# 基础问题

## Relu在零点不可导，那么在反向传播中怎么处理？

caffe源码~/caffe/src/caffe/layers/relu\_layer.cpp倒数第十行代码：  
bottom\_diff[i] = top\_diff[i] \* ((bottom\_data[i] > 0)+ negative\_slope \* (bottom\_data[i] <= 0));  
这句话就是说间断点的求导按左导数来计算。也就是默认情况下（negative\_slope=0）,间断点处的导数认为是0.

## 如何处理神经网络中的过拟合问题？

1. [L1/L2正则化](https://zhuanlan.zhihu.com/p/35893078)
2. [dropout](https://zhuanlan.zhihu.com/p/35948928)
3. [data argumentation](https://zhuanlan.zhihu.com/p/35977339)
4. [early stop](https://zhuanlan.zhihu.com/p/35977339)

## Relu激活函数的优缺点？

优点包括：

1. 解决了梯度消失、爆炸的问题
2. 计算方便，计算速度快，求导方便
3. 加速网络训练

缺点包括：

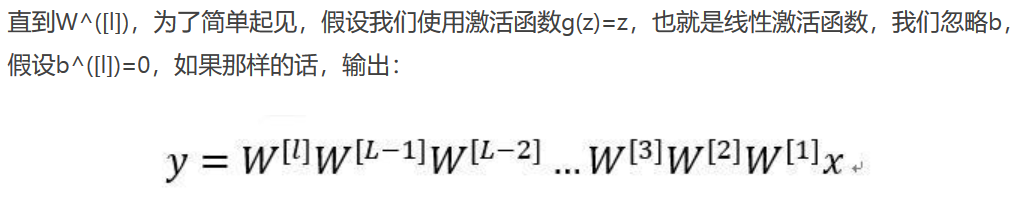
1. 由于负数部分导数恒为0，会导致一些神经元无法激活，很多神经元参数无法更新，死掉了。改进：采用weak relu激活函数，即若小于零时，也让其有点梯度。比如小于零时，激活函数为：0.01Z
2. 输出不是以0为中心

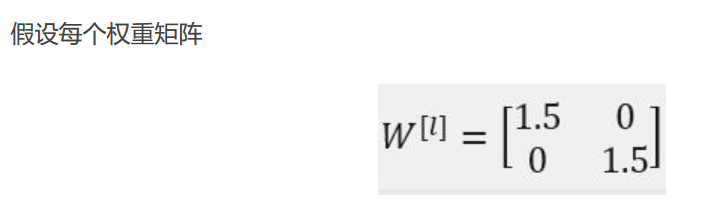
## 神经网络中的梯度爆炸和梯度消失 ?

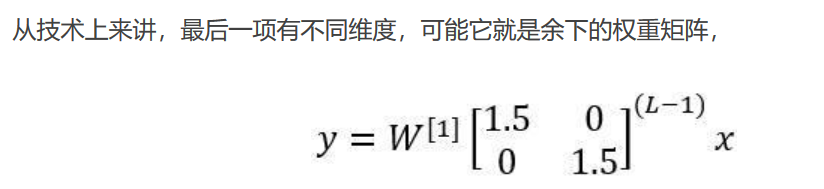
两种情况下梯度消失经常出现，一是在深层网络中，二是采用了不合适的损失函数，比如sigmoid。梯度爆炸一般出现在深层网络和权值初始化值太大的情况下，下面分别从这两个角度分析梯度消失和爆炸的原因

**训练神经网络，尤其是深度神经所面临的一个问题就是梯度消失或梯度爆炸，也就是你训练神经网络的时候，导数或坡度有时会变得非常大，或者非常小，甚至于以指数方式变小，这加大了训练的难度，不光是梯度，神经网络的神经元的值也会出现消失或是爆炸。**

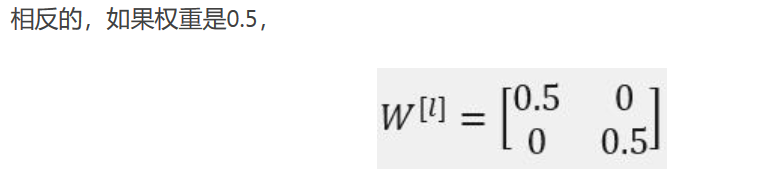
**我们先来来看神经元的问题。**

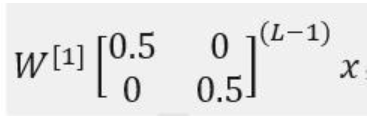






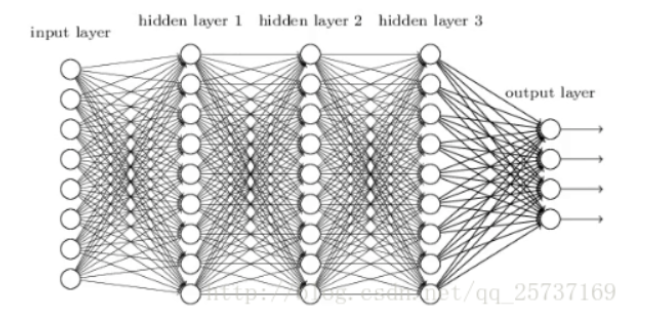
如果对于一个深度神经网络来说L值较大，那么y^的值也会非常大，实际上它呈指数级增长的，它增长的比率是1.5^L，因此对于一个深度神经网络，y的值将**爆炸式增长。**

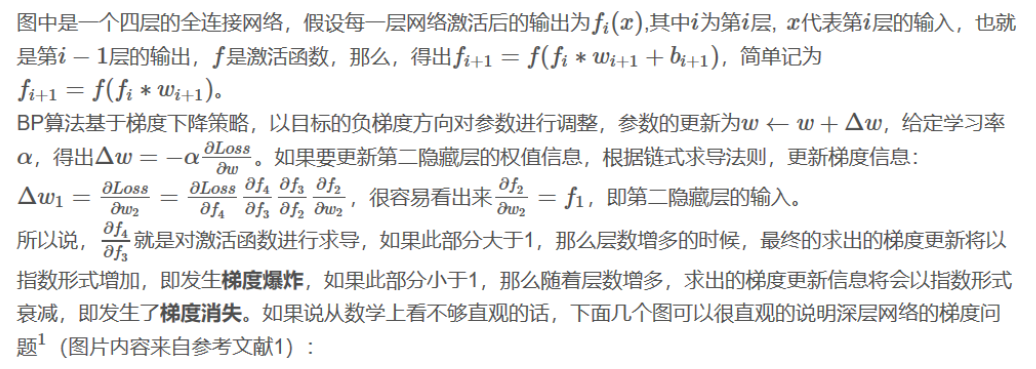




**激活函数的值将以指数级下降，它是与网络层数数量L相关的函数，在深度网络中，激活函数以指数级递减。值得注意的是神经元值得爆炸与消失发生在正向传播过程中，这个过程权值w不变的。**

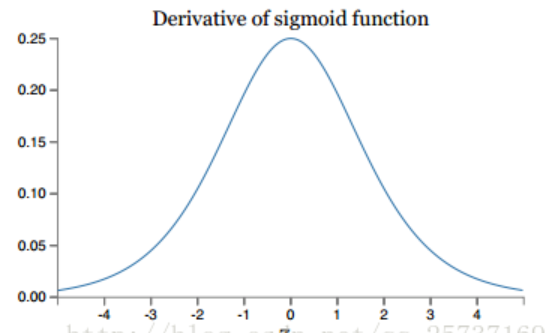
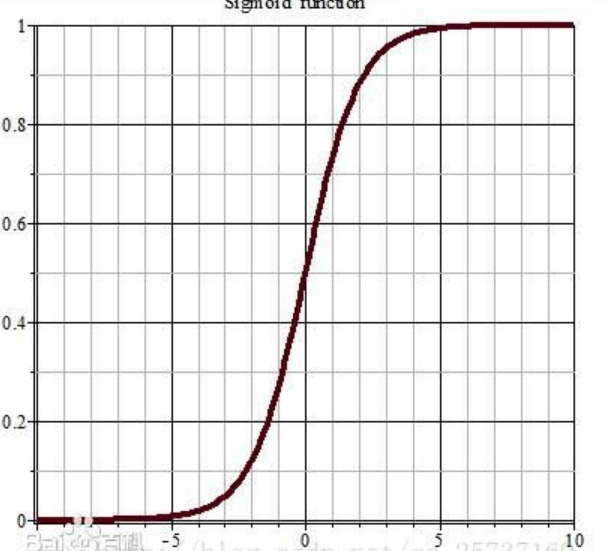
**我们来看反向传播过程**





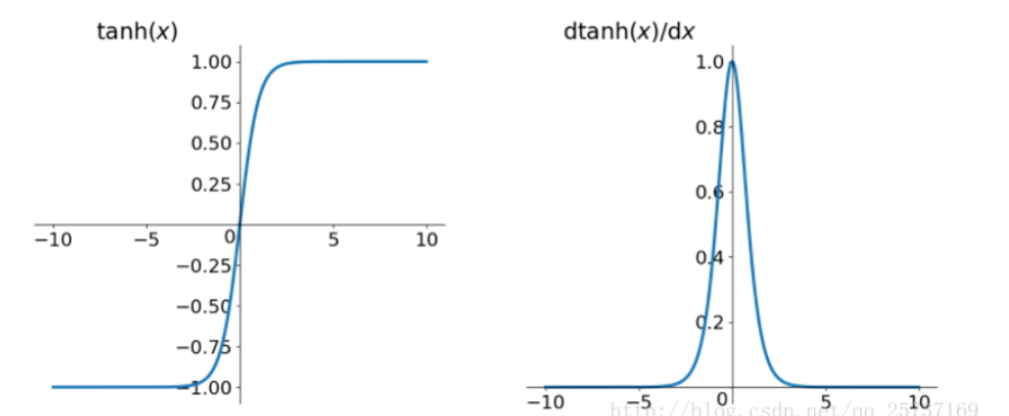
反向传播过程是链式求导法则，也是一个连乘的过程，如果我们选择simgod作为激活函数

，很容易就发生梯度消失的问题了，



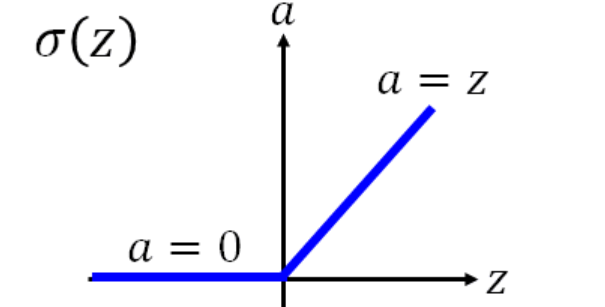
因为sigmod的梯度是不会超过0.25的，连乘的结果就是快速变小。

同理，tanh作为损失函数，它的导数图如下，可以看出，tanh比sigmoid要好一些，但是它的倒数仍然是小于1的



那样什么样的激活函数能够避免这样的情况呢。

RELU函数



让梯度变为1。

梯度爆炸的问题，就是反向传播中，求导的链式法则，将梯度不断相乘相乘再相乘，如果都是>1的话，最后得到的梯度会很大，损失函数将会发散而不会收敛

然后导致网络权重的大幅更新，并因此使网络变得不稳定。在极端情况下，权重的值变得非常大，以至于溢出，导致 NaN 值。

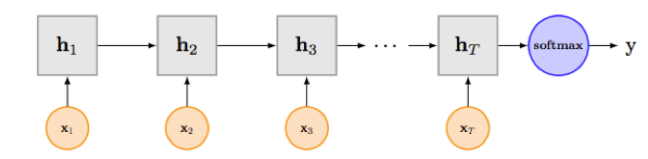
解决方法通常包括

1. 使用ReLU等激活函数，梯度只会为0或者1，每层的网络都可以得到相同的更新速度
2. 采用LSTM
3. 进行梯度裁剪(clip), 如果梯度值大于某个阈值，我们就进行梯度裁剪，限制在一个范围内
4. 使用正则化，这样会限制参数 的大小，从而防止梯度爆炸
5. 设计网络层数更少的网络进行模型训练
6. batch normalization

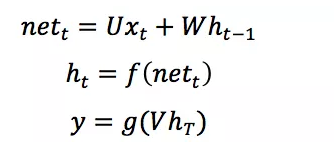
# RNN系列

## 什么是循环神经网络？如何用它产生文本表示？

RNN（Recurrent Neural Network，循环神经网络）的目的便是处理像文本这样变长并且有序的输入序列。它模拟了人阅读一篇文章的顺序，从前到后阅读文章中的每一个单词，并且将前面阅读到的有用信息编码到状态变量中去，从而拥有了一定的记忆能力，可以更好地理解之后的文本。下图展示了一个典型RNN的网络结构



一个长度为T的序列用户循环神经网络建模，展开之后可以看做是一个T层的前馈神经网络，其中第t层的隐含状态ht编码了序列中前t个输入的信息，可以通过当前输入xt和上一层神经网络的状态ht-1得到，最后一层的状态ht编码了整个序列的信息，因此可以作为整篇文章的压缩表示，例如在Ht后面直接接一个softmax层输出预测概率y。就可以实现文本分类。Ht和y的计算公式如下

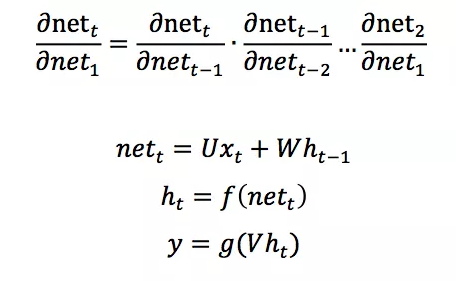


其中f和g为激活函数，U为输入层到隐含层的权重矩阵，W为隐含层从上一时刻到下一时刻状态转移的权重矩阵，在文本分类任务中（如上图），f可以选取Tanh或者ReLU函数，g可以采用Softmax函数。

一般RNN又分为，N vs 1， 1 vs N ，N vs M 三种类型，多个输入一个输出，比如说文字预测，文本感情分类，文本分类等等，一个输入多个输出，比如图片描述，文字生成等等。多对多的模式就是典型的Encode-Decode架构，典型的应用比如语言机器翻译等等。

## **RNN为什么会出现梯度的消失或爆炸？有什么样的改进方案？**

RNN模型的求解可以采用BPTT（Back Propagation Through Time）算法实现 ，实际上是反向传播（Back Propagation）算法的简单变种；如果将RNN按时间展开成T层的前馈神经网络来理解，就和普通的反向传播算法没有什么区别了。RNN的设计初衷之一就是能够捕获长距离输入之间的依赖。从结构上来看，RNN也绝对能够做到这一点。然而实践发现，使用BPTT算法学习的RNN并不能成功捕捉到长距离的依赖关系，这一现象可以用梯度消失来解释。传统的RNN梯度可以表示成连乘的形式：



由于预测的误差是沿着神经网络的每一层反向传播的，因此当Jacobian矩阵的最大特征值大于1时，随着离输出越来越远，每层的梯度大小会呈指数增长，导致梯度爆炸（gradient explosion）；反之，若Jacobian矩阵的最大特征值小于1，梯度的大小会呈指数缩小，即产生了梯度消失（gradient vanishing）。对于普通的前馈网络来说，梯度消失导致无法通过加深网络层次来改善神经网络的预测效果，因为无论如何加深网络，只有靠近输出的若干层才真正起到的学习的作用。对于RNN来说，这导致模型很难学习到输入序列中的长距离依赖关系。靠前的神经元将得不到学习，参数基本无法更新。进而失去对最终结果的作用。而对于RNN来说，LSTM(Long short-term memory)[4]及其变种GRU(Gated recurrent unit)[5]等模型通过加入门控机制(Gate)，很大程度上改善了梯度消失所带来的损失