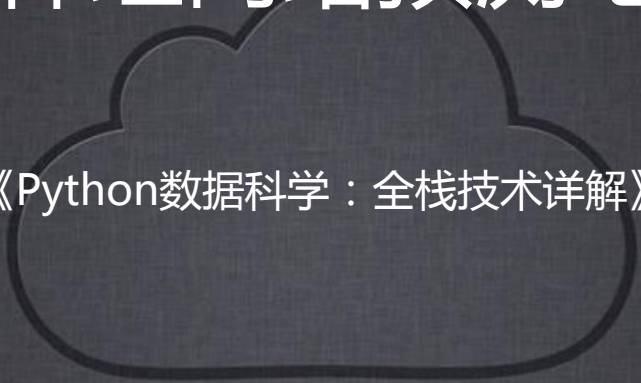


第10章：用神经网络预测电信用户离网



《Python数据科学：全栈技术详解》

讲师：Ben

自我介绍

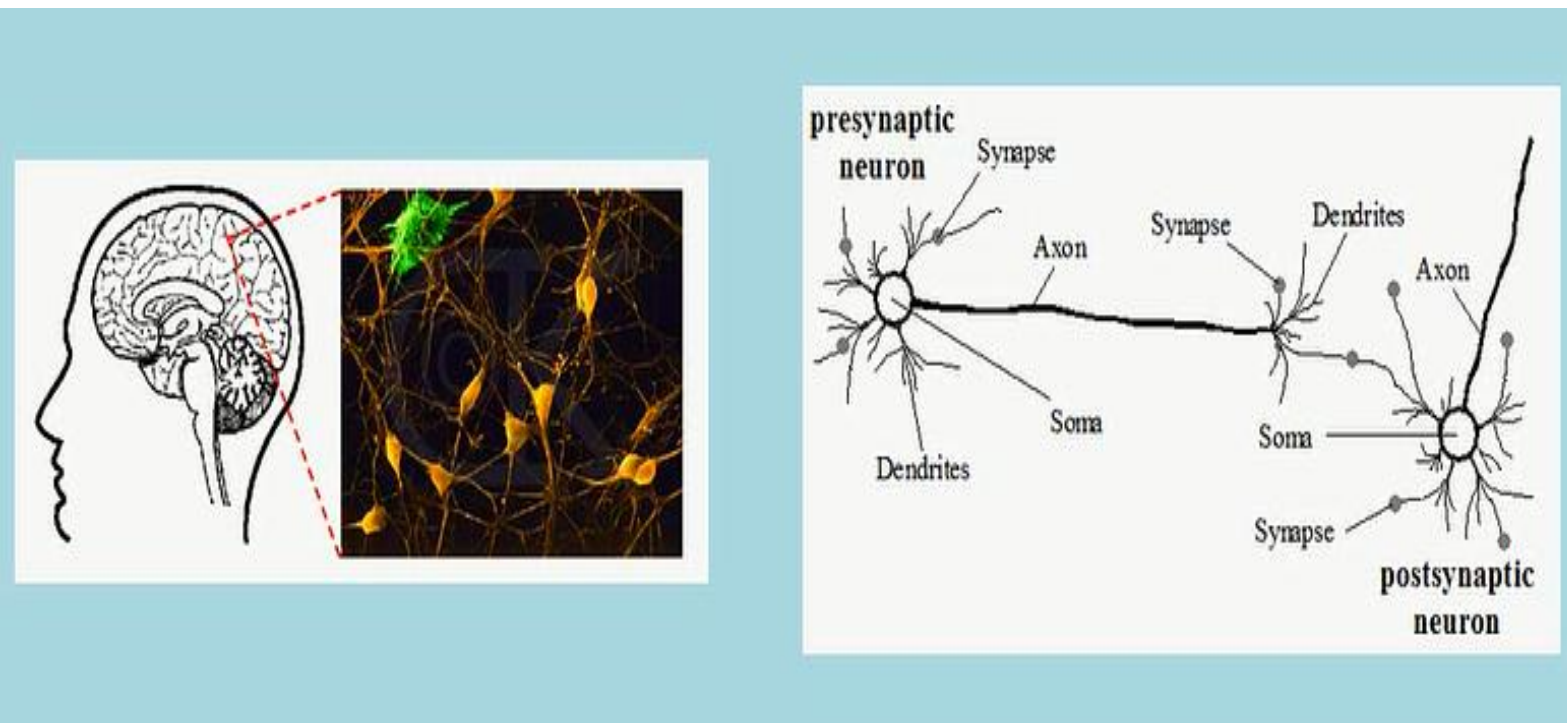
- 天善商业智能和大数据社区 讲师 – Ben
- 天善社区 ID - Ben_Chang
- <https://www.hellobi.com> – 学习过程中有任何相关的问题都可以提到技术社区数据挖掘版块。

主要内容

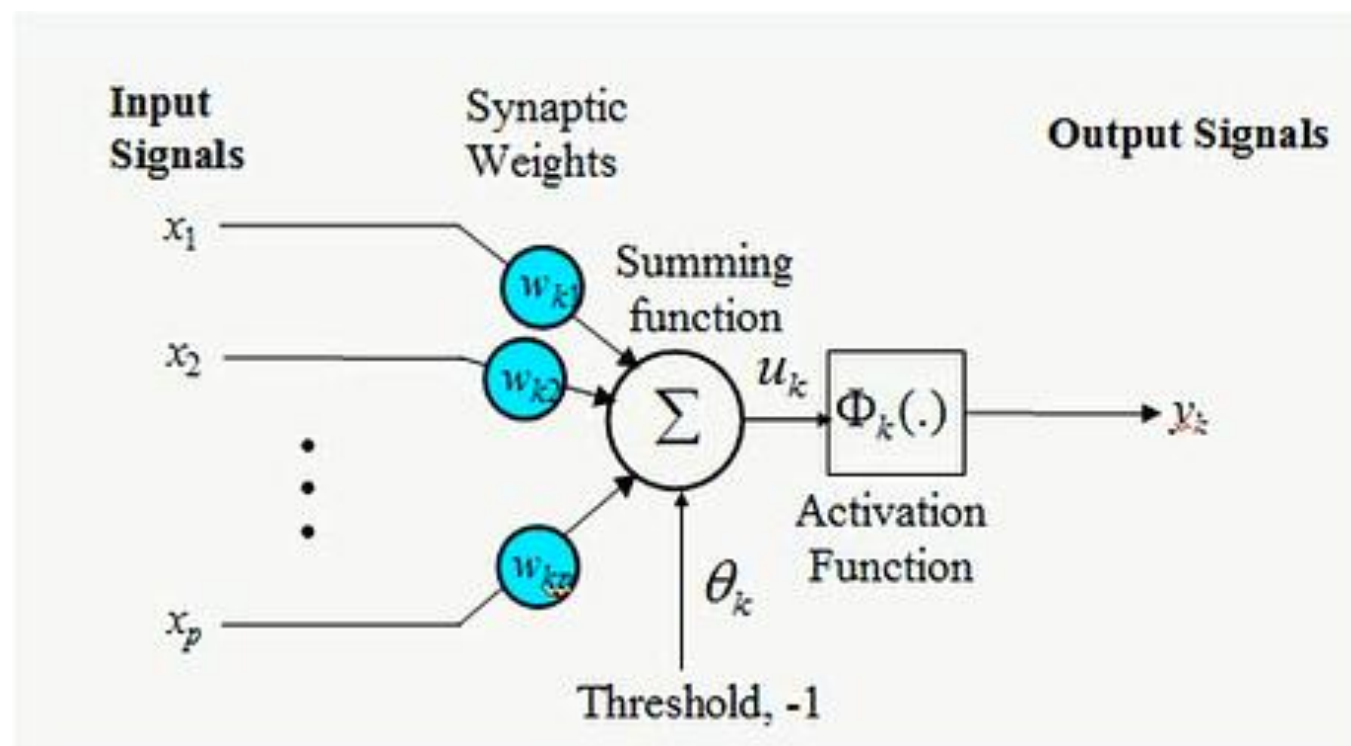
- 了解神经网络的基本概念
- 明确人工神经网络结构
- 掌握BP神经网络学习算法

1 基本概念

什么是神经网络？



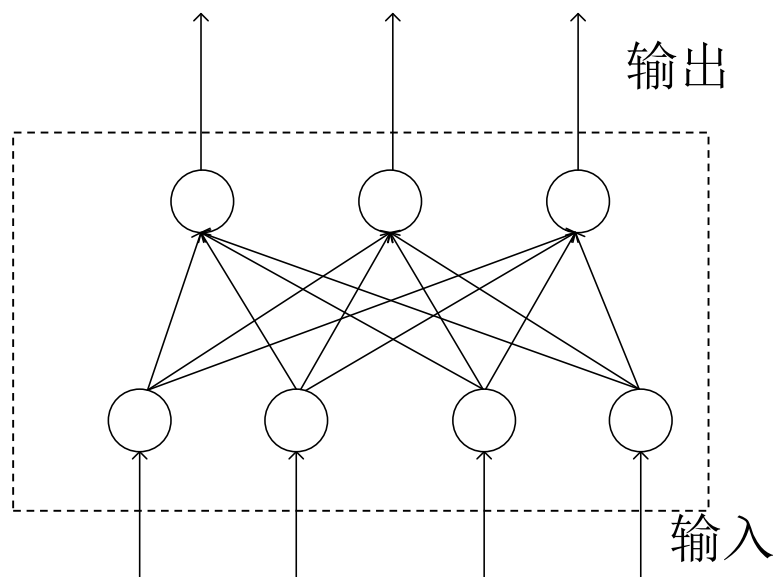
什么是神经网络？



神经网络的正式定义：

- 人工神经网络（Artificial Neural Network, ANN），也称为神经网络（Neural Network, NN），是由大量处理单元广泛互联而成的网络，是对人脑的抽象、简化和模拟，反映人脑的基本特征。人工神经网络的研究是从人脑的生理结构出发来研究人的智能行为，模拟人脑信息处理的功能。
- ❖ 人工神经网络是一种模拟人神经网络行为特征，进行分布式并行信息处理的算法数学模型。这种网络依靠系统的复杂程度，通过调整内部大量节点之间相互连接的关系，从而达到处理信息的目的。

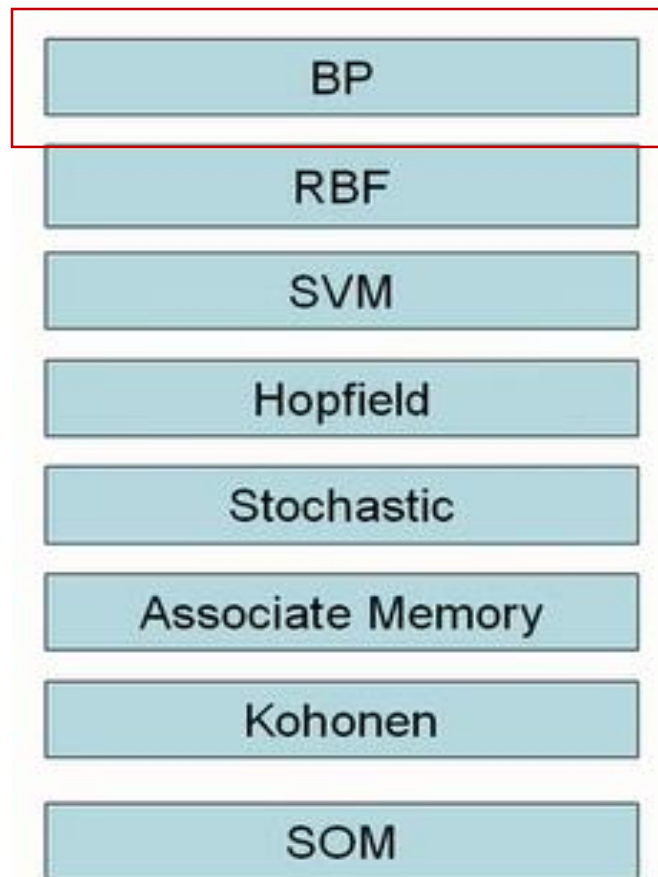
神经网络的一般框架结构



图中每一个圆圈代表一个神经元（也称处理单元或节点node），每个节点之间通过相互连接形成一个网络拓扑。这个拓扑的模式称为**神经网络的互联模式**。

神经网络以外的部分（虚线方框以外的部分）统称为**神经网络的环境**。

几种常见的神经网络



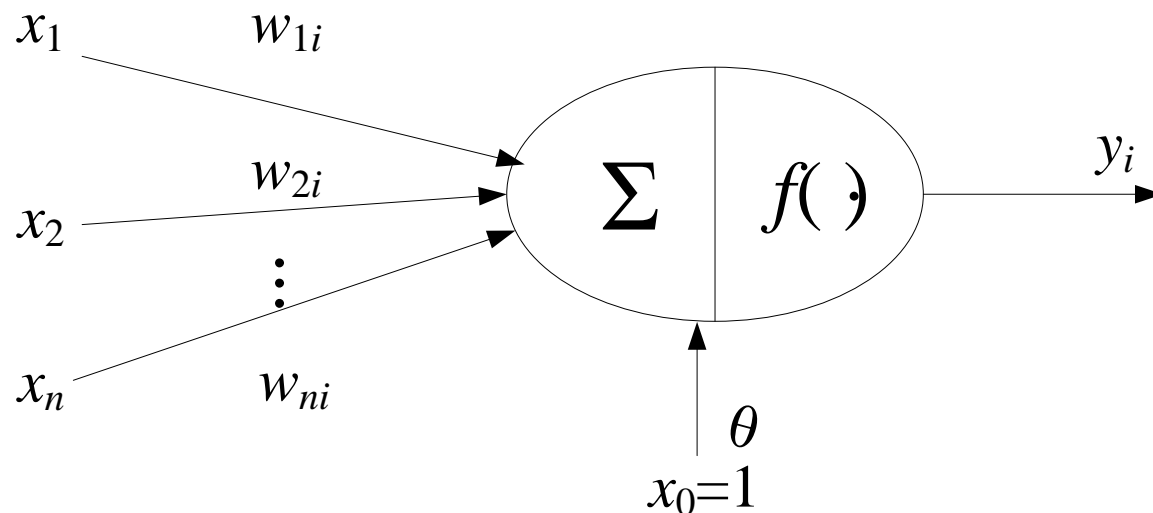
2人工神经网络结构

三、人工神经网络结构

- 人工神经网络
- 人工神经元模型
- 常见响应函数

- 直观理解
 - 神经网络是一个并行和分布式的信息处理网络结构
 - 它一般由大量神经元组成
 - 每个神经元只有一个输出，可以连接到很多其他的神经元
 - 每个神经元输入有多个连接通道，每个连接通道对应于一个连接权系数

- 人工神经元是对生物神经元的一种模拟与简化。它是神经网络的基本处理单元。下图显示了一种简化的人工神经元结构。它是一个多输入、单输出的非线性元件。



人工神经元结构模型

- 人工神经元输入输出关系可描述为

$$I_i = \sum_{j=1}^n w_{ji} x_j - \theta_i \quad (11-1)$$

$$y_i = f(I_i) \quad (11-2)$$

其中 $x_j (j = 1, 2, \dots, n)$ 是从其它神经元传来的输入信号

w_{ji} 表示从神经元 j 到神经元 i 的连接权值

θ_i 为阈值

$f(\cdot)$ 称为激发函数或作用函数

有时为了方便起见，常把 $-\theta_i$ 看成是恒等于1的输入 x_0 的权值，这时式(11-1)的和式可写成

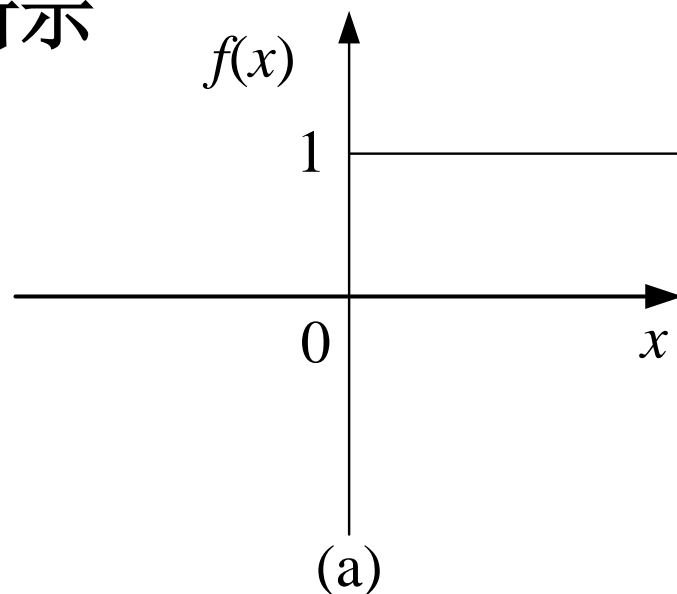
$$I_i = \sum_{j=0}^n w_{ji} x_j \quad x_0 = 1 \quad w_{0i} = -\theta_i \quad (11-3)$$

- 输出激发函数 $f(\cdot)$ 又称为变换函数，它决定神经元(节点)的输出。该输出为1或0，取决于其输入之和大于或小于内部阈值。函数一般具有非线性特征。下面几个图表示了几种常见的激发函数。

(1) 阈值函数(见图(a),(b))

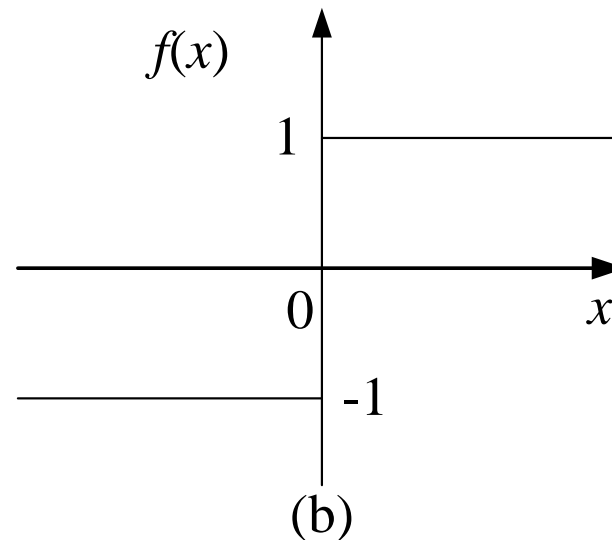
当 y_i 取0或1时， $f(x)$ 为图(a)所示的阶跃函数：

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (11-4)$$



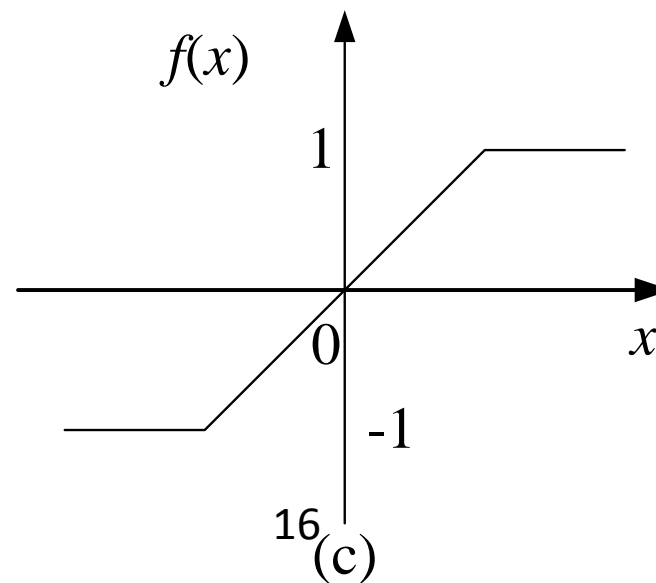
当 y_i 取-1或1时, $f(x)$ 为图 (b)所示的sgn函数:

$$\operatorname{sgn}(x) = f(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases} \quad (11-5)$$



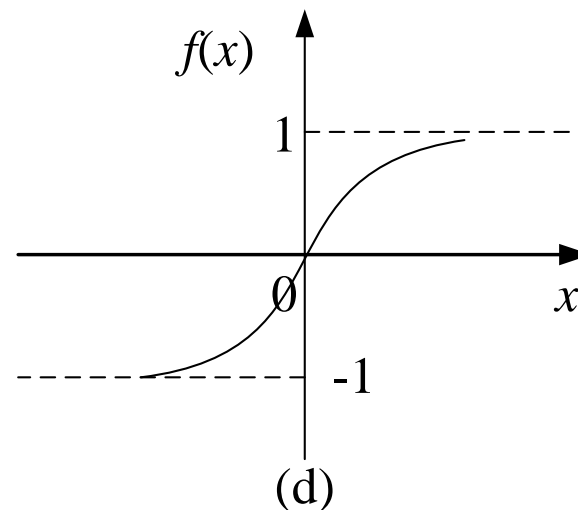
(2) 饱和型函数(见图 (c))

$$f(x) = \begin{cases} 1, & x \geq \frac{1}{k} \\ kx, & -\frac{1}{k} \leq x < \frac{1}{k} \\ -1, & x < -\frac{1}{k} \end{cases} \quad (11-6)$$



(4)双曲函数(见图(d))

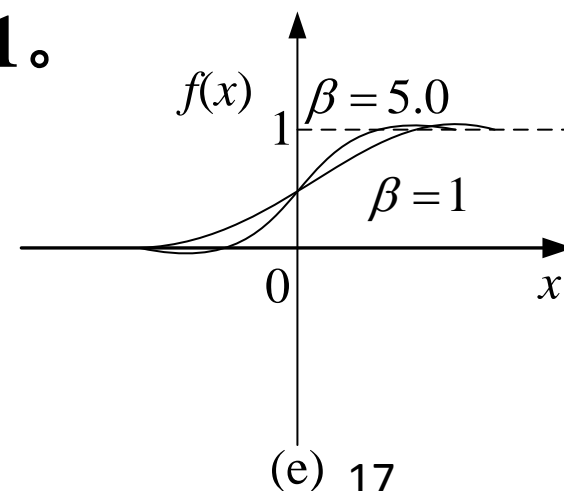
$$f(x) = \tanh(x) \quad (11-7)$$



(5)S型函数(见图(e))

神经元的状态与输入作用之间的关系是在(0, 1)内连续取值的单调可微函数，称为Sigmoid函数，简称S型函数。当 β 趋于无穷时，S型曲线趋于阶跃函数，通常情况下， β 取值为1。

$$f(x) = \frac{1}{1+\exp(-\beta x)}, \beta > 0 \quad (11-8)$$



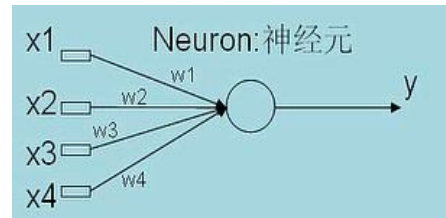
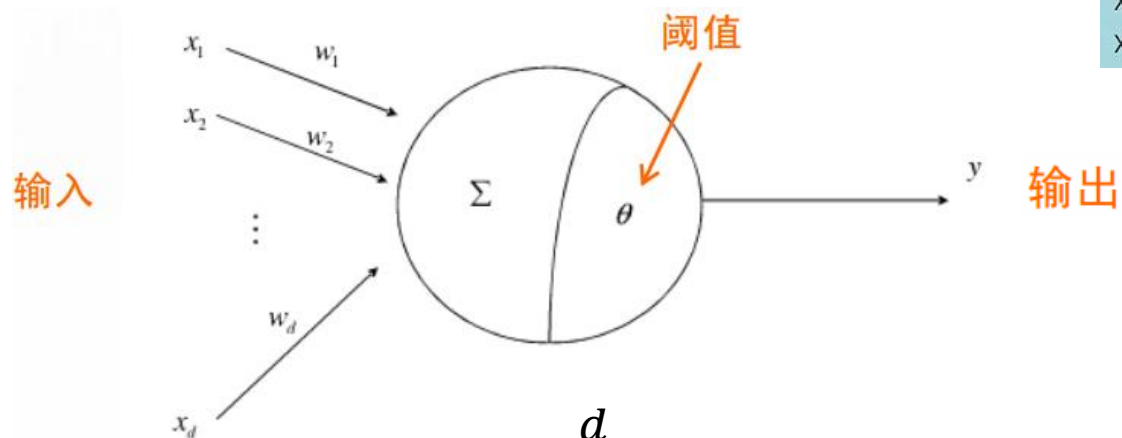
3 感知器与BP网络

前馈神经网络是神经网络中的一种典型的分层结构，信息从输入层进入网络后逐层向前传递至输出层。根据前馈网络中神经元转移函数、隐层数以及权值调整规则的不同，可以形成具有各种功能特点的神经网络。

感知器 | 学习算法

感知器是由美国计算机科学家罗森布拉特（F.Roseblatt）于1957年提出的。感知器可謂是最早的人工神经网络。单层感知器是一个具有一层神经元、采用阈值激活函数的前向网络。通过对网络权值的训练，可以使感知器对一组输入矢量的响应达到元素为0或1的目标输出，从而实现了对输入矢量分类的目的

感知器 | 基本结构



$$y = f\left(\sum_{i=1}^d w_i x_i - \theta\right) \quad (11-10)$$

f为一阶跃函数:

$$y = f(x) = \begin{cases} 1, & \sum_{i=1}^d w_i x_i - \theta \geq 0 \\ -1, & \sum_{i=1}^d w_i x_i - \theta < 0 \end{cases} = \text{sgn}(w_0^T x - \theta) \quad (11-11)$$

感知器原理 | 学习算法

感知器的学习是一种有监督的学习方式，其学习规则称之为delta规则。
若以 t 表示目标输出(Y)， a 表示实际输出(\hat{Y})，则

$$e = t - a \quad (11-12)$$

网络训练的目的就是要使 $a \rightarrow t$

当 $e=0$ 时，得到最优的网络权值和阈值；

当 $e>0$ 时，实际输出小于目标输出，应加大网络权值和阈值；

当 $e<0$ 时，实际输出大于目标输出，就减小网络权值和阈值；

其权值阈值学习算法为：

$$\begin{aligned} W(k+1) &= W(k) + ep^T \\ b(k+1) &= b(k) + e \end{aligned} \quad (11-13)$$

式中：

e ——误差向量， $e=t-a$

W ——权值向量

b ——阈值向量

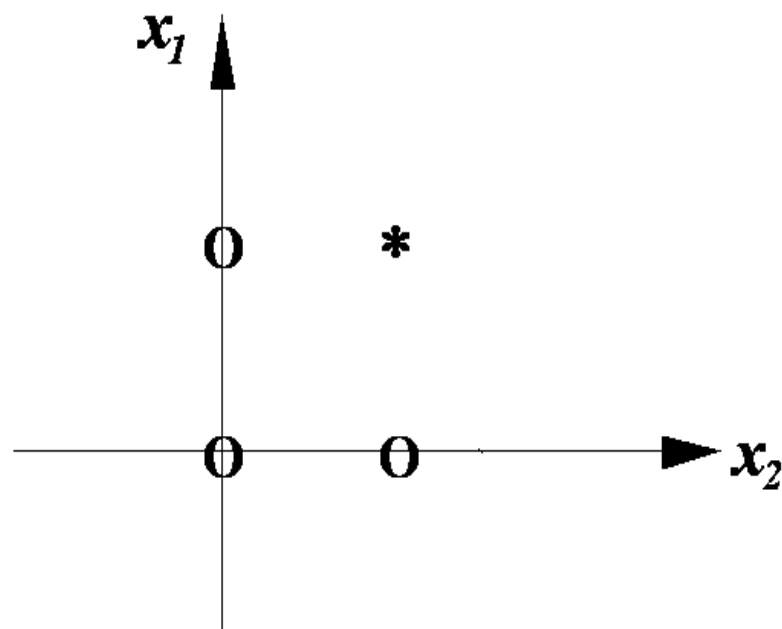
p ——输入向量

k ——表示第 k 步学习过程

例 用感知器实现逻辑“与”功能

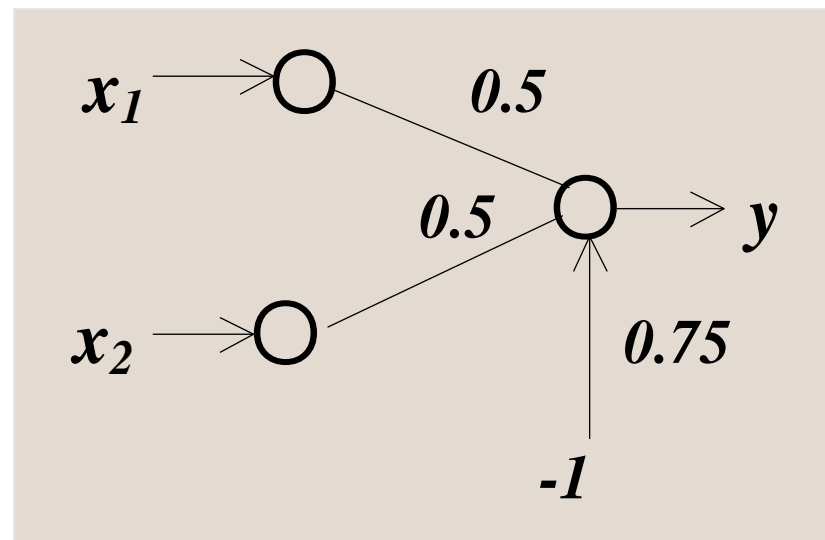
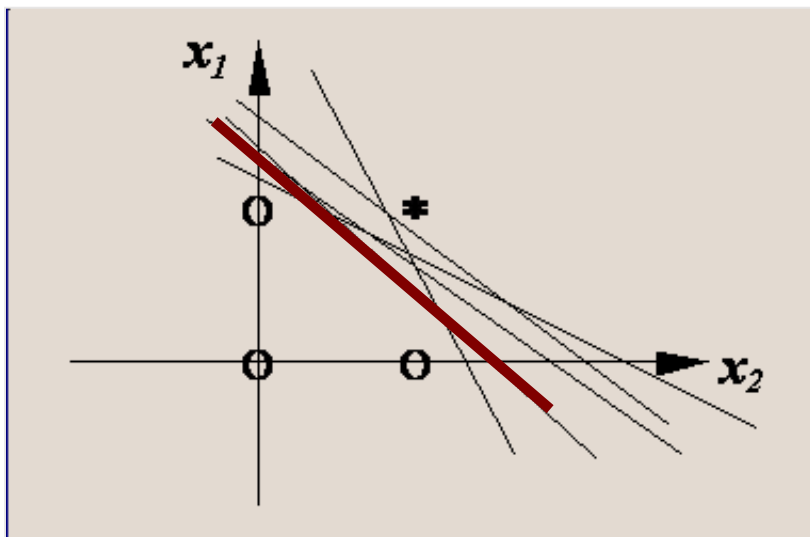
逻辑“与”真值表

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

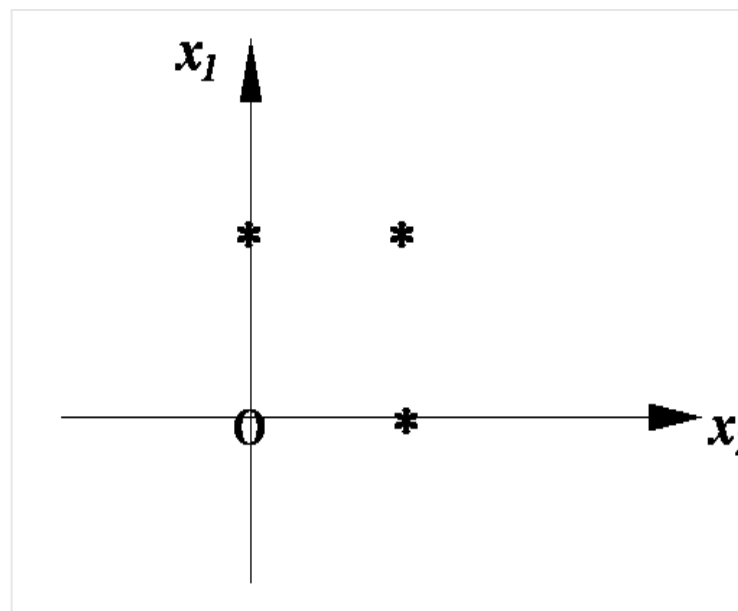


$$\begin{aligned}w_1x_1 + w_2x_2 - T &= 0 \\0.5x_1 + 0.5x_2 - 0.75 &= 0\end{aligned}\tag{11-14}$$

例 用感知器实现逻辑“或”功能

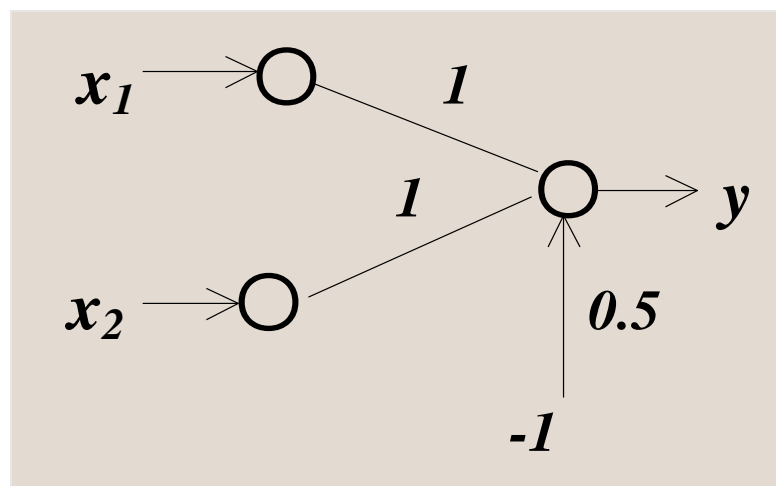
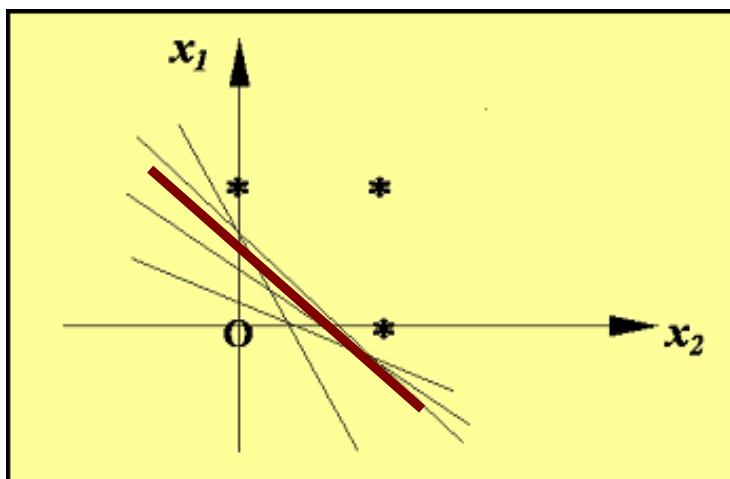
逻辑“或”真值表

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

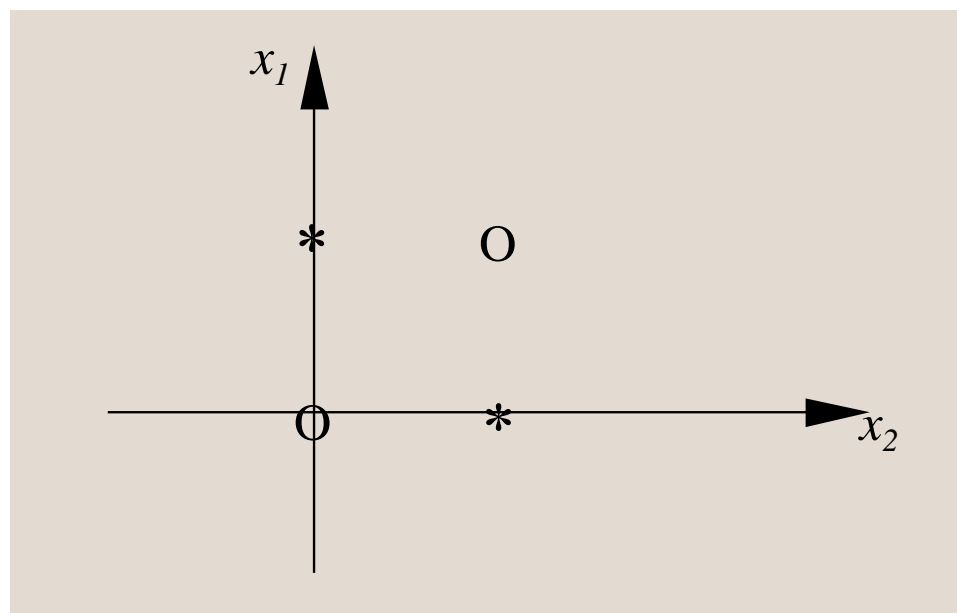


$$\begin{aligned}w_1x_1 + w_2x_2 - T &= 0 \\x_1 + x_2 - 0.5 &= 0\end{aligned}\tag{11-15}$$

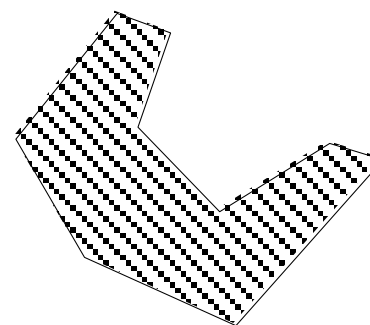
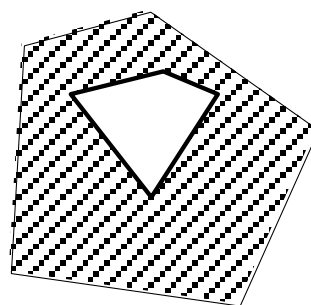
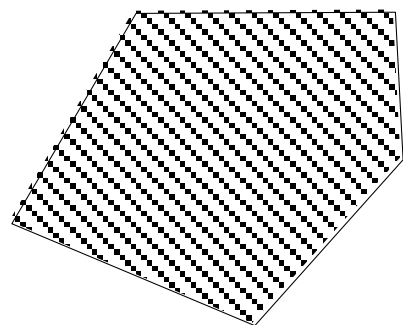
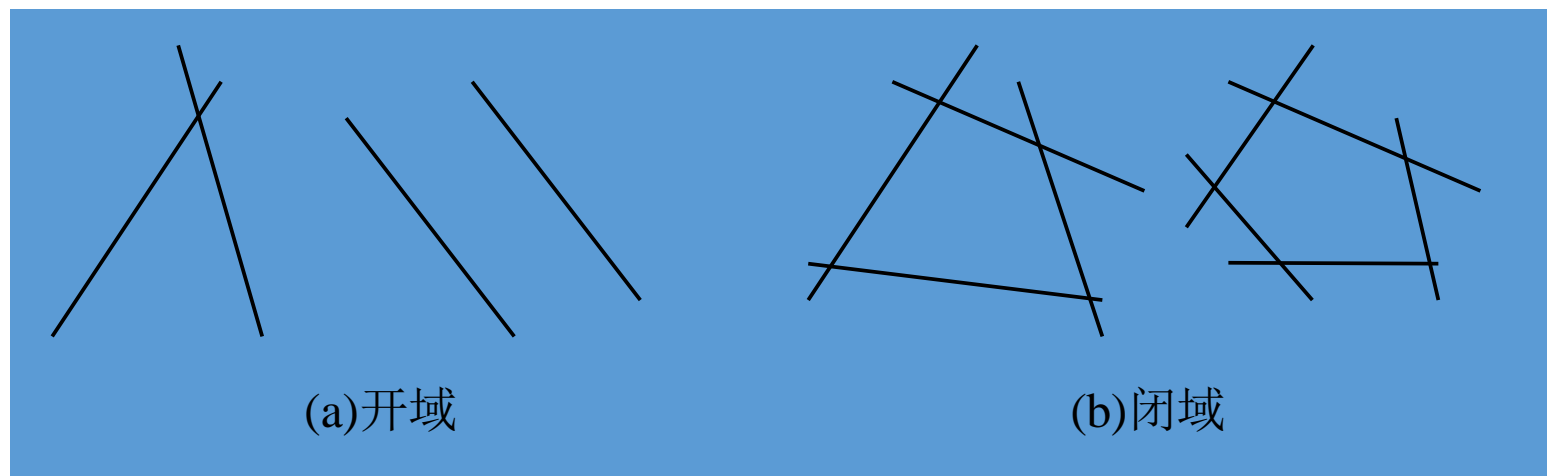
问题：能否用感知器实现“异或”功能？

“异或”的真值表

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


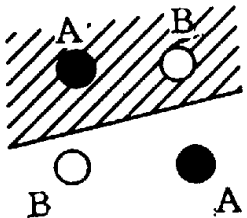
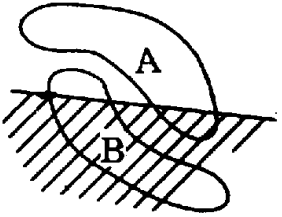

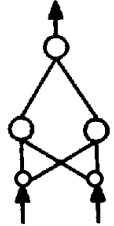
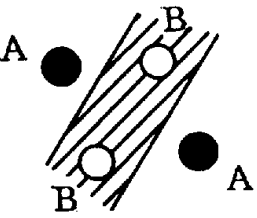
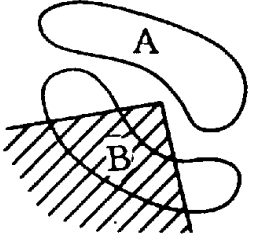

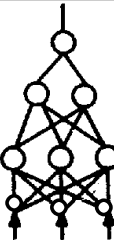
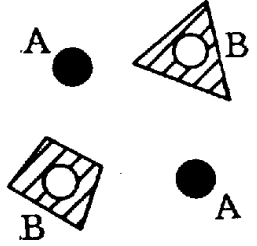
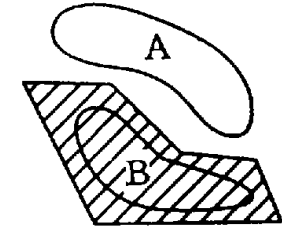
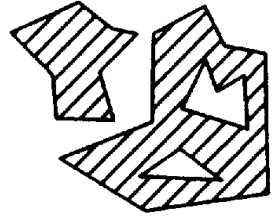


多层感知器



多层感知器

$$\Delta W_j(t) = \eta [d_j - o_j(t)] X \quad (11-16)$$

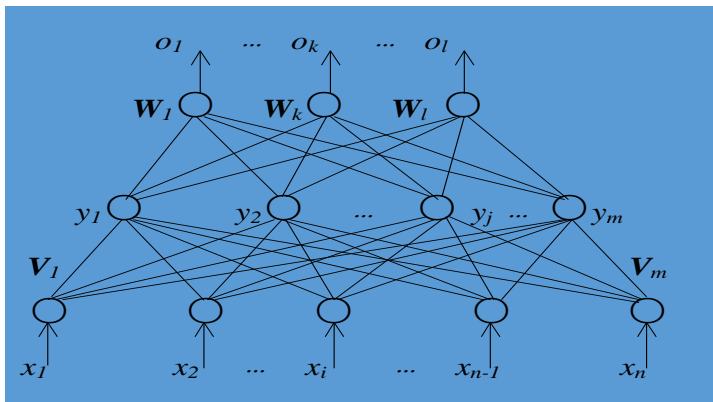
感知器结构	异或问题	复杂问题	判决域形状	判决域
无隐层 				半平面
单隐层 				凸 域
双隐层 				任意复杂 形状域

具有不同隐层数的感知器的分类能力对比

BP神经网络原理 | 基本概念

BP (Back Propagation) 网络是1986年由Rumelhart和McClelland为首的科学家小组提出，是一种按误差逆传播算法训练的多层前馈网络。是目前应用广泛的神经网络模型之一。 BP网络能学习和存贮大量的输入-输出模式映射关系，而无需事前提示描述这种映射关系的数学方程，它是一种监督学习算法

BP算法由数据流的前向计算（正向传播）和误差信号的反向传播两个过程构成



- 将误差分摊给各层的所有单元 - 各层单元的误差信号，用于修正各单元权值
- 网络输出的误差减少到可接受的程度
- 进行到预先设定的学习次数为止

BP网络的传递函数有多种。Log-sigmoid型函数的输入值可取任意值，输出值在0和1之间； tan-sigmoid型传递函数tansig的输入值可取任意值，输出值在-1和+1之间；线性传递函数purelin的输入与输出值可取任意值。

BP网络通常有一个或多个隐层，该层中的神经元均采用sigmoid型传递函数，输出层的神经元则采用线性传递函数，整个网络的输出可以取任意值

BP神经网络原理 | 学习算法（概述）

输入层和输出层可以根据实际问题确定，但权值和阈值初始化，却没有严格的理论基础。而BP神经网络可以通过误差最速下降法，从而调整权值与阈值：

第一步，网络初始化。给各连接权值分别赋一个区间 $(-1, 1)$ 内的随机数，设定误差函数 e ，给定计算精度值 ε 和最大学习次数 M

第二步，随机选择第 k 个输入样本及对应期望输出

第三步，计算隐含层各神经元的输入和输出

第四步，利用网络期望输出和实际输出，计算误差函数对输出层的个神经元的偏导数 $\delta_o(k)$

第五步，利用隐含层到输出层的连接权值、输出层的 $\delta_o(k)$ 和隐含层的输出计算误差函数对隐含层个神经元的偏导数 $\delta_h(k)$

第六步，利用输出层各神经元的 $\delta_o(k)$ 和隐含层各神经元的输出来修正连接权值 $w_{ho}(k)$

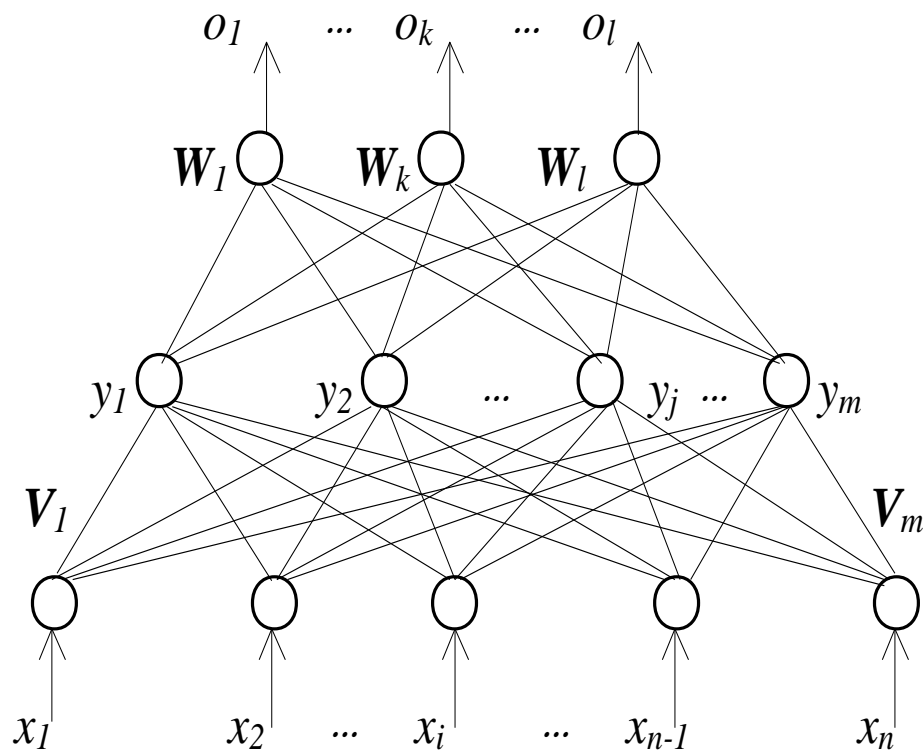
第七步，利用隐含层各神经元的 $\delta_h(k)$ 和输入层各神经元的输入修正连接权

第八步，计算全局误差

$$E = \frac{1}{2m} \sum_{k=1}^m \sum_{o=1}^q (d_o(k) - y_o(k))^2 \quad (11-17)$$

第九步，判断网络误差是否满足要求。当误差达到预设精度或学习次数大于设定的最大次数，则结束算法。否则，选取下一个学习样本及对应的期望输出，返回第三步，进入下一轮学习

BP神经网络原理 | 学习算法（详述）



• 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：

$$W=(W_1,W_2,\dots,W_k,\dots,W_l)$$

BP神经网络原理 | 学习算法（详述）

对于输出层：

$$\begin{aligned} o_k &= f(net_k) \quad k = 1, 2, \dots, l \\ net_k &= \sum_{j=0}^m w_{jk} y_j \quad k = 1, 2, \dots, l \end{aligned} \quad (11-18)$$

对于隐层：

$$\begin{aligned} y_j &= f(net_j) \quad j = 1, 2, \dots, m \\ net_j &= \sum_{i=0}^n v_{ij} x_i \quad j = 1, 2, \dots, m \end{aligned} \quad (11-19)$$

BP神经网络原理 | 学习算法（详述）

单极性Sigmoid函数：

$$f(x) = \frac{1}{1 + e^{-x}} \quad (11-20)$$

双极性Sigmoid函数：

$$f(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \quad (11-21)$$

BP神经网络原理 | 学习算法（详述）

输出误差 E 定义：

$$E = \frac{1}{2}(\mathbf{d} - \mathbf{o})^2 = \frac{1}{2} \sum_{k=1}^l (d_k - o_k)^2 \quad (11-22)$$

将以上误差定义式展开至隐层：

$$E = \frac{1}{2} \sum_{k=1}^l [d_k - f(\text{net}_k)]^2 = \frac{1}{2} \sum_{k=1}^l \left[d_k - f \left(\sum_{j=0}^m w_{jk} y_j \right) \right]^2 \quad (11-23)$$

BP神经网络原理 | 学习算法（详述）

进一步展开至输入层：

$$\begin{aligned} E &= \frac{1}{2} \sum_{k=1}^l \left\{ d_k - f \left[\sum_{j=0}^m w_{jk} f(\text{net}_j) \right] \right\}^2 \\ &= \frac{1}{2} \sum_{k=1}^l \left\{ d_k - f \left[\sum_{j=0}^m w_{jk} f \left(\sum_{i=0}^n v_{ij} x_i \right) \right] \right\}^2 \end{aligned} \quad (11-24)$$

BP神经网络原理 | 学习算法（详述）

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} \quad j = 0, 1, 2, \dots, m; k = 1, 2, \dots, l \quad (11-25)$$

$$\Delta v_{ij} = -\eta \frac{\partial E}{\partial v_{ij}} \quad i = 0, 1, 2, \dots, n; j = 1, 2, \dots, m \quad (11-26)$$

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 $j=0, 1, 2, \dots, m; k=1, 2, \dots, l$

对隐层有 $i=0, 1, 2, \dots, n; j=1, 2, \dots, m$

BP神经网络原理 | 学习算法（详述）

对于输出层，式(11-25)可写为

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} = -\eta \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial w_{jk}} \quad (11-27)$$

对隐层，式(11-26)可写为

$$\Delta v_{ij} = -\eta \frac{\partial E}{\partial v_{ij}} = -\eta \frac{\partial E}{\partial net_j} \frac{\partial net_j}{\partial v_{ij}} \quad (11-28)$$

对输出层和隐层各定义一个误差信号，令

$$\delta_k^o = -\frac{\partial E}{\partial net_k} \quad \delta_j^y = -\frac{\partial E}{\partial net_j} \quad (11-29)$$

BP神经网络原理 | 学习算法（详述）

综合应用式(11-29)和(11-27)，可将式（11-25）的权值调整式改写为

$$\Delta w_{jk} = \eta \delta_k^o y_j \quad (11-30)$$

综合应用式(11-29)和(11-28)，可将式（11-26）的权值调整式改写为

$$\Delta v_{ij} = \eta \delta_j^y x_i \quad (11-31)$$

可以看出，只要计算出式中的误差信号 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 。

BP神经网络原理 | 学习算法（详述）

对于输出层， δ^o 可展开为

$$\delta_k^o = -\frac{\partial E}{\partial net_k} = -\frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial net_k} = -\frac{\partial E}{\partial o_k} f'(net_k) \quad (11-32)$$

对于隐层， δ^y 可展开为

$$\delta_j^y = -\frac{\partial E}{\partial net_j} = -\frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial net_j} = -\frac{\partial E}{\partial y_j} f'(net_j) \quad (11-33)$$

下面求式(11-22)中网络误差对各层输出的偏导。

BP神经网络原理 | 学习算法（详述）

对于输出层，利用式(11-22)：
$$E = \frac{1}{2} \sum_{k=1}^l (d_k - o_k)^2$$

可得：
$$\frac{\partial E}{\partial o_k} = -(d_k - o_k) \quad (11-34)$$

对于隐层，利用式(11-23)：
$$E = \frac{1}{2} \sum_{k=1}^l \left[d_k - f \left(\sum_{j=0}^m w_{jk} y_j \right) \right]^2$$

可得：
$$\frac{\partial E}{\partial y_j} = - \sum_{k=1}^l (d_k - o_k) f'(net_k) w_{jk} \quad (11-35)$$

BP神经网络原理 | 学习算法（详述）

将以上结果代入式(11-33)，并应用式(11-35)

得到：

$$\delta_k^o = (d_k - o_k) o_k (1 - o_k) \quad (11-36)$$

$$\begin{aligned} \delta_j^y &= \left[\sum_{k=1}^l (d_k - o_k) f'(net_k) w_{jk} \right] f'(net_j) \\ &= \left(\sum_{k=1}^l \delta_k^o w_{jk} \right) y_j (1 - y_j) \end{aligned} \quad (11-37)$$

至此两个误差信号的推导已完成。

BP神经网络原理 | 学习算法（详述）

将式(11-35)代回到式(11-32)，得到三层前馈网的BP学习算法权值调整计算公式为：

$$\left\{ \begin{array}{l} \Delta w_{jk} = \eta \delta_k^o y_j = \eta (d_k - o_k) o_k (1 - o_k) y_j \quad (11-38) \\ \Delta v_{ij} = \eta \delta_j^y x_i = \eta \left(\sum_{k=1}^l \delta_k^o w_{jk} \right) y_j (1 - y_j) x_i \quad (11-39) \end{array} \right.$$

BP神经网络原理 | 学习算法（详述）

$$o_k = f(net_k)$$

$$y_j = f(net_j)$$

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

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

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

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

$$= \frac{1}{2} \sum_{k=1}^l [d_k - f(net_k)]^2$$

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

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

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

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

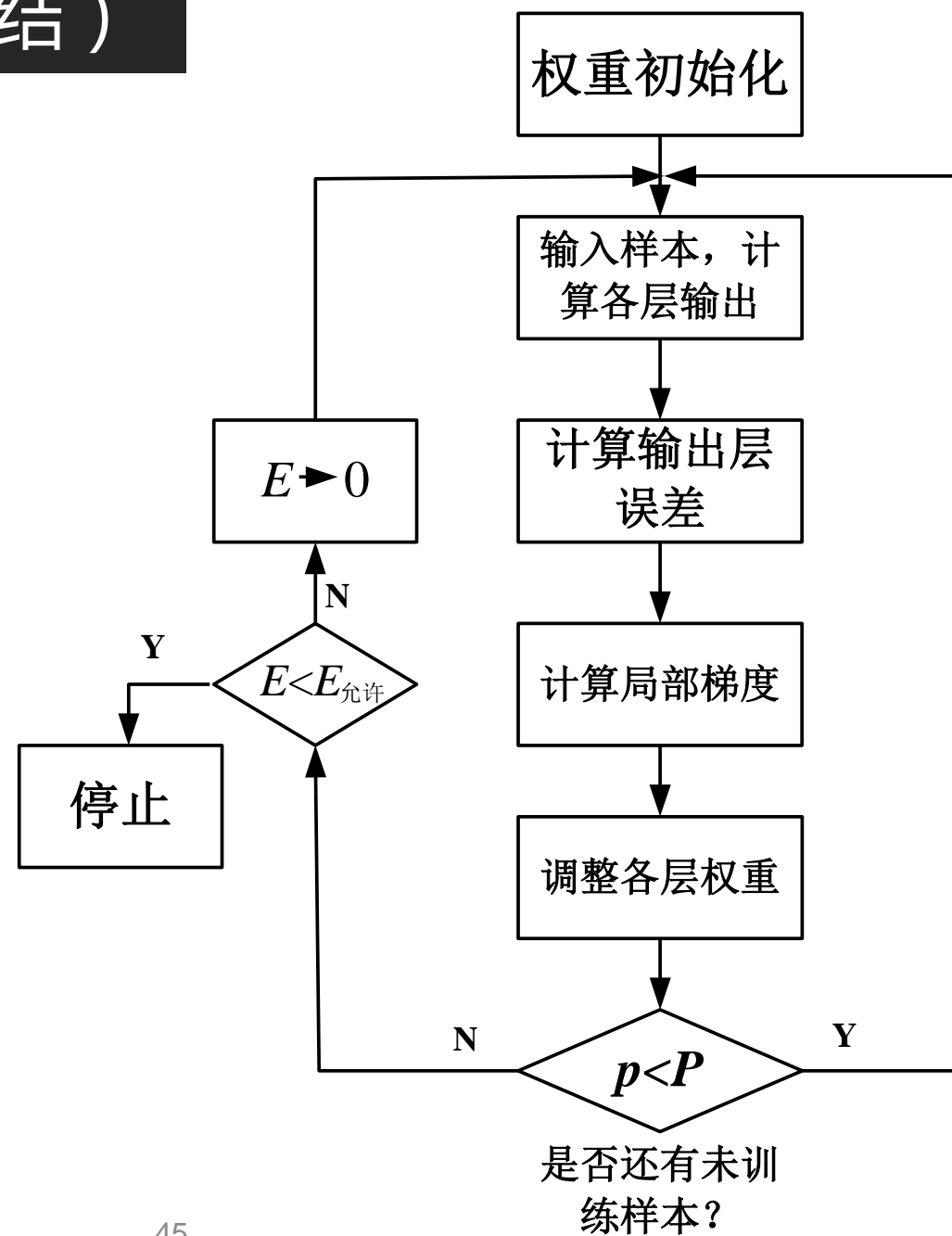
$$\delta_k^o = -\frac{\partial E}{\partial net_k}$$

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

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

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

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



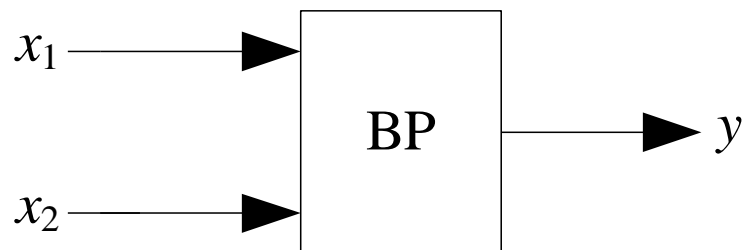
在使用BP算法时，应注意的几个问题是：

- 学习开始时，各隐层连接权系数的初值应以设置较小的随机数较为适宜。
- 采用S型激发函数时，由于输出层各神经元的输出只能趋于1或0，不能达到1或0。在设置各训练样本时，期望的输出分量不能设置为1或0，以设置为0.9或0.1较为适宜。
- 在学习开始阶段，选较大的值可以加快学习速率。学习接近优化区时，学习速率必须相当小，否则权值将产生振荡而不收敛。

BP神经网络原理 | 学习举例

样本：输入和期望输出

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



激发函数 $f(x) = \frac{1}{1 + e^{-x}}$

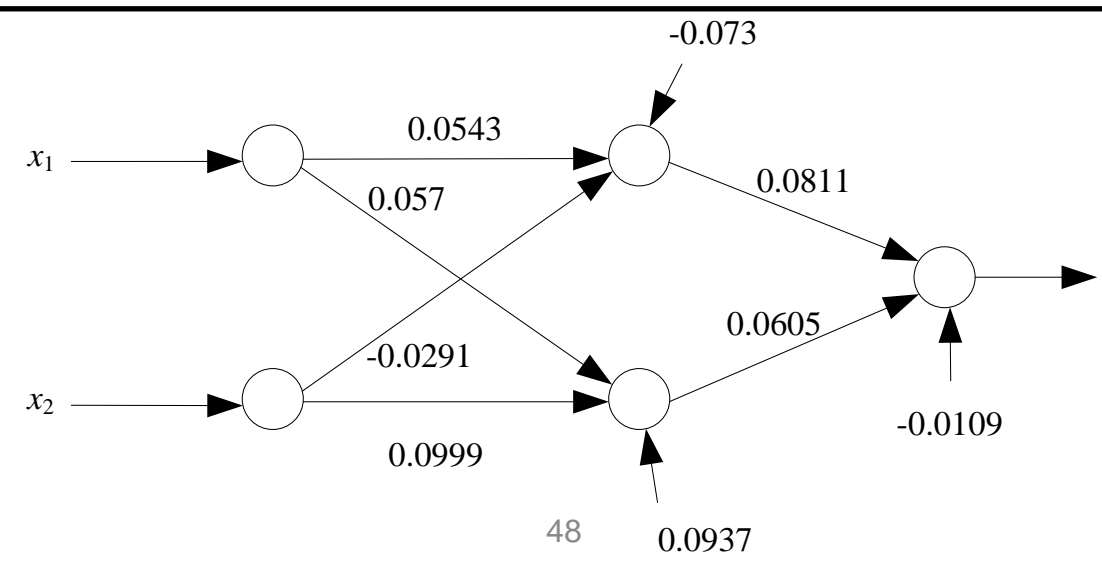
学习速率 $\eta = 0.6$

网络包含一个隐层，设输入层与隐层的权值 w_{jk} ，隐层与输出层的权值 w_{ij} 初始值 $w_{jk}, w_{ij} \in [-0.1, 0.1]$

BP神经网络原理 | 学习举例

迭代1次后的BP网络

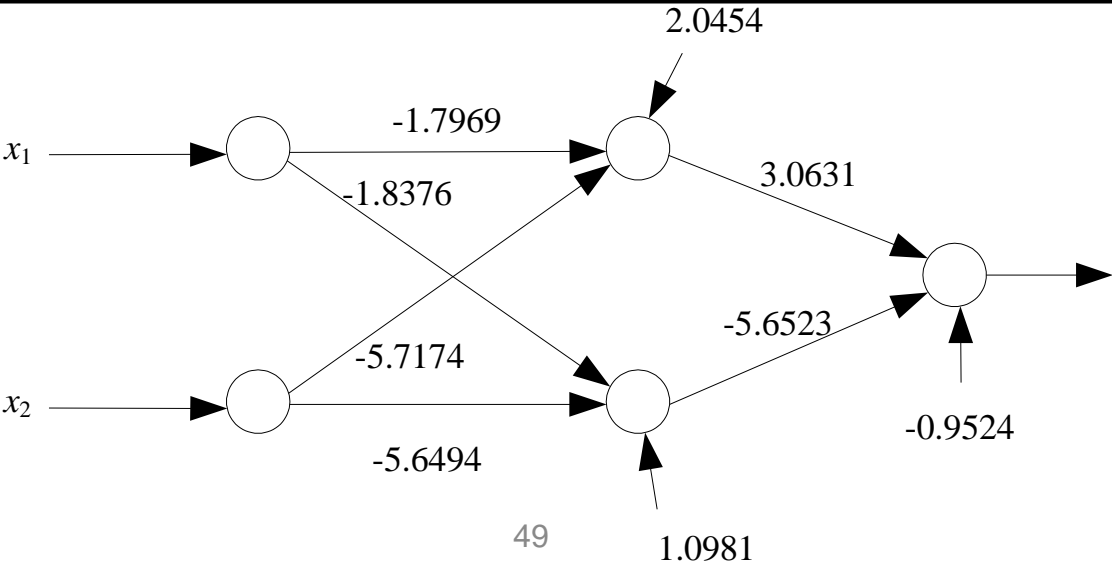
x_1	x_2	y	实际输出	误差
0	0	0	0.5	0.5
0	1	1	0.5	
1	0	1	0.5	
1	1	0	0.5	



BP神经网络原理 | 学习举例

迭代8000次后的BP网络

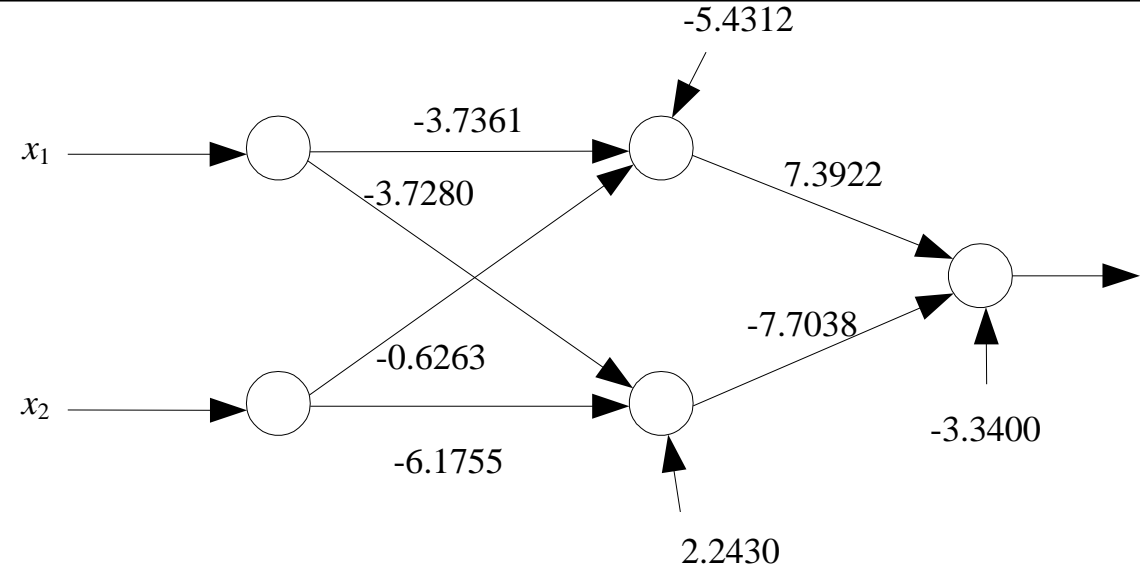
x_1	x_2	y	实际输出	误差
0	0	0	0.119	0.166
0	0	1	0.727	
1	0	1	0.734	
1	1	0	0.415	



BP神经网络原理 | 学习举例

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	



BP神经网络原理 | 评述

➤ 非线性映射能力

能学习和存贮大量输入-输出模式映射关系，而无需事先了解描述这种关系的数学方程。只要能提供足够多的样本模式对供网络进行学习训练，它便能完成由 n 维输入空间到 m 维输出空间的非线性映射

➤ 泛化能力

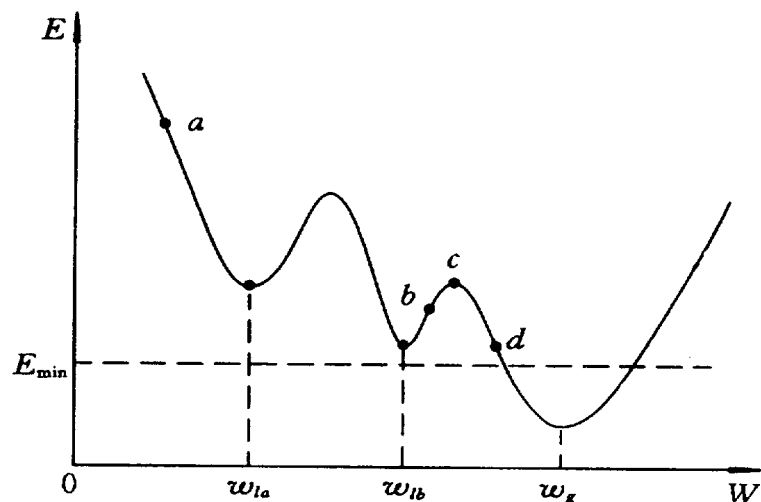
当向网络输入训练时未曾见过的非样本数据时，网络也能完成由输入空间向输出空间的正确映射。这种能力称为泛化能力。这部分内容在模型复杂度部分讨论。

➤ 容错能力

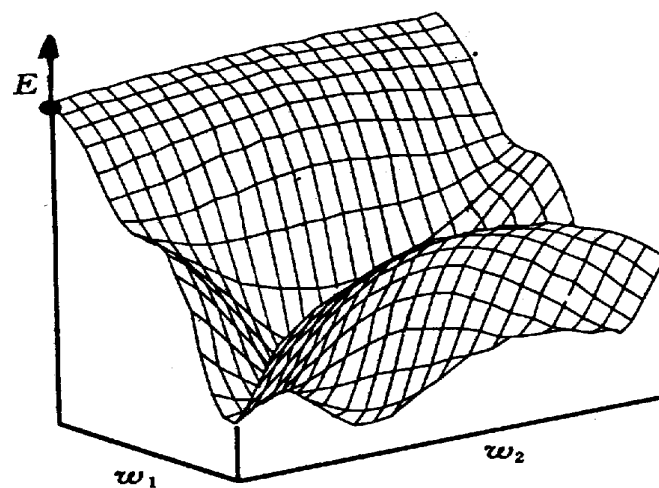
输入样本中带有较大的误差甚至个别错误对网络的输入输出规律影响很大。

BP神经网络原理 | 局限性

误差 E 是 n_w+1 维空间中一个形状极为复杂的曲面，该曲面上的每个点的“高度”对应于一个误差值，每个点的坐标向量对应着 n_w 个权值，因此称这样的空间为误差的权空间。



局限性： 单权值



双权值

- 1、多数极小点都是局部极小，只能计算局部最优。
- 2、误差平面有时平缓，有时陡峭，导致采取固定学习效率时，要么学习缓慢，要么跳跃性太大。

BP神经网络原理 | 改进

●思路

- 误差曲面的形状 - - 固有的
- 算法的作用是什么？
 - 调整权值，找到最优点
- 那么如何更好地调整权值？
 - 利用算法使得权值在更新的过程中，‘走’合适的路径，比如跳出平坦区来提高收敛速度，跳出局部最小点等等
- 如何操作？
 - 需要在进入平坦区或局部最小点时进行一些判断，通过改变某些参数来使得权值的调整更为合理。

BP神经网络原理 |改进

- 引入动量项

标准BP算法实质上是一种简单的最速下降静态寻优算法，在修正 $w(t)$ 时，只按 t 时刻的负梯度方式进行修正，而没有考虑以前积累的经验，即以前时刻的梯度方向，从而常使学习过程发生振荡，收敛缓慢。为此提出如下改进算法：

$$\Delta w(t+1) = \eta \delta X + \alpha \Delta w(t) \quad (11-40)$$

增加动量项即从前一次权值调整量中提取出一部分迭代到本次权值调整量中。该方法所加入的动量项实质上相当于阻尼项，它减小了学习过程的振荡趋势，改善了收敛性，这是目前应用比较广泛的种改进算法。

BP神经网络原理 |改进

- 变步长法

一阶梯度法寻优收敛较慢的一个重要原因是 η (学习率)不好选择。 η 选的太小，收敛太慢，若 η 选的太大，则有可能修正过头，导致振荡甚至发散。下面给出的变步长法即是针对这个问题而提出的。

$$\begin{aligned}w(t+1) &= w(t) + \eta(t)D(t) \\ \eta(t) &= 2^\lambda \eta(t-1) \\ \lambda &= \text{sgn}[D(t)D(t-1)]\end{aligned}\tag{11-41}$$

这里 w 表示某个连接权系数。

BP神经网络原理 |改进

●变步长法

上面的算法说明，当连续两次迭代其梯度方法相同时，表明下降太慢，这时可使步长加倍；当连续两次迭代其梯度方向相反时，表明下降过头，这时可使步长减半。当需要引入动量项时，上述算法的第二项可修改为

$$w(t+1) = w(t) + \alpha(t)[(1-\eta)D(t) + \eta D(t-1)] \quad (11-42)$$

在使用该算法时，由于步长在迭代过程中自适应调整，因此对于不同的连接权系数实际采用了不同的学习率，也就是说误差代价函数 E 在超曲面上在不同地方按照各自比较合理的步长向极小点逼近。

- Ian , Goodfellow 《深度学习 deep learning 》 , 人民邮电出版社
- 韩力群 , 《 人工神经网络理论、设计及应用》 , 化学工业出版社
- 陈明 , 《MATLAB神经网络原理与实例精解》 , 清华大学出版社
- 哈根 , 戴葵等 《 神经网络设计》 , 机械工业出版社
- Simon Haykin , 《 神经网络与机器学习》 , 机械工业出版社

秦路主讲

七周成为数据分析师

七周为期，Get一条数据分析师职业黄金通道！



Python

数据分析与挖掘

集Python爬虫、数据采集、数据处理、数据分析与数据挖掘于一体，打造Python全栈工程师

主讲老师：韦玮

VIP会员群+在线答疑+录播复习+1年反复观看

案例为师，实战为王

开启Python机器学习之路

科学规划全套课程体系，从入门到进阶，从理论到技巧，嵌入丰富课程案例讲解，逐步推进

讲师：唐宇迪 深度学习领域多年一线实践研究专家

独一无二的数据库建模指南系列教程升级版

- 从企业视角进行数据规划以及数据库模型的搭建
- 高质量的数据库模型和技巧，以及丰富的例子
- 数据库架构理论和实践要领

资深讲师：BAO胖子 15年+BI从业经验
涉足电力、快消品、医药、信息服务行业的BI老兵

业务知识一站通

技术+业务，挣钱有门路！

讲师：陈文



自己动手 丰衣足食

Python3网络爬虫实战案例

一循序渐进，案例为王，诠释全面，思路制胜一

讲师：崔庆才 北航硕士，百万级热度爬文博主



讲师 丘祐玮

人人都爱数据科学家

Python数据科学精华实战课程

数据分析报告制作

秘籍升级版

讲师：陈丹奕 知乎大神，前百度资深数据分析师

先机致胜 破冰AI

深度学习模型/框架与实战

讲师：唐宇迪 同济大学硕士
深度学习领域多年一线实践研究专家



BI、商业智能
数据挖掘 大数据
数据分析师
R语言 Python
机器学习
深度学习
人工智能
Hive Hadoop
Tableau
BIEE ETL
数据科学家
PowerBI