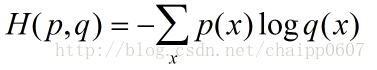
## 交叉熵

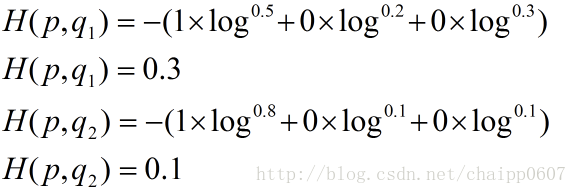
## 1. 交叉熵的意义

交叉熵刻画的是实际输出（概率）与期望输出（概率）的距离，也就是交叉熵的值越小，两个概率分布就越接近。假设概率分布p为期望输出，概率分布q为实际输出，H(p,q)为交叉熵，则：



这个公式如何表征距离呢，举个例子：

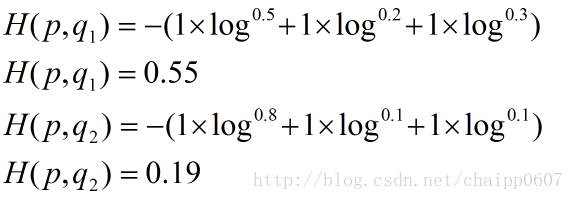
假设N=3，期望输出为p=(1,0,0)，实际输出q1=(0.5,0.2,0.3)，q2=(0.8,0.1,0.1)，那么：



很显然，q2与p更为接近，它的交叉熵也更小。

除此之外，交叉熵还有另一种表达形式，还是使用上面的假设条件：

其结果为：



## 2. softmax\_cross\_entropy\_with\_logits

softmax\_cross\_entropy\_with\_logits函数集成了softmax与交叉熵的计算，那么这个函数到底如何使用呢，下面来看实验。

实验1：

import tensorflow as tf

logits=tf.constant([[1.0,2.0,3.0],[1.0,2.0,3.0],[1.0,2.0,3.0]])

y\_=tf.constant([[0.0,0.0,1.0],[0.0,0.0,1.0],[0.0,0.0,1.0]])

result = tf.nn.softmax\_cross\_entropy\_with\_logits(logits=logits, labels= y\_)

with tf.Session() as sess:

result\_o = sess.run(result)

print(result\_o)

打印结果为：

[ 0.40760595 0.40760595 0.40760595]

可见softmax\_cross\_entropy\_with\_logits返回的是一个数组，一般tensorflow中求交叉熵会加上tf.reduce\_mean用来求平均值，这里为0.407606。

# 'x' is [[1., 2.]

# [3., 4.]]

tf.reduce\_mean(x) ==> 2.5 #如果不指定第二个参数，那么就在所有的元素中取平均值

tf.reduce\_mean(x, 0) ==> [2., 3.] #指定第二个参数为0，则第一维的元素取平均值，即每一列求平均值

tf.reduce\_mean(x, 1) ==> [1.5, 3.5] #指定第二个参数为1，则第二维的元素取平均值，即每一行求平均值

那么softmax\_cross\_entropy\_with\_logits中到底做了些什么事情呢？

**1.对logits进行softmax运算**

logits=tf.constant([[1.0,2.0,3.0],[1.0,2.0,3.0],[1.0,2.0,3.0]])

y=tf.nn.softmax(logits)

y为：

[[ 0.09003057 0.24472848 0.66524094]

[ 0.09003057 0.24472848 0.66524094]

[ 0.09003057 0.24472848 0.66524094]]

**2.两个矩阵对应位置相乘**

y\_=tf.constant([[0.0,0.0,1.0],[0.0,0.0,1.0],[0.0,0.0,1.0]])

matrix = y\_\*tf.log(y)

matrix为：

[[-0. -0. -0.40760601]

[-0. -0. -0.40760601]

[-0. -0. -0.40760598]]

**3.按行求和**

cross\_entropy = -tf.reduce\_sum(matrix,axis=1)

cross\_entropy为：

[ 0.40760601 0.40760601 0.40760598]

**完整代码如下：**

import tensorflow as tf

logits=tf.constant([[1.0,2.0,3.0],[1.0,2.0,3.0],[1.0,2.0,3.0]])

y=tf.nn.softmax(logits)

y\_=tf.constant([[0.0,0.0,1.0],[0.0,0.0,1.0],[0.0,0.0,1.0]])

matrix = y\_\*tf.log(y)

cross\_entropy = -tf.reduce\_sum(y\_\*tf.log(y),axis=1)

with tf.Session() as sess:

softmax=sess.run(y)

print(softmax)

matrix\_result = sess.run(matrix)

print(matrix\_result)

loss = sess.run(cross\_entropy)

print(loss)

**手动计算softmax交叉熵容易出现的问题：**

1.当log(x)中x为0时，log(x)=NAN，所以要保证x的值不为0，有时候需要对x做一个额外的操作：

tf.clip\_by\_value(x, 1e-10, 1.0)

也就是将x的值的范围设定为1e-10到1之间，避免了0的出现。

2.在求softmax的时候，会进行exp(x)操作，如果x值过大，容易溢出，结果为无穷大inf，需要做归一化处理，例如对如下数据做exp操作。

logits=tf.constant([10.281,-80390.062,14042.887,175657.875,-70161.148,42373.883,-242375.438,297174.312,-29638.275,107964.125])

y = tf.exp(logits)

输出的结果为：

[ 29173.03515625 ,0. ,inf ,inf ,0. ,inf ,0. ,inf ,0. ,inf]

所以尽量用官方提供的函数进行交叉熵计算。

## 3. sigmoid\_cross\_entropy\_with\_logits

import tensorflow as tf

z = 0.8

x = 1.6

cross\_entropy4 = tf.nn.sigmoid\_cross\_entropy\_with\_logits(labels=z, logits=x)

cross\_entropy5 = - z \* tf.log(tf.nn.sigmoid(x)) - (1-z) \* tf.log(1-tf.nn.sigmoid(x))

with tf.Session() as sess:

cross\_entropy4\_ = sess.run(cross\_entropy4)

cross\_entropy5\_ = sess.run(cross\_entropy5)

print(cross\_entropy4\_)

print(cross\_entropy5\_)

cross\_entropy4 与cross\_entropy5 的值是相等的。sigmoid\_cross\_entropy\_with\_logits的具体实现就是 

其中  就是labels，是一个概率值。  是logits,  才是概率。sigmoid\_cross\_entropy\_with\_logits函数求的就是两点分布 和两点分布  之间的交叉熵。

如果sigmoid\_cross\_entropy\_with\_logits的labels和logits是相同shape的张量,输出结果则是与labels和logits有相同shape的张量。不同于softmax系列函数是张量中向量与向量间的运算。sigmoid\_cross\_entropy\_with\_logits函数则是张量中标量与标量间的运算。