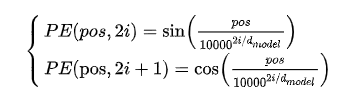
transformer答疑

1. 如何理解transformer的位置向量？



pos代表一个句子中的一个词。

如：我 想 谈恋爱

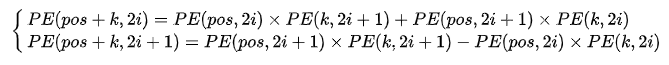
PE(0,2i) 代表位置为0的词我这个词的词向量的第2i个位置

PE(0,2i+1) 代表位置为0的词我这个词的词向量的第2i+1个位置

位置向量的目的是：希望模型学习到词与词之间的相对位置。

为啥这个公式可以表示词与词之间的位置信息呢？



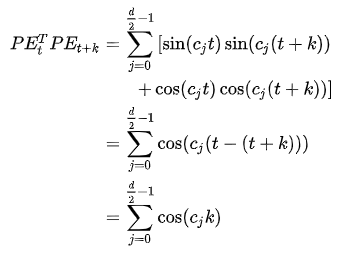


这个公式的含义是：

句子中的第pos+k个位置，某个词之后的第k个位置的词，2i是这个词对应的词向量的第2i个值，可以表示为第pos个位置的词向量的线性表示。

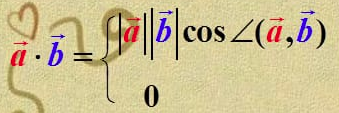
但是这个线性表示引入的东西比较多，包括第k个位置词向量的2i+1个值和2i个值，还有第pos个位置词的2i和2i+1的位置。

这样做一个线性表示就能表示词与词之间的相对位置吗？



怎样计算两个向量之间的距离：

因为每个位置向量的模都是一样的大小。跟词向量维度有关，如果维度为512，那么模的大小为256^0.5



衡量距离可以用向量和向量之间的角度来衡量。因此点积就可以衡量向量之间的距离。

可以看出第k个位置词和第t+k个位置之间的距离是k的函数。

因此那种表示是可以衡量相对位置的。

但是，思考另外一个问题：



这个公式意味着词与词之间的距离一致时便无法分清先后顺序。

例如：我 想 谈恋爱。针对想这个词，我 和 谈恋爱这两个词是无法定位其先后顺序的。

同时，位置向量需要通过长途跋涉才会到达decoder。

在第一个encoder中就出现了



这个意味着中间加了个矩阵作为乘法，虽然说最终的结果是一个标量，但是w是一个可以学习得到的参数，是具有可变性的，可变的东西意味着不再是一个仅仅是关于k的一个自变量的函数，而是更加复杂，更加没有规律性。

这可能就是transformer中位置信息做的不够细致的一个原因吧。后面会有很多人进行改进。

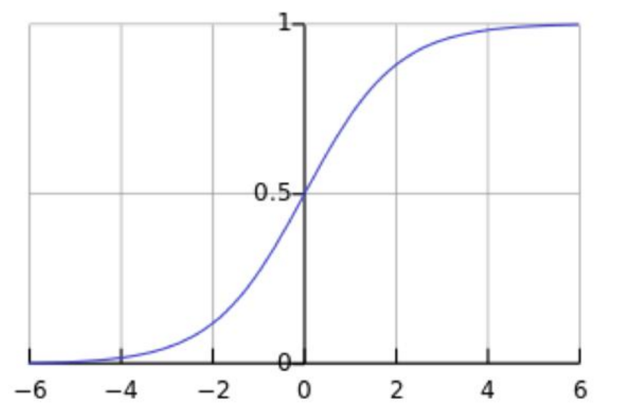
**Batch normalization**

1. 理解layer norm之前需要先理解BN

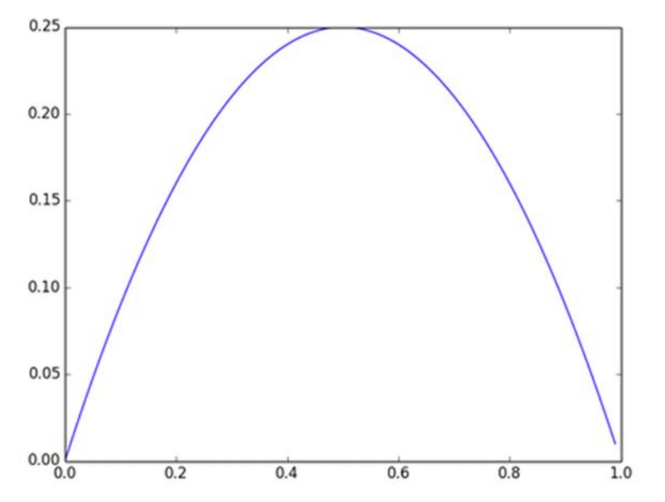
iid是统计学习中非常重要的假设，如果通过一批数据训练好了一个模型，上线了之后n天之后发现效果越来越差了，可能是因为数据的分布发生了变化。bn就是用到深度学习中的保持iid的一个东西。

1. 假如说深度学习中激活函数都用sigmoid

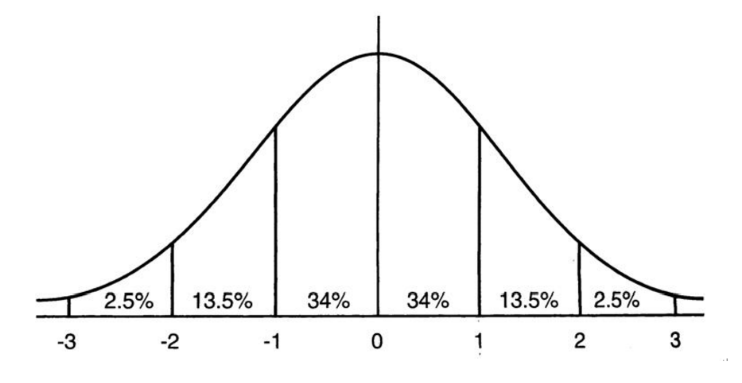
如下图可以看到当x属于-2到2之间的时候几乎非线性激活函数激活效果如同线性函数，意味着这一块区域的导数最大，激活函数看的是激活函数的导数，但是越接近两边导数越接近0. 在求深度学习的参数的时候，需要对w进行迭代，迭代的时候我们希望导数越大越好，越大收敛越快。



sigmoid函数的导数：



标准正态分布。



正态分布是一个特别普遍的分布，根据中心极限定理定理可知：

任意k个变量的均值组成的符合变量的分布都服从正态分布。

所以可以假设训练样本服从正态分布，还有深度学习中每一层隐函数往后面激活的时候，激活前的那些每个维度的变量也服从正态分布。

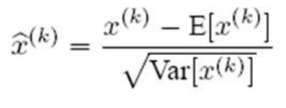
将正态分布和sigmoid函数联合起来看：

当x属于标准正态分布时，x在-2和2之间的比例达到27%+68%=95%。

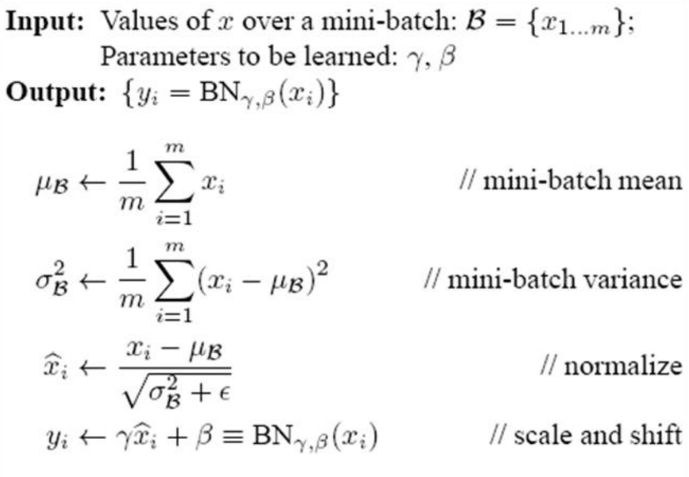
因此如果我们令输入深度学习中，进行激活函数之前的值都尽量保证在-2到2之间，那么损失函数的收敛速度是非常快的。

如何做呢？

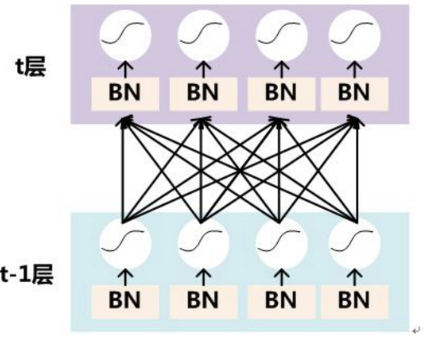
深度学习训练中最常用的就是minibatch sgd，在minibatch中强制将每个维度的数据都转化为标准正态分布。



然鹅，即使全部转化为标准正态分布，还是有问题，并不能保证每个数据都能落在-2到2之间。这个时候可以再做一个过分的事情，对minibatch中的样本再进行加乘，因为这样可以将x拉到线性变化区，有点玄幻的感觉，但是这就是深度学习的精髓啊。其中r和b都是待学习参数。咋学习的？暂时没研究。可见加了bn之后模型收敛速度可是非常的快。



bn操作的位置是激活之前



模型训练好了，之后需要在线预测

在线预测需要几个每个隐含层激活前的那些每个维度的均值方差信息，训练的时候保存下来，预测的时候利用他们求全局的均值方差即可。同时scale参数和shift参数也是训练好的，直接拿来用即可。