# 激活函数

如果不用激活函数（此时相当于激活函数是），在这种情况下，每一层节点的输入都是上层输出的线性函数。此时，无论神经网络有多少层，输出都是输入的线性组合，与没有隐藏层效果相当，那么网络的逼近能力就相当有限。而引入非线性函数作为激活函数，这样深层神经网络几乎可以逼近任意函数。

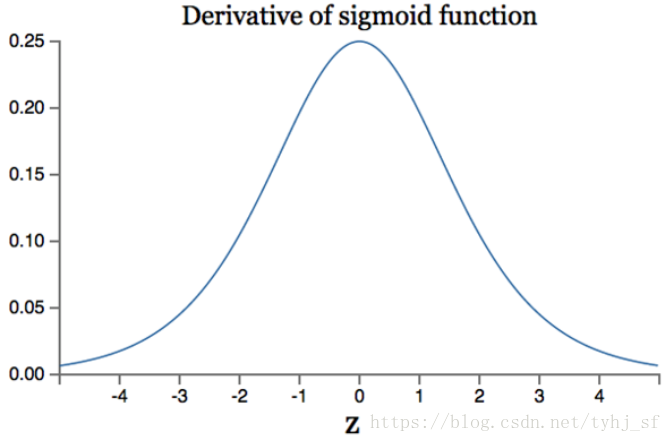
## Sigmoid函数

特点：能把输入的连续实值变换为0和1之间的输出。

缺点：

1、在深度网络中梯度反向传递时导致**梯度爆炸和梯度消失（Gradient Vanishing）**。

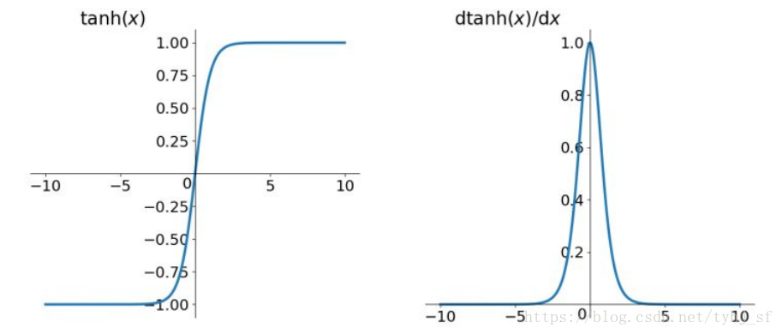
A：如果初始化神经网络的权值为之间的随机值，那么各个层相乘都是0~1之间的小数，而激活函数的导数也是0~1之间的小数，其连乘后，结果会变得很小，导致**梯度消失**。若初始化的很大，大到乘以激活函数的导数都大于1，那么连乘后，结果会变得很大，导致**梯度爆炸**。



2、Sigmoid的输出不是0均值的（zero-centered）。这会导致后一层的神经元得到生一层的非0均值的输出信号作为输入。其后果是，如，那么对求局部梯度则都为正，这样在反向传播时，要么都往正方向更新，要么都往负方向更新，导致有一种捆绑的效果，使得收敛缓慢。但如果按batch去训练，其可能得到不同的信号，还是可以得到缓解的。

3、解析式中含有幂运算，计算机求解时相对比较耗时。

## tanh函数



特点：解决了Sigmoid函数不是zero-centered的输出问题。而梯度消失、梯度爆炸和幂运算的问题仍然存在。

## Relu函数

特点：1）在正区间，解决了梯度消失问题；2）计算速度非常快；3）收敛速度远快于Sigmoid和tanh。

缺点：

1. 输出不是zero-centered的。
2. **Dead Relu Problem**，指某些神经元可能不会被激活，导致响应的参数永远不能被更新。导致其发生的主要原因为1）不幸的参数初始化；2）learning rate太高导致在训练过程中参数更新太大，不幸使网络进入这种状态。解决方法是使用**Xavier初始化方法**，及避免将learning rate设置太大，或使用adagrad等自动调节learning rate的算法。

## Leaky Relu函数

特点：解决了Dead Relu问题。

## ELU函数

特点：1）不会有Dead Relu问题；2）输出均值接近0。

# Embedding

就是用一个低维的向量表示一个物体、或是一个词、或是一个商品、或是一个电影等。Embedding向量的性质是能使距离相近的向量对应的物体有相近的含义，如Embedding(复仇者联盟)和Embedding(钢铁侠)之间的距离就会很接近。

此外，Embedding还具有数学运算的关系，如Embedding(马德里)-Embedding(西班牙)+Embedding(法国)=Embedding(巴黎)。

# 感知机

# 卷积神经网络