



普通高等教育“十一五”国家级规划教材

人工神经网络理论、设计及应用

第二版

韩力群 编著



化学工业出版社



自动化·信息工程类系列规划教材

部分书目

教育部面向21世纪课程教材

自动检测技术及仪表控制系统(第二版)

张毅

普通高等教育“九五”国家级重点教材

控制仪表与计算机控制装置

周泽魁

普通高等教育“十五”国家级规划教材

自动检测技术与装置

张宏建

普通高等教育“十一五”国家级规划教材

微型计算机原理及应用(第二版)

侯晓霞

信号与系统(第二版)

于慧敏

微处理器原理与接口技术

王汀

自动控制原理(第二版)

王永骥

现代控制理论

王金城

自动检测技术

王化祥

工业过程控制工程

王树青

计算机控制系统(第二版)

王慧

电力拖动与运动控制系统(第二版)

罗飞

电气控制与可编程序控制器

张培志

系统工程导论

梁军

MATLAB与控制系统的数字仿真及CAD

黄道平

计算机仿真技术

吴旭光

人工神经网络理论、设计及应用(第二版)

韩力群

嵌入式操作系统原理与应用

吴旭光

EDA原理与应用(第二版)

付家才

系统工程原理与应用

胡保生

预测控制

钱积新

人工智能控制

蔡自兴

工业过程辨识与控制

李少远

现代检测技术

张宏建

现代检测与控制技术

张志君

自动控制原理(非自动化专业适用)

王敏

过程自动化及仪表(非自动化专业适用)(第二版)

俞金寿

化工自动化及仪表(工艺类专业适用)

杨丽明

化工仪表及自动化(化学工程与工艺专业适用)(第四版)

厉玉鸣

ISBN 978-7-5025-9523-4



9 787502 595234 >

定价：29.80元

2007

TP183/43=2

2007

普通高等教育“十一五”国家级规划教材

人工神经网络理论、设计及应用

第二版

韩力群 编著



化学工业出版社

·北京·



该书系统地论述了人工神经网络的主要理论和设计基础，给出了大量应用实例，旨在使读者了解神经网络的发展背景和研究对象，理解和熟悉其基本原理和主要应用，掌握其结构模型和基本设计方法，为以后的深入研究和应用开发打下基础。

作者连续 11 年为电气信息类专业研究生及本科高年级学生开设“人工神经网络理论与应用”课程，2002 年在多次修改讲义和多项科研成果基础上形成本书的第一版。本书第二版对原书约 1/3 的内容进行了更新，对保留内容进行了修改。取材注意内容的典型性和先进性，编排注意内容的逻辑性，阐述注重物理概念的清晰性，举例与思考练习的安排注意了内容的实践性，常用神经网络及算法的介绍着重于实用性。

本书适合高等院校电气信息类专业和经济管理类专业的研究生、本科生以及科研人员阅读。

图书在版编目(CIP)数据

人工神经网络理论、设计及应用/韩力群编著. —2 版.
北京：化学工业出版社，2007.7

普通高等教育“十一五”国家级规划教材

ISBN 978-7-5025-9523-4

I. 人… II. 韩 III. 人工神经元网络-高等学校-教材 IV. TP183

中国版本图书馆 CIP 数据核字 (2007) 第 096047 号

责任编辑：唐旭华

责任校对：徐贞珍

文字编辑：焦欣渝

装帧设计：韩飞

出版发行：化学工业出版社（北京市东城区青年湖南街 13 号 邮政编码 100011）

印 刷：大厂聚鑫印刷有限责任公司

装 订：三河市延风装订厂

787mm×1092mm 1/16 印张 16 字数 389 千字 2007 年 9 月北京第 2 版第 1 次印刷

购书咨询：010-64518888（传真：010-64519686） 售后服务：010-64518899

网 址：<http://www.cip.com.cn>

凡购买本书，如有缺损质量问题，本社销售中心负责调换。

定 价：29.80 元

版权所有 违者必究

序

中国人工智能学会副理事长、北京工商大学信息工程学院韩力群教授在多年教学与科研工作基础上，编著的“十一五”国家级规划教材《人工神经网络理论、设计及应用》，是一本内容丰富、结构合理、深入浅出、学用结合的好书，受到广大师生和读者的欢迎与好评。为了满足高等院校“人工神经网络”精品课程建设的需要，将修订再版。特赋小诗，以表祝贺：

欣闻教材出新版
精品课程有靠山
神经网络善学习
智能科技喜接班

中国人工智能学会 荣誉理事长

涂序彦

2007年2月26日



前　　言

人类具有高度发达的大脑，大脑是人类思维活动的物质基础，而思维是人类智能的集中体现。长期以来，人们想方设法了解人脑的工作机理和思维的本质，向往能构造出具有类人智能的人工智能系统，以模仿人脑功能，完成类似于人脑的工作。人脑的思维有逻辑思维、形象思维和灵感思维三种基本方式。逻辑思维的基础是概念、判断与推理，即将信息抽象为概念，再根据逻辑规则进行逻辑推理。由于概念可用符号表示，而逻辑推理宜按串行模式进行，这一过程可以事先写成串行的指令由机器来完成。可以认为，20世纪40年代问世的第一台电子计算机就是这样一种用机器模拟人脑逻辑思维的人工智能系统，也是人类实现这一追求的重要里程碑。

众所周知，现代计算机构成单元的速度是人脑中神经元速度的几百万倍，对于那些特征明确，推理或运算规则清楚的可编程问题，可以高速有效地求解，在数值运算和逻辑运算方面的精确与高速极大地拓展了人脑的能力，从而在信息处理和控制决策等各方面为人们提供了实现智能化和自动化的先进手段。但由于现有计算机是按照冯·诺依曼原理，基于程序存取进行工作的，历经半个多世纪的发展，其结构模式与运行机制仍然没有跳出传统的逻辑运算规则，因而在很多方面的功能还远不能达到人的智能水平。随着现代信息科学与技术的飞速发展，这方面的问题日趋尖锐，促使科学和技术专家们寻找解决问题的新出路。

当人们的思路转向研究大自然造就的精妙的人脑结构模式和信息处理机制时，推动了脑科学的深入发展以及人工神经网络和脑模型的研究。随着对生物脑的深入了解，人工神经网络获得长足发展。在经历了漫长的初创期和低潮期后，人工神经网络终于以其不容忽视的潜力与活力进入了发展高潮。60多年来，它的结构与功能逐步改善，运行机制渐趋成熟，应用领域日益扩大，在解决各行各业的难题中显示出巨大的潜力，取得了丰硕的成果。

正是由于人工神经网络是一门新兴的学科，它在理论、模型、算法、应用和实现等方面都还有很多空白点需要努力探索、研究、开拓和开发。因此，许多国家的政府和企业都投入了大量的资金，组织大量的科学和技术专家对人工神经网络的广泛问题立项研究。从人工神经网络的模拟程序和专用芯片的不断推出、论文的大量发表以及各种应用的报道可以看到，在这个领域里一个百花齐放、百家争鸣的局面已经形成。我国对人工神经网络的研究起步于20世纪70年代末期，90年代以来发展迅速。目前，人工神经网络已在我国科研、生产和生活中产生了普遍而巨大的影响。

为了适应人工神经网络应用不断深化的形势，作者在多年为研究生讲授“人工神经网络”课程的基础上，撰写了本书的第一版。自2002年1月本书第一版发行以来，受到广大读者的欢迎和好评，许多院校用作教材或参考书，近百篇学术论文作者将其列为参考文献，众多研究人员和师生与作者联系交流，一些专业网站上刊登了读者的评价与推介。考虑到近年来神经网络的理论与应用又有了进一步发展，首先，随着应用的日益深入普及，许多网络模型及算法已成为解决特定类型问题的常用工具，需要在教材中得到体现；其次，作者近4年来使用本书第一版进行教学的过程中，积累了新的体会和经验，希望融入教材与同行共享。

基于以上考虑，本书第二版在第一版基础上进行了较大的修订，增加了径向基函数网络、学习向量量化、主分量分析和支持向量机等内容，删除了自适应线性单元、神经网络的硬件实现、人工神经系统等内容。对第一版中保留的内容进行了改写和修订，并补充了许多应用实例和练习题。

第二版全书共分 10 章。第 1 章对人脑与计算机信息处理能力与机制进行了比较，归纳了人脑生物神经网络的基本特征与功能，介绍了人工神经网络的发展简史及主要应用领域。第 2 章介绍了人工神经网络的基础知识，包括生物神经元信息处理机制、人工神经元模型、人工神经网络模型以及几种常用学习算法。第 3 章介绍了单层感知器和多层感知器，从几何意义上对单层感知器求解线性可分问题进行了分析，作为读者学习神经网络并建立自学习、自适应和自组织等概念的基础；重点论述了基于误差反向传播算法的多层前馈网络的拓扑结构、算法原理、设计方法及应用实例。第 4 章介绍了竞争学习的概念与原理，在此基础上论述了自组织特征映射网（SOM）和自适应共振网（ART）两种重要的自组织神经网络的结构、原理及算法，并重点介绍了自组织特征映射网络的设计与应用。第 5 章介绍了两种将竞争学习与监督学习两种思想相组合的网络，包括学习向量量化网（LVQ）和对向传播网（CPN）。第 6 章介绍了几种反馈神经网络，包括用于联想记忆离散型 Hopfield 网络与用于优化计算的连续型 Hopfield 网络、双向联想记忆网络、随机神经网络 Boltzmann 机，以及用于动态信号与系统处理的递归神经网络。第 7 章简要介绍智能控制中常用的局部逼近神经网络——小脑模型控制器。上述章节阐述的神经网络为应用最广的经典网络类型，建议作为神经网络的核心教学内容。第 8 章集中讨论了 3 种基于数学原理的人工神经网络，包括用于非线性映射与分类的径向基函数网络的原理及应用，用于特征提取的主分量分析神经网络，以及用于分类的支持向量机神经网络，这部分可作为学习人工神经网络的扩充内容。第 9 章介绍了神经网络系统设计与开发方面的基本知识。本书在前 7 章后面配有思考与练习题，附录 1 给出神经网络的 C 语言算法源程序，附录 2 给出人工神经网络领域常用术语的中英文对照表。

本书内容已制作成用于教学的多媒体课件，需要的读者可登陆北京工商大学网站 <http://www.btbu.edu.cn/>，在“精品课程”栏目中找到“人工神经网络”课程的相关材料。

本书第一版曾得到清华大学阎平凡教授的指点和帮助，为第二版奠定了较好的基础，作者借此机会对阎教授再次表示感谢。此外，作者特别感激那些以此书第一版作为教材的同行和研究生们，他们对本书第一版的肯定和种种有益的建议为第二版的出版提供了强大的支持。

人工神经网络领域的理论与技术正处于蓬勃发展时期，由于作者水平所限，书中难免存在不足之处，恳请各位读者指正。

编著者

2007 年 4 月

目 录

1 绪论	1
1.1 人脑与计算机	1
1.1.1 人脑与计算机信息处理能力的比较	2
1.1.2 人脑与计算机信息处理机制的比较	3
1.1.3 什么是人工神经网络	4
1.2 人工神经网络发展简史	5
1.2.1 启蒙时期	5
1.2.2 低潮时期	7
1.2.3 复兴时期	9
1.2.4 新时期	10
1.2.5 国内研究概况	11
1.3 神经网络的基本特征与功能	14
1.3.1 神经网络的基本特征	14
1.3.2 神经网络的基本功能	14
1.4 神经网络的应用领域	16
1.4.1 信息处理领域	16
1.4.2 自动化领域	16
1.4.3 工程领域	17
1.4.4 经济领域	17
1.4.5 医学领域	18
本章小结	19
思考与练习	19
2 神经网络基础知识	20
2.1 人工神经网络的生物学基础	20
2.1.1 生物神经元的结构	20
2.1.2 生物神经元的信息处理机理	21
2.2 人工神经元模型	24
2.2.1 神经元的建模	24
2.2.2 神经元的数学模型	25
2.2.3 神经元的转移函数	26
2.3 人工神经网络模型	27
2.3.1 网络拓扑结构类型	28
2.3.2 网络信息流向类型	29
2.4 神经网络学习	30
2.4.1 Hebb 学习规则	31
2.4.2 Perceptron 学习规则	33

2.4.3 δ 学习规则	33
2.4.4 LMS 学习规则	35
2.4.5 Correlation 学习规则	35
2.4.6 Winner-Take-All 学习规则	35
2.4.7 Outstar 学习规则	36
本章小结	36
思考与练习	37
3 监督学习神经网络	38
3.1 单层感知器	38
3.1.1 感知器模型	38
3.1.2 单节点感知器的功能分析	39
3.1.3 感知器的学习算法	42
3.1.4 感知器的局限性及解决途径	44
3.2 基于误差反传的多层感知器——BP 神经网络	47
3.2.1 BP 网络模型	47
3.2.2 BP 学习算法	48
3.2.3 BP 算法的程序实现	51
3.2.4 BP 网络的主要能力	52
3.2.5 误差曲面与 BP 算法的局限性	53
3.3 BP 算法的改进	54
3.3.1 增加动量项	54
3.3.2 自适应调节学习率	54
3.3.3 引入陡度因子	55
3.4 BP 网络设计基础	55
3.4.1 网络信息容量与训练样本数	55
3.4.2 训练样本集的准备	56
3.4.3 初始权值的设计	60
3.4.4 BP 网络结构设计	60
3.4.5 网络训练与测试	61
3.5 BP 网络应用与设计实例	62
3.5.1 BP 网络用于催化剂配方建模	62
3.5.2 BP 网络用于汽车变速器最佳挡位判定	63
3.5.3 BP 网络用于图像压缩编码	64
3.5.4 BP 网络用于水库优化调度	64
3.5.5 BP 网络用于证券预测	65
3.5.6 BP 网络用于信用评价模型及预警	66
本章小结	67
思考与练习	67
4 竞争学习神经网络	71
4.1 竞争学习的概念与原理	71
4.1.1 基本概念	71

4.1.2 竞争学习原理	73
4.2 自组织特征映射神经网络	76
4.2.1 SOM 网的生物学基础	76
4.2.2 SOM 网的拓扑结构与权值调整域	76
4.2.3 自组织特征映射网的运行原理与学习算法	77
4.3 自组织特征映射网络的设计与应用	81
4.3.1 SOM 网的设计基础	81
4.3.2 设计与应用实例	83
4.4 自适应共振理论	89
4.4.1 ART I 型网络	90
4.4.2 ART I 型网络的应用	94
4.4.3 ART II 型网络	98
4.4.4 ART II 型网络的应用	101
本章小结	104
思考与练习	105
5 组合学习神经网络	107
5.1 学习向量量化神经网络	107
5.1.1 向量量化	107
5.1.2 LVQ 网络结构与工作原理	107
5.1.3 LVQ 网络的学习算法	108
5.1.4 LVQ 网络的设计与应用	110
5.2 对向传播神经网络	112
5.2.1 网络结构与运行原理	112
5.2.2 CPN 的学习算法	113
5.2.3 改进的 CPN 网举例	115
5.2.4 CPN 网的应用	116
本章小结	117
思考与练习	118
6 反馈神经网络	119
6.1 离散型 Hopfield 神经网络	119
6.1.1 网络的结构与工作方式	119
6.1.2 网络的稳定性与吸引子	120
6.1.3 网络的权值设计	126
6.1.4 网络的信息存储容量	127
6.2 连续型 Hopfield 神经网络	128
6.2.1 网络的拓扑结构	128
6.2.2 能量函数与稳定性分析	129
6.3 Hopfield 网络应用与设计实例	130
6.3.1 应用 DHNN 网解决联想问题	130
6.3.2 应用 CHNN 网解决优化计算问题	130
6.4 双向联想记忆神经网络	134

6.4.1 BAM 网结构与原理	134
6.4.2 能量函数与稳定性	135
6.4.3 BAM 网的权值设计	136
6.4.4 BAM 网的应用	137
6.5 随机神经网络	138
6.5.1 模拟退火原理	139
6.5.2 Boltzmann 机	140
6.6 递归神经网络	144
6.6.1 递归网络模型	144
6.6.2 递归网络的学习算法	146
6.6.3 递归网络应用举例	151
本章小结	152
思考与练习	152
7 小脑模型神经网络	154
7.1 CMAC 网络的结构	154
7.2 CMAC 网络的工作原理	155
7.2.1 从 X 到 M 的映射	155
7.2.2 从 M 到 A 的映射	157
7.2.3 从 A 到 A_p 的映射	158
7.2.4 从 A_p 到 F 的映射	158
7.3 CMAC 网络的学习算法	159
7.3.1 CMAC 网络的学习算法	159
7.3.2 CMAC 网络的特点	159
7.4 CMAC 网络的应用	160
7.4.1 CMAC 网络在机器人手臂协调控制中的应用	160
7.4.2 CMAC 网络在有源噪声控制中的应用	161
本章小结	163
思考与练习	163
8 基于数学原理的神经网络	164
8.1 径向基函数 RBF	164
8.1.1 基于径向基函数技术的函数逼近与内插	164
8.1.2 正则化 RBF 神经网络	166
8.1.3 广义 RBF 神经网络	167
8.1.4 RBF 网络与 BP 网络的比较	171
8.1.5 RBF 网络设计应用实例	172
8.2 主分量分析	177
8.2.1 主分量分析方法概述	177
8.2.2 前向 PCA 神经网络及学习算法	181
8.2.3 侧向连接自适应 PCA 神经网络及 APEX 算法	183
8.3 支持向量机	184
8.3.1 支持向量机的基本思想	184

8.3.2 支持向量机神经网络	188
8.3.3 支持向量机的学习算法	189
8.3.4 支持向量机处理 XOR 问题	190
本章小结	191
9 神经网络的系统设计与软件实现	193
9.1 神经网络系统总体设计	193
9.1.1 神经网络的适用范围	193
9.1.2 神经网络的设计过程与需求分析	194
9.1.3 神经网络的性能评价	195
9.1.4 输入数据的预处理	197
9.2 神经网络的软件实现	198
9.2.1 软件运行的若干问题	198
9.2.2 软件实现的若干问题	199
9.3 神经网络的高级开发环境	200
9.3.1 神经网络的开发环境及其特征	201
9.3.2 MATLAB 神经网络工具箱	201
9.3.3 其他神经网络开发环境简介	203
10 神经网络研究展望	206
10.1 人工神经网络研究中的几个问题	206
10.2 人工神经网络研究展望	207
10.2.1 应用研究的新特点——多学科综合	207
10.2.2 实现技术研究的当务之急——神经网络的硬件实现	207
10.2.3 理论研究的新方向——从人工神经网络到人工神经系统	207
附录 1 常用神经网络 C 语言源程序	209
附录 2 神经网络常用术语英汉对照	240
参考文献	242



1 緒論

1.1 人脑与计算机

人类具有高度发达的大脑，大脑是思维活动的物质基础，而思维是人类智能的集中体现。长期以来，人们想方设法了解人脑的工作机理和思维的本质，向往能构造出具有人类智能的人工智能系统，以模仿人脑功能，完成类似于人脑的工作。钱学森先生认为，人脑的思维有逻辑思维、形象思维和灵感思维三种基本方式。逻辑思维的基础是概念、判断与推理，即将信息抽象为概念，再根据逻辑规则进行逻辑推理。由于概念可用符号表示，而逻辑推理宜按串行模式进行，这一过程可以事先写成串行的指令由机器来完成。20世纪40年代问世的第一台电子计算机就是这样一种用机器模拟人脑逻辑思维的人工智能系统，也是人类实现这一追求的重要里程碑。

现代计算机的计算速度是人脑的几百万倍，对于那些特征明确，推理或运算规则清楚的可编程问题，可以高速有效地求解，其在数值运算和逻辑运算方面的精确与高速极大地拓展了人脑的能力。关于“电脑”战胜人脑的最著名的例子是发生在1997年的一次轰动全世界的国际象棋人机大战。媒体是这样报道的：

“1997年5月11日，清晨4时50分（北京时间），一台名为‘深蓝’的超级电脑将棋盘上的一个兵走到C4位置时，人类有史以来最伟大的国际象棋名家卡斯帕罗夫不得不沮丧地承认自己输了。世纪末的一场人机大战终于以计算机的微弱优势取胜。这场比赛是继去年卡斯帕罗夫与IBM的超级电脑‘深蓝’比赛获胜后，与改进型的‘深蓝’的第二次较量。比赛于5月3日~11日在纽约的公平大厦举行。整个比赛引起了全世界传媒的巨大关注。比赛吸引人们注视目光的原因之一是世界象棋冠军卡斯帕罗夫赛前充满信心，发誓要为捍卫人类之优于机器的尊严而战。然而，最后的结果却是他所捍卫的人类尊严在一台冷漠的1.4吨重的庞然大物‘蓝色巨人’面前被无情地击溃了。虽然人类的骄傲可以把这场比赛的结果仍然归咎于人类的胜利，毕竟‘深蓝’自己也是人类所研制出来的一台计算机而已，但人类所创造的工具击溃了人类，并且是在人类引以为傲的智慧领域，这在一定程度上带来了恐惧，并由此引发了一场有关人类创造物与自身关系的深层讨论。‘深蓝’是IBM公司生产的世界上第一台超级国际象棋电脑。是一台RS6000SP2超级并行处理计算机，计算能力惊人，平均每秒可计算棋局变化200万步。”

2006年8月上旬，在中国人工智能学会主办的“庆祝人工智能学科诞生50周年”系列活动中，由5位中国象棋大师组成的联队与浪潮天梭超级计算机进行对弈，计算机又一次战胜了人类棋手，显示了“电脑”对人脑逻辑推理功能的拓展和延伸水平。

然而迄今为止，计算机在解决与形象思维和灵感思维相关的问题时，却显得无能为力。例如人脸识别、骑自行车、打篮球等涉及联想或经验的问题，人脑可以从中体会那些只可意

2 人工神经网络理论、设计及应用

会、不可言传的直觉与经验，可以根据情况灵活掌握处理问题的规则，从而轻而易举地完成此类任务，而计算机在这方面则显十分笨拙。为什么计算机在处理此类问题时表现出来的能力远不及人脑呢？通过以下的比较，从中不难得出答案。

1.1.1 人脑与计算机信息处理能力的比较

电子计算机能够迅速准确地完成各种数值运算和逻辑运算，成为现代社会不可缺少的信息处理工具，被人们称为“电脑”。人脑本质上是一种信息加工器官，而“电脑”则是人类对自己大脑的某些功能进行模拟而设计的一种信息加工机器。比较人脑与“电脑”的信息处理能力会发现，现有“电脑”和人脑还有很大的差距。

1.1.1.1 记忆与联想能力

人脑有大约 1.4×10^{11} 个神经细胞并广泛互联，因而能够存储大量的信息，并具有对信息进行筛选、回忆和巩固的联想记忆能力。人脑不仅能对已学习的知识进行记忆，而且能在外界输入的部分信息刺激下，联想到一系列相关的存储信息，从而实现对不完整信息的自联想恢复，或关联信息的互联想，而这种互联想能力在人脑的创造性思维中起着非常重要的作用。

计算机从一问世起就是按冯·诺依曼（Von Neumann）方式工作的。基于冯·诺依曼方式的计算机是一种基于算法的程序存取式机器，它对程序指令和数据等信息的记忆由存储器完成。存储器内信息的存取采用按顺序寻址的方式。若要从大量存储数据中随机访问某一数据，必须先确定数据的存储单元地址，再取出相应数据。信息一旦存入便保持不变，因此不存在遗忘问题；在某存储单元地址存入新的信息后会覆盖原有信息，因此不可能对其进行回忆；相邻存储单元之间互不相干，“老死不相往来”，因此没有联想能力。

尽管关系数据库或联想汉卡等由软件设计实现的系统也具有一定的联想功能，但这种联想功能不是计算机的信息存储机制所固有的，其联想能力与联想范围取决于程序的查询能力，因此不可能像人脑的联想功能那样具有个性、不确定性和创造性。

1.1.1.2 学习与认知能力

人脑具有从实践中不断抽取知识、总结经验的能力。刚出生的婴儿脑中几乎是一片空白，在成长过程中通过对外界环境的感知及有意识的训练，知识和经验与日俱增，解决问题的能力越来越强。人脑这种对经验作出反应而改变行为的能力就是学习与认知能力。

计算机所完成的所有工作都是严格按照事先编制的程序进行的，因此它的功能和结果都是确定不变的。作为一种只能被动地执行确定的二值命令的机器，计算机在反复按指令执行同一程序时得到的永远是同样的结果，它不可能在不断重复的过程中总结或积累任何经验，因此不会主动提高自己解决问题的能力。

1.1.1.3 信息加工能力

在信息处理方面，人脑具有复杂的回忆、联想和想象等非逻辑加工功能，因而人的认识可以逾越现实条件下逻辑所无法越过的认识屏障，产生诸如直觉判断或灵感一类的思维活动。在信息的逻辑加工方面，人脑的功能不仅局限于计算机所擅长的数值或逻辑运算，而且可以上升到具有语言文字的符号思维和辩证思维。人脑具有的这种高层次的信息加工能力使人能够深入事物内部去认识事物的本质与规律。

计算机没有非逻辑加工功能，因而不能逾越有限条件下逻辑的认识屏障。计算机的逻辑

加工能力也仅限于二值逻辑，因此只能在二值逻辑所能描述的范围内运用形式逻辑，而缺乏辩证逻辑能力。

1.1.1.4 信息综合能力

人脑善于对客观世界千变万化的信息和知识进行归纳、类比和概括，综合起来解决问题。人脑的这种综合判断过程往往是一种对信息的逻辑加工和非逻辑加工相结合的过程。它不仅遵循确定性的逻辑思维原则，而且可以经验地、模糊地甚至是直觉地作出一个判断。大脑所具有的这种综合判断能力是人脑创造能力的基础。

计算机的信息综合能力取决于它所执行的程序。由于不存在能完全描述人的经验和直觉的数学模型，也不存在能完全正确模拟人脑综合判断过程的有效算法，因此计算机难以达到人脑所具有的融会贯通的信息综合能力。

1.1.1.5 信息处理速度

人脑的信息处理是建立在大规模并行处理基础上的，这种并行处理所能够实现的高度复杂的信息处理能力远非传统的以空间复杂性代替时间复杂性的多处理机并行处理系统所能达到的。人脑中的信息处理是以神经细胞为单位，而神经细胞间信息的传递速度只能达到毫秒级，显然比现代计算机中电子元件纳秒级的计算速度慢得多，因此似乎计算机的信息处理速度要远高于人脑，事实上在数值处理等只需串行算法就能解决问题的应用方面确实是如此。然而迄今为止，计算机处理文字、图像、声音等类信息的能力与速度却远远不如人脑。例如，几个月大的婴儿能从人群中一眼认出自己的母亲，而计算机解决这个问题时需要对一幅具有几十万个像素点的图像逐点进行处理，并提取脸谱特征进行识别。又如，一个篮球运动员可以不假思索地接住队友传给他的球，而让计算机控制机器人接球则要判断篮球每一时刻在三维空间的位置坐标、运动轨迹、运动方向及速度等。显然，在基于形象思维、经验与直觉的判断方面，人脑只要零点几秒就可以圆满完成的任务，计算机用几十分钟甚至几小时也不一定能达到人脑的水平。

1.1.2 人脑与计算机信息处理机制的比较

人脑与计算机信息处理能力特别是形象思维能力的差异来源于两者系统结构和信息处理机制的不同。主要表现在以下 4 个方面：

1.1.2.1 系统结构

人脑在漫长的进化过程中形成了规模宏大、结构精细的群体结构，即神经网络。脑科学的研究结果表明，人脑的神经网络是由数百亿神经元相互连接组合而成的。每个神经元相当于一个超微型信息处理与存储机构，只能完成一种基本功能，如兴奋与抑制。而大量神经元广泛连接后形成的神经网络可进行各种极其复杂的思维活动。

计算机是一种由各种二值逻辑门电路构成的按串行方式工作的逻辑机器，它由运算器、控制器、存储器和输入/输出设备组成。其信息处理是建立在冯·诺依曼体系基础上，基于程序存取进行工作的。

1.1.2.2 信号形式

人脑中的信号形式具有模拟量和离散脉冲两种形式。模拟量信号具有模糊性特点，有利于信息的整合和非逻辑加工，这类信息处理方式难以用现有数学方法进行充分描述，因而很难用计算机进行模拟。

4 人工神经网络理论、设计及应用

计算机中信息的表达采用离散的二进制数和二值逻辑形式，二值逻辑必须用确定的逻辑表达式来表示。许多逻辑关系确定的信息加工过程可以分解为若干二值逻辑表达式，由计算机来完成。然而，客观世界存在的事物关系并非都可以分解为二值逻辑的关系，还存在着各种模糊逻辑关系和非逻辑关系。对这类信息的处理计算机是难以胜任的。

1.1.2.3 信息存储

与计算机不同的是，人脑中的信息不是集中存储于一个特定的区域，而是分散地存储于整个系统中。此外，人脑中存储的信息不是相互孤立的，而是联想式的。人脑这种分布式、联想式的信息存储方式使人类非常擅长于从失真和缺省的模式中恢复出正确的模式，或利用给定信息寻找期望信息。

1.1.2.4 信息处理机制

人脑中的神经网络是一种高度并行的非线性信息处理系统。其并行性不仅体现在结构上和信息存储上，而且体现在信息处理的运行过程中。由于人脑采用了信息存储与信息处理一体化的群体协同并行处理方式，信息的处理受原有存储信息的影响，处理后的信息又留在在神经元中成为记忆。这种信息处理 ↔ 存储的构建模式是广泛分布在大量神经元上同时进行的，因此呈现出来的整体信息处理能力不仅能快速完成各种极复杂的信息识别和处理任务，而且能产生高度复杂而奇妙的效果。

计算机采用的是有限集中的串行信息处理机制，即所有信息处理都集中在一个或几个 CPU 中进行。CPU 通过总线同内外存储器或 I/O 接口进行顺序的“个别对话”，存取指令或数据。这种机制的时间利用率很低，在处理大量实时信息时不可避免地会遇到速度“瓶颈”。即使采用多 CPU 并行工作，也只是在一定发展水平上缓解矛盾。

1.1.3 什么是人工神经网络

综上所述，计算机在解决具有形象思维特点的问题时难以胜任的根本原因在于，计算机与人脑采取的信息处理机制完全不同。

迄今为止，各代计算机都是基于冯·诺依曼工作原理：其信息存储与处理是分开的，即存储器与处理器相互独立；处理的信息必须是形式化信息，即用二进制编码定义的文字、符号、数字、指令和各种规范化的数据格式、命令格式等；而信息处理的方式必须是串行的，即 CPU 不断地重复取址、译码、执行、存储这四个步骤。这种计算机的结构和串行工作方式决定了它只擅长于数值和逻辑运算。

人类的大脑大约有 1.4×10^{11} 个神经细胞，亦称为神经元。每个神经元有数以千计的通道同其他神经元广泛相互连接，形成复杂的生物神经网络。生物神经网络以神经元为基本信息处理单元，对信息进行分布式存储与加工，这种信息加工与存储相结合的群体协同工作方式使得人脑呈现出目前计算机无法模拟的神奇智能。为了进一步模拟人脑的形象思维方式，人们不得不跳出冯·诺依曼计算机的框架另辟蹊径。而从模拟人脑生物神经网络的信息存储、加工处理机制入手，设计具有人类思维特点的智能机器，无疑是最有希望的途径之一。

用计算方法对神经网络信息处理规律进行探索称为计算神经科学，该方法对于阐明人脑的工作原理具有深远意义。人脑的信息处理机制是在漫长的进化过程中形成和完善的。虽然近年来，在细胞和分子水平上对脑结构和脑功能的研究已经有了长足的发展。然而到目前为止，人类对神经系统内的电信号和化学信号是怎样被用来处理信息的只有模糊的概念。尽管如此，把通过分子和细胞水平的技术所达到的微观层次与通过行为研究达到的系统层次结合

起来，可以形成对人脑神经网络的基本认识。在此基本认识的基础上，以数学和物理方法以及信息处理的角度对人脑神经网络进行抽象，并建立某种简化模型，就称为人工神经网络（artificial neural network, ANN）。人工神经网络远不是人脑生物神经网络的真实写照，而只是对它的简化、抽象与模拟。揭示人脑的奥妙不仅需要各学科的交叉和各领域专家的协作，还需要测试手段的进一步发展。尽管如此，目前已提出上百种人工神经网络模型。令人欣慰的是，这种简化模型的确能反映出人脑的许多基本特性。它们在模式识别、系统辨识、信号处理、自动控制、组合优化、预测预估、故障诊断、医学与经济学等领域已成功地解决了许多现代计算机难以解决的实际问题，表现出良好的智能特性和潜在的应用前景。

目前关于人工神经网络的定义尚不统一，例如，美国神经网络学家 Hecht Hielsen 关于人工神经网络的一般定义是：“神经网络是由多个非常简单的处理单元彼此按某种方式相互连接而形成的计算系统，该系统是靠其状态对外部输入信息的动态响应来处理信息的。”美国国防高级研究计划局关于人工神经网络的解释是：“人工神经网络是一个由许多简单的并行工作的处理单元组成的系统，其功能取决于网络的结构、连接强度以及各单元的处理方式。”综合人工神经网络的来源、特点及各种解释，可以简单表述为：人工神经网络是一种旨在模仿人脑结构及其功能的信息处理系统。为叙述简便，后面常将人工神经网络简称为神经网络。

1.2 人工神经网络发展简史

神经网络的研究可追溯到 19 世纪末期，其发展历史可分为四个时期：第一个时期为启蒙时期，开始于 1890 年美国著名心理学家 W. James 关于人脑结构与功能的研究，结束于 1969 年 Minsky 和 Papert 出版《感知器》（*Perceptrons*）一书，第二个时期为低潮时期，开始于 1969 年，结束于 1982 年 J. J. Hopfield 发表著名的文章《神经网络和物理系统》（*Neural Network and Physical System*）；第三个时期为复兴时期，开始于 J. J. Hopfield 的突破性研究论文，结束于 1986 年 D. E. Rumelhart 和 J. L. McClelland 领导的研究小组编写出版的《并行分布式处理》（*Parallel Distributed Processing*）一书；第四个时期为高潮时期，以 1987 年首届国际人工神经网络学术会议为开端，迅速在全世界范围内掀起人工神经网络的研究应用热潮，至今势头不衰。

下面按年代顺序介绍对人工神经网络研究有重大贡献的学者及其著作，以使读者在了解神经网络的发展历史时看到它与神经生理学、数学、电子学、计算机科学以及人工智能学科之间的千丝万缕的联系，也使读者对神经网络的某些概念有粗略的了解。

1.2.1 启蒙时期

1890 年，美国心理学家 William James (1842—1910) 出版了第一部详细论述人脑结构及功能的专著《心理学原理》（*Principles of Psychology*），对与学习、联想记忆相关的基本原理做了开创性研究。James 指出：“让我们假设所有后继推理的基础遵循这样的规则：当两个基



William James

6 人工神经网络理论、设计及应用

本的脑细胞曾经一起或相继被激活过，其中一个受刺激重新激活时会将刺激传播到另一个。”这一点与联想记忆和相关学习关系最密切。另外，他曾预言神经细胞激活是细胞所有输入叠加的结果。他认为，在大脑皮层上任意点的刺激量是其他所有发射点进入该点刺激的总和。

半个世纪后，生理学家 W. S. McCulloch 和数学家 W. A. Pitts 于 1943 年发表了一篇神经网络方面的著名文章。在这篇文章中，他们在已知的神经细胞生物学基础上，从信息处理的角度出发，提出形式神经元的数学模型，称为 M-P 模型。该模型把神经细胞的动作描述为：①神经元的活动表现为兴奋或抑制的二值变化；②任何兴奋性突触有输入激励后，使神经元兴奋，与神经元先前的动作与状态无关；③任何抑制性突触有输入激励后，使神经元抑制；④突触的值不随时间改变；⑤突触从感知输入到传送出一个输出脉冲的延迟时间是 0.5ms。尽管现在看来 M-P 模型过于简单，而且其观点也并非完全正确，但其理论贡献在于：① McCulloch 和 Pitts 证明了任何有限逻辑表达式都能由 M-P 模型组成的人工神经网络来实现；② 他们是从 W. James 以来最早采用大规模并行计算结构描述神经元和网络的学者；③ 他们的工作奠定了网络模型和以后开发神经网络步骤的基础。为此，M-P 模型被认为开创了神经科学理论研究的新时代。



Donald Olding Hebb

启蒙时期的另一位重要学者是心理学家 Donald Olding Hebb，他在 1949 年出版了一本名为《行为构成》(*Organization of Behavior*) 的书。在该书中他首先建立了人们现在称为 Hebb 算法的连接权训练算法。他也是首先提出“连接主义”(connectionism) 这一名词的人之一，这一名词的含义为大脑的活动是靠脑细胞的组合连接实现的。Hebb 认为，如果源和目的神经元均被激活兴奋时，它们之间突触的连接强度将会增强。这就是最早且最著名的 Hebb 训练算法的生理学基础。Hebb 对神经网络理论作出的四点主要贡献是：①指出在神经网络中，信息存储在连接权中；②假设连接权的学习（训练）速率正比于神经元各活化值之积；

③假定连接是对称的，也就是从神经元 A 到神经元 B 的连接权与从 B 到 A 的连接权是相同的（虽然这一点在神经网络中未免过于简单化，但它往往应用到人工神经网络的各种现实方案中）；④提出细胞连接的假设，并指出当学习训练时，连接权的强度和类型发生变化，且由这种变化建立起细胞间的连接。Hebb 提出的这四点看法，在当今的人工神经网络中至少在某种程度上都得到了实现。

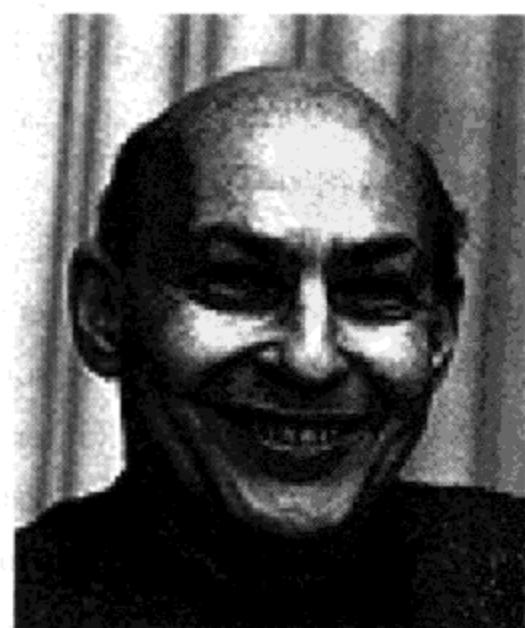
1958 年计算机学家 Frank Rosenblatt (1928—1969) 发表一篇有名的文章，提出了一种具有三层网络特性的神经网络结构，称为“感知机”(perceptron)。这或许是世界上第一个真正优秀的人工神经网络，这一神经网络是用一台 IBM704 计算机模拟实现的。从模拟结果可以看出，感知机具有通过学习改变连接权值，将类似的或不同的模式进行正确分类的能力，因此也称它为“学习的机器”。Rosenblatt 用感知机来模拟一个生物视觉模型，输入节点群由视网膜上某一范围内细胞的随机集合组成。每个细胞

连到下一层内的联合单元 (association unit, AU)。AU 双向连接到第三层 (最高层) 中的响应单元 (response unit, RU)。感知机的目的是对每一实际的输入去激活正确的 RU。Rosenblatt 利用他的感知机模型说明两个问题：一个问题是信息存储或记忆采用什么形式，他认为信息被包含在相互连接或联合之中，而不是反映在拓扑结构的表示法中；另一个问题是如何存储影响认知和行为的信息，他的回答是，存储的信息在神经网络系统内开始形成新的连接或传送链路后，新的刺激将会通过这些新建立的链路自动地激活适当的响应部分，而不要求任何识别或鉴定它们的过程。这种原始的感知学习机在激励-响应特性方面是“自组织”或“自联合”的。在“自组织”响应中被响应的节点，起初是随机的，然后逐渐地通过彼此竞争而形成支配的统治地位。这篇文章提出的算法与后来的反向传播算法和 Kohonen 的自组织算法类似，因此 Rosenblatt 所发表的网络基本结构是相当有活力，尽管后来它遭到 Minsky 和 Papert 的抨击。

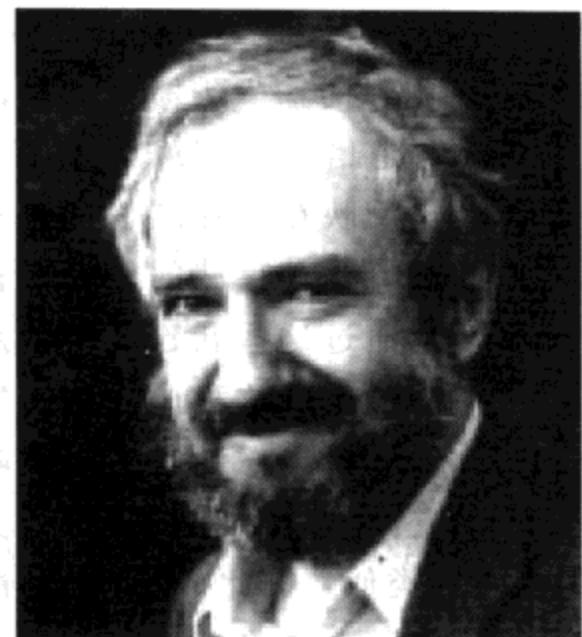
启蒙时期的最后两位代表人物是电机工程师 Bernard Widrow 和 Marcian Hoff。1960 年他们发表了一篇题为《自适应开关电路》(Adaptive Switching Circuits) 的文章。从工程技术的角度看，这篇文章是神经网络技术发展中极为重要的文章之一。Widrow 和 Hoff 不仅设计了在计算机上仿真的人工神经网络，而且还用硬件电路实现了他们的设计。Widrow 和 Hoff 提出一种称为“Adaline”的模型，即自适应线性单元 (adaptive linear)。Adaline 是一种累加输出单元，输出值为 ± 1 的二值变量，权在 Widrow 和 Hoff 的文章中称为增益 (gain)。他们用这一名称反映他们工程学的背景，因为增益是指电信号通过放大器所放大的倍数。这比一般称为权也许更能说明它所起的作用，且更容易被工程技术人员所理解。Adaline 精巧的地方是 Widrow-Hoff 的学习训练算法，它根据加法器输出端误差大小来调整增益，使得训练期内所有样本模式的平方和最小，因而速度较快且具有较高的精度。由于这一原因，Widrow-Hoff 算法也称为 δ (误差大小) 算法或最小均方 (LMS) 算法，在数学上就是人们熟知的梯度下降法。Widrow-Hoff 指出，如果用计算机建立自适应神经元，它的具体结构可以由设计者通过训练给出，而不是通过直接设计来确定。他们用硬件电路实现人工神经网络方面的工作为今天用超大规模集成电路实现神经网络计算机奠定了基础。他们是开发神经网络硬件最早的主要贡献者。

1.2.2 低潮时期

在 20 世纪 60 年代，掀起了神经网络研究的第一次热潮。由于当时对神经网络的学习能力的估计过于乐观，而随着神经网络研究的深入开展，人们遇到了来自认识方面、应用方面和实现方面的各种困难和迷惑，使得一些人产生了怀疑和失望。人工智能的创始人之一，M. Minsky 和 S. Papert 研究数年，对以感知器为代表的网络系统的功能及其局限性从数学上作了深入研究，于 1969 年出版了轰动一时的评论人工神经网络的书——《感知器》(Perceptrons)。该书指出，简单的神经网络只能运用于线性问题的求解，能够求解非线性问题的网络应具有隐层，而从理论上还不能证明将感知机模型扩展到多层网络



Marvin Minsky



Seymour Papert

是有意义的。由于 Minsky 在学术界的地位和影响，其悲观论点极大地影响了当时的人工神经网络研究，为刚燃起的人工神经网络之火，泼了一大盆冷水。不久，几乎所有为神经网络提供的研究基金都枯竭了，很多领域的专家纷纷放弃了这方面课题的研究，开始了神经网络发展史上长达 10 年的低潮时期。

使神经网络研究处于低潮的更重要的原因是，20 世纪 70 年代以来集成电路和微电子技术的迅猛发展，使传统的 Von Neumann 型计算机进入发展的全盛时期，基于逻辑符号处理方法的人工智能得到迅速发展并取得显著成就，它们的问题和局限性尚未暴露，因此暂时掩盖了发展新型计算机和寻求新的神经网络的必要性和迫切性。

在 Minsky 和 Papert 的书出版后的 10 年中，神经网络研究园地中辛勤耕耘的研究人员数量大幅度减少，但仍有为数不多的学者在困难时期坚持致力于神经网络的研究。他们在极端艰难的条件下作出难能可贵的扎实奉献，为神经网络研究的复兴与发展奠定了理论基础。

1969 年，美国波士顿大学自适应系统中心的 S. Grossberg 教授和他的夫人 G. A. Carpenter 提出了著名的自适应共振理论 (adaptive resonance theory) 模型。在 Grossberg 早期著作中介绍的原理，有许多已用在当前的神经网络中。其中的基本论点是：若在全部神经结点中有一个神经结点特别兴奋，其周围的所有结点将受到抑制。这种周围抑制的观点也用在 Kohonen 的自组织网络中。Grossberg 对网络的记忆理论也作出很大的贡献。他提出了关于短期记忆和长期记忆的机理，以及短期记忆如何与神经结点的激活值有关，而长期记忆如何与连接权有关。结点的激活值与连接权都会随时间的衰减而衰减，具有“忘却”特性。结点激活值的衰减相当快（短期记忆），而连接权有较长的记忆能力，衰减较慢。在其后的若干年里，Grossberg 和 Carpenter 发展了他们的自适应共振理论，并有 ART I、ART II、ART III 三个 ART 系统的版本。ART I 网络只能处理二值的输入。ART II 比起 ART I 复杂但能处理连续值输入。ART III 是一种强有力的神经网络研究模型，但由于 ART III 的复杂性，不能作为一种比较实用的神经网络工具。

1972 年，两位研究者分别在欧洲和美洲异地发表了类似的神经网络开发结果：一位是芬兰的 T. Kohonen 教授，提出了自组织映射 (SOM) 理论，并称其神经网络结构为“联想存储器” (associative memory)；另一位是美国的神经生理学家和心理学家 J. Anderson，提出了一个类似的神经网络，称为“交互存储器” (interactive memory)。他们在网络结构，学习算法和传递函数方面的技术几乎是相同的。今天的神经网络主要是根据 Kohonen 的工作来实现的，因为 SOM 模型是一类非常重要的无导师学习网络，主要应用于模式识别、语音识别、分类等场合。而 Anderson 的主要兴



Teuvo Kohonen

趣在对网络结构与训练算法的生物仿真性及模型的研究。在 Kohonen 1972 年发表的文章中首先值得注意的是，他所用的神经结点或处理单元是线性连续的，而不是 McCulloch, Pitts 和 Widrow-Hoff 提出的二进制方式。再一点值得注意的是，Kohonen 网络用许多邻近的同时激活的输入与输出结点。这一类结点，在分析可视图像和语言声谱时是非常需要的。在这种情况下，不是由单个“优胜”神经元的动作电位来表示网络的输出，而是用相当大数目的一组输出神经结点来表示输入模式的分类，这使得网络能更好地进行概括推论且减少噪声的影响。最值得注意的是，文章提出的神经网络类型与先前提出的感知机有很大的不同。目前用得最普遍的实用多层感知机（误差反传网络）的学习训练是一种有指导的训练。而各种 Kohonen 网络形式被认为是自组织的网络，它的学习训练方式是无指导训练。这种学习训练方式往往是当不知有哪些分类类型存在时，用作提取分类信息的一种训练。

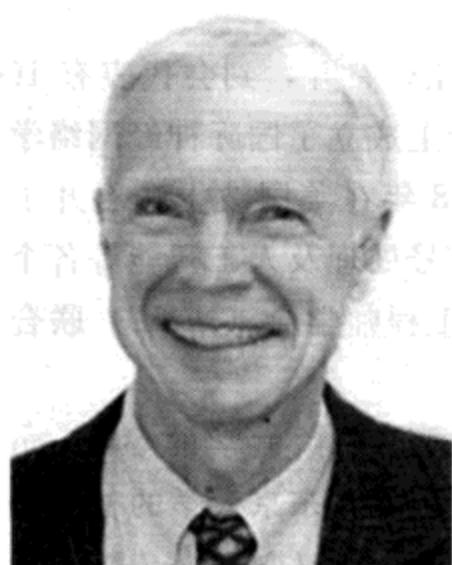
低潮时期第三位重要的研究者是日本东京 NHK 广播科学实验室的福岛邦彦 (Kunihiko Fukushima)。他开发了一些神经网络结构与训练算法，其中最有名的是 1980 年发表的《新认知机》(Neocognitron)。此后还有一系列改进的报道文章。“新认知机”是视觉模式识别机制模型，它与生物视觉理论相符合，其目的在于综合出一种神经网络模型，让它像人类一样具有进行模式识别的能力。这类网络起初为自组织的无指导训练，后来采用有指导的训练。福岛邦彦等人在 1983 年发表的文章中承认，有指导的训练方式能更好地反映设计模式识别装置的工程师的立场，而不是纯生物学的模型。福岛邦彦给出的神经认知机，能正确识别手写的 0~9 十个数字。其中包括样本模式变形、不完全的样本模式和受噪声干扰的样本模式等。尽管今天看来这似乎不难，但当时这的确是一项重大的成就。

在整个低潮时期，上述开创性的研究成果和有意义的工作虽然未能引起当时人们的普遍重视，但是其科学价值不可磨灭，它们为神经网络的进一步发展奠定了基础。

1.2.3 复兴时期

进入 20 世纪 80 年代后，经过十几年迅速发展起来的以逻辑符号处理为主的人工智能理论和 Von Neumann 计算机在诸如视觉、听觉、形象思维、联想记忆等智能信息处理问题上受到了挫折，在大型复杂计算方面显示出巨大威力的计算机却很难“学会”人们习以为常的普通知识和经验。这一切迫使人们不得不慎重思考：智能问题是否可以完全由人工智能中的逻辑推理规则来描述？人脑的智能是否可以在计算机中重现？

1982 年，美国加州理工学院的优秀物理学家 John J. Hopfield 博士发表了一篇对神经网络研究的复苏起了重要作用的文章。他总结并吸取前人对神经网络研究的成果与经验，把网络的各种结构和各种算法概括起来，塑造出一种新颖的强有力的网络模型，称为 Hopfield 网络。Hopfield 在 1984 年和 1986 年连续发表了有关其网络应用的文章，获得工程技术界与学术界的重视。Hopfield 没有提出太多的新原理，但他创造性地把前人的观点概括综合在一起。其中最



John J. Hopfield

具有创新意义的是他对网络引用了物理力学的分析方法，把网络作为一种动态系统并研究这种网络动态系统的稳定性。他把李雅普诺夫（Lyapunov）能量函数引入网络训练这一动态系统中。他指出：对已知的网络状态存在一个正比于每个神经元的活动值和神经元之间的连接权的能量函数，活动值的改变向能量函数减小的方向进行，直到达到一个极小值。换句话说，他证明了在一定条件下网络可以到达稳定状态。Hopfield 网络还有一个显著的优点，即与电子电路存在明显的对应关系，使得它易于理解且便于用集成电路来实现。

Hopfield 网络一出现，很快引起半导体工业界的注意。在他 1984 年的文章发表后的三年，美国电话与电报公司的贝尔实验室声称利用 Hopfield 理论首先在硅片上制成硬件的神经计算机网络，继而仿真出耳蜗与视网膜等硬件网络。继 Hopfield 的文章之后，不少研究非线性电路的科学家、物理学家和生物学家在理论和应用上对 Hopfield 网络进行了比较深刻的讨论和改进。G. E. Hinton 和 T. J. Sejnowski 借助统计物理学的概念和方法提出了一种随机神经网络模型——玻尔兹曼（Boltzmann）机，其学习过程采用模拟退火技术，有效地克服了 Hopfield 网络存在的能量局部极小问题。不可否认，是 Hopfield 博士点亮了人工神经网络复兴的火炬，掀起了各学科关注神经网络的热潮。

在 1986 年贝尔实验室宣布制成神经网络芯片前不久，美国的 David E. Rumelhart 和 James L. McClelland 及其领导的研究小组发表了《并行分布式处理》（Parallel Distributed Processing，简称 PDP）一书的前两卷，接着在 1988 年发表了带有软件程序的第三卷。由 Hopfield 燃起的神经网络复苏之火，激起了他们写这部著作的热情，而他们提出的 PDP 网络思想，则为神经网络研究新高潮的到来起到了推波助澜的作用。书中涉及到神经网络的三个主要特征，即结构、神经元的传递函数（也称传输函数，转移函数、激励函数）和它的学习训练方法，这部书介绍了这三方面各种不同的网络类型。这部书最重要的贡献之一是发展了多层感知机的反向传播训练算法，把学习的结果反馈到中间层次的隐节点，改变其连接权值，以达到预期的学习目的。该算法已成为当今影响最大的一种网络学习方法。

这一时期大量而深入的开拓性工作大大发展了神经网络的模型和学习算法，增强了对神经网络系统特性的进一步认识，使人们对模仿脑信息处理的智能计算机的研究重新充满了希望。

1.2.4 新时期

1987 年 6 月，首届国际神经网络学术会议在美国加州圣地亚哥召开，到会代表有 1600 余人。这标志着世界范围内掀起神经网络开发研究的热潮。在会上成立了国际神经网络学会（International Neural Network Society，简称 INNS），并于 1988 年在美国波士顿召开了年会，会议讨论的议题涉及生物、电子、计算机、物理、控制、信号处理及人工智能等各个领域。自 1988 年起，国际神经网络学会和国际电气工程师与电子工程师学会（IEEE）联合召开每年一次的国际学术会议。

这次会议后不久，由世界著名的三位神经网络学家，美国波士顿大学的 Stephen Grossberg 教授、芬兰赫尔辛基技术大学的 Teuvo Kohonen 教授及日本东京大学的甘利俊一（Shunichi Amari）教授，主持创办了世界第一份神经网络杂志《Neural Network》。随后，IEEE 也成立了神经网络协会并于 1990 年 3 月开始出版神经网络会刊。各种学术期刊的神经

网络特刊也层出不穷。

从以上现象可以清楚地看到，神经网络的研究出现了新的高潮，进入新的发展时期。神经网络研究再度掀起高潮，除了神经科学研究本身的突破和进展之外，更重要的动力是计算机科学和人工智能发展的需要，以及 VLSI 技术、生物技术、超导技术和光学技术等领域的迅速发展提供了技术上的可能性。

从 1987 年以来，神经网络的理论、应用、实现及开发工具均以令人振奋的速度快速发展。神经网络理论已成为涉及神经生理科学、认知科学、数理科学、心理学、信息科学、计算机科学、微电子学、光学、生物电子学等多学科交叉、综合的前沿学科。神经网络的应用已渗透到模式识别、图像处理、非线性优化、语音处理、自然语言理解、自动目标识别、机器人、专家系统等各个领域，并取得了令人瞩目的成果。与此同时，美国、日本等国在神经网络计算机的硬件实现方面也取得了一些实实在在的成绩。应当指出，对神经网络及神经计算机的研究决不意味着数字计算机将要退出历史舞台。在智能的、模糊的、随机的信息处理方面，神经网络及神经计算机具有巨大优势；而在符号逻辑推理、数值精确计算等方面，数字计算机仍将发挥其不可替代的作用。两种计算机互为补充、共同发展。从真正的“电脑”意义上来说，未来的智能型计算机应该是精确计算和模糊处理功能均备、逻辑思维和形象思维兼优的崭新计算机。

表 1.1 列出神经网络发展过程中起过重要作用的十几种著名神经网络的情况，它也是神经网络发展史的一个缩影。

1.2.5 国内研究概况

我国最早涉及人工神经网络的著作是涂序彦先生等于 1980 年出版的《生物控制论》一书，书中将“神经系统控制论”单独设为一章，系统地介绍了神经元和神经网络的结构、功能和模型。该书出版时人工神经网络的研究尚未进入复苏时期，国内学术界对该领域的情况知之甚少，研究热点主要集中在人工智能方面。随着人工神经网络 20 世纪 80 年代在世界范围的复苏，国内学术界大约在 80 年代中期也逐步掀起了研究热潮。北京大学非线性研究中心在 1988 年 9 月发起举办了 Beijing International Workshop on Neural Networks: Learning and Recognition, a Modern Approach。1989 年召开了一个非正式的全国神经网络会议。1990 年 2 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”。这次会议以“八学会联盟，探智能奥秘”为主题，收到了来自全国 300 多篇论文，从而开创了我国人工神经网络及神经计算机方面科学的新纪元。1991 年在南京召开了中国神经网络学术大会（第二届），会上成立了中国神经网络学会。我国“863”高技术研究计划和“攀登”计划于 1990 年批准了人工神经网络的 3 项课题，国家自然科学基金和国防科技预研基地也都把神经网络的研究列入选题指南。许多全国性学术年会和学术刊物把神经网络理论及应用领域的论文列为重点。1992 年由国际神经网络学会和 IEEE 神经网络委员会主办的国际性学术会议在北京召开。经过十几年的发展，我国学术界和工程界在人工神经网络的理论研究与应用方面取得了丰硕成果，学术论文、应用成果和研究人员的数量逐年增长。目前，人工神经网络已在我国科研、生产和生活中产生了普遍而巨大的影响。

表 1.1 对神经网络发展有重要影响的神经网络

名称	提出者	诞生年	典型应用领域	弱 点	特 点
Perceptron(感知机)	Frank Rosenblatt (康奈尔大学)	1957	文字识别, 声音识别, 声纳信号识别, 学习记忆问题研究	不能识别复杂字形, 对字的大小、平移和倾斜敏感	最早的神经网络, 已很少应用。有学习能力, 只能进行线性分类
Adaline(自适应线性单元) 和 Madaline(多个 Adaline 的组合网络)	Bernard Widrow (斯坦福大学)	1960~1962	雷达天线控制, 自适应回波抵消, 适应性调制解调, 电话线中适应性补偿等	要求输入-输出之间为线性关系	学习能力较强, 较早开始商业应用, Madaline 是 Adaline 的功能扩展
Avalanche(雪崩网)	S. Grossberg (波士顿大学)	1967	连续语音识别, 机器人手臂运动的教学指令	不易改变运动速度和插入运动	类似于 Avalanche 网络, 能调节和各种指令序列, 按需要缓慢地插入动作
Cerellatron(小脑自动机)	D. Marr(麻省理工学院)	1969~1982	控制机器人的手臂运动	需要复杂的控制输入	多层次前馈网络, 采用最小均方差学习方式, 是应用最广泛的网络
Back Propagation (误差反传网络)	P. Werbos(哈佛大学) David Rumelhart(斯坦福大学) James McClelland(斯坦福大学)	1974~1985	语音识别, 工业过程控制, 贷款信用评估, 股票预测, 自适应控制等	需要大量输入-输出数据, 训练时间长, 易陷入局部极小	ART I 用于二进制, ART II 用于连续信号
Adaptive Resonance Theory (自适应共振理论 ART) 有 ART I, ART II 和 ART III 3 种类型	G. Carpenter and S Grossberg (波士顿大学)	1976~1990	模式识别领域, 擅长识别复杂模式或未知的模式	受平移、旋转及尺度的影响。系统比较复杂, 难以用硬件实现	可以对任意多和任意复杂的二维模式进行自组织学习, ART I 用于二进制, ART II 用于连续信号
Brain State in a Box (盒中脑 BSB 网络)	James Anderson(布朗大学)	1977	解释概念形成, 分类和知识处理	只能作一次性决策, 无重复性共振	具有最小均方差的单层自联想网络, 类似于双向联想记忆, 可对片段输入补充

续表

名称	提出者	诞生年	典型应用领域	弱点	特点
Neocognition (新认知机)	Fukushima K 福岛邦彦 (日本广播协会)	1978~1984	手写字母识别	需要大量加工单元和联系	多层次结构化字符识别网络, 与输入模式的大小、平移和旋转无关, 能识别复杂字形
Self-Organizing feature map (自组织特征映射网络)	Tuevo Konhonnen (芬兰赫尔辛基技术大学)	1980	语音识别, 机器人控制, 工业过程控制, 图像压缩, 专家系统等	模式类型数需预先知道	对输入样本自组织聚类, 映射样本空间的分布
Hopfield 网络	John Hopfield (加州理工学院)	1982	求解 TSP 问题, 线性规划, 联想记忆和用于辨识	无学习能力, 连接要对称, 权值要预先给定	单层自联想网络, 可从有缺损或有噪声输入中恢复完整信息
Boltzman machine (玻尔兹曼机)	J. Hinton (多伦多大学) T. Sejnowski (霍布金斯大学)	1985~1986	图像、声纳和雷达等模式识别	玻尔兹曼机训练时间长, 柯西机在某些统计分布下产生噪声	一种采用随机学习算法的网络, 可训练时实现全局最优
Bidirectional Associative Memory (BAM, 双向联想记忆网)	Baart Kosko (南加州大学)	1985~1988	内容寻址的联想记忆	存储的密度低, 数据必须适应编码	双向联想式单层网络, 具有学习功能, 简单易学
Counter Propagation (CPN, 双向传播网)	Robert Hecht-Nielsen	1986	神经网络计算机, 图像分析和统计分析	需要大量处理单元和连接, 需要高度准确	一种在功能上作为统计最优化和概率密度函数分析的网络
Radial Basis Functions (RBF, 径向基函数网络)	Broomhead Lowe	1988	非线性函数逼近, 时间序列分析, 模式识别、信息处理、图像处理、系统建模	需要大量输入-输出数据, 计算较复杂	网络设计采用原理化方法, 有坚实的数学基础
Support Vector Machine (SVM, 支持向量机)	Vapnik	1992~1998	模式分类, 非线性映射	训练时间长, 支持向量的选择困难, 学习算法的推导较深奥	在模式分类问题上能提供良好的泛化性能

1.3 神经网络的基本特征与功能

人工神经网络是基于对人脑组织结构、活动机制的初步认识提出的一种新型信息处理体系。通过模仿脑神经系统的组织结构以及某些活动机理，人工神经网络可呈现出人脑的许多特征，并具有人脑的一些基本功能。

1.3.1 神经网络的基本特征

下面将神经网络的特征归纳为结构特征和能力特征分别介绍。

(1) 结构特征——并行处理、分布式存储与容错性 人工神经网络是由大量简单处理元件相互连接构成的高度并行的非线性系统，具有大规模并行性处理特征。虽然每个处理单元的功能十分简单，但大量简单处理单元的并行活动使网络呈现出丰富的功能并具有较快的速度。结构上的并行性使神经网络的信息存储必然采用分布式方式，即信息不是存储在网络的某个局部，而是分布在所有的连接权中。一个神经网络可存储多种信息，其中每个神经元的连接权中存储的是多种信息的一部分。当需要获得已存储的知识时，神经网络在输入信息激励下采用“联想”的办法进行回忆，因而具有联想记忆功能。神经网络内在的并行性与分布性表现在其信息的存储与处理都是空间上分布、时间上并行的。这两个特点必然使神经网络在两个方面表现出良好的容错性：一方面，由于信息的分布式存储，当网络中部分神经元损坏时不会对系统的整体性能造成影响，这一点就像人脑中每天都有神经细胞正常死亡而不会影响大脑的功能一样；另一方面，当输入模糊、残缺或变形的信息时，神经网络能通过联想恢复完整的记忆，从而实现对不完整输入信息的正确识别，这一特点就像人可以对不规范的手写体进行正确识别一样。

(2) 能力特征——自学习、自组织与自适应性 自适应性是指一个系统能改变自身的性能以适应环境变化的能力，它是神经网络的一个重要特征。自适应性包含自学习与自组织两层含义。神经网络的自学习是指当外界环境发生变化时，经过一段时间的训练或感知，神经网络能通过自动调整网络结构参数，使得对于给定输入能产生期望的输出，训练是神经网络学习的途径，因此经常将学习与训练两个词混用。神经系统能在外部刺激下按一定规则调整神经元之间的突触连接，逐渐构建起神经网络，这一构建过程称为网络的自组织（或称重构）。神经网络的自组织能力与自适应性相关，自适应性是通过自组织实现的。

1.3.2 神经网络的基本功能

人工神经网络是借鉴于生物神经网络而发展起来的新型智能信息处理系统，由于其结构上“仿造”了人脑的生物神经系统，因而其功能上也具有了某种智能特点。下面对神经网络的基本功能进行简要介绍。

1.3.2.1 联想记忆

由于神经网络具有分布存储信息和并行处理信息的特点，因此它具有对外界刺激信息和输入模式进行联想记忆的能力。这种能力是通过神经元之间的协同结构以及信息处理的集体行为而实现的。神经网络是通过其突触权值和连接结构来表达信息的记忆，这种分布式存储使得神经网络能存储较多的复杂模式和恢复记忆的信息。神经网络通过预先存储信息和学习机制进行自适应训练，可以从不完整的信息和噪声干扰中恢复原始的完整信息，这一能力使其在图像复原、图像和语音处理、模式识别、分类等方面具有巨大的潜在应用价值。

联想记忆有两种基本形式：自联想记忆与异联想记忆，见图 1.1。

(1) 自联想记忆 如图 1.1(a) 所示，网络中预先存储（记忆）多种模式信息，当输入某个已存储模式的部分信息或带有噪声干扰的信息时，网络能通过动态联想过程回忆起该模式的全部信息。

(2) 异联想记忆 如图 1.1(b) 所示，网络中预先存储了多个模式对，每一对模式均由两部分组成，当输入某个模式对的一部分时，即使输入信息是残缺的或叠加了噪声的，网络也能回忆起与其对应的另一部分。

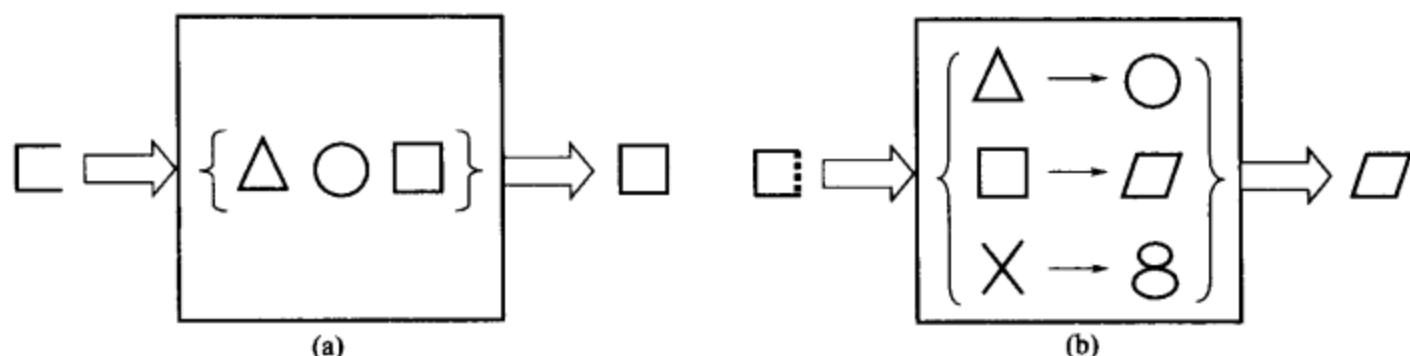


图 1.1 联想记忆

1.3.2.2 非线性映射

在客观世界中，许多系统的输入与输出之间存在复杂的非线性关系，对于这类系统，往往很难用传统的数理方法建立其数学模型。设计合理的神经网络通过对系统输入输出样本对进行自动学习，能够以任意精度逼近任意复杂的非线性映射。神经网络的这一优良性能使其可以作为多维非线性函数的通用数学模型。该模型的表达是非解析的，输入输出数据之间的映射规则由神经网络在学习阶段自动抽取并分布式存储在网络的所有连接中。具有非线性映射功能的神经网络应用十分广阔，几乎涉及所有领域。

1.3.2.3 分类与识别

神经网络对外界输入样本具有很强的识别与分类能力。对输入样本的分类实际上是在样本空间找出符合分类要求的分割区域，每个区域内的样本属于一类。传统分类方法只适合解决同类相聚、异类分离的识别与分类问题。但客观世界中许多事物（例如，不同的图像、声音、文字等）在样本空间中的区域分割曲面是十分复杂的，相近的样本可能属于不同的类，而远离的样本可能同属一类。神经网络可以很好地解决对非线性曲面的逼近，因此比传统的分类器具有更好的分类与识别能力。

1.3.2.4 优化计算

优化计算是指在已知的约束条件下，寻找一组参数组合，使由该组合确定的目标函数达到最小值。某些类型的神经网络可以把待求解问题的可变参数设计为网络的状态，将目标函数设计为网络的能量函数。神经网络经过动态演变过程达到稳定状态时对应的能量函数最小，从而其稳定状态就是问题的最优解。这种优化计算不需要对目标函数求导，其结果是网络自动给出的。

1.3.2.5 知识处理

知识是人们从客观世界的大量信息以及自身的实践中总结归纳出来的经验、规则和判断。神经网络获得知识的途径与人类似，也是从对象的输入输出信息中抽取规律而获得关于对象的知识，并将知识分布在网络的连接中予以存储。神经网络的知识抽取能力使其能够在

没有任何先验知识的情况下自动从输入数据中提取特征，发现规律，并通过自组织过程将自身构建成适合于表达所发现的规律。另一方面，人的先验知识可以大大提高神经网络的知识处理能力，两者相结合会使神经网络智能得到进一步提升。

1.4 神经网络的应用领域

神经网络的智能化特征与能力使其应用领域日益扩大，潜力日趋明显。许多用传统信息处理方法无法解决的问题采用神经网络后取得了良好的效果。限于篇幅下面简要介绍一下目前神经网络的几个主要应用领域，以使读者对神经网络能做什么有一个初步印象。

1.4.1 信息处理领域

神经网络作为一种新型智能信息处理系统，其应用贯穿信息的获取、传输、接收与加工利用等各个环节，这里仅列举几个方面的应用：

(1) 信号处理 神经网络广泛应用于自适应信号处理和非线性信号处理中。前者如信号的自适应滤波、时间序列预测、谱估计、噪声消除等；后者如非线性滤波、非线性预测、非线性编码、调制/解调等。在信号处理方面，神经网络有着许多成功应用的实例，第一个成功应用的实例就是电话线中回声的消除，其他还有雷达回波的多目标分类、运动目标的速度估计、多探测器的信息融合等。

(2) 模式识别 模式识别涉及模式的预处理变换和将一种模式映射为其他类型的操作，神经网络在这两个方面都有许多成功的应用。神经网络不仅可以处理静态模式，如固定图像、固定能谱等；还可以处理动态模式，如视频图像、连续语音等。众所周知的静态模式识别的成功实例有手写字的识别，动态模式识别的成功实例有语音信号的识别。目前电脑市场上随处可见的手写输入和语音输入系统进一步表明神经网络在模式识别方面的应用已经商品化。

(3) 数据压缩 在数据传送与存储时，数据压缩至关重要。神经网络可对待传送（或待存储）的数据提取模式特征，只将该特征传出（或存储）、接收后（或使用时），再将其恢复成原始模式。

1.4.2 自动化领域

20世纪80年代以来，神经网络和控制理论与控制技术相结合，发展为自动控制领域的一个前沿学科——神经网络控制。它是智能控制的一个重要分支，为解决复杂的非线性、不确定、不知系统的控制问题开辟了一条新的途径。神经网络用于控制领域，已取得以下主要进展：

(1) 系统辨识 在自动控制问题中，系统辨识的目的是为了建立被控对象的数学模型。多年来控制领域对于复杂的非线性对象的辨识，一直未能很好地解决。神经网络所具有的非线性特性和学习能力，使其在系统辨识方面有很大的潜力，为解决具有复杂的非线性、不确定性和不知对象的辨识问题开辟了一条有效途径。基于神经网络的系统辨识是以神经网络作为被辨识对象的模型，利用其非线性特性，可建立非线性系统的静态或动态模型。

(2) 神经控制器 由于控制器在实时控制系统中起着“大脑”的作用，神经网络具有自学习和自适应等智能特点，因而非常适合于作控制器。对于复杂非线性系统，神经控制器所达到的控制效果往往明显好于常规控制器。近年来，神经控制器在工业、航空以及机器人等

领域的控制系统应用中已取得许多可喜的成就。

(3) 智能检测 所谓智能检测一般包括干扰量的处理、传感器输入输出特性的非线性补偿、零点和量程的自动校正以及自动诊断等。这些智能检测功能可以通过传感元件和信号处理元件的功能集成来实现。随着智能化程度的提高,功能集成型已逐渐发展为功能创新型,如复合检测、特征提取及识别等,而这类信息处理问题正是神经网络的强项。在对综合指标的检测(例如对环境舒适度这类综合指标的检测)中,以神经网络作为智能检测中的信息处理元件便于对多个传感器的相关信息(如温度、湿度、风向和风速等)进行复合、集成、融合、联想等数据融合处理,从而实现单一传感器所不具备的功能。

1.4.3 工程领域

20世纪80年代以来,神经网络的理论研究成果已在众多的工程领域取得了丰硕的应用成果,下面的介绍仅助读者窥其一斑:

(1) 汽车工程 汽车在不同状态参数下运行时,能获得最佳动力性与经济性的挡位称为最佳挡位。利用神经网络的非线性映射能力,通过学习优秀驾驶员的换挡经验数据,可自动提取蕴含在其中的最佳换挡规律。神经网络在汽车刹车自动控制系统中也有成功的领用,该系统能在给定煞车距离、车速和最大减速度的情况下,以人体感受到最小冲击实现平稳刹车,而不受路面坡度和车重的影响。随着国内外对能源短缺和环境污染问题的日趋关切,燃油消耗率和排烟度愈来愈受到人们的关注。神经网络在载重车柴油机燃烧系统方案优化中的应用,有效地降低了油耗和排烟度,获得了良好的社会经济效益。

(2) 军事工程 神经网络同红外搜索与跟踪系统配合后可发现与跟踪飞行器。一个成功的例子是,利用神经网络检测空间卫星的动作状态是稳定、倾斜、旋转还是摇摆,正确率可达95%。利用声呐信号判断水下目标是潜艇还是礁石,是军事上常采用的办法。借助神经网络的语音分类与信号处理上的经验对声纳信号进行分析研究,对水下目标的识别率可达90%。密码学研究一直是军事领域中的重要研究课题,利用神经网络的联想记忆特点可设计出密钥分散保管方案;利用神经网络的分类能力可提高密钥的破解难度;利用神经网络还可设计出安全的保密开关,如语音开关、指纹开关等。

(3) 化学工程 20世纪80年代中期以来,神经网络在制药、生物化学、化学工程等领域内的研究与应用蓬勃开展,取得了不少成果。例如,在光谱分析方面,应用神经网络在红外光谱、紫外光谱、折射光谱和质谱与化合物的化学结构间建立某种确定的对应关系方面的成功应用实例比比皆是。此外,还有将神经网络用于判定化学反应的生成物,用于判定钾离子、钙离子、硝酸离子、氯离子等的浓度,用于研究生命体中某些化合物的含量与其生物活性的对应关系等大量应用实例。

(4) 水利工程 近年来,我国水利工程领域的科技人员已成功地将神经网络的方法用于水力发电过程辨识和控制、河川径流预测、河流水质分类、水资源规划、混凝土性能预估、拱坝优化设计、预应力混凝土桩基等结构损伤诊断、砂土液化预测、岩体可爆破性分级及爆破效应预测、岩土类型识别、地下工程围岩分类、大坝等工程结构安全监测、工程造价分析等许多实际问题中。

1.4.4 经济领域

(1) 在微观经济领域的应用 用人工神经网络构造的企业成本预测模型,可以模拟生产、管理各个环节的活动,跟踪价值链的构成,适应企业的成本变化,预测的可靠性强。利

用人工神经网络对销售额进行仿真实验，可以准确地预测未来的销售额。另外，神经网络在各类企业的信用风险、财务风险、金融风险的评级和评价方面也得到大量应用。与其他预测方法比较，它具有处理非线性问题的能力和自学习特点。

(2) 在宏观经济领域的应用 主要用于国民经济参数的测算、通货膨胀率和经济周期的预测，经济运行态势的预测预警。例如，对汇率和利率的测算，对GDP和各种总产值的预测等。使用人工神经网络对宏观经济变量进行测算和预测只需少量训练样本就可以确定网络的权重和阈值，精度较高，能够对宏观经济系统中的非线性关系进行描述，使建立的非线性模型与实际系统更加接近。

(3) 在证券市场中的应用 股票的收益性是投资者购买股票的主要依据，用BP网络可以准确预测盈利水平的未来走向。在投资组合的选择时遇到的证券投资组合模型是一个多目标的非线性规划问题，考虑到各种证券的收益性、风险性和投资期的搭配等多种因素的综合作用，模型的规模较大，用一般的分析工具找到最优投资组合的过程非常困难。人工神经网络的自学习、自组织和非线性动态特征，能够实现并行处理和快速运算，使它在选择投资组合时有独特的优势。

(4) 在金融领域的应用 在金融领域用BP网络构造的信用评价模型可以对贷款申请人的信用等级进行评价，对公司信用和财务状况作出综合评价，进行破产风险分析，对贷款产生的效益针对不同的利益主体进行综合评价。采用人工神经网络为金融和实物期权定价，能较好地克服现有定价方法缺乏相关信息、价格确定过程主观化等不足，使定价更客观准确，为投资决策提供科学的定价依据。

(5) 在社会经济发展评价和辅助决策中的应用 社会经济是多目标、多层次的大系统，其发展状况评价的分析工具必须适应大系统动态化的要求。人工神经网络能够逼近任何函数，这一特点使大系统的综合评价、模糊评价和动态评价有了科学依据。同时在评价中可以产生一系列的决策参数，使人工神经网络成为辅助决策的可靠工具。例如在产业竞争力评价、可持续发展评价、选址决策、运输方案决策、规划问题、区域发展战略研究等领域运用神经网络作出的评价结果和决策方案能真实地反映客观实际，具有科学性和可靠性。

1.4.5 医学领域

(1) 检测数据分析 许多医学检测设备的输出数据都是连续波形的形式，这些波的极性和幅值常常能够提供有意义的诊断依据。神经网络在这方面的应用非常普遍，一个成功的应用实例是用神经网络进行多道脑电棘波的检测。很多癫痫病人常规检测往往无效，但他们可得益于脑电棘波检测系统。脑电棘波的出现通常意味着脑功能的某些异常，棘波的极性和幅值经常提供了异常的部位和程度信息，因而神经网络脑电棘波检测系统可用来提供脑电棘波的实时检测和癫痫的预报。

(2) 生物活性研究 用神经网络对生物学检测数据进行分析，可提取致癌物的分子结构特征建立分子结构和致癌活性之间的定量关系，并对分子致癌活性进行预测。分子致癌性的神经网络预测具有生物学检测所不具备的优点，它不仅可对新化合物的致癌性和致突变性预先作出评价，从而避免盲目投入造成浪费，而且检测费用低，可作为致癌物大面积预筛的工具。

(3) 医学专家系统 专家系统在医疗诊断方面有许多应用。虽然专家系统的研究与应用取得了重大进展，但由于知识“爆炸”和冯·诺依曼计算机的“瓶颈”问题使其应用受到严重挑战。以非线性并行分布式处理为基础的神经网络为专家系统的研究开辟了新的途径，利

用其学习功能、联想记忆功能和分布式并行信息处理功能，来解决专家系统中的知识表示、获取和并行推理等问题取得了良好效果。

本章小结

本章讨论了人脑与计算机信息处理能力的差异，分析了两者在信息处理机制方面的特点，并阐述了人工神经网络的概念。通过对人工神经网络曲折的发展过程的叙述，展示了该领域的主要研究内容与理论成果。此外，简要说明了神经网络的基本特征与主要功能，并通过简要介绍神经网络的广泛应用使读者初步了解了神经网络在信息处理方面表现出来的巨大潜力。

本章要点是：

(1) 什么是人工神经网络 在对人脑神经网络的基本认识的基础上，用数理方法从信息处理的角度对人脑神经网络进行抽象，并建立某种简化模型，就称为人工神经网络。人工神经网络远不是人脑生物神经网络的真实写照，而只是对它的简化、抽象与模拟。因此，人工神经网络是一种旨在模仿人脑结构及其功能的信息处理系统。

(2) 神经网络的发展 可分为四个时期：启蒙时期开始于 1890 年 W. James 关于人脑结构与功能的研究，结束于 1969 年 Minsky 和 Papert 出版《感知器》(Perceptrons) 一书；低潮时期开始于 1969 年，结束于 1982 年 Hopfield 发表著名的文章《神经网络和物理系统》(Neural Network and Physical System)；复兴时期开始于 J. J. Hopfield 的突破性研究论文，结束于 1986 年 D. E. Rumelhart 和 J. L. McClelland 领导的研究小组编写出版的《并行分布式处理》(Parallel Distributed Processing) 一书。高潮时期以 1987 年首届国际人工神经网络学术会议为开端，迅速在全世界范围内掀起人工神经网络的研究应用热潮。

(3) 神经网络的基本特征 结构上的特征是处理单元的高度并行性与分布性，这种特征使神经网络在信息处理方面具有信息的分布存储与并行计算、存储与处理一体化的特点。而这些特点必然给神经网络带来较快的处理速度和较强的容错能力。能力方面的特征是神经网络的自学习、自组织与自适应性。自适应性是指一个系统能改变自身的性能以适应环境变化的能力，它包含自学习与自组织两层含义。自学习是指当外界环境发生变化时，经过一段时间的训练或感知，神经网络能通过自动调整网络结构参数，使得对于给定输入能产生期望的输出。自组织是指神经系统能在外部刺激下按一定规则调整神经元之间的突触连接，逐渐构建起神经网络。

(4) 神经网络的基本功能 神经网络的 5 种功能具有智能特点，重点是其前两种功能：①联想记忆功能，指神经网络能够通过预先存储信息和学习机制进行自适应训练，从不完整的信息和噪声干扰中恢复原始的完整信息；②非线性映射功能，指神经网络能够通过对系统输入输出样本对的学习自动提取蕴涵其中的映射规则，从而以任意精度拟合任意复杂的非线性函数。

思考与练习

- 1.1 根据自己的体会，列举人脑与电脑信息处理能力有哪些不同。
- 1.2 神经网络的功能特点是由什么决定的？
- 1.3 根据人工神经网络的特点，你认为它善于解决哪类问题？
- 1.4 神经网络研究在 20 世纪 70~80 年代处于低潮的主要原因是什么？
- 1.5 神经网络研究于 20 世纪 80 年代中期复兴的动力是什么？

2 神经网络基础知识

20世纪40年代第一台电子计算机的问世是人类改造大自然进程中的一个重要里程碑。电子计算机作为具有计算和存储能力的“电脑”，物化延伸了人脑的智力，这是探索构造具有脑智能的人工系统的一个重大进步。以电子计算机为基础的人工智能和专家系统，在信息的加工、处理中起到了“智能化”的作用。然而，人工智能的表示必须是形式化的，且处理的方式是计算机串行处理，而作为真正具有智能的人脑并不是以这种方式进行思维活动。

下面将说明，作为“智能”物质基础的大脑是如何构成和如何工作的，在构造新型智能信息处理系统时可以从中得到什么启示。

2.1 人工神经网络的生物学基础

神经生理学和神经解剖学的研究结果表明，神经元（neuron）是脑组织的基本单元，是神经系统结构与功能的单位。据估计，人类大脑大约包含有 1.4×10^{11} 个神经元，每个神经元与大约 $10^3 \sim 10^5$ 个其他神经元相连接，构成一个极为庞大而复杂的网络，即生物神经网络。生物神经网络中各神经元之间连接的强弱，按照外部的激励信号作自适应变化，而每个神经元又随着接收到的多个激励信号的综合结果呈现出兴奋与抑制状态。大脑的学习过程就是神经元之间连接强度随外部激励信息作自适应变化的过程，大脑处理信息的结果由各神经元状态的整体效果确定。显然，神经元是人脑信息处理系统的最小单元。

2.1.1 生物神经元的结构

人脑中神经元的形态不尽相同，功能也有差异，但从组成结构来看，各种神经元是有共性的。图2.1给出一个典型神经元的基本结构和与其他神经元发生连接的简化示意图。

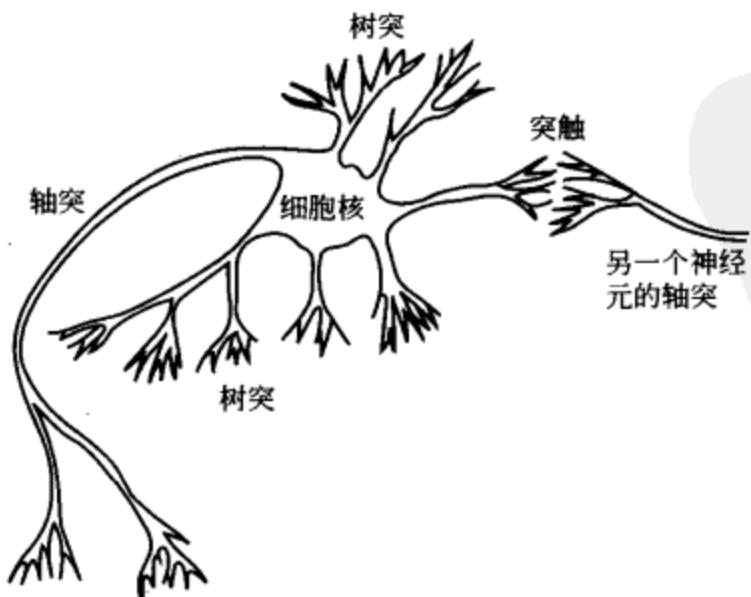


图2.1 生物神经元简化示意图

神经元在结构上由细胞体、树突、轴突和突触四部分组成：

(1) 细胞体 (cell body) 细胞体是神经元的主体，由细胞核、细胞质和细胞膜三部分构成。细胞核占据细胞体的很大一部分，进行着呼吸和新陈代谢等许多生化过程。细胞体的外部是细胞膜，将膜内外细胞液分开。由于细胞膜对细胞液中的不同离子具有不同的通透性，使得膜内外存在着离子浓度差，从而出现内负外正的静息电位。

(2) 树突 (dendrite) 从细胞体向外延伸出许多突起的神经纤维，其中大部分突起较短，其分支多群集在细胞体附近形成灌木丛状，这些突起称为树突。神经元靠树突接受来自其他神经元的输入信号，相当于细胞体的输入端。

(3) 轴突 (axon) 由细胞体伸出的最长的一条突起称为轴突。轴突比树突长而细，用来传出细胞体产生的输出电化学信号。轴突也称神经纤维，其分支倾向于在神经纤维终端处长出，这些细的分支称为轴突末梢或神经末梢。每一条神经末梢可以向四面八方传出信号，相当于细胞体的输出端。

(4) 突触 (synapse) 神经元之间通过一个神经元的轴突末梢和其他神经元的细胞体或树突进行通信连接，这种连接相当于神经元之间的输入输出接口，称为突触。突触包括突触前、突触间隙和突触后三个部分。突触前是第一个神经元的轴突末梢部分，突触后是指第二个神经元的树突或细胞体等受体表面。突触在轴突末梢与其他神经元的受体表面相接触的地方有 $15\sim50\text{nm}$ 的间隙，称为突触间隙，在电学上把两者断开，见图 2.2。每个神经元大约有 $10^3\sim10^5$ 个突触，多个神经元以突触连接即形成神经网络。

2.1.2 生物神经元的信息处理机理

在生物神经元中，突触为输入输出接口，树突和细胞体为输入端，接受突触点的输入信号；细胞体相当于一个微型处理器，对各树突和细胞体各部位收到的来自其他神经元的输入信号进行组合，并在一定条件下触发，产生一输出信号；输出信号沿轴突传至末梢，轴突末梢作为输出端通过突触将这一输出信号传向其他神经元的树突和细胞体。下面对生物神经元产生、传递、接收和处理信息的机理进行分析。

2.1.2.1 信息的产生

研究认为，神经元间信息的产生、传递和处理是一种电化学活动。由于细胞膜本身对不同离子具有不同的通透性，使膜内外细胞液中的离子存在浓度差，从而造成膜内外存在电位差。神经元在无神经信号输入时，其细胞膜内外因离子浓度差而造成的电位差为 -70mV (内负外正) 左右，称为静息电位，此时细胞膜的状态称为极化状态 (polarization)，神经元的状态为静息状态。当神经元受到外界的刺激时，如果膜电位从静息电位向正偏移，称之为去极化 (depolarization)，此时神经元的状态为兴奋状态；如果膜电位从静息电位向负偏移，称之为超级化 (hyperpolarization)，此时神经元的状态为抑制状态。神经元细胞膜的去极化和超级化程度反映了神经元的兴奋和抑制的强烈程度。在某一给定时刻，神经元总是处于静息、兴奋和抑制三种状态之一。“兴奋”或“抑制”是对神经元所处状态的描述，其实质为

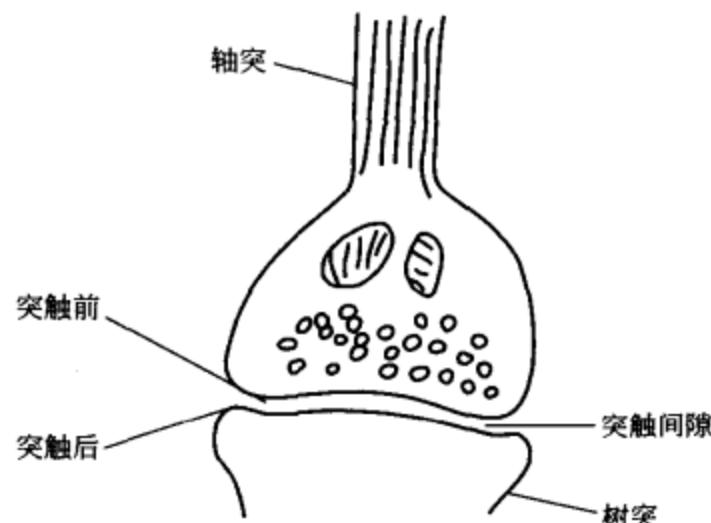


图 2.2 突触结构示意图

细胞膜电位的去极化或超极化。神经元中信息的产生与兴奋程度相关，在外界刺激下，当神经元的兴奋程度超过了某个限度，也就是细胞膜去极化程度超过了某个阈值电位时，神经元被激发而输出神经脉冲。每个神经脉冲产生的经过如下：当膜电位以静息膜电位为基准高出 15mV ，即超过阈值电位（ -55mV ）时，该神经细胞变成活性细胞，其膜电位自发地急速升高，在 1ms 内比静息膜电位上升 100mV 左右，此后膜电位又急速下降，回到静止时的值。这一过程称作细胞的兴奋过程，兴奋的结果产生一个宽度为 1ms ，振幅为 100mV 的电脉冲，又称神经冲动，如图 2.3 所示。

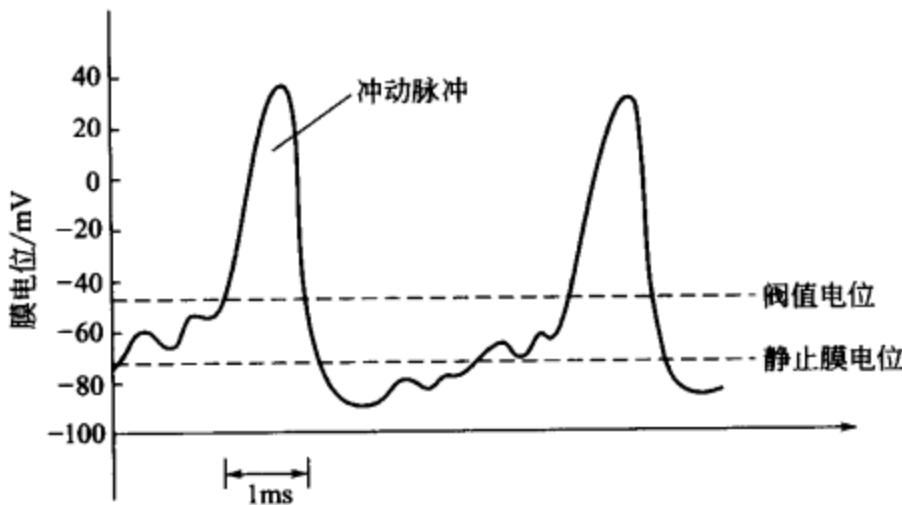


图 2.3 膜电位变化

值得注意的是，当细胞体产生一个电脉冲后，即使受到很强的刺激，也不会立刻产生兴奋，这是因为神经元发放电脉冲时，暂时性阈值急速升高，持续 1ms 后慢慢下降到 -55mV 这一正常状态，这段时间约为数毫秒，称为不应期。其中， 1ms 的持续时间称为绝对不应期，数毫秒的下降时间为相对不应期。不应期结束后，若细胞受到很强的刺激，则再次产生兴奋性电脉冲。由此可见，神经元产生的信息是具有电脉冲形式的神经冲动。各脉冲的宽度和幅度相同，而脉冲的间隔是随机变化的。某神经元的输入脉冲密度越大，其兴奋程度越高，在单位时间内产生的脉冲串的平均频率也越高，但由于脉冲的最小间隔不会小于不应期，因此在单位时间内产生的脉冲串的平均频率是有上限的，即神经元的输出具有饱和特性。

2.1.2.2 信息的传递与接收

神经元对信息的传递和接收都是通过突触进行的。突触间隙使突触前和突触后在电路上断开，神经脉冲信号是如何通过的呢？沿轴突传向其末端各个分支的脉冲信号，在轴突的末端触及突触前时，突触前的突触小泡能释放一种化学物质，称为递质。在前一个神经元发放脉冲并传到其轴突末端后，这种递质从突触前膜释放出，经突触间隙的液体扩散，在突触后膜与特殊受体相结合。受体的性质决定了递质的作用是兴奋的还是抑制的，并据此改变后膜的离子通透性，从而使突触后膜电位发生变化。根据突触后膜电位的变化，可将突触分为两种：兴奋性突触和抑制性突触。兴奋性突触的后膜电位随递质与受体结合数量的增加而向正电位方向增大，抑制性突触的后膜电位随递质与受体结合数量的增加向更负电位方向变化。从化学角度看，当兴奋性化学递质传送到突触后膜时，后膜对离子通透性的改变使流入细胞膜内的正离子增加，从而使突触后成分去极化，产生兴奋性突触后电位；当抑制性化学递质传送到突触后膜时，后膜对离子通透性的改变使流出细胞膜外的正离子增加，从而使突触后成分超极化，产生抑制性突触后电位。

当突触前膜释放的兴奋性递质使突触后膜的去极化电位超过了某个阈电位时，后一个神经元就有神经脉冲输出，从而把前一神经元的信息传递给了后一神经元（图 2.4）。这种传递的效率与突触的连接强度（或称耦合系数）有关，不同的突触连接可以将传递的信息增强或削弱。

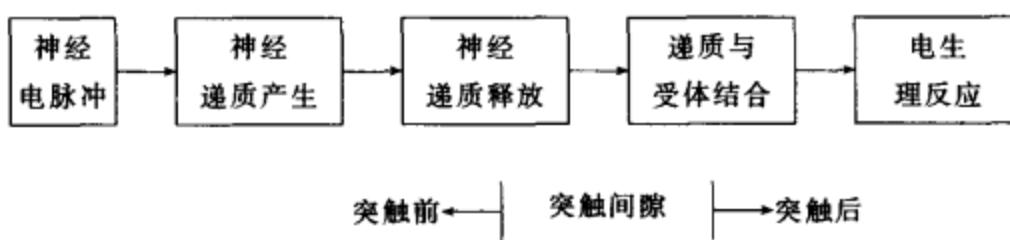


图 2.4 突触信息传递过程

从脉冲信号到达突触前膜，到突触后膜电位发生变化，有 0.2~1ms 的时间延迟，称为突触延迟（synaptic delay），这段延迟是化学递质分泌、向突触间隙扩散、到达突触后膜并在那里发生作用的时间总和。由此可见，突触对神经冲动的传递具有延时作用。

在人脑中，神经元间的突触联系大部分是在出生后由于给予刺激而成长起来的。外界刺激性质不同，能够改变神经元之间的突触联系，即突触后膜电位变化的方向与大小。从突触信息传递的角度看，表现为放大倍数和极性的变化。正是由于各神经元之间的突触连接强度和极性有所不同并可进行调整，因此人脑才具有学习和存储信息的功能。

2.1.2.3 信息的整合

神经元对信息的接收和传递都是通过突触来进行的。单个神经元可以与多达上千个其他神经元的轴突末梢形成突触连接，接受从各个轴突传来的脉冲输入。这些输入可到达神经元的不同部位，输入部位不同，对神经元影响的权重也不同。在同一时刻产生的刺激所引起的膜电位变化，大致等于各单独刺激引起的膜电位变化的代数和。这种加权求和称为空间整合。另外，各输入脉冲抵达神经元的时间先后也不一样。由一个脉冲引起的突触后膜电位很小，但在其持续时间内有另一脉冲相继到达时，总的突触后膜电位增大。这种现象称为时间整合。

一个神经元的输入信息在时间和空间上常呈现一种复杂多变的形式，神经元需要对它们进行积累和整合加工，从而决定输出的时机和强弱。正是神经元的这种时空整合作用，才使得亿万个神经元在神经系统中可以有条不紊、夜以继日地处理着各种复杂的信息，执行着生物中枢神经系统的各种信息处理功能。

2.1.2.4 生物神经网络

由多个生物神经元以确定方式和拓扑结构相互连接即形成生物神经网络，它是一种更为灵巧、复杂的生物信息处理系统。研究表明，每一个生物神经网络系统均是一个有层次的、多单元的动态信息处理系统，它们有其独特的运行方式和控制机制，以接受生物系统内外环境的输入信息，加以综合分析处理，然后调节控制机体对环境作出适当反应。生物神经网络的功能不是单个神经元信息处理功能的简单叠加。每个神经元都有许多突触与其他神经元连接，任何一个单独的突触连接都不能完全表现一项信息。只有当它们集合成为整体时才能对刺激的特殊性质给出明确的答复。由于神经元之间突触连接方式和连接强度的不同并且具有可塑性，神经网络在宏观上呈现出千变万化的复杂的信息处理能力。

人类社会的组成与生物神经网络的组成有异曲同工之妙。人类社会以人（脑）为基本单

位，每个人都与其周围的人形成或强或弱的、性质不同的连接关系（例如父子、母子、兄弟姐妹、同学、邻居、同事、上下级、师生等），从而形成各自的社会关系网。

人与人之间的联系强度不是固定不变的，而是随着各种激励信息作自适应变化，从而使社会整体上呈现出一定时期的社会风气、社会思潮和社会现象。

2.2 人工神经元模型

人工神经网络是在现代神经生物学研究基础上提出的模拟生物过程，反映人脑某些特性的一种计算结构。它不是人脑神经系统的真实描写，而只是它的某种抽象、简化和模拟。根据前面对生物神经网络的介绍可知，神经元及其突触是神经网络的基本器件。因此，模拟生物神经网络应首先模拟生物神经元。在人工神经网络中，神经元常被称为“处理单元”，有时从网络的观点出发常把它称为“节点”。人工神经元是对生物神经元的一种形式化描述，它对生物神经元的信息处理过程进行抽象，并用数学语言予以描述；对生物神经元的结构和功能进行模拟，并用模型图予以表达。

2.2.1 神经元的建模

目前人们提出的神经元模型已有很多，其中最早提出且影响最大的，是 1943 年心理学家 McCulloch 和数学家 W. Pitts 在分析总结神经元基本特性的基础上首先提出的 M-P 模型。该模型经过不断改进后，形成目前广泛应用的形式神经元模型。关于神经元的信息处理机制，该模型在简化的基础上提出以下 6 点假定进行描述：

- (1) 每个神经元都是一个多输入单输出的信息处理单元；
- (2) 神经元输入分兴奋性输入和抑制性输入两种类型；
- (3) 神经元具有空间整合特性和阈值特性；
- (4) 神经元输入与输出间有固定的时滞，主要取决于突触延搁；
- (5) 忽略时间整合作用和不应期；
- (6) 神经元本身是非时变的，即其突触时延和突触强度均为常数。

显然，上述假定是对生物神经元信息处理过程的简化和概括，它清晰地描述了生物神经元信息处理的特点，而且便于进行形式化表达。下面根据上述假定，对神经元进行形式化描述，即建立神经元的人工模型，包括图解表达和公式表达两种形式。

图解表达可用图 2.5 中的神经元模型示意图表示。图 2.5(a) 表明，正如生物神经元有许多激励输入一样，人工神经元也应该有许多的输入信号（图中每个输入的大小用确定数值 x_i 表示），它们同时输入神经元 j ，输出也同生物神经元一样仅有一个，用 o_j 表示神经元输出。图 2.5(b) 表明，生物神经元具有不同的突触性质和突触强度，其对输入的影响是使有些输入在神经元产生脉冲输出过程中所起的作用比另外一些输入更为重要。图中对神经元的每一个输入都有一个加权系数 w_{ij} ，称为权重值，其正负模拟了生物神经元中突触的兴奋和抑制，其大小则代表了突触的不同连接强度。图 2.5(c) 表示作为人工神经网络的基本处理单元，必须对全部输入信号进行整合，以确定各类输入的作用总效果，组合输入信号的“总和值”相当于生物神经元的膜电位。图 2.5(d) 表示神经元激活与否取决于某一阈值电平，即只有当其输入总和超过阈值 T 时，神经元才被激活而发放脉冲，否则神经元不会产生输出信号。输出与输入之间的对应关系可用图的某种函数 f 来表示，这种函数一般都是非线性的。

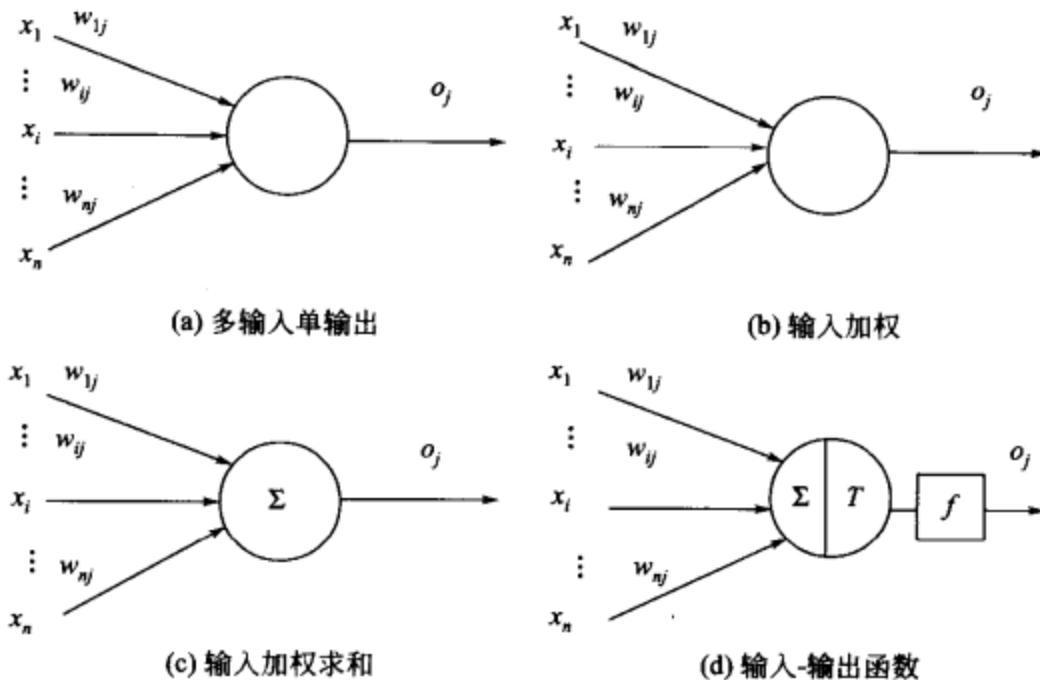


图 2.5 神经元模型示意图

2.2.2 神经元的数学模型

上述内容可用一个数学表达式进行抽象与概括。令 $x_i(t)$ 表示 t 时刻神经元 j 接收的来自神经元 i 的输入信息， $o_j(t)$ 表示 t 时刻神经元 j 的输出信息，则神经元 j 的状态可表达为：

$$o_j(t) = f \left\{ \left[\sum_{i=1}^n w_{ij} x_i(t - \tau_{ij}) \right] - T_j \right\} \quad (2.1)$$

式中 τ_{ij} —— 输入输出间的突触时延；

T_j —— 神经元 j 的阈值；

w_{ij} —— 神经元 i 到 j 的突触连接系数或称权重值；

$f(\cdot)$ —— 神经元转移函数。

为简单起见，将上式中的突触时延取为单位时间，则式(2.1) 可写为：

$$o_j(t+1) = f \left\{ \left[\sum_{i=1}^n w_{ij} x_i(t) \right] - T_j \right\} \quad (2.2)$$

上式描述的神经元数学模型全面表达了神经元模型的 6 点假定。其中输入 x_i 的下标 $i = 1, 2, \dots, n$ ，输出 o_j 的下标 j 体现了神经元模型假定（1）中的“多输入单输出”。权重值 w_{ij} 的正负体现了假定（2）中“突触的兴奋与抑制”。 T_j 代表假定（3）中神经元的“阈值”；“输入总和”常称为神经元在 t 时刻的净输入，用下式表示：

$$\text{net}'_j(t) = \sum_{i=1}^n w_{ij} x_i(t) \quad (2.3)$$

$\text{net}'_j(t)$ 体现了神经元 j 的空间整合特性，而未考虑时间整合，当 $\text{net}'_j(t) - T_j > 0$ 时，神经元才能被激活。 $o_j(t+1)$ 与 $x_i(t)$ 之间的单位时差意味着所有神经元具有相同的工作节律，对应于假定（4）中的“突触延搁”； w_{ij} 与时间无关体现了假定（6）中神经元的“非时变”。

为简便起见，在后面用到式(2.3) 时，常将其中的 (t) 省略。式(2.3) 还可表示为权重向量 \mathbf{W}_j 和输入向量 \mathbf{X} 的点积：

$$\text{net}'_j = \mathbf{W}_j^T \mathbf{X} \quad (2.4)$$

其中 \mathbf{W}_j 和 \mathbf{X} 均为列向量, 定义为:

$$\begin{aligned}\mathbf{W}_j &= (w_{1j}, w_{2j}, \dots, w_{nj})^T \\ \mathbf{X} &= (x_1, x_2, \dots, x_n)^T\end{aligned}$$

如果令 $x_0 = -1$, $w_{0j} = T_j$, 则有 $-T_j = x_0 w_{0j}$, 因此净输入与阈值之差可表达为:

$$\text{net}'_j - T_j = \text{net}_j = \sum_{i=0}^n w_{ij} x_i = \mathbf{W}_j^T \mathbf{X} \quad (2.5)$$

显然, 式(2.4) 中列向量 \mathbf{W}_j 和 \mathbf{X} 的第一个分量的下标均从 1 开始, 而式(2.5) 中则从 0 开始。采用式(2.5) 的约定后, 净输入改写为 net_j , 与原来的区别是包含了阈值。综合以上各式, 神经元模型可简化为:

$$o_j = f(\text{net}_j) = f(\mathbf{W}_j^T \mathbf{X}) \quad (2.6)$$

2.2.3 神经元的转移函数

神经元的各种不同数学模型的主要区别在于采用了不同的转移函数, 从而使神经元具有不同的信息处理特性。神经元的信息处理特性是决定人工神经网络整体性能的三大要素之一, 因此转移函数的研究具有重要意义。神经元的转移函数反映了神经元输出与其激活状态之间的关系, 最常用的转移函数有以下 4 种形式:

(1) 阈值型转移函数 图 2.6 给出两种阈值型转移函数, 图 2.6(a) 为单极性阈值型转移函数, 采用了由下式定义的单位阶跃函数:

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (2.7)$$

具有这一作用方式的神经元称为阈值型神经元, 这是神经元模型中最简单的一种, 经典的 M-P 模型就属于这一类。图 2.6(b) 为双极性阈值型转移函数, 采用了由下式定义的符号函数:

$$\text{sgn}(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases} \quad (2.8)$$

这是神经元模型中常用的一种, 许多处理离散信号的神经网络采用符号函数作为转移函数。阈值型函数中的自变量 x 代表 $\text{net}'_j - T_j$, 即当 $\text{net}'_j \geq T_j$ 时, 神经元为兴奋状态, 输出为 1; 当 $\text{net}'_j < T_j$ 时, 神经元为抑制状态, 输出为 0 或 -1。

(2) 非线性转移函数 非线性转移函数为实数域 R 到 $[0,1]$ 闭集的非减连续函数, 代表了状态连续型神经元模型。最常用的非线性转移函数是单极性 Sigmoid 函数曲线, 简称 S 型函数, 其特点是函数本身及其导数都是连续的, 因而在处理上十分方便。单极性 S 型函数定义为:

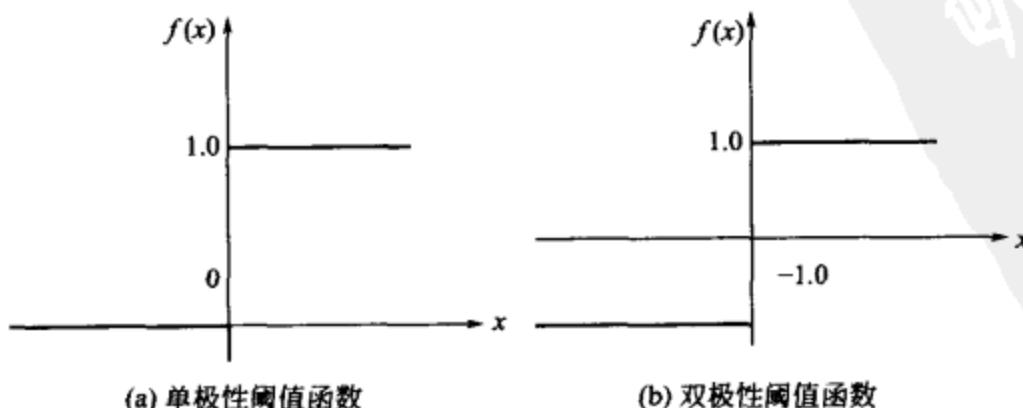


图 2.6 阈值型转移函数

$$f(x) = \frac{1}{1+e^{-x}} \quad (2.9)$$

有时也常采用双极性 S 型函数等形式：

$$f(x) = \frac{2}{1+e^{-x}} - 1 = \frac{1-e^{-x}}{1+e^{-x}} \quad (2.10)$$

S 型函数其曲线特点见图 2.7。

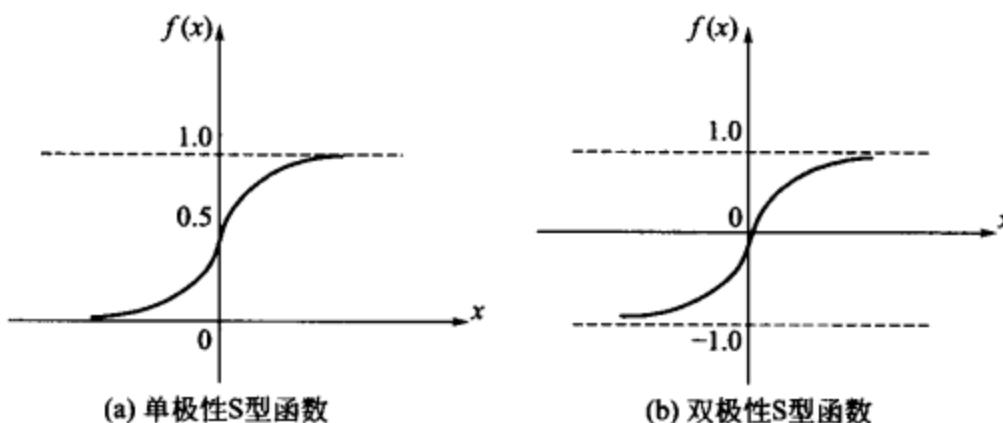


图 2.7 S 型转移函数

(3) 分段线性转移函数 该函数的特点是神经元的输入与输出在一定区间内满足线性关系。由于具有分段线性的特点，因而在实现上比较简单，这类函数也称为伪线性函数。单极性的分段线性转移函数表达式如下：

$$f(x) = \begin{cases} 0, & x \leq 0 \\ cx, & 0 < x \leq x_c \\ 1, & x_c < x \end{cases} \quad (2.11)$$

图 2.8 给出该函数曲线。

(4) 概率型转移函数 采用概率型转移函数的神经元模型其输入与输出之间的关系是不确定的，需用一个随机函数来描述其输出状态为 1 或为 0 的概率。设神经元输出为 1 的概率为：

$$P(1) = \frac{1}{1+e^{-x/T}} \quad (2.12)$$

式中， T 称为温度参数。由于采用该转移函数的神经元输出状态分布与热力学中的玻尔兹曼 (Boltzmann) 分布相类似，因此这种神经元模型也称为热力学模型。

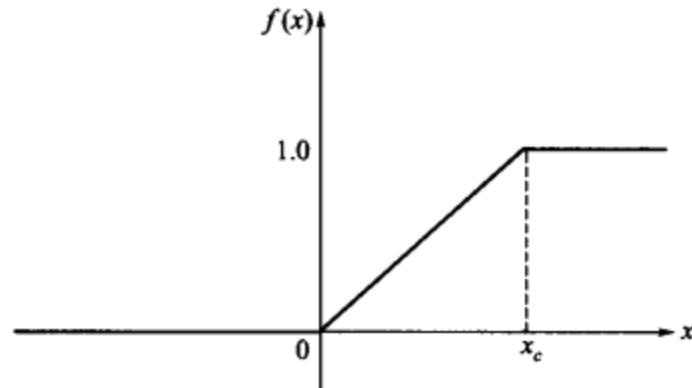


图 2.8 分段线性转移函数

2.3 人工神经网络模型

大量神经元组成庞大的神经网络，才能实现对复杂信息的处理与存储，并表现出各种优越的特性。神经网络的强大功能与其大规模并行互连、非线性处理以及互连结构的可塑性密切相关。因此必须按一定规则将神经元连接成神经网络，并使网络中各神经元的连接权按一定规则变化。生物神经网络由数以亿计的生物神经元连接而成，而人工神经网络限于物理实现的困难，且为了计算简便，是由相对少量的神经元按一定规律构成的网络。人工神经网络中的神经元常称为节点或处理单元，每个节点均具有相同的结构，其动作在时间和空间上均同步。

人工神经网络的模型很多，可以按照不同的方法进行分类。其中常见的两种分类方法是：按网络连接的拓扑结构分类和按网络内部的信息流向分类。

2.3.1 网络拓扑结构类型

神经元之间的连接方式不同，网络的拓扑结构也不同。根据神经元之间连接方式，可将神经网络结构分为两大类：

2.3.1.1 层次型结构

具有层次型结构的神经网络将神经元按功能分成若干层，如输入层、中间层（也称为隐层）和输出层，各层顺序相连，如图 2.9 所示。输入层各神经元负责接收来自外界的输入信息，并传递给中间各隐层神经元；隐层是神经网络的内部信息处理层，负责信息变换，根据信息变换能力的需要，隐层可设计为一层或多层；最后一个隐层传递到输出层各神经元的信息经进一步处理后即完成一次信息处理，由输出层向外界（如执行机构或显示设备）输出信息处理结果。层次型网络结构有 3 种典型的结合方式：

(1) 单纯型层次网络结构 在图 2.9 所示的层次型网络中，神经元分层排列，各层神经元接收前一层输入并输出到下一层，层内神经元自身以及神经元之间不存在连接通路。

(2) 输出层到输入层有连接的层次型网络结构 图 2.10 所示为输出层到输入层有连接路径的层次型网络结构。其中输入层神经元既可接收输入，也具有信息处理功能。

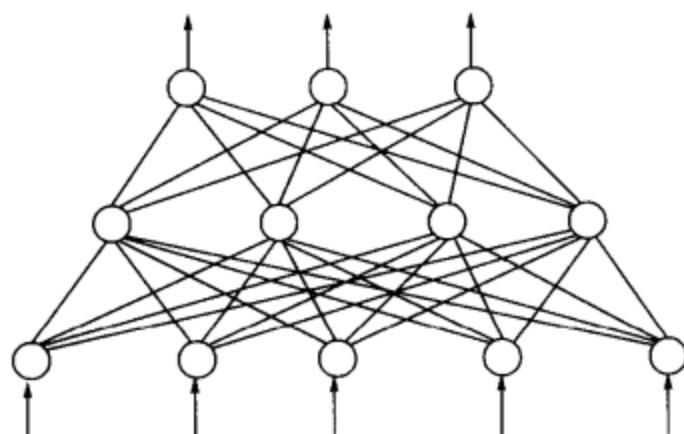


图 2.9 层次型网络结构示意图

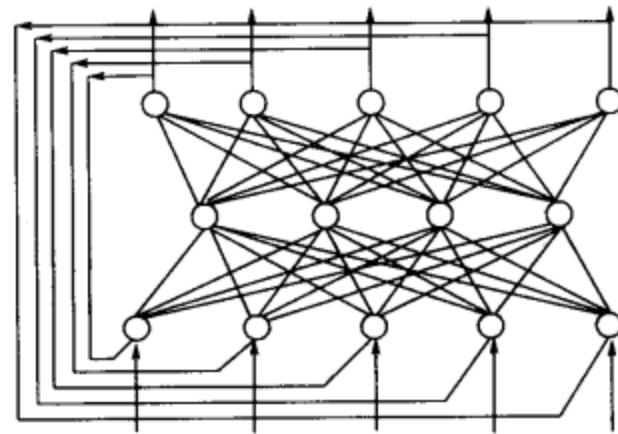


图 2.10 输出层到输入层有连接的
层次型网络结构示意图

(3) 层内有互连的层次型网络结构 图 2.11 所示为同一层内神经元有互连的层次型网络结构，这种结构的特点是在同一层内引入神经元间的侧向作用，使得能同时激活的神经元个数可控，以实现各层神经元的自组织。

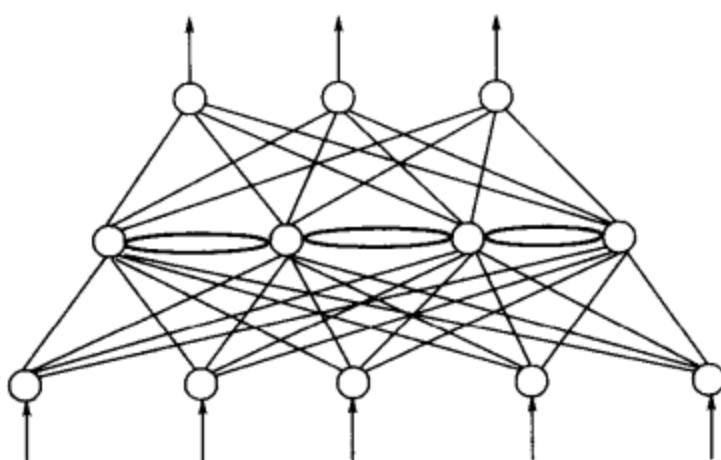


图 2.11 层内有互连的层次型网络结构示意图

2.3.1.2 互连型结构

对于互连型网络结构，网络中任意两个节点之间都可能存在连接路径，因此可以根据网络中节点的互连程度将互连型网络结构细分为三种情况：

(1) 全互连型 网络中的每个节点均与所有其他节点连接，如图 2.12 所示。

(2) 局部互连型 网络中的每个节点只与其邻近的节点有连接，如图 2.13 所示。

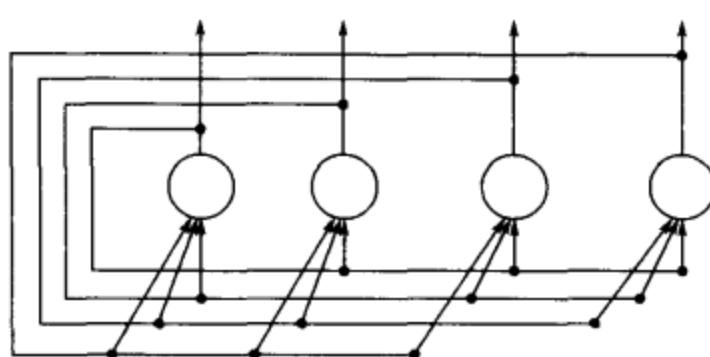


图 2.12 全互连型网络结构示意图

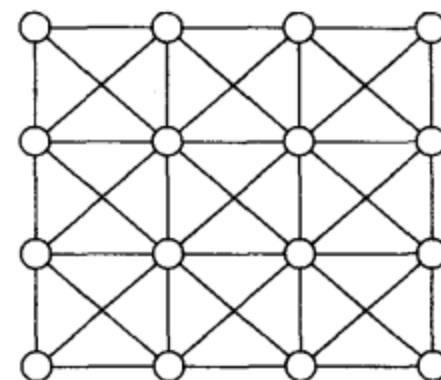


图 2.13 局部互连型网络结构示意图

(3) 稀疏连接型 网络中的节点只与少数相距较远的节点相连。

2.3.2 网络信息流向类型

根据神经网络内部信息传递方向来分，有以下两种类型：

2.3.2.1 前馈型网络

单纯前馈型网络的结构特点与图 2.9 中所示的分层网络完全相同，前馈是因网络信息处理的方向是从输入层到各隐层再到输出层逐层进行而得名。从信息处理能力看，网络中的节点可分为两种：一种是输入节点，只负责从外界引入信息后向前传递给第一隐层；另一种是具有处理能力的节点，包括各隐层和输出层节点。前馈网络中除输出层外，任一层的输出是下一层的输入，信息的处理具有逐层传递进行的方向性，一般不存在反馈环路。因此这类网络很容易串联起来建立多层前馈网络。

多层前馈网络可用一个有向无环路的图表示。其中输入层常记为网络的第一层，第一个隐层记为网络的第二层，其余类推。所以，当提到具有单层计算神经元的网络时，指的应是一个两层前馈网络（输入层和输出层）；当提到具有单隐层的网络时，指的应是一个三层前馈网络（输入层、隐层和输出层）。

2.3.2.2 反馈型网络

单纯反馈型网络的结构特点与图 2.12 中的网络结构完全相同，称为反馈网络是指其信息流向的特点。在反馈网络中所有节点都具有信息处理功能，而且每个节点既可以从外界接收输入，同时又可以向外界输出。单纯全互连结构网络是一种典型的反馈型网络，可以用图 2.14 所示的完全的无向图表示。

上面介绍的分类方法、结构形式和信息流向只是对目前常见的网络结构的概括和抽象。实际应用的神经网络可能同时兼有其中一种或几种形式。例如，从连接形式看，层次网络中可能出现局部的互连；从信息流向看，前馈网络中可能出现局部反馈。综合来看，图 2.9~图 2.13 中的网络模型可分别称为：前馈层次型、输入输出有反馈的前馈层次型、前馈层内互连型、反馈全互连型和反馈局部互连型。

神经网络的拓扑结构是决定神经网络特性的第二大要素，其特点可归纳为分布式存储记忆与分布式信息处理、高度互连性、高度并行性和结构可塑性。

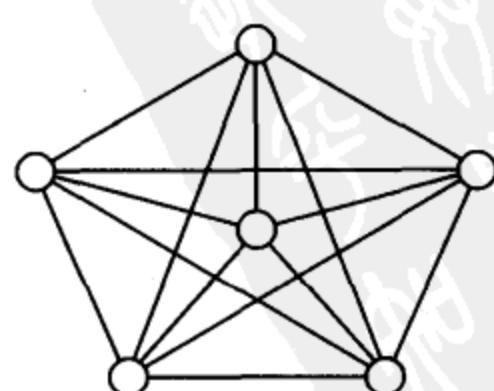


图 2.14 反馈型网络结构示意图

2.4 神经网络学习

人类具有学习能力。从行为主义的观点看，人的知识和智慧是在不断的学习与实践中逐渐形成和发展起来的。关于学习，可定义为：“根据与环境的相互作用而发生的行为改变，其结果导致对外界刺激产生反应的新模式的建立。”

学习过程离不开训练，学习过程就是一种经过训练而使个体在行为上产生较为持久改变的过程。例如，游泳等体育技能的学习需要反复的训练才能提高，数学等理论知识的掌握需要通过大量的习题进行练习。一般来说，学习效果随着训练量的增加而提高，这就是通过学习获得的进步。

关于学习的神经机制，涉及神经元如何分布、处理和存储信息。这样的问题单用行为研究是不能回答的，必须把研究深入到细胞和分子水平。正如两位心理学家 D. Hebb 和 J. Konorski 提出的，学习和记忆一定包含有神经回路的变化。因此，从生理学角度看，学习涉及的记忆与思维等心理功能，均归因于神经细胞组群的活动。在大脑中，要建立功能性的神经元连接，突触的形成是关键。神经元之间的突触联系，其基本部分是先天就有的，从而构成个体在某一方面的学习优势或天赋。但其他部分则是由于在后天的学习过程中频繁地给予刺激而成长起来的。突触的形成、稳定与修饰均与刺激有关，随着外界给予的刺激性质不同，能形成和改变神经元间的突触联系。

正如人脑的智能是不同突触分布的宏观表现，人脑的学习能力即形成和改变突触联系的能力，人工神经网络的功能特性和智能体现由其连接的拓扑结构和突触连接强度（即连接权值）决定。神经网络的全体连接权值可用一个矩阵 W 表示，其整体反映了神经网络对于所解决问题的知识存储。神经网络能够通过对样本的学习训练，不断改变网络的连接权值以及拓扑结构，以使网络的输出不断地接近期望的输出。这一过程称为神经网络的学习或训练，其本质是可变权值的动态调整。神经网络的学习方式是决定神经网络信息处理性能的第三大要素，因此有关学习的研究在神经网络研究中具有重要地位。改变权值的规则称为学习规则或学习算法（亦称训练规则或训练算法）。在单个处理单元层次，无论采用哪种学习规则进行调整，其算法都十分简单，但当大量处理单元集体进行权值调整时，网络就呈现出“智能”特性，其中有意义的信息就以分布的形式存储在调节后的权值矩阵中。

神经网络的学习算法很多，根据一种广泛采用的分类方法，可将神经网络的学习算法归纳为 3 类：一类是有导师学习；一类为无导师学习；还有一类是灌输式学习。

有导师学习也称为有监督学习，这种学习模式采用的是纠错规则。在学习训练过程中需要不断给网络成对提供一个输入模式和一个期望网络正确输出的模式，称为“教师信号”。将神经网络的实际输出同期望输出进行比较，当网络的输出与期望的教师信号不符时，根据差错的方向和大小按一定的规则调整权值，以使下一步网络的输出更接近期望结果。对于有导师学习，网络在能执行工作任务之前必须先经过学习，当网络对于各种给定的输入均能产生所期望的输出时，即认为网络已经在导师的训练下“学会”了训练数据集中包含的知识和规则，可以用来进行工作了。

无导师学习也称为无监督学习，学习过程中，需要不断地给网络提供动态输入信息，网络能根据特有的内部结构和学习规则，在输入信息流中发现任何可能存在的模式和规律，同时能根据网络的功能和输入信息调整权值，这个过程称为网络的自组织，其结果是使网络能

对属于同一类的模式进行自动分类。在这种学习模式中，网络的权值调整不取决于外来教师信号的影响，可以认为网络的学习评价标准隐含于网络的内部。

在有导师学习中，提供给神经网络学习的外部指导信息越多，神经网络学会并掌握的知识越多，解决问题的能力也就越强。但是，有时神经网络所解决的问题的先验信息很少，甚至没有，这种情况下无导师学习就显得更有实际意义。

灌输式学习是指先将网络设计成能记忆特别的例子，以后当给定有关该例子的输入信息时，例子便被回忆起来。灌输式学习中网络的权值一旦设计好了就不再变动，因此其学习是一次性的，而不是一个训练过程。

网络的运行一般分为训练和工作两个阶段。训练阶段的目的是为了从训练数据中提取隐含的知识和规律，并存储于网络中供工作阶段使用。

可以认为，一个神经元是一个自适应单元，其权值可以根据它所接收的输入信号、它的输出信号以及对应的监督信号进行调整。日本著名神经网络学者 Amari 于 1990 年提出一种神经网络权值调整的通用学习规则，该规则的图解表示见图 2.15。图中的神经元 j 是神经网络中的某个节点，其输入用向量 X 表示，该输入可以来自网络外部，也可以来自其他神经元的输出。第 i 个输入与神经元 j 的连接权值用 w_{ij} 表示，连接到神经元 j 的全部权值构成了权向量 W_j 。应当注意的是，该神经元的阈值 $T_j = w_{0j}$ ，对应的输入分量 x_0 恒为 -1。图中， $r = r(W_j, X, d_j)$ 代表学习信号，该信号通常是 W_j 和 X 的函数，而在有导师学习时，它也是教师信号 d_j 的函数。通用学习规则可表达为：权向量 W_j 在 t 时刻的调整量 $\Delta W_j(t)$ 与 t 时刻的输入向量 $X(t)$ 和学习信号 r 的乘积成正比。用数学式表示为：

$$\Delta W_j = \eta r [W_j(t), X(t), d_j(t)] X(t) \quad (2.13)$$

式中， η 为正数，称为学习常数，其值决定了学习速率。基于离散时间调整时，下一时刻的权向量应为：

$$W_j(t+1) = W_j(t) + \eta r [W_j(t), X(t), d_j(t)] X(t) \quad (2.14)$$

不同的学习规则对 $r(W_j, X, d_j)$ 有不同的定义，从而形成各种各样的神经网络。下面对常用学习算法作一简要介绍，其具体应用将在后续各章中展开。

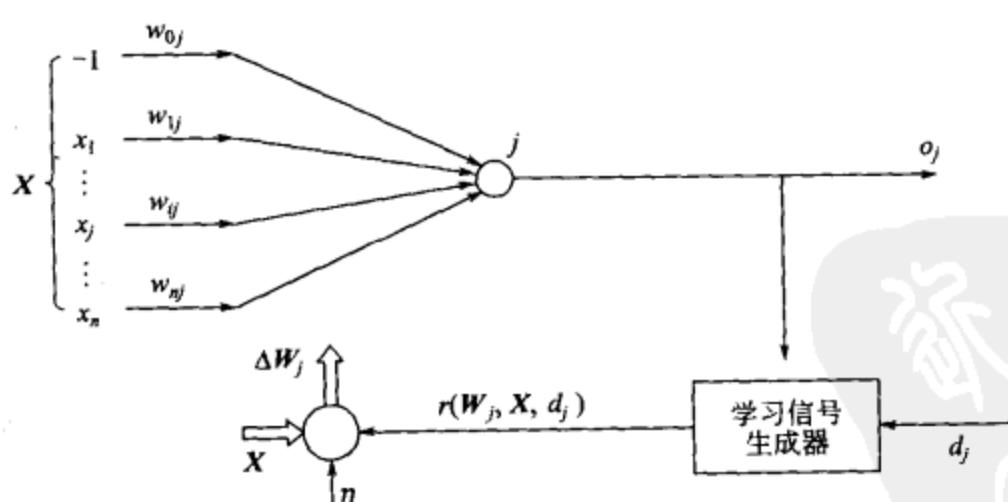


图 2.15 权值调整的一般情况

2.4.1 Hebb 学习规则

1949 年，心理学家 D. O. Hebb 最早提出了关于神经网络学习机理的“突触修正”的假设。该假设指出，当神经元的突触前膜电位与后膜电位同时为正时，突触传导增强，当前膜电位与后膜电位正负相反时，突触传导减弱。也就是说，当神经元 i 与神经元 j 同时处于兴

奋状态时，两者之间的连接强度应增强。根据该假设定义的权值调整方法，称为 Hebb 学习规则。

在 Hebb 学习规则中，学习信号简单地等于神经元的输出：

$$r = f(\mathbf{W}_j^T \mathbf{X}) \quad (2.15)$$

权向量的调整公式为：

$$\Delta \mathbf{W}_j = \eta f(\mathbf{W}_j^T \mathbf{X}) \mathbf{X} \quad (2.16a)$$

权向量中，每个分量的调整由下式确定：

$$\begin{aligned} \Delta w_{ij} &= \eta f(\mathbf{W}_j^T \mathbf{X}) x_i \\ &= \eta o_j x_i \quad i = 0, 1, \dots, n \end{aligned} \quad (2.16b)$$

上式表明，权值调整量与输入输出的乘积成正比。显然，经常出现的输入模式将对权向量有最大的影响。在这种情况下，Hebb 学习规则需预先设置权饱和值，以防止输入和输出正负始终一致时出现权值无约束增长。

此外，要求权值初始化，即在学习开始前 ($t = 0$)，先对 $\mathbf{W}_j(0)$ 赋予零附近的小随机数。

Hebb 学习规则代表一种纯前馈、无导师学习。该规则至今仍在各种神经网络模型中起着重要作用。

下面用一个简单的例子说明 Hebb 学习规则的应用。

例 2.1 设有 4 输入单输出神经元网络，其阈值 $T=0$ ，学习率 $\eta=1$ ，3 个输入样本向量和初始权向量分别为 $\mathbf{X}^1=(1, -2, 1.5, 0)^T$, $\mathbf{X}^2=(1, -0.5, -2, -1.5)^T$, $\mathbf{X}^3=(0, 1, -1, 1.5)^T$, $\mathbf{W}(0)=(1, -1, 0, 0.5)^T$ 。

解 首先设转移函数为符号函数 $f(\text{net})=\text{sgn}(\text{net})$ ，权值调整步骤为：

(1) 输入第一个样本 \mathbf{X}^1 ，计算净输入 net^1 ，并调整权向量 $\mathbf{W}(1)$

$$\text{net}^1 = \mathbf{W}(0)^T \mathbf{X}^1 = (1, -1, 0, 0.5)(1, -2, 1.5, 0)^T = 3$$

$$\begin{aligned} \mathbf{W}(1) &= \mathbf{W}(0) + \eta \text{sgn}(\text{net}^1) \mathbf{X}^1 = (1, -1, 0, 0.5)^T + (1, -2, 1.5, 0)^T \\ &= (2, -3, 1.5, 0.5)^T \end{aligned}$$

(2) 输入第二个样本 \mathbf{X}^2 ，计算净输入 net^2 ，并调整权向量 $\mathbf{W}(2)$

$$\text{net}^2 = \mathbf{W}(1)^T \mathbf{X}^2 = (2, -3, 1.5, 0.5)(1, -0.5, -2, -1.5)^T = -0.25$$

$$\begin{aligned} \mathbf{W}(2) &= \mathbf{W}(1) + \eta \text{sgn}(\text{net}^2) \mathbf{X}^2 = (2, -3, 1.5, 0.5)^T - (1, -0.5, -2, -1.5)^T \\ &= (1, -2.5, 3.5, 2)^T \end{aligned}$$

(3) 输入第三个样本 \mathbf{X}^3 ，计算净输入 net^3 ，并调整权向量 $\mathbf{W}(3)$

$$\text{net}^3 = \mathbf{W}(2)^T \mathbf{X}^3 = (1, -2.5, 3.5, 2)(0, 1, -1, 1.5)^T = -3$$

$$\begin{aligned} \mathbf{W}(3) &= \mathbf{W}(2) + \eta \text{sgn}(\text{net}^3) \mathbf{X}^3 = (1, -2.5, 3.5, 2)^T - (0, 1, -1, 1.5)^T \\ &= (1, -3.5, 4.5, 0.5)^T \end{aligned}$$

可见，当转移函数为符号函数且 $\eta=1$ 时，Hebb 学习规则的权值调整将简化为权向量加（或减）输入向量。

下面设转移函数为双极性连续函数 $f(\text{net}) = \frac{1-e^{-\text{net}}}{1+e^{-\text{net}}}$ ，权值调整步骤同上。

$$(1) \quad \text{net}^1 = \mathbf{W}(0)^T \mathbf{X}^1 = 3$$

$$o^1 = f(\text{net}^1) = \frac{1-e^{-\text{net}^1}}{1+e^{-\text{net}^1}} = 0.905$$

$$(2) \quad \mathbf{W}(1) = \mathbf{W}(0) + \eta f(\text{net}^1) \mathbf{X}^1 = (1.905, -2.81, 1.357, 0.5)^T$$

$$\text{net}^2 = \mathbf{W}(1)^T \mathbf{X}^2 = -0.154$$

$$o^2 = f(\text{net}^2) = \frac{1 - e^{-\text{net}}}{1 + e^{-\text{net}}} = -0.077$$

$$(3) \quad \mathbf{W}(2) = \mathbf{W}(1) + \eta f(\text{net}^2) \mathbf{X}^2 = (1.828, -2.772, 1.512, 0.616)^T$$

$$\text{net}^3 = \mathbf{W}(2)^T \mathbf{X}^3 = -3.36$$

$$o^3 = f(\text{net}^3) = \frac{1 - e^{-\text{net}}}{1 + e^{-\text{net}}} = -0.932$$

$$\mathbf{W}(3) = \mathbf{W}(2) + \eta f(\text{net}^3) \mathbf{X}^3 = (1.828, -3.70, 2.44, -0.785)^T$$

比较两种权值调整结果可以看出，两种转移函数下的权值调整方向是一致的，但采用连续转移函数时，权值调整力度减弱。

2.4.2 Perceptron 学习规则

1958年，美国学者 Frank Rosenblatt 首次定义了一个具有单层计算单元的神经网络结构，称为 Perceptron（感知器）。感知器的学习规则规定，学习信号等于神经元期望输出（教师信号）与实际输出之差：

$$r = d_j - o_j \quad (2.17)$$

式中， d_j 为期望的输出， $o_j = f(\mathbf{W}_j^T \mathbf{X})$ 。感知器采用了符号函数作为转移函数，其表达为：

$$f(\mathbf{W}_j^T \mathbf{X}) = \text{sgn}(\mathbf{W}_j^T \mathbf{X}) = \begin{cases} 1, & \mathbf{W}_j^T \mathbf{X} \geq 0 \\ -1, & \mathbf{W}_j^T \mathbf{X} < 0 \end{cases} \quad (2.18)$$

因此，权值调整公式应为：

$$\Delta \mathbf{W}_j = \eta [d_j - \text{sgn}(\mathbf{W}_j^T \mathbf{X})] \mathbf{X} \quad (2.19a)$$

$$\Delta w_{ij} = \eta [d_j - \text{sgn}(\mathbf{W}_j^T \mathbf{X})] x_i \quad i = 0, 1, \dots, n \quad (2.19b)$$

式中，当实际输出与期望值相同时，权值不需要调整；在有误差存在情况下，由于 d_j 和 $\text{sgn}(\mathbf{W}_j^T \mathbf{X}) \in \{-1, 1\}$ ，权值调整公式可简化为：

$$\Delta \mathbf{W}_j = \pm 2\eta \mathbf{X} \quad (2.19c)$$

感知器学习规则只适用于二进制神经元，初始权值可取任意值。

感知器学习规则代表一种有导师学习。由于感知器理论是研究其他神经网络的基础，该规则对于神经网络的有导师学习具有极为重要的意义。

2.4.3 δ 学习规则

1986年，认知心理学家 McClelland 和 Rumelhart 在神经网络训练中引入了 δ (Delta) 规则，该规则亦可称为连续感知器学习规则，与上述离散感知器学习规则并行。 δ 规则的学习信号规定为：

$$\begin{aligned} r &= [d_j - f(\mathbf{W}_j^T \mathbf{X})] f'(\mathbf{W}_j^T \mathbf{X}) \\ &= (d_j - o_j) f'(\text{net}_j) \end{aligned} \quad (2.20)$$

上式定义的学习信号称为 δ 。式中， $f'(\mathbf{W}_j^T \mathbf{X})$ 是转移函数 $f(\text{net}_j)$ 的导数。显然， δ 规则要求转移函数可导，因此只适用于有导师学习中定义的连续转移函数，如 Sigmoid 函数。

事实上， δ 规则很容易由输出值与期望值的最小平方误差条件推导出来。定义神经元输

出与期望输出之间的平方误差为：

$$\begin{aligned} E &= \frac{1}{2}(d_j - o_j)^2 \\ &= \frac{1}{2}[d_j - f(\mathbf{W}_j^T \mathbf{X})]^2 \end{aligned} \quad (2.21)$$

其中，误差 E 是权向量 \mathbf{W}_j 的函数。欲使误差 E 最小， \mathbf{W}_j 应与误差的负梯度成正比，即：

$$\Delta \mathbf{W}_j = -\eta \nabla E \quad (2.22)$$

式中，比例系数 η 是一个正常数。由式(2.21)，误差梯度为：

$$\nabla E = -(d_j - o_j) f'(\mathbf{W}_j^T \mathbf{X}) \mathbf{X} \quad (2.23)$$

将此结果代入式(2.22)，可得权值调整计算式：

$$\Delta \mathbf{W}_j = \eta(d_j - o_j) f'(\text{net}_j) \mathbf{X} \quad (2.24a)$$

可以看出，上式中 η 与 \mathbf{X} 之间的部分正是式(2.20) 中定义的学习信号 δ 。 $\Delta \mathbf{W}_j$ 中每个分量的调整由下式计算：

$$\Delta w_{ij} = \eta(d_j - o_j) f'(\text{net}_j) x_i \quad i=0,1,\dots,n \quad (2.24b)$$

δ 学习规则可推广到多层前馈网络中，权值可初始化为任意值。

下面举例说明 δ 学习规则的应用。

例 2.2 设有 3 输入单输出神经元网络，将阈值含于权向量内，故有 $w_0 = T$ ， $x_0 = -1$ ，学习率 $\eta = 0.1$ ，3 个输入向量和初始权向量分别为 $\mathbf{X}^1 = (-1, 1, -2, 0)^T$ ， $\mathbf{X}^2 = (-1, 0, 1.5, -0.5)^T$ ， $\mathbf{X}^3 = (-1, -1, 1, 0.5)^T$ ， $d^1 = -1$ ， $d^2 = -1$ ， $d^3 = 1$ ， $\mathbf{W}(0) = (0.5, 1, -1, 0)^T$ 。

解 设转移函数为双极性连续函数 $f(\text{net}) = \frac{1-e^{-\text{net}}}{1+e^{-\text{net}}}$ ，权值调整步骤为：

(1) 输入样本 \mathbf{X}^1 ，计算净输入 net^1 及权向量 $\mathbf{W}(1)$

$$\text{net}^1 = \mathbf{W}(0)^T \mathbf{X}^1 = 2.5$$

$$o^1 = f(\text{net}^1) = \frac{1-e^{-\text{net}^1}}{1+e^{-\text{net}^1}} = 0.848$$

$$f'(\text{net}^1) = \frac{1}{2}[1-(o^1)^2] = 0.14$$

$$\begin{aligned} \mathbf{W}(1) &= \mathbf{W}(0) + \eta(d^1 - o^1) f'(\text{net}^1) \mathbf{X}^1 \\ &= (0.526, 0.974, -0.948, 0)^T \end{aligned}$$

(2) 输入样本 \mathbf{X}^2 ，计算净输入 net^2 及权向量 $\mathbf{W}(2)$

$$\text{net}^2 = \mathbf{W}(1)^T \mathbf{X}^2 = -1.948$$

$$o^2 = f(\text{net}^2) = \frac{1-e^{-\text{net}^2}}{1+e^{-\text{net}^2}} = -0.75$$

$$f'(\text{net}^2) = \frac{1}{2}[1-(o^2)^2] = 0.218$$

$$\begin{aligned} \mathbf{W}(2) &= \mathbf{W}(1) + \eta(d^2 - o^2) f'(\text{net}^2) \mathbf{X}^2 \\ &= (0.531, 0.974, -0.956, 0.002)^T \end{aligned}$$

(3) 输入样本 \mathbf{X}^3 ，计算净输入 net^3 及权向量 $\mathbf{W}(3)$

$$\text{net}^3 = \mathbf{W}(2)^T \mathbf{X}^3 = -2.416$$

$$o^3 = f(\text{net}^3) = \frac{1-e^{-\text{net}^3}}{1+e^{-\text{net}^3}} = -0.842$$

$$\begin{aligned} f'(\text{net}^3) &= \frac{1}{2}[1 - (o^3)^2] = 0.145 \\ \mathbf{W}(3) &= \mathbf{W}(2) + \eta(d^3 - o^3)f'(\text{net}^3)\mathbf{X}^3 \\ &= (0.505, 0.947, -0.929, 0.016)^T \end{aligned}$$

2.4.4 LMS 学习规则

1962 年, Bernard Widrow 和 Marcian Hoff 提出了 Widrow-Hoff 学习规则。因为它能使神经元实际输出与期望输出之间的平方差最小, 所以又称为最小均方规则 (LMS)。LMS 学习规则的学习信号为:

$$r = d_j - \mathbf{W}_j^T \mathbf{X} \quad (2.25)$$

权向量调整量为:

$$\Delta \mathbf{W}_j = \eta(d_j - \mathbf{W}_j^T \mathbf{X})\mathbf{X} \quad (2.26a)$$

$\Delta \mathbf{W}_j$ 的各分量为:

$$\Delta w_{ij} = \eta(d_j - \mathbf{W}_j^T \mathbf{X})x_i \quad i=0, 1, \dots, n \quad (2.26b)$$

实际上, 如果在 δ 学习规则中假定神经元转移函数为 $f(\mathbf{W}_j^T \mathbf{X}) = \mathbf{W}_j^T \mathbf{X}$, 则有 $f'(\mathbf{W}_j^T \mathbf{X}) = 1$, 此时式(2.20) 与式(2.25) 相同。因此, LMS 学习规则可以看成是 δ 学习规则的一个特殊情况。该学习规则与神经元采用的转移函数无关, 因而不需要对转移函数求导数, 不仅学习速度较快, 而且具有较高的精度。权值可初始化为任意值。

2.4.5 Correlation 学习规则

Correlation (相关) 学习规则规定学习信号为:

$$r = d_j \quad (2.27)$$

易得出 $\Delta \mathbf{W}_j$ 及 Δw_{ij} 分别为:

$$\Delta \mathbf{W}_j = \eta d_j \mathbf{X} \quad (2.28a)$$

$$\Delta w_{ij} = \eta d_j x_i \quad i=0, 1, \dots, n \quad (2.28b)$$

该规则表明, 当 d_j 是 x_i 的期望输出时, 相应的权值增量 Δw_{ij} 与两者的乘积 $d_j x_i$ 成正比。

如果 Hebb 学习规则中的转移函数为二进制函数, 且有 $o_j = d_j$, 则相关学习规则可看作 Hebb 规则的一种特殊情况。应当注意的是, Hebb 学习规则是无导师学习, 而相关学习规则是有导师学习。这种学习规则要求将权值初始化为零。

2.4.6 Winner-Take-All 学习规则

Winner-Take-All (胜者为王) 学习规则是一种竞争学习规则, 用于无导师学习。一般将网络的某一层确定为竞争层, 对于一个特定的输入 \mathbf{X} , 竞争层的所有 p 个神经元均有输出响应, 其中响应值最大的神经元为在竞争中获胜的神经元, 即:

$$\mathbf{W}_j^T \cdot \mathbf{X} = \max_{i=1, 2, \dots, p} (\mathbf{W}_i^T \mathbf{X}) \quad (2.29)$$

只有获胜神经元才有权调整其权向量 \mathbf{W}_m , 调整量为:

$$\Delta \mathbf{W}_j = \alpha (\mathbf{X} - \mathbf{W}_j) \quad (2.30)$$

式中, $\alpha \in (0, 1]$, 是一个小的学习常数, 一般其值随着学习的进展而减小。由于两个向量的点积越大, 表明两者越近似, 所以调整获胜神经元权值的结果是使 \mathbf{W}_m 进一步接近当前输入 \mathbf{X} 。显然, 当下次出现与 \mathbf{X} 相似的输入模式时, 上次获胜的神经元更容易获胜。在反复的竞争学习过程中, 竞争层的各神经元所对应的权向量被逐渐调整为输入样本空间的聚类中心。

在有些应用中，以获胜神经元为中心定义一个获胜邻域，除获胜神经元调整权值外，邻域内的其他神经元也程度不同地调整权值。权值一般被初始化为任意值并进行归一化处理。

2.4.7 Outstar 学习规则

神经网络中有两类常见节点，分别称为内星节点和外星节点，其特点见图 2.16。图 2.16(a) 中的内星节点总是接受来自各神经元的输入加权信号，因此是信号的汇聚点，对应的权值向量称为内星权向量；图 2.16(b) 中的外星节点总是向各神经元发出输出加权信号，因此是信号的发散点，对应的权值向量称为外星权向量。内星学习规则规定内星节点的输出响应是输入向量 X 和内星权向量 W_j 的点积。该点积反映了 X 与 W_j 的相似程度，其权值按式(2.30) 调整。因此 Winner-Take-All 学习规则与内星规则一致。下面介绍 Outstar (外星) 学习规则。

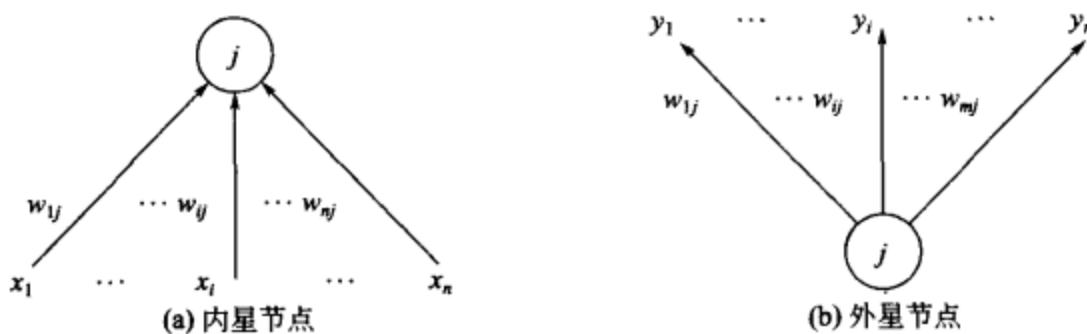


图 2.16 内星节点与外星节点

外星学习规则属于有导师学习，其目的是为了生成一个期望的 m 维输出向量 d ，设对应的外星权向量用 W_j 表示，学习规则如下：

$$\Delta W_j = \eta(d - W_j) \quad (2.31)$$

式中， η 的规定与作用和式(2.30) 中的 α 相同。正像式(2.30) 给出的内星学习规则使节点 j 对应的内星权向量向输入向量 X 靠拢一样，式(2.31) 给出的外星学习规则使节点 j 对应的外星权向量向期望输出向量 d 靠拢。

以上集中介绍了神经网络中几种常用的学习规则，有些规则之间有着内在联系，读者通过比较可体会其异同。对上述各种学习规则的对比总结列于表 2.1 中。

表 2.1 常用学习规则

学习规则	权值调整		权值初始化	学习方式	转移函数
	向量式	元素式			
Hebb	$\Delta W_j = \eta f(W_j^T X) X$	$\Delta w_{ij} = \eta f(W_j^T X) x_i$	0	无导师	任意
Perceptron	$\Delta W_j = \eta [d_j - \text{sgn}(W_j^T X)] X$	$\Delta w_{ij} = \eta [d_j - \text{sgn}(W_j^T X)] x_i$	任意	有导师	二进制
Delta	$\Delta W_j = \eta (d_j - o_j) f(\text{net}_j) X$	$\Delta w_{ij} = \eta (d_j - o_j) f(\text{net}_j) x_i$	任意	有导师	连续
Widrow-Hoff	$\Delta W_j = \eta (d_j - W_j^T X) X$	$\Delta w_{ij} = \eta (d_j - W_j^T X) x_i$	任意	有导师	任意
相关	$\Delta W_j = \eta d_j X$	$\Delta w_{ij} = \eta d_j x_i$	0	有导师	任意
Winner-Take-All	$\Delta W_m = \eta (X - W_m)$	$\Delta W_m = \eta (x_i - w_{im})$	随机、归一化	无导师	连续
Outstar	$\Delta W_j = \eta (d - W_j)$	$\Delta w_{kj} = \eta (d_k - w_{kj})$	0	有导师	连续

本章小结

本章重点介绍了生物神经元的结构及其信息处理机制、人工神经元数理模型、常见的网络拓扑结构和学习规则。其中，神经元的数学模型、神经网络的连接方式以及神经网络的学习规则是决定神经网络信息处理性能的三大要素，因而是本章学习的重点。

(1) 生物神经元的信息处理 树突和轴突用来完成神经元间的通信。树突接收来自其他

神经元的输入，轴突给其他神经元提供输出。神经元与神经元之间的通信连接称为突触。神经元对在各突触点的输入信号以各种方式进行组合，在一定条件下触发产生输出信号，这一信号通过轴突传递给其他神经元。神经元突触是神经信息处理的关键。

(2) 神经元模型 神经元模型可以从“6点假定”的文字描述、模型示意图的符号描述以及解析表达式的数学描述3个方面来理解和掌握。每一个神经元都是一个最基本的信息处理单元，其处理能力表现为对输入信号的整合，整合结果称为净输入。当净输入超过阈值时，神经元激发并通过一个转移函数得到输出。不同的神经元模型是其转移函数不同，本章介绍了4种常用的转移函数。

(3) 神经网络模型 按一定规则将神经元连接成神经网络，才能实现对复杂信息的处理与存储。根据网络的连接特点和信息流向特点，可将其分为前馈层次型、输入输出有反馈的前馈层次型、前馈层内互连型、反馈全互连型和反馈局部互连型等几种常见类型。

(4) 神经网络学习 神经网络在外界输入样本的刺激下不断改变网络的连接权值乃至拓扑结构，以使网络的输出不断地接近期望的输出，这一过程称为神经网络的学习，其本质是对可变权值的动态调整。在学习过程中，网络中各神经元的连接权需按一定规则调整变化，这种权值调整规则称为学习规则。本章简要介绍了几种常用的学习规则，在后面的章节中将结合具体的网络结构进行详细介绍。

思考与练习

- 2.1 人工神经元模型是如何体现生物神经元的结构和信息处理机制的？
- 2.2 若权值只能按1或-1变化，对神经元的学习有何影响？试举例说明。
- 2.3 举例说明什么是有导师学习和无导师学习。
- 2.4 双输入单输出神经网络，初始权向量 $\mathbf{W}(0)=(1, -1)^T$ ，学习率 $\eta=1$ ，4个输入向量为 $\mathbf{X}^1=(1, -2)^T, \mathbf{X}^2=(0, 1)^T, \mathbf{X}^3=(2, 3)^T, \mathbf{X}^4=(1, 1)^T$ ，若采用 Hebb 学习规则，对以下两种情况求第四步训练后的权向量：
 - ① 神经元采用离散型转移函数 $f(\text{net})=\text{sgn}(\text{net})$ ；
 - ② 神经元采用双极性连续型转移函数 $f(\text{net})=\frac{1-e^{-\text{net}}}{1+e^{-\text{net}}}$ 。
- 2.5 某神经网络的转移函数为符号函数 $f(\text{net})=\text{sgn}(\text{net})$ ，学习率 $\eta=1$ ，初始权向量 $\mathbf{W}(0)=(0, 1, 0)^T$ ，两对输入样本为 $\mathbf{X}^1=(2, 1, -1)^T, d^1=-1; \mathbf{X}^2=(0, -1, -1)^T, d^2=1$ 。试用感知器学习规则对以上样本进行反复训练，直到网络输出误差为零，写出每一训练步的净输入 $\text{net}(t)$ 。
- 2.6 某神经网络采用双极性 Sigmoid 函数，学习率 $\eta=0.25$ ，初始权向量 $\mathbf{W}(0)=(1, 0, 1)^T$ ，两对输入样本为 $\mathbf{X}^1=(2, 0, -1)^T, d^1=-1; \mathbf{X}^2=(1, -2, -1)^T, d^2=1$ 。试用 δ 学习规则进行训练，并写出前两步训练结果 [提示：双极性 Sigmoid 函数的导数为 $f'(\text{net})=1/2(1-\sigma^2)$]。
- 2.7 神经网络数据同 2.6 题，试用 Widrow-Hoff 学习规则进行训练，并写出前两步训练结果。
- 2.8 上机编程练习 要求程序具有以下功能：
 - ① 能对6输入单节点网络进行训练；
 - ② 能选用不同的学习规则；
 - ③ 能选用不同的转移函数；
 - ④ 能选用不同的训练样本。
 程序调试通过后，用以上各题提供的数据进行训练。训练时应给出每一步的净输入和权向量调整结果。

3 监督学习神经网络

感知器神经网络是一种典型的前馈神经网络，具有分层结构，信息从输入层进入网络，逐层向前传递至输出层。根据感知器神经元转移函数、隐层数以及权值调整规则的不同，可以形成具有各种功能特点的神经网络。

3.1 单层感知器

1958年，美国心理学家 Frank Rosenblatt 提出一种具有单层计算单元的神经网络，称为 Perceptron，即感知器。感知器模拟人的视觉接受环境信息，并由神经冲动进行信息传递。感知器研究中首次提出了自组织、自学习的思想，而且对所能解决的问题存在着收敛算法，并能从数学上严格证明，因而对神经网络的研究起了重要推动作用。

单层感知器的结构与功能都非常简单，以至于目前在解决实际问题时很少被采用，但由于它在神经网络研究中具有重要意义，是研究其他网络的基础，而且较易学习和理解，适合于作为学习神经网络的起点。

3.1.1 感知器模型

单层感知器是指只有一层处理单元的感知器，如果包括输入层在内，应为两层。其拓扑结构如图 3.1 所示。图中输入层也称为感知层，有 n 个神经元节点，这些节点只负责引入外部信息，自身无信息处理能力，每个节点接收一个输入信号， n 个输入信号构成输入列向量 \mathbf{X} 。输出层也称为处理层，有 m 个神经元节点，每个节点均具有信息处理能力， m 个节点向外输出处理信息，构成输出列向量 \mathbf{O} 。两层之间的连接权值用权值列向量 \mathbf{W}_j 表示， m 个权向量构成单层感知器的权值矩阵 \mathbf{W} 。

3 种列向量分别表示为：

$$\mathbf{X} = (x_1, x_2, \dots, x_i, \dots, x_n)^T$$

$$\mathbf{O} = (o_1, o_2, \dots, o_i, \dots, o_m)^T$$

$$\mathbf{W}_j = (w_{1j}, w_{2j}, \dots, w_{ij}, \dots, w_{nj})^T \quad j=1, 2, \dots, m$$

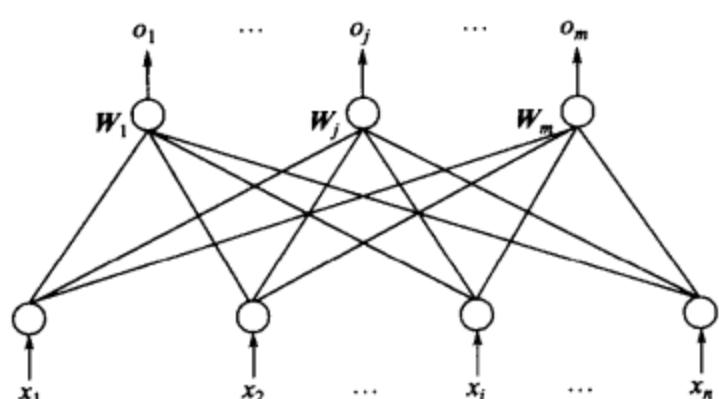


图 3.1 单层感知器

由第 2 章介绍的神经元数学模型知，对于处理层中任一节点，其净输入 net_j 为来自输入层各节点的输入加权和：

$$net'_j = \sum_{i=1}^n w_{ij} x_i \quad (3.1)$$

输出 o_j 由节点的转移函数决定，离散型单计算层感知器的转移函数一般采用符号函数（或单极性阈值函数）。

$$o_j = \text{sgn}(\text{net}'_j - T_j) = \text{sgn}\left(\sum_{i=0}^n w_{ij} x_i\right) = \text{sgn}(W_j^T \mathbf{X}) \quad (3.2)$$

3.1.2 单节点感知器的功能分析

为便于直观分析, 考虑图 3.2 中单节点感知器的情况。不难看出, 单节点感知器实际上就是一个 M-P 神经元模型, 由于采用了符号转移函数, 又称为符号单元。式(3.2)可进一步表达为:

$$o_j = \begin{cases} 1, & W_j^T \mathbf{X} > 0 \\ -1(0), & W_j^T \mathbf{X} \leq 0 \end{cases}$$

下面分三种情况讨论单计算节点感知器的功能。

(1) 设输入向量 $\mathbf{X} = (x_1, x_2)^T$, 则两个输入分量在几何上构成一个二维平面, 输入样本可以用该平面上的一个点表示。节点 j 的输出为:

$$o_j = \begin{cases} 1, & w_{1j}x_1 + w_{2j}x_2 - T_j > 0 \\ -1, & w_{1j}x_1 + w_{2j}x_2 - T_j \leq 0 \end{cases}$$

则由以下方程确定的直线成为二维输入样本空间上的一条分界线:

$$w_{1j}x_1 + w_{2j}x_2 - T_j = 0 \quad (3.3)$$

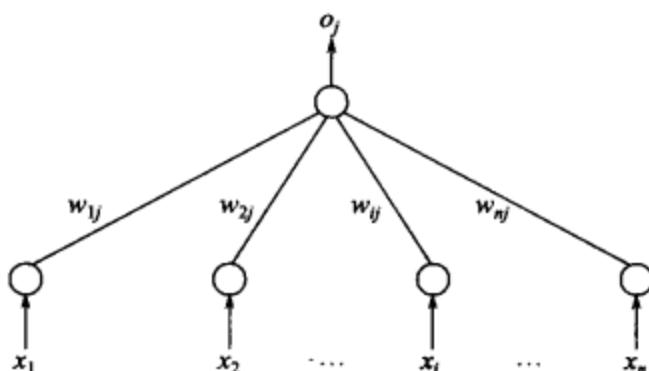


图 3.2 单计算节点感知器

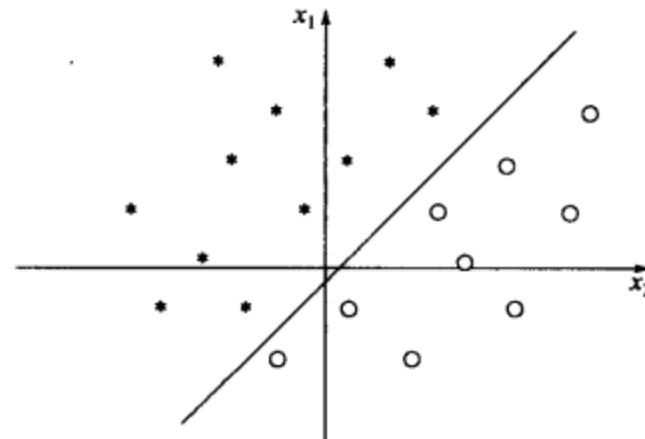


图 3.3 单计算节点感知器对二维样本的分类

线上方的样本用*表示, 它们使 $\text{net}_j > 0$, 从而使输出为 1; 线下方的样本用○表示, 它们使 $\text{net}_j < 0$, 从而使输出为 -1, 见图 3.3。显然, 由感知器权值和阈值确定的直线方程规定了分界线在样本空间的位置, 从而也体现了如何将输入样本分为两类的分类知识与确定规则。假如分界线的初始位置不能将*类样本同○类样本正确分开, 改变权值和阈值, 分界线也会随之改变, 因此总可以将其调整到正确分类的位置。

(2) 设输入向量 $\mathbf{X} = (x_1, x_2, x_3)^T$, 则 3 个输入分量在几何上构成一个三维空间。节点 j 的输出为:

$$o_j = \begin{cases} 1, & w_{1j}x_1 + w_{2j}x_2 + w_{3j}x_3 - T_j > 0 \\ -1, & w_{1j}x_1 + w_{2j}x_2 + w_{3j}x_3 - T_j \leq 0 \end{cases}$$

则由以下方程确定的平面成为三维输入样本空间上的一个分界平面:

$$w_{1j}x_1 + w_{2j}x_2 + w_{3j}x_3 - T_j = 0 \quad (3.4)$$

平面上方的样本用*表示, 它们使 $\text{net}_j > 0$, 从而使输出为 1; 平面下方的样本用○表示, 它们使 $\text{net}_j < 0$, 从而使输出为 -1。同样, 由感知器权值和阈值确定的平面方程规定了分界

平面在样本空间的方向与位置，从而也确定了如何将输入样本分为两类。假如分界平面的初始位置不能将•类样本同○类样本正确分开，改变权值和阈值即改变了分界平面的方向与位置，因此总可以将其调整到正确分类的位置。

(3) 将上述两个特例推广到 n 维输入空间的一般情况，设输入向量 $\mathbf{X} = (x_1, x_2, x_3)^T$ ，则 n 个输入分量在几何上构成一个 n 维空间。由如下方程可定义一个 n 维空间上的超平面：

$$w_{1j}x_1 + w_{2j}x_2 + \cdots + w_{nj}x_n - T_j = 0 \quad (3.5)$$

此平面可以将输入样本分为两类。

通过以上分析可以看出，一个最简单的单计算节点感知器具有分类功能。其分类原理是将分类知识存储于感知器的权向量（包含了阈值）中，由权向量确定的分类判决界面将输入模式分为两类。

下面研究用单计算节点感知器实现逻辑运算问题。

首先，用感知器实现逻辑“与”功能。逻辑“与”的真值表及感知器结构如下：

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

从真值表中可以看出，4个样本的输出有两种情况，一种使输出为0，另一种使输出为1，因此属于分类问题。用感知器学习规则进行训练，得到的连接权值标在图3.4中。令净输入为零，可得到分类判决方程为：

$$0.5x_1 + 0.5x_2 - 0.75 = 0$$

由图3.5可以看出，该方程确定的直线将输出为1的样本点*和输出为0的样本点○正确分开了。从图中还可以看出，该直线并不是唯一解。

同样，可以用感知器实现逻辑“或”功能。逻辑“或”的真值表如下：

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

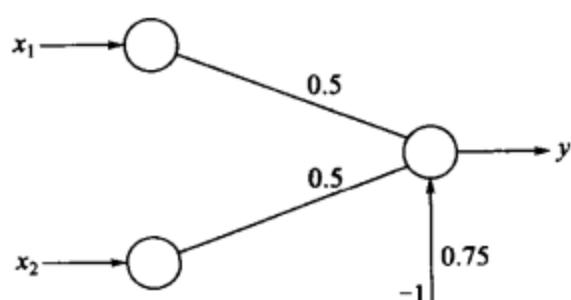


图 3.4 “与”逻辑感知器

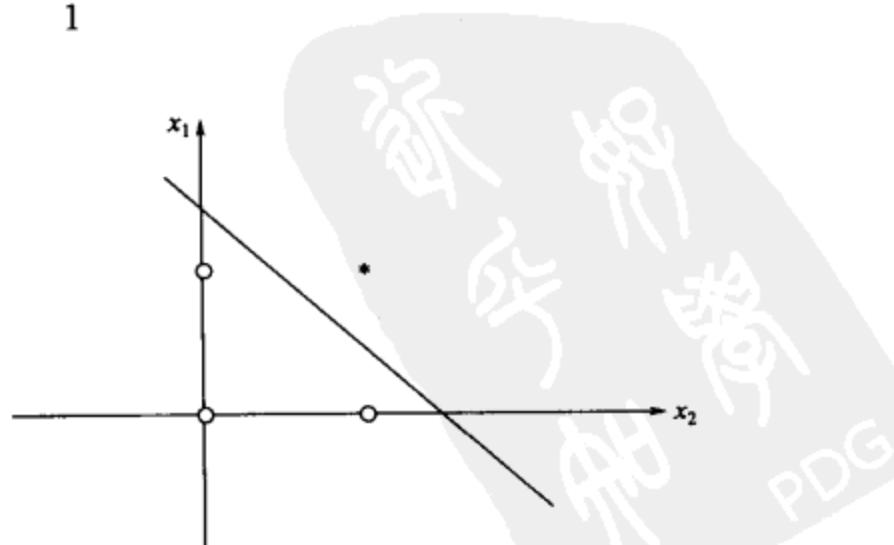


图 3.5 “与”运算的分类

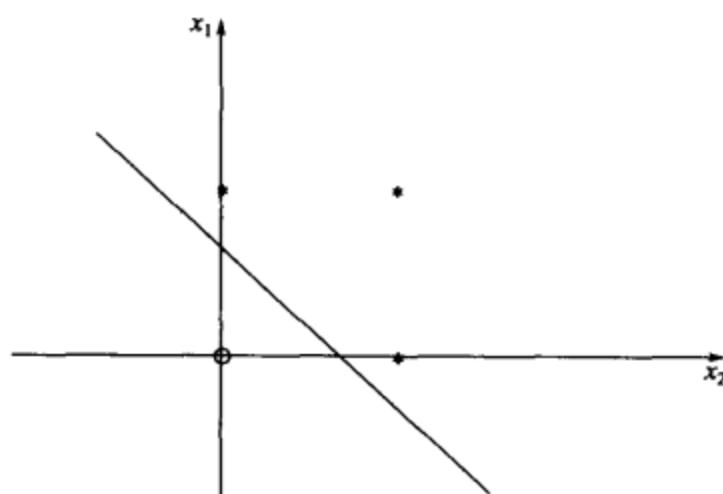


图 3.6 “或” 运算的分类

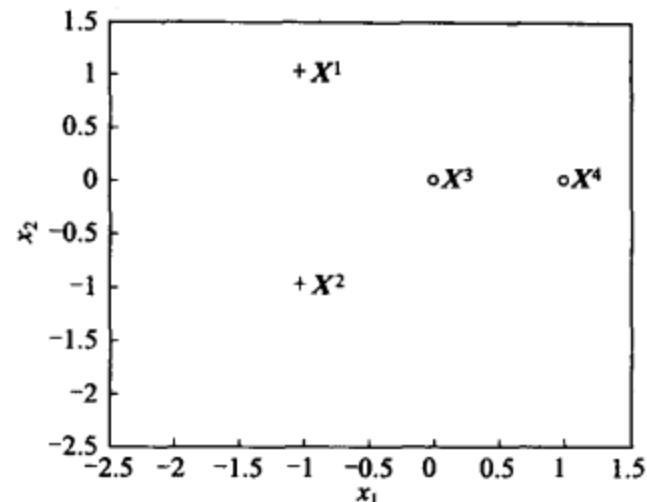


图 3.7 例 3.1 中 4 个输入样本的分布

从真值表中可以看出，4 个样本的输出也分两类，一类使输出为 0，另一类使输出为 1。用感知器学习规则进行训练，得到的连接权值为 $w_1 = w_2 = 1$ ， $T = -0.5$ ，令净输入为零，得分类判决方程为：

$$x_1 + x_2 + 0.5 = 0$$

该直线能把图 3.6 中的两类样本分开，显然，该直线也不是唯一解。

例 3.1 考虑下面定义的分类问题：

$$\left\{ \mathbf{X}^1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, d^1 = 1 \right\} \quad \left\{ \mathbf{X}^2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, d^2 = 1 \right\} \quad \left\{ \mathbf{X}^3 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, d^3 = -1 \right\} \quad \left\{ \mathbf{X}^4 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, d^4 = -1 \right\}$$

其中， $\mathbf{X}^i = (x_1^i, x_2^i)$ 为样本的输入， d^i 为样本的目标输出 ($i = 1, \dots, 3, 4$)。能否用单节点感知器能够求解这个问题？试设计该感知器解决分类问题，用以上 4 个输入向量验证。该感知器分类的正确性，对以下 4 个输入向量进行分类。

$$\mathbf{X}^5 = \begin{bmatrix} -2 \\ 0 \end{bmatrix} \quad \mathbf{X}^6 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{X}^7 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \mathbf{X}^8 = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$$

解 首先将 4 个输入样本标在图 3.7 所示的输入平面上，立即看出，可以找到一条直线将这两类样本分开，因此可以用单节点感知器解决该问题。设分界线方程为：

$$\text{net}_i = \sum_{n=1}^2 w_{ni} x_{in} - T_i = 0$$

其中权值和阈值可以用下一节介绍的感知器学习算法进行训练而得到，也可以采用求解联立方程的方法。

取直线上的两个点分别代入方程，如对于本例可取 $(-0.5, 0)$ 和 $(-0.5, 1)$ ，得到：

$$\begin{cases} -0.5 \times w_{1i} - T_i = 0 \\ -0.5 \times w_{2i} + w_{1i} - T_i = 0 \end{cases}$$

此方程可有无穷多组解。取 $w_{1i} = -1$ ，则有 $w_{2i} = 0$ ， $T_i = 0.5$ ，分别将 4 个输入向量代入感知器的输出表达式 $o = \text{sgn}(W^T X - T)$ ，可得网络的输出分别为 $1, 1, -1, -1$ ，即感知器的输出和教师信号相符，可以进行正确分类。

分别将 5~8 号待分类样本输入设计好的感知器，可以得到感知器的输出分别为 $1, -1, -1, 1$ ；因此在 8 个样本中，如图 3.8 所示， \mathbf{X}^1 ，

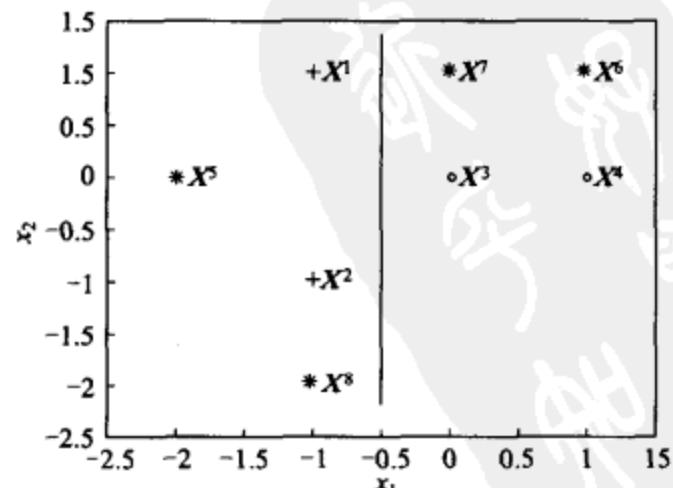


图 3.8 例 3.1 中 8 个样本的分布

$\mathbf{X}^2, \mathbf{X}^5, \mathbf{X}^8$ 属于一类, $\mathbf{X}^3, \mathbf{X}^4, \mathbf{X}^6, \mathbf{X}^7$ 属于另一类。此外, 从图 3.8 中还可以看出, 样本 $\mathbf{X}^5, \mathbf{X}^6$ 的分类不依赖于权值和阈值的选择, 而样本 $\mathbf{X}^7, \mathbf{X}^8$ 的分类则依赖于权值和阈值的选择。

例 3.2 考虑以下 4 类线性可分样本的分类问题:

$$\text{第一类 } \mathbf{X}^1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{X}^2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\text{第二类 } \mathbf{X}^3 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \mathbf{X}^4 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$\text{第三类 } \mathbf{X}^5 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, \mathbf{X}^6 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$$

$$\text{第四类 } \mathbf{X}^7 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \mathbf{X}^8 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$$

试设计一种感知器网络求解该问题。

解 首先画出 8 个输入样本在平面上的分布, 如图 3.9(a) 所示。图中第一类样本用空心圆○表示, 第二类样本用空心方块□表示, 第三类样本用实心圆●表示, 第四类样本用实心方块■表示。从图中可以看出, 可以两条直线将全部样本分为 4 类: 先用一条分界线将 8 个样本分为两组。即第一、三类为一组, 第二、四类为另一组。然后再用一条分界线将四类分开, 其结果如图 3.9(b) 所示。

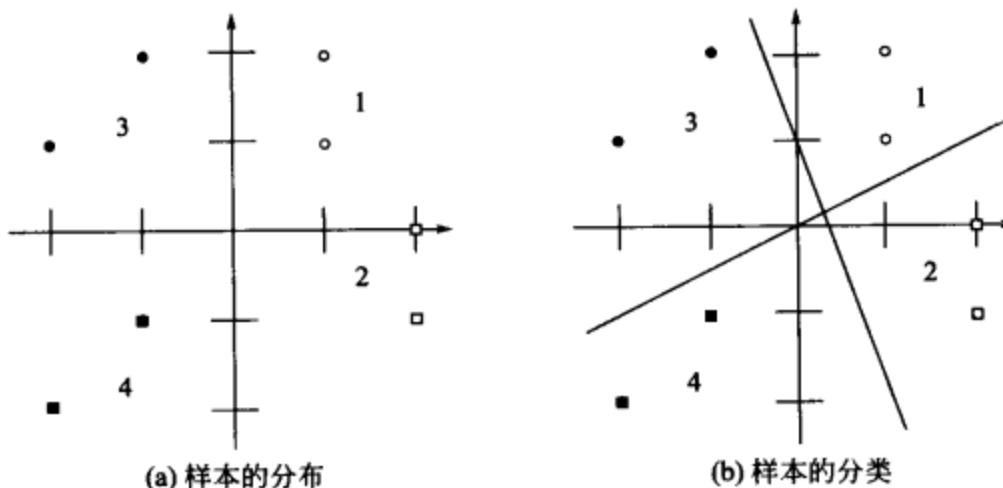


图 3.9 例题 3.2 中 4 类样本的分类

由于每个感知器中的每个计算节点对应的权值和阈值确定了样本空间的一个线性判决边界, 本例中的感知器应有 2 个节点, 如图 3.10 所示, 对应于各样本的期望输出为:

$$\text{第一类 } \mathbf{d}^1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mathbf{d}^2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\text{第二类 } \mathbf{d}^3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{d}^4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\text{第三类 } \mathbf{d}^5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{d}^6 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\text{第四类 } \mathbf{d}^7 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{d}^8 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

可以推知, 具有 M 个节点的单层感知器可对 2^M 个线性可分类别进行分类。

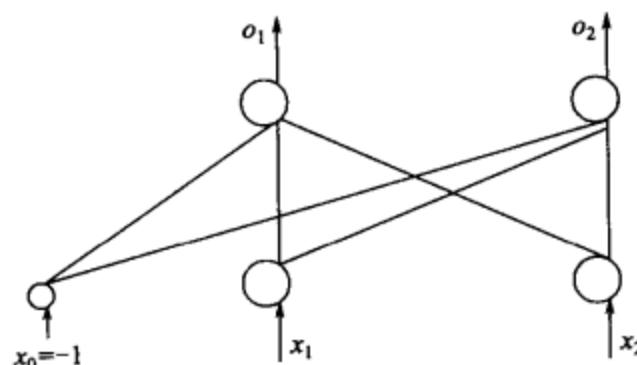


图 3.10 例题 3.2 中的感知器

3.1.3 感知器的学习算法

感知器采用第 2 章介绍的感知器学习规则。考虑到训练过程是感知器权值随每一步调整改变的过程, 为此用 t 表示学习步的序号, 权值看作 t 的函数。 $t=0$ 对应于学习开始前的初始状态, 此时对应的权值为初始化值。训练可按如下步骤进行:

(1) 对各权值 $w_{0j}(0), w_{1j}(0), \dots, w_{nj}(0), j=$

$1, 2, \dots, m$ (m 为计算层的节点数) 赋予较小的非零随机数;

(2) 输入样本对 $\{\mathbf{X}^p, \mathbf{d}^p\}$, 其中 $\mathbf{X}^p = (-1, x_1^p, x_2^p, \dots, x_n^p)$, $\mathbf{d}^p = (d_1^p, d_2^p, \dots, d_m^p)$ 为期望的输出向量 (教师信号), 上标 p 代表样本对的模式序号, 设样本集中的样本总数为 P , 则 $p=1, 2, \dots, P$;

(3) 计算各节点的实际输出 $o_j^p(t) = \text{sgn}[\mathbf{W}_j^T(t)\mathbf{X}^p]$, $j=1, 2, \dots, m$;

(4) 调整各节点对应的权值, $\mathbf{W}_j(t+1) = \mathbf{W}_j(t) + \eta[d_j^p - o_j^p(t)]\mathbf{X}^p$, $j=1, 2, \dots, m$, 其中 η 为学习率, 用于控制调整速度, η 值太大会影响训练的稳定性, 太小则使训练的收敛速度变慢, 一般取 $0 < \eta \leq 1$;

(5) 返回到步骤(2)输入下一对样本。

以上步骤周而复始, 直到感知器对所有样本的实际输出与期望输出相等。

许多学者已经证明, 如果输入样本线性可分, 无论感知器的初始权向量如何取值, 经过有限次调整后, 总能够稳定到一个权向量, 该权向量确定的超平面能将两类样本正确分开。应当看到, 能将样本正确分类的权向量并不是唯一的, 一般初始权向量不同, 训练过程和所得到的结果也不同, 但都能满足误差为零的要求。

例 3.3 某单计算节点感知器有 3 个输入。给定 3 对训练样本如下:

$$\mathbf{X}^1 = (-1, 1, -2, 0)^T \quad d^1 = -1$$

$$\mathbf{X}^2 = (-1, 0, 1.5, -0.5)^T \quad d^2 = -1$$

$$\mathbf{X}^3 = (-1, -1, 1, 0.5)^T \quad d^3 = 1$$

设初始权向量 $\mathbf{W}(0) = (0.5, 1, -1, 0)^T$, $\eta = 0.1$ 。注意, 输入向量中第一个分量 x_0 恒等于 -1, 权向量中第一个分量为阈值, 试根据以上学习规则训练该感知器。

解 第一步, 输入 \mathbf{X}^1 , 得:

$$\mathbf{W}^T(0)\mathbf{X}^1 = (0.5, 1, -1, 0)(-1, 1, -2, 0)^T = 2.5$$

$$o^1(0) = \text{sgn}(2.5) = 1$$

$$\mathbf{W}(1) = \mathbf{W}(0) + \eta[d^1 - o^1(0)]\mathbf{X}^1$$

$$= (0.5, 1, -1, 0)^T + 0.1(-1-1)(-1, 1, -2, 0)^T$$

$$= (0.7, 0.8, -0.6, 0)^T$$

第二步, 输入 \mathbf{X}^2 , 得:

$$\mathbf{W}^T(1)\mathbf{X}^2 = (0.7, 0.8, -0.6, 0)(-1, 0, 1.5, -0.5)^T = -1.6$$

$$o^2(1) = \text{sgn}(-1.6) = -1$$

$$\mathbf{W}(2) = \mathbf{W}(1) + \eta[d^2 - o^2(1)]\mathbf{X}^2$$

$$= (0.7, 0.8, -0.6, 0)^T + 0.1[-1-(-1)](-1, 0, 1.5, -0.5)^T$$

$$= (0.7, 0.8, -0.6, 0)^T$$

由于 $d^2 = o^2(1)$, 所以 $\mathbf{W}(2) = \mathbf{W}(1)$ 。

第三步, 输入 \mathbf{X}^3 , 得:

$$\mathbf{W}^T(2)\mathbf{X}^3 = (0.7, 0.8, -0.6, 0)(-1, -1, 1, 0.5)^T = -2.1$$

$$o^3(2) = \text{sgn}(-2.1) = -1$$

$$\mathbf{W}(3) = \mathbf{W}(2) + \eta[d^3 - o^3(2)]\mathbf{X}^3$$

$$= (0.7, 0.8, -0.6, 0)^T + 0.1[1-(-1)](-1, -1, 1, 0.5)^T$$

$$= (0.5, 0.6, -0.4, 0.1)^T$$

第四步, 继续输入 \mathbf{X} 进行训练, 直到 $d^p = o^p = 0$, $p=1, 2, 3$ 。

3.1.4 感知器的局限性及解决途径

3.1.4.1 感知器的局限性

以上两例说明单计算节点感知器可具有逻辑“与”和逻辑“或”的功能。那么它是否也具有“异或”功能呢？请看下一个例子。

例 3.4 用感知器实现“异或”功能？

“异或”的真值表如下：

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

解 表中的 4 个样本也分为两类，但把它们标在图 3.11 的平面坐标系中可以发现，任何直线也不可能把两类样本分开。

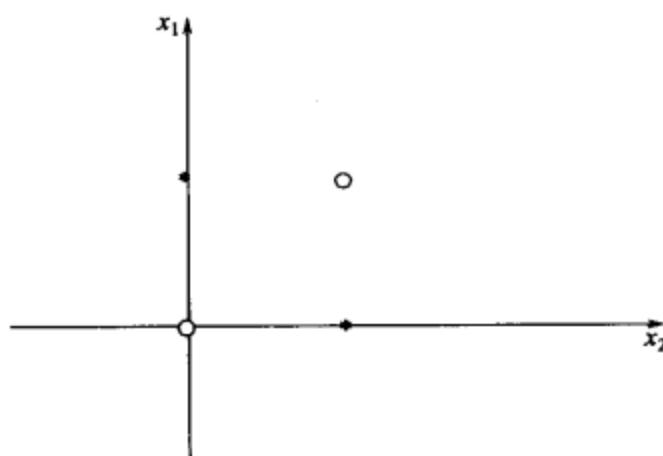


图 3.11 “异或”问题的线性不可分性

如果两类样本可以用直线、平面或超平面分开，称为线性可分，否则为线性不可分。由感知器分类的几何意义可知，由于净输入为零确定的分类判决方程是线性方程，因而它只能解决线性可分问题而不可能解决线性不可分问题。由此可知，单计算层感知器的局限性是：仅对线性可分问题具有分类能力。

例 3.5 证明下面的问题对于两输入单输出感知器而言是不可解的。

$$\left\{ \mathbf{X}^1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, d^1 = 1 \right\} \left\{ \mathbf{X}^2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, d^2 = -1 \right\} \left\{ \mathbf{X}^3 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, d^3 = 1 \right\} \left\{ \mathbf{X}^4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, d^4 = -1 \right\}$$

解 两输入单输出感知器输出方程为 $o = \text{sgn}(W^T \mathbf{X} - T)$ ，将 4 个样本分别代入 $W^T \mathbf{X} - T$ ，并根据相应的 d 值可以得到：

$$\begin{cases} -1 \times w_1 + w_2 - T > 0 & (\text{I}) \\ -1 \times w_1 - w_2 - T < 0 & (\text{II}) \\ 1 \times w_1 - w_2 - T > 0 & (\text{III}) \\ 1 \times w_1 + w_2 - T < 0 & (\text{IV}) \end{cases}$$

将式(I)与式(III)相加，可得 $T < 0$ ；将式(II)与式(IV)相加，可得 $T > 0$ 。两个结论是矛盾的，因此用两输入单输出感知器无法解决该问题。

3.1.4.2 解决途径

前面的分析表明，单计算层感知器只能解决线性可分问题，而大量的分类问题是线性不可分的。克服单计算层感知器这一局限性的有效办法是，在输入层与输出层之间引入隐层作

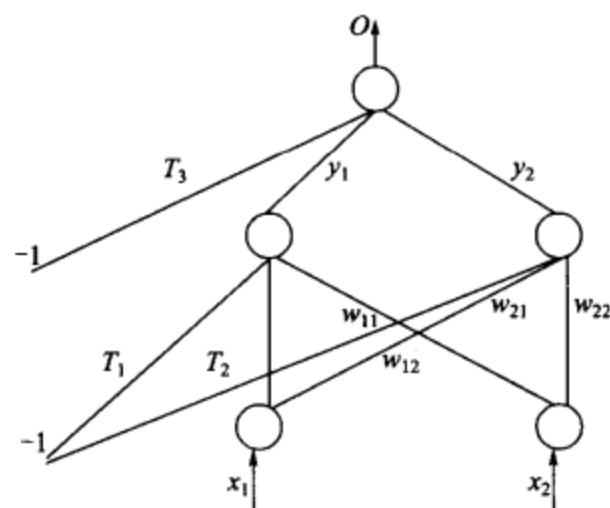


图 3.12 具有两个计算层的感知器

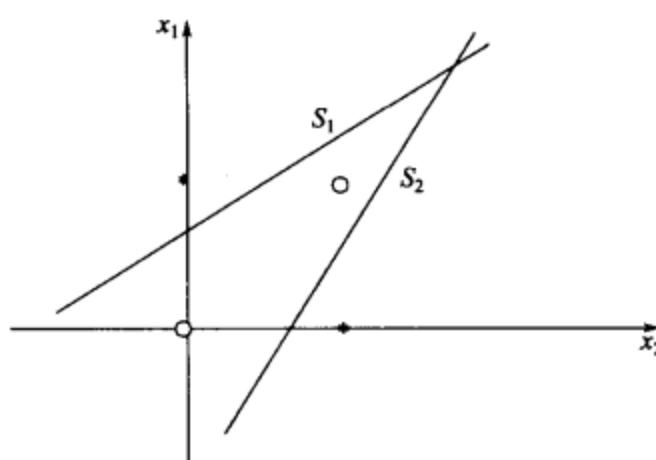


图 3.13 “异或”问题分类

为输入模式的“内部表示”，将单计算层感知器变成多（计算）层感知器。多层感知器是否可以解决线性不可分问题？下面通过一个例子进行分析。

例 3.6 用两计算层感知器解决“异或”问题。

解 图 3.12 给出一个具有单隐层的感知器，其中隐层的两个节点相当于两个独立的符号单元（单计算节点感知器）。根据上节所述，这两个符号单元可分别在 x_1-x_2 平面上确定两条分界直线 S_1 和 S_2 ，从而构成图 3.13 所示的开放式凸域。显然，通过适当调整两条直线的位置，可使两类线性不可分样本分别位于该开放式凸域内部和外部。此时对隐节点 1 来说，直线 S_1 下面的样本使其输出为 $y_1=1$ ，而直线上面的样本使其输出为 $y_1=0$ ；对隐节点 2 来说，直线 S_2 上面的样本使其输出为 $y_2=1$ ，而直线下面的样本使其输出为 $y_2=0$ 。

当输入样本为○类时，其位置处于开放式凸域内部，即同时处在直线 S_1 下方和直线 S_2 上方。根据以上分析，应有 $y_1=1, y_2=1$ 。

当输入样本为•类时，其位置处于开放式凸域外部，即或者同时处在两直线 S_1, S_2 上方，使 $y_1=0, y_2=1$ ；或者同时处在两直线 S_1, S_2 下方，使 $y_1=1, y_2=0$ 。

输出层节点以隐层两节点的输出 y_1, y_2 作为输入，其结构也相当于一个符号单元。如果经过训练，使其具有逻辑“与非”功能，则异或问题即可得到解决。根据“与非”逻辑，当隐节点输出为 $y_1=1, y_2=1$ 时，该节点输出为 $o=0$ ；当隐节点输出为 $y_1=1, y_2=0$ 时，或 $y_1=0, y_2=1$ 时，该节点输出为 $o=1$ 。将 4 种输入样本与各节点的输出情况列于表 3.1，可以看出单隐层感知器确实可以解决异或问题，因此具有解决线性不可分问题的能力。

表 3.1 4 种输入样本与各节点的输出情况

样本	x_1	x_2	y_1	y_2	o	样本	x_1	x_2	y_1	y_2	o
\mathbf{x}^1	0	0	1	1	0	\mathbf{x}^3	1	0	1	0	1
\mathbf{x}^2	0	1	0	1	1	\mathbf{x}^4	1	1	1	1	0

图 3.1 中给出的是单隐层感知器的一般形式。根据上述原理，不难想象，当输入样本为二维向量时，隐层中的每个节点确定了二维平面上的一条分界直线。多条直线经输出节点组合后会构成图 3.14 所示的各种形状的凸域（所谓凸域是指其边界上任意两点之连线均在域内）。通过训练调整凸域的形状，可将两类线性不可分样本分为域内和域外。输出层节点负责将域内外的两类样本进行分类。

当单隐层感知器具有多个节点时，节点数量增加可以使多边形凸域的边数增加，从而在输入空间构建出任意形状的凸域。如果在此基础上再增加一层，成为第二个隐层，则该层的每个节点确定一个凸域，各种凸域经输出层节点组合后成为图 3.15 中的任意形状域。由图中可以看出，由

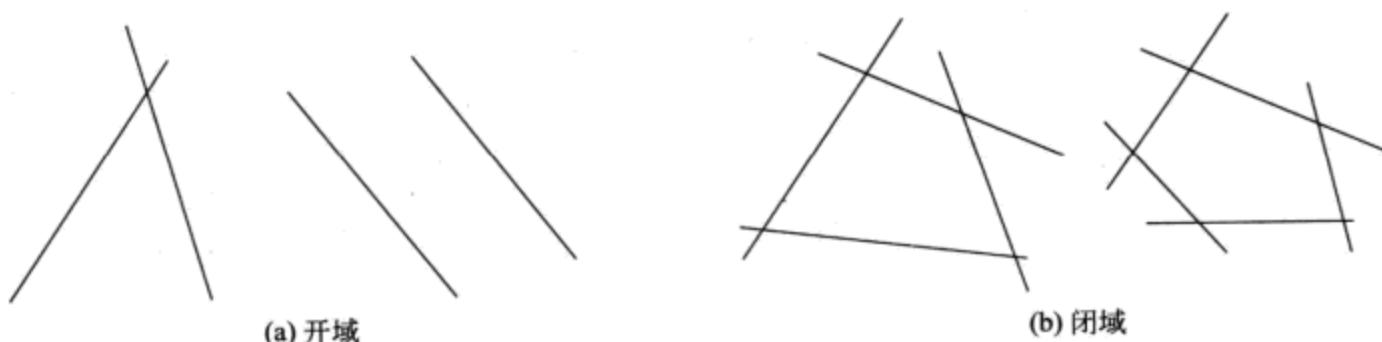


图 3.14 二维平面上的凸域

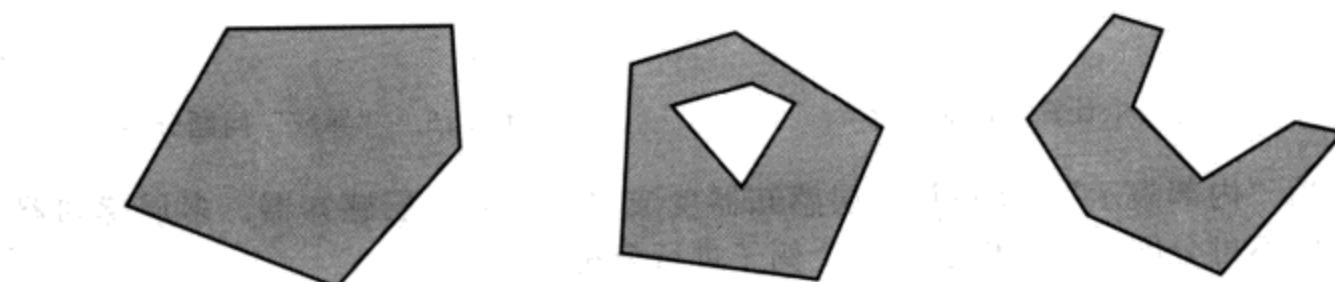


图 3.15 凸域组合的任意形状

凸域组合成任意形状后，意味着双隐层的分类能力比单隐层大大提高。分类问题越复杂，不同类别样本在样本空间的布局越趋于犬牙交错，因而隐层需要的神经元节点数也越多。Kolmogorov 理论指出：双隐层感知器足以解决任何复杂的分类问题。该结论已经过严格的数学证明。

表 3.2 给出具有不同隐层数感知器的分类能力对比。

表 3.2 不同隐层数感知器的分类能力

感知器结构	异或问题	复杂问题	判决域形状	判决域
无隐层				半平面
单隐层				凸域
双隐层				任意复杂形状域

为便于直观描述感知器的分类能力，在上述分析中，将转移函数限定为符号函数或单位阶跃函数。实际上，提高感知器分类能力的另一个途径是，采用非线性连续函数作为神经元节点的转移函数。这样做好处是能使区域边界线的基本线索由直线变成曲线，从而使整个边界线变成连续光滑的曲线。

关于感知器中增加隐层后可以解决非线性可分问题，也可以从非线性映射的角度来理解。若将表 3.1 中的数据点分别放置在由 x_1-x_2 构成的输入空间以及由 y_1-y_2 构成的隐（藏）空间，可以看出由于样本 X^1 和样本 X^4 被映射到隐空间的同一位置，从而使得在输入（藏）空间非线性可分的 4 个样本点映射到隐空间后成为线性可分，因而由输出层节点确定的分类判决界可将映射到隐空间的两类样本分开。

Minsky 和 Papert 在颇具影响的《Perceptron》一书中指出，简单的感知器只能求解线性问题，能够求解非线性问题的网络应具有隐层，但对隐层神经元的学习规则尚无所知。的确，多层感知器从理论上可解决线性不可分问题，但从前面介绍的感知器学习规则看，其权值调整量取决于感知器期望输出与实际输出之差，即 $\Delta W_j(t) = \eta [d_j - o_j(t)] X$ 。对于各隐层节点来说，不存在期望输出，因而该学习规则对隐层权值不适用。

前面已指出，含有隐层的多层感知器能大大提高网络的分类能力，但长期以来没有提出解决权值调整问题的有效算法。1986 年，Rumelhart 和 McClelland 领导的科学家小组在《Parallel Distributed Processing》一书中，对具有非线性连续转移函数的多层感知器的误差反向传播（error back propagation，简称 BP）算法进行了详尽的分析，实现了 Minsky 关于多层网络的设想。由于多层感知器的训练经常采用误差反向传播算法，人们也常把将多层感知器直接称为 BP 网络。

3.2 基于误差反传的多层感知器——BP 神经网络

BP 算法的基本思想是，学习过程由信号的正向传播与误差的反向传播两个过程组成。正向传播时，输入样本从输入层传入，经各隐层逐层处理后，传向输出层。若输出层的实际输出与期望的输出（教师信号）不符，则转入误差的反向传播阶段。误差反传是将输出误差以某种形式通过隐层向输入层逐层反传，并将误差分摊给各层的所有单元，从而获得各层单元的误差信号，此误差信号即作为修正各单元权值的依据。这种信号正向传播与误差反向传播的各层权值调整过程是周而复始地进行的。权值不断调整的过程，也就是网络的学习训练过程。此过程一直进行到网络输出的误差减少到可接受的程度，或进行到预先设定的学习次数为止。

3.2.1 BP 网络模型

采用 BP 算法的多层感知器是至今为止应用最广泛的神经网络，在多层感知器的应用中，以图 3.16 所示的单隐层网络的应用最为普遍。一般习惯将单隐层感知器称为三层感知器，所谓三层包括了输入层、隐层和输出层。

三层感知器中，输入向量为 $X=(x_1, x_2, \dots, x_i, \dots, x_n)^T$ ，图中 $x_0=-1$ 是为隐层神经元引入阈值而设置的；隐层输出向量为 $Y=(y_1, y_2, \dots, y_j, \dots, y_m)^T$ ，图中 $y_0=-1$ 是为输出层神经元引入阈值而设置的；输出层输出向量为 $O=(o_1, o_2, \dots, o_k, \dots, o_l)^T$ ；期望输出向量为 $d=$

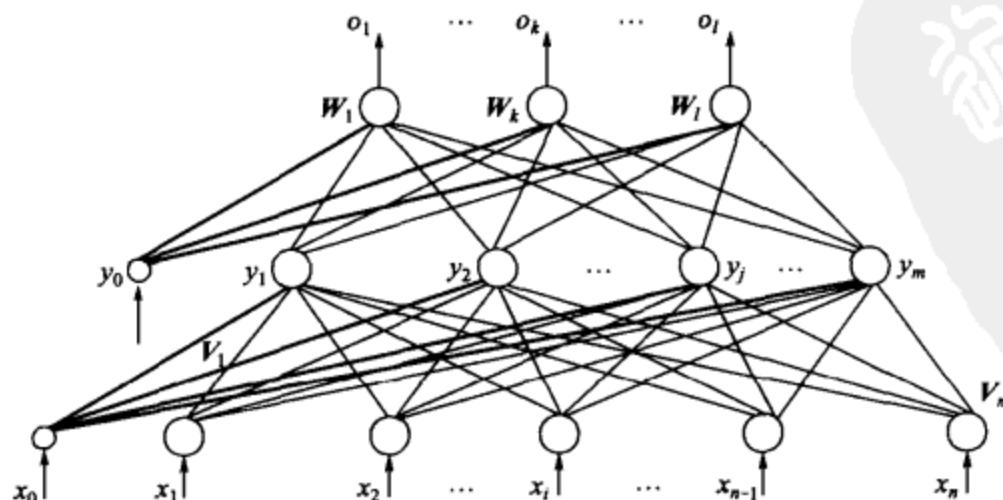


图 3.16 三层 BP 网

$(d_1, d_2, \dots, d_k, \dots, d_l)^T$ 。输入层到隐层之间的权值矩阵用 \mathbf{V} 表示, $\mathbf{V} = (\mathbf{V}_1, \mathbf{V}_2, \dots, \mathbf{V}_j, \dots, \mathbf{V}_m)$, 其中列向量 \mathbf{V}_j 为隐层第 j 个神经元对应的权向量; 隐层到输出层之间的权值矩阵用 \mathbf{W} 表示, $\mathbf{W} = (\mathbf{W}_1, \mathbf{W}_2, \dots, \mathbf{W}_k, \dots, \mathbf{W}_l)$, 其中列向量 \mathbf{W}_k 为输出层第 k 个神经元对应的权向量。下面分析各层信号之间的数学关系。

对于输出层, 有:

$$o_k = f(\text{net}_k) \quad k=1, 2, \dots, l \quad (3.6)$$

$$\text{net}_k = \sum_{j=0}^m w_{jk} y_j \quad k=1, 2, \dots, l \quad (3.7)$$

对于隐层, 有:

$$y_j = f(\text{net}_j) \quad j=1, 2, \dots, m \quad (3.8)$$

$$\text{net}_j = \sum_{i=0}^n v_{ij} x_i \quad j=1, 2, \dots, m \quad (3.9)$$

以上两式中, 转移函数 $f(x)$ 均为单极性 Sigmoid 函数:

$$f(x) = \frac{1}{1+e^{-x}} \quad (3.10)$$

$f(x)$ 具有连续、可导的特点, 且有:

$$f'(x) = f(x)[1-f(x)] \quad (3.11)$$

根据应用需要, 也可以采用双极性 Sigmoid 函数(或称双曲线正切函数):

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

式(3.6) ~ 式(3.10) 共同构成了三层感知器的数学模型。

3.2.2 BP 学习算法

下面以三层感知器为例介绍 BP 学习算法, 然后将所得结论推广到一般多层感知器的情况。

3.2.2.1 网络误差定义与权值调整思路

当网络输出与期望输出不等时, 存在输出误差 E , 定义如下:

$$E = \frac{1}{2} (\mathbf{d} - \mathbf{O})^2 = \frac{1}{2} \sum_{k=1}^l (d_k - o_k)^2 \quad (3.12)$$

将以上误差定义式展开至隐层, 有:

$$\begin{aligned} E &= \frac{1}{2} \sum_{k=1}^l [d_k - f(\text{net}_k)]^2 \\ &= \frac{1}{2} \sum_{k=1}^l [d_k - f(\sum_{j=0}^m w_{jk} y_j)]^2 \end{aligned} \quad (3.13)$$

进一步展开至输入层, 有:

$$\begin{aligned} E &= \frac{1}{2} \sum_{k=1}^l \{d_k - f[\sum_{j=0}^m w_{jk} f(\text{net}_j)]\}^2 \\ &= \frac{1}{2} \sum_{k=1}^l \{d_k - f[\sum_{j=0}^m w_{jk} f(\sum_{i=0}^n v_{ij} x_i)]\}^2 \end{aligned} \quad (3.14)$$

由上式可以看出, 网络误差是各层权值 w_{jk} 、 v_{ij} 的函数, 因此调整权值可改变误差 E (从最小化误差函数的角度看, 误差函数也称为目标函数或代价函数)。

显然, 调整权值的原则是使误差不断地减小, 因此应使权值的调整量与误差的梯度下降成正比, 即:

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} \quad j=0,1,2,\dots,m; \quad k=1,2,\dots,l \quad (3.15a)$$

$$\Delta v_{ij} = -\eta \frac{\partial E}{\partial v_{ij}} \quad i=0,1,2,\dots,n; \quad j=1,2,\dots,m \quad (3.15b)$$

式中, 负号表示梯度下降, 常数 $\eta \in (0,1)$ 表示比例系数, 在训练中反映了学习速率。可以看出, BP 算法属于 δ 学习规则类, 这类算法常被称为误差的梯度下降 (gradient descent) 算法。

3.2.2.2 BP 算法推导

式(3.15) 仅是对权值调整思路的数学表达, 而不是具体的权值调整计算式。下面推导三层 BP 算法权值调整的计算式。事先约定, 在全部推导过程中, 对输出层均有 $j=0,1,2,\dots,m, k=1,2,\dots,l$; 对隐层均有 $i=0,1,2,\dots,n, j=1,2,\dots,m$ 。

对于输出层, 式(3.15a) 可写为:

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} = -\eta \frac{\partial E}{\partial \text{net}_k} \frac{\partial \text{net}_k}{\partial w_{jk}} \quad (3.16a)$$

对隐层, 式(3.15b) 可写为:

$$\Delta v_{ij} = -\eta \frac{\partial E}{\partial v_{ij}} = -\eta \frac{\partial E}{\partial \text{net}_j} \frac{\partial \text{net}_j}{\partial v_{ij}} \quad (3.16b)$$

对输出层和隐层各定义一个误差信号, 令:

$$\delta_k^o = -\frac{\partial E}{\partial \text{net}_k} \quad (3.17a)$$

$$\delta_j^y = -\frac{\partial E}{\partial \text{net}_j} \quad (3.17b)$$

综合应用式(3.7) 和式(3.17a), 可将式(3.16a) 的权值调整式改写为:

$$\Delta w_{jk} = \eta \delta_k^o y_j \quad (3.18a)$$

综合应用式(3.9) 和式(3.17b), 可将式(3.16b) 的权值调整式改写为:

$$\Delta v_{ij} = \eta \delta_j^y x_i \quad (3.18b)$$

可以看出, 只要计算出式(3.18) 中的误差信号 δ_k^o 和 δ_j^y , 权值调整量的计算推导即可完成。下面继续推导如何求 δ_k^o 和 δ_j^y 。

对于输出层, δ_k^o 可展开为:

$$\delta_k^o = -\frac{\partial E}{\partial \text{net}_k} = -\frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial \text{net}_k} = -\frac{\partial E}{\partial o_k} f'(\text{net}_k) \quad (3.19a)$$

对于隐层, δ_j^y 可展开为:

$$\delta_j^y = -\frac{\partial E}{\partial \text{net}_j} = -\frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial \text{net}_j} = -\frac{\partial E}{\partial y_j} f'(\text{net}_j) \quad (3.19b)$$

下面求式(3.19) 中网络误差对各层输出的偏导。

对于输出层, 利用式(3.12), 可得:

$$\frac{\partial E}{\partial o_k} = -(d_k - o_k) \quad (3.20a)$$

对于隐层, 利用式(3.13), 可得:

$$\frac{\partial E}{\partial y_j} = - \sum_{k=1}^l (d_k - o_k) f'(\text{net}_k) w_{jk} \quad (3.20b)$$

将以上结果代入式(3.19)，并应用式(3.11)，得：

$$\delta_k^o = (d_k - o_k) o_k (1 - o_k) \quad (3.21a)$$

$$\begin{aligned} \delta_j^y &= \left[\sum_{k=1}^l (d_k - o_k) f'(\text{net}_k) w_{jk} \right] f'(\text{net}_j) \\ &= \left(\sum_{k=1}^l \delta_k^o w_{jk} \right) y_j (1 - y_j) \end{aligned} \quad (3.21b)$$

至此两个误差信号的推导已完成，将式(3.21)代回到式(3.18)，得到三层感知器的BP学习算法权值调整计算公式为：

$$\Delta w_{jk} = \eta \delta_k^o y_j = \eta (d_k - o_k) o_k (1 - o_k) y_j \quad (3.22a)$$

$$\Delta v_{ij} = \eta \delta_j^y x_i = \eta \left(\sum_{k=1}^l \delta_k^o w_{jk} \right) y_j (1 - y_j) x_i \quad (3.22b)$$

对于一般多层感知器，设共有 h 个隐层，按前向顺序各隐层节点数分别记为 m_1, m_2, \dots, m_h ，各隐层输出分别记为 y^1, y^2, \dots, y^h ，各层权值矩阵分别记为 $W^1, W^2, \dots, W^h, W^{h+1}$ ，则各层权值调整计算公式为：

输出层

$$\Delta w_{jk}^{h+1} = \eta \delta_k^{h+1} y_j^h = \eta (d_k - o_k) o_k (1 - o_k) y_j^h \quad j=0, 1, 2, \dots, m_h; k=1, 2, \dots, l$$

第 h 隐层

$$\Delta w_{ij}^h = \eta \delta_j^h y_i^{h-1} = \eta \left(\sum_{k=1}^l \delta_k^o w_{jk}^{h+1} \right) y_j^h (1 - y_j^h) y_i^{h-1} \quad i=0, 1, 2, \dots, m_{h-1}; j=1, 2, \dots, m_h$$

按以上规律逐层类推，则第一隐层权值调整计算公式：

$$\Delta w_{pq}^1 = \eta \delta_q^1 x_p = \eta \left(\sum_{r=1}^{m_2} \delta_r^2 w_{qr}^2 \right) y_q^1 (1 - y_q^1) x_p \quad p=0, 1, 2, \dots, n; q=1, 2, \dots, m_1$$

三层感知器的BP学习算法也可以写成向量形式。

对于输出层，设 $\mathbf{Y} = (y_0, y_1, y_2, \dots, y_j, \dots, y_m)^T$, $\boldsymbol{\delta}^o = (\delta_1^o, \delta_2^o, \dots, \delta_k^o, \dots, \delta_l^o)^T$ ，则：

$$\Delta \mathbf{W} = \eta (\boldsymbol{\delta}^o \mathbf{Y}^T)^T \quad (3.23a)$$

对于隐层，设 $\mathbf{X} = (x_0, x_1, x_2, \dots, x_i, \dots, x_n)^T$, $\boldsymbol{\delta}^y = (\delta_1^y, \delta_2^y, \dots, \delta_j^y, \dots, \delta_m^y)^T$ ，则：

$$\Delta \mathbf{V} = \eta (\boldsymbol{\delta}^y \mathbf{X}^T)^T \quad (3.23b)$$

容易看出，BP学习算法中，各层权值调整公式形式上都是一样的，均由3个因素决定，即：学习率 η 、本层输出的误差信号 $\boldsymbol{\delta}$ 以及本层输入信号 \mathbf{Y} （或 \mathbf{X} ）。其中输出层误差信号与网络的期望输出和实际输出之差有关，直接反映了输出误差，而各隐层的误差信号与前面各层的误差信号都有关，是从输出层开始逐层反传过来的。

3.2.2.3 BP 算法的信号流向

BP算法的特点是信号的前向计算和误差的反向传播，图3.17清楚地表达了算法的信号流向特点。

由图中可以看出，前向过程是：输入信号 \mathbf{X} 从输入层进入后，通过隐层各节点的内星权向量 \mathbf{V}_j 得到该层的输出信号 \mathbf{Y} ；该信号向前输入到输出层，通过其各节点内星权向量 \mathbf{W}_k 得到该层输出 \mathbf{O} 。反向过程是：在输出层期望输出 \mathbf{d} 与实际输出 \mathbf{O} 相比较得到误差信号 $\boldsymbol{\delta}^o$ ，

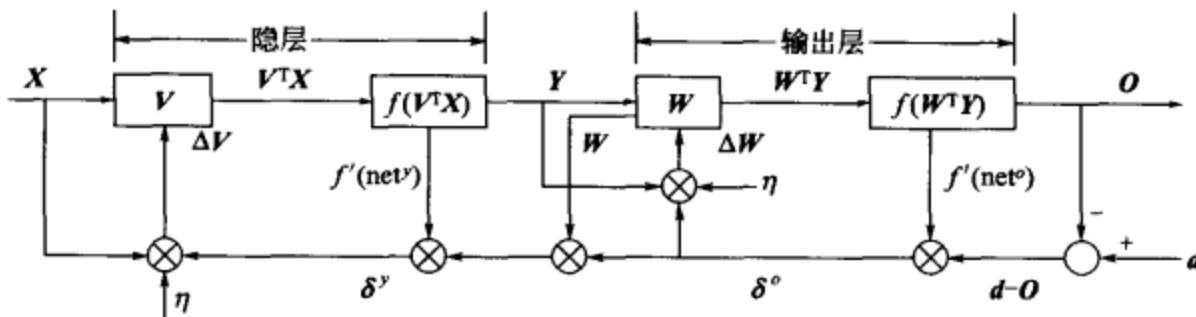


图 3.17 BP 算法的信号流向

由此可计算出输出层权值的调整量；误差信号 δ^o 通过隐层各节点的外星向量反传至隐层各节点，得到隐层的误差信号 δ^y ，由此可计算出隐层权值的调整量。

3.2.3 BP 算法的程序实现

前面推导出的算法是 BP 算法的基础，称为标准 BP 算法。由于目前神经网络的实现仍以软件编程为主，下面介绍标准 BP 算法的编程步骤：

(1) 初始化 对权值矩阵 W 、 V 赋随机数，将样本模式计数器 p 和训练次数计数器 q 置为 1，误差 E 置 0，学习率 η 设为 $(0,1]$ 区间内的小数，网络训练后达到的精度 E_{\min} 设为一正的小数。

(2) 输入训练样本对，计算各层输出 用当前样本 X^p 、 d^p 对向量数组 X 、 d 赋值，用式(3.17)和式(3.15)计算 Y 和 O 中各分量。

(3) 计算网络输出误差 设共有 P 对训练样本，网络对于不同的样本具有不同的误差 E^p

$$= \sqrt{\sum_{k=1}^l (d_k^p - o_k^p)^2} \text{，可将全部样本输出误差的平}$$

方 $(E^p)^2$ 进行累加再开方，作为总输出误差，也可用诸误差中的最大者 E_{\max} 代表网络的总输出误差，实用中更多采用均方根误差 $E_{\text{RME}} =$

$$\sqrt{\frac{1}{P} \sum_{p=1}^P (E^p)^2} \text{ 作为网络的总误差。}$$

(4) 计算各层误差信号 应用式(3.21a)和式(3.21b)计算 δ_k^o 和 δ_j^y 。

(5) 调整各层权值 应用式(3.22a)和式(3.22b)计算 W 、 V 中各分量；

(6) 检查是否对所有样本完成一次轮训 若 $p < P$ ，计数器 p 、 q 增 1，返回步骤(2)，否则转步骤(7)；

(7) 检查网络总误差是否达到精度要求 例如，当用 E_{RME} 作为网络的总误差时，若满足 $E_{\text{RME}} < E_{\min}$ ，训练结束，否则 E 置 0， p 置 1，返回步骤(2)。

标准 BP 算法流程如图 3.18 所示。

目前在实际应用中有两种权值调整方法。从

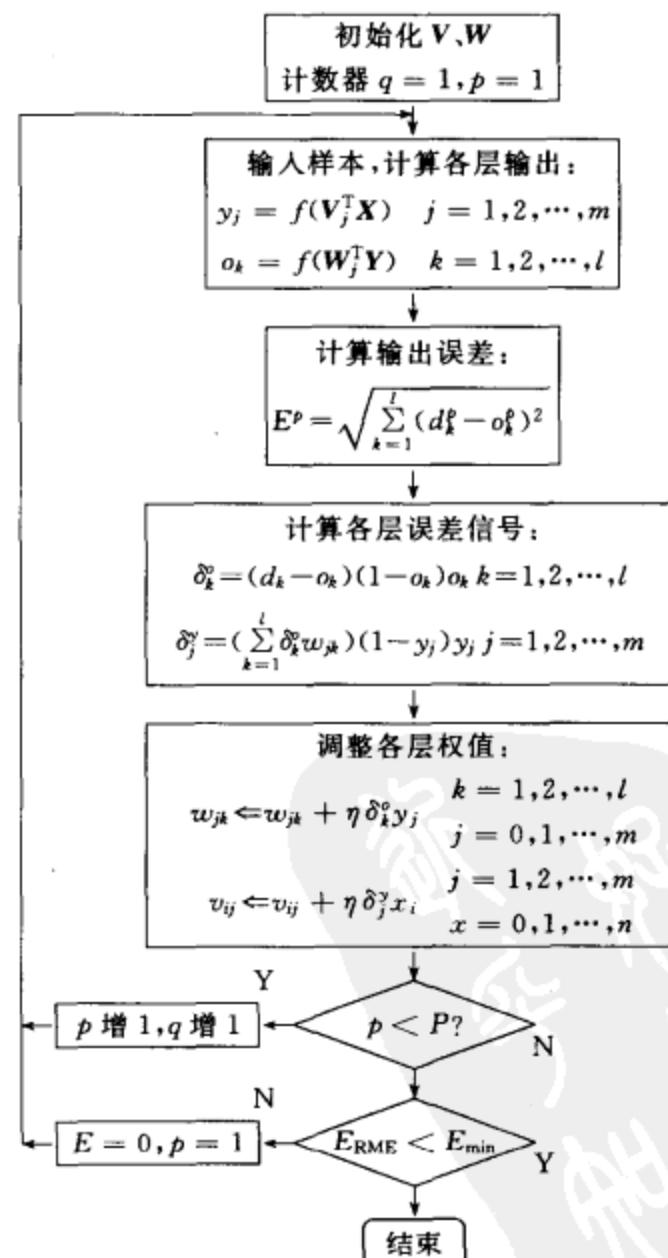


图 3.18 标准 BP 算法流程

以上步骤可以看出，在标准 BP 算法中，每输入一个样本，都要回传误差并调整权值，这种对每个样本轮训的权值调整方法又称为单样本训练。由于单样本训练遵循的是只顾眼前的“本位主义”原则，只针对每个样本产生的误差进行调整，难免顾此失彼，使整个训练的次数增加，导致收敛速度过慢。另一种方法是在所有样本输入之后，计算网络的总误差 $E_{\text{总}}$ ：

$$E_{\text{总}} = \sqrt{\frac{1}{2} \sum_{p=1}^P \sum_{k=1}^l (d_k^p - o_k^p)^2} \quad (3.24)$$

然后根据总误差计算各层的误差信号并调整权值，这种累积误差的批处理方式称为批 (batch) 训练或周期 (epoch) 训练。由于批训练遵循了以减小全局误差为目标的“集体主义”原则，因而可以保证总误差向减小方向变化。在样本数较多时，批训练比单样本训练时的收敛速度快。批训练流程参见图 3.19。

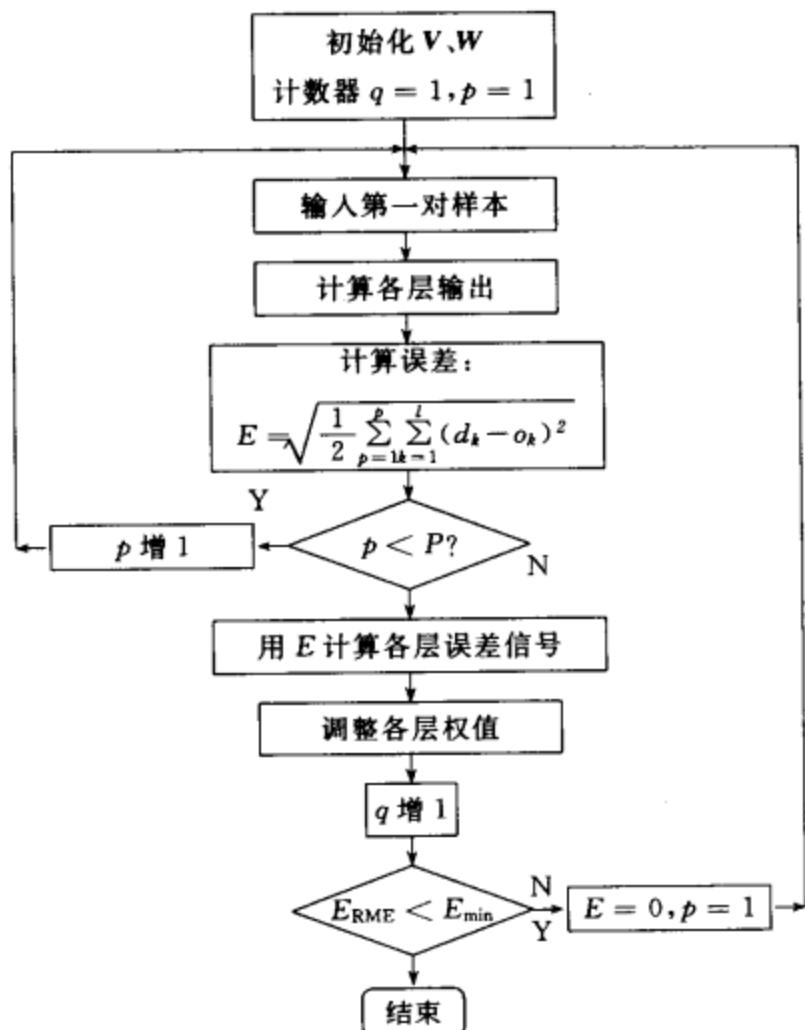


图 3.19 批训练 BP 算法流程

3.2.4 BP 网络的主要能力

BP 网络是目前应用最多的神经网络，这主要归结于基于 BP 算法的多层感知器具有以下一些重要能力：

(1) 非线性映射能力 BP 网络能学习和存储大量输入-输出模式映射关系，而无需事先了解描述这种映射关系的数学方程。只要能提供足够多的样本模式对供 BP 网络进行学习训练，它便能完成由 n 维输入空间到 m 维输出空间的非线性映射。在工程上及许多技术领域中经常遇到这样的问题：对某输入-输出系统已经积累了大量相关的输入-输出数据，但对其内部蕴涵的规律仍未掌握，因此无法用数学方法来描述该规律。这一类问题的共同特点是：①难以得到解析解；②缺乏专家经验；③能够表示和转化为模式识别或非线性映射问题。对于这类问题，多层感知器具有无可比拟的优势。

(2) 泛化能力 BP 网络训练后将所提取的样本对中的非线性映射关系存储在权值矩阵中，在其后的工作阶段，当向网络输入训练时未曾见过的非样本数据时，网络也能完成由输入空间向输出空间的正确映射。这种能力称为多层感知器的泛化能力，它是衡量多层感知器性能优劣的一个重要方面。

(3) 容错能力 BP 网络的魅力还在于，允许输入样本中带有较大的误差甚至个别错误。因为对权矩阵的调整过程也是从大量的样本对中提取统计特性的过程，反映正确规律的知识来自全体样本，个别样本中的误差不能左右对权矩阵的调整。

3.2.5 误差曲面与 BP 算法的局限性

BP 网络的误差是各层权值和输入样本对的函数，因而可表达为：

$$E = F(\mathbf{X}^p, \mathbf{W}, \mathbf{V}, \mathbf{d}^p)$$

从式(3.14)可以看出，误差函数的可调整参数的个数 n_w 等于各层权值数加上阈值数，即： $n_w = m \times (n+1) + l \times (m+1)$ 。所以，误差 E 是 $n_w + 1$ 维空间中一个形状极为复杂的曲面，该曲面上的每个点的“高度”对应于一个误差值，每个点的坐标向量对应着 n_w 个权值，因此称这样的空间为误差的权空间。为了直观描述误差曲面在权空间的起伏变化，图 3.20 给出二维权空间的误差曲面分布情况。通过这样一种简单的情况可以看出或想到，误差曲面的分布有以下两个特点：

(1) 存在平坦区域 从图中可以看出，误差曲面上有些区域比较平坦，在这些区域中，误差的梯度变化很小，即使权值的调整量很大，误差仍然下降缓慢。造成这种情况的原因与各节点的净输入过大有关。以输出层为例，由误差梯度表达式知：

$$\frac{\partial E}{\partial w_{ik}} = -\delta_k^o y_j$$

因此，误差梯度小意味着 δ_k^o 接近零。而从 δ_k^o 的表达式：

$$\delta_k^o = (d_k - o_k)o_k(1 - o_k)$$

可以看出， δ_k^o 接近零有 3 种可能：一种可能是 o_k 充分接近 d_k ，此时应对应着误差的某个谷点；第二种可能是 o_k 始终接近 0；第三种可能是 o_k 始终接近 1。在后两种情况下误差 E 可以是任意值，但梯度很小，这样误差曲面上就出现了平坦区。 o_k 接近 0 或 1 的原因在于 Sigmoid 转移函数具有饱和特性，从图 2.7 可以看出，当净输入（即转移函数的自变量）的绝对值 $\left| \sum_{j=0}^m w_{jk} y_j \right| > 3$ 时， o_k 将处于接近 1 或 0 的饱和区内，此时对权值的变化不太敏感。BP 算法是严格遵从误差梯度降的原则调整权值的，训练进入平坦区后，尽管 $d_k - o_k$ 仍然很大，但由于误差梯度小而使权值调整力度减小，训练只能以增加迭代次数为代价缓慢进行。只要调整方向正确，调整时间足够长，总可以退出平坦区而进入某个谷点。

(2) 存在多个极小点 二维权空间的误差曲面像一片连绵起伏的山脉，其低凹部分就是误差函数的极小点。可以想象，高维权空间的误差曲面“山势”会更加复杂，因而会有更多的极小点。多数极小点都是局部极小，即使是全局极小往往也不是唯一的，但其特点都是误

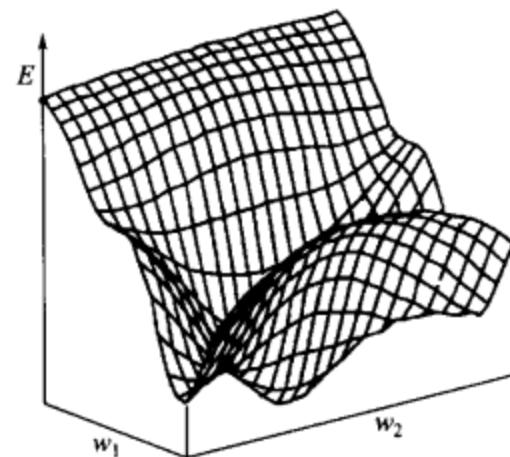


图 3.20 二维权空间的误差曲面

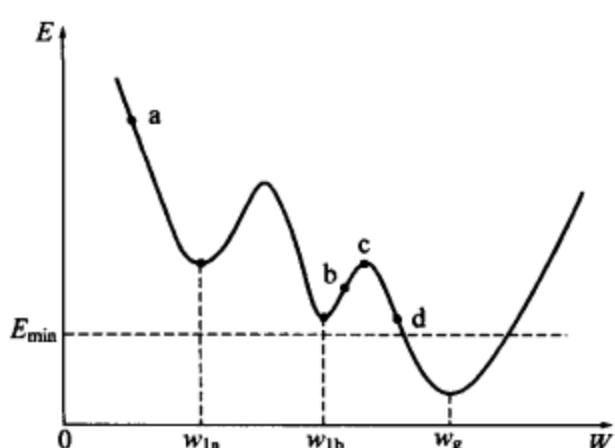


图 3.21 单权值调整时的误差局部极小

差梯度为零。误差曲面的这一特点使以误差梯度降为权值调整依据的 BP 算法无法辨别极小点的性质，因而训练经常陷入某个局部极小点而不能自拔。图 3.21 以单权值调整为例描述了局部极小问题。

误差曲面的平坦区域会使训练次数大大增加，从而影响了收敛速度；而误差曲面的多极小点会使训练陷入局部极小，从而使训练无法收敛于给定误差。以上两个问题都是 BP 算法的固有缺陷，其根源在于基于误差梯度降的权值调整原则每一步求解都取局部最优〔该调整原则即所谓贪心（greedy）算法的原则〕。此外，对于较复杂的多层感知器，标准 BP 算法能否收敛是无法预知的，因为训练最终进入局部极小还是全局极小与网络权值的初始状态有关，而初始权值是随机确定的。

3.3 BP 算法的改进

将 BP 算法用于具有非线性转移函数的三层感知器，可以以任意精度逼近任何非线性函数，这一非凡优势使多层感知器得到越来越广泛的应用。然而标准的 BP 算法在应用中暴露出不少内在的缺陷：

- ① 易形成局部极小而得不到全局最优；
- ② 训练次数多，使得学习效率低，收敛速度慢；
- ③ 隐节点的选取缺乏理论指导；
- ④ 训练时学习新样本有遗忘旧样本的趋势。

针对上述问题，国内外已提出不少有效的改进算法，下面仅介绍其中 3 种较常用的方法。

3.3.1 增加动量项

一些学者于 1986 年提出，标准 BP 算法在调整权值时，只按 t 时刻误差的梯度降方向调整，而没有考虑 t 时刻以前的梯度方向，从而常使训练过程发生振荡，收敛缓慢。为了提高网络的训练速度，可以在权值调整公式中增加一动量项。若用 W 代表某层权矩阵， X 代表某层输入向量，则含有动量项的权值调整向量表达式为：

$$\Delta W(t) = \eta \delta X + \alpha \Delta W(t-1) \quad (3.25)$$

可以看出，增加动量项即从前一次权值调整量中取出一部分迭加到本次权值调整量中， α 称为动量系数，一般有 $\alpha \in (0, 1)$ 。动量项反映了以前积累的调整经验，对于 t 时刻的调整起阻尼作用。当误差曲面出现骤然起伏时，可减小振荡趋势，提高训练速度。目前，BP 算法中都增加了动量项，以至于有动量项的 BP 算法成为一种新的标准算法。

3.3.2 自适应调节学习率

学习率 η 也称为步长，在标准 BP 算法中定为常数，然而在实际应用中，很难确定一个从始至终都合适的学习率。从误差曲面可以看出，在平坦区域内 η 太小会使训练次数增加，因而希望增大 η 值；而在误差变化剧烈的区域， η 太大会因调整量过大而跨过较窄的

“坑凹”处，使训练出现振荡，反而使迭代次数增加。为了加速收敛过程，一个较好的思路是自适应改变学习率，使其该大时增大，该小时减小。

改变学习率的办法很多，其目的都是使其在整个训练过程中得到合理调节。这里介绍其中一种方法：

设一初始学习率，若经过一批次权值调整后使总误差 $E_{\text{总}} \uparrow$ ，则本次调整无效，且 $\eta(t+1) = \beta\eta(t)$ ($\beta < 1$)；若经过一批次权值调整后使总误差 $E_{\text{总}} \downarrow$ ，则本次调整有效，且 $\eta(t+1) = \theta\eta(t)$ ($\theta > 1$)。

3.3.3 引入陡度因子

前面的分析指出，误差曲面上存在着平坦区域。权值调整进入平坦区的原因是神经元输出进入了转移函数的饱和区。如果在调整进入平坦区后，设法压缩神经元的净输入，使其输出退出转移函数的饱和区，就可以改变误差函数的形状，从而使调整脱离平坦区。实现这一思路的具体做法是，在原转移函数中引入一个陡度因子 λ ：

$$o_k = \frac{1}{1 + e^{-net_k/\lambda}} \quad (3.26)$$

当发现 ΔE 接近零而 $d_k - o_k$ 仍较大时，可判断已进入平坦区，此时令 $\lambda > 1$ ；当退出平坦区后，再令 $\lambda = 1$ 。从图 3.22 可以看出，当 $\lambda > 1$ 时， net_k 坐标压缩了 λ 倍，神经元的转移函数曲线的敏感区段变长，从而可使绝对值较大的 net_k 退出饱和值。当 $\lambda = 1$ 时，转移函数恢复原状，对较小的 net_k 具有较高的灵敏度。应用结果表明该方法对于提高 BP 算法的收敛速度十分有效。

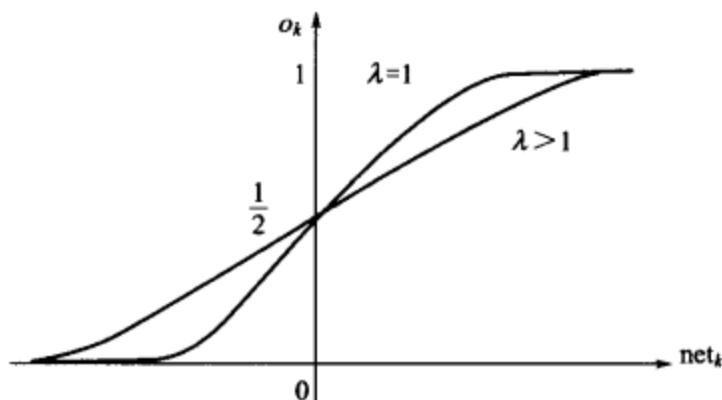


图 3.22 net 压缩前后的转移函数曲线

3.4 BP 网络设计基础

尽管神经网络的研究与应用已经取得巨大的成功，但是在网络的开发设计方面至今还没有一套完善的理论作为指导。应用中采用的主要设计方法是，在充分了解待解决问题的基础上将经验与试探相结合，通过多次改进性试验，最终选出一个较好的设计方案。许多人原以为只要掌握了几种神经网络的结构和算法，就能直接应用了，但真正用神经网络解决问题时才会发现，应用原来不是那么简单。为帮助读者在神经网络应用方面尽快入门或作为参考，本节介绍多层感知器开发设计中常用的基本方法与实用技术，其中关于数据准备等内容设计的原则与方法也适合于后面将要介绍的其他网络。

3.4.1 网络信息容量与训练样本数

多层感知器的分类能力与网络信息容量相关。如用网络的权值和阈值总数 n_w 表征网络

信息容量，研究表明，训练样本数 P 与给定的训练误差 ϵ 之间应满足以下匹配关系：

$$P \approx \frac{n_w}{\epsilon}$$

上式表明，网络的信息容量与训练样本数之间存在着合理匹配关系。在解决实际问题时，训练样本数常常难以满足以上要求。对于确定的样本数，网络参数太少则不足以表达样本中蕴涵的全部规律，而网络参数太多则由于样本信息少而得不到充分训练。因此，当实际问题不能提供较多的训练样本时，必须设法减少样本维数，从而降低 n_w 。

3.4.2 训练样本集的准备

训练数据的准备工作是网络设计与训练的基础，数据选择的科学合理性以及数据表示的合理性对于网络设计具有极为重要的影响。数据准备包括原始数据的收集、数据分析、变量选择和数据预处理等诸多步骤，下面分几个方面介绍有关的知识：

3.4.2.1 输入输出量的选择

一个待建模系统的输入-输出就是神经网络的输入-输出变量。这些变量可能是事先确定的，也可能不够明确，需要进行一番筛选。一般来讲，输出量代表系统要实现的功能目标，其选择确定相对容易一些，例如系统的性能指标、分类问题的类别归属或非线性函数的函数值等。输入量必须选择那些对输出影响大且能够检测或提取的变量，此外还要求各输入变量之间互不相关或相关性很小，这是输入量选择的两条基本原则。如果对某个变量是否适合作网络输入没有把握，可分别训练含有和不含有该输入的两个网络，对其效果进行对比。

从输入、输出量的性质来看，可分为两类：一类是数值变量；一类是语言变量。数值变量的值是数值确定的连续量或离散量。语言变量是用自然语言表示的概念，其“语言值”是用自然语言表示的事物的各种属性。例如，颜色、性别、规模等都是语言变量，其语言值可分别取为红、绿、蓝，男、女，大、中、小等。当选用语言变量作为网络的输入或输出变量时，需将其语言值转换为离散的数值量。

3.4.2.2 输入量的提取与表示

很多情况下，神经网络的输入量无法直接获得，常常需要用信号处理与特征提取技术从原始数据中提取能反映其特征的若干特征参数作为网络的输入。提取的方法与待解决的问题密切相关，下面仅讨论几种典型的情况：

(1) 文字符号输入 在各类字符识别的应用中，均以字符为输入的原始对象。BP 网络的输入层不能直接接受字符输入，必须先对其进行编码，变成网络可接受的形式。下面举一个简单的例子进行说明。

例 3.7 识别英文字母 C、I、T。

如图 3.23 所示，将每个字符纳入 3×3 网格，用数字 1~9 表示网格的序号。设计一个具有 9 个分量的输入向量 X ，其中每一个分量的下标与网格的序号相对应，其取值为 1 或 0

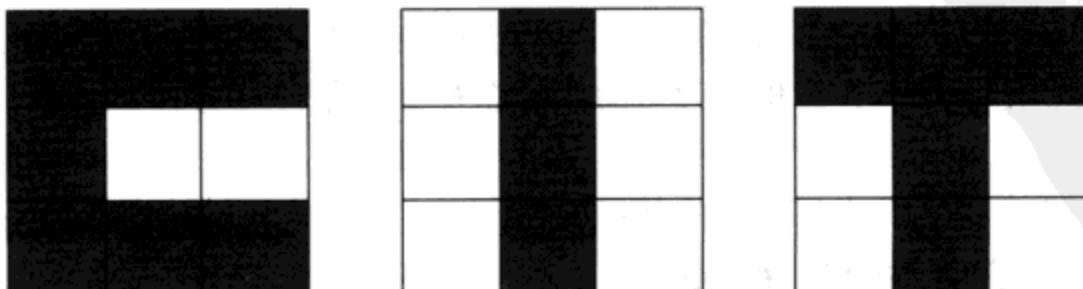


图 3.23 字符的网格表示

代表网格内字符笔迹的有无。则代表 3 个字符样本的输入向量分别为： $\mathbf{X}^C = (111100111)^T$ 、 $\mathbf{X}^I = (010010010)^T$ 和 $\mathbf{X}^T = (111010010)^T$ ，对应的期望输出应为：C 类、I 类和 T 类。关于输出量的表示稍后讨论。

当字符较复杂或要区分的类型较多时，网格数也需增加。此外，对于有笔锋的字符，可用 0~1 之间的小数表达其充满网格的情况，从而反映字符笔画在不同位置的粗细情况。

(2) 曲线输入 多层感知器在模式识别类应用中常被用来识别各种设备输出的波形曲线，对于这类输入模式，常用的表示方法是提取波形在各区间分界点的值，以其作为网络输入向量的分量值。各输入分量的下标表示输入值在波形中的位置，因此分量的编号是严格有序的。

例 3.8 控制系统过渡过程曲线。

为了将图 3.24 的曲线表示为神经网络能接受的形式，将该过程按一定的时间间隔采样，整个过渡过程共采得 n 个样本值，于是某输入向量可表示为：

$$\mathbf{X}^p = (x_1^p, x_2^p, \dots, x_i^p, \dots, x_n^p)^T \quad p=1, 2, \dots, P$$

式中， P 为网络要学习的曲线类型总数。

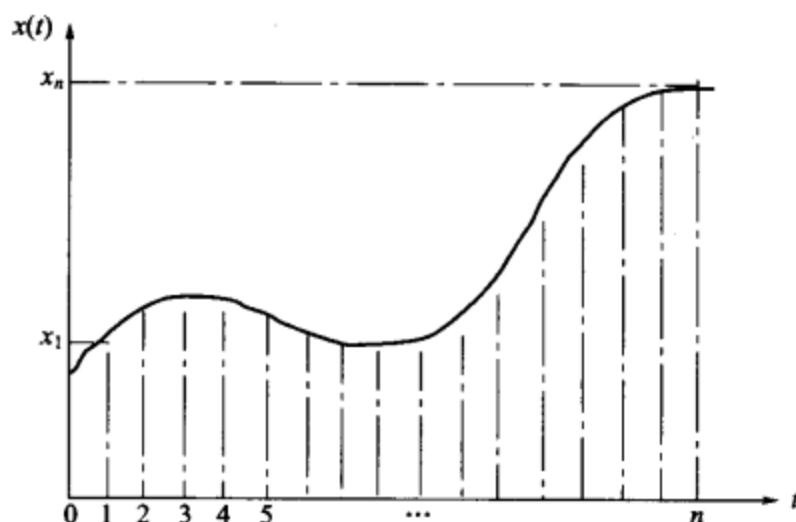


图 3.24 过渡过程曲线的区间划分

采样周期的大小应满足香农采样定理的要求，周期越小，对输入曲线的表达也越精确，但要求网络输入层节点数也越多。采样区间的划分也可以采用不等分的方法，对于曲线变化较大的部分或能提供重要信息的部分，可以将区间分得较细，而对于曲线较平缓的部分，可将区间放宽。

(3) 函数自变量输入 用多层感知器建立系统的数学模型属于典型的非线性映射问题。一般当系统已有大量输入-输出数据对，建模的目的是提取其中隐含的映射规则（即函数关系）。这类应用的输入表示比较简单，一般有几个输入量就设几个分量，一个输入分量对应一个输入层节点。

例 3.9 用 BP 网络实现环境舒适程度测量。

舒适程度无法直接测量，该应用是利用 BP 网络进行多传感器数据融合，得出关于舒适程度的综合结果，属于多自变量函数的建模问题。影响环境舒适程度的变量有很多，可选温度、湿度、风向和风速等影响较大的参数作为输入量，这样网络的输入向量应有 4 个分量，各代表一个影响参数。显然，这种情况下，各分量具有不同的物理意义和量纲。

(4) 图像输入 当需要对物体的图像进行识别时，很少直接将每个像素点的灰度值作为网络的输入。因为图像的像素点常数以万计，不适合作为网络的输入，而且难以从中提取有价值的输入-输出规律。在这类应用中，一般先根据识别的具体目的从图像中提取一些有用

的特征参数，再根据这些参数对输入的贡献进行筛选，这种特征提取属于图像分析的范畴。

例 3.10 天然皮革的外观效果分类。

在真皮服装的制作中，要求做一件成衣所用的数张皮料外观效果一致。如用 BP 网络对天然皮革的外观效果进行分类，不能直接用皮料图像作为网络输入量。在实际应用中，应用图像分析技术从皮革图像中提取了 6 个特征参数，其中 3 个参数描述其纹理特征，另外 3 个描述其颜色特征。一幅像素点为 150×150 的图像只用了 6 个输入分量便可描述其视觉特征。

3.4.2.3 输出量的表示

所谓输出量实际上是指为网络训练提供的期望输出，一个网络可以有多个输出变量，其表示方法通常比输入量容易得多，而且对网络的精度和训练时间影响也不大。输出量可以是数值变量，也可以是语言变量。对于数值类的输出量，可直接用数值量来表示，但由于网络实际输出只能是 $0 \sim 1$ 或 $-1 \sim 1$ 之间的数，所以需要将期望输出进行尺度变换处理，有关的方法在样本的预处理中介绍。下面介绍几种语言变量的表示方法：

(1) “ n 中取 1” 表示法 分类问题的输出变量多用语言变量类型，如质量可分为优、良、中、差 4 个类别。“ n 中取 1” 是令输出向量的分量数等于类别数，输入样本被判为哪一类，对应的输出分量取 1，其余 $n-1$ 个分量全取 0。例如，用 0001、0010、0100 和 1000 分别表示优、良、中、差 4 个类别。这种方法的优点是比较直观，当分类的类别数不是太多时经常采用。

(2) “ $n-1$ ” 表示法 上述方法中没有用到编码全为 0 的情况，如果用 $n-1$ 个全为 0 的输出向量表示某个类别，则可以节省一个输出节点。如上面提到的 4 个类别也可以用 000、001、010 和 100 表示。特别是当输出只有两种可能时，只用一个二进制数便可以表达清楚。如用 0 和 1 代表性别的男和女，考察结果的合格与不合格，性能的好和差等。

(3) 数值表示法 二值分类只适于表示两类对立的分类，而对于有些渐进式的分类，可以将语言值转化为二值之间的数值表示。例如，质量的差与好可以用 0 和 1 表示，而较差和较好这样的渐进类别可用 0 和 1 之间的数值表示，如用 0.25 表示较差、0.5 表示中等、0.75 表示较好等。数值的选择要注意保持由小到大的渐进关系，并要根据实际意义拉开距离。

3.4.2.4 输入输出数据的预处理

(1) 尺度变换 尺度变换也称归一化或标准化，是指通过变换处理将网络的输入、输出数据限制在 $[0,1]$ 或 $[-1,1]$ 区间内。进行尺度变换的主要原因有：①网络的各个输入数据常常具有不同的物理意义和不同的量纲，如某输入分量在 $0 \sim 1 \times 10^5$ 范围内变化，而另一输入分量则在 $0 \sim 1 \times 10^{-5}$ 范围内变化。尺度变换使所有分量都在 $0 \sim 1$ 或 $-1 \sim 1$ 之间变化，从而使网络训练一开始就给各输入分量以同等重要的地位；②BP 网的神经元均采用 Sigmoid 转移函数，变换后可防止因净输入的绝对值过大而使神经元输出饱和，继而使权值调整进入误差曲面的平坦区；③Sigmoid 转移函数的输出在 $0 \sim 1$ 或 $-1 \sim 1$ 之间，作为教师信号的期望输出数据如不进行变换处理，势必使数值大的分量绝对误差大，数值小的分量绝对误差小，网络训练时只针对输出的总误差调整权值，其结果是在总误差中占份额小的输出分量相对误差较大，对输出量进行尺度变换后这个问题可迎刃而解。此外，当输入或输出向量的各分量量纲不同时，应对不同的分量在其取值范围内分别进行变换；当各分量物理意义相同且为同一量纲时，应在整个数据范围内确定最小值 x_{\min} 和最大值

x_{\max} , 进行统一的变换处理。

将输入输出数据变换为 $[0,1]$ 区间的值常用以下变换式:

$$\bar{x}_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \quad (3.27)$$

式中, x_i 代表输入或输出数据; x_{\min} 代表数据变化范围的最小值; x_{\max} 代表数据变化范围的最大值。

将输入输出数据变换为 $[-1,1]$ 区间的值常用以下变换式:

$$x_{\text{mid}} = \frac{x_{\max} + x_{\min}}{2}$$

$$\bar{x}_i = \frac{x_i - x_{\text{mid}}}{\frac{1}{2}(x_{\max} - x_{\min})} \quad (3.28)$$

其中, x_{\min} 代表数据变化范围的中间值, 按上述方法变换后, 处于中间值的原始数据转化为零, 而最大值和最小值分别转换为 1 和 -1。当输入或输出向量中的某个分量取值过于密集时, 对其进行以上预处理可将数据点拉开距离。

(2) 分布变换 尺度变换是一种线性变换, 当样本的分布不合理时, 线性变换只能统一样本数据的变化范围, 而不能改变其分布规律。适于网络训练的样本分布应比较均匀, 相应的样本分布曲线应比较平坦。当样本分布不理想时, 最常用的变换是对数变换, 其他常用的还有平方根、立方根等。由于变换是非线性的, 其结果不仅压缩了数据变化的范围, 而且改善了其分布规律。

3.4.2.5 训练集的设计

网络的性能与训练用的样本密切相关, 设计一个好的训练样本集既要注意样本规模, 又要注意样本质量, 下面讨论这两个问题:

(1) 训练样本数的确定 一般来说训练样本数越多, 训练结果越能正确反映其内在规律, 但样本的收集整理往往受到客观条件的限制。此外, 当样本数多到一定程度时, 网络的精度也很难再提高, 网络误差与训练样本数之间的关系如图 3.25 所示。实践表明, 网络训练所需的样本数取决于输入-输出非线性映射关系的复杂程度, 映射关系越复杂, 样本中含的噪声越大, 为保证一定映射精度所需要的样本数就越多, 而且网络的规模也越大。因此, 可以参考这样一个经验规则, 即: 训练样本数是网络连接权总数的 5~10 倍。

(2) 样本的选择与组织 网络训练中提取的规律蕴涵在样本中, 因此样本一定要有代表性。样本的选择要注意样本类别的均衡, 尽量使每个类别的样本数量大致相等。即使是同一类样本也要照顾样本的多样性与均匀性。按这种“平均主义”原则选择的样本能使网络在训练时见多识广, 而且可以避免网络对样本数量多的类别“印象深”, 而对出现次数少的类别“印象浅”。样本的组织要注意将不同类别的样本交叉输入, 或从训练集中随机选择输入样本。因为同类样本太集中会使网络训练时倾向于只建立与其匹配的映射关系, 当另一类样本集中输入时, 权值的调整又转向新的映射关系而将前面的训练结果否定。当各类样本轮流集中输入时, 网络的训练

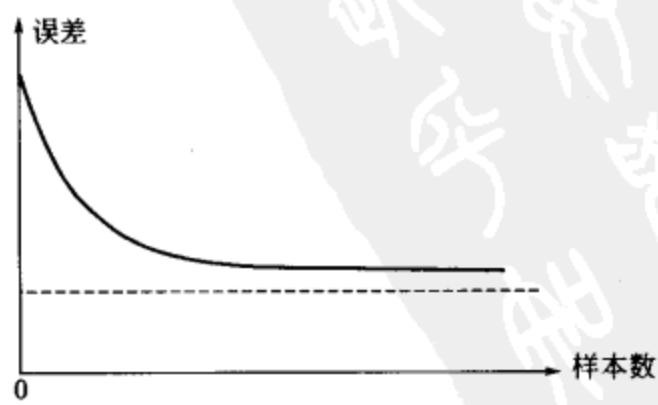


图 3.25 网络误差与训练样本数的关系

会出现振荡，使训练时间延长。

3.4.3 初始权值的设计

网络权值的初始化决定了网络的训练从误差曲面的哪一点开始，因此初始化方法对缩短网络的训练时间至关重要。神经元的转移函数都是关于零点对称的，如果每个节点的净输入均在零点附近，则其输出均处在转移函数的中点。这个位置不仅远离转移函数的两个饱和区，而且是其变化最灵敏的区域，必然使网络的学习速度较快。从净输入的表达式(2.5)可以看出，为了使各节点的初始净输入在零点附近，有两种办法可以采用：一种办法是，使初始权值足够小；另一种办法是，使初始值为+1和-1的权值数相等。应用中对隐层权值可采用第一种办法，而对输出层可采用第二种办法。因为从隐层权值调整公式(3.22b)来看，如果输出层权值太小，会使隐层权值在训练初期的调整量变小，因此采用了第二种权值与净输入兼顾的办法。按以上方法设置的初始权值可使每个神经元一开始都工作在其转移函数变化最大的位置。

3.4.4 BP 网络结构设计

网络的训练样本问题解决以后，网络的输入层节点数和输出层节点数便已确定。因此，BP 网络的结构设计主要是解决设几个隐层和每个隐层设几个隐节点的问题。对于这类问题，不存在通用性的理论指导，但神经网络的设计者们通过大量的实践已经积累了不少经验，下面进行简要介绍以供读者借鉴。

3.4.4.1 隐层数的设计

理论分析证明，具有单隐层的感知器可以映射所有连续函数，只有当学习不连续函数（如锯齿波等）时，才需要两个隐层，所以 BP 网络最多只需两个隐层。在设计 BP 网络时，一般先考虑设一个隐层，当一个隐层的隐节点数很多仍不能改善网络性能时，才考虑再增加一个隐层。经验表明，采用两个隐层时，如在第一个隐层设置较多的隐节点而第二个隐层设置较少的隐节点，则有利于改善 BP 网络的性能。此外，对于有些实际问题，采用双隐层所需要的隐节点总数可能少于单隐层所需的隐节点数。所以，对于增加隐节点仍不能明显降低训练误差的情况，应该想到尝试一下增加隐层数。

3.4.4.2 隐节点数的设计

隐节点的作用是从样本中提取并存储其内在规律，每个隐节点有若干个权值，而每个权值都是增强网络映射能力的一个参数。隐节点数量太少，网络从样本中获取的信息能力就差，不足以概括和体现训练集中的样本规律；隐节点数量过多，又可能把样本中非规律性的内容（如噪声等）也学会记牢，从而出现所谓“过度吻合”问题，反而降低了泛化能力。此外，隐节点数太多还会增加训练时间。

设置多少个隐节点取决于训练样本数的多少、样本噪声的大小以及样本中蕴涵规律的复杂程度。一般来说，波动次数多、幅度变化大的复杂非线性函数要求网络具有较多的隐节点来增强其映射能力。

确定最佳隐节点数的一个常用方法称为试凑法，可先设置较少的隐节点训练网络，然后逐渐增加隐节点数，用同一样本集进行训练，从中确定网络误差最小时对应的隐节点数。在用试凑法时，可以一些确定隐节点数的经验公式。这些公式计算出来的隐节点数只是一种粗

略的估计值，可作为试凑法的初始值。下面介绍几个公式：

$$m = \sqrt{n+l} + \alpha \quad (3.29)$$

$$m = \log_2 n \quad (3.30)$$

$$m = \sqrt{nl} \quad (3.31)$$

式中， m 为隐层节点数； n 为输入层节点数； l 为输出节点数； α 为 1~10 之间的常数。

试凑法的另一种作法是先设置较多的隐节点，进行训练时采用以下误差代价函数：

$$E_f = E_{\text{总}} + \epsilon \sum_{h,j,i} |w_{ij}^h| \quad h=1,2; j=1,2,\dots,m; i=1,2,\dots,n$$

式中， $E_{\text{总}}$ 为式(3.24) 所定义的网络输出误差的平方和；对于单隐层 BP 网，第二项中 n 表示输入层节点数； m 为隐层节点数，其作用相当于引入一个遗忘项，其目的是为了使训练后的连接权值尽量小。为此求 E_f 对 w_{ij}^h 的偏导为：

$$\frac{\partial E_f}{\partial w_{ij}^h} = \frac{\partial E_{\text{总}}}{\partial w_{ij}^h} + \epsilon \text{sgn}(w_{ij}^h)$$

利用上式，仿照 3.2.2.2 的推导过程可得出相应的学习算法。根据该算法，在训练过程中影响小的权值将逐渐衰减到零，因此可以去掉相应的节点，最后保留下来的即为最佳隐节点数。

3.4.5 网络训练与测试

网络设计完成后，要应用设计值进行训练。训练时对所有样本正向运行一轮并反向修改权值一次称为一次训练。在训练过程中要反复使用样本集数据，但每一轮最好不要按固定的顺序取数据。通常训练一个网络需要成千上万次。

网络的性能好坏主要看其是否具有很好的泛化能力，而对泛化能力的测试不能用训练集的数据进行，要用训练集以外的测试数据来进行检验。一般的做法是，将收集到的可用样本随机地分为两部分：一部分作为训练集；另一部分作为测试集。如果网络对训练集样本的误差很小，而对测试集样本的误差很大，说明网络已被训练得过度吻合，因此泛化能力很差。如用 $*$ 代表训练集数据，用 \circ 代表测试集数据，过度训练的极端情况下网络实现的是类似查表的功能，如图 3.26 所示。

在隐节点数一定的情况下，为获得好的泛化能力，存在着一个最佳训练次数 t_0 。为了说明这个问题，训练时将训练与测试交替进行，每训练一次记录一个训练均方误差，然后保持网络权值不变，用测试数据正向运行网络，记录测试均方误差。利用两种误差数据可绘出图 3.27 中的两条均方误差随训练次数变化的曲线。

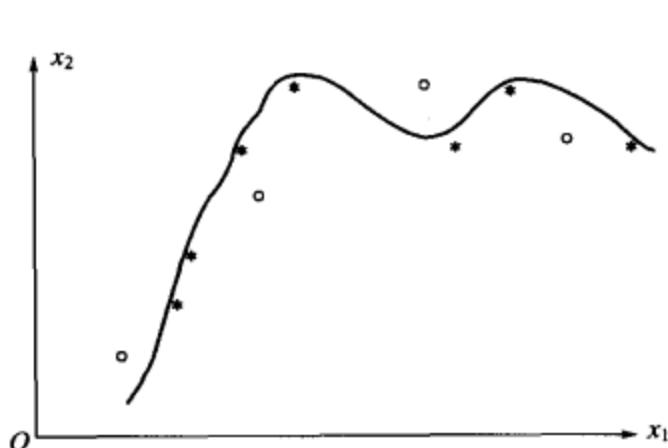


图 3.26 训练误差小而测试误差大

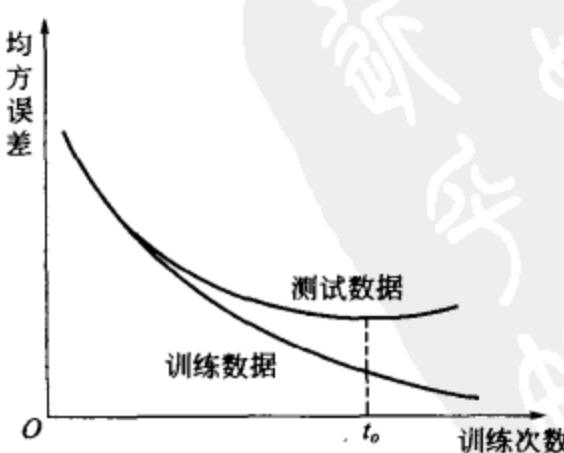


图 3.27 两种均方误差曲线

从误差曲线可以看出，在某一个训练次数 t_0 之前，随着训练次数的增加，两条误差曲线同时下降。当超过这个训练次数时，训练误差继续减小而测试误差则开始上升。因此，该训练次数 t_0 即为最佳训练次数，在此之前停止训练称为训练不足，在此之后则称为训练过度。

3.5 BP 网络应用与设计实例

采用 BP 算法的多层感知器是神经网络在各个领域中应用最广泛的一类网络，已经成功地解决了大量实际问题。本节介绍几例应用，通过了解例子中解决问题的思路可以进一步掌握应用 BP 网络解决实际问题的设计方法和技巧。

3.5.1 BP 网络用于催化剂配方建模

随着化工技术的发展，各种新型催化剂不断问世，在产品的研制过程中，需要制定优化指标并设法找出使指标达到最佳值的优化因素组合，因此属于典型的非线性优化问题。目前常用的方法是采用正交设计法安排实验，利用实验数据建立指标与因素间的回归方程，然后采用某种寻优法，求出优化配方与优化指标。这种方法的缺陷是，数学模型粗糙，难以描述优化指标与各因素之间的非线性关系，以其为基础的寻优结果误差较大。

理论上已经证明，三层前馈神经网络可以任意精度逼近任意连续函数。本例采用 BP 神经网络对脂肪醇催化剂配方的实验数据进行学习，以训练后的网络作为数学模型映射配方与优化指标之间的复杂非线性关系，获得了较高的精度。网络设计方法与建模效果如下：

(1) 网络结构设计与训练 首先利用正交表安排实验，得到一批准确的实验数据作为神经网络的学习样本。根据配方的因素个数和优化指标的个数设计神经网络的结构，然后用实验数据对神经网络进行训练。完成训练之后的多层前馈神经网络，其输入与输出之间形成了一种能够映射配方与优化指标内在联系的连接关系，可作为仿真实验的数学模型。图 3.28 给出针对五因素、三指标配方的实验数据建立的三层前馈神经网络。五维输入向量与配方组成因素相对应，三维输出向量与三个待优化指标〔脂肪酸甲脂转化率 TR (%)、脂肪醇产率 Y_{OH} (%) 和脂肪醇选择性 S_{OH} (%)〕相对应。通过试验确定隐层结点数为 4。正交表安排了 18 组实验，从而得到 18 对训练样本。训练时采用了式(3.25) 中的改进 BP 算法。

(2) BP 网络模型与回归方程仿真结果的对比 表 3.3 给出 BP 网络配方模型与回归方程建立的配方模型的仿真结果对比。其中回归方程为经二次多元逐步回归分析，在一定置信

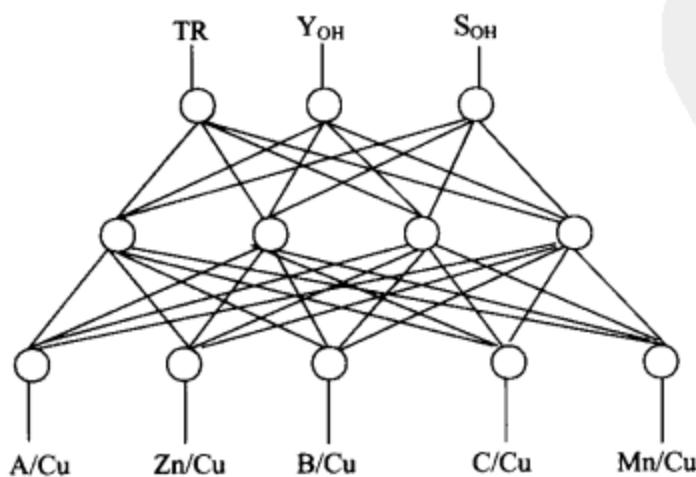


图 3.28 催化剂配方的神经网络模型

水平下经过 F 检验而确定的最优回归方程。从表中可以看出，采用 BP 算法训练的多层前馈神经网络具有较高的仿真精度。

表 3.3 催化剂配方的神经网络模型与回归方程模型输出结果对比

编号	A/Cu	Zn/Cu	B/Cu	C/Cu	Mn/Cu	TR ₁ /%	TR ₂ /%	TR ₃ /%	Y _{OH₁} /%	Y _{OH₂} /%	Y _{OH₃} /%	S _{OH₁} /%	S _{OH₂} /%	S _{OH₃} /%
1	0.0500	0.130	0.080	0.140	0.040	94.50	94.62	83.83	96.30	96.56	95.98	97.80	97.24	102.83
2	0.0650	0.070	0.120	0.160	0.020	88.05	88.05	92.43	75.50	75.97	76.50	86.5	86.67	79.65
3	0.0800	0.190	0.080	0.060	0.000	60.25	60.43	82.03	40.21	41.43	44.87	96.25	95.36	81.92
4	0.0950	0.110	0.060	0.160	0.040	93.05	93.11	94.31	97.31	96.29	105.11	99.30	99.39	103.08
5	0.1100	0.050	0.020	0.060	0.020	94.65	94.72	85.79	88.55	88.06	77.89	95.20	97.49	87.12
6	0.1250	0.170	0.000	0.140	0.000	96.05	95.96	97.08	95.50	96.69	105.43	99.50	99.52	104.71
7	0.1400	0.090	0.160	0.040	0.040	61.00	61.13	65.39	59.72	58.90	54.76	67.35	69.10	73.52
8	0.155	0.030	0.120	0.140	0.020	70.40	70.39	80.44	37.50	41.83	46.36	52.25	51.38	71.45
9	0.1700	0.150	0.100	0.040	0.000	83.30	83.32	70.22	82.85	82.48	59.50	99.20	96.53	74.30
10	0.0500	0.070	0.060	0.120	0.050	84.50	85.27	70.22	90.90	90.46	91.51	95.90	97.87	92.75
11	0.0650	0.190	0.040	0.020	0.030	69.50	69.45	80.77	61.80	65.03	55.22	88.20	92.41	98.44
12	0.0800	0.130	0.000	0.120	0.010	94.55	94.60	94.75	97.60	95.74	92.44	103.40	97.93	101.65
13	0.095	0.050	0.160	0.020	0.050	70.95	69.51	92.88	62.54	60.40	52.50	60.10	62.63	68.12
14	0.110	0.170	0.140	0.100	0.030	87.20	87.16	78.64	91.00	89.19	76.92	103.60	99.36	92.22
15	0.125	0.110	0.100	0.000	0.010	64.20	64.08	69.59	58.30	59.12	54.02	58.90	60.22	72.50
16	0.140	0.030	0.080	0.100	0.050	86.15	86.15	82.40	75.65	61.43	29.93	86.50	78.07	79.28
17	0.155	0.150	0.040	0.000	0.030	77.15	77.17	75.23	71.90	71.72	83.94	91.80	91.74	94.23
18	0.170	0.090	0.020	0.080	0.010	96.05	96.00	87.05	94.60	94.62	94.61	98.00	99.12	90.35

注：表中，下标 1 表示实测结果，下标 2 表示神经网络输出结果，下标 3 表示回归方程计算结果。

3.5.2 BP 网络用于汽车变速器最佳挡位判定

汽车在不同状态参数下运行时，能获得最佳动力性与经济性的挡位称为最佳挡位。最佳挡位与汽车运行状态参数之间具有某种非线性关系，称为换挡规律。通常获得换挡规律有两种方法：一是通过学习优秀驾驶员的换挡经验，提取最佳换挡规律；二是根据汽车自动变速理论，在一定约束条件下按某种目标函数通过优化实验获取换挡规律。无论哪种方法，所获得的换挡规律都是离散数据。需要用各种数据处理的方法建立数学模型来表达蕴藏在其中的内在规律。在汽车运动状态的参数较多而要求挡位判定精度又较高的情况下，用传统的函数拟合等方法非常费事。

从神经网络角度看，汽车最佳换挡问题是一个十分简单的非线性分类问题。可以直接用过去积累的数据对 BP 网络进行离线训练，也可以让 BP 网络在线向优秀驾驶员学习。网络设计方法如下：

(1) 输入输出设计 汽车运行状态参数包括车速 v 、油门开度 α 和加速度 a 三个参数，因此输入向量 X 有 3 个分量，分别代表 3 个状态参数。对于汽车的 4 个挡位，输出层应设 4 个节点，网络输出的挡位信号可用“ n 中取 1”法编码，即 1000 代表 1 挡，0100 代表 2 挡，0010 代表 3 挡，0001 代表 4 挡。

(2) 隐层设计 前已述及，BP 网络隐层及隐层节点数的设计与样本中蕴涵规律的复杂程度相关。汽车运行状态参数与最佳挡位之间的规律是分段非线性的，为简单直观，图

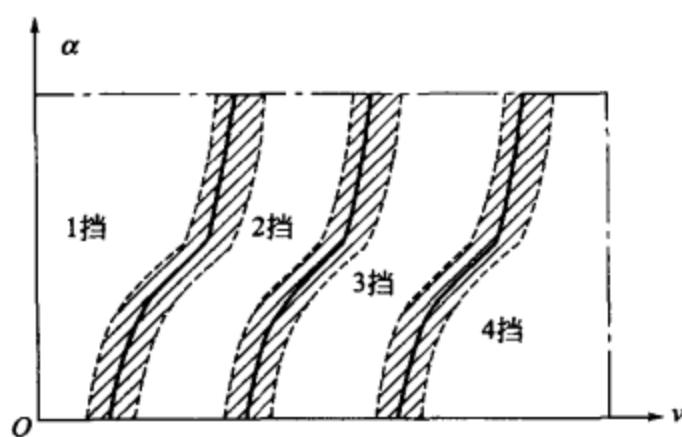


图 3.29 两参数换挡规律

3.29 给出某车型的两参数换挡规律。由于图中换挡规律曲线不连续，BP 网络需要设两个隐层。通过试验比较，确定该 BP 网络各层节点数为 3-3-3-4。

3.5.3 BP 网络用于图像压缩编码

Ackley 和 Hinton 等 1985 年提出了利用多层前馈神经网络的模式变换能力实现数据编码的基本思想。其原理是，把一组输入模式通过少量的隐层节点映射到一组输出模式，并使输出模式等同于输入模式。当中间隐层的节点数

比输入模式维数少时，就意味着隐层能更有效地表达输入模式，并把这种表达传给输出层。在这个过程中，输入层和隐层的变换可以看成是压缩编码的过程；而隐层和输出层的变换可以看成是解码过程。

用 BP 网络实现图像数据压缩时，只需一个隐层，网络结构如图 3.30 所示。输入层和输出层均含有 $n \times n$ 个神经元，每个神经元对应于 $n \times n$ 图像分块中的一个像素。隐层神经元的数量由图像压缩比决定，如 $n=16$ 时，取隐层神经元数为 $m=8$ ，则可将 256 像素的图像块压缩为 8 像素。设用于学习的图像有 $N \times N$ 个像素，训练时从中随机抽取若干个 $n \times n$ 图像块作为训练样本，并使教师模式和输入模式相等。通过调整权值使训练集图像的重建误差达到最小。训练后的网络就可以用来执行图像的数据压缩任务了，此时隐层输出向量便是数据压缩结果，而输出层的输出向量便是图像重建的结果。

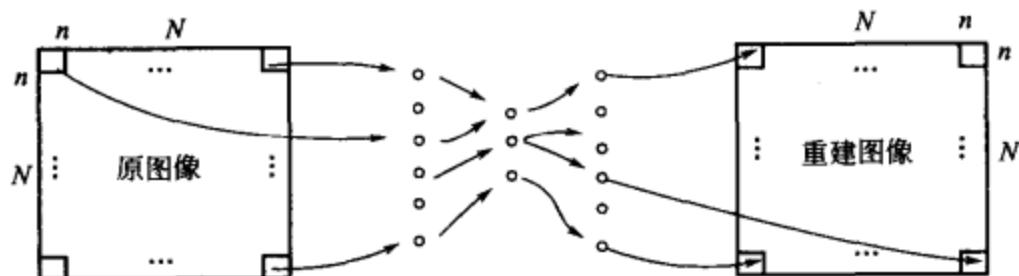


图 3.30 用于图像压缩编码的 BP 网络

3.5.4 BP 网络用于水库优化调度

对水库进行优化调度的目的是有效地利用水资源。其中一个重要问题是建立水库调度模型。常规的方法是选用广义线性函数作为调度函数，但由于选择基函数和求解系数方面的困难，求得的调度函数难以表达水库调度决策变量及其影响因子之间固有的复杂非线性关系。多层前馈神经网络作为调度函数，可以克服常规方法的缺陷，具有良好的应用前景。

水库群调度属于优化问题，其约束条件为时段发电量约束、保证供水量约束、水量平衡约束、渠道输水能力约束、变量可行域约束和弃水约束等。优化的目标函数是调度期供水量最大。

水库群选北方某严重缺水地区的 3 个并联供水水库为实例。训练样本集的数据来自 1919~1984 年共 66 年的实测径流资料，将每年划分为 18 个时段，对应于每个时段设计一个神经网络，共 18 个网络，各获得 66 个训练样本。每个网络的输入层有 18 个神经元，分

别对应于本时段和前两个时段 3 个水库的入库流量和库存水量。输出层有 3 个神经元，对应于 3 个水库的供水量。

训练后的网络作为水库群调度函数模型应用于该水库群 1985~1990 年联合调度模拟运行，较准确地反映出调度函数中因变量和自变量之间的非线性关系。表 3.4 列出 BP 网络模型与关联平衡（IBM）法所得结果的对比。

表 3.4 IBM 法和人工神经网络法（ANN 法）时段供水量对比

时段	A 库供水量		B 库供水量		C 库供水量	
	IBM 法	ANN 法	IBM 法	ANN 法	IBM 法	ANN 法
1	0.674	0.651	0.763	0.748	0.275	0.290
2	0.821	0.813	0.541	0.545	0.280	0.293
3	1.327	1.307	0.872	0.865	0.284	0.280
4	0.992	0.971	1.406	1.309	0.776	0.790
5	1.420	1.320	1.112	1.137	0.721	0.712
...
17	0.476	0.473	0.573	0.554	0.395	0.383
18	0.621	0.639	0.687	0.665	0.417	0.392

3.5.5 BP 网络用于证券预测

在证券预测研究中，常采用基本面分析和技术面分析两种方法。基本面分析是一种宏观分析法，重点考察影响证券走势的宏观因素、产业因素、市场因素及企业因素等最基本的因素；技术面分析是一种微观分析法，它使用技术手段分析证券市场交易数量和价格走势，从而预测证券市场行情变动趋势。传统的技术面分析方法多采用图表与指标作为工具，依靠主观判断及经验理论，很大程度降低了证券交易的可靠性。由于神经网络能够自动抽取数据集合中的非线性关系并进行模拟，常用于分析类似于股价预测等多因素、不确定、非线性的时序数据。

下面简要介绍用 BP 算法预测证券指数短期变动趋势的一则实例：

(1) 预测模型的设计 选取前日收盘价、前日成交量、昨日收盘价、昨日成交量、本日收盘价、本日成交量及与系统状态有关的两个指数 RSI 和 DMA 共 8 个参数作为网络的输入，明日收盘价作为网络的输出。采用 3 层感知器结构，经试验确定隐节点数为 4，网络结构为 8-4-1。初始权值利用随机函数生成。动量因子是 BP 算法改进后增加的参数，其值在(0, 1) 范围内取偏小值。转移函数中增加了调整参数 r ：

$$f(x, r) = \frac{1 - e^{-2rx}}{1 + e^{-2rx}}$$

r 在每次运算时按以下规则进行调整：

$$\Delta r = \eta \frac{\delta x}{r}$$

(2) 预测模型实证分析 股票指数是股票市场总体的反映，对某年 4 月 11 日~4 月 17 日的上证指数进行预测。基于 BP 算法的预测模型预测结果与其他算法的预测结果如表 3.5 所示。BP 算法的预测结果比回归预测提高了 0.39%，比指数预测提高了 0.29%，比灰色预测提高了 0.13%。

表 3.5 BP 算法与其他算法的证券预测结果

	实际值	预测值			
		BP 网络	回归预测	指数预测	灰色预测
日期	4.11	1649.53	1648.736	1622.38	1635.24
	4.12	1658.98	1654.430	1635.47	1647.55
	4.15	1649.50	1661.290	1673.25	1665.55
	4.16	1640.29	1657.046	1677.63	1671.63
	4.17	1644.40	1634.778	1679.83	1660.25
平均相对误差/%		0.87	1.34	1.12	1.06
相对误差的标准差/%		0.67	0.95	0.92	0.87
最大相对误差/%		1.03	4.66	3.57	2.85

3.5.6 BP 网络用于信用评价模型及预警

利用 BP 网络建立神经网络信用评价模型，用来对我国 2000 年 106 家上市公司进行信用评级，并进一步对我国 2001 年公布的 13 家预亏公司进行预警研究。仿真结果表明，神经网络信用评价模型有很高的分类准确率，且有很强的适应能力，因而可以进一步用来对企业的财务危机进行预警研究。信用评价网络模型设计与应用效果如下：

(1) 评价模型的设计 按照各上市公司的经营状况分为“好”和“差”两类，每一类由 53 家上市公司构成数据样本。对于每家上市公司，考虑反映其经营状况的主要 4 个财务指标：每股收益、每股净资产、净资产收益率和每股现金流量。因此网络输入层设 4 个节点分别接收 4 个财务指标，输出层设 1 个节点以 1 或 0 表示经营状况评为“好”或“差”。定义第一类错误为将经营“差”的企业误判为经营“好”的企业，第二类错误为将经营“好”的企业误判为经营“差”的企业。由表 3.6 知，当隐层节点的个数为 4 时，误判率最低，故评价模型的网络结构为 4-4-1。在 106 个样本中，选择 32 家亏损企业和 31 家不亏损企业作为“好”和“差”两类的训练样本，其余 43 个样本作为测试样本。

表 3.6 训练样本的误判率

隐层节点数	训练样本集			
	第一类错误/个	第二类错误/个	总误判/个	误判率/%
1	32	0	32	50.79
2	9	0	9	14.29
3	0	5	5	7.94
4	0	1	1	1.59
5	0	17	17	26.98
6	10	0	10	15.87
7	0	4	4	6.35
8	11	0	11	19.05

(2) 评价模型的分类实证 对 43 个测试样本的第一类误判数为 1，第二类误判数为 0，因此测试样本集的分类准确率达到 97.67%。总体 106 个样本的误判数为 2，故该神经网络信用评价模型分类的准确率达到 98.11%。

(3) 评价模型的预警实证 利用所建立的 BP 网络信用评价模型对我国 2001 年公布的 13 家预亏企业进行预警实证分析。将 13 家企业的 4 项财务指标输入信用评价模型，其输出结果表明总误判率为 0，预警准确率达到 100%。

本章小结

本章介绍了两种基于监督学习的前馈型神经网络：由线性阈值单元组成的单层感知器和由非线性单元组成的多层感知器以及误差反向传播算法。采用 BP 算法的多层感知器简称 BP 网络。学习重点如下：

(1) 感知器 单层感知器只能解决线性可分的分类问题，多层感知器则可解决线性不可分的分类问题。感知器的每个隐节点可构成一个线性分类判决界，多个节点构成样本空间的凸域，输出节点可将凸域内外的样本分类。

(2) 标准 BP 算法 BP 算法的实质是把一组输入输出问题转化为非线性映射问题，并通过梯度下降算法迭代求解权值。BP 算法分为净输入前向计算和误差反向传播两个过程。网络训练时，两个过程交替出现直到网络的总误差达到预设精度。网络工作时各权值不再变化，对每一给定输入，网络通过前向计算给出输出响应。

(3) 改进的 BP 算法 针对标准 BP 算法存在的缺陷已提出许多改进算法。本章介绍了增加动量项法、变学习率法和引入陡度因子法。应用 BP 网络解决实际问题时，应尽量采用较成熟的改进算法。

(4) 采用 BP 算法的多层前馈网络的设计 神经网络的设计涉及训练样本集设计、网络结构设计和训练与测试三个方面。训练样本集设计包括原始数据的收集整理、数据分析、变量选择、特征提取及数据预处理等多方面的工作。网络结构设计包括隐层数和隐层节点数的选择、初始权值（阈值）的选择等，由于缺乏理论指导，主要靠经验和试凑。训练与测试交替进行可找到一个最佳训练次数，以保证网络具有较好的泛化能力。

思考与练习

3.1 考虑下面的分类问题：

$$\begin{aligned} \left\{ \mathbf{X}^1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, d^1 = 1 \right\} & \quad \left\{ \mathbf{X}^2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, d^2 = 1 \right\} & \quad \left\{ \mathbf{X}^3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, d^3 = 1 \right\} \\ \left\{ \mathbf{X}^4 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, d^4 = 0 \right\} & \quad \left\{ \mathbf{X}^5 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, d^5 = 0 \right\} \end{aligned}$$

- ① 画出能求解此问题的单节点感知器结构图。
- ② 画出输入数据点的分布图，并根据目标值对其进行标记。
- ③ 给分类问题能否用单节点感知器求解。

3.2 请画出图 3.31 中三个简单分类问题的分类判决界，并求出相应的权值和阈值。

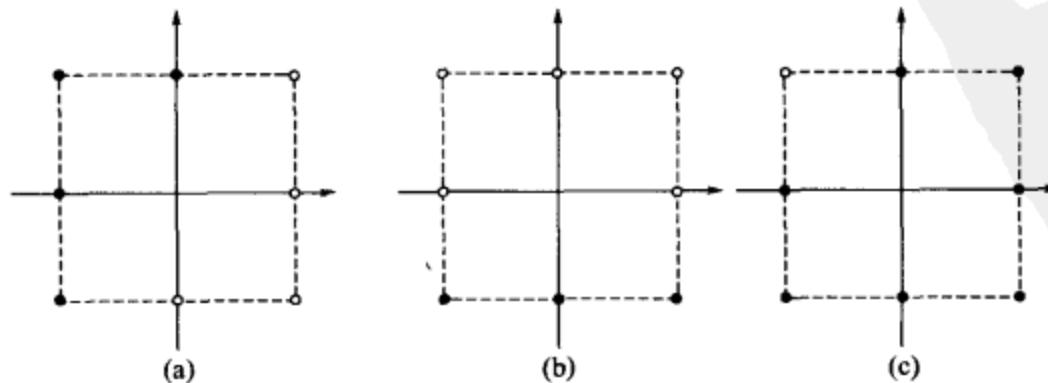


图 3.31 习题 3.2 附图

3.3 用感知器学习规则训练一分类器，算法中 $\eta=1$ ，初始权值 $\mathbf{W}=0$ ，写出训练后的权值和阈值。训练样本如下：

$$\text{1类 } \mathbf{X}^1 = (0.8, 0.5, 0)^T, \mathbf{X}^2 = (0.9, 0.7, 0.3)^T, \mathbf{X}^3 = (1, 0.8, 0.5)^T$$

$$\text{2类 } \mathbf{X}^4 = (0, 0.2, 0.3)^T, \mathbf{X}^5 = (0.2, 0.1, 1.3)^T, \mathbf{X}^6 = (0.2, 0.7, 0.8)^T$$

3.4 用感知器学习规则对例 3.1 中的感知器进行训练，并考察该感知器对 8 个样本的分类情况。

3.5 已知以下样本分属于两类：

$$\text{1类 } \mathbf{X}^1 = (5, 1)^T, \mathbf{X}^2 = (7, 3)^T, \mathbf{X}^3 = (3, 2)^T, \mathbf{X}^4 = (5, 4)^T$$

$$\text{2类 } \mathbf{X}^5 = (0, 0)^T, \mathbf{X}^6 = (-1, -3)^T, \mathbf{X}^7 = (-2, 3)^T, \mathbf{X}^8 = (-3, 0)^T$$

① 判断两类样本是否线性可分。

② 试确定一直线，并使该直线与两类样本中心连线相垂直。

③ 设计一单节点感知器，如用上述直线方程作为其分类判决方程 $\text{net}=0$ ，写出感知器的权值与阈值。

④ 用上述感知器对以下 3 个样本进行分类：

$$\mathbf{X} = (4, 2)^T, \quad \mathbf{X} = (0, 5)^T, \quad \mathbf{X} = \left(\frac{36}{13}, 0 \right)^T$$

3.6 将下面定义的问题转换为由一组不等式约束的权值和阈值所定义的等价问题：

$$\left\{ \mathbf{X}^1 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, d^1 = 1 \right\} \left\{ \mathbf{X}^2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, d^2 = 1 \right\} \left\{ \mathbf{X}^3 = \begin{bmatrix} 0 \\ -2 \end{bmatrix}, d^3 = 0 \right\} \left\{ \mathbf{X}^4 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, d^4 = 0 \right\}$$

3.7 试设计一个单隐层感知器，将图 3.32 所示的三角形凸域内外的样本分开。设凸域内样本的期望输出为 1，凸域外样本的期望输出为 0。

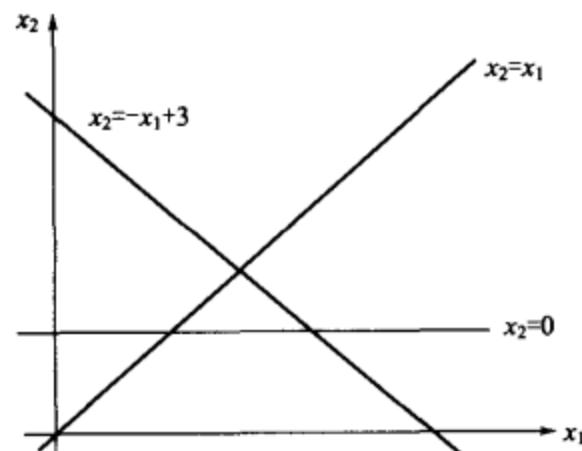


图 3.32 习题 3.7 附图

3.8 试利用感知器学习规则训练本章例 3.2 中的感知器，使其能对 4 类样本正确分类。训练集为：

$$\text{第一类: } \mathbf{X}^1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{X}^2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad d^1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad d^2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\text{第二类: } \mathbf{X}^3 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \quad \mathbf{X}^4 = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \quad d^3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad d^4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\text{第三类: } \mathbf{X}^5 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, \quad \mathbf{X}^6 = \begin{bmatrix} -2 \\ 1 \end{bmatrix} \quad d^5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad d^6 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\text{第四类: } \mathbf{X}^7 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad \mathbf{X}^8 = \begin{bmatrix} -2 \\ -2 \end{bmatrix} \quad d^7 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad d^8 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

初始权值和阈值分别为：

$$\mathbf{W}(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{T}(0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

训练结束后请将最终的分类判决边界画在输入平面上，并与图 3.9 中的结果进行比较。

3.9 下面给出的训练集由玩具兔和玩具熊组成。输入样本向量的第一个分量代表玩具的重量，第二个分量代表玩具耳朵的长度，教师信号为 -1 表示玩具兔，教师信号为 1 表示玩具熊。

$$\left\{ \mathbf{X}^1 = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, d^1 = -1 \right\} \left\{ \mathbf{X}^2 = \begin{bmatrix} 1 \\ 5 \end{bmatrix}, d^2 = -1 \right\} \left\{ \mathbf{X}^3 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, d^3 = -1 \right\} \left\{ \mathbf{X}^4 = \begin{bmatrix} 2 \\ 5 \end{bmatrix}, d^4 = -1 \right\}$$

$$\left\{ \mathbf{X}^5 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, d^5 = 1 \right\} \quad \left\{ \mathbf{X}^6 = \begin{bmatrix} 3 \\ 2 \end{bmatrix}, d^6 = 1 \right\} \quad \left\{ \mathbf{X}^7 = \begin{bmatrix} 4 \\ 1 \end{bmatrix}, d^7 = 1 \right\} \quad \left\{ \mathbf{X}^8 = \begin{bmatrix} 4 \\ 2 \end{bmatrix}, d^8 = 1 \right\}$$

① 用 MATLAB 训练一个感知器，求解此分类问题。

② 用输入样本对所训练的感知器进行验证。

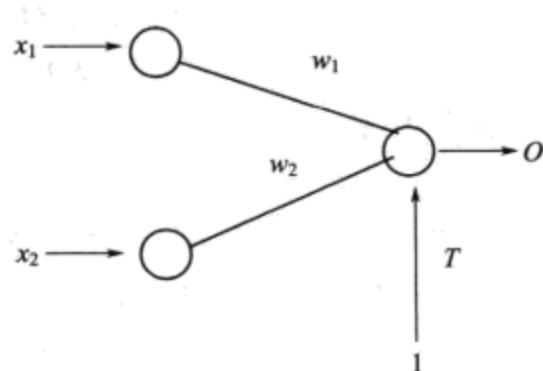


图 3.33 习题 3.10 附图

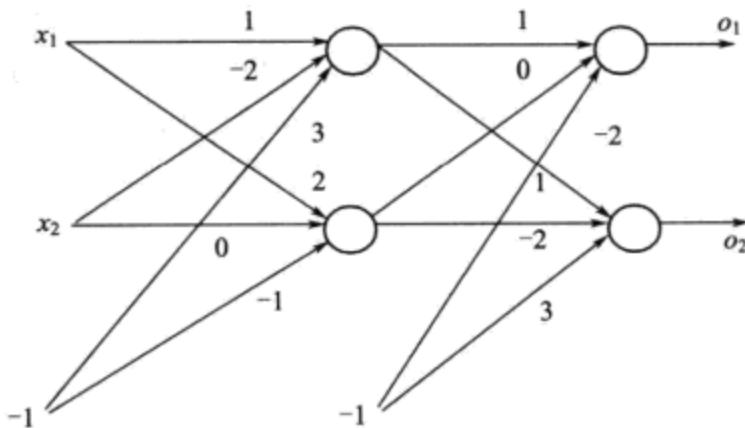


图 3.34 习题 3.14 附图

3.10 单节点感知器如图 3.33 所示, $w_1=1$, $w_2=-1$, $T=0$ 。考虑模为 1 的分布在单位圆周的输入向量 $\mathbf{X}=(x_1, x_2)^T$ 。

① 哪些输出向量与权向量得到正内积? 哪些输出向量与权向量得到负内积?

② 从上述结果可以看出, 该单节点感知器确定的分界线将输入平面对分为两半: 一半对应于正内积样本; 另一半对应于负内积样本。请画出对应于零内积的输入样本的轨迹。

③ 如果 $T=0.5$, 结果将如何改变?

3.11 BP 网络有哪些长处与缺陷, 试各列举出三条。

3.12 什么是 BP 网络的泛化能力? 如何保证 BP 网络具有较好的泛化能力?

3.13 BP 网络擅长解决哪些问题? 试举几例。

3.14 BP 网络结构如图 3.34 所示, 初始权值已标在图中。网络的输入模式为 $\mathbf{X}=(-1, 1, 3)^T$, 期望输出为 $\mathbf{d}=(0.95, 0.05)^T$ 。试对单次训练过程进行分析, 求出:

- ① 隐层权值矩阵 \mathbf{V} 和输出层权值矩阵 \mathbf{W} ;
- ② 各层净输入和输出: net^y 、 \mathbf{Y} 和 net^o 、 \mathbf{O} , 其中上标 y 代表隐层, o 代表输出层;
- ③ 各层输出的一阶导数 $f'(net^y)$ 和 $f'(net^o)$;
- ④ 各层误差信号 δ^o 和 δ^y ;
- ⑤ 各层权值调整量 $\Delta\mathbf{V}$ 和 $\Delta\mathbf{W}$;
- ⑥ 调整后的权值矩阵 \mathbf{V} 和 \mathbf{W} 。

3.15 根据图 3.19 给出的流程图上机编程实现三层前馈神经网络的 BP 学习算法。要求程序具有以下功能:

- ① 允许选择各层节点数;
- ② 允许选用不同的学习率 η ;
- ③ 能对权值进行初始化, 初始化用 $[-1, 1]$ 区间的随机数;
- ④ 允许选用单极性或双极性两种不同 Sigmoid 型转移函数。

程序调试通过后, 可用以下各题提供的数据进行训练。

3.16 设计一个神经网络字符分类器对图 3.35 中的英文字母进行分类。输入向量含一

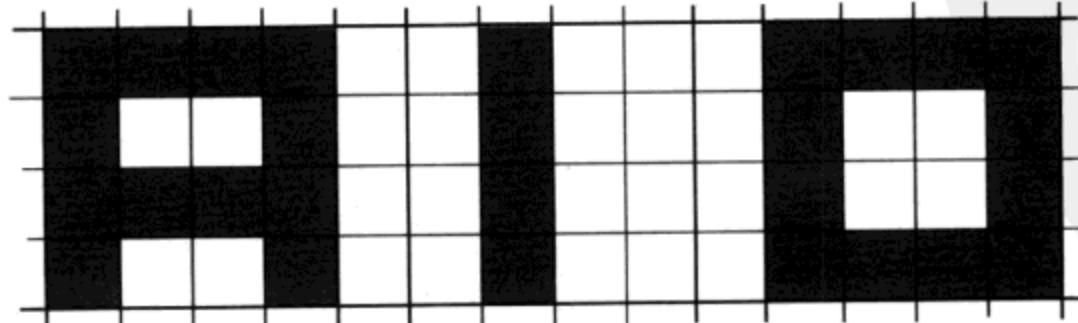


图 3.35 习题 3.16 附图

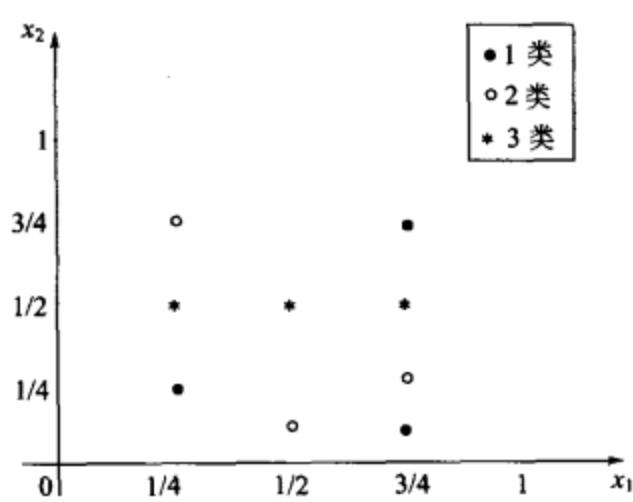


图 3.36 习题 3.17 附图

个分量，输出向量分别用 $(1, -1, -1)^T$ 、 $(-1, 1, -1)^T$ 和 $(-1, -1, 1)^T$ 代表字符 **A**、**I** 和 **O**。试用标准 BP 学习算法训练网络，训练时可选择不同的隐节点数及不同的学习率，对达到同一训练误差的训练次数进行对比。

3.17 设计一个神经网络对图 3.36 中的三类线性不可分模式进行分类。期望输出向量分别用 $(1, -1, -1)^T$ 、 $(-1, 1, -1)^T$ 、 $(-1, -1, 1)^T$ 代表三类，输入用样本坐标。要求：

- ① 选择合适的隐节点数；
- ② 用 BP 算法训练网络对图中 9 个样本进行正确分类。

3.18 图 3.37 所示神经网络的功能是逼近某单变量 t 的连续函数。该网络使用双极性 S 型函数，具有 10 个隐节点和 1 个输出节点。训练后网络权值矩阵如下：

$$W^T = (-1.35 \quad 0.14 \quad 4.26 \quad 1.18 \quad -1.02 \quad 1.20 \quad 0.55 \quad 1.33 \quad -1.27 \quad -1.20 \quad 0.45)$$

$$V = \begin{pmatrix} 1.12 & 2.46 & 6.11 & -1.08 & 0.96 & -1.03 & -0.58 & -1.11 & 1.13 & 1.05 \\ 0.36 & 0.27 & 0.09 & 0.28 & 0.24 & -0.29 & 0.12 & -0.34 & 0.05 & 0.06 \end{pmatrix}$$

试用 $-1 \leq t \leq 1$ 范围的输入数据测试该网络，指出其映射的是何函数关系。

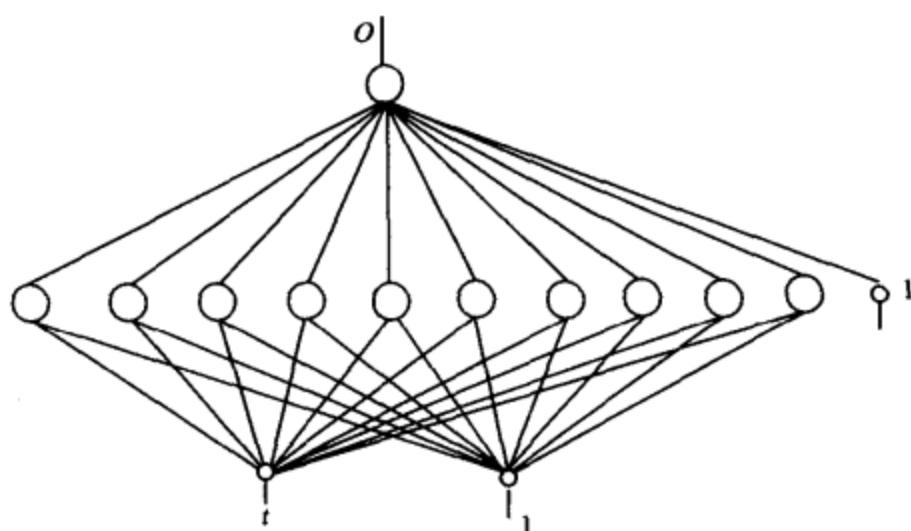


图 3.37 习题 3.18 附图

4 竞争学习神经网络

采用有导师学习规则的神经网络要求对所学习的样本给出“正确答案”，以便网络据此判断输出的误差，根据误差的大小改进自身的权值，提高正确解决问题的能力。然而在很多情况下，人在认知过程中没有预知的正确模式，人获得大量知识常常是靠“无师自通”，即通过对客观事物的反复观察、分析与比较，自行揭示其内在规律，并对具有共同特征的事物进行正确归类。对于人的这种学习方式，基于有导师学习策略的神经网络是无能为力的。自组织神经网络的无导师学习方式更类似于人类大脑中生物神经网络的学习，其最重要的特点是通过自动寻找样本中的内在规律和本质属性，自组织、自适应地改变网络参数与结构。这种学习方式大大拓宽了神经网络在模式识别与分类方面的应用。

自组织网络结构上属于层次型网络，有多种类型，其共同特点是都具有竞争层。最简单的网络结构具有一个输入层和一个竞争层，如图 4.1 所示。输入层负责接受外界信息并将输入模式向竞争层传递，起“观察”作用，竞争层负责对该模式进行“分析比较”，找出规律以正确归类。这种功能是通过下面要介绍的竞争机制实现的。

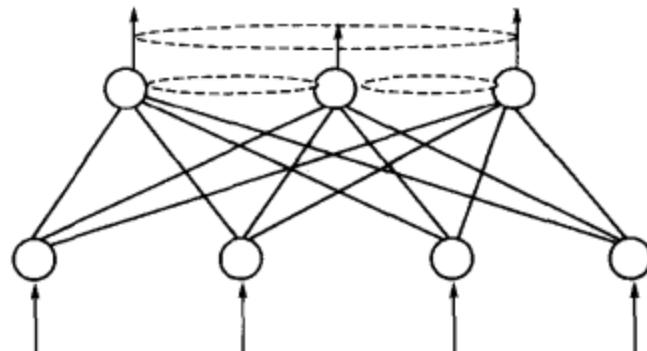


图 4.1 自组织网络的典型结构

4.1 竞争学习的概念与原理

竞争学习是自组织网络中最常采用的一种学习策略。为使后面的叙述清楚明了，首先说明与之相关的几个基本概念。

4.1.1 基本概念

4.1.1.1 模式、分类、聚类与相似性

在神经网络应用中，输入样本、输入模式和输入模式样本这类术语经常混用。一般当神经网络涉及识别、分类问题时，常常用到输入模式的概念。模式是对某些感兴趣的客体的定量描述或结构描述，模式类是具有某些共同特征的模式的集合。分类是在类别知识等导师信号的指导下，将待识别的输入模式分配到各自的模式类中去。无导师指导的分类称为聚类，聚类的目的是将相似的模式样本划归一类，而将不相似的分离开，其结果实现了模式样本的类内相似性和类间分离性。通过聚类，可以发现原始样本的分布与特性。

由于无导师学习的训练样本中不含有期望输出，因此对于某一输入模式样本应属于哪一类并没有任何先验知识。对于一组输入模式，只能根据它们之间的相似程度分为若干类，因

此相似性是输入模式的聚类依据。关于聚类分析的研究，需要解决的问题是：如何决定相似度；如何决定聚类的类别数；如何决定哪种分类的结果是理想的。

4.1.1.2 相似性测量

神经网络的输入模式用向量表示，比较两个不同模式的相似性可转化为比较两个向量的距离，因而可用模式向量间的距离作为聚类判据。传统模式识别中常用到的两种聚类判据是欧式距离法和余弦法。下面分别予以介绍：

(1) 欧式距离法 为了描述两个输入模式的相似性，常用的方法是计算其欧式距离，即：

$$\|\mathbf{X} - \mathbf{X}_i\| = \sqrt{(\mathbf{X} - \mathbf{X}_i)^T (\mathbf{X} - \mathbf{X}_i)} \quad (4.1)$$

两个模式向量的欧式距离越小，两个向量越接近，因此认为这两个模式越相似，当两个模式完全相同时其欧式距离为零。如果对同一类内各个模式向量间的欧式距离作出规定，不允许超过某一最大值 T ，则最大欧式距离 T 就成为一种聚类判据。从图 4.2(a) 可以看出，同类模式向量的距离小于 T ，两类模式向量的距离大于 T 。

(2) 余弦法 描述两个模式向量的另一个常用方法是计算其夹角的余弦，即：

$$\cos\psi = \frac{\mathbf{X}^T \mathbf{X}_i}{\|\mathbf{X}\| \|\mathbf{X}_i\|} \quad (4.2)$$

从图 4.2(b) 可以看出，两个模式向量越接近，其夹角越小，余弦越大。当两个模式方向完全相同时，其夹角余弦为 1。如果对同一类内各个模式向量间的夹角作出规定，不允许超过某一最大角 ψ_T ，则最大夹角 ψ_T 就成为一种聚类判据。同类模式向量的夹角小于 ψ_T ，两类模式向量的夹角大于 ψ_T 。余弦法适合模式向量长度相同或模式特征只与向量方向相关的相似性测量。

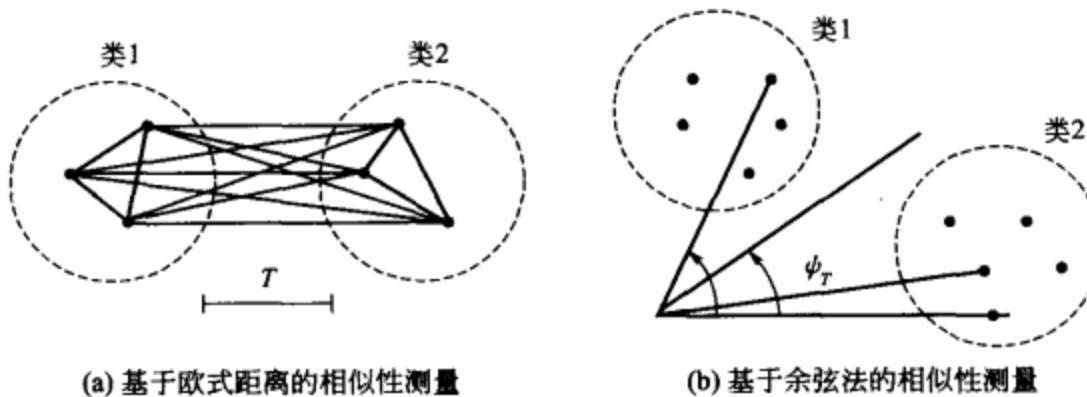


图 4.2 聚类的相似性测量

(3) 内积法 描述两个模式向量的第三种常用方法是计算其内积，即：

$$\mathbf{X}^T \mathbf{X}_i = \|\mathbf{X}\| \|\mathbf{X}_i\| \cos\psi$$

内积值越大则相似度越高，当两个模式方向完全相同且长度相等时，其相似度取最大值。

不同的相似度会导致所形成之聚类几何特性不同。如图 4.3 所示，若用欧式距离法度量相似度，会形成大小相似且紧密的圆形聚类；若用余弦法度量相似度，将形成大体同向的狭长形聚类。倘若用内积法度量相似度，则不一定会形成大体同向的狭长形聚类，因为即使两个向量角度几乎相同，但其长度有很大差别，其内积值仍会有较大差异。

4.1.1.3 侧抑制与竞争

实验表明，在人眼的视网膜、脊髓和海马中存在一种侧抑制现象，即当一个神经细

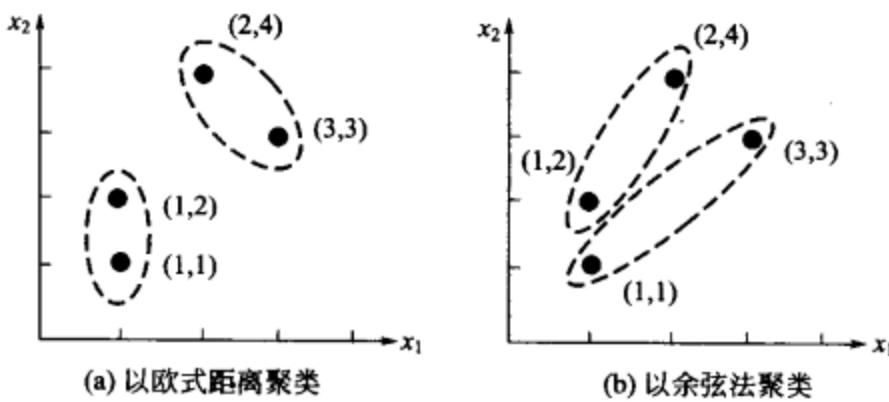


图 4.3 不同相似性测量的聚类结果

胞兴奋后，会对其周围的神经细胞产生抑制作用。这种侧抑制使神经细胞之间呈现出竞争，开始时可能多个细胞同时兴奋，但一个兴奋程度最强的神经细胞对周围神经细胞的抑制作用也越强，其结果使其周围神经细胞兴奋度减弱，从而该神经细胞是这次竞争的“胜者”，而其他神经细胞在竞争中失败。为了表现这种侧抑制，图 4.1 所示的网络在竞争层各神经元之间加了许多虚线连接线，它们是模拟生物神经网络层内神经元相互抑制现象的权值。这类抑制性权值常满足一定的分布关系，如距离近的抑制强，距离远的抑制弱。由于权值一般是固定的，训练过程中不需要调整，在各类自组织网络拓扑图中一般予以省略。最强的抑制作用是竞争获胜者“唯我独兴”，不允许其他神经元兴奋，这种做法称为胜者为王。

4.1.1.4 向量归一化

不同的向量有长短和方向的区别，向量归一化的目的是将向量变成方向不变长度为 1 的单位向量。二维和三维单位向量可以在单位圆和单位球上直观表示。单位向量进行比较时，只需比较向量的夹角。向量归一化按下式进行：

$$\hat{\mathbf{X}} = \frac{\mathbf{X}}{\|\mathbf{X}\|} = \left[\frac{x_1}{\sqrt{\sum_{j=1}^n x_j^2}} \quad \dots \quad \frac{x_n}{\sqrt{\sum_{j=1}^n x_j^2}} \right]^T \quad (4.3)$$

式中，归一化后的向量用“ $\hat{\cdot}$ ”标记。

4.1.2 竞争学习原理

竞争学习采用的规则是胜者为王，第 2 章曾作过简单介绍，下面结合图 4.1 的网络结构和竞争学习的思想进一步学习该规则。

4.1.2.1 竞争学习规则

在竞争学习策略中采用的典型学习规则称为胜者为王（Winner-Take-All）。该算法可分为 3 个步骤：

(1) 向量归一化 首先将自组织网络中的当前输入模式向量 \mathbf{X} 和竞争层中各神经元对应的内星向量 $\mathbf{W}_j (j=1, 2, \dots, m)$ 全部进行归一化处理，得到 $\hat{\mathbf{X}}$ 和 $\hat{\mathbf{W}}_j (j=1, 2, \dots, m)$ 。

(2) 寻找获胜神经元 当网络得到一个输入模式向量 $\hat{\mathbf{X}}$ 时，竞争层的所有神经元对应的内星权向量 $\hat{\mathbf{W}}_j (j=1, 2, \dots, m)$ 均与 $\hat{\mathbf{X}}$ 进行相似性比较，将与 $\hat{\mathbf{X}}$ 最相似的内星权向量判为竞争获胜神经元，其权向量记为 $\hat{\mathbf{W}}_j^*$ 。测量相似性的方法是对 $\hat{\mathbf{W}}_j$ 和 $\hat{\mathbf{X}}$ 计算欧式距离（或夹角余弦）：

$$\|\hat{\mathbf{X}} - \hat{\mathbf{W}}_j^*\| = \min_{j \in \{1, 2, \dots, m\}} \{\|\hat{\mathbf{X}} - \hat{\mathbf{W}}_j\|\} \quad (4.4)$$

将上式展开并利用单位向量的特点，可得：

$$\begin{aligned}\|\hat{\mathbf{X}} - \hat{\mathbf{W}}_j^*\| &= \sqrt{(\hat{\mathbf{X}} - \hat{\mathbf{W}}_j^*)^T (\hat{\mathbf{X}} - \hat{\mathbf{W}}_j^*)} = \sqrt{\hat{\mathbf{X}}^T \hat{\mathbf{X}} - 2\hat{\mathbf{W}}_j^{T*} \hat{\mathbf{X}} + \hat{\mathbf{W}}_j^{T*} \hat{\mathbf{W}}_j^*} \\ &= \sqrt{2(1 - \hat{\mathbf{W}}_j^{T*} \hat{\mathbf{X}})}\end{aligned}$$

从上式可以看出，欲使两单位向量的欧式距离最小，须使两向量的点积最大。即：

$$\hat{\mathbf{W}}_j^{T*} \hat{\mathbf{X}} = \max_{j \in \{1, 2, \dots, m\}} (\hat{\mathbf{W}}_j^T \hat{\mathbf{X}}) \quad (4.5)$$

于是按式(4.4)求最小欧式距离的问题就转化为按式(4.5)求最大点积的问题，而权向量与输入向量的点积正是竞争层神经元的净输入。

(3) 网络输出与权值调整 胜者为王竞争学习算法规定，获胜神经元输出为 1，其余输出为零。即：

$$o_j(t+1) = \begin{cases} 1, & j = j^* \\ 0, & j \neq j^* \end{cases} \quad (4.6)$$

只有获胜神经元才有权调整其权向量 \mathbf{W}_j^* ，调整量后权向量为：

$$\begin{cases} \mathbf{W}_j^*(t+1) = \hat{\mathbf{W}}_j^*(t) + \Delta \mathbf{W}_j^* = \hat{\mathbf{W}}_j^*(t) + \alpha(\hat{\mathbf{X}} - \hat{\mathbf{W}}_j^*), & j = j^* \\ \mathbf{W}_j(t+1) = \hat{\mathbf{W}}_j(t), & j \neq j^* \end{cases} \quad (4.7)$$

式中， $\alpha \in (0, 1]$ 为学习率，一般其值随着学习的进展而减小。可以看出，当 $i \neq j^*$ 时，对应神经元的权值得不到调整，其实质是“胜者”对它们进行了强侧抑制，不允许它们兴奋。

应当指出，归一化后的权向量经过调整后得到的新向量不再是单位向量，因此需要对调整后的向量重新归一化。步骤(3)完成后回到步骤(1)继续训练，直到学习率 α 衰减到 0。

4.1.2.2 竞争学习原理

设输入模式为 2 维向量，归一化后其矢端可以看成分布在图 4.4 单位圆上的点，用“○”表示。设竞争层有 4 个神经元，对应的 4 个内星向量归一化后也标在同一单位圆上，用“*”表示。从输入模式点的分布可以看出，它们大体上聚集为 4 簇，因而可以分为 4 类。然而自组织网络的训练样本中只提供了输入模式而没有提供关于分类的指导信息，网络是如何通过竞争机制自动发现样本空间的类别划分呢？

自组织网络在开始训练前先对竞争层的权向量进行随机初始化。因此在初始状态时，单位圆上的*是随机分布的。前面已经证明，两个等长向量的点积越大，两者越近似，因此以点积最大获胜的神经元对应的权向量应最接近当前输入模式。从图 4.4 可以看出，如果当前输入模式用空心圆○表示，单位圆上各*点代表的权向量依次同○点代表的输入向量比较距离，结果是离得最近的那个*点获胜。从获胜神经元的权值调整式可以看出，调整的结果是使 \mathbf{W}_m 进一步接近当前输入 \mathbf{X} 。这一点从图 4.5 的向量合成图上可以看得很清楚。调整后，获胜*点的位置进一步移向○点及其所在的簇。显然，当下次出现与○点相像的同簇内的输入模式时，上次获胜的*点更容易获胜。依此方式经过充分训练后，单位圆上的 4 个*点会逐渐移入各输入模式的簇中心，从而使竞争层每个神经元的权向量成为输入模式一个聚类中心。当向网络输入一个模式时，竞争层中哪个神经元获胜使输出为 1，当前输入模式就归为哪类。

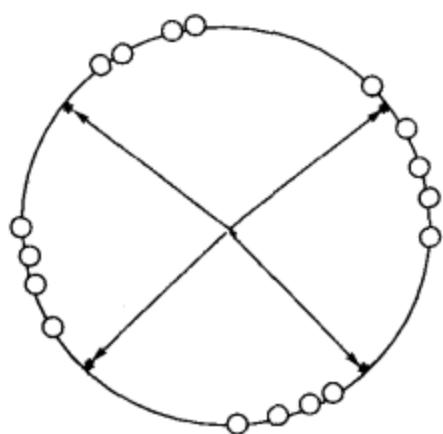


图 4.4 竞争学习的几何意义

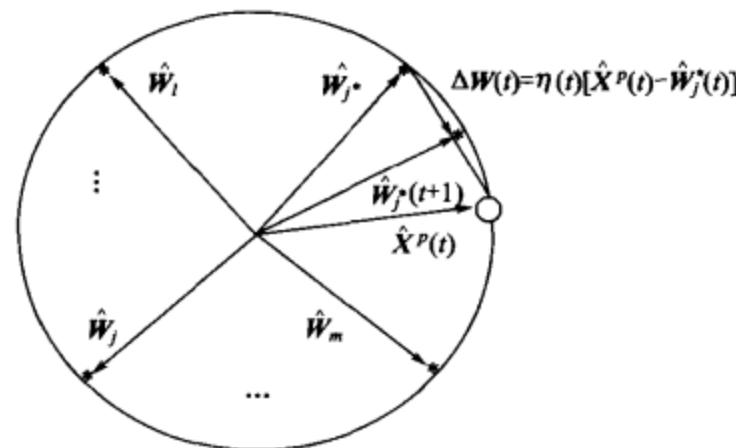


图 4.5 自组织权向量调整

例 4.1 用竞争学习算法将下列各模式分为两类:

$$\mathbf{X}^1 = \begin{pmatrix} 0.8 \\ 0.6 \end{pmatrix} \quad \mathbf{X}^2 = \begin{pmatrix} 0.1736 \\ -0.9848 \end{pmatrix} \quad \mathbf{X}^3 = \begin{pmatrix} 0.707 \\ 0.707 \end{pmatrix} \quad \mathbf{X}^4 = \begin{pmatrix} 0.342 \\ -0.9397 \end{pmatrix} \quad \mathbf{X}^5 = \begin{pmatrix} 0.6 \\ 0.8 \end{pmatrix}$$

解 为作图方便, 将上述模式转换成极坐标形式:

$$\mathbf{X}^1 = 1\angle 36.89^\circ \quad \mathbf{X}^2 = 1\angle -80^\circ \quad \mathbf{X}^3 = 1\angle 445^\circ \quad \mathbf{X}^4 = 1\angle -70^\circ \quad \mathbf{X}^5 = 1\angle 53.13^\circ$$

竞争层设两个权向量, 随机初始化为单位向量:

$$\mathbf{W}_1(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 1\angle 0^\circ \quad \mathbf{W}_2(0) = \begin{pmatrix} -1 \\ 0 \end{pmatrix} = 1\angle 180^\circ$$

取学习率 $\eta=0.5$, 按 1~5 的顺序依次输入模式向量, 用式(4.7)给出的算法调整权值, 每次修改后重新进行归一化。前 20 次训练中两个权向量的变化情况列于表 4.1 中。

表 4.1 权向量调整过程

训练次数	\mathbf{W}_1	\mathbf{W}_2	训练次数	\mathbf{W}_1	\mathbf{W}_2
1	18.43°	-180°	11	40.5°	-100°
2	-30.8°	-180°	12	40.5°	-90°
3	7°	-180°	13	43°	-90°
4	-32°	-180°	14	43°	-81°
5	11°	-180°	15	47.5°	-81°
6	24°	-180°	16	42°	-81°
7	24°	-130°	17	42°	-80.5°
8	34°	-130°	18	43.5°	-80.5°
9	34°	-100°	19	43.5°	-75°
10	44°	-100°	20	48.5°	-75°

如将 5 个输入模式和 2 个权向量标在单位圆中, 可以明显看出, \mathbf{X}^1 、 \mathbf{X}^3 、 \mathbf{X}^5 属于同一模式类, 其中心向量应为 $1/3(\mathbf{X}^1 + \mathbf{X}^3 + \mathbf{X}^5) = 1\angle 45^\circ$; \mathbf{X}^2 、 \mathbf{X}^4 属于同一模式类, 其中心向量应为 $1/2(\mathbf{X}^2 + \mathbf{X}^4) = 1\angle -75^\circ$ 。经过 20 次训练, \mathbf{W}_1 和 \mathbf{W}_2 就已经非常接近 $1\angle 45^\circ$ 和 $1\angle -75^\circ$ 了。如果训练继续下去, 两个权向量是否会最终收敛于两个模式类中心呢? 事实上, 如果训练中学习率保持为常数, \mathbf{W}_1 和 \mathbf{W}_2 将在 $1\angle 45^\circ$ 和 $1\angle -75^\circ$ 附近来回摆动, 永远也不可能收敛。只有当学习率随训练时间不断下降, 才有可能使摆动减弱至终止。下面将要介绍的自组织特征映射网就是采取了这种训练方法。

4.2 自组织特征映射神经网络

1981 年芬兰 Helsinki 大学的 T. Kohonen 教授提出一种自组织特征映射网 (self-organizing feature map, 简称 SOM)，又称 Kohonen 网。Kohonen 认为，一个神经网络接受外界输入模式时，将会分为不同的对应区域，各区域对输入模式具有不同的响应特征，而且这个过程是自动完成的。自组织特征映射正是根据这一看法提出来的，其特点与人脑的自组织特性相类似。

4.2.1 SOM 网的生物学基础

生物学研究的事实表明，在人脑的感觉通道上，神经元的组织原理是有序排列。因此当人脑通过感官接受外界的特定时空信息时，大脑皮层的特定区域兴奋，而且类似的外界信息在对应区域是连续映象的。例如，生物视网膜中有许多特定的细胞对特定的图形比较敏感，当视网膜中有若干个接收单元同时受特定模式刺激时，就使大脑皮层中的特定神经元开始兴奋，输入模式接近，对应兴奋神经元也相近。在听觉通道上，神经元在结构排列上与频率的关系十分密切，对于某个频率，特定的神经元具有最大的响应，位置邻近的神经元具有相近的频率特征，而远离的神经元具有频率特征差别也较大。大脑皮层中神经元的这种响应特点不是先天安排好的，而是通过后天的学习自组织形成的。

对于某一图形或某一频率的特定兴奋过程是自组织特征映射网中竞争机制的生物学基础。而神经元的有序排列以及对外界信息的连续映象在自组织特征映射网中也有反映，当外界输入不同的样本时，网络中哪个位置的神经元兴奋开始是随机的，但自组织训练后会在竞争层形成神经元的有序排列，功能相近的神经元非常靠近，功能不同的神经元离得较远。这一特点与人脑神经元的组织原理十分相似。

4.2.2 SOM 网的拓扑结构与权值调整域

4.2.2.1 拓扑结构

SOM 网共有两层，输入层各神经元通过权向量将外界信息汇集到输出层的各神经元。输入层的形式与 BP 网相同，节点数与样本维数相等。输出层也是竞争层，神经元的排列有多种形式，如一维线阵、二维平面阵和三维栅格阵。常见的是前两种类型，下面分别予以介绍。

输出层按一维阵列组织的 SOM 网是最简单的自组织神经网络，其结构特点与图 4.1 中的网络相同，图 4.6(a) 中的一维阵列 SOM 网的输出层只标出相邻神经元间的侧向连接。

输出按二维平面组织是 SOM 网最典型的组织方式，该组织方式更具有大脑皮层的形象。输出层的每个神经元同它周围的其他神经元侧向连接，排列成棋盘状平面，结构如图 4.6(b) 所示。

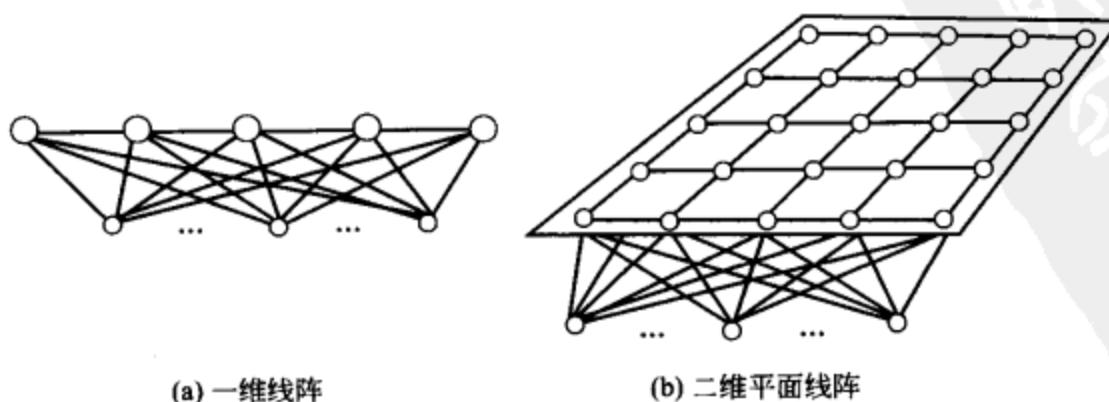


图 4.6 SOM 网的输出阵列

4.2.2.2 权值调整域

SOM 网采用的学习算法称为 Kohonen 算法，是在胜者为王算法基础上加以改进而成的，其主要区别在于调整权向量与侧抑制的方式不同。在胜者为王算法中，只有竞争获胜神经元才能调整权向量，其他任何神经元都无权调整，因此它对周围所有神经元的抑制是“封杀”式的。而 SOM 网的获胜神经元对其邻近神经元的影响是由近及远，由兴奋逐渐转变为抑制，因此其学习算法中不仅获胜神经元本身要调整权向量，它周围的神经元在其影响下也要程度不同地调整权向量。这种调整可用图 4.7 中的三种函数表示，其中图 4.7(b) 中的函数曲线是由图 4.7(a) 中的两个正态曲线组合而成的。

将图 4.7(b)、(c) 和 (d) 中的三种函数沿中心轴旋转后可形成形状似帽子的空间曲面，按顺序分别称为墨西哥帽函数、大礼帽函数和厨师帽函数。其中墨西哥帽函数是 Kohonen 提出来的，它表明获胜节点有最大的权值调整量，邻近的节点有稍小的调整量，离获胜节点距离越大，权的调整量越小，直到某一距离 R 时，权值调整量为零。当距离再远一些时，权值调整量略负，更远时又回到零。墨西哥帽函数表现出的特点与生物系统的十分相似，但其计算上的复杂性影响了网络训练的收敛性。因此在 SOM 网的应用中常使用与墨西哥函数类似的简化函数，如大礼帽函数和进一步简化的厨师帽函数。

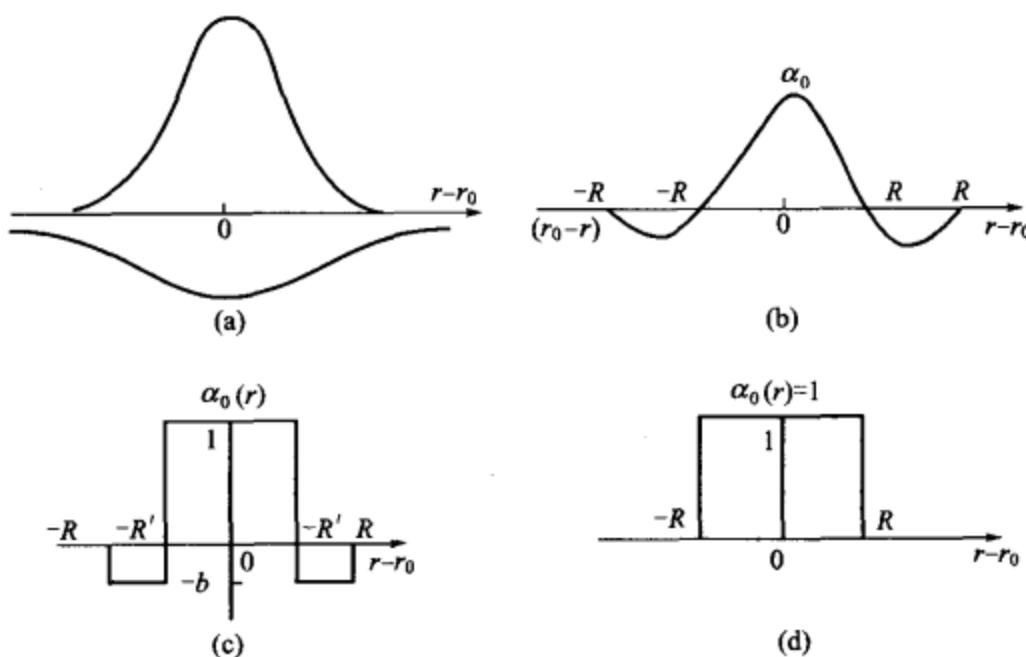


图 4.7 三种激励函数

以获胜神经元为中心设定一个邻域半径，该半径圈定的范围称为优胜邻域。在 SOM 网学习算法中，优胜邻域内的所有神经元均按其离开获胜神经元的距离远近不同程度地调整权值。优胜邻域开始定得很大，但其大小随着训练次数的增加不断收缩，最终收缩到半径为零。

4.2.3 自组织特征映射网的运行原理与学习算法

4.2.3.1 运行原理

SOM 网的运行分训练和工作两个阶段。在训练阶段，对网络随机输入训练集中的样本。对某个特定的输入模式，输出层会有某个节点产生最大响应而获胜，而在训练开始阶段，输出层哪个位置的节点将对哪类输入模式产生最大响应是不确定的。当输入模式的类别改变时，二维平面的获胜节点也会改变。获胜节点周围的节点因侧向相互兴奋作用也产生较大响应，于是获胜节点及其优胜邻域内的所有节点所连接的权向量均向输入向量的方向作程度不

同的调整，调整力度依邻域内各节点距获胜节点的远近而逐渐衰减。网络通过自组织方式，用大量训练样本调整网络的权值，最后使输出层各节点成为对特定模式类敏感的神经细胞，对应的内星权向量成为各输入模式类的中心向量。并且当两个模式类的特征接近时，代表这两类的节点在位置上也接近。从而在输出层形成能够反映样本模式类分布情况的有序特征图。

SOM 网训练结束后，输出层各节点与各输入模式类的特定关系就完全确定了，因此可用作模式分类器。当输入一个模式时，网络输出层代表该模式类的特定神经元将产生最大响应，从而将该输入自动归类。应当指出的是，当向网络输入的模式不属于网络训练时见过的任何模式类时，SOM 网只能将它归入最接近的模式类。

4.2.3.2 学习算法

对应于上述运行原理的学习算法称为 Kohonen 算法，按以下步骤进行：

(1) 初始化 对输出层各权向量赋小随机数并进行归一化处理，得到 \hat{W}_j , $j=1, 2, \dots, m$ ；建立初始优胜邻域 $N_{j^*}(0)$ ；学习率 η 赋初始值。

(2) 接受输入 从训练集中随机选取一个输入模式并进行归一化处理，得到 \hat{X}^p , $p \in \{1, 2, \dots, P\}$ 。

(3) 寻找获胜节点 计算 \hat{X}^p 与 \hat{W}_j 的点积， $j=1, 2, \dots, m$ ，从中选出点积最大的获胜节点 j^* ；如果输入模式未经归一化，应按式(4.4)计算欧式距离，从中找出距离最小的获胜节点。

(4) 定义优胜邻域 $N_{j^*}(t)$ 以 j^* 为中心确定 t 时刻的权值调整域，一般初始邻域 $N_{j^*}(0)$ 较大，训练过程中 $N_{j^*}(t)$ 随训练时间逐渐收缩，如图 4.8 所示。

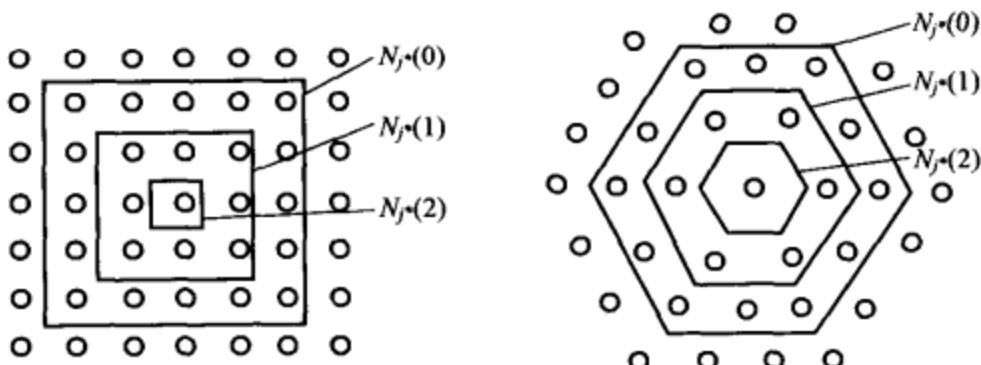


图 4.8 邻域 $N_{j^*}(t)$ 的收缩

(5) 调整权值 对优胜邻域 $N_{j^*}(t)$ 内的所有节点调整权值：

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t, N)[x_i^p - w_{ij}(t)] \quad i=1, 2, \dots, n; j \in N_{j^*}(t) \quad (4.8)$$

式中， $\eta(t, N)$ 是训练时间 t 和邻域内第 j 个神经元与获胜神经元 j^* 之间的拓扑距离 N 的函数，该函数一般有以下规律：

$$t \uparrow \rightarrow \eta \downarrow, \quad N \uparrow \rightarrow \eta \downarrow$$

很多函数都能满足以上规律，例如可构造如下函数：

$$\eta(t, N) = \eta(t) e^{-N} \quad (4.9)$$

式中， $\eta(t)$ 可采用 t 的单调下降函数，图 4.9 给出几种可用的类型。这种随时间单调下降的函数也称为退火函数。

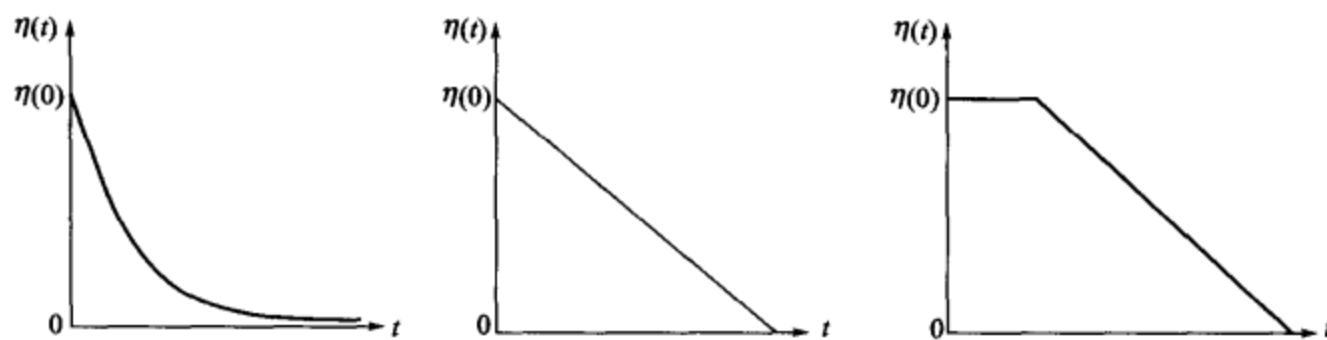


图 4.9 随时间衰减的学习率

(6) 结束检查 SOM 网的训练不存在类似 BP 网中的输出误差概念, 训练何时结束是以学习率 $\eta(t)$ 是否衰减到零或某个预定的正小数为条件, 不满足结束条件则回到步骤 (2)。

Kohonen 学习算法的程序流程图如图 4.10 所示, 对应的源程序见附录 1。

4.2.3.3 功能分析

SOM 网的功能特点之一是保序映射。即能将输入空间的样本模式类有序地映射在输出层上, 下面通过例 4.2 进行说明。

例 4.2 动物属性特征映射。

1989 年 Kohonen 给出一个 SOM 网的著名应用实例, 即把不同的动物按其属性特征映射到二维输出平面上, 使属性相似的动物在 SOM 网输出平面上的位置也相近。该例训练集中共有 16 种动物, 每种动物用一个 29 维向量来表示, 其中前 16 个分量构成符号向量, 对不同的动物进行“16 取 1”编码; 后 13 个分量构成属性向量, 描述动物的 13 种属性, 用 1 或 0 表示某动物该属性的有或无。表 4.2 中的各列给出 16 种动物的属性列向量。

SOM 网的输出平面上有 10×10 个神经元, 用 16 个动物模式轮番输入进行训练, 最后输出平面上出现图 4.11 所示的情况。可以看出, 属性相似的动物在输出平面上挨在一起, 实现了特征的有序分布。

SOM 网的功能特点之二是数据压缩。数据压缩

是指将高维空间的样本在保持拓扑结构不变的条件下投影到低维空间。在这方面, SOM 网具有明显的优势。无论输入样本空间是多少维的, 其模式样本都可以在 SOM 网输出层的某个区域得到响应。SOM 网经过训练后, 在高维空间相近的输入样本, 其输出响应节点的位置也接近。因此, 对于任意 n 维输入空间的样本, 均可通过投影到 SOM 网的一维或二维输出层上完成数据压缩。如上例中的输入样本空间为 29 维, 通过 SOM 网后压缩为二维平面的数据。

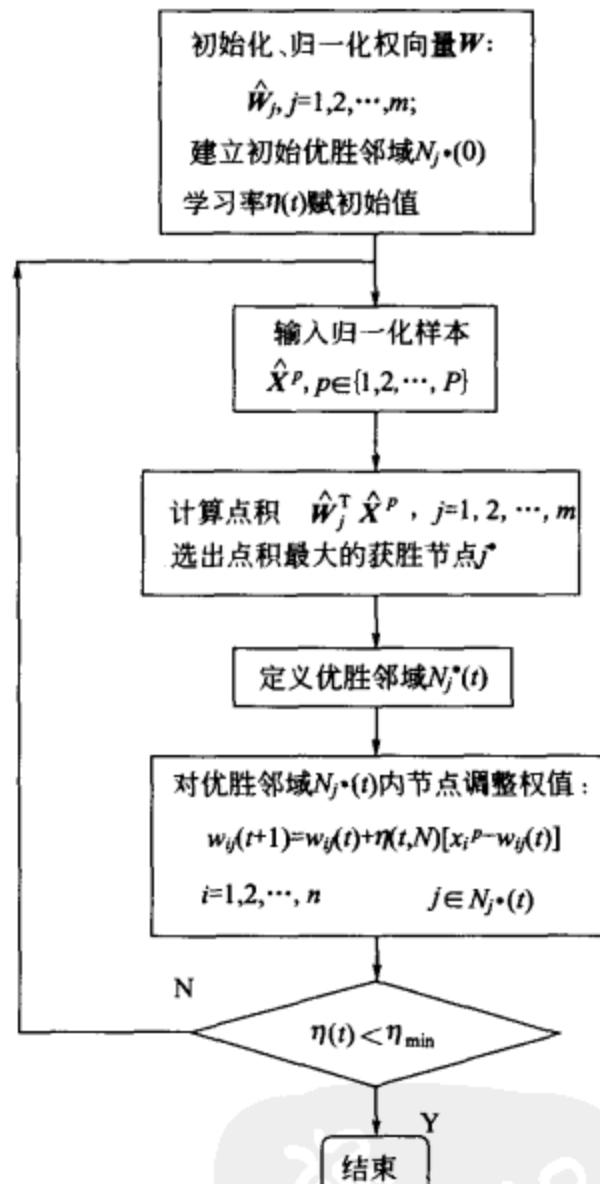


图 4.10 Kohonen 学习算法程序流程

表 4.2 16 种动物的属性向量

属性	鸽子	母鸡	鸭	鹅	猫头鹰	隼	鹰	狐狸	狗	狼	猫	虎	狮	马	斑马	牛
小	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
中	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
大	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
2只腿	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
4只腿	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
毛	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
蹄	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
鬃毛	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
羽毛	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
猎	0	0	0	0	1	1	1	1	0	1	1	1	0	0	0	0
跑	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
飞	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
泳	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

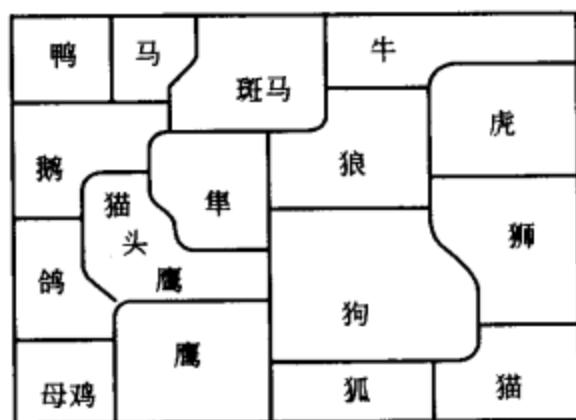


图 4.11 动物属性特征映射

SOM 网的功能特点之三是特征抽取。从特征抽取的角度看高维空间样本向低维空间的映射，SOM 网的输出层相当于低维特征空间。在高维模式空间，很多模式的分布具有复杂的结构，从数据观察很难发现其内在规律。当通过 SOM 网映射到低维输出空间后，其规律往往一目了然，因此这种映射就是一种特征抽取。高维空间的向量经过特征抽取后可以在低维特征空间更加清晰地表达，因此映射的意义不仅仅是单纯的数据压缩，更是一种规律发现。下面以字符排序为例进行分析。

例 4.3 SOM 网用于字符排序。

用 32 个字符作为 SOM 网的输入样本，包括 26 个英文字母和 6 个数字（1~6）。每个字符对应于一个五维向量，各字符与相应向量 X 的 5 个分量的对应关系见表 4.3。由表 4.3 可以看出，代表 A、B、C、D、E 的各向量中有 4 个分量相同，即 $x_i^A = x_i^B = x_i^C = x_i^D = x_i^E = 0$ ， $i=1,2,3,4$ ，因此应为一类；代表 F、G、H、I、J 的向量中有 3 个分量相同，同理也应归为一类；其他类推。这样就可根据表 4.3 中输入向量的相似关系，将对应的字符标在图 4.12 所示的树形结构图中。

表 4.3 字符与对应向量

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	1	2	3	4	5	6
x_0	1	2	3	4	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3		
x_1	0	0	0	0	0	1	2	3	4	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3			
x_2	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	3	3	3	6	6	6	6	6	6	6	6	6			
x_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	1	2	3	4	2	2	2	2			
x_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6		

SOM 网络输出阵列为二维平面阵，该阵列由 70 个神经元组成，每个神经元用五维内星向量与五维输入模式相联。将训练集中代表各字符的输入向量 X^p 随机选取后送入网络进行训练，经过 10000 步训练，各权向量趋于稳定，此时可对该网络输出进行校准，即根据输出阵列神经元与训练集的已知模式向量的对应关系贴标号。例如，当输入向量 B 时，输出平面的左上角神经元在整个阵列中产生最强的响应，于是该神经元被标为 B。在输出层的 70 个神经元中，有 32 个神经元有标号而另外 38 个为未用神经元。

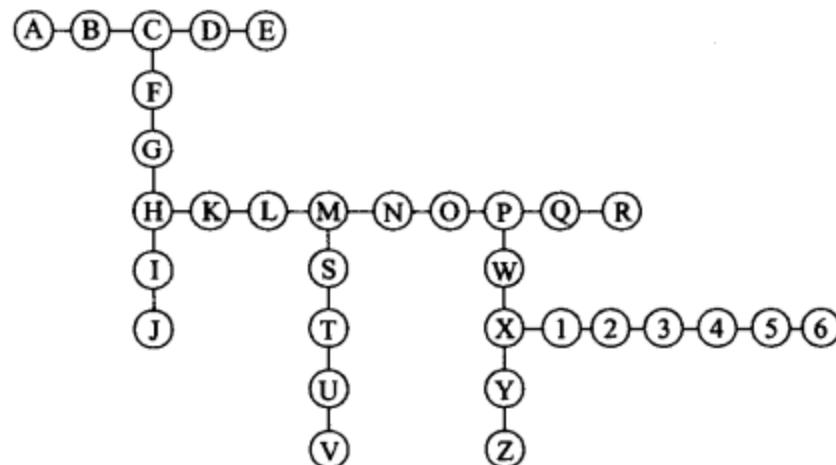


图 4.12 字符相似关系的树形结构

图 4.13 给出通过自组织学习后的输出结果。SOM 网完成训练后，对于每一个输入字符，输出平面中都有一个特定的神经元对其最敏感，这种输入-输出的映射关系在输出特征平面中表现得非常清楚。SOM 网经自组织学习后在输出层形成了有规则的拓扑结构，在神经元阵列中各字符之间的相互位置关系与它们在树状结构中的相互位置关系相当类似，两者结构特征上的一致性是非常明显的。输出平面上的“·”表示处于自由状态的神经元，它们对任何输入样本都不会发生兴奋。

B	C	D	E	•	Q	R	•	Y	Z
A	•	•	•	•	P	•	•	X	•
•	F	•	N	O	•	W	•	•	1
•	G	•	M	•	•	•	•	2	•
H	K	L	•	T	U	•	3	•	•
•	I	•	•	•	•	•	4	•	•
•	J	•	S	•	V	•	5	6	

图 4.13 SOM 网字符排序输出阵列

4.3 自组织特征映射网络的设计与应用

4.3.1 SOM 网的设计基础

SOM 网输入层的设计与 BP 网相似，而输出层的设计以及网络参数的设计比 BP 网复杂得多，是网络设计的重点。下面分几个方面讨论：

4.3.1.1 输出层设计

输出层的设计涉及两个问题：一个是节点数的设计；另一个是节点排列的设计。节点数与训练集样本有多少模式类有关。如果节点数少于模式类数，则不足以区分全部模式类，训练的结果势必将相近的模式类合并为一类。这种情况相当于对输入样本进行“粗分”。如果节点数多于模式类数，一种可能是将类别分得过细，而另一种可能是出现“死节点”，即在训练过程中，某个节点从未获胜过且远离其他获胜节点，因此它们的权向量从未得到过调整。在解决分类问题时，如果对类别数没有确切信息，宁可先设置较多的输出节点，以便较好地映射样本的拓扑结构，如果分类过细，再酌情减少输出节点。“死节点”问题一般可通过重新初始化权值得到解决。

输出层的节点排列成哪种形式取决于实际应用的需要，排列形式应尽量直观反映出实际问题的物理意义。例如，对于旅行路径类的问题，二维平面比较直观；对于一般的分类问题，一个输出节点就能代表一个模式类，用一维线阵意义明确且结构简单；而对于机器人手臂控制问题，按三维栅格排列的输出节点更能反映出手臂运动轨迹的空间特征。

4.3.1.2 权值初始化问题

SOM 网的权值一般初始化为较小的随机数，这样做的目的是使权向量充分分散在样本空间。但在某些应用中，样本整体上相对集中于高维空间的某个局部区域，权向量的初始位置却随机地分散于样本空间的广阔区域，训练时必然是离整个样本群最近的权向量被不断调整，并逐渐进入全体样本的中心位置，而其他权向量因初始位置远离样本群而永远得不到调整。如此训练的结果可能使全部样本聚为一类。解决这类问题的思路是尽量使权值的初始位置与输入样本的大致分布区域充分重合。图 4.14 给出两种初始权值的分布情况，显然，当初始权向量与输入模式向量整体上呈混杂状态时，不仅不会出现所有样本聚为一类的情况，而且会大大提高训练速度。

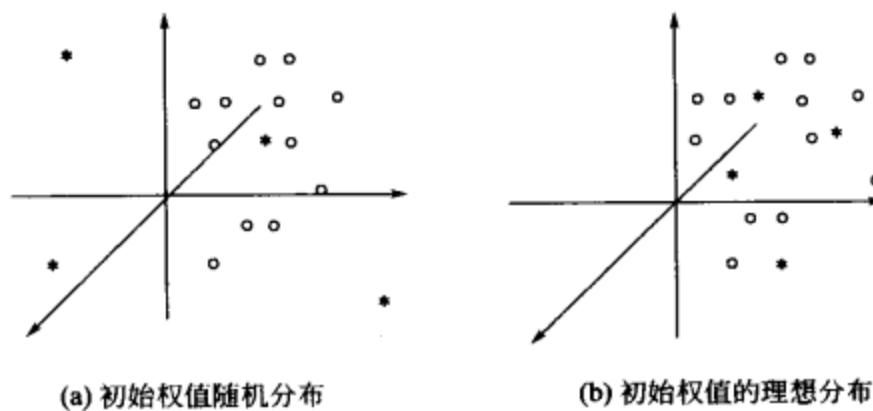


图 4.14 权向量的初始化

根据上述思路，一种简单易行的方法是从训练集中随机抽出 m 个输入样本作为初始权值，即：

$$\mathbf{W}_j(0) = \mathbf{X}^{k_{ram}} \quad j=1, 2, \dots, m \quad (4.10)$$

式中， k_{ram} 是输入样本的顺序随机数， $k_{ram} \in \{1, 2, \dots, P\}$ 。因为任何 $\mathbf{X}^{k_{ram}}$ 一定是输入空间某个模式类的成员，各个权向量按上式初始化后从训练一开始就分别接近了输入空间的各模式类，占据了十分有利的“地形”。另一种可行的办法是先计算出全体样本的中心向量：

$$\bar{\mathbf{X}} = \frac{1}{P} \sum_{p=1}^P \mathbf{X}^p \quad (4.11)$$

在该中心向量基础上叠加小随机数作为权向量初始值，也可将权向量的初始位置确定在样本群中。

4.3.1.3 优胜邻域 $N_j \cdot (t)$ 的设计

优胜邻域 $N_j \cdot (t)$ 的设计原则是使邻域不断缩小，这样输出平面上相邻神经元对应的权向量之间既有区别又有相当的相似性，从而保证当获胜节点对某一类模式产生最大响应时，其邻近节点也能产生较大响应。邻域的形状可以是正方形、六边形或圆形。

优胜邻域的大小用邻域半径表示， $r(t)$ 的设计目前还没有一般化的数学方法，通常凭经验选择。下面给出两种计算式：

$$r(t) = C_1 \left(1 - \frac{t}{t_m} \right) \quad (4.12)$$

$$r(t) = C_1 \exp \left(\frac{-B_1 t}{t_m} \right) \quad (4.13)$$

式中， C_1 为与输出层节点数 m 有关的正常数； B_1 为大于 1 的常数； t_m 为预先选定的最大训练次数。

4.3.1.4 学习率 $\eta(t)$ 的设计

$\eta(t)$ 是网络在时刻 t 的学习率，在训练开始时 $\eta(t)$ 可以取值较大，之后以较快的速度下降，这样有利于很快捕捉到输入向量的大致结构。然后 $\eta(t)$ 又在较小的值上缓降至 0 值，这样可以精细地调整权值使之符合输入空间的样本分布结构，按此规律变化的 $\eta(t)$ 表达式如下：

$$\eta(t) = C_2 \left(1 - \frac{t}{t_m}\right) \quad (4.14)$$

还有一种 $\eta(t)$ 随训练时间线性下降至 0 值的规律

$$\eta(t) = C_2 \exp\left(\frac{-B_2 t}{t_m}\right) \quad (4.15)$$

式中， C_2 为 0~1 之间的常数； B_2 为大于 1 的常数。

SOM 网的功能特色明显，但也存在以下局限性：

- ① 隐层神经元数目难以确定，因此隐层神经元往往未能充分利用，某些距离学习向量远的神经元不能获胜，从而成为死节点；
- ② 聚类网络的学习速率需人为确定，学习终止往往需要人为控制，影响学习进度；
- ③ 隐层的聚类结果与初始权值有关。

4.3.2 设计与应用实例

4.3.2.1 SOM 网用于物流中心城市分类评价

近年来，随着现代物流理念在国内的普及，北京、天津、上海、广州、深圳、苏州、厦门、芜湖、武汉等发达城市纷纷投资，积极筹建各类不同层次、规模和功能的物流中心。一个城市是否可以作为物流中心城市，除了其自身需要具备一定条件外，还要考虑与区域内其他城市之间的协调发展，同时考虑到各个城市在区域内具有不同的地位和作用，可能存在多个不同层次和类型的物流中心城市，因此这实际上是一个分类评价问题。

(1) 物流中心城市评价指标与数据样本 物流中心城市除了要有一定的经济实力外，还要具备较强的区位优势和良好的交通运输条件，因此可以从经济实力和运输条件两方面来构建评价指标体系。为了说明 SOFM 方法的可应用性，仅简单选取 5 个评价指标作为网络输入： X_1 —人均 GDP (元)； X_2 —工业总产值 (亿元)； X_3 —社会消费品零售总额 (亿元)； X_4 —批发零售贸易总额 (亿元)； X_5 —货运总量 (万吨)。以国内 44 个公路主枢纽城市作为分类评价对象，建立如表 4.4 所示的数据样本。

表 4.4 物流中心城市分类评价样本

城市	X_1	X_2	X_3	X_4	X_5	城市	X_1	X_2	X_3	X_4	X_5
北京	27527	2738.30	1494.83	3055.63	30500	上海	40788	6935.57	1531.89	3921.2	49499
天津	22073	2663.56	782.33	1465.65	28151	南京	26697	1579.21	401.20	1253.73	14120
石家庄	25584	467.42	156.02	763.46	12415	徐州	19727	295.73	108.17	187.39	7124
唐山	19387	338.67	95.73	199.69	14522	连云港	17869	112.18	47.94	134.89	4096
太原	13919	304.13	141.94	155.22	15170	杭州	31784	1615.63	373.28	1788.29	15841
呼和浩特	13738	82.23	69.27	108.12	2415	宁波	46471	751.58	167.70	529.68	11182
沈阳	21736	729.04	590.26	1752.4	15156	温州	29781	381.93	233.44	272.84	6292
大连	34659	1003.56	431.83	728.08	19736	合肥	19770	330.14	140.14	328.98	2903
长春	24799	900.26	309.75	173.99	10346	福州	33570	379.51	209.72	613.24	7280
哈尔滨	20737	402.73	360.38	762.94	8814	厦门	42039	803.29	186.55	620.47	2547

续表

城市	X_1	X_2	X_3	X_4	X_5	城市	X_1	X_2	X_3	X_4	X_5
南昌	19923	238.82	14.09	348.21	3246	南宁	17715	109.39	142.08	264.32	3371
济南	25642	616.97	323.08	462.39	13057	柳州	17598	256.76	68.93	159.44	3397
青岛	29682	1212.02	182.80	598.06	29068	海口	24782	100.13	81.03	142.54	2018
烟台	21017	298.73	92.71	227.39	8178	成都	22956	412.23	400.56	754.07	23724
郑州	17330	261.80	215.63	402.98	7373	重庆	9778	870.82	389.60	823.72	29470
武汉	17882	1020.84	685.82	1452	16244	贵阳	13176	207.95	108.93	285.27	4885
长沙	26327	241.76	269.93	369.83	7550	昆明	24554	303.78	227.44	428.64	12084
衡阳	12386	61.53	63.95	72.65	3004	西安	16002	449.14	323.37	558.27	7728
广州	42828	2446.97	1166.10	3214.19	24500	兰州	16629	354.30	163.97	374.9	5401
深圳	152099	3079.63	609.26	801.06	5167	西宁	7261	38.00	48.95	91.14	1837
汕头	19414	192.93	112.96	280.84	1443	银川	12779	77.74	41.22	53.16	1573
湛江	15290	228.45	99.08	149.16	5524	乌鲁木齐	19793	251.19	129.05	277.8	9283

注：表中的数据来自 2002 年城市统计年鉴。

(2) 物流中心城市的分类和评价分析 从物流中心城市在区域经济发展和区域物流网络中的地位和作用来看，可以划分为全国性物流中心城市、区域性物流中心城市和地区性物流中心城市 3 个层次。而从城市在经济水平和货运总量两方面的发展状况来衡量，又可以分为综合型和货运型两大类。按照 SOM 算法步骤，取开始的 1000 次迭代为排序阶段，学习率为 0.9；其后为收敛阶段，学习率为 0.02。将表 4.4 提供的数据样本进行归一化处理，输入网络进行训练。经过试验比较，最终取类别数为 8，得到如表 4.5 所示的分类结果。

表 4.5 物流中心城市分类结果

类别	城市	X_1 均值	X_2 均值	X_3 均值	X_4 均值	X_5 均值
1	北京, 上海, 广州	37048	4040.3	1397.6	3397	34833
2	天津	22073	2663.6	782.33	1465.7	28151
3	沈阳, 南京, 杭州, 武汉	24525	1236.2	512.64	1561.6	15340
	深圳	152099	3079.63	609.26	801.06	5167
4	大连, 青岛, 成都, 重庆	24269	874.66	351.19	726.98	25500
5	石家庄, 唐山, 太原, 宁波, 济南, 昆明	25926	463.76	185.32	423.18	13072
6	长春, 哈尔滨, 温州, 福州, 厦门, 郑州, 长沙, 西安	26323	477.55	263.6	471.82	7241.3
7	徐州, 合肥, 烟台, 兰州, 乌鲁木齐	19387	306.02	126.81		
8	呼和浩特, 连云港, 南昌, 衡阳, 汕头, 湛江, 南宁, 柳州, 海口, 贵阳, 西宁, 银川	15994	142.18	74.868	174.15	3067.4

根据表 4.5 中的分类和评价结果，可以得出以下认识：

类别 1 和类别 2 中的城市（北京、上海、广州、天津）属于全国性物流中心城市，通过对比可以看出，类别 1 偏于综合型，类别 2 偏于货运型。

类别 3、类别 4 和类别 5 中的城市（沈阳、南京、杭州、武汉、大连、青岛、成都、重庆、石家庄、唐山、太原、宁波、济南、昆明）属于区域性物流中心城市，其中类别 3 偏于综合型，而类别 4 和类别 5 偏于货运型。

类别 6 和类别 7 中的城市（长春、哈尔滨、温州、福州、厦门、郑州、长沙、西安、徐州、合肥、烟台、兰州、乌鲁木齐）属于地区性物流中心城市。

类别 8 中的城市从其目前的经济发展状况和货运总量来看，成为物流中心城市时机和条件尚不是很成熟。

从表 4.5 的分类结果来看，类别 3 中还包括深圳。其本身具有很强的经济实力，但物流量优势并不明显，而且和其距离很近的广州作为全国性物流中心城市，两者所面对的物流吸引区基本一致，深圳如果再要建设物流中心城市，必须考虑与广州的协调发展。

4.3.2.2 SOM 网用于遥感影像分类

遥感影像主要通过像元亮度值的差异或其空间梯度变化来表示不同地物间的差异。像元间的亮度差异反映了地物的光谱信息的差异，而空间变化的差异则反映了地物的空间信息，这是遥感影像分类的物理依据。遥感影像的传统统计模式识别方法是采用 Bayes 分类器实现的，该方法假定各类服从 Gauss 正态分布，按待分像元与已知模式的相近程度进行分类。在多源数据的情况下，不同来源的空间数据可能不具备正态分布特征。另外，一些地面实测的离散类别数据不一定符合统计分布模型，传统统计分类方法难以对此进行分类。人工神经网络方法可解决上述存在的困难，下面介绍 SOM 网络在遥感影响分类中的应用：

(1) 土地利用分类类别的确定 根据“浙江省国土资源遥感综合调查”项目，将浙江省绍兴市及其附近地貌类型丰富地区作为实验区，实验区影像为 700×420 个像元。

使用美国地球资源探测卫星 Landsat TM1~5 和 TM7 波段数据^①，应用 SOM 网结合地理辅助数据对遥感影像的土地利用和覆盖情况进行分类。参照全国农业区划委员会 1984 年颁发的《土地利用现状调查规程》，结合实地考察结果，确定土地利用的类别为 7 类，即水体、林地、水田、茶园、旱地、居民地和桑园等。

(2) 数据预处理 图 4.15 为分类类别在 TM 各波段的波谱特征。可以看出水体和林地有较好的光谱分辨性，而旱地、桑园和茶园等其他地类光谱特征较为接近，因此需要增加一些地理辅助数据以获得较好的分类精度。从图 4.15 还可以看出，TM1、TM2、TM7 波段中各分类类别的亮度值较为接近，所以对 TM1、TM2、TM7 波段进行 KL (主成分) 变换，取其第一分量 KL1 作为一个新波段。因此，SOM 网络分类所采用的光谱数据为 KL1、TM3、TM4 和 TM5。随光谱数据同时输入神经网络的地理辅助数据为 DEM^② 数据和坡度数据，所有数据均需进行归一化处理。

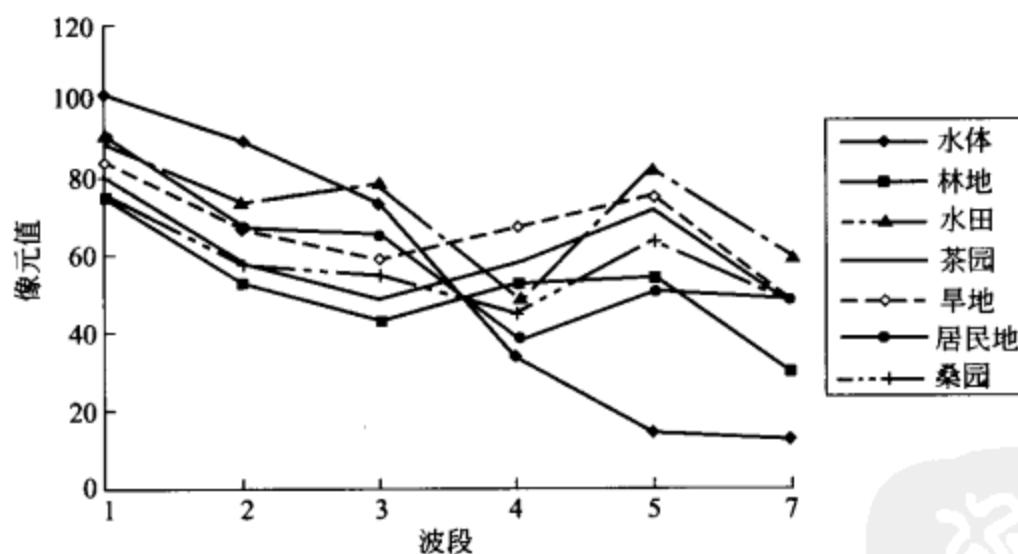


图 4.15 分类类别在 TM1~5 和 TM7 波段的波谱特征

(3) SOM 网分类 通过上述数据预处理，得到四种光谱数据以及两种辅助数据，从而构成六维输入模式向量。由于实验区土地利用类型共分为 7 类，输出层的自组织图采用 10×10 的二维平面阵，以表达各种分类类别的特征。共使用 600 个像元样本来训练网络，设定学习率

① 1972 年起，美国发射了系列陆地卫星，包括陆地卫星 Landsat1 号至 7 号，所携带的传感器由四波段的多光谱扫描仪发展到 20 世纪 80 年代初投入使用专题制图仪 (TM，7 个波段)。

② 数字高程模型 (DEM) 是一定区域范围内规则格网点的平面坐标及其高程的数据集或经维度和海拔高度的数据集。

7	7	7	7	7	0	6	0	5	5
7	7	7	0	0	6	6	0	0	5
7	0	0	0	3	0	6	0	5	5
0	0	0	3	3	0	3	3	5	5
4	4	0	0	4	3	3	0	5	5
4	4	4	4	4	0	0	4	0	0
0	0	4	4	4	4	0	1	1	1
2	2	0	2	0	0	0	1	1	1
2	2	2	2	2	0	1	1	1	1
2	0	2	2	0	1	0	1	1	1

图 4.16 用于最后分类的 SOM 网的自组织图

$\eta(t)$ 的初始值为 0.5，邻域半径为 1，以 $\eta(t)$ 衰减到 0.007 为训练结束条件。图 4.16 为训练结束后 SOM 网的自组织图，用 1~7 分别表示水田、旱地、桑园、茶园、林地、居民地和水体，0 表示没有归属类的神经元。由图 4.16 可看出，共有 33 个神经元没有归属类，说明其周围各类之间较少重叠，因此该 SOM 网能较好地区分出各类别。

(4) 实验结果评价 为了验证该方法的有效性，采用基于 Bayes 统计理论的最大似然法与 BP 算法进行影像分类并与其比较。对实验区土地选取 500 个点作为实测数据，然后将 500 个点的实测数据分别与上述 3 种方法的分

类结果进行比较。表 4.6 为各分类器对每类地物的分类精度及该分类器 Kappa 系数和分类总精度。对于水体，由于其光谱可分性较好，3 种方法均能获得较好的分类精度，精度可达 95%。3 种方法中 SOM 网的有林地分类精度最高，可达 86%，最大似然法的分类精度最低。

表 4.6 不同分类方法的分类精度对照表

分类器	水体	旱地	桑园	茶园	林地	居民地	水田	Kappa 系数	总精度
最大似然法	96	73.54	64.47	70.21	85.22	78.03	85.72	75.67	77.87
BP 网	95	73.63	70.91	73.34	84.87	82.41	86.34	77.50	79.56
SOM 网	96	75.71	74.58	75.83	86.31	83.34	87.23	80.32	82.41

3 种方法中居民地分类精度相差不大，均在 84% 左右，主要误差来源于水田，居民地的零星分布和水田存在一定的混合像元，从而产生误分类。除水田外，3 种方法的桑园、茶园和旱地的分类精度均低于 76%，其中桑园和茶园、水田和旱地的光谱特征较为接近，存在较多的误分，其中最大似然法分类中桑园误分较多。水田分布较广，与其他地类均有误分。SOM 网和 BP 网的茶园和桑园分类精度均高于最大似然法的分类精度，其中 SOM 网的分类精度最高。

4.3.2.3 SOM 网用于皮革外观效果分类

皮革颜色纹理外观效果聚类常称为配皮。人工配皮的主要缺点是：配皮的结果与配皮工的个人经验密切相关，光线强弱的变化会对配皮结果造成影响，配皮工的劳动强度大，工作效率低。为了提高皮革服装的生产效率与质量，降低工人劳动强度，采用模式识别技术进行皮革颜色纹理自动聚类。然而，由于皮革颜色纹理的复杂性和聚类规则的模糊性，传统的模式识别方法难以胜任。Kohonen 的自组织映射神经网络能在输入-输出映射中保持输入（颜色纹理）空间的拓扑特性，从而使其相邻聚类神经元所对应的颜色纹理模式类子空间也相邻，这一特点非常适合于皮革配皮这类应用。

(1) 初始权向量设计 训练前各个神经元权向量须赋以初始值，常用的做法是以小随机数赋初值。但皮革颜色与纹理的分类子空间在整个高维样本空间中相对集中，而以随机数对权向量赋初值的结果是使其随机地分布于整个高维样本空间。由于 SOM 网络训练采用竞争机制，只有与输入样本最匹配的权向量才能得到最强的调整，其结果势必使所有样本都集中于某个神经元所代表的一个子类空间，而无法达到分类的目的。本应用解决这个问题的对策

是, 从 P 个输入样本中随机取 m 个对各神经元权重赋初值。实践证明, 该做法不仅可解决上述问题, 而且使训练次数大大减少。

(2) 网络结构设计 从皮革图像中提取了三个颜色特征和三个纹理特征, 输入模式为六维向量。聚类时每批 100 张皮, 平均每件皮衣需要 5~6 张皮, 因此将输出层设置 20 个神经元。每个神经元代表一类外观效果相似的皮料, 如果聚为一类的皮料不够做一件皮衣, 可以和相邻类归并使用。

(3) 网络参数设计 对于自组织神经网算法中的两个随训练次数 t 下降的函数 $\eta(t)$ 和 $N_j \cdot (t)$ 的选择, 目前尚无一般化的数学方法。本例对 $\eta(t)$ 采用了图 4.17 所示的模拟退火函数, 表达式如下:

$$\eta(t)=\begin{cases} \eta_0, & t \leq t_p \\ \eta_0 [1-(t-t_p)/(t_m-t_p)], & t > t_p \end{cases}$$

图中, t_p 为模拟退火的起始点, t_m 为模拟退火的终止点, $0 < \eta_0 \leq 1$ 。在网络训练初期, 为了很快地捕捉到输入样本空间的大致概率结构, 希望有较强的权值调整能力, 因此当训练次数 $t \leq t_p$ 时, $\eta(t)$ 取最大值 η_0 。当训练次数 $t > t_p$ 时, $\eta(t)$ 均匀下降至 0 以精细调整权值, 使之符合样本空间的概率结构。当网络神经元的权值与样本空间结构匹配后, 所对应的训练次数为 t_m 。 t_p 可取为 t_m 的分数, 如取 $t_p = 0.5t_m$ 。对于稍微复杂一些的问题, SOM 算法

常常需要上万次的迭代训练次数。但在本例中 t_m 只有几千次, 这是由于设计了合理的权重初值从而使训练次数大大减少。本例网络参数取 $\eta_0 = 0.95$, $t_m = 5000$, $t_p = 1500$ 。 $N_j \cdot (t)$ 优胜邻域在训练开始时覆盖整个输出线阵, 以后训练次数每增加 $\Delta t = t_m/P$, $N_j \cdot (t)$ 邻域两端各收缩一个神经元, 直至邻域内只剩下获胜神经元。

(4) 皮革纹理分类结果及分析 用 SOM 皮革纹理分类器对 1000 余张猪皮分 10 批进行配皮实验, 请有经验的配皮工进行人工分类, 结果证明 SOM 网的分类效果与人工分类相当。图 4.18 给出的直方图描述了某批 100 张皮在输出层的映射结果。

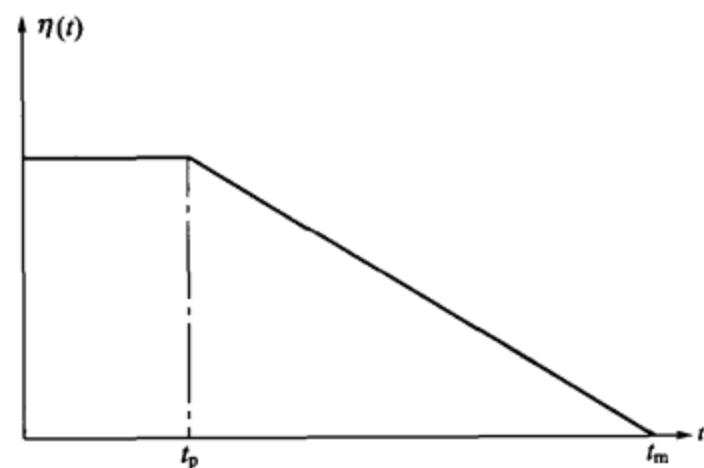


图 4.17 $\eta(t)$ 随训练次数 t 的变化

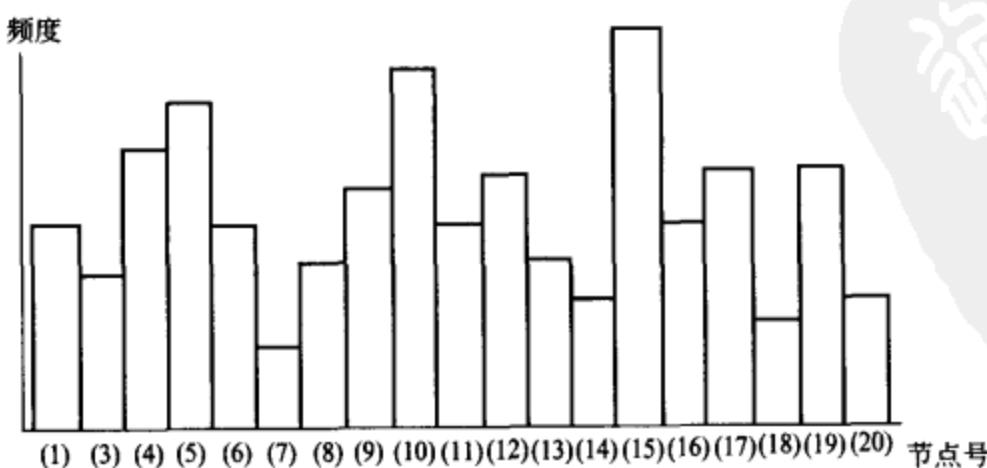


图 4.18 SOM 网配皮结果分布情况

4.3.2.4 SOM 网络用于火焰燃烧诊断

在我国的火电机组中，煤粉锅炉占了大多数。煤粉炉要求在炉膛内组织稳定均匀的火焰，保证强烈充分的燃烧。燃烧不稳定，不仅会降低锅炉热效率，产生污染物和噪声，在极端情况下还会引发炉膛燃爆事故。因此，必须在锅炉上安装有效的火检和燃烧诊断装置。燃烧火焰是表征燃烧状态稳定与否最直接的反映，而煤粉火焰辐射光的一个重要特点就是随时间变化的火焰脉动。因此，可以采集火焰的光强脉动信号，通过自组织神经网络的学习训练，对火焰处于何种燃烧工况进行有效的识别，给出燃烧诊断的判断结果。

(1) 燃烧工况特征提取 稳定燃烧和不稳定燃烧工况下的火焰辐射强度的时间序列信号较难区分，但两者在频谱分布图上却有着明显的不同，图 4.19 给出火焰燃烧不同状态的比较。当功率谱估计得到的结果显示火焰的低频波动能量变大时，表明燃烧的稳定程度恶化，发展下去可能导致熄火的发生。由此可知，低频分量功率谱幅值的大小构成了火焰燃烧稳定与否的识别标准。基于这个特点，可以采集火焰的光强脉动信号，通过快速傅里叶变换算法(FFT) 处理，得到功率谱火焰辐射强度频谱分布图，由于表征火焰燃烧稳定与否的高能量频率集中在低频部分，只需取前 30 个低频值作为 SOM 网的输入向量。

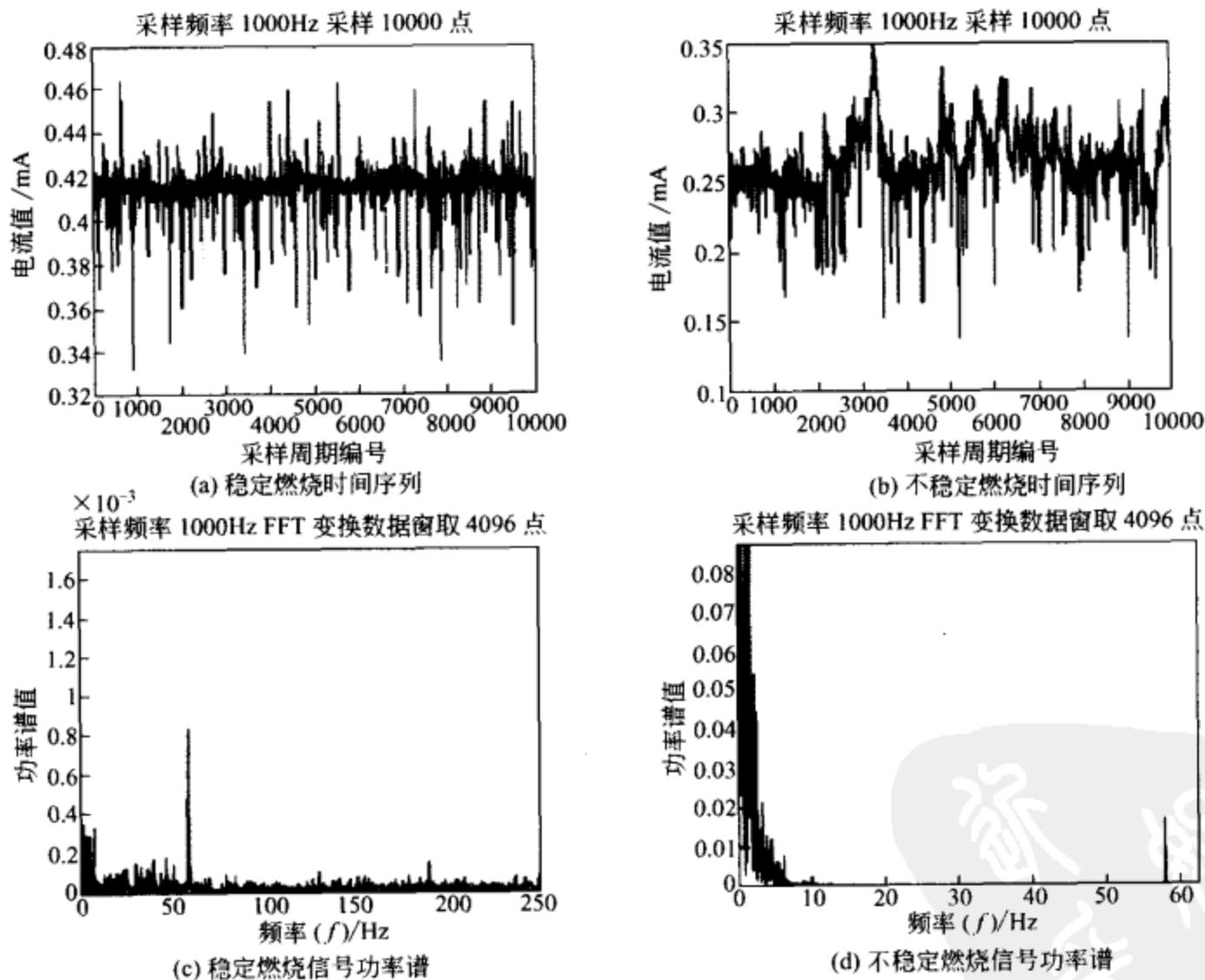


图 4.19 火焰燃烧不同状态比较

(2) 神经网络训练 用于训练的数据共 25 个输入向量，每个向量包含 30 个频率分量。其中 20 个是稳定燃烧工况下的低频谱估计值，5 个是非稳定燃烧工况下的低频谱估计值。网络使用 1×100 的一维线阵来捕捉 25 个输入向量的空间特征。应用 SOM 算法对网络进行训练，其中优胜邻域为：

$$r(t) = C_1 \left(1 - \frac{t}{t_m}\right)$$

式中, $C_1 > 0$, 是与输出层节点数 m 有关的常数; t_m 为预先选定的最大训练次数。显然, $r(t)$ 是训练次数 t 的函数, 它能动态缩小权值调整域。经反复试验, 取参数 $C_1 = 80$,

学习率 $\eta = 0.95$, 最大训练次数 t_m 取 10000 次。25 个样本向量经过 10000 次训练运算后, 得到如表 4.7 所示的神经网络分类输出。自组织训练完成后, 如果某个神经元节点被某个输入向量选为代表神经元, 即该神经元的权向量与该输入向量最接近, 则该神经元节点的序号就与输入向量组数序号对应起来, 神经元节点的序号表示输入向量经过神经网络映射之后在一维线阵的输出位置。

根据两种燃烧工况下, 输入向量在输出节点不同的位置, 可以把输出的一维线阵进行区域划分, 具体划分见表 4.8。

通过对神经网络的训练和划分可以看到: 该网络对不同燃烧工况有明显不同的反应敏感区域。对燃烧稳定工况下的火焰信号低频功率谱值的输入, 网络输出的反应敏感区在节点 1~70, 而对燃烧不稳定工况下的火焰信号低频功率谱值的输入, 网络输出的反应敏感区在节点 85~95。对输出线阵的划分并没有严格的规定, 即稳定和不稳定工况的区域边界是不确定的。如节点 1~84 以及节点 96~100 这两个区域就可以看成是两种工况的重叠区域, 即不能判别火焰稳定与否。对区域的划分是看在多次试验时, 各个工况的输入向量绝大部分落在哪个输出区域, 是从统计的角度来划分的, 而不是根据某个点, 或者是某几次的输出结果来划分的。

表 4.7 神经网络对 25 个输入向量进行训练后的输出节点位置

状态	样本号	输出节点位置
稳定	1	99
	2	1
	3	12
	4	43
	5	60
	6	79
	7	24
	8	32
	9	11
	10	30
	11	41
	12	47
	13	50
	14	10
	15	57
	16	63
	17	67
	18	71
	19	15
	20	99
不稳定	21	91
	22	89
	23	90
	24	87
	25	87

(3) 自组织神经网络的验证 为了验证该网络对没有参与训练的燃烧工况火焰信号识别的正确性, 将另外 6 组向量输入网络, 观察输出点是否落在已经划分好的识别工况区域里。其中前 3 组是稳定燃烧工况火焰信号, 后 3 组是不稳定燃烧工况火焰信号。根据上面的划分, 期待中的情况应该是: 前 3 组落在 1~70 之间, 后 3 组落在 85~95 之间。火焰低频向量信号的输出确实如预期的那样, 前 3 组工况的输出节点为 32、24、41, 后 3 组工况的输出节点为 91、91、89。表明该网络能够对火焰的燃烧稳定与否作出正确的诊断。

4.4 自适应共振理论

1976 年, 美国 Boston 大学学者 G. A. Carpenter 提出自适应共振理论 (adaptive reso-

nance theory, ART), 他多年来一直试图为人类的心理和认知活动建立统一的数学理论, ART 就是这一理论的核心部分。随后 G. A. Carpenter 又与 S. Grossberg 提出了 ATR 网络。经过了多年的研究和不断发展, ART 网已有 3 种形式: ART I 型处理双极型或二进制信号; ART II 型是 ART I 的扩展形式, 用于处理连续型模拟信号; ART III 型是分级搜索模型, 它兼容前两种结构的功能并将两层神经元网络扩大为任意多层神经元网络。由于 ART III 型在神经元的运行模型中纳入了生物神经元的生物电-化学反应机制, 因而具备了很强的功能和可扩展能力。

前面介绍的神经网络根据学习方式可分为有导师学习和无导师学习两类。对于有导师学习网络, 通过对网络反复输入样本模式使其达到稳定记忆后, 如果再加入新的样本继续训练, 前面的训练结果就会受到影响。对于无导师学习网络, 输入新数据将会对某种聚类典型向量进行修改, 这种修改意味着对新知识的学习会带来对旧知识的忘却。事实上, 许多无导师学习网络的权值调整式中都包含了对数据的学习项和对旧数据的忘却项, 通过控制其中的学习系数和忘记系数的大小来达到某种折中。但是, 如何确定这些系数的相对大小, 目前尚无通用方法。因此, 无论是有导师学习还是无导师学习, 由于给定网络的规模是确定的, 因而由矩阵 W 所能记忆的模式类别信息总是有限的, 新输入的模式样本必然会对已经记忆的模式样本产生抵消或遗忘, 从而使网络的分类性能受影响。靠无限扩大网络规模解决上述问题是不现实的。

如何保证在适当增加网络规模的同时, 在过去记忆的模式和新输入的训练模式之间作出某种折中, 既能最大限度地接收新的模式信息(灵活性), 同时又能保证较少地影响过去的模式样本(稳定性)呢? ART 网较好地解决了稳定性和灵活性兼顾的问题。

ART 网络及算法在适应新的输入模式方面具有较大的灵活性, 同时能够避免对网络先前所学模式的修改。解决这一两难问题的思路是, 当网络接受来自环境的输入时, 按预先设计的参考门限检查该输入模式与所有存储模式类典型向量之间的匹配程度, 以确定相似度。对相似度超过参考门限的所有模式类, 选择最相似的作为该模式的代表类, 并调整与该类别相关的权值, 以使以后与该模式相似的输入再与该模式匹配时能得到更大的相似度。若相似度都不超过参考门限, 就需要在网络中设立一个新的模式类, 同时建立与该模式类相连的权值, 用以代表和存储该模式以及后来输入的所有同类模式。

4.4.1 ART I 型网络

4.4.1.1 网络系统结构

从图 4.20 给出的 ART 模型结构可以看出, 该模型的结构与前面出现过的网络拓扑结构有较大区别。ART I 网络由两层神经元构成两个子系统, 分别称为比较层 C(或称注意子系统)和识别层 R(或称取向子系统)。此外还有 3 种控制信号: 复位信号(简称 Reset), 逻辑控制信号 G_1 和 G_2 。下面对图 4.20 中各部分功能作一介绍:

(1) C 层结构 C 层展开后的结构如图 4.21 所示, 该层有 n 个节点, 每个节点接受来自 3 个方面的信号: 一个是来自外界的输入信号 x_i ; 另一个是来自 R 层获胜神经元的外星向量 T_j^* 的返回信号 t_{ij}^* ; 还有一个是来自 G_1 的控制信号。C 层节点的输出是根据 2/3 的“多数表决”原则产生的, 即输出值 c_i 与 x_i 、 t_{ij}^* 、 G_1 3 个信号中的多数信号值相同。

网络开始运行时, $G_1 = 1$, 识别层尚未产生竞争获胜神经元, 因此反馈回送信号为 0。由 2/3 规则知, C 层输出应由输入信号决定, 有 $C=X$ 。当网络识别层出现反馈回送信号时, $G_1 = 0$, 由 2/3 规则, C 层输出应取决于输入信号与反馈信号的比较情况, 如果 $x_i = t_{ij}^*$, 则 $c_i = x_i$; 否则 $c_i = 0$ 。可以看出, 控制信号 G_1 的作用是使比较层能够区分网络运行的不同

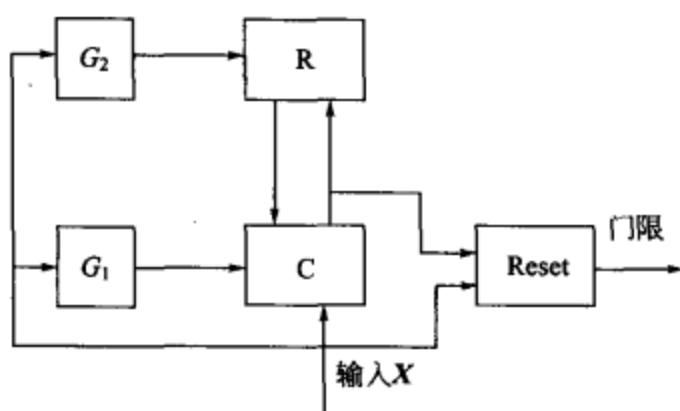


图 4.20 ART I 型网络结构

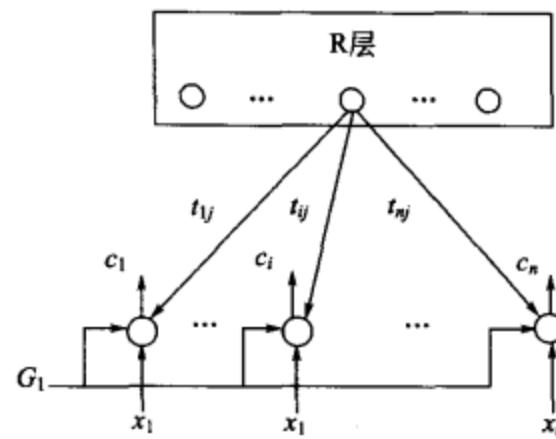


图 4.21 比较层结构示意

阶段，网络开始运行阶段 G_1 的作用是使 C 层对输入信号直接输出；之后 G_1 的作用是使 C 层行使比较功能，此时 c_i 为对 x_i 和 t_{ij}^* 的比较信号，两者为 1 时 c_i 为 1，否则为 0。可以看出，从 R 层返回的信号 t_{ij}^* 对 C 层输出有调节作用。

(2) R 层结构 R 层展开后的结构如图 4.22 所示，其功能相当于一种前馈竞争网。设 R 层有 m 个节点，用以表示 m 个输入模式类。 m 可动态增长，以设立新模式类。由 C 层向上连接到 R 第 j 个节点的内星权向量用 $\mathbf{B}_j = (b_{1j}, b_{2j}, \dots, b_{nj})$ 表示。C 层的输出向量 \mathbf{C} 沿 m 个内星权向量 \mathbf{B}_j ($j=1, 2, \dots, m$) 向前传送，到达 R 层各个神经元节点后经过竞争再产生获胜节点 j^* ，指示本次输入模式的所属类别。获胜节点输出 $r_{j^*} = 1$ ，其余节点输出为 0。R 层的每个神经元都对应两个权向量：一个是将 C 层前馈信号汇聚到 R 层的内星权向量 \mathbf{B}_j ；另一个是将 R 层反馈信号散发到 C 层的外星权向量 \mathbf{T}_j ，该向量为对应与 R 层各模式类节点的典型向量。

(3) 控制信号 3 个控制信号的作用分别是：信号 G_2 检测输入模式 X 是否为 0，它等于 X 各分量的逻辑“或”，如果 x_i ($i=1, 2, \dots, n$) 全为 0，则 $G_2 = 0$ ，否则 $G_2 = 1$ 。设 R 层输出向量各分量的逻辑“或”用 R_0 表示，则信号 $G_1 = G_2 \bar{R}_0$ ，当 R 层输出向量 \mathbf{R} 的各分量全为 0 而输入向量 X 不是零向量时， $G_1 = 1$ ，否则 $G_1 = 0$ 。正如前面所指出的， G_1 的作用是在网络开始运行时为 1，以使 $C=X$ ；其后为 0，以使 C 值由输入模式和反馈模式的比较结果决定。Reset 信号的作用是使 R 层竞争获胜神经元无效，如果根据某种事先设定的测量标准， T_j 与 X 未达到预先设定的相似度 ρ ，表明两者未充分接近，于是系统发出 Reset 信号使竞争获胜神经元无效。

4.4.1.2 网络运行原理

网络运行时接受来自环境的输入模式，并检查输入模式与 R 层所有模式类之间的匹配程度。对于匹配程度最高的模式类，网络要继续考察该模式的典型向量与当前输入模式的相似程度。相似程度按照预先设计的参考门限来考察，可能出现的情况无非有两种。①如果相似度超过参考门限，选该模式类作为当前输入模式的代表类。权值调整规则是，相似度超过参考门限的模式类调整其相应的内外星权向量，以使其以后遇到与当前输入模式接近的样本时能得到更大的相似度；对其他权值向量则不作任何变动。②如果相似度不超过门限值，则对 R 层匹配程度次高的模式类进行相似程度考察，若超过参考门限网络的运行回到情况①，

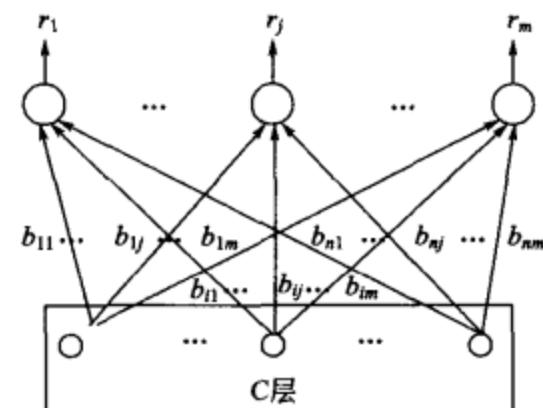


图 4.22 识别层结构示意

否则仍然回到情况②。可以想到，运行反复回到情况②意味这最终所有的模式类与当前输入模式的相似度都没有超过参考门限，此时需在网络输出端设立一个代表新模式类的节点，用以代表及存储该模式，以便于参加以后的匹配过程。网络对所接收的每个新输入样本，都进行上面的运行过程。对于每一个输入，模式网络运行过程可归纳为四个阶段：

(1) 匹配阶段 网络在没有输入模式之前处与等待状态，此时输入端 $X=0$ ，因此信号 $G_2=0$, $R_0=0$ 。当输入不全为 0 的模式 X 时， $G_2=1$, $R_0=0$ ，使得 $G_1=G_2\bar{R}_0=1$ 。 G_1 为 1 时允许输入模式直接从 C 层输出，并向前传至 R 层，与 R 层节点对应的所有内星向量 \mathbf{B}_j 进行匹配计算：

$$\text{net}_j = \mathbf{B}_j^T X = \sum_{i=1}^n b_{ij} x_i \quad j=1, 2, \dots, m \quad (4.16)$$

选择具有最大匹配度（即具有最大点积）的竞争获胜节点： $\text{net}_{j^*} = \max_j \{\text{net}_j\}$ ，使获胜节点输出 $r_{j^*}=1$ ，其他节点输出为 0。

(2) 比较阶段 R 层输出信息通过外星向量返回到 C 层。 $r_{j^*}=1$ 使 R 层获胜节点所连的外星权向量 \mathbf{T}_{j^*} 激活，从节点 j^* 发出的 n 个权值信号 t_{ij^*} 返回到 C 层的 n 个节点。此时，R 层输出不全为 0， $R_0=1$ ，而 $G_1=G_2\bar{R}_0=0$ ，所以 C 层最新输出状态 C' 取决于由 R 层返回的外星权向量 \mathbf{T}_{j^*} 和网络输入模式 X 的比较结果，即 $c_i=t_{ij^*}x_i$, $i=1, 2, \dots, n$ 。由于外星权向量 \mathbf{T}_{j^*} 是 R 层模式类的典型向量，该比较结果 C' 反映了在匹配阶段 R 层竞争排名第一的模式类的典型向量 \mathbf{T}_{j^*} 与当前输入模式 X 的相似程度。相似程度的大小可用相似度 N_0 反映，定义为：

$$N_0 = X^T \mathbf{T}_{j^*} = \sum_{i=1}^n t_{ij^*} x_i = \sum_{i=1}^n c_i \quad (4.17)$$

因为输入 x_i 为二进制数 0 或 1， N_0 实际上表示获胜节点的类别模式典型向量与输入模式样本之间相互重叠的非零分量数。设输入模式样本中的非零分量数为：

$$N_1 = \sum_{i=1}^n x_i \quad (4.18)$$

用于比较的警戒门限为 ρ ，在 0~1 范围取值。检查输入模式与模式类典型向量之间的相似性是否低于警戒门限，如果有：

$$N_0/N_1 < \rho$$

则 X 与 \mathbf{T}_{j^*} 的相似程度不满足要求，网络发出 Reset 信号使第一阶段的匹配失败，竞争获胜节点无效，网络进入搜索阶段。如果有：

$$N_0/N_1 > \rho$$

则表明 X 与获胜节点对应的类别模式非常接近，称 X 与 \mathbf{T}_{j^*} 发生“共振”，第一阶段的匹配结果有效，网络进入学习阶段。

(3) 搜索阶段 网络发出 Reset 重置信号后即进入搜索阶段，重置信号的作用是使前面通过竞争获胜的神经元受到抑制，并且在后续过程中受到持续的抑制，直到输入一个新的模式为止。由于 R 层中的竞争获胜的神经元被抑制，从而再度出现 $R_0=0$, $G_1=1$ ，因此网络又重新回到起始的匹配状态。由于上次获胜的节点受到持续的抑制，此次获胜的必然是上次匹配程度排在第二的节点。然后进入比较阶段，将该节点对应的外星权向量 \mathbf{T}_{j^*} 与输入模式进行匹配计算。如果对 R 层所有的模式类，在比较阶段的相似度检查中相似度都不能满足要求，说明当前输入模式无类可归，需要在网络输出层增加一个节点来代表并存储该模式

类, 为此将其内星向量 \mathbf{B}_j 设计成当前输入模式向量, 外星向量 \mathbf{T}_j 各分量全设为 1。

(4) 学习阶段 在学习阶段要对发生共振的获胜节点对应的模式类加强学习, 使以后出现与该模式相似的输入样本时能获得更大的共振。

ART 网络运行中存在两种记忆方式: C 层和 R 层输出信号称为短期记忆, 用 STM (short time memory) 表示, 短期记忆在运行过程中会不断发生变化; 两层之间的内外星权向量称为长期记忆, 用 LTM (long time memory) 表示, 长期记忆在运行过程中不会变化。下面对两种记忆形式进行分析:

C 层输出信号是按照 2/3 原则取值的, 在网络开始运行时, C 层输出 \mathbf{C} 与输入模式 \mathbf{X} 相等, 因而 \mathbf{C} 是对输入模式 \mathbf{X} 的记忆。当 R 层返回信号 \mathbf{T}_j 到达 C 层时, 输出 \mathbf{C} 立刻失去对 \mathbf{X} 的记忆而变成对 \mathbf{T}_j 和 \mathbf{X} 的比较信号。R 层输出信号是按照胜者为王原则取值的, 获胜神经元代表的模式类是对输入模式的类别记忆。但当重置信号 Reset 作用于 R 层时, 原获胜神经元无效, 因此原记忆也消失。由此可见, C 和 R 对输入模式 \mathbf{X} 的记忆时间非常短暂, 因此称为短期记忆。

权向量 \mathbf{T}_j 和 \mathbf{B}_j 在运行过程中不会发生变化, 只在学习阶段进行调整以进一步加强记忆。经过学习后, 对样本的记忆将留在两组权向量中, 即使输入样本改变, 权值依然存在, 因此称为长期记忆。当以后输入的样本类似已经记忆的样本时, 这两组长期记忆将 R 层输出回忆到记忆样本的状态。

4.4.1.3 网络学习算法

ART I 网络可以用学习算法实现, 也可以用硬件实现。学习算法从软件角度体现了网络的运行机制, 与图 4.20 中带有硬件特色的系统结构并不一一对应。例如, 学习算法中没有显示表现三个控制信号的作用。训练可按以下步骤进行:

(1) 网络初始化 从 C 层向上 R 层的内星权向量 \mathbf{B}_j 赋予相同的较小数值, 如:

$$b_{ij}(0) = \frac{1}{1+n} \quad i=1, 2, \dots, n; j=1, 2, \dots, m \quad (4.19)$$

从 R 层向下到 C 层的外星权向量 \mathbf{T}_j 各分量均赋 1。

$$t_{ij} = 1 \quad i=1, 2, \dots, n; j=1, 2, \dots, m \quad (4.20)$$

权初始值对整个算法影响重大, 内星权向量按式(4.19)设置, 可保证输入向量能够收敛到其应属类别而不会轻易动用未使用的节点。外星权向量按式(4.20)设置, 可保证对模式进行相似性测量时能正确计算其相似性。

相似性测量的警戒门限 ρ 设为 0~1 之间的数, 它表示两个模式相距多近才认为是相似的, 因此其大小直接影响到分类精度。

(2) 网络接受输入 给定一个输入模式, $\mathbf{X}=(x_1, x_2, \dots, x_n)$, $x_i \in (0, 1)^n$ 。

(3) 匹配度计算 对 R 层所有内星向量 \mathbf{B}_j 计算与输入模式 \mathbf{X} 的匹配度:

$$\mathbf{B}_j^T \mathbf{X} = \sum_{i=1}^n b_{ij} x_i \quad j=1, 2, \dots, m$$

(4) 选择最佳匹配节点 在 R 层有效输出节点集合 J^* 内选择竞争获胜的最佳匹配节点 j^* , 使得:

$$r_j = \begin{cases} 1, & j=j^* \\ 0, & j \neq j^* \end{cases}$$

(5) 相似度计算 R 层获胜节点 j^* 通过外星送回获胜模式类的典型向量 \mathbf{T}_{j^*} , C 层输

出信号给出对向量 T_j^* 和 X 的比较结果 $c_i = t_{ij} \cdot x_i$, $i=1,2,\dots,n$, 由此结果可计算出两向量的相似度为:

$$N_0 = \sum_{i=1}^n c_i \quad N_1 = \sum_{i=1}^n x_i$$

(6) 警戒门限检验 如果 $N_0/N_1 < \rho$, 表明 X 与 T_j^* 的相似程度不满足要求, 本次竞争获胜节点无效, 因此从 R 层有效输出节点集合 J^* 中取消该节点并使 $r_{j^*} = 0$, 训练转入步骤 (7); 如果 $N_0/N_1 > \rho$, 表明 X 应归为 T_j^* 代表的模式类, 转向步骤 (8) 调整权值。

(7) 搜索匹配模式类 如果有效输出节点集合 J^* 不为空, 转向步骤 (4) 重选匹配模式类; 若 J^* 为空集, 表明 R 层现存的所有模式类典型向量均与 X 不相似, X 无类可归, 需在 R 层增加一个节点。设新增节点的序号为 n_c , 应使 $B_{n_c} = X$, $t_{in_c} = 1$, $i=1,2,\dots,n$, 此时有效输出节点集合为 $J^* = \{1,2,\dots,m,m+1,\dots,m+n_c\}$, 转向步骤 (2) 输入新模式。

(8) 调整网络权值 修改 R 层节点 j^* 对应的权向量, 网络的学习采用了两种规则, 外星向量的调整按以下规则:

$$t_{ij^*}(t+1) = t_{ij^*}(t)x_i \quad i=1,2,\dots,n; j^* \in J^* \quad (4.21)$$

外星向量为对应模式类的典型向量或称聚类中心。按上述规则学习, 可保证相似的模式越来越聚类, 不同的模式越来越分离。内星向量的调整按以下规则:

$$b_{ij^*}(t+1) = \frac{t_{ij^*}(t)x_i}{0.5 + \sum_{i=1}^n t_{ij^*}(t)x_i} = \frac{t_{ij^*}(t+1)}{0.5 + \sum_{i=1}^n t_{ij^*}(t+1)} \quad i=1,2,\dots,n \quad (4.22)$$

可以看出, 如果不计分母中的常数 0.5, 上式相当于对外星权向量的归一化。

ART 网络的特点是非离线学习, 即不是对输入集样本反复训练后才开始运行, 而是边学习边运行实时方式。每个输出节点可以看成一类相近样本的代表, 每次最多只有一个输出节点为 1。当输入样本距某一个内星权向量较近时, 代表它的输出节点才响应。通过调整警戒门限的大小可调整模式的类数, ρ 小, 模式的类别少; ρ 大, 则模式的类别多。

用硬件实现 ART I 模型时, C 层和 R 层的神经元都用电路来实现, 作为长期记忆的权值用 CMOS 电路完成, 具体电路可参考有关资料。

4.4.2 ART I 型网络的应用

4.4.2.1 ART I 型网络在模式分类中的应用

1987 年 Carpenter 和 Grossberg 提出 ART I 型网络时用到一个例子, 如图 4.23 所示。对图中给出的 4 种模式进行分类, 输入模式 X 的维数为 $n=25$, 4 个待分类模式最多可能分成 4 类, 故取 $m=4$ 。 $X \in \{0,1\}^{25}$, 用 1 代表输入模式中的黑色, 0 代表白色, 则 4 个输入模式向量为:

$$\begin{aligned} X^A &= (1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1)^T \\ X^B &= (1,0,0,0,1,0,1,0,1,0,0,0,1,0,0,0,1,0,1,0,1,0,0,0,1)^T \\ X^C &= (1,0,0,0,1,0,1,0,1,1,1,1,1,0,1,0,1,0,1,0,0,0,1)^T \\ X^D &= (1,0,0,0,1,1,1,0,1,1,1,1,1,1,1,0,1,1,1,0,0,0,1)^T \end{aligned}$$

设 $\rho=0.7$, 取初始权值 $b_{ij} = 1/(1+n) = 1/26$, $t_{ij} = 1$, 由于 b_{ij} 的值在 0 与 1 之间, 可想象为白与黑之间的灰色。因此, 初始化后的所有内星权值向量可表示为灰度值全为 1/26 的 5×5 浅灰色网格; 初始化后的所有外星权值向量可表示为灰度值全为 1 的 5×5 黑色网格。

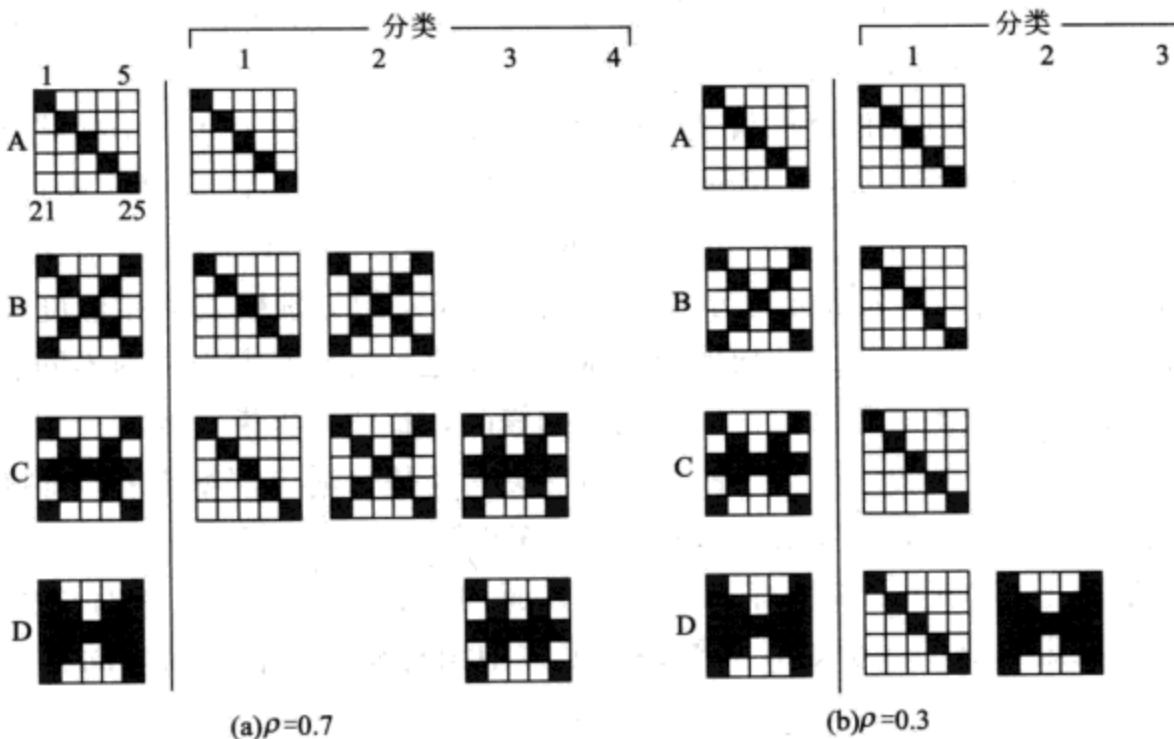


图 4.23 ART I 用于模式分类

第1步：当输入模式 \mathbf{X}^A 时，将 R 层的 4 个节点中输出最大的一个命名为节点 1，有 $j^* = 1$ 。由于初始化后 $t_{ij} = 1$ ，所以相似度 $N_0/N_1 = 1$ ，大于警戒门限 ρ ，故第一个模式被命名为第一类模式。按式(4.21)修改节点 1 的外星权向量，得 $\mathbf{T}_1 = (1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1)^T$ ，按式(4.22)修改节点 1 的内星权向量，得 $b_{1,1} = b_{7,1} = b_{13,1} = b_{19,1} = b_{25,1} = 2/11$ ，其余仍为初始值 $1/26$ 。对比输入模式 \mathbf{X}^A ，可以看出，以上调整结果使外星权向量 $\mathbf{T}_1 = \mathbf{X}^A$ ，内星权向量中与 \mathbf{X}^A 对应的分量则由浅灰（灰度值为 $1/26$ ）调整为较深的灰色（灰度值为 $2/11$ ），从而使得模式 \mathbf{X}^A 在 5×5 浅灰色网格上凸现出来。换言之，权值调整的结果使将模式 \mathbf{X}^A 分别存储在神经元 1 的内外星权向量中。

第2步：当输入模式 \mathbf{X}^B 时，R层只有一个已存储模式，故不存在类别归属的竞争，只需判断该模式与已存储模式 $\mathbf{T}_1 = \mathbf{X}^A$ 的相似度，得 $N_0/N_1 = 5/9 < \rho = 0.7$ 。从相似度可以看出，模式 \mathbf{X}^B 有 9 个黑像素，而 \mathbf{X}^A 与 \mathbf{X}^B 只有 5 个黑像素完全重合，故相似度检验不合格。由于 R 层已没有其他已存储模式类可供选择，需动用一个新节点，命名为节点 2，用以代表新模式 \mathbf{X}^B 。节点 2 的外星权向量为 $\mathbf{T}_2 = \mathbf{X}^B$ ，内星权向量为 $b_{1,2} = b_{5,2} = b_{7,2} = b_{9,2} = b_{13,2} = b_{17,2} = b_{19,2} = b_{21,2} = b_{25,2} = 2/19$ ，其余分量均为初始值。

第3步：输入模式 \mathbf{X}^C 时，节点1和节点2进行竞争，节点1的净输入为 1.217，节点2净输入为 1.101，所以节点1获胜。计算 \mathbf{T}_1 与 \mathbf{X} 的相似度，得 $N_0/N_1 = 5/13 < 0.7$ 。其中分子表明模式 \mathbf{X}^C 与记忆了模式 \mathbf{X}^A 的权向量 \mathbf{T}_1 只有 5 个黑像素重合，而分子表明模式 \mathbf{X}^C 中共有 13 个黑像素，因此两个模式的相似度较低，不能将模式 \mathbf{X}^C 归为模式 \mathbf{X}^A 类。节点1失效后，网络应在其余的存储模式类节点中搜索，对于本例，只能取节点2作为获胜节点。于是计算 \mathbf{X}^C 与代表 \mathbf{X}^B 的 $\mathbf{T}_2 \mathbf{X}$ 的相似度，得 $N_0/N_1 = 9/13 < 0.7$ 。该结果仍不能满足要求，只能把模式视为第3类模式。并按式(4.21) 和式(4.22) 修改节点3的外内星权向量。

第4步：输入模式 \mathbf{X}^D 后，节点1、节点2和节点3参加竞争，结果是节点3获胜，计算模式 \mathbf{X}^D 与 \mathbf{X}^C 的相似度，得 $N_0/N_1=13/17=0.765>0.7$ ，于是 \mathbf{X}^D 归入已存储的 \mathbf{X}^C 类，并按式(4.21)和式(4.22)修改节点3的外内星权向量。

ART I 网完成对 4 个模式的分类及存储之后，运行时当向网络输入这 4 个模式中的任一模式时，R 层中代表该类的节点输出将最大。

需要指出的是：若提高相似度的警戒门限值 ρ ，取 $\rho \geq 0.8$ ，则上述 4 个模式便不得不分成 4 类；若取 $\rho=0.3$ ，则如图 4.23(b) 所示，4 个模式被分成两类，A、B 和 C 被归入第 1 类，D 被归入第 2 类；而取 $\rho=0.2$ 时，则 4 个模式均属于同一类。由此不难看出， ρ 值的选择对分类过程的影响很大。 ρ 值过大，大部分待分类的模式与已存储的类别模式的相似度测试均难以通过，只好不断地存储新的类别模式，导致分类剧增；反之，若 ρ 值太小，则不同的模式均划为同一类别，已存储的类别模式频繁地作较大幅度的修改，致使该类模式的特征很不明显。目前尚无有效的理论来指导 ρ 值的选择。一种可行的解决途径是自适应调整 ρ 值，即随机地给出一个 ρ 值，将分类结果作为反馈信号来调整 ρ 值，直至得到合适的分类数为止。

在无噪声情况下，ART I 在训练与运行两方面均有很好的性能。它对单极性二进制输入向量的分类是稳定可靠的。但是，只要训练模式中稍有噪声，就会引发问题。

4.4.2.2 ART I 型网络对含噪声模式的分类

图 4.24 第一行所示的 4 个无噪声字符 A、B、C 与 D，用单极性二进制描述，每字符为 5×5 点阵。

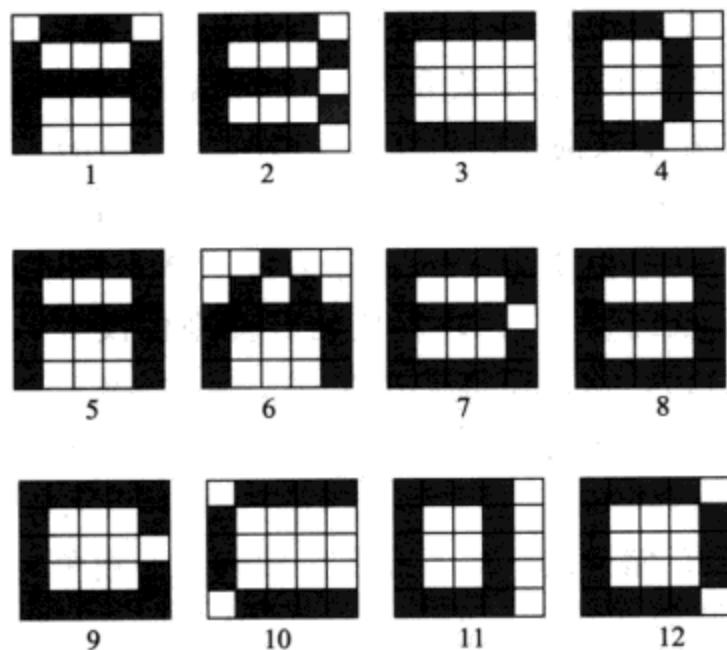


图 4.24 带噪声模式分类

首先用 4 个无噪声模式训练网络，当 ρ 值较大 ($0.85 \sim 0.95$) 时，4 个模式被分成 4 类。此后，若将图 4.24 中的 5 号样本即带有噪声的模式 A 输入网络，由表 4.9 可以看出，只有当 $\rho \leq 0.85$ 时，才能被划入模式 A 类，否则被视为第 5 类模式。当把 6 号样本即另一个带噪声的字符 A 输入网络时，如果 $\rho=0.85$ ，则该输入归为第 5 类（第一个带噪声的字符 A），如果 $\rho \geq 0.9$ ，则视为第 6 类。

第 7、8 个样本都是带噪声的模式 B，其分类情况与上述相似。如果 $\rho=0.85$ ，样本 7 归入模式 2 类，样本 8 归入模式 6 类；当 $\rho=0.9$ 时，均视为第 7 类；而 $\rho=0.95$ 时，则分别视

为新建的第 7 类和第 8 类。在不同 ρ 值下，其他带噪声样本的分类情况也列入表 4.9 中。

表 4.9 不同 ρ 值时网络对带噪声模式的分类结果

样本序号	期望分类结果	实际分类结果		
		$\rho=0.95$	$\rho=0.90$	$\rho=0.85$
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	1	5	5	1
6	1	6	6	5
7	2	7	7	2
8	2	8	7	6
9	3	7	7	3
10	3	3	3	3
11	4	9	8	4
12	4	8	7	4

4.4.2.3 ART I型网络的在直流提升机故障诊断中的应用

(1) 故障样本编码 通过总结和归纳，直流提升机的故障共有 9 种。根据图 4.25 所示的层次关系，按一定特征对故障类型进行编码，以形成故障样本。9 种故障的诊断过程分为三级，故进行二进制编码时对各位的定义也分成三级。左起第 1、2 位为第一级（包括一位冗余）；第 3、4 位为第二级，因为第二级只有 4 类故障，需要 2 位编码；第 5、6、7 位为第三级，共有 7 类故障，需要 3 位编码。根据以上编码规则得到编码表 4.10。

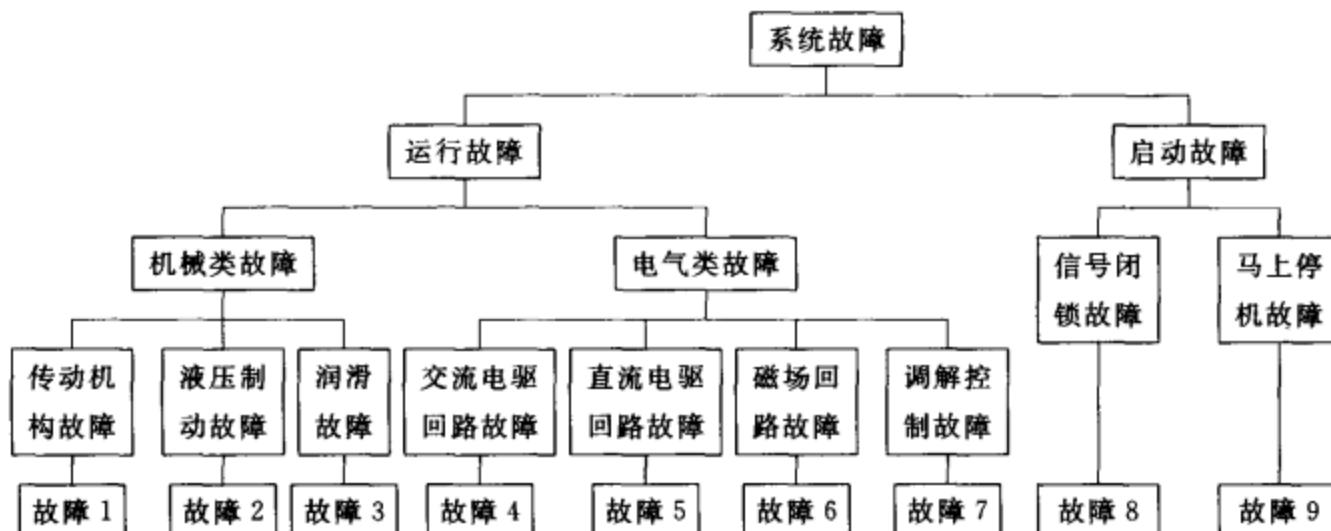


图 4.25 直流提升机故障类型

表 4.10 故障样本二进制编码

故障类型	故障编码	故障类型	故障编码
故障 1	1100000	故障 6	1101101
故障 2	1100001	故障 7	1101110
故障 3	1100010	故障 8	0010000
故障 4	1101011	故障 9	0011000
故障 5	1101100		

(2) 网络学习 首先依次输入 9 个故障样本编码让网络学习，此时由于网络不具备任何知识，故每输入一个样本就产生一个新故障分类存于网络识别层内。然后再输入一组已学习或未学习的样本编码作为测试样本，以验证 ART 诊断的正确性。仿真结果如表 4.11 所示。可以看出，对于测试样本，若超过警戒门限则产生新的分类模式，若在警戒门限内，则认为其误差为噪声信号。由于第一级编码采取了冗余技术，使得网络在警戒门限内正确地诊断了带有噪声的故障。因此，在各级都加入冗余位是必要的。

表 4.11 故障分类结果

组别	输入样本	匹配模式	新模式
训练组	1100000		故障 1
	1100001		故障 2
	1100010		故障 3
	1101011		故障 4
	1101100		故障 5
	1101101		故障 6
	1101110		故障 7
	0010000		故障 8
	0011000		故障 9

续表

组别	输入样本	匹配模式	新模式
测试组	1100001	故障 2	
	1101011	故障 4	
	1101101	故障 6	
	1001110	故障 7	
	0011000	故障 9	
	0110000	故障 8	
	0011111	故障 10	

4.4.3 ARTⅡ型网络

ARTⅡ神经网络不仅能对双极型或二进制输入模式分类，而且能够对模拟输入模式的任意序列进行自组织分类，其基本设计思想仍然是采用竞争学习策略和自稳机制。

4.4.3.1 网络结构与运行原理

ARTⅡ结构如图 4.26 所示，图 4.27 中给出第 i 个处理单元的拓扑连接。ARTⅡ由注意子系统和取向子系统组成。注意子系统中包括短期记忆 STM 特征表示场 F_1 和短期记忆类别表示场 F_2 。 F_1 相当于 ARTⅠ中的比较层，包括几个处理级和增益控制系统。 F_2 相当于 ARTⅠ中的识别层，负责对当前输入模式进行竞争匹配。 F_1 和 F_2 共有 N 个神经元，其中 F_1 场有 M 个， F_2 场有 $N-M$ 个，共同构成了 N 维状态向量代表网络的短期记忆。 F_1 和 F_2 之间的内外星连接权向量构成了网络的自适应长期记忆 LTM，由下至上的权值用 z_{ij} 表示，由上至下的权值用 z_{ji} 表示。取向子系统由图 4.27 左侧的复位系统组成。

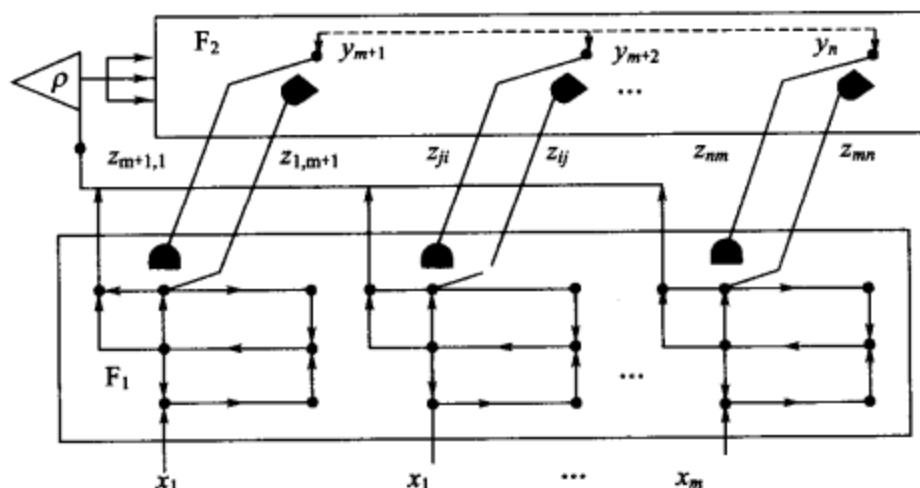


图 4.26 ARTⅡ神经网络

F_1 场的 M 个神经元从外界接受输入模式 X ，经场内的特征增强与噪声抑制等处理后通过由下至上的权值 z_{ij} 送到 F_2 场。 F_2 场的 $N-M$ 个神经元接收 F_1 场上传的信号，经过竞争确定哪个神经元获胜，获胜神经元被激活，其他则均被抑制。与激活神经元相连的内外星权向量进行调整。增益控制系统负责比较输入模式与 F_2 场激活神经元的外星权向量之间的相似程度，当两向量的相似度低于警戒门限时，复位子系统发出信号抑制 F_2 场的激活神经元。网络将在 F_2 场另选一个获胜神经元，直到相似度满足要求。如果 F_2 场的神经元数 $N-M$ 大于可能的输入模式类别数，总可以为所有新增的模式类分配一个代表神经元。

结合图 4.26 和以上分析看出，ARTⅡ网中有两种存储机制、两种连接权和两种抑制信号。两种存储机制是指：①长期记忆—— F_1 与 F_2 之间的权值；②短期记忆—— F_1 与 F_2 中的神经元状态。两种连接权是指：① $F_1 \rightarrow F_2$ 的内星权，决定 F_2 中哪个神经元获胜；② $F_1 \leftarrow$

F_2 的外星权, 用作 F_1 输入模式的类别编码。两种抑制信号是指: ① F_1 场神经元的抑制信号, 来自增益控制子系统; ② F_2 场神经元的抑制信号, 来自复位子系统。

从以上分析可知, ART II 与 ART I 的原理类似, 主要区别是 ART II 的比较层 F_1 场的结构与功能更为复杂一些。

4.4.3.2 网络的数学模型与学习算法

(1) 特征表示场 F_1 数学模型 特征表示场 F_1 由三层神经元构成, 底层接受来自外界的输入, 顶层接受来自 F_2 的外星反馈输入, 在中间层对这两种输入进行相应的转换、比较并保存结果, 将输出返回顶层节点及底层节点。

输入模式 X 是一个 M 维模拟向量, 表示为:

$$X = (x_1, x_2, \dots, x_M)$$

在 F_1 中有相应的 M 个处理单元, 每个单元都包括上、中、下三层, 每层都包含有由神经生理学导出的两种不同功能的神经元: 一种用空心圆表示; 另一种用实心圆表示。它们的功能分别为: 空心圆神经元有两种输入激励, 完成的功能是比较其两种不同的输入, 兴奋激励和抑制激励。实心圆神经元完成的功能是求输入向量的模。

在图 4.27 中, F_1 的底层和中层构成一个闭合的正反馈回路, 其中标记为 z_i 的神经元接受输入信号 x_i , 而标记为 v_i 的神经元接受上层送来的信号 $bf(s_i)$ 。这个回路中还包括两次规格化运算和一次非线性变换, 其中底层输入方程和规格化运算为:

$$\begin{cases} z_i = x_i + au_i \\ q_i = z_i / (e + \|Z\|) \end{cases} \quad (4.23)$$

式中, e 为很小的正实数, 相对于 $\|Z\|$ 可以忽略不计。

中层输入方程和规格化运算为:

$$\begin{cases} v_i = f(q_i) + bf(s_i) \\ u_i = v_i / (e + \|V\|) \end{cases} \quad (4.24)$$

式中, e 为很小的正实数, 相对于 $\|V\|$ 可以忽略不计。

从式(4.23) 和式(4.24) 可以看出, 输入模式 X 经历了去噪和归一化处理后成为 U , 其过程为 $x \xrightarrow{+au} z \xrightarrow{/(e + \|Z\|)} q \xrightarrow{f} f(q) \xrightarrow{bf(s)} v \xrightarrow{/(e + \|V\|)} u$ 。

底层至中层和中层至上层之间的非线性变换函数 $f(x)$ 可以采用如下形式:

$$f(x) = \begin{cases} 0, & 0 \leq x < \theta \\ x, & x \geq \theta \end{cases} \quad (4.25)$$

式中, a , b 和 θ 由实验而定。

F_1 的中层和上层也构成一个闭合正反馈回路, 其中标记为 p_i 的神经元接受来自中层的信号 u_i 和来自 F_2 场的信号, 这个回路包括的运算是:

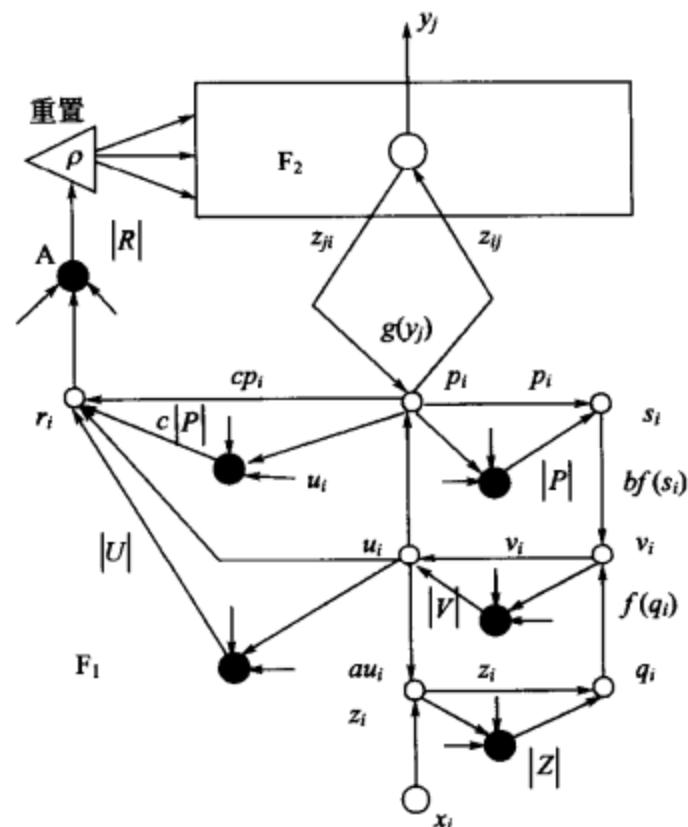


图 4.27 ART II 网络拓扑示意图

$$p_i = u_i + \sum_{j=M+1}^N g(y_j) z_{ji} \quad (4.26)$$

$$s_i = \frac{p_i}{e + \|P\|} \quad (4.27)$$

式(4.26)中第二项是 F_2 场对神经元 p_i 的输入, z_{ji} 是自上而下的 LTM 系数。式(4.27)中 e 为很小的正实数, 相对于 $\|P\|$ 可以忽略不计。

(2) 类别表示场 F_2 的数学模型 类别表示场 F_2 的作用是通过竞争确定最大激活节点, 该节点的权向量与输入模式有最大的相似性。设 F_2 场中第 j 个节点的输入为:

$$T_j = \sum_{i=1}^M p_i z_{ij} \quad j=M+1, \dots, N \quad (4.28)$$

F_2 按下式进行选择:

$$T_{j^*} = \max\{T_j\} \quad j=M+1, \dots, N \quad (4.29)$$

当选择节点 j^* 为最大激活时, 其余节点处于抑制状态, 有:

$$g(y_j) = \begin{cases} d, & j=j^* \\ 0, & j \neq j^* \end{cases} \quad (4.30)$$

式中, d 为自上而下 ($F_1 \rightarrow F_2$) 的反馈参数, $0 < d < 1$ 。由上式, 式(4.26)可简化为:

$$p_i = \begin{cases} u_i + dz_{ji}, & j=j^* \\ u_i, & j \neq j^* \end{cases} \quad (4.31)$$

输入模式 \mathbf{X} 经过 F_1 场后处理后, 其输出 P 是去噪后的归一化输入 U 和某一类模式中心 Z_j 的线性组合。 P 与场间权值 z_{ji} 和 z_{ij} 运算后输入到 F_2 场进行竞争, 场间权值 z_{ji} 和 z_{ij} 是与模式中心密切相关的一些向量。 T_j 的值越高, 则表示输入模式与存储模式间的相似度也越高, 反之表示相似度越低。这种相似度的表示也正是两向量最小距离的一种体现。

(3) 权值调整规则——LTM 方程 对长期记忆 LTM 权值的调整, 按以下两个 LTM 方程进行。自上而下 ($F_1 \rightarrow F_2$) LTM 方程为:

$$\frac{dz_{ji}}{dt} = g(y_j)(p_j - z_{ji}) \quad (4.32)$$

自下而上 ($F_2 \rightarrow F_1$) LTM 方程为:

$$\frac{dz_{ij}}{dt} = g(y_j)(p_j - z_{ij}) \quad (4.33)$$

当 F_2 确定选择 j^* 节点后, 对于 $j \neq j^*$, 有:

$$\frac{dz_{ji}}{dt} = 0 \quad \frac{dz_{ij}}{dt} = 0$$

当 $j=j^*$ 时, 则有:

$$\frac{dz_{ji}}{dt} = d(p_i - z_{j^* i}) = d(1-d)\left(\frac{u_i}{1-d} - z_{j^* i}\right) \quad (4.34)$$

$$\frac{dz_{ij}}{dt} = d(p_i - z_{ij}) = d(1-d)\left(\frac{u_i}{1-d} - z_{ij}\right) \quad (4.35)$$

初始化时, 可取 $z_{ji} = 0$, $z_{ij} = \frac{1}{(1-d)\sqrt{M}}$, $i=1, 2, \dots, M$; $j=M+1, M+2, \dots, M+N$,

$d=0.9$ 。

权值调整公式也可写成以下形式:

$$z_{ij} \cdot (k+1) = z_{ij} \cdot (k) + d(1-d) \left(\frac{u_i(k)}{1-d} - z_{ij} \cdot (k) \right) \quad (4.36)$$

$$z_{j^*i} \cdot (k+1) = z_{j^*i} \cdot (k) + d(1-d) \left(\frac{u_i(k)}{1-d} - z_{j^*i} \cdot (k) \right) \quad (4.37)$$

(4) 取向子系统 图 4.27 中左侧为取向子系统, 其功能是根据 F_1 的短期记忆模式与被激活节点的长期记忆模式之间的匹配度决定 F_2 的重置。匹配度定义为:

$$r_i = \frac{u_i + c p_i}{e + \|U\| + \|cP\|} \quad i=1, 2, \dots, M \quad (4.38)$$

式中, e 可忽略。实心圆 A 的输出为匹配度的模, 用 $\|R\|$ 表示。设警戒门限为 ρ , $0 < \rho < 1$, 当 $\|R\| > \rho$ 时, 选中该类, 否则, 取向子系统需对 F_2 重置。

4.4.4 ART II 型网络的应用

4.4.4.1 ART II 型网络在系统辨识中的应用

控制系统中常将被控对象看作二阶系统, 其性能可用从单位阶跃响应曲线中抽取的特征参数 (如上升时间、调整时间、超调量等) 来描述。一组特征参数构成的特征向量即代表一个二阶对象。ART II 对二阶系统的辨识实际上是对其特征向量进行分类, 系统辨识的实施方案如图 4.28 所示。其中系统模拟器用来对各种二阶系统的单位阶跃响应进行仿真, 特征抽取器从仿真曲线中提取了 6 个特征参数, 送入 ART II 网作为输入模式。ART II 网对输入模式向量的分类是一种有导师学习方式, 其中导师信号来自系统模拟器所仿真的二阶系统数学模型中的参数。二阶系统传递函数的标准形式可写为:

$$G(s) = \frac{\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2}$$

式中, 无阻尼振荡频率 ω_0 和阻尼比 ζ 可唯一确定二阶系统的传递函数。两个系统参数的不同组合可确定多种二阶系统, 每个二阶系统可对应于一组从阶跃响应曲线中抽取的特征参数。

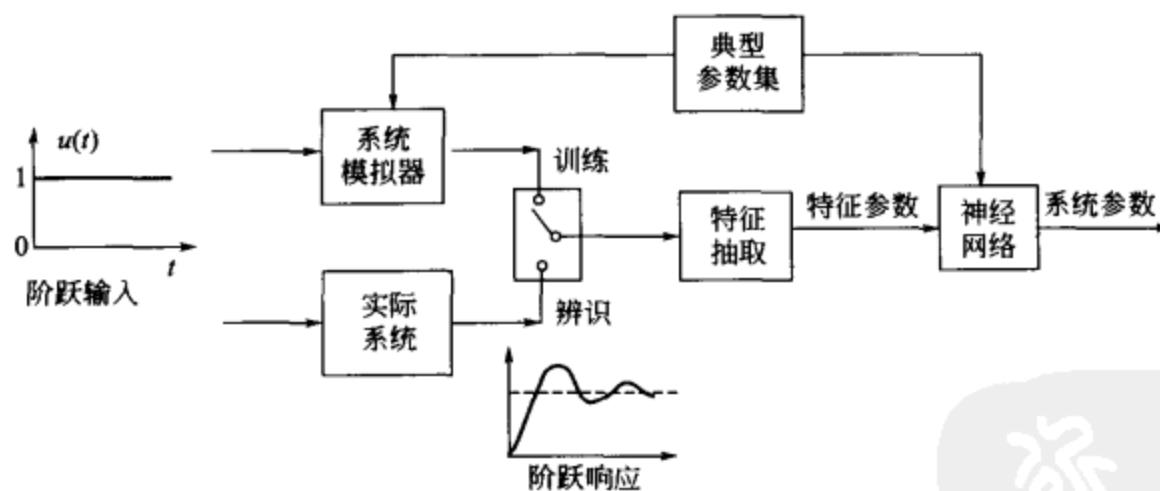


图 4.28 ART II 网系统辨识器

设 $\zeta \in [0.3, 1.3]$, $\omega_0 \in [1.5, 2.0]$, 在该取值范围内列出 17 种传递函数模式供系统模拟器产生阶跃响应曲线。因此该 ART II 网系统辨识器的 F_1 场有 6 个神经元, F_2 场有 17 个神经元。将训练集中的模式依次输入网络进行训练, 网络根据输入的特征向量修改相应的 LTM 权值, 并在 F_2 场指定一神经元作为该模式的代表。训练结束后 F_2 场的每个神经元即代表一种系统特征模式, 当该系统辨识器实际使用时, 来自实际系统的阶跃响应曲线经过特征抽取器后输入系统辨识器激活相应的神经元, 复位子系统对该模式与存储模式类的相似性进行检查, 如大于警戒门限则将其归类, 否则指定一个新神经元代表该模式类。

警戒门限的大小对分类的粗细具有调节作用，本例将其设为 0.99，以保证分类具有较高的分辨力。

4.4.4.2 ARTⅡ型网络在压力容器焊缝超声波探伤中的应用

在压力容器焊缝超声波探伤中，除了确定焊缝中缺陷的位置和大小外，还应尽可能判定缺陷的性质。有些焊缝的缺陷对压力容器的整体性能影响不大，相对于压力容器的用途，并不构成致命的威胁；有一些焊缝的缺陷对压力容器的整体结构而言，却是不能忽视的。目前 A 型超声波探伤仪只能提供缺陷回波的时间和幅度两方面的信息，仅仅根据这两方面的信息很难判定压力容器焊缝缺陷的性质。在实际探伤中常常是根据经验，结合压力容器的加工工艺、缺陷特征、缺陷波形和底波情况来分析焊缝缺陷的性质，这就需要检测人员有较高的材料学知识，并具备一定的探伤经验。传统的探伤方法靠人工识别，人主观因素的干扰非常大，不利于自动化检测的要求，在对自动化生产要求日益提高的情况下，研究自动识别缺陷性质的方法尤显迫切。

(1) 检测系统 检测系统采用全数字式超声波探伤仪对压力容器焊缝缺陷样本进行人工手动扫查。压力容器焊缝构件经超声仪检测后，把回波图形保存在超声仪中，通过数据线传入计算机。专用程序在屏幕上显示原始波形、主缺陷波形、傅里叶变换后的幅频图和相频图。

(2) 检测样本 从压力容器焊缝超声波探伤的缺陷回波中，以及回波的傅里叶变换后的幅值和相位图形中提取能反映缺陷组织结构特征的特征值，应用 ART 网络进行识别。缺陷内含物、缺陷表面的粗糙情况、界面反射率对回波的波形都有重要影响。所选用的样本中包括密集型气孔、夹渣、未焊透、未熔合四种类型的缺陷，其中夹渣类样本数=4，密集型气孔类样本数=3，未焊透类样本数=10，未熔合类样本数=10，共计 27 个样本。

(3) 实验结果 应用上述的方法得到的检测结果如表 4.12 所示。

表 4.12 压力容器焊缝缺陷识别结果

样本类型	样本数量	识别结果		正确识别的数量
夹渣	4	1类	3个	3
		2类	0个	
密集型气孔	3	2类	3个	3
		3类	9个	
未焊透	10	5类	1个	9
		4类	8个	
未熔合	10	5类	1个	8
		6类	1个	

正确识别率达到 85.20%，效果较理想，特别是对于未焊透和气孔区分的效果较好。

4.4.4.3 ARTⅡ型网络在企业综合经济效益评估中的应用

企业经济效益是反映企业素质的重要指标，企业经济效益的评价是评估企业综合实力的重要组成部分。企业综合经济效益评价方法有多指标加权综合指数法、功效系数法和模糊综合评价法等。它们或是将多个指标归结为一个能够反映企业综合经济效益的综合性指标，以作为评价企业综合经济效益高低的依据；或是以统计方法对企业进行分类，找出评价判别函数。企业综合经济效益评价是一个非常复杂的问题，由于企业的经营往往受到市场机制、价格体制、政策导向等诸多因素的影响，使得以上方法均有一定的局限性。

企业的综合评价问题，从本质上讲属于一类模式识别问题，而人脑在这类问题的处理上

有很大的优势。由于人工神经网络模仿了人的思维方式，在涉及认识问题的领域有着显著的优越性，为该问题的解决提供了新途径。下面简述基于 ART 网络的综合评估模型的设计与应用：

(1) 输入数据的预处理 以连续值作为 ART 网络的输入。由于企业评估中使用的各项指标数值上相差很大，不能进行直接比较。因此，在评估前需进行数据的标准化。设用 N 个指标构成的经济效益指标体系来评价 P 个企业的综合经济效益，第 p 个企业的第 i 个指标为 x_i^p ，则对统计指标进行标准化处理的公式为：

$$y_i^p = \frac{x_i^p - \bar{x}_i}{s_i}$$

式中， y_i^p 是 x_i^p 的标准化数据； \bar{x}_i 是未标准化的第 i 个指标平均值； s_i 是未标准化的第 i 个指标的标准差。 \bar{x}_i 和 s_i 计算公式分别为：

$$\bar{x}_i = \frac{1}{P} \sum_{p=1}^P x_i^p$$

$$s_i = \sqrt{\frac{1}{P-1} \sum_{p=1}^P (x_i^p - \bar{x}_i)^2}$$

设 F_1 场的参数 a, b, e, θ 均为 0，采用企业评估指标 X 经标准化处理后的 Y^p 取代 P 作为 F_1 场的输出，上传到 F_2 场参加竞争。

(2) 企业样本间相似性度量的选取 衡量两个企业间相似性的度量很多，如贴近度、绝对值距离、夹角余弦、欧氏距离等。具体采用何种度量，应视具体情况和应用效果来定。以两个企业间的欧氏距离作为企业间相似性的度量，第 p 个企业和第 j 个企业之间的欧氏距离为：

$$d_{pj} = \sqrt{\sum_{i=1}^N (x_i^p - x_i^j)^2} \quad (4.39)$$

式中， N 为经济指标数； x_i^p 为第 p 个企业的第 i 个经济指标； x_i^j 为第 j 个企业的第 i 个经济指标。

(3) 评估模型的训练 以企业的各项经济效益的标准化指标作为评估网络的输入模式，网络的输入节点数等于经济指标数，输出节点数等于企业的类别数，训练过程中随分类数的改变而改变。则网络中从下至上的过程，是计算该输入企业和典型企业之间的相似性，以确定和哪类企业最相近。若用欧式距离度量企业的相似性，式(4.28) 应与式(4.39) 类似，即：

$$T_j = \sqrt{\sum_{i=1}^M (P_i - Z_{ij})^2}$$

获胜节点可由式 $T_j = \min\{T_j\}$, $j=M+1, \dots, N$ 确定。

F_2 场中具有最小输出值的节点为竞争获胜节点，意味着当前输入企业与该节点对应的企业类型最相似。当确定输入企业与某类企业最相似后，并不保证该企业一定属于这类企业，需要通过计算输入企业与最相似企业类型之间的距离 d 并与警戒门限值 ρ 比较而定。 $d < \rho$ 表明该输入企业可以归类为获胜节点代表的企业类型，并对该节点对应的内星权值进行修正。 $d > \rho$ 则应另分出一种企业类型，即在 F_2 场增加一个节点，新增节点所属的连接权应赋值为该输入企业的各项经济指标。ART 网络对所有企业样本进行训练后， F_2 场节点的

数目反映了企业的类别数，每个节点对应的内星权向量，是某类企业的代表值。每个待分类企业的经济指标与这些代表值进行比较后，决定其属于哪一种企业类别。

(4) 警戒门限值 ρ 的选取 ρ 值决定了分类的精细程度，识别层节点的数目（即企业的分类数）与 ρ 值大小关系密切。 ρ 值的选取可考虑两点：一是对企业分类的要求；二是在选择某 ρ 值后的分类情况与实际情况的吻合程度。如按第二点来考虑，就要依据分类情况与实际情况的吻合程度来调整 ρ 值。这种吻合程度的判断受到人们主观的影响，因此可以认为 ρ 值的选取是有监督训练的。通过不断地调整 ρ 值，并利用已有的数据对网络进行训练，直到得出满意的分类为止。

(5) 实测结果分析 对某行业的 17 家企业进行评估实测。首先将 17 家企业的经济效益指标（见表 4.13）进行标准化处理，将标准化数据作为 ART 网络的输入。在训练中不断调整 ρ 值，使其对企业的分类达到要求的类数，结果见表 4.14。可以看出，当值 ρ 改变时，分类数由 5 类变为 2 类。

表 4.13 17 家企业经济效益指标

序号	人均利润/(元/人)	资金利税率/%	成本利税率/%	产值利税率/%
1	4339.83	0.192	0.249	0.268
2	1095.57	0.143	0.341	0.142
3	4309.78	0.159	0.259	0.299
4	4590.74	0.290	0.400	0.289
5	2310.38	0.203	0.201	0.180
6	1268.81	0.129	0.167	0.175
7	1548.86	0.173	0.447	0.159
8	1179.62	0.174	0.135	0.140
9	2146.06	0.159	0.147	0.139
10	4429.63	0.356	0.249	0.216
11	1853.62	0.097	0.149	0.145
12	4504.59	0.270	0.306	0.299
13	1543.92	0.098	0.158	0.136
14	1372.54	0.186	0.175	0.194
15	4381.41	0.320	0.396	0.335
16	2377.62	0.255	0.224	0.202
17	1080.46	0.178	0.221	0.146

表 4.14 企业分类数与 ρ 值的关系

ρ 值	分类数	分类结果
1.8	5	(1,3), (2), (4,12,15), (10), (5,6,7,8,9,11,13,14,16,17)
2.2	4	(1,3), (4,12,15), (10), (2,5,6,7,8,9,11,13,14,16,17)
2.4	2	(1,3,4,10,12,15), (2,5,6,7,8,9,11,13,14,16,17)

可以看出，本例采用的网络直接用标准化的输入模式与 F_2 场的内星权向量比较相似性，并未涉及自上而下的外星权值，因此是 ART II 网络的一种变形。

本章小结

本章介绍了两种采用无导师学习方式的自组织神经网络。自组织神经网络最重要的特点是通过自动寻找样本中的内在规律和本质属性，自组织、自适应地改变网络参数与结构。这种学习方式大大拓宽了神经网络在模式识别与分类方面的应用。自组织的共同特点是都具有竞争层。输入层负责接受外界信息并将输入模式向竞争层传递，起“观察”作用，竞争层负责对该模式进行“分析比较”，找出规律以正确归类。这种功能是通过竞争机制实现的。本

章学习要点是：

(1) 竞争学习策略 竞争学习是自组织网络中最常采用的一种学习策略，胜者为王是竞争学习的基本算法。该算法将输入模式向量同竞争层所有节点对应的权向量进行比较，将欧式距离最小的判为竞争获胜节点，并仅允许获胜节点调整权值。按照胜者为王的竞争学习算法训练竞争层的结果必将使各节点的权向量成为输入模式的，称聚类中心。

(2) SOM 神经网络 SOM 网络模型中的竞争机制具有生物学基础，该模型模拟了人类大脑皮层对于某一图形或某一频率等输入模式的特定兴奋过程。SOM 网的结构特点是输出层神经元可排列成线阵或平面阵。在网络训练阶段，对某个特定的输入模式，输出层会有某个节点产生最大响应而获胜。获胜节点周围的节点因侧向相互兴奋作用也产生较大响应，于是获胜节点及其优胜邻域内的所有节点所连接的权向量均向输入向量的方向作程度不同的调整，调整力度依邻域内各节点距获胜节点的远近而逐渐衰减。网络通过自组织方式，用大量训练样本调整网络的权值，最后使输出层各节点成为对特定模式类敏感的神经细胞，对应的内星权向量成为各输入模式类的中心向量。并且当两个模式类的特征接近时，代表这两类的节点在位置上也接近，从而在输出层形成能够反映样本模式类分布情况的有序特征图。

(3) ART 神经网络 ART 网络有三种类型，本章介绍了 I 型和 II 型。ART I 网络的运行分为 4 个阶段：①匹配阶段接受来自环境的输入模式，并在输出层与所有存储模式类进行匹配竞争，产生获胜节点；②比较阶段按参考门限检查该输入模式与获胜模式类的典型向量之间的相似程度，相似度达标进入学习阶段，不达标则进入搜索阶段；③搜索阶段对相似度不超过参考门限的输入模式重新进行模式类匹配，如果与所有存储模式类的匹配不达标，就需要在网络中设立一个新的模式类，同时建立与该模式类相连的权值，用以代表和存储该模式以及后来输入的所有同类模式；④学习阶段对相似度超过参考门限的所有模式类，选择最相似的作为该模式的代表类，并调整与该类别相关的权值，以使以后与该模式相似的输入再与该模式匹配时能得到更大的相似度。ART II 网络的运行原理与 ART I 相似，主要区别在识别层。ART 网络及算法在适应新的输入模式方面具有较大的灵活性，同时能够避免对网络先前所学模式的修改。

思考与练习

- 4.1 在自组织神经网络中，“自组织”的含义是什么？
- 4.2 自组织特征映射网中竞争机制的生物学基础是什么？
- 4.3 试述 ART I 网络的运行原理。
- 4.4 试比较 ART I 网与 ART II 网的异同。
- 4.5 自组织网由输入层与竞争层组成，初始权向量已归一化为：

$$\hat{\mathbf{w}}_1(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \hat{\mathbf{w}}_2(0) = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

设训练集中共有 4 个输入模式，均为单位向量：

$$\{\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3, \mathbf{X}^4\} = \{1 \angle 45^\circ, 1 \angle -135^\circ, 1 \angle 90^\circ, 1 \angle -180^\circ\}$$

试用胜者为王学习算法调整权值，写出迭代一次的调整结果。

4.6 设图 4.1 中的竞争层有 3 个节点，试用胜者为王学习算法训练权向量，使其能对输入模式 C、I 和 T 正确分类，给出训练后的权向量 \mathbf{W}^C 、 \mathbf{W}^I 和 \mathbf{W}^T 。用含噪声样本测试该网络，给出

分类结果。

提示：设含噪声样本与正确样本的海明距离为1。两个向量的海明距离 $dH(\mathbf{X}^a, \mathbf{X}^b)$ 是指两个向量中不相同元素的个数。

4.7 采用胜者为王学习算法训练一个竞争网络，将下面的输入模式分为两类：

$$\mathbf{X}^1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad \mathbf{X}^2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{X}^3 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

① $\eta=0.5$ ，初始权值矩阵为：

$$\mathbf{W} = \begin{bmatrix} \sqrt{2} & 0 \\ 0 & \sqrt{2} \end{bmatrix}$$

将输入模式按顺序训练一遍并图示训练结果，观察模式如何聚类。

② 如果输入模式的顺序改变，训练结果是否改变？请解释原因。

③ 令 $\eta=0.25$ ，重做①，这种改变对训练有何影响？

4.8 给定5个4维输入模式如下：

$$\mathbf{X}^1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{X}^2 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{X}^3 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \mathbf{X}^4 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{X}^5 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

试设计一个具有 5×5 神经元平面阵的 SOM 网，建议学习率 $\eta(t)$ 在前 1000 步训练中从 0.5 线性下降至 0.04，然后在训练到 10000 步时减小至 0。优胜邻域半径初值设为 2 个节点（即优胜邻域覆盖整个输出平面），1000 个训练步时减至 0（即只含获胜节点）。每训练 200 步保留一次权向量，观察其在训练过程中的变化。给出训练结束后，5 个输入模式在输出平面的映射图。

4.9 SOM 网有 15 个输出神经元，二维输入样本点均匀分布在一个三角形区域。训练后的权值分布如图 4.29 所示，图中各点的连线仅表明权值的相邻关系。试根据自己的理解给出该 SOM 网的结构，以及神经元的排列情况。

4.10 设计一个 ART I 网对下面给出的 3 个输入模式进行分类。设计一个合适的警戒门限，使得 ART I 网能将 3 个输入模式分为 3 类。写出训练的

前 3 步结果，及训练结束后的 \mathbf{B} 阵和 \mathbf{T} 阵。3 个输入模式为：

$$\mathbf{X}^1 = (1, 0, 0, 0, 1, 0, 0, 0, 1)^T$$

$$\mathbf{X}^2 = (1, 1, 0, 0, 1, 0, 0, 1, 1)^T$$

$$\mathbf{X}^3 = (1, 0, 1, 0, 1, 0, 1, 0, 1)^T$$

4.11 设计一个 ART I 网对本章 4.4.2.1 和 4.4.2.2 中的数据进行训练，并将训练结果同表 4.9 进行对照。

4.12 用下面的输入模式训练 ART I 网，警戒门限分别为 $\rho=0.3$, $\rho=0.6$, $\rho=0.9$ 。

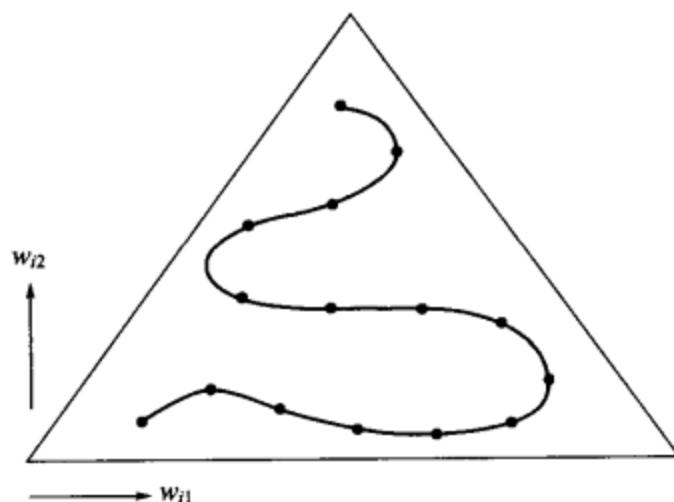


图 4.29 习题 4.9 附图

5 组合学习神经网络

有教师信号的监督学习和无教师信号的自组织竞争学习是两类各具特色的学习方式，若将两类方式有机结合，则可以充分发挥两者的优势，使所设计的神经网络具有更强的功能。本章讨论两种组合学习方式的神经网络，其特点是网络的输出层采用监督学习算法而隐层采用竞争学习策略。

5.1 学习向量量化神经网络

学习向量量化 (learning vector quantization, LVQ) 网络是在竞争网络结构的基础上提出的，LVQ 网络将竞争学习思想和有监督学习算法相结合，在网络学习过程中，通过教师信号对输入样本的分配类别进行规定，从而克服了自组织网络采用无监督学习算法带来的缺乏分类信息的弱点。

5.1.1 向量量化

在信号处理领域，量化是针对标量进行的，指将信号的连续取值或者大量可能的离散取值近似为有限多个或较少的离散值的过程。向量量化是对标量量化的扩展，适用于高维数据。向量量化的思路是，将高维输入空间分成若干不同的区域，对每个区域确定一个中心向量作为聚类中心，与其处于同一区域的输入向量可用该中心向量来代表，从而形成以各中心向量为聚类中心的点集。在图像处理领域，常用各区域中心点（向量）的编码代替区域内的点来存储或传输，从而提出了各种基于向量量化的有损压缩技术。

在二维输入平面上表示的中心向量分布称为 Voronoi 图，如图 5.1 所示。前面介绍的 Winner-Take-All 和 SOFM 竞争学习算法都是向量量化算法，都能用少量聚类中心表示原始数据，从而起到数据压缩作用。但 SOFM 的各相邻聚类的中心向量具有某种相似的特征，而一般向量量化的中心不具有这种相似性。

自组织映射可以起到聚类的作用，但还不能直接分类或识别，因此这只是自适应解决模式分类问题两步中的第一步。第二步是学习向量量化，采用有监督方法，在训练中加入教师信号作为分类信息对权值进行细调，并对输出神经元预先指定其类别。

5.1.2 LVQ 网络结构与工作原理

LVQ 网络的结构如图 5.2 所示，由输入层、竞争层和输出层神经元组成。输入层有 N

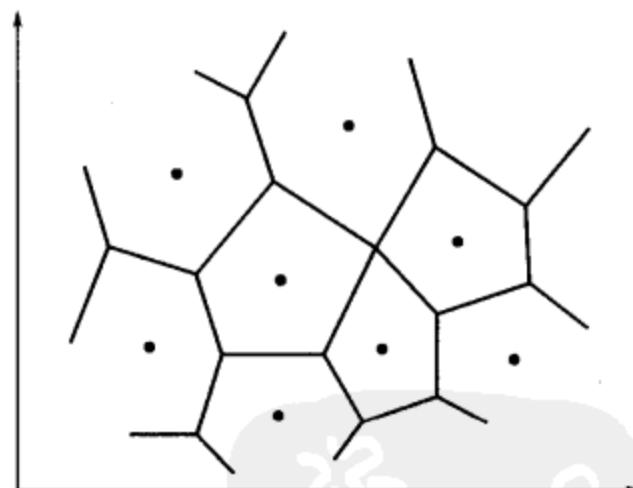


图 5.1 二维向量量化

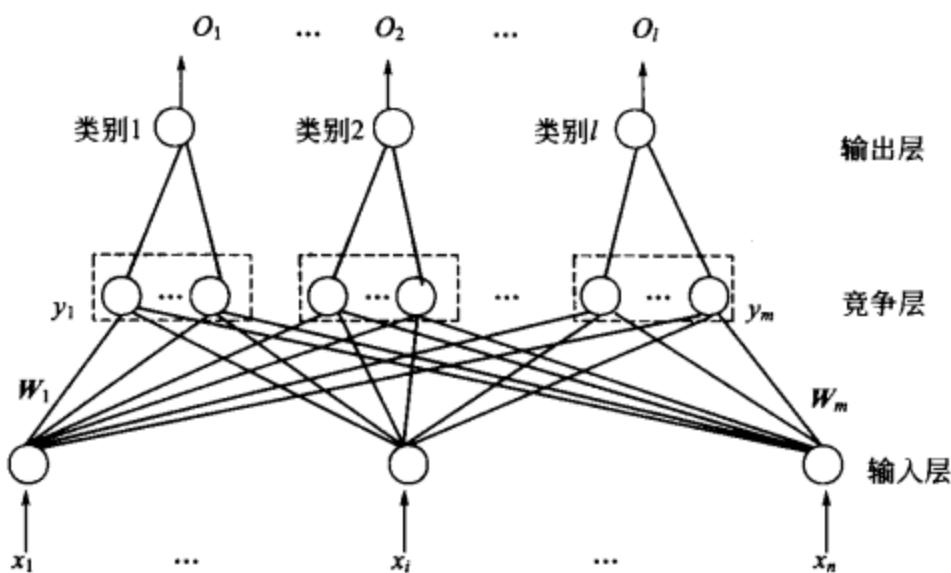


图 5.2 学习向量量化网络

一个神经元接受输入向量，与竞争层之间完全连接；竞争层有 M 个神经元，分为若干组并呈一维线阵排列；输出层每个神经元只与竞争层中的一组神经元连接，连接权值固定为 1。在 LVQ 网络的训练过程中，输入层和竞争层之间的连接权值被逐渐调整为聚类中心。当一个输入样本被送至 LVQ 网时，竞争层的神经元通过胜者为王竞争学习规则产生获胜神经元，容许其输出为 1，而其他神经元输出均为 0。与获胜神经元所在组相连接的输出神经元其输出也为 1，而其他输出神经元输出为 0，从而给出当前输入样本的模式类。将竞争层学习得到的类称为子类，将输出层学习得到的类称为目标类。

LVQ 网络各层的数学描述如下：设输入向量用 \mathbf{X} 表示：

$$\mathbf{X} = (x_1, x_2, \dots, x_N)^T$$

竞争层的输出用 \mathbf{Y} 表示：

$$\mathbf{Y} = (y_1, y_2, \dots, y_m)^T, \quad y_j \in \{0, 1\}, \quad j = 1, 2, \dots, M$$

输出层的输出用 \mathbf{O} 表示：

$$\mathbf{O} = (o_1, o_2, \dots, o_l)^T$$

网络的期望输出用 \mathbf{d} 表示：

$$\mathbf{d} = (d_1, d_2, \dots, d_l)^T$$

输入层到竞争层之间的权值矩阵用 \mathbf{W}^1 表示：

$$\mathbf{W}^1 = (\mathbf{W}_1^1, \mathbf{W}_2^1, \dots, \mathbf{W}_j^1, \dots, \mathbf{W}_M^1)$$

其中列向量 \mathbf{W}_j^1 为隐层第 j 个神经元对应的权值向量。

竞争层到输出层之间的权值矩阵用 \mathbf{W}^2 表示：

$$\mathbf{W}^2 = (\mathbf{W}_1^2, \mathbf{W}_2^2, \dots, \mathbf{W}_k^2, \dots, \mathbf{W}_l^2)$$

其中列向量 \mathbf{W}_k^2 为输出层第 k 个神经元对应的权值向量。

5.1.3 LVQ 网络的学习算法

LVQ 网络的学习规则结合了竞争学习和有导师学习规则，需要一组有教师信号的样本对网络进行训练。设有训练样本集： $\{(x^1, d^1), \dots, (x^p, d^p), \dots, (x^P, d^P)\}$ ，其中每个教师向量 d^p ($p=1, 2, \dots, P$) 中只有一个分量为 1，其他分量均为 0。通常把竞争层的每一个神经元指定给一个输出神经元，相应的权值为 1，从而得到输出层的权值矩阵 \mathbf{W}^2 。例如，某 LVQ 网络竞争层有 6 个神经元，输出层有 3 个神经元，代表 3 个类。若将竞争层的 1、3 号神经元指定为第 1 个输出神经元，第 2、5 号神经元指定为第 2 个输出神经元，第 4、6 号神

经元指定为第 3 个输出神经元。则权值矩阵 \mathbf{W}^2 定义为：

$$\mathbf{W}^2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

\mathbf{W}^2 的列表示类，行表示子类，每一行只有一个元素为 1，该元素所在的列表示这个子类所属的类。对任一输入样本，网络的输出为：

$$\mathbf{O} = (\mathbf{W}^2)^T \mathbf{Y}$$

LVQ 网络在训练前预先定义好 \mathbf{W}^2 ，从而指定了输出神经元类别。训练中 \mathbf{W}^2 不再改变，网络的学习是通过改变 \mathbf{W}^1 来进行的。根据输入样本类别（教师信号）和获胜神经元所属类别，可判断当前分类是否正确。如图 5.3 所示，若分类正确则将获胜神经元的权向量向输入向量方向调整，分类错误则向相反方向调整。

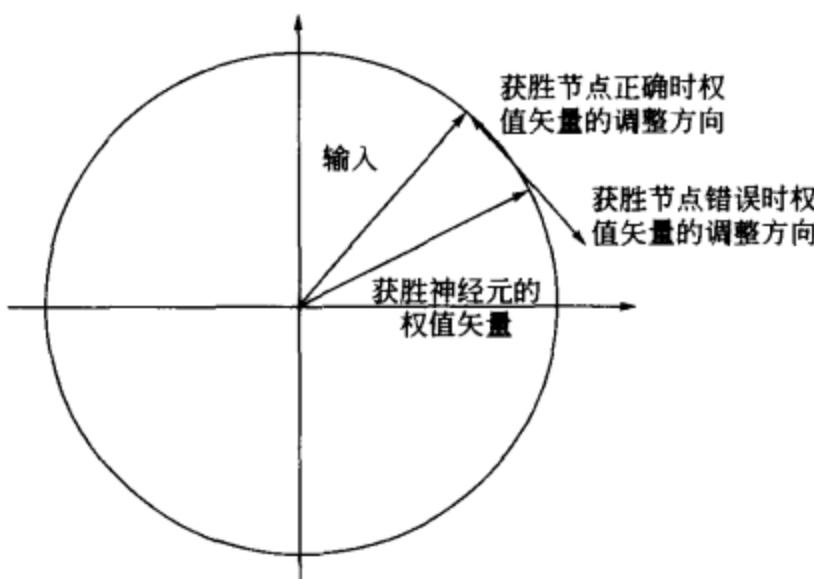


图 5.3 学习向量量化的权值调整

LVQ 网络学习算法的步骤如下：

- (1) 初始化 竞争层各神经元权值向量 \mathbf{W}_j^1 , $j=1, 2, \dots, M$ 赋小随机数，确定初始学习速率 $\eta(0)$ 和训练次数 K 。
- (2) 输入样本向量 \mathbf{X} 。
- (3) 寻找获胜神经元 j^* :

$$\|\mathbf{X} - \mathbf{W}_j^1\| = \min_j \|\mathbf{X} - \mathbf{W}_j^1\| \quad j=1, 2, \dots, M$$

- (4) 根据分类是否正确按不同规则调整获胜神经元的权值。当网络分类结果与教师信号一致时，向输入样本方向调整权值：

$$\mathbf{W}_j^1(k+1) = \mathbf{W}_j^1(k) + \eta(k)[\mathbf{X} - \mathbf{W}_j^1(k)] \quad (5.1)$$

否则，将逆输入样本方向调整权值：

$$\mathbf{W}_j^1(k+1) = \mathbf{W}_j^1(k) - \eta(k)[\mathbf{X} - \mathbf{W}_j^1(k)] \quad (5.2)$$

其他非获胜神经元的权值保持不变。

(5) 更新学习速率

$$\eta(k) = \eta(0) \left(1 - \frac{k}{K}\right) \quad (5.3)$$

当 $k < K$ 时, $k = k + 1$, 转到步骤 (2) 输入下一个样本, 重复各步骤直到 $k = K$ 。

在上述训练过程中, 须保证 $\eta(k)$ 为单调下降函数。此外, 寻找获胜神经元时直接用最小欧式距离进行判断, 因此不需要对权值向量和输入向量进行归一化处理。

LVQ 网络是 SOFM 网络的一种有监督形式的扩展, 两者有效结合可更好地发挥竞争学习和有监督学习的优点。

5.1.4 LVQ 网络的设计与应用

5.1.4.1 LVQ 网络在证券投资基金分类中的应用

截至 2004 年底, 我国的证券投资基金已发展到 154 只。面对不断增加的基金, 基金投资者的选择范围和选择的难度也越来越大。准确的基金分类可以帮助投资者将其资金分配到不同类别的基金中, 以期达到分散风险的效果。采用基金市场表现的相关指标对基金进行分类, 主要采多元统计方法, 分类准确率不高。神经网络在解决分类问题时具有独特的优势, 下面介绍一种将 SOM 网络与 LVQ 网络结合起来对基金进行分类的方法:

(1) 基金分类模型设计 训练样本来自我国 2003 年 10 月 1 日以前发行的 54 只开放式证券投资基金, 研究区间为 2003 年第四季至 2005 年第 1 季度。训练样本取考察区间中基金表现最优、中等和最差的三个季度共 162 个样本。

① 应用 SOM 网络对全部样本进行聚类分析 设计 SOM 网络竞争层节点数时须知道类别数, 若无先验知识, 可采用测试法确定。令类别数为 2~10, 用 SOM 网络分别进行聚类, 通过比较认为类别数为 4 时, 分类结果最满意。分类结果具体情况见表 5.1。根据各类基金的指标值, 可以进行相应的命名: 第一类的平均收益率为负值且詹森指数很小, 命名为“低绩效基金”; 第二类基金 β 系数及标准都较小, 说明受市场波动影响极小, 命名为“低风险基金”; 第三类基金的各指标均处第二位, 略优于所有基金的平均值, 命名为“稳健基金”; 第四类基金的平均收益率及詹森指数都最大, 命名为“高绩效基金”。

表 5.1 SOM 神经网络分类结果

指标(均值)	第一类 (34 个)	第二类 (41 个)	第三类 (35 个)	第四类 (52 个)
平均收益率/%	-6.03	1.01	2.03	10.03
β 系数	0.563	0.129	0.585	0.555
标准差	0.826	0.260	1.323	0.962
詹森指数	0.005	0.006	0.035	0.094
命名	低绩效基金	低风险基金	稳健基金	高绩效基金

② 使用 LVQ 网络对聚类结果进行判别并预测 从 162 个样本取出 100 个作为训练样本, 其余 62 个为测试样本。当网络训练 200 次以后, LVQ 网络分类的误差趋于稳定, 降至 0.02。利用训练好的 LVQ 网络对 62 个为测试样本进行预测的结果为: 共有 2 个训练样本被错误归到相邻类中, 准确率为 98%; 共有 3 个测试样本被错误归到相邻类中, 准确率为 95.16%。全部样本预测的准确率为 96.91%, 表明利用神经网络进行基金业绩分类可取得较好的效果。

(2) 基金分类情况分析 利用训练好的神经网络模型分别对 54 只开放式基金连续五个

季度所属类别进行计算，整理结果见表 5.2。经分析可以得到以下结论：

① 开放式基金的业绩波动幅度较大、持续性差 在研究区间内，约有 60% 的基金其业绩跨越 3 个以上的类别。只有 6 只基金（占 11.11%）保持所属的类别不变，这些基金分别是 5 只低风险基金和 1 只高绩效基金。

表 5.2 开放基金业绩波动情况

所属类别数	基金数量	比例/%	所属类别数	基金数量	比例/%
1	6	11.11%	3	31	57.41%
2	15	27.78%	4	2	3.70%

② 以债券为投资对象的债券型基金市场表现平稳 所考察的 9 只债券型基金有 5 只基金始终属于第二类型基金，其他 4 只基金只在两个类型之间变换，这相对整个开放型基金业是很平稳的。

③ 我国的基金业投资逐渐趋于理性 虽然我国开放式基金的业绩波动幅度较大，每个季度所属类别发生变化的基金还占 50% 以上，但可以看到，随着理性投资理念的逐渐建立，所属类别发生变化的基金数目趋于减少，具体情况见图 5.4。这说明，我国的开放式基金正在从极不稳定的变化中走向理性发展。

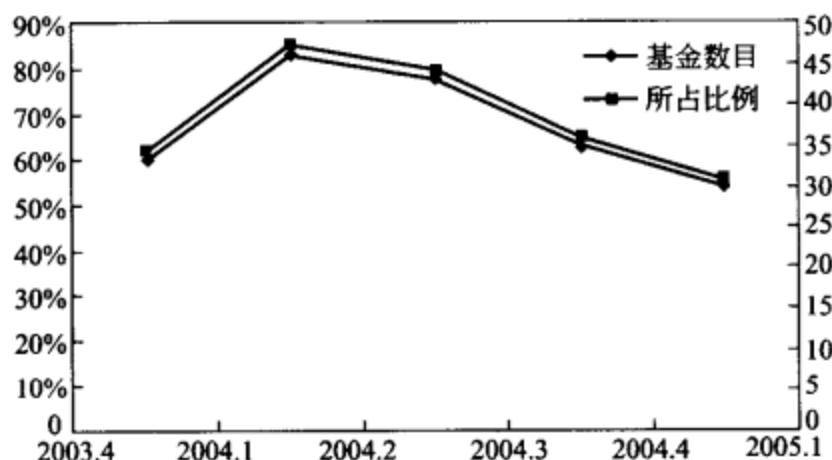


图 5.4 五个季度中类别发生变化的基金的数目及比例

5.1.4.2 LVQ 网络在探地雷达探雷中的应用

探地雷达作为非破坏性探测手段正被广泛应用于地下目标（如空洞、管道、地雷等）的探测。如何对雷达回波信号进行处理以识别地下埋设的目标始终是困扰探地雷达应用的难题。目前主要依赖于成像技术，处理结果一般由人工加以解释，该方法不仅无法进行实时处理，同时也对操作者的技术经验提出了很高的要求。针对上述问题，用信号的学习向量化神经网络对目标进行分类，取得了较好的效果。处理过程如图 5.5 示。

(1) 预处理与特征提取 数据集合来自中心频率为 1GHz 的无载频脉冲探地雷达系统，试验对象为反步兵地雷、金属盒及砖块，试验介质分别为沙滩和中性土壤，埋深分别为 20cm 及 40cm。原始信号经过延时校正、滤波及减背景操作后，突出了目标信号，提高了信噪比。信号的特征提取可以从时域、频域、自相关、功率谱、双谱、小波等方面考虑，这里采用谱估计理论来提取目标信号的特征。由于雷达信号是非平稳、非周期且非瞬态的信号，不能直接使用傅里叶变换，可优先使用部分扫描的 Welch 平均重

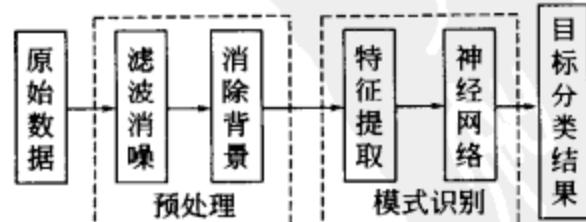


图 5.5 探地雷达处理流程图

叠周期谱对信号谱计算，该谱估计方法对区分目标和对目标进行定位非常有效。

(2) LVQ 网络的设计 训练样本为 3360 个，测试样本为 240 个。将谱估计的结果进行 K-L 变换，将信号从 64 维降低到 16 维，从而缩小了样本空间，有效降低了神经网络学习的计算量。整个神经网络由输入层、竞争层和线性输出层组成，输入层包括了 16 个神经元，对应于每个谱值；竞争层的神经元个数选为 16 个；输出层包含 3 个神经元，分别对应于反步兵地雷、砖块和金属盒三种目标。

(3) LVQ 网络的训练和测试 训练前先将训练集与测试集样本归一化，经过训练后的 LVQ 网就可以用来对探地雷达信号中提取的浅表目标进行测试分类。表 5.3 列出了训练好的 LVQ 网络对三种不同介质中的反步兵地雷目标进行识别的结果。从测试结果可以看出，介质均匀，会导致识别率的降低，其主要原因是由于干扰增多。

表 5.3 不同介质下反步兵地雷神经网络测试结果

介质	测试一	测试二	平均识别率
	识别次数/测试次数	识别次数/测试次数	
沙滩介质	32/40	52/60	84%
沙土介质	25/40	38/60	63%
黏土介质	17/40	19/60	36%

5.2 对向传播神经网络

1987 年，美国学者 Robert Hecht-Nielsen 提出了对向传播神经网络模型（counter propagation network，缩写为 CPN），最早是用来实现样本选择匹配系统的。CPN 网能存储二进制或模拟值的模式对，因此这种网络模型也可用作联想存储、模式分类、函数逼近、统计分析和数据压缩等用途。

5.2.1 网络结构与运行原理

图 5.6 给出了对向传播网络的标准三层结构，各层之间的神经元通过全互连连接。从拓扑结构看，CPN 网与三层 BP 网没有什么区别，但实际上它是由 Kohonen 的自组织网和

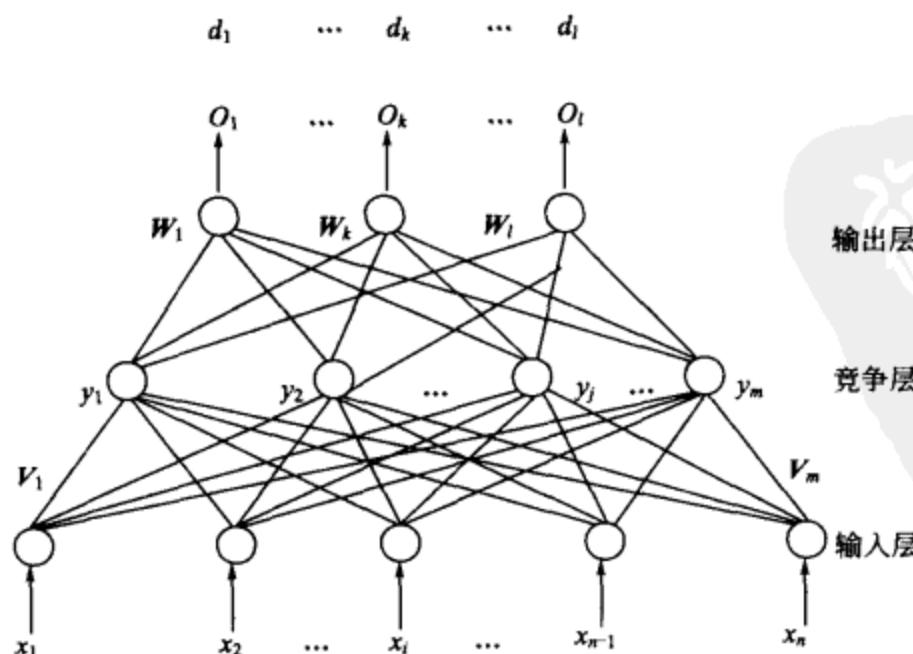


图 5.6 CPN 网的拓扑结构

Grossberg 的外星网组合而成的。其中隐层为 Kohonen 网的竞争层，该层的竞争神经元采用无导师的竞争学习规则进行学习，输出层为 Grossberg 层，它与隐含层全互连，采用有导师的 Widrow-Hoff 规则或 Grossberg 规则进行学习。

网络各层的数学描述如下：

设输入向量用 \mathbf{X} 表示：

$$\mathbf{X} = (x_1, x_2, \dots, x_n)^T$$

竞争结束后竞争层的输出用 \mathbf{Y} 表示：

$$\mathbf{Y} = (y_1, y_2, \dots, y_m)^T, \quad y_i \in \{0, 1\}, \quad i=1, 2, \dots, m$$

网络的输出为 \mathbf{O} 表示：

$$\mathbf{O} = (o_1, o_2, \dots, o_l)^T$$

网络的期望输出用 \mathbf{d} 表示：

$$\mathbf{d} = (d_1, d_2, \dots, d_l)^T$$

输入层到竞争层之间的权值矩阵用 \mathbf{V} 表示：

$$\mathbf{V} = (V_1, V_2, \dots, V_j, \dots, V_m)$$

其中列向量 V_j 为隐层第 j 个神经元对应的内星权向量。竞争层到输出层之间的权值矩阵用 \mathbf{W} 表示：

$$\mathbf{W} = (W_1, W_2, \dots, W_k, \dots, W_l)$$

其中列向量 W_k 为输出层第 k 个神经元对应的权向量。

网络各层按两种学习规则训练好之后，运行阶段首先向网络送入输入向量，隐含层对这些输入进行竞争计算。若某个神经元的净输入值为最大则竞争获胜，成为当前输入模式类的代表，同时该神经元成为图 5.7(a) 所示的活跃神经元，输出值为 1；而其余神经元处于非活跃状态，输出值为 0。

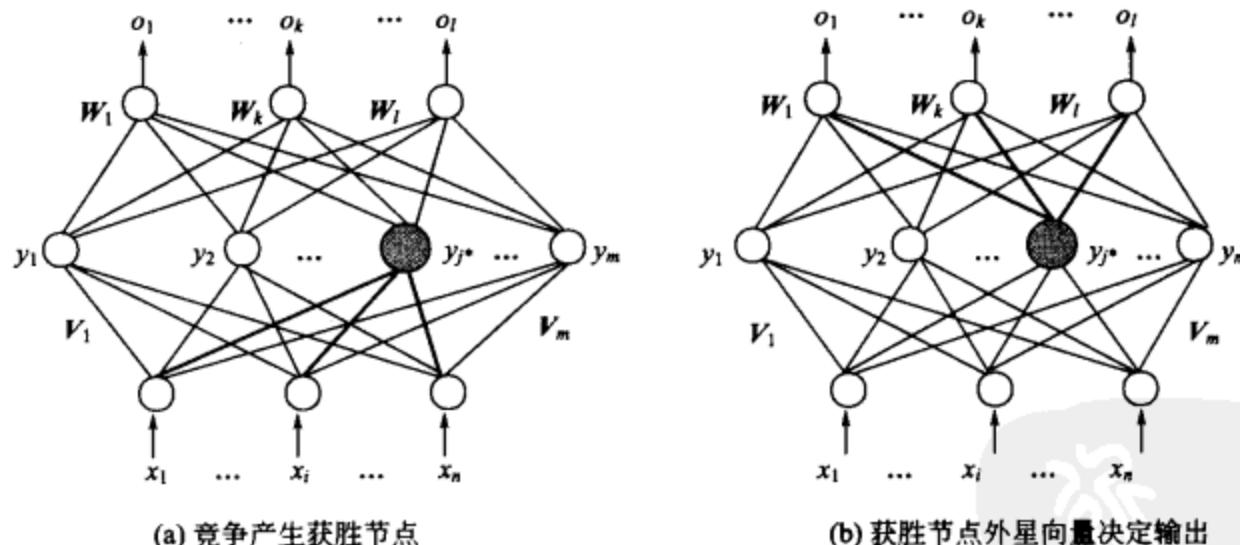


图 5.7 CPN 网运行过程

竞争取胜的隐含神经元激励输出层神经元，使其产生如图 5.7(b) 所示的输出模式。由于竞争失败的神经元输出值为 0，故它们在输出层神经元的净输入中没有贡献，不影响其输出值。因此，输出就由竞争胜利的神经元所对应的外星向量来确定。

5.2.2 CPN 的学习算法

网络的学习规则由无导师学习和有导师学习组合而成，因此训练样本集中输入向量与期望输出向量应成对组成，即： $\{\mathbf{X}^p, \mathbf{d}^p\}$ ， $p=1, 2, \dots, P$ ， P 为训练集中的模式总数。

训练分为两个阶段进行，每个阶段采用一种学习规则。第一阶段用竞争学习算法对输入层至隐层的内星权向量进行训练，步骤如下：

- (1) 将所有内星权随机地赋以 0~1 之间的初始值，并归一化为单位长度，得 \hat{V} ；训练集内的所有输入模式也要进行归一化，得 \hat{X} 。
- (2) 输入一个模式 X^p ，计算净输入 $net_j = \hat{V}_j^T \hat{X}$, $j=1, 2, \dots, m$ 。
- (3) 确定竞争获胜神经元 $net_{j^*} = \max_j \{net_j\}$ ，使 $y_{j^*} = 1$, $y_j = 0$, $j \neq j^*$ 。
- (4) CPN 网络的竞争算法不设优胜邻域，因此只调整获胜神经元的内星权向量，调整规则为：

$$W_{j^*}(t+1) = \hat{W}_{j^*}(t) + \eta(t)[\hat{X} - \hat{W}_{j^*}(t)] \quad (5.4)$$

式中， $\eta(t)$ 为学习率，是随时间下降的退火函数。由以上规则可知，调整的目的是使权向量不断靠近当前输入模式类，从而将该模式类的典型向量编码到获胜神经元的内星权向量中。

- (5) 重复步骤 (2) 至步骤 (4) 直到 $\eta(t)$ 下降至 0。需要注意的是，权向量经过调整后必须重新作归一化处理。

第二阶段采用外星学习算法对隐层至输出层的外星权向量进行训练，步骤如下：

- (1) 输入一个模式对 X^p, d^p ，计算净输入 $net_j = \hat{V}_j^T \hat{X}$, $j=1, 2, \dots, m$ ，其中输入层到隐层的权值矩阵保持第一阶段的训练结果。
- (2) 确定竞争获胜神经元 $net_{j^*} = \max_j \{net_j\}$ ，使：

$$y_j = \begin{cases} 0, & j \neq j^* \\ 1, & j = j^* \end{cases} \quad (5.5)$$

- (3) 调整隐层到输出层的外星权向量，调整规则为：

$$W_{jk}(t+1) = W_{jk}(t) + \beta(t)[d_k - o_k(t)] \quad j=1, 2, \dots, m; k=1, 2, \dots, l \quad (5.6)$$

式中， $\beta(t)$ 为外星规则的学习率，也是随时间下降的退火函数； o_k 是输出层神经元的输出值，由下式计算：

$$o_k(t) = \sum_{j=1}^l w_{jk} y_j \quad (5.7)$$

由式(5.5)，上式应简化为：

$$o_k(t) = w_{j^*k} y_{j^*} = w_{j^*k} \quad (5.8)$$

将上式代入式(5.6)，得外星权向量调整规则如下：

$$w_{jk}(t+1) = \begin{cases} w_{jk}(t), & j \neq j^* \\ w_{jk}(t) + \beta(t)[d_k - w_{jk}(t)], & j = j^* \end{cases} \quad (5.9)$$

由以上规则可知，只有获胜神经元的外星权向量得到调整，调整的目的是使外星权向量不断靠近并等于期望输出，从而将该输出编码到外星权向量中。

- (4) 重复步骤 (1) 至步骤 (3) 直到 $\beta(t)$ 下降至 0。

同目前应用最为广泛的误差反向传播网络 (back propagation network, BPN) 比较，CPN 由于采用了混合学习方式，其优点在于收敛速度快、泛化能力强，但同时继承了 SOM 的缺陷。

5.2.3 改进的 CPN 网举例

5.2.3.1 双获胜节点 CPN 网

在标准的对向传播网络中，竞争层上只允许有一个神经元获胜。作为一种改进方案，在完成训练后的运行期间允许隐层有两个神经元同时获得竞争胜利，这两个获胜神经元均取值为1，其他的神经元则取值为0。于是两个获胜神经元同时按照式(5.8)影响网络的输出。

图5.8给出了这种情况的一个例子。其中图5.8(a)表示3个训练样本对，在图5.8(b)中利用这些样本对CPN网进行训练。训练完成后，作为标准CPN网运行时，对于每个输入模式只允许有一个隐层神经元获胜，因此送入该网络一个输入模式，网络就以一个对应的输出模式来响应。但当作为改进CPN网运行时，对于每个输入模式允许有两个隐层神经元同时获胜，此时若给网络送入一个图5.8(c)所示的由两个训练样本线性组合而合成的新模式(复合模式)，那么网络的输出就是与复合输入模式中包含的样本相对应的输出模式的组合。CPN网能对复合输入模式包含的所有训练样本对应的输出进行线性叠加，这种能力对于图像的叠加等应用是十分合适的。

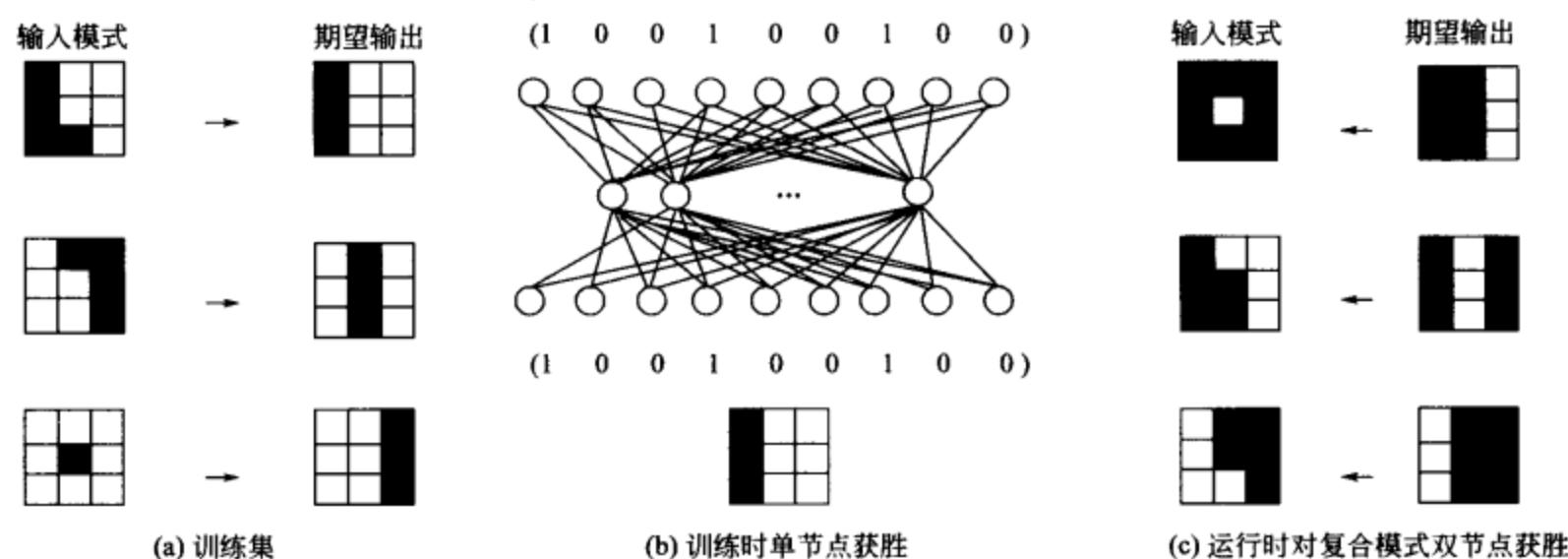


图5.8 允许双获胜节点的CPN网运行情况

5.2.3.2 双向CPN网

将CPN网的输入层和输出层各自分为两组，可变换为图5.9的形式。该网络有两个输入向量 \mathbf{X} 和 \mathbf{Y}' ，两个与之对应的输出向量 \mathbf{Y} 和 \mathbf{X}' 。训练隐层内星权向量时，将两个输入向量作为一个输入向量处理；训练隐层的外星向量时，将两个输出向量作为一个输出向量处理。两种权向量的调整规则与标准CPN网完全相同。

双向CPN网的优点是可以同时学习两个函数。例如：

$$\mathbf{Y} = f(\mathbf{X})$$

$$\mathbf{X}' = f(\mathbf{Y}')$$

当向网络输入 $(\mathbf{X}, 0)$ 时，网络输出为 $(\mathbf{Y}, 0)$ ；当向网络输入 $(0, \mathbf{Y}')$ 时，网络输出为 $(0, \mathbf{X}')$ ，当向网络输入 $(\mathbf{X}, \mathbf{Y}')$ 时，网络输

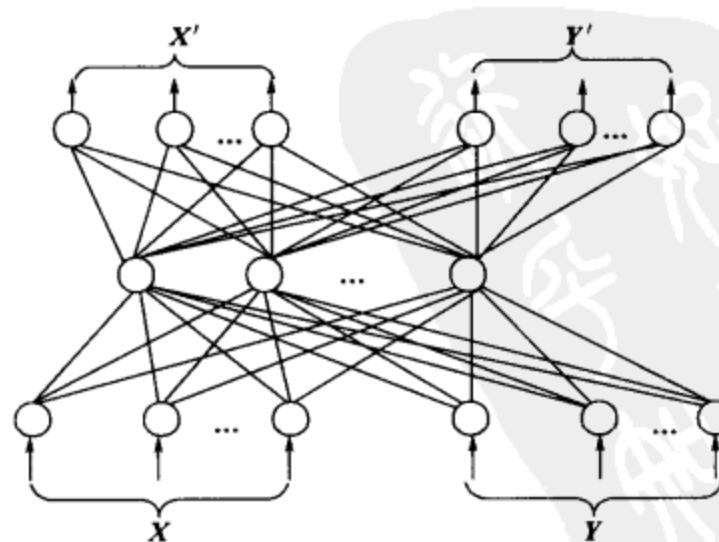


图5.9 双向CPN网络拓扑结构

出为 (Y, X') 。

当两个函数为互逆时, 有 $X=X'$, $Y=Y'$ 。双向 CPN 可用于数据压缩与解压缩, 可将其中一个函数 f 用作压缩函数, 将其逆函数 g 用作解压缩函数。

事实上双向 CPN 网并不要求两个互逆函数是解析表达的, 一般的情况下 f 和 g 是互逆的映射关系, 从而可利用双向 CPN 网实现互联想。

5.2.4 CPN 网的应用

5.2.4.1 CPN 网在颜色分类中的应用

图 5.10 给出 CPN 网络用于烤烟烟叶颜色模式分类的情况。其中输入样本分布在图 5.10(a) 所示的三维颜色空间中, 该空间的每个点用一个三维向量表示, 各分量分别代表烟叶的平均色调 H 、平均亮度 L 和平均饱和度 S 。烤烟烟叶的颜色与生长部分相关, 其中上部组烟叶可分为红棕、橘黄、柠檬黄三个小类; 中部组烟叶分为橘黄、柠檬黄两个小类, 下部烟叶分为橘黄、柠檬黄两个小类; 此外还有一类成熟度较差的烟叶颜色青黄色。若将全部烟叶按颜色分为四类模式, 分别为红棕类、橘黄类、柠檬黄类和青黄类。

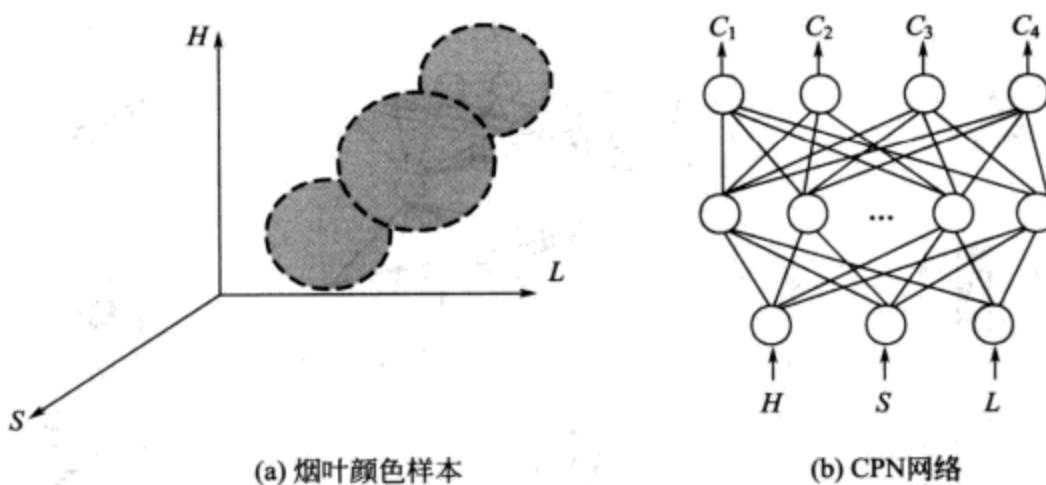


图 5.10 用于烟叶颜色模式分类的 CPN

图 5.10(b) 给出了 CPN 网络结构, 隐层共设了 8 个神经元, 用于对烟叶样本颜色聚类; 输出层设 4 个神经元, 用于对样本进行颜色分类。学习速率为随训练时间下降的函数, 经过 2000 次训练之后, 网络分类的正确率达 96%。

5.2.4.2 CPN 网在图像压缩中的应用

向量量化的理论基础是香农的速率失真理论, 其基本原理是用码书中与输入向量最匹配的码字的索引代替输入向量进行传输和存储, 而解码时只需简单的查表操作。向量量化的三大关键技术是码书设计、码字搜索和码字索引分配。

从 CPN 网络结构和学习规则来看, 标准 CPN 可以用作向量量化器, 用于图像压缩。该量化器模型描述如下:

(1) 输入层到竞争层部分作为编码器, 输入层节点数等于向量的维数 N , 竞争层神经元数目 M 等于码书大小。在训练阶段, CPN 竞争层各神经元相互竞争及调整权值的过程相当于输入向量聚类和生成码书的过程; 竞争层学习结束后, 每一个神经元代表输入向量的一个类别, 神经元的序号作为类别的标号即码字索引值, 权值向量存储的就是各类别的码字。在工作阶段, CPN 计算输入向量与所有权值向量的距离并选择获胜神经元的过程相当于码书搜索过程。

(2) 竞争层到输出层部分作为解码器, 输出层神经元数目也等于向量的维数 N 。在竞

争层完成学习过程后，输出层权值无需再进行训练，只需将某类输入所对应的获胜神经元 j^* 的内星权值 W_{j^*} 直接存放在其外星权值 V_{j^*} 中。工作时，根据获胜神经元的索引值 j^* 就可以直接查找到对应的近似输出 $O=W_{j^*} \cdot o$ ，这个过程相当于解码过程。

5.2.4.3 CPN 网络在人脸识别中的应用

人脸识别实验所采用的人脸图像均来自国际通用的人脸数据库 (oliver research laboratory, ORL)。该数据库存有 40 人的图像，每人在不同光照、不同角度、不同表情和不同细节条件下摄取 10 幅图像，每幅人脸图像像素数为 92×112 ，灰度级为 256。

训练集采用 ORL 库中前 20 人的图像，每人取前 n 幅图像，因此共有 $20 \times n$ 个训练样本。测试集 1 中的样本由训练集中 20 人的剩余图像组成，共有 $200 - 20 \times n$ 个样本测试样本，用于测试 CPN 网络对已知人脸的识别率。测试集 2 由 ORL 库中其余 20 人的图像组成，每人 10 幅图像共计 200 个样本，用于测试 CPN 网络对未知人脸的拒识率。

利用 CPN 网络识别人脸的方法如下：

(1) 将人脸图像各点像素值按照一定顺序组成一个高维向量作为输入模式送入 CP 网络，网络输入层的神经元数目为人脸图片像素数。

(2) 输出层神经元数目由模式类别数确定，因此神经元数 = 20。在训练过程中，每个输入模式的希望输出为 $[0, \dots, 0, 1, 0, \dots, 0]$ ，输出为 1 的神经元即对应于当前输入模式所属的模式类别。

(3) 竞争层的神经元数大于或等于输出层的神经元数。误差设置为 0.001，学习率 $\eta(t)$ 和 β 不超过 0.7。

(4) 当竞争获胜神经元对应的内星权向量与输入模式的相似性低于某个门限值时，将该输入判为拒识样本。

实验结果如表 5.4 所示。

表 5.4 n 取不同值时的识别率

n	训练集 样本数	收敛步数	训练集 识别率/%	测试集 1 识别率/%	测试集 2 拒识率/%
3	60	160	100	85.00	100
4	80	221	100	85.83	100
5	100	243	100	92.00	100
6	120	265	100	93.75	100

本章小结

本章介绍了两种将竞争学习与监督学习相结合的组合学习神经网络，其特点是网络的输出层采用监督学习算法而隐层采用竞争学习策略。学习重点如下：

(1) LVQ 神经网络 LVQ 网络将竞争学习和监督学习算法相结合，在网络学习过程中，输入层和竞争层之间的连接权值被逐渐调整为聚类中心。通过教师信号对输入样本的分配类别进行规定，从而克服了自组织网络采用无监督学习算法带来的缺乏分类信息的弱点。LVQ 网络是 SOFM 网络的一种有监督形式的扩展，两者有效结合可更好地发挥竞争学习和有监督学习的优点。

(2) CPN 神经网络 CPN 神经网络的拓扑结构与三层 BP 网相同，其隐层为竞争层，按照胜者为王规则调整其内星权向量，调整的目的是使权向量不断靠近当前输入模式类，从

而将该模式类的典型向量编码到获胜神经元的内星权向量中。输出层采用有导师的 Grossberg 外星学习规则调整，调整的目的是使外星权向量不断靠近并等于期望输出，从而将该输出编码到外星权向量中。

思考与练习

5.1 试分析 LVQ 网络与 SOM 网络的联系和区别。

5.2 试设计一个 LVQ 网络实现下述向量的分类：

$$\text{类 1} \left\{ \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \right\} \quad \text{类 2} \left\{ \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \right\} \quad \text{类 3} \left\{ \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} \right\}$$

试确定学习向量量化网各层的神经元数，并确定隐层和输出层的权值，然后对所设计的网络进行测试。

5.3 设输入为二维向量，共有 10 个输入样本：

$$\{(-6,0), (-4,2), (-2,-2), (0,1), (0,2), (0,-2), (0,1), (2,2), (4,-2), (6,0)\}$$

样本类别依次为：[1112222111]。试设计一个 LVQ 网络对样本进行分类。

5.4 某 LVQ 网络的权值矩阵如下：

$$W^1 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \quad W^2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

① 该 LVQ 网有多少个类和多少个子类？

② 画图展示第一层权值向量以及将输入空间分成子类的判决边界。

③ 在每个子类区域上标明其所属的类。

5.5 试设计一个 CPN 网实现对模式对 (A,C)、(I,I) 和 (O,T) 的异联想功能。给出训练后的权矩阵 W 和 V 。

5.6 试设计一个 CPN 网实现将四维输入模式映射为三维输出模式。两个输入模式为 $X^1 = (1, -1, 1, 1)^T$ 和 $X^2 = (1, 1, -1, -1)^T$ ，输出模式自行设计。请给出训练后的权矩阵 W 和 V 。

5.7 已知某人本星期应该完成的工作量和他的思想情绪状态，试用 MATLAB 设计一个 CPN 网络对此人星期日下午的活动安排提出建议。训练样本模式如下：

训练样本模式

工作量		思想情绪		活动安排	
没有	0.0	低	0.0	在家里看电视	10000
有一些	0.5	低	0.0	在家里看电视	10000
没有	0.0	一般	0.5	去商场购物	01000
很多	1.0	高	1.0	到公园散步	00100
有一些	0.5	高	1.0	与朋友吃饭	00010
很多	1.0	一般	0.5	工作	00001

6 反馈神经网络

根据神经网络运行过程中的信息流向，可分为前馈式和反馈式两种基本类型。第 3 章讨论的前馈网络通过引入隐层以及非线性转移函数，网络具有复杂的非线性映射能力。但前馈网络的输出仅由当前输入和权矩阵决定，而与网络先前的输出状态无关。

美国加州理工学院物理学家 J. J. Hopfield 教授于 1982 年发表了对神经网络发展颇具影响的论文，提出一种单层反馈神经网络，后来人们将这种反馈网络称作 Hopfield 网。J. J. Hopfield 教授在反馈神经网络中引入了“能量函数”的概念，这一概念的提出对神经网络的研究具有重大意义，它使神经网络运行稳定性的判断有了可靠依据。1985 年 Hopfield 还与 D. W. Tank 一道用模拟电子线路实现了 Hopfield 网，并成功地求解了优化组合问题中具有代表意义的 TSP 问题，从而开辟了神经网络用于智能信息处理的新途径，为神经网络的复兴立下了不可磨灭的功劳。

在前馈网络中，不论是离散型还是连续型，一般均不考虑输出与输入之间在时间上的滞后性，而只是表达两者间的映射关系。但在 Hopfield 网中，考虑了输出与输入间的延迟因素。因此，需要用微分方程或差分方程来描述网络的动态数学模型。

神经网络的学习方式有三种类型：其中有导师学习和无导师学习方式在第 3 章和第 4 章均已涉及；第三类学习方式是“死记硬背”，即网络的权值不是经过反复学习获得，而是按一定规则事前计算出来。Hopfield 网络便采用了这种学习方式，其权值一经确定就不再改变，而网络中各神经元的状态在运行过程中不断更新，网络演变到稳定时各神经元的状态便是问题之解。

Hopfield 网络分为离散型和连续型两种网络模型，分别记作 DHNN (Discrete Hopfield Neural Network) 和 CHNN (Continues Hopfield Neural Network)，本章重点讨论前一种类型。

1988 年 B. Kosko 提出一种双向联想记忆网络模型，记为 BAM (Bidirectional Associative Memory)。该网络也分为离散型和连续型两种类型，在联想记忆方面的应用非常广泛，本章重点介绍离散型 BAM 网。

6.1 离散型 Hopfield 神经网络

6.1.1 网络的结构与工作方式

离散型反馈网络的拓扑结构如图 6.1 所示。这是一种单层全反馈网络，共有 n 个神经元。其特点是任一神经元的输出 x_i 均通过连接权 w_{ij} 反馈至所有神经元 x_j 作为输入。换句话说，每个神经元都通过连接权接收所有神经元输出反馈回来的信息，其目的是为了让任一神经元的输出都能受所有神经元输出的控制，从而使各神经元的输出能相互制约。每个神经元

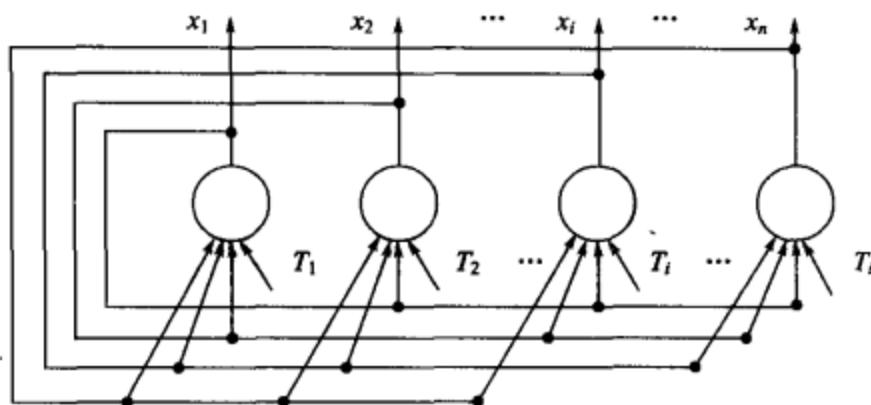


图 6.1 DHNN 网的拓扑结构

均设有一个阈值 T_j ，以反映对输入噪声的控制。DHNN 网可简记为 $N=(W, T)$ 。

(1) 网络的状态 DHNN 网中的每个神经元都有相同的功能，其输出称为状态，用 x_j 表示，所有神经元状态的集合就构成反馈网络的状态 $\mathbf{X}=[x_1, x_2, \dots, x_n]^T$ 。反馈网络的输入就是网络的状态初始值，表示为 $\mathbf{X}(0)=[x_1(0), x_2(0), \dots, x_n(0)]^T$ 。反馈网络在外界输入激发下，从初始状态进入动态演变过程，其间网络中每个神经元的状态在不断变化，变化规律由下式规定：

$$x_j = f(\text{net}_j) \quad j=1, 2, \dots, n$$

式中， $f(\cdot)$ 为转移函数，DHNN 网的转移函数常采用符号函数：

$$x_j = \text{sgn}(\text{net}_j) = \begin{cases} 1, & \text{net}_j \geq 0 \\ -1, & \text{net}_j < 0 \end{cases} \quad j=1, 2, \dots, n \quad (6.1)$$

式中，净输入为：

$$\text{net}_j = \sum_{i=1}^n (w_{ij} x_i - T_j) \quad j=1, 2, \dots, n \quad (6.2)$$

对于 DHNN 网，一般有 $w_{ii}=0$ ， $w_{ij}=w_{ji}$ 。

反馈网络稳定时每个神经元的状态都不再改变，此时的稳定状态就是网络的输出，表示为：

$$\lim_{t \rightarrow \infty} \mathbf{X}(t)$$

(2) 网络的异步工作方式 网络的异步工作方式是一种串行方式。网络运行时每次只有一个神经元 i 按式(6.1) 进行状态的调整计算，其他神经元的状态均保持不变，即：

$$x_j(t+1) = \begin{cases} \text{sgn}[\text{net}_j(t)], & j=i \\ x_j(t), & j \neq i \end{cases} \quad (6.3)$$

神经元状态的调整次序可以按某种规定的次序进行，也可以随机选定。每次神经元在调整状态时，根据其当前净输入值的正负决定下一时刻的状态，因此其状态可能会发生变化，也可能保持原状。下次调整其他神经元状态时，本次的调整结果即在下一个神经元的净输入中发挥作用。

(3) 网络的同步工作方式 网络的同步工作方式是一种并行方式，所有神经元同时调整状态，即：

$$x_j(t+1) = \text{sgn}[\text{net}_j(t)] \quad j=1, 2, \dots, n \quad (6.4)$$

6.1.2 网络的稳定性与吸引子

反馈网络是一种能存储若干个预先设置的稳定点（状态）的网络。运行时，当向该网络

作用一个起原始推动作用的初始输入模式后，网络便将其输出反馈回来作为下次的输入。经若干次循环（迭代）之后，在网络结构满足一定条件的前提下，网络最终将会稳定在某一预先设定的稳定点。

设 $\mathbf{X}(0)$ 为网络的初始激活向量，它仅在初始瞬间 $t=0$ 时作用于网络，起原始推动作用。 $\mathbf{X}(0)$ 移去之后，网络处于自激状态，即由反馈回来的向量 $\mathbf{X}(1)$ 作为下一次的输入取而代之。

反馈网络作为非线性动力学系统，具有丰富的动态特性，如稳定性、有限环状态和混沌（chaos）状态等。

6.1.2.1 网络的稳定性

由网络工作状态的分析可知，DHNN 网实质上是一个离散的非线性动力学系统。网络从初态 $\mathbf{X}(0)$ 开始，若能经有限次递归后，其状态不再发生变化，即 $\mathbf{X}(t+1)=\mathbf{X}(t)$ ，则称该网络是稳定的。如果网络是稳定的，它可以从任一初态收敛到一个稳态，如图 6.2(a) 所示；若网络是不稳定的，由于 DHNN 网每个节点的状态只有 1 和 -1 两种情况，网络不可能出现无限发散的情况，而只可能出现限幅的自持振荡，这种网络称为有限环网络，图 6.2(b) 给出了它的相图。如果网络状态的轨迹在某个确定的范围内变迁，但既不重复也不停止，状态变化为无穷多个，轨迹也不发散到无穷远，这种现象称为混沌，其相图如图 6.2(c) 所示。对于 DHNN 网，由于网络的状态是有限的，因此不可能出现混沌现象。

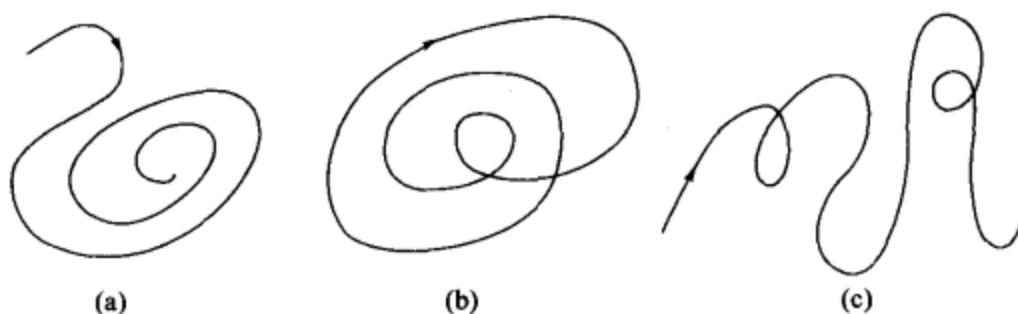


图 6.2 反馈网络的三种相图

利用 Hopfield 网的稳态，可实现联想记忆功能。Hopfield 网在拓扑结构及权矩阵均一定的情况下，能存储若干个预先设置的稳定状态；而网络运行后达到哪个稳定状态将与其初始状态有关。因此，若用网络的稳态代表一种记忆模式，初始状态朝着稳态收敛的过程便是网络寻找记忆模式的过程。初态可视为记忆模式的部分信息，网络演变的过程可视为从部分信息回忆起全部信息的过程，从而实现了联想记忆功能。

网络的稳定性与下面将要介绍的能量函数密切相关，利用网络的能量函数可实现优化求解功能。网络的能量函数在网络状态按一定规则变化时，能自动趋向能量的极小点。如果把一个待求解问题的目标函数以网络能量函数的形式表达出来，当能量函数趋于最小时，对应的网络状态就是问题的最优解。网络的初态可视为问题的初始解，而网络从初态向稳态的收敛过程便是优化计算过程，这种寻优搜索是在网络演变过程中自动完成的。

6.1.2.2 吸引子与能量函数

网络达到稳定时的状态 \mathbf{X} ，称为网络的吸引子。一个动力学系统的最终行为是由它的吸引子决定的，吸引子的存在为信息的分布存储记忆和神经优化计算提供了基础。如果把吸引子视为问题的解，那么从初态朝吸引子演变的过程便是求解计算的过程。若把需记忆的样本信息存储于不同的吸引子，当输入含有部分记忆信息的样本时，网络的演变过程便是从部分



信息寻找全部信息，即联想回忆的过程。

下面给出 DHNN 网吸引子的定义和定理：

定义 6.1 若网络的状态 \mathbf{X} 满足 $\mathbf{X} = f(\mathbf{WX} - \mathbf{T})$ ，则称 \mathbf{X} 为网络的吸引子。

定理 6.1 对于 DHNN 网，若按异步方式调整网络状态，且连接权矩阵 \mathbf{W} 为对称阵，则对于任意初态，网络都最终收敛到一个吸引子。

下面通过对能量函数的分析对定理 6.1 进行证明。

定义网络的能量函数为：

$$E(t) = -\frac{1}{2} \mathbf{X}^T(t) \mathbf{WX}(t) + \mathbf{X}^T(t) \mathbf{T} \quad (6.5)$$

令网络能量的改变量为 ΔE ，网络状态的改变量为 $\Delta \mathbf{X}$ ，有：

$$\Delta E(t) = E(t+1) - E(t) \quad (6.6)$$

$$\Delta \mathbf{X}(t) = \mathbf{X}(t+1) - \mathbf{X}(t) \quad (6.7)$$

将式(6.4) 和式(6.5) 代入式(6.6)，则网络能量可进一步展开为：

$$\begin{aligned} \Delta E(t) &= E(t+1) - E(t) \\ &= -\frac{1}{2} [\mathbf{X}(t) + \Delta \mathbf{X}(t)]^T \mathbf{W} [\mathbf{X}(t) + \Delta \mathbf{X}(t)] + [\mathbf{X}(t) + \Delta \mathbf{X}(t)]^T \mathbf{T} - [-\frac{1}{2} \mathbf{X}^T(t) \mathbf{WX}(t) + \mathbf{X}^T(t) \mathbf{T}] \\ &= -\Delta \mathbf{X}^T(t) \mathbf{WX}(t) - \frac{1}{2} \Delta \mathbf{X}^T(t) \mathbf{W} \Delta \mathbf{X}(t) + \Delta \mathbf{X}^T(t) \mathbf{T} \\ &= -\Delta \mathbf{X}^T(t) [\mathbf{WX}(t) - \mathbf{T}] - \frac{1}{2} \Delta \mathbf{X}^T(t) \mathbf{W} \Delta \mathbf{X}(t) \end{aligned} \quad (6.8)$$

由于定理 6.1 规定按异步工作方式，第 t 个时刻只有 1 个神经元调整状态，设该神经元为 j ，将 $\Delta \mathbf{X}(t) = [0, \dots, 0, \Delta x_j(t), 0, \dots, 0]^T$ 代入上式，并考虑到 \mathbf{W} 为对称矩阵，有：

$$\Delta E(t) = -\Delta x_j(t) \left[\sum_{i=1}^n (w_{ij} x_i - T_j) \right] - \frac{1}{2} \Delta x_j^2(t) w_{jj}$$

设各神经元不存在自反馈，有 $w_{jj} = 0$ ，并引入式(6.3)，上式可简化为：

$$\Delta E(t) = -\Delta x_j(t) \text{net}_j(t) \quad (6.9)$$

下面考虑上式中可能出现的所有情况。

情况 a $x_j(t) = -1, x_j(t+1) = 1$ ，由式(6.7) 得 $\Delta x_j(t) = 2$ ，由式(6.1) 知， $\text{net}_j(t) \geq 0$ ，代入式(6.9)，得 $\Delta E(t) \leq 0$ 。

情况 b $x_j(t) = 1, x_j(t+1) = -1$ ，所以 $\Delta x_j(t) = -2$ ，由式(6.1) 知， $\text{net}_j(t) < 0$ ，代入式(6.9)，得 $\Delta E(t) < 0$ 。

情况 c $x_j(t) = x_j(t+1)$ ，所以 $\Delta x_j(t) = 0$ ，代入式(6.9)，从而有 $\Delta E(t) = 0$ 。

以上三种情况包括了式(6.9) 可能出现的所有情况，由此可知在任何情况下均有 $\Delta E(t) \leq 0$ ，也就是说，在网络动态演变过程中。能量总是在不断下降或保持不变。由于网络中各节点的状态只能取 1 或 -1，能量函数 $E(t)$ 作为网络状态的函数是有下界的，因此网络能量函数最终将收敛于一个常数，此时 $\Delta E(t) = 0$ 。

下面分析当 $E(t)$ 收敛于常数时，是否对应于网络的稳态。当 $E(t)$ 收敛于常数时，有 $\Delta E(t) = 0$ ，此时对应于以下两种情况：

情况 a $x_j(t) = x_j(t+1) = 1$ 或 $x_j(t) = x_j(t+1) = -1$ ，这种情况下神经元 j 的状态不再改变，表明网络已进入稳态，对应的网络状态就是网络的吸引子。

情况 b $x_j(t) = -1, x_j(t+1) = 1, \text{net}_j(t) = 0$ ，这种情况下网络继续演变时， $x_j = 1$ 将



不会再变化。因为如果 x_j 由 1 变回到 -1，则有 $\Delta E(t) < 1$ ，与 $E(t)$ 收敛于常数的情况相矛盾。

综上所述，当网络工作方式和权矩阵均满足定理 6.1 的条件时，网络最终将收敛到一个吸引子。

事实上，对 $w_{jj}=0$ 的规定是为了数学推导的简便，如不作此规定，上述结论仍然成立。此外当神经元状态取 1 和 0 时，上述结论也将成立。

定理 6.2 对于 DHNN 网，若按同步方式调整状态，且连接权矩阵 W 为非负定对称阵，则对于任意初态，网络都最终收敛到一个吸引子。

证明 由式(6.8) 得：

$$\begin{aligned}\Delta E(t) &= E(t+1) - E(t) = -\Delta \mathbf{X}^T(t)[W\mathbf{X}(t) - \mathbf{T}] - \frac{1}{2}\Delta \mathbf{X}^T(t)W\Delta \mathbf{X}(t) \\ &= -\Delta \mathbf{X}^T(t)\text{net}(t) - \frac{1}{2}\Delta \mathbf{X}^T(t)W\Delta \mathbf{X}(t) \\ &= -\sum_{j=1}^n \Delta x_j(t)\text{net}_j(t) - \frac{1}{2}\Delta \mathbf{X}^T(t)W\Delta \mathbf{X}(t)\end{aligned}$$

前已证明，对于任何神经元 j ，有 $-\Delta x_j(t)\text{net}_j(t) \leq 0$ ，因此上式第一项不大于 0，只要 W 为非负定阵，第二项也不大于 0，于是有 $\Delta E(t) \leq 0$ 。也就是说， $E(t)$ 最终将收敛到一个常数值，对应的稳定状态是网络的一个吸引子。

比较定理 6.1 和定理 6.2 可以看出，网络采用同步方式工作时，对权值矩阵 W 的要求更高，如果 W 不能满足非负定对称阵的要求，网络会出现自持振荡。异步方式比同步方式有更好的稳定性，应用中较多采用，但其缺点是失去了神经网络并行处理的优势。

以上分析表明，在网络从初态向稳态演变的过程中，网络的能量始终向减小的方向演变，当能量最终稳定于一个常数时，该常数对应于网络能量的极小状态，称该极小状态为网络的能量井，能量井对应于网络的吸引子。

6.1.2.3 吸引子的性质

下面介绍吸引子的几个性质：

性质 1 若 \mathbf{X} 是网络的一个吸引子，且阈值 $T=0$ ，在 $\text{sgn}(0)$ 处， $x_j(t+1)=x_j(t)$ ，则 $-\mathbf{X}$ 也一定是该网络的吸引子。

证明 因为 \mathbf{X} 是吸引子，即 $\mathbf{X}=f(W\mathbf{X})$ ，从而有：

$$f[W(-\mathbf{X})] = f(-W\mathbf{X}) = -f(W\mathbf{X}) = -\mathbf{X}$$

所以， $-\mathbf{X}$ 也是该网络的吸引子。

性质 2 若 \mathbf{X}^a 是网络的一个吸引子，则与 \mathbf{X}^a 的海明距离 $dH(\mathbf{X}^a, \mathbf{X}^b)=1$ 的 \mathbf{X}^b 一定不是吸引子。

证明 首先说明，两个向量的海明距离 $dH(\mathbf{X}^a, \mathbf{X}^b)$ 是指两个向量中不相同元素的个数。不妨设 $x_1^a \neq x_1^b$ ， $x_j^a \neq x_j^b$ ， $j=2, 3, \dots, n$ 。

因为 $w_{11}=0$ ，由吸引子定义，有：

$$x_1^a = f\left(\sum_{i=2}^n w_{ii}x_i^a - T_1\right) = f\left(\sum_{i=2}^n w_{ii}x_i^b - T_1\right)$$

由假设条件知， $x_1^a \neq x_1^b$ ，故：

$$x_1^b \neq f\left(\sum_{i=2}^n w_{ij} x_i^b - T_1\right)$$

所以, $-X$ 也是该网络的吸引子。

性质 3 若有一组向量 $X^p (p=1, 2, \dots, P)$ 均是网络的吸引子, 且在 $\text{sgn}(0)$ 处, $x_j(t+1) = x_j(t)$, 则由该组向量线性组合而成的向量 $\sum_{p=1}^P a_p X^p$ 也是该网络的吸引子。

该性质请读者自己证明。

6.1.2.4 吸引子的吸引域

能使网络稳定在同一吸引子的所有初态的集合, 称该吸引子的吸引域。下面给出关于吸引域的两个定义。

定义 6.2 若 X^a 是吸引子, 对于异步方式, 若存在一个调整次序, 使网络可以从状态 X 演变到 X^a , 则称 X 弱吸引到 X^a ; 若对于任意调整次序, 网络都可以从状态 X 演变到 X^a , 则称 X 强吸引到 X^a 。

定义 6.3 若对某些 X , 有 X 弱吸引到吸引子 X^a , 则称这些 X 的集合为 X^a 的弱吸引域; 若对某些 X , 有 X 强吸引到吸引子 X^a , 则称这些 X 的集合为 X^a 的强吸引域。

欲使反馈网络具有联想能力, 每个吸引子都应该具有一定的吸引域。只有这样, 对于带有一定噪声或缺损的初始样本, 网络才能经过动态演变而稳定到某一吸引子状态, 从而实现正确联想。反馈网络设计的目的就是要使网络能落到期望的稳定点(问题的解)上, 并且还要具有尽可能大的吸引域, 以增强联想功能。

例 6.1 设有 3 节点 DHNN 网, 用图 6.3(a) 所示的无向图表示, 权值与阈值均已标在图中, 试计算网络演变过程的状态。

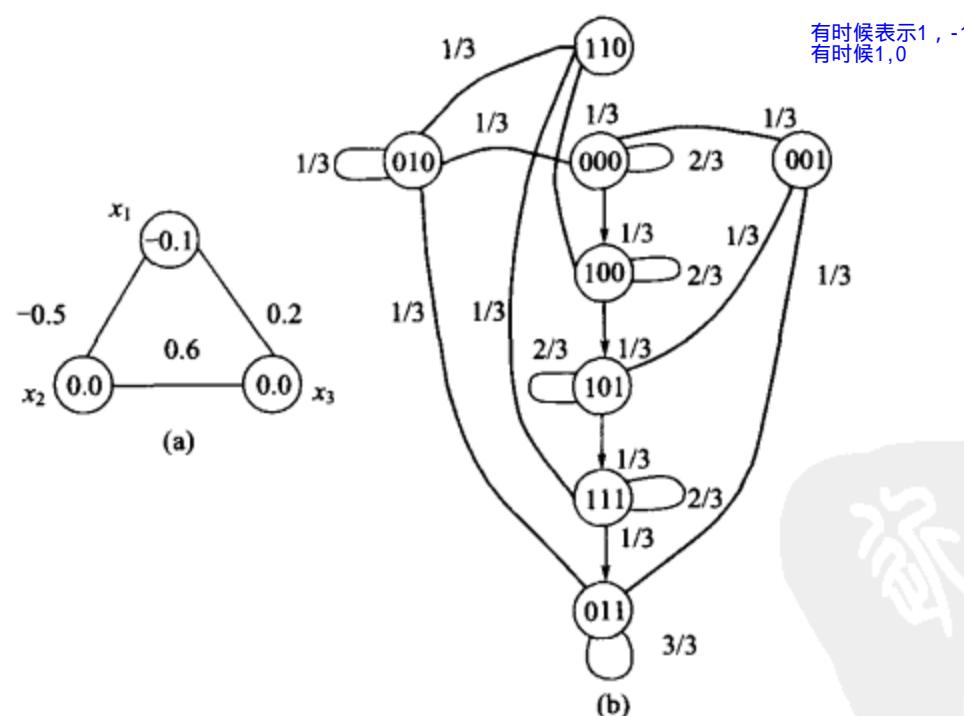


图 6.3 DHNN 网络状态演变示意图

解 设各节点状态取值为 1 或 0, 3 节点 DHNN 网络应有 $2^3 = 8$ 种状态。不妨将 $X = (x_1, x_2, x_3)^T = (0, 0, 0)^T$ 作为网络初态, 按 $1 \rightarrow 2 \rightarrow 3$ 的次序更新状态。

第 1 步, 更新 $x_1, x_1 = \text{sgn}[-0.5 \times 0 + 0.2 \times 0 - (-0.1)] = \text{sgn}(0.1) = 1$, 其他节点状态不变, 网络状态由 $(0, 0, 0)^T$ 变成 $(1, 0, 0)^T$ 。如果先更新 x_2 或 x_3 , 网络状态将仍为

$(0,0,0)^T$, 因此初态保持不变的概率为 $2/3$, 而变为 $(1,0,0)^T$ 的概率为 $1/3$ 。

第 2 步, 此时网络状态为 $(1,0,0)^T$, 更新 x_2 后, 得 $x_2 = \text{sgn}[-0.5 \times 1 + 0.6 \times 0 - 0] = \text{sgn}(-0.5) = 0$, 其他节点状态不变, 网络状态仍为 $(1,0,0)^T$ 。如果本步先更新 x_1 或 x_3 , 网络相应状态将为 $(1,0,0)^T$ 和 $(1,0,1)^T$, 因此本状态保持不变的概率为 $2/3$, 而变为 $(1,0,0)^T$ 的概率为 $1/3$ 。

第 3 步, 此时网络状态为 $(1,0,0)^T$, 更新 x_3 得, $x_3 = \text{sgn}[0.2 \times 1 + 0.6 \times 0 - 0] = \text{sgn}(0.2) = 1$ 。

同理可算出其他状态之间的演变历程和状态转移概率, 图 6.3(b) 给出了 8 种状态的演变关系。图中, 圆圈内的二进制串代表网络的状态 $x_1x_2x_3$, 有向线表示状态转移方向, 线旁标出了相应的状态转移概率。从图中可以看出, $X=(011)^T$ 是网络的一个吸引子, 网络从任意状态出发, 经过几次状态更新后都将达到此稳定状态。

例 6.2 有一 DHNN 网, $n=4$, $T_j=0$, $j=1,2,3,4$, 向量 X^a 、 X^b 和权值矩阵 W 分别为:

$$X^a = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad X^b = \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \end{pmatrix} \quad W = \begin{pmatrix} 0 & 2 & 2 & 2 \\ 2 & 0 & 2 & 2 \\ 2 & 2 & 0 & 2 \\ 2 & 2 & 2 & 0 \end{pmatrix}$$

检验 X^a 和 X^b 是否为网络的吸引子, 并考察其是否具有联想记忆能力。

解 本例要求验证吸引子和检查吸引域, 下面分两步进行。

① 检验吸引子 由吸引子定义:

$$f(WX^a) = f \begin{pmatrix} 6 \\ 6 \\ 6 \\ 6 \end{pmatrix} = \begin{pmatrix} \text{sgn}(6) \\ \text{sgn}(6) \\ \text{sgn}(6) \\ \text{sgn}(6) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = X^a$$

所以 X^a 是网络的吸引子, 因为 $X^b = -X^a$, 由吸引子的性质 1 知, X^b 也是网络的吸引子。

② 考察联想记忆能力 设有样本 $X^1=(-1,1,1,1)^T$, $X^2=(1,-1,-1,-1)^T$, $X^3=(1,1,-1,-1)^T$, 试考察网络以异步方式工作时两个吸引子对三个样本的吸引能力。

令网络初态 $X(0)=X^1=(-1,1,1,1)^T$ 。设神经元状态调整次序为 $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$, 有:

$$X(0)=(-1,1,1,1)^T \rightarrow X(1)=(1,1,1,1)^T=X^a$$

可以看出该样本比较接近吸引子 X^a , 事实上只按异步方式调整了一步, 样本 X^1 即收敛于 X^a 。

令网络初态 $X(0)=X^2=(1,-1,-1,-1)^T$ 。设神经元状态调整次序为 $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$, 有:

$$X(0)=(1,-1,-1,-1)^T \rightarrow X(1)=(-1,-1,-1,-1)^T=X^b$$

可以看出样本 X^2 比较接近吸引子 X^b , 按异步方式调整一步后, 样本 X^2 收敛于 X^b 。

令网络初态 $X(0)=X^3=(1,1,-1,-1)^T$, 它与两个吸引子的海明距离相等。若设神经元状态调整次序为 $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$, 有:

$$X(0)=(1,1,-1,-1)^T \rightarrow X(1)=(-1,1,-1,-1)^T \rightarrow X(2)=(-1,-1,-1,-1)^T=X^b$$

若将神经元状态调整次序改为 $3 \rightarrow 4 \rightarrow 1 \rightarrow 2$, 则有:

$$\mathbf{X}(0) = (1, 1, -1, -1)^T \rightarrow \mathbf{X}(1) = (1, 1, 1, -1)^T \rightarrow \mathbf{X}(2) = (1, 1, 1, 1)^T = \mathbf{X}^a$$

从本例可以看出, 当网络的异步调整次序一定时, 最终稳定于哪个吸引子与其初态有关; 而对于确定的初态, 网络最终稳定于哪个吸引子与其异步调整次序有关。

6.1.3 网络的权值设计

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

- ① 为保证异步方式工作时网络收敛, \mathbf{W} 应为对称阵;
- ② 为保证同步方式工作时网络收敛, \mathbf{W} 应为非负定对称阵;
- ③ 保证给定的样本是网络的吸引子, 并且要有一定的吸引域。

根据应用所要求的吸引子数量, 可以采用以下不同的方法:

6.1.3.1 联立方程法

下面将以图 6.3(a) 中的 3 节点 DHNN 网为例, 说明权值设计的联立方程法。设要求设计的吸引子为 $\mathbf{X}^a = (010)^T$ 和 $\mathbf{X}^b = (111)^T$, 权值和阈值在 $[-1, 1]$ 区间取值, 试求权值和阈值。

考虑到 $w_{ij} = w_{ji}$, 对于状态 $\mathbf{X}^a = (010)^T$, 各节点净输入应满足:

$$\text{net}_1 = w_{12} \times 1 + w_{13} \times 0 - T_1 = w_{12} - T_1 < 0 \quad (6.10)$$

$$\text{net}_2 = w_{12} \times 0 + w_{23} \times 0 - T_2 = -T_2 > 0 \quad (6.11)$$

$$\text{net}_3 = w_{13} \times 0 + w_{23} \times 1 - T_3 = w_{23} - T_3 < 0 \quad (6.12)$$

对于 $\mathbf{X}^b = (111)^T$ 状态, 各节点净输入应满足:

$$\text{net}_1 = w_{12} \times 1 + w_{13} \times 1 - T_1 > 0 \quad (6.13)$$

$$\text{net}_2 = w_{12} \times 1 + w_{23} \times 1 - T_2 > 0 \quad (6.14)$$

$$\text{net}_3 = w_{13} \times 1 + w_{23} \times 1 - T_3 > 0 \quad (6.15)$$

联立以上 6 项不等式, 可求出 6 个未知量的允许取值范围。如取 $w_{12} = 0.5$, 则由式 (6.10), 有 $1 < T_1 \leq 1$, 取 $T_1 = 0.7$;

由式(6.13), 有 $0.2 < w_{13} \leq 1$, 取 $w_{13} = 0.4$;

由式(6.11), 有 $-1 \leq T_2 < 0$, 取 $T_2 = -0.2$;

由式(6.14), 有 $-0.7 < w_{23} \leq 1$, 取 $w_{23} = 0.1$;

由式(6.15), 有 $-1 \leq T_3 < 0.5$, 取 $T_3 = 0.4$ 。

可以验证, 利用这组参数构成的 DHNN 网对于任何初态最终都将演变到两个给定吸引子之一, 读者不妨一试。

当所需要的吸引子较多时, 可采用下面的方法。

6.1.3.2 外积和法

更为通用的权值设计方法是采用 Hebb 规则的外积和法。设给定 P 个模式样本 \mathbf{X}^p , $p=1, 2, \dots, P$, $x \in \{-1, 1\}^n$, 并设样本两两正交, 且 $n > P$, 则权值矩阵为记忆样本的外积和:

$$\mathbf{W} = \sum_{p=1}^P \mathbf{X}^p (\mathbf{X}^p)^T \quad (6.16)$$

若取 $w_{jj} = 0$, 上式应写为:

$$\mathbf{W} = \sum_{p=1}^P [\mathbf{X}^p (\mathbf{X}^p)^T - \mathbf{I}] \quad (6.17)$$

式中, \mathbf{I} 为单位矩阵。上式写成分量元素形式, 有:

$$w_{ij} = \begin{cases} \sum_{p=1}^P x_i^p x_j^p, & i \neq j \\ 0, & i = j \end{cases} \quad (6.18)$$

按以上外积和规则设计的 \mathbf{W} 阵必然满足对称性要求。下面检验所给样本能否称为吸引子。

因为 P 个样本 \mathbf{X}^p , $p=1, 2, \dots, P$, $x \in \{-1, 1\}^n$ 是两两正交的, 有:

$$(\mathbf{X}^p)^T \mathbf{X}^k = \begin{cases} 0, & p \neq k \\ n, & p = k \end{cases}$$

所以:

$$\begin{aligned} \mathbf{W} \mathbf{X}^k &= \sum_{p=1}^P [\mathbf{X}^p (\mathbf{X}^p)^T - \mathbf{I}] \mathbf{X}^k = \sum_{p=1}^P [\mathbf{X}^p (\mathbf{X}^p)^T \mathbf{X}^k - \mathbf{X}^k] \\ &= \mathbf{X}^k (\mathbf{X}^k)^T \mathbf{X}^k - P \mathbf{X}^k \\ &= n \mathbf{X}^k - P \mathbf{X}^k = (n - P) \mathbf{X}^k \end{aligned}$$

因为 $n > P$, 所以有:

$$f(\mathbf{W} \mathbf{X}^p) = f[(n - P) \mathbf{X}^p] = \text{sgn}[(n - P) \mathbf{X}^p] = \mathbf{X}^p$$

可见给定样本 \mathbf{X}^p , $p=1, 2, \dots, P$ 是吸引子。需要指出的是, 有些非给定样本也是网络的吸引子, 它们并不是网络设计所要求的解, 这种吸引子称为伪吸引子。

6.1.4 网络的信息存储容量

当网络规模一定时, 所能记忆的模式是有限的。对于所容许的联想出错率, 网络所能存储的最大模式数 P_{\max} 称为网络容量。网络容量与网络的规模、算法以及记忆模式向量的分布都有关系。下面给出 DHNN 网络存储容量的有关定理:

定理 6.3 若 DHNN 网络的规模为 n , 且权矩阵主对角线元素为 0, 则该网络的信息容量上界为 n 。

定理 6.4 若 P 个记忆模式 \mathbf{X}^p , $p=1, 2, \dots, P$, $x \in \{-1, 1\}^n$ 两两正交, $n > P$, 且权值矩阵 \mathbf{W} 按式(6.17) 得到, 则所有 P 个记忆模式都是 DHNN 网 $(\mathbf{W}, 0)$ 的吸引子。

定理 6.5 若 P 个记忆模式 \mathbf{X}^p , $p=1, 2, \dots, P$, $x \in \{-1, 1\}^n$ 两两正交, $n \geq P$, 且权值矩阵 \mathbf{W} 按式(6.16) 得到, 则所有 P 个记忆模式都是 DHNN 网 $(\mathbf{W}, 0)$ 的吸引子。

由以上定理可知, 当用外积和设计 DHNN 网时, 如果记忆模式都满足两两正交的条件, 则规模为 n 维的网络最多可记忆 n 个模式。一般情况下, 模式样本不可能都满足两两正交的条件, 对于非正交模式, 网络的信息存储容量会大大降低。下面进行简要分析。

DHNN 网的所有记忆模式都存储在权矩阵 \mathbf{W} 中。由于多个存储模式互相重叠, 当需要记忆的模式数增加时, 可能会出现所谓“权值移动”和“交叉干扰”。如将式(6.17) 写为:

$$\begin{cases} \mathbf{W}^0 = 0 \\ \mathbf{W}^p = \mathbf{W}^{p-1} + \mathbf{X}^p (\mathbf{X}^p)^T - \mathbf{I} & p = 1, 2, \dots, P \end{cases}$$

可以看出, \mathbf{W} 对要记忆的模式 \mathbf{X}^p , $p=1, 2, \dots, P$, 是累加实现的。每记忆一个新模式 \mathbf{X}^p , 就要向原权值矩阵 \mathbf{W}^{p-1} 加入一项该模式的外积 $\mathbf{X}^p \mathbf{X}^p$, 从而使新的权值矩阵 \mathbf{W}^p 从原来

的基础上发生移动。如果在加入新模式 \mathbf{X}^p 之前存储的模式都是吸引子，应有 $\mathbf{X}^k = f(\mathbf{W}^{p-1}\mathbf{X}^k)$, $k=1, 2, \dots, p-1$, 那么在加入模式 \mathbf{X}^p 之后由于权值移动为 \mathbf{W}^p , 式 $\mathbf{X}^k = f(\mathbf{W}^p\mathbf{X}^k)$ 就不一定对所有 $k=(1, 2, \dots, p-1)$ 均同时成立，也就是说网络在记忆新样本的同时可能会遗忘已记忆的样本。随着记忆模式数的增加，权值不断移动，各记忆模式相互交叉，当模式数超过网络容量 P_{\max} 时，网络不但逐渐遗忘了以前记忆的模式，而且也无法记住新模式。

事实上，当网络规模 n 一定时，要记忆的模式数越多，联想时出错的可能性越大；反之，要求的出错概率越低，网络的信息存储容量上限越小。研究表明存储模式数 P 超过 $0.15n$ 时，联想时就有可能出错。错误结果对应的是能量的某个局部极小点，或称为伪吸引子。

提高网络存储容量有两个基本途径：一是改进网络的拓扑结构；二是改进网络的权值设计方法。常用的改进方法有：反复学习法、纠错学习法、移动兴奋门限法、伪逆技术、忘记规则和非线性学习规则等。读者可参考有关文献。

6.2 连续型 Hopfield 神经网络

1984 年 Hopfield 把 DHNN 进一步发展成连续型 Hopfield 网络，缩写为 CHNN 网。CHNN 的基本结构与 DHNN 相似，但 CHNN 中所有神经元都同步工作，各输入输出量均是随时间连续变化的模拟量，这就使得 CHNN 比 DHNN 在信息处理的并行性、实时性等方面更接近于实际生物神经网络的工作机理。

CHNN 可以用常系数微分方程来描述，但用模拟电子线路来描述，则更为形象直观，易于理解也便于实现。

6.2.1 网络的拓扑结构

在连续 Hopfield 网中，所有神经元都随时间 t 并行更新，网络状态随时间连续变化。图 6.4 给出了基于模拟电子线路的 CHNN 的拓扑结构，可以看出 CHNN 模型可与电子线路直接对应，每一个神经元可以用一个运算放大器来模拟，神经元的输入与输出分别用运算放大器的输入电压 u_j 和输出电压 v_j 表示， $j=1, 2, \dots, n$ ，而连接权 w_{ij} 用输入端的电导表示，其作用是把第 i 个神经元的输出反馈到第 j 个神经元作为输入之一。每个运算放大器均有一个正相输出和一个反相输出。与正相输出相连的电导表示兴奋性突触，而与反相输出相连的电导表示抑制性突触。另外，每个神经元还有一个用于设置激活电平的外界输入偏置电流 I_j ，其作用相当于阈值。

图中， C_j 和 $1/g_j$ 分别为运放的等效输入电容和电阻，用来模拟生物神经元的输出时间常数。根据基尔霍夫定律可写出以下方程：

$$c_j \frac{du_j}{dt} + g_j u_j = \sum_{i=1}^n (w_{ij} v_i - u_j) + I_j$$

对上式移项合并，并令 $\sum_{i=1}^n w_{ij} + g_j = \frac{1}{R_j}$ ，则有：

$$c_j \frac{du_j}{dt} = \sum_{i=1}^n w_{ij} v_i - \frac{u_j}{R_j} + I_j \quad (6.19)$$

CHNN 中的转移函数为 S 型函数：

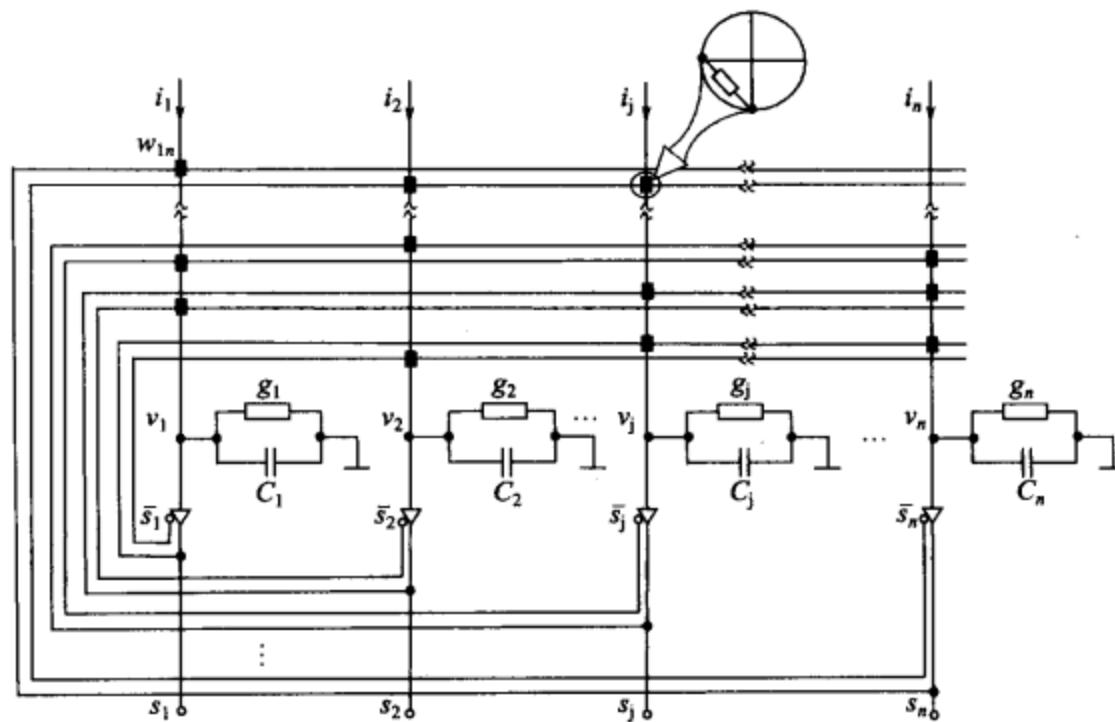


图 6.4 CHNN 的拓扑结构

$$v_j = f(u_j) \quad (6.20)$$

利用其饱和特性可限制神经元状态 v_j 的增长范围，从而使网络状态能在一定范围内连续变化。联立以上两式可描述 CHNN 网的动态过程。

CHNN 模型对生物神经元的功能作了大量简化，只模仿了生物系统的几个基本特性：S 型转移函数；信息传递过程中的时间常数；神经元间的兴奋及抑制性连接；以及神经元间的相互作用和时空作用。

6.2.2 能量函数与稳定性分析

定义 CHNN 的能量函数为：

$$E = -\frac{1}{2} \sum_{j=1}^n \sum_{i=1}^n w_{ij} v_i v_j - \sum_{j=1}^n v_j I_j + \sum_{j=1}^n \frac{1}{R_j} \int_0^{v_j} f^{-1}(\nu) d\nu \quad (6.21)$$

写成向量式为：

$$E = -\frac{1}{2} \mathbf{V}^T \mathbf{W} \mathbf{V} - \mathbf{I}^T \mathbf{V} + \sum_{j=1}^n \frac{1}{R_j} \int_0^{v_j} f^{-1}(\nu) d\nu \quad (6.22)$$

式中， f^{-1} 为神经元转移函数的反函数。对于式(6.21) 所定义的能量函数，存在以下定理。

定理 6.6 若神经元的转移函数 f 存在反函数 f^{-1} ，且 f^{-1} 是单调连续递增的，同时网络权值对称，即 $w_{ij} = w_{ji}$ ，则由任意初态开始，CHNN 网络的能量函数总是单调递减的，即 $\frac{dE}{dt} \leq 0$ ，当且仅当 $\frac{dv_j}{dt} = 0$ 时，有 $\frac{dE}{dt} = 0$ ，因而网络最终能够达到稳态。

证明 将能量函数对时间求导，得：

$$\frac{dE}{dt} = \sum_{j=1}^n \frac{\partial E}{\partial v_j} \frac{dv_j}{dt} \quad (6.23)$$

由式(6.21) 和 $u_j = f^{-1}(v_j)$ 及网络的对称性，对某神经元 j ，有：

$$\frac{\partial E}{\partial v_j} = -\frac{1}{2} \sum_{i=1}^n w_{ij} v_i - I + \frac{u_j}{R_j} \quad (6.24)$$

将上式代入式(6.23)，并考虑到式(6.19)，可整理得出下式：

$$\begin{aligned}\frac{dE}{dt} &= \sum_{j=1}^n \frac{du_j}{dt} c_j \frac{d\nu_j}{dt} = - \sum_{j=1}^n c_j \frac{du_j}{d\nu_j} \left(\frac{d\nu_j}{dt} \right)^2 \\ &= - \sum_{j=1}^n c_j f^{-1}(\nu_j) \left(\frac{d\nu_j}{dt} \right)^2\end{aligned}$$

可以看出，上式中 $c_j > 0$ ，单调递增函数 $f^{-1}(\nu_j) > 0$ ，故有：

$$\frac{dE}{dt} \leq 0 \quad (6.25)$$

只有对于所有 j 均满足 $\frac{d\nu_j}{dt} = 0$ 时，才有 $\frac{dE}{dt} = 0$ 。

如果图 6.4 中的运算放大器接近理想运放，式(6.21)中的积分项可以忽略不计，网络的能量函数可写为：

$$E = -\frac{1}{2} \sum_{j=1}^n \sum_{i=1}^n w_{ij} \nu_i \nu_j - \sum_{j=1}^n \nu_j I_j \quad (6.26)$$

由定理 6.6 可知，随着状态的演变，网络的能量总是降低的。只有当网络中所有节点的状态不再改变时，能量才不再变化，此时到达能量的某一局部极小点或全局最小点，该能量点对应着网络的某一个稳定状态。

Hopfield 网用于联想记忆时，正是利用了这些局部极小点来记忆样本，网络的存储容量越大，说明网络的局部极小点越多。然而在优化问题中，局部极小点越多，网络就越不容易达到最优解，而只能达到较优解。

为保证网络的稳定性，要求网络的结构必须对称，否则运行中可能出现极限环或混沌状态。

6.3 Hopfield 网络应用与设计实例

Hopfield 网络在图像、语音和信号处理、模式分类与识别、知识处理、自动控制、容错计算和数据查询等领域已经有许多成功的应用。Hopfield 网络的应用主要有联想记忆和优化计算两类，其中 DHNN 网主要用于联想记忆，CHNN 网主要用于优化计算。

6.3.1 应用 DHNN 网解决联想问题

神经网络的联想记忆只需存储输入-输出模式间的转换机制，而不必像传统计算机那样存储各输入、输出模式本身。神经网络的权矩阵就是把各种输入模式映射成相应输出模式的转换机制。这种映射是对模式的整体而言的，在组成输入-输出模式的各元素之间，并不存在一对一的映射关系，并且输入-输出模式的维数也不要求相同。

传统数字计算机的地址寻址方式要求给出地址的全部信息。而对按内容寻址记忆方式工作的神经网络来说，只给出输入模式的部分信息，网络便能正确地联想到完整的输出模式。这是因为在分布式存储方式中，不论是输入模式还是权矩阵中，少量且分散的局部信息出错，对整个转换结果的全局而言是无关紧要的。神经网络的这种容错性使它具有识别含噪声、畸变或残缺的模式的能力。

6.3.2 应用 CHNN 网解决优化计算问题

用 CHNN 网解决优化问题一般需要以下几个步骤：

- ① 对于特定的问题，要选择一种合适的表示方法，使得神经网络的输出与问题的解相对应；
- ② 构造网络能量函数，使其最小值对应于问题的最佳解；
- ③ 将能量函数与式(6.26)中的标准形式进行比较，可推出神经网络的权值与偏流的表达式，从而确定了网络的结构；
- ④ 由网络结构建立网络的电子线路并运行，其稳态就是在一定条件下的问题优化解。也可以编程模拟网络的运行方式，在计算机上实现。

本节介绍应用 CHNN 解决 TSP 问题的网络设计。本书在第 4 章中已指出，TSP 问题是一个经典的人工智能难题。对 n 个城市而言，可能的路径总数为 $n! / 2n$ 。随着 n 的增加，路径数将按指数率急剧增长，即所谓“指数爆炸”。当 n 值较大时，用传统的数字计算机也无法在有限时间内寻得答案。例如， $n=50$ 时，即使用每秒 1 亿次运算速度的巨型计算机按穷举搜索法，也需要 5×10^{48} 年时间。即使是 $n=20$ 个城市，也需求解 350 年。

1985 年 Hopfield 和 Tank 用 CHNN 网络为解决 TSP 难题开辟了一条崭新的途径，获得了巨大的成功。其基本思想是把 TSP 问题映射到 CHNN 网络中去，并设法用网络能量代表路径总长。这样，当网络的能量随着模拟电子线路状态的变迁，最终收敛于极小值（或最小值）时，问题的较佳解（或最佳解）便随之求得。此外，由于模拟电子线路中的全部元件都是并行工作的，所以求解时间与城市数的多少无关，仅是运算放大器工作所需的微秒级时间，显著地提高了求解速度，充分展示了神经网络的巨大优越性。

6.3.2.1 TSP 问题描述

为使 CHNN 网络完成优化计算，必须找到一种合适的表示旅行路线的方法。鉴于 TSP 的解是 n 个城市的有序排列，因此可用一个由 $n \times n$ 个神经元构成的矩阵（称为换位阵）来描述旅行路线。图 6.5 给出 8 城市 TSP 问题中的一条可能的有效路线的换位阵。

由于每个城市仅能访问一次，因此换位阵中每城市行只允许且必须有一个 1，其余元素均为 0。为了用神经元的状态表示某城市在某一有效路线中的位置，采用双下标 v_{xi} ，第一个下标 x 表示城市名， $x=1, 2, \dots, n$ ；第二个下标 i 表示该城市在访问路线中的位置， $i=1, 2, \dots, n$ 。例如， $v_{46}=1$ 表示旅途中第 6 站应访问城市 4；若 $v_{46}=0$ 则表示第 6 站访问的不是城市 4，而是其他某个城市。图 6.5 中的换位阵所表示的旅行路线为：4→2→5→8→1→3→7→6→4，旅行路线总长为 $d_{42} + d_{25} + d_{58} + d_{81} + d_{13} + d_{37} + d_{76} + d_{64}$ 。

6.3.2.2 能量函数设计

用 CHNN 求解 TSP 问题的关键是构造一个合适的能量函数。TSP 问题的能量函数由 4 部分组成：

(1) 能量 E_1 ——城市行约束 当每个城市行中的 1 不多于一个时，应有第 x 行的全部元素 v_{xi} 按顺序两两相乘之和为 0，即：

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^n v_{xi} v_{xj} = 0$$

	1	2	3	4	5	6	7	8
城市名	0	0	0	0	1	0	0	0
x	0	1	0	0	0	0	0	0
	1	2	3	4	5	6	7	8
位置 i								

图 6.5 8 城市 TSP 问题中的有效路线换位阵

从而全部 n 行的所有元素按顺序两两相乘之和也应为零，即：

$$\sum_{x=1}^n \sum_{i=1}^{n-1} \sum_{j=i+1}^n v_{xi} v_{xj} = 0$$

按此约束可定义能量 E_1 为：

$$E_1 = \frac{1}{2} A \sum_{x=1}^n \sum_{i=1}^{n-1} \sum_{j=i+1}^n v_{xi} v_{xj} \quad (6.27)$$

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

(2) 能量 E_2 ——位置列约束 同理，当每个位置列中的 1 不多于一个时，应有第 i 列的全部元素 v_{xi} 按顺序两两相乘之和为 0，即：

$$\sum_{x=1}^{n-1} \sum_{y=x+1}^n v_{xi} v_{yi} = 0$$

因此，全部 n 列的所有元素按顺序两两相乘之和也应为零，即：

$$\sum_{i=1}^n \sum_{x=1}^{n-1} \sum_{y=x+1}^n v_{xi} v_{yi} = 0$$

按此约束可定义能量 E_2 为：

$$E_2 = \frac{1}{2} B \sum_{i=1}^n \sum_{x=1}^{n-1} \sum_{y=x+1}^n v_{xi} v_{yi} \quad (6.28)$$

式中， B 为大于 0 的常数。显然，当 $E_2 = 0$ 时就能确保每次访问的城市数不超过一个。

(3) 能量 E_3 ——换位阵全局约束 $E_1 = 0$ 和 $E_2 = 0$ 只是换位阵有效的必要条件，但不是充分条件。容易看出，当换位阵中各元素均为“0”时，也能满足 $E_1 = 0$ 和 $E_2 = 0$ ，但这显然是无效的。因此，还需引入第三个约束条件——全局约束条件，以确保换位阵中 1 的数目等于城市数 n ，即：

$$\sum_{x=1}^n \sum_{i=1}^n v_{xi} = n$$

因此，定义能量 E_3 为：

$$E_3 = \frac{1}{2} C \left(\sum_{x=1}^n \sum_{i=1}^n v_{xi} - n \right)^2 \quad (6.29)$$

式中， C 为大于 0 的常数。则 $E_3 = 0$ 可保证换位阵中 1 的数目正好等于 n 。

(4) 能量 E_4 ——旅行路线长度 同时满足以上三个约束条件只能说明路线是有效的，但不一定是最佳的。依题意，在路线有效的前提下，其总长度应最短。为此在能量函数中尚须引入一个能反映路线总长度的分量 E_4 ，其定义式要能保证 E_4 随路线总长度的缩短而减小。为设计 E_4 ，设任意两城市 x 与 y 间的距离为 d_{xy} 。访问这两个城市有两种途径，从 x 到 y ，相应的表达式为 $d_{xy} v_{xi} v_{y,i+1}$ ；从 y 到 x ，则相应的表达式为 $d_{yx} v_{xi} v_{y,i-1}$ 。如果城市 x 和 y 在旅行顺序中相邻，则当 $v_{xi} v_{y,i+1} = 1$ 时，必有 $v_{xi} v_{y,i-1} = 0$ ；反之亦然。因此，有 $d_{xy} (v_{xi} v_{y,i+1} + v_{xi} v_{y,i-1}) = d_{xy}$ 。若定义 n 个城市各种可能的旅行路线长度为：

$$E_4 = \frac{1}{2} D \sum_{x=1}^n \sum_{y=1}^n \sum_{i=1}^n d_{xy} (v_{xi} v_{y,i+1} + v_{xi} v_{y,i-1}) \quad (6.30)$$

式中， D 为大于 0 的常数，当 E_4 最小时旅行路线最短。

综合以上 4 项能量，可得 TSP 问题的能量函数如下：

$$E = E_1 + E_2 + E_3 + E_4$$

$$\begin{aligned} &= \frac{1}{2}A \sum_{x=1}^n \sum_{i=1}^{n-1} \sum_{j=i+1}^n v_{xi} v_{xj} + \frac{1}{2}B \sum_{i=1}^n \sum_{x=1}^{n-1} \sum_{y=x+1}^n v_{xi} v_{yi} + \frac{1}{2}C \left(\sum_{x=1}^n \sum_{i=1}^n v_{xi} - n \right)^2 + \\ &\quad \frac{1}{2}D \sum_{x=1}^n \sum_{y=1}^n \sum_{i=1}^n d_{xy} (v_{xi} v_{y,i+1} + v_{xi} v_{y,i-1}) \end{aligned} \quad (6.31)$$

为从式(6.31)得到式(6.26)中的能量函数形式, 应使神经元 x_i 和 y_j 之间的权值和外部输入的偏置电流按下式给出:

$$\begin{cases} w_{x_i, y_j} = -2A\delta_{xy}(1-\delta_{ij}) - 2B\delta_{ij}(1-\delta_{xy}) - 2C - 2Dd_{xy}(\delta_{j,i+1} + \delta_{j,i-1}) \\ I_{xi} = 2cn \end{cases} \quad (6.32)$$

式中

$$\delta_{xy} = \begin{cases} 1 & x=y \\ 0 & x \neq y \end{cases} \quad \delta_{ij} = \begin{cases} 1 & i=j \\ 0 & i \neq j \end{cases}$$

网络构成后, 给定一个随机的初始输入, 便有一个稳定状态对应于一个旅行路线, 不同的初始输入所得到的旅行路线并不相同, 这些路线都是较佳的或最佳的。

用计算机模拟 CHNN 时, 将网络结构式(6.32)代入网络的运行方程 (6.19), 可得:

$$\begin{cases} c_{ij} \frac{du_{xi}}{dt} = -2A \sum_{j \neq i}^n v_{xj} - 2B \sum_{y \neq x}^n v_{yi} - 2c \left(\sum_{x=1}^n \sum_{j=1}^n v_{xj} - n \right) - 2D \sum_{y \neq x}^n d_{xy} (v_{y,i+1} + v_{y,i-1}) - \frac{u_{xi}}{R_{xi} C_{xi}} \\ v_{xi} = f(u_{xi}) = \frac{1}{2} \left[1 + \tanh \left(\frac{u_{xi}}{u_0} \right) \right] \end{cases}$$

式中, u_0 是初始值, 对上式编程可用软件实现求解 TSP 问题的 CHNN 算法。

图 6.6 给出用 CHNN 网解决 10 城市 TSP 问题的结果。图 6.6(a) 为最优解, 图 6.6(b) 为较佳解。

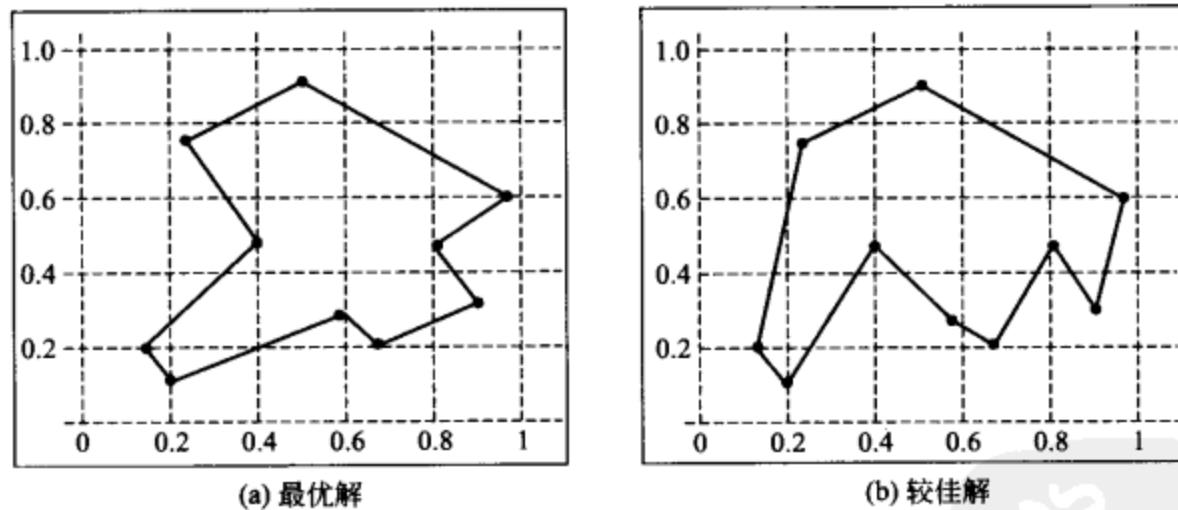


图 6.6 10 城市 TSP 问题的解

按照穷举法, 我国 31 个 (尚未计人香港特区) 直辖市、省会和自治区首府的巡回路径应有约 1.326×10^{32} 种。我国学者对中国旅行商 CTSP (Chinese TSP) 问题进行了大量的研究, 最新成果已达到 15449km。他们的做法是将两个最远城市间的距离定义为 1, 用归算法得到城市间的直达距离相对值 d_{xy} ($0 \leq d_{xy} \leq 1$)。近年的主要研究成果是: 采用 Greedy 组合算法, 从某一城市开始, 依次找出与之最靠近的城市, 然后连成一条闭合路径 15449km, 所得最短巡回路径为 17102km。采用 Hopfield 经典算法, 所得到的 400 个解中最短路径为 21777km。在 Hopfield 经典算法基础上增加约束条件, 最短路径为 16262km。在 Hopfield 经典算法基础上将所有城市分成三部分后, 求得最短路径为 15904km。

在实际应用中, TSP 类型的问题通常并不苛求非要得到最优解不可, 只要是接近最优即可满足要求。CHNN 用于优化问题时恰恰能做到这一点, 类似人脑分析这类问题时的特点。

J. J. Hopfield 的别具匠心的主要贡献在于, 他把能量函数的概念引入了神经网络, 从而把网络的拓扑结构与所要解决的问题联系起来, 把待优化的目标函数与网络的能量函数联系起来, 通过网络运行时能量函数自动最小化而得到问题的最优解, 从而开辟了求解优化问题的新途径。此外, 他还把神经网络与具体的模拟电子线路对应起来, 不仅易于理解, 更重要的是做到了理论与实践的有机结合。

6.4 双向联想记忆神经网络

联想记忆网络的研究是神经网络的重要分支, 在各种联想记忆网络模型中, 由 B. Kosko 于 1988 年提出的双向联想记忆 (bidirectional associative memory, 缩写为 BAM) 网络的应用最为广泛。前面介绍的 Hopfield 网可实现自联想, CNP 网可实现异联想, 而 BAM 网可实现双向异联想。BAM 网有离散型、连续型和自适应型等多种形式, 本节重点介绍常用的离散型 BAM 网络。

6.4.1 BAM 网结构与原理

BAM 网的拓扑结构如图 6.7 所示。该网是一种双层双向网络, 当向其中一层加入输入信号时, 另一层可得到输出。由于初始模式可以作用于网络的任一层, 信息可以双向传播, 所以没有明确的输入层或输出层, 可将其中的一层称为 X 层, 有 n 个神经元节点, 另一层称为 Y 层, 有 m 个神经元节点。两层的状态向量可取单极性二进制 0 或 1, 也可以取双极性离散值 1 或 -1。如果令由 X 到 Y 的权矩阵为 W , 则由 Y 到 X 的权矩阵便是其转置矩阵 W^T 。

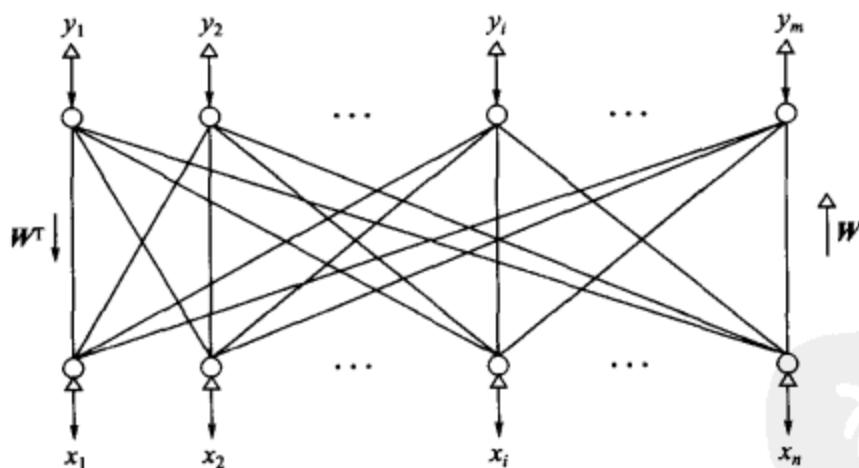


图 6.7 BAM 网拓扑结构

BAM 网实现双向异联想的过程是网络运行从动态到稳态的过程。对已建立权值矩阵的 BAM 网, 当将输入样本 X^P 作用于 X 侧时, 该侧输出 $X(1)=X^P$ 通过 W 阵加权传到 Y 侧, 通过该侧节点的转移函数 f_y 进行非线性变换后得到输出 $Y(1)=f_y[WX(1)]$; 再将该输出通过 W^T 阵加权从 Y 侧传回 X 侧作为输入, 通过 X 侧节点的转移函数 f_x 进行非线性变换后得到输出 $X(2)=f_x[W^TY(1)]=f_x\{W^T[f_y(WX(1))]\}$ 。这种双向往返过程一直进行到两侧所有神经元的状态均不再发生变化为止。此时的网络状态称为稳态, 对应的 Y 侧输出向量 Y^P

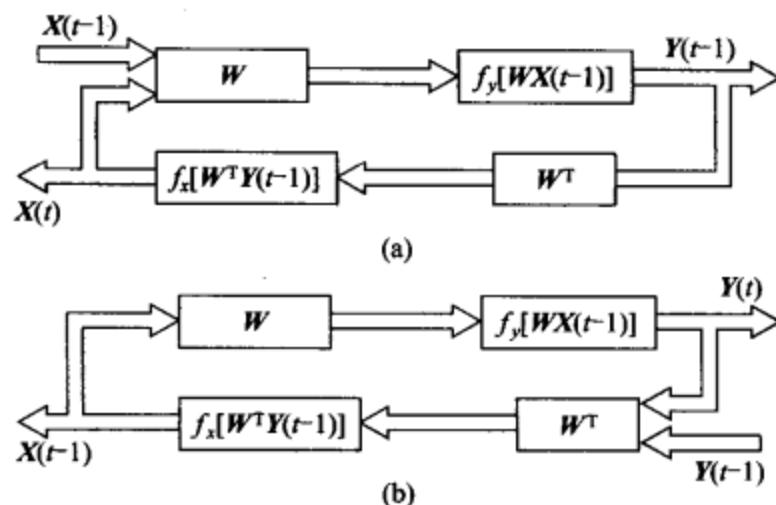


图 6.8 BAM 网的双向联想过程

便是模式 X^P 经双向联想后所得的结果。同理，如果从 Y 侧送入模式 Y^P ，经过上述双向联想过程， X 侧将输出联想结果 X 。这种双向联想过程可用图 6.8 表示。

由图 6.8(a) 和(b)，有：

$$X(t+1) = f_x\{W^T f_y[WX(t)]\} \quad (6.33a)$$

$$Y(t+1) = f_y\{W f_x[W^T Y(t)]\} \quad (6.33b)$$

对于经过充分训练的权值矩阵，当向 BAM 网络一侧输入有残缺的已存储模式时，网络经过有限次运行不仅能在另一侧实现正确的异联想，而且在输入侧重建了完整的输入模式。

6.4.2 能量函数与稳定性

若 BAM 网络的阈值为 0，能量函数定义为：

$$E = -\frac{1}{2}X^T W^T Y - \frac{1}{2}Y^T W X \quad (6.34)$$

BAM 网双向联想的动态过程就是能量函数量沿其状态空间中的离散轨迹逐渐减少的过程。当达到双向稳态时，网络必落入某一局部或全局能量最小点。

证明 式 (6.34) 中两项的计算结果均为标量，且有：

$$Y^T W X = (Y^T W X)^T = (W X)^T Y = X^T W^T Y$$

上式简化为：

$$E = -X^T W^T Y \quad (6.35)$$

当 X 侧节点状态变化时， ΔX 引起的能量变化 ΔE_X 为：

$$\Delta E_X = -\Delta X^T W^T Y = -\sum_{i=1}^n \Delta x_i \sum_{j=1}^m w_{ji} y_j = -\sum_{i=1}^n \Delta x_i \text{net}_{Xi}$$

当 BAM 网的转移函数取符号函数时，上式的分析可仿照 DHNN 网进行。可知对于 Δx_i 与 net_{Xi} 的任意组合均有 $\Delta E_X \leq 0$ ，其中 $\Delta E_X = 0$ 对应于 $\Delta x_i = 0, i = 1, 2, \dots, n$ 。

当 Y 侧节点状态变化时， ΔY 引起的能量变化 ΔE_Y 为：

$$\Delta E_Y = -\Delta Y^T W X = -\sum_{j=1}^m \Delta y_j \sum_{i=1}^n w_{ij} x_i = -\sum_{j=1}^m \Delta y_j \text{net}_{Yj}$$

同理，对于 Δy_j 与 net_{Yj} 的任意组合均有 $\Delta E_Y \leq 0$ ，而 $\Delta E_Y = 0$ 对应于 $\Delta y_j = 0, j = 1, 2, \dots, m$ 。

归纳以上分析，有：

$$\begin{cases} \Delta E < 0, & \Delta X \neq 0, \Delta Y \neq 0 \\ \Delta E = 0, & \Delta X = 0, \Delta Y = 0 \end{cases}$$

上式表明，BAM 网的能量在动态运行过程中不断下降，当网络达到能量极小点时即进入稳定状态，此时网络两侧的状态都不再变化。

以上证明过程对 BAM 网权矩阵的学习规则并未作任何限制，而且得到的稳定性的结论与状态更新方式为同步还是异步无关。考虑到同步更新比异步更新时能量变化大，收敛速度比串行异步方式快，故采常用同步更新方式。

6.4.3 BAM 网的权值设计

对于离散 BAM 网络，一般选转移函数 $f(\cdot) = \text{sgn}(\cdot)$ 。当网络只需存储一对模式 $(\mathbf{X}^1, \mathbf{Y}^1)$ 时，若使其成为网络的稳定状态，应满足条件：

$$\text{sgn}(\mathbf{W}\mathbf{X}^1) = \mathbf{Y}^1 \quad (6.36a)$$

$$\text{sgn}(\mathbf{W}^T\mathbf{Y}^1) = \mathbf{X}^1 \quad (6.36b)$$

容易证明，若 \mathbf{W} 是向量 \mathbf{Y}^1 和 \mathbf{X}^1 外积，即：

$$\mathbf{W} = \mathbf{Y}^1 \mathbf{X}^{1T}$$

$$\mathbf{W} = \mathbf{X}^1 \mathbf{Y}^{1T}$$

则式(6.36) 的条件必然成立。

当需要存储 P 对模式时，将以上结论扩展为 P 对模式的外积和，从而得到 Kosko 提出的权值学习公式：

$$\mathbf{W} = \sum_{p=1}^P \mathbf{Y}^p (\mathbf{X}^p)^T \quad (6.37a)$$

$$\mathbf{W}^T = \sum_{p=1}^P \mathbf{X}^p (\mathbf{Y}^p)^T \quad (6.37b)$$

用外积和法设计的权矩阵，不能保证任意 P 对模式的全部正确联想，但下面的定理表明，如对记忆模式对加以限制，用外积和法设计 BAM 网具有较好的联想能力。

定理 6.7 若 P 个记忆模式 \mathbf{X}^p , $p=1, 2, \dots, P$, $x \in \{-1, 1\}^n$ 两两正交，且权值矩阵 \mathbf{W} 按式(6.37) 得到，则向 BAM 网输入 P 个记忆模式中的任何一个 \mathbf{X}^p 时，只需一次便能正确联想起对应的模式 \mathbf{Y}^p 。

证明 若网络权值矩阵按外积和规则设计，当向其 X 侧输入某模式 \mathbf{X}^k 时，在 Y 侧应得到如下输出：

$$\begin{aligned} \mathbf{Y}(1) &= f_y(\mathbf{W}\mathbf{X}^k) = f_y\left(\sum_{p=1}^P \mathbf{Y}^p (\mathbf{X}^p)^T \mathbf{X}^k\right) \\ &= f_y\left[\mathbf{Y}^k (\mathbf{X}^k)^T \mathbf{X}^k + \sum_{p \neq k}^P \mathbf{Y}^p (\mathbf{X}^p)^T \mathbf{X}^k\right] \end{aligned}$$

当 P 个模式向量两两正交时，上式中的第二项应为零， Y 侧输出为：

$$\mathbf{Y}(1) = f_y[\mathbf{Y}^k (\mathbf{X}^k)^T \mathbf{X}^k] = f_y[\mathbf{Y}^k \| \mathbf{X}^k \|^2] = \text{sgn}[\mathbf{Y}^k \| \mathbf{X}^k \|^2] = \mathbf{Y}^k$$

定理得证。

当输入含有噪声的模式时，BAM 网需要经历一定的演变过程才能达到稳态，并分别在 X 侧和 Y 侧恢复模式对的本来面目。

例 6.3 某 BAM 网 X 层有 14×10 个节点， Y 层有 12×9 个节点。设网络已存储了三对联想字符，其中一对字符是 (S, E) 。当向网络的 X 侧输入有 40% 噪声的字符 S 时，网络开始动态演变过程。从图 6.9 可以看出，初始输入模式很难辨认，随着网络的运行，字符对 (S, E) 在网络 X 和 Y 两侧的往返过程中逐渐清晰，最终稳定于正确模式。

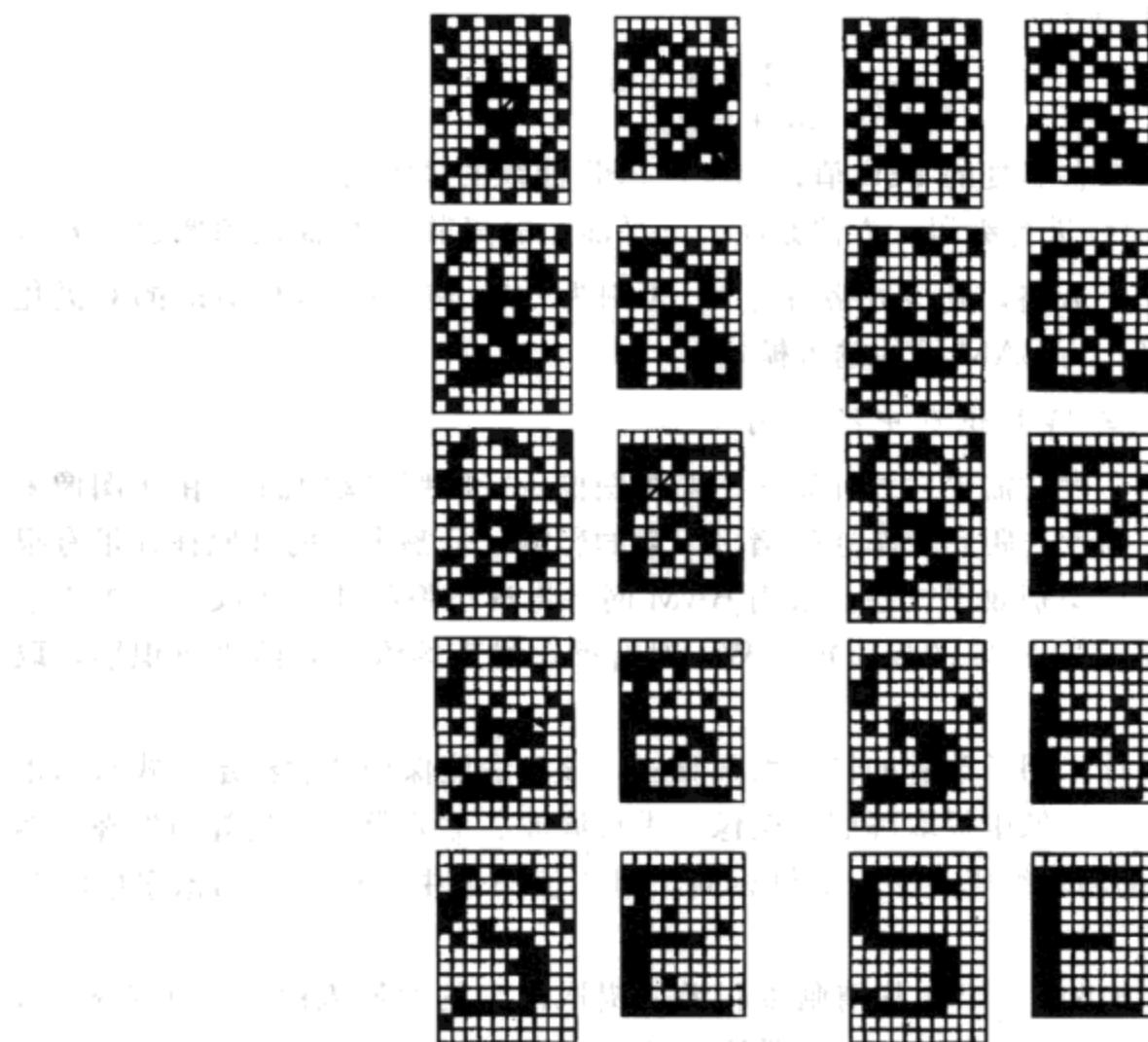


图 6.9 含噪声字符的联想过程

Kosko 已证明, BAM 网的存储容量为:

$$P_{\max} \leq \min(n, m)$$

为提高 BAM 网的存储容量和容错能力, 人们对 BAM 网提出多种改进算法和改进的网络结构, 如多重训练法、快速增强法、自适应 BAM 网络、竞争 BAM 网等。

6.4.4 BAM 网的应用

BAM 网络的设计比较简单, 只需由几组典型输入、输出向量构成权矩阵。运行时, 由实测到的数据向量与权矩阵作内积运算便可得到相应的信息输出。这是一种大规模并行处理大量数据的有效方法, 具有实时性和容错性。更具魅力的是, 这种联想记忆法无需对输入向量进行预处理, 便可直接进入搜索, 省去了编码与解码工作。下面给出两个应用实例:

6.4.4.1 BAM 网在功率谱密度函数分类中的应用

工业生产过程中, 经常要求对检测到的曲线进行分类, 以便据此作出某种判断。1989 年 Mathai G. 等应用 BAM 网成功地解决了纤维制造过程中的功率谱密度函数 PSD 分类问题。由于 PSD 具有较大的变异性, 因此即使是同一类谱, 用传统方法进行分类也很困难。Mathai 对纤维制造中所得到的各种 PSD 曲线进行分析后, 发现只有两个典型类别: 第 1 类以 26Hz 及 14Hz 附近出现两个谱峰为特征; 第 2 类以 32Hz 及 39Hz 处有两个谱峰为其特性。因此, 分类的判据便是 PSD 曲线中谱峰出现的频率值。如果将两类典型 PSD 曲线作为 BAM 网的记忆模式样本, 通过同维模式的自联想即可完成对其他 PSD 模式的分类工作。

为提高分类准确率, 需先对 PSD 进行预处理, 以增强谱峰, 突出特征。处理方法是用

非线性函数对离散的 PSD 进行归一化：

$$\hat{y}(v_i) = \frac{\ln[1+y(v_i)]}{\ln(1+y_{\max})}$$

式中， $y(v_i)$ 为各频率点 v_i 处的 PSD 值； y_{\max} 为 PSD 曲线的最大值。

对归一化后的 PSD 曲线进行编码，方法是将 $y-v$ 平面上的离散 PSD 曲线图像置于 $r \times s$ 网格中，若 \hat{y} 点落入某个小方格，则该方格值为 1，否则为 -1。将 $r \times s$ 网格对应的双极化矩阵各行首尾连接后即可作为 BAM 网的输入模式。

6.4.4.2 BAM 网在汽车牌照识别中的应用

公安部门在缉查失窃车辆时需要对过往车辆的监视图像进行牌照自动识别。由于图像采集质量受天气阴晴、拍摄角度与距离及车速等诸多因素的影响，分割出来的牌照往往带有很大的噪声，用传统方法进行识别效果较差。采用 BAM 网络将汽车牌照涉及的汉字、英文字母及数字作为记忆模式存入 24×24 的权值矩阵 W ，对有严重噪声的汽车牌照进行识别，取得了较好的效果。

该识别系统的权值设计采用了改进的快速增强算法，该算法能保证任意给定模式对的正确联想。识别时首先从汽车图像中提取牌照子图像，进行滤波、缩放及二值化等预处理，然后对 24×24 的牌照图像编码。编码方法与上例相同，但二值图像中灰度为 0 的像素应将灰度值变为 1。

阴雨天及大于 45° 角拍摄的汽车图片清晰度很差，提取出来的牌照人眼难以正确辨认，采用基于 BAM 网络的识别系统可取得令人满意的效果。

6.5 随机神经网络

如果将 BP 算法中的误差函数看作一种能量函数，则 BP 算法通过不断调整网络参数使其能量函数按梯度单调下降，而反馈网络使通过动态演变过程使网络的能量函数沿着梯度单调下降，在这一点上两类网络的指导思想是一致的。正因如此，常常导致网络落入局部极小点而达不到全局最小点。对于 BP 网，局部极小点意味着训练可能不收敛；对于 Hopfield 网，则得不到期望的最优解。导致这两类网络陷入局部极小点的原因是，网络的能量函数或能量函数是具有多个极小点的非线性空间，而所用的算法却一味追求网络误差或能量函数的单调下降。也就是说，算法赋予网络的是只会“下山”而不会“爬山”的能力。如果为具有多个局部极小点的系统打一个形象的比喻，设想托盘上有一个凸凹不平的多维能量曲面，若在该曲面上放置一个小球，它在重力作用下，将滚入最邻近的一个低谷（局部最小点）而不能自拔。但该低谷不一定就是曲面上最低的那个低谷（全局最小点）。因此，局部极小问题只能通过改进算法来解决。

本章要介绍的随机网络可赋予网络既能“下坡”也能“爬山”的本领，因而能有效地克服上述缺陷。随机网络与其他神经网络相比有两个主要区别：①在学习阶段，随机网络不像其他网络那样基于某种确定性算法调整权值，而是按某种概率分布进行修改；②在运行阶段，随机网络不是按某种确定性的网络方程进行状态演变，而是按某种概率分布决定其状态的转移。神经元的净输入不能决定其状态取 1 还是取 0，但能决定其状态取 1 还是取 0 的概率。这就是随机神经网络算法的基本概念。图 6.10 给出了随机网络算法与梯度下降算法区别的示意图。

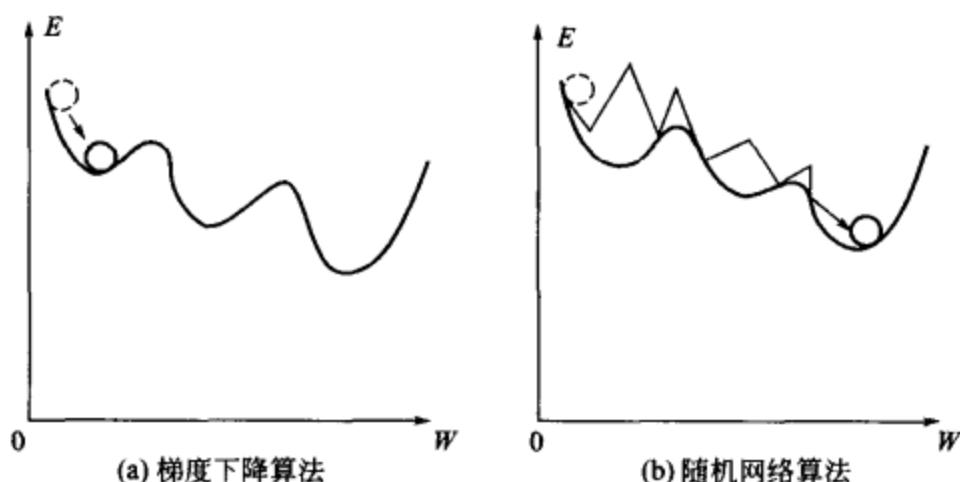


图 6.10 随机网络算法与梯度下降算法的区别

6.5.1 模拟退火原理

模拟退火算法是随机网络中解决能量局部极小问题的一个有效方法，其基本思想是模拟金属退火过程。金属退火过程大致是，先将物体加热至高温，使其原子处于高速运动状态，此时物体具有较高的内能；然后，缓慢降温，随着温度的下降，原子运动速度减慢，内能下降；最后，整个物体达到内能最低的状态。模拟退火过程相当于沿水平方向晃动托盘，温度高则意味着晃动的幅度大，小球肯定会从任何低谷中跳出，而落入另一个低谷。这个低谷的高度（网络能量）可能比小球原来所在低谷的高度低（网络能量下降），但也可能反而比原来高（能量上升）。后一种情况的出现，从局部和当前来看，这个运动方向似乎是错误的；但从全局和发展的角度看，正是由于给小球赋予了“爬山”的本事，才使它有可能跳出局部低谷而最终落入全局低谷。当然，晃动托盘的力度要合适，并且还要由强至弱（温度逐渐下降），小球才不致因为有了“爬山”的本领而越爬越高。

在随机网络学习过程中，先令网络权值作随机变化，然后计算变化后的网络能量函数。网络权值的修改应遵循以下准则：若权值变化后能量变小，则接受这种变化；否则也不应完全拒绝这种变化，而是按预先选定的概率分布接受权值的这种变化。其目的在于赋予网络一定的“爬山”能力。实现这一思想的一个有效方法就是 Metropolis 等提出的模拟退火算法。

设 X 代表某一物质体系的微观状态（一组状态变量，如粒子的速度和位置等）， $E(X)$ 表示该物质在某微观状态下的内能，对于给定温度 T ，如果体系处于热平衡状态，则在降温退火过程中，其处于某能量状态的概率与温度的关系遵循 Boltzmann 分布规律。分布函数为：

$$P(E) \propto \exp\left[-\frac{E(X)}{KT}\right] \quad (6.38)$$

式中， K 为玻尔兹曼常数。在下面讨论中把常数 K 合并到温度 T 中。

由式(6.38)可以看出，当温度一定时，物质体系的能量越高，其处于该状态的概率就越低，因此物质体系的内能趋向于向能量降低的方向演变。如给定不同的温度，上式表示的曲线变化如图 6.11 所示：当物体温度 T 较高时， $P(E)$ 对能量 E 的大小不敏感，因此物体处于高能或低能状态的概率相差不大；随着温度 T 的

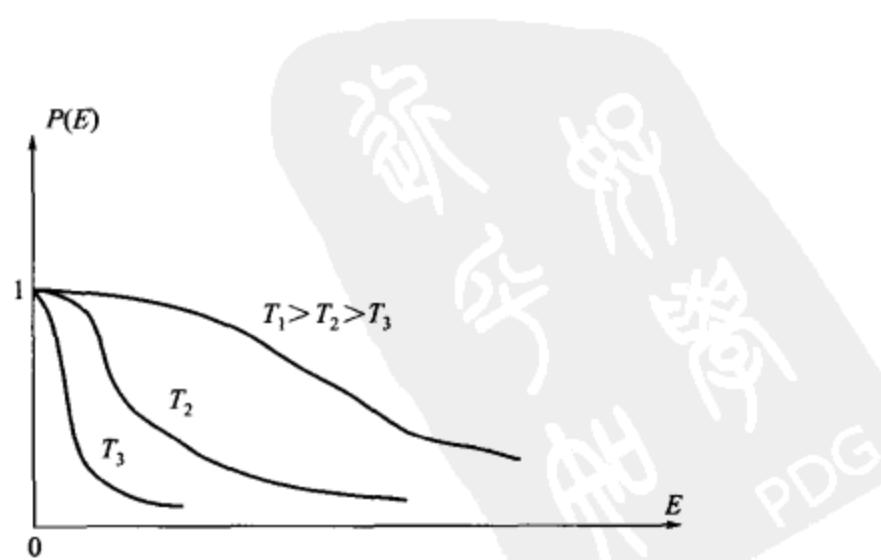


图 6.11 能量状态的概率曲线与温度的关系

下降，物质处于高能状态的概率随之减小，而处于低能状态的概率增加；当温度接近 0 时，物体处于低能状态的概率接近 1。由此可见，温度参数 T 越高，状态越容易变化。为了使物质体系最终收敛到低温下的平衡态，应在退火开始时设置较高的温度，然后逐渐降温，最后物质体系将以相当高的概率收敛到最低能量状态。

用随机神经网络解决优化问题时，通过数学算法模拟了以上退火过程。模拟方法是，定义一个网络温度以模仿物质的退火温度，取网络能量为欲优化的目标函数。网络运行开始时温度较高，调整权值时允许目标函数偶尔向增大的方向变化，以使网络能跳出那些能量的局部极小点。随着网络温度不断下降至 0，最终以概率 1 稳定在其能量函数的全局最小点，从而获得最优解。

6.5.2 Boltzmann 机

G. E. Hinton 等于 1983~1986 年提出一种称为 Boltzmann 机的随机神经网络。在这种网络中神经元只有两种输出状态，即单极性二进制的 0 或 1。状态的取值根据概率统计法则决定，由于这种概率统计法则的表达形式与著名统计力学家 L. Boltzmann 提出的 Boltzmann 分布类似，故将这种网络取名 Boltzmann Machine (BM)。

6.5.2.1 BM 网络的拓扑结构与运行原理

(1) BM 网络的拓扑结构 BM 网络的拓扑结构比较特殊，介于 DHNN 网的全互连结构与 BP 网的层次结构之间。从形式上看，BM 网络与单层反馈网络 DHNN 网相似，具有对称权值，即 $w_{ij} = w_{ji}$ ，且 $w_{ii} = 0$ 。但从神经元的功能上看，BM 网络与三层 BP 网相似，具

有输入节点、隐节点和输出节点。一般将输入输出节点称为可见节点，而将隐节点称为不可见节点。训练时输入输出节点接收训练集样本，而隐节点主要起辅助作用，用来实现输入与输出之间的联系，使训练集能在可见单元再现。BM 网络的三类节点之间没有明显的层次，连接形式可用图 6.12 中的有向图表示。

(2) 神经元的转移概率函数 设 BM 网络中单个神经元的净输入为：

$$\text{net}_j = \sum_i (w_{ij}x_i - T_j)$$

与 DHNN 网不同的是，以上净输入并不能通过符号转移函数直接获得确定的输出状态，实际的输出状态将按某种概率发生，神经元的净输入可通过 S 型函数获得输出某种状态的转移概率：

$$P_j(1) = \frac{1}{1 + e^{-\text{net}_j/T}} \quad \begin{array}{l} \text{输出概率跟hopfield不一样} \\ \text{概率跟权重, 输入, 阈值有关} \\ \text{还跟当前温度T有关} \end{array} \quad (6.39)$$

式中， $P_j(1)$ 表示神经元 j 输出状态为 1 的概率，状态为 0 的概率应为：

$$P_j(0) = 1 - P_j(1) \quad 0,1 \text{ 分布概率符合玻尔兹曼分布}$$

可以看出，如果净输入为 0，则 $P_j(1) = P_j(0) = 0.5$ 。净输入越大，神经元状态取 1 的概率越大；净输入越小，神经元状态取 0 的概率越大。而温度 T 的变化可改变概率曲线的形状。从图 6.13 可以看出，对于同一净输入，温度 T 较高时概率曲线变化平缓，对于同一净输入 $P_j(1)$ 与 $P_j(0)$ 的差别小；而温度 T 低时概率曲线变陡峭，对于同一净输入 $P_j(1)$

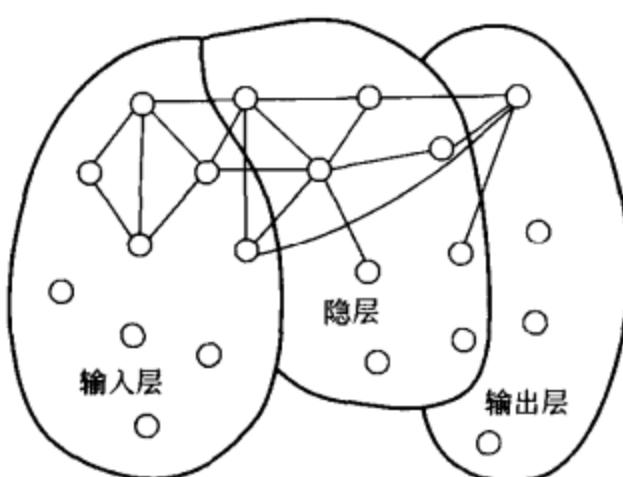


图 6.12 BM 网络的拓扑结构

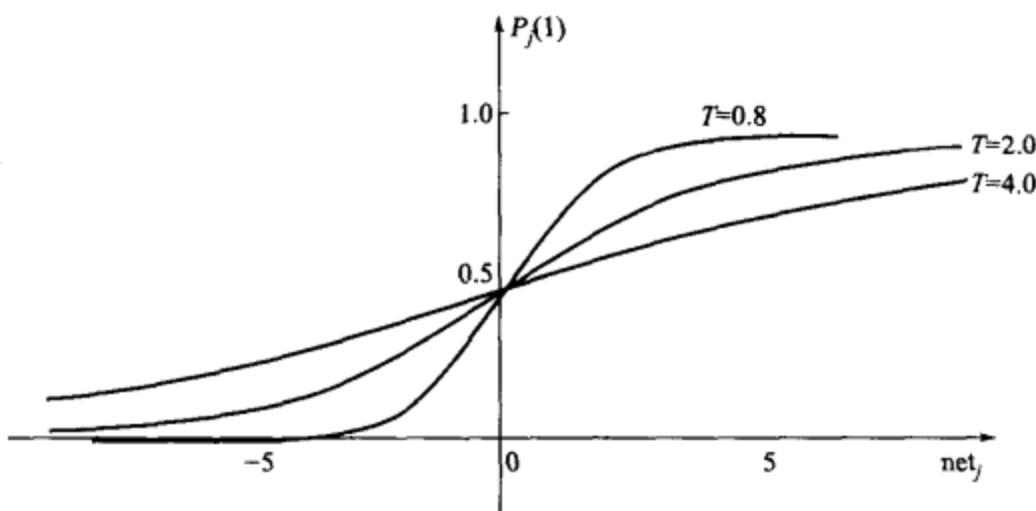


图 6.13 神经元状态概率与净输入和温度的关系

与 $P_j(0)$ 的差别大。当 $T=0$ 时, 式(6.39) 中的概率函数退化为符号函数, 神经元输出状态将无随机性可言。

(3) 网络能量函数与运行的搜索机制 BM 网络采用了与 DHNN 网相同的能量函数描述网络状态:

$$\begin{aligned} E(t) &= -\frac{1}{2} \mathbf{X}^T(t) \mathbf{W} \mathbf{X}(t) + \mathbf{X}^T(t) \mathbf{T} \\ &= -\frac{1}{2} \sum_{j=1}^n \sum_{i=1}^n w_{ij} x_i x_j + \sum_{i=1}^n T_i x_i \end{aligned} \quad (6.40)$$

设 BM 网络按异步方式工作, 每次第 j 个神经元改变状态, 根据式(6.9), 有:

$$\Delta E(t) = -\Delta x_j(t) \text{net}_j(t) \quad (6.41)$$

下面对式(6.41) 的各种情况进行讨论:

① 当 $\text{net}_j > 0$ 时, 由式(6.39) 有 $P_j(1) > 0.5$, 即神经元 j 有较大的概率取 $x_j = 1$ 。若原来 $x_j = 1$, 则 $\Delta x_j = 0$, 从而 $\Delta E = 0$; 若原来 $x_j = 0$, 则 $\Delta x_j = 1$, 从而 $\Delta E < 0$ 。
超过50%的概率能量减小

② 当 $\text{net}_j < 0$ 时, 由式(6.39) 有 $P_j(1) < 0.5$, 即神经元 j 有较大的概率取 $x_j = 0$ 。若原来 $x_j = 0$, 则 $\Delta x_j = 0$, 从而 $\Delta E = 0$; 若原来 $x_j = 1$, 则 $\Delta x_j = -1$, 从而 $\Delta E < 0$ 。

以上对各种可能情况讨论的结果与 6.1.2.2 中的讨论结果一致。但需要注意的是, 对于 BM 网络, 随着网络状态的演变, 从概率的意义上网络的能量总是朝着减小的方向变化。这就意味着尽管网络能量的总趋势是向着减小的方向演变, 但不排除在有些步神经元状态可能会按小概率取值, 从而使网络能量暂时增加。正是因为有了这种可能性, BM 网络才具有了从局部极小的低谷中跳出的“爬山”能力, 这一点是 BM 网络与 DHNN 网能量变化的根本区别。由于采用了神经元状态按概率随机取值的工作方式, BM 网络的能量具有不断跳出位置较高的低谷搜索位置较低的新低谷的能力。这种运行方式称为搜索机制, 即网络在运行过程中不断地搜索更低的能量极小值, 直到达到能量的全局最小。从模拟退火法的原理可以看出, 温度 T 不断下降可使网络能量的“爬山”能力由强减弱, 这正是保证 BM 网络能成功搜索到能量全局最小的有效措施。

(4) BM 网络的 Boltzmann 分布 设 $x_j = 1$ 时对应的网络能量为 E_1 , $x_j = 0$ 时网络能量为 E_0 。根据前面的分析结果, 当 x_j 由 1 变为 0 时, 有 $\Delta x_j = -1$, 于是有:

$$E_0 - E_1 = \Delta E = -(-1) \text{net}_j = \text{net}_j$$

式(6.39) 变为:

$$P_j(1) = \frac{1}{1+e^{-\text{net}_j/T}} = \frac{1}{1+e^{-\Delta E/T}} \quad (6.42)$$

$$P_j(0) = 1 - P_j(1) = \frac{e^{-\Delta E/T}}{1+e^{-\Delta E/T}}$$

两式相除，有：

$$\frac{P_j(0)}{P_j(1)} = e^{-\Delta E/T} = e^{-(E_0 - E_1)/T} = \frac{e^{-E_0/T}}{e^{-E_1/T}} \quad (6.43)$$

当 x_j 由 0 变为 1 时，有 $\Delta x_j = 1$ ，于是有：

$$E_1 - E_0 = \Delta E = -\text{net}_j$$

式(6.39) 变为：

$$P_j(1) = \frac{1}{1+e^{-\text{net}_j/T}} = \frac{1}{1+e^{\Delta E/T}} \quad (6.44)$$

$$P_j(0) = 1 - P_j(1) = \frac{e^{\Delta E/T}}{1+e^{\Delta E/T}}$$

两式相除，仍可得到式(6.43)：

$$\frac{P_j(0)}{P_j(1)} = e^{\Delta E/T} = e^{(E_1 - E_0)/T} = \frac{e^{-E_0/T}}{e^{-E_1/T}}$$

将式(6.43) 推广到网络中任意两个状态出现的概率与对应能量之间的关系，有：

$$\frac{P(\alpha)}{P(\beta)} = \frac{e^{-E_\alpha/T}}{e^{-E_\beta/T}} \quad (6.45)$$

式(6.45) 就是著名的 Boltzmann 分布。从式中可以得出两点结论：①BM 网络处于某一状态的概率主要取决于此状态下的能量 E ，能量越低，概率越大；②BM 网络处于某一状态的概率还取决于温度参数 T ，温度越高，不同状态出现的概率越接近，网络能量较易跳出局部极小而搜索全局最小；温度低则情况相反。这正是采用模拟退火方法搜索全局最小的原因所在。

用 BM 网络进行优化计算时，可构造一个类似于式(6.40) 的目标函数作为网络的能量函数。为防止目标函数陷入局部极小，采用上述模拟退火算法进行最优解的搜索。即搜索开始时将温度设置得很高，此时神经元为 1 状态或 0 状态的机会几乎相等，因此网络能量可以达到任意可能的状态，包括局部极小或全局最小。当温度下降时，不同状态的概率发生变化，能量低的状态出现的概率大，而能量高的状态出现的概率小。当温度逐渐降至 0 时，每个神经元要么只能取 1 要么只能取 0，此时网络的状态就“凝固”在目标函数的全局最小附近。对应的网络状态就是优化问题的最优解。

用 BM 网络进行联想时，可通过学习用网络稳定状态的概率来模拟训练集样本的出现概率。根据学习类型，BM 网络可分为自联想和异联想两种情况，如图 6.14 所示。自联想型 BM 网络中的可见节点 V 与 DHNN 网中的节点相似，既是输入节点又是输出节点，隐节点 H 的数目由学习的需要而定，最少可以为 0。异联想 BM 网络中的可见节点 V 需按功能分为输入节点组 I 和输出节点组 O 。

6.5.2.2 BM 网络的学习算法

(1) 学习过程 通过有导师学习，BM 网络可以对训练集中各模式的概率分布进行模拟，从而实现联想记忆。学习的目的是通过调整网络权值使训练集中的模式在网络状态中以相同的概率再现。学习过程可分为两个阶段：第一阶段称为正向学习阶段或输入期，即向网

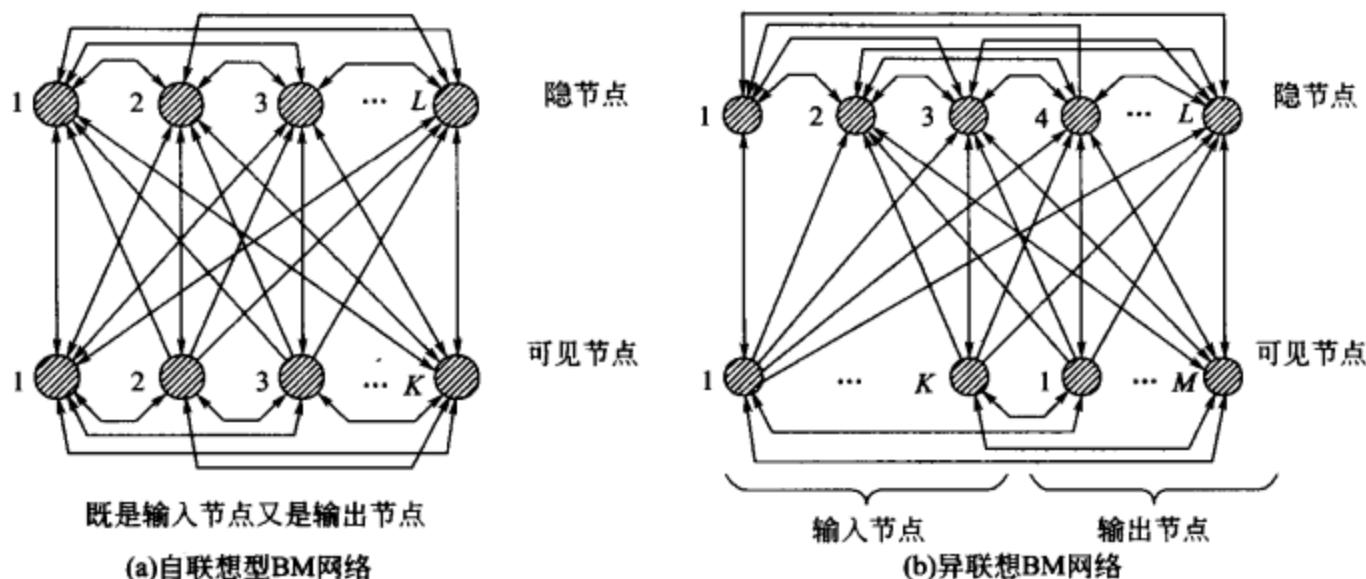


图 6.14 BM 网络学习网络结构

络输入一对输入输出模式，将网络输入输出节点的状态“钳制”到期望的状态，而让隐节点自由活动，以捕捉模式对之间的对应规律；第二阶段称为反向学习阶段或自由运行期，对于异联想学习，用输入模式“钳住”输入节点而让隐节点和输出节点自由活动，对于自联想学习，让可见节点和隐节点都自由活动，以体现网络对输入输出对应规律的模拟情况。输入输出的对应规律表现为网络达到热平衡时，相连节点状态同时为 1 的平均概率。期望对应规律与模拟对应规律之间的差别就表现为两个学习阶段所对应的平均概率的差值，此差值便作为权值调整的依据。设 BM 网络隐节点数为 m ，可见节点数为 n ，则可见节点可表达的状态 X （对于异联想， X 中部分分量代表输入模式，另一部分代表输出模式）共有 2^n 种。设训练集提供了 P 对模式，一般有 $P < n$ 。训练集用一组概率分布表示各模式对出现的概率：

$$P(X^1), P(X^2), \dots, P(X^P)$$

以上也是在正向学习阶段期望的网络状态概率分布。当网络自由运行时，相应模式出现的概率为：

$$P'(X^1), P'(X^2), \dots, P'(X^P)$$

训练的目的是使以上两组概率分布相同。

(2) 网络热平衡状态 为统计以上概率，需要反复使 BM 网络按模拟退火算法运行并达到热平衡状态。具体步骤如下：

① 在正向学习阶段，用一对训练模式 X^p 钳住网络的可见节点；在反向学习阶段，用训练模式中的输入部分钳住可见节点中的输入节点。

② 随机选择自由活动节点 j ，使其更新状态：

$$s_j(t+1) = \begin{cases} 1, & s_j(t) = 0 \\ 0, & s_j(t) = 1 \end{cases}$$

③ 计算节点 j 状态更新而引起的网络能量变化 $\Delta E_j = -\Delta s_j(t) \text{net}_j(t)$ 。

④ 若 $\Delta E_j < 0$ ，则接受状态更新；若 $\Delta E_j > 0$ ，当 $P[s_j(t+1)] > \rho$ 时接受新状态，否则维持原状态。 $\rho \in (0, 1)$ 是预先设置的数值，在模拟退火过程中，温度 T 随时间逐渐降低，从式(6.5)可以看出，对于常数 ρ ，为使 $P[s_j(t+1)] > \rho$ ，必须使 ΔE_j 也在训练中不断减小，因此网络的爬山能力是不断减小的。

⑤ 返回步骤②~④直到自由节点被全部选择一遍。

⑥ 按事先选定的降温方程降温，退火算法的降温规律没有统一规定，一般要求初始温

度 T_0 足够高，降温速度充分慢，以保证网络收敛到全局最小。下面给出两种降温方程：

$$T(t) = \frac{T_0}{1 + \ln t} \quad (6.46)$$

$$T(t) = \frac{T_0}{1 + t} \quad (6.47)$$

⑦ 返回步骤②~⑥直到对所有自由节点均有 $\Delta E_j = 0$ ，此时认为网络已达到热平衡状态。此状态可供学习算法中统计任意两个节点同时为 1 的概率时使用。

(3) 权值调整算法与步骤 BM 网络的学习算法步骤如下：

① 随机设定网络的初始权值 $w_{ij}(0)$ 。

② 正向学习阶段按已知概率 $P(\mathbf{X}^p)$ 向网络输入学习模式 \mathbf{X}^p , $p=1, 2, \dots, P$ 。在 \mathbf{X}^p 的约束下按上述模拟退火算法运行网络到热平衡状态，统计该状态下网络中任意两节点 i 与 j 同时为 1 的概率 p_{ij} 。

③ 反向学习阶段在无约束条件下或在仅输入节点有约束条件下运行网络到热平衡状态，统计该状态下网络中任意两节点 i 与 j 同时为 1 的概率 p'_{ij} 。

④ 权值调整算法为：

$$\Delta w_{ij} = \eta(p_{ij} - p'_{ij}) \quad \eta > 0 \quad (6.48)$$

⑤ 重复以上步骤直到 p_{ij} 与 p'_{ij} 充分接近。

6.6 递归神经网络

反馈网络有两类基本功能：一类是联想记忆或优化，如上述网络模型；另一类是动态信号与系统的处理。本节将要讨论的网络模型即为具有第二类功能的反馈网络。递归神经网络具有一个或多个反馈环，由于动态系统的一个主要特点是其输出不仅与当前输入有关，而且与系统前些时刻的输入、系统状态以及系统输出都有关，因而模拟这类系统的递归神经网络需要加入用延时环节表示的短时记忆。

用于动态信号处理的递归网络依时序响应外部的输入信号，在输入-输出非线性映射方面比静态映射网络有着更大的应用范围，在非线性动态系统建模与预测、通信信道自适应平衡、语音处理、设备控制、故障诊断等方面得到广泛应用。

6.6.1 递归网络模型

递归网络的结构布局有多种形式，但共同特点是结合一个静态多层感知器，并利用多层感知器的非线性映射能力。

递归网络的反馈可以是局部的或全局的，给定多层感知器作为基本模块。全局反馈可以有不同的形式，如从多层感知器的输出层反馈到输入层，或从隐层反馈到输入层。当感知器有多个隐层时，全局反馈的可能形式将更为丰富，下面简要介绍其中三种模型：

6.6.1.1 输入输出递归网络通用模型

图 6.15 给出一种在静态多层感知器基础上加入延时单元的通用递归网络模型。其中外部单输入信号和网络的单输出信号都通过延迟单元展成空间表示后再送给多层感知器作为输入。

图 6.15 中的递归网络模型也可以称为有外部输入的非线性自回归模型 (nonlinear autoregressive with exogenous inputs model, NARX)。网络的输入层接受两类信号：来自网

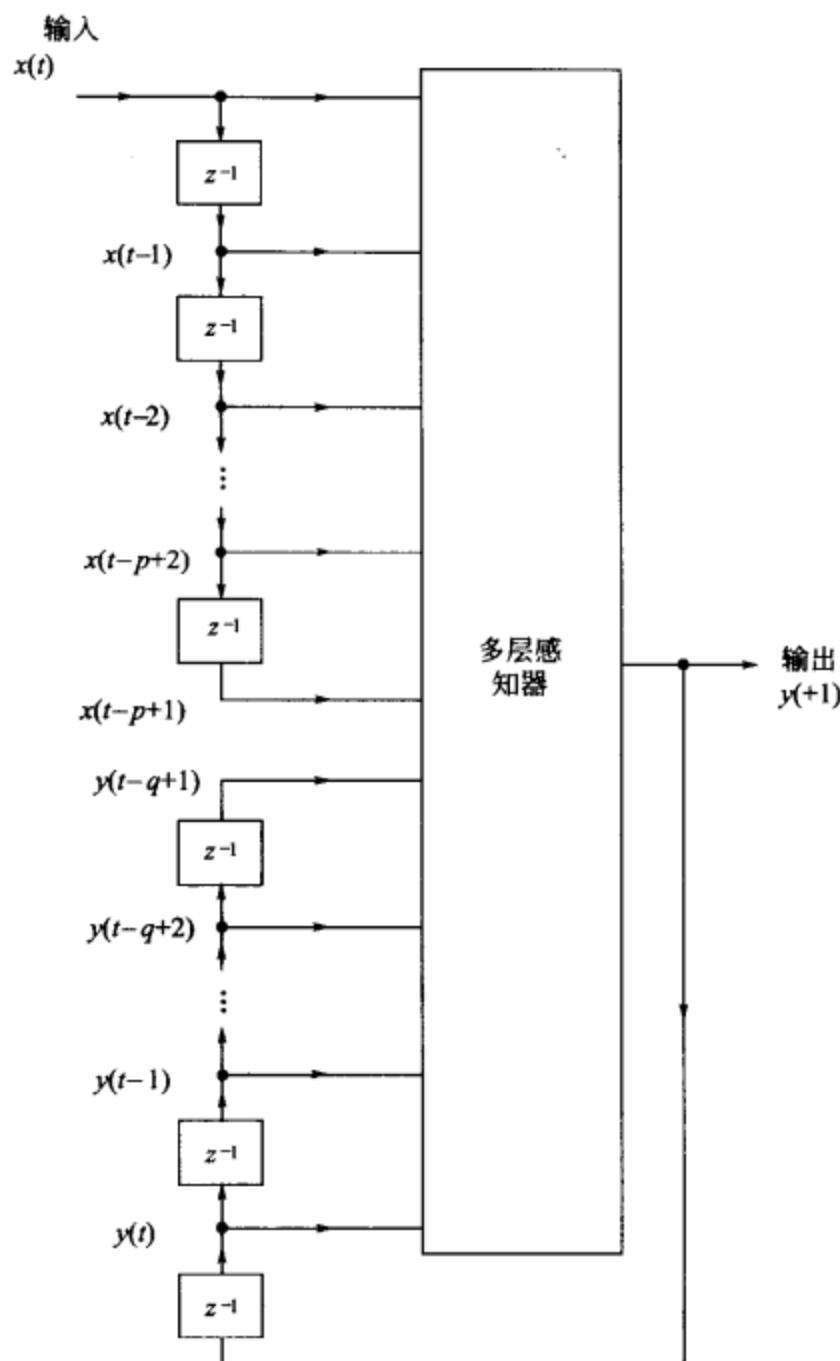


图 6.15 一种通用递归网络模型

络外部的输入表示为 $x(n), x(n-1), \dots, x(n-p+1)$; 来自网络输出的反馈信号表示为 $y(n), y(n-1), \dots, y(n-q+1)$ 。网络作为非线性系统的动态描述为:

$$y(n+1) = F[y(n), \dots, y(n-q+1), x(n), \dots, x(n-p+1)] \quad (6.49)$$

6.6.1.2 Elman 网络模型

J. L. Elman 于 1990 年提出一种简单的递归网络模型, 如图 6.16 所示。该网络输入层接受两种信号: 一种是外加输入 $U(t)$; 一种是来自隐层的反馈信号 $X^c(t)$ 。将接受反馈的节点称为联系单元 (context unit), $X^c(t)$ 表示联系单元在时刻 t 的输出; 隐层输出为 $X(t+1)$, 输出为 $Y(t+1)$ 。当输出节点采用线性转移函数时, 有如下方程:

隐单元	$X(t+1) = F[X^c(t), U(t)]$
联系单元	$X^c(t) = X(t-1)$
输出单元	$Y(t+1) = WX(t+1)$

6.6.1.3 递归多层感知器模型

递归多层感知器 (recurrent multilayer perceptron, RMLP) 的结构如图 6.17 所示。该模型有一个或多个隐层, 每一个计算层均对其邻近层有一个反馈。

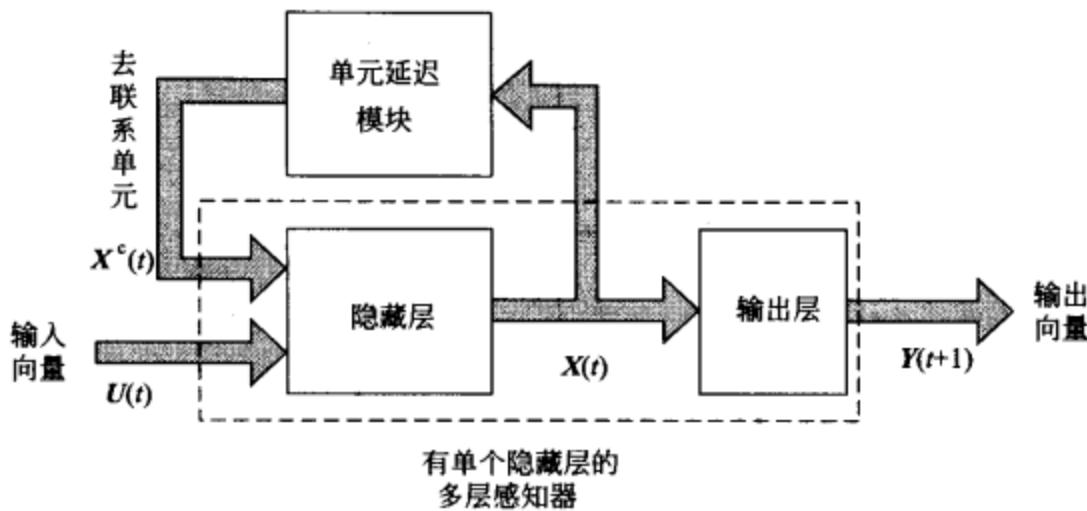


图 6.16 Elman 网络模型

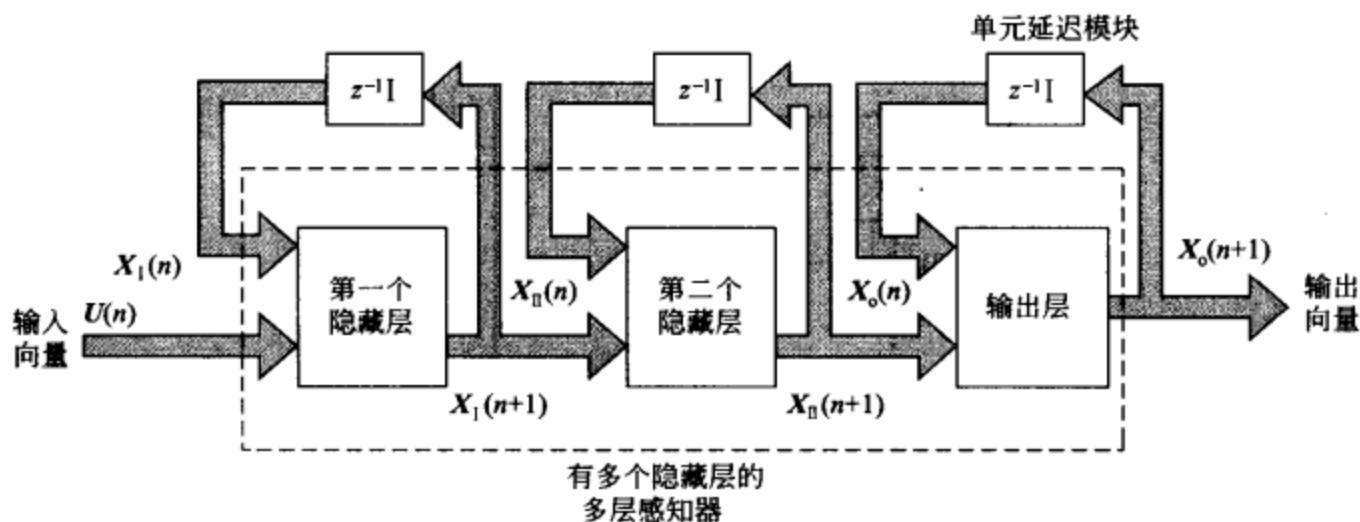


图 6.17 递归多层感知器模型

第一个隐层的输出用向量 $X_I(t)$ 表示，第二个隐层的输出用向量 $X_{II}(t)$ 表示，以此类推。输出层的输出用向量 $X_o(t)$ 表示。RMLP 对输入的动态响应可用下列联立方程组描述：

$$\begin{aligned} X_I(t+1) &= F_I[X_I(t), U(t)] \\ X_{II}(t+1) &= F_{II}[X_{II}(t), X_I(t)] \\ X_o(t+1) &= F_o[X_o(t), X_K(t)] \end{aligned}$$

式中， $F_I(\cdot, \cdot)$, $F_{II}(\cdot, \cdot)$, ..., $F_o(\cdot, \cdot)$ 分别为各隐层和输出层节点的转移函数； K 为隐层数。

6.6.2 递归网络的学习算法

从上节的介绍可知，针对动态系统的递归网络是在多层感知器基础上增加延时和反馈单元而形成的，其学习算法有以下两种方式：

(1) 分时段 (epochwise) 训练 在不同的时段内，待模拟的系统可以从许多不同的初始状态出发并达到不同的稳态，因此每一个“时段”对应于第 3 章中普通多层感知器的一个训练模式。在某一给定时段内，递归网络的训练从一个初始状态出发，到达一个新的状态后停止，然后对于下一个时段训练又从一个新的初始状态出发。分时段训练适合于对动态系统的有限状态的模拟，用于离线训练的场合。

(2) 连续训练 对于需要在线学习的情况或无法区分时段的情况，网络学习和信号处理必须同时进行，学习过程永不停止。例如，采用递归网络对语音信号建模。

6.6.2.1 历时反向传播学习算法

历时反向传播学习算法 (back-propagation through time, BPTT) 是对标准 BP 算法的扩展，其思路是将网络的时序处理展开成一个分层的前向网络，下面以图 6.18 中给出的具有 2 个神经元的递归网络为例说明 BPTT 算法的思路。

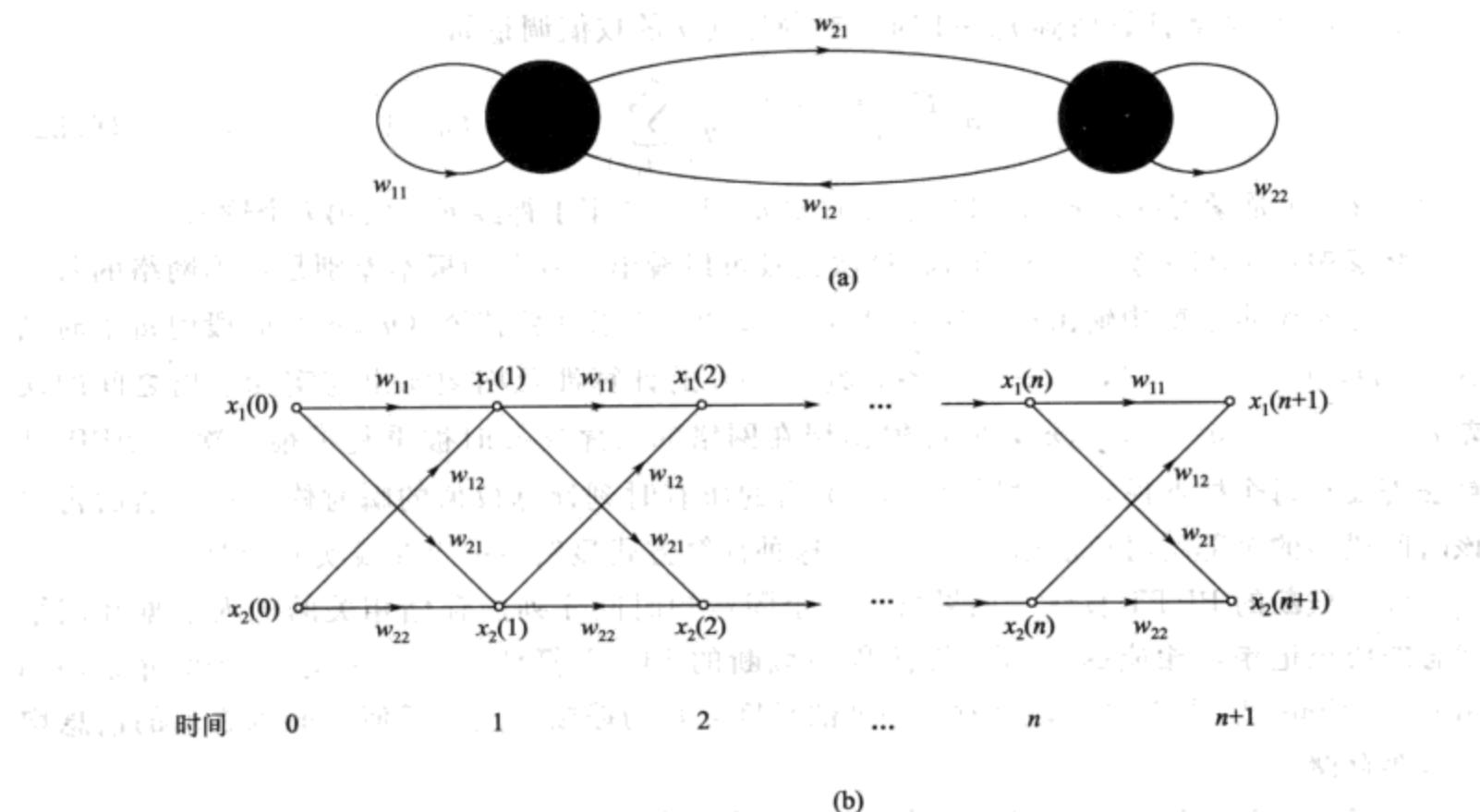


图 6.18 具有 2 个神经元的递归网络

BPTT 算法通过一步一步地展开网络的时序操作，得到一个图 6.18(b) 所示的前向网络。设 $n=0$ 为训练的起始时间，则每一步时序操作都复制新的一层加入网络。

(1) 逐时段 (epochwise) 训练的 BPTT 算法 采用逐时段训练方式，将训练数据分为 P 组，每组为一个训练时段，设 n_0 为一个训练时段的起始时间， n_1 为一个训练时段的结束时间，在这个时段内定义目标函数为：

$$E_{\text{总}}(n_0, n_1) = \frac{1}{2} \sum_{n=n_0}^{n_1} \sum_{j \in A} e_j^2(n) = \frac{1}{2} \sum_{n=n_0}^{n_1} \sum_{j \in A} [d_j(n) - y_j(n)]^2 \quad (6.50)$$

式中， $e_j(n)$ 为瞬时误差信号； A 为有外加给定输入的节点的集合。

与 BP 算法权值调整思路一样，逐时段训练 BPTT 算法采用目标函数的负梯度计算每个时刻的权值修正量，神经元 j 的局部梯度定义为：

$$\delta_j(n) = -\frac{\partial E_{\text{总}}(n_0, n_1)}{\partial \text{net}_j(n)}$$

式中， $\text{net}_j(n)$ 是神经元 j 的净输入。

在标准 BP 算法的基础上，逐时段 BPTT 算法按下列方法进行：

① 对时段 (n_0, n_1) 内各时刻的数据，按图 6.18(b) 中的信号流图逐层进行前向计算，保存输入数据、网络权值、网络输出期望输出。

② 应用上述前向计算结果，执行一个逐层反向传播过程，计算公式为：

$$\delta_j(n) = \begin{cases} f'[v_j(n)]e_j(n), & n=n_1 \\ f'[\text{net}_j(n)]\{e_j(n) + \sum_{k \in A} w_{kj}\delta_k(n+1)\}, & n_0 < n < n_1 \end{cases} \quad (6.51)$$

式中, $f'(\cdot)$ 是转移函数对其自变量的导数。上述计算从时刻 n_1 开始反向进行直到时刻 n_0 , 所计算的步数即等于时段 (n_0, n_1) 内的时间序列长度。

③ 当反向传播计算回到 n_0+1 时, 对神经元 j 的权值调整如下:

$$\Delta w_{ij} = -\eta \frac{\partial E_{\text{总}}(n_0, n_1)}{\partial w_{ij}} = \eta \sum_{n=n_0+1}^{n_1} \delta_j(n)x_i(n-1) \quad (6.52)$$

式中, η 是学习率; $x_i(n-1)$ 是在时刻 $n-1$ 时作用于神经元 j 的第 i 个输入。

将逐时段 BPTT 算法与标准 BP 算法比较可以看出, 两者的根本差别是: ① 网络的每一个计算层都给定了期望输出值, 这是因为式(6.52)要求计算整个 (n_0, n_1) 时段内每个时刻的 $\delta_j(n)$, $n=n_0, n_0+1, \dots, n_1$, 而各时刻 $\delta_j(n)$ 的计算涉及期望输出与实际输出之间的误差 $e_j(n)$, $n=n_0, n_0+1, \dots, n_1$, 实际输出层在网络的时序展开时被重复了很多次; ② BPTT 算法需要对每个样本模式在时段 (n_0, n_1) 内的所有时刻计算权值的瞬时修正量, 然后再对该时间段内的全部瞬时权值修正量求和。这种计算方式显然不适于在线实时计算。

(2) 截断的 BPTT 算法 如果只在一个固定的时间序列内存存储相关的输入数据和网络状态的历史记录, 相应的 BPTT 算法称为截断的 BPTT 算法 [truncated back-propagation through time, BPTT(h)], 该时间序列的长度 h 称为截断深度, 任何 h 时刻之前的信息均不需要存储。

为了得到 BPTT 的实时形式, 将需要最小化的目标函数定义为瞬时误差的平方和, 即:

$$E(n) = \frac{1}{2} \sum_{j \in A} e_j^2(n) \quad (6.53)$$

BPTT(h) 算法对神经元 j 的局部梯度定义为:

$$\delta_j(l) = -\frac{\partial E(l)}{\partial \text{net}_j(l)} \quad \text{对于 } j \in A \text{ 且 } n-h < l \leq n$$

BPTT(h) 算法按下列方法进行:

① 对区间 $(n-h, n)$ 内各时刻的数据, 按图 6.18(b) 中的信号流图逐层进行前向计算, 保存输入数据和网络权值。

② 应用上述前向计算结果, 执行一个逐层反向传播过程, 计算公式为:

$$\delta_j(l) = \begin{cases} f'[\text{net}_j(l)]e_j(l), & l=n \\ f'[\text{net}_j(l)] \sum_{k \in A} w_{kj}(l)\delta_k(l+1), & n-h < l < n \end{cases} \quad (6.54)$$

上述计算反向进行到时刻 $n-h+1$ 时, 所计算的步数等于 h 。

③ 当反向传播计算到 $n-h+1$ 时, 对神经元 j 的权值进行如下调整:

$$\Delta w_{ij}(n) = \eta \sum_{l=n-h+1}^n \delta_j(l)x_i(l-1) \quad (6.55)$$

式中, η 和 $x_i(l-1)$ 定义如前。由于式(6.54)中使用了 $w_{kj}(l)$, 训练时要求保留权值的历史记录。

比较式(6.51)和式(6.54)可以看出, BPTT(h) 算法中只需要计算当前时间的误差信号 $e_j(n)$, 因此不需要保存过去的期望输出值。

6.6.2.2 实时递归学习算法

实时递归学习算法 (real time recurrent learning, RTRL) 适用的递归网络结构如图 6.19 所示。该网络的第一层为输入-反馈并置的联合输入层，部分节点负责接受计算层传来的反馈信号，部分节点负责接受外界的输入信号；第二层为计算处理层，该层神经元的输出全部通过延时单元反馈到联合输入层的反馈信号接受节点，同时其中一部分作为网络的输出，因此计算处理层既包含隐节点又包含输出节点。

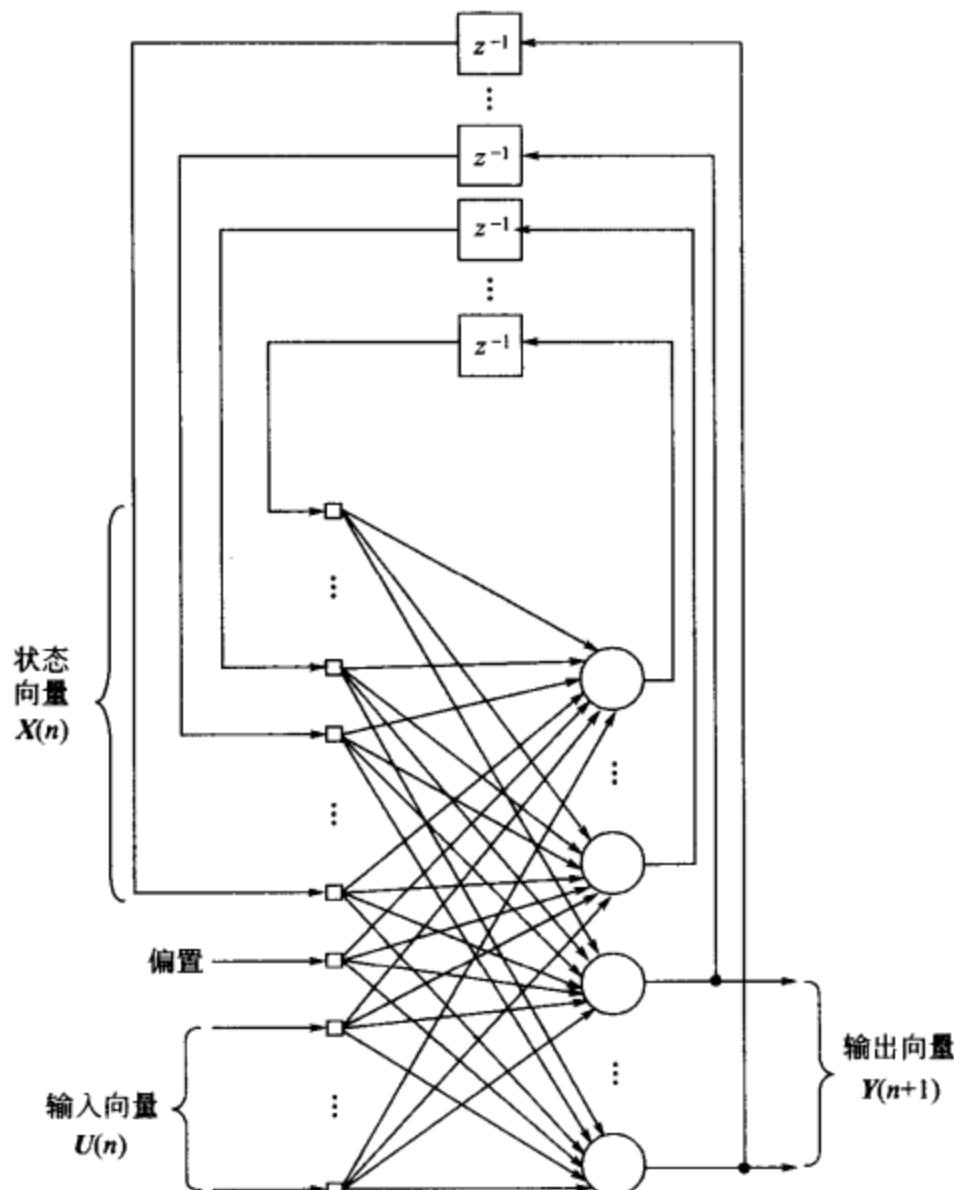


图 6.19 RTRL 网络结构

用 $U(n)$ 表示 n 时刻的 m 维外部输入向量， $X(n)$ 表示由计算处理层反馈到联合输入层的 q 维向量， $Z(n)$ 表示联合输入层的 $q+m$ 维总输入向量， $Y(n+1)$ 表示下一时刻网络的输出向量。网络的连接权也由前馈和反馈两部分构成，共有 $(q+m) \times q$ 个前向连接以及 q^2 个反馈连接，其中 q 个为自反馈连接。

n 时刻计算层神经元 j 的净输入为：

$$\text{net}_j(n) = \sum_{i=1}^{q+m} w_{ij} z_i(n) \quad j=1, 2, \dots, q \quad (6.56)$$

式中， $w_{i1}, w_{i2}, \dots, w_{iq}$ 为输入层中反馈部分与输出层的连接权值； $w_{i,q+1}, w_{i,q+2}, \dots, w_{i,q+m}$ 为输入层中外部输入部分与输出层的连接权值。

下一时刻神经元 j 的输出为：

$$y_j(n+1) = f[\text{net}_j(n)] \quad j \in C(n) \quad (6.57)$$

式中, $C(n)$ 表示输出节点的集合, 在不同的时刻 $C(n)$ 可以不同。

下面利用梯度下降法推导 RTRL 算法。首先定义 q 维误差向量 $E(n)$ 中的第 j 个分量为:

$$e_j(n) = \begin{cases} d_j(n) - y_j(n), & j \in C(n) \\ 0, & j \notin C(n) \end{cases}$$

定义 n 时刻的瞬时误差平方和为:

$$E(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n)$$

所有时间的总误差为:

$$E_{\text{总}} = \sum_n E(n)$$

某一权值 $w_{lk}(n)$ 在 n 时刻的修正量为:

$$\Delta w_{lk}(n) = -\eta \frac{\partial E(n)}{\partial w_{lk}(n)}$$

因为:

$$\frac{\partial E(n)}{\partial w_{lk}(n)} = \sum_{j \in C} e_j(n) \frac{\partial e_j(n)}{\partial w_{lk}(n)} = -\sum_{j \in C} e_j(n) \frac{\partial y_j(n)}{\partial w_{lk}(n)}$$

根据式(6.56) 与式(6.57), 并利用微分的链式规则, 得到:

$$\frac{\partial y_j(n+1)}{\partial w_{lk}(n)} = \frac{\partial y_j(n+1)}{\partial \text{net}_j(n)} \frac{\partial \text{net}_j(n)}{\partial w_{lk}(n)} = f'[\text{net}_j(n)] \frac{\partial \text{net}_j(n)}{\partial w_{lk}(n)} \quad (6.58)$$

利用式(6.56) 对 $w_{ij}(n)$ 求导, 得到:

$$\frac{\partial \text{net}_j(n)}{\partial w_{lk}(n)} = \sum_{i=1}^{q+m} \frac{\partial [w_{ij} z_i(n)]}{\partial w_{lk}(n)} = \sum_{i=1}^{q+m} \left[w_{ij}(n) \frac{\partial z_i(n)}{\partial w_{lk}(n)} + \frac{\partial w_{ij}(n)}{\partial w_{lk}(n)} z_i(n) \right]$$

只有当 $j=k$, $i=l$ 时, $\partial w_{ij}(n)/\partial w_{lk}(n)$ 为 1, 否则为零。因此上式可写为:

$$\frac{\partial \text{net}_j(n)}{\partial w_{lk}(n)} = \sum_{i=1}^{q+m} w_{ij}(n) \frac{\partial z_i(n)}{\partial w_{lk}(n)} + \delta_{jk} z_j(n) \quad (6.59)$$

其中

$$\delta_{kj} = \begin{cases} 1, & j=k \\ 0, & j \neq k \end{cases}$$

$$\frac{\partial z_i(n)}{\partial w_{lk}(n)} = \begin{cases} 0, & z_i(n) = u_i(n) \\ \frac{\partial y_j(n)}{\partial w_{lk}(n)}, & z_i(n) = x_i(n) \end{cases} \quad (6.60)$$

由式(6.58)~式(6.60) 可得到:

$$\frac{\partial y_j(n+1)}{\partial w_{lk}(n)} = f'[\text{net}_j(n)] \left[\sum_{i=1}^q w_{ij}(n) \frac{\partial y_j(n)}{\partial w_{lk}(n)} + \delta_{jk} z_l(n) \right] \quad (6.61)$$

设 $n=0$ 时, $\partial y_j(0)/\partial w_{lk}(0)=0$, $j, k=1, 2, \dots, q$, $l=1, 2, \dots, q+m$ 。

根据上述结果, RTRL 算法的训练步骤如下:

(1) 将权值初始化为均匀分布的随机数, 从 $n=0$ 起, 对每一时刻 n , 用式(6.56) 和式(6.57) 进行前向计算, 得出联合输入向量 $Z(n) = [z_1, z_2, \dots, z_{q+m}]^T = [x_1, x_2, \dots, x_q, u_1, u_2, \dots, u_m]^T$ 。

(2) 对 $j, k=1, 2, \dots, q$, $l=1, 2, \dots, q+m$, 按初始条件 $\partial y_j(0)/\partial w_{lk}(0)=0$ 和式

(6.61) 计算 $\partial y_j(n+1)/\partial w_{lk}(n)$ 。

(3) 将 $\partial y_j(n+1)/\partial w_{lk}(n)$ 和误差信号 $e_j(n)=d_j(n)-y_j(n)$ 代入权值修正公式:

$$\Delta w_{ij}(n+1)=\eta \sum_{j \in C} e_j(n) \frac{\partial y_j(n)}{\partial w_{lk}(n)} \quad (6.62)$$

(4) 计算权值 $w_{lk}(n+1)=w_{lk}(n)+\Delta w_{lk}(n)$, 重复上述步骤直到误差小于预先设定的允许值。

6.6.3 递归网络应用举例

系统辨识就是在输入和输出数据的基础上, 从一组给定的模型类中确定一个与所测系统等价的模型。系统辨识在工业生产中有着广泛的应用, 由辨识所得的被控对象模型可用于控制系统的设计, 或分析改进已有的控制系统。在模型结构确定的情况下, 通过辨识建立其时变模型, 可实现系统参数的预测和预测。系统辨识还可由系统的运行状态信息推测系统动态特性的变化, 监测系统的运行状态, 进行故障诊断。长期以来, 对被辨识对象模型结构的选择是建立在线性系统的理论基础之上的, 对于复杂的非线性、不确定及不知系统的辨识, 一直未能找到有效的解决方法。而基于多层感知器递归神经网络的非线性函数逼近能力和学习能力, 为非线性系统的辨识提供了一种有效的新途径。

采用系统的一般描述, 设被辨识的非线性系统差分方程为:

$$y(t+1)=f[a_0 y(t), a_1 y(t-1), a_2 y(t-2), u(t), u(t-1)]$$

采用 NARX 网络模型时, 网络输入层有 5 个神经元, 分别接受系统的 3 个来自输出反馈的信号序列和 2 个外部输入信号序列, 输出层有 1 个神经元, 其输出对应于 $\hat{y}(t+1)$, 网络的隐层一般设 1 层, 神经元个数由经验和实验确定。系统辨识可用图 6.20(a) 中的并联结构实现, 也可用图 6.20(b) 中的串-并联结构实现。

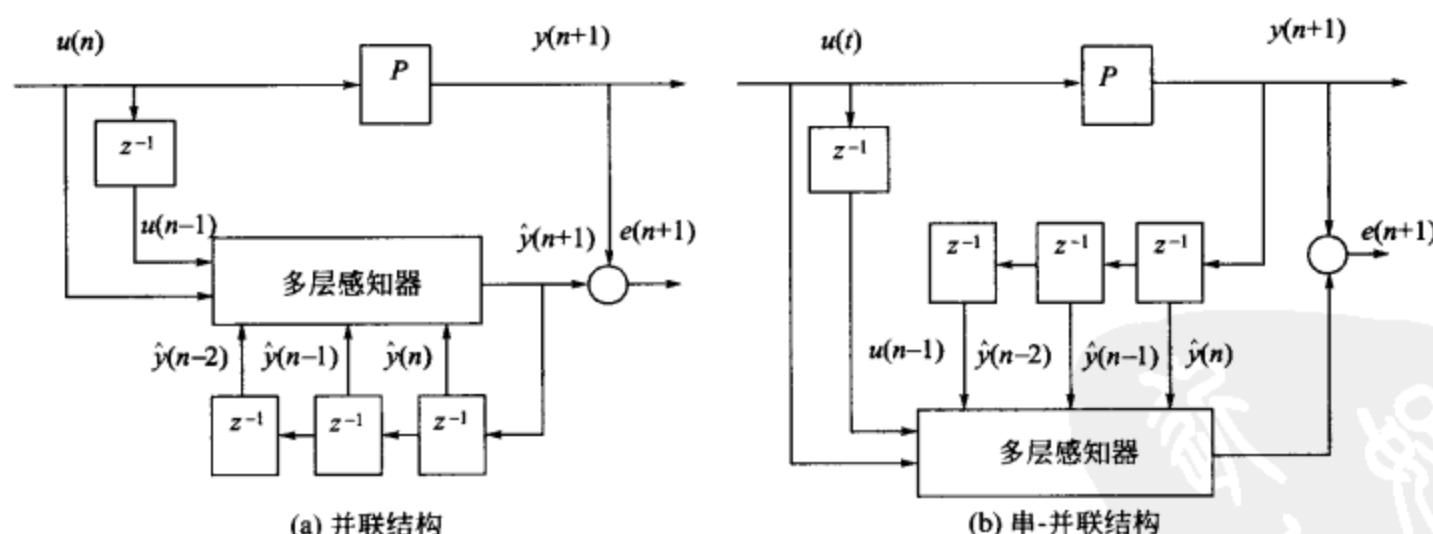


图 6.20 系统模型辨识的两种结构

(1) 系统模型辨识的并联结构 由于在辨识过程的初始阶段, 神经网络的实际输出 $\hat{y}(t+1)$ 很难接近被辨识系统的实际输出 $y(t+1)$, 而且可能不稳定, 因此由各延时单元 z^{-1} 输出的 $\hat{y}(t), \hat{y}(t-1), \hat{y}(t-2)$ 在网络训练开始时均不可靠, 在这种情况下, 不一定能保证系统辨识收敛。

(2) 系统模型辨识的串-并联结构 在辨识过程中, 神经网络始终以系统的实际输出

$y(t), y(t-1), y(t-2)$ 作为训练样本，一般情况下系统辨识能够收敛。

本章小结

本章介绍了四类反馈神经网络：离散型 Hopfield 网络和连续型 Hopfield 网络、离散型双向联想记忆神经网络、随机神经网络和递归网络。前两类网络学习方式的共同特点是，网络的权值不是经过反复学习获得，而是按一定规则进行设计，网络权值一经确定就不再改变。本章需重点理解的问题是：

(1) 网络的稳定性 反馈网络实质上是一个非线性动力学系统。网络从初态 $X(0)$ 开始，若能经有限次递归后，其状态不再发生变化，则称该网络是稳定的。如果网络是稳定的，它可以从任一初态收敛到一个稳态；若网络是不稳定的，网络可能出现限幅的自持振荡现象。利用网络的稳态可实现联想记忆和优化计算。

(2) 网络的能量函数 反馈网络用“能量函数”描述其状态，在反馈网络结构满足一定条件的前提下，若按一定规则不断更新网络的状态，则具有特定形式的能量函数将单调减小，最后达到能量的某一极小点，网络所有神经元的状态将不再改变，那便是反馈网络的稳定状态。

(3) 网络的记忆容量 当网络规模一定时，所能记忆的模式是有限的。对于所容许的联想出错率，网络所能存储的最大模式数 P_{\max} 称为网络容量。网络容量与网络的规模、算法以及记忆模式向量的分布都有关系。

(4) 网络的权值设计 对于前两类反馈网络，没有权值调整的训练过程，因此没有“学习”意义上的调整，只有一次性的“记住”。要求网络记住的权值可按某种设计规则进行计算，如 DHNN 网和 BAM 网的权值设计均采用外积和规则进行计算。CHNN 网用于解决优化计算，其权值设计是在根据实际问题构造网络的能量函数时解决的。

(5) 随机网络的运行原理 BM 网络是一种典型的随机神经网络，采用神经元状态按概率随机取值的工作方式。随着网络状态的演变，从概率的意义上网络能量的总趋势总是朝着减小的方向变化，但不排除在有些步神经元状态可能会按小概率取值，从而使网络具有了从能量局部极小点逃出的能力，这一点是 BM 网络与 DHNN 网能量变化的根本区别。模拟退火算法的温度参数 T 不断下降可使网络能量的“爬山”能力由强减弱，这是保证 BM 网络能成功搜索到能量全局最小的有效措施。

(6) 针对动态系统的递归神经网络是在多层感知器基础上增加延时和反馈单元而形成的。递归网络的结构布局有多种形式，但共同特点是结合一个静态多层感知器并利用多层感知器的非线性映射能力。递归网络的反馈可以是局部的或全局的，全局反馈可以有不同的形式。

思考与练习

- 6.1 如何利用 DHNN 网的吸引子进行联想记忆？
- 6.2 如何利用 CHNN 网的稳态进行优化计算？
- 6.3 如何利用 BAM 网实现双向联想？
- 6.4 为什么 BM 网络可避免陷入能量局部极小？
- 6.5 DHNN 网权值矩阵 W 给定为：

$$\mathbf{W} = \begin{bmatrix} 0 & 1 & -1 & -1 & -3 \\ 1 & 0 & 1 & 1 & -1 \\ -1 & 1 & 0 & 3 & 1 \\ -1 & 0 & 3 & 0 & 1 \\ -3 & -1 & 1 & 1 & 0 \end{bmatrix}$$

已知各神经元阈值为 0, 试计算网络状态为 $\mathbf{X}=[-1, 1, 1, 1, 1]^T$ 和 $\mathbf{X}=[-1, -1, 1, -1, -1]^T$ 时的能量值。

6.6 DHNN 网络如图 6.21 所示, 部分权值已标在图中。试求:

- ① 该网络的权值矩阵 \mathbf{W} ;
- ② 从初始状态开始按 1, 2, … 顺序进行异步更新, 给定初始状态为:

$$\mathbf{X}^1(0) = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{X}^2(0) = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}, \quad \mathbf{X}^3(0) = \begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \\ -1 \end{bmatrix}, \quad \mathbf{X}^4(0) = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}, \quad \mathbf{X}^5(0) = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

③ 以上哪个状态是网络的吸引子?

④ 计算对应于吸引子的能量值。

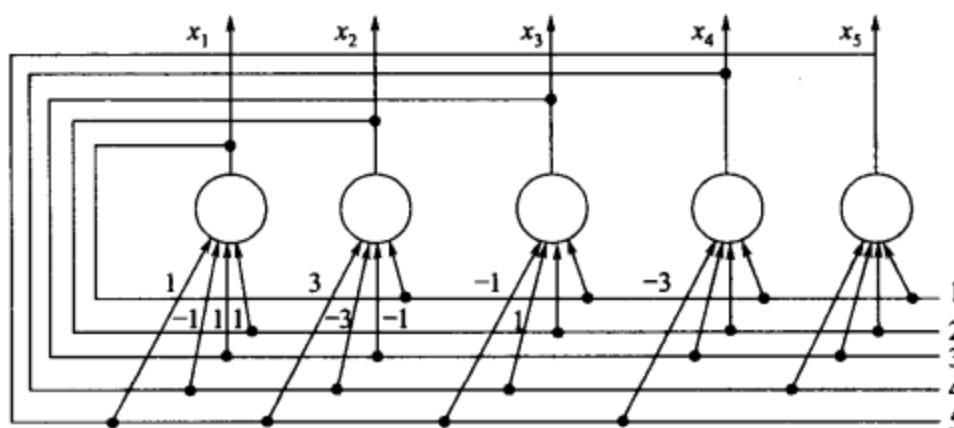


图 6.21 习题 6.6 附图

6.7 表 6.1 给出城市的距离, 试学习 6.3.2 中的方法用 Hopfield 网络求解 TSP 问题。

表 6.1 城市距离 (单位: km)

城市	天津	石家庄	太原	呼和浩特	上海
北京	119	263	398	401	1078
天津	0	262	426	504	963
石家庄	262	0	171	394	989
太原	426	171	0	341	1096
呼和浩特	504	394	341	0	1381

7

小脑模型神经网络

1975 年, J. S. Albus 提出一种模拟小脑功能的神经网络模型, 称为 Cerebellar Model Articulation Controller, 简称 CMAC。CMAC 网络是仿照小脑控制肢体运动的原理而建立的神经网络模型。小脑指挥运动时具有不假思索地作出条件反射式迅速响应的特点, 这种条件反射式响应是一种迅速联想。CMAC 网络有三个特点: 其一是, 作为一种具有联想功能的神经网络, 它的联想具有局部推广(或称泛化)能力, 因此相似的输入将产生相似的输出, 远离的输入将产生独立的输出; 其二是, 对于网络的每一输出, 只有很少的神经元所对应的权值对其有影响, 哪些神经元对输出有影响则由输入决定; 其三是, CMAC 的每个神经元的输入输出是一种线性关系, 但其总体上可看作一种表达非线性映射的表格系统。由于 CMAC 网络的学习只在线性映射部分, 因此可采用简单的 δ 算法, 其收敛速度比 BP 算法快得多, 且不存在局部极小问题。CMAC 最初主要用来求解机械手的关节运动, 其后进一步用于机器人控制、模式识别、信号处理以及自适应控制等领域。

7.1 CMAC 网络的结构

简单的 CMAC 结构如图 7.1 所示, 图中 X 表示 n 维输入状态空间, A 为具有 m 个单元的存储区(亦称为相联空间或概念记忆空间)。设 CMAC 网络的输入向量用 n 维输入状态空间 X 中的点 $X^p = (x_1^p, x_2^p, \dots, x_n^p)^T$ 表示, 对应的输出向量用 $Y^p = F(x_1^p, x_2^p, \dots, x_n^p)$ 表示, 图中 $p=1, 2, 3$ 。输入空间的一个点 X^p 将同时激活 A 中的 C 个元素(图 7.1 中 $C=4$), 使其同时为 1, 而其他大多数元素为 0, 网络的输出 Y^p 即为 A 中 4 个被激活单元对应的权值累加和。 C 值与泛化能力有关, 称为泛化参数。也可以将其看作信号检测单元的感受野大小。

一般来说, 实际应用时输入向量的各分量来自不同的传感器, 其值多为模拟量, 而 A 中每个元素只取 0 或 1 两种值。为使 X 空间的点映射为 A 空间的离散点, 必须先将模拟量 X^p 量化, 使其成为输入状态空间的离散点。设输入向量 X 的每一分量可量化为 q 个等级, 则 n 个分量可组合为输入状态空间 q^n 种可能的状态 X^p , $p=1, 2, \dots, q^n$ 。其中每一个状态 X^p 都要映射为 A 空间存储区的一个集合 A^p , A^p 的 C 个元素均为 1。从图 7.1 可以看出, 在 X 空间接近的样本 X^2 和 X^3 在 A 中的映射 A^2 和 A^3 出现了交集 $A^2 \cap A^3$, 即它们对应的 4 个权值中有两个是相同的, 因此由权值累加和计算的两个输出也较接近。从函数映射的角度看, 这一特点可起到泛化的作用。显然, 对相距很远的样本 X^1 和 X^3 , 映射到 A 中的 $A^1 \cap A^3$ 为空集, 这种泛化不起作用, 因此是一种局部泛化。输入样本在输入空间距离越近, 映射到 A 存储区后对应交集中的元素数就越接近 C , 其对应的输出也越接近。从分类角度看, 不同输入样本在 A 中产生的交集起到了将相近样本聚类的作用。

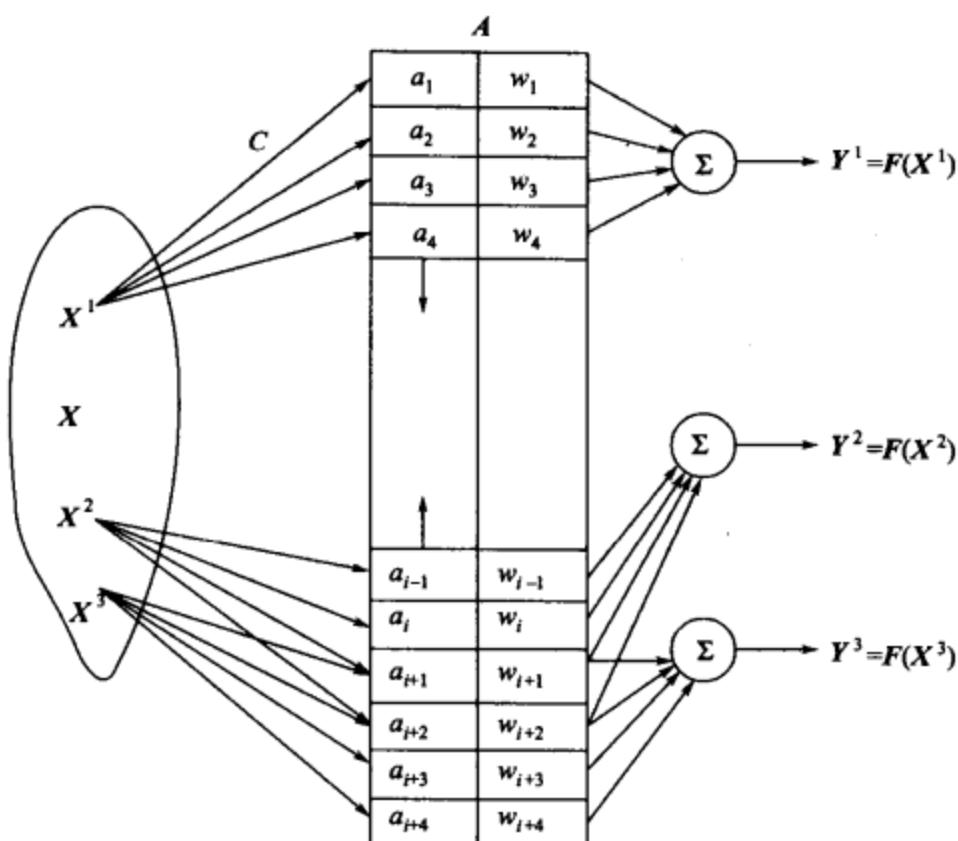


图 7.1 简单 CMAC 模型结构

为使对于 X 空间的每一个状态在 A 空间均存在唯一的映射，应使 A 存储区中单元的个数至少等于 X 空间的状态个数，即：

$$m \geq q^n$$

设将三维输入的每个分量量化为 10 个等级，则 $m \geq 1000$ 。对于许多实际系统， q^n 往往要比这个数字大得多，但由于大多数学习问题不会包含所有可能的输入值，实际上不需要 q^n 个存储单元来存放学习的权值。 A 相当于一种虚拟的内存地址，每个虚拟地址与输入状态空间的一个样本点相对应。通过哈希编码（Hash-coding）可将具有 q^n 个存储单元的地址空间 A 映射到一个小得多的物理地址空间 A_p 中。

对于每个输入， A 中只有 C 个单元为 1，而其余 q^n 个均为 0，因此 A 是一个稀疏矩阵。哈希编码是压缩稀疏矩阵的常用技术，具体方法是通过一个产生随机数的程序来实现的。以 A 的地址作为随机数产生程序的变量，产生的随机数作为 A_p 的地址。由于产生的随机数限制在一个较小的整数范围内，因此 A_p 远比 A 小得多。显然，从 A 到 A_p 的压缩是一种多对少的随机映射。在 A_p 中，对每一输入样本有 C 个随机地址与之对应， C 个地址存放的权值须通过学习得到，其累加和即作为 CMAC 的输出。

7.2 CMAC 网络的工作原理

为详细分析 CMAC 网络的工作原理，以二维输入/一维输出模型为例进行讨论，并将图 7.1 中的 CMAC 模型细化为图 7.2 所示。网络的工作过程可分解为四步映射。

7.2.1 从 X 到 M 的映射

二维 X 空间的两个分量为模拟信号 θ_1 和 θ_2 ，来自两个传感器，例如 θ_1 和 θ_2 可以代表机器人的两个关节的角度。 M 为输入量化器，分为 M_{θ_1} 和 M_{θ_2} 两组，分别对应着两个输入信号。图中 M 的每一个小格代表一个感知器，感知器的个数就是对输入信号的量化级数。 M_{θ_1}

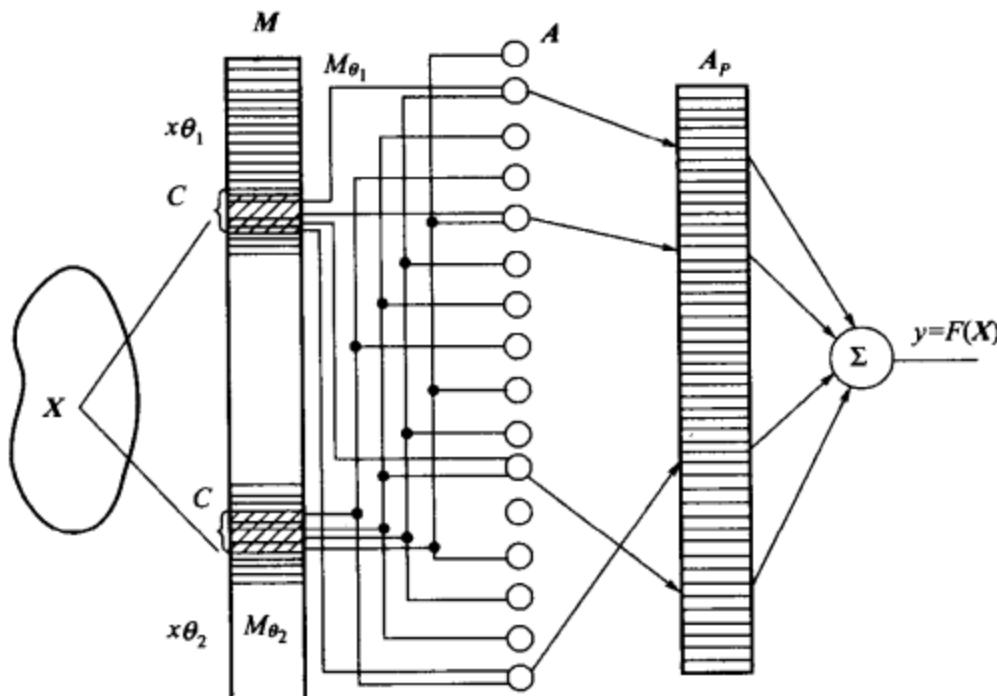


图 7.2 二维输入/一维输出 CMAC 模型

和 M_{θ_2} 的量化级数不一定相同，它们分别表示对输入信号的分辨率。 x_{θ_1} 和 x_{θ_2} 分别为表示输入信号的量化值，对任意输入信号 θ_1 和 θ_2 ，在 M_{θ_1} 和 M_{θ_2} 中必然各有一个与其量化值对应的感知器被激活。但为了泛化的需要，在与输入量化值对应的感知器周围可有 C 个感知器同时激活。 C 代表了泛化范围，其值是设计时由设计者选定的，一般可以选得很大，如 10~100。当 C 选定后， M_{θ} 中的感知器个数应在量化级数基础上增加 $C-1$ 个。设某个输入分量的量化级数为 9，每个量化值同时激活的感知器数量为 $C=4$ ，则对于各量化值的激活情况如表 7.1 所示。

表 7.1 感知器激活情况

x_{θ}	μ_a	μ_b	μ_c	μ_d	μ_e	μ_f	μ_g	μ_h	μ_i	μ_j	μ_k	μ_l
1	1	1	1	1	0	0	0	0	0	0	0	0
2	0	1	1	1	1	0	0	0	0	0	0	0
3	0	0	1	1	1	1	0	0	0	0	0	0
4	0	0	0	1	1	1	1	0	0	0	0	0
5	0	0	0	0	1	1	1	1	0	0	0	0
6	0	0	0	0	0	1	1	1	1	0	0	0
7	0	0	0	0	0	0	1	1	1	1	0	0
8	0	0	0	0	0	0	0	1	1	1	1	0
9	0	0	0	0	0	0	0	0	1	1	1	1

表中列出 M_{θ} 中的 12 个感知器，分别用 $\mu_a, \mu_b, \dots, \mu_l$ 表示。从各行情况可以看出，对于输入信号 θ 的任意一个量化值 x_{θ} ，总有 4 个感知器被激活为 1；从各列情况可以看出，对于每个感知器，其对应输入信号量化值的范围最宽可达到 $C=4$ 。如感知器 $\mu_d \dots \mu_l$ 对应的 x_{θ} 取值范围均为 4。

为了分析在 X 空间靠近的样本在 M 中是否也靠近，下面考虑 X 为一维的情况。将被激活为 1 的感知器用其下标字母表示，将被某一输入信号量化值同时激活为 1 的感知器集合用 m^* 表示，其中包含的兴奋元素的个数用 $|m^*|$ 表示，表 7.1 可转化为表 7.2。可以看出，在 X 空间接近的样本，在对应的感知器集合也接近（重叠元素多）。如用 H_{ij} 表示输入空间中两个样本向量量化值的差，则有：

$$H_{ij} = |x_i - x_j| = |m_i^*| - |m_i^* \cap m_j^*|$$

例如, 对于一维输入空间两个接近的量 $x_\theta=5$ 和 $x_\theta=4$, 其接近程度为 $H_{ij} = |x_i - x_j| = 1$, 则在输出 m^* 中有 $|m_i^*|=4$, $m_i^* \cap m_j^* = \{e, f, g\}$, $|m_i^* \cap m_j^*|=3$, 其接近程度也为 $H_{ij} = |m_i^*| - |m_i^* \cap m_j^*| = 4 - 3 = 1$ 。可见, 在输入空间接近的量输出时也接近。

表 7.2 与输入信号量化值对应的感知器

x_θ	m^*				x_θ	m^*			
1	a	b	c	d	6	i	f	g	h
2	e	b	c	d	7	i	j	g	h
3	e	f	c	d	8	i	j	k	h
4	e	f	g	d	9	i	j	k	l
5	e	f	g	h					

一般情况下, 输入是多维的, 需要用组合滚动的方式对感知器编号。以二维输入为例, 设 $x_{\theta 1}$ 量化为 5 级, $x_{\theta 2}$ 量化为 7 级, 对应的激活感知器编号分别用大写和小写字母表示, 结果如表 7.3 和表 7.4 所示。

n 维情况下, $\mathbf{X}=(x_{i1}, x_{i2}, \dots, x_{in})^\top$, 则从 \mathbf{X} 到 \mathbf{M} 的映射为:

$$\mathbf{X} \rightarrow \mathbf{M} = \begin{cases} x_{i1} \rightarrow m_{i1}^* \\ x_{i2} \rightarrow m_{i1}^* \\ \vdots \\ x_{in} \rightarrow m_{in}^* \end{cases}$$

表 7.3 与 $x_{\theta 1}$ 对应的感知器

$x_{\theta 1}$	$m_{\theta 1}$			
1	A	B	C	D
2	E	B	C	D
3	E	F	C	D
4	E	F	G	D
5	E	F	G	H

表 7.4 与 $x_{\theta 2}$ 对应的感知器

$x_{\theta 2}$	$m_{\theta 2}$			
1	a	b	c	d
2	e	b	c	d
3	e	f	c	d
4	e	f	g	d
5	e	f	g	h
6	i	f	g	h
7	i	j	g	h

7.2.2 从 \mathbf{M} 到 \mathbf{A} 的映射

从 \mathbf{M} 到 \mathbf{A} 的映射是通过滚动组合得到, 其原则仍然是在输入空间相近的向量在输出空间也接近。如果感知器的泛化范围为 C , 则在 \mathbf{A} 中映射的地址也应为 C 个, 而与输入维数无关。仍以二维输入情况为例, 从 \mathbf{X} 到 \mathbf{M} 的映射如表 7.3 和表 7.4 所示。将两表中的感知器用“与”的关系进行组合, 得到 \mathbf{A} 的地址如表 7.5 所示。

表 7.5 由 $m_{\theta 1}$ 和 $m_{\theta 2}$ 组合的 \mathbf{A} 地址

$x_{\theta 2}$	\mathbf{A}^*				
7ijgh	AiBjCgDh	EiBjCgDh	EiFjGgDh	EiFjGgHh	EiFjGgHh
6ifgh	AiBfCfgh	EiBfCgDh	EiFfCgDh	EiFfGgDh	EiFfGgHh
5efgh	AeBfCgDh	EeBfCgDh	EeFfCgDh	EeFfGgDh	EeFfGgHh
4efgd	AeBfCgDd	EeBfCgDd	EeFfCgDd	EeFfGgDd	EeFfGgHd
3efcd	AeBfCcDd	EeBfCcDd	EeFfCcDd	EeFfGcDd	EeFfGcHd
2ebcd	AeBbCcDd	EeBbCcDd	EeFbCcDd	EeFbGcDd	EeFbGcHd
1abcd	AaBbCcDd	EaBbCcDd	EaFbCcDd	EaFbGcDd	EaFbGcHd
$x_{\theta 1}$	1 ABCD	2 EBCD	3 EFCD	4 EFGD	5 EFGH

从表 7.5 可以看出，每个 A^* 都是由 $x_{\theta 1}$ 和 $x_{\theta 2}$ 对应的 $m_{\theta 1}$ 和 $m_{\theta 2}$ 组合而成的。 A^* 中含有 C 个单元，即 A 中有 C 个存储单元被激活。以 $X=(1, 7)^T$ 为例， $x_{\theta 1}=1$ 对应的激活感知器为 $m_{\theta 1}=ABCD$ ，而 $x_{\theta 2}=7$ 对应的激活感知器为 $m_{\theta 2}=ijgh$ ，组合后的单元用 $A^*=(Ai\ Bj\ Cg\ Dh)$ 表示， A^* 是由大写字母和小写字母为标记的 C 个存储单元的集合。A 中有足够的存储单元组合 A^* ，可代表 X 所有可能的值。在输入空间中比较相近的向量经过从 X 到 M，再从 M 到 A 的映射，得到的 A^* 集合也较相近。A 中集合间的接近程度可从其交集的大小，即交集所含的元素数得到反映，因此将 A_i^* 和 A_j^* 的交集称为的 A_i^* 邻域。A 中集合间的分离程度可用其距离反映。A 中两个集合之间的距离可表示为：

$$d_{ij} = |A^*| - |A_i^* \cap A_j^*|$$

由表 7.5 可以看出，对于同一列中的任意相邻行或同一行中的任意相邻列， X_i 和 X_j 的距离 H_{ij} 均为 1，对应的 A_i^* 和 A_j^* 的距离 d_{ij} 也等于 1。而隔行且隔列的 A_i^* 和 A_j^* 对应的输入样本相距较远，其距离 d_{ij} 也相应较大，读者不妨从表 7.5 中进一步分析这种规律。

任何两个输入样本 X_i 和 X_j 映射到 A 中的 A_i^* 和 A_j^* 上，两集合交集的大小 $|A_i^* \cap A_j^*|$ 与输入样本 X_i 和 X_j 的邻近程度成正比，而与输入向量的维数无关。 A^* 邻域的大小除了相交集合对应的输入样本的邻近程度有关外，还和 C 的选择以及输入向量的分辨率有关。

7.2.3 从 A 到 A_p 的映射

表 7.5 中，大写字母 A, B, C, D, …, H 和小写字母 a, b, c, d, …, j 分别表示 A 存储器中前 P_f 个地址的编号和后 P_r 个地址的编号，而 Ai 、 Bj 、 Cg 、 Dh 等表示虚拟的存储地址，在存储器 A 中的 C 个虚拟地址组合成 A^* ，它代表了 X 空间中的输入向量。设 X 空间为 n 维，每一维有 g 个量化级，则 A 中至少有 g^n 个相应的 A^* ，它对应于 X 空间的每一个样本点。 A^* 占据的存储空间很大，但是对于特定的问题，系统并不会历经整个输入空间，这样在 A 中被激励的单元是稀疏的，采用杂散技术，可以将 A 压缩到一个比较小的实际空间 A_p 中去。

杂散技术是将分布稀疏、占用较大存储空间的数据作为一个伪随机发生器的变量，产生一个占用空间小的随机地址，用于存放 A 中的数据。实现这一压缩的最简单方法是用 A 中的 A^* 地址除以一个大的质数，所得余数就作为一个伪随机码，表示为 A_p 中的地址。例如 Dg 可用两位 BCD 码表示，第一位表示 D 的编号，第二位表示 g 的编号。一位 BCD 码需 4 比特，在 A 中需 $2^8=256$ 个地址，若 A_p 中有 16 个地址，可取质数 17 去除 A 的地址，余数为 A_p 的地址，照此可得到从 2^8 个地址到 16 个地址中的映射。读者容易想到，A 中不同的地址在 A_p 中会映射到同一地址。事实上用杂散技术确实会不可避免地带来地址冲撞的问题，但如果映射的随机性很强，将大大减少冲撞的概率。在 CMAC 中忽略这种冲撞，是因为冲撞不强烈时，可将其看作一种随机扰动，通过学习算法的迭代过程，可逐步将影响减小，而不影响输出结果。

7.2.4 从 A_p 到 F 的映射

经过以上映射，在 A_p 中有 $|A^*|$ 个随机分布的地址，每个地址中都存放了一个权值，CMAC 网络的输出就是这些权值的叠加，即：

$$Y = F(X) = \sum_{i \in A^*} w_i$$

对于某一输入样本 X , 通过下面将要介绍的学习算法调整权值, 可使 CMAC 产生期望的输出。

7.3 CMAC 网络的学习算法

7.3.1 CMAC 网络的学习算法

CMAC 网络采用 δ 学习算法调整权值, 图 7.3 给出其示意图。用 F_0 表示对应于输入 X 的期望的输出向量, $F_0 = (F_{01}, F_{02}, \dots, F_{0r})$, 权值调整公式为:

$$\delta_j = F_{0j} - F(X) \quad (7.1)$$

$$w_{ij}(t+1) = w_{ij}(t) + \eta \frac{\delta_j}{|A^*|} \quad i=1, 2, \dots, n; j=1, 2, \dots, r \quad (7.2)$$

网络的 r 个输出为:

$$y_j = F_j(X) = \sum_{i \in A^*} w_{ij} \quad j = 1, 2, \dots, r \quad (7.3)$$

CMAC 的权值调整有两种情况: 一种为批学习方式, 即将训练样本输入一轮后用累积的 δ 值代入式(7.2) 调整权值; 另一种为轮训方式, 即每个样本输入后都调整权值。前一种方式可采用线性代数方程的雅可比迭代法, 后一种则可采用高斯-赛德尔迭代法。

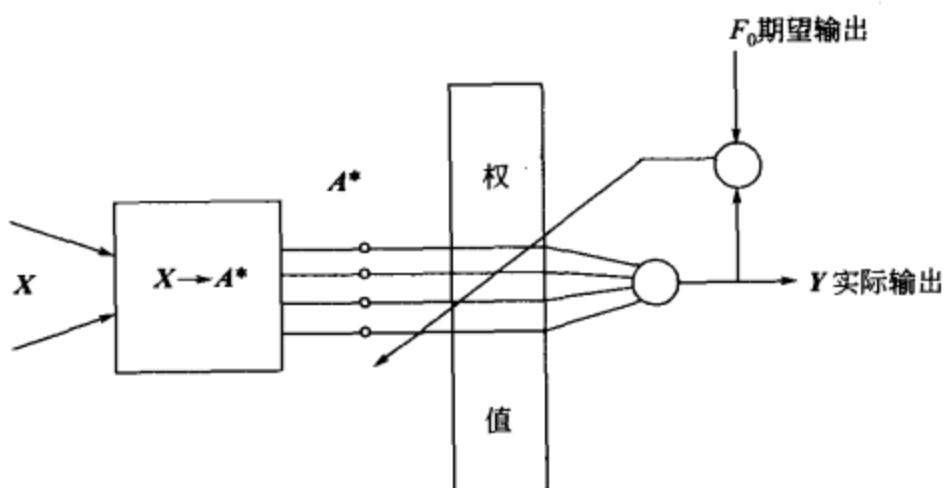


图 7.3 CMAC 网络的权值调整

7.3.2 CMAC 网络的特点

从神经网络的函数逼近功能这个角度来分, 神经网络可以分为全局逼近网络和局部逼近网络。当神经网络的一个或多个可调参数 (权值和阈值) 在输入空间的每一点对任何一个输出都有影响, 则称该神经网络为全局逼近网络, 如采用 BP 算法的多层前馈网络是全局逼近网络的典型例子。对于每个输入输出数据对, 网络的每一个连接权均需进行调整, 从而导致全局逼近网络学习速度很慢, 对于有实时性要求的应用来说常常是不可容忍的。如果对网络输入空间的某个局部区域只有少数几个连接权影响网络的输出, 则称该网络为局部逼近网络。对于每个输入输出数据对, 只有少量的连接权需要进行调整, 从而使局部逼近网络具有学习速度快的优点, 这一点对于有实时性要求的应用来说至关重要。从学习算法可以看出, CMAC 网络具有以下特点:

- ① CMAC 网络是一种局部逼近神经网络, 因此学习速度快, 不存在局部极小问题。
- ② CMAC 网络的泛化能力与感知器的泛化范围 C 相关, C 值增大则网络泛化能力增强。

③ CMAC 网络与 RBF 网络同为局部逼近网，但隐层的接收函数不同，且 CMAC 网络主要用于机器人控制中，而 RBF 网络的应用范围与 BP 网络类似。

④ 在非线性逼近方面，CMAC 网络用多个局部超平面拟合非线性超曲面，RBF 网络用多个局部超曲面拟合非线性超曲面，而 BP 网络用一个整体超曲面拟合非线性超曲面。

7.4 CMAC 网络的应用

综上所述可知，CMAC 是一种通过多种映射实现联想记忆的神经网络。这种映射实际上是一种智能查表技术，它模拟了小脑皮层神经系统感受信息和存储信息，并通过联想利用信息的功能。CMAC 网络不仅学习速度快，而且精度高，在智能控制领域具有重要应用价值。

7.4.1 CMAC 网络在机器人手臂协调控制中的应用

图 7.4 中给出一个 CMAC 与机器人关节臂相连的系统。设 θ 、 θ' 和 θ'' 分别表示机器人手臂关节的角度向量、角速度向量和角加速度向量， T 表示机器人手臂关节的驱动力矩向量，机器人关节的动力学方程为：

$$\theta'' = g(\theta, \theta', T)$$

为使机器人关节获得角加速度，须在关节上施加一定的力矩 T ，其表达式应为：

$$T = g^{-1}(\theta, \theta', \theta'')$$

式中， g^{-1} 为 g 函数的逆函数，描述了机器人关节的动力学特性。若 g 已知且 g^{-1} 存在，可用上式计算 T 。当 g^{-1} 未知时，可用 CMAC 网络学习函数 g^{-1} ，从而使 CMAC 的输出 Y 与 T 一致。

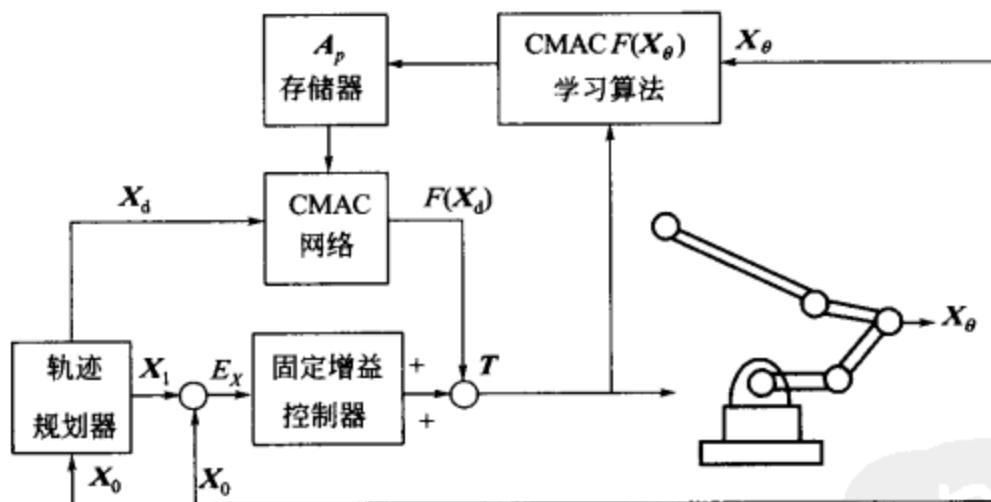


图 7.4 CMAC 网络用于机器人关节控制

当用该系统控制机器人的一个手臂关节时，变量 θ 、 θ' 和 θ'' 均为标量，其量化值 x_θ 、 $x_{\theta'}$ 、 $x_{\theta''}$ 共同构成了 CMAC 网络的三维输入空间 X_θ 。系统工作过程如下：将对应于机器人手臂关节实际状态的输入 X_θ 加到一个 CMAC 学习算法上，该算法输出 $F(x_\theta)$ ，学习过程中的权值存放在 A_p 的存储单元中。图中另一个 CMAC 网络是专供输出使用的，两个 CMAC 网共用 A_p 存储器。学习 CMAC 网负责将调整后的权值存入 A_p 存储器，而输出 CMAC 网负责根据 A_p 中存放的权值和期望状态 X_d 产生输出 $F(X_d)$ 。

在系统的每个控制周期，由轨迹规划器产生一个理想状态 X_i ，而机器人的实际输出状态为 x_θ ，两者之差为 E_x 。该误差经过固定增益控制器产生误差驱动力矩 T 。此外，轨迹规

划器还根据系统的实际状态 x_θ 和下一控制周期的理想状态 X_i 规划出系统的期望状态 X_d ，以其作为输出 CMAC 网络的输入。系统开始运行时， A_p 存储器中权值为零，所以第一次运行时 CMAC 网络的输出为 $F(X_d)=0$ 。固定增益控制器将误差放大后直接作为初始驱动力矩去控制机器人的手臂关节。在下一个控制周期，根据系统的实际输出 x_θ 状态，学习 CMAC 网络通过 CMAC 算法计算出权值调整量为：

$$\Delta W = \frac{\mu[T - F(x_\theta)]}{|A^*|}$$

式中， T 为上一控制周期中实际施加于机器人手臂的力矩， $F(x_\theta)$ 为 CMAC 网络在 x_θ 输入后得到的输出。调整后的权值存入 A_p 存储器后，输出 CMAC 网络根据期望状态产生的 $F(X_d)$ 不再为零。 $F(X_d)$ 与增益控制器输出的力矩相叠加得到驱动力矩 T 去控制机器人手臂。

经过几次训练后，机器人的手臂运动很快就与要求的轨迹相一致。训练结束后， x_θ 与 X_d 相同， $E_X=0$ ，因此驱动力矩 $T=F(X_d)$ ，即 $F(X_d)$ 体现了 $g^{-1}(\theta, \theta', \theta'')$ 的特性。系统工作时如受到外界干扰，会在机器人手臂运动中叠加一个错误扰动 x_θ' ，由该扰动产生的 $F(x_\theta')$ 使 CMAC 学习网络工作，对权值进行调整，系统很快会适应外界变化。

7.4.2 CMAC 网络在有源噪声控制中的应用

有源噪声控制（ANC）是通过人工产生的次级声源与噪声进行干涉，在期望的位置降低噪声级。下面讨论一个将 CMAC 网络与 ANC 系统相结合实现空调机噪声有源控制的应用实例。

空调机的有源噪声控制原理如图 7.5 所示。图 7.5(a) 给出前馈有源噪声控制结构的几何布放系统简图。空调机产生的噪声为初级声源，该噪声声波通过空气介质传播，检测器检测到该噪声后传给控制器进行处理，控制器控制次级声源产生与噪声反相的声波。在观测点附近，初级声源和次级声源发出的声波产生干涉叠加，从而使噪声级降低。

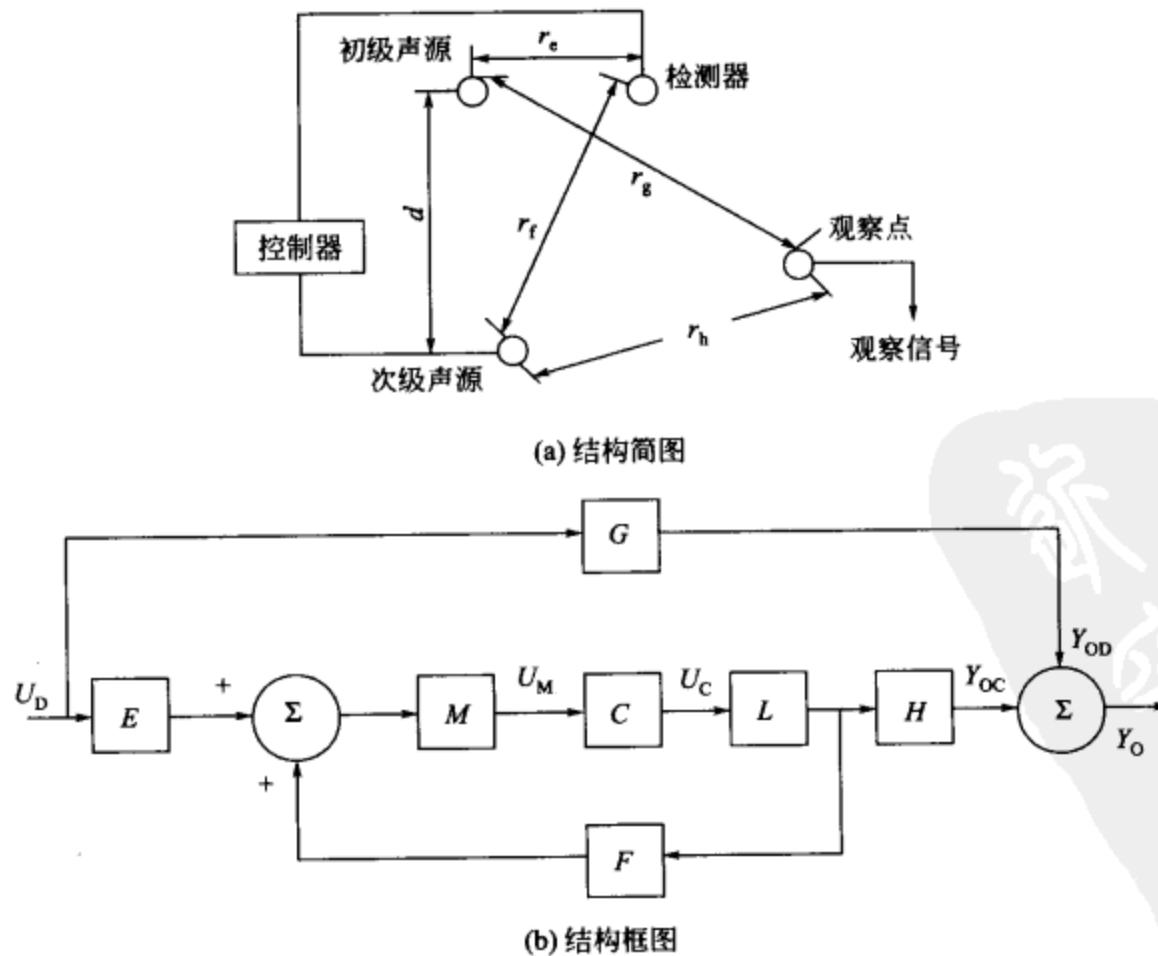


图 7.5 前馈有源噪声控制系统

图 7.5(b) 为该有源噪声控制结构的频域等效框图。图中的 E 、 F 、 G 和 H 分别表示距离为 r_e 、 r_f 、 r_g 和 r_h 的声波通道的传递函数， M 、 C 和 L 分别表示检测器、控制器和次级声源的传输特性， U_D 和 U_C 分别为初级声源和次级声源的噪声， Y_{OD} 和 Y_{OC} 分别对应初级和次级声源在观测点的信号， U_M 是检测信号， Y_O 是观测点处的信号。

若能使初级信号与次级信号在观测点处振幅相等且相位差为 180°，则 $Y_O=0$ ，相当于在随机环境中达到最优降噪。为此需满足：

$$Y_O = Y_{OD} + Y_{OC} = 0$$

由图 7.5(b) 得 $Y_{OD} = U_D G$ ， $Y_{OC} = [(U_D E M C L) / (1 - F M C L)] H$ ，将 Y_{OD} 和 Y_{OC} 代入上式，经整理可得在观测点对宽带噪声理想降噪所需的控制器传递函数为：

$$C = G / (F G - E H)$$

系统部件的布放方案确定后，通过实验和计算可得到系统各环节的传递特性 E 、 F 、 G 和 H 及相应的脉冲响应，因此可用实验测得的空调噪声作为初级声源，通过 MATLAB 仿真计算出使 $Y_O=0$ 的理想控制器输出特性。由于系统各环节传递特性准确度及计算窗尺寸的影响，图 7.6 所示为理想控制器降噪前后自功率谱密度的最好结果。因此，可以用该理想控制器的输入及相应的输出结果训练 CMAC 网络，实现神经网络控制器的设计。

使用上述 CMAC 控制器，在观测点得到 ANC 系统对空调机噪声的控制结果。图 7.7 为

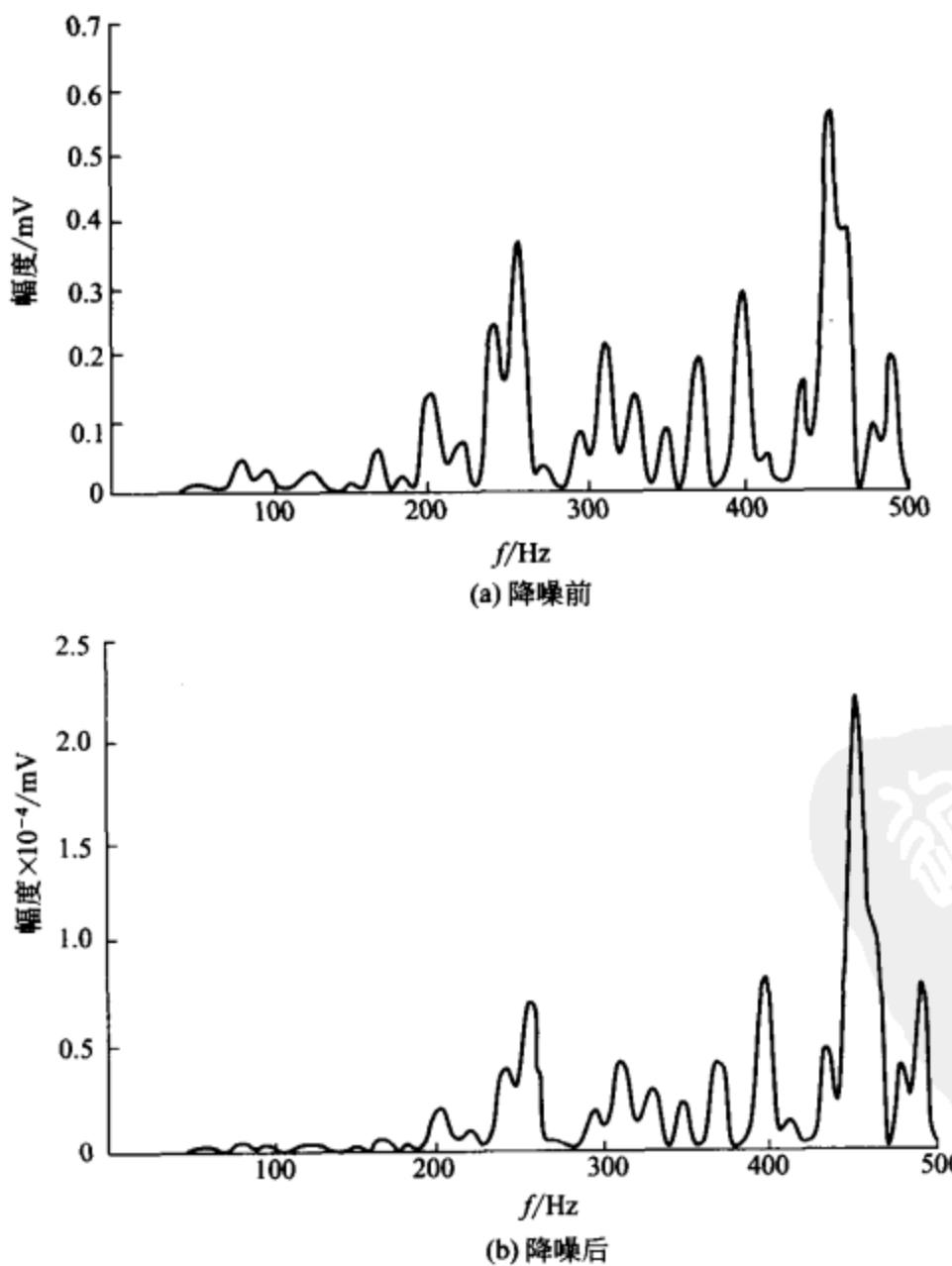


图 7.6 理想控制器降噪前后的功率谱

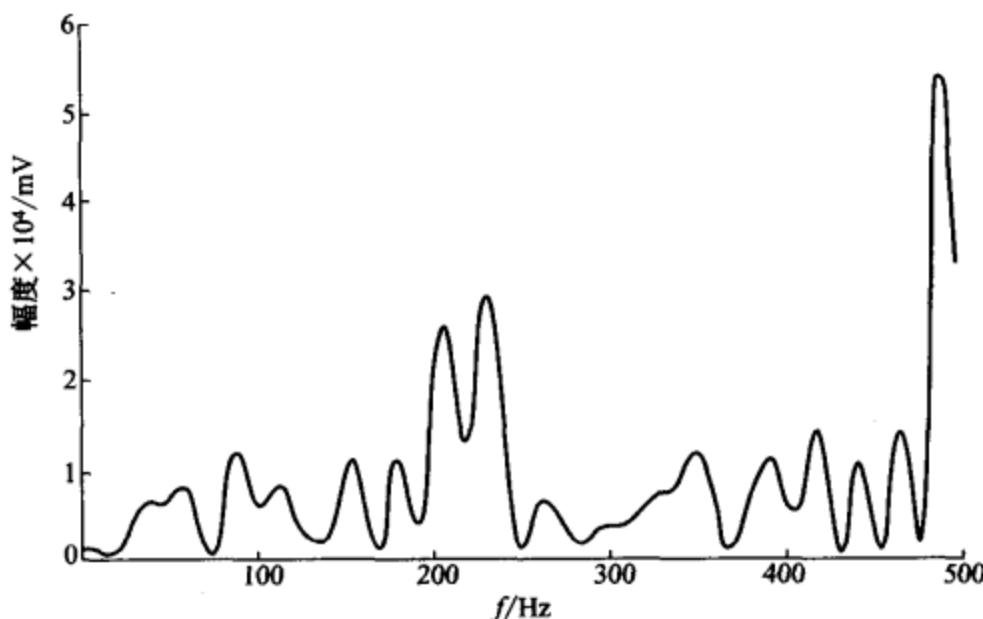


图 7.7 CMAC 降噪后的功率谱

采用 CMAC 控制器降噪后的自功率谱密度，与图 7.6(a) 比较可以看出，神经自适应有源噪声控制系统在窗式空调机降噪中取得了良好的效果。

本章小结

CMAC 网络是一种仿照小脑控制肢体运动的原理而建立的神经网络模型，因学习收敛速度快，精度较高，在实时工作时非常有用。CMAC 网络最初主要用来求解机械手的关节运动，其后进一步用于机器人控制、模式识别、信号处理以及自适应控制等领域。CMAC 有三个特点：

① 作为一种具有联想功能的神经网络，它的联想具有局部泛化能力，因此相似的输入将产生相似的输出，远离的输入将产生独立的输出。

② 对于网络的每一输出，只有很少的神经元所对应的权值对其有影响，哪些神经元对输出有影响则由输入决定。

③ CMAC 的每个神经元的输入输出是一种线性关系，但其总体上可看作一种表达非线性映射的表格系统。

CMAC 网络的学习只在线性映射部分，因此可采用简单的 δ 算法，其收敛速度比 BP 算法快得多，且不存在局部极小问题。

思考与练习

- 7.1 试述 CMAC 网络的工作原理。
- 7.2 CMAC 网络的 4 种映射各自有何作用？
- 7.3 比较 CMAC 网络与 RBF 网络、BP 网络的异同。
- 7.4 为什么说 CMAC 网络的联想具有局部泛化能力？

8 基于数学原理的神经网络

前面各章讨论的各类神经网络均受到某种生物学原理的启发。本章将要讨论另一类神经网络，这类网络模型的建立均受到某种数学原理的指导。

针对某类问题，人们常常能够从数学上提出相应的解决思路。但由于问题的复杂性和不确定性，描述解决思路的数学方程往往难以求解。基于求解问题的数学原理，可在原理性方法的指导下构造出相应的神经网络模型，使其通过对样本的学习自动实现问题的求解。本章讨论三种基于数学原理的神经网络模型，即基于多变量有限点严格（精确）插值技术的径向基函数网络、基于 K-L 变换方法的主分量分析网络以及基于最优超平面构建的支持向量机网络。

8.1 径向基函数 RBF

8.1.1 基于径向基函数技术的函数逼近与内插

理解 RBF 网络的工作原理可从两种不同的观点出发：①当用 RBF 网络解决非线性映射问题时，用函数逼近与内插的观点来解释，对于其中存在的不适定（ill posed）问题，可用正则化理论来解决；②当用 RBF 网络解决复杂的模式分类任务时，用模式可分性观点来理解比较方便，其潜在合理性基于 Cover 关于模式可分的定理。下面阐述基于函数逼近与内插观点的工作原理。

1963 年 Davis 提出高维空间的多变量插值理论。径向基函数技术则是 20 世纪 80 年代后期，Powell 在解决“多变量有限点严格（精确）插值问题”时引入的，目前径向基函数已成为数值分析研究中的一个重要领域。

考虑一个由 N 维输入空间到一维输出空间的映射。设 N 维空间有 P 个输入向量 \mathbf{X}^p , $p=1,2,\dots,P$ ，它们在输出空间相应的目标值为 d^p , $p=1,2,\dots,P$, P 对输入-输出样本构成了训练样本集。插值的目的是寻找一个非线性映射函数 $F(\mathbf{X})$ ，使其满足下述插值条件：

$$F(\mathbf{X}^p) = d^p \quad p=1,2,\dots,P \quad (8.1)$$

式中，函数 F 描述了一个插值曲面。所谓严格插值或精确插值，是一种完全内插，即该插值曲面必须通过所有训练数据点。

采用径向基函数技术解决插值问题的方法是，选择 P 个基函数，每一个基函数对应一个训练数据，各基函数的形式为：

$$\varphi(\|\mathbf{X}-\mathbf{X}^p\|) \quad p=1,2,\dots,P \quad (8.2)$$

式中，基函数 φ 为非线性函数，训练数据点 \mathbf{X}^p 是 φ 的中心。基函数以输入空间的点 \mathbf{X} 与中心 \mathbf{X}^p 的距离作为函数的自变量。由于距离是径向同性的，故函数 φ 被称为径向基函

数。基于径向基函数技术的插值函数定义为基函数的线性组合：

$$F(\mathbf{X}) = \sum_{p=1}^P w_p \varphi(\|\mathbf{X} - \mathbf{X}^p\|) \quad (8.3)$$

将式(8.1)的插值条件代入上式，得到 P 个关于未知系数 w^p , $p=1, 2, \dots, P$ 的线性方程组：

$$\begin{aligned} \sum_{p=1}^P w^p \varphi(\|\mathbf{X}^1 - \mathbf{X}^p\|) &= d^1 \\ \sum_{p=1}^P w^p \varphi(\|\mathbf{X}^2 - \mathbf{X}^p\|) &= d^2 \\ &\vdots \\ \sum_{p=1}^P w^p \varphi(\|\mathbf{X}^P - \mathbf{X}^p\|) &= d^P \end{aligned} \quad (8.4)$$

令 $\varphi_{ip} = \varphi(\|\mathbf{X}^i - \mathbf{X}^p\|)$, $i=1, 2, \dots, P$, $p=1, 2, \dots, P$, 则上述方程组可改写为：

$$\begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1P} \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2P} \\ \vdots & \vdots & & \vdots \\ \varphi_{P1} & \varphi_{P2} & \cdots & \varphi_{PP} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix} = \begin{bmatrix} d^1 \\ d^2 \\ \vdots \\ d^P \end{bmatrix} \quad (8.5)$$

令 Φ 表示元素为 φ_{ip} 的 $P \times P$ 阶矩阵， \mathbf{W} 和 \mathbf{d} 分别表示系数向量和期望输出向量，式(8.5)还可写成下面的向量形式：

$$\Phi \mathbf{W} = \mathbf{d} \quad (8.6)$$

式中， Φ 称为插值矩阵。若 Φ 为可逆矩阵，就可以从式(8.6)中解出系数向量 \mathbf{W} ，即：

$$\mathbf{W} = \Phi^{-1} \mathbf{d} \quad (8.7)$$

如何保证插值矩阵的可逆性？Micchelli 定理给出了如下条件：

对于一大类函数，如果 $\mathbf{X}^1, \mathbf{X}^2, \dots, \mathbf{X}^P$ 各不相同，则 $P \times P$ 阶插值矩阵是可逆的。

大量径向基函数满足 Micchelli 定理，如式(8.8)~式(8.10)所示，其曲线形状分别如图 8.1 所示。

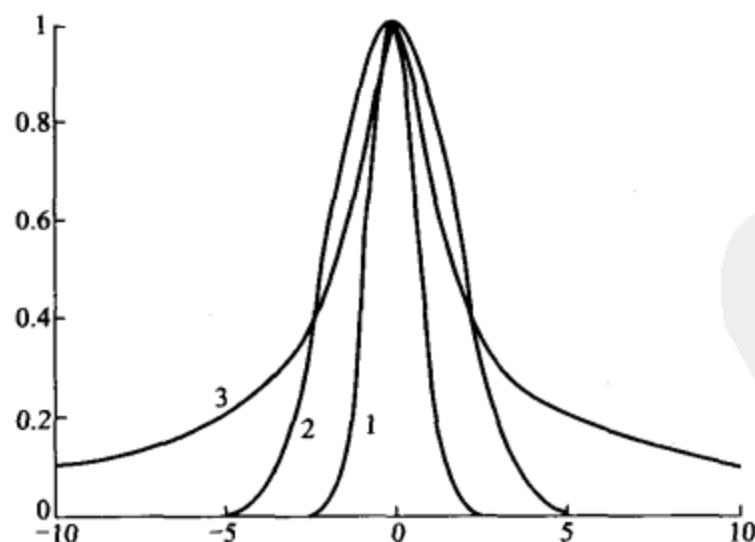


图 8.1 3 种常用的径向基函数

(1) Gauss (高斯) 函数

$$\varphi(r) = \exp\left(-\frac{r^2}{2\delta^2}\right) \quad (8.8)$$

(2) Reflected sigmoidal (反演 S 型) 函数

$$\varphi(r) = \frac{1}{1 + \exp\left(\frac{r^2}{\delta^2}\right)} \quad (8.9)$$

(3) Inverse multiquadratics (拟多二次) 函数

$$\varphi(r) = \frac{1}{(r^2 + \delta^2)^{\frac{1}{2}}} \quad (8.10)$$

式(8.8)~式(8.10) 中的 δ 称为该基函数的扩展常数或宽度, 从图 8.1 可以看出, 径向基函数的宽度越小, 就越具有选择性。

8.1.2 正则化 RBF 神经网络

8.1.2.1 正则化 RBF 网络的结构及特点

正则化 RBF 网络的结构如图 8.2 所示。其特点是: 网络具有 N 个输入节点, P 个隐节点, l 个输出节点; 网络的隐节点数等于输入样本数, 隐节点的激活函数常具有式(8.8) 所示的 Gauss 形式, 并将所有输入样本设为径向基函数的中心, 各径向基函数取统一的扩展常数。

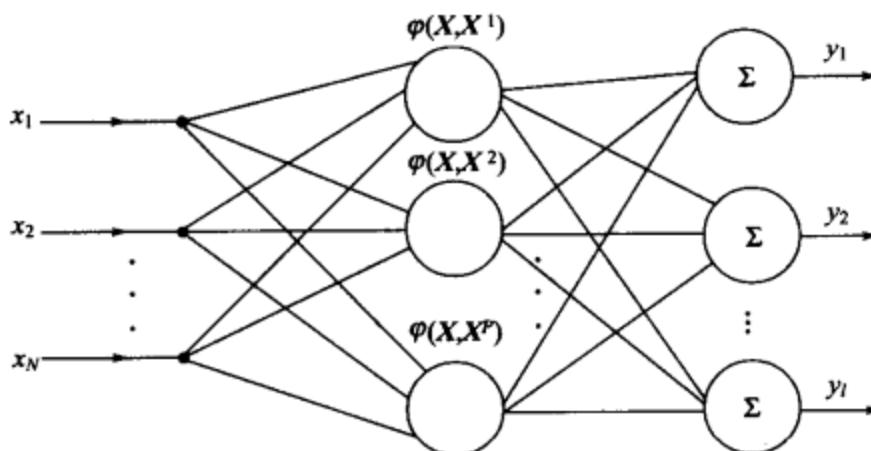


图 8.2 正则化 RBF 网络

设输入层的任一节点用 i 表示, 隐层的任一节点用 j 表示, 输出层的任一节点用 k 表示。对各层的数学描述如下: $\mathbf{X}=(x_1, x_2, \dots, x_N)^T$ 为网络输入向量; $\varphi_j(\mathbf{X})$ ($j=1, 2, \dots, P$) 为任一隐节点的激活函数, 称为“基函数”, 一般选用 Gauss 函数; \mathbf{W} 为输出权矩阵, 其中 w_{jk} ($j=1, 2, \dots, P$; $k=1, 2, \dots, l$) 为隐层第 j 个节点与输出层第 k 个节点间的突触权值; $\mathbf{Y}=(y_1, y_2, \dots, y_l)^T$ 为网络输出; 输出层神经元采用线性激活函数。

当输入训练集中的某个样本 \mathbf{X}^p 时, 对应的期望输出 d^p 就是教师信号。为了确定网络隐层到输出层之间的 P 个权值, 需要将训练集中的样本逐一输入一遍, 从而可得到式(8.4) 中的方程组。网络的权值确定后, 对训练集的样本实现了完全内插, 即对所有样本误差为 0。而对非训练集的输入模式, 网络的输出值相当于函数的内插, 因此径向基函数网络可用作函数逼近。

正则化 RBF 网络具有以下 3 个特点:

① 正则化网络是一种通用逼近器, 只要有足够的隐节点, 它可以以任意精度逼近紧集上的任意多元连续函数。

② 具有最佳逼近特性, 即任给一个未知的非线性函数 f , 总可以找到一组权值使得正则化网络对于 f 的逼近优于所有其他可能的选择。

③ 正则化网络得到的解是最佳的，所谓“最佳”体现在同时满足对样本的逼近误差和逼近曲线的平滑性。

8.1.2.2 正则化 RBF 网络的学习算法

当采用正则化 RBP 网络结构时，隐节点数即样本数，基函数的数据中心即为样本本身，只需考虑扩展常数和输出节点的权值。

径向基函数的扩展常数可根据数据中心的散布而确定，为了避免每个径向基函数太尖或太平，一种选择方法是将所有径向基函数的扩展常数设为：

$$\delta = \frac{d_{\max}}{\sqrt{2P}} \quad (8.11)$$

式中， d_{\max} 是样本之间的最大距离； P 是样本的数目。

输出层的权值常采用第 2 章介绍的最小均方算法（LMS），LMS 算法的输入向量即隐节点的输出向量。权值调整公式为：

$$\Delta \mathbf{W}_k = \eta(d_k - \mathbf{W}_k^T \Phi) \Phi \quad (8.12a)$$

$\Delta \mathbf{W}_k$ 的各分量为：

$$\Delta w_{jk} = \eta(d_k - \mathbf{W}_k^T \Phi) \varphi_j \quad j=0, 1, \dots, P; k=1, 2, \dots, l \quad (8.12b)$$

权值可初始化为任意值。

8.1.3 广义 RBF 神经网络

8.1.3.1 模式的可分性

下面通过研究模式的可分性来深入了解 RBF 网络作为模式分类器是如何工作的。

从第 3 章关于单层感知器的讨论可知，若 N 维输入样本空间的样本模式是线性可分的，总存在一个用线性方程描述的超平面，使两类线性可分样本截然分开。若两类样本是非线性可分的，则不存在一个这样的分类超平面。但根据 Cover 定理，非线性可分问题可能通过非线性变换获得解决。

Cover 定理可以定性地表述为：将复杂的模式分类问题非线性地投射到高维空间比投射到低维空间更可能是线性可分的。

设 F 为 P 个输入模式 X^1, X^2, \dots, X^P 的集合，其中每一个模式必属于两个类 F_1 和 F_2 中的某一类。若存在一个输入空间的超曲面，使得分别属于 F_1 和 F_2 的点（模式）分成两部分，就称这些点的二元划分关于该曲面是可分的；若该曲面为线性方程 $\mathbf{W}^T \mathbf{X} = 0$ 确定的超平面，则称这些点的二元划分关于该平面是线性可分的。设有一组函数构成的向量 $\varphi(\mathbf{X}) = [\varphi_1(\mathbf{X}), \varphi_2(\mathbf{X}), \dots, \varphi_M(\mathbf{X})]$ ，将原来 N 维空间的 P 个模式点映射到新的 M 维空间 ($M > N$) 相应点上，如果在该 M 维 φ 空间存在 M 维向量 \mathbf{W} ，使得：

$$\begin{cases} \mathbf{W}^T \varphi(\mathbf{X}) > 0, & \mathbf{X} \in F^1 \\ \mathbf{W}^T \varphi(\mathbf{X}) < 0, & \mathbf{X} \in F^2 \end{cases}$$

则由线性方程 $\mathbf{W}^T \varphi(\mathbf{X}) = 0$ 确定了 M 维 φ 空间中的一个分界超平面，这个超平面使得映射到 M 维 φ 空间中的 P 个点在 φ 空间是线性可分的。而在 N 维 \mathbf{X} 空间，方程 $\mathbf{W}^T \varphi(\mathbf{X}) = 0$ 描述的是 \mathbf{X} 空间的一个超曲面，这个超曲面使得原来在 \mathbf{X} 空间非线性可分的 P 个模式点分为两类，此时称原空间的 P 个模式点是可分的。

在 RBF 网络中，将输入空间的模式点非线性地映射到一个高维空间的方法是，设置一个隐层，令 $\varphi(\mathbf{X})$ 为隐节点的激活函数，并令隐节点数 M 大于输入节点数 N ，从而形成一

个维数高于输入空间的高维隐（藏）空间。如果 M 够大，则在隐空间输入是线性可分的，从隐层到输出层，可采用与第 3 章单层感知器类似的解决线性可分问题的算法。

Cover 定理关于模式可分性思想的要点是“非线性映射”和“高维空间”。事实上，对于不太复杂的非线性模式分类问题，有时仅使用非线性映射就可以使模式在变换后的同维空间变得线性可分。下面通过解决 XOR（异或）问题进一步理解模式的 ϕ 可分性。

如图 8.3(a) 所示，XOR 问题中的 4 个模式在二维输入空间的分布是非线性可分的。设计一个单隐层神经网络，定义其两个隐节点的激活函数为 Gauss 函数：

$$\varphi_1(\mathbf{X}) = e^{-\|\mathbf{X} - \mathbf{C}_1\|^2} \quad \mathbf{C}_1 = [1, 1]^T$$

$$\varphi_2(\mathbf{X}) = e^{-\|\mathbf{X} - \mathbf{C}_2\|^2} \quad \mathbf{C}_2 = [0, 0]^T$$

轮流以 XOR 问题的 4 个模式作为 2 个隐节点激活函数的输入，其对应的 4 个输出为： $(0,0) \rightarrow (0.1353, 1)$, $(0,1) \rightarrow (0.3678, 0.3678)$, $(1,0) \rightarrow (0.3678, 0.3678)$, $(1,1) \rightarrow (1, 0.1353)$ 。可以看出，隐节点的上述非线性映射将模式 $(0,1)$ 和 $(1,0)$ 映射为隐空间中的同一个点 $(0.3678, 0.3678)$ 。因此，在图 8.3(a) 的输入空间中非线性可分的点映射到图 8.3(b) 的隐空间后成为线性可分的点。

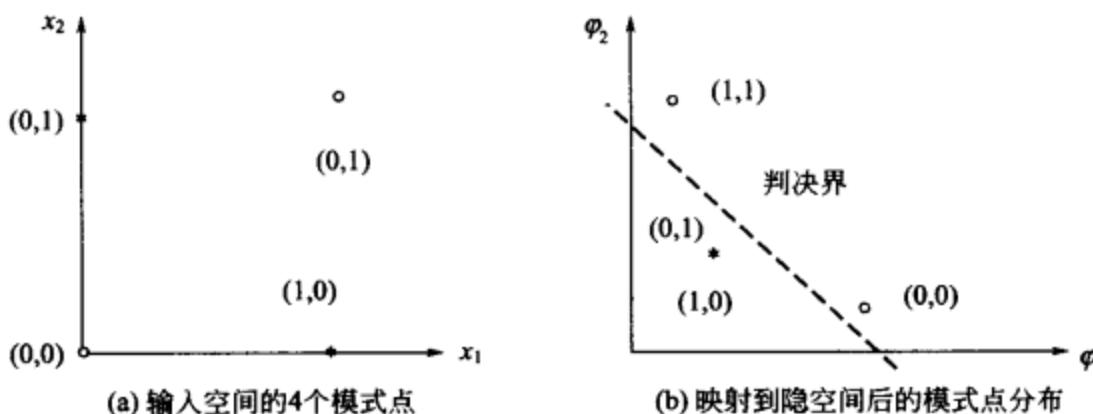


图 8.3 XOR 问题的 4 个模式在输入空间和隐空间的分布

在本例中，隐空间的维数和输入空间的维数相同，可见，仅采用 Gauss 函数进行非线性变换，就足以将 XOR 问题转化为一个线性可分问题。

8.1.3.2 广义 RBF 网络

由于正则化网络的训练样本与“基函数”是一一对应的。当样本数 P 很大时，实现网络的计算量将大得惊人，此外 P 很大则权值矩阵也很大，求解网络的权值时容易产生病态问题 (ill conditioning)。为解决这一问题，可减少隐节点的个数，即 $N < M < P$ ， N 为样本维数， P 为样本个数，从而得到广义 RBF 网络。

广义 RBF 网络的基本思想是：用径向基函数作为隐单元的“基”，构成隐含层空间。隐含层对输入向量进行变换，将低维空间的模式变换到高维空间内，使得在低维空间内的线性不可分问题在高维空间内线性可分。

图 8.4 所示为 $N-M-l$ 结构的 RBF 网，即网络具有 N 个输入节点， M 个隐节点， l 个输出节点，且 $M < P$ 。 $\mathbf{X} = (x_1, x_2, \dots, x_N)^T$ 为网络输入向量； $\varphi_j(\mathbf{X})$ ($j = 1, 2, \dots, M$) 为任一隐节点的激活函数，称为“基函数”，一般选用格林 (Green) 函数； \mathbf{W} 为输出权矩阵，其中 w_{jk} ($j = 1, 2, \dots, M$; $k = 1, 2, \dots, l$) 为隐层第 j 个节点与输出层第 k 个节点间的突触权值； $\mathbf{T} = (T_1, T_2, \dots, T_l)^T$ 为输出层阈值向量； $\mathbf{Y} = (y_1, y_2, \dots, y_l)^T$ 为网络输出；输出层神经元采用线性激活函数。

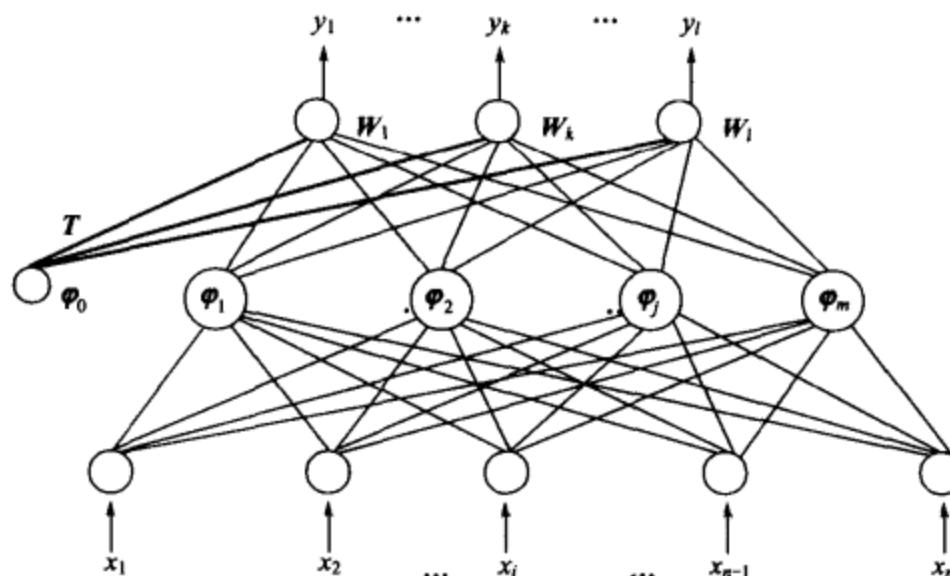


图 8.4 广义 RBF 网络

与正则化 RBF 网络相比，广义 RBF 网络有以下几点不同：

- ① 径向基函数的个数 M 与样本的个数 P 不相等，且 M 常常远小于 P 。
- ② 径向基函数的中心不再限制在数据点上，而是由训练算法确定。
- ③ 各径向基函数的扩展常数不再统一，其值由训练算法确定。
- ④ 输出函数的线性中包含阈值参数，用于补偿基函数在样本集上的平均值与目标值之间的差别。

8.1.3.3 广义 RBF 网络的设计方法

广义 RBP 网络的设计包括结构设计和参数设计。结构设计主要解决如何确定网络隐节点数的问题，参数设计一般需考虑包括三种参数：各基函数的数据中心和扩展常数，以及输出节点的权值。

根据数据中心的取值方法，广义 RBF 网的设计方法可分为两类：

- (1) 第一类方法：数据中心从样本输入中选取 一般来说，样本密集的地方中心点可以适当多些，样本稀疏的地方中心点可以少些；若数据本身是均匀分布的，中心点也可以均匀分布。总之，选出的数据中心应具有代表性。径向基函数的扩展常数是根据数据中心的散布而确定的，为了避免每个径向基函数太尖或太平，一种选择方法是将所有径向基函数的扩展常数设为：

$$\delta = \frac{d_{\max}}{\sqrt{2M}} \quad (8.13)$$

式中， d_{\max} 是所选数据中心之间的最大距离； M 是数据中心的数目。

- (2) 第二类方法：数据中心的自组织选择 常采用各种动态聚类算法对数据中心进行自组织选择，在学习过程中需对数据中心的位置进行动态调节，常用的方法是 K-means 聚类，其优点是能根据各聚类中心之间的距离确定各隐节点的扩展常数。由于 RBF 网的隐节点数对其泛化能力有极大的影响，所以寻找能确定聚类数目的合理方法，是聚类方法设计 RBF 网时需首先解决的问题。除聚类算法外，还有梯度训练方法、资源分配网络（RAN）等。

8.1.3.4 广义 RBF 网络数据中心的聚类算法

1989 年，Moody 和 Darken 提出一种由两个阶段组成的混合学习过程的思路。第一阶段常采用 Duda 和 Hart 1973 年提出的 K-means 聚类算法，其任务是用自组织聚类方法为隐

层节点的径向基函数确定合适的数据中心，并根据各中心之间的距离确定隐节点的扩展常数。第二阶段为监督学习阶段，其任务是用有监督学习算法训练输出层权值，一般采用梯度法进行训练。

在聚类确定数据中心的位置之前，需要先估计中心的个数 M （从而确定了隐节点数），一般需要通过试验来决定。由于聚类得到的数据中心不是样本数据 \mathbf{X}^p 本身，因此用 $\mathbf{c}(k)$ 表示第 k 次迭代时的中心。应用 K-means 聚类算法确定数据中心的过程如下：

(1) 初始化 选择 M 个互不相同向量作为初始聚类中心： $\mathbf{c}_1(0), \mathbf{c}_2(0), \dots, \mathbf{c}_M(0)$ ，选择时可采用对各聚类中心向量赋小随机数的方法。

(2) 计算输入空间各样本点与聚类中心点的欧式距离：

$$\|\mathbf{X}^p - \mathbf{c}_j(k)\| \quad p=1, 2, \dots, P; j=1, 2, \dots, M$$

(3) 相似匹配 令 j^* 代表竞争获胜隐节点的下标，对每一个输入样本 \mathbf{X}^p 根据其与聚类中心的最小欧式距离确定其归类 $j^*(\mathbf{X}^p)$ ，即当有如下等式时：

$$j^*(\mathbf{X}^p) = \min_j \|\mathbf{X}^p - \mathbf{c}_j(k)\| \quad p=1, 2, \dots, P \quad (8.14)$$

\mathbf{X}^p 被归为第 j^* 类，从而将全部样本划分为 M 个子集： $\mathbf{U}_1(k), \mathbf{U}_2(k), \dots, \mathbf{U}_M(k)$ ，每个子集构成一个以聚类中心为典型代表的聚类域。

(4) 更新各类的聚类中心 可采用两种调整方法，一种方法是对各聚类域中的样本取均值，令 $\mathbf{U}_j(k)$ 表示第 j 个聚类域， N_j 为第 j 个聚类域中的样本数，则：

$$\mathbf{c}_j(k+1) = \frac{1}{N_j} \sum_{\mathbf{x} \in \mathbf{U}_j(k)} \mathbf{x} \quad (8.15)$$

另一种方法是采用竞争学习规则进行调整，即：

$$\mathbf{c}_j(k+1) = \begin{cases} \mathbf{c}_j(k) + \eta[\mathbf{X}^p - \mathbf{c}_j(k)], & j=j^* \\ \mathbf{c}_j(k), & j \neq j^* \end{cases} \quad (8.16)$$

式中， η 是学习率，且 $0 < \eta < 1$ 。可以看出，当 $\eta=1$ 时，该竞争规则即为 Winner-Take-All 规则。

(5) 将 k 值加 1，转到第 (2) 步。重复上述过程直到 \mathbf{c}_k 的改变量小于要求的值。

各聚类中心确定后，可根据各中心之间的距离确定对应径向基函数的扩展常数。令：

$$d_j = \min_i \|\mathbf{c}_j - \mathbf{c}_i\|$$

则扩展常数取：

$$\delta_j = \lambda d_j \quad (8.17)$$

式中， λ 为重叠系数。

利用 K-means 聚类算法得到各径向基函数的中心和扩展常数后，混合学习过程的第二步是用有监督学习算法得到输出层的权值，常采用 LMS 算法。更简捷的方法是用伪逆法直接计算。以单输出 RBF 网络为例，设输入为 \mathbf{X}^p 时，第 j 个隐节点的输出为 $\varphi_{pj} = \varphi(\|\mathbf{X}^p - \mathbf{c}_j\|)$ ， $p=1, 2, \dots, P$ ， $j=1, 2, \dots, M$ ，则隐层输出矩阵为：

$$\hat{\Phi} = [\varphi_{pj}]_{P \times M}$$

若 RBF 网络的待定输出权值为 $\mathbf{W} = [w_1, w_2, \dots, w_M]$ ，则网络输出向量为：

$$\mathbf{F}(\mathbf{X}) = \hat{\Phi} \mathbf{W} \quad (8.18)$$

令网络输出向量等于教师信号 \mathbf{d} ，则 \mathbf{W} 可用 $\hat{\Phi}$ 的伪逆 $\hat{\Phi}^+$ 求出：

$$\mathbf{W} = \hat{\Phi}^+ \mathbf{d} \quad (8.19)$$

$$\hat{\Phi}^+ = (\hat{\Phi}^T \hat{\Phi})^{-1} \hat{\Phi}^T \quad (8.20)$$

8.1.3.5 广义 RBF 网络数据中心的监督学习算法

关于数据中心的监督学习算法，最一般的情况是对输出层各权向量赋小随机数并进行归一化处理，隐节点 RBF 函数的中心、扩展常数和输出层权值均采用监督学习算法进行训练，即所有参数都经历一个误差修正学习过程，其方法与第 3 章采用 BP 算法训练 BP 网络的原理类似。下面以单输出 RBF 网络为例，介绍一种梯度下降算法。

定义目标函数为：

$$E = \frac{1}{2} \sum_{i=1}^P e_i^2 \quad (8.21)$$

式中， P 为训练样本数； e_i 为输入第 i 个样本时的误差信号，定义为：

$$e_i = d_i - \mathbf{F}(\mathbf{X}_i) = d_i - \sum_{j=1}^M w_j G(\|\mathbf{X}_i - \mathbf{c}_j\|) \quad (8.22)$$

上式的输出函数中忽略了阈值。

为使目标函数最小化，各参数的修正量应与其负梯度成正比，即：

$$\Delta c_j = -\eta \frac{\partial E}{\partial c_j}$$

$$\Delta \delta_j = -\eta \frac{\partial E}{\partial \delta_j}$$

$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j}$$

具体计算式为：

$$\Delta c_j = \eta \frac{w_j}{\delta_j^2} \sum_{i=1}^P e_i G(\|\mathbf{X}_i - \mathbf{c}_j\|) (\mathbf{X}_i - \mathbf{c}_j) \quad (8.23)$$

$$\Delta \delta_j = \eta \frac{w_j}{\delta_j^3} \sum_{i=1}^P e_i G(\|\mathbf{X}_i - \mathbf{c}_j\|) \|\mathbf{X}_i - \mathbf{c}_j\|^2 \quad (8.24)$$

$$\Delta w_j = \eta \sum_{i=1}^P e_i G(\|\mathbf{X}_i - \mathbf{c}_j\|) \quad (8.25)$$

上述目标函数是所有训练样本引起的误差的总和，导出的参数修正公式是一种批处理式调整，即所有样本输入一轮后调整一次。目标函数也可定义为瞬时值形式，即当前输入样本引起的误差：

$$E = \frac{1}{2} e_i^2 \quad (8.26)$$

使上式中目标函数最小化的参数修正式为单样本训练模式，即：

$$\Delta c_j = \eta \frac{w_j}{\delta_j^2} e_i G(\|\mathbf{X} - \mathbf{c}_j\|) (\mathbf{X} - \mathbf{c}_j) \quad (8.27)$$

$$\Delta \delta_j = \eta \frac{w_j}{\delta_j^3} e_i G(\|\mathbf{X} - \mathbf{c}_j\|) \|\mathbf{X} - \mathbf{c}_j\|^2 \quad (8.28)$$

$$\Delta w_j = \eta e_i G(\|\mathbf{X} - \mathbf{c}_j\|) \quad (8.29)$$

8.1.4 RBF 网络与 BP 网络的比较

RBF 网络与 BP 网络都是非线性多层前向网络，它们都是通用逼近器。对于任一个 BP 网络，总存在一个 RBF 网络可以代替它，反之亦然。但是，这两个网络也存在着很多不

同点：

- ① RBF 网络只有一个隐层，而 BP 网络的隐层可以是一层也可以是多层的。
- ② BP 网络的隐层和输出层其神经元模型是一样的；而 RBF 网络的隐层神经元和输出层神经元不仅模型不同，而且在网络中起到的作用也不一样。
- ③ RBF 网络的隐层是非线性的，输出层是线性的。然而，当用 BP 网络解决模式分类问题时，它的隐层和输出层通常选为非线性的。当用 BP 网络解决非线性回归问题时，通常选择线性输出层。
- ④ RBF 网络的基函数计算的是输入向量和中心的欧氏距离，而 BP 网络隐单元的激励函数计算的是输入单元和连接权值间的内积。
- ⑤ RBF 网络使用局部指数衰减的非线性函数（如高斯函数）对非线性输入输出映射进行局部逼近。BP 网络的隐节点采用输入模式与权向量的内积作为激活函数的自变量，而激活函数则采用 Sigmoid 函数或硬限幅函数，因此 BP 网络是对非线性映射的全局逼近。RBF 网最显著的特点是隐节点采用输入模式与中心向量的距离（如欧氏距离）作为函数的自变量，并使用径向基函数（如 Gauss 函数）作为激活函数。径向基函数关于 N 维空间的一个中心点具有径向对称性，而且神经元的输入离该中心点越远，神经元的激活程度就越低。隐节点的这个特性常被称为“局部特性”。

由于 RBF 网络能够逼近任意的非线性函数，可以处理系统内在的难以解析的规律性，并且具有很快的学习收敛速度，因此 RBF 网络有较为广泛的应用。目前 RBF 网络已成功地用于非线性函数逼近、时间序列分析、数据分类、模式识别、信息处理、图像处理、系统建模、控制和故障诊断等。

8.1.5 RBF 网络设计应用实例

8.1.5.1 RBF 网络在液化气销售量预测中的应用

某液化气公司两年液化气销售量如表 8.1 所示。为预测未来年月的液化气销售量，以表 8.1 中的 24 组数据作为训练样本，再加上季节性因素、月度指数、周期系数和突发系数等，共计有 5 个影响销售量的因素。设计一个 RBF 网络作为预测模型，通过反复试验，确定隐层设 12 个数据中心，因此对于该 RBF 网络有： $P=24$ ， $N=5$ ， $M=12$ ，满足 $N < M < P$ 。

表 8.1 某液化气公司两年液化气销售量 (单位: kg)

年月	销售量	年月	销售量	年月	销售量	年月	销售量
2000. 1	5230	2000. 7	6000	2001. 1	5400	2001. 7	6500
2000. 2	5000	2000. 8	6200	2001. 2	5100	2001. 8	7000
2000. 3	5200	2000. 9	6200	2001. 3	5300	2001. 9	6800
2000. 4	5400	2000. 10	6050	2001. 4	5500	2001. 10	6500
2000. 5	5500	2000. 11	5500	2001. 5	5850	2001. 11	6250
2000. 6	5800	2000. 12	5400	2001. 6	6200	2001. 12	6000

采用梯度下降算法对数据中心、扩展常数和权值等网络参数进行训练，参数调整采用式(8.23)~式(8.25)。

训练前需要对网络参数进行初始化，对不同的参数应采用不同的方法。例如，可根据经验从输入样本中选取 12 个作为数据中心的初始值，再利用式(8.13) 得到扩展常数的初始值，权重的初始化则可采用较小的随机数。

8.1.5.2 RBF 网络在地表水质评价中的应用

《地表水环境质量标准》(GHZB—1999)与某市1998年7个地表水点的监测数据分别见表8.2和表8.3。

表 8.2 地表水质评价标准 (单位: mg/L)

评价指标	I 级	II 级	III 级	IV 级	V 级
DO *	0.1111	0.1667	0.2000	0.3333	0.5000
COD _{Mn}	2	4	8	10	15
COD _{Cr}	15	16	20	30	40
BOD ₅	2	3	4	6	10
NH ₄ -N	0.4	0.5	0.6	1.0	1.5
挥发酚	0.001	0.003	0.005	0.010	0.100
总砷	0.01	0.05	0.07	0.10	0.11
Cr ⁺⁶	0.01	0.03	0.05	0.07	0.10

注: DO * 代表 DO 的倒数(表8.3同)。

表 8.3 地表水质监测数据 (单位: mg/L)

评价指标	待评样本						
	1	2	3	4	5	6	7
DO *	0.1925	0.3130	0.1587	0.1908	0.2532	0.4651	0.1653
COD _{Mn}	9.175	10.375	0.925	6.120	17.910	19.940	0.810
COD _{Cr}	49.6	47.84	18.68	47.33	99.40	71.31	1.65
BOD ₅	7.13	14.24	2.33	9.26	17.58	6.68	0.51
NH ₄ -N	21.21	8.43	0.29	13.78	7.51	12.33	0.32
挥发酚	0.005	0.007	0.000	0.004	0.016	0.015	0.001
总砷	0.041	0.188	0.006	0.018	0.057	0.088	0.004
Cr ⁺⁶	0.023	0.030	0.012	0.018	0.040	0.034	0.017
网络输出	4.252	4.252	1.5581	4.252	4.252	4.252	1.3369
水质等级	V 级	V 级	II 级	V 级	V 级	V 级	II 级

下面采用径向基网络方法进行该市地表水质评价。

(1) 训练样本集、检测样本集及其期望目标的生成

① 训练样本集 为了解决仅用评价标准作为训练样本, 训练样本数过少和无法构建检测样本的问题, 在各级评价标准内按随机均匀分布方式线性插生成训练样本, 小于I级标准的生成500个, I、II级标准之间的生成500个, 其余以此类推, 共形成2500个训练样本。

② 测试样本集 用相同的方法生成检测样本, 小于I级标准生成100个, I、II级标准之间生成100个, 其余以此类推, 共形成500个检测样本。

③ 期望目标 小于I级标准的训练样本和检测样本的期望目标为0~1之间的数值, I、II级标准之间的训练样本和检测样本的期望目标为1~2之间的数值, II、III级标准之间的训练样本和检测样本的期望目标为2~3之间的数值, 其余以此类推。根据各生成样本的内插比例可计算出其期望目标值在各取值区间的对应值。据上述思路可以确定I、II、III、IV、V各级水的网络输出范围分别为: <1, 1~2, 2~3, 3~4, >4。

(2) 原始数据的预处理 试验两种预处理方案: 一种是将原始数据归一化到-1~1之间; 另一种是对原始数据进行预处理。

(3) 径向基网络的设计与应用效果

① 利用 MATLAB 6.15 构建径向基网络 RBF 网络输入层神经元数取决于水质评价的指标数，据题意确定为 8，输出层神经元数设定为 1，利用 MATLAB 6.15 中的 NEWRB 函数训练网络，自动确定所需隐层单元数。隐层单元激励函数为 RADBAS，加权函数为 DIST，输入函数为 NETPROD，输出层神经元的激励函数为纯线性函数 PURELIN，加权函数为 DOTPROD，输入函数为 NETSUM。

② 网络的应用效果 采用连续目标、归一化原始数据进行网络训练与测试，当训练次数等于 9 时，训练样本的均方误差为 0.0003，对于 2500 个训练样本与 500 个检测样本的错判率等于零。将该训练好的网络应用于 7 个待评点的评价，所得网络输出与评价结果见表 8.3。

8.1.5.3 RBF 网络在汽油干点软测量中的应用

软测量技术是工业过程分析、控制和优化的有力工具，所谓软测量就是根据可以检测的过程变量（如温度、流量、压力等）推断出某些难以检测或根本无法检测的工艺参数。建立软测量模型可以从两个方面入手：一种是通过分析工业过程的机理得到机理模型；另一种是根据反映过程运行的数据直接建立模型。由于机理方程推导和运算的复杂性，通常采用第二种方法。在这类方法中，基于前向神经网络建立软测量模型是比较有效的一种，与其他前向网络相比，RBF 神经网络不仅具有良好的泛化能力，而且计算量少，学习速度也比其他算法快得多。

(1) 混合模型设计 利用 SOFM 算法的自组织聚类特点以及 RBF 网络的非线性逼近能力，构造基于 SOFM 和 RBF 网络的混合网络模型。其中，SOFM 网络作为聚类网络，竞争层神经元以一维阵列形式排列；RBF 网络作为基础网络，采用单输出的网络结构。该模型通过 SOFM 网络对输入样本数进行粗分类，各分类中心对应的连接权值向量传递给 RBF 网络，作为 RBF 网络径向基函数中心；RBF 网络中隐层到输出层的连接权值采用有监督学习方法来确定。

混合网络的训练过程如下：首先对输入样本数据进行归一化处理，然后通过 SOFM 网络进行自组织分类，得到 M 种样本类别，样本类别个数 M 即为 RBF 神经网络径向基函数中心的个数。同时可以确定 RBF 网络隐层各个基函数的中心和宽度：第 j 个基函数的中心为 SOFM 网络的第 j 个聚类域中获胜神经元对应的权值向量，该基函数的宽度可按式(8.17)求得，也可令它们等于各自聚类中心与聚类域中训练样本间的平均距离，即：

$$\delta_j = \frac{1}{N_j} \sum_{\mathbf{x} \in U_j} (\mathbf{x} - \mathbf{c}_j) \quad (8.30)$$

(2) 仿真应用 汽油干点是反映炼油厂常压塔产品质量的一个重要参数。影响汽油干点的因素主要有塔顶温度、塔顶压力和塔顶循环回流温差。建立以上述 3 个工况参数为输入、以汽油干点为输出的基于 SOM 和 RBF 的混合网络模型。选择 160 组具有代表性的实际工况数据和对应的汽油干点化验值组成样本集，其中 80 组用于训练网络，其余 80 组用于网络泛化性能测试，测试结果如图 8.5 所示。可以看出，软测量模型的估计值与实际化验值能较好地吻合。

8.1.5.4 RBF 网络在地下温度预测中应用

由卫星热红外遥感数据反演的陆面温度、地表面温度与地表层 20cm 处的温度之间存在着较复杂的非线性关系，而地表层 20cm 以下各层的温度分布呈现出较好的规律性，因此地表层 20cm 处为温度分布规律的转折点。

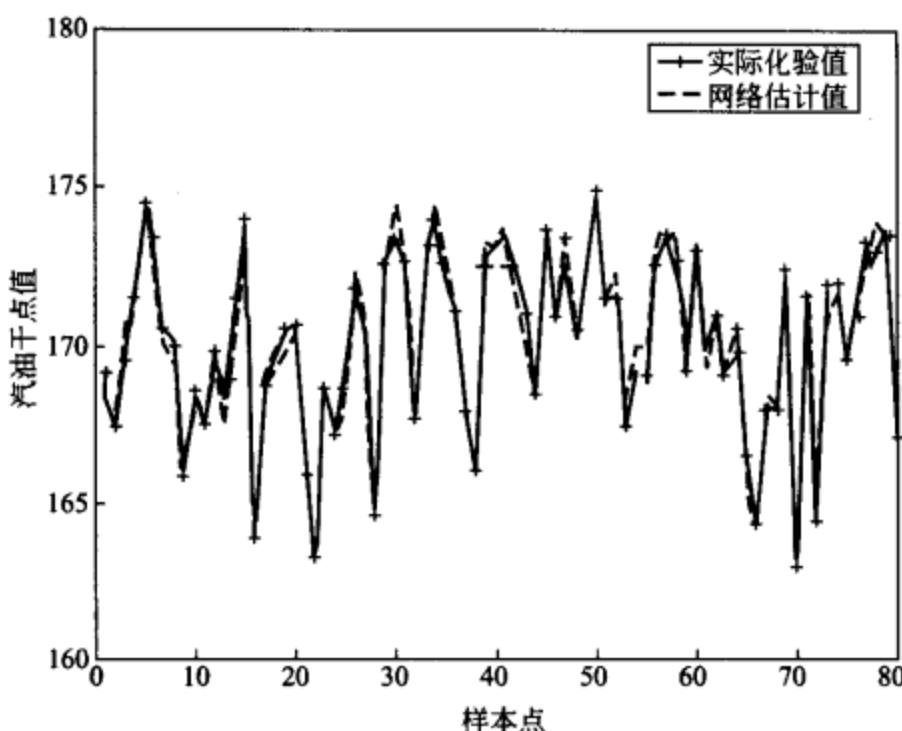


图 8.5 RBF 网络模型估计值与实际化验值的比较

通过对实测数据进行训练，可建立反映各陆面影响因素与地表层 20cm 处温度关系的神经网络模型，采用径向基函数网建模可获得满意的网络特性。利用径向基函数网络模型研究卫星热红外遥感反演陆面数据与地表层 20cm 温度 T_{-20} 的相关性，可合理地描述和解释各陆面因素对 T_{-20} 的影响。

(1) 地表层 20cm 深处温度与陆面影响因素的 RBF 网络模型 由于地表面处于多种热源叠加的环境中，如陆面气温的热传导、日光的热辐射以及地热的传导和红外辐射等。因此，影响地表面温度 (T_0) 的主要因素可包括：陆面气温 (T)、地表层 20cm 温度 (T_{-20})、地质状况 (g)、天气状况 (w)、测定时间 (t)、风速 (v)、高程 (h)、经纬度 (e, n) 等多种因素，可用下式表示为：

$$T_0 = G(T, T_{-20}, g, w, t, v, h, n, e)$$

为预测地表层 20cm 深处温度 T_{-20} ，通过上式可得到关于 T_{-20} 的函数关系如下：

$$T_{-20} = F(T, T_0, g, w, t, v, h, n, e)$$

上式表明，地表层 20cm 深处温度 T_{-20} 是高维空间向一维空间的非线性映射。但事实上，该式的解析表达无法给出，因此采用 RBF 网络对上式建模。

① 训练样本集设计 神经网络的输出为地表层 20cm 深处温度 T_{-20} ，输入为影响 T_{-20} 的各个因素。各影响因素的表示方法如下：a. 地质状况采用十进制；b. 天气状况分为晴、阴、雨三种，分别用 1、0、-1 表示；c. 测定时间采用二进制码；d. 高程编号规则为 2900~3000 编为 1，3000~3100 编为 2，3100~3200 编为 3，依次类推，直到 25；e. 其余影响因素直接用其测量值表示。

② 训练集与测试集 对 108 个钻孔数据组进行筛选，去除测量误差过大的奇异样本和相关信息（测定时间、天气状况、高程、气温、风速）不完整的样本。在 60 组数据完整的有效钻孔温度数据中，在保证覆盖各种地质状况的条件下选出包含输入、输出最值的数据构成训练样本集，数量占总数据的 3/4；其余 1/4 钻孔温度数据作为测试集样本。

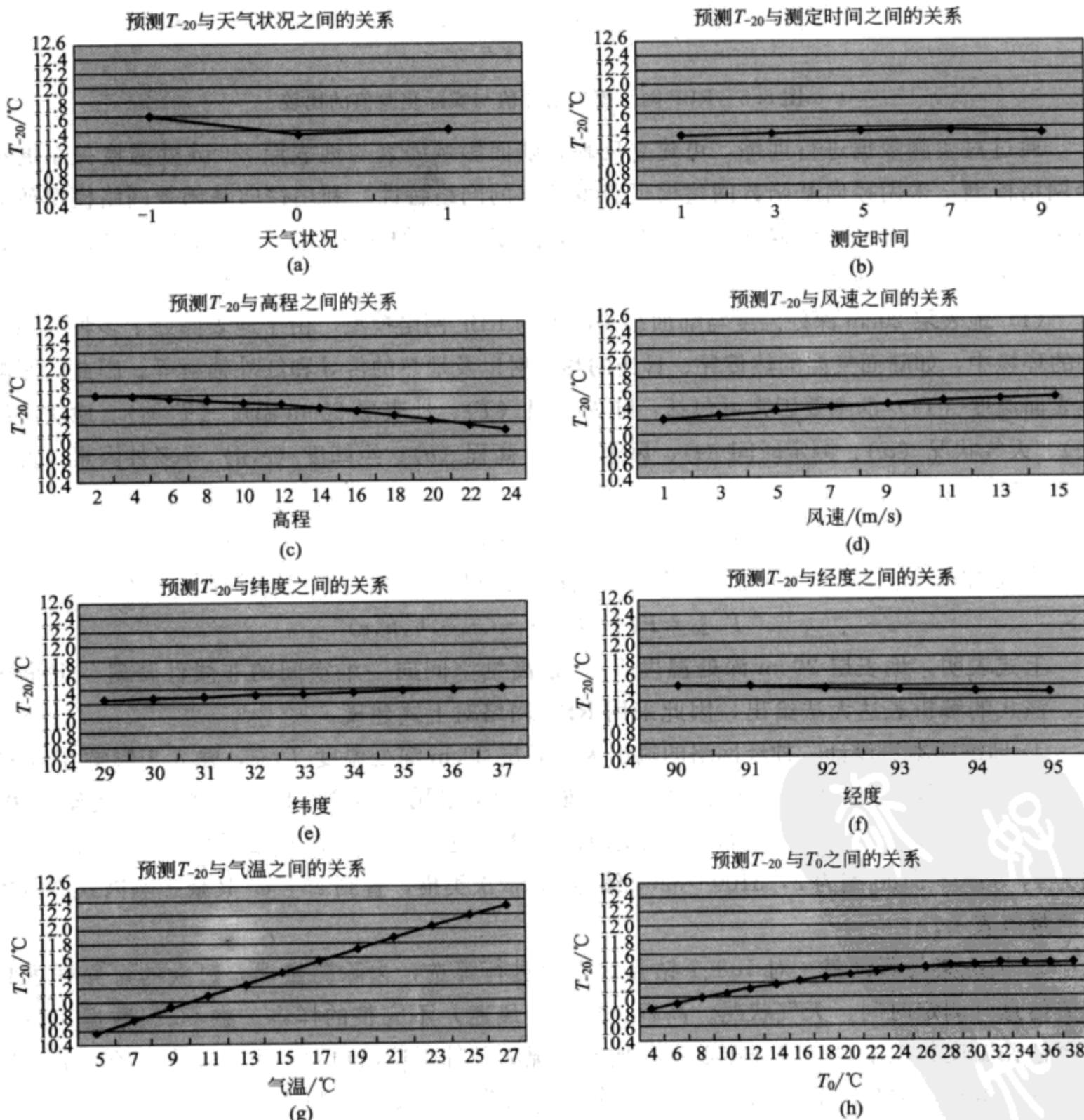
③ 基于 MATLAB 工具箱的网络设计 利用 MATLAB 的神经网络工具箱进行网络的设计和

表 8.4 网络训练误差与测试误差

序号	训练集均方误差		测试误差/℃
	℃	%	
1	0.10	0.56	1.90
2	0.50	2.79	1.70
3	0.8	4.47	1.72
4	1.00	5.59	1.20
5	1.20	6.70	1.30

表 8.4 列出 5 种训练误差与测试误差的情况，可以看出第 4 种情况最好。

(2) 基于 RBF 网络模型的陆面影响因素与地表层 20cm 深处温度的相关性分析 利用已经训练好的 RBF 网络对各影响因素进行分析。在保持其他影响因素数值为平均值的情况下，令待分析因素在适当的范围内等间隔取值，利用所建立的 RBF 网络模型对 T_{-20} 进行预测，从而可绘制一组 T_{-20} 预测值随各影响因素变化的曲线，如图 8.6 所示。

图 8.6 各陆面因素与 T_{-20} 预测值的相关性

训练，其中网络训练采用 NEWRB 函数，自动确定所需隐层单元数。隐层单元激励函数采用 RADBAS，加权函数采用 DIST，输入函数采用 NETPROD，输出层神经元激励函数采用纯线性函数 PURELIN，加权函数为 DOTPROD，输入函数为 NETSUM。

由图 8.6 可以看出，各单一因素对 T_{-20} 预测值的影响分别为：

- ① 天气状况 雨天使 T_{-20} 预测值较晴天升高，阴天使 T_{-20} 预测值较晴天降低。
- ② 测定时间 从上午 9:00~10:00（编号 1）后 T_{-20} 预测值随时间缓慢升高，午后 15:00~16:00（编号 7）时段 T_{-20} 预测值达到全天最高，其后逐渐降低。
- ③ 高程 同等陆面条件下高程值上升，则 T_{-20} 预测值降低。
- ④ 风速 同等陆面条件下风速值越大， T_{-20} 预测值越高。
- ⑤ 经度 在东经 90~95 度之间，随着该值上升， T_{-20} 预测值略有降低。
- ⑥ 纬度 在北纬 29~37 度之间，随着该值增大， T_{-20} 预测值略有升高。
- ⑦ 气温 气温越高， T_{-20} 预测值越高，是影响 T_{-20} 的主要因素之一。
- ⑧ 地表温度 T_0 在 T_0 较低的区段， T_{-20} 预测值随 T_0 升高而显著升高；在 T_0 较高的区段， T_{-20} 预测值随 T_0 升高而略有升高，也是影响 T_{-20} 的主要因素之一。

相关性分析结果表明，除了地质状况外，相关程度最大的 4 个因素依次为：气温 T 、地表面温度 T_0 、高程和风速。若仅以地质状况类型和上述 4 个因素为神经网络的输入，则网络的结构将得到进一步简化，当训练均方误差设为 0.8°C 时，测试均方误差 1.2348°C (6.90%)。

8.2 主分量分析

在许多数据处理的应用中，要求保存尽可能多的信息并得到较好的数据压缩。降低输入变量的维数对数据压缩十分必要，但降维不能简单地对 \mathbf{X} 进行截断，因为截断所带来的均方误差等于截掉得各分量方差之和。因此需要一种可逆的线性变换 \mathbf{T} ，使得通过该变换将原高维空间的数据 \mathbf{X} 投影为低维空间的数据 $\mathbf{T}(\mathbf{X})$ 后，对 $\mathbf{T}(\mathbf{X})$ 的截断在均方差意义下为最优，从而仍能保留原数据的主要信息。主分量分析 (principle components analysis, PCA) 方法能很好地满足这一要求。

8.2.1 主分量分析方法概述

主分量分析是 Karhunen 于 1947 年提出的，Loeve 于 1963 年对其进行了归纳总结，因此 PCA 又称为 K-L 变换。主分量分析是分析一个随机向量过程相关结构的十分有用的统计技术，并已经广泛地应用于现代信号处理的许多领域，如高分辨谱估计、系统辨识、数据压缩、特征提取、模式识别、数字通信、计算机视觉等。

主分量分析包括特征选择和特征提取过程。特征选择过程通过一种可逆变换 \mathbf{T} 将从数据空间映射到同维特征空间，从而获得输入的特征，即输入的主分量；而特征提取过程的目的是降维，即对变换后的特征空间向量进行截断，选取主要特征分量而舍去其他特征分量。主分量分析提供的可逆变换 \mathbf{T} 能够保证对 $\mathbf{T}(\mathbf{X})$ 的截断在均方差意义下为最优。

8.2.1.1 特征向量的选择

特征选择过程的关键是选取特征向量并获得输入向量在特征向量上的投影。令 \mathbf{X} 表示 n 维随机向量，不失一般性，假设其均值：

$$E(\mathbf{X}) = 0$$

若 \mathbf{X} 均值不为零，可令 $\mathbf{X}' = \mathbf{X} - E(\mathbf{X})$ ，从而得到 $E(\mathbf{X}') = 0$

令 \mathbf{U}_j 表示一个 n 维单位向量， \mathbf{X} 在 \mathbf{U}_j 上的投影为：

$$y_j = \mathbf{U}_j^T \mathbf{X}$$

因此 y_j 也是均值为零的随机变量，其方差为：

$$E(y_j^2) = E[(\mathbf{U}_j^T \mathbf{X})(\mathbf{U}_j^T \mathbf{X})] = \mathbf{U}_j^T E[(\mathbf{X} \mathbf{X}^T)] \mathbf{U}_j = \mathbf{U}_j^T \mathbf{R}_{XX} \mathbf{U}_j \quad (8.31)$$

式中， \mathbf{R}_{XX} 为 \mathbf{X} 的自相关阵，由于 $E(\mathbf{X}) = 0$ ， \mathbf{R}_{XX} 也为协方差阵。

可以看出，投影 y_j 的方差是单位向量 \mathbf{U}_j 的函数，当 \mathbf{U}_j 改变方向时，投影 y_j 的方差也随之改变。若希望找到一个方向，使得投影 y_j 的方差达到最大，理论证明 \mathbf{U}_j 应满足以下条件：

$$\mathbf{R}_{XX} \mathbf{U}_j = \lambda \mathbf{U}_j$$

以上是矩阵 \mathbf{R}_{XX} 的特征值方程，因此 \mathbf{U}_j 是 \mathbf{R}_{XX} 的特征向量。 \mathbf{R}_{XX} 是一个 $n \times n$ 实对称阵，具有 n 个非负实数特征值，对应的 n 个单位特征向量可用下式计算：

$$\mathbf{R}_{XX} \mathbf{U}_i = \lambda_i \mathbf{U}_i \quad i=1, 2, \dots, n \quad (8.32)$$

对应于不同特征值的特征向量是两两正交的，从而构成一个 n 维特征空间。 \mathbf{X} 在 n 个正交特征向量 \mathbf{U}_i ， $i=1, 2, \dots, n$ 上的投影构成特征空间中的向量 \mathbf{Y} ，表示为：

$$\mathbf{Y} = \mathbf{U}^T \mathbf{X} \quad (8.33)$$

其中特征向量矩阵 $\mathbf{U} = [\mathbf{U}_1, \mathbf{U}_2, \dots, \mathbf{U}_n]$ ， $\mathbf{Y} = [y_1, y_2, \dots, y_n]^T$ ， \mathbf{Y} 的第 i 个分量 y_i 为输入 \mathbf{X} 的第 i 个主分量。

式(8.33) 表明，将输入空间向量 \mathbf{X} 映射为特征空间向量 \mathbf{Y} 的线性变换正是特征向量矩阵 \mathbf{U} ，由于 $\mathbf{U}\mathbf{U}^T = \mathbf{E}$ ，故通过下面的逆变换可以重构 \mathbf{X} ：

$$\mathbf{X} = \mathbf{UY} = \sum_{i=1}^n \mathbf{U}_i y_i \quad (8.34)$$

输入向量 \mathbf{X} 可表示为特征向量的线性组合，组合系数是 \mathbf{X} 在各特征向量上进行投影而获得的各主分量。

8.2.1.2 降维处理

在上述特征选择过程中已获得输入 \mathbf{X} 的全部主分量，特征提取过程中需要提取主要特征而截断次要特征，以达到降维的目的。

将式(8.31) 写为 \mathbf{Y} 的自相关阵：

$$\mathbf{R}_{YY} = E(\mathbf{YY}^T) = E[(\mathbf{U}^T \mathbf{X})(\mathbf{U}^T \mathbf{X})] = \mathbf{U}^T E[(\mathbf{X} \mathbf{X}^T)] \mathbf{U} = \mathbf{U}^T \mathbf{R}_{XX} \mathbf{U} \quad (8.35)$$

由于 $E(\mathbf{Y}) = E(\mathbf{U}^T \mathbf{X}) = \mathbf{U}^T E(\mathbf{X}) = 0$ ， \mathbf{Y} 的自相关阵也是协方差阵，且有：

$$\mathbf{R}_{YY} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \quad (8.36)$$

因此 \mathbf{R}_{YY} 取决于 \mathbf{R}_{XX} 。

为保证对 \mathbf{Y} 的截断是在均方误差意义下最优，将特征向量对应的特征值从大到小排序，即 $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ 。从式(8.36) 可以看出，在重构输入向量 \mathbf{X} 时，特征值越大，所对应的特征向量的贡献也越大。因此当考虑将 n 维数据降为 m 维 ($1 \leq m < n$) 时，只需保留前 m 个大特征值而舍掉后面的 $n-m$ 个小特征值。此时重构 \mathbf{X} 的估计值为：

$$\hat{\mathbf{X}} = \sum_{i=m+1}^n \mathbf{U}_i y_i \quad (8.37)$$

由式(8.31), 原始输入向量的 n 个分量的总方差为:

$$\sum_{i=1}^n E(y_i^2) = \sum_{i=1}^n \mathbf{U}_i^T \mathbf{R}_{XX} \mathbf{U}_i = \sum_{i=1}^n \lambda_i \quad (8.38)$$

而变换后的向量 $\hat{\mathbf{X}}$ 的前 m 个分量的方差为:

$$\sum_{i=1}^m E(y_i^2) = \sum_{i=1}^m \mathbf{U}_i^T \mathbf{R}_{XX} \mathbf{U}_i = \sum_{i=1}^m \lambda_i \quad (8.39)$$

因此截断 $n-m$ 个小特征值带来的均方误差为:

$$E[(\mathbf{X} - \hat{\mathbf{X}})^2] = \sum_{i=m+1}^n \lambda_i \quad (8.40)$$

前 m 个分量的方差贡献率定义为:

$$\phi(m) = \frac{\sum_{i=1}^m \lambda_i}{\sum_{i=1}^n \lambda_i} \times 100 \% \quad (8.41)$$

满足给定方差贡献率时, 即可将前 m 个特征向量构成的空间作为降维后的低维投影空间。

从上述结果可以得出主分量分析方法的步骤如下:

- ① 计算输入向量的自相关矩阵 \mathbf{R}_{XX} 的特征值和特征向量;
- ② 将特征向量归一化, 将特征值从大到小重新排序;
- ③ 将原始输入向量投影到前 m 个特征值对应的特征向量构成的子空间, 得到 $\hat{\mathbf{X}} = [\hat{x}_1, \hat{x}_2, \dots, \hat{x}_m]$, 其中第一个分量具有的方差最大, 其余依次减小。

为了说明主分量分析的几何意义, 考虑图 8.7 所示二维数据集的例子。图中 x_1 轴和 x_2 轴构成原始数据空间, 标号为 1 和 2 的旋转坐标轴构成该数据集主分量分析产生的特征空间。数据集投影到 1 号轴的方差比投影到其他任何方向时都大, 因此 1 号轴代表的是主分量方向。可以看出, 数据集在 1 号轴方向的投影具有双峰的特点, 抓住了数据集有两个聚类的主要特征, 而在 2 号轴方向的投影隐藏了数据集内在的双峰特征。对于更一般的高维数据集来说, 其固有的聚类结构一般无法看出, 因此需要进行类似于主分量分析的统计分析。

上述主分量分析的实质是随机向量的正交归一变换: 将 n 维向量 \mathbf{X} 映射为 n 维向量 \mathbf{Y} , 虽然维数未变, 但是其协方差阵 \mathbf{R}_{YY} 变成了对角阵, 表示其各维独立。按照 \mathbf{X} 协方差阵的特征值从大到小排序, 特征值越小, 特征空间中对应方向上分布的信息就越少, 可作为次要分量抛弃, 剩下的 m 维即用于重构 $\hat{\mathbf{X}}$ 。

例 8.1 采用主分量分析方法对下面 4 个输入向量进行分析。

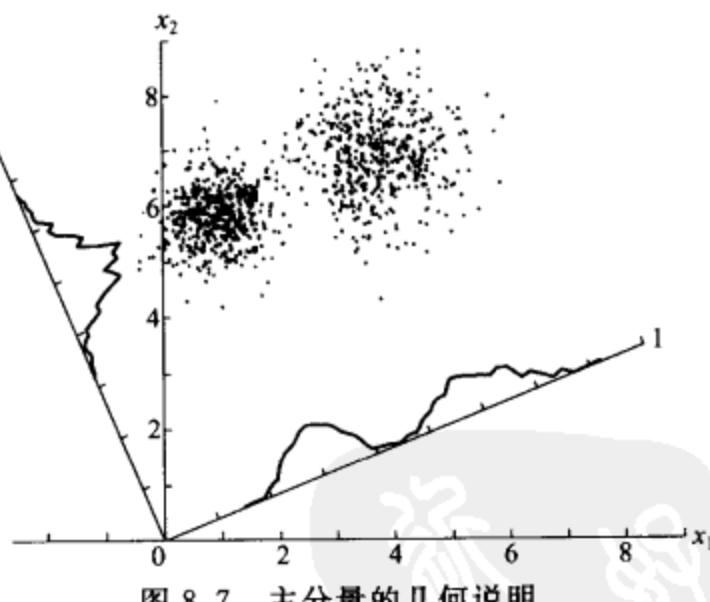


图 8.7 主分量的几何说明

$$\mathbf{X}(1) = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{X}(2) = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}, \quad \mathbf{X}(3) = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{X}(4) = \begin{bmatrix} 1 \\ 4 \\ 1 \end{bmatrix}$$

解 首先检验 \mathbf{X} 是否满足均值为零的条件, 计算:

$$E[\mathbf{X}] = \frac{1}{4} \begin{bmatrix} 1+2+0+1 \\ 0+3+1+4 \\ 1+1+1+1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

为满足均值为零的条件, 将输入向量转换为 $\mathbf{X}' = \mathbf{X} - E[\mathbf{X}]$, 有:

$$\mathbf{X}'(1) = \begin{bmatrix} 0 \\ -2 \\ 0 \end{bmatrix}, \quad \mathbf{X}'(2) = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{X}'(3) = \begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix}, \quad \mathbf{X}'(4) = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix}$$

第二步, 求 \mathbf{X} 的协方差矩阵和特征值。

$$\begin{aligned} \mathbf{R}_{\mathbf{XX}} &= \mathbf{R}_{\mathbf{X}'\mathbf{X}'} = E(\mathbf{X}'\mathbf{X}'^T) \\ &= \frac{1}{4} [\mathbf{X}'(1)\mathbf{X}'(1)^T + \mathbf{X}'(2)\mathbf{X}'(2)^T + \mathbf{X}'(3)\mathbf{X}'(3)^T + \mathbf{X}'(4)\mathbf{X}'(4)^T] \\ &= \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 2.5 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

求出 $\mathbf{R}_{\mathbf{XX}}$ 的 3 个特征值并从大到小排序: $\lambda_1 = 2.618$, $\lambda_2 = 0.382$, $\lambda_3 = 0$, 对应的特征向量为:

$$\mathbf{U}_1 = \begin{bmatrix} 0.2298 \\ 0.9732 \\ 0 \end{bmatrix}, \quad \mathbf{U}_2 = \begin{bmatrix} -0.9732 \\ 0.2298 \\ 0 \end{bmatrix}, \quad \mathbf{U}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

第三步, 降维处理。可以看出, 第一个最大特征值的贡献率为 87.27%, 若将各输入向量降为一维, 可保留原模式 87.27% 的能量; 第三个特征值的贡献率为 0, 若将各输入向量降为二维, 原模式将不损失任何信息。

将输入向量压缩到一维, 各输入向量的第一个主分量用式 $y_1 = \mathbf{U}_1^T \mathbf{X}'$ 计算, 得到:

$$y_1(1) = -1.9465, \quad y_1(2) = 1.2030, \quad y_1(3) = -1.2030, \quad y_1(4) = 1.9465$$

用第一主分量重构各输入向量 $\hat{\mathbf{X}'}$, 结果用式 $\hat{\mathbf{X}'} = \mathbf{U}_1 y_1$ 计算, 得到:

$$\hat{\mathbf{X}'}(1) = \begin{bmatrix} -0.4472 \\ -1.8944 \\ 0 \end{bmatrix}, \quad \hat{\mathbf{X}'}(2) = \begin{bmatrix} 0.2764 \\ 1.1708 \\ 0 \end{bmatrix}, \quad \hat{\mathbf{X}'}(3) = \begin{bmatrix} -0.2764 \\ -1.1708 \\ 0 \end{bmatrix}, \quad \hat{\mathbf{X}'}(4) = \begin{bmatrix} 0.4472 \\ 1.8944 \\ 0 \end{bmatrix}$$

重构 \mathbf{X}' 带来的均方误差为:

$$e_1 = E[(\mathbf{X}' - \hat{\mathbf{X}}')^2] = 0.382 = \lambda_2$$

将输入向量压缩到二维, 各输入向量的第二个主分量用式 $y_2 = \mathbf{U}_2^T \mathbf{X}'$ 计算, 得到:

$$y_2(1) = -0.4595, \quad y_2(2) = -0.7435, \quad y_2(3) = 0.7435, \quad y_2(4) = 0.4595$$

用前两个主分量重构各输入向量 $\hat{\mathbf{X}'}$, 结果用式 $\hat{\mathbf{X}'} = \mathbf{U}_1 y_1 + \mathbf{U}_2 y_2$ 计算, 得到:

$$\hat{\mathbf{X}'}(1) = \begin{bmatrix} 0 \\ -2.0000 \\ 0 \end{bmatrix}, \quad \hat{\mathbf{X}'}(2) = \begin{bmatrix} 1.0000 \\ 1.0000 \\ 0 \end{bmatrix}, \quad \hat{\mathbf{X}'}(3) = \begin{bmatrix} -1.0000 \\ -1.0000 \\ 0 \end{bmatrix}, \quad \hat{\mathbf{X}'}(4) = \begin{bmatrix} 0 \\ 2.0000 \\ 0 \end{bmatrix}$$

重构 \mathbf{X}' 带来的均方误差为：

$$e_2 = E[(\mathbf{X}' - \hat{\mathbf{X}}')^2] = 0 = \lambda_3$$

8.2.2 前向 PCA 神经网络及学习算法

可以看出，当原始维数 n 较大时，直接计算 \mathbf{R}_{XX} 的特征值很困难。如果利用神经网络的学习能力，可通过训练逐步进行主分量分析。训练后的网络权值作为 \mathbf{R}_{XX} 的特征向量，网络输出作为输入 \mathbf{X} 在低维空间各方向上的投影。下面介绍两种前向 PCA 网络模型及其算法。

8.2.2.1 单神经元 PCA 模型及 Oja 算法

单神经元 PCA 模型如图 8.8 所示，具有 n 输入-单输出结构。其输出为：

$$y = \mathbf{W}^T \mathbf{X} = \sum_{i=1}^n w_i x_i \quad (8.42)$$

理论上已证明，如果采用 Hebb 学习规则，将得到方差最大的输出，对应于第一个主分量，因此权向量 \mathbf{W} 正是与 \mathbf{R}_{XX} 的最大特征值对应的特征向量 \mathbf{U}_1 。

由 2.4.1 知，简单的 Hebb 规则会导致学习过程发散。E. Oja 于 1982 年提出了基于 Hebb 规则的 Oja 规则如下：

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \eta[y(t)\mathbf{X}(t) - y^2(t)\mathbf{W}(t)] = \mathbf{W}(t) + \eta y(t)[\mathbf{X}(t) - y(t)\mathbf{W}(t)] \quad (8.43)$$

式中，学习率 $\eta \in (0, 1)$ 。

基于 Oja 学习算法的线性神经元 PCA 模型相当于一个最大特征滤波器，它将以概率 1 收敛于一个固定点，其特征是：当 $t \rightarrow \infty$ 时，模型输出的方差趋向于 \mathbf{R}_{XX} 的最大特征值 λ_1 ；模型的权向量趋向相应的特征向量 \mathbf{U}_1 ，且有 $\lim_{t \rightarrow \infty} \|\mathbf{W}(t)\| = 1$ 。

采用 Oja 学习算法的训练步骤如下：

- ① 初始化网络权值为小随机数，设置网络收敛的阈值 $\epsilon > 0$ ，学习率 $\eta \in (0, 1)$ ；
- ② 输入一个训练样本 $\mathbf{X}(t)$ ，按式(8.42) 计算网络输出 $y(t)$ ；
- ③ 根据式(8.43) 计算权值修正量 $\Delta \mathbf{W} = \mathbf{W}(t+1) - \mathbf{W}(t)$ ；
- ④ 若 $\|\Delta \mathbf{W}\| < \epsilon$ ，训练结束，否则转到 (2) 继续训练。

例 8.2 采用例 8.1 中的数据，用 Oja 算法提取第一主分量，为满足均值为零的要求，用 \mathbf{X}' 作为训练数据。

解 将权值初始化为 $-1 \sim 1$ 之间的小随机数。学习率的选择对 PCA 网络的收敛有较大影响，应取较小的值，使训练缓慢进行，本例取 $\eta = 0.05$ 。判断网络训练收敛的参数定义为由主分量重建的输入向量矩阵与原始输入向量矩阵之差的范数，即 $\epsilon = \|[\hat{\mathbf{X}}'(1), \hat{\mathbf{X}}'(2), \hat{\mathbf{X}}'(3), \hat{\mathbf{X}}'(4)] - [\mathbf{X}'(1), \mathbf{X}'(2), \mathbf{X}'(3), \mathbf{X}'(4)]\|$ ，网络训练 8 步后， ϵ 稳定在 1.2361 左右。

用 Oja 学习算法的训练 25 步后，权值向量为：

$$\mathbf{W} = [0.2298, 0.9732, 0.0000]^T$$

与例 8.1 中的统计 PCA 方法计算出来的第一主分量对应的特征向量 \mathbf{U}_1 完全一致。

网络训练后对于 4 个输入向量计算的输出分别为：

$$y(1) = -1.9465, y(2) = 1.2030, y(3) = -1.2030, y(4) = 1.9465$$

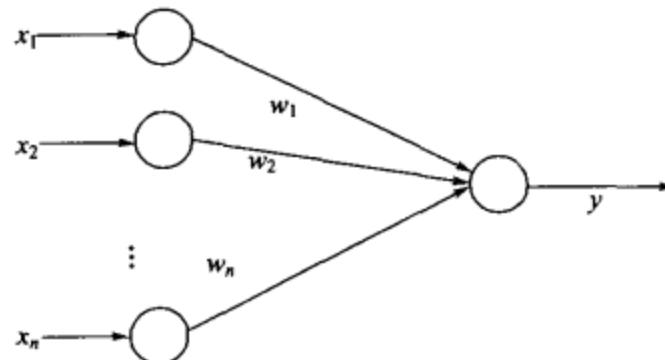


图 8.8 单神经元 PCA 模型

与例 8.1 计算出来的第一主分量值相同。

8.2.2.2 单层 PCA 模型及 Sanger 算法

T. D. Sanger 于 1989 年提出一种可以任选 m 个主分量 ($m \leq n$) 的 PCA 模型，该网络模型将单神经元的学习扩展到图 8.9 所示的单层网络，输出层各节点均采用线性转移函数

$$y_j(t) = \mathbf{W}_j^T(t) \mathbf{X}(t)$$

$$= \sum_{i=1}^n w_{ij}(t)x_i(t) \quad j = 1, 2, \dots, m \quad (8.44)$$

Sanger 提出的权值调整规则为：

$$\begin{aligned} \mathbf{W}_j(t+1) &= \mathbf{W}_j(t) + \eta [y_j(t)\hat{\mathbf{X}}(t) - y_j^2(t)\mathbf{W}_j(t)] \\ &= \mathbf{W}_j(t) + \eta y_j(t)[\hat{\mathbf{X}}(t) - y_j(t)\mathbf{W}_j(t)] \quad j = 1, 2, \dots, m \end{aligned} \quad (8.45)$$

可以看出，Sanger 算法与 Oja 算法的权值调整公式形式上完全一致，不同之处是用 $\hat{\mathbf{X}}(t)$ 代替了 $\mathbf{X}(t)$ ，其中：

$$\hat{\mathbf{X}}(t) = \mathbf{X}(t) - \sum_{i=1}^{j-1} y_i(t)\mathbf{W}_i(t) \quad (8.46)$$

式中的求和项是为了使各权向量正交化，以满足特征向量正交的要求。下面对通过各输出神经元逐个展示求和项的作用进行分析：

① 对输出层第一个神经元来说， $j=1$ ， $\hat{\mathbf{X}}(t)=\mathbf{X}(t)$ ，式(8.45)与式(8.43)相同，相当于单神经元的情况，因此第一个神经元的输出 y_1 就是最大主分量。

② 对第二个神经元， $j=2$ ， $\hat{\mathbf{X}}(t)=\mathbf{X}(t)-y_1(t)\mathbf{W}_1(t)$ ，如果第一个神经元已收敛于第一个主分量，则第二个神经元得到的输入向量 $\hat{\mathbf{X}}$ 是已经除去第一个主分量之后的结果，抽取 $\hat{\mathbf{X}}$ 的最大主分量等效于原始输入 \mathbf{X} 的第二大主分量。

③ 对第三个神经元， $j=3$ ， $\hat{\mathbf{X}}(t)=\mathbf{X}(t)-y_1(t)\mathbf{W}_1(t)-y_2(t)\mathbf{W}_2(t)$ ，因此第三个神经元得到的输入向量 $\hat{\mathbf{X}}(t)$ 是已经除去第一个和第二个主分量之后的结果，抽取 $\hat{\mathbf{X}}(t)$ 的最大主分量等效于原始输入 $\mathbf{X}(t)$ 的第三大主分量。

依此类推，第 j 个神经元的输出 y_j 就是输入 $\mathbf{X}(t)$ 的第 j 大主分量。事实上，网络的各神经元是并行工作的，上述分析只是为了便于理解。

采用 Sanger 学习算法的训练步骤如下：

- ① 初始化网络权值为小随机数，设置网络收敛的阈值 $\epsilon > 0$ ，学习率 $\eta \in (0, 1)$ ；
- ② 输入一个训练样本 $\mathbf{X}(t)$ ，按式(8.44)计算网络各节点输出 $y_j(t)$ ；
- ③ 根据式(8.45)和式(8.46)计算权值修正量 $\Delta \mathbf{W} = \mathbf{W}(t+1) - \mathbf{W}(t)$ ；
- ④ 若 $\|\Delta \mathbf{W}\| < \epsilon$ ，训练结束，否则转到②继续训练。

通过上述训练，网络收敛后其权值矩阵的各列对应于 \mathbf{R}_{XX} 的前 m 个特征值对应的特征向量，网络的输出对应于输入向量 \mathbf{X} 在这些特征向量方向上的投影，即 \mathbf{X} 的前 m 个主分量，从而实现了对输入数据的主分量提取。

例 8.3 网络参数为 $n=3$, $m=2$ ，采用例 8.1 中的数据，为满足均值为零的要求，用 \mathbf{X}' 作为训练数据。

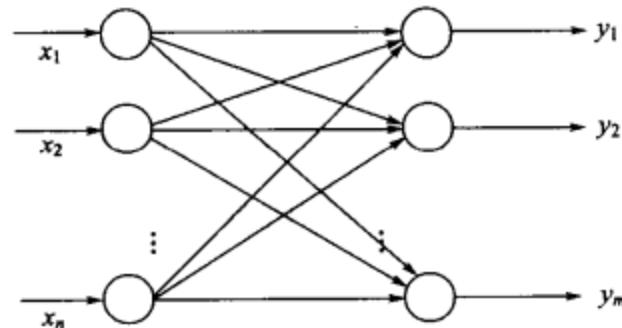


图 8.9 单层 PCA 网络模型

解 将权值初始化为 $-1\sim 1$ 的小随机数；取 $\eta=0.05$ 。判断网络训练收敛的参数定义与例8.2中相同，网络采用Sanger学习算法训练100步以后， ϵ 接近于0，表明前两个主分量的方差贡献率已足够大。

训练结束后，网络的权值矩阵为：

$$\mathbf{W} = \begin{bmatrix} 0.2298 & -0.9732 \\ 0.9732 & 0.2299 \\ 0.0000 & -0.0000 \end{bmatrix}$$

其中两个权值列向量与前2个主分量方向上的特征向量相同。

网络训练后的输出用 $\mathbf{Y}(t)=\mathbf{W}^T \mathbf{X}'(t)$ 计算为：

$$\mathbf{Y}(1) = \begin{bmatrix} -1.9465 \\ -0.4597 \end{bmatrix}, \quad \mathbf{Y}(2) = \begin{bmatrix} 1.2030 \\ -0.7434 \end{bmatrix}, \quad \mathbf{Y}(3) = \begin{bmatrix} -1.2030 \\ 0.7434 \end{bmatrix}, \quad \mathbf{Y}(4) = \begin{bmatrix} 1.9465 \\ 0.4597 \end{bmatrix}$$

8.2.3 侧向连接自适应PCA神经网络及APEX算法

S. Y. Kung于1990年提出一种具有侧向连接的自适应PCA网络模型及算法，称为APEX(Adaptive Principle Components Extraction)算法。

APEX网络的特点是，若给出前 j 个主分量，可用递推方式计算出第 $j+1$ 个主分量，它与前面 j 个主分量均正交。

APEX网络的结构如图8.10所示，同前面相同的是输出层每个神经元都是线性单元，不同的是网络中有两种连接：一种是由输出层到输出层的前向连接；一种是在输出层从神经元 $1, 2, \dots, j-1$ 到第 j 个神经元间的侧向连接。

APEX网络的学习是依次进行的，图中的粗线表示神经元 j 的两种连接权。前向连接权向量表示为：

$$\mathbf{W}_j(t) = [w_{1j}(t), w_{2j}(t), \dots, w_{nj}(t)]^T$$

按照Hebb学习规则进行训练。侧向连接的权向量表示为：

$$\mathbf{A}_j(t) = [a_{1j}(t), a_{2j}(t), \dots, a_{j-1,j}(t)]^T$$

按照反Hebb学习规则进行训练。

神经元 j 的输出为：

$$y_j(t) = \mathbf{W}_j^T(t) \mathbf{X}(t) + \mathbf{A}_j^T(t) \mathbf{Y}_{j-1}(t) \quad (8.47)$$

输出表达式的第一项由前向连接确定，第二项由侧向连接确定。其中，反馈信号向量 $\mathbf{Y}_{j-1}(t)$ 由前 $j-1$ 个神经元的输出定义：

$$\mathbf{Y}_{j-1}(t) = [y_1(t), y_2(t), \dots, y_{j-1}(t)]^T \quad (8.48)$$

假设图8.10中网络的前 $j-1$ 个神经元权向量已经收敛到以下稳定条件：

$$\mathbf{W}_k = \mathbf{U}_k, \quad k = 1, 2, \dots, j-1 \quad (8.49)$$

$$\mathbf{A}_k = 0, \quad k = 1, 2, \dots, j-1 \quad (8.50)$$

式中， \mathbf{U}_k 是与 \mathbf{R}_{XX} 的第 k 个特征值对应的特征向量。利用式(8.47)~式(8.50)，前 $j-1$ 个神经元的输出可以写成：

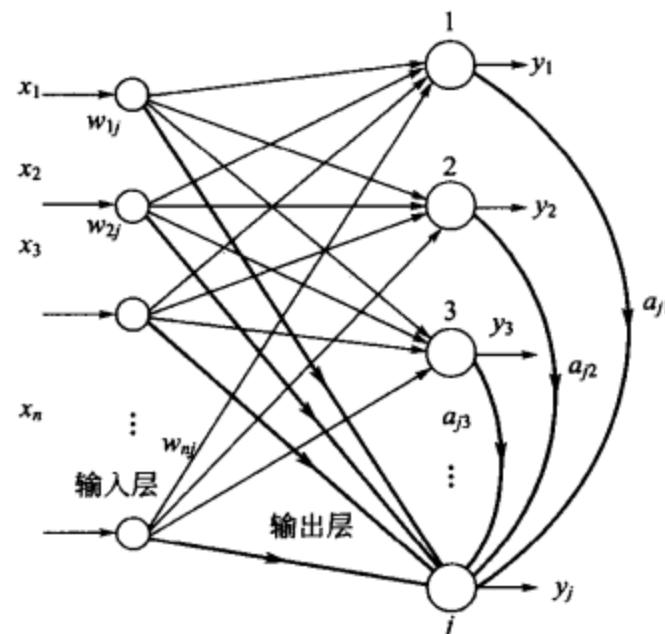


图8.10 有侧向连接的PCA网络模型

$$\mathbf{Y}_{j-1}(t) = [\mathbf{U}_1^T \mathbf{X}(t), \mathbf{U}_2^T \mathbf{X}(t), \dots, \mathbf{U}_{j-1}^T \mathbf{X}(t)]^T = \mathbf{U} \mathbf{X}(t) \quad (8.51)$$

其中

$$\mathbf{U} = [\mathbf{U}_1, \mathbf{U}_2, \dots, \mathbf{U}_{j-1}]^T$$

下面针对第 j 个神经元，给出其权值修正公式如下：

$$\mathbf{W}_j(t+1) = \mathbf{W}_j(t) + \eta [y_j(t) \mathbf{X}(t) - y_j^2(t) \mathbf{W}_j(t)] \quad (8.52)$$

$$\mathbf{A}_j(t+1) = \mathbf{A}_j(t) - \beta [y_j \mathbf{Y}_{j-1} + y_j^2 \mathbf{A}_j(t)] \quad (8.53)$$

可以看出，式(8.52)为 Oja 学习算法，方括号中的第一项代表 Hebb 学习规则，第二项保证算法的稳定性；式(8.53)方括号中的第一项代表反 Hebb 学习规则，第二项保证算法的稳定性。该算法的特点是前向连接权是激励性的，按照 Hebb 学习规则，从而起到自增强作用；侧向连接按照反 Hebb 学习规则，从而起到抑制作用。

理论上已经证明，在上述权值修正规则的作用下，将收敛于 \mathbf{X} 的自相关矩阵 \mathbf{R}_{XX} 的第 j 个特征值对应的特征向量 \mathbf{U}_j 。

APEX 学习算法的具体步骤如下：

(1) 初始化前向权向量 \mathbf{W}_j 和侧向权向量 \mathbf{A}_j 为小随机数， $j=1, 2, \dots, m$ ，设置网络收敛的阈值 $\epsilon > 0$ ，学习率 $\eta, \beta \in (0, 1)$ 。

(2) 置 $j=1$ ，执行 Oja 学习算法。对 $t=1, 2, \dots$ ，计算：

$$y_1(t) = \mathbf{W}_1^T(t) \mathbf{X}(t)$$

$$\mathbf{W}_1(t+1) = \mathbf{W}_1(t) + \eta [y_1(t) \mathbf{X}(t) - y_1^2(t) \mathbf{W}_1(t)]$$

直到 $\|\Delta \mathbf{W}_1\| < \epsilon$ ，置 $j=2$ 。

(3) 对 $t=1, 2, \dots$ ，分别按式(8.48)、式(8.47)、式(8.52)和式(8.53)计算 \mathbf{Y}_{j-1} 、 y_j 、 $\mathbf{W}_j(t+1)$ 和 $\mathbf{A}_j(t+1)$ ，直到 $\|\Delta \mathbf{W}_j\| < \epsilon$ ， $\|\Delta \mathbf{A}_j\| < \epsilon$ 。

(4) j 增加 1，返回第 (3) 步，直到 $j=m$ ，其中 m 为期望的主分量数。

当 t 很大时， $\|\Delta \mathbf{W}_j\| < \epsilon$ ， $\|\Delta \mathbf{A}_j\| < \epsilon$ ，可认为训练收敛，此时有 $\mathbf{W}_j(t) \mathbf{U}_j, \mathbf{A}_j(t) \rightarrow 0$ 。

8.3 支持向量机

BP 网络和 RBF 网络都擅长解决模式分类与非线性映射问题。由 Vapnik 首先提出的支持向量机 (support vector machine, SVM)，是本书介绍的第三种通用前馈神经网络，同样可用于解决模式分类与非线性映射问题。

从线性可分模式分类的角度看，支持向量机的主要思想是建立一个最优决策超平面，使得该平面两侧距平面最近的两类样本之间的距离最大化，从而对分类问题提供良好的泛化能力。对于非线性可分模式分类问题，根据 Cover 定理：将复杂的模式分类问题非线性地投射到高维特征空间可能是线性可分的，因此只要变换是非线性的且特征空间的维数足够高，则原始模式空间能变换为一个新的高维特征空间，使得在特征空间中模式以较高的概率为线性可分的。此时，应用支持向量机在特征空间建立分类超平面，即可解决非线性可分的模式识别问题。

与前面各章讨论的神经网络均基于某种生物学原理的情况不同，支持向量机基于统计学习理论的原理性方法，因此需要较深的数学基础。

8.3.1 支持向量机的基本思想

第 3 章的单层感知器对于线性可分数据的二值分类机理可理解为，系统随机产生一个超

平面并移动它，直到训练集中属于不同类别的样本点正好位于该超平面的两侧。显然，这种机理能够解决线性分类问题，但不能够保证产生的超平面是最优的。支持向量机建立的分类超平面能够在保证分类精度的同时，使超平面两侧的空白区域最大化，从而实现对线性可分问题的最优分类。下面讨论线性可分情况下支持向量机的分类原理。

8.3.1.1 最优超平面的概念

考虑 P 个线性可分样本 $\{(X^1, d^1), (X^2, d^2), \dots, (X^p, d^p), \dots, (X^P, d^P)\}$ ，对于任一输入样本 X^p ，其期望输出为 $d^p = \pm 1$ ，分别代表两类的类别标识。用于分类的超平面方程为：

$$W^T X + b = 0 \quad (8.54)$$

式中， X 为输入向量； W 为权值向量； b 为偏置，相当于前几章中的负阈值 ($b = -T$)，则有：

$$\begin{aligned} W^T X^p + b &> 0, & \text{当 } d^p = +1 \\ W^T X^p + b &< 0, & \text{当 } d^p = -1 \end{aligned}$$

由式(8.54) 定义的超平面与最近的样本点之间的间隔称为分离边缘，用 ρ 表示。支持向量机的目标是找到一个使分离边缘最大的超平面，即最优超平面。图 8.11 给出二维平面中最优超平面的示意图。可以看出，最优超平面能提供两类之间最大可能的分离，因此确定最优超平面的权值 W_0 和偏置 b_0 应是唯一的。在式(8.54) 定义的一簇超平面中，最优超平面的方程应为：

$$W_0^T X + b_0 = 0 \quad (8.55)$$

由解析几何知识可得样本空间任一点到最优超平面的距离为：

$$r = \frac{W_0^T X + b_0}{\|W_0\|} \quad (8.56)$$

从而有判别函数：

$$g(X) = r \|W_0\| = W_0^T X + b_0 \quad (8.57)$$

给出从 X 到最优超平面的距离的一种代数度量。

将判别函数进行归一化，使所有样本都满足：

$$\begin{aligned} W_0^T X^p + b_0 &\geq 1, & \text{当 } d^p = +1 \\ W_0^T X^p + b_0 &\leq -1, & \text{当 } d^p = -1 \end{aligned} \quad p=1, 2, \dots, P \quad (8.58)$$

则对于离最优超平面最近的特殊样本 X^s 满足 $|g(X^s)| = 1$ ，称为支持向量。由于支持向量最靠近分类决策面，是最难分类的数据点，因此这些向量在支持向量机的运行中起着主导作用。

式(8.58) 中的两行也可以组合起来用下式表示：

$$d^p (W^T X^p + b) \geq 1 \quad p=1, 2, \dots, P \quad (8.59)$$

式中， W_0 用 W 代替。

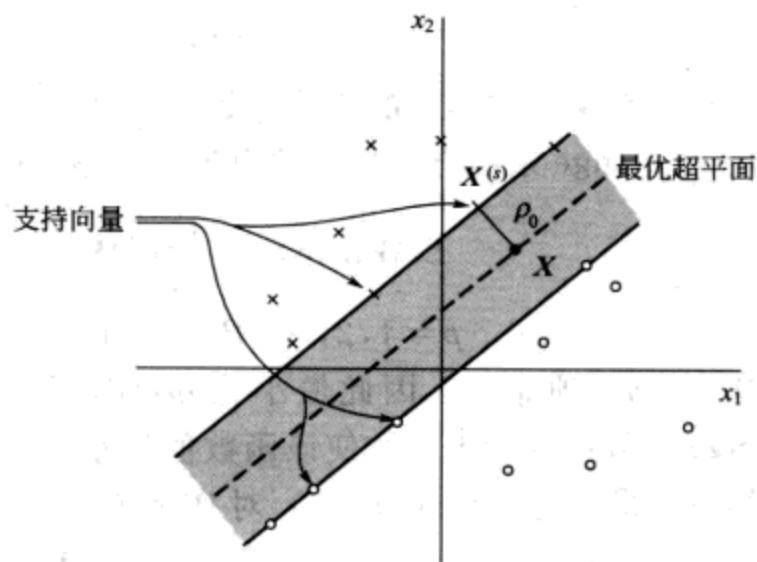


图 8.11 二维平面中的最优超平面

由式(8.56) 可导出从支持向量到最优超平面的代数距离为:

$$r = \frac{g(\mathbf{X}^s)}{\|\mathbf{W}_0\|} = \begin{cases} \frac{1}{\|\mathbf{W}_0\|}, & d^s = +1, \mathbf{X}^s \text{ 在最优超平面的正面} \\ -\frac{1}{\|\mathbf{W}_0\|}, & d^s = -1, \mathbf{X}^s \text{ 在最优超平面的负面} \end{cases} \quad (8.60)$$

因此, 两类之间的间隔可用分离边缘表示为:

$$\rho = 2y = \frac{2}{\|\mathbf{W}_0\|} \quad (8.61)$$

上式表明, 分离边缘最大化等价于使权值向量的范数 $\|\mathbf{W}\|$ 最小化。因此, 满足式(8.59) 的条件且使 $\|\mathbf{W}\|$ 最小的分类超平面就是最优超平面。

8.3.1.2 最优超平面的构建

根据上面的讨论, 建立最优线性分类超平面问题可以表示成如下的约束优化问题, 即对于给定的训练样本 $\{(\mathbf{X}^1, \mathbf{d}^1), (\mathbf{X}^2, \mathbf{d}^2), \dots, (\mathbf{X}^P, \mathbf{d}^P)\}$, 找到权值向量 \mathbf{W} 和阈值 T 的最优值, 使其在式(8.59) 的约束下, 最小化代价函数:

$$\Phi(\mathbf{W}) = \frac{1}{2} \|\mathbf{W}\|^2 = \frac{1}{2} \mathbf{W}^T \mathbf{W} \quad (8.62)$$

这个约束优化问题的代价函数是 \mathbf{W} 的凸函数, 且关于 \mathbf{W} 的约束条件是线性的, 因此可以用 Lagrange 系数方法解决约束最优问题。引入 Lagrange 函数如下:

$$L(\mathbf{W}, b, \alpha) = \frac{1}{2} \mathbf{W}^T \mathbf{W} - \sum_{p=1}^P \alpha_p [\mathbf{d}^p (\mathbf{W}^T \mathbf{X}^p + b) - 1] \quad (8.63)$$

式中, $\alpha_p \geq 0$, $p = 1, 2, \dots, P$ 称为 Lagrange 系数。式(8.63) 中的第一项为代价函数 $\Phi(\mathbf{W})$, 第二项非负, 因此最小化 $\Phi(\mathbf{W})$ 就转化为求 Lagrange 函数的最小值。观察 Lagrange 函数可以看出, 欲使该函数值最小化, 应使第一项 $\Phi(\mathbf{W})$ 减小, 使第二项增大。为使第一项最小化, 将式(8.63) 对 \mathbf{W} 和 b 求偏导, 并使结果为零:

$$\begin{aligned} \frac{\partial L(\mathbf{W}, b, \alpha)}{\partial \mathbf{W}} &= 0 \\ \frac{\partial L(\mathbf{W}, b, \alpha)}{\partial b} &= 0 \end{aligned} \quad (8.64)$$

利用式(8.63) 和式(8.64), 经过整理可导出最优化条件 1:

$$\mathbf{W} = \sum_{p=1}^P \alpha_p \mathbf{d}^p \mathbf{X}^p \quad (8.65)$$

利用式(8.63) 和式(8.64) 可导出最优化条件 2:

$$\sum_{p=1}^P \alpha_p \mathbf{d}^p = 0 \quad (8.66)$$

为使第二项最大化, 将式(8.63) 展开如下:

$$L(\mathbf{W}, b, \alpha) = \frac{1}{2} \mathbf{W}^T \mathbf{W} - \sum_{p=1}^P \alpha_p \mathbf{d}^p \mathbf{W}^T \mathbf{X}^p - b \sum_{p=1}^P \alpha_p \mathbf{d}^p + \sum_{p=1}^P \alpha_p$$

根据式(8.66), 上式中的第三项为零。根据式(8.65), 可将上式表示为:

$$\begin{aligned} L(\mathbf{W}, b, \alpha) &= \frac{1}{2} \mathbf{W}^T \mathbf{W} - \mathbf{W}^T \sum_{p=1}^P \alpha_p \mathbf{d}^p \mathbf{X}^p + \sum_{p=1}^P \alpha_p \\ &= \frac{1}{2} \mathbf{W}^T \mathbf{W} - \mathbf{W}^T \mathbf{W} + \sum_{p=1}^P \alpha_p \end{aligned}$$

$$= -\frac{1}{2} \mathbf{W}^T \mathbf{W} + \sum_{p=1}^P \alpha_p$$

根据式(8.65) 可得到:

$$\mathbf{W}^T \mathbf{W} = \mathbf{W}^T \sum_{p=1}^P \alpha_p \mathbf{d}^p \mathbf{X}^p = \sum_{p=1}^P \sum_{j=1}^P \alpha_p \alpha_j \mathbf{d}^p \mathbf{d}^j (\mathbf{X}^p)^T \mathbf{X}^p$$

设关于 α 的目标函数为 $Q(\alpha) = L(\mathbf{W}, b, \alpha)$, 则有:

$$Q(\alpha) = \sum_{p=1}^P \alpha_p - \frac{1}{2} \sum_{p=1}^P \sum_{j=1}^P \alpha_p \alpha_j \mathbf{d}^p \mathbf{d}^j (\mathbf{X}^p)^T \mathbf{X}^p \quad (8.67)$$

至此, 原来的最小化 $L(\mathbf{W}, b, \alpha)$ 函数问题转化为一个最大化函数 $Q(\alpha)$ 的“对偶”问题, 即: 给定训练样本 $\{(\mathbf{X}^1, d^1), (\mathbf{X}^2, d^2), \dots, (\mathbf{X}^p, d^p), \dots, (\mathbf{X}^P, d^P)\}$, 求解使式(8.67) 为最大值的 Lagrange 系数 $\{\alpha_1, \alpha_2, \dots, \alpha_p, \dots, \alpha_P\}$, 并满足约束条件 $\sum_{p=1}^P \alpha_p \mathbf{d}^p = 0$; $\alpha_p \geq 0, p=1, 2, \dots, P$ 。

以上为不等式约束的二次函数极值问题 (quadratic programming, QP)。由 Kuhn-Tucker 定理知, 式(8.67) 的最优解必须满足以下最优化条件 (KKT 条件):

$$\alpha_p [(\mathbf{W}^T \mathbf{X}^p + b) \mathbf{d}^p - 1] = 0 \quad p=1, 2, \dots, P \quad (8.68)$$

可以看出, 在两种的情况下上式中的等号成立: 一种情况是 α_p 为零; 另一种情况是 α_p 不为零而 $(\mathbf{W}^T \mathbf{X}^p + b) \mathbf{d}^p = 1$ 。显然, 第二种情况仅对应于样本为支持向量的情况。

设 $Q(\alpha)$ 的最优解为 $\{\alpha_{01}, \alpha_{02}, \dots, \alpha_{0p}, \dots, \alpha_{0P}\}$, 可通过式(8.65) 计算最优权值向量, 其中多数样本的 Lagrange 系数为零, 因此:

$$\mathbf{W}_0 = \sum_{p=1}^P \alpha_{0p} \mathbf{d}^p \mathbf{X}^p = \sum_{\substack{\text{所有支} \\ \text{持向量}}} \alpha_{0p} \mathbf{d}^s \mathbf{X}^s \quad (8.69)$$

即最优超平面的权向量是训练样本向量的线性组合, 且只有支持向量影响最终的划分结果, 这就意味着如果去掉其他训练样本再重新训练, 得到的分类超平面是相同的。但如果一个支持向量未能包含在训练集内时, 最优超平面会被改变。

利用计算出的最优权值向量和一个正的支持向量, 可通过式(8.58) 进一步计算出最优偏置:

$$d_0 = 1 - \mathbf{W}_0^T \mathbf{X}^s \quad (8.70)$$

求解线性可分问题得到的最优分类判别函数为:

$$f(\mathbf{X}) = \operatorname{sgn} \left[\sum_{p=1}^P \alpha_{0p} \mathbf{d}^p (\mathbf{X}^p)^T \mathbf{X} + b_0 \right] \quad (8.71)$$

在上式中的 P 个输入向量中, 只有若干个支持向量的 Lagrange 系数不为零, 因此计算复杂度取决于支持向量的个数。

对于线性可分数据, 该判别函数对训练样本的分类误差为零, 而对非训练样本具有最佳泛化性能。

若将上述思想用于非线性可分模式的分类时, 会有一些样本不能满足式(8.59) 的约束, 而出现分类误差。因此需要对适当放宽该式的约束, 将其变为:

$$\mathbf{d}^p (\mathbf{W}^T \mathbf{X}^p + b) \geq 1 - \xi_p \quad p=1, 2, \dots, P \quad (8.72)$$

式中引入了松弛变量 $\xi_p \geq 0, p=1, 2, \dots, P$, 它们用于度量一个数据点对线性可分理想条件的偏离程度。当 $0 \leq \xi_p \leq 1$ 时, 数据点落入分离区域的内部, 且在分类超平面的正确

一侧；当 $\xi_p > 1$ 时，数据点进入分类超平面的错误一侧；当 $\xi_p = 0$ 时，相应的数据点即为精确满足式(8.59)的支持向量 \mathbf{X}^s 。

建立非线性可分数据的最优超平面可以采用与线性可分情况类似的方法，推导过程与上述方法相同，得到的结果为：

$$\sum_{p=1}^P \alpha_p d^p = 0 \\ 0 \leq \alpha_p \leq C, p = 1, 2, \dots, P \quad (8.73)$$

可以看出，线性可分情况下的约束条件 $\alpha_p \geq 0$ 在非线性可分情况下被替换为约束更强的 $0 \leq \alpha_p \leq C$ ，因此线性可分情况下的约束条件 $\alpha_p \geq 0$ 可以看作非线性可分情况下的一种特例。

此外， \mathbf{W} 和 b 的最优解必须满足的最优化条件改变为：

$$\alpha_p [(\mathbf{W}^\top \mathbf{X}^p + b) d^p - 1 + \xi_p] = 0 \quad p = 1, 2, \dots, P \quad (8.74)$$

最终推导得到的 \mathbf{W} 和 b 的最优解计算式以及最优分类判别函数与式(8.69)、式(8.70)和式(8.71)完全相同。

8.3.2 支持向量机神经网络

在解决模式识别问题时，经常遇到非线性可分模式的情况。支持向量机的方法是，将输入向量映射到一个高维特征向量空间，如果选用的映射函数适当且特征空间的维数足够高，则大多数非线性可分模式在特征空间中可以转化为线性可分模式，因此可以在该特征空间构造最优超平面进行模式分类。

设 \mathbf{X} 为 N 维输入空间的向量，令 $\Phi(\mathbf{X}) = [\phi_1(\mathbf{X}), \phi_2(\mathbf{X}), \dots, \phi_M(\mathbf{X})]^\top$ 表示从输入空间到 M 维特征空间的非线性变换，称为输入向量 \mathbf{X} 在特征空间诱导出的“像”。参照前述思路，可以在该特征空间定义构建一个分类超平面：

$$\sum_{j=1}^M w_j \phi_j(\mathbf{X}) + b = 0 \quad (8.75)$$

式中， $w_j (j = 1, 2, \dots, M)$ 为将特征空间连接到输出空间的权值； b 为偏置或负阈值。

令 $\phi_0(\mathbf{X}) = 1$ ， $w_0 = b$ ，上式可简化为：

$$\sum_{j=0}^M w_j \phi_j(\mathbf{X}) = 0 \quad (8.76)$$

或写成：

$$\mathbf{W}^\top \Phi(\mathbf{X}) = 0 \quad (8.77)$$

将适合线性可分模式输入空间的式(8.65)用于特征空间中线性可分的“像”，只需用 $\Phi(\mathbf{X})$ 替换 \mathbf{X} ，得到：

$$\mathbf{W} = \sum_{p=1}^P \alpha_p d^p \Phi(\mathbf{X}^p) \quad (8.78)$$

将上式代入式(8.77)，可得特征空间的分类超平面为：

$$\sum_{p=1}^P \alpha_p d^p \Phi^\top(\mathbf{X}^p) \Phi(\mathbf{X}) = 0 \quad (8.79)$$

式中 $\Phi^\top(\mathbf{X}^p) \Phi(\mathbf{X})$ 表示第 p 个输入模式 \mathbf{X}^p 在特征空间的像 $\Phi(\mathbf{X}^p)$ 与输入向量 \mathbf{X} 在特征空间的像 $\Phi(\mathbf{X})$ 的内积，因此在特征空间构造最优超平面时，仅使用特征空间中的内积。

支持向量机的思想是，对于非线性可分数据，在进行非线性变换后的高维特征空间实现线性分类，此时最优分类判别函数为：

$$f(\mathbf{X}) = \operatorname{sgn} \left[\sum_{p=1}^P \alpha_{0p} \mathbf{d}^p \Phi^T(\mathbf{X}^p) \Phi(\mathbf{X}) + b_0 \right] \quad (8.80)$$

令支持向量的数量为 N_s ，去除系数为零的项，上式可改写为：

$$f(\mathbf{X}) = \operatorname{sgn} \left[\sum_{s=1}^{N_s} \alpha_{0s} \mathbf{d}^s \Phi(\mathbf{X}^s) \Phi(\mathbf{X}) + b_0 \right] \quad (8.81)$$

从支持向量机分类判别函数的形式上看，它类似于一个 3 层前馈神经网络。其中隐层节点对应于输入样本与支持向量的像的内积，而输出节点对应于隐层输出的线性组合。图 8.12 给出支持向量机神经网络的示意图。

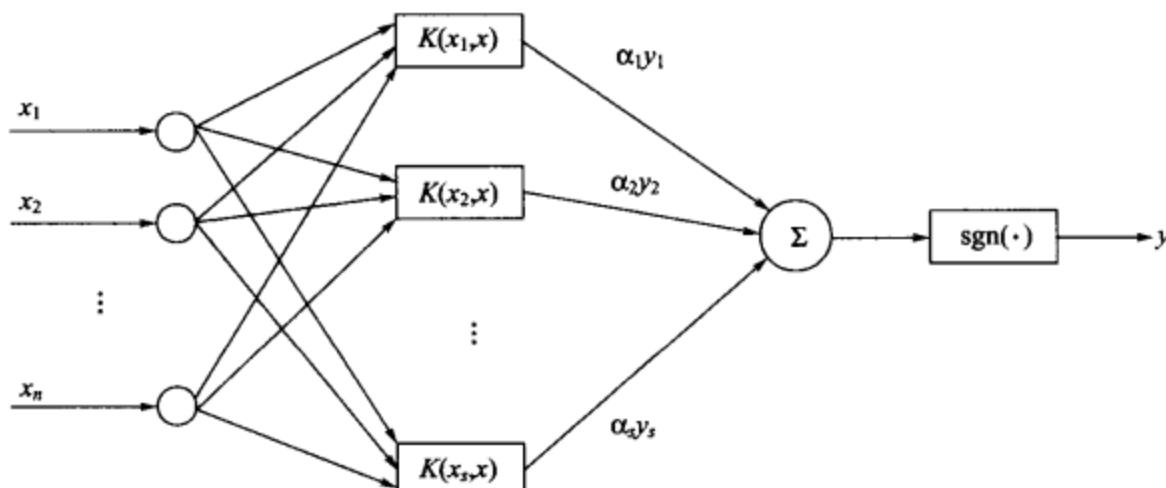


图 8.12 支持向量机神经网络

设计一个支持向量机时，需构建非线性映射 $\Phi(\cdot)$ 。设输入数据为二维平面的向量 $\mathbf{X} = [x_1, x_2]^T$ ，共有 3 个支持向量，因此应将二维输入向量非线性映射为三维空间的向量 $\Phi(\mathbf{X}) = [\phi_1(\mathbf{X}), \phi_2(\mathbf{X}), \phi_3(\mathbf{X})]^T$ 。

8.3.3 支持向量机的学习算法

在能够选择变换 ϕ （取决于设计者在这方面的知识）的情况下，用支持向量机进行求解的学习算法如下：

- (1) 通过非线性变换 ϕ 将输入向量映射到高维特征空间。
- (2) 在约束条件 $\sum_{p=1}^P \alpha_p d^p = 0$, $0 \leq \alpha_p \leq C$ (或 $\alpha_p \geq 0$), $p = 1, 2, \dots, P$ 下求解使目标函数

$$Q(\alpha) = \sum_{p=1}^P \alpha_p - \frac{1}{2} \sum_{p=1}^P \sum_{j=1}^P \alpha_p \alpha_j d^p d^j \Phi^T(\mathbf{X}^p) \Phi(\mathbf{X}^j) \quad (8.82)$$

最大化的 α_{0p} 。

- (3) 计算最优权值：

$$\mathbf{W}_0 = \sum_{p=1}^P \alpha_{0p} \mathbf{d}^p \Phi(\mathbf{X}^p) \quad (8.83)$$

- (4) 对于待分类模式 \mathbf{X} ，计算分类判别函数：

$$f(\mathbf{X}) = \operatorname{sgn} \left[\sum_{p=1}^P \alpha_{0p} \mathbf{d}^p \Phi^T(\mathbf{X}^p) \Phi(\mathbf{X}) + b_0 \right] \quad (8.84)$$

根据 $f(\mathbf{X})$ 为 1 或 -1，决定 \mathbf{X} 的类别归属。

支持向量机常被用于径向基函数网络和多层感知器的设计中。在径向基函数类型的支持向量机中，径向基函数的数量和它们的中心分别由支持向量的个数和支持向量的值决定，而传统的 RBF 网络对这些参数的确定则依赖于经验知识。在单隐层感知器类型的支持向量机中，隐节点的个数和它们的权值向量分别由支持向量的个数和支持向量的值决定。

与径向基函数和多层感知器相比，支持向量机的算法不依赖于设计者的经验知识，且最终求得的是全局最优值而不是局部极值，因而具有良好的泛化能力而不会出现过学习现象。但支持向量机由于算法复杂导致训练速度较慢，目前提出的一些改进训练算法是基于循环迭代的思想。

8.3.4 支持向量机处理 XOR 问题

为了说明支持向量机的设计过程，讨论如何用 SVM 处理 XOR 问题。4 个输入样本和对应的期望输出如图 8.13(a) 所示。

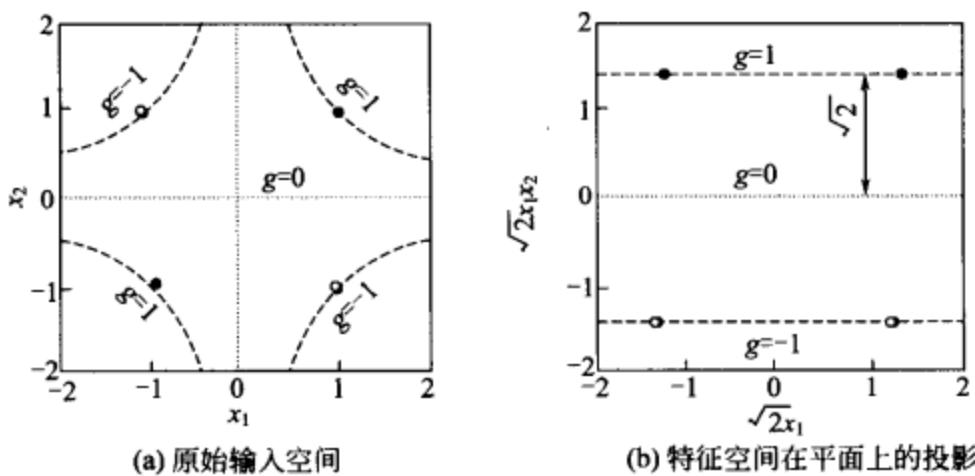


图 8.13 XOR 问题的样本分布

选择映射函数 $\Phi(\mathbf{X}) = [\phi_1(\mathbf{X}), \phi_2(\mathbf{X}), \dots, \phi_M(\mathbf{X})]^T$ ，将输入样本映射到更高维的空间，使其在该空间是线性可分的。有许多这样的映射函数，例如， $\Phi(\mathbf{X}) = [1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2]^T$ 可将 2 维训练样本映射到一个六维特征空间。这个六维空间在平面上的投影如图 8.13(b) 所示。可以看出分离边缘为 $\rho = \sqrt{2}$ ，通过支持向量的超平面在正负两侧平行于最优超平面，其方程为 $\sqrt{2}x_1x_2 = \pm 1$ ，对应于原始空间的双曲线 $x_1x_2 = \pm 1$ 。

寻求使

$$\begin{aligned} Q(\alpha) &= \sum_{p=1}^P \alpha_p - \frac{1}{2} \sum_{p=1}^P \sum_{j=1}^P \alpha_p \alpha_j d^p d^j \Phi^T(\mathbf{X}^p) \Phi(\mathbf{X}_j) \\ &= \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2} (9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 + 9\alpha_2^2 + 2\alpha_2\alpha_3 \\ &\quad - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2) \end{aligned}$$

最大化的 Lagrange 系数，约束条件为：

$$\alpha_1 - \alpha_2 + \alpha_3 - \alpha_4 = 0$$

$$\alpha_p \geq 0 \quad p=1, 2, 3, 4$$

从该问题的对称性，可取 $\alpha_1 = \alpha_3$, $\alpha_2 = \alpha_4$ 。 $Q(\alpha)$ 对 α_p , $p=1, 2, 3, 4$ 求导并令导数为零，得到下列联立方程组：

$$\begin{aligned} 9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 &= 1 \\ -\alpha_1 - 9\alpha_2 + \alpha_3 - \alpha_4 &= 1 \\ -\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 &= 1 \\ \alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 &= 1 \end{aligned}$$

解得 Lagrange 系数的最优值为 $\alpha_{0p} = 1/8$, $p=1, 2, 3, 4$, 可见 4 个样本都是支持向量, $Q(\alpha)$ 的最优值为 $1/4$ 。根据式(8.83) 可写出:

$$\begin{aligned} \mathbf{W}_0 &= \sum_{p=1}^4 \alpha_{0p} \mathbf{d}^p \Phi(\mathbf{X}^p) = \frac{1}{8} [-\phi(\mathbf{X}^1) + \phi(\mathbf{X}^2) + \phi(\mathbf{X}^3) - \phi(\mathbf{X}^4)] \\ &= \frac{1}{8} \left[-\begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ -\sqrt{2} \\ \sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \\ \sqrt{2} \end{bmatrix} \right] = \begin{bmatrix} 0 \\ 0 \\ -1/\sqrt{2} \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{aligned}$$

在六维特征空间中找到的最优超平面为:

$$\mathbf{W}_0^T \Phi(\mathbf{X}) = \begin{bmatrix} 0 & 0 & \frac{-1}{\sqrt{2}} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \end{bmatrix} = -x_1x_2 = 0$$

图 8.13 中将最优超平面 $x_1x_2=0$ 投影到二维空间后成为与 $\sqrt{2}x_1$ 轴平行的直线。

支持向量机是一种通用的前馈神经网络, 可用于解决模式分类与非线性映射问题。根据结构风险最小化准则, 支持向量机在使训练样本分类误差极小化的前提下, 尽量提高分类器的泛化推广能力。从实施的角度, 训练支持向量机的核心思想等价于求解一个线性约束的二次规划问题, 从而构造一个最优决策超平面, 使得该平面两侧距平面最近的两类样本之间的距离最大化, 从而对分类问题提供良好的泛化能力。

对于非线性可分模式的分类问题, 支持向量机将输入模式非线性地映射到高维特征空间, 在该空间样本模式以较高的概率成为线性可分的。此时, 应用支持向量机算法在特征空间建立分类超平面, 即可解决非线性可分的模式识别问题。

上面讨论的支持向量机只能解决二分类问题, 目前没有一个统一的方法将其推广到多分类的情况, 但已有不少设计者针对具体问题提出了值得借鉴的方法, 读者可参考相关论文。

本章小结

本章讨论了 3 种基于数学原理的神经网络模型, 各网络模型的特点如下:

(1) 基于多变量有限点严格 (精确) 插值技术的径向基函数网络 用 RBF 网络解决插值问题时, 其于正则化理论的 RBF 网络称为正则化网络。其特点是隐节点数等于输入样本数, 隐节点的激活函数为 Green 函数, 并将所有输入样本设为径向基函数的中心, 各径向基

函数取统一的扩展常数。当 $N < M$ 时, 得到广义 RBF 网络, 其各径向基函数的扩展常数不再统一, 且输出函数的线性中包含阈值参数。介绍了应用 K-means 聚类算法确定数据中心的方法以及基于监督学习规则的梯度下降算法。

(2) 基于 K-L 变换方法的主分量分析网络 主分量分析包括特征选择和特征提取过程。特征选择过程通过一种可逆变换 T 将从数据空间映射到同维特征空间, 从而获得输入的特征, 即输入的主分量; 而特征提取过程的目的是降维, 即对变换后的特征空间向量进行截断, 选取主要特征分量而舍去其他特征分量。主分量分析提供的可逆变换 T 能够保证对 $T(X)$ 的截断在均方差意义下为最优。

(3) 基于最优超平面构建的支持向量机网络 支持向量机是一种通用的前馈神经网络, 可用于解决模式分类与非线性映射问题。训练支持向量机的核心思想是构造一个最优决策超平面, 使得该平面两侧距平面最近的两类样本之间的距离最大化, 从而对分类问题提供良好的泛化能力。对于非线性可分模式的分类问题, 支持向量机将输入模式非线性地映射到高维特征空间, 在该空间样本模式的以较高的概率成为线性可分的。此时, 应用支持向量机在算

法在特征空间建立分类超平面, 即可解决非线性可分的模式识别问题。



9

神经网络的系统设计与软件实现

人工神经网络的发展为探索新型智能计算机开辟了一条崭新的途径，要使神经网络的并行处理能力得以充分发挥，还有赖于其硬件实现。然而，目前各类神经计算机的研究还远未成熟，应用神经网络解决实际问题时主要靠软件进行虚拟实现。因此了解并掌握神经网络软件设计与开发方面的基本知识对于提高应用神经网络解决问题的能力和水平至关重要。本章主要从系统设计和软件开发与运行等几个方面加以阐述。

9.1 神经网络系统总体设计

一个设计良好的神经网络系统能代表问题求解的系统方法。开发神经网络系统的总体设计过程中应考虑这样几个问题：①首先分析哪类问题需要使用神经网络；②神经网络系统的整体处理过程的设计，即系统总图；③系统需求分析；④设计系统的各项性能指标；⑤预处理问题。下面就这几个问题进行讨论：

9.1.1 神经网络的适用范围

神经网络能用来解决多种问题，但并不是擅长解决所有问题。可以把要解决的问题分为四种情况：第一种情况是除了神经网络方法还没有已知的其他解决方法；第二种情况是或许存在别的处理方法，但使用神经网络显然最容易给出最佳的结果；第三种情况是用神经网络与用别的方法性能不相上下，且实现的工作量也相当；第四种情况是显然有比使用神经网络更好的处理方法。为了在不同情况下使用最适合的方法，首先要判断待解决的问题属于以上哪一种情况。这种判断需始终着眼于系统进行，力求最佳的系统整体性能。

一般来说，最适合于使用神经网络分析的那类问题应具有如下特征：关于这些问题的知识（数据）具有模糊、残缺、不确定等特点，或者这些问题的数学算法缺少清晰的解析分析。然而最重要的还是要有足够的数据来产生充足的训练和测试模式集，以有效地训练和评价神经网络的工作性能。训练一个网络所需的数据量依赖于网络的结构、训练方法和待解决的问题。例如，对BP网来说，对每个输出分类大约需要十几个至几十个输入模式向量；而对自组织网络来说，在选择输出节点数时需要把估计的分类数作为一个因素考虑在内，因此每种可能的分类取十几至几十个模式只是指导性的出发点。设计测试模式集所需的数据量与用户的需求和特定应用密切相关。因为神经网络的性能必须用足够的检测实例和分布来表示，而用于分析结果的统计方法和特性指标必须有意义和有说服力。对于哪些问题用神经网络解决效果最好，开发者需要逐渐积累经验，总结出自己的原则。

当确定一个问题要用神经网络解决后，接着就要确定用什么样的网络模型和算法。如果有一组确知分类的输入模式数据，就可通过训练BP网开始试探解决问题。若不知道答案（分类）应该是什么，可从某种自组织学习网络结构入手。试验时可尝试使用不同的网络结

构和网络参数（如学习率或动量系数等），并对其效果进行比较。

神经网络在应用中常常作为一个子系统在系统中的一个或多个位置出现，系统中的一个或多个神经网络往往起着各种各样的作用，在系统的详细设计过程中要尽可能开放思路，考虑不同的作用与组合。事实上，在许多应用中都使用若干个网络或多次使用网络，还有可能采用子网络构造大结构，甚至不同的网络也可拓扑组合成一个单一的结构。例如，用自组织网络对数据进行预处理，然后用其输出节点作为执行最终分类的反向传播网络的输入节点。又如，神经网络可作为专家系统中的数据预处理子系统，或作为从原始数据中提取参数的特征提取子系统。有时需要将多个网络模型结合使用，其中每个网络均作为综合网中的子网出现。总之，神经网络在实际应用中存在许多可能的形式，因此应用神经网络解决问题时要放开思路。

9.1.2 神经网络的设计过程与需求分析

神经网络的设计开发过程可以用图 9.1 所示的系统总图来描述。设计过程要完成的工作任务有三项：首先要做的是系统需求分析；接下来是数据准备，包括训练与测试数据的选择、数据特征化和预处理以及产生模式文件，在此过程中强调要求系统的最终用户参加，目的是保证训练数据和测试结果的有效性；第三个是与计算机有关的任务，包括软件编程与系统调试等内容。

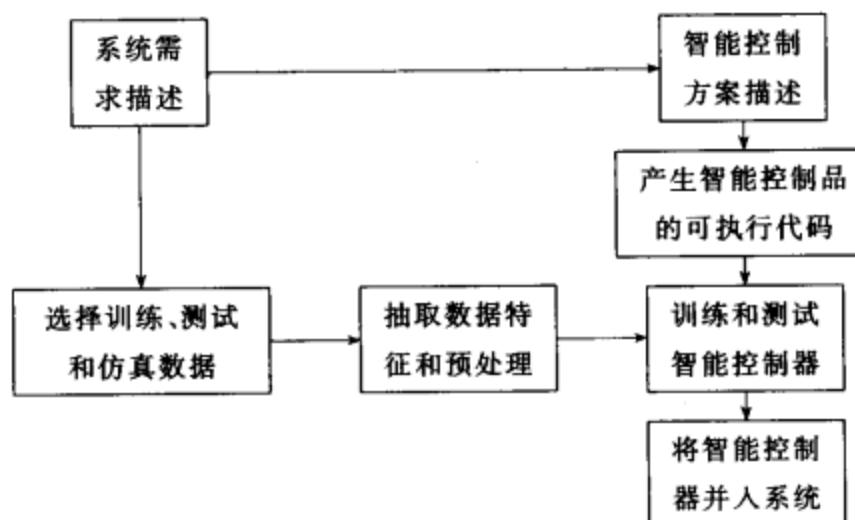


图 9.1 神经网络系统开发总图

系统需求分析一般应包括以下内容：

(1) 系统需求说明 系统分析是系统开发过程中最重要的工作之一，因为此阶段的错误和疏忽会对项目产生代价昂贵的后果。系统分析阶段的产物是系统详细需求分析文档，以便准确描述系统的行为和评估完成状况。

(2) 结构化分析 结构化分析使用一套工具来产生结构化需求说明，由数据流图、数据词典和结构化文字几部分组成。数据流显示的是在系统和环境间以及处理过程间的信息和控制信号流，并将需求模型图形化和生动化。使用结构化文字强调的是可读性而不是自动分析的能力，其目的是在某种程度上能与不懂计算机的用户沟通。

(3) 层次结构 数据流图是分等级按层次构造的，可用一些相互关联的图表示。如图 9.2 将整个系统看成单一的处理过程以及系统与环境间的数据流，图 9.3 将单一过程扩展，揭示了关于系统模型的更详尽的细节，包括一些过程、流和数据存储。图 9.3 中的每个过程都能够依次扩展成更详细的图表，分解可一直继续至任意的细节水平直至最终使用结构化文字能够充分描述的最低层次。层次结构是系统建模和系统构造的有力工具，它证明了结构化分析

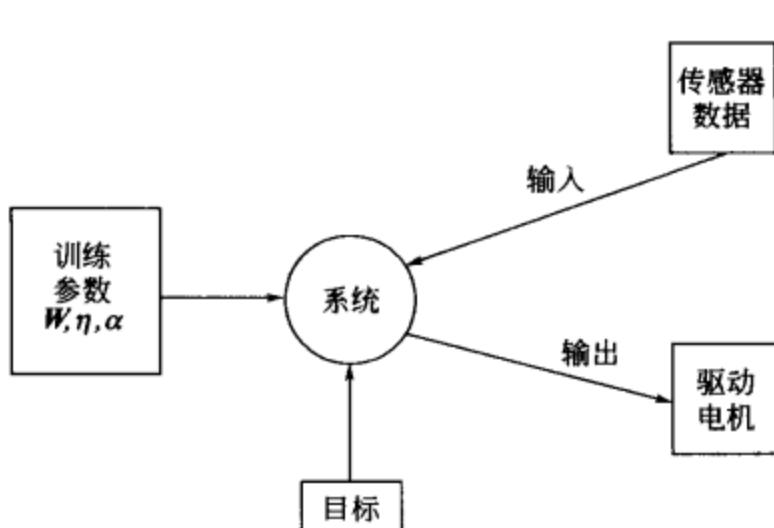


图 9.2 数据流图

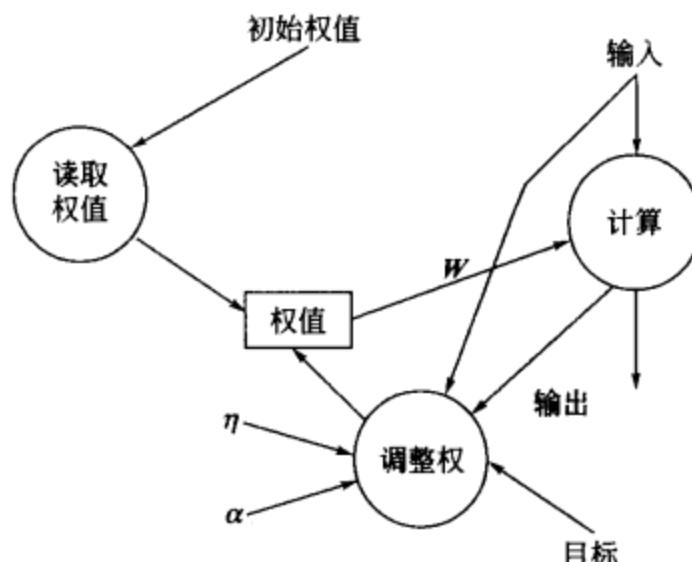


图 9.3 处理过程的细化

是较有效的系统描述技术。

(4) 数据词典 数据词典是结构化分析的主要工具, 目前普遍被用作一种称为系统的百科全书的软件工程数据库, 以存储数据元素说明以及系统模型中对象与方法的需求说明。

9.1.3 神经网络的性能评价

为了评价一个系统的运行质量, 需要把对系统进行测试运行时得到的数据和已建立的标准相比较。为了研究有关神经网络的运行质量, 必须首先建立一些能反映其质量的性能指标, 这些指标应对不同的网络具有通用性和可比性。目前在这方面尚缺乏系统而深入的研究, 但仍可借鉴相关领域的运行检测技术。下面简要介绍关于神经网络性能评价的几个常用指标, 而应用时选用哪一种指标取决于系统的类型以及使用者的技术水平等因素。

(1) 百分比正确率 神经网络运行的百分比正确率就是根据某种分类标准作出正确判断的百分比。神经网络用于模式识别和分类等问题时, 常用到该指标。但在某些神经网络应用中, 百分比正确率的概念不太适用, 应采用其他指标。为了计算正确率应选择合适的分类标准和有代表的训练集和测试集。有两个因素会影响以上选择的合理性: 一个是分类标准本身的不确定问题; 另一个是样本集的代表性。例如, 当神经网络用于对印刷体字母分类时, 不存在判断标准的不确定性问题。但是有些分类任务会存在较大的主观因素, 例如烤烟烟叶质量定级的分类, 随专家的观点不同分类结果会略有差异。在用神经网络检测癫痫棘波时, 6位神经科医生中任何 2 位共同认定的单个棘波的平均一致率仅为 60%。因此在分类之前统一观点是十分重要的, 这需要系统最终用户的积极参与, 才能正确建立统一的分类标准。训练集和测试集样本的代表性是目前神经网络开发工作正在研究课题之一。训练集和测试集的样本及由专家认定的代表类必须分布在所有类的范围内, 包括那些在判断临界点附近的样本。设计者的人为因素也是非常重要的, 应该避免设计者自己闭门造车地进行代表样本的分类确定工作, 而要让系统的用户参与这一过程。虽然设计人员能为用户提供系统运行的技术要求等信息, 但在样本设计和整个设计过程中都应该让用户尽可能发挥作用。在测试和训练中要使用不同的样本集, 当样本数不充足时, 可以循环利用已有的样本进行训练和测试。此外, 所选的训练集应该使每种样本的分类结果具有相同的数目。即, 如果神经网络有三个输出节点, 对于每一次分类有一个相应的节点激活, 则训练样本集中每种分类结果的样本数目应该定为总样本数目的 1/3。

(2) 均方误差 神经网络的均方误差为总误差除以样本总数, 而总误差定义为:

$$E = \frac{1}{2} \sum_{j=1}^m \sum_{p=1}^P (d_j^p - o_j^p)^2 \quad (9.1)$$

正如第3章所讨论的，应用反向传播算法训练神经网络的目的是使均方误差最小。在应用均方误差时，应注意两种情况：第一，均方误差的定义公式中包括乘积因子1/2，但是在许多应用场合都省略了该因子，因此在比较各种不同的神经网络时应注意均方误差的计算中是否包含乘积因子1/2；第二，误差项对所有输出节点求和时会产生一个潜在的问题，均方误差无法精确地反映具有不同节点数的神经网络结构之间的差别。如果训练一个单输出节点的神经网络能达到一固定误差，而训练一个结构基本相同的多输出节点的神经网络时，误差可能会增大，这是因为均方误差定义为除以训练集或测试集中的样本数而不是除以节点数。在某些应用场合，用户要求计算每个节点的误差，可以定义节点平均均方误差为（样本平均）均方误差除以输出节点数。由于平均节点均方误差主要用于反向传播算法，所以它主要用于BP网络的性能评价。

(3) 归一化误差 Pinda 提出了一种与神经网络结构无关，取值为0~1的误差标准 E_{mean} 。定义为：

$$E_{\text{mean}} = \frac{1}{2} \sum_{j=1}^m \sum_{p=1}^P (d_j^p - \bar{d}_j)^2 \quad (9.2)$$

式中， \bar{d}_j 为所有样本在第 j 个输出节点的期望输出值的平均值； d_j^p 是 j 个输出节点的期望输出值； P 为样本总数； m 为输出节点数。则归一化误差 E_n 定义为式(9.1)的总误差 E 除以上式的 E_{mean} ：

$$E_n = E / E_{\text{mean}} \quad (9.3)$$

归一化误差对BP神经网络十分有用。当神经网络“猜测”正确的输出值是平均目标值时，出现“最坏的情况”是 $E_n=1$ 。当样本学习结束后， E_n 的值趋向于零，其速度取决于神经网络的结构。归一化误差反映的是基于误差的输出方差的比例，而与神经网络本身的结构（包括初始化的随机权值）无关。因此，在大多数场合，归一化误差标准是BP神经网络中最有价值的误差标准之一。

(4) 接收操作特性曲线 评价神经网络系统的另一个途径是接收操作特性曲线(ROC)。ROC曲线用来反映系统某一个输出节点在作出一个判断时的正确性，因此下面的讨论集中于单输出节点网络。若用判断的阳性和阴性表示将某一输入样本判断为某类的肯定与否定，一个给定输出神经元所表示的判断存在四种可能性，列于表9.1中。第一种可能性称为真阳性判断(TP)，即系统的阳性判断与根据标准得到的阳性判断相一致，比如系统鉴别出神经科医生确认的癫痫棘波；第二种可能性称为假阳性判断(FP)，即系统作出阳性判断而标准作出阴性判断，例如系统判断出的癫痫棘波神经科医生认为不对；第三种可能性是假阴性判断(FN)，即标准作出阳性判断而系统作出阴性的判断，例如神经科医生鉴别出的癌病棘波系统却未找出；第四种可能性是真阴性判断(TN)，即系统和标准都作出阴性判断，如系统和神经科医生都判断不存在廓瘤棘波。

表 9.1 ROC 曲线定义中的可能性表

系统判断	标准判断	
	阳性	阴性
阳性	TP	FP
阴性	FN	TN

利用上述这四种可能性的两种比例可绘出 ROC 曲线：第一种比例是 $TP/(TP+FN)$ ，称为真阳性率（在某些应用场合称为灵敏度）；第二种比例是 $FP/(FP+TN)$ ，称为假阳性率。ROC 曲线由真阳性率轴和假阳性率轴上的点连接而成。为了画出真阳性率/假阳性率坐标轴中的点，可对输出节点设置不同的判断阈值。对于每个选定的阈值，统计出系统判断结果的真阳性率和假阳性率作为 ROC 曲线上点的坐标值。图 9.4 给出两种不同结构的神经网络的 ROC 曲线，曲线 NNT2 代表的系统比 NNT1 所代表的系统整体运行性能更好。坐标轴对角线上的虚线表示真/假阳性率相等，即无法判断的情况。

如果用单一指标评价系统运行情况，可以通过计算 ROC 曲线下所包围的面积决定，这实际上是用 ROC 曲线来评价系统运行性能的主要方法。整图的面积是一个单位方格，ROC 曲线以下的面积是整图的一个部分，曲线以下的面积必定在 $0.5\sim1.0$ 之间，前者是当系统无法判断时对角线以下部分的面积，后者是当系统判断完全正确时曲线以下的面积。一种简单的计算方法是用直线线段连接相邻的点，并计算梯形折线以下的面积。为得到较光滑的 ROC 曲线，大约需要 9~10 个点。

(5) 灵敏度、精度和特异度 灵敏度是指实际存在的事物能被检测到的可能性，也称为回忆度，其定义与 ROC 曲线定义中的真阳性率相同。在某些要求防止出现漏检事件的场合，如在预后严重的 AIND 病检测中，该指标变得非常重要。精度是系统所作的正确的阳性判断数目除以系统作出的所有阳性判断的总数，在表 9.1 中，就是 $TP/(TP+FP)$ ，它包含着假阳性判断的强度。特异度是指一件实际不存在的事物被检测为不存在的可能性，定义为 $TN/(FP+TN)$ ，或称为真阴性率。可以看出，灵敏度、精度和特异度是表 9.1 中四种数量的另一种表达方式。

9.1.4 输入数据的预处理

在设计神经网络时，预处理是最难处理的问题之一。一方面预处理有许多种类，另一方面预处理有许多种实现方法。大多数神经网络通常需要归一化的输入，即每个输入的值要始终在 $0\sim1$ 之间，或者每个输入向量的长度要为常量（如 1）。前者用于反向传播网络；而后者用于自组织网络。虽然对 BP 网的输入归一化的必要性看法不一，但对多数应用来说归一化是一种好的做法。

数据归一化通常有两种情况。一种常见的情况是，输入 BP 网各个输入节点的原始数据是同源的，常常代表了时间间隔的取样。例如，电压波形以一定的频率采样得到一定数目的采样值成组地输入网络。在这种情况下，归一化必须在所有的通道上统一地进行。如果数据分布在最大值 (X_{max}) 与最小值 (X_{min}) 之间，则首先把所有的值加上 $-X_{min}$ ，使它们分布于 0 和 $(X_{max}-X_{min})$ 之间，然后再用每个值除以 $X_{max}-X_{min}$ 。这样所有的值被归一化成 0 和 1 之间的数。 X_{max} 和 X_{min} 很可能是从不同的通道上获得的，所以又称为交叉通道归一化。第二种常见的情况是，用不同种类的参数作为输入，如电压、持续时间、波形的尖峰参数等。更有可能是一些统计参数如标准方差、相关系数和 K 方检验参数等。

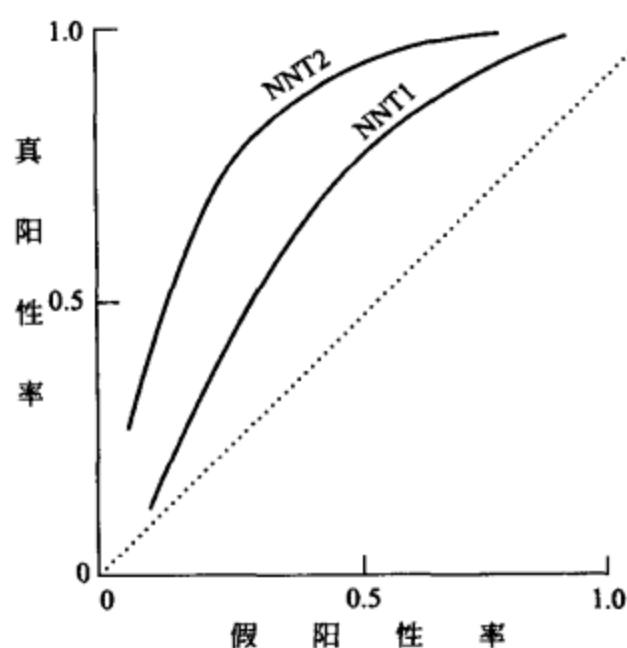


图 9.4 ROC 曲线示例

这种情况下，跨所有通道的归一化会使网络的训练失败。例如，某些通道表示波形尖锐度，它们只能从 $-0.1 \sim +0.1$ 变化。其他通道表示波形振幅，能从 $-50 \sim +50$ 之间变化，显然归一化后尖锐度将被振幅所淹没。每种类型通道中若存在0.1单位的偏差，归一化后将变为0.001单位的偏差。0.1%的动态范围在振幅通道里可能很容易接受，然而在波形尖锐度中，0.1的偏差代表了50%的动态变化，所以会严重影响网络训练或测试时对尖锐度参数的分辨能力。当输入为不同种类的参数时，对每个通道单独归一化的优点是每个通道可在 $0 \sim 1$ 区间反映其动态变化范围，缺点是任意两个通道之间的关系偏离了一个偏移量和多重因素的范围，因此每个通道单独归一化有时会在训练网络时造成困难。例如，用从生物电位波形中提取的参数训练网络，其中2个参数是振幅，3个是宽度，3个是尖锐度，另一个参数是斜率。振幅的测量单位是伏特，宽度是秒，而尖锐度是度数。对于宽度参数，把其中两个相加成为过零点间半波的宽度，第三个是半波二阶导数的两点间宽度。在原始数据中，前两个宽度之和总是大于第三个宽度，而其中任意一个又小于第三个宽度。这一类关系以及在两个振度或3个尖锐度参数之间存在的类似的其他关系，在单独通道的归一化处理中就被遮掩了。

9.2 神经网络的软件实现

神经网络软件要求在硬件平台上实现神经网络。神经网络编程语言既可用高级语言也可用低级语言。C语言是神经网络应用软件的基本编程工具，但其他高级语言也同样适用。汇编语言虽然只在程序代码中占很小一部分，但它是开发人员的另一种基本工具。它常用于提高神经网络的已有功能或解决与硬件相关的难点。

9.2.1 软件运行的若干问题

运行神经网络软件时主要涉及以下几个问题：

(1) 网络的输入输出 神经网络一般采用文本文件保存输入输出模式对和权值，其优点在于用户能直接阅读和用字处理工具进行编辑，而且能通过磁盘互相传递，通过打印机输出或通过电子邮件进行传递；另一个优点是容易输入数据库并进行分析。文本文件的缺点是占用字节数多，每个数据可能占用多达10个字符，而用浮点二进制数表示仅需4个字节。

(2) 数据归一化 输入模式必须先归一化后再输入神经网络，较好的方式是把归一化处理放在预处理阶段进行，而不是在神经网络内作归一化，这样不仅能减少神经网络的代码，而且也具有灵活方便、可移植性和通用性强的优点。

(3) 连接权 输入网络的连接权和阈值是一组初值。输入循环的形式是：

```
for(i=0; i<=nInputNode; i++)
```

C语言中数组的N个元素按0到 $N-1$ 标记，对于各隐层和输出层的每个神经元，上述循环从相应的输入层读取每个连接权和阈值，阈值存储在权向量的第一项或最后一项。开始训练时，权值随机分布在一个很小的范围之内（如 ± 0.3 ），权初始值文件可以由一个随机数产生器产生。训练结束后，最后得到的权值矩阵写入输出文件。输出文件和输入样本文件相仿，通常是文本文件。把权值写入文件有两个原因：

第一，训练结束后获得的权值代表了系统学习的结果，设计者可以利用这些权值来测试系统识别和分类新模式的能力。通常，网络对一组训练模式训练一次，然后反复运用训练结果来对新模式进行识别。在识别过程中网络进行的是正向传递运算，对每一个模式仅需迭代

运算一次。在训练时，网络必须进行正向和反向运算，反复迭代导致运算量急剧增大，因此可以在大型机上训练大型神经网络，然后把训练得到的权值文件送到 PC 机上运行神经网络。这是一种十分有价值的机器之间传递“学习”的技术，一台机器通过训练所得的结果可以快速传送到另一台机器，在不同的环境下运行神经网络。

第二，训练结束后保存下来的权值可以重新输入网络继续训练。在某些情况下需要重新训练权值，对它进行修正以改善神经网络的性能，这时只需将以前保存的权值作为网络的初始权值进行训练。

(4) 特性参数 神经网络的特性参数包括网络的拓扑结构、处理单元数、转移函数类型、学习规则、学习率和动量因子等。其中多数特性参数设计是由代码实现的算法，后在运行过程中难以更改。但学习率和动量因子等参数可以在运行过程中进行修改。

(5) 运行状况监测 反向传播算法的目的是使均方误差最小，一种较好的观察收敛趋势的方法是用显示出误差相对于迭代次数的曲线。通常用均方误差作为最基本的网络运行监测目标，但有时其他监测指标也十分有价值，例如神经网络的净输入和权值。神经网络的状态可以在运行时连续显示，即在每一次迭代后刷新。由于显示占用了神经网络的处理时间，所以在实用系统中不太理想。另外，如果网络的迭代速度太快时，无法看清屏幕上快速变化的内容。通常，可以设定经过许多次迭代后刷新一次显示状态，或者暂停运行来观察“冻结”的显示状态。另外一种方法就是在神经网络运行结束后非实时地显示。

9.2.2 软件实现的若干问题

在神经网络的实现过程中应考虑到下面一些问题：

(1) 解释和编译 许多文章介绍的或已商品化的神经网络系统中使用的是解释器而不是编译器。这在超高速主机中是可以接受的，因为这些主机的 CPU 速度快，完全可以补偿超前解释的时间，尤其在运行节点和连接权相对较少的神经网络时更是如此。然而，在 PC 机上这种方法基本上行不通，因为实时问题需要大量的节点和连接权。在大多数情况下，为了追求高精度，不得不把大量的注意力放在代码的实时效率和充分利用 PC 机上，而神经网络代码的编译是一种解决方法。采用解释器的一个主要原因是为了充分利用它所提供的良好的调试和软件开发环境，例如 Lisp 和 Basic 语言。Lisp 在人工智能领域中已比较完善，并且成为能运用于神经网络环境中的候选语言。然而 Lisp 通常运行于大型机或专用的 Lisp 机上，作为在 PC 机上实现神经网络的语言来说显得速度太慢并且内存不够。而编译器一般具有相当完善的集成开发环境，包括灵活多变的调试工具，基本替代了解释器过去所拥有的优点。

(2) 代码优化

① 优化代码 在神经网络领域中，对一小段代码的具体细节仔细设计能使无法很好运行的系统在一段可以接受的时间内完成任务。

② 优化编译器 通常编译器擅长于在代码段、寄存器分配、循环等方面进行优化，在总体系统结构和高精度算法方面还需要用户自己来决定。例如，在 BP 神经网络代码中，输入模式在训练时存储于内存中。尽管对神经网络的大量代码来说这仅仅是一个次要的细节，但是由于从磁盘上读取模式向量需要几个毫秒或者更多时间，而从内存中读取仅需要几个微秒，仅此一点，就对系统运行性能产生很大的影响。

③ 产生机器代码或汇编语言代码 对于由编译器生成的代码用手工优化使速度再加快一点是完全有可能的。对于神经网络来说，被代码占用的存储空间很少，因此可以基本上不考虑存储空间限制，而只要求速度最快。为此需要一种能嵌入汇编语句的编程语言工具，否

则只能通过调用外部函数实现。调用外部函数的方式抵消了汇编语言的好处，因为调用一个函数必须做一些压入和弹出参数、保存和恢复寄存器之类的前置工作。

(3) 内存的限制 运行神经网络所需用的内存容量取决于可执行代码的大小和网络的规模。对于 3 层 BP 神经网络，所需内存的估算可用下面经验公式：

$$\text{总存储量(字节数)} = (\text{权值总数} + \text{输入值总数}) \times 4 \times 2$$

式中， $\times 4$ 是由于每个浮点数占 4 个字节； $\times 2$ 是由于 δ 和 Δw 使内存容量加倍。

此外：

$$\text{权值总数} = \text{输出节点数} \times (\text{隐节点数} + 1) + \text{隐节点数} \times (\text{输入节点数} + 1)$$

$$\text{输入值总数} = \text{样本数} \times \text{输入节点数}$$

式中， $+1$ 是由于阈值使每个节点增加了一个权值。为了加快训练速度，输入样本应一次性从输入设备全部读入内存。另一种方法是当网络运行用到需要输入时反复从磁盘中读入相应的模式。两种方法都是可行的，前一种方法的运行速度要快得多，但需要更大的内存容量；而把模式保留在磁盘上尽管速度慢，但是所需的内存空间少。对于网络大小的最终限制将取决于处理器的速度。

(4) 浮点运算 尽管在 PC 机上运行整型数表示的神经网络比用浮点数快得多，且节省一半内存空间，这里还是建议使用浮点数表示法。浮点数表示的数值在动态范围和有效长度方面具有优势，可以保证运算精度。

(5) 神经网络的调试 神经网络的调试包括对所选网络结构与参数的调试以及对神经网络实现代码的调试。前一种调试的基本做法是根据网络的运行性能和设计经验对其进行改进，有关内容已在第 3 章进行了讨论，下面讨论神经网络的代码调试。神经网络的代码调试与一般的程序调试无本质区别，常用方法是利用编译器的调试工具来检查和验证网络运行的正确性。例如，单步执行程序以检查任一点的状态，在程序中设置断点检查或改变变量和寄存器的内容，在关键点设置确认语句以检查网络运行中某一步的给定条件是否满足等。通过这些方法可以快速找出问题所在。

9.3 神经网络的高级开发环境

许多研究人员在开发自己的神经网络软件时都会碰到这样的问题：花在人工编码和跟踪调试上的巨大工作量大大影响了真正要进行的模型设计工作。开发环境能快捷方便地建立一个新的网络模型，通过试验立即获得反馈值，从而可以评价所设计的模型的运行结果。神经网络开发环境是设计调试网络模型的非常有用的工具，可以大大缩短开发时间，提高工作效率。

目前神经网络的开发方法有以下 3 种模式：

(1) 人工编码 神经网络开发的一种常用模式是人工编码，即研究人员针对某个具体问题选择一种算法，然后由人工编制代码在计算机上实现一个神经网络系统。这种开发方式在前面已经介绍过。

(2) 算法库 神经网络开发的第二种模式是使用神经网络算法库。神经网络开发工具有各种有效的算法可供研究人员选择，同时神经网络的实现是自动形成的。因此，研究人员很容易实现所设计的神经网络。选择算法实际上是从算法库中提取了一个代码的结构，然后填上代码段 (fragment) 以规定网络的特性参数。

(3) 生成网络模型 神经网络开发的第三种模式是形成神经网络模型，使得研究人员能建立任何设想的网络而不仅是在库中定义的算法。

9.3.1 神经网络的开发环境及其特征

神经网络开发环境具有大多数个人计算机软件开发环境的特点，例如编辑、编译、解释、链接、库函数、跟踪调试、扩展图表、数据库、图形、字处理、通讯等。将这些工具应用于开发环境，神经网络软件开发就变得十分简单了。此外，神经网络开发环境还融入了人工智能、仿真和模型软件包的一些概念，具有模块系统提供语言和工具、动态描述、运行、数据提取、信号传送、结果分析、显示或图示结果等功能。

理想的神经网络开发环境应具有使用简单、功能强大、有效性和可扩展性等关键特征。因此开发环境应具有描述和运行网络模型的良好的用户界面，使研究人员不必掌握操作系统或实现神经网络模型的计算机硬件知识就能进行网络模型的开发。开发环境应允许研究人员选择网络模型及其特性或定义新的网络模型及其特性，应能执行、监视、显示和控制神经网络的运行，并能将网络与其他处理功能连接。有效性是指神经网络开发环境要尽可能有效地使用计算机。可扩展性意味着能定义和建立新网络类型的网络原始结构，由于有时无法预见将来需要何种网络，所以必须提供处理这种不确定性的功能。可扩展性是人工智能语言的关键特征，神经网络中同样需要这一技术。

9.3.2 MATLAB 神经网络工具箱

9.3.2.1 MATLAB 神经网络工具箱概述

MATLAB 是英文 Matrit Laboratory 的缩写，意思是矩阵实验室，是一种以矩阵为基本数据元素，面向科学计算与工程计算的高级语言。MATLAB 集科学计算、自动控制、信号处理、神经网络、图像处理等多种功能于一体，具有极高的编程效率。MATLAB 的系列产品包括 MATLAB 主包和各种工具箱（称为 TOOBOX），功能丰富的工具箱为不同领域内的研究开发者提供了一条捷径。迄今已有的 30 多个工具箱，大致可分为两类：功能型工具箱和领域型工具箱。功能型工具箱主要用来扩充 MATLAB 的符号计算功能、图形建模仿真功能、文字处理功能以及与硬件实时交互功能，可用于多种学科。而领域型工具箱是专业性很强的工具箱，每个工具箱都有一门专业理论作为背景，神经网络工具箱即属于这类工具箱。神经网络工具箱将神经网络理论中所涉及到的公式运算和操作，全都编写成了 MATLAB 环境下的子程序。设计者只要根据自己的需要，通过直接调用函数名，输入变量，运行函数，便可立即得到结果，从而大大节省了设计人员的编程和调试时间。由于 MATLAB 神经网络工具箱具有很强的专门知识要求，使用者必须首先掌握神经网络的原理和算法，然后才能够理解工具箱中每个函数的意义以及所要达到的目的和所要解决的问题，从而正确地使用工具箱很好地为自己服务。

神经网络工具箱以人工神经网络理论为基础，用 MATLAB 语言构造出典型神经网络的激活函数，使设计者对所选定网络输出的计算，变成对激活函数的调用。另外，根据各种典型的学习规则和网络的训练过程，用 MATLAB 编写出各种网络权值训练的子程序。神经网络的设计者可以根据需要调用工具箱中有关神经网络的设计与训练的程序，使自己从繁琐的编程中解脱出来，集中精力去思考问题和解决问题，从而提高效率和解题质量。

目前流行的神经网络工具箱是 Neural Networks Toolbox 2.0 版本，它几乎概括了现有的神经网络的成果，涉及的网络模型有：

- ◆ 感知器；
- ◆ 线性网络；
- ◆ BP 网络；
- ◆ 径向基函数网络；
- ◆ 自组织网络；
- ◆ 回归网络。

神经网络工具箱对各种网络模型集成了多种学习算法，为使用者提供了极大的方便。该工具箱丰富的函数使神经网络的初学者可以深刻理解各种算法的内容实质，而其强大的扩充功能更令研究人员工作起来游刃有余。此外，神经网络工具箱中还给出大量示例程序，为使用工具箱提供了生动实用的范例。

9.3.2.2 神经网络工具箱常用函数

神经网络工具箱的常用函数可按功能分为以下 10 类：

(1) 网络设计类

<code>solvehop</code>	反馈网络设计
<code>solvelin</code>	线性网络设计

(2) 转移函数类

<code>compet</code>	竞争层函数
<code>hardlim</code>	硬限幅（阶跃）函数
<code>hardlims</code>	对称硬限幅（符号）函数
<code>logsig</code>	对数 S 型函数
<code>purelin</code>	线性函数
<code>satlin</code>	饱和线性函数
<code>tansig</code>	正切 S 型函数

(3) 学习规则类

<code>earmbp</code>	反向传播学习规则
<code>learnbpm</code>	含动量项的反向传播学习规则
<code>learnh</code>	Hebb 学习规则
<code>learnhd</code>	带衰减因子的 Hebb 学习规则
<code>learnis</code>	内星学习规则
<code>learnos</code>	外星学习规则
<code>learnwk</code>	Kohonen 学习规则
<code>learnnp</code>	感知器学习规则
<code>learnwh</code>	Widrow-Hoff 学习规则

(4) 初始化类

<code>nwlog</code>	对具有对数 S 型转移函数的神经元产生 N-W 随机数
<code>mwtan</code>	对具有正切 S 型转移函数的神经元产生 N-W 随机数
<code>randnc</code>	产生归一化列随机数
<code>randnr</code>	产生归一化行随机数
<code>rands</code>	产生对称随机数

(5) 网络训练类

trainbp	采用 BP 算法的网络训练
trainbpa	带有动量项的 BP 网络训练
trainbpm	带有自适应学习率的 BP 网络训练
trainbpx	带有动量项和自适应学习率 BP 网络训练
trainc	训练竞争层
trainfm	训练特性图
trainp	采用感知器规则的训练
trainwh	采用 W-H 规则的训练
(6) 分析类	
errsurf	计算误差表面
maxlinlr	计算最大学习速率
presflop	计算浮点运算的表达式
(7) 邻域函数类	
neighb1d	一维邻域函数
neighb2d	二维邻域函数
(8) 矩阵类	
normc	计算归一化列矩阵
normr	计算归一化行矩阵
pnormc	计算伪归一化列矩阵
sumsqr	计算平方和
(9) 绘图类	
barerr	每个输出向量的条形图表
hintonw	绘制具有平方和的权值图
hintonwb	绘制具有平方和的权值和偏差的图
ploterr	绘制网络误差与训练次数的关系图
plotfa	绘制目标模式与网络函数的逼近图
plotlr	绘制网络学习速率与训练次数的关系图
plotmap	绘制具有任意邻近函数的特性图
plotpc	绘制感知器对硬限幅神经元的分类图
plotpv	绘制感知器对硬限幅神经元的训练向量图
plottr	绘制网络误差记录以及自适应学习率
plotv	绘制始于坐标原点的单位模长向量图
(10) 仿真类	
simhop	模拟 Hopfield 网络

9.3.3 其他神经网络开发环境简介

下面简要介绍其他已有的神经网络开发环境。

9.3.3.1 Plexi 神经网络开发环境

Plexi 是一个内置 Lisp 机的功能强大的综合系统，不能运行于个人计算机。该系统具有许多有价值的特点，既能支持高级用户试验新模型，又能满足初学者使用已定义的神经网络

的要求。它有两个高分辨率的显示器显示图形用户界面，图形网络编辑器能通过在层图标上击键和通过神经束连接各层来表达结构。网络的作用可通过几个图形工具（如 Hinton 图、图形和点图）来观察。学习规则、转移函数和更新规则等特性可以通过描述获得，但同时也提供缺省值。该软件包括一些标准神经网络模型，但也可以自行定义神经网络模型。只要会使用 Lisp 语言，就能设计任何神经网络。

9.3.3.2 Neuroshell 神经网络开发环境

Neuroshell 是一个有价值而又使用简便的神经网络应用工具，它具有内置反向传播模型的软件包，可以快速地开发应用程序，并有详细样例供研究、演示和实验。Neuroshelt 具有图形界面，可以对若干天的连续运行数据进行绘图和显示，用以分析趋势和因素之间的关系。

9.3.3.3 Neuralworks Professional II 神经网络开发环境

Neuralworks Professional II 是一个昂贵的综合软件包，具有图形用户界面，容易生成新的神经网络或从大量已定义的网络类型中选择所需要的神经网络，可以方便地描述层的数量和节点数量等参数。图形显示界面上能显示网络的每个神经元节点和连接，网络结构也可在图形屏幕上进行编辑。可以用标准二进制码、ASC II 文件或用户自行编写的 C 函数预处理数据训练或测试网络，能用条形图和直方图显示网络运行结果。“探针”功能可用来观察网络内部的连接权、隐含的激励和 Hinton 图等。

9.3.3.4 NETSET II 神经网络开发环境

该软件包运行于个人计算机的 Windows 环境下，用屏幕上从左到右排列的、用箭头连接的小图标代表对象。NETSET II 能良好地表示系统级的数据流程图，可以通过图标建立系统模型，图标有数据库、管道、动态数据交换、数据变换、神经网络和图形显示设备等。对象之间用反应信息流方向的箭头表示，绘制好系统流程图后，必须定义处理细节，如数据文件的输出、数据类型、预定义的格式等。描述一个神经网络对象可以从 19 个已封装的神经网络“实现”中选择。该系统有一个称为“Style”的功能，它可在任何级别由用户自行定义任何对象作为样本，命名并存储后可重复使用。

9.3.3.5 N-NET210 神经网络开发环境

N-NET210 系统只有有指导学习和无指导学习两种固定的特征算法。开发系统为应用、训练和测试网络设定参数，但不支持图形环境。“处理”的概念可简单地表示为几个网络的连接，但目前还不能对处理进行自动连接。在文本窗口编辑神经网络的处理对象时，应输入要读取和写入的文件名。通过 C 语言库函数可以将神经网络功能编入其他应用系统中去。

9.3.3.6 CaseNet 神经网络开发环境

CaseNet 是一种应用于个人计算机的神经网络开发环境，如果神经网络软件能合理地设计并加以优化，大多数类型的问题就能在个人计算机上有效地解决。CaseNet 提供了一个图形界面，设计者可在屏幕上画出网络结构并通过屏幕上的菜单输入网络特性。网络图形、特性及图上的结点构成了神经网络的特征描述，由此可自动生成可执行的编码。CaseNet 的特点是将用户定义的网络特性生成高度优化的机器码，以便最大限度地利用现有个人计算机的功能。

CaseNet 系统由网络定义器、分析器、编码生成器和编译器 4 个主要工具组成。网络定义器是设计网络结构的图形编辑工具，后处理工具将用户设计的网络图形翻译成一种标准表达形式。分析器确认网络的定义并产生代码段。编译器将代码段和通用的网络框架编译成可执行的神经网络。在神经网络生成过程中，以用中间状态文件保存各阶段的结果，以便用户修改和优化网络各阶段的内容。



10 神经网络研究展望

如果将 19 世纪末期美国心理学家 William James 发表第一部详细论述人脑结构及功能的专著，对相关学习、联想记忆的基本原理进行了开创性研究这一史实看作人工神经网络研究的启蒙，则人工神经网络的研究已经历了整整 1 个世纪。如果将 1982 年 J. J. Hopfield 发表重要论文看作掀起人工神经网络研究高潮的起点，则至今已有 25 年的历程。如果将 1990 年在北京召开首届中国神经网络学术大会看作国内吹响了向人工神经网络研究领域大规模进军的号角，则中国学术界也在这个研究方向探索了 17 年时间。经过各个国家和各个领域专家学者长期的艰辛探索，人工神经网络已获得长足发展，其结构与功能逐步改善，运行机制渐趋成熟，应用领域日益扩大，在解决各行各业的难题中显示出巨大的潜力，取得了丰硕的成果。

10.1 人工神经网络研究中的几个问题

对人工神经网络的研究进行总结、回顾和反思，对展望未来、认清今后的发展方向并确立研究工作的主攻目标具有十分重要的意义。从论文统计数字以及学术会议和刊物上发表的研究结果分析，可以看出：

(1) 人工神经网络的应用全面普及 由于人工神经网络的应用领域日益扩大，并在解决各行各业的难题中显示出巨大的潜力，越来越多的工程技术人员加入到人工神经网络的应用研究行列，表现为应用研究论文的比例最高并呈增长趋势。但是，大部分应用类论文属于验证性论文，即用人工神经网络方法完成任务并与其他方法进行对比，这类应用并没有显现出人工神经网络不可替代的优越性。

(2) 人工神经网络的模型原理及学习算法研究无重大进展 目前采用的人工神经元模型只能模拟生物神经元的神经电位脉冲电路系统，而对同样起重要作用的化学递质系统则没有涉及，因此需要研究能较全面描述生物神经元电化学机制的新模型。从学习算法来看，大多数算法具有学新忘旧的通病。因此有必要进一步研究更有效的学习算法，使神经网络具有积累知识的能力，做到在接受新信息的同时不遗忘或基本上不遗忘原有信息。

(3) 人工神经网络的一些重要理论分析课题没有获得突破性进展 例如，如何确定网络容量，如何选择隐含层节点数，如何设计初始权值以及如何避免局部极值等。长期以来，不少学者对上述问题进行了大量的研究，然而到目前为止没有一个问题找到了公认的答案。事实上，对亿万个空气分子构成的系统无法描述每个分子作不规则运动的轨迹，但可以掌握和利用该系统宏观上表现出来的大气压力。因此，对于复杂的大系统，不应指望搞清楚系统工作时各基本单元在每一时刻的具体形态。由大量神经元组成的人工神经网络正是一种复杂的非线性动态大系统，研究的重点应该是掌握和利用系统宏观表现出来的智能。

(4) 人工神经网络硬件实现研究呈递减趋势 迄今为止, 关于人工神经网络硬件实现方面的研究论文所占比例很小, 且呈递减趋势。人工神经网络技术还停留在利用 Von Neumann 计算机进行模拟仿真的起步阶段。

(5) 人工神经网络构件化 目前的研究与应用习惯于将人工神经网络看作一种具有高度并行互连结构和自适应处理能力的构件, 用作实际系统中不易进行数学建模的环节或子系统。而将人工神经网络作为一种智能体系结构方案的研究成果则甚少, 即从研究方向上未将人工神经网络研究向进一步模仿和接近生物神经系统高级智能活动方面拓展。

10.2 人工神经网络研究展望

人工神经网络的研究主要分为神经网络应用研究、神经网络实现技术研究和神经网络理论研究三个方面。在对人工神经网络的研究进行总结回顾的基础上展望未来, 该领域今后的发展主要有以下三个重要研究方向:

10.2.1 应用研究的新特点——多学科综合

应当清醒地认识到, 利用人工方法实现人脑高级智能的任务艰巨复杂, 任重道远。揭示人脑的奥妙需要各学科的交叉和各领域专家的协作, 而模拟人脑的智能更需要各学科理论、方法和技术的交叉融合。目前人工神经网络技术只是一种简单模仿人脑并行互连结构及其信息处理机制的初级工具, 必须将其与各相关学科的人工智能技术有机结合才能形成强大的综合优势。近年来, 人工神经网络技术与模糊逻辑、遗传算法及专家系统相结合, 取得的应用成果明显优于各种智能技术单兵作战的结果。随着智能科学的发展, 今后人工神经网络技术将会与更多相关的智能技术相结合, 取长补短, 相辅相成, 不断提高智能水平。人工神经网络可以结合的相关理论和技术主要有: 粗集理论、优化理论、算法复杂性理论、非线性理论, 混沌与分形以及区组设计等技术。

10.2.2 实现技术研究的当务之急——神经网络的硬件实现

人工神经网络的实现技术可以分为全硬件实现和虚拟实现两个方面: 全硬件实现的人工神经网络称为神经计算机, 它是根据神经网络结构及其计算特点, 用电子器件、光学器件及分子或化学器件构成的计算系统; 虚拟实现包括用 Von Neumann 计算机进行神经网络的模拟仿真、神经计算的多机并行技术以及神经网络加速器等技术。由于采用了软件, 虚拟实现具有较好的通用性和灵活性, 适于进行神经计算体系结构的研究。

目前在人工神经网络的应用中主要采用软件模拟的方法, 由于该方法是用串行计算体系模拟并行计算体系, 因而无法真正发挥神经网络并行处理的优势。硬件实现的神经网络中每个处理单元以及各处理单元之间的连接均有对应的物理器件, 因而其最大优点是能够充分体现并行系统的固有优越性。在神经网络技术的发展过程中, 硬件实现技术的研究远滞后于应用研究, 以至于当前绝大部分应用不得不采用串行计算机进行软件模拟, 几十年来, 应用人工神经网络的人们尚未有机会真正体会到基于神经网络的并行结构的种种优越性。

10.2.3 理论研究的新方向——从人工神经网络到人工神经系统

这是未来人工神经网络研究中一个富有挑战性的发展方向。人类的大脑大约有 1.4×10^{11} 个神经元, 每个神经元有数以千计的通道同其他神经元广泛相互连接, 形成复杂的生物神经网络。生物神经网络以神经元为基本信息处理单元, 对信息进行分布式存储与加工, 这

种信息加工与存储相结合的群体协同工作方式使得人脑呈现出目前计算机无法模拟的神奇智能。至今为止，人工神经网络的理论研究主要着眼于对人脑生物神经网络的信息存储和加工处理机制的模拟，而很少考虑其群体协同工作方式。事实上，由巨量神经细胞的广泛互连形成的人脑生物神经网络是以神经系统的形式进行群体协同工作的。当强调大量神经元的“网络”属性时，必然会关注其连接方式，从而出现具有各种网型的人工神经网络（如：多层次前馈网、全互连反馈网等）。当强调大量神经元的“系统”属性时，自然要研究构成系统的方式及系统中各组成环节的关系，从而可望出现具有各种特定功能的人工神经系统。

半个多世纪以来，人工神经网络作为一种脑式信息处理系统，对脑的工作原理的模拟一直停留在细胞水平上，因此基于人工神经元的经典人工神经网络模型只能模拟简单的、初步的脑的信息处理功能。正如脑的高级功能的研究不可能期待通过细胞和分子事件的阐述来解决一样，人工脑或人工智能信息处理系统的高级目标也不可能期待通过现有人工神经元模型的拓扑变化和学习规则变化得到解决。一种可行的途径是：在现有细胞层次的基础上，从不同系统层次模拟脑功能，建立不同系统水平的人工神经系统。这种人工神经系统不是基于细胞模拟的人工神经网络的规模扩大，而是对不同层次生物神经系统的直接借鉴和模拟。随着脑科学的研究在系统水平上不断突破，这一研究途径将不断从中获更加深刻的认识和启发；反之，人工神经系统的理论、模型和应用成果也将对脑科学从系统层次上认识和揭示脑的高级功能提供宝贵的思路。



附录 1 常用神经网络 C 语言源程序

程序名称	起始行号
01. Single Bipolar Perceptron Dichotomizer	1~70
02. Discrete, Bipolar Discrete Perceptron Layer	72~169
03. Bipolar Single Layer Delta Learning Network	171~266
04. Bipolar Backpropagation Training	268~446
05. Discrete Bipolar Hopfield Associative Memories	448~525
06. Discrete Bipolar Bidirectional Associative Memory	528~611
07. Hamming/Maxnet Training	613~717
08. Winner Take All Training	720~836
09. Kohonen Feature Mapping	838~964
10. Art1 Training	966~1128
11. K-means Training	

```

25 repeat
26   for i:=1 to leninvecs+1 do weightvecs[1][i]:= 2.0 * random-1.0;
27   write('Enter learning constant: ');
28   readln( alphla );
29   linect:= linect + 1;
30   alpha:= -abs(alpha); {force atpha neg for later use}
31   iter:= 0 ;
32   cLear_work_area;
33   print_fkey_footings;
34   dichot_output(false) ;
35   if not abort thv
36   repeat
37     iter:= iter + 1;
38     finiehed:= true; {assume a successful training cycle}
39     for i:= 1 to numinvecs do {for each input vector ... }
40       begin
41         sum:= 0.0; {... calc scalar product}
42         for j:= 1 to leninvecs+1 do
43           sum:= snm + weightvecs [1][j] * inputvecs[i][j] ;
44         if sgn(sum) <> outputvecs[i][1] then {... if not correct sign}
45           begin {... cycle ts not success}
46             finished:= false;
47             for j:=1 to leninvecs+1 do {... adjust weights +/-}
48               weightvecs[1][j]:= weightvecs[1][j] +
49                             alpha * sgn(sum) * inputvecs[i][j] ;
50             end;
51           end;
52         dichot_output(finished);
53       until finished or abort or quittrain;
54       if not abort then
55         while not abort and not quittest do {allow testing of new patts}
56           begin
57             writeln;
58             write('Enter a test vector of dimension', Leninvecs,':');
59             for i:= 1 to leninvecs do read(testvec[i] );
60             readln;
61             testvec [leninvecs+1]:= 1.0;
62             sum:= 0.0;
63             for i:= 1 to leninvecs+1 do
64               sum:= sum + weightvecs[1][i] * testvec[i] ;
65             if sgn(sum) < 0.0 then writeln('Classified as pattern 1')
66                           else writeln('Classifited as pattern 2');
67             hitakey;
68           end;
69         until quitruns;
70       end ;
71
72

```



```

121      finished:= true;           {assume no weight changes required}
122      for i:= 1 to numinvecs do {for all training vectors do}
123          begin
124              for j:= 1 to numclasses do {caLc scaLar prod and perceptron .. }
125                  begin                 {.. outputs}
126                      net[j]:= 0.0;
127                      for k:= 1 to leninvecs + 1 do
128                          net[j]:= net[j] + weightvecs[k][j] * inputvecs[i][k];
129                      fnet[j]:= sgn(net[j]);
130                  end;
131                  for j:= 1 to numclasses do {f any output not as desired .. }
132                      if fnet[j] <> outputvecs[i][j] then
133                          begin
134                              finished:= false;    {... not trained, update weights}
135                              for k:= 1 to lenivecs + 1 do
136                                  weightvecs[k][j]:= weightvecs[k][j] +
137                                              fnet[j] * alpha * inputvecs[i][k]
138                          end;
139          end ;
140          dpl_output(finished) ;
141      until abort or finished or quittrain;
142      if not abort then
143          while not abort and not quittest do
144              begin
145                  writeln;
146                  write('Enter a test vector of dimension',leninvecs,':');
147                  for i:= 1 to leninvecs do read(testvec[i]);
148                  readln;
149                  testvec[leninvecs+1]:= 1.0;
150                  for i:= 1 to numclasses do
151                      begin
152                          net[i]:= 0;
153                          for j:= 1 to leninvecs + 1 do
154                              net[i]:= net[i] + weightvecs[j][i] * testvec[j];
155                          fnet[i]:= sgn(net[i]);
156                      end;
157                      k:= 0;
158                      for i:= 1 to numclasses do
159                          if fnet[i] > 0.0 then
160                              begin
161                                  k:= k + 1;
162                                  j:= i;
163                              end;
164                          if k = 1 then writeln('Classified es pattern',j)
165                                  else writeln('Not Classified');
166                          hitakey;
167                      end;
168      until quitruns;
169 end;
170

```



```

218     delta_output( false );
219     if not abort then
220     repeat
221         iter:= iter + 1;
222         error:= 0.0;
223         for i:= 1 to numinvecs do {each iter one training cycle}
224             begin
225                 for k:= 1 to numclasses do
226                     begin
227                         net[k]:= 0.0;           {calc scalar prod and output}
228                         for j:= 1 to leninvecs+1 do
229                             net[k]:= net[k] + weightvecs[j][k] * inputvecs[i][j];
230                         ok[k]:= fnet(net[k]);
231                         error:= error + sqr(ok[k] - outputvecs[i][k])/Z_0;
232                         sigma:= alpha * (outputvecs[i][k]) * (1-sqr(ok[k]))/2;
233                         for j:= 1 to leninvecs+1 do {always adjust weights}
234                             weightvecs[j][k]:= weightvecs[j][k]+ sigma * inputvecs[i][j];
235                         end;
236                     end ;
237                     finished:= error <= term_error;
238                     delta output(finished);
239                 until abort or finished or quittrain;
240                 if not abort then {submit vecs for classification}
241                     while not abort and not quittest do
242                         begin
243                             writeln;
244                             write('Enter a test vector of dimension' ,lenivecs,':');
245                             for i:= 1 to leninvecs do read(testvec[i]);
246                             readln;
247                             testvec[leninvecs+1]:= 1.0;
248                             for i:= 1 to numclasses do
249                                 begin
250                                     net[i]:= 0.0;;
251                                     for j:= 1 to leninvecs+1 do
252                                         net[i]:= net[i] + weightvecs[j][i] * testvec[j];
253                                     end;
254                                     k:= 0;
255                                     for i:= 1 to numclasses do
256                                         if fnet(net[i]) > 0.5 then
257                                             begin
258                                                 k:= k+ 1;
259                                                 j:= i;
260                                             end ;
261                                         if k = 1 then writeln('Classified as pattern',j)
262                                             else writeln('Not Classified');
263                                         hitakey;
264                                     end ;
265                 until quitruns;
266 end;

```

```
267
268 { * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * }
269 { * * * BIPOLAR BACKPROPAGATION TRAINING * * * * * * * * * * * * * * * * * * * * * * * * }
270
271 procedure backprop_headings ;
272 begin
273     check_key;
274     gotoxy(1,4);
275     textbackground(7) ; textcolor(0) ;
276     write('Training Cycle' ,iter,' : Outputs');
277     textbackground(0) ; textcolor(7);
278     writeln(' (rows,cols) = ( input vectore,neurons)' );
279     writeln;
280     linect:= 6;
281 end;
282
283 procedure backprop_outline(vecnum: integer;
284                               vec: vector;
285                               finished: boolean);
286 var i,j,lines: integer;
287 begin
288     lines:= (numclasses div 11) + 1;
289     if screen full(lines) then
290         begin
291             clear_work_area;
292             backprop_headings ;
293         end
294     else if vecnum = 1 then beckprop_headings;
295     for j:= 1 to numclasses do write(vec[j]:7:3,"");
296     writeln;
297     linect:= linect + lines;
298     if vecnum = numinvecs then
299         begin
300             if screen_full(2) then backprop_headings;
301             writeln;
302             writeln('Error =',error:8:4);
303             linect:= linect + 2;
304             write_outputend( finished) ;
305         end ;
306 end;
307
308 procedure backprop_outweights( w: wmatrix;
309                                 rows,cols, layer: integer);
310 var i,j,lines: integer;
311 begin
312     clear_work_area;
313     textbackground(7); textcolor(0);
```

```

314 if layer = 1 then write('Final hidden layer weights')
315           else write('Final output layer weights');
316 textbackground(0); textcolor(7);
317 if layer = 1 then writeln(' (row, column) = (hidden,input)')
318           else writeln(' (row, column) = (output,hidden)' );
319 writeln;
320 linect:= linect + 2;
321 lines:= (cols div 11) + 1;
322 for i:= 1 to rows do
323 begin
324   if screen_full( lines) then clear_work_area;
325   for j:= 1 to cols do write(w[i][j]:7:3,"");
326   writeln;
327   linect:= linect + lines;
328 end;
329 hitakey;
330 end;
331
332 procedure backprop;
333 var sum: real;
334   o1, o2,
335   delta_j ,
336   delta_k: vector;
337   i,j,k: integer;
338   numhiddenodes: integer;
339 begin
340   load_patterns( 1 ) ;
341   repeat
342     write('Enter # of hidden nodes:');
343     readln(numhiddenodes);
344     write('Enter learning constant:');
345     readln(aLpha);
346     write('Enter termination error:');
347     readln( term_error) ;
348     iter:= 0;
349     for i:= 1 to numinvecs do
350       intputvecs[i] [leninvecs+1]:= 1.0;
351     for i:= 1 to max_vec_len do
352       for j:= 1 to max_vec_len do
353         begin
354           weightvecs[i] [j]:= 2.0 * random - 1.0;
355           weight2vecs[i] [j]:= 2.0 * random - 1.0;
356         end ;
357     clear_work_area;
358     backprop_headings;
359     print_fkey_footings;
360     repeat
361       iter:= iter + 1;
362       error:= 0.0;

```

```

363   for i:= 1 to numinvecs do
364     begin
365       for j:= 1 to numhiddennodes do
366         begin
367           sum:= 0;
368           for k:= 1 to leninvecs+1 do
369             sum:= sum + weightvecs[j] [k] * inputvecs[i] [k];
370             o1 [j]:= fnet(sum);
371           end;
372           o1 [numhiddennodes+1]:= 1.0;
373           for j:= 1 to numclasses do
374             begin
375               sum:= 0;
376               for k:= 1 to numhiddennodes+1 do
377                 sum:= sum + weight2vecs[j] [k] * o1 [k];
378                 o2[j]:= fnet(sum);
379                 error:= error + sqr(o2[j]-outputvecs[i] [j])/2;
380               end;
381               for j:= 1 to numclasses do
382                 delta_k[j]:= (outputvecs[i][j]-o2[j]) * (1-sqr(o2[j]))/2;
383               for j:= 1 to numhiddennodes+1 do
384                 begin
385                   sum:= 0;
386                   for k:= 1 to numclasses do
387                     sum:= sum + delta_k[k] * weight2vecs[k] [j];
388                     delta_j[j]:= (1 - sqr(o1[j])) * sum/2;
389                   end;
390                   for j:= 1 to numclasses do
391                     for k:= 1 to numhiddennodes+1 do
392                       weight2vecs[j] [k]:= weight2vecs[j] [k] +
393                                     alpha * delta_k[j] * o1[k];
394                     for j:= 1 to numhiddennodes+1 do
395                       for k:= 1 to leninvecs+1 do
396                         weightvecs[j] [k]:= weightvecs[j] [k] +
397                           alpha * delta_j[j] * inputvecs [i] [k];
398                         finished:= error <= term_error;
399                         backprop_outline(i,o2, finished);
400                       end;
401 until abort or finished or quittrain;
402 if not abort then
403   begin
404     output_mode:= screenoutput; {force to view each screenful}
405     backprop_outweights(weightvecs,numhiddennodes, leninvecs+1, 1 );
406     backprop_outweights(weight2vecs,numclasses, numhiddennodes+1, 2)
407   end ;
408 if not abort then
409   while not abort and not quittest do
410     begin
411       writeln;

```

218 人工神经网络理论、设计及应用

```

459  linect:= 6;
460  lines:=(leninvecs div 21) + 1;
461  for i:= 1 to leninvecs do
462    begin
463      if screen_full(lines) then clear work area;
464      for j:= 1 to leninvecs do
465        write(trunc(weightvecs[i] [j]),3,"");
466        writeln;
467      linect:= linect + lines;
468    end ;
469    write_outputend(finished) ;
470 end;
471
472 procedure hopfield;
473 var i, j, k, numrelax: integer;
474   sum: real;
475 begin
476   load_patterns(1 );
477   hitafey;'
478   clear_work_area;
479   print_fkey_footings;
480   for i:= 1 to leninvecs do
481     for j:= 1 to leninvecs do
482       weightvecs[i] [j]:= 0; {init weights zero}
483     iter:= 0; {simple training algorithm}
484     repeat
485       iter:= iter + 1;
486       for j:= 1 to leninvecs do
487         for k:= 1 to leninvecs do
488           if (j<>k) then weightvecs[j] [k]:= weightvecs[j] [k] +
489                         (inputvecs [iter] [j] * inputvecs [iter] [k]);
490       finished:= iter = numinvecs;
491       hopfield_output(finished);
492     until abort or finished or quittrain;
493     if not abort then {submit vectors to classify}
494       while not abort and not quittest do
495         begin
496           writeln;
497           writeln('Enter a discrete, bipolar test vector of dimension',
498                 leninvecs,';');
499           for i:= 1 to leninvecs do read(testvec[i]);
500           readln;
501           for i:= 1 to leninvecs do outputvecs[l] [i]:= 0;
502           i:= 0;
503           repeat
504             i:= i + 1;
505             for j:= 1 to leninvecs do
506               begin
507                 sum:= 0.0;

```

```

508         for k:= 1 to leninvecs do
509             sum:= sum + weightvecs[j] [k] * testvec[k] ;
510             testvec[j]:= sgn(sum);
511             end;
512             finished:= true;
513             for j:= 1 to leninvecs do
514                 begin
515                     if testvec[j] <> outputvecs[1] [j] then finished:= false;
516                     outputvecs[1] [j]:= testvec[j];
517                     end;
518             until finished or (i > 100);
519             writeln;
520             writeln('Output Vector:');
521             for i:=1 to leninvecs do write(trunc(outputvecs[1] [i]),"");
522             writeln;
523             hitakey;
524             end;
525 end;
526
527
528 { * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * }
529 { * * * DISCRETE BIPOLAR BIDIRECTIONAL ASSOCIATIVE MEMORY * * * * * * * * * * * }
530
531 procedure bam_output(finished: boolean);
532 var i,j,lines: integer;
533 begin
534     check_key;
535     gotoxy(1,4);
536     textbackground(7); textcolor(0);
537     write('Training vector',iter,',: weight Matrix W');
538     textbackground(0); textcolor(7);
539     writeln('(rows,cols) = (inputs,neurons)');
540     writeln;
541     linect:= 6;
542     lirtes:= (numclasses div 11) + 1;
543     for i:= 1 to leninvece do
544         begin
545             if screen_full(lines) then clear_work_area;
546             for j:= 1 to numclassee do
547                 write(trunc(weightvecs[i] [j]):3,"");
548             writeln;
549             linect:= linect + lines;
550         end ;
551     write_outputend( finished) ;
552 end;
553
554
555 procedure bam;

```

```

556 var i,j,k: integer;
557   a       : vector;
558 begin
559   load_patterns( 1 );
560   hitakey;
561   clear_work_area;
562   print_fkey_footings;
563   for i:= 1 to leninvecs do
564     for j:= 1 to numclasses do
565       weightvecs[i][j]:= 0;
566   iter:= 0;
567   repeat
568     iter:= iter + 1;
569     for j:= 1 to teninvecs do
570       for k:= 1 to numclasses do
571         weightvecs[j][k]:= weightvecs[j][k] +
572           inputvecs * [iter][j] * outputvecs [iter][k];
573     finished:= iter = numivnvecs;
574     bam_output(finished);
575   until abort or finished;
576   if not abort then
577     while not abort and not quittest do
578       begin
579         writeln;
580         writeln('Enter a discrete, bipolar test vector of dimension',
581               leninvecs,':');
582         for i:= 1 to leninvecs do read(testvec[i]);
583         readln;
584         for i:= 1 to leninvecs do a[i]:= testvec[i];
585         repeat
586           for i:= 1 to numcLasses do
587             begin
588               outputvecs[1][i]:= 0;
589               for j:= 1 to leninvecs do
590                 outputvecs[1][i]:= outputvecs[1][i]+a[j] * weightvecs[j][i];
591               outputvecs[1][t]:= sgn(outputvecs[1][i]);
592             end;
593           finished:= true;
594           for i:= 1 to leninvwcs do
595             begin
596               outputvecs[2][i]:= 0;
597               for j:= 1 to numclasses do
598                 outputvecs [2][i]:= outputvecs[2][i] +
599                   outputvecs [1][j] * weightvecs ;
600               outputvecs [2][i]:= sgn(outputvecs[2][i]);
601               if outputvecs[2][i] <> testvec[i] then finished:= false;
602               testvec[i]:= outputvecs[2][i];
603             end;
604           until finished;

```

```

605      writeln;
606      writeln('Output Vector:');
607      for i:= 1 to numclasses do write(trunc(outputvecs[1] [i]),");
608      writeln;
609      hitakey;
610      end;
611 end;
612
613 { * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * }
614 { * * * HAMMING/MAXNET TRAIUIUG * * * * * * * * * * * * * * * * * * * * * * * }
615
616 procedure hamming_outweights; {output weights used for training}
617 var i, j: integer;
618 begin
619   check_key;
620   clear_work_erea;
621   textbackground(7); textcolor(0);
622   writeln('Maxnet Weights');
623   textbackground(0); textcolor(7) ;
624   writeln;
625   for i:= 1 to numinvecs do
626     begin
627       for j:= 1 to numinvecs do write(weightvecs[i] [j]:7:3,");
628       writeln;
629     end;
630   linect:= linect + numinvecs + 1;
631   hitakey;
632 end;
633
634 procedure hamming_output(vec: vector;
635                           patt: integer;
636                           finished: boolean);
637 var i: integer;
638 begin
639   check_key;
640   gotoxy( 1 ,4);
641   textbackground(7); textcolor(0);
642   write('Classification Step',iter,' outputs');
643   textbackground(0); textcolor(7);
644   writeln(' (rows) = (input vecs)');
645   writeln;
646   linect:= 6;
647   for i:= 1 to numinvecs do writeln(vec[i]:7:3);
648   if output_mode = screenoutput then hitakey;
649 end;
650
651 procedure hamming;

```

```

652 var net,                      {current similarity value for each class}
653     oldnet,                     {previous neuron outputs for comparison}
654     fnet      : vector;        {current class (neuron) output for each}
655     i,j,                         {misc counters}
656     patt,                        {current cluster matched with}
657     num nonzero: integer;       {number nonzero outputs in current iteration}
658     temp   : real;             {misc vaLue}

659 begin
660     load_patterns( 2 ) ;
661     repeat
662         write('Enter the learning constant (less than' ,
663                 (1/numinvecs),6,3,'):');
664         readln( alpha) ;
665         write('Enter a discrete, bipolar test vector of dimension',
666                 leninvecRs,':');
667         for i:= 1 to leninvecs do read(testvec[i] );
668         readln;
669         linect:= linect + 2;
670         for i:= 1 to numinvecs do {create weight matrix}
671             for j:= 1 to numinvecs do
672                 if i = j then weightvecs[i][j]:= 1
673                 else weightvecs[i][j]:= -alpha;
674         hamming_outweights; {output weight matrix}
675         if not abort then
676             begin
677                 for i:= 1 to numinvecs do {scalar prod of each input vec}
678                     begin
679                         temp:= 0;
680                         for j:= 1 to leninvecs do
681                             temp:= temp + inputvece[i][j] * testvec [j] ;
682                         net[i]:= (temp + leninvecs)/2;
683                         fnet[i]:= nete[i]/leninvecs;
684                     end;
685                     iter:= 0;
686                     clear_work_area;
687                     hamming_output(fnet , 0, false) ;
688                 end;
689                 if not abort then
690                     repeat {until all but one output = 0}
691                         num nonzero:= 0; {assume all outputs > 0}
692                         iter:= iter + 1;
693                         for i:= 1 to numinvecs do oldfnet[i]:= fnet[i]; {save outputs}
694                         for i:= 1 to numinvecs do
695                             begin
696                                 net[i]:= 0; {calc new scalar prod outputs}
697                                 for j:= 1 to numinvecs do
698                                     net[i]:= net[i] + weightvecs[i][j] * oldfnet[j] ;
699                                     fnet[i]:= ramx(net[i] ,0.0);
700                                     if abs(fnet[i]) > 0.001 then {see how many > 0 and save}

```

```

701      begin
702          num nonzero := num nonzero + 1 ;
703          patt := i;
704          end;
705      end;
706      finished := num nonzero <= 1;
707      hamming_output(fnet, patt, finished); {display new outputs}
708      until finished or abort or quittrain;
709      if not abort then
710          begin
711              writeln;
712              writeln('Training Complete');
713              if num nonzero = 1 then writeln('CLassified as pattern',patt)
714                  else writeln('Not classified.');
715          end;
716      until quitrungs;
717 end;
718
719
720{ **** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * }
721{ *** WINNER TAKE ALL TRAINING * * * * * * * * * * * * * * * * * * * * * * * * * }
722
723 procedure wta_output(finished: boolean);
724 var i,j,k,lines: integer;
725 begin
726     check_key;
727     gotoxy(1 ,4);
728     textbackground(7); textcolor(0);
729     write('Training Step',iter,' : Weight Matrix');
730     textbackground(0); textcolor(7);
731     writeln(' (rows,cols) = (neurons, inputs)');
732     writeln('Alpha =' ,alpha:0:4);
733     writeln;
734     linect := 7;
735     lines := (leninvecs div 11)+1; {lines for one neuron weights}
736     for i:= 1 to numclasses do
737         begin
738             if screen_full(lines) then clear_work_area;
739             for j:= 1 to leninvecs do write(weightvecs[i] [j]:7:3,"");
740             writeln;
741             linect := linect + Lines;
742         end ;
743     write_outputend( finished) ;
744 end;
745
746 procedure normalize_vector(var vec: vector;
747                               len: integer);

```

```

748 var i: integer;
749     temp: real;
750 begin
751     temp:= 0;
752     for i:= 1 to len do temp:= temp + sqr(vec[i]);
753     temp:= sqrt(temp);
754     if temp <> 0 then
755         for i:= 1 to len do vec[i]:= vec[i]/temp;
756 end;
757
758 procedure winnertakeall;
759 var i,j,k,                      { misc counters}
760     successes,                  { number of consecutive successful training iters}
761     minneur: integer;           { vector with max scalar product in curr iter}
762     temp: real;                 { misc value}
763     success: boolean;          { is current iter a success}
764     dist: vector;
765 begin
766     load_patterns( 2 );
767     repeat
768         write('Enter number of classes:');
769         readln( numclasses );
770         write('Enter similarity error limit:');
771         readln(term error);
772         for i:= 1 to numinvecs do normalize_vector(inputvecs[i],leninvecs);
773         for j:= 1 to numclasses do {weights random on [-1,1] , then normalized}
774             for k:= 1 to leninvecs do
775                 weightvecs[j][k]:= 2.0 * random -1.0;
776             for j ,= 1 to numclasses do normalize_vector(weightvecs[j] , leninvecs);
777             iter:= 0;
778             successes:= 0;
779             i:= 0;
780             clear_work_area;
781             print_fkey_footings;
782             alpha:= 0.7;
783             wta_output( false );
784             if not abort then
785                 repeat                      { til abort or complete cycle of success}
786                     iter:= iter + 1;
787                     alpha:= 0.7 * exp(-iter/1000);
788                     if i = numinvecs then i:= 1
789                         else i:= i + 1;
790                     for j:= 1 to numclasses do
791                         begin
792                             dist[j]:= 0.0;
793                             for k:= 1 to leninvecs do
794                                 dist[j]:= dist[j] + sqr(inputvecs[i][k] -weightvecs[j][k] );
795                         end ;
796                     minneur:= 1;

```



```

844 var i,j,k,lines: integer;
845 begin
846   check_key;
847   gotoxy( 1 ,4);
848   textbackground(7); textcolor(0);
849   write('Training Step',iter,' : weights');
850   textbackground(0); textcolor(7) ;
851   writeln(' (rows,cols) = (inputs,neurons)');
852   writeln('Neighborhood =',Nrad,', Alpha =',alpha:7:3,
853           ', Total Adjustment =',adj:10:3);
854   writeln;
855   linect:= 7;
856   lines:=(leninvecs div 11) + 1; {lines for one neuron weights}
857   for i:= 1 to leninvecs do {for all neurons}
858     begin
859       if screen_full(lines) then clear_work_area;
860       for j:=1 to numclasses do write(weightvecs[j] [i]:7:3,"");
861       writeln;
862       linect:= linect + lines;
863     end ;
864   write_outputend( finished) ;
865 end;
866
867 procedure featuremap;                                {feature nnp training}
868 var
869   i,j,k,m, indx,                                     {misc counters and array indices}
870   minneur,                                         {neuron with minimum dist inputs to weights}
871   Nrad,                                              {curr radius of neighborhood for wvec adjust}
872   alphadectime,                                     {time at which alpha starts to decline}
873   Ndecint,                                           {interval at which Nrad continues to decline}
874   successes,                                         {# of consecutive successes current cycle}
875   rows, cols : integer; {dimensions of feature mapping grid}
876   dist       : vector;    {distance weights to inputs at curr iter}
877   adjust,                                            {adj to indiv neur weights during current iter}
878   totadjbst: real; {tot adj to all neur weights during curr iter}
879   success  : boolean;{is curr input a success}
880 begin
881   load_patterns(2);
882   repeat
883     write('Enter # rows in rectangular feature space (1 for 1-D):');
884     readln( rows );
885     write('Enter # cols in rectangular feature space:');
886     readln(cols);
887     write('Enter total adjustment limit:');
888     readln( term_error );
889     write('Enter iterations before begin reducing alpha:');
890     readln(alphadectime);
891     write('Enter iterations between neighborhood reductions:');
892     readln(Ndecint);

```



```

942     fmap_output(nrad, totadjust, finished)
943     until abort or finished or quittrain;
944     it not abort then
945         whiLe not abort and not quittest do
946             begin
947                 writeln;
948                 writeln('Enter a test vector of dimension',leninvecs,':');
949                 for i:= 1 to leninvecs do read(testvec[i]);
950                 readln;
951                 for i:= 1 to numclasses do
952                     begin
953                         dist[i]:= 0.0;;
954                         for j:= 1 to leninvecs do
955                             dist[i]:= dist[i] + sqr(testvec[j] - weightvece[i][j]);
956                         end;
957                         minneur:= 1;
958                         for i:= 2 to numclasses do
959                             if dist[i] < dist[minneur] then minneur:= i;
960                         writeln('Classified as pattern' ,minneur);
961                         hitakey;
962                         end ;
963         until quiruns;
964 end;
965
966 { * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * }
967 { * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ART1 TRAINING * * * * * * * * * * * }
968
969 procedure artl_headings(patt: integer;
970                           thresh: real); {display output headings}
971 begin
972     cheek_key;
973     clear_work_area;
974     textbackground(7) ; textcolor(0) ;
975     write('Input Pattern',patt,' of',numinvecs);
976     textbackground(0) ; textcolor(7);
977     writeln(' Threshold =' ,thresh:7:3);
978     writeln;
979     linect:= linect + 2;
980 end;
981
982 procedure artl_outweights( patt,neur: integer;
983                             thresh: real);
984 var i,j,k, lines: integer;
985 begin
986     check_key;
987     lines:= (leninvecs div 11) + 1;
988     if screen_full( lines+1 ) then artl_headings(patt, thresh);

```

```

989 writeln('weight Vector W, Neuron' ,neur,':');
990 for i:= 1 to leninvecs do write(weightvecs[neur] [i]:7:3,"");
991 writeln;
992 linect:= linect + lines + 1;
993 if screen_full(lines+1) then art1_headings(patt,thresh);
994 writeln('Weight Vector V, Neuron' ,neur,':');
995 for i:= 1 to leninvecs do write(weight2vecs[neur] [i]:7:3,"");
996 writeln;
997 linect:= linect + lines + 1;
998 hitakey;
999 end;
1000
1001 procedure art1 ;
1002 var
1003   fnet      : vector; {current scalar prod inputs to each neuron}
1004   disabled   : array[1.. max_vec_len] of boolean;
1005                   {neur discarded as choice for current pattern?}
1006   i,j,k,       {misc counters}
1007   maxneuron,    {curr neuron considered with max output}
1008   maxusedneuron, {curr last neuron assigned to a class}
1009   numdisabled; integer; {num neurs discarded in classifying curr pattern}
1010   temp,
1011   thresh,        {user supplied threshold value}
1012   maxval,        {output value of maxneuron}
1013   simtest,       {scalar product of revweights and curr pattern}
1014   meg      : real; {number of 1's in current pattern}
1015   finisbed     : boolean; {finished training}
1016 begin
1017   load_patterns(2) ;
1018   repeat
1019     write('Enter vigilance threshold on (0,1):');
1020     readln(thresh);
1021     maxusedneuron:= 0; {so far no classifications made}
1022     for i:= 1 to max_neurons do {init wvecs as in text}
1023       for j:= 1 to leninvecs do
1024         begin
1025           weightvecs[i] [j]:= 1/(leninvecs+1);
1026           weight2vecs[i] [j]:= 1;
1027         end;
1028         i:= 1; {first invec auto class one}
1029         art1_headings( i , thresh);
1030         print_fkey_footings;
1031         writeln('Automatically mapped to pattern one' );
1032         linect:= linect + 1;
1033         simtest:= 0; {calc rev scalar prod neur one}
1034         for j:= 1 to leninvecs do
1035           simtest:= simtest + weight2vecs[i] [j] * inputvecs[i] [j] ;

```



```

1085         weight2vecs[maxneuron][j]:= inputvecs[i][j] *
1086             weight2vecs[maxneuron][j];
1087         end;
1088         arti_outweights(i,maxneuron,thresh);
1089     end
1090     else {failed threshold test, try remaining eligibles or ...}
1091     begin      {... create new class}
1092         writeln('Failed');
1093         linect:= linect + 1;
1094         disabled[maxneuron]:= true;
1095         numdisabled:= numdisabled + 1;
1096         if maxusedneuron >= max_neurons then {exceed Pgm
constraints}
1097         begin
1098             writeln('# of needed classes exceeds program maximum');
1099             linect:= linect + 1;
1100         end
1101     else
1102         if numdisabled < maxusedneuron then finished:= false
1103         else {form a new class}
1104         begin
1105             maxusedneuron:= maxusedneuron+ 1;
1106             maxneuron:= maxusedneuron;
1107             writeln('Adding as new pattern'+chr($30+maxneuron));
1108             linect:= linect + 1;
1109             temp:= 0;
1110             for j:= 1 to leninvecs do
1111                 temp:= temp+ weight2vecs[maxneuron][j] * inputvecs[i][j];
1112                 for j:= 1 to leninvecs do
1113                     begin
1114                         if inputvecs[i][j] = 1 then {adjust new neur
weights}
1115                             weightvecs[maxneuron][j]:= =
1116                             (weight2vecs[maxneuron][j] * inputvecs[i][j])/
1117                             (0.5+ temp);
1118                             weight2vecs[maxneuron][j]:= inputvecs[i][j] *
1119                             weight2vecs [maxneuron] [j];
1120                         end;
1121                         art1_outweights(i,maxneuron,thresh);
1122                     end;
1123                 end;
1124             until finished or abort;
1125         end;
1126         if not abort then writeln('Classification Complete');
1127         until quitruncs;
1128 end;

```



```

        strcat(cbuf, cp);
    } /* endif */
} /* endif */
cp=&cbuf[0];
return cp;
}

// ***** Defined structures & classes *****
struct aCluster {
    double      Center[MAXVECTDIM];
    int         Member[MAXPATTERN]; //Index of Vectors belonging to this cluster
    int         NumMembers;
};

struct aVector {
    double      Center[MAXVECTDIM];
    int         Size;
};

class System {
private:
    double      Pattern[MAXPATTERN][MAXVECTDIM+1];
    aCluster    Cluster[MAXCLUSTER];
    int         NumPatterns;           // Number of patterns
    int         SizeVector;          // Number of dimensions in vector
    int         NumClusters;         // Number of clusters
    void        DistributeSamples(); // Step 2 of K-means algorithm
    int         CalcNewClustCenters(); // Step 3 of K-means algorithm
    double      EucNorm(int, int);   // Calc Euclidean norm vector
    int         FindClosestCluster(int); //ret indx of clust closest to pattern
                                      //whose index is arg
public:
    system();
    int LoadPatterns(char * fname);      // Get pattern data to be clustered
    void InitClusters();                // Step 1 of K-means algorithm
    void RunKMeans();                  // Overall control K-means process
    void ShowClusters();                // Show results on screen
    void SaveClusters(char * fname);    // Save results to file
    void ShowCenters();
};

void System::ShowCenters(){
int i,j;
printf("Cluster centers:\n");
for (i=0; i<NumClusters; i++) {
    Cluster[i].Member[0]=i;
    printf("ClusterCenter[%d]=(%f,%f)\n",i,Cluster[i].Center[0],Cluster[i].Center[1]);
} /* endfor */
printf("\n");
}

```

```

}

int System::LoadPatterns(char * fname){
    FILE * InFilePtr;
    int i,j;
    double x;
    if((InFilePtr = fopen(fname, "r")) ==NULL
        return FAILURE;
    fscanf(InFilePtr, "%d", &NumPatterns); // Read # of patterns
    fscanf(InFilePtr, "%d", &SizeVector); // Read dimension of vector
    fscanf(InFilePtr, "%d", &NumClusters); // Read # of clusters for K-Means
    for (i=0; i<NumPatterns; i++) { // For each vector
        for (j=0; j<SizeVector; j++) { // create a pattern
            fscanf(InFilePtr, "%lg", &x); // consisting of all elements
            Pattern[i][j]=x;
        } /* endfor */
    } /* endfor */
    printf("Input patterns:\n");
    for (i=0; i<NumPatterns; i++) {
        printf("Pattern[%d]=(%2.3f,%2.3f)\n",i,Pattern[i][0],Pattern[i][1]);
    } /* endfor */
    printf("\n-----\n");
    return SUCCESS;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// InitClusters
// Arbitrarily assign a vector to each of the K clusters
// We choose the first K vectors to do this
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
void System::InitClusters(){
    int i,j;
    printf("Initial cluster centers:\n");
    for (i=0; i<NumClusters; i++) {
        Cluster[i].Member[0]=i;
        for (j=0; j<SizeVector; j++) {
            Cluster[i].Center[j]=Pattern[i][j];
        } /* endfor */
    } /* endfor */
    for (i=0; i<NumClusters; i++) {
        printf("ClusterCenter[%d]=(%f,%f)\n",i,Cluster[i].Center[0],Cluster[i].Center[1]);
    } /* endfor */
    printf("\n");
}

void System::RunKMeans(){
    int converged;
    int pass;

```

```

pass=1;
converged=FALSE;
while (converged==FALSE) {
    printf("PASS=%d\n", pass++);
    DistributeSamples();
    converged=CalcNewClustCenters();
    ShowCenters();
} /* endwhile */
}

double System::EucNorm(int p, int c){ // Calc Euclidean norm of vector difference
    double dist,x; // between pattern vector, p, and cluster
    int i; // center, c.
    char zout[128];
    char znum[40];
    char * pnum;

    pnum=&znum[0];
    strcpy(zout,"d=sqrt(");
    printf("The distance from pattern %d to cluster %d is calculated as:\n",c,p);
    dist=0;
    for (i=0; i<SizeVector ;i++) {
        x=(Cluster[c].Center[i]-Pattern[p][i])*(Cluster[c].Center[i]-Pattern[p][i]);
        strcat(zout,f2a(x,4));
        if (i==0)
            strcat(zout,"+");
        dist += (Cluster[c].Center[i]-Pattern[p][i])*(Cluster[c].Center[i]-Pattern[p][i]);
    } /* endfor */
    printf("%s)\n",zout);
    return dist;
}

int System::FindClosestCluster(int pat){
    int i, ClustID;
    double MinDist, d;
    MinDist = 9.9e+99;
    ClustID=-1;
    for (i=0; i<NumClusters; i++) {
        d=EucNorm(pat,i);
        printf("Distance from pattern %d to cluster %d is %f\n\n",pat,i,sqrt(d));
        if (d<MinDist) {
            MinDist=d;
            ClustID=i;
        } /* endif */
    } /* endfor */
    if (ClustID<0) {
        printf("Aaargh");
        exit(0);
    } /* endif */
}

```

```

return ClustID;
}

void System::DistributeSamples(){
int i, pat, Clustid, MemberIndex;
//Clear membership list for all current clusters
for (i=0; i<NumClusters; i++) {
    Cluster[i]. NumMembers=0;
}
for (pat=0; pat<NumPatterns; pat++) {
    //Find cluster center to which the pattern is closest
    Clustid= FindClosestCluster(pat);
    printf("pattern %d assigned to cluster %d\n\n", pat, Clustid);
    //post this pattern to the cluster
    MemberIndex=Cluster[Clustid]. NumMembers;
    Cluster[Clustid]. Member[MemberIndex]=pat;
    Cluster[Clustid]. NumMembers++;
} /* endfor */
}

int System::CalcNewClustCenters(){
int ConvFlag, VectID, i, j, k;
double tmp[MAXVECTDIM];
char xs[255];
char ys[255];
char nc1[20];
char nc2[20];
char * pnc1;
char * pnc2;
char * fpv;

pnc1=&nc1[0];
pnc2=&nc2[0];
ConvFlag=TRUE;
printf("The new cluster centers are now calculated as:\n");
for (i=0; i<NumClusters; i++) { //for each cluster
    pnc1=itoa(Cluster[i]. NumMembers, nc1, 10);
    pnc2=itoa(i, nc2, 10);
    strcpy(xs, "Cluster Center");
    strcat(xs, nc2);
    strcat(xs, "(1/"));
    strcpy(ys, "(1/");
    strcat(xs, nc1);
    strcat(ys, nc1);
    strcat(xs, ")");
    strcat(ys, ")");
    for (j=0; j<SizeVector; j++) { // clear workspace
        tmp[j]=0.0;
    } /* endfor */
}
}

```

```

for (j=0; j<Cluster[i].NumMembers; j++) { //traverse member vectors
    VectID=Cluster[i].Member[j];
    for (k=0; k<SizeVector; k++) {           //traverse elements of vector
        tmp[k] += Pattern[VectID][k];          //add (member) pattern elmnt into temp
        if (k==0) {
            strcat(xs,f2a(Pattern[VectID][k],3));
        } else {
            strcat(ys,f2a(Pattern[VectID][k],3));
        } /* endif */
    } /* endfor */
    if(j<Cluster[i].NumMembers-1){
        strcat(xs,"+");
        strcat(ys,"+");
    }
    else {
        strcat(xs,")");
        strcat(ys,")");
    }
} /* endfor */
for (k=0; k<SizeVector; k++)           //traverse elements of vector
tmp[k]=tmp[k]/Cluster[i].NumMembers;
if (tmp[k] != Cluster[i].Center[k])
    ConvFlag=FALSE;
Cluster[i].Center[k]=tmp[k];
} /* endfor */
printf("%s,\n",xs);
printf("%s\n",ys);
} /* endfor */
return ConvFlag;
}

void System::ShowClusters(){
    int cl;
    for (cl=0; cl<NumClusters; cl++) {
        printf("\nCLUSTER %d ==>[%f,%f]\n", cl,Cluster[cl].Center[0],Cluster[cl].Center[1]);
    } /* endfor */
}

void System::SaveClusters(char * fname){

}

main(int argc, char * argv[]){
    System kmeans;
    if (argc<2) {
        printf("USAGE: KMEANS PATTERN_FILE\n");
        exit(0);
    }
    if (kmeans.LoadPatterns(argv[1]) == FAILURE){
}

```

```
printf("UNABLE TO READ PATTERN_FILE: %s\n", argv[1]);
exit(0);
}
kmeans. InitClusters();
kmeans. RunKMeans();
kmeans. ShowClusters();
}

→
```

附录 2 神经网络常用术语英汉对照

activation function	激活函数	hidden layer	隐层
adaptive linear element unit	自适应线性单元	hidden neuron	隐神经元
adaptive resonance theory	自适应共振理论	hyperplanar	超平面
artificial control	智能控制	image recognition	图像识别
artificial neural network	人工神经网络	initial weight	初始权值
asynchronous update	异步更新	instar vector	内星向量
attractor	吸引子	iterating	迭代
autoassociation	自联想	interconnection scheme	连接方式
autoassociative memory	联想记忆	judgement curve	判决曲线
axon	轴突	karush-Kuhn-Tucker	KKT 条件
back propagation algorithm	反向传播算法	kernel function	核函数
basin of attraction	吸引域	lagrange coefficient	拉格朗日系数
bidirectional association	互(双向)联想	learning	学习
biological neuron	生物神经元	learning algorithm	学习算法
bipolar	双极性	learning factor	学习率
category	类别	learning rules	学习规则
cell body	细胞体	learning vector quantization	学习矢量化
cerebellum model	小脑模型	least mean squared error	最小均方误差
character recognition	特征识别, 符号识别	linearly nonseparable pattern	线性不可分模式
chunking algorithm	块算法	linearly separable	线性可分
classifier	分类器	local minimum	局部极小
clustering	聚类	long term memory	长期记忆
competitive learning	竞争学习	margin	分类间隔
content addressable memory	内容寻址记忆	mechanism	机理
convergence	收敛	memory capacity	存储容量
counter propagation network	对偶传播网络	momentum term	动量项
dendrite	树突	multilayer feedforward network	多层前馈网络
dynamical network	动态网络	network topological	网络拓扑结构
empirical risk minimization	经验风险最小化	neurocomputing	神经计算
energy function	能量函数	normalizing	归一化
equilibrium point	平衡点	on-center off-surround	近兴奋远抑制
error function	误差函数	optical neural network	光学神经网络
error target function	目标函数	optimal hyperplane	最优超平面
expert system	专家系统	outer product	外积
fault tolerance	容错性	outstar vector	外星向量
feature mapping	特征映射	overfittin	过学习
feedback network	反馈网络	parallel	并行
feedforward network	前馈网络	perceptron	感知器
gaussian function	高斯函数	population variance	总体方差
generalization capability	泛化能力	precision	精度
global minimum	全局最小	quadratic programming	二次函数极值问题
gradient descent	梯度下降	quantizer	量化器
hamming distance	海明距离	radial basis function	径向基函数
heteroassociation	异联想	recall	回忆

refractory period	不应期	structural risk minimization	结构风险最小化
sample	样本	subspace	子空间
sample variance	样本方差	summation	整合
self-adapting	自适应	supervised/unsupervised	有/无监督学习
self-learning	自学习	support vector machine	支持向量机
self-organization feature maps	自组织特征映射	synapse	突触
self-organizing	自组织	synchronous update	同步更新
sequential minimal optimization	SMO 算法	testing sets	测试集
serial	串行	threshold value	阈值
short term memory	短期记忆	training	训练
sigmoidal function	S型函数	training error	训练误差
simulated annealing	模拟退火	training sets	训练集
speech recognition	语音识别	traveling salesman problem	旅行商问题
stability	稳定性	unipolar	单极性
stable state	稳态	vigilance threshold	警戒门限
standardizing	标准化	weight adjustment	权值调整量
statistical learning theory	统计学习理论	winning neighborhood	获胜邻域
stochastic network	随机网络		



参 考 文 献

- [1] 涂序彦等. 生物控制论. 北京: 科学出版社, 1980.
- [2] McClelland J L, Rumelhart D E. Explorations in Parallel Distributed Processing, A Handbook of Models, Programs, and Exercises. Cambridge: MITPress, 1986.
- [3] 涂序彦. 人工智能及其应用. 北京: 电子工业出版社, 1988.
- [4] Kohonen T. Self-Organization and Associative Memory. New York: Springer-Verlag, 1989.
- [5] Ahmad S, Tesuro G. Scaling and Generalization in Neural Networks. Canada: Proc. of 1988 Connectionist Models Summer School, 1989.
- [6] Robert Hecht-Nielsen. Neurocomputing. Reading: Addison-Wesley Publishing Company, 1990.
- [7] 焦李成. 神经网络系统理论. 西安: 西安电子科技大学出版社, 1990.
- [8] Marilyn McCord Nelson, Illingworth W T. A Practical Guide to Neural Nets. Reading: Addison-Wesley Publishing Company, 1991.
- [9] 钟义信. 信息科学方法与神经网络研究. 自然杂志, 1991, 14 (6): 37~41.
- [10] Jacek M Zurada. Introduction to Artificial Neural Systems. St. Paul: West Publishing Company, 1992.
- [11] 杨行峻, 郑君里. 人工神经网络. 北京: 高等教育出版社, 1992.
- [12] 张立明. 人工神经网络的模型及其应用. 上海: 复旦大学出版社, 1993.
- [13] 阎平凡, 黄端旭. 人工神经网络——模型·分析与应用. 合肥: 安徽教育出版社, 1993.
- [14] 周继成. 人工神经网络——第六代计算机的实现. 北京: 科学普及出版社, 1993.
- [15] 胡守仁, 余少波, 戴葵. 神经网络导论. 长沙: 国防科技大学出版社, 1993.
- [16] 沈清, 胡德文, 时春. 神经网络应用技术. 长沙: 国防科技大学出版社, 1993.
- [17] 涂序彦. 大系统控制论. 北京: 国防工业出版社, 1994.
- [18] Tuxuyan. Information Structure and Pattern of Neural Systems. Beijing: Invited Speech of International Processing, 1995.
- [19] 李孝安, 张晓绩. 神经网络与神经计算机导论. 西安: 西北工业大学出版社, 1995.
- [20] 陈明. 神经网络模型. 大连: 大连理工大学出版社, 1995.
- [21] 焦李成. 神经网络计算. 西安: 西安电子科技大学出版社, 1995.
- [22] 胡铁松. 水库群优化调度的人工神经网络方法研究. 水利学进展, 1995, 6 (1), 53~60.
- [23] 徐庐生. 微机神经网络. 北京: 中国医药科技出版社, 1996.
- [24] 黄德双. 神经网络模式识别系统理论. 北京: 电子工业出版社, 1996.
- [25] 唐丽艳, 李卫东, 景瑜. 应用人工神经网络进行企业综合经济效益评估. 管理工程学报, 1996, 10 (2): 89~94.
- [26] 何振亚主编. 神经智能——认知科学中若干重大问题的研究. 长沙: 湖南科学技术出版社, 1997.
- [27] 孙增圻. 智能控制理论与技术. 北京: 清华大学出版社. 南宁: 广西科学技术出版社, 1997.
- [28] Liqun Han. Initial Weight Selection Methods for Self-Organizing Training. Proc. of the IEEE CIPS'97. 北京: 万国出版社, 1997: 406.
- [29] Cai Min, Liqun Han. Neural Network Based Computer Leather Matching System. Proc. of the IEEE CIPS'97, 北京: 万国出版社, 1997: 377.
- [30] 韩力群. 基于自组织神经网络的皮革纹理分类. 中国皮革, 1997, 26 (6): 11.
- [31] 徐章英, 顾力兵. 智力工程概论. 北京: 人民教育出版社, 1997.
- [32] 王文成. 神经网络及其在汽车工程中的应用. 北京: 北京理工大学出版社, 1998.
- [33] 杨雄里. 脑科学的现代进展. 上海: 上海科技教育出版社, 1998.
- [34] 丛爽. 面向 MATLAB 工具箱的神经网络理论与应用. 合肥: 中国科学技术大学出版社, 1998.
- [35] Liqun Han. Two Neural Network Based Methods for Leather Pattern Recognition. Proc. of CAIE'98, 武汉: 华中理工大学出版社, 1998: 355.
- [36] 韩力群等. 测量仪表特性线性化的神经网络方法. 北京轻工业学院学报, 1998, 16 (3): 38.
- [37] 李士勇. 模糊控制·神经控制和智能控制. 哈尔滨: 哈尔滨工业大学出版社, 1998.
- [38] 戴葵. 神经网络实现技术. 长沙: 国防科技大学出版社, 1998.
- [39] 袁曾任. 人工神经元网络及其应用. 北京: 清华大学出版社, 1999.
- [40] 韩力群. 催化剂配方的神经网络建模与遗传算法优化. 化工学报, 1999, 50 (4): 500~503.
- [41] 韩力群等. 一种远程水污染神经网络监测系统. 北京轻工业学院学报, 1999, 17 (3): 7.
- [42] 赵林明等. 多层前向人工神经网络. 郑州: 黄河水利出版社, 1999.
- [43] 斯蕃. 神经计算智能基础、原理、方法. 成都: 西南交通大学出版社, 2000.
- [44] Vladimir N Vapnik 著. 统计学习理论的本质. 张学工译. 北京: 清华大学出版社, 2000.

- [45] 韩力群. 教学质量评价体系的神经网络模型. 北京轻工业学院学报, 2000, 18 (2): 34~36.
- [46] 杨天奇. 基于 ART 网的直流提升机故障诊断. 计算机自动测量与控制, 2000, 8 (2): 22~23.
- [47] 涂晓媛. 人工鱼——计算机动画的人工生命方法. 北京: 清华大学出版社, 2001.
- [48] 何为, 韩力群. 基于神经元网络模型的稳压变压器优化设计. 变压器, 2001, 38 (9): 24~25.
- [49] 丁雪梅, 李英梅, 论立军. 基于神经网络的证券预测技术研究. 哈尔滨师范大学自然科学学报, 2003, 19 (4): 57~60.
- [50] 王群等. 基于神经网络的探地雷达探雷研究. 电波科学学报, 2001, 16 (3): 398~403.
- [51] 张菊香, 邱阳. 基于 CMAC 的有源噪声控制研究. 西安电子科技大学学报(自然科学版), 2002, 29 (5): 614~618.
- [52] [美] Hagan M T 等著. 神经网络设计. 戴葵等译. 北京: 机械工业出版社, 2002.
- [53] 庞素琳, 王燕鸣, 罗育中. 多层感知器信用评模型及预警研究. 数学的实践与认识, 2003, 33 (9): 55~62.
- [54] 徐丽娜. 神经网络控制. 北京: 电子工业出版社, 2003.
- [55] [美] Simon Haykin 著. 神经网络原理. 叶世伟, 史忠植译. 北京: 机械工业出版社, 2004.
- [56] 赵闯, 刘凯, 李电生. SOFM 神经网络在物流中心城市分类评价中的应用. 中国公路学报, 2004, 17 (4): 119~122.
- [57] 张友水等. Kohonen 神经网络在遥感影像分类中的应用研究. 遥感学报, 2004, 8 (2): 178~183.
- [58] 马骏等. 基于频谱分析和自组织神经网络的火焰燃烧诊断研究. 动力工程, 2004, 24 (6): 853~856.
- [59] 朱贵明. 压力容器焊缝缺陷探伤的定性分析. 无损探伤, 2005, 29 (3): 43~46.
- [60] 尹豪, 朱晓丽. 略论人工神经网络方法在经济管理领域的应用. 现代管理科学, 2005, 8: 52~53.
- [61] 阎平凡, 张长水. 人工神经网络与模拟进化计算. 第 2 版. 北京: 清华大学出版社, 2005.
- [62] 刘小铭. 我国证券投资基金分类实证分析. www.cenet.org.cn/cn/CEAC/2005in/j1016.doc (经济学资源数据库).
- [63] 韩力群等. 青藏高原热红外遥感与地表层温度相关性研究. 中国科学(E辑), 2006, 36 (s): 109~115.
- [64] Kumar S. Neural Networks (影印本). 北京: 清华大学出版社, 2006.
- [65] 韩力群. 人工神经网络教程. 北京: 北京邮电大学出版社, 2006.



[General Information]

书名=人工神经网络理论、设计及应用 第2版

作者=%E9%9F%A9%E5%8A%9B%E7%BE%A4

页码=243

I S B N = 2 4 3

S S 号 = 1 1 8 7 9 1 5 5

d x N u m b e r = 0 0 0 0 0 4 9 3 8 7 6 7

出版时间=2007

出版社=北京市：化学工业出版社

定价：29.80

试读地址=http://book.szdn.net.org.cn/views/specif
ic/2929/bookDetail.jsp?dxNumber=00000493876
7&d=75BCC41477EE4DFC6263F8766EA96599&fenlei
=1817020603#ctop

全文地址=http://ttoa.5read.com/image/ss2jpg.dl
?did=b60&pid=1BC49CE010A92A241BB6CD9A7151D4
9D42F09BDD32962DBEA3CF184A1828ABD8122FF910E
D451549246EC7DE8AAED02799E210B404D76FCE7608
4B6361E9D136AB7B04DA5A601A987B83A7943CE9ACC
C982F41C7A317F1F1961C14A5B98C5913E3618E0306
155985604AD7929A6599CCE12E&jid=/