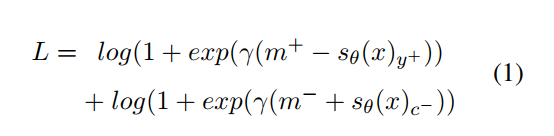
针对分类任务，排序loss，因为我们的三类是无效语义，肯定和否定三类。针对无效语义，因为无效语义数据之间的差别非常大，在训练的时候不容易学习到无效语义的特征，但是我们可以学习到肯定和否定语句的特征。排序loss的思想是，把三分类变为二分类，就是肯定和否定。模型的输出是关于肯定和否定的概率分数f1和f2，当f1<0&f2<0的时候，认为分类结果是无效语义，loss=0，否则的话loss的计算方式为：



更具体的理论可以参考下面的论文。下面是实现代码

def ranking\_loss(labels, logits):

lm = tf.constant(1.0)

m\_plus = tf.constant(2.5)

m\_minus = tf.constant(1.0)

L = tf.constant(0.0)

i = tf.constant(0)

cond = lambda i, l: tf.less(i, config.batch\_size)

def d\_loss(logits):

\_, cminus\_indices = tf.nn.top\_k(logits, k=1)

sminus = logits[cminus\_indices[0]]

l = tf.log((1.0+tf.exp((lm\*(m\_minus+sminus)))))

return l

def u\_loss(logits, cplus):

cplus = tf.cond(tf.equal(cplus, 1), lambda: 0, lambda: 1)

sminus = tf.cond(tf.equal(cplus, 0),

lambda: logits[1], lambda: logits[0])

splus = logits[cplus]

l = tf.log((1.0+tf.exp((lm\*(m\_plus-splus))))) + \

tf.log((1.0+tf.exp((lm\*(m\_minus+sminus)))))

return l

def loop\_body(i, L):

cplus = labels[i]

l = tf.cond(tf.equal(cplus, 0), lambda:d\_loss(logits[i]), lambda:u\_loss(logits[i], cplus))

return [tf.add(i, 1), tf.add(L, l)]

\_, L = tf.while\_loop(cond, loop\_body, loop\_vars=[i, L])

nbatch = tf.to\_float(config.batch\_size)

L = L/nbatch

return L

但是当时我们做的实验，排序loss并不比直接按照三分类效果好。

参考论文：Classifying Relations by Ranking with Convolutional Neural Networks