第5章

MATLAB自组织神经网络

脑神经科学研究表明：传递感觉的神经元排列是按某种规律有序进行的，这种排列往往反映所感受的外部刺激的某些物理特征。例如，在听觉系统中，神经细胞和纤维是按照其最敏感的频率分布而排列的。为此，Kohonen 认为，神经网络在接受外界输入时，将会分成不同的区域，不同的区域对不同的模式具有不同的响应特征，即不同的神经元以最佳方式响应不同性质的信号激励，从而形成一种拓扑意义上的有序图。这种有序图也称为特征图，它实际上是一种非线性映射关系，它将信号空间中各模式的拓扑关系几乎不变地反映在这张图上，即各神经元的输出响应上。由于这种映射是通过无监督的自适应过程完成的，所以也称它为自组织特征图。

在这种网络中，输出节点与其邻域其他节点广泛相连，并相互激励。输入节点和输出节点之间通过强度 $W_{ij}(t)$ 相连接。通过某种规则，不断地调整 $W_{ij}(t)$，使得在稳定时，每一邻域的所有节点对某种输入具有类似的输出，并且它的聚类的概率分布与输入模式的概率分布相接近。自组织神经网络最大的优点是自适应权值，极大方便寻找最优解，但同时，在初始条件较差时，易陷入局部极小值。

而本章要讲的自组织神经网络是一类采用无教师学习方式的神经网络模型。它无须期望输出，只是根据数据样本进行学习，并调整自身的权重以达到训练的目的，这也是自组织名称的由来。自组织神经网络的学习规则大都采用竞争型的学习规则，除了学习规则是以竞争型规则为主以外，自组织神经网络的结构也是不同的，有一维输出层的、二维输出层的、带有层反馈的等。模型不同，相应的竞争型学习算法也有变化。

5.1 自组织特征映射网络

自组织特征映射网络也称为 Kohonen 网络，或者称为 Self-Organizing Feature Map (SOM)网络，它是由芬兰学者 Teuvo Kohonen 于 1981 年提出的。该网络是一个由全连接的神经元阵列组成的无教师自组织、自学习网络。Kohonen 认为，处于空间中不同区域的神经元有不同的分工，当一个神经网络接收外界输入模式时，将会分为不同的反应区域，各区域对输入模式具有不同的响应特性。

5.1.1 特征映射网络的模型

特征映射网络结构如图 5-1 所示。

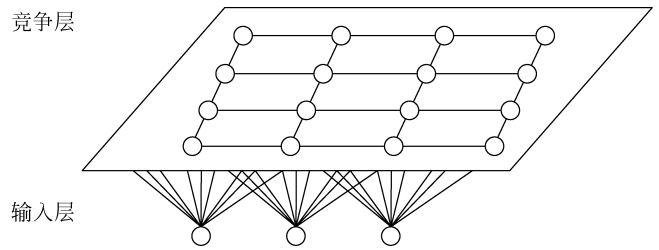

图 5-1　特征映射网络结构

特征映射网络的一个典型特性就是可以在一维或二维的处理单元阵列上，形成输入信号的特征拓扑分布，因此特征映射网络具有抽取输入信号模式特征的能力。特征映射网络一般只包含有一维阵列和二维阵列，但也可以推广到多维处理单元阵列中去。下面只讨论应用较多的二维阵列。特征映射网络模型由以下 4 部分组成。

(1) 处理单元阵列。用于接收事件输入，并且形成对这些信号的“判别函数”。

(2) 比较选择机制。用于比较“判别函数”，并选择一个具有最大函数输出值的处理单元。

(3) 局部互连作用。用于同时激励被选择的处理单元及其最邻近的处理单元。

(4) 自适应过程。用于修正被激励的处理单元的参数，以增加其对应于特定输入“判别函数”的输出值。

假定网络输入为 $\boldsymbol{X}\in\mathbf{R}^n$，输出神经元 i 与输入单元的连接权值为 $\boldsymbol{W}_i\in\mathbf{R}^n$，则输出神经元 i 的输出 o_i 为：

$$o_i=\boldsymbol{W}_i\boldsymbol{X} \tag{5-1}$$

对网络实际具有响应的输出单元 k，该神经元的确定是通过“赢者通吃”的竞争机制得到的，其输出为：

$$o_k=\max_i\{o_i\} \tag{5-2}$$

以上两式可修正为：

$$o_i=\sigma\left(\varphi_i+\sum_{t\in S_i}r_k o_t\right),\quad \varphi_i=\sum_{j=1}^{m}w_{ij}x_j,\quad o_k=\max_i\{o_i\}-\varepsilon$$

其中，w_{ij} 为输出神经元 i 和输入神经元 j 之间的连接权值。x_j 为输入神经元 j 的输出。$\sigma(t)$为非线性函数，即

$$\sigma(t)=\begin{cases}0 & t<0\\ \sigma(t) & 0\leqslant t\leqslant A\\ A & t>A\end{cases} \tag{5-3}$$

ε 为一个很小的正数，r_k 为系数，它与权值及横向连接有关。S_i 为与处理单元 i 相关的处理单元集合，o_k 称为浮动阈值函数。

5.1.2　自组织特征映射网络的学习

特征映射的学习算法过程如下。

(1) 初始化。对 N 个输入神经元到输出神经元的连接权值赋予较小的值。选取输出神经元 j 个“邻接神经元”的集合 S_j。其中，$S_j(0)$表示时刻 $t=0$ 的神经元 j 的“邻接神经元”的集合，$S_j(t)$表示时刻 t 的“邻接神经元”的集合。区域 $S_j(t)$随着时间的增长而不断缩小。

(2) 提供新的输入模式 $\boldsymbol{X}$。

(3) 计算欧氏距离 d_j，即输入样本与每个输出神经元 j 之间的距离：

$$d_j = \| \boldsymbol{X} - \boldsymbol{W}_j \| = \sqrt{\sum_{i=1}^{N} [x_i(t) - w_{ij}(t)]^2} \tag{5-4}$$

并计算出一个具有最小距离的神经元 j^*，即确定出某个单元 k，使得对于任意的 j，都有 $d_k = \min_j(d_j)$。

(4) 给出一个周围的领域 $S_k(t)$。

(5) 按照下式修正输出神经元 j^* 及其“邻接神经元”的权值：

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)[x_i(t) - w_{ij}(t)]$$

其中，η 为一个增益项，并随时间变化逐渐下降到零，一般取

$$\eta(t) = \frac{1}{t} \text{ 或 } \eta(t) = 0.2\left(1 - \frac{t}{10000}\right)$$

(6) 计算输出 o_k：

$$o_k = f(\min_j \| \boldsymbol{X} - \boldsymbol{W}_j \|)$$

其中，$f(\cdot)$一般为 0-1 函数或其他非线性函数。

(7) 提供新的学习样本来重复上述学习过程。

5.1.3　特征映射网络的人口分类

【例 5-1】 人口分类是人口统计中的一个重要指标。由于各方面的原因，我国人口的出生率在性别上的差异比较大，具体表现在同一个时期出生的人口中，一般男的占多数，大大超过了正常的比例。因此，正确地进行人口分类是制定合理的人口政策的基础。

1. 样本设计

通过分析历史资料，得到了在 1999 年 12 月共 20 个地区的人口出生比例情况，如表 5-1 所示。

表 5-1　人口出生比例

男/%	0.5512	0.5123	0.5087	0.5001	0.6012	0.5298	0.5000	0.4965	0.5103	0.5003
女/%	0.4488	0.4877	0.4913	0.4999	0.3988	0.4702	0.5000	0.5035	0.4897	0.4997

将表 5-1 中的数据作为网络的输入样本 P。P 是一个二维随机向量，它的分布情况如图 5-2 所示。

```
P = [0.5512  0.5123  0.5087  0.5001  0.6012  0.5298  0.5000  0.4965  0.5103  0.5003;
     0.4488  0.4877  0.4913  0.4999  0.3988  0.4702  0.5000  0.5035  0.4897  0.4997];
plot(P(1,:),P(2,:),'*r');
hold on
```

2. 网络创建

利用 12 个神经元的特征映射网络对输入向量 P 进行分类。该网络竞争层神经元的组织结构为 3×4，通过距离函数 linkdist 来计算距离。网络创建代码如下。

```
net = newsom([0 1;0 1],[3 4]);
w1_init = net.iw{1,1};
plotsom(w1_init,net.layers{1}.distances);
```

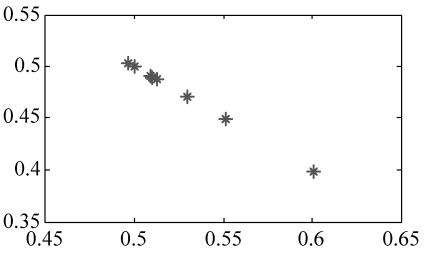

图 5-2　样本数据的分布

运行结果如图 5-3 所示，图中每一点表示一个神经元，由于网络的初始权值都被设置为 0.5，所以这些点在图中是重合的，看起来就像一个点，实际上是 12 个点。

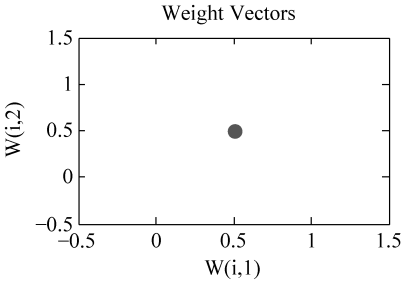

图 5-3　网络初始权值的分布

在命令窗口中查看 w1_init 的值，可得：

```
w1_init =
    0.5000    0.5000
    0.5000    0.5000
    0.5000    0.5000
    0.5000    0.5000
    0.5000    0.5000
    0.5000    0.5000
    0.5000    0.5000
    0.5000    0.5000
    0.5000    0.5000
    0.5000    0.5000
    0.5000    0.5000
    0.5000    0.5000
```

3. 网络训练与测试

接下来利用训练函数 train 对网络进行训练，设想经过训练的网络可对输入向量进行正确分类。网络训练步数对于网络性能的影响比较大，所以这里将步数设置为 100、300 和 500，并分别观察其权值分布。

步数为 100 时的权值分布如图 5-4 所示。

```
%训练步数为100时的训练代码
net = train(net,P);
figure;
w1 = net.iw{1,1};
plotsom(w1,net.layers{1}.distances)
```

步数为 300 时的权值分布如图 5-5 所示。

```
%训练步数为300时的训练代码
net.trainParam.epochs = 300;
net = init(net);
net = train(net,P);
figure;
w1 = net.iw{1,1};
plotsom(w1,net.layers{1}.distances)
```

步数为 500 时的权值分布如图 5-6 所示。

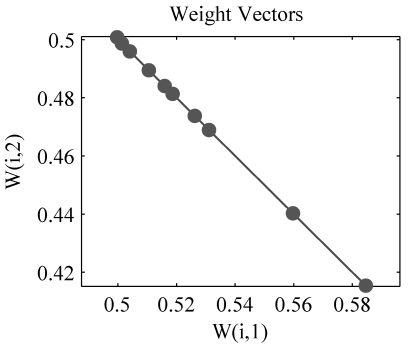

图 5-4　权值分布(训练步数为 100)

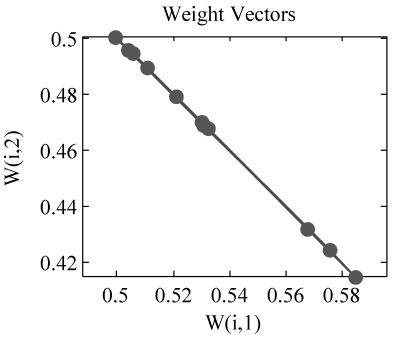

图 5-5　权值分布(训练步数为 300)

```
% 训练步数为 500 时的训练代码
net.trainParam.epochs = 500;
net = init(net);
net = train(net,P);
figure;
w1 = net.iw{1,1};
plotsom(w1,net.layers{1}.distances)
```

从图 5-4～图 5-6 可以看出,训练了 100 步以后,神经元就开始自组织地分布了,每个神经元可以区分不同的样本。随着训练步数的增加,神经元的分布更加合理,但是,当训练次数达到一定值后,权值分布的改变就不明显了。比如,训练 300 步和训练 500 步后的权值分布就比较相似。

网络训练结束后,权值也就固定了。以后每输入一个值,网络就会自动地对其进行分类。因此,可利用这一点对网络进行测试。首先,利用仿真函数 sim 来观察网络对样本数据的分类结果。

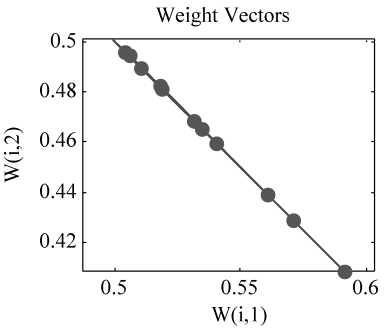

图 5-6　权值分布(训练步数为 500)

```
y = sim(net,P);
Y = vec2ind(y)
```

输出结果为:

```
Y =
     4    10    10    12     1     6    12    12    10    12
```

对结果进行分析,如表 5-2 所示。

表 5-2　聚类结果

样 本 序 号	类　　别	激发神经元的索引
1	1	4
2	2	3
3	3	10
4、7、10	4	11
5	5	1

续表

样 本 序 号	类　　别	激发神经元的索引
6	6	7
8	7	12
9	8	6

现在，输入一个某地的出生性别比例，检验它属于哪一类。

```
p = [0.5;0.5];
y = sim(net,p);
y = vec2ind(y)
```

结果为 11。由此可见，此时激发了网络的第 11 个神经元，所以 p 属于第 4 类。通过直接对比数据可知，p 确定与样本中的第 4 组、第 7 组和第 10 组数据非常接近。

5.2　竞争型神经网络

竞争型神经网络是基于无教师学习方法的神经网络的一种重要类型，它经常作为基本的网络形式，构成其他一些具有自组织能力的网络，如自组织映射网络、自适应共振理论网络、学习向量量化网络等。

生物神经网络存在一种侧抑制现象，即一个神经细胞兴奋后，通过它的分支会对周围其他神经细胞产生抑制，这种抑制使神经细胞之间出现竞争：在开始阶段，各神经元对相同的输出具有相同的响应机会，但产生的兴奋程度不同，其中兴奋最强的一个神经细胞对周围神经细胞的抑制作用也最强，从而使其他神经元的兴奋程度得到最大强度的抑制，而兴奋程度最强的神经细胞却“战胜”了其他神经元的抑制作用脱颖而出，成为竞争的胜利者，并因为获胜其兴奋的程度得到进一步加强，正所谓“胜者为王，败者为寇”。竞争型神经网络在学习算法上，模拟了生物神经网络中神经元之间的兴奋、抑制与竞争的机制，进行网络的学习与训练。

5.2.1　竞争型神经网络模型

竞争型神经网络模型如图 5-7 所示。

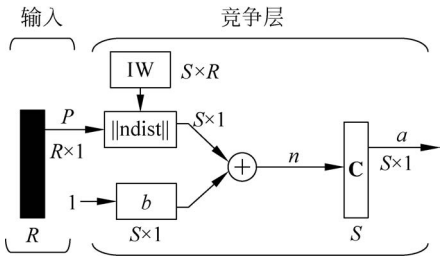

图 5-7　竞争型神经网络模型

可以看出竞争型神经网络为单层网络。‖ ndist ‖的输入为输入向量 $\boldsymbol{R}$ 和输入权值向量 IW，其输出为 $S\times1$ 的列向量，列向量中的每个元素为输入向量 $\boldsymbol{R}$ 和输入权值向量 IW 距离的负数(negative)，在神经网络工具箱中以距离函数 negdist 进行计算。

n 为竞争层传输函数的输入，其值为输入向量 $\boldsymbol{R}$ 和输入权值向量 IW 距离的负数与阈值 b 之和。如果所有的阈值向量为 0，则当输入向量 $\boldsymbol{R}$ 和输入权值向量 IW 相等时，n 为最大值 0。

对于 n 中最大的元素，竞争层传输函数输出 1(即竞争的“获胜者”输出为 1)，而其他元素均输出 0。如果所有的阈值向量为 0，则当神经元的权值向量接近输入向量时，它在 n 中各元素中的负值最小，而值最大，从而赢得竞争，对应的输出为 1。在 MATLAB 工具箱中，创建竞争型神经网络的函数是 newc。

5.2.2　竞争型神经网络的学习

1. Kohonen 权值学习规则

竞争型神经网络按 Kohonen 学习规则对获胜神经元的权值进行调整。假若第 i 个神经元获胜，则输入权值向量的第 i 行元素（即获胜神经元的各连接权）按下式进行调整：

$$_iIW(k) = {}_iIW(k-1) + \alpha[p(k) - {}_iIW(k-1)] \tag{5-5}$$

而其他神经元的权值不变。

Kohonen 学习规则通过输入向量进行神经元权值的调整，因此在模式识别的应用中是很有用的。通过学习，那些最靠近输入向量的神经元的权值向量得到修正，使之更靠近输入向量，其结果是获胜的神经元在一次相似的输入向量出现时，获胜的可能性会更大；而对于那些与输入向量相差很远的神经元权值向量，获胜的可能性将变得很小。这样，当经过越来越多的训练样本学习后，每一个网络层中的神经元权值向量很快被调整为最接近某一类输入向量的值。最终的结果是，如果神经元的数量足够多，则具有相似输入向量的各类模式作为输入向量时，其对应的神经元输出为 1；而对于其他模式的输入向量，其对应的神经元输出为 0。所以，竞争型网络具有对输入向量进行学习分类的能力。在 MATLAB 工具箱中，learnk 函数用于实现 Kohonen 学习规则。

2. 阈值学习规则

竞争型神经网络的一个局限性是，某些神经元可能永远也派不上用场，换句话说，某些神经元的权值向量从一开始就远离所有的输入向量，从而使得该神经元不管进行多长的训练，也永远不会赢得竞争。这些神经元称为“死神经元”，它们实现不了任何有用的函数映射。

为了避免这一现象的发生，对那些很少获胜（甚至从来不曾获胜）的神经元赋以较大的阈值，而对那些经常获胜的神经元赋以较小的阈值。正的阈值与距离的负值相加，使获胜很少的神经元竞争层传输函数的输入就像获胜的神经元一样。这一过程就像人们“同情”弱者一样，表现出一个人的“良心”。

这一过程的实现，需要用到神经元输出向量的平均值，它等价于每个神经元输出为 1 的百分比，显然，经常获胜的神经元，其输出为 1 的百分比较大。在 MATLAB 工具箱中，learncon 函数用于进行阈值的修正。

对学习函数 learncon 进行阈值修正时，神经元输出向量的平均值越大，其“良心”值越大，所以凭“良心”获得的阈值就越小，而让那些不经常获胜的神经元的阈值逐渐变大。其算法如下：

$$c(k) = (1 - \mathrm{lr}) \times c(k-1) + \mathrm{lr} \times a(k-1) \tag{5-6}$$

$$b(k) = \exp[1 - \log(c(k))] - b(k-1) \tag{5-7}$$

式中，c 为“良心”值；a 为神经元输出的平均值；lr 为学习率。

一般将 learncon 的学习率设置成默认值或比 learnk 的学习率小的值，使其在运行过程中能够较精确地计算神经元的输出平均值。

结果那些不经常产生响应的神经元的阈值相对于那些经常产生响应的神经元，其阈值不断增大，使其产生响应的输入空间也逐渐增大，即对更多的输入向量产生响应，最终各神经元对输入向量产生响应的数目大致相等。

这样做有以下两点好处。

（1）如果某个神经元因为远离所有的输入向量而始终不能在竞争中获胜，则其阈值会变得越来越大，使其终究可以获胜。当这一情况出现后，它将逐渐向输入向量的某一类聚集，一

旦神经元的权值靠近输入向量的某一类模式，该神经元将经常获胜，其阈值将逐渐减小到0，这样就解决了“死神经元”的问题。

(2) 学习函数 learncon 强迫每个神经元对输入向量的分类百分比大致相同，所以如果输入空间的某个区域比另外一个区域聚集了更多的输入向量，那么输入向量密度大致的区域将吸引更多的神经元，从而获得更细的分类。

5.2.3 竞争型神经网络存在的问题

对于模式样本本身具有较明显的分类特征，竞争型神经网络可以对其进行正确的分类，网络对同一类或相似的输入模式具有较稳定的输出响应。但也存在以下一些问题。

(1) 当学习模式样本本身杂乱无章，没有明显的分类特征时，网络对输入模式的响应呈现振荡的现象，即对同一类输入模式的响应可能激活不同的输出神经元，从而不能实现正确的分类。当各类模式的特征相近时，也会出现同样的情况。

(2) 在权值和阈值的调整过程中，学习率的选择在收敛速度和稳定性之间存在矛盾，而不像前面介绍的其他学习算法，可以在刚开始时采用较大的学习率，而在权值和阈值趋于稳定时，采用较小的学习率。竞争型神经网络在增加新的学习样本时，对权值和阈值可能需要做比前一次更大的调整。

(3) 网络的分类性能与权值和阈值的初始值、学习率、训练样本的顺序、训练时间的长短(训练次数)等都有关系，而目前还没找到有效的方法对各种因素的影响进行评判。

(4) 在 MATLAB 神经网络工具箱中，以函数 trainr 进行竞争型神经网络的训练，用户只能限定训练的最长时间或训练的最大次数，以此终止训练，但终止训练时网络的分类性能究竟如何，没有明确的评判指标。

5.2.4 竞争型神经网络的 MATLAB 实现

竞争型神经网络适用于具有明显分类特征的模式分类。其 MATLAB 仿真程序设计主要包括：

(1) 创建竞争型神经网络。首先根据给定的问题确定训练样本的输入向量，当不足以区分各类模式时，应想办法增加特征值；其次根据模式分类数确定神经元的数目。

(2) 训练网络。训练最大次数的默认值为100，当训练结果不能满足分类的要求时，可尝试增加训练的最大次数。

(3) 以测试样本进行仿真。

【例 5-2】 利用竞争层网络对样本数据进行分类。

本例中待分类的样本数据由 nngenc 函数随机产生，即

```
P = nngenc(rand,class,num,std);
```

其中，参数 class 表示样本数据的类别个数。然后利用 newc 函数建立竞争层网络：

```
net = newc(range,class,klr,clr);
```

其中，class 是数据类别个数，也是竞争层神经元的个数；klr 和 clr 分别是网络的权值学习率和阈值学习率。竞争层网络在训练时不需要目标输出，网络通过对数据分布特性的学习，自动地将数据划分为指定类别。网络的训练语句如下(其中，默认的训练函数为 trainr)：

```
net = train(net,P);
```

在对训练好的网络进行仿真时，网络的输出为单值矢量组，为了观察方便，一般要将单值矢量组转化为下列矩阵的形式：

```
Y = sim(net,P);
Y1 = vec2ind(Y);
```

本例完整的 MATLAB 程序如下：

```
% 产生样本数据 P,P 中包括三类共 30 个二维矢量
range = [ - 1 1; - 1 1];
class = 3;
num = 10;
std = 0.1;
P = nngenc(range,class,num,std);
% 画出样本数据分布图
plot(P(1,:),P(2,:),' * ','markersize',6);axis([ - 1.5 1.5  - 1.5 1.5]);
% 建立竞争层网络
klr = 0.1;
clr = 0.01;
net = newc(range,class,klr,clr);
% 对网络进行训练
net.trainParam.epochs = 5;
net = train(net,P);
% 画出竞争层神经元权值
w = net.iw{1};
hold on;
plot(w(:,1),w(:,2),'ob');
title('Input data & Weights');
% 利用原始样本数据对网络进行仿真
Y = sim(net,P);
Y1 = vec2ind(Y)
% 用不同符号标注数据分类结果
figure;
for i = 1:30
    if Y1(i) == 1
        plot(P(1,i),P(2,i),' * ','markersize',6);
    elseif Y1(i) == 2
        plot(P(1,i),P(2,i),' + ','markersize',6);
    else
        plot(P(1,i),P(2,i),'x','markersize',6);
    end
    hold on;
end
axis([ - 1.5 1.5  - 1.5 1.5]);
title('class 1: *   class 2: +   class 3:x');
% 利用一组新的输入数据检验网络性能
p = [ - 0.4  - 0.4; - 0.1 0.9];
y = sim(net,p);
y1 = vec2ind(y)
```

窗口显示结果为：

```
TRAINR, Epoch 0/5
TRAINR, Epoch 5/5
```

```
TRAINR, Maximum epoch reached.
y1 =
     1     2
```

程序运行结果如图 5-8 和图 5-9 所示。在图 5-8 中，待分类的样本数据用星号标注，网络训练完毕后的竞争层神经元权值用圆圈标注。在图 5-9 中，已经划分好的三类数据分别用星号、加号和“×”符号标注。

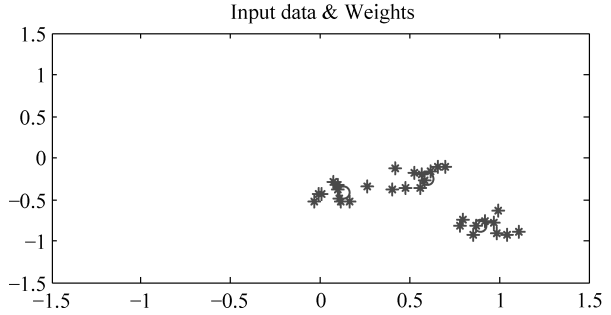

图 5-8　待分类的样本数据和权值

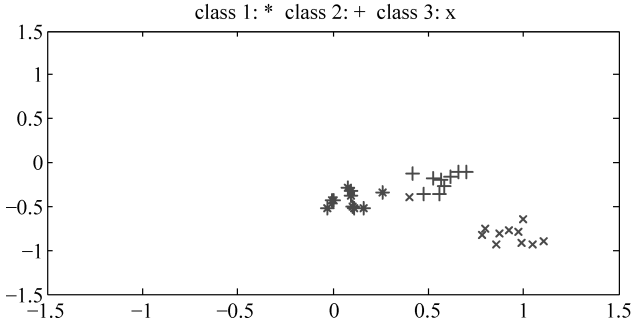

图 5-9　网络分类结果

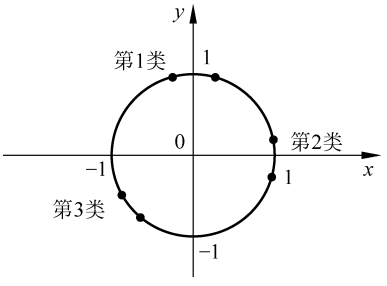

图 5-10　待分类模式

【例 5-3】 以竞争型神经网络完成如图 5-10 所示的三类模式的分类。

$$\boldsymbol{p}_1=\begin{bmatrix}-0.1961\\0.9806\end{bmatrix},\quad \boldsymbol{p}_2=\begin{bmatrix}0.1961\\0.9806\end{bmatrix},\quad \boldsymbol{p}_3=\begin{bmatrix}0.9806\\0.1961\end{bmatrix}$$

$$\boldsymbol{p}_4=\begin{bmatrix}0.9806\\-0.1961\end{bmatrix},\quad \boldsymbol{p}_5=\begin{bmatrix}-0.5812\\-0.8137\end{bmatrix},\quad \boldsymbol{p}_6=\begin{bmatrix}-0.8137\\-0.5812\end{bmatrix}$$

图中的三类模式从它们在二维平面上的位置特征看，具有明显的分类特征，可以以竞争型神经网络完成其分类。

本例的 MATLAB 程序如下。

```
clear all
%定义输入向量
P=[-0.1961 0.1961 0.9806 0.9806 -0.5812 -0.8137;
    0.9806 0.9806 0.1961 -0.1961 -0.8137 -0.5812];
%创建竞争型网络
net=newc([-1 1;-1 1],3);
%训练神经网络
net=train(net,P);
```

```
%定义待测试样本输入向量
p=[-0.1961 0.1961 0.9806 0.9806 -0.5812 -0.8137;
    0.9806 0.9806 0.1961 -0.1961 -0.8137 -0.5812];
%网络仿真
y=sim(net,p);
%输出仿真结果
yc=vec2ind(y)
```

仿真结果为：

```
yc =
     2     2     1     1     3     3
```

结果很好地完成了分类。

5.3　自适应共振理论

自适应共振理论英文全称为 Adaptive Resonance Theory，简称为 ART，它是由 S. Grossberg 和 A. Carpentent 等人于 1986 年提出的。Grossberg 的研究工作主要是采用数学方法描述人的心理和认知活动，致力于为人类的心理和认知活动建立一个统一的数学模型。以其思想基础提出的 ART 模型成功地解决了神经网络学习中的稳定性（固定某一分类集）和可塑性（调整网络固有参数的学习状态）的关系问题。

ART 是以认知和行为模式为基础的一种无教师、向量聚类和竞争学习的算法。在数学上，ART 为非线性微分方程的形式；在网络结构上，ART 是全反馈结构，且各层节点各有不同的性质。

ART 网络共有 3 种类型；ART-1、ART-2 和 ART-3。这里主要介绍 ART-1 型网络。

5.3.1　自适应共振理论模型

ART-1 型网络结构如图 5-11 所示。由图可见，网络分为输入和输出两层，一般根据各层所有的功能特征称输入层为比较层，输出层为识别层。和其他阶层型网络的显著区别是，ATR-1 型网络不仅具有从输入层到输出层的前馈连接权，还有从输出层到输入层的反馈连接权。

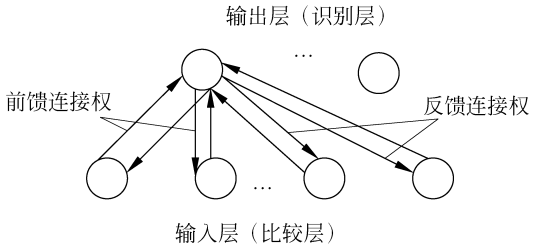

图 5-11　ART-1 型网络结构

假定网络输入层有 N 个神经元，输出层有 M 个神经元，二值输入模式和输出向量分别为 $\boldsymbol{A}_k=(a_1^k,a_2^k,\cdots,a_N^k)$ 和 $\boldsymbol{B}_k=(b_1^k,b_2^k,\cdots,b_M^k)$，其中 $k=1,2,\cdots,p$，p 为输入学习模式的数目。前馈连接权和反馈权分别为 w_{ij} 和 t_{ij}，$j=1,2,\cdots,M$。

ART-1 网络的学习及工作过程，是通过反复地将输入学习模式由输入层向输出层自下而上地识别、由输出层向输入层自上而下地比较来实现的。当这种自下而上的识别和自上而下

的比较达到共振，即输入向量可以正确反映输入学习模式的分类，且网络原有记忆没有受到不良影响时，网络对一个输入学习模式的记忆和分类就算完成。网络的学习和工作过程可以分为初始化阶段、识别阶段、比较阶段和探寻阶段，下面将详细介绍网络的学习及工作过程。

5.3.2 自适应共振理论的学习

ART-1 网络的学习及工作可以归纳为如下过程。

(1) 初始化。令 $t_{ij}(0)=1, w_{ij}(0)=\dfrac{1}{N+1}, i=1,2,\cdots,N, j=1,2,\cdots,M$。其中，临界参数 $0<\rho\leqslant 1$。

(2) 将输入模式 $\boldsymbol{A}_k=(a_1^k,a_2^k,\cdots,a_N^k)$ 提供给网络的输入层。

(3) 计算输出层各个神经元的输入加权和。

$$s_j=\sum_{i=1}^{N} w_{ij}a_i^k, \quad j=1,2,\cdots,M$$

(4) 选择输入模式的最佳分类结果：

$$s_g=\max_{j=1,2,\cdots,M} s_j$$

令神经元 g 的输出为 $\boldsymbol{T}$。

(5) 计算以下 3 式，并进行判断：

$$|\boldsymbol{A}_k|=\sum_{i=1}^{N} a_i^k$$

$$|\boldsymbol{T}_g\cdot\boldsymbol{A}_k|=\sum_{i=1}^{N} t_{gi}a_i^k$$

$$\frac{|\boldsymbol{T}_g\cdot\boldsymbol{A}_k|}{|\boldsymbol{A}_k|}>\rho$$

如果最后一式成立，则转入步骤(7)，否则转入步骤(6)。

(6) 取消识别结果，将输出层神经元 g 的输出值复位为 0，并将这一神经元排除在下次识别的范围之外，返回步骤(4)。当所有已利用过的神经元都无法满足步骤(5)中的最后一式时，则选择一个新的神经元作为分类结果，并进入步骤(7)。

(7) 接受识别结果，调整连接权值：

$$w_{ig}(t+1)=\frac{t_{gi}(t)a_i}{0.5+\sum_{i=1}^{N} t_{gi}(t)a_i}$$

$$t_{gi}(t+1)=t_{gi}(t)a_i$$

其中，$i=1,2,\cdots,N$。

(8) 将步骤(6)中复位的所有神经元重新加入识别范围中，返回步骤(2)对下一个模式进行识别。

无论网络学习还是网络回想，都使用以上的规则。只不过在网络回想时，只对那些与未使用过的输出神经元有关的连接权值 w_{ij} 和 t_{ij} 才进行初始化。其他连接权值保持网络学习后的值不变。当输入模式是一个网络已记忆的学习模式时，不需要再进行网络权值的调整，这是因为当输入模式和网络记忆的学习模式完全相等，在按照权值调整公式进行调整时，网络连接权值不会发生任何变化。而当输入模式与网络记忆模式存在一定差异时，按照权值调整公式进行调整，将会影响网络原有模式的记忆效果。但是如果输入的是一个全新的模式，需要利用

网络对其另外记忆时，则必须按照权值调整公式对网络连接权值进行调整。

尽管 ART-1 网络具有许多其他网络所没有的优点，但是它仅以输出层中某个神经元代表分类结果，而不是像 Hopfield 网络那样，把分类结果分散在各个神经元上来表示。所以，一旦输出层中某个输出神经元损坏，则会导致该神经元所代表类别的模式信息全部消失。这是 ART-1 网络一个很大的缺陷。

ART-2 型网络与 ART-1 型网络的主要区别是，ART-2 型网络以模拟量作为输入模式，同时在算法上做了一些相应的改进，并采用慢速学习方式，其抗干扰能力大大增强。ART-3 型网络是由多个 ART-1 型网络组成的复合阶层型网络。

5.3.3　自适应共振理论的 MATLAB 实现

MATLAB 神经网络工具箱没有为 ART 型网络提供专门的函数，因此，利用现有的神经网络工具箱是无法实现 ART-1 网络的。但是，我们可以借助于 MATLAB 强大的数学计算功能来实现 ART-1 网络的训练和联想记忆功能。

【例 5-4】 现举一个简单的例子来演示利用 MATLAB 实现 ART-1 网络的过程。如图 5-12 所示，设 ART-1 网络有 5 个输入神经元和 20 个输出神经元。现有两组输入模式 $\boldsymbol{A}_1=(1,1,0,0,0)$ 和 $\boldsymbol{A}_2=(1,0,0,0,1)$，要求利用这两组模式来训练网络。根据 5.3.2 节中的训练过程，该网络的训练步骤分为下面几步。

(1) 初始化。令 $w_{ij}=1/n+1=1/6$，$t_{ji}=1$，其中 $i=1,2,\cdots,5$，$j=1,2,\cdots,20$，令 $\rho=0.8$。

(2) 将输入模式 $\boldsymbol{A}_1$ 提供给网络的输入层。

(3) 求获胜的神经元。因为在网络的初始状态下，所有的前馈连接权 w_{ij} 均取相等的权值 1/6，所以各输入神经元均具有相同的输入加权和 s_j。这时可取任一个神经元作为 $\boldsymbol{A}_1$ 的分类代表，如第 1 个，令其输出值为 1。

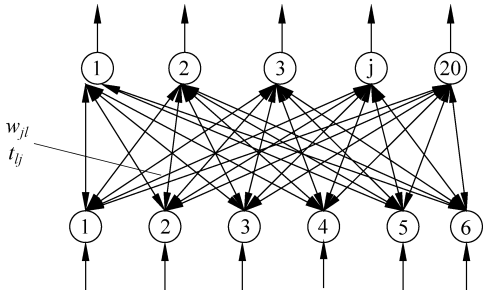

图 5-12　ART-1 网络实例

(4) 计算下式

$$|\boldsymbol{A}_1|=\sum_{i=1}^{5}a_i=2,\quad |\boldsymbol{T}_1\boldsymbol{A}_1|=\sum_{i=1}^{5}t_{1i}a_i=2$$

(5) 计算 $\dfrac{|\boldsymbol{T}_1\boldsymbol{A}_1|}{|\boldsymbol{A}_1|}=1>0.8$，接受这次识别结果。

(6) 调整权值。

$$\boldsymbol{W}_1=(w_{11},w_{12},w_{13},w_{14},w_{15})=(0.4,0.4,0,0,0)$$

$$\boldsymbol{T}_1=(t_{11},t_{21},t_{31},t_{41},t_{51})=(1,1,0,0,0)$$

至此，$\boldsymbol{A}_1$ 已经被记忆在网络中了。

(7) 将输入模式 $\boldsymbol{A}_2$ 提供给网络的输入层。

(8) 求获胜神经元,$s_1=0.4$,$s_2=1/6$,$s_3=\cdots=s_{20}=1/6$,由于 $s_1>s_2=s_3=\cdots=s_{20}$,所以取神经元 1 作为获胜神经元,但这显然与 $\boldsymbol{A}_1$ 的识别结果相矛盾。又因为

$$\frac{|\boldsymbol{T}_2\boldsymbol{A}_2|}{|\boldsymbol{A}_2|}=\frac{1}{2}<0.8$$

所以拒绝这次识别结果,重新进行识别。由于 $s_2=s_3=\cdots=s_{20}=1/6$,故可从中任选一个神经元作为 $\boldsymbol{A}_2$ 的分类结果,如神经元 20。

(9) 调整权值。

$$\boldsymbol{W}_2=(w_{21},w_{22},w_{23},w_{24},w_{25})=(0.4,0,0,0,0.4)$$

$$\boldsymbol{T}_2=(t_{12},t_{22},t_{32},t_{42},t_{52})=(1,0,0,0,1)$$

至此,$\boldsymbol{A}_2$ 也记忆在网络中了。

按照上述步骤,可以编写以下 MATLAB 代码。

```
%竞争层的输出
xiu = rands(20);
%正向权值 W 和反向权值 T
W = rands(20,5);
T = rands(20,5);
%警戒参数
xiuxiu = 0.8;
%两组模式 A1 和 A2
A1 = [1 1 0 0 0];
A2 = [1 0 0 0 1];
%初始化
for i = 1:20
    for j = 1:5
        W(i,j) = 1/6;
        T(i,j) = 1;
    end
end
%判定是否接受识别结果
normalA1 = norm(A1,1);
normalTA1 = T(1,:) * A1';
count = 1;
if normalTA1/normalA1 > xiuxiu
    xiu(count) = 1;
end
%权值调整
W(1,:) = [0.4 0.4 0 0 0];
T(1,:) = [1 1 0 0 0];
%寻找可以记忆 A2 的神经元
for k = 1:20
    s(k) = W(k,:) * A2';
    if s(k) == max(s)
        count = k;
    end
end
%如果和 A1 的神经元重复,继续寻找
if xiu(count) == 1
    newcount = count + 1
end
for i = 1:(count - 1)
```

```
    p(i) = s(i);
end
for i = count:19
    p(i) = s(i + 1);
end
for k = newcount:20
    if s(k) == max(p)
        count = k;
    end
end
% 确定找到的神经元序号 count,并令其对应的输出为 1
xiu(count) = 1;
% 权值调整
W(count,:) = [0.4,0,0,0,0.4];
T(count,:) = [1,0,0,0,1];
xiu'
```

运行结果为：

```
xiu' =
1.0000    0.6375   - 0.1397    0.7806    0.4698    0.3746   - 0.3078   - 0.6679   - 0.6888
- 0.6178   - 0.1551   0.7120   - 0.0195    0.6319   - 0.0785   - 0.0853   - 0.0986
- 0.1756    0.8032    1.0000
```

可见，第 1 个和第 20 个神经元的输出均为 1，表示它们记忆了输入模式。

5.4　学习矢量量化的神经网络

5.4.1　矢量量化的神经网络模型

对于学习矢量量化(Learning Vector Quantization，LVQ)神经网络，因为用户指定了目标分类结果，所以网络可以通过监督学习完成对输入矢量模式的准确分类。LVQ 神经网络模型如图 5-13 所示。

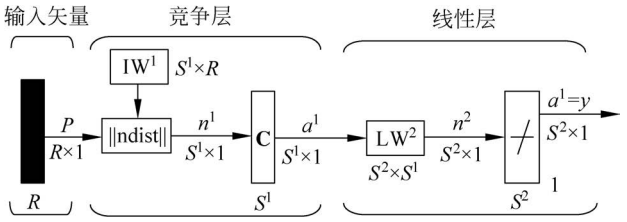

图 5-13　LVQ 神经网络模型

LVQ 神经网络有两个网络层，即竞争层和线性层。竞争层对输入矢量的学习分类与前面所阐述的竞争层一样，我们把竞争层的分类称为子分类；线性层根据用户的要求将竞争层的分类结果映射到目标分类结果中，我们把线性层的分类称为目标分类。

竞争层和线性层的每一个神经元的输出都对应一个分类(子分类或目标分类)结果，所以竞争层通过学习，可以得到 S^1 类子分类结果；然后，线性层将 S^1 类子分类结果再分成 S^2 类目标分类结果(S^1 始终大于 S^2)。例如，假设竞争层的第 1～3 个神经元对输入空间的子分类所对应的线性层的目标分类为第 2 类，则竞争层的第 1～3 个神经元与线性层的第 2 个神经元的连接权将全部为 1，而与其他线性层神经元的连接权全部为 0，这样，当竞争层的第 1～3 个

神经元中的任意一个神经元在竞争中获胜时，线性层的第 2 个神经元将输出 1。在 MATLAB 神经网络工具箱中，创建 LVQ 网络的函数为 newlvq。

5.4.2 矢量量化的神经网络学习

矢量量化(LVQ1)神经网络的学习与其他有导师学习方法一样，其训练样本集的输入向量和目标向量是成对出现的，即

$$\{\boldsymbol{p}_1,\boldsymbol{t}_1\},\{\boldsymbol{p}_2,\boldsymbol{t}_2\},\cdots,\{\boldsymbol{p}_Q,\boldsymbol{t}_Q\}$$

每个目标向量，除了有一个元素为 1 以外，其余元素均为 0，目标向量中元素为 1 的行即为相应的输入矢量模式。例如，对具有 3 个输入元素、4 个输出模式的 LVQ 网络，若

$$\left\{\boldsymbol{p}_1=\begin{bmatrix}1\\-2\\0\end{bmatrix},\boldsymbol{t}_1=\begin{bmatrix}0\\1\\0\\0\end{bmatrix}\right\}$$

则表示第 1 个训练样本对应于第 2 个模式，LVQ 网络输出层的第 2 个神经元输出 1。

LVQ1 网络进行训练时，对每一个输入矢量 $\boldsymbol{p}$，先以函数 ndist 计算它与输入权值向量 IW^1 每一行元素的距离，使隐层神经元进行竞争。假设 n^1 的第 i 个元素值最大，则竞争层的第 i 个神经元将赢得竞争，这使得竞争层的输出 a^1 的第 i 个元素值为 1，而其余元素值为 0。

当 a^1 与第二网络层的权值 LW^2 相乘时，在 a^1 中唯一的一个元素值为 1 的元素被认为是与输入矢量对应的第 k 个分类模式，所以网络认为输入矢量 $\boldsymbol{p}$ 为 k 个分类模式，则 a^2 的第 k 个元素输出 1。当然，该分类结果可能正确，也可能不正确，因为 t_k 可能为 1，也可能为 0，它取决于输入矢量模式是否属于第 k 个分类模式。

可以根据目标矢量，调整 IW^1 的第 i 行，当分类结果正确时，使该行元素的值向输入矢量 $\boldsymbol{p}$ 靠拢；当分类结果错误时，使该行元素的值远离输入矢量 $\boldsymbol{p}$。当输入矢量 $\boldsymbol{p}$ 得到正确的分类时，

$$a_k^2=t_k=1 \tag{5-8}$$

IW^1 的第 i 行可以按下式进行修正：

$${}_i\mathrm{IW}^1(q)={}_i\mathrm{IW}^1(q-1)+\alpha[p(q)-{}_i\mathrm{IW}^1(q-1)] \tag{5-9}$$

当输入矢量 $\boldsymbol{p}$ 得到错误的分类时，

$$a_k^2=1,\quad t_k=0,\quad a_k^2\neq t_k \tag{5-10}$$

IW^1 的第 i 行可以按下式进行修正：

$${}_i\mathrm{IW}^1(q)={}_i\mathrm{IW}^1(q-1)-\alpha[\boldsymbol{p}(q)-{}_i\mathrm{IW}^1(q-1)] \tag{5-11}$$

在调整 IW^1 的第 i 行时，其他行不变，所以输出误差反向传播到第 1 网络层，对 IW^1 的其他行没有影响。按上述方法进行修正的结果使得隐层的神经元趋近落入相应输入模式的输入矢量，远离其他模式的输入矢量，以构成其子分类。

所谓矢量量化，就是将矢量邻近的区域看作同一量化等级，用其中心值表示，即用少量的聚类中心表示原始数据。SOFM 和 LVQ 都具有矢量量化作用，不同的是，SOFM 的各中心(输出阵列中的神经元)的排列是有结构性的，即各相邻中心点对应的输入数据中的某种特征是相似的，而 LVQ 的中心没有这种排序功能。在 MATLAB 神经网络工具箱中，LVQ1 神经网络的第 1 网络层权值调整的学习函数是 learnlv1。

5.4.3 LVQ1 学习算法的改进

LVQ1 学习算法的改进(LVQ2)是在 LVQ1 的基础上进行的，它可以改善 LVQ1 学习结

果的性能。在 MATLAB 神经网络工具箱中，LVQ2 神经网络的第 1 网络层权值调整的学习函数是 learnlv2。

LVQ2 的学习过程与 LVQ1 类似，在应用 LVQ1 进行学习后，再用 LVQ2 进行学习，不同的是，LVQ2 是针对最接近输入矢量的两个相邻神经元的权值进行的，其中一个神经元对应正确的分类模式，另一个神经元对应错误的分类模式，而输入向量位于定义的窗口时，

$$\min\left(\frac{d_i}{d_j},\frac{d_j}{d_i}\right)>s,\quad s=\frac{1-w}{1+w} \tag{5-12}$$

式中，d_i，d_j 分别表示输入矢量 $\boldsymbol{p}$ 与 ${}_i\mathrm{IW}^1$，${}_j\mathrm{IW}^1$ 的欧几里得距离，w 的取值范围为 0.2～0.3。例如，当 w 取值为 0.25 时，$s=0.6$，那么，当 d_i 和 d_j 两个距离之比大于 0.6 时，则对 IW_i^1，IW_j^1 都需进行调整。

当第 i 个神经元对应的输出分类模式错误时，IW^1 的第 i 行可以按下式进行修正：

$${}_i\mathrm{IW}^1(q)={}_i\mathrm{IW}^1(q-1)+\alpha[\boldsymbol{p}(q)-{}_i\mathrm{IW}^1(q-1)] \tag{5-13}$$

当第 j 个神经元对应的输出分类模式正确时，IW^1 的第 j 行可以按下式进行修正：

$${}_j\mathrm{IW}^1(q)={}_j\mathrm{IW}^1(q-1)+\alpha[\boldsymbol{p}(q)-{}_j\mathrm{IW}^1(q-1)] \tag{5-14}$$

这样，如果给定两个很相近的输入矢量，其中一个对应正确的分类，而另一个对应错误的分类，则 LVQ2 也能对靠得非常近，甚至对刚刚可分的模式进行正确的分类，从而提高子分类结果的鲁棒性。

5.4.4　LVQ 神经网络的 MATLAB 实现

【例 5-5】 画出具有 3×2 栅格拓扑结构的 SOFM，并以该神经网络完成对图 5-14 所示输入矢量模式的分类，分别画出当最大训练步长 epochs=40，60，100 时，调整权值后的神经元拓扑结构图。

$$\boldsymbol{p}=\begin{bmatrix}0.1 & 0.3 & 1.2 & 1.1 & 1.8 & 1.7 & 0.1 & 0.3 & 1.2 & 1.1 & 1.8 & 1.7\\ 0.2 & 0.1 & 0.3 & 0.1 & 0.3 & 0.2 & 0.8 & 0.8 & 0.9 & 0.9 & 0.7 & 0.8\end{bmatrix}$$

(1) 画出 3×2 栅格型 SOFM 的特征映射图，如图 5-15 所示。

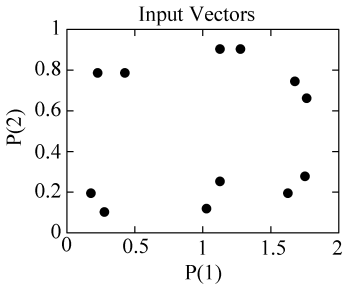

图 5-14　例 5-5 的输入向量

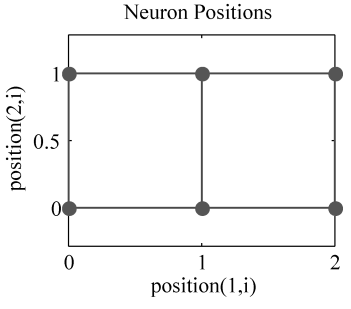

图 5-15　3×2 栅格 SOFM 特征映射图

绘制 3×2 栅格型 SOFM 特征映射图的 MATLAB 程序。

```
pos = gridtop(3,2);
plotsom(pos)
```

(2) 创建和训练 SOFM 神经网络的 MATLAB 程序设计：

```
clear all
 % 创建 SOFM 网络
net = newsom([0 2;0 1],[3 2],'gridtop');
```

```
%定义输入向量
P=[0.1 0.3 1.2 1.1 1.8 1.7 0.1 0.3 1.2 1.1 1.8 1.7
   0.2 0.1 0.3 0.1 0.3 0.2 0.8 0.8 0.9 0.9 0.7 0.8];
%绘制输入矢量
plot(P(1,:),P(2,:),'.g','markersize',18);
%训练 SOFM 网络
%设置训练步长的最大步长
net.trainParam.epochs=100;
net=train(net,P);
%绘制训练后的 SOFM 神经网络特征映射图
hold on;
plotsom(net.iw{1,1},net.layers{1}.distances);
hold off;
```

设置不同的最大步长进行训练，调整权值后的 SOFM 特征映射图如图 5-16 所示。

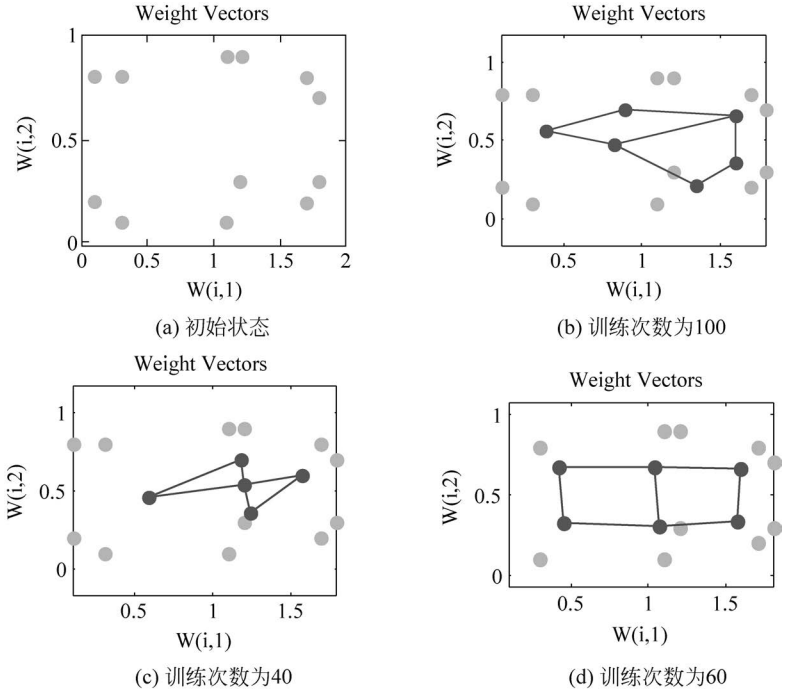

图 5-16　SOFM 神经网络权值的调整过程

SOFM 神经网络的学习，就是使权向量的方向朝着输入模式向量的方向进行调整，使各个权向量分别向各个聚类模式群的中心位置靠拢，同时，使网络权向量几何点的排列与竞争层各神经元的自然排列基本一致(拓扑结构一致)。从图 5-16 权值的调整过程来看，调整的目标是使网络权向量几何点的排列与竞争层各神经元的拓扑结构基本一致，即为 3×2 栅格型结构。

(3) SOFM 神经网络的 MATLAB 仿真程序设计：

```
%定义输入向量
P=[0.1 0.3 1.2 1.1 1.8 1.7 0.1 0.3 1.2 1.1 1.8 1.7
   0.2 0.1 0.3 0.1 0.3 0.2 0.8 0.8 0.9 0.9 0.7 0.8];
%网络仿真
y=sim(net,P);
%输出仿真结果
yc=vec2ind(y)
```

仿真结果为：

```
yc =
     4     4     5     5     6     6     1     1     2     2     3     3
```

因为输入向量模式具有明显的分类特征，所以 SOFM 网络很好地完成了分类。

【例 5-6】 假设两种分类模式如图 5-17 所示，模式 1 表示竖线，模式 2 表示横线。试设计一 LVQ 神经网络，完成这两种模式的分类。

（1）问题分析：

以图 5-18 表示输入向量的对应元素，则 LVQ 网络有 4 个输入元素，输出两类模式，所以线性层有 2 个神经元，设 01 表示横线，10 表示竖线。若以 0 表示线条划过的小方块，1 表示线性未划过的小方块，则图 5-17 所示的两类模式构成如下训练样本集：

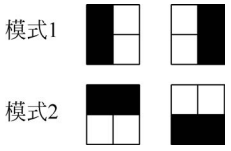

图 5-17　例 5-6 待分类的模式

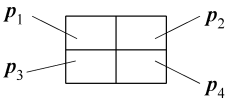

图 5-18　输入向量对应元素

$$\boldsymbol{p}_1=\begin{bmatrix}0\\1\\0\\1\end{bmatrix},\quad \boldsymbol{t}_1=[1\quad 0],\quad \boldsymbol{p}_2=\begin{bmatrix}1\\0\\1\\0\end{bmatrix},\quad \boldsymbol{t}_2=[1\quad 0]$$

$$\boldsymbol{p}_3=\begin{bmatrix}0\\0\\1\\1\end{bmatrix},\quad \boldsymbol{t}_3=[0\quad 1],\quad \boldsymbol{p}_4=\begin{bmatrix}1\\1\\0\\0\end{bmatrix},\quad \boldsymbol{t}_4=[0\quad 1]$$

（2）创建和训练 LVQ 神经网络的 MATLAB 程序设计：

```
clear all
%定义输入向量和目标向量
P=[0 1 0 1;1 0 1 0;0 0 1 1;1 1 0 0]';
T=[1 1 0 0;0 0 1 1];
%创建 LVQ 网络
net=newlvq(minmax(P),4,[0.5 0.5],0.01,'learnlv1');
%训练 LVQ 网络
net=train(net,P,T);
```

运行结果为(如图 5-19 所示)：

```
TRAINR, Epoch 0/100
TRAINR, Epoch 4/100
TRAINR, Performance goal met.
```

（3）LVQ 神经网络的 MATLAB 仿真程序设计：

```
%定义输出向量
P=[0 1 0 1;1 0 1 0;0 0 1 1;1 1 0 0]';
%网络仿真
y=sim(net,P)
```

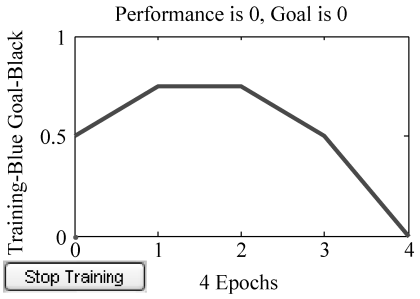

图 5-19　训练的误差性能曲线

仿真结果为：

```
y =
     1     1     0     0
     0     0     1     1
```

结果很好地完成了分类。

5.5　对向传播网络

对向传播网络(Counter Propagation Network，CPN)，是将 Kohonen 特征映射网络与 Grossberg 基本竞争型网络相结合，发挥各自特长的一种新型特征映射网络。这一网络是美国计算机专家 Robert Hecht-Nielsen 于 1987 年提出的。这种网络被广泛地应用于模式分类、函数近似、统计分析和数据压缩等领域。

5.5.1　对向传播网络简介

CPN 网络结构如图 5-20 所示。由图可见，网络分为输入层、竞争层和输出层。输入层与竞争层构成 SOM 网络，竞争层与输出层构成基本竞争型网络。从整体上看，网络属于有教师型的网络，而由输入层和竞争层构成的 SOM 网络又是一种典型的无教师型神经网络。因此，这一网络既汲取了无教师型网络分类灵活、算法简练的优点，又采纳了有教师型网络的分类精细、准确的长处，使两种不同类型的网络有机地结合起来。

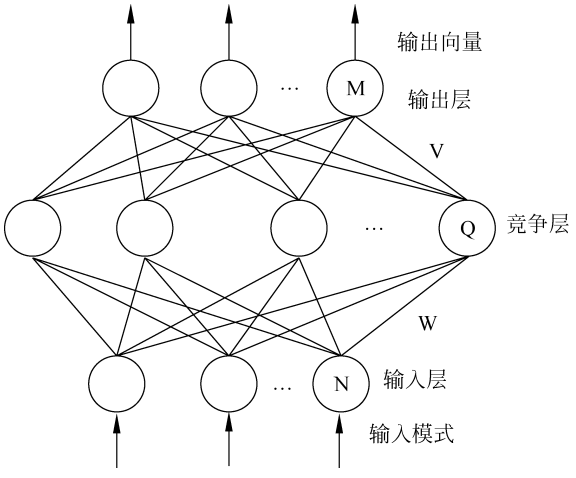

图 5-20　CPN 网络结构

CPN 的基本思想是，由输入层至输出层，网络按照 SOM 学习规则产生竞争层的获胜神经元，并按这一规则调整相应的输入层至竞争层的连接权；由竞争层到输出层，网络按照基本竞争型网络学习规则，得到各输出神经元的实际输出值，并按照有教师型的误差校正方法，修正由竞争层到输出层的连接权。经过这样的反复学习，可以将任意的输入模式映射为输出模式。

从这一基本思想可以发现，处于网络中间位置的竞争层获胜神经元及与其相关的连接权向量，既反映了输入模式的统计特性，又反映了输出模式的统计特性。因此，可以认为，输入、输出模式通过竞争层实现了相互映射，即网络具有双向记忆的功能。如果输入输出采用相同的模式对网络进行训练，则由输入模式至竞争层的映射可以认为是对输入模式的压缩；而由竞争层至输出层的映射可以认为是对输入模式的复原。利用这一特性，可以有效地解决图像处理及通信中的数据压缩及复原问题，并可得到较高的压缩性。

接下来介绍 CPN 的学习及工作规则。假定输入层有 N 个神经元，p 个连续值的输入模式为 $\boldsymbol{A}_k=(a_1^k,a_2^k,\cdots,a_N^k)$，竞争层有 Q 个神经元，对应的二值输出向量为 $\boldsymbol{B}_k=(b_1^k,b_2^k,\cdots,b_Q^k)$，输出层有 M 个神经元，其连续值的输出向量为 $\boldsymbol{C}'_k=(c'^k_1,c'^k_2,\cdots,c'^k_M)$，目标输出向量为 $\boldsymbol{C}_k=(c_1^k,c_2^k,\cdots,c_M^k)$，以上 $k=1,2,\cdots,p$。由输入层至竞争层的连接权值向量为 $\boldsymbol{W}_j=(w_{j1},w_{j2},\cdots,w_{jN})$，$j=1,2,\cdots,Q$；由竞争层到输出层的连接权向量为 $\boldsymbol{V}_l=(v_{l1},v_{l2},\cdots,v_{lQ})$，$l=1,2,\cdots,M$。网络学习和工作规则如下所述。

(1) 初始化。将连接权向量 $\boldsymbol{W}_j$ 和 $\boldsymbol{V}_l$ 赋予区间[0,1]内的随机值。将所有的输入模式 A_k 进行归一化处理：

$$a_i^k=\frac{a_i^k}{\|\boldsymbol{A}_k\|},\quad \|\boldsymbol{A}_k\|=\sqrt{\sum_{i=1}^{N}(a_i^k)^2},\quad i=1,2,\cdots,N$$

(2) 将第 k 个输入模式 $\boldsymbol{A}_k$ 提供给网络的输入层。

(3) 将连接权值向量 $\boldsymbol{W}_{ji}$ 按照下式进行归一化处理：

$$w_{ji}=\frac{w_{ji}}{\|w_{ji}\|},\quad \|w_{ji}\|=\sqrt{\sum_{i=1}^{N}w_{ji}^2},\quad i=1,2,\cdots,N$$

(4) 求竞争层中每个神经元的加权输入和：

$$s_j=\sum_{i=1}^{N}a_i^k w_{ji},\quad j=1,2,\cdots,Q$$

(5) 求连接权向量 $\boldsymbol{W}_j$ 中与 $\boldsymbol{A}_k$ 距离最近的向量 $\boldsymbol{W}_g$：

$$W_g=\max_{j=1,2\cdots,Q}\sum_{i=1}^{N}a_j^k w_{ji}=\max_{j=1,2\cdots,Q}s_j$$

将神经元 g 的输出设定为 1，其余竞争层神经元的输出设定为 0：

$$b_j=\begin{cases}1 & j=g\\ 0 & j\neq g\end{cases}$$

(6) 将连接权向量 $\boldsymbol{W}_g$ 按照下式进行修正：

$$w_{gi}(t+1)=w_{gi}(t)+\alpha(a_i^k-w_{gi}(t))\quad i=1,2,\cdots,N$$

其中，$-1<\alpha<1$ 为学习率。

(7) 将连接权向量 $\boldsymbol{W}_g$ 重新归一化，归一化算法同上。

(8) 按照下式修正竞争层到输出层的连接权向量 $\boldsymbol{V}_l$：

$$v_{li}(t+1)=v_{li}(t)+\beta b_j(c_l-c'_l)\quad l=1,2,\cdots,M,j=1,2,\cdots,Q$$

其中，$-1<\beta<1$ 为学习率。由步骤(5)可将上式简化为：

$$v_{lg}(t+1)=v_{lg}(t)+\beta b_j(c_l-c'_l)$$

由此可见，只需要调整竞争层获胜神经元 g 到输出层神经元的连接权向量 $\boldsymbol{V}_g$ 即可，其他连接权向量保持不变。

(9) 求输出层各神经元的加权输入，并将其作为输出神经元的实际输出值，$c'_l=\sum_{j=1}^{O}b_j v_{lg}$，$l=1,2,\cdots,M$，同理可将其简化为 $c'_l=v_{lg}$。

(10) 返回步骤(2)，直到将 p 个输入模式全部提供给网络。

(11) 令 $t=t+1$，将输入模式 $\boldsymbol{A}_k$ 重新提供给网络学习，直到 $t=T$。其中 T 为预先设定的学习总次数，一般取 $500<T<100000$。

5.5.2 对向传播网络的 MATLAB 实现

【例 5-7】 这里举一个非常简单而且与日常生活相关的例子来说明 CPN 网络的应用。现在需要创建一个 CPN 网络，其任务是在已知一个人本星期应该完成的工作量和此人当时的思想状态的情况下，对此人星期日下午的活动安排提出建议。

按照一般情况，将工作量分为 3 个档次，即没有、有一些和很多，所对应的量化值分别为 0.0、0.5 和 1.0；把思想情绪也分为 3 个水平，即低、一般和高，所对应的量化值分别为 0.0、0.5 和 1.0。可选择的活动有 5 个，即在家里看画报、去商场购物、到公园散步、与朋友一起吃饭和干工作。工作量和思想情绪状态一共有 6 种组合，这 6 种组合分别对应各自的最佳活动选择。样本模式如表 5-3 所示。

把这组训练样本提供给网络进行充分学习后，网络就具有了一种“内插”功能，即当网络输入一对在(0,1)区间中反映工作量和情绪的量化值后，网络将自动根据原有的记忆，找出对应于这对量化值的最佳活动选择，以输出模式的形式提供给用户作为决策参数。

表 5-3 网络训练样本模式

工作量	思想情绪	活动安排	目标输出
没有(0.0)	低(0.0)	看画报	10000
有一些(0.5)	低(0.0)	看画报	10000
没有(0.0)	一般(0.5)	购物	01000
很多(1.0)	高(1.0)	公园散步	00100
有一些(0.5)	高(1.0)	吃饭	00010
很多(1.0)	一般(0.5)	工作	00001

实际上，不光 CPN 网络具有这种“内插”功能，BP 网络、SOM 网络都具有这种功能。从模式识别的角度上讲，这些网络具有对输入模式进行分类的功能。

可惜的是，对功能如此强大的 CPN 网络，神经网络工具箱中竟然没有为之支持的函数工具。但是，既然 5.5.1 节中已经给出了有关 CPN 的学习和训练算法过程，因此，我们可以利用 MATLAB 强大的数学计算功能，实现解决该问题的 CPN 网络。

根据题意，该网络的输入层应该有 2 个神经元，输出层应该有 5 个神经元。为了更加准确地解决问题，将竞争层神经元设置为 18 个，网络结构如图 5-21 所示。

由表 5-3 可得，网络的输入向量为：

```
P=[0 0;0.5 0.5;0 0.5;1 1;0.5 1;1 0.5];
```

目标向量为：

```
T=[1 0 0 0 0;1 0 0 0 0;0 1 0 0 0;0 0 1 0 0;0 0 0 1 0;0 0 0 0 1]';
```

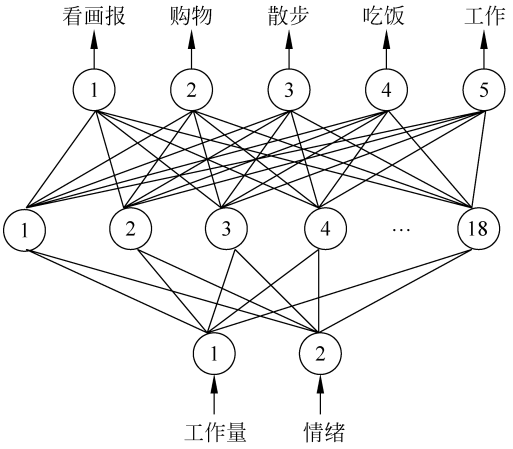

图 5-21　星期日下午活动安排决策 CPN 网络

下面对网络进行一个周期的学习。令输入层和竞争层之间的连接权向量矩阵用 $\boldsymbol{W}$ 表示，竞争层和输出层之间的权向量矩阵用 $\boldsymbol{V}$ 表示。可知 $\boldsymbol{W}$ 为一个 18×2 的矩阵，$\boldsymbol{V}$ 是一个 5×18 的矩阵。学习速率设定为 0.1。

(1) 初始化。利用 MATLAB 中的随机数产生函数 W 和 V，并赋以区间[0，1]之间的随机值，代码为：

```
W = rands(18,2)/2 + 0.5;
V = rands(5,18)/2 + 0.5;
```

由于函数 rands 产生的随机数位于区间(－1，1)，所以这里做了这样的处理，使得产生的随机数既不影响随机性能，又位于区间[0 1]中。

对输入向量进行归一化处理：

```
W = rands(18,2)/2 + 0.5;
V = rands(5,18)/2 + 0.5;
for i = 1:6
    if(P(i,:) == [0 0])
        P(i,:) = P(i,:)
    else
        P(i,:) = P(i,:)/norm(P(i,:));
    end
end
```

之所以要在循环中进行数据判断，是因为向量[0 0]是无法归一化处理的，比如 P 的第一组元素，就是正在用的这一组中。

(2) 将第一个输入样本(0 0)提供给网络的输入层神经元。

(3) 对连接权向量 W 进行归一化处理。

(4) 求每一个竞争层神经元的加权输入 s_j，j＝1，2，…，18：

```
for i = 1:18
    W(i,:) = W(i,:)/norm(W(i,:));
    s(i) = P(1,:) * W(i,:)';
end
```

循环语句中第一句用于对连接权向量 W 进行归一化处理，第二句可以求出竞争层每个神

经元的输出。结果为：

```
s =
   0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0
```

(5) 求连接权向量 W 中与(0 0)距离最近的向量，由于输出全部为 0，所以可任选一个权值向量 Wg，这里选为 18，并将该神经元的输出设定为 1。

```
for i = 1:18
    W(i,:) = W(i,:)/norm(W(i,:));
    s(i) = P(1,:) * W(i,:)';
end
temp = max(s);
for i = 1:18
    if temp == s(i)
        count = i;
    end
end
% 将所有竞争层神经元的输出置为 0
for i = 1:18
    s(i) = 0;
end
% 选中的神经元输出为 1
s(count) = 1;
```

(6) 调整连接权向量 W18，并重新将其归一化。

```
W(count,:) = W(count,:) + 0.1 * [P(1,:) - W(count,:)];
W(count,:) = W(count,:)/norm(W(count,:))
```

输出结果为：

```
W(18,:) =
      0.9997    0.0251
```

经检验，此时的 W18 确实已经归一化了。

(7) 调整竞争层神经元到输出层神经元之间的连接权向量 V：

```
V(:,count) = V(:,count) + 0.1 * (T(1,:)' - T_out(1,:)');
```

由于此时输出层神经元的输出与目标向量是一致的，所以这里的权值是没有经过调整的，等到来了下一个模式后，权值才会真正得到调整。

(8) 计算输出层各神经元的加权输入，并将其作为神经元的实际输出值：

```
T_out(1,:) = V(:,count)';
```

第一组实际输出就等于竞争层中第 18 个神经元与输出层各神经元之间调整后的连接权值。

(9) 返回步骤(2)，将输入向量中的(0.5 0.5)提供给网络。

(10) 继续学习，直到训练次数达到设定的最大值。

CPN 网络训练结束后，按照以下步骤进行网络回想。

（1）将输入模式 A 提供给网络的输入层。

（2）根据下式求出竞争层的获胜神经元 g。

$$b_g = \max_{i=1,2,\cdots,Q}\left(\sum_{i=1}^{N} w_{ji} a_i\right)$$

（3）令 $b_g=1$，其余的输出都等于 0。按照下式求得输出层各神经元的输出。

$$c_j = v_{jg} b_g$$

由此产生了输出模式 $C=(c_1, c_2, \cdots, c_M)$，从而就得到了输入 A 的分类结果。

```
clear all
%初始化正向权值 W 和反向权值 V
W = rands(18,2)/2 + 0.5;
V = rands(5,18)/2 + 0.5;
%输入向量 P 和目标向量 T
P = [0 0;0.5 0.5;0 0.5;1 3;0.5 1;1 0.5];
T = [1 0 0 0 0;1 0 0 0 0;0 1 0 0 0;0 0 1 0 0;0 0 0 1 0;0 0 0 0 1]';
T_out = T;
%设定学习步数为 1000 次
epoch = 1000;
%归一化输入向量 P
for i = 1:6
    if P(i,:) == [0 0]
        P(i,:) = P(i,:);
    else
        P(i,:) = P(i,:)/norm(P(i,:));
    end
end
%开始训练
while epoch > 0
    for j = 1:6
        %归一化正向权值 W
        for i = 1:18
            W(i,:) = W(i,:)/norm(W(i,:));
            s(i) = P(j,:) * W(i,:)';
        end
        %求输出最大的神经元,即获胜神经元
temp = max(s);
for i = 1:18
    if temp == s(i)
        count = i;
    end
end
%将所有竞争层神经元的输出置为 0
for i = 1:18
    s(i) = 0;
end
%选中的神经元输出为 1
s(count) = 1;
%权值调整
W(count,:) = W(count,:) + 0.1 * [P(j,:) - W(count,:)];
W(count,:) = W(count,:)/norm(W(count,:));
V(:,count) = V(:,count) + 0.1 * (T(j,:)' - T_out(j,:)');
%计算网络输出
T_out(j,:) = V(:,count)';
    %end
    %训练次数递减
```

```
    epoch = epoch - 1;
end
% 训练结束
T_out
% 网络回想
% 网络的输入模式 Pc
Pc = [0.5 1;1 3];
% 初始化 Pc
for i = 1:2
    if Pc(i,:) == [0 0]
        Pc(i,:) = Pc(i,:);
    else
        Pc(i,:) = P(i,:)/norm(Pc(i,:));
    end
end
% 网络输出
Outc = [0 0 0 0 0;0 0 0 0 0];
for j = 1:2
    for i = 1:18
        sc(i) = Pc(j,:) * W(i,:)';
    end
    tempc = max(sc);
    for i = 1:18
        if tempc == sc(i)
            countp = i;
        end
        sc(i) = 0;
    end
    sc(countp) = 1;
    Outc(j,:) = V(:,countp)';
end
% 回想结束
Outc
```

输出结果为：

```
T_out =
     1     1     0     0     0     0
     0     0     1     0     0     0
     0     0     0     1     0     0
     0     0     0     0     1     0
     0     0     0     0     0     1
```

由此可见，经过 1000 次训练后，网络的实际输出和目标输出就一致了，这说明训练过程是有效的。

```
Outc =
    0.3050    0.8744    0.0150    0.7680    0.9708
    0.3784    0.8600    0.8537    0.5936    0.4966
```

Outc 是网络回想的输出，实际上也就是网络测试的结果，在这里给出了两种特定的组合状态，即(0.5 1)和(1 1)的组合，这两种组合分别对应吃饭和到公园散步，可见，网络给出了正确的建议。

在以上代码中，训练输入向量 P 和回想输入向量 Pc 中的(1 1)都被(1 3)所替代，这是因为经过归一化处理后，P 中两个本来不一致的样本(0.5 0.5)和(1 1)变得一致了，而它们对应的输出向量却并不一致。因此，将其用(1 3)替代就是为了避免这种情况，对结果并没有影响。