几种典型的人工网络

引言

- 感知器
- 径向基函数神经网络
- Hopfield神经网络

感知器

1. 感知器模型和用途
2. 多层感知器模型和学习算法
3. 多层感知器应用实例

一、感知器模型和用途

❖ 1958年，美国心理学家Rosenblatt提出一种具有单层计算单元的神经网络，称为Preceptron，即感知器。

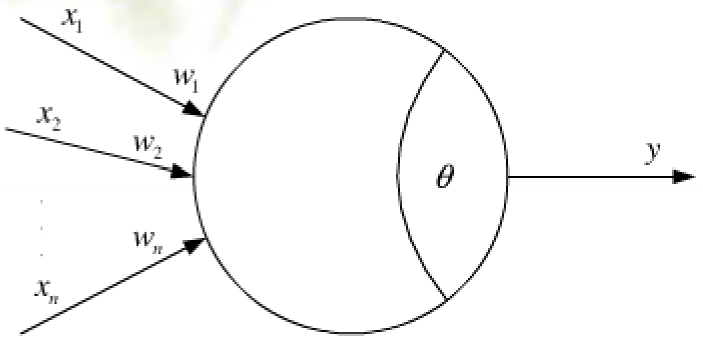
❖感知器模拟人的视觉接受环境信息，并由神经冲动进行信息传递的层次型神经网络。

❖感知器研究中首次提出了自组织、自学习的思想，而且对所能解决的问题存在着收敛算法，并能从数学上严格证明，因而对神经网络研究起了重要推动作用。

❖单层感知器的结构与功能都非常简单，以至于在解决实际问题时很少采用，但研究中具有重要意义，是研究其它网络的基础。

一、感知器模型和用途

1、结构和数学模型：



向量形式：

$\theta = -w_0$ $x_0 = 1$

$x = [x_0, x_1, \cdots, x_n]^T$ $w = [w_0, w_1, \cdots, w_n]^T$

神经元输入： $Net = \sum_{i=1}^{n} w_i x_i - \theta$

神经元输出： $y = \begin{cases} 1 & Net > 0 \\ 0 & Net \le 0 \end{cases}$

$Net = w^T x$

$y = \begin{cases} 1 & Net > 0 \\ 0 & Net \le 0 \end{cases}$

一、感知器模型和用途

1、结构和数学模型（续）：

神经元输入： $Net=\sum_{i=1}^{n}w_ix_i-\theta$

神经元输出： $y=\begin{cases}1 & Net>0\\ 0 & Net\le 0\end{cases}$

（1）原始感知器模型采用阈值函数类型的激励函数。

（2）现在常用S型函数作为激励函数。

$$y=\frac{1}{1+e^{-\beta\cdot Net}}$$

（3）针对阈值函数讨论其用途。

一、感知器模型和用途

2、两种用途：

（1）模式识别器（分类器）

**解决只有2类模式的识别问题。

**只能识别具有线性边界的识别问题。

$x \in R^n \qquad y \in \{0,1\}$

$w^T x \leq 0$ y=0，x属于第一类

$w^T x > 0$ y=1，x属于第二类

一、感知器模型和用途

2、两种用途（续）：

（2）逻辑函数

**二值逻辑元

**能实现布尔代数的某些基本运算，如：与、或、非

**不能实现布尔代数的全部运算，如：异或

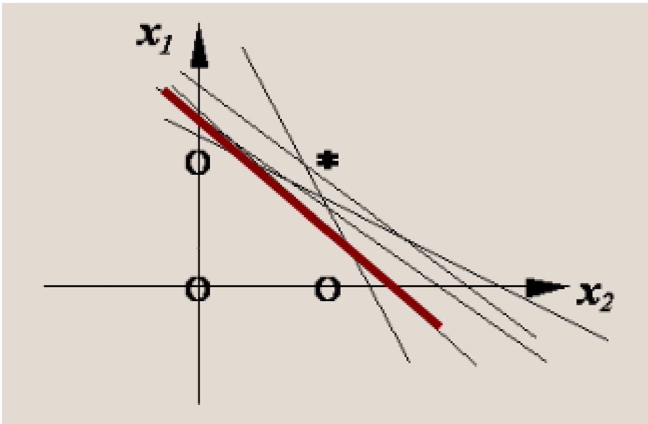
$$Y = f\left(w_1 x_1 + w_2 x_2 - \theta\right)$$

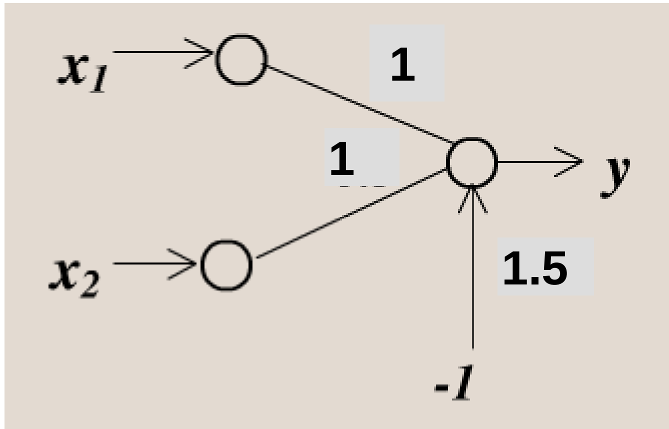
一、感知器模型和用途

✓"与"运算

当取 $w_1 = w_2 = 1.0$， $\theta = 1.5$ 时，

完成逻辑"与"的运算。





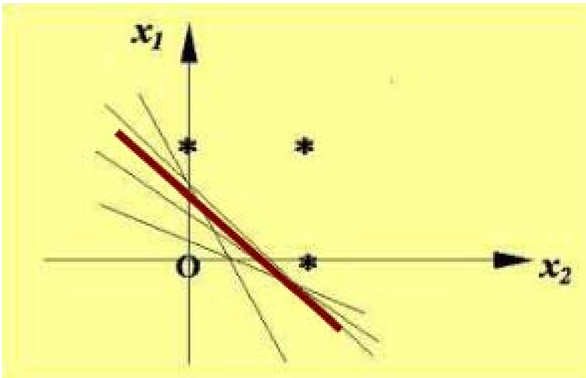
$$Y = f\left(x_1 + x_2 - 1.5\right)$$

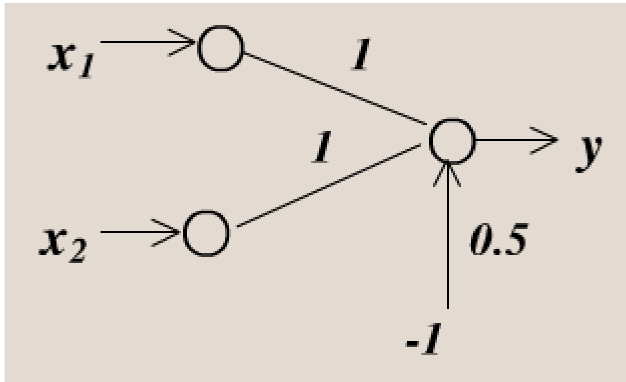
一、感知器模型和用途

✓"或"运算

当取 $w_1 = w_2 = 1.0$，$\theta = 0.5$ 时，

完成逻辑"或"的运算。





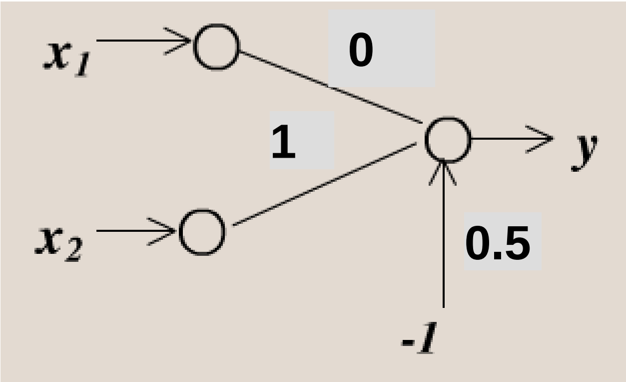
$$Y = f\left(x_1 + x_2 - 0.5\right)$$

一、感知器模型和用途

✓“非”运算

当取 $w_1 = 0,\quad w_2 = 1,\quad \theta = 0.5$ 时，

完成逻辑“非”的运算。



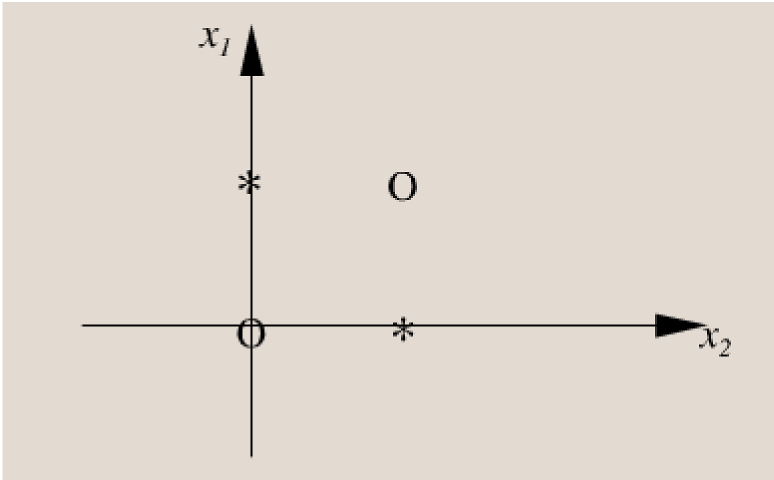
$$Y = f\left(x_2 - 0.5\right)$$

一、感知器的局限性

✓"异或"运算

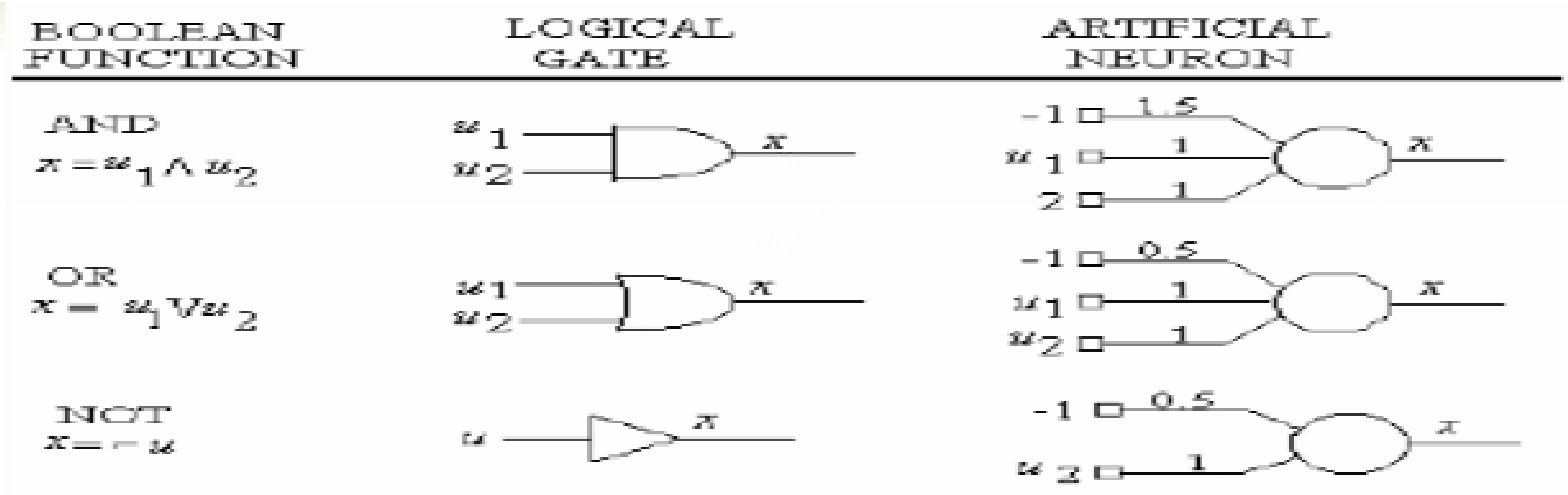
"异或" 的真值表

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



一、感知器小结

- 通过适当的选择权重，单层感知机能够实现and，or，not布尔运算。



- 单层感知机找不到相应的权重来实现XOR逻辑。
- 单层感知机不具备非线性分类的能力。

二、多层感知器

❖ **单计算层感知器的局限性**

只能解决线形可分的问题，而大量的分类问题是线性不可分的。

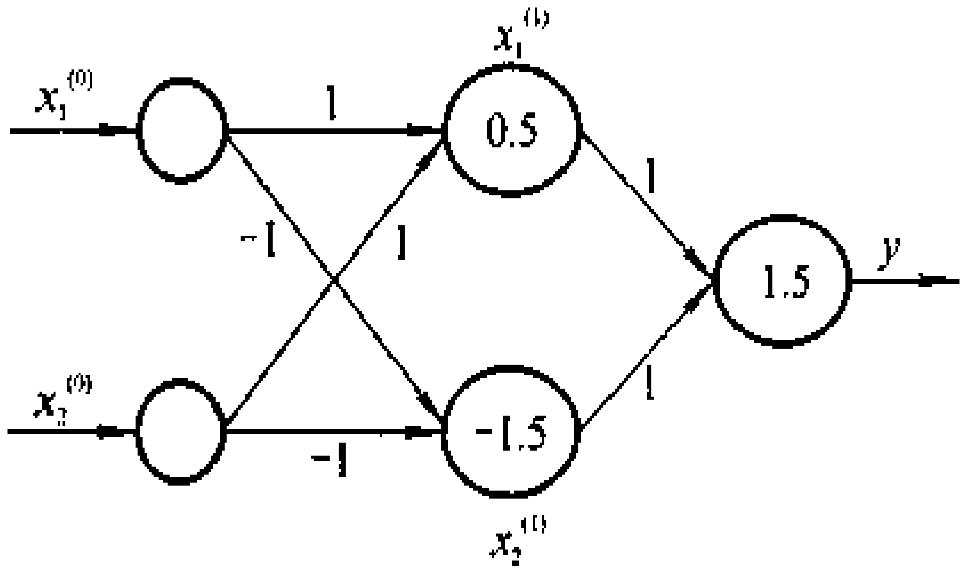
❖ **解决的有效方法**

➢ 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。

二、多层感知器

❖ “异或”问题



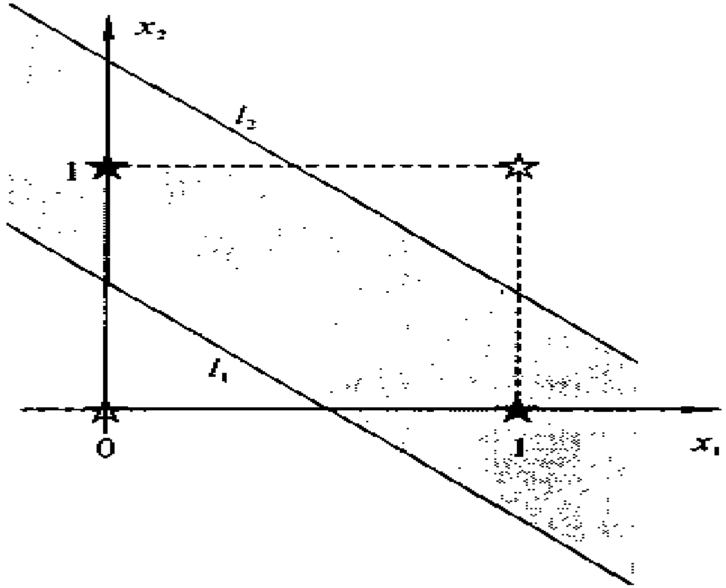
$$x_1^{(1)} = f\left(1 \cdot x_1^{(0)} + 1 \cdot x_2^{(0)} - 0.5\right),$$

$$x_2^{(1)} = f\left((-1) \cdot x_1^{(0)} + (-1) \cdot x_2^{(0)} - (-1.5)\right)$$

$$y = f\left(1 \cdot x_1^{(1)} + 1 \cdot x_2^{(1)} - 1.5\right)。$$

二、多层感知器

❖ “异或”问题（续）



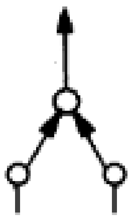
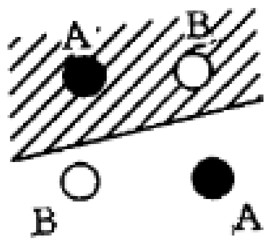
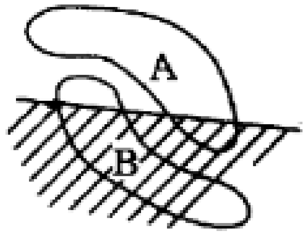
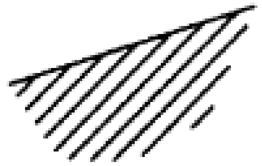
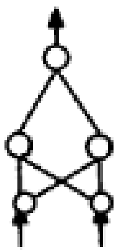
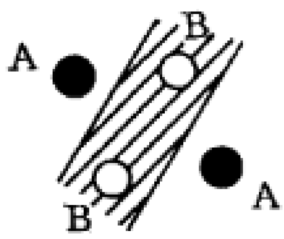
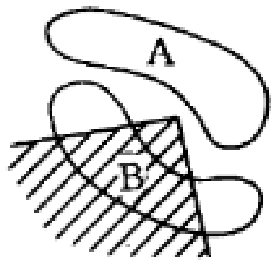
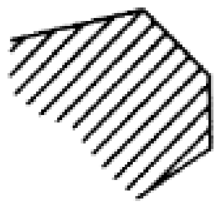
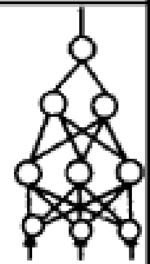
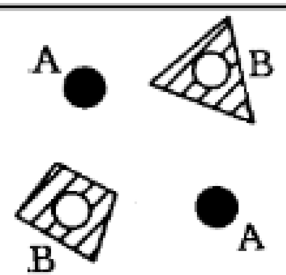
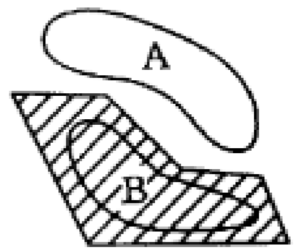
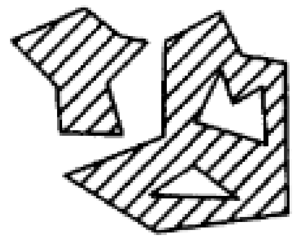
在模式空间中用两个超平面去划分样本，即用两条直线：

x1+x2=0.5

x1+x2=1.5

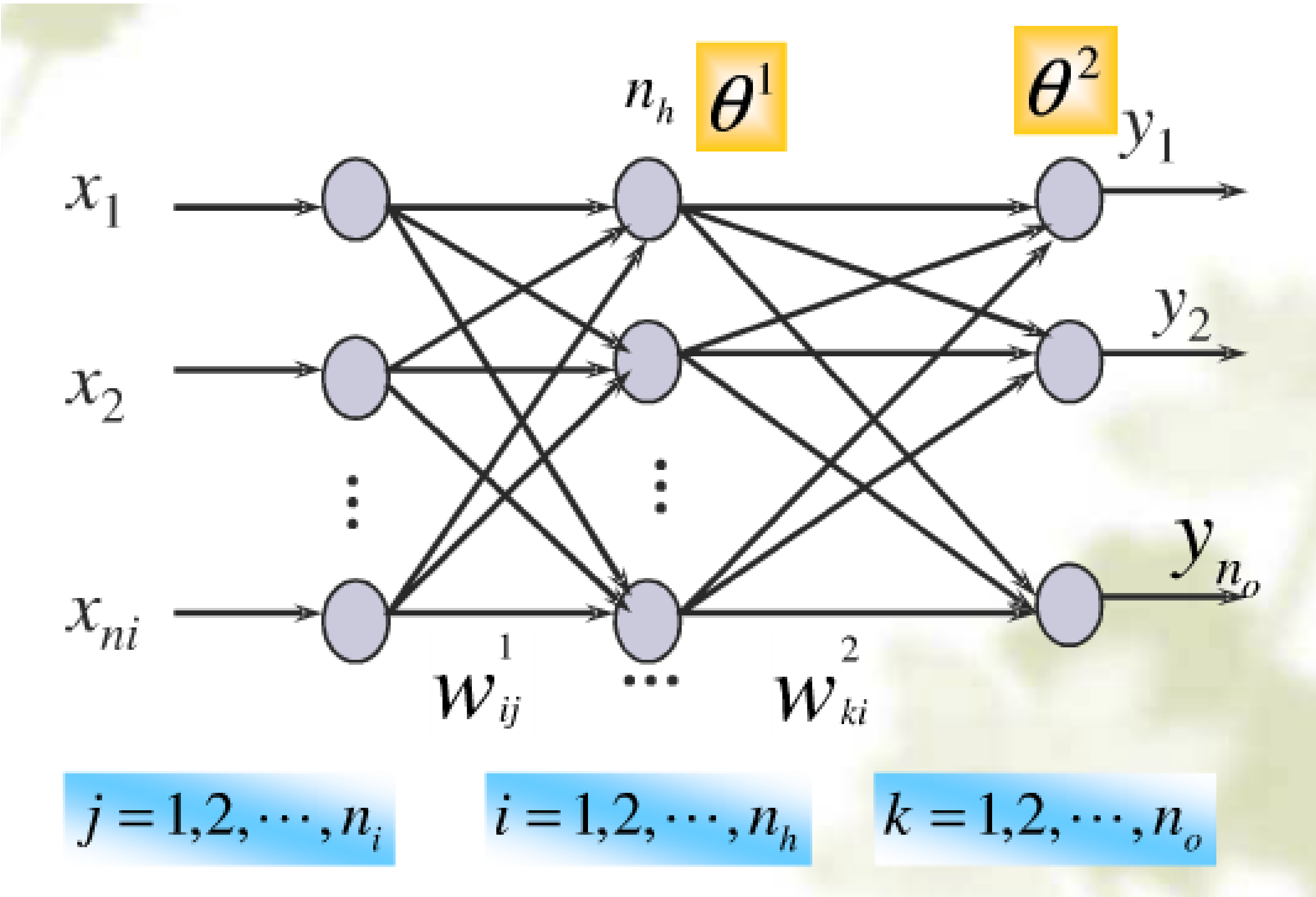
二、多层感知器

❖ **具有不同隐层数的感知器分类能力**

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

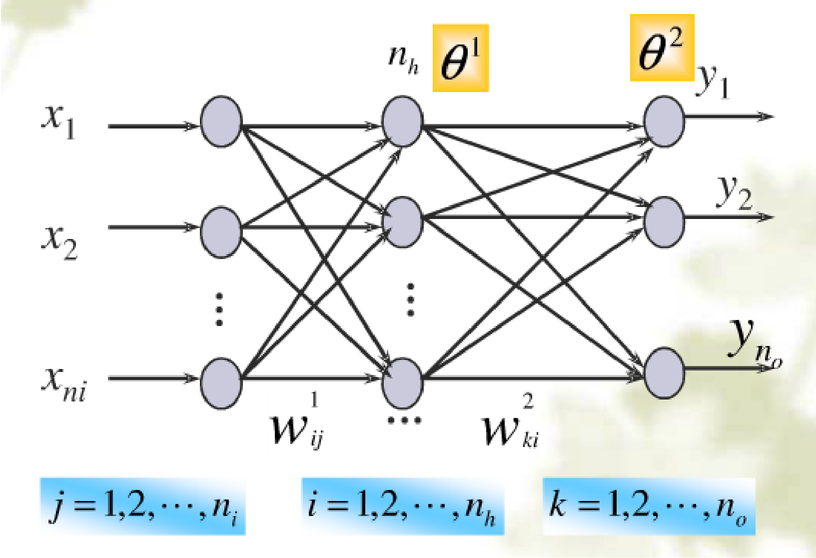
二、多层感知器模型和学习算法

❖ 含一个隐层的感知器模型



二、多层感知器模型和学习算法

❖ 数学描述



➢ 隐含层输出

$$o_i=\rho\left(\sum_{j=1}^{n_i} w_{ij}^1 x_j+\theta_i^1\right)$$

$$O=\rho\left(W^1X+\theta^1\right)$$

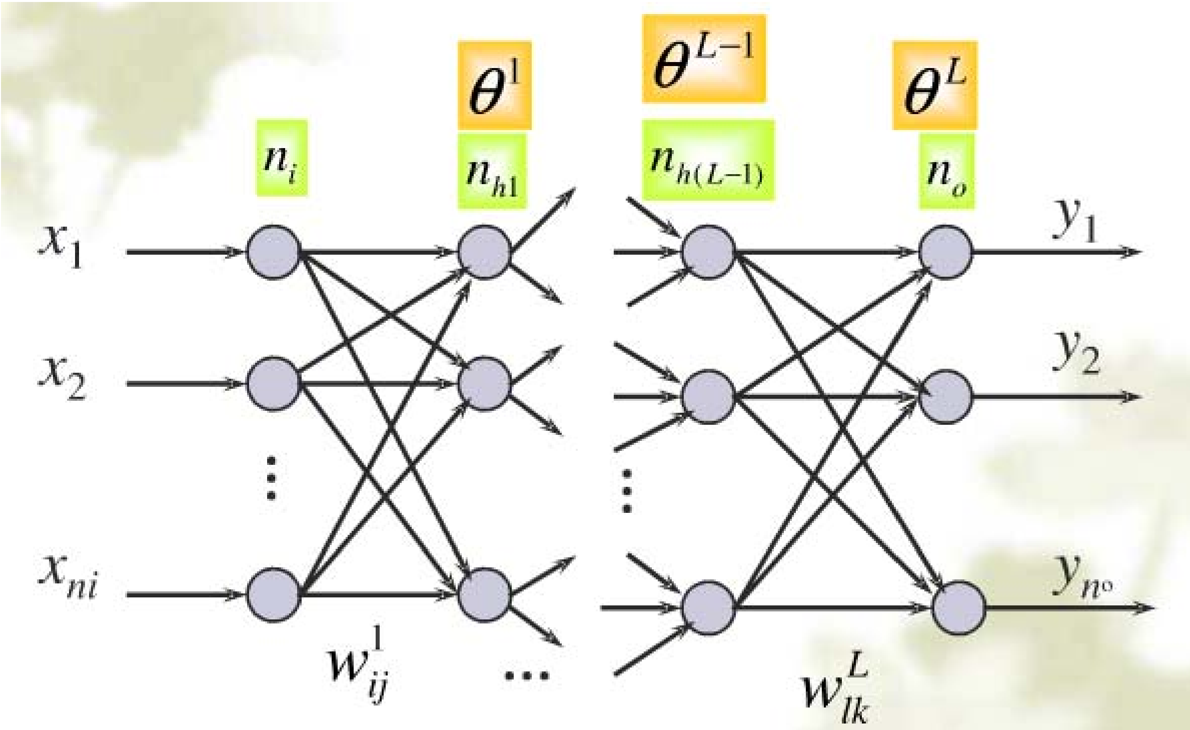
➢ 输出层输出

$$y_k=\sigma\left(\sum_{i=1}^{n_h} w_{ki}^2 o_i+\theta_k^2\right)$$

$$\begin{aligned}Y&=\sigma\left(W^2O+\theta^2\right)=\sigma\left\{W^2\rho\left(W^1X+\theta^1\right)+\theta^2\right\}\\&=\Gamma_2\left\{W^2\Gamma_1\left(W^1X+\theta^1\right)+\theta^2\right\}\end{aligned}$$

二、多层感知器模型和学习算法

❖ **L+1层感知器模型**



$$Y=\Gamma_L\left(W^L\Gamma_{L-1}\left\{W^{L-1}\Gamma_{L-2}\left[\cdots\left(\Gamma_1\left(W^1X+\theta^1\right)\right)+\cdots+\theta^{L-2}\right]+\theta^{L-1}\right\}+\theta^L\right)$$

二、多层感知器模型和学习算法

❖ 多层感知器模型的功能

✓ 实现任意的布尔函数。

✓在模式识别中，它能划分输入空间，生成复杂的边界。

✓能逼近从R^n到R^m的任意连续映射。

❖ 说明：

✓激励函数只要求连续、光滑、单增、上下有界的非线性函数即可。

✓为简化计算，输出层常采用线性神经元。

二、多层感知器模型和学习算法

❖ 学习算法

✓ 前向传播网络实质上表示的是一种从输入空间到输出空间的映射。

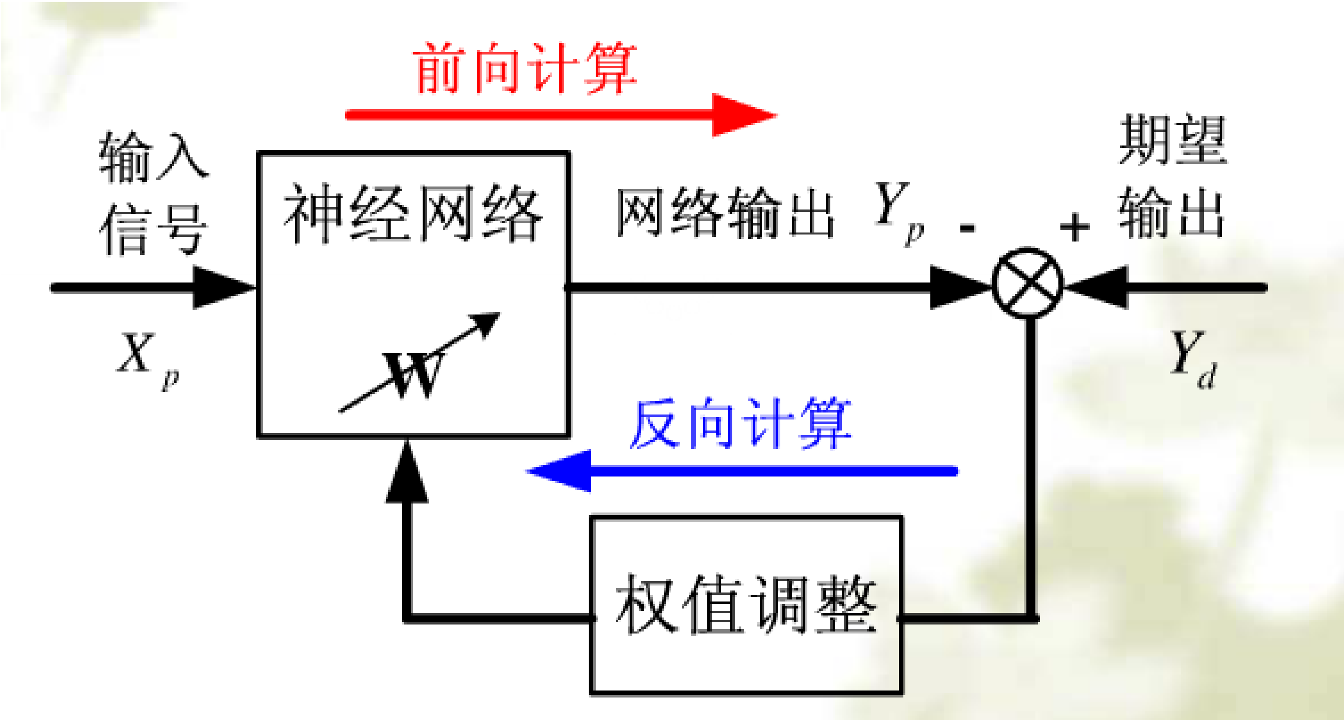
✓ 网络的训练实质上是对突触权阵的调整，以满足当输入X_p时，其输出应为Y_d。

❖ 思想

✓ 前向计算得到网络的输出，反向计算得到误差的积累，由梯度下降法调整权值。

二、多层感知器模型和学习算法

❖ 学习算法机构图



二、多层感知器模型和学习算法

❖ 性能指标

N个样本

正定的、可微的凸函数

$$J=\sum_{p=1}^{N}E_p=\sum_{p=1}^{N}\sum_{i=1}^{n_o}\phi\left(y_{di}-y_{pi}\right)$$

误差的平方和

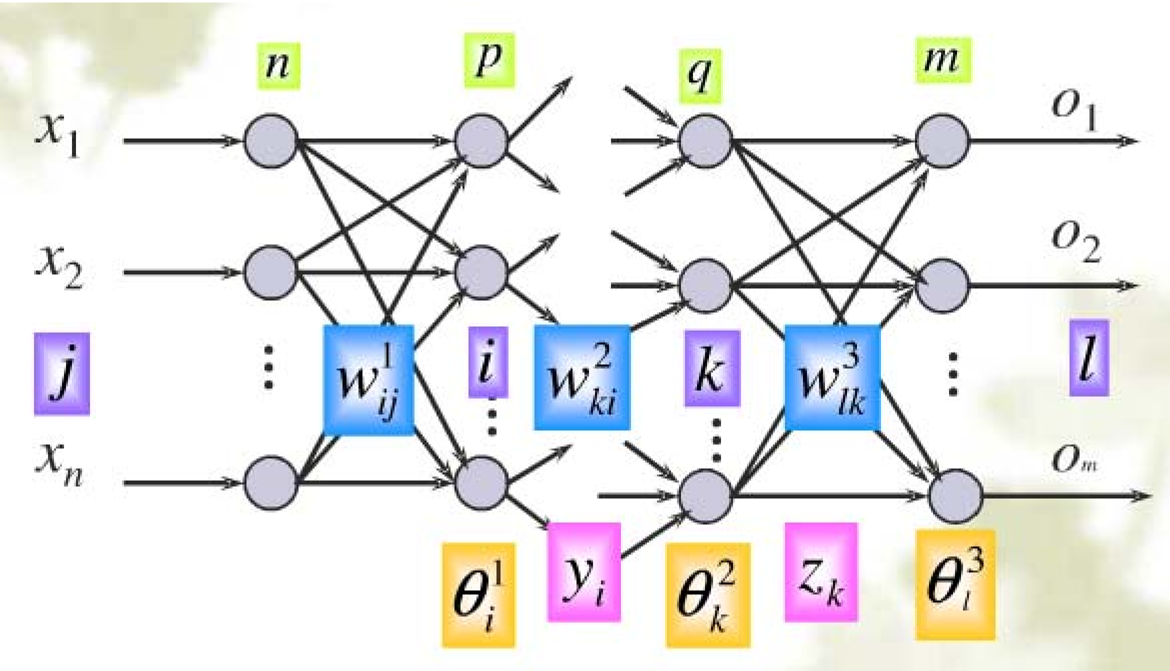
$$E_p=\frac{1}{2}\sum_{i=1}^{n_o}\left(y_{di}-y_{pi}\right)^2$$

✓ 梯度下降法：权值的变化与误差梯度的下降成正比，使误差指标不断减小。

$$\Delta W=-\eta\frac{\partial J}{\partial W}$$

二、多层感知器模型和学习算法

❖ 学习算法



二、多层感知器模型和学习算法

❖ 学习算法（续）

✓ 神经网络输入： $X^T=[x_1,x_2,\cdots,x_n]$

✓ 神经网络输出： $O^T=[o_1,o_2,\cdots,o_m]$

✓ 期望输出： $D^T=[d_1,d_2,\cdots,d_m]$

✓ 隐层输出： $Y^T=[y_1,y_2,\cdots,y_p]$ $Z^T=[z_1,z_2,\cdots,z_q]$

✓ 加权矩阵： $\{w_{ij}^1\}_{p\times n},\{w_{ki}^2\}_{q\times p},\{w_{lk}^3\}_{m\times q}$

✓ 阈值向量： $\{\theta_i^1\}_{p\times 1},\{\theta_k^2\}_{q\times 1},\{\theta_l^3\}_{m\times 1}$

✓ 非线性作用函数（激活函数）： $f(Net_i^1)=y_i$ $f(Net_k^2)=z_k$ $f(Net_l^3)=o_l$

二、多层感知器模型和学习算法

❖ 学习算法（续）

✓ 第一隐层输出：

$$y_i = f\left(Net_i^1\right) = f\left(\sum_{j=1}^{n} w_{ij}^1 \cdot x_j + \theta_i^1\right)$$

$$j = 1, 2, \cdots, n;\ i = 1, 2, \cdots, p;$$

✓ 第二隐层输出：

$$z_k = f\left(Net_k^2\right) = f\left(\sum_{i=1}^{p} w_{ki}^2 \cdot y_i + \theta_k^2\right)$$

$$k = 1, 2, \cdots, q$$

✓ 输出层输出：

$$o_l = f\left(Net_l^3\right) = f\left(\sum_{k=1}^{q} w_{lk}^3 \cdot z_k + \theta_l^3\right)$$

$$l = 1, 2, \cdots, m$$

二、多层感知器模型和学习算法

❖ 学习算法（续）

$$J = \frac{1}{2}\sum_{l=1}^{m}(d_l - o_l)^2$$

✓ 第一步：计算梯度 $\nabla_{W^3}J,\ \nabla_{\theta^3}J$

$$\begin{aligned}\frac{\partial J}{\partial w_{lk}^3} &= \frac{\partial J}{\partial Net_l^3}\frac{\partial Net_l^3}{\partial w_{lk}^3} \\ &= \frac{\partial J}{\partial o_l}\frac{\partial o_l}{\partial Net_l^3}\frac{\partial Net_l^3}{\partial w_{lk}^3} \\ &= -(d_l - o_l)f'(Net_l^3)z_k \\ &= -\delta_l^3 z_k\end{aligned}$$

$$\begin{aligned}\frac{\partial J}{\partial \theta_l^3} &= \frac{\partial J}{\partial Net_l^3}\frac{\partial Net_l^3}{\partial \theta_l^3} \\ &= \frac{\partial J}{\partial o_l}\frac{\partial o_l}{\partial Net_l^3}\frac{\partial Net_l^3}{\partial \theta_l^3} \\ &= -(d_l - o_l)f'(Net_l^3) \\ &= -\delta_l^3\end{aligned}$$

$$\delta_l^3 \overset{\Delta}{=} (d_l - o_l)f'(Net_l^3)$$

二、多层感知器模型和学习算法

✓ **第二步：计算梯度** $\nabla_{W^2} J, \nabla_{\theta^2} J$

$$\frac{\partial J}{\partial w_{ki}^2} = \frac{\partial J}{\partial Net_k^2}\frac{\partial Net_k^2}{\partial w_{ki}^2} = \frac{\partial J}{\partial z_k}\frac{\partial z_k}{\partial Net_k^2}\frac{\partial Net_k^2}{\partial w_{ki}^2} = \frac{\partial J}{\partial z_k} f'(Net_k^2) y_i$$

$$= \left(\sum_{l=1}^{m} \frac{\partial J}{\partial o_l}\frac{\partial o_l}{\partial Net_l^3}\frac{\partial Net_l^3}{\partial z_k} \right) f'(Net_k^2) y_i$$

$$= \left(\sum_{l=1}^{m} -(d_l - o_l) f'(Net_l^3) w_{lk}^3 \right) f'(Net_k^2) y_i$$

$$= -\left(\sum_{l=1}^{m} \delta_l^3 w_{lk}^3 \right) f'(Net_k^2) y_i = -\delta_k^2 y_i$$

$$\frac{\partial J}{\partial \theta_k^2} = -\left(\sum_{l=1}^{m} \delta_l w_{lk}^3 \right) f'(Net_k^2) = -\delta_k^2$$

二、多层感知器模型和学习算法

✓ 第三步：计算梯度 $\nabla_{w^1} J$，$\nabla_{\theta^1} J$

$$\frac{\partial J}{\partial w_{ij}^1} = \frac{\partial J}{\partial Net_i^1}\frac{\partial Net_i^1}{\partial w_{ij}^1} = \frac{\partial J}{\partial y_i}\frac{\partial y_i}{\partial Net_i^1}\frac{\partial Net_i^1}{\partial w_{ij}^1}$$

$$= \left(\sum_{k=1}^{q}\frac{\partial J}{\partial Net_k^2}\frac{\partial Net_k^2}{\partial y_i}\right) f'(Net_i^1)x_j$$

$$= -\left(\sum_{k=1}^{q}\delta_k^2 w_{ki}^2\right) f'(Net_i^1)x_j = -\delta_i^1 x_j$$

$$\frac{\partial J}{\partial \theta_i^1} = -\delta_i^1$$

二、多层感知器模型和学习算法

❖ 学习算法描述

✓ 选定初始权阵（一般给一组较小的随机数）： W^1, W^2, W^3

✓ 对每个样本，重复下述过程，直到收敛：

- 正向过程计算： $Net^1, Y, Net^2, Z, Net^3, O, D-O$
- 反向过程计算： $f'(Net_l^3), \delta_l^3, \frac{\partial J}{\partial w_{lk}^3}, \frac{\partial J}{\partial \theta_l^3}, f'(Net_k^2), \delta_k^2, \frac{\partial J}{\partial w_{ki}^2}, \frac{\partial J}{\partial \theta_k^2}, f'(Net_i^1), \delta_i^1, \frac{\partial J}{\partial w_{ij}^1}, \frac{\partial J}{\partial \theta_i^1};$
- 修正权值 $\Delta W^r = -\eta_r \frac{\partial J}{\partial W^r}, \quad r = 1,2,3.$

$$W^r(t+1) = W^r(t) - \eta_r \frac{\partial J}{\partial W^r}$$

二、多层感知器模型和学习算法

❖ 学习算法描述（续）

$$\Delta W^r = -\eta_r \frac{\partial J}{\partial W^r}, \quad r = 1,2,3.$$

$$W^r(t+1) = W^r(t) - \eta_r \frac{\partial J}{\partial W^r}$$

✓ 输出层与第二隐层连接权值及阈值更新：

$$\frac{\partial J}{\partial w_{lk}^3} = -\delta_l^3 z_k \qquad \frac{\partial J}{\partial \theta_l^3} = -\delta_l^3 \qquad \delta_l^3 \stackrel{\Delta}{=} -\frac{\partial J}{\partial Net_l^3} = (d_l - o_l) f'(Net_l^3)$$

✓ 第二隐层与第一隐层连接权值及阈值更新：

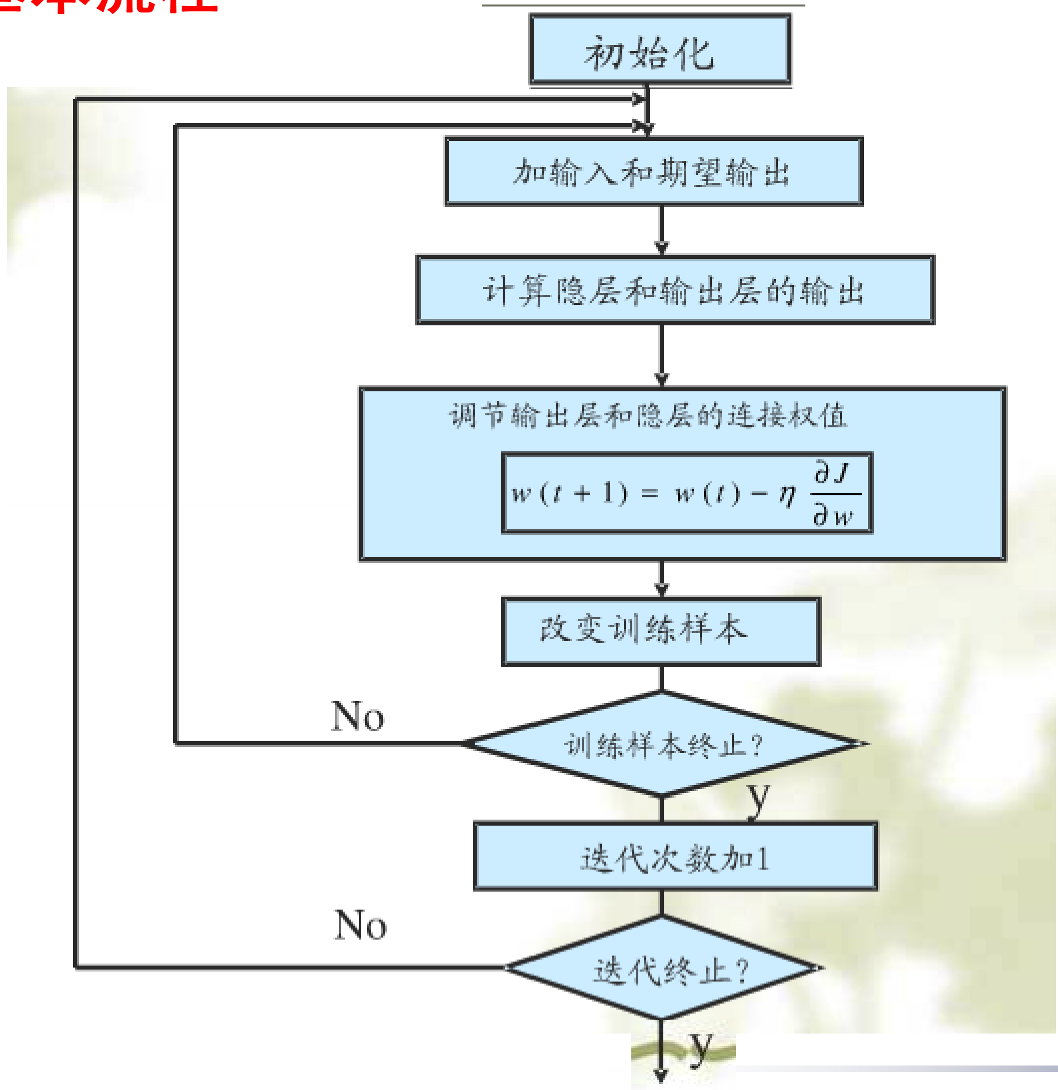
$$\frac{\partial J}{\partial w_{ki}^2} = -\delta_k^2 y_i \qquad \frac{\partial J}{\partial \theta_k^2} = -\delta_k^2 \qquad \delta_k^2 \stackrel{\Delta}{=} -\frac{\partial J}{\partial Net_k^2} = \left(\sum_{l=1}^{m} \delta_l^3 w_{lk}^3\right) f'(Net_k^2)$$

✓ 第一隐层与输入层连接权值及阈值更新：

$$\frac{\partial J}{\partial w_{ij}^1} = -\delta_i^1 x_j \qquad \frac{\partial J}{\partial \theta_i^1} = -\delta_i^1 \qquad \delta_i^1 \stackrel{\Delta}{=} -\frac{\partial J}{\partial Net_i^1} = \left(\sum_{k=1}^{q} \delta_k^2 w_{ki}^2\right) f'(Net_i^1)$$

二、多层感知器模型和学习算法

❖ 学习基本流程



二、多层感知器模型和学习算法

❖ 影响BP学习算法的因素

✓ 权系数的初值：

随机选较小的值，尽量均匀覆盖权值空间，避免出现初始值相同的情况。

✓ 学习方式：

增量型学习方法效果好，累积型学习方法速度快。

✓ 激励函数：

非减可微函数。可通过调节S函数的斜率或采用其他激励函数来改善网络的学习性能。

✓ 学习速率：

学习速率小，训练速度慢。学习速率大，训练速度快，可能出现震荡现象。

二、多层感知器模型和学习算法

❖ BP学习算法的局限性

✓ 非线性优化的局部最小，或震荡不收敛。

✓ 收敛速度很慢。

✓ 新样本的加入会影响以学习过的老样本。

二、多层感知器模型和学习算法

❖ BP学习算法的改进

✓ 选用不同的作用函数、性能指标。

✓ 解决局部极小问题：

选用不同的初值迭代；激励函数加入斜率因子；模拟退火方法；分解子网。

✓ 加快收敛速度：

采用不同的激励函数；变学习率方法；利用激励函数的二阶导数；最速下降法；组合学习方法；权值修正引入动量因子；遗传算法，等等。

二、多层感知器模型和学习算法

❖ 自适应变学习率方法

✓ 准则

检查权值的修正值是否真正降低了误差函数，若是，则说明所选取的学习速率值小了，可以对其增加一个量。

若不是，而产生了过调，那么就应该减小学习速率的值。

✓ 调整公式

$$\eta(k+1)=\begin{cases}1.05\eta(k) & J(k+1)<J(k)\\ 0.7\eta(k) & J(k+1)>1.04J(k)\\ \eta(k) & \text{其它}\end{cases}$$

二、多层感知器模型和学习算法

❖ 附加动量法

在修正其权值时，不仅考虑误差在梯度上的作用，而且考虑在误差曲面上变化趋势的影响。利用附加动量的作用则有可能滑过局部极小值。

该方法是在反向传播法的基础上，在每一个权值的变化上加上一项正比于前次权值变化量的值，并根据反向传播法来产生新的权值变化。

✓ 权值调节公式

$$\Delta w_{ij}(k) = -\eta \frac{\partial J}{\partial w_{ij}} + \alpha \Delta w_{ij}(k-1)$$

动量因子，一般取0.95左右

二、多层感知器模型和学习算法

❖ BP网络的设计

- ✓ 网络的层数
- ✓ 隐含层的神经元数
- ✓ 初始权值的选取
- ✓ 学习速率的选取
- ✓ 期望误差的选取

二、多层感知器模型和学习算法

❖ BP网络的层数

理论上已经证明：至少一个S型隐含层加上一个线性输出层的网络，能够逼近任何有理函数。

增加层数主要可以更进一步的降低误差，提高精度，但同时也使网络复杂化，从而增加了网络权值的训练时间。

一般情况下，应优先考虑增加隐含层中的神经元数。

二、多层感知器模型和学习算法

❖ **BP网络隐含层的神经元数**

网络训练精度的提高，可以通过采用一个隐含层，而增加其神经元数的方法来获得。

具体设计时：通过对不同神经元数进行训练对比，然后适当地加上一点余量。

❖ **初始权值的选取**

一般选取初始权值在（-1，1）之间的随机数。

❖ BP网络学习速率的选取

●学习速率决定每一次循环训练中所产生的权值变化量。

●大的学习速率可能导致系统的不稳定。

●小的学习速率导致较长的训练时间，可能收敛很慢，不过能保证网络的误差值不跳出误差表面的低谷而最终趋于最小的误差值。

●一般情况下，倾向于选取小的学习速率以保证系统的稳定性。范围：0.01~0.8之间。

二、多层感知器模型和学习算法

❖ BP网络期望误差的选取

●在设计网络的训练过程中，期望误差值也应通过对比训练后确定一个合适的值。

●“合适”：相对于所需要的隐含层的节点数来确定，因为较小的期望误差值是要靠增加隐含层的节点，以及训练时间来获得的。

●一般情况下，作为对比，可以同时对两个不同期望误差值的网络进行训练，最后通过综合因素的考虑来确定采用其中一个网络。

二、多层感知器应用实例

❖ **BP神经网络水处理系统的模拟与预测。**

训练样本

实验号	臭氧浓度(mg/L)	入口UV_{254}	UV_{254}去除率(%)
1	1.16	0.116	50.2
2	1.35	0.104	59.5
3	1.72	0.078	58.8
4	1.86	0.107	66.2
5	1.97	0.136	65.5
6	2.15	0.082	64.5
7	2.23	0.125	73.6
8	2.48	0.076	76.4
9	2.79	0.122	78.5
10	2.85	0.092	79.2
11	3.07	0.081	81.4
12	3.45	0.068	90.3
13	3.59	0.077	93.1
14	3.80	0.108	98.2
15	3.93	0.128	97.3
16	4.14	0.063	98.1
17	4.46	0.135	97.3
18	4.55	0.070	98.8
19	4.84	0.126	96.9
20	5.03	0.087	98.6

二、多层感知器应用实例

❖ **BP神经网络水处理系统的模拟与预测。**

检验样本

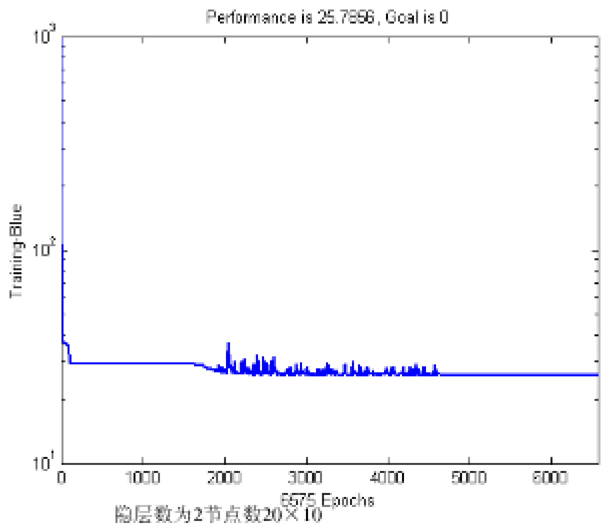
实验号	臭氧浓度(mg/L)	入口UV_{254}	UV_{254}去除率(%)
1	1.42	0.086	?
2	2.51	0.071	?
3	3.21	0.107	?
4	4.29	0.096	?
5	5.24	0.65	?

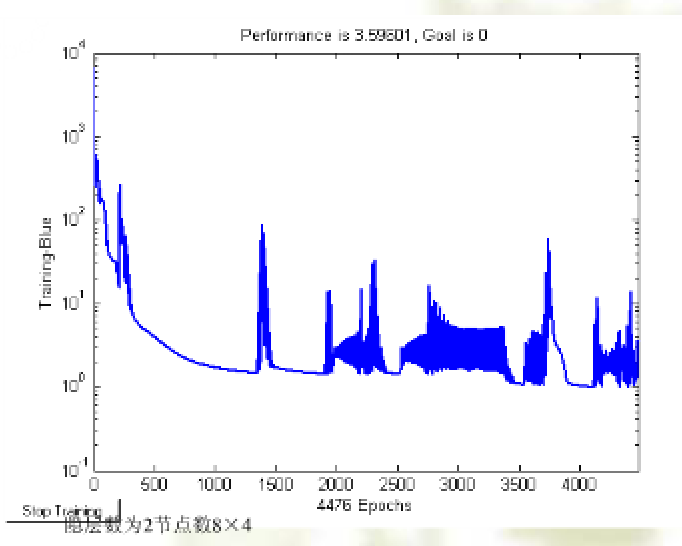
上海交通大学 Shanghai Jiao Tong University

二、多层感知器应用实例

❖ 隐含层神经元数量的选择

隐含层为20*10和8*4，训练结果比较。

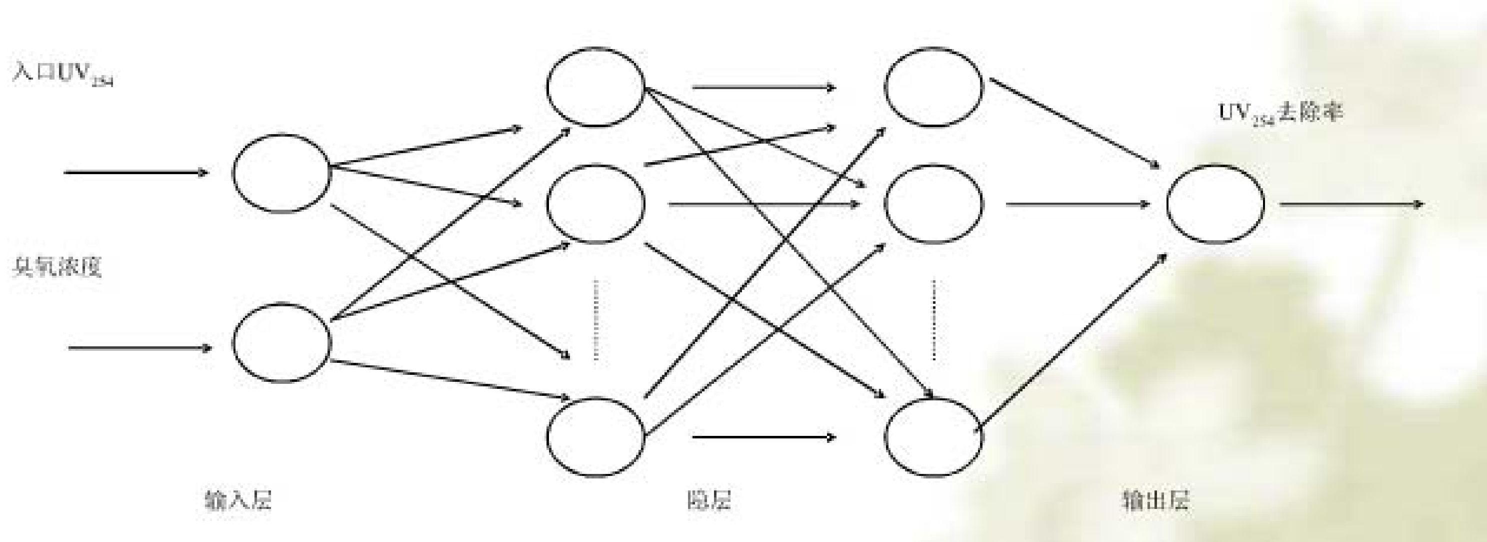




二、多层感知器应用实例

❖ 各层节点数的选择

输入层节点2个，第一隐含层12个，第二隐含层6个，输出层6个。



二、多层感知器应用实例

❖ 学习率和动量因子的选择

$$\eta = 0.7, \qquad \alpha = 0.9$$

❖ 初始权值的选择

$-0.5 \sim +0.5$ 之间的随机数。

❖ 收敛误差界值的选择

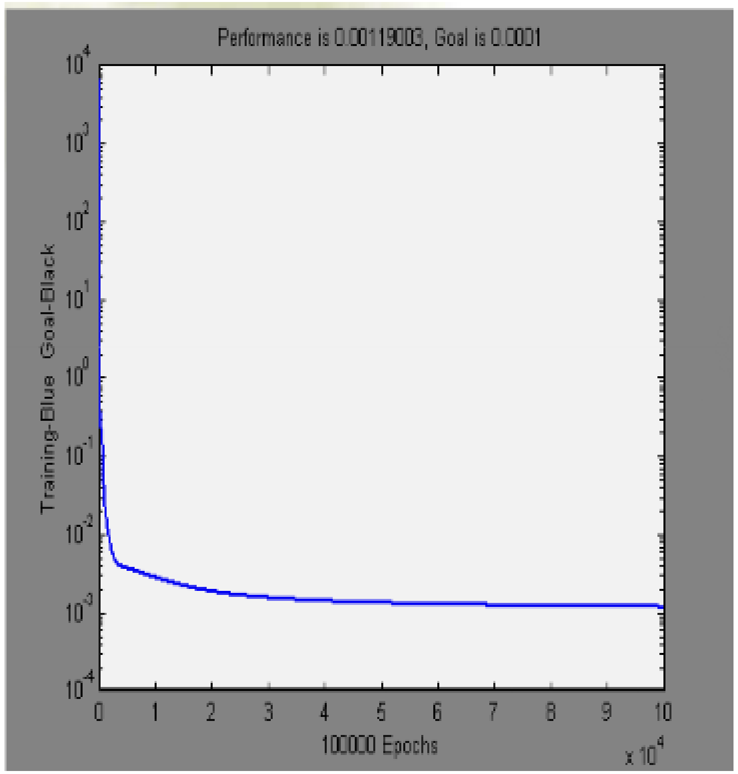
$$E_{\min} = 0.0001$$

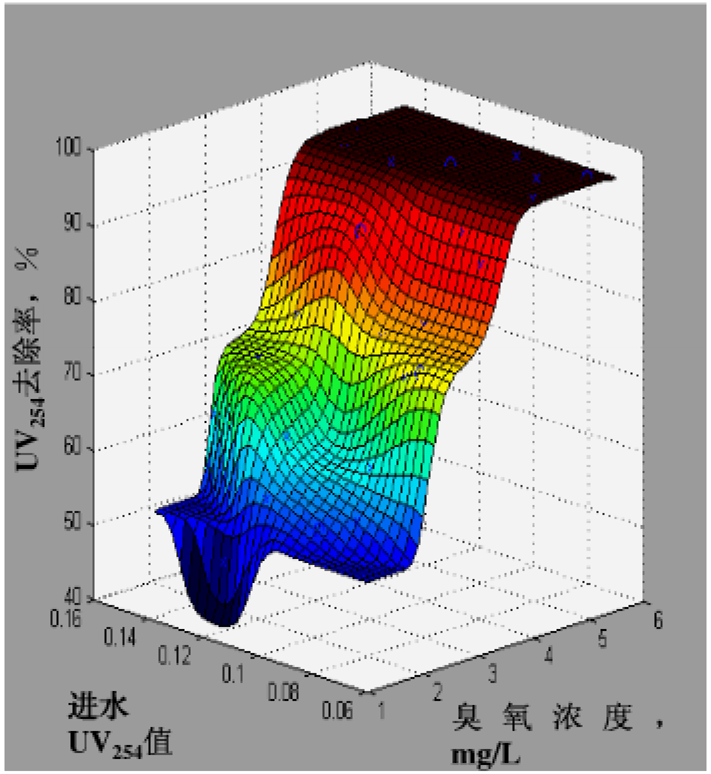
❖ 输入数据的预处理

输入样本归一化（或称正则化）处理。

二、多层感知器应用实例

❖ BP网络训练误差曲线和网络模型





二、多层感知器应用实例

❖ 模型检测结果与实测值比较

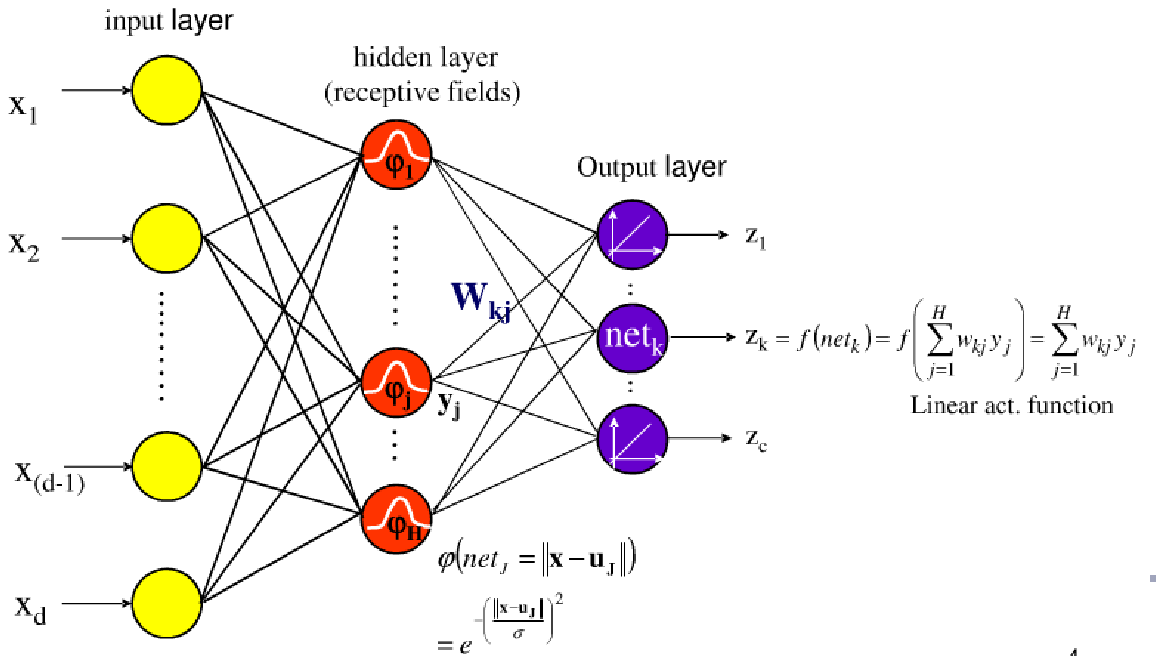
实验号	臭氧 (mg/L)	UV_{254}去除率(%)		相对误差 (%)
		实测值	网络预测值	
1	1.42	58.1	57.3	-1.47
2	2.51	78.8	77.7	-1.47
3	3.21	89.6	90.5	0.96
4	4.29	96.5	97.9	1.45
5	5.24	97.8	97.9	0.14

三、径向基函数神经网络

- RBF神经网络的结构
- RBF神经网络的基本功能
- RBF神经网络的学习算法
- RBF神经网络的扩展

三、径向基函数神经网络

- 1985年，Powell提出了多变量插值的径向基函数（Radial Basis Function，简称RBF）方法。
- 1988年，Moody和Darken将RBF应用于神经网络设计，从而构成了RBF神经网络。它是一种局部逼近的神经网络。
- RBF网络是三层前馈网络，也可用于函数逼近以及分类。



三、径向基函数神经网络

❖ RBF网络基本思想：

✓ 用RBF作为隐单元的“基”构成隐含层空间，将输入矢量直接（即不需要通过权连接）映射到隐空间。

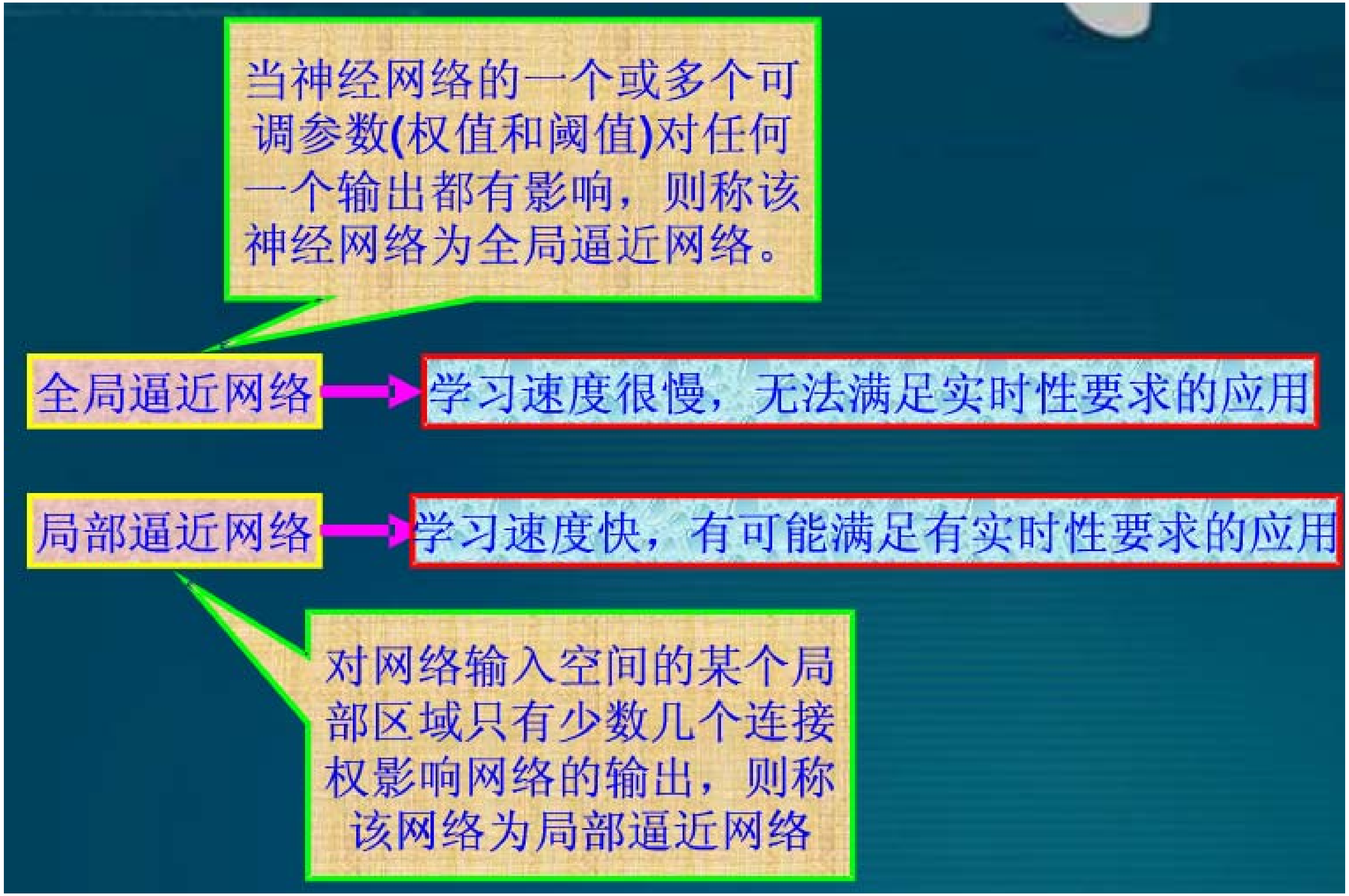
✓ 当RBF的中心点确定后，映射关系也就确定。

✓ 隐含层空间到输出空间的映射是线性的。

三、径向基函数神经网络

- RBF网络的最显著特点：隐节点基函数采用距离函数（如欧式距离），并使用径向基函数（如Gaussian函数）作为激活函数。
- 径向基函数：关于n维空间的一个中心点具有径向对称性，而且神经元的输入离该中心点越远，神经元的激活程度就越低，隐节点的这一特性常被称为“局部特性”。
- RBF网络的每个隐节点都具有一个数据中心。

三、径向基函数神经网络


当神经网络的一个或多个可调参数(权值和阈值)对任何一个输出都有影响，则称该神经网络为全局逼近网络。
全局逼近网络
学习速度很慢，无法满足实时性要求的应用
局部逼近网络
学习速度快，有可能满足有实时性要求的应用
对网络输入空间的某个局部区域只有少数几个连接权影响网络的输出，则称该网络为局部逼近网络

三、径向基函数神经网络

在RBF 中，输入层到隐含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，变换到另一线性可分的空间（通常是高维空间），通过合理选择这一变换使在新空间中原问题线性可分，然后用一个线性单元来解决问题，从而很容易的达到从非线性输入空间向输出空间映射的目的。

三、径向基函数神经网络

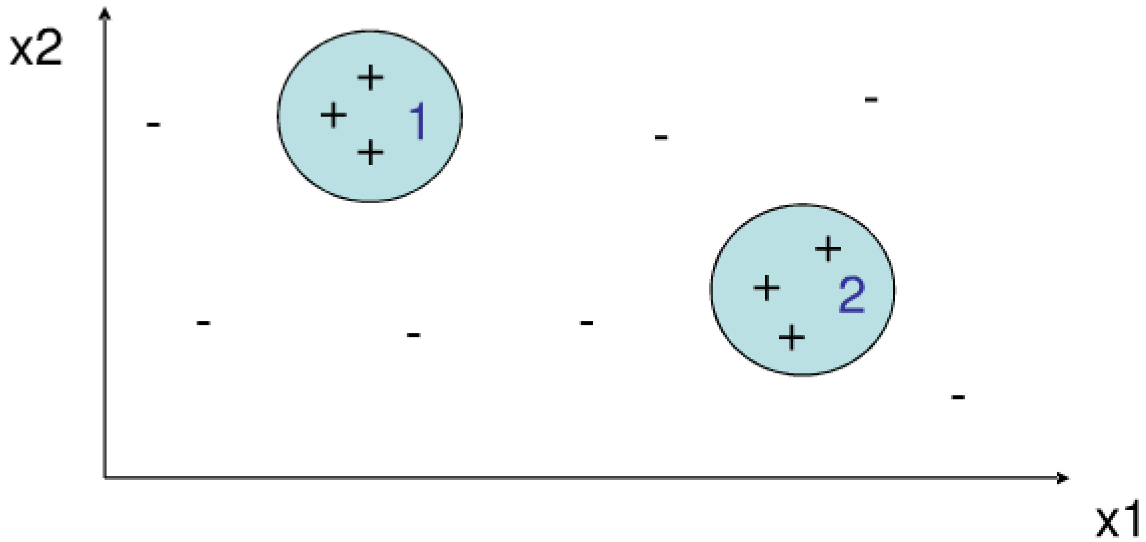
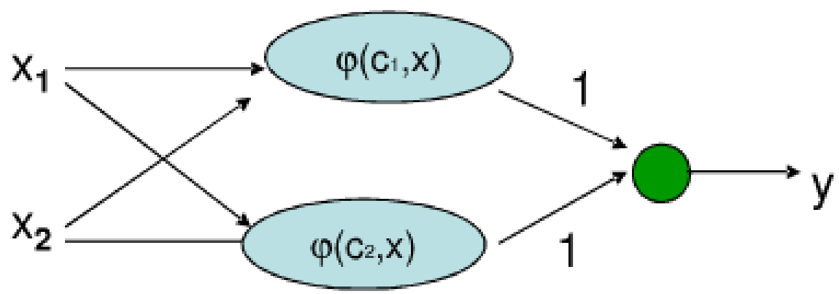


图1和圈2中的样本数据分别属于一类，圈外样本属于另一类。
RBF如何划分这两类？（非线性分类）

三、径向基函数神经网络

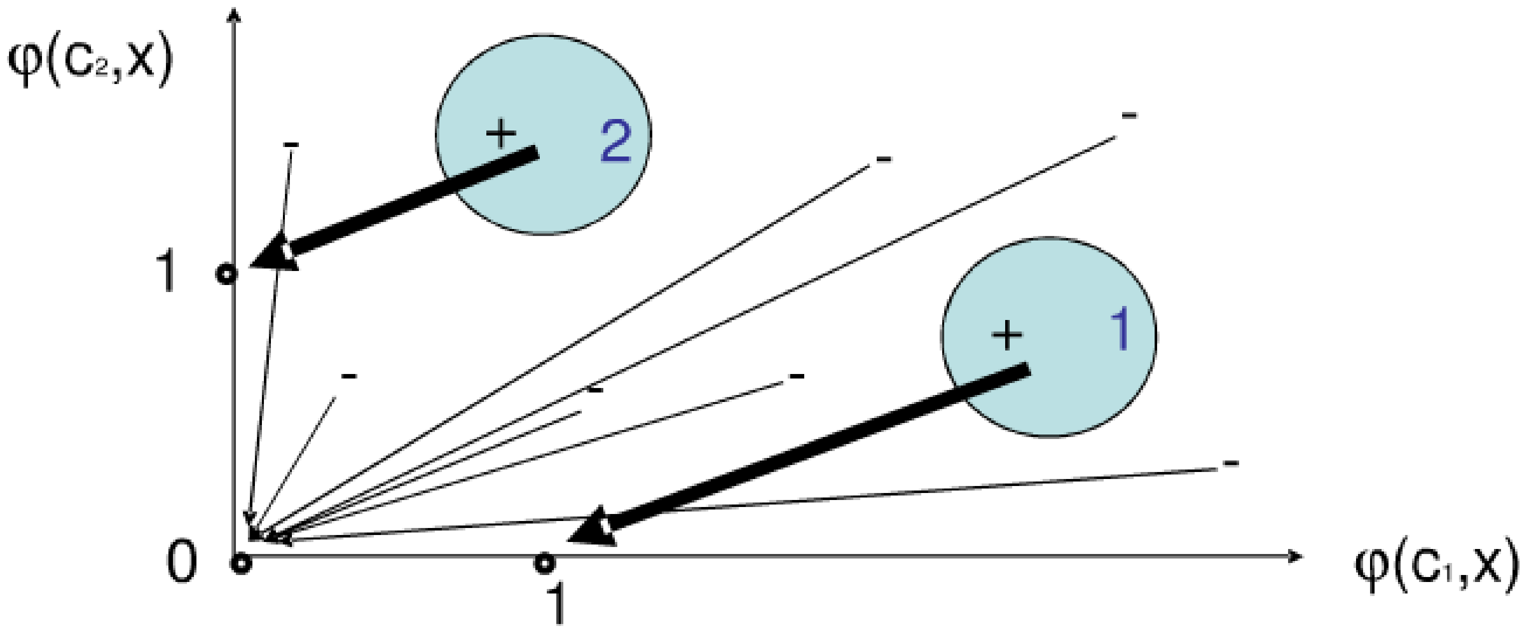
设： c_1,c_2 和 r_1, r_2 分别是圈1和圈2的中心和半径，
样本$x = (x_1,x_2)$



$\varphi(c_1,x)$ = 1 if distance of x from c_1 less than r1 and 0 otherwise
$\varphi(c_2,x)$ = 1 if distance of x from c_2 less than r2 and 0 otherwise

φ： Hyperspheric radial basis function

三、径向基函数神经网络



通过隐层特征空间（φ(c, x)）的作用，圈2中的样本被映射到(0,1)，圈1 中的样本被映射到(1,0)，圈外的样本均被映射到(0,0).

这一两分类问题在隐层特征空间中变成线性可分!

三、径向基函数神经网络

❖ 常用径向基函数：

(1). *Gaussian*函数

$$\Phi_i(t) = e^{-\frac{t^2}{\delta_i^2}}$$

式中 δ_i 称为扩展常数或宽度。δ_i 越小，径向基函数的宽度就越小，基函数就越具有选择性。

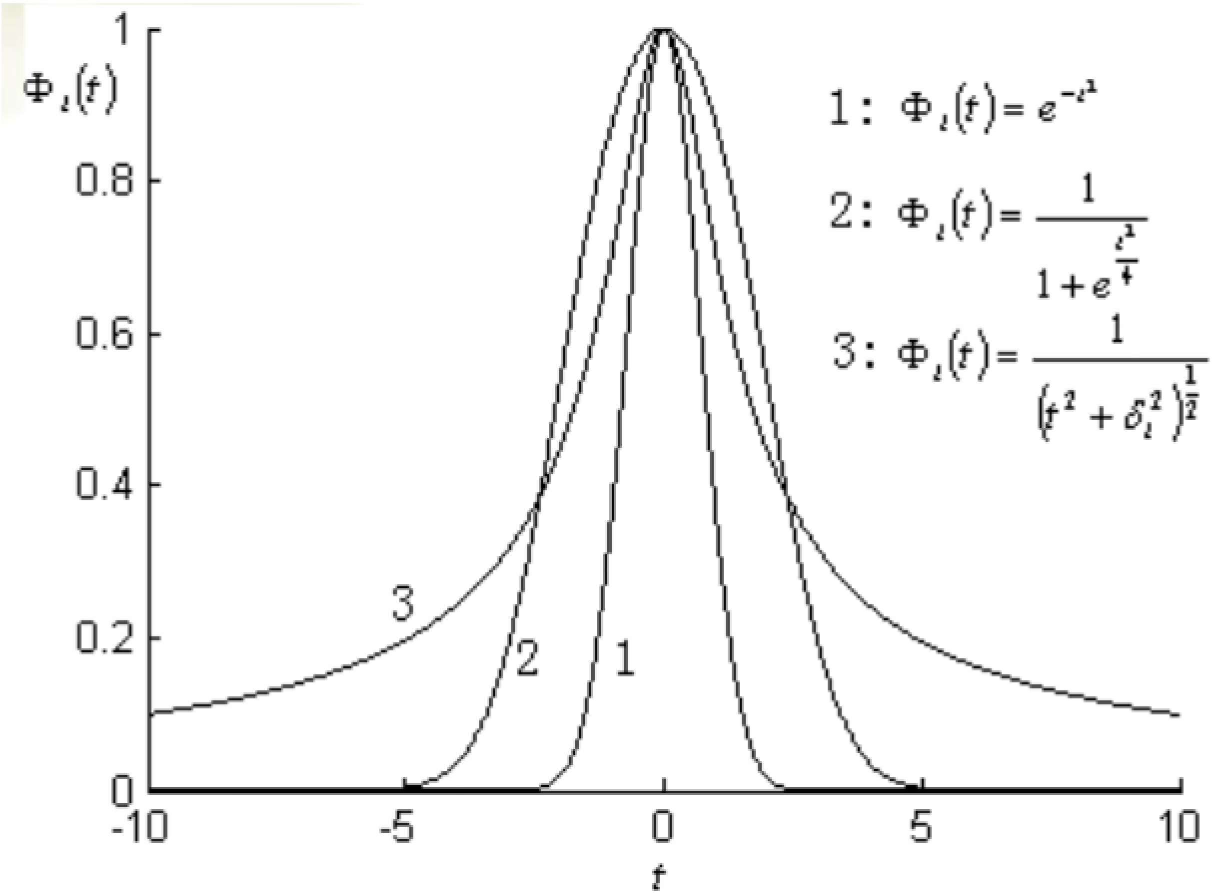
(2). *Reflected sigmoid*函数

$$\Phi_i(t) = \frac{1}{1 + e^{\frac{t^2}{\delta^2}}}$$

(3). 逆*Multiquadric*函数 $\Phi_i(t) = \dfrac{1}{\left(t^2 + \delta_i^2\right)^\alpha}, \quad \alpha > 0$

三、径向基函数神经网络

❖ 常用径向基函数：



三、径向基函数神经网络

❖ 采用高斯激活函数的优点：

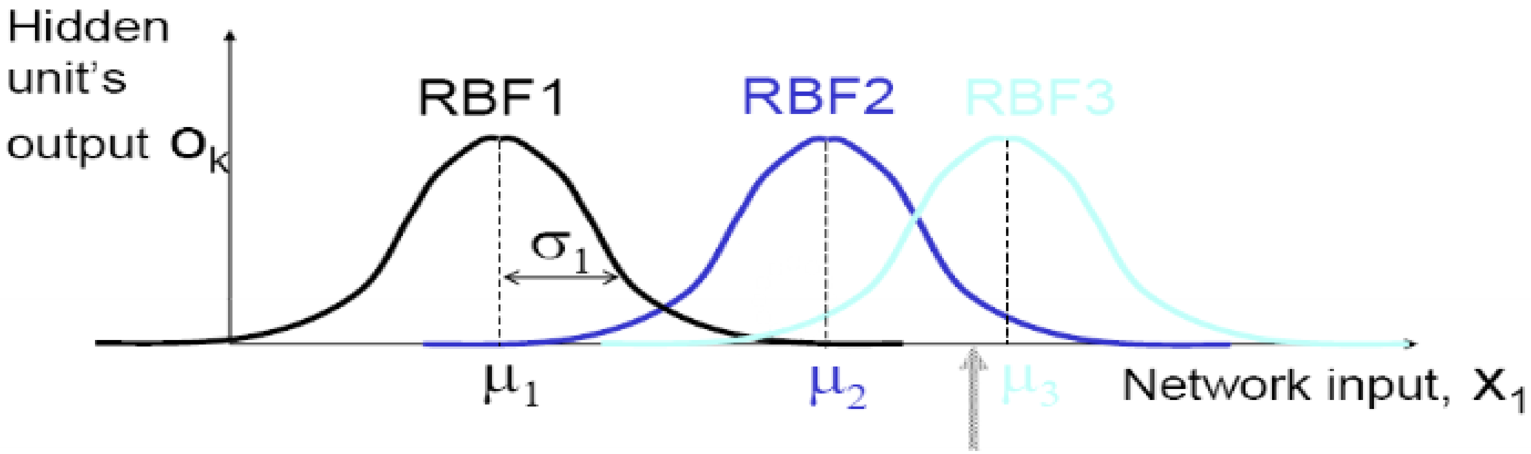
1）表示形式简单，即使对于多变量输入也不增加太多的复杂性。

2）径向对称。

3）光滑性好，任意阶导数存在。

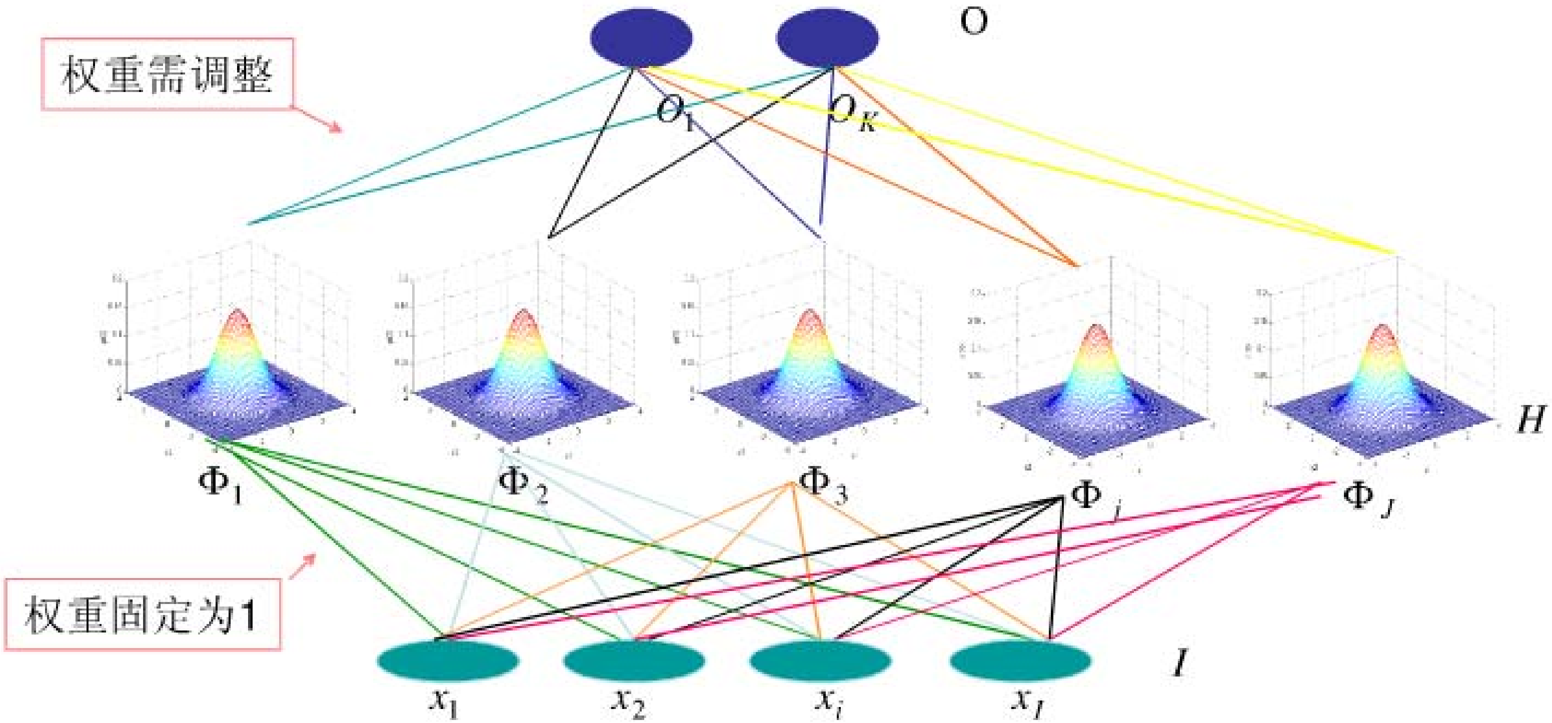
4）由于该基函数表示简单且解析性好，因而便于进行理论分析。

三、径向基函数神经网络



- 三个隐单元具有不同的中心值。
- 对某个输入值（如箭特头所示），RBF3输出最大。因为输入离μ_3最近。
- 每个RBF有一个接收场，即输入空间的某个区域或子空间（有生理基础）

三、径向基函数神经网络



三、径向基函数神经网络

❖ RBF网络的学习算法：

注意：由于RBF 网络的权值算法是单层进行的，它的工作原理采用的是聚类功能，由训练得到输入数据的聚类中心，通过σ值来调节基函数的宽度。虽然网络结构看上去是全连结的，实际网络是局部工作的，即对输入的一组数据，网络只有一个神经元被激活，其他神经元被激活的程度可忽略。所以RBF网络是一个局部逼近网络，训练速度比BP 网络快2～3 个数量级。

三、径向基函数神经网络

- ❖ **RBF网络的学习算法：**
  - ✓ 学习算法需要求解的参数

    1）径向基函数的中心

    2）方差

    3）隐含层到输出层的权值

三、径向基函数神经网络

❖ **RBF网络的学习算法：**

✓ 数据中心的聚类算法类

1）基于K-均值聚类方法求取基函数中心

2）确定扩展常数：各聚类中心确定后，可根据各中心之间的距离确定对应径向基函数的扩展常数。

3）计算隐含层和输出层之间的权值：可采用最小二乘法直接计算得到。

三、径向基函数神经网络

❖ **RBF网络的学习算法小结：**

1) 初始化：对权值ω (0)赋 0 到 1 之间的随机数，隐层神经元数*m*，初始误差*E* 置0，最大误差ε设为一正的小数，学习率η为 0 到 1 之间的小数。

2) 采用模糊*K* 均值聚类算法确定基函数的中心 $c_i(0)$及方差$\sigma_i(0)$，i=1，2，…，m。

3) 按上述梯度下降法调整网络权值ω (0)直到误差$E<\varepsilon$，结束。

三、径向基函数神经网络

❖ **RBF网络的Matlab函数及功能：**

函 数 名	功 能
newrb()	新建一个径向基神经网络
newrbe()	新建一个严格的径向基神经网络
newgrnn()	新建一个广义回归径向基神经网络
newpnn()	新建一个概率径向基神经网络

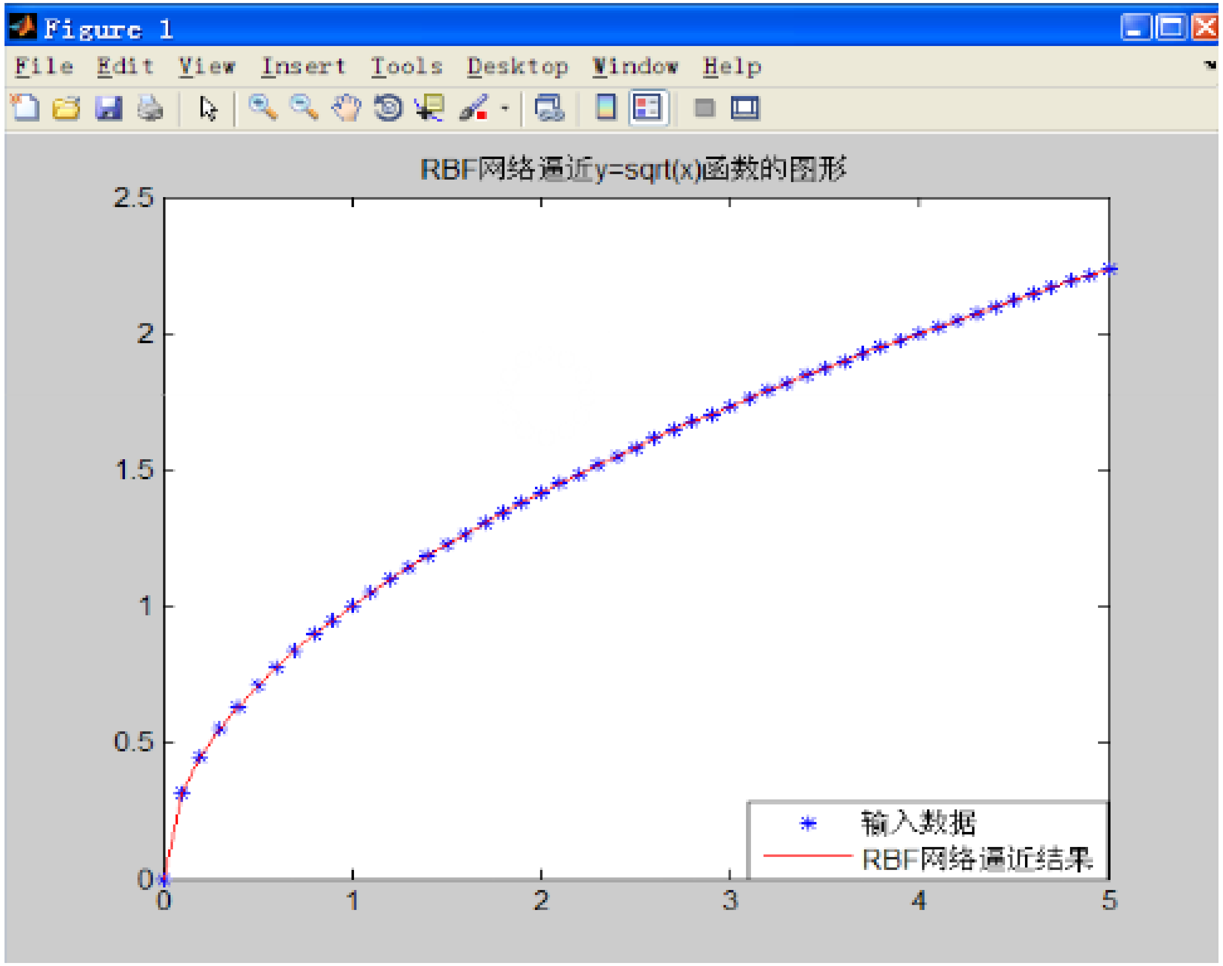
三、径向基函数神经网络

❖ 例如：建立一个RBF网络，对非线性函数y=sqrt(x)，进行逼近，并做出网络的逼近误差曲线。

```
%输入从0开始变化到5，每次变化幅度为0.1
x=0:0.1:5;
y=sqrt(x);
%建立一个目标误差为0，径向基函数的分布密度为
%0.5，隐含层神经元个数的最大值为20，每增加5个
%神经元显示一次结果
net=newrb(x,y,0,0.5,20,5);
t=sim(net,x);
%在以输入x和函数值与网络输出之间的差值y-t坐标
%上绘出误差曲线，并用"*"来标记函数值与网络输
%出之间的差值
plot(x,y-t,'*-')
```

三、径向基函数神经网络

❖ 误差曲线和逼近曲线。



三、小结

❖ RBF网络与多层感知器的比较：均是非线性多层前向网络，不同点如下

1）RBF网络只有一个隐层，而多层感知器的隐层可以是一层或多层。

2）多层感知器的隐层和输出层其神经元模型是一样的，而RBF网络的隐层神经元和输出层神经元不仅模型不同，而且在网络中起到的作用也不一样。

3）RBF网络的隐层是非线性的，输出层是线性的。而当用多层感知器解决模式分类问题时，它的隐层和输出层通常选为非线性的。当用多层感知器解决非线性回归问题时，通常选择线性输出层。

三、小结

4）RBF网络的基函数计算的是输入向量和中心的欧式距离，而多层感知器隐单元的激励函数计算的是输入单元和连接权的内积。

5）RBF网络使用局部指数衰减的非线性函数（如高斯函数）对非线性输入输出映射进行局部逼近。

多层感知器的隐节点采用输入模式与权向量的内积作为激活函数的自变量，而激活函数则采用S型函数或硬限幅函数，因此多层感知器是对非线性映射的全局逼近。

三、小结

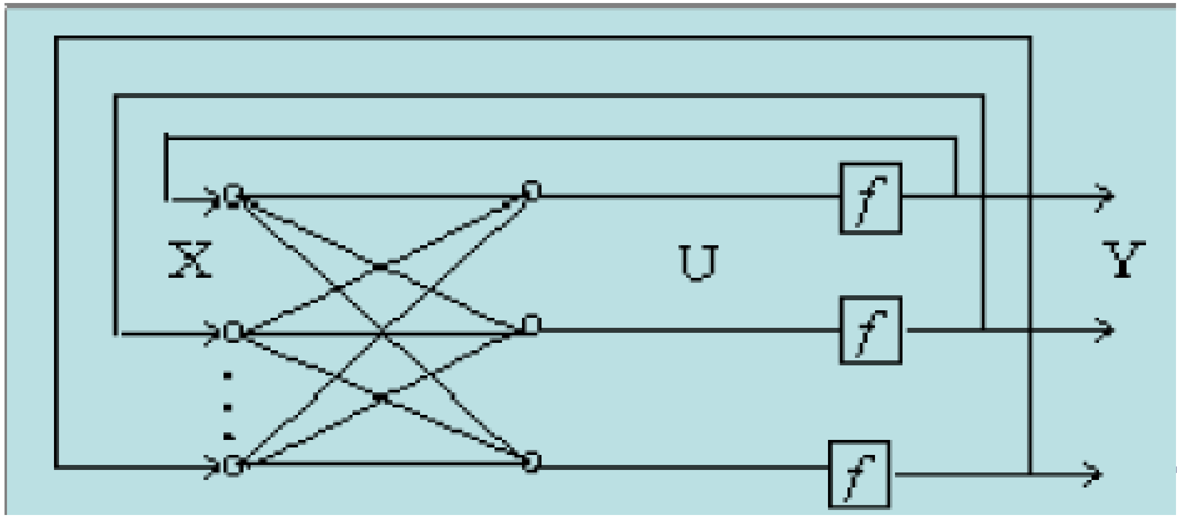
由于RBF网络能够逼近任意的非线性函数，可以处理系统内在的难以解析的规律性，并且具有很快的学习收敛速度，因此RBF网络有着较为广泛的应用。

目前RBF网络已成功用于非线性函数逼近、时间序列分析、数据分类、模式识别、信息处理、图像处理、系统建模、控制和故障诊断等。

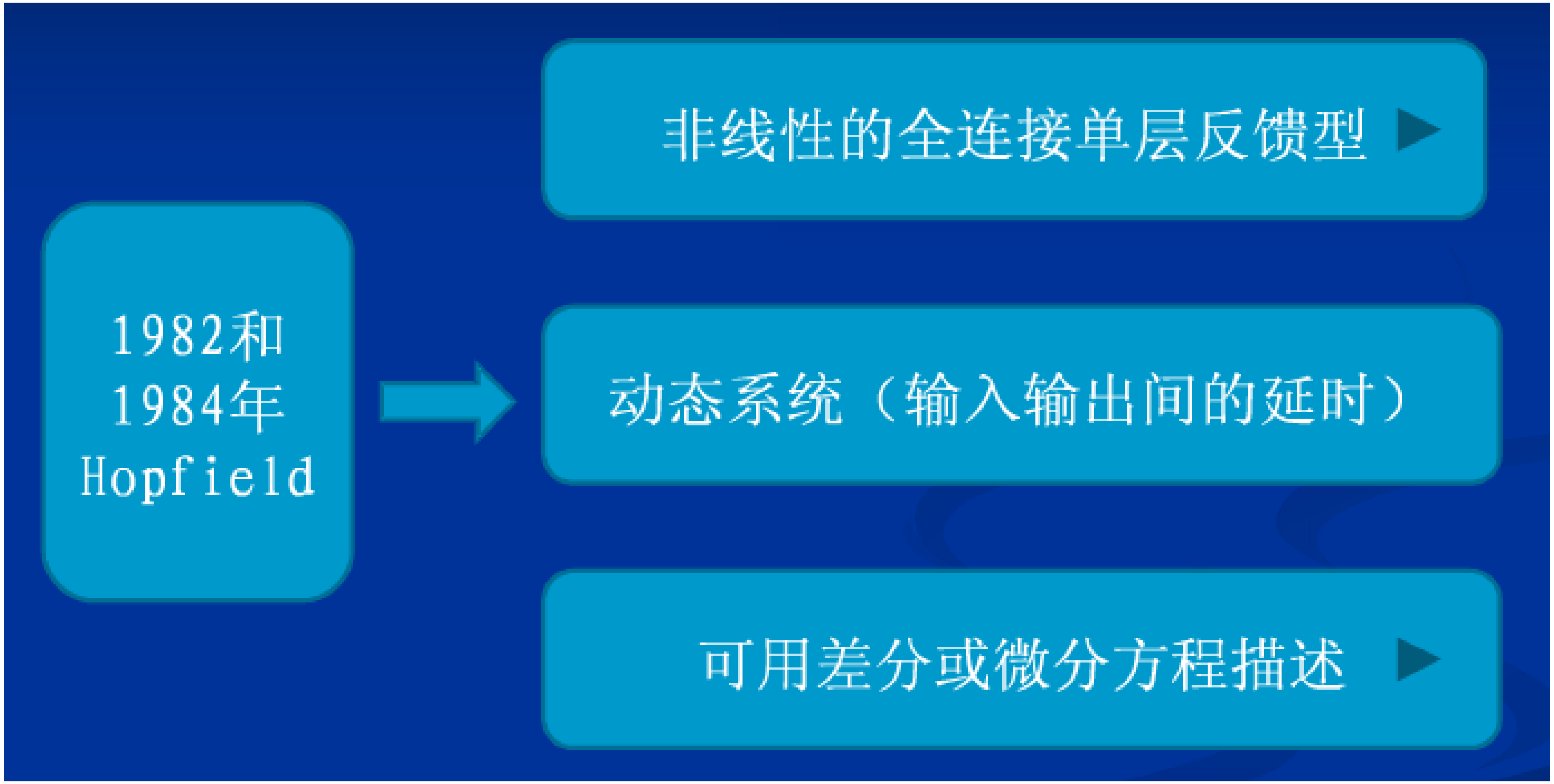
四、Hopfield神经网络

❖ 1982年，美国加州工学院物理学家J.J.Hopfield提出了Hopfield神经网格模型，引入了“计算能量”概念，给出了网络稳定性判断。

❖ 1984年，他又提出了连续时间Hopfield神经网络模型，为神经计算机的研究做了开拓性的工作，开创了神经网络用于联想记忆和优化计算的新途径，有力地推动了神经网络的研究。

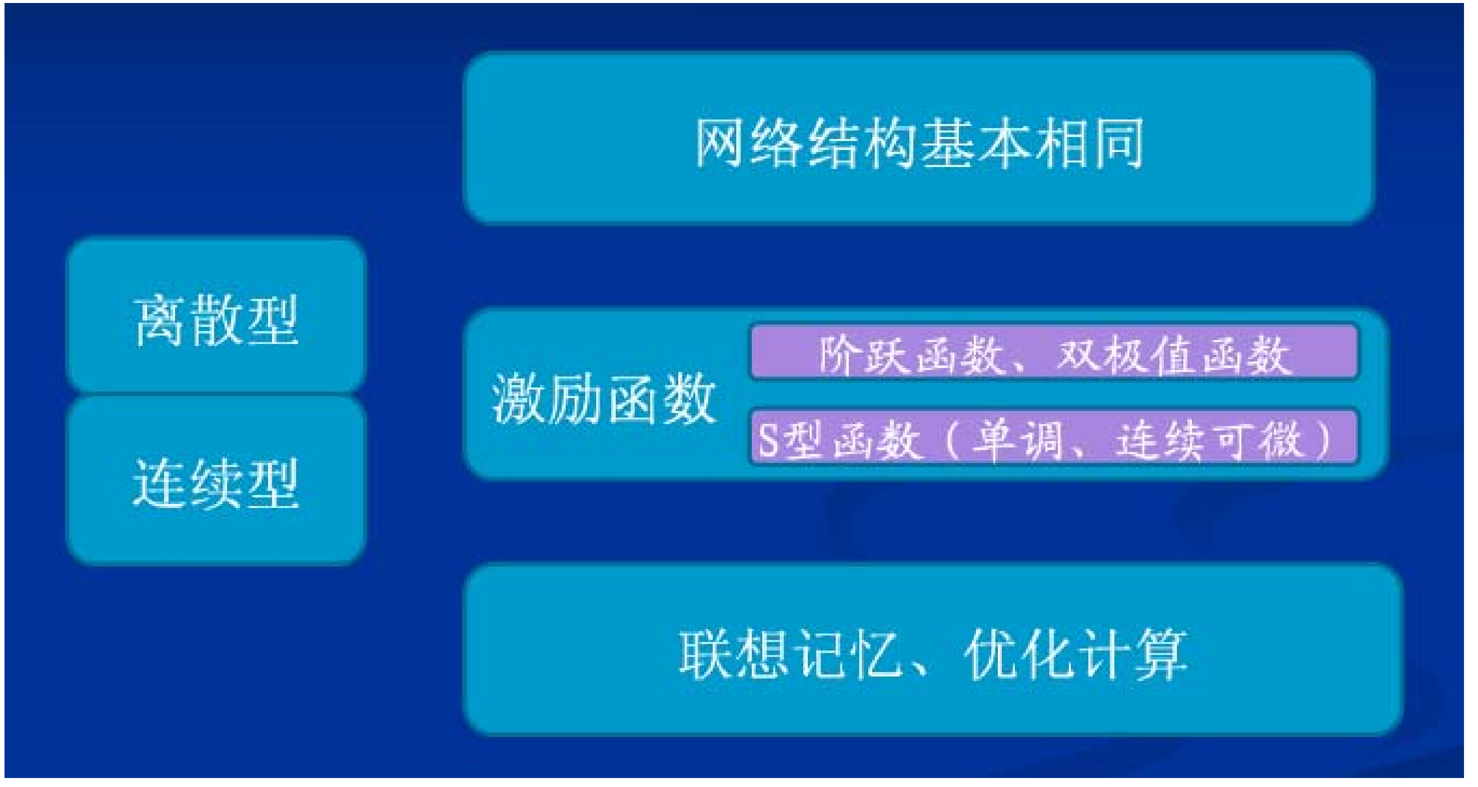


四、Hopfield神经网络



四、Hopfield神经网络

❖根据激励函数选取的不同，Hopfield网络有离散型和连续型两种（DHNN，CHNN）。



四、Hopfield神经网络

❖ **DHNN网络的学习：**

DHNN的学习只是在此神经网络用于联想记忆时才有意义。

其实质是通过一定的学习规则自动调整连接权值，使网络具有期望的能量井分布，并经记忆样本存储在不同的能量井中。

Hebb学习规则

δ 学习规则

四、Hopfield神经网络

❖ DHNN网络的学习（续）：

Hebb学习规则的训练方法

设有n个神经元，每个神经元的活化状态y_i只能取0或1，学习过程中调节w_{ij}的原则为：若i与j两个神经元同时处于兴奋状态，则它们之间的连接应加强，即

$$\Delta w_{ij} = \eta y_i y_j \qquad (i, j = 1,2,\cdots,n; \quad \eta > 0)$$

δ学习规则的训练方法

$$w_{ij}(k+1) = w_{ij}(k) + \eta\left[t - y_i(k)\right] y_j(k)$$

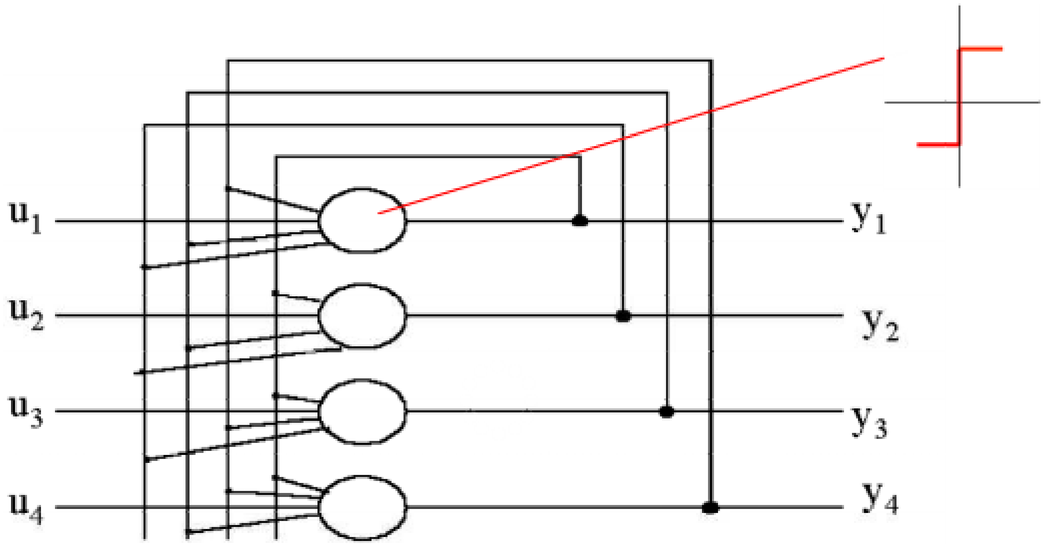
四、Hopfield神经网络

❖ **Hopfield网络结构：**

➢ 单层反馈网络，有n个神经元节点。

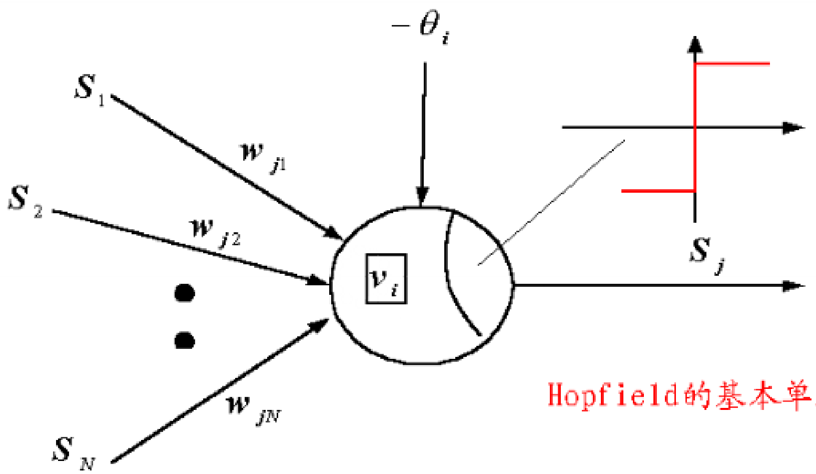
➢ 每个神经元的输出连接到其它神经元的输入，各个节点自己没有反馈。

➢ 每个节点都附有一个阈值和权系数。每个节点都可处于一种可能的状态（1或-1）。



四、Hopfield神经网络

❖ **Hopfield网络基本单元：**



Hopfield的基本单元

$$v_i = \sum_{j=1}^{N} w_{ji} s_j + \theta_i$$

$$s_i = \begin{cases} +1 & v_i > 0 \\ \text{保持以前状态} & v_i = 0 \\ -1 & v_i < 0 \end{cases}$$

四、Hopfield神经网络

❖ Hopfield网络结构：

- 如果Hopfield网络是一个能收敛的稳定网络，则反馈与迭代过程所产生的变化越来越小，一旦到达了稳定平衡状态，那么Hopfield网络就会输出一个稳定的恒值。

- 对于一个Hopfield网络，关键在于确定它在稳定条件下的权系数。

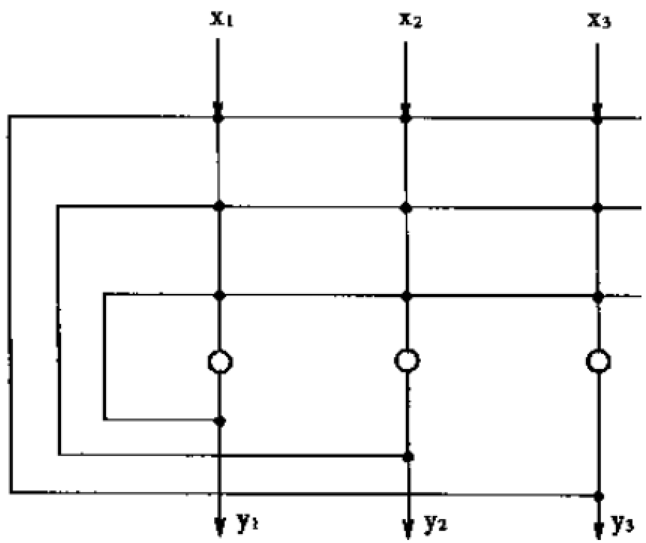
- 应当指出：反馈网络有稳定的，也有不稳定的。对于Hopfield网络，需要判别其稳定性。

四、Hopfield神经网络

❖ 离散Hopfield网络：

Hopfield最早提出的网络是二值神经网络，神经元的输出只取1和0这两个值，也称离散Hopfield神经网络。

离散Hopfield神经网络中，所采用的神经元是二值神经元，因此，所输出的离散值1和0分别表示神经元处于激活和抑制状态。



四、Hopfield神经网络

❖ 离散Hopfield网络（续）：

对于一个离散Hopfield网络，其网络状态是输出神经元信息的集合。

如果输出层为n个神经元，则其t时刻的状态为一个n维向量：

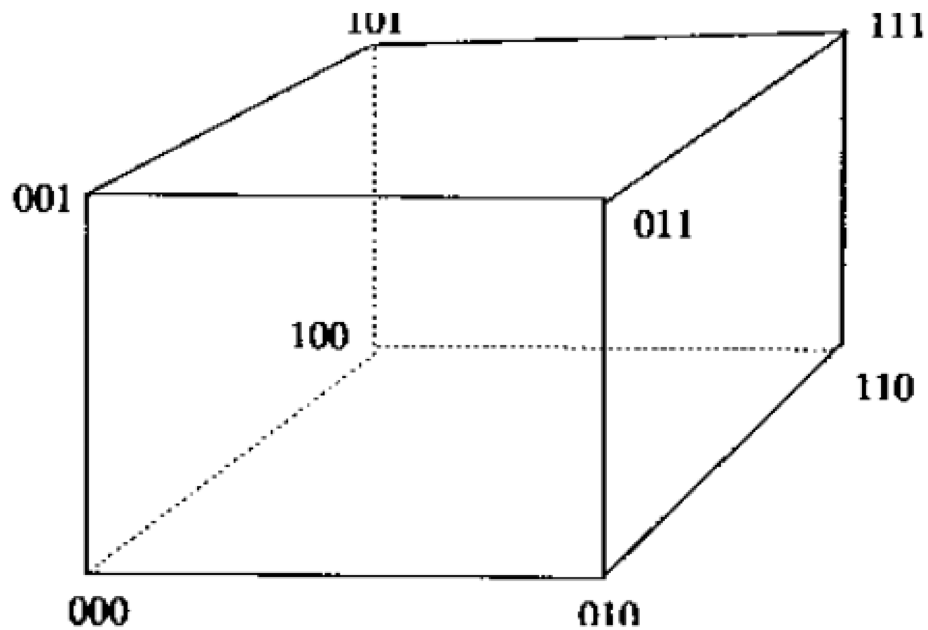
$$Y(t)=[Y1(t),Y2(t),...,Yn(t)]^T$$

因为$Y_j(t)(j=1,\ldots,n)$，取值为1或0，故n维向量Y(t)有2^n个状态。

四、Hopfield神经网络

❖ 离散Hopfield网络（续）：

n=3，8个网络状态。



如果Hopfield网络是一个稳定网络，那么在网络的输入端加入一个输入向量，则网络的状态会发生变化，即从超立方体的一个顶角转移至另一个顶角，并且最终稳定于一个特定的顶角。

四、Hopfield神经网络

❖ 离散Hopfield网络（续）：

对于一个由n个神经元组成的离散Hopfield网络，则有n*n权系数矩阵w：

$$W=\{W_{ij}\}\ i=1,2,...,n\quad j=1,2,...,n$$

同时，有n维阀值向量θ：

$$\theta=[\theta_1, \theta_2,\dots \theta_n]^T$$

一般而言，w和θ可以确定一个唯一的离散Hopfield网络。

四、Hopfield神经网络

❖ 离散Hopfield网络（续）：

✓考虑离散Hopfield网络的一般节点状态，用$Y_j(t)$表示第j个神经元，即节点j在时刻t的状态，则节点的下一个时刻(t+1)的状态可以求出如下：

$$Y_j(t+1) = f[U_j(t)] = \begin{cases} 1 & ,U_j \geqslant 0 \\ 0 & ,U_j < 0 \end{cases}$$

$$U_j(t) = \sum_{i=1}^{n} W_{ij} Y_i(t) + X_j - \theta_j$$

✓当W_{ij}在i＝j时，等于0，则说明一个神经元的输出并不会反馈到它自己的输入；这时，离教的Hopfield网络称为无自反馈网络。

✓当W_{ij}在i＝j时，不等于0，则说明—个神经元的输出会反馈到它自己的输入；这时，离散的Hopfield网络称为有自反馈的网络。

四、Hopfield神经网络

❖ 网络工作方式

1、串行(异步)方式

在时刻t时，只有某一个神经元j的状态产生变化，而其它n-1个神经元的状态不变，这时称串行工作方式。

2、并行(同步)方式

在任一时刻t，所有的神经元的状态都产生了变化，则称并行工作方式。

四、Hopfield神经网络

❖ **连续Hopfield网络：**

✓连续Hopfield网络的拓朴结构和离散Hopfield网络的结构相同。

✓这种拓朴结构和生物的神经系统中大量存在的神经反馈回路是相一致的。

✓在连续Hopfield网络中，和离散Hopfield网络一样，其稳定条件也要求$W_{ij}=W_{ji}$。

✓连续Hopfield网络和离散Hopfield网络不同的地方在于其函数g不是阶跃函数，而是S形的连续函数。

四、Hopfield神经网络

❖ 连续Hopfield网络（续）：

✓当Hopfield网络的神经元传递函数g是连续且有界的，例如Sigmoid函数，并且网络的权系数矩阵对称，则这个连续Hopfield网络是稳定的。

✓在实际应用中，任何一个系统，如果其优化问题可以用能量函数E(t)作为目标函数，那么，总可以用连续Hopfield网络对其进行求解。

四、Hopfield神经网络

❖ 连续Hopfield网络（续）：

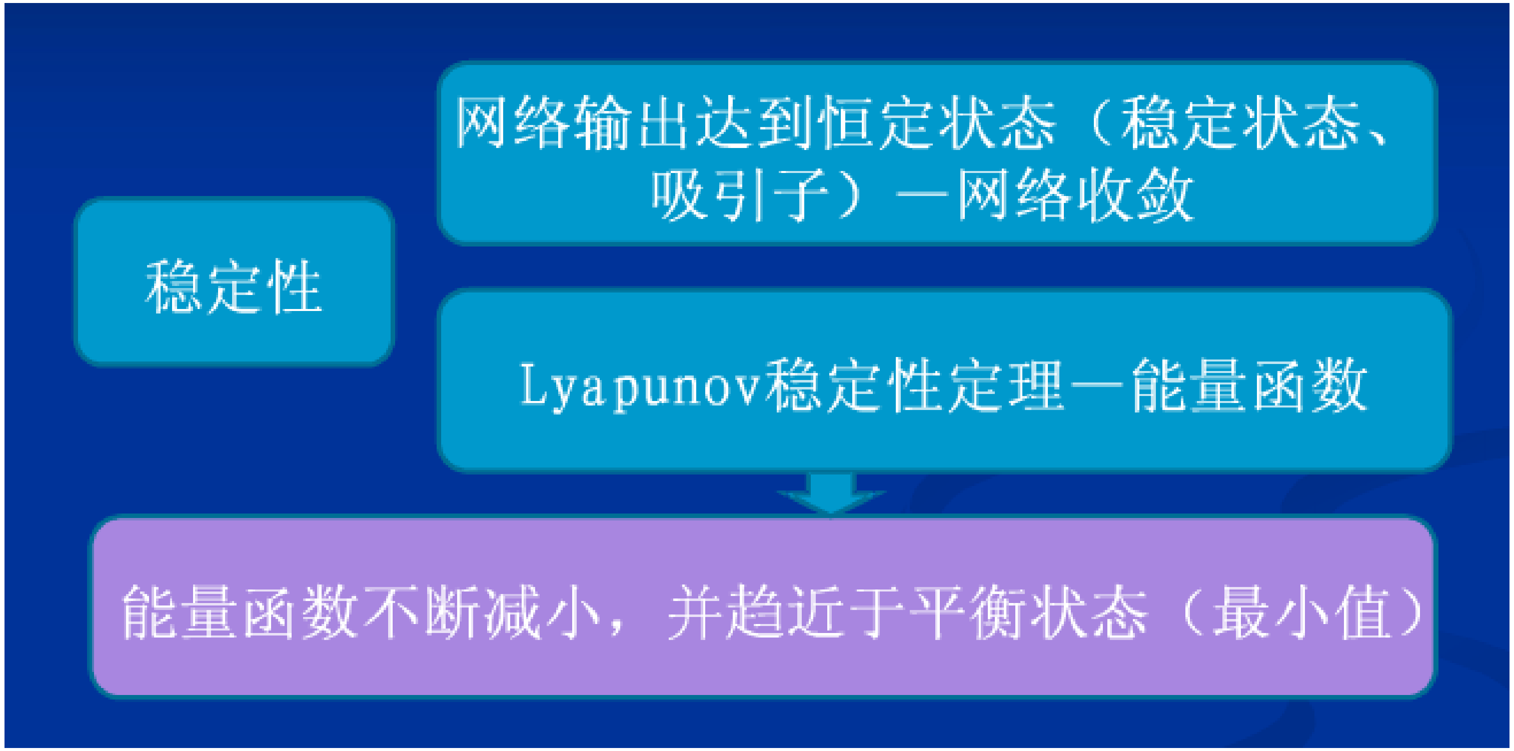
✓由于引入能量函数E(t)，Hopfield使神经网络和问题优化直接对应，这种工作是具开拓性的。

✓利用神经网络进行优化计算，就是在神经网络这一动力系统给出初始的估计点，即初始条件；然后随网络的运动传递而找到相应极小点。

✓这样，大量的优化问题都可以用连续的Hopfield网来求解。这也是Hopfield网络用于神经计算的基本原因。

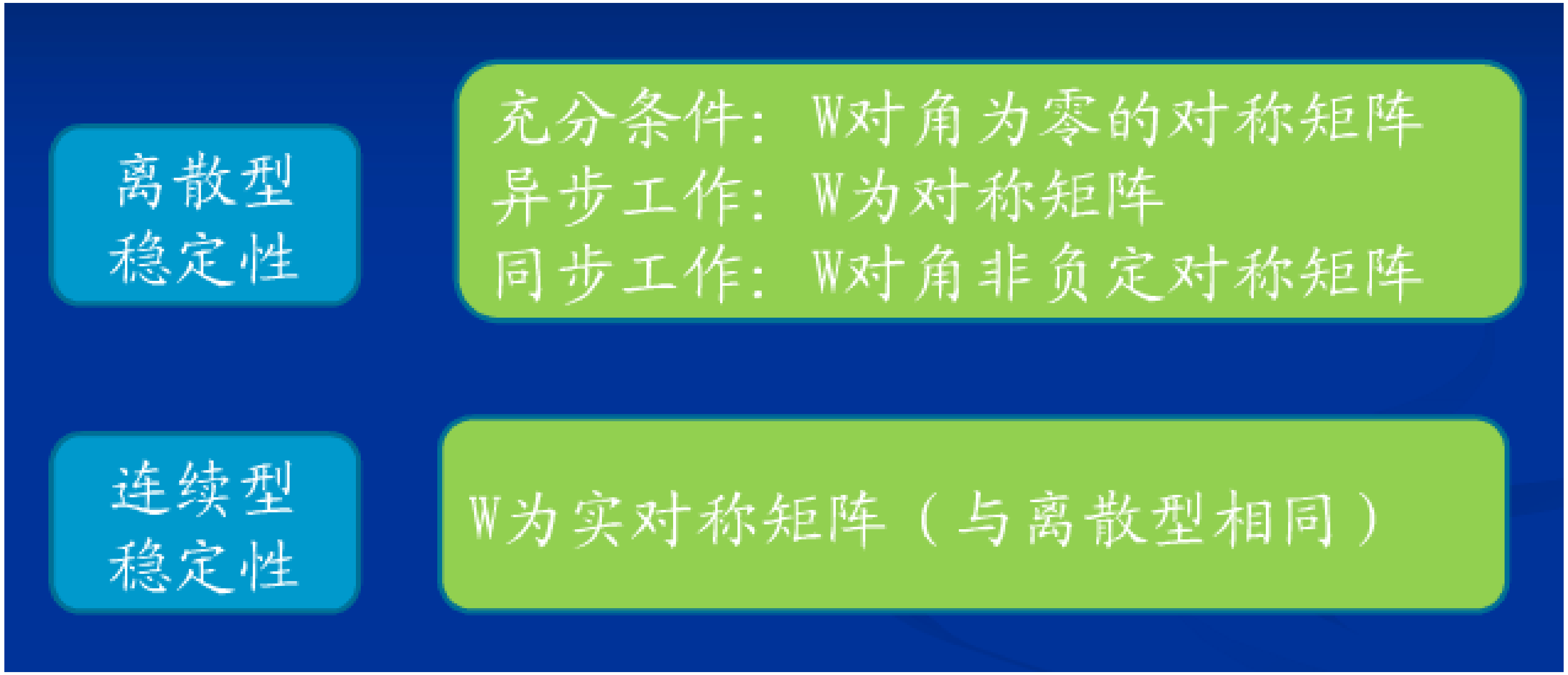
四、Hopfield神经网络

❖ **Hopfield网络的稳定性：**



四、Hopfield神经网络

❖ **Hopfield网络的稳定性（续）：**



四、Hopfield神经网络

❖ Hopfield网络的稳定性（续）：

离散型能量函数	$E(t)=-\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n}\omega_{ij}x_i(t)y_j(t)+\sum_{j=1}^{n}\theta_j y_j(t)$
连续型能量函数	$E(t)=-\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n}\omega_{ij}x_i(t)y_j(t)+\sum_{j=1}^{n}\theta_j y_j(t)+\sum_{j=1}^{n}\frac{1}{\tau}\int_0^{y_j}f^{-1}(y)dy$

四、Hopfield神经网络

❖ Matlab中Hopfield网络的主要函数：

函数名	功　能
satlin( )	饱和线性传递函数
satlins( )	对称饱和线性传递函数
newhop( )	生成一个Hopfield回归网络
nnt2hop( )	更新NNT 2.0 Hopfield回归网络

四、小结

❖ 联系记忆：

因网络能收敛于稳态，故可用于联想记忆。若将稳态视为一个记忆，则由初态向稳态收敛的过程就是寻找记忆的过程，初态认为是给定的部分信息，收敛过程可认为是从部分信息找到了全部信息，实现了联想记忆的功能。联想记忆模型的一个重要特性：由噪声输入模式，反映出训练模式。

❖ 优化计算：

若将稳态视为某一优化计算问题目标函数的极小点，则由初态向稳态收敛的过程就是优化计算过程。先把问题表述成能量函数，进一步由能量函数推出网络权结构，然后在某种条件下让网络运行，网络的稳定状态一般来说就对应与问题的解答。

四、小结

❖ 联系记忆与优化计算的关系：

网络用于计算时W已知，目的是为了寻找稳态；用于联想记忆时稳态是给定的，由学习求得权系值。因此，二者是对偶的。

❖ 网络渐进稳定的前提：

前提是 $w_{ij} = w_{ji}$，否则，系统的运动无准则。

Hopfield反馈神经网络应用于联想与优化计算是相对偶的。应用于优化计算时，网络权矩阵W为已知，目的是寻找具有最小能量值的稳定状态。用作联想时，稳定状态时给定的，目的是通过学习过程来得到合适的权矩阵W。

四、小结

❖ **Hopfield网络应用：**

Hopfield网络已成功的应用在多种场合。从概念上讲，Hopfield网络的运行主要有两种形式。相应的应用方式也主要有两种——联想存取与优化计算。而具体的应用方向主要集中在图像处理、语声处理、控制、信号处理、数据查询、容错计算、模式分类、模式识别和知识处理等领域。

四、小结

❖ 基本思想归纳如下：

1)对于特定的问题，选择一种合适的表示方法，使得神经网络的输出与问题的解对应起来。

2)构造神经网络的能量函数，使其最小值对应问题的最佳解。

3)由能量函数反推出神经网络的结构。

总结

名称	提出者	年代	典型应用领域	局限性	特点
Perceptron(感知器)	**Frank Rosenblatt(康奈尔大学)**	**1958**	文字识别、声音识别、声纳信号识别、学习记忆问题研究	不能识别识别复杂字符，对字的大小、平移和倾斜敏感	最早的神经网络，已很少应用；有学习能力，只能进行线形分类
Adaline(自适应线形单元)和Madaline(多个Adaline的组合网络)	**Bernard Widrow(斯坦福大学)**	**1960 ~ 1962**	雷达天线控制,自适应回波抵消，适应性调制解调，电话线中适应性补偿等	要求输入-输出之间为线性关系	学习能力较强，较早开始商业应用，**Madaline**是**Adaline**的功能扩展
Avalanche(雪崩网)	**S.Drossberg(波士顿大学)**	**1967**	连续语音识别，机器人手臂运动的教学指令	不易改变运动速度和插入运动	
Cerellatron（小脑自动机）	**D.Marr**（麻省理工学院）	**1969 ~1982**	控制机器人的手臂运动	需要复杂的控制输入	类似于**Avalanche**网络，能调和各种指令序列，按需要缓缓地插入动作

总结

名称	提出者	年代	典型应用领域	局限性	特点
Back Propagation(误差反传网络)	**P.Werbos(哈佛大学) David umlhart(斯坦福大学)James MeClelland(斯坦福大学)**	**1974~1985**	语音识别，工业过程控制，贷款信用评估，股票预测，自适应控制等	需要大量输入-输出数据，训练时间长，易陷入局部极小	多层前馈网络，采用最小均方差学习方式，是目前应用最广泛的学习网络
Self-organizing feature map(自组织特征映射网络)	**Tuevo Konhonen(芬兰赫尔辛基技术大学)**	**1980**	语音识别，机器人控制，工业过程控制，图像压缩，专家系统等	模式类型数需预先知道	对输入样本自组织聚类，可映射样本空间的分布
Hopfield网络	**John Hopfield(加州理工学院)**	**1982**	求解TSP问题，线性规划，联想记忆和用于辨识	无学习能力，连接要对称，权值要预先给定	单层自联想网络，可从有缺陷和有噪声输入中恢复完整信息
Boltzman machine(玻尔兹曼机) Cauchy machine(柯西机)	**J.Hinton(多伦多大学) T.Sejnowski(霍尔金斯大学)**	**1985~1986**	图像、声纳和雷达等模式识别	波尔兹曼机训练时间长，柯西机在某些统计分布下产生噪声	采用随机学习算法的网络，可训练实现全局最优

总结

名称	提出者	年代	典型应用领域	局限性	特点
Bidirectional Associative Memory(BAM,双向联想记忆网)	Bart Kosko(南加州大学)	1985~1988	内容寻址的联想记忆	存储的密度低，数据必须适应编码	双向联想式单层网络，具有学习功能，简单易学
Counter Propagation(CPN,双向传播网)	Robert Hecht-Nielsen	1986	神经网络计算机，图像分析和统计分析	需要大量处理单元和连接，需要高度准确	一种在功能上作为统计最优化和概率密度函数分析的网络

上海交通大學 Shanghai Jiao Tong University

总结

名称	提出者	年代	典型应用领域	局限性	特点
Adaptive Resonance Theory(自适应共振理论ART)有ART1、ART2和ART3 3种类型	G.Carpenter and S Grossberg(波士顿大学)	1976～1990	模式识别领域，擅长识别复杂模式或未知的模式	受平移、旋转及尺度的影响；系统比较复杂，难以用硬件实现	可以对任意多和任意复杂的二维模式进行自组织学习，ART1用于二进制，ART2用于连续信号
Brain State in a Box(盒中脑BSB网络)	James Anderson（布朗大学）	1977	解释概念形成，分类和知识处理	只能作一次性决策，无重复性共振	具有最小均方差的单层自联想网络，类似于双向联想记忆，可对片段输入补全
Neocognition(新认知机)	Fukushima K福岛邦彦(日本广播协会)	1978～1984	手写字母识别	需要大量加工单元和联系	多层结构化字符识别网络，与输入模式的大小、平移和旋转无关，能识别复杂字形