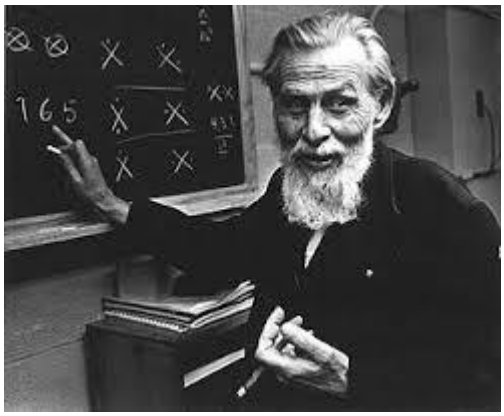
深度学习(deep learning)发展史

- **1943 年**

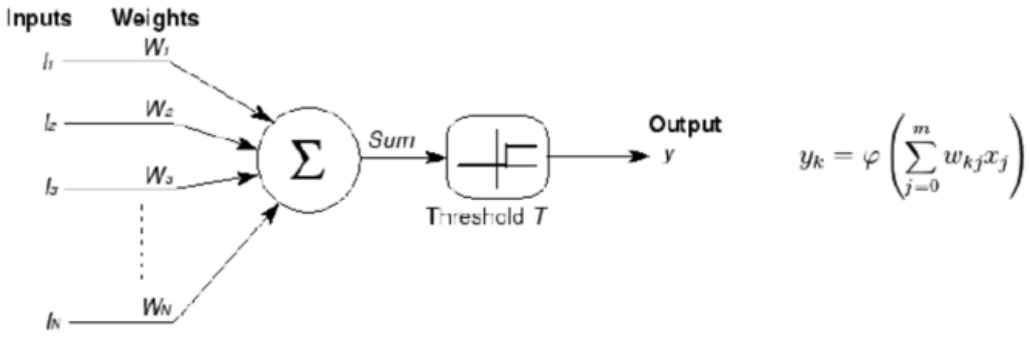
由神经科学家**麦卡洛克(W.S.McCilloch)** 和**数学家皮兹（W.Pitts)** 在《数学生物物理学公告》上发表论文《神经活动中内在思想的逻辑演算》（A Logical Calculus of the Ideas Immanent in Nervous Activity）。建立了神经网络和数学模型，称为 MCP 模型。所谓 **MCP** 模型，其实是按照生物神经元的结构和工作原理构造出来的一个抽象和简化了的模型，也就诞生了所谓的“模拟大脑”，人工神经网络的大门由此开启。



麦卡洛克(W.S.McCilloch)

皮兹（W.Pitts)

MCP 当时是希望能够用计算机来模拟人的神经元反应的过程，该模型将神经元简化为了三个过程：输入信号线性加权，求和，非线性激活（阈值法）。如下图所示



- **1958 年**

计算机科学家**罗森布拉特（ Rosenblatt)** 提出了两层神经元组成的神经网络，称之为**“感知器”(Perceptrons)**。第一次将 MCP 用于**机器学习（machine learning）分类(classification)**。“感知器”算

法算法使用 MCP 模型对输入的多维数据进行二分类，且能够使用梯度下降法从训练样本中自动学习更新权值。1962 年,该方法被证明为能够收敛，理论与实践效果引起第一次神经网络的浪潮。

- **1969 年**

  纵观科学发展史，无疑都是充满曲折的，深度学习也毫不例外。
  1969 年，美国数学家及人工智能先驱 **Marvin Minsky** 在其著作中证明了感知器本质上是一种**线性模型（linear model）**，只能处理线性分类问题，就连最简单的 XOR（亦或）问题都无法正确分类。这等于直接宣判了感知器的死刑，神经网络的研究也陷入了将近 20 年的停滞。

- **1986 年**

  由神经网络之父 **Geoffrey Hinton** 在 1986 年发明了适用于多层感知器（MLP）的 **BP（Backpropagation）**算法，并采用 **Sigmoid** 进行非线性映射，有效解决了非线性分类和学习的问题。该方法引起了神经网络的第二次热潮。

  注：**Sigmoid** 函数是一个在生物学中常见的 S 型的函数，也称为 S 型生长曲线。在信息科学中，由于其单增以及反函数单增等性质，Sigmoid 函数常被用作神经网络的阈值函数，将变量映射到 0,1 之间。

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

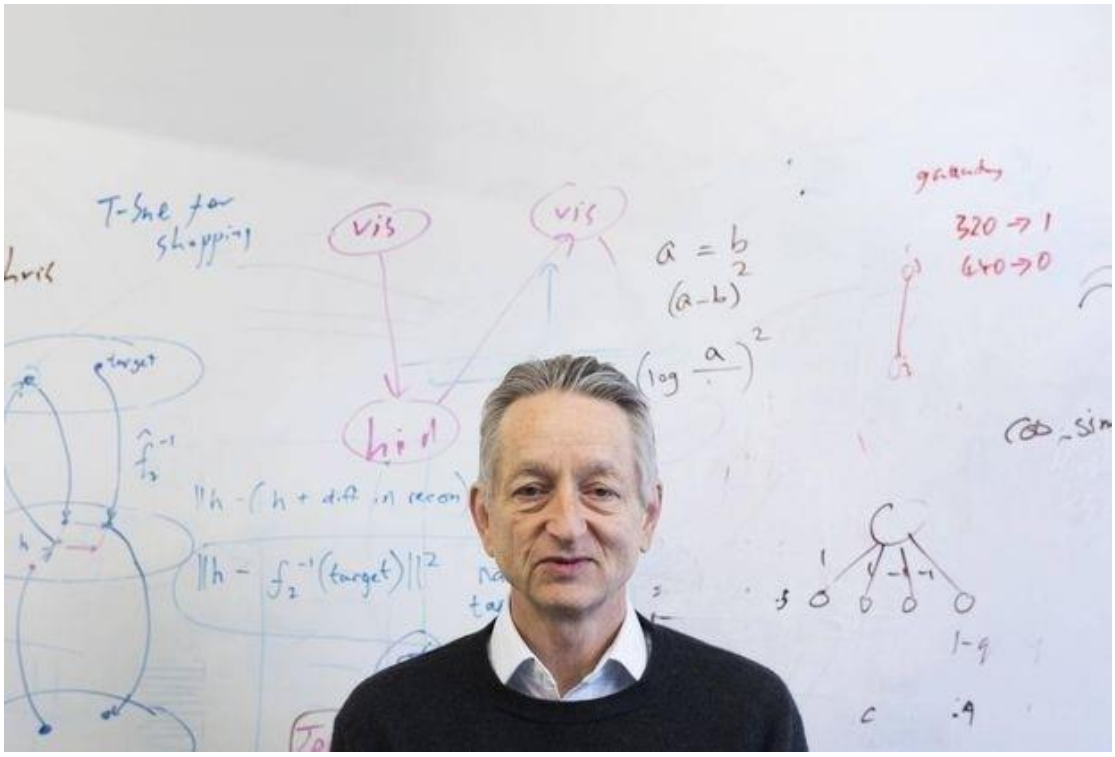
- **90 年代时期**

  1991 年 BP 算法被指出存在梯度消失问题，也就是说在误差梯度后项传递的过程中，后层梯度以乘性方式叠加到前层，由于 Sigmoid 函数的饱和特性，后层梯度本来就小，误差梯度传到前层时几乎为 0，因此无法对前层进行有效的学习，该问题直接阻碍了深度学习的进一步发展。

  此外 90 年代中期，支持向量机算法诞生（SVM 算法）等各种浅层机器学习模型被提出，SVM 也是一种有监督的学习模型，应用于模式识别，分类以及回归分析等。支持向量机以统计学为基础，和神经网络有明显的差异，支持向量机等算法的提出再次阻碍了深度学习的发展。

- **发展期 2006 年 - 2012 年**

2006 年，加拿大多伦多大学教授、机器学习领域泰斗、神经网络之父——Geoffrey Hinton 和他的学生 Ruslan Salakhutdinov 在顶尖学术刊物《科学》上发表了一篇文章，该文章提出了深层网络训练中梯度消失问题的解决方案：**无监督预训练对权值进行初始化+有监督训练微调。**斯坦福大学、纽约大学、加拿大蒙特利尔大学等成为研究深度学习的重镇，至此开启了深度学习在学术界和工业界的浪潮。



Geoffrey Hinton

2011 年，ReLU 激活函数被提出，该激活函数能够有效的抑制梯度消失问题。2011 年以来，微软首次将 DL 应用在语音识别上，取得了重大突破。微软研究院和 Google 的语音识别研究人员先后采用 DNN

技术降低语音识别错误率 20%~30%，是语音识别领域十多年来最大的突破性进展。2012 年，DNN 技术在图像识别领域取得惊人的效果，在 ImageNet 评测上将错误率从 26%降低到 15%。在这一年，DNN 还被应用于制药公司的 DrugeActivity 预测问题，并获得世界最好成绩。

- **爆发期 2012 - 2017**

2012 年，Hinton 课题组为了证明深度学习的潜力，首次参加 ImageNet 图像识别比赛，其通过构建的 CNN 网络 AlexNet 一举夺得冠军，且碾压第二名（SVM 方法）的分类性能。也正是由于该比赛，CNN 吸引到了众多研究者的注意。

AlexNet

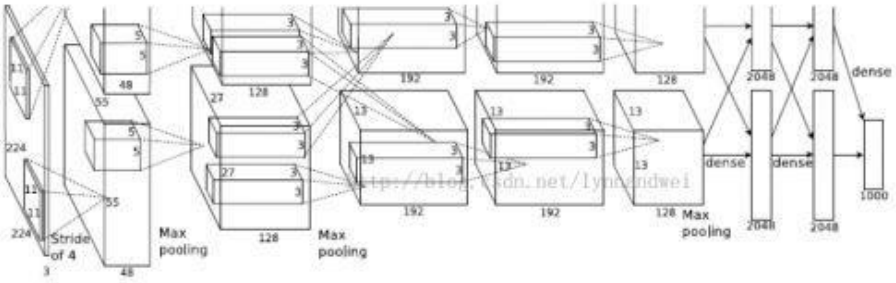


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

From: (2012_NIPS) Imagenet classification with deep convolutional neural networks.

AlexNet 的创新点在于:

(1)首次采用 ReLU 激活函数，极大增大收敛速度且从根本上解决了梯度消失问题。

(2)由于 ReLU 方法可以很好抑制梯度消失问题，AlexNet 抛弃了“预训练+微调”的方法，完全采用有监督训练。也正因为如此，DL 的主流学习方法也因此变为了纯粹的有监督学习。

(3)扩展了 LeNet5 结构，添加 Dropout 层减小过拟合，LRN 层增强泛化能力/减小过拟合。

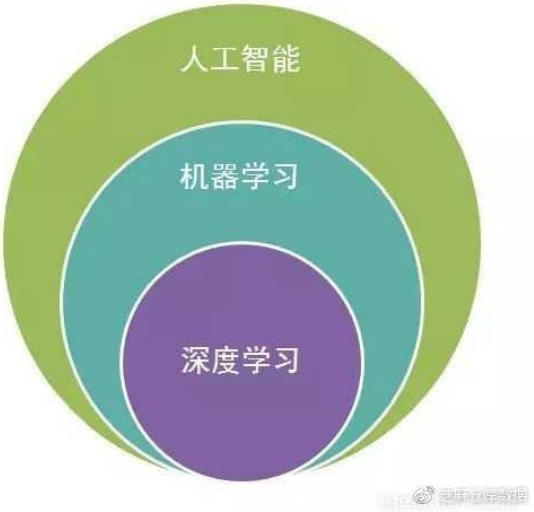
(4)第一次使用 GPU 加速模型计算。

2013、2014、2015、2016 年，通过 ImageNet 图像识别比赛，DL 的网络结构，训练方法，GPU 硬件的不断进步，促使其在其他领域也在不断的征服战场。

2016 年 3 月，由谷歌（Google）旗下 DeepMind 公司开发的 AlphaGo(基于深度学习)与围棋世界冠军、职业九段棋手李世石进行围棋人机大战，以 4 比 1 的总比分获胜；2016 年末 2017 年初，该程序在中国棋类网站上以“大师”（Master）为注册帐号与中日韩数十位围棋高手进行快棋对决，连续 60 局无一败绩；2017 年 5 月，在中国乌镇围棋峰会上，它与排名世界第一的世界围棋冠军柯洁对战，以 3 比 0 的总比分获胜。围棋界公认阿尔法围棋的棋力已经超过人类职业围棋顶尖水平。

人工智能、机器学习、深度学习有什么区别和联系

机器学习是一种实现人工智能的方法，深度学习是一种实现机器学习的技术。如下图（来源http://baijiahao.baidu.com/s?id=1588563162916669654&wfr=spider&for=pc）：



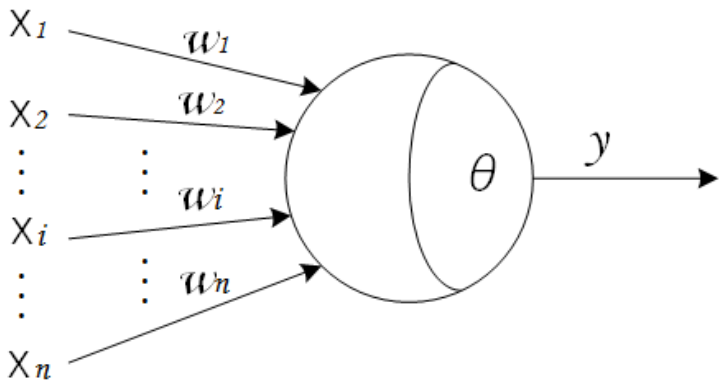
下面一张图能更加细分其关系：

注意：在上幅图中，我们可以看下机器学习下的深度学习和监督学习以及非监督学习，那它们之间是什么关系呢，其实就是分类方法不同而已，他们之间可以互相包含。打个比方：一个人按性别可以分为男人和女人，而按年龄来分可以分为老人和小孩子。所以在深度学习中我们可以用到监督学习和非监督学习，而监督学习中可以用到很基础的不含神经元的算法（KNN 算法）也可以用到添加了多层神经元的深度学习算法。

神经元、单层感知机、多层感知机

神经网络(neural networks)是由具有适应性的简单单元组成的广泛并行互连的网络，它的组织能够模拟生物神经系统对真实世界物体所作出的交互反应。

神经网络中最基本的成分是神经元(neuron)模型，即上述所说的“简单单元”。在生物神经网络中，每个神经元与其他神经元相连，当它“兴奋”时，就会向相连的神经元发送化学物质，从而改变这些神经元内的电位；如果某神经元的电位超过了一个“阈值”，那么它就会被激活，即“兴奋”起来，向其他神经元发送化学物质。



1943 年，McCulloch 和 Pitts 将上述情形抽象为上图所示的简单模型，这就是一直沿用至今的 M-P 神经元模型。神经元接收来自 nn 个其他神经元传递过来的输入信号 xx，这些输入信号通过带权重 ww 的连接(connection)进行传递，神经元接收到的总输入值 ∑ni=1wixi∑i=1nwixi 将与神经元的阈值 θθ 进行比较，然后通过“激活函数”(activation function) ff 处理产生神经元的输出 y=f(∑ni=1wixi−θ)y=f(∑i=1nwixi−θ)。

1 单层感知机的模型

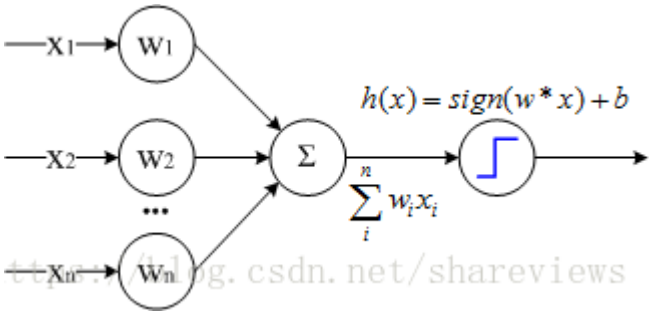
单层感知机目标是将被感知数据集划分为两类的分离超平面，并计算出该超平面。单层感知机是二分类的线性分类模型，输入是被感知数据集的特征向量，输出时数据集的类别{+1,-1}。感知器的模型可以简单表示为：

[f (x) = s i g n (w . x + b) [f(x)=sign(w.x+b)
[f(x)=sign(w.x+b)

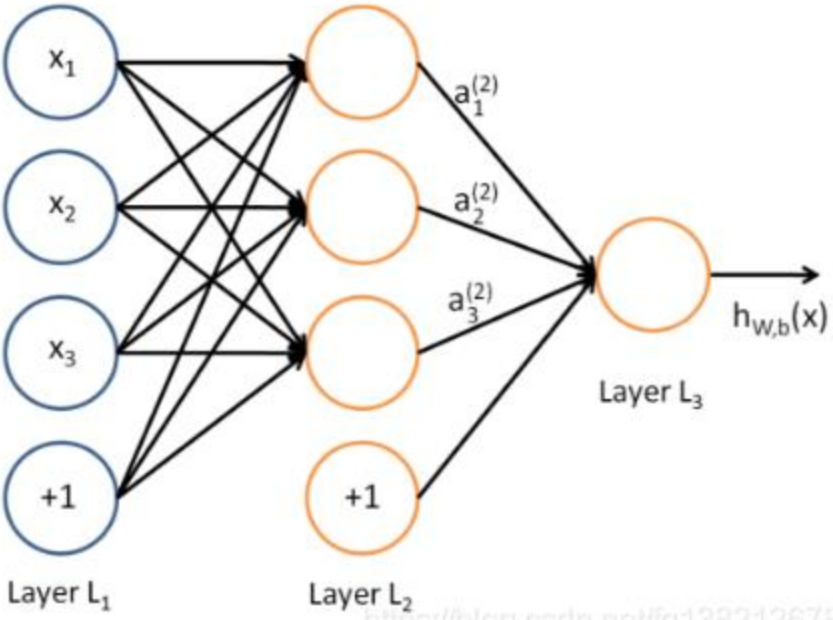
该函数称为单层感知机，其中 w 是网络的 N 维权重向量，b 是网络的 N 维偏置向量, w.x 是 w 和 x 的内积，w 和 b 的 N 维向量取值要求在实数域。

sign 函数是感知机的早期激活函数，后面又演化出一系列的激活函数。激活函数一般采用非线性激活函数，以增强网络的表达能力。常见的激活函数有：sign, sigmoid,tanh,ReLU 等。

s i g n (x) = { + 1 x > 0 + 0 x < 0 sign(x) =
{+1+0amp;xamp;xgt;0lt;0
{+1amp;xgt;0+0amp;xlt;0
sign(x)={ +1+0x>0x<0
为单层感知机与逻辑回归的差别就是感知机激活函数是 sign，逻辑回归的激活函数是 sigmoid。sign(x)将大于 0 的分为 1，小于 0 的分为-1；sigmoid 将大于 0.5 的分为 1，小于 0.5 的分为 0。因此 sign 又被称为单位阶跃函数，逻辑回归也被看作是一种概率估计。



多层感知机（MLP，Multilayer Perceptron）也叫人工神经网络（ANN，Artificial Neural Network），除了输入输出层，它中间可以有多个隐层，最简单的 MLP 只含一个隐层，即三层的结构，如下图：

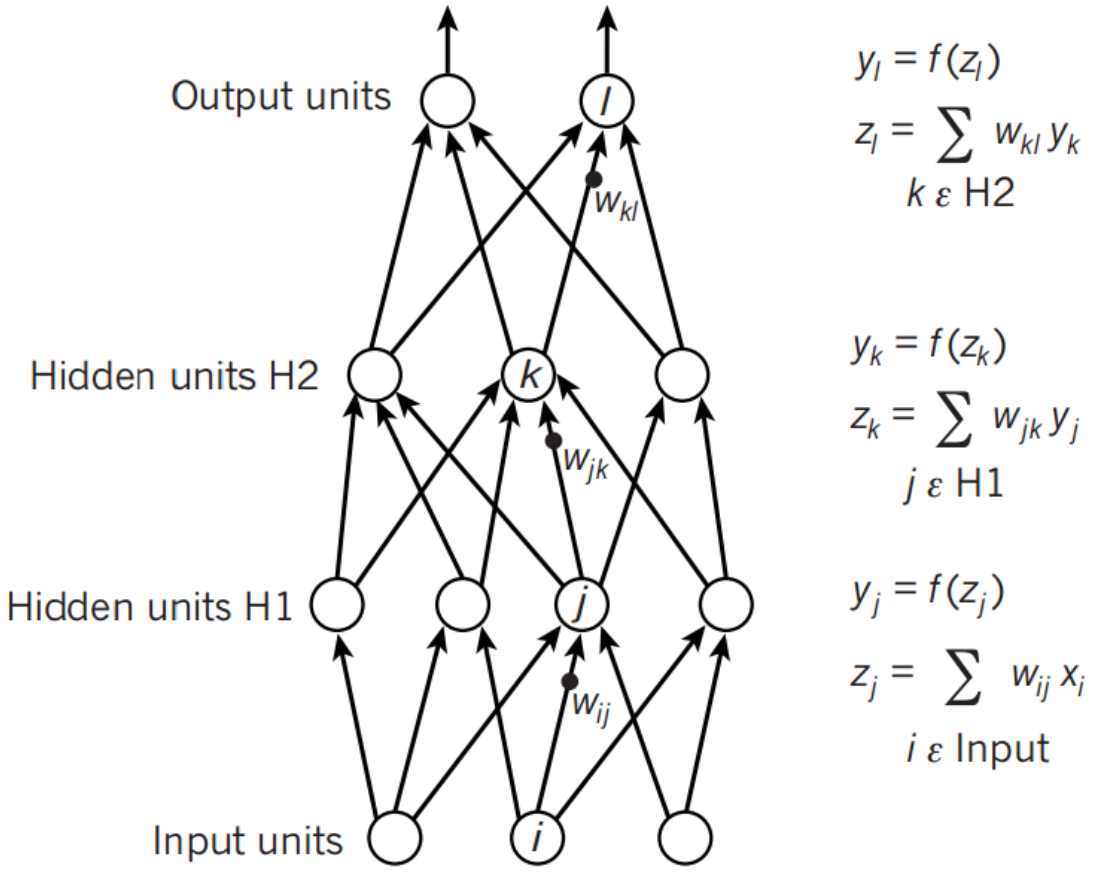


从上图可以看到，多层感知机层与层之间是全连接的。多层感知机最底层是输入层，中间是隐藏层，最后是输出层。

隐藏层的神经元怎么得来？首先它与输入层是全连接的，假设输入层用向量 X 表示，则隐藏层的输出就是 f (W1X+b1)，W1 是权重（也叫连接系数），b1 是偏置，函数 f 可以是常用的 sigmoid 函数或者 tanh 函数

什么是前向传播

1.前向传播



如图所示，这里讲得已经很清楚了，前向传播的思想比较简单。

举个例子，假设上一层结点 i,j,k,…等一些结点与本层的结点 w 有连接，那么结点 w 的值怎么算呢？就是通过上一层的 i,j,k 等结点以及对应的连接权值进行加权和运算，最终结果再加上一个偏置项（图中为了简单省略了），最后在通过一个非线性函数（即激活函数），如 ReLu，sigmoid 等函数，最后得到的结果就是本层结点 w 的输出。

最终不断的通过这种方法一层层的运算，得到输出层结果。

对于前向传播来说，不管维度多高，其过程都可以用如下公式表示:

a 2 = σ (z 2) = σ (a 1 * W 2 + b 2) a^2 = \sigma(z^2) = \sigma(a^1 * W^2 + b^2)$a2=\sigma(z2)=\sigma(a1*W2+b2)$

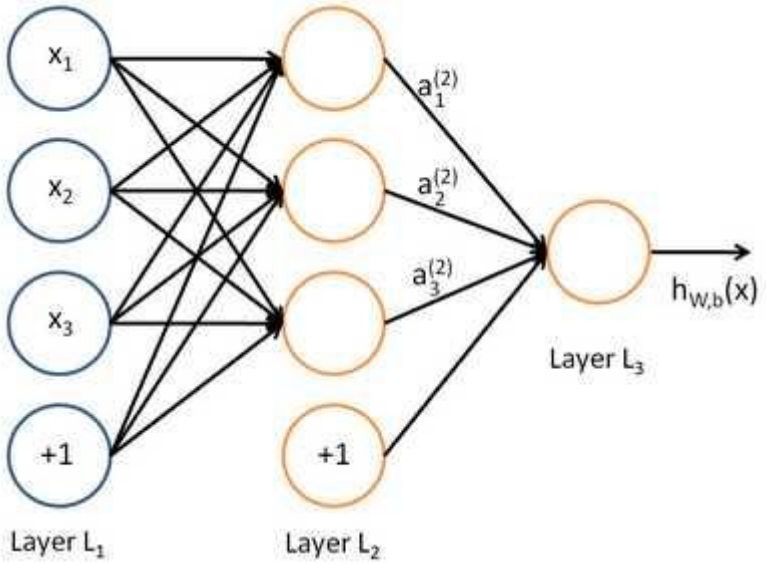
其中，上标代表层数，星号表示卷积，b 表示偏置项 bias，σ \sigma$\sigma$ 表示激活函数。

什么是反向传播

BackPropagation 算法是多层神经网络的训练中举足轻重的算法。简单的理解，它的确就是复合函数的链式法则，但其在实际运算中的意义比链式法则要大的多。要回答题主这个问题“如何直观的解释 back propagation 算法？”需要先直观理解多层神经网络的训练。

机器学习可以看做是数理统计的一个应用，在数理统计中一个常见的任务就是拟合，也就是

给定一些样本点，用合适的曲线揭示这些样本点随着自变量的变化关系.
深度学习同样也是为了这个目的，只不过此时，样本点不再限定为(x, y)点对，而可以是由向量、矩阵等等组成的广义点对(X,Y)。而此时，(X,Y)之间的关系也变得十分复杂，不太可能用一个简单函数表示。然而，人们发现可以用多层神经网络来表示这样的关系，而多层神经网络的本质就是一个多层复合的函数。借用网上找到的一幅图[1]，来直观描绘一下这种复合关系。



其对应的表达式如下:

$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$
$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$
$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$
$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$