随机失活(Dropout)、早停止(Early Stopping)、批正则化(Batch Normalization，BN) 、标签平滑（Label Smoothing）、Flooding方法等。这些技术都旨在通过不同的方式约束模型的复杂度，减少过拟合风险，并提高模型的泛化能力。

Label Smoothing方法：让分类之间的cluster更加紧凑，增加类间距离，减少类内距离，避免over high confidence的adversarial examples。

Flooding方法：当training loss大于一个阈值时，进行正常的梯度下降；当training loss低于阈值时，会反过来进行梯度上升，让training loss保持在一个阈值附近，让模型持续进行“random walk”，并期望模型能被优化到一个平坦的损失区域。代码：flood = (loss - b).abs() + b



因此，可以将狭义正则化视为正则化的一种特定形式，而广义正则化则更广泛地包括了各种控制模型复杂度和提高泛化能力的技术和策略。

Dropout

Dropout是一种正则化技术，主要用于防止神经网络的过拟合。在训练过程中，Dropout通过随机丢弃神经网络中的一些神经元来实现正则化。具体来说，在每个训练批次中，每个神经元以概率*p*被暂时移除（即其输出被置为0），而以概率1−*p*被保留。这种随机丢弃的过程可以看作是对多个子网络进行平均，从而减少模型的过拟合。

在没有Dropout时，激活值的期望为：

引入Dropout后，每个神经元以概率保留，并且**为了保持期望不变**，我们进行缩放，即激活值变为：

其中是一个随机向量，其每个元素独立地以概率取值为1，以概率*p*取值为0。因此，激活值的期望为：

由于

因此，保留下来的激活值的期望为：

因为我们进行了缩放补偿，使得均值保持不变。

在没有Dropout时，假设激活值的方差为：

在使用Dropout并进行缩放后，激活值的方差为：

由于

因此，保留下来的激活值的方差为：

因此，经过Dropout处理后，激活值的均值不变，但方差会增加一个因子。

MSAs 与Convs

Convs本质上就是一堆算子的结合，可视为一堆高通滤波器的叠加，其会不断强化高频信息，而MSAs由于其大量线性计算+聚合的特性，本质上是个低通滤波器，会不断强化图像的底层语义信息。这表明MSAs更擅长处理形状，而Convs更擅长处理纹理。因此Convs容易受到高频噪声的影响，而MSAs容易受到低频噪声的影响。因此MSAs和Convs是互补的，且可以在每一个stage的最后接入一个MSAs，使用Convs转换特征图，MSAs聚合转换后的特征图预测。