详解cross-entropy交叉熵损失函数及其反向传播

**交叉熵函数的定义：**

假设在一个多分类的问题中，用表示为实际概率，为计算得出的预测概率，常有：

若由softmax函数计算得出，则天然满足：

于是，交叉熵损失函数的定义为：

log表示以e为底数的自然对数

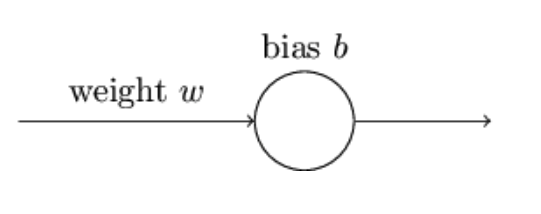
**反向传播梯度：**

**交叉熵的损失为：**

**c对s的梯度向量为：**

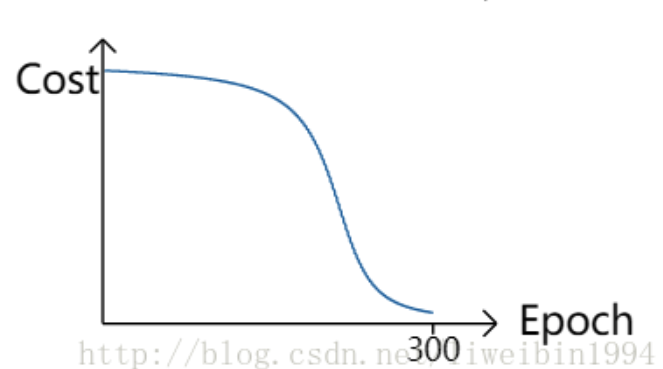
**那么为什么要用交叉熵损失函数呢，而不使用平方差损失函数：**

我们看一下下面的一个简单的例子：



上图是一个只有一个神经元的模型。我们希望输入为1的时候，我们希望输入为1的时候，模型会输出0（也就是说，我们只有一个样本(x=1, y=0)）。假设我们随机初始化权重参数w=2.0，偏置参数b=2.0。激活函数为sigmoid函数，所以模型的第一次输出为：

可见，模型的第一次输出跟标签相差很大，很错误的一个输出。然后我们不断地使用梯度下降算法更新参数，重复训练。于是我们得到了下面的这个图：

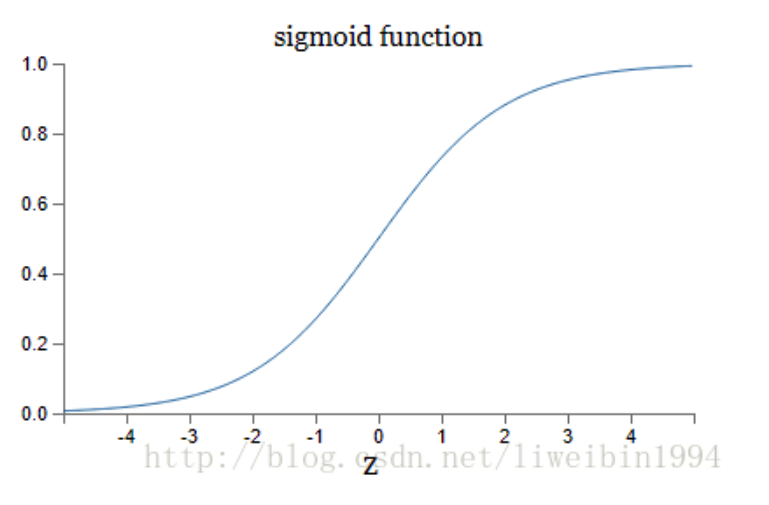


从图中可以看出，随着训练次数的增加，模型的输出越来越接近0了。但是在训练的前部分，cost并没有显著地减少，也就是权重参数w和偏置参数b的变化不明显。而我们期望，当错误很大时，它能够很快的把错误降下来，但是图中的曲线一开始都是很缓慢的变化。

那么，究竟是什么原因使得模型的cost在一开始的时候下降很慢呢？我们都知道在用梯度下降更新参数的时候，我们是计算了下面这两个偏导数：

其中：，a为模型的输出，y为实际的标签值，所以上面说cost的变化不明显，也就是这两个偏导数的值很小，我们将a用上面计算output的公式代替，即，我们就可以得到：

我们可以看下sigmoid的曲线：



我们可以看到，当模型的输出a()接近于1或者接近于0的时候，曲线就很平滑，所以也就很小了（斜率很小）。因此，上面的两个偏导数的结果就很小了。这就是为什么一开始曲线下降很慢的缘故。【注：即使的输出不是接近于1或者0，由于取值范围在(0,1/4)之间，所以会缩小的值，也就导致参数更新不大，cost降不下来。】

**我们希望在误差越大时，误差降低的越快，为了实现这一目的，引入了交叉熵损失函数：**

交叉熵损失函数的公式如下：

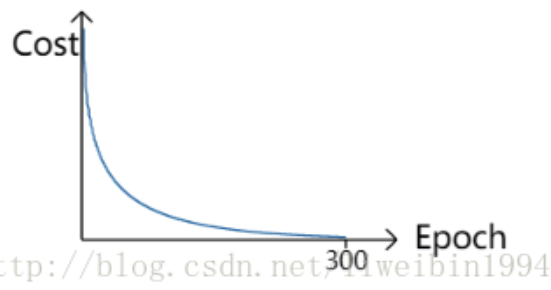
其中，n是训练样本的总数，从这个公式我们并不能很清晰的看出解决了学习速度慢的问题。我们对一个权重参数求导：

因为,我们将他代入上士可得：

同时我们可以把平方差的参数梯度放在这对比一下，下面是平凡差损失函数求出的参数梯度：

对比可以发现，交叉熵损失函数求出的权重参数的梯度直接由控制，没有了（取值范围为0~1/4），这样当模型的输出与标签y之间的偏差越大，也就是的值越大，那么偏导数也就会越大，学习也就会越快。这正是我们想要的结果。

我们仍然用一开始的那个例子，但是将损失函数由平方差函数换成交叉熵损失函数，可以得到如下曲线：



可以看到，这一次，在一开始的时候，曲线的速度变快了，这就是为什么在大多数的机器学习模型中，经常采用交叉熵损失函数的原因了。