第5章

Hopfield神经网络

CHAPTER 5

1982年，美国加州理工学院的生物物理学家Hopfield教授提出了一种模拟人脑联想记忆功能的单层反馈神经网络，后来人们将这种反馈网络称为Hopfield网络。1984年，Hopfield和Tank用模拟电子线路实现了Hopfield网络，并用它成功地求解了旅行商问题。

5.1　Hopfield 神经网络概述

Hopfield 神经网络(Hopfield Neural Network)是一种具有反馈连接结构的递归神经网络,输入信息经过 Hopfield 神经网络处理后,计算得到的输出会反馈到输入端,从而对下一时刻的系统状态产生影响,在计算过程中 Hopfield 神经网络的反馈过程会一直反复进行,直至停止输入。Hopfield 神经网络是由单层神经元互相完全连接构成的反馈神经网络,其结构如图 5-1 所示,网络的全连接结构使得其在进行计算时,每个神经元既是输入也是输出,即每个神经元接收其他所有神经元传递的信息。网络的反馈结构使得神经网络每一时刻的计算都与神经网络前一时刻的计算相关。

Hopfield 神经网络循环递归的网络迭代过程使得网络最终可以达到稳定状态,因此可以引入能量函数的概念,采用能量函数的形式描述网络收敛的过程,即用能量函数达到极小值的计算过程描述网络收敛到稳态的过程。能量函数的应用为神经网络运行的状态是否达到稳态提供了可靠的判断依据。递归神经网络的网络状态与时序相关,随着网络的迭代最终能达到稳定状态,这是与前向网络比较明显的区别之一,但递归神经网络收敛结果与网络参数的选取有很强的相关性,合理的网络参数是递归神经网络正常工作的基础。

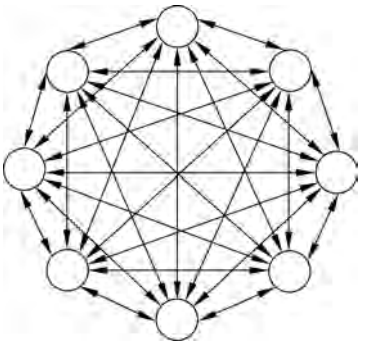

图 5-1　Hopfield 神经网络结构图

Hopfield 神经网络模型有离散型和连续型两种,离散型适用于联想记忆,连续型适合处理优化问题。

5.2　离散型 Hopfield 神经网络

Hopfield 神经网络作为一种全连接型的神经网络,曾经为神经网络的发展开辟了新的研究途径。它利用与层次型神经网络不同的结构特征和学习方法,模拟生物神经网络的记忆机理,获得了令人满意的结果。离散型 Hopfield 神经网络(Discrete Hopfield Neural Network, DHNN)是 Hopfield 最早提出的单层反馈网络,其网络结构中神经元的输出只有“1”和“−1”两种取值,分别表示神经元激活和抑制状态。

5.2.1　离散型 Hopfield 神经网络结构及工作方式

离散型 Hopfield 神经网络的网络结构如图 5-2 所示。

由图 5-2 可以看出离散型 Hopfield 神经网络中的每个神经元 $x_j(j=1,2,\cdots,n)$都是彼此连接的,神经元的阈值为 $\theta_j(j=1,2,\cdots,n)$,连接网络中的交叉点表示的是连接的权值用 $w_{ij}(i,j=1,2,\cdots,n)$表示。离散型 Hopfield 神经网络中的神经元都是二值输出神经元,即神经元根据权值对输入进行累加求和,经过非线性映射将输入映射为输出“1”或“−1”。

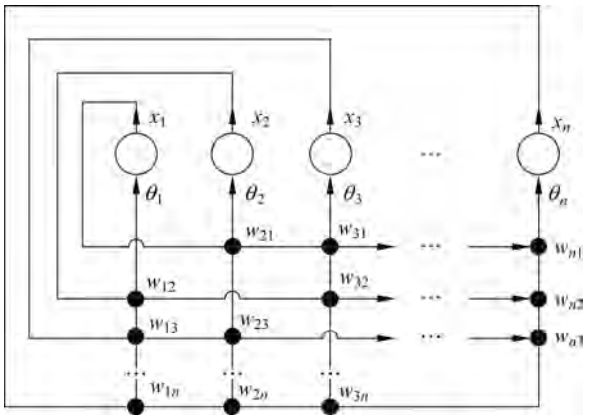

图 5-2　离散型 Hopfield 神经网络的网络结构

离散型 Hopfield 神经网络的单个神经元结构如图 5-3 所示。

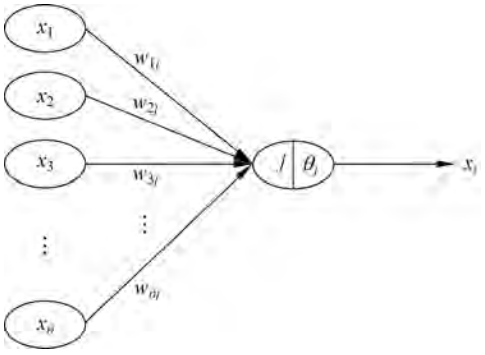

图 5-3　离散型 Hopfield 神经网络的单个神经元结构

离散型 Hopfield 神经网络中的神经元采用经典感知机模型进行信息计算且每个神经元的功能都相同，其输出值称为状态，用 $x_j(j=1,2,\cdots,n)$ 表示，w_{ij} 表示神经元 i 与神经元 j 之间的连接权值，神经元的非线性映射由激活函数 $f(\cdot)$ 表示，每个神经元均有一个阈值 $\theta_j(j=1,2,\cdots,n)$。n 个神经元的状态集合构成的离散型 Hopfield 神经网络的状态，如式(5-1)所示。

$$\boldsymbol{X}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}} \tag{5-1}$$

网络的输入即网络的初始状态如式(5-2)所示。

$$\boldsymbol{X}(0)=[x_1(0),x_2(0),\cdots,x_n(0)]^{\mathrm{T}} \tag{5-2}$$

神经元 j 的净输入 s_j 与输出 x_j 分别如式(5-3)和式(5-4)所示。

$$s_j=\sum_{i=1}^{n}x_i w_{ij}-\theta_j \tag{5-3}$$

$$x_j=f(s_j)=\mathrm{sgn}(s_j)=\begin{cases}1, & s_j>0\\ -1, & s_j\leqslant 0\end{cases} \tag{5-4}$$

当网络中各个神经元的状态改变时，网络的状态也会随之改变，网络稳定的标志是网络中每个神经元的状态都不再改变，此时的稳定状态就是该网络的输出。

Hopfield 网络的工作方式主要有两种形式：

串行(异步)工作方式：在任一时刻 t，只有某一神经元(随机的或确定的)的状态发生变化，而其他神经元的状态不变。

并行(同步)工作方式：在任一时刻 t，部分神经元或全部神经元的状态同时改变。

1) 串行(异步)工作方式

离散型 Hopfield 神经网络串行工作时，神经网络按照预先设定的顺序或随机选定的方式每次激活一个神经元进行状态调整，而其他神经元保持不变，神经元状态调整计算过程如式(5-5)所示。

$$x_j(t+1)=\begin{cases}\operatorname{sgn}(s_j(t)), & j=i\\ s_j(t), & j\neq i\end{cases},\quad i,j=1,2,\cdots,n \tag{5-5}$$

2) 并行(同步)工作方式

离散型 Hopfield 神经网络并行工作时，神经网络中的神经元同时进行状态调整，神经元状态调整计算过程如式(5-6)所示。

$$x_j(t+1)=\operatorname{sgn}(s_j(t)),\quad j=1,2,\cdots,n \tag{5-6}$$

经过迭代更新，一个离散型 Hopfield 神经网络就会逐渐达到稳定状态，判断稳定的依据如式(5-7)所示。

$$x_j(t+1)=x_j(t)=f(s_j(t)),\quad j=1,2,\cdots,n \tag{5-7}$$

从初始状态开始，网络沿能量递减的方向进行不断演化并最终趋于稳定。离散型 Hopfield 神经网络的运行流程如下：

(1) 初始化网络状态。

(2) 从网络中随机选取一个神经元 i。

(3) 按照式(5-3)计算神经元 i 在 t 时刻的净输入 $s_i(t)$。

(4) 按照式(5-4)计算神经元 i 在 $t+1$ 时刻的输出 $x_i(t+1)$，除 i 之外的其他神经元状态保持不变。

(5) 使用式(5-7)判断网络是否达到稳定状态，若未达到则跳转到(2)继续向下执行；若已达到稳定状态则停止网络运行。

离散型 Hopfield 神经网络的运行流程如例 5-1 所示。

【例 5-1】 一个三个结点的离散型 Hopfield 神经网络的初始状态为 $\boldsymbol{X}(0)=(-1,-1,-1)$，权值矩阵 $\boldsymbol{W}$ 和阈值 $\boldsymbol{\theta}$ 如下所示，求解该网络的稳定状态。

$$\boldsymbol{W}=\begin{pmatrix}0 & -0.5 & 0.5\\ -0.5 & 0 & 0.6\\ 0.5 & 0.6 & 0\end{pmatrix}\quad \boldsymbol{\theta}=\begin{pmatrix}-0.1\\ 0\\ 0\end{pmatrix}$$

解：

$t=0$：神经网络初始状态为 $\boldsymbol{X}(0)=(-1,-1,-1)$。

$t=1$：选取结点 1，则结点 1 的状态变化如下。

$x_1(1)=\text{sgn}[-1\times0+(-1)\times(-0.5)+(-1)\times0.5-(-0.1)]=\text{sgn}(0.1)=1$,此时网络状态为 $\boldsymbol{X}(1)=(1,-1,-1)$。

$t=2$：选取结点 2,则结点 2 的状态变化如下。

$x_2(2)=\text{sgn}[1\times(-0.5)+(-1)\times0+(-1)\times0.6-0]=\text{sgn}(-1.1)=-1$,此时网络状态为 $\boldsymbol{X}(2)=(1,-1,-1)$。

$t=3$：选取结点 3,则结点 3 的状态变化如下。

$x_3(3)=\text{sgn}[1\times0.5+(-1)\times0.6+(-1)\times0-0]=\text{sgn}(-0.1)=-1$,此时网络状态为 $\boldsymbol{X}(3)=(1,-1,-1)$。

$t=4$：选取结点 1,则结点 1 的状态变化如下。

$x_1(4)=\text{sgn}[1\times0+(-1)\times(-0.5)+(-1)\times0.5-(-0.1)]=\text{sgn}(0.1)=1$,此时网络状态为 $\boldsymbol{X}(4)=(1,-1,-1)$。

$t=5$：选取结点 2,则结点 2 的状态变化如下。

$x_2(5)=\text{sgn}[1\times(-0.5)+(-1)\times0+(-1)\times0.6-0]=\text{sgn}(-1.1)=-1$,此时网络状态为 $\boldsymbol{X}(5)=(1,-1,-1)$。

$t=6$：选取结点 3,则结点 3 的状态变化如下。

$x_2(6)=\text{sgn}[1\times0.5+(-1)\times0.6+(-1)\times0-0]=\text{sgn}(-0.1)=-1$,此时网络状态为 $\boldsymbol{X}(6)=(1,-1,-1)$。

至此各神经元输出已经不再改变,故判定网络已经进入稳定状态(1,−1,−1)。

5.2.2　离散型 Hopfield 神经网络的吸引子与能量函数

Hopfield 神经网络有一个显著特点,就是加入了“能量函数”的概念,同时将神经网络与动力学之间的关系进行了阐述。网络的吸引子(attractor or fixed-point)是指网络达到稳定时的状态 $\boldsymbol{X}$。吸引子能够决定一个动力学系统的最终行为,它同时能够作为信息的分布存储记忆和神经优化计算的基础。若把需要记忆的样本信息存储于不同的吸引子,当输入含有部分信息的样本时,网络的演变过程就是从部分信息中还原全部信息的过程,即实现联想记忆。

如果网络的状态 $\boldsymbol{X}$ 满足 $\boldsymbol{X}=f(\boldsymbol{WX}-\boldsymbol{T})$,则称 $\boldsymbol{X}$ 为网络的吸引子。在一个 Hopfield 网络中,通常有多个吸引子,每个吸引子为一个能量的局部最优点。

离散型 Hopfield 神经网络实质上是一个离散的非线性动力学系统,网络从初态 $\boldsymbol{X}(0)$ 经过有限次的迭代达到 $\boldsymbol{X}(t)=\boldsymbol{X}(t+1)$状态,则表示网络已处于稳定状态,这种稳定状态也称为吸引子。反馈网络的相图通常如图 5-4 所示。

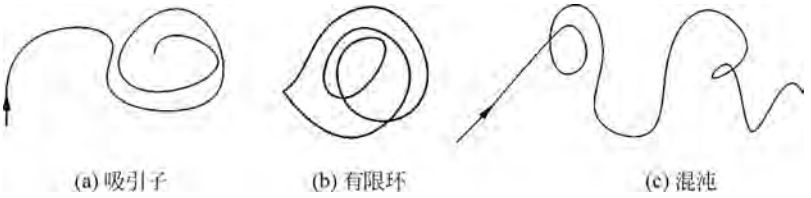

图 5-4　反馈网络的 3 种相图

如图 5-4(a)所示，网络逐渐迭代到稳定状态。如图 5-4(b)所示，网络处于不稳定状态，但由于离散型 Hopfield 神经网络状态只有 1 和－1 两种情况，因此网络只会出现在限幅内的自持振荡，通常也将此种网络称为有限环网络。如图 5-4(c)所示，网络状态的轨迹在某个确定的范围内变迁，既不重复也不停止，状态运动轨迹不会发散到无穷远，这种现象称为混沌。由于离散型 Hopfield 神经网络状态是有限的，因此不存在混沌现象。

(1) 对于离散型 Hopfield 神经网络，若按照异步方式调整网络状态，对于任意初态离散 Hopfield 神经网络，网络都最终收敛到一个吸引子。能量函数如式(5-8)所示。

$$\begin{aligned}\boldsymbol{E}(t)&=-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}x_ix_j+\sum_{i=1}^{n}\theta_ix_i\\&=-\frac{1}{2}\boldsymbol{X}^{\mathrm{T}}(t)\boldsymbol{W}\boldsymbol{X}(t)+\boldsymbol{X}^{\mathrm{T}}(t)\boldsymbol{T}\end{aligned}\tag{5-8}$$

设网络能量函数的改变量为 $\Delta\boldsymbol{E}$，网络状态的改变量为 $\Delta\boldsymbol{X}$，如式(5-9)所示。

$$\begin{aligned}\Delta\boldsymbol{E}(t)&=\boldsymbol{E}(t+1)-\boldsymbol{E}(t)\\\Delta\boldsymbol{X}(t)&=\boldsymbol{X}(t+1)-\boldsymbol{X}(t)\end{aligned}\tag{5-9}$$

因此：

$$\begin{aligned}\Delta\boldsymbol{E}(t)&=\boldsymbol{E}(t+1)-\boldsymbol{E}(t)\\&=-\frac{1}{2}[\boldsymbol{X}(t)+\Delta\boldsymbol{X}(t)]^{\mathrm{T}}\boldsymbol{W}[\boldsymbol{X}(t)+\Delta\boldsymbol{X}(t)]+[\boldsymbol{X}(t)+\Delta\boldsymbol{X}(t)]^{\mathrm{T}}\boldsymbol{T}-\\&\quad\left[-\frac{1}{2}\boldsymbol{X}^{\mathrm{T}}(t)\boldsymbol{W}\boldsymbol{X}(t)+\boldsymbol{X}^{\mathrm{T}}(t)\boldsymbol{T}\right]\\&=-\Delta\boldsymbol{X}^{\mathrm{T}}(t)\boldsymbol{W}\boldsymbol{X}(t)-\frac{1}{2}\Delta\boldsymbol{X}^{\mathrm{T}}(t)\boldsymbol{W}\Delta\boldsymbol{X}(t)+\Delta\boldsymbol{X}^{\mathrm{T}}(t)\boldsymbol{T}\\&=-\Delta\boldsymbol{X}^{\mathrm{T}}(t)[\boldsymbol{W}\boldsymbol{X}(t)-\boldsymbol{T}]-\frac{1}{2}\Delta\boldsymbol{X}^{\mathrm{T}}(t)\boldsymbol{W}\Delta\boldsymbol{X}(t)\end{aligned}\tag{5-10}$$

由于网络状态异步更新，即第 t 个时刻只有一个神经元 j 调整状态，将 $\Delta\boldsymbol{X}(t)=[0,0,\cdots,0,\Delta\boldsymbol{x}_j(t),0,\cdots,0]^{\mathrm{T}}$ 代入式(5-10)，可得式(5-11)。

$$\Delta\boldsymbol{E}(t)=-\Delta\boldsymbol{x}_j(t)\left[\sum_{i=1}^{n}(\boldsymbol{w}_{ij}-\boldsymbol{T}_j)\right]-\frac{1}{2}\Delta\boldsymbol{x}_j^2(t)\boldsymbol{w}_{jj}\tag{5-11}$$

对于离散型 Hopfield 神经网络来说，$\boldsymbol{W}$ 为对称矩阵，神经元之间不存在自反馈，即 $\boldsymbol{W}_{jj}=\boldsymbol{0}$，因此式(5-11)可以简化为式(5-12)。

$$\Delta\boldsymbol{E}(t)=-\Delta\boldsymbol{x}_j(t)\boldsymbol{s}_j(t)\tag{5-12}$$

由于离散型 Hopfield 神经网络状态包括 1 和－1 两种，因此式(5-12)可能会出现以下情况：

情况一：$x_j(t)=-1$，$x_j(t+1)=1$，则由状态变化公式(5-9)得 $\Delta x_j(t)=2$，由 $s_j(t)\geqslant 0$，由能量变化量式(5-12)得 $\Delta E(t)\leqslant 0$。

情况二：$x_j(t)=1$，$x_j(t+1)=-1$，故 $\Delta x_j(t)=-2$，$s_j(t)<0$，得 $\Delta E(t)<0$。

情况三：$x_j(t)=x_j(t+1)$，故 $\Delta x_j(t)=0$，所以有 $\Delta E(t)=0$。

综合三种情况可得 $\Delta E(t)\leqslant 0$，即网络的演变过程中能量一直处于非递增状态，又由于离散型 Hopfield 神经网络中结点状态只有 1 或－1 两种情况，能量函数 $E(t)$ 拥有下界，因此网络会收敛到 $\Delta E(t)=0$。

(2) 对于离散型 Hopfield 神经网络,若按照同步方式调整网络状态,且连接权矩阵 $\boldsymbol{W}$ 为非负定对称阵,则对于任意初态,网络都最终收敛到一个吸引子。如式(5-13)所示。

$$\begin{aligned}\Delta \boldsymbol{E}(t) &= \boldsymbol{E}(t+1) - \boldsymbol{E}(t) \\ &= \Delta \boldsymbol{X}^{\mathrm{T}}(t)[\boldsymbol{W}\boldsymbol{X}(t) - \boldsymbol{T}] - \frac{1}{2}\Delta \boldsymbol{X}^{\mathrm{T}}(t)\boldsymbol{W}\Delta \boldsymbol{X}(t) \\ &= -\Delta \boldsymbol{X}^{\mathrm{T}}(t)\boldsymbol{s}_j(t) - \frac{1}{2}\Delta \boldsymbol{X}^{\mathrm{T}}(t)\boldsymbol{W}\Delta \boldsymbol{X}(t) \\ &= -\sum_{j=1}^{n}\Delta \boldsymbol{x}_j(t)\boldsymbol{s}_j(t) - \frac{1}{2}\Delta \boldsymbol{X}^{\mathrm{T}}(t)\boldsymbol{W}\Delta \boldsymbol{X}(t) \end{aligned} \tag{5-13}$$

由于 $-\Delta \boldsymbol{x}_j(t)\boldsymbol{s}_j(t) \leqslant 0$,$\boldsymbol{W}$ 为非负定对称阵,因此 $\Delta \boldsymbol{E}(t) \leqslant 0$,即网络最终会收敛到常数,对应的稳定状态称为网络的一个吸引子。

通过上述分析可以看出,如果连接权值矩阵 $\boldsymbol{W}$ 设计不能满足非负定对称阵的要求,网络就会发生振荡。异步工作方式相比同步工作方式更好的稳定性,但没有并行处理能力。

1. 吸引子的性质

性质 1:若 X 是一个网络的吸引子,且阈值为 0,在 sgn(0)处有 $x_j(t+1) = x_j(t)$,则 $-X$ 也是该网络的一个吸引子。

证明:因为 X 是吸引子,则 $X = f(WX)$,从而有式(5-14)。

$$f[W(-X)] = f[-WX] = -f[WX] = -X \tag{5-14}$$

性质 2:若 X^a 是网络的一个吸引子,$W_{ii} = 0$,且 sgn(0) = 1,则与 X^a 的海明距离 $d_{H(X^a, X^b)=1}$ 的 X^b 一定不是吸引子。

证明:两个向量的海明距离 $d_H(\boldsymbol{X}^a, \boldsymbol{X}^b)$ 是指两个向量中不同元素的个数。设 $\boldsymbol{X}_1^a \neq \boldsymbol{X}_1^b$,$\boldsymbol{X}_j^a = \boldsymbol{X}_j^b$;$j = 1, 2, 3, \cdots, n$。因为 $W_{11=0}$,由吸引子的定义可得式(5-15)。

$$X_1^a = f\left(\sum_{i=2}^{n} W_{ii}X_i^a - T_1\right) = f\left(\sum_{i=2}^{n} W_{ii}X_i^b - T_1\right) \tag{5-15}$$

由于 $X_1^a \neq X_1^b$,因此可得式(5-16)。

$$X_1^b \neq f\left(\sum_{i=2}^{n} W_{ii}X_i^b - T_1\right) \tag{5-16}$$

故可得 X^b 不是该网络的吸引子。

2. 吸引域的基本概念

大量的样本存在缺损或包含噪声的情况,当离散型 Hopfield 神经网络想要实现正确的联想记忆,就要求当样本状态脱离吸引子一定范围后,应该有能力调整回原状态,吸引子的这种吸引范围称为吸引域。

定义 5-1　设 $\boldsymbol{X}^a$ 为吸引子,若从 $\boldsymbol{X}$ 到 $\boldsymbol{X}^a$ 存在一条路径可达,则称 $\boldsymbol{X}$ 弱吸引到 $\boldsymbol{X}^a$,若对集合 $\boldsymbol{X} = \{\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_n\}$ 均有 $\boldsymbol{X}$ 弱吸引到 $\boldsymbol{X}^a$,则称集合 $\boldsymbol{X}$ 为 $\boldsymbol{X}^a$ 的弱吸引域;若从 $\boldsymbol{X}$ 到 $\boldsymbol{X}^a$ 迭代中每条路径均可达,则称 $\boldsymbol{X}$ 强吸引到 $\boldsymbol{X}^a$,若对集合 $\boldsymbol{X} = \{\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_n\}$ 均有 $\boldsymbol{X}$ 强吸引到 $\boldsymbol{X}^a$,则称集合 $\boldsymbol{X}$ 为 $\boldsymbol{X}^a$ 的强吸引域。

【例 5-2】 在离散型 Hopfield 神经网络中，$n=4$；$T_j=0$；$j=1,2,3,4$；向量 $\boldsymbol{X}^a$、$\boldsymbol{X}^b$ 和权值矩阵 $\boldsymbol{W}$ 分别为式(5-17)所示。

$$\boldsymbol{X}^a=\begin{bmatrix}1\\1\\1\\1\end{bmatrix},\quad \boldsymbol{X}^b=\begin{bmatrix}-1\\-1\\-1\\-1\end{bmatrix},\quad \boldsymbol{W}=\begin{bmatrix}0&2&2&2\\2&0&2&2\\2&2&0&2\\2&2&2&0\end{bmatrix} \tag{5-17}$$

尝试探讨 $\boldsymbol{X}^a$，$\boldsymbol{X}^b$ 是否为该网络的吸引子，并验证其是否具有联想记忆能力。

解：由吸引子定义式(5-14)可得式(5-18)。

$$f(\boldsymbol{W}\boldsymbol{X}^a)=f\begin{bmatrix}6\\6\\6\\6\end{bmatrix}=\begin{bmatrix}\mathrm{sgn}(6)\\\mathrm{sgn}(6)\\\mathrm{sgn}(6)\\\mathrm{sgn}(6)\end{bmatrix}=\begin{bmatrix}1\\1\\1\\1\end{bmatrix}=\boldsymbol{X}^a \tag{5-18}$$

因此，$\boldsymbol{X}^a$ 是网络的吸引子，根据吸引子性质 $\boldsymbol{X}^b=-\boldsymbol{X}^a$，所以，$\boldsymbol{X}^b$ 也是网络的吸引子。

验证离散型 Hopfield 神经网络的联想记忆能力：

设有样本 $\boldsymbol{X}^1=\begin{bmatrix}-1\\1\\1\\1\end{bmatrix}$，$\boldsymbol{X}^2=\begin{bmatrix}1\\-1\\-1\\-1\end{bmatrix}$，$\boldsymbol{X}^3=\begin{bmatrix}1\\1\\-1\\-1\end{bmatrix}$，当离散型 Hopfield 神经网络以异步方式工作时 $\boldsymbol{X}^a$、$\boldsymbol{X}^b$ 两个吸引子对三个样本的吸引能力。

设离散型 Hopfield 神经网络初态 $\boldsymbol{X}(0)=\boldsymbol{X}^1=\begin{bmatrix}-1\\1\\1\\1\end{bmatrix}$，神经元的调整顺序为[1,2,3,4]，则有 $\boldsymbol{X}(1)=\boldsymbol{X}^a=\begin{bmatrix}1\\1\\1\\1\end{bmatrix}$，可以得到异步方式只经过一步迭代，样本 $\boldsymbol{X}^1$ 即收敛于 $\boldsymbol{X}^a$。

同样，设离散型 Hopfield 神经网络初态 $\boldsymbol{X}(0)=\boldsymbol{X}^2=\begin{bmatrix}1\\-1\\-1\\-1\end{bmatrix}$，神经元的调整顺序为[1,2,3,4]，则有 $\boldsymbol{X}(1)=\boldsymbol{X}^b=\begin{bmatrix}-1\\-1\\-1\\-1\end{bmatrix}$，可以得到异步方式只经过一步迭代，样本 $\boldsymbol{X}^2$ 即收敛于 $\boldsymbol{X}^b$。

现假设离散型 Hopfield 神经网络初态 $\boldsymbol{X}(0)=\boldsymbol{X}^3=\begin{bmatrix}1\\1\\-1\\-1\end{bmatrix}$，神经元的调整顺序为

[1,2,3,4],则有 $\boldsymbol{X}(1)=\begin{bmatrix}-1\\1\\-1\\-1\end{bmatrix}$，$X(2)=\begin{bmatrix}-1\\-1\\-1\\-1\end{bmatrix}=\boldsymbol{X}^b$,可以得到异步方式只经过两步迭代，样本 $\boldsymbol{X}^3$ 即收敛于 $\boldsymbol{X}^b$。现修改神经元调整顺序为[3,4,1,2],则有 $\boldsymbol{X}(1)=\begin{bmatrix}1\\1\\1\\-1\end{bmatrix}$,$\boldsymbol{X}(2)=\begin{bmatrix}1\\1\\1\\1\end{bmatrix}=\boldsymbol{X}^a$。

通过上述实例分析可以看出当离散型 Hopfield 神经网络异步更新时,异步调整顺序确定稳定的吸引子与初态有关；初态确定稳定的吸引子与异步调整顺序有关。

5.2.3　离散型 Hopfield 神经网络的连接权值设计

5.2.2 节介绍了离散型 Hopfield 神经网络的能量函数。如果能够将记忆的样本信息存储在不同的能量极值点上,然后给网络输入某一状态,网络“学习”后达到稳定状态,就可以发挥联想记忆功能,联想出之前记忆的样本。这便是离散型 Hopfield 神经网络的一个重要功能——联想记忆功能。要实现联想记忆,离散型 Hopfield 神经网络必须具有两个基本条件：①网络能收敛到稳定的平衡状态,并以其作为样本的记忆信息；②具有回忆能力,能够从某一残缺的信息回忆起所属的完整的记忆信息。

离散型 Hopfield 神经网络实现联想记忆过程分为以下两个阶段。

学习记忆阶段：其也称为离散型 Hopfield 神经网络的学习过程。指研究者通过设计一种方法确定一组合适的权值,使离散型 Hopfield 神经网络记忆达到稳定状态。

联想回忆阶段：联想过程就是给定输入模式,联想记忆网络通过动力学的演化过程达到稳定状态,即收敛到吸引子,回忆起已存储模式的过程。

网络的记忆样本,即能量函数的最小值的分布,由网络的连接权值和阈值决定。所以,离散型 Hopfield 神经网络实现联想记忆的重点就是根据能量极值点设计一组适合的网络连接权值和阈值。

吸引子的分布是由网络的权值(包括阈值)决定的,设计吸引子的核心就是如何设计一组合适的权值。为了使所设计的权值满足要求,权值矩阵应符合以下要求：

① 为保证异步方式工作时网络收敛,$\boldsymbol{W}$ 应为对称阵；

② 为保证同步方式工作时网络收敛,$\boldsymbol{W}$ 应为非负定对称阵；

③ 保证给定的样本是网络的吸引子,并且要有一定的吸引域。

具体设计时,可以采用以下不同的方法。

(1) 比较常用的离散型 Hopfield 神经网络连接权值设计方法是基于 Hebb 学习规则的外积和法。设共有 K 个记忆样本 $\boldsymbol{X}_k$,$k=1,2,\cdots,K$,$\boldsymbol{X}\in\{-1,1\}^n$,并设样本两两正交,且 $n>K$,则连接权值矩阵 $\boldsymbol{W}$ 为记忆样本的外积和如式(5-19)所示。

$$W=\alpha\sum_{k=1}^{K}\boldsymbol{X}^k(\boldsymbol{X}^k)^{\mathrm{T}} \tag{5-19}$$

又知 $w_{ii}=0$，所以式(5-19)可改写为式(5-20)。

$$W=\alpha\sum_{k=1}^{K}[\boldsymbol{X}^k(\boldsymbol{X}^k)^{\mathrm{T}}-I] \tag{5-20}$$

其中，$\boldsymbol{I}$ 为单位矩阵；α 为常数，且 $\alpha>0$，α 一般取 1 或 $\frac{1}{n}$。

(2) 除基于 Hebb 学习规则的外积和法之外，还有更简单的联立方程法、δ 学习规则方法、伪逆法、正交化法等，但是一般更通用的还是外积和法。

δ 学习规则方法基本公式是式(5-21)和式(5-22)。

$$\Delta W=\eta\cdot\delta\cdot P \tag{5-21}$$

$$w_{ij}(t+1)=w_{ij}(t)+\eta[T(t)-A(t)]P(t) \tag{5-22}$$

即通过计算该神经元结点的实际激活值 $A(t)$，与期望状态 $T(t)$ 进行比较，若不满足要求，将两者的误差的一部分作为调整量，若满足要求，则相应的权值保持不变。

伪逆法具体方法为：

设输入样本 $\boldsymbol{X}=[\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_N]$，输入输出之间用权重 $\boldsymbol{W}$ 来映射，则如式(5-23)所示。

$$\boldsymbol{Y}=\mathrm{sgn}(\boldsymbol{W}\boldsymbol{X}) \tag{5-23}$$

由此可得式(5-24)。

$$\boldsymbol{W}=\boldsymbol{N}^*\boldsymbol{P}^* \tag{5-24}$$

其中，$\boldsymbol{P}^*$ 为伪逆，有 $\boldsymbol{P}^*=(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{P})^{-1}\boldsymbol{P}^{\mathrm{T}}$，如果输入样本之间是线性无关的，则 $\boldsymbol{P}^{\mathrm{T}}\boldsymbol{P}$ 满秩，其逆存在，则可求出连接权矩阵 $\boldsymbol{W}$。

在网络连接矩阵确定后，网络就可以进行工作，能够接收输入模式向量，实现其联想记忆功能。

但是实际上，离散型 Hopfield 神经网络的联想记忆功能是与其记忆容量和样本差异有关。当记忆样本数量少并且相互之间区别大时，网络的记忆效果较准确；而当记忆样本过多且相互之间较为相似时，容易引起混淆，网络最终达到的稳定状态就不一定会是之前记忆的样本。这两项中有一项不满足要求，最后结果的出错率就会大大提高。

5.2.4　离散型 Hopfield 神经网络的信息存储容量

信息存储是指将一组向量存储在网络中的过程，存储过程主要是调整神经元之间的连接权重，因此可以看作是一种学习的过程。

信息存储容量是神经网络处理能力的重要衡量标准之一。当网络规模一定时，神经网络所能记忆的样本数量是有限的，不同的神经网络拥有不同的存储能力，神经网络的存储容量与神经网络的规模、连接权值的设计算法以及记忆模式向量的分布都有关。离散型 Hopfield 神经网络的联想记忆能力是通过将一些样本存储在不同的能量极值上实现的，联想记忆能力与离散型 Hopfield 的信息存储容量密切相关。

以下列出一些关于离散型 Hopfield 神经网络的信息存储容量的定义和定理。

定义 5-2　网络能够存储的最大样本模式数称为网络的信息存储容量。

定理 5-1　若具有 n 个神经元的离散型 Hopfield 神经网络的连接权矩阵的主对角元素的值为 0,则该离散型 Hopfield 神经网络的最大信息存储容量为 n。

定理 5-2　若有 m 个两两正交的记忆模式 $\boldsymbol{X}^k=(x_1^k,x_2^k,\cdots,x_n^k)(k=1,2,\cdots,m)$,$x_i\in\{-1,1\}(i=1,2,\cdots,m)$,其中 $n>m$,并且网络连接权矩阵采用外积和法,由式(5-20)计算得到,则这 m 个记忆模式都是具有 n 个神经元的离散型 Hopfield 神经网络的稳定状态。

定理 5-3　若有 m 个两两正交的记忆模式 $\boldsymbol{X}^k=(x_1^k,x_2^k,\cdots,x_n^k)(k=1,2,\cdots,m)$,$x_i\in\{-1,1\}(i=1,2,\cdots,m)$,其中 $n\geqslant m$,并且网络连接权矩阵采用外积和法,由式(5-19)计算得到,则这 m 个记忆模式都是具有 n 个神经元的离散型 Hopfield 神经网络的稳定状态。

从以上定理可知,当用外积和法设计离散型 Hopfield 神经网络时,如果记忆模式都满足两两正交的条件,则规模为 n 的离散型 Hopfield 神经网络最多可记忆 n 个样本模式,但是实际情况下,模式样本不可能都满足两两正交的条件,对于非正交模式,离散型 Hopfield 神经网络的信息存储能力会大大降低,其信息存储容量一般为 $0.13n\sim0.15n$。

当离散型 Hopfield 神经网络只记忆一个稳定模式时,该模式肯定被其准确无误地记住,但当需要记忆的模式增加时,记忆模式相互影响,会造成连接权值的改变,可能会出现“权值移动”和“交叉干扰”两种情况。

在离散型 Hopfield 神经网络的学习过程中,$\boldsymbol{X}^k=(x_1^k,x_2^k,\cdots,x_n^k)^{\mathrm{T}}$,$(k=1,2,\cdots,m)$,$x_i\in\{-1,1\}(i=1,2,\cdots,n)$,网络对权值的更新是逐步实现的,即对权值 W,网络连接权值的学习遵循下列模式,如式(5-25)所示。

$$\begin{cases}\boldsymbol{W}^0=0\\ k=1,2,\cdots,m\\ \boldsymbol{W}^k=\boldsymbol{W}^{k-1}+(\boldsymbol{X}^k)^{\mathrm{T}}(\boldsymbol{X}^k)-\boldsymbol{I}\end{cases}\tag{5-25}$$

从上述连接权值学习过程的描述可以看出,连接权矩阵是每次在上一次得到权值的基础上进行累加,即在原来值的基础上进行了移动,这样网络就有可能遗忘掉部分先前的记忆模式。

从动力学的角度来看,k 值较小时,记忆样本会成为该离散型 Hopfield 神经网络的稳定状态。但是随着 k 值的增加,连接权值不断移动,各个记忆模式相互交叉,不但难以使后来的样本成为该网络的稳定状态,即新的样本记不住,而且有可能遗忘之前已经记忆住的模式,这种现象称为“疲劳”。

离散型 Hopfield 神经网络在学习多个样本后,在联想回忆阶段即验证该记忆样本时,所产生的干扰称为交叉干扰。

事实上,当网络规模 n 一定时,如果需要记忆的模式很多的话,联想记忆时出现错误的可能性就会很大,反之,要求出错率越低,网络的信息存储容量上限越小。有研究表明当存储容量超过 $0.15n$ 时,联想记忆时就有可能出错,错误结果对应的是能量的局部极小点。可以通过改进网络拓扑结构或改进网络的权值设计方法提高网络存储容量。

5.3　连续型 Hopfield 神经网络

1984 年,美国加州工学院物理学家 Hopfield 使用运算放大器实现神经元,用电子线路模拟神经元的连接,设计并研制了连续型 Hopfield 神经网络(Continuous Hopfield Neural

Network,CHNN),并用它解决了旅行商计算难题。连续型 Hopfield 神经网络在原理上基本与离散型 Hopfield 神经网络一致,但连续型 Hopfield 模型神经元的输出是(0,1)区间内的连续值,且神经元之间采取同步工作的方式。因此,它在并行性、实时性、分布性和协同性等方面会比离散型 Hopfield 神经网络更接近生物神经系统。

5.3.1　连续型 Hopfield 神经网络的结构

与离散型 Hopfield 神经网络类似,连续型 Hopfield 神经网络的结构如图 5-5 所示,它是单层反馈的双向对称全互连非线性网络,每个结点的输出会反馈至结点的输入。其连接权值如式(5-26)和式(5-27)所示。

$$w_{ij}=w_{ji} \tag{5-26}$$

$$w_{ii}=0 \tag{5-27}$$

图 5-5 中,三角形状的器件为具有正反向输出的运算放大器,它的输入和输出用来模仿神经元输入和输出之间的非线性关系。电子线路与神经网络之间关系为:运算放大器——神经元;神经元输入——运放输入电压 u_j;神经元输出——运放输出电压 V_j(输出有正向输出 V_j,和反向输出 V_j);连接权值 W_{ij}——输入端电导(电阻的倒数);阈值——输入偏置电流 I_j。

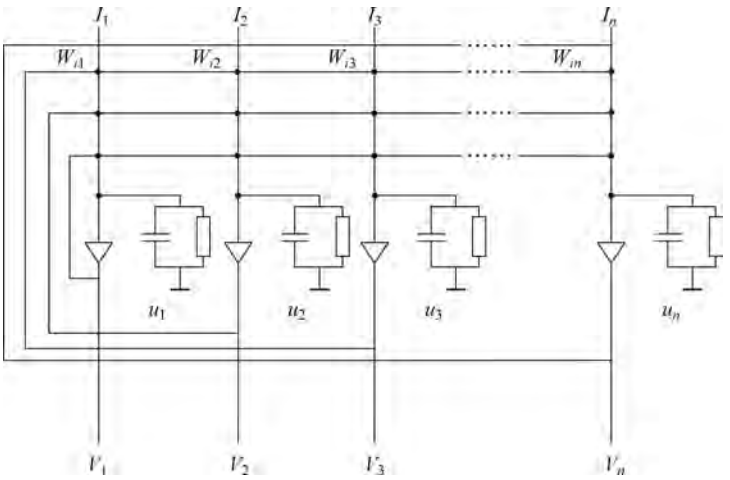

图 5-5　连续型 Hopfield 神经网络的结构

5.3.2　连续型 Hopfield 神经网络的神经元

用模拟电路实现的连续型 Hopfield 神经网络的神经元结点如图 5-6 所示。

如模拟电路所示,$(V_1,V_2,\cdots,V_n)$是电路的输入电压,表示的是上层各神经元的输出;$(R_{1j},R_{2j},\cdots,R_{nj})$是电路的连接电阻,其倒数表示的是上层各神经元与神经元 j 的连接权值;I_j 是电路的外加偏置电流,表示的是神经元 j 的阈值;U_j 是电路中放大器的输入电压,表示的是神经元 j 的净输入;电路中的等效电阻 P_j 和输入电容 C_j 存在着并联关系,这些元件决定了神经元 j 的时间常数,目的是模拟生物神经元中存在的延时特性;f 是电路的放大器,表示的是神经元 j 的激活函数;V_j 是电路中放大器的输出电压,表示的是神

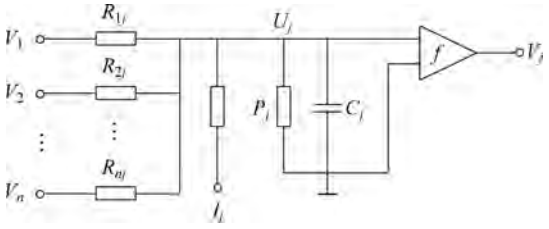

图 5-6　连续型 Hopfield 神经网络神经元

经元 j 的输出，如式(5-28)所示：

$$V_j = f(U_j) \tag{5-28}$$

其中，激活函数 $f(\cdot)$是一个 S 型曲线函数，其表达式如式(5-29)所示：

$$f(x) = \frac{1}{1+\mathrm{e}^{-2\frac{x}{s_0}}} \tag{5-29}$$

其中，s_0 为常数，决定了激活函数的曲线形状。当 s_0 趋近 0 时，则该 S 型曲线函数趋向于变为阶跃函数，这表明离散型 Hopfield 神经网络本质上是连续型 Hopfield 神经网络的一种。

5.3.3　连续型 Hopfield 神经网络的能量函数

对于连续型 Hopfield 神经网络的稳定性，同样可以采用能量函数加以判别。现假设一个具有 n 个神经元的连续型 Hopfield 神经网络，若其满足如下条件，则可判定其能达到稳定状态：①网络的结构对称，即 $w_{ij}=w_{ji}$；②当 $i=j$ 时，$w_{ij}=0$；③网络神经元的激活函数单调、连续、递增且具有反函数。满足上述条件的连续型 Hopfield 神经网络的能量值随着网络状态的变化而减少，即该网络的能量函数 E 是单调递减函数。当能量值耗尽或者达到最小时，代表连续型 Hopfield 神经网络达到稳定状态。能量函数 E 如式(5-30)所示：

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}V_iV_j + \sum_{i=1}^{n}\frac{1}{R_i}\int_1^{V_i} f^{-1}(v)\mathrm{d}v - \sum_{i=1}^{n}V_iI_i \tag{5-30}$$

下面对能量函数的单调递减性进行证明：

$$\begin{aligned}
\frac{\mathrm{d}E}{\mathrm{d}t} &= \sum_{i=1}^{n}\frac{\mathrm{d}E}{\mathrm{d}V_i}\cdot\frac{\mathrm{d}V_i}{\mathrm{d}t} \\
&= \sum_{i=1}^{n}\frac{\mathrm{d}\left(-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}V_iV_j + \sum_{i=1}^{n}\frac{1}{R_i}\int_1^{V_i} f^{-1}(v)\mathrm{d}v - \sum_{i=1}^{n}V_iI_i\right)}{\mathrm{d}V_i}\cdot\frac{\mathrm{d}V_i}{\mathrm{d}t} \\
&= \sum_{i=1}^{n}\left(-\frac{1}{2}\frac{\mathrm{d}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}V_iV_j\right)}{\mathrm{d}V_i} + \frac{\mathrm{d}\left(\sum_{i=1}^{n}\frac{1}{R_i}\int_1^{V} f^{-1}(v)\mathrm{d}v\right)}{\mathrm{d}V_i} - \right. \\
&\quad \left.\frac{\mathrm{d}\left(\sum_{i=1}^{n}V_iI_i\right)}{\mathrm{d}V_i}\right)\cdot\frac{\mathrm{d}V_i}{\mathrm{d}t}
\end{aligned} \tag{5-31}$$

其中，

$$\sum_{i=1}^{n}\frac{\mathrm{d}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}V_iV_j\right)}{\mathrm{d}V_i}=\sum_{i=1}^{n}\frac{\mathrm{d}\left(\sum\begin{bmatrix}0 & w_{12}V_1V_2 & \cdots & w_{1n}V_1V_n\\ w_{21}V_2V_1 & 0 & \cdots & w_{2n}V_2V_n\\ \vdots & \vdots & \ddots & \vdots\\ w_{n1}V_nV_1 & w_{n2}V_nV_2 & \cdots & 0\end{bmatrix}\right)}{\mathrm{d}V_i}$$

$$=2\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}V_j \tag{5-32}$$

$$\sum_{i=1}^{n}\frac{\mathrm{d}\left(\sum_{i=1}^{n}\frac{1}{R_i}\int_1^{v_i}f^{-1}(v)\mathrm{d}v\right)}{\mathrm{d}V_i}=\sum_{i=1}^{n}\frac{U_i}{R_i} \tag{5-33}$$

$$\sum_{i=1}^{n}\frac{\mathrm{d}\left(\sum_{i=1}^{n}V_iI_i\right)}{\mathrm{d}V_i}=\sum_{i=1}^{n}I_i \tag{5-34}$$

综合式(5-32)和式(5-34)可得：

$$\frac{\mathrm{d}E}{\mathrm{d}t}=\sum_{i=1}^{n}\left(-\frac{1}{2}\frac{\mathrm{d}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}V_iV_j\right)}{\mathrm{d}V_i}+\frac{\mathrm{d}\left(\sum_{i=1}^{n}\frac{1}{R_i}\int_1^{V_i}f^{-1}(v)\mathrm{d}v\right)}{\mathrm{d}V_i}-\frac{\mathrm{d}\left(\sum_{i=1}^{n}V_iI_i\right)}{\mathrm{d}V_i}\right)\cdot\frac{\mathrm{d}V_i}{\mathrm{d}t}$$

$$=\left(-\frac{1}{2}\times2\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}V_j+\sum_{i=1}^{n}\frac{U_i}{R_i}-\sum_{i=1}^{n}I_i\right)\cdot\frac{\mathrm{d}V_i}{\mathrm{d}t}$$

$$=-\sum_{i=1}^{n}\left(\sum_{j=1}^{n}w_{ij}V_j+I_i-\frac{U_i}{R_i}\right)\cdot\frac{\mathrm{d}V_i}{\mathrm{d}t} \tag{5-35}$$

代入 Hopfield 网络运动方程 $C_i\frac{\mathrm{d}U_i}{\mathrm{d}t}=\sum_{j=1}^{n}w_{ij}V_j-\frac{U_i}{R_i}+I_i$ 可得：

$$\frac{\mathrm{d}E}{\mathrm{d}t}=-\sum_{i=1}^{n}\left(\sum_{j=1}^{n}w_{ij}V_j+I_i-\frac{U_i}{R_i}\right)\cdot\frac{\mathrm{d}V_i}{\mathrm{d}t}$$

$$=-\sum_{i=1}^{n}\left(C_i\frac{\mathrm{d}U_i}{\mathrm{d}t}\right)\cdot\frac{\mathrm{d}V}{\mathrm{d}t}=-\sum_{i=1}^{n}C_i\cdot\frac{\mathrm{d}U_i}{\mathrm{d}t}\cdot\frac{\mathrm{d}V}{\mathrm{d}U_i}\cdot\frac{\mathrm{d}U_i}{\mathrm{d}t}$$

$$=-\sum_{i=1}^{n}C_i\cdot\frac{\mathrm{d}V_i}{\mathrm{d}U_i}\cdot\left(\frac{\mathrm{d}U_i}{\mathrm{d}t}\right)^2$$

$$=-\sum_{i=1}^{n}C_i\cdot f'(U_i)\cdot\left(\frac{\mathrm{d}U_i}{\mathrm{d}t}\right)^2 \tag{5-36}$$

其中，C_i 为输入电容，$C_i>0$。由上面章节可知，$V_i=f(U_i)$为单调递增函数，故其导数大于 0，而$\left(\frac{\mathrm{d}U_i}{\mathrm{d}t}\right)^2$ 为非负数，因此可得$\frac{\mathrm{d}E}{\mathrm{d}t}\leqslant0$，即能量函数单调不增。

5.4 Hopfield 神经网络的应用

Hopfield 神经网络作为一种反馈型神经网络,已经在多个领域取得了成功。Hopfield 网络的应用形式有联想记忆和优化计算两种形式,不同形式的 Hopfield 神经网络对应不同的应用形式:离散型 Hopfield 神经网络主要用于联想记忆,连续型 Hopfield 神经网络主要用于优化计算。接下来分别对离散型 Hopfield 神经网络和连续型 Hopfield 网络的应用展开说明。

5.4.1 离散型 Hopfield 神经网络的应用

离散型 Hopfield 神经网络采用内容寻址存储(Content Addressable Memory, CAM)方式存取信息,使得信息与存储地址不存在一对一的关系,即当只给出输入模式的部分信息时,神经网络能够联想出完整的信息。CAM 方式是一种分布式存储方式,因此即使少量且分散的局部信息出错,也可以忽略其对全局信息产生的影响。利用神经网络的这种良好的容错性,能够将不完整的、含噪声的、畸变的信息恢复成完整的原型,通常用于识别和分类等任务。

1. 联想记忆

神经网络虽然具有联想功能,但是不同于人类的联想功能。人类的联想是将一种事物联系到与之相关的事物或者其他事物,而神经网络的联想功能是指网络在给定的一组信号的刺激下能够回忆出与之相对应的信号。记忆是联想的前提,联想记忆的过程就是信息的存取过程。

Hopfield 神经网络也常被称为联想记忆网络,可以将其看成是非线性联想记忆或按内容寻址器。Hopfield 神经网络模拟了生物神经网络的记忆功能,采用内容寻址存储方式存取信息,根据给定的部分信息,通过不断的学习联想出完整的信息。由于采用了内容寻址存储方式,因此 Hopfield 神经网络具备了信息存储量大、高速读取、信息检索时间与信息存储量大小无关等特点。

联想记忆可以分为两种:一种是自联想记忆(auto-associative memory);另一种是异联想记忆(hetero-associative memory)。异联想记忆与自联想记忆的不同在于输出信息的形式不同。

自联想记忆:假定联想记忆网络作为联想存储器在学习过程中存储了 m 个样本,即 $\{X^k\}$;$k=1,2,3,\cdots,m$;$k=1,2,3,\cdots,m$。$X'=X^k+V$ 作为联想记忆网络在联想过程中的输入,其中 X^k 表示联想记忆网络存储的 m 个样本中的任意一个样本,V 表示偏差项(可代表噪声、缺损与畸变等),采用自联想记忆方式的联想记忆网络的输出为 X^k,即存在对应关系:$X'\rightarrow X^k$。

自联想记忆能够将某个事物的不完整的信息恢复成此事物的完整的原型。同时,自联想记忆具有容错性,能够识别含有噪声的信息。比如将一张破损的故宫照片提供给网络,能够获得一张完整的故宫照片。

异联想记忆：异联想记忆是自联想记忆的特例，假定两组模式对之间存在对应关系：$X^k \rightarrow Y^k$；$k=1,2,\cdots,m$，其中 m 表示网络记忆的样本个数，$X'=X^k+V$ 作为联想记忆网络在联想过程中的输入，其中 X^k 表示联想记忆网络存储的 m 个样本中的任意一个样本，V 表示偏差项(可代表噪声、缺损与畸变等)，采用异联想记忆方式的联想记忆网络的输出为 Y'，其中 $Y' \in \{Y^k\}$。比如将某个地方的破损的照片提供给网络，能得到某个地方的名称。

异联想记忆能够根据某一事物的信息联想出另一事物的信息，即在受到具有一定噪声的输入模式激发时，网络能够通过状态的演化联想出对应的样本模式。

2. 具体应用实例

联想记忆网络的特点决定了它对含噪声的、畸变的、缺损的信息问题有着良好的处理效果，所以联想记忆神经网络有着非常广泛的应用领域，比如模式识别、模式分类、图像处理等。在模式识别领域，可以将联想记忆神经网络用于手势识别、语音识别及人脸识别；在模式分类领域，可以用于地下工程围岩稳定性分类、遥感影像分类；在图像处理领域，可以利用联想记忆网络对模糊，残缺图像的记忆和回忆能力，对图像及复杂背景的车牌进行识别、对在运输或者印刷过程中污损的二维码图像进行复原等。

联想记忆网络的去噪能力，在智能交通方面发挥着重要作用，本节以小型汽车的车牌识别为例，重点介绍离散型 Hopfield 神经网络的构建及使用联想记忆复原车牌字符的过程。

车牌识别流程图如图 5-7 所示。

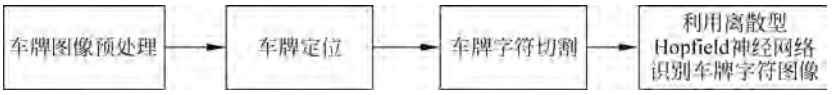

图 5-7　车牌识别流程图

首先车牌识别中出现的相关概念如下：

1）标准字符模板

标准字符模板的作用是用来匹配分割出来的车牌字符。普通民用车牌主要由汉字、字母以及阿拉伯数字三部分组成。本实例中采用的标准字符模板大小为 40×20 的图像，包括 31 个汉字，10 个阿拉伯数字(0～9)，以及 24 个大写英文字母。

2）车牌定位

车牌定位是车牌识别工作的前提，作用是排除车牌字符信息以外的干扰。

3）车牌校正

智能交通摄像机在摄取车牌图像时，由于拍摄角度、车速等原因导致采集到的车牌图像存在不同程度的倾斜，很难直接对扭曲的车牌字符进行特征提取，因此在车牌字符分割前对车牌图像进行倾斜校正，目前主要的倾斜校正方法有旋转投影法、Hough 变换法等。

4）车牌字符分割

将车牌字符单个分离出来，有利于网络进行逐个识别，准确识别出车牌号码，针对我国车牌特征，目前常用的车牌字符分割法有：基于垂直投影的字符分割法等。

5）存储容量

由先验知识可知，当 Hopfield 神经网络接受外界输入信号和需要联想匹配的样本过

多,网络不能正确联想。研究表明,在一个连续型 Hopfield 神经网络中,所需记忆样本数量超过神经元总数的 15%时,网络的联想记忆就有可能出错。本实例中标准化处理后的样本大小为 40×20,因此神经网络的神经元的个数为 40×20=800 个。网络的存储容量最多为 800×15%=120 个,因此在设计网络时,记忆样本的个数最好不要超过 120 个。

结合以上原理,构建 Hopfield 神经网络记忆标准模板字符,然后利用构造好的网络对受到噪声干扰的字符图像进行去噪,具体实现步骤如下:

(1) 确定记忆样本,对记忆样本进行编码,以便取得取值为 1 和−1 的记忆样本。将所有待记忆的且经过二值化图像处理的模板字符存储成 40×20 的二阶矩阵,将二阶矩阵转换为元素个数为 800 个的一维列向量,元素的取值只有 0 或 1,有内容的部分为 1,无字符的位置为 0。假设记忆样本个数为 m 个,将所有转换后的列向量合成一个 $n\times 800$ 的二阶矩阵 $\boldsymbol{T}$。将矩阵 $\boldsymbol{T}$ 中值为 0 的元素转化为−1,$X^k=(x_1^k,x_2^k,\cdots,x_n^k)^{\mathrm{T}}$;$k=1,2,\cdots,m(m<n)$,其中 X_k 表示第 k 个记忆模式,m 为记忆模式的个数,且不大于 120,神经元 j 的状态 x_j^k 的值为 1 或−1。

(2) 利用矩阵 $\boldsymbol{T}$ 设计网络的连接权值,采用 Hebb 学习规则的外积和法和(串行)异步工作方式构建神经网络模型。

(3) 采用产生噪声的方法(如随机噪声产生法、固定噪声产生法等)对待识别的记忆样本模拟产生带噪声的二阶矩阵,同时将二阶矩阵转化为元素个数为 800 个的一维列向量。

(4) 初始化网络状态,将步骤(3)获得的含有噪声的列向量 $\boldsymbol{X}'=(x_1',x_2',\cdots,x_n')^{\mathrm{T}}$ 作为输入样本载入网络,$x_i(0)=x_i'(i=1,2,\cdots,n)$,$x_i(t)$表示神经元 i 在 t 时刻的输出状态。

(5) 随机更新网络中某一个神经元的状态,反复迭代直到网络状态为稳定状态,网络输出去除噪声的车牌字符。

(1)、(2)属于离散型 Hopfield 神经网络的记忆阶段,(3)～(5)属于联想阶段。

5.4.2　连续型 Hopfield 神经网络的应用

1. 组合优化问题

连续型 Hopfield 神经网络主要应用于优化计算,换句话说,就是解决组合优化问题。组合优化问题也就是最优解问题,是指在给定的约束条件下,求出使目标函数获得极小值(或极大值)的变量组合问题。

2. 连续型 Hopfield 神经网络解决组合优化问题

连续型 Hopfield 神经网络解决组合优化问题利用了网络的能量函数 E 在网络状态的变化过程中单调递减的原理。把需要优化的问题映射到一种神经网络的特定组合状态上,能量函数 E 等价于解决优化问题的代价函数,当网络处于稳定状态时,能量函数 E 收敛于极小值点,此时的网络状态就是优化问题可能出现的解。连续型 Hopfield 神经网络计算优化问题的步骤如下:

(1) 选择一种合适的表示方法表示待优化的问题,使得神经网络的输出与待优化问题的解对应;

(2) 构造能量函数 E,使能量函数的最小值点与最优解相对应;

(3) 将步骤(2)中构造的能量函数 E 与连续型 Hopfield 神经网络的能量函数做对比，反向推出神经网络的连接权值和结构；

(4) 由步骤(3)获得的网络结构所建立的网络的稳定运行状态，就是待优化问题的最优解。

3. 连续型 Hopfield 神经网络的具体应用举例

旅行商最优路径问题(Traveling Salesman Problem，TSP)是要找出一条最短路线，这条路线经过每个城市恰好一次，并回到出发点。假设有 n 个城市的集合$\{C_i\}$；$i=1,2,\cdots,n$，任意两个城市 C_x 与 C_y 之间的距离用 d_{xy} 表示，n 个城市之间可能的路线有$\frac{(n-1)!}{2}$条。

目前，旅行商问题的解决方法(穷举法、贪心法等)存在“组合爆炸”问题，即当城市数 n 较大时，冯·诺伊曼体系结构的计算机在有限时间内无法获得答案。直到 1984 年，Hopfield 和 Tank 使用连续型 Hopfield 网络成功解决了当 $n=30$ 时的旅行商问题，为解决组合优化 NP-hard 问题开辟了一条新的途径，展现了神经网络的优越性。其主要解决步骤包括：将旅行商问题转换成适合神经网络处理的形式，用 $n\times n$ 个神经元构成的矩阵表示旅行路线，然后利用约束条件构造能量函数，确定连接权值，运行网络得到结果。旅行商问题优化计算过程如下：

第一步：选择合适的表示方法。

旅行商问题的解是 n 个城市的有序排列，并且任意城市在任意一条路线上的位置可以用 n 维向量表示，因此可以使用 n 阶矩阵 $\boldsymbol{V}$ 来描述一次有效的旅行路线。矩阵中的元素与网络中的神经元对应，$V_{xi}(x=1,2,\cdots,n;\ i=1,2,\cdots,n)$表示神经元 x_i 的状态，其中，x 表示第 x 个城市 C_x；i 表示访问的顺序。矩阵中的元素表示某一城市在某一条有效路线中的位置，行向量表示城市 $C_x(x=1,2,\cdots,n)$的位置，列向量表示路线中第 $i(i=1,2,\cdots,n)$个位置上的城市。

$$V_{xi}=\begin{cases}1, & C_x \text{ 是路线中第 } i \text{ 个被访问的城市}\\ 0, & C_x \text{ 不是路线中第 } i \text{ 个被访问的城市}\end{cases}$$

假如有 4 个城市 C_1、C_2、C_3、C_4，一条有效路线为 $C_2\to C_1\to C_4\to C_3\to C_2$，旅行路线总长为 $d=d_{21}+d_{14}+d_{43}+d_{32}$，矩阵表示形式如图 5-8 所示。

城市	次序			
	1	2	3	4
C_1	0	1	0	0
C_2	1	0	0	0
C_3	0	0	0	1
C_4	0	0	1	0

图 5-8　4 个城市 TSP 中一次可能的路线

第二步：构造能量函数。

解决旅行商问题的核心步骤是设计能量函数，首先分析旅行商问题和对应矩阵 V 的特

点,总结如下:

(1) 一个城市只能被访问一次,即矩阵行向量有且仅有 1 个元素为 1,其余 $n-1$ 个元素为 0。

(2) 一次只能访问一个城市,即矩阵列向量有且仅有 1 个元素为 1,其余 $n-1$ 个元素为 0。

(3) 每个城市都要到过一次,即矩阵 $\boldsymbol{V}$ 中有 n 个数值为 1 的元素。

(4) 旅行路线最短,即网络能量函数的最小值对应旅行商问题的最优解,即访问路线的最短距离。

结合上述特点可知,在连续型 Hopfield 神经网络能量函数的一般性基础上设计旅行商问题的能量函数时,还需考虑以下两点要求:

(1) TSP 的能量函数需要量化地翻译置换矩阵的规则。

(2) 能量函数要有利于量化表示最短路线的解。

在考虑以上两点要求之后,将 TSP 的能量函数划分成 4 部分:

(1) 设计 TSP 能量函数的第一项 E_1。

由特点(1)可知第 x 行的全部元素按顺序两两相乘之和为 0,即 $\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}V_{xi}V_{xj}=0$,其中位置 j 表示与 i 相邻的位置,从而全部 n 行的所有元素按顺序两两相乘之和也应为 0,即 $\sum_{x=1}^{n}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}V_{xi}V_{xj}=0$,因此由行约束构造的能量函数 E_1 表达式如式(5-37)所示:

$$E_1=\frac{A}{2}\sum_{x=1}^{n}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}V_{xi}V_{xj} \tag{5-37}$$

其中,A 为大于 0 的常数。显然,当 $E_1=0$ 时可保证对每个城市访问的次数不超过一次。

(2) 设计 TSP 能量函数的第二项 E_2。

同理,由特点(2)可知第 i 列的全部元素按顺序两两相乘之和为 0,即 $\sum_{x=1}^{n-1}\sum_{y=x+1}^{n}V_{xi}V_{yi}=0$,其中 y 表示与城市 x 相邻的城市,从而全部 n 列的所有元素按顺序两两相乘之和也应为 0,即 $\sum_{i=1}^{n}\sum_{x=1}^{n-1}\sum_{y=x+1}^{n}V_{xi}V_{yi}=0$,因此由列约束构造的能量函数 E_2 表达式如式(5-38)所示:

$$E_2=\frac{B}{2}\sum_{i=1}^{n}\sum_{x=1}^{n-1}\sum_{y=x+1}^{n}V_{xi}V_{yi} \tag{5-38}$$

其中,B 为大于 0 的常数。显然,当 $E_2=0$ 时可保证每次访问的城市个数不超过一个。

(3) 设计 TSP 能量函数的第三项 E_3。

由特点(3)可知矩阵中 $V_{xi}=1$ 的数目等于城市数 n,即 $\sum_{x=1}^{n}\sum_{i=1}^{n}V_{xi}=n$ 因此由全局约束条件得到能量函数 E_3 的表达式如式(5-39)所示:

$$E_3=\frac{C}{2}\left(\sum_{x=1}^{n}\sum_{i=1}^{n}v_{xi}-n\right)^2 \tag{5-39}$$

其中,C 为大于 0 的常数。$E_3=0$ 可保证总共访问的城市个数为 n 个,同时平方项也表示了当访问城市总数不为 n 时的一种惩罚。

(4) 设计 TSP 能量函数的第四项 E_4。

同时满足以上约束条件(1)(2)(3)只能保证路线是有效的，并不能保证一定是最优的，即路线总长度不一定最短。因此，在保证路线有效的前提下，在能量函数中加入能反映路线长度的能量分量 E_4，同时保证 E_4 随路线总长度的缩短而减小。

设计 E_4 时，考虑到访问任意两城市 C_x 与 C_y 有以下两种状况。

情况一：先访问城市 C_x，再访问城市 C_y，即 $C_x \to C_y$，相应的距离表达式为 $d_{xy}V_{xi}V_{y,i+1}$。

情况二：先访问城市 C_y，再访问城市 C_x，即 $C_y \to C_x$，相应的距离表达式为 $d_{xy}V_{xi}V_{y,i-1}$。

如果城市 C_x 和城市 C_y 在一条路线中相邻，结合上述两种情况可得式(5-40)。

$$\begin{cases} C_x \to C_y：d_{xy}V_{xi}V_{y,i+1}=1, \quad d_{xy}V_{xi}V_{y,i-1}=0 \\ C_y \to C_x：d_{xy}V_{xi}V_{y,i+1}=0, \quad d_{xy}V_{xi}V_{y,i-1}=1 \end{cases} \tag{5-40}$$

因此，可以定义城市 C_x 和城市 C_y 之间的距离为式(5-41)所示。

$$d_{xy}=d_{xy}V_{xi}V_{y,i+1}+d_{xy}V_{xi}V_{y,i-1}=d_{xy}V_{xi}(V_{y,i+1}+V_{y,i-1}) \tag{5-41}$$

由路线长度约束条件得到的能量函数 E_4 表达式如式(5-42)所示。

$$E_4=\frac{D}{2}\sum_{x=1}^{n}\sum_{y=1}^{n}\sum_{i=1}^{n}d_{xy}V_{xi}(V_{y,i+1}+V_{y,i-1}) \tag{5-42}$$

其中，$\sum_{x=1}^{n}\sum_{y=1}^{n}\sum_{i=1}^{n}d_{xy}V_{xi}(V_{y,i+1}V_{y,i-1})$ 表示 n 个城市两两之间所有可能的访问路线的长度，D 为大于 0 的常数。当 E_4 达到最小值时保证当前路线为最短路线。

综合式(5-37)～式(5-42)，可得解决旅行商问题的连续型 Hopfield 神经网络的能量函数 E 如式(5-43)所示：

$$\begin{aligned} E&=E_1+E_2+E_3+E_4 \\ &=\frac{A}{2}\sum_{x=1}^{n}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}V_{xi}V_{xj}+\frac{B}{2}\sum_{i=1}^{n}\sum_{x=1}^{n-1}\sum_{y=x+1}^{n}V_{xi}V_{yi}+\frac{C}{2}\left(\sum_{x=1}^{n}\sum_{i=1}^{n}v_{xi}-n\right)^2+ \\ &\quad\frac{D}{2}\sum_{x=1}^{n}\sum_{y=1}^{n}\sum_{i=1}^{n}d_{xy}V_{xi}(V_{y,i+1}+V_{y,i-1}) \end{aligned} \tag{5-43}$$

其中，参数 A、B、C、D 称为权值；E_1～E_3 称为惩罚项，只有在满足约束条件的情况下，TSP 能量函数的分量 E_1～E_3 才为 0，保证了路线的有效性；E_4 对应组合优化问题的目标函数，其最小值表示最短路线长度，保证了路线的合理性。

针对旅行商问题，网络的能量函数达到极小值的前提是要满足约束条件，也就是说能量函数分量 E_1～E_3 均为 0 是网络的能量函数达到极小值的前提。

第三步：确定神经元间的连接权值。

设置网络的初始连接权值的目的是使网络能够收敛到全局极小值。对比式(5-30)与式(5-43)，可得式(5-44)。

$$\begin{cases} w_{xi,yi}=-A\delta_{xy}(1-\delta_{ij})-B\delta_{ij}(1-\delta_{xy})-C-Dd_{xy}(\delta_{j,i+1}+\delta_{j,i-1}) \\ \theta_{xi}=C_n \end{cases} \tag{5-44}$$

其中，$w_{xi,yi}$ 表示神经元 x_i 与神经元 y_j 的初始连接权值；θ_{xi} 表示外部激励(输入电流)，

δ 函数定义如式(5-45)所示。

$$\delta_{xy}=\begin{cases}1, & x=y\\ 0, & x\neq y\end{cases}\qquad \delta_{ij}=\begin{cases}1, & i=j\\ 0, & i\neq j\end{cases} \tag{5-45}$$

运行构造好的网络

$$\begin{cases}C_{xi}\dfrac{\mathrm{d}u_{xi}}{\mathrm{d}t}=-A\sum\limits_{j=1,j\neq i}^{n}V_{xi}-B\sum\limits_{j=1,y\neq x}^{n}V_{yi}-C\left(\sum\limits_{x=1}^{n}\sum\limits_{i=1}^{n}V_{xi}-n\right)-\\ \qquad\qquad D\sum\limits_{y=1,y\neq x}^{n}d_{xy}(V_{y,i+1}+V_{y,i-1})-\dfrac{u_{xi}}{R_{xi}}\\ V_{xi}=f(u_{xi})=\dfrac{1}{1+\mathrm{e}^{\frac{-2u_{xi}}{u_0}}}\end{cases} \tag{5-46}$$

给定一个随机的初始值输入,使得构造好的网络按照式(5-46)运行。式(5-46)中 u_0 是初始值,输入的初始值不同,得到的网络的稳定状态也不同,即不同的初始输入得到不同的旅行路线,这些路线可能是旅行商问题的最优解或者次优解。

使用连续型 Hopfield 神经网络解决 TSP 问题时需要注意以下问题。

(1) 在初始化连续型 Hopfield 神经网络时,可以将初始值设为任意值。但是初始值会影响程序运行的速度和精确度,因此为了达到更好的效果,要不停地对初始值进行调试。

(2) 由于每次运行程序时,输入状态都是在给定范围内的随机数,因此每次计算得到的次优解数据也会略有不同,但是只要多次运行程序就能发现多次迭代后的最优解只有一个。

(3) 用 Hopfield 神经网络求解 TSP 问题是较为有效的,但是很明显,由于很难选择合适的惩罚参数,导致利用 Hopfield 神经网络求解组合优化问题对缺乏经验的神经网络开发者来说不是一个容易实现的工作,反倒是经典算法中的分治法等方法对初学者更为友好。

(4) Hopfield 神经网络虽然可以求解小型 TSP,但是对于大规模的 TSP 求解很难获得较好的效果,因此要像其他大规模组合优化问题一样先分解成子问题再解决。


微课视频

5.5 本章实践

5.5.1 离散型 Hopfield 神经网络实践

离散型 Hopfield 神经网络的一项重要应用就是联想记忆,它能够将一些样本模式存储在不同的能量极值点上。联想的前提是记忆,首先应该把相应的信息保存起来,才能按照某种方式或规则读取出相关信息。联想记忆分为自联想记忆和异联想记忆,离散型 Hopfield 属于自联想记忆。

本次实验以记忆数字矩阵为例。

1. 定义要记忆的信息

```
Vmat1 = np.zeros((5, 5))
Vmat1[2, :] = 1
```

```
Vmat1[:, 2] = 1
print("V1",Vmat1)
plt.imshow(Vmat1)
plt.title("加")
plt.show()
```

V1 的结果如下。数字矩阵“加”如图 5-9 所示。

```
[[0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0.]
 [1. 1. 1. 1. 1.]
 [0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0.]]
```

类似的记忆“乘”-“×”,如图 5-10 所示。

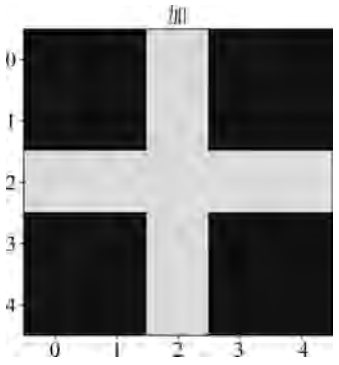

图 5-9　数字矩阵“加”

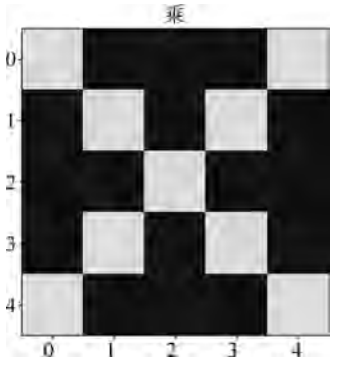

图 5-10　数字矩阵“乘”

2. 定义 Hopfield 模型

```
node_nums = 25
Vs = [Vmat1.reshape(-1), Vmat2.reshape(-1)]          # 拼接
hopnet = HopfieldNet(node_nums, Vs)
```

3. 模型训练

对于权值矩阵的获得可以分别采用下面两种方式：直接计算法和 Hebb 学习规则。

```
def __init__(self, node_nums, Vs):
    self.node_nums = node_nums
    self.W = np.zeros((node_nums, node_nums))
    for i in range(node_nums):
        for j in range(node_nums):
            if i == j:
                self.W[i, j] = 0
            else:
                self.W[i, j] = sum([(2 * Vs[a][i] - 1) * (2 * Vs[a][j] - 1) for a in
range(len(Vs))])
```

```
def learnW(self, Vs):
    for i in range(100):
        for j in range(len(Vs)):
            for c in range(len(Vs[j])):
                for r in range(len(Vs[j])):
                    if c != r:
                        if Vs[j][c] == Vs[j][r]:
                            self.W[c, r] += 0.1
                        else:
                            self.W[c, r] -= 0.1
```

4. 模型测试

定义测试数据,残缺数据考验 Hopfield 联想能力。模型测试如图 5-11 所示。

```
testmat = np.zeros((5, 5))
testmat[2, :] = 0
testmat[:, 2] = 0
testmat[2, 2] = 1
testmat[0, 0] = 1
plt.imshow(testmat)
plt.title("测试")
plt.show()
v = testmat.reshape(-1)
```

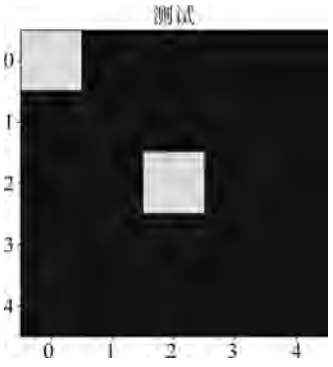

图 5-11 模型测试

测试数据 reshape 后导入 Hopfield 模型。

```
def fit(self, v):
    new_v = np.zeros(len(v))
    print("new",new_v)
    indexs = range(len(v))
    while np.sum(np.abs(new_v - v)) != 0:
        new_v = copy.deepcopy(v)
        for i in indexs:
            temp = np.dot(v, self.W[:, i])
```

```
            if temp >= 0:
                v[i] = 1
            else:
                v[i] = 0
    return v
```

5. 查看测试结果

测试结果如图 5-12 所示。

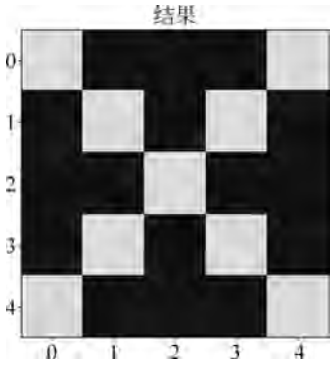

图 5-12　测试结果

可以看出 Hopfield 神经网络成功地对数据进行了联想，但是这种数据的联想不是稳定的，记忆有一定的记忆容量。并且，权值矩阵会随着记忆模式的引入发生偏移，神经元的记忆顺序与联想顺序也会影响模型结果。综上，程序的流程图如图 5-13 所示。

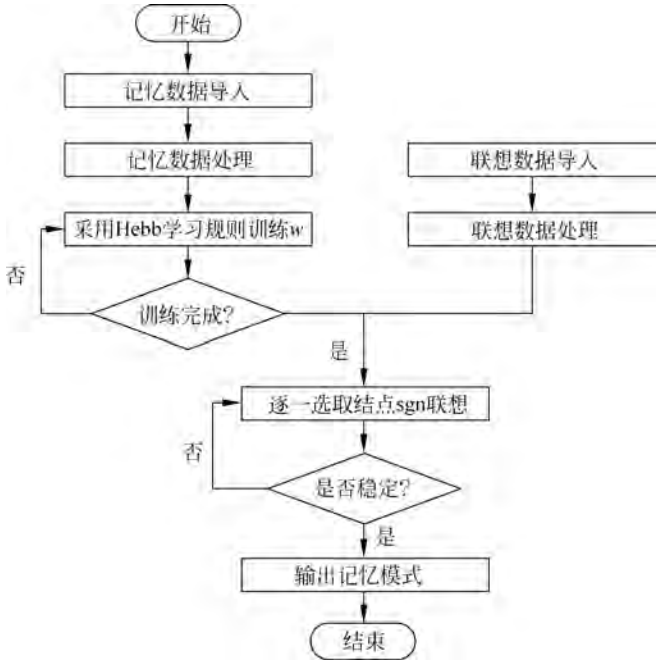

图 5-13　离散 Hopfield 实现记忆数字矩阵程序流程图

5.5.2　连续型 Hopfield 神经网络实践

本实验通过连续型 Hopfield 神经网络解决 TSP 问题。

1. 将 TSP 转化成适合于神经网络处理的形式

鉴于 TSP 的解是 n 个城市的有序排列，因此，可以使用一个由 $n\times n$ 个神经元构成的矩阵来描述旅行路线。该矩阵中的每个元素对应神经网络中的每个神经元，神经元的状态可以表示某一城市在某一条有效路线中的位置。例如，神经元 x_i 的状态用 V_{xi}($x=1,2,\cdots,n$；$i=1,2,\cdots,n$)表示，其中 x 表示第 x 个城市 C_x，i 表示 C_x 是一条有效路线中第 i 个经过的城市，$V_{xi}=1$ 表示城市 C_x 在路线中第 i 个位置出现，$V_{xi}=0$ 表示城市 C_x 在路线中第 i 个位置不出现，此时第 i 个位置上为其他城市。

由此可见，$n\times n$ 矩阵 $\boldsymbol{V}$ 可以表示解决 n 个城市 TSP 的一次有效的旅行路线，即矩阵 $\boldsymbol{V}$ 可以唯一地确定对所有城市的访问次序。例如，对于 8 个城市 TSP，一次有效路线构成的矩阵 $\boldsymbol{V}$ 中的各个元素，如图 5-14 所示，其中纵向表示城市名，横向表示每个城市在矩阵 $\boldsymbol{V}$ 中的访问次序。

城市	次序							
	1	2	3	4	5	6	7	8
C_1	0	0	0	0	1	0	0	0
C_2	0	1	0	0	0	0	0	0
C_3	0	0	0	0	0	1	0	0
C_4	1	0	0	0	0	0	0	0
C_5	0	0	1	0	0	0	0	0
C_6	0	0	0	0	0	0	0	1
C_7	0	0	0	0	0	0	1	0
C_8	0	0	0	1	0	0	0	0

图 5-14　8 个城市 TSP 中的一次可能路线

由图 5-14 可知，这 8 个城市的访问顺序为 $C_4\rightarrow C_2\rightarrow C_5\rightarrow C_8\rightarrow C_1\rightarrow C_3\rightarrow C_7\rightarrow C_6\rightarrow C_4$，旅行路线总长为 $d=d_{42}+d_{25}+d_{58}+d_{81}+d_{13}+d_{37}+d_{76}+d_{64}$。

通过分析 TSP 的定义以及对应的矩阵 $\boldsymbol{V}$ 可以发现，置换矩阵负责翻译并遵守 TSP 的规则：

(1) 一个城市只能被访问一次，即矩阵每行有且只有一个 1，其余元素均 0。

(2) 一次只能访问一个城市，即矩阵每列有且只有一个 1，其余元素均为 0。

(3) 一共要访问 n 个城市，即矩阵的全部元素中 1 的数量为 n。

在神经网络迭代优化过程中，每次神经元输出的状态集合只要满足上述置换矩阵的规则，则证明该组输出状态就是一个 TSP 问题的解，只要在这些解中找到最小代价的解即可。其中(1)～(3)反映了路线的有效性。

2. 构造能量函数

要让旅行路线最短，就需要网络能量函数的最小值对应于 TSP 最短路线的距离。在分析 TSP 及其对应的矩阵 $\boldsymbol{V}$ 的特点之后，我们可以发现，构造合适的能量函数是解决 TSP 的关键步骤。

对于 TSP 问题，在 CHNN 能量函数的一般性基础上，需要考虑以下两点：

(1) TSP 的能量函数需要量化地翻译置换矩阵的规则。

(2) 在 TSP 问题中的 $n!$ 条合法路线中，能量函数要有利于量化表示最短路线的解。

在考虑上述的要求之后，将 TSP 的能量函数划分成 4 部分。

(1) 设计 TSP 能量函数的第一项，如式(5-47)所示：

$$E_1=\frac{A}{2}\sum_{x=1}^{n}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}v_{xi}v_{xj} \tag{5-47}$$

其中，A 为常数，并且 $A>0$。即每一行中的每一个城市 x，必须有且只有一个 1，符合置换矩阵的第一条规则。

(2) 设计 TSP 能量函数的第二项，如式(5-48)所示：

$$E_2=\frac{B}{2}\sum_{i=1}^{n}\sum_{x=1}^{n-1}\sum_{y\neq x}^{n}v_{xi}v_{yi} \tag{5-48}$$

其中，B 为常数，并且 $B>0$。即每一列中的每一个城市 x，必须有且只有一个 1，符合置换矩阵的第二条规则。

(3) 设计 TSP 能量函数的第三项，如式(5-49)所示：

$$E_3=\frac{C}{2}\left(\sum_{x=1}^{n}\sum_{i=1}^{n}v_{xi}-n\right)^2 \tag{5-49}$$

其中，C 为常数，并且 $C>0$。即整个矩阵有 n 个 1，符合置换矩阵的第三条规则。

(4) 设计 TSP 能量函数的第四项，如式(5-50)所示：

$$E_4=\frac{D}{2}\sum_{x=1}^{n}\sum_{y\neq x}^{n}\sum_{i=1}^{n}d_{xy}v_{xi}(v_{y,i+1}+v_{y,i-1}) \tag{5-50}$$

其中，D 为常数，并且 $D>0$。式(5-50)包含神经网络输出中有效解的路径长度信息，d_{xy} 表示城市 x 到城市 y 的距离。

当 E_4 保证当前路线为最短时，E_4 也达到了最小值。

整合 E_1、E_2、E_3、E_4 后，得到 TSP 的能量函数 E，如式(5-51)所示：

$$\begin{aligned}E&=E_1+E_2+E_3+E_4\\&=\frac{A}{2}\sum_{x=1}^{2}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}v_{xi}v_{xj}+\frac{B}{2}\sum_{i=1}^{n}\sum_{x=1}^{n-1}\sum_{y\neq x}^{n}v_{xi}v_{yi}+\frac{C}{2}\left(\sum_{x=1}^{n}\sum_{i=1}^{n}v_{xi}-n\right)^2+\\&\quad\frac{D}{2}\sum_{x=1}^{n}\sum_{y\neq x}^{n}\sum_{i=1}^{n}d_{xy}v_{xi}(v_{y,i+1}+v_{y,i-1})\end{aligned} \tag{5-51}$$

其中，参数 A、B、C、D 称为权值，前三项是满足 TSP 置换矩阵的约束条件，称为惩罚项；最后一项包含优化目标函数项，即优化要求——使得旅行路线最短。

3. 确定权值连接

为了使网络能够收敛到全局极小值，需要设置网络的初始连接权值。将整合后的 TSP

能量函数 E 进行优化,可以得到以下表达式。

(1) 网络的初始连接权值,如式(5-52)所示:

$$w_{xi,yi}=-A\delta_{xy}(1-\delta_{ij})-B\delta_{ij}(1-\delta_{xy})-C-Dd_{xy}(\delta_{j,i+1},\delta_{j,i-1}) \tag{5-52}$$

其中,δ 函数定义,如式(5-53)所示:

$$\delta_{ij}=\begin{cases}1, & i=j\\ 0, & i\neq j\end{cases} \tag{5-53}$$

(2) 外部激励(输入电流),如式(5-54)所示:

$$\theta_{xi}=C_n \tag{5-54}$$

4. 运行网络

给定一个随机的初始值输入,使得网络按照如式(5-55)运行。

$$\begin{cases}C_n\dfrac{\mathrm{d}u_{xi}}{\mathrm{d}t}=-A\sum\limits_{j=1,j\neq i}^{n}v_{xj}-B\sum\limits_{j=1,y\neq x}^{n}v_{yi}-C\left(\sum\limits_{x=1}^{n}\sum\limits_{i=1}^{n}v_{xi}-n\right)-\\ \qquad D\sum\limits_{y=1,y\neq x}^{n}d_{xy}(v_{y,i+1}+v_{y,i-1})-\dfrac{u_{xi}}{R_{xi}}\\ v_{xi}=f(u_{xi})=\dfrac{1}{1+\mathrm{e}^{\frac{-2u_{xi}}{u_0}}}\end{cases} \tag{5-55}$$

得到网络稳定状态,即对应着一条最优的旅行路线。初始值不同也将会得到不同的稳定状态,即得到不同的旅行路线,这些旅行路线不是最优路线,就是次优路线。在图 5-15 和图 5-16 给出了两个解。

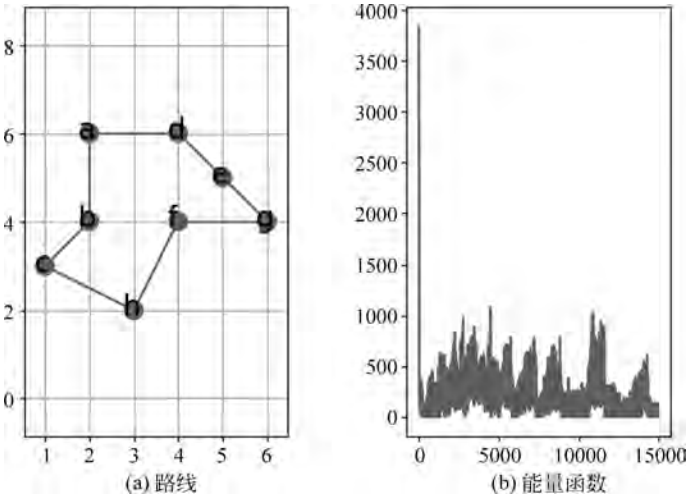

图 5-15　8 城市 TSP 最优解

这是程序最后输出的一组哈密顿回路和能量趋势图,左图表示找到的最短距离路线,右图表示在神经网络的优化过程中能量函数的波动。从图 5-15 和图 5-16 可以看到,初值设置得当时,Hopfield 神经网络找到的最优解已经和真实最优解一致。

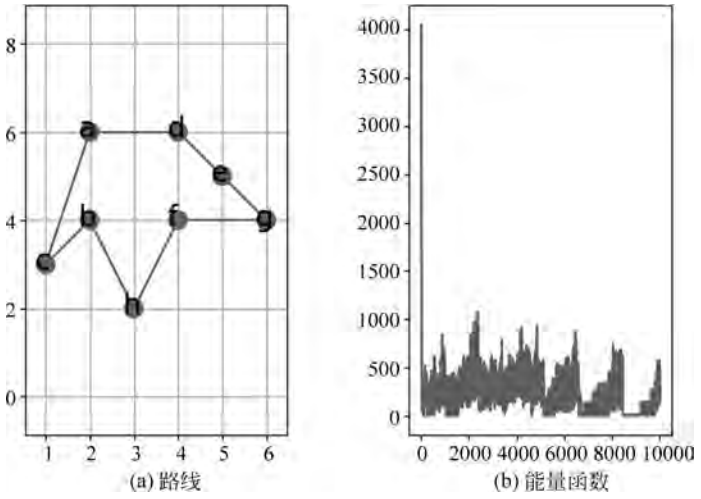

图 5-16　8 城市 TSP 次优解

5. 代码实现过程

1）初始化 Hopfield 神经网络的初值（如输入电压 U_0、迭代次数）和权值（A、D）

```
# 给定城市位置
citys = np.array([[2, 6], [2, 4], [1, 3], [4, 6], [5, 5], [4, 4], [6, 4], [3, 2]])
N = len(citys)
# 设置初始值
A = N * N
D = N / 2
U0 = 0.0009                          # 初始电压
step = 0.0001                        # 步长
num_iter = 10000                     # 迭代次数
```

2）构建 n 个城市之间的距离矩阵 D_{xy}

```
# 得到城市之间的距离矩阵
def get_distance(citys):
    N = len(citys)
    distance = np.zeros((N, N))
    for i, curr_point in enumerate(citys):
        line = []
        [line.append(price_cn(curr_point, other_point)) if i != j else line.append(0.0) for j,
other_point in enumerate(citys)]
        distance[i] = line
    return distance

distance = get_distance(citys)
```

3) 利用 CHNN 动态方程计算输入状态的增量

```
# 动态方程计算微分方程 du
def calc_du(V, distance):
    a = np.sum(V, axis = 0) - 1                    # 按列相加
    b = np.sum(V, axis = 1) - 1                    # 按行相加
    t1 = np.zeros((N, N))
    t2 = np.zeros((N, N))
    for i in range(N):
        for j in range(N):
            t1[i, j] = a[j]
    for i in range(N):
        for j in range(N):
            t2[j, i] = b[j]
    # 将第一列移动到最后一列
    c_1 = V[:, 1:N]
    c_0 = np.zeros((N, 1))
    c_0[:, 0] = V[:, 0]
    c = np.concatenate((c_1, c_0), axis = 1)
    c = np.dot(distance, c)
    return - A * (t1 + t2) - D * c
```

4) 分别由一阶欧拉方法和 Sigmoid 函数更新神经网络下个时刻的输入和输出状态

```
# 更新神经网络的输入电压 U
def calc_U(U, du, step):
    return U + du * step
# 更新神经网络的输出电压 V
def calc_V(U, U0):
    return 1 / 2 * (1 + np.tanh(U / U0))
```

5) 计算当前的能量函数 E

```
def calc_energy(V, distance):
    t1 = np.sum(np.power(np.sum(V, axis = 0) - 1, 2))
    t2 = np.sum(np.power(np.sum(V, axis = 1) - 1, 2))
    idx = [i for i in range(1, N)]
    idx = idx + [0]
    Vt = V[:, idx]
    t3 = distance * Vt
    t3 = np.sum(np.sum(np.multiply(V, t3)))
    e = 0.5 * (A * (t1 + t2) + D * t3)
    return e
```

6) 检查当前神经网络的输出状态集合,是否满足 TSP 置换矩阵的规则

```
# 检查路径的正确性
def check_path(V):
```

```
    newV = np.zeros([N, N])
    route = []
    for i in range(N):
        mm = np.max(V[:, i])
        for j in range(N):
            if V[j, i] == mm:
                newV[j, i] = 1
                route += [j]
                break
    return route, newV
```

以上 3)～6)是在迭代训练优化 Hopfield 神经网络过程中的主要步骤，具体实现如下。

```
    n = 0
# 开始迭代训练网络
while n < num_iter:
    # 利用动态方程计算 du
    du = calc_du(V, distance)
    #更新下个时间的输入状态(电路的输入电压 U)
    U = calc_U(U, du, step)
    #更新下个时间的输出状态(电路的输出电压 V)
    V = calc_V(U, U0)
    # 计算当前网络的能量 E
    energys[n] = calc_energy(V, distance)
    # 检查路径的合法性
    route, newV = check_path(V)
    if len(np.unique(route)) == N:
        route.append(route[0])
        dis = calc_distance(route)
        if dis < best_distance:
            H_path = []
            best_distance = dis
            best_route = route
            [H_path.append((route[i], route[i + 1])) for i in range(len(route) - 1)]
            print('第{}次迭代找到的次优解距离为：{}，能量为：{}，路径为：'.format(n, best_
distance, energys[n]))
            [print(chr(97 + v), end = ',' if i < len(best_route) - 1 else '\n') for i, v in
enumerate(best_route)]
    n = n + 1
```

根据给定的初值，Hopfield 神经网络在经过 10000 次迭代优化后，找到的 TSP 的最优路线解如图 5-17 所示。

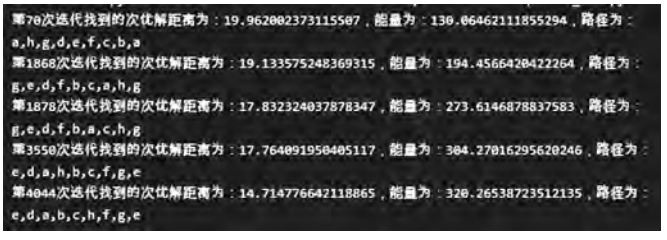

图 5-17　Hopfield 神经网络找到的 TSP 路线

综上,该程序实现的具体流程如图 5-18 所示。

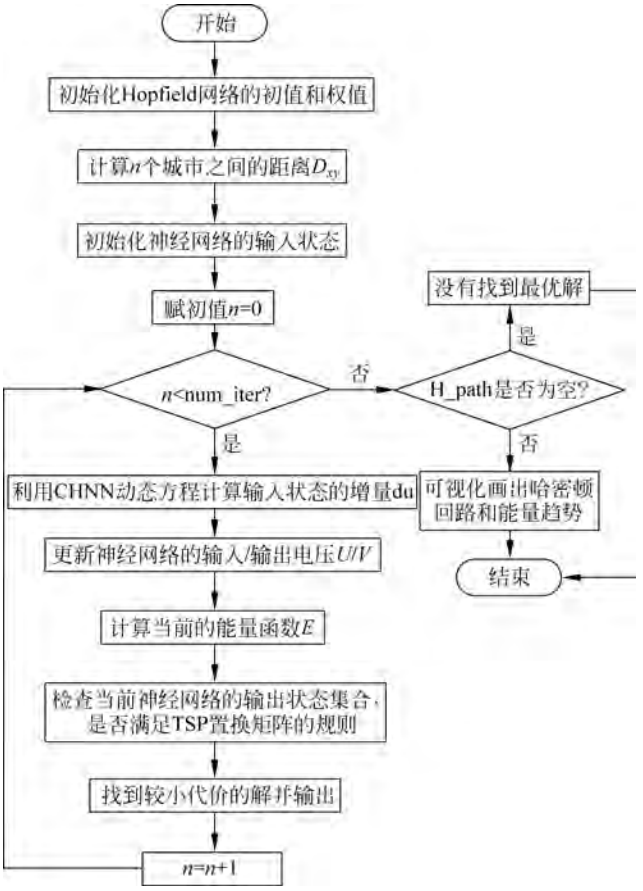

图 5-18　连续型 Hopfield 神经网络解决 TSP 问题的流程图

5.6　习题

1. 如何判断离散型 Hopfield 神经网络是否达到稳定状态?
2. 设计离散型 Hopfield 神经网络,并考察其联想性能。

$$X=T=\begin{bmatrix}1 & -1 & 1\\ 1 & -1 & -1\\ -1 & 1 & 1\end{bmatrix}$$

3. 试分析自联想记忆与互联想记忆的异同,并设计一个简单的联想记忆存储器。
4. 为什么连续型 Hopfield 神经网络能解决组合优化问题?
5. 为什么离散型 Hopfield 神经网络具有联想记忆功能?
6. 离散型 Hopfield 神经网络的训练过程主要为哪几个步骤?
7. 连续型 Hopfield 神经网络解决优化问题的一般步骤。
8. 解决 TSP 问题的算法有:神经网络算法、_____、_____、_____等。
9. 请实现离散型 Hopfield 神经网络的联想记忆功能。