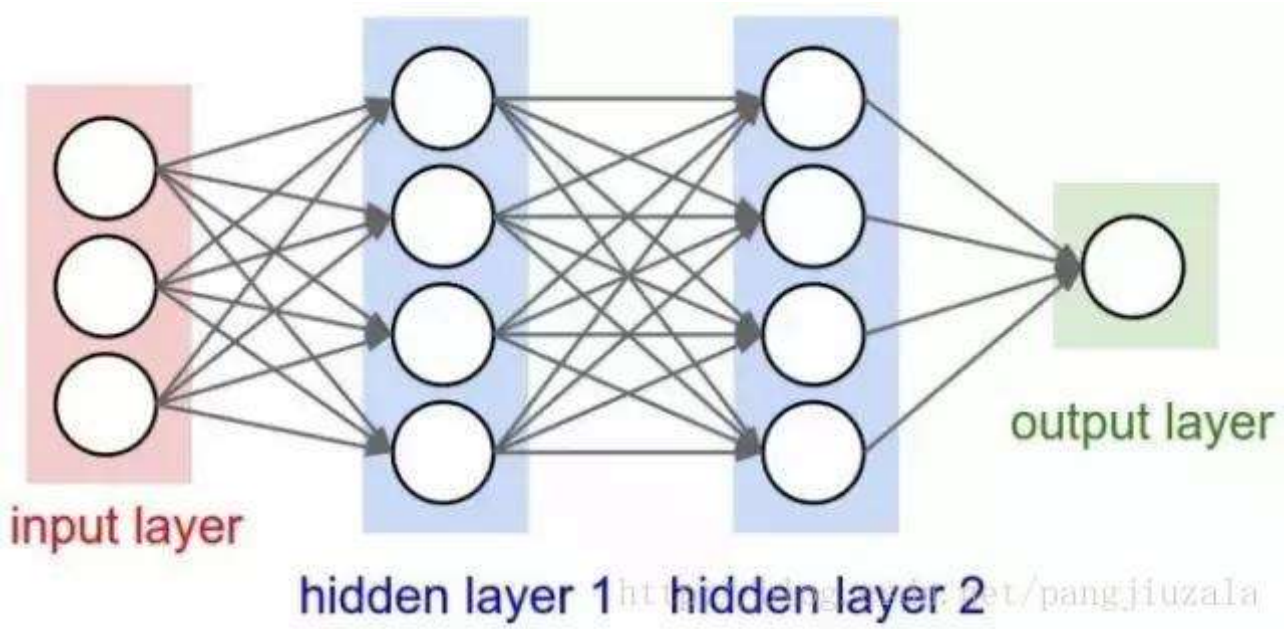
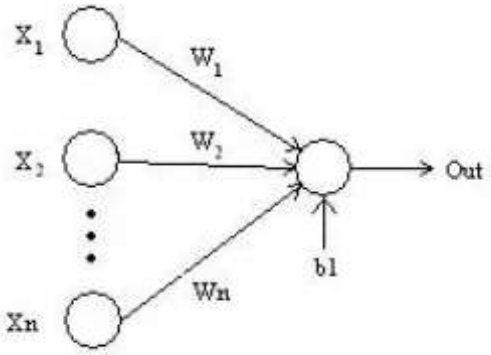
了解神经网络

神经网络的结构

神经网络由输入/输出/隐藏层组成



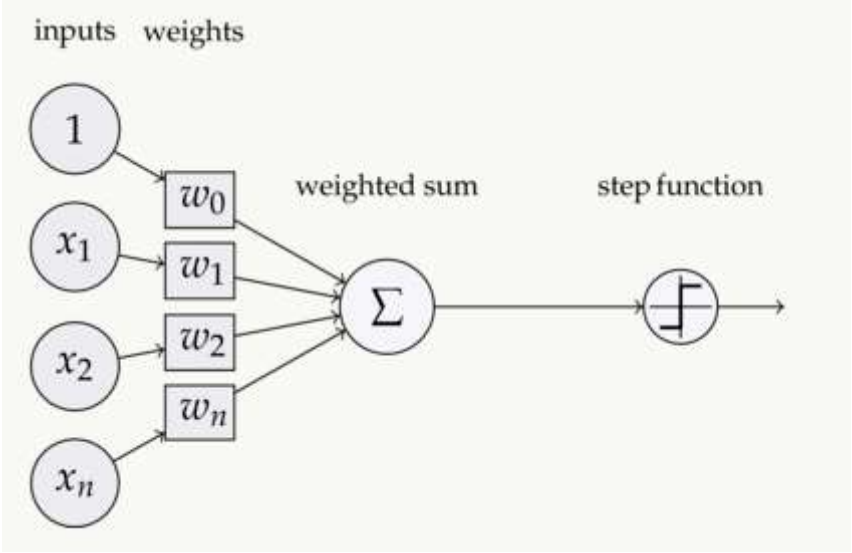
这些**Layer**都是由一个个**神经元**组成，每个神经元都接受多个输入，并产生一个输出，就好像人的神经元突触一样。神经元在接收输入时，会各自乘以一定的**权重**(有时候还会加上一个**bias**)，并根据自己的**激活函数**产生输出。权重大则说明神经网络认为它的信息比较重要，权重小则认为神经网络认为它的信息不那么重要。



激活函数

什么是激活函数

如下图，在神经元中，输入的 inputs 通过加权，求和后，还被作用了一个函数，这个函数就是激活函数 Activation Function。

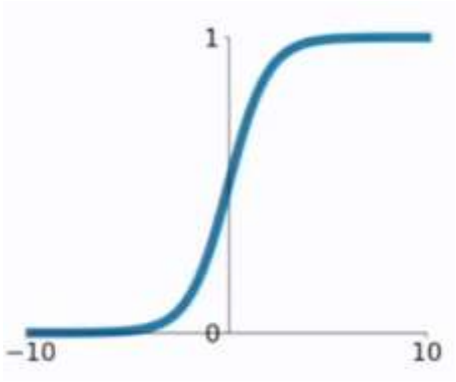


为什么要用

如果不用激励函数，每一层输出都是上层输入的线性函数，无论神经网络有多少层，输出都是输入的线性组合。 如果使用的话，激活函数给神经元引入了非线性因素，使得神经网络可以任意逼近任何非线性函数，这样神经网络就可以应用到众多的非线性模型中。 常用的激活函数有三种，Sigmoid， ReLU 和 Softmax。

Sigmoid函数计算如下：

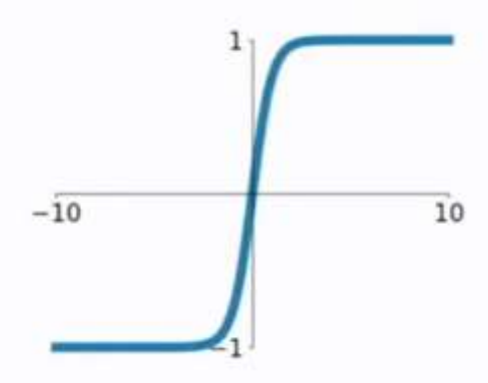
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



- softmax函数最好在分类器的输出层使用。
- 它可以将一个实数映射到(0,1)的区间，可以用来做二分类。（它不像SVM直接给出一个分类的结果，Logistic Regression给出的是这个样本属于正类或者负类的可能性是多少，当然在多分类的系统中给出的是属于不同类别的可能性，进而通过可能性来分类。）
- 在特征相差比较复杂或是相差不是特别大时效果比较好。
- 优点：
  i. Sigmoid函数的输出映射在(0,1)之间，单调连续，输出范围有限，优化稳定，可以用作输出层。
  ii. 求导容易。
- 缺点：
  i. 由于其软饱和性，容易产生梯度消失，导致训练出现问题。
  ii. 其输出并不是以0为中心的。

tanh函数计算如下：

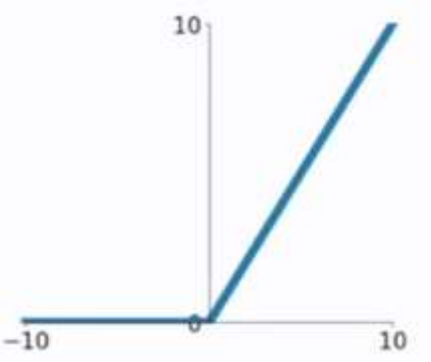
$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



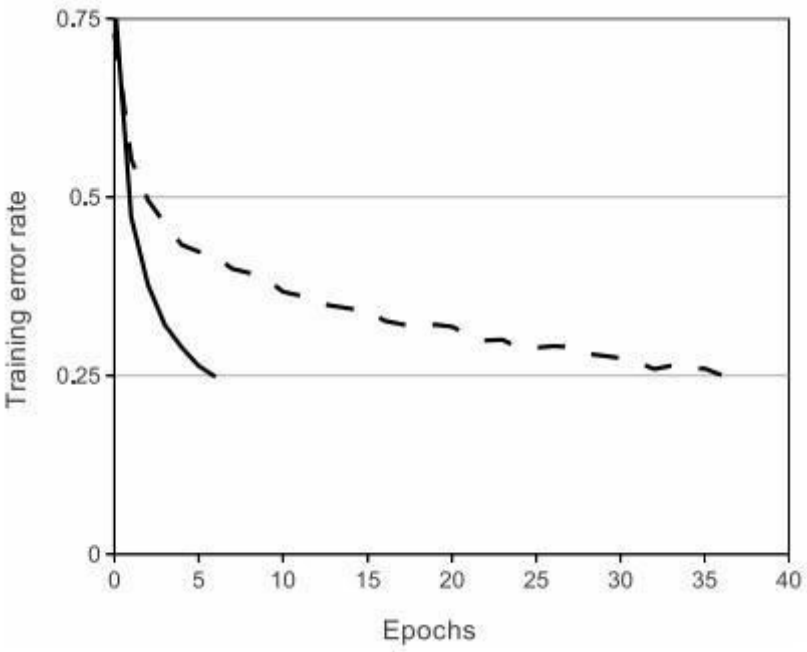
- 优点：
  i. 比Sigmoid函数收敛速度更快。
  ii. 相比Sigmoid函数，其输出以0为中心。
- 缺点：
  i. 还是没有改变Sigmoid函数的最大问题——由于饱和性产生的梯度消失。

ReLU函数计算如下：

$$f(x) = max(0, x)$$



- ReLU是近几年非常受欢迎的激活函数。
- 当x<0时，ReLU硬饱和，而当x>0时，则不存在饱和问题。所以，ReLU 能够在x>0时保持梯度不衰减，从而**缓解梯度消失问题**。这让我们能够直接**以监督的方式训练深度神经网络**，而无需依赖无监督的逐层预训练。
- 优点：
  i. 相比起Sigmoid和tanh，ReLU(e.g. a factor of 6 in Krizhevsky et al.)在SGD中能够快速收敛。例如在下图的实验中，在一个四层的卷积神经网络中，实线代表了ReLU，虚线代表了tanh，ReLU比起tanh更快地到达了错误率0.25处。据称，这是因为它线性、非饱和的形式。



2. Sigmoid和tanh涉及了很多很expensive的操作（比如指数），ReLU可以更加简单的实现。

3. 有效缓解了梯度消失的问题。

4. 在没有无监督预训练的时候也能有较好的表现。

Neuron	MNIST	CIFAR10	NISTP	NORB
With unsupervised pre-training				
Rectifier	**1.20%**	**49.96%**	**32.86%**	**16.46%**
Tanh	**1.16%**	**50.79%**	35.89%	17.66%
Softplus	**1.17%**	**49.52%**	**33.27%**	19.19%
Without unsupervised pre-training				
Rectifier	**1.43%**	**50.86%**	**32.64%**	**16.40%**
Tanh	1.57%	52.62%	36.46%	19.29%
Softplus	1.77%	53.20%	35.48%	17.68%

5. 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。

- 缺点：
  i. 随着训练的进行，可能会出现神经元死亡，权重无法更新的情况。如果发生这种情况，那么流经神经元的梯度从这一点开始将永远是0。也就是说，ReLU神经元在训练中不可逆地死亡了。

LReLU、PReLU与RReLU

未研究，详情见参考

什么是饱和，硬饱和

$$\lim_{n \to +\infty} h'(x) = 0$$

a.饱和 当一个激活函数h(x)满足

时我们称之为右饱和。

当一个激活函数h(x)满足 $\lim_{n \to -\infty} h'(x) = 0$

时我们称之为左饱和。当一个激活函数，既满足左饱和又满足又饱和时，我们称之为饱和。

b.硬饱和与软饱和 对任意的x，如果存在常数c，当x>c时恒有 h'(x)=0则称其为右硬饱和，当x<c时恒 有h'(x)=0则称其为左硬饱和。若既满足左硬饱和，又满足右硬饱和，则称这种激活函数为硬饱和。但如果只有在极限状态下偏导数等于0的函数，称之为软饱和。

如何选择正确的激活函数?

激活函数好或坏，不能凭感觉定论。然而，根据问题的性质，我们可以为神经网络更快更方便地收敛作出更好的选择。

- 用于分类器时，Sigmoid函数及其组合通常效果更好。
- 由于梯度消失问题，有时要避免使用sigmoid和tanh函数。
- ReLU函数是一个通用的激活函数，目前在大多数情况下使用。
- 如果神经网络中出现死神经元，那么PReLU函数就是最好的选择。
- ReLU函数只能在隐藏层中使用。(为什么)

做的时候，可以从ReLU函数开始，如果ReLU函数没有提供最优结果，再尝试其他激活函数。

代价函数

https://www.cnblogs.com/Belter/p/6653773.html 此处有比较好的阐述，暂未仔细研究

各种优化方法

http://www.sohu.com/a/149921578_610300 此处有比较好思路，但时间不足未能仔细研究，后根据需要补上。 一般都是采用梯度下降。梯度下降的幅度即使learning_rate，幅度过大，则在接近最优解时难以收敛，幅度过小，则收敛速度过慢。真正的生产环节中，可以是先采取大的learning_rate，再慢慢减小。

GAN

https://www.youtube.com/watch?v=yYUN_k36u5Q

生成-对抗网络的训练过程就是训练评估模型与生成模型。一边纠正评估模型，一边让评估模型指导生成模型的生成。 详细地说，就是先让**初始生成模型**生成一组，再与样本数据合在一起让**初始判别模型**判断，并得到判断的准确性的评估。让这个评估作为反馈，进行BP，让生成模型和判别模型都反馈学习。

参考

- 理解这25个概念，你的「深度学习」才算入门
- 激活函数比较
- 各种激活函数比较
- The Activation Function in Deep Learning浅谈深度学习中的激活函数
- 如何选择激活函数