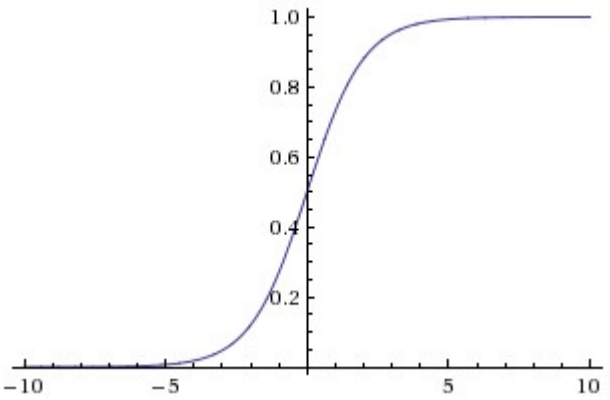
常用的非线性激活函数有sigmoid\tanh\relu等，前两者比较常见于全连接层，后者relu常见于卷积层。

1.sigmoid

sigmoid的函数表达式如下：

$$g(z) = \frac{1}{1+e^{-z}}$$

其中z是一个线性组合，比如z可以等于：b+w1*x1+w2*x2.通过代入很大的正数或很小的负数到g(z)函数中可知，其结果趋近于0或1.



也就是说，sigmoid函数的功能是相当于把一个实数压缩至0到1之间。有何用处呢？用处是这样一来便可以把激活函数看作一种“分类的概率”，比如激活函数的输出为0.9的话便可以解释为90%的概率为正样本。

综上，sigmod函数，是逻辑斯蒂回归的压缩函数，它的性质是可以把分隔平面压缩到[0,1]区间一个数（向量），在线性分割平面值为0时

候正好对应sigmod值为0.5，大于0对应sigmod值大于0.5、小于0对应sigmod值小于0.5；0.5可以作为分类的阀值；exp的形式最值求解时候比较方便，用相乘形式作为logistic损失函数，使得损失函数是凸函数；

缺点：

- 容易饱和，使得在反向传播中容易出现梯度消失，导致权重无法更新，无法学习输入数据。另外需要尤其注意参数的初始值来避免饱和的情况，如果初始值很大的话，大部分神经元可能都会处在饱和状态。
- 输出不是0均值的。这会导致后层的神经元的输入是非0均值的信号，对梯度产生影响：假设后层神经元的输入都为正，那么对w求局部梯度则都为正，这样在反向传播的过程中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。

2.tanh

$$tanhx = \frac{sinhx}{coshx} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Tanh和Sigmoid是有异曲同工之妙的，它的图形如上图右所示，不同的是它把实值得输入压缩到-1~1的范围，因此它基本是0均值的，也就解决了上述Sigmoid缺点中的第二个，所以实际中tanh会比sigmoid更常用。但是它还是存在梯度饱和的问题。Tanh是sigmoid的变形。

3.relu

- 优点1：Krizhevsky et al. 发现使用 ReLU 得到的SGD的收敛速度会比 sigmoid/tanh 快很多(如上图右)。有人说这是因为它是linear，而且梯度不会饱和

- 优点2：相比于 sigmoid/tanh需要计算指数等，计算复杂度高，ReLU 只需要一个阈值就可以得到激活值。
- 缺点1： ReLU在训练的时候很”脆弱”，一不小心有可能导致神经元”坏死”。举个例子：由于ReLU在x<0时梯度为0，这样就导致负的梯度在这个ReLU被置零，而且这个神经元有可能再也不会被任何数据激活。如果这个情况发生了，那么这个神经元之后的梯度就永远是0了，也就是ReLU神经元坏死了，不再对任何数据有所响应。实际操作中，如果你的learning rate 很大，那么很有可能你网络中的40%的神经元都坏死了。 当然，如果你设置了一个合适的较小的learning rate，这个问题发生的情况其实也不会太频繁。

4.Leak relu

Leaky ReLUs 就是用来解决ReLU坏死的问题的。和ReLU不同，当x<0时，它的值不再是0，而是一个较小斜率(如0.01等)的函数。也就是说f(x)=1(x<0)(ax)+1(x>=0)(x),其中a是一个很小的常数。这样，既修正了数据分布，又保留了一些负轴的值，使得负轴信息不会全部丢失。关于Leaky ReLU 的效果，众说纷纭，没有清晰的定论。有些人做了实验发现 Leaky ReLU 表现的很好;有些实验则证明并不是这样。