**一、激活函数**

作用：

（1）控制数据输出格式，解决梯度爆炸和梯度消失的问题

（2）加入非线性因素，增强模型的非线性表达能力



**二、常用的激活函数**

**1、sigmoid函数**

**特点：**

它能够**把输入的连续实值变换为0和1之间的输出**，特别的，如果是**非常大的负数，那么输出就是0**；如果是**非常大的正数，输出就是1**

**缺点:**

**（1）缺点1：**在深度神经网络中**梯度反向传递时**导致梯度爆炸和梯度消失，其中**梯度爆炸发生的概率非常小，而梯度消失发生的概率比较大**

**（2）缺点2：**Sigmoid 的 output 不是0均值（即zero-centered）。这是不可取的，因为这会导致**后一层的神经元将得到上一层输出的非0均值的信号作为输入**。 产生的一个结果就是：

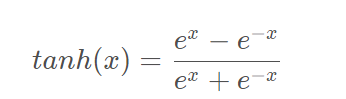
截图.png

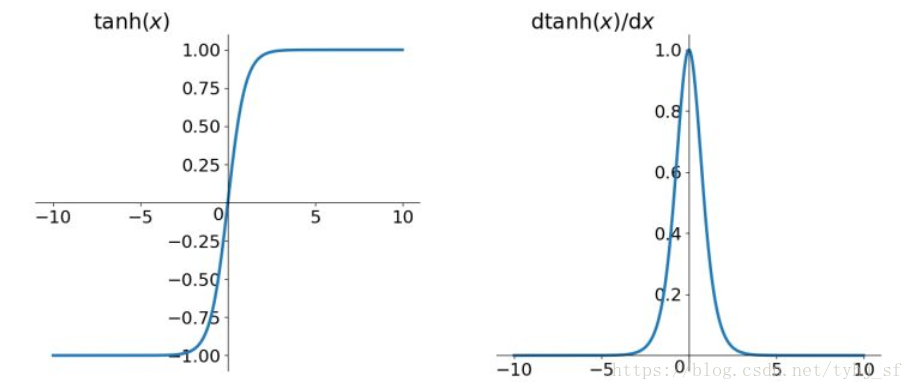
那么**对w求局部梯度则都为正**，这样**在反向传播的过程中w要么都往正方向更新，要么都往负方向更新**，导致**有一种捆绑的效果，使得收敛缓慢**。 当然了，如果按batch去训练，那么batch可能得到不同的信号，所以这个问题还是可以缓解一下的。

因此，非0均值这个问题虽然会产生一些不好的影响，不过跟上面提到的梯度消失问题相比还是要好很多的。

**（3）缺点3：**其解析式中含有幂运算，计算机求解时相对来讲比较耗时。对于规模比较大的深度网络，这会较大地增加训练时间。

**2、tanh函数**





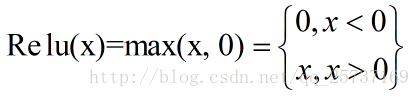
特点:

（1）tahn解决了Sigmoid函数的**不是zero-centered输出**问题

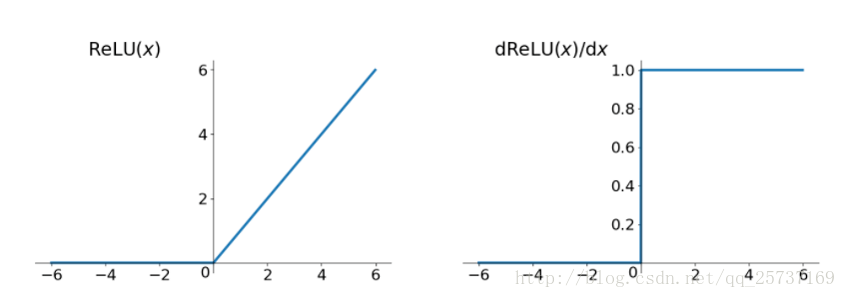
（2）然而，梯度消失（gradient vanishing）的问题和幂运算的问题仍然存在

**3、relu、leakrelu、elu**

**Relu:**思想也很简单，如果激活函数的导数为1，那么就不存在梯度消失爆炸的问题了，每层的网络都可以得到相同的更新速度，relu就这样应运而生。先看一下relu的数学表达式：



其函数图像：



从上图中，我们可以很容易看出，relu函数的导数在正数部分是恒等于1的，因此在深层网络中使用relu激活函数就不会导致梯度消失和爆炸的问题。

**（1）relu**

主要贡献在于：

*-- 解决了梯度消失、爆炸的问题*

*-- 计算方便，计算速度快*

*-- 加速了网络的训练*

同时也存在一些**缺点**：

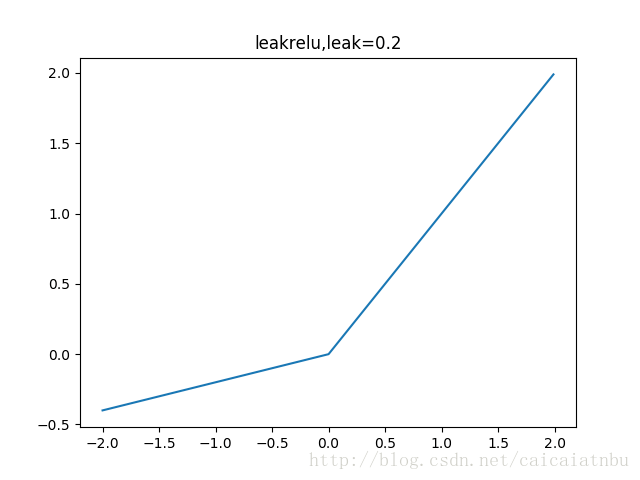
*-- 由于负数部分恒为0，会导致一些神经元无法激活（可通过设置小学习率部分解决）*

*-- 输出不是以0为中心的*

尽管relu也有缺点，但是仍然是目前使用最多的激活函数

**（2）leakrelu**

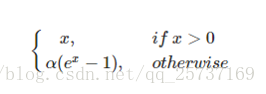
为了解决relu的0区间带来的影响，其数学表达为：其中k是leak系数，一般选择0.01或者0.02，或者通过学习而来



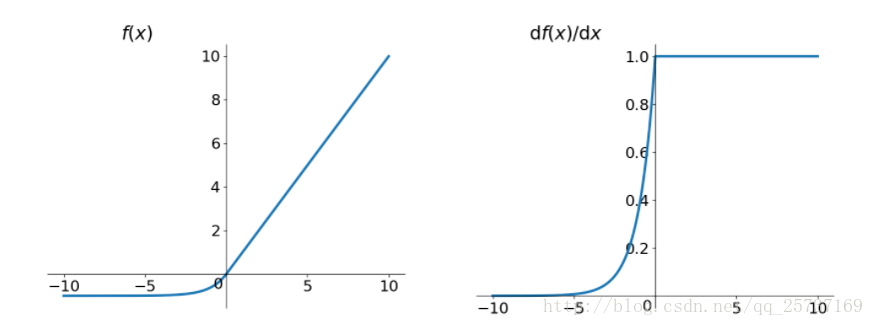
leakrelu解决了0区间带来的影响，而且包含了relu的所有优点

**（3）elu**

elu激活函数也是为了解决relu的0区间带来的影响，其数学表达为：



其函数及其导数数学形式为：



但是elu相对于leakrelu来说，计算要更耗时间一些

**三、更多的激活函数**

