目　录

第1章 绪 论

探讨人脑思维的奥秘是人类长期以来的梦想。自从公元前亚里士多德时代开始，人们就开始研究具有思维能力的机器。第一台电子计算机的问世使这方面的研究有了实质性的进展。1956年，人工智能技术的出现，使人们又朝思维机器的研究方向进了一步。现在，神经网络技术又为人们进一步了解人脑思维的奥秘开辟了新的途径。

1.1 当代信息高科技热点——神经网络

80年代后期，在美国、日本等一些工业发达国家里，掀起了一股竞相研究开发神经网络(Neural Networks，简称NN)的热潮。1987年6月，首届国际神经网络学术会议在美国加利佛尼亚州召开，到会代表有1600余人。在会上成立了国际神经网络学会(International Neural Network Society)。接着于1988年，由当今世界著名的三位神经网络学家，即日本东京大学的Shunichi Amari(甘利俊一)教授，美国波士顿大学的Stephen Grossberg教授和芬兰赫尔辛基技术大学的Teuvo Kohonen教授，主持创办了世界第一份神经网络杂志《Neural Networks》。随后，国际电气工程师与电子工程师学会(IEEE)也成立了神经网络协会并出版神经网络刊物。

近几年来，在神经网络这个涉及多种学科的新的高科技领域中，吸引了众多的神经生理学家，心理学家，数理科学家，计算机与信息科学家以及工程师和企业家等。大量的有关神经网络机理、模型、算法特性分析，以及在各方面应用的学术论文像雨后春笋般在报刊杂志上和许多国际学术会议中涌现，神经网络以及建立在神经网络原理基础上的神经计算机(Neuro Computer)成为当代高科技领域中方兴未艾的竞争热点。

面对世界各国出现的神经网络开发研究热潮，我国也及时地迈出了急追直赶的步伐。从1986年到1988年，先后在北京召开了脑的工作原理研讨会，神经网络战略研讨会和学习与识别神经网络国际讨论会。1989年10月和11月，分别在北京和广州召开了神经元网络及其应用学术讨论会和第一届全国信息处理——神经网络学术会议。

1990年12月，由我国八个学会(即中国电子学会、计算机学会、人工智能学会、自动化学会、通信学会、物理学会、生物物理学会和心理学会)联合在北京召开"中国神经网络首届学术大会"。这个规模空前的盛会，以"八学会联盟，探智能奥秘"为主题，收到了来自各方面的论文300余篇，从而开创了我国神经网络及神经计算机方面科学研究的新纪元。

神经网络信息处理的原理方法和神经计算机的研制开发应用，是一个涉及生物、医学、心理学、认知学、信息论、计算机、数学、物理和微电子技术等多种学科的综合性高科技。以1988年在美国波士顿召开的国际神经网络学会年会为例，会上研究讨论的中心议题包括：

- 神经计算机(Neuro Computer)；
- 联想学习(Associative Learning)；
- 自组织(Self-Organization)；

- 局部回路神经生理(Local Circuit Neurobiology)；
- 网络动态分析(Analysis of Network Dynamics)；
- 认知信息处理(Cognitive Information Processing)；
- 视觉信息处理(Vision and Image Processing)；
- 语音与语言(Speak and Language Control)；
- 传感电机控制与机器人(Sensory-motor Control and Robotics)；
- 模式识别(Pattern Recognition)；
- 组合优化(Conbinational Optimization)；
- 电子实现(Electronic Implementation)；
- 光学实现(Optical Implementation)；
- 应用(Application)。

上述这些会议中心议题，大致覆盖了神经网络这门新兴学科领域中的重点研究方向。

1.2 神经网络的发展史

从人脑的生理结构出发来研究人的智能行为，模拟人脑信息处理的过程，即人工神经网络的研究，经历了一条曲折的路程。大致分为兴起、萧条和兴盛三个时期。

早在 1943 年，心理学家 Mcculloch 和数学家 Pitts 在数学生物物理学会刊 Bulletin of Mathematical Biophysics 上发表文章，总结了生物神经元的一些基本生理特性，提出了形式神经元的数学描述与结构方法，即 M-P 模型。在 M-P 模型中，赋予形式神经元的功能较弱，但网络的计算能力巨大，这种巨大的计算潜力在于网络中足够多的神经元以及神经元之间丰富的联系，同时神经元还具有并行计算的能力。*M-P* 模型的提出兴起了对神经网络的研究。

1949 年心理学家 D. O. Hebb 提出神经元之间突触联系强度可变的假设。他认为学习过程是在突触上发生的，突触的联系强度随其前后神经元的活动而变化。根据这一假设提出的学习率为神经网络的学习算法奠定了基础。

50 年代末，Rosenblatt 提出感知机，第一次把神经网络的研究付诸工程实践。这是一种学习和自组织的心理学模型，它基本上符合神经生物学的知识，模型的学习环境是有噪声的，网络构造中存在随机连接，这符合动物学习的自然环境，当时人们对神经网络研究过于乐观，认为只要将这种神经元互连成一个网络，就可解决人脑思维的模拟问题，以后碰到了理论上和实现技术上的困难，加上其它因素的影响，使得对神经网络的研究进入了低潮。

60 年代，美国著名人工智能学者 Minsky 和 Papert 对 Rosenblatt 的工作进行了深入的研究，写了很有影响的《感知机》一书，指出感知机的处理能力有限，甚至连 XOR 这样的问题也不能解决，并指出如果引入隐含神经元，增加神经网络的层次，可提高神经网络的处理能力，但是研究对应的学习方法非常困难。加以当时人工智能的以功能模拟为目标的另一分支出现了转机，产生了以知识信息处理为基础的知识工程，给人工智能从实验室走向实用带来了希望。同时，微电子技术的发展，使传统计算机的处理能力有很大提高，数字计算机的发展使当时科学界普遍认为它能解决一切问题，包括模式识别、机器人控制等。因而不必去寻找新的计算理论与实现方法。而且，当时的工艺水平还未能达到制作实用的具有足够规模的神经网络，用分离的电子管即使是晶体管所制作的神经网络也只能作示教性的表演。这些因素的共同作用，促

使人们降低了对神经网络研究的热情，从而使神经网络进入萧条时期。

不过，还是有不少学者继续对神经网络进行研究，仍取得一些积极的成果。其中包括 Arbib 的竞争模型，1977 年 Kohonen 的自组织映射模型，Grossberg 的自适应谐振模型和 Fukushima 的新认知机等。特别是有的学者提出了连接机制和并行分布处理概念等，具有较大影响。

到了 80 年代中期，神经网络的研究进入了“柳暗花明又一村”的新境界，一个竞相研究神经网络和设计构造神经计算机的热潮在世界范围内掀起。产生这种转折变化的一个重要原因，是美国加州理工学院生物物理学家霍普菲尔德采用全互连型神经网络模型，利用所定义的计算能量函数，成功地求解了计算复杂度为 NP 完全型的旅行商问题。这项突破性的进展引起了广大学者对神经网络潜在能力的高度重视，从而掀起了研究神经网络信息处理方法和研制神经计算机的热潮。

经过近半个世纪的研究探索，人们开始懂得，人工智能的发展，与对人脑机理的研究和认识学科的发展密不可分，因而出现包括人工智能（Artificial Intelligence），脑模型（Brain Model）和认知科学（Cognitive Science）的所谓 ABC 理论。

1.3 神经细胞及神经网络

一个神经细胞的构造如图 1-1 所示，主要包括细胞体、树突、轴突和细胞之间相互关联的突触。

细胞体是由细胞核、细胞浆、细胞膜等组成。在高等动物的神经细胞中，除了特殊的无“轴突”神经元外，一般每个神经元都由胞体的轴丘处发出一根粗细均匀，表面光滑的突起，长度从几个 μm 到 1m 左右，称为轴突，它的功能是传出从细胞体来的神经信息。树突为细胞体向外伸出的很多其它突起，它们像树枝一样向四处分散开来，在胞体附近比轴突粗得多，但离开细胞体不远马上就很快分支变细，形成无数粗细不等的树突。它们的作用是向四方收集由其它神经细胞来的信息。信息流是从树突出发，经过细胞体，然后由轴突输出。突触是两个细胞之间连接的基本单元，主要有：

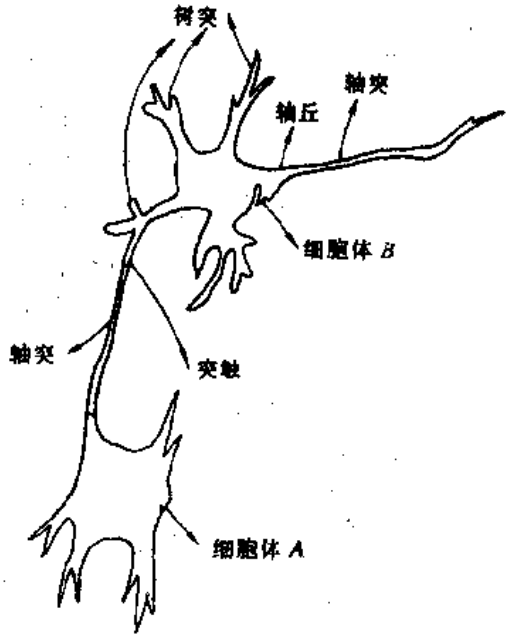


图 1-1 神经细胞的示意图

(1)一个神经细胞的轴突与另一个神经细胞的树突发生接触；

(2)一个神经细胞的轴突与另一个神经细胞的胞体接触。

突触包含两个部分：一个为突触前成分，表示在轴突的末梢；一个是突触后成分，为树突的

始端，或细胞体与轴突末端接触的部分。突触的联接如图 1-2 所示：

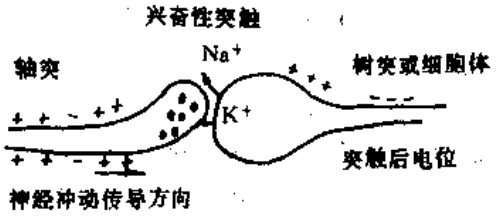


图 1-2　突触联接的示意图

根据突触对下一个神经细胞的功能活动的影响，突触又可分为兴奋性的和抑制性的两种。兴奋性的突轴：当神经细胞有冲动传递时，它能使突触后成分去极化，产生兴奋性突触后电位，可能引起下一个神经细胞兴奋。抑制性的突触能使突触后成分超极化，产生抑制突触后电位，使下一个神经细胞抑制。

神经细胞单元的信息是宽度和幅度都相同的脉冲串。脉冲串的间隔是随机变化的。如某个神经细胞兴奋，其轴突输出的脉冲串在单位时间内的平均频率高；如神经细胞抑制时，脉冲发放率减少，甚至无脉冲发放。信息传递在突触处主要是发生化学和电的变化，即在化学突触的传递过程中，突触前成分囊泡里的神经递质被释放到突轴后膜，当脉冲到来时，贮存在突触囊泡内的神经递质进行排放，这样改变了突触后膜对钠离子(Na^+)、钾离子(K^+)和氯离子(Cl^-)的通透性，使突触后神经细胞相位发生电位变化。

兴奋性突触在脉冲刺激下，对下一个神经细胞产生兴奋性突触后的电位变化，抑制性突触在脉冲刺激下产生抑制性突触后的电位变化。很多神经细胞通过各自的突触对某一个神经细胞的作用，都形成该神经细胞的后电位变化，电位的变化是可以累加的，该神经细胞后电位是它所有的突触产生的电位总和，当该神经细胞的后电位升高到超过一个阈值，就会产生一个脉冲，从而总和的膜电位直接影响该神经细胞发放的脉冲数。一般说，每个神经细胞的轴突大约联接 100～1000 个其它神经细胞，神经细胞的信息就是这样从一个神经细胞传递到另一个神经细胞。

在人脑中大约有 140 亿个神经细胞单元，根据 Stubbz 估计，这些神经细胞被安排进约 1000 个主要模块内，每个模块有上百个神经网络，每个网络约有 10 万个神经细胞，信息的传递是从一个神经细胞传递到另一个神经细胞，从一种类型的神经细胞传递到另一类神经细胞，从一个网络传递到另一个网络，有时也从一个模块传递到另一个模块。

1.4　脑神经信息活动的特征

如果将人脑神经活动的特点与现行的计算机的工作方式作一对照比较，就可以看出以下的重大差别：

(1)人在识别一幅图像或作出一项决策时，并不像现行计算机那样按事先编好的程序一条一条地执行指令，而是把存在于脑中多方面的知识和经验同时迸发出来，迅速地作出解答。人脑中多达 10^{10}～10^{11} 数量级的神经元提供了巨大的空间存储容量，存储有大量的知识和经验，在需要时能以很高的反应速度作出处理判断。这就是说，巨量并行性是人脑信息活动的一个重要特点。

(2)人脑中信息存储和信息处理是合在一起的，而不像现行计算机那样，存储地址和存储内容是彼此分开的。由于人脑神经元兼有信息处理和存储功能，所以在进行回忆时，不但不存在先找存储地址而后再调出所存内容的问题，而且可以由一部分内容恢复全部内容。

信息处理和存储单元结合在一起，是人脑信息处理的又一特点。

(3)人脑不像现行计算机那样，只能被动地执行已编好的程序，而是能够通过内部自组织，自学习的能力不断适应外界环境，有效地处理各种模拟的、模糊的或随机的问题。

人脑这种活的自组织自学习功能，能不断地积累知识经验，适应新的环境要求，这更是传统电脑望尘莫及的。当然，在人脑活动中，还有感情、联想、灵感和创造等高层次的信息活动，这些都包含着有待进一步认识和阐明的机理作用。

表 1-1 列出了传统计算机和人脑间的比较。从表中可以看出，在那些需要根据逻辑推理进行精确计算的场合，传统计算机能比人脑更有成效地工作。而在那些需要根据多种因素和经验迅速作出综合判断，以求得满意解的场合，人脑要比传统计算机强得多。

表 1-1 传统计算机与人脑比较

比较内容	传统计算机	人脑
基本单元	半导体元件	神经元
单元数目	$10^5 \sim 10^7$	$10^{10} \sim 10^{11}$
信号形式	电脉冲	活动电位
动作速度	10^{-9}s	10^{-3}s
记忆容量	10^{10}比特	$10^{13} \sim 10^{20}$比特
记忆形式	按地址记忆	按内容联想
故障率	5×10^{-22}	5×10^{-21}
信息处理方式	数字集中处理	模拟分布处理
系统结构	串行处理	并行处理
抗干扰性	低	高
容错能力	弱	强
信息再现性	完全	不完全
工作方式	被动执行程序	主动学习创新

由此不难理解，研究人脑神经的结构特点和活动机制，并在此基础上模拟仿造出人工神经网络和神经计算机，绝对不是要否定和取代传统计算机在精确数值计算方面的巨大优势，而是作为应用于模式识别，组合优化和决策判断方面的扩展和补充。考虑到人类日常生活和工作中绝大多数情况下要处理的信息都是模拟的、模糊的和随机的信息，因此这种扩展和补充将会带来巨大的效益。

1.5 人工神经网络及其特征

人工神经网络是采用物理可实现的器件或采用现有的计算机来模拟生物体中神经网络的某些结构与功能，并反过来用于工程或其它的领域。

图 1-3 是一个人工神经元的示意图。图中 $x_1, x_2, \cdots, x_n$ 表示其它神经元的轴突输出，w_1，$w_2, \cdots, w_n$ 为其它 n 个神经元与第 i 个神经元的突轴联接，其符号的正负分别表示为兴奋性突

触和抑制性突触；其数值的大小表示突触的不同的化学变化情况。

每个人工神经元满足：

$$s_i = \sum_{j=1}^{n} w_j x_j - \theta_i \tag{1-1}$$

$$u_i = g(s_i) \tag{1-2}$$

$$y_i = f(u_i) \tag{1-3}$$

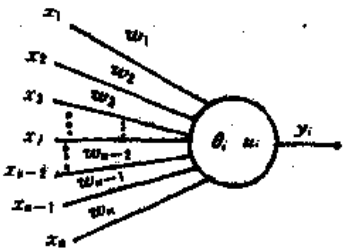


图 1-3 人工神经元的示意图

其中式(1-1)表示神经元 i 突触后电位的累加值，θ_i 为阈值。式(1-3)中，u_i 为细胞 i 的状态，y_i 为神经元 i 的输出。

人工神经网络是对生物神经系统的模拟，其信息处理功能是由网络的单元(神经元)的输入输出特性(激活特性)，网络的拓扑结构(神经元的连接方式)所决定的。按突触修正假说，神经网络在拓扑结构固定时，其学习归结为连接权的变化。

下面比较一下传统计算机和人工神经网络对问题求解的方式。传统计算机需要人将解题的步骤进行编程并输入，最后由计算机一步一步地得到问题的解；相反地，人工神经网络(以下简称神经网络)可以经过训练来解答问题。换句话说，神经网络给自身编程去求解问题。神经网络是一个由大量简单神经元连接成的复杂的网络，这样一个网络可以由硬件或软件构成。神经网络的训练(学习)由图 1-4 示意。训练一个神经网络包括用一系列的输入例子和理想的输

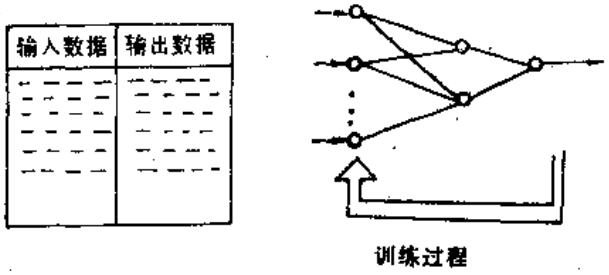


图 1-4 神经网络的学习过程

出作为训练“样本”，根据一定的训练算法对网络进行足够的训练，使得神经网络能够学会包含在解中(样本中的理想输出)的基本原理。当训练完成后，该解(由训练所确定的网络)可以用来求解相同的问题。

虽然人工神经网络与真正的神经网络有很多差别，但由于它吸取了生物神经网络的部分优点，因而有其固有的特点：

(1)人工神经网络在结构上与目前的计算机根本不同，它是由许多小的处理单元互相联接而成，每个处理单元的结构如图 1-3 所示，并由方程(1-1)、(1-2)、(1-3)来描述。每个单元的功能简单，但大量简单的处理单元集体的、并行的活动得到预期的识别、计算的结果，具有较快的速度。

(2)人工神经网络具有很强的容错性，即局部的或部分的神经元损坏后，不影响全局的活动。

(3)人工神经网络所记忆的信息是存储在神经元之间的权中，从单个权看不出其储存信息的内容，因而是分布式的存贮方式。

(4)人工神经网络具有十分强的学习功能，人工神经网络的连接权和连接结构都可以通过学习而得到。

1.6 神经网络研究的意义及应用前景

1.6.1 现行计算机编程的瓶颈问题

限制计算机进行应用的因素大多数场合不再是运行速度和程序的长度，这是由于台式PC机比10年前大型机的功能还要强得多，现在的问题是开发软件的困难。程序是复杂的，并且编制和保留这些程序的费用都很昂贵。甚至在很多问题中，有一个更基本的问题：即我们不知道让计算机去做什么。比如说，我们不认识人群中朋友的面孔，不知道如何进行决策，所以写不出让计算机执行的步骤。这是因为对许多困难问题，必须从经验中学习。例如一个股票经纪人，从自身的股票买卖中增长知识和经验。尽管有大量的因素影响他，但他的决择取决于他以前经历过的相同股市的例子。如果把计算机认为是进行信息处理的"黑箱子"，用我们希望它做的例子来训练它，就能克服编程瓶颈问题。所以说，神经计算是解决这一问题的方法。

1.6.2 神经网络研究的意义

智能信息处理的研究并不是始于今日的新鲜事物，这项研究至少可以追溯到50年代人工智能的初创期，更早些可以追溯到图灵自动机理论。从其研究历史来看，它主要是沿着两条途径展开的：一条是基于心理学的符号处理方法(强调脑的功能模拟)；另一条是基于生理学的模式处理方法(即神经网络)，强调模拟脑的神经系统结构的基础上实现脑的功能。

符号处理方法主要是计算机模拟人脑的思维功能，重点研究的是机器的思维问题，解决问题的关键在于知识的表示、获取、存储和使用。各种专家系统在其发展中碰到了许多难以克服的困难。

神经网络从脑的神经系统结构出发来研究脑的功能，研究大量简单的神经元的集团信息处理能力及其动态行为。现在神经网络对30多年来一直困扰计算机科学和符号处理的一些难题可以得到比较令人满意的解答，特别是对那些时空信息存储及并行搜索，自组织相联存储，时空数据统计描述的自组织以及从一些相互关联的活动中自动获取知识等一般问题的求解，更显示出了其独特的能力，由此而引起了智能研究者们的广泛关注，并普遍认为神经网络方法适合于低层次的模式处理。

符号处理与神经网络是一种互补关系。神经网络的研究重点在于模拟和实现人的认识过程中的感知觉过程、形象思维、分布式记忆和自学习自组织过程。而符号处理则侧重于模拟人的逻辑思维。因此，神经网络与符号处理相结合，可能会使人们对人的认知过程有一个较全面的理解。在这一领域中的任何一项基础理论上的进展，必将对计算机科学和智能产业产生实际的影响。

神经网络的数学理论本质上是非线性数学理论。因此，现代非线性科学方面的进展必将推动神经网络的研究；同时，神经网络也会对非线性科学提出新课题。

神经网络在国民经济和国防科技现代化建设中具有广阔的应用领域和应用前景。关于智能的模拟和机器再现，必将开始发展出一代新兴产业。因此，我们必须对这一领域的进展密切

注意，同时积极加强研究和开拓应用。

1.6.3 神经网络的应用前景

下面简要介绍神经网络的一些主要应用领域。

(1)传感器信息处理。

传感器信息处理涉及到两个主要的问题：模式预处理变换和模式识别。预处理变换接受一种形式模式，可以应用神经网络把它们转换为更多的想要的或可用的形式模式。比如：图像的压缩/扩充，图像的边缘抽取，图像对比增强，图像或相应的基础函数(Fourier，Fourier-Mellin，Gabar)扩充和模式噪音压缩。模式识别则是把一模式映射到其它类型或类别的操作。这可以是固定的静态映射，或许是更复杂的操作。神经网络可以执行许多的模式预处理变换或模式识别操作，可以处理静态模式(固定的图像，固定的能谱)和动态模式(动态的视频图像，连续的语音，声纳和雷达的多普勒频率)。

(2)信号处理。

神经网络还被广泛地应用于信号处理，如目标检测，杂波去噪声或畸变波形的恢复，雷达回波的多目标分类，运动目标的速度估计，多目标跟踪等。神经网络也可用于多探测器信号的融合(Fusion)，即对多个探测器收集到的信号进行处理，尽可能获取有关被测目标的完整信息。Mitch Eggers 和 Tim Khuon 利用神经网络检测空间中卫星飞行动作是稳定、倾斜、旋转还是摇摆四个状态，正确率可达 95%。概括地说，神经网络在信号处理领域主要应用于：自适应信号处理(自适应滤波、时间序列预测、谱估计、阵列处理、消除噪声、检测等)，非线性信号处理(非线性滤波、非线性预测、非线性谱估计、非线性编码等)。

(3)自动控制。

早在 1962 年，Widrow 就表明一个神经网络可以成功地学会平衡一个干扰抑制器的控制算法，并提出了著名的 LMS 算法。

Grossberg/Kuperstein 的视觉运动控制神经网络，能够执行传感器表面一个图像传感器的反馈控制和图像平面的非线性关系的计算，并能把图像传感器瞄准到正在运动的指定客体上。显然，这可以用到机器人的摄像机控制上。而且还可以应用到诸如火炮之类的武器系统中去。

神经网络还被用于机器人控制中，比如机械手的控制等。在鱼雷的控制上，有的也采用了神经网络方法。

(4)知识处理。

神经网络可以从数据中自动地获取知识，逐步地把新知识结合到其映射函数中去，并执行逻辑的假设检验。这种能力使得神经网络非常适合于处理某类知识，特别是不精确的知识。Anderson 的知识处理神经网络，通过将知识编为长的属性向量码来进行工作。这可以有效地处理矛盾与丢失的信息。在矛盾的情况下，在基于"证据权"做出决策(即选择具有最多的支持例子作为响应)。在丢失信息的时候，系统在现有的属性之间的已知联系的基础上进行猜测。这个系统的一个缺点是要有一个"硬"的知识库，即构造系统的数据必须是精确的。

Kosko 的模糊认知映射系统，是以神经网络的形式实现类图结构，它能够存储作为变元概念的客体之间的因果关系。模糊认知映射可以处理不精确的，矛盾的甚至是错误的数据。

(5)运输与通信。

运输与通信问题在国民经济中有着极为重要的现实意义。运输与通信问题的关键是如何在运输网或通信网中来调度货物或信息以达到最为经济。

最优的调度算法是一个NP完全性问题。而神经网络法则可以根据运输网或通信网中当前及以前的货物/信息情况，最佳地(或局部最佳地)来调度网中的货物源/信息源，以达到货物/信息在网中的传递最为经济的目的。

Hopfield模型可用来求解"旅行推销商问题"，但其解只是局部最优解。Bruck则证明用Hopfield模型可以求解最小割集问题。

(6)其它问题。

除了上面所讨论的一些应用领域外，神经网络在下面的这些领域也有着广泛的应用前景。

①零售分析。用神经网络来分析各种商品的零售量及价格。

②信用分析。

③航空与航天。

④医用诊断系统。

我们相信，随着神经网络和神经计算机在理论和实践方面的不断发展完善，将为它们的实际应用开辟更广阔的天地。

第 2 章　神经网络的计算模型

建立神经模型时，由于对神经元生理现象的性质进行观察的着重点不同，可以考虑不同的神经网络模型。

按照网络的性能即可分为连续型神经网络和离散型神经网络，又可分为确定型网络和随机型神经网络；按网络结构分，可分为反馈型神经网络和前馈型神经网络；按学习方式分可分为有导师学习和无导师学习(自组织学习)。

人工神经网络的结构，算法将在以后逐一介绍。这里从神经元输入输出关系上来归纳一下。神经元的关系满足方程(1-1)、(1-2)、(1-3)，神经元状态 u_i 和输出 y_i 的关系是一个非线性关系，一般有下面几种，如图 2-1 所示。

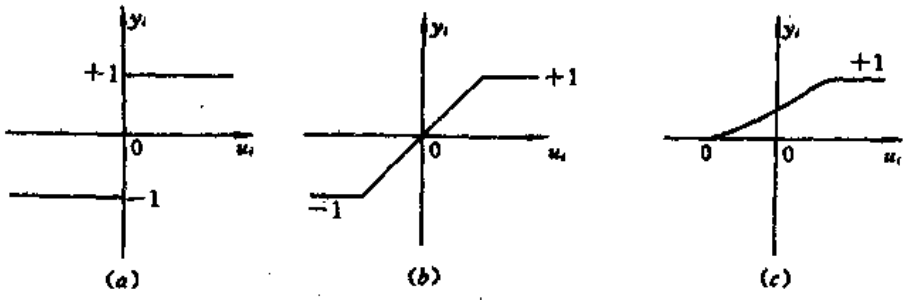


图 2-1　神经元的输入输出关系

(a)硬函数；　(b)分段线性函数；　(c)非线性单调上升函数

图 2-1*a* 表示神经元的兴奋活动输出值 y_i 是一个二值变量，如果在方程(1-1)中 x_j 也是二值变量，那么这种神经网络为离散的网络，神经元的输出可以是±1，也可以是 0、1 二值。这种关系称为硬函数，例如 Perceptron(感知器)模型，M-P 模型以及 Hopfield 在 1982 年所提出的模型，都是属于二值变量的硬函数。图 2-1*b* 表示输入输出关系是分段线性关系，当然其值也可以从 0 到 1。图 2-1*c* 是一个单调上升的光滑的非线性函数，可以用式(2-1)或(2-2)来表示。

$$y_i = f(u_i) = \frac{1}{1 + e^{-u_i}} \tag{2-1}$$

$$y_i = f(u_i) = \frac{1}{2}(1 + \text{th}\,\frac{u_i}{2}) \tag{2-2}$$

在 Back-Propagation 的模型中都是用这一类输入输出变换函数。在方程(1-2)中，$u_i = g(s_i)$，在多数人工神经网中，g 函数为一个线性函数，有的简化为 $u_i = s_i$，但在有些模型中，s_i 与 u_i 满足一个状态方程。

2.1 简单人工神经元模型

2.1.1 M-P 模型

M-P 模型最初是由 Mcculloch 和 Pitts 提出的，它是由固定的结构和权组成，其结构如图 2-2 所示。

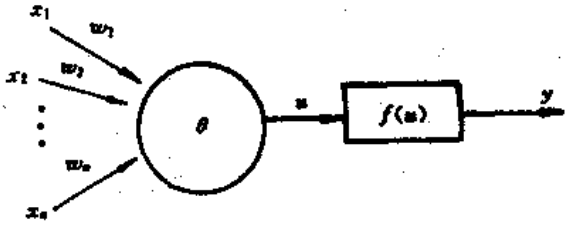


图 2-2 简单人工神经元 M-P 模型

M-P 模型神经元的输入为一个 n 维的实数向量 $X\in R^n$，其权 W 也是一个 n 维的矢量。其中 $X=(x_1,x_2,\cdots,x_n)^T$，权向量 $W=(w_1,w_2,\cdots,w_n)^T$ 神经元本身具有阈值 θ。输出 y 是一个二值变量。从输入与输出关系看，它们满足线性关系

$$u=\sum_{i=1}^{n}w_ix_i-\theta \tag{2-3}$$

$$y=\mathrm{sgn}(u)=\begin{cases}1, & u\geqslant 0\\ 0, & u<0\end{cases} \tag{2-4}$$

M-P 模型中的权 $w_i=1$，这种模型是早期提出的，可以用来表示一些逻辑关系。后来 M-P 模型中权分为兴奋性突触权($w_i=1$)和抑制性突触权($w_j=-1$)两类。如抑制性突触被激活，则神经元被抑制，输出为零，而兴奋性突触被激活，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，输出为 1。它的结构由图 2-3 所示。

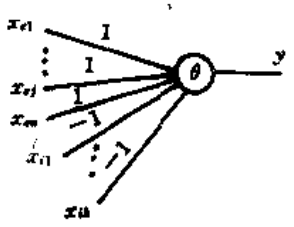


图 2-3 具有±1 突触的模型

其中 x_{ej} 为兴奋性突触的输入，$j=1,2,\cdots n$，x_{ik} 为抑制突触的输入，$k=1,2,\cdots,n_1$，$y=\mathrm{sgn}(u)$，则输入输出关系式为

$$y=\begin{cases}1, & \sum\limits_j x_{ej}-\sum\limits_k x_{ik}\geqslant\theta\\ 0, & \sum\limits_j x_{ej}-\sum\limits_k x_{ik}<\theta\end{cases} \tag{2-5}$$

从上面的介绍可以看出，M-P 模型神经元的特点是：

(1)多输入-单输出方式；

(2)阈值作用；

(3)输出与输入的两态(兴奋和抑制)；

(4)每个输入通过数值来表征它对神经元的耦合程度(如无耦合 $w_i=0$)。

M-P 网络的权、输入、输出都是二值变量，这同由逻辑门组成的逻辑式的实现区别不大，又由于它的权无法调节，因而现在很少有人单独使用。

2.1.2 连续的神经元模型

为反映神经元状态参数连续变化的性质，常用一阶非线性微分方程来模拟生物神经元膜电位随时间变化的规律，即

$$\tau\frac{\mathrm{d}u}{\mathrm{d}t}=-u(t)+\sum_{j=1}^{n}w_jx_j(t)-\theta \tag{2-6}$$

$$y(t)=f(u(t))$$

神经元的结构如图 2-2 所示。但输入量 $x_1,x_2,\cdots,x_n$ 为模拟量。式(2-6)中，τ 为时间常数，θ 为静止膜电位，$f(u)$为输入-输出函数。

图 2-4 表示函数 f 的四种可能形式。

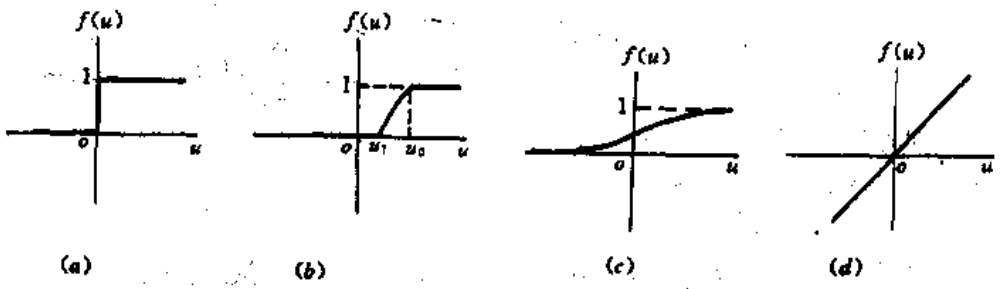


图 2-4 神经元的输入—输出函数表达

(a)阶跃函数；(b)分段线性函数；(c)S 函数；(d)恒等函数

需要说明的是，图 2-4 中 Sigmoid 函数由式(2-1)表示。

图 2-4 中(a)为阶跃函数

$$f(u)=\begin{cases}1, & u\geqslant 0\\ 0, & u<0\end{cases} \tag{2-7}$$

图 2-4 中(b)为分段线性函数

$$f(u)=\begin{cases}1, & u\geqslant u_0\\ au+b, & u_1\leqslant u<u_0\\ 0, & u<u_1\end{cases} \tag{2-8}$$

式中 a、b 为常数。图 2-4 中(d)为恒等线性函数

$$f(u)=u \tag{2-9}$$

2.2 感知器模型

在 M-P 模型和 Hebb 学习规则的基础上，1957 年 Rosenblatt 提出了具有学习能力的感知器模型。感知器的基本元素称为阈值逻辑单元(Threhold Logic Unit，简称 TLU)，它与 M-P 模型形式类似。所不同的是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，这就保证了它具有学习的能力。感知器可以对输入的样本矢量进行分类，而且多层感知器在某些样本点上对函数任意逼近。感知器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它前馈网络(即网络的信息只能从输入单元到它上面一层的单元，结构是分层的)的基础，我们可以从感知器的讨论为其它网络的分析提供依据。

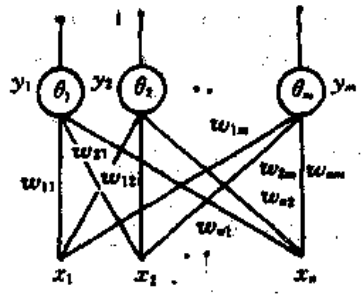


图 2-5 单层感知器网络

2.2.1 单层感知器网络

图 2-5 所示为一个单层感知器网络，输入向量 $X=(x_1,x_2,\cdots,x_n)^T$，输出向量 $Y=(y_1,y_2,\cdots,y_m)^T$，网络由 m 个神经元组成。联接权 w_{ij} 为输入节点 i 与第 j 个神经元之间的耦合系数，其中 $i=1,2,\cdots,n;j=1,2,\cdots,m$。

2.2.1.1 单层感知器的分类功能

下面简单分析一下单层感知器网络对样本分类的几何意义：

网络第 j 个神经元输出为

$$y_j=f\left(\sum_{i=1}^{n}w_{ij}x_i-\theta_j\right)$$

$$f(u_j)=\begin{cases}1, & u_j\geqslant 0\\ -1, & u_j<0\end{cases} \tag{2-10}$$

如果输入向量 X 有 k 个样本，即 $X^p,p=1,2,\cdots,k$，把样本 X^p 看作为 n 维空间的一个矢量，那个 k 个样本就是输入空间的 k 个矢量。由于单个神经元的输出 y_j 只有两种可能，即+1 或−1。这样，方程(2-10)就把这 n 维输入空间分为两个子空间，其分界线为 $n-1$ 维的超越平面。通过调节权 $w_{ij},i=1,2,\cdots,n$ 以及 θ_j 可以改变这个 $n-1$ 维超越平面的位置以达到对样本的正确划分。推而广之，单层感知器网络的 m 个神经元就可以产生 m 个独立的 $n-1$ 维超越平面。那么，这样形成的超平面是否能够完成对 k 个样本的正确分类呢？

以一个两维输入空间为例，如图 2-6 所示，输入矢量 $X=(x_1,x_2)^T$，权矢量 $W=(w_1,w_2)^T$，则输出 $y=f(w_1x_1+w_1w_2-\theta)$。在两维的输入空间中用“。”代表 A 类样本，集中在平面的左上角，用“·”表示 B 类样本，集中在平面的右下角。

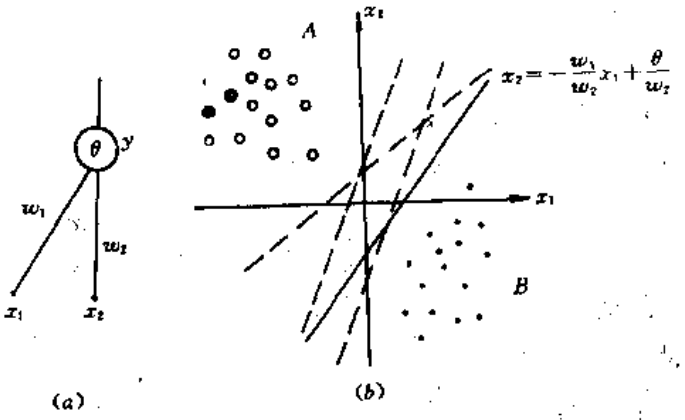


图 2-6 二维单层感知器样本分类

(a)二维单个神经元；(b)二维单层感知器在状态空间中的划分

我们希望找到一根直线，把 A、B 两类样本分开，其分界线为

$$x_2=-\frac{w_1}{w_2}x_1+\frac{\theta}{w_2} \tag{2-11}$$

从图 2-6 中可以看出，如果 A、B 两类样本是线性可分的，而且如图那样有一段距离，那么式(2-11)的解有无数个，如图 2-6(b)中虚线所示，分类的实质在于：根据适当的算法确定合适的

w_1,w_2 和 θ,使得当输入矢量 $x=(x_1,x_2)^T$ 属于 A 类时,神经元输出 $y=1$;而输入量属于 B 类样本时,$y=-1$。这样,网络就完成了正确的分类。这样的分类被称为线性分类,即作一条直线(或一个超平面)将样本分成两大类。单层感知器只能进行简单的线性分类,甚至不能解决简单的异或问题。

用来求解异或问题的神经网络结构如图 2-6(a)所示。其输入-输出样本表列如图 2-7(a)所示。另外,异或问题从几何意义上看,输入向量相当于二维平面上的一个正方形的四个顶点,如图 2-7(b)所示。由于不存在一条直线能将两个顶点(0,1)、(1,0)和另两个顶点(0,0)、(1,1)分开,故单层感知器无法实现异或功能。

XOR 真值表

输入样本		输出
0	0	0
0	1	1
1	0	1
1	1	0

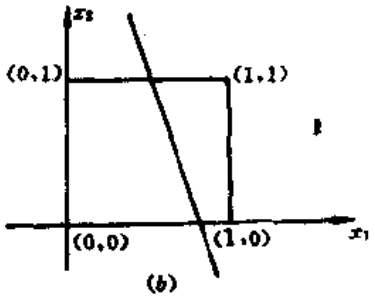


图 2-7 异或问题

(a)异或问题表列　　(b)异或问题的二维图形

2.2.1.2 单层感知器的逻辑功能

单层感知器的结构如图 2-6(a)所示。利用单层感知器可以实现逻辑与、或、非的运算。

图 2-6(a)中,如果选取不同的权向量 $W=(w_1,w_2)^T$ 和阈值 θ,就可以构成简单的逻辑单元。由前面的讨论可知,图 2-6(a)中神经元的输出

$$y=f(x_1w_1+w_2x_2-\theta) \text{ 且 } \quad y(u)=\begin{cases}1, & u\geqslant 0\\ 0, & u<0\end{cases}$$

(1)"与"运算　$y=x_1\cdot x_2$。

此时可取 $w_1=w_2=1,\theta=2$　从而

$$y=f(x_1+x_2-2)=\begin{cases}1, & x_1+x_2-2\geqslant 0\\ 0, & x_1+x_2-2<0\end{cases}$$

(2)"或"运算　$y=x_1+x_2$。

此时取 $w_1=w_2=1,\theta=1$。

(3)"非"运算　$y=\bar{x}_1$。

此时可取 $w_2=0,w_1=-1,\theta=0$。

以上逻辑运算关系由图 2-8 所示。

2.2.2 多层感知器网络

单层感知器只能满足线性分类,如果有两类样本 A、B,它们不能用一个超平面将它分开。如图 2-9 所示的二维平面中,A 类和 B 类是分布在此平面中的一些点或区域,不能用一根直线把 A、B 类分开,这样的样本集称为不能线性划分的样本集。

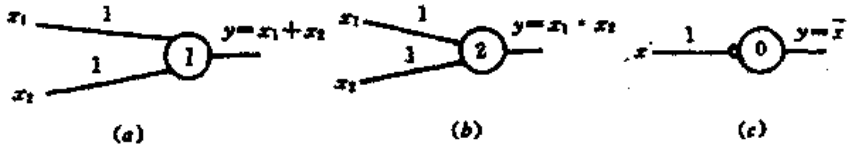


图 2-8 简单逻辑关系图

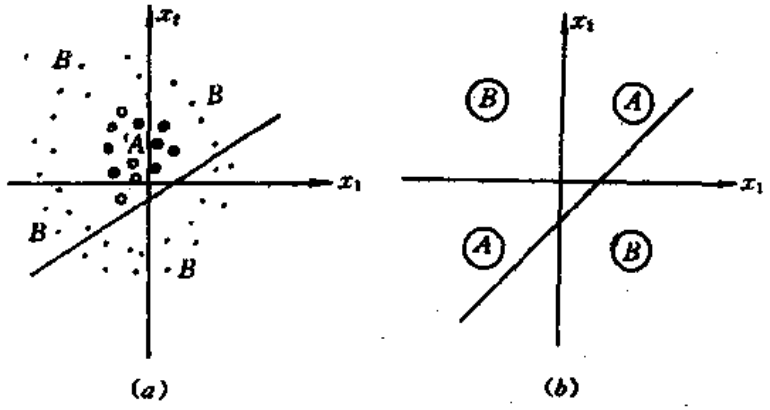


图 2-9 不能线性划分的样本集

2.2.2.1 二层感知器网络

二层感知器网络可以用来解决上述问题。二层感知器网络如图 2-10 所示。输入层与输出层之间存在一个隐含层。输入量共有 n 个 $x_1, x_2, \cdots, x_n$。中间层(隐层)有 n_1 个神经元，其对应的神经元输出为 $h_1, h_2, \cdots, h_{n_1}$。联接权 w_{ij} 为第 i 个输入节点与隐层第 j 个神经元之间的耦合系数，$i=1,2,\cdots,n$，$j=1,2,\cdots,n_1$。输出层神经元个数为 1，输出为 y。隐层到输出层之间的连接权用 v_k 表示，$k=1,2,\cdots,n_1$。

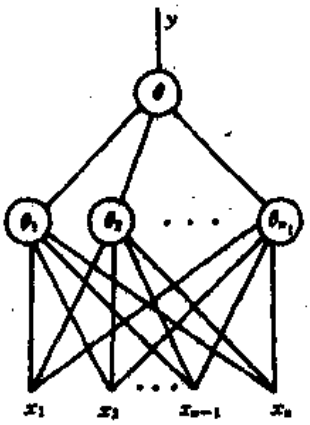


图 2-10 二层感知器网络

由二层感知器网络结构图可知，输出 y 的表达式为

$$y = f\left(\sum_{j=1}^{n} v_j h_j - \theta\right) \tag{2-12}$$

隐层神经元输出

$$h_k = f\left(\sum_{i=1}^{n} w_{ik} x_i - \theta_k\right) \tag{2-13}$$

其中 w_{ik} 为第 i 个输入与第 k 个隐单元之间的权系数，θ_k 为第 k 个隐单元的阈值。此时隐层与 n 个输入单元的关系如同单层感知器一样，形成一些 $n-1$ 维超平面。为了简明地叙述二层感知器的分类作用，不妨设输入维数 $n=2$，隐层神经元数 $n_1=3$。

这时，隐层神经元的输出为

$$\begin{cases} h_1 = f(\sum_{i=1}^{2} w_{i1}x_i - \theta_1) \\ h_2 = f(\sum_{i=1}^{2} w_{i2}x_i - \theta_2) \\ h_3 = f(\sum_{i=1}^{2} w_{i3}x_i - \theta_3) \end{cases} \tag{2-14}$$

其中 h_1, h_2, h_3 为隐层单元的输出。根据单层感知器的分析可知，式(2-14)的几何图形是二维输入空间的三根直线，因为 w_{ij} 和 θ_i 的不同，此三直线的斜率和截距均不同。于是利用这三个隐单元所得到的一个封闭区域，就有可能实现样本集的正确划分。

然而，从输入层到隐层仅仅得到三条相交的直线，要完成二维输入空间上的区域分割，必须使输出 y 满足下式即可得到正确划分：

$$y = \begin{cases} (x_1, x_2)/[(w_{11}x_1 + w_{21}x_2 - \theta_1) > 0 \quad \wedge (w_{12}x_1 + w_{22}x_2 - \theta_2) \\ > 0 \quad \wedge (w_{13}x_1 + w_{23}x_2 - \theta_3) > 0] \end{cases}$$

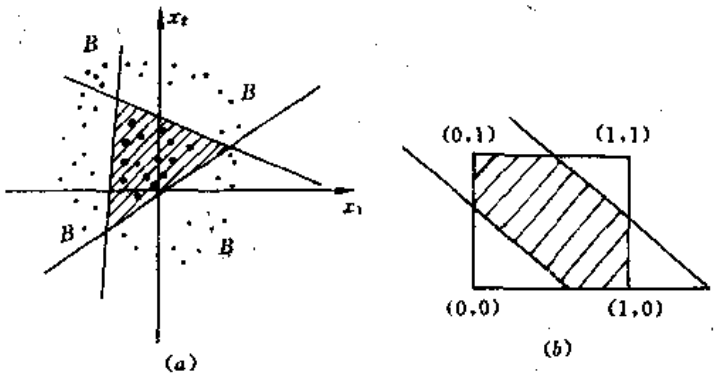


图 2-11　二层感知器在输入空间的划分

从隐单元到输出层正是利用单层感知器的逻辑“与”功能，使隐层形成的三根直线在二维平面上分割的区域相“与”，形成平面上的封闭区域，实现样本集的正确划分。二层感知器网络在输入空间的划分如图 2-11 所示。

由图 2-11 可知，异或问题可以用二层感知器来实现。(其输入节点个数 $n=2$，隐层单元个数 $n_1=2$，输出个数为 1)。简言之，二层感知器的隐层为线性化分作用，输出层为“与”组合。

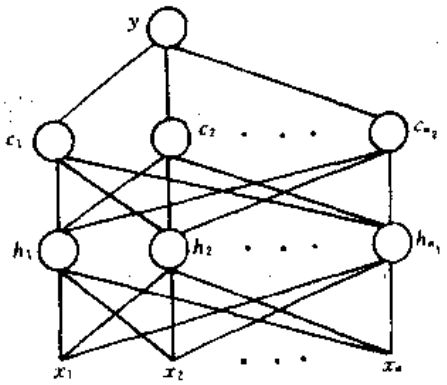


图 2-12　多层感知器网络

2.2.2.2　多层感知器网络

多层感知器网络如图 2-12 所示。输入层与输出层之间存在一些隐层，输入节点为 n 个，即 $x_1, x_2, \cdots, x_n$，第一隐层有 n_1 个神经元，其对应的输出为 $h_1, h_2, \cdots, h_{n_1}$，第二隐层有 n_2 个神经元，其对应的神经元输出为 $c_1, c_2, \cdots, c_{n_2}$，输出为 y。它们以前馈方式联接，其输入输出关系式为

$$\begin{cases} h_j = f(\sum_{i=1}^{n} w_{ij}x_i - \theta_j), j = 1,1,\cdots,n_2 \\ c_k = f(\sum_{j=1}^{n_1} v_{jk}h_j - \theta_k), k = 1,2,\cdots,n_1 \\ y = f(\sum_{k=1}^{n_2} w_k c_k - \theta) \end{cases} \tag{2-15}$$

在式(2-15)中，h_j 为第一隐层第 j 个单元的输出，w_{ij}为输入层第 i 个节点与第一隐层第 j个单元之间的连接权，θ_j 为第一隐层第 j 个单元的阈值；同理，c_k 为第二隐层第 k 个单元的输出，θ_k 为其对应的阈值，v_{jk}是第一隐层第 j 个单元与第二隐层第 k 单元间的连接权，y 为网络输出，w_k 为第二隐层第 k 个单元与输出之间的连接权。式中变换函数 $f(\cdot)$由式(2-4)确定。

感知器网络的信息是逐层前向传播的，下层的各个单元均与上一层的每个单元互连。输入单元依(2-15)式逐层进行操作，层间的连接权可以通过学习规则进行调整。

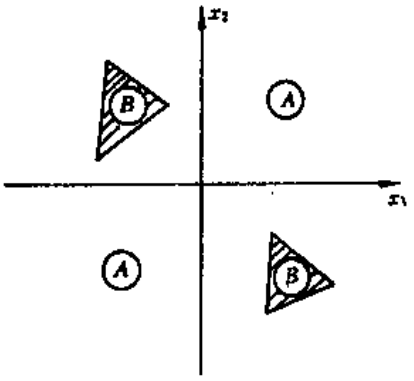


图 2-13 多层感知器在二维平面上划分

可以看出，多层感知器可通过单层感知器进行适当的组合达到任何形状的划分。对于图2-13，可以采用四层感知器网络来完成：第一层为输入层，第二层(第一隐层)为线性划分，第三层(第二隐层)为“与”组合，第四层(输出层)为“或”组合。

2.2.3 多层感知器的设计

如何来设计多层感知器隐单元的个数，这里给出隐单元数的上下限，以供设计时参考。

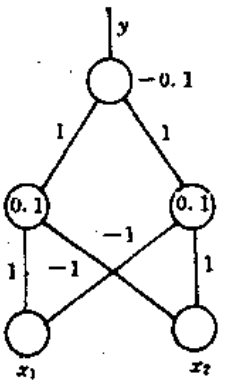


图 2-14 异或问题感知器网络

在式(2-15)中，令隐单元输出 $h_j=\mathrm{sgn}(\sum_{i=1}^{n} w_{ij}x_i-\theta_j)$是一个线性阈值分割函数，每个隐单元把 n 维输入空间 s 划为 s_A 和 s_B 两个部分。设 g 是定义在 s 空间的一个函数，对于感知器网络对应的 k 个样本输入，希望其满足：

$$y^l = \mathrm{sgn}(\sum_{j=1}^{n_1} h_j w_j^* - \theta^*) = g(x^l), \quad l = 1,2,\cdots,k$$

问题是能否找到隐单元数 n_1，通过调整输出单元的系数 w_j^*，θ^* 使上式满足条件，即可实现任何一个 g 函数的映射。

在一个输入空间 s 上，存在有 k 个样本，如果需要用 n_1 个超平面划分为 d 个区域，使 $d>k$，并保证每个区域对应于一个样本，则最大的隐单元数 n_1 应满足：

$$n_1 = k - 1 \tag{2-16}$$

其中 k 为输入样本数。

隐单元数为 $k-1$，是设计网络的上限，因为 $k-1$ 超平面是不相交的，因此只可分为 k 个区域，但考虑到超平面可交且只有一个隐层的情况下，输入为 n 个单元，n_1 个隐单元可把输入

空间划分成一定数目的区域，如果这个区域是封闭的，称之为闭区域，反之为开区域。在数学上可得到独立区域数为

$$p(n_1,n)=\sum_{i=0}^{n}\binom{n_1}{i},\qquad n_1\geqslant i \tag{2-17}$$

其中

$$\binom{n_1}{i}=0,\qquad n_1<i$$

那么，对 k 个样本进行线性分割则要求：

$$n_1=\min[p(n_1,n)]\geqslant k \tag{2-18}$$

式(2-18)给出隐单元数的下限，即使得 $p(n_1,n)$ 为最小($p(n_1,n)$应大于样本数)时的 n_1 值。式(2-16)和(2-18)可作为选择三层感知器隐单元个数的参考，在实际问题中有可能减小。

2.2.4 多层感知器实例

用多层感知器求解异或问题，其样本表列如图 2-7*a* 所示。异或问题的输入是一个二维向量，输出的类别为 2，样本数为 4，异或问题的网络拓扑结构由以下确定：

输入节点(单元)$n=2$，输出节点 $m=1$。隐层节点数由式(2-16)和(2-18)确定。根据公式(2-16)，$n_1=k-1=3$，根据式(2-18)$n_1=2$，故 $2\leqslant n_1\leqslant 3$。我们取 $n_1=2$。感知器的结构定了以后，可用学习的方法对网络进行训练。

感知器的训练采用由一组样本组成的集合来进行。在训练期间，将这些样本重复送到感知器的输入层，通过调整权值使感知器的输出达到所要求的理想输出。训练集中的每个样本是一个由输入向量和输出向量组成的向量组对。对于异或问题，训练集中的 4 个样本分别是(0 0 1)、(1 0 0)、(0 1 0)和(1 1 1)。在训练中，重复地把样本的前两个元素(输入向量)送到感知器的输入节点，后一个元素(理想输出)作为导师与实际网络的输出 y 进行比较，不断修正网络的权系数和阈值，最终使感知器输出达到理想输出。异或问题的感知器网络结构如图 2-14 所示。

多层感知器可对输入空间矢量进行分类，也可对一些离散函数进行映射，它的输入、输出间的映射是比较确定的。它的抗干扰性和鲁棒性比较差，在样本的畸变、噪声加入的情况下，不能得到正确分类；另一方面，为了实现非线性划分，需要增加感知器的层数，这给计算增加了难度，使得感知器的应用范围受到了限制。

2.3 Hopfield 网络模型

J·Hopfield 在 1982 年发表的论文宣告了神经网络第二次浪潮的到来。他表明了 Hopfield 模型可用作联想存储器，后来他将这一网络应用于解决最优化问题，取得了良好的效果。

Hopfield 网络分为两类：离散型 Hopfield 网络(Discrete Hopfield Neural Network)和连续型 Hopfield 网络(Continuous Hopfield Neural Network)。

2.3.1 离散型Hopfield网络

2.3.1.1 离散型 Hopfield 网络

离散型Hopfield 网络只有一个神经元层次，这种单层网络的每个神经元的输出都与其它

神经元的输入相连，称为单层全反馈网络。其结构如图 2-15 所示。

离散 Hopfield 网络每个单元均有一个状态值，它取两个可能的值之一。这里，用值 0 和 1 表示。整个网络的状态由单个神经元的状态组成。网络的状态可用一个 0/1 组成的向量来表示。向量的某个元素对应于网络中某个神经元的状态。这样，在任意给定的时刻，网络的状态可以表示为

$$U=(u_1,u_2,\cdots,u_n)$$

这里，$u_i,i=1,2,\cdots,n$ 用 0 或 1 表示。Hopfield 网络中的各个神经元之间是全互连的。即各个神经元之间均相互连接，神经元之间的连接是双向的。这种连接方式使得网络中的每个神经元的输出均反馈到同一层次的其它神经元的输入上(包括自反馈，当 $T_{ii}\neq 0,i=1,2,\cdots,n$)。这样的网络在没有外部输入的情况下也能进入稳定状态。我们沿用 Hopfield 的表记形式，用 T_{ji} 表示神经元 j 与神经元 i 之间的连接权值。权值 T_{ji} 和 T_{ij} 具有相同的值，即 $T_{ji}=T_{ij}$。当 $T_{ji}=T_{ij}$ 的条件满足时，网络肯定能够收敛到某一个稳定值。但许多网络 $T_{ji}\neq T_{ij}$ 时，也能收敛。在某一时刻 t，每个神经元依下面两式计算状态值：

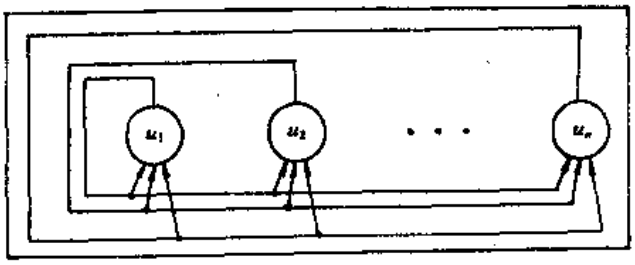


图 2-15 Hopfield 网络

$$u_i(t+1)=\mathrm{sgn}(H_i(t))=\begin{cases}1, & H_i(t)\geqslant 0\\ 0, & H_i(t)<0\end{cases} \tag{2-19}$$

对所有的 $1\leqslant i\leqslant n$ 有

$$H_i(t)=\sum_{j=1}^{n}T_{ij}u_j+I_i \tag{2-20}$$

式中 I_i 为第 i 个神经元的外输入信号。如果神经网络从 0 时刻的状态 $U(0)$ 出发，按式(2-19)、(2-20)演化，直到某个 t 时刻到达 $U(t_1)$，并且 $t>t_1$ 之后，整体状态始终不变，则称网络处于稳定状态。

离散 Hopfield 网络有两种工作方式：

(1)异步工作方式：在某一时刻 t，只有一个神经元按照式(2-19)、(2-20)进行演化，而其余的神经元的输出保持不变。这一变化的神经元可以随机选取也可以按照预定的顺序来选择。如选到的神经元为第 j 个，则有

$$u_j(t+1)=\mathrm{sgn}(\sum_{i=1}^{n}T_{ji}u_i(t)+I_j)$$

$$u_i(t+1)=u_i(t)\quad i\neq j$$

(2)同步工作方式：在任何 t 时刻，所有的神经元同时按照式(2-19)、(2-20)演化。

2.3.1.2 离散型 Hopfield 网络的稳定分析

离散型 Hopfield 网络是一种单层的，其输入、输出为二值的反馈式网络，它主要用于联想

记忆。当输入向量 $I=(I_1,I_2,\cdots,I_n)^T$ 作为一个初值时，网络通过反馈演化，从网络输出端得到一个向量 $U=(u_1,u_2,\cdots,u_n)^T$，U 是从初值 I 演化而联想到的一个稳定记忆。对于反馈网络来说，稳定性是一个很重要的问题。

网络从一个初始态 $U(t_0)$ 开始，经过一个有限的时间 t，网络的输出不再发生变化，这些输出为网络的稳定点。当网络处于稳定点时，每个神经元的输出满足：

$$u_j(t+1)=u_j(t)=\mathrm{sgn}\Big[\sum_{i=1}^{n}T_{ji}u_i(t)+I_j\Big] \tag{2-21}$$

对于不同的输入样本向量，如果希望每一个输入的样本为系统的初值，最后又都能演化到自己，即每一个样本向量都是网络最终的稳定点。

(1)网络的稳定性

网络存在多个稳定点，其中有些即使是渐近稳定点，只要存在一个极限环，此网络就不可避免地在某些初值时出现振荡。网络可以看作一个多输入、多输出带阈值(如果 I_j 永远接在输入端，则 I_j 可看成一个阈值 $I_j=-\theta_j$)的二态非线性动力系统。在满足一定的参数条件下，某种能量函数在网络运行过程中不断降低，最后趋于稳定的平衡状态。

可以利用这种能量函数：

$$E=-\frac{1}{2}\sum_i\sum_j T_{ij}u_iu_j-\sum_i I_iu_i \tag{2-22}$$

在演算过程中，能量的变化量

$$\Delta E=\Delta u_j\Big(-\sum_j T_{ij}u_j-I_i\Big)$$

从任一初始状态出发，只要每次迭代能满足 $\Delta E\leqslant 0$，那么网络的能量就会越来越小，最后趋向于稳定点 $\Delta E=0$。以下分两种工作方式讨论它的稳定性。

异步方式工作状况下，满足 $T_{ij}=T_{ji}$，$T_{ii}\geqslant 0$　$i,j=1,2,\cdots,n$，则能量函数单调下降，网络稳定；同步方式工作状况下，如果满足 $T_{ij}=T_{ji}$，网络将收敛于一个稳定点，或者网络收敛于一个周期为 2 的极限环。(证明从略)

2.3.1.3　离散 Hopfield 网络拓扑设计

用输入样本矢量的外积来设计离散 Hopfield 网络的权，这种方法被称为外积型设计。外积型设计主要用于联想记忆，步骤如下：

(1)根据需要记忆的样本 $U^1,U^2,\cdots,U^m$，用外积型设计权

$$T_{ij}=\begin{cases}\sum_{k=1}^{m}u_i^ku_j^k, & i\neq j\\ 0, & i=j\end{cases} \tag{2-23}$$

$$I_j=0$$

式(2-23)中 u_i^k 为第 k 个样本向量的第 j 个分量。

(2)令输入样本作为网络输出的初值。

(3)用下面迭代公式进行演算：

$$u_i(t+1)=\mathrm{sgn}\Big[\sum_{j=1}^{n}T_{ij}u_j(t)\Big]$$

(4)重复迭代直至每个输出单元不变为止，即

$$u_i(t+1)=u_i(t),\quad i=1,2,\cdots,n$$

用式(2-23)设计的网络满足$T_{ii}=0$和$T_{ij}=T_{ji}$，且外界输入$I=0$。如果要求存贮的样本向量是两两正交的，即样本$U^1,U^2,\cdots,U^m$中任意两个不同样本向量的内积等于零。

$$(U^i)^T(U^j)=0,\qquad 1\leqslant i,j\leqslant m,i\neq j。$$

在这种情况下，任取一个输入样本U^i作为初始输入进行迭代计算。只要满足$n>m$，则U^i为网络的一个稳定点。n为网络神经元个数，m为样本向量个数；对于非正交的样本，如果满足$n>\sqrt{(m-1)n}$，则网络仍可收敛到其存贮样本上。

2.3.1.4 应用举例

离散Hopfield网络的一个应用是用作联想存储器。图2-16给出了其用作联想存储器时的存贮内容及联想情况，每一个存储项(图2-16(*a*))用1或0组成的向量来表示。在这个例子中，存储项由两部分组成，前一部分为事物的名称，后一部分为其颜色，为清晰起见，图中用"|"将它们分开。每个向量对应于网络处于能量极小(稳定)时的状态。图2-16(*b*)给出了网络输入某一存储项的一部分，而联想出另一部分的情况，即首先给网络一个初始输入，名称为树，而颜色未知(为空)，网络通过计算收敛后，其对应的状态就是要查找的存储项，即名部分为"树"，颜色部分为"青色"，也就是说网络通过计算由"树"而联想到了"青色"。图2-16(*c*)给出了带有噪音的初始输入送入网络时的查找结果，此时名称部分为带有噪音的"天空"，颜色部分也是带有噪音的蓝色，网络经过计算收敛后，其状态表示了完全正确的"天空"与"蓝色"编码。这一结果表明Hopfield网络具有抗噪音的能力。

树	0 0 1 1 1 1 0 0 0 0 1 0 | 0 0 1 1 0 1 1 0 0 1 0 0	青色
西红柿	1 0 0 0 1 0 0 1 1 0 1 1 | 0 1 0 0 0 1 1 0 1 0 1 1	红色
天 空	0 1 1 0 0 1 1 1 0 0 0 1 | 1 0 1 1 1 1 0 0 0 0 0 1	蓝色

(*a*)

输 入:树	0 0 1 1 1 1 0 0 0 0 1 0 | 0 0 0 0 0 0 0 0 0 0 0 0	空
联想结果:	0 0 1 1 1 1 0 0 0 0 1 0 | 0 0 1 1 0 1 1 0 0 1 0 0	青色

(*b*)

噪音输入:天空	0 1 1 1 0 1 1 0 0 0 0 1 | 1 0 0 0 0 1 0 0 0 0 0 1	蓝色
去噪结果:天空	0 1 1 0 0 1 1 1 0 0 0 1 | 1 0 1 1 1 1 0 0 0 0 0 1	蓝色

(*c*)

图2-16 Hopfield网络用作联想存贮器

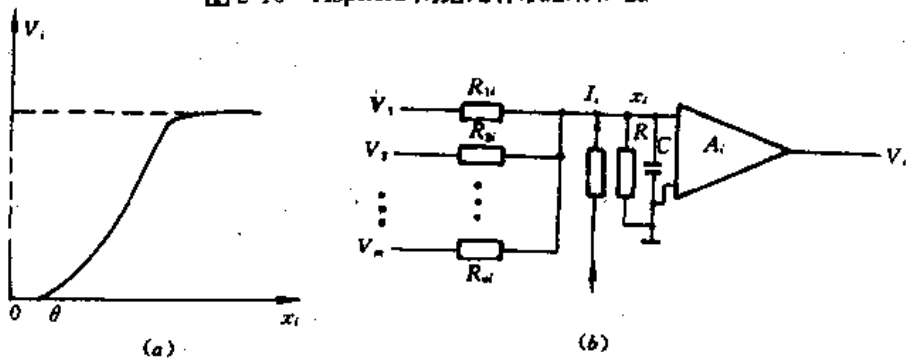


图2-17 Hopfield网神经单元电路模型

(*a*)v_i与u_i关系曲线；(*b*)运算放大器神经元模拟模型

2.3.2 连续型 Hopfield 网络

连续型 Hopfield 网络具有与离散型 Hopfield 网络相同的结构，这种拓扑结构反映了生物神经系统中广泛存在的神经回路现象。

在连续 Hopfield 网络中，各个神经元的状态取值是连续的，它根据下面所规定的规律变化：

$$\begin{cases} c_j \dfrac{\mathrm{d}u_j}{\mathrm{d}t} = \sum\limits_j T_{ji} v_j - \dfrac{u_j}{R_j} + I_j \\ v_j = f(u_j), \qquad j = 1,2,\cdots,n \end{cases} \tag{2-24}$$

其中 u_j 为神经元 j 的状态，c_j 表征细胞膜输入电容，为一常数且大于 0；R_j 表示细胞膜电阻也为正数，I_j 为外部输入，v_i 为神经元 i 的输出，输出函数 $f(\cdot)$ 为 s 型函数，即

$$f(x) = 1/(1 + \mathrm{e}^{-x}) \tag{2-25}$$

s 型函数对应的曲线如图 2-17(a)所示。

从式(2-24)的形式看，如果把神经细胞信息的传递看作是电压信号的传递，可以用一个简单的放大器来仿真形成人工神经元，图 2-17(b)中，第 j 个运算放大器的输入为状态 u_j，它与输出 v_j 之间的关系满足式(2-25)，其电源电压为输出 v_j 的最大值。

根据克希霍夫定律可得

$$\begin{cases} c \dfrac{\mathrm{d}u_i}{\mathrm{d}t} = -\left(\dfrac{1}{R} + \sum\limits_{j=1}^{n} \dfrac{1}{R_{ij}}\right) u_i + \sum\limits_{j=1}^{n} \dfrac{1}{R_{ij}} v_j + I_i \\ v_j = f(u_j) \end{cases} \tag{2-26}$$

对于具有 n 个互相连接的人工神经元网络，每个神经元都满足式(2-26)。故可以用一个 n 维非线性微分方程来描述连续 Hopfield 网络。

2.3.2.1 连续 Hopfield 网络的稳定性

Hopfield 在 80 年代初提出了一个对单层反馈动态神经网络的稳定性判别函数，这个函数有确定的物理意义。

对于神经元 j 的状态方程如式(2-22)所示。网络的能量函数定义为

$$E = -\frac{1}{2} \sum_i \sum_j T_{ij} v_i v_j - \sum_i v_i I_i + \sum_i \frac{1}{R_i} \int_0^{v_i} f^{-1}(\eta)\mathrm{d}\eta \tag{2-27}$$

其中 v_i，I_i 分别表示第 i 个神经元的输出和外部输入，$f^{-1}(v_i)$ 表示为函数 v_i 的逆函数；即 $f^{-1}(v_i) = u_i$，u_i 为第 i 个神经元的状态变量。如果能量函数 E 随时间单调下降，就总可以达到能量的极小点，也是系统的稳定点。

稳定条件：若 $T_{ij} = T_{ji}$ 且输入输出关系是单调上升函数($\dfrac{\mathrm{d}v_i}{\mathrm{d}u_i} > 0$)，则网络最终到达稳定点。

证明：

对式(2-25)求导

$$\frac{\mathrm{d}E}{\mathrm{d}v_j} = -\left(\sum_j T_{ij} v_i + I_i - \frac{u_i}{R_i}\right) = -c_i \frac{\mathrm{d}u_i}{\mathrm{d}t}$$

从而有

$$\frac{dE}{dt}=\sum_j \frac{dE}{dv_i}\frac{dv_i}{dt}=-\sum_i c_i \frac{du_i}{dt}\cdot\frac{dv_i}{dt}$$

$$=-\sum_i c_i \frac{dv_i}{du_i}\left(\frac{du_i}{dt}\right)^2 \tag{2-28}$$

由于输入输出是单调上升关系，有$\frac{dv_i}{du_i}>0$，故$\frac{dE}{dt}\leqslant 0$。

当$\frac{dE}{dt}=0$时，则有$\frac{du_i}{dt}=0$，$i=1,2,\cdots,n$表示能量极小点与$\frac{du_i}{dt}=0$的平衡点一致。

2.3.2.2 Hopfield 网络应用于 TSP 问题

TSP 问题(Travelling Salesman Problem)是一种十分难的优化问题：一个推销员去 n 个城市推销他的产品，从他所在的城市出发访问 $n-1$ 个城市，最后又回到原来的城市，要求每个城市只经过一次，并且所有的城市都要走到，要求经过的路程最短，如果已知城市 A、B、C、…间的距离为 d_{AB}，d_{BC}，…，那么总的距离 $d=d_{AB}+d_{BC}+\cdots$，对于这种动态规划去求得 $\min(d)$的解，在城市数少的情况下尚可以计算，当城市数目增加，计算量就会增加到无法进行的地步。对于这个问题如果采用 Hopfield 能量函数来设计，在城市数<100 时，采用电路实现时，它的计算速度非常快。

表 2-1 关联矩阵

城市＼顺序	1	2	3	4	5
A	0	1	0	0	0
B	0	0	0	1	0
C	1	0	0	0	0
D	0	0	0	0	1
E	0	0	1	0	0

n 个城市用 n^2 个神经元表示，如以 5 个城市 A、B、C、D、E 为例，可以用一个关联矩阵表示 TSP 问题，其中行表示城市，列表示推销员到达某城市的先后顺序，如表 2-1 所示。为了保证每个城市只去一次(不包括初始出发城市)，则在关联矩阵上每一行只能有一个为 1，其它均为零，同时，每一列只能有一个元为 1，其它的为零，关联矩阵中元素“1”的和应为“n”。从表 2-1 所示的关联矩阵中反映推销员的访问路径为 $C\to A\to E\to B\to D\to C$，而推销员所走的距离为

$$d=d_{CA}+d_{AE}+d_{EB}+d_{BD}+d_{DC}$$

以下讨论如何把这个问题转化为一个能量函数，再由能量函数转化为电路，问题就解决了。

(1)TSP 问题的目标函数 $f(v)$。

把关联矩阵的每个元用符号 v_{xi}，v_{yj}…来表示，下标 x，y 表示城市 A、B、C，…，下标 i，j 表示顺序，目标函数为

$$f(v)=\frac{p}{2}\sum_x\sum_{\substack{j\\ x\neq y}}\sum_y d_{xy}v_{xi}(v_{y,i+1}+v_{y,i-1})$$

v_{xi}取 0 或 1 二值。$v_{xi}=0$ 表示不经过城市 x，$v_{xi}=1$ 表示经过城市 x。d_{xy}表示两个不同城市之间的距离。如果 $v_{xi}=1$，那么寻找与 i 相邻次数的城市。如在关联矩阵中 $v_{A2}=1$，那么就可以找到 v_{c1}和 v_{E3}两个与 v_{A2}相邻列上的非零值，此时在 $f(v)$中得到 d_{AE}和 d_{CA}两个相加的量。依次类推，那些推销员走过的距离累加起来，在 $f(v)$中每个距离计算了两次，$\frac{1}{2}$是为了除去重复计算而

设的。

(2)约束条件 $g(v)$。

由以上讨论可知,约束条件是要保证关联矩阵每一行和每一列只有一个值为 1,其余均为零,为 1 的总数为 n。约束条件

$$g(v)=\frac{Q}{2}\sum_{i}\sum_{\substack{x\\x\neq y}}\sum_{y}v_{xi}v_{yi}+\frac{S}{2}\sum_{x}\sum_{\substack{i\\i\neq j}}\sum_{j}v_{xi}v_{xj}+\frac{T}{2}\left(\sum_{x}\sum_{i}v_{xi}-n\right)^2$$

第一项表示同一次只能到达一个城市,满足约束时,该项为零;第二项表示每个城市只能去一次,第三项表示 v_{xi} 为 1 的次数最多为 n,若满足约束,这些项为零。

(3)Hopfield 能量函数。

$$\begin{aligned}E&=f(v)+g(v)\\&=\frac{P}{2}\sum_{x}\sum_{\substack{i\\x\neq y}}\sum_{y}d_{xy}v_{xi}(v_{y,i+1}+v_{y,i-1})+\frac{Q}{2}\sum_{i}\sum_{\substack{x\\x\neq y}}\sum_{y}v_{xi}v_{yi}\\&\quad+\frac{S}{2}\sum_{x}\sum_{\substack{i\\i\neq j}}\sum_{j}v_{xi}v_{xj}+\frac{T}{2}\left(\sum_{x}\sum_{i}v_{xi}-n\right)^2\end{aligned}$$

系数 P、Q、S、T 和距离 d_{xy} 都是大于零的,因而 $E>0$,然后根据能量函数 E 求得神经元的状态方程,满足

$$\frac{\partial E}{\partial u_i}=-\frac{\mathrm{d}u_i}{\mathrm{d}t},\qquad i=1,2,\cdots,n。$$

可以设计出由电路组成的 Hopfield 网络的权和输入 I,系统将最后收敛到一些稳定点,那就是较优路径的解。

2.3.2.3 Hopfield 网络能量函数与优化计算

对于 Hopfield 网络,能量函数是一个反映多维神经元状态的标量函数。用简单的电路形成人工神经元,网络随时间变化自然收敛到稳定点上,在这些稳定点上的能量函数最小。因此,如果人为地设计神经网的联接权矩阵 W 和输入 I,把优化问题中的目标函数、约束条件与 Hopfield 能量函数联系起来,那么电路的平衡点就是能量函数的极小点,也是优化中满足约束条件下的目标函数的极小点。这就是应用人工神经网络求解优化问题。

下面给出能量函数的一般设计方法。

设优化的目标函数 $f(x)$,$x\in R^n$ 为人工神经元网络的状态,$g(x)=0$ 为约束条件。优化问题归结为满足约束条件下使目标函数最小。

能量函数

$$E=f(x)+c|g(x)|>0$$

这里 $c|g(x)|$ 也称惩罚函数,因为在约束条件不能满足时,$c|g(x)|$ 的值将很大,造成 E 很大,迫使 x 满足约束条件。

若 E 的全导数要小于零,则要求满足

$$\frac{\partial E}{\partial x_i}=-\frac{\mathrm{d}x_i}{\mathrm{d}t},\qquad i=1,2,\cdots,n \tag{2-29}$$

因为

$$\frac{\mathrm{d}E}{\mathrm{d}t}=\sum_{i}\frac{\partial E}{\partial x_i}\frac{\mathrm{d}x_i}{\mathrm{d}t}=\sum_{i}-\left(\frac{\mathrm{d}x_i}{\mathrm{d}t}\right)^2\leqslant 0$$

按照 Hopfield 能量函数的要求，只要 E 在负的方向有界；$E>E_{\min}$，同时$\frac{dE}{dt}\leqslant 0$，则系统最后总能达到 E 最小且$\frac{dE}{dt}=0$ 的点，同时又是系统的稳定点$\frac{dx}{dt}=0$。

如果能量函数不是直接与状态变量 x 有关，而是与输出变量 v 有关，只要$\frac{\partial v_i}{\partial x_i}>0$，其结果都是相同的。

具体设计步骤如下：

(1)根据要求的目标函数，写出能量函数第一项 $f(x)$；

(2)根据约束条件 $g(x)$，写出惩罚函数，使其满足约束条件时惩罚函数最小，作为能量函数的第二项；

(3)加上一项 $\sum_i \frac{1}{R_i}\int_0^{v_i} F^{-1}(\eta)d\eta$。这一项是人为的，因为在神经元状态方程中，存在一项 $-\frac{x_i}{R}$，它是人工神经网络电路设计中产生的，为了使设计优化的结果能在电路中得以实现，在能量函数中加上了这一项。这是一个正值函数，表现出在边缘上能量大，并且 v_i 值越小，能量值也越小，当运算放大器放大倍数是够大时，又可忽略，因而它对能量 E 和优化问题的结果影响不大($F^{-1}(v_i)=x_i$)。

(4)根据能量函数 E 求得状态方程，满足(2-29)式

$$\frac{\partial E}{\partial x_i}=-\frac{dx_i}{dt},\qquad i=1,2,\cdots,n$$

(5)根据状态方程，对照下列公式求出 $w_{ij},I_i,i,j=1,2,\cdots,n$

$$c_i\frac{dx_i}{dt}=\frac{-x_i}{R_i}+\sum_j w_{ij}v_j+I_i \tag{2-30}$$

$$v_i=F(x_i),\qquad i,j=1,2,\cdots,n$$

式(2-30)与式(2-24)为等价方程，均为神经元的状态方程。这里 x_i 是第 i 个神经元的状态。

(6)用类似于图 2-18(b)的电路实现：$\frac{1}{R_{ij}}=w_{ij}$ 对所有的 i,j。

Hopfield 的能量函数可以用来解优化问题，它的主要优点是用一个人工神经网络的电路就可以完成优化工作，因此计算速度快，而且避免了复杂性问题。但是 Hopfield 网络电路本身还存在以下两个问题：

(1)联接权的数目随神经元数增加，有 N 个神经元组成的网络，则其联接权数目为 N^2 个，这对电路的实现会产生较大的困难。

(2)网络是非线性的，因而随着 N 的增加，其局部极小点的数目也随之增加，使优化的结果不能达到全局最优。

2.4 自组织竞争网络模型

在实际的神经网络中，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它神经细胞产生抑制。这种抑制使神经细胞之间出现竞争，一个兴奋最强的神经细胞对周围神经细胞的抑制也强，虽然一开始各个神经细胞都处于兴奋状态，但最后是那个输出最大的神经细胞“赢”了，而其周围的神经细胞输了。

另外，在认知过程中除了从教师那儿得到知识外，还有一种不需要教师指导的学习。例如：婴儿出生后，听到外界声音的刺激，他自然会发出声，并自己在外界环境中学习抓东西，走路等。这种直接依靠外界刺激，“无师自通”达到的功能有时也称为自学习、自组织的学习方法。

2.4.1 自适应共振理论模型

2.4.1.1 自适应共振理论模型结构

自适应共振理论 ART(Adaptive Resonance Theory)模型是一种比较接近于实际神经系统的一种模型，它的记忆容量可以随学习样本的增加而增加，记忆形式也与生物中的记忆形式类似，它的结构如图 2-18 所示。

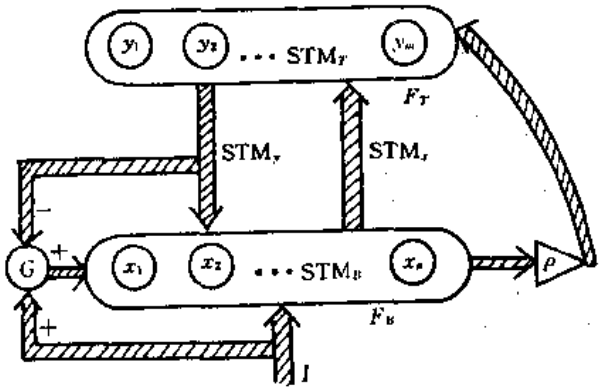


图 2-18 ART 模型结构

ART 一般由两层神经元和一些控制部分结合而成的。在图 2-18 的结构图中，底层为 F_B 层，由 n 个神经元组成，它接受三种信号：外界输入信号 I，上一层来的信号和增益控制器的控制信号。上层为输出层 F_T，输出层 F_T 接受两种信号：来自底层 F_B 的信号和控制器 ρ 的控制信号。F_T 层神经元个数为 m，m 的数目决定了 ART 网络的容量。

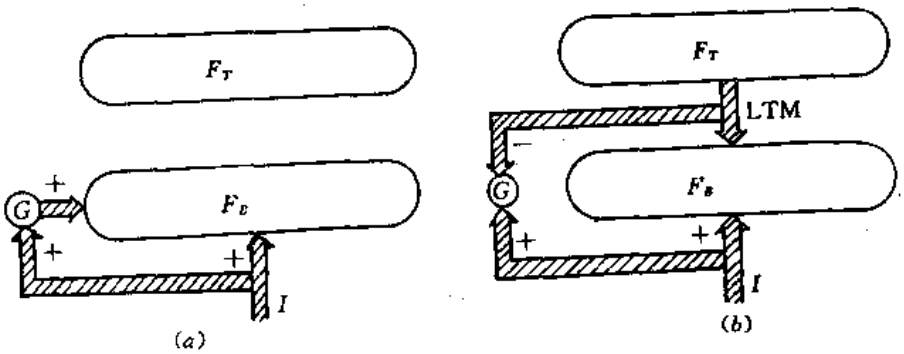


图 2-19 两种不同情况的 2/3 规则

(1)ART 模型网络的记忆方式。

ART 网络存在两种记忆方式，一种是短期记忆(Short Time Memory)，用缩写符号 STM

表示。另一种是长期记忆(Long Time Memory)，用LTM表示。ART网络的F_B和F_T层中神经元的状态在网络运行过程中是变化的，属于短期记忆，底层状态x_i，$i=1,2,\cdots,n$，由输入到F_B的三种信号所决定，当输入样本I改变时，x_i也会改变。当且仅当在某一固定的输入时刻，x_i被暂时记忆在F_B中，用STM_B来表示。上层神经元状态y_j，$j=1,2,\cdots,m$也是短期记忆，用STM_T表示。底层到上层之间的两组权为长期记忆。一组由下向上的权用LTM_x表示；另一组是由上向下的权用LTM_y表示。一旦网络经过学习后，对样本的记忆留在两组权中，即使输入样本改变了，这两组权依然存在。而且当输入样本I为已经记忆的那个样本，那么这两组长期记忆将使神经元输出回忆到原来的状态。

(2)ART网络的控制信号。

F_T、F_B的输出信号是由信号的流向进行控制的，在F_B中的神经元满足2/3规则。这个规则是：F_B的输入由三方面信号控制，即输入信号I、增益控制G和从F_T返回的信号。如果这三个信号中存在两个信号输入，则F_B中的神经元状态x_i就可以改变。第一种情形：如果外界有输入信号I，增益控制器G接受输入信号I，输出一个加强信号到F_B，此时F_B受输入I和控制器G两个信号控制，F_B层神经元状态$x_i(i=1,2,\cdots,n)$会受到I的影响，如图2-19(a)所示。另一种情形：如果F_T有信号输出，F_T将返回信号送到F_B层，同时输出一个抑制信号到G，使G对F_B的输出抑制。这样F_B层神经元状态x_i由输入I和F_T输出返回信号控制，如图2-19(b)所示。除了这两种情况，其他的情况，即只有一方面的信号控制，都不能使F_B中状态x_i改变。这个规则可以防止信号流对F_B的控制发生混乱。

在图2-18中，还有一个控制输出器ρ，ρ是当底层F_B接受到输入I和F_T返回信号时才产生作用，ρ称为ART匹配控制器。当F_T的返回信号与外界输入I匹配程度较好时，则ρ控制器无输出；当F_T返回信号与外界输入不匹配时，则ρ输出，并将F_T中"赢"的那个神经元抑制，新的竞争重新开始。

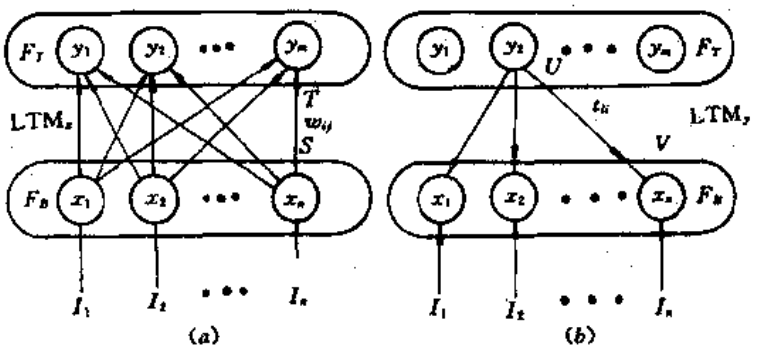


图2-20 ART模型的长期记忆

2.4.1.2 ART的工作流程

(1)自F_B到F_T的流程。

输入样本I，$I\in R^n$，它的每个分量加到F_B的每个神经元上。由于输入I的存在和增益控制器G对F_B中每个神经元有输入，满足2/3规则，F_B中产生了神经元状态$X=(x_1,x_2,\cdots,x_n)^T$。此时F_B对ρ控制器输出为负值，抑制了ρ控制器的输出。如图2-19(a)。

X状态经过F_B中神经元的变换，产生其输出$S=(s_1,s_2,\cdots,s_n)$。$s_i(x_i)$为F_B中第i个神经

元的输出，$i=1,2,\cdots,n$。通过长期记忆权(LTM_x)，对 s 进行累加形成 F_T 中每个神经元的输入

$$T_j=\sum_{i=1}^{n}s_i(x_i)w_{ij}\qquad j=1,2,\cdots,m \tag{2-31}$$

F_T 中的输入向量 $T=(T_1,T_2,\cdots,T_m)^T$。

当 T 输入 F_T 层后，得到一个新的短期记忆 $Y\in R^m$，F_T 层中第 j 个神经元状态 y_j 各不相同。简述上述流程为

$$I\longrightarrow \underset{(STM_B)}{X}\longrightarrow \underset{(LTM_z)}{S}\longrightarrow T\longrightarrow \underset{(STM_T)}{Y}$$

流程如图 2-20(a)。

(2)在 F_T 中的竞争过程。

F_T 中的状态 y_i 可以经过竞争算法或排序方法得到一个最大的输出 y_l，并抑制了其它的输出。假定状态 y_l 的输出 $u_l(y_l)$，那么 $u_l(y_l)=1$，而 $u_j(y_j)=0$，$j\neq l$。可以说第 l 个单元是“赢者”。

(3)从 F_T 反馈回 F_B 的过程。

从 $u_l(y_l)=1$ 回到 F_B 是经过一个长期记忆权 LTM_y 进行的。图 2-20(b)表示从 F_T 回到 F_B 的情况。长期记忆权 LTM_y 用 $t_{li}(i=1,2,\cdots,n)$来表示，F_T 层中第 l 个单元的输出 $u_l(y_l)$乘上 t_{li}加到 F_B 中的神经元 i 的输入端，这种映射为 V. 根据图 2-19(b)，F_T 的输出还产生一个抑制信号，该信号去抑制增益矩阵 G。这时输入到 F_B 中的信号由两部分组成，一部分是 V，另一部分是 I。根据 2/3 规则，将改变 F_B 中的状态 X 为 X^*。从 F_T 到 F_B 的过程可归纳为

$$\underset{(STM_T)}{Y}\longrightarrow \underset{(LTM_y)}{U}\longrightarrow V\longrightarrow \underset{(STM_B)}{X^*}$$

(4)匹配控制器 ρ 对 F_T 状态的控制。

当 ART 工作流程处于第一阶段，即仅有输入 I 和增益控制器 G 作用到 F_B 层时，ρ 控制器没有输出。而当 F_T 层的返回信号 V 和输入 I 同时作用于 F_B 层时，ρ 控制器根据 V 和 I 的匹配程度发出控制信号。如果匹配程度低，则 ρ 控制器有输出，迫使 F_T 层中的 y_l 为 0 值，这时 F_B 中的状态 X 不变。因为 y_l 被认为不可能再“赢”，则恢复原状态 X，并在此基础上重复上述过程直到 V 和 I 的匹配程度高为止；如果 V 和 I 匹配程度高，ρ 控制器没有输出。

2.4.2 自组织映射模型

自组织映射模型是由 Kohonen 提出来的，Kohonen 网络作为无导师学习的神经网模型广泛地应用于样本分类、样本排序和样本检测等方面。

2.4.2.1 Kohonen 网络的结构

Kohonen 网络是由输入层和输出层两层神经网络组成的。如图 2-21 所示。输入层中的每一个神经元，通过权与输出层的每一个神经元相联。输出层中的神经元一般是以二维形式排列的，它们中的每个神经元是输入样本的“映象”。在输出层中竞争是这样进行的：对于“赢”的那个神经元 c，在其周围的 N_c 区域内神经元在不同程度上得到兴奋，而在 N_c 以外的神经元都被抑制，这个 N_c 区域可以是正方形，也可以是六角形，如图 2-22 所示。N_c 是时间 t 的函数，随着 t 的增加，N_c 的面积成比例地缩小，最后只剩下一个神经元，也可能是一个组的神经元，它们反映了一类样本的属性。如果输入样本用向量 X^k 表示，$k=1,2,\cdots,m$，共有 m 个样本，样本的分

量 x_i^k 与输入的第 i 个神经元相联，输入的第 i 个神经元与输出层第 j 个神经元之间的权为 μ_{ij}，输出层第 j 个神经元输出为 y_j。

输出神经元 y_j 满足下列状态方程

$$\frac{dy_j}{dt} = \sum_{i=1}^{n} \mu_{ij} x_i^k - r(y_j) \qquad j = 1,2,\cdots,n_1 \tag{2-32}$$

其中 μ_{ij} 是输入层第 i 个神经元与输出第 j 个神经元间的权，x_i^k 为第 k 个输入样本的第 i 个分量，n 为输入层神经元个数，n_1 为输出层神经元个数。

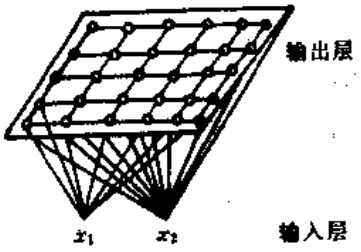


图 2-21 Kohonen 网络结构

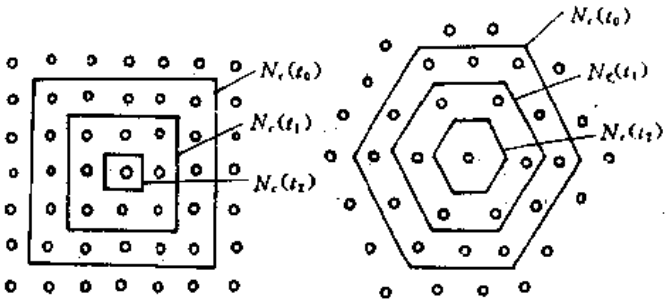


图 2-22 $N_c(t)$的形状及变化

式中第一项为输入的加权累加；第二项是一个与 y_j 有关的函数，此函数是非线性的，它的效果是使 y_j 的变化速率变慢。当外界没有输入或输入的加权累加比较小时，输出单元 y_j 的值减小，直到 0 为止；当上式的第一项比较大时，y_j 增长快，但 y_j 的增加又会引起 $r(y_j)$的增加，图 2-23(a)为一个非线性的 $r(y_j)$关系，直到$\frac{dy_j}{dt}=0$。y_j 可用下式描述：

$$y_j = \sigma\left(\sum_{i=1}^{n} \mu_{ij} x_i^k\right)$$

$\sigma(\cdot)$是一个单调上升的非线性函数，如图 2-23(b)所示。

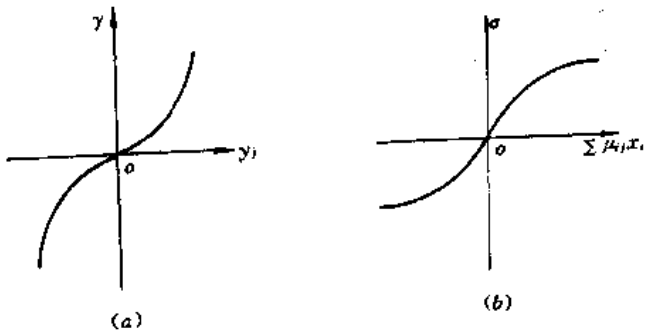


图 2-23 输出函数关系

2.4.2.2 Kohonen 网络的工作过程

输入样本 $X^k=(x_1^k,x_2^k,\cdots,x_n^k)(k=1,2,\cdots,m)$施加于输入层的 n 个神经元上，通过输入层与输出层之间的连接权 μ_{ij}，按照其对应的动态方程进行演化产生输出 $y_j(j=1,2,\cdots,n)$。竞争在输出层 y_j 中进行。那个 y_j 值最大的单元就是竞争后“赢”的神经元。μ_{ij}的学习是满足 Hebb 规则的，变化正比于输入与输出状态值的乘积：

$$\frac{\mathrm{d}\mu_{ij}}{\mathrm{d}t} = \alpha y_j x_i^k - \beta(y_j)\mu_{ij} \tag{2-33}$$

式中 α 是一个常数，前一项服从 Hebb 规则，当 x_i 与 y_j 都兴奋时，μ_{ij}的增长较快，后一项 $\beta(y_j)\mu_{ij}$是一个遗忘因子，当外界没有输入时，权 μ_{ij}会随时间而减小，这反映了遗忘特性。在设计中，直接应用式(2－33)是较少的，因为竞争在算法中只要对比 x_i 与 μ_{ij}的相近程度就可得到。

为了清楚看出竞争学习的意义，考虑一下特殊情况。当 y_j 为 0、1 二值函数，且满足 $y_j=0$ 时，$\beta(y_j)=0$；$y_j=1$ 时，$\beta(y_j)=\alpha$，则方程(2－33)变为

$$\begin{cases} \dfrac{\mathrm{d}\mu_{ij}}{\mathrm{d}t} = \alpha y_j x_i^k - \beta(y_j)\mu_{ij} = \alpha(x_i^k - \mu_{ij}), & y_j = 1 \\ \dfrac{\mathrm{d}\mu_{ij}}{\mathrm{d}t} = 0, & y_j = 0 \end{cases} \tag{2-34}$$

由上式派生出来的算法结果是：使学习后的权 μ_{ij}越来越靠近输入的 x_i^k，即第 k 个输入样本的第 i 个分量被记忆在输入第 i 个单元与输出层“赢”者(第 j 个单元)之间的权中。把式(2-34)写成矢量形式，即

$$V_j = (\mu_{1j}, \mu_{2j}, \cdots, \mu_{nj})^T, X^k = (x_1^k, x_2^k, \cdots, x_n^k)^k$$

则

$$\dot{V}_j = \alpha(X^k - V_j)$$

这里 α 是时间和距离的函数。以“赢”的那个神经元所在位置为中心，比较靠近“赢”的神经元的那些神经元权调整系数 α 大，而远离的那些神经元的 α 小。那个“赢”的神经元周围的调整区域 $N_c(t)$随时间而减小，直到只剩下一个神经元为止，如图 2-22 所示。权调整系数 α 的函数关系可根据需要任取，这里不再专门介绍。

从上面的讨论可以看出，Kohonen 网络有如下特点：

(1)网络中的权是输入样本的记忆。如果输出神经元 j 与输入 n 个神经元之间的连接权用 V_j 表示，对应于某一类样本 X^k 输入，使 y_j 达到匹配最大，那么 V_j 通过学习后十分靠近 X^k，因此以后当 X^k 再次输入时，y_j 这个神经元必定会兴奋，y_j 是样本 X^k 的代表。

(2)网络学习时对权的调整，不只是对兴奋的那个神经元所对应的权进行，而对其周围 N_c 区域内的神经元同时进行调整。因此对于在 N_c 内的神经元可以代表不只是一个样本 X^k，而是与 X^k 比较相近的样本都可以在 N_c 内得到反映，因此这种网络对于样本的畸变和噪声的容差大。

(3)网络学习的结果使比较相近的样本在输出二维平面上的位置也比较近。

第 3 章　神经网络的学习机理

3.1　人工神经网络的学习功能

揭示并模拟大脑神经系统的学习机理是研制新一代智能信息处理系统的关键之一。传统计算机的智能来自编程者，是预先设置的。在求解时对随机事件、模糊问题难以处理，对环境的自适应性很差。而大脑的智能来自两方面——遗传和后天的学习。后天的学习对智能的形成起着非常重要的作用。所以，大脑能随环境的变化而不断地学习，表现出比计算机更强的智能。

学习过程中神经系统究竟哪些部分发生了变化，这是神经系统学习的实质问题。当前神经电生理学，组织学和行为实验学为解决这一问题提供了线索，从而形成了两大学派。一是化学学说，认为神经系统把学习后的信息记录在某些生物分子(蛋白质、核酸、神经递质等)上，正像遗传信息记录在 DNA 上一样；另一是突触修正假说，认为学习过程中神经元之间的突触联系发生了变化。这两种假说中，后一种为多数神经生理学家所接受。

人工神经网络是对生物神经系统的模拟。其信息处理功能是由网络的单元(神经元)的输入输出特性(激活特性)，网络的拓扑结构(神经元的连接方式)所决定的。按突触修正假说，神经网络在拓扑结构固定时，其学习归结为连接权的变化。

到目前为止，已经出现许多神经网络模型及相应的学习算法。对学习算法的分类也有多种，如联想式与非联想式学习，以区别来自环境刺激模式的多少；监督与非监督学习，以区别学习时有无教师示教；以及以网络连接形式(阶层还是相互连接)的学习分类。在人工神经网络中，权是一个反应信息存贮的关键量，在结构和转换函数定了以后，如何设计权使网络达到一定的要求，这是人工神经网络必不可少的部分，大多数神经网络权的设计是通过学习得到的，目前可分为下列几种。

(1)死记式学习。

网络的权是事先设计的，值是固定的。如 Hopfield 网络在做优化时，权可根据优化的目标函数和约束条件来设计，一旦设计好了就不能变动。早期的 M-P 模型也是用设计好的固定权来完成与、或、非等逻辑关系的。

(2)δ 学习律。

这种方法是用已知例子作为教师对网络的权进行学习，称为 δ 学习律。其规则是通过神经网络理想输出和实际输出之间的误差来修正网络的权。在很多神经网络中，都采用了这种 δ 学习律，如 Perceptron Adaline 和 Back-propagation 算法等。

(3)自组织的学习和 *Heddian* 学习律。

两个神经元之间的连接权，正比于两个神经元的活动值，如 v_i，v_j 表示两个神经元的输出值，则它们之间的权的变化为

$$\Delta w_{ij} = \eta v_i v_j$$

这里 η 为步长或常数。

(4)相近学习。

设 w_{ij} 为从神经元 i 到神经元 j 的权，v_i 为神经元的输出，则

$$\Delta w_{ij} = a(v_i - w_{ij})$$

在这个学习中，使 w_{ij} 十分逼近 v_i 的值。如 Kohonen 和 ART 等都采用这类学习方法。

下面各节分别详细介绍各种常用学习算法。

3.2 误差修正型学习

误差修正型学习是一个监督学习过程，其基本思想是利用神经元希望输出与实际输出之间的误差作为连接权调整的参考，最终减小这种误差。下面介绍这种最基本的误差修正规则。

δ 规则

这种规则(或称学习律)是用已知例子作为教师对网络的权进行学习，称为 δ 规则。

设 $(X^i, \overline{Y}^i)$，$i=1,2,\cdots,k$ 为已知的输入、输出例子(或称样本)。X^i，$\overline{Y}^i$ 为 n 和 m 维矢量。$X^i=(x_1^i, x_2^i, \cdots, x_n^i)^T$，$\overline{Y}^i=(\overline{y}_1^i, \overline{y}_2^i, \cdots, \overline{y}_m^i)^T$。在输入 X^i 的作用下，网络的实际输出为 $Y^i=(y_1^i, y_2^i, \cdots, y_m^i)^T$。设任一神经元 i 到输出神经元 j 的权 w_{ij} 的改变量为

$$\Delta w_{ij} = \eta \delta_j v_i \tag{3-1}$$

$$\delta_j = F(\overline{y}_j^i - y_j^i) \tag{3-2}$$

其中 η 为步长，$\overline{y}_j^i - y_j^i$ 为误差(即希望值与实际值之差)，v_i 为第 i 个神经元的输出。$F(\cdot)$ 函数根据不同的情况而定，多数神经元网络 $\delta_j=\overline{y}_j^i-y_j^i$；$F(x)=x$。

在许多神经网络中，都采用了 δ 规则，如 Perceptron Adaline 和 Back-Propagation 算法等。

3.2.1 感知器学习

感知器网络是由 Rosenblatt 于 1957 年提出的，后来他本人进行了广泛深入的研究。这里介绍的是最简单的感知器的一种形式，其拓扑结构如图 3-1 所示。

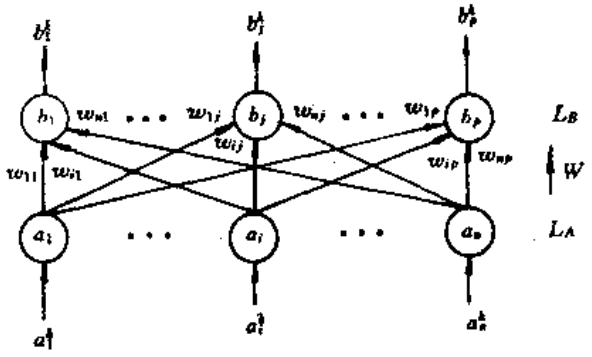


图 3-1 感知器拓扑结构

3.2.1.1 感知器学习算法

网络用误差修正规则(δ 规则)学习。存储样本对 $(A^k, B^k)k=1,2,\cdots,m$。第 k 个模式对中的输入矢量 $A^k=(a_1^k, a_2^k, \cdots, a_n^k)$ 为模拟值模式，输出矢量 $B^k=(b_1^k, b_2^k, \cdots, b_p^k)$ 为二值模式。网络中，L_A 层的 n 个单元对应模式 A^k 的 n 个分量，L_B 层的 P 个单元对应模式 B^k 的 P 个分量。网络离线学习，按离散时间方式回想。

由于 L_B 层中每个单元只取值+1 或−1，因此可将它视作输入模式 $A^k(k=1,2,\cdots,m)$ 两个可能的分类。在学习开始时，由各连接权决定的超平面随机地被放到 n 维空间。随着学习的进行，这个超平面渐渐移动，直到它能将两类模式恰当划分为止。学习算法描述如下：

(1)连接权 w_{ij}初始化。将 L_A 层到 L_B 层单元的连接权 w_{ij},$i=1,2,\cdots,n$,$j=1,2,\cdots,p$ 及 L_B 层单元阈值 θ_j,$j=1,2\cdots,p$ 赋予[−1,+1]间的随机值。

(2)对每一模式对(A^k,B^k)$k=1,2,\cdots,m$ 完成下面操作;

①将 A^k 的值送到 L_A 层单元,L_A 层单元的输出之加权和作为 L_B 层单元的输入,计算 L_B 层单元的输出

$$b_j = f\left(\sum_{i=1}^{n} w_{ij}a_i - \theta_j\right) \tag{3-3}$$

式中 $j=1,2,\cdots,p$,f 为双极值阶跃函数

$$f(x) = \begin{cases} 1, & x > 0 \\ -1, & x \leqslant 0 \end{cases} \tag{3-4}$$

②计算 L_B 层单元希望输出与实际输出间误差

$$d_j = b_j^k - b_j \quad , \quad j = 1,2,\cdots,p \tag{3-5}$$

③调整 L_A 层单元与 L_B 层单元之间的连接权

$$\Delta w_{ij} = \eta a_i d_j \tag{3-6}$$

式中 $i=1,2,\cdots,n$,$j=1,2,\cdots,p$,η 为常数 $0<\eta<1$。

(3)重复步骤(2)直到误差 d_j($j=1,2,,\cdots,p$ 且 $k=1,2,\cdots,m$)变得足够小或变为零为止。

在这个学习过程中可以看出,当实际输出与希望输出相同时,误差为零,连接权保持不变;当实际输出不同时,连接权增加或减去两倍的 ηa_i。学习时对各种不同的输入模式,需轮换地反复地进行以上操作。要注意的是,L_B 层单元的阈值 θ_j 可视作一固定输出−1 的单元到输出单元 j 的连接权,因此,可将 θ_j 的学习合并在 w_{ij}中考虑,即 $\theta_j=w_{0j}$。

学习后的网络在回想时(Recall),使用公式(3-3)。如果将 θ_j 合并在 W 中,即 $\theta_j=w_{0j}$则式(3-3)变为

$$b_j = f\left(\sum_{i=0}^{n} w_{ij}a_i\right) \quad , \quad j = 1,2,\cdots,p$$

其中 $a_0=-1$。从上式可以看出,当且仅当 $\sum_{i=0}^{n} w_{ij}a_i > 0$ 时,把 A^k 划为"+1"类,当且仅当 $\sum_{i=0}^{n} w_{ij}a_i \leqslant 0$ 时,把 A^k 划为"−1"类。这就是说,输入模式空间中的模式 $A^k(k=1,2,\cdots,m)$是由超平面 $\sum_{i=0}^{n} w_{ij}a_i = 0$ 划分的。很容易理解,如果输入模式是线性可分的,学习后能对输入模式正确分类;如果输入模式本身是线性不可分的,那么学习后的网络不能对输入模式正确分类。

3.2.1.2 算法的讨论

在第 2 章感知器模型一节已经讨论了单层感知器在分类中的局限性,解决线性不可分问题的途径是增加感知器的层数。此外,可以看出学习算法中采用的是 δ 规则,这种网络需要较长的离线学习才能达到收敛。然而感知器分类的性质决定了它能存储丰富的输入输出模式对(只要输入模式是线性可分的)。感知器的优点还表现在它能对模式的直接联想和容易理解的工作方式。

感知器学习中的一个重要性质,就是学习的收敛定理。

定理 当输入模式线性可分时,误差修正过程必在有限次数内收敛,这时所得到的各连接

权而形成的超平面，能对所有输入模式正确分类。

下面介绍在形式上与 δ 规则一致的 LMS 算法(Least Mean Square Algorithm)。

3.2.2 Adaline/Madaline 学习

3.2.2.1 Adaline 模型

自适应线性神经元(Adaptive Linear Neuron，简称 Adaline)，网络是由 Widrow 和 Hoff 于 1960 年提出的一种网络模型。它是一个自适应可调的网络，适用于信号处理中的自适应滤波，预测和模型识别。构成这种网络的基本单元 Adaline 是一个类似于感知器单元，如图 3-2 所示。如在第 k 时刻，有向量 X_k 输入，权向量为 W_k，此时有输出 y_k(模拟量)和二进制输出 q_k。y_k 与要求的理想响应的差值，通过 LMS 算法，修改 W_k，从而减少了 y_k 与理想响应的误差。这个单元的输入与输出关系满足：

$$y_k = \sum_{i=1}^{n} w_{ik}x_{ik} - \theta_k$$

$$q_k = \mathrm{sgn}(y_k) \qquad (3\text{-}7)$$

如使 $\theta_k = w_{0k}, x_{0k} = -1$

则 $y_k = \sum_{i=0}^{n} w_{ik}x_{ik}$ 。

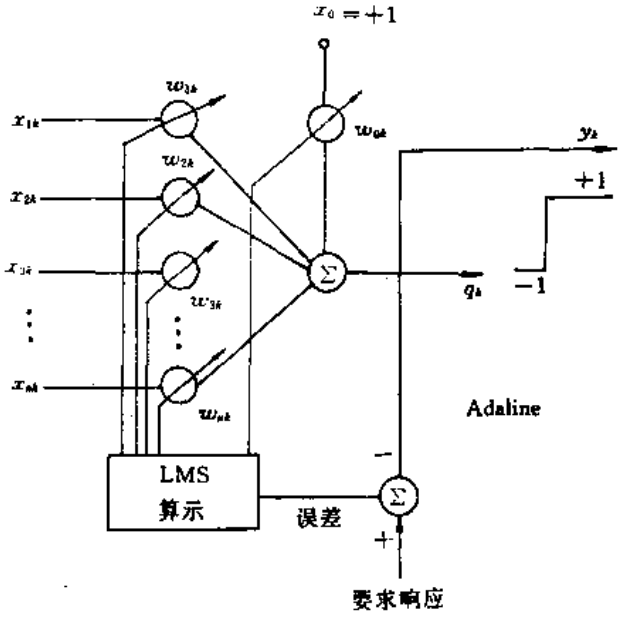


图 3-2 Adaline 原理图

比较式(3-7)与(2-10)，其差别只是输出分为模拟和数字两部分。从数字部分看，与感知器单元完全相同，它可进行线性分割。从模拟输出看，它是作为误差调节之用，对单个 Adaline，其误差 为模拟输出和理想输出，也为模拟量。用 LMS 算法能保证这种网络在自适应学习时的收敛性。Adaline 的特点是：可以根据外界输入的情况和要求的响应随时学习，十分灵活。

3.2.2.2 Madaline 模型

对于单层的 Adaline 网络，由于它的形式与感知器相同，所以只能适用于一些线性分割的情况。为了实现非线性分割可以采用 Madaline 网络，Madaline 网络是由多个 Adaline 单元组合成的多层网络，图 3-3 为两个 Adaline 和一个与门组合成的 Madaline 网络。它的非线性分割与多层感知器一样，如输入矢量为 n 维，利用组合很多的 $n-1$ 维超平面来实现。

单层 Adaline 权的修正用 LMS 算法来完成。

3.2.2.3 LMS 算法

设 Adaline 输入信号为模拟量 $X_k=(x_{1k},x_{2k},\cdots,x_{nk})^T$ 输出为两个量，一个量是模拟量 y_k，提供给 LMS 算法用来调整该层的输入权，另一个量是数字量，用于多层 Adaline 网络层间的传递信号，单层 Adaline 原理图如图 3-2 所示，其中理想输出用 y_{k0} 表示。

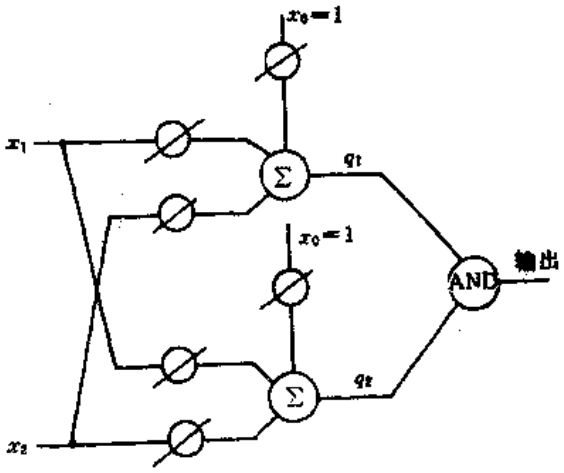


图 3-3 Madaline 模型

算法步骤：

(1)网络中初始化权和阈值

$W_i(0)=\beta$ random$(\cdot)$　　对所有的 i

随机数前面的系数 $0<\beta<1$。

(2)任选一个输入向量 X_k 作为网络输入，计算实际输出为

$$y_k=\sum_{i=1}^{n}w_{ik}x_{ik}-\theta_k$$

$$q_k=\mathrm{sgn}(y_k)$$

如使 $\theta_k=w_{0k}$，$x_0=-1$，则

$$y_k=\sum_{i=0}^{n}w_{ik}x_{ik} \tag{3-8}$$

(3)调整权值。设 k 为迭代次数，则权值调整的向量表达式为

$$W_{k+1}=W_k+\frac{a}{|X_k|^2}\varepsilon_k X_k,\qquad 0<a<1$$

$$\varepsilon_k=y_{k0}-y_k \tag{3-9}$$

其中 W_{k+1} 为下一次权值向量的取值，W_k 为现在的权向量，X_k 为现在的输入向量，ε_k 为现在的误差(理想输出与模拟(实际)输出之间的差值)，a 为系数，$|X_k|$ 为输入向量的模。

LMS 算法通过调整单个神经元的权值，以使误差 ε_k 为最小。由 LMS 规则，权的变化值可以写为

$$\Delta W_k=W_{k+1}-W_k=\frac{a}{|X_k|^2}\varepsilon_k X_k \tag{3-10}$$

比较式(3-6)与(3-10)，权的调整具有相同的形式，即 LMS 算法在形式上与 δ 规则是一致的。我们讨论一下式(3-10)的几何意义。由误差增量

$$\Delta\varepsilon_k=\Delta(y_{k0}-X_k^TW_k)=-X_k^T\Delta W_k$$

则

$$\Delta\varepsilon_k/\Delta W_k=-X_k \tag{3-11}$$

可以看出，δ 规则和 LMS 算法虽然在形式上相同，这两种网络学习的本质区别在于：前者的数学基础是超平面位置调整，后者的数学基础是误差曲面上的梯度下降。图 3-4 给出了 LMS 规则的几何意义图示。

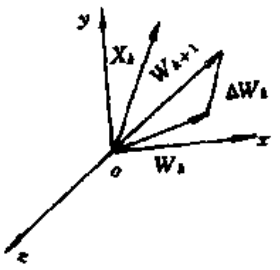


图 3-4 LMS 规则的权值修正

权调整矢量 ΔW_k 与输入矢量 X_k 一致，即与误差的负梯度方向一致。误差的改变量 $\Delta\varepsilon_k$ 为乘积 $-X_k^T\Delta W_k$，当 ΔW_k 极小时就修正了误差。

3.2.2.4 多层 Adaline 网络学习算法

多层 Adaline 网络如图 3-5 所示。图中 AD 表示 Adaline 的符号。

我们知道，用 LMS 算法采用进行学习时，必须知道 Adaline 的相应误差 ε。这样 LMS 规则不能直接应用到图 3-5 所示的多层网络中去。(因为隐层 Adaline 的相应误差是不能直接知道的。多层的 Adaline 的学习分两部分进行，先利用 MRII 的算法计算出每一层的输出要求，然后再采用单层的 LMS 算法来完成每一层间的权的学习。如果输入是一个模拟量，第一层的 Adaline 的输出是两个量，一是模拟量 y_k，提供给 LMS 算法用来调整该层的输入权，另一是数字量 q_k，输入到下一层的网络。这样从第二层 Adaline 开始，其输入矢量是二进制的量，MRII 算法就是建立在这些二进制的基础上。它的思想是，对网络干扰最小的二进制输出量，改变其符号，如改变第一层中某一个 Adaline 输出的符号，(从+1→−1，或从−1→+1)观察其总的网络的输出响应和要求的值之间的误差，如误差减小，就确认这种改变，若误差不减小，恢复原来符号，每次选中的 Adaline 应是那些模拟量最接近于 0 的单元，这样可使网络的干扰最小。

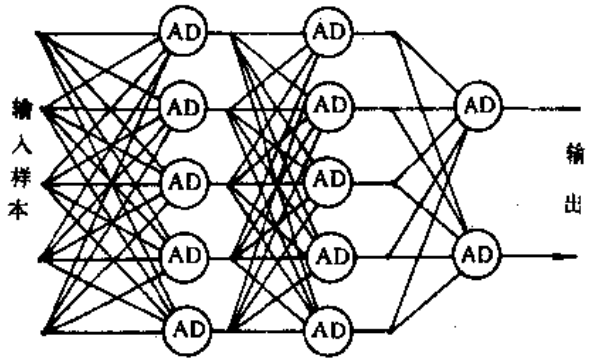


图 3-5 多层 Adaline 网络

MRII 的学习步骤：

(1)初始化多层 Adaline 网络中的权，用一个随机数作为各个权的初始权值。

(2)输入一个样本矢量 X_k 和要求的输出 t_k，用式(3-8)按层一步一步地计算出实际输出，并求出要求的输出 t_k 和实际输出的误差。

(3)根据最小干扰原则，从第一层开始，找第一层中神经元模拟输出最接近于 0 的那个神经元，让它的输出数字量改变符号。

(4)计算网络最后的输出和要求响应之间的误差，如果误差减小，则这种改变被接受，如没有减小，上一步改变的输出数字量的符号恢复。权值恢复到原来的值。

(5)第一个神经元训练结束后，转入下一个神经元，即模拟输出第二个按近零的单元，仍按上面的规则训练。

(6)当这一层单个神经元的训练完了，再按两个一组地训练，然后再按三个一组，…直至输出与要求响应之间的距离减到最小为止。然后用同样的方法训练第二层、第三层，到相同时为止。

(7)另一个新的样本输入，用同样的方法进行训练，一直到达到误差最小为止。

(8)通过 MRII 的学习可以得到每个样本输入对应的每一层相应的输出，从而对每一单层的 Adaline 可以用 LMS 算法进行学习，得到全部权的解。

单层的学习算法如下：

$$W(n_0+1)=W(n_0)+\frac{a}{\|X(n_0)\|}\varepsilon(n_0)X(n_0) \tag{3-12}$$

这里，$W(n_0+1)$为 $W(n_0)$的下一个时刻的权矢量，$X(n_0)$为当前输入样本矢量，$\varepsilon(n_0)$为当

前的误差。n_0 为迭代的次数。

如果 $X(n_0)$ 矢量的分量为±1，那么 $\| X(n_0) \|$ 为

$$\| X(n_0) \|^2 = x_1^2(n_0) + x_2^2(n_0) + \cdots + x_{n+1}^2(n_0)$$

对于每个调节周期，可按照式(3-12)进行调节。误差 $\varepsilon(n_0)$ 为

$$\varepsilon(n_0) = t(n_0) - X^T(n_0)W(n_0)$$

当权改变时，其误差改变为式(3-11)所示

$$\Delta\varepsilon(n_0) = - X^T(n_0)\Delta W(n_0)$$

由式(3-12)可得

$$\Delta W(n_0) = \frac{a}{\| X(n_0) \|}\varepsilon(n_0)X(n_0) \tag{3-13}$$

综合式(3-11)和式(3-13)得

$$\Delta\varepsilon(n_0) = - a\varepsilon(n_0) \tag{3-14}$$

这说明，当网络的输入不变时，误差的变化是缩小当前误差的 a 倍，选择 a 因子可以控制收敛的速度和稳定性，a 太大，误差将可能增大。一般取 $1.0>a>0.1$。

3.2.3' 反向传播网络学习

反向传播神经网络(Back-Propagation Neural Networks)的结构如图 3-6 所示，又称 BP 模型。在这一神经网络模型中引入了中间隐含神经元层。故标准的 BP 模型由三个神经元层次组成，其最下层称为输入层，中间层称为隐含层，最上层称为输出层。各层次之间的神经元形成全互连接，各层次内的神经元之间没有连接。

BP 网络用作异联想函数估值器，它能存储任意连续值模式对 (A_k, C_k)，$k=1,2,\cdots,m$。在第 k 个模式对中，模拟值模式 $A_k=(a_1^k, a_2^k, \cdots, a_n^k)$，$C_k=(c_1^k, c_2^k, \cdots, c_q^k)$。网络通过多层误差修正梯度下降法离线学习，按离散时间方式进行。其中输入层的 n 个单元对应于 A_k 的 n 个分量，而输出层的 q 个单元对应于 C_k 的 q 个分量。

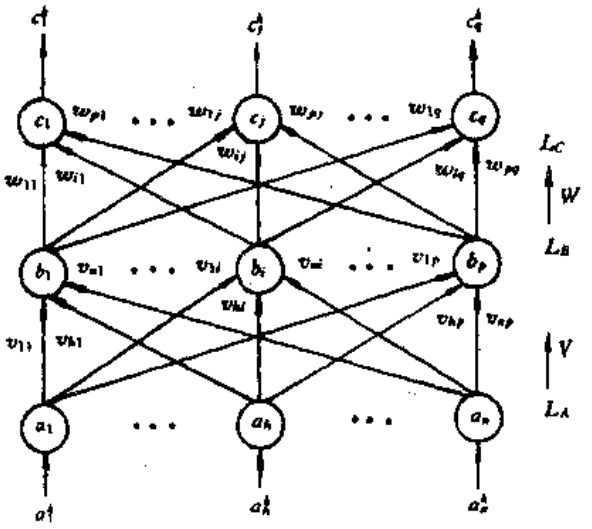


图 3-6 最基本的 BP 网络拓扑结构

误差反向传播算法(BP 算法)是通过一个使代价函数最小化过程来完成输入到输出的映射。通常代价函数定义为所有输入模式上输出层单元希望输出与实际输出的误差平方和。当然，代价函数也可以有其它形式。

3.2.3.1 BP 学习算法

下面以代价函数取误差平方和的形式为例，说明图 3-6 所示网络的误差反向传播学习算法。这个学习算法的数学原理将在这之后给予介绍，误差反向传播学习算法的步骤如下：

(1)给输入层单元到隐含层单元的连接权 $v_{hi}, h=1,2,\cdots n, i=1,2,\cdots,p$，隐含层单元到输出层单元连接权 $w_{ij}, i=1,2,\cdots p, j=1,2,\cdots q$ 以及隐层单元的阈值 θ_i，输出层单元的阈值 r_j 赋予(−1,+1)区间的随机值；

(2)对于样本模式对 $(A_k, C_k)(k=1,2,\cdots m)$ 进行下列操作：

①将 A_k 的值送到输入层单元，通过连接权矩阵 V 送到隐层单元，产生隐层单元新的激活值

$$b_i = f\left(\sum_{h=1}^{n} v_{hi} a_h + \theta_i\right) \tag{3-15}$$

式中 $i=1,2,\cdots,p$，f 为S型函数：

$$f(x) = (1+\mathrm{e}^{-x})^{-1} \tag{3-16}$$

②计算输出层单元的激活值

$$c_j = f\left(\sum_{i=1}^{p} w_{ij} b_i + r_j\right)$$

$$j = 1,2,\cdots,q \tag{3-17}$$

③计算输出层单元的一般化误差

$$d_j = c_j(1-c_j)(c_j^t - c_j) \tag{3-18}$$

式中 $j=1,2,\cdots,q$，c_j^t 为输出层单元 j 的希望输出；

④计算隐含层单元对于每个 d_j 的误差

$$e_j = b_i(1-b_i)\sum_{j=1}^{a} w_{ij} d_j \tag{3-19}$$

式中 $i=1,2,\cdots,p$；上式相当于将输出层单元的误差反向传播到隐含层；

⑤调整隐含层到输出层的连接权

$$\Delta w_{ij} = \lambda\, b_i\, d_j \tag{3-20}$$

式中 $i=1,2,\cdots,p$ 和 $j=1,2,\cdots,q$，λ 为学习率 $0<\lambda<1$。

⑥调整输入层到隐含层的连接权

$$\Delta v_{hi} = \beta\, a_h\, e_i \tag{3-21}$$

式中 $h=1,2,\cdots,n$ 且 $i=1,2,\cdots,p$，$0<\beta<1$；

⑦调整输出层单元的阈值

$$\Delta r_j = \lambda\, d_j \tag{3-22}$$

式中 $j=1,2,\cdots q$；

⑧调整隐含层单元的阈值

$$\Delta\theta_i = \beta\, e_i \tag{3-23}$$

式中 $i=1,2,\cdots,p$；

(3)重复步骤(2)，直到对于 $k=1,2,\cdots,m$ 误差 $d_j, j=1,2,\cdots,q$ 变得足够小或变为零为止。

从上面的学习过程可以看出，误差反向传播学习分成两个阶段；在第一阶段，对于给定的网络输入，通过现有连接权将其正向传播，获得各个单元的实际输出，在第二阶段，首先计算出输出层各单元的一般化误差，这些误差逐层向输入层方向逆向传播，以获得调整各连接权所需各单元参考误差。

学习后的网络在回想时使用正向传播公式，即式(3-15)和式(3-17)。

3.2.3.2 BP 算法的数学原理

误差反向传播学习实现了定义在整个模式训练集上代价函数曲面上的梯度下降，它可以看作是 LMS 算法在多层网络中的推广，这一点可以从误差反向传播学习算法的推导过程看出。

设 E_k 为给网络提供模式对(A_k, C_k)时输出层上的代价函数，因而整个模式训练集上的全局代价函数为

$$E = \sum_{k=1}^{m} E_k$$

对于第 k 个模式对，输出层单元 j 的加权输入(将阈值 θ_j 归入权系数中)为

$$Netc_j = \sum_{i=1}^{p} w_{ij} b_i \tag{3-24}$$

该单元的实际输出为

$$c_j = f(Netc_j) \tag{3-25}$$

而隐含层单元 i 的加权输入为

$$Netb_i = \sum_{h=1}^{n} v_{hi} a_h \tag{3-26}$$

该单元的实际输出为

$$b_i = f(Netb_i) \tag{3-27}$$

式(3-25)和(3-27)中的函数 f 为可微分非递减函数。对于输出单元 j，定义一般化误差

$$d_j = -\sum \frac{\partial E_k}{\partial Netc_j} \tag{3-28}$$

上式可写成下列形式：

$$d_j = -\frac{\partial E_k}{\partial c_j} \cdot \frac{\partial c_j}{\partial Net\ c_j} = -f'(Netc_j)\frac{\partial E_k}{\partial c_j} \tag{3-29}$$

对于隐含单元 i，同样定义误差

$$e_i = -\frac{\partial E_k}{\partial Net\ b_i}$$

类似地，e_i 在下面形式上可视作前层误差逆传播到该层单元的误差

$$\begin{aligned} e_i &= -\frac{\partial E_k}{\partial b_i} \cdot \frac{\partial b_i}{\partial Netb_i} \\ &= f'(Netb_i)\left(-\frac{\partial E_k}{\partial b_i}\right) \\ &= f'(Netb_i)\left(-\sum_{j=1}^{q} \frac{\partial E_k}{\partial Netc_j} \cdot \frac{\partial Netc_j}{\partial b_i}\right) \\ &= f'(Netb_i)\sum_{j=1}^{a} w_{ij} d_j \end{aligned} \tag{3-30}$$

在现有的连接权 w_{ij} 和 v_{hi} 下，为了减小代价函数 E_k，可由梯度下降原则改变连接权。因此，有

$$\Delta w_{ij} = -\lambda \frac{\partial E_k}{\partial W_{ij}}$$

$$= -\lambda \frac{\partial E_k}{\partial Netc_j} \cdot \frac{\partial Netc_j}{\partial w_{ij}}$$

$$= -\lambda d_j b_i \tag{3-31}$$

同理

$$\Delta v_{hi} = -\beta \frac{\partial E_k}{\partial v_{hi}}$$

$$= -\beta e_i a_h \tag{3-32}$$

上面两式中，λ, β 为学习率($0<\lambda<1, 0<\beta<1$)。

由于全局代价函数是定义在整个训练集上的，要实现 E 曲面上真正的梯度下降，需在整个模式训练集中每一模式对提供给网络期间，保持连接权不变求出 E 对连接权 w_{ij} 的负梯度，即

$$-\frac{\partial E}{\partial w_{ij}} = \sum_{k=1}^{m}\left(-\frac{\partial E_k}{\partial w_{ij}}\right)$$

此时连接权的变化为

$$\Delta w_{ij} = -\lambda \frac{\partial E}{\partial w_{ij}}$$

$$= \sum_{k=1}^{m}\left(-\lambda \frac{\partial E_k}{\partial w_{ij}}\right) \tag{3-33}$$

和

$$\Delta v_{hi} = -\beta \frac{\partial E}{\partial v_{hi}} = \sum_{k=1}^{m}\left(-\beta \frac{\partial E_k}{\partial v_{hi}}\right) \tag{3-34}$$

从以上两式可以看出，连接权变化正比于整个模式集上各个模式对对应的负梯度之和，因此，这种学习算法称为累积误差逆传播算法。

然而，实际上往往是每给网络提供一个模式对，都计算单元的误差系数并进行连接权调整。因此，这种算法的梯度下降偏离了 E 上真正的梯度下降，但是当式(3-31)和(3-32)中的学习率 λ、β 足够小时，这种偏离是可以忽略的。这种每提供一模式对便对连接权进行一次调整的学习算法通常称为标准误差逆传播算法，前面描述的误差逆传播算法就是标准的误差逆传播学习算法。

需要指出的是，用标准逆传播算法时，一般要求将训练集上的模式对随机地提供给网络，这样可使网络获得更快的收敛速度。此外，当训练集并不太大时，累积逆传播学习比标准逆传播学习通常收敛更快。

如果代价函数定义为整个模式训练集上的误差平方和

$$E = \frac{1}{2}\sum_{k=1}^{m}\sum_{j=1}^{n}(c_j^k - c_j)^2 \tag{3-35}$$

且 f 为S型函数

$$f(x) = (1 + e^{-x})^{-1}$$

那么式(3-29)和式(3-30)表示的单元误差函数具有同式(3-18)和式(3-19)相同的形式；式(3-31)和式(3-32)表示的连接权变化与式(3-20)和式(3-21)具有相同的形式。

在BP网络中，逆传播学习要求单元的输入输出函数是可微分的。线性函数虽然是可微分的，但如果多层网络中所有单元的输入输出函数均采用线性函数，那么网络将失去应有的映射

能力，其原因在于单层线性网络与多层线性网络是等价的。关于 BP 网络的映射能力，许多人进行过研究，现总结成下面的完全性定理。

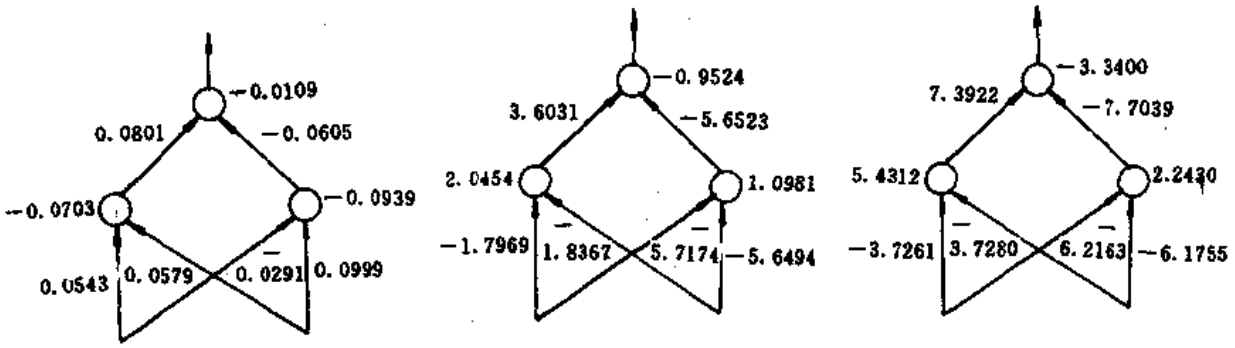
* 定理　假定 BP 网络中隐单元可以根据需要自由设定，那么一个三层网络，可以实现以任意精度近似任意连续函数。

该定理的证明需要繁锁的数学推导，这里不再给出。

误差逆传播学习仅仅实现了代价函数曲面上的梯度下降。由于 BP 网络中非线性隐含单元的存在，代价函数不再像 Adaline/Madaline 的 LMS 算法学习中的均方误差函数只有一个极小点，而是存在多个极小点。因此梯度下降不能保证求出全局最小。对于 BP 网络的误差曲面，有以下三个特点：第一，有很多全局的最小的解。存在一些平坦区，在此区内误差改变很小，这些平坦区多数发生在神经元的输出接近于 0 或 1 的情况下。对于不同的映射，其平坦区的位置，范围各不相同。第三，存在不少局部最小点，在某些初始值条件下，算法的结果会陷入局部最小，使算法不收敛。如何求得全局最小仍是需要进一步研究的问题。下面用一个简单的例子介绍标准误差反向传播算法。

3.2.3.3　BP 网络学习举例

用一个“2-2-1”BP 网络学习解决“异或”问题。



输入	希望输出	实际输出	误差
00	0	0.5	$E\doteq0.5$
01	1	0.5	
10	1	0.5	
11	0	0.5	

(a)

输入	希望输出	实际输出	误差
00	0	0.119	$E\doteq0.166$
01	1	0.727	
10	1	0.734	
11	0	0.415	

(b)

输入	希望输出	实际输出	误差
00	0	0.05	$E\doteq0.008$
01	1	0.941	
10	1	0.941	
11	0	0.078	

(c)

图 3-7　“2-2-1”BP 网络学习求解“异或”问题

(a)学习开始时的网络；(b)迭代 8000 次后的网络；(c)迭代 11050 次后的网络

设该网络的两个隐含单元和一个输出单元均为 S 型函数即 $f(x)=(1+e^{-x})^{-1}$。学习采用上面介绍的误差逆传播算法：每给网络提供一输入输出模式时，首先进行前向传播并计算出各单元的实际输出。在输出层计算出各单元的一般化误差，然后逆向往输入层传播，求出各单元的参考误差。当各单元的参考误差都求出后，进行连接权和各单元阈值的调整，这样便完成一次迭代。对下一模式对重复上述过程，当四个模式对各自的迭代都完成之后，又重复对第一模

式对的迭代，这样循环下去，直至输出层单元的误差满足要求为止。在该网络的学习中，学习率取 $\lambda=\beta=0.6$。

学习开始以前，给网络各单元之间的连接权及单元阈值随机地赋于〔-0.1，0.1〕之间的值，此时网络对四个模式所产生的输出都为 0.5 左右，全局误差为 0.5，当进行 8000 次迭代后，全局误差为 0.165876，若以输出小于 0.5 判为"0"，输出大于 0.5 判为"1"，可以认为此时网络已学会解决异或问题。

求解异或问题的学习过程如图 3-7 所示。从网络的学习过程可以看出，随着学习的不断进行，定义在整个模式训练集上的网络全局误差逐渐下降，而特定的模式对的输出误差在一个周期内不一定减小，这说明逆传播算法实现的是训练集上平方误差的梯度下降，而不是特定模式对误差的梯度下降。

3.2.3.4 BP 算法的改进

为了克服 BP 算法收敛速度慢和局部极小点问题，许多学者从不同的侧面对 BP 算法进行修正。

(1)变步长算法。

BP 算法是在梯度法基础上推算出来的，在一般最优梯度法中，步长 η 是由一维搜索求得的，求解有下面几个步骤：

①给定初始权 $W(0)$和允许误差 $\varepsilon>0$；

②计算误差 E 的负梯度方向 $d^{(n)}=-\nabla E(w)$；

③若 $\| d^{(n)} \| <\varepsilon$，则停止计算；否则从 $W(n_0)$出发，沿 $d^{(n)}$作一维搜索，求出最优步长 $\eta(n_0)$；

④进行权的迭代：$W(n_0+1)=W(n_0)+\eta(n_0)d^{(n_0)}$并转②。但是在 BP 算法中步长 λ(学习率)是不变的，其原因是由于 $E(W)$是一个十分复杂的非线性函数，很难通过最优求极小的方向得到最优的步长 λ。可是从 BP 网络的误差曲面看出，有平坦区存在，如果在平坦区上 λ 太小使迭代次数增加，而当 W 落到误差剧烈变化的地方，步长太大又使误差增加，反而使迭代次数增加影响了学习收敛的速度，变步长方法可以使步长得到合理的调节。这里推荐一种方法：先设一初始步长，若一次迭代后误差函数 E 增大，则将步长乘以小于 1 的常数 β 沿原方向重新计算下一个迭代点，若下一次迭代后误差函数 E 减小，则将步长乘一个大于 1 的常数 ψ，这样既不增加太多的计算，又使步长得到合理的调整。

$$\begin{aligned} &\lambda=\lambda\psi, \quad \psi>1, \quad \text{当 } \Delta E<0 \\ &\lambda=\lambda\beta, \quad \beta<1, \quad \text{当 } \Delta E>0 \end{aligned} \tag{3-36}$$

这里 ψ、β 为常数，$\Delta E=E(n_0)-E(n_0-1)$。

当然变步长也可用其它方法进行。很多文章都讨论了这种步长改变和选择的方法。确定了步长以后，可得到其迭代公式：

$$W(n_0+1)=W(n_0)+\lambda(n_0)d(n_0) \tag{3-37}$$

由于$d(n_0)$是第n_0次迭代时刻误差函数E对连接权W的负梯度，则变步长算法可对照式(3-20)写成

$$w_{ij}(n_0+1)=w_{ij}(n_0)+\lambda(n_0)b_i\,d_j \tag{3-38}$$

(2)加动量项。

为了加速收敛和防止振荡，在许多文献中都建议引入一个动量因子 a

$$W(n_0+1)=W(n_0)+\lambda(n_0)d(n_0)+a\Delta W(n_0) \tag{3-39}$$

其中第三项是记忆上一时刻权的修改方向，而在时刻 n_0 的修改方向为(n_0-1)时刻的方向与 n_0 时刻方向的组合。将式(3-39)改写为

$$W(n_0+1)=W(n_0)+\lambda(n_0)[d(n_0)+\frac{a}{\lambda(n_0)}\Delta W(n_0)]$$

$$=W(n_0)+\lambda(n_0)[d(n_0)+\frac{a\lambda(n_0-1)}{\lambda(n_0)}d(n_0-1)]$$

上式中动量因子 $0<a<1$。建议在 $\lambda(n_0)$进行调整时，碰到 $\Delta E>0$，λ 要减小时，让 $a=0$，然后调节到 λ 增加时，使 a 恢复。

还有很多改进的 BP 算法，为叙述简洁，不一一列出。

3.2.3.5 BP 网络的设计考虑

(1)输入与输出层的设计。

输入的神经单元可以根据需要求解的问题和数据表示的方式而定。如果输入的是模拟信号波形，那么输入层可以根据波形的采样点数决定输入单元的维数，也可以用一个单元输入，这时输入样本为采样的时间序列。如果输入为图像，则输入单元可以为图像的像素，也可以是经过处理后的图像特征。

输出层维数根据使用者的要求来确定。如果 BP 网络用作分类器，其类别为 m 个，有两种方法确定输出层神经元个数：

①输出层有 m 个神经元，其训练样本集中 X^{p_1} 属于第 j 类，要求其输出为

$$y=(0,0,\cdots 0,\underset{j}{1},0,0,\cdots,0)^T$$

即第 j 个神经元输出为 1，其余输出均为 0。

②输出层有 $\log_2 m$ 个神经元。这种方式是根据类别进行编码。

(2)隐层的数目

1989 年 Robert Hecht－Nielson 证明了对于任何在闭区间内的一个连续函数都可以用一个隐层的 BP 网络来逼近，因而一个三层的 BP 网络可以完成任意的 n 维到 m 维的映射。

(3)隐单元数目的选择。

对于隐单元数的选择是一个十分复杂的问题，往往根据设计得的经验和试验来确定，因而没有一个很好的解析式来表示。可以说隐单元数与问题的要求，输入输出单元的多少都有直接的关系。

对于用作分类的 BP 网络，可以参照感知器中间隐单元数的公式(2-16)和(2-18)，但由于 BP 网络隐单元的输入输出函数为非线性函数，因此比感知器要求的隐单元数目少。隐单元数太多会导致学习时间过长，误差也不一定最佳；隐单元数太少，容错性差，不能识别以前没有看到的样本。因此存在一个最佳的隐单元数，提供以下几个公式供参考。

①
$$k<\sum_{i=0}^{n} C_{n_1}^{i} \tag{3-40}$$

式中 k 为样本数，n_1 为隐单元数，n 为输入单元数。如果 $i>n_1$，$C_{n_1}^{i}=0$

②
$$n_1=\sqrt{n+m}+a \tag{3-41}$$

其中 m 为输出神经元数，n 为输入神经元数，a 为 1～10 之间的常数。

③
$$n_1=\log_2 n \tag{3-42}$$

n 为输入神经元数。

对于用于数据压缩情况下的 BP 网络，隐单元与输入单元的比为其数据的压缩比，它可参照式(3-42)来考虑。

还有一种考虑是使隐单元的数目可变，或初始放入足够多的隐单元，然后把学习后那些不起作用的隐单元逐步去掉，一直减小到不可收缩为止。也可在初始放入比较少的隐单元，学习一定次数后，不成功再增加隐单元数，一直达到比较合理的隐单元数为止。

(4)初始值的选取。

由于系统是非线性的，初始值对于学习是否达到局部最小和是否能收敛的关系很大。一个重要的要求是希望初始权在输入累加时使每个神经元的状态值接近于零，这样可保证一开始不落到那些平坦区上。权一般取随机数，而且权的值要比较小，这样可以保证每个神经元一开始都在它们的转换函数变化最大的地方进行。

对于输入样本同样希望能够归一，使那些比较大的输入仍落在神经元的转换函数梯度大的那些地方。

3.3 自组织竞争学习

感知和认知的基本问题是研究人是如何发现、学习和认知环境的不变特性，当对环境信息的编码自发地在神经网络与环境的交互中产生时，就认为神经网络是在进行自组织。竞争学习是指同一层次神经元相互之间的竞争，竞争胜利的神经元修改与其相连的连接权值。自组织竞争学习是一种无监督学习。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，并将其划分到其相应的模式类中或对外界环境进行感知。

在自组织网络中，权的学习是用 Hebb 学习律或竞争学习律得到的，若两个神经元输出兴奋(神经元输出大于零)，则它们之间的联接权增加，反之减少，按其学习公式来分，大致有以下几种学习规则：

(1)信号 Hebb 学习规则。

$$\dot{w}_{ij}=-w_{ij}+s_i(x_i)s'_j(h_j) \tag{3-43}$$

w_{ij}代表第 i 个输入神经元与第 j 个输出神经元之间的连接权。如果 $w_{ij}>0$ 为兴奋连接权，$w_{ij}<0$ 为抑制性权，$s_i(x_i)$与 $s'_j(h_j)$为两个不同层上的神经元输出，$s_i(x_i)\in[0,1]$，$s'_j(h_j)\in[0,1]$，它们为 0 与 1 之间的模拟量。如果两个神经元都兴奋，那么 w_{ij}就会增加，否则 w_{ij}会减少，甚至达到遗忘。例如 $s'_j(h_j)=0$ 时，$\dot{w}_{ij}=-w_{ij}$，此时 w_{ij}的值会减少，直到零为止。

(2)竞争学习律。

$$\dot{w}_{ij}=s'_j(h_j)[s_i(x_i)-w_{ij}] \qquad \text{对所有的 } i \tag{3-44}$$

如果 $s'_j(h_j)$是一个 Sigmoid 函数，$s'_j\in[0,1]$，因此如果在某一时刻 i，s'_j“赢”了，$s'_j(h_j)=1$，那么与第 j 个神经元相连的所有的权都发生变化，w_{ij}的改变是使 $s_i(x_i)\approx w_{ij}$，$i=1,2,\cdots,n$。如果 $s_j(h_j)$“输”了，即 $s_j(h_j)=0$，则 $\dot{w}_{ij}=0$，权 w_{ij}不进行调整。考虑一个特殊情况：$s_i(x_i)=x_i$，那么式(3-44)就变为

$$\dot{w}_{ij} = \dot{s}_j'(h_j)(x_i - w_{ij})$$

(3)微分 Hebb 学习律

$$\dot{w}_{ij} = -w_{ij} + s_i(x_i)s_j'(h_j) + \dot{s}(x_i)\dot{s}'_j(h_j) \tag{3-45}$$

比较式(3-43)和式(3-45)，微分的 Hebb 规则是多了一个与状态变化率有关的量 $\dot{s}_i(x_i) \cdot \dot{s}_j'(h_j)$，因为 $\dot{s}_i$ 与 $\dot{s}_j'$ 有正有负，它改变了权 w_{ij} 变化的情况，$\dot{s}_i$ 与 $\dot{s}_j'$ 大，权的改变也大，当 $\dot{s}_i=0$ 或 $\dot{s}_j'=0$ 时，则式(3-45)与式(3-43)相同。

(4)微分竞争学习规律。

$$\dot{w}_{ij} = \dot{s}_j'(h_j)[s_i(x_i) - w_{ij}] \tag{3-46}$$

微分竞争学习只与输出的变化有关，当 $\dot{s}_j'(h_j)$ 达到平坦区，$\dot{s}_j'(h_j)=0$，w_{ij} 也不会改变了。

以上讨论的四个学习规律是在不同的自组织网络中运行的，它们没有外加教师，因此这些学习是有一定的随机性。当网络达到稳定时，能完成一定的功能。下面介绍几种常用的自组织竞争网络的学习算法。

3.3.1 竞争学习

竞争学习网络的第一个层次是输入层次，它接收样本输入。第二个层次是竞争层次，它对输入样本进行分类。这两个层次的神经元之间进行全互连连接。如图 3-8 所示。即第一层的每个神经元与第二层的每个神经元均相连。为简单起见，限定权值在 0 和 1 之间，并且相应于某个神经元 j，所有连接权之和为 1，即

$$\sum_i w_{ij} = 1 \tag{3-47}$$

输入样本为二值为量，其各个元素取值 0 或 1。在初始情况下，w_{ij} 的值随机地设置并满足式(3-47)。设 s_j 为竞争层第 j 个神经元的状态

$$s_j = \sum_i w_{ij}x_i$$

其中 w_{ij} 为输入层神经元 i 到竞争神经元 j 之间的连接权值，x_i 为输入样本向量的第 i 个元素。在 WTA(Winner Tales All)机制中，竞争层上具有最大的权和的神经元 j 便赢得竞争胜利，其输出为

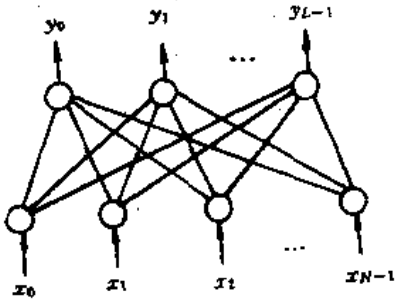


图 3-8 竞争学习的基本结构

$$a_j = \begin{cases} 1, & s_j > s_i, \text{对所有的 } i \neq j \\ 0, & \text{其它} \end{cases} \tag{3-48}$$

竞争后的权值依下式修正：

$$\Delta w_{ij} = g\left(\frac{x_i}{m} - w_{ij}\right), \text{对所有的 } i \tag{3-49}$$

其中 g 为学习参数($0<g\ll1$)，m 为输入层上输入值为 1 的神经元个数，x_i 为输入样本向量的第 i 个元素。

在式(3-49)中，g 是一个较小的正数，一般取 0.01～0.3，式中的 x_i/m 项表明当 x_i 为 1 时，权值增加，而当 $x_i=0$ 时，权值减少。由于所有权值之和为 1，故当第 i 个权值增加或减少时，对应其它权值就可能减少或增加。式(3-49)中的第二项则保证整个权值的调整能满足式

(3-47)。即所有权值的调整量之和为 0。也就是：

$$\sum_i \Delta w_{ij} = g\left(\frac{1}{m}\sum_i x_i - \sum_i w_{ij}\right)$$

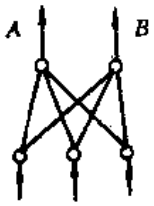
$$= g(1-1) = 0$$

例：简单模式的分类。

从这个例子中，我们可以看出如何用竞争算法将简单的输入样本分成两类。样本为有三个元素的二值向量如图 3-9(*a*)。所采用的二层结构如图 3-9(*b*)所示。图 3-9(*c*)给出了两个样本之间的类似程度，即两个输入向量之间相对应元素不相同的数目，也称之为汉明距离。图 3-9(*d*)给出了训练后网络所形成的分类。每个模式类中有两个样本，同一模式类中的样本只有一个向量元素不同，也就是说竞争算法将类似的样本分在同一类中。图 3-9(*e*)对这个分类问题给出了一个简单的几何说明。训练后的权向量，即图 3-9(*e*)中的向量 W_A 和 W_B，分别表示神经元 A 和神经元 B 与输入层神经元之间的互连接权值。训练集向量用立方体的顶点来表示。神经元 A 的权向量 W_A 与模式 A 的两个向量相接近，也就是说它们与神经元 A 的权向量之间的夹角最小，或类 A 中的模式向量到权向量 W_A 的投影的模最大，而到权向量 W_B 的投影的模较小。如果用 S_A，S_B 分别表示类 A 中模式向量到权向量 W_A 和 W_B 的投影，则有 $S_A > S_B$。这样，当属于模式 A 的向量输入时，神经元 A 就呈活跃状态，将输入向量归为类 A 中。对于神经元 B，以及类 B 的输入向量，情况完全与上述类似。

$(101) = P_1$
$(100) = P_2$
$(010) = P_3$
$(011) = P_4$

(*a*)



(*b*)

汉明距离

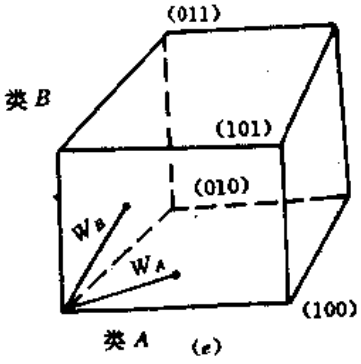
	P_1	P_2	P_3	P_4
P_1	0	1	3	2
P_2	1	0	2	3
P_3	3	2	0	1
P_4	2	3	1	0

(*c*)

$P_1 = (101)$
$P_2 = (100)$
} 类 A

$P_3 = (010)$
$P_4 = (011)$
} 类 B

(*d*)



(*e*)

图 3-9　简单的模式分类

3.3.2 自组织映射学习

自组织映射模型是 Teuvo Kohonen 于 1981 年提出的。这种模型可以在一维或二维的处理单元阵列上形成输入信号分布拓扑图。自组织映射模型(Kohonen 网络)的结构和工作原理已在本书第 2.4 中详细地讨论过,其结构图如图 3-10 所示。

Kohonen 的自组织映射模型系统的基本结构由四个部分组成:

(1)一个处理单元阵列。这个处理单元阵列从事件空间中接收相关的输入,并且形成这些输入信号的简单的辨别函数。

(2)一种比较机制。这种机制比较辨别函数并选择一个具有最大函数输出值的处理单元。

(3)某种局部互连作用。这种局部互连作用同时激励被选择的处理单元及其最相邻的邻接处理单元。

(4)一个自适应过程。这个自适应过程修正被激励的处理单元的参数,以增加其相应于特定输入的辨别函数输出值。

对于图 3-10 那样的 Kohonen 网络,输入神经元数与问题的要求有关。设输入为 n 个神经元,输出为一个二维平面,希望在输出平面上的神经元数足够多,设为 n_1 个,$n_1 \gg n$,定义一个 d 函数,表示输入样本矢量 X 与权 $W_j(j=1,2,\cdots,n_1)$ 匹配的程度,用欧氏距离表示:

$$d(X,W_j) = \| X - W_j \| \tag{3-50}$$

如果在输出层中有一个神经元与输入 X 的匹配最好,记为 c,则

$$d(X,W_c) = \min_j d(X,W_j)$$

W_c 所对应的输出为

$$y_c = \max_j y_j$$

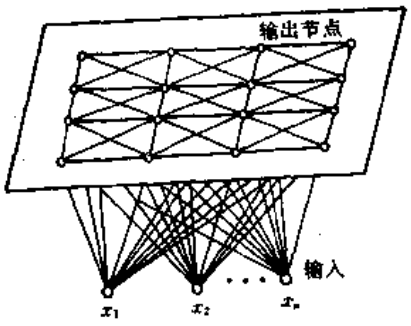


图 3-10 二维排列

而权的修正则对 W_c 和 $y_c \in N_c$ 中的 W_j 进行,具体算法为

(1)初始权 $W_j = W_j(0)$ 为一个小的随机量。

(2)在样本 $X^1, X^2, \cdots, X^p$ 中,任取一个样本作为 Kohonen 网络的输入。

(3)计算 $d = \| X^p - W_j \| = \sum_{i=1}^{n} (x_i^p - w_{ij})^2$,$j=1,2,\cdots,n_1$,取其中最小的 d,对应于 y_c 最大,作为竞争得胜的神经元。

(4)对权 w_{ij} 进行修正

$$\begin{cases} W_j(t+1) = W_j(t) + a_0(\lambda, r_0)(X - W_j(t)), & j \in N_c(t) \\ W_j(t+1) = W_j(t), & j \notin N_c(t) \end{cases} \tag{3-51}$$

修正的区域 $N_c(t)$,一开始很大,约为 $\frac{1}{2}$ 的输出平面,中心点为 y_c,形状可用正方形或六角形,然后每次迭代都按下式的形式减小。

$$N_c(t) = A_1 + A_2 e^{-t/c_2}$$

其中 C_2,A_1 和 A_2 均为常数。

(5)$a_0(t,r_0)$为权调整系数,r_0表示"赢"的那个神经元所在的位置,在其周围其它神经元的位置用半径r表示。$a_0(t,r_0)$可以采用以下各式:

$$a_0(r)=\begin{cases}1, & |r-r_0|<R' \\ -b, & R'<|r-r_0|<R \\ 0, & |r-r_0|<R'\end{cases} \tag{3-52}$$

$$a_0(r)=\begin{cases}1, & |r-r_0|\leqslant R \\ 0, & |r-r_0|>R\end{cases} \tag{3-53}$$

$$a_0=0.9\left(1-\frac{t}{1000}\right)$$

$$\text{或}\ a_0(t)=Ae^{-t/\tau_1} \tag{3-54}$$

(6)回到步骤(2),反复(2)、(3)、(4)、(5),直到在输出神经元平面上的兴奋神经元与输入样本稳定对应为止。

从上面讨论的Kohonen网络的公式和算法来看,Kohonen网络有如下几个特点:

①网络中的权是输入样本的记忆。如果当样本X^t输入时,y_i最大,那么学习后的网络y_i可以看作X^t的代表。

②比较相近的输入样本激励网络时,它们在输出二维平面上的位置也比较接近。

例:TSP问题。

在第2章中,利用Hopfield网络来解TSP问题,权的设计是采用最优化的方法得到的。在Kohonen网络中,利用自组织学习方法得到在输出神经元空间中的拓扑映射。如果把输入样本C看作是一个城市的集合,其中每一个城市是定义在二维平面上的,如同地图上表示城市的位置坐标一样,在集合C中每个样本用$C_j(x_j,y_j)$来表示。因此这种网络的输入为两个神经元,表示城市的x,y坐标,这种网络的输出为100个以上的神经元,组成了一个平面。网络中的连接权初始为一随机数,按本书前面的算法进行学习。其邻域N_c一开始取得很大,而随时间增加,$a(t,r)$也随之改变。由于Kohonen网络中的映照特性和聚类特性,使得在地图上相近的城市,在输出平面中的位置也比较相近,把相近的点连成链状。这个链无疑是按照城市远近排列的,于是,这个链就是TSP问题的解了,如图3-11所示。

图3-11 TSP问题的解

3.3.3 二值自适应共振理论网络的学习

人类的大脑能够对来自外界的新信息加以记忆,并且对新信息的记忆不会影响已记忆的信息。然而,一般人工神经网络却不能同时很好地解决稳定性和可塑性问题。如一个完全学习好的BP网络,当它学习一个新的模式时,将使已有的连接权打乱,导致已学习的模式信息消失。要使已学习模式信息不消失,须将原来的学习模式连同这个新模式一起重新训练网络。

二值自适应共振理论(简称ART1)网络是由(Carpenter和Grossberg于1986年提出的一种两层网络。这个网络是建立在Grossberg的ART原始概念之上的。ART1网络用作最邻

近分类器，能存储任意数目的二值模式 $A_k=(a_1^k,\cdots,a_n^k)$，$k=1,2,\cdots,m$。图 3-12 为其拓扑结构。其中 L_A 层的 n 个单元对应于 A_k 的 n 个分量，而 L_B 层的 p 个单元对应于引发模式的 p 个分量。ART1 网络用竞争学习规则在线学习，可以按连续时间或离散时间方式运行。学习分快速学习（对应离散时间）和慢速学习（对应于连续时间），这里仅对快速学习方法加以介绍。

ART1 网络把外界送来的一输入模式 A_k 分类到已存储的与之最相似的模式所在的类别中，已存放存储模式的类别由 L_B 层的某个单元表示。如果这个输入模式与任何一个已存储模式不匹配，那么通过存储该输入模式而建立一个新的类别，这个存储的新的模式作为对应新类别的典型模式；一旦一个已存储的模式（该模式在规定的警戒线以内匹配于这个模式）被找到，这个类别的典型模式向着该输入模式更相似的方向调整。这就是说，新的模式建立新的类别，而不影响已存储的模式（除非它们十分接近）。网络将各类别的典型模式存储于由下向上和由上向下的连接权上，利用 L_A 层与 L_B 层单元之间的反馈连接实现输入模式与已存储模式间恰当匹配上的共振。L_B 层单元利用中心兴奋和邻域抑制竞争机制确定胜者，以表示输入模式的恰当分类，网络边学习边运行，其运行过程描述如下：

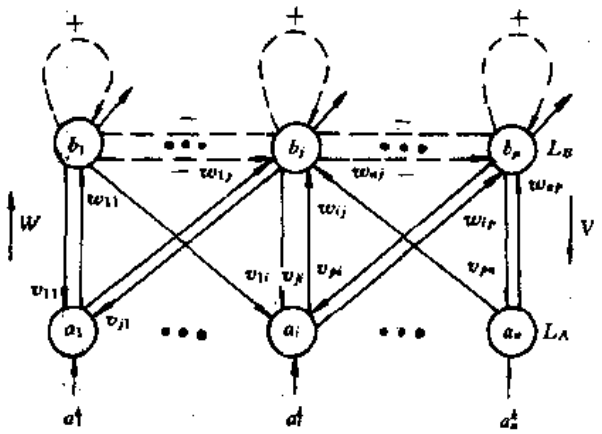


图 3-12　ART1 网络的拓扑结构

（1）初始化。为保证能用恰当的方式访问已存储的模式，Carpenter 和 Grossberg 指出：网络所有由下向上初始连接权 w_{ij} 满足下面的“直接访问不等式”

$$0<w_{ij}<\frac{L}{L-1+n}$$

这里 L 为记录长度。由于 w_{ij} 太大可能使网络将所有 L_B 层单元分配给某个输入模式，L 典型的取值为 2。同时，所有由上向下的初始连接权 v_{ji} 须满足“模板学习不等式”

$$\frac{1}{2}<v_{ji}\leqslant 1$$

由于 v_{ji} 太小会使 L_A 层不出现匹配，从而导致网络学习失败。因此，一般 v_{ji} 取为 1。

警戒参数 ρ 可在 0 和 1 之间取值，这取决于可接受的当前输入与存储模式间的错误匹配度。ρ 较低，模式类别数少；ρ 较大，分类精细，模式类别数多，在学习过程中，ρ 可按要求变化，开始时取较小的 ρ 以作粗略分类，然后逐渐增大 ρ，最终形成精细的分类。

（2）将一输入模式 $A_k=(a_1^k,a_2^k,\cdots,a_n^k)$ 送到 L_A 层，并通过连接权 w_{ij} 送到 L_B 层的每个单元，产生各单元的激活值

$$b_j=\sum_{i=1}^{n}w_{ij}a_i$$

式中　$j=1,2,\cdots,p$ 且 $a_i=a_i^k(i=1,2,\cdots,n)$。

（3）L_B 层单元通过中心兴奋邻域抑制方法相互竞争，产生出胜者（如单元 c），有

$$b_c=\max_{j=1}^{p}\left[\sum_{i=1}^{n}w_{ij}a_i\right]$$

且令 $b_c=1$

（4）胜者（单元 c）通过由上向下连接权 v_{ci} 向 L_A 层发送信号，并建立 L_A 层单元新的激活值

向量 X，其分量

$$x_i = v_{ci} a_i^k$$

式中 $i=1,2,\cdots,n$。

(5)计算输入模式 A_k 与这个新激活值向量 X 间的匹配度

$$s = \frac{|X|}{|A_k|}$$

这里 $|X|$ 和 $|A_k|$ 分别为向量 X 和 A_k 中元素为 1 的个数。

(6)如果这两个向量间的差别超过警戒参数 ρ，即 $s<\rho$，那么，这个竞争获胜的单元 c 并不代表 A_k 的恰当分类。此时，将单元 c 从已获胜单元集合中(已有类别)消去，并作下面判断：

①如果从已获胜单元集合中消去单元 c 后，仍有其它单元存在，则转向步骤(2)，从剩下的集合单元中搜集 A_k 的恰当分类；

②如果从已获胜单元集合中消去单元 c 后，无其它单元存在，则从 L_B 中任选一还没有用来表示已学习模式类别的单元，作为 A_k 的恰当类别。

(7)如果两向量间的差别不超过警戒线参数 ρ，即 $s\geqslant\rho$，那么单元 c 表示 A_k 的恰当分类。

(8)一旦表示 A_k 恰当类别的单元(如单元 c)被找到，单元 c 相关的连接权向量 W_c 和 V_c (表征 A_k 类别的典型模式)将向着与 A_k 更相似的方向调整，其学习方程为

$$w_{ic} = \frac{Lx_i}{L-1+\sum_{j=1}^{n} x_j}$$

式中 $i=1,2,\cdots,n$ 并且 $V_{ci}=x_i, i=1,2,\cdots,n$。

从上面可以看出，表示某输入模式恰当类别的 L_B 单元的确定是通过由下向上在 L_B 层单元的竞争和由上向下在 L_A 层的匹配来完成的，这两个阶段构成了共振过程。输入模式的存储是通过将所在类别的典型模式(W_c 和 V_c)向该输入模式靠近的方式来完成的。

Carpenter 和 Grossberg 已证明了 ART1 网络的有关定理，其中两个最重要的结论是：

①当网络学习稳定以后，如果一个已学习过的模式提供给网络，那么该模式将正确地激活表示所在类别的 L_B 层神经元，并且共振状态出现。这种“直接访问”特性意味着网络能迅速访问已存储模式。

②学习过程是稳定的。这就是说，网络对任何一个输入模式序列进行有限次学习后，能够产生一组稳定的连接权向量。当这个输入模式序列重复地提供给网络时，不会引起网络的连接权无休止地循环调节。此外，网络对任何一个输入模式，试图将它进行已有类别的分类，如果分类不成功，将它归入一个新的类别。不管分类成功与否，都将它存储于网络之中。这个过程使网络边学习边回想，实现了在线学习功能。进一步地，网络通过选择恰当的警戒参数 ρ，可对任意数目的输入模式进行要求的分类。

ART1 网络的这种快速学习方法由于直接的提供给网络和输入模式一次性存储于连接权上，因而网络所形成的输入模式类别并不是输入模式统计特征的表示。ART1 的慢速学习却只允许每次输入模式中一小部分特征“溶解”到连接权上。通过将输入模式集反复地提供给网络，从而使输入模式的最显著的统计特征最终全部地存储在网络的连接权上。因此，当输入模式集合受到噪声污染时，宜采用慢速学习。

第 4 章　神经网络在机械制造业中的应用

到目前为止，我们已经介绍了神经网络的基本网络结构及其学习机制。由于神经元网络具有自组织，自学习和并行分布式信息处理的特点，其应用范围日益扩大。本章介绍神经网络在机械制造业中的应用。

在高级制造系统中，人们遇到了越来越多，越来越复杂的决策问题。自 50 年代起人工智能问世以来，人工智能极大地推动了制造业的智能化和一体化发展。在过去的几十年中，基于知识库的专家系统被作为最有开发前途的应用技术。然而，专家系统对于不断变化的，复杂的和开放的机械制造系统环境不十分有效，另一方面，神经网络的鲁棒性、自适应性以及联想能力在机械制造业显示了很多优势，具有潜在的应用前景。尤其是将专家系统和神经网络技术结合在一起，可以优势互补，改善专家系统的灵活性，给用户提供一个单一的、灵活的人机界面。

4.1　神经网络与成组技术

成组技术(Group Technology)是将设计和制造工艺上相似的各种零件和产品归并成组，以便提高生产效率，在采用多品种小批量生产方式中，从设计到加工装配的每个零件往往被看作是孤立的，但是，实际上从零件的设计或制造工艺来看，可以将相似设计和制造工艺的零件分组而形成零件族。这样，就可以通过更加有效的设计合理化和信息的检索，以及制造的标准化和合理化来提高生产率。因此，应用成组技术，可以为多品种小批量生产带来大量的经济效益。

成组技术的原理可用于许多领域，包括机械制造、零件设计、生产规划、制造加工单元设计等等。通过应用成组技术，可以有效地减少装配时间，产品的前置时间以及生产现场库存等。我们知道，与成组技术相关的两个工程问题为：零件分类和零件族形成。如果不考虑分类和零件族形成的方法，核心的问题是如何保护一致性。传统的视检方法在很大程度上依赖于人的主观判断。为了解决这个问题，一些研究人员应用神经网络的自组织，自学习性能，给神经网络提供由成组技术构造出的一致性准则供其学习，实例说明神经网络可以有效地处理大规模零件族形成问题。

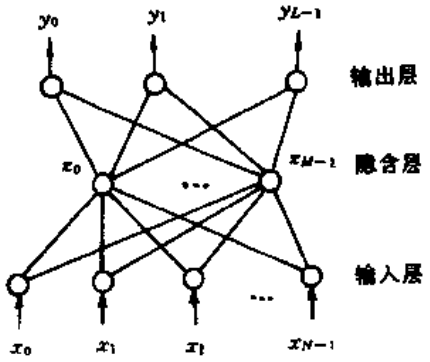


图 4-1　零件族形成神经网络

Kaparthi 和 Suresh(1991)提出了用神经网络零件形状进行分类和对旋转部件编码。他们采用三层 BP 网络，如图 3-6 所示。网络的输入是零件图的二值映射(二值向量)，网络样本输出是与零件几何形状相对应的奥匹兹(Opitz)编码。用反向传播算法训练 BP 网络，得到对零件几何形状的恰当的分类。还有很多人研究用神经网络解决零件族形成问题。在他们提供的方法

中，一个三层前馈连接神经网用反向传播算法进行训练。其网络结构图见图 4-1。图中输入向量 $X=(x_1,x_2,\cdots x_n)^T$ 是零件的特征向量，向量 X 的第 i 个分量 x_i 表示零件的第 i 个特征；输出向量 $Y=(y_1,y_2,\cdots,y_m)^T$ 是零件的类别向量。若零件属于第 j 类 $1\leqslant j\leqslant m$，则 Y 向量的第 j 个元素 $y_j=1$，其余元素均为零（$y_e=0,e\neq j,1\leqslant e\leqslant m$）。对神经网的训练可以看作一个教师示教过程，教师按照零件的特征来描述这个零件，并且告诉网络该零件属于哪一个零件族。基于误差反向传播不断地修正网络的连接权值，训练后的网络通过一系列的连接权存储了大量的零件分类规则。网络的训练按第 3 章 BP 算法进行。

零件分类和零件族形成是密切相关的。通常可以用同一方法解决这两个问题。如果分类和形成存在于同一过程而不是两个互不相关的过程时，一致性问题可以有效地描述出来，Kao 和 Moon(1991)提出了在分类过程中利用神经网络的学习能力自动进行零件族形成的过程，这个过程分为四个阶段：(1)播种阶段；(2)映射阶段；(3)训练阶段；(4)执行阶段。

在播种阶段，从零件库中选择 3—5 个明显不同的零件作为种子零件（零件族代表）；在映射阶段，对每一个零件进行编码，神经网络的拓扑结构也在这一阶段形成。输入节点数等于分类编码系统中所含的特征数目，输出节点数目为零件族数目；在训练阶段，一系列训练样本输送给网络，网络在样本集的示教下用 BP 算法训练；在执行阶段，将任一被描述的零件特征向量提供给网络，如果零件的特征与任一现有的零件族产生的特征相似，网络就输出属于同一零件族代码；如果没有任意零件族与之相同，神经网输出将不显示任一现有的零件族代码。

4.2 神经网络与工程设计

神经网络在工程设计中的应用可分为两个分支：产品设计和机械制造系统的设计。产品设计可以看作功能空间到物理结构空间的映射。一个有经验的设计人员通常十分了解结构与相应的特定功能之间的关系。在他的记忆中存储了大量的物理装置的表述。根据产品功能与结构之间的联系，设计者可以有选择地恢复这些设计（即设计解）。神经网络非常适合给人的联想记忆建模，所以设计师们对在产品设计中应用神经网络很感兴趣。

下面介绍神经网络(FAM 网)在椅子设计中的应用实例。

现在我们的任务是设计一把椅子。对椅子的功能要求为十分舒适，限制条件为：在公共场所使用。

根据要求和限定条件，我们怎样才能找出设计解——既舒适又能在公共场所使用的椅子。考虑到椅子设计中的限制因素（例如尺寸、价格、几何尺寸等等）和与功能（很舒服）相关的因素，我们可以把设计要求进一步分解，给出知识表达形式。

4.2.1 知识表达

文献[12]提出个 11 个设计约束条件和 6 个功能要求，并将其用于各种类型椅子的设计。这六个功能要求是：能调节椅子(FR_1)，能移动(FR_2)，能折叠(FR_3)，牢固(FR_4)，可堆放(FR_5)和舒服(FR_6)。11 个设计约束条件为：价格(C_1)，尺寸(C_2)，重量(C_3)，餐厅用(C_4)，办公用(C_5)，休息用(C_6)，家里用(C_7)，教室用(C_8)，公共场所用(C_9)，打字用(C_{10})，美观(C_{11})。定义 FR_s 为功能集合，C_s 为设计约束集合。

$$FR_s=\{FR_1,FR_2,FR_3,FR_4,FR_5,FR_6\}$$

$$C_s = \{C_1, C_2, \cdots, C_9, C_{10}, C_{11}\}$$

其中 $FR_i(i=1,2,\cdots,6)$为硬函数，即

$$FR_i = \begin{cases} 1, & \text{具有第 } i \text{ 项功能} \\ 0, & \text{没有第 } i \text{ 项功能} \end{cases} \tag{4-1}$$

而 $C_i(i=1,2,\cdots,11)$取(0,1)区间的连续值，表示满足第 i 个约束条件的可能性大小(隶属度函数)。例如，$C_5=0.5$ 表示办公用的可能性为50%；若 $C_5=1$ 表示一定在办公时用。假定数据库中存放着11种不同的设计解(不同椅子的知识描述)。例如数据库的第二种椅子的知识表达为

$$f_r(Chair_2) = \{1.0/FR_1, 1.0/FR_2, 1.0/FR_6\}$$

$$C_s(chair_2) = \{1.0/C_5, 0.7/C_6, 0.3/C_7, 1.0/C_{11}\}$$

把以上表达式用语言来描述为

第二种椅子非常舒适并具有可调性和可移动性，该椅子用于办公的可能性比家用可能性要大，它也可以用于休息，而且美观。

4.2.2 求解策略

模糊联想记忆网络(FAM)被用来恢复设计解。FAM网只能存储一个模糊模式对。即每一个模式对(一种设计)就对应于一个独立的FAM网络，那么 n 种可能的设计就对应于 n 个FAM网络。文章提出了包括激励和求解两个阶段的求解算法。其基本思想是：在激励阶段，将输入向量(由设计的功能和约束条件所形成的特征向量)输送给 n 个FAM网络。输入向量在网络中进行正向和反向传播，则每一个网络均产生一个向量。在求解阶段，如果某网络产生的向量最接近于输入向量，那么这个网络就被选中，即该网络表征的设计被接纳。

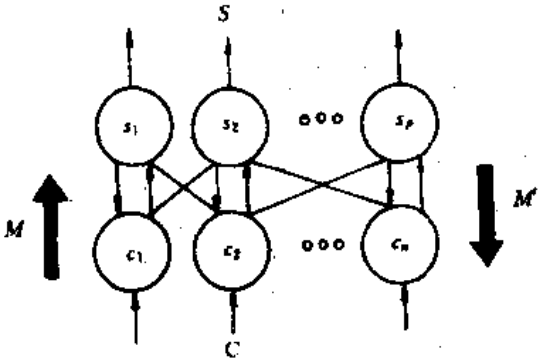


图4-2 FAM拓扑图

由以上讨论可知，对于椅子设计问题，用功能集合 FR_i 和约束条件集合 C_i 来表示。为了简单化，把这两个集合合并在一起作为椅子的特征，并把这些特征所形成的特征向量作为FAM网络的输入向量，即FAM网络输入层有17个节点。例如，基于功能要求和约束条件，第二种椅子(*chair2*)可用下面的特征向量来描述：

$$C_2 = (1.0 \quad 1.0 \quad 0 \quad 0 \quad 0 \quad 1.0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1.0 \quad 0.7 \quad 0.3 \quad 0 \quad 0 \quad 0 \quad 1.0)$$

第一个元素对应于 FR_1(可调节)，第二个元素对应于 FR_2(可移动)……等等。

FAM网络的输出层有11个节点，每一个节点对应一种椅子。例如，第二种椅子可表示为

$$S_2 = (0 \quad 1.0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$$

4.2.3 算法

设第 k 个FAM网络的拓扑图如图4-2所示，$k=1,2,\cdots,11$。

假设FAM网络的连接权已用模糊赫布型规则获得，即

$$w_{ij} = \min(c_i^k, s_j^k) \qquad k = 1,2,\cdots,11$$

也就是说，第 k 种椅子的设计规则可以用第 k 个 FAM 网络来实现。图中 $n=17$，$p=11$。

现在把输入向量 $R=(r_1,r_2,\cdots,r_n)$（该向量表达了对椅子的要求条件）提供给所有的网络，则每一个网络产生一个输出向量 s_k，然后 s_k 又反向传播产生模式向量 $Q_k(k=1,2,\cdots,11)$，用下式来度是模式向量 Q_k 与输入向量 R 的相似程度。

$$s(R,Q_k)=\sum_{j=1}^{n}q_{kj}r_j-|(\sum_{j=1}^{n}(r_j-q_{kj})^2)^{1/2}| \tag{4-2}$$

此时，获得最大 s 值的网络作为胜者（若第 i 个网络获胜），那么第 i 个网络的输出就是所获得的设计解。

从以上的讨论可知，神经网络的联想记忆完成了功能空间到分类空间的转换。如果注意到机械设计专家阅读二维机械图时的研究行为，就知道人脑的知识加工过程是将机械图的几何形式转换成机械部件的功能，有些研究人员用 ART 网络来实现这种映射。尤其是神经网络可以在不精确的、残缺的输入信号下产生正确的输出，可以有效地用于产品设计和设计数据的恢复。

4.3 神经网络与故障诊断

全自动化机械生产是现代化生产的趋势。实现这一目标的关键是对制造过程、工具和机器的状态进行全面的监测。一个良好的监测系统意味着优良的产品质量，更好的规划机器的维护以及更有效的制造策略等等。监测过程通常包括信号采集、信号分析和故障识别这几个阶段。从各种传感器采集的信号通常是随机的，需要对信号进行分析并从中抽取有意义的特征量。故障识别过程实质上是模式识别过程，在这个过程中，被分析的信号用来进行机器的故障诊断。神经网络具有自学习和自组织功能，它在模式识别中的应用显示了极大的优势，被广泛地用于工况监测和故障诊断，实践证明，神经网络技术比传统的统计模式识别法、归纳推理和模糊逻辑方法更为有效。

4.3.1 神经网络在工件焊接诊断中的应用

在工件投入使用之前，首先应该检测工件的质量，及早发现工件的疵点，这样不仅可以节省大量时间和运行费用，也保证了生产运行的可靠性。

在焊接工件的焊点诊断中，用超声波来探测焊点，并通过一系列传感器采集数据，然后对数据进行分析，诊断出焊接疵点的性质及其出现疵点的原因。

常见的焊接的疵点有三种类型。其中的两类是“良性”的（渣状焊点和气泡焊点），另一种是“恶性”的，即有裂缝。有裂缝的焊点又可以分成两类：粗糙裂缝和光滑裂缝，光滑裂缝可能由缺少焊锡引起，也可能由氢引起。图 4-3 为焊接疵点分类图。

神经元网络被用来进行焊接疵点的分类。神经网络的训练样本由上述几种典型疵点的特征数据和其对应的归属类别构成。为了观察神经网络的分类能力，原始数据库被随机地分成两部分，训练数据和测试数据，训练数据作为样本示教神经网络进行分类，测试数据（从未在网络学习时出现过）用来检验学习后的神经网络的分类正确率。在学习阶段，仅仅使用原理数据库的 $\frac{1}{4}$ 对网络进行训练；在测试阶段，学习后的网络可以对测试数据库中 90% 的例子实现正确分类。

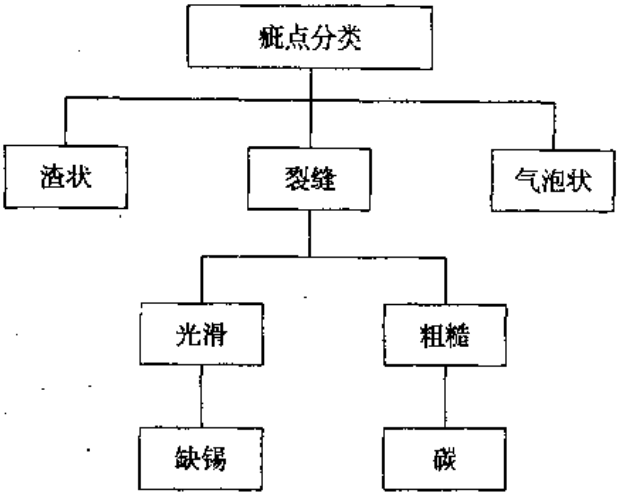


图 4-3 焊接疵点分类图

4.3.2 阀门密封圈表面疵点自动识别

在现代机械制造业中，为了取得并保持产品的高质量，不能只以开环的形式使用自动视觉检测，而应该在闭环模式中利用返回生产线上产品疵点的性质和形式的信息，采取正确的控制手段改善产品质量。

神经网络可作为自动视觉检测设备对自动生产线上的阀密封圈的表面疵点的性质进行分类。这条生产线的每年最高产量是2000万密封圈。密封圈的材料是黑橡胶，疵点大小在0.1～0.2mm 之间，生产要求要对百分之一的密封圈进行视检。产品规模，材料的颜色，疵点的大小的视检的要求都给人工视检造成困难，使得有必要采用自动视检技术。

文献[4]提出的质量改进系统有三个单元组成：自动视检设备(AVI)，过程监测计算机(PMC)和质量管理计算机(QMC)。密封圈被送到 AVI 设备进行视检，PMC 把有关密封圈质量的信息提供给 QMC，同时，PMC 把诊断信息送到 QMC，例如温度、压力和周期。利用与被观测过程状态和过程参数设置有关的密封圈生产过程的模型，QMC 计算并把反馈信息传送密封圈生产机械进行调节。

AVI 设备获取来自 CCD 摄像机的密封圈图像，这些图像经过常规图像处理算法分离出疵点的区域。对于每一个有疵点的区域，要着重提取疵点几何特征的信息。对于每一个疵点，用 23 维向量代表它的几何特征。在这 23 维向量中，前 3 个元素对应于疵点的大小，其余每一个元素表示 20 种几何特征中任一种的频率值。几何特征是指密封圈轮廓处的特征(例如弧状、线状等等)。

神经网络模式存在于 AVI(自动视检)设备中。这些神经网络用来完成提取的特征的分类。根据不同的特征向量提供的例子来训练神经网络。

文中对神经网络的模式进行了比较和试验，分别选用了自适应逻辑网络(ALN)，反向传播多层感知机(BMLP)和 Kohonen 网络。

BMLP 网络中的神经元通常具有非线性激活函数，这使得 BMLP 可以完成复杂的映射。误差信号，即网络的实际输出与理想输出之差反传到输入层，从而调节网络中神经元之间的连接权，调节按照误差梯度方向进行。训练过程由学习因子 η 和记忆常数 α 控制，η 和 α 都位于(0,1)区间。η 过大容易引起不稳定，反之，η 太小使学习过程非常缓慢，在某些场合，可以初始取较大的 η，然后减小 η 以使梯度收敛达到全局最小。记忆常数 α 用来调节权系数。一般来说，

α 增大可以加速训练过程。

一个 3 层的 BMLP 网络，包含 23 个输入节点，10 个隐单元和三个输出单元。23 个输入节点对应于疵点几何特征的 23 维特征向量，3 个输出单元对应于 3 种类型的疵点(纹状，圆斑和粗糙斑点)。所有的神经元都具有 Sigmoid 激活函数。学习因子 η 是 0.7，记忆常数 α 为 0.8。训练 50 万次达到输出误差小于 0.001。训练时间大约 20min。BMLP 的正确分类率达到 90%。

Kohonen 自组织特征映射具有一维或二维阵列，输出平面上的每一个节点代表一个分类模式的特征向量。一个特征向量的各个分量是这个节点和输入诸节点之间的联接权。输入节点提供待分类的模式。连接权在训练开始时被赋于很小的随机值。当特征平面(输出平面)进行训练时，连接权渐渐被修正以致平面上相邻的节点具有相似的特征向量。Euclidean 距离被用来度量相似性。训练中由训练集合提供模式或通过输入节直接从在线过程获取。KOHONEN 特征平面由 10×10 个节点组成，这些节点联接到 23 个输入节点以获得代表疵点的特征向量，训练中采用正方形邻域，训练迭代次数 18000 次，大约用 17min。特征平面正确分类率达 79%。为了扼要叙述起见，自适应逻辑网络 ALN 的应用情况不再详述，网络正确分类率为 78%。

BMLP 网络协调结构：显然，如果按照疵点的尺寸大小分类，在以上三种疵点(纹状、圆斑、粗糙斑)分类情况下，又可以分成子类别(大、中、小)。我们知道，一个单一神经网难以对疵点的所有尺寸大小进行分类。文中提出了用一组神经网的方法。每一个特定的神经网只在特定的尺寸下对疵点进行分类，然后这些特定神经网络的输出把带有疵点尺寸的信息送到一个上层神经网，该神经网给出这组神经网的总输出。这种协调结构可以改善分类精度。

BMLP 网络协调结构为：3 个特定神经网和一个上层(决策层)神经网。选择 BMLP 网络的原因是单 BMLP 网络的分类精度最高。3 个特定神经网的结构相同。采用某一几何尺寸下的疵点模式来训练某一特定 BMLP 网(几何尺寸大、中、小)，所有的训练数据有 180 个模式，每个特定 BMLP 网络训练误差小于 0.001 所需的迭代次数大约 50 万次。一个 BMLP 的训练时间为 8min。上层神经网有 12 个输入节点(其中 9 个节点来自 3 个特定神经网的输出，另外 3 个来自有关疵点几何尺寸的信息)。6 个隐层节点和 3 个输出节点，迭代次数达 30 万次(误差小于 0.01)，训练时间 6min。

这种协调式结构的分类正确率达到 93%。表 4-1 给出不同神经网作为分类器的特性比较。

表 4-1　不同神经网的比较

分类器	学习规则复杂度	训练时间 min	精度%
ALN	比 BMLP 复杂	~5	78
Kohonen	简单	~17	79
BMLP	简单	~20	90
协调式 BMLP	简单	~30	93

从表中看出 ALN 网的精度最低，这是由于 ALN 网的二进制逻辑决策过程对噪声输入敏感；Kohonen 网的特征是在无需人干预下完成特征向量分类，但对于非完善定义的特征向量

来说，二进制决策逻辑的效果是不好的。BMLP 网络具有连续决策逻辑具有最好的分类效果。BMLP 网还具有易于与 AVI 设备组合的商业优势，因而仅需 4s 就完成对未知特征向量的分类，当 BMLP 组合成协调系统，性能可得到进一步改善，然而，这种协调系统的代价是需要更多的训练数据而且把这些数据分成几个子类，在某些应用场合或许是困难的。表 4-1 给出的是密封圈疵点分类的正确率，并不是视检疵点的正确率，视检正确率可达到 100%。

4.3.3 加工机床的故障诊断

机械加工技术这一现代机械制造业的支柱正在迅猛发展。50 年代出现了第一代数控机床，70 年代第一台计算机控制机床(CNC)问世，现在以柔性加工系统为标志的复杂的自动化机床已投入使用。与现代化机床迅速更新换代相对照，机床或机器加工中心的维护管理技术却远远落后了。

与传统的机械加工不同的是，对于柔性加工系统来说，系统自身的可靠性和可维护性是至关重要的，任何短时的故障造成的生产中断都会给生产造成巨大损失。这就需要对加工系统部件的运行工况进行在线监测，给出故障的预报，以提醒维护人员对部件进行检修或更换，尽可能减少系统的故障停运时间。

故障诊断一般分为两类：硬故障和软故障。硬故障表示部件报废，元件损坏等；软故障是一种进行性故障，例如部件的磨损，运行状态不良。在硬故障诊断方面已经有了很多成果，人们采用专家系统排列出故障树，用二进制码表征被测部件状态的情况，“1”表示故障，“0”表示正常。然而对软故障的诊断却很棘手，这是因为“正常”状态与“故障”状态之间没有一个“门槛”值，例如一个磨损的齿轮，虽然还在照常运转，但运行工况已不佳了，需要维护人员在一定时间内更换以确保整个系统正常运行，不然，总有一天软故障成为了硬故障，造成机床停产。早期关于软故障诊断的研究限于模型参考方法，即用辨识的方法得到正常工况下的模型，得到正常工况的动态曲线，然后根据实际工况与正常工况下动态曲线之前差别判断故障的类别及故障等级。这种诊断方法鲁棒性和准确性比较差。神经元网络的一个成功的应用就是根据先前知识进行对所研究系统的学习来实现正确分类。下面介绍机床冷却系统的故障诊断。

工件加工对需启动冷却系统，冷却剂不仅可以清洁工件表面，而且可以减小刀具磨损。冷却剂是回流的，即喷嘴→贮槽→管道→喷嘴。在冷却剂回流通道中，切削加工时残留在冷却剂中的颗粒可能堵塞通道，造成冷却系统的故障。造成润滑系统故障的主要原因有：贮槽堵塞，泵出口堵塞和压力释放阀功能不正常。故障诊断系统取贮槽，泵出口和压力释放阀处压力信号作为故障诊断的特征信号。

4.3.3.1 建立故障库

为了使神经网络能诊断出故障的类别(贮槽堵塞，泵出口堵塞和释放阀不正常)和故障的级别(小故障、一般故障、严重故障、完全堵塞(损坏))，必须采集在各种情况下检测的压力信号，形成故障库，并把它们作为神经网络的训练信号。

在采集故障状态信号之前，首先获取正常状态下贮槽出口压力，泵出口压力和释放阀处压力信号，形成正常工作状态的数据库，然后对某一特定故障，用实验的方法模拟故障的级别(严重程度)采集实验数据。例如在建立贮槽堵塞故障库时，人为地把管道口堵塞 20%、60%、80%和 100%以观察并采集贮槽出口处压力检测器的输出信号。对泵出口堵塞和释放阀工作不正

常采用类似的方法获得故障库。显然，采集的信号是带有噪声的。要对信号进行预处理。另外，在将信号送到神经网络进行训练之前，要对信号进行归一化处理。因为各物理量(压力)变化的范围不同，必须经过归一化处理转化为幅值在[0,1]之间。

4.3.3.2 冷却系统故障诊断的神经网络结构

选择合适的诊断系统神经网络结构要考虑很多因素，第一，识别故障的准确性；第二，判断故障严重程度的准确性；第三，神经网络的可训练性和训练的快速性。虽然以上所谈的各点都有自身的重要性，但从故障库的建立来看，用作故障诊断的神经网是基于故障库中的输入一目标对来学习的，显然，识别故障的准确性反映了网络的性能，即诊断技术的可靠性。只要已知的故障模式不变，那么不管学习速率和困难程度，网络都不需要再训练了。因此，诊断技术的可靠性网络学习指标更为重要。同样，识别故障的原因和识别故障严重程度(水平)相比，前者更为重要。基于上述考虑，采用二级 BP 网络进行故障诊断：第一级识别故障的原因，第二级识别故障严重程度；把系统正常工况下的模式放入故障库中；每一种故障都以 20%作为间隔具有 4 个故障等级；系统能够预报新的故障，图 4-4 是二级 BP 神经网的框图。从图中看出，第一级神经网络提供系统是否正常的信息，如果不正常(有故障)时，给出故障的原因(贮槽、泵、释放阀或新故障)；当出现以上三种故障情况下，第二级判断故障的程度。

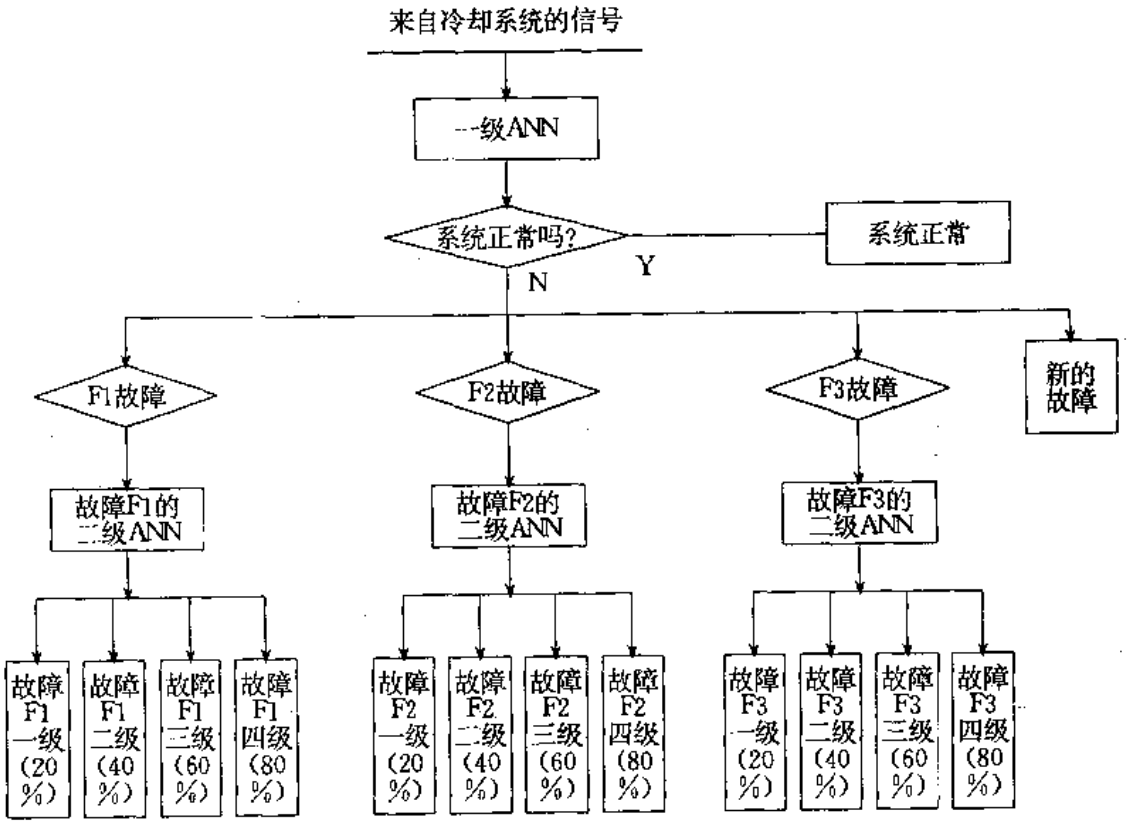


图 4-4 两级神经网络框图

4.3.3.3 网络的训练

在故障库中的每一个模式都用100个数据点表示，采样数据的周期为0.01s，总的时间长度为1s，同时，把每个模式(100个数据点)提供给BP网络的输入层，即出现在输入层的模式都与冷却系统的工作状态有关，因此冷却系统的状态可能是正常运行或任一种故障状态。系统工作状态(目标集)用一个8位二进制码来表示，或称目标模式。换句话说，我们给网络提供的是输入模式——目标模式对，当网络的训练完成后，无论何时某一输入出现在网络输入端，那么它所对应的目标模式就出现在网络的输出端。两级神经网络的训练是并行的，每一级(个)神经网的拓扑结构为：输入层100个神经元，隐层20个，输出层8个。

网络的训练是这样进行的：在第一级，依次把同一故障原因下不同故障程度的模式和对应的目标模式提供给神经网进行训练，以实现对故障原因的分类，表4-2提供了神经网的训练状态和对应的目标模式。例如泵出口堵塞故障共有4种可能的模式(堵塞20%、堵塞40%、堵塞80%，全部堵塞时泵出口处的压力信号)。把这四种模式依次送给神经网时，对应的目标模式(网络理想输出)均为"11110000"。在第二极，依次把同一故障程度不同故障原因的故障库中模式和对应的目标模式提供给神经网进行训练。很显然，第二级神经网可以识别故障程度。

表4-2 网络训练状态和目标值

ANN		输入模式	目标值
第一级	H	H_1,H_2,H_3,H_4	11001100
	F	20%,40%,60%,80%	00110011
	P	20%,40%,60%,80%	11110000
	R	20%,40%,60%,80%	00001111
第二级	20%	F,P,R	11110000
	40%	F,P,R	00110011
	60%	F,P,R	11000000
	80%	F,P,R	00001111

H——正常状态；

F——贮槽堵塞；

P——泵出口堵塞；

R——压力释放阀不正常。

前面已经提到，把正常运行状态作为故障的一种参与诊断过程。图中H_1,H_2,H_3,H_4是正常状态下的四种模式。

4.3.3.4 故障诊断

当网络训练完毕后，网络记忆了故障库中所有的模式，就具有了进行自动故障诊断能力。在训练中，故障的原因和故障程度是以两个神经网的目标值形式出现的；而在实地诊断中，网

络的输出与目标值不一致，那么，如何判断故障模式呢？一个十分简单的方法是用偏差

$$dev_i = \sum_{j=1}^{8} O_j - t_{ij} \qquad i = \begin{cases} 1,2,3,4 & \text{第一级网} \\ 1,2,3 & \text{第二级网} \end{cases}$$

式中 O_j 是网络输出层第 j 个神经元的输出，t_{ij} 是第 i 个目标模式（目标值向量）的第 j 个分量。那么，dev_i 就是被测模式相对于第 i 个标准模式的逼近程度，取 dev_i 最小值所对应的标准模式为被测模式的故障模式。

下面讨论如何识别未知模式（未在故障库出现的模式）。研究发现在未知模式和现有的标准故障模式之间存在一个偏差 dev_i 的门槛值，称为 dev_{max}（最大可容许的偏差）。对于不同的现有故障模式，这个门槛值是不同的，表 4-3 给出不同的门槛值分布。

表 4-3　实验状态及对应的偏差值

状　　态	dev_{max}
正常运行	0.002743
贮槽堵塞	0.034876
泵出口堵塞	0.033636
压力释放阀不正常	0.000169

在实验时，如果一个未知故障模式（例如混合型故障）提供给第一级网络时，计算的偏差大于表 4-3 中任一值，那么网络认为这个故障模式是一种新模式，没有必要输入给第二级网络。

BP 网络的缺点是不能对新模式进行再学习，而 ART 网络则可以进行连续学习和记忆。总之，两级 BP 网络成功地实现了机床冷却系统的故障诊断，它具有以下特点：

(1)从故障状态中分离出正常工作状态；

(2)未知故障的识别；

(3)每一种故障下的四种故障程度的诊断。

两级神经网络进行故障诊断的实验结果如表 4-4 所示。从表中看出，这种机床冷却系统故障诊断精度相当高。

表 4-4　两级神经网络故障诊断准确度

模式性质	模式数量	故障诊断数量	
		故障原因	故障
正常状态	87	2	—
新故障模式	30	1	—
贮槽堵塞	67	0	2
泵出口堵塞	69	0	4
阀不正常	92	0	5
总　计	345	3	11
准确度%	—	99.13	96.8

4.4 机器视觉和物体识别

在无人化自动装卸仓库，机械手控制中的子系统要完成物体识别、方位判断等。这个子系统要完成的任务可分为两部分。第一部分是获取和处理图像以确定物体的相关特征及属性(包括物体的重心和方位角)；第二部分是对这些特征信息进行分类。图像的处理用传统算法完成，人工神经网络在子系统中完成两个任务：对处理过的物体图像特征进行分类和进行机器视觉与机械手之间的校准，即把所观察到的物体图像点转换成搬起并移动这个物体对所需的机械手坐标。

最流行的神经网络分类器是基于BP算法的多层前馈网络。然而由于ART1提供在线学习并能分辨出输入模式微小差异，故采用ART1作为视觉子系统的物体分类器。

传统方法处理视觉系统/机械手校准问题是建立系统的数学模型，然后估计从机器视觉坐标系到机械手坐标系之间的传递参数。神经网络方法对解决这一问题提供了一个新的途径。下面着介绍这一方法。

机器视觉/机械手校准：

摄像机被置于机械手工作台上方大约21m处。基于校准过程的神经网要学习摄像机坐标系和机械手坐标系之间的关系。

校准过程的第二步是把物体置放在摄像机视野范围内的已知位置。20个白色方块均匀地放置在摄像机视野内，由机械手逐一拾起来。在这个过程中，就得到了训练集合，该训练集合包括来自每个物体坐标建立的向量。向量由机械手坐标和图像坐标组成。

下一步就是训练神经网学习图像(摄像)机坐标和机械手坐标之间的传递关系。神经网共有四层，一个输入层，二个隐层和一个输出层。输入层有两个神经元，对应于来自摄像机的物体中心的x,y坐标。输出层也有两个神经元，对应于物体中心的机械手坐标。每一个隐层有20个神经元。神经网的初始实验表明双隐层比单隐层收敛要快。

在训练阶段，每一个向量对(图像坐标/机械手坐标)提供给神经网，网络的权不断调整直到得到满意的精度。网络的误差是希望输出和网络输出的累加。

$$\text{error}_n = \sum_{i=1}^{20} |O_d - O_n|$$

O_d为网络的希望输出，O_n为网络实际输出。初始的网络训练要用几天时间来达到误差小于0.0049。但初始训练完成后，如果把摄像机稍微移动一下(小于2.54cm)，对网络进行再训练(系统再校准)时达到同样的误差要求仅用1h。所以，一旦初始校准网络训练后，任何再校准过程会相当快。

然而训练时间过长，利用其它复杂的训练算法可以大大减少训练时间。

相对于传统校准方法，人工神经网络方法易于实现。传统方法需要建立数学模型，进行编码，有效性检验，还需通过图像进行参数估计。尽管神经网的训练时间较长，但整个校准过程比传统方法快得多。

在测试基于校准的神经网时，物体置于训练集位置之间，摄取其图像并计算物体的中心位置。然后用神经网把这个中心位置进行转换产生机械手坐标。这些坐标与置放的物体实际位置相比较。结果表明神经网络输出相对于实际位置的误差小于0.1016cm，最大误差为0.1422cm。结果是令人鼓舞的，说明神经网络有广阔的应用价值。

第 5 章　神经网络在自动控制中的应用

5.1 引　　论

传统的控制理论缺乏对非线性系统，未知动力学和环境参数系统的有效控制手段，虽然自适应控制在线性时不变系统的应用中取得了重大的进展，并有各种稳定的自适应规则相继问世，但是对非线性系统的自适应控制研究的进展还相当缓慢。由于许多自适应规则的计算量大，使得其实时应用碰到了巨大困难。人工神经网络理论研究的兴起和发展，为非线性系统理论的研究开辟了一条新途径。由于神经网络具有大规模并行性、容错性，本质的非线性及自组织、自学习、自适应能力，已经成功地应用到许多不同的领域。本章将讨论自动控制的发展及面临的挑战，阐述神经网络解决非线性系统控制问题的优势并给出神经网络在自动控制中的应用实例。

5.1.1 自动控制的发展与面临的挑战

从自动控制理论的建立至今已 50 年了。现在我们可以对各种类型的工业过程进行自动控制。按照常规的分类，把自动控制分为两个分支，连续控制和离散控制。在连续控制中，所控制的工业过程是时变的，并且已由过去的模拟量闭环控制转为由计算机实施的数字控制；离散控制系统在很大程度上已趋于继电开关式控制（可编程序控制器 PLC），系统实质上进行启动、停止、等待或监测一系列的离散事件。如果从传统的观点来看这两个形式的控制，可以看出大部分工业过程控制是这两种控制形式的混合，即在很多场合，PLC 被用作启动批处理，一旦启动结束后就进入 PID 控制状态。

无论从哪一个角度评价自动控制技术的成果，我们毫不夸张地指出，基于已开发的非常复杂的系统，已经成功地把自动控制技术用于降低产品费用，实现系统鲁棒控制和确保生产过程运行的可靠性等等。然而，大部分自动控制仍局限在小范围内，局限于某一物理参量或某几个物理参量的控制；自动控制应该实现全方位的过程（或对象）的最优调节，这不仅包括机器本身，而且还应考虑人的行为。另一方面，为了能够预测系统在某些情况下的控制策略，必须尽可能准确地对生产或控制过程进行建模。然而，实际上似乎不可能建立一个完整的、复杂的系统模型，而且在很多场合下也是不必要的。需要强调指出的是，当自动控制任一过程时，不但仅仅着眼于局部情形，而且必须关注总的代价（资源、能量和耗费的工作量）。这不可避免地使自动控制走向高度集中的，全面的控制，即无论什么样的被控对象，我们必须对组成整个过程的所有参量进行集中控制。而当我们观察需要集中控制的各种各样的参量的特性时，传统控制理论遇到了棘手的问题，这些系统某些相关的特性为：

(1)固有的不稳定性。

整个过程（或过程的某些组成部分）可能是动态不稳定的。在这样的系统中，不改变控制的状态仍可能导致系统性能的恶化。

(2)不完整/过多的数据。

事实上我们观察到大量数据是不可信的,有噪声并且不完整。另一方面,我们获取的数据太多,以致于必须从中筛选出有用的数据。

(3)不可辨识的过程。

大多数被控过程是可辨识的,但仍存在一些过程不可辨识,这给控制造成了困难。

传统的控制理论是建立在数学模型基础上的,即在设计系统的控制器之前必须建立被控过程(对象)的数学模型。甚至我们认为,如果不能建立系统的线性化模型,就不能实现合理的,满意的控制。因为对于非线性控制系统和时变系统,目前还找不到易于处理的统一的数学方法,现有的进展还只是在某些特殊类型的非线性系统和慢时变过程方面。对于一个复杂的控制系统,基于数学模型的控制算法需要耗费大量的机时,无法实现有效的全面的集中控制。为了解决自动控制中的"瓶颈"问题,我们必须寻找新的途径和方法。

控制系统的数学模型和控制信号的定量描述属于精确知识,可是在自然界中还有一大类不能用数学定量描述来表达的知识,如操作人员的经验,规则,启发性知识,符号逻辑等,同时,在人的生产活动中,确有一大类控制是不用数学模型的。计算机在快速,精确计算方面的能力超过人类,但在处理非精确化知识方面远逊于人类;如果将人脑的智能、计算机技术和自动控制理论结合起来,必定会给自动控制领域带来新的飞跃,人工神经网络是对生物神经系统的模拟,神经网络的自学习、自组织,容错性和信息处理的并行性吸引了国内外许多控制界的学者,并把它引进了工业控制领域。

5.1.2 神经网络用于控制的优越性

从60年代开始,Widrow 和 Hoff 就开始研究神经网络在控制中的应用了。1969年,Minsky 和 Papert 的 perceptron(《感知机》)的出版,对多层神经网络进行了"盖棺定论"的评价,认为感知机的处理能力是十分有限的,加之其它因素的影响,使神经网络的研究进入了萧条时期。自80年代中期以来,一个竞相研究神经网络和设计构造神经计算机的热潮在世界范围的掀起。近年来,神经网络的研究进入了自动控制界。"Neurocontrol"(神经控制)这一新名词已悄然兴起。

为了对这一新技术进行正确评价,我们可以把神经网络方法与传统方法进行对比。神经网络具有以下重要的特征和优势:

(1)非线性特性。

神经网络在解决非线系统控制问题中具有广阔的前景。这种特性来自它理论上能以任意精度实现非线性映射,网络还可以实现较其它方法更优越的系统建模。

(2)并行分布式信息处理。

神经网络具有并行结构,可以进行并行数据处理。这种方法具有较其它过程更强的容错能力。

(3)学习和自适应能力。

神经网络是基于所研究系统过去的数据记录来进行训练的。当提供给网络的输入不包含在训练集中时,一个适当的训练了的神经网络具有归纳能力。神经网络也可以在线进行自适应调节。

(4)多变量系统。

神经网络可以处理很多输入信号,并具有许多输出量,所以很容易用于多变量系统。

从控制理论的角度看,神经网络的处理非线性系统的能力是最重要的。因为至今尚无系统

的、普遍可适用的理论来指导非线性系统的设计。然而，神经网络可以表示非线性映射，因此可以建立非线性系统的模型。一个很有前途的方法是用神经网络提供非线性系统的模型，然后基于理论方法或优化技术设计非线性控制器。

在控制学科中，自适应系统理论是新近发展起来的控制理论分支。在过去的 15 年中取得了令人鼓舞的理论结果。尽管已建立的自适应系统理论是建立在线性定常系统的假设上，但所提出的概念和理论上面临的问题与神经网络领域是相似的。

控制理论领域与神经网络之间的一般关系示于图 5-1。其中空白框表示没有明显的类似关系。

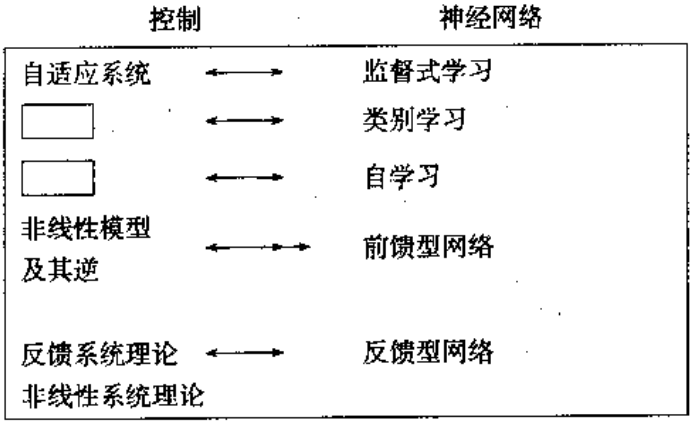


图 5-1　控制与神经网络的关系

5.2　神经网络与系统辨识

一个动态系统可以用两种模型来表示，输入输出模型和状态空间模型。这节分别介绍用前馈神经网络实现这两种模型和辨识，并给出应用实例。

5.2.1　系统辨识的神经网络结构

5.2.1.1　系统数学模型

(1)输入输出模型。

系统的输入输出模型描述系统的动态行为。在离散时间域，输入输出模型可以具有 NARMA 型或 Hammerstein 型。系统的输入输出模型可以根据系统过去时刻的输入和输出量预报系统未来的行为。假定单输入单输出(SISO)系统的输入输出模型由(5-1)式表示：

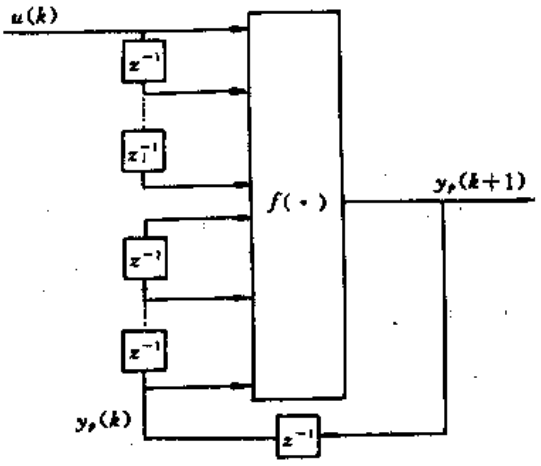


图 5-2　输入输出模型

$$y(k)=f(y(k-1),y(k-2),\cdots,y(k-n),u(k-1),\cdots,u(k-m)) \tag{5-1}$$

式中 $u(k)$，$y(k)$表示 k 时刻系统的输入和输出，n 表示与 $y(k)$相关的过去时刻输出的个数(系

统的阶数)，m 表示与 $y(k)$相关的过去时刻输入的个数。通常 $m \leqslant n$，f 为非线性函数。

(2)状态空间模型。

一个 n 阶多输入多输出(MIMO)时不变非线性系统的动态方程为

$$X(k+1) = \varphi[X(k), U(k)]$$
$$Y(k) = \psi[X(k)] \tag{5-2}$$

这里 $X(k)=[x_1(k),\cdots,x_n(k)]^T$ 是系统的维状态向量，$U(k)=[u_1(k),\cdots,u_r(k)]^T$ 是系统的输入向量，并且 $Y(k)=[y_1(k),\cdots,y_m(k)]^T$ 是系统的输出向量，φ 和 ψ 表示静态非线性映射。

5.2.1.2 基于系统输入输出的辨识(I)

前馈神经网络没有动态存储功能，但可以以任意精度逼近任意非线性函数。系统辨识的任务就是找到适当的映射来近似的描述一个动态系统的输入输出关系。图 5-2 给出了式(5-1)的方框表示。从图中可以看出，一个动态系统由函数 f 和 m、n 来表示。如果 m 和 n 已经给出，则唯一的任务就是找出函数 f，对于时不变系统，f 是不随时间 t 变化的。

由于前馈神经网络可以表示静态的映射，所以前馈神经网络可以用来近似函数 f。按照辨识系统的结构，有两种辨识方案：并行结构和串并联结构，分别示于图 5-3(a)和图 5-3(b)。

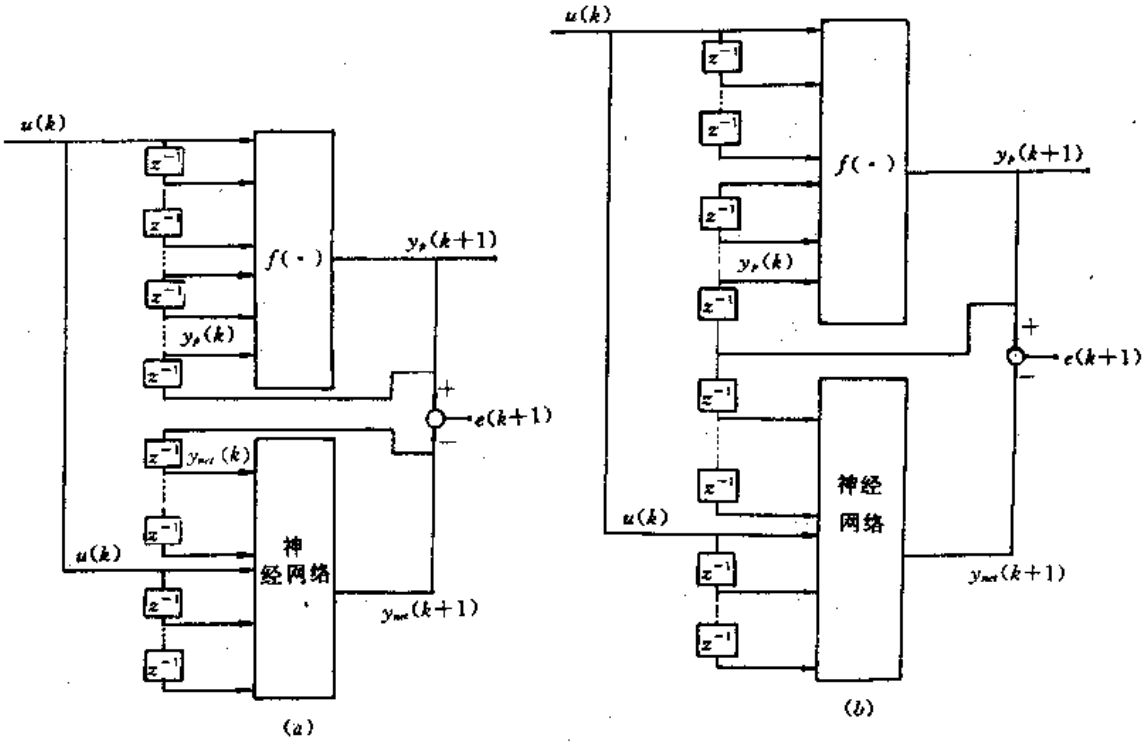


图 5-3 系统辨识结构图

(a)并行辨识结构；(b)串并联结构

在并行结构中，神经网络和系统接受同一外部输入信号，系统的输出不作为网络的输入信号，故网络和系统是相互独立的过程，它们的输入互不干扰，在串并联结构中，与并联不同的是系统的输出作为网络的部分输入信号，这样，网络和系统不再是两个相互独立的过程，网络的动态行为受系统的影响。当图 5-3 所示的辨识结构用于系统辨识，我们假设系统在输入作用

下是BIBO 稳定的。然而，当用并行结构时，并不能确保权重的学习是收敛的或网络和系统间的误差趋于零。

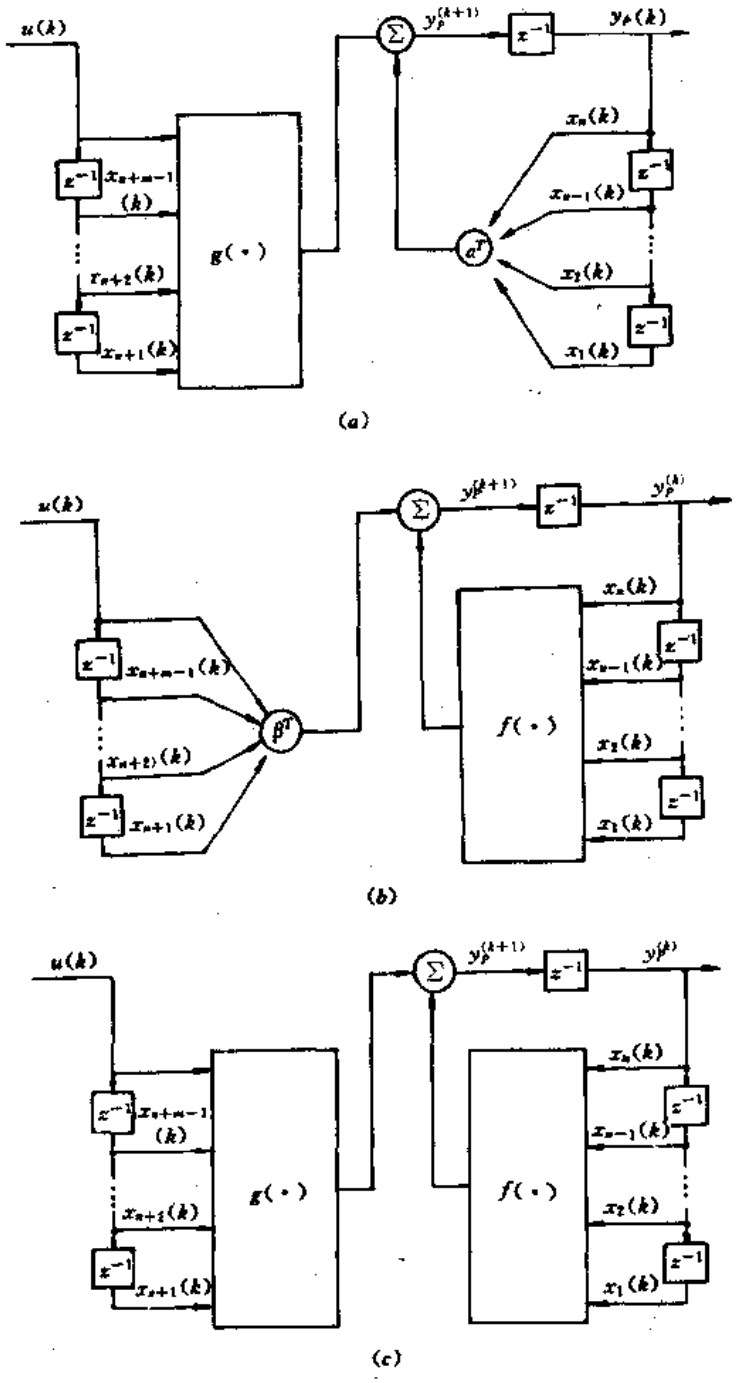


图 5-4　单输入单输出系统的表示

5.2.1.3　基于输入输出的辨识(Ⅱ)

除了式(5-1)描述的输入输出非线性特性外，还有另外三种非线性形式，现表述如下：

模型Ⅰ　　$y_p(k+1)=\sum_{i=0}^{n}a_i y_p(k-i)+g[u(k),\cdots,u(k-m+1)]$；

模型Ⅱ $$y_p(k+1)=f[y_p(k),\cdots,y_p(1)]+\sum_{i=0}^{m-1}\beta_i u(k-i);$$

模型Ⅲ $y_p(k+1)=f[y_p(k),\cdots,y_p(k-n+1)]+g[u(k),\cdots,u(k-m+1)]$。

这里，$[u(k),y_p(k)]$表示 SISO 系统在时刻 k 的输入输出对，并且 $m\leqslant n$。各种模型的方框图表示示于图 5-4。在模型Ⅰ中输出 $y_p(k+1)$和过去时刻值 $y_p(k-i)$之间的关系是线性的，而模型Ⅱ中 $y_p(k)$与过去时刻的输入值 $u(k-j)$呈线性关系。模型Ⅲ输出 $y_p(k+1)$与 $y_p(k-i)$和 $u(k-j)$之间的非线性关系是可分离的。

当我们假设模型中线性部分的参数是已知的，那么多层神经网络可以用来实现非线性函数 $f(\cdot)$和 $g(\cdot)$。设已知神经网络的结构(在这个结构下，网络可通过训练以足够的精度逼近非线性函数 $f(\cdot),g(\cdot)$，则可以用以上介绍的并行辨识结构或串并联辨识结构进行各种模型下的系统辨识，最终用神经网络表征被控制对象的动态行为。

需要指出的是，如果用输入输出数据进行系统辨识，必须假定在允许的输入范围内输出是有限的。这意味着被选择的网络模型也应满足这一特性。在模型Ⅰ中，意味着系统特征方程 $Z^n-a_0Z^{n-1}-\cdots-a_{n-1}=0$ 的根位于单位圆内。对于另外的模型，不存在简单的代数约束，所以网络的稳定性是一个重要的研究领域。

5.2.1.4 基于可观测状态的辨识

式(5-2)$\varphi(\cdot)$和 $\psi(\cdot)$是任意的函数。$\varphi(\cdot)$是状态 $X(k)$和 $U(k)$到新的状态 $X(k+1)$的映射；$\psi(\cdot)$是状态 $X(k)$到输出 $Y(k)$的转换函数。当用神经网实现由式(5-2)描述的系统的辨识时，应满足两条假定条件：(a)所有的系统状态是可观测的；(b)系统是稳定的，基于以上两个假设，图 5-5 结构可用于系统辨识。图 5-5 表明辨识系统需要两个神经网络。网络 1(N_1)把系统输入和状态作为它的输入信号，于是 N_1 的输入层有 $n+r$ 个神经元，又因为 N_1 把系统预测新状态 $\bar{x}(k+1)$作为它的输出，所以输出层有 n 个单元。隐层个数和隐单元数按照精度要求确定。同样地，神经网络 2(N_2)需要 n 个输入单元和 m 个输出单元。

5.2.2 逆动力学系统辨识

动态系统的逆动力学模型在控制系统结构中起着重要的作用，从概念上讲，最简单的逆动力模型是直接逆模型，由图 5-6 所示(这种结构也被称为泛化逆学习)。图中，系统的输出作为网络的输入，网络的输出和训练信号(系统输入)进行比较，这个误差信号被用来训练网络。可以看出，这种结构驱使网络表征系统的逆模型。然而，这种方法的缺陷在于：

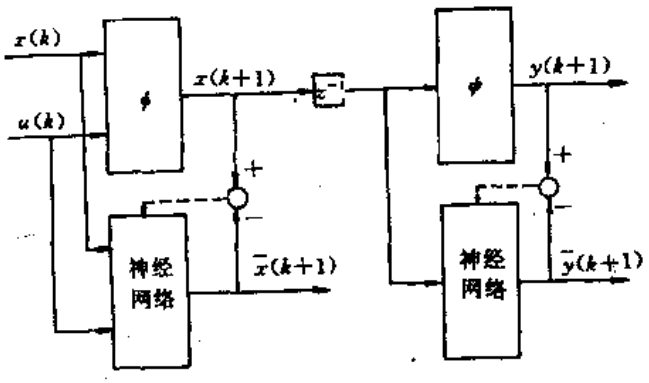


图 5-5 可观测状态的辨识

第一，必须选择输入信号覆盖大范围的系统输入，并且实际工作信号难以确定先前数据记录。控制的目标是使系统以理想的方式运行，但直接逆模型的训练信号并不与这一目标相对应。

第二，如果非线性系统不是一对一的映射，那么可能得到不正确的逆模型。

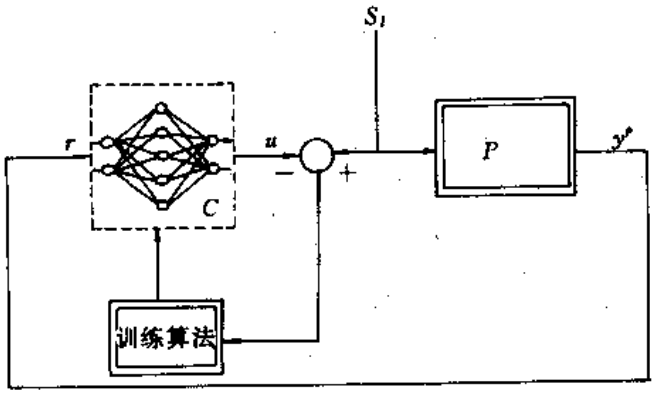


图 5-6 直接逆方法

另一种逆动力模型称为特定逆学习结构，由图 5-7 所示。图中神经网络逆模型位于系统之前，直接利用设定给逆模型的期望输入和受控对象的输出之差调节网络的权值，这个学习结构也包含一个已训练好的系统的正模型网络与系统并行联接。在这种情况下，训练算法中的误差信号是训练信号与系统输出之差(也可以是训练信号与正向模型输出之差，当实际系统的输出不能直接测量时可以用正模型输出代替)。在这个方法中，正模型神经网络所起的只是误差回传传递作用，它即使有一点误差，也不是至关重要的，一般只影响逆模型神经网络的收敛速度，并不影响其最终收敛精度。

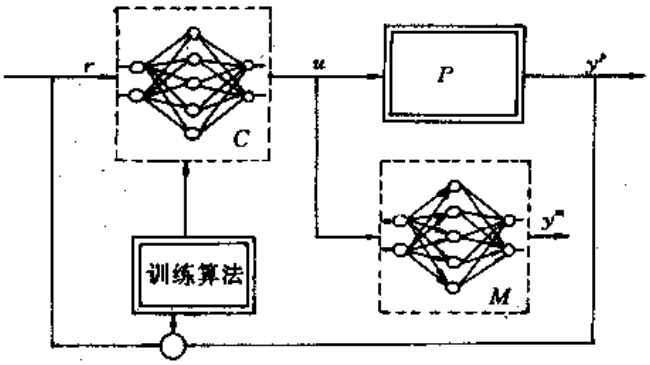


图 5-7 特定逆学习结构

特定逆学习方法与直接逆相比具有下述特征：

第一，学习过程是目标驱使下进行的，因为训练是基于理想系统的输出和实际输出的误差进行的。

第二，如果系统正向模型不是一对一的，那么可以找出一个特定的逆模型。(Jordan 和 Rrmelhart(1991)讨论了找出特定逆模型的方法)

5.2.3 系统辨识仿真实验

研究表明神经网络可有效地用于动态系统的辨识。由输入输出模型表征的非线性系统可以按照模型的的非线性特性进行分类，并且辨识系统的结构为串并联型。高度非线性系统可以用带有 Sigmoid 处理单元(PE)的神经网络进行辨识。研究表明，当被辨识的系统是线性的，用

线性处理单元进行辨识；当被辨识的系统非线性的，可以在神经网络的隐层采用非线性处理单元。读者自然会问一些问题，对于线性系统，线性网络和非线性网络中哪一个更合适？

5.2.3.1 系统模型

三个系统的仿真模型如下：

(1)线性模型。

一采样系统的离散动态方程为

$$y(k) = A_1y(k-1) + A_2y(k-2) + B_1u(k-1) + B_2u(k-2) \tag{5-3}$$

式中 $A_1=1.752821, B_1=0.011698, A_2=-0.818731, B_2=-0.010942$

(2)非线性模型。

一单摆系统的离散时间描述为

$$y(k)=(2-\frac{\lambda T}{ML^2})y(k-1)+(\frac{\lambda T}{ML^2}-1-\frac{gT^2}{L})y(k-2)$$

$$+gT^2/6L\ y^3(k-2)-\frac{T^2}{ML^2}u(k-2) \tag{5-4}$$

式中 M 为摆的质量，L 为长度，g 重力加速度，λ 摩擦系数，y 摆偏离垂直位置的角度，u 施加于摆的外力，可将上式写成

$$y(k) = A_1y(k-1) + A_2y(k-2) + A_3y^3(k-2) - B_1u(k-2) \tag{5-5}$$

具体参数如下：

$$T = 0.2\text{s} \quad g = 9.8\text{m/s}^2 \quad \lambda = 1.2\text{kgm}^2/\text{s} \quad M = 1.0\text{kg} \quad L = 0.5\text{m}$$

(3)"强"非线性模型

该模型由 Narendra 和 Parthasarathy 1990 年提出：

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

5.2.3.2 神经网络辨识器

辨识中用三种类型的神经网络。所有的神经网络都是由带有输入层，单隐层(带有阈值)和不带阈值的输出层构成了多层感知器网络。网络的区别仅在于隐层。

类型Ⅰ——线性网络：隐层中所有的处理单元都具有线性激活函数。

类型Ⅱ——非线性网络：隐层中所有的处理单元具有非线性激活函数：

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{5-6}$$

式中 x 是提供给某一处理单元的总输入，$f(x)$是该处理单元的输出。

类型Ⅲ——混合网络：隐层中一半处理单元具有线性激活函数，另一半处理单元具有式(5-6)所示的非线性激活函数。

5.2.3.3 仿真设置

所有的神经网络具有$(n+m)$个输入单元，预先设置的隐层单元和一个输出单元。训练数据库中包含 400 个数据，在训练中把这个数据库在给定的时间内提供给网络。一次训练的均方

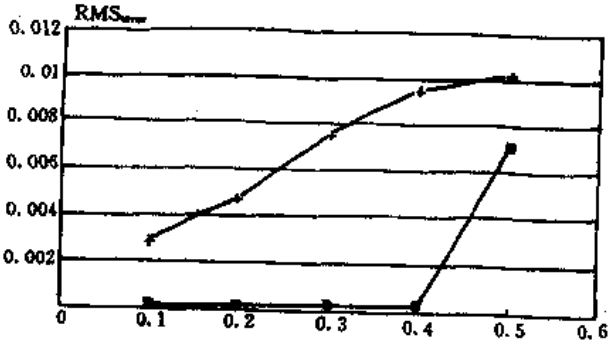
误差被定义为

$$\mathrm{RMS}_{\mathrm{error}}=\sqrt{\frac{\sum_{k=1}^{400}[y(k)-y_n(k)]^2}{400}} \tag{5-7}$$

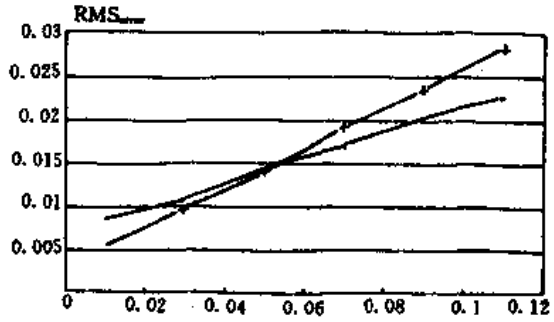
均方误差用来衡量给定网络辨识器的性能。$y(k)$是模型的输出，这个准则也可表示学习过程的收敛速度。学习速度和记忆因子在训练前已被确定。

5.2.3.4 实验结果

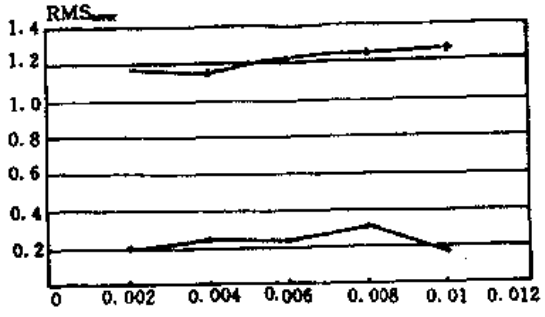
(1)线性系统辨识。



(a)——线性 +非线性 □混合式



(b)——非线性 +混合式



(c)——非线性 +混合式

图 5-8 实验结果

模型由式(5-3)提供。仿真中采用了线性网络，非线性网络和混合型网络。所有的网络有 4

个输入单元，6个隐单元和1个输出单元。采用的学习速度位于0.1～0.5之间，记忆因子=0.1；初始随机权重在[−1，+1]区间内，$u(k)$随机产生位于[−1，+1]区间。训练次数为50。实验结果由图5-8(a)所示。

(2)非线性系统辨识。

线性网络不能辨识非线性系统，因此采用非线性网络(类型Ⅱ)和混合型网络(类型Ⅲ)。网络的参数和结构与上述相同，实验模型数据由式(5-5)提供，实验结果示于图5-8(b)。当实验模型为强非线性时(由式(5-6)产生)，实验结果由图5-8(c)给出。

由图5-8可以看出，线性网络辨识器仅能用于线性系统辨识，而且具有良好的精度；当系统非线性较弱时，采用非线性辨识器或混合型辨识器辨识效果相当；但出现强非线性时，非线性网络辨识器效果最好。

5.2.4 系统辨识应用

5.2.4.1 舰船雷达目标识别

自从雷达问世以来，人们就希望雷达在发现目标之后，还能判别所发现的目标。雷达目标识别就是要通过雷达对目标的类型，真假或属性作出某种判别。依其解决途径，可分为成象识别与特征识别两大类。特征识别是从目标回波中提供出对识别有用的特征信息，经适当处理后，据以判断目标的类型。

舰船雷达目标视频回波的特征抽取由两个处理过程级联而成：富里叶-梅林(Fourier－Mellin)变换(FAT)和编码变换。FAM变换的主要目的在于产生不依赖于舰船雷达视频回波时延与比例变化的物理特征集合，针对目标视频回波 $\vec{X}$ 的FMT变换的结果 $\vec{Z}$ 进行编码变换，将 $\vec{Z}$ 中非负值分量编为代码"1"，其余的编为代码"0"，这样便得到一组二元序列，在这组二元序列中选取特定的双位代码(01，10，00，11)和三位代码(000，001，…，111)在该二元序列中出现的频率(F_{00}，F_{01}，F_{10}，F_{11}，F_{000}，F_{001}，…，F_{111})作为特征来表示舰船雷达目标。显然，他们具有对目标运动姿态(即距离和方位)变化不敏感特性。

小规模神经网络(BP网络)对按上述方法抽取的几个舰船雷达目标视频回波特征 F_{00}，F_{01}，F_{10}，F_{11}，F_{000}，…，F_{111}进行分类识别。网络的学习算法采用基本BP算法和广义 δ 规则。文献[36]根据实地录取的舰船雷达目标视频回波，并对数据进行归一化处理后，将其作为训练集。神经元网络为多层前馈型，输入单元数为12，表征目标视频回波的特征向量，输出单元数为3，表征目标的分类，隐层单元数由实验确定。仿真结果表明，利用BP网络可以较好地实现舰船目标的识别，但收敛速度与选取的参数有关。按广义 δ 规则进行训练可以提高收敛速度，但在误差小于0.1的准则下，迭代次数达800，为了减少迭代次数，加快收敛速度，文献[36]提出了函数联接网络。采用函数联接网络，可以用两层感知机网络进行目标识别，其收敛速度和学习速率均高于BP网络及其改进算法。

5.2.4.2 青酶素批量进料发酵中的生物量估计

受目前工艺水平和理论研究水平的限制，尚不能对发酵过程进行精确的描述。青酶素发酵过程中要控制酶中的生物量，但现有的传感器技术还不能在线测量生物量。为了优化青酶素生产，生物量的增长率必须控制在一个低水准内。生物量的增长率可以由操作量来影响，但不能

在线检测它的变化。于是用吸氧率OUR作为在线测量变量。

连续发酵系统一般工作在稳态附近，一般可采用线性估计器。但批量进料的青霉素发酵过程要复杂得多，该过程跨越非常广的非线性动态范围，没有稳态，可以说是一个复杂过程的建模。吸氧率的当前测量值提供了关于酶中生物量浓度的信息，可以作为神经网络的一个输入量，又由于发酵的特征是一个时间的函数，可以把批量时间作为网络的另一个输入量。由于构成一个2×3×1的前馈网络，输出量表示生物量的浓度。用以往发酵过程的数据记录训练该神经网，就可以估计青酶素发酵生物量浓度。结果证明神经网络构成的估计器比以往的非线性观测器估计的效果要好。

5.3 神经网络非线性控制

过程控制的对象有些是复杂的非线性系统，其控制依赖于人的智能经验，仅靠常规控制策略是无法完成的。控制系统的动态模型及其逆动力学模型可以直接地用于系统控制中。在上一节中，讨论了用神经网络可以实现复杂非线性系统的建模，本节要利用系统的神经网络模型及其逆模型进行非线性控制系统结构的探讨和应用实例介绍。

5.3.1 神经控制结构(Ⅰ)

尽管文献中提供了各种各样的神经控制结构，但从控制理论的角度，选择几种较成熟的控制结构加以阐述。

5.3.1.1 监督式控制

在许多工况下，人必须对生产过程进行手工操作来形成反馈控制，因为在这些工况下用常规控制技术难以设计一个自动控制器。在这种情况下，如果能够模拟人的行为来设计一个控制器将是十分理想的。

由于神经网络提供了一种知识表达的手段，即用网络结构和权重实现(记忆)任何复杂的输入输出关系，所以神经网络可用来实现这一智能控制器。在原理上，训练一个神经网络和上一节所述的学习系统的正向模型是类似的。然而，在这种情况下，网络的输入对应于人接受到的传感器的信息，训练时使用的网络目标输出(理想输出)对应于人对系统的控制作用。这种方法在标准的小车-倒摆控制系统中得到了应用(Grant and zhang(1989))。

图5-9是监督神经网络控制(简称SNC)。除受控过程以外，它包括一个导师和一个可训练的控制器(神经网络)。

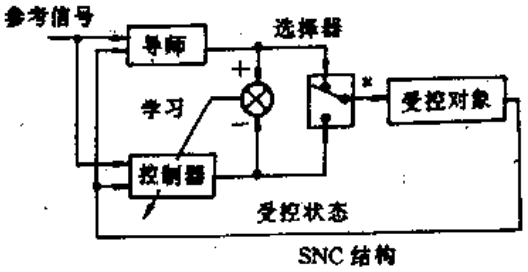


图5-9 SNC结构模型

在人的控制过程中，人作用于系统的控制信号及过程的状态通过传感器采集起来，用于网络的训练。在训练过程中，采集到的过程状态及当前时刻之前的控制信号作为网络的输入，而当前时刻的控制信号作为网络的期望输出。训练完成了，就可以用网络来实现正确的控制了。训练可以在线进行，也可以离线进行。后

者的好处是可以将有限采样数据反复使用，节省现场测试时间。网络训练好后可以代替人实现自动控制。

5.3.1.2 直接逆控制

直接逆控制(Direct Inverse Control)利用逆动力学模型。逆模型直接与被控制系统相串联以致于实现理想响应和被控系统输出的等同映射。在这样的结构中，神经网直接作为控制器。直接逆控制常用于机器人控制。这个方法依赖于作为控制器的逆动力学模型。显然，由于没有反馈存在，这种系统缺乏鲁棒性。这个问题在某种程度上用在线学习来解决，即逆模型的参数可以在线调节。

5.3.1.3 多层神经网控制器

多层神经网控制器实质上是一个前向控制器，由 Psaltis 于 1988 年提出。

(1)间接学习结构

间接学习结构由图 5-10 给出。这个结构中，有两个需要训练的神经网，每一个网络作为一个逆动力学辨识器。训练的目标是从理想响应 d 中找出合适的控制作用 u。网络权根据网络 I 和网络 II 输出之间误差来调节以使 e 最小；因为如果网络完成训练后，则 $y=d, u=u^*$。然而，这种结构不能确保理想输出 d 和实际输出 y 之间的误差是最小的。

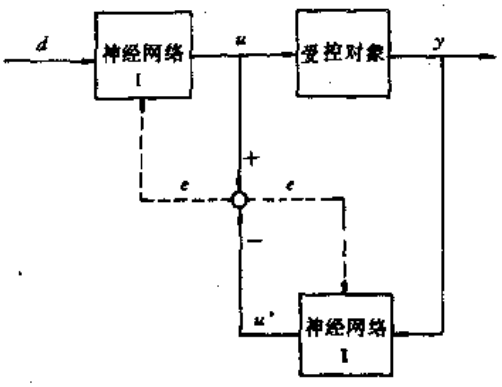


图 5-10 间接学习结构

(2)泛化学习结构

泛化学习结构示于图 5-11。在这个结构中，以最小化被控制对象 p 的输入 u 和网络输出 u^* 为目标来训练神经网。在训练中，u 应该位于某一变化区域内以使得输出 y 覆盖理想的输出 d。训练完成之后，如果理想输出 d 作用到网络，那么这个神经网络就可以提供适当的 u^* 给被控制对象，这个结构的局限性在于不知道什么样的 u 与理想的输出 d 相对应，所以神经网必须用跨越很大范围的 u 来训练，使得在训练中系统输入 y 包含 d 的理想值。

(3)特定学习结构

图 5-12 给出了特定学习结构。在这个结构中，理想的输出 d 是网络的输入。通过应用误差反向传播算法，在训练中理想输出 d 和系统的实际输出 y 之间的误差成为最小。于是，不仅可以得到良好的输出响应，而且可以使训练位于理想输出的范围内。为了能够训练这个网络，必

须知道系统的动态模型或者取得某些近似。多层神经网络控制器的训练采用误差反向传播算法，误差可以是理想输出和实际输出之差，也可以是正确的被控制对象输入信号和由神经网计算出的输入之差。

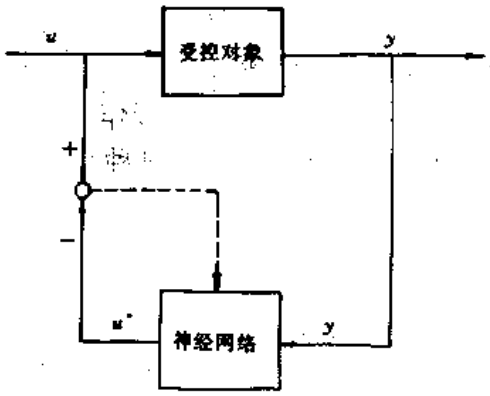


图 5-11 泛化学习结构

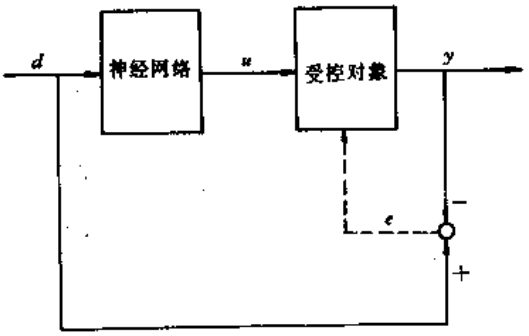


图 5-12 特定学习结构

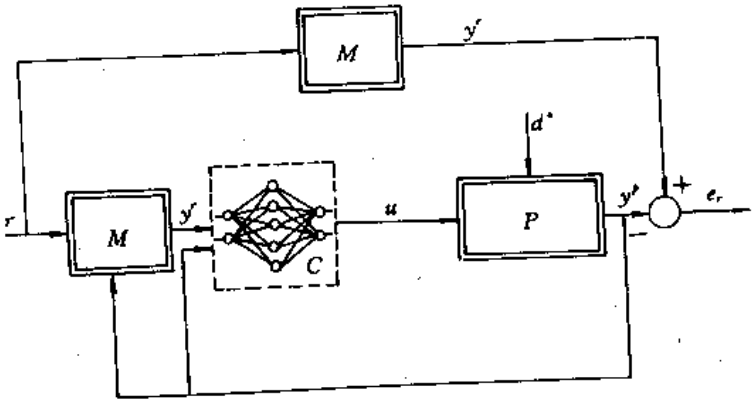


图 5-13 模型参考结构

5.3.1.4 模型参考控制

假设闭环系统的希望特性由一个稳定的参考模型 M 提供，可以由它的输入-输出对$\{r(t), y^r(t)\}$来定义。控制系统试图使系统输出 $y^p(t)$渐近地逼近参考模型输出。即

$$\lim_{t\to\infty} \| y^r(t) - y^p(t) \| \leqslant \varepsilon$$

ε 是任意给定的常数$\varepsilon \leqslant 0$。实用的模型参考结构如图 5-13 所示。在这个结构中，上述定义的误差用来训练作为控制器的神经网络。显然，这种方法与以上谈到的系统逆动力模型的训练有关。如果参考模型为一等同映射，那么这两种方法就是一回事了。

5.3.2 神经控制结构(Ⅱ)

5.3.2.1 内模控制

在内模控制中，着重于系统正向和逆模型的作用。在这个结构中，系统的正模型和逆模型被直接作为反馈回路中的部件。内模控制具有良好的鲁棒性和稳定性，并且易于扩展为非线性控制。

在内模控制中，系统的模型与实际系统并列，系统输出与模型输出之差作为反馈信号。然后，这个反馈信号馈送到系统前向通道中的控制器的子系统，内模控制的性质表明控制器部分与系统的逆模型有关(内模控制的神经网实现示于图 5-14)。给定系统正向和逆向动态网络模型，就可以直接实现神经网络内模控制；系统模型 M 和控制器 N_c(逆模型)用神经网实现。子系统 F 通常是一个线性滤波器，用于增强系统的鲁棒性和提高闭环系统的跟踪性能。

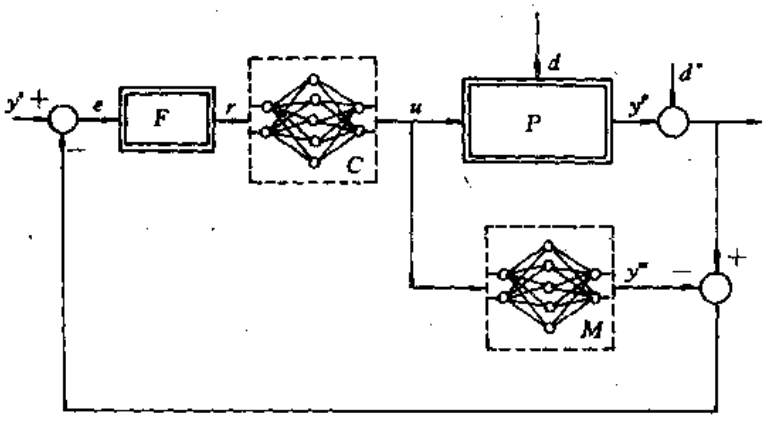


图 5-14 内模控制结构

应该注意到内模控制仅限于开环稳定的系统。然而，内模控制广泛地用于过程控制中。从控制理论的角度看，内模控制方法是广泛受欢迎的。

内模控制有如下重要特性

(1)假设对象和控制器是输入输出稳定的，且模型是对象的完备表示，则闭环系统是输入输出稳定的；

(2)假设描述对象模型的算子的逆存在，而且这个逆用作控制器，构成的闭环系统是输入输出稳定的，则控制是完备的，即总有 $y(k)=y^r(k)$；

(3)假定稳定状态模型算子的逆存在，稳定状态控制器算子与之相等，且用此控制器时闭环系统是输入输出稳定的；那么，对于常值输入，控制是渐近无偏差的。

5.3.2.2 预测控制

预测控制是70年代后期发展起来的一类新型计算机控制算法，这种算法的本质是预测模型、滚动优化和反馈校正。

预测模型根据系统当前的输入、输出信息和未来的输入，预测未来的输出值。系统的预测值由神经网络提供，并且该预测值输送给非线性优化器。优化器根据特定的性能指标计算出最优指标下的合适的控制信号。系统选择控制信号 u' 以使平方性能指标极小

$$J=\sum_{j=N_1}^{N_2}[y'(t+j)-y^m(t+j)]^2+\sum_{j=1}^{N_2}\lambda_j[u'(t+j-1)-u'(t+j-2)]^2$$

这里 N_1,N_2 由跟踪误差的时间长度和所考虑的控制增量来决定，λ 为控制加权系数。预测控制结构如图5-15所示。

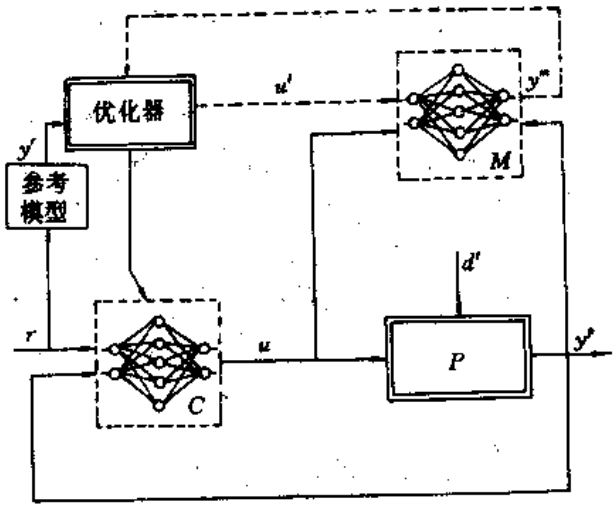


图5-15 预测控制结构

由图5-15可知，可以通过训练一个神经网来模拟优化轨迹。控制器网络被用来训练产生和优化轨迹相同的控制输出 u。这个方法的优点在于：一旦训练完成后，包括系统模型和优化路径在内的外部闭合环路可以舍去；外回路的原始功能是消除预报误差，克服系统中存在的不确定性，这就是预测控制的第三个要点，即反馈校正。

预测控制算法如下：

第一步：获取未来期望输出序列

$$y'(t+j)\qquad j=N_1,N_1+1,\cdots,N_2$$

第二步：采用神经网络模型，产生预报输出

$$y^p(t+j/t)\qquad j=N_1,N_1+1,\cdots,N_2$$

第三步：计算与未来时刻期望值误差

$$e(t+j)=y^p(t+j/t)-y'(t+j)$$

第四步：极小化性能指标 J，获取最优控制序列

$$u'(t+j)\qquad j=0,1,2,\cdots,N$$

第五步：采用第一控制量 $u(t)$，返回第一步。

5.4 神经网络自适应控制

在过去的 20 年中，对于含有未知参数的线性时变系统的研究和应用取得了很多成果。在 70 年代来，大多理论工作面向如何确定自适应律来调整控制参数向量 $\theta(k)$ 以使得整个系统是稳定的。在 1980 年，人们确信当有关系统传递函数的信息已知时，无论对离散时间或连续时间系统来说，稳定的自适应律都是存在的，从那时起，很多研究工作趋于如何确定在不同形式扰动下保证鲁棒性的工作条件。

相反，在非线性系统的自适应控制方面几乎没有什么成果。这是因为几乎没有可依据的理论来指导非线性自适应系统的分析。在自适应控制理论的发展史上存在两条不同的分支：直接自适应控制和间接自适应控制。所谓直接自适应控制，就是直接依据对象的知识来调整控制器的内部参数，以使得对象的输出误差尽可能小，间接自适应控制就是在任一时刻 k，系统要估计被控对象的参数向量 $\hat{P}(k)$，然后假定对象参数向量 $\hat{P}(k)$ 足以表征实际参数向量 $P(k)$，并选择控制器参数向量 $\theta(k)$。即使系统是线性时不变的，直接自适应控制器或间接自适应控制器的加入使系统成为非线性系统。图 5-16 是自适应控制框图。图 5-16(a)为直接自适应控制，图 5-16(b)是间接自适应控制。

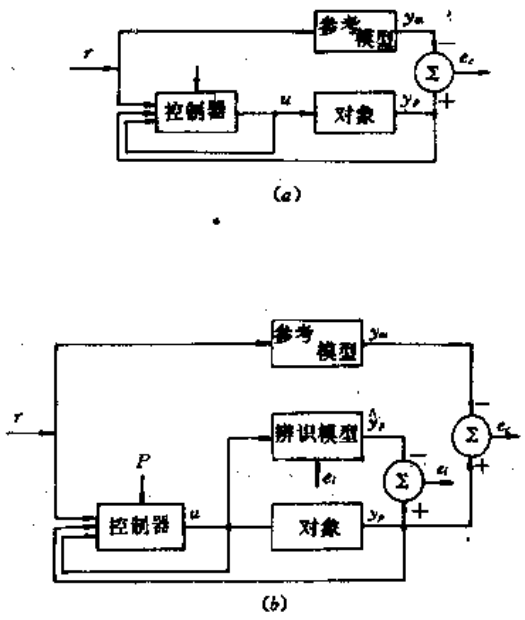


图 5-16 自适应控制器框图

5.4.1 神经网络直接自适应控制

图 5-17 是神经网络直接自适应控制器框图。与图 5-16(a)相对照，控制器由神经网络替代，并且参考模型输出与系统激励信号相同。

设非线性控制系统的描述为

$$y(k+1) = f[y(k), u(k)] \tag{5-8}$$

神经网络训练准则应取为

$$J_1 = \| u(k) - u_d(k) \|^2$$

或

$$J_2 = \sum_{k=1}^{\infty} \| u(k) - u_d(k) \|^2$$

其中 $u(k)$ 为神经网络控制器的输出量，$u_d(k)$ 是当期

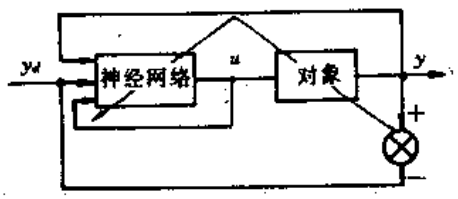


图 5-17 神经网络直接自适应控制

望的对象输出为 $y_d(k)$时，神经网络应提供的控制量。在对象模型未知的情况下，怎样才能知道对象应具有的输入 $u_d(k)$呢？为了解决这一矛盾，可以把神经网络控制器与对象看作一个整体，即一个更多层的神经网络，网络最后一层为被控对象，当然也是未知的，这时取训练准则为

$$J = \| y(k+1) - y_d(k+1) \|_Q^2 = \| e(k+1) \|_Q^2 \tag{5-9}$$

我们的目标很明确，选取当前控制量 $u(k)$，使下一步（或多步）的输出误差尽可能小，上式中 Q 是对误差向量 $e(k+1)$各分量的加权矩阵，为正定矩阵，为简单起见，取为时间 k 的常数阵。

神经网络自适应控制调节算法：

神经网络作为自适应控制器，其控制作用的调整依赖网络权系数的改变。设神经网络最后一层权重 $\theta(k)$的调整规则为

$$\theta(k+1) = \theta(k) - \eta \frac{\partial J}{\partial \theta(k)} \tag{5-10}$$

为了计算$\frac{\partial J}{\partial \theta_{(k)}}$，可导出

$$\frac{\partial J}{\partial \theta(k)} = \frac{\partial J}{\partial u^T(k)} \cdot \frac{\partial u(k)}{\partial \theta(k)} \tag{5-11}$$

和

$$\frac{\partial J}{\partial u^T(k)} = \frac{\partial J}{\partial y^T(k+1)} \cdot \frac{\partial y(k+1)}{\partial u^T(k+1)}$$

$$= 2[e(k+1)]^T Q \cdot \frac{\partial y(k+1)}{\partial u^T(k)} \tag{5-12}$$

为了避免直接计算对象的 Jacobi 矩阵$\frac{\partial y(k+1)}{\partial u^T(k)}$，利用先验知识确定 $u(k)$各分量加入后 $y(k+1)$的各分量变化趋势，并用 $u(k)$各分量对 $y(k+1)$各分量影响的符号组成的矩阵 $\mathrm{sgn}[\frac{\partial y(k+1)}{\partial u^T(k)}]$来替代式(5-12)中的$\frac{\partial y(k+1)}{\partial u^T(k)}$，得到

$$\frac{\partial J}{\partial u^T} = 2[e(k+1)]^T Q \ \mathrm{sgn}[\frac{\partial y(k+1)}{\partial u^T(k)}] \tag{5-13}$$

同时我们知道 $u_j(k)$是向量 $U(k)$的第 j 个分量，也是神经网络最后一层第 j 个神经元的输出。根据 BP 算法，有

$$\frac{\partial u_j}{\partial \theta_i(k)} = \sigma' u_j(k) \cdot c_i(k) \tag{5-14}$$

式中 $c_i(k)$ 为紧靠输出层的前一层第 i 个神经元的输出，$\sigma(\cdot)$是该单元的神经元特性。将式(5-11)与式(5-13)、式(5-14)结合起来，可得到神经网络最后一层的权重修改公式

$$w_{ij}(k+1) = w_{ij}(k) - \eta \delta_j(k) c_i(k)$$

$$\delta_j(k) = 2\sigma' u_j(k)[e(k+1)]^T Q \ \mathrm{sgn}[\frac{\partial y(k+1)}{\partial u_j(k)}] \tag{5-15}$$

对于其它各层权重调整，仍按标准 BP 算法，逐层计算。

可以看出，直接自适应控制有下述两个显著特点：

第一，没有特定的学习阶段，不依赖于被控对象的数学模型辨识，仅用少量的先验知识，控制器是直接根据控制效果在线完成设计的，第二，控制器的参数调整是一个随时间变化的自适应过程。

5.4.2 神经网络间接自适应控制

由于被控对象的非线性特性是未知的，用直接自适应控制是相当困难。已有的仿真实验表明，即使式(5-12)中 Jacobi 矩阵已知，用 BP 算法训练控制器时收敛速度很慢。所以，非线性自适应控制系统必须采用间接自适应控制器，这意味着在实现控制之前，必须用 5.2 节介绍的系统辨识方法在线辨识被控对象的模型；然后利用建立了的神经网络模型来调整控制器参数。其神经网络间接自适应控制结构如图 5-18 所示。图中 TDL 表示时滞环节。

仿真实验

自适应地控制一个非线性对象的过程很大程度上依赖于对已知对象了解的先验知识。例如系统在无外作用情况下平衡状态的个数和平衡点的稳定性，以及使输出有界的输入信号的幅值范围。

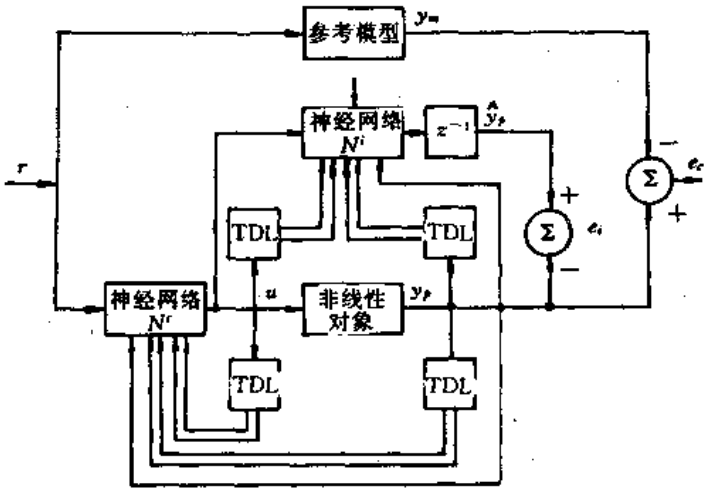


图 5-18 神经网络间接自适应控制

设一非线性系统的差分方程为

$$y_p(k+1) = f[(y_p(k), y_p(k-1)] + u(k)$$

这里函数

$$f[y_p(k), y_p(k-1)] = \frac{y_p(k)y_p(k-1)[y_p(k)+2.5]}{1+y_p^2(k)+y_p^2(k-1)} \tag{5-16}$$

被假设是未知的。参考模型由下式表示：

$$y_m(k+1) = 0.6y_m(k) + 0.2y_m(k-1) + r(k)$$

$r(k)$是一个有界的参考输入。如果输出 $e_c(k)$被定义为 $e_c(k)=y_p(k)-y_m(k)$，控制的目标就是确定一个有界的控制输入 $u(k)$，使得 $e_c(k)$随时间趋于零。如果式(5-16)中的函数 $f(\cdot)$是已知的，$u(k)$就可以根据 $y_p(k)$以及它过去时刻的值计算出来，即

$$u(k) = -f[y_p(k), y_p(k-1)] + 0.6y_p(k) + 0.2y_p(k-1) + r(k) \tag{5-17}$$

这时 $e_c(k+1)=0.6e_c(k)+0.2e_c(k-1)$。由于参考模型是渐近稳定的，对于任意初始条件，$e_c(k)$随时间趋于零。然而，$f(\cdot)$是未知数的，必须采用神经网络 N 来估计函数 $f(\cdot)$，记为 $N[y_p(k), y_p(k-1)]$。

在任一时刻 k，用 $N[\cdot]$代替 $f(\cdot)$所得的控制作用

$$u(k) = -N[y_p(k), y_p(k-1)] + 0.6y_p(k) + 0.2y_p(k-1) + r(k) \tag{5-18}$$

那么，非线性差分方程

$$y_p(k+1) = f[y_p(k), y_p(k-1)] - N[y_p(k), y_p(k-1)] + 0.6y_p(k) + 0.2y_p(k-1) + r(k) \tag{5-19}$$

整个系统的结构示于图 5-19。

在第一阶段，用随机输入离线辨识未知对象，然后用式(5-18)产生控制输入。

在第二阶段，对辨识和控制用不同的采样周期来控制。T_i 表示辨识采样周期，T_c 表示控制采样周期。一般来说，我们希望控制参数的调整速率要慢于辨识参数的调整速率。文献[1]表明，当 $T_i=T_c=1$s 时，系统的响应是渐近稳定的。当 $T_i=1$s 且 $T_c=3$s 时，系统仍是渐近稳定的，但当 $T_i=T_c=10$s 时，系统的输出出现发散形式。

仿真结果表明，对于稳定有效控制的系统，在控制器投入系统之前，神经网络辨识器必须足够精确地逼近系统模型，并且慎重地选择采样周期 T_i 和 T_c。

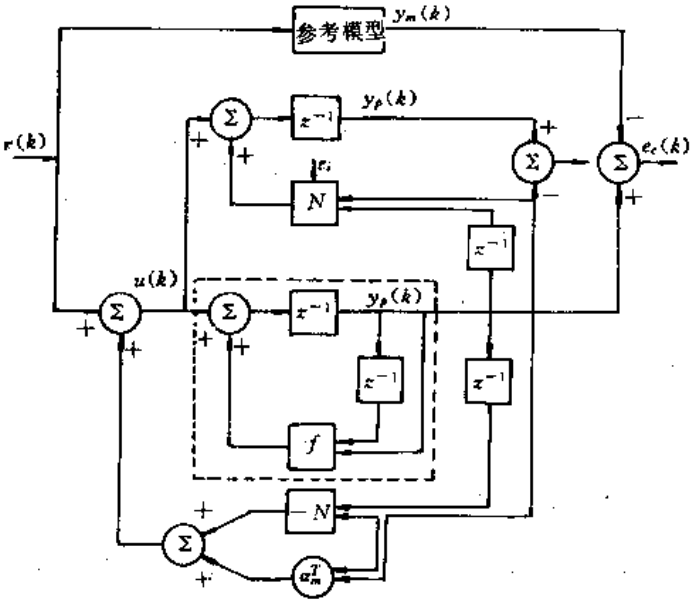


图 5-19 神经网络自适应控制结构图

5.4.3 神经网络自适应线性控制

Hopfield 网络可以用于线性系统的动态控制器，在这种情况下，变结构理论可用来建造控制器，而且这种控制器有强鲁棒性。自适应线性控制的一般结构如图 5-20 所示。

Hopfield 网络可用为自适应机制的一部分，这时，网络位于自适应通道中，它被用来极小化所有状态的平方误差。这时网络的输出代表系统线性模型的参数。这种方法在时变和时不变系统的最小方差估计的应用中很有效。

5.4.4 神经网络自校正控制

用自适应方法来解决跟踪参考轨迹的跟踪控制问题，当实际对象是线性系统时，较有效的系统有自校正调节器和模型参考自适应控制系统，当对象为非线性时，Narendra 提出用神经网络作为控制器和对象的辨识模型，前者组成基于神经网络的自校正调节器，后者组成基于神经网络的模型参考控制器，能够在一定程度上解决非线性系统的跟踪控制问题。

神经网络自校正间接调节系统如图 5-21 所示。其中 $f(y_k, W)$ 表示逼近 $f(\cdot)$ 的权空间 W，输入等于 y_k 的神经网络的输出，而 $g(y_k, V)$ 表示逼近 $g(\cdot)$ 的权空间为 V，输入等于 y_k 的神经网络的输出。sign(y_k) 已知，取误差函数为 $E_k=(d_{k+1}-y_{k+1})^2/2$，按 BP 算法实时调整 W 和 V。仿真结果表明，该方法有一定的有效性，但收敛速度慢，且可能出现振荡，有待于进一步改进，以使系统满足稳定性的要求。

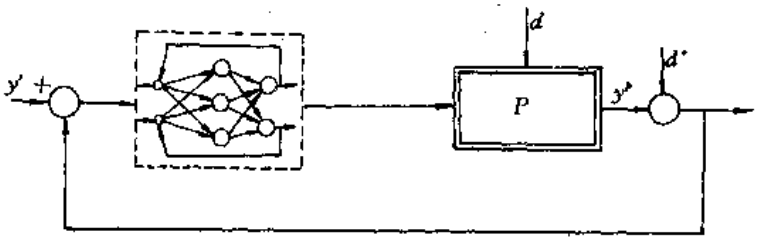


图 5-20　控制器的 Hopfield 网络结构

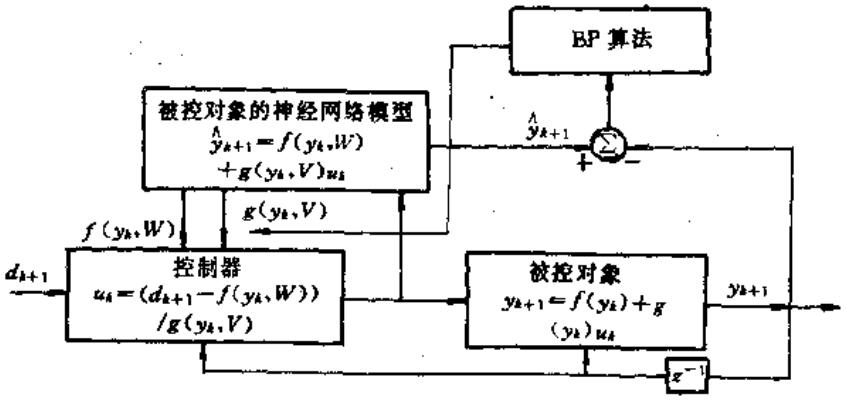


图 5-21　神经网络自校正间接调节系统

5.5　神经控制仿真研究

神经网络的出现给非线性系统的建模和控制提供了很好的工具，神经网络控制系统中具有较好仿真结果的控制方案很多，但理论研究尚处于起步阶段，仿真研究是迈向实际应用中不可缺少的一步，对神经控制理论的发展和应用的开发都是很重要的。

5.5.1　小车-倒摆控制

图 5-22 是一个模拟的小车-倒摆控制系统，它包括一个倒摆，长度 $2L$，质量 m，铰在质量为 M 的小车上。小车在水平导轨上运行，倒摆在与导轨垂直的平面内运动。

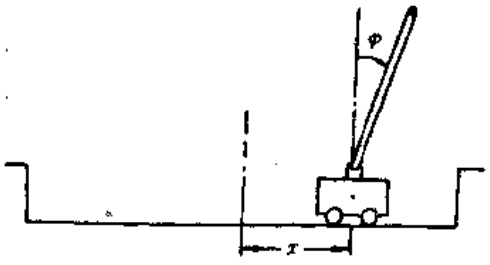


图 5-22　小车倒摆控制系统

为了简化分析，假定倒摆是均匀细杆，无执行器动力学及轴上摩擦问题，系统的动力学方程为

$$\ddot{x}=\frac{m\left[L\dot{\psi}^2\sin\psi-\frac{3}{8}g\sin2\psi\right]-f\dot{x}+u}{M+m\left[1-\frac{3}{4}\cos^2\psi\right]} \tag{5-20}$$

$$\ddot{\psi} = \frac{3}{4L}(g\sin\psi - \ddot{x}\cos\psi) \tag{5-21}$$

这是一个模拟的四维非线性动力学系统。其中 g 为重力加速度。状态变量为小车位置 x，小车速度 $\dot{x}$，杆的角度 ψ 和杆的角速度 $\dot{\psi}$，记为向量

$$Z = [x, \dot{x}, \psi, \dot{\psi}]$$

现在构造一个 4 层神经网络作为控制器。网络的输入层包含 4 个神经元，第一隐层包含 16 个神经元，第二隐层 4 个神经元，输出层只有一个神经元。输入层神经元特性是线性的，其它层神经元特性均为 Sigmoid 型曲线，其中输出神经元的输出值在 $-k$ 到 $+k$ 之间连续变化。k 为小车-倒摆系统控制信号的最大幅度。神经网络的输入与四个状态向量对应，输出为作用在小车-倒摆系统中的控制信号 u。

小车-倒摆系统的状态以及导师加入系统后控制信号的采样值被记录下来作为神经网络的训练数据。用误差反向传播算法(BP 算法)进行离线训练。

计算机仿真时，取 $M=1\text{kg}$，$m=0.1\text{kg}$，$L=1\text{m}$，$f=5\text{kg/s}$，$g=9.81\text{m/s}^2$。

5.5.1.1 线性导师监督学习

首先将小车-倒摆系统在 $\psi=0$ 附近线性化，根据式(5-20)和式(5-21)得到

$$\ddot{x} = \frac{1}{M}(u - f\dot{x})$$

$$\ddot{\psi} = \frac{3}{4L}(g\psi - \ddot{x})$$

作为导师的线性控制律为

$$u = k_1 x + k_2 \dot{x} + k_3 \psi + k_4 \dot{\psi}$$

其中 $k_1=11.01$，$k_2=19.68$，$k_3=96.49$，$k_4=35.57$。在对网络训练 20000 次后，网络记忆了线性控制律。用线性导师和训练好的监督式神经网控制器比较，从初态开始，二者的运行轨迹几乎是相同的。

5.5.1.2 非线性导师监督学习

上述的线性导师监督控制只能保证在原点附近是合理，有效的。对于偏离原点很远的初态，线性化模型的误差太大，不能达到要求。为解决这个问题，需要寻找更复杂的导师。其办法是采用反馈线性化和解耦变换(FLDT)的非线性反馈，由此来抵消系统的非线性并把它变换成一个线性可控的形式。这种控制的一个主要优点是变换后的系统极点可由状态反馈任意配置。

把式(5-20)，(5-21)改写为

$$\ddot{x} = \frac{f_1 + u}{f_2}$$

$$\ddot{\psi} = h_1 - h_2 x \tag{5-22}$$

其中

$$h_1 = \frac{3}{4L} g\sin\psi$$

$$h_2 = \frac{3}{4L} g\cos\psi$$

$$f_1 = m[L\psi'^2\sin\psi - \frac{3}{8}g\sin 2\psi] - fx'$$

$$f_2 = M + m[1 - \frac{3}{4}\cos^2\psi] \tag{5-23}$$

用作导师 FLDT 控制律为

$$u = \frac{f_2}{h_2}[h_1 + k_1(\psi - \psi_d) + k_2\psi' + c_1(x - x_d) + c_2x'] - f_1$$

其中

$$k_1 = 25, k_2 = 10, c_1 = 1, c_2 = 26$$

网络训练的方法与线性控制律训练相同，但训练时间比较长，大约要 800000 次，仿真表明，FLDT 控制与其训练出的监督式控制有基本相同的暂态和稳态性能。

5.5.2 单神经元自适应 PID 控制器

传统的 PID 调节器由于其结构简单、调整方便，因而在过程控制中获得广泛应用，但对一些复杂过程且参数慢时变的系统，由于 PID 的参数不易实时在线调整，因而在应用中遇到一些困难。针对上述情况，设计一个具有自适应、自学习功能的单神经元智能控制器，对提高这类系统的控制效果和鲁棒性有积极意义。下面来讨论单神经元自适应 PID 智能控制器。

5.5.2.1 PID 调节器的离散差分形式

众所周知，连续系统 PID 调节器的算式为

$$u(t) = K_p[e(t) + \frac{1}{T_i}\int e(t)\mathrm{d}t + T_d\frac{\mathrm{d}e(t)}{\mathrm{d}t}]$$

式中 K_p——比例增益；

T_i——积分时间常数；

T_d——微分时间常数。

当采样周期 T_0 较短时，可通过离散化将这一方程直接化为差分方程。为此用一阶差分代替微商，用求和代替积分，这时用矩形积分来求连续积分的近似值，即可求出 PID 调节器的离散方程

$$\begin{aligned}\Delta u(k) &= K_p[\Delta e(k) + \frac{T_0}{T_i}e(k) + \frac{T_d}{T_0}\Delta^2 e(k)] \\ &= K_I e(k) + K_P\Delta e(k) + K_D\Delta^2 e(k)\end{aligned} \tag{5-24}$$

式中 K_I——积分比例系数，$K_I = K_pT_0/T_i$；

K_D——微分比例系数，$K_D = K_pT_d/T_0$；

Δ^2——差分的平方，$\Delta^2 = 1 - 2Z^{-1} + Z^{-2}$。

5.5.2.2 单神经元自适应 PID 控制器及其学习算法

用单神经元实现自适应 PID 控制的结构框图如图 5-23 所示。图中转换器的输入反映被控过程及控制设定的状态，如设定为 $y_r(k)$，输出为 $y(k)$，经转换器后转换成为神经元学习控制所需要的状态量 x_1、x_2、x_3。这里 $x_1(k) = e(k)$，$x_2(k) = \Delta e(k)$，$x_3(k) = e(k) - 2e(k-1) + e(k-2)$，$z(k) = y_r(k) - y(k) = e(k)$ 为性能指标或递进信号，$w_i(k)$ 为对应于 $x_i(k)$ 的加权系数，K 为

神经元的比例系数，$K>0$。神经元通过关联搜索来产生控制信号，即

$$u(k)=u(k-1)+K\sum_{i=1}^{3}w_i(k)x_i(k) \tag{5-25}$$

单神经元自适应控制器通过对加权系数的调整来实现自适应、自组织功能，而加权系数的调整采用有监督的 Hebb 学习规则。在 3.3 中介绍了连续型 Hebb 学习规则，对于无监督离散型 Hebb 学习，其神经元权系数调整的算式为

$$\Delta w_{ij}(k)=\eta O_j(k)O_i(k)$$

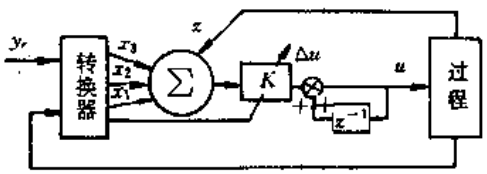


图 5-23 单神经元自适应 PID 控制简图

式中 η 为学习率，$O_j(k)$ 第 j 个神经元的输出值，$O_i(k)$ 第 i 个神经的激活值。上式表明两神经元间权系数的调整与它们的激活值有关。将无监督学习和 Hebb 学习结合起来。可组成有监督的 Hebb 学习规则

$$\Delta w_{ij}(k)=\eta[d_j(k)-O_j(k)]O_i(k)O_j(k) \tag{5-26}$$

上式表明神经元是在教师信号($d_j(k)-O_j(k)$)的指导下进行相关学习和自组织，使相应的输出增强或削弱。

采用式(5-26)所示的有监督 Hebb 学习规则，神经元连接权调整算式为

$$w_i(k+1)=(1-c)w_i(k)+\eta r_i(k) \tag{5-27}$$

$$r_i(k)=z(k)u(k)x_i(k) \tag{5-28}$$

式中 $r_i(k)$——递进信号，$r_i(k)$随过程进行逐渐衰减；

$z(k)$——输出误差信号，$z(k)=y_r(k)-y(k)$，类似于式(5-26)的 $d_j(k)-O_j(k)$；

η——学习速率，$\eta>0$；

c——常数，$c>0$。

将式(5-28)代入式(5-27)后有

$$\Delta w_i(k)=-c\left[w_i(k)-\frac{\eta}{c}z(k)u(k)x_i(k)\right] \tag{5-29}$$

式中 $\Delta w_i(k)=w_i(k+1)-w_i(k)$。

如果存在一函数 $f_i(w_i(k),z(k),u(k),x_i(k))$，则有

$$\frac{\partial f_i}{\partial u_2}=w_i(k)-\frac{\eta}{c}\gamma_i(z(k),u(k),x_i(k))$$

则式(5-29)可写成

$$\Delta w_i(k)=-c\frac{\partial f_i(\cdot)}{\partial w_i(k)} \tag{5-30}$$

上式表明：加权系数 $w_i(k)$的修正按函数 $f(\cdot)$对应于 $w_i(k)$的负梯度方向进行搜索。应用随机逼近理论可以证明：当 c 充分小时，使用上述学习算法，$w_i(k)$可收敛到某一稳定值 w_i^*，且其与期望值的偏差在允许范围内。

为保证上述单神经元自适应 PID 控制学习算法式(5-27)，式(5-30)的收敛性和鲁棒性，对上述学习算法进行规范化处理后：

$$\left.\begin{aligned} u(k) &= u(k-1) + K\sum_{i=1}^{3} w_i(k)x_i(k) \\ w_i'(k) &= w_i(k)/\sum_{i=1}^{3}|w_i(k)| \\ w_1(k+1) &= w_1(k) + \eta_I z(k)u(k)x_1(k) \\ w_2(k+1) &= w_2(k) + \eta_P z(k)u(k)x_2(k) \\ w_3(k+1) &= w_3(k) + \eta_D z(k)u(k)x_3(k) \end{aligned}\right\} \tag{5-31}$$

式中 $x_1(k)=e(k)$；

$x_2(k)=\Delta e(k)$；

$x_3(k)=\Delta^2 e(k)=e(k)-2e(k-1)+e(k-2)$。

η_I、η_P、η_D——积分、比例、微分的学习速率。

这里对积分(I)、比例(P)、微分(D)分别采用了不同的学习速率 η_I、η_P、η_D，以便对它们各自的权系数能根据需要分别进行调整。

其取值由仿真与实验确定，且取 $c=0$。

例：设被控过程模型为

$$y(k)=0.368y(k-1)+0.264y(k-2)+u(k-d)+0.632u(k-d-1)+\zeta(k)$$

应用神经元自适应 PID 控制器进行仿真研究。系统启动时先进行开环控制，$u=0.225$，待输出到达期望值的 0.95 时，神经元控制器投入运行。对比例(P)、积分(I)、微分(D)三部分加权系数调整的学习速率取相同和不同时的仿真结果示于图 5-24。在图 5-24a 中，$\eta=100$，$k=0.02$，$d=10$，到 48 步时的超调整量为 1.35%。在图 5-24b 中 $\eta_P=7000$，$\eta_I=200$，$\eta_D=20$，$d=10$，到 37 步时的超调量为 0.27%。

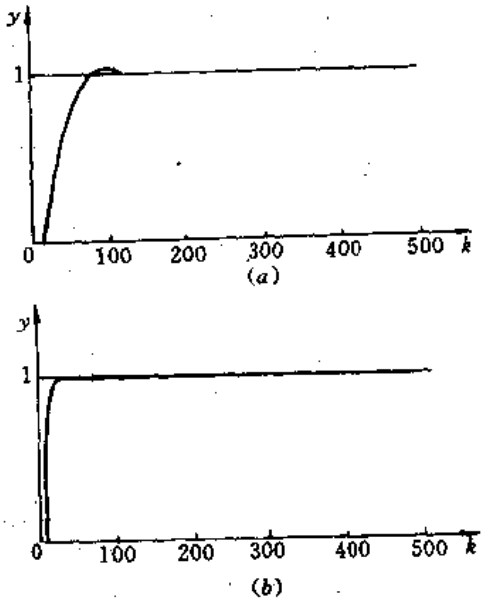


图 5-24 单神经元自适应 PID 控制的仿真曲线
(a)当学习速率 η 相同时；
(b)当学习速率 η 不相同时

仿真结果表明，采用不同学习速率的单神经元自适应 PID 控制器较采用相同学习速率的具有较好的快速性、较小的超调量和较强的鲁棒性。

关于加权系数初值的选择问题，由于神经元有自学习功能，它的赋值并不影响权系数以后的学习结果，也不会影响控制器的控制效果。

K 值的选择非常重要。K 大，则快速性好，但超调量大，甚至可能使系统不稳定。当被控过程时延增大时，K 的值必须减少，以保证系统稳定。K 值选择过小，会使系统的快速性变差。

5.5.2.3 单神经元自适应 PID 控制器学习算法的改进

在大量工程实际应用中，人们通过实践总结出 P、I、D 三个参数的在线学习修正，主要与 $e(k)$ 和 $\Delta e(k)$ 有关。基于此可将单神经元自适应 PID 控制算法式(5-31)中的加权系数学习修

正部分作些修改，即将其中的 $x_i(k)$ 改为 $e(k)+\Delta e(k)$，改进后的算法如下：

$$\left.\begin{aligned}
&\Delta u(k) = K\sum_{i=1}^{3} w_j(k)x_i(k)\\
&w_i'(k) = w_i(k)/\sum_{i=1}^{3}|w_i(k)|\\
&w_1(k+1) = w_1(k) + \eta_I z(k)u(k)[e(k)+\Delta e(k)]\\
&w_2(k+1) = w_2(k) + \eta_P z(k)u(k)[e(k)+\Delta e(k)]\\
&w_3(k+1) = w_3(k) + \eta_D z(k)u(k)[e(k)+\Delta e(k)]
\end{aligned}\right\} \tag{5-32}$$

式中 $\Delta e(k)=e(k)-e(k-1)$；

$z(k)=e(k)$。

采用上述改进算法后，权系数的在线学习修正，就不完全是根据神经网络学习原理，而是参考实际经验制定的。

例：设被控过程的数学模型同上例

$$y(k) = 0.368y(k-1) + 0.264y(k-2) + u(k-d) + 0.632u(k-d-1) + \zeta(k)$$

采用单神经元自适应 PID 控制器的改进算法式(5-32)，对系统进行仿真研究。启动时采用开环控制 $u=0.225$，等输出到达期望值的 0.95 时，投入神经元控制器，其输出波形如图 5-25 所示。仿真时取 $\eta_P=7000$、$\eta_I=200$、$\eta_D=20$，$K=0.05$，$d=10$，到达 36 步时的超调量为 0.22%。

对比图 5-24 与图 5-25 的输出波形显然可见：改进后的学习控制算法具有较小的超调量和较好的控制效果。

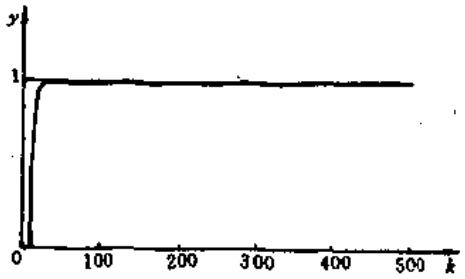


图 5-25 单神经元自适应 PID 控制改进算法仿真曲线

5.5.3 非线性系统自学习控制

人工神经网络理论研究的兴起和发展，为非线性系统理论的研究开辟了一条新途径，为了有效地达到控制目的，人们必须预先掌握被控对象的动态特性，这对有些场合是相当困难的，神经网络控制器是一种特殊形式的自适应控制器，它是在充分吸收人脑控制特性基础上的一种新型的复杂控制器，它具有三个重要特征：(1)大量传感信息的利用；(2)集成处理能力；(3)自适应能力。

与传统控制器的设计思想相仿，神经元控制器的目的在于如何设计一个有效的神经元网络完全代替传统控制器的作用，使得系统的输出跟随系统的期望输出。不失一般性，假设非线性系统不存在纯滞后环节，其动态系统模型为

$$y(k)=f(y(k-1),\cdots,y(k-n),u(k),\cdots,u(k-m)) \tag{5-33}$$

基于神经元控制器设计的主要困难是如何利用已知的误差信息 $e(k)=y_d(k)-y(k)$来调整神经元控制器的权阵，使得 $y(k)\to y_d(k)$。($y_d(k)$为任一 k 时刻系统的期望输出，$y(k)$为系统实际输出)基于神经元控制器的设计可分为两大类：无导师的学习控制器和有导师学习控制器。对于有导师的学习控制器又可以进一步分成两大类：系统模型已知的网络控制和系统模型未知的网络控制。

5.5.3.1 已知动力系统模型的神经元控制

反

如果系统动力学描述方程(5-33)已知，则我们可以选用性能指标函数

$$E=\frac{1}{2}\sum_{k=0}^{s}(y_d(k)-y(k))^2=\frac{1}{2}\sum_{k=0}^{s}E_k$$

这种神经元控制系统构成如图 5-26 所示。

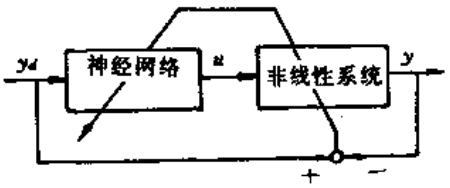


图 5-26 模型已知的网络控制结构图

选用多层感知机网络作为控制器的网络结构，则控制器输出 $u(k)$为神经网络输出层单元激活值。根据 Delta 学习规则神经元权阵修正规则为

$$w_{ij}(k+1)=w_{ij}(k)-\eta\frac{\partial E_k}{\partial w_{ij}}$$

$$=w_{ij}(k)-\eta(y_d(k)-y(k))\frac{\partial y(k)}{\partial w_{ij}(k)} \tag{5-34}$$

$$\frac{\partial y(k)}{\partial w_{ij}(k)}=\frac{\partial y(k)}{\partial u(k)}\cdot\frac{\partial u(k)}{\partial w_{ij}(k)}$$

如果系统模型已知，则$\frac{\partial y(k)}{\partial u(k)}$可以求得，而$\frac{\partial u(k)}{\partial w_{ij}(k)}$可根据广义的 Delta 规则计算得到。从上看出，式(5-34)中运用了系统模型的信息，即把非线性系统看作神经网络的最后一层，对于其它各层权重调整，仍按 Delta 算法逐层计算。这种控制结构实质上是直接逆控制。然而，不幸的是，一旦系统的模型发生变化，这种控制器就失去了原来具有的优点，而使控制偏离了期望轨迹。

5.5.3.2 未知动力学系统模型的神经元控制

在系统模型不确定条件下，可以利用神经网络辨识的方法首先建立初步的动态神经元模型，并在学习过程中进一步改善模型的准确性，达到高精度控制的目的。

对于非线性系统(5-33)，控制器的目的在于希望找到 $u(k)$使得 $y_d(k)\to y(k)$。假定系统逆动力学模型存在，即

$$u(k)=g(y(k),\cdots,y(k-n),u(k-1),\cdots,u(k-m)) \tag{5-35}$$

我们知道，给定的 $u(k)$，其相应的系统输出为 $y(k)$。$u(k)$的作用是迫使 $y(k)$趋向 $y_d(k)$。因此，可以用 y_d 去代替(5-35)中的 y，从而产生其控制器的输出为

$$u(k)=g(y_d(k),\cdots,y_d(k-n),u(k-1),\cdots,u(k-m)) \tag{5-36}$$

这就是直接逆动力学控制的基本思想。由于单纯的逆动力学辨识模型准确度有限，且缺乏自学习能力，从而使系统控制精度很低。图 5-27 给出了多神经网络自学习控制结构图。该系统充分

利用非线性系统输入输出信息，并在此基础上逐步改善系统的辨识精度，同时吸收了逆动力学的控制思想，利用系统模型网络辨识器去构造系统的希望控制值(网络控制器的期望输出)，从而控制非线性系统输出 $y(k)$跟随希望输出 $y_d(k)$。

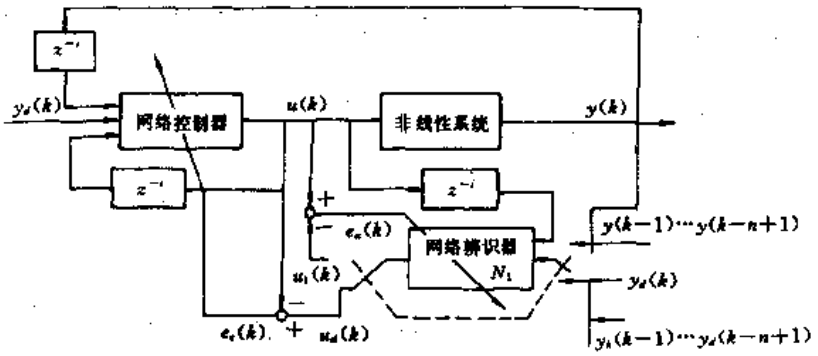


图 5-27　多网络自学习控制结构图

图 5-27 中 N_1 为逆动力学神经网络模型，利用系统输入输出信息$\{u(k),y(k)\}$来实时在线地辨识逆动力学模型 N_1，辨识器的学习规则为

$$\begin{cases}\delta_j^{(1)} = e_n(j) & \text{输出层}\\ \delta_j^{(1)} = O_j^{(1)}(1-O_j^{(1)})\sum_i \delta_i w_{ij}^{(1)} & \text{隐含层}\end{cases}$$

$$w_{ij}^{(1)}(k+1) = w_{ij}^{(1)}(k) - \eta_1\delta_j^{(1)}O_i^{(1)} + \alpha_1\Delta W_{ij}^{(1)}(k) \tag{5-37}$$

同时，利用此逆动力学模型和系统期望输出 $y_d(k)$去逼近一个期望的控制量 $u_d(k)$，从而达到更新控制器 N_2 的连接权系数的目的。网络控制器 N_2 权阵的学习规则为

$$\delta_j^{(2)} = e_c(j) \qquad \text{输出层}$$

$$\delta_j^{(2)} = O_j^{(2)}(1-O_j^{(2)})\sum_i \delta_i^{(2)} w_{ij} \quad \text{隐含层}$$

$$w_{ij}^{(2)}(k+1) = w_{ij}^{(2)}(k) - \eta_2\delta_j^{(2)}O_i^{(2)} + \alpha_2\Delta w_{ij}^{(2)}(k) \tag{5-38}$$

η_1,η_2 分别为两个网络的学习因子，α_1,α_2 为一常数。

例：考虑如下单输入一单输出非线性离散系统

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

为了构成基于逆动力学模型的网络控制方法，首先要对实际系统进行离线逆动力学辨识，得到逆动力学模型

$$u(k) = g(y(k),y(k-1),y(k-2),y(k-3),u(k-1))$$

在辨识模型中，神经网络 $N_1(\pi^3_{5,25,12,1})$用于逼近非线性函数 $g(\cdot)$，经过 1000 步迭代，辨识模型的精度已控制在 5%以内，基于这个精度的逆模型多网络控制器可以满足整个网络控制系统在初始的控制器训练学习时稳定运行。此外，考虑到控制网络 N_2 初始权系数是随机选取的，为了保证控制输出尽可能不要太长，在初始权阵选择时尽量取小的随机数，取 N_2 初始权系数为$[-0.5,0.5]$之间的随机数。控制网络N_2的结构取为$\pi^3_{6,20,10,1}$，输入矢量为$[y_d(k),y(k),y(k-1),y(k-2),y(k-3),u(k-1)]$，输出量 $u(k)$，取控制网络学习训练时的系统期望输出为

$$y_d(k) = \sin\frac{2\pi k}{100} \qquad k = 0,1,\cdots,100$$

则学习训练开始时系统的响应曲线如图 5-28(a)所示。随着网络控制器学习的进行，系统的实际输出越来越接近系统的期望输出，经过 100 个周期的学习后，其跟踪精度已控制在 1%以内，系统的输出响应曲线如图 5-28(b)所示，实线表示系统希望的输出，虚线表示实际的系统输出。多网络控制自学习系统具有以下特点：

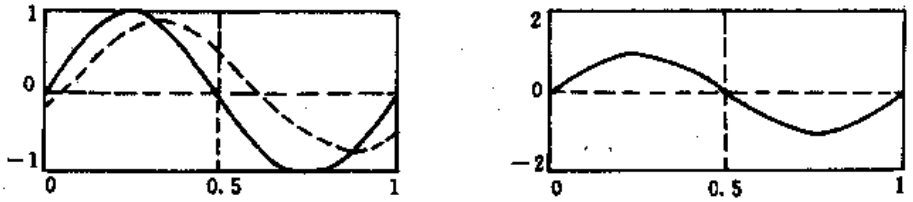


图 5-28　多网络控制响应曲线

(1)边控制边对逆动力学模型进行在线辨识和校正，利用实际系统的输入输出信息去更新逆动力学神经网络的权矩阵，达到对系统的准确识别。

(2)利用期望轨迹 y_d，通过逆动力学网络来产生期望的控制量，从而使广义 Delta 学习规则成为可能。

(3)由于网络具有在线辨识和控制的能力，因此多网络结构控制器完全能满足环境参数变化下的控制性能要求，具有很好的自适应、自学习能力。

(4)如果非线性系统的逆映射是多值的，那么基于逆模型的神经网络自学习控制难以实行，这时可用正向网络模型学习法。

5.6　神经控制应用

在过去的近几年中，神经网络在复杂动力学系统控制和辨识的应用引起了普遍的关注，神经网络基于自身的结构和特征特别适合解决复杂的、非线性和非确定性的系统问题。

5.6.1　神经预测

复杂耦合非线性系统的控制无法单独利用常规控制方法来实现，往往需要人的智能经验。在以石化企业为代表的复杂连续系统控制中，这类问题尤为明显。一方面过程体系庞大，反应繁多，机理模型在现场检测信号数量小和低精度下经常被迫简化到不足以反映系统性能的程度；另一方面，操作人员根据经验控制的能力有限，控制质量难以提高。

茂石化二重整装置的目的是将原料油中的烃类，在催化剂的作用下进行环烷脱氢。重整过程化学反应繁多，具有高度的耦合非线性特征，控制系统多采用单回路 PID 控制。由于原料油品种变化频繁，进料芳烃波动很大，由此造成产品稳定油的辛烷值也有较大的波动，操作人员的控制依据只是每天一次的产品稳定油芳含化验和每月一次的进料芳潜化验，因此只能根据粗略的估算得到仪表设定值。控制不可能达到准确。控制目的是建立以芳含为控制目标的静态芳含控制模型，以便及时准确地调整仪表给定值。

5.6.1.1 预测模型的建立

优化稳定油芳含设定值的关键是建立静态芳含模型。产品稳定油的芳含主要受6个因素的影响：(1)进料质量；(2)反应温度；(3)装置压力；(4)空速；(5)氢油化；(6)催化剂性能。建立预测模型的目标是进行高频预测(5min1次)，并通过实际的低频采样值不断校正，以跟踪拟合现时的系统状况，这一模型是用很强自组织、自学习能力的神经网络来建立的，但我们知道一般BP网络的泛化能力与样本的选择和网络的结构选择很有关系，通常情况下没有确定的选择依据，因而学成的网络很难进行外推，即出现样本外的输入时其输出与实际值可能有较大的偏差，为此采用一种隐节点校正(HNR)网络来部分地解决这个问题，应用在这套稳定芳含设定值优化控制系统中取得了一定的成效，从而保证了预测值的准确性。

5.6.1.2 HNR算法

我们知道，多层前馈网络(MLN)的学习性能在很大程度上取决于隐节点的数目。理论上讲，任何一个定义在紧致集上的连续多元函数 $f(x_1,x_2,\cdots,x_p)$ 都可以用一个三层网络来实现，即：对于任意上述定义的 $f(x_1,x_2,\cdots,x_p)$ 我们都可以构造一个函数

$$\bar{f}(x_1,x_2,\cdots,x_p)=\sum_{i=1}^{n}b_i\psi\left(\sum_{j=1}^{p}a_{ij}x_j+\theta_i\right)$$

或

$$\bar{f}(x_1,x_2,\cdots,x_p)=\sum_{i=1}^{n}b_iz_i \tag{5-39}$$

$$z_i=\psi\left(\sum_{j=1}^{p}a_{ij}x_j+\theta_i\right)\qquad i=1,2,\cdots,n \tag{5-40}$$

对任意给定的误差限 ε，都存在函数 ψ，常数 N 和系数 b_i,a_{ij},θ_i，当 $n>N$ 时，有

$$\max|f(x_1,x_2,\cdots,x_p)-\bar{f}(x_1,x_2,\cdots,x_p)|<\varepsilon$$

从这个定理出发，根据(5-39)，(5-40)式分两步来构造 $\bar{f}(x_1,x_2,\cdots,x_p)$：

(1)找到 ψ,a_{ij},θ_i，使由(5-40)式得到的 z_i 与 $\bar{f}(x_1,x_2,\cdots,x_p)$ 线性相关；

(2)找到 b_i，使满足(5-39)式。

这就是HNR算法的实质。HNR算法步骤如下：

给定 m 组样本 $(x_{i1},y_1),(x_{i2},y_2),\cdots,(x_{im},y_m)\quad i=1,2,\cdots,p$

(1)定义复相关系数 $\rho(z,y)$，令

$$E=\frac{1}{2}(\rho(z,y)-1)^2$$

寻优 a_{ij}，使 E 收敛。

其中复相关系数 $\rho(z,y)$ 定义如下：

$$y=\psi+\sum_{i=1}^{n}b_iz_i$$

且有 m 个样本 $(y_1,z_{i1}),(y_2,z_{i2}),\cdots,(y_m,z_{im}),i=1,2,\cdots,n$，则

$$\rho(y_1,z_i)=\frac{1}{L_{00}}\sum_{i=1}^{n}L_{i0}b_i$$

其中

$$L_{00}=\sum_{l=1}^{m}(y_l-\bar{y})^2$$

$$\bar{y}=\sum_{l=1}^{m}y_l/m$$

$$L_{i0}=\sum_{l=1}^{m}(z_{il}-\bar{z}_i)(y_l-\bar{y})$$

$$\bar{z}_i=(\sum_{l=1}^{m}z_{il})/m \qquad i=1,2,\cdots,n$$

(2)根据(1)步的a_{ij}结果，由(2)式计算出的$z_{il}(i=1,2,\cdots,n,l=1,2,\cdots,m)$和样本给定的$y_m$，利用最小二乘法回归系数$b_i$，并计算相应的相关系数$\rho'$；

(3)重复(1)，(2)步，直到ρ'收敛。

HNR 算法的优点在于它的第二步采用最小二乘算法，寻优过程仅限于对a_{ij}的优化，因此大大减少了收敛步数，仿真实验表明 HNR 的收敛速度比原始 BP 算法快 10^2 数量级，同时还表现出较好的泛化能力。

芳含预测模型的输入量为：入口温度，三个反应器温降，空速，压力，氢油比，以 20 个样本为学习单位，网络结构取 7-8-1。芳潜模型的输入量为：入口温度，三个反应器温降，空速，以 8 个样本为学习单位，网络结构取 5-7-1。两个模型均采用 HNR 网络实现，经 HND 算法训练，其收敛步数分别约为 385 和 183，用同样的样本训练 BP 网络，收敛步数分别为 76995 和 56719，两个模型训练时的收敛精度均取 0.01。

系统学习完成后进行了芳含预测，数据列于表 5-1。可以看到芳含预测的绝对误差在 5% 以内。现场造成误差的原因很多，需进一步进行分析，以能更好地预测。

表 5-1

入口温度 (C°)	压力 (MPa)	氢流量 (km^3/h)	· · ·	芳含实测(%)	芳含预测(1%)
496	1.1	60		52.88	52.97
497	1.1	60		51.49	49.07
⋮			· · ·		
497	1.1	55		53.44	52.97
496	1.1	55		55.40	54.69

5.6.2 工业过程故障诊断

当过程中的故障发生时，应该尽可能早地检测故障。故障诊断系统必须能够指出过程中发生了什么故障。通常采用的故障检测和诊断技术有两大类：估计方法和模式识别。

5.6.2.1 估计方法

估计方法需要建立实际过程的数学模型。然而，模型不能太复杂，因为过于复杂的模型导

致计算时间过长。

(1)基于状态变量估计的故障检测。

在过程的状态变量中,很少的变量是可测量的,所以必须估计不可测量的状态变量。可以依据模型的随机特性采用不同的估计,然后根据估计变量和实际测量变量之间的偏差进行故障检测。通常采用统计实验方法,这种方法需要了解线性化系统的相当准确的参数,除此之外,过程必须运行在线性化的工作点附近,否则模型就不能准确地描述系统动态行为。

(2)基于参数估计的故障检测。

在许多场合下,过程的模型参数对物理过程系数有一个很复杂的关系,不正常的状态通常影响这些物理系数并且这些影响可以从过程参数中反映出来。由于不是所有的物理过程参数是直接可测量的,所以只能通过估计过程参数来计算出它们的变化。

5.6.2.2 模式识别方法

故障检测和诊断的模式识别方法不需要过程的数学模型。其基本思想是根据测量数据把过程的运行状态分类。实质上这是从测量空间到决策空间的映射。

传统的模式识别和分类分为三个阶段:测量,特征提取和分类。首先采集适当的数据,然后计算特征向量,特征提取应该把多余的信息从测量数据中去除并且建立一个决策平面。最后这个特征向量被分为一个或多个类别。当测量和诊断同时进行,这些类别为:正常运行,故障Ⅰ,故障Ⅱ等等。传统模式识别方法着重于找出特征的分类。问题在于如何计算这些特征。一般难以知道哪些特征是重要的,哪些是无关的。适当的特征选择导致简单的分类规则;反之,不适当的特征选择使得分类规则相当复杂。

人类具有模式识别的能力。某人可以列举出某类事物的例子但难以说出他的分类规则。像人类一样,神经网络可以用一组例子进行训练。当神经网络实现了分类,那么同时就完成了测量空间到决策空间的映射。

尽管目前神经网络还达不到像人那样的能力,但提供了很有前途的方法进行模式识别和分类。神经网络把传统模式识别方法的步骤结合为一体,因为它能够自动地抽取特征。由于它的非参数性,神经网络是一种实用方法,研究表明神经网络分类器的精度优于传统方法。

(1)催化过程故障检测和诊断。

这个过程包括一个热交换器和一个连续搅拌反应器,过程的原理示意图由图 5-29 给出。图中圆圈内的符号意义附后。

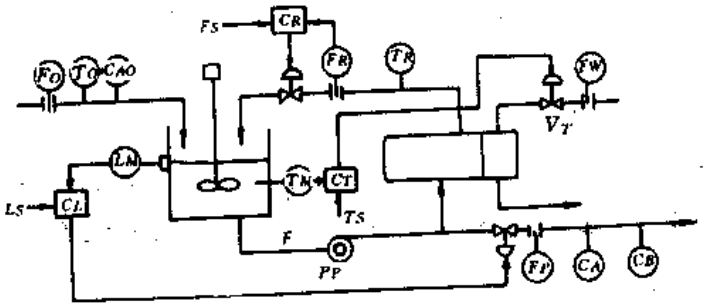


图 5-29 催化过程原理示意图

这是一个催化放热过程，反应器流出液流经热交换器，在热交换器中流出液被外部提供的冷水冷却，然后把冷却了的流出液返送回反应器来控制反应器的温度。这个过程有三个闭环控制回路，每个回路采用 PI 控制器。这些控制器保持循环速率，反应器的温度和反应器水位不变。选择这个过程的目的是由于它代表一大类工业过程的特征。

过程的被测变量有 14 个，有 10 个被测故障点，对每一个故障点存储 14 个测量数据用于训练神经网络。

10 个故障点为：(1)输入管道部分堵塞；(2)回流管道部分堵塞；(3)输入浓度高；(4)回流设定位置高；(5)污物雍塞的热交换器；(6)不活跃催化剂；(7)温度控制阀放置太高；(8)反应器泄漏；(9)回流流量计放置太高；(10)泵工作不正常。数据测量主要是静态，但在某些情况下，变化太慢，数据采集发生在进入稳态之前。所有的测量数据被归一化处理，位于−1 和+1 之间，这种预处理使网络易于学习，因为原始数据幅值大小不一，相差太大。例如反应器水位高度是 3m，而输入浓度为 1200mol/m³，如果不进行处理，大的测量值的波动就垄断了神经网络的学习过程，使得不能反应小的测量值变量的变化。

多层感知机网络用于故障检测和诊断。输入层包含 14 个神经元，隐层 4 个单元和输出层 10 个单元，输入层 14 个单元代表 14 个测量值；输出层 10 个单元表示 10 个故障点。sigmoid 函数用于隐层和输出层神经元，训练数据包括 440 个测量模式，其中 40 个正常工作模式，每一个故障有 40 个测量模式。在正常工作模式下，网络所有的输出应该是零，故障 I 时，网络输出层第一个单元应该是 1，其余是 0，以此类推，在网络实验阶段(训练完成之后)，如果网络的所有输出小于 0.5，被认为是正常工作模式；如果只有一个输出大于 0.5，那么故障就被诊断出了。

训练迭代次数为5000次。初始网络权重随机分布在$[-0.1,+0.1]$之间，学习速率$\eta=0.95$，记忆系数 $\alpha=0.6$。研究发现，改变学习参数(η,α)就不能大大加速学习过程，但是在隐层用双曲正切函数作为非线性函数可以大大改变这个情况，学习过程快得多。研究还发现，增加隐层节点可以改善训练速率，但是不知为什么网络对训练集之外的模式分类能力变差了，即泛化 能力差了。隐层节点数必须足够大以构成给定问题的复杂决策域，另一方面，隐层节点也要足够小使得网络有良好的泛化能力。

两个隐层的网络使得训练变慢了。例如，14-6-6-10 型的多感知机网络(用 sigmoid 函数)需要 8000 次训练，这是因为层增加之后，需要调节的权也增加了。

实验证明多层感知机网络可以成功地用于过程的静态故障诊断。如果某个故障突然发生，要 求尽快检测出这个故障。要解决这个问题，需要采集各故障点的动态模式，用动态模式来训练神经网。那么，对于这一过程，故障点数目多，模式也很多，单一网可能难以实现故障诊断，还须形成新的网络结构。

图 5-29 催化过程示意图中符号说明：

F_O——输入流体速率；
T_O——输入温度；
C_{AO}——热交换器输入浓度；
L_M——罐中液体高度；
C_B——反应器浓度；
C_R——回流浓度；
F_R——回流速率；
F_P——输出液体速率；
T_R——回流温度；
C_A——热交换浓度；
F_W——冷却水流速；
T_M——反应器温度。

5.6.3 批量进料发酵过程适应控制

发酵是一个缓慢的过程。在这个过程中，微生物吸收提供的养料繁殖，生物量保持和成形。在大部分工业现场，发酵过程在无菌的情况下进行，置于装备有搅拌，供气系统及温度、PH值、分解氧浓度控制系统的发酵罐内。在批量进料发酵过程中，发酵过程从初始投放量(微生物和培养基浓度)开始，培养基不断地放入发酵罐中，直至总量达到最大容许水平。

这个过程由以下动力学方程表示：

$$\dot{x} = f(x,u,P) + \epsilon(x), \quad x(0) = x_0$$

这里 $x(t)$ 是 t 时刻状态，$x(t) \in R^5$，在 t 时刻的控制量 $u(t)$，$u(t) \in R^2$，参数向量 $P \in R^L$，过程未知部分用 ϵ 表示。状态变量 $x_j(\cdot)$ $j=1,2,\cdots,5$ 定义如下：

x_1——微生物浓度；

x_2——培养基浓度；

x_3——抑制物浓度；

x_4——乙烷浓度；

x_5——发酵物容积。

控制量 $u_1(\cdot)$ 和 $u_2(\cdot)$ 定义为

$u_1(\cdot)$——培养基进料量；

$u_2(\cdot)$——施加培养基浓度。

过程从零时刻开始，终止在时刻 T。T 一般从几小时到几天。对于这个问题，$T=15$h。

5.6.3.1 过程模型

基于 baker 发酵过程的动态描述为

$$\dot{x}_1 = \mu_m \frac{x_1 x_2}{k_s + x_2} \frac{1}{1 + x_3^2} - \frac{x_1}{x_5} u_1 + 0.48 x_1 \xi_1(x) \tag{5-41}$$

$$\dot{x}_2 = \frac{-\mu_m}{k_y} \frac{x_1 x_2}{k_s + x_2} \frac{1}{1 + x_3^2} + \frac{u_2 - x_2}{x_5} u_1 - k_m x_1 - \frac{1}{0.51} x_1 \xi_2(x) \tag{5-42}$$

$$\dot{x}_3 = 0.0023 x_1 + 0.007 \mu_m \frac{x_1 x_2}{k_s + x_2} \frac{1}{1 + x_3^2} - \frac{x_3}{x_5} u_1 \tag{5-43}$$

$$\dot{x}_4 = x_1 [\xi_2(x) - \xi_1(x)] - \frac{x_4}{x_5} u_1 \tag{5-44}$$

$$\dot{x}_5 = u_1 \tag{5-45}$$

式中 μ_m，k_s，k_y 和 k_m 为参数向量 P 的元素，即 $P = [\mu_m \quad k_s \quad k_y \quad k_m]^T$。非线性 $\xi_1(\cdot)$ 和 $\xi_2(\cdot)$ 对应于乙烷消耗和产生速率，通常是未知的，表示系统未知部分。如果没有关于 $\xi_1(\cdot)$ 和 $\xi_2(\cdot)$ 的知识，进行适当的控制是很困难的。幸好，已知乙烷生成从培养基浓度 x_2^* 开始，关于它们的先验知识为

(1)当培养基浓度$\leqslant x_2^*$，则 $\xi_2(x)=0$，$\xi_1(x)=0$；

(2)ξ_1，ξ_2 为非负值；

(3)这个过程的特点是微生物可能消耗或产生乙烷，这意味着当 $\xi_1(x)>0$ 时，$\xi_2(x)=0$，反之亦然。

控制目的是确定 $u_1(\cdot)$和 $u_2(\cdot)$在$[0,T]$时间的最大量地产生微生物，同时乙烷的浓度保持低水平。控制的困难在于过程的非线性和不确性因素的存在。

传统发酵过程采用两个 PI 控制器，这些控制器不能达到理想的性能，某些形式的适应控制是很必要和重要的。

线性自适应控制的控制性能较差，而且对设计参数变化很敏感。当不能得到足够的先验知识的情况下，构成非线性控制器可采用的方法之一就是神经网络控制器。如前所述，系统有 5 个状态变量，其中 4 个 (x_1，x_2，x_3 和 x_5)用来产生控制输入，3 个(x_1，x_2，和 x_3)用来产生适应控制律。研究表明多层神经网络 MNN 具有较高的精度，而且这种网络既适用于非线性系统的辨识，也适用于未知非线性系统的控制。

在式(5-41)—(5-45) 中，假设式(5-41)中 x_1u_1/x_5 和式(5-42)中$(u_2-x_2)u_1/x_5$ 是已知的，则式(5-41)和(5-42)可写成下列形式：

$$\dot{x}_1 = f_{10}(x,p) - \frac{x_1}{x_5}u_1$$

$$\dot{x}_2 = f_{20}(x,p) + \frac{u_2 - x_2}{x_5}u_1$$

f_{10}和 f_{20}是未知函数，我们的目的是确定 u_1 和 u_2 以使 x_1，x_2 跟踪理想输出 $x_1^*(t)$和 $x_2^*(t)$。如果取 u_1 和 u_2 为

$$u_1 = \frac{x_5}{x_1}[f_{10}(x,p) - f_1^*(x^*,p^*) + \lambda_1 e_1] \tag{5-46}$$

$$u_2 = x_2 + \frac{x_5}{u_1}[f_{20}(x,p) - \lambda_2 e_2] \tag{5-47}$$

则误差方程有 $\dot{e}_1=\lambda_1 e_1$ 和 $\dot{e}_2=\lambda_2 e_2$ 的形式，所以理想的响应依赖于 λ_1 和 λ_2 的选择。式中 $e_j=x_j-x_j^*$ $(j=1,2)$。如果用神经网络在线估计未知的非线性函数，就可以得到误差 e 的模型

$$\dot{e}_1 = -\lambda_1 e_1 + f_{10}(x,p) - \hat{f}_{10}(x,\theta) \tag{5-48}$$

$$\dot{e}_2 = -\lambda_2 e_2 + f_{20}(x,p) - \hat{f}_{20}(x,\theta) \tag{5-49}$$

这里 $\hat{f}_{10}(x,\theta)$和 $\hat{f}_{20}(x,\theta)$用神经网络来实现。由于发酵过程是时变的，要确定神经网络的权基于什么来调节。因为未知非线性系统位于输出和可调节权之间，基于式(5-48)(5-49)的误差 e_1 和 e_2 可用来调节神经网的权。在式(5-46)和(5-47)中，函数 $f_{10}(\cdot)$的估计为

$$\hat{f}_{10}(x,p) = N_1(x,\theta_1) + k(t)x_1$$

$$\hat{f}_{20}(x,p) = -N_2(x,\theta_2) - 10000\int_0^t e_2(\tau)\mathrm{d}\tau$$

这里 $\theta=[\theta_1^T\ \theta_2^T]$，$k(t)$被限于$[0,0.3]$内，$k=100\,e_1x_1$ 并且 $k(0)=0.3$。

5.6.3.2 信号选择和处理

选择网络的输入，输出信号以及信号的处理对控制律的执行是十分重要的。取 x_1，x_2 和 x_3 作为网络的输入，网络输出为 $N_1(x,\theta_1)$和 $N_2(x,\theta_2)$。多层神经网络 MNN 具有 sigmoid 非线性函数，由于这些输入信号的饱和值大于1，网络的输入应压缩到$[0,1]$区间。压缩因子对

$x_1(t)$,$x_2(t)$,$x_3(t)$分别为100,1,1。网络的初始权值取[0,0.01]内的随机值。网络$N_1(\cdot)$和$N_2(\cdot)$都有4个输入单元(压缩了的$x_1(t)$,$x_2(t)$和$x_3(t)$以及阈值)和一个输出单元。在大量的仿真实验之后,给出三层神经网络(其隐层含有10个神经元)作为合适的网络结构)。

为了确定神经网络权值调节律,取性能准则为

$$I_j(\theta_j)=\frac{1}{T}\int_0^t e_j^2(t,\theta_j)\mathrm{d}t \quad (j=1,2)$$

所以

$$\frac{\partial I_j}{\partial \theta_{ji}}=e_j\frac{\partial e_j}{\partial \theta_{ji}} \quad i=1,2,\cdots,n_{pj};\quad j=1,2$$

这里整数$n_{pj}(j=1,2)$表示每一个神经网络权值总数目,网络的权值用梯度法调节。

$$\theta_{ji}=-r_{ji}e_j\frac{\partial e_j}{\partial \theta_{ji}} \qquad i=1,2,\cdots,n_{pj},j=1,2$$

这里r_{ji}是适应增益,上式中偏微分是用BP算法产生的,权的初始值取[0, 0.1]内随机值:网络权值的调节律为

$$\theta_{ji}(k+1)=\theta_{ji}(k)-\eta_{ji}e_j(k)\frac{\partial e_j(k)}{\partial \theta_{ji}(k)} \qquad i=1,2,\cdots,n_{pj},j=1,2$$

这里η_{ji}是步长,对第一个网络步长$\eta_{1i}=0.25\times10^{-6}(i=1,2\cdots,n_{p1})$,对第二个网络步长$\eta_{2i}=0.025$ $(i=1,2,\cdots,n_{p2})$式(5-46)和(5-47)中$\lambda_1=\lambda_2=20$。

MNN神经网的响应如图5-30所示。文献[8]比较了线性自适应控制器,RBFN神经网络非线性控制器和MNN网络非线性控制器的效果,结果表明:

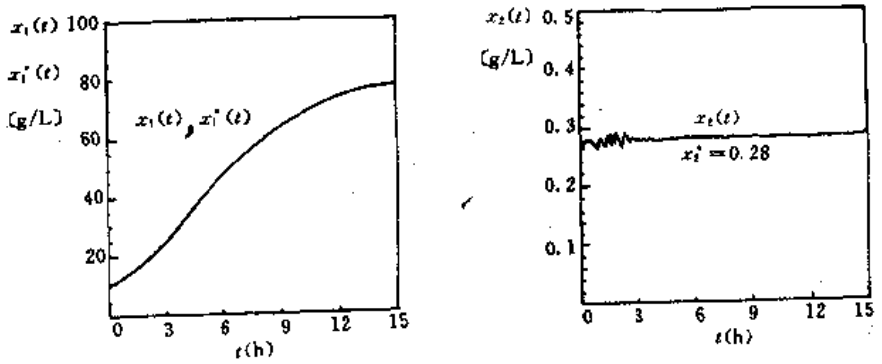


图5-30 MNN自适应控制器下的过程响应

(1)当变量的变化在线性化模型工作点附近时,线性自适应控制可以达到满意的控制性能。

(2)在非线性自适应控制器中,由于没有关于系统未知模型的先验知识,当系统未知部分的影响显著时,就会产生较大的控制误差。然而,神经网络MNN自适应控制器在线辨识系统的非线性(包括未知模型部分),所以能够覆盖大范围过程参量的变化。

对于发酵过程控制,是否选用神经网络控制器取决于以下几个因素:(1)设计者期望的参数范围和参数偏差理想值的初始状态;(2)设计者所容许的误差范围以及对不同非线性的先验知识。文献[8]认为,在大多数场合,线性自适应控制器和MNN神经自适应控制器是仅供选择的两种方法。在这两种方法中,如果控制精度和鲁棒性是至关重要的,那么神经网络自适应控制器具有明显的优势。

第 6 章　神经网络在模糊控制中的应用

将神经网络(Neural Network)和模糊控制(FC)相结合，构成模糊-神经网络系统(FNNS——Fuzzy Neural Network System)，是当前颇受人们关注的最新颖控制策略研究方向之一，并且已取得不少理论研究成果。它们两者均无需依赖于被控对象模型，其学习和推理功能十分相似于人类学习和知识推理过程。模糊-神经网络系统通常由模糊控制器和一个神经网络组成。其中，模糊控制器采用模糊推理规则，以模仿人在不确定性环境下的决策行为，但要从经验中自动产生规则，并修改其控制决策的自学习功能较为薄弱。神经网络的引入为模糊控制器提供了一种良好的学习功能，为模糊控制系统增强学习功能，给深入研究和广泛应用提供了极大可能性。

6.1　模糊集理论简介

模糊集是 Zadeh 于 1965 年引入的。其目的是作为描述与处理广泛存在的不精确的、模糊的事件的工具。该理论经过不断的发展，已形成了有关纯数学与应用数学的许多分支，其中包括拓扑学、图论、映射、自动控制和模式识别等，它可表示语言的信息，如“许多”、“很少”、“经常”、“有时”等等，并可度量某种模式(事物)出现的程度。

传统的集合论只描述有清晰界限的事物，即这种事物是否出现，而不会在是否出现之间存在一个中间地带。利用概率论可以说明某种事物将出现的可能性。但是，这与事物的模糊度量在性质上是不同的。例如用概率来说明白马是否出现，它只描述白马是出现还是不出现。而模糊理论则是描述这马是纯白的、灰白的还是半黑半白的，等等。

下面将简要介绍模糊集理论的几个基本概念及基本运算，以及在信息处理方面，它与人工神经网络的异同，以便为后面的模糊神经网络的引入打下必要的基础。

6.1.1　隶属概念

在一个参考集 E 中，我们用隶属函数 $\mu_A(x)$ 来表征一个模糊子集 A，$\mu_A(x)$ 是在一个有序的集 M 中取值的。M 为区间[0,1]，称为“隶属集”。因此，E 中的模糊子集 A 是一个有序对的集合，表示为

$$A = \{x \mid \mu_A(x)\} \qquad \forall x \in E$$

$\mu_A(x)$ 描述 x 属于 A 的程度，称为隶属函数。若取 $M=\{0,1\}$，则 A 变为一个普通的集合。在一般情况下，E 中的一个元素 x 并非“属于”$A(\mu_A(x)=1)$ 或“不属于”$A(\mu_A(x)=0)$。x 属于 A 的程度一般是逐渐转变的。由此可见，模糊集理论是传统的集合论的一种推广。

对于 $E=\{x_i\}_1^N=1$，则模糊集 A 可表示为

$$A = \sum_i (x_i \mid \mu_A(x_i)) \quad x_i \in E$$

这里，运算“+”是指逻辑加。

当 E 为连续域时，A 可写成

$$A=\int_E x|\mu_A(x)$$

隶属函数 $\mu_A(x)$ 不是随意给定的，是根据人的评判而确定的，因而是属主观指定的。

6.1.2 模糊子集的简单运算

设 A 和 B 是在 E 中的两个模糊子集，则一些基本的定义与运算，如相等、包含、交、并、补的定义如下：

(1)相等($A=B$)

$\mu_A(x)=\mu_B(x) \quad \forall x\in E$

(2)包含($A\subset B$)

$\mu_A(x)<\mu_B(x) \quad \forall x\in E$

(3)并集($A\cup B$)

$\mu_{A\cup B}(x)=\max(\mu_A(x),\mu_B(x)) \quad \forall x\in E$

(4)交集($A\cap B$)

$\mu_{A\cap B}(x)=\min(\mu_A(x),\mu_B(x)) \quad \forall x\in E$

(5)补集(A^c)

$\mu_{A^C}(x)=1-\mu_A(x) \quad \forall x\in E$

6.1.3 模糊向量及其笛卡尔乘积

若对任意的 $i(i=1,2,\cdots,n)$ 都有 $a_i\in[0,1]$，则称向量

$$\boldsymbol{a}=(a_1,a_2,\cdots,a_n)$$

为模糊向量。

对有限论域 $X=\{x_1,x_2,\cdots,x_n\}$ 上的模糊子集 A，令 $a_i\triangleq\mu_A(x_i)$ $(i=1,2,\cdots,n)$，则可用模糊向量方便地表示 X 上的一个模糊子集。

设有两个模糊向量 $\boldsymbol{a}\in M_{1\times n}$ 和 $\boldsymbol{b}\in M_{1\times m}$，其中 $M_{1\times n}$ 为全体 $1\times n$ 维模糊向量，$M_{1\times m}$ 为全体 $1\times m$ 维模糊向量。

定义 $\boldsymbol{a}$ 与 $\boldsymbol{b}$ 的笛卡尔乘积如下：

$$\boldsymbol{s}=\boldsymbol{a}\times\boldsymbol{b}=\boldsymbol{a}^T\circ\boldsymbol{b}$$

其中，运行“$\circ$”的定义为：

$$s_{ik}=\min(a_i,b_k) \quad 1\leqslant i\leqslant n, \quad 1\leqslant k\leqslant m$$

或简记为

$$s_{ik}=a_i\wedge b_k \quad 1\leqslant i\leqslant n, \quad 1\leqslant k\leqslant m$$

其中“$\wedge$”为取小运算“min”。

例：$\boldsymbol{a}=(0.2,\ 0.6)$，$\boldsymbol{b}=(0.7,\ 0.4,\ 0.1)$

$$\boldsymbol{a}\times\boldsymbol{b}=\boldsymbol{a}^T\circ\boldsymbol{b}=\begin{bmatrix}0.2\\0.6\end{bmatrix}\circ[0.7,\ 0.4,\ 0.1]$$

$$=\begin{bmatrix}0.2\wedge0.7 & 0.2\wedge0.4 & 0.2\wedge0.1\\0.6\wedge0.7 & 0.6\wedge0.4 & 0.6\wedge0.1\end{bmatrix}=\begin{bmatrix}0.2 & 0.2 & 0.1\\0.6 & 0.4 & 0.1\end{bmatrix}$$

6.1.4 模糊矩阵与模糊关系

若对任意的 $i \leqslant n$ 及 $j \leqslant m$ 都有 $r_{ij} \in [0,1]$，则称 $R=(r_{ij})_{n\times m}$ 为模糊矩阵。

模糊矩阵的基本运算，如并、交、补运算的定义如下：

对任意的 $R,S \in M_{n\times m}$，$M_{n\times m}$ 为全体 n 行 m 列的模糊矩阵，设 $R=(r_{ij})_{n\times m}$，$S=(s_{ij})_{n\times m}$，则 R 与 S 的并为

$$R \cup S = (r_{ij} \vee s_{ij})_{n\times m}$$

R 与 S 的交为

$$R \cap S = (r_{ij} \wedge s_{ij})_{n\times m}$$

R 的补为

$$R^c = (1 - r_{ij})_{n\times m}$$

其中，$\vee$ 表示取大，$\wedge$ 表示取小。

值得注意的是，模糊矩阵不满足互补律，即 $R \cup R^c \neq E$，E 为全矩阵，即其元素均取1值；$R \cap R^c \neq \mathbf{0}$，$\mathbf{0}$ 为零矩阵。

两个模糊矩阵可以合成。模糊矩阵的合成定义如下：

设两个模糊矩阵 $Q=(q_{ij})_{n\times m}$，$R=(r_{ik})_{m\times l}$，它们的合成 $Q \circ R$ 生成一个 n 行 l 列的模糊矩阵 S，S 的元素 s_{ik} 定义如下：

$$s_{ik} = \bigvee_{j=1}^{m} (q_{ij} \wedge r_{ik}) \quad 1 \leqslant i \leqslant n,\ 1 \leqslant k \leqslant l$$

其中，$\vee$ 表示取大，$\wedge$ 表示取小。

合成的规则实际上与普通矩阵的乘运算类似，只需将普通矩阵的乘运算改为取小运算，将加法运算改为取大运算即可。

合成运算满足以下规律：

(1) $(Q \circ R) \circ S = Q \circ (R \circ S)$

(2) $(Q \cup R) \circ S = (Q \circ S) \cup (R \circ S)$

$S \circ (Q \cup R) = (S \circ Q) \cup (S \circ R)$

但有

$(Q \cap R) \circ S \neq (Q \circ S) \cap (R \circ S)$

$S \circ (Q \cap R) \neq (S \circ Q) \cap (S \circ R)$

(3) $\mathbf{0} \circ R = R \circ \mathbf{0} = \mathbf{0}$

$1 \circ R = R \circ 1 = R$

其中，$\mathbf{0}$ 为零矩阵，1 为单位阵。

(4) 若 $Q \subset R$，则 $Q \circ S \subset R \circ S$。

值得注意的是，合成运算不满足交换律，即

$$Q \circ R \neq R \circ Q$$

模糊关系是普通关系的推广。普通关系描述元素之间是否有关联，而模糊关系则可描述元素之间关联程度的多少。

设 X、Y 是两个非空集合，则直积

$$X \times Y = \{(x,y) \mid x \in X, y \in Y\}$$

中的一个模糊子集R称为从X到Y的一个模糊关系，模糊关系R由其隶属函数$\mu_R: X\times Y\to[0,1]$来描述。序偶(x,y)的隶属度$\mu_R(x,y)$可表示(x,y)具有关系R的程度。当论域X和Y都是有限集时，模糊关系R可以用模糊矩阵R来表示。例如，设$X=\{x_1,x_2,\cdots,x_n\}$，$Y=\{y_1,y_2,\cdots,y_m\}$，模糊矩阵R的元素r_{ij}表示在论域X中第i个元素x_i与论域Y中的第j个元素y_j对于关系R的隶属程度，即

$$\mu_R(x_i,y_j)=r_{ij}$$

模糊关系也可以合成。例如，设U、V、W是论域，Q是U到V的一个模糊关系，R是V到W的一个模糊关系。则Q与R的合成$Q\circ R$指的是由U到W的一个模糊关系，它具有的隶属函数为

$$\mu_{Q\cdot R}(u,w)=\bigvee_{v\in V}(\mu_Q(u,v)\wedge\mu_R(v,w))$$

当论域U、V、W为有限时，模糊关系的合成可以用模糊矩阵的合成来表示，即设Q、R、S三个模糊关系对应的模糊矩阵分别为

$$Q=(q_{ij})_{n\times m},\quad R=(r_{jk})_{m\times l},\quad S=(s_{ik})_{n\times l}$$

若

$$S=Q\circ R$$

则有

$$s_{ik}=\bigvee_{j=1}^{m}(q_{ij}\wedge r_{jk})$$

6.1.5 常见的模糊条件语句及其对应的模糊关系R

(1)if A then B

$R=A\times B$

(2)if A then B else C

$R=(A\times B)+(A^c\times C)$

(3)if A and B then C

$R=(A\times B)\cdot(B\times C)$

if A then if B then C

$R=A\times(B\times C)=A\times B\times C$

(4)if A or B and C or D then E

$R=[(A+B)\times E]\cdot[(C+D)\times E]$

(5)if A then B and if A then C

$R=(A\times B)\cdot(A\times C)$

if A then B,C

$R=(A\times B)\cdot(A\times C)$

(6)if A_1 then B_1 or A_2 then B_2

$R=(A_1\times B_1)+(A_2\times B_2)$

6.1.6 精确量与模糊量的相互转换

精确量转化为模糊量的过程称为模糊化。模糊化的方法一般是先将精确量离散化，分成若干个离散值，然后对每个离散值给定一个隶属函数，便可得到一个离散值与不同模糊量隶属度的一张表格。当读入一个精确量的数值时，先将它归入与之最接近的离散值，然后从表格便可

得到不同模糊量的隶属度。例如：将在[−4，+4]间连续变化的量转为模糊量，正大(*PB*)，正小(*PS*)，零(0)，负小(*NS*)，负大(*NB*)。首先，将[−4，+4]离散化，然后，将+4 对应“正大(*PB*)”的隶属度定为 1.0，并分配以一定的隶属函数(例如三角形函数)，对其它如“正小(*PS*)”、“负小(*NS*)”等也作如上处理，便可得到表 6-1。

表 6-1

	−4	−3	−2	−1	0	1	2	3	4
PB	0	0	0	0	0	0	0.3	0.6	1.0
PS	0	0	0	0	0	0.3	0.6	1.0	0.6
0	0	0	0.3	0.6	1.0	0.6	0.3	0	0
NS	0.3	0.6	1.0	0.6	0.3	0	0	0	0
NB	1.0	0.6	0.3	0	0	0	0	0	0

当读入精确量$x_i=3.2$时，可归入最近的离散值3，从表中可得：$\mu_{PB}(3)=0.6$，$\mu_{PS}(3)=0.6$，等等。从而，把一个精确量转换成模糊量。

反之，也可把一个模糊量转换成精确量。这个过程称为非模糊化，其基本方法有以下几种：

(1)选择最大隶属度法。

选择模糊子集中隶属度最大的元素作为输出。例如，模糊子集为 A，所选的输出 x^* 应满足

$$\mu_A(x^*)\geqslant\mu_A(x)\quad x\in X$$

若有多个 x_i^* 均满足以上条件，则可取其平均值或中点值作输出。

(2)取中位数法。

按照求出的模糊子集的隶属函数曲线与横坐标所围成的区域面积的平分线所对应的数值选取输出值。这种方法利用的隶属函数的信息较多，但计算量也大。

(3)加权平均法。

按下式选取输出 x^* 值

$$x^*=\frac{\sum_{i=1}^{N}K_ix_i}{\sum_{i=1}^{N}k_i}$$

$k_i(i=1,2,\cdots,N)$是事先选定的加权值。若取 $k_i=\mu_A(x_i)$，可得

$$x^*=\frac{\sum_i\mu_A(x_i)\cdot x_i}{\sum_i\mu_A(x_i)}$$

这种方法性能较好，但计算量也较大。

6.2 神经模糊控制的系统特点与结构

6.2.1 神经网络与模糊控制的特点

6.2.1.1 模糊控制特点

1974年英国教授Mamdani根据模糊集合论研制了世界上第一个FUZZY控制器，这是一种闭环负反馈非线性控制方式，图6-1表示了它的系统结构，它用于锅炉和蒸汽机的自动控制，取得了成功。

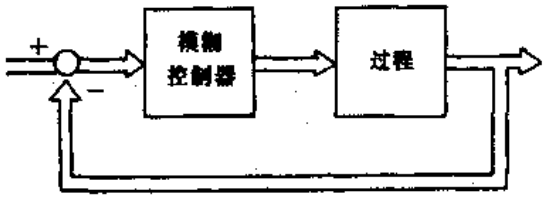


图6-1 闭环负反馈模糊控制系统

(1)模糊控制具有以下优点：

①模糊控制系统不依赖于系统精确的数学模型，特别适宜于复杂系统(或过程)。

②模糊控制中的知识表示、模糊规则和合成推理是基于专家知识或熟练操作者的成熟经验，并通过学习可不断更新，因此它具有智能性和自学习性。

③模糊控制系统的核心是模糊控制器，而模糊控制器均以计算机(微机，单片机等)为主体，因此它兼有计算机控制系统的特点，如具有数字控制的精确性与软件编程的灵活性。

④模糊控制系统的人机界面具有一定程度的友好性，它对于有一定操作经验而对控制理论并不熟悉的工作人员来说，很容易掌握和学会。

(2)模糊控制目前需解决的问题。

①模糊控制系统的稳定性理论探讨。

②模糊控制效果不够理想。

③模糊控制在非线性复杂系统应用中对于建立模糊规则及隶属函数比较困难。

④自学习模糊控制策略和智能化系统结构及其实现。

⑤简单，实用且具有模糊推理功能的模糊集成芯片和模糊控制装置，通用模糊控制系统的开发和推广应用。

(3)模糊控制系统现代应用的发展

①从过去的大型机械设备和生产过程为对象，向以家用电器为应用对象方面发展。

②向复杂系统、智能系统、人类与社会系统以及自然系统等方向扩展。

③在硬件方面向研制模糊控制，模糊推理专用芯片，模糊控制用的通用系统发展。

图6-2表示了模糊系统研究课题图

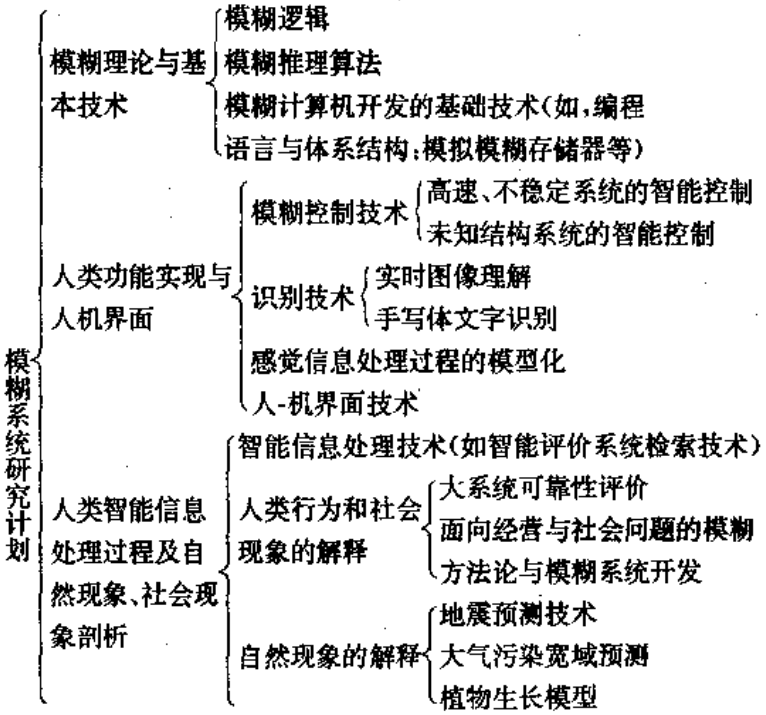


图 6-2　模糊系统研究课题图

6.2.1.2　神经网络的特点

(1)神经网络具有以下基本特点:

①具有分布式存贮信息的特点。它存贮信息是采用分布式存贮,即一个信息分布在不同的位置。这种分布式存贮方式即使当局部网络受损仍具有恢复原来信息的优点。

②具有并行处理信息和推理过程的特点,实现系统实时识别。

③对信息的处理具有自组织、自学习的特点。

(2)神经网络目前存在的问题。

①神经网络学习算法周期长,当网络节点多时计算工作量大,对在线快速实时控制有影响。

②容易陷入局部极小值点。

6.2.1.3　神经模糊控制的特点

随着现代科学技术的迅猛发展,人们所面临的问题日益复杂多变,采用单一的神经网络或模糊控制算法难以满足实际的需要。当前将模糊控制与神经网络二者相结合的神经网络模糊控制技术是控制理论研究者们倾注的焦点。

神经模糊控制具有以下特点:

①可以直接从经验数据中获取知识,自动建立模糊规则和隶属函数。

②不用查表,节省内存,只需通过在线计算,便可得到控制器输出。

③具有较强的适应能力和联想能力,在实时控制中对于未出现过的样本,神经模糊控制器可通过记忆、联想产生合适的输出量对系统进行控制。

6.2.2　神经模糊控制的系统结构

采用神经网络实现的模糊控制,对于知识的表达并不是通过显式的一条条规则,而是把这

些规则隐含地分布在整个网络之中。在控制应用时，不必进行复杂费时的规则搜索，推理，而只需通过高速并行分布计算就可产出输出结果。神经网络与模糊控制器相组合形成一个混合系统，这种混合系统可以具有这两种技术的长处，其组合一般有如图 6-3 所示的三种形式。

图 6-4 是带有神经网络的模糊控制结构框图。这种结构是用联想记忆神经网络记忆模糊控制规则。

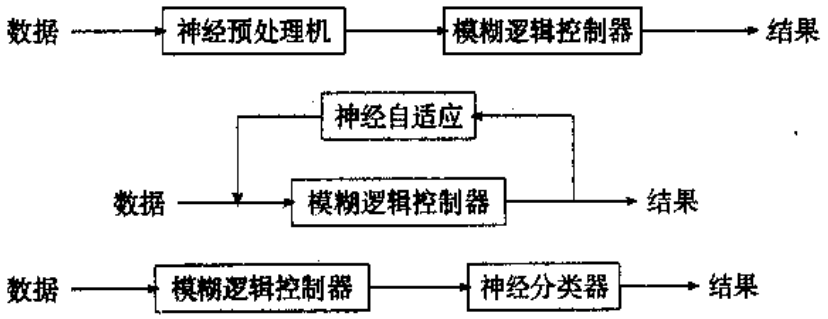


图 6-3　神经网络与模糊控制的结合形式

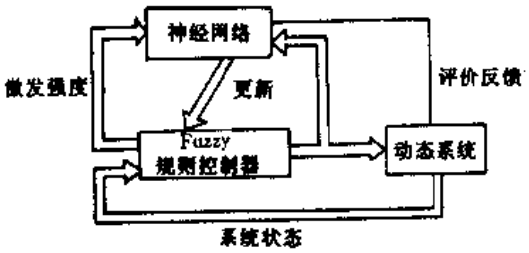


图 6-4　神经模糊控制结构框图

$$\text{if} \quad (E = A_i \quad \text{and} \quad \Delta E = B_i) \ \text{then} \quad U = C_i$$

这里 E、ΔE 分别为二个输入变量(例如误差与误差变化率)它对应于神经网络的两个输入节点(节点个数由输入语言变量的级数决定)。U 为输出变量(如校正量)，对应于神经网络的所有输出节点(输出节点个数由输出语言变量的级数决定)。如用 Back-Propagation 学习算法训练神经网络，调整网络的权重，使网络能够实现样本的输入输出的对应映射，那么这种多层神经网络还有隐节点层。目前对于 BP 神经网络的隐节点数目的选择尚缺少理论指导，须由试验确定。

6.3　神经网络在模糊控制中的应用

6.3.1　用神经网络实现模糊逻辑控制

6.3.1.1　概述

具有一个隐层的多层前馈网络的结构如图 6-5 所示。网络中每一个节点的总输入是它所有输入的加权和，每一个节点的传递函数可用 Sigmoid 非线性函数来表示。设 $X_k=[x_{0k}, x_{1k}, \cdots, x_{nk}]$，$x_{0k}=1$ 是网络第 k 个输入的样本，则第 j 个隐节点的输出 z_{jk} 可表示成

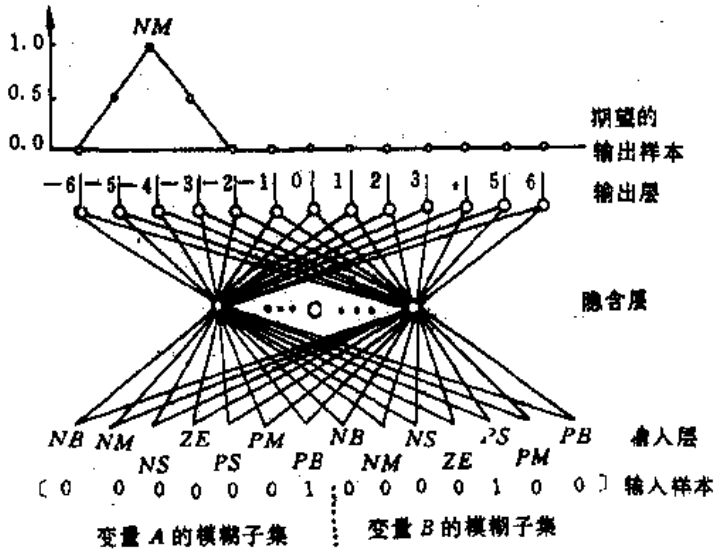


图 6-5 多层前馈网络结构

$$z_{jk}=\varphi\left(\sum_{i=1}^{n}w_{ij}x_{ik}\right) \tag{6-1}$$

式中，w_{ij}是第 i 个输入单元和第 j 个隐层单元的连接权值，$\phi(\cdot)$是 Sigmoid 函数，定义为 $\phi(v)=1/(1+e^{-v})$。隐层单元的输出，又作为输出层单元的输入。当网络接受输入样本后，就向前传播，生成输出信号 $Y_k=(y_{1k},y_{2k},\cdots,y_{mk})$。网络可以通过训练后，使一组给定输入映射到一组目标输出 $D_k=(d_{1k},d_{2k},\cdots,d_{mk})$上，这种训练是通过修正权值使输出平方误差最小来实现的。

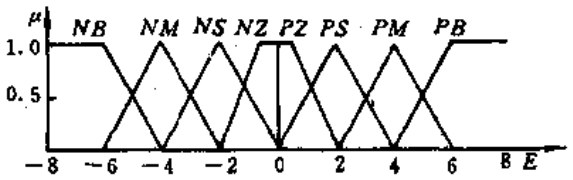


图 6-6 模糊子集规则定义

表 6-2 模糊规则表

A \ B	PB	PM	PS	ZE	NS	NM	NB
PB	NS		NM				
PM	PS			NS		NB	
PS							
ZE	PM			ZE		NM	
NS				PS	NS		
NM	PB				PS	NS	
NB				PM		PS	

为了训练网络来建立模糊关系，借助于数值样本来表示输入输出模糊子集。设两个输入变量 A 和 B，一个输出变量 C，均由图 6-6 所示模糊子集来定义，模糊规则由表 6-2 给出。网络输入空间对应于变量 A、B 被划分为两个部分。由于网络的每一个单元都对应输入变量的某一个

模糊子集，所以每一个输入变量都有7个输入单元与其7个模糊子集相对应，见图6-5。网络的输入信号格式如下：

$〔\mu_{NB}(a),\mu_{NM}(a),\mu_{NS}(a),\mu_{ZE}(a),\cdots,\mu_{ZE}(b),\mu_{PS}(b),\mu_{PM}(b),\mu_{PB}(b)〕$而网络的每一输出单元都对应着输出变量空间中的一个量化值，因此，输出的模糊子集 $\underset{\sim}{y}$ 就可用量化空间上的隶属函数来表示。其输出信号格式为

$$〔\mu_y(c_1),\mu_y(c_2),\cdots,\mu_y(c_{m-1},\mu_y(c_m)〕$$

变量 c 的论域分为13档，即$\{-6,-5,\cdots,5,6\}$。由上述定义，对于控制规则

"若 A 是 $\underset{\sim}{PB}$ 且 B 是 $\underset{\sim}{PS}$ 则 C 为 NM"

的输入信号可表示为

$$〔0,0,0,0,0,0,1;0,0,0,0,1,0,0〕$$

其中 $\mu_{\underset{\sim}{PB}}(a)=1.0$，$\mu_{\underset{\sim}{PS}}(b)=1.0$，其余均为0，输出信号为

$$〔0,0.5,1.0,0.5,0,0,0,0,0,0,0,0,0〕$$

其中 $\mu_{\underset{\sim}{NM}}(-5)=0.5$，$\mu_{\underset{\sim}{NM}}(-4)=1$，$\mu_{\underset{\sim}{NM}}(-3)=0.5$，其他为0，见图6-5。

由此可见，所有控制规则都可用一系列输入输出数字信号来表示。反向传播算法用于网络训练，使输入信号对应于期望的输出值。因此，经过训练的网络就相当于一个模糊关系存储器。

如果想要由经过训练的神经网络中推导其他结论，只要把输入变量的实际值模糊化后再输入网络，如 $a=2.5$，$b=-0.5$，则模糊化后输入信号为

$$〔0,0,0,0,0.75,0.25,0;0.5,0.5,0,0,0,0,0〕$$

在其输出端就会得到一个输出模糊子集，再经过解模糊来得到真实输出量。

经过训练后的网络能够对输入训练数据准确地做出响应。由于网络信息的存储结构十分巧妙，因此，它激发若干条规则的响应情况是很难准确描述的。在实践中，人们发现网络的推理机构受下列因素影响。

(1)一般规律。如果输入模糊子集接近于训练网络时用的模糊子集，则输出几乎和该条训练规则的结果相同，即若 $\underset{\sim}{A'}$ 和 $\underset{\sim}{B'}$ 是两个类似于 $\underset{\sim}{A_k}$、$\underset{\sim}{B_k}$ 的模糊子集，则输出应当符合规则$(\underset{\sim}{A_k},\underset{\sim}{B_k}\rightarrow\underset{\sim}{C_k})$，即 $\underset{\sim}{C'}$ 应该近似于 $\underset{\sim}{C_k}$。

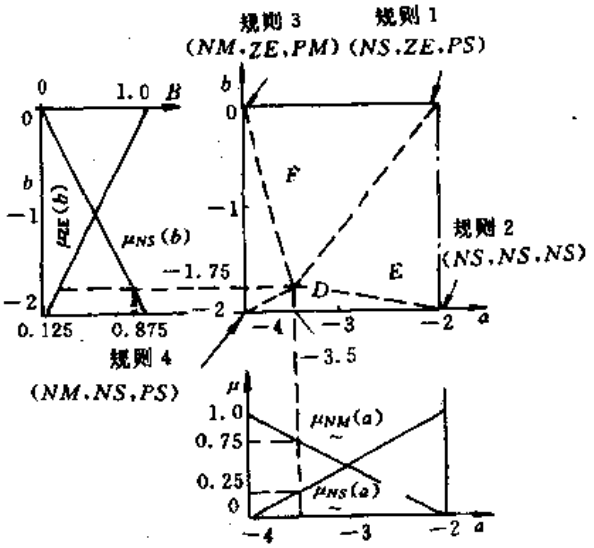


图6-7 变量矩形域

(2)非线性插值规律。当输入信号和训练用的模糊子集有较大差别时，输入后就有一系列相关节点被激发，输出量就是这些被激发规则的非线性插值。一条规则的输出响应是和当前输入与训练该规则时所用输入的距离成反比。例如，规则$(\underset{\sim}{A_k},\underset{\sim}{B_k}\rightarrow\underset{\sim}{C_k})$训练时用的输入为 $X_k=〔x_{1k},x_{2k},\cdots,x_{nk}〕$，而当前输入为 $X=〔x_1,x_2,\cdots,x_n〕$，则这两个信号的距离定义为

$$d(X,X_k)=\left(\sum_{i=1}^{n}(x_i-x_{ik})^2\right)^{1/2} \tag{6-2}$$

6.3.1.2 应用举例

设模糊关系如表 6-2 所示，输入变量 A、B 和输出变量 C 的模糊子集定义如图 6-6 所示，用图 6-5 的网络结构来学习该映射关系。该网络具有 14 个输入节点，13 个隐节点和 13 个输出节点。网络的初始权值取[−0.5,0.5]上的随机值，该网络用反向传播法进行训练，取 $a=0.5$、$\eta=0.8$。当输出误差的平方和小于某一限值时，可以认为训练结束。这里取限值为 0.5。这时 BP(Back—Propagation)算法需要训练 150 次。

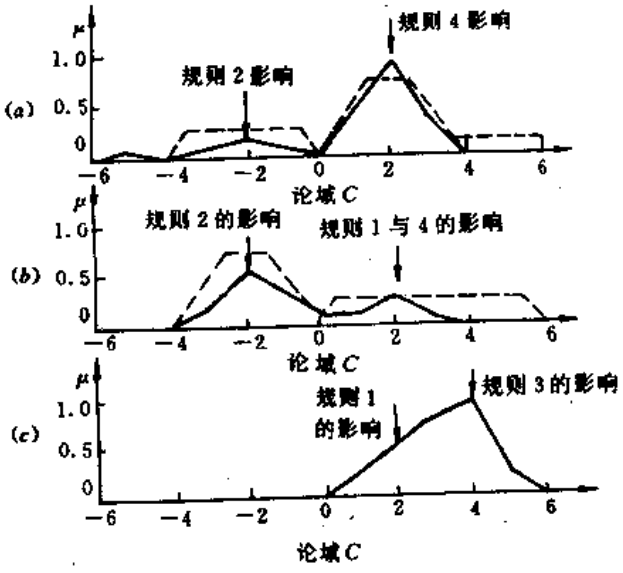


图 6-8 网络推算过程

这里，研究输入变量 $a\in(-4,2)$ 和 $b\in(-2,0)$ 所属的两个区间，在输入空间中表示为矩形域，如图 6-7 所示。矩形域的四个顶点代表四条规则：

(1)R_1:(NS,$ZE\rightarrow PS$)；

(2)R_2:(NS,$NS\rightarrow NS$)；

(3)R_3:(NM,$ZE\rightarrow PM$)；

(4)R_4:(NM,$NS\rightarrow PS$)。

每个顶点只能激发一条规则。在矩形内部，这四条规则的被激发程度决定于 a 和 b 的隶属度，这些规则的训练信号为

R_1:[0,0,1,0,0,0,0,0,0,0,1,0,0,0]；

R_2:[0,0,1,0,0,0,0,0,0,1,0,0,0,0]；

R_3:[0,1,0,0,0,0,0,0,0,0,1,0,0,0]；

R_4:[0,1,0,0,0,0,0,0,0,1,0,0,0,0]。

由不同的输入，经过网络推算如何得到输出模糊子集的过程如图 6-8 所示。

6.3.1.3 采用神经网络的自组织模糊控制器

一个自组织模糊控制器(FLSOC—FUZZY Logic self—organizing controller)的系统框图如图 6-9 所示。它通常由以下几个部分组成：

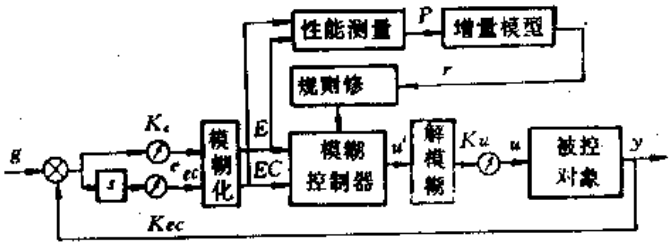


图 6-9 FLSOL 系统框图

(1)一个基于模糊规则推理算法的模糊控制器；

(2)用来对控制器自身作用进行评估的性能测量部分(Performance measure)；

(3)一个增量处理模型(Process model)，用于修改模糊控制器中的规则，以对控制作用给

出增补。

图 6-10(*a*)给出一个基于神经网络的自组织模糊控制器(NNSOC)。其控制规则通过一个神经网络来执行,而规则修正由反向传播学习算法实现。组成 NNSOC 中的性能测量和处理模型部分与 FLSOC 中的类同,下面仅阐述它们的不同之处。

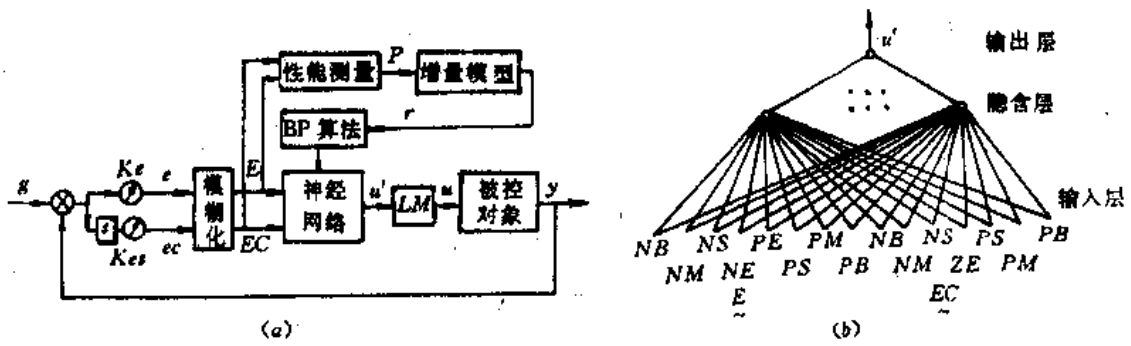


图 6-10 NNSOC
(*a*)框图;(*b*)网络组成

6.3.1.4 控制器

FLSOC 中采用的模糊控制器是由一组模糊规则与基于模糊推理合成算法的推理机组成。而 NNSOC 中的控制器是由神经网络来实现的,网络组成如图 6-10(*b*)所示。其输入编码格式与前节所述相同,网络输出代表了一个真实控制量,而不是一个模糊子集,因此,无需解模糊。在 FLSOC 中,采用比例因子 K_e,K_{ec}与 K_u 的目的是使输入、输出量变化范围折算到相应模糊子集的定义论域上。因此,网络的输出 u' 始终在区间[0,1]内,并与控制对象的输入给定成线性关系。

6.3.1.5 性能测量和数据转换

被控对象的工作情况可通过测量实际输出和期望输出给定间的误差来获得。对于每一采样时刻,可以通过误差 e 以及变化率 ec 来检测,性能测量通常采用判决表的形式,根据 e 和 ec 值来确定如何校正输出。表 6-3 给出的是 FLSOC 的判决表,并设定所有变量的模糊子集都被定义在区间[−6,6]之间,如图 6-6 所示。

表 6-3 FLSOC 性能测量判决 $r(k)$ 值

$r(k)$ $ec(k)$ / $e(k)$	−6	−5	−4	−3	−2	−1	0	1	2	3	4	5	6
−6	6	6	6	6	6	6	6	0	0	0	0	0	0
−5	6	6	6	6	6	6	6	3	2	2	0	0	0
−4	6	6	6	6	6	6	6	4	5	4	2	0	0

（续）

$r(k)$ \ $ec(k)$ / $e(k)$	−6	−5	−4	−3	−2	−1	0	1	2	3	4	5	6
−3	6	5	5	4	4	4	4	3	2	2	0	0	0
−2	6	5	4	3	2	2	2	0	0	0	0	0	0
−1	6	5	4	2	1	1	1	0	0	0	0	0	0
−0	4	3	2	1	0	0	0	0	0	0	0	0	0
+0	0	0	0	0	0	0	0	0	0	−1	−2	−3	−4
1	0	0	0	0	0	0	−1	−1	−1	−2	−3	−4	−6
2	0	0	0	0	0	0	−2	−2	−2	−3	−4	−5	−6
3	0	0	0	−2	−5	−5	−4	−4	−4	−4	−5	−5	−6
4	0	0	0	−2	−5	−5	−4	−6	−6	−6	−6	−6	−6
5	0	0	0	−2	−3	−3	−6	−6	−6	−6	−6	−6	−6
6	0	0	0	0	0	0	−6	−6	−6	−6	−6	−6	−6

由判决表得到的输出校正量需要转换成输入增量，通常使用一种近似的处理模型 M（矩阵表示），它和系统的雅可比矩阵相关，由它把修正环节的输入变化矢量 r 和性能测量部分的输出校正矢量 ρ 联系起来，即

$$r = M^{-1}\rho \tag{6-3}$$

对于单输入-单输出（SISO）系统，如果模型经过归一化后，则 $M=1$。在 NNSOC 中，可以采用同样的性能测量与数据转换。

6.3.1.6 控制器自校正

在 FLSOC 中，规则修正过程是：

（1）确定造成当前不良结果的控制行为；

（2）确定该控制行为产生于哪一条规则；

（3）由输出偏移来计算输入增补量；

（4）修正相应规则，以使在类似情况时增强控制作用。

对于 SISO 系统（见图 6-9），设 k 时刻发生不良的控制结果是由某一控制作用的前 m 个采样值 $u(k-m)$ 所产生，并且对应该控制作用的规则是 $(\underset{\sim}{E}_k, \underset{\sim}{EC}_k \rightarrow \underset{\sim}{U}_k)$。又设 $r(k)$ 是 k 时刻由性能测量和处理模型计算得到的修改输入量，则在 $(k-m)$ 时刻的控制量应该是 $u(k-m)+r(k)$，相应于该控制作用的规则应为 $(\underset{\sim}{E}_k, \underset{\sim}{EC}_k \rightarrow \underset{\sim}{U}'_k)$。对于 FLSOC 来说，规则修改可以是：

（1）修改关系矩阵。但由于关系矩阵通常容量很大，对它进行搜索与修改需要很多时间。

（2）修改相应规则。通常由生成的新规则替代原来规则。但往往由于每条旧规则的修改会由多于 1 条，甚至是 3 条新的规则所代替，这样会导致规则总数急剧增多。随着控制器中规则数的增加，用于控制行为评估的计算时间也将相应增加。

在NNSOC中，可以通过调整网络的权值来修正控制量，从而使网络在$(k-m)$时刻的输入v_{k-m}所对应的输出为$u(k-m)+r(k)$，而不仅仅是$u(k-m)$。采用BP算法进行一定次数的反复计算，就能修正网络的权值而收到期望效果。NNSOC的学习过程具有如下优点：

(1)它不需要寻找在$(k-m)$时刻产生控制作用的那些规则；

(2)它既不需要对控制器的输出解模糊，也不需要对输入变量模糊化；

(3)由于网络的前向推算时间是固定的，因此，估算控制行为的时间也将不变；

(4)如果能研制出具有在线学习能力的神经网络芯片，则就有可能实现控制过程的高速处理，大大增强实时性。

在FLSOC(NNSOC)中形成一套成功的控制策略之前，需要进行若干次试验(训练)，每次试验(训练)都由一批预定数量的采样值来完成，规则(权值)的修改是在试验(训练)过程中的每一个采样时刻进行。修正后的规则(权值)被用于下一次试验(训练)，并且每次试验(训练)前均应对控制变量初始化。每次试验(训练)使系统响应特性(即上升时间，超调量等)逐渐得到改善，不断地重复试验(训练)，直到FLSOC(NNSOC)呈现出可以被接受的性能为止。

在给定的采样时刻，采用BP算法，其反复计算次数决定于网络的大小，通常，大型网络需要更多的训练次数。如果，在每一采样时刻，计算的次数太多，则网络也有可能忘记以前学习过的内容，可见在NNSOC中计算次数的选择也极为重要。

设一个二阶系统，其状态方程如下：

$$\begin{cases} \dot{X} = AX + Bu(t) \\ y(t) = CX \end{cases} \tag{6-4}$$

式中，$X=[x_1, x_2]^T$是状态变量，$u(t)$是外界输入，$y(t)$是输出变量，系数阵$A=\begin{bmatrix} 0 & 1 \\ a_2 & a_1 \end{bmatrix}$，$B=\begin{bmatrix} 0 \\ a_3 \end{bmatrix}$，$C=\begin{bmatrix} 1 \\ 0 \end{bmatrix}$，取$a_1=-1.5$，$a_2=-0.5$，$a_3=10$。输出期望值为$s(t)=1.0$。控制器有两个输入变量，误差$e(t)=s(t)-y(t)$和误差变化率$ec(t)=de(t)/dt$，一个输出变量$u(t)$。变量$EC$的模糊子集定义见图6-11。为了使输出在接近期望值时控制更为精确，因此，变量E增加了两个模糊子集PZ与NZ。

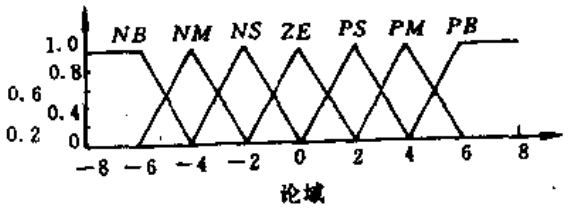


图6-11 E的模糊子集定义

采用四阶龙格-库塔法对该系统进行仿真，取步长为0.01s。自组织控制器采用一个单隐层网络，它具有15个输入单元：前8个代表$e(t)$的模糊子集；后7个代表$ec(t)$；8个隐节点和1个输出单元。网络初始权值为随机值，即表示该控制器未受过任何规则训练。本例取$K_e=3$，$K_{ec}=30$，网络输出u'在(0,1)区间取值，并与实际控制对象的输入u呈现线性关系，由于这里$u\in[-1,1]$，所以$u=2u'-1$。

这里，性能测量判决表采用表6-3所列，定义区间均为[−6,6]。假定$(k-1)$时刻的控制作用决定了控制器在k时刻的输出量。在采样时刻k由判决表查得修正环节的输入变化量为$r(k)$，并且$(k-1)$时刻的输入信号v_{k-1}的目标输出被设定为$u(k-1)+Kr(k)$。这里取$K=0.16$是把信号$r(k)$从[−6,6]区间折算到[−1,1]上的一个比例常数，为了得到期望值，在每个采

样时刻通过 BP 算法计算 5 次，来修正网络权值。BP 算法中，$\eta=0.2$，$a=0.7$。

控制器的学习过程要经过若干次训练，每次训练采用 400 个采样样本值，训练前要对控制变量作初始化，学习训练一定要持续到获得满意的输出响应为止。选择不同的训练次数（如 1 次、6 次与 21 次），该闭环系统的阶跃响应如图 6-12(*a*)所示。同样的控制器，当取 $a_1=-1$、$a_2=-6$、$a_3=30$，比例因子不变时，经过不同训练次数（如 1 次、6 次与 21 次）后的系统响应如图 6-12(*b*)所示。通过多次试验表明：

(1)网络的训练次数越多，控制器输出越稳定；

(2)同样结构的控制器，可适用于许多不同被控对象，仅响应特性有所不同；

(3)当比例因子在很大范围内变化时，也不会影响该控制器的自组织性能。因此，该控制器具有很强适用性。

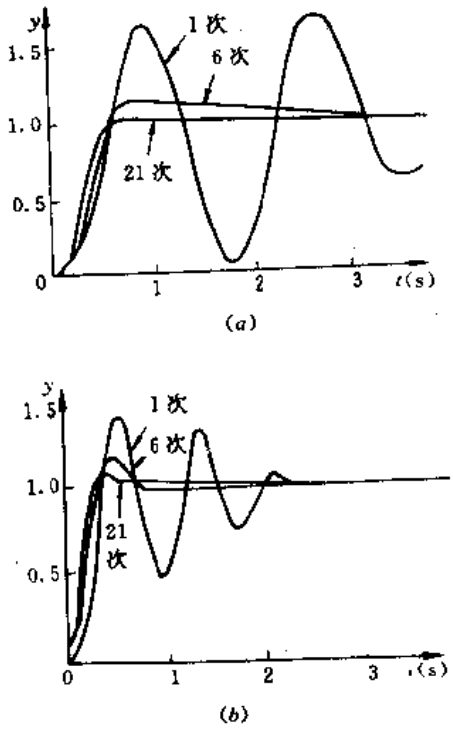


图 6-12 NNSOC 仿真结果

6.3.2 神经模糊控制洗衣机

6.3.2.1 概述

自 90 年代初，日本松下等公司相继推出了神经模糊控制全自动洗衣机。这种洗衣机能够自动判断被洗衣物的质地软硬、衣量多少、脏污程度和性质（即油污或泥污）；应用神经模糊控制技术，自动生成模糊控制规则和隶属度函数，预设洗衣水位，水流强度和洗涤时间，在整个洗衣过程适时调整这些运行参数，达到最佳洗涤效果。本节将简要介绍神经模糊控制洗衣机的基本工作原理。

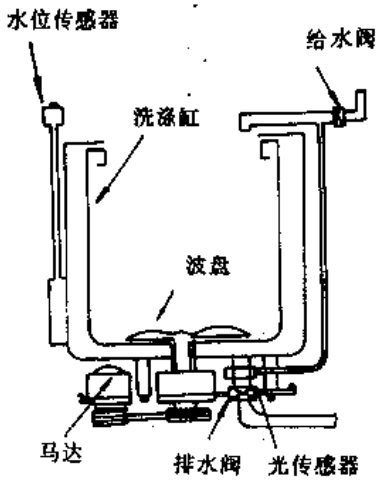


图 6-13 神经模糊洗衣机的结构图

6.3.2.2 神经模糊洗衣机的结构

图 6-13 是神经模糊全自动洗衣机的结构略图。它由马达、波盘、给排水阀、光传感器、水位传感器，负载量传感器等组成。

(1)洗衣机状态信息检测。

①衣物脏污程度，脏污性质及洗净度的检测，用红外光电传感器，通过分析透光度与时间的变化关系，应用模糊推论，得到测量结果，这是一种硬软件相结合的软传感器。传感器的按装部位见图 6-14 所示。

利用光传感器、检测光的透过率来测定洗涤液的混浊度、设定洗涤清洗状态。

洗涤时(*a*)图→(*b*)图洗涤终了

清洗时(*b*)图→(*a*)图清洗终了

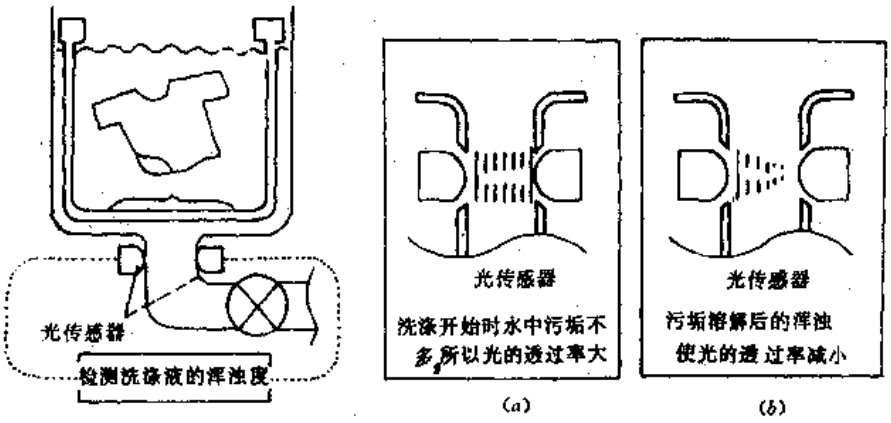


图 6-14　红外光电传感器工作原理

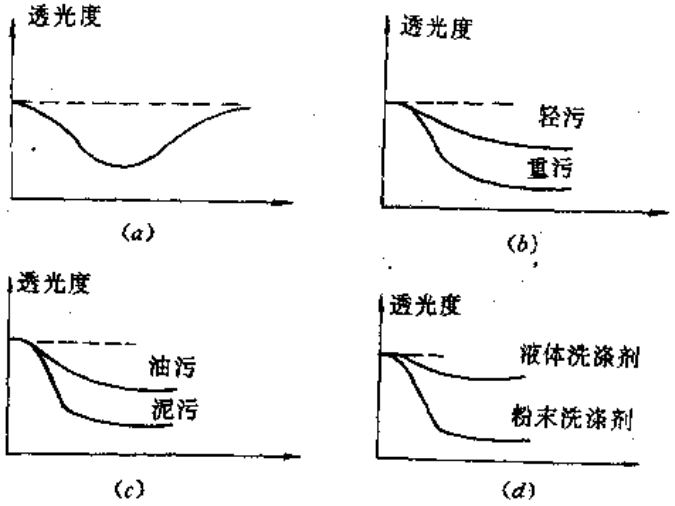


图 6-15　洗涤过程中透光度的变化曲线

在洗衣机排水管两侧分别按装红外发光管和光电接收管，当红外发光管工作电流一定时，光电接收管的感应电势便反应管内水的透光度。在洗涤过程中，透光度的变化曲线如图 6-15 所示。

图 6-15(a)中是从洗涤开始到漂洗结束全过程透光度的变化曲线。洗涤开始后，随着水的变浊，透光度逐渐降低，最后会达到一个饱和值，而当漂洗时，随着水的变清，透光度逐渐升高，并渐近于初始值。这个渐近程度可用来检测洗净度，以确定漂洗的结束。图中 6-15

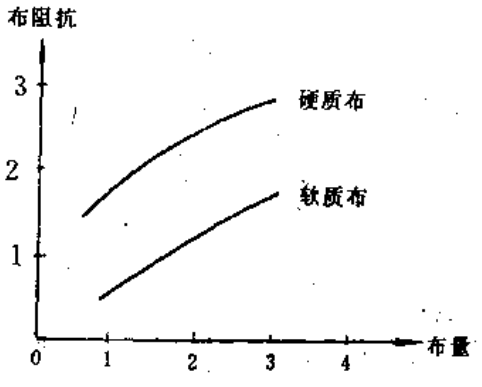


图 6-16　布阻抗曲线之一

(*b*)、(*c*)、(*d*)分别是污浊程度(轻污、重污)，脏污性质(油污、泥污)和洗涤各类(液体、粉末)的透光度变化曲线。由图可见，根据透光度的时变率便可判别衣物在脏污程度，脏污性质及所有洗涤剂的种类。

②布质、布量的检测。由于当水位一定时，不同的布质、布量产生的布阻抗不同，如图 6-16 所示，而当布量一定时，不同的布质和水位产生的布阻抗也不同。如图 6-17 所示，因此，可以通过布阻抗的大小将洗衣电机在正常运转下突然断电，使其以惯性继续运转，然后检测其反电势，并将此反电势半波整流后放大整形得到一矩形脉冲系列，如图 6-18 所示。这个脉冲系列反映了布阻抗对惯性运转电机的阻尼过程，亦即反映了布阻抗的大小和变化来测知布质和布量，布阻抗可这样测量，分析这一脉冲系列的个数和脉冲宽度，通过模糊推论，就可得出布量和布质。例如，如布量多，布质硬则布阻抗大，电机反电势衰减快，反映在脉冲系列上就是脉宽较快变窄(脉宽与电势幅值相对应)，脉冲个数较少。

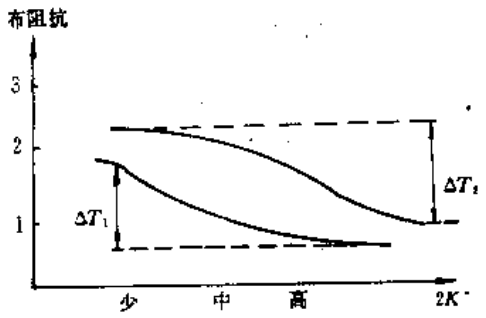


图 6-17　布阻抗曲线之二

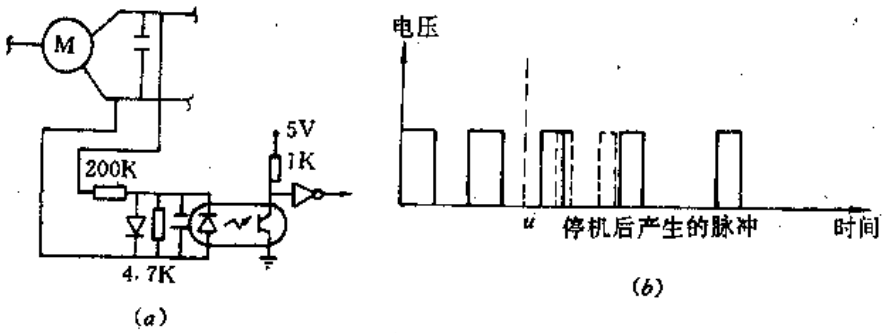


图 6-18　布质、布量的检测

(2)硬件构成。

图 6-19 为神经模糊洗衣机的硬件电路构成框图，以单片机为中枢，由负载量(即布质、布量)检测电路、温度检测电路、水位检测电路、显示电路、键盘矩阵变换电路、执行机构控制电路等构成。

单片机的基本规格如下：

- 处理位数　8 位
- ROM　16K 字节
- RAM　512 字节
- 机器周期　0.5μs

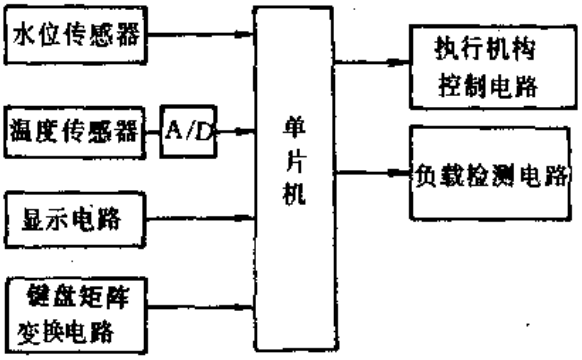


图 6-19　神经模糊洗衣机的电路构成

6.3.2.3　洗衣机的模糊控制

分析洗衣机的运行过程可以看出：其主要被控参量是洗涤时间和水流强度，影响这一输出参量的主要因素是被洗物的混浊程度和浑浊性质，后者可以用浑浊度的变化率来表征。在洗涤过程中，油污的衣服浑浊度变化率小，泥污的浑浊度变化率大。因此，浑浊度和浑浊度变化率可以作为控制系统的输入参量，而洗涤时间和水流强度可作为控制量，即系统的输出。实际分析证明：洗衣过程中的这类输入和输出之间很难用一定的数学模型来描述，系统的界面条件具有较大的不确定性，控制过程在很大程度上依赖于操作者的经验，用常规的方法进行控制难以取得好的效果。然而应用专家知识，"如果……(条件)，那么……(结果)"的形式进行控制决策，往往容易实现优化控制。这就是在洗衣机及家用电器中引入模糊控制技术的主要原因。

6.3.2.4　模糊控制规则的设计

根据上述分析的模糊控制的基本原理，作出确定洗涤时间的模糊推理框图如图 6-20 所示。其输入量为洗涤水的浑浊度和浑浊度变化率，输出量为洗涤时间。考虑到适度的控制要求和简化程序，定义输入量浑浊度的模糊词集为：{清、较浊、浊、很浊}，定义输入量浑浊度变化率的模糊词集为：{零、小、中、大}，定义输出变量洗涤时间的模糊词集为：{短、较短、标准、长}。

描述输入、输出变量的词集都具有模糊特性，可用模糊集合来表示。因此，模糊概念的确定问题就直接转化为求取模糊集合隶属函数的问题。

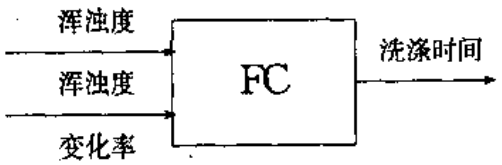


图 6-20　确定洗涤时间的模糊推理框图

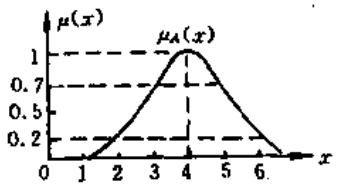


图 6-21　模糊变量 A 的隶属度函数

6.3.2.5　定义各模糊变量的模糊子集

通常定义一个模糊子集，实际上就是要确定模糊子集隶属函数的形状。将确定的隶属函数

曲线离散化，就得到了有限个点上的隶属度，构成了一个相应的模糊子集。如图 6-21 所示的隶属函数曲线表示论域 x 中元素 x 对模糊变量 A 的隶属程度，设定论域 x 为

$$x=(-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6)$$

则有

$$\mu_A(2)=\mu_A(6)=0.2;\mu_A(3)=\mu_A(5)=0.7;\mu_A(4)=1。$$

论域 x 内除 $x=2,3,4,5,6$ 外各点的隶属度均为零。则模糊变量 A 的模糊子集为 $A=0.2/2+0.7/3+1/4+0.7/5+0.2/6$。

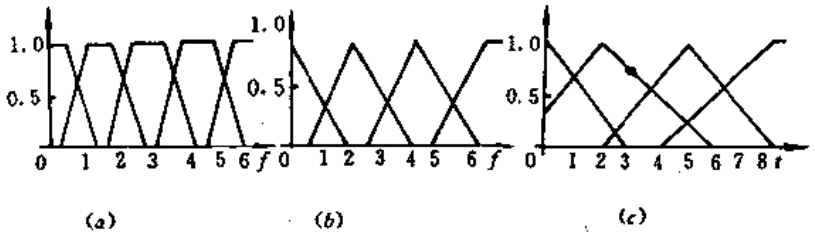


图 6-22　洗衣机模糊变量隶属函数

(a)浑浊度 f；(b)浑浊度变化率 f；(c)洗涤时间 t

由上面的例子可以看出，确定了隶属函数的曲线后，就很容易定义出一个模糊变量的模糊子集。洗衣机模糊控制的输入，输出变量的隶属函数如图 6-22 所示，可以依次来定义它们的模糊子集。

依据实验分析和操作者的经验，洗衣机的模糊控制规则可以归纳为 16 条：

如果浑浊度为清且浑浊度变化率为零，则洗涤时间短，如果浑浊度为浊且浑浊度变化率为中，则洗涤时间为长；……将以上规则写入控制规则表如表 6-4 所示。

表 6-4　控制规则

洗涤时间 T 变化率 \ 浑浊度	清	较浊	浊	很浊
变化为零	短	较短	标准	标准
很小	标准	标准	标准	标准
中	标准	长	长	长
大	标准	标准	长	长

6.3.2.6　模糊控制器

模糊控制洗衣机控制部分框图可以用图 6-23 表示。模糊控制器如图中虚线框所示。模糊控制器的控制规律由计算机程序实现。由洗衣机获取的浑浊度信息由传感器送到信息处理单元，分为浑浊度和浑浊度变化率送入模糊控制器。

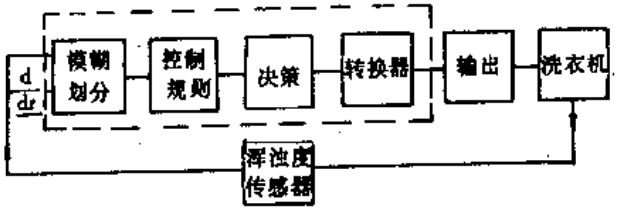


图 6-23　模糊控制器构成

对于输入的模拟量，需要将其变换成模糊变量，通过单片机，应用查表法，按模糊推理法则作出决策，这时仍为模糊变量，经非模糊化单元处理，再由执行机构去修改洗涤时间，完成一次模糊控制算法过程。

6.3.2.7 洗衣机的神经模糊控制器设计

一般模糊控制洗衣机将“专家经验”通过模糊控制规则表体现出来，运行中采用查表法作出控制决策（洗涤时间和水流强度），它比需要操作者预洗设定程序的电脑程控洗衣机前进了一大步；但这种洗衣机由于规则表需占用大量内存，查表反应速度慢，只能按已编入的规则进行控制，因此不够理想。把神经网络与模糊控制相结合，能够解决这些缺陷。

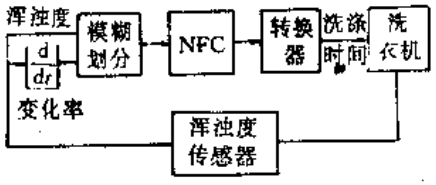


图 6-24 浑浊神经模糊控制框图

6.3.2.8 洗衣机神经模糊控制器结构框图

神经模糊控制洗衣机是利用离线训练后确定的神经网络，经过在线计算即可得到最佳输出，反应速度快，而且神经网络又具有自学习功能和联想能力，对未在训练中出现的样本，也可通过联想、记忆的功能，作出控制决策，表现非常灵活。

神经模糊控制洗衣机控制系统内含有多个神经模糊环节，下面介绍以浑浊度和浑浊度变化率为输入参量来确定洗涤时间的控制器。该环节控制框图如图 6-24 所示，神经网络结构如图 6-25 所示。

神经模糊控制器在输入、输出参量的选择以及模糊论域和模糊子集的确定等方面与一般模糊控制器没有什么区别，只是在实现模糊推理手段上引入了神经网络。

设定 $x_1 \sim x_7$ 为输入量浑浊度的模糊子集；$x_8 \sim x_{14}$ 为输入量浑浊度变化率的模糊子集；$y_1 \sim y_8$ 为输出控制量的模糊子集。

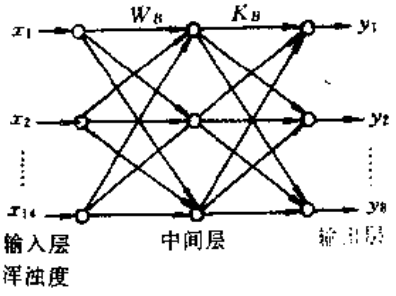


图 6-25 浑浊度神经网络结构图

从模糊控制规则表可以看出，共有 16 条规则，每条规则都是一对样本，共有 16 个样本。例如当浑浊度为“清”，浑浊度变化率为“零”时，洗涤时间应为“短”，这个样本应表示为

$$x = [1\ 0.6\ 0.1\ 0\ 0\ 0\ 0\ 1.0\ 0.5\ 0\ 0\ 0\ 0\ 0]^T_{1\times 14}$$

其中各元素是对应的隶属函数，即模糊子集的赋值。

$$y = [1.0\ 0.5\ 0\ 0\ 0\ 0\ 0\ 0]^T_{1\times 8}$$

同理可以列出其它 15 个样本对。将它们依次送入神经网络进行离线训练，当训练结束后，神经网络已记忆了模糊控制规则，使用时具有联想记忆功能。训练时的 BP 算法用 C 语言编程实现，使用时正向运算由单片机汇编语言完成。

洗衣机神经模糊控制器的结构框图如图 6-26 所示。框图中的“模糊推论”是用于脏污程度、脏污性质、布质、布量的软传感器检测。框图中“神经模糊控制”是用于洗衣控制的。程序启动后，先完成初始化工作，然后进行键查询处理，启动洗涤功能。在洗涤功能程序中完成布质、布量、脏污程序和脏污性质的检测，而后对这些量进行神经模糊运算和决策，并按决策要求自

动设定最佳注水量(水位)、洗涤剂投放量、洗涤时间、洗衣机电动机运转参数(转速,正反转时间),然后进入正式洗涤。洗涤完毕,转入漂洗、脱水程序,直到结束鸣报。

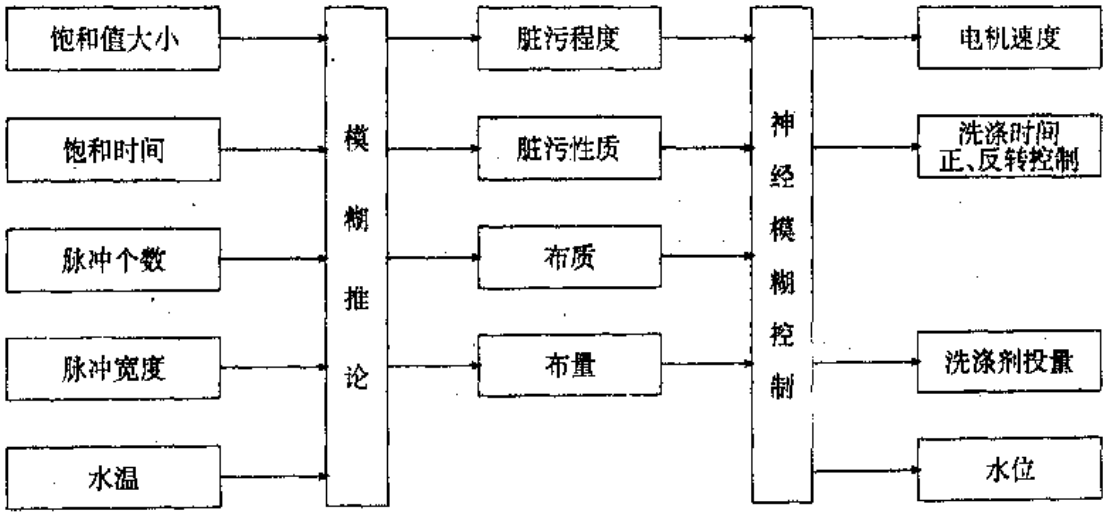


图 6-26 神经模糊控制结构框图

这种智能控制洗衣机,任何人都可以简单地操作,即可节约能源,又能保护衣料,给人们的日常生活带来极大方便,使神经模糊理论的应用更加引起人们的兴趣,为越来越多的人所关注。

参考文献

[1] K. S. Narendra, k. parthasarathy, *Identification and control of Dynamical systems using Neural Nerworks*, IEEE Trans. on Neural Networks , Vol. 1, No. 1, March, 1990.

[2] Y. Iiguni, H. Sakai and H. Tokumaru, *A Nonlinear Regulator Design in the presece of system uncertainties using Multilayered Neural Networks* IEEE Trans. Neural Networks, vol. 2, No. 4, July 1991.

[3] K. J. Hunt, D. Sbarbaro, *Neural Networks for control systems —A survey*, Automatia, vol. 28, No. 6, pp. 1083—1112, 1992.

[4] D. T. pham and E. J. Bayro—corrochano, *Neural Networks for classifying surface Derfects on Automotive valve stem seals*Int. J. Mach. Tools Manufact. vol. 35, No. 35, No. 8, pp. 1115—1124, 1995.

[5] T. Sorsa, H. N. Koivo, *Neural Networks in process Fault Diagnosis*, IEEE Trans. on systems, Man, and Cybernetics, vol. 21, No. 4, July/August 1991.

[6] L. Ma, *Condition Monitoring and Fault Diagnosis using Neural Networks*, proceedings of International Intelligence, knowledge and Integrated Manufacturing , vol. 1, pp. 53—58, 1995.

[7] D. T. Pham and L. Xing, *Neural Networks for Identification, prediction and control*, London Berlin Heidelberg New York Paris Tokyo Hongkong Barcelona Budapest, 1995.

[8] J. D. Boslovic and K. S. Narendra, *comparison of linear, Nonlinear and Neural—network based Adaptive controllers for a class of Fed—batch Fermentation Processes*, Automatia, vol. 31, No. 6, pp. 817—840, 1995.

[9] M. G. Rodd and H. B. Verbruggen, Artificial Intelligence in Real time control, Engng Applic. Artif. Intell. vol. 5, No. 5, pp. 385—399, 1992.

[10] C. L. Giles, G. M. Kuhn, and R. J. Williams, *Dynamics Recurrent Neural Networks. Theory and Application*, IEEE Trans. on Nearal Networks, vol. 5, No. 2, pp. 153—156, March, 1994.

[11] H. C. Zhang, S. H. Huang, *Application of neural networks in manufacturing: a state—of—the art survey* , INT. J. RROD. RES; 1995, Vol. 33, No. 3, pp. 705—728.

[12] A. Bahrami, M. Lynch, and C. H. Dagli, *Intelligent design retrieval and packaging system application of neural networks in design and manufacturing*, INT, J, PPOD, RES, 1995, Vol. 33, No. 2, pp. 405—426.

[13] S. A. Barton, *Two—dimensional Movement Controlled by a chaotic Neural Network*, Automatica, vol. 31, no. 8, pp. 1149—1155, 1995.

[14] D. T. Pham and E. oztemel, *An integrated neural network and expert system tool for*

statistical process control, Journal of Engineering Manufacture, pp. 91—97, 1995.
[15] K. L. Moore, *A Reinforcement—learning Neural network for the control of nonlinear systems*, J. Amer. stat. Assac, vol. 84, pp. 1003—1013, 1992.
[16] G. Bastin, and D. Dochain, *On—line Estimation and Adaptive control of Biereuctors*, Elsevier, New York, 1990.
[17] K. S. Narendra and K. Parthasarathy, *Gradient methods for optimization of Dynamical systems containing Neural Networks*, IEEE Trans. Neural Networks, NN—2, pp. 252—262, 1991.
[18] L. B. Almeida, *Backpropagation in perceptrons with feedback*, INR. Eckmiller and ch. r. d. Malsburg (Eds), Neural computers. springer—verlag, Berlin (1988).
[19] C. W. Anderson, *Learning to control an inverted pendulum using neural networks*, IEEE Control systems Magazine, 9, pp. 31—37, 1989.
[20] J. J. Slotine, W. Li, *on the adaptive control of robot manipulators*, *Int, J. of Robotics Research*, 6(3), 49—59.
[21] E. R. Kandel, and R. D. Hawkins, *The biological basis of learning and individuality*, sci, Am, september, 79—86, 1992.
[22] M. E. Cohen and D. L. Hudson, *Approaches to the handling of Fuzzy input data in Neural Networks*, Fuzzy—IEEE 92 93—100.
[23] J. H. Williams, A. Pavies and P. R. Drake, *Condition—Based maintenance and machine diagnosis*, chapman Hall, London SE1 8HN, U. k, 1994.
[24] L. ma, H. Gang, *Artificial intelligence applications*, Journal of zhengzhou Institnte of light Iudnstry, vol. 10, No. 3, pp. 67—72, 1995.
[25] 张立明，神经网络及其应用，复旦大学出版社，1992.
[26] 胡守仁主编，神经网络及其应用，国防科技大学出版社，1995.
[27] 靳蕃等，神经网络与神经计算机原理·应用，西南交通大学出版社，1991.
[28] [美]包约翰著，自适应模式识别与神经网络，科学出版社，1992.
[29] 焦李成著，神经网络系统理论，西安电子科技大学出版社，1991.
[30] 汪培庄，模糊集合论及其应用，上海科学技术出版社，1983.
[31] 应行仁，曾南，采用 BP 神经网络记忆模糊规则的控制，自动化学报 1(1991)，63—67.
[32] 徐冬玲等，交通系统的模糊控制及其神经网络实现，信息与控制，No. 2，pp. 74—79，1992.
[33] 黄战，戴冠中，神经网络控制器的研究，信息与控制，No. 6，1991，pp. 27—32.
[34] 刘延年，冯纯伯，神经网络在控制领域中的应用，东南大学学报，vol. 24，No. 1，Jan. 1994.
[35] 谭明，琉松桂，控制系统故障诊断的一种智能算法，控制理论与应用，1991，8(3)：339～343.
[36] 黄国建等，基于神经网络的舰船雷达目标识别，信息与控制，vol. 24，No. 2，Apr. 1995.
[37] 卢进等，神经预测和模糊推理在催化重整优化控制中的应用，信息与控制，vol. 24，No. 2，Apr. 1995.

[38] 韦巍，蒋静坪，非线性系统的多神经网络自学习控制，信息与控制，vol. 24，No. 6，1995.

[39] 陈燕庆，鹿浩编著，神经网络理论及其在控制工程中的应用，西北工业大学出版社，1991.

[40] 胡德文，王正志，多层神经网络逼近精度研究，中国神经网络首届学术大会论文集，1990.

[41] R. G. Hoffman，崔良泽，谈人工智能研究途径，计算机科学，1990.

[42] 刘锡荟，王海燕，网络模糊随机分析，电子工业出版社，1990.

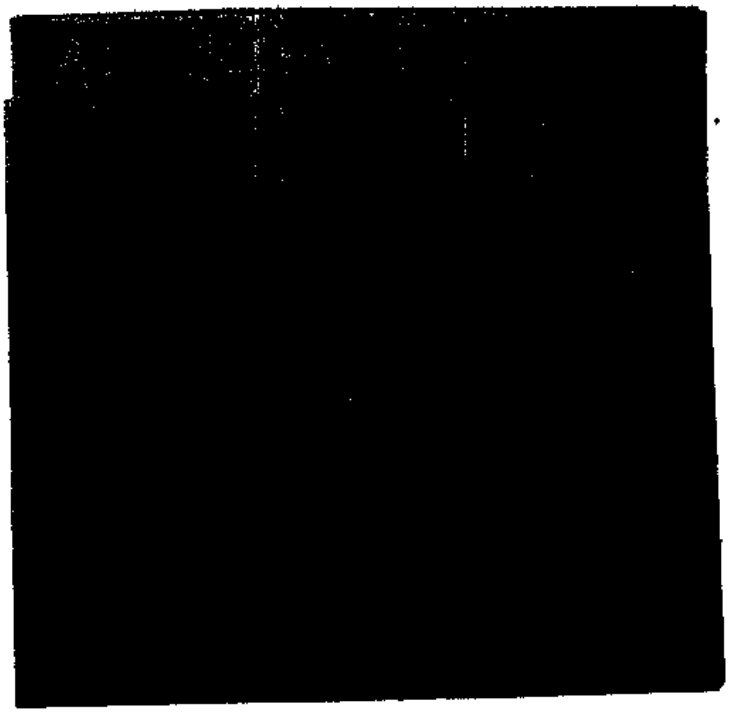
[43] 尹红风，戴汝为，人工神经网络信息处理原理，模式识别与人工智能，vol. 3，No. 1，1990.

[44] 杜利民，侯自强，多层前馈神经网络快速学习算法的实现，电子学报，1992，10.

[45] 田明，戴汝为，神经元网络控制系统，信息与控制，vol. 21，No. 3，pp. 156-161，1992. 6.

[46] 邵世煌等，一个基于类神经元模型的智能控制系统及其在柔性臂上的应用研究，控制与决策，vol. 5，No. 2，pp. 32－37，1990.

[47] 马莉，神经网络及其应用，郑州轻工学院校内教材，1995.



[38] 韦巍，蒋静坪，非线性系统的多神经网络自学习控制，信息与控制，vol. 24，No. 6，1995.

[39] 陈燕庆，鹿浩编著，神经网络理论及其在控制工程中的应用，西北工业大学出版社，1991.

[40] 胡德文，王正志，多层神经网络逼近精度研究，中国神经网络首届学术大会论文集，1990.

[41] R. G. Hoffman，崔良泽，谈人工智能研究途径，计算机科学，1990.

[42] 刘锡荟，王海燕，网络模糊随机分析，电子工业出版社，1990.

[43] 尹红风，戴汝为，人工神经网络信息处理原理，模式识别与人工智能，vol. 3，No. 1，1990.

[44] 杜利民，侯自强，多层前馈神经网络快速学习算法的实现，电子学报，1992，10.

[45] 田明，戴汝为，神经元网络控制系统，信息与控制，vol. 21，No. 3，pp. 156-161，1992. 6.

[46] 邵世煌等，一个基于类神经元模型的智能控制系统及其在柔性臂上的应用研究，控制与决策，vol. 5，No. 2，pp. 32－37，1990.

[47] 马莉，神经网络及其应用，郑州轻工学院校内教材，1995.