softmax函数与交叉熵函数的配合使用

飞狗
茫然的低欲青年

Softmax

Softmax在机器学习中有非常广泛的应用，但是刚刚接触机器学习的人可能对Softmax的特点以及好处并不理解，其实你了解了以后就会发现，Softmax计算简单，效果显著，非常好用。

我们先来直观看一下，Softmax究竟是什么意思

我们知道max，假如说我有两个数，a和b，并且a>b，如果取max，那么就直接取a，没有第二种可能

但有的时候我不想这样，因为这样会造成分值小的那个饥饿。所以我希望分值大的那一项经常取到，分值小的那一项也偶尔可以取到，那么我用softmax就可以了 现在还是a和b，a>b，如果我们 取按照softmax来计算取a和b的概率，那a的softmax值大于b的，所以a会经常取到，而b也会偶尔 取到，概率跟它们本来的大小有关。所以说不是max，而是 Soft max 那各自的概率究竟是多少 呢，我们下面就来具体看一下

定义

假设我们有一个数组，V，Vi表示V中的第i个元素，那么这个元素的Softmax值就是

$$S_i = \frac{e^{V_i}}{\sum_j e^{V_j}}$$

也就是说，是该元素的指数，与所有元素指数和的比值

这个定义可以说非常的直观，当然除了直观朴素好理解以外，它还有更多的优点

1.计算与标注样本的差距

在神经网络的计算当中，我们经常需要计算按照神经网络的正向传播计算的分数S1，和按照正确标 注计算的分数S2，之间的差距，计算Loss，才能应用反向传播。Loss定义为交叉熵

$$L_i = -log(\frac{e^{f_{y_i}}}{\sum_j e^j})$$

> 取log里面的值就是这组数据正确分类的Softmax值，它占的比重越大，这个样本的Loss也就越小，这种定义符合我们的要求

2.计算上非常非常的方便

当我们对分类的Loss进行改进的时候，我们要通过梯度下降，每次优化一个step大小的梯度 我们定义选到yi的概率是

$$P_{y_i} = \frac{e^{f_{y_i}}}{\sum_j e^j}$$

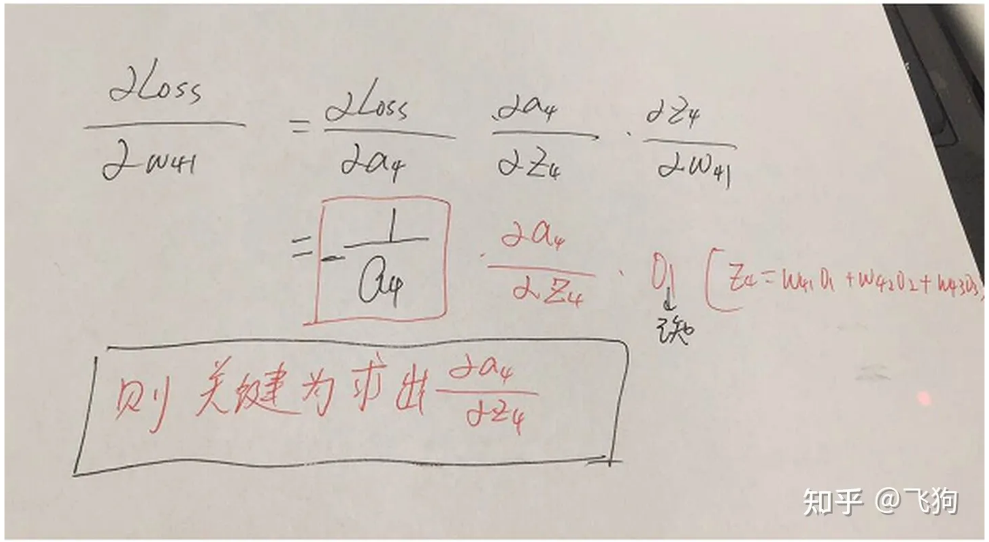
使用交叉熵作为损失函数

$$Loss = -\sum_i y_i ln a_i$$

为了形式化说明，我这里认为训练数据的真实输出为第j个为1，其它均为0！

那么Loss就变成了$Loss=-y_jlna_j$,累和已经去掉了

其中$y_j=1$,那么形式变为$Loss=-lna_j$



这里分为俩种情况：

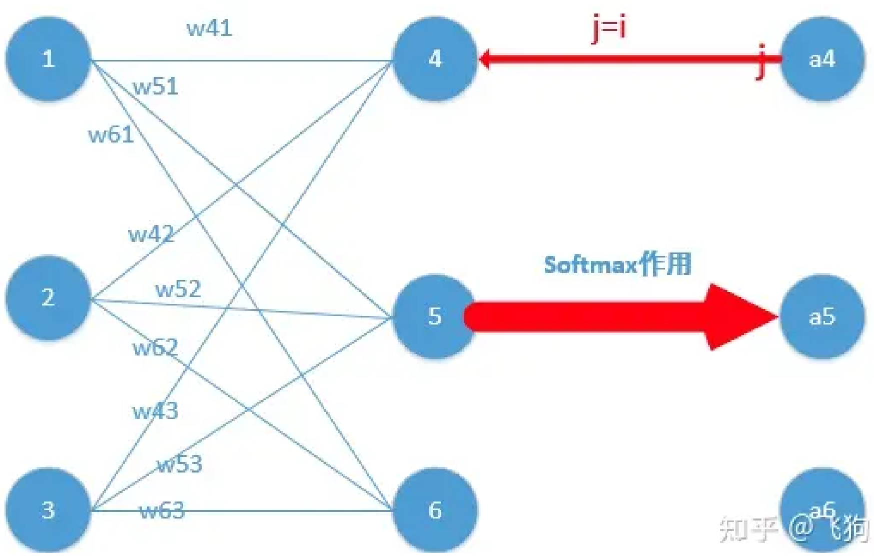
$$\text{if} \quad j = i:$$

$$\frac{\partial a_j}{\partial z_i} = \frac{\partial}{\partial z_i}\left(\frac{e^{z_j}}{\sum_k e^{z_k}}\right)$$

$$= \frac{(e^{z_j})' \cdot \sum_k e^{z_k} - e^{z_j} \cdot e^{z_j}}{\left(\sum_k e^{z_k}\right)^2}$$

$$= \frac{e^{z_j}}{\sum_k e^{z_k}} - \frac{e^{z_j}}{\sum_k e^{z_k}} \cdot \frac{e^{z_j}}{\sum_k e^{z_k}} = a_j(1 - a_j)$$

j=i对应例子里就是如下图所示：



那么由上面求导结果再乘以交叉熵损失函数求导

$Loss = -lna_j$，它的导数为$-\frac{1}{a_j}$,应用链式法则与上面$a_j(1-a_j)$相乘为$a_j - 1$，**形式非常简单，这说明我只要正向求一次得出结果，然后反向传梯度的时候，只需要将它结果减1即可，**

$$\frac{\partial L_i}{\partial f_{y_i}} = \frac{\partial(-\ln(\frac{e^{f_{y_i}}}{\sum_j e^j}))}{\partial f_{y_i}} = P_{f_{y_i}} - 1$$

第二种情况为：

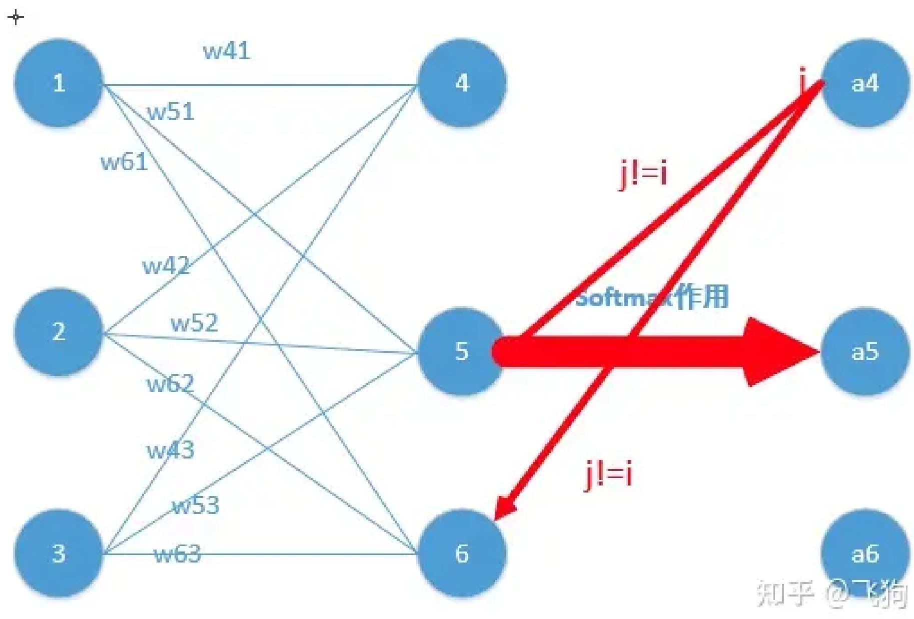
$$\text{if} \quad j \neq i:$$

$$\frac{\partial a_j}{\partial z_i} = \frac{\partial}{\partial z_i}\left(\frac{e^{z_j}}{\sum_k e^{z_k}}\right)$$

$$= \frac{0 \cdot \sum_k e^{z_k} - e^{z_j} \cdot e^{z_i}}{\left(\sum_k e^{z_k}\right)^2}$$

$$= -\frac{e^{z_j}}{\sum_k e^{z_k}} \cdot \frac{e^{z_i}}{\sum_k e^{z_k}} = -a_j a_i$$

这里对应我的例子图如下，我这时对的是j不等于i，往前传：



那么由上面求导结果再乘以交叉熵损失函数求导

$Loss = -lna_j$，它的导数为$-\frac{1}{a_j}$,与上面$-a_ja_i$相乘为a_i（**形式非常简单，这说明我只要正向求一次得出结果，然后反向传梯度的时候，只需要将它结果保存即可，后续例子会讲到**）**这里就求出了除4之外的其它所有结点的偏导，然后利用链式法则继续传递过去即可！我们的问题也就解决了！**

举个例子，通过若干层的计算，最后得到的某个训练样本的向量的分数是[1, 5, 3], 那么概率分别就是[0.015,0.866,0.117],如果这个样本正确的分类是第二个的话，那么计算出来的偏导就是[0.015,0.866−1,0.117]=[0.015,−0.134,0.117]，然后再根据这个进行back propagation就可以了

作者：忆臻
链接：zhihu.com/question/2376...
来源：知乎

编辑于 2021-03-07 22:48