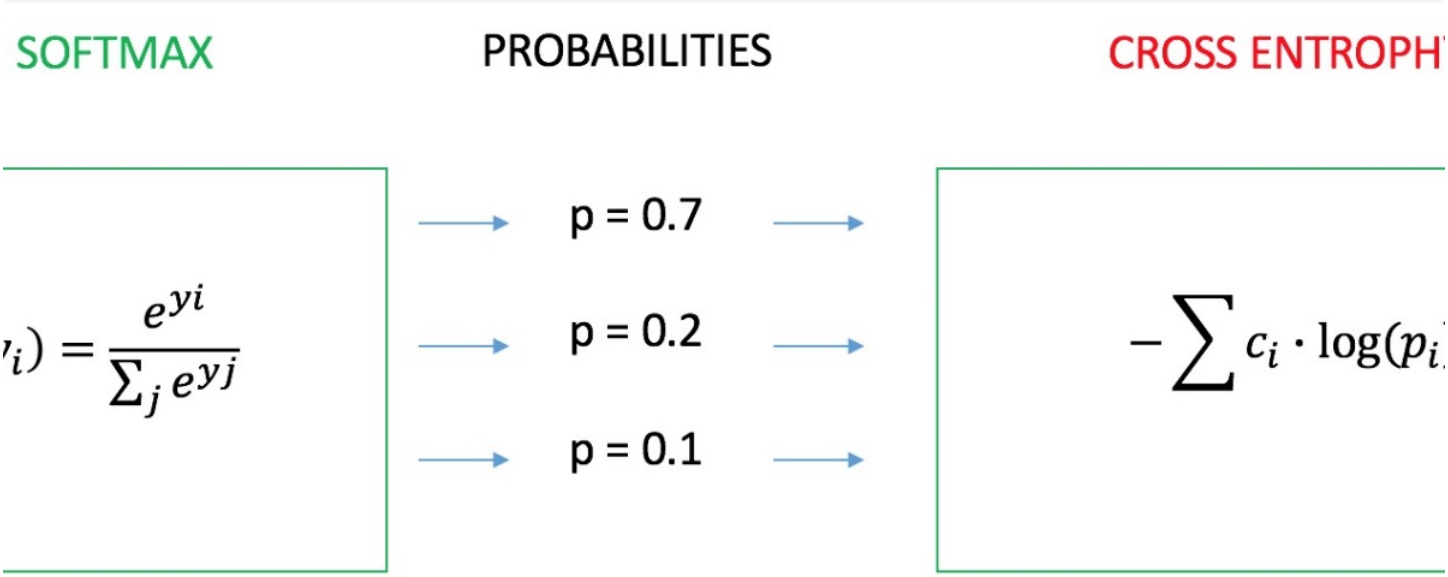


损失函数——交叉熵损失函数

小飞鱼
腾讯 算法工程师

关注他

64 人赞同了该文章

这篇文章中，讨论的Cross Entropy损失函数常用于分类问题中，但是为什么它会在分类问题中这么有效呢？我们先从一个简单的分类例子来入手。

预测政治倾向例子

我们希望根据一个人的年龄、性别、年收入等相互独立的特征，来预测一个人的政治倾向，有三种可预测结果：民主党、共和党、其他党。假设我们当前有两个逻辑回归模型（参数不同），这两个模型都是通过sigmoid的方式得到对于每个预测结果的概率值：

模型1：

COMPUTED	TARGETS	CORRECT?
0.3 0.3 0.4	0 0 1（民主党）	正确
0.3 0.4 0.3	0 1 0（共和党）	正确
0.1 0.2 0.7	1 0 0（其他党）	错误

模型1预测结果

模型1对于样本1和样本2以非常微弱的优势判断正确，对于样本3的判断则彻底错误。

模型2：

COMPUTED	TARGETS	CORRECT?
0.1 0.2 0.7	0 0 1（民主党）	正确
0.1 0.7 0.2	0 1 0（共和党）	正确
0.3 0.4 0.3	1 0 0（其他党）	错误

模型2预测结果

模型2对于样本1和样本2判断非常准确，对于样本3判断错误，但是相对来说没有错得太离谱。

好了，有了模型之后，我们需要通过定义损失函数来判断模型在样本上的表现了，那么我们可以定义哪些损失函数呢?

Classification Error（分类错误率）

最为直接的损失函数定义为： $classification\ error = \frac{count\ of\ error\ items}{count\ of\ all\ items}$

模型1： $classification\ error = \frac{1}{3}$

模型2： $classification\ error = \frac{1}{3}$

我们知道，**模型1**和**模型2**虽然都是预测错了1个，但是相对来说**模型2**表现得更好，损失函数值照理来说应该更小，但是，很遗憾的是， $classification\ error$ 并不能判断出来，所以这种损失函数虽然好理解，但表现不太好。

Mean Squared Error (均方误差)

均方误差损失也是一种比较常见的损失函数，其定义为： $MSE = \frac{1}{n}\sum_{i}^{n}(\hat{y_i} - y_i)^2$

模型1： $MSE = \frac{0.54 + 0.54 + 1.34}{3} = 0.81$

模型2： $MSE = \frac{0.14 + 0.14 + 0.74}{3} = 0.34$

我们发现，MSE能够判断出来**模型2**优于**模型1**，那为什么不采样这种损失函数呢？主要原因是逻辑回归配合MSE损失函数时，采用梯度下降法进行学习时，会出现模型一开始训练时，学习速率非常慢的情况（MSE损失函数）。

有了上面的直观分析，我们可以清楚的看到，对于分类问题的损失函数来说，分类错误率和平方和损失都不是很好的损失函数，下面我们来看一下交叉熵损失函数的表现情况。

Cross Entropy Error Function（交叉熵损失函数）

表达式

二分类

在二分的情况下，模型最后需要预测的结果只有两种情况，对于每个类别我们的预测得到的概率为p和1-p。此时表达式为：

$$J = -[y \cdot log(p) + (1-y) \cdot log(1-p)]$$

其中：
- y——表示样本的label，正类为1，负类为0
- p——表示样本预测为正的概率

多分类

多分类的情况实际上就是对二分类的扩展：

$$J = -\sum_{c=1}^{M} y_c \log(p_c)$$

其中：
- M——类别的数量；
- y——指示变量（0或1）,如果该类别和样本的类别相同就是1，否则是0；
- p——对于观测样本属于类别c的预测概率。

现在我们利用这个表达式计算上面例子中的损失函数值：

模型1：

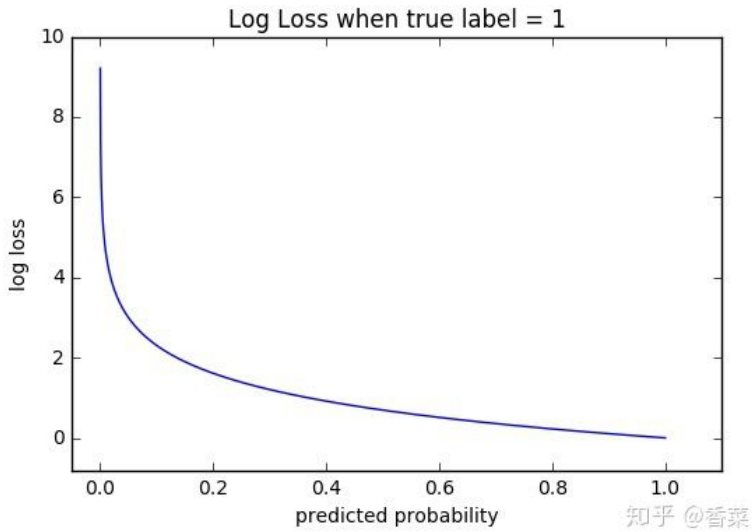
$$\begin{aligned} CEE &= -[0 \times log0.3 + 0 \times log0.3 + 1 \times log0.4] \\ &\quad -[0 \times log0.3 + 1 \times log0.4 + 0 \times log0.3] \\ &\quad -[1 \times log0.1 + 0 \times log0.2 + 0 \times log0.7] \\ &= 0.397 + 0.397 + 1 \\ &= 1.8 \end{aligned}$$

模型2：

$$\begin{aligned} CEE &= -[0 \times log0.1 + 0 \times log0.2 + 1 \times log0.7] \\ &\quad -[0 \times log0.1 + 1 \times log0.7 + 0 \times log0.2] \\ &\quad -[1 \times log0.3 + 0 \times log0.4 + 0 \times log0.3] \\ &= 0.15 + 0.15 + 0.397 \\ &= 0.697 \end{aligned}$$

可以发现，交叉熵损失函数可以捕捉到**模型1**和**模型2**预测效果的差异。

函数性质



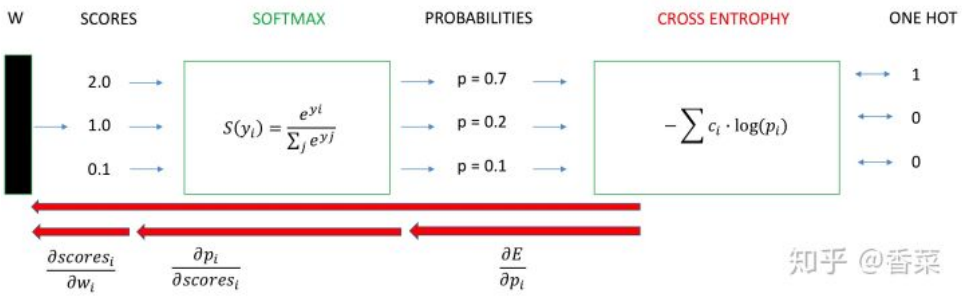
可以看出，该函数是凸函数，求导时能够得到全局最优值。

导函数性质

交叉熵损失函数经常用于分类问题中，特别是在神经网络做分类问题时，也经常使用交叉熵作为损失函数，此外，由于交叉熵涉及到计算每个类别的概率，所以交叉熵几乎每次都和softmax函数一起出现。

我们用神经网络最后一层输出的情况，来看一眼整个模型预测、获得损失和学习的流程：

1. 神经网络最后一层得到每个类别的得分**scores**；
2. 该得分经过softmax转换为概率输出；
3. 模型预测的类别概率输出与真实类别的one hot形式进行cross entropy损失函数的计算。



下面，我们来推导一下整个求导公式，我们将求导分成三个过程，即拆成三项偏导的乘积：

$$\frac{\partial J}{\partial w_i} = \frac{\partial J}{\partial p_i} \cdot \frac{\partial p_i}{\partial score_i} \cdot \frac{\partial score_i}{\partial w_i}$$

计算第一项： $\frac{\partial J}{\partial p_i}$

$$\begin{aligned}\frac{\partial J}{\partial p_i} &= \frac{\partial -[ylog(p)+(1-y)log(1-p)]}{\partial p_i} \\ &= -\frac{\partial y_i log p_i}{\partial p_i} - \frac{\partial (1-y_i)log(1-p_i)}{\partial p_i} \\ &= -\frac{y_i}{p_i} - [(1-y_i)\cdot\frac{1}{1-p_i}\cdot(-1)] \\ &= -\frac{y_i}{p_i} + \frac{1-y_i}{1-p_i}\end{aligned}$$

计算第二项： $\frac{\partial p_i}{\partial score_i}$

这一项要计算的是softmax函数对于score的导数，我们先回顾一下分数求导的公式：

> $$f(x)=\frac{g(x)}{h(x)}=\frac{g'(x)h(x)-g(x)h'(x)}{h^2(x)}$$

考虑k等于i的情况：

$$\begin{aligned}\frac{\partial p_i}{\partial score_i}&=\frac{(e^{y_i})'\cdot(\sum_i e^{y_i})-e^{y_i}\cdot(\sum_j e^{y_i})'}{(\sum_j e^{y_i})^2}\\&=\frac{e^{y_i}\cdot\sum_i e^{y_i}-(e^{y_i})^2}{(\sum_j e^{y_i})^2}\\&=\frac{e^{y_i}}{\sum_j e^{y_i}}-\frac{(e^{y_i})^2}{(\sum_j e^{y_i})^2}\\&=\frac{e^{y_i}}{\sum_j e^{y_i}}\cdot(1-\frac{e^{y_i}}{\sum_j e^{y_i}})\\&=\sigma(y_i)(1-\sigma(y_i))\end{aligned}$$

考虑k不等于i的情况：

$$\begin{aligned}\frac{\partial p_k}{\partial score_i}&=\frac{(e^{y_k})'\cdot(\sum_i e^{y_i})-e^{y_i}\cdot(\sum_j e^{y_i})'}{(\sum_j e^{y_i})^2}\\&=\frac{0\cdot\sum_i e^{y_i}-(e^{y_i})\cdot(e^{y_k})}{(\sum_j e^{y_i})^2}\\&=-\frac{e^{y_i}\cdot e^{y_k}}{(\sum_j e^{y_i})^2}\\&=-\frac{e^{y_i}}{\sum_j e^{y_i}}\cdot\frac{e^{y_k}}{\sum_j e^{y_i}}\\&=-\sigma(y_i)\cdot\sigma(y_k)\\&=\sigma(y_k)\cdot(1-\sigma(y_i))\end{aligned}$$

综上可得softmax损失函数的求导结果：

$$\frac{\partial p_k}{\partial score_i}=\begin{cases}\sigma(y_i)(1-\sigma(y_i)) & if\ k=i\\ \sigma(y_k)\cdot(1-\sigma(y_i)) & if\ k\neq i\end{cases}$$

则可统一写成：

$$\frac{\partial p_i}{\partial score_i}=\sigma(y_i)(1-\sigma(y_i))$$

计算第三项： $\frac{\partial score_i}{\partial w_i}$

一般来说，scores是输入的线性函数作用的结果，所以有：

$$\frac{\partial score_i}{\partial w_i}=x_i$$

计算结果 $\frac{\partial J}{\partial w_i}$

$$\begin{aligned}\frac{\partial J}{\partial w_i}&=\frac{\partial J}{\partial p_i}\cdot\frac{\partial p_i}{\partial score_i}\cdot\frac{\partial score_i}{\partial w_i}\\&=[-\frac{y_i}{\sigma(y_i)}+\frac{1-y_i}{1-\sigma(y_i)}]\cdot\sigma(y_i)(1-\sigma(y_i))\cdot x_i\\&=[-\frac{y_i}{\sigma(y_i)}\cdot\sigma(y_i)\cdot(1-\sigma(y_i))+\frac{1-y_i}{1-\sigma(y_i)}\cdot\sigma(y_i)\cdot(1-\sigma(y_i))]\cdot x_i\\&=[-y_i+y_i\cdot\sigma(y_i)+\sigma(y_i)-y_i\cdot\sigma(y_i)]\cdot x_i\\&=[\sigma(y_i)-y_i]\cdot x_i\end{aligned}$$

可以看到，我们得到了一个非常漂亮的结果，所以，使用交叉熵损失函数，不仅可以很好的衡量模型的效果，又可以很容易的的进行求导计算。

优缺点

优点

在用梯度下降法做参数更新的时候，模型学习的速度取决于两个值：一、**学习率**；二、**偏导值**。其中，学习率是我们需要设置的超参数，所以我们重点关注偏导值。从上面的式子中，我们发现，偏导值的大小取决于 x_i 和 $[\sigma(y_i)-y_i]$，我们重点关注后者，后者的大小值反映了我们模型的错误程度，该值越大，说明模型效果越差，但是该值越大同时也会使得偏导值越大，从而模型学习速度更快。所以，使用逻辑函数得到概率，并结合交叉熵当损失函数时，在模型效果差的时候学习速度比较快，在模型效果好的时候学习速度变慢。

参考

[1]. 神经网络的分类模型 LOSS 函数为什么要用 CROSS ENTROPY

[2]. Softmax as a Neural Networks Activation Function

[3]. A Gentle Introduction to Cross-Entropy Loss Function,

编辑于 2018-09-29

机器学习