神经网络通常不是一个凸优化的问题，他处处充满了局部最优解。

神经科学家进行大脑能量消耗的研究中发现，神经元编码的工作方式具有稀疏性，推测大脑同时被激活的神经元只有

1%-4%，神经元只会对输入信号有少部分的选择性相应，大量不相关信号被屏蔽，这样可以更高效的提取重要特征。

传统的sigmoid函数有近半神经元被激活，不符合神经科学的研究。softplus虽有单侧抑制，却没有稀疏激活性。

因而ReLU成了最符合实际神经元的模型。

输入层一般用ReLU，输出层一般使用sigmoid,因为接近概率输出

dropout是指在深度学习网络的训练过程中，对于神经网络单元，按照一定的概率将其暂时从网络中丢弃。注意是暂时，对于随机梯度下降来说，由于是随机丢弃，故而每一个mini-batch都在训练不同的网络。

SGD本身也不是一个比较稳定的算法，结果可能在最优解附近波动，不同的学习速率可能导致神经网络落入截然不同的局部最优置空。

Adagrad,Adam,Adadelta等自适应方法可以减轻调参负担，是一种自适应的学习率，如前期学习率大，后期学习率小。

信号在超过某个阈值时，神经元才会进入兴奋和激活状态，平时则处于抑制状态。ReLu = max(0,x)可以很好的

传递梯度，经过多层的反向传播，梯度依旧不会大幅缩小，非常适合训练深层神经网络，从正面解决了梯度弥散问题。

sigmoid函数在反向传播中梯度值会逐渐减小，经过多层传递后呈指数级急剧减小。

ReLU:单侧抑制，相对宽阔的兴奋边界，稀疏激活性。

softplus:单侧抑制，无稀疏激活性 log(1+ex)

梯度弥散，梯度越来越小，甚至消失

梯度爆炸，梯度越来越大

如果不用激励函数（其实相当于激励函数是f(x) = x），在这种情况下你每一层输出都是上层输入的线性函数，很容易验证，无论你神经

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，这种情况就是最原始的感知机（Perceptron）了。 正因为上面的原

因，我们决定引入非线性函数作为激励函数，这样深层神经网络就有意义了（不再是输入的线性组合，可以逼近任意函数）。最早的想法

是sigmoid函数或者tanh函数，输出有界，很容易充当下一层输入（以及一些人的生物解释balabala）。

从数学上来看，Sigmoid函数对中央区的信号增益较大，对两侧区的信号增益小，在信号的特征空间映射上，有很好的效果。

　　从神经科学上来看，中央区酷似神经元的兴奋态，两侧区酷似神经元的抑制态，因而在神经网络学习方面，可以将重点特征推向中央区，

TanHyperbolic(tanh)函数又称作双曲正切函数，数学表达式为y=(ex − e−x)/(ex + e− x),其函数曲线与Sigmoid函数相似

第一，采用sigmoid等函数，算激活函数时（指数运算），计算量大，反向传播求误差梯度时，求导涉及除法，计算量相对大，而采用

Relu激活函数，整个过程的计算量节省很多。

第二，对于深层网络，sigmoid函数反向传播时，很容易就会出现梯度消失的情况（在sigmoid接近饱和区时，变换太缓慢，导数趋于0，

这种情况会造成信息丢失），从而无法完成深层网络的训练。

第三，Relu会使一部分神经元的输出为0，这样就造成了网络的稀疏性，并且减少了参数的相互依存关系，缓解了过拟合问题的发生（以

及一些人的生物解释balabala）。