数据挖掘原理

主讲教师：李志勇

数据科学系
数字农业工程技术研究中心
移动：13882213811 电邮: lzy@sicau.edu.cn

第八章：神经网络

——巨量并行，智慧无限

主讲教师：李志勇

主要介绍内容

- 8.1 人工神经网络概述
- 8.2 神经元模型
- 8.3 简单感知机
- 8.4 梯度下降法
- 8.5 多层感知器

8.1 人工神经网络概述

假设一个系统有三个输入和一个输出。通过实验得到了8条数据，如表所示。第9行数据不是通过实验获得，而是事先假设了一组输入。怎样才能推断这一行数据的输入对应的输出数据呢？

序号	x_1	x_2	x_3	y
1	2.5	3.16	-9.75	15.84
2	3.4	2.9	10.15	7.31
3	5.65	3.27	8.43	24.3
4	4.63	8.18	1.67	514.29
5	-7.37	-8.51	2.71	333.37
6	-5.6	3.27	8.83	281.67
7	10.17	15.2	0.03	1895.62
8	19.1	-4.8	7.87	27.62
9	7.32	-5.69	2.93	

发现规律
进行预测

8.1 人工神经网络概述

$$y=\frac{\left(x_1+x_2^2-x_1x_3\right)^2+\left(x_1x_2+x_2x_3\right)^2}{\sqrt{x_1^2+x_2^2+x_3^2}+\left(x_1-x_2+x_3\right)^2}$$

- 对于由第1式给出的系统，可以产生一系列的数据

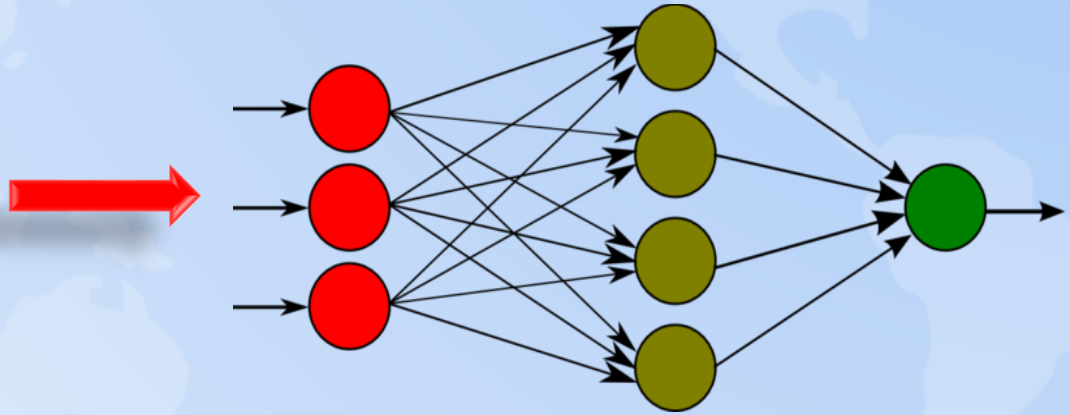
- 假设第1式我们并不知道，但是我们可以用第2式给出的多元线性回归找到近似规律，但这种方式并没有很好的精度

$$y=C_0+C_1x_1+C_2x_2+C_3x_3$$

- 人工神经网络可以逼近任意非线性规律

8.1 人工神经网络概述

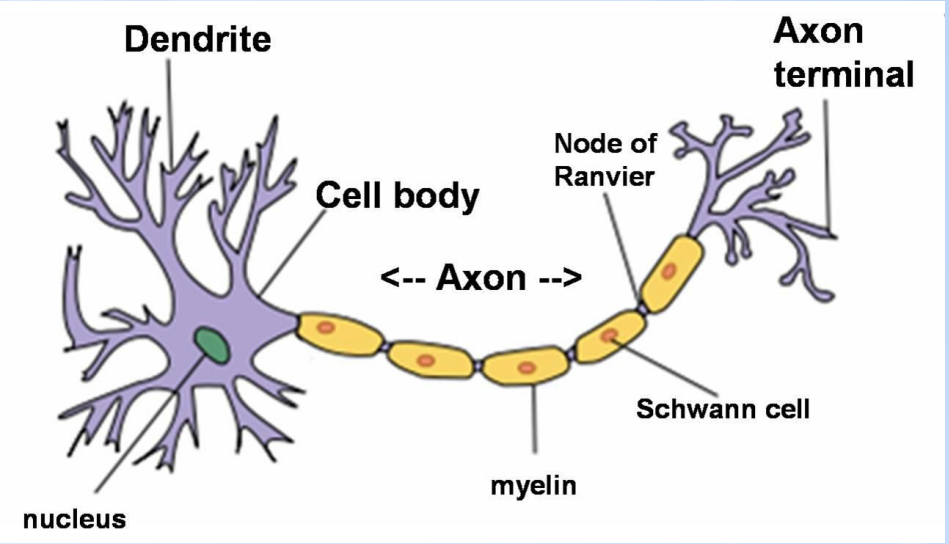
神经网络是一种模拟人脑的神经网络以期能够实现类人工智能的机器学习技术。人脑中的神经网络是一个非常复杂的组织。成人的大脑中估计有1000亿个神经元之多。那么机器学习中的神经网络是如何实现这种模拟的，并且达到一个惊人的良好效果的？

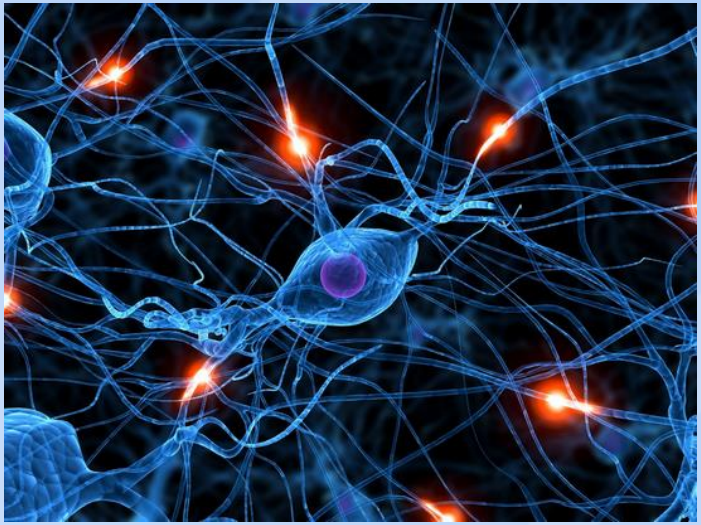


8.1 人工神经网络概述

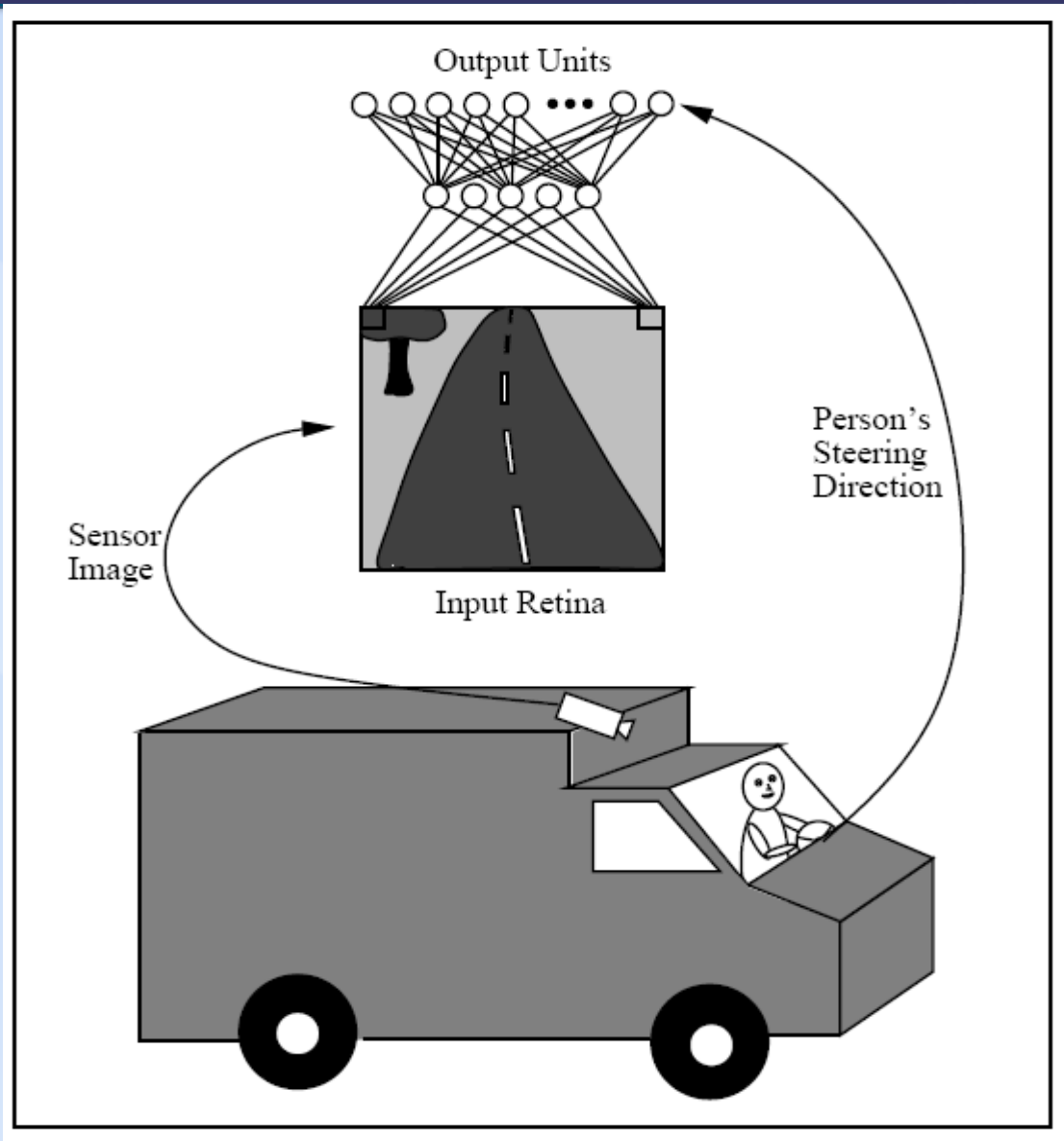
对于神经元的研究由来已久，1904年生物学家就已经知晓了神经元的组成结构。

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”。



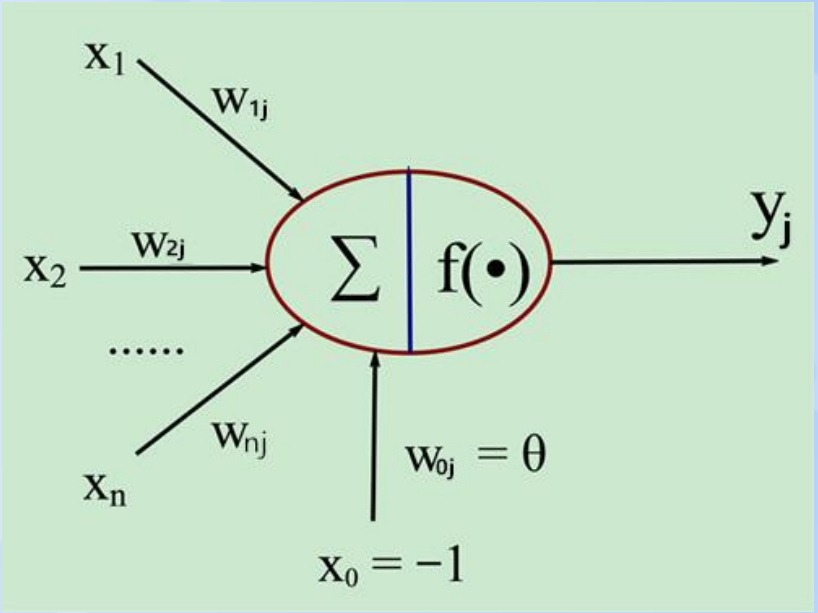


8.1 人工神经网络概述


Output Units
Person's
Steering
Direction
Sensor
Image
Input Retina

8.2 神经元模型

1943年，心理学家McCulloch和数学家Pitts参考了生物神经元的结构，发表了抽象的神经元模型MP。



- 其中$x_1,x_2,\ldots,x_n$是输入，y是输出
- ∑为内部状态的反馈信息和，θ为阈值
- 特性函数f(·)又包括分段线性特性函数、阈值特性函数、S型逻辑特性函数

8.2 神经元模型

1. 分段线性特性函数

$$x_i' = F(X_i) = kX_i$$

其中k为常数。

2. 阈值特性函数

$$x_i' = F(X_i) = \begin{cases} 1 & X_i \geq \theta_i \\ 0 & X_i < \theta_i \end{cases}$$

这是最早提出的一种离散型的二值函数。

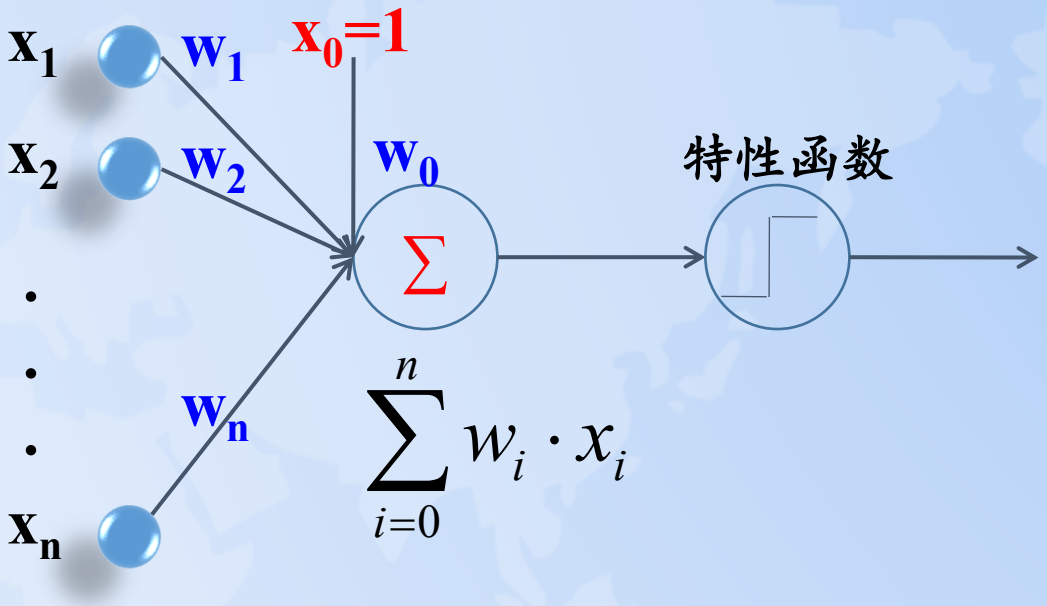
3. S型逻辑特性函数

$$x_i' = F(X_i) = \left(1 + e^{-X_i}\right)^{-1}$$

这类特性函数常用来表示输入和输出的S型曲线关系，反映神经元的“压缩”和“饱和”特性，即把神经元定义为具有非线性增益特性的电子系统。

8.3 简单感知机

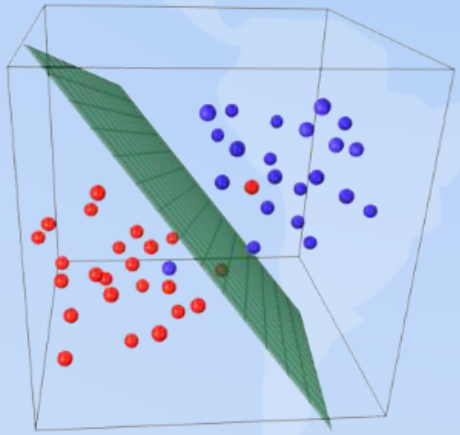
简单感知器模型实际上仍然是MP模型的结构，由美国学者F.Rosenblatt于1957年提出。



$$\sum_{i=0}^{n} w_i \cdot x_i$$

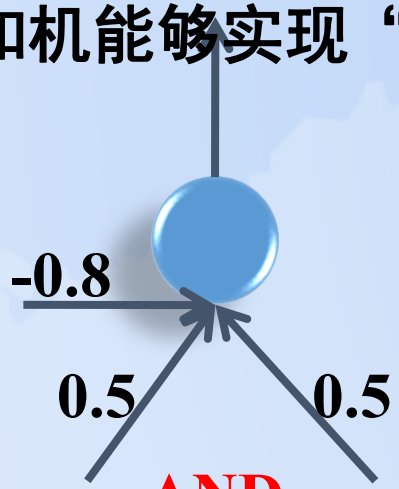
$$o = \begin{cases} 1, & if \sum_{i=0}^{n} w_i \cdot x_i > 0 \\ 0, & otherwise \end{cases}$$

$$o(x_1,\ldots,x_n) = \begin{cases} 1, & if\ w_0 + w_1 \cdot x_1 + \cdots + w_n \cdot x_n > 0 \\ 0, & otherwise \end{cases}$$



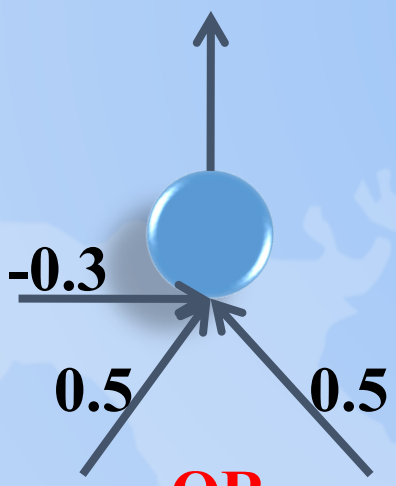
8.3 简单感知机

感知机能够实现“逻辑运算”功能



AND

Input		∑	Output
0	0	-0.8	0
0	1	-0.3	0
1	0	-0.3	0
1	1	0.2	1



OR

Input		∑	Output
0	0	-0.3	0
0	1	0.2	1
1	0	0.2	1
1	1	0.7	1

AND:两位同时为“1”，结果才为“1”，否则为0

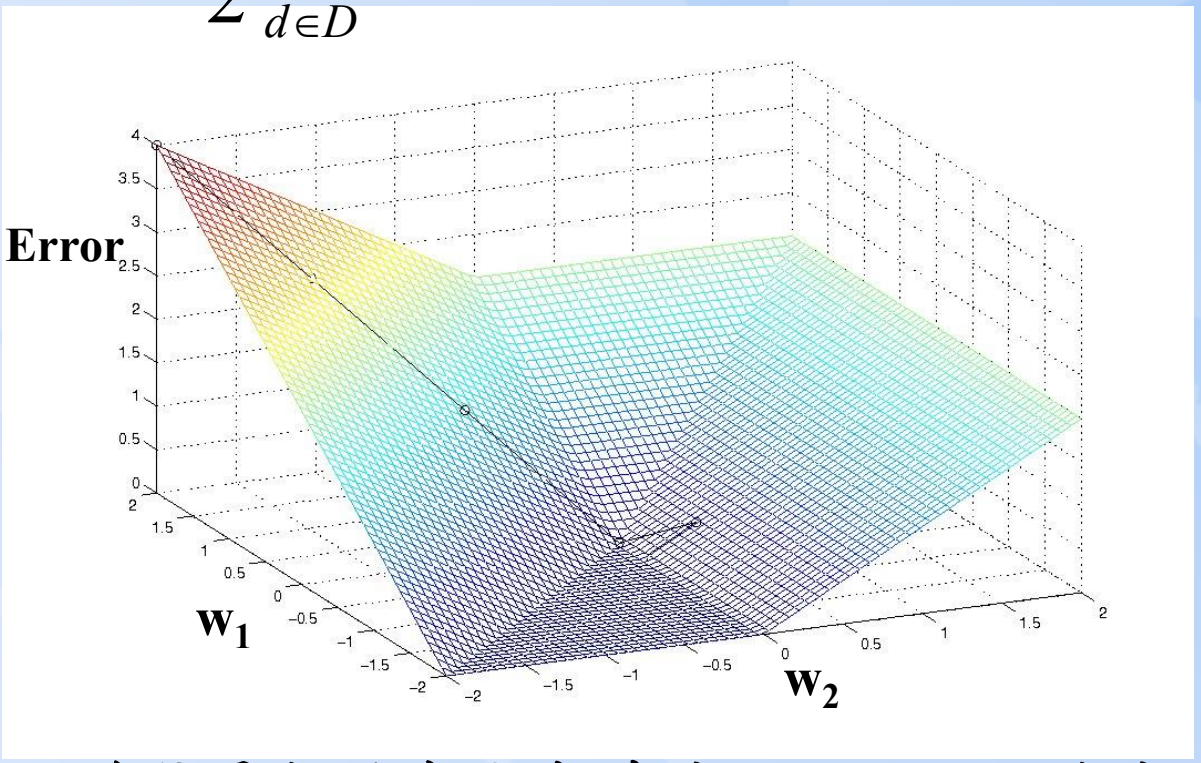
OR:参加运算的两个对象只要有一个为1，其值为1。

8.4 梯度下降法

实际的输出数据和计算得到的输出数据之间有一个误差，这个误差反映了该神经网络对实际系统的逼近能力。

$$E(\vec{w}) \equiv \frac{1}{2}\sum_{d \in D}(t_d - o_d)^2$$

Batch Learning



让参数w沿着偏导数的负方向前进就可以使得总体误差下降

8.4 梯度下降法

计算误差对参数w的导数

$$\nabla E(\vec{w}) \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n} \right]$$

$$w_i \leftarrow w_i + \Delta w_i \quad where \quad \Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Learning Rate

8.4 梯度下降法

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{d\in D}(t_d - o_d)^2$$

$$= \frac{1}{2}\sum_{d\in D}\frac{\partial}{\partial w_i}(t_d - o_d)^2$$

$$= \frac{1}{2}\sum_{d\in D}2(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - o_d)$$

$$= \sum_{d\in D}(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - w\cdot x_d)$$

$$= \sum_{d\in D}(t_d - o_d)(-x_{id})$$

$$\Delta w_i = \eta\sum_{d\in D}(t_d - o_d)x_{id}$$

$$o(x) = w\cdot x$$

8.4 梯度下降法

梯度下降法的实现步骤：

- 将每个w_i初始化为一些小的随机值，
- 直到满足终止条件，否则
  - 将每个Δw_i初始化为零，
  - 对于训练中的每个<x，t>，将实例x输入到神经元并计算输出o，
  - 对于每个Δw_i，计算 $\Delta w_i \leftarrow \Delta w_i + \eta(t-o)x_i$
  - 对于每个 w_i，计算 $w_i \leftarrow w_i + \Delta w_i$

8.4 梯度下降法

$$w_i \leftarrow w_i + \Delta w_i \qquad where \quad \Delta w_i = \eta(t-o)x_i$$

例如, 如果 x_i=0.8, η=0.1, t=1 and o=0

$\Delta w_i = \eta(t\text{-}o)x_i = 0.1 \times (1\text{-}0) \times 0.8 = 0.08$

$w_{i+1} = w_i + \Delta w_i$

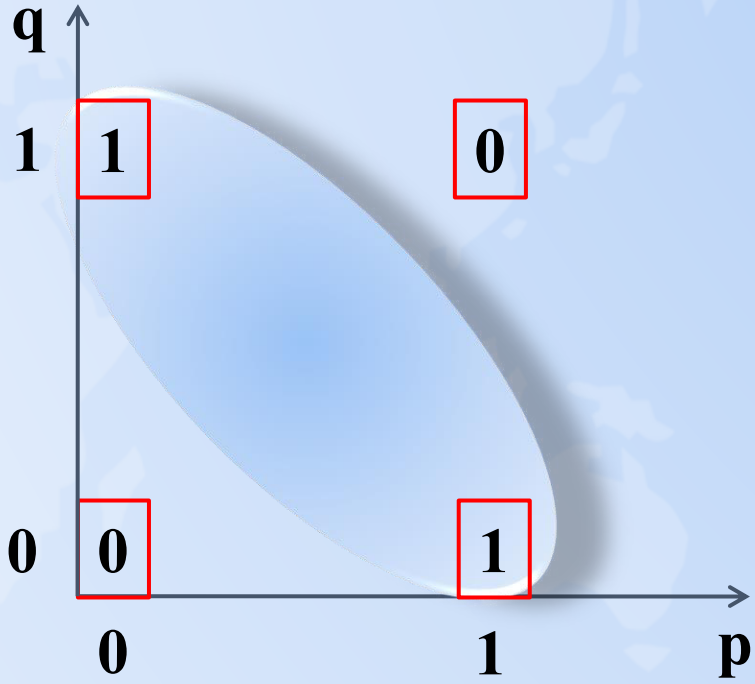
8.4梯度下降法

Input			Target	Initial Weights			Output					Error	Correction	Final Weights		
							Individual			Sum	Final Output					
x_0	x_1	x_2	t	w_0	w_1	w_2	C_0	C_1	C_2	S	o	E	R	w_0	w_1	w_2
							$x_0 \cdot w_0$	$x_1 \cdot w_1$	$x_2 \cdot w_2$	$C_0+C_1+C_2$		t-o	LR x E			
1	0	0	1	0	0	0	0	0	0	0	0	1	+0.1	0.1	0	0
1	0	1	1	0.1	0	0	0.1	0	0	0.1	0	1	+0.1	0.2	0	0.1
1	1	0	1	0.2	0	0.1	0.2	0	0	0.2	0	1	+0.1	0.3	0.1	0.1
1	1	1	0	0.3	0.1	0.1	0.3	0.1	0.1	0.5	0	0	0	0.3	0.1	0.1
1	0	0	1	0.3	0.1	0.1	0.3	0	0	0.3	0	1	+0.1	0.4	0.1	0.1
1	0	1	1	0.4	0.1	0.1	0.4	0	0.1	0.5	0	1	+0.1	0.5	0.1	0.2
1	1	0	1	0.5	0.1	0.2	0.5	0.1	0	0.6	1	0	0	0.5	0.1	0.2
1	1	1	0	0.5	0.1	0.2	0.5	0.1	0.2	0.8	1	-1	-0.1	0.4	0	0.1
1	0	0	1	0.4	0	0.1	0.4	0	0	0.4	0	1	+0.1	0.5	0	0.1
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	1	0	1	0.8	-.2	-.1	0.8	-.2	0	0.6	1	0	0	0.8	-.2	-.1

threshold=0.5 Learning Rate=0.1

8.5 多层感知器

对于一个两输入的简单感知器，每个输入取值为0和1，所有输入样本有四个，记为(p，q)：(0，0)，(0，1)，(1，0)，(1，1)，构成了样本输入空间。例如，在二维平面上，对于“异或”运算，各个样本的分布如下图所示。如果p、q两个值不相同，则异或结果为1。如果p、q两个值相同，异或结果为0。

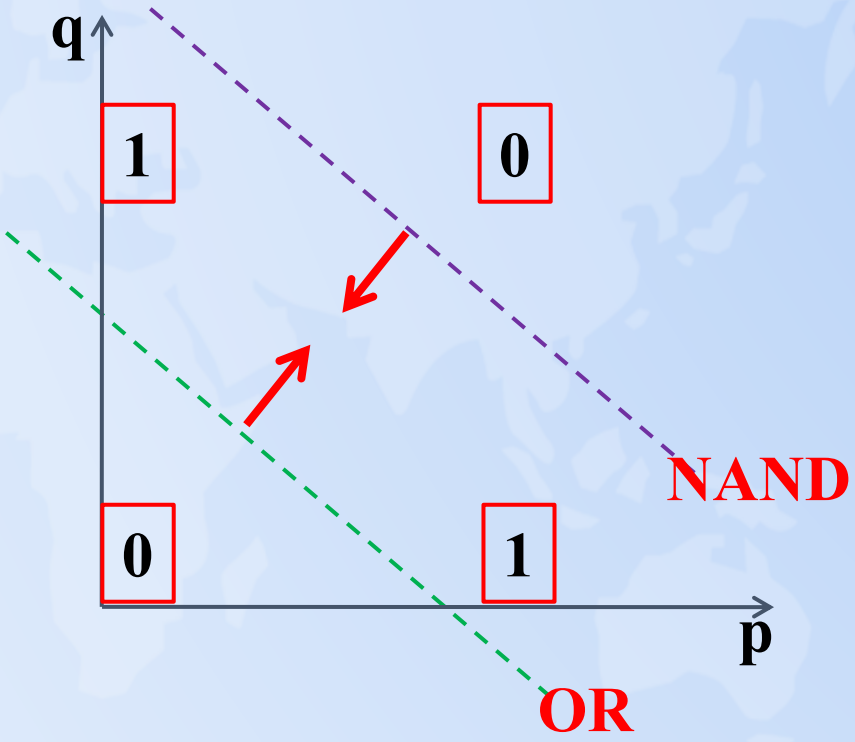


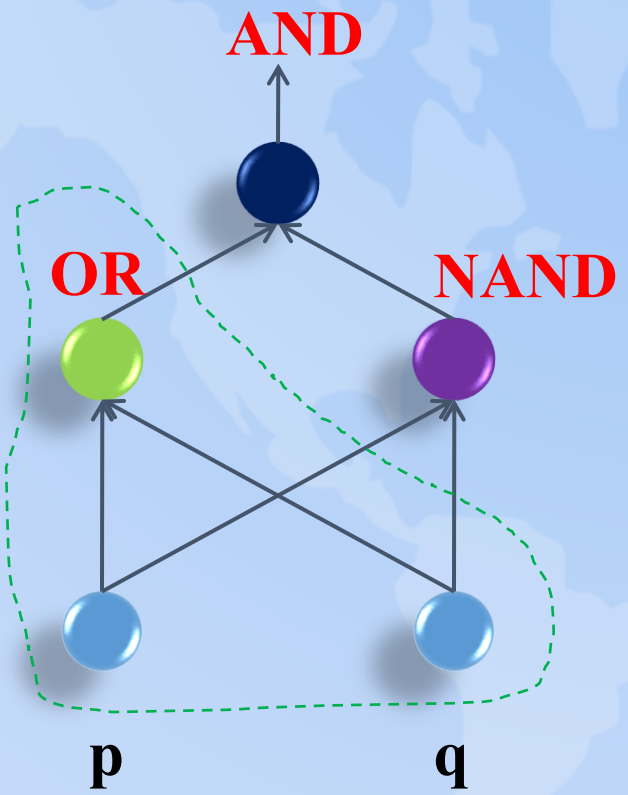
Input		Output
0	0	0
0	1	1
1	0	1
1	1	0

Cannot be separated by a single line.

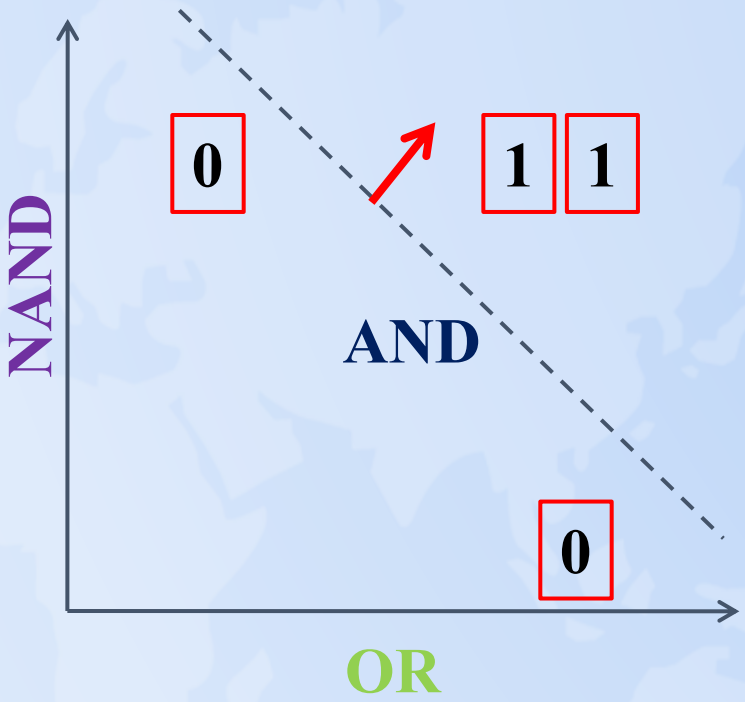
8.5 多层感知器

异或的数学符号为"⊕"　　$p \oplus q = \neg(p \wedge q) \wedge (p \vee q)$



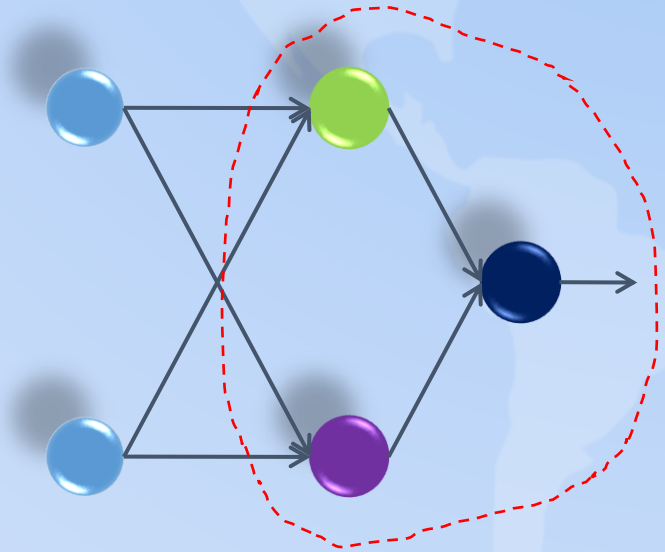


8.5 多层感知器

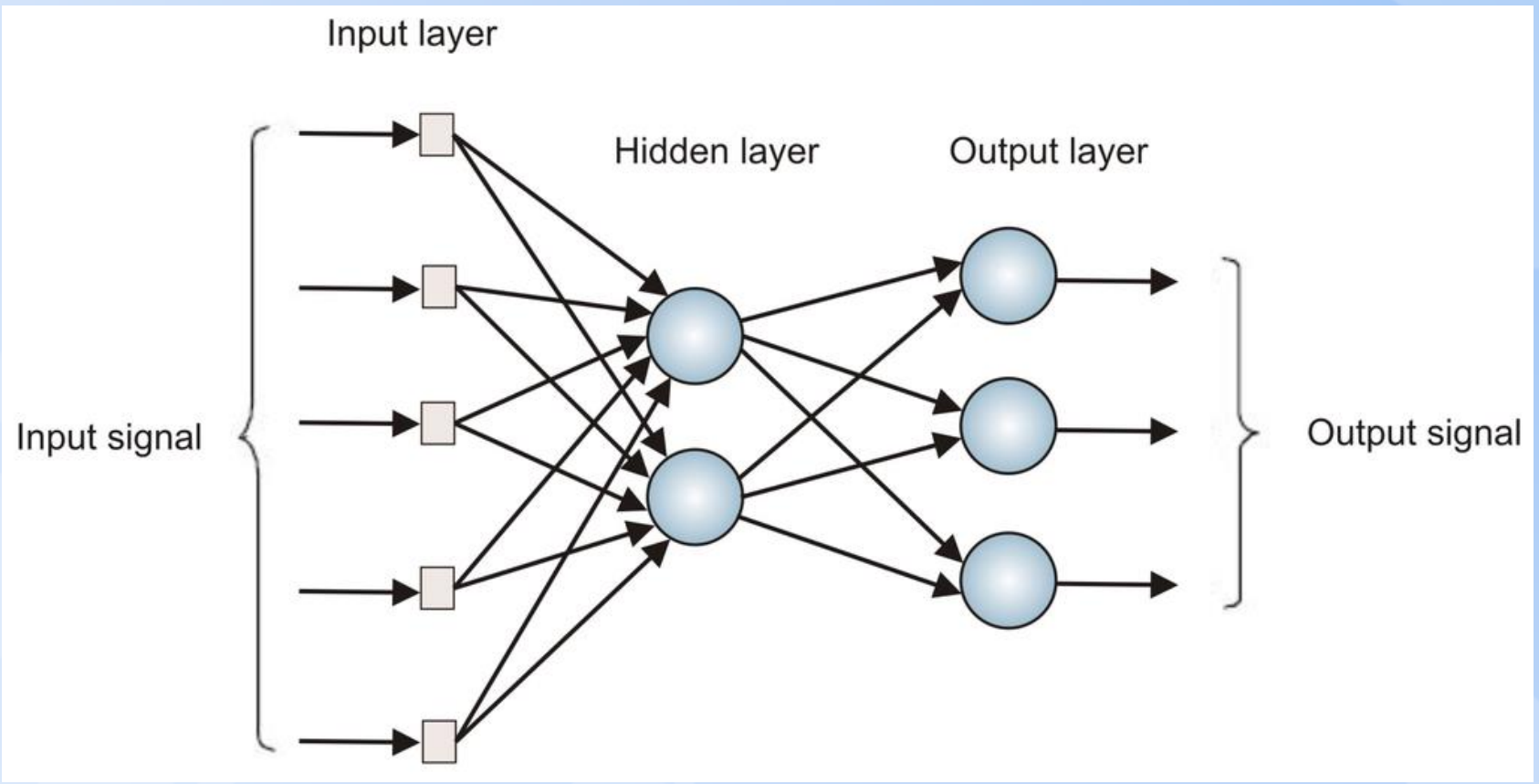


p	q	OR	NAND	AND
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

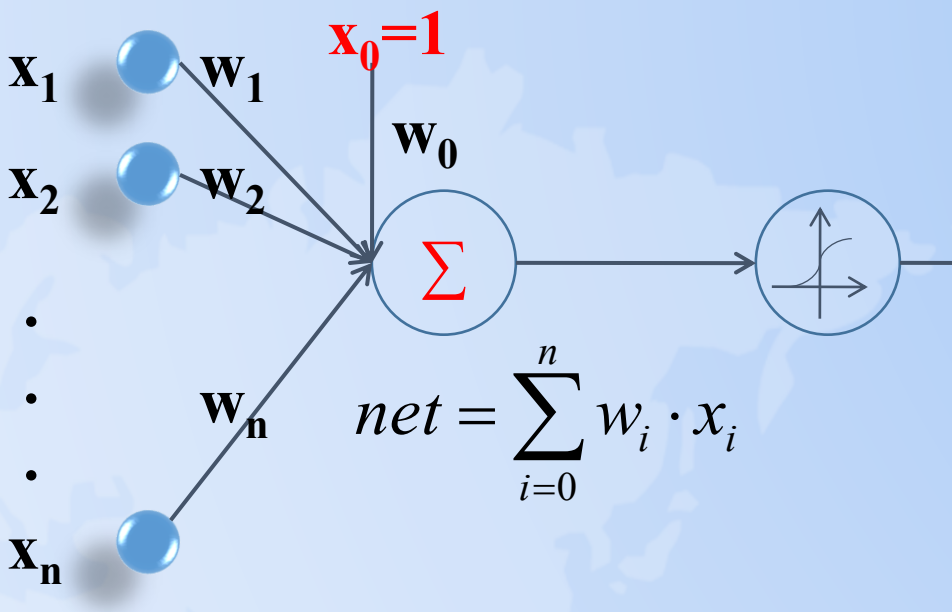
Input Hidden Output



8.5 多层感知器


Input layer
Hidden layer
Output layer
Input signal
Output signal

8.5 多层感知器



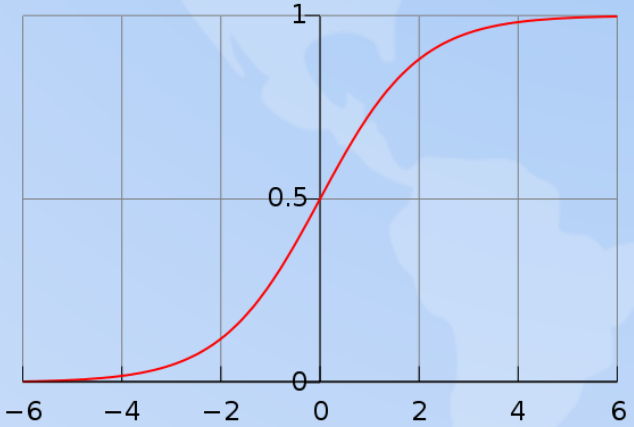
$$net = \sum_{i=0}^{n} w_i \cdot x_i$$

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

Sigmoid Function

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

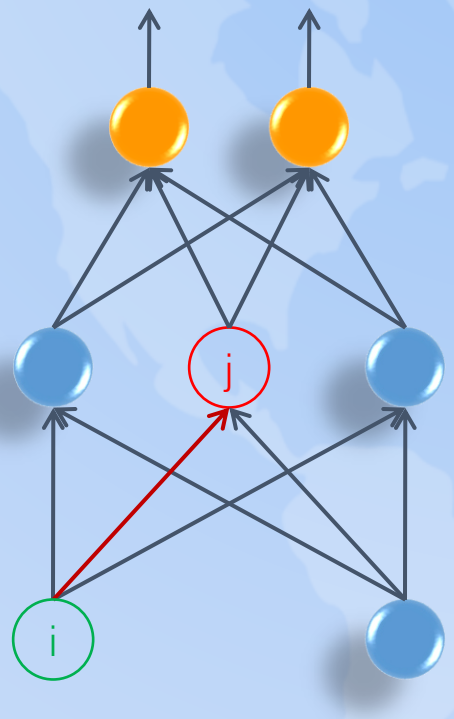
$$\frac{d\sigma(y)}{dy} = \sigma(y)\cdot(1-\sigma(y))$$



8.5 多层感知器

$$E_d(\vec{w}) \equiv \frac{1}{2}\sum_{k\in outputs}(t_k - o_k)^2 \qquad \Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j}\cdot\frac{\partial net_j}{\partial w_{ji}} = \boxed{\frac{\partial E_d}{\partial net_j}} x_{ji}$$



- x_{ji} = 第j个神经元的第 i 个输入
- w_{ji} = 第j个神经元的第 i 个权重
- $net_j = \sum w_{ji}x_{ji}$ (第j个神经元的输入和)
- o_j = 第j个神经元的预测输出
- t_j = 第j个神经元的目标输出
- σ = sigmoid 函数
- *outputs* = 最后一层网络的输出
- *Downstream* (j) = 第j个神经元的下游的所有神经元

8.5 多层感知器

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2$$

$$= \frac{1}{2} 2(t_j - o_j) \frac{\partial (t_j - o_j)}{\partial o_j}$$

$$= -(t_j - o_j)$$

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j} = o_j(1 - o_j)$$

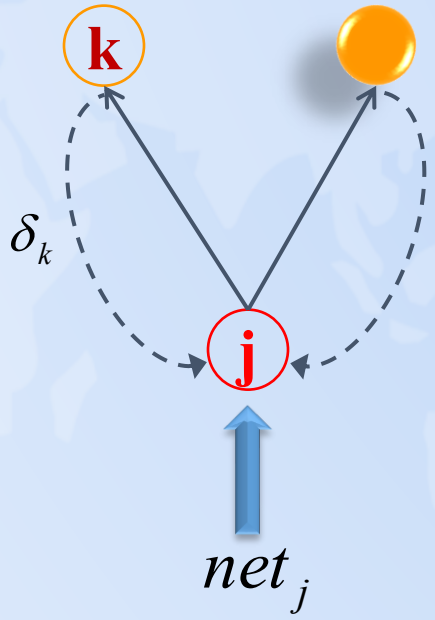
$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j) o_j (1 - o_j)$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}} = \eta (t_j - o_j) o_j (1 - o_j) x_{ji}$$

8.5 多层感知器

$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j} = \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k w_{kj} \frac{\partial o_j}{\partial net_j} = \sum_{k \in Downstream(j)} -\delta_k w_{kj} o_j (1 - o_j)$$



$$\delta_k = -\frac{\partial E_d}{\partial net_k}$$

$$\delta_j = o_j (1 - o_j) \sum_{k \in Downstream(j)} \delta_k w_{kj}$$

$$\Delta w_{ji} = \eta \delta_j x_{ji}$$

8.5 多层感知器

输　入：给定训练集X_{train}，其中每一个训练样本都是由一组输入和一组输出构成，所有的输入和输出都是[0,1]之间的浮点数据；神经网络结构：隐含层节点数目；神经网络每个节点的、参数化了的特征函数。

输　出：神经网络每个节点特征函数的参数。

（1）按照有序导数计算公式计算总体误差对于每个参数的有序导数公式；

（2）任意选择一组数据作为初始参数，一般选取（0,0,…,0），把这组初始参数作为当前参数；

（3）根据当前参数和总体误差计算公式计算总体误差，如果误差足够小，就把当前参数作为输出，退出；否则，继续下面的步骤；

（4）根据参数调整公式和当前参数数值，计算总体误差对于各参数的有序导数数值；

（5）计算各个参数的调整大小，并计算调整后的参数大小。把调整后的参数作为当前参数，回到第（3）步。

8.5 多层感知器

- 收敛性与局部极小问题
  - 搜索空间可能是高度多模态的.
  - 可能很容易陷入局部解决方案.
  - 需要用不同的初始权重进行多次试验.

- 进化神经网络
  - 黑盒优化技术(e.g., Genetic Algorithms)
  - 通常准确度更高
  - Xin Yao (1999) “Evolving Artificial Neural Networks”, Proceedings of the IEEE, pp. 1423-1447.

8.5 多层感知器

- 过拟合
  - 往往发生在后面的迭代中
  - 必要时使用验证数据集终止训练。

$$\Delta w_{ji}(n) = \eta \delta_j x_{ji} + \alpha \Delta w_{ji}(n-1)$$

- 实际考虑
  - 自适应学习率
    - 过小：收敛缓慢
    - 过大：快速收敛，不稳定