交叉熵误差损失函数

神经网络学习的过程，实质是求最优解的过程，更确切的说，是求最优权重的过程。为了反应最优解的状态，通过损失函数 (loss function) 来衡量。损失函数是表示神经网络性能的“恶劣程度”的指标，即当前的神经网络对监督数据在多大程度上不拟合，在多大程度上不一致。神经网络的最优解往往是求最小损失函数值的过程，因此使用梯度下降法求解。

1 交叉熵误差

在多分类任务中，一般使用交叉熵误差 (cross entropy error) 损失函数，其定义如公式(1)所示：

$$E = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{C} t_{(n,k)} \ln y_{(n,k)} \tag{1}$$

这里，假设数据有 N 个，下标 (n,k) 表示第 n 个数据的第 k 个元素的值，$t_{(n,k)}$ 是正确标签，$y_{(n,k)}$ 是神经网络的输出，C 表示分类类别数。除以 N，可以求单个数据的“平均损失函数”。通过这样的平均化，可以获得和训练数据的数量无关的统一指标。

为了更好地理解交叉熵误差损失函数，取 $N = 1$ 的情况来分析，那么如公式(2)所示：

$$E = -\sum_{k=1}^{C} t_k \ln y_k \tag{2}$$

t, y 均为一阶 C 维向量。t 用 one-hot 表示，即只有正确标签的索引为 1，其他均为 0。取 $t = (0,1)$，若 $y = (0.2, 0.6)$，那么损失值为 0.51，若若 $y = (0.6, 0.2)$，那么损失值为 1.61。也就是说，交叉熵误差损失值是由正确标签对应的输出结果决定。易得，交叉熵误差损失值随着正确标签对应的输出结果的增大而增大。

2 为什么要设定损失函数

神经网络的输出一般可以写作 $y = \mathrm{f}(w, x)$，w 表示训练权重，x 表示训练数据。对权重参数的损失函数求导，表示的是“如果稍微改变这个权重参数的值，损失函数的值会如何变化”。如果导数的值为负，通过使该权重参数向正方向改变，可以减小损失函数的值；反过来，如果导数的值为正，则通过使该权重参数向负方向改变，可以减小损失函数的值。不过，当导数的值为 0 时，无论权重参数向哪个方向变化，损失函数的值都不会改变，此时该权重参数的更新会停在此处。

为什么不能用分类精度作为指标呢？因为分类精度对微小的参数变化基本上没有什么反应，即便有反应，它的值也是不连续地、突然地变化。举个很简单的例子，100 个样品，正确样品的为 32 时，分类精度为 32%；正确样品的为 33 时，分类精度为 33%。在 32% 和33% 之间，分类精度是不连续的，对微小的参数变化基本上没有什么反应。

3 PyTorch 中的交叉熵损失函数

PyTorch 中的交叉熵损失函数定义如图1所示：

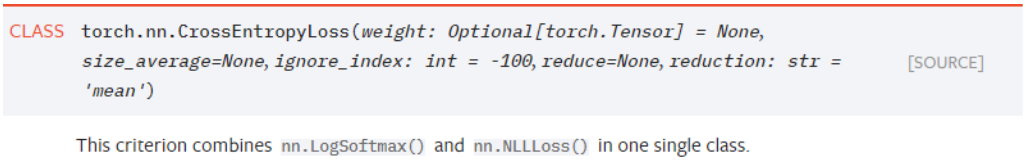

图 1 PyTorch 中的交叉熵损失函数。

PyTorch 为了计算的快速和稳定性，对标准交叉熵损失函数作了优化。这里，笔者实现了标准的交叉熵损失函数 (包括 one-hot 形式和标签形式)，已开源：https://github.com/mengzhu0308/PyTorch-CategoricalCrossentropy