第8章 高级神经网络

Chapter 8 Advanced Neural Networks

内容提要

8.1 模糊RBF神经网络

8.2 小脑模型神经网络

8.3 Hopfield神经网络

8.1 模糊RBF网络

- 在模糊系统中，模糊集、隶属度函数和模糊规则的设计是建立在经验知识基础上的。这种设计方法存在很大的主观性。将学习机制引到模糊系统中，使模糊系统能够通过不断学习来修改和完善隶属函数和模糊规则，是模糊系统的发展方向。

- 模糊系统与模糊神经网络既有联系又有区别，其联系表现为模糊神经网络在本质上是模糊系统的实现，其区别表现为模糊神经网络又具有神经网络的特性。

- 采用RBF神经网络模型构成模糊神经网络的最大优势是模糊推理过程和RBF函数具函数等价性，RBF神经网络的功能已被证明能等效于模糊推理系统 (FIS)

8.1 模糊RBF网络

神经网络与模糊系统的比较见下表。模糊神经网络充分地利用了神经网络和模糊系统各自的优点，因而受到了重视。

	模糊系统	神经网络
获取知识	专家经验	算法实例
推理机制	启发式搜索	并行计算
推理速度	低	高
容错性	低	非常高
学习机制	归纳	调整权值
自然语言实现	明确的	不明显
自然语言灵活性	高	低

8.1 模糊RBF网络

- 将神经网络的学习能力引到模糊系统中，将模糊系统的模糊化处理、模糊精确化计算通过分布式的神经网络来表示是实现模糊系统自组织、自学习的重要途径。在模糊神经网络中，神经网络的输入、输出节点用来表示模糊系统的输入、输出信号，神经网络的隐含节点用来表示隶属函数和模糊规则，利用神经网络的并行处理能力使得模糊系统的推理能力大大提高。
- 模糊神经网络在本质上是将常规的神经网络赋予模糊输入信号和模糊权值，其学习算法通常是神经网络学习算法或其推广。模糊神经网络技术已经获得了广泛的应用，当前的应用主要集中在以下几个领域：模糊回归、模糊控制、模糊专家系统、模糊矩阵方程、模糊建模和模糊模式识别。

8.1 模糊RBF网络

- 模糊神经网络是将模糊系统和神经网络相结合而构成的网络。利用RBF网络与模糊系统相结合，构成了模糊RBF网络。

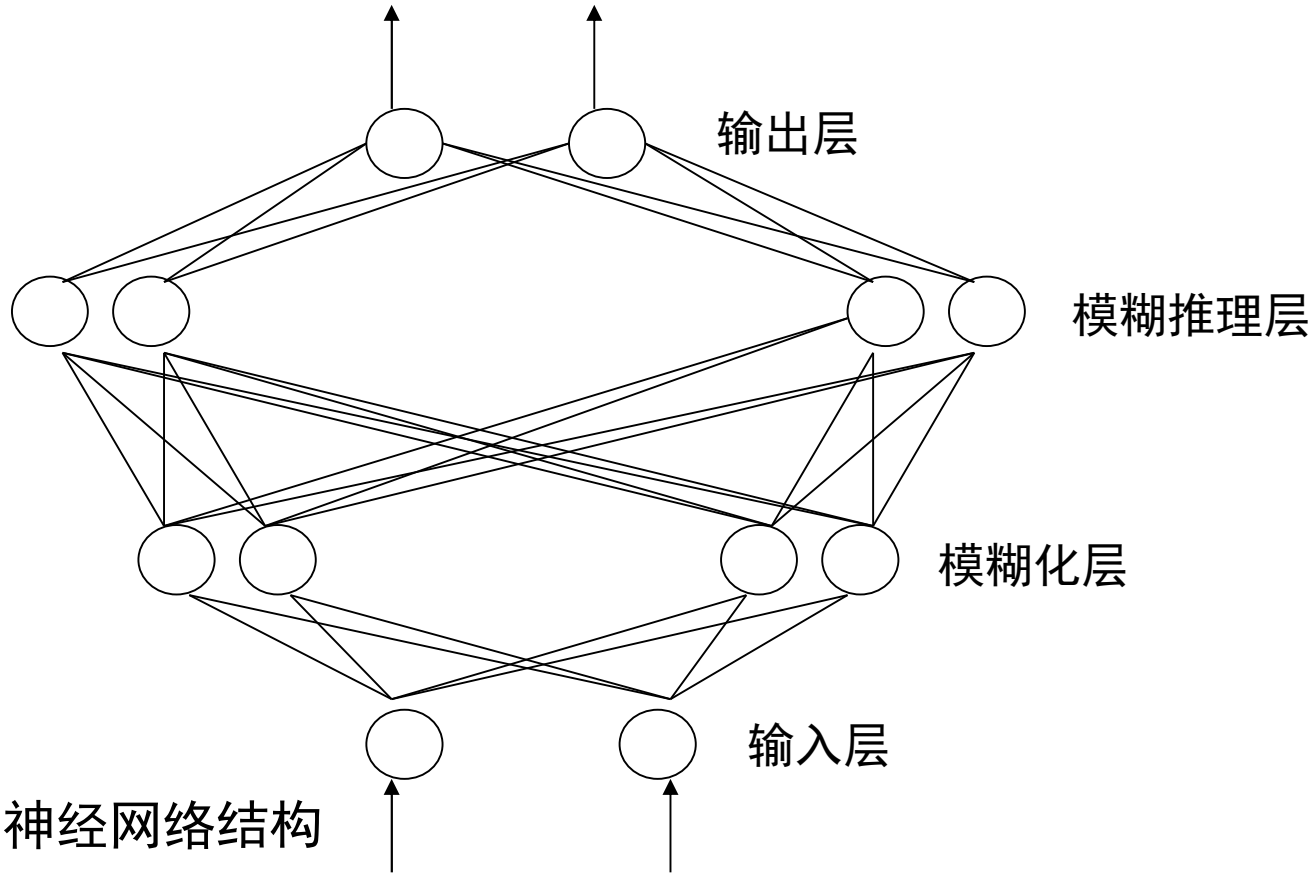


图8-1 模糊RBF神经网络结构

8.1 模糊RBF网络

（一）网络结构

采用图8-1所示的模糊RBF神经网络系统，其模糊推理系统主要由输入层、模糊化层、模糊推理层和输出层构成。

网络中信号传播及各层的功能表示如下：

第一层：输入层

该层的各个节点直接与输入量的各个分量连接，将输入量传到下一层。对该层的每个节点i的输入输出表示为：

$$f_1(i) = x_i$$

8.1 模糊RBF网络

第二层：模糊化层，即隶属函数层

该层的每个节点具有隶属函数的功能，采用高斯型函数作为隶属函数。对第j个节点：

$$f_2(i,j) = \exp(net_j^2) = \mu_j$$

$$net_j^2 = -\frac{(f_1(i) - c_{ij})^2}{(b_j)^2}$$

其中c_{ij}和b_j分别是第i个输入变量的第j个模糊集合高斯函数的均值和标准差。

8.1 模糊RBF网络

第三层：规则层，即模糊推理层

该层通过与模糊化层的连接来完成模糊规则的匹配，各个节点之间实现模糊运算，即通过各个模糊节点的组合得到相应的点火强度。每个节点j的输出为该节点所有输入信号的乘积，即

$$f_3(j)=\prod_{j=1}^{N} f_2(i,j)$$

或

$$f_3(j)=\min\{f_2(i,1), f_2(i,2),\ldots, f_2(i,N)\}$$

其中 $N=\prod_{i=1}^{n} N_i$，N_i为输入层中第i个输入隶属函数的个数，即模糊化层节点数。

8.1 模糊RBF网络

第四层：输出层

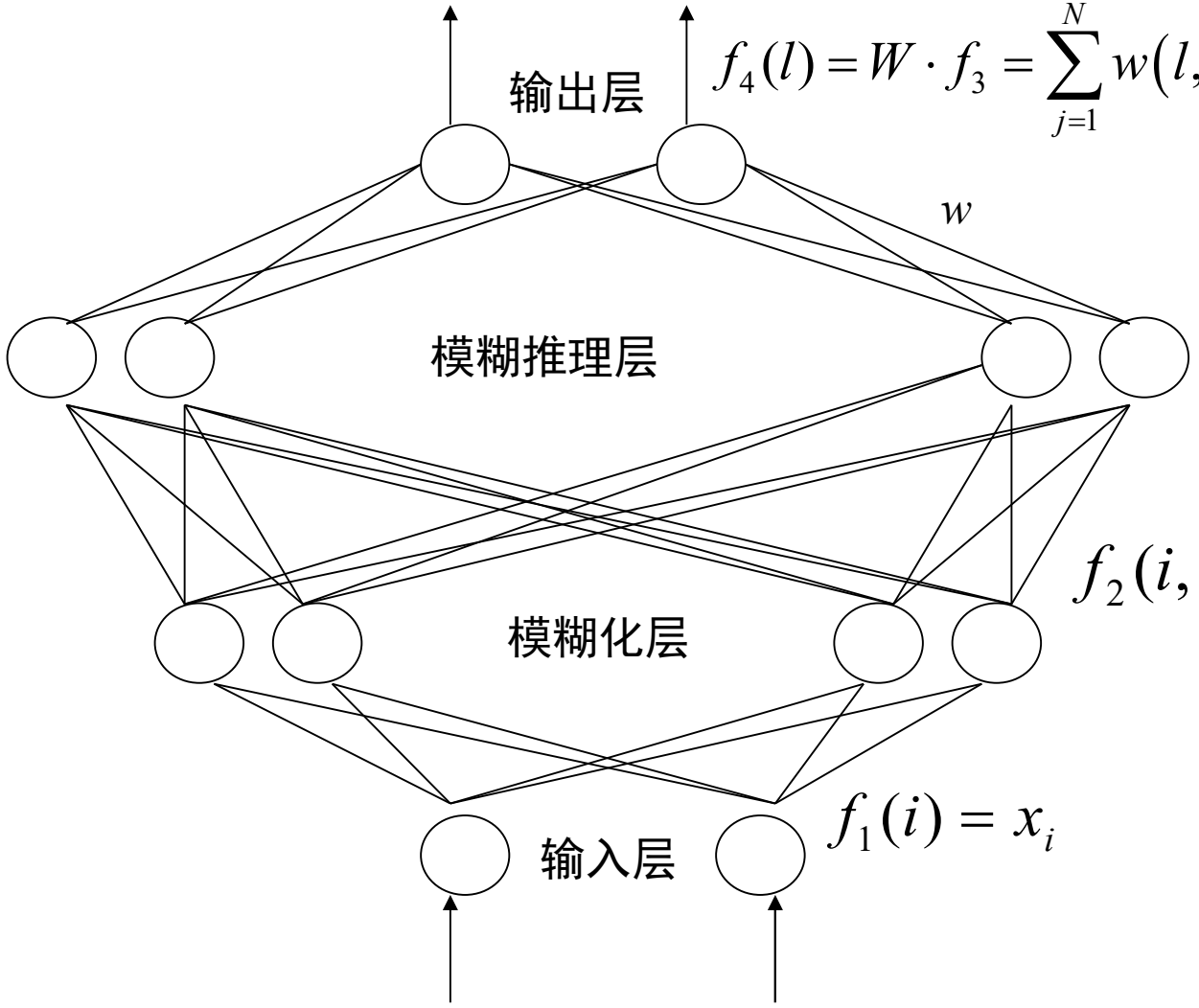
该层的每个节点的输出为该节点所有输入信号的加权和，即

$$f_4(l) = W \cdot f_3 = \sum_{j=1}^{N} w(l,j) \cdot f_3(j)$$

其中l为输出层节点的个数，W为输出节点与第三层各节点的连接权矩阵。

在此模糊神经网络中，可调参数有三类：一类为规则的权系数w；第二类和第三类为高斯函数的均值和标准差c_{ij}和b_j，即输入隶属函数的参数。

8.1 模糊RBF网络



$$f_4(l) = W \cdot f_3 = \sum_{j=1}^{N} w(l,j) \cdot f_3(j)$$

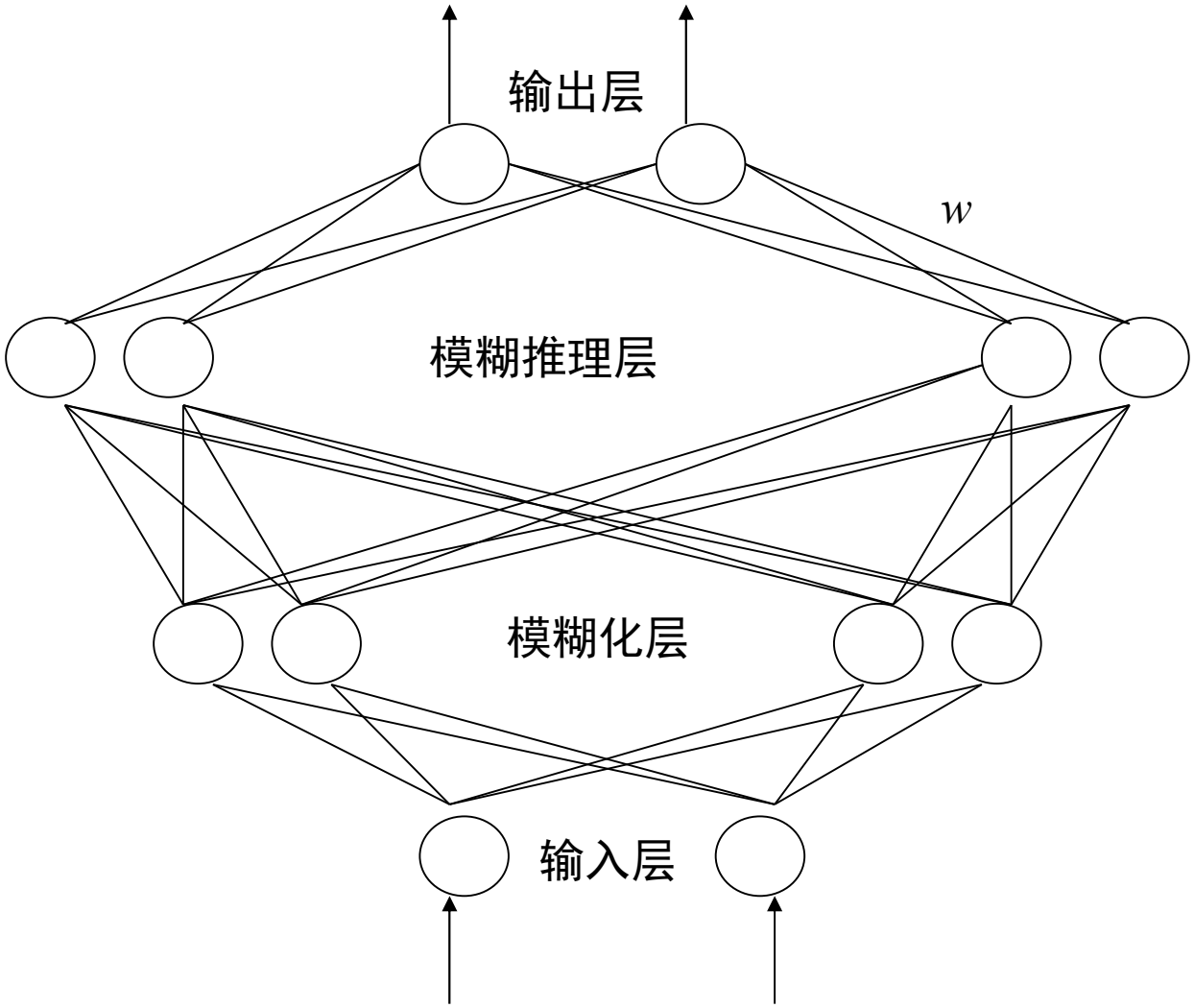
$$f_3(j) = \prod_{j=1}^{N} f_2(i,j)$$

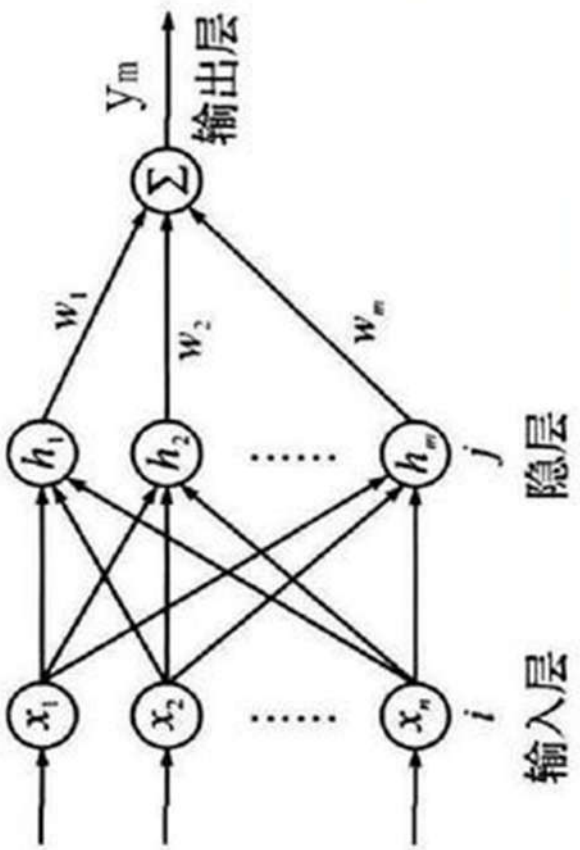
$$f_2(i,j) = \exp(net_j^2) = \mu_j$$

$$f_1(i) = x_i$$

图8-1 模糊RBF神经网络结构

8.1 模糊RBF网络

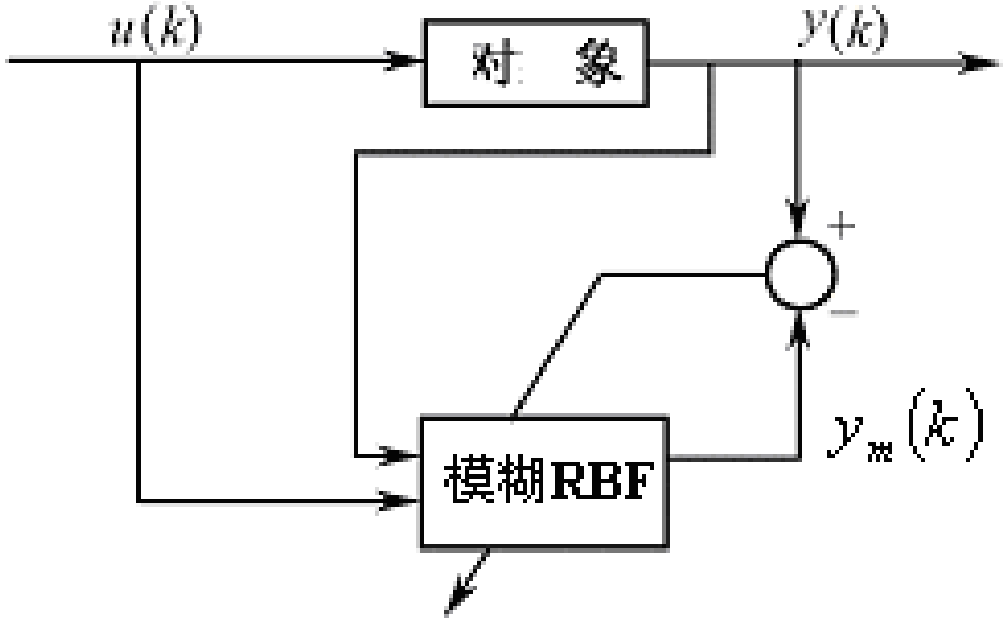




8.1 模糊RBF网络

（二）基于模糊RBF网络的逼近算法

采用模糊RBF网络逼近对象，取网络结构为2-4-1，如图所示。



8.1 模糊RBF网络

取 $y_m(k)=f_4$，$y_m(k)$和$y(k)$分别表示网络输出和理想输出。网络的输入x_1和x_2为$u(k)$和$y(k)$，则网络逼近误差为：

$$e(k)=y(k)-y_m(k)$$

采用梯度下降法来修正可调参数，定义目标函数为：

$$E(k)=\frac{1}{2}e(k)^2$$

8.1 模糊RBF网络

网络的学习算法如下：

输出层的权值通过如下方式来调整：

$$\Delta w(k) = -\eta \frac{\partial E}{\partial w} = -\eta \frac{\partial E}{\partial e} \frac{\partial e}{\partial y_m} \frac{\partial y_m}{\partial w} = \eta e(k) f_3$$

则输出层的权值学习算法为：

$$w(k) = w(k-1) + \Delta w(k) + \alpha \left(w(k-1) - w(k-2) \right)$$

其中η为学习速率，α为动量因子。

8.1 模糊RBF网络

在RBF网络设计中，需要将c_{ij}和b_j值设计在网络输入有效的映射范围内，否则高斯基函数将不能保证实现有效的映射，导致RBF网络失效。如果将c_{ij}和b_j的初始值设计在有效的映射范围内，则只调节网络的权值便可实现RBF网络的有效学习。

8.1 模糊RBF网络

（三）仿真实例

使用模糊RBF网络逼近对象：

$$y(k) = u(k)^3 + \frac{y(k-1)}{1 + y(k-1)^2}$$

其中采样时间为1ms。

模糊RBF网络逼近程序见chap8_1.m。

8.1 模糊RBF网络

输入信号为$u(k)=sin(0.1t)$，神经网络权值W的初始值取[-1,+1]之间的随机值，考虑到网络的第一个输入范围为[-1,+1]，离线测试可得第二个输入范围为[-1.5,+1.5]，高斯基函数的参数取值为

$$\boldsymbol{c}=\left[\boldsymbol{c}_{ij}\right]=\begin{bmatrix}-1 & -0.5 & 0 & 0.5 & 1\\ -2 & -1 & 0 & 1 & 2\end{bmatrix}^{\mathrm{T}}$$

$$b_j=3.0 \qquad i=1,2 \qquad j=1,2,3,4,5$$

网络的学习参数取

$$\eta=0.50 \qquad \alpha=0.05$$

8.1 模糊RBF网络

模糊RBF网络逼近程序见chap8_1.m。仿真结果如图8-3和8-4所示。由于本方法采用梯度下降法来调节权值，只能保证局部范围内的逼近，无法保证全局逼近。限于梯度下降法的局限性，本方法只适合慢时变系统的逼近，且学习速率不能过大。

8.1 模糊RBF网络

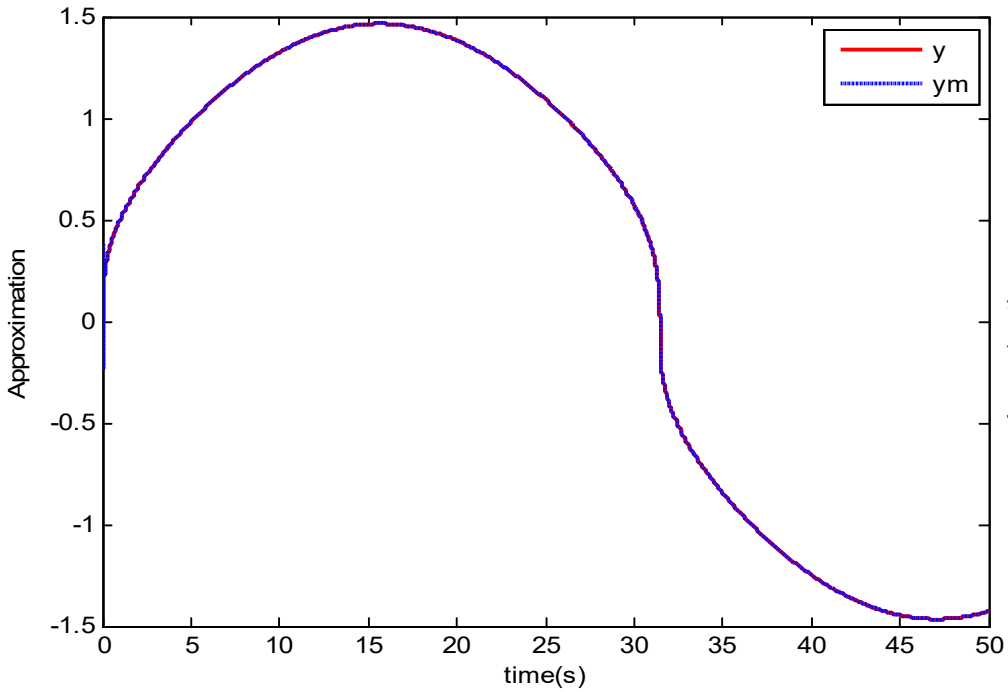


图8-3 模糊RBF网络逼近效果

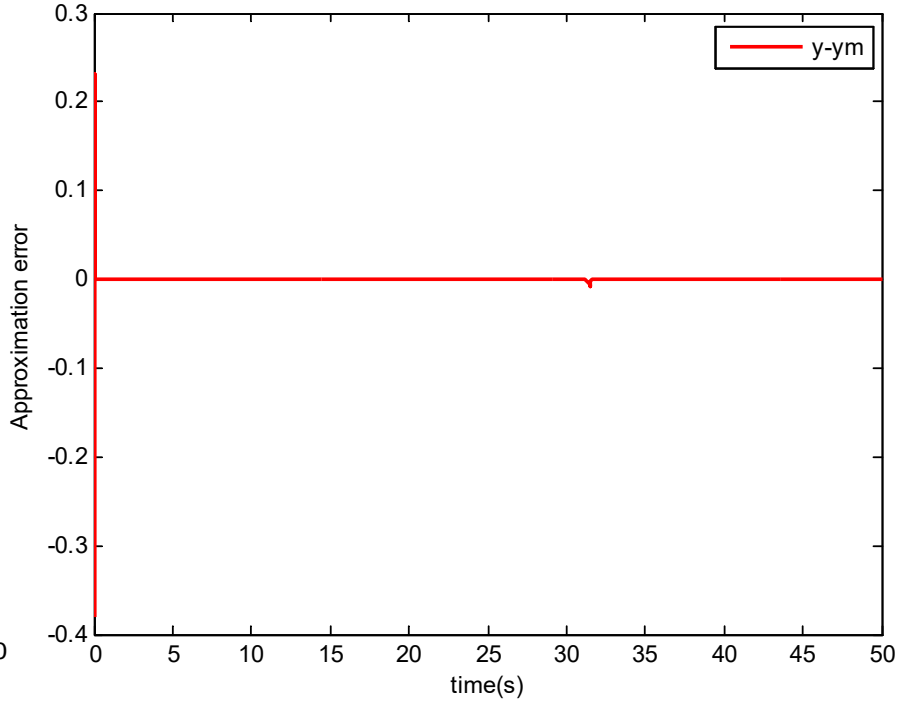


图8-4 模糊RBF网络逼近误差

8.2 小脑模型神经网络

（一）CMAC概述

人的小脑具有管理和协调运动功能。小脑皮层的神经系统从肌肉、四肢、关节、皮肤等接受感受信息，并感受反馈信息，然后将这些获取的信息整合到一特定的存储区域记忆着。

当需要的时候，将这些存储的信息从中取出，作为驱动和协调肌肉运动的控制指令。当感受信息和反馈信息出现差异，通过联想调整达到协调运动控制的目的，这一过程就是学习。

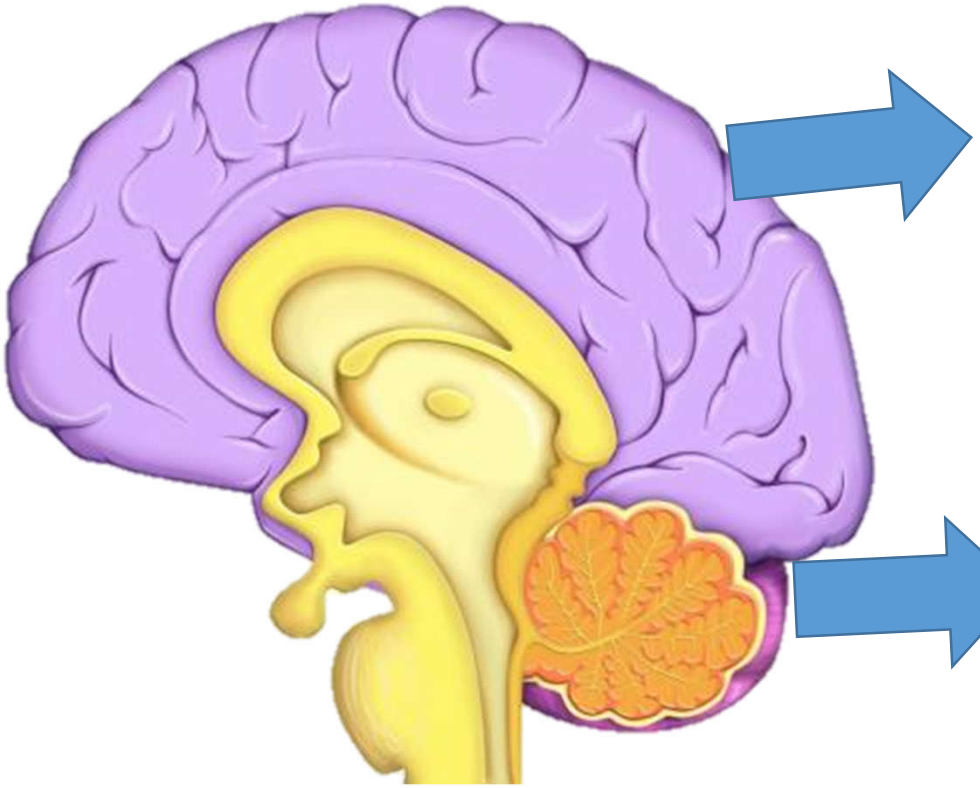
8.2 小脑模型神经网络

小脑模型神经网络（CMAC-Cerebellar Model Articulation Controller，即小脑模型关节控制器）是一种表达复杂非线性函数的表格查询型自适应神经网络，Albus最初于1975年基于神经生理学提出，该网络可通过学习算法改变表格的内容，具有信息分类存储的能力。

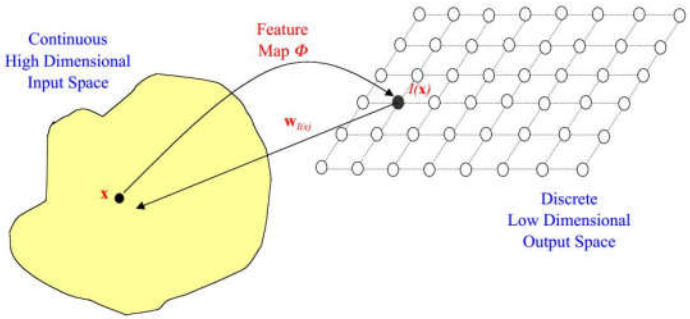
CMAC已被公认为是一类联想记忆网络的重要组成部分，能够学习任意多维非线性映射，CMAC算法被证明可有效地用于非线性函数逼近、动态建模、控制系统设计等。

CMAC最初主要用来求解机械手的关节运动，后来被进一步应用于机器人控制、模式识别、信号处理以及自适应控制等领域。

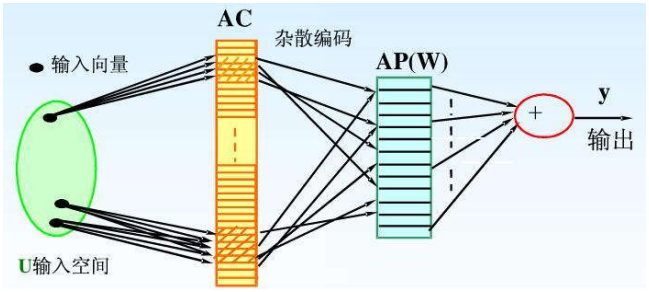
8.2 小脑模型神经网络



大脑模型→自组织网络



小脑模型→ CMAC神经网络



小脑承担运动控制中**预期性的前馈控制**和**基于误差的监督学习**这两种功能！

8.2 小脑模型神经网络

一般CMAC网络的整体结构如下图，第一层是输入层；第二层是虚拟联想空间；第三层是物理存储空间，即根据第二层给出的索引找到对应的权值；第四层即输出层。

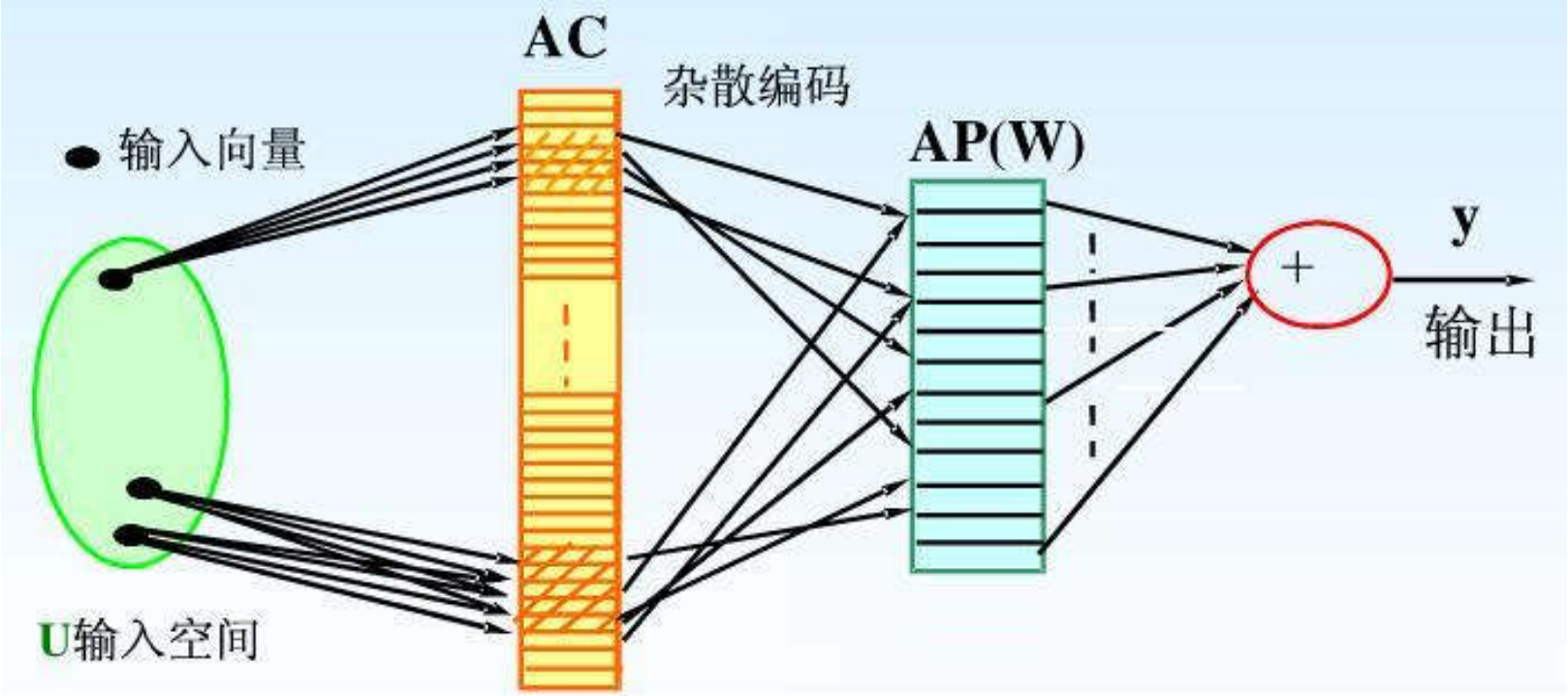


图8-5 CMAC神经网络结构

8.2 小脑模型神经网络

CMAC具有以下的优点：

（1）小脑模型是基于局部学习的神经网络，它把信息存储在局部结构上，使每次修正的权极少。在保证函数非线性逼近性能的前提下，学习速度快，适合于实时控制；

（2）具有一定的泛化能力，即所谓相近输入产生相近输出，不同输入给出不同输出；

（3）具有连续（模拟）输入输出能力；

8.2 小脑模型神经网络

(4)　采用寻址编程方式，在利用串行计算机仿真时，它将使响应速度加快；

(5)　CMAC函数非线性逼近器对学习数据出现的次序不敏感。

由于CMAC所具有的上述优越性能，使它比一般神经网络具有更好的非线性逼近能力，更适合于复杂动态环境下非线性实时控制的要求。

8.2 小脑模型神经网络

（二）一种典型CMAC算法

CMAC网络由输入层，中间层和输出层组成。在输入层与中间层、中间层与输出层之间分别为由设计者预先确定的输入层非线性映射和输出层权值自适应性线性映射。

CMAC神经网络的设计主要包括输入空间的划分、输入层至输出层非线性映射的实现及输出层权值学习算法。

8.2 小脑模型神经网络

CMAC是前馈网络，输入输出之间的非线性关系由以下两个基本映射实现。

（1）概念映射（U→AC）

概念映射是从输入空间U至概念存储器AC的映射。考虑单输入映射至AC中c个存储单元的情况。取u(k)作为网络输入，采用如下线性化函数对输入状态进行量化，实现CMAC的概念映射：

8.2 小脑模型神经网络

AC中的地址

U中的输入

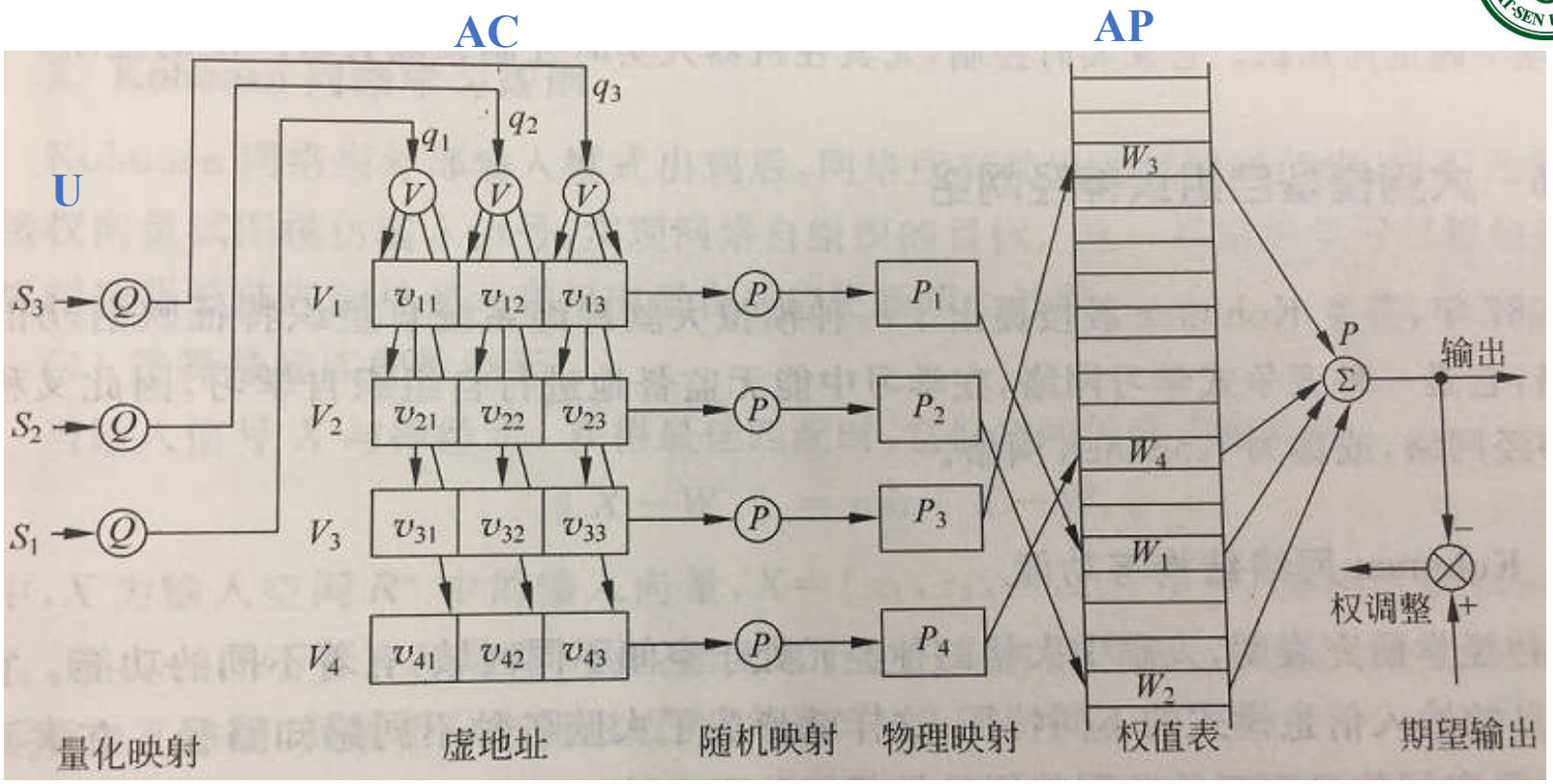
$$s_i(k)=\text{round}\left(u(k)-x_{\min}\right)\frac{M}{x_{\max}-x_{\min}}+i$$

式中$x_{\max}$和$x_{\min}$为输入的最大和最小值，M为$x_{\max}$量化后所对应的初始地址，round(·)为四舍五入函数，i=1, 2, ⋯, c。

由上式可见，当$u(k)$为$x_{\min}$时，$u(k)$映射地址为1, 2, ⋯, c；当$u(k)$为$x_{\max}$时，$u(k)$映射地址为M+1, M+2, ⋯, M+c。

映射原则为：在输入空间邻近的两个点，在AC中有部分的重叠单元被激励。距离越近，重叠越多；距离越远的点，在AC中不重叠，这称为局域泛化，c为泛化常数。

8.2 小脑模型神经网络



离散输入量到虚地址的映射，映射后变成三段合成的四个虚地址，即当某一输入量化值变化一个等级时，只有一个虚地址段变化为1，而其他都保持不变，这样保证了在相邻量化值的指示权值的4个虚拟地址中有3个是相同的。意味着在输入空间中相近的输入量能给出相近的输出。

8.2 小脑模型神经网络

（2）实际映射 （AC→AP）

实际映射是由概念存储器AC中的c个单元映射至实际存储器AP的c个单元，c个单元中存放着相应权值。网络的输出为AP中c个单元的权值的和。

采用散列编码技术中除留余数法实现CMAC的实际映射。设散列表长为m，以虚拟地址$s_i(k)$除以某数N后所得余数+1作为物理地址，实现实际映射，即

$$ad(i)=\left(s_i(k)\quad MOD\quad N\right)+1$$

式中，MOD为取余的Matlab函数, i=1, 2, ⋯, c。

8.2 小脑模型神经网络

若只考虑单输出，则输出为

$$y_n = \sum_{i=1}^{c} w\left(ad(i)\right)$$

```
for i=1:1:c
   s(k,i)=round((u(k)-xmin)*M/(xmax-xmin))+i;   % Quantity: U→AC
   ad(i)=mod(s(k,i),N)+1;                       % Hash transfer: AC → AP
end
```

CMAC采用的学习算法如下：

采用梯度下降法调整权值，权值调整指标为

$$E(k) = \frac{1}{2} e(k)^2$$

其中 $e(k) = r(k) - y_n(k)$

8.2 小脑模型神经网络

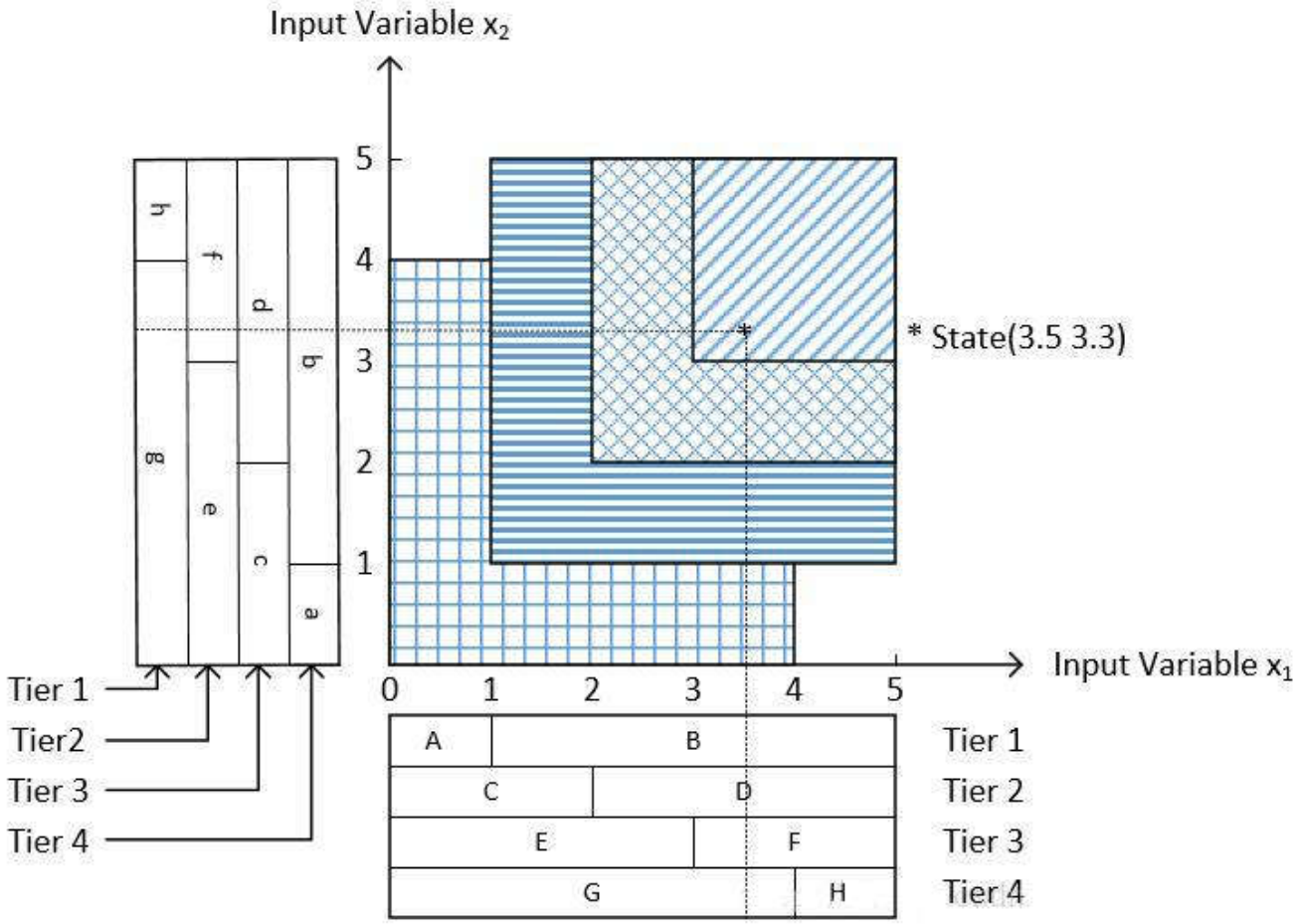
由梯度下降法，权值按下式调整：$\frac{\partial y_n}{\partial w_j}=\frac{\partial\sum_{i=1}^{c} w\left(ad(i)\right)}{\partial w_j}=1$

$$\Delta w_j(t)=-\eta\frac{\partial E}{\partial w_j}=\eta(y(t)-y_n(t))\frac{\partial y_n}{\partial w_j}=\eta e(t)$$

$$w_j(\mathrm{t})=w_j(\mathrm{t}\text{-}1)+\Delta w_j(t)+\alpha(w_j(t-1)-w_j(t-2))$$

其中α为惯性系数/动量因子，$j=ad(\mathrm{i}),\mathrm{i}=1,2,\cdots,\mathrm{c}$ 。

8.2 小脑模型神经网络



假设输入是二维，即$X=(x1, x2)$，$x1, x2 \in [0, 5]$（即使不是也可以先标准化，$[0, 5]$不是固定的，可以是其他）。

8.2 小脑模型神经网络

(1) 每一维的输入xi都具有层m和块nb的概念（注意，这个层的意思和一般神经网络层的意思不一样）。在图中Tier是层的意思，也就是每一维具有m=4层，每层有nb=2块，例如第一层Tier1具有两个块A、B。根据m和nb确定每一维等分切片的个数，即$m\cdot(nb-1)+1$，例如上图中，每维被切分为4*(2-1)+1=5等份。

(2) 输入的维度之间，相同的层所激活的块联合起来对应一个权值地址空间。如上图，当前X=(3.5,3.3)，x1在不同层激活的块是B、D、F、G，x2是b、d、f、g，此时所对应的权值索引是Bb、Dd、Ff、Gg，把这4个权值加起来就是输出。由此可以看出，有多少层就激活多少个权值。同时，不存在同维度块联合、不同维度不同层块联合的情况，也就是说不存在AB、AC、Ad等情况。由此算来，可能使用到的权值个数为$m\cdot nb^n$，n为输入的维度。

8.2 小脑模型神经网络

（3）如果输入维度很大，根据$m \cdot nb^n$，权值个数指数增长，然而也许只有少部分的权值被使用（有些权值从来没被激活），所以可以使用哈希表方法存储权值，减小无用空间的开支。

（4）权值更新公式为

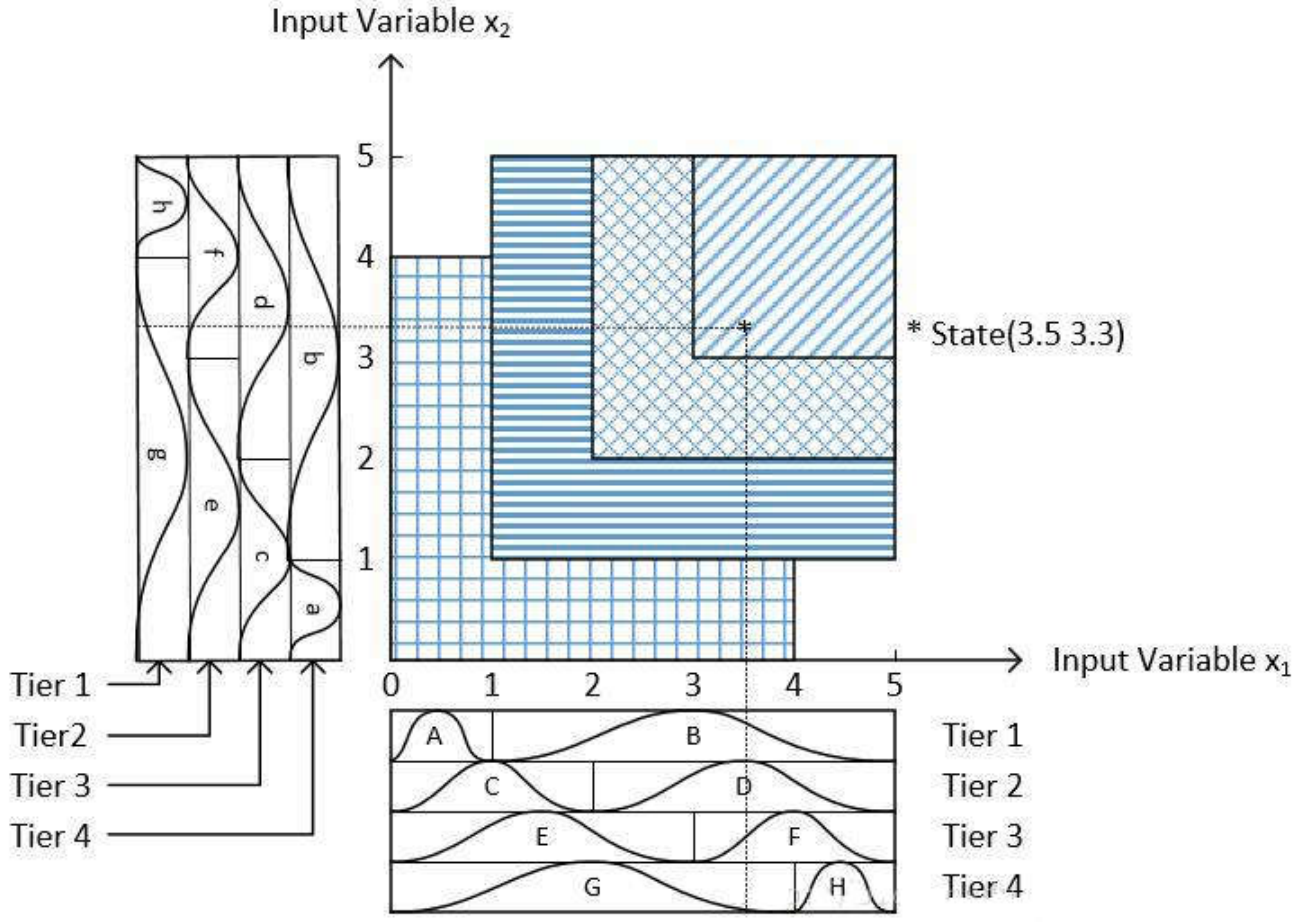
$$w_{t+1}=w_t+\frac{\alpha}{m}\cdot e$$

α为学习率，m是层的个数，e为样本真值与网络预测的误差，只有激活的权值被更新。可以把m拿掉，看成使用一般神经网络的梯度下降法。

8.2 小脑模型神经网络

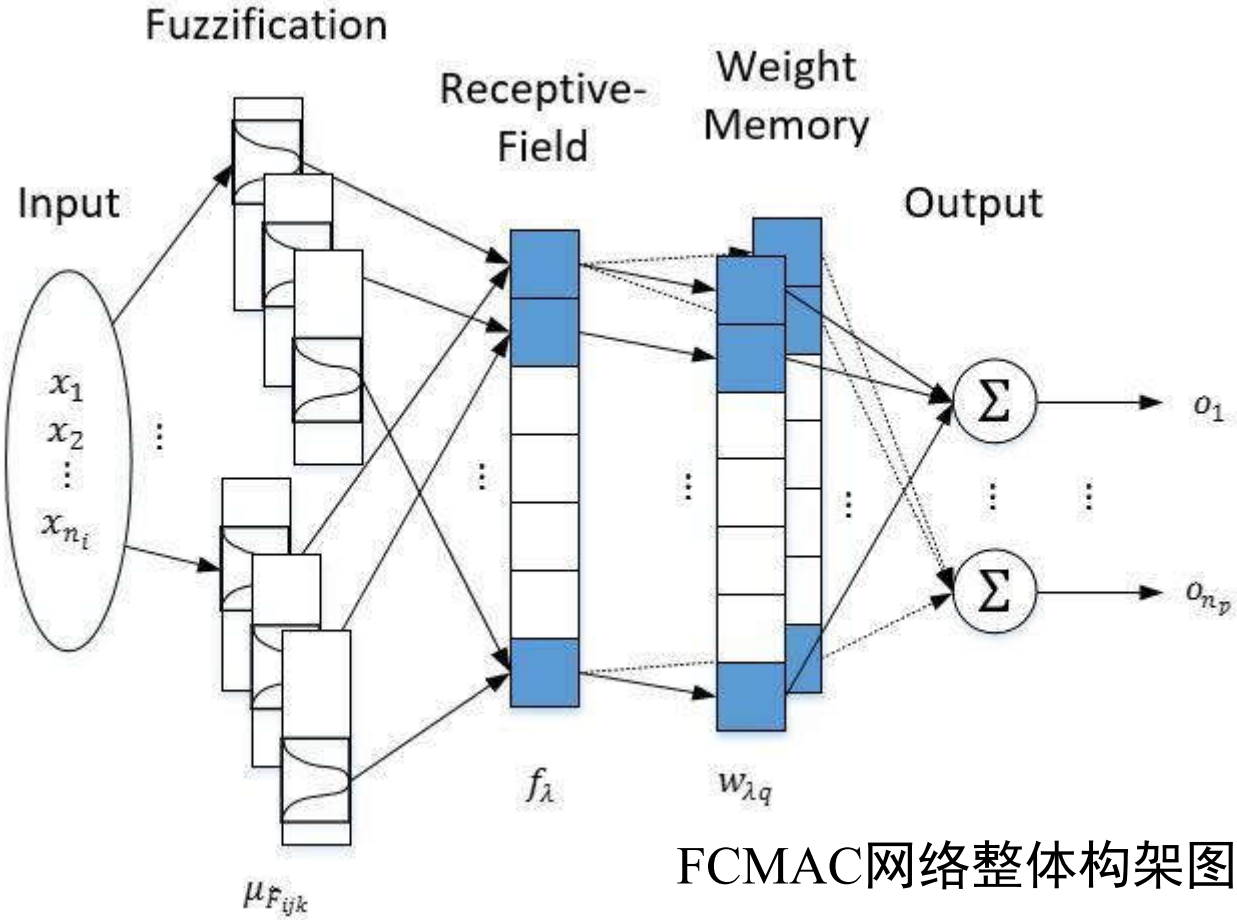
模糊小脑神经网络（FCMAC）结构

FCMAC实际上就是在每个块里面加入了一个模糊隶属度函数，例如如果使用的是高斯模糊，则是在块里加入高斯函数。



8.2 小脑模型神经网络

高斯函数的均值μ和标准差σ可以固定，也可以是可调整的，类似CMAC的权值更新方法，用求梯度的方法就可以迭代更新μ和σ了，当然，只有被激活的块才会更新。



FCMAC网络整体构架图

8.2 小脑模型神经网络

（三）仿真实例

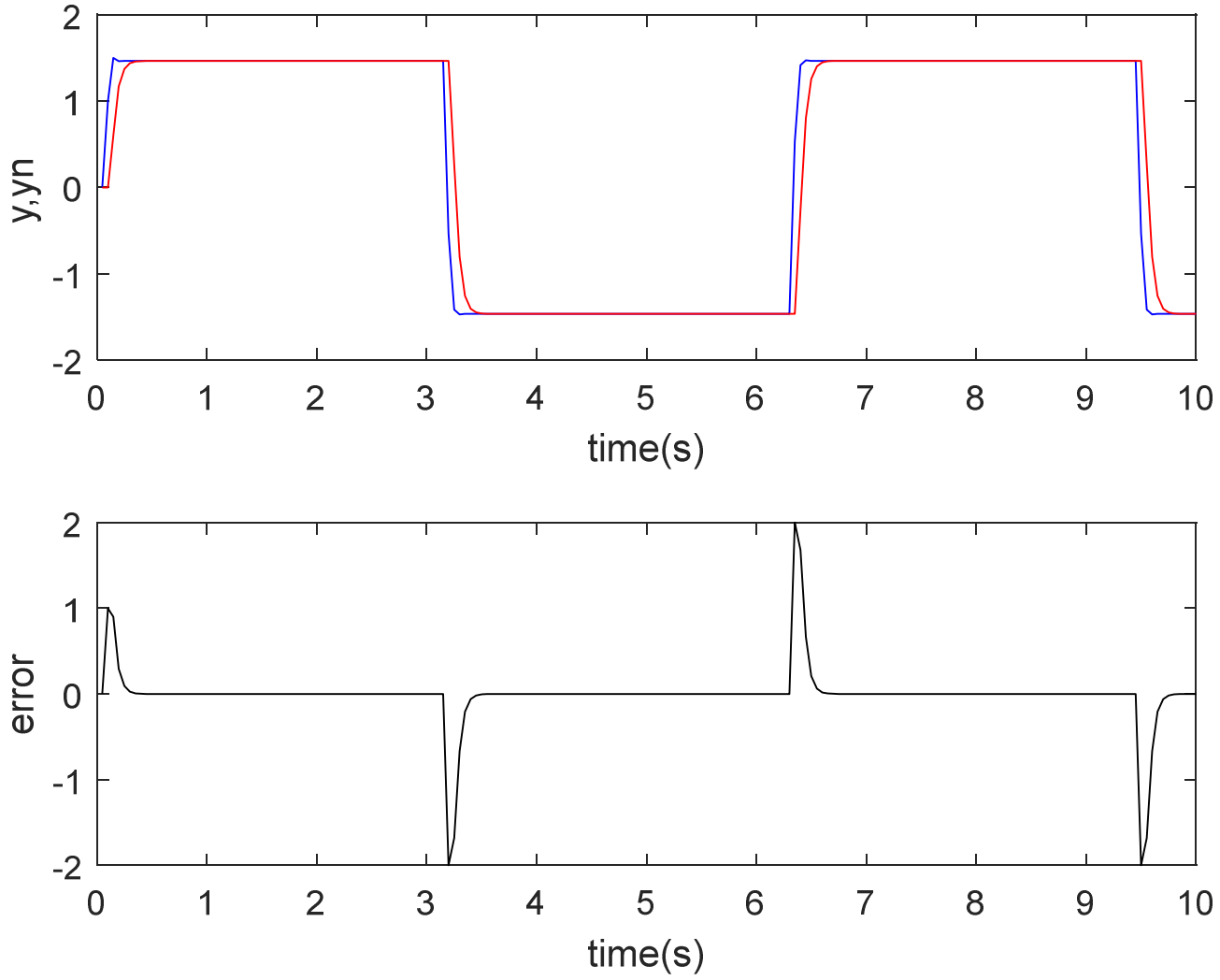
采用CMAC网络逼近非线性对象

$$y(k) = u(k-1)^3 + y(k-1)/(1+y(k-1)^2)$$

在仿真中，取M=200，N=100，取泛化参数c=3，η=0.20，α=0.05，可保证$c \ll M$及$c \le N \le M$。

CMAC网络逼近程序为chap8_2.m。

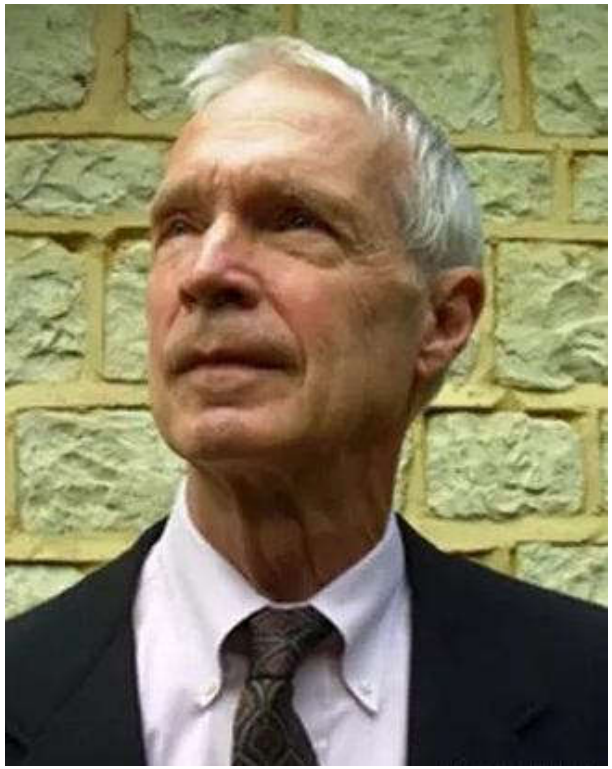
8.2 小脑模型神经网络



8.3 Hopfield神经网络

（一）Hopfield网络原理

1986年美国物理学家J.J.Hopfield利用非线性动力学系统理论中的能量函数方法研究反馈人工神经网络的稳定性，提出了Hopfield神经网络，并建立了求解优化计算问题的方程。



8.3 Hopfield神经网络

- 从学习的观点来看，前向神经网络是一个强有力的学习系统，结构简单，易于实现；从系统的观点来看，是一个非线性映射，通过简单非线性处理单元的复合映射可获得非线性处理能力。
- 从计算的观点来看，它不是一个强有力的系统，缺乏丰富的动力学行为。反馈神经网络是一种反馈动力学系统，具有较强的计算功能。
- 从历史发展来看，动力学系统的概念本身来源于常微分方程定性理论。一个动力学系统是在给定空间φ中，对所有点x随时间变化所经过的路径的描述。
- Hopfield神经网络模型是由一系列互联的神经元组成的反馈型网络，分为连续型（CHNN）和离散型（DHNN）两种。

8.3 Hopfield神经网络

基本的Hopfield神经网络是一个由非线性元件构成的全连接型单层反馈系统，Hopfield网络：

- 每一个神经元都将自己的输出通过连接权传送给所有其它神经元，
- 同时又都接收所有其它神经元传递过来的信息，
- 是一个反馈型神经网络，网络中的神经元在t时刻的输出状态实际上间接地与自己的t-1时刻的输出状态有关，其状态变化可以用差分方程来描述。

8.3 Hopfield神经网络

- 反馈型网络的一个重要特点就是它具有稳定状态，当网络达到稳定状态的时候，也就是它的能量函数达到最小的时候。
- Hopfield神经网络的能量函数不是物理意义上的能量函数，而是在表达形式上与物理意义的能量概念一致，表征网络状态的变化趋势，并可以依据Hopfield工作运行规则不断进行状态变化，最终能够达到的某个极小值的目标函数。网络收敛就是指能量函数达到极小值。
- 如果把一个最优化问题的目标函数转换成网络的能量函数，把问题的变量对应于网络的状态，那么Hopfield神经网络就能够用于解决优化组合问题。

8.3 Hopfield神经网络

- Hopfield神经网络工作时，各个神经元的连接权值是固定的，更新的只是神经元的输出状态。

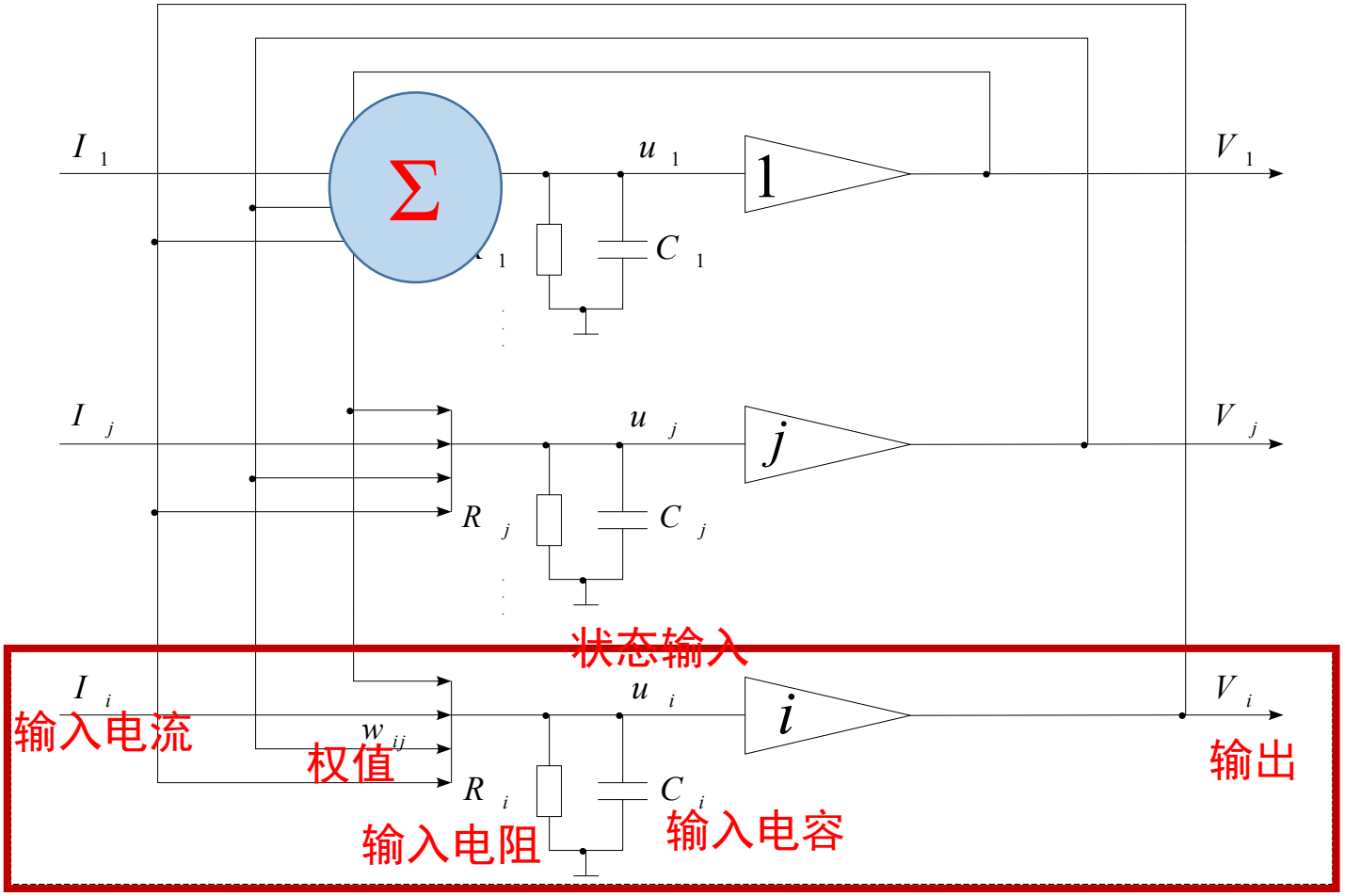
- Hopfield神经网络的运行规则：

  首先从网络中随机选取一个神经元u_i进行加权求和，再计算u_i的下一时刻的输出值。除u_i以外的所有神经元的输出值保持不变，直至网络进入稳定状态。

  当有输入之后，可以求得Hopfield网络的输出，这个输出反馈到输入从而产生新的输出，这个反馈过程一直进行下去。如果Hopfield网络是一个能收敛的稳定系统，则这个反馈与迭代的计算过程所产生的变化越来越小，一旦达到了稳定平衡状态， Hopfield网络就会输出一个稳定的恒值。

8.3 Hopfield神经网络

Hopfield用模拟电路（运算放大器）模仿生物神经网络的特性：



8.3 Hopfield神经网络

其中w_{ij}为第j个神经元到第i个神经元的连接权值，v_i为神经元的输出，是神经元状态变量u_i的非线性函数。当给定了网络的初始状态和外部输入后，Hopfield网络将形成一个动态系统。

对于Hopfield神经网络第i个神经元，采用微分方程建立其输入输出关系，即：

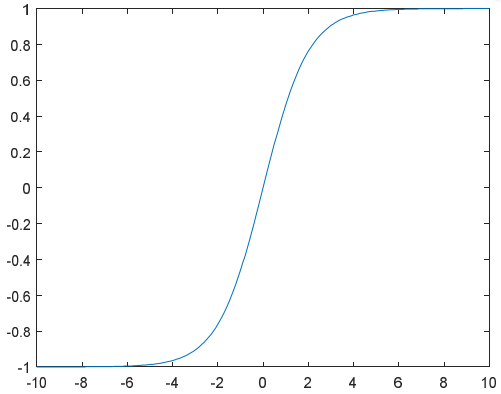
$$\begin{cases} C_i \dfrac{du_i}{dt} = \sum_{j=1}^{n} w_{ij} v_j - \dfrac{u_i}{R_i} + I_i \\ v_i = g(u_i) \end{cases}$$

式中 $i = 1,2,\ldots,n$

8.3 Hopfield神经网络

其中g(•)为双曲函数，一般取为：

$$g(x)=\rho\frac{1-e^{-\lambda x}}{1+e^{-\lambda x}}$$



式中，$\rho>0$　$\lambda>0$

Hopfield网络的动态特性要在状态空间中考虑，令

$\mathbf{u}=(u_1,u_2,\ldots,u_n)^T$为$n$个神经元的网络状态向量；

两类运放输入

$\mathrm{V}=(v_1,v_2,\ldots,v_n)^T$ 为网络的输出向量；

来自其他运放的输出

$\mathbf{I}=(I_1,I_2,\ldots,I_n)^T$为网络的外加输入。

常数电流，构成阈值

8.3 Hopfield神经网络

为了描述Hopfield网络的动态特性，定义的标准能量函数为

$$E_N = -\frac{1}{2}\sum_i\sum_j w_{ij}v_i v_j + \sum_i \frac{1}{R_i}\int_0^{v_i} g_i^{-1}(v)dv - \sum_i I_i v_i$$

若权值矩阵$\boldsymbol{W}$是对称的（ $w_{ij} = w_{ji}$ ），则

$$\frac{dE_N}{dt} = \sum_{i=1}^{n}\frac{\partial E_N}{\partial v_i}\cdot\frac{dv_i}{dt} + \sum_{i=1}^{n}\frac{\partial E_N}{\partial \omega_{ij}}\cdot\frac{d\omega_{ij}}{dt} + \sum_{i=1}^{n}\frac{\partial E_N}{\partial I_i}\cdot\frac{dI_i}{dt}$$

上式中右边后两项可以很小。

8.3 Hopfield神经网络

上式中右边后两项可以很小，原因如下：

- 首先，由于 $\frac{\partial E_N}{\partial w_{ij}} = -\sum_i \sum_j v_i v_j \qquad \frac{\partial E_N}{\partial I_i} = -\sum_i v_i$

  故 $\frac{\partial E_N}{\partial w_{ij}}$ 和 $\frac{\partial E_N}{\partial I_i}$ 与v_i，v_j有关，而v_i，v_j为双曲函数g(·)的有界输出；

- 其次，$\boldsymbol{W}$和$\boldsymbol{I}$的表达式与系统状态有关，如取u为低速激活信号，则系统状态值很小，从而 $\frac{\mathrm{d}w_{ij}}{\mathrm{d}t}$ 和 $\frac{\mathrm{d}I_i}{\mathrm{d}t}$ 很小。令

$$\sum_i^n \frac{\partial E_N}{\partial w_{ij}} \frac{dw_{ij}}{dt} + \sum_i^n \frac{\partial E_N}{\partial I_i} \frac{dI_i}{dt} = \Delta$$

$$\frac{dE_N}{dt} = -\sum_i^n \frac{\partial E_N}{\partial t} \bullet \frac{dv_i}{dt} + \Delta = -\sum_i^n \frac{dv_i}{dt} \left(\sum_j w_{ij} v_j - \frac{u_i}{R_i} + I_i \right) + \Delta = -\sum_i^n \frac{dv_i}{dt} \left(C_i \frac{du_i}{dt} \right) + \Delta$$

8.3 Hopfield神经网络

由于 $v_i = g(u_i) \rightarrow u_i = g^{-1}(v_i)$，则

$$\frac{dE_N}{dt} = -\sum_i C_i \frac{dg^{-1}(v_i)}{dv_i}\left(\frac{dv_i}{dt}\right)^2 + \Delta$$

由于C_i>0，双曲函数是单调上升函数，显然它的反函数$g^{-1}(v_i)$也为单调上升函数，即有 $\frac{dg^{-1}(v_i)}{dv_i} > 0$，取$C_i$足够大，若能保证Δ足够小，则可得到 $\frac{dE_N}{dt} \le 0$，即能量函数E_N具有负的梯度，当且仅当 $\frac{dv_i}{dt} = 0$ 时

$$\frac{dE_N}{dt} = 0 \quad (i=1,2,\ldots,n)$$

负半定，满足Lyapunov稳定性

8.3 Hopfield神经网络

由此可见，随着时间的演化，网络的解在状态空间中总是朝着能量E_N减少的方向运动。网络最终输出向量V为网络的稳定平衡点，即E_N的极小点。

Hopfield神经网络的应用主要有联想记忆和优化计算两类。其中DHNN主要用于联想记忆，CHNN主要用于优化计算。Hopfield网络在优化计算中得到成功应用，有效地解决了著名的旅行推销商问题（TSP问题），另外，它在智能控制和系统辨识中也有广泛应用。

8.3 Hopfield神经网络

- 在优化计算过程中，首先应根据优化的目标函数和约束条件构造一个能量函数，通过对能量函数的分析设计一个渐近稳定的Hopfield网络，由该网络来完成优化计算。

通常，求解最优化问题是在满足一定约束条件下求取某个目标函数的极小（极大）值，即求x，使

$$\begin{cases} f(x) = f(x_1, x_2, \cdots, xn) = min \\ g(x) \geq 0 \ (\leq 0) \qquad \text{约束条件} \end{cases}$$

如果能把优化问题中的目标函数和约束条件与Hopfield网络中的能量函数E联系起来，E的极小点也就是优化问题中满足约束条件下的目标函数的极小点，那么就可以用该Hopfield网络求解最优化问题了。

8.3 Hopfield神经网络

（二）基于Hopfield网络的路径优化

1 旅行商问题的描述

旅行商问题（Traveling Salesman Problem，简称TSP）可描述为：已知N个城市之间的相互距离，现有一推销员必须遍访这N个城市，并且每个城市只能访问一次，最后又必须返回出发城市。如何安排他对这些城市的访问次序，使其旅行路线总长度最短。

8.3 Hopfield神经网络

- 旅行商问题是一个典型的组合优化问题，其可能的路径数目与城市数目N呈指数型增长的，一般很难精确的求出其最优解。

✓ 特别是当N的数目很大时，对庞大的搜索空间中寻求最优解，用常规的方法求解计算量太大。因而寻找其有效的近似求解算法具有重要的理论意义。

✓ 另一方面，很多实际应用问题，经过简化处理后，均可化为旅行商问题，因而对旅行商问题求解方法的研究具有重要的应用价值。

8.3 Hopfield神经网络

2 求解TSP问题的Hopfield网络设计

Hopfield等采用神经网络求得经典组合优化问题（TSP）的最优解，开创了优化问题求解的新方法。

TSP问题是在一个城市集合 $\{A_c, B_c, C_c \cdots\}$ 中找出一个最短且经过每个城市各一次并回到起点的路径。为了将TSP问题映射为一个神经网络的动态过程，Hopfield采取了换位矩阵的表示方法，用N×N矩阵表示商人访问N个城市。

8.3 Hopfield神经网络

【例】有四个城市$\{A_c,B_c,C_c,D_c\}$，访问路线是：

$$D_c \to A_c \to C_c \to B_c \to D_c$$

则Hopfield网络输出所代表的有效解用下面的二维矩阵表来表示：

表8-2 四个城市的访问路线

次序 城市	1	2	3	4
A_c	0	1	0	0
B_c	0	0	0	1
C_c	0	0	1	0
D_c	1	0	0	0

8.3 Hopfield神经网络

表8-2构成了一个4x4的矩阵V，该矩阵中，各行各列只有一个元素为1，其余为0，否则是一个无效的路径。采用V_{xi}表示神经元(x,i)的输出，相应的输入用U_{xi}表示。如果城市x 在i位置上被访问，则$V_{xi}=1$，否则$V_{xi}=0$。

针对TSP问题，Hopfield定义了如下形式的能量函数：

$$E=\frac{A}{2}\sum_{x=1}^{N}\sum_{i=1}^{N}\sum_{j=1}^{N}V_{xi}V_{xj}+\frac{B}{2}\sum_{i=1}^{N}\sum_{x=1}^{N}\sum_{y=x}^{N}V_{xi}V_{yj}+\frac{C}{2}\left(\sum_{x=1}^{N}\sum_{i=1}^{N}V_{xi}-N\right)^{2}+\frac{D}{2}\sum_{x=1}^{N}\sum_{y=1}^{N}\sum_{i=1}^{N}d_{xy}V_{xi}\left(V_{y,i+1}+V_{y,i-1}\right) \tag{8.21}$$

式中，A, B, C, D是权值，d_{xy}表示城市x到城市y之间的距离。

8.3 Hopfield神经网络

式（8.21）中，E的前三项是问题的约束项，最后一项是优化目标项：

- E的第一项为保证矩阵V的每一行不多于一个1时E最小（即每个城市只去一次）；
- E的第二项保证矩阵的每一列不多于一个1时E最小（即每次只访问一城市）；
- E的第三项保证矩阵V中1的个数恰好为N时E最小。

Hopfield将能量函数概念引入神经网络，开创了求解优化问题的新方法。但该方法在求解上存在局部最小、不稳定等问题。为此，作出如下改进：

第三项仅在网络输出全为0时方起约束作用，否则前两项已经保证了第三项成立。如果前两项作这样的修改：

$$\frac{\mathrm{A}}{2}\sum_{x=1}^{N}\left(\sum_{i=1}^{N}V_{xi}-1\right)^2+\frac{\mathrm{A}}{2}\sum_{i=1}^{N}\left(\sum_{x=1}^{N}V_{xi}-1\right)^2$$

8.3 Hopfield神经网络

那么第三项完全可以省去。于是，TSP的能量函数简化为：

$$E=\frac{\mathrm{A}}{2}\sum_{x=1}^{N}\left(\sum_{i=1}^{N}V_{xi}-1\right)^{2}+\frac{\mathrm{B}}{2}\sum_{i=1}^{N}\left(\sum_{x=1}^{N}V_{xi}-1\right)^{2}+\frac{\mathrm{D}}{2}\sum_{x=1}^{N}\sum_{y=1}^{N}\sum_{i=1}^{N}V_{xi}d_{xy}V_{y,i+1}$$

由于行列的对称性取B=A，D 项由式(1)的 $V_{y,i+1}+V_{y,i-1}$ 两项简化为一项，方程式仍满足优化目标约束之要求，可得

$$E=\frac{\mathrm{A}}{2}\sum_{x=1}^{N}\left(\sum_{i=1}^{N}V_{xi}-1\right)^{2}+\frac{\mathrm{A}}{2}\sum_{i=1}^{N}\left(\sum_{x=1}^{N}V_{xi}-1\right)^{2}+\frac{\mathrm{D}}{2}\sum_{x=1}^{N}\sum_{y=1}^{N}\sum_{i=1}^{N}V_{xi}d_{xy}V_{y,i+1}$$

（8.22）

针对TSP问题的Hopfield网络设计中，**W**和**I**为固定值（与时间无关），则Δ=0，可保证$\frac{dE_N}{dt}\le 0$，实现E_N极小。

8.3 Hopfield神经网络

$$\begin{cases} C_i \dfrac{du_i}{dt} = \sum_{j=1}^{n} w_{ij} v_j - \dfrac{u_i}{R_i} + I_i \\ v_i = g(u_i) \end{cases}$$

$$E_N = -\frac{1}{2}\sum_i \sum_j w_{ij} v_i v_j + \sum_i \frac{1}{R_i}\int_0^{v_i} g_i^{-1}(v)dv - \sum_i I_i v_i$$

Hopfield网络的动态方程为：

$$\frac{\mathrm{d}U_{xi}}{\mathrm{d}t} = -\frac{\partial E_N}{\partial V_{xi}} \quad (x, i = 1, 2, \cdots, N-1)$$

$$= \sum_{i=1}^{N} w_{ij} v_j + I_i \quad (\text{取} C_i = 1, R_i \to \infty)$$

按上式求出U_{xi}，则可保证E_N逐渐减小，最终达到极小值。

按E设计能量函数E_N：

$$\frac{\mathrm{d}U_{xi}}{\mathrm{d}t} = -\frac{\partial E}{\partial V_{xi}} = -\mathrm{A}\left(\sum_{i=1}^{N} V_{xi} - 1\right) - \mathrm{A}\left(\sum_{y=1}^{N} V_{yi} - 1\right) - \mathrm{D}\sum_{y=1}^{N} d_{xy} V_{y,i+1} \qquad (8.23)$$

8.3 Hopfield神经网络

采用Hopfield网络求解TSP问题的算法描述如下：

(1) 置初值：t=0，A=1.5，D=1.0，μ=50；

```
%Step 1: 设置初始值
A=1.5;
D=1;
Mu=50;
Step=0.01;
```

(2) 计算N个城市之间的距离d_{xy} (x, y=1,2,…,N)；

```
%Step 2: %计算N各城市之间距离，计算初始路径长度
N=8;
cityfile = fopen( 'city8.txt', 'rt' );
cities = fscanf( cityfile, '%f %f',[ 2,inf] )
fclose(cityfile);
Initial_Length=Initial_RouteLength(cities);   %计算初始路径长度
DistanceCity=dist(cities',cities);
```

8.3 Hopfield神经网络

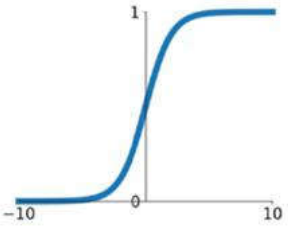
(3) 神经网络输入$U_{xi}(t)$的随机初始化在0附近产生；

并计算初始的神经元输出

$$V_{xi}(t)=\frac{1}{1+e^{-\mu U_{xi}(t)}}$$

Sigmoid

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




```
%Step 3: 神经网络输入的初始化
U=0.001*rands(N,N);
V=1./(1+exp(-Mu*U)); % S函数
```

8.3 Hopfield神经网络

(4) 利用动态方程（8.23）计算 $\frac{\mathrm{d}U_{xi}}{\mathrm{d}t}$；

$$\frac{\mathrm{d}U_{xi}}{\mathrm{d}t}=-\mathrm{A}\left(\sum_{i=1}^{N}V_{xi}-1\right)-\mathrm{A}\left(\sum_{y=1}^{N}V_{yi}-1\right)-\mathrm{D}\sum_{y=1}^{N}d_{xy}V_{y,i+1}$$

(5) 根据一阶欧拉法计算

$$U_{xi}\left(t+1\right)=U_{xi}\left(t\right)+\frac{\mathrm{d}U_{xi}}{\mathrm{d}t}\Delta T$$

(6) 为了保证收敛于正确解，即矩阵V各行各列只有一个元素为1，其余为0，采用Sigmoid函数计算

$$V_{xi}\left(t\right)=\frac{1}{1+e^{-\mu U_{xi}(t)}} \qquad (8.25)$$

其中μ>0，μ值的大小决定了Sigmoid函数的形状。

8.3 Hopfield神经网络

(7) 根据式（8.22），计算能量函数E；

(8) 检查路径的合法性，判断迭代次数是否结束，如果结束，则终止，否则返回到第（4）步；

(9) 显示输出迭代次数、最优路径、最优能量函数、路径长度的值，并作出能量函数随时间的变化的曲线图。

8.3 Hopfield神经网络


```
for k=1:1:1200   %神经网络优化
    times(k)=k;
    %Step 4: 计算du/dt
        dU=DeltaU(V,DistanceCity,A,D);
    %Step 5: 计算u(t)
        U=U+dU*Step;
    %Step 6: 计算网络输出
        V=1./(1+exp(-Mu*U)); % S函数
    %Step 7: 计算能量函数
        E=Energy(V,DistanceCity,A,D);
        Ep(k)=E;
    %Step 8: 检查路径合法性
        [V1,CheckR]=RouteCheck(V);
end
```

8.3 Hopfield神经网络


```
%Step 9:显示及作图
if(CheckR==0)
    Final_E=Energy(V1,DistanceCity,A,D);
    Final_Length=Final_RouteLength(V1,cities); %计算最终路径长度
     disp('迭代次数');k
     disp('寻优路径矩阵');V1
     disp('最优能量函数:');Final_E
     disp('初始路径:');Initial_Length
     disp('最短路径:');Final_Length

     PlotR(V1,cities);   %寻优路径作图
else
     disp(寻优路径矩阵:');V1
     disp('寻优路径矩阵无效，需要重新对神经网络输入进行初始化');
end
```

8.3 Hopfield神经网络

3 仿真实例

在TSP的Hopfield网络能量函数中，取 A=1.5，D=1.0。采样时间取 ΔT =0.01，网络输入$U_{xi}(t)$初始值选择在 [-0.001,+0.001]间的随机值，在式（8.25）的函数中，取较大的μ，以使函数比较陡峭，从而稳态时 $V_{xi}(t)$ 能够趋于1或趋于0。

以8个城市的路径优化为例，其城市路径坐标保存在当前路径的程序city8.txt中。如果初始化的寻优路径有效，即路径矩阵中各行各列只有一个元素为1，其余为0，则给出最后的优化路径，否则停止优化，需要重新运行优化程序。如果本次寻优路径有效，经过2000次迭代，最优能量函数为Final_E=1.4468，初始路程为Initial_Length=4.1419，最短路程为Final_Length =2.8937。

8.3 Hopfield神经网络

由于网络输入 $U_{xi}(t)$初始选择的随机性，可能会导致初始化的寻优路径无效，即路径矩阵中各行各列不满足“只有一个元素为1，其余为0”的条件，此时寻优失败，停止优化，需要重新运行优化程序。仿真过程表明，在100次仿真实验中，有90次以上可收敛到最优解。

仿真结果如图8-11和图8-12所示，其中图8-11为初始路径及优化后的路径的比较，图8-12为能量函数随时间的变化过程。由仿真结果可见，能量函数E单调下降，E的最小点对应问题的最优解。

8.3 Hopfield神经网络

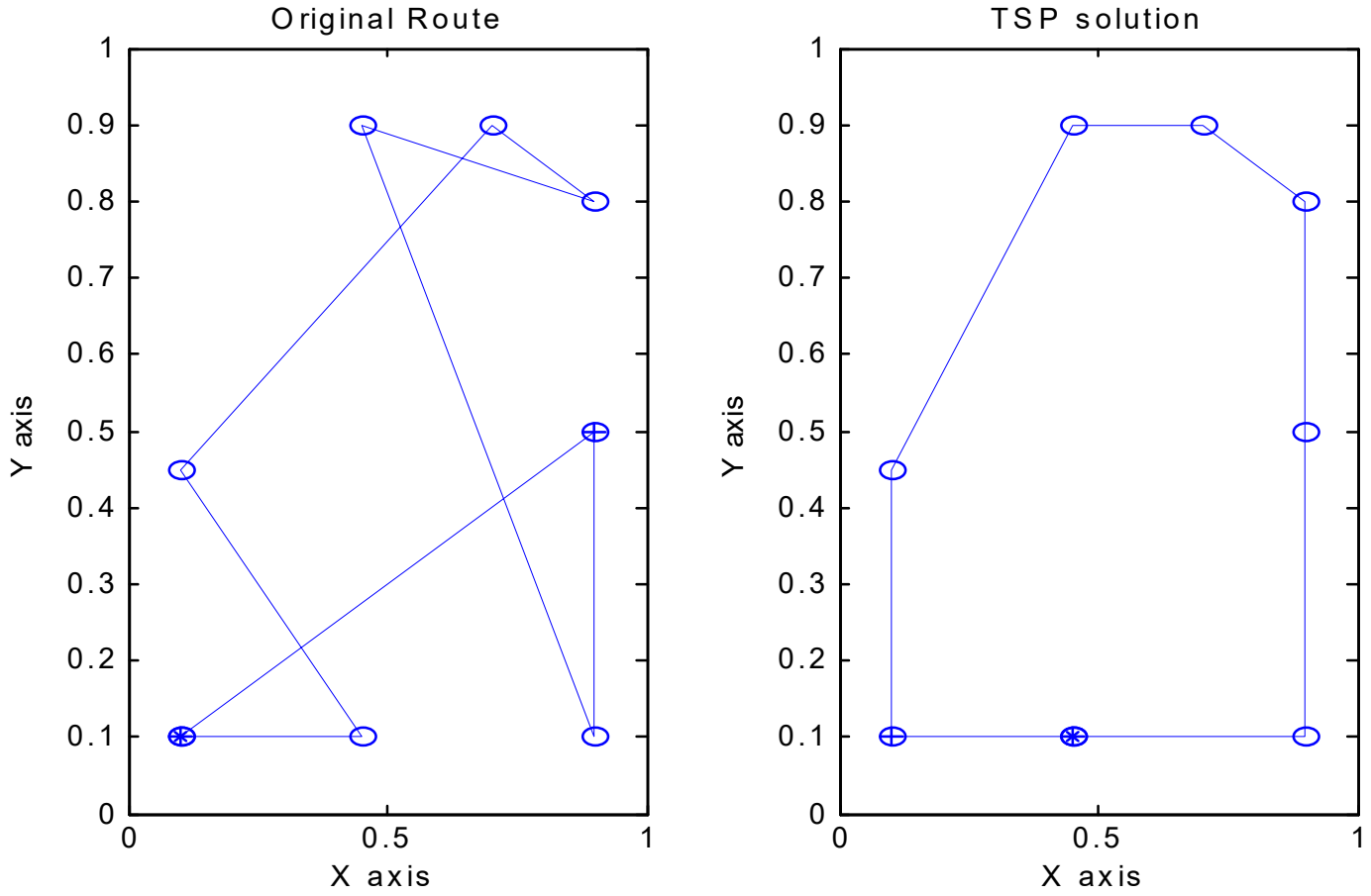


图8-11 初始路径及优化后的路径

8.3 Hopfield神经网络

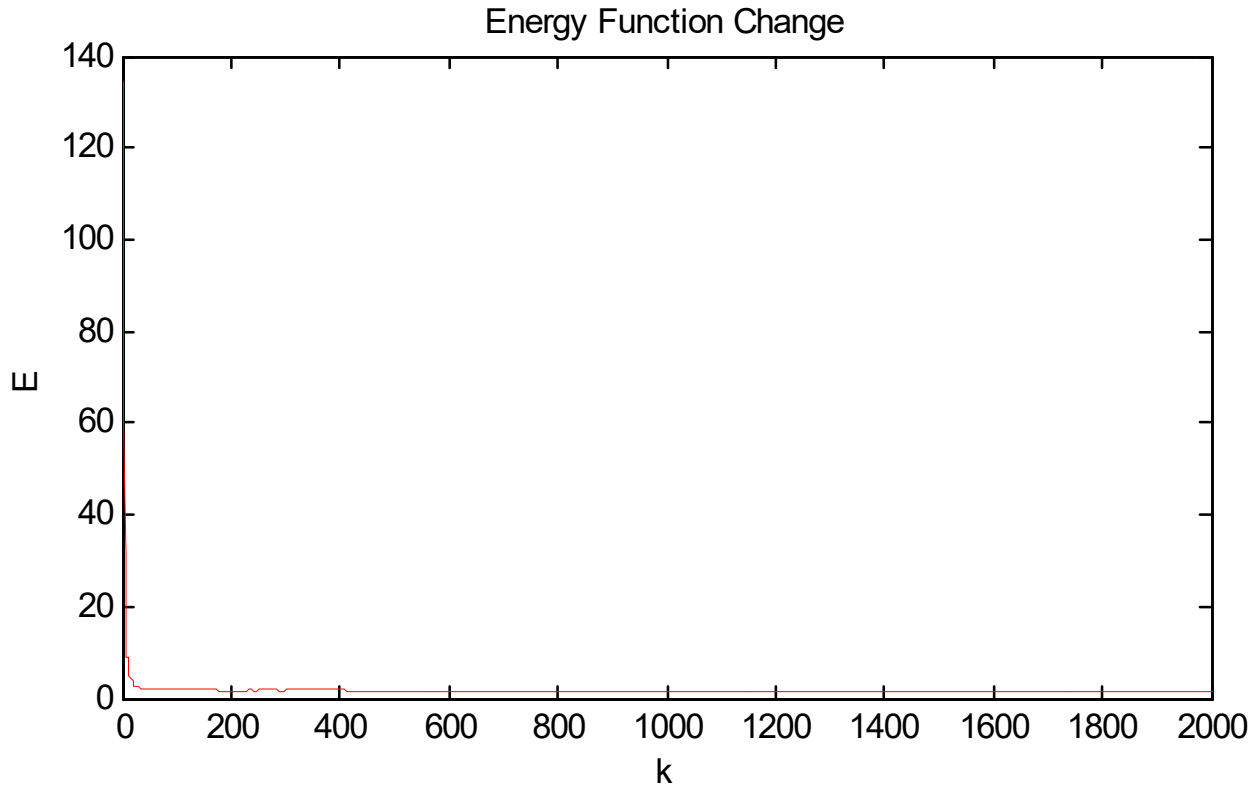


图8-12 能量函数随迭代次数的变化

8.3 Hopfield神经网络

仿真程序说明：仿真中所采用的关键命令如下：

- sumsqr(X)：求矩阵X中各元素的平方值之和；
- Sum(X)或Sum(X,1)为矩阵中各行相加，Sum(X,2)为矩阵中各列相加；
- repmat：用于矩阵复制，例如，$\mathbf{X}=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$，则 $\mathrm{repmat}(\mathbf{X},1,1)=\mathbf{X}$

$$\mathrm{repmat}(\mathbf{X},1,2)=\begin{bmatrix}1 & 2 & 1 & 2\\ 3 & 4 & 3 & 4\end{bmatrix} \qquad \mathrm{repmat}(\mathbf{X},2,1)=\begin{bmatrix}1 & 2\\ 3 & 4\\ 1 & 2\\ 3 & 4\end{bmatrix}$$

- $\mathrm{dist}(\mathbf{x},\mathbf{y})$：计算两点间的距离，例如 $\mathbf{x}=[1 \quad 1]$，$\mathbf{y}=[2 \quad 2]'$，则

$$\mathrm{dist}(\mathbf{x},\mathbf{y})=\sqrt{(2-1)^2+(2-1)^2}=\sqrt{2}$$

思考与练习

第八章的全部课后习题

7月10日前交！

命名规则：姓名_学号_第八章作业

ic_sysu@163.com