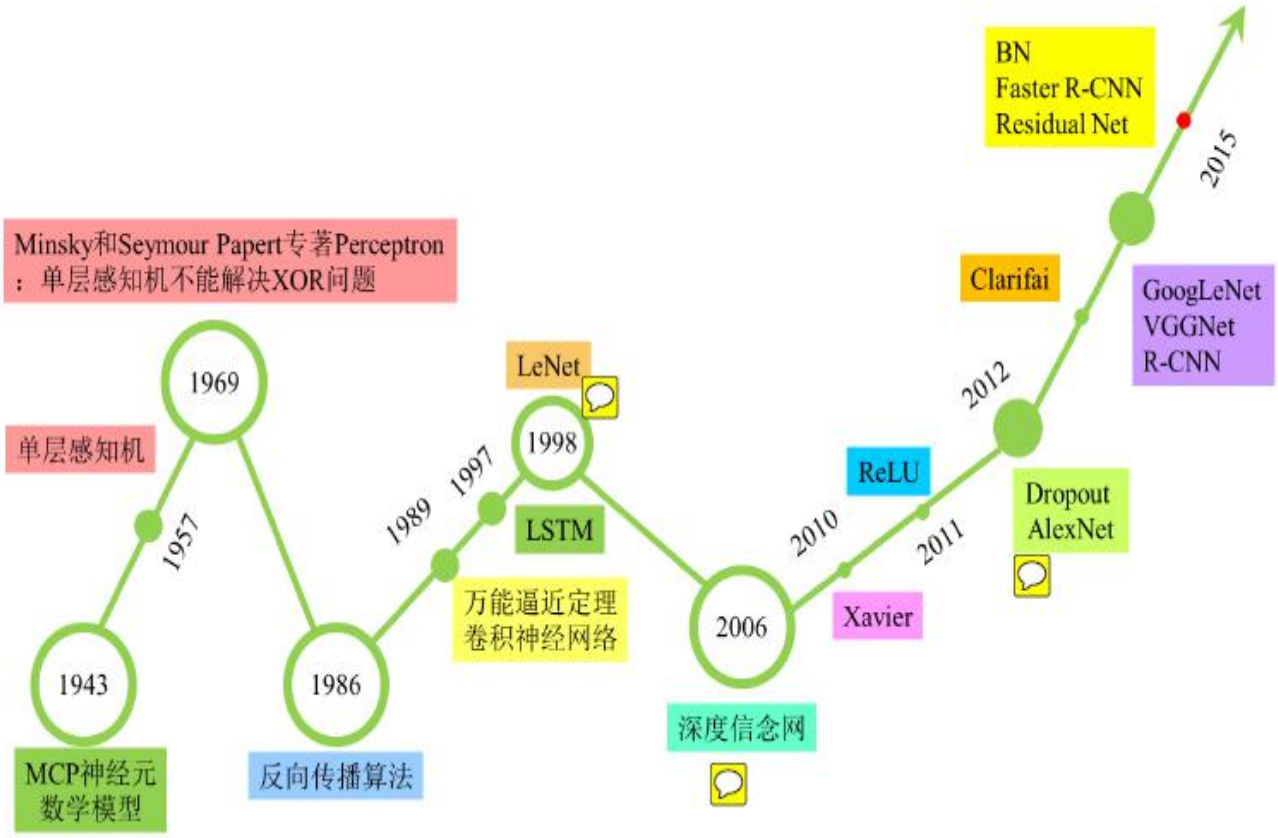
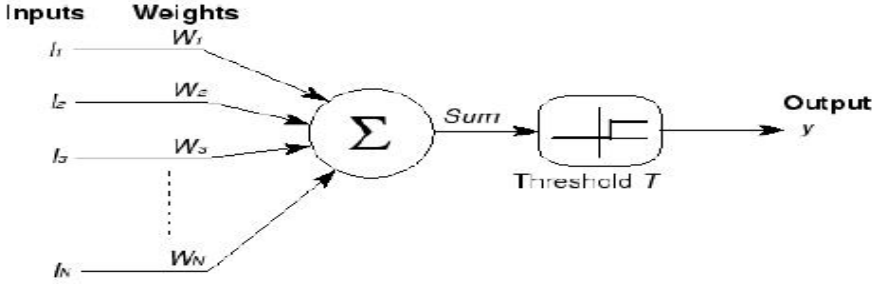
1、深度学习发展历史



由图可以明显看出 DL 在从 06 年崛起之前经历了两个低谷，这两个低谷也将神经网络的发展分为了三个不同的阶段。

第一代神经网络（1958~1969）

最早的神经网络的思想起源于 1943 年的 MCP 人工神经元模型，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



$$y_k = \varphi\left(\sum_{j=0}^{m} w_{kj}x_j\right)$$

第一次将 MCP 用于机器学习（分类）的当属 1958 年 Rosenblatt 发明的感知器（perceptron）算法。该算法使用 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。

然而学科发展的历史不总是一帆风顺的。

1969 年，美国数学家及人工智能先驱 Minsky 在其著作中证明了感知器本质上是一种线性模型，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近 20 年的停滞。

第二代神经网络（1986~1998）

第一次打破非线性诅咒的当属现代 DL 大牛 Hinton，其在 1986 年发明了适用于多层感知器（MLP）的 BP 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。

1989 年，Robert Hecht-Nielsen 证明了 MLP 的万能逼近定理，即对于任何闭区间内的一个连续函数 f，都可以用含有一个隐含层的 BP 网络来逼近该定理的发现极大的鼓舞了神经网络的研究人员。

也是在 1989 年，LeCun 发明了卷积神经网络-LeNet，并将其用于数字识别，且取得了较好的成绩，不过当时并没有引起足够的注意。

值得强调的是在 1989 年以后由于没有特别突出的方法被提出，且 NN 一直缺少相应的严格的数学理论支持，神经网络的热潮渐渐冷淡下去。冰点来自于 1991 年，BP 算法被指出存在梯度消失问题，即在误差梯度后向传递的过程中，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，该发现对此时的 NN 发展雪上加霜。

1997 年，LSTM 模型被发明，尽管该模型在序列建模上的特性非常突出，但由于正处于 NN 的下坡期，也没有引起足够的重视。

统计学习方法的春天（1986~2006）

1986 年，决策树方法被提出，很快 ID3，ID4，CART 等改进的决策树方法相继出现，到目前仍然是非常常用的一种机器学习方法。该方法也是符号学习方法的代表。

1995 年，线性 SVM 被统计学家 Vapnik 提出。该方法的特点有两个：由非常完美的数学理论推导而来（统计学与凸优化等），符合人的直观感受（最大间隔）。不过，最重要的还是该方法在线性分类的问题上取得了当时最好的成绩。

1997 年，AdaBoost 被提出，该方法是 PAC（Probably Approximately Correct）理论在机器学习实践上的代表，也催生了集成方法这一类。该方法通过一系列的弱分类器集成，达到强分类器的效果。

2000 年，KernelSVM 被提出，核化的 SVM 通过一种巧妙的方式将原空间线性不可分的问题，通过 Kernel 映射成高维空间的线性可分

问题，成功解决了非线性分类的问题，且分类效果非常好。至此也更加终结了 NN 时代。

2001 年，随机森林被提出，这是集成方法的另一代表，该方法的理论扎实，比 AdaBoost 更好的抑制过拟合问题，实际效果也非常不错。

2001 年，一种新的统一框架-图模型被提出，该方法试图统一机器学习混乱的方法，如朴素贝叶斯，SVM，隐马尔可夫模型等，为各种学习方法提供一个统一的描述框架。

第三代神经网络-DL（2006-至今）

该阶段又分为两个时期：快速发展期（2006~2012）与爆发期（2012~至今）

快速发展期（2006~2012）

2006 年，DL 元年。是年，Hinton 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。其主要思想是先通过自学习的方法学习到训练数据的结构（自动编码器），然后在该结构上进行有监督训练微调。但是由于没有特别有效的实验验证，该论文并没有引起重视。

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，微软首次将 DL 应用在语音识别上，取得了重大突破。

爆发期（2012~至今）

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。

2、人工智能、机器学习、深度学习有什么区别联系

严格意义上说，人工智能和机器学习没有直接关系，只不过机器学习的方法被大量的应用于解决人工智能的问题而已。机器学习是人工智能的一种实现方式，也是最重要的实现方式。人工智能已经变成了一个很泛泛的学科了。

严格意义上说，人工智能和机器学习没有直接关系，只不过目前机器学习的方法被大量的应用于解决人工智能的问题而已。目前机器学习是人工智能的一种实现方式，也是最重要的实现方式。

早期的机器学习实际上是属于统计学，而非计算机科学的;而二十世纪九十年代之前的经典人工智能跟机器学习也没有关系。所以今天的 AI 和 ML 有很大的重叠，但并没有严格的从属关系。

不过如果仅就计算机系内部来说，ML 是属于 AI 的。AI 今天已经变成了一个很泛泛的学科了。

深度学习是机器学习现在比较火的一个方向，其本身是神经网络算法的衍生，在图像、语音等富媒体的分类和识别上取得了非常好的效果。

术语“人工智能”，“机器学习”和“深度学习”描述了过去几十年来

建立在自身之上的过程，因为世界在计算能力，数据传输和其他技术目标方面取得了巨大进步。

对话应从人工智能开始，人工智能是计算机或技术能够模拟人类思想或大脑活动的任何功能的广义术语。从某种意义上说，人工智能是从早期开始的，它使用简单的计算机国际象棋下棋程序以及其他开始模仿人类决策和思想的程序。

从个人计算机的早期开始，人工智能就一直在发展，直到互联网时代，最后一直到云计算，虚拟化和复杂网络的时代。人工智能已经作为关键技术行业以多种方式发展壮大。

人工智能的里程碑之一是机器学习的出现和采用，这是实现人工智能目标的一种特殊方法。

机器学习使用复杂的算法和程序来帮助计算机软件在性能环境中更好地做出某些决策。机器学习不再像 1970 年代和 1980 年代的手工编码程序那样简单地一遍又一遍地编程计算机来完成一组事情，而是开始使用启发式，行为建模和其他类型的预测来允许技术，以改善其决策并随着时间的推移而发展。机器学习已应用于打击垃圾邮件，实现 IBM Watson 等人工智能个性以及以其他方式实现人工智能目标。

反过来，深度学习则建立在机器学习的基础上。专家将深度学习描述为使用算法来驱动高级抽象，例如使用人工神经网络来训练任务技术。深度学习通过尝试对实际人脑活动进行建模并将其应用于人工决策或其他认知工作，从而将机器学习提升到一个新的水平。

深度学习已通过最先进的供应链优化程序，实验室设备程序以及其他类型的创新（例如，生成对抗网络）等例子得到了证明，其中两个相对的网络（生成和歧视性网络）相互竞争，以模拟人类歧视的思维过程。这种特定类型的深度学习可以应用于图像处理和其他用途。

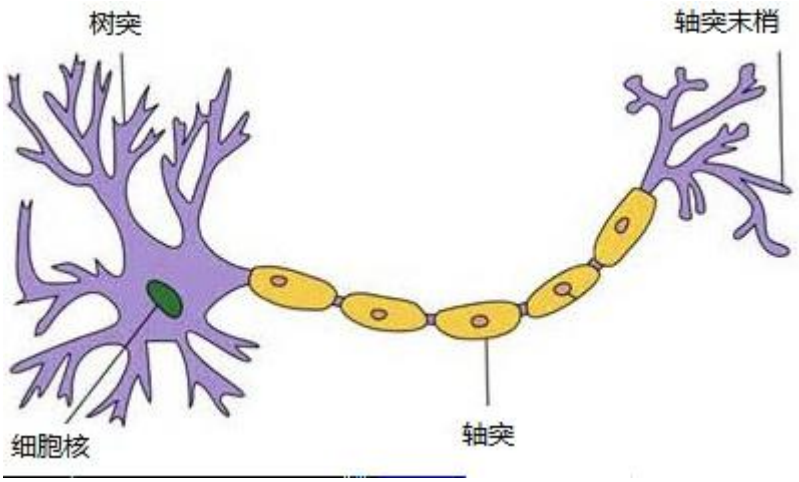
现实情况是，深度学习使人工智能更接近于专家认为是“强大的AI”的东西，人工智能或多或少完全具有复制许多人类思维功能的能力。这就引起了关于如何有效地处理这些新兴技术以及如何照料计算机以与我们相同的方式思考世界的争论。

3、神经元、单层感知机、多层感知机

神经网络的起源是人们力图寻找或者构建一种结构来模拟人脑神经元的功能，并最终达到模拟人脑进行工作甚至达到仿真人的级别。实际上现行的神经网络已经沦为一种算法，该算法与其他算法最主要的区别是能够很好的 处理高度非线性问题。现在对于神将网络的研究主要集中在两个方面，一是挖掘算法的深度，也就是不断地优化改进算法，提出新方法以满足现有的生活实际需要，如人们常说的深度学习网络、循环神经网络等；而是提升现有算法的速度，一般都是基于硬件基础的实现，也就是人们常说的类脑神经计算等。

神经元

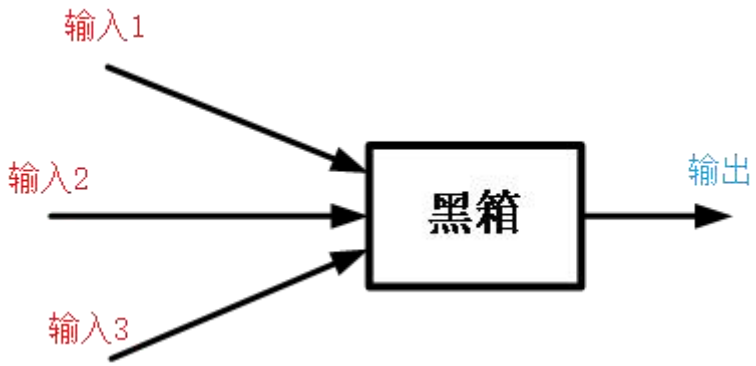
在现有的神经网络算法中，神经元的作用确实体现的较小，但是在硬件（材料、结构）上的实现时，神经元的深入研究可以更好的推进对材料功能实现的机理理解。



一个神经元通常具有多个树突，主要用来接受传入信息；而轴突只有一条，轴突尾端有许多轴突末梢可以给其他多个神经元传递信息。轴突末梢跟其他神经元的树突产生连接，从而传递信号。这个连接的位置在生物学上叫做“突触”。突触之间的交流通过神经递质实现。

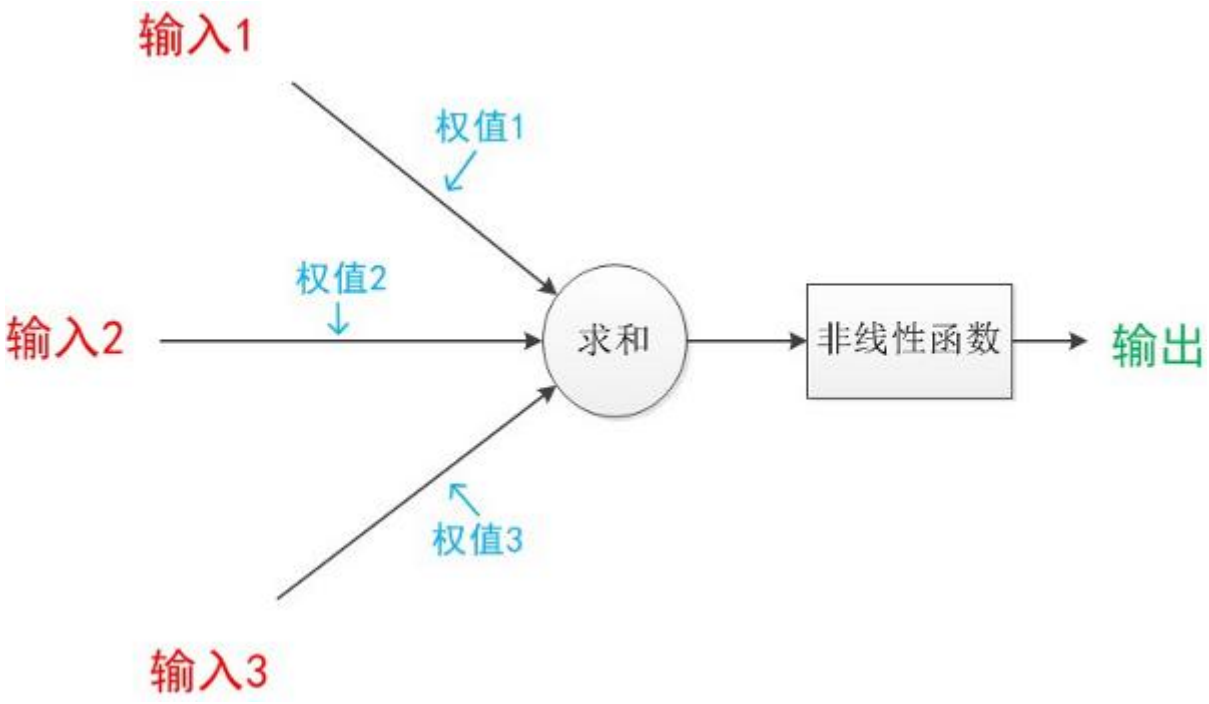
数学模型：

对上面的这个模型进行抽象处理。首先考虑到神经元结构有多个树突，一个轴突可将其抽象为下图的黑箱结构：



但是黑箱结构有诸多不便，首先是不知道黑箱中的函数结构就不能为我们所用，其次是输入输出与黑箱的关系也无法量化。因此考虑将上述结构简化，首先把树突到细胞核的阶段简化为线性加权的过程

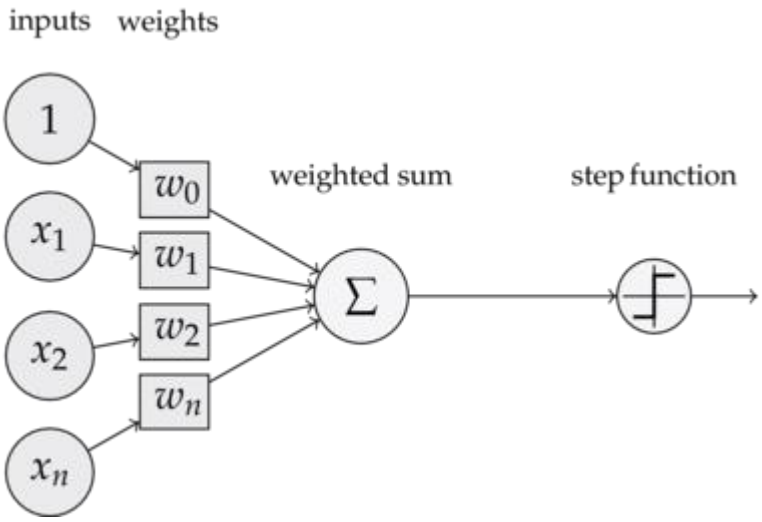
（当然了，该过程也有可能是非线性的，但是我们可以把其非线性过程施加到后面的非线性函数以及多层网络结构中），其次把突触之间的信号传递简化为对求和结果的非线性变换，那么上述模型就变得清晰了：



单层感知机

上面我们介绍的神经元的基本模型实际就是一个感知机的模型，该词最早出现于 1958 年，计算科学家 Rosenblatt 提出的由两层神经元组成的神经网络。

对前面的模型进一步符号化，如下图所示：



可以看到，感知机的基本模型包括：

输入：x 1 ，x 2 ，. . . ，x n 实际可能回比这更多,此处添加了一个偏置 1，是为了平衡线性加权函数总是过零点的问题。

权值：对应于每个输入都有一个加权的权值 w 1 ，w 2 ，. . . ，w n 。

激活函数：激活函数 f 对应于一个非线性函数，其选择有很多，本文后面会详细介绍。

输出 y：由激活激活函数进行处理后的结果，往往是区分度较大的非连续值用于分类。

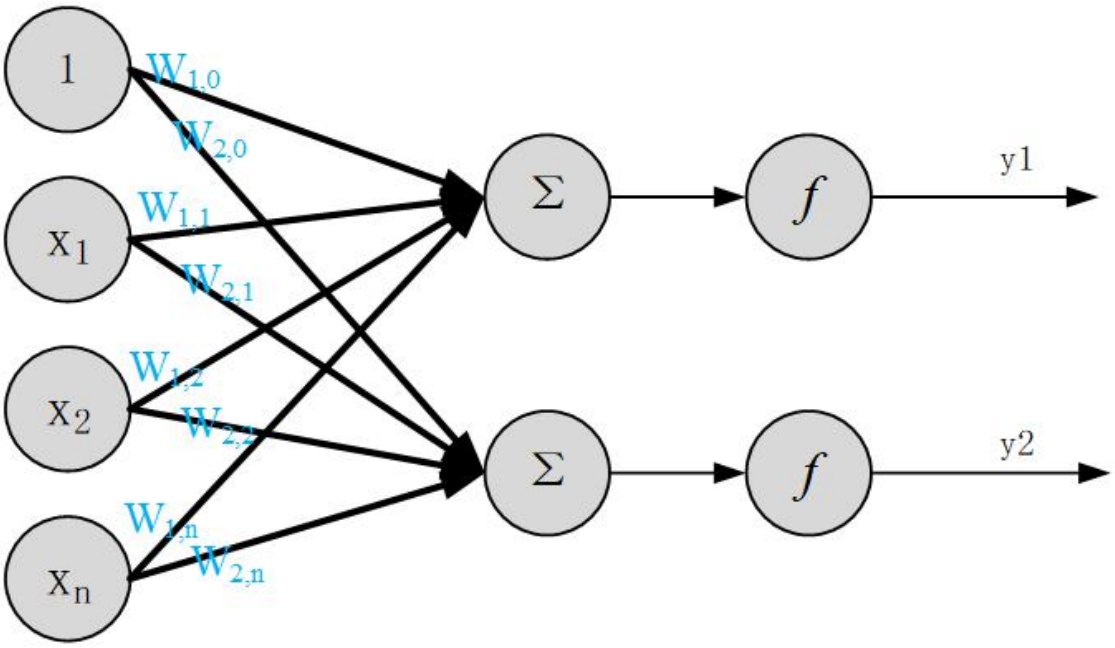
$$y=f(w0 \ +x1 \ * \ w1 \ +x2 \ * \ w2 \ +\cdots+xn \ * \ wn \ )$$

考虑向量的点乘过程，上式又可以简化为：

$$y=f(W\cdot x+w0 \ )$$

多层感知机

上面介绍了一个多输入单输出的感知机结构，单输出感知机会很大成都的影响到神经元的级联。实际上在人的大脑中，细胞核通常会通过多个树突与多个神经元相连，这就要考虑到如下的结构了：



上面这个图，稍显复杂，主要是在权重的体现上，权重的标号实际上由后一层向一层决定。如 w_1,1 是链接第二层的第一个节点与第一层的第一个节点（偏置不计），以此类推。那么根据上图就可以得到：

$$y1 = f(w1,0 + x1 * w1,1 + x2 * w1,2 + \cdots + xn * w1,n)$$

$$y2 = f(w2,0 + x1 * w2,1 + x2 * w2,2 + \cdots + xn * w2,n)$$

那么上式可以简化为矩阵向量相称的形式，即：

$$y = f(W \cdot x + w0)$$

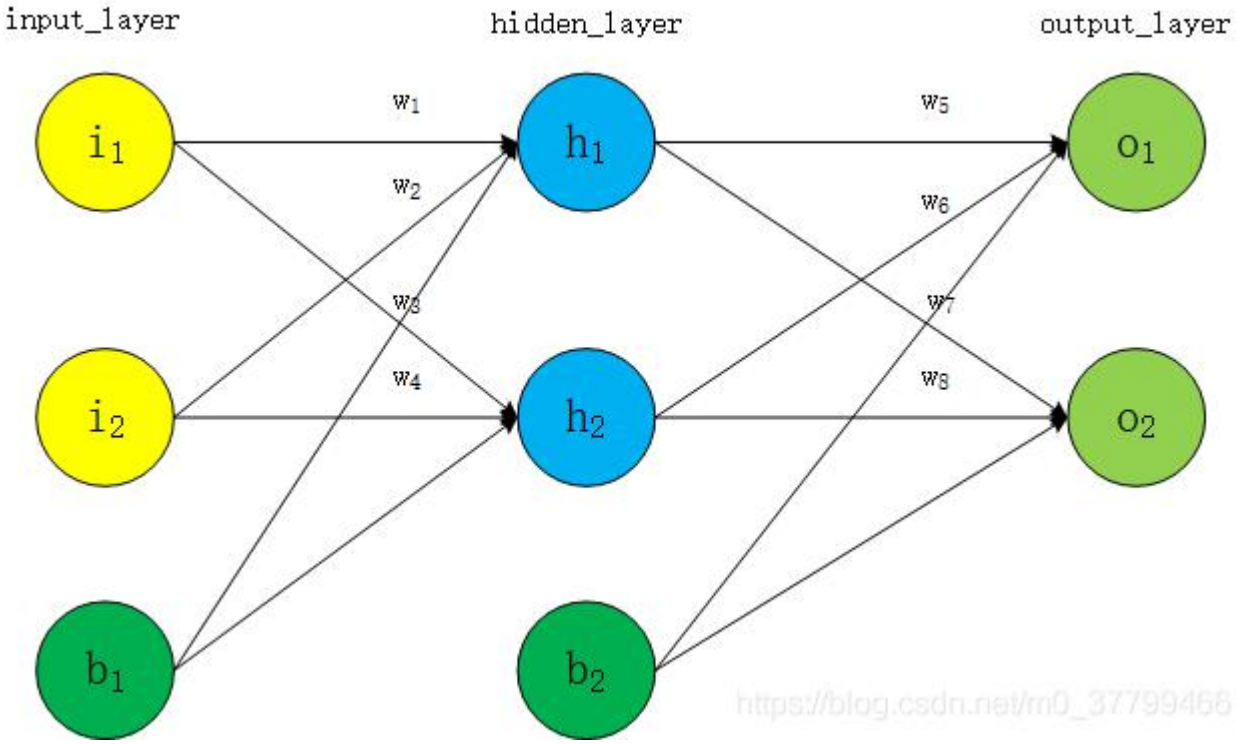
注意，此时式中

$W=[w1,1 \quad w2,1 \quad w1,2 \quad w2,2 \quad \ldots\ldots \quad w1,n \quad w2,n \quad]$

$x=\lfloor \mid \mid \lceil \quad x1 \quad x2 \quad \ldots xn \quad \rfloor \mid \mid \rceil$

4、什么是前向传播（包括图文示例）

前向传播的作用就是为了获取误差损失；



上图是一个典型的神经网络结构，包括了输入层、隐含层和输出层，为了更好的讲解，现在对其进行赋值：

输入数据：$i_1 = 0.05, i_2 = 0.10$

输出数据（即标签）：$o_1 = 0.01, o_2 = 0.99$

初始权重值：$w_1 = 0.15, w_2 = 0.20, w_3 = 0.25, w_4 = 0.30;$

初始权重值：$w_5 = 0.40, w_6 = 0.45, w_7 = 0.50, w_8 = 0.55;$

初始化偏置：$b_1 = 0.35, b_2 = 0.60$

目标：给出输入数据 i1,i2(0.05 和 0.10)，使输出尽可能与原

始输出 o1,o2(0.01 和 0.99)接近。

（1）.输入层----->隐含层：

计算神经元 h1 的输入加权和：

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

神经元 h1 的输出:(此处用到激活函数为 sigmoid 函数)：

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

同理，可计算出神经元 h2 的输出：

$$out_{h2} = 0.596884378$$

（2）.隐含层----->输出层：

同样的方法计算输出层神经元 o1 和 o2 的值：

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$$

$$out_{o2} = 0.772928465$$

这样前向传播的过程就结束了，我们得到输出值为[0.75136079，0.772928465]，与实际值[0.01，0.99]相差还很远，现在我们对误差进行反向传播，更新权值，重新计算输出。

5、什么是反向传播（包括图文示例）

反向传播的作用就是为了将误差传给每层，然后根据误差调整权重参数；

（1）.计算总误差

总误差：(square error)

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

但是有两个输出，所以分别计算 o1 和 o2 的误差，总误差为两者之和：

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

$$E_{o2} = 0.023560026$$

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

（2）.隐含层---->输出层的权值更新：

以权重参数 w5 为例，如果我们想知道 w5 对整体误差产生了多少影响，可以用整体误差对 w5 求偏导求出：（链式法则）

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

下面的图可以更直观的看清楚误差是怎样反向传播的：