

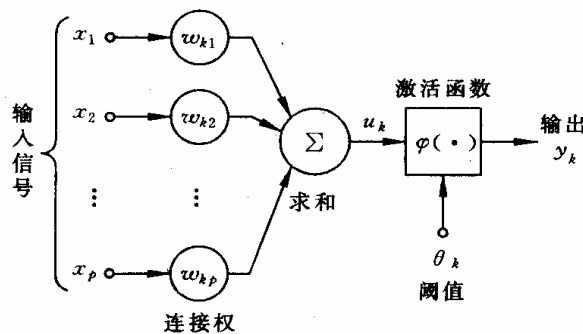
## 第十九章 神经网络模型

### §1 神经网络简介

人工神经网络是在现代神经科学的基础上提出和发展起来的，旨在反映人脑结构及功能的一种抽象数学模型。自 1943 年美国心理学家 W. McCulloch 和数学家 W. Pitts 提出形式神经元的抽象数学模型—MP 模型以来，人工神经网络理论技术经过了 50 多年曲折的发展。特别是 20 世纪 80 年代，人工神经网络的研究取得了重大进展，有关的理论和方法已经发展成一门介于物理学、数学、计算机科学和神经生物学之间的交叉学科。它在模式识别，图像处理，智能控制，组合优化，金融预测与管理，通信，机器人以及专家系统等领域得到广泛的应用，提出了 40 多种神经网络模型，其中比较著名的有感知机，Hopfield 网络，Boltzman 机，自适应共振理论及反向传播网络（BP）等。在这里我们仅讨论最基本的网络模型及其学习算法。

#### 1.1 人工神经元模型

下图表示出了作为人工神经网络（artificial neural network，以下简称 NN）的基本单元的神经元模型，它有三个基本要素：



(i) 一组连接（对应于生物神经元的突触），连接强度由各连接上的权值表示，权值为正表示激活，为负表示抑制。

(ii) 一个求和单元，用于求取各输入信号的加权和（线性组合）。

(iii) 一个非线性激活函数，起非线性映射作用并将神经元输出幅度限制在一定范围内（一般限制在 (0,1) 或 (-1,1) 之间）。

此外还有一个阈值  $\theta_k$ （或偏置  $b_k = -\theta_k$ ）。

以上作用可分别以数学式表达出来：

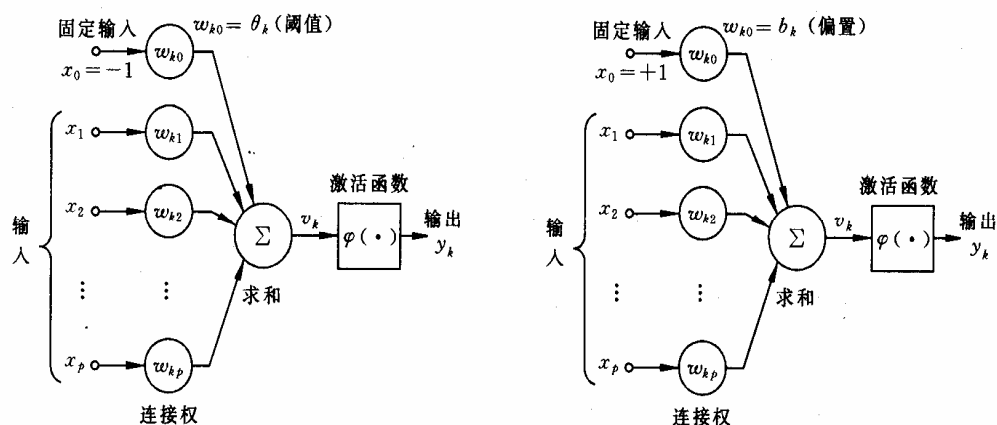
$$u_k = \sum_{j=1}^p w_{kj} x_j, \quad v_k = u_k - \theta_k, \quad y_k = \varphi(v_k)$$

式中  $x_1, x_2, \dots, x_p$  为输入信号， $w_{k1}, w_{k2}, \dots, w_{kp}$  为神经元  $k$  之权值， $u_k$  为线性组合结果， $\theta_k$  为阈值， $\varphi(\cdot)$  为激活函数， $y_k$  为神经元  $k$  的输出。

若把输入的维数增加一维，则可把阈值  $\theta_k$  包括进去。例如

$$v_k = \sum_{j=0}^p w_{kj} x_j, \quad y_k = \varphi(u_k)$$

此处增加了一个新的连接，其输入为  $x_0 = -1$ （或  $+1$ ），权值为  $w_{k0} = \theta_k$ （或  $b_k$ ），如下图所示。



激活函数  $\varphi(\cdot)$  可以有以下几种:

(i) 阈值函数

$$\varphi(v) = \begin{cases} 1, & v \geq 0 \\ 0, & v < 0 \end{cases} \quad (1)$$

即阶跃函数。这时相应的输出  $y_k$  为

$$y_k = \begin{cases} 1, & v_k \geq 0 \\ 0, & v_k < 0 \end{cases}$$

其中  $v_k = \sum_{j=1}^p w_{kj} x_j - \theta_k$ , 常称此种神经元为  $M-P$  模型。

(ii) 分段线性函数

$$\varphi(v) = \begin{cases} 1, & v \geq 1 \\ \frac{1}{2}(1+v), & -1 < v < 1 \\ 0, & v \leq -1 \end{cases} \quad (2)$$

它类似于一个放大系数为 1 的非线性放大器, 当工作于线性区时它是一个线性组合器, 放大系数趋于无穷大时变成一个阈值单元。

(iii) sigmoid 函数

最常用的函数形式为

$$\varphi(v) = \frac{1}{1 + \exp(-\alpha v)} \quad (3)$$

参数  $\alpha > 0$  可控制其斜率。另一种常用的是双曲正切函数

$$\varphi(v) = \tanh\left(\frac{v}{2}\right) = \frac{1 - \exp(-v)}{1 + \exp(-v)} \quad (4)$$

这类函数具有平滑和渐近性, 并保持单调性。

Matlab 中的激活 (传递) 函数如下表所示:

函数名	功 能
purelin	线性传递函数

hardlim	硬限幅传递函数
hardlims	对称硬限幅传递函数
satlin	饱和线性传递函数
satlins	对称饱和线性传递函数
logsig	对数 S 形传递函数
tansig	正切 S 形传递函数
radbas	径向基传递函数
compet	竞争层传递函数

各个函数的定义及使用方法，可以参看 Matlab 的帮助（如在 Matlab 命令窗口运行 help tansig，可以看到 tansig 的使用方法，及 tansig 的定义为  $\varphi(v) = \frac{2}{1 + e^{-2v}} - 1$ ）。

## 1.2 网络结构及工作方式

除单元特性外，网络的拓扑结构也是 NN 的一个重要特性。从连接方式看 NN 主要有两种。

### (i) 前馈型网络

各神经元接受前一层的输入，并输出给下一层，没有反馈。结点分为两类，即输入单元和计算单元，每一计算单元可有任意个输入，但只有一个输出（它可耦合到任意多个其它结点作为其输入）。通常前馈网络可分为不同的层，第  $i$  层的输入只与第  $i-1$  层输出相连，输入和输出结点与外界相连，而其它中间层则称为隐层。

### (ii) 反馈型网络

所有结点都是计算单元，同时也可接受输入，并向外界输出。

NN 的工作过程主要分为两个阶段：第一个阶段是学习期，此时各计算单元状态不变，各连线上的权值可通过学习来修改；第二阶段是工作期，此时各连接权固定，计算单元状态变化，以达到某种稳定状态。

从作用效果看，前馈网络主要是函数映射，可用于模式识别和函数逼近。反馈网络按对能量函数的极小点的利用来分类有两种：第一类是能量函数的所有极小点都起作用，这一类主要用作各种联想存储器；第二类只利用全局极小点，它主要用于求解最优化问题。

## §2 蠼螋分类问题与多层前馈网络

### 2.1 蠼螋分类问题

蠼螋分类问题可概括叙述如下：生物学家试图对两种蠼螋（Af 与 Apf）进行鉴别，依据的资料是触角和翅膀的长度，已经测得了 9 支 Af 和 6 支 Apf 的数据如下：

Af: (1.24,1.27), (1.36,1.74), (1.38,1.64), (1.38,1.82), (1.38,1.90), (1.40,1.70),  
(1.48,1.82), (1.54,1.82), (1.56,2.08).

Apf: (1.14,1.82), (1.18,1.96), (1.20,1.86), (1.26,2.00), (1.28,2.00), (1.30,1.96).

现在的问题是：

(i) 根据如上资料，如何制定一种方法，正确地区分两类蠼螋。

(ii) 对触角和翼长分别为(1.24,1.80)，(1.28,1.84)与(1.40,2.04)的 3 个标本，用所得到的方法加以识别。

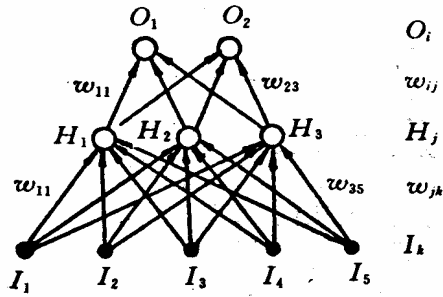
(iii) 设 Af 是宝贵的传粉益虫，Apf 是某疾病的载体，是否应该修改分类方法。

如上的问题是有代表性的，它的特点是要求依据已知资料（9 支 Af 的数据和 6 支 Apf 的数据）制定一种分类方法，类别是已经给定的（Af 或 Apf）。今后，我们将 9 支

Af 及 6 支 Apf 的数据集合称之为学习样本。

## 2.2 多层前馈网络

为解决上述问题，考虑一个其结构如下图所示的人工神经网络，



激活函数由

$$\varphi(v) = \frac{1}{1 + \exp(-\alpha v)}$$

来决定。图中最下面单元，即由●所示的一层称为输入层，用以输入已知测量值。在我们的例子中，它只需包括两个单元，一个用以输入触角长度，一个用以输入翅膀长度。中间一层称为处理层或隐单元层，单元个数适当选取，对于它的选取方法，有一些文献进行了讨论，但通过试验来决定，或许是最好的途径。在我们的例子中，取三个就足够了。最上面一层称为输出层，在我们的例子中只包含二个单元，用以输出与每一组输入数据相对应的分类信息。任何一个中间层单元接受所有输入单元传来的信号，并把处理后的结果传向每一个输出单元，供输出层再次加工，同层的神经元彼此不相联接，输入与输出单元之间也没有直接联接。这样，除了神经元的形式定义外，我们又给出了网络结构。有些文献将这样的网络称为两层前传网络，称为两层的理由是，只有中间层及输出层的单元才对信号进行处理；输入层的单元对输入数据没有任何加工，故不计算在层数之内。

为了叙述上的方便，此处引入如下记号上的约定：令  $s$  表示一个确定的已知样品标号，在蠓虫问题中， $s = 1, 2, \dots, 15$ ，分别表示学习样本中的 15 个样品；当将第  $s$  个样品的原始数据输入网络时，相应的输出单元状态记为  $O_i^s (i = 1, 2)$ ，隐单元状态记为  $H_j^s (j = 1, 2, 3)$ ，输入单元取值记为  $I_k^s (k = 1, 2)$ 。请注意，此处下标  $i, j, k$  依次对应于输出层、中间层及输入层。在这一约定下，从中间层到输出层的权记为  $w_{ij}$ ，从输入层到中间层的权记为  $\bar{w}_{jk}$ 。如果  $w_{ij}$ ， $\bar{w}_{jk}$  均已给定，那么，对应于任何一组确定的输入  $(I_1^s, I_2^s)$ ，网络中所有单元的取值不难确定。事实上，对样品  $s$  而言，隐单元  $j$  的输入是

$$\bar{h}_j^s = \sum_{k=1}^2 \bar{w}_{jk} I_k^s \quad (5)$$

相应的输出状态是

$$H_j^s = \varphi(\bar{h}_j^s) = \varphi\left(\sum_{k=1}^2 \bar{w}_{jk} I_k^s\right) \quad (6)$$

由此，输出单元  $i$  所接收到的迭加信号是

$$h_i^s = \sum_{j=1}^3 w_{ij} H_j^s = \sum_{j=1}^3 w_{ij} \varphi \left( \sum_{k=1}^2 \bar{w}_{jk} I_k^s \right) \quad (7)$$

网络的最终输出是

$$O_i^s = \varphi(h_i^s) = \varphi \left( \sum_{j=1}^3 w_{ij} H_j^s \right) = \varphi \left( \sum_{j=1}^3 w_{ij} \varphi \left( \sum_{k=1}^2 \bar{w}_{jk} I_k^s \right) \right) \quad (8)$$

这里，没有考虑阈值，正如前面已经说明的那样，这一点是无关紧要的。还应指出的是，对于任何一组确定的输入，输出是所有权  $\{w_{ij}, \bar{w}_{jk}\}$  的函数。

如果我们能够选定一组适当的权值  $\{w_{ij}, \bar{w}_{jk}\}$ ，使得对应于学习样本中任何一组 Af 样品的输入  $(I_1^s, I_2^s)$ ，输出  $(O_1^s, O_2^s) = (1, 0)$ ，对应于 Apf 的输入数据，输出为  $(0, 1)$ ，那么蠓虫分类问题实际上就解决了。因为，对于任何一个未知类别的样品，只要将其触角及翅膀长度输入网络，视其输出模式靠近  $(1, 0)$  亦或  $(0, 1)$ ，就可能判断其归属。当然，有可能出现介于中间无法判断的情况。现在的问题是，如何找到一组适当的权值，实现上面所设想的网络功能。

### 2.3 向后传播算法

对于一个多层网络，如何求得一组恰当的权值，使网络具有特定的功能，在很长一段时间内，曾经是使研究工作者感到困难的一个问题，直到 1985 年，美国加州大学的一个研究小组提出了所谓向后传播算法 (Back-Propagation)，使问题有了重大进展，这一算法也是促成人工神经网络研究迅猛发展的一个原因。下面就来介绍这一算法。

如前所述，我们希望对应于学习样本中 Af 样品的输出是  $(1, 0)$ ，对应于 Apf 的输出是  $(0, 1)$ ，这样的输出称之为理想输出。实际上要精确地作到这一点是不可能的，只能希望实际输出尽可能地接近理想输出。为清楚起见，把对应于样品  $s$  的理想输出记为  $\{T_i^s\}$ ，那么

$$E(W) = \frac{1}{2} \sum_{i,s} (T_i^s - O_i^s)^2 \quad (9)$$

度量了在一组给定的权下，实际输出与理想输出的差异，由此，寻找一组恰当的权的问题，自然地归结为求适当  $W$  的值，使  $E(W)$  达到极小的问题。将式 (8) 代入 (9)，有

$$E(W) = \frac{1}{2} \sum_{s,i} [T_i^s - \varphi \left( \sum_{j=1}^3 w_{ij} \varphi \left( \sum_{k=1}^2 \bar{w}_{jk} I_k^s \right) \right)]^2 \quad (10)$$

易知，对每一个变量  $w_{ij}$  或  $\bar{w}_{jk}$  而言，这是一个连续可微的非线性函数，为了求得其极小点与极小值，最为方便的就是使用最速下降法。最速下降法是一种迭代算法，为求出  $E(W)$  的（局部）极小，它从一个任取的初始点  $W_0$  出发，计算在  $W_0$  点的负梯度方向  $-\nabla E(W_0)$ ，这是函数在该点下降最快的方向；只要  $\nabla E(W_0) \neq 0$ ，就可沿该方向移动一小段距离，达到一个新的点  $W_1 = W_0 - \eta \nabla E(W_0)$ ， $\eta$  是一个参数，只要  $\eta$  足够小，定能保证  $E(W_1) < E(W_0)$ 。不断重复这一过程，一定能达到  $E$  的一个（局部）极小点。就本质而言，这就是 BP 算法的全部内容，然而，对人工神经网络问题而言，这一算法的具体形式是非常重要的，下面我们就来给出这一形式表达。

对于隐单元到输出单元的权  $w_{ij}$  而言，最速下降法给出的每一步的修正量是

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = \eta \sum_s [T_i^s - O_i^s] \varphi'(h_i^s) H_j^s = \eta \sum_s \delta_i^s H_j^s \quad (11)$$

此处令

$$\delta_i^s = \varphi'(h_i^s) [T_i^s - O_i^s] \quad (12)$$

对输入单元到隐单元的权  $\bar{w}_{jk}$

$$\begin{aligned} \Delta \bar{w}_{jk} &= -\eta \frac{\partial E}{\partial \bar{w}_{jk}} = \eta \sum_{s,i} [T_i^s - O_i^s] \varphi'(h_i^s) w_{ij} \varphi'(h_j^s) I_j^s \\ &= \eta \sum_{s,i} \delta_i^s w_{ij} \varphi'(h_j^s) I_j^s = \eta \sum_s \bar{\delta}_j^s I_k^s \end{aligned} \quad (13)$$

此处

$$\bar{\delta}_j^s = \varphi'(h_j^s) \sum_i w_{ij} \delta_i^s$$

从 (11) 和 (13) 式可以看出, 所有权的修正量都有如下形式, 即

$$\Delta w_{pq} = \eta \sum_s \delta_p^s v_q^s \quad (14)$$

指标  $p$  对应于两个单元中输出信号的一端,  $q$  对应于输入信号的一端,  $v$  或者代表  $H$  或者代表  $I$ 。形式上看来, 这一修正是“局部”的, 可以看作是 Hebb 律的一种表现形式。还应注意,  $\delta_i^s$  由实际输出与理想输出的差及  $h_i^s$  决定, 而  $\bar{\delta}_j^s$  则需依赖  $\delta_i^s$  算出, 因此, 这一算法才称为向后传播算法。稍加分析还可知道, 利用由 (11) ~ (13) 式所给出的计算安排, 较之不考虑  $\delta_p^s$  的向后传播, 直接计算所有含  $\varphi'$  的原表达式, 极大地降低了计算工作量。这组关系式称作广义  $\delta$ -法则, 它们不难推广到一般的多层网络上。

利用这一迭代算法, 最终生成在一定精度内满足要求的  $\{w_{ij}, \bar{w}_{jk}\}$  的过程, 称为人工神经网络的学习过程。可以看出, 这里所提供的学习机制是元与元之间权的不断调整, 学习样本中任何一个样品所提供的信息, 最终将包含在网络的每一个权之中。参数  $\eta$  的大小则反映了学习效率。

为了更有效地应用 BP 算法, 我们做出如下一些补充说明。

(i) 在式 (11) 与 (13) 中,  $\Delta w_{ij}, \Delta \bar{w}_{jk}$  表示为与所有样品  $s$  有关的求和计算。

实际上, 我们还可以每次仅考虑输入一个样品所造成的修正, 然后, 按照随机选取的顺序, 将所有样品逐个输入, 不断重复这一手续, 直至收敛到一个满意的解为止。

(ii) 在如上的算法中, 利用实际输出与理想输出差的平方和作为度量  $\{w_{ij}, \bar{w}_{jk}\}$  优劣的标准, 这并不是唯一的度量方式, 完全可以从其它的函数形式出发, 例如从相对熵出发, 导出相应的算法。

(iii) 在如上的讨论中使用的是最速下降法, 显然, 这也不是唯一的选择, 其它的非线性优化方法, 诸如共轭梯度法, 拟牛顿法等, 都可用于计算。为了加速算法的收敛速度, 还可以考虑各种不同的修正方式。

(iv) BP 算法的出现, 虽然对人工神经网络的发展起了重大推动作用, 但是这一算法仍有很多问题。对于一个大的网络系统, BP 算法的工作量仍然是十分可观的, 这主要在于算法的收敛速度很慢。更为严重的是, 此处所讨论的是非线性函数的优化, 那么它就无法逃脱该类问题的共同困难: BP 算法所求得解, 只能保证是依赖于初值选取的局部极小点。为克服这一缺陷, 可以考虑改进方法, 例如模拟退火算法, 或从多个

随机选定的初值点出发，进行多次计算，但这些方法都不可避免地加大了工作量。

#### 2.4 蠼螋分类问题的求解

下面利用上文所叙述的网络结构及方法，对蠼螋分类问题求解。编写 Matlab 程序如下：

```
clear
p1=[1.24,1.27;1.36,1.74;1.38,1.64;1.38,1.82;1.38,1.90;
    1.40,1.70;1.48,1.82;1.54,1.82;1.56,2.08];
p2=[1.14,1.82;1.18,1.96;1.20,1.86;1.26,2.00
    1.28,2.00;1.30,1.96];
p=[p1;p2]';
pr=minmax(p);
goal=[ones(1,9),zeros(1,6);zeros(1,9),ones(1,6)];
plot(p1(:,1),p1(:,2),'h',p2(:,1),p2(:,2),'o')
net=newff(pr,[3,2],{'logsig','logsig'});
net.trainParam.show = 10;
net.trainParam.lr = 0.05;
net.trainParam.goal = 1e-10;
net.trainParam.epochs = 50000;
net = train(net,p,goal);
x=[1.24 1.80;1.28 1.84;1.40 2.04]';
y0=sim(net,p)
y=sim(net,x)
```

### §3 处理蠼螋分类的另一种网络方法

#### 3.1 几个有关概念

在介绍本节主要内容之前，首先说明几个不同的概念。在上一节中，我们把利用 BP 算法确定联接强度，即权值的过程称为“学习过程”，这种学习的特点是，对任何一个输入样品，其类别事先是已知的，理想输出也已事先规定，因而从它所产生的实际输出与理想输出的异同，我们清楚地知道网络判断正确与否，故此把这一类学习称为在教师监督下的学习；与它不同的是，有些情况下学习是无监督的，例如，我们试图把一组样品按其本身特点分类，所要划分的类别是事先未知的，需要网络自身通过学习来决定，因而，在学习过程中，对每一输入所产生的输出也就无所谓对错，对于这样的情况，显然 BP 算法是不适用的。

另一个有关概念是所谓有竞争的学习。在上节所讨论的蠼螋分类网络中，尽管我们所希望的理想输出是 (1,0) 或 (0,1)，但实际输出并不如此，一般而言，两个输出单元均同时不为 0。与此不同，我们完全可以设想另外一种输出模式：对应任何一组输入，所有输出单元中，只允许有一个处于激发态，即取值为 1，其它输出单元均被抑制，即取值为 0。一种形象的说法是，对应任何一组输入，要求所有的输出单元彼此竞争，唯一的胜利者赢得一切，失败者一无所获，形成这样一种输出机制的网络学习过程，称为有竞争的学习。

#### 3.2 最简单的无监督有竞争的学习

本节叙述一种无监督有竞争的网络学习方法，由此产生的网络可用于将一组输入样品自动划分类别，相似的样品归于同一类别，因而激发同一输出单元，这一分类方式，是网络自身通过学习，从输入数据的关系中得出的。

蠼螋分类问题对应有教师的网络学习过程，显然不能由如上的方法来解决。但在这种无监督有竞争的学习阐明之后，很容易从中导出一种适用于有监督情况的网络方法；此外，本节所介绍的网络，在数据压缩等多种领域，都有其重要应用。

考虑一个仅由输入层与输出层组成的网络系统,输入单元数目与每一样品的测量值数目相等,输出单元数目适当选取。每一个输入单元与所有输出单元联接,第  $j$  个输入元到第  $i$  个输出元的权记为  $w_{ij}$ , 同层单元间无横向联接。不妨假设所有输入数值均已规化到  $[-1,1]$  之间,又因为是有竞争的学习,输出单元只取 0 或 1 两个值,且对应每一组输入,只有一个输出元取 1。

取 1 的输出元记为  $i^*$ , 称之为优胜者。对于任何一组输入  $s$ , 规定优胜者是有最大净输入的输出元, 即对输入  $I = (I_1, \dots, I_n)$  而言,

$$h_i = \sum_j w_{ij} I_j \equiv W_i \cdot I \quad (15)$$

取最大值的单元, 其中  $W_i$  是输出元  $i$  所有权系数组成的向量, 也就是说

$$W_{i^*} \cdot I \geq W_i \cdot I, \quad (\forall i) \quad (16)$$

如果权向量是按照  $\sum_j w_{ij}^2 = 1$  的方式标准化的, (16) 式等价于

$$|W_{i^*} - I| \leq |W_i - I|, \quad (\forall i) \quad (17)$$

即优胜者是其标准化权向量最靠近输入向量的输出元。令  $O_{i^*} = 1$ , 其余的输出  $O_i = 0$ 。这样的输出规定了输入向量的类别, 但为了使这种分类方式有意义, 问题化为如何将学习样本中的所有样品, 自然地划分为聚类, 并对每一聚类找出适当的权向量。为此, 采用如下的算法: 随机取定一组不大的初始权向量, 注意不使它们有任何对称性。然后, 将已知样品按照随机顺序输入网络。对输入样品  $s$ , 按上文所述确定优胜者  $i^*$ , 对所有与  $i^*$  有关的权作如下修正

$$\Delta w_{i^*j} = \eta(I_j^s - w_{i^*j}) \quad (18)$$

所有其它输出单元的权保持不变。注意到  $O_{i^*} = 1$ ,  $O_i = 0 (i \neq i^*)$ , 所有权的修正公式可统一表示为

$$\Delta w_{ij} = \eta O_i (I_j^s - w_{ij})$$

这一形式也可视为 Hebb 律的一种表现。(18) 式的几何意义是清楚的, 每次修正将优胜者的权向量向输入向量移近一小段距离, 这使得同一样品再次输入时,  $i^*$  有更大的获胜可能。可以合理地预期, 反复重复以上步骤, 使得每个输出单元对应了输入向量的一个聚类, 相应的权向量落在了该聚类样品的重心附近。当然, 这只是一个极不严密的说明。

特别应当指出, 上述算法, 对于事先按照  $\sum I_j = 1$  标准化了的输入数据更为适用, 整个过程不难由计算机模拟实现。

为了更有效地使用如上算法, 下面对实际计算时可能产生的问题, 作一些简要说明。

首先, 如果初始权选择不当, 那么可能出现这样的输出单元, 它的权远离任何输入向量, 因此, 永远不会成为优胜者, 相应的权也就永远不会得到修正, 这样的单元称之为死单元。为避免出现死单元, 可以有多种方法。一种办法是初始权从学习样本中抽样选取, 这就保证了它们都落在正确范围内; 另一种办法是修正上述的学习算法, 使得每一步不仅调整优胜者的权, 同时也以一个小得多的  $\eta$  值, 修正所有其它的权。这样, 对于总是失败的单元, 其权逐渐地朝着平均输入方向运动, 最终也会在某一次竞争中取胜。此外, 还存在有多种处理死单元的方法, 感兴趣的读者可从文献中找到更多的方法。



另外一个问题是这一算法的收敛性。如果式(18)或(19)中反映学习效率的参数 $\eta$ 取为一个固定常数,那么权向量永远不会真正在某一有限点集上稳定下来。因此,应当考虑在公式中引进随学习时间而变化的收敛因子。例如,取 $\eta = \eta(t) = \eta_0 t^{-a}$ ,  $0 < a \leq 1$ 。这一因子的适当选取是极为重要的, $\eta$ 下降太慢,无疑增加了不必要工作量, $\eta$ 下降太快,则会使学习变得无效。

### 3.3 LVQ 方法

上述有竞争学习的一个最重要应用是数据压缩中的向量量子化方法 (Vector Quantization)。它的基本想法是,把一个给定的输入向量集合 $I^s$ 分成 $M$ 个类别,然后用类别指标来代表所有属于该类的向量。向量分量通常取连续值,一旦一组适当的类别确定之后,代替传输或存储输入向量本身,可以只传输或存储它的类别指标。所有的类别由 $M$ 个所谓“原型向量”来表示,我们可以利用一般的欧氏距离,对每一个输入向量找到最靠近的原型向量,作为它的类别。显然,这种分类方法可以通过有竞争的学习直接得到。一旦学习过程结束,所有权向量的集合,便构成了一个“电码本”。

一般而言,上述无监督有竞争的学习,实际提供了一种聚类分析方法,对如蠓虫分类这种有监督的问题并不适用。1989年, Kohonen 对向量量子化方法加以修改,提出了一种适用于有监督情况的学习方法,称为学习向量量子化 (Learning Vector Quantization),该方法可用于蠓虫分类问题。在有监督的情况下,学习样品的类别是事先已知的,与此相应,每个输出单元所对应的类别也事先作了规定,但是,代表同一类别的输出单元可以不止一个。

在 LVQ 中,对于任一输入向量,仍按无监督有竞争的方式选出优胜者 $i^*$ ,但权的修正规则则依输入向量的类别与 $i^*$ 所代表的是否一致而不同,确切地说,令

$$\Delta w_{i^*j} = \begin{cases} \eta(I_j^s - w_{i^*j}) & \text{一致情况} \\ -\eta(I_j^s - w_{i^*j}) & \text{不一致情况} \end{cases}$$

前一种情况,修正和无监督的学习一致,权朝向样品方向移动一小段距离;后一种则相反,权向离开样品方向移动,这样就减少了错误分类的机会。

对于上述的蠓虫分类问题,我们编写 Matlab 程序如下:

```
clear
p1=[1.24,1.27;1.36,1.74;1.38,1.64;1.38,1.82;1.38,1.90;
    1.40,1.70;1.48,1.82;1.54,1.82;1.56,2.08];
p2=[1.14,1.82;1.18,1.96;1.20,1.86;1.26,2.00
    1.28,2.00;1.30,1.96];
p=[p1;p2]'
pr=minmax(p)
goal=[ones(1,9),zeros(1,6);zeros(1,9),ones(1,6)]
net = newlvq(pr,4,[0.6,0.4])
net = train(net,p,goal)
Y = sim(net,p)
x=[1.24 1.80;1.28 1.84;1.40 2.04]
sim(net,x)
```

### 习 题 十 九

1. 利用 BP 算法及 sigmoid 函数,研究以下各函数的逼近问题

(i)  $f(x) = \frac{1}{x}, \quad 1 \leq x \leq 100$

(ii)  $f(x) = \sin x, \quad 0 \leq x \leq \frac{\pi}{2}$

对每一函数要完成如下工作：

① 获取两组数据，一组作为训练集，一组作为测试集；

② 利用训练集训练一个单隐层的网络；用测试集检验训练结果，改变隐层单元数，研究它对逼近效果的影响。

2. 给定待拟合的曲线形式为

$$f(x) = 0.5 + 0.4 \sin(2\pi x)$$

在  $f(x)$  上等间隔取 11 个点的数据，在此数据的输出值上加均值为 0，均方差  $\sigma = 0.05$  的正态分布噪声作为给定训练数据，用多项式拟合此函数，分别取多项式的阶次为 1，3 和 11 阶，图示出拟合结果，并讨论多项式阶次对拟合结果的影响。