TensorFlow四种Cross Entropy算法实现和应用

tobe

tobe-陈迪豪 2016-11-30 11:06:42 举报

深入理解交叉熵算法定义和TensorFlow深度学习框架的函数实现

交叉熵介绍

交叉熵（Cross Entropy）是Loss函数的一种（也称为损失函数或代价函数），用于描述模型预测值与真实值的差距大小，常见的Loss函数就是均方平方差（Mean Squared Error），定义如下。

$$C = \frac{(y-a)^2}{2}$$

平方差很好理解，预测值与真实值直接相减，为了避免得到负数取绝对值或者平方，再做平均就是均方平方差。注意这里预测值需要经过sigmoid激活函数，得到取值范围在0到1之间的预测值。

平方差可以表达预测值与真实值的差异，但在分类问题种效果并不如交叉熵好，原因可以参考这篇博文 https://jamesmccaffrey.wordpress.com/2013/11/05/why-you-should-use-cross-entropy-error-instead-of-classification-error-or-mean-squared-error-for-neural-network-classifier-training/ 。交叉熵的定义如下，截图来自 https://hit-scir.gitbooks.io/neural-networks-and-deep-learning-zh_cn/content/chap3/c3s1.html 。

神经元的输出为$a = \sigma(z)$，这里$z = \sum_j w_j x_j + b$。我们定义这个神经元的交叉熵代价函数为：

$$C = -\frac{1}{n} \sum_x [y \ln a + (1-y) \ln(1-a)],$$

这里n是训练数据的个数，这个加和覆盖了所有的训练输入x，y是期望输出。

上面的文章也介绍了交叉熵可以作为Loss函数的原因，首先是交叉熵得到的值一定是正数，其次是预测结果越准确值越小，注意这里用于计算的“a”也是经过sigmoid激活的，取值范围在0到1。如果label是1，预测值也是1的话，前面一项y * ln(a)就是1 * ln(1)等于0，后一项(1 - y) * ln(1 - a)也就是0 * ln(0)等于0，Loss函数为0，反之Loss函数为无限大非常符合我们对Loss函数的定义。

这里多次强调sigmoid激活函数，是因为在多目标或者多分类的问题下有些函数是不可用的，而TensorFlow本身也提供了多种交叉熵算法的实现。

TensorFlow的交叉熵函数

TensorFlow针对分类问题，实现了四个交叉熵函数，分别是

- tf.nn.sigmoid_cross_entropy_with_logits、
- tf.nn.softmax_cross_entropy_with_logits、
- tf.nn.sparse_softmax_cross_entropy_with_logits和
- tf.nn.weighted_cross_entropy_with_logits，详细内容参考API文档
-

https://www.tensorflow.org/versions/master/api_docs/python/nn.html#sparse_softmax_cross_entropy_with_logits

sigmoid_cross_entropy_with_logits详解

我们先看sigmoid_cross_entropy_with_logits，为什么呢，因为它的实现和前面的交叉熵算法定义是一样的，也是TensorFlow最早实现的交叉熵算法。这个函数的输入是logits和targets，logits就是神经网络模型中的 W * X矩阵，注意不需要经过sigmoid，而targets的shape和logits相同，就是正确的label值，例如这个模型一次要判断100张图是否包含10种动物，这两个输入的shape都是[100, 10]。注释中还提到这10个分类之间是独立的、不要求是互斥，这种问题我们成为多目标，例如判断图片中是否包含10种动物，label值可以包含多个1或0个1，还有一种问题是多分类问题，例如我们对年龄特征分为5段，只允许5个值有且只有1个值为1，这种问题可以直接用这个函数吗？答案是不可以，我们先来看看sigmoid_cross_entropy_with_logits的代码实现吧。

For brevity, let `x = logits, z = targets`. The logistic loss is

```
  z * -log(sigmoid(x)) + (1 - z) * -log(1 - sigmoid(x))
= z * -log(1 / (1 + exp(-x))) + (1 - z) * -log(exp(-x) / (1 + exp(-x)))
= z * log(1 + exp(-x)) + (1 - z) * (-log(exp(-x)) + log(1 + exp(-x)))
= z * log(1 + exp(-x)) + (1 - z) * (x + log(1 + exp(-x))
= (1 - z) * x + log(1 + exp(-x))
= x - x * z + log(1 + exp(-x))
```

可以看到这就是标准的Cross Entropy算法实现，对W * X得到的值进行sigmoid激活，保证取值在0到1之间，然后放在交叉熵的函数中计算Loss。对于二分类问题这样做没问题，但对于前面提到的多分类，例如年轻取值范围在0~4，目标值也在0~4，这里如果经过sigmoid后预测值就限制在0到1之间，而且公式中的1 - z就会出现负数，仔细想一下0到4之间还不存在线性关系，如果直接把label值带入计算肯定会有非常大的误差。因此对于多分类问题是不能直接代入的，那其实我们可以灵活变通，把5个年龄段的预测用onehot encoding变成5维的label，训练时当做5个不同的目标来训练即可，但不保证只有一个为1，对于这类问题TensorFlow又提供了基于Softmax的交叉熵函数。

softmax_cross_entropy_with_logits详解

Softmax本身的算法很简单，就是把所有值用e的n次方计算出来，求和后算每个值占的比率，保证总和为1，一般我们可以认为Softmax出来的就是confidence也就是概率，算法实现如下。

For each batch `i` and class `j` we have

```
softmax = exp(logits) / reduce_sum(exp(logits), dim)
```

softmax_cross_entropy_with_logits和sigmoid_cross_entropy_with_logits很不一样，输入是类似的logits和lables的shape一样，但这里要求分类的结果是互斥的，保证只有一个字段有值，例如CIFAR-10中图片只能分一类而不像前面判断是否包含多类动物。想一下问什么会有这样的限制？在函数头的注释中我们看到，这个函数传入的logits是unscaled的，既不做sigmoid也不做softmax，因

为函数实现会在内部更高效得使用softmax，对于任意的输入经过softmax都会变成和为1的概率预测值，这个值就可以代入变形的Cross Entroy算法- y * ln(a) - (1 - y) * ln(1 - a)算法中，得到有意义的Loss值了。如果是多目标问题，经过softmax就不会得到多个和为1的概率，而且label有多个1也无法计算交叉熵，因此这个函数只适合单目标的二分类或者多分类问题，TensorFlow函数定义如下。

```
tf.nn.softmax_cross_entropy_with_logits(logits, labels, dim=-1, name=None)
```

Computes softmax cross entropy between `logits` and `labels`.

Measures the probability error in discrete classification tasks in which the classes are mutually exclusive (each entry is in exactly one class). For example, each CIFAR-10 image is labeled with one and only one label: an image can be a dog or a truck, but not both.

再补充一点，对于多分类问题，例如我们的年龄分为5类，并且人工编码为0、1、2、3、4，因为输出值是5维的特征，因此我们需要人工做onehot encoding分别编码为00001、00010、00100、01000、10000，才可以作为这个函数的输入。理论上我们不做onehot encoding也可以，做成和为1的概率分布也可以，但需要保证是和为1，和不为1的实际含义不明确，TensorFlow的C++代码实现计划检查这些参数，可以提前提醒用户避免误用。

sparse_softmax_cross_entropy_with_logits详解

sparse_softmax_cross_entropy_with_logits是softmax_cross_entropy_with_logits的易用版本，除了输入参数不同，作用和算法实现都是一样的。前面提到softmax_cross_entropy_with_logits的输入必须是类似onehot encoding的多维特征，但CIFAR-10、ImageNet和大部分分类场景都只有一个分类目标，label值都是从0编码的整数，每次转成onehot encoding比较麻烦，有没有更好的方法呢？答案就是用sparse_softmax_cross_entropy_with_logits，它的第一个参数logits和前面一样，shape是[batch_size, num_classes]，而第二个参数labels以前也必须是[batch_size, num_classes]否则无法做Cross Entropy，这个函数改为限制更强的[batch_size]，而值必须是从0开始编码的int32或int64，而且值范围是[0, num_class)，如果我们从1开始编码或者步长大于1，会导致某些label值超过这个范围，代码会直接报错退出。这也很好理解，TensorFlow通过这样的限制才能知道用户传入的3、6或者9对应是哪个class，最后可以在内部高效实现类似的onehot encoding，这只是简化用户的输入而已，如果用户已经做了onehot encoding那可以直接使用不带“sparse”的softmax_cross_entropy_with_logits函数。

weighted_sigmoid_cross_entropy_with_logits详解

weighted_sigmoid_cross_entropy_with_logits是sigmoid_cross_entropy_with_logits的拓展版，输入参数和实现和后者差不多，可以多支持一个pos_weight参数，目的是可以增加或者减小正样本在算Cross Entropy时的Loss。实现原理很简单，在传统基于sigmoid的交叉熵算法上，正样本算出的值乘以某个系数接口，算法实现如下。

For brevity, let `x = logits, z = targets, q = pos_weight`. The loss is:

```
  qz * -log(sigmoid(x)) + (1 - z) * -log(1 - sigmoid(x))
= qz * -log(1 / (1 + exp(-x))) + (1 - z) * -log(exp(-x) / (1 + exp(-x)))
= qz * log(1 + exp(-x)) + (1 - z) * (-log(exp(-x)) + log(1 + exp(-x)))
= qz * log(1 + exp(-x)) + (1 - z) * (x + log(1 + exp(-x))
= (1 - z) * x + (qz +  1 - z) * log(1 + exp(-x))
= (1 - z) * x + (1 + (q - 1) * z) * log(1 + exp(-x))
```

总结

这就是TensorFlow目前提供的有关Cross Entropy的函数实现，用户需要理解多目标和多分类的场景，根据业务需求（分类目标是否独立和互斥）来选择基于sigmoid或者softmax的实现，如果使用sigmoid目前还支持加权的实现，如果使用softmax我们可以自己做onehot coding或者使用更易用的sparse_softmax_cross_entropy_with_logits函数。

TensorFlow提供的Cross Entropy函数基本cover了多目标和多分类的问题，但如果同时是多目标多分类的场景，肯定是无法使用softmax_cross_entropy_with_logits，如果使用sigmoid_cross_entropy_with_logits我们就把多分类的特征都认为是独立的特征，而实际上他们有且只有一个为1的非独立特征，计算Loss时不如Softmax有效。这里可以预测下，未来TensorFlow社区将会实现更多的op解决类似的问题，我们也期待更多人参与TensorFlow贡献算法和代码 :)

tobe
tobe-陈迪豪
小米深度学习工程师