



對外經濟貿易大學
UNIVERSITY OF INTERNATIONAL BUSINESS AND ECONOMICS

人工神经网络

信息学院-黄浩

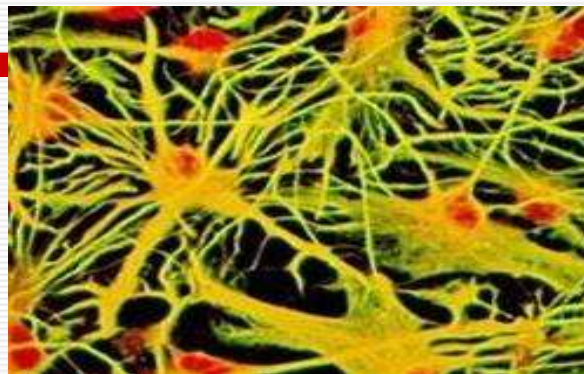
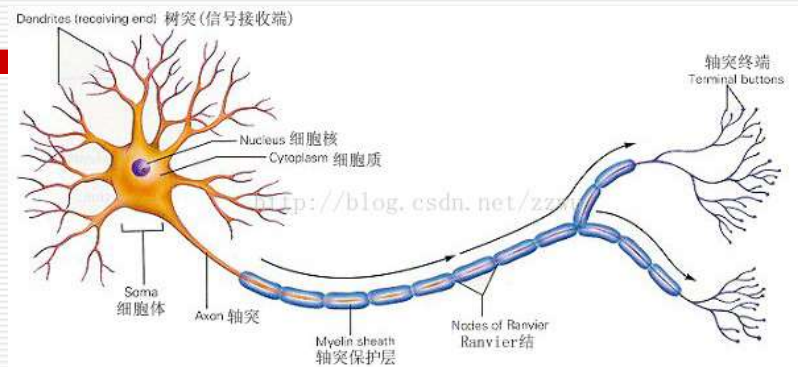
2022年3月14日星期一

本章主要介绍最常用的两种前馈网络：**BP (Back Propagation)**神经网络和径向基函数神经网络，以及它的前身感知器。

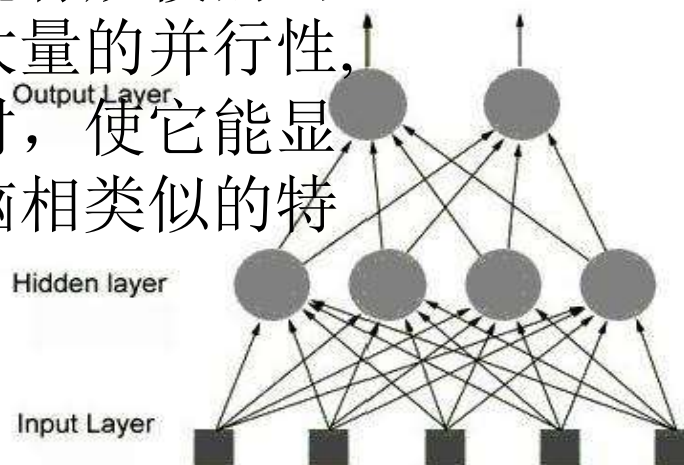
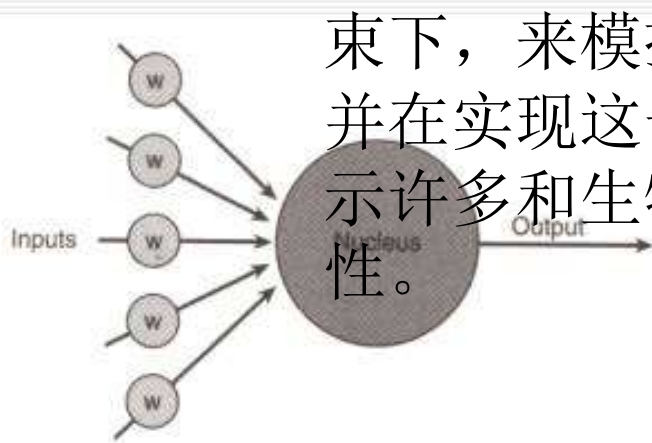
神经网络分类：



人工神经网络



一个神经网络(Artificial neural network,简称ANN)就是要在当代数字计算机现有规模的约束下,来模拟这种大量的并行性,并在实现这一工作时,使它能显示许多和生物学大脑相类似的特性。

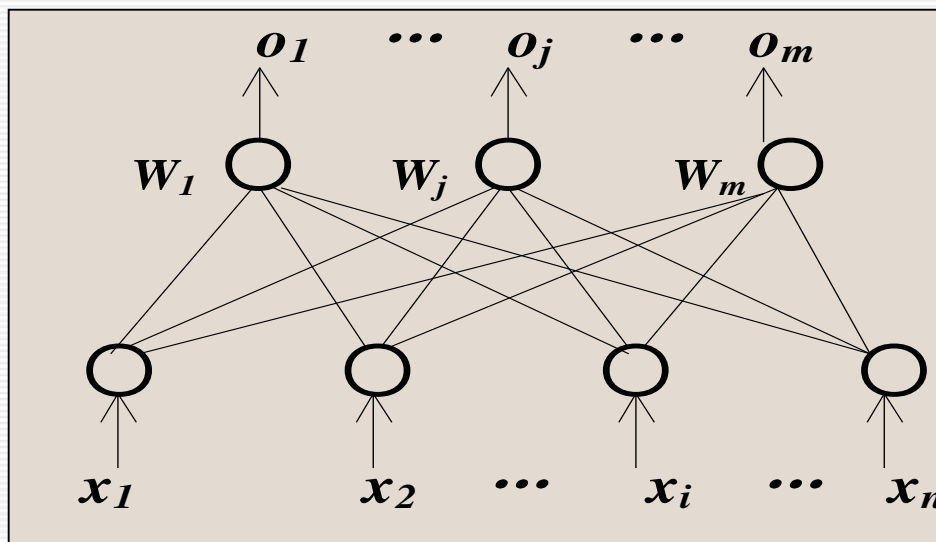


感知器

■1958年，美国心理学家Frank Rosenblatt提出一种具有单层计算单元的神经网络，即感知器。感知器是模拟人的视觉接受环境信息，并由神经冲动进行信息传递。感知器研究中首次提出了自组织、自学习的思想，而且对所解决的问题存在着收敛算法，并能从数学上严格证明，因而对神经网络的研究起了非常重要的推动作用。

单层感知器的结构和功能都很简单，以至于在解决实际问题中很少采用，但由于它在神经网络研究中具有重要意义，是研究其他网络的基础，而且较易学习和理解，适合于作为神经网络的起点。

感知器模型



$$\mathbf{X} = (x_1, x_2, \dots, x_i, \dots, x_n)^T$$

$$\mathbf{O} = (o_1, o_2, \dots, o_i, \dots, o_m)^T$$

$$W_j = (w_{1j}, w_{2j}, \dots, w_{ij}, \dots, w_{mj})^T \quad j = 1, 2, \dots, m$$

感知器模型

净输入：

$$net_j = \sum_{i=1}^n w_{ij} x_i$$

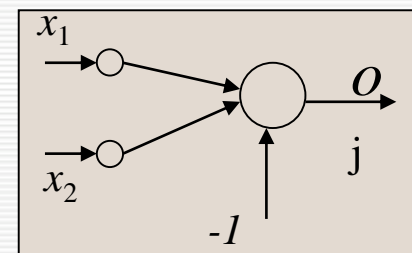
输出：

$$o_j = \text{sgn}(net_j - T_j) = \text{sgn}\left(\sum_{i=0}^n w_{ij} x_i\right) = \text{sgn}(\mathbf{W}_j^T \mathbf{X})$$

感知器的功能

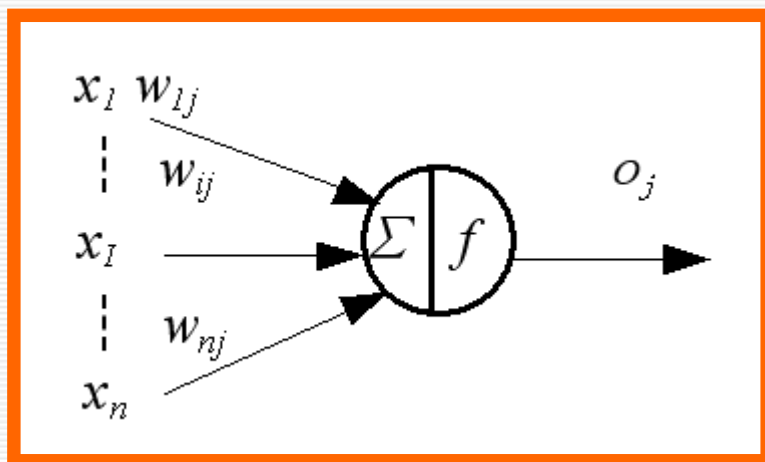
(1) 设输入向量 $X=(x_1, x_2)^T$

$$\text{输出: } o_j = \begin{cases} 1 & w_{1j}x_1 + w_{2j}x_2 - T_j > 0 \\ -1 & w_{1j}x_1 + w_{2j}x_2 - T_j < 0 \end{cases}$$

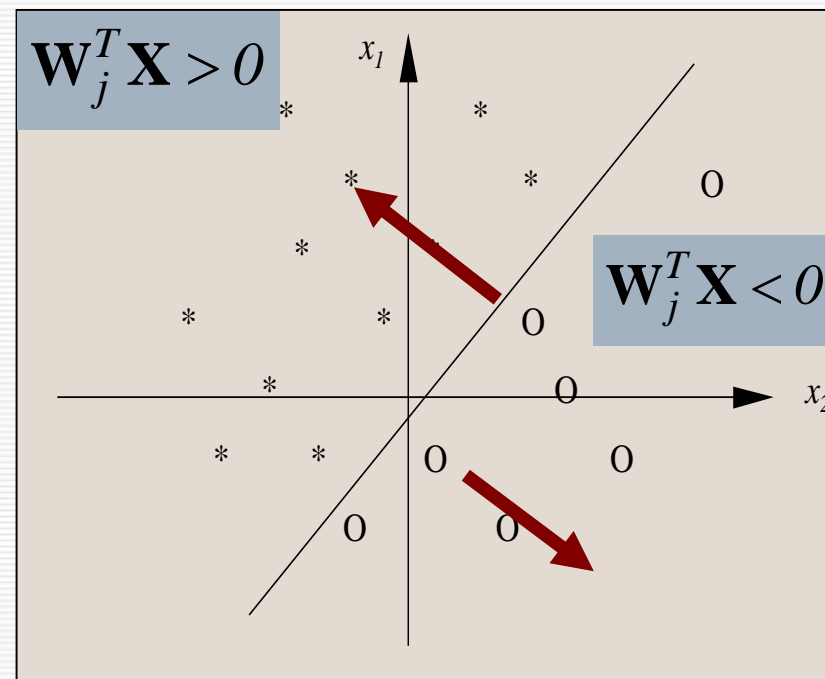


则由方程 $w_{1j}x_1 + w_{2j}x_2 - T_j = 0$
确定了二维平面上的一条分界线。

感知器的功能（二维）



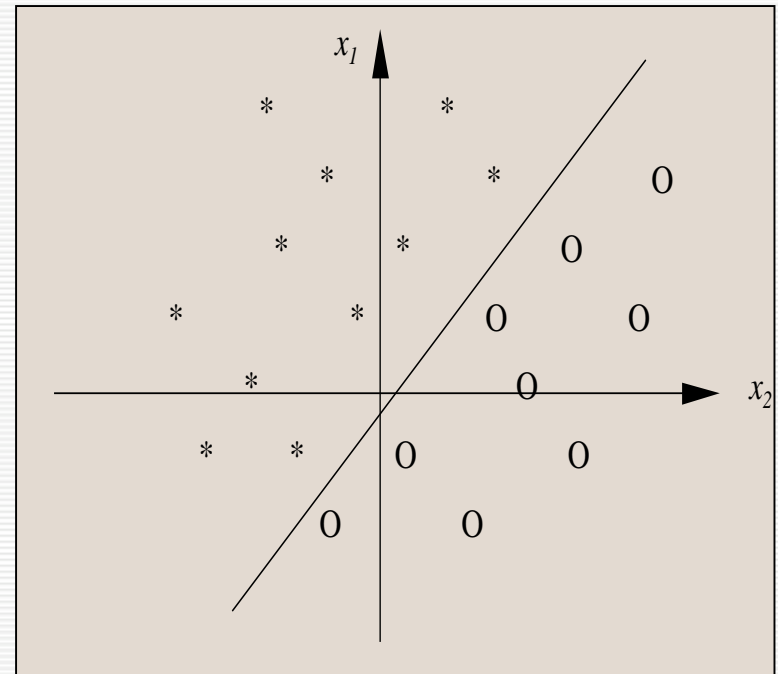
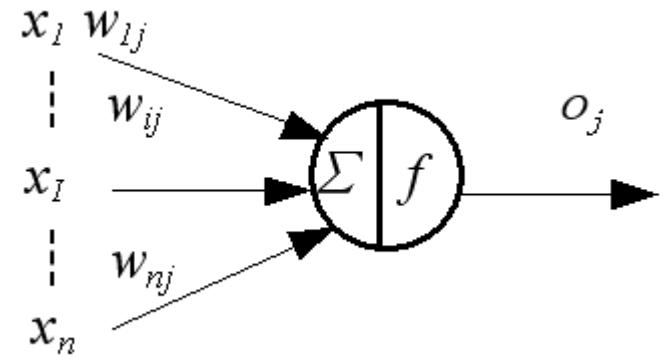
$$\mathbf{W}_j^T \mathbf{X} = 0$$



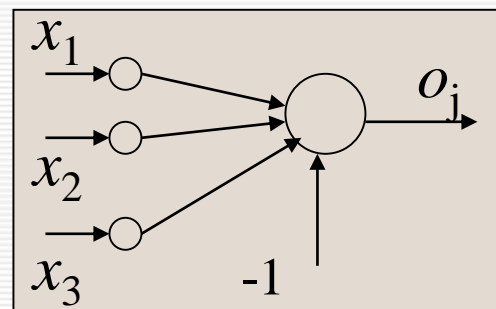
$$w_{ij} x_1 + w_{2j} x_2 - T_j = 0$$

$$w_{ij} x_1 = T_j - w_{2j} x_2$$

$$\begin{aligned} x_1 &= (T_j - w_{2j} x_2) / w_{ij} \\ &= - (w_{2j} / w_{ij}) x_2 + T_j / w_{ij} \\ &= a x_2 + c \end{aligned}$$



设输入向量 $X=(x_1, x_2, x_3)^T$



$$\text{输出： } o_j = \begin{cases} 1 & w_{1j}x_1 + w_{2j}x_2 + w_{3j}x_3 - T_j > 0 \\ -1 & w_{1j}x_1 + w_{2j}x_2 + w_{3j}x_3 - T_j = 0 \end{cases}$$

则由方程 $w_{1j}x_1 + w_{2j}x_2 + w_{3j}x_3 - T_j = 0$

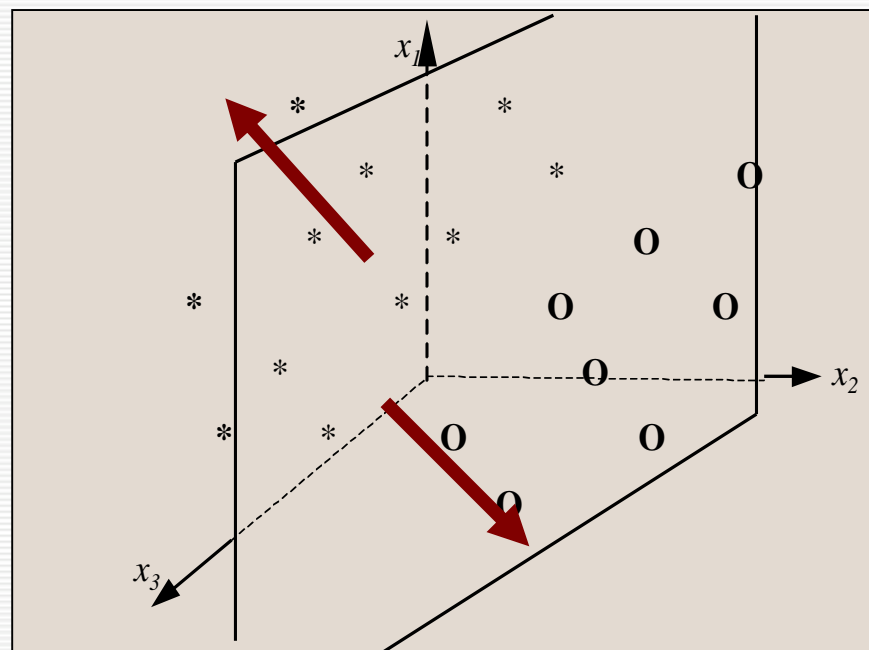
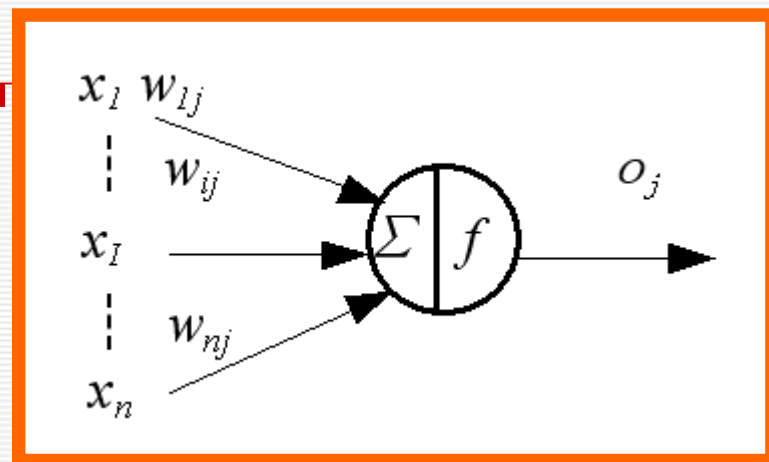
确定了三维空间上的一个分界平面。

$$\mathbf{W}_j^T \mathbf{X} = 0$$

是什么？

$$w_{1j}x_1 + w_{2j}x_2 + w_{3j}x_3 - T_j = 0$$

$$x_1 = a x_2 + b x_3 + c$$



设输入向量 $X=(x_1, x_2, \dots, x_n)^T$

输出 $w_{1j}x_1 + w_{2j}x_2 + \dots + w_{nj}x_n - T_j = 0$

则由方程 $w_{1j}x_1 + w_{2j}x_2 + \dots + w_{nj}x_n - T_j = 0$

确定了n维空间上的一个分界平面。

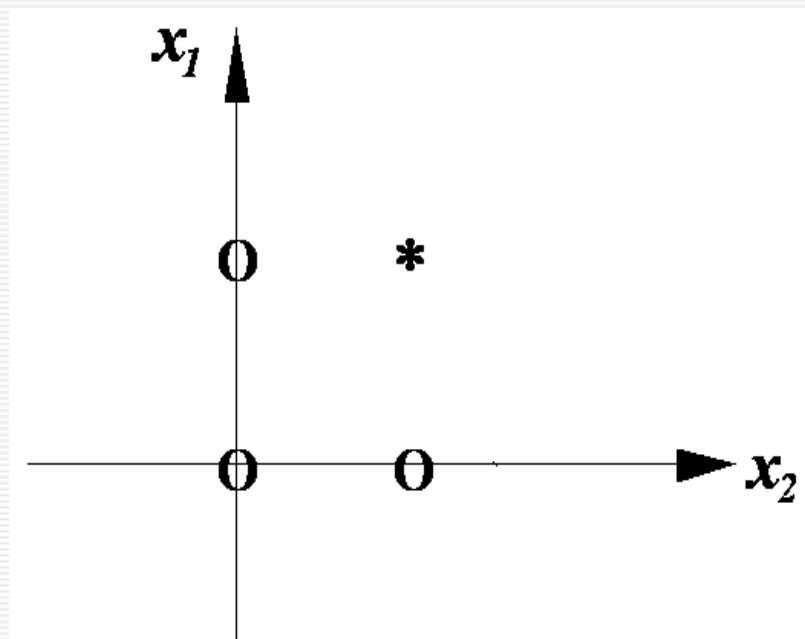
感知器的功能

一个最简单的单计算节点感知器具有分类功能。其分类原理是将分类知识存储于感知器的权向量（包含了阈值）中，由权向量确定的分类判决界面将输入模式分为两类。

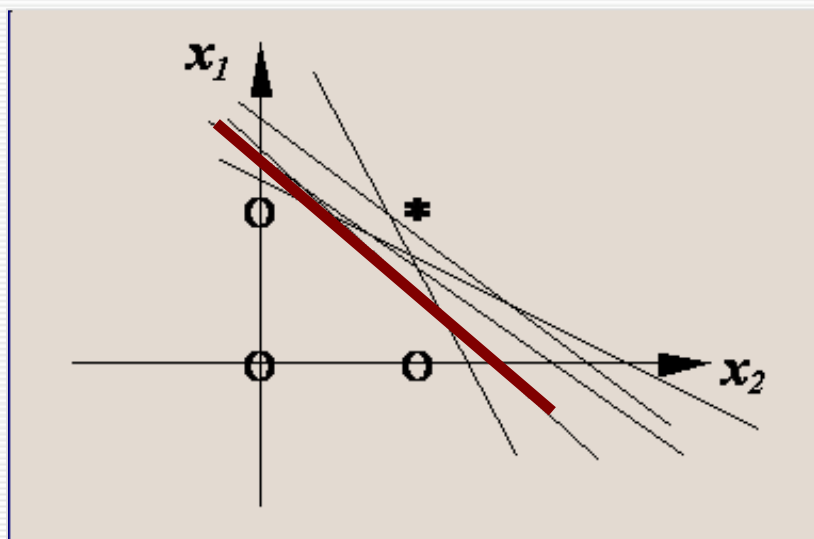
例 用感知器实现逻辑“与”功能

逻辑“与”真值表

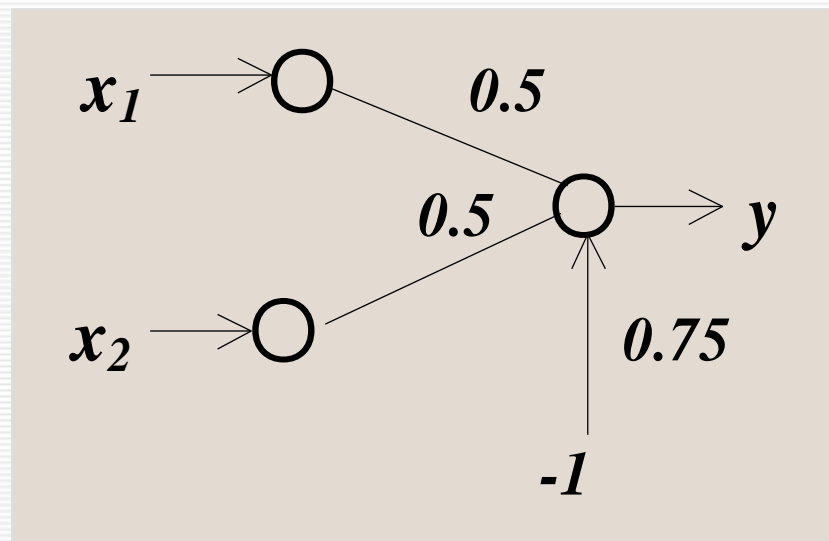
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



例一 用感知器实现逻辑“与”功能



感知器结构



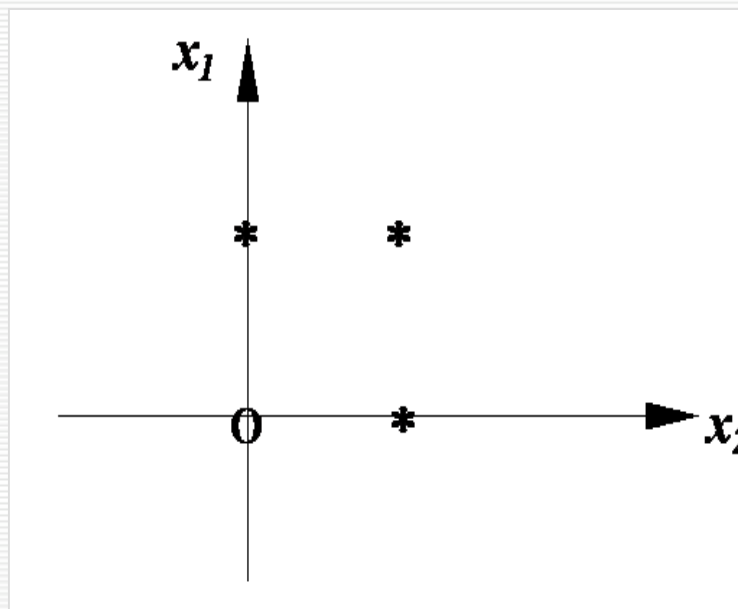
$$w_1x_1 + w_2x_2 - T = 0$$

$$0.5x_1 + 0.5x_2 - 0.75 = 0$$

例 用感知器实现逻辑“或”功能

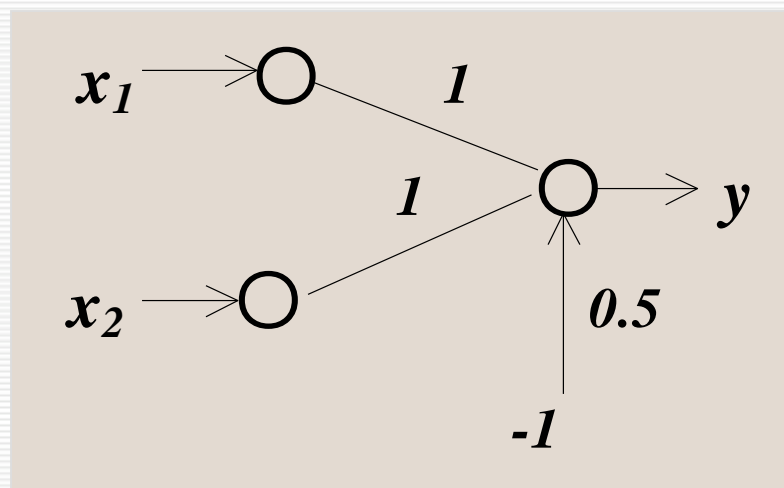
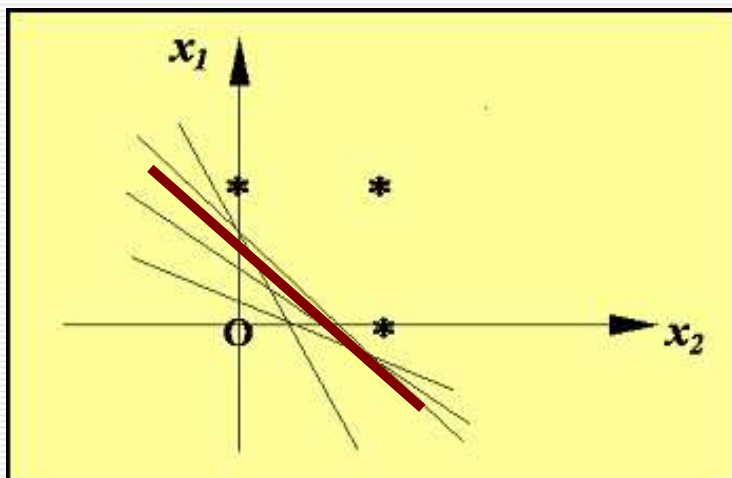
逻辑“或”真值表

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1



用感知器实现逻辑“或”功能

感知器结构

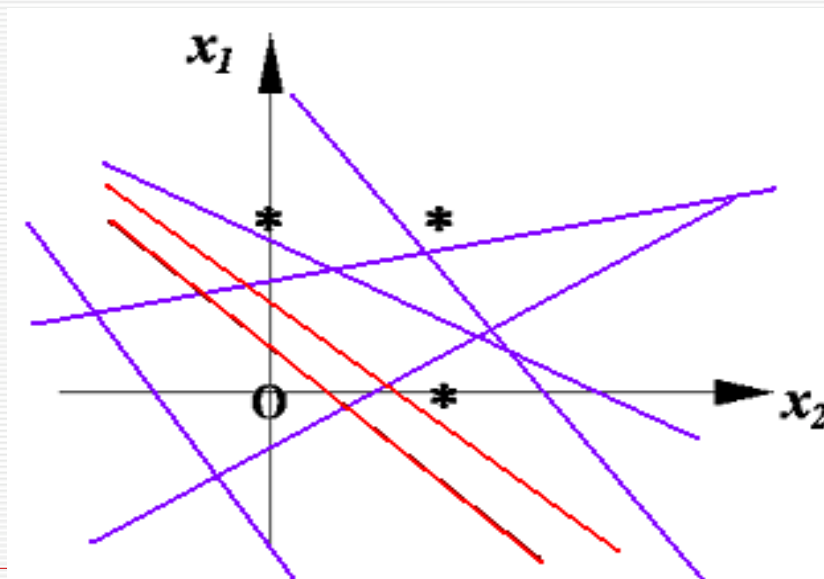
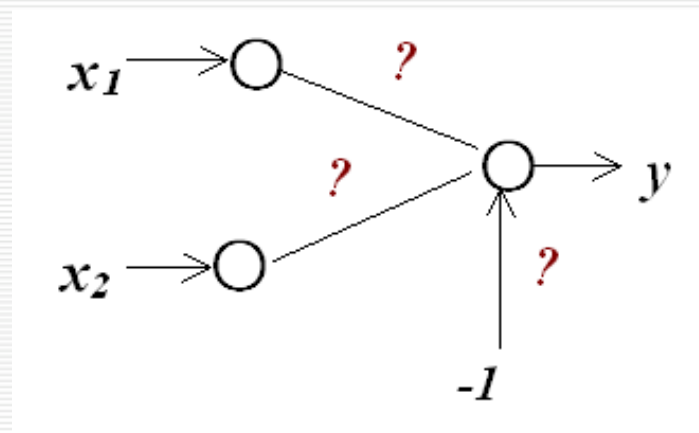


$$w_1x_1 + w_2x_2 - T = 0$$

$$x_1 + x_2 - 0.5 = 0$$

□ 关键问题就是求

$$\mathbf{W}_j^T \mathbf{X} = 0$$



感知器的学习

学习信号等于神经元期望输出(教师信号)与实际输出之差

$$r = d_j - o_j \quad (2.15)$$

式中 d_j 为期望的输出, $o_j = f(W_j^T X)$ 。

因此, 权值调整公式应为

$$\Delta W_j = \eta [d_j - \text{sgn}(W_j^T X)] X \quad (2.17a)$$

$$\Delta w_{ij} = \eta [d_j - \text{sgn}(W_j^T X)] x_i \quad i = 0, 1, \dots, n \quad (2.17b)$$

式中, 当实际输出与期望值相同时, 权值不需要调整。感知器学习规则代表一种有导师学习。

感知器的学习

感知器学习规则的训练步骤：

- (1) 对各权值 $w_{0j}(0), w_{1j}(0), \dots, w_{nj}(0)$, $j=1, 2, \dots, m$
(m 为计算层的节点数) 赋予较小的非零随机数；
- (2) 输入样本对 $\{X^p, d^p\}$ ，其中 $X^p=(-1, x_1^p, x_2^p, \dots, x_n^p)$ ，
 d^p 为期望的输出向量（教师信号），上标 p 代表
样本对的模式序号，设样本集中的样本总数为 P ，
则 $p=1, 2, \dots, P$ ；

感知器的学习

- (3) 计算各节点的实际输出 $o_j^p(t) = \text{sgn}[W_j^T(t)X^p]$, $j=1, 2, \dots, m$;
- (4) 调整各节点对应的权值, $W_j(t+1) = W_j(t) + \eta[d_j^p - o_j^p(t)]X^p$, $j=1, 2, \dots, m$, 其中 η 为学习率, 用于控制调整速度, 太大会影响训练的稳定性, 太小则使训练的收敛速度变慢, 一般取 $0 < \eta \leq 1$;
- (5) 返回到步骤(2)输入下一对样本, 周而复始直到对所有样本, 感知器的实际输出与期望输出相等。

感知器的学习

例 单计算节点感知器，3个输入。给定3对训练样本对如下：

$$X^1 = (-1, 1, -2, 0)^T \quad d^1 = -1$$

$$X^2 = (-1, 0, 1.5, -0.5)^T \quad d^2 = -1$$

$$X^3 = (-1, -1, 1, 0.5)^T \quad d^3 = 1$$

设初始权向量 $W(0) = (0.5, 1, -1, 0)^T$ ， $\eta = 0.1$ 。注意，输入向量中第一个分量 x_0 恒等于 -1，权向量中第一个分量为阈值，试根据以上学习规则训练该感知器。

感知器的学习

解：第一步 输入 X^1 ，得

$$W^T(0)X^1=(0.5,1,-1,0)(-1,1,-2,0)^T=2.5$$

$$o^1(0)=\text{sgn}(2.5)=1$$

$$\begin{aligned}W(1) &= W(0) + \eta [d^1 - o^1(0)] X^1 \\&= (0.5, 1, -1, 0)^T + 0.1(-1-1)(-1, 1, -2, 0)^T \\&= (0.7, 0.8, -0.6, 0)^T\end{aligned}$$

感知器的学习

第二步 输入 X^2 , 得

$$W^T(1)X^2=(0.7,0.8,-0.6,0)(-1,0,1.5,-0.5)^T=-1.6$$

$$o^2(1)=\text{sgn}(-1.6)=-1$$

$$\begin{aligned}W(2) &= W(1) + \eta [d^2 - o^2(1)] X^2 \\&= (0.7, 0.8, -0.6, 0)^T + 0.1 [-1 - (-1)] (-1, 0, 1.5, -0.5)^T \\&= (0.7, 0.8, -0.6, 0)^T\end{aligned}$$

由于 $d^2 = o^2(1)$, 所以 $W(2) = W(1)$ 。

感知器的学习

第三步 输入 X^3 , 得

$$W^T(2)X^3=(0.7,0.8,-0.6,0)(-1,-1,1,0.5)^T=-2.1$$

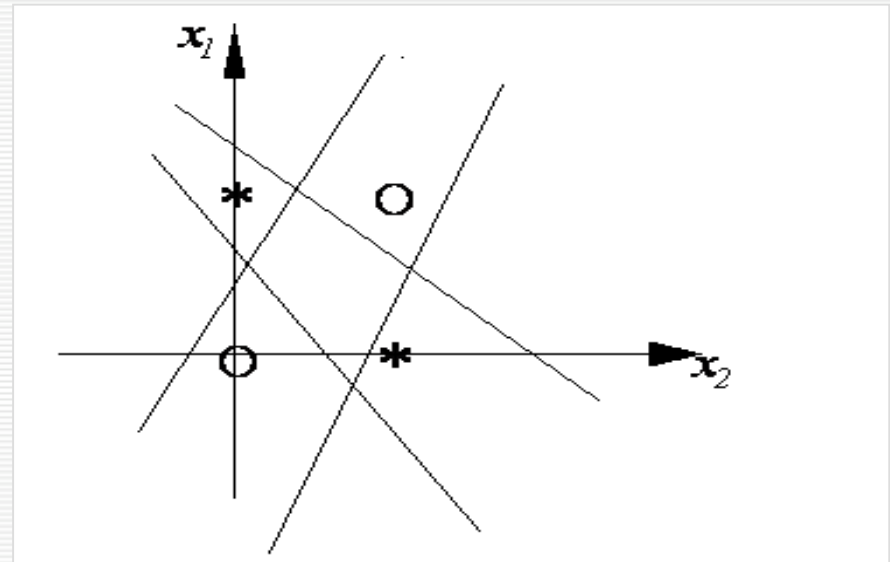
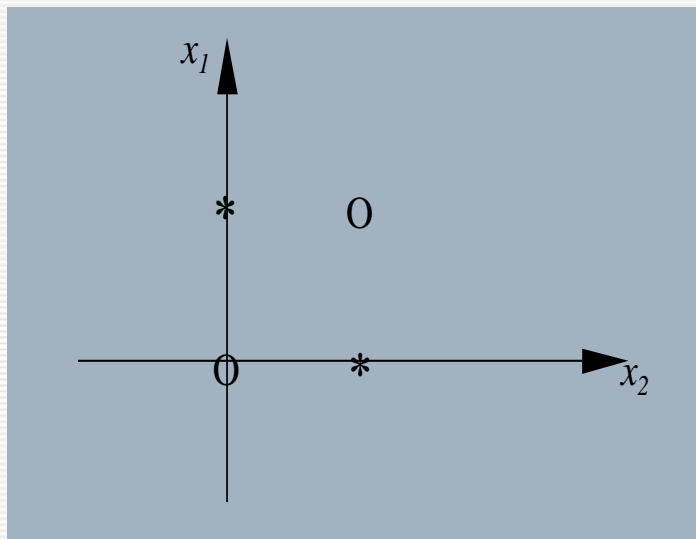
$$O^3(2)=\text{sgn}(-2.1)=-1$$

$$\begin{aligned}W(3) &= W(2) + \eta [d^3 - o^3(2)] X^3 \\&= (0.7, 0.8, -0.6, 0)^T + 0.1 [1 - (-1)] (-1, -1, 1, 0.5)^T \\&= (0.5, 0.6, -0.4, 0.1)^T\end{aligned}$$

第四步 返回到第一步, 继续训练直到 $d^p - o^p = 0$, $p=1, 2, 3$ 。

单层感知器的局限性

□ 问题：能否用感知器解决如下问题？

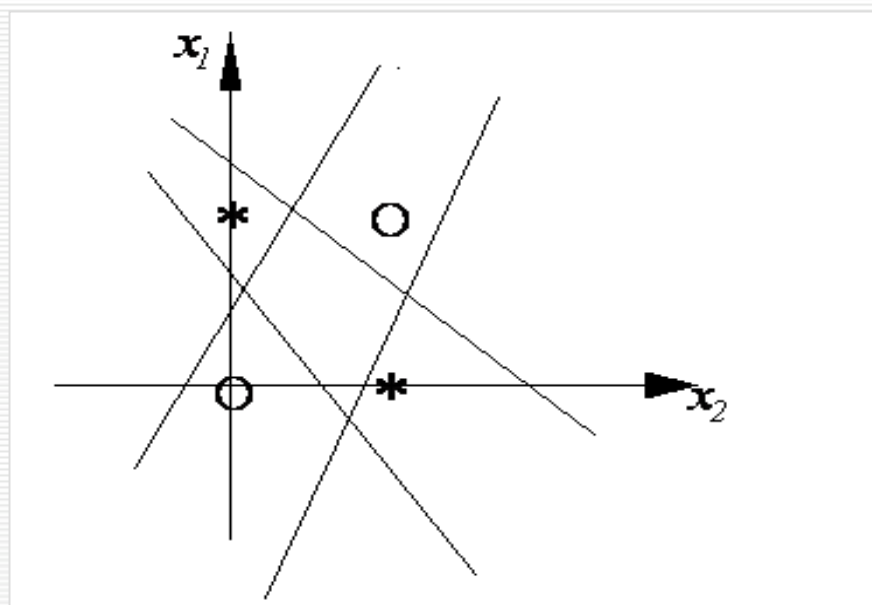


单层感知器的局限性

□ 无法解决“异或”问题

“异或”的真值表

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

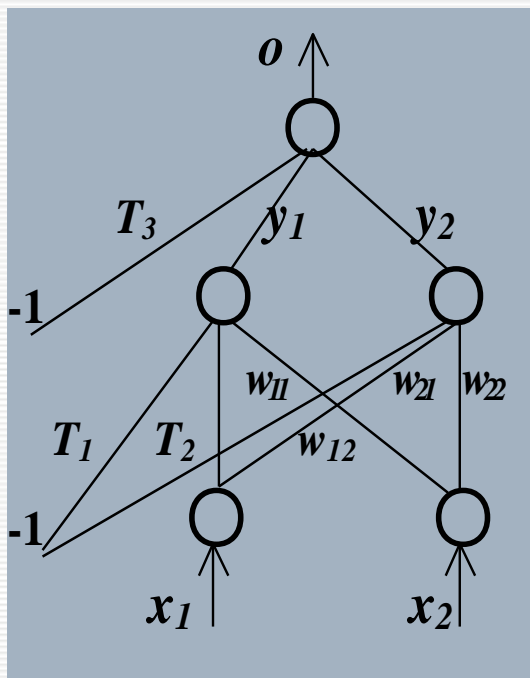


□ 只能解决线性可分问题

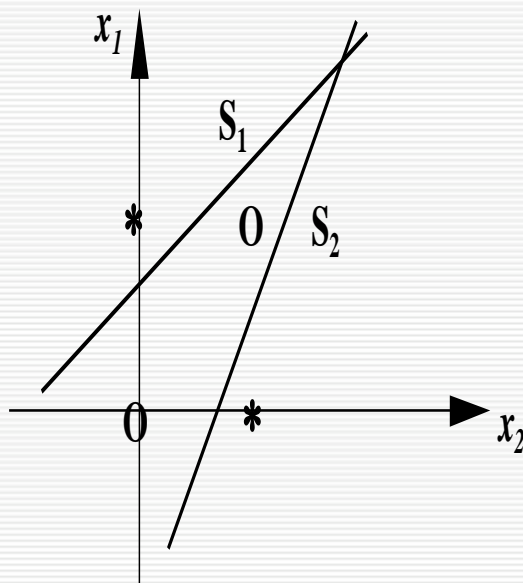
多层感知器

例四 用两计算层感知器解决“异或”问题。

双层感知器



“异或”问题分类



“异或”的真值表

x_1	x_2	y_1	y_2	o
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	1	1	0

多层感知器的提出

■ 单计算层感知器的局限性

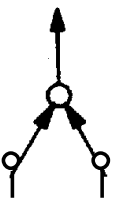
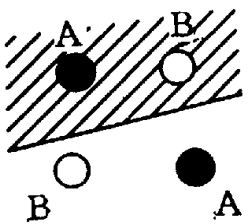
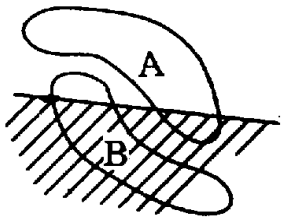

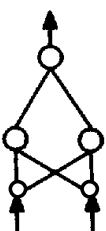
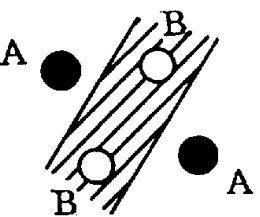
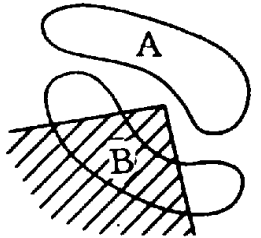

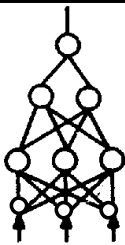
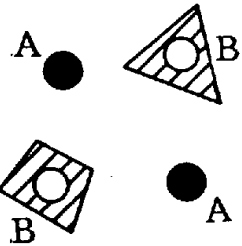
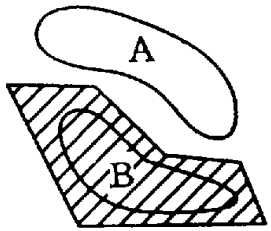
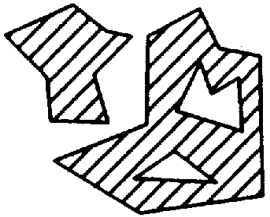
只能解决线性可分问题，而大量的分类问题是线性不可分的。

■ 解决的有效办法

在输入层与输出层之间引入隐层作为输入模式的“内部表示”，将单计算层感知器变成多（计算）层感知器。

采用非线性连续函数作为转移函数，使区域边界线的基本线素由直线变成曲线，从而使整个边界线变成连续光滑的曲线。

多层感知器

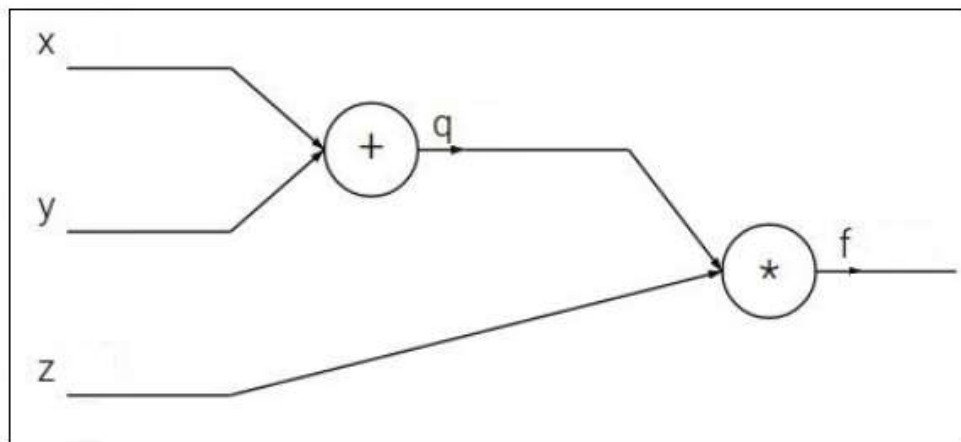
感知器结构	异或问题	复杂问题	判决域形状	判决域
无隐层 				半平面
单隐层 				凸 域
双隐层 				任意复杂 形状域

具有不同隐层数的感知器的分类能力对比

计算图

Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

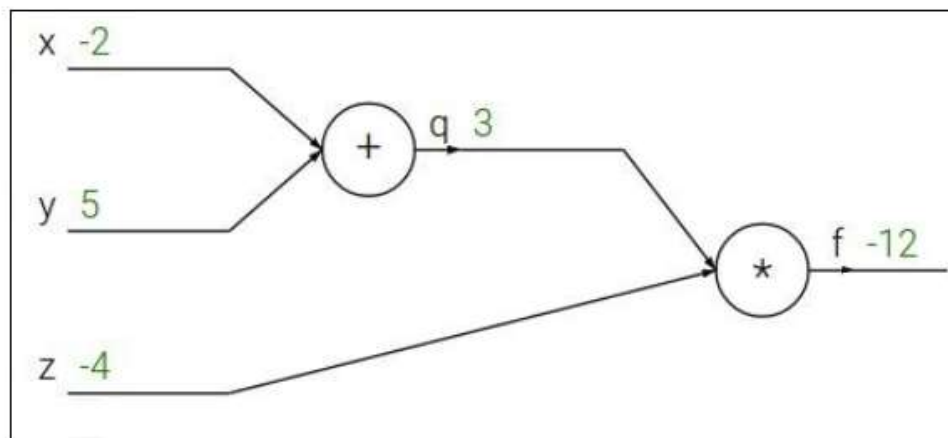


计算图

Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$



计算图

Backpropagation: a simple example

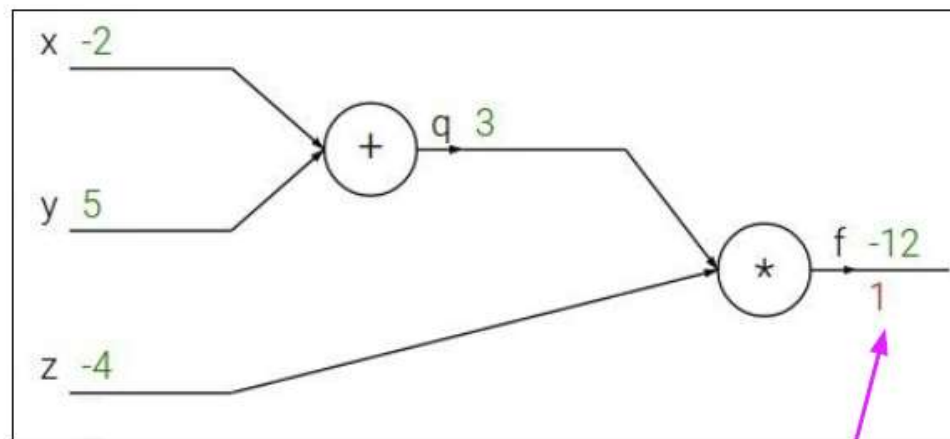
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial f}$$

计算图

Backpropagation: a simple example

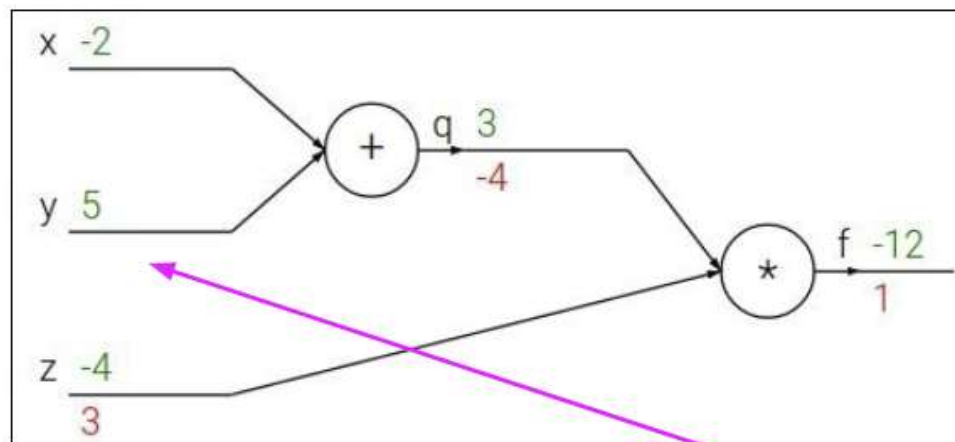
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

Upstream
gradient

Local
gradient

$$\frac{\partial f}{\partial y}$$

计算图

Backpropagation: a simple example

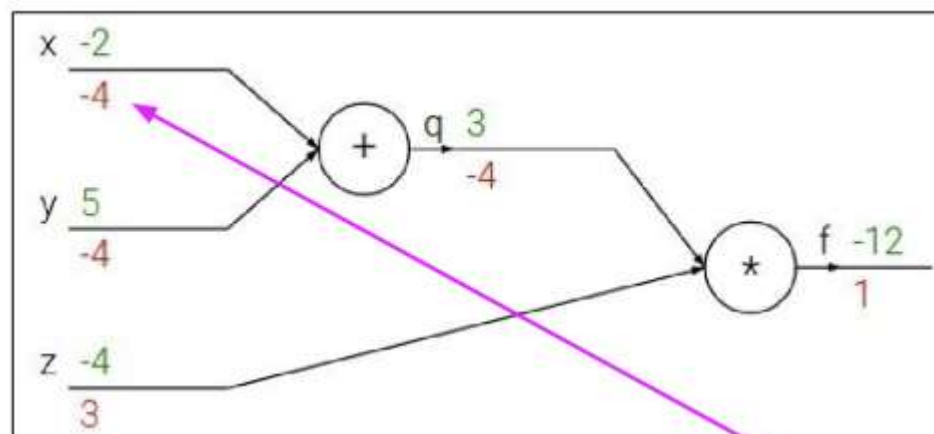
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

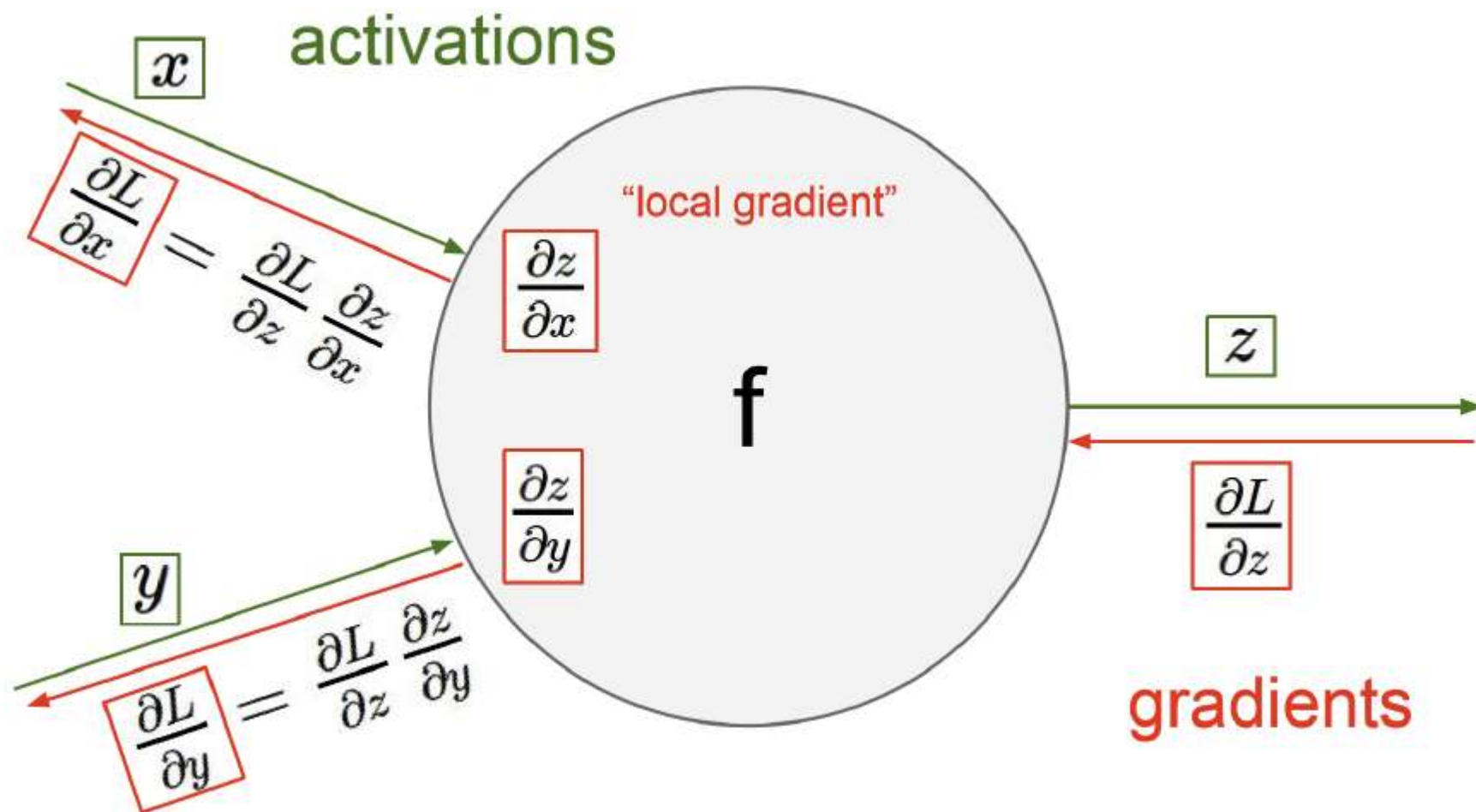


$$\frac{\partial f}{\partial x}$$

Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

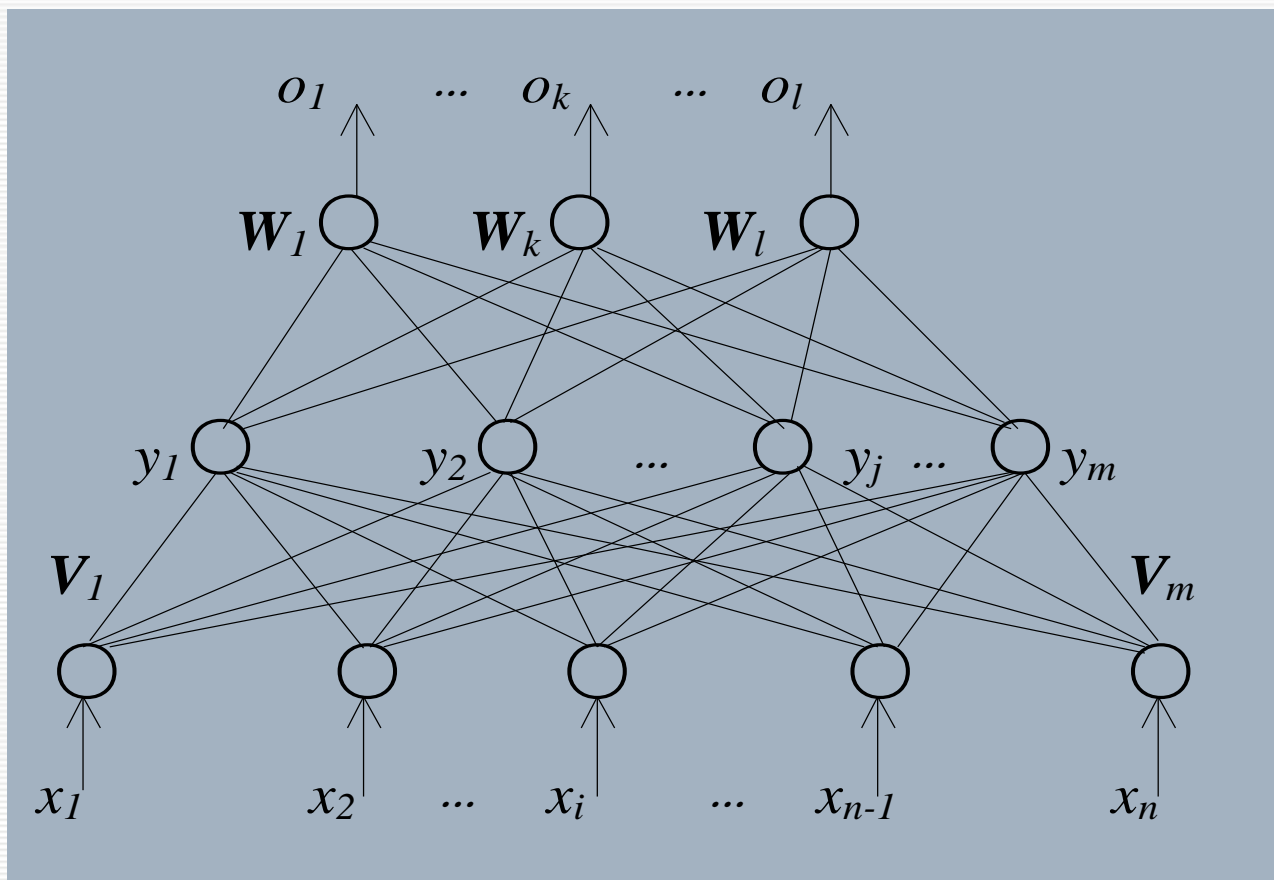
计算图



误差反传（BP）算法

- 误差反向传播算法简称BP算法，其基本思想是最小二乘法。它采用梯度搜索技术，以期使网络的实际输出值与期望输出值的误差均方值为最小。
- BP算法的学习过程由正向传播和反向传播组成。在正向传播过程中，输入信息从输入层经隐含层逐层处理，并传向输出层，每层神经元（节点）的状态只影响下一层神经元的状态。如果在输出层不能得到期望输出，则转入反向传播，将误差信号沿原来的连接通路返回，通过修改各层神经元的权值，使误差信号最小。

误差反传 (BP) 算法



基于BP算法的多层前馈网络模型



对外经济贸易大学
UNIVERSITY OF INTERNATIONAL BUSINESS AND ECONOMICS

□ 模型的数学表达

输入向量: $X=(x_1, x_2, \dots, x_i, \dots, x_n)^T$

隐层输出向量: $Y=(y_1, y_2, \dots, y_j, \dots, y_m)^T$

输出层输出向量: $O=(o_1, o_2, \dots, o_k, \dots, o_l)^T$

期望输出向量: $d=(d_1, d_2, \dots, d_k, \dots, d_l)^T$

输入层到隐层之间的权值矩阵: $V=(V_1, V_2, \dots, V_j, \dots, V_m)$

隐层到输出层之间的权值矩阵: $W=(W_1, W_2, \dots, W_k, \dots, W_l)$

各个变量之间如何建立联系, 来描述整个网络?

BP学习算法

对于输出层: $o_k = f(net_k) \quad k=1,2,\dots,l \quad (3.1)$

$$net_k = \sum_{j=0}^m w_{jk} y_j \quad k=1,2,\dots,l \quad (3.2)$$

对于隐层: $y_j = f(net_j) \quad j=1,2,\dots,m \quad (3.3)$

$$net_j = \sum_{i=0}^n v_{ij} x_i \quad j=1,2,\dots,m \quad (3.4)$$

BP学习算法

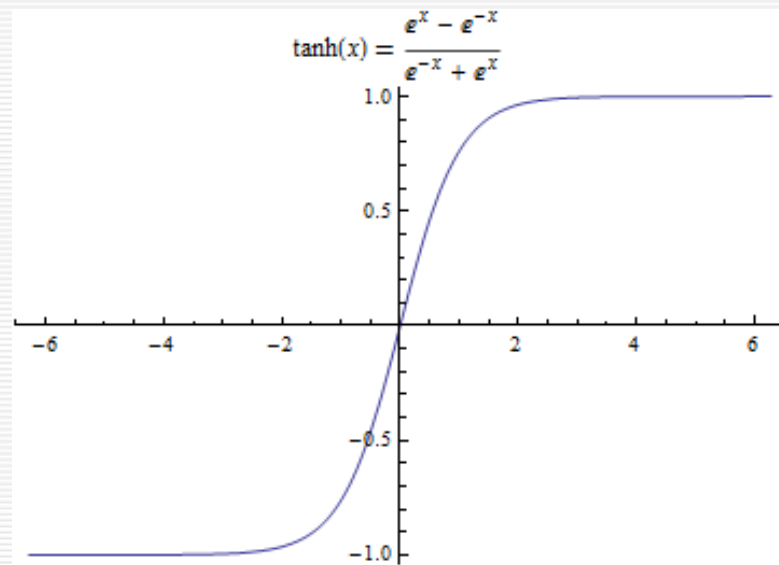
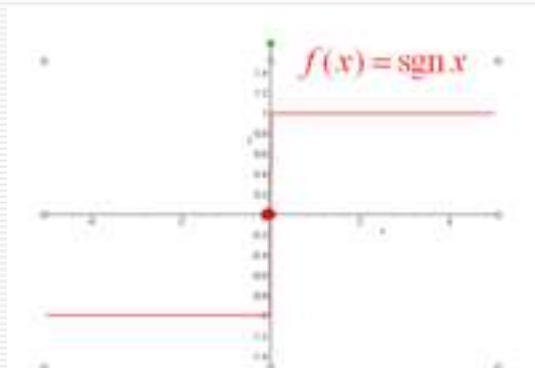
单极性Sigmoid函数：

$$f(x) = \frac{1}{1 + e^{-x}} \quad (3.5)$$

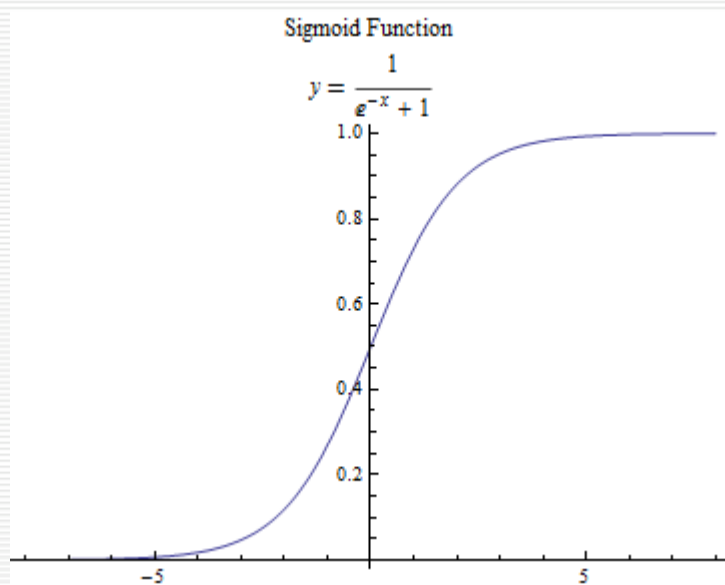
双极性Sigmoid函数：

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

BP学习算法



双曲正切函数图像



BP学习算法

输出误差 E 定义：

$$E = \frac{1}{2} (d - O)^2 = \frac{1}{2} \sum_{k=1}^l (d_k - o_k)^2 \quad (3.6)$$

将以上误差定义式展开至隐层：

$$E = \frac{1}{2} \sum_{k=1}^l [d_k - f(net_k)]^2 = \frac{1}{2} \sum_{k=1}^l [d_k - f(\sum_{j=0}^m w_{jk} y_j)]^2 \quad (3.7)$$

BP学习算法

进一步展开至输入层：

$$E = \frac{1}{2} \sum_{k=1}^l \{d_k - f[\sum_{j=0}^m w_{jk} f(net_j)]\}^2 \quad (3.8)$$

$$= \frac{1}{2} \sum_{k=1}^l \{d_k - f[\sum_{j=0}^m w_{jk} f(\sum_{i=0}^n v_{ij} x_i)]\}^2$$

BP学习算法

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}}$$

$$j=0,1,2,\dots,m; \quad k=1,2,\dots,l \quad (3.9a)$$

$$\Delta v_{ij} = -\eta \frac{\partial E}{\partial v_{ij}}$$

$$i=0,1,2,\dots,n; \quad j=1,2,\dots,m \quad (3.9b)$$

式中负号表示梯度下降，常数 $\eta \in (0,1)$ 表示比例系数。

在全部推导过程中，对输出层有 $j=0,1,2,\dots,m; \quad k=1,2,\dots,l$
对隐层有 $i=0,1,2,\dots,n; \quad j=1,2,\dots,m$

BP学习算法

对于输出层，式(3.9a)可写为

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} = -\eta \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial w_{jk}} \quad (3.10a)$$

对隐层，式(3.9b)可写为

$$\Delta v_{ij} = -\eta \frac{\partial E}{\partial v_{ij}} = -\eta \frac{\partial E}{\partial net_j} \frac{\partial net_j}{\partial v_{ij}} \quad (3.10b)$$

对输出层和隐层各定义一个误差信号，令

$$\delta_k^o = -\frac{\partial E}{\partial net_k} \quad (3.11a)$$

$$\delta_j^y = -\frac{\partial E}{\partial net_j} \quad (3.11b)$$

综合应用式(3.2)和(3.11a), 可将式 (3.10a)的权值调整式改写为

$$\Delta w_{jk} = \eta \delta_k^o y_j \quad (3.12a)$$

综合应用式(3.4)和(3.11b), 可将式 (3.10b)的权值调整式改写为

$$\Delta v_{ij} = \eta \delta_j^y x_i \quad (3.12b)$$

可以看出, 只要计算出式(3.12)中的误差信号 δ^o 和 δ^y , 权值调整量的计算推导即可完成。下面继续推导如何求误差信号 δ^o 和 δ^y 。

对于输出层, δ^o 可展开为

$$\delta_k^o = -\frac{\partial E}{\partial net_k} = -\frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial net_k} = -\frac{\partial E}{\partial o_k} f'(net_k) \quad (3.13a)$$

对于隐层, δ^y 可展开为

$$\delta_j^y = -\frac{\partial E}{\partial net_j} = -\frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial net_j} = -\frac{\partial E}{\partial y_j} f'(net_j) \quad (3.13b)$$

下面求式(3.13)中网络误差对各层输出的偏导。

对于输出层，利用式(3.6)：

$$E = \frac{1}{2} \sum_{k=1}^l (d_k - o_k)^2$$

可得：

$$\frac{\partial E}{\partial o_k} = -(d_k - o_k) \quad (3.14a)$$

对于隐层，利用式(3.7)：

$$E = \frac{1}{2} \sum_{k=1}^l [d_k - f(\sum_{j=0}^m w_{jk} y_j)]^2$$

可得：

$$\frac{\partial E}{\partial y_j} = - \sum_{k=1}^l (d_k - o_k) f'(net_k) w_{jk} \quad (3.14b)$$

将以上结果代入式(3.13)，并应用式(3.15)

得到：

$$\delta_k^o = (d_k - o_k) o_k (1 - o_k) \quad (3.15a)$$

$$\delta_j^y = \left[\sum_{k=1}^l (d_k - o_k) f'(net_k) w_{jk} \right] f'(net_j)$$

$$= \left(\sum_{k=1}^l \delta_k^o w_{jk} \right) y_j (1 - y_j) \quad (3.15b)$$

至此两个误差信号的推导已完成。

将式(3.15)代回到式(3.12)，得到三层前馈网的BP学习算法权值调整计算公式为：

$$\left\{ \begin{array}{l} \Delta w_{jk} = \eta \delta_k^o y_j = \eta (d_k - o_k) o_k (1 - o_k) y_j \quad (3.16a) \\ \Delta v_{ij} = \eta \delta_j^y x_i = \eta \left(\sum_{k=1}^l \delta_k^o w_{jk} \right) y_j (1 - y_j) x_i \quad (3.16b) \end{array} \right.$$

$$o_k = f(net_k)$$

$$y_j = f(net_j)$$

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

$$net_k = \sum_{j=0}^m w_{jk} y_j$$

$$net_j = \sum_{i=0}^n v_{ij} x_i$$

$$\begin{aligned} E &= \frac{1}{2} \sum_{k=1}^l (d_k - o_k)^2 = \frac{1}{2} \sum_{k=1}^l [d_k - f(net_k)]^2 = \frac{1}{2} \sum_{k=1}^l [d_k - f(\sum_{j=0}^m w_{jk} y_j)]^2 \\ &= \frac{1}{2} \sum_{k=1}^l \{d_k - f[\sum_{j=0}^m w_{jk} f(net_j)]\}^2 = \frac{1}{2} \sum_{k=1}^l \{d_k - f[\sum_{j=0}^m w_{jk} f(\sum_{i=0}^n v_{ij} x_i)]\}^2 \end{aligned}$$

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} = -\eta \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial w_{jk}}$$

$$\delta_k^o = -\frac{\partial E}{\partial net_k}$$

$$\Delta w_{jk} = \eta \delta_k^o y_j$$

$$\Delta v_{ij} = -\eta \frac{\partial E}{\partial v_{ij}} = -\eta \frac{\partial E}{\partial net_j} \frac{\partial net_j}{\partial v_{ij}}$$

$$\delta_j^y = -\frac{\partial E}{\partial net_j}$$

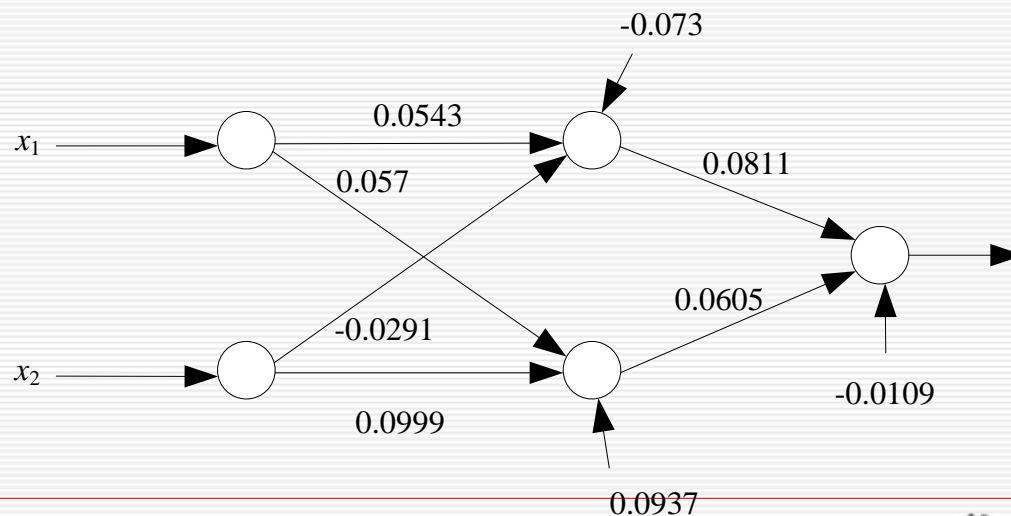
$$\Delta v_{ij} = \eta \delta_j^y x_i$$

BP学习算法流程：

- 1) 初始化 置所有权值为最小的随机数；
- 2) 提供训练集 给定输入向量和期望的目标输出向量；
- 3) 计算实际输出 计算隐含层、输出层各神经元输出；
- 4) 计算目标值与实际输出的偏差；
- 5) 计算局部梯度；
- 6) 调整各层权重；
- 7) 返回2) 重复计算，直到误差满足要求为止。

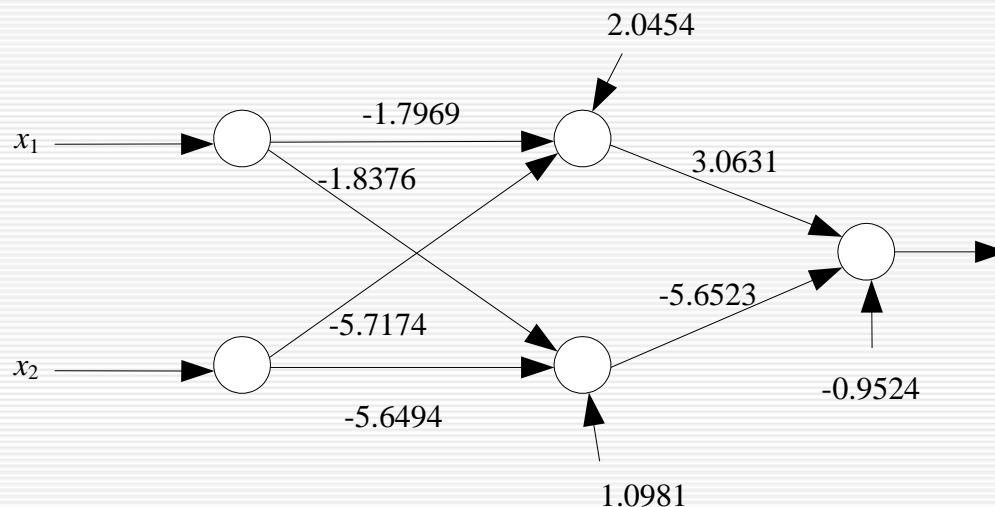
迭代1次后的BP网络

x_1	x_2	y	实际输出	误差
0	0	0	0.5	0.5
0	1	1	0.5	
1	0	1	0.5	
1	1	0	0.5	



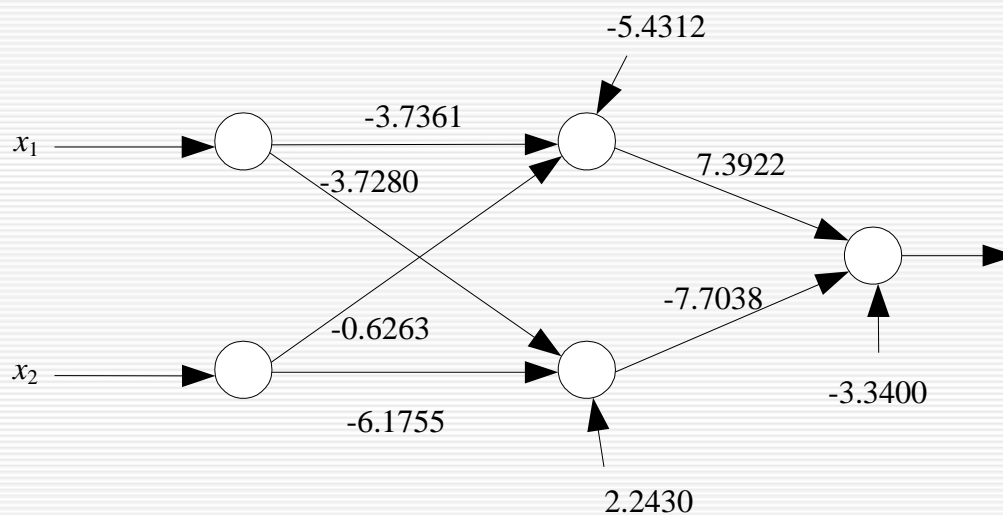
迭代8000次后的BP网络

x_1	x_2	y	实际输出	误差
0	0	0	0.119	0.166
0	0	1	0.727	
1	0	1	0.734	
1	1	0	0.415	

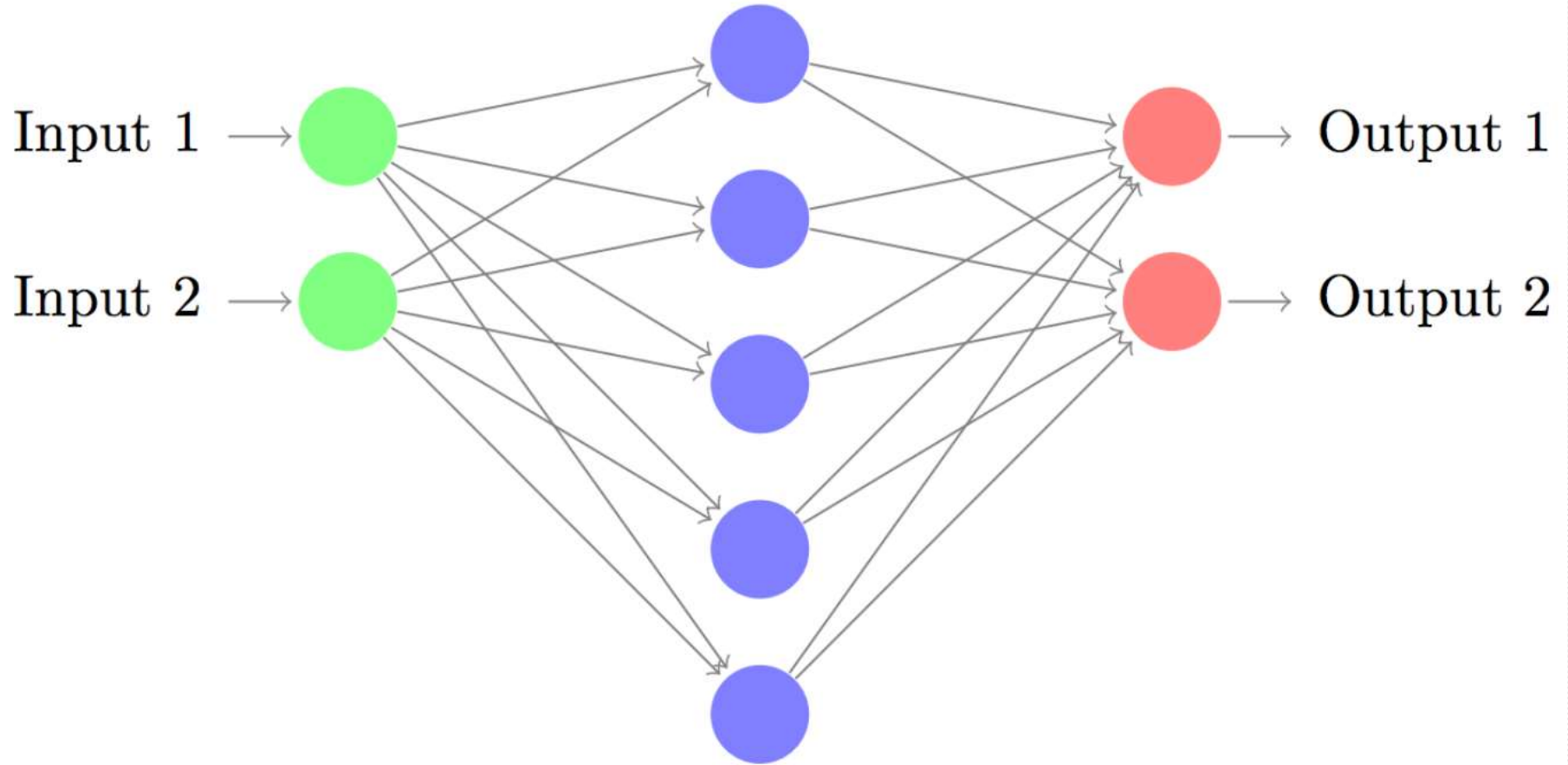


迭代11050次后的BP网络

x_1	x_2	y	实际输出	误差
0	0	0	0.05	0.008
0	1	1	0.941	
1	0	1	0.941	
1	1	0	0.078	



代码分析



代码分析

$$z_1 = xW_1 + b_1$$

$$a_1 = \tanh(z_1)$$

$$z_2 = a_1 W_2 + b_2$$

$$a_2 = \hat{y} = \text{softmax}(z_2)$$

前向传播

损失函数

$$L(y, \hat{y}) = - \frac{1}{N} \sum_{n \in N} \sum_{i \in C} y_{n,i} \log \hat{y}_{n,i}$$

N training examples and C classes

$$(W_1, b_1, W_2, b_2)$$

模型参数

反向传播

$$\delta_3 = y - \hat{y}$$

$$\delta_2 = (1 - \tanh^2 z_1) \circ \delta_3 W_2^T$$

$$\frac{\partial L}{\partial W_2} = a_1^T \delta_3$$

$$\frac{\partial L}{\partial b_2} = \delta_3$$

$$\frac{\partial L}{\partial W_1} = x^T \delta_2$$

$$\frac{\partial L}{\partial b_1} = \delta_2$$

谢谢