第一篇　神经网络控制及其 MATLAB 实现

第 1 章　神经网络理论

人脑是一部不寻常的智能机，它能以惊人的高速度解释感觉器官传来的含糊不清的信息。它能觉察到喧闹房间内的窃窃私语，能识别出光线暗淡的胡同中的一张面孔，更能通过不断的学习而产生伟大的创造力。古今中外，许许多多科学家为了揭开大脑机能的奥秘，从不同的角度进行着长期不懈的努力和探索，逐渐形成了一个多学科交叉的前沿技术领域——神经网络（Neural Network）。

人工神经系统的研究可以追溯到 1800 年 Frued 的精神分析学时期，他已经做了一些初步工作。1913 年人工神经系统的第一个实践是由 Russell 描述的水力装置。1943 年美国心理学家 Warren S McCulloch 与数学家 Walter H Pitts 合作，用逻辑的数学工具研究客观事件在形式神经网络中的描述，从此开创了对神经网络的理论研究。他们在分析、总结神经元基本特性的基础上，首先提出神经元的数学模型，简称 MP 模型。从脑科学研究来看，MP 模型不愧为第一个用数理语言描述脑的信息处理过程的模型。后来 MP 模型经过数学家的精心整理和抽象，最终发展成一种有限自动机理论，再一次展现了 MP 模型的价值，此模型沿用至今，直接影响着这一领域研究的进展。1949 年心理学家 D.O.Hebb 提出关于神经网络学习机理的“突触修正假设”，即突触联系效率可变的假设，现在多数学习机仍遵循 Hebb 学习规则。1957 年，Frank Rosenblatt 首次提出并设计制作了著名的感知机（Perceptron），第一次从理论研究转入过程实现阶段，掀起了研究人工神经网络的高潮。虽然，从 20 世纪 60、70 年代，MIT 电子研究实验室的 Marvin Minsky 和 Seymour Dapret 就开始对感知机做深入的评判，并于 1969 年他们出版了“Perceptron”一书，对 Frank Rosenblatt 的感知机的抽象版本做了详细的数学分析，认为感知机基本上是一个不值得研究的领域，曾一度使神经网络的研究陷入低谷。但是，从 1982 年美国物理学家 Hopfield 提出 Hopfield 神经网络，以及 1986 年 D.E.Rumelhart 和 J. L. McClelland 提出一种利用误差反向传播训练算法的 BP（Back Propagation）神经网络开始，在世界范围内再次掀起了神经网络的研究热潮。今天，随着科学技术的迅猛发展，神经网络正以极大的魅力吸引着世界上众多专家、学者为之奋斗，在世界范围内再次掀起了神经网络的研究热潮。难怪有关国际权威人士评论指出，目前对神经网络的研究其重要意义不亚于第二次世界大战时对原子弹的研究。

人工神经网络特有的非线性适应性信息处理能力，克服了传统人工智能方法对于直觉，如模式、语音识别、非结构化信息处理方面的缺陷，使之在神经专家系统、模式识别、智能控制、组合优化、预测等领域得到成功应用。人工神经网络与其他传统方法相结合，将推动人工智能和信息处理技术不断发展。近年来，人工神经网络正向模拟人类认知的道路上更加深入发展，与模糊系统、遗传算法、进化机制等结合，形成计算智能，成为人工智能的一个重要方向，将在实际应用中得到发展。

使用神经网络的主要优点是能够自适应样本数据，当数据中有噪声、形变和非线性时，它也能够正常地工作，很容易继承现有的领域知识，使用灵活，能够处理来自多个资源和决策系统的数据；提供简单工具进行自动特征选取，产生有用的数据表示，可作为专家系统的

前端（预处理器）。此外，神经网络还能提供十分快的优化过程，尤其以硬件直接实现网络时，可以加速联机应用程序的运行速度。当然，过分夸大神经网络的应用能力也是不恰当的，毕竟它不是无所不能的。这就需要在实际工作中具体问题具体分析，合理选择。

基于神经网络的控制称为神经网络控制（NNC），简称神经控制（Neuro Control，NC）这一新词是在国际自控联杂志《自动化》（Automatica）1994 年 No.11 首次使用的，最早源于 1992 年 H.Tolle 和 E.Ersu 的专著《Neuro Control》。基于神经网络的智能模拟用于控制，是实现智能控制的一种重要形式，近年来获得了迅速发展。本章介绍神经网络的基本概念、基本结构和学习算法等。

1.1　神经网络的基本概念

1.1.1　生物神经元的结构与功能特点

神经生理学和神经解剖学证明了人的思维是由人脑完成的。神经元是组成人脑的最基本单元，它能够接收并处理信息，人脑大约由 10^{11}～10^{12} 个神经元组成，其中每个神经元约与 10^4～10^5 个神经元通过突触连接，因此，人脑是一个复杂的信息并行加工处理巨系统。探索脑组织的结构、工作原理及信息处理的机制，是整个人类面临的一项挑战，也是整个自然科学的前沿领域。

1．生物神经元的结构

生物神经元，也称为神经细胞，是构成神经系统的基本单元。生物神经元主要由细胞体、树突和轴突构成，其基本结构如图 1-1 所示。

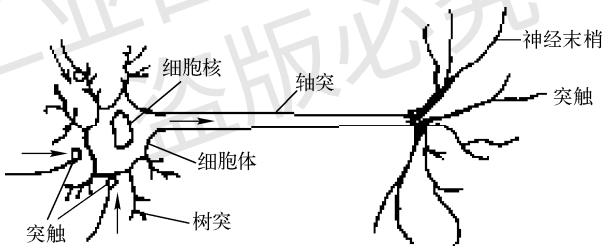


图 1-1　生物神经元结构

1）细胞体

细胞体由细胞核、细胞质与细胞膜等组成。一般直径为 5～100μm，大小不等。细胞体是生物神经元的主体，它是生物神经元的新陈代谢中心，同时还负责接收并处理从其他神经元传递过来的信息。细胞体的内部是细胞核，外部是细胞膜，细胞膜外是许多外延的纤维，细胞膜内外有电位差，称为膜电位，膜外为正，膜内为负。

2）轴突

轴突是由细胞体向外伸出的所有纤维中最长的一条分支。每个生物神经元只有一个轴突，长度最大可达 1m 以上，其作用相当于生物神经元的输出电缆，它通过尾部分出的许多神经末梢以及梢端的突触向其他生物神经元输出神经冲动。

轴突终端的突触，是生物神经元之间的连接接口，每一个生物神经元有 10^4～10^5 个突触。一个生物神经元通过其轴突的神经末梢，经突触与另一生物神经元的树突连接，以实现信息的传递。

3）树突

树突是由细胞体向外伸出的除轴突外的其他纤维分支，长度一般均较短，但分支很多。它相当于神经元的输入端，用于接收从四面八方传来的神经冲动。

2．生物神经元的功能特点

从生物控制论的观点来看，作为控制和信息处理基本单元的生物神经元，具有以下功能特点。

1）时空整合功能

生物神经元对于不同时间通过同一突触传入的信息，具有时间整合功能；对于同一时间通过不同突触传入的信息，具有空间整合功能。两种功能相互结合，使生物神经元具有时空整合的输入信息处理功能。

2）动态极化性

在每一种生物神经元中，信息都是以预知的确定方向流动的，即从生物神经元的接收信息部分（细胞体、树突）传到轴突的起始部分，再传到轴突终端的突触，最后再传给另一生物神经元。尽管不同的生物神经元在形状及功能上都有明显的不同，但大多数生物神经元都是按这一方向进行信息流动的。

3）兴奋与抑制状态

生物神经元具有两种常规工作状态，即兴奋状态与抑制状态。所谓兴奋状态是指生物神经元对输入信息经整合后使细胞膜电位升高，且超过了动作电位的阈值，此时产生神经冲动并由轴突输出。抑制状态是指对输入信息整合后，细胞膜电位值下降到低于动作电位的阈值，从而导致无神经冲动输出。

4）结构的可塑性

由于突触传递信息的特性是可变的，也就是它随着神经冲动传递方式的变化，传递作用强弱不同，形成了生物神经元之间连接的柔性，这种特性又称为生物神经元结构的可塑性。

5）脉冲与电位信号的转换

突触界面具有脉冲与电位信号的转换功能。沿轴突传递的电脉冲是等幅的、离散的脉冲信号，而细胞膜电位变化为连续的电位信号，这两种信号是在突触接口进行变换的。

6）突触时延和不应期

突触对信息的传递具有时延和不应期，在相邻的两次输入之间需要一定的时间间隔，在此期间，无激励，不传递信息，这称为不应期。

7）学习、遗忘和疲劳

由于生物神经元结构的可塑性，突触的传递作用有增强、减弱和饱和三种情况。所以，神经细胞也具有相应的学习、遗忘和疲劳效应（饱和效应）。

1.1.2　人工神经元模型

生物神经元经抽象化后，可得到如图 1-2 所示的一种人工神经元模型，它有三个基本要素。

1．连接权

连接权对应于生物神经元的突触，各个神经元之间的连接强度由连接权的权值表示，权值为

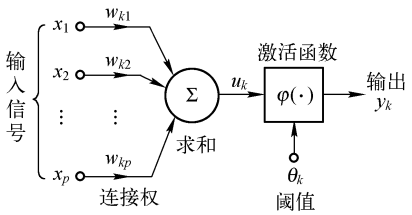


图 1-2　基本神经元模型

正表示激活，为负表示抑制。

2．求和单元

用于求取各输入信号的加权和（线性组合）。

3．激活函数

激活函数起非线性映射作用，并将人工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。激活函数也称为传输函数。

此外，还有一个阈值θ_k（或偏值 $b_k=-\theta_k$）。

图 1-2 中的作用可分别以数学式表达出来，即

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

式中，$x_1,x_2,\cdots,x_p$ 为输入信号，它相当于生物神经元的树突，为人工神经元的输入信息；$w_{k1},w_{k2},\cdots,w_{kp}$ 为神经元 k 的权值；u_k 为线性组合结果；θ_k 为阈值；$\varphi(\cdot)$ 为激活函数；y_k 为神经元 k 的输出，它相当于生物神经元的轴突，为人工神经元的输出信息。

在人工神经网络中，为了简化起见，通常将图 1-2 所示的基本神经元模型表示成图 1-3（a）或（b）所示的简化形式。

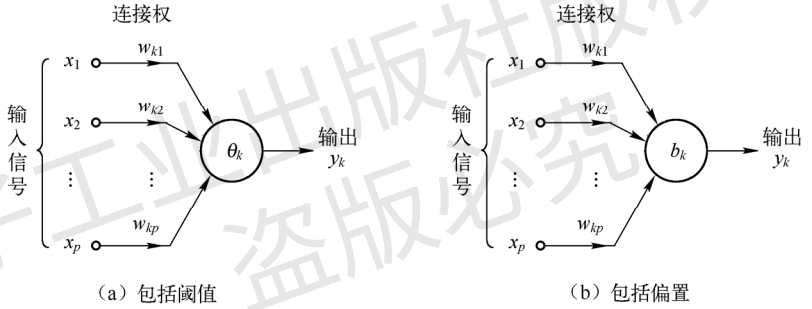


图 1-3　神经元模型的简化形式

图 1-3（a）和（b）中的输入与输出之间的关系，可分别采用以下数学表达式进行描述

$$y_k=\varphi\left(\sum_{j=1}^{p}w_{kj}x_j-\theta_k\right)\qquad\text{和}\qquad y_k=\varphi\left(\sum_{j=1}^{p}w_{kj}x_j+b_k\right)$$

激活函数$\varphi(\cdot)$一般采用以下几种形式。

1）阶跃函数

函数表达式为

$$y=\varphi(x)=\begin{cases}1 & (x\geqslant 0)\\ -1 & (x<0)\end{cases}$$

2）分段线性函数

函数表达式为

$$y=\varphi(x)=\begin{cases}1 & (x\geqslant 1)\\ \dfrac{1}{2}(1+x) & (-1<x<1)\\ -1 & (x\leqslant -1)\end{cases}$$

3）Sigmoid 型函数

最常用的 Sigmoid 型函数为

$$\varphi(x)=\frac{1}{1+\exp(-ax)}$$

式中，参数 a 可控制其斜率。

Sigmoid 型函数也简称 S 型函数，上式表示的是一种非对称 S 型函数。

另一种常用的 Sigmoid 型函数为双曲正切对称 S 型函数，即

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

这类函数具有平滑和渐近线，并保持单调性。

1.1.3　神经网络的结构

人工神经网络（Artificial Neural Networks，ANN）是由大量人工神经元经广泛互连而组成的，它可用来模拟脑神经系统的结构和功能。人工神经网络可以看成以人工神经元为节点，用有向加权弧连接起来的有向图。在此有向图中，人工神经元（以下在不易引起混淆的情况下，人工神经元简称神经元）就是对生物神经元的模拟，而有向加权弧则是轴突—突触—树突对的模拟。有向弧的权值表示相互连接的两个人工神经元间相互作用的强弱。

人工神经网络是生物神经网络的一种模拟和近似。它主要从两个方面进行模拟。一种是从生理结构和实现机理方面进行模拟，它涉及生物学、生理学、心理学、物理及化学等许多基础科学。由于生物神经网络的结构和机理相当复杂，现在距离完全认识它们还相差甚远。另外一种是从功能上加以模拟，即尽量使得人工神经网络具有生物神经网络的某些功能特性，如学习、识别、控制等功能。本书仅讨论后者，从功能上来看，人工神经网络（以下简称神经网络或 NN）根据连接方式主要分为两类。

1．前馈型网络

前馈神经网络是整个神经网络体系中最常见的一种网络，其网络中各个神经元接收前一级的输入，并输出到下一级，网络中没有反馈，如图 1-4 所示。节点分为两类，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他节点作为输入）。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，输入和输出节点与外界相连，而其他中间层称为隐含层，它们是一种强有力的学习系统，其结构简单而易于编程。从系统的观点看，前馈神经网络是一静态非线性映射，通过简单非线性处理的复合映射可获得复杂的非线性处理能力。但从计算的观点看，前馈神经网络并非是一种强有力的计算系统，不具有丰富的动力学行为。大部分前馈神经网络是学习网络，并不注意系统的动力学行为，它们的分类能力和模式识别能力一般强于其他类型的神经网络。

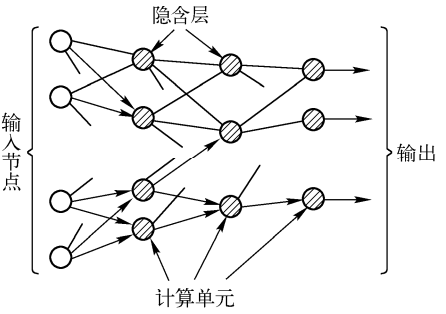


图 1-4　前馈网络

2．反馈型网络

反馈神经网络又称为递归网络或回归网络。在反馈神经网络（Feedback NN）中，输入

信号决定反馈系统的初始状态，然后系统经过一系列状态转移后，逐渐收敛于平衡状态。这样的平衡状态就是反馈网络经计算后输出的结果，由此可见，稳定性是反馈网络中最重要的问题之一。如果能找到网络的 Lyapunov 函数，则能保证网络从任意的初始状态都能收敛到局部最小点。反馈神经网络中所有节点都是计算单元，同时也可接收输入，并向外界输出，可画成一个无向图，如图 1-5（a）所示，其中每个连接弧都是双向的，也可画成如图 1-5（b）所示的形式。若总单元数为 n，则每一个节点有 n–l 个输入和一个输出。

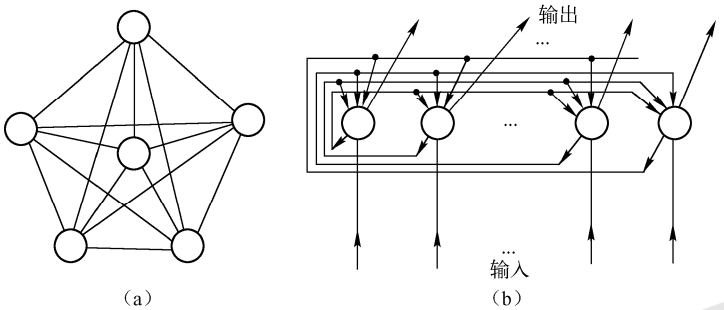


图 1-5　单层全连接反馈网络

1.1.4　神经网络的工作方式

神经网络的工作过程主要分为两个阶段：第一阶段是学习期，此时各计算单元状态不变，各连接权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。

1.1.5　神经网络的学习

1. 学习方式

通过向环境学习获取知识并改进自身性能是神经网络的一个重要特点，在一般情况下，性能的改善是按某种预定的度量调节自身参数（如权值）随时间逐步达到的，学习方式（按环境所供信息的多少分）有以下三种。

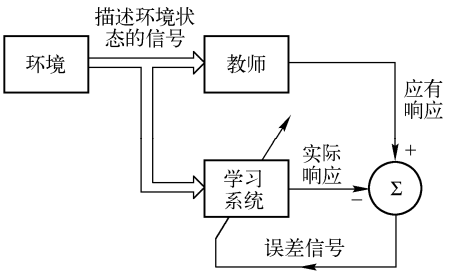


图 1-6　有监督学习框图

1）有监督学习（有教师学习）

有监督学习方式需要外界存在一个“教师”，它可对一组给定输入提供应有的输出结果（正确答案），这组已知的输入/输出数据称为训练样本集。学习系统可根据已知输出与实际输出之间的差值（误差信号）来调节系统参数，如图 1-6 所示。在有监督学习当中，学习规则由一组描述网络行为的训练集给出：

$$\{\boldsymbol{x}^{(1)},\boldsymbol{t}^{(1)}\},\{\boldsymbol{x}^{(2)},\boldsymbol{t}^{(2)}\},\cdots,\{\boldsymbol{x}^{p},\boldsymbol{t}^{p}\},\cdots,\{\boldsymbol{x}^{N},\boldsymbol{t}^{N}\}$$

其中，$\boldsymbol{x}^{p}$ 为网络的第 p 个输入数据向量；$\boldsymbol{t}^{p}$ 为对应 $\boldsymbol{x}^{p}$ 的目标输出向量；N 为训练集中的样本数。

当输入作用到网络时，网络的实际输出与目标输出相比较，然后学习规则调整网络的权值和阈值，从而使网络的实际输出越来越接近于目标输出。

2）无监督学习（无教师学习）

无监督学习时不存在外部教师，学习系统完全按照环境所提供数据的某些统计规律来调节自身参数或结构（这是一种自组织过程），以表示外部输入的某种固有特性（如聚类，或某种统计上的分布特征），如图 1-7 所示。在无监督学习中，仅仅根据网络的输入调整网络的权值和阈值，它没有目标输出。乍一看，这种学习似乎并不可行：不知道网络的目的是什么，还能够训练网络吗？实际上，大多数这种类型的算法都是要完成某种聚类操作，学会将输入模式分为有限的几种类型。这种功能特别适合于诸如向量量化等应用问题。

3）强化学习（或再励学习）

这种学习介于上述两种情况之间，外部环境对系统输出结果只给出评价（奖或罚）而不是给出正确答案，学习系统通过强化那些受奖励的动作来改善自身性能，如图 1-8 所示。强化学习与有监督的学习类似，只是它不像有监督的学习一样为每一个输入提供相应的目标输出，而是仅仅给出一个级别。这个级别（或评分）是对网络在某些输入序列上的性能测度。当前这种类型的学习要比有监督的学习少见。它最为适合控制系统应用领域。

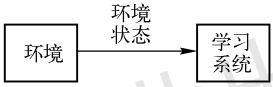


图 1-7　无监督学习框图

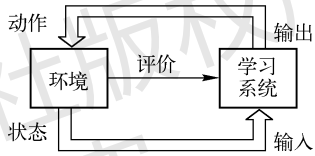


图 1-8　强化学习框图

2．学习算法

1）δ学习规则（误差纠正规则）

若 $y_i(k)$为输入 $\boldsymbol{x}(k)$时神经元 i 在 k 时刻的实际输出，$t_i(k)$表示相应的期望输出，则误差信号可写为

$$e_i(k)= t_i(k)-y_i(k)$$

误差纠正学习的最终目的是使某一基于 $e_i(k)$的目标函数达最小，以使网络中每一输出单元的实际输出在某种统计意义上最逼近期望输出。一旦选定了目标函数形式，误差纠正学习就成为一个典型的最优化问题。最常用的目标函数是均方误差判据，定义为

$$J = E\left\{\frac{1}{2}\sum_{i=1}^{L}(t_i - y_i)^2\right\}$$

式中，E 是统计期望算子；L 为网络输出数。

上式的前提是被学习的过程是宽而平稳的，具体方法可用最陡梯度下降法。直接用 J 作为目标函数时，需要知道整个过程的统计特性，为解决这一困难，用 J 在时刻 k 的瞬时值 $J(k)$代替 J，即

$$J(k) = \frac{1}{2}\sum_{i=1}^{L}(t_i - y_i)^2 = \frac{1}{2}\sum_{i=1}^{L}e_i^2(k)$$

问题变为求 $J(k)$对权值 w_{ij} 的极小值，根据最陡梯度下降法可得：

$$\Delta w_{ij}(k) = \eta \cdot \delta_i(k) \cdot x_j(k) = \eta \cdot e_i(k) \cdot f'(\boldsymbol{W}_i\boldsymbol{x}) \cdot x_j(k)$$

式中，η为学习速率或步长（$0<\eta\leqslant 1$）；$f(\cdot)$为激活函数。这就是通常说的误差纠正学习规则（或称为δ 规则），用于控制每次误差修正值。它是基于使输出方差最小的思想而建立的。

2）Hebb 学习规则

神经心理学家 Hebb 提出的学习规则可归结为“当某一突触（连接）两端的神经元的激活同步（同为激活或同为抑制）时，该连接的强度应增加，反之则应减弱”，用数学方式可描述为

$$\Delta w_{ij}(k)=F(y_i(k),x_j(k))$$

式中，$y_i(k)$，$x_j(k)$分别为 w_{ij}两端神经元的状态，其中最常用的一种情况为

$$\Delta w_{ij}(k)=\eta\cdot y_i(k)\cdot x_j(k)$$

式中，η为学习速率。

由于 $w_{ij}(k)$与 $y_i(k)$、$x_j(k)$的相关成比例，故有时称之为相关学习规则。上式定义的 Hebb 规则实际上是一种无监督的学习规则，它不需要关于目标输出的任何相关信息。

原始的 Hebb 学习规则对权值矩阵的取值未做任何限制，因而学习后权值可取任意值。为了克服这一弊病，在 Hebb 学习规则的基础上增加一个衰减项，即

$$\Delta w_{ij}(k)=\eta\cdot y_i(k)\cdot x_j(k)-d_r*w_{ij}(k)$$

衰减项的加入能够增加网络学习的“记忆”功能，并且能够有效地对权值的取值加以限制。衰减系数 d_r的取值在[0,1]之间。当取 0 时，就变成原始的 Hebb 学习规则。

另外，Hebb 学习规则还可以采用有监督的学习，对于有监督学习的 Hebb 学习规则而言，是将目标输出代替实际输出。由此，算法被告知的就是网络应该做什么，而不是网络当前正在做什么，可描述为

$$\Delta w_{ij}(k)=\eta\cdot t_i(k)\cdot x_j(k)$$

3）竞争（Competitive）学习规则

顾名思义，在竞争学习时网络各输出单元互相竞争，最后达到只有一个最强者激活。最常见的一种情况是输出神经元之间有侧向抑制性连接，如图 1-9 所示。这样众多输出单元中如有某一单元较强，则它将获胜并抑制其他单元，最后只有比较强者处于激活状态。最常用的竞争学习规则有以下三种。

Kohonen 规则：$\Delta w_{ij}(k)=\begin{cases}\eta(x_j-w_{ij}), & \text{若神经元 } j\text{竞争获胜}\\ 0, & \text{若神经元 } j\text{竞争失败}\end{cases}$

Instar 规则：$\Delta w_{ij}(k)=\begin{cases}\eta y_i(x_j-w_{ij}), & \text{若神经元 } j\text{竞争获胜}\\ 0, & \text{若神经元 } j\text{竞争失败}\end{cases}$

Outstar 规则：$\Delta w_{ij}(k)=\begin{cases}\eta(y_i-w_{ij})/x_j, & \text{若神经元 } j\text{竞争获胜}\\ 0, & \text{若神经元 } j\text{竞争失败}\end{cases}$

3．学习与自适应

当学习系统所处环境平稳时（统计特征不随时间变化），从理论上说通过监督学习可以学到环境的统计特征，这些统计特征可被学习系统（神经网络）作为经验记住。如果环境是非平稳的（统计特征随时间变化），通常的监督学习没有能力跟踪这种变化，为解决此问题需要网络有一定的自适应能力，此时对每一个不同输入都作为一个新的例子对待，其工作过程如图 1-10 所示。此时模型（如 NN）被当作一个预测器，基于前一时刻输出 $x(k-1)$和模型在 $k-1$ 时刻的参数，它估计出 k 时刻的输出 $\hat{x}(k)$，$\hat{x}(k)$与实际值 $x(k)$（作为应有的正确答案）比

较，其差值 $e(k)$称为“新息”，如新息 $e(k)=0$，则不修正模型参数，否则应修正模型参数以便跟踪环境的变化。

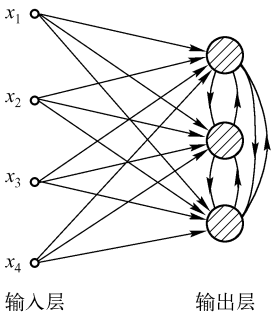


图 1-9　竞争学习网络

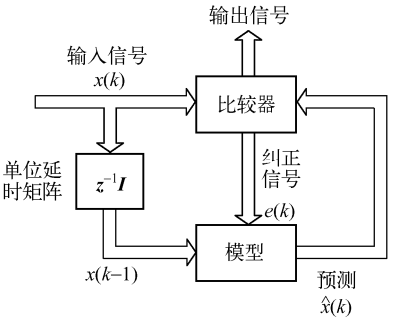


图 1-10　自适应学习框图

1.1.6　神经网络的分类

神经网络根据不同的情况，可按以下几方面进行分类。

（1）按功能分：连续型与离散型、确定型与随机型、静态与动态神经网络。

（2）按连接方式分：前馈（或称前向）型与反馈型神经网络。

（3）按逼近特性分：全局逼近型与局部逼近型神经网络。

（4）按学习方式分：有监督学习、无监督学习和强化学习神经网络。

1.2　典型神经网络的模型

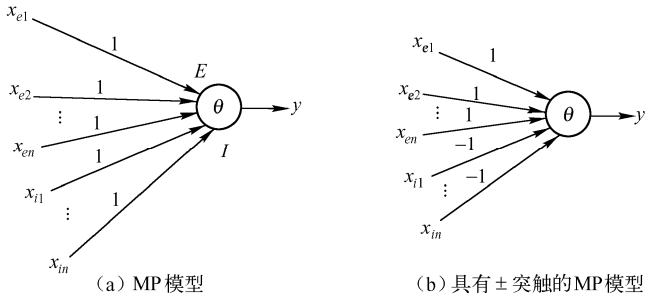
自 1957 年 F. Rosenblatt 在第一届人工智能会议上展示他构造的第一个人工神经网络模型以来，据统计到目前为止已有上百种神经网络问世。根据 HCC 公司及 IEEE 的调查统计，有十多种神经网络比较著名，以下按照神经网络的拓扑结构与学习算法相结合的方法，将神经网络的类型分为前馈网络、竞争网络、反馈网络和随机网络四大类，并按类介绍 MP 模型、感知机神经网络、自适应线性神经网络（Adaline）、BP 神经网络、径向基神经网络、自组织竞争神经网络、自组织特征映射神经网络（SOM）、反传神经网络（CPN）、自适应共振理论（ART）神经网络、学习向量量化（LVQ）神经网络、Elman 神经网络、Hopfield 神经网络和 Boltzmann 神经网络的特点、拓扑结构、工作原理和学习机理，以揭示神经网络所具有的功能和特征。运用这些神经网络模型可实现函数逼近、数据聚类、模式分类、优化计算等功能。因此，神经网络广泛应用于人工智能、自动控制、机器人、统计学等领域的信息处理中。

1.2.1　MP 模型

MP 模型最初是由美国心理学家 McCulloch 和数学家 Pitts 在 1943 年共同提出的，它是由固定的结构和权组成的，它的权分为兴奋性突触权和抑制性突触权两类，如抑制性突触权被激活，则神经元被抑制，输出为零。兴奋性突触权能否激活，则要看它的累加值是否大于一个阈值，大于该阈值神经元即兴奋。

早期提出的 MP 模型结构如图 1-11（a）所示，其中图 1-11（a）有以下关系式

$$E=\sum_{j=1}^{n}x_{ej},I=\sum_{k=1}^{n}x_{ik}$$



（a）MP模型　　（b）具有±突触的MP模型

图 1-11　MP 模型中单个神经元示意图

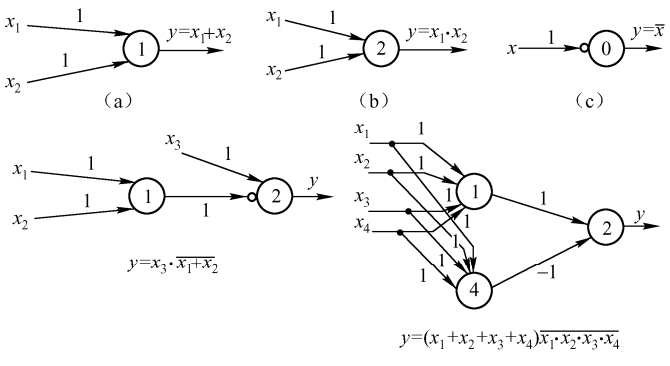
式中，$x_{ej}(j=1,2,\cdots,n)$为兴奋性突触的输入，$x_{ik}(k=1,2,\cdots,n)$为抑制性突触的输入。在图 1-11（a）中，模型的权值均为 1，它可以用来完成一些逻辑性关系，其输入输出关系为

$$y=\begin{cases}1 & (I=0,\ E\geqslant\theta)\\ 0 & (I=0,\ E<\theta)\\ 0 & (I>0)\end{cases}$$

如果兴奋与抑制突触用权±1 表示，而总的作用用加权的办法实现，兴奋为 1，抑制为 −1，则 MP 模型可表示成如图 1-11（b）所示的形式，则有

$$y=\begin{cases}1 & \left(\sum_{j=1}^{n}x_{ej}-\sum_{k=1}^{n}x_{ik}\geqslant\theta\right)\\ 0 & \left(\sum_{j=1}^{n}x_{ej}-\sum_{k=1}^{n}x_{ik}<\theta\right)\end{cases}$$

图 1-12（a）、（b）、（c）、（d）和（e）是利用 MP 模型分别表示的或、与、非以及一些逻辑关系式。



（a）　（b）　（c）　（d）　（e）

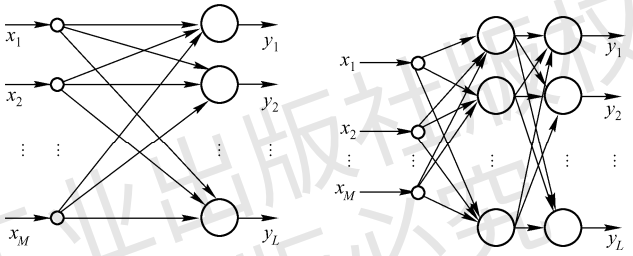
图 1-12　用 MP 模型实现的布尔逻辑

MP 模型的权值、输入和输出都是二值变量，这同由逻辑门组成的逻辑关系式的实现区别不大，又由于它的权值无法调节，因而现在很少有人单独使用。但它是人工神经元模型的基础，也是神经网络理论的基础。

1.2.2　感知机

1．感知机的网络结构

1957 年美国心理学家 Frank Rosenblatt 及其合作者为了研究大脑的存储、学习和认知过程而提出了一类神经网络模型，并称其为感知机（Perceptron）。感知机较 MP 模型又进一步，它的输入可以是非离散量，它的权不仅是非离散量，而且可以通过调整学习而得到。感知机可以对输入的样本矢量进行模式分类，而且多层的感知机，在某些样本点上对函数进行逼近，但感知机是一个线性阈值单元组成的网络，在结构和算法上都成为其他前馈网络的基础，尤其它对隐单元的选取比其他非线性阈值单元组成的网络容易分析，而对感知机的讨论，可以对其他网络的分析提供依据。由于感知机的权值可以通过学习调整而得到，因此它被认为是最早提出的一种神经网络模型。图 1-13 为感知机的两种结构。



（a）简单感知机　（b）两层感知机

图 1-13　感知机的结构

在这种模型中，输入模式 $\boldsymbol{x}=[x_1,x_2,\cdots,x_M]^{\mathrm{T}}$ 通过各输入端点分配给下一层的各节点，下一层可以是中间层（或称为隐含层），也可以是输出层，隐含层可以是一层也可以是多层，最后通过输出层节点得到输出模式 $\boldsymbol{y}=[y_1,y_2,\cdots,y_L]^{\mathrm{T}}$。在这类前馈网络中没有层内连接，也没有隔层的前馈连接。每一节点只能前馈连接到其下一层的所有节点。然而，对于含有隐含层的多层感知机，当时没有可行的训练方法，所以初期研究的感知机为一层感知机或称为简单感知机，通常就把它称为感知机。虽然简单感知机有其局限性，但人们对它进行了深入的研究，有关它的理论仍是研究其他网络模型的基础。

如果在输入层和输出层单元之间加入一层或多层处理单元，即可构成多层感知机，因而多层感知机由输入层、隐含层、输出层组成。隐含层的作用相当于特征检测器，提取输入模式中包含的有效特征信息，使输出单元所处理的模式是线性可分的。但需注意，多层感知机模型只允许一层连接权值可调，这是因为无法设计出一个有效的多层感知机学习算法。图 1-14 是一个两层感

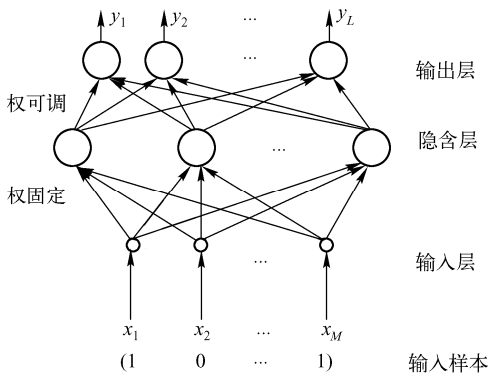


图 1-14　两层感知机的结构

知机结构（包括输入层、一个隐含层和一个输出层），有两层连接权，其中输入层和隐含层单元间的连接权值是随机设定的固定值，不可调节；隐含层与输出层单元间的连接权值是可调的。

值得注意的是，在神经网络中，由于输入层仅仅起输入信号的等值传输作用，而不对信号进行运算，故在定义多少层神经网络时，一般不把输入层计算在内，如上所述。也就是说，一般把隐含层称为神经网络的第一层，输出层称为神经网络的第二层（假如只有一个隐含层）。如果有两个隐含层，则第一个隐含层称为神经网络的第一层，第二个隐含层称为神经网络的第二层，而输出层称为神经网络的第三层。如果有多个隐含层，则以此类推。在 MATLAB 神经网络工具箱中的定义也类同。

对于具有 M 个输入、L 个输出的单层感知机，如图 1-13（a）所示。该网络通过一组权值 $w_{ij}(i=1,2,\cdots,L;\ j=1,2,\cdots,M)$与 L 个神经元组成。根据结构图，输出层的第 i 个神经元的输入总和和输出分别为

$$\text{net}_i=\sum_{j=1}^{M}w_{ij}x_j,\quad y_i=f\left(\text{net}_i-\theta_i\right)\qquad (i=1,2,\cdots,L) \tag{1-1}$$

式中，θ_i 为输出层神经元 i 的阈值；M 为输入层的节点数，即输入的个数；$f(\cdot)$为激活函数。

感知机中的激活函数使用了阶跃限幅函数，因此感知机能够将输入向量分为两个区域，即

$$f(x)=\begin{cases}1, & x\geqslant 0\\ 0, & x<0\end{cases}$$

2．感知机学习

感知机的学习是典型的有教师学习，可以通过样本训练达到学习的目的。训练的条件有两个：训练集和训练规则。感知机的训练集就是由若干个输入/输出模式对构成的一个集合，所谓输入/输出模式对是指一个输入模式及其期望输出模式所组成的向量对。它包括二进制值输入模式及其期望输出模式，每个输出对应一个分类。F. Rosenblatt 业已证明，如果两类模式是线性可分的（指存在一个超平面将它们分开），则算法一定收敛。

设有 N 个训练样本，在感知机训练期间，不断用训练集中的每个模式对训练网络。当给定某一个样本 p 的输入/输出模式对时，感知机输出单元会产生一个实际输出向量，用期望输出与实际的输出之差来修正网络连接权值。权值的修正采用简单的误差学习规则（即δ规则），它是一个有教师的学习过程，其基本思想是利用某个神经单元的期望输出与实际的输出之间的差来调整该神经单元与上一层中相应神经单元的连接权值，最终减小这种偏差。也就是说，神经单元之间连接权的变化正比于输出单元期望输出与实际的输出之差。

简单感知机输出层的任意神经元 i 的连接权值 w_{ij} 和阈值θ_i 修正公式为

$$\Delta w_{ij}=\eta(t_i^p-y_i^p)\cdot x_j^p=\eta e_i^p\cdot x_j^p\qquad (i=1,2,\cdots,L;\ j=1,2,\cdots,M) \tag{1-2}$$

$$\Delta\theta_i=\eta(t_i^p-y_i^p)\cdot 1=\eta e_i^p\qquad (i=1,2,\cdots,L) \tag{1-3}$$

式中，t_i^p 表示在样本 p 作用下的第 i 个神经元的期望输出；y_i^p 表示在样本 p 作用下的第 i 个神经元的实际输出；η为学习速率（$0<\eta\leqslant 1$），用于控制权值调整速度。学习速率η较大时，

学习过程加速，网络收敛较快。但是η太大时，学习过程变得不稳定，且误差会加大。因此学习速率的取值很关键。

感知机的学习规则属误差修正规则，该法已被证明，经过若干次迭代计算后，可以收敛到正确的目标向量。由上可知，该算法无须求导数，因此比较简单，又具有收敛速度快和精度高的优点。

期望输出与实际输出之差为

$$e_i^p = t_i^p - y_i^p(k) = \begin{cases} 1 & (t_i^p = 1, y_i^p(k) = 0) \\ 0 & (t_i^p = y_i^p(k)) \\ -1 & (t_i^p = 0, y_i^p(k) = 1) \end{cases} \tag{1-4}$$

由此可见，权值变化量与两个量有关：输入状态 x_j 和输出误差 e_i。当且仅当输出单元 i 有输出误差且相连输入状态 x_j 为 1 时，修正权值或增加一个量或减少一个量。

感知机的学习过程又称为最小方差学习过程。根据权向量分布，可以构造一个多维权空间，其中，每个权对应一个轴，另一个轴表示学习过程中的误差度量。对每个权向量都会有一定输出误差，由权空间某点的“高度”表示。学习过程中所有这些点形成的一个空间表面，称为误差表面。线性输出单元的感知机，其误差表面成一碗形，其水平截线为椭圆，垂直截线为抛物线。显然，该碗形表面只有一个极小点，沿误差表面按梯度下降法就能达到该点，这涉及感知机学习的收敛性，下面还要详细讨论。

3．感知机的线性可分性

感知机可以对线性可分性输入模式进行分类，例如，两维输入 x_1 和 x_2，其分界线为 n–1 维（2–1=l）直线，则 $w_1x_1+w_2x_2-\theta=0$。

根据式（1-1）可知，当且仅当 $w_1x_1+w_2x_2 \geqslant \theta$ 时，y=1，此时把输入模式划分为“1”类，用“•”代表输出为 1 的输入模式，即目标输出为 1 的两个输入向量用黑心圆圈“•”表示；当且仅当 $w_1x_1+w_2x_2<\theta$ 时，y=0，此时把输入模式划分为“0”类，用“o”代表输出为 0 的输入模式，即目标输出为 0 的两个输入向量用空心圆圈“o”表示，其对应的线性分割如图 1-15 所示。感知机对与、或、非问题均可以线性分割。感知机模式只能对线性输入模式进行分类，这是它的主要功能局限。

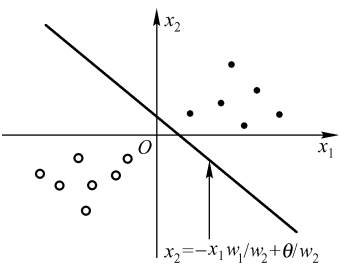


图 1-15　线性分割图

例 1-1　利用简单感知机对“与”、“或”和“异或”问题进行分类。

解：逻辑“与（AND）”、“或（OR）”和“异或（XOR）”的真值表如表 1-1 所示。

表 1-1　真值表

x_1	x_2	y		
		AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

对于逻辑“与”和逻辑“或”可将输入模式按照其输出分成两类：输出为 0 的属于“0”类，用“o”代表；输出为 1 的属于“1”类，用“•”代表，如图 1-16（a）、（b）所示。输入模式可以用一条决策直线划分为两类，即逻辑“与”和逻辑“或”是线性可分的。所以简单感知机可以解决逻辑“与”和逻辑“或”的问题。

对于逻辑“异或”，现仍然将输入模式按照其输出分成两类，即这四个输入模式分布在二维空间中，如图 1-16（c）所示。显然无法用一条决策直线把这四个输入模式分成两类，即逻辑“异或”是线性不可分的。所以简单感知机无法解决逻辑“异或”问题。

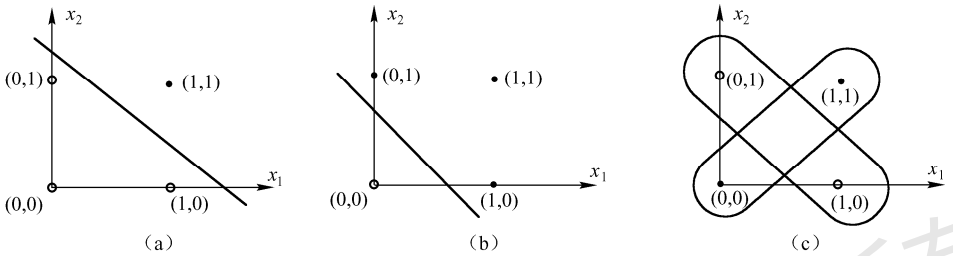


图 1-16　AND，OR 和 XOR 输入模式的空间分布

感知机为解决逻辑异或问题，可以设计一个多层的网络，即含有输入层、隐含层和输出层的结构。

可以证明，只要隐含层单元数足够多，用多层感知机可实现任何模型分类。但是，隐单元的状态不受外界直接控制，这给多层网络的学习带来极大困难。

4．感知机收敛性定理

定理 1.1　如果样本输入函数是线性可分的，那么感知机学习算法经过有限次迭代后可收敛到正确的权值或权向量。

定理 1.2　假定隐含层单元可以根据需要自由设置，那么用双隐含层感知机可以实现任意的二值逻辑函数。

5．感知机学习算法的计算步骤

（1）初始化：置所有的加权系数为最小的随机数。

（2）提供训练集：给出顺序赋值的输入向量 $\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)},\cdots,\boldsymbol{x}^{N}$ 和期望的输出向量 $\boldsymbol{t}^{(1)},\boldsymbol{t}^{(2)},\cdots,\boldsymbol{t}^{N}$。

（3）计算实际输出：按式（1-1）计算输出层各神经元的输出。

（4）按式（1-4）计算期望值与实际输出的误差。

（5）按式（1-2）和式（1-3）调整输出层的加权系数 w_{ij} 和阈值 θ_i。

（6）返回计算（3）步，直到误差满足要求为止。

例 1-2　建立一个感知机，使其能够完成“与”的功能。

解：为了完成“与”函数，建立一个两输入、单输出的一个感知机。根据真值表 1-1 可知，与门可提供四组 2 输入的训练样本，且每组训练样本都有 1 个与之对应的目标输出。将这四组训练样本和对应的目标输出分别组成 1 个 2×4 维的矩阵和 1 个 1×4 维的向量后，可知该感知机训练集的输入矩阵为：$\boldsymbol{X}=\begin{pmatrix}0&0&1&1\\0&1&0&1\end{pmatrix}$，目标向量为：$\boldsymbol{T}$=[0 0 0 1]。根据感知机学习算

法的计算步骤，利用 MATLAB 语言编写的程序如下。

```
%ex1_2.m
%感知机的第一阶段——学习期（训练加权系数Wij）
err_goal=0.001;lr=0.9; max_epoch=10000;    %给定期望误差最小值、学习速率和训练的最大次数
X=[0 0 1 1;0 1 0 1];T=[0 0 0 1];           %提供四组训练集（2输入1输出）
%初始化Wij（M-输入节点j的数量, L-输出节点i的数量, N-为训练集对数量）
[M,N]=size(X);[L,N]=size(T);Wij=rand(L,M);y=0;b=rand(L);
for epoch=1:max_epoch
    %计算各神经元输出
    NETi=Wij*X;
    for j=1:N
      for i=1:L
        if (NETi(i,j)>=b(i)) y(i,j)=1;else y(i,j)=0;end
      end
    end
    %计算误差函数
    E=(T-y);EE=0;
    for j=1:N;EE=EE+abs(E(j));end
    if (EE<err_goal) break;end
    Wij=Wij+lr*E*X'; b=b+sqrt(EE);         %调整输出层加权系数和输出层神经元的阈值
end
epoch,Wij,b                                %显示计算次数、加权系数和阈值
%感知机的第二阶段——工作期（根据训练好的Wij和给定的输入计算输出）
X1=X;                                      %给定输入
%计算各神经元输出
NETi=Wij*X1; [M,N]=size(X1);
for j=1:N
  for i=1:L
    if (NETi(i,j)>=b(i)) y(i,j)=1;else y(i,j)=0;end
  end
end
y                                          %显示网络的输出
```

结果显示：

```
Wij =
    2.6462    2.3252
b =
    4.6169
y =
    0    0    0    1
```

1.2.3　自适应线性神经网络

线性神经网络是一种简单的神经元网络，它可以由一个或多个线性神经元构成。1962 年由美国斯坦福大学教授 Berhard Widrow 提出的自适应线性元件网络（Adaptive Linear Element，Adaline）是线性神经网络最早的典型代表，它是一个由输入层和输出层构成的单层

前馈型网络。它与感知机的不同之处在于其每个神经元的传输函数为线性函数，因此自适应线性神经网络的输出可以取任意值，而感知机的输出只能是 1 或 0。线性神经网络采用由 Berhard Widrow 和 Marcian Hoff 共同提出的一种新的学习规则，也称为 Widrow-Hoff 学习规则，或者 LMS（Least Mean Square）算法来调整网络的权值和阈值。自适应线性神经网络的学习算法比感知机的学习算法的收敛速度和精度都有较大的提高。自适应线性神经网络主要用于函数逼近、信号预测、系统辨识、模式识别和控制等领域。

1．自适应线性神经网络结构

自适应线性神经网络结构同感知机，不同之处在于其每个神经元的传输函数为线性函数。对于具有 M 个输入、L 个输出的线性神经网络。输出层的第 i 个神经元的输入总和与输出分别为

$$\text{net}_i = \sum_{j=1}^{M} w_{ij} x_j, \quad y_i = f(\text{net}_i - \theta_i) \qquad (i=1,2,\cdots,L) \tag{1-5}$$

式中，θ_i 为输出层神经元 i 的阈值；M 为输入层的节点数，即输入的个数；$f(\cdot)$为激活函数，它为线性函数的传输函数。

2．自适应线性神经网络的学习

自适应线性神经网络的学习也是典型的有教师学习，采用 Widrow-Hoff 学习规则，即 LMS 学习规则。在训练期间，不断用训练集中的每个模式对训练网络。当给定某一训练模式时，输出单元会产生一个实际输出向量，用期望输出与实际的输出之差来修正网络连接权值。

在训练网络的学习阶段，设有 N 个训练样本，先假定用其中的某一个样本 p 的输入/输出模式对$\{\boldsymbol{x}^p\}$和$\{\boldsymbol{t}^p\}$对网络进行训练，输出层的第 i 个神经元在样本 p 作用下的输入为

$$\text{net}_i^p = \sum_{j=1}^{M} w_{ij} x_j^p \qquad (i=1,2,\cdots,L)$$

式中，θ_i 为输出层神经元 i 的阈值；M 为输入层的节点数，即输入的个数。

输出层第 i 个神经元的输出为

$$y_i^p = f(\text{net}_i^p - \theta_i) \qquad (i=1,2,\cdots,L)$$

式中，$f(\cdot)$为线性激活函数。它将网络的输入原封不动地输出，因此有

$$y_i^p = \text{net}_i^p - \theta_i = \sum_{j=1}^{M} w_{ij} x_j^p - \theta_i \qquad (i=1,2,\cdots,L) \tag{1-6}$$

若对于每一样本 p 的输入模式对的二次型误差函数为

$$J_p = \frac{1}{2}\sum_{i=1}^{L}(t_i^p - y_i^p)^2 = \frac{1}{2}\sum_{i=1}^{L} e_i^2 \tag{1-7}$$

则系统对所有 N 个训练样本的总误差函数为

$$J = \sum_{p=1}^{N} J_p = \frac{1}{2}\sum_{p=1}^{N}\sum_{i=1}^{L}(t_i^p - y_i^p)^2 = \frac{1}{2}\sum_{p=1}^{N}\sum_{i=1}^{L} e_i^2 \tag{1-8}$$

式中，N 为模式样本对数；L 为网络输出节点数；t_i^p 表示在样本 p 作用下的第 i 个神经元的期望输出；y_i^p 表示在样本 p 作用下的第 i 个神经元的实际输出。

自适应线性神经网络加权系数修正采用 Widrow-Hoff 学习规则，又称为最小均方误差算法（LMS）。它的实质是利用梯度最速下降法，是权值沿误差函数的负梯度方向改变。Widrow-Hoff 学习规则的权值变化量正比于网络的输出误差及网络的输入矢量。根据梯度法，可得输出层的任意神经元 i 的加权系数修正公式为

$$\Delta w_{ij} = -\eta \frac{\partial J_p}{\partial w_{ij}} \qquad (i=1,2,\cdots,L;\ j=1,2,\cdots,M)$$

式中，学习速率 $\eta = \dfrac{\alpha}{\left\| \boldsymbol{x}^p \right\|_2^2}$，$\alpha$ 为常值，当 $0<\alpha<2$ 时，可使算法收敛。η 随着输入样本 $\boldsymbol{x}^p$ 自适应地调整。

因为

$$\frac{\partial J_p}{\partial w_{ij}} = \frac{\partial J_p}{\partial \mathrm{net}_i^p} \cdot \frac{\partial \mathrm{net}_i^p}{\partial w_{ij}} = \frac{\partial J_p}{\partial \mathrm{net}_i^p} \cdot \frac{\partial}{\partial w_{ij}} \left(\sum_{j=1}^{M} w_{ij} x_j^p \right) = \frac{\partial J_p}{\partial \mathrm{net}_i^p} \cdot x_j^p$$

定义 $\delta_i^p = -\dfrac{\partial J_p}{\partial \mathrm{net}_i^p} = -\dfrac{\partial J_p}{\partial y_i^p} \cdot \dfrac{\partial y_i^p}{\partial \mathrm{net}_i^p} = (t_i^p - y_i^p) \cdot f'(\mathrm{net}_i^p - \theta_i) = e_i \cdot f'(\mathrm{net}_i^p - \theta_i)$

则

$$\Delta w_{ij} = \eta \cdot \delta_i^p \cdot x_j^p$$

由于激活函数 $f(\cdot)$为线性函数，故可令 $f'(\mathrm{net}_i^p - \theta_i) = 1$。

所以

$$\frac{\partial J_p}{\partial w_{ij}} = -e_i \cdot x_j^p$$

输出层的任意神经元 i 的加权系数修正公式为

$$\Delta w_{ij} = \eta \cdot (t_i^p - y_i^p) \cdot x_j^p = \eta \cdot e_i \cdot x_j^p \qquad (1\text{-}9)$$

同理，阈值θ_i的修正公式为

$$\Delta \theta_i = \eta (t_i - y_i) = \eta e_i \qquad (1\text{-}10)$$

以上两式构成了最小均方误差算法（LMS）或 Widrow-Hoff 学习算法，它实际上也是δ学习规则的一种特例。

3．自适应线性神经网络学习算法的计算步骤

（1）初始化：置所有的加权系数为最小的随机数。

（2）提供训练集：给出顺序赋值的输入向量 $\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)},\cdots,\boldsymbol{x}^{N}$ 和期望的输出向量 $\boldsymbol{t}^{(1)},\boldsymbol{t}^{(2)},\cdots,\boldsymbol{t}^{N}$。

（3）计算实际输出：按式（1-5）和式（1-6）计算输出层各神经元的输出。

（4）按式（1-7）或式（1-8）计算期望值与实际输出的误差。

（5）按式（1-9）和式（1-10）调整输出层的加权系数 w_{ij} 和阈值θ_i。

（6）返回计算（3）步，直到误差满足要求为止。

1.2.4　BP 神经网络

1986 年 D.E.Rumelhart 和 J.L.McClelland 提出了一种利用误差反向传播训练算法的神经网络，简称 BP（Back Propagation）网络，是一种有隐含层的多层前馈网络，系统地解决了

多层网络中隐含单元连接权的学习问题。

如果网络的输入节点数为 M、输出节点数为 L，则此神经网络可看成从 M 维欧氏空间到 L 维欧氏空间的映射。这种映射是高度非线性的。其主要用于以下几方面。

（1）模式识别与分类：用于语言、文字、图像的识别，医学特征的分类和诊断等。

（2）函数逼近：用于非线性控制系统的建模、机器人的轨迹控制及其他工业控制等。

（3）数据压缩：编码压缩和恢复，图像数据的压缩和存储，以及图像特征的抽取等。

1．BP 算法原理

BP 学习算法的基本原理是梯度最速下降法，它的中心思想是调整权值使网络总误差最小。也就是采用梯度搜索技术，以期使网络的实际输出值与期望输出值的误差均方值为最小。网络学习过程是一种误差边向后传播边修正权系数的过程。

多层网络运用 BP 学习算法时，实际上包含了正向和反向传播两个阶段。在正向传播过程中，输入信息从输入层经隐含层逐层处理，并传向输出层，每一层神经元的状态只影响下一层神经元的状态。如果在输出层不能得到期望输出，则转入反向传播，将误差信号沿原来的连接通道返回，通过修改各层神经元的权值，使误差信号最小。

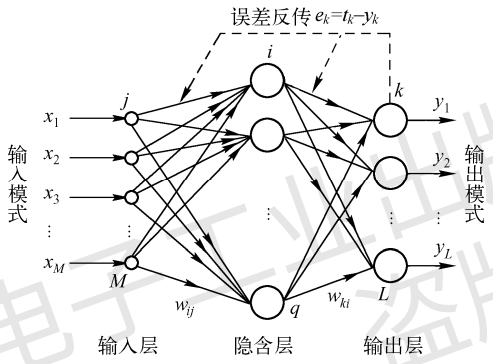


图 1-17　BP 网络的结构图

将一层节点的输出传送到另一层时，通过调整连接权系数 w_{ij} 来达到增强或削弱这些输出的作用。除了输入层的节点外，隐含层和输出层节点的净输入是前一层节点输出的加权和。每个节点的激活程度由它的输入信号、激活函数和节点的偏值（或阈值）来决定。但对于输入层，输入模式送到输入层节点上，这一层节点的输出即等于其输入。

注意，这种网络没有反馈存在，实际运行仍是单向的，所以不能将其看成一非线性动力学系统，而只是一种非线性映射关系。具有隐含层 BP 网络的结构如图 1-17 所示，图中设有 M 个输入节点，L 个输出节点，网络的隐含层共有 q 个神经元。其中，$x_1,x_2,\cdots,x_M$ 为网络的实际输入，$y_1,y_2,\cdots,y_L$ 为网络的实际输出，$t_k(k=1,2,\cdots,L)$为网络的目标输出，$e_k(k=1,2,\cdots,L)$为网络的输出误差。

BP 网络结构与多层感知机结构相比，二者是类似的，但差异也是显著的。首先，多层感知机结构中只有一层权值可调，其他各层权值是固定的、不可学习的；BP 网络的每一层连接权值都可通过学习来调节。其次，感知机结构中的处理单元采用了阶跃限幅函数，其单元输出为二进制数的 0 或 1；而 BP 网络的基本处理单元（输入层除外）为非线性的输入/输出关系，一般选用 S 型函数，其单元输出可为 0～1 之间的任意值。

2．BP 网络的前馈计算

在训练网络的学习阶段，设有 N 个训练样本，先假定用其中的某一个样本 p 的输入/输出模式对$\{\boldsymbol{x}^p\}$和$\{\boldsymbol{t}^p\}$对网络进行训练，隐含层的第 i 个神经元在样本 p 作用下的输入为

$$\text{net}_i^p=\sum_{j=1}^{M}w_{ij}o_j^p=\sum_{j=1}^{M}w_{ij}x_j^p\quad(i=1,2,\cdots,q)\tag{1-11}$$

式中，x_j^p 和 o_j^p 分别为输入节点 j 在样本 p 作用时的输入和输出，对输入节点而言两者相当；w_{ij} 为输入层神经元 j 与隐含层神经元 i 之间的连接权值；M 为输入层的节点数，即输入的个数。

隐含层第 i 个神经元的输出为

$$o_i^p = g(\text{net}_i^p - \theta_i) \qquad (i=1,2,\cdots,q) \tag{1-12}$$

式中，$g(\cdot)$为隐含层的激活函数；θ_i 为隐含层神经元 i 的阈值。

对于 Sigmoid 型激活函数，

$$g(x) = \frac{1}{1+\exp[-(x+\theta_1)/\theta_0]}$$

式中，参数θ_1 表示偏值，正的θ_1 使激活函数水平向左移动。θ_0 的作用是调节 Sigmoid 函数形状的，较小的θ_0 使 Sigmoid 函数逼近一个阶跃限幅函数，而较大的θ_0 将使 Sigmoid 函数变得较为平坦，如图 1-18 所示。

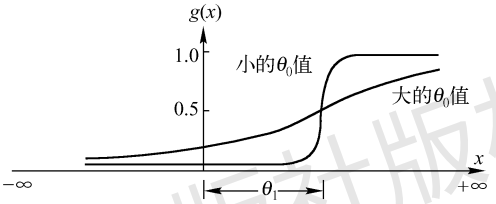


图 1-18　具有偏值和形状调节的 Sigmoid 函数

隐含层激活函数 $g(\text{net}_i^p - \theta_i)$的微分函数为

$$g'(\text{net}_i^p - \theta_i) = g(\text{net}_i^p - \theta_i)[1 - g(\text{net}_i^p - \theta_i)] = o_i^p(1-o_i^p) \quad (i=1,2,\cdots,q) \tag{1-13}$$

隐含层第 i 个神经元的输出 o_i^p 将通过权系数向前传播到输出层第 k 个神经元作为它的输入之一，而输出层第 k 个神经元的总输入为

$$\text{net}_k^p = \sum_{i=1}^{q} w_{ki} o_i^p \qquad (k=1,2,\cdots,L) \tag{1-14}$$

式中，w_{ki} 为隐含层神经元 i 与输出层神经元 k 之间的连接权值；q 为隐含层的节点数。

输出层的第 k 个神经元的实际输出为

$$o_k^p = g(\text{net}_k^p - \theta_k) \qquad (k=1,2,\cdots,L) \tag{1-15}$$

式中，$g(\cdot)$ 为输出层的激活函数；θ_k 为输出层神经元 k 的阈值。

输出层激活函数 $g(\text{net}_k^p - \theta_k)$的微分函数为

$$g'(\text{net}_k^p - \theta_k) = g(\text{net}_k^p - \theta_k)[1 - g(\text{net}_k^p - \theta_k)] = o_k^p(1-o_k^p) \quad (k=1,2,\cdots,L) \tag{1-16}$$

若其输出与给定模式对的期望输出 t_k^p 不一致，则将其误差信号从输出端反向传播回来，并在传播过程中对加权系数不断修正，直到在输出层神经元上得到所需要的期望输出值 t_k^p 为止。对样本 p 完成网络权系数的调整后，再送入另一样本模式对进行类似学习，直到完成 N 个样本的训练学习为止。

3．BP 网络权系数的调整规则

对于每一样本 p 的输入模式对的二次型误差函数为

$$J_p=\frac{1}{2}\sum_{k=1}^{L}(t_k^p-o_k^p)^2 \tag{1-17}$$

则系统对所有 N 个训练样本的总误差函数为

$$J=\sum_{p=1}^{N}J_p=\frac{1}{2}\sum_{p=1}^{N}\sum_{k=1}^{L}(t_k^p-o_k^p)^2 \tag{1-18}$$

式中，N 为模式样本对数；L 为网络输出节点数。

一般来说，基于 J_p 还是基于 J 来完成加权系数空间的梯度搜索会获得不同的结果。在 Rumelhart 等人的学习加权的规则中，学习过程按使误差函数 J_p 减小最快的方向调整加权系数直到获得满意的加权系数集为止。这里加权系数的修正是顺序操作的，网络对各模式对一个一个地顺序输入并不断进行学习，类似于生物神经网络的处理过程，但不是真正的梯度搜索过程。系统最小误差的真实梯度搜索方法是基于式（1-18）的最小化方法。

1）输出层权系数的调整

权系数应按 J_p 函数梯度变化的反方向调整，使网络逐渐收敛。根据梯度法，可得输出层每个神经元权系数的修正公式为

$$\Delta w_{ki}=-\eta\frac{\partial J_p}{\partial w_{ki}}=-\eta\frac{\partial J_p}{\partial \mathrm{net}_k^p}\cdot\frac{\partial \mathrm{net}_k^p}{\partial w_{ki}}=-\eta\frac{\partial J_p}{\partial \mathrm{net}_k^p}\cdot\frac{\partial}{\partial w_{ki}}\left(\sum_{i=1}^{q}w_{ki}o_i^p\right)=-\eta\frac{\partial J_p}{\partial \mathrm{net}_k^p}\cdot o_i^p$$

式中，η为学习速率，η>0。

定义 $$\delta_k^p=-\frac{\partial J_p}{\partial \mathrm{net}_k^p}=-\frac{\partial J_p}{\partial o_k^p}\cdot\frac{\partial o_k^p}{\partial \mathrm{net}_k^p}=(t_k^p-o_k^p)\cdot g'(\mathrm{net}_k^p-\theta_k)=(t_k^p-o_k^p)o_k^p(1-o_k^p) \tag{1-19}$$

因此输出层的任意神经元 k 的加权系数修正公式为

$$\Delta w_{ki}=\eta\delta_k^p o_i^p=\eta o_k^p(1-o_k^p)(t_k^p-o_k^p)o_i^p \tag{1-20}$$

式中，o_k^p 为输出节点 k 在样本 p 作用时的输出；o_i^p 为隐含节点 i 在样本 p 作用时的输出；t_k^p 为在样本 p 输入/输出对作用时输出节点 k 的目标值。

2）隐含层权系数的调整

根据梯度法，可得隐含层每个神经元权系数的修正公式为

$$\Delta w_{ij}=-\eta\frac{\partial J_p}{\partial w_{ij}}=-\eta\frac{\partial J_p}{\partial \mathrm{net}_i^p}\cdot\frac{\partial \mathrm{net}_i^p}{\partial w_{ij}}=-\eta\frac{\partial J_p}{\partial \mathrm{net}_i^p}\cdot\frac{\partial}{\partial w_{ij}}\left(\sum_{j=1}^{M}w_{ij}o_j^p\right)=-\eta\frac{\partial J_p}{\partial \mathrm{net}_i^p}\cdot o_j^p$$

式中，η为学习速率，η>0。

定义 $$\delta_i^p=-\frac{\partial J_p}{\partial \mathrm{net}_i^p}=-\frac{\partial J_p}{\partial o_i^p}\cdot\frac{\partial o_i^p}{\partial \mathrm{net}_i^p}=-\frac{\partial J_p}{\partial o_i^p}\cdot g'(\mathrm{net}_i^p-\theta_i)=-\frac{\partial J_p}{\partial o_i^p}\cdot o_i^p(1-o_i^p)$$

由于隐含层一个单元输出的改变会影响与该单元相连接的所有输出单元的输入，即

$$-\frac{\partial J_p}{\partial o_i^p}=-\sum_{k=1}^{L}\frac{\partial J_p}{\partial \mathrm{net}_k^p}\cdot\frac{\partial \mathrm{net}_k^p}{\partial o_i^p}=-\sum_{k=1}^{L}\frac{\partial J_p}{\partial \mathrm{net}_k^p}\cdot\frac{\partial}{\partial o_i^p}\left(\sum_{i=1}^{q}w_{ki}o_i^p\right)$$

$$=\sum_{k=1}^{L}\left(-\frac{\partial J_p}{\partial \mathrm{net}_k^p}\right)\cdot w_{ki}=\sum_{k=1}^{L}\delta_k^p\cdot w_{ki}$$

则
$$\delta_i^p = \left(\sum_{k=1}^{L} \delta_k^p \cdot w_{ki}\right) o_i^p (1 - o_i^p) \tag{1-21}$$

因此隐含层的任意神经元 i 的加权系数修正公式为

$$\Delta w_{ij} = \eta \delta_i^p o_j^p = \eta \left(\sum_{k=1}^{L} \delta_k^p \cdot w_{ki}\right) o_i^p (1 - o_i^p) o_j^p \tag{1-22}$$

式中，o_i^p 为隐含节点 i 在样本 p 作用时的输出；o_j^p 为输入节点 j 在样本 p 作用时的输出，即输入节点 j 的输入（因对输入节点两者相当）。

输出层的任意神经元 k 在样本 p 作用时的加权系数增量公式为

$$w_{ki}(k+1) = w_{ki}(k) + \eta \delta_k^p o_i^p \tag{1-23}$$

隐含层的任意神经元 i 在样本 p 作用时的加权系数增量公式为

$$w_{ij}(k+1) = w_{ij}(k) + \eta \delta_i^p o_j^p \tag{1-24}$$

由式（1-23）和式（1-24）可知，对于给定的某一个样本 p，根据误差要求调整网络的加权系数使其满足要求；对于给定的另一个样本，再根据误差要求调整网络的加权系数使其满足要求，直到所有样本作用下的误差都满足要求为止。这种计算过程称为在线学习过程。

如果学习过程按使误差函数 J 减小最快的方向调整加权系数，采用类似的推导过程可得输出层和隐含层的任意神经元 k 和 i 在所有样本作用时的加权系数增量公式为

$$w_{ki}(k+1) = w_{ki}(k) + \eta \sum_{p=1}^{N} \delta_k^p o_i^p \tag{1-25}$$

$$w_{ij}(k+1) = w_{ij}(k) + \eta \sum_{p=1}^{N} \delta_i^p o_j^p \tag{1-26}$$

式中，δ_k^p 和 δ_i^p 的计算方法同上，即

$$\delta_k^p = o_k^p (1 - o_k^p)(t_k^p - o_k^p) \tag{1-27}$$

$$\delta_i^p = o_i^p (1 - o_i^p) \left(\sum_{k=1}^{L} \delta_k^p \cdot w_{ki}\right) \tag{1-28}$$

根据式（1-25）和式（1-26）所得加权系数的修正是在所有样本输入后，计算完总的误差后进行的，这种计算过程称为离线学习过程。离线学习可保证其总误差 J 向减小的方向变化，在样本多的时候，它比在线学习的收敛速度快。

因此，BP 网络的学习可采用两种方式，即在线学习和离线学习。在线学习是对训练集内每个模式对逐一更新网络权值的一种学习方式，其特点是学习过程中需要较少的存储单元，但有时会增加网络的整体输出误差，以上推导即为在线学习过程。因此，使用在线学习时一般使学习因子足够小，以保证用训练集内每个模式训练一次后，权值的总体变化充分接近于最快速下降。所谓离线学习也称为批处理学习，是指用训练集内所有模式依次训练网络，累加各权值修正量并统一修正网络权值的一种学习方式，它能使权值变化沿最快速下降方向进行。其缺点是学习过程中需要较多的存储单元，好处是学习速度较快。具

体实际应用中，当训练模式很多时，可以将整个训练模式分成若干组，采用分组批处理学习方式。

4．BP 网络学习算法的计算步骤

（1）初始化：置所有的加权系数为最小的随机数。

（2）提供训练集：给出顺序赋值的输入向量 $\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)},\cdots,\boldsymbol{x}^{N}$ 和期望的输出向量 $\boldsymbol{t}^{(1)},\boldsymbol{t}^{(2)},\cdots,\boldsymbol{t}^{N}$。

（3）计算实际输出：按式（1-11）～式（1-16）计算隐含层、输出层各神经元的输出。

（4）按式（1-17）或式（1-18）计算期望值与实际输出的误差。

（5）按式（1-23）或式（1-25）调整输出层的加权系数 w_{ki}。

（6）按式（1-24）或式（1-26）调整隐含层的加权系数 w_{ij}。

（7）返回计算（3）步，直到误差满足要求为止。

5．使用 BP 算法时应注意的几个问题

（1）学习速率 η 的选择非常重要。在学习初始阶段，η 选得大些可使学习速度加快，但当逼近最佳点时，η 必须相当小，否则加权系数将产生反复振荡而不能收敛。采用变学习速率方案，令学习速率 η 随着学习的进展而逐步减小，可收到较好的效果。引入惯性系数 a 的办法，也可使收敛速度加快，a 的取值可选在 0.9 左右。

（2）采用 S 型激活函数时，由于输出层各神经元的理想输出值只能接近于 1 或 0，而不能达到 1 或 0。这样在设置各训练样本的期望输出分量 t_k^p 时，不能设置为 1 或 0，以设置为 0.9 或 0.1 较为适宜。

（3）由图 1-18 可知，S 型非线性激活函数 $g(x)$ 随着 $|x|$ 的增大梯度下降，即 $|g'(x)|$ 减小并趋于 0，不利于权值的调整，因此希望 $|x|$ 工作在较小的区域，故应考虑网络的输入。若实际问题给以网络的输入量较大，需做归一化处理，网络的输出也要进行相应的处理。对于具体问题，需经调试而定，且需经验知识的积累。

（4）学习开始时如何设置加权系数 w_{ij} 和 w_{ki} 的初值非常重要，如将所有初值设置为相等值，则由式（1-22）可知，所有隐含层加权系数的调整量相同，从而使这些加权系数总相等，因此各加权系数的初值以设置为随机数为宜。

（5）BP 网络的另一个问题是学习过程中系统可能陷入某些局部最小值，或某些静态点，或在这些点之间振荡。在这种情况下，不管进行多少次迭代，系统都存在很大误差。因此，在学习过程中，应尽量避免落入某些局部最小值点上，引入惯性项有可能使网络避免落入某一局部最小值。

6．BP 网络的特点

BP 网络总括起来，具有以下主要优点：

（1）只要有足够多的隐含层和隐节点，BP 网络可以逼近任意的非线性映射关系；

（2）BP 网络的学习算法属于全局逼近的方法，因而它具有较好的泛化能力。

它的主要缺点是：

（1）收敛速度慢；

（2）局部极值；

（3）难以确定隐含层和隐节点的个数。

从原理上，只要有足够多的隐含层和隐节点，即可实现复杂的映射关系，但是如何根据

特定的问题来具体确定网络的结构尚无很好的方法，仍需要凭借经验和试凑。

BP 网络能够实现输入/输出的非线性映射关系，但它并不依赖于模型。其输入与输出之间的关联信息分散地存储于连接权中。由于连接权的个数很多，个别神经元的损坏只对输入/输出关系有较小的影响，因此 BP 网络显示了较好的容错性。

7．BP 网络学习算法的改进

BP 网络由于其很好的逼近非线性映射的能力，因而它可应用于信息处理、图像识别、模型辨识、系统控制等多个方面。对于控制方面的应用，其很好的逼近特性和泛化能力是一个优点。而收敛速度慢却是一个很大的缺点，这一点难以满足具有适应功能的实时控制的要求，它影响了该网络在许多方面的实际应用。为此，许多人对 BP 网络的学习算法进行了广泛的研究，提出了许多改进的算法，下面介绍典型的几种。

1）引入惯性项

有时为了使收敛速度快些，可在加权系数修正公式中增加一个惯性项，使加权系数变化更平稳些。

输出层的任意神经元 k 在样本 p 作用时的加权系数增量公式为

$$w_{ki}(k+1)=w_{ki}(k)+\eta\delta_k^p o_i^p+\alpha[w_{ki}(k)-w_{ki}(k-1)] \tag{1-29}$$

隐含层的任意神经元 i 在样本 p 作用时的加权系数增量公式为

$$w_{ij}(k+1)=w_{ij}(k)+\eta\delta_i^p o_j^p+\alpha[w_{ij}(k)-w_{ij}(k-1)] \tag{1-30}$$

式中，α为惯性系数，$0<\alpha<1$。

2）引入动量项

上述标准 BP 算法实质上是一种简单的最速下降静态寻优算法，在修正 $\boldsymbol{w}(k)$时，只是按 k 时刻的负梯度方式进行修正，而没有考虑以前积累的经验，即以前时刻的梯度方向，从而常常使学习过程发生振荡，收敛缓慢。为此，有人提出了如下的改进算法，即

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)+\eta[(1-\alpha)\boldsymbol{D}(k)+\alpha\boldsymbol{D}(k-1)] \tag{1-31}$$

式中，$\boldsymbol{w}(k)$既可表示单个的连接权系数，也可表示连接权向量（其元素为连接权系数）；$\boldsymbol{D}(k)=-\partial\boldsymbol{J}/\partial\boldsymbol{w}(k)$为 k 时刻的负梯度；$\boldsymbol{D}(k-1)$是 $k-1$ 时刻的负梯度；η为学习速率，$\eta>0$；α为动量项因子，$0\leqslant\alpha<1$。

该方法所加入的动量项实质上相当于阻尼项，它减小了学习过程的振荡趋势，改善了收敛性，这是目前应用比较广泛的一种改进算法。

3）变尺度法

标准的 BP 学习算法所采用的是一阶梯度法，因而收敛较慢。若采用二阶梯度法，则可以大大改善收敛性。二阶梯度法的算法为

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)-\eta[\Delta^2\boldsymbol{J}(k)]^{-1}\Delta\boldsymbol{J}(k)$$

式中，$\Delta\boldsymbol{J}(k)=\dfrac{\partial\boldsymbol{J}}{\partial\boldsymbol{w}(k)},\Delta^2\boldsymbol{J}(k)=\dfrac{\partial^2\boldsymbol{J}}{\partial\boldsymbol{w}^2(k)},0<\eta\leqslant1$。

虽然二阶梯度法具有比较好的收敛性，但是它需要计算 $\boldsymbol{J}$ 对 $\boldsymbol{w}$ 的二阶导数，这个计算量是很大的。所以一般不直接采用二阶梯度法，而常常采用变尺度法或共轭梯度法，它们具有二阶梯度法收敛较快的优点，而又无须直接计算二阶梯度。下面具体给出变尺度法的算法，即

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)+\eta\boldsymbol{H}(k)\boldsymbol{D}(k)$$

$$\boldsymbol{H}(k)=\boldsymbol{H}(k-1)-\frac{\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^{\mathrm{T}}(k)}{\Delta\boldsymbol{w}^{\mathrm{T}}(k)\Delta\boldsymbol{D}(k)}-\frac{\boldsymbol{H}(k-1)\Delta\boldsymbol{D}(k)\Delta\boldsymbol{D}^{\mathrm{T}}(k)\boldsymbol{H}(k-1)}{\Delta\boldsymbol{D}^{\mathrm{T}}(k)\boldsymbol{H}(k-1)\Delta\boldsymbol{D}(k)}$$

$$\Delta\boldsymbol{w}(k)=\boldsymbol{w}(k)-\boldsymbol{w}(k-1)$$

$$\Delta\boldsymbol{D}(k)=\boldsymbol{D}(k)-\boldsymbol{D}(k-1)$$

4）变步长法

一阶梯度法寻优收敛较慢的一个重要原因是η（学习速率）不好选择。η选得太小，收敛太慢，选得太大，则有可能修正过头，导致振荡甚至发散。为了解决这个问题提出了如下的变步长算法，即

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)+\eta(k)\boldsymbol{D}(k)$$

$$\eta(k)=2^{\lambda}\eta(k-1)$$

$$\lambda=\mathrm{sgn}[\boldsymbol{D}(k)\boldsymbol{D}(k-1)]$$

以上公式说明，当连续两次迭代其梯度方向相同时，表明下降速度太慢，这时可使步长加倍；当连续两次迭代其梯度方向相反时，表明下降速度过快，这时可使步长减半。需要引入动量项时，上述算法可修正改为

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)+\eta(k)[(1-\eta)\boldsymbol{D}(k)+\eta\boldsymbol{D}(k-1)]$$

$$\eta(k)=2^{\lambda}\eta(k-1)$$

$$\lambda=\mathrm{sgn}[\boldsymbol{D}(k)\boldsymbol{D}(k-1)]$$

在使用该算法时，由于步长在迭代过程中自适应进行调整，因此对于不同的连接权系数实际采用了不同的学习速率，也就是说误差代价函数 $\boldsymbol{J}$ 在超曲面上在不同的方向按照各自比较合理的步长向极小点逼近。

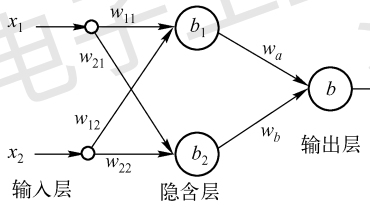


图 1-19　简单 BP 网络

例 1-3　一个具有两个输入单元，两个隐含单元和一个输出单元的两层 BP 神经网络，如图 1-19 所示，设学习样本为 N=4，写出 BP 网络学习算法批处理（离线学习）的计算步骤。假设样本集的输入为 $\boldsymbol{X}=\{\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)},\boldsymbol{x}^{(3)},\boldsymbol{x}^{(4)}\}$，目标为 $\boldsymbol{T}=\{t^{(1)},t^{(2)},t^{(3)},t^{(4)}\}$。$\boldsymbol{x}^p=[x_1^p;\ x_2^p]$ (p=1,2,3,4)。

解：（1）置所有的加权系数及偏值的初始值为最小的随机数，即

隐含层的加权系数及偏值的初始值为：$w_{11}(0),w_{12}(0),w_{21}(0),w_{22}(0),b_1(0),b_2(0)$

输出层的加权系数及偏值的初始值为：$w_a(0),w_b(0),b(0)$

（2）根据式（1-12）和式（1-16）分别计算样本集中所有样本的隐含层和输出层各节点的输出值，即

隐含层第 1 个神经元的输出为：$o_1^p=g(w_{11}\cdot x_1^p+w_{12}\cdot x_2^p+b_1)$　　(p=1,2,3,4)

隐含层第 2 个神经元的输出为：$o_2^p=g(w_{21}\cdot x_1^p+w_{22}\cdot x_2^p+b_2)$　　(p=1,2,3,4)

输出层神经元的输出为：$y^p=g(w_a\cdot o_1^p+w_b\cdot o_2^p+b)$　　(p=1,2,3,4)

（3）根据式（1-21）和式（1-22）及目标值分别计算在所有样本作用下的各层误差，即

输出层的误差为：$\delta^p=y^p(1-y^p)(t^p-y^p)$　　(p=1,2,3,4)

隐含层第 1 个神经元的误差为：$\delta_1^p=\delta^p\cdot w_a\cdot o_1^p(1-o_1^p)$　　(p=1,2,3,4)

隐含层第 2 个神经元的误差为：$\delta_2^p = \delta^p \cdot w_b \cdot o_2^p(1-o_2^p)$　　(p=1,2,3,4)

（4）根据式（1-25）和式（1-26）调整各层的加权系数及偏值，即

输出层的加权系数及偏值修正公式为

$$w_a(k+1) = w_a(k) + \eta\sum_{p=1}^{4}\delta^p o_1^p \ , \quad w_b(k+1) = w_b(k) + \eta\sum_{p=1}^{4}\delta^p o_2^p$$

$$b(k+1) = b(k) + \eta\sum_{p=1}^{4}\delta^p$$

隐含层的加权系数及偏值修正公式为

$$w_{11}(k+1) = w_{11}(k) + \eta\sum_{p=1}^{4}\delta_1^p x_1^p \ , \quad w_{12}(k+1) = w_{12}(k) + \eta\sum_{p=1}^{4}\delta_1^p x_2^p$$

$$w_{21}(k+1) = w_{21}(k) + \eta\sum_{p=1}^{4}\delta_2^p x_1^p \ , \quad w_{22}(k+1) = w_{22}(k) + \eta\sum_{p=1}^{4}\delta_2^p x_2^p$$

$$b_1(k+1) = b_1(k) + \eta\sum_{p=1}^{4}\delta_1^p \ , \quad b_2(k+1) = b_2(k) + \eta\sum_{p=1}^{4}\delta_2^p$$

（5）计算输出误差，即

$$J = \frac{1}{2}[(t^{(1)}-y^{(1)})^2 + (t^{(2)}-y^{(2)})^2 + (t^{(3)}-y^{(3)})^2 + (t^{(4)}-y^{(4)})^2]$$

存在一个ε>0，使 J<ε，否则返回（2）重新计算，直到误差满足要求为止。

例 1-4　利用一个引入惯性项的 3 输入 2 输出的两层 BP 神经网络训练一个输入为$[1\ -1\ 1]^T$，希望的输出为$[1\ 1]^T$ 的神经网络系统。激活函数取

$$g(x) = \frac{2}{1+\exp(-x)} - 1$$

解：根据 BP 网络学习算法的计算步骤，用 MATLAB 语言编写的在线学习程序如下。

```
%ex1_4.m
%两层BP算法的第一阶段——学习期（训练加权系数Wki，Wij）
%初始化
lr=0.05;err_goal=0.001;                %lr为学习速率；err_goal为期望误差最小值
max_epoch=10000;a=0.9;                 %训练的最大次数，a为惯性系数
Oi=0;Ok=0;                             %置隐含层和输出层各神经元输出初值为零
%提供一组训练集和目标值（3输入2输出）
X=[1;-1;1];T=[1;1];
%初始化Wki,Wij（M-输入节点j的数量，q-隐含层节点i的数量，L-输出节点k的数量）
[M,N]=size(X);q=8;[L,N]=size(T);       %N为训练集对数量
Wij=rand(q,M);Wki=rand(L,q);Wij0=zeros(size(Wij));Wki0=zeros(size(Wki));
for epoch=1:max_epoch
    %计算隐含层各神经元输出
    NETi=Wij*X;
    for j=1:N
      for i=1:q
        Oi(i,j)=2/(1+exp(-NETi(i,j)))-1;
```

```
        end
    end
    %计算输出层各神经元输出
    NETk=Wki*Oi;
    for i=1:N
        for k=1:L
            Ok(k,i)=2/(1+exp(-NETk(k,i)))-1;
        end
    end
    %计算误差函数
    E=((T-Ok)'*(T-Ok))/2;if (E<err_goal) break;end
    %调整输出层加权系数
    deltak=Ok.*(1-Ok).*(T-Ok);W=Wki;Wki=Wki+lr*deltak*Oi'+a*(Wki-Wki0);Wki0=W;
    %调整隐含层加权系数
    deltai=Oi.*(1-Oi).*(deltak'*Wki)';W=Wij;Wij=Wij+lr*deltai*X'+a*(Wij-Wij0);Wij0=W;
end
epoch                                                    %显示计算次数
%BP算法的第二阶段——工作期（根据训练好的Wki、Wij和给定的输入计算输出）
X1=X;                                                    %给定输入
%计算隐含层各神经元输出
NETi=Wij*X1;
for j=1:N
    for i=1:q
        Oi(i,j)=2/(1+exp(-NETi(i,j)))-1;
    end
end
%计算输出层各神经元输出
NETk=Wki*Oi;
for i=1:N
    for k=1:L
        Ok(k,i)=2/(1+exp(-NETk(k,i)))-1;
    end
end
Ok                                                       %显示网络输出层的输出
```

结果显示：

```
epoch =
     3
Ok =
    0.9955
    0.9969
```

由此可见，对于例 1-4，当惯性系数为 0.9 时，引入惯性项的 BP 网络经过 3 次训练就可训练成功。读者可以验证，当惯性系数为 0（即采用不带惯性项的基本 BP 网络）时，网络需要训练上千次才能训练成功。

1.2.5　径向基神经网络

1985 年，Powell 提出了多变量插值的径向基函数（Radial Basis Function，RBF）方法。1988 年，Broomhead 和 Lowe 首先将 RBF 应用于神经网络设计，从而构成了 RBF 神经网络。它是一种局部逼近的神经网络。众所周知，BP 网络用于函数逼近时，权值的调节采用的是负梯度下降法，这种调节权值的方法有它的局限性，即存在着收敛速度慢和局部极小等缺点。而 RBF 神经网络无论在逼近能力、分类能力和学习速度等方面均优于 BP 网络。径向基函数网络比 BP 网络需要更多的神经元，但是它能够按时间片来训练网络。径向基网络是一种局部逼近网络，已证明它能以任意精度逼近任一连续函数。当有很多的训练向量时，这种网络很有效果。

RBF 网络的结构与多层前向网络类似，它是具有单隐层的一种两层前向网络。输入层由信号源节点组成。隐含层的单元数视所描述问题的需要而定。输出层对输入的作用做出响应。从输入空间到隐含层空间的变换是非线性的，而从隐含层空间到输出层空间的变换是线性的。隐单元的变换函数是 RBF，它是一种局部分布的对中心点径向对称衰减的非负非线性函数。

构成 RBF 网络的基本思想是：用 RBF 作为隐单元的“基”构成隐含层空间，这样就可将输入矢量直接（即不通过权连接）映射到隐空间。当 RBF 的中心点确定以后，这种映射关系也就确定了。而隐含层空间到输出空间的映射是线性的，即网络的输出是隐单元输出的线性加权和。此处的权即为网络可调参数。由此可见，从总体上看，网络由输入到输出的映射是非线性的，而网络输出对可调参数而言却又是线性的。这样，网络的权就可由线性方程组直接解出或用 LMS 方法计算，从而大大加快了学习速度并避免了局部极小问题。

1．径向基函数网络模型

RBF 网络由两层组成，其结构如图 1-20 所示。输入层节点只是传递输入信号到隐含层，隐含层节点（也称为 RBF 节点）由像高斯函数那样的辐射状作用函数构成，而输出层节点通常是简单的线性函数。

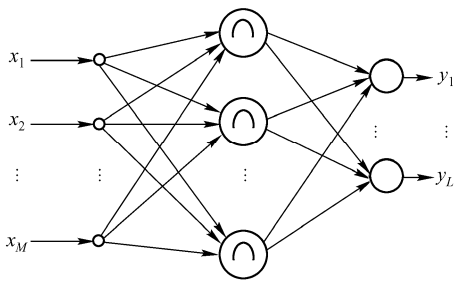


图 1-20　RBF 网络的结构

隐含层节点中的作用函数（核函数）对输入信号将在局部产生响应，也就是说，当输入信号靠近该函数的中央范围时，隐含层节点将产生较大的输出。由此可看出这种网络具有局部逼近能力，所以径向基函数网络也称为局部感知场网络。

2．网络输出

设网络输入 $\boldsymbol{x}$ 为 M 维向量，输出 $\boldsymbol{y}$ 为 L 维向量，输入/输出样本对长度为 N。

RBF 网络的输入层到隐含层实现 $\boldsymbol{x}\rightarrow u_i(\boldsymbol{x})$的非线性映射，径向基网络隐含层节点的作用函数一般取下列几种形式，即

$$u_i(\boldsymbol{x}) = \exp[-(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}/\delta_i^2)]$$

$$u_i(\boldsymbol{x})=\frac{1}{(\delta_i^2+\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x})^a},a>0$$

$$u_i(\boldsymbol{x})=(\delta_i^2+\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x})^{\beta},a<\beta<1$$

上面这些函数都是径向对称的，虽然有各种各样的激活函数，但最常用的是高斯激活函数，如 RBF 网络隐含层第 i 个节点的输出可由下式表示，即

$$u_i(\boldsymbol{x})=\exp\left[-\frac{(\boldsymbol{x}-\boldsymbol{c}_i)^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{c}_i)}{2\sigma_i^2}\right]\qquad(i=1,2,\cdots,q)\tag{1-32}$$

式中，u_i 是第 i 个隐节点的输出；σ_i 是第 i 个隐节点的标准化常数；q 是隐含层节点数，$\boldsymbol{x}=[x_1,x_2,\cdots,x_M]^{\mathrm{T}}$ 是输入样本；$\boldsymbol{c}_i$ 是第 i 个隐节点高斯函数的中心向量，此向量是一个与输入样本 $\boldsymbol{x}$ 的维数相同的列向量，即 $\boldsymbol{c}_i=[c_{i1},c_{i2},\cdots,c_{iM}]^{\mathrm{T}}$。

由式（1-32）可知，节点的输出范围在 0 和 1 之间，且输入样本越靠近节点的中心，输出值越大。当 $\boldsymbol{x}=\boldsymbol{c}_i$ 时，$u_i=1$。

采用高斯函数，具备如下优点：

（1）表示形式简单，即使对于多变量输入也不增加太多的复杂性。

（2）径向对称。

（3）光滑性好，任意阶导数存在。

（4）由于该基函数表示简单且解析性好，因而便于进行理论分析。

考虑到提高网络精度和减少隐含层节点数，也可以将网络激活函数改成多变量正态密度函数，即

$$u_i(\boldsymbol{x})=\exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{c}_i)K(\boldsymbol{x}-\boldsymbol{c}_i)^{\mathrm{T}}\right]\tag{1-33}$$

式中，$K=E[(\boldsymbol{x}-\boldsymbol{c}_i)^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{c}_i)]^{-1}$ 是输入协方差阵的逆。

注意，这时式（1-33）已不再是径向对称。

RBF 网络的隐含层到输出层实现 $u_i(\boldsymbol{x})\rightarrow y_k$ 的线性映射，即

$$y_k=\sum_{i=1}^{q}w_{ki}u_i-\theta_k\qquad(k=1,2,\cdots,L)\tag{1-34}$$

式中，u_i 是隐含层第 i 个节点的输出；y_k 是输出层第 k 个节点的输出；w_{ki} 是隐含层到输出层的加权系数；θ_k 是隐含层的阈值；q 是隐含层节点数。

3．RBF 网络的学习过程

设有 N 个训练样本，则系统对所有 N 个训练样本的总误差函数为

$$J=\sum_{p=1}^{N}J_p=\frac{1}{2}\sum_{p=1}^{N}\sum_{k=1}^{L}(t_k^p-y_k^p)^2=\frac{1}{2}\sum_{p=1}^{N}\sum_{k=1}^{L}e_k^2\tag{1-35}$$

式中，N 为模式样本对数；L 为网络输出节点数；t_k^p 表示在样本 p 作用下的第 k 个神经元的期望输出；y_k^p 表示在样本 p 作用下的第 k 个神经元的实际输出。

RBF 网络的学习过程分为两个阶段。第一阶段是无教师学习。它是根据所有的输入样本决定隐含层各节点的高斯核函数的中心向量 $\boldsymbol{c}_i$ 和标准化常数 σ_i。第二阶段是有教师学习。在决定好隐含层的参数后，根据样本，利用最小二乘原则，求出隐含层和输出层的权值 w_{ki}。有时在完成第二阶段的学习后，再根据样本信号，同时校正隐含层和输出层的参数，以进一步

提高网络的精度。下面具体介绍一下这两个阶段。

1）无教师学习阶段

无教师学习也称为非监督学习，是对所有样本的输入进行聚类，求得各隐含层节点的 RBF 的中心向量 $\boldsymbol{c}_i$。这里介绍 k–均值聚类算法调整中心向量，此算法将训练样本集中的输入向量分为若干族，在每个数据族内找出一个径向基函数中心向量，使得该族内各样本向量距该族中心的距离最小。算法步骤如下：

（1）给定各隐节点的初始中心向量 $\boldsymbol{c}_i(0)$ (i=1,2,⋯,q)，学习速率$\beta(0)$（0<$\beta(0)$<1）和判定停止计算的阈值ε。

（2）计算距离（欧氏距离）并求出最小距离的节点。

$$\begin{cases} d_i(k)=\left\|\boldsymbol{x}(k)-\boldsymbol{c}_i(k-1)\right\| ,1\leqslant i\leqslant q \\ d_{\min}(k)=\min d_i(k)=d_r(k) \end{cases} \tag{1-36}$$

式中，k 为样本序号；r 为中心向量 $\boldsymbol{c}_i(k-1)$与输入样本 $\boldsymbol{x}(k)$距离最近的隐节点序号。

（3）调整中心。

$$\begin{cases} \boldsymbol{c}_i(k)=\boldsymbol{c}_i(k-1),1\leqslant i\leqslant q,i\neq r \\ \boldsymbol{c}_r(k)=\boldsymbol{c}_r(k-1)+\beta(k)[\boldsymbol{x}(k)-\boldsymbol{c}_r(k-1)] \end{cases} \tag{1-37}$$

式中，$\beta(k)$是学习速率。$\beta(k)=\beta(k-1)/(1+\text{int}(k/q))^{1/2}$；int(·)表示对(·)进行取整运算。可见，每经过 q 个样本之后，调小一次学习速率，逐渐减至零。

（4）判定聚类质量。

对于全部样本 k(k=1,2,⋯,N)反复进行以上（2）、（3）步，直至满足以下条件，则聚类结束。

$$J_e=\sum_{i=1}^{q}\left\|\boldsymbol{x}(k)-\boldsymbol{c}_i(k)\right\|^2\leqslant\varepsilon \tag{1-38}$$

2）有教师学习阶段

有教师学习也称为有监督学习。当 $\boldsymbol{c}_i$ 确定以后，训练由隐含层至输出层之间的权值，由上可知，它是一个线性方程组，则求权值就成为线性优化问题。因此，问题有唯一确定的解，不存在 BP 网络中所遇到的局部极小值问题，肯定能获得全局最小点。类似于线性网络，RBF 网络的隐含层至输出层之间的连接权值 w_{ki}(k=1,2,⋯,L; i=1,2,⋯,q)学习算法为

$$w_{ki}(k+1)=w_{ki}(k)+\eta(t_k-y_k)u_i(\boldsymbol{x})/\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u} \tag{1-39}$$

式中，$\boldsymbol{u}=\left[u_1(\boldsymbol{x}),u_2(\boldsymbol{x}),\cdots,u_q(\boldsymbol{x})\right]^{\mathrm{T}}$；$u_i(\boldsymbol{x})$为高斯函数；$\eta$ 为学习速率。

可以证明当$0<\eta<2$时可保证该迭代学习算法的收敛性，而实际上通常只取$0<\eta<1$。t_k 和 y_k 分别表示第 k 个输出分量的期望值和实际值。由于向量 $\boldsymbol{u}$ 中只有少量几个元素为 1，其余均为零，因此在一次数据训练中只有少量的连接权需要调整。正是由于这个特点，才使得 RBF 神经网络具有比较快的学习速度。另外，由于当 $\boldsymbol{x}$ 远离 $\boldsymbol{c}_i$ 时，$u_i(\boldsymbol{x})$非常小，因此可作为 0 对待。因此，实际上只有当 $u_i(\boldsymbol{x})$大于某一数值（如 0.05）时才对相应的权值 w_{ki} 进行修改。经这样处理后，RBF 神经网络也同样具备局部逼近网络学习收敛快的优点。

4．RBF 网络有关的几个问题

（1）从理论上而言，RBF 网络和 BP 网络一样可近似任何的连续非线性函数。两者的主

要不同点是在非线性映射上采用了不同的作用函数。BP 网络中的隐含层节点使用的是 Sigmoid 函数，其函数值在输入空间中无限大的范围内为非零值，即作用函数为全局的；而 RBF 网络中的隐含层节点使用的是高斯函数，即它的作用函数是局部的。

（2）已证明 RBF 网络具有唯一最佳逼近的特性，且无局部极小。

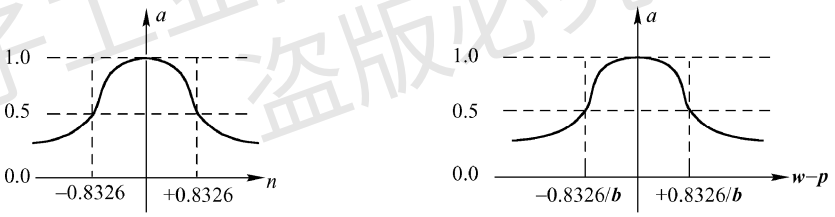
（3）求 RBF 网络隐节点的中心向量 $\boldsymbol{c}_i$ 和标准化常数 σ_i 是一个困难的问题。

（4）径向基函数，即径向对称函数有多种。对于同一组样本，如何选择合适的径向基函数，如何确定隐节点数，以使网络学习达到要求的精度，目前还无解决办法。当前，用计算机选择、设计、再检验是一种通用的手段。

（5）RBF 网络用于非线性系统辨识与控制，虽具有唯一最佳逼近的特性，以及无局部极小的优点，但隐节点的中心难求，这是该网络难以广泛应用的原因。

（6）与 BP 网络收敛速度慢的缺点相反，RBF 网络学习速度很快，适于在线实时控制。这是因为 RBF 网络把一个难题分解成了两个较易解决的问题。首先，通过若干个隐节点，用聚类方式覆盖全部样本模式；然后，修改输出层的权值，以获得最小映射误差。而这两步都比较直观。

（7）图 1-21 是径向基函数神经元传输函数 radbas( )的示意图，从图中可以注意到 RBF 网络的输入同前面介绍的神经网络的表达式有所不同。其网络输入为权值向量 $\boldsymbol{W}$ 与输入向量 $\boldsymbol{x}$ 之间的向量距离乘以阈值 $\boldsymbol{b}$，即 d=radbas(dist($\boldsymbol{W}$, $\boldsymbol{x}$)$\boldsymbol{b}$)。由左图可知，径向基函数的输入为 0 时，输出为极大值 1。由右图可知，径向基函数的输出随权值向量 $\boldsymbol{W}$ 和输入向量 $\boldsymbol{x}$ 之间的距离减小而增大，因此可以直观地想象：当输入向量 $\boldsymbol{x}$ 同权值向量 $\boldsymbol{W}$ 完全相同时，径向基函数的输出为 1，这时，径向基函数扮演了信号检测器的角色。阈值 $\boldsymbol{b}$ 可调节 RBF 神经元的敏感程度。例如，如果一个神经元的阈值为 0.1，那么，当向量距离为 8.326(0.8326/$\boldsymbol{b}$)时，神经元的输出为 0.5。



（a）a=radbas(n)　　（b）a=radbas(dist($\boldsymbol{w}$,$\boldsymbol{p}$)$\boldsymbol{b}$)

图 1-21　传输函数 radbas( )的示意图

例 1-5　对于单输入单输出，隐含层有三个节点的高斯 RBF 网络，如图 1-22 所示，已知三个隐含节点的中心 c_1=−1,c_2=0,c_3=2.5；标准化参数 σ_i^2=1(i=1,2,3)。三个非线性作用函数（高斯 RBF）如图 1-23 所示，隐含层至输出层的权系数 $\boldsymbol{w}$=[1.1,−1,0.5]$^{\mathrm{T}}$。

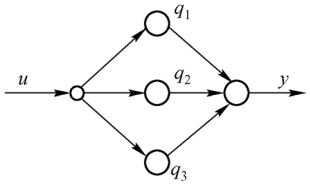


图 1-22　RBF 网络结构

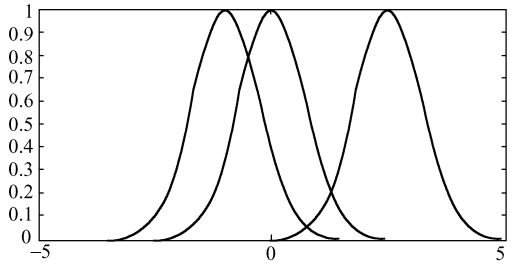


图 1-23　三个高斯 RBF

解：（1）在上述条件下，设网络输入 x= –5∶0.5∶4.5，如图 1-24（a）所示。由式（1-32）和式（1-34）可得到网络的输出 t，如图 1-24（b）所示，即得到样本对 x^p/t^p (p=1～20)。

（2）由样本对 x^p/t^p，设隐含节点中心为已知，训练网络权系数 $\boldsymbol{w}=[w_1,w_2,w_3]^{\mathrm{T}}$。取初始权值 $\boldsymbol{w}(0)$，采用式（1-39）训练网络，次数 $p\geqslant 5$，网络输出与期望输出相等，即样本 $y^p=t^p$，如图 1-24（c）所示。对于目标函数式（1-35），当次数 $p\geqslant 5$ 时，$J(p)=0$，如图 1-24（e）所示。

（3）设隐含层节点中心为未知，采用 k-均值与 LMS 算法，由式（1-36）～式（1-39）训练网络，由于隐含层节点中心很难搜索到所设值，因此网络输出与样本输出有相当的误差，$y^p\neq t^p$，如图 1-24（d）所示。当训练次数 p 增加时，目标函数为非零常值，如图 1-24（f）所示。

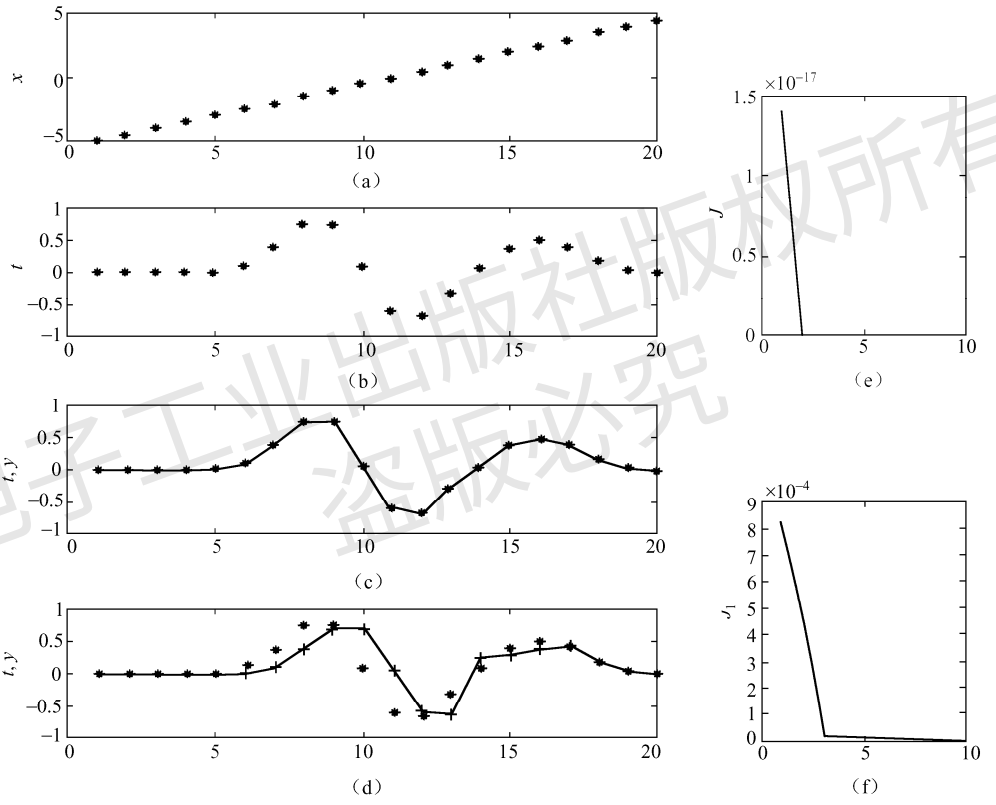


图 1-24　高斯 RBF 网络学习实例

1.2.6　竞争学习神经网络

竞争学习是一种典型无教师学习策略，是指同一神经元层次上各个神经元相互之间进行竞争，竞争胜利的神经元修改与其相连的连接权值，它模拟人类根据过去经验自动适应无法预测的环境变化。在无监督学习中，只向网络提供一些学习样本，而不提供理想的输出。网络根据输入样本进行自组织，并将其划分到相应的模式类中。因为没有教师信号，所以这类网络通常利用竞争的原则进行。学习时只需给定一个输入模式集作为训练集，网络自行组织训练模式，并将其分成不同类型。与 BP 学习相比，这种学习能力进一步拓宽了神经网络在

模式识别、分类方面的应用。竞争学习网络的核心——竞争层是许多神经网络模型的重要组成部分，例如自组织竞争神经网络（Self organizing NN）、Kohonen 提出的自组织特征映射网络（SOM）、Hecht-Nielson 提出的反传网络（CPN），Grossberg 与 Carpenter 提出的自适应共振理论（ART）网络模型等均包含竞争层。以下将分别详细讲述这些竞争网络结构的基本思想和学习规则。

1．自组织竞争神经网络（Self organizing NN）

自组织竞争神经网络的形成是受生物神经系统的启发，之所以称为自组织竞争神经网络，是因为它一般是由输入层和竞争层构成的单层网络。自组织竞争神经网络能够识别成组的相似向量，常用于进行模式分类。

1）自组织竞争神经网络的形成

在生物神经系统中，存在一种“侧抑制”的现象，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。这种侧抑制式神经细胞之间出现竞争，虽然开始阶段各个神经细胞都处于程度不同的兴奋状态，由于侧抑制的作用，各细胞之间相互竞争的最终结果是：兴奋作用最强的神经细胞所产生的抑制作用战胜了它周围所有其他细胞的抑制作用而“赢”了，其周围的其他神经细胞则全“输”了。

自组织竞争神经网络正是基于上述生物结构和现象形成的。它是一种以无教师示教的方式进行网络训练的，具有自组织功能的神经网络，网络通过自身训练，自动对输入模式进行分类。在网络结构上，自组织竞争神经网络一般是由输入层和竞争层构成的单层网络。网络没有隐含层，两层之间各神经元实现双向连接，有时竞争层各神经元之间还存在横向连接。在学习算法上，它模拟生物神经系统依靠神经元之间的兴奋、协调与抑制、竞争的作用来进行信息处理的动力学原理，指导网络的学习与工作。

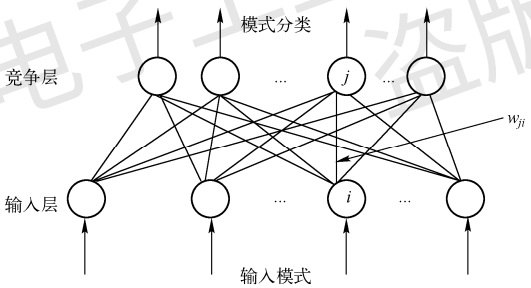


图 1-25　单层自组织竞争神经网络

2）自组织竞争神经网络的结构

自组织竞争神经网络由输入层和竞争层组成，输入层由接收输入模式的处理单元构成；竞争层的竞争单元争相响应输入模式，胜者表示输入模式的所属类别。输入层单元到竞争层单元的连接为全互连方式，连接权是可调节的。对于给定的一个输入模式，只调节获胜单元的连接权。图 1-25 所示为一个单层自组织竞争神经网络。

3）竞争学习机理

在自组织竞争神经网络中，每个竞争单元和输入层单元都有一个连接权，其取值在 0 与 1 之间。为了简化网络，假设任意一个给定竞争单元的权值和总是为 1，即

$$\sum_j w_{ji} \approx 1 \tag{1-40}$$

网络学习时，初始权值一般满足上式的一组小的随机数；输入模式是二进制的 0/1 向量。图 1-26 所示为竞争层的一个处理单元。

竞争单元的处理分为两步，首先计算每个单元输入的加权和，然后在竞争中产生输出。对于第 j 个竞争单元，其输出总和为

$$S_j = \sum_i w_{ji} x_i \tag{1-41}$$

当竞争层所有单元的输入总和计算完毕，便开始竞争。根据“胜者为王，败者为寇”的道理，竞争层中具有最高输入总和的单元被确定为胜者，其输出状态为 1，其他各单元状态为 0，即

$$x_j^c = \begin{cases} 1, & S_j > \max(S_k, k \neq j, k = 1,2,\cdots,n) \\ 0, & \text{其他条件下} \end{cases} \tag{1-42}$$

式中，x_j^c 表示竞争层第 j 个单元输出状态。当 $S_j = S_k (k \neq j)$ 时，通常取位于 j 左边的处理单元状态为 1。

对于某一输入模式，当竞争胜者单元被确定后，更新权值，也只有获胜单元的权值被增加一个量，使得再次遇到该输入模式时，该单元有更大的输入总和。权值更新规则表示为

$$\Delta w_{ji} = \eta \left(\frac{x_i}{m} - w_{ji} \right) \tag{1-43}$$

式中，η 表示学习因子，0<η<1，反映权值更新速率，一般取值为 0.01～0.3；m 表示输入层状态为 1 的单元个数；各单元初始权值选其和为 1 的一组随机数。

竞争学习的基本思想：竞争获胜单元的权值修正，当获胜单元的输入状态为 1 时，相应权值增加；当相连输入单元状态为 0 时，相应权值减小。学习过程中，权值越来越接近于相应的输入状态，这个变化可由如图 1-27 所示例子来反映。图中，第一、第五、第六个输入单元权值不断增大，其他权值不断减小。如果相同的输入模式立刻再次送给网络，那么上一次权值修正的结果将使获胜单元输入总和稍微变大。另外，类似于当前输入的输入模式，也将使相应获胜单元产生较大的输入总和。因此，当用相同或类似模式重复学习时，原已获胜单元有可能再次获胜。

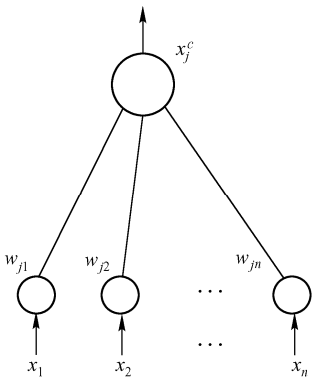


图 1-26　竞争层处理单元

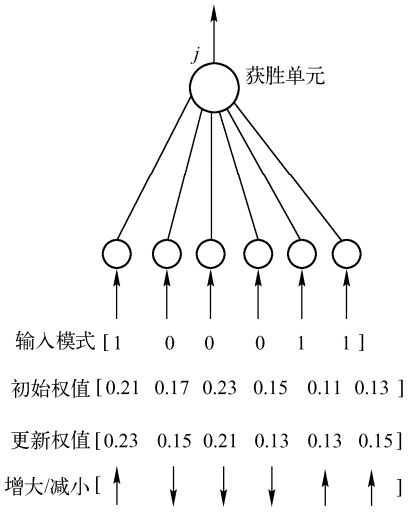


图 1-27　竞争学习中修正权值

按照权值修正算式（1-43），获胜单元的一些权值减小一个量，另一些则增大一个量，其结果是获胜单元的权值之和仍然满足为 1 的约束。因为将式（1-43）权值修正量对所有输入求和，结果为 0。

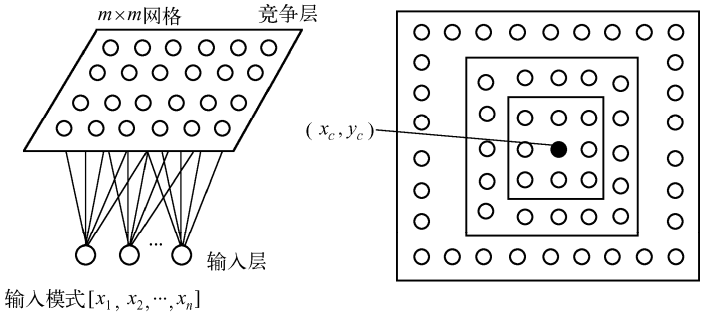
$$\sum_i \Delta w_{ji} = \eta\left(\frac{1}{m}\sum_i x_i - \sum_i w_{ji}\right) = \eta(1-1) = 0$$

2．自组织特征映射网络（SOM）

自组织特征映射网络（Self-Organizing feature Map，SOM）是由芬兰赫尔辛基大学神经网络专家 Kohonen 教授在 1981 年提出的。他认为一个神经网络接受外界输入模式时，将会分为不同的区域，各区域对输入模式具有不同的响应特征，同时这一过程是自动完成的。各神经元的连接权值具有一定的分布。最邻近的神经元互相刺激，而较远一些的则具有较弱的刺激作用。这种网络模拟大脑神经系统自组织特征映射的功能，它是一种竞争式学习网络，在学习中能无监督地进行自组织学习。自组织特征映射神经网络根据输入向量在输入空间的分布情况对它们进行分类。与自组织竞争网络不同的是，在自组织特征映射神经网络中邻近的神经元能够识别输入空间中邻近的部分。这样，自组织特征映射神经网络不但能够像自组织竞争神经网络一样学习输入的分布情况，而且可以学习进行训练输入向量的拓扑结构。总之，自组织特征映射网络是一种无教师的聚类方法，与传统的模式聚类方法相比，它所形成的聚类中心能映射到一个曲面或平面上，而保持拓扑结构不变。自组织特征映射网络应用广泛，可用于语言识别、图像压缩、机器人控制、优化问题等。

1）自组织特征映射网络的结构

自组织特征映射网络（SOM）的基本网络结构也是由输入层和竞争层组成的。与自组织竞争学习网络不同之处是，其竞争层按二维网络阵列方式组织，而且权值更新的策略也不同。如图 1-28（a）所示，输入层神经元数为 n，竞争层由 $M=m^2$ 个神经元组成，且构成一个二维平面阵列。输入层与竞争层之间实行全互连接，有时竞争层各神经元之间还实行侧抑制连接。网络中有两种连接权，一种是神经元对外部输入反应的连接权值，另一种是神经元之间的连接权值，它的大小控制着神经元之间的交互作用的大小。对于给定的输入模式，训练过程不仅要调节竞争获胜单元的各连接权值，而且还要调节获胜单元的邻域单元权值。假设竞争获胜单元位于(x_c,y_c)处，则该获胜单元的领域单元定义为包括在下述矩形框内的所有单元。其中，d 表示由中心点(x_c,y_c)到邻域单元位置(x,y)的距离。图 1-28（b）中标注了 d 为 1,2,3 时获胜单元的邻域单元。



（a）SOM基本结构　　（b）获胜单元的邻域单元

图 1-28　SOM 结构

2）自组织特征映射网络的学习机理

自组织特征映射算法是一种无教师示教的聚类方法，它能将任意输入模式在输出层映射

成一维或二维离散图形，并保持其拓扑结构不变。即在无教师示教的情况下，通过对输入模式的自组织学习，在竞争层将分类结果表示出来。此外，网络通过对输入模式的反复学习，可以使连接权矢量空间分布密度与输入模式的概率分布趋于一致，即连接权矢量空间分布能反映输入模式的统计特征。

自组织映射网络的竞争学习过程包括三个关键点：第一，对于给定输入模式，确定竞争层获胜单元；第二，按照学习规则修正获胜单元及其邻域单元的连接权值；第三，逐渐减小邻域及学习过程中权值的变化量。竞争获胜单元及其邻域单元权值更新规则可表达为

$$w_{ji}(k+1)=w_{ji}(k)+\Delta w_{ji} \tag{1-44}$$

且

$$\Delta w_{ji}=\begin{cases}\eta_c(x_i-w_{ji}), & \text{获胜单元}\\ \eta_N(x_i-w_{ji}), & \text{获胜邻域单元}\end{cases}$$

式中，η_c，η_N 分别表示获胜单元及其邻域单元的权值学习因子，取值在(0,1)区间，且 $\eta_c>\eta_N$；x_i，w_{ji} 表示竞争获胜单元的输入及连接权。

特别要强调的是，在学习过程中，权值学习因子 η_c 及竞争层获胜单元的邻域 N_c 的大小是逐渐减小的。学习因子 η_N 的初值一般取得比较大，随着学习过程的迭代 η_N 值逐渐减小，常用的一个调整策略表达式为

$$\eta_N=\eta_{N_0}\left(1-\frac{t}{T}\right) \tag{1-45}$$

式中，η_{N_0} 表示 η_N 初始值，通常在 0.2～0.5 之间取值；t 表示迭代次数；T 表示整个迭代设定次数。

邻域的宽度也随着学习过程的迭代而减小，因此，由竞争获胜单元到邻域单元的距离 d 相应减小。假设 d 的初值记 d_0，一般取值为 1/2 或 1/3 竞争层宽度，d 的减小策略可表达为

$$d=d_0\left(1-\frac{t}{T}\right) \tag{1-46}$$

自组织映射网络的学习使竞争获胜单元的邻域单元受到激励，邻域之外较远的单元受到抑制。

3．反传网络（CPN）

反传网络的结构如图 1-29 所示，从形式上看，CPN 也是一个多层前向网络，可用于实现模式映射，这一点与 BP 网络相似，但是在工作机理方面，二者有明显的差异。CPN 除了输入层外，还有两个功能层：第一层为竞争层，其权值学习采用竞争学习策略；第二层为功能输出层，又称为 Grossberg 层，它根据竞争层获胜单元的输出，产生一输出模式，采用有教师学习策略（如δ学习规则，Grossberg 学习算法）修正权值，以获得期望输出。之所以这种网络称为反传网络，是当它用作联想记忆时是由网络的组织得来的。如图 1-29（b）所示，假设输入模式由向量 $\boldsymbol{X}$ 和向量 $\boldsymbol{Y}$ 组成，即（$\boldsymbol{X}$,$\boldsymbol{Y}$）；期望输出模式仍为（$\boldsymbol{X}$,$\boldsymbol{Y}$）；网络实际输出模式为（$\boldsymbol{X}'$,$\boldsymbol{Y}'$）；当用做联想记忆时，如给定输入为（$\boldsymbol{X}$,*），则网络能回忆出（$\boldsymbol{X}$,$\boldsymbol{Y}$）；相反，若给定输入为（*,$\boldsymbol{Y}$），则能回忆出（$\boldsymbol{X}$,$\boldsymbol{Y}$）。

图 1-30 所示为一个 CPN 对于给定的输入模式的一次训练。给定输入模式，竞争层单元竞争响应，只有一个单元获胜，输出状态为 1，其余为 0。如图 1-30（a）、（b）所示，竞争获胜单元激活输出层产生一个输出模式。如图 1-30（c）所示，竞争层、输出层的权值采用不同学习策略调节。

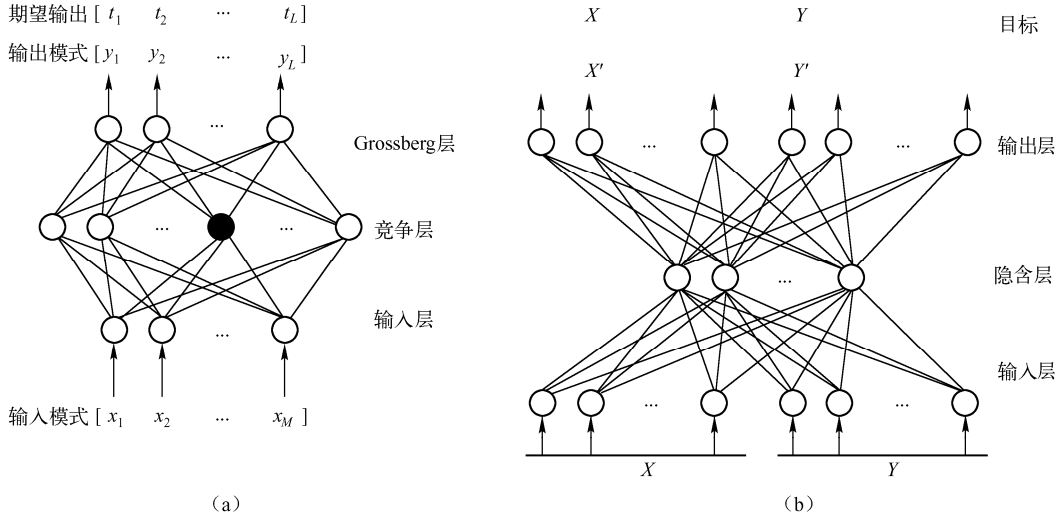


图 1-29 CPN 结构

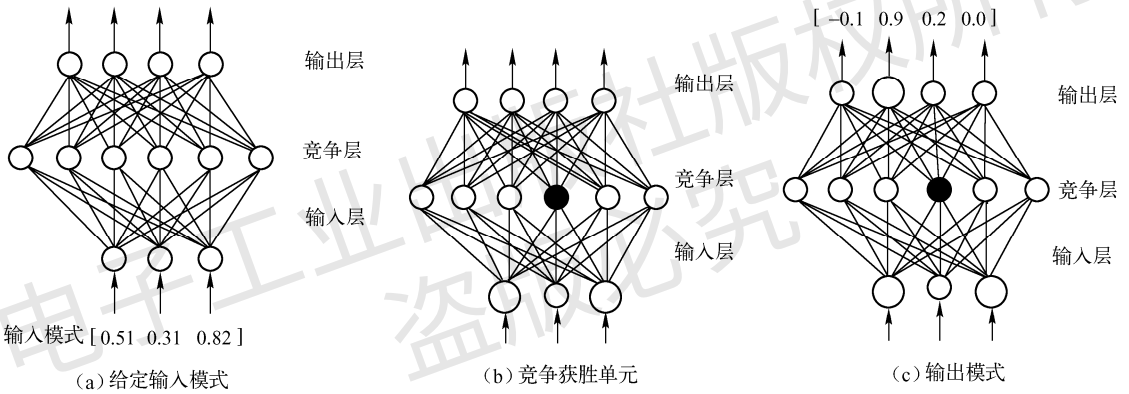


图 1-30 CPN 的训练

4．自适应共振理论（ART）

自适应共振理论（Adaptive Resonance Theory，ART）是由美国学者 S.Grossberg 和 G.Carpenter 在 20 世纪 80 年代提出的，它的最初模型为 ART-1，只能用于二进制输入，后来有了可适用于连续信号输入的 ART-2，然后又发展了 ART-3。ART-3 是在 ART-2 的基础上发展起来的。它兼容了前两代的功能，同时把两层网络扩大为任意多层。ART-3 是当前最复杂的神经网络之一。以下仅重点介绍 ART-1。

对于 BP 网络，如果学习所用的模式是已知的，且是固定的，那么网络经过反复学习可以记住这些固定模式。一旦出现新的模式，网络的学习往往会修改甚至删除已学习的结果，从而网络只记得最新的模式。

ART 网络是一种向量模式的识别器，它根据存储的模式对输入向量进行分类，其简化的结果如图 1-31 所示。当存储的模式中有模式和输入模式相匹配时，代表该存储模式的参数就被调整以更接近输入模式。反之，如果在存储模式中，没有发现和输入模式相匹配的模式时，则输入模式作为新的模式被存储到网络中，其他的存储模式保持不变。如图 1-31

所示，ART 网络由比较和识别两层神经元组成，增益控制 l、2 和复置用来控制网络的学习和分类。

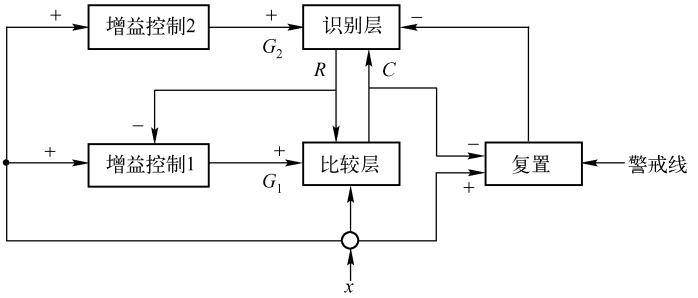


图 1-31　ART 网络的简化结构

ART 网络的工作过程如下。当没有输入时，$\boldsymbol{x}$ 的所有成分为 0，使得增益矩阵 2 的信号均为 0，因此也使得识别层的输出全为 0。当加入输入向量 $\boldsymbol{x}$ 时，因为 $\boldsymbol{x}$ 必含有不为 0 的元素，因此增益控制 1 和 2 的信号均为 1，从而使比较层的输出向量 $\boldsymbol{C}$ 和输入向量 $\boldsymbol{x}$ 完全相同。接着识别层寻找和 $\boldsymbol{C}$ 最匹配的神经元 j，并使其输出为 1，即 r_j=l，其他神经元输出均为 0。然后由 r_j=1 决定比较层中对应的存储模式 $\boldsymbol{T}_j=\{t_{j1},t_{j2},\cdots,t_{jm}\}$作为向量 $\boldsymbol{P}$。由于 r_j=l，增益控制 1 的信号被强制为 0，这时比较层的输出就是由比较 $\boldsymbol{x}$ 和 $\boldsymbol{P}$ 产生的结果。如果由上而下的反馈信号和输入模式不匹配（$\boldsymbol{P}$ 和 $\boldsymbol{x}$ 对应的成分不同）时，$\boldsymbol{C}$ 的相应成分就变为 0。因此若 $\boldsymbol{C}$ 的成分中 0 多，而 $\boldsymbol{x}$ 的成分中 1 多时，表明由上而下的反馈模式 $\boldsymbol{P}$ 不是所寻找的模式，此时产生复置信号，使识别层被激活的神经元复原，即使其输出为 0。

如果没有产生复置信号，则表明由上而下的反馈模式和输入模式匹配。反之，则必须搜索其他存储模式，看是否和输入模式相匹配，这一过程反复进行，直到找到一个相匹配的存储模式。如果在所有的存储模式中都没有找到相匹配的模式时，输入模式作为新的模式存储到网络中。

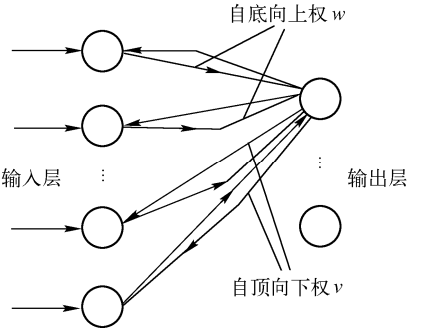


图 1-32　ART-1 网络

自适应共振理论（ART）网络具有不同的版本。图 1-32 为 ART-1 版本，用于处理二元输入。新的版本，如 ART-2，能够处理连续值输入。

从如图 1-32 所示可见，一个 ART-1 网络含有两层，一个输入层和一个输出层。这两层完全互连，该连接沿着正向（自底向上）和反馈（自顶向下）两个方向进行。自底向上连接至一个输出神经元 i 的权矢量 $\boldsymbol{w}_i$ 形成它所表示的类的一个样本。全部权矢量 $\boldsymbol{w}_i$ 构成网络的长期存储器，用于选择优胜的神经元，该神经元的权矢量 $\boldsymbol{w}_i$ 最相似于当前输入模式。自顶向下从一个输出神经元 i 连接的权矢量 $\boldsymbol{v}_i$ 用于警戒测试，即检验某个输入模式是否足够靠近已存储的样本。警戒矢量 $\boldsymbol{v}_i$ 构成网络的短期存储器。$\boldsymbol{v}_i$ 和 $\boldsymbol{w}_i$ 是相关的，$\boldsymbol{w}_i$ 是 $\boldsymbol{v}_i$ 的一个格化副本，即

$$\boldsymbol{w}_i=\frac{\boldsymbol{v}_i}{\varepsilon+\sum v_{ji}} \tag{1-47}$$

在式（1-47）中，ε为一小的常数，v_{ji} 为 $\boldsymbol{v}_i$ 的第 j 个分量（即从输出神经元 i 到输入神经

元 j 连接的权值）。

当 ART-1 网络在工作时，其训练是连续进行的，且包括下列步骤：

（1）对于所有输出神经元，预置样本矢量 $\boldsymbol{w}_i$ 及警戒矢量 $\boldsymbol{v}_i$ 的初值，设定每个 $\boldsymbol{v}_i$ 的所有分量为 1，并根据式（1-47）计算 $\boldsymbol{w}_i$。如果一个输出神经元的全部警戒权值均置 1，则称为独立神经元，因为它不被指定表示任何模式类型。

（2）给出一个新的输入模式 $\boldsymbol{x}$。

（3）使所有的输出神经元能够参加激发竞争。

（4）从竞争神经元中找到获胜的输出神经元，即这个神经元的 $\boldsymbol{x}\boldsymbol{w}_i$ 值为最大；在开始训练时或不存在更好的输出神经元时，优胜神经元可能是一个独立神经元。

（5）检查该输入模式 $\boldsymbol{x}$ 是否与获胜神经元的警戒矢量 $\boldsymbol{v}_i$ 足够相似。相似性是由 $\boldsymbol{x}$ 的微分式 r 检测的，即

$$r=\frac{\boldsymbol{x}\boldsymbol{v}_i}{\sum x_i} \tag{1-48}$$

如果 r 值小于警戒阈值 ρ $(0<\rho<1)$，那么可以认为 $\boldsymbol{x}$ 与 $\boldsymbol{v}_i$ 是足够相似的。

（6）如果 $r\geqslant\rho$，即存在共振，则转向第（7）步；否则，使获胜神经元暂时无力进一步竞争，并转向第（4）步，重复这一过程直至不存在更多的有能力的神经元为止。

（7）调整最新获胜神经元的警戒矢量 $\boldsymbol{v}_i$，对它逻辑加上 $\boldsymbol{x}$，删去 $\boldsymbol{v}_i$ 内而不出现在 $\boldsymbol{x}$ 内的位，根据式（1-48），用新的 $\boldsymbol{v}_i$ 计算自底向上样本矢量 $\boldsymbol{w}_i$，激活该获胜神经元。

（8）转向第（2）步。

上述训练步骤能够做到，如果同样次序的训练模式被重复地送至此网络，那么其长期和短期存储器保持不变，即该网络是稳定的。假定存在足够多的输出神经元来表示所有不同的类，那么新的模式总是能够学会。因为如果它不与原来存储的样本很好匹配的话（即该网络是塑性的），新模式可被指定给独立输出神经元。

总的来说，ART 的特点如下：

（1）ART 是一种非监督的、向量聚类的竞争学习算法；

（2）ART 对“弹性-稳定性”问题提供了一种解决办法；

（3）ART 是以认知学和行为学为基础的模型；

（4）ART 在输入层与输出层使用大量反馈连接；

（5）ART 可用非线性微分方程组描述；

（6）ART 能工作于实数模式或二值模式输入。

5．竞争学习网络的特征

竞争学习网络的主要特征表现在竞争层，它采用无教师学习策略，每个竞争单元都相当于一个特征分类器。模式分类是这种网络的重要功能。在竞争学习方案中，网络通过极小化簇内模式距离及极大化不同簇间的距离实现模式分类。注意，这里所说模式距离是指海明距离，如模式 010 与 101 的海明距离为 3。通常，竞争学习网络的分簇响应结果与初始权值的设置以及输入模式的组织有一定的关系。可以设想，用一组输入模式来训练网络，网络将输入模式按其海明距离自然分成三类，如图 1-33 所示，假如竞争层的初始权值都是相同的，那么竞争分簇的结果是：首先训练的模式属于类 l，由竞争单元 1 表示；随后训练的模式如果不属于类 1，它就使竞争单元 2 表示类 2；当然，剩下的不属于前两类的模式使单元 3 获胜，为类 3。假如不

改变初始权值分布，只改变模式训练的次序，或者不改变上述训练次序，只改变初始权值分布，这两种情况都可能使竞争层单元对分簇响应不一样。对第一种情况，竞争单元 1 获胜，表示类 1；此时竞争单元 1 获胜，可能代表的是类 2 或类 3。有时，如果输入模式组织得不好，那么分类学习可能不稳定。会出现对同一输入模式，先由某一单元响应，以后又由另一单元响应，训练过程中就这样跳来跳去。因此，在竞争学习网络训练时要注意这一关系。

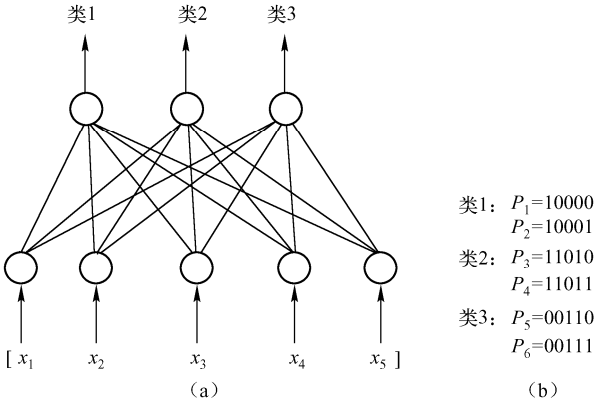


图 1-33　竞争学习聚类分析

竞争学习网络所实现的模式分类情况与典型的 BP 网络分类有所不同。BP 网络分类学习必须知道要将给定模式分成几类；而竞争网络能将给定的模式分成几类预先是不知道的，只有在学习以后才能知道，这种分类能力在许多场合是很有用的；从模式映射能力来看，像 CPN 这样的竞争网络，由于其竞争层仅有一个输出为 1 的获胜单元，所以不能得到某些映射所要求的复杂内部表示；BP 网络能够在最小均方意义上实现输入/输出映射的最优逼近。

竞争学习网络存在一些性能局限。首先，只用部分输入模式训练网络，当用一个明显不同的新的输入模式进行分类时，网络的分类能力可能降低。这是因为竞争学习采用非推理方式调节权值。另外，竞争学习对模式变换不冗余，其分类不是大小和旋转不变的，因为竞争学习网络没有从结构上支持大小和旋转不变的模式分类。

例 1-6　给定一个竞争网络，如图 1-34（a）所示，要求通过训练将输入模式集划分为两大类。设输入模式记为

$$(101)=\boldsymbol{P}_1;\ (100)=\boldsymbol{P}_2;\ (010)=\boldsymbol{P}_3;\ (011)=\boldsymbol{P}_4$$

解：竞争网络是如何将输入模式分成不同类呢？下面先分析一下训练集内四个模式的相似性。观察每两个模式之间的海明距离——两个二进制输入模式不同状态的个数，得到以下关系矩阵。

$$\boldsymbol{P}=(p_{ij})=\begin{pmatrix}0&1&3&2\\1&0&2&3\\3&2&0&1\\2&3&1&0\end{pmatrix}$$

所谓两个模式彼此相似，是指海明距离小于某个常量。

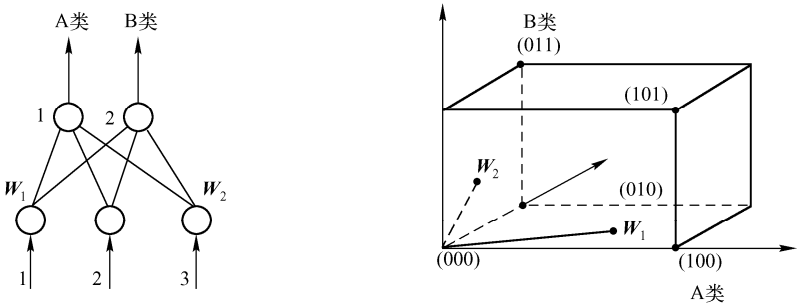
这里，模式 $\boldsymbol{P}_1$、$\boldsymbol{P}_2$ 彼此相似，$\boldsymbol{P}_3$、$\boldsymbol{P}_4$ 彼此相似；前两个模式 $\boldsymbol{P}_1$、$\boldsymbol{P}_2$ 与后两个模式 $\boldsymbol{P}_3$、$\boldsymbol{P}_4$ 的海明距离较大。因此，输入模式自然分成两类。该网络训练完后得到如下两类

A 类：$\boldsymbol{P}_1=(101), \boldsymbol{P}_2=(100)$　　B 类：$\boldsymbol{P}_3=(010), \boldsymbol{P}_4=(011)$

每一类包含两个输入模式，同一类模式海明距离为 1，不同类模式的海明距离为 2 或 3。网络的分类原则来源于输入模式的固有特征。用不同的初始权值反复进行训练，网络便能自组织学习，完成正确的模式分组。

图 1-34（b）表示训练完成后的权向量及训练集内输入模式的空间分布。所有 3 输入二

进制向量位于三维立方体的各顶点，竞争层单元 1 的权向量最接近于 A 类的两个模式；竞争单元 2 的权向量最接近于 B 类的两个模式。若输入模式为 A 类模式，则单元 1 竞争获胜；同样，当输入为 B 类的两个模式时，单元 2 竞争获胜。图中，权向量相对比较短，这是由单元权值之和必须为 1 约束的结果。



（a）竞争网络　　（b）全向量空间分布

图 1-34　竞争分类

1.2.7　学习向量量化神经网络

学习向量量化（LVQ）神经网络是一种由输入层、竞争层（又称为隐含层）和线性输出层组成的混合网络，LVQ 神经网络的学习结合了竞争学习和有监督的学习来形成分类。学习规则为

$$\Delta w_{ij}(k)=\begin{cases}\eta(x_j-w_{ij}), & \text{若神经元 } j \text{ 竞争获胜}\\ 0, & \text{若神经元 } j \text{ 竞争失败}\end{cases} \tag{1-49}$$

在 LVQ 神经网络中，第一层的每个神经元都指定给某个类，常常几个神经元被指定给同一个类。每类再被指定给第二层的一个神经元。第一层的神经元的个数与第二层的神经元个数至少相同，并且通常要大一些。和竞争网络一样，LVQ 神经网络的第一层的每个神经元学习原型向量，它可以对输入空间的区域分类。然而，不是通过计算内积得到输入和权值向量中最接近者，而是通过直接计算距离的方法来模拟 LVQ 网络。直接计算距离的一个优点是向量不必先格式化，当向量规格化了，无论是采用计算内积的方法还是直接计算距离，网络的响应将是相同的。

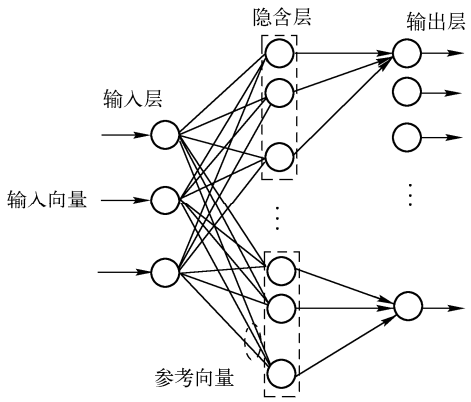


图 1-35　学习向量量化（LVQ）神经网络

LVQ 神经网络与竞争网络的特性几乎相同（至少对规格化向量）。然而，对于竞争网络，有非零输出的神经元表示输入向量属于那个类。而对于 LVQ 神经网络，竞争获胜的神经元表示的是一个子类而非一个类。一个类可能有几个不同的神经元（子类）组成。

图 1-35 给出一个学习向量量化神经网络，它是由输入层、竞争层（又称为隐含层）和输出层组成的。该网络在输入层和竞争层间为完全连接，而在竞争层与输出层间为部分连接，每个输出神经元与竞争神经元的不同组相连接。竞争层和输出层间的连接权值固定为 1。输入层和竞争层间的连接权值建立参考向量的分量（对每个竞争神经元指定一个参考向量）。在网络的训练过程中，这些权值被修改。竞

争神经元（又称为隐含神经元）和输出神经元都具有二进制输出值。当某个输入模式送至网络时，参考向量最接近输入模式的竞争神经元因获得激发而赢得竞争，因而允许它产生一个“1”。其他竞争神经元都被迫产生“0”。与包含获胜神经元的竞争神经元组成相连接的输出神经元也发出“1”，而其他输出神经元均发出“0”。产生“1”的输出神经元给出输入模式的类，每个输出神经元被表示为不同的类。

最简单的 LVQ 神经网络训练步骤如下：

（1）预置参考向量的初始权值。

（2）提供给网络一个训练输入模式。

（3）计算输入模式与每个参考向量间的 Euclidean 距离。

（4）更新最接近输入模式的参考向量（即获胜竞争神经元的参考向量）的权值。如果获胜竞争神经元以输入模式一样的类属于连接至输出神经元的缓冲器，那么参考向量应更接近输入模式。否则，参考向量就离开输入模式。

（5）返回（2），以某个新的训练输入模式重复本过程，直至全部训练模式被正确地分类或者满足某个终止准则为止。

1.2.8　Elman 神经网络

Elman 神经网络包含一个双曲正切 S 型隐含层和一个线性输出层，S 型隐含层接收网络输入和自身的反馈，线性输出层从 S 型隐含层得到输入。它是反馈网络中最有代表性的例子，这种网络也有多层结构，如图 1-36 所示，在这种网络中，除了普通的隐含层外，还有一个特别的隐含层，有时称为上下文层或状态层。该层从普通隐含层接收反馈信号，上下文层内的神经元输出被前向传输至隐含层。Elman 神经网络的这种组合结构特点使得其能在有限的时间内以任意精度逼近任意函数。这一点需要通过给递归层设置足够的神经元来实现。当需要逼近的函数复杂性增加时，需要的递归层神经元数目也要增加。由于 Elman 网络是 S 型/线性（Sigmoid/Linear）网络，它能够表达含有有限个不连续点的函数。又因为它们有一个反馈连接，所以它被训练后不仅能够识别和产生空间模式，还能够识别和产生时间模式。

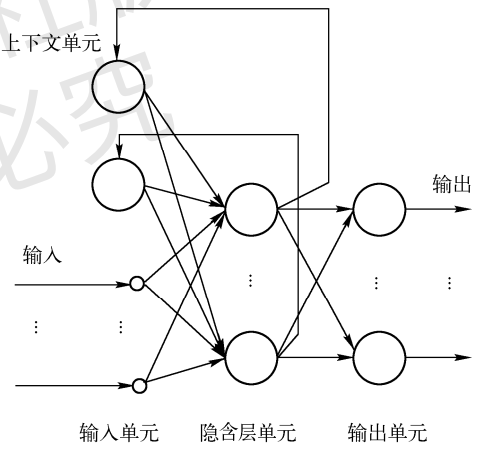


图 1-36　Elman 神经网络

对于多输入多输出网络，设上下文层的输出为 $\boldsymbol{y}_c(k)$，隐含层的输入和输出分别为 $\boldsymbol{x}_0(k)$和 $\boldsymbol{o}(k)$，网络在外部输入时间序列 $\boldsymbol{x}(k)$作用下的网络输出序列为 $\boldsymbol{y}(k)$，则有

$$\begin{aligned}\boldsymbol{x}_0(k+1)&=\boldsymbol{W}_1\boldsymbol{y}_c(k+1)+\boldsymbol{W}\boldsymbol{x}(k)+\boldsymbol{\theta}_1\\ \boldsymbol{y}_c(k)&=\boldsymbol{o}(k-1)=f(\boldsymbol{x}_0(k-1))\\ \boldsymbol{y}(k)&=\boldsymbol{W}_2\boldsymbol{o}(k)+\boldsymbol{\theta}_2\end{aligned}\tag{1-50}$$

式中，$\boldsymbol{W}_1$ 为输入层与隐含层间的连接权值；$\boldsymbol{W}_2$ 为隐含层与输出层间的连接权值；$f(\cdot)$ 为 S 型

激活函数。

当 Elman 神经网络的上下文层存在增益为α的自反馈连接时，称为改进型 Elman 神经网络。此时网络能模拟更高阶的动态系统，其上下文层的输出$\boldsymbol{y}_c(k)$变为

$$\boldsymbol{y}_c(k)=\boldsymbol{o}(k-1)+\alpha\boldsymbol{y}_c(k-1) \tag{1-51}$$

如果 Elman 神经网络只有正向连接是适用的，而反馈连接被预定为恒值，那么这些网络可视为普通的前馈网络。而且，Elman 神经网络可以用 BP 算法进行训练；否则，可采用遗传算法。

1.2.9 Hopfield 神经网络

Hopfield 神经网络是 1982 年由美国物理学家 Hopfield 首先提出来的。前面讨论的 BP 前向神经网络的特点是网络中没有反馈连接，不考虑输出与输入间在时间上的滞后影响，其输出与输入间仅是一种映射关系。而 Hopfield 网络则不同，它采用反馈连接，考虑输出与输入间在时间上的传输延迟，所表示的是一个动态过程，需要用差分方程或微分方程来描述，因而 Hopfield 网络是一种由非线性元件构成的反馈系统，其稳定状态的分析比 BP 网络要复杂得多。此外，在网络的学习训练，即加权系数的调整方面也不同，BP 前向网络采用的是一种有监督的误差均方修正方法，学习计算过程较长，收敛速度较慢。而 Hopfield 网络则是采用有监督的 Hebb 规则（用输入模式作为目标模式）来设计连接权。在一般情况下，计算的收敛速度很快。该网络主要用于联想记忆和优化计算。

在 Hopfield 网络中，每一个神经元都和所有其他神经元相连接，所以又称为全互连网络。研究表明，当连接加权系数矩阵无自连接并具有对称性质，即

$$w_{ii}=0,\ w_{ij}=w_{ji}\quad (i\neq j)\quad (i,j=1,2,\cdots,n)$$

时，算法是收敛的。

根据网络的输出是离散量或是连续量，Hopfield 网络分为离散型和连续型两种。下面分别对它们进行讨论。

1．离散型 Hopfield 网络

1）网络的结构和工作方式

离散型 Hopfield 网络的结构如图 1-37 所示。

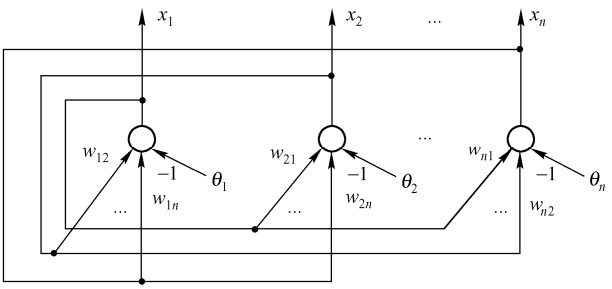


图 1-37 离散型 Hopfield 网络的结构

可以看出，它是一个单层网络，共有 n 个神经元节点，每个节点输出均连接到其他神经元的输入，同时所有其他神经元的输出均连到该神经元的输入。对于每一个神经元节点，其工作方式仍同以前一样，即

$$\begin{cases} s_i(k)=\sum_{j=1}^{n} w_{ij}x_j(k)-\theta_i \\ x_i(k+1)=f(s_i(k)) \end{cases} \tag{1-52}$$

式中，$w_{ii}=0$；θ_i 为阈值；$f(\cdot)$是变换函数。对于离散型 Hopfield 网络，$f(\cdot)$通常取为二值函数，即

$$f(s)=\begin{cases} 1, & s\geqslant 0 \\ -1, & s<0 \end{cases} \quad 或 \quad f(s)=\begin{cases} 1, & s\geqslant 0 \\ 0, & s<0 \end{cases} \tag{1-53}$$

整个网络有如下两种工作方式。

（1）异步方式或串行工作方式

在某一时刻只有一个神经元按照式（1-52）改变状态，而其余神经元的输出保持不变，这一变化的神经元可以按照随机方式或预定的顺序来选择。例如，若达到的神经元为第 i 个，则有

$$\begin{cases} x_i(k+1)=f\left(\sum_{j=1}^{n} w_{ij}x_j(k)-\theta_i\right) \\ x_j(k+1)=x_j(k), \quad (j\neq i) \end{cases} \tag{1-54}$$

其调整次序可以随机选定，也可按规定的次序进行。

（2）同步方式或并行工作方式

在某一时刻有 $n_1(0<n_1\leqslant n)$个神经元按照式（1-52）改变状态，而其余神经元的输出保持不变。变化的这一组神经元可以按照随机方式或预定的顺序来选择。当 $n_1=n$ 时，称为全并行方式，此时所有神经元都按照式（1-52）改变状态，即

$$x_i(k+1)=f\left(\sum_{j=1}^{n} w_{ij}x_j(k)-\theta_i\right),\ (i=1,2,\cdots,n) \tag{1-55}$$

上述同步计算方式也可写成如下的矩阵形式：

$$\boldsymbol{x}(k+1)=\boldsymbol{f}(\boldsymbol{W}\boldsymbol{x}(k)-\boldsymbol{\theta})$$

其中，$\boldsymbol{x}(k)=[x_1(k),x_2(k),\cdots,x_n(k)]^{\mathrm{T}}$ 和 $\boldsymbol{\theta}=[\theta_1,\theta_2,\cdots,\theta_n]^{\mathrm{T}}$ 是向量；$\boldsymbol{W}$ 是由 w_{ij} 所组成的 $n\times n$ 矩阵；$\boldsymbol{f}(s)$ 是向量函数，它表示 $\boldsymbol{f}(s)=[f(s_1),f(s_2),\cdots,f(s_n)]^{\mathrm{T}}$

该网络是动态的反馈网络，其输入是网络的状态初值：

$$\boldsymbol{x}(0)=[x_1(0),x_2(0),\cdots,x_n(0)]^{\mathrm{T}}$$

输出是网络的稳定状态 $\lim\limits_{k\to\infty}\boldsymbol{x}(k)$。

2）稳定性和吸引子

从上述工作中可以看出，离散型 Hopfield 网络实质上是一个离散的非线性动力学系统。如果系统是稳定的，则它可以从任意初态收敛到一个稳定状态；若系统是不稳定的，由于网络节点输出只有 1 和–1（或 1 和 0）两种状态，因而系统不可能出现无限发散，只可能出现限幅的自持振荡或极限环。

在 Hopfield 网络的拓扑结构及权值矩阵均一定的情况下，网络的稳定状态将与其初始状态有关。也就是说，Hopfield 网络是一种能储存若干个预先设置的稳定状态的网络。若将稳态视为一个记忆样本，那么初态朝稳态的收敛过程便是寻找记忆样本的过程。初态可认为是

给定样本的部分信息，网络改变的过程可认为是从部分信息找到全部信息，从而实现了联想记忆的功能。

若将稳态与某种优化计算的目标函数相对应，并作为目标函数的极小点。那么初态朝稳态的收敛过程便是优化计算过程。该优化计算是在网络演变过程中自动完成的。

（1）稳定性

若网络的状态 $\boldsymbol{x}$ 满足 $\boldsymbol{x}=f(\boldsymbol{W}\boldsymbol{x}-\boldsymbol{\theta})$，则称 $\boldsymbol{x}$ 为网络的稳定点或吸引子。

定理 1.3　对于离散型 Hopfield 网络，若按异步方式调整状态，且连接权矩阵 $\boldsymbol{W}$ 为对称阵，即 $w_{ij}=w_{ji}$，则对于任意初态，网络都最终收敛到一个吸引子。

证明：定义网络的能量函数为

$$E(k)=-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}x_j(k)x_i(k)+\sum_{i=1}^{n}\theta_i x_i(k)=-\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}(k)\boldsymbol{W}\boldsymbol{x}(k)+\boldsymbol{x}^{\mathrm{T}}(k)\boldsymbol{\theta}$$

由于神经元节点的状态只能取 1 和–1（或 1 和 0）两种状态，因此上述定义的能量函数 $E(k)$ 是有界的。令 $\Delta E(k)=E(k+1)-E(k),\Delta\boldsymbol{x}(k)=\boldsymbol{x}(k+1)-\boldsymbol{x}(k)$，则

$$\begin{aligned}\Delta E(k)&=E(k+1)-E(k)\\&=-\frac{1}{2}[\boldsymbol{x}(k)+\Delta\boldsymbol{x}(k)]^{\mathrm{T}}\boldsymbol{W}[\boldsymbol{x}(k)+\Delta\boldsymbol{x}(k)]+[\boldsymbol{x}(k)+\Delta\boldsymbol{x}(k)]^{\mathrm{T}}\boldsymbol{\theta}-\left[-\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}(k)\boldsymbol{W}\boldsymbol{x}(k)+\boldsymbol{x}^{\mathrm{T}}(k)\boldsymbol{\theta}\right]\\&=-\Delta\boldsymbol{x}^{\mathrm{T}}(k)\boldsymbol{W}\boldsymbol{x}(k)-\frac{1}{2}\Delta\boldsymbol{x}^{\mathrm{T}}(k)\boldsymbol{W}\Delta\boldsymbol{x}(k)+\Delta\boldsymbol{x}^{\mathrm{T}}(k)\boldsymbol{\theta}\\&=-\Delta\boldsymbol{x}^{\mathrm{T}}(k)[\boldsymbol{W}\boldsymbol{x}(k)-\boldsymbol{\theta}]-\frac{1}{2}\Delta\boldsymbol{x}^{\mathrm{T}}(k)\boldsymbol{W}\Delta\boldsymbol{x}(k)\end{aligned}$$

由于假定异步工作方式，因此可设第 k 时刻只有第 i 个神经元调整状态，即将 $\Delta\boldsymbol{x}(k)=[0\ \ \cdots\ \ 0\ \ \Delta x_i(k)\ \ 0\ \ \cdots\ \ 0]^{\mathrm{T}}$ 代入上式，则有

$$\Delta E(k)=-\Delta x_i(k)\left[\sum_{j=1}^{n}w_{ij}x_j(k)-\theta_i\right]-\frac{1}{2}\Delta x_i^2 w_{ii}$$

令 $s_i(k)=\sum_{j=1}^{n}w_{ij}x_j(k)-\theta_i$

则　　$\Delta E(k)=-\Delta x_i(k)\left[s_i(k)+\frac{1}{2}\Delta x_i(k)w_{ii}\right]=-\Delta x_i(k)s_i(k)\qquad(w_{ii}=0)$

设神经元节点取 1 和–1 两种状态，则

$$x_i(k+1)=f[s_i(k)]=\begin{cases}1 & ,\ \ s_i(k)\geqslant 0\\-1 & ,\ \ s_i(k)<0\end{cases}$$

下面考虑 $\Delta x_i(k)$ 可能出现的各种情况：

① $x_i(k)=-1,x_i(k+1)=f[s_i(k)]=1$ 时，有 $\Delta x_i(k)=2,s_i(k)\geqslant 0$，则

$$\Delta E(k)\leqslant 0$$

② $x_i(k)=1,x_i(k+1)=f[s_i(k)]=-1$ 时，有 $\Delta x_i(k)=-2,s_i(k)<0$，则

$$\Delta E(k)<0$$

③ $x_i(k+1)=x_i(k)$ 时，有 $\Delta x_i(k)=0$，则

$$\Delta E(k)=0$$

可见，在任何情况下均有 $\Delta E(k)\leqslant 0$，由于 $E(k)$ 有下界，所以 $E(k)$ 将收敛到一常数。下面需考察 $E(k)$ 收敛到常数时是否对应于网络的吸引子。根据上述分析，当 $\Delta E(k)=0$ 时，相应于以下两种情况之一。

① $x_i(k+1)=x_i(k)=1$ 或 $x_i(k+1)=x_i(k)=-1$

② $x_i(k)=-1, x_i(k+1)=1, s_i(k)=0$

对于情况①，表明 x_i 已进入稳定状态；对于情况②，网络继续演变时 $x_i=1$ 也将不会再变化，因为若 x_i 由 1 再变回-1，则有 $\Delta E<0$，它与 $E(k)$ 已收敛到常数相矛盾。所以网络最终将收敛到吸引子。

上述分析时假设 $w_{ii}=0$，实际上不难看出，当 $w_{ii}>0$ 时上述结论仍成立，而且收敛过程将更快。

上面证明时假设神经元节点取 1 和–1 两种状态，不难验证 x 取 1 和 0 两种状态时，上述结论也成立。

定理 1.4　对于离散型 Hopfield 网络，若按同步方式调整状态，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。

证明：前已求得

$$\Delta E(k)=E(k+1)-E(k)=-\Delta \boldsymbol{x}^{\mathrm{T}}(k)[\boldsymbol{W}\boldsymbol{x}(k)-\theta]-\frac{1}{2}\Delta \boldsymbol{x}^{\mathrm{T}}(k)\boldsymbol{W}\Delta \boldsymbol{x}(k)$$

$$=-\Delta \boldsymbol{x}^{\mathrm{T}}(k)\boldsymbol{s}(k)-\frac{1}{2}\Delta \boldsymbol{x}^{\mathrm{T}}(k)\boldsymbol{W}\Delta \boldsymbol{x}(k)=\sum_{i=1}^{n}\Delta x_i(k)s_i(k)-\frac{1}{2}\Delta \boldsymbol{x}^{\mathrm{T}}(k)\boldsymbol{W}\Delta \boldsymbol{x}(k)$$

前已证得，对于所有的 i，有 $-\Delta x_i(k)s_i(k)\leqslant 0$，因此只要 $\boldsymbol{W}$ 为非负定阵，即有 $\Delta E(k)\leqslant 0$，也即 $E(k)$ 最终将收敛到一个常数值，并按照如上面同样的分析可说明网络最终将收敛到吸引子。

可见对于同步方式，它对连接权矩阵 $\boldsymbol{W}$ 的要求更高了，若不满足 $\boldsymbol{W}$ 为非负定对称阵的要求，则网络可能出现自持振荡或极限环。

由于异步工作方式比同步工作方式有更好的稳定性能，因此实现时较多采用异步工作方式。异步工作方式的主要缺点是失去了神经网络并行处理的优点。

（2）吸引子的性质

定理 1.5　若 $\boldsymbol{x}$ 是网络的一个吸引子，且对于所有的 i，$\theta_i=0, \sum_{j=1}^{n}w_{ij}x_i\neq 0$，则–$\boldsymbol{x}$ 也一定是该网络的吸引子。

证明：由于 $\boldsymbol{x}$ 是吸引子，既 $\boldsymbol{x}=f[\boldsymbol{W}\boldsymbol{x}]$，从而有 $f[\boldsymbol{W}(-\boldsymbol{x})]=f[-\boldsymbol{W}\boldsymbol{x}]=-f[\boldsymbol{W}\boldsymbol{x}]=-\boldsymbol{x}$，即–$\boldsymbol{x}$ 也是网络的吸引子。

定理 1.6　若 $\boldsymbol{x}^{(a)}$ 是网络的吸引子，则与 $\boldsymbol{x}^{(a)}$ 的海明距离 $d_H(\boldsymbol{x}^{(a)},\boldsymbol{x}^{(b)})=1$ 的 $\boldsymbol{x}^{(b)}$ 一定不是吸引子，海明距离定义为两个向量中不相同的元素的个数。

证明：不失一般性，设 $x_1^{(a)}\neq x_1^{(b)}, x_i^{(a)}=x_i^{(b)}(i=2,3,\cdots,n)$。因为 $w_{11}=0$，所以有

$$x_1^{(a)}=f\left[\sum_{j=2}^{n}w_{1j}x_j^{(a)}-\theta_1\right]=f\left[\sum_{j=2}^{n}w_{1j}x_j^{(b)}-\theta_1\right]\neq x_1^{(b)}$$

所以 $\boldsymbol{x}^{(b)}$ 一定不是网络的吸引子。

推论：若 $\boldsymbol{x}^{(a)}$ 是网络的吸引子，且对于所有的 i，有 $\theta_i=0,\sum_{j=1}^{n}w_{ij}x_i\neq 0$，则 $d_H(\boldsymbol{x}^{(a)},\boldsymbol{x}^{(b)})=n-1$ 的 $\boldsymbol{x}^{(b)}$ 一定不是该网络的吸引子。

证明：若 $d_H(\boldsymbol{x}^{(a)},\boldsymbol{x}^{(b)})=n-1$，则 $d_H(-\boldsymbol{x}^{(a)},\boldsymbol{x}^{(b)})=1$。根据定理 1.5，$\boldsymbol{x}^{(a)}$ 是网络的吸引子，$-\boldsymbol{x}^{(a)}$ 也是网络的吸引子，根据定理 1.6，$\boldsymbol{x}^{(b)}$ 一定不是吸引子。

（3）吸引域

为了能实现联想记忆，对于每一个吸引子应该有一定的吸引范围，这个吸引范围便称为吸引域。下面给出较严格的定义。

① 若 $\boldsymbol{x}^{(a)}$ 是吸引子，对于异步方式，若存在一个调整次序可以从 $\boldsymbol{x}$ 演变到 $\boldsymbol{x}^{(a)}$，则称 x 弱吸引到 $\boldsymbol{x}^{(a)}$；若对于任意调整次序都可以从 $\boldsymbol{x}$ 演变到 $\boldsymbol{x}^{(a)}$，则称 $\boldsymbol{x}$ 强吸引到 $\boldsymbol{x}^{(a)}$。

② 对所有 $\boldsymbol{x}\in R(\boldsymbol{x}^{(a)})$ 均有 $\boldsymbol{x}$ 弱（强）吸引到 $\boldsymbol{x}^{(a)}$，则称 $R(\boldsymbol{x}^{(a)})$ 为 $\boldsymbol{x}^{(a)}$ 的弱（强）吸引域。

对于同步方式，由于无调整次序问题，所以相应的吸引域也无强弱之分。

对于异步方式，对同一个状态，若采用不同的调整次序，则有可能弱吸引到不同的吸引子。

3）连接权的设计

Hopfield 网络的权值不被训练，也不需要再学习。它的权值矩阵是事前利用 Lyapunov 函数的设计思想采用 Hebb 规则计算出来的。在这种网络中，不断更新的不是权值，而是网络中各神经元的状态，网络演变到稳定时各神经元的状态便是问题的解。为了保证 Hopfield 网络在异步方式工作时能稳定收敛，连接权矩阵 $\boldsymbol{W}$ 应是对称的。若要保证同步方式收敛，则要求 $\boldsymbol{W}$ 为非负定阵，这个要求比较高。因而设计一般只保证异步方式收敛。另外一个要求是对于给定的样本必须是网络的吸引子，而且要有一定的吸引域，这样才能正确实现联想记忆功能。

设给定 m 个样本 $\boldsymbol{x}^{(k)}(k=1,2,\cdots,m)$，为了实现上述功能，通常采用有监督的 Hebb 规则（用输入模式作为目标模式）来设计连接权，连接权可按以下两种情况进行计算。

（1）当网络节点状态为 1 和–1 两种状态，即 $\boldsymbol{x}\in\{-1,1\}^n$ 时，相应的连接权为

$$w_{ij}=\begin{cases}\sum_{k=1}^{m}x_i^{(k)}x_j^{(k)} & (i\neq j)\\ 0 & (i=j)\end{cases} \tag{1-56}$$

写成矩阵形式则为

$$\boldsymbol{W}=[\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)},\cdots,\boldsymbol{x}^{(m)}]\begin{pmatrix}\boldsymbol{x}^{(1)\mathrm{T}}\\ \boldsymbol{x}^{(2)\mathrm{T}}\\ \vdots\\ \boldsymbol{x}^{(m)\mathrm{T}}\end{pmatrix}-m\boldsymbol{I}$$

$$=\sum_{k=1}^{m}\boldsymbol{x}^{(k)}\boldsymbol{x}^{(k)T}-m\boldsymbol{I}=\sum_{k=1}^{m}(\boldsymbol{x}^{(k)}\boldsymbol{x}^{(k)\mathrm{T}}-\boldsymbol{I}) \tag{1-57}$$

式中，$\boldsymbol{I}$ 为单位矩阵。

证明：当输入为单个模式 $\boldsymbol{x}$ 时，若网络是稳定的，则稳定时输出 sgn($\boldsymbol{Wx}$)应该也是 $\boldsymbol{x}$，即

应有

$$\boldsymbol{x}=\mathrm{sgn}(\boldsymbol{W}\boldsymbol{x})$$

或

$$x_i=\mathrm{sgn}\left(\sum_{j=1}^{n}w_{ij}x_j\right)$$

根据上式，考虑到符号函数的性质，可知 x_i 与 $\sum_{j=1}^{n}w_{ij}x_j$ 同符号，因此有

$$x_i\sum_{j=1}^{n}w_{ij}x_j>0$$

而为了满足上式，权值可按有监督的 Hebb 规则（用输入模式作为目标模式）来设置，即

$$w_{ij}=\eta x_i x_j$$

这是因为，此时满足

$$x_i\sum_{j=1}^{n}w_{ij}x_j=x_i\sum_{j=1}^{n}\eta x_i x_j^2=\eta x_i^2\sum_{j=1}^{n}x_j^2=\eta x_i^2 n>0$$

当输入为 m 个样本 $\boldsymbol{x}^{(k)}(k=1,2,\cdots,m)$ 时，权值可根据以上直接推广为由下式来设置：

$$w_{ij}=\eta\sum_{k=1}^{m}x_i^{(k)}x_j^{(k)}$$

若取 $w_{ii}=0,\eta=1$，则上式可写为

$$w_{ij}=\sum_{\substack{k=i\\ i\neq j}}^{m}x_i^{(k)}x_j^{(k)}$$

证毕。

（2）当网络节点状态为 1 和 0 两种状态，即 $\boldsymbol{x}\in\{0,1\}^n$ 时，相应的连接权为

$$w_{ij}=\begin{cases}\sum_{k=1}^{m}(2x_i^{(k)}-1)(2x_j^{(k)}-1) & (i\neq j)\\ 0 & (i=j)\end{cases}\qquad(1\text{-}58)$$

写成矩阵形式则为

$$\boldsymbol{W}=\sum_{k=1}^{m}(2\boldsymbol{x}^{(k)}-\boldsymbol{b})(2\boldsymbol{x}^{(k)}-\boldsymbol{b})^{\mathrm{T}}-m\boldsymbol{I}\qquad(1\text{-}59)$$

式中，$\boldsymbol{b}=[1\quad 1\quad\cdots\quad 1]^{\mathrm{T}}$。

显然，上面所设计的连接权矩阵满足对称性的要求。

（3）下面进一步分析所给样本是否为网络的吸引子，这一点是十分重要的。下面以 $\boldsymbol{x}\in\{-1,1\}^n$ 的情况为例进行分析。

若 m 个样本 $\boldsymbol{x}^{(k)}(k=1,2,\cdots,m)$ 是两两正交的，即

$$\begin{cases}\boldsymbol{x}^{(i)\mathrm{T}}\boldsymbol{x}^{(j)}=0 & (i\neq j)\\ \boldsymbol{x}^{(i)\mathrm{T}}\boldsymbol{x}^{(i)}=n\end{cases}$$

则有

$$\boldsymbol{W}\boldsymbol{x}^{(k)}=\left(\sum_{i=1}^{m}\boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)\mathrm{T}}-m\boldsymbol{I}\right)\boldsymbol{x}^{(k)}=\sum_{i=1}^{m}\boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)\mathrm{T}}\boldsymbol{x}^{(k)}-m\boldsymbol{x}^{(k)}$$
$$=n\boldsymbol{x}^{(k)}-m\boldsymbol{x}^{(k)}=(n-m)\boldsymbol{x}^{(k)}$$

可见，只要满足 $n-m>0$，便有

$$f[\boldsymbol{W}\boldsymbol{x}^{(k)}]=f[(n-m)\boldsymbol{x}^{(k)}]=\boldsymbol{x}^{(k)}$$

也即 $\boldsymbol{x}^{(k)}$ 是网络的吸引子。

若 m 个样本 $\boldsymbol{x}^{(k)}(k=1,2,\cdots,m)$ 不是两两正交的，且设向量之间的内积为：$\boldsymbol{x}^{(i)\mathrm{T}}\boldsymbol{x}^{(j)}=\beta_{ij}$，显然 $\beta_{ii}=n,(i=1,2,\cdots,m)$。则有

$$\boldsymbol{W}\boldsymbol{x}^{(k)}=\sum_{i=1}^{m}\boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)\mathrm{T}}\boldsymbol{x}^{(k)}-m\boldsymbol{x}^{(k)}=(n-m)\boldsymbol{x}^{(k)}+\sum_{\substack{i=1\\i\neq k}}^{m}\boldsymbol{x}^{(i)}\beta_{ik}$$

取其中第 j 个元素

$$\boldsymbol{W}\boldsymbol{x}_j^{(k)}=(n-m)\boldsymbol{x}_j^{(k)}+\sum_{\substack{i=1\\i\neq k}}^{m}\boldsymbol{x}_j^{(i)}\beta_{ik}$$

若能使得对于所有的 j 有

$$n-m>\left|\sum_{\substack{i=1\\i\neq k}}^{m}x_j^{(i)}\beta_{ik}\right|$$

则 $\boldsymbol{x}^{(k)}$ 是网络的吸引子。上式右端可进一步化为

$$\left|\sum_{\substack{i=1\\i\neq k}}^{m}x_j^{(i)}\beta_{ik}\right|\leqslant\sum_{\substack{i=1\\i\neq k}}^{m}\left|\beta_{ik}\right|\leqslant(m-1)\beta_m$$

其中，$\beta_m\triangleq\left|\beta_{ik}\right|_{\max}$。进而若能使得 $n-m>(m-1)\beta_m$，即

$$m<\frac{n+\beta_m}{1+\beta_m}$$

则可以保证所有的样本均为网络吸引子。

m 个样本满足：

$$\alpha n\leqslant d_H(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)})\leqslant(1-\alpha)n$$

其中，$i,j=1,2,\cdots,m;i\neq j;0<\alpha<0.5$，则有

$$\left|\beta_{ij}\right|\leqslant n-2\alpha n=\beta_m$$

从而得出 m 个样本均为网络吸引子的条件为

$$m<\frac{2n(1-\alpha)}{1+n(1-2\alpha)}$$

注意，上式仅为充分条件。当不满足上述条件时，需要具体检验才能确定。

4）记忆容量

所谓记忆容量是指在网络结构参数一定的条件下，要保证联想功能的正确实现，网络所能存储的最大的样本数。也就是说，给定网络节点数 n，样本数 m 最大可为多少，这些样本向量不仅本身应为网络的吸引子，而且应有一定的吸引域，这样才能实现联想记忆的功能。

记忆容量不仅与节点数 n 有关，它还与连接权的设计有关，适当地设计连接权可以提高网络的记忆容量。记忆容量还与样本本身的性质有关，对于用 Hebb 规则设计连接权的网络，如果输入样本是正交的，则可以获得最大的记忆容量。实际问题的样本不可能都是正交的，所以在研究记忆容量时通常都假设样本向量是随机的。

记忆容量还与要求的吸引域大小有关，要求的吸引域越大，记忆容量便越小。一个样本向量 $\boldsymbol{x}^{(k)}$ 的吸引域可以看成以该向量为中心的球体。在该球体中的向量 $\boldsymbol{x}^{(s)}$ 满足：

$$d_H(\boldsymbol{x}^{(s)},\boldsymbol{x}^{(k)}) \leqslant \alpha n \qquad (0 \leqslant \alpha \leqslant 0.5)$$

式中，α 为吸引半径。

对于给定的网络，严格的分析并确定其记忆容量并不是一件很容易的事情。Hopfield 曾提出了一个数量范围，即 $m \leqslant 0.15n$。按照样本为随机分布的假设所做的理论分析表明，当 $n \to \infty$ 时，其记忆容量为

$$m < \frac{(1-2\alpha)^2 n}{2\ln n}$$

其中，α 为要求的吸引半径。

上面提到，当样本为两两正交时，可以有最大的记忆容量。对于一般的记忆样本，可以通过改进连接权的设计来提高记忆容量。下面介绍其中的一种方法。

设给定 m 个样本向量 $\boldsymbol{x}^{(k)}(k=1,2,\cdots,m)$，首先组成如下的 $n\times(m-1)$ 阶矩阵：

$$\boldsymbol{A}=[\boldsymbol{x}^{(1)}-\boldsymbol{x}^{(m)},\boldsymbol{x}^{(2)}-\boldsymbol{x}^{(m)},\cdots,\boldsymbol{x}^{(m-1)}-\boldsymbol{x}^{(m)}]$$

对 $\boldsymbol{A}$ 进行奇异值分解：

$$\boldsymbol{A}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$$

其中

$$\boldsymbol{\Sigma}=\begin{pmatrix}\boldsymbol{S} & 0\\ 0 & 0\end{pmatrix},\boldsymbol{S}=\mathrm{diag}(\sigma_1,\sigma_2,\cdots,\sigma_r)$$

式中，$\boldsymbol{U}$ 是 $n\times n$ 正交阵；$\boldsymbol{V}$ 是 $(m-1)\times(m-1)$ 正交阵；$\boldsymbol{U}$ 可表示成

$$\boldsymbol{U}=[\boldsymbol{u}_1,\boldsymbol{u}_2,\cdots,\boldsymbol{u}_r,\boldsymbol{u}_{r+1},\cdots,\boldsymbol{u}_n]$$

则 $\boldsymbol{u}_1,\boldsymbol{u}_2,\cdots,\boldsymbol{u}_r$ 是对应于非零奇异值 $\sigma_1,\sigma_2,\cdots,\sigma_r$ 的左奇异向量，且组成了 $\boldsymbol{A}$ 的值域空间的正交基；$\boldsymbol{u}_{r+1},\cdots,\boldsymbol{u}_n$ 是 $\boldsymbol{A}$ 的值域的正交补空间的正交基。

按如下方法组成连接权矩阵 $\boldsymbol{W}$ 和阈值向量 $\boldsymbol{\theta}$。

$$\boldsymbol{W}=\sum_{k=1}^{r}\boldsymbol{u}_k\boldsymbol{u}_k^{\mathrm{T}}$$

$$\boldsymbol{\theta}=\boldsymbol{W}\boldsymbol{x}^{(m)}-\boldsymbol{x}^{(m)}$$

显然，按上述方法求得的连接权矩阵是对称的。因而可以保证异步工作方式的稳定性，下面进一步证明给定的样本向量 $\boldsymbol{x}^{(k)}(k=1,2,\cdots,m)$ 都是吸引子。

由于 $\boldsymbol{u}_1,\boldsymbol{u}_2,\cdots,\boldsymbol{u}_r$ 是 $\boldsymbol{A}$ 的值域空间的正交集，所以 $\boldsymbol{A}$ 中的任一向量 $\boldsymbol{x}^{(k)}-\boldsymbol{x}^{(m)}(k=1,$