

# 模式识别

## 第五章 非线性判别函数

郭园方

北京航空航天大学计算机学院

# 回顾

- \* 线性判别函数的基本概念
- \* Fisher线性判别函数
- \* 感知准则函数
- \* 最小平方误差准则函数
- \* 多类问题
- \* 决策树

# 内 容

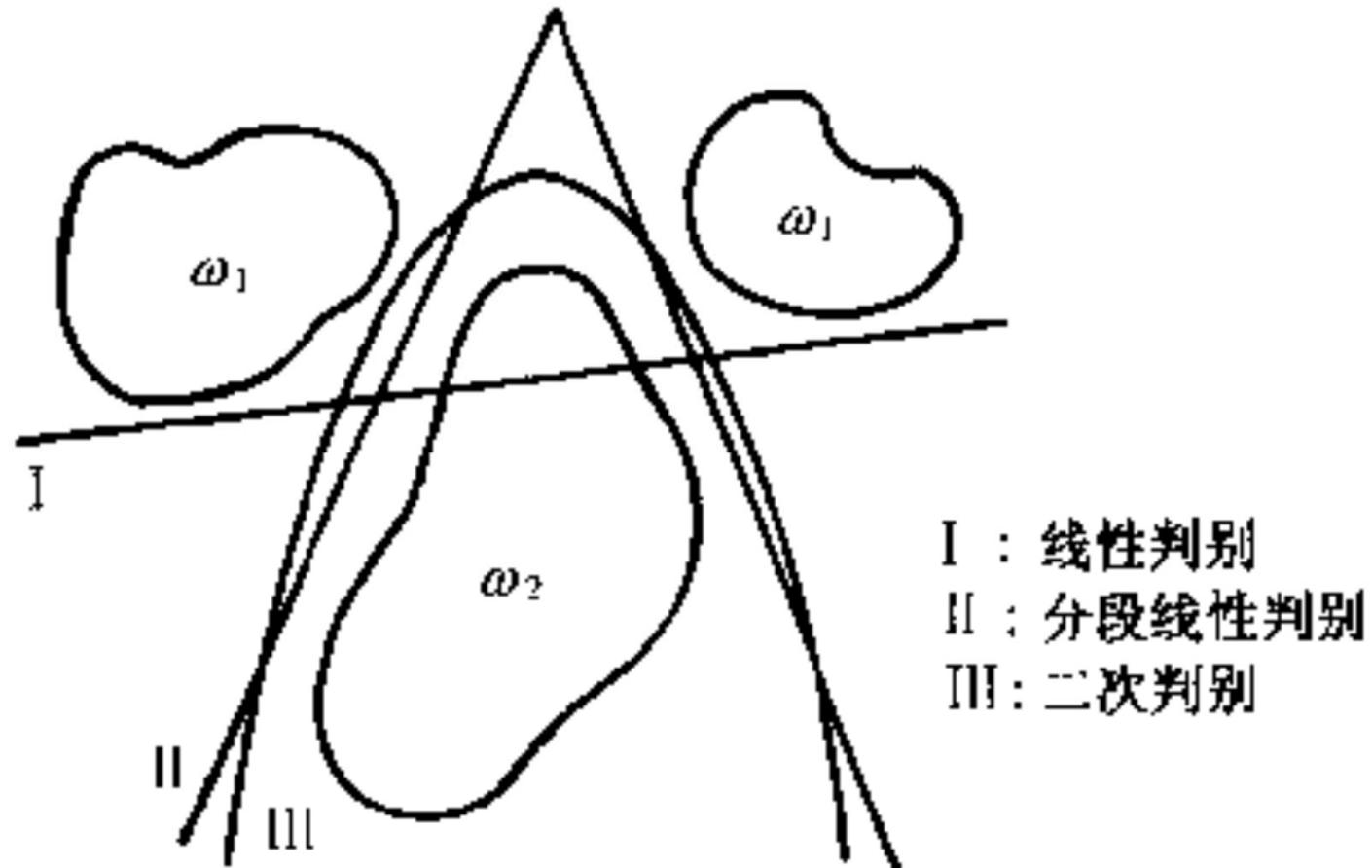
- \* 引言
- \* 分段线性判别函数
- \* 分段线性判别函数的设计
- \* 二次判别函数
- \* 神经网络概述
- \* **BP**神经网络
- \* 应用实例
- \* **BP** 网络模型改进
- \* 其他常用神经网络模型

# 引言

- \* 线性判别函数：简单、经济、实用
- \* 但线性不可分时错误率可能较大

问题线性不可分 {  
    噪声影响                          1. 使用新的特征  
    问题本身 {  
        2. 非线性变换  
        3. 非线性分类器}

# 引言



I : 线性判别  
II : 分段线性判别  
III : 二次判别

# 引言

- \* 实际问题中数据分布可能更复杂
- \* 需要：更复杂的非线性判别函数来分类
- \* 注：非线性判别函数并不明确指代一类判别函数，而是指除了线性判别函数之外的各种判别函数！

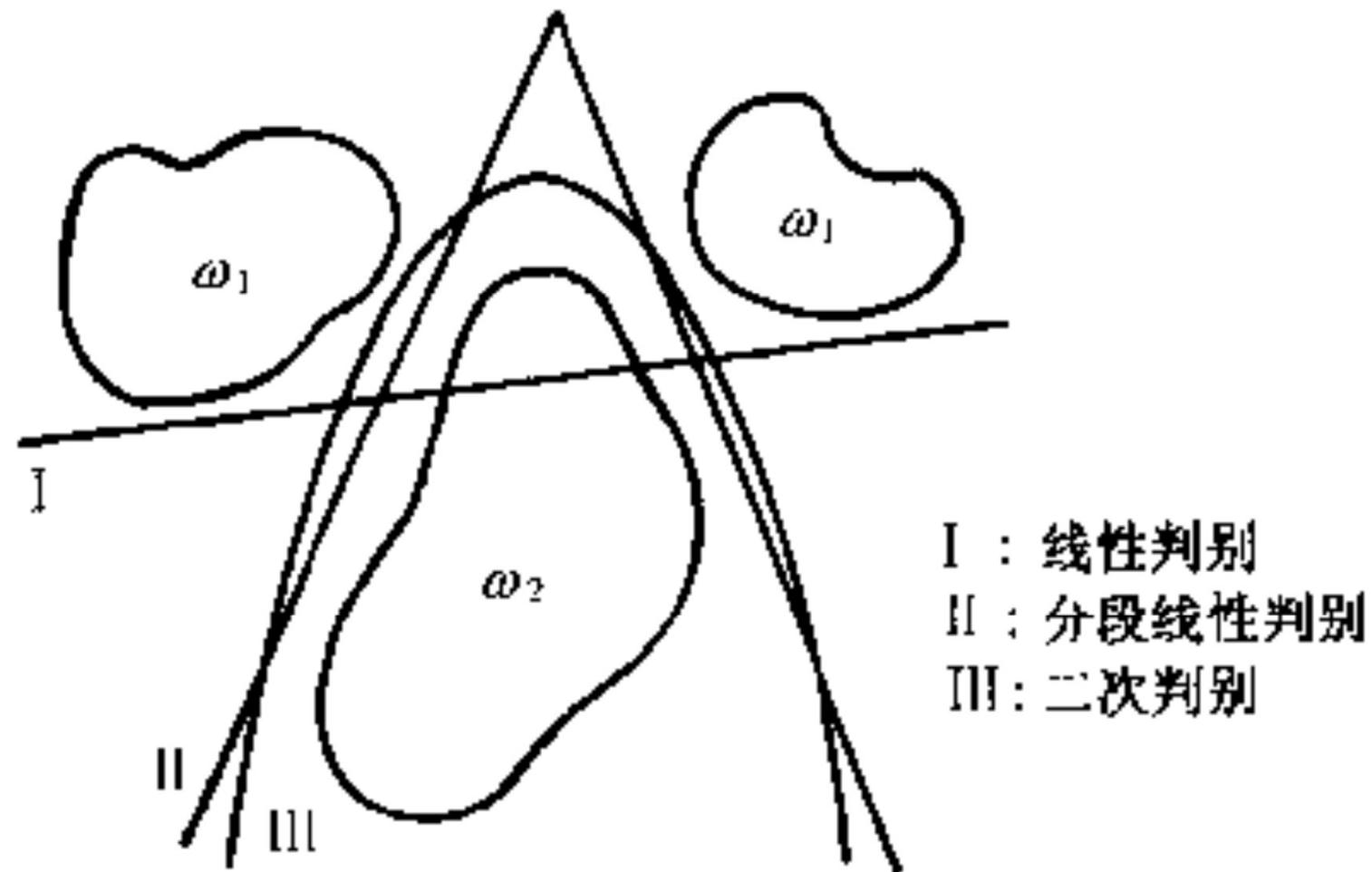
# 分段线性判别函数

- \* 决策面由若干个超平面段组成，计算相对比较简单
- \* 能够逼近各种形状的超平面，适应性很强
- \* 多类情况下的线性判别函数分类；
- \* 树状分类；

思路：

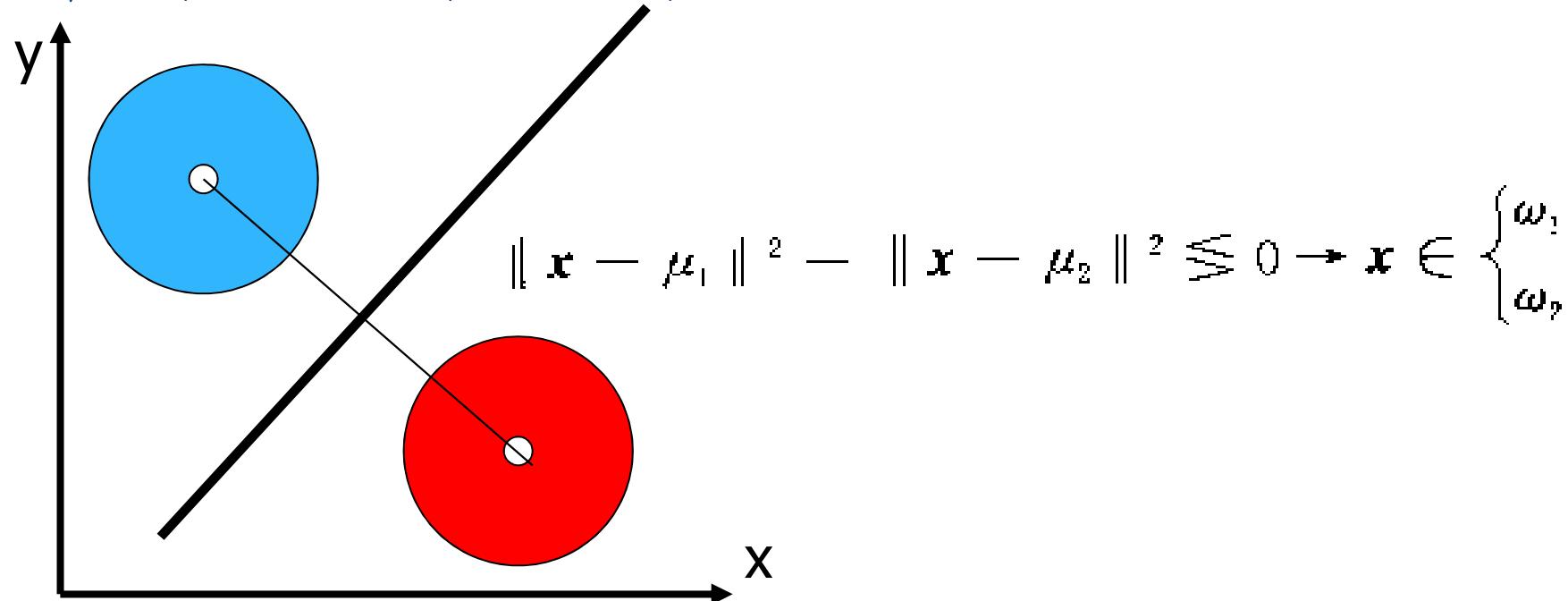
如果两类可以划分为线性可分的若干子类，  
则可以设计多个线性分类器，实现分段线性分类器。

# 分段线性判别函数



# 基于距离的分段线性判别函数

- \* 当两类的类条件概率密度为正态分布且两类先验概率相等，各维特征独立且方差相等时，最小错误率贝叶斯决策是基于最小距离的分段线性判别函数



# 基于距离的分段线性判别函数

- \* 最小距离分类器：把各类样本特征的均值向量作为各类的代表点（prototype），根据待识样本到各类别代表点的最小距离判别其类别。决策面就是两类别均值连线的垂直平分面。

# 基于距离的分段线性判别函数

- \* 分段线性距离分类器：将各类别划分为相对密集的子类，每个子类以它们的均值作为代表点，然后按最小距离分类
- \* 数学表达？

# 基于距离的分段线性判别函数

- \* 判别函数定义： $\omega_i$ 有 $l_i$ 个子类，即属于 $\omega_i$ 的决策域 $R_i$ 分成 $l_i$ 个子域 $R_i^1, R_i^2, \dots, R_i^{l_i}$ ，每个子区域用均值 $m_i^k$ 代表

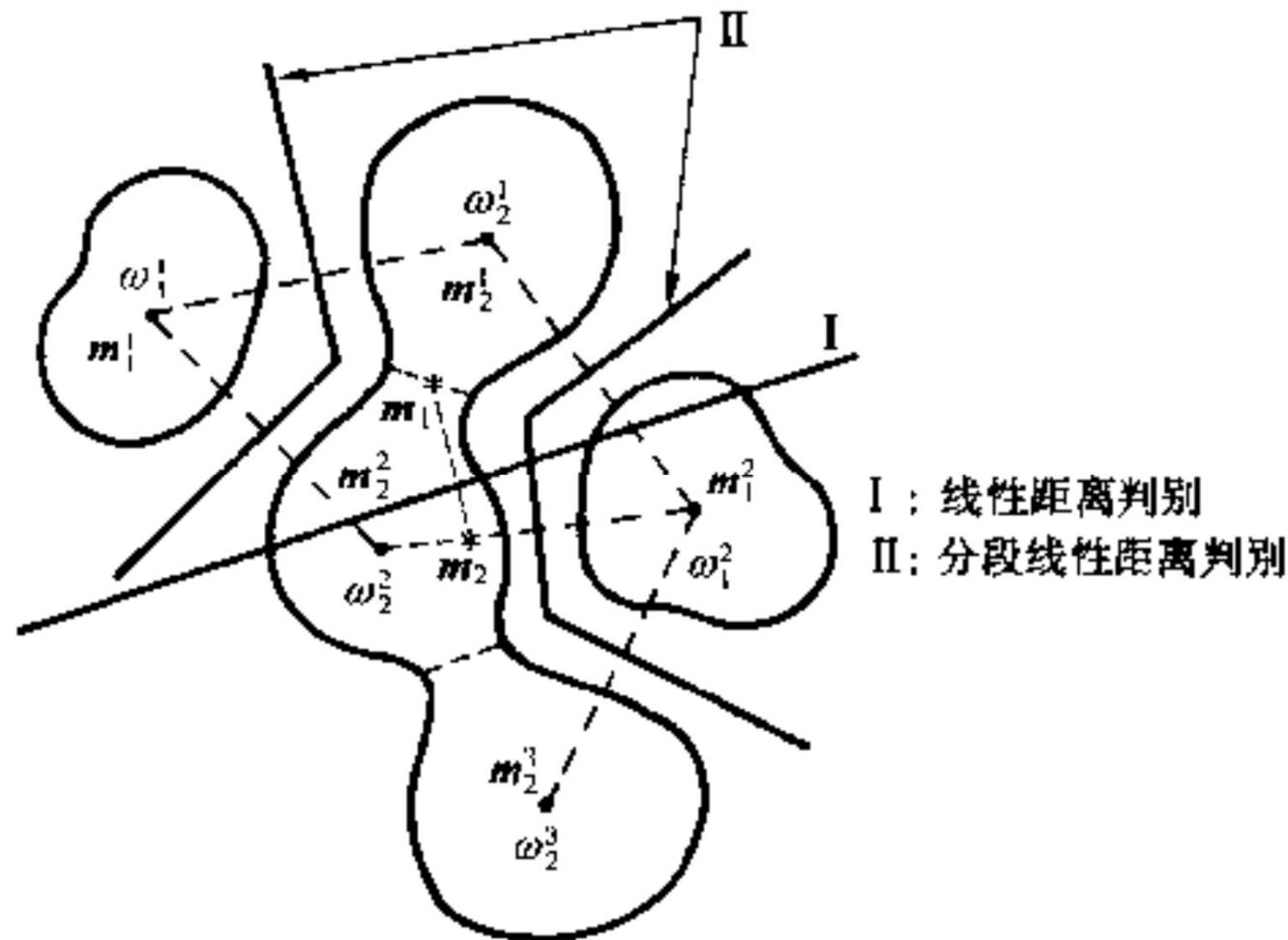
$$g_i(x) = \min_{k=1, \dots, l_i} \|x - m_i^k\|$$

- \* 判别规则：

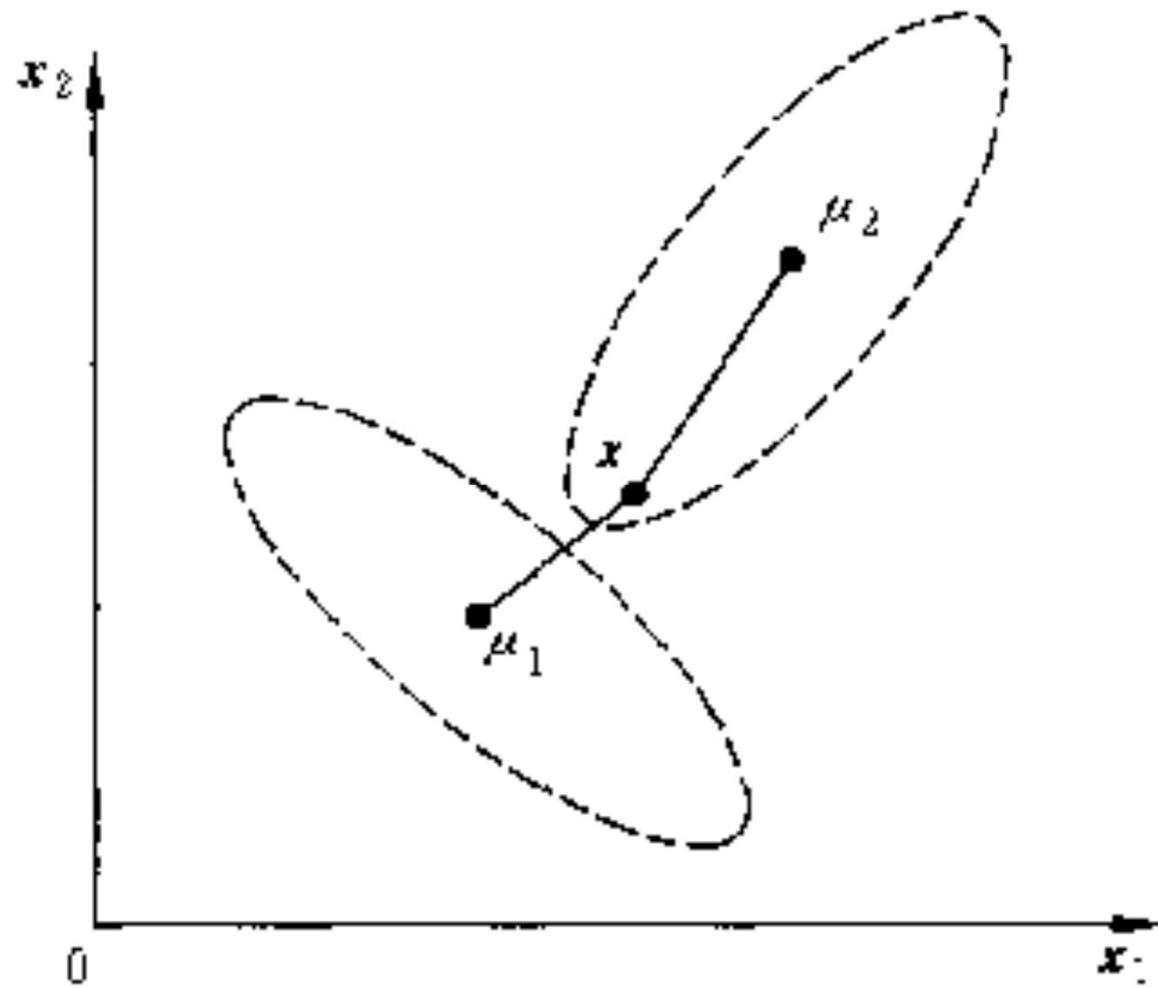
$$j = \operatorname{argmin}_{i=1, \dots, c} g_i(x)$$

- \* 如果 $g_j(x) = \operatorname{argmin}_{i=1, \dots, c} g_i(x)$ 则 $x$ 属于 $\omega_j$

# 基于距离的分段线性判别函数



# 一般的分段线性判别函数



# 一般的分段线性判别函数

- \* 解决办法：
- \* 将每个大类分成若干子类，针对每个子类定义一个线性判别函数

# 一般的分段线性判别函数

- \* 分段线性判别函数的一般形式： $g_i^k(x)$ 表示第*i*类第*k*段线性判别函数， $l_i$ 是第*i*类所具有的判别函数个数， $w_i^k$ 和 $w_{i0}^k$ 分别是第*k*段的权向量与阈值权。

$$g_i^k(x) = w_i^{(k)T} x + w_{i0}^k$$

其中  $k = 1, 2, \dots, l_i$ ,  $i = 1, 2, \dots, c$

# 一般的分段线性判别函数

\* 第*i*类判别函数：

$$g_i(x) = \max_{k=1,2,\dots,l_i} g_i^k(x)$$

\* 判别规则：

如果  $g_j(x) = \operatorname{argmin}_{i=1,\dots,c} g_i(x)$  则  $x$  属于  $\omega_j$

# 一般的分段线性判别函数

- \* 决策面取决于相邻的决策域，如果第*i*类的第*n*个子类与第*j*类的第*m*个子类相邻，则由它们共同决定的决策面方程为：

$$g_i^n(x) = g_j^m(x)$$

- \* 问题：

如何确定子类数目？

如何求得各子类的线性判别函数？

# 分段线性判别函数的设计

- \* 第一种情况：如果已知子类划分，则可直接使用多类线性分类方法
  - \* 如何知道子类划分？
  - \* 先验、聚类分析

# 分段线性判别函数的设计

- \* 第二种情况：只知道子类数目，不知道子类划分，用下述的错误修正法
- \* 注：此法介绍使用增广的判别函数形式表示

$$g_i^k(y) = \alpha_i^{(k)T} y$$

其中  $k = 1, 2, \dots, l_i$ ,  $i = 1, 2, \dots, c$

# 分段线性判别函数的设计

- \* 假设  $\omega_i, i = 1, 2, \dots, c$ ,  $\omega_i$  类中有  $l_i$  个子类，每一类均存在一定数量的训练样本
- \* (1) 初始化：任意给定各类各子类的权值  $\alpha_i^{(k)}(\mathbf{0})$ ，通常可以使用小的随机数
- \* (2) 在时刻  $t$ : 当前权值为  $\alpha_i^{(k)}(t)$ ，考虑某个训练样本  $y_v \in \omega_j$ ，找出  $\omega_j$  类的子类中最大的判别函数

$$\alpha_j^{(m)}(t)^T y_v = \max_{k=1,2,\dots,l_i} \alpha_j^{(k)}(t)^T y_v$$

# 分段线性判别函数的设计

- \* 考察当前权值对样本 $y_v$ 的分类情况：
  - \* 若 $\alpha_j^{(m)}(t)^T y_v > \alpha_i^{(k)}(t)^T y_v, \forall i = 1, 2, \dots, c,$   
 $i \neq j, k = 1, 2, \dots, l_i$ , 则 $\alpha_i^{(k)}(t)$ 不变
  - \* 若对某个 $i \neq j, k = n$ , 有 $\alpha_j^{(m)}(t)^T y_v \leq \alpha_i^{(n)}(t)^T y_v$ , 则 $y_v$ 被错分, 对其中最大者, 记为 $(i', n')$

# 分段线性判别函数的设计

\* 修正

$$\alpha_j^{(m)}(t+1) = \alpha_j^{(m)}(t) + \rho_t y_t$$

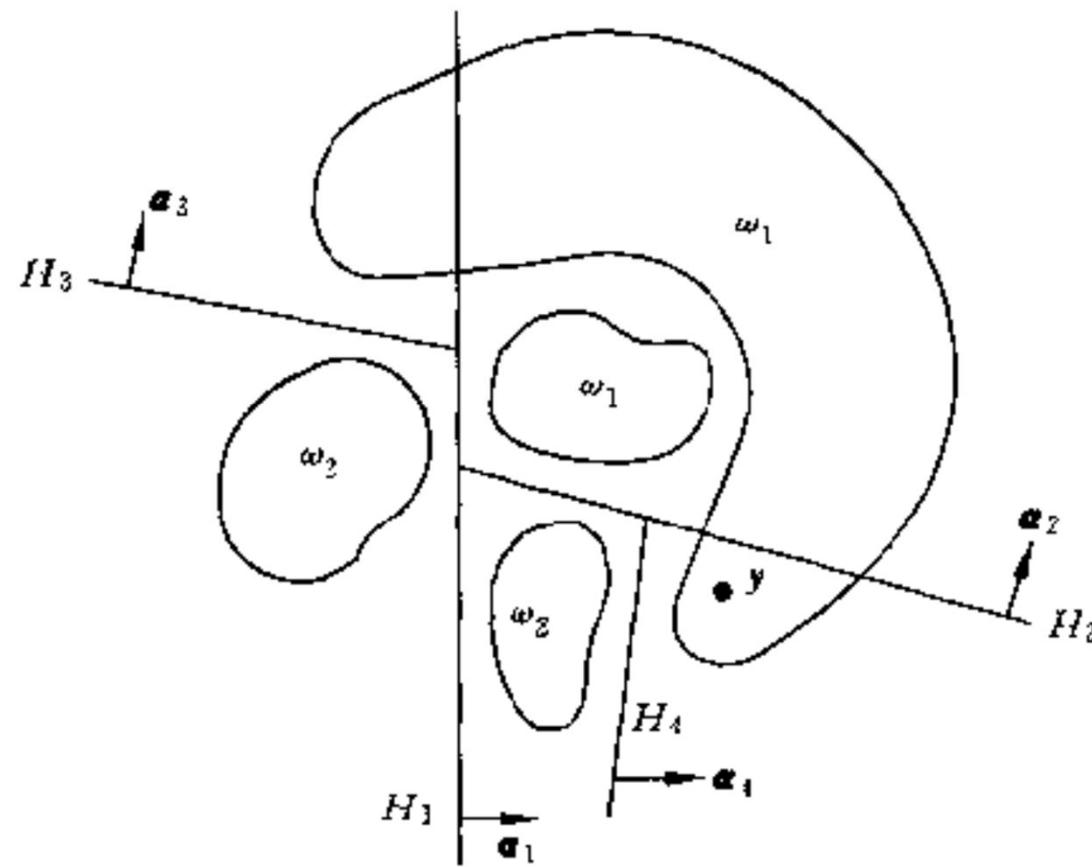
$$\alpha_{i'}^{(n')}(t+1) = \alpha_{i'}^{(n')}(t) - \rho_t y_t$$

\* (3) 对下一个样本重复 (2) , 直到收敛

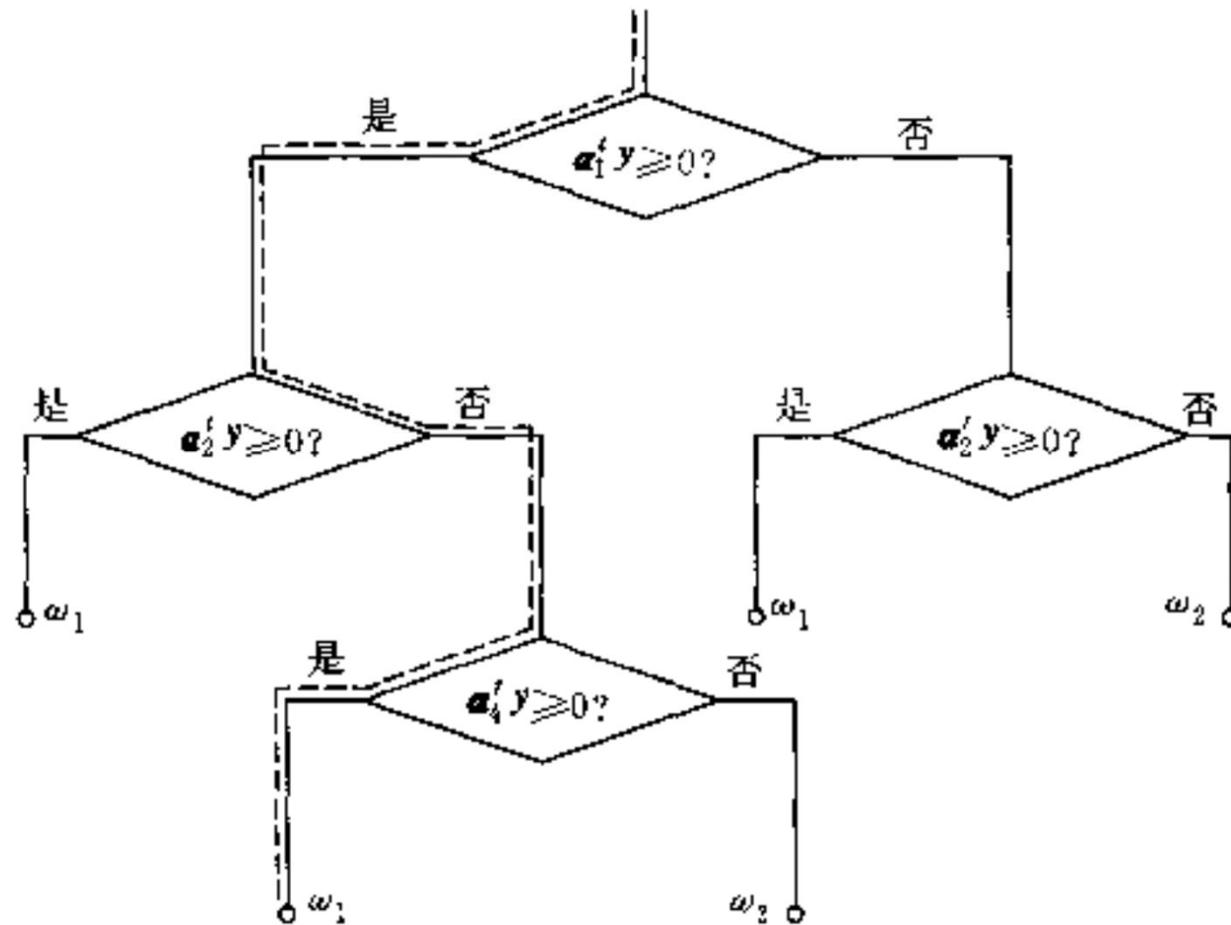
# 分段线性判别函数的设计

- \* 第三种情况：未知子类数目
- \* 可使用树状分段线性分类器
  - \* 首先设计一个线性分类器，分成两个子类
  - \* 若子类中有错分，则在其中再分，直到全部正确分类

# 分段线性判别函数的设计



# 分段线性判别函数的设计



# 用凹函数的并表示分段线性函数

- \* 设  $I_i$  为线性判别函数， $i=1,2,\dots,r$ ，则：
- \* (a):  $I_1, I_2, \dots, I_r$  都是分段线性判别函数
- \* (b): 若  $A, B$  都是分段线性判别函数，则：  
 $A \wedge B, A \vee B$  也是分段线性判别函数。 $A \wedge B$  取最小， $A \vee B$  取最大。
- \* (c): 对任何分段线性函数都可以表示成如下二种形式：

# 用凹函数的并表示分段线性函数

- \* 1) 析取范式(这是经常采用的形式)

$$P = (L_{11} \wedge L_{12} \wedge \dots \wedge L_{1m}) \vee \dots \vee (L_{q1} \wedge L_{q2} \wedge \dots \wedge L_{qm})$$

- \* 2) 合取范式

$$Q = (L_{11} \vee L_{12} \vee \dots \vee L_{1m}) \wedge \dots \wedge (L_{q1} \vee L_{q2} \vee \dots \vee L_{qm})$$

- \* 每个 $(L_{11} \wedge L_{12} \wedge \dots \wedge L_{1m})$ 都称为凹函数。

# 用凹函数的并表示分段线性函数

\* 对于多峰二类问题：设第一类有 $q$ 个峰，则有 $q$ 个凹函数。

$$\text{即 } P = P_1 \vee P_2 \vee \dots \vee P_q$$

每个凹函数 $P_i$ 由 $m$ 个线性判别函数来构成。

$$\therefore P_i = L_{i1} \wedge L_{i2} \wedge \dots \wedge L_{im}$$

# 用凹函数的并表示分段线性函数

假设对于每个子类线性判别函数 $L_{ij}$ 都设计成：

$$L_{ij} = w_{ij}x \begin{cases} > 0, x \in \omega_1, i = 1, 2, \dots, q \text{ 子类。} \\ < 0, x \in \omega_2, j = 1, 2, \dots, m \text{ 每个子类的判别函数数。} \end{cases}$$

\* 最终：

判别规则:

$$\begin{cases} P > 0, \text{ 则 } x \in \omega_1 \\ P \leq 0, \text{ 则 } x \in \omega_2 \end{cases}$$

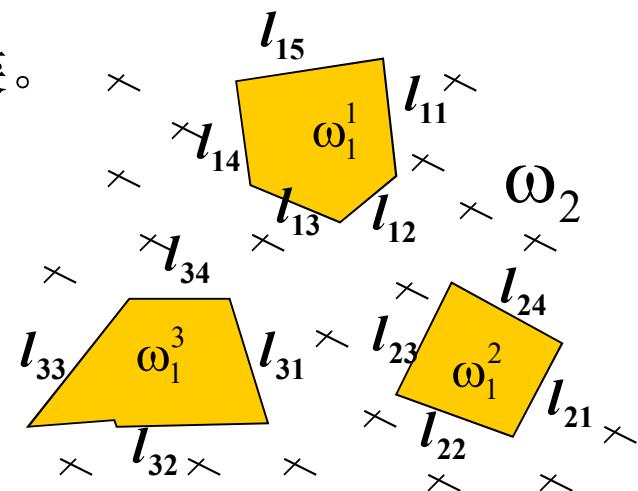
# 用凹函数的并表示分段线性函数

## \* 示例：

$\omega_1$  分三个峰， $q = 3$  这样它有三个子类。

判别函数个数：
$$\begin{cases} m_1 = 5 \\ m_2 = 4 \\ m_3 = 4 \end{cases}$$

∴ 有 13 个分段判别函数



# 用凹函数的并表示分段线性函数

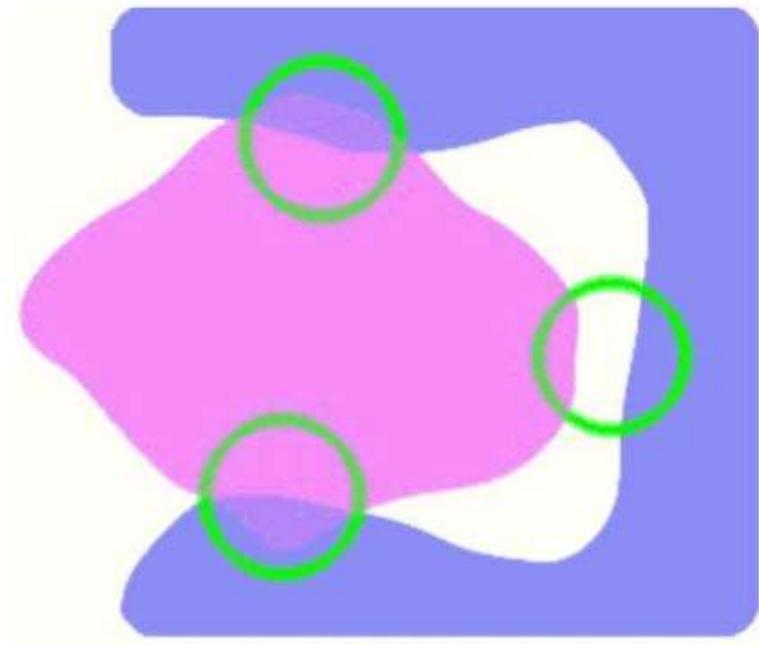
$$* \therefore P = (L_{11} \wedge L_{12} \wedge L_{13} \wedge L_{14} \wedge L_{15}) \vee (L_{21} \wedge L_{22} \wedge L_{23} \wedge L_{24}) \vee (L_{31} \wedge L_{32} \wedge L_{33} \wedge L_{34})$$

$$\therefore P = \max\{\min(l_{11}, l_{12}, \dots, l_{15}), \min(l_{21}, \dots, l_{24}), \min(l_{31}, \dots, l_{34})\}$$

若  $P > 0$ , 则  $x \in \omega_1^\circ$  若  $P \leq 0$ , 则  $x \in \omega_2^\circ$

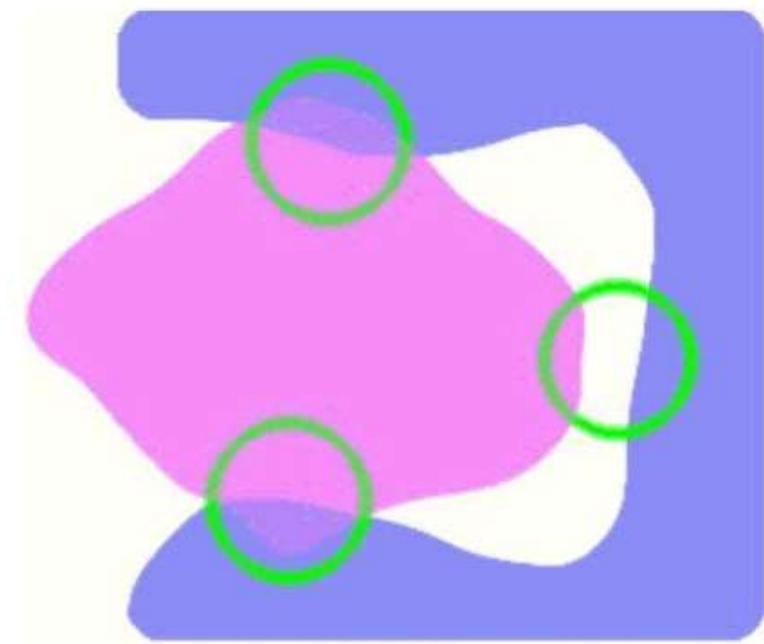
# 用交遇区的样本设计分段线性函数

- \* 思想：寻找两类中最靠近的样本子集，用它们设计分类器！
- \* 出发点：与前述方法不同，其出发点是类间的分界面必然处在两类样本的交界处，因此只需找到交界处样本，对这些邻近不同类样本确定分类面即可。



# 用交遇区的样本设计分段线性函数

- \* 实际上，决策面都处在不同类别样本分布的交界处或邻接处所在区域内，如右图所示，两类物体在特征空间分布中有若干处很接近或者有交叠。



# 用交遇区的样本设计分段线性函数

\* 实现步骤：

- (1) 用聚类分析等方法把每类样本分为若干子类（原型区）
- (2) 考查子类之间的距离

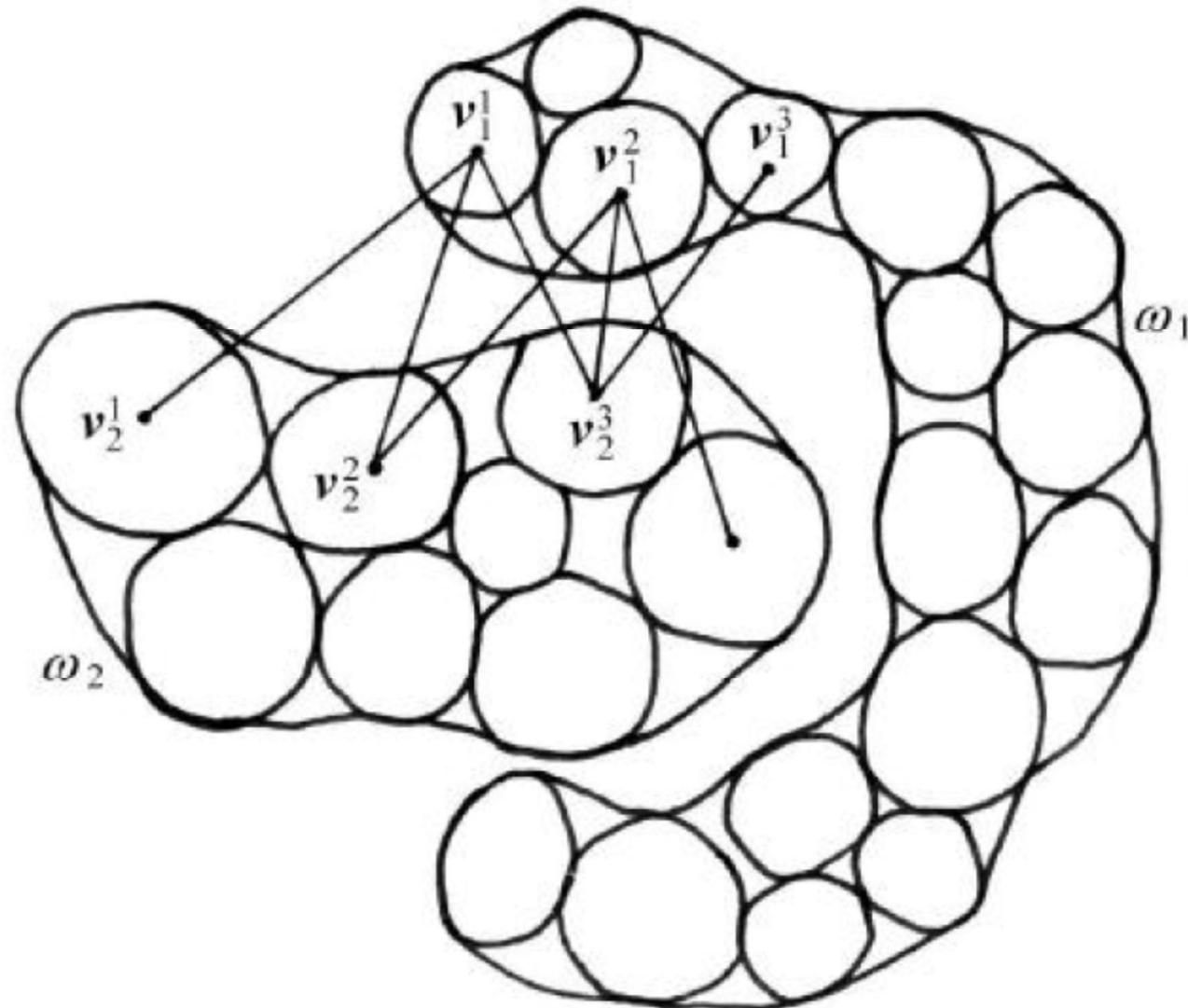
$$d(v_i^m, v_j^n)$$

- (3) 寻找紧互对原型对

$$d(v_i^m, v_j^n) = \min_{l=1, \dots, l_i} d(v_i^l, v_j^n) = \min_{l=1, \dots, l_j} d(v_i^m, v_j^l)$$

- (4) 用紧互对原型对设计分类面

# 用交遇区的样本设计分段线性函数



# 局部训练法

具体步骤：

## 步骤一：产生初始决策面

首先由紧互对原型对集合中最近的一对，产生一个初始决策面的方程。例如可由这两个原型的垂直平分平面作为初始分界面，表示成  $H_1$ ；

## 步骤二：初始决策面最佳化

确定超平面  $H_1$  能正确分类的所有紧互对原型对，并用这些原型对代表的聚类中的所有样本组成局部训练样本集，按所使用的准则设计出线性决策面  $H_1^*$ ，该决策面对现有局部样本集来说是最佳的。对该决策面  $H_1^*$  又可找出它能正确分类的所有紧互对原型对。如果  $H_1^*$  与  $H_1$  的分类效果相同，则  $H_1^*$  不需要再调整。否则由  $H_1^*$  作为初始决策面，重复上述过程，直到两者完全相同为止。

# 局部训练法

## 步骤三： 新决策面的产生与最佳化

在找到一个最佳的决策面段后，将相应的局部训练样本从原紧互对原型对集合中拿走，然后在剩下的紧互对原型对集合重复上述步骤，产生第二个超平面段。

## 步骤四： 分段线性分类器

重复上述步骤，直到所有紧互对原型对都被处理完毕，得到一系列超平面，组成分段线性分类器。

在使用上述方法得到一组超平面作为分段线性分类器的分界面后，仅对交遇区的样本集进行性能检测有时不能发现存在的问题，需要使用全体样本对其进行性能检验，观察其能否对全体样本作出合理的划分？

# 用交遇区的样本设计分段线性函数

(5) 决策规则 (考虑有错分情况)

设最后得到  $m$  个超平面  $H_i: \alpha_i^T y = 0, i = 1, \dots, m$

记  $z_i(x) = \begin{cases} 1, & \alpha_i^T y > 0 \\ 0, & \alpha_i^T y \leq 0 \end{cases}, \quad i = 1, \dots, m$

得  $z(x) = [z_1(x), z_2(x), \dots, z_m(x)]$   $m$  维二值向量,  $2^m$  种值

对  $z(x)$  的每一种可能的取值  $z_j, j = 1, \dots, 2^m$ , 统计其在  $X_1$  和  $X_2$  两

类样本中出现的次数  $N_1(z_j)$  和  $N_2(z_j)$ 。

# 用交遇区的样本设计分段线性函数

定义  $\Omega(z_j)$ : 若  $N_1(z_j)$  和  $N_2(z_j)$  都很小, 则  $\Omega(z_j) = \delta$

否则若  $L = \frac{N_1(z_j)}{N_1(z_j) + N_2(z_j)} > \frac{1}{2}$ , 则  $\Omega(z_j) = \begin{cases} 1 \\ 0 \end{cases}$

# 用交遇区的样本设计分段线性函数

\* 最终：

决策规则：对输入  $x$ ，若

$$\begin{cases} \Omega(z(x)) = \delta, & \text{则拒绝} \\ \Omega(z(x)) = 1, & \text{则 } x \in \omega_1 \\ \Omega(z(x)) = 0, & \text{则 } x \in \omega_2 \end{cases}$$

若对某些  $z_j$ ， $L \approx \frac{1}{2}$ ，则说明两类差别不大，应进一步划分子类，增大  $m$

# 二次判别函数

\* 二次判别函数一般可表示成：

$$\begin{aligned}g(x) &= X^T \bar{W} X + W^T X + W_0 \\&= \sum_{i=1}^n w_{ii} x_i^2 + 2 \sum_{j=1}^{n-1} \sum_{i=j+1}^n w_{ji} x_j x_i + \sum_{j=1}^n w_j x_j + W_0\end{aligned}$$

其中：  $\bar{W}$  是  $n \times n$  维的权向量。

$W$  为  $n$  维权向量

$g(x)$  的系数一共有  $l = \frac{1}{2}n(n+3)+1$  (非常复杂，计算量很大)

# 二次判别函数

二次决策面为超二次曲面。(超球面， 超双曲面等)

(1)若已知样本 $\omega_1$ 分布比较集中, 形成单峰,  $\omega_2$ 分布分散。如下图:  
定义 $\omega_1$ 判别函数 :

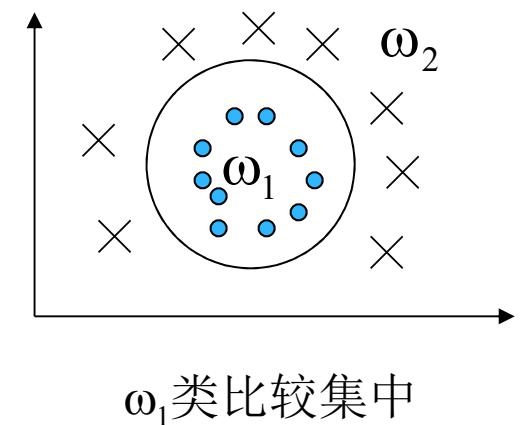
$$g(x) = k^2 - (x - \bar{\mu}_1)^T \sum_1^{-1} (x - \bar{\mu}_1), k\text{的大小, 决定超平面的大小。}$$

其中:  $\bar{\mu}_1$ 为 $\omega_1$ 均值,  $\sum_1$ 为 $\omega_1$ 协方差

判别规则:

$$\begin{cases} g(x) > 0, x \in \omega_1 \\ g(x) < 0, x \in \omega_2 \end{cases}$$

判别平面:  $g_1(x) = 0$ 是个超球面  
由k控制大小

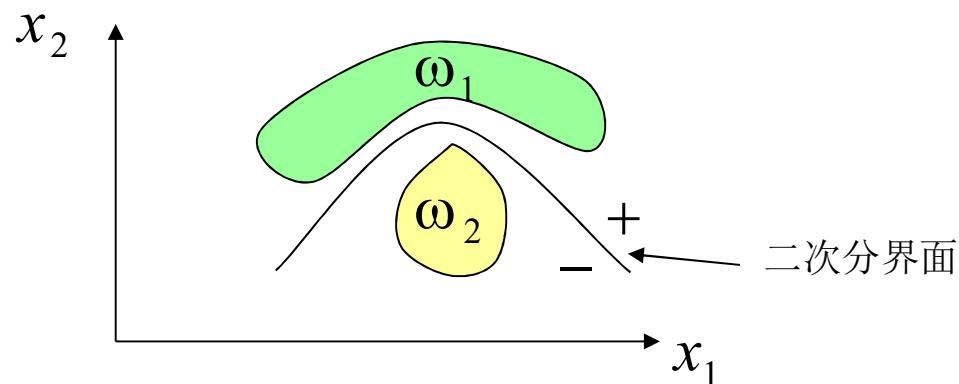


# 二次判别函数

(2) 如果  $\omega_1, \omega_2$  都比较集中，那么定义 两个判别函数：

$$g_i(x) = k_i^2 - (x - \bar{\mu}_i)^T \sum_i^{-1} (x - \bar{\mu}_i), \quad i = 1, 2$$

其中：  $\bar{\mu}_i$  为  $\omega_1, \omega_2$  均值,  $\sum_i$  为  $\omega_1, \omega_2$  协方差



# 二次判别函数

判别平面方程:

$$g(x) = g_1(x) - g_2(x)$$
$$= -x^T (\sum_1^{-1} - \sum_2^{-1})x + 2(\bar{\mu}_1^T \sum_1^{-1} - \bar{\mu}_2^T \sum_2^{-1})x -$$
$$(\bar{\mu}_1^T \sum_1^{-1} \bar{\mu}_1 - \bar{\mu}_2^T \sum_2^{-1} \bar{\mu}_2) + (k_1^2 - k_2^2) = 0$$

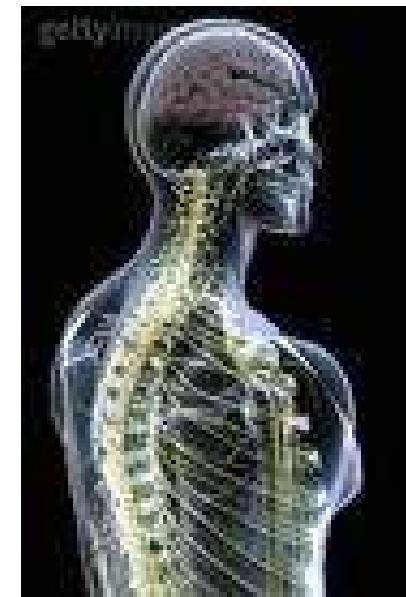
判别规则:

$$g(x) \begin{cases} > 0, x \in \omega_1 \\ < 0, x \in \omega_2 \end{cases}$$

$k_1, k_2$  可用来调整二类错误率。

# 神经网络简介

- \* 神经网络（NNs）它是一种模拟动物神经网络行为特征，进行分布式并行信息处理的算法。
- \* 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。



人类的神经网络

# 神经网络基础知识

- \* 构成：大量简单的基本元件——神经元相互连接
- \* 工作原理：模拟生物的神经处理信息的方式
- \* 功能：进行信息的并行处理和非线性转化
- \* 特点：
  - \* 比较轻松地实现非线性映射过程
  - \* 具有大规模的计算能力

# 人工神经网络研究的发展

- \* 40年代初，美国McCulloch和Pitts从信息处理的角度，研究神经细胞行为的数学模型表达。提出了二值神经元模型。**MP**模型的提出开始了对神经网络的研究进程。
- \* 1949年心理学家Hebb提出著名的**Hebb**学习规则，即由神经元之间结合强度的改变来实现神经学习的方法。
- \* 50年代末期，Rosenblatt提出感知机模型，它基本符合神经营生生理学的原理。在60代掀起了神经网络研究的第一次高潮。
- \* 1969年，人工智能创始人之一的Minsky和Papert以出版了《感知器》，从数学上深入分析了感知器的原理，指出其局限性。

# 人工神经网络研究的发展

- \* Minsky的结论是悲观的，神经网络的研究进入了低潮。主要有自适应共振理论，自组织映射，认知机网络模型理论，BSB模型等等，为神经网络的发展奠定了理论基础。
- \* 1982年，Hopfield提出了人工神经网络的一种数学模型，引入了能量函数的概念，设计出用电子线路实现这一网络的方案，大大促进了人工神经网络的研究。
- \* 1986年，Rumelhart及LeCun等学者提出了多层感知器的反向传播算法，解决了多层前向神经网络的学习问题，证明了多层神经网络具有很强的学习能力。
- \* 90年代末以来，许多具备不同信息处理能力的神经网络已被提出来并应用于许多信息处理领域，如模式识别、自动控制、信号处理、决策辅助、人工智能等方面。

# 人工神经网络研究的发展

- \* 当前
  - \* 削弱的趋势
  - \* 深度神经网络的兴起和繁荣

# 神经网络的优劣势

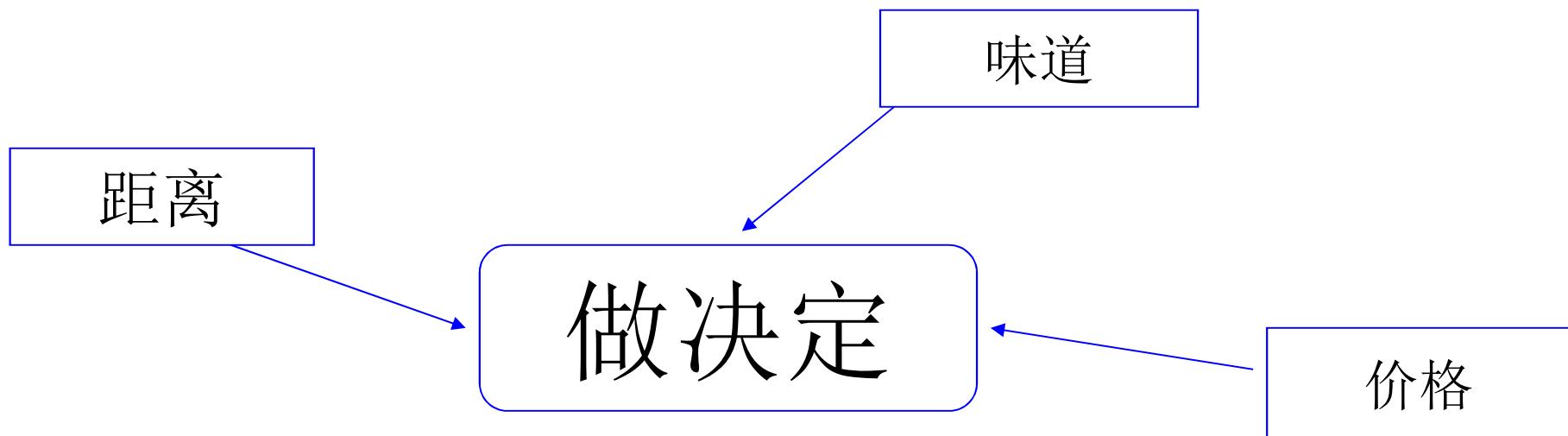
## \* 优势

- \* 具有很强的自适应学习能力；
- \* 可以实现特征空间较复杂的划分；
- \* 能够适于用高速并行处理系统实现。

## \* 劣势

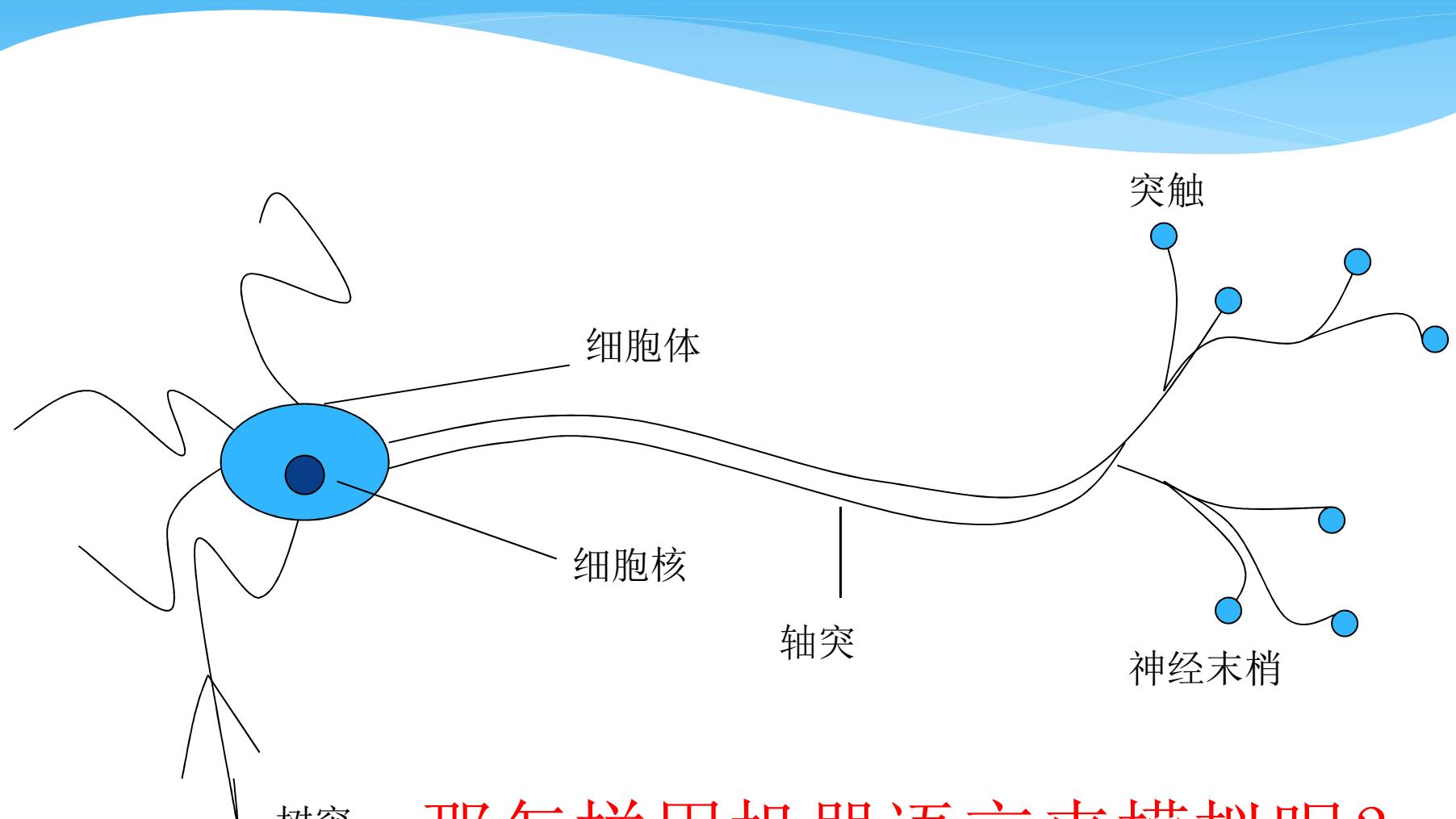
- \* 需要更多的训练数据；
- \* 在通常的计算机上实现模拟运行速度较慢；
- \* 无法得到所使用的决策过程的透彻理解。

# 神经网络的本质



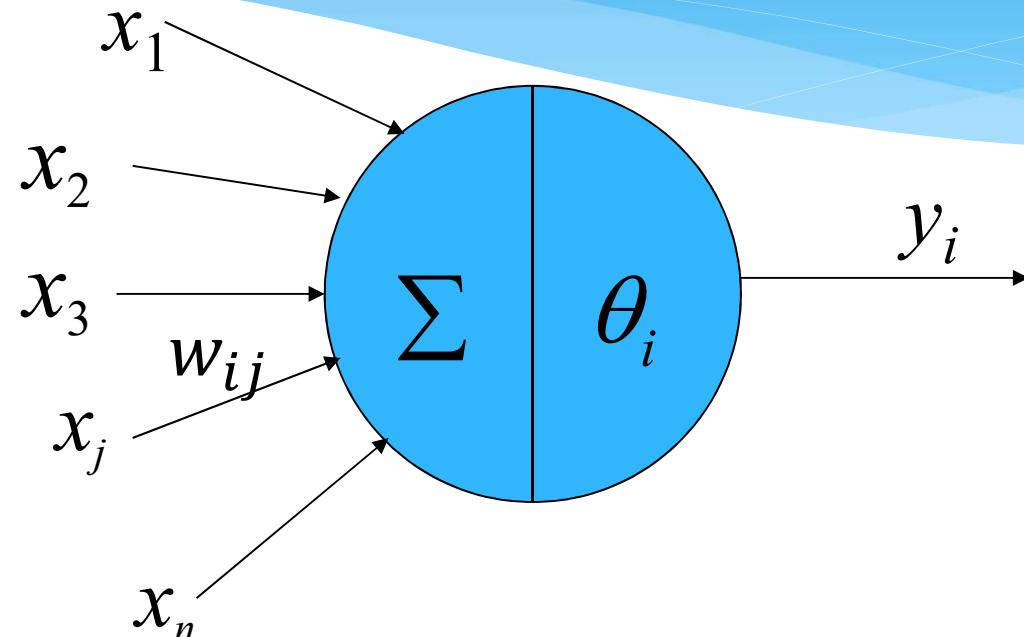
- \* 神经网络的本质就是利用计算机语言模拟人类大脑做决定的过程

# 生物神经元结构



那怎样用机器语言来模拟呢？

# 神经元结构模型



- \*  $x_j$  为输入信号,  $\theta_i$  为阈值,  $w_{ij}$  表示与神经元  $x_j$  连接的权值
- \*  $y_i$  表示输出值
- \* 判断  $x_j w_{ij}$  是否大于阈值  $\theta_i$

# 什么是阈值？

- \* 临界值。比如有一头驴，往它身上压稻草，一根一根地压，当压到N根时，还没有被压倒，又压了一根，倒了，这时所有压在驴身上的稻草的多少，就是压倒这头驴的阈值。
- \* 神经网络是模仿大脑的神经元，当外界刺激达到一定的阈值时，神经元才会受刺激，影响下一个神经元。

# 什么是阈值？

- \* 判断 $\sum_{j=1}^n x_j w_{ij}$ 是否大于阈值 $\theta_i$
- \* 若 $\sum_{j=1}^n x_j w_{ij}$ 大于阈值 $\theta_i$ ，则此神经元接受此信息，输出 $y_i = f(\sum_{j=1}^p x_j w_{ij})$
- \* 若 $\sum_{j=1}^n x_j w_{ij}$ 小于阈值 $\theta_i$ ，则此神经元不接受此信息的传递

# 几种代表性的网络模型

- \* 单层前向神经网络—线性网络
- \* 阶跃网络
- \* 多层前向神经网络(反推学习规则即**BP**神经网络)
- \* **Elman**网络、**Hopfield**网络、双向联想记忆网络、自组织竞争网络等

# 神经网络能干什么？

- \* 运用这些网络模型可实现**函数逼近、数据聚类、模式分类、优化计算**等功能。因此，神经网络广泛应用于人工智能、自动控制、机器人、统计学等领域的信息处理中。
- \* 虽然神经网络的应用很广，但是在具体的使用过程中到底应当选择哪种网络结构比较合适是值得考虑的。这就需要我们对各种神经网络结构有一个较全面的认识

# 神经网络的应用范围

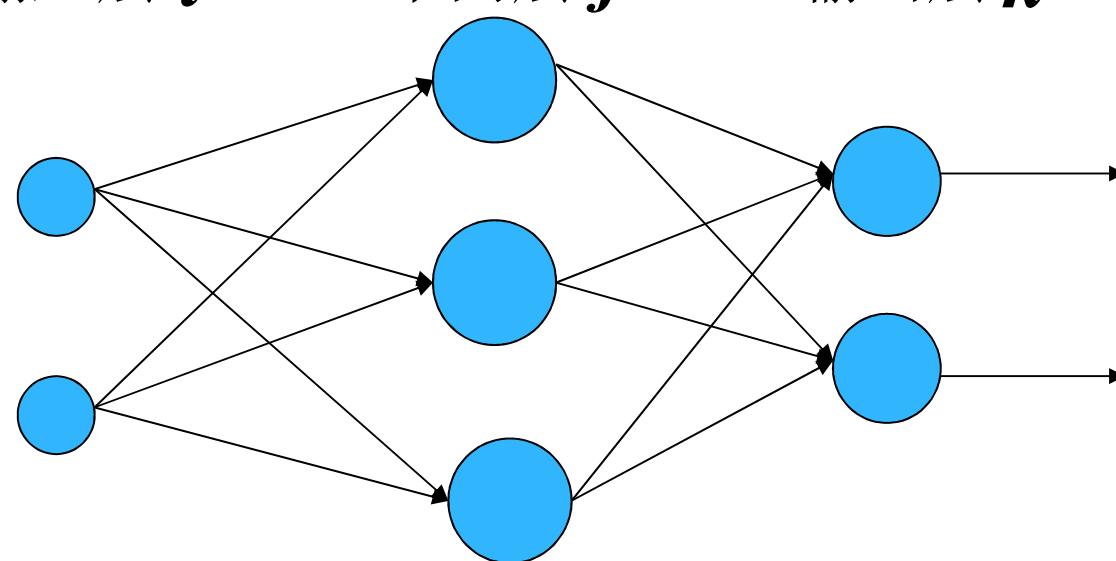
- \* 医学：疾病识别
- \* 图像：识别、去噪、增强、配准、融合
- \* 金融：股票和有价证券的预测分析、资本收益的预测和分析、风险管理、信用评估等等

# BP(Back-Propagation)神经网络

- \* BP 神经网络是一种按误差逆传播算法训练的多层前馈网络，是目前应用最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，而无需事前揭示描述这种映射关系的数学方程。
- \* 它的学习规则是使用最速下降法，通过反向传播来不断调整网络的权值和阈值，使网络的误差平方和最小。
- \* BP 神经网络模型拓扑结构包括输入层(input layer)隐层(hidden layer)和输出层(output layer)

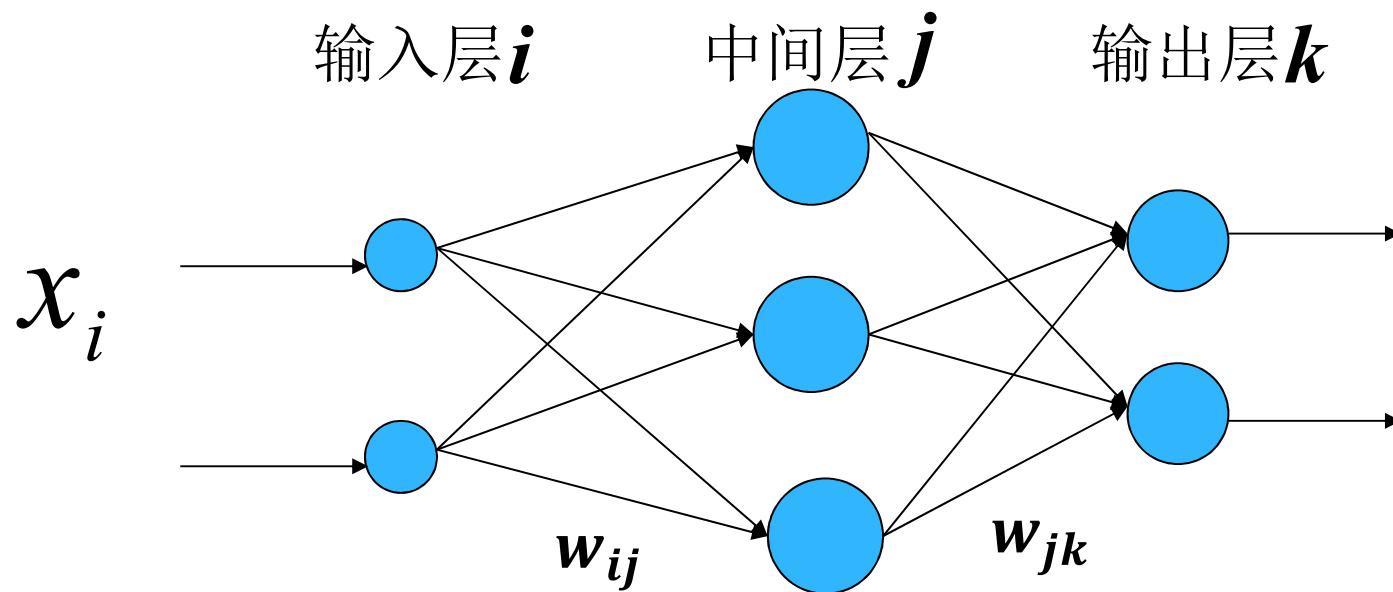
# BP神经网络

- \* 特点：多层前馈神经网络，信号向前传播，误差向后传播。
- \* 结构：输入层  $i$  中间层  $j$  输出层  $k$



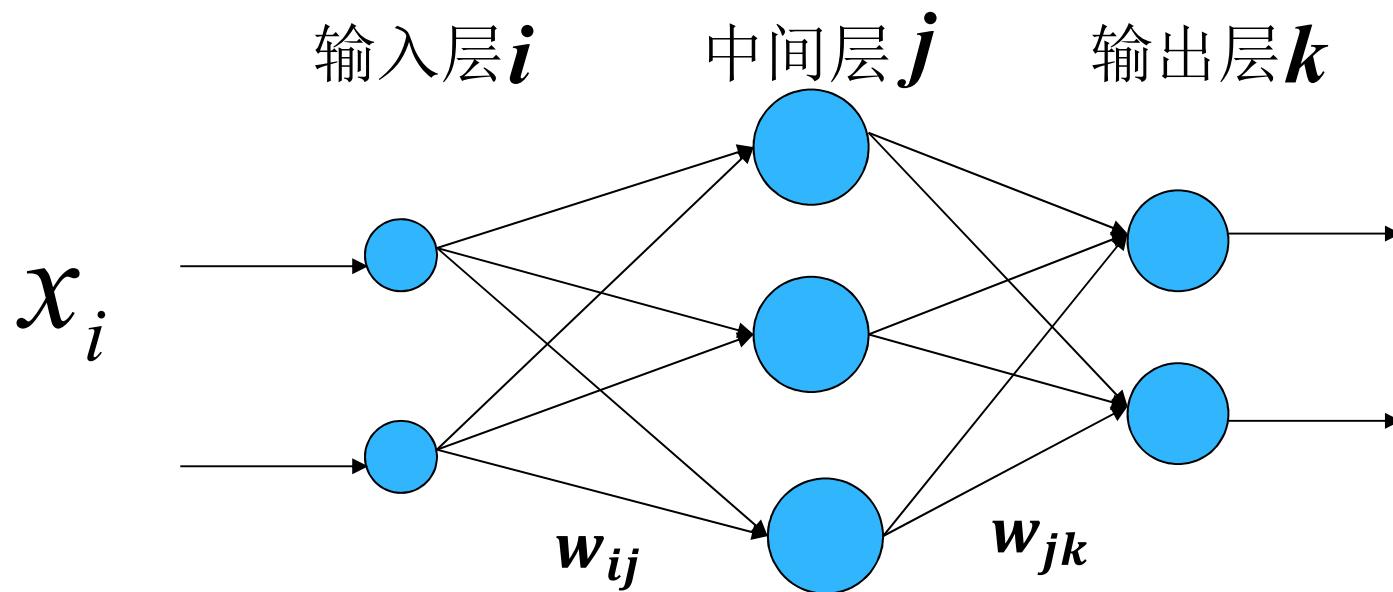
# BP神经网络

- \* 中间层输入:  $u_j = \sum_{i=1}^p x_i w_{ij}$
- \* 中间层输出:  $v_j = f(u_j)$



# BP神经网络

- \* 输出层输入:  $u_k = \sum_{j=1}^n v_j w_{jk}$
- \* 输出层输出:  $v_k = f(u_k)$



# BP神经网络 - 传递函数

\* 阈值型:

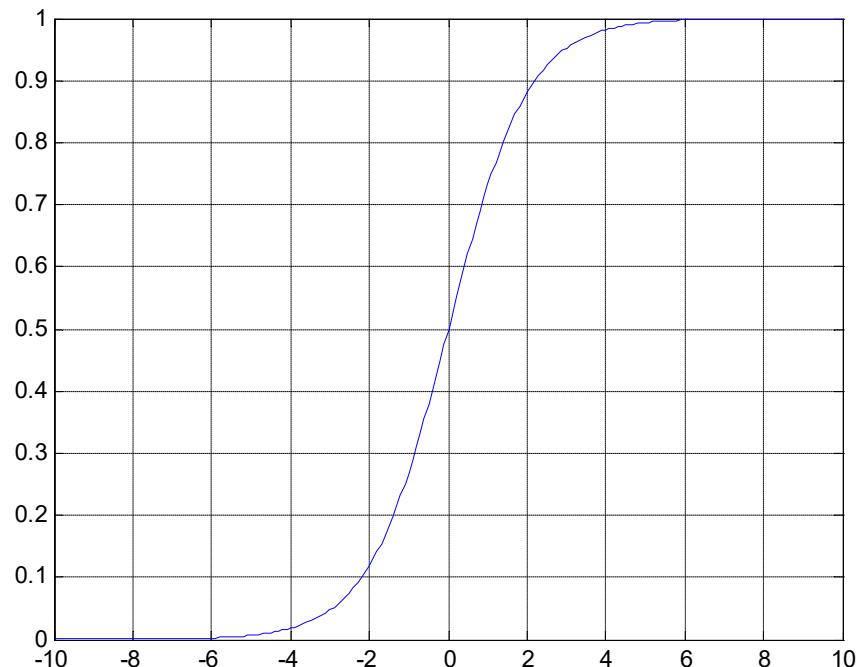
$$f(x) = \begin{cases} 1 & x_i \geq 0 \\ 0 & x_i < 0 \end{cases}$$

\* 线性型:

$$f(x_i) = \begin{cases} 1 & x_i \geq x_2 \\ ax_i + b & x_1 \leq x_i < x_2 \\ 0 & x_i < x_1 \end{cases}$$

# BP神经网络 - 传递函数

\* S型:  $f(x_i) = \frac{1}{1 + e^{\left(\frac{-x_i}{c}\right)^2}}$



# BP神经网络 - 传递函数

- \* **tansig**(双曲正切S型传递函数):  $f(x) = \frac{2}{1+e^{-2x}} - 1$
- \* 调用格式: **A=tansig(n)**
- \* 例:

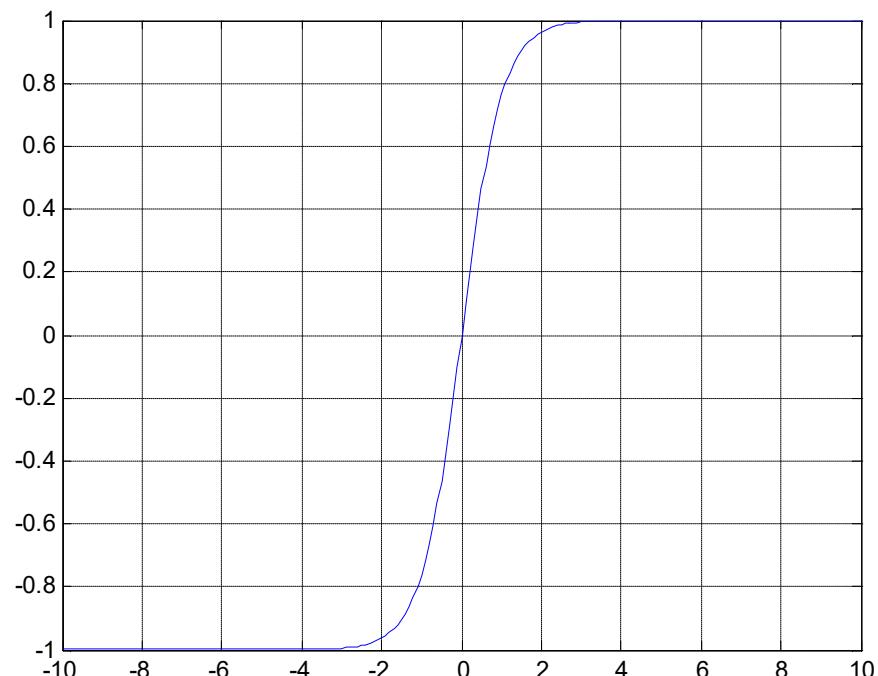
**n=-10:0.1:10**

**a=tansig(n)**

**plot(n,a)**

**grid on**

结果如右图所示

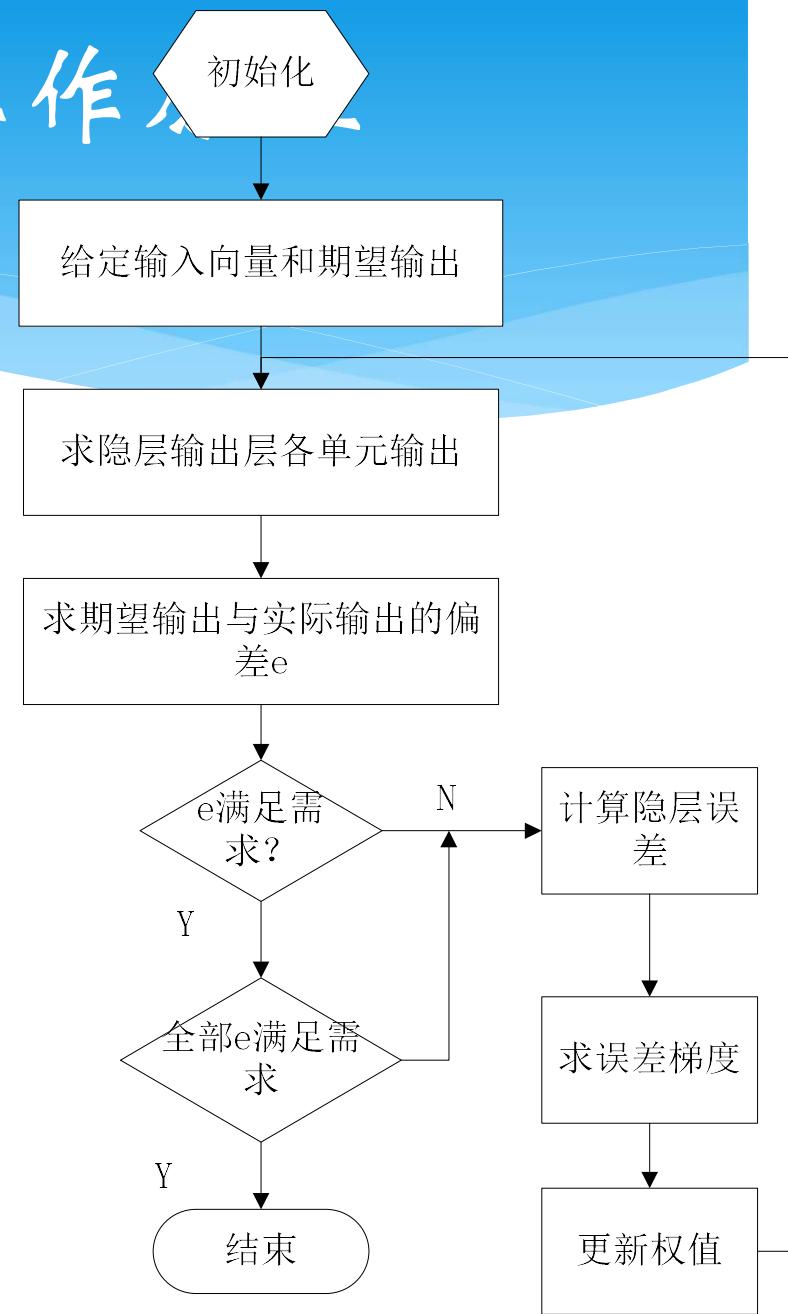
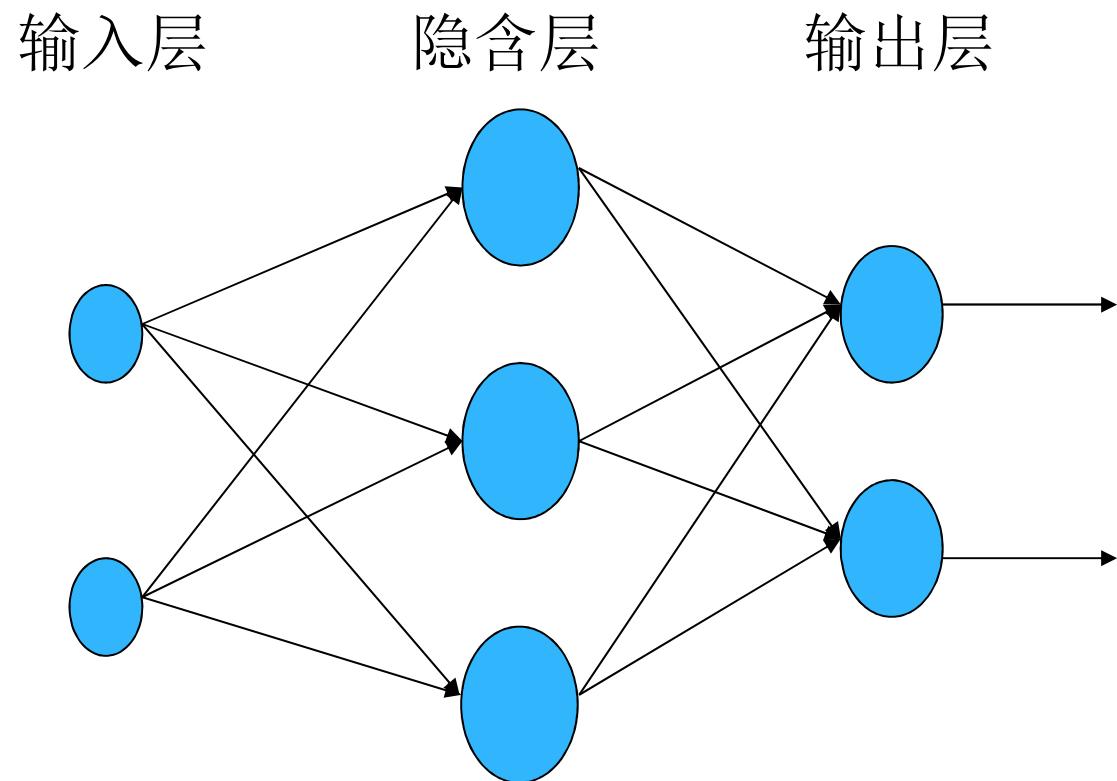


# BP神经网络工作原理

- \* 基本BP算法包括两个方面：信号的前向传播和误差的反向传播。即计算实际输出时按从输入到输出的方向进行，而权值和阈值的修正从输出到输入的方向进行。利用输出后的误差来估计输出层的直接前一层的误差，再用这个误差估计更前一层的误差，如此一层一层的反传下去，就获得了所有其他各层的误差估计。

# BP神经网络工作流程

\* 误差向后传播



# BP神经网络工作流程

- \* 1. 网络初始化：根据训练数据确定网络的输入神经元数n，隐含神经元数l，输出神经元数m，初始化各层神经元之间的连接权值 $w_{ij}$ 和 $w_{jk}$ ，初始化隐含层和输出层的阈值a和b，给定学习速率和神经元传递函数。
- \* 2. 隐含层输出计算：根据输入向量X，输入层和隐含层连接权值 $w_{ij}$ 以及隐含层阈值a，计算隐含层输出H

# BP神经网络工作流程

- \* 3. 输出层输出计算：根据隐含层输出的 $H$ ,连接权值 $w_{ij}$ 和阈值 $b$ ，计算BP神经网络的预测输出 $O$ .
- \* 4. 误差计算：根据预测输出 $O$ 和期望输出 $Y$ 计算网络预测误差 $e$
- \* 5. 权值更新：根据网络误差更新网络权值 $w_{ij}$ ， $w_{jk}$

# BP神经网络工作流程

- \* 6. 阈值更新：根据网络误差网络 $e$ ，神经元阈值 $a, b$
- \* 7. 判断算法迭代是否结束，若没有结束返回步骤2

# BP网络的标准学习算法-学习过程

- \* 正向传播：
  - \* 输入样本——输入层——各隐层——输出层
- \* 判断是否转入反向传播阶段：
  - \* 若输出层的实际输出与期望的输出（教师信号）不符
  - \* 误差反传
  - \* 误差以某种形式在各层表示——修正各层单元的权值
- \* 网络输出的误差减少到可接受的程度或者进行到预先设定的学习次数为止

# BP标准算法步骤

- \* 网络结构
  - \* 输入层有n个神经元， 隐含层有p个神经元， 输出层有q个神经元
- \* 变量定义
  - \* 输入向量：  $x = (x_1, x_2, \dots, x_n)$
  - \* 隐含层输入向量：  $hi = (hi_1, hi_2, \dots, hi_p)$
  - \* 隐含层输出向量：  $ho = (ho_1, ho_2, \dots, ho_p)$
  - \* 输出层输入向量：  $yi = (yi_1, yi_2, \dots, yi_q)$
  - \* 输出层输出向量：  $yo = (yo_1, yo_2, \dots, yo_q)$
  - \* 期望输出向量：  $d_o = (d_1, d_2, \dots, d_q)$

# BP标准算法步骤

- \* 变量定义
  - \* 输入层与中间层的连接权值:  $w_{ih}$
  - \* 隐含层与输出层的连接权值:  $w_{ho}$
  - \* 隐含层各神经元的阈值:  $b_h$
  - \* 输出层各神经元的阈值:  $b_o$
  - \* 样本数据个数:  $k = 1, 2, \dots, m$
  - \* 激活函数:  $f(\cdot)$
  - \* 误差函数:  $e = \frac{1}{2} \sum_{o=1}^q (d_o(k) - y_{o_o}(k))^2$

# BP标准算法步骤

- \* 1. 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 和 最大学习次数 $M$ 。

- \* 2. 随机选取一个输入样本及对应期望输出：

$$x(k) = (x_1(k), x_2(k), \dots, x_n(k))$$

$$d_o(k) = (d_1(k), d_2(k), \dots, d_q(k))$$

# BP标准算法步骤

- \* 3. 计算隐含层和输出层各神经元的输入和输出：

$$hi_h(k) = \sum_{i=1}^n w_{ih} x_i(k) - b_h \quad h = 1, 2, \dots, p$$

$$ho_h(k) = f(hi_h(k)) \quad h = 1, 2, \dots, p$$

$$yi_o(k) = \sum_{h=1}^p w_{ho} ho_h(k) - b_o \quad o = 1, 2, \dots, q$$

$$yo_o(k) = f(yi_o(k)) \quad o = 1, 2, \dots, q$$

# BP标准算法步骤

- \* 4. 利用网络期望输出和实际输出，计算误差函数对输出层的各神经元的偏导数。

$$\frac{\partial e}{\partial w_{ho}} = \frac{\partial e}{\partial y_i_o} \frac{\partial y_i_o}{\partial w_{ho}} \quad \frac{\partial y_i_o(k)}{\partial w_{ho}} = \frac{\partial (\sum_h^p w_{ho} h_o(k) - b_o)}{\partial w_{ho}} = h_o(k)$$

$$\frac{\partial e}{\partial y_i_o} = \frac{\partial (\frac{1}{2} \sum_{o=1}^q (d_o(k) - y_o(k))^2)}{\partial y_i_o} = -(d_o(k) - y_o(k)) y'_o(k)$$

$$= -(d_o(k) - y_o(k)) f'(y_o(k)) \square -\delta_o(k)$$

# BP标准算法步骤

- \* 5. 利用隐含层到输出层的连接权值、输出层的输出和隐含层的输出，计算误差函数对隐含层各神经元的偏导数。

$$\frac{\partial e}{\partial w_{ho}} = \frac{\partial e}{\partial y_i} \frac{\partial y_i}{\partial w_{ho}} = -\delta_o(k) h_o(k)$$

$$\frac{\partial e}{\partial w_{ih}} = \frac{\partial e}{\partial h_i} \frac{\partial h_i}{\partial w_{ih}}$$

$$\frac{\partial h_i}{\partial w_{ih}} = \frac{\partial (\sum_{i=1}^n w_{ih} x_i(k) - b_h)}{\partial w_{ih}} = x_i(k)$$

# BP标准算法步骤

$$\begin{aligned}\frac{\partial e}{\partial h_i(k)} &= \frac{\partial(\frac{1}{2} \sum_{o=1}^q (d_o(k) - y_{o_o}(k))^2)}{\partial h_o(k)} \frac{\partial h_o(k)}{\partial h_i(k)} \\ &= \frac{\partial(\frac{1}{2} \sum_{o=1}^q (d_o(k) - f(y_{i_o}(k)))^2)}{\partial h_o(k)} \frac{\partial h_o(k)}{\partial h_i(k)} \\ &= \frac{\partial(\frac{1}{2} \sum_{o=1}^q ((d_o(k) - f(\sum_{h=1}^p w_{ho} h_o(k) - b_o))^2))}{\partial h_o(k)} \frac{\partial h_o(k)}{\partial h_i(k)} \\ &= -\sum_{o=1}^q (d_o(k) - y_{o_o}(k)) f'(y_{i_o}(k)) w_{ho} \frac{\partial h_o(k)}{\partial h_i(k)} \\ &= -(\sum_{o=1}^q \delta_o(k) w_{ho}) f'(h_i(k)) \square -\delta_h(k)\end{aligned}$$

# BP标准算法步骤

- \* 6. 利用输出层各神经元的和隐含层各神经元的输出来修正连接权值。

$$\Delta w_{ho}(k) = -\mu \frac{\partial e}{\partial w_{ho}} = \mu \delta_o(k) h o_h(k)$$

$$w_{ho}^{N+1} = w_{ho}^N + \eta \delta_o(k) h o_h(k)$$

# BP标准算法步骤

- \* 7. 利用隐含层各神经元的和输入层各神经元的输入修正连接权。

$$\Delta w_{ih}(k) = -\mu \frac{\partial e}{\partial w_{ih}} = -\mu \frac{\partial e}{\partial h_i(k)} \frac{\partial h_i(k)}{\partial w_{ih}} = \delta_h(k) x_i(k)$$

$$w_{ih}^{N+1} = w_{ih}^N + \eta \delta_h(k) x_i(k)$$

# BP标准算法步骤

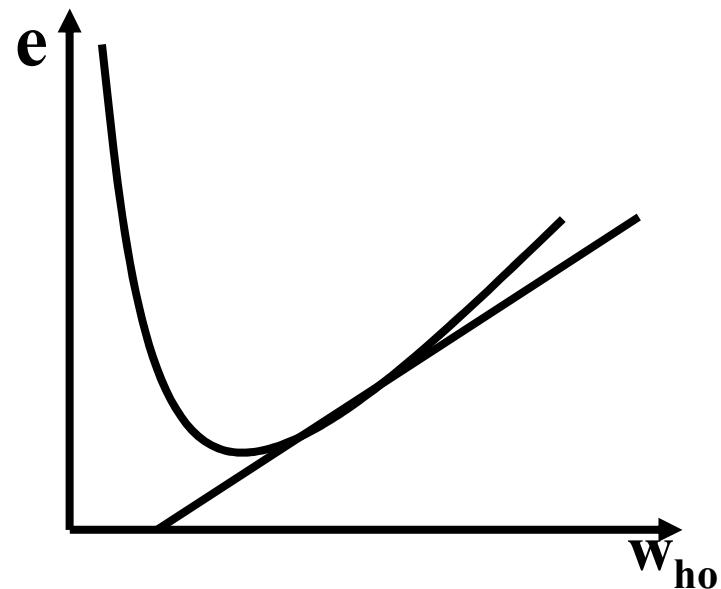
- \* 8. 计算全局误差

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

- \* 9. 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，返回到第三步，进入下一轮学习。

# BP标准算法直观解释

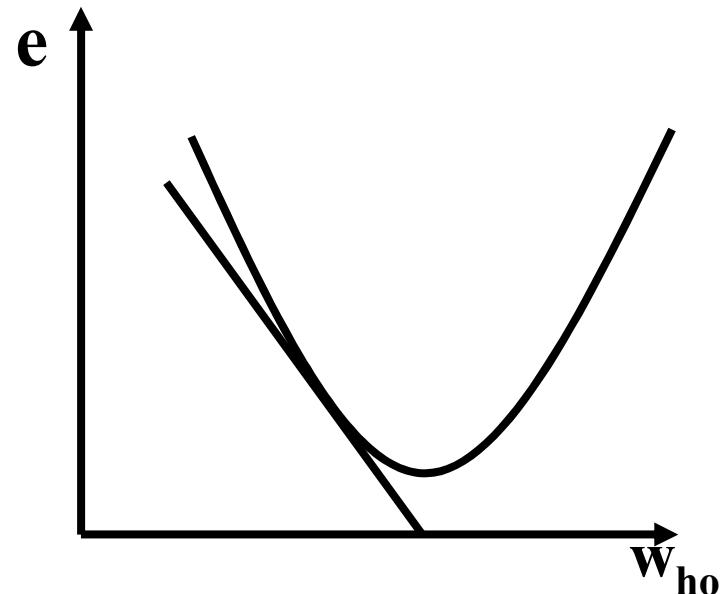
- \* 情况一、直观表达  
当误差对权值的偏导数大于零时，权值调整量为负，实际输出大于期望输出，权值向减少方向调整，使得实际输出与期望输出的差减少。



$$\frac{\partial e}{\partial w_{ho}} > 0, \text{ 此时 } \Delta w_{ho} < 0$$

# BP标准算法直观解释

\* 情况二、直观表达  
当误差对权值的偏导数小于零时，权值调整量为正，实际输出少于期望输出，权值向增大方向调整，使得实际输出与期望输出的差减少。



$$\frac{\partial e}{\partial w_{ho}} < 0, \text{ 此时 } \Delta w_{ho} > 0$$

# 应用实例

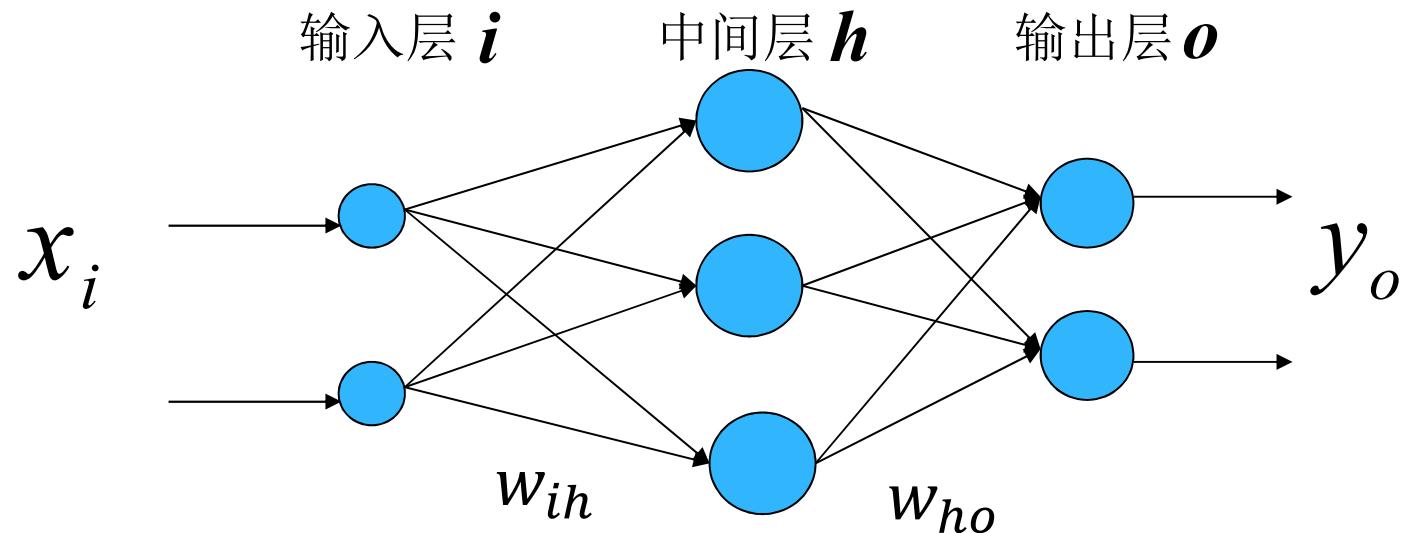
- \* 蠼虫的分类问题：
- \* 生物学家试图对两种蠼虫（Af & Apf）进行分类鉴别，依据的资料是触角和翅膀的长度。
- \* 怎么基于BP神经网络进行分类？

# 应用实例

- \* 思路：
- \* 1、触角长和翼长作为输入信息，分别记 $x_1, x_2$   
目标输出：(1,0)、(0,1)。  
 $A_f$ 类记为(1,0)， $A_{pf}$ 类记为(0,1)。  
输出层有两个神经元

$$y_{o_1} = 0 \text{或} 1 \quad y_{o_2} = 0 \text{或} 1$$

# 应用实例



- \* 2、通过已知样本训练出合适的权值 $w_{ih}$ 和 $w_{ho}$ ，使输出为(0,1)或(1,0)。
- \* 3、将待区分的蠓虫数据输入网络，求值。

# 应用实例

\* 权值求法：向后传播法

理想输出 Af类(1,0), Apf类(0,1)记为  $d_o(k)$

则有误差：

$$E(w) = \frac{1}{2} \sum_{o=1}^q (d_o(k) - y_o(k))^2$$

使得  $E(w)$  最小的  $w_{ih}$  和  $w_{ho}$  作为所需的权值

# BP神经网络的改进

- \* 基本BP算法的缺陷
- \* 隐含层的层数与节点数的改进

# 基本BP算法的缺陷

- \* BP算法因其简单、易行、计算量小、并行性强等优点，是传统神经网络训练采用最多也是最成熟的训练算法之一。
- \* 其算法的实质是求解误差函数的最小值问题，由于它采用非线性规划中的最速下降方法，按误差函数的负梯度方向修改权值，因而通常存在以下问题：
  - \* 学习效率低，收敛速度慢
  - \* 易陷入局部极小状态

# 隐含层的层数与节点数的改进

- \* BP神经网络的构建必须注意隐含层神经元数的选择：
  - \* 如果隐含层神经元数过少，BP神经网络难以建立复杂的映射关系，网络预测误差较大，
  - \* 如果隐含层神经元数过多，网络学习的时间增加，并且可能出现过拟合的现象，就是把样本中非规律性的内容（如噪声等）也学会记忆，从而出现训练样本准确，但是其他样本预测误差较大。

# 隐含层的层数与节点数的改进

- \* 如何确定隐含层的神经元个数?
- \* 目前没有什么成熟的理论能够确定各层神经元的神经元个数和含有几层网络，大多数还是靠经验

# 隐含层的层数与节点数的改进

\* 最佳隐含层神经元数选择可参考如下经验公式：

$m$ 为隐含层神经元数，

$n$ 为输入层神经元数， $l$ 为输出层神经元数

$a$ 为1—10之间的常数

$$m = \sqrt{n + l} + a \quad m = \log_2 n$$

$$m = \sqrt{nl} \quad \dots\dots$$

# 隐含层的层数与节点数的改进

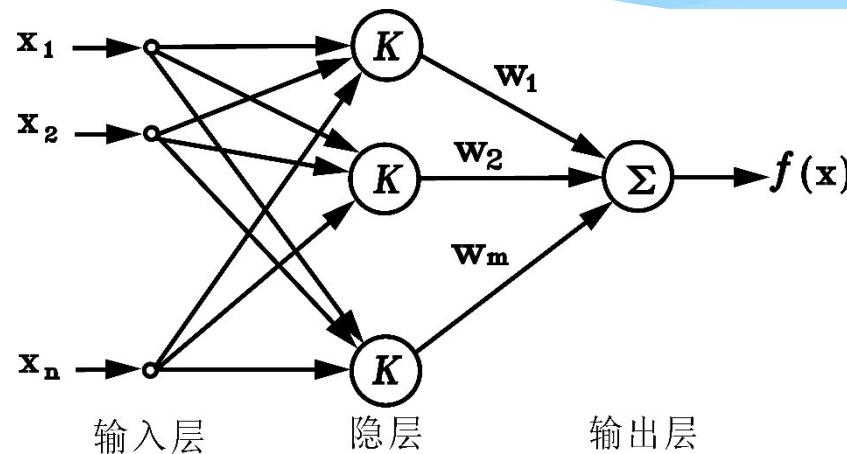
- \* 实际问题中，隐含层神经元数的选择首先是参考公式来确定神经元的大致范围，然后试凑法确定最佳点数
- \* 注意对于一般问题BP神经网络的分类误差随着隐含层神经元的增加呈现先减少后增加的趋势

# 其他常用传统神经网络模型

- \* 径向基函数网络 (**RBF**)
- \* **Hopfield**网络
- \* 自适应共振理论神经网络 (**ART**)
- \* 自组织特征映射神经网络 (**SOM**)

# 径向基函数网络 (RBF)

\* 基本结构：



\* 网络特点：只有一个隐层，隐层单元采用径向基函数作为其输出函数，输入层到隐层之间的权值均固定为1；输出节点为线性求和单元，隐层到输出节点之间的权值可调，因此，输出为隐层的加权求和。

# 径向基函数网络 (RBF)

- \* 径向基函数(Radial Basis Function, 简称RBF), 就是某种沿径向对称的标量函数。
- \* 通常定义为空间中任一点 $x$ 到某一中心 $x_c$ 之间欧氏距离的单调函数, 可记作 $k(||x-x_c||)$ , 其作用往往是局部的, 即当 $x$ 远离 $x_c$ 时函数取值很小。

# 径向基函数网络 (RBF)

\* 最常用的径向基函数是高斯核函数，形式为：

$$k(\|x - x_c\|) = \exp \left\{ -\frac{\|x - x_c\|^2}{2\sigma^2} \right\}$$

其中  $x_c$  为核函数中心， $\sigma$  为函数的宽度参数，控制了函数的径向作用范围。

# 径向基函数网络 (RBF)

- \* **RBF** 网络的作用：

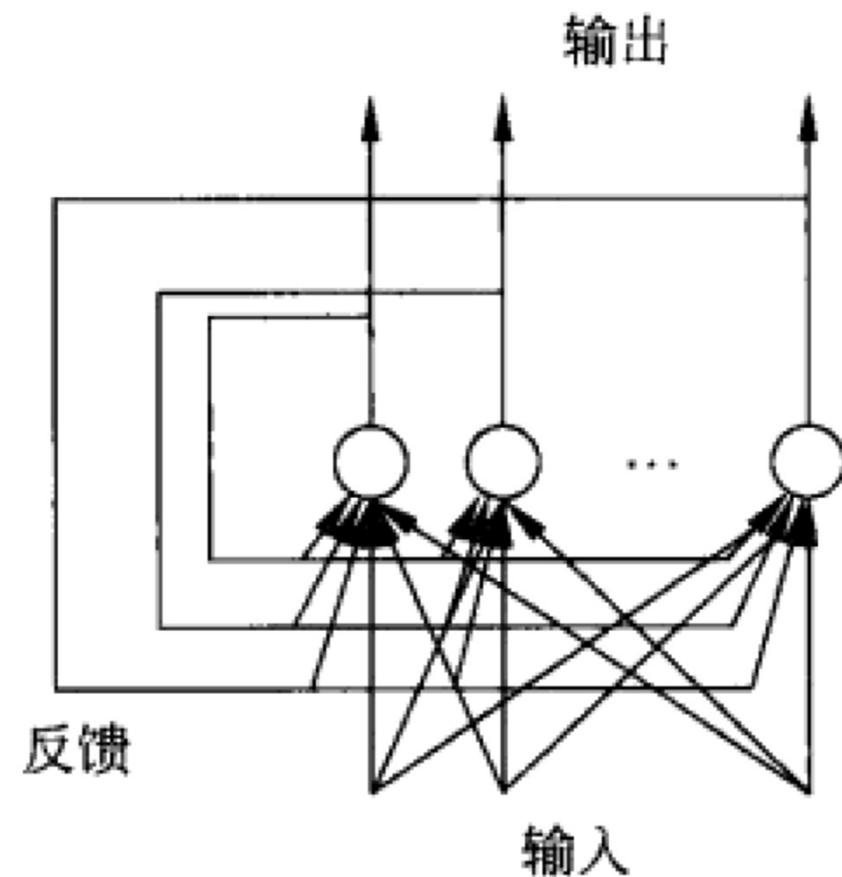
- \* 把网络看成对未知函数 $f(x)$ 的逼近器。一般任何函数都可表示成一组基函数的加权和，这相当于用隐层单元的输出函数构成一组基函数来逼近 $f(x)$ 。
- \* 在**RBF**网络中，从输入层到隐层的基函数输出是一种非线性映射，而输出则是线性的。这样，**RBF**网络可以看成是首先将原始的非线性可分的特征空间变换到另一空间(通常是高维空间)，通过这一变换使在新空间中线性可分，然后用一个线性单元来解决问题。

# 径向基函数网络 (RBF)

- \* 在典型的**RBF**网络中有3组可调参数：隐层基函数中心、方差，以及输出单元的权值。
- \* 用于模式识别问题的**RBF**网络在一定意义上等价于首先用非参数方法估计出概率密度，然后用它进行分类。

# Hopfield 网 络

\* Hopfield网络是一种反馈网络。反馈网络具有一般非线性系统的许多性质，如稳定性问题、各种类型的吸引子以及混沌现象等，在某些情况下还有随机性、不可预测性。因此，它比前馈网络的内容复杂。



# Hopfield 网 络

- \* Hopfield网络除了具有上述反馈网络的结构和性质之外，还满足以下条件：
- \* (1) 权值对称，即 $w_{ij} = w_{ji}$ ，权矩阵 $\mathbf{W} = \mathbf{W}^T$ ，为对称阵。
- \* (2) 无自反馈，即 $w_{ii}$ ，权矩阵 $\mathbf{W}$ 的对角线元素为零。

# 自适应共振理论神经网络(ART)

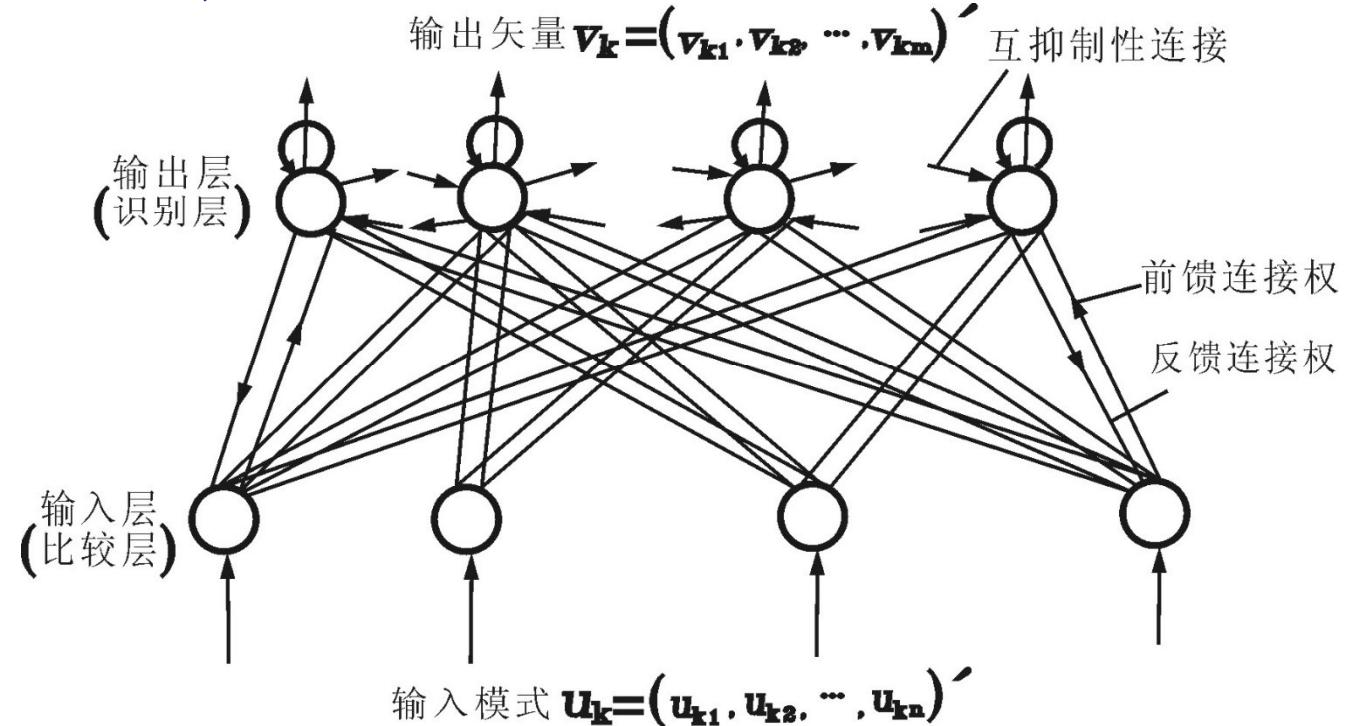
- \* 自适应共振理论神经网络既能模拟人脑的可塑性，可以学习知识，又能模拟人脑的稳定性，学习新的知识但不破坏原有知识，即这种网络不仅能记忆新的知识，而且还保留已记忆的内容。
- \* 这种网络主要依据自适应共振理论(**Adaptive Resonance Theory—ART**)，用生物神经细胞自兴奋与侧抑制的动力学原理指导学习，输出层各神经元竞争对输入模式的响应，在竞争中还可能采用侧抑制的方法，最后只有一个神经元获胜，获胜神经元则代表该输入模式的类别，权值的调整只在与获胜神经元关联的连接权中进行，通过网络双向连接权的记忆和比较，来完成对输入模式的记忆、回想，并以同样的方式实现模式的识别。

# 自适应共振理论神经网络 (ART)

- \* 当提供给网络的输入模式是一个网络已记忆的或与已记忆的模式十分相似的模式时，网络会把这个模式回想出来，并提供正确的分类；如果输入模式是网络不曾记忆的新模式，则网络将在不影响原有记忆的前提下，将这个模式记下来，并分配一个尚未使用过的输出层神经元作为这一记忆模式的记忆分类标志。

# ART网络的结构及原理

- \* 结构：ART网络主要有ART1和ART2两种模型，其主要区别是前者为二值输入，后者为模拟输入。这里主要介绍ART1模型。
- \* ART1如图所示



# ART网络的结构及原理

## \* 原理：

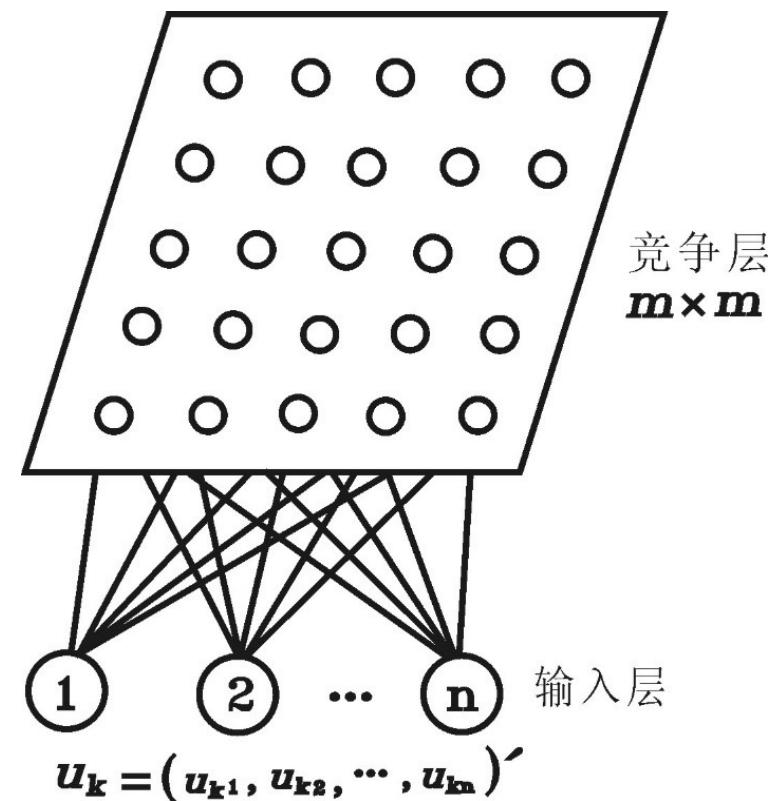
ART1网络的学习和工作是通过反复地将输入学习模式由输入层向输出层自下而上地短时记忆和由输出层向输入层自上而下地长期记忆和比较来实现的。当这种记忆和比较达到共振时，输出矢量可以正确地反映出输入学习模式的分类，且网络原有的记忆不受影响。至此，对一个输入模式的记忆和分类即告完成。

# 自组织特征映射神经网络 (SOM)

- \* 人脑的记忆不是神经元与记忆模式的一一对应，而是一群神经元对应一个模式。
- \* 生理实验表明，某一外界信息所引起的兴奋刺激并不只针对一个神经细胞，而是对以某一个神经元为中心的一个区域内各神经元的兴奋刺激，并且刺激强度以区域中心为最大，随着与中心距离的增大，强度逐渐减弱，远离区域中心的神经元反而受到抑制。
- \* 自组织特征映射(**Self-Organizing Feature Map, SOM**)人工神经网络可以很好地模拟人类大脑的功能区域性、自组织特性及神经元兴奋刺激规律。

# SOM网络模型及原理

- \* SOM网络是由Kohonen提出来的，其模型结构由输入层和输出竞争层组成，输入层与输出层之间是全互连的。有时输出层各神经元之间还有侧抑制连接。



# 自组织特征映射神经网络 (SOM)

- \* SOM能将任意维输入模式在输出层映射成一维或二维离散图形，并保持其拓扑结构不变。
- \* 在输出层，获胜的那个神经元 $g$ 的邻域 $N_g$ 内的神经元在不同程度上都得到兴奋，而在 $N_g$ 以外的神经元都被抑制。
- \* 这个 $N_g$ 可以是任意形状，但一般是对称的，如正方形，六边形。 $N_g$ 是时间的函数，用 $N_g(t)$ 表示，随时间 $t$ 增大， $N_g(t)$ 减小，最后可能剩下下一个神经元，也可能是一组神经元。
- \* 最终得到的区域反映了一类输入模式的属性。

# 深度神经网络？

谢谢