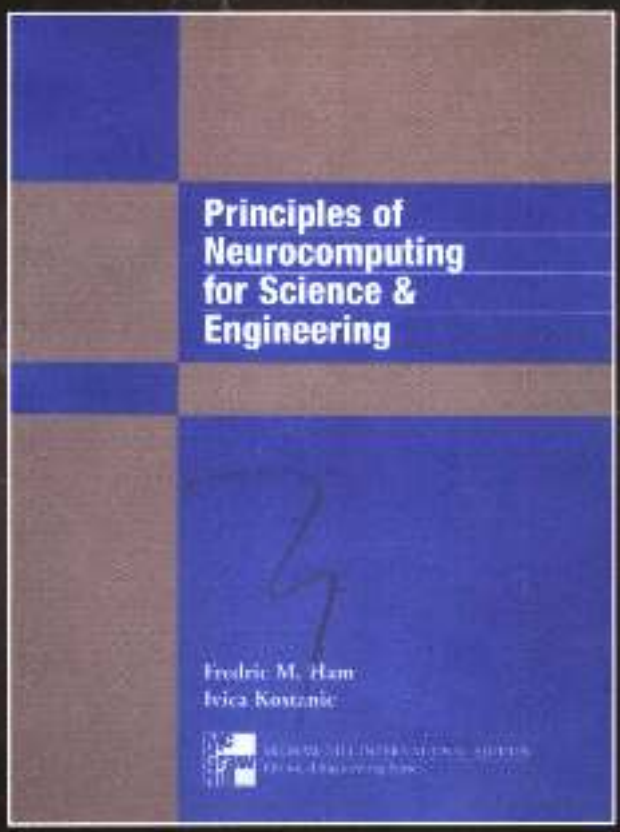
计 算 机 科 学 丛 书

# 神经计算原理

(美) Fredric M. Ham Ivica Kostanic 著 叶世伟 王海娟 译




# Principles of Neurocomputing for Science & Engineering

机械工业出版社
China Machine Press

# 神经计算原理

本书是神经网络领域中的一部优秀教材，着重讲述人工神经网络基本原理以及如何运用各种神经计算技术来解决科学和工程领域中的现实问题，如模式识别、最优化、事件分类、非线性系统的控制和识别以及统计分析等。

主要特点：

- 算法——很多算法用框线明确标出，便于读者查找。
- MATLAB Toolbox——书中大量使用MATLAB的Neural Network Toolbox，举例说明神经计算概念。
- Web站点——登录http://www.mhhe.com/ham，可获取最新、最全面的信息。
- 示例和附录——各章有详尽的示例，阐述重要的神经计算概念，附录A全面介绍了神经计算的数学基础。


作者简介

**Fredric M. Ham** 博士现任佛罗里达理工学院的Harris教授，佛罗里达理工学院信息处理实验室的主任。他于1980年在爱荷华州立大学获得电气工程博士学位。他在信号处理、生物医学工程、神经网络和控制系统领域发表了许多论文，是电气与电子工程师协会(IEEE)的高级会员，还是Eta Kapoa Nu，Tau Beta Pi，Phi Kappa Phi和Sigma Xi的会员。



**Ivica Kostanic** 博士现任佛罗里达理工学院电气与计算机工程系助理教授。他于2003年在佛罗里达中央大学获得博士学位。他的主要研究兴趣在无线通信、数字信号处理和神经网络的实际应用等方面。他还是电气与电子工程师协会(IEEE)的会员，在无线通信、神经网络和非线性控制领域发表了数篇论文。





Mc Graw Hill Education

ISBN 978-7-111-20637-8

9 787111 206378

HZ BOOKS 华章图书

上架指导：计算机/人工智能/神经网络

华章网站 http://www.hzbook.com

网上购书：www.china-pub.com

投稿热线：(010) 88379604

购书热线：(010) 68995259, 68995264

读者信箱：hzjsj@hzbook.com

ISBN 978-7-111-20637-8

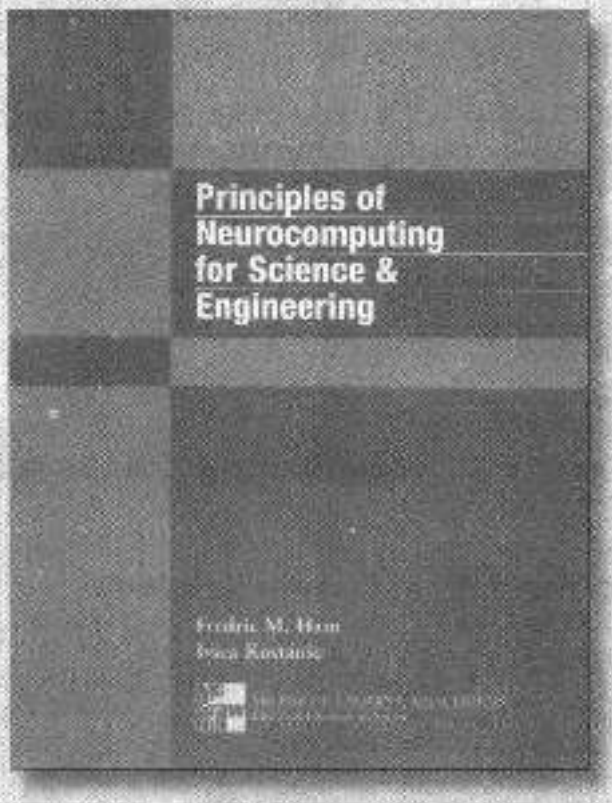
定价：59.00 元

TP183
54
2007

计 算 机 科 学 丛

# 神经计算原理

(美) Fredric M. Ham Ivica Kostanic 著 叶世伟 王海娟 译




**Principles of Neurocomputing for Science & Engineering**

机械工业出版社
China Machine Press

本书比较系统全面地介绍了人工神经网络的理论和实际应用，特别在神经网络模型和工程应用方面有极为深入的分析和讲解。全书不仅深入分析神经网络的基本概念，而且详细介绍神经网络应用方面的最新发展趋势和主要研究方向。本书理论和实际应用紧密结合，为神经网络的相关理论知识在具体问题中的应用打下了坚实的基础。

本书适合作为高等院校计算机专业高年级本科生或研究生的教材，也可供人工智能及神经网络方面的研究人员和专业技术人员参考。

Fredric M. Ham and Ivica Kostanic: Principles of Neurocomputing for Science and Engineering (ISBN 0-07-025966-6).

Copyright © 2001 by The McGraw-Hill Companies, Inc.

Original English edition published by The McGraw-Hill Companies, Inc. All rights reserved. No part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

Simplified Chinese translation edition jointly published by McGraw-Hill Education (Asia) Co. and China Machine Press.

本书中文简体字翻译版由机械工业出版社和美国麦格劳-希尔教育（亚洲）出版公司合作出版。未经出版者预先书面许可，不得以任何方式复制或抄袭本书的任何部分。

本书封面贴有McGraw-Hill公司防伪标签，无标签者不得销售。

**版权所有，侵权必究。**

**本书法律顾问　北京市展达律师事务所**

**本书版权登记号：图字：01-2003-5006**

**图书在版编目（CIP）数据**

神经计算原理/（美）哈姆，（美）科斯塔尼克著；叶世伟，王海娟译. -北京：机械工业出版社，2007.5

（计算机科学丛书）

书名原文：Principles of Neurocomputing for Science and Engineering

ISBN 978-7-111-20637-8

Ⅰ.神… Ⅱ.①哈… ②科… ③叶… ④王… Ⅲ.人工神经元网络-计算 Ⅳ.TP183

中国版本图书馆CIP数据核字（2007）第031475号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：王　玉

三河市明辉印装有限公司 印刷 · 新华书店北京发行所发行

2007年5月第1版第1次印刷

184mm×260mm · 32.25印张

定价：59.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

# 出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专诚为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业

的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzjsj@hzbook.com
联系电话：(010) 68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

# 专家指导委员会

（按姓氏笔画顺序）

| | | | | |
|---|---|---|---|---|
| 尤晋元 | 王　珊 | 冯博琴 | 史忠植 | 史美林 |
| 石教英 | 吕　建 | 孙玉芳 | 吴世忠 | 吴时霖 |
| 张立昂 | 李伟琴 | 李师贤 | 李建中 | 杨冬青 |
| 邵维忠 | 陆丽娜 | 陆鑫达 | 陈向群 | 周伯生 |
| 周克定 | 周傲英 | 孟小峰 | 岳丽华 | 范　明 |
| 郑国梁 | 施伯乐 | 钟玉琢 | 唐世渭 | 袁崇义 |
| 高传善 | 梅　宏 | 程　旭 | 程时端 | 谢希仁 |
| 裘宗燕 | 戴　葵 | | | |

# 译 者 序

神经计算研究的重要意义已经为许多科学家所共识，神经计算已成为智能计算发展的一个主流方向。20世纪80年代中期以来，神经网络的发展已经成为近代非线性科学和计算智能研究的主要内容之一。特别是神经网络经历了近20年的迅速发展，其独特的知识表示结构和信息处理的原则，使其在许多应用领域中取得了显著效果，成为信息处理的一个有力工具，为解决一些传统计算机极难求解的问题提供了全新的思路。

目前，神经网络的研究（包括信息处理机制、原理和应用）取得了长足的发展，实际上神经网络已成为智能信息处理的主要技术之一。然而，如何有效地掌握神经网络的基本理论，如何利用神经网络的信息处理特点对实际问题求解还有待进一步的研究。在对神经网络感兴趣的科技工作者中，既有从事神经网络模型和原理方面研究的理论工作者，也有很多希望利用神经网络新颖的信息处理机制求解实际问题的应用研究者。如何紧扣神经网络的发展方向，面向应用、面向广大神经网络的理论和应用研究者，如何介绍神经网络的系统理论和具体应用，已成为神经网络课程教学面临的重大挑战。当然，神经网络的理论研究和应用不是割裂开来的，二者有着紧密的联系。本书正是从这一点入手，首先介绍神经网络的基本原理和系统方法，然后，紧密联系神经网络的最新发展成果，从神经网络的具体应用领域中挑选比较典型的应用，详细介绍如何利用神经网络解决实际问题，比较已有的解决方法和神经网络解决方法的异同，具体而深入地介绍神经网络应用的原理、方法和结果。这些都是本书比较突出的特点。

在这本书的翻译中，我们力求忠实、准确地反映原著的内容，同时，也力求保留原著的风格。由于神经网络的迅速发展，许多神经网络的新名词和概念还没有确定的中文译法，所以在本书中，凡是我们认为不能完全确定的名词或术语都在其第一次出现的地方给出了对应的英文原文，有一些甚至保留了英文原文，在全书最后还有中英文索引对照。

神经网络属于多学科交叉领域，研究范围很广。近年来，研究成果层出不穷。同时，由于译者水平有限，书中错误和不准确之处在所难免，恳请作者和读者批评指正。

译 者

2007年1月

# 前　言

亲身经历是最好的学习。

——阿尔伯特·爱因斯坦

任何一本教材在前言里都要回答四个主要问题：（1）本书讲述什么主题？（2）为什么出版本书？（3）本书的预期读者有哪些，需要哪些预备知识？（4）书中包含什么具体内容？

**问**：本书讲述什么主题？

**答**：这本教材是关于人工神经网络（或神经网络）的。更具体地说，本书论述神经计算。所以问题实际是：什么是神经计算？神经计算通常就是指信息处理。与程序计算不同，神经计算中的信息处理首先在人工神经网络（神经网络）结构内进行学习，这个神经结构根据预定的学习规则学会或自适应响应输入；在神经网络学好它所需要知道的东西之后，训练后的神经网络根据特定的应用可以用于执行特定的任务。神经网络以类似生物的交互方式，从它们的环境学习并适应环境。在求解那些用其他方法解决很困难的科学或工程问题中，神经计算会发挥重要作用，这些问题包括：模式识别、优化、事件分类、非线性系统的控制和辨识以及统计分析等。因此，本书主要面向希望理解人工神经网络的基本原理及其在科学和工程中应用的读者。

**问**：为什么出版本书？

**答**：神经网络的领域非常宽广，且与多学科交叉。神经网络技术已经引起了许多不同领域的研究者的兴趣，而且成果非常庞大。关于神经网络技术的书籍有很多。但是，作者感到需要一本专门针对科学家和工程师的书，即针对那些希望应用神经网络求解复杂问题的科学家和工程师。这并不意味着本书只讲解神经结构及其相应训练算法，实际上，本书给出了许多可用于求解科学和工程中大量问题的各种神经计算方法。在介绍几乎所有的神经计算概念时，都给出详细的数学推导，以及与这个特定结构相伴的说明性例子和相应的训练算法。

**问**：本书的预期读者有哪些，需要哪些预备知识？

**答**：本书主要用于研究生水平的神经网络课程。但是，高年级本科学生可以在具备恰当背景（即具备下面介绍的预备知识）的基础上使用这本教材。而且，应用工程师和科学家也可以自学本书。使用这本教材成功学习神经计算所需的预备知识包括：线性代数和微分方程组，最好具有随机变量和随机过程领域的知识，但这不是必要的，因为这些也包含在本书附录A中（内容虽然简单但也足够）。

**问**：书中包含什么具体内容？

**答**：本书分成两个主要部分，细节如下。附录A包含神经计算的数学基础。

**第一部分：神经计算的基本概念和部分神经网络体系结构及其学习规则**

包括第1～5章。

**第1章**为读者介绍神经网络和神经计算的基本思想，同时包括神经网络的简单历史。

**第2章**首先讨论作为神经网络的构建模块的人工神经元的基本模型。接着讨论激活函数的

不同类型，然后，给出了自适应线性单元（Adaline）和多重自适应线性单元（Madaline）。接着给出了最小均方（LMS）算法；然后详细介绍简单感知器，简单讨论多层前馈感知器。然后包含一些基本的学习规则。这些学习规则是训练更加复杂神经网络结构的基础。第2章最后总结一些经过精心挑选的数据处理方法。如果读者对人工神经网络不熟悉，则应该学习第2章的所有内容。这章是深入理解第3～5章中精选的神经网络结构及其相关算法的基础。

**第3章**介绍多种映射神经网络，以联想记忆开始，然后介绍用于训练多层前馈感知器的反向传播，对反向传播给出了更加高级的训练方法。还介绍了对传网络，本章最后给出径向基函数神经网络。

**第4章**讨论部分自组织神经网络。这包括Kohonen自组织映射（SOM）和学习向量量化（LVQ）。最后介绍自适应共振理论（ART）神经网络，并给出ART1网络的细节。

**第5章**介绍递归神经网络和时间前馈网络（它也是递归网络），介绍了这些时间前馈神经网络和那些不是多层前馈网络之间的区别。该章包括霍普菲尔德网络、模拟退火、玻尔兹曼机、简单递归神经网络（SRN）、时延网络和分布式时滞前馈神经网络。

**第二部分：神经计算的应用**

包括第6～10章。

**第6章**介绍用于求解约束最优化问题的部分神经计算方法。给出了用于线性规划和二次规划问题的神经网络。最后讨论用于非线性连续约束最优化问题的神经网络。这章包括用于非线性规划惩罚方法和障碍函数方法的神经网络，也包括用于普通的和增广的拉格朗日乘子方法的神经网络。

**第7章**讨论用于求解各种矩阵代数问题的结构化神经网络体系结构和相关的学习规则。给出了相当多的重要矩阵分解（或因子分解）以及每个方法的神经计算解。也给出了应用神经计算方法实例，如矩阵的伪逆、求解代数李雅普诺夫方程和求解代数里卡蒂方程。

**第8章**讨论用于求解线性代数方程组的神经计算方法。这些方法包括最小二乘神经计算方法、共轭梯度学习规则、广义鲁棒神经计算方法、用于具有未定数值秩的不适定问题的正则化方法、用于迭代离散时间方法的矩阵分裂和总体最小二乘问题。还给出了求解线性代数方程组的$L_\infty$范数和$L_1$范数的神经网络方法。

**第9章**包含许多用于数据统计分析的神经网络体系结构，包括用于主成分分析（PCA）、主成分回归（PCR）、经典的最小二乘（CLS）的神经网络，用于非线性PCA和鲁棒PCA的神经网络，用于部分最小二乘回归（PLSR）的神经网络，以及用于鲁棒PLSR的神经网络方法。

**第10章**包含信号处理应用、线性和非线性系统辨识、非线性控制和估计的神经网络，详细解释了许多例子。也包括对盲源分离使用神经网络的独立成分分析（ICA）。另外，介绍了快速ICA算法以及应用快速ICA算法分离数字图像的例子。

**本书的主要特征**

- 突出大多数的训练算法，使得它们很容易查找。
- 这些训练方法中的一部分在正文中给出了它们的MATLAB函数实现。代码相当短，只需花费几分钟就可以进入MATLAB。
- 另外，广泛使用MATLAB神经网络工具箱，以便用实验说明一些神经计算的概念。
- 本书的一些问题涉及的数据保留在McGraw-Hill高等教育出版社的网页上，并且很容易

访问到。本书的URL为：http://www.mhhe.com/ham。⊖

- 在许多章节里给出了详细的例子以阐明神经计算的概念。
- 在每章（除第1章外）的最后都给出了涉及广泛内容的大量习题。一些习题需要使用MATLAB和MATLAB神经网络工具箱。一些情况下提供了MATLAB函数的代码。
- 附录A包含神经计算的综合数学基础。

Fredric M. Ham
Ivica Kostanic

⊖ 本网址上的一些教辅资源（包括习题答案等）只提供给采用本书作为教材的老师，需要者请与McGraw-Hill公司北京代表处联系，联系方式见书后所附的教学服务沟通表。——编辑注

# 致　谢

本书作者希望感谢本书的评阅者：Okan K. Ersoy，Purdue University；Sylvian R. Ray，University of Illinois；Yu Hen Hu，University of Wisconsin-Madison；Bruce MacLennan，University of Tennessee；Simon Y. Foo，Florida State University和Risto Miikkulainen，The University of Texas at Austin。他们的评价和建议在准备本书的最终稿方面是非常珍贵和必不可少的。

我们也希望对以下人员表达谢意：Glenn Cohen 对于1.3节“神经计算和神经科学”的贡献；Sungjin Park对于第10章最终准备的协助；Tom McDowall 对于9.7节“鲁棒PLSR：一种神经网络方法”的贡献；佛罗里达理工大学Ham教授讲授的ECE 5268和CSE 5294（“神经网络理论和应用”）两门课程的许多学生；Joseph C. Wheeler 和 Boeing公司的帮助；Matt Ham对于一些图形制作的帮助；Edwin Sherman对于他在信息处理实验室的协助和Math Works公司，特别是Naomi Fernandes、Brian Bostek和Peter Trogos的支持。

MathWorks公司的联系信息：

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760
USA
Tel: 508-647-7000
Fax: 508-647-7001
E-mail: info@mathworks.com
Web:www.mathworks.com

# 重要符号和算符

| | |
|---|---|
| $\boldsymbol{A}>0$ | 正定矩阵$\boldsymbol{A}$ |
| $\boldsymbol{A}\geqslant 0$ | 半正定（非负定）矩阵$\boldsymbol{A}$ |
| $\boldsymbol{A}<0$ | 负定矩阵$\boldsymbol{A}$ |
| $\boldsymbol{A}\leqslant 0$ | 半负定（非正定）矩阵$\boldsymbol{A}$ |
| $\boldsymbol{A}_{:k}$ | 代表矩阵$\boldsymbol{A}$的第$k$列 |
| $\boldsymbol{A}_{k:}$ | 代表矩阵$\boldsymbol{A}$的第$k$行 |
| $\tilde{\boldsymbol{A}}$ | 集合$\boldsymbol{A}$的补 |
| $a_{ij}$ | 矩阵$\boldsymbol{A}$的$i$行$j$列元素 |
| arg | 复数量的辐角 |
| adj | 伴随算子 |
| | |
| $\hat{\boldsymbol{b}}_f$ | 校正模型 |
| $\beta$ | 偏置 |
| | |
| $\mathscr{C}$ | 复数集 |
| $\mathscr{C}^{n\times m}$ | 复$n\times m$矩阵 |
| $\mathscr{C}^{n\times n}$ | 复$n\times n$（方）阵 |
| $\mathscr{C}^{n\times 1}$ | 复$n$维列向量 |
| $\mathscr{C}^{1\times n}$ | 复$n$维行向量 |
| $\mathscr{C}^{n}$ | 复$n$维向量（行或列） |
| $(\mathscr{C}^{n},\ \mathscr{C})$ | $n$维复向量空间 |
| $C_X$ | 自协方差函数 |
| $\boldsymbol{C}_X$ | 协方差矩阵 |
| cof | 余因子算子 |
| $\mathrm{cond}_p(\boldsymbol{A})$ | 矩阵$\boldsymbol{A}$的条件数 |
| | |
| $\Delta\mathscr{E}$ | 能量变化 |
| diag | 挑选方阵对角元素的算子 |
| $\delta_{ij}$ | 克罗内克Δ |
| det | 行列式 |
| | |
| exp | 指数 |
| E | 期望算子 |
| $\mathscr{E}$ | 能量函数 |
| $E$ | 能量函数 |
| $e$ | 指数 |
| | |
| $\mathscr{F}$ | 域 |
| $f_{\mathrm{bs}}$ | 二值S形激活函数 |
| $f_{\mathrm{lin}}$ | 线性激活函数 |
| $f_L$ | logistic函数 |
| $f_{\mathrm{hl}}$ | 硬限幅激活函数 |
| $f_H$ | Huber函数 |
| $f_{\mathrm{hts}}$ | 双曲正切S形激活函数 |
| $f_M$ | M估计器函数 |
| $f_T$ | Talwar函数 |

（续）

| | |
|---|---|
| $f_s$ | 采样频率，赫兹 |
| $f_{shl}$ | 对称硬限幅激活函数 |
| $f_{sl}$ | 饱和线性激活函数 |
| $f_{ssl}$ | 对称饱和线性激活函数 |
| | |
| $g_{bs}$ | 二值S形激活函数的导数 |
| $g_L$ | logistic函数的导数 |
| $g_{hts}$ | 双曲正切S形激活函数的导数 |
| $g_H$ | Huber函数的导数 |
| $g_M$ | M估计器函数的导数 |
| $g_T$ | Talwar函数的导数 |
| $g \circ f$ | $f$和$g$的复合 |
| $\gamma$ | 遗忘因子或泄漏因子 |
| | |
| $\boldsymbol{H}$ | 黑塞矩阵 |
| $h^o$ | 因子的最优数 |
| | |
| $\boldsymbol{I}_n$ | $n \times n$单位矩阵 |
| $\boldsymbol{I}$ | 适当维数的单位矩阵 |
| inf | 下确界 |
| | |
| $J$ | 代价函数 |
| $\boldsymbol{J}$ | 雅可比矩阵 |
| | |
| $k$ | 离散时间指标 |
| kurt | 峭度算子 |
| | |
| $L$ | 李雅普诺夫函数 |
| LT | 下三角矩阵算子 |
| $\mathscr{L}$ | 拉格朗日函数，拉普拉斯变换算子，或逐次松弛矩阵 |
| $\ell$ | 学习信号 |
| $\lambda$ | 特征值或者拉格朗日乘子 |
| | |
| $m_x$ | 随机过程的均值 |
| $\boldsymbol{m}_x$ | 向量随机过程的均值向量 |
| MIN | 模糊交 |
| MAX | 模糊并 |
| min | 最小值算子 |
| max | 最大值算子 |
| $\mu$ | 学习率参数 |
| | |
| $v$ | 零维（数） |
| $\boldsymbol{\Omega}$ | 非线性映射 |
| $O(\gamma^2)$ | 在$\gamma$的高阶效应 |
| $\omega_S$ | 采样频率，$\frac{\text{rad}}{\text{sec}}$ |
| | |
| Pr | 概率 |
| $p_X$ | 概率密度函数 |

（续）

| | |
|---|---|
| $\mathscr{P}_X$ | 概率分布函数 |
| $\Psi$ | 损失函数 |
| $\boldsymbol{\Phi}$ | 回归矩阵 |
| $\boldsymbol{\phi}$ | 状态转移矩阵 |
| $\boldsymbol{\phi}_h$ | 原型记忆 |
| | |
| $\Re$ | 实数集 |
| $\Re^{n\times m}$ | 实$n\times m$矩阵 |
| $\Re^{n\times n}$ | 实$n\times n$（方）阵 |
| $\Re^{n\times 1}$ | 实$n$维列向量 |
| $\Re^{1\times n}$ | 实$n$维行向量 |
| $\Re^n$ | 实$n$维向量（行或列） |
| $(\Re^n, \Re)$ | 实$n$维向量空间 |
| $(\Re^n(s), \Re(s))$ | $n$维有理向量空间 |
| $\mathscr{R}$ | 瑞利商 |
| $\boldsymbol{R}_X$ | 相关矩阵 |
| $R_x$ | 自相关函数 |
| $\mathscr{R}_x$ | 时间自相关函数 |
| $\rho$ | 矩阵秩或警戒参数 |
| $\rho_X$ | 相关系数 |
| | |
| $\boldsymbol{S}_x$ | 功能谱密度矩阵 |
| $\mathscr{S}$ | 概率空间或采样空间 |
| sup | 上确界 |
| sgn | 符号函数 |
| $\sigma$ | 标准偏差，奇异值，扩展参数或正则化参数 |
| $\sigma^2$ | 方差 |
| $\boldsymbol{\sigma}_h$ | 固定的稳定点 |
| | |
| $t$ | 连续时间 |
| tr | 矩阵的迹 |
| trace | 矩阵的迹 |
| $T$ | 温度 |
| $T_{binary}$ | 阈值逻辑算子 |
| $T_S$ | 采样周期 |
| Tr | 对所有可能构形求和 |
| θ | 阈值 |
| $\boldsymbol{\theta}$ | 参数向量 |
| | |
| $V(\boldsymbol{x})$ | 状态向量$\boldsymbol{x}$的李雅普诺夫函数 |
| var | 方差算子 |
| vec | 通过“堆栈堆放”一个矩阵的列形成一个向量的算符 |
| vecd | 选择一个方阵的主对角元素的算符 |
| $\mathscr{X}$ | 向量集 |
| $(\mathscr{X},\mathscr{F})$ | 向量（线性）空间 |
| $\overline{X}$ | 随机变量$X$的均值 |
| $\bar{x}$ | $x$的非或补 |
| $\boldsymbol{x}^*$ | 向量$\boldsymbol{x}$的复共轭转置或最优解 |
| $\boldsymbol{x}^T\boldsymbol{y}$ | 两个向量$\boldsymbol{x}$和$\boldsymbol{y}$的内积 |

（续）

| | |
|---|---|
| $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ | 两个向量$\boldsymbol{x}$和$\boldsymbol{y}$的内积 |
| $\boldsymbol{x}\boldsymbol{y}^T$ | 两个向量$\boldsymbol{x}$和$\boldsymbol{y}$的外积 |
| $z^{-1}$ | 单位延迟算子 |
| | |
| $(\boldsymbol{A})^T$ | 矩阵$\boldsymbol{A}$的转置 |
| $(\boldsymbol{A})^{-1}$ | 矩阵$\boldsymbol{A}$的逆 |
| $(\boldsymbol{A})^{1/2}$ | 矩阵$\boldsymbol{A}$的平方根 |
| $(\boldsymbol{A})^{T/2}$ | 矩阵$\boldsymbol{A}$平方根的转置 |
| $(\boldsymbol{A})^{+}$ | 矩阵$\boldsymbol{A}$的伪逆 |
| $(\boldsymbol{A})^{*}$ | 矩阵$\boldsymbol{A}$的复共轭转置 |
| $(\boldsymbol{A})^{H}$ | 矩阵$\boldsymbol{A}$的埃尔米特转置 |
| $\lvert\boldsymbol{A}\rvert$ | 矩阵$\boldsymbol{A}$的行列式 |
| $\lvert\alpha + j\beta\rvert$ | 复数量的幅度或绝对值 |
| $\angle(\alpha + j\beta)$ | 复数量的角度或辐度 |
| $\det(\boldsymbol{A})$ | 矩阵$\boldsymbol{A}$的行列式 |
| $\Delta \boldsymbol{w}$ | 权值向量变化量 |
| $\nabla$ | 梯度算子 |
| $\nabla_x \mathrm{E}(\boldsymbol{x})$ | 函数$\varepsilon$关于向量$\boldsymbol{x}$的梯度 |
| $\nabla^T \boldsymbol{f}(\boldsymbol{x})$ | 向量函数$\boldsymbol{f}$关于向量$\boldsymbol{x}$的雅可比矩阵 |
| $\nabla^2$ | 拉普拉斯算子 |
| $\nabla_x^2 f(\boldsymbol{x})$ | 函数$f$关于向量$\boldsymbol{x}$的黑塞矩阵 |
| $\oplus$ | 克罗内克求和或者或逻辑符号 |
| $\otimes$ | 克罗内克乘积 |
| $\odot$ | Khatri-Rao乘积 |
| $\cup$ | 并 |
| $\cap$ | 交 |
| $\subset$ | 包含于 |
| $\in$ | 属于 |
| $\notin$ | 不属于 |
| $\forall$ | 对所有 |
| $\ni$ | 使得 |
| $\wedge$ | 与（也是MIN运算符） |
| $\vee$ | 或（也是MAX运算符） |
| $\varnothing$ | 空集 |
| $\Rightarrow$ | 蕴涵 |
| $\rightarrow$ | 映射到 |
| $\lVert\boldsymbol{x}\rVert_p$ | 向量$\boldsymbol{x}$的$L_p$范数 |
| $\lVert\boldsymbol{x}\rVert_1$ | 向量$\boldsymbol{x}$的$L_1$范数（绝对值范数） |
| $\lVert\boldsymbol{x}\rVert_2$ | 向量$\boldsymbol{x}$的$L_2$范数（欧几里得范数） |
| $\lVert\boldsymbol{x}\rVert_\infty$ | 向量$\boldsymbol{x}$的$L_\infty$范数（切比雪夫范数） |
| $\lVert\boldsymbol{x}\rVert_{-\infty}$ | 向量$\boldsymbol{x}$的$L_{-\infty}$范数（负无穷大范数） |
| $\lVert\boldsymbol{x}\rVert_W$ | 向量$\boldsymbol{x}$的内积生成范数 |
| $\lVert\boldsymbol{x}\rVert_{2-Q}$ | 向量$\boldsymbol{x}$的加权欧几里得范数 |
| $\lVert\boldsymbol{A}\rVert_p$ | 矩阵$\boldsymbol{A}$的$L_p$范数 |
| $\lVert\boldsymbol{A}\rVert_1$ | 矩阵$\boldsymbol{A}$的$L_1$范数（最大的列绝对值求和） |
| $\lVert\boldsymbol{A}\rVert_2$ | 矩阵$\boldsymbol{A}$的谱范数 |
| $\lVert\boldsymbol{A}\rVert_\infty$ | 矩阵$\boldsymbol{A}$的$L_\infty$范数（最大的行绝对值求和） |
| $\lVert\boldsymbol{A}\rVert_F$ | 矩阵$\boldsymbol{A}$的弗罗贝尼乌斯范数 |
| $\sigma_r(\boldsymbol{A})$ | 矩阵$\boldsymbol{A}$的谱半径 |

# 重要缩写词

| | |
|---|---|
| Adaline | adaptive linear element，自适应线性单元 |
| AIC | Akaike's information theoretic criterion, Akaike的信息论准则 |
| AND | AND logic function，与逻辑函数 |
| ANN | artificial neural network，人工神经网络 |
| APEX | adaptive principal component extraction，自适应主成分提取 |
| ARMA | autoregressive moving average，自回归滑动平均 |
| ARMAX | autoregressive moving average with exogenous inputs，具有外部输入的自回归滑动平均 |
| BER | bit error rate，位误差率 |
| BFGS | Broyden-Fletcher-Goldfarb-Shanno |
| BIBO | bounded-input bounded-output，有界输入有界输出 |
| BP | backpropagation, 反向传播 |
| CAM | content addressable memory，按内容可寻址记忆 |
| CLS | classical least-squares，经典的最小二乘 |
| CPCA | constrained PCA，约束PCA |
| DOA | direction of arrival, 到达方向 |
| DPC | discrete Picard condition，离散皮卡条件 |
| DTLFNN | distributed time lagged feedforward neural network，分布式的时滞前馈神经网络 |
| EVD | eigenvalue decomposition，特征值分解 |
| FFPA | fast fixed-point algorithm，快速的固定点算法 |
| FFT | fast Fourier transform，快速傅里叶变换 |
| FIR | finite impulse response，有限冲击响应 |
| FMMC | fuzzy min-max classifier，模糊最小-最大分类器 |
| GHA | generalized Hebbian algorithm，广义Hebb算法 |
| GSVD | generalized SVD，广义SVD |
| IC | independent component，独立成分 |
| ICA | independent component analysis，独立成分分析 |
| ILS | inverse least squares, 逆最小二乘 |
| isL | in the sense of Lyapunov，李雅普诺夫意义下 |
| KO | Karhunen-Oja |
| LAPART | laterally primed adaptive resonance theory，侧向初始自适应共振理论 |
| LDU | lower diagonal upper matrix decomposition，矩阵的三角LDU分解 |
| LMBP | Levenberg-Marquardt backpropagation，Levenberg-Marquardt反向传播 |
| LMS | least mean-square，最小均方 |
| LP | linear programming，线性规划 |
| LVQ | learning vector quantization，学习向量量化 |
| LSR | linear shift register，线性移位寄存器 |
| LU | lower upper matrix decomposition，矩阵的三角LU分解 |

（续）

| | |
|---|---|
| Madaline | multiple Adaline，多重自适应线性单元 |
| MAJ | majority logic function，逻辑表决函数 |
| MAW | mountain associated wave，波峰 |
| MDL | minimum description length，最小描述长度 |
| MIMO | multiple-input multiple-output，多输入多输出 |
| MLP | multilayer perceptron，多层感知器 |
| MLP NN | multilayer perceptron neural network，多层感知器神经网络 |
| MRAC | model reference adaptive control，模型参考自适应控制 |
| MRI | Madaline rule I，Madaline 规则 I |
| MRII | Madaline rule II，Madaline 规则 II |
| MSE | mean square error, 均方误差 |
| MUSIC | multiple signal classification，多信号分类 |
| NARMA | nonlinear autoregressive moving average，非线性自回归滑动平均 |
| NARMAX | nonlinear autoregressive moving average with exogenous inputs，具有外部输入的非线性自回归滑动平均 |
| NARX | nonlinear autoregressive with exogenous inputs，具有外部输入的非线性自回归 |
| NGE | nested generalized exemplar，嵌套广义标本 |
| NIPALS | nonlinear iterative partial least squares，非线性迭代的部分最小二乘 |
| NIR | near-infrared，近红外 |
| NLPCA | nonlinear PCA，非线性PCA |
| $NN_c$ | neural network controller，神经网络控制器 |
| $NN_I$ | neural network to perform system identification，执行系统辨识的神经网络 |
| NOR | NOT-OR logic function，非或逻辑函数 |
| NP | nonlinear programming or nondeterministic polynomial (time complete)，非线性规划或非确定多项式（时间完全） |
| OLS | orthogonal least-squares，正交最小二乘 |
| OR | OR logic function，或逻辑函数 |
| PCA | principal component analysis，主成分分析 |
| PCR | principal component regression，主成分回归 |
| PLSNET | partial least-squares regression neural network，部分最小二乘回归神经网络 |
| PLSNET-C | PLSNET-calibration，PLSNET校准 |
| PLSNET-P | PLSNET-prediction，PLSNET预测 |
| PLSR | partial least-squares regression，部分最小二乘回归 |
| PN | pseudo noise，伪噪声 |
| PRESS | predicted residual error sum of squares，预测残量误差平方和 |
| QP | quadratic programming，二次规划 |
| RBF | radial basis function，径向基函数 |
| RBF NN | radial basis function neural network，径向基函数神经网络 |
| RLS | recursive least-squares，递归最小二乘 |
| RMS | root-mean-square，均方根 |
| ROB | robust，健壮，鲁棒 |
| RWLS | recursive weighted least-squares，递归加权最小二乘 |

（续）

| | |
|---|---|
| SEC | standard error of calibration，校正的标准误差 |
| SEE | standard error of estimation，估计的标准误差 |
| SEP | standard error of prediction，预测的标准误差 |
| SISO | single-input single-output，单输入单输出 |
| SGA | stochastic gradient ascent，随机梯度上升 |
| SNR | signal-to-noise ratio，信噪比 |
| SOM | self-organizing map，自组织映射 |
| SOR | successive overrelaxation，逐次超松弛 |
| SRN | simple recurrent network，简单递归网络 |
| SSE | sum-squared error，误差平方和 |
| SVD | singular value decomposition，奇异值分解 |
| TDNN | time delay neural network，时间延迟神经网络 |
| TLS | total least-squares，总体最小二乘 |
| TLU | threshold logic unit，阈值逻辑单元 |
| TSVD | truncated SVD，截断SVD |
| VOL | volcano，火山 |
| wss | wide-sense stationary，宽平稳的 |
| XNOR | exclusive NOR logic function，异或非逻辑函数 |
| XOR | exclusive OR logic function，异或逻辑函数 |

# 目　录

出版者的话
专家指导委员会
译者序
前言
致谢
重要符号和算符
重要缩写词

**第一部分　神经计算的基本概念和部分神经网络体系结构及其学习规则**

第1章　神经计算概述 ……1
1.1　神经计算是什么 ……1
1.2　神经计算的发展历史……3
1.3　神经计算和神经科学……7
1.4　神经网络的分类 ……11
1.5　本书指南……12
参考文献 ……13
第2章　神经计算的基本概念……16
2.1　概述 ……16
2.2　人工神经元的基本模型 ……16
2.3　基本激活函数 ……18
2.4　人工神经元的霍普菲尔德模型 ……23
2.5　自适应线性单元和多重自适应线性单元 ……25
2.5.1　简单自适应线性组合器和LMS算法 ……25
2.5.2　自适应线性单元 ……31
2.5.3　多重自适应线性单元……37
2.6　简单感知器……39
2.6.1　Mays感知器学习规则 ……40
2.6.2　具有S形激活函数的简单感知器……41
2.7　前馈多层感知器 ……44
2.8　单个神经元基本学习规则概述 ……45
2.8.1　广义的LMS学习规则……46
2.8.2　Hebb学习 ……50
2.8.3　Oja学习规则 ……52
2.8.4　位势学习规则 ……54
2.8.5　相关学习规则 ……55
2.8.6　标准感知器学习规则……55
2.8.7　广义感知器学习规则……56
2.9　数据预处理……57
2.9.1　规整 ……58
2.9.2　变换 ……58
2.9.3　傅里叶变换 ……58
2.9.4　主成分分析 ……60
2.9.5　部分最小二乘回归……60
2.9.6　小波和小波变换 ……61
习题 ……61
参考文献 ……66
第3章　映射网络……71
3.1　概述 ……71
3.2　联想记忆网络……71
3.2.1　一般的线性分布式联想记忆 ……72
3.2.2　相关矩阵记忆 ……73
3.2.3　相关矩阵记忆的误差修正方法 ……76
3.3　反向传播学习算法 ……77
3.3.1　前馈多层感知器的基本反向传播算法 ……78
3.3.2　使用标准反向传播中的一些实际问题 ……80
3.3.3　具有动量更新的反向传播学习算法 ……83
3.3.4　批量更新 ……83
3.3.5　搜索然后收敛方法……84
3.3.6　可变学习率的批量更新 ……84
3.3.7　反向传播算法的向量矩阵形式 ……85
3.4　加速学习反向传播算法 ……87
3.4.1　前馈多层感知器的共轭梯度反向传播 ……87
3.4.2　基于最小二乘的递归反向传播

算法 ……91
3.4.3 具有自适应激活函数斜度的反向传播 ……93
3.4.4 Levenberg-Marquardt算法 ……95
3.5 对传 ……98
3.6 径向基函数神经网络 ……101
3.6.1 训练具有固定中心的RBF NN ……102
3.6.2 用随机梯度方法训练RBF NN ……104
3.6.3 正交最小二乘 ……106
习题 ……109
参考文献 ……117
第4章 自组织网络 ……120
4.1 概述 ……120
4.2 Kohonen自组织映射 ……120
4.3 学习向量的量化 ……125
4.4 自适应共振理论(ART)神经网络 ……131
4.4.1 ART1 ……133
4.4.2 模糊ART和模糊ARTMAP ……136
习题 ……137
参考文献 ……142
第5章 递归网络和时间前馈网络 ……144
5.1 概述 ……144
5.2 递归神经网络概述 ……144
5.3 霍普菲尔德联想记忆 ……144
5.4 模拟退火 ……152
5.5 玻尔兹曼机 ……156
5.6 时间前馈网络概述 ……160
5.7 简单递归网络 ……161
5.8 时延神经网络 ……164
5.9 分布式时滞前馈神经网络 ……165
习题 ……167
参考文献 ……173

## 第二部分 神经计算的应用

第6章 用神经网络解决最优化问题 ……177
6.1 概述 ……177
6.2 解决线性规划问题的神经网络 ……177
6.2.1 解决LP问题标准形式的神经网络 ……180
6.2.2 解决LP问题非标准形式的神经网络 ……181
6.3 解决二次规划问题的神经网络 ……186
6.4 解决非线性连续约束最优化问题的神经网络 ……195
6.4.1 罚函数NP方法的神经网络 ……196
6.4.2 障碍函数NP方法的神经网络 ……201
6.4.3 普通拉格朗日乘子NP方法的神经网络 ……201
6.4.4 增广拉格朗日乘子方法的神经网络 ……205
习题 ……209
参考文献 ……212
第7章 用神经网络解决矩阵代数问题 ……214
7.1 概述 ……214
7.2 矩阵的逆和伪逆 ……215
7.3 LU分解 ……220
7.4 QR因子分解 ……223
7.5 舒尔分解 ……227
7.6 谱因子分解——特征值分解（EVD）（对称特征值问题） ……229
7.7 对称特征值问题的神经网络方法 ……230
7.8 奇异值分解 ……235
7.9 求解代数李雅普诺夫方程的神经计算方法 ……239
7.10 求解代数里卡蒂方程的神经计算方法 ……241
习题 ……245
参考文献 ……249
第8章 使用神经网络求解线性代数方程组 ……251
8.1 概述 ……251
8.2 联立线性代数方程组 ……251
8.3 线性方程组的最小二乘解 ……253
8.4 求解线性方程组的最小二乘神经计算方法 ……254
8.5 求解线性方程组的共轭梯度学习规则 ……258
8.6 求解受噪声侵扰的线性方程组的广义鲁棒方法 ……260
8.7 带病态确定数值秩的不适定问题的正则化方法 ……266
8.8 求解线性方程组的离散时间迭代方法

中的矩阵分裂 ……273
8.9 总体最小二乘问题 ……278
8.10 求解线性方程组的$L_\infty$范数（最小最大）神经网络 ……280
8.11 求解线性方程的$L_1$范数（最小绝对偏差）神经网络 ……282
习题 ……285
参考文献 ……290
第9章 使用神经网络的统计方法 ……294
9.1 概述 ……294
9.2 主成分分析 ……294
9.3 神经网络自适应主成分估计的学习算法 ……297
9.3.1 第一主成分估计——Oja的正规化Hebb学习规则 ……297
9.3.2 多个主成分估计——对称子空间学习规则 ……301
9.3.3 多个主成分估计——广义Hebb算法 ……303
9.3.4 多个主成分估计——随机梯度上升算法 ……305
9.3.5 多个主成分估计——自适应主成分提取算法 ……305
9.3.6 非线性主成分分析（NLPCA）和鲁棒PCA ……312
9.4 主成分回归 ……316
9.5 部分最小二乘回归 ……322
9.6 部分最小二乘回归的神经网络方法 ……327
9.7 鲁棒PLSR：一种神经网络方法 ……333
习题 ……337
参考文献 ……340
第10章 使用神经网络进行辨识、控制和估计 ……348
10.1 概述 ……348
10.2 线性系统的表示法 ……348
10.3 自回归滑动平均模型 ……349
10.4 用ARMA模型的线性系统辨识 ……349
10.5 应用PLSNET进行线性系统的参数系统辨识 ……352
10.6 非线性系统的表示法 ……355
10.6.1 非线性输入-状态-输出表示法 ……355
10.6.2 非线性ARMA ……356
10.7 非线性动态系统的辨识和控制 ……359
10.7.1 非线性系统的辨识 ……360
10.7.2 非线性控制 ……365
10.8 独立成分分析：未知源信号的盲分离 ……371
10.8.1 独立成分分析的概述 ……371
10.8.2 用神经网络进行独立成分分析 ……372
10.8.3 用于ICA的快速固定点算法 ……381
10.9 可加噪声中的正弦曲线的谱估计 ……389
10.9.1 问题描述 ……389
10.9.2 频率估计问题的PLSR解 ……391
10.10 其他案例分析 ……395
10.10.1 从近红外谱模拟数据估计葡萄糖浓度 ……395
10.10.2 使用次声数据进行事件分类 ……399
习题 ……402
参考文献 ……407
附录A 神经计算的数学基础 ……411
A.1 引言 ……411
A.2 线性代数 ……411
A.2.1 域和向量空间 ……411
A.2.2 矩阵的表示和运算 ……413
A.2.3 内积和外积 ……414
A.2.4 向量的线性无关 ……415
A.2.5 矩阵的秩和线性无关 ……415
A.2.6 矩阵的确定性 ……415
A.2.7 矩阵的逆和伪逆 ……416
A.2.8 正交矩阵、酉矩阵和共轭向量 ……418
A.2.9 特征值和特征向量 ……418
A.2.10 相似变换 ……421
A.2.11 若当标准形 ……421
A.2.12 动态系统的状态空间描述 ……422
A.2.13 向量和矩阵的范数 ……426
A.2.14 奇异值分解 ……428
A.2.15 矩阵条件数 ……430
A.2.16 分块矩阵运算 ……431
A.2.17 克罗内克积与和 ……433
A.2.18 实数和复数方阵的重要性质小结 ……436
A.2.19 模式化矩阵和特殊矩阵 ……437
A.3 多变量分析的原理 ……440

A.3.1 集合和函数 ……440
A.3.2 二次型 ……442
A.3.3 链式法则 ……442
A.3.4 矩阵微积分 ……443
A.3.5 黑塞矩阵 ……445
A.3.6 雅可比矩阵 ……446
A.3.7 泰勒级数展开式 ……447
A.4 李雅普诺夫直接法 ……447
A.5 无约束最优化方法 ……449
A.5.1 极值的充分必要条件 ……449
A.5.2 最速下降法 ……450
A.5.3 牛顿法 ……451
A.5.4 改进的牛顿法和拟牛顿法 ……452
A.5.5 共轭梯度法 ……453
A.6 约束非线性规划 ……455
A.6.1 库恩-塔克条件 ……455
A.6.2 拉格朗日乘子法 ……456
A.7 随机变量和随机过程 ……458
A.7.1 随机变量 ……458
A.7.2 概率分布函数 ……460
A.7.3 概率密度函数 ……460
A.7.4 期望值、均值和矩 ……461
A.7.5 随机过程 ……463
A.7.6 向量随机过程 ……466
A.7.7 功率谱密度函数和功率谱密度矩阵 ……466
A.7.8 白噪声驱动的线性系统和谱因子分解 ……467
A.8 模糊集合论 ……469
A.9 部分三角恒等式 ……469
参考文献 ……471
主题索引 ……473

# 第一部分　神经计算的基本概念和部分神经网络体系结构及其学习规则

## 第1章　神经计算概述

### 1.1　神经计算是什么

神经计算与信息处理有关。与相对应的程序计算不同，用神经计算方法进行信息处理的方法是：首先在人工神经网络（或神经网络）[⊖]体系结构内进行学习，这个体系结构按照学习规则对输入作出相应的响应。在神经网络已经学会需要知道什么以后，训练后的神经网络依据特定应用可以用来执行特定的任务。神经网络具有从环境中学习的能力和以类似生物的交互方式适应环境的能力。确实，因为存在用人工神经网络来执行某些功能的巨大可能性，使得用人工神经网络（在一定程度上）能够模仿与生物类似的功能。这是一个激动人心的前景。

例如，一个人能够相当好地完成模式识别任务。我们在街上“看见”一辆经过的汽车，它“吸引我们的注意力”。可能汽车正飞快地行驶，但我们能够识别车体式样的突出特征，把它和我们头脑中记忆的一辆1984年的红色法拉利跑车（Ferrari Testarossa）的图像联想在一起。这个例子清楚地说明我们仅仅捕获汽车的瞬间状态，就足以正确地识别车型。我们认为这个过程是理所当然的，然而，它是非常复杂的。我们为了正确地识别车型而细察车的侧放和车每分钟的详情并不是必要的。正相反，对运动中的汽车的快速一瞥（如观察它的独特一侧散热窗和后挺站立的马形商标）是足够做出正确辨识的。这类模式识别能够由人工神经网络来完成，特别由一个霍普菲尔德（Hopfield）（参考5.3节）神经网络来完成。这种网络的有力特征之一是能够从给出的局部输入数据回想起一个存储记忆的能力，例如，一侧散热窗和在红色法拉利跑车的站立的马形商标。

虽然人不如数字计算机快或精确（例如，一台数字计算机在做2个7位数乘法时比人快得多），但是人在感知和识别自然界场景中感兴趣的物体、解释自然语言和其他许多自然界的认知任务上比数字计算机好得多。人们执行这些任务为什么比数字计算机好得多？虽然人们仍旧从总体上不知道这个问题的答案，但是人们已经了解到足够的知识使得人们执行得非常好的某些功能能够被人工神经网络模仿。例如，为什么人们在复杂情境下能更好地识别物体？原因之一是由于人脑的组织方式。人脑结构适用于解决非常复杂的问题，而数字计算机解决这些复杂问题很困难，需要很长时间的计算。人脑的基本处理单元是神经元（神经细胞），而在数字计算机中用于计算的单元是硅做的逻辑门。神经元大概比硅逻辑门事件慢六个数量级。然而，人脑通过以一种大量互连的高度并行结构进行数据处理以弥补神经元相对较慢的运算速度。据估计人脑包含有大约$10^{11}$数量级的神经元和大约比神经元数量级的3倍还多的连接或突触。因此，人脑是一个自适应的、非线性的和并行的计算机。能组织神经元完成特定任务。用神经计算系统可以很粗略地模拟人脑。虽然对人脑建模领域的研究很有趣并且激动人心，

⊖　人工神经网络也称为连接网络（或系统）、并行分布式处理系统或神经计算机。

但是这不是本书的要旨。1.3节给出了生物神经网络的总体概述；然而，这样做是为了内容完整性，并给出一些与人工神经网络有直接关系的神经生物概念。我们从工程师的角度来描述人工神经网络，然而，这些素材对科学家和工程师都可用。因此，本书不仅适用于那些想了解用于神经计算的人工神经网络的基本原理的读者，也适用于那些希望能应用各种神经计算技术解决科学和工程中的问题的读者。

通过例子学习（learn）和泛化（generalize）的能力是人工神经网络的主要特征。根据学习规则（算法），通过给出一些必须学会的模式来训练神经网络。在上面的例子中，首先能够
4 认出红色法拉利跑车，意味着在以前已经观察到这类汽车至少一次，并且知道它是什么车型（训练过程）。学习过程使得神经网络和数字计算机处理信息不一样，后者需要被程序化。假如神经网络已经被泛化，这就意味着神经网络能把输入模式分类为一个可接受的精确水平，即使这些输入模式在训练过程中从来没有使用过。上面例子中的红色法拉利跑车，也许识别者以前从未看见过，但是曾经观察过与它类似的其他车。

人工神经网络将在训练过程中学到的知识存储在神经元的突触权值中。任何人工神经网络的构件是人工神经元[⊖]。怎样组织神经元的网络，如何规定突触权值调整的学习规则，以及如何确定训练过程什么时候完成的标准，所有这些都表征了特定类型的神经网络。神经网络的种类有很多，它们具有不同程度的复杂性；然而，各种神经网络具有相似的特征。比如，使之能快速计算的基本并行计算结构可能是几乎所有神经网络类型的共同点。大多数人工神经结构具有多个高度互联的（很多突触）神经元，和相对应的生物神经元一样。神经元输出中的非线性也是一个神经网络中的显著部分，虽然也有几种神经网络类型是“线性”结构的。例如，在第7章中给出的结构化网络都是线性网络。

现在，神经网络的领域非常宽广且涉及多种学科，吸引了许多不同领域的研究者的兴趣；比如，工程学（包含生物医学工程）、物理学、神经学、心理学、医学、数学、计算机科学、化学和经济学。人工神经网络为解决复杂问题提供了一个神经计算的方法，而该问题用其他方法可能得不到一个易处理的解决办法。神经网络的应用包括（但不局限于）：预测与预报、联想记忆、函数逼近、聚类、数据压缩、语音识别与合成、非线性系统建模、非线性控制、模式分类、特征提取、组合优化、矩阵代数问题求解、盲源分离和微分方程求解。利用神经计算方法解决某些问题有许多优点，比如具有容错能力。神经网络的硬件实现往往具有内在的容错能力。假如神经网络结构有一个神经元损坏，或者是连接损坏，整个网络的性能通常只会受到这种损害的轻微影响。这对于神经网络的鲁棒特性是发挥作用的因素。因为信息分布在神经网络中，所以能使得神经网络完全失败的损害应该是非常严重的（即许多被损
5 害的神经元和/或连接）。如上所述，人工神经元通常是非线性的；因此，网络自身是一个非线性系统。许多时候一个非线性系统被视为灾难性的；然而，神经网络的非线性是非常重要的特征，尤其是当相关的物理过程是完全非线性的，并且将测量该系统得到的量值用于训练神经网络。神经网络的自适应特性也是一个重要性质。由于网络的突触权值的自适应性，网络能够和它的环境相互作用并且做出响应。因此，如果在最初的训练后环境改变了，网络能对环境变化做出响应，并且能进行基本的重新自我训练。更进一步，对于非稳定环境，神经网络能设计成实时执行自身权值的自适应。神经网络擅长于执行如像自适应性控制、自适应模式分类和自适应信号处理这样的功能。因此，在本书中，许多内容描述了自适应信号处理的方法。

---

⊖ 人工神经元也称作（信息）处理元素、单元、细胞、节点和神经节（neurodes）。

我们已经定义了人工神经网络是什么，以及利用神经计算方法解决某些问题的优点，下面我们将给出一些有关神经网络领域的历史背景。这不是关于神经网络历史的综合论述，仅希望读者适当了解本书内容的背景。

## 1.2 神经计算的发展历史

本节并不打算给出神经网络的完整历史，而是介绍一些主要的研究成果，这些成果为其他人开辟了道路。早期重要成果的概览给读者提供如何认识在过去岁月中某些成就导致该领域的发展。早期人工神经网络研究的结果在参考资料[1-4]中可以找到。

历史概览

• McCulloch和Pitts, 1943

据说神经网络的近代史开始于Warren S. McCulloch 和Walter Pitts 1943[5]的工作。他们给出了5种支配神经元运行的假设。这些假设描述了现在熟知的McCulloch-Pitts神经元。这些神经元没有训练，然而，能够实现某些逻辑功能。McCulloch-Pitts神经元模型为神经网络未来的发展打下了基础。

• Hebb, 1949

在1949[6]的文章中Donald Hebb描述了一个从神经生物观点假设的学习过程。Hebb认为 6
信息存储在神经元之间的连接中，并且提出一个调节连接权值的学习策略。这是第一次给出允许调节突触权值的学习规则，对该领域后来的工作具有重要的影响。

• von Neumann, 1958

John von Neumann在数字计算机的发展方面起了重要作用，是20世纪前期科学史上的重要人物之一。他也是曼哈顿工程的骨干成员。他从计算机的最早时期开始就看到了数字计算机与人脑的潜在相似性[7]。例如，他提及在生物神经系统中记忆的重要性如同在电子系统中一样。

• Rosenblatt, 1958

在1958年，Frank Rosenblatt发展了感知器（perceptron）的最初概念[8]。它是第一个准确定义的面向计算的神经网络。它的复杂的适应行为引起了工程师的极大注意。它是一个可以训练用来区分某些模式的机器。

• Widrow和Hoff, 1960

Bernard Widrow的Adaline（自适应线性单元，adaptive linear element）使用最小均方（LMS）学习规则[9]来训练，与Rosenblatt的感知器极为相似。Adaline推广至多个Adaline（Madaline）。Adaline和Madaline有许多应用，如在自适应控制和模式识别方面。

• Minsky和Papert, 1969

在1969年（以及之后数年），Marvin Minsky和Seymour Papert 延缓了神经网络的研究，当年在他们的《Perceptrons》（感知器）[10]书中指出单层神经网络能力有限。一个这样的例子是异或（XOR）问题。Rosenblatt已经研究了具有多层神经元的体系结构，并且相信它们能够克服简单感知器的局限性；然而，当时不知道学习规则[11]。Minsky和Papert怀疑能够发现这种学习规则。

• Kohonen, 1972和Anderson, 1972

尽管在20世纪70年代神经网络研究缓慢，仍有人继续研究。最主要的领域之一是按内容可寻址联想记忆。在1972年，Teuvo Kohonen发表了关于相关矩阵记忆的论文[12]。同年，James Anderson独自提出了和Kohonen同样的模型[13]。

• von der Malsburg, 1973

Christoph von der Malsburg的开创性工作研究具有突触修正规则的网络，该规则可以产生一个能显示自我修正和组织能力的模型皮层[14]。他的工作受到20世纪70年代早期动物实验的启发。

• Werbos, 1974

在1974[15]年，Werbos给出了用于训练多层前馈感知器的反向传播算法的首次描述。

• Little和Shaw, 1975

7 Little和Shaw在论文[16]中描述了使用概率神经元模型代替确定性神经元模型的神经网络。

• Lee和Lee, 1975

Lee和Lee介绍了模糊McCulloch-Pitts神经元模型[17]。

• Grossberg, 1976

在1976年，Grossberg受皮层组织的发展生理学启发而发表的一篇文章提出了一种理论分析[18]。他陈述了视觉皮层的特征探测器响应环境而发展和变化的有力证据。

• Amari, 1977

Amari在1977[19]论文中讨论了模式联想器（pattern associator）。在一种类型的联想器中，输入模式激发一个适当的、但不同的输出模式。在另一种类型的联想器中，使用他称为概念形成（concept-forming）的网络（这些是递归网络），输入和输出模式是相同模式，并且输出模式能够反馈到网络的输入。

• Hopfield, 1982

据说，神经网络的现代史开始于John Hopfield[20]（诺贝尔物理学奖获得者）论文的发表。Hopfield复杂而综合地描述了递归神经网络的运行方式及其功能。网络能够在动态稳定的环境下存储信息（如模式），并且能够执行数据存储和检索的功能。对网络给定一个有噪声的输入，尽管呈现给网络的是模式的不完整（受损的）版本，但网络能够恰当地检索出存储在记忆中的相关模式。

• Kohonen, 1982

在1982[21]中，Teuvo Kohonen介绍了自组织的特征映射。它是一个无监督的、竞争学习的聚类网络，同一时刻只有一个神经元（或一组中只有一个神经元）是“激活”的。

• Oja, 1982

Erkki Oja介绍了如何使用规范的Hebb学习规则来训练单个线性神经元，使它成为一个主成分分析器[22]。这个神经元能够从输入数据中自适应地提取第一主特征向量。他的工作的推广导致不同的神经网络方法用于自适应地估计多重主特征向量。

• Fukushima, Miyake和Ito, 1983

神经认知机（neocognitron）是由Fukushima、Miyake和Ito开发的[23]，用于字符识别。

• Kirkpatrick, Gelatt和Vecchi, 1983

虽然Kirkpatrick、Gelatt和Vecchi的论文[24]并不是一篇神经网络的论文，但它为Boltzmann机做好了铺垫。

• Kampfner和Conrad,1983

Kampfner和Conrad研究了用于神经网络训练的进化计算方法[25]。进化计算的历史在D. B. Fogel的著作[26]中有详细的阐述。

• Ackley, Hinton和Sejnowski, 1985

Ackley、Hinton和Sejnowski在1985[27]给出了Boltzmann机的学习算法。

• Parker, 1985和LeCun, 1985

反向传播训练算法分别被Parker[28]和LeCun[29]独立发现。 8

• Herault, Jutten和Ans, 1985

Herault、Jutten和Ans将神经网络用于独立源信号的盲分离[30]，即用于盲源分离的独立成分分析(independent-component analysis,ICA)的神经网络实现[31]。

• Rumelhart, Hinton和Williams,1986

反向传播训练算法分别被Rumelhart、Hinton及Williams独立发现[32,33]。

• Carpenter和Grossberg, 1987

Carpenter和Grossberg在自适应共振理论（adaptive resonance theory, ART）基础上发展了自组织神经网络[34]。

• Sivilotti, Mahowald和Mead,1987

神经网络的第一次VLSI实现归功于Sivilotti、Mahowald和Mead[35]。

• Broomhead和Lowe,1988

在神经网络设计中第一次探索径向基函数归功于Broomhead和Lowe[36]。

McCulloch-Pitts神经元

正如前文所述，在神经网络研究的开头，1943年Warren S. McCulloch和Walter Pitts的工作本质上开启了神经网络研究的新时代。因此，以他们的神经元概念作为开始看起来是适当的。

McCulloch-Pitts神经元是一个非常简单的双态（two-state）装置。它为开或关；也就是说，输出是二值的。一个特定神经元的输出不能连接其他神经元的输出；然而，它能分支到其他神经元和作为该神经元的输入而终止，或它自身终止。以下两种类型的终止都是允许的：兴奋性的（excitatory）输入或抑制性的（inhibitory）输入。一个神经元可以有任意数目的输入。神经元无论是开（点火，firing）还是关（寂静，quiet），取决于神经元的阈值（threshold）。下列是对简单神经元做出的物理假设[5]：

1. 神经元活动是一个全或无（all-or-nothing）的过程，即神经元的激活是二值的。在神经元“点火”的任何离散时间步，激活是1；当它是“寂静”的时间步，激活是0。这是神经元两种可能的状态。

2. 为了使神经元兴奋，一定数目的固定（加权）神经元突触在一个离散时间步内必须是兴奋的，并且这个数目独立于任何先前的活动。

3. 在“神经系统”内唯一有意义的延迟是突触延迟，即花费在突触上传播信息的时间。

4. 任何非零抑制性突触的活动将在离散时间步中绝对阻止神经元兴奋。

5. 神经网络的结构并不随时间而改变。 9

图1-1显示一个McCulloch-Pitts神经元的例子。从$x_1$到$x_n$的连接是兴奋性输入，这是由于突触权值$w$是正数。从$x_{n+1}$到$x_{n+m}$的连接是抑制性的，这是由于突触权值$(-c)$是负数。所以，对神经元$y$有$n$个兴奋性输入和$m$个抑制性输入。在一个离散时间步内，信号通过神经元输入（$x_1$到$x_{n+m}$）到$y$。神经元$y$在离散时刻$k$时的状态取决于在时刻$k-1$时的$x_1$到$x_{n+m}$的输入。接受到的总的输入信号是$u$，如果$u \geqslant \theta$，那么神经元输出是$y = 1$（即神经元输出是兴奋，或点火）；然而，如果$u < \theta$，那么$y = 0$（即神经元是抑制，或寂静），也就是说：

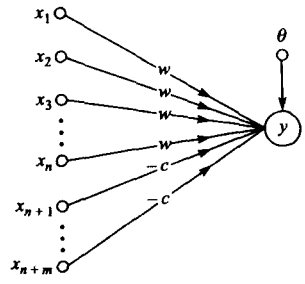



图1-1 阈值为$\theta$的McCulloch-Pitts神经元$y$的体系结构

$$y=\begin{cases}1 & \text{当}u\geqslant\theta\\ 0 & \text{当}u<\theta\end{cases} \tag{1-1}$$

神经元的绝对抑制状态需要下面条件成立：

$$nw-c<\theta \tag{1-2}$$

这是前面陈述的假设4的直接结果；也就是说，任何非零抑制突触将绝对禁止神经元兴奋。（它仅取一个$c$!）如果神经元的阈值设置为$\theta=hw$（$h$是一个整数），那么如果$hw\geqslant\theta$，神经元$y$将点火，也就是说，如果$h$个或更多（最多为$n$）兴奋输入被接受而没有任何抑制输入。通过设置权值和McCulloch-pitts神经元的阈值，能够实现一些简单的逻辑（布尔）函数。

**例1.1**　实现AND逻辑函数可使用具有相等权值为1的两个突触连接和阈值$\theta=2$的McCulloch-Pitts神经元（参见图1-2）。仅当两个输入均为开时，AND逻辑函数产生为真（true）的响应；否则，神经元响应为假（false）（即一个或两个输入都是关，也称为抑制）。所以，真响应表示为1，而假响应表示为0。与AND逻辑函数相关的“真值”表如下所示：

10

**AND逻辑函数**

| $x_1$ | $x_2$ | → | $y$ |
|---|---|---|---|
| 1 | 1 | | 1 |
| 1 | 0 | | 0 |
| 0 | 1 | | 0 |
| 0 | 0 | | 0 |

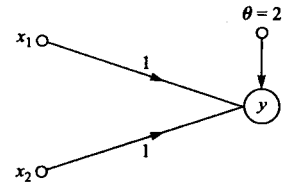



图1-2　用于执行AND逻辑函数的McCulloch-Pitts神经元

上面的AND逻辑函数表内的“模式”可以成为不同神经网络的训练模式（记住，这里为McCulloch-Pitts神经元，它的所有权值和阈值都预先设置）。特别是，$x$为训练输入模式，相应的$y$为目标（期望输出）值。在后面讨论神经网络的监督训练时我们将深入探讨这一点。

**例1.2**　实现OR逻辑函数同样可使用两个突触连接的McCulloch-Pitts神经元；然而，现在权值设置为2，阈值仍设置成$\theta=2$（参见图1-3）。如果两个输入中的一个或者两个同时是开，那么神经元响应总是1（真）；但是，如果两个输入都是关，神经元响应该是0（假）。这也称为同或（OR）逻辑函数，因为两个输入同时是开（真），神经元输出是开（真）。OR逻辑函数的真值表如下所示：

11

**OR逻辑函数**

| $x_1$ | $x_2$ | → | $y$ |
|---|---|---|---|
| 1 | 1 | | 1 |
| 1 | 0 | | 1 |
| 0 | 1 | | 1 |
| 0 | 0 | | 0 |

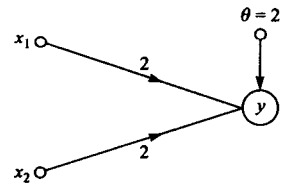



图1-3　用于执行OR逻辑函数的McCulloch-Pitts神经元

**例1.3**　McCulloch-Pitts神经元更多采用XOR（异或）逻辑函数。实际上采用三个神经元来实现这个逻辑函数。XOR逻辑函数跟前面的（同或）OR逻辑函数唯一的不同是：只有当输入的单个值是开（或真）时，神经元点火（即神经元产生一个真响应1）；否则，响应是假（或关）。因此，这个函数的真值表如下：

图1-4的神经网络是两层网络。我们来看在$x_1$和$x_2$均为1时的情况。因为在神经元$x_3$和$x_4$接受到的总输入信号小于阈值：$u_3 < \theta$和$u_4 < \theta$（即$1 < 2$），所以，在神经网络中间层$x_3 = x_4 = 0$。从而，由于$x_3$和$x_4$均是抑制的，所以输出$y = 0$，这一点精确地显示在上面的XOR逻辑函数表中。如果两个输入是关，即$x_1 = x_2 = 0$，显然$y = 0$（上面的XOR逻辑表所示）。让我们分析一下在图1-4中如果$x_1 = 1$，并且$x_2 = 0$会发生什么。在$x_3$，由于该神经元总的接收输入信号等于神经元阈值，神经元响应为开；所以，$x_3 = 1$。然而，在$x_4$，结论为真，即$x_4 = 0$，这是由于该神经元接受的总输入信号小于神经元的阈值。总之，网络输出（响应）将会是$y = 1$（激活，或真），这是由于该神经元接受的总输入信号等于神经元的阈值。对于输入$x_1 = 0$和$x_2 = 1$，与上面情况类似，结果是相同的，即$y = 1$。这就完成了上面的XOR逻辑函数真值表。

XOR逻辑函数

| $x_1$ | $x_2$ | → | $y$ |
|---|---|---|---|
| 1 | 1 | | 0 |
| 1 | 0 | | 1 |
| 0 | 1 | | 1 |
| 0 | 0 | | 0 |

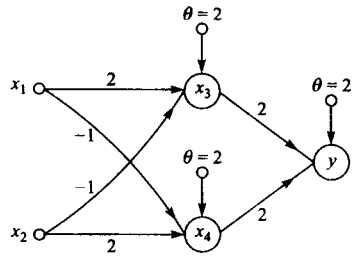



图1-4 用于执行XOR逻辑函数的McCulloch-Pitts神经网络

如同我们将在第2章中看到的那样，奠定神经计算基础的基本概念都来自简单的McCulloch-Pitts神经元。从McCulloch和Pitts的早期研究以来，神经网络研究已经有巨大的进展。这确实是一个多学科性的主题，我们没有理由不相信，这种趋势将继续下去，从而神经网络领域中将产生新的理论以及新的应用。 12

## 1.3 神经计算和神经科学

如前所述，神经计算系统是对人脑非常粗糙的建模。事实上，神经网络领域的许多研究人员并不承认在神经科学与神经计算之间有任何联系。然而，神经科学的研究促进了神经计算的许多发展。因此，在此有必要介绍生物神经网络的概况。在神经科学中有许多综合研究报告，我们的目的并不是给出神经科学领域全面或广泛的编年史，然而，我们确实想给出生物神经网络的相关资料细节，以便使读者了解某些人工神经网络概念同它们的生物对应概念之间的联系。

### 生物神经网络

神经系统是巨大而复杂的神经网络。人脑是神经系统的中心元素。人脑同感受器(receptor)相连接，感受器来回输送感觉信息给人脑；而人脑传递动作命令给效应器(effector)。人脑本身是由大约$10^{11}$个神经元组成的网络，这些神经元通过称为细胞核的子网络相互连接。细胞核由一系列有特定的确定功能的神经元簇（束）构成。子网络对其所收到的感觉信息在发送到其他子网之前通常要分割和修改。经过处理的信号的最终形式传递给效应器以产生一个动作。

人脑中的感觉系统和子网络非常擅长把复杂的感觉信息分解成为具有本质感觉特征的基本组成部分。对每一种官能这些分解是不同的。例如，眼睛和脑依据颜色、强度、指向特性、动作、比例、双眼视觉特性来分割一个视觉图像。这些成分不是重构成原始的视觉形式，而

是传递到其他子网络进行选择性的评价和局部重构。

生物神经元对于人工神经元（处理单元）具有启发作用。因此，两者具有相近的结构。Fausett[37]讲述了生物上的可信性对最有效地模拟生物神经元和人工神经元的重要性。人脑和神经系统的其他部分由许多不同种类的神经元构成，这些神经元的电学性质、数量、大小以及连接模式方面均显著不同（参见图1-5和表1-1）。这些差异经常相当大。

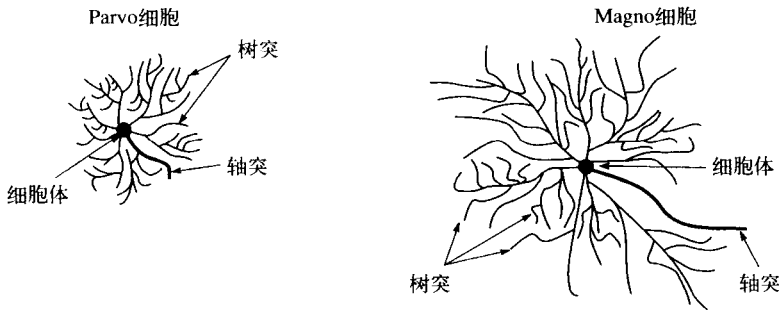



图1-5　parvo（小）和magno（大）细胞的例子

**表1-1　parvo和magno神经节细胞间挑选的解剖学和生理学上的差异以及一些行为的可能后果**

| | Parvo神经节细胞 | Magno神经节细胞 |
|---|---|---|
| 解剖学上的差异 | 细胞体积小 | 细胞体积大 |
| | 分叉稠密 | 分叉稀疏 |
| | 分叉短 | 分叉长 |
| | 多数细胞 | 少数细胞 |
| 生理学上的差异 | 传导率慢 | 传导率快 |
| | 持续响应 | 短暂响应 |
| | 接受域小 | 接受域大 |
| | 低对比敏感度 | 高对比敏感度 |
| | 颜色敏感 | 色盲 |
| 可能的行为后果 | 细节形式分析 | 运动检测 |
| | 空间分析 | 时间分析 |
| | 颜色视觉 | 深度感知 |

生物神经元由三个主要部分构成：树突、细胞体和轴突，参看图1-6a。树突收到来自其他神经元的信号。其他神经元的轴突通过称为突触的连接器把树突和细胞体表面连接起来。依据神经元的类型，同其他神经元相连的突触连接数目从几百个到10 000个不等[39]。因为神经元膜的电学性质，到达树突的信号随时间和距离（时间和空间）迅速衰减，因此失去激活神经元的能力，除非被其他的几乎同时并且/或者邻近位置发生的信号所增强，参见图1-7。

13
~
14

细胞体（soma）叠加来自于树突的信号，也叠加来自于它表面的大量突触的信号。当收到的输入足以刺激神经元到达它的阈值时，神经元产生动作位势，即点火，并沿着轴突传递动作位势给其他神经元或神经系统外的目标细胞，如肌肉。然而，假若输入没有到达阈值，则输入将迅速衰减并且不产生动作位势，参看图1-8a。因此，动作位势的产生认为是全或无，

因为轴突或者产生，或者不产生。而且，输入的强度由每秒产生动作位势的数量而不是大小来表示。例如，强输入和弱输入信号产生的动作位势的大小和形状是一样的。然而，强输入比弱输入每秒（单位时间）产生动作位势的数量多。

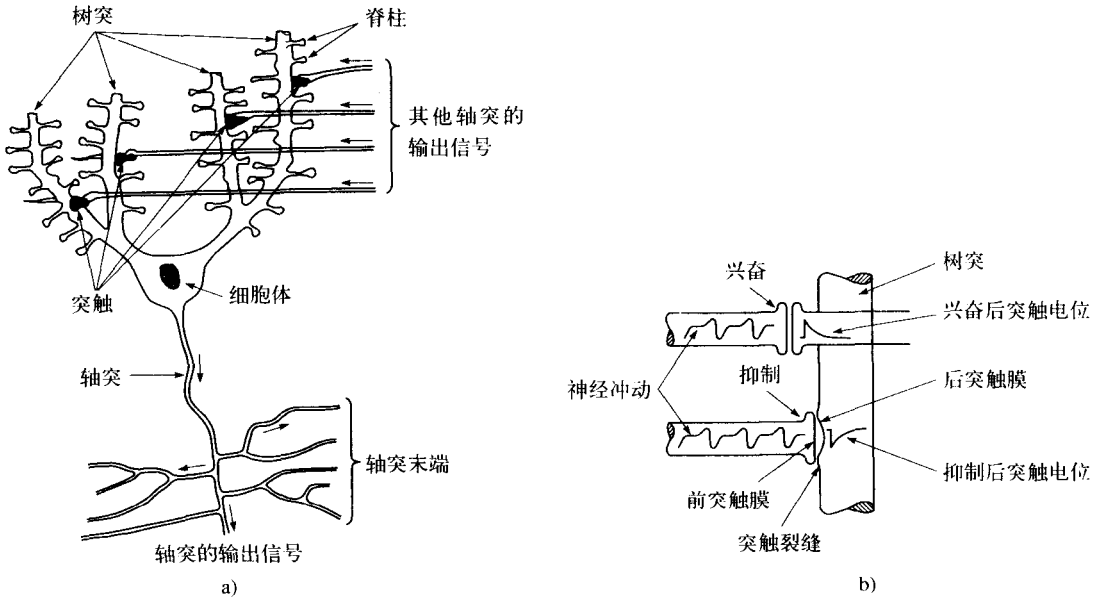



图1-6 a) 生物神经元的示意性结构；b) 突触的简化图示

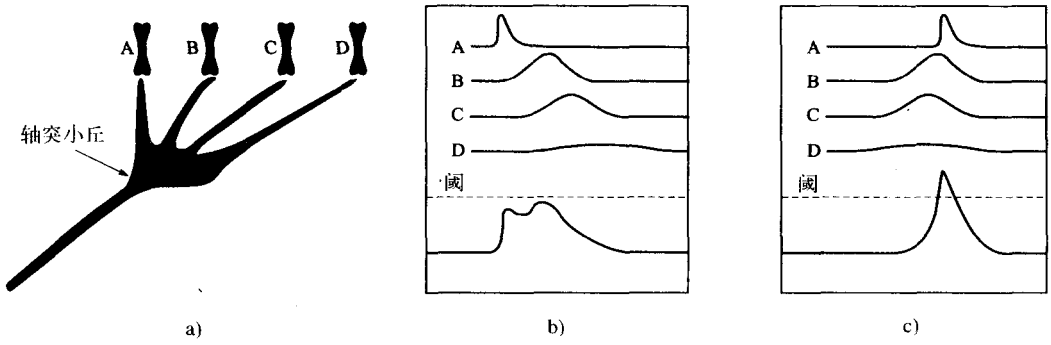
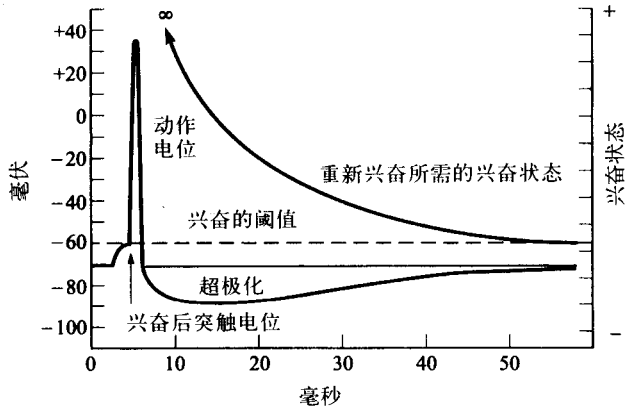


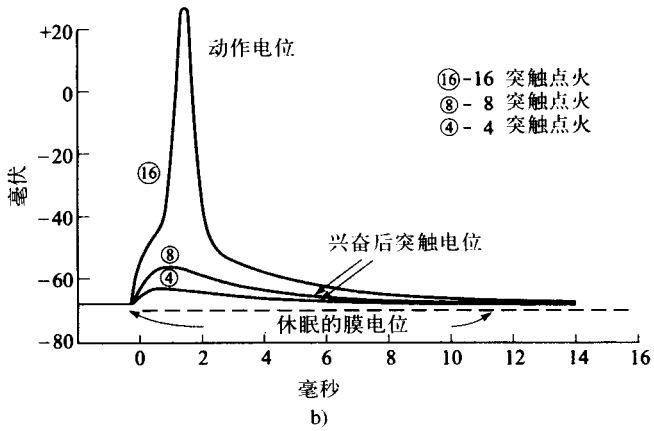

图1-7 a) 因为具有树突的几何形状而用作运动检测器的神经元；b) 使*A*, *B*, *C*, *D*同时兴奋并不能使神经元兴奋；c) 然而，如果突触按*D*-*C*-*B*-*A*的顺序激活，使四个顶点近似同时达到轴突的小丘，组合效果将超过阈值，从而神经元激活

突触是连接轴突末端到目标的接触点。这些特殊化的结构，如连接轴突到树突、细胞体、神经末梢、肌肉或腺，是由特定结构的、化学的和电学特性所表征。突触由三个基本元素构成：（1）神经末梢，（2）突触裂缝或间隙，（3）后突触膜，参看图1-9。当动作位势到达（“侵袭”）神经末梢时，神经末梢通过一系列的生化事件将电信号转化成化学信号。在转化的最后阶段神经末梢释放一种称为神经传递素的化学物质。神经传递素对突触膜起作用。神经系统释放许多不同的神经传递素。然而，单个的神经末梢仅释放一种神经传递素。在大约2毫秒（ms）时间内，神经传递素通过突触裂缝扩散，而突触裂缝是相对开阔的空间。每个神经 15

传递素附着在称为接受器的特定绑定点，而接受器是嵌入后突触膜的。当神经传递素附着在接受器上时，将触发电的和生化的响应，导致后突触膜上电压的变化。有两类主要的神经传
16 递素：兴奋和抑制。兴奋神经传递素去极化隔膜，但单个突触太小，不能由自身产生一个动作位势。当对数百个其他突触同时发生的去极化进行叠加后，能够集中产生一个动作位势。抑制神经传递素产生相反的效果，使后突触膜超极化，从而取消兴奋神经传递素的动作，并且在某些情况下阻止动作位势的产生。因此，动作表示神经元兴奋活动的和。相应地，动作电位传递到轴突末端，在轴突末端神经元之间通过突触进行通信。突触充当直接连接，使神经元之间建立神经回路。




a)




b)

图1-8　a) 动作位势的例子；b) 兴奋的后突触位势，表明仅仅一些同时激活的突触并不产生足够的总电压。然而，许多同时激活的突触将提升总电压到兴奋的阈值，从而引起一个叠加的动作位势

17 作为人工和生物型神经网络相似性的一个例子，表1-2比较了三个不同的神经体系结构。

**表1-2　神经网络结构和它们的神经系统对应物的例子**

| 神经网络 | 神经系统 |
|---|---|
| 单层前馈网络 | 除了在最简单的反射弧路径外少见 |
| 多层前馈网络 | 常见且复杂；通常具有几个隐藏层。局部连接比全连接更常见 |
| 递归网络 | 负反馈；比神经网络的类比物复杂得多。神经系统的一个本质和显著特征 |

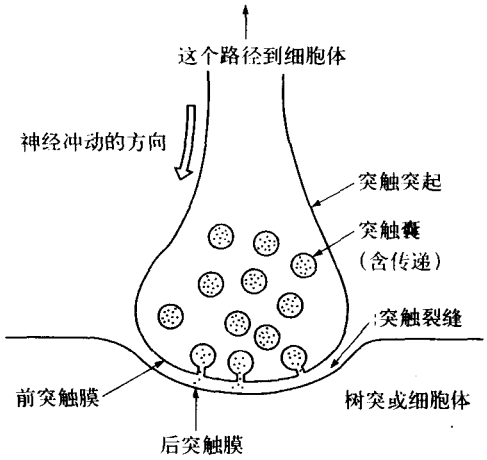



图1-9 一个典型的突触，突触囊储藏着传导物质，这些物质穿过突触的裂缝来释放

## 1.4 神经网络的分类

人工神经网络有多种不同分类方法。例如：按照神经网络学习或需要的训练类型，或者能够完成的各种应用，使用的是激活函数还是基函数，是递归的还是非递归的，以及训练的输入类型[43]等等。所以，神经网络的分类很复杂。例如，神经网络如果按照无监督学习还是监督学习来分类，那么某些网络可能具有多种类别。图1-10显示了神经网络不同类型及其如何按照学习类型（即无监督学习还是监督学习）分类的例子。在这个例子中就无法放置对传网络（参考3.5节），另外也无法放置霍普菲尔德网络（参考5.3节），这个网络是一个递归神经网络，实际上被认为是一个固定权值的网络。 18

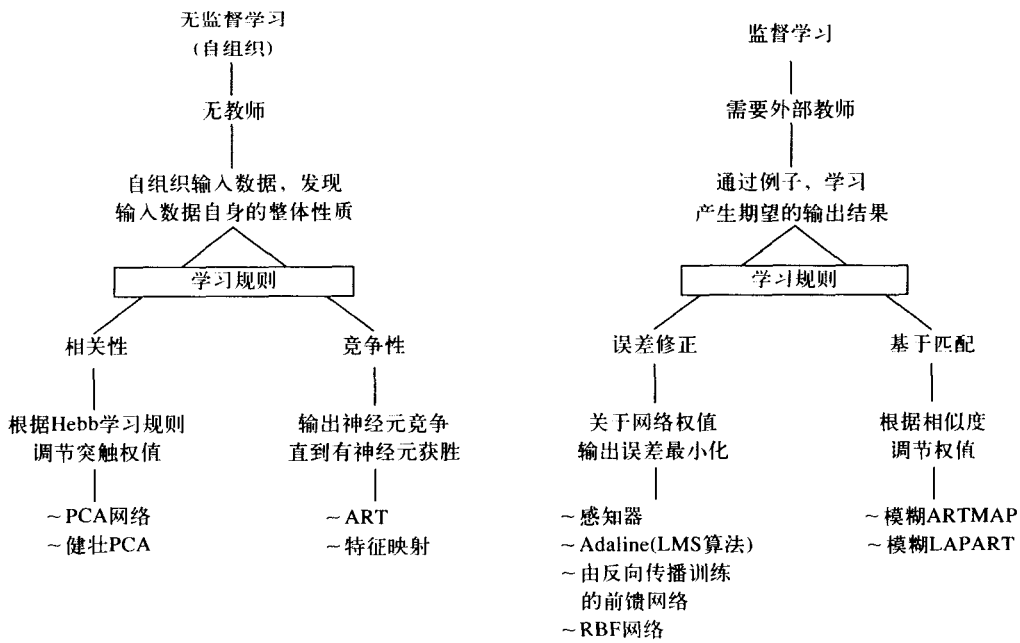



图1-10 按照如何学习（即监督学习还是无监督学习）选择神经网络的类别

## 1.5 本书指南

本书分成两大主要部分：

- 第一部分：神经计算的基本概念和部分神经网络体系结构及其学习规则
- 第二部分：神经计算的应用

*第一部分包含什么题材*

第一部分由第1～5章组成。第1章是概述，对读者介绍神经网络和神经计算的基本概念。第2章以人工神经元的基本模型介绍开始，人工神经元是神经网络的构件，接着，对激活函数的不同类型进行讨论。在介绍最小均方（least mean-square, LMS）算法之后，给出自适应线
19 性单元（Adaline）和多重自适应线性单元（Madaline）。详细地阐述了简单感知器，接着是对前馈多层感知器的简短讨论。在其后讨论一些基本学习规则，这些规则是前面几章介绍的某些情况的扩展。这些学习规则的大部分是用于训练更复杂神经网络体系结构的基础。第2章对选择的数据预处理方法做了概述。以作者的观点，这是非常重要的专题，但神经计算的研究中没有足够强调这点。如果读者还不熟悉人工神经网络的话，应深入学习第2章所有内容。这章是深入理解第3～5章中精选的神经网络体系结构及其相关学习规则的基础。这些章综合介绍了精选的具有历史意义的和在现实世界中获得重大应用的神经网络及其学习规则。此外，这几章包含的大量信息同本书第二部分讨论的题材有关。另外，第一部分为高级读者介绍各种神经体系结构和学习规则，这样使读者能为解决实际问题建立自己健壮的神经网络处理方法。

第3章给出多种映射神经网络。从联想记忆开始，接着是用于训练前馈多层感知器的反向传播，对于反向传播给出更先进的训练方法。还介绍对传网络，这章还包括对径向基函数神经网络的描述。第4章讨论部分自组织神经网络。包括Kohonen的自组织映射（self-organizing map, SOM）、学习向量量化（learning vector quantization, LVQ）和自适应共振理论（adaptive resonance theory, ART）神经网络；对ART1网络进行了详细介绍。第5章给出时序前馈网络（也是递归神经网络）和递归神经网络。介绍时序前馈网络和那些不是多层前馈网络的网络之间的区别。在该章中包括有霍普菲尔德网络、模拟退火、玻尔兹曼机、简单递归网络（simple recurrent network, SRN）、时延网络以及分布式时滞前馈神经网络。本书用于神经计算的研究，这决定了第3～5章包含了很多内容。

*第二部分包含什么题材*

第二部分由第6～10章组成。这几章包含神经计算在解决工程和科学的种种难题中的许多不同应用。第6章介绍用于求解约束最优化问题的部分神经计算方法，包括用于线性规划和二次规划的神经网络，和用于非线性连续约束最优化问题的神经网络。第7章给出用于求解矩阵
20 代数问题的一系列结构化神经网络。对于每个方法同时给出各种各样的重要的矩阵分解（或因式分解）和神经计算解法。神经计算方法也用于计算矩阵伪逆，求解代数李雅普诺夫（Lyapunov）方程和代数里卡蒂（Riccati）方程。第8章讲解如何用神经网络求解线性代数方程。这些方法包括最小二乘神经计算方法、共扼梯度学习规则、广义鲁棒性神经计算方法、用于解决具有不确定数值秩的不适定问题的正则化方法、用于迭代离散时间方法的矩阵分裂和总体最小二乘问题。并且，给出了用于解决线性代数方程的$L_\infty$范数和$L_1$范数的神经网络方法。第9章给出用于统计分析的不同神经计算方法的深入讨论。包括如下内容：用于主成分分析（principal-component analysis, PCA）、主成分回归（PCR）和经典最小二乘（CLS）的神经网络，用于非线性PCA和鲁棒PCA的神经网络，用于部分最小二乘回归（partial least-

squares regression, PLSR）的神经网络方法，以及用于鲁棒PLSR的神经网络方法。第10章包括的神经网络应用涉及信号处理、线性和非线性系统辨识、非线性控制和估计，包含许多不同例子的细节，也包括使用神经网络用于盲源分离的独立成分分析（independent-component analysis, ICA）。另外，给出快速ICA算法以及将快速ICA算法应用于数字图像分离的例子。最后包括附录，给出一些用于研究和应用神经计算技术的数学基础。

## 参考文献


1. J. A. Anderson and E. Rosenfeld, eds., *Neurocomputing: Foundations of Research*, Cambridge, MA: M.I.T. Press, 1988.
2. G. Nagy, "Neural Networks — Then and Now," *IEEE Transactions on Neural Networks*, vol. 2, 1991, pp. 316–18.
3. B. Widrow and M. A. Lehr, "30 Years of Neural Networks: Perceptron, Madaline and Backpropagation," *Proceedings of the IEEE*, vol. 78, 1990, pp. 1415–42.
4. S. Saarinen, R. B. Bramley, and G. Cybenko, "Neural Networks, Backpropagation, and Automatic Differentiation," in *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, eds.: A. Griewank and G. F. Corliss, Philadelphia, PA: Society for Industrial and Applied Mathematics, 1992, pp. 31–42.
5. W. S. McCulloch and W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, vol. 5, 1943, pp. 115–33. Reprinted in 1988, Anderson and Rosenfeld [1], pp. 18–27.
6. D. O. Hebb, *The Organization of Behavior*, New York: Wiley, 1949, introduction and chapter 4, "The First Stage of Perception: Growth of the Assembly," pp. xi–xix, 60–78. Reprinted in 1988, Anderson and Rosenfeld [1], pp. 484–507.
7. J. von Neumann, *The Computer and the Brain*, New Haven, CT: Yale University Press, 1958.
8. F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, vol. 65, 1958, pp. 386–408. Reprinted in 1988, Anderson and Rosenfeld [1], pp. 92–114.
9. B. Widrow and M. E. Hoff, Jr., "Adaptive Switching Circuits," *IRE WESCON Convention Record*, part 4, New York: IRE, 1960, pp. 96–104. Reprinted in 1988, Anderson and Rosenfeld [1], pp. 126–34.
10. M. L. Minsky and S. A. Papert, *Perceptrons*, Cambridge, MA: M.I.T. Press, 1969.
11. J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Redwood City, CA: Addison-Wesley, 1991.
12. T. Kohonen, "Correlation Matrix Memories," *IEEE Transactions on Computers*, vol. C-21, 1972, pp. 353–59. Reprinted in 1988, Anderson and Rosenfeld [1], pp. 174–80.
13. J. A. Anderson, "A Simple Neural Network Generating an Interactive Memory," *Mathematical Bioscience*, vol. 14, 1972, pp. 197–220. Reprinted in 1988, Anderson and Rosenfeld [1], pp. 181–92.
14. C. von der Malsburg, "Self-Organization of Orientation Sensitive Cells in the Striata Cortex," *Kybernetik*, vol. 14, 1973, pp. 85–100. Reprinted in 1988, Anderson and Rosenfeld [1], pp. 212–27.
15. P. J. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," Ph.D. thesis, Cambridge, MA: Harvard University, 1974.
16. W. A. Little and G. L. Shaw, "A Statistical Theory of Short and Long Term Memory," *Behavioral Biology*, vol. 14, 1975, pp. 115–33. Reprinted in 1988, Anderson and Rosenfeld [1], pp. 231–41.
17. S. C. Lee and E. T. Lee, "Fuzzy Neural Networks," *Mathematical Biosciences*,

vol. 23, 1975, pp. 155–77.
18. S. Grossberg, "Adaptive Pattern Classification and Universal Recoding: I. Parallel Development and Coding of Neural Feature Detectors," *Biological Cybernetics*, vol. 23, 1976, pp. 121–34. Reprinted in 1988, Anderson and Rosenfeld [1], pp. 245–58.
19. S.-I. Amari, "Neural Theory of Association and Concept-Formation," *Biological Cybernetics*, vol. 26, 1977, pp. 175–85. Reprinted in 1988, Anderson and Rosenfeld [1], pp. 271–81.
20. J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proceedings of the National Academy of Sciences*, vol. 79, 1982, pp. 2554–58. Reprinted in 1988, Anderson and Rosenfeld [1], pp. 460–4.
21. T. Kohonen, "Self-Organizing Formation of Topologically Correct Feature Maps," *Biological Cybernetics*, vol. 43, 1982, pp. 59–69. Reprinted in 1988, Anderson and Rosenfeld [1], pp. 511–21.
22. E. Oja, "A Simplified Neuron Model as a Principal Component Analyzer," *Journal of Mathematical Biology*, vol. 15, 1982, pp. 267–73.
23. K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, 1983, pp. 826–34. Reprinted in 1988, Anderson and Rosenfeld [1], pp. 526–34.
24. S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, 1983, pp. 671–80. Reprinted in 1988, Anderson and Rosenfeld [1], pp. 554–67.
25. R. R. Kampfner and M. Conrad, "Computational Modeling of the Evolutionary Learning Processes in the Brain," *Bulletin of Mathematical Biology*, vol. 45, 1983, pp. 931–68.
26. D. B. Fogel, ed., *Evolutionary Computation: The Fossil Record*, Piscataway, NJ: IEEE Press, 1998.
27. D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A Learning Algorithm for Boltzmann Machines," *Cognitive Science*, vol. 9, 1985, pp. 147–69. Reprinted in
22 1988, Anderson and Rosenfeld [1], pp. 638–49.
28. D. Parker, "Learning Logic," Technical Report TR-87, Cambridge, MA: Center for Computational Research in Economics and Management, M.I.T., 1985.
29. Y. LeCun, "Une procedure d'apprentissage pour reseau a seuil asymmetrique," in *In Cognitiva 85: A la Frontiere de l'Intelligence Artificielle des Sciences de la Connaissance des Neurosciences*, CESTA, vol. 85, Paris, 1985, pp. 599–604.
30. J. Herault, C. Jutten, and B. Ans, "Détection de grandeur primitives dans un message composite par une architecture de calcul neuromimétique en apprentissage nonsupervisé," in *Proceedings GRETSI Conference*, Nice, France, May 1985, pp. 1017–22 (in French).
31. C. Jutten and J. Herault, "Blind Separation of Sources, Part I: An Adaptive Algorithm Based on Neuromimetic Architecture," *Signal Processing*, vol. 24. 1991, pp. 1–10.
32. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, vol. 1, *Foundations*, eds. D. E. Rumelhart and J. L. McClelland, Cambridge, MA: M.I.T. Press, 1986, pp. 318–62. Reprinted in 1988, Anderson and Rosenfeld [1], pp. 675–95.
33. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-Propagating Errors," *Nature*, vol. 323, 1986, pp. 533–36. Reprinted in 1988, Anderson and Rosenfeld [1], pp. 696–9.
34. G. A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," *Computer Vision, Graphics,*

*and Image Processing*, vol. 37, 1987, pp. 54–115.
35. M. A. Sivilotti, M. A. Mahowald, and C. A. Mead, "Real-Time Visual Computations Using Analog CMOS Processing Arrays," *Advanced Research in VLSI: Proceedings of the 1987 Stanford Conference*, ed. P. Losleben, Cambridge, MA: M.I.T. Press, 1987, pp. 295–312. Reprinted in 1988, Anderson and Rosenfeld [1], pp. 703–11.
36. D. S. Broomhead and D. Lowe, "Multivariable Functional Interpolation and Adaptive Networks," *Complex Systems*, vol. 2, 1988, pp. 321–55.
37. D. W. Fausett, "Strictly Local Backpropagation," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 3. San Diego, CA, 1990, pp. 125–30.
38. S. Coren, L. M. Ward, and J. T. Enns, *Sensation and Perception*, 4th ed., Orlando, FL: Harcourt Brace, 1994.
39. G. M. Shepherd, *Neurobiology*, 3rd ed., New York: Oxford University Press, 1994.
40. A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, New York: Wiley, 1993.
41. N. K. Bose and P. Liang, *Neural Network Fundamentals with Graphs, Algorithms, and Applications*, New York: McGraw-Hill, 1996.
42. A.C. Guyton, *Textbook of Medical Physiology*, 8th ed., Philadelphia, PA: W.B. Saunders, 1991.
43. *DARPA Neural Networks Study*, AFCEA International Press, November 1988, Chapter 8, "Classification and Clustering Models."

# 第2章 神经计算的基本概念

## 2.1 概述

本章所讲述的是可执行各种功能的神经结构的基础。正如第1章所述，人工神经网络由许多相互连接的处理单元（人工神经元或节点）组成。因此，本章先从人工神经元模型的表示开始讲起。这些模型可以用在无数应用中的许多不同神经网络类型的构件。本章主要关注从科学和工程的角度对基本概念的表述，利用人工神经网络解决问题。我们不讨论关于生物神经网络的议题，如认知、神经建模、神经生理学的逻辑问题和人脑的细节。

在2.8节将给出多个用于单个神经元的基本学习规则。这些概念可以扩展至具有多个神经元的网络。因此，这里给出的很多学习规则在后面几章中用来建造更复杂的神经结构。这章的最后一部分主要讨论数据预处理这个非常重要的课题。一些预处理方法将被提及。特定神经网络的性能取决于训练阶段（特别是所使用的训练数据）。多数情况下，有必要对训练数据预处理，从可用的数据中抽取重要特征，取代“原始”数据训练网络。因此，对训练数据的
24 预处理可以提高神经网络的性能。

## 2.2 人工神经元的基本模型

在第1章中给出了人工神经网络（ANN）的基本概念，在那里阐明，神经网络通常由许多单个神经元构成。一个人工神经元也可称为处理基元、节点或阈值逻辑单元；但是，我们通常称之为*神经元*。神经元是一个信息处理单元，它与其生物对应物大致相似（参看1.3节）。图2-1显示一个人工神经元模型。模型由四个基本组成部分：（1）一组同突触权值相联系的突触。如图2-1所示，对突触输入的连续值是一个向量信号$\boldsymbol{x} \in \Re^{n\times 1}$，$\boldsymbol{x}$分量为$x_j$，$j = 1, 2, \cdots, n$，即$\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T$。因此，每个向量分量$x_j$输入给第$j$个突触，并且通过突触权值$w_{qj}$与神经元$q$相连接；即，$x_j$用突触权值$w_{qj}$相乘。对于突触权值下标的约定是第一个下标与一个特定神经元相联系，第二个下标同和突触权值相乘的输入向量元素关联。这种约定带有随意性（也可以采用相反表示法）；然而，为了网络结构中的一致性必须始终遵守这种约定。（2）求和装置将迭加所有传播到加法器的信号，也就是说，每个输入与它相联系的突触权值相乘，然后再相加求和。包括加法器的输出$u_q$在内的所有运算构成一个线性的组合器，由于$u_q$是对神经突触输入的线性组合。（3）如图2-1所示，当激活函数（或挤压函数）$f(\cdot)$是非线性时，函数$f(\cdot)$将限制神经元输出$y_q$的幅度。激活函数可以是连续值的、二值的、或双极值的；或在某些情况下可以是线性函数。当激活函数是非线性时，$f(\cdot)$的有限极限通常规整到[0, 1]（二值）的范围或者[－1, 1]（双极值的）的范围。在一个高度互连，大规模并行的人工神经体系结构中，
25 非线性对网络的分类、近似表示和抗噪音干扰等能力有增强作用。（4）阈值$\theta_q$通常在外部应用，并且降低对激活函数的累积输入。所以，$\theta_q$在用于激活之前，从线性组合器的输出$u_q$中减掉。激活函数的输入可以通过添加一项$u_q$即*偏置*而得到增加。在这种情况下，$\theta_q$将添加到$u_q$；因而，偏置是阈值的负数。特别地，有效内在*激活电位*（activation potential）或*活动水平*（activity level）

$$v_q = u_q - \theta_q \tag{2-1}$$

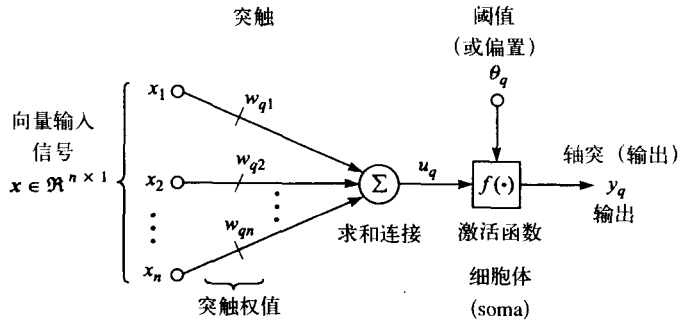



图2-1 人工神经元非线性模型

与神经元$q$的线性组合器输出$u_q$之间的关系依赖于阈值$\theta_q$是正数还是负数。可以把阈值想像成对线性组合器的输出$u_q$应用一次仿射变化[1]。

在数学上，我们可以通过下面写出的几个等式来描述在图2-1中的人工神经元的操作。线性组合器的输出由公式

$$u_q = \sum_{j=1}^{n} w_{qj}x_j = \boldsymbol{w}_q^T\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{w}_q \tag{2-2}$$

给出，其中$\boldsymbol{x}$在上面描述，$\boldsymbol{w}_q = [w_{q1}, w_{q2}, \cdots, w_{qn}]^T \in \Re^{n\times 1}$，并且激活函数的输出是

$$y_q = f(v_q) = f(u_q - \theta_q) \tag{2-3}$$

所以，借助式（2-2）和式（2-3），神经元的输出为

$$y_q = f\left(\sum_{j=1}^{n} w_{qj}x_j - \theta_q\right) \tag{2-4}$$

图2-2显示人工神经元的另外一种模型。在该模型中，阈值（或偏置）合并到神经元$q$的突触权值向量$\boldsymbol{w}_q$中，并且输入向量增加$x_0$。所以$\boldsymbol{w}_q \in \Re^{n+1\times 1}$，$\boldsymbol{x} \in \Re^{n+1\times 1}$。当前有效的内在激活电位写作

$$v_q = \sum_{j=0}^{n} w_{qj}x_j \tag{2-5}$$

26

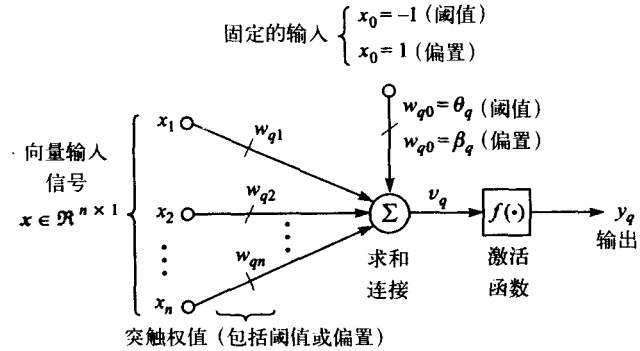



图2-2 人工神经元的另一个非线性模型

并且神经元$q$的输出写作

$$y_q = f(v_q) \tag{2-6}$$

所以，在式（2-5）中增加一个新的突触，并且由图2-2，根据外部作用的是阈值还是偏置，对该突触的输入分别是$x_0=-1$或$x_0=1$，并且对于外部应用阈值，相关的突触权值为$w_{q0}=\theta_q$，或者对于外部应用的偏置$w_{q0}=\beta_q$。

## 2.3 基本激活函数

如图2-1或图2-2所示的*激活函数*（也称为*传递函数*）可以是线性或非线性函数。有许多不同种类的激活函数。由神经元（或神经网络）所需解决的特定问题决定选择某类激活函数。这里我们给出四类最常见的激活函数。在接下来的几章中将根据所解决问题的类型给出更复杂的函数。激活函数表示的基本参考模型如图2-2所示。除非特别指明，一般假设阈值（或偏置）为0。

第一类是*线性*（或*恒等*）*函数*，它为连续值的。数学上，神经元$q$的线性激活函数的输出可以写为

$$y_q=f_{\text{lin}}(v_q)=v_q \tag{2-7}$$

其中，如图2-2所示，$v_q$（有效内部激活电位）是线性组合器的输出，并且是激活函数$f(\cdot)$的输入。恒等函数的输出，即神经元$y_q$的输出，就是$v_q$（或线性组合器的输出），如图2-3所示。这或许是一个平凡的激活函数，然而，以后我们将看到在一些线性网络中是非常有用的。

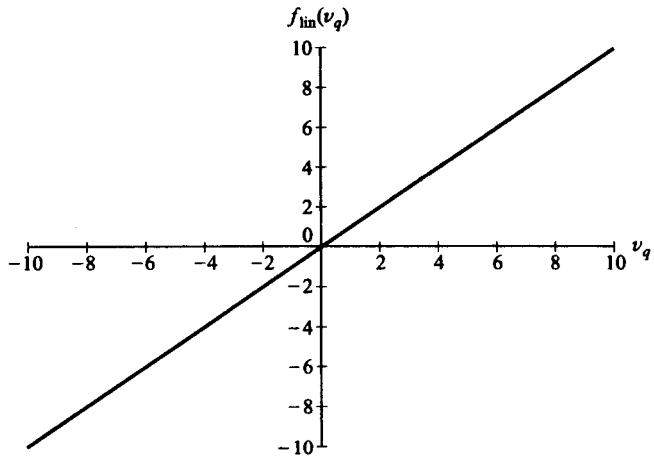



图2-3　线性（恒等）激活函数

第二类激活函数是*硬限幅器函数*。这是一个*二值*（或*双极值*）函数，对于二值类型，硬限幅器函数的输入为0或1，而对于双极值类型，硬限幅器函数的输入为$-1$或1。有时二值硬限制称为*阈值函数*（我们仅称其为硬限幅器），而双极值硬限幅器称为*对称硬限幅器*，即由函数值与输入值$v_q$为坐标所画的图形是对称的。二值硬限幅器的输出（或神经元$q$输出）可以写为

$$y_q=f_{\text{hl}}(v_q)=\begin{cases}0 & \text{如果 } v_q<0\\ 1 & \text{如果 } v_q\geqslant 0\end{cases} \tag{2-8}$$

对于对称硬限幅器（下标“shl”），神经元$q$的输出可写为

$$y_q = f_{\text{shl}}(v_q) = \begin{cases} -1 & \text{如果 } v_q < 0 \\ 0 & \text{如果 } v_q = 0 \\ 1 & \text{如果 } v_q > 0 \end{cases} \tag{2-9}$$

有时将此函数称为符号（*signum*，或*sign*）函数，即$f(\cdot) = \text{sgn}(\cdot)$。硬限幅器和对称硬限幅器的特征分别如图2-4和图2-5所示。具有硬限幅器激活函数的人工神经元称为McCulloch-Pitts模型（或阈值单元）[2]，同第1章讨论的一样。在最初的研究中，神经元权值连同阈值水平，都是预置的。因此，没有与神经元相关的训练；而由分析导出的预置权值，神经元能执行简单的逻辑功能，在第1章中已经讲述了三类这种简单的逻辑函数。许多人认为人工神经网络领域的起源来自于他们的先驱性工作。 28

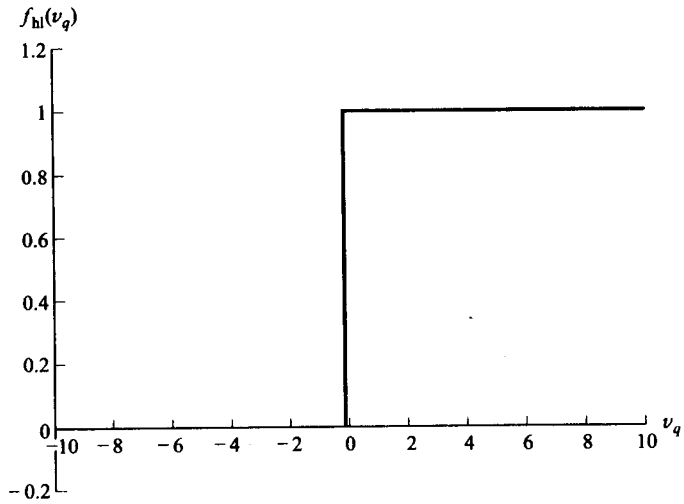



图2-4 硬限幅器激活函数

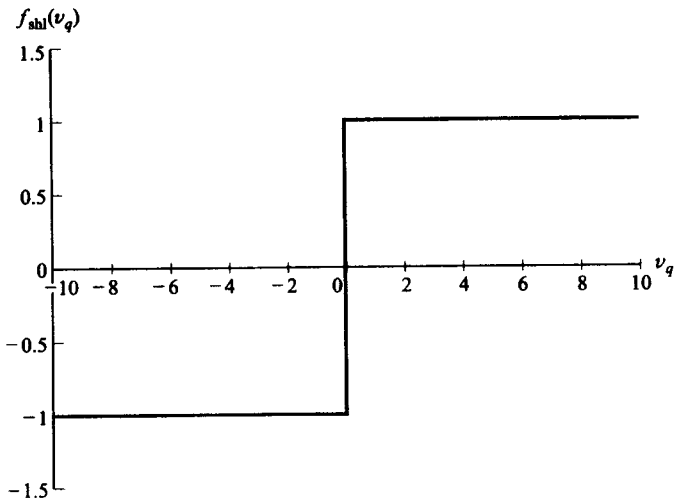



图2-5 对称硬限幅器激活函数

第三类基本激活函数是饱和线性函数或分段线性函数。对于输出的饱和限制，这类函数有二值或双极值值域。双极值饱和线性函数称为对称饱和线性函数。饱和线性函数的神经元$q$的输出（即二值输出）表示为

$$y_q = f_{sl}(v_q) = \begin{cases} 0 & \text{如果} \quad v_q < -\dfrac{1}{2} \\ v_q + \dfrac{1}{2} & \text{如果} \quad -\dfrac{1}{2} \leqslant v_q \leqslant \dfrac{1}{2} \\ 1 & \text{如果} \quad v_q > \dfrac{1}{2} \end{cases} \tag{2-10}$$

而对称饱和线性函数的输出为

$$y_q = f_{ssl}(v_q) = \begin{cases} -1 & \text{如果} \quad v_q < -1 \\ v_q & \text{如果} \quad -1 \leqslant v_q \leqslant 1 \\ 1 & \text{如果} \quad v_q > 1 \end{cases} \tag{2-11}$$

饱和线性函数和对称饱和线性函数的特征分别如图2-6和图2-7所示。

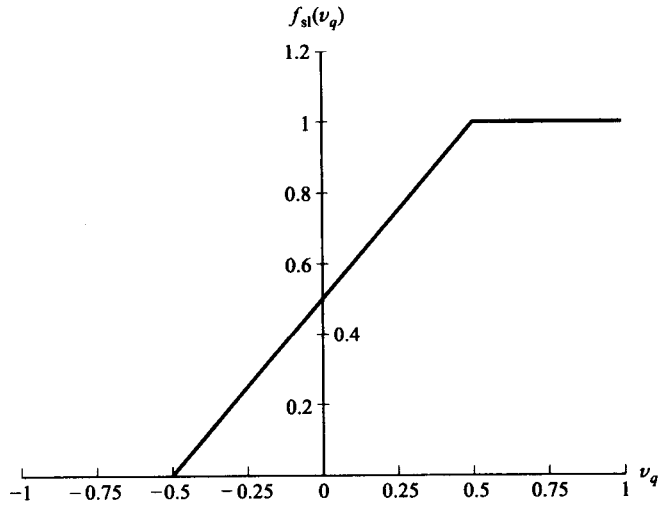



图2-6　饱和线性激活函数

第四类基本激活函数一般称为sigmoid（S形）函数，这里我们给出两类。非线性S形函数用来构造人工神经网络最常用的激活类型。在数学上它是良态的、严格递增函数。第一类S形函数是二值S形函数。此函数的饱和输出值有一个二值值域，神经元$q$的输出在数学上可写为

29

$$y_q = f_{bs}(v_q) = \frac{1}{1 + e^{-\alpha v_q}} \tag{2-12}$$

其中$\alpha$是二值S形函数的倾斜参数。通过改变这个参数，可以得到函数的不同形状，如图2-8所示。与硬限幅器在原点无导数不同，二值S形函数是连续可微函数。后面我们将注意到，激活函数的可微性在神经计算中起着重要作用。关于线性组合器输出的二值S形函数的导数可表示为

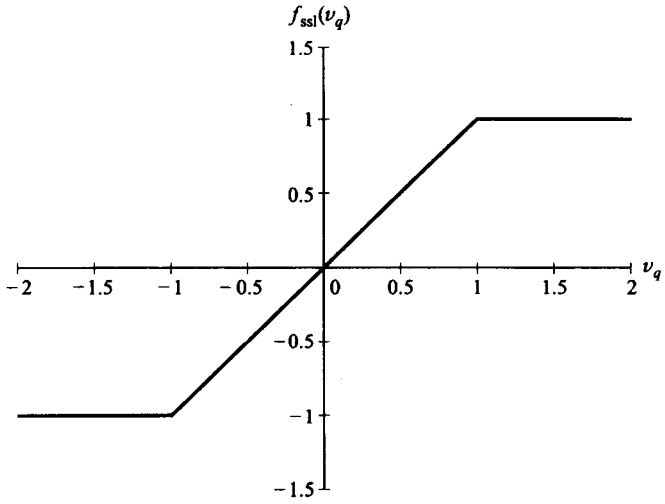



图2-7　对称饱和线性激活函数

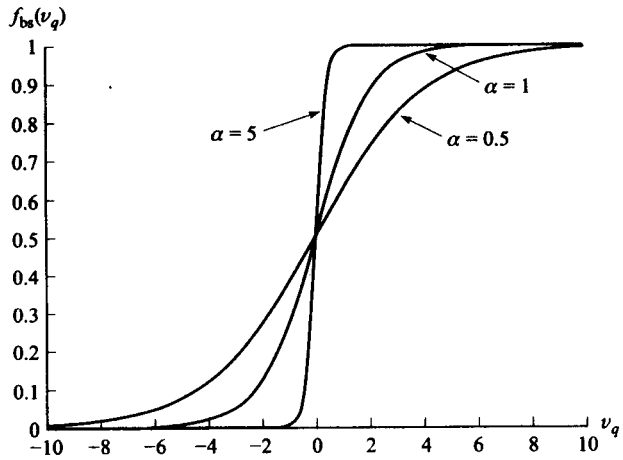



图2-8　对于倾斜参数三个不同值的二值S形激活函数

$$g_{bs}(v_q) = \frac{df_{bs}(v_q)}{dv_q} = \frac{\alpha e^{-\alpha v_q}}{(1+e^{-\alpha v_q})^2} = \alpha f_{bs}(v_q)[1 - f_{bs}(v_q)] \tag{2-13}$$

在坐标原点，用$\alpha/4$表示二值S形函数的倾斜度。通过在式（2-13）中设置$v_q = 0$，这一点是显然的。因此，随着$\alpha$的增加，二值S形函数逼近硬限幅器。图2-9画出了倾斜参数的两个不同值$\alpha = 1$和$\alpha = 0.5$的二值S形函数的导数。

S形函数的双极值形式可以是*双曲正切S形函数*。因此，这个函数的饱和限制为双极值值域，神经元$q$的输出可表示为

$$y_q = f_{hts}(v_q) = \tanh(\alpha v_q) = \frac{e^{\alpha v_q} - e^{-\alpha v_q}}{e^{\alpha v_q} + e^{-\alpha v_q}} = \frac{1 - e^{-2\alpha v_q}}{1 + e^{-2\alpha v_q}} \tag{2-14}$$

其中$\alpha$为倾斜参数。图2-10画出了三种不同倾斜参数的双曲正切S形函数。双曲正切S形函数关于$v_q$的导数由

$$g_{\text{hts}}(v_q)=\frac{\mathrm{d}f_{\text{hts}}(v_q)}{\mathrm{d}v_q}=\alpha\left[1+f_{\text{hts}}(v_q)\right]\left[1-f_{\text{hts}}(v_q)\right] \tag{2-15}$$

30
~
31

给出。值得指出，式（2-13）和式（2-15）中分别由式（2-12）和式（2-14）中给出的两个S形激活函数的导数，可以由激活函数本身表示。这一点在后面将非常重要，例如，在我们建立训练简单感知器（参考2.6节）和前馈多层感知器的学习规则时（参考3.3节）。

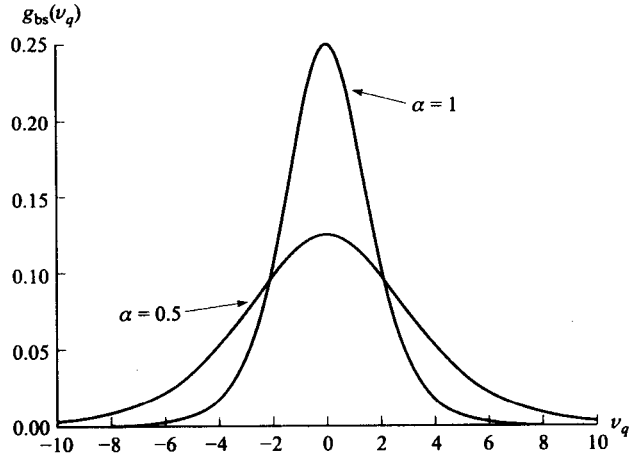



图2-9　对于倾斜参数两个不同值的二值S形激活函数的导数

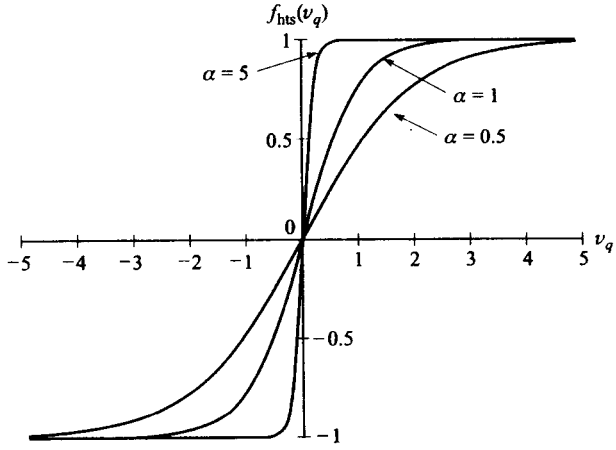



图2-10　对于倾斜参数三个不同值的双曲正切S形激活函数

迄今为止，我们所提到的激活函数可取正负值，这样有助于分析，正如我们在下一章将看到的一样。而且，使用这样的激活函数有神经生理学的实验证据支持[3]。然而，很少有像
32
双曲正切S形激活函数一样关于原点精确反对称的。

如图2-2所示，阈值$\theta_q$和偏置$\beta_q$的效果可以通过观察式（2-12）中的二值S形函数来说明。图2-11表示了二值S形函数的三种图形，分别为阈值$\theta_q=2$，偏置$\beta_q=2$和通常情况（$\theta_q=\beta_q=0$）。对于这三种情形，倾斜参数设为单位值（即$\alpha=1$），从图2-11可看出，使用阈值相当于延迟时间域信号，添加偏置相当于信号的提前。

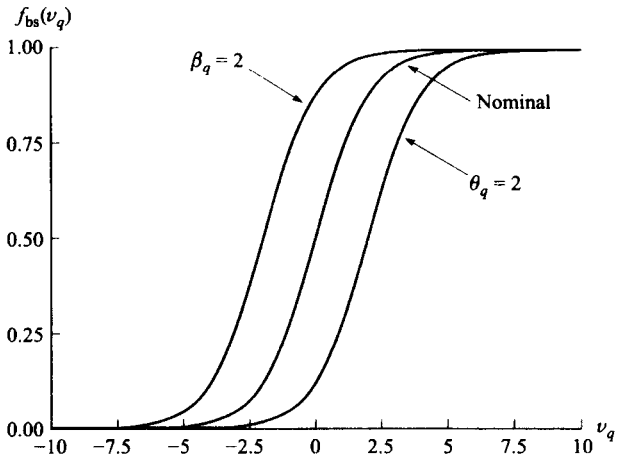



图2-11 具有阈值$\theta_q = 2$和偏置$\beta_q = 2$的二值S形激活函数（$\alpha = 1$）

## 2.4 人工神经元的霍普菲尔德模型

关于John Hopfield的论文[5]，正如[4]中陈述的那样，“照至今公开的资料来看，神经网络的新纪元开始于John Hopfield发表的这篇论文。”在这篇1982年的论文中，霍普菲尔德给出了由简单处理单元组成的神经体系结构，这种处理单元建立在McCulloch和Pitts[2]的形式神经元基础上。尽管以前进行了神经网络领域的大量研究，但是霍普菲尔德的论文明确描述了在文献上似乎不相关的若干概念并把它们联系在一起。这篇文章的影响主要在于作者用一种高度一致的方式表达这些概念，并阐述神经网络如何工作的理论思想和实际问题的关系，以及它的可能的应用。

图2-2给出了基本模型，在图2-4中表明激活函数$f(\cdot)$为硬限幅器。神经元具有两种状态，由它的激活电位决定：打开状态（或激活状态）由神经元输出$y_q = 1$给出；关闭状态（或非激活状态）由神经元输出$y_q = 0$表示。基于这些神经元的霍普菲尔德神经网络是异步并行处理的、 33
完全互连的、内容可寻址的记忆（或联想记忆）；它具有检索存储模式的基本功能，以响应有噪声或不完全形式的模式。通常对于离散时间模型来说，把激活函数作为对称硬限幅器（如图2-5所示）；对于连续时间模型而言，把双曲正切S形函数（如图2-10所示）作为激活函数以利用双极值输出。和前馈感知器不同，霍普菲尔德网络具有反馈功能，正因如此，霍普菲尔德网络视为递归神经网络。我们将在5.3节中讨论霍普菲尔德网络的细节。这里我们想通过给出离散时间和连续时间的霍普菲尔德神经元模型来打下基础。

人工神经元的离散时间霍普菲尔德模型如图2-12所示。从图2-12中，我们可以把神经元在单位延迟$z^{-1}$之前的输出写成

$$y_q(k+1) = f_{shl}\,[v_q(k+1)] \tag{2-16}$$

其中

$$v_q(k+1) = \sum_{j=1}^{n} w_{qj} x_j(k) - \theta_q \tag{2-17}$$

$\theta_q$是外部应用的阈值，并且$v_q(k) = v_q(kT_S)$，其中$k = 0, 1, 2, \cdots$是离散时间下标，并且$T_S$是采样周期。为了不失一般性，假定采样周期规整化为单位值（即$T_S = 1$）。利用式（2-16）和式（2-17），神经元的输出$y_q(k+1)$可表示为

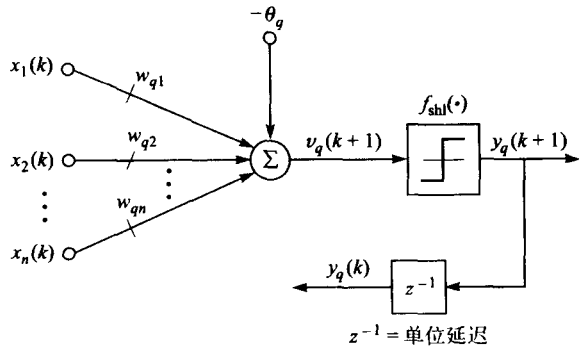



图2-12　霍普菲尔德人工神经元的离散时间模型

$$y_q(k+1) = f_{\text{shl}}\left[\sum_{j=1}^{n} w_{qj} x_j(k) - \theta_q\right] \tag{2-18}$$

单位延迟$z^{-1}$延迟激活函数输出的一个采样周期，以给出$y_q(k)$，其中$z$是复数变量[6，7]。由于在离散时间情况下，形成神经元的响应公式，所以，式（2-18）是*差分方程*。如将在5.3节中
34 看到的一样，在霍普菲尔德网络中，这个量$y_q(k)$反馈至其他神经元（即除延迟输出被提取神经元外的其他所有神经元）作为输入，组成离散时间霍普菲尔德神经网络。因此，每一神经元的输入均是双极值，对于离散时间的霍普菲尔德神经网络突触权值矩阵$\boldsymbol{W} = [w_{qj}](q, j = 1, 2, \cdots, n)$是对角线为零的实对称矩阵。

人工神经元的连续时间霍普菲尔德模型如图2-13所示。在图2-13中，$T_{cq} = R_q C_q$是第$q$神经元的积分时间常量，$\theta_q$是外部应用的阈值。利用运算放大器[8]、电容器$C_q$和电阻器$R_q$实现（泄漏）积分器，并且$\gamma_q > 0$称为积分器的*泄漏*（或*遗忘*）*因子*，如图2-13中的反馈回路所示。泄漏因子对于0输入，迫使内部信号$v_q$变为0。从图2-13中我们可以写出激活电位$v_q(t)$的*差分方程*：

$$T_{cq}\frac{\mathrm{d}v_q}{\mathrm{d}t} = -\gamma_q v_q + \left(\sum_{j=1}^{n} w_{qj} x_j - \theta_q\right) \tag{2-19}$$

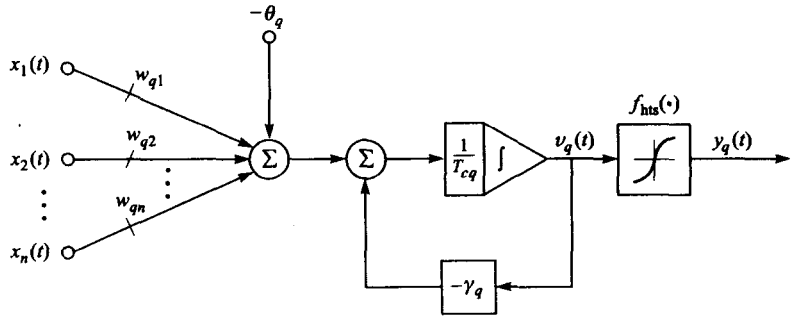



图2-13　霍普菲尔德人工神经元的连续时间模型

图2-13所示的神经元输出可表示为

$$y_q = f_{\text{hts}}(v_q) \tag{2-20}$$

其中激活函数$f_{\text{hts}}(\cdot)$是双曲正切S形函数，如图2-10所示并且在式（2-14）中定义。若令$\mathrm{d}v_q/\mathrm{d}t = 0$和$\gamma_q = 1$，双曲正切S形函数倾斜参数$\alpha \gg 1$，则从式（2-19）中可导出离散时间模型的

差分方程（2-17）[9]。由连续时间系统[10]的状态空间模型建立离散时间状态空间模型实际上是一个经典练习题。

## 2.5 自适应线性单元和多重自适应线性单元

在1960[11]中，Bernard Widrow和M. E.(Ted) Hoff, Jr.（Widrow的学生）在加州斯坦福的斯坦福大学最先提出了最小均方（Least-Mean-Square, LMS）算法。这个学习规则有时称作 35
Widrow-Hoff学习规则或δ规则[12, 13]。Adaline（自适应线性单元）是根据LMS算法来更新突触权值的单个神经元[14]。在斯坦福大学，Widrow和他的学生们开发了第一个可训练的分层神经网络，它具有多个自适应单元，称作Madaline（多重Adaline）[14-16]。用于训练单层网络的LMS算法是用于前馈多层感知器（见3.3节）的反向传播学习规则的前身。Memistor公司（由Bernard Widrow创建）的第一批商业神经计算机的基础是Adaline和Madaline。LMS学习规则（δ规则）是用于计算神经元突触权值调整的自适应算法。该算法以最速下降方法为基础[17, 18]，调整神经元权值，以最小化输入向量与权值向量的内积同神经元期望输出之间的均方误差。我们从解释一个简单自适应线性组合器开始，然后讨论LMS算法，Adaline和Madaline的细节。

### 2.5.1 简单自适应线性组合器和LMS算法

图2-14给出了简单自适应线性组合器的结构[11]。线性神经元的基本结构可从图2-2导出，$f(\cdot)$ 用作线性激活函数，$x_0 = 1$且$w_{q0} = \beta_q$（偏置），$q = 1$（因为我们只处理单个神经元；因而扔掉这个下标）。在神经元的监督训练过程中，自适应线性组合器用训练输入向量$\boldsymbol{x}(k) \in \Re^{n\times 1}$（即$\boldsymbol{x}(k) = [x_1(k)\ x_2(k)\cdots x_n(k)]^T$）和相应的期望响应$d(k) \in \Re$, $k = 1, 2, 3, \cdots$（离散时间下 36
标）表示，采样周期假定为$T_S = 1$。$\boldsymbol{x}$的分量既可以是连续模拟值［可能的采样如上所示，即具有$T_S = 1$的$\boldsymbol{x}(k)$］或二值（或双极值）。假定输入$\boldsymbol{x}(k)$为零均值（即$E\{\boldsymbol{x}\} = \boldsymbol{0}$），宽平稳向量随机过程，其中$E(\cdot)$是期望算子（见A.7.4节）。显然假定$E\{\boldsymbol{x}\} = \boldsymbol{0}$不是对LMS算法的一个限制。网络任意时间步$k$的输出是由输入向量$\boldsymbol{x}(k)$和权值向量$\boldsymbol{w}(k) = [w_1(k), w_2(k), \cdots, w_n(k)]^T \in \Re^{n\times 1}$的内积计算得到的，其中$x_0 = 1$, $w_0(k) = \beta$。我们可以像前面那样，把偏置与权值向量合并且把$x_0 = 1$合并到$\boldsymbol{x}(k)$。然而，不失一般性，假定偏置$\beta = 0$；因而网络输出为

$$v(k) = \boldsymbol{x}^T(k)\,\boldsymbol{w}(k) = \boldsymbol{w}^T(k)\,\boldsymbol{x}(k) \tag{2-21}$$

将输出$v(k)$与期望响应$d(k)$相比较，计算两个量的差异[或误差$e(k)$]。误差和输入向量$\boldsymbol{x}(k)$一起反馈给自适应器或学习算法，如图2-14所示。学习算法决定了如何改进网络突触权值，以使网络输出（或网络响应）$v(k) = \boldsymbol{w}^T(k)\,\boldsymbol{x}(k)$和期望响应$d(k)$的差尽可能小。期望响应作为网络的辅助输入，仅在训练阶段才使用。通常，学习算法被导出，使它们优化某些定义的误差标准（这一点在下面详细解释），导出LMS算法。

具有数字信号处理背景的读者容易把线性组合器当作线性横向滤波器[19]或有限冲击响应滤波器(finite impulse response, (FIR) filter)[20]。确实，线性组合器广泛应用在自适应信号处理中[19]。为了得到线性组合器的学习规则，最通常使用的性能标准是最小化网络输出与期望响应之间的误差平方。这称为均值平方误差（mean square error, MSE）标准。从MSE标准得到的最简单学习规则是最小均值平方（least mean-square, LMS）学习规则，或Widrow-Hoff学习规则，或delta规则[19]。在建立LMS学习规则前，先了解用于确定称为Wiener-Hopf解的最优权值向量$\boldsymbol{w}^*$的传统方法[19, 21]。

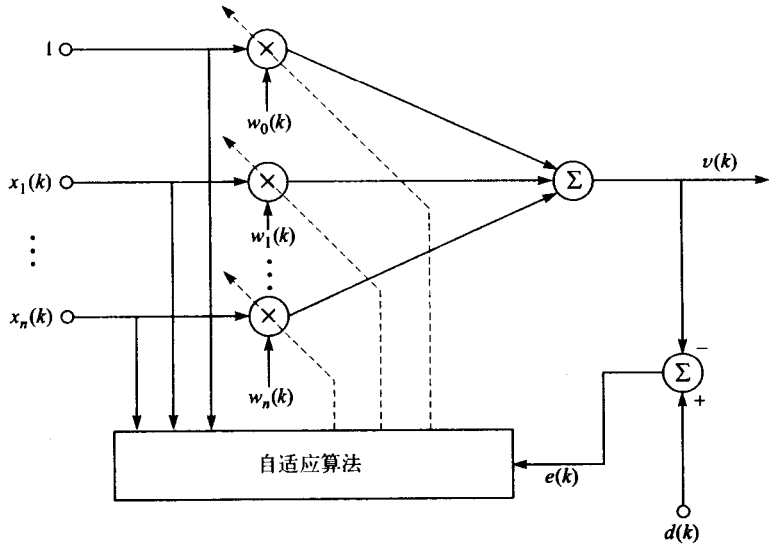



图2-14　简单自适应线性组合器

令$\boldsymbol{x}(k)$和$d(k)$分别表示训练输入和期望响应，在第$k$次迭代时呈现给网络；令$\boldsymbol{w}(k)$表示当前网络的权值向量。式（2-21）给出线性组合器的响应和误差，即期望响应与网络响应的差异，可写为

$$e(k) = d(k) - v(k) = d(k) - \boldsymbol{w}^T(k)\,\boldsymbol{x}(k) \tag{2-22}$$

假定$\boldsymbol{x}(k)$与$d(k)$均为统计上宽平稳总体，则MSE标准可以表示成由

37

$$\boldsymbol{J}(\boldsymbol{w}) = \frac{1}{2}\mathrm{E}\{e^2(k)\} = \frac{1}{2}\mathrm{E}\left\{\left[d(k) - \boldsymbol{w}^T(k)\boldsymbol{x}(k)\right]^2\right\} \tag{2-23}$$

给出的总体平均。考虑到学习规则的目的是适应性地修改网络权值以使式（2-23）最小化。因此，展开式（2-23），可以写作

$$\boldsymbol{J}(\boldsymbol{w}) = \frac{1}{2}\mathrm{E}\{d^2(k)\} - \mathrm{E}\{d(k)\boldsymbol{x}^T(k)\}\boldsymbol{w}(k) + \frac{1}{2}\boldsymbol{w}^T(k)\mathrm{E}\{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\}\boldsymbol{w}(k) \tag{2-24}$$

因为$\boldsymbol{x}(k)$和$d(k)$是宽平稳随机过程；式（2-24）可写为

$$\boldsymbol{J}(\boldsymbol{w}) = \frac{1}{2}\mathrm{E}\{d^2(k)\} - \boldsymbol{p}^T\boldsymbol{w}(k) + \frac{1}{2}\boldsymbol{w}^T(k)\boldsymbol{C}_x\boldsymbol{w}(k) \tag{2-25}$$

其中$\boldsymbol{p} = \mathrm{E}\{d(k)\,\boldsymbol{x}(k)\}$表示期望响应与输入模式之间的互相关向量，$\boldsymbol{C}_x = \mathrm{E}\{\boldsymbol{x}(k)\,\boldsymbol{x}^T(k)\}$表示输入模式的协方差矩阵［考虑到输入向量$\boldsymbol{x}(t)$是零均值］。等式（2-25）是突触权值的二次函数。因此，对于所有网络权值式（2-25）均为正。在权值向量空间中，对于$\boldsymbol{J}(\boldsymbol{w})$的MSE曲面有唯一的最小值[19]。因此，在式（2-25）中可以从数学上计算出对于权值向量$\boldsymbol{w}$的性能度量的梯度，用于最优条件，可设定这个结果为0，即

$$\nabla_{\boldsymbol{w}}\boldsymbol{J}(\boldsymbol{w}) = \frac{\partial \boldsymbol{J}(\boldsymbol{w})}{\partial \boldsymbol{w}} = -\boldsymbol{p} + \boldsymbol{C}_x\boldsymbol{w}(k) = \boldsymbol{0} \tag{2-26}$$

关于一个向量的数量微分，其详情请参看A.3.4.1节。另外在式（2-26）中MSE关于权值向量的导数是连续导数（即权值在振幅上是连续的）。从式（2-26）中可得到最优权值$\boldsymbol{w}^*$为

$$\boldsymbol{w}^* = \boldsymbol{C}_x^{-1}\boldsymbol{p} \tag{2-27}$$

等式（2-27）是关于著名的线性组合器最优权值的Wiener-Hopf解的向量矩阵形式[19]。在神经网络与信号处理中，式（2-27）的实际应用受到限制，有两点原因：（1）求协方差矩阵的逆的计算量非常大；（2）式（2-27）不适合于权值的联机修改，因为在多数情况下协方差矩阵和互相关向量事先并不知道。

为了克服这些问题，Widrow和Hoff[11]建立了LMS算法。通过分析图2-15表示的MSE曲面的特征很容易解释LMS算法的思想。图2-15描述了式（2-25）定义的一个典型的具有两个权值的线性组合器的误差曲面，即$w \subset \mathbb{R}^{2\times 1}$。为了在$J(w)$最小时得到突触权值的最优值，利用梯度下降法搜索误差曲面，找到最小值（即梯度为0时）。这和在式（2-26）中设定$J(w)$的梯度等于0的思想是一样的。显然，通过沿曲面的负梯度方向改变权值可到达图2-15中误差曲面的底部。由于不知道输入协方差矩阵和互相关向量时不能计算曲面的梯度，所以在迭代训练过程中必须估计它们。估计MSE梯度曲面最简单的、因此也是最粗略的方法，可以通过取瞬时误差曲面的梯度得到。即在式（2-24）中去掉期望算子$E(\cdot)$，$J(w)$的梯度近似为

38

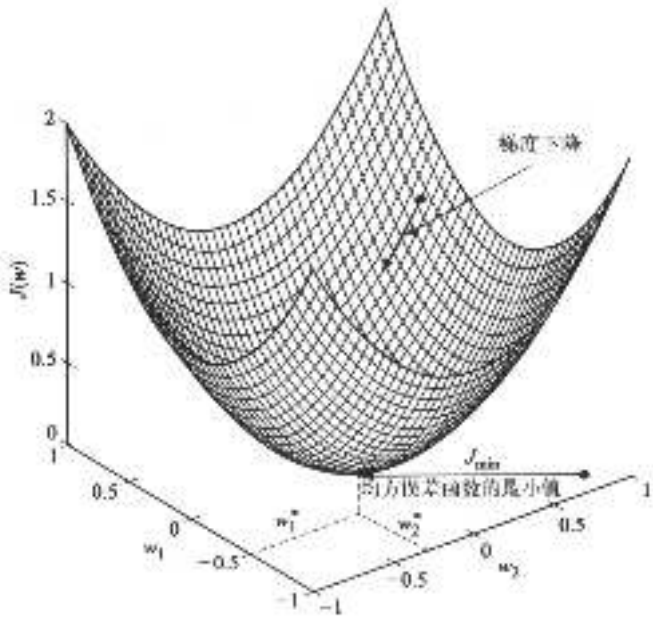



图2-15 自适应线性组合器的典型MSE曲面

$$
\begin{aligned}
\nabla_w J(w) &= \frac{1}{2}\frac{\partial e^2(k)}{\partial w}\bigg|_{w=w(k)} \\
&= \frac{1}{2}\frac{\partial}{\partial w(k)}[d^2(k)-2d(k)x^T(k)w(k)+w^T(k)x(k)x^T(k)w(k)] \\
&= -d(k)x(k)+x(k)x^T(k)w(k) = -d(k)x(k)+w^T(k)x(k)x(k) \\
&= -\underbrace{[d(k)-w^T(k)x(k)]}_{e(k)}x(k) = -e(k)x(k)
\end{aligned}
\tag{2-28}
$$

利用式（2-28）得到的MSE标准梯度的估计结果，运用最速下降梯度法更新权值的学习

规则可写为

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)+\mu[-\nabla_{\boldsymbol{w}}\boldsymbol{J}(\boldsymbol{w})]=\boldsymbol{w}(k)+\mu e(k)\,\boldsymbol{x}(k) \tag{2-29}$$

其中误差$e(k)$在式（2-22）中定义；式（2-22）和式（2-29）一起称为自适应线性组合器权值
39 更新的LMS算法。LMS算法有很多应用，如自适应平衡和噪音的消除。在式（2-29）中，实参$\mu>0$通常称为学习率参数。它指定在负梯度方向权值更新步骤的幅度。若$\mu$值选择得太小，学习算法将缓慢修改权值，到达误差曲面底部需要相当多的迭代次数。另一方面，若学习参数的值设定得太大，学习规则可能变得在数值上不稳定。这是由于在式（2-28）中的梯度估计使用了近似值。因此，学习参数设置得太大会导致对误差的重复加倍，使权值不收敛，即可能的发散问题。从式（2-22）和式（2-29）中可分别直接写出LMS算法的标量形式

$$e(k)=d(k)-\sum_{h=1}^{n}w_h(k)x_h(k) \tag{2-30}$$

和

$$w_i(k+1)=w_i(k)+\mu e(k)\,x_i(k) \tag{2-31}$$

其中$i=1,2,\cdots,n$（即对所有突触权值）。

对于LMS算法的连续时间形式，学习率参数$\mu$取任意大而不影响该算法的数值稳定性。然而，对于式（2-22）和式（2-29）的离散时间形式，为了确保稳定性，学习率参数必须设置上界。对LMS算法离散时间形式的收敛性质的严格讨论[19, 22]建立了

$$0<\mu<\frac{2}{\lambda_{\max}} \tag{2-32}$$

给出的学习率参数的界，其中$\lambda_{\max}$代表输入协方差矩阵$\boldsymbol{C}_x$的最大特征值。简单地说，若学习率参数为正，上界为$2/\lambda_{\max}$，则LMS算法按均值收敛。为了使LMS算法收敛，并且对稳定性的影响减小，则学习率参数可接受值通常限制为：

$$0<\mu<\frac{2}{\operatorname{trace}\{\boldsymbol{C}_x\}} \tag{2-33}$$

不等式（2-33）可由*LMS算法的均方收敛分析*得到。在式（2-33）中的学习率参数的界比式（2-32）中的更保守。这是事实，因为

$$\operatorname{trace}\{\boldsymbol{C}_x\}=\sum_{h=1}^{n}\lambda_h=\sum_{h=1}^{n}c_{x_{hh}}\geqslant\lambda_{\max} \tag{2-34}$$

[23]，其中$h=1,2,\cdots,n$，$c_{x_{hh}}$是协方差矩阵$\boldsymbol{C}_x$的对角线元素，并且$\lambda_h$是$\boldsymbol{C}_x$的特征值，且非负。
40 所以，如果LMS算法在均方意义下收敛，那么它也按均值收敛。然而，相反的情形是不正确的。而且，协方差矩阵的迹等于总的输入功率。从而，在式（2-33）中，上界可理解为输入功率总和倒数的两倍。

式（2-32）和式（2-33）假定我们至少有一个输入协方差矩阵的估计。在大多数情况下，要得到这种估计是相当困难的。然而，即使协方差矩阵的估计是可得到的，我们也常设定学习率参数为固定值。虽然，固定学习率参数值是产生最速下降算法的最简单形式，但是随时间改变学习率参数也许更适合。固定学习率参数（即使选择它使得LMS算法收敛）的最主要的问题之一是结果的精度。就是说，若固定的学习率参数设置相对较大，则突触权值的数值精度直接取决于该参数值。参数值越小，结果越精确。然而，若学习率参数设置相当小，

LMS算法的收敛是极其缓慢的。在随机逼近论里，它可以追溯到Robbins和Monro的求根算法1951[24]，学习率参数随时间变化。在随机逼近文献中，最常用的形式为

$$\mu(k)=\frac{\kappa}{k} \tag{2-35}$$

其中$\kappa$为常量。若$\kappa$选择相当小，式（2-35）将保证随机逼近算法的收敛[25, 26]。从式（2-35）中可知，随着训练的进行，$\mu$值将减小。相反，若$\kappa$选择太大，例如超出式（2-32）给出的范围，则即使对于小的时间步长$k$，算法也将发生发散。

采用式（2-35）中的随机逼近进度表的基本问题是学习率参数在初始阶段迅速减小。合理的学习过程似乎是，$\mu$在训练开始阶段很大，然后随着网络收敛逐渐减小。这正好是Darken和Moody[27]的*搜索而且收敛算法*所要完成的。在搜索而且收敛策略的第一阶段（称为搜索阶段），$\mu$相当大，并且几乎是常数（即$\mu$减小得非常慢）；在第二阶段（称为收敛阶段），$\mu$指数减少到0。式（2-36）给出了学习率参数的自适应调整的最简单形式。

$$\mu(k)=\frac{\mu_0}{1+k/\tau} \tag{2-36}$$

在式（2-36）中，$\mu_0>0$和$\tau \gg 1$（称为*搜索时间常量*）。然而，通常$100\leqslant\tau\leqslant500$。如果适当地选择$\mu_0$和$\tau$，用于简单自适应线性组合器的训练的LMS算法，收敛速度可以得到极大改进（参看例2.1）。这些调整学习率参数的方法通常称为*学习率进度表*。从式（2-36）中可知，对 41
于小$k$（即$\mu$自适应的开始阶段），$k$相对于搜索时间常数$\tau$来说很小，$\mu(k)\approx\mu_0$。因此，算法基本表现为具有固定学习率参数$\mu_0$的典型LMS算法。然而，若$k$相对于搜索时间常数$\tau$很大时，则$\mu$的适应性本质上由式（2-35）给出的随机逼近进度表，其中$\kappa=\tau\mu_0$。理想情况下，式（2-36）中的$\mu_0$应在式（2-33）所允许的范围内取相对较大的值。图2-16对于LMS学习率参数，给出了随机逼近和搜索然后收敛的进度表的比较。在式（2-36）中关于$\mu$给出了很多不同的简单调度策略；例如，参看[28, 29]。

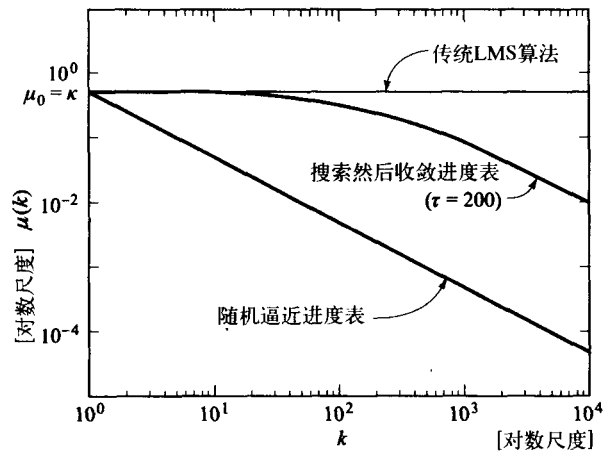



图2-16 两个学习率进度表的比较：随机逼近调度和搜索且收敛调度

用于调整学习率参数的另一种方法是非调度类型的调整，它像随机逼近和搜索且收敛调度一样随着时间值不断减小。相反，第三种方法在调整学习率参数的意义下是自适应正规化

的方法，它按照每时间步的输入数据来调整$\mu$值

$$\mu(k)=\frac{\mu_0}{\|\boldsymbol{x}(k)\|_2^2} \tag{2-37}$$

其中$\mu_0$是固定常量。若$0<\mu_0<2$，则可以确保稳定。然而，实际范围是$0.1\leqslant\mu_0\leqslant1$[16, 22]。表2-1总结了简单自适应线性组合器使用式（2-36）中给出的运用搜索然后收敛策略调整学习率参数的LMS算法。在式（2-35）中的随机逼近进度表或在式（2-37）中的自适应正规化方法，甚至常量学习率参数（对传统LMS算法）均可替代表2-1中用于调整学习率参数的搜索然后收
42 敛的进度表。

**表2-1　LMS算法的小结**

**步骤1**　设$k=1$，初始化突触权值向量$\boldsymbol{w}(k=1)$，并且为$\mu_0$和$\tau$选值。

**步骤2**　按如下公式计算学习率参数

$$\mu(k)=\frac{\mu_0}{1+k/\tau}$$

**步骤3**　计算误差

$$e(k)=d(k)-\sum_{i=1}^{n}w_i(k)x_i(k)$$

**步骤4**　更新突触权值$w_i(k+1)=w_i(k)+\mu(k)\,e(k)\,x_i(k)\quad i=1,2,\cdots,n$。

**步骤5**　如果收敛，停止；否则，设$k\leftarrow k+1$，然后转到步骤2。

**例2.1**　这个例子说明自适应线性组合器通过LMS算法训练来估计线性模型的参数。输入数据由1 000个有三个分量的零均值高斯随机向量组成，也就是，$\boldsymbol{x}\in\Re^{3\times1}$；并且偏置设为零，或$\beta=0$。$\boldsymbol{x}$的分量的方差分别是5，1和0.5。假定的线性模型由$\boldsymbol{b}=[1,\ 0.8,\ -1]^T$给定。为了产生目标值（期望输出），1 000个输入向量用来形成一个矩阵$\boldsymbol{X}=[\boldsymbol{x}_1\boldsymbol{x}_2\cdots\boldsymbol{x}_{1\,000}]$，期望输出按照$\boldsymbol{d}=\boldsymbol{b}^T\boldsymbol{X}$来计算。向量输入信号的协方差矩阵可以估计为[30]

$$\boldsymbol{C}_x\approx\frac{1}{1\,000}\sum_{i=1}^{1\,000}\boldsymbol{x}\boldsymbol{x}^T=\frac{\boldsymbol{X}\boldsymbol{X}^T}{1\,000}$$

在表2-1中使用LMS算法，其中取值$\mu_0=0.9/\lambda_{\max}=0.1936$，其中$\lambda_{\max}$是协方差矩阵$\boldsymbol{C}_x$的最大特征值，$\tau=200$（搜索时间常量），输入向量与相应期望输出值一起提交给线性组合器。用于结束学习过程的标准包括监测每次执行步骤$k$的MSE值的平方根。当$\sqrt{J}=\sqrt{1/2e^2(k)}\leqslant10^{-8}$，学习过程结束，其中$e(k)=d(k)-\boldsymbol{w}^T(k)\,\boldsymbol{x}(k)$。突触权值向量初始值选作为零均值的高斯随机数，其方差为0.25，$\boldsymbol{w}_{\text{initial}}=[-0.304\ 3,\ -0.819\ 5,\ 0.385\ 5]^T$。LMS学习过程在仅仅204次迭代后（训练时期）结束。换句话说，在前204个输入向量与相应期望输出值提交后，网络收敛。最后的突触权值矩阵是$\boldsymbol{w}_{\text{final}}=[1.000\ 000,\ 0.800\ 000,\ -1.000\ 000]^T$，它恰好是假定的线性模式$\boldsymbol{b}$（直到六位小数位置）。事实上，线性模式$\boldsymbol{b}$与最后权值向量$\boldsymbol{w}_{\text{final}}$的差异的$L_2$范式是$\|\boldsymbol{b}-\boldsymbol{w}_{\text{final}}\|_2=1.505\,404\times10^{-7}$。图2-17显示出学习率参数按照搜索然后收敛进度表来调整的进展。正如我们从图上看到的，在训练的开始阶段，$\mu$并没有太大的改变；然后在训练的结束阶段它变小了很多。图2-18显示出性能度量的均方根值（RMS），也就是说，$\sqrt{J}$随网络训练改变。这个练习与估计一个参数向量的系统辨识问题相似，仅仅给定来自系统的输入/输出数据估计系统的动态模
43 型相关的一个参数常量，即参数系统识别[31]。

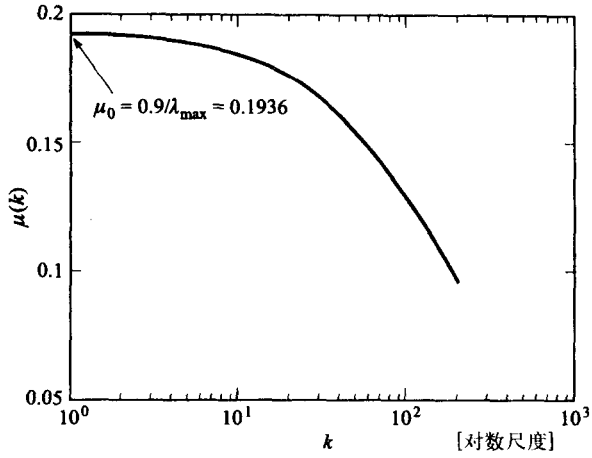



图2-17 用于搜索然后收敛进度表的学习率

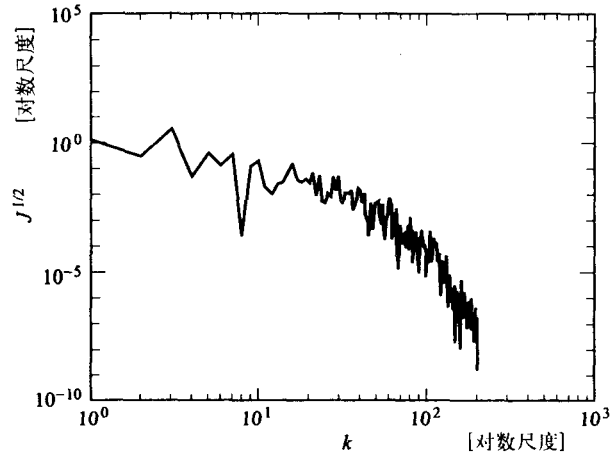



图2-18 由LMS算法训练的自适应线性组合器的收敛

### 2.5.2 自适应线性单元

Adaline（自适应线性单元）是利用LMS算法训练的自适应模式分类网络。Adaline是许多神经网络使用到的基本构件块。图2-19给出了Adaline的结构[11, 16]。可以看出该网络由线性组合器和对称硬限幅器（或对称硬限幅量化器）串联而成，即符号函数。对称硬限幅量化器产生双极（±1）输出，$y(k) = \text{sgn}[v(k)]$，虽然这不是一个限制，比如也可以使用硬限幅器产生的二元输出{0, 1}。与常量输入$x_0(k) = 1$相连接的可调整偏置权值$w_0(k) = \beta$，有效地控制量化器的阈值水平。在网络训练期间，由期望输出和线性组合器输出的差异产生误差，结果误差称为线性误差。线性误差是前面章节所讨论的LMS学习算法的基础。描述误差的另外方法是采用期望输出和对称硬限幅器的输出的差异。这个误差称为量化器误差，如图2-19所示，它是感知器学习规则（参考2.6节）的基础。因此，Adaline和感知器非常相似，在2.6节将讨论它们的不同。 44

在Adaline训练期间，输入向量$\boldsymbol{x} = [1, x_1, x_2, \cdots, x_n] \in \Re^{n+1\times 1}$和相应目标（或期望输出）值$(d \in \Re)$提交给网络。根据线性LMS算法，突触权值$\boldsymbol{w} = [\beta, w_1, w_2, \cdots, w_n] \in \Re^{n+1\times 1}$自适应地改变。在训练Adaline之后，提交输入向量给具有固定权值的网络将导致一个标量输出。因此，

网络实现一个从$n$维向量空间到标量的映射($\Re^n \to \Re$)。在Adaline训练期间，没有用到激活函数。因此，训练过程等同于自适应线性组合器的训练过程。对称硬限幅器（量化器）仅在训练了Adaline之后才使用。一旦适当调整了权值，可以利用在训练阶段没有用过的各种输入测试所训练神经元的响应。若Adaline产生响应（输出）与测试输入以很高的概率相容，则可以说产生了泛化。训练和泛化过程是Adaline的两个非常重要属性，一般来说对神经网络也一样。Adaline的一个常见应用是少部分逻辑函数的实现，如：AND、NOT、OR和MAJ（多数）逻辑函数[9, 32]。只有这些线性可分的逻辑函数能利用单个Adaline实现。其中，三个逻辑函数在数学上可表示为

$$y = \operatorname{sgn}\left[\left(\sum_{j=1}^{n} x_j\right) + 1 - n\right] = \mathrm{AND}(x_1, x_2, \cdots, x_n) = \begin{cases} +1 & \text{若所有} x_j = +1 \\ -1 & \text{其他} \end{cases} \tag{2-38}$$

$$y = \operatorname{sgn}\left[\left(\sum_{j=1}^{n} x_j\right) + n - 1\right] = \mathrm{OR}(x_1, x_2, \cdots, x_n) = \begin{cases} +1, & \text{若有} x_j = +1 \\ -1 & \text{其他} \end{cases} \tag{2-39}$$

$$y = \operatorname{sgn}\left[\sum_{j=1}^{n} x_j\right] = \mathrm{MAJ}(x_1, x_2, \cdots, x_n) = \begin{cases} +1 & \text{若} x_j = +1 \text{为大多数} \\ -1 & \text{其他} \end{cases} \tag{2-40}$$

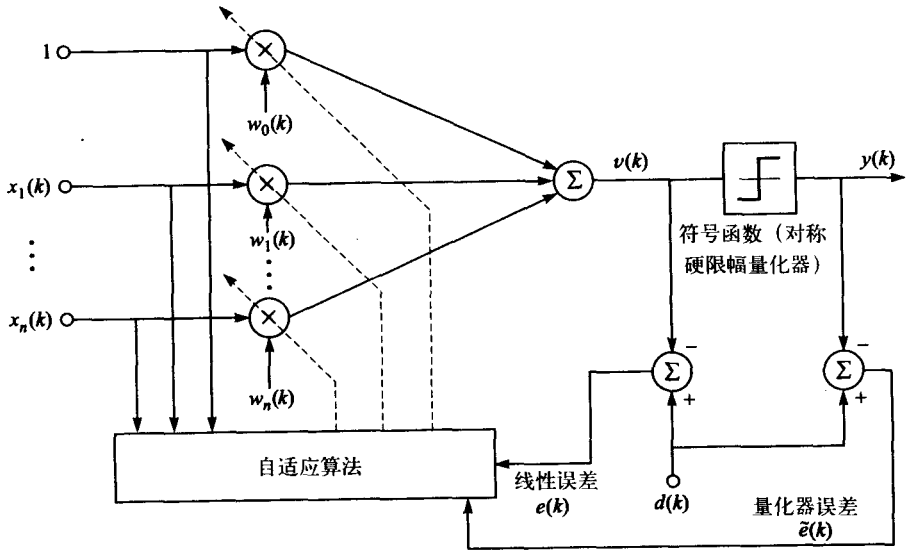



图2-19　自适应线性单元

图2-20给出了分别由式（2-38）、式（2-39）和式（2-40）中的AND, OR和MAJ逻辑函数的单个神经元（Adaline）实现。

**线性可分性**

当提交一个输入模式时，Adaline产生－1或1的输出（假定对称硬限幅器激活函数）。从而，Adaline充当将所有可能输入分成两类的分类器。如图2-21所示，考虑Adaline有2个输入的简单例子。虽然实际上有3个输入，但通常假定一个偏置（阈值）。线性组合器的输出可表示为

$$v(k) = w_1(k)x_1(k) + w_2(k)x_2(k) + w_0(k) \tag{2-41}$$

$v(k)$ 的符号决定了硬限幅器的输出，故此分类的边界线定义为

$$v(k) = 0 \tag{2-42}$$

即

$$w_1(k)x_1(k) + w_2(k)x_2(k) + w_0(k) = 0 \tag{2-43}$$

或

$$x_2(k) = -\frac{w_1(k)}{w_2(k)}x_1(k) - \frac{w_0(k)}{w_2(k)} \tag{2-44}$$

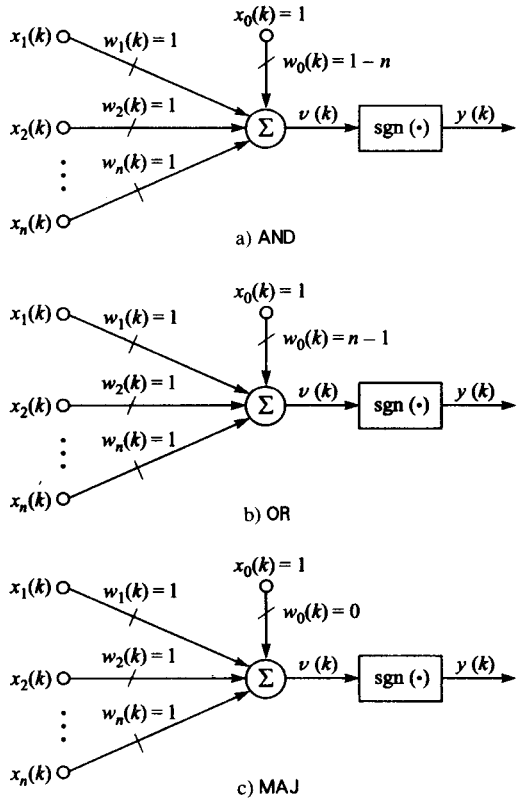



图2-20　a)AND、b) OR和c) MAJ逻辑函数的单个神经元（Adaline）实现。对于c)中的MAJ（多数）逻辑函数，$n$总假定为奇数

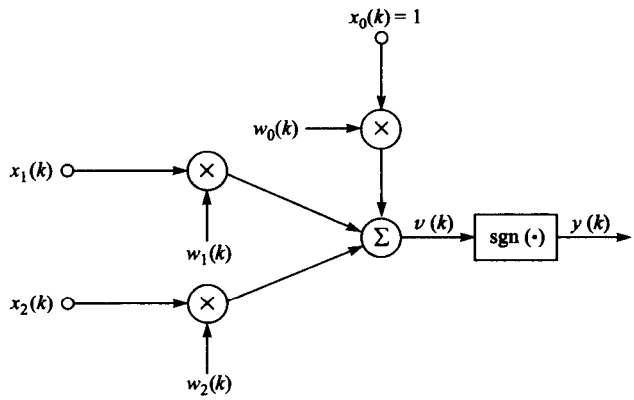



图2-21　两个输入的Adaline

等式（2-44）表示为输入向量的二维向量空间的一条直线。如图2-22所示。

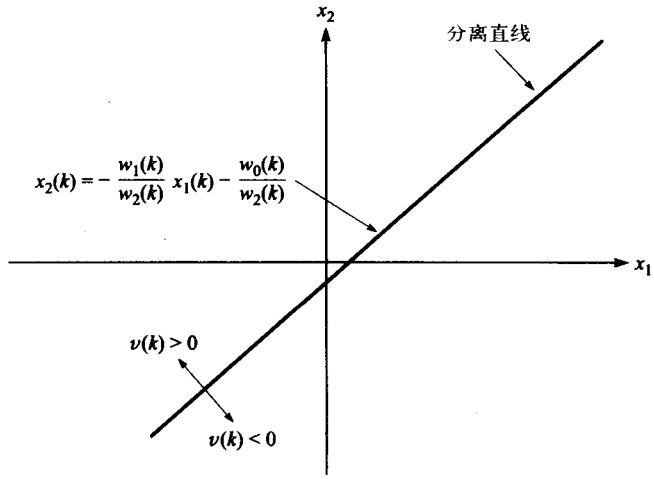
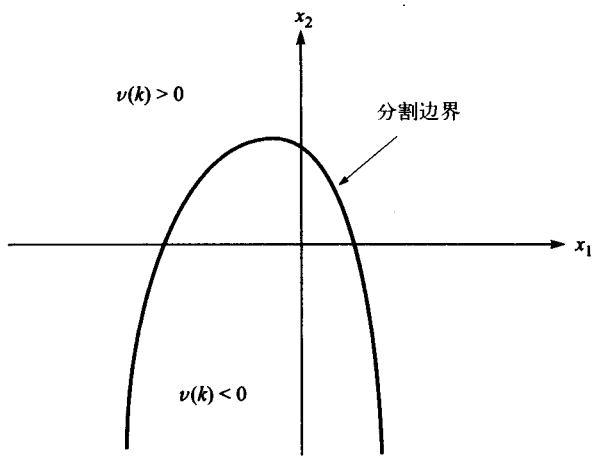



图2-22　Adaline的线性可分性

45
~
47

从图2-22中可以看出，直线把输入空间分割成了两个域；即$v\ (k) > 0$和$v\ (k) < 0$。所有属于同一区域的输入向量将分在同一类,不是1，就是－1。若输入向量（不包括偏置）是3维的，则区域将被平面分割。若输入向量的维数大于3，则边界将是超平面的。一般来说，Adaline代表线性分类器，因此Adaline的应用限制在输入模式是线性可分离的。为了说明这一点，考虑图2-23的例子。假设训练的网络完成对图2-23所示的分离边界表示的输入空间进行分离。在这种情况下，由于边界不是直线，则不存在线性可分，故Adaline不能完成该任务。




48

图2-23　非线性分离问题

具有非线性变换输入的Adaline（多项式判别函数）

为了解决非线性可分离的模式分类问题，Adaline的输入可由固定非线性进行预处理。有用的非线性包括多项式函数预处理网络输入[16]。考虑如图2-24所示的具有二维输入的网络。在这种情形下，输入到对称硬限幅器的信号可表示为：

$$v(k) = w_0(k) + w_1(k)x_1^2 + w_2(k)x_1 + w_3(k)x_1x_2 + w_4(k)x_2 + w_5(k)x_2^2 \qquad (2\text{-}45)$$

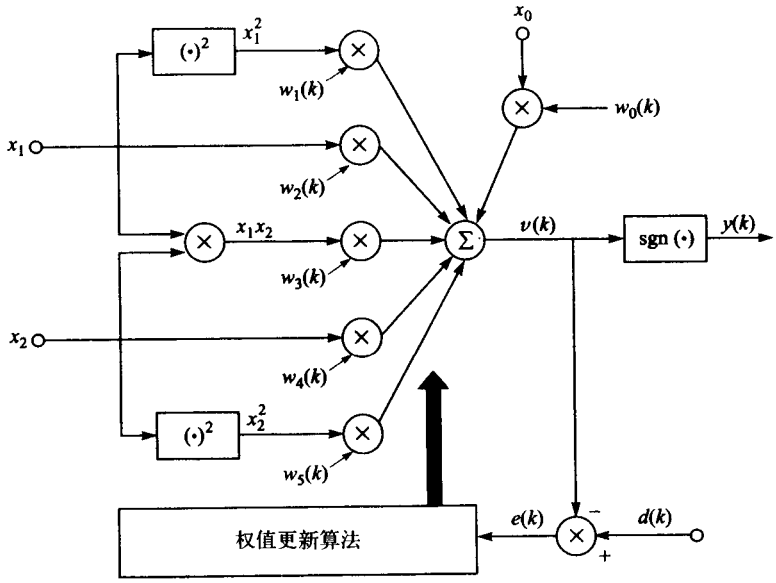



图2-24 带有非线性变换输入的Adaline

当式（2-45）中$v(k)$置为0时，产生用于这个具有非线性变换输入的Adaline的临界值条件。如图2-25所示，这个条件表示二维输入向量空间的椭圆，也是非NOR (XNOR) 问题的解。通过引入输入层的非线性，产生了非直线的分离边界（即，椭圆分离边界）。因此，若恰当选择非线性，可以训练网络将输入空间分割成两个非线性可分离子空间。通常，具有非线性变换输入的Adaline可以使用与训练线性Adaline网络一样的方式进行[16]。

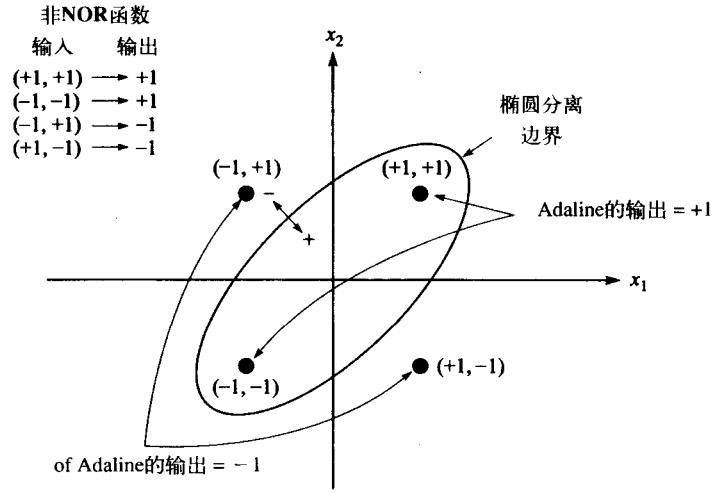



图2-25 实现带有椭圆分离边界的非线性可分离函数（非NOR函数，即，XNOR）。对于这类逻辑门的代数表示为 $y = x_1x_2 + \bar{x}_1\bar{x}_2 = \overline{x_1 \oplus x_2}$，其中 $\bar{x}_j$ 表示 $x_j$ 的NOT（或补操作）

如上面例子所示，非线性的应用可以推广到大于二维的网络输入和许多其他类型的非线性。运用非线性函数变换网络输入的显著优点是构成非线性分离边界成为可能。因此，通过 49
采用单一Adaline可以实现许多自适应非线性判别函数[16]。Specht[33，34]利用多项式判别函数来分类和分析心电图（ECG）数据，并且在这个领域的其他工作可以在[35-38]中找到。

线性误差修正规则

对于Adaline，有两个基本的线性修正规则用来自适应调整网络的突触权值，分别称为$\mu$-LMS和$\alpha$-LMS。对于这些线性修正规则，网络权值的改变与线性组合器的输出和期望的输出信号的差成比例。$\mu$-LMS学习规则同式（2-22）和式（2-29）给出的简单自适应线性组合器的LMS学习规则相同。而$\alpha$-LMS学习规则可视为$\mu$-LMS学习规则的自适应正规化形式。根据$\alpha$-LMS学习规则更新网络的权值为

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)+\alpha\frac{e(k)\boldsymbol{x}(k)}{\|\boldsymbol{x}(k)\|_2^2} \tag{2-46}$$

其中$e(k)$由式（2-22）给出。比较式（2-37）中用于调整在式（2-22）和式（2-29）中的LMS算法的学习率参数的自适应正规化方法和在式（2-46）中的$\alpha$-LMS学习规则，我们发现$\mu_0=\alpha$时二者相同。而且，比较式（2-22）和式（2-29）中传统的LMS算法（具有固定的学习率参数）与式（2-46）中的$\alpha$-LMS学习规则，我们看出两个学习规则的唯一不同在于$\alpha$-LMS算法
50 的正规化项。然而，对于两种学习规则机理的解释是完全不同的。具体地说，$\alpha$-LMS算法是依照最小化扰动原理设计的[16]，即，为了恰当响应新的输入模式而调整时，前面的训练模式的响应（平均）受到扰动最小。因此，$\mu$-LMS建立在MSE曲面的最小化，而$\alpha$-LMS更新权值，以便减小当前误差。为了阐明这一点，对于$\alpha$-LMS，误差的改变可写为[16]

$$\begin{aligned}\Delta e(k)&=e(k+1)-e(k)=[d(k)-\boldsymbol{w}(k+1)^T\boldsymbol{x}(k)]-e(k)\\&=\left\{d(k)-\left[\boldsymbol{w}^T(k)+\alpha\frac{e(k)\boldsymbol{x}^T(k)}{\|\boldsymbol{x}(k)\|_2^2}\right]\boldsymbol{x}(k)\right\}-e(k)\\&=\left[e(k)-\alpha\frac{e(k)\boldsymbol{x}^T(k)\boldsymbol{x}(k)}{\|\boldsymbol{x}(k)\|_2^2}\right]-e(k)=-\alpha e(k)\end{aligned} \tag{2-47}$$

由式（2-47）我们发现$\alpha=-\Delta e(k)/e(k)$，它代表了误差相对变化或是训练过程每一步修正的当前部分的误差。$\alpha$的选择控制稳定性和收敛速度[22]，$\alpha$的通常设置范围是

$$0.1<\alpha<1 \tag{2-48}$$

$\alpha$-LMS算法中$\alpha$的选择不依赖于网络输入的幅度，从这一角度来说它是自正规化的。当输入是双极值[－1, 1]时，在式（2-46）中正规化项$\|\boldsymbol{x}(k)\|_2^2$等同于权值数目，且不随输入模式的变化而变化。然而，若输入是二值[0, 1]时，对于这些具有零输入的权值不会发生调整，但对于双极值输入，所有网络权值每次循环均调整，且收敛趋于更快。因此，双极值输入模式一般常用。

$\mu$-LMS和$\alpha$-LMS学习规则的详细比较

由下面的方式演示$\mu$-LMS和$\alpha$-LMS学习规则的关系。由式（2-46）给出的$\alpha$-LMS算法开始

$$\begin{aligned}\boldsymbol{w}(k+1)&=\boldsymbol{w}(k)+\alpha\frac{e(k)\boldsymbol{x}(k)}{\|\boldsymbol{x}(k)\|_2^2}\\&=\boldsymbol{w}(k)+\alpha\frac{[d(k)-\boldsymbol{w}^T(k)\boldsymbol{x}(k)]\boldsymbol{x}(k)}{\|\boldsymbol{x}(k)\|_2^2}\\&=\boldsymbol{w}(k)+\alpha\left[\frac{d(k)}{\|\boldsymbol{x}(k)\|_2}-\boldsymbol{w}^T(k)\frac{\boldsymbol{x}(k)}{\|\boldsymbol{x}(k)\|_2}\right]\frac{\boldsymbol{x}(k)}{\|\boldsymbol{x}(k)\|_2}\end{aligned} \tag{2-49}$$

51

然后定义

$$\hat{d}(k) \triangleq \frac{d(k)}{\| \boldsymbol{x}(k) \|_2} \tag{2-50}$$

并定义

$$\hat{x}(k) \triangleq \frac{\boldsymbol{x}(k)}{\| \boldsymbol{x}(k) \|_2} \tag{2-51}$$

分别作为正规化期望响应和正规化训练向量。使用式（2-50）和式（2-51）的定义，能够把式（2-49）改写成

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \alpha \left[ \hat{d}(k) - \boldsymbol{w}^T(k) \hat{\boldsymbol{x}}(k) \right] \hat{\boldsymbol{x}}(k) \tag{2-52}$$

它同$\mu$-LMS学习规则的形式一样。因此，我们可以得出结论：对于正规化输入模式而言，$\alpha$-LMS算法表示$\mu$-LMS算法的学习策略。

虽然$\alpha$-LMS算法在分析和实现方面比$\mu$-LMS算法复杂些，但对于给定的传播给网络权值梯度噪声水平而言，当相关联的协方差矩阵$\boldsymbol{C}_x = \mathrm{E}\{\boldsymbol{x}\boldsymbol{x}^T\}$有相对大的特征值范围时，可以证明有更快的收敛[16]。梯度噪声是梯度估计和实际梯度之间的差异。然而，可以证明$\mu$-LMS具有收敛到MSE曲面的最小点的优点，而$\alpha$-LMS对于非双极值输入在均值意义下收敛到最优最小二乘解的近似值[16]。而且，当输入模式具有相同的范数（如均为双极值输入）时，这两个学习规则将产生相同的结果。

非线性权值修正规则

已经证明在某些情况下，前面给出的线性权值修正规则不能分隔某些线性可分离的训练模式[39]。当出现这种情况时，使用非线性学习规则是有助于网络权值的调整。我们将在2.6节讨论学习规则的这些类型。

### 2.5.3 多重自适应线性单元

我们在2.5.2节已经阐述，除非使用输入的非线性变换，单个Adaline不能解决分离边界是非线性的输入空间分离问题。解决非线性分离问题（不是网络非线性变换输入）的方法之一是应用Madaline（多重Adaline）网络。Madaline网络的基本结构是由几个Adaline结合成的单一前馈组织构成。Madaline网络的基本类型有两种：称为Madaline I和Madaline II。Madaline I 52
是最先被Widrow[15]和Hoff[42]引入的单层网络。由三个Adaline构成的Madaline I结构的例子如图2-26所示。为了产生Madaline I网络的输出$y(k)$，把固定逻辑函数（如OR、AND或MAJ逻辑基元）应用于Adalines的输出；Madaline II结构是具有多输出的多层网络[32，43，44]。Madaline II结构的例子如图2-27所示。在这一点上，讨论计算多层网络层数是重要的，多层网络层数定义为拥有处理单元的总层数。因此，在计算多层网络的总层数时不考虑输入层。例：如图2-27所示，有两层处理单元，则Madaline II为两层网络。

在讨论调整Madaline权值的方法之前，先讨论XNOR问题并考虑用Madaline结构实现这一逻辑函数。由两个Adaline构成的两输入Madaline I结构如图2-28所示。Adaline的两输出传递给逻辑AND门，逻辑AND的输出$y$给出逻辑值1或－1。两线性组合器的输出设置为0，即：$v_1(k) = 0$和$v_2(k) = 0$，将在两维输入（模式）空间构成两条边界线

$$x_2(k) = -\frac{w_{11}(k)}{w_{12}(k)} x_1(k) - \frac{w_{10}(k)}{w_{12}(k)} \tag{2-53}$$

和

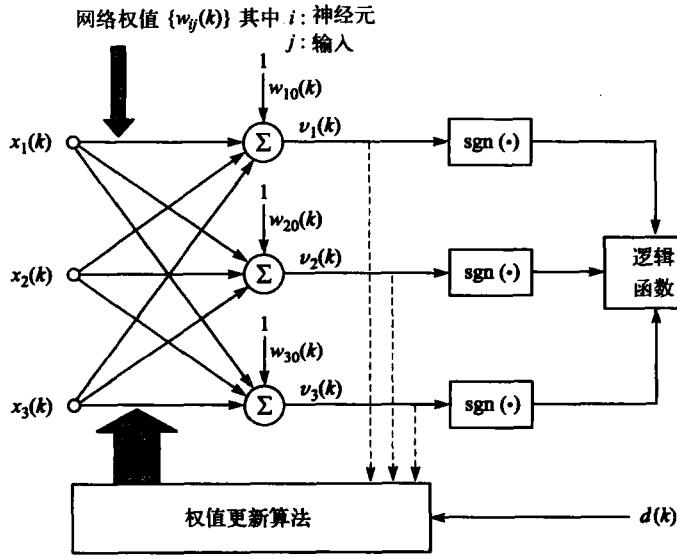



图2-26 包含三个Adaline的Madaline I网络的例子

53

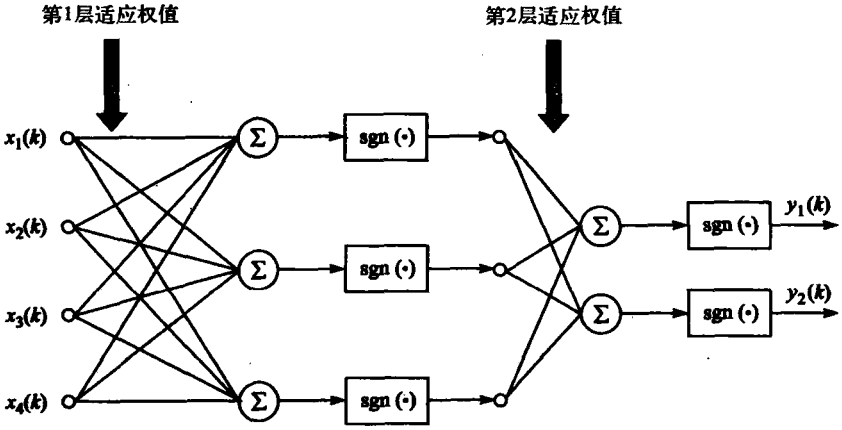



图2-27 两层Madaline II结构

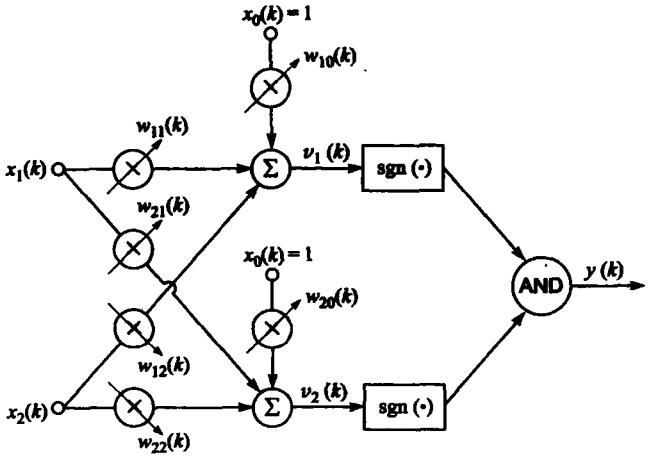



图2-28 XNOR逻辑函数的Madaline I实现

$$x_2(k) = -\frac{w_{21}(k)}{w_{22}(k)}x_1(k) - \frac{w_{20}(k)}{w_{22}(k)} \tag{2-54}$$

只要恰当设置网络权值，分离边界上的边界线可以实现XNOR逻辑函数的功能，如图2-29所示。

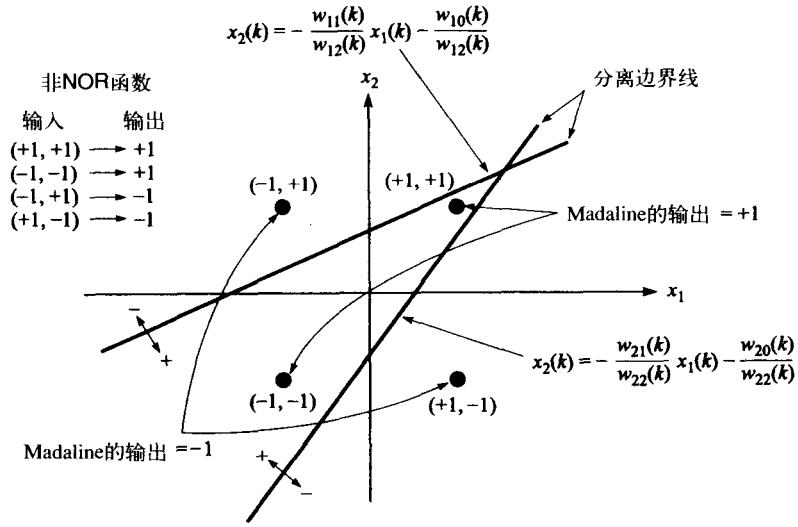



图2-29 用于XNOR问题的Madaline I分离属性

Madaline学习策略

调整Madaline权值有两个基本学习策略[16]。第一个是Madaline规则I(MRI)，它是Madaline I权值自适应的学习策略[32, 43]。在图2-26所示的Madaline I结构中，假定逻辑函数是MAJ函数，权值更新算法将调整权值，修正相对于期望响应$d$的输出$y$。网络权值最初设置为任意小的值。此学习策略有许多不同的变化，如：绝对修正（或“快速”学习）或运用$\alpha$-LMS算法统计（或“慢速”）学习。基本思想是调整线性输出$v_j(k)$最接近0的那些神经元的权值（即：模拟响应最靠近期望响应的神经元），因为这些权值反转其输出响应只需要最小的权值变化。一般地，为了修正输出决策和任何“死区”约束，仅仅调整必需的神经元，在这个意义下MRI遵循最小扰动原则。使用MRI算法有可能“挂停”在那些局部极小值上[16]。第二个基本学习策略是Madaline规则II(MRII)，是MRI的扩展[44]。如MRII将用来调整图2-27所示的两层MRII结构的权值。最初，权值设置为任意小的值。训练模式以随机方式提出，以最小化训练集上的平均汉明误差为目标。与MRI算法一样，MRII也可能“挂停”在局部极小值上[16]。在这些学习策略中达不到目标的重要成分是通过结构反向传播误差的能力，这种结构可能用于调整网络权值[16]。这是能用于多层前馈网络的反向传播学习规则。第3章将详细地讨论反向传播。

54 ~ 55

## 2.6 简单感知器

正如今天我们所知道的，简单感知器（单层感知器）也许对神经网络有非常重要的影响，这如同在2.5.2节提到的，与Adaline非常类似。虽然有几种不同类型的感知器，但是20世纪50年代末期，Frank Rosenblatt[40, 41, 45]提出了原始概念以及调整网络权值的学习程序。他的感知器以神经元的McCulloch-Pitts模型为基础[2]。其他感知器概念在Block[46]和Minsky和Papert[47, 48]的著作中提到。从使用感知器不能解决的问题类型的角度，Minsky和Papert[47]

讨论了感知器的局限性。其中一个重要的局限性就是感知器不能解决异或（XOR）问题。但后来证实只要有恰当的处理层，感知器可以解决XOR问题，或者用它的更一般的形式：奇偶函数 [49]。尽管Minsky和Papert举例说明了简单感知器的局限性，然而，在20世纪70年代关于神经网络研究仍在继续。简单（单层）感知器与称为最大似然高斯分类器[1, 50]的典型模式分类器密切相关，二者都可以认为是线性分类器[51]。

大多数感知器是根据监督学习规则训练的，但一些感知器是自组织的。在Rosenblatt的早期研究中，感知器有三层：第一层是感觉曲面（"视网膜"），它投影到下一层，他称为联想区域，且具有局部化的随机连接。联想区域也称为A单元（即，联想单元），A单元与第三层和最后一层互逆连接，构成R个单元（即，响应单元）。对于给定的输入模式（或输入模式类）激活适当的R单元；并且同一时间只允许激活一个R单元。一系列的互逆连接用于完成这一任务，因此，当一个R单元被激活时，它间接地抑制其他竞争者。作为许多神经网络模型的一部分，胜者全得系统有相似的行为[52]。

图2-19可视为简单感知器。在Rosenblatt的原始感知器中输入是二值的，不包括偏置。在这里我们考虑双极值输入和与神经元有关的偏置。根据图2-19，神经元的输出$y \in \{-1, 1\}$是对称硬限幅器（即量化器）的输出，从期望输出$d \in \{-1, 1\}$减去之后，构成量化器误差$\tilde{e}$。量化器误差用来调整神经元的突触权值。调整神经元权值（感知器学习规则[16]）的自适应算法如下

56

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \alpha \frac{\tilde{e}(k)}{2}\boldsymbol{x}(k) \tag{2-55}$$

其中量化器误差为

$$\tilde{e}(k) = d(k) - \operatorname{sgn}[\boldsymbol{w}^T(k)\boldsymbol{x}(k)] = d(k) - y(k) \tag{2-56}$$

而Rosenblatt通常设定式（2-55）中的$\alpha$为单位值。与$\alpha$-LMS算法不同，学习率参数$\alpha$的选择不影响感知器学习规则的数值稳定性。然而，$\alpha$会影响收敛速度。如在Adaline中那样，由于用（非线性的）量化器误差代替线性误差，所以感知器学习规则是非线性算法。在正确分类所有输入模式之前，感知器学习规则执行权值更新。此后，对于所有训练模式输入，量化器误差为0，不再发生权值调整。由于此学习规则不是基于定义好的优化准则，因此，在任何意义下不能保证权值是最优的。感知器的学习规则同Widrow-Hoff的delta规则相似。但是，其行为是极为不同的[16]。比较式（2-46）中$\alpha$-LMS算法与式（2-55）中的感知器学习规则，我们发现，若式（2-46）中的正规化线性误差$e(k)/\|x(k)\|_2^2$用$\tilde{e}(k)/2$（量化器误差的一半）来代替，可以得到式（2-55）中的感知器学习规则。与Adaline的学习规则不同，已经证明感知器学习规则可以分离任何线性可分离的训练模式[39, 41, 46, 53]。在这里我们将不讨论如何对简单感知器的误差修正学习算法做收敛分析，关于这方面的资料可参考其他书籍[1, 48, 50, 54, 55]。

### 2.6.1 Mays感知器学习规则

Mays提出了标准感知器学习规则的两个修正算法[16]。这两个修正算法均使用所谓的死亡区，在零点附近$\pm\gamma$。若线性组合器的幅度小于$\gamma$，即$|v(k)| < \gamma$，则线性组合器的输出$v(k)$（如图2-19所示）在死亡区内。Mays的权值自适应算法可以总结如下。

Mays的增量自适应算法

$$\boldsymbol{w}(k+1) = \begin{cases} \boldsymbol{w}(k) + \alpha\tilde{e}(k)\dfrac{\boldsymbol{x}(k)}{2\|\boldsymbol{x}(k)\|_2^2} & \text{如果} |v(k)| \geqslant \gamma \\ \boldsymbol{w}(k) + \alpha d(k)\dfrac{\boldsymbol{x}(k)}{\|\boldsymbol{x}(k)\|_2^2} & \text{如果} |v(k)| < \gamma \end{cases} \tag{2-57}$$

其中 $\tilde{e}(k)$（量化器误差）在式（2-56）中给出。若死亡区设置为0，则Mays的增量自适应算法变为式（2-55）给出的感知器学习规则的规范化形式。若训练模式线性可分离，Mays已证明他的增量自适应算法总是收敛的，能在有限步内分离模式。对于训练模式是非线性可分的情形，由于死亡区的原因，Mays的增量自适应算法通常会优于标准感知器学习规则。由于存 57
在非常合理的解，足够大的死亡区将导致权值向量的自适应远离0，并且停留在一个具有相对低的平均误差的区域。对于标准感知器学习规则，非线性可分离输入模式导致训练无法终结，且常常无法产生低误差解（即使解存在）。特别是这种情况下，权值向量趋向于0 。Mays也已经证实，死亡区降低了权值对于误差的灵敏性。在Mays之前，已经有人从不同角度提出了增量自适应算法[46]。

Mays的修正松弛算法为

$$\boldsymbol{w}(k+1)=\begin{cases}\boldsymbol{w}(k) & \text{如果}|\upsilon(k)|\geqslant\gamma\text{和}\ \tilde{e}(k)=0\\ \boldsymbol{w}(k)+\alpha e(k)\dfrac{\boldsymbol{x}(k)}{\|\boldsymbol{x}(k)\|_2^2} & \text{其他}\end{cases}\tag{2-58}$$

其中，$\tilde{e}(k)$为式（2-56）中给出的量化器误差，$e(k)$为式（2-22）中给出的用于简单自适应线性组合器的线性误差。用于感知器的这个学习规则像以前用于Adaline的$\alpha$-LMS算法，当死亡区趋于无穷时($\gamma\to\infty$)，修正的松弛算法逼近标准感知器学习规则。对于死亡区$0<\gamma<1$和学习率$0<\alpha\leqslant 2$，修正松弛算法确保收敛性，并且在有限步内分离任何线性可分离输入模式。然而，若输入模式非线性可分离，算法将与Mays的增量自适应算法相似。Mays的两个感知器学习算法均获得与模式分离相似的结果，对于标准感知器学习规则，数值稳定性都不受学习率$\alpha$选择的影响。

前面对于训练简单感知器的所有讨论均没有涉及基于定义的性能度量的学习策略。在John Shynk1990[56]把感知器学习算法视为基于迭代最小化的瞬时性能函数的最速下降方法，还导出了另一个感知器学习规则。

### 2.6.2 具有S形激活函数的简单感知器

下面考虑图2-30所示的具有S形激活函数的简单感知器替代以前使用的如图2-19所示的对称硬限幅器。像Widrow和Hoff的LMS算法一样，在这种情况下的感知器学习规则以最速下降方法为基础，试图最优化瞬时性能函数。从基于MSE的性能度量导出用于调整网络权值的学习规则，即

$$\boldsymbol{J}(\boldsymbol{w}_q)=\frac{1}{2}\mathrm{E}\{\tilde{e}_q^2(k)\}\tag{2-59}$$ 58

其中 $\tilde{e}_q(k)=d_q(k)-y_q(k)$ 为图2-30给出的非线性误差，E[ · ]为期望算子。然而，受到瞬时值的限制，最小化的瞬时性能函数给出为

$$\boldsymbol{J}(\boldsymbol{w}_q)=\frac{1}{2}\tilde{e}_q^2(k)=\frac{1}{2}[d_q(k)-y_q(k)]^2=\frac{1}{2}[d_q^2(k)-2d_q(k)y_q(k)+y_q^2(k)]\tag{2-60}$$

其中

$$y_q(k)=f[\upsilon_q(k)]=f[\boldsymbol{x}^T(k)\,\boldsymbol{w}_q(k)+\theta_q]\tag{2-61}$$

假定S形激活函数是双曲正切S形函数，见式（2-14）。因此，图2-30中的神经元输出可写为

$$y_q(k)=f_{\mathrm{hts}}[\upsilon_q(k)]=\tanh[\alpha\upsilon_q(k)]\tag{2-62}$$

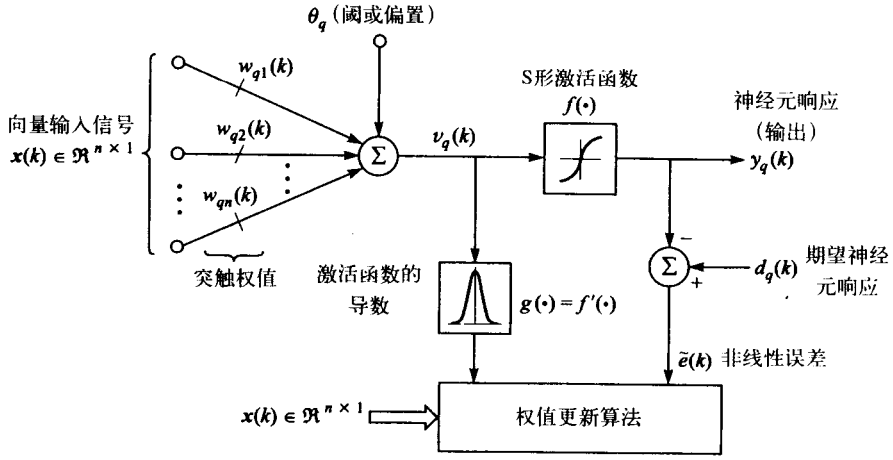



图2-30　带有S形激活函数的简单感知器。这可能是多层前馈感知器的第$q$个神经元（参看2.7节）

其中$\alpha$为函数的倾斜参数。为了方便，我们将省略双曲正切S形函数式（2-62）的下标“hts”。根据式（2-15），双曲正切S形函数关于神经元激活水平$v_q$的导数为

$$g[v_q(k)] = f'[v_q(k)] = \alpha\{1 - f^2[v_q(k)]\} \tag{2-63}$$

我们考虑的突触权值在幅度上是连续的，而在时间上是离散的，因此，式（2-63）中的激活函数的导数是连续导数。在图2-30中，具有S形激活函数的感知器的离散时间学习规则采用最速下降方法，有下面的形式

59

$$\boldsymbol{w}_q(k+1) = \boldsymbol{w}_q(k) - \mu\nabla_{\boldsymbol{w}}\boldsymbol{J}(\boldsymbol{w}_q) \tag{2-64}$$

其中$k$是离散时间指标，而$\boldsymbol{J}(\boldsymbol{w}_q)$是式（2-60）给出的瞬时性能函数。计算式（2-64）的梯度公式为

$$\begin{aligned}\nabla_{\boldsymbol{w}}\boldsymbol{J}(\boldsymbol{w}_q) &= -d_q(k)f'[v_q(k)]\boldsymbol{x}(k) + f[v_q(k)]f'[v_q(k)]\boldsymbol{x}(k)\\ &= \underbrace{\{-d_q(k) + f[v_q(k)]\}}_{-\tilde{e}_q(k)}f'[v_q(k)]\boldsymbol{x}(k)\\ &= -\tilde{e}_q(k)f'[v_q(k)]\boldsymbol{x}(k)\end{aligned} \tag{2-65}$$

根据式（2-63），我们将$f'[v_q(k)] = \alpha\{1 - f^2[v_q(k)]\}$代入式（2-65）中，得

$$\nabla_{\boldsymbol{w}}\boldsymbol{J}(\boldsymbol{w}_q) = -\alpha\tilde{e}_q(k)\{1 - \underbrace{f^2[v_q(k)]}_{y_q^2(k)}\}\boldsymbol{x}(k) = -\alpha\tilde{e}_q(k)[1 - y_q^2(k)]\boldsymbol{x}(k) \tag{2-66}$$

从而，利用式（2-66）的梯度结果，对于简单感知器由式（2-64）我们可写出离散时间学习规则（以向量形式）

$$\boldsymbol{w}_q(k+1) = \boldsymbol{w}_q(k) + \mu\alpha\tilde{e}_q(k)[1 - y_q^2(k)]\boldsymbol{x}(k) \tag{2-67}$$

方程（2-67）可写成标量形式

$$w_{qj}(k+1) = w_{qj}(k) + \mu\alpha\tilde{e}_q(k)[1 - y_q^2(k)]x_j(k) \tag{2-68}$$

其中$j = 1, 2, \cdots, n$。从图2-30可得

$$\tilde{e}_q(k) = d_q(k) - y_q(k) \tag{2-69}$$

和

$$y_q(k) = f[v_q(k)] = f\left[\sum_{p=1}^{n} x_p(k) w_{qp}(k) + \theta_q\right] \tag{2-70}$$

对于图2-30所示的训练简单感知器，式（2-68）、式（2-69）和式（2-70）认为是反向传播训练算法的标准形式[57]。反向传播训练算法在第3章中扩展至前馈多层感知器。同2.3节讨论的一样，二值S形函数和双曲正切S形函数的导数可由它们的原函数表示。这一点可从式（2-63）中的双曲正切S形函数看出。显然，对于激活函数来说，这是一个非常有价值的特征。

**例2.2** 在这个例子中，我们使用图2-30所示的具有S形激活函数的简单感知器来学习一个字符，也就是E字符，如图2-31a所示。该字符（图像）由25个像素组成，即，一个5×5像素数组，图像中黑色像素给定值为数字1，"关闭"（白色）像素给定值为数字0。所以，二值数组为方形，且vec运算（参看A.2.17节）作用于这个数组，将产生一个用于训练感知器的25×1的二值向量，

$$x = [1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0]^T$$

60

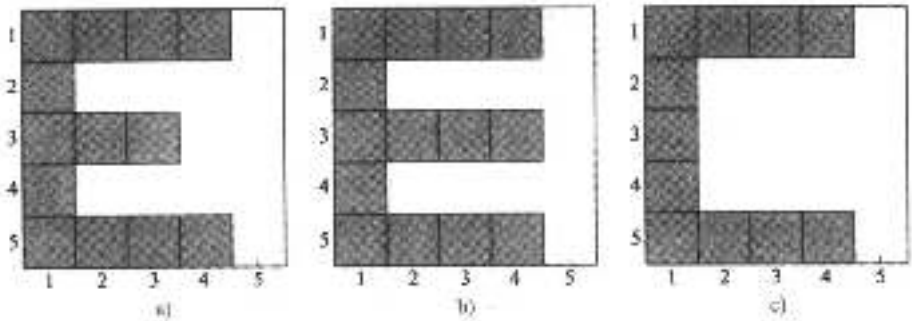



图2-31 a) 原始E字符，b) 用于"测试"的原始字符的修正，c) 另一个用于测试的原始字符的修正

使用式（2-67）的感知器学习规则，$\alpha = 1$，且学习率参数设置为$\mu = 0.25$，期望的神经元响应设置为$d = 0.5$，即训练完成后辨识字符E的数值，而输入模式，上面表示的向量$x$提交给神经元。$10^{-3}$为终止训练的误差目标，换句话说，当期望响应$d$和实际响应$y$的差异的平方小于$10^{-3}$时，停止神经元训练。神经元的初始权值是随机的，在输入模式提交39次之后，实际神经元输出为$y = 0.500\ 09$，图2-32显示了在训练过程中神经元输出误差的情况。对字符识别采用神经计算方法的一个吸引人之处在于网络"校正"或对噪声污染的输入的补偿能力。

61

在这个例子中，可以证明：单个神经元不能校正有噪声的输入；也就是说，它不具备单独校正误差的能力。例如：若修改E如图2-31b所示，结果输入到单个神经元产生$y = 0.520\ 4$的输出。这是一个相对接近确认"真实"E的实际值，因为修改后的图像看起来和E相似。但是，简单感知器没有执行任何的误差校正。若修改E如图2-31c所示，结果神经元的输出为$y = 0.680\ 5$，这与E的真实目标值相差更远。现在图2-31c的图像看起来更像C。但是，这仅仅为了识别E而训练的神经元，却不知道假定C"看起来像"代表什么意思。很明显，需要更多的神经元不仅仅是为了允许对噪声输入的补偿，而是为了识别更多的字符。这个可以利用多层感知器（参考3.3节）来实现。当我们研究霍普菲尔德联想记忆（参考5.3节）时，将证明这个递归网络具有对提交给它的有噪声或不完整的输入进行补偿的能力，并且具备识别许多不同字符的能力。

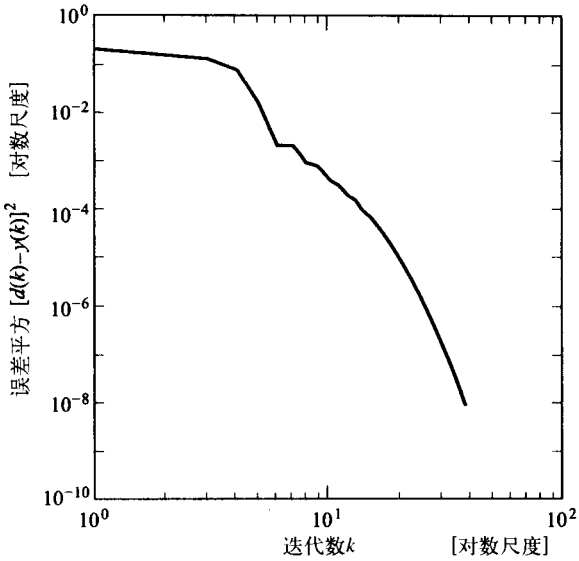



图2-32　训练期间迭代数目与神经元输出误差的平方

## 2.7　前馈多层感知器

图2-33给出了一个具有三层的标准前馈多层感知器（MLP）。这类结构是大量具有神经元级联层的前馈神经网络的组成部分。这种神经网络结构具有一个共同特征：同一层（有时称为槽）的所有神经元通过单向分支连接邻近层的所有神经元。即分支或连接仅能向一个方向即“前馈方向”传递信息。与分支相联系的传送物质，即突触权值，能根据确定的学习规则调整。前馈网络不允许此结构的任何层内的神经元之间相互连接。对于每一个神经元，线性组合器的输出，即神经元激活水平$v_q$，是非线性激活函数$f(\cdot)$的输入,它的输出是神经元的响应。网络中神经元的典型激活水平范围为[－1，1]，在一些应用中使用[0, 1]。在图2-33中实际有四层。但是在2.5.3节中，多层网络的层数定义为只有处理单元的总层数。从而，在图2-33中我们看到“0”层（或输入层）不执行任何计算，只馈送输入信号给“第一”层（第一隐藏层）神经元。第一隐藏层的输入提交给第二隐藏层（“第二”层）神经元，第二层神经元的输出是“第三”层（或输出层）的输入。输出层的输出是网络响应向量。我们可以称这类结构*h-p-m*的前馈MLP神经网络，即*h*个神经元（节点）在第一层（第一隐藏层），*p*个神经元在第二层（第二隐藏层），*m*个节点在第三层（输出层）。因此，网络能执行非线性输入/输出映射$\boldsymbol{\Omega}$：$\Re^{n\times 1}\rightarrow\Re^{m\times 1}$。通常，该结构中隐藏层层数可以任意多。然而，从实践角度看，仅有一个
62 或两个隐藏层应用最广泛。事实上，可以证明只有一个隐藏层，且具有足够多神经元的MLP可以充当非线性映射的通用逼近器（参考3.3.2节）。

在图2-33中，每一层均有从前一层连接到下一层对应的突触权值矩阵，即$\boldsymbol{W}^{(\ell)}$，$\ell=1,2,3$。第一层权值矩阵$\boldsymbol{W}^{(1)}=[w_{ji}^{(1)}]\in\Re^{h\times n}$，第二层的权值矩阵$\boldsymbol{W}^{(2)}=[w_{rj}^{(2)}]\in\Re^{p\times h}$，第三层的权值矩阵$\boldsymbol{W}^{(3)}=[w_{sr}^{(3)}]\in\Re^{m\times p}$，其中$i=1,2,\cdots,n$，$j=1,2,\cdots,h$，$r=1,2,\cdots,p$，$s=1,2,\cdots,m$。从图2-33中，可直接定义非线性输入－输出映射$\boldsymbol{\Omega}$：$\Re^{n\times 1}\rightarrow\Re^{m\times 1}$。首先定义一个对角非线性运算矩阵

$$\boldsymbol{f}^{(\ell)}[\bullet]\triangleq\operatorname{diag}[f^{(\ell)}[\bullet],f^{(\ell)}[\bullet],\cdots,f^{(\ell)}[\bullet]] \tag{2-71}$$

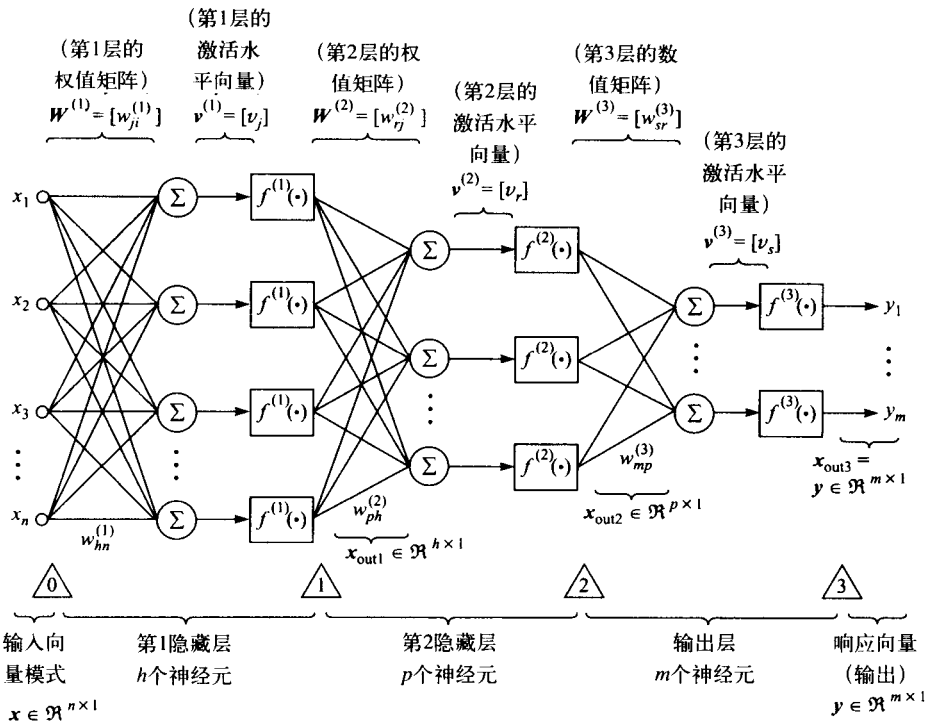



图2-33 前馈三层感知器体系结构，其中$i=1,2,\cdots,n$，$j=1,2,\cdots,h$，$r=1,2,\cdots,p$，$s=1,2,\cdots,m$，$f^{(1)}(\cdot)$是在第一层中每个神经元的非线性激活函数，$f^{(2)}(\cdot)$是在第二层中每个神经元的非线性激活函数，$f^{(3)}(\cdot)$是在第三层中每个神经元的非线性激活函数。每个神经元有一个偏置（或阈值）

63

它的维数取决于$\ell$。当$\ell=1$时，$f^{(1)}[\cdot]$是$h\times h$的对角矩阵；当$\ell=2$时，$f^{(2)}[\cdot]$在维数上是$p\times p$；当$\ell=3$时，$f^{(3)}[\cdot]$是$m\times m$维。给定网络输入向量$x\in\Re^{n\times 1}$。第一层的输出$x_{out1}\in\Re^{h\times 1}$可写为

$$x_{out1}=f^{(1)}[v^{(1)}]=f^{(1)}[W^{(1)}x] \tag{2-72}$$

它是第二层的输入。第二层的输出$x_{out2}\in\Re^{p\times 1}$可写为

$$x_{out2}=f^{(2)}[v^{(2)}]=f^{(2)}[W^{(2)}x_{out1}] \tag{2-73}$$

它是第三层的输入。第三层的输出是网络的响应$y=x_{out3}\in\Re^{m\times 1}$可以表示为

$$y=x_{out3}=f^{(3)}[v^{(3)}]=f^{(3)}[W^{(3)}x_{out2}] \tag{2-74}$$

将式（2-72）代入式（2-73）代替$x_{out1}$，再将这个结果代入式（2-74）代替$x_{out2}$，网络最终响应为

$$y=f^{(3)}[W^{(3)}f^{(2)}[W^{(2)}f^{(1)}[W^{(1)}x]]]=\Omega[x] \tag{2-75}$$

在式（2-75）的非线性映射中，假定突触权值是固定值。然而，在适当调整权值以获得期望映射之前必须执行训练处理，如解决模式分类问题。关于用反向传播的MLPs训练细节将在第3章讨论。

## 2.8 单个神经元基本学习规则概述

这里讲述了单个神经元的几条基本学习规则。这些概念可扩展到多维网络（即不止一个

神经元的网络）。因此，在随后章节中将运用这里提到的许多学习规则建立更复杂的神经结构。

### 2.8.1 广义的LMS学习规则

2.5.1节提出了LMS算法，它最初为了讨论简单线性组合器权值的自适应性而提出。接着证明了利用线性误差项可以应用这个算法训练Adaline（参看2.5.2节）。紧接着，LMS算法的推广导致用于简单（单层）感知器的训练算法（参看2.6节）。最后，在2.6.2节对具有S形激活函数的简单感知器导出学习规则。这里我们想建立单个神经元的广义学习规则，从这个一般
64 形式导出几个重要变化形式。首先，定义最小化的性能函数（或能量函数），记作：

$$\mathscr{E}(\boldsymbol{w})=\psi(e)+\frac{\alpha}{2}\|\boldsymbol{w}\|_2^2 \tag{2-76}$$

其中$\|\boldsymbol{w}\|_2$是向量$\boldsymbol{w}$的欧几里得范数（参看附A.2.13节）。在式（2-76）中$\psi(\cdot)$可以是任何可微函数，$e\in\Re$是线性误差（参看图2-19），即，

$$e=d-\boldsymbol{w}^T\boldsymbol{x} \tag{2-77}$$

其中$d\in\Re$是期望（线性）输出，$\boldsymbol{x}\in\Re^{n+1\times 1}$是输入向量，$\boldsymbol{w}\in\Re^{n+1\times 1}$是权值向量。利用最速下降方法，可以将连续时间的学习规则表示为一组向量微分方程的形式

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t}=-\mu\nabla_{\boldsymbol{w}}\mathscr{E}(\boldsymbol{w}) \tag{2-78}$$

式（2-78）的离散时间形式可以写为：

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)-\mu\nabla_{\boldsymbol{w}}\mathscr{E}(\boldsymbol{w}) \tag{2-79}$$

其中$\boldsymbol{w}(k)=\boldsymbol{w}(kT_s)$，$k$是离散时间项，$T_s$是采样周期（然而，为了不失一般性，假定$T_s$规范化到单位值，即$T_s=1$）。采样周期$T_s$规范化是因为在式（2-78）中欧拉近似的导数可写为$[\boldsymbol{w}(k+1)-\boldsymbol{w}(k)]/T_s$，并且学习规则的离散时间形式为

$$\frac{\boldsymbol{w}(k+1)-\boldsymbol{w}(k)}{T_s}=-\mu\nabla_{\boldsymbol{w}}\mathscr{E}(\boldsymbol{w})$$

然而，两边乘以采样周期$T_s$，两边再加上$\boldsymbol{w}(k)$，将采样周期$T_s$与学习率参数$\mu$合并，得出式（2-79）。在2.5.1节中，式（2-32）给出了LMS算法的学习率参数的取值范围。因此，不管$T_s$的实际值是多少，为了确保LMS算法收敛（在均值意义下），必须满足不等式（2-32）。计算式（2-76）中能量函数的梯度如下

$$\begin{aligned}\nabla_{\boldsymbol{w}}\mathscr{E}(\boldsymbol{w})&=\frac{\partial\mathscr{E}(\boldsymbol{w})}{\partial\boldsymbol{w}}=\frac{\partial\psi(e)}{\partial\boldsymbol{w}}+\frac{\alpha}{2}\frac{\partial\boldsymbol{w}^T\boldsymbol{w}}{\partial\boldsymbol{w}}=\underbrace{\psi'(e)}_{g(e)}(-\boldsymbol{x})+\alpha\boldsymbol{w}\\&=-g(e)\boldsymbol{x}+\alpha\boldsymbol{w}\end{aligned} \tag{2-80}$$

利用式（2-78）中的学习规则的连续时间形式和式（2-80）中的梯度结果，我们可以写出一般的*LMS*算法为

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t}=\mu[g(e)\boldsymbol{x}-\alpha\boldsymbol{w}] \tag{2-81}$$

并且从式（2-79），离散时间形式为

65

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)+\mu[g(e)\boldsymbol{x}(k)-\alpha\boldsymbol{w}(k)] \tag{2-82}$$

其中$\mu > 0$是学习率参数，且$\alpha \geqslant 0$是泄漏因子。若在式（2-76）中，函数$\psi(t)=\frac{1}{2}t^2$（二次加权），且$\psi'(t) = g(t) = t$，其中$t$为哑变量，则式（2-81）可写为

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} = \mu(e\boldsymbol{x} - \alpha\boldsymbol{w}) = \mu e\boldsymbol{x} - \underbrace{\mu\alpha}_{\gamma}\boldsymbol{w} = \mu e\boldsymbol{x} - \gamma\boldsymbol{w} \tag{2-83}$$

而式（2-83）的离散时间形式给出为

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \mu e(k)\,\boldsymbol{x}(k) - \gamma\boldsymbol{w}(k) = (1-\gamma)\,\boldsymbol{w}(k) + \mu e(k)\,\boldsymbol{x}(k) \tag{2-84}$$

其中$0 \leqslant \gamma < 1$是泄漏因子。称此学习规则为*泄漏LMS算法*。现在，若式（2-84）中的泄漏因子设为0，即$\gamma = 0$，则学习规则变为

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \mu e(k)\,\boldsymbol{x}(k) \tag{2-85}$$

它为*标准LMS算法*[参看式（2-29）]。以标量形式可写为

$$w_j(k+1) = w_j(k) + \mu e(k)\,x_j(k) \tag{2-86}$$

对$j = 0, 1, 2, \cdots, n$，其中 $e(k) = d(k) - \sum_{j=1}^{n} w_j(k)x_j(k)$

标准LMS算法有三种重要变化。第一种修改涉及在式（2-85）右边添加动量项。动量项的目的在于在感受到平均下山“力”的方向上提供特定惯量（动量），以改变权值向量，从而避免在训练过程中持续振荡。动量项可以表示为当前和前一步权值向量的权值差异，也就是，

$$\alpha\Delta\boldsymbol{w}(k) = \alpha\,[\boldsymbol{w}(k) - \boldsymbol{w}(k-1)] \tag{2-87}$$

因此，式（2-85）可重写为

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \mu e(k)\,\boldsymbol{x}(k) + \alpha\,[\boldsymbol{w}(k) - \boldsymbol{w}(k-1)] \tag{2-88}$$

其中$0 < \alpha < 1$是*动量参数*，且式（2-88）称为*具有动量的标准LMS算法*。

标准LMS算法的第二种变化是*递归加权最小二乘*。这是标准算法在基于自回归滑动平均（ARMA）模型[7, 31]和自适应滤波[19]的参数系统辨识上的应用。系统辨识问题通常涉及估计与严格真的有理传递函数相联系的参数向量的元素，或涉及线性时间不变系统正则形式的状态空间实现（参看第10章）。[7]中的递归加权最小二乘算法涉及参数向量$\boldsymbol{w} \in \Re^{n+1\times 1}$，*增益向量*$\boldsymbol{L} \in \Re^{n+1\times 1}$，*加权矩阵*$\boldsymbol{P} \in \Re^{n+1\times n+1}$和$\boldsymbol{P}^{\boldsymbol{T}} = \boldsymbol{P}$的更新表达式。参数向量的更新表达式为

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \boldsymbol{L}(k+1)\,e(k) \tag{2-89}$$

66

增益向量更新表达式为

$$\boldsymbol{L}(k+1) = \boldsymbol{P}(k)\,\boldsymbol{x}(k)\,[\lambda + \boldsymbol{x}^T(k)\,\boldsymbol{P}(k)\,\boldsymbol{x}(k)]^{-1} \tag{2-90}$$

和加权矩阵更新表达式为

$$\boldsymbol{P}(k+1) = \frac{1}{\lambda}[\boldsymbol{P}(k) - \boldsymbol{L}(k+1)\boldsymbol{x}^T(k)\boldsymbol{P}(k)] \tag{2-91}$$

其中误差项$e(k)$为

$$e(k) = d(k) - \boldsymbol{w}^T(k)\,\boldsymbol{x}(k) \tag{2-92}$$

且$\lambda$决定加权类型，如：若$0 < \lambda < 1$，导致指数加权递归最小二乘。将式（2-90）代入式（2-89）得

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \underbrace{\frac{e(k)\boldsymbol{P}(k)\boldsymbol{x}(k)}{\lambda + \boldsymbol{x}^T(k)\boldsymbol{P}(k)\boldsymbol{x}(k)}}_{\mu^{-1}(k)} \tag{2-93}$$

因此，修改后的突触权值向量更新表达式为

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)+\mu(k)\,e(k)\,\boldsymbol{P}(k)\,\boldsymbol{x}(k) \tag{2-94}$$

其中

$$\mu(k)=\frac{1}{\lambda+\boldsymbol{x}^T(k)\boldsymbol{P}(k)\boldsymbol{x}(k)} \tag{2-95}$$

是自适应学习率参数。将式（2-90）代入式（2-91），加权矩阵的更新表达式根据增益向量可写为

$$\begin{aligned}\boldsymbol{P}(k+1)&=\frac{1}{\lambda}\left[\boldsymbol{P}(k)-\frac{\boldsymbol{P}(k)\boldsymbol{x}(k)\boldsymbol{x}^T(k)\boldsymbol{P}(k)}{\lambda+\boldsymbol{x}^T(k)\boldsymbol{P}(k)\boldsymbol{x}(k)}\right]\\&=\frac{1}{\lambda}[\boldsymbol{P}(k)-\mu(k)\boldsymbol{P}(k)\boldsymbol{x}(k)\boldsymbol{x}^T(k)\boldsymbol{P}(k)]\\&=\frac{1}{\lambda}[\boldsymbol{I}-\mu(k)\boldsymbol{P}(k)\boldsymbol{x}(k)\boldsymbol{x}^T(k)]\boldsymbol{P}(k)\end{aligned} \tag{2-96}$$

由式（2-96），加权矩阵的更新表达式可写为

$$\boldsymbol{P}(k+1)=\frac{1}{\lambda}[\boldsymbol{I}-\mu(k)\boldsymbol{P}(k)\boldsymbol{x}(k)\boldsymbol{x}^T(k)]\boldsymbol{P}(k) \tag{2-97}$$

其中$\mu(k)$由式（2-95）给出。因此，式（2-94）、式（2-95）和式（2-97）组成*递归加权最小二乘(RWLS)算法*。

标准LMS算法的第三种变化涉及最小干扰原则（参看2.5.2节）。对式（2-46）的分母引进
67 正常数，这样确保权值向量更新不变成无界的。因此，修改的规范化*LMS*算法可写为：

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)+\mu\left\{\frac{e(k)\boldsymbol{x}(k)}{\alpha+\|\boldsymbol{x}(k)\|_2^2}\right\} \tag{2-98}$$

其中$\alpha\geqslant 0$，学习率参数根据$0<\mu<2$来设，通常为$0.1<\mu<1$（参看式（2-48））。在Douglas的文章中[58]，提到规范化的LMS算法的变化推广。这导致一簇基于$L_p$最小化滤波系数变化的类似投影算法集。标准的LMS算法还有其他变化，在Cichocki和Unbehauen[9]中可找到这些变化的总结。表2-2小结了本节讲述的LMS算法的各种离散时间形式。

**表2-2　修改的LMS算法（离散时间，向量矩阵形式）**

| 算法名称 | 算法公式 |
|---|---|
| 一般LMS算法 | $w(k+1)=w(k)+\mu[g(e)x(k)-\alpha w(k)],\ \mu>0,\ \alpha\geqslant 0$ |
| 泄漏LMS算法 | $w(k+1)=(1-\gamma)w(k)+\mu e(k)x(k),\ \mu>0,\ 0\leqslant\gamma<1$ |
| 标准LMS算法 | $w(k+1)=w(k)+\mu e(k)x(k),\ \mu>0$ |
| 具有动量的标准LMS算法 | $w(k+1)=w(k)+\mu e(k)x(k)+\alpha[w(k)-w(k-1)],$ $\mu>0,\ 0<\alpha<1$ |
| 递归加权最小二乘（RWLS）算法 | $w(k+1)=w(k)+\mu(k)e(k)P(k)x(k)$ $\mu(k)=\dfrac{1}{\lambda+x^T(k)P(k)x(k)}\quad 0<\lambda<1$ $P(k+1)=\dfrac{1}{\lambda}[I-\mu(k)P(k)x(k)x^T(k)]P(k)$ |
| 修正的规范化LMS算法 | $w(k+1)=w(k)+\mu\left[\dfrac{e(k)x(k)}{\alpha+\|x(k)\|_2^2}\right],$ $\alpha\geqslant 0,\ 0<\mu<2$，通常$0.1<\mu<1$ |

**例2.3** 本例的问题与例2.1相同。比较表2-2中除普通LMS算法外各种修改的LMS算法和在表2-1小结中的学习率参数的搜索然后收敛进度表的标准LMS算法。不比较分析普通的LMS算法，因为它实际上是另一类鲁棒性的学习算法。学习规则的这些类型将在第8、9章讨论。对于运行在模拟环境下所有情况，使用和例2.1相同的初始权值向量，使用适当的与例2.1中相同的初始学习率即 $\mu_0 = {}^{0.9}/_{\lambda_{max}} = 0.1936$ 。RWLS算法是明显的例外。修正的规范化LMS算法是另一个例外。对于该学习规则，为了在一个合理数量的训练步骤中达到收敛，设置比$\mu_0 = {}^{0.9}/_{\lambda_{max}} = 0.1936$大得多的学习率参数是必要的。另外，对于所有六种情况，使用同样的学习过程结束标准，像例2.1讨论的那样，RMS值 $\sqrt{J} = \sqrt{\frac{1}{2}e^2(k)} \leqslant 10^{-8}$，其中$e(k) = d(k) - \boldsymbol{w}^T(k)\boldsymbol{x}(k)$。 68

对于每个LMS算法，当具体参数适当时，以经验为主地最优化来产生“最好的”性能结果。表2-3给出了模拟结果。图2-34给出每个LMS算法的收敛规则，通过观察表2-3和图2-34，我们明显看出RWLS算法是较好的学习规则。

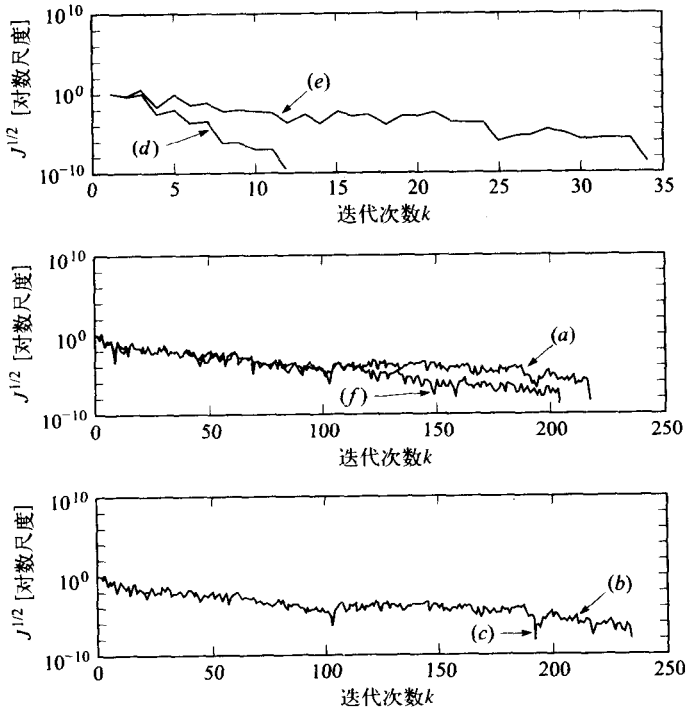



图2-34 每个LMS算法的收敛大致情况：a) 泄漏LMS算法（217次迭代）；b) 标准LMS算法（234次迭代）；c) 带有动量的标准LMS算法（192次迭代）；d)RWLS算法（12次迭代）；e) 修改的规范化LMS算法（34次迭代）；f）带有搜索然后收敛进度表的标准LMS算法（204次迭代）

**表2-3 用于例2.3的模拟结果**

| 使用的算法 | 相关参数 | 收敛所需训练次数 | $\|\boldsymbol{b} - \boldsymbol{w}_{final}\|_2$真实模型：$\boldsymbol{b} = [1, 0.8, -1]^T$ |
|---|---|---|---|
| 泄漏LMS算法 | $\mu_0 = 0.1936, \gamma = 10^{-8}$ | 217 | $1.940\,037 \times 10^{-6}$ |
| 标准LMS算法 | $\mu_0 = 0.1936$ | 234 | $5.803\,031 \times 10^{-7}$ |
| 具有动量的标准LMS算法 | $\mu_0 = 0.1936, \alpha = 0.01$ | 192 | $1.077\,056 \times 10^{-5}$ |

（续）

| 使用的算法 | 相关参数 | 收敛所需训练次数 | $\|\boldsymbol{b}-\boldsymbol{w}_{\text{final}}\|_2$真实模型：$\boldsymbol{b}=[1, 0.8, -1]^T$ |
|---|---|---|---|
| 递归加权最小二乘 (RWLS) 算法 | $\lambda=0.1, \boldsymbol{P}(0)=\boldsymbol{I}_{3\times3}$ | 12 | $3.105\,226\times10^{-10}$ |
| 修正的规范化LMS算法 | $\mu_0=0.999, \alpha=0.1$ | 34 | $3.394\,817\times10^{-7}$ |
| 具有搜索然后收敛调度的标准LMS算法 | $\mu_0=0.193\,6, \tau=200$ | 204 | $1.505\,354\times10^{-7}$ |

### 2.8.2 Hebb学习

在Donald Hebb[4，59]的最初工作中，从神经生物的角度提出了一个学习过程。基于细胞团的概念，Hebb认为皮质网络里的一些细胞子集倾向于按功能单元行动，它们具有整齐的激活模式，和在整个网络中突触强度的变化一致[60]。细胞子集称为团。当A点火后，在非常短的时间延迟内B跟着点火。对于这种情形，Hebb认为在A、B细胞间的突触的强度稍微增加。即突触任意边的两神经元同步激活，然后，突触强度增加。然而，这不是数学描述，Hebb也没有提供定量的数学学习规则。因此，定义的几个数学学习规则可以称为*Hebb*突触。Stent[61]和Changeux和Danchin[62]讲述了Hebb概念的扩展。这些扩展实质上是对Hebb最初描述进行了扩充，使之当突触两边的两个神经元异步激活时导致突触减弱或突触消除。此外，Rumelhart和McClelland[57]还指出Hebb最初的假设对于定量发展一个显式模型是不够的。他们陈述了Hebb最初规则的一个扩展，考虑正和负激活值如下：

根据它们同时激活的乘积调整单元A和B间的连接强度。⊖

简单地说，这个描述暗示，若激活的乘积是正的，则使突触连接的修正更加兴奋；但是，若乘积是负的，则更加抑制对突触连接的修正。

69
~
70

现在我们可以对Hebb突触更精确地定义。*Hebb*突触定义为这样一个突触，高度局部、时间依赖性和强相互作用机制来增强突触有效性，为前突触和后突触激活水平相关的函数[1]。从这个定义出发，Hebb突触的四个主要特性可陈述如下[63]：（1）时间依赖机制。这指的是Hebb突触的变化取决于前突触与后突触激活水平同时发生的准确时间。（2）局部机制。在突触内，在前后突触单元内不断发展的激活水平（局部可用信息）被Hebb突触用于产生输入依赖的局部突触修正。在由Hebb突触构成的神经网络内局部机制中，为非监督学习局部机制提供方法。（3）相互作用机制。Hebb学习的任何形式依赖于（确定的或统计的）前突触和后突触活动的相互作用。（4）连接（相关）机制。在相对短的时间间隔，“共同发生”的前突触和后突触活动对于产生突触修正是足够的。因此，Hebb突触可以称为连接突触（conjunctional synapse）。Hebb学习假设的另一个观点是基于Hebb突触内相互作用机制的统计特征。也就是说，前突触和后突触激活在时间上的相关性确定突触修正。因此，Hebb突触也可称为相关突触（correlational synapse）。

我们可以在Hebb突触内考虑增强或减弱激活。也就是，连接一对神经元的正相关激活，导致突触加强（或提高），然而，另一种不相关或负相关激活产生突触减弱（或突触衰减）。当前突触或后突触激活不是同时发生时，突触衰减也可能发生。突触活动分类为Hebb、反Hebb或非Hebb[64]。相应地，对于正相关的前突触或后突触激活，Hebb突触增强它的强度，当激活是非相关或负相关时，强度下降。反Hebb突触增强负相关前突触和后突触激活，减弱

⊖ 经允许引用自Rumelhart和McClelland[57]，p. 36。

正相关的激活。非Hebb突触并不具有Hebb和反Hebb突触的强交互作用、高度局部、时间依赖机制。

考虑到上面的这些观点，对于单一神经元，可以从如下定义的能量函数导出标准Hebb学习规则

$$\mathscr{E}(\boldsymbol{w}) = -\psi(\boldsymbol{w}^T\boldsymbol{x}) + \frac{\alpha}{2}\|\boldsymbol{w}\|_2^2 \tag{2-99}$$

其中$\boldsymbol{w} \in \Re^{n+1\times 1}$是突触权值向量（包括偏置或阈值），$\boldsymbol{x} \in \Re^{n+1\times 1}$是神经元输入，$\Psi(\cdot)$是可微函数，且$\alpha \geqslant 0$是遗忘因子。并且，

$$y = \frac{\mathrm{d}\psi(\upsilon)}{\mathrm{d}\upsilon} = f(\upsilon) \tag{2-100}$$

是神经元输出，其中$\upsilon = \boldsymbol{w}^T\boldsymbol{x} \in \Re$是神经元的活动水平。利用最速下降方法导出连续学习规则

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} = -\mu\nabla_{\boldsymbol{w}}\mathscr{E}(\boldsymbol{w}) \tag{2-101}$$

其中$\mu > 0$是学习率参数。我们可以看出式（2-99）中关于突触权值向量，能量函数的梯度必须计算出，即$\nabla_{\boldsymbol{w}}\mathscr{E}(\boldsymbol{w}) = \partial\mathscr{E}(\boldsymbol{w})/\partial\boldsymbol{w}$。式（2-99）的梯度可表示为：

71

$$\nabla_{\boldsymbol{w}}\mathscr{E}(\boldsymbol{w}) = -\underbrace{f(\upsilon)}_{y}\underbrace{\frac{\partial\upsilon}{\partial\boldsymbol{w}}}_{\boldsymbol{x}} + \alpha\boldsymbol{w} = -y\boldsymbol{x} + \alpha\boldsymbol{w} \tag{2-102}$$

因此，利用式（2-101）和式（2-102）的结果，单个神经元的连续时间标准Hebb学习规则可写作

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} = \mu[y\boldsymbol{x} - \alpha\boldsymbol{w}] \tag{2-103}$$

离散时间标准Hebb学习规则（以向量形式）可表示为：

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \mu[y(k)\,\boldsymbol{x}(k) - \alpha\boldsymbol{w}(k)] \tag{2-104}$$

以标量离散时间形式为

$$w_j(k+1) = w_j(k) + \mu[y(k)\,x_j(k) - \alpha w_j(k)] \tag{2-105}$$

其中$j = 0, 1, \cdots, n$。

可以从更一般的情况导出上面的结论。Amari[65]证明当选择合适的能量或李雅普诺夫函数$\mathscr{E}(\boldsymbol{w})$时，广义的Hebb学习规则可以看作是梯度优化的过程，即

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} = -\mu\frac{\partial\mathscr{E}(\boldsymbol{w})}{\partial\boldsymbol{w}} \tag{2-106}$$

得到的广义Hebb学习规则的结果为

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} = \mu(\ell\boldsymbol{x} - \alpha\boldsymbol{w}) \tag{2-107}$$

其中$\ell \triangleq \ell(\boldsymbol{w}, \boldsymbol{x}, \upsilon, y, d)$是学习信号，且$d \in \Re$是期望信号。等式（2-107）的离散时间形式可写为

$$\begin{aligned}\boldsymbol{w}(k+1) &= \boldsymbol{w}(k) + \mu[\ell(k)\boldsymbol{x}(k) - \alpha\boldsymbol{w}(k)] \\ &= \boldsymbol{w}(k) + \underbrace{-\mu\alpha}_{\gamma}\boldsymbol{w}(k) + \mu\ell(k)\boldsymbol{x}(k) \\ &= (1-\gamma)\boldsymbol{w}(k) + \mu\ell(k)\boldsymbol{x}(k)\end{aligned} \tag{2-108}$$

其中$\mu > 0$且$0 \leqslant \gamma < 1$（遗忘因子）。从式（2-107）和式（2-108）可导出学习算法的许多形式。假定只根据局部信号调整突触权值，且假定学习信号是神经元的输出，则可从式（2-107）立即导出一个局部学习算法，即，从式（2-100）得

$$\ell = y = \frac{\mathrm{d}\psi(\upsilon)}{\mathrm{d}\upsilon} = f(\upsilon) \tag{2-109}$$

从而，式（2-107）变为

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} = \mu(y\boldsymbol{x} - \alpha\boldsymbol{w}) \tag{2-110}$$

72 它与式（2-103）相同。式（2-110）中出现的$y\boldsymbol{x}$是典型的Hebb共生项。图2-35给出了单个神经元的连续时间标准Hebb学习（局部学习）规则。Hebb学习规则有许多变化形式。一个非常重要的扩展是Oja学习规则。

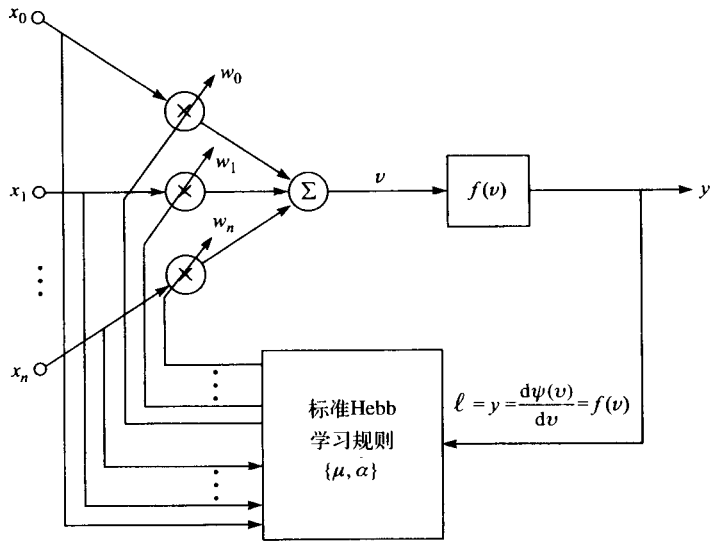



图2-35　单个神经元的标准Hebb学习规则

### 2.8.3　Oja学习规则

利用最小化能量函数可导出Oja学习规则[66]

$$\mathscr{E}(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{e}\|_2^2 \tag{2-111}$$

其中

$$\boldsymbol{e} = \boldsymbol{x} - \hat{\boldsymbol{x}} \tag{2-112}$$

表示误差，给定神经元输入和估计值$\hat{\boldsymbol{x}}$。做两个基本假设：（1）神经元突触权值向量是规范化的，即$\|\boldsymbol{w}\|_2 = 1$；（2）假设一个线性激活函数，即$y = \upsilon = \boldsymbol{w}^T\boldsymbol{x}$。假定输入估计是神经元输出乘以突触权值向量，即

$$\hat{\boldsymbol{x}} = \boldsymbol{w}y \tag{2-113}$$

因此，利用式（2-112）和式（2-113），式（2-111）的能量函数可表示为

73

$$\mathscr{E}(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{w}y\|_2^2 = \frac{1}{2}(\boldsymbol{x}^T - \boldsymbol{w}^T y)(\boldsymbol{x} - \boldsymbol{w}y) = \frac{1}{2}(\boldsymbol{x}^T\boldsymbol{x} - 2\boldsymbol{w}^T\boldsymbol{x}y + \boldsymbol{w}^T\boldsymbol{w}y^2) \tag{2-114}$$

利用最速下降方法，Oja连续时间学习规则可以写为向量微分方程，如下

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} = -\mu\nabla_{\boldsymbol{w}}\mathscr{E}(\boldsymbol{w}) \tag{2-115}$$

其中$\mu > 0$是学习率参数。计算式（2-114）中能量函数的必要梯度为

$$\nabla_{\boldsymbol{w}}\mathscr{E}(\boldsymbol{w}) = -\boldsymbol{x}y + \boldsymbol{w}y^2 \tag{2-116}$$

将式（2-116）代入式（2-115）中，得Oja连续时间学习规则

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} = \mu(\boldsymbol{x}y - \boldsymbol{w}y^2) \tag{2-117}$$

其中

$$y = v = \boldsymbol{w}^T\boldsymbol{x} \tag{2-118}$$

式（2-117）中右边第一项是典型的Hebb共生项，第二项是活跃衰减（或稳定）项，通过规范化向量到单位长度，即$\|\boldsymbol{w}\|_2 = 1$，以阻止突触权值向量变得无界。

在式（2-117）中的学习规则可写为离散时间形式如下

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \mu y(k)\,[\boldsymbol{x}(k) - \boldsymbol{w}(k)\,y(k)] \tag{2-119}$$

式（2-119）的标量形式可写为

$$w_j(k+1) = w_j(k) + \mu y(k)\,[x_j(k) - w_j(k)\,y(k)] \tag{2-120}$$

对$j = 0, 1, \cdots, n$。

从Hebb学习的典型（简单）形式也可导出Oja学习规则[66]。我们仅讨论学习规则的离散时间标量形式。在Hebb学习的简单形式中，学习规则仅包括共生项，即

$$w_j(k+1) = w_j(k) + \mu y(k)\,x_j(k) \tag{2-121}$$

对$j = 0, 1, \cdots, n$。这一简单Hebb学习形式代表了Rumelhart和McClelland[57]对Hebb最初提议的一个推广，其在2.8.2节引用。然而，如果没有规范化（或饱和）形式合并入学习规则中，式（2-121）表示的学习规则将导致突触权值的无限增长。Oja[66]在学习规则中通过将式（2-121）右边除以神经元相关突触的整个集$w_j(k) + \mu y(k)\,x_j(k)$（包括一个塑性系数）的$L_2$范数来实现规范化。在适当的假定下，得到的规范化的学习规则和式（2-120）给出的学习规则相同。由式（2-121）的规范化导出式（2-120）的细节将在9.3.1节给出。当我们研究自适应抽取主成分方法时，规范化Hebb的Oja学习规则在第9章将变得非常重要（参看9.3节）。下面的例子证明了这一点。

74

**例2.4** 此例中我们要分析零均值随机数据，特别是5 000个随机向量，这些向量中各自分量均来自正态（高斯）分布。第一个分量有一个10的方差，其他两个有相等的0.002的方差。由式（2-119）给出的Oja离散时间学习规则用于处理每个提交给单一神经元学习规则的连续向量，并相应地调整权值向量。在式（2-119）中使用固定的学习率参数$\mu = 0.001$。使用一个准则决定训练是否足够。这可以简单地按如下规则监视权值向量收敛过程的进展：

如果$\|\boldsymbol{w}(k) - \boldsymbol{w}(k-1)\|_2 \leqslant 10^{-6}$→停止

否则→继续

初始权值向量随机化到$\boldsymbol{w}(0) = [0.5949, -0.5585, 0.4811]^T$。经过1 182次迭代（即，仅仅前1 182个向量提交给单个神经元）后，达到收敛。最后的突触权值向量是

$$\boldsymbol{w} = [1.0000, -0.0007, 0.0002]^T$$

这是非常接近于使用整个数据集合（即，所有5 000个向量）的估计方差矩阵的第一主成分。这个向量很容易计算为

$$w_1 = [1.0000, 0.0001, 0.0001]^T$$

与该特征向量相关的（最大的）特征值是$\lambda_1 = 9.9772$。可以用适当的MATLAB函数作这种分析。由Oja学习规则可发现，从上面权值向量($w$)计算出的特征值是

$$\lambda = \text{var}(w^T X) = 9.9792$$

其中$X \in \Re^{3\times 5000}$（所有5 000个随机向量），并且相应近似于由估计协方差矩阵计算的最大特征值。var函数是用来计算向量（或矩阵列）方差值的标准MATLAB函数。图2-36a显示当神经元进行训练时权值向量的分量。最终值为上面显示在权值向量$w$中的那些值。注意该向量的范数本质上为1（它是用于Oja学习规则的约束之一）。图2-36b显示出在范数意义下权值向量残留的进展。

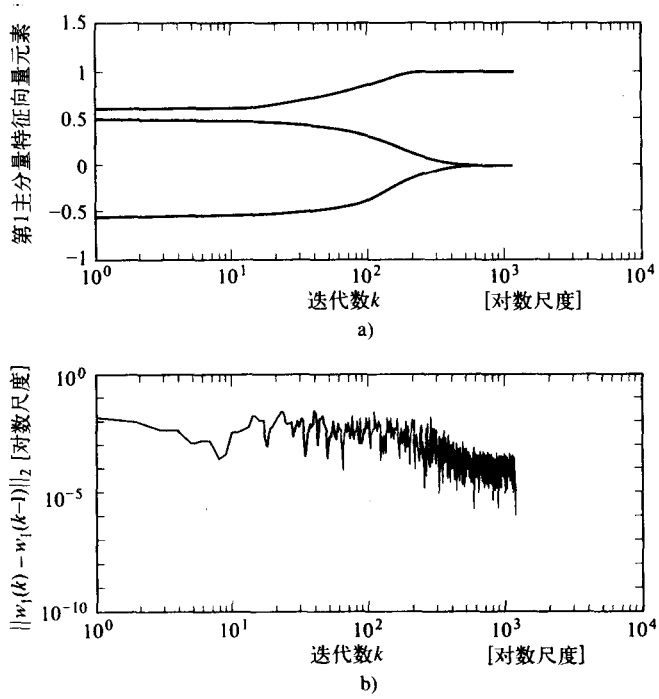



图2-36　a) 使用Oja学习规则训练期间，权值向量元素的收敛；b) 训练期间连续权值向量差异的$L_2$范数

### 2.8.4　位势学习规则

由于位势学习不依赖于期望的信号，所以，它属于无监督学习类型。但其学习的执行完全基于内部电位[65]，也就是$v$，神经元的活动水平。位势学习规则能够通过最小化能量函数来导出

$$\mathscr{E}(w) = -\psi(w^T x) + \frac{\alpha}{2}\|w\|_2^2 \tag{2-122}$$

其中$\alpha \geqslant 0$，$v = w^T x$，并且$\psi(\cdot)$是损失函数。式（2-122）关于突触权值向量的梯度可为

$$\nabla_w \mathscr{E}(w) = -\ell(v)\, x + \alpha w \tag{2-123}$$

其中学习信号是

$$\ell(v)=\frac{\mathrm{d}\psi(v)}{\mathrm{d}v}$$

使用式（2-106）中的Amari结果和式（2-123）的梯度，我们得到连续时间位势学习规则

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t}=\mu[\ell(v)\boldsymbol{x}-\alpha\boldsymbol{w}] \tag{2-124}$$

式（2-124）的离散时间形式可以写为

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)+\mu\,[\ell(v)\,\boldsymbol{x}(k)-\alpha\boldsymbol{w}(k)] \tag{2-125}$$

离散时间标量形式是

$$w_j(k+1)=w_j(k)+\mu\,[\ell(v)\,x_j(k)-\alpha w_j(k)] \tag{2-126}$$

其中$j=0,1,\cdots,n$。

75
~
76

### 2.8.5 相关学习规则

通过最小化如下能量函数，能够导出相关学习规则

$$\mathscr{E}(\boldsymbol{w})=-d\boldsymbol{w}^T\boldsymbol{x}+\frac{\alpha}{2}\|\boldsymbol{w}\|_2^2 \tag{2-127}$$

其中关于突触权值矩阵的梯度为

$$\nabla_{\boldsymbol{w}}\mathscr{E}(\boldsymbol{w})=-d\boldsymbol{x}+\alpha\boldsymbol{w} \tag{2-128}$$

其中$\ell=d$是学习信号，且是关于$\boldsymbol{x}$的期望响应。所以，相关学习是监督学习。同样，使用式（2-106）中Amari结果和式（2-128）的梯度，我们得到连续时间的相关学习规则

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t}=\mu(d\boldsymbol{x}-\alpha\boldsymbol{w}) \tag{2-129}$$

式（2-129）的离散时间形式可以写为

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)+\mu[d(k)\,\boldsymbol{x}(k)-\alpha\boldsymbol{w}(k)] \tag{2-130}$$

离散时间标量形式是

$$w_j(k+1)=w_j(k)+\mu[d(k)\,x_j(k)-\alpha w_j(k)] \tag{2-131}$$

其中$j=0,1,\cdots,n$。相关学习规则通常应用在具有二值响应神经元的存储网络中来记录数据。有趣的是，如果式（2-129）中的$d$用$y$（神经元输出）代替，我们得到式（2-103）的Hebb学习规则（无监督学习）。

### 2.8.6 标准感知器学习规则

在2.6.2节中，我们研究了具有S形激活函数的简单感知器。现在，我们想开发基于任何可微激活函数的更一般的学习规则。生成的学习规则可以称为标准感知器学习规则，能够通过最小化MSE准则（即瞬时能量函数）得到

$$\mathscr{E}(\boldsymbol{w})=\frac{1}{2}e^2 \tag{2-132}$$

其中$e=d-y$。神经元输出可写为

$$y=f(\boldsymbol{w}^T\boldsymbol{x})=f(v) \tag{2-133}$$

$f(\cdot)$是神经元激活函数。现在我们假定阈值（或偏置）项包括在突触权值向量中。所以，$\boldsymbol{w}\in\Re^{n+1\times 1}$。采用最速下降方法，连续时间学习规则为

77

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t}=-\mu\nabla_{\boldsymbol{w}}\mathscr{E}(\boldsymbol{w}) \tag{2-134}$$

其中式（2-132）的梯度为

$$\begin{aligned}\nabla_{\boldsymbol{w}}\mathscr{E}(\boldsymbol{w})&=-d\frac{\mathrm{d}f(\upsilon)}{\mathrm{d}\upsilon}\boldsymbol{x}+\psi(\upsilon)\frac{\mathrm{d}f(\upsilon)}{\mathrm{d}\upsilon}\boldsymbol{x}=-\underbrace{[d-\psi(\upsilon)]}_{e}\frac{\mathrm{d}f(\upsilon)}{\mathrm{d}\upsilon}\boldsymbol{x}\\&=-\underbrace{e\frac{\mathrm{d}f(\upsilon)}{\mathrm{d}\upsilon}}_{\ell}\boldsymbol{x}=-\ell\boldsymbol{x}\end{aligned} \tag{2-135}$$

其中

$$\ell=e\frac{\mathrm{d}f(\upsilon)}{\mathrm{d}\upsilon}=ef'(\upsilon)=eg(\upsilon) \tag{2-136}$$

是学习信号。所以，使用式（2-134）、式（2-135）和式（2-136），能够写出用于单个神经元的连续时间标准感知器学习规则

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t}=\mu\ell\boldsymbol{x} \tag{2-137}$$

等式（2-137）以离散时间形式可写为

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)+\mu\ell(k)\,\boldsymbol{x}(k) \tag{2-138}$$

其中$\ell(k)$是式（2-136）的离散时间形式。式（2-138）中的离散时间学习规则的标量形式可写为

$$w_j(k+1)=w_j(k)+\mu\ell(k)\,x_j(k) \tag{2-139}$$

对$j=0,1,\cdots,n$。

### 2.8.7　广义感知器学习规则

当能量函数不需要满足MSE准则时，在2.8.6节的标准感知器学习规则可以推广。也就是说，我们能够定义一个一般能量函数为：

$$\mathscr{E}(\boldsymbol{w})=\psi(e)=\psi(d-y) \tag{2-140}$$

其中$\psi(\cdot)$是一个可微（加权或损失）函数。如果$\psi(e)=1/2\,e^2$，这将产生一个标准感知器学习规则。然而，对于任何合适的$\psi(\cdot)$函数，*普遍感知器学习规则*能够通过对突触权值向量$\boldsymbol{w}$的最小化式（2-140）得到。通常使用式（2-140）中的广义能量函数来导出*鲁棒*条件下的学习规则（参看第8、9章）。广义能量函数是鲁棒性的，主要在于误差的加权将少于二次，导致拒绝出格点。如式（2-132），MSE性能准则对误差进行二次加权。使用最速下降方法，在式（2-134）中给出了连续时间的一般感知器学习规则形式。所以，必须计算式（2-140）的梯度，

78

并且可以通过使用表链规则来确定

$$\nabla_{\boldsymbol{w}}\mathscr{E}(\boldsymbol{w})=\frac{\partial\psi}{\partial e}\underbrace{\frac{\partial e}{\partial y}}_{-1}\frac{\partial y}{\partial\upsilon}\underbrace{\frac{\partial\upsilon}{\partial\boldsymbol{w}}}_{\boldsymbol{x}} \tag{2-141}$$

其中

$$\frac{\mathrm{d}\psi(e)}{\mathrm{d}e}=\psi'(e)\triangleq\delta(e) \tag{2-142}$$

和

$$y = f(\boldsymbol{w}^T\boldsymbol{x}) = f(\upsilon) \tag{2-143}$$

其中$f(\cdot)$是可微函数，并且

$$\frac{\mathrm{d}y(\upsilon)}{\mathrm{d}\upsilon} = \frac{\mathrm{d}f(\upsilon)}{\mathrm{d}\upsilon} = f'(\upsilon) \triangleq g(\upsilon) \tag{2-144}$$

所以，式（2-141）可以写为

$$\Delta_w \mathscr{E}(\boldsymbol{w}) = -\delta(e)\, g(\upsilon)\, \boldsymbol{x} \tag{2-145}$$

且连续时间的一般感知器学习规则如下

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} = \mu\delta(e)g(\upsilon)\boldsymbol{x} \tag{2-146}$$

其中$\mu > 0$是学习率参数。如果我们定义学习信号为

$$\ell \triangleq \delta(e)g(\upsilon) \tag{2-147}$$

等式（2-146）可写为

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} = \mu\ell\boldsymbol{x} \tag{2-148}$$

广义感知器学习规则的离散时间形式写作

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \mu\ell(k)\, \boldsymbol{x}(k) \tag{2-149}$$

其中$\boldsymbol{w}(k) = \boldsymbol{w}(kT_s)$（$T_s$是采样周期），$\ell(k)$在式（2-147）给出，式（2-149）的标量形式为

$$w_j(k+1) = w_j(k) + \mu\ell(k)\, x_j(k) \tag{2-150}$$

其中$j = 0, 1, \cdots, n$。图2-37表明了广义感知器学习规则。

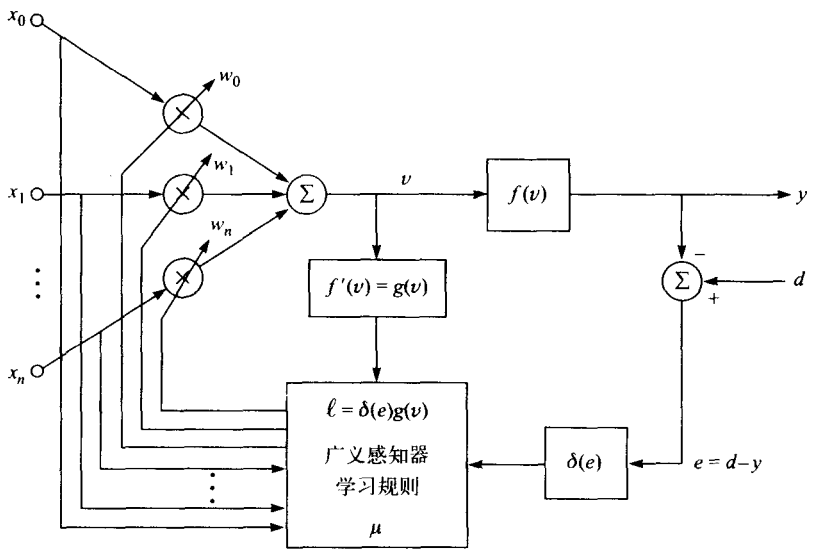



图2-37　广义感知器学习规则

## 2.9　数据预处理

通常，神经网络的性能主要依赖于对训练数据进行的预处理[67, 68]。若对输入模式和目标值执行一定的预处理步骤，则神经网络训练过程会更有效率。也就是说，很多情况下“未

加工的”数据对于训练一个神经网络并不是最好的数据。例如，在用于训练前馈感知器的反向传播算法中，如果使用二值S形函数（参见2.3节）作为网络中神经元的非线性激活函数，饱和度极限是0和1。如果和这些极限相比，训练模式具有较大的值，非线性激活函数几乎能够完全在一个饱和方式下操作，且禁止网络训练。所以，训练数据（输入模式和目标值）应该规整范围来避免这个问题。训练数据的这类预调节定义为规整预处理。将首先讨论这个问题，接着讨论称为变换预处理的问题。这里并不打算介绍数据预处理的所有形式和方法。给出的方法已经广泛使用到训练神经网络的预处理数据。

### 2.9.1 规整

训练数据可以用两种基本方法放缩幅度：使模式的值位于－1和1之间，或使模式的值位于在0和1之间。这两类幅度规整通常称为最小/最大规整。在模糊神经网络条件下，规整输入数据必须在[0, 1]范围内[69, 71]。在MATLAB神经网络工具箱[68]中有函数premnmx可用于规整输入数据，使其在[－1, 1]的范围内。premnmx函数在规整输入数据或规整输入和目标数据时有一个选项。

79 ~ 80

另一个重要的规整过程叫做均值中心和方差规整[68, 72-76]，能够证明对于训练神经网络是有用的。我们假定在矩阵$A \in \Re^{n\times m}$中输入模式（向量）按列排列，在矩阵$C \in \Re^{p\times m}$中目标向量按列排列。均值中心过程涉及对$A$和$C$每行计算一个均值。所以，$A$有$n$个均值，$C$有$p$个均值。对于$A$和$C$的所有行中的某一行，每一元素减去相应的均值。方差规整涉及计算输入矩阵$A$和目标矩阵$C$各行的标准偏差。对于$A$和$C$的各行中某一行，每个元素除以相应的标准偏差。均值中心化和方差能够单独或一起执行。在MATLAB神经网络工具箱[68]里，函数prestd将输入和目标数据或仅仅是输入数据进行均值中心化和方差规整。如果数据包含偏置，均值中心化是重要的；如果$A$是均值中心化的，那么$C$也应该是均值中心化[72, 73]。如果用于训练所收集的数据是由不同的单位所度量，方差规整是可取的。再者，如果$A$是方差规整的，那么$C$也应是方差规整[72, 73]。许多人坚持认为，数据应该总是均值中心化和方差规整的，另一些人则认为，数据决不应该以这种方式预调节[77]。我们认为除非有特殊原因，均值中心化和方差规整数据应该可以在这种方式下预处理。否则，这些不应该随意地执行处理。

### 2.9.2 变换

很多情况下某些“原始的”信号特征用于神经网络的训练输入可以提供比原始的信号本身更好的结果。所以，一个前端特征抽取器能够用来辨别显著的或突出的数据特征，并且这些信号特征随后能够用作训练神经网络的输入。缩减训练模式的输入向量长度是我们非常期望的，这样通常可以减少整个网络体系结构的大小。下面将讨论四种变换方法，它们可以看作特征抽取器以及用于预处理训练数据的预调节信号预处理方法的高期望特征。

### 2.9.3 傅里叶变换

傅里叶（Fourier）变换对预调节训练神经网络的数据有用。从实用的观点，时间信号的快速傅里叶变换（FFT）是执行离散傅里叶变换（DFT）的一个高效方法。FFT嵌入在MATLAB中的fft函数中。使用FFT的谱幅度的主要优点是它对于信号相位不灵敏。所以，如果信号相位不重要，可使用FFT幅度样本作为每个训练模式（信号）的特征向量。特别是，恰当表示信号需要的傅里叶频率的数目远远小于原始信号自身长度。图2-38说明这两点。首先，在图2-38a中，我们可观察到三个完全相同的信号（每个有1024个时间样本）和图2-38b所示相应的FFT 幅度响应（前16个样本）。观察图2-38b，我们看出FFT幅度响应是完全相同的。

81

其次，从图2-38b看来，在所有可能情况下，前16个幅度样本将充分描述1024个样本长度信号，如同数据唯一特征集合一样。所以，16个幅度样本将形成一个训练向量，它比由原始时间域信号表示的输入模式小得多。如图2-38c～e所示，各自的相位数据，实部特征和虚部特征并不具有幅度响应所具有的相不变特征。

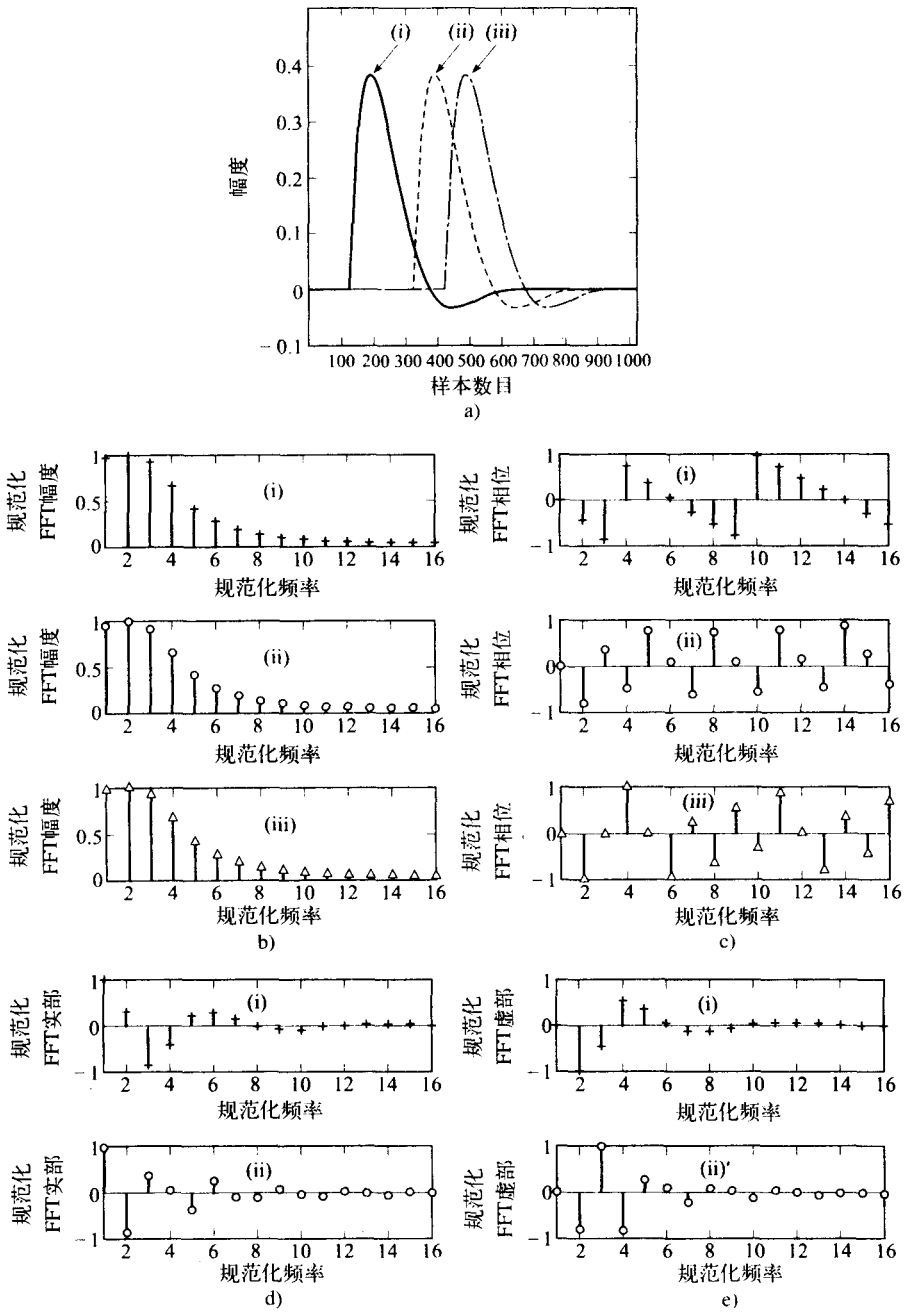



图2-38 a) 三个相同信号在时间上平移。每个信号有1024个时间样本；b) 每个信号的前16个FFT幅度样本；c) 每个信号的前16个FFT相位样本；d) 每个信号的前16个FFT实部样本；e) 每个信号的前16个FFT虚部样本

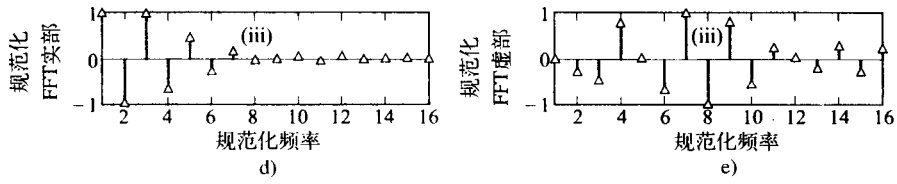



图2-38　(续)

与FFT方法相关的一个方法涉及使用训练模式的复倒谱[20, 78]。复倒谱多次使用可以分离那些卷积信号。这叫做同态解卷积。与用于FFT一样，输入模式的复倒谱用来抽取数据的重要特征，然后，将这些主要特征可以用于训练神经网络。用于说话者（具有鲁棒性）识别的语音信号倒谱处理在Mammone et al.[79]的文章中详细地介绍。

### 2.9.4　主成分分析

主成分分析（PCA）（参看9.2节）能够用来“压缩”输入训练数据集合（或缩减输入维数）。按数据方差的评价确定数据的重要特征，在使用PCA时允许以此缩减输入向量的维数。结果“压缩的”输入向量将具有不相关的元素。如下事实证明，变换后的输入估计协方差矩阵是对角的。在PCA应用之前，数据应该均值中心化。在MATLAB神经网络工具箱中[68]，提供prepca函数对训练数据执行PCA。

82
83

给定一组训练数据$\boldsymbol{A} \in \Re^{n \times m}$，其中假定$m \gg n$。然而，$n$（输入训练模式的维数）假定比较大。使用PCA，可以决定一个“最佳”正交变换矩阵$\boldsymbol{W}_{pca} \in \Re^{h \times n}$（参看9.2节），其中，通常$h \ll n$（维数缩减的程度）。使用这个变换矩阵，输入向量的维数（即，$\boldsymbol{A}$的列数）能够按照如下变换缩减：

$$\boldsymbol{A}_r = \boldsymbol{W}_{pca}\boldsymbol{A} \tag{2-151}$$

其中$\boldsymbol{A}_r \in \Re^{h \times m}$是训练模式的缩减维集合。$\boldsymbol{A}_r$的列是$\boldsymbol{A}$的每个输入（即，$\boldsymbol{A}$中对应的列）的主成分。注意在9.2节中，我们定义正交转换矩阵$\boldsymbol{W}$、在式（2-151）中的$\boldsymbol{W}_{pca}$的行数为正交主特征向量。

### 2.9.5　部分最小二乘回归

部分最小二乘回归（PLSR）（参看9.5节）也可以用来压缩输入训练数据集合。在监督训练的神经网络中PLSR被限制使用，因为其需要输入和目标训练数据。此外，仅仅允许标量目标值（即，标量响应变元）。⊖在PLSR中的因子分析（参看9.5节）可以决定输入数据的压缩程度。也就是说，在确定PLSR因子$h$的最优值后，加权负载向量可以用2.9.4节的PCA相似的方法用来变换数据。所以，最佳数量加权负载向量可以形成一个像矩阵$\boldsymbol{W}_{plsr} \in \Re^{n \times h}$的列数那样的正交变换矩阵。使用这个变换矩阵，输入向量的维（即，$\boldsymbol{A}$的列数）可以按照如下变换压缩。

$$\boldsymbol{A}_r = \boldsymbol{W}_{plsr}^T \boldsymbol{A} \tag{2-152}$$

式（2-152）中的压缩向量实际上在局部最小二乘算法中（参看9.5节）产生的分数。这数据压

⊖ 这是一个限制，仅因为本书中给出了单分量情形（参看9.5节）。但是，一般情况下，PLSR的目标值（响应变量）可以为向量[76]。

缩方法已成功地用来训练一个多层前馈感知器（通过反向传播）[80]。图2-39给出了一个用于数据压缩的PCA与PLSR正交变换向量的比较。两个向量集合的不同之处在于：PLSR使用输入数据和目标数据来产生正交变换$W_{plsr}$的加权负载向量。

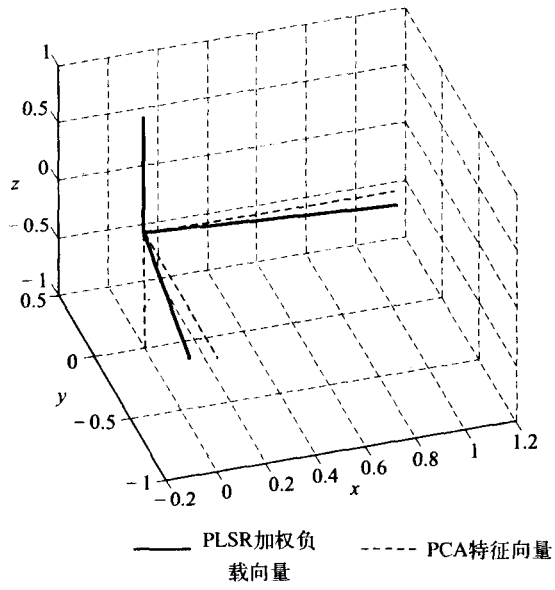



图2-39　用于数据压缩的PCA和PLSR正交变换向量

### 2.9.6　小波和小波变换

通常，波是一个时间（或空间）的振荡函数，例如，具有特定振幅和频率的正弦。傅里叶分析用来分析波动，即某些函数（或信号）能够根据正弦波（或复指数）展开。当我们处理周期的、时不变的或稳定的物理数据时，信号分析方法是非常有用的。所以，在傅里叶级数的情形中，选择正弦函数作为基函数；然后，分析由此得到的展开的性质。傅里叶变换提供了一个分析工具，可以用于将信号变换到频率域，以及给出关于基函数信号频率内容的显式表示（即，合成该信号需要的这个频率成分的多少）。

在小波分析中，定义期望的特性。然后，由此得到基函数。小波能够看作一个小的波[81]，它的能量是集中的（在时间上）。小波是分析时变的、瞬时的或非平稳的信号的有用工具。因此，不适合于使用傅里叶方法分析的许多类型的信号可以采用小波方法进行研究。此外，小波还可以允许同时进行时间和频率的分析。

小波变换[81-84]比传统的傅里叶变换更局部化。也就是说，小波是局部的波；而不是振荡和无限的，它们最终衰减到0。这与建立在无限正弦函数基上的傅里叶变换不同。小波对于信号和图像的压缩、检测和降噪等均有用。小波变换能够提供信号的时频描述并能够压缩用于训练神经网络的数据[85, 86]。在MATLAB小波工具箱中[87]，有一些用于信号和图像的降噪和压缩的函数。

84
~
85

## 习题

2.1　标量函数$f(\boldsymbol{x}):\Re^{n\times 1}\rightarrow\Re$对向量的导数定义为：

$$\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \dfrac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \dfrac{\partial f(\boldsymbol{x})}{\partial x_2} \\ \vdots \\ \dfrac{\partial f(\boldsymbol{x})}{\partial x_n} \end{bmatrix}$$

证明如下等式：

(a) $\dfrac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{h}^T\boldsymbol{x}) = \dfrac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x}^T\boldsymbol{h}) = \boldsymbol{h}$ 其中$\boldsymbol{x}, \boldsymbol{h} \in \Re^{n\times 1}$。

(b) $\dfrac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}) = 2\boldsymbol{A}\boldsymbol{x}$ 其中$\boldsymbol{x} \in \Re^{n\times 1}, \boldsymbol{A} \in \Re^{n\times n}$，并且$\boldsymbol{A} = \boldsymbol{A}^T$。

2.2　考虑图2-19所示的Adaline。使用$\mu$-LMS算法训练网络执行OR逻辑函数。当训练输入使用(a) 双极向量、(b) 二值向量、(c) 比较(a)、(b) 两种情况的收敛速度，解释这个差异。

2.3　使用$\alpha$-LMS算法，重复问题2.2。

2.4　写一个用于训练如图2-24给出的带有非线性变换输入的Adaline的计算机程序，实现逻辑函数XOR。训练输入使用双极向量。解释该网络结构比感知器有更好的可分离特性的原因。

2.5　XOR函数可表示为

A XOR B = (NOT A AND B) OR (A AND NOT B)

(a) 设计一个Madaline网络，实现上面的逻辑函数。

(b) 使用MRI学习规则训练网络。

(c) 画出边界分离线。

2.6　考虑如图2-40所示的分离问题。圆和方形显然不是线性可分离。

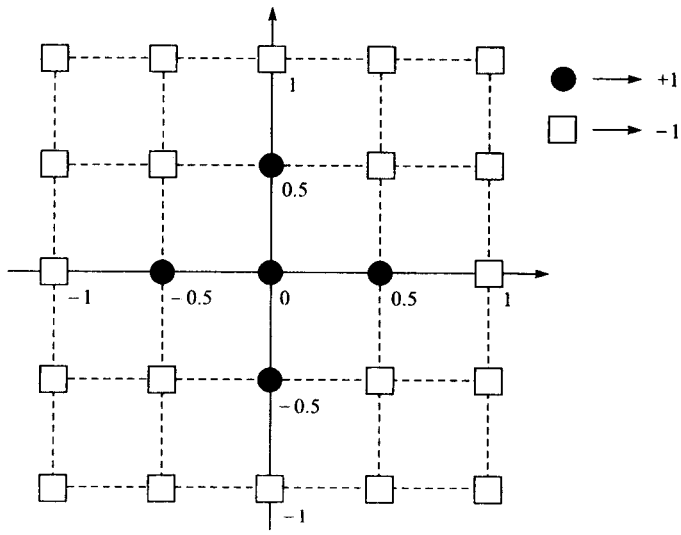



图2-40　问题2.6的模式分离

(a) 写一个计算机程序，实现带有非线性变换输入的Adaline，使用LMS算法来训练。

(b) 使用你的程序分离图2-40给出的圆和方形。

(c) 修改你的代码，以便使用$\alpha$-LMS学习来实现训练。

2.7 考虑一个二维向量集合，该集合定义为：$\{\boldsymbol{x} \in \Re^{n\times 1} \mid -2 \leqslant x_1 \leqslant 2$，并且$-2 \leqslant x_2 \leqslant 2\}$。训练图2-24中带有非线性变换输入的Adaline神经网络来实现下面的分类：

如果 $x_1^2 + x_2^2 < 1$，则$\boldsymbol{x}$分作1类；否则，$\boldsymbol{x}$分作0类。

2.8 写一个使用感知器（参看图2-30）来分类图2-41给出的数字的程序。在输出层神经元数量应该等于数字的个数。每个数字表示成一个$9\times 4$二值（或双极值的）数字矩阵。对每个表示数字的矩阵应用vec运算（参见A.2.17节），从每个数字可产生输入训练模式。当网络训练以后，在数字表示中引入随机噪声，测试神经网络性能。试验输出层具有不同激活函数。

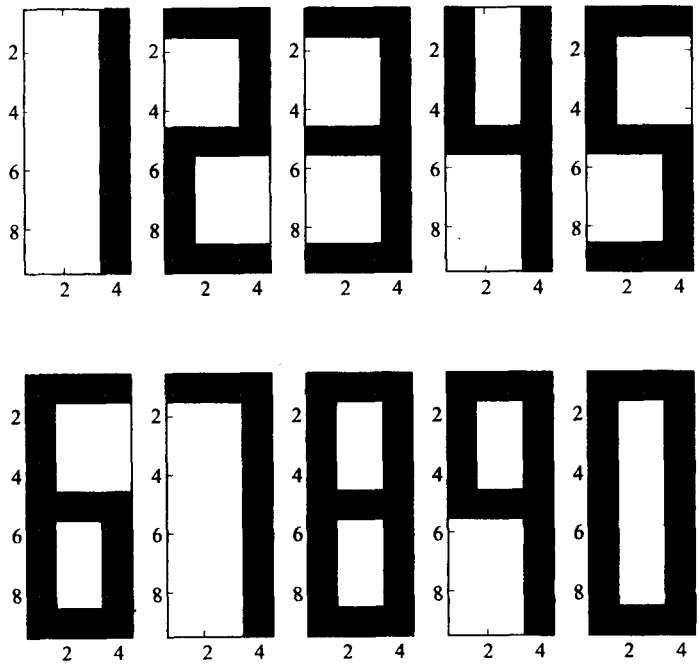


图2-41 用于问题2.8的数字

86
~
87

2.9 (a) 将式（2-27）代入到式（2-25），证明

$$J_{\min}(\boldsymbol{w}) = \frac{1}{2}\mathrm{E}\{d^2(k)\} - \frac{1}{2}\boldsymbol{P}^T\boldsymbol{w}^*$$

(b) 使用在（a）中导出的等式，证明式（2-25）可表示为

$$J(\boldsymbol{w}) = J_{\min}(\boldsymbol{w}) + (\boldsymbol{w} - \boldsymbol{w}^*)^T\boldsymbol{C}_x(\boldsymbol{w} - \boldsymbol{w}^*)$$

2.10 假设一个简单线性组合器的权值按照式（2-27）来设置。

(a) 证明

$$\mathrm{E}\{e(k)\,\boldsymbol{x}(k)\} = \boldsymbol{0}$$

(b) (a) 中结论的一个物理意义是什么？

2.11 自适应线性组合器的最受欢迎的应用之一是用于通信信道的自适应均衡。考虑数据传输到一个未知干扰损坏的轨道的问题，如图2-42所示。对这个问题，假定干扰是频率为$f$的正弦曲线。为了得到一个无干扰的连接，Adaline网络放置在传输线的末端。在信息

数据比特流的传输前，一个已知的序列发送到线上，如图所示。在链接的末端，Adaline比较带有已知序列的接受信号，并且调整它的权值使二者之间的差异最小化。

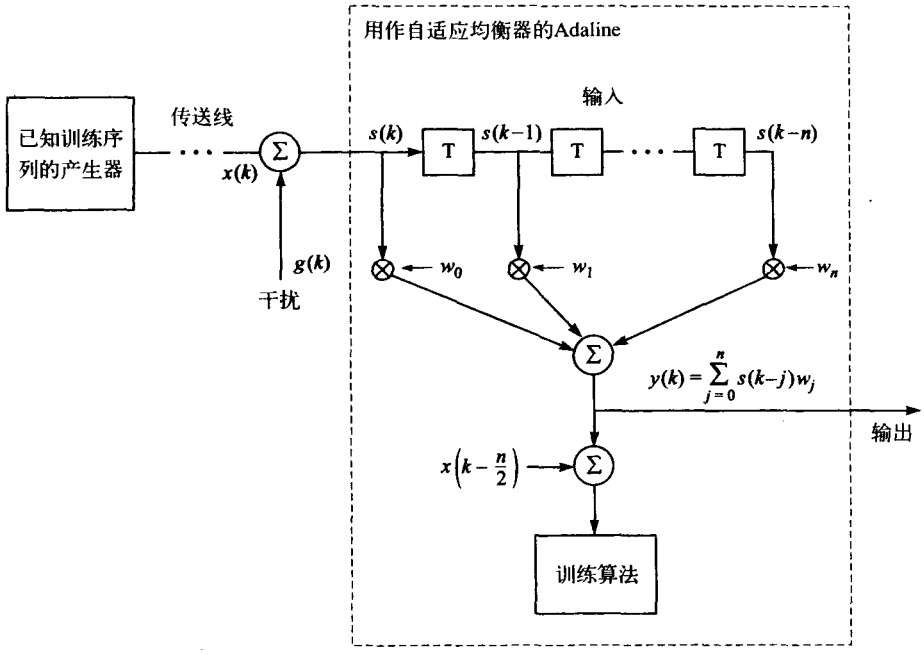



图2-42　使用Adaline的自适应信道均衡

执行一个计算机模拟来实现如下的功能:

(a) 产生一个1024位长的随机比特序列（通过使用一个随机数字发生器来产生一个比特序列）。让每个位用至少四个样本表示（双极值的或二值的）。

(b) 添加一个干扰信号到比特序列（参看图2-43）。

```
N = 1024;                          % size of the sequence
x0 = round(rand(N,1));             % generate a random binary sequence
x = zeros(N*4,1);                  % let each bit be represented with
x(1:4:4*N,1) = x0;                 % four samples
x(2:4:4*N,1) = x0;
x(3:4:4*N,1) = x0;
x(4:4:4*N,1) = x0;

t = 1:4*N;                         % generate the interference
g = sin(2*pi*t/10)';

s = x+g;                           % signal at the end of the channel
```

图2-43　产生训练数据的MATLAB代码的例子

(c) 使用LMS算法训练Adaline，它带有$n$个神经元输入，在输出层有一个神经元。令输入向量是一个信号$s$ ($k$)的延迟版本，且期望的网络输出是图2-42所示的已知序列的相应位。

(d) 在训练之后测试Adaline的输出，并且比较它的输出和无扭曲的已知序列。

(e) 用不同长度的序列、不同的输入神经元数量和不同学习规则（也就是，$\alpha$-LMS，$\mu$-LMS）进行试验。

2.12　在LMS算法的推导中，通过利用瞬时误差表面的导数来估计权值更新的方向。在本质上，MSE曲面的梯度为

$$\nabla_w J(w) = -\mathrm{E}\{d(k)\,x(k)\} + \mathrm{E}\{x(k)\,x^T(k)\}\,w(k)$$

近似为

$$\begin{aligned}\nabla_w J(w) &\approx -d(k)\,x(k) + w^T(k)\,x(k)\,x(k) = -[d(k) - w^T(k)\,x(k)]\,x(k)\\ &= -e(k)\,x(k)\end{aligned}$$

瞬时误差是MSE曲面的最粗略的近似值，为了确保算法的稳定行为，学习率必须保持相对地小。

(a) 使用下面的近似写一个计算机程序

$$\mathrm{E}\{d(k)x(k)\} \cong \sum_{m=k-M+1}^{k} d(m)x(m)$$

且

$$\mathrm{E}\{x(k)x^T(k)\} \cong \sum_{m=k-M+1}^{k} x(m)x^T(m)$$

(b) 用问题2.11描述的自适应均衡的例子测试你的程序。用不同$M$值试验并且讨论结果。例如，比较(a) 中发展的学习算法和原始LMS算法的速度、内存需要、稳定性。 88 ~ 89

2.13　(a) 令$e \in \Re^{n\times 1}$，证明

$$\frac{\partial \|e\|_2}{\partial e} = \frac{e}{\|e\|_2}$$

(b) 在式（2-84）的泄漏LMS学习规则的推导中（参见2.8.1节），我们已经假定$\psi(t) = {}^1/_2 t^2$。通常，为了使学习规则相对于训练样本中的出格点和噪声更具鲁棒性，函数$\psi(\cdot)$定义为$\psi(t) = |t|$。证明在这种情况下，式（2-82）中的一般（离散时间）LMS学习规则假定为如下形式

$$w(k+1) = w(k) + \mu\left[\frac{e}{\|e\|_2}x(k) - \alpha w(k)\right]$$

2.14　在标准LMS算法的应用中屡次遇到的问题是当网络运行在误差曲面的平坦区域时，它性能差。克服该问题的一种方式是使用如下形式的泄漏LMS算法（参看表2-2）

$$w(k+1) = \mu e(k)\,x(k) + (1-\gamma)\,w(k)$$

证明：如果误差曲面相对平坦，泄漏LMS算法能够近似为

$$w(k+1) = \frac{\mu}{\gamma}e(k)x(k) + (1-\gamma)^{n+1}w(k-n)$$

我们可以发现学习率参数的有效值是$\mu_{eff} = {}^{\mu}/_{\gamma}$。

2.15　具有动量的LMS算法为

$$w(k+1) = w(k) + \mu e(k)\,x(k) + \alpha\,[w(k) - w(k-1)]$$

当误差曲面相当平坦时，它胜过标准LMS算法。证明：对于平坦误差曲面，具有动量的更新能够近似为

$$w(k+1) \approx w(k) + \frac{1}{1-\alpha}\mu e(k)x(k)$$

90 注意现在有效学习率是$\mu_{eff} = [1/(1-\alpha)]\mu$。

## 参考文献


1. S. Haykin, *Neural Networks—A Comprehensive Foundation*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 1999.
2. W. S. McCulloch and W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, vol. 5, 1943, pp. 115–33. Reprinted in 1988, Anderson and Rosenfeld [4], pp. 18–27.
3. F. H. Eeckman and W. J. Freeman, "The Sigmoid Nonlinearity in Neural Computation: An Experimental Approach," in *Neural Networks for Computing*, ed. J. S. Denker, New York: American Institute of Physics, 1986, pp. 135–45.
4. J. A. Anderson and E. Rosenfeld, eds., *Neurocomputing: Foundations of Research*, Cambridge, MA: M.I.T. Press, 1988.
5. J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proceedings of the National Academy of Sciences*, vol. 79, 1982, pp. 2554–8. Reprinted in 1988, Anderson and Rosenfeld [4], pp. 460–4.
6. R. V. Churchill, J. W. Brown, and R. F. Verhey, *Complex Variables and Applications*, 3rd ed., New York: McGraw-Hill, 1976.
7. G. F. Franklin, J. D. Powell, and M. L. Workman, *Digital Control of Dynamical Systems*, 2nd ed., Reading, MA: Addison-Wesley, 1990.
8. D. A. Bell, *Operational Amplifiers: Applications, Troubleshooting, and Design*, Englewood Cliffs, NJ: Prentice-Hall, 1990.
9. A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, New York: Wiley, 1993.
10. B. C. Kuo, *Digital Control Systems*, 2nd ed., New York: Saunders College Publ., 1992.
11. B. Widrow and M. E. Hoff, Jr., "Adaptive Switching Circuits," *IRE WESCON Convention Record*, part 4, New York, IRE, 1960, pp. 96–104. Reprinted in 1988, Anderson and Rosenfeld [4], pp. 126–34.
12. M. Caudill, "Neural Networks Primer: Part II," *AI Expert*, 1988, pp. 55–61.
13. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing*, vol. 1, chap. 8, eds. D. E. Rumelhart and J. L. McClelland, Cambridge, MA: M.I.T. Press, 1986.
14. B. Widrow, "ADALINE and MADALINE—1963, Plenary Speech," *Proceedings of First IEEE International Conference on Neural Networks*, vol. 1, San Diego, CA, June 23, 1987, pp. 145–58.
15. B. Widrow, "Generalization and Information Storage in Networks of Adaline 'Neurons,'" in *Self-Organizing Systems 1962*, eds. M. Yovitz, G. Jacobi, and G. Goldstein, Washington: Spartan Books, 1962, pp. 435–61.
16. B. Widrow and M. A. Lehr, "30 Years of Neural Networks: Perceptron, Madaline and Backpropagation," *Proceedings of the IEEE*, vol. 78, 1990, pp. 1415–42.
17. D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd ed., Reading, MA: Addison-Wesley, 1984.
18. S. G. Nash and A. Sofer, *Linear and Nonlinear Programming*, New York: McGraw-Hill, 1996.
19. S. Haykin, *Adaptive Filter Theory*, 3rd ed., Upper Saddle River, NJ: Prentice-Hall, 1996.
20. J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 3rd ed., Upper Saddle River, NJ: Prentice-Hall, 1996.
91 21. N. Wiener, *The Extrapolation, Interpolation, and Smoothing of Stationary Time Series*, New York: Wiley, 1949.

22. B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1985.
23. G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd ed., Baltimore, MD: Johns Hopkins University Press, 1996.
24. H. Robbins and S. Monro, "A Stochastic Approximation Method," *Annals of Mathematical Statistics*, vol. 22, 1951, pp. 400–7.
25. L. Ljung, "Analysis of Recursive Stochastic Algorithms," *IEEE Transactions on Automatic Control*, vol. AC-22, 1977, pp. 551–75.
26. H. J. Kushner and D. S. Clark, *Stochastic Approximation Methods for Constrained and Unconstrained Systems*, New York: Springer-Verlag, 1978.
27. C. Darken and J. Moody, "Towards Faster Stochastic Gradient Search," in *Advances in Neural Information Processing Systems 4*, eds. J. E. Moody, S. J. Hanson, and R.P. Lippmann, San Mateo, CA: Morgan Kaufmann, 1992, pp. 1009–16.
28. C. Darken and J. Moody, "Learning Rate Schedules for Faster Stochastic Gradient Search," in *Proceedings of the 1992 IEEE Workshop on Neural Networks for Signal Processing 2*, Piscataway, NJ: IEEE Press, 1992.
29. R. S. Sutton, "Gain Adaptation Beats Least Squares?" in *Proceedings of the Seventh Yale Workshop on Adaptive and Learning Systems*, New Haven, CT: Yale University, 1992, pp. 161–6.
30. G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis—Forecasting and Control*, 3rd ed., Englewood Cliffs, NJ: Prentice-Hall, 1994.
31. L. Ljung, *System Identification: Theory for the User*, Englewood Cliffs, NJ: Prentice-Hall, 1987.
32. B. Widrow and R. G. Winter, "Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition," *IEEE Computer*, vol. 31, 1988, pp. 25–39.
33. D. F. Specht, "Vector Cardiographic Diagnosis Using the Polynomial Discriminant Method of Pattern Recognition," *IEEE Transactions on Biomedical Engineering*, vol. BME-14, 1967, pp. 90–95.
34. D. F. Specht, "Generation of Polynomial Discriminant Functions for Pattern Recognition," *IEEE Transactions on Electronic Computation*, vol. EC-16, 1967, pp. 308–19.
35. A. R. Barron, "Adaptive Learning Networks: Development and Application in the United States of Algorithms Related to GMDH," in *Self-Organizing Methods in Modeling*, ed. S. J. Farlow, New York: Marcel Dekker, 1984, pp. 25–65.
36. A. R. Barron, "Predicted Squared Error: A Criterion for Automatic Model Selection," in *Self-Organizing Methods in Modeling*, ed. S. J. Farlow, New York: Marcel Dekker, 1984, pp. 87–103.
37. A. R. Barron and R. L. Barron, "Statistical Learning Networks: A Unifying View," *1988 Symposium on the Interface: Statistics and Computing Science*, Reston, VA, April 21–23, 1988, pp. 192–203.
38. A. G. Ivakhnenko, "Polynomial Theory of Complex Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-1, 1971, pp. 364–78.
39. C. H. Mays, "Adaptive Threshold Logic," Ph.D. thesis, Tech. Report 1557-1, Stanford Electronic Labs., Stanford, CA, April 1963.
40. F. Rosenblatt, "On the Convergence of Reinforcement Procedures in Simple Perceptrons," *Cornell Aeronautical Lab. Report VG-1196-G-4*, Buffalo, NY, February 1960.
41. F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Washington: Spartan Books, 1962.
42. M. E. Hoff, Jr., "Learning Phenomena in Networks of Adaptive Switching Circuits," Ph.D. thesis, Tech. Report 1554-1, Stanford Electronic Labs.,

92

Stanford, CA, July 1962.
43. B. Widrow, R. G. Winter, and R. Baxter, "Learning Phenomena in Layered Neural Networks," in *Proceedings of the First IEEE International Conference on Neural Networks*, vol. 2, San Diego, CA, June 1987, pp. 411–29.
44. R. G. Winter, "Madaline Rule II: A New Method for Training Networks of Adalines," Ph.D. thesis, Stanford University, Stanford, CA, January 1989.
45. F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, *Psychological Review*, vol. 65, 1958, pp. 386–408. Reprinted in 1988, Anderson and Rosenfeld [4], pp. 92–114.
46. H. D. Block, "The Perceptron: A Model for Brain Functioning. I," *Reviews of Modern Physics*, vol. 34, 1962, pp. 123–35. Reprinted in 1988, Anderson and Rosenfeld [4], pp. 138–50.
47. M. L. Minsky and S. A. Papert, *Perceptrons*, Cambridge, MA: M.I.T. Press, 1969. Introduction, pp. 1–20, and p. 73 (Figure 5.1) reprinted in 1988, Anderson and Rosenfeld [4], pp. 161–9.
48. M. L. Minsky and S. A. Papert, *Perceptrons*, expanded ed., Cambridge, MA: M.I.T. Press, 1988.
49. N. K. Bose and P. Liang, *Neural Network Fundamentals with Graphs, Algorithms, and Applications*, New York, McGraw-Hill, 1996.
50. R. P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, April 1987, pp. 4–22.
51. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, New York: Wiley, 1973.
52. J. A. Feldman and D. H. Ballard, "Connectionist Models and Their Properties," *Cognitive Science*, vol. 6, 1982, pp. 205–54. Reprinted in 1988, Anderson and Rosenfeld [4], pp. 484–507.
53. N. Nilsson, *Learning Machines*, New York: McGraw-Hill, 1965.
54. J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Redwood City, CA: Addison-Wesley, 1991.
55. M. A. Arbib, *Brains, Machines, and Mathematics*, 2nd ed., New York: Springer-Verlag, 1987.
56. J. J . Shynk, "Performance Surfaces of a Single-Layer Perceptron," *IEEE Transactions on Neural Networks*, vol. 1, 1990, pp. 268–74.
57. D. E. Rumelhart and J. L. McClelland, eds., *Parallel Distributed Processing*, vol. 1, Cambridge, MA: M.I.T. Press, 1986.
58. S. C. Douglas, "A Family of Normalized LMS Algorithms," *IEEE Signal Processing Letters*, vol. 1, 1994, pp. 49–51.
59. D. O. Hebb, *The Organization of Behavior*, New York, Wiley, 1949. Introduction and chapter 4, "The First Stage of Perception: Growth of the Assembly," pp. xi–xix, 60–78. Reprinted in 1988, Anderson and Rosenfeld [4], pp. 484–507.
60. R. J. MacGregor and E. R. Lewis, *Neural Modeling: Electrical Signal Processing in the Nervous System*, New York: Plenum Press, 1977.
61. G. S. Stent, "A Physiological Mechanism for Hebb's Postulate of Learning," in *Proceedings of the National Academy of Sciences of the U.S.A.*, vol. 70, 1973, pp. 997–1001.
62. J. P. Changeux and A. Danchin, "Selective Stabilization of Developing Synapses as a Mechanism for the Specification of Neural Networks," *Nature* (London), vol. 264, 1976, pp. 705–12.
93
63. T. H. Brown, E. W. Kairiss, and C. L. Keenan, "Hebbian Synapses: Biophysical Mechanisms and Algorithms," *Annual Review of Neuroscience*, vol. 13, 1990, pp. 475–511.
64. G. Palm, *Neural Assemblies: An Alternative Approach*, New York: Springer-

Verlag, 1982.
65. S. Amari, "Mathematical Theory of Neural Learning," *New Generation Computing*, vol. 8, 1991, pp. 281–94.
66. E. Oja, "A Simplified Neuron Model as a Principal Component Analyzer," *Journal of Mathematical Biology*, vol. 15, 1982, pp. 267–73.
67. K. Yale, "Preparing the Right Data Diet for Training Neural Networks," *IEEE Spectrum*, vol. 34, March 1997, pp. 64–6.
68. H. Demuth and M. Beale, *Neural Network Toolbox—For Use with MATLAB*, version 3, User's Guide, The Mathworks, Inc., Natick, MA: 1998.
69. G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System," *Neural Networks*, vol. 4, 1991, pp. 759–71.
70. G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps," *IEEE Transactions on Neural Networks*, vol. 3, 1992, pp. 698–713.
71. F. M. Ham, G. Han, and L. V. Fausett, "Fuzzy LAPART: A Neural Architecture for Supervised Learning through Inferencing for Stable Category Recognition," *Journal of Artificial Neural Networks*, vol. 2, 1995, pp. 241–64.
72. F. M. Ham and I. Kostanic, "Partial Least-Squares: Theoretical Issues and Engineering Applications in Signal Processing," *Mathematical Problems in Engineering*, vol. 2, 1995, pp. 63–93.
73. D. M. Haaland and E. V. Thomas, "Partial Least-Squares Methods for Spectral Analysis. 1. Relation to Other Quantitative Calibration Methods and the Extraction of Qualitative Information," *Analytical Chemistry*, vol. 60, 1988, pp. 1193–1202.
74. P. Geladi and B. R. Kowalski, "Partial Least-Squares: A Tutorial," *Analytica Chimica Acta*, vol. 185, 1986, pp. 1–17.
75. K. R. Beebe and B. R. Kowalski, "An Introduction to Multivariate Calibration and Analysis," *Analytical Chemistry*, vol. 59, 1987, pp. 1007A–17A.
76. H. Martens and T. Naes, *Multivariate Calibration*, New York: Wiley, 1989.
77. E. R. Malinowski, *Factor Analysis in Chemistry*, 2nd ed., New York: Wiley, 1991.
78. A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1989.
79. R. J. Mammone, X. Zhang, and R. P. Ramachandran, "Robust Speaker Recognition: A Feature-Based Approach," *IEEE Signal Processing Magazine*, vol. 13, 1996, pp. 58–71.
80. P. Bhandare, Y. Mendelson, R. Peura, G. Janatsch, J. D. Kruse-Jarres, R. Marbach, and H. M. Heise, "Multivariate Determination of Glucose in Whole Blood Using Partial Least-Squares and Artificial Neural Networks Based on Mid-infrared Spectroscopy," *Applied Spectroscopy*, vol. 47, 1993, pp. 1214–21.
81. C. S. Burrus, R. A. Gopinath, and H. Guo, *Introduction to Wavelets and Wavelet Transforms: A Primer*, Upper Saddle Creek, NJ: Prentice-Hall, 1998.
82. O. Rioul and M. Vetterli, "Wavelets and Signal Processing," *IEEE Signal Processing Magazine*, October 1991, pp. 14–38.
83. G. Strang and T. Nguyen, *Wavelets and Filter Banks*, Wellesley, MA: Wellesley-Cambridge Press, 1996.
84. I. Daubechies, *Ten Lectures on Wavelets*, Philadelphia, PA: Society for Industrial and Applied Mathematics (CBMS-61), 1992.
85. H. H. Szu, X. Y. Yang, B. A. Telfer, and Y. Sheng, "Optical Wedge-Filter on Wavelet Transform Domain for Scale-Invariant Signal Classification," in *Proceedings of the World Congress on Neural Networks*, Portland, OR, vol. 4,

1993, pp. 803–7.

86. H. Szu, Y. Zhang, M. Sun, and C. C. Li, "Neural Network Adaptive Digital Image Screen Halftoning (DISH) Based Wavelet Transform Preprocessing," in *Proceedings of the International Joint Conference on Neural Networks*, Nagoya, Japan, vol. 2, 1993, pp. 1215–18.
87. M. Misiti, Y. Misiti, G. Oppenheim, and J.-M. Poggi, *Wavelet Toolbox—For Use with MATLAB*, version 1, User's Guide, Natick, MA: The Mathworks, 1997.

95

# 第3章 映射网络

## 3.1 概述

本章将讲述几种映射神经网络结构和相关的学习规则。如1.4节所述，有几种用于分类神经网络的方法，因而神经网络的分类法并不一定是简单明了的。图1-10给出的分类方法是根据网络如何学习，即监督学习与非监督学习，对精选神经网络进行分类。这一章将介绍四类重要的映射网络。选择这组网络是由于它代表监督方式下训练的网络，并且有许多不同能力，例如，模式联想与分类、函数逼近和估计等。四类不同的映射网络是：(1) 联想记忆网络；(2) 利用反向传播算法训练的前馈多层感知器，具有反向传播学习算法的一些变体；(3) 对传网络；(4) 径向基函数网络。由图3-1可观察出这四类映射神经网络的共性。图3-1所示的映射函数$\Omega(\cdot)$可以是线性或非线性的。通常情况下是非线性的。映射$\Omega$：$\Re^{n\times 1}\rightarrow\Re^{m\times 1}$可提供存储和检索不同模式的方法，例如，在字符识别系统中。




图3-1 用于映射ANN的通用结构

对联想记忆网络来说，运行相对简单。在学习过程中，提交关键模式给网络（网络从“空白记录”开始），并且记忆转换这些模式为记忆（或存储）模式。在学习过程中网络突触权值得到调整。训练之后，回忆（或回溯）阶段涉及提交刺激（输入）给网络。这个输入可 96
以是不完整的或有噪声的。然而，即使损坏的输入，记忆网络也有能力恰当回忆“相关的”正确模式。甚至亚里士多德（Aristotle）意识到联想是人类记忆的显著特征。利用反向传播算法训练的前馈多层感知器在当今可能是最有名、最经常使用的神经网络。标准的反向传播算法是以最速下降方法（参看A.5.2节）为基础，并且在网络中，突触权值与网络输出的计算误差，即实际输出和期望输出之间的差异，成比例地更新。训练后的结果是从网络输入到输出的特定非线性映射。标准反向传播算法的变体能提高网络的收敛速度和它的性能。对传神经网络像统计上最优的自编程查找表一样运行。这些网络提供了输入与输出训练模式间的双向映射。径向基函数网络是另一种有力的监督训练网络，可用于模式分类和函数逼近。当它们的输入在输入空间的相当小的局部化区域时，在网络的隐藏层的基函数（即，非线性的）产生一个很大的非零响应。在许多情况下，径向基函数网络的训练比利用反向传播来训练的前馈多层感知器快得多。

## 3.2 联想记忆网络

在任何类型的神经网络中，存储和检索信息的能力是至关重要的。对于信息处理系统记忆和推断所存储的信息而言，记忆能力是至关重要的。存储的信息必须在网络的存储中恰当地分配地址，并且输出到外面的世界。即给定一个关键（key）输入（刺激），从（联想）记忆中检索适当的记忆模式，输出作为刺激的适当响应。对于可能不完整（有噪声）的输入关键或网络串扰作出响应，这是网络达到的最佳能力。

在神经生物系统中，记忆概念意味着神经变化，它由器官结构同环境的交互作用所引起 97

[1]。若不发生这个变化，记忆将不存在。同样，如果记忆是有用的，那么记忆必须是神经系统可访问的，以便能发生学习和能执行信息检索。对任何信息处理系统，无论是人工还是生物的，前面给出的描述是最基本的。通过一个学习过程，活动模式存储在记忆中。因此，记忆和学习具有深奥的联系。特定活动模式被学习后将存储在记忆中。此后当需要获得那个特定信息时能被检索（回忆）。有两类记忆：*长期记忆*和*短期记忆*。这两类记忆的区别取决于保持时间。

下面将研究有联想操作的信息处理系统的记忆动力学，联想是人类记忆的本质特征。大多数认知模型以某种形式运用联想[1-11]，我们称任何运用联想的记忆系统为*联想记忆*。许多神经元组织在一起使得需要存储在记忆中的活动模式在记忆中形成一个包含激励信息的较大的空间模式，在此意义下，联想记忆为分布式的。因此，记忆执行一个*分布式映射*（distributed mapping），将输入空间的活动模式变换到输出空间的另一个活动模式。通常，存储在记忆中的单个模式间存在相互作用。这种情况是必需的，否则记忆必须非常大以对所有单个模式之间提供相互完全隔离。因为相互作用，在模式检索过程中，可能发生记忆误差。有两类基本的联想记忆：*自联想记忆*和*异联想记忆*。在自联想记忆情况下，关键输入向量联想（或映射）到自身。然而，异联想记忆，关键输入向量联想（或映射）到任意记忆向量。在这种情况下，输入空间与输出空间维数可能不同（对于自联想记忆，输入和输出空间的维数相同）。

### 3.2.1　一般的线性分布式联想记忆

在一个一般线性分布式联想记忆的例子中，学习过程涉及提交一个关键输入模式（向量）给网络；然后记忆把这个向量变换成一个存储（或记忆）模式。在图3-2所示的线性神经网络结构能够作为一个联想记忆模型，使用简单线性组合器作为神经元。这个单层线性神经网络有一个输入（向量）

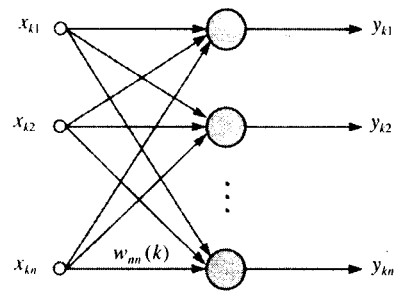



图3-2　单层线性神经网络联想记忆。假定输入和输出维数相同，为$n$

$$\boldsymbol{x}_k = [x_{k1}, x_{k2}, \cdots, x_{kn}]^T \tag{3-1}$$

叫做*关键输入模式*，并且一个输出（向量）

$$\boldsymbol{y}_k = [y_{k1}, y_{k2}, \cdots, y_{kn}]^T \tag{3-2}$$

叫做*记忆模式*。$\boldsymbol{x}_k$和$\boldsymbol{y}_k$的元素既可以是正值也可以是负值，这在神经生物学上是不一定合理的。对于一个特定的维数$n$，图3-2中的神经结构能够联想$h$个模式；然而$h \leqslant n$（$n$是网络的最大存储能力）。实际上，网络的工作能力是$h < n$。关键向量$\boldsymbol{x}_k$和$\boldsymbol{y}_k$记忆向量$\boldsymbol{y}_k$之间的线性联想映射可用矩阵形式表示为：

$$\boldsymbol{y}_k = \boldsymbol{W}(k)\,\boldsymbol{x}_k \tag{3-3}$$

其中权值矩阵$\boldsymbol{W}(k) \in \Re^{n\times n}$由输入/输出对$\{\boldsymbol{x}_k, \boldsymbol{y}_k\}$决定，$k = 1, 2, \cdots, h$。对每一输入/输出对，有相应的权值矩阵$\boldsymbol{W}(1)$, $\boldsymbol{W}(2)$, $\cdots$, $\boldsymbol{W}(h)$。由这组权值矩阵可构造记忆矩阵$\boldsymbol{M} \in \Re^{n\times n}$，$\boldsymbol{M}$描述每个输入/输出对或者是模式联想的整个集合的权值矩阵的*总和*。这可写为

$$\boldsymbol{M} = \sum_{k=1}^{h} \boldsymbol{W}(k) \tag{3-4}$$

这个记忆矩阵能定义为任何输入模式（关键模式）和相关输出模式（记忆模式）之间的整体连接。而且，记忆矩阵可认为是通过提交$h$个输入/输出模式给网络而获得的集体经验的表示。等式（3-4）用递归形式可表示为：

$$\boldsymbol{M}_k = \boldsymbol{M}_{k-1} + \boldsymbol{W}(k), \quad \text{for} \quad k = 1, 2, \cdots, h \tag{3-5}$$

其中$\boldsymbol{M}_0 = \boldsymbol{0}$。从式（3-5）得到的最终结果与式（3-4）所示的结果一致。当记忆矩阵从每个由第$k$个$\{\boldsymbol{x}_k, \boldsymbol{y}_k\}$联想产生的权值矩阵增量$\boldsymbol{W}(k)$“构建”时，当前权值矩阵$\boldsymbol{W}(k)$在与其他权值矩阵混合中失去独特性质。然而，关于当前联想的信息在与其他联想的突触混合也许没有完全失去。下一节将讲述对于给定的关键模式和相应的记忆模式的记忆矩阵的估计方法。

98 ~ 99

### 3.2.2 相关矩阵记忆

式（3-4）所示的记忆矩阵$\boldsymbol{M}$的估计可由关键模式和记忆模式对构成

$$\hat{\boldsymbol{M}} = \sum_{k=1}^{h} \boldsymbol{y}_k \boldsymbol{x}_k^T \tag{3-6}$$

从式（3-6）中可知，记忆矩阵$\boldsymbol{M}$的估计的基是$h$个外积矩阵$\boldsymbol{y}_k\boldsymbol{x}_k^T(k = 1, 2, \cdots, h)$的和。因此，每一外积矩阵是权值矩阵$\boldsymbol{W}(k)$的估计，它映射输出模式$\boldsymbol{y}_k$到输入模式$\boldsymbol{x}_k$。对于每一输入/输出对$\{\boldsymbol{x}_k, \boldsymbol{y}_k\}$，权值矩阵估计中的元素是$y_{ki}\,x_{kj}$。对于第$k$个联想，相关的权值矩阵元素$w_{ij}(k)$有一个作为网络输入（$x_{kj}$）的前突触节点$j$和在网络输出层的第$i$个神经元是后突触节点（$y_{ki}$）。式（3-6）中的学习规则实际上是局部化的学习过程，可视为Hebb学习（参看2.8.2节）的一种形式。权值矩阵估计中每一项（$y_{ki}x_{kj}$）是Hebb学习中的共生项，在Hebb学习中很典型。

式（3-6）所示的学习过程称为外积规则，因为记忆矩阵估计$\hat{\boldsymbol{M}}$由$h$个外积矩阵$\boldsymbol{y}_k\boldsymbol{x}_k^T(k = 1, 2, \cdots, h)$的和构成。因此，以这种方式设计的联想记忆称为相关矩阵记忆[12]。也可以将式（3-6）所示的相关学习过程以递归形式可表示为：

$$\hat{\boldsymbol{M}}_k = \hat{\boldsymbol{M}}_{k-1} + \boldsymbol{y}_k \boldsymbol{x}_k^T, \quad \text{for} \quad k = 1, 2, \cdots, h \tag{3-7}$$

在式（3-7）中当$k = 1$时，与式（3-5）表示的一般递归形式一样，$\hat{\boldsymbol{M}}_0 = \boldsymbol{0}$。式（3-7）中的最后得到的值$\hat{\boldsymbol{M}}_k\big|_{k=h} = \hat{\boldsymbol{M}}_h$与式（3-6）得到的结果$\hat{\boldsymbol{M}}$是一致的。

给定记忆矩阵$\hat{\boldsymbol{M}}$的估计，当关键模式引入网络时，希望寻址和回忆存储在联想记忆中的恰当记忆模式。给定$h$个模式联想，假设联想记忆的矩阵估计由式（3-6）所示的学习过程构成。任意选择关键模式$\boldsymbol{x}_q$，并且作为记忆刺激应用于网络，相应的响应是：

$$\boldsymbol{y} = \hat{\boldsymbol{M}}\boldsymbol{x}_q = \sum_{k=1}^{h} \boldsymbol{y}_k \boldsymbol{x}_k^T \boldsymbol{x}_q = \sum_{k=1}^{h} (\boldsymbol{x}_k^T \boldsymbol{x}_q)\boldsymbol{y}_k \tag{3-8}$$

等式（3-8）可改写为：

$$\boldsymbol{y} = (\boldsymbol{x}_q^T \boldsymbol{x}_q)\boldsymbol{y}_q + \sum_{\substack{k=1 \\ k \neq q}}^{h} (\boldsymbol{x}_k^T \boldsymbol{x}_q)\boldsymbol{y}_k \tag{3-9}$$

假设每一关键向量$\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_h$规范化到单位长度，即：

$$\boldsymbol{x}_k^T \boldsymbol{x}_k = 1 \quad \text{for} \quad k = 1, 2, \cdots, h \tag{3-10}$$

由欧几里得范数除以每个关键向量，可实现。即：

100

$$x_k \leftarrow \frac{x_k}{\| x_k \|_2} \tag{3-11}$$

式（3-9）可改写为：

$$y = y_q + \sum_{\substack{k=1 \\ k \neq q}}^{h} (x_k^T x_q) y_k = y_q + z_q \tag{3-12}$$

其中

$$z_q = \sum_{\substack{k=1 \\ k \neq q}}^{h} (x_k^T x_q) y_k \tag{3-13}$$

从式（3-12）中可看出，当联想关键模式传给网络时，如果联想记忆能完整恢复记忆模式，则$y = y_q$且$z_q = 0$。因此，式（3-12）的$y_q$是期望响应（信号），$z_q$视为噪声或串扰（crosstalk）。从式（3-13）中可看出，由于关键向量$x_q$（即，刺激）与存储在记忆中的其他关键向量相互作用产生串扰。如果各模式是统计上独立的，从中心极限定理[13]可导出：噪声向量$z_q$是具有高斯分布元素的随机向量。这是一个从加性噪声中分离出有用信号的典型问题[13]。对于存储在联想记忆中的记忆模式，“噪声”水平将确定重构的精确程度。

从式（3-13）中显然可看出，如果各种关键向量构成正交集合（实际上是标准正交集合，因为已假定关键向量规范化到单位长度）。那么串扰是0，并且记忆模式完全恢复。给定一线性独立而不一定正交的关键向量集，该向量集合将在生成记忆矩阵$\hat{M}$之前完成格拉姆-施密特正交化[14]过程。即：给定关键向量集$\{x_1, x_2, \cdots, x_k\}$，将创建一个新的正交向量集$\{g_1, g_2, \cdots, g_k\}$，该向量集与原始集是线性一一对应的。执行格拉姆-施密特正交化首先通过假定$g_1 = x_1$，然后，剩下的向量$g_k$(k = 2, 3, …, $h$)可由下式产生

$$g_k = x_k - \sum_{i=1}^{k-1} \left( \frac{g_i^T x_k}{g_i^T g_i} \right) g_i \tag{3-14}$$

在格拉姆-施密特正交化之后，对于$\{g_k, y_k\}$对($k$ =1, 2, …, $h$)执行联想。如果关键向量构成正
101 交向量集，联想记忆的存储容量的上限是$n$（$n$是网络维数，或是输入空间维数）。通常，联想记忆的存储容量是$\rho(\hat{M}) \leqslant n$。换言之，存储极限取决于记忆矩阵的秩。

**例3.1**　要发展一个自联想记忆，该记忆用如下三个关键向量来训练

$$x_1 = \begin{bmatrix} -0.3333 \\ 0.7778 \\ 0.5329 \end{bmatrix} \quad x_2 = \begin{bmatrix} 0.4444 \\ -0.5556 \\ 0.7027 \end{bmatrix} \quad x_3 = \begin{bmatrix} 0.4969 \\ 0.6667 \\ 0.5556 \end{bmatrix} \tag{3-15}$$

每个向量均有单位长度。正如上面讨论的，如果训练向量是相互正交的，串扰将不存在，存储在联想记忆中的每个记忆模式将完整地再现。所以，计算这些向量之间的角度将是有益的，并且因此决定记忆回忆的程度如何。式（3-15）中向量之间的角度可计算为：

$$\theta_{12} = \cos^{-1} \frac{x_1 x_2^T}{\| x_1 \|_2 \| x_2 \|_2} = 101.9^\circ \tag{3-16}$$

$$\theta_{13} = \cos^{-1} \frac{x_1 x_3^T}{\| x_1 \|_2 \| x_3 \|_2} = 49.5^\circ \tag{3-17}$$

$$\theta_{23} = \cos^{-1} \frac{x_2 x_3^T}{\| x_2 \|_2 \| x_3 \|_2} = 76.1^\circ \tag{3-18}$$

从式（3-16）～式（3-18），我们发现这三个向量远不是相互正交的。自联想网络的记忆矩阵根据式（3-6）得到如下：

$$\hat{\boldsymbol{M}} = \boldsymbol{x}_1\boldsymbol{x}_1^T + \boldsymbol{x}_2\boldsymbol{x}_2^T + \boldsymbol{x}_3\boldsymbol{x}_3^T = \begin{bmatrix} 0.5555 & -0.1749 & 0.4107 \\ -0.1749 & 1.3582 & 0.3945 \\ 0.4107 & 0.3945 & 1.0865 \end{bmatrix} \tag{3-19}$$

使用式（3-19）中的记忆矩阵，通过提交式（3-15）中的关键向量得到输入关键模式的估计。其结果为

$$\hat{\boldsymbol{x}}_1 = \hat{\boldsymbol{M}}\boldsymbol{x}_1 = \begin{bmatrix} -0.1023 \\ 1.3249 \\ 0.7489 \end{bmatrix} \quad \hat{\boldsymbol{x}}_2 = \hat{\boldsymbol{M}}\boldsymbol{x}_2 = \begin{bmatrix} 0.6326 \\ -0.5551 \\ 0.7268 \end{bmatrix} \quad \hat{\boldsymbol{x}}_3 = \hat{\boldsymbol{M}}\boldsymbol{x}_3 = \begin{bmatrix} 0.3876 \\ 1.0378 \\ 1.0707 \end{bmatrix} \tag{3-20}$$

把这些结果与式（3-15）的原始关键向量作比较，发现估计并不是完全的副本。这是可预料的，由于向量的非正交性，从式（3-16）～式（3-18）向量间的夹角所证明，也就是说，并不是所有的夹角都是90°。然而，如果我们计算式（3-15）中每个规范化关键输入和它的响应（输出）向量之间的欧几里得距离，我们能够得到响应和原始关键向量多么“靠近”的度量。

让我们首先确定响应向量 $\hat{\boldsymbol{x}}_1$ 和每个关键向量之间的欧几里得距离。给出如下结果： 102

$$\delta_{11} = \|\boldsymbol{x}_1 - \hat{\boldsymbol{x}}_1\|_2 = 0.6319 \quad \delta_{21} = \|\boldsymbol{x}_2 - \hat{\boldsymbol{x}}_1\|_2 = 1.9589 \quad \delta_{31} = \|\boldsymbol{x}_3 - \hat{\boldsymbol{x}}_1\|_2 = 0.9108 \tag{3-21}$$

从式（3-21），因为 $\delta_{11}$ 是最小的，很明显，响应向量 $\hat{\boldsymbol{x}}_1$ 和比 $\boldsymbol{x}_2$ 或 $\boldsymbol{x}_3$ 更靠近 $\boldsymbol{x}_1$。用响应向量 $\hat{\boldsymbol{x}}_2$ 做相同运算，得到

$$\delta_{12} = \|\boldsymbol{x}_1 - \hat{\boldsymbol{x}}_2\|_2 = 1.6575 \quad \delta_{22} = \|\boldsymbol{x}_2 - \hat{\boldsymbol{x}}_2\|_2 = 0.1898 \quad \delta_{32} = \|\boldsymbol{x}_3 - \hat{\boldsymbol{x}}_2\|_2 = 1.2412 \tag{3-22}$$

正如期望的一样，我们发现响应向量 $\hat{\boldsymbol{x}}_2$ 最靠近关键向量 $\boldsymbol{x}_2$。对于响应向量 $\hat{\boldsymbol{x}}_3$，欧几里得距离计算如下：

$$\delta_{13} = \|\boldsymbol{x}_1 - \hat{\boldsymbol{x}}_3\|_2 = 0.9363 \quad \delta_{23} = \|\boldsymbol{x}_2 - \hat{\boldsymbol{x}}_3\|_2 = 1.6363 \quad \delta_{33} = \|\boldsymbol{x}_3 - \hat{\boldsymbol{x}}_3\|_2 = 0.6442 \tag{3-23}$$

并且如期望的一样，响应向量 $\hat{\boldsymbol{x}}_3$ 最靠近 $\boldsymbol{x}_3$。

我们试作一个不同的（单元长度）关键向量集合，如下：

$$\boldsymbol{x}_1 = \begin{bmatrix} 0.1309 \\ -0.9779 \\ -0.1629 \end{bmatrix} \quad \boldsymbol{x}_2 = \begin{bmatrix} -0.7548 \\ 0.0587 \\ -0.6533 \end{bmatrix} \quad \boldsymbol{x}_3 = \begin{bmatrix} -0.6354 \\ -0.2370 \\ 0.7349 \end{bmatrix} \tag{3-24}$$

这些向量之间的夹角为

$$\theta_{12} = \cos^{-1}\frac{\boldsymbol{x}_1\boldsymbol{x}_2^T}{\|\boldsymbol{x}_1\|_2\|\boldsymbol{x}_2\|_2} = 92.9° \tag{3-25}$$

$$\theta_{13} = \cos^{-1}\frac{\boldsymbol{x}_1\boldsymbol{x}_3^T}{\|\boldsymbol{x}_1\|_2\|\boldsymbol{x}_3\|_2} = 88.3° \tag{3-26}$$

$$\theta_{23} = \cos^{-1}\frac{\boldsymbol{x}_2\boldsymbol{x}_3^T}{\|\boldsymbol{x}_2\|_2\|\boldsymbol{x}_3\|_2} = 90.8° \tag{3-27}$$

从式（3-25）～式（3-27），我们将期望自联想回忆过程比以前好，因为式（3-24）中的单元长度向量比式（3-15）中的那些向量更“接近于”相互正交。在图3-3中表明了这一点。给定记忆矩阵为：

$$\hat{\boldsymbol{M}} = \boldsymbol{x}_1\boldsymbol{x}_1^T + \boldsymbol{x}_2\boldsymbol{x}_2^T + \boldsymbol{x}_3\boldsymbol{x}_3^T = \begin{bmatrix} 0.9906 & -0.0217 & 0.0048 \\ -0.0217 & 1.0159 & -0.0532 \\ 0.0048 & -0.0532 & 0.9934 \end{bmatrix} \tag{3-28}$$

使用式（3-28），对式（3-24）中关键向量的响应为：

$$\hat{\boldsymbol{x}}_1 = \hat{\boldsymbol{M}}\boldsymbol{x}_1 = \begin{bmatrix} 0.1501 \\ -0.9876 \\ -0.1092 \end{bmatrix} \quad \hat{\boldsymbol{x}}_2 = \hat{\boldsymbol{M}}\boldsymbol{x}_2 = \begin{bmatrix} -0.7521 \\ 0.1108 \\ -0.6558 \end{bmatrix} \quad \hat{\boldsymbol{x}}_3 = \hat{\boldsymbol{M}}\boldsymbol{x}_3 = \begin{bmatrix} -0.6207 \\ -0.2661 \\ 0.7396 \end{bmatrix} \tag{3-29}$$

103 来自每个关键向量的响应向量 $\hat{\boldsymbol{x}}_1$ 的欧几里得距离为：

$$\delta_{11} = \|\boldsymbol{x}_1 - \hat{\boldsymbol{x}}_1\|_2 = 0.0579 \quad \delta_{21} = \|\boldsymbol{x}_2 - \hat{\boldsymbol{x}}_1\|_2 = 1.4865 \quad \delta_{31} = \|\boldsymbol{x}_3 - \hat{\boldsymbol{x}}_1\|_2 = 1.3758 \tag{3-30}$$

由式（3-30），响应向量 $\hat{\boldsymbol{x}}_1$ 与 $\boldsymbol{x}_2$ 或 $\boldsymbol{x}_3$ 相比更靠近 $\boldsymbol{x}_1$。将这些结果与由式（3-21）给出的关键向量先前集合的类似结果进行比较，我们发现关键向量的第二个集合由于它们更加正交而产生更好的结果。对于响应向量 $\hat{\boldsymbol{x}}_2$，结果为：

$$\delta_{12} = \|\boldsymbol{x}_1 - \hat{\boldsymbol{x}}_2\|_2 = 1.4859 \quad \delta_{22} = \|\boldsymbol{x}_2 - \hat{\boldsymbol{x}}_2\|_2 = 0.0522 \quad \delta_{32} = \|\boldsymbol{x}_3 - \hat{\boldsymbol{x}}_2\|_2 = 1.4382 \tag{3-31}$$

如期望的一样，$\hat{\boldsymbol{x}}_2$ 最靠近 $\boldsymbol{x}_2$。最后，对于响应向量 $\hat{\boldsymbol{x}}_3$，结果是：

$$\delta_{13} = \|\boldsymbol{x}_1 - \hat{\boldsymbol{x}}_3\|_2 = 1.3734 \quad \delta_{23} = \|\boldsymbol{x}_2 - \hat{\boldsymbol{x}}_3\|_2 = 1.4365 \quad \delta_{33} = \|\boldsymbol{x}_3 - \hat{\boldsymbol{x}}_3\|_2 = 0.0329 \tag{3-32}$$

$\hat{\boldsymbol{x}}_3$ 最接近 $\boldsymbol{x}_3$。

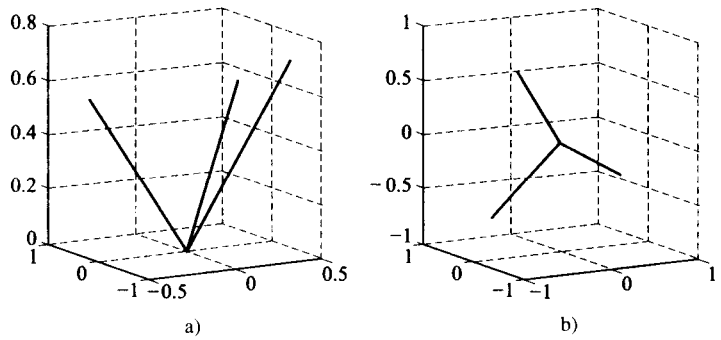



图3-3　a)式（3-15）所示的关键向量，向量之间的夹角由式（3-16）～式（3-18）给出；b)式（3-24）所示的关键向量，向量之间的夹角由式（3-25）～式（3-27）给出

### 3.2.3　相关矩阵记忆的误差修正方法

虽然相关矩阵记忆是相当简单的设计，对于联想记忆来说这种方法的主要弊端是在回忆过程中将发生大量的错误。在最简单的形式中，给定关键输入刺激，相关矩阵记忆没有预备修正在回忆记忆模式过程中可能产生的误差。网络不能修正误差的根本原因在于缺乏从输出到输入的反馈。对于由联想 $\boldsymbol{x}_k \rightarrow \boldsymbol{y}_k$，$k = 1, 2, \cdots, h$，根据式（3-6）构成的记忆矩阵 $\hat{\boldsymbol{M}}$，由刺
104 激 $\boldsymbol{x}_q$ 产生的实际输出响应 $\boldsymbol{y}$ 并不是足够“接近”（欧几里得意义下）完全联想记忆的期望（或真实）响应 $\boldsymbol{y}_q$。

我们想把误差修正机制融入递归公式，以强制记忆的设计能达到完全联想[15, 16]。因此，这里的主要目的是在最优意义下，使联想记忆重构记忆模式，从而提高记忆响应 $\boldsymbol{y}_k$ 的质量。假设 $\hat{\boldsymbol{M}}(\tau)$ 表示学习迭代 $\tau$ 时的记忆矩阵，随机选择关键向量模式 $\boldsymbol{x}_k$ 为 $\tau$ 次迭代应用于记忆，使用

结果实际响应 $\hat{M}(\tau)x_k$ 构成误差向量

$$e_k(\tau) = y_k - \hat{M}(\tau)x_k \tag{3-33}$$

其中$y_k$是与关键输入模式$x_k$相对应的期望活动模式。希望利用该误差项以某种方式计算正在构建的记忆矩阵在第$\tau$次迭代的调整，降低重构的误差。我们将采用最速下降方法（参看A.5.2节）发展离散时间学习规则，它是式（3-6）中递归的一种变形。这个基于最速下降的离散学习规则的形式为：

$$\hat{M}(\tau+1) = \hat{M}(t) - \mu\nabla_{\hat{M}}\mathrm{E}\ (\hat{M}) \tag{3-34}$$

其中$\mathrm{E}\ (\hat{M})$是能量（或李雅普诺夫）函数，即：

$$\mathrm{E}\ (\hat{M}) = \frac{1}{2}\| e_k \|_2^2 \tag{3-35}$$

由式（3-33），$e_k = y_k - \hat{M}x_k$，在式（3-34）中$\mu > 0$是学习率参数。计算式（3-35）关于记忆矩阵$\hat{M}$的梯度：

$$\nabla_{\hat{M}}\mathrm{E}\ (\hat{M}) = -y_k x_k^T + \hat{M}x_k x_k^T \tag{3-36}$$

其中使用了A.3.4.2节的结果（即，标量关于矩阵的微分）。将式（3-36）的结果代入式（3-34）得

$$\hat{M}(\tau+1) = \hat{M}(\tau) + \mu[y_k - \hat{M}(\tau)x_k]x_k^T \tag{3-37}$$

注意到式（3-37）中方括号内是式（3-33）中定义的误差向量$e_k$。因此，当学习记忆矩阵$\hat{M}$时，学习策略有一个误差修正机制嵌入算法以修正误差。将式（3-37）改写为：

$$\hat{M}(\tau+1) = \hat{M}(\tau) + \mu y_k x_k^T - \mu\hat{M}(\tau)x_k x_k^T \tag{3-38}$$

将式（3-38）与式（3-7）的原始递归相比较表明：附加项 $\hat{M}(\tau)x_k x_k^T$ 负责记忆矩阵的修正。式（3-37）中基于误差修正的监督学习算法反复应用于$h$个联想的每一个，即

$$x_k \rightarrow y_k \tag{3-39}$$

其中$k = 1, 2, \cdots, h$是任意选择的。 105

注意在选择学习率参数$\mu$时，必须确保在学习过程中反馈的稳定性。可以利用固定的学习率参数，或关于时间可调的学习率参数（参看2.5.1节）。对于每个联想，对式（3-37）中记忆矩阵的迭代调整继续到式（3-33）中的误差向量$e_k(\tau)$变得相当小。换言之，对第$k$个联想$x_k \rightarrow y_k$的学习当 $\hat{M}(\tau)x_k$ “靠近”期望响应$y_k$时停止，即 $y_k - \hat{M}(\tau)x_k \cong \mathbf{0}$。这将导致式（3-35）中的性能标准最小化，并且从优化意义下允许联想记忆重构记忆模式。为了初始化，运用式（3-37）中的误差修正机制，根据输入/输出对$\{x_k, y_k\}$学习记忆矩阵$\hat{M}$，我们置 $\hat{M}(0) = \mathbf{0}$。将这些结果与2.5.1节的最小均方（LMS）算法相比较，可发现式（3-37）是LMS算法形式，或delta规则。

## 3.3 反向传播学习算法

现在我们考虑前馈多层感知器（MLP）中的监督学习。特别地，我们想研究用于训练MLP的反向传播算法或广义的delta规则。当今在神经网络学习过程中运用最广泛的是反向传播，反向传播在1974[17]由Werbos最先研究。然而随后的许多年这项工作依旧不为人知[18, 19]。这个方法被重新发现了很多次，在1982年由Parker[20]（也可参考[21, 22]），1985年由

LeCun[23]和Rumelhart et al.在1986[24, 25]先后发现了该方法。也许是Rumelhart等人对反向传播的描述使该算法在科学和工程领域流行。运用反向传播算法来训练MLP导致非线性映射或联想任务。因此，给出两组数据，即输入/输出对，MLP能使突触权值用反向传播算法来调整，以产生特定非线性映射（参考2.7节）。经过训练过程之后，具有固定权值的MLP能提供联想任务用于分类、模式辨认、诊断等。在MLP的训练阶段，突触权值被调整以使MLP的实际输出与期望输出之间的差异最小，该差异是对所有输入模式（或学习例子）求平均而得。

### 3.3.1 前馈多层感知器的基本反向传播算法

本节我们将阐述标准反向传播学习算法的推导。为了简化，我们将导出具有三层权值
106 （即一个输出层和两个隐藏层）的多层感知器神经网络（MLP NN）的学习规则，因为三层权值是运用最频繁的MLP NN结构。这类神经网络的例子如图3-4所示。将推导扩展至网络具有多于两个隐藏层的一般情况是很直接的。

对MLP NN的训练，标准反向传播算法是建立在最速下降梯度基础上的，应用于表示瞬时误差的能量函数的最小化。

换言之，我们希望最小化的函数定义为

$$E_q = \frac{1}{2}(\boldsymbol{d}_q - \boldsymbol{x}_{\text{out}}^{(3)})^T(\boldsymbol{d}_q - \boldsymbol{x}_{\text{out}}^{(3)}) = \frac{1}{2}\sum_{h=1}^{n_3}(d_{qh} - x_{\text{out},h}^{(3)})^2 \tag{3-40}$$

其中$\boldsymbol{d}_q$代表第$q$个输入模式的期望网络输出，$\boldsymbol{x}_{out}^{(3)} = \boldsymbol{y}_q$是图3-4所示的MLP网络的实际输出。通常由式（3-40）最小化导出的权值更新方法称为在线方法，强调具有最小记忆存储需求。

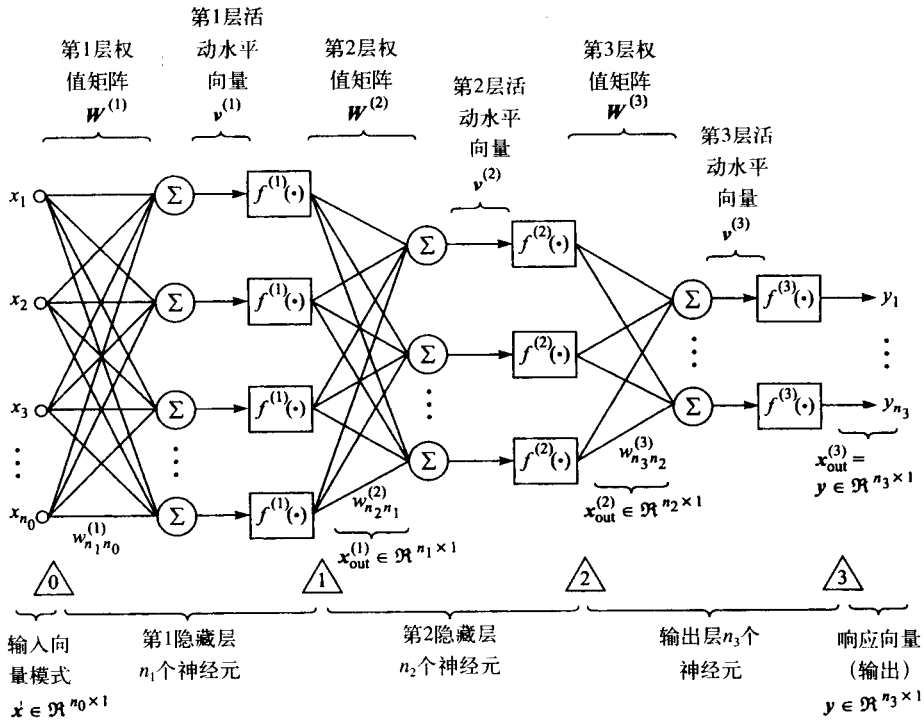



图3-4　一个三层前馈MLP NN结构的例子

运用最速下降梯度方法，对于网络任意层的网络权值学习规则可表示为：

$$\Delta w_{ji}^{(s)} = -\mu^{(s)} \frac{\partial E_q}{\partial w_{ji}^{(s)}} \tag{3-41}$$

107

其中$s = 1, 2, 3$标出了适当的网络层，$\mu^{(s)} > 0$是对应的学习率参数。基于很快就会明了的原因，对MLP NN隐藏层和输出层权值，分别导出学习规则。首先考虑网络的输出层。输出层权值可以根据下式更新

$$\Delta w_{ji}^{(3)} = -\mu^{(3)} \frac{\partial E_q}{\partial w_{ji}^{(3)}} \tag{3-42}$$

运用偏导数的链式法则，式（3-42）可改写为：

$$\Delta w_{ji}^{(3)} = -\mu^{(3)} \frac{\partial E_q}{\partial v_j^{(3)}} \frac{\partial v_j^{(3)}}{\partial w_{ji}^{(3)}} \tag{3-43}$$

式（3-43）中的分离项可求值为：

$$\frac{\partial v_j^{(3)}}{\partial w_{ji}^{(3)}} = \frac{\partial}{\partial w_{ji}^{(3)}} \left( \sum_{h=1}^{n_2} w_{jh}^{(3)} x_{\text{out},h}^{(2)} \right) = x_{\text{out},i}^{(2)} \tag{3-44}$$

和

$$\frac{\partial E_q}{\partial v_j^{(3)}} = \frac{\partial}{\partial v_j^{(3)}} \left\{ \frac{1}{2} \sum_{h=1}^{n_3} \left[ d_{qh} - f(v_h^{(3)}) \right]^2 \right\} = -\left[ d_{qj} - f(v_j^{(3)}) \right] g(v_j^{(3)}) \tag{3-45}$$

或

$$\frac{\partial E_q}{\partial v_j^{(3)}} = -(d_{qj} - x_{\text{out},j}^{(3)}) g(v_j^{(3)}) \hat{=} -\delta_j^{(3)} \tag{3-46}$$

其中$g(\cdot)$表示非线性激活函数$f(\cdot)$的一阶导数。式（3-46）中定义的项通常称为*局部误差*或*delta*。

合并式（3-43）、式（3-44）和式（3-46），可得到网络输出层权值的学习规则方程

$$\Delta w_{ji}^{(3)} = \mu^{(3)} \delta_j^{(3)} x_{\text{out},i}^{(2)} \tag{3-47}$$

或

$$w_{ji}^{(3)}(k+1) = w_{ji}^{(3)}(k) + \mu^{(3)} \delta_j^{(3)} x_{\text{out},i}^{(2)} \tag{3-48}$$

在网络隐藏层权值的更新方程可由同样方法导出。运用最速下降梯度方法，我们有

$$\Delta w_{ji}^{(2)} = -\mu^{(2)} \frac{\partial E_q}{\partial w_{ji}^{(2)}} = -\mu^{(2)} \frac{\partial E_q}{\partial v_j^{(2)}} \frac{\partial v_j^{(2)}}{\partial w_{ji}^{(2)}} \tag{3-49}$$

108

式（3-49）右边的二阶偏导数可求值为：

$$\frac{\partial v_j^{(2)}}{\partial w_{ji}^{(2)}} = \frac{\partial}{\partial w_{ji}^{(2)}} \left( \sum_{h=1}^{n_1} w_{jh}^{(2)} x_{\text{out},h}^{(1)} \right) = x_{\text{out},i}^{(1)} \tag{3-50}$$

由于$v_j^{(2)}$的改变通过网络输出层传播进而影响整个网络输出，因此，式（3-49）的一阶偏导数计算更复杂。我们能够通过表达该量为一个已知的量和更容易求值的其他项的函数来得到这个导数。为了继续进行，我们能够写：

$$\frac{\partial E_q}{\partial v_j^{(2)}} = \frac{\partial}{\partial x_{\text{out},j}^{(2)}} \left\{ \frac{1}{2} \sum_{h=1}^{n_3} \left[ d_{qh} - f\left( \sum_{p=1}^{n_2} w_{hp}^{(3)} x_{\text{out},p}^{(2)} \right) \right]^2 \right\} \frac{\partial x_{\text{out},j}^{(2)}}{\partial v_j^{(2)}} \tag{3-51}$$

或

$$\frac{\partial E_q}{\partial v_j^{(2)}} = -\left[\sum_{h=1}^{n_3}(d_{qh} - x_{\text{out},h}^{(3)})g(v_h^{(3)})w_{hj}^{(3)}\right]g(v_j^{(2)})$$
$$= -\left(\sum_{h=1}^{n_3}\delta_h^{(3)}w_{hj}^{(3)}\right)g(v_j^{(2)}) \triangleq -\delta_j^{(2)} \tag{3-52}$$

联合式（3-49）、式（3-50）和式（3-52）得到：

$$\Delta w_{ji}^{(2)} = \mu^{(2)}\delta_j^{(2)}x_{\text{out},i}^{(2)} \tag{3-53}$$

或

$$w_{ji}^{(2)}(k+1) = w_{ji}^{(2)}(k) + \mu^{(2)}\delta_j^{(2)}x_{\text{out},i}^{(2)} \tag{3-54}$$

比较式（3-48）与式（3-54），我们发现输出层和隐藏层的权值更新方程形式是一样的，唯一的不同在于如何计算局部误差。对于输出层，局部误差与实际网络输出同期望输出之间的差异成比例。延伸同样的观点到隐藏层的“输出”，隐藏层中一个神经元的局部误差可视为与该特定神经元的实际输出同期望输出之间的差异成比例。当然，在训练过程中，隐藏层神经元的期望输出是未知的，因此，局部误差需要根据所有相连神经元的误差信号来递推估计。等式（3-54）可推广到有任意多隐藏层的MLP NN。对于任意多隐藏层的网络可表示为：

$$w_{ji}^{(s)}(k+1) = w_{ji}^{(s)}(k) + \mu^{(s)}\delta_j^{(s)}x_{\text{out},i}^{(s)} \tag{3-55}$$

其中对于输出层：

109

$$\delta_j^{(s)} = (d_{qh} - x_{\text{out},j}^{(s)})g(v_j^{(s)}) \tag{3-56}$$

对于隐藏层：

$$\delta_j^{(s)} = \left(\sum_{h=1}^{n_{s+1}}\delta_h^{(s+1)}w_{hj}^{(s+1)}\right)g(v_j^{(s)}) \tag{3-57}$$

*标准反向传播算法的小结*

根据下面算法来完成运用标准反向传播算法进行MLP NN的训练。

**标准反向传播算法**

**步骤1**　初始化网络突触权值为很小的随机值。

**步骤2**　从训练输入/输出对的集合中，提交一个输入模式并计算网络响应。

**步骤3**　比较期望的网络响应与实际的网络输出，并且通过式（3-56）和式（3-57），计算所有局部误差。

**步骤4**　按照式（3-55）更新网络权值。

**步骤5**　通过步骤2到步骤4，直到网络对于所有训练模式产生适当的响应到达一个预先确定的精确度水平。　□

从上面的算法可看出，典型反向传播算法能用来完成两个独立任务：第一个是从输出层节点到隐藏层节点的误差的反向传播；第二个是运用LMS算法更新每一层的权值。

### 3.3.2　使用标准反向传播中的一些实际问题

到目前为止，对于MLP NN的训练而言，标准反向传播及其衍生是应用最广泛的学习算法。在这节将讨论一些涉及有效应用的实际问题。

*突触权值的初始化*

最初MLP NN的权值设置是小的随机值。它们不得不足够小，使得网络训练不从与一些饱和节点相对应的误差空间上的一个点开始。当网络运行在饱和状态时，对于学习收敛需要许多次迭代。用于权值初始化的一个普遍使用的启发算法是设置权值为 $-0.5/fan_in$到$0.5/fan_in$区间均匀分布的随机数，在这里$fan_in$表示权值为馈入的层中的神经元的总数量[26]。对于有一个隐藏层的MLP NN情况，在Nguyen and Widrow[27]中建议使用另外一种方法。作者说明该方法能够显著地提高网络训练速度。Nguyen和Widrow的MLP NN初始化能够用如下算法来小结。

110

**Nguyen和Widrow的初始化算法**

定义：

$n_0$=输入层分量的个数

$n_1$=隐藏层神经元的个数

$\gamma$=缩放因子

**步骤1** 按照如下式子计算缩放因子

$$\gamma = 0.7\sqrt[n_0]{n_1} \tag{3-58}$$

**步骤2** 初始化任一层的权值$w_{ij}$为在$-0.5$到$0.5$之间的随机值。

**步骤3** 按照下式重新初始化权值：

$$w_{ij} = \gamma \frac{w_{ij}}{\sqrt{\sum_{i=1}^{n_1} w_{ij}^2}} \tag{3-59}$$

**步骤4** 对于隐藏层第$i$个神经元，设置偏置为一个在$-w_{ij}$到$w_{ij}$之间的随机值。 □

*网络设置和网络的泛化能力*

MLP NN的设置由隐藏层的数量、每个隐藏层神经元的数量以及用于神经元的激活函数的类型决定。已经证实网络的性能并不十分依靠激活函数的类型（只要它是非线性的），隐藏层数量和每个隐藏层单元的数量的选择是关键。

Hornik et al.[28]建立了只有一个隐藏层，并且有足够数量神经元的MLP NN，充当一个非线性映射的通用逼近器。在实际中，决定足够数量的神经元必须达到逼近精度的期望程度是十分困难的。通常，隐藏层单元的数目是由试错法决定的。此外，如果网络仅仅有一个隐藏层，神经元看起来彼此之间“相互作用”[29]。在这种情况下，提高映射在一个点的逼近而在其他点不恶化它的逼近程度是十分困难的。基于以上原因，MLP NN通常设计为带有两个隐藏层。

典型地，为了使用MLP NN解决实际问题，需要训练一个相对大的神经网络结构。当提交属于训练集合的输入模式时，隐藏层有大量单元保证有良好的网络性能。然而，一个“过设计”结构将趋向于“过拟合”训练数据[30-32]，导致网络泛化属性的丢失。为了弄清楚这点，考虑如下例子。

**例3.2** 训练一个MLP NN在区间[0, 4]内逼近非线性函数

$$y = e^{-x}\sin(3x) \tag{3-60}$$

111

对于带有50个神经元组成的一个隐藏层的神经网络来说，这是一个相当简单的问题。对于这个例子，区间[0, 4]用21个点抽样，彼此之间间隔为0.2。使用MATLAB中的过程`trainlm`实现网

络训练，激活函数使用双曲正切非线性函数，且在整个数据集合上目标平方误差为0.01。网络仅仅在5个回合（epoch）后收敛，图3-5a表明了对于训练数据集合的期望输出与实际网络输出的一致程度。为了测试网络的泛化能力，同一间隔用401个点抽样，它们彼此之间间隔0.01，在图3-5b中给出了网络响应。如图3-5b，网络响应并没有与我们试图逼近的函数表现出良好的一致。这是由于训练数据过适应。在这种情况下，一个带有较小数量神经元的网络将以更好的方式实现逼近。

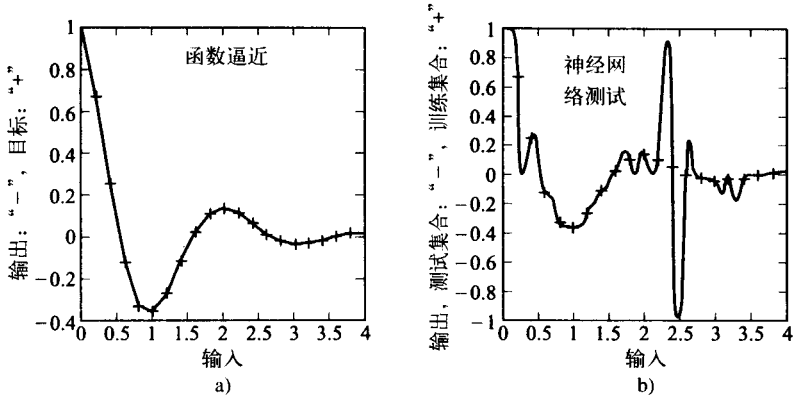



图3-5　数据过适应图例a)训练输入的网络响应；b)测试数据集合的网络响应

独立检验

仅以训练数据为基础来评估神经网络泛化特性绝不是一个好主意。运用训练数据来评估网络的最终表现性能可能导致过适应（overfitting）。运用统计上称为独立检验的标准方法可避免过适应（参看9.4节）。该方法涉及将可用数据划分为训练集和测试集。首先，整个数据集通常被随机化。接着训练数据分成两部分；第一部分用来更新网络权值；另一部分用来评估（检验）训练性能（例如，用来决定什么时候停止训练）。然后，测试数据用来评估网络具有怎么样的推广性。

收敛速度

从标准反向传播算法的推导可看出，它是2.5.1节陈述的LMS算法的推广。相反，用于训练单层感知器的LMS算法可视为标准反向传播算法的特例。在2.5.1节已说明LMS算法的收敛
112 性（尤其速度和稳定性）主要取决于学习率参数的范围。为了确保网络收敛和避免训练过程的振荡，学习率参数必须设为相对小的值。由于小的学习率参数限制网络权值的改变，从而显著影响算法的速度。而且，如果网络训练的起始点远离全局最小值，则一些神经元将饱和运行。当发生这种情况时，激活函数的导数很小。由于权值改变的范围直接取决于激活函数导数的范围，因此，网络可能陷入误差曲面的平坦区域，从而将经过许多次迭代才能收敛。对于中等复杂的现实问题网络训练需要数小时、甚至数天并不是罕见的。

反向传播算法的慢收敛鼓励了用于MLP NN训练的另一种（快速）算法的研究。快速算法的研究粗略地划分为两类：第一类由标准的反向算法各种启发式改进而来。虽然有用，且在许多情况下容易理解，启发式算法还是非常具体特殊的，它们的性能特征并不容易建立。第二种类型包含标准数值最优化技术的应用。这种类型的大多数算法以增加网络计算复杂性为代价，使网络收敛速度有重大的改进。这两种类型的一些代表和流行算法在后面的章节介绍。

除了反向传播学习的修改，输入数据的预处理和简化也能够导致性能改善和训练加速。也就是说，网络规模的减少将削减它的复杂性，并且大大地提高收敛速度。对于数据预处理

的一些方法在2.9节中提及。

### 3.3.3 具有动量更新的反向传播学习算法

具有动量更新的反向传播是对3.3.1节给出的标准算法的最流行的修正方法之一。该算法的思想是更新权值沿瞬时误差曲面当前梯度和训练前一步获得的权值更新之间的线性组合方向。也就是，权值根据下面公式更新

$$\Delta w_{ji}^{(s)}(k+1)=\mu^{(s)}\delta_j^{(s)}(k)x_{\text{out},i}^{(s)}(k)+\alpha\nabla w_{ji}^{(s)}(k-1) \tag{3-61}$$

或

$$w_{ji}^{(s)}(k+1)=w_{ji}^{(s)}(k)+\mu^{(s)}[\delta_j^{(s)}(k)x_{\text{out},i}^{(s)}(k)+\alpha\delta_j^{(s)}(k-1)x_{\text{out},i}^{(s)}(k-1)] \tag{3-62}$$

其中$\alpha$通常称为遗忘因子，通常在区间（0, 1）中选值。式（3-61）中的第二项称为动量项， 113
它通过在权值更新中引进稳定性来提高标准反向传播的收敛速度。直观上根据式（3-61），如果权值更新的方向与前一步相同，则变化率增加；反之，如果当前步的改变与前一步的方向不一致，则变化率降低。对一些利用标准反向传播很难处理的重要情况，这类学习显著提高了收敛性。首先，如果训练模式包含一些不确定性因素，例如噪声，则具有动量的更新通过阻止该权值更新方向的迅速变化来提供一类低通滤波；其次，这种行为使训练对出格点（outlier）或错误训练对的出现具有免疫能力。同时，如果网络在误差曲面的平坦区域运行，则动量的出现将提高权值变化率，且收敛速度将增加。考虑以下权值更新方程可方便解释这一点[26]

$$\Delta w_{ji}^{(s)}(k+1)=-\mu^{(s)}\frac{\partial E_q}{\partial w_{ji}^{(s)}}+\alpha\Delta w_{ji}^{(s)}(k-1) \tag{3-63}$$

如果网络在误差曲面的平坦区域运行，每一步的梯度值将不会显著变化，因此，式（3-63）可近似为

$$\begin{aligned}\Delta w_{ji}^{(s)}(k+1)&\approx-\mu^{(s)}\frac{\partial E_q}{\partial w_{ji}^{(s)}}-\alpha\mu^{(s)}\frac{\partial E_q}{\partial w_{ji}^{(s)}}-\alpha^2\mu^{(s)}\frac{\partial E_q}{\partial w_{ji}^{(s)}}+\cdots\\&=-\mu^{(s)}(1+\alpha+\alpha^2+\cdots)\frac{\partial E_q}{\partial w_{ji}^{(s)}}\\&\approx-\frac{\mu^{(s)}}{1-\alpha}\frac{\partial E_q}{\partial w_{ji}^{(s)}}\end{aligned} \tag{3-64}$$

因遗忘因子$\alpha$总是比单位值小，因此，具有动量的更新将有效学习率提高为

$$\mu_{\text{eff}}^{(s)}=\frac{\mu^{(s)}}{1-\alpha} \tag{3-65}$$

### 3.3.4 批量更新

标准反向传播算法假定权值由每一输入/输出训练对更新，而批量更新方法在执行更新之前累计几个训练模式（可能为整个回合）的权值修正。更新通常由每个输入/输出对的修正的均值构成。 114

批量更新方法的优点如下：

1. 运用几个（也许全部）训练对，对误差曲面给出比用于标准反向传播的瞬时值更好的估计。

2. 通过修正平均的处理，批量更新步骤提供某种固有的训练数据低通滤波。在训练数据被噪声损坏时这是有利的。

3. 批量算法适用于更复杂的优化过程，如共轭梯度方法或牛顿方法。

上面列出的优点是以下面这些因素为代价的：

1. 从存储需求的观点看批量更新需要更多。显然，我们需要额外的存储空间用于更新权值前的权值修正。在具有大量权值的网络中记忆储存需求变得非常严重。

2. 权值修正的平均额外增加了算法的计算复杂性。

3. 对于非批量训练模式（如标准反向传播），训练对中噪声的出现有助于网络训练逃离误差曲面的局部最小。批量更新的平滑效果使算法更趋向于收敛到一个局部最小。

通常，批量更新反向传播算法的性能非常依赖范例。对于在整个训练集执行平均的批量更新和标准反向传播之间的一个良好折中是在更新权值之前累积几个训练对的变化。这使学习算法在不显著增加陷入局部最小的可能性前提下产生误差曲面的更好估计。

### 3.3.5 搜索然后收敛方法

Darken et al.[33, 34]提出的用于加速反向传播学习的搜索然后收敛方法是相对简单的启发式策略（参看2.5.1节）。根据这一策略，MLP NN中的反向传播学习能划分为两个阶段：在第一阶段，网络离全局最小相当远，这个阶段称为*搜索阶段*。在搜索阶段学习率相当大，几乎是常数，以便网络能向（误差）性能曲面的最小值方向迅速下降。第二阶段称为*收敛阶段*，
115 当网络接近全局最小时开始。在收敛阶段每次迭代时，学习率减少，允许网络执行权值的精细调整。实际上，判断网络离全局最小值有多远是不可能的，并且学习率减少的策略必须提前采用，即：在网络训练开始之前。已经提出了两个通用的学习率下降策略[33-35]：

$$\mu(k)=\mu_0\frac{1}{1+k/k_0} \tag{3-66}$$

和

$$\mu(k)=\mu_0\frac{1+(c/\mu_0)(k/k_0)}{1+(c/\mu_0)(k/k_0)+k_0(k/k_0)^2} \tag{3-67}$$

其中$\mu_0>0$代表初始学习率参数，$c$和$k_0$是恰当选择的常量。典型地，$1<c/\mu_0<100$且$100<k_0<500$。在式（3-66）和式（3-67）中，当$k\ll k_0$时，学习率参数近似为常量$\mu_0$。这对应于搜索阶段。当$k\gg k_0$时，在式（3-66）中学习率以$1/k$的比例降低，在式（3-67）中以$1/k^2$的比例降低。已经证明，恰当地选择参数$c$和$k_0$，搜索然后收敛策略能显著提高反向传播算法的速度[35]。

### 3.3.6 可变学习率的批量更新

可变学习率的批量更新代表提高批量更新的反向传播算法收敛速度的一个简单启发式策略。该策略的思想是，如果前一步的学习已经降低了总误差函数，则增加学习率的范围。相反地，如果增加误差函数，则学习率需要降低。算法可小结如下[32]：

1. 如果在整个训练集合中误差函数已经降低，通过乘一个数$\eta>1$（典型地，$\eta=1.05$）来增加学习率。

2. 如果误差函数增加超过百分之$\xi$（典型地为百分之几），通过乘一个数$\chi<1$（典型地，$\chi=0.7$）来降低学习率。

3. 如果误差功能增加少于百分之$\xi$，学习率维持不变。

应用可变学习率进行批量更新在平滑而缓慢下降的误差函数情况下能够显著加快收敛。然而，该算法可能易于陷入误差曲面的一个局部最小。为了避免这点，学习率不允许降低到特定值$\mu_{\min}$之下。 116

### 3.3.7 反向传播算法的向量矩阵形式

这里我们介绍反向传播算法的向量矩阵形式。从实际实现角度，由于现代大多数的硬件和软件没有并行处理能力，算法的向量矩阵形式只有有限的应用。然而，矩阵向量形式给出了反向传播的可观了解，更重要地，它允许用于加快学习收敛的某些高级数值优化技术更直接的应用。

参考图3-4，指定能量函数为

$$E_q = \frac{1}{2}(\boldsymbol{d}_q - \boldsymbol{x}_{\text{out},q}^{(3)})^T(\boldsymbol{d}_q - \boldsymbol{x}_{\text{out},q}^{(3)}) \tag{3-68}$$

其中$\boldsymbol{d}_q$表示第$q$个输入模式的期望网络输出，$\boldsymbol{x}_{\text{out}}^{(3)} = \boldsymbol{y}_q$是图3-4所示MLP网络的实际输出。所以，式（3-68）表示误差曲面的一个瞬时估计。使用最速下降方法，突触权值更新方程能够写为：

$$w_{ij}^{(s)}(k+1) = w_{ij}^{(s)}(k) - \alpha^{(s)}\frac{\partial E_q}{\partial w_{ij}^{(s)}(k)} \tag{3-69}$$

其中$s$ = 1, 2, 3表示网络层数量，$\alpha^{(s)}$是与特定层相关的学习率参数。应用链式法则到式（3-68）的能量函数中偏导数，我们得到：

$$\frac{\partial E_q}{\partial w_{ij}^{(s)}} = \frac{\partial E_q}{\partial v_i^{(s)}} \cdot \frac{\partial v_i^{(s)}}{\partial w_{ij}^{(s)}} \tag{3-70}$$

其中为了简化起见离散时间的指标$k$被省略。

式（3-70）右边第二项能按下式求值：

$$\frac{\partial v_i^{(s)}}{\partial w_{ij}^{(s)}} = \frac{\partial}{\partial w_{ij}^{(s)}}\left(\sum_{h=1}^{n_{s-1}} w_{ih}^{(s)} x_{\text{out},h}^{(s-1)}\right) = x_{\text{out},j}^{(s-1)} \tag{3-71}$$

式（3-70）右边第一项通常作为一个*灵敏度*（sensitivity）项。实际上，它确定式（3-68）中由于第$s$层的权值改变引起的能量函数的变化。所以，我们可以定义

$$\delta_i^{(s)} \triangleq -\frac{\partial E_q}{\partial v_i^{(s)}} \tag{3-72}$$

并且使用式（3-70）和式（3-71）来重写式（3-69）为

$$w_{ij}^{(s)}(k+1) = w_{ij}^{(s)}(k) + \alpha^{(s)}\delta_i^{(s)} x_{\text{out},j}^{(s-1)} \tag{3-73}$$ 117

或以向量矩阵形式

$$\boldsymbol{W}^{(s)}(k+1) = \boldsymbol{W}^{(s)}(k) + \alpha^{(s)}\boldsymbol{D}^{(s)}\boldsymbol{x}_{\text{out}}^{(s-1)} \tag{3-74}$$

其中$\boldsymbol{D}^{(s)}$表示第$s$层的灵敏度向量，定义为：

$$\boldsymbol{D}^{(s)} = -\left[\frac{\partial E_q}{\partial v_1^{(s)}}, \frac{\partial E_q}{\partial v_2^{(s)}}, \cdots, \frac{\partial E_q}{\partial v_{n_s}^{(s)}}\right]^T \tag{3-75}$$

*灵敏度计算*

让我们首先考虑图3-4给出的网络输出层。在灵敏度向量中的单个项求值为：

$$\begin{aligned}\frac{\partial E_q}{\partial v_i^{(3)}} &= \frac{\partial}{\partial v_i^{(3)}}\left[\frac{1}{2}(\boldsymbol{d}_q - \boldsymbol{x}_{\text{out}}^{(3)})^T(\boldsymbol{d}_q - \boldsymbol{x}_{\text{out}}^{(3)})\right] \\ &= \frac{\partial}{\partial v_i^{(3)}}\left\{\frac{1}{2}\sum_{h=1}^{n_3}\left[d_{qh} - f(v_h^{(3)})\right]^2\right\} \\ &= -\left[d_{qi} - f(v_i^{(3)})\right]f'(v_i^{(3)}) = -(d_{qi} - x_{\text{out},i}^{(3)})g(v_i^{(3)})\end{aligned} \tag{3-76}$$

其中$g(x) = \mathrm{d}f(x)/\mathrm{d}x$表示激活函数的一阶导数。

把式（3-76）代入式（3-75），把网络输出层的灵敏度向量表示为：

$$\boldsymbol{D}^{(3)} = \boldsymbol{G}(v^{(3)})(\boldsymbol{d}_q - \boldsymbol{x}_{\text{out}}^{(3)}) \tag{3-77}$$

其中 $\boldsymbol{G}(\boldsymbol{v}^{(3)}) = \text{diag}[g(v_1^{(3)}), g(v_2^{(3)}), \cdots, g(v_{n_3}^{(3)})]$。现在，我们寻找网络的第二层和第一层的灵敏度向量。使用式（3-75）中灵敏度向量的定义和链式法则，能够将第二层的灵敏度写为：

$$\boldsymbol{D}^{(2)} = -\frac{\partial E_q}{\partial v^{(3)}} = -\left(\frac{\partial v^{(3)}}{\partial v^{(2)}}\right)^T \frac{\partial E_q}{\partial v^{(3)}} = \left(\frac{\partial v^{(3)}}{\partial v^{(2)}}\right)^T \boldsymbol{D}^{(3)} \tag{3-78}$$

等式（3-78）表明第二层灵敏度向量能够作为输出层灵敏度的函数。在式（3-78）中的两个灵敏度之间的线性变换用如下的雅可比矩阵表示：

$$\left(\frac{\partial v^{(3)}}{\partial v^{(2)}}\right)^T = \begin{bmatrix} \frac{\partial v_1^{(3)}}{\partial v_1^{(2)}} & \frac{\partial v_1^{(3)}}{\partial v_2^{(2)}} & \cdots & \frac{\partial v_1^{(3)}}{\partial v_{n_2}^{(2)}} \\ \frac{\partial v_2^{(3)}}{\partial v_1^{(2)}} & \frac{\partial v_2^{(3)}}{\partial v_2^{(2)}} & \cdots & \frac{\partial v_2^{(3)}}{\partial v_{n_2}^{(2)}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial v_{n_3}^{(3)}}{\partial v_1^{(2)}} & \frac{\partial v_{n_3}^{(3)}}{\partial v_2^{(2)}} & \cdots & \frac{\partial v_{n_3}^{(3)}}{\partial v_{n_2}^{(2)}} \end{bmatrix}^T \tag{3-79}$$

118

类似地，我们能够证明第一层的灵敏度向量可表示为：

$$\boldsymbol{D}^{(1)} = \left(\frac{\partial v^{(2)}}{\partial v^{(1)}}\right)^T \boldsymbol{D}^{(2)} \tag{3-80}$$

考虑式（3-79）给出的变换雅可比矩阵的一个单项：

$$\begin{aligned}\frac{\partial v_i^{(3)}}{\partial v_j^{(2)}} &= \frac{\partial}{\partial v_j^{(2)}}\left[\sum_{h=1}^{n_2} w_{ih}^{(3)} x_{\text{out},h}^{(2)}\right] = w_{ij}^{(3)}\frac{\partial x_{\text{out},j}^{(2)}}{\partial v_j^{(2)}} \\ &= w_{ij}^{(3)}\frac{\partial f(v_j^{(2)})}{\partial v_j^{(2)}} = w_{ij}^{(3)} f'(v_j^{(2)}) = w_{ij}^{(3)} g(v_j^{(2)})\end{aligned} \tag{3-81}$$

把式（3-81）代入式（3-79），我们发现雅可比矩阵能够重写为：

$$\left(\frac{\partial v^{(3)}}{\partial v^{(2)}}\right)^T = [\boldsymbol{W}^{(3)}\boldsymbol{G}(v^{(2)})]^T = \boldsymbol{G}(v^{(2)})\boldsymbol{W}^{(3)T} \tag{3-82}$$

其中 $\boldsymbol{G}(\boldsymbol{v}^{(2)}) = \text{diag}[g(v_1^{(2)}), g(v_2^{(2)}), \cdots, g(v_{n_2}^{(2)})]$。

最后，合并式（3-82）与式（3-80）得到

$$\boldsymbol{D}^{(2)} = \boldsymbol{G}(\boldsymbol{v}^{(2)})\, \boldsymbol{W}^{(3)T}\boldsymbol{D}^{(3)} \tag{3-83}$$

使用同样的方法，我们能够把第一层的灵敏性表示为：

$$\boldsymbol{D}^{(1)} = \boldsymbol{G}(\boldsymbol{v}^{(1)})\, \boldsymbol{W}^{(2)T}\boldsymbol{D}^{(2)} \tag{3-84}$$

等式（3-74）、式（3-77）、式（3-83）和式（3-84）构成反向传播学习规则的向量矩阵表达式。

**反向传播算法的向量矩阵形式**

**步骤1** 提交一个输入模式并且计算网络和在所有内部层的输出。

**步骤2** 对于每一层，按照下式计算灵敏度向量：

对于输出层：$\boldsymbol{D}^{(s)} = \boldsymbol{G}(\boldsymbol{v}^{(s)})(\boldsymbol{d}_q - \boldsymbol{x}_{\text{out}}^{(s)})$

对于所有隐藏层：$\boldsymbol{D}^{(s-1)} = \boldsymbol{G}(\boldsymbol{v}^{(s-1)})\, \boldsymbol{w}^{(s)T}\boldsymbol{D}^{(s)}$

**步骤3** 按照下式更新网络突触权值

$$\boldsymbol{W}^{(s)}(k+1) = \boldsymbol{W}^{(s)} + \alpha^{(s)}\boldsymbol{D}^{(s)}\boldsymbol{x}_{out}^{(s-1)T}$$

**步骤4** 继续步骤1到步骤3直到网络达到期望的映射精度。 □ 119

## 3.4 加速学习反向传播算法

这节介绍标准反向传播算法的几个修改。如先前指出的，MLP NN的学习实际上是最小化实际的与期望的网络输出之间的均方误差（MSE）。使用MSE标准的相关问题在数值分析中已经很好地研究。这里介绍的反向传播算法的修改基于对MLP NN训练问题应用高级的数值技术。

### 3.4.1 前馈多层感知器的共轭梯度反向传播

共轭梯度方法（参看A.5.5节）是用于解决各种最优化问题的著名数值技术。由于它代表了最速下降算法的简单性与牛顿方法（参看A.5.3节）的快速二次收敛性之间的一个良好折中，所以被广泛使用。已经发展了几个基于共轭梯度方法的训练MLP NN的方法，在[36]中给出了综合性的总结。这些算法的绝大部分建立在假定解的邻域内网络所有权值的误差函数能够用一个二次函数来准确地近似。也就是说，

$$\boldsymbol{J}(\boldsymbol{w}) = \frac{1}{2P}\sum_{p=1}^{P}\sum_{h=1}^{n_s}(d_{ph} - y_{ph})^2 = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{Q}\boldsymbol{w} - \boldsymbol{b}^T\boldsymbol{w} \tag{3-85}$$

其中$\boldsymbol{w}$代表网络中所有权值，$\boldsymbol{P}$是训练模式的总数，$n_s$是输出层中神经元的数目，$d_{ph}$是输出层中第$h$个神经元对第$p$个训练输入的期望输出，$y_{ph}$是输出层中第$h$个神经元对第$p$个训练输入的实际输出。式（3-85）中矩阵$\boldsymbol{Q}$是二阶偏导数方阵，即黑塞（Hessian）矩阵。黑塞矩阵的阶数等于网络中权值的总数。共轭梯度算法试图发现在误差表面的共轭方向系统并执行这些方向上的权值更新。由于黑塞矩阵$\boldsymbol{Q}$非常大，它的计算是不可行的，绝大部分用于训练MLP NN的共轭梯度算法试图不用显式计算黑塞矩阵来发现共轭梯度方向。用于训练MLP NN的共轭梯度方法的一些实际计算在[35, 37, 38]中找到。

在这节介绍的算法采用和基于共轭梯度的MLP NN训练不同的方法。代替考虑误差表面， 120
该算法在每一个神经元处构造一组*法方程*，然后使用共轭梯度方法迭代求解。在该算法的表达中，紧密地遵循在[39]中的处理。

在推导标准反向传播算法的过程中，我们已经指出它可以看作在网络训练中一起工作的两个不同过程的折中。第一个过程是在MLP NN的每个节点处局部误差的估计。通过输出层误差的反向传播完成，输出层误差能够由隐藏层神经元的实际与期望响应的差异来显式计算。第二过程是网络权值的更新。在3.3.1节中，标准反向传播算法使用LMS算法来执行权值更新。

理解这两个过程是分开的并且本质上相互独立是非常重要的。它允许不同权值更新算法的发展而保持误差反向传播过程。使用更复杂的梯度下降技术更新网络权值将加快训练过程。

当每个训练模式网络输出在期望输出的指定容忍度内的时候，MLP NN认为是训练好的。为了达到这个目的，网络的每个神经元不得不恰当地训练。换句话说，训练MLP NN意味着训练网络内的每个节点达到期望响应。参看图3-4，每个节点由自适应线性单元（通常称为线性组合器）和其后的S形非线性所组成。非线性的存在是MLP NN的强大映射能力的来源。另一方面，正是这些非线性的出现增加了网络训练的复杂性。我们可以观察到，如果线性组合器产生一个恰当的输入给激活函数，非线性激活函数的输出将是期望响应。所以，我们可以归纳为：训练MLP NN主要涉及调整权值，使网络的每个线性组合器产生期望输出。

*线性组合器的法方程*

考虑图3-4给出的MLP NN第$s$层第$i$个线性组合器。当第$q$个输入模式提交给网络时，组合器的输出当作组合器权值 $\boldsymbol{w}_i^{(s)} \in \Re^{n_{s-1}\times 1}$ 与特定层的输入向量 $\boldsymbol{x}_{\text{out},q}^{(s-1)} \in \Re^{n_{s-1}\times 1}$ 之间的内积。即，

$$v_i^{(s)} = \boldsymbol{w}_i^{(s)T}\boldsymbol{x}_{\text{out},q}^{(s-1)} \tag{3-86}$$

假定某时刻对于特定组合器的期望输出是 $d_{i,q}^{(s)}$，它对训练集合中的每个模式是已知的。有效的训练MLP NN假设为它的所有线性组合器的训练。所以，学习算法的目标是最小化如下的二
121 乘误差代价函数

$$J_i^{(s)} = \frac{1}{2}\sum_{q=1}^{M}(d_{i,q}^{(s)} - v_{i,q}^{(s)})^2 \tag{3-87}$$

其中$M$表示训练集合中向量的总数。把式（3-86）代入式（3-87）中，可写为

$$J_i^{(s)} = \frac{1}{2}\sum_{q=1}^{M}(d_{i,q}^{(s)} - \boldsymbol{w}_i^{(s)T}\boldsymbol{x}_{\text{out},q}^{(s-1)})^2 \tag{3-88}$$

为了寻找最小化式（3-88）给出的代价函数的权值向量，可以求得它的关于 $\boldsymbol{w}_i^{(s)}$ 的偏导数，并且使之为0，即

$$\frac{\partial \boldsymbol{J}_i^{(s)}}{\partial \boldsymbol{w}_i^{(s)}} = \sum_{q=1}^{M}(-d_{i,q}^{(s)}\boldsymbol{x}_{\text{out},q}^{(s-1)} + \boldsymbol{x}_{\text{out},q}^{(s-1)}\boldsymbol{x}_{\text{out},q}^{(s-1)T}\boldsymbol{w}_i^{(s)}) = 0 \tag{3-89}$$

定义

$$\boldsymbol{C}_i^{(s)} \triangleq \sum_{q=1}^{M}\boldsymbol{x}_{\text{out},q}^{(s-1)}\boldsymbol{x}_{\text{out},q}^{(s-1)T} \tag{3-90}$$

且

$$\boldsymbol{p}_i^{(s)} \triangleq \sum_{q=1}^{M} d_{i,q}^{(s)}\boldsymbol{x}_{\text{out},q}^{(s-1)} \tag{3-91}$$

等式（3-89）以向量矩阵形式重新整理为

$$\boldsymbol{C}_i^{(s)}\boldsymbol{w}_i^{(s)} = \boldsymbol{p}_i^{(s)} \tag{3-92}$$

矩阵 $\boldsymbol{C}_i^{(s)}$ 能够解释为第$s$层的输入向量间的协方差矩阵的估计，并且向量 $\boldsymbol{p}_i^{(s)}$ 是第$s$层输入与线性组合器的期望输出之间的互相关向量的一个估计。注意矩阵 $\boldsymbol{C}_i^{(s)}$ 并不依靠线性组合器在层内的位置，因此，下标$i$可以省略。式（3-92）中的矩阵向量方程在自适应滤波的背景中以*确定法方程*（deterministic normal equation）的名称而著名[29]。法方程的解最小化式（3-87）

给出的二次误差函数。总之，对于MLP NN的每个线性组合器输出都可以写成式（3-92）给出的方程组形式，并且，网络训练能够很方便地看作一个涉及它们解的过程。

*求解法方程的共轭梯度方法*

有一些用于求解式（3-92）给出的线性等式系统的数值技术。其中之一是共轭梯度方法。在这节中，我们仅仅给出算法。强烈鼓励读者仔细阅读A.5.5节。 122

**求解法方程的共轭梯度方法**

**步骤1** 初始化权值向量 $\boldsymbol{w}_i^{(s)}(0)\in\Re^{n_s\times 1}$ 的分量为一些小的任意值。

**步骤2** 设置$k=0$。计算初始化共轭方向$\boldsymbol{d}_0$和增益向量$\boldsymbol{g}_0$，且

$$\boldsymbol{d}_0=-\boldsymbol{g}_0=\boldsymbol{p}_i^{(s)}-\boldsymbol{C}^{(s)}\boldsymbol{w}_i^{(s)}(0)$$

**步骤3** 确定共轭向量系数

$$\alpha_k=-\frac{\boldsymbol{g}_k^T\boldsymbol{d}_k}{\boldsymbol{d}_k^T\boldsymbol{C}^{(s)}\boldsymbol{d}_k}\text{ 其中 }\boldsymbol{g}_k=\boldsymbol{C}^{(s)}\boldsymbol{w}_i^{(s)}(k)-\boldsymbol{p}_i^{(s)}$$

**步骤4** 更新权值向量

$$\boldsymbol{w}_i^{(s)}(k+1)=\boldsymbol{w}_i^{(s)}(k)+\alpha_k\boldsymbol{d}_k$$

**步骤5** 确定新的增益向量

$$\boldsymbol{g}_{k+1}=\boldsymbol{C}^{(s)}\boldsymbol{w}_i^{(s)}(k+1)-\boldsymbol{p}_i^{(s)}$$

**步骤6** 确定新的共轭梯度方向

$$\boldsymbol{d}_{k+1}=-\boldsymbol{g}_{k+1}+\beta_k\boldsymbol{d}_k\text{ 其中 }\beta_k=\frac{\boldsymbol{g}_{k+1}^T\boldsymbol{C}^{(s)}\boldsymbol{d}_k}{\boldsymbol{g}_k^T\boldsymbol{g}_k}$$

**步骤7** 设$k=k+1$，并且检验终止条件。如果$k<n$，转到步骤3；否则，停止。 □

完成上述步骤得到的向量 $\boldsymbol{w}_i^{(s)*}$求解特定线性组合器的法方程。

*训练算法*

以前一节给出的形式，存在应用共轭梯度算法相关的一些主要困难。除输出层节点外，特定节点的期望输出并不知道。这意味着对隐藏层的所有节点不得不估计。为了完成这点，参考3.3.1节及局部误差的物理解释。我们已经提到，局部误差表示神经元的实际输出与期望输出之间的误差估计。该估计仅仅基于一个训练输入/输出对和网络权值的当前值，因此，我们不奢望在网络训练的初期它是准确的。然而，随着训练进行，误差的估计变得更精确。知道特定节点的实际输出和局部误差，能够计算出第$s$层第$i$个神经元对第$q$个训练模式的期望输出

$$\bar{d}_{i,q}^{(s)}=x_{\text{out},i,q}^{(s)}+\mu\delta_{i,q}^{(s)}\tag{3-93}$$

其中$\mu$是一正数，通常取值在10到400之间[39]。式（3-93）给出了MLP NN的每个神经元的期望输出的估计。当导出法方程组时，假设已知线性组合器的期望输出。由于激活函数通常选择为单调增长S形函数，神经元的输出与线性组合器的输出之间存在一一对应。给出神经元的输出，组合器的输出可以计算为 123

$$\hat{v}_{i,q}^{(s)}=f^{-1}(\bar{d}_{i,q}^{(s)})\tag{3-94}$$

通过选择合适的激活函数，能够容易地表示出(3.94）中的逆。例如，如果激活函数选择为

$$y=f(t)=\frac{1-e^{-\sigma t}}{1+e^{\sigma t}}\tag{3-95}$$

其中$\sigma$是控制倾斜度的参数，逆函数可表示为

$$t = f^{-1}(y) = \frac{1}{\sigma}\ln\frac{1+y}{1-y} \tag{3-96}$$

应用式（3-96）～式（3-95），得到线性组合器的期望输出为

$$\hat{v}_{i,q}^{(s)} = \frac{1}{\sigma}\ln\frac{1+\bar{d}_{i,q}^{(s)}}{1-\bar{d}_{i,q}^{(s)}} \tag{3-97}$$

前面一节中描述的共轭梯度算法假定显式知道协方差矩阵$\boldsymbol{C}^{(s)}$和互相关向量$\boldsymbol{p}_i^{(s)}$。当然，事先并不知道，所以不得不在训练过程中估计它们。做到这一点的方便办法为：对于每次提交的输入/输出训练对更新它们的估计。对第$s$层的相关矩阵估计可写为

$$\boldsymbol{C}^{(s)}(k+1) = b\boldsymbol{C}^{(s)}(k) + \boldsymbol{x}_{\text{out},q}^{(s-1)}\boldsymbol{x}_{\text{out},q}^{(s-1)T} \tag{3-98}$$

类似地，每个线性组合器的互相关向量可估计为

$$\boldsymbol{p}_i^{(s)}(k+1) = b\boldsymbol{p}_i^{(s)}(k) + \hat{v}_i^{(s)}\boldsymbol{x}_{\text{out},q}^{(s-1)} \tag{3-99}$$

在式（3-98）和式（3-99）中的系数$b$称为遗忘因子，决定以前的协方差矩阵和互相关向量（在式（3-98）和式（3-99）右边的第一项）的瞬间估计的权值。典型地，$b$设置在0.9～0.99的范围内。

基于上面的评述，我们提供算法的一个小结。

**训练MLP NN的基于共轭梯度算法**

**步骤1** 初始化网络权值为一些小的随机值。在3.3.2节中描述的任何权值初始化技术都可以使用。

**步骤2** 传播第$q$个训练模式到整个网络，计算每个节点的输出。

**步骤3** 计算网络中每个节点的局部误差。对于输出节点，局部误差计算为

$$\delta_{i,q}^{(s)} = (d_{i,q} - x_{\text{out},i,q}^{(s)})g(v_{i,q}^{(s)})$$

124 其中$g(\cdot)$是激活函数$f(\cdot)$的导数。对于每个隐藏层节点，局部误差计算为

$$\delta_{i,q}^{(s)} = \left(\sum_{h=1}^{n_{s+1}}\delta_{h,q}^{(s+1)}w_{h,i}^{(s+1)}\right)g(v_{i,q}^{(s)})$$

**步骤4** 对于每个线性组合器的估计，期望输出为

$$\hat{v}_{i,q}^{(s)} = f^{-1}(\bar{d}_{i,q}^{(s)}) \text{ 其中 } \bar{d}_{i,q}^{(s)} = x_{\text{out},i,q}^{(s)} + \mu\delta_{i,q}^{(s)}$$

**步骤5** 更新每层的协方差矩阵估计

$$\boldsymbol{C}^{(s)}(k) = b\boldsymbol{C}^{(s)}(k-1) + \boldsymbol{x}_{\text{out},q}^{(s-1)}\boldsymbol{x}_{\text{out},q}^{(s-1)T}$$

更新每个节点的互相关向量的估计

$$\boldsymbol{p}_i^{(s)}(k) = b\boldsymbol{p}_i^{(s)}(k-1) + \hat{v}_i^{(s)}\boldsymbol{x}_{\text{out},q}^{(s-1)}$$

其中$k$为模式提交的指标。

**步骤6** 更新网络中每个节点权值向量如下

(a) 在每个节点处计算

$$\boldsymbol{g}_i^{(s)}(k) = \boldsymbol{C}^{(s)}(k)\boldsymbol{w}_i^{(s)}(k) - \boldsymbol{p}_i^{(s)}\text{，否则}$$

如果$\mathbf{g}_i^{(s)} = 0$，并不更新节点的权值向量，转到步骤7；否则执行下面的步骤：

(b) 找到方向$\boldsymbol{d}(k)$。如果迭代次数是节点权值数量的整数倍，则

$$\boldsymbol{d}_i^{(s)}(k) = -\boldsymbol{g}_i^{(s)}(k)$$

否则

$$\boldsymbol{d}_i^{(s)}(k) = -\boldsymbol{g}_i^{(s)}(k) + \beta_i^{(s)}\boldsymbol{d}_i^{(s)}(k-1)$$

其中

$$\beta_i^{(s)} = -\boldsymbol{g}_i^{(s)T}(k)\frac{\boldsymbol{C}^{(s)}(k)\boldsymbol{d}_i^{(s)}(k-1)}{\boldsymbol{d}_i^{(s)T}(k-1)\boldsymbol{C}^{(s)}(k)\boldsymbol{d}_i^{(s)}(k-1)}$$

(c) 计算步长大小

$$\alpha_i^{(s)}(k) = -\frac{\boldsymbol{g}_i^{(s)T}(k)\boldsymbol{d}_i^{(s)}(k)}{\boldsymbol{d}_i^{(s)T}(k)\boldsymbol{C}^{(s)}(k)\boldsymbol{d}_i^{(s)}(k)}$$

(d) 修改权值向量根据

$$\boldsymbol{w}_i^{(s)}(k) = \boldsymbol{w}_i^{(s)}(k-1) + \alpha_i^{(s)}(k)\boldsymbol{d}_i^{(s)}(k)$$

**步骤7** 如果网络并未收敛，回到步骤2。 □

一些评注是关于上面介绍的算法的。用于求解法方程的共轭梯度算法至少需要$n$步才收敛。初看起来，很明显，每次提交一个新训练输入/输出对时，线性组合器权值向量更新需要执行$n$步。然而，并不这样做是由于下面的原因。在训练期间，仅仅知道协方差矩阵和交叉相关向量的估计。所以，相应的法方程的精确解并不产生所需要的权值向量。在网络训练的初期这是非常真实的。随着训练的进行，互相关向量和协方差矩阵的估计逐渐稳定，因为它们对于几个训练输入/输出对保持近似不变，对于每个输入/输出对执行一个共轭梯度步骤就足够了。当然，由于整个过程本来是非线性的，网络训练将并不在$n$步内完成。从步骤6我们发现在$n$个共轭梯度步骤以后，共轭梯度更新重新初始化（即开始于最速下降步骤）。 125

### 3.4.2 基于最小二乘的递归反向传播算法

我们在前一节中已经看见MLP NN的训练可以理解为一个求解确定法方程组的过程。网络中每个线性组合器的法方程组可写为：

$$\boldsymbol{C}^{(s)}\boldsymbol{w}_i^{(s)} = \boldsymbol{p}_i^{(s)} \tag{3-100}$$

其中$\boldsymbol{C}^{(s)}$表示第$s$层输入的协方差矩阵，$\boldsymbol{p}_i^{(s)}$是第$s$层输入与第$s$层上的第$i$个线性组合器期望输出之间的互相关向量，并且$\boldsymbol{w}_i^{(s)}$表示到第$s$层的第$i$个线性组合器的突触权值向量。如果假定协方差矩阵$\boldsymbol{C}^{(s)}$和互相关向量$\boldsymbol{p}_i^{(s)}$已知，适当的权值向量可以计算为

$$\boldsymbol{w}_i^{(s)} = [\boldsymbol{C}^{(s)}]^{-1}\boldsymbol{p}_i^{(s)} \tag{3-101}$$

然而，我们并没有清楚知道协方差矩阵或互相关向量，并且在网络训练的整个过程中不得不估计它们。通过使用式（3-98）和式（3-99）能够得到这些估计，式（3-100）中的法方程组系统不得不用一个迭代方法来求解。在前一节中，我们运用共轭梯度方法来完成这个任务。这里，我们探索基于递归最小二乘（RLS）方法的求解式（3-100）的方程组的另外一种方法[29]。RLS算法能够通过对每次提交的输入/输出训练对直接实现相关矩阵的逆$[\boldsymbol{C}^{(s)}]^{-1}$的自适应递归估计来求解（3-101）中的权值向量。RLS算法可以视为一个卡尔曼滤波（Kalman filtering）的特例[29]，由于这个原因这里介绍的方法有时称作基于卡尔曼滤波的反向传播算法[40]。为了推进RLS反向传播算法的进展，我们从一个在矩阵代数中熟知为*矩阵求逆引理*的重要结论开始。

*矩阵求逆引理*（伍德伯里(Woodbury)恒等式）

假定$\boldsymbol{A} \in \Re^{m\times m}$和$\boldsymbol{B} \in \Re^{m\times m}$是具有下列关系的两个正定矩阵（见A.2.7节）：

$$\boldsymbol{A} = \boldsymbol{B}^{-1} + \boldsymbol{C}\boldsymbol{D}^{-1}\boldsymbol{C}^T \tag{3-102}$$

其中$D \in \Re^{n\times n}$是另一个正定矩阵，$C \in \Re^{m\times n}$。那么，矩阵$A$的逆矩阵可写成

126
$$A^{-1}=B-BC(D+C^TBC)^{-1}C^TB \tag{3-103}$$

RLS反向传播算法的推导

考虑式（3-98）中的协方差矩阵的更新方程，为了方便在式（3-104）中重复

$$C^{(s)}(k)=bC^{(s)}(k-1)+x_{\text{out},q}^{(s-1)}(k)x_{\text{out},q}^{(s-1)T}(k) \tag{3-104}$$

比较式（3-104）与式（3-102），可以令$A=C^{(s)}(k)$，$B^{-1}=bC^{(s)}(k-1)$，$C=x_{\text{out},q}^{(s-1)}(k)$并且$D=1$。为了简化符号，我们定义

$$x_{\text{out},q}^{(s)}(k)\triangleq X^{(s)} \tag{3-105}$$

使用式（3-103）中得到的结果，能够得到

$$[C^{(s)}(k)]^{-1}=b^{-1}[C^{(s)}(k-1)]^{-1}-\frac{b^{-1}[C^{(s)}(k-1)]^{-1}X^{(s-1)}X^{(s-1)T}b^{-1}[C^{(s)}(k-1)]^{-1}}{1+X^{(s-1)T}b^{-1}[C^{(s)}(k-1)]^{-1}X^{(s-1)}} \tag{3-106}$$

定义

$$K(k)\triangleq\frac{[C^{(s-1)}(k-1)]^{-1}X^{(s-1)}}{b+X^{(s-1)T}[C^{(s-1)}(k-1)]^{-1}X^{(s-1)}} \tag{3-107}$$

可以把式（3-107）代入式（3-106），并写出

$$[C^{(s)}(k)]^{-1}=b^{-1}\{[C^{(s)}(k-1)]^{-1}-K(k)X^{(s-1)T}[C^{(s)}(k-1)]^{-1}\} \tag{3-108}$$

等式（3-107）和式（3-108）给出了MLP NN中每个隐藏层的协方差矩阵的逆的递归估计机制。联合式（3-99）和式（3-101）得到

$$w_i^{(s)}(k)=[C^{(s)}(k)]^{-1}p^{(s)}(k)=[C^{(s)}(k)]^{-1}[bp_i^{(s)}(k-1)+\hat{v}_i^{(s)}(k)X^{(s-1)}] \tag{3-109}$$

把式（3-108）中的协方差矩阵的逆的表达式代入式（3-109）得到

$$\begin{aligned}w_i^{(s)}(k)&=b^{-1}\{[C^{(s)}(k-1)]^{-1}-K(k)X^{(s-1)T}[C^{(s)}(k-1)]^{-1}\}\\&\quad\cdot[bp_i^{(s)}(k-1)+\hat{v}_i^{(s)}(k)X^{(s-1)}]\\&=[C^{(s)}(k-1)]^{-1}p_i^{(s)}(k-1)-K(k)X^{(s-1)T}[C^{(s)}(k-1)]^{-1}\\&\quad p_i^{(s)}(k-1)+b^{-1}[C^{(s)}(k-1)]^{-1}\hat{v}_i^{(s)}(k)X^{(s-1)}-b^{-1}K(k)X^{(s-1)T}\\&\quad\cdot[C^{(s)}(k-1)]^{-1}\hat{v}_i^{(s)}(k)X^{(s-1)}\end{aligned} \tag{3-110}$$

127

或者

$$\begin{aligned}w_i^{(s)}(k)&=w_i^{(s)}(k-1)-K(k)X^{(s-1)T}w_i^{(s)}(k-1)+\hat{v}_i^{(s)}(k)b^{-1}[C^{(s)}(k-1)]^{-1}X^{(s-1)}\\&\quad-\hat{v}_i^{(s)}(k)b^{-1}\frac{[C^{(s)}(k-1)]^{-1}X^{(s-1)}X^{(s-1)T}[C^{(s)}(k-1)]^{-1}X^{(s-1)}}{b+X^{(s-1)T}[C^{(s)}(k-1)]^{-1}X^{(s-1)}}\end{aligned} \tag{3-111}$$

通过合并式（3-111）的最后两项，权值更新方程简化为：

$$\begin{aligned}w_i^{(s)}(k)&=w_i^{(s)}(k-1)-K(k)X^{(s-1)T}w_i^{(s)}(k-1)+\hat{v}_i^{(s)}(k)K(k)\\&=w_i^{(s)}(k-1)+K(k)[\hat{v}_i^{(s)}(k)-X^{(s-1)T}w_i^{(s)}(k-1)]\\&=w_i^{(s)}(k-1)+K(k)[\hat{v}_i^{(s)}(k)-v_i^{(s)}(k)]\end{aligned} \tag{3-112}$$

等式（3-112）是网络第$s$层第$i$个线性组合器的权值更新方程。更新和线性组合器的期望输出与实际输出之间的差异成比例的。这是Hebb类型的所有学习算法的情况，RLS仅是一个特殊例子。RLS算法的关键属性是对于网络中每个神经元和网络训练的每一步的学习率动态地改

变。学习率当作式（3-112）中的Kalman 增益矩阵的一个项来计算。

基于更新方程（3-112），训练MLP NN的RLS算法可以小结为如下步骤[40]：

**反向传播算法的递归最小二乘形式**

**步骤1** 按照3.3.2节讨论的任意标准最初始化过程来初始化网络权值。

**步骤2** 提交一个输入模式，计算所有线性组合器的响应 $v_i^{(s)}$ 和网络中所有神经元输出 $x_{\text{out},i}^{(s)}$。

**步骤3** 对于网络的每一层，按照下面的式子计算Kalman增益矩阵，并更新协方差矩阵估计：

$$\boldsymbol{A}^{(s)}(k)=[\boldsymbol{C}^{(s)}(k-1)]^{-1}\boldsymbol{x}_{\text{out}}^{(s-1)}(k)$$

更新第$s$层的Kalman 增益矩阵

$$\boldsymbol{K}^{(s)}(k)=\frac{\boldsymbol{A}^{(s)}(k)}{b+\boldsymbol{x}_{\text{out}}^{(s-1)T}(k)\boldsymbol{A}^{(s)}(k)}$$

根据下式更新第$s$层的协方差矩阵

$$[\boldsymbol{C}^{(s)}(k)]^{-1}=b^{-1}\{[\boldsymbol{C}^{(s)}(k-1)]^{-1}-\boldsymbol{K}^{(s)}(k)\boldsymbol{A}^{(s)T}(k)\}$$

**步骤4** 按照下式计算和反向传播输出层的局部误差

$$\delta_i^{(s)}=(d_{qh}-x_{\text{out},i}^{(s)})g(v_i^{(s)})$$

128

对于隐藏层

$$\delta_i^{(s)}=\left(\sum_{h=1}^{n_2}\delta_h^{(s+1)}w_{hi}^{(s+1)}\right)g(v_i^{(s)})$$

其中$g(z)=df(z)/dz$，而$f(z)$是神经元激活函数。

**步骤5** 对于每个线性组合器，按照下式估计期望输入：

$$\hat{v}_i^{(s)}=f^{-1}(x_{\text{out},i}^{(s)}+\mu\delta_i^{(s)})$$

其中$f^{-1}(z)$是神经元激活函数的反函数。

**步骤6** 按照以下式子更新网络每层的权值

$$w_i^{(s)}(k)=w_i^{(s)}(k-1)+\boldsymbol{K}(k)[\hat{v}_i^{(s)}(k)-v_i^{(s)}(k)]$$

**步骤7** 如果网络收敛则停止；否则转到步骤2。 □

从上面的算法小结看，RLS算法很明显与标准反向传播算法比较对MLP NN的训练涉及增加计算。然而，在[40]和[41]的详细分析说明了RLS算法达到收敛需要更少的迭代次数，当和标准反向传播比较时，整个训练时间减少一个数量级还多。RLS算法已经证明对初始的权值选择更不灵敏，并且学习率的自适应特性使算法更不易陷入局部最小值。在算法性能的重大改进能够由[40]中描述的并行实现来取得。

### 3.4.3 具有自适应激活函数斜度的反向传播

从迄今为止我们已经看到的反向传播的权值更新方程组，我们观察到更新速率与非线性激活函数的导数成比例。如以前讨论的那样，在MLP NN中一个神经元的典型激活函数具有一个钟形导数的S形函数，如图3-6所示。在网络训练期间，线性组合器的输出可能落入激活函数的饱和区域内。在该区域的激活函数的导数非常小，而由于权值更新直接依赖导数的大小，所以，学习速率变得极端地慢。线性组合器的输出移出饱和区域之前，可能需要许多次迭代。阻止这种饱和的直接方法就是通过降低激活函数的斜度来增加非饱和部分的大小。然而，降低斜度使得网络行为更像一个线性网络，实际上减弱了多层网络的优势（由于任意多的具有线性激活函数的层被单层所代替）。因此，激活函数斜度有一个最优值，以平衡网络训练速度和它的映射能力。对任何网络神经元该值不必相同。由于MLP NN结构的复杂性，在 129

训练开始以前决定网络每个神经元的激活函数的最佳斜度是不可能的。所以，它们的一个自适应方式的估计提供一个可行的替代。这是我们在本节采用的方法。

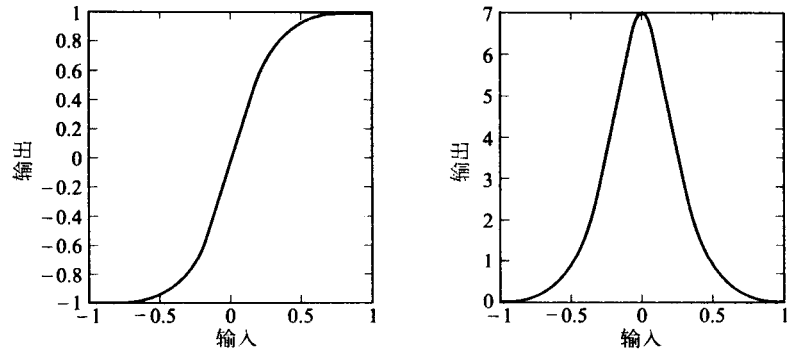



图3-6　a) 典型的MLP NN激活函数；b) a) 中激活函数的导数

斜度的自适应估计过程使用与用于导出权值更新方程组一样的最优化准则来导出。特别地，选择斜度使得性能准则最小化

$$E_q = \frac{1}{2}(\boldsymbol{d}_q - \boldsymbol{x}_{\text{out}}^{(s)})^T(\boldsymbol{d}_q - \boldsymbol{x}_{\text{out}}^{(s)}) = \frac{1}{2}\sum_{h=1}^{n_s}(d_{qh} - x_{\text{out},h}^{(s)})^2 \tag{3-113}$$

其中$s$指网络中层数并且$\boldsymbol{d}_q \in \Re^{n_s \times 1}$和$\boldsymbol{x}_{\text{out}}^{(s)}$分别代表第$q$个训练模式对应的网络期望输出和实际输出。考虑如下表达式给出的一个S形激活函数

$$f(v,\gamma) = \frac{1-\exp(-\gamma v)}{1+\exp(-\gamma v)} \tag{3-114}$$

其中$v$是非线性的输入（线性组合器的输出），而$\gamma$是校正的倾斜参数，以使式（3-113）最小化。考虑网络第$s$层第$i$个神经元的非线性特性，应用与用于导出标准反向传播的权值更新方程组的一样的方法，得到

$$\gamma_i^{(s)}(k+1) = \gamma_i^{(s)}(k) - \beta\frac{\partial E_q}{\partial \gamma_i^{(s)}} \tag{3-115}$$

130 使用链式法则，在式（3-115）右边的第二项能够重新写成：

$$\frac{\partial E_q}{\partial \gamma_i^{(s)}} = \frac{\partial E_q}{\partial v_i^{(s)}}\frac{\partial v_i^{(s)}}{\partial x_{\text{out},i}^{(s)}}\frac{\partial x_{\text{out},i}^{(s)}}{\partial \gamma_i^{(s)}} = -\delta_i^{(s)}\frac{1}{\partial x_{\text{out},i}^{(s)}/\partial v_i^{(s)}}\frac{\partial x_{\text{out},i}^{(s)}}{\partial \gamma_i^{(s)}} = -\delta_i^{(s)}\frac{f_\gamma(v,\gamma)}{f_v(v,\gamma)} \tag{3-116}$$

其中$\delta_i^{(s)}$是第$s$层第$i$个神经元的局部误差，$f_v(v,\gamma)$和$f_\gamma(v,\gamma)$分别表示激活函数关于$v$和$\gamma$的偏导数。如果使用式（3-114）中的激活函数，我们有

$$f_\gamma(v,\gamma) = \frac{1}{2}[1-f(v,\gamma)^2] \tag{3-117}$$

和

$$f_v(v,\gamma) = \frac{1}{2}[1-f(v,\gamma)^2] \tag{3-118}$$

通过把式（3-116）、式（3-117）和式（3-118）代入式（3-115），用于激活函数斜度的更新方程变成

$$\gamma_i^{(s)}(k+1) = \gamma_i^{(s)}(k) + \beta\delta_i^{(s)} \tag{3-119}$$

通常，添加一个动量项到更新方程（3-119）来提高稳定性。另外，为了避免神经元网络映射的可能线性化，限制斜度以免变得比预先给定值$\gamma_{min}$更小。具有自适应斜度的整个反向传播算法能够小结为以下六个步骤：

**自适应激活函数倾斜的反向传播算法**

**步骤1** 按照3.3.2节讨论的标准初始化过程初始化网络权值。

**步骤2** 从训练输出对的集合中提交输入模式并且计算网络响应。

**步骤3** 比较网络实际输出和期望的网络响应，并且用下式计算局部误差

对于输出层： $\delta_i^{(s)} = (d_{qh} - x_{out,i}^{(s)})g(v_i^{(s)})$

对于隐藏层： $\delta_i^{(s)} = \left(\sum_{h=1}^{n_2} \delta_h^{(s+1)} w_{hi}^{(s+1)}\right) g(v_i^{(s)})$

**步骤4** 网络权值按照下式更新

$$w_{ij}^{(s)}(k+1) = w_{ij}^{(s)}(k) + \mu^{(s)}\delta_i^{(s)} x_{out,j}^{(s)}$$

**步骤5** 按照如下等式更新激活函数的斜度

$$\gamma_i^{(s)}(k+1) = \gamma_i^{(s)}(k) + \beta\delta_i^{(s)} + \rho[\gamma_i^{(s)}(k) - \gamma_i^{(s)}(k-1)]$$

如果 $\gamma_i^{(s)}(k+1) < \gamma_{min}$，那么 $\gamma_i^{(s)}(k+1) = \gamma_{min}$。

**步骤6** 如果网络收敛则停止，否则回到步骤2。 □ 131

比较标准反向传播和上面的算法，我们发现仅仅在第5步不同，在第5步执行斜度的更新。由于局部误差已经作为一个权值更新方程的必要部分被计算，斜度更新并不添加多大的计算负担。

虽然这个方法作为标准反向传播算法的改进给出，结合其他的反向传播算法形式也可实现倾斜的自适应。另外，当激活函数为分段线性或量化，在[42, 43]中给出了自适应倾斜方法的有趣形式。从MLP NN的实际硬件实现来看，分段线性和量化的激活函数是极端重要的。

### 3.4.4 Levenberg-Marquardt 算法

Levenberg-Marquardt反向传播（LMBP）算法代表应用于训练MLP NN问题的牛顿方法的一个简单版本。牛顿方法是明确建立的具有二次收敛速度的数值最优化技术。这里，我们仅仅给出算法。在A.5.3节中提供牛顿方法的详细解释，强烈推荐读者阅读该节。

*牛顿最优化算法小结*

考虑寻找向量$\boldsymbol{w} \in \Re^{N\times 1}$最小化给定能量函数$E(\boldsymbol{w}):\Re^{N\times 1} \to \Re^{+}$的数值最优化问题。根据牛顿方法，完成这个最小化任务的迭代过程可建立如下：

步骤1 初始化向量$\boldsymbol{w} \in \Re^{N\times 1}$的分量为一些随机值。

步骤2 按照如下式子更新向量$\boldsymbol{w}$：

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \boldsymbol{H}_k^{-1}\boldsymbol{g}_k \tag{3-120}$$

其中矩阵 $\boldsymbol{H}_k^{-1} \in \Re^{N\times N}$ 表示黑塞矩阵的逆。黑塞矩阵如下：

$$\boldsymbol{H} = \begin{bmatrix} \frac{\partial^2 \boldsymbol{E}(\boldsymbol{w})}{\partial \boldsymbol{w}_1^2} & \frac{\partial^2 \boldsymbol{E}(\boldsymbol{w})}{\partial \boldsymbol{w}_1 \partial \boldsymbol{w}_2} & \cdots & \frac{\partial^2 \boldsymbol{E}(\boldsymbol{w})}{\partial \boldsymbol{w}_1 \partial \boldsymbol{w}_N} \\ \frac{\partial^2 \boldsymbol{E}(\boldsymbol{w})}{\partial \boldsymbol{w}_2 \partial \boldsymbol{w}_1} & \frac{\partial^2 \boldsymbol{E}(\boldsymbol{w})}{\partial \boldsymbol{w}_2^2} & \cdots & \frac{\partial^2 \boldsymbol{E}(\boldsymbol{w})}{\partial \boldsymbol{w}_2 \partial \boldsymbol{w}_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 \boldsymbol{E}(\boldsymbol{w})}{\partial \boldsymbol{w}_N \partial \boldsymbol{w}_1} & \frac{\partial^2 \boldsymbol{E}(\boldsymbol{w})}{\partial \boldsymbol{w}_N \partial \boldsymbol{w}_2} & \cdots & \frac{\partial^2 \boldsymbol{E}(\boldsymbol{w})}{\partial \boldsymbol{w}_N^2} \end{bmatrix} \tag{3-121}$$

132

在点$\boldsymbol{w}=\boldsymbol{w}(k)$处进行取值。向量$\boldsymbol{g}_k\in\Re^{N\times 1}$表示能量函数的梯度，计算为

$$\boldsymbol{g}=\left[\frac{\partial E(\boldsymbol{w})}{\partial w_1}\quad\frac{\partial E(\boldsymbol{w})}{\partial w_2}\quad\cdots\quad\frac{\partial E(\boldsymbol{w})}{\partial w_N}\right]^T \tag{3-122}$$

在点$\boldsymbol{w}=\boldsymbol{w}(k)$处取值。

牛顿方法的一个明显问题是计算黑塞矩阵的逆所涉及的计算需求。即使对中等规模的神经网络，算法的复杂性也限制了它的实际应用。LMBP算法提供了牛顿方法一个可行替代，它具有近似相同的收敛速度和显著减少的复杂度。为了应用LMBP算法，训练MLP NN的问题必须用一个非线性最优化问题来表示。

考虑图3-4所示的MLP NN。神经网络训练的任务可以看作训练集合中所有模式的期望与实际网络输出的误差最小的一组网络权值。如果模式数量有限，能量函数可以写成

$$E(\boldsymbol{w})=\frac{1}{2}\sum_{q=1}^{Q}(\boldsymbol{d}_q-\boldsymbol{x}_{\text{out},q}^{(3)})^T(\boldsymbol{d}_q-\boldsymbol{x}_{\text{out},q}^{(3)})=\frac{1}{2}\sum_{q=1}^{Q}\sum_{h=1}^{n_3}(d_{qh}-x_{\text{out},qh}^{(3)})^2 \tag{3-123}$$

其中$\boldsymbol{Q}$是总的训练模式总数，$\boldsymbol{w}$表示网络中包含所有权值的向量，$\boldsymbol{d}_q$是期望输出，$\boldsymbol{x}_{\text{out},q}^{(3)}$是第$q$个训练模式的实际网络输出。按照牛顿方法，使式（3-123）中能量函数最小的最优权值集合可以通过应用下式找到

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)-\boldsymbol{H}_k^{-1}\boldsymbol{g}_k \tag{3-124}$$

其中

$$\boldsymbol{H}_k=\nabla^2E(\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}(k)} \tag{3-125}$$

和

$$\boldsymbol{g}_k=\nabla E(\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}(k)} \tag{3-126}$$

通过定义$\boldsymbol{P}=n_3\boldsymbol{Q}$，式（3-123）可以改写成

$$\boldsymbol{E}(\boldsymbol{w})=\frac{1}{2}\sum_{p=1}^{P}(d_p-x_{\text{out},p}^{(3)})^2=\frac{1}{2}\sum_{p=1}^{P}e_p^2 \tag{3-127}$$

其中

133

$$e_p=d_p-x_{\text{out},p}^{(3)} \tag{3-128}$$

在式（3-126）中能量函数的梯度可以计算为

$$\boldsymbol{g}=\frac{\partial E(\boldsymbol{w})}{\partial(\boldsymbol{w})}=\frac{1}{2}\begin{bmatrix}\dfrac{\partial\sum_{p=1}^{P}e_p^2}{\partial w_1}\\[2ex]\dfrac{\partial\sum_{p=1}^{P}e_p^2}{\partial w_2}\\[2ex]\vdots\\[2ex]\dfrac{\partial\sum_{p=1}^{P}e_p^2}{\partial w_N}\end{bmatrix}=\begin{bmatrix}\sum_{p=1}^{P}e_p\dfrac{\partial e_p}{\partial w_1}\\[2ex]\sum_{p=1}^{P}e_p\dfrac{\partial e_p}{\partial w_2}\\[2ex]\vdots\\[2ex]\sum_{p=1}^{P}e_p\dfrac{\partial e_p}{\partial w_N}\end{bmatrix}=\boldsymbol{J}^T\boldsymbol{e} \tag{3-129}$$

其中$\boldsymbol{J}\in\Re^{P\times N}$是雅可比矩阵，定义如下

$$\boldsymbol{J}=\begin{bmatrix}\dfrac{\partial e_1}{\partial w_1} & \dfrac{\partial e_1}{\partial w_2} & \cdots & \dfrac{\partial e_1}{\partial w_N}\\ \dfrac{\partial e_2}{\partial w_1} & \dfrac{\partial e_2}{\partial w_2} & \cdots & \dfrac{\partial e_2}{\partial w_N}\\ \cdots & \cdots & \cdots & \cdots\\ \dfrac{\partial e_P}{\partial w_1} & \dfrac{\partial e_P}{\partial w_2} & \cdots & \dfrac{\partial e_P}{\partial w_N}\end{bmatrix} \tag{3-130}$$

下一步需要发现黑塞矩阵的表达式。黑塞矩阵中的$k,j$元素可以表示为

$$[\nabla^2 E(\boldsymbol{w})]_{k,j}=\frac{\partial^2 E(\boldsymbol{w})}{\partial w_k \partial w_j}=\sum_{p=1}^{P}\left(\frac{\partial e_p}{\partial w_k}\frac{\partial e_p}{\partial w_j}+e_p\frac{\partial^2 e_p}{\partial w_k \partial w_j}\right) \tag{3-131}$$

通过使用式（3-130）中雅可比矩阵的表达式，黑客矩阵可以表示为

$$\nabla^2 E(\boldsymbol{w})=\boldsymbol{J}^T\boldsymbol{J}+\boldsymbol{S} \tag{3-132}$$

其中矩阵$\boldsymbol{S}\in\Re^{N\times N}$是二阶导数矩阵，给出为

$$\boldsymbol{S}=\sum_{p=1}^{P}e_p\nabla^2 e_p \tag{3-133}$$

当靠近能量函数的最小值时，矩阵$\boldsymbol{S}$的元素变得很小，黑塞矩阵可以近似地表达为

$$\boldsymbol{H}\approx\boldsymbol{J}^T\boldsymbol{J} \tag{3-134}$$

把式（3-129）和式（3-134）代入式（3-124）给出的牛顿方法的表达式，得到

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)-[\boldsymbol{J}_k^T\boldsymbol{J}_k]^{-1}\boldsymbol{J}_k^T\boldsymbol{e}_k \tag{3-135}$$

134

其中下标$k$表明了对相应矩阵在$\boldsymbol{w}=\boldsymbol{w}(k)$处取值。

式（3-135）中迭代更新的一个问题是需要可能为病态或甚至奇异矩阵$\boldsymbol{H}=\boldsymbol{J}^T\boldsymbol{J}$的逆。该问题通过对式（3-134）的修改很容易解决

$$\boldsymbol{H}\approx\boldsymbol{J}^T\boldsymbol{J}+\mu\boldsymbol{I} \tag{3-136}$$

其中$\mu$是很小的数，$\boldsymbol{I}\in\Re^{N\times N}$是一个单位矩阵。把式（3-136）代入式（3-135），得到更新网络权值的Levenberg-Marquardt算法[35]

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)-[\boldsymbol{J}_k^T\boldsymbol{J}_k+\mu_k\boldsymbol{I}]^{-1}\boldsymbol{J}_k^T\boldsymbol{e}_k \tag{3-137}$$

在说明神经网络环境下式（3-137）如何实现之前，需要认识到它表示从最速下降方法到牛顿方法的过渡。对于式（3-137）中$\mu_k$的一个较小的值，它趋向于式（3-135）给出的近似牛顿算法。当$\mu_k$值增加时，式（3-137）中方括号内的第二项变成优势，并且更新方程可以写成：

$$\begin{aligned}\boldsymbol{w}(k+1)&=\boldsymbol{w}(k)-[\boldsymbol{J}_k^T\boldsymbol{J}_k+\mu_k\boldsymbol{I}]^{-1}\boldsymbol{J}_k^T\boldsymbol{e}_k\\ &\approx\boldsymbol{w}(k)-[\mu_k\boldsymbol{I}]^{-1}\boldsymbol{J}_k^T\boldsymbol{e}_k=\boldsymbol{w}(k)-\frac{1}{\mu_k}\boldsymbol{J}_k^T\boldsymbol{e}_k\end{aligned} \tag{3-138}$$

定义$\alpha_k=1/\mu_k$，使用式（3-129），式（3-138）能重新写成

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)-\alpha_k\boldsymbol{g}_k \tag{3-139}$$

它是最速下降梯度方法。

实现LMBP算法的最大问题是雅可比矩阵$\boldsymbol{J}(\boldsymbol{w})$的计算。矩阵的每一项有如下形式

$$J_{i,j}=\frac{\partial e_i}{\partial w_j} \tag{3-140}$$

计算式（3-140）中导数的最简单计算方法是使用近似

$$J_{i,j} \simeq \frac{\Delta e_i}{\Delta w_j} \tag{3-141}$$

其中$\Delta e_i$表示由于权值$\Delta w_j$的小扰动引起的输出误差的变化。该方法相对直接并且实现简单。权值的扰动保持很小，至少在幅度上比当前学习率参数$\mu_k$小一个数量级。计算雅可比矩阵以后，可以使用式（3-137）实现权值更新。

**反向传播算法的Levenberg-Marquardt形式**

135 **步骤1** 初始化网络权值为小的随机值。使用3.3.2节建议的初始化过程。设置学习率参数。
**步骤2** 提交一个输入模式，计算网络输出。
**步骤3** 使用式（3-141）计算与输入/输出对相关的雅可比矩阵的元素。
**步骤4** 当提交最后输入/输出对后，使用式（3-137）实现权值更新。
**步骤5** 如果网络收敛则停止，否则回到步骤2。 □

补充评论：

1. 这里给出的权值更新方法是LMBP算法的批量形式。算法的一个标量形式在[35]中。

2. 在式（3-137）中学习率参数$\mu_k$在训练期间可以动态修改。在网络训练的早期阶段它应该保持相对小，算法接近牛顿方法。为了阻止振荡，并且为了实现网络权值的精细调整，在网络训练的后期阶段增加学习率参数，算法接近最速下降梯度方法。

3. 在更新方程中使用的雅可比矩阵不必对于输入/输出训练对的整个集合进行计算。为了减少存储需求，在提交训练模式的子集给网络后就可以执行更新。

## 3.5 对传

对传网络由Hecht-Nielsen[44-46]开发，它扮演一个自编程的最优查看表的功能，提供一个输入和输出训练模式之间的双向映射。当网络训练速度是首要考虑的时候，它可以作为一个由反向传播训练的MLP NN的替代来使用。通常对传网络比MLP NN收敛得更快。然而，达到期望精度需要的神经元数量通常比MLP NN需要的大得多。所以，对传最普通的用途是用于开发神经计算系统的原型阶段。在使用对传开发一个神经计算系统后，该网络用MLP NN替代。在本节中将介绍对传网络的两种形式，仅有前向的（forward-only）对传和全面的（full）对传。

*仅有前向对传神经网络*

仅有前向对传网络的结构如图3-7所示。从图中可以看出，网络由一个输入、一个输出和一个隐藏层构成。有两组权值，通过两个不同训练算法来修正。连接输入与隐藏层的权值通过使用Kohonen自组织学习规则来训练，而隐藏层与输出层之间的权值通过使用Grossberg学习规则来训练。

在训练过程期间，把期望映射的例子提交给网络，也就是提交输入向量$\boldsymbol{x} \in \Re^{n\times 1}$和输出向
136 量$\boldsymbol{y} \in \Re^{m\times 1}$。在Kohonen层和Grossberg层中的权值各自单独训练。首先，网络计算输入向量与输入层和隐藏层中每个神经元的连接权值之间的距离。该距离可以如下计算

$$z_j = \mathrm{dist}(\boldsymbol{x}, \boldsymbol{w}_j) = \| \boldsymbol{x} - \boldsymbol{w}_j \|_2 = \sqrt{\sum_{k=1}^{n} (x_k - w_{jk})^2} \tag{3-142}$$

计算距离后，隐藏层中的神经元允许竞争，并且它们的输出如下设置

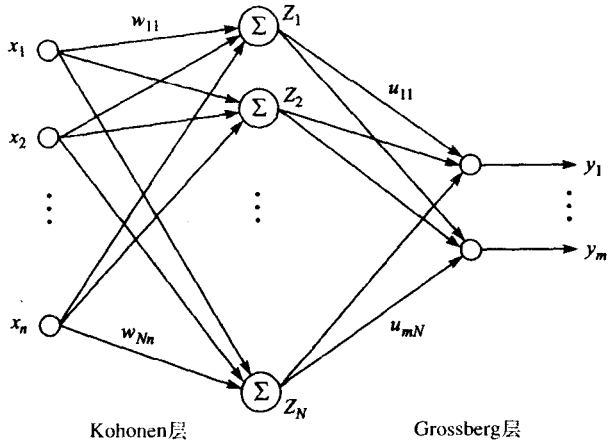



图3-7 仅有前向对传网络结构

$$z_i = \begin{cases} 1 & \text{如果}i\text{为对任何}j\text{，满足}z_i \leqslant z_j\text{的最小整数} \\ 0 & \text{其他} \end{cases} \tag{3-143}$$

换句话说，权值“最接近”于输入模式的神经元获胜，它的输出设为1。所有其他神经元输出为0。最后，连接获胜神经元的权值按照Kohonen自组织学习规则更新

$$\boldsymbol{w}_j(k+1) = [1-\alpha(k)]\,\boldsymbol{w}_j(k) + \alpha(k)\,\boldsymbol{x} \tag{3-144}$$

隐藏层的其他处理单元不调整它们的权值。学习率$\alpha$通常开始于一个相对大的值，比如，$\alpha = 0.9$，随网络训练过程逐渐减少。在Kohonen层的权值根据输入向量的统计特性分布。因而，训练的这个阶段主要执行输入向量空间的最优采样。

对传网络输出层权值按照Grossberg学习规则更新

$$u_{ji}(k+1) = u_{ji}(k) + \beta(k)\,[-u_{ji}(k) + y_j]\,z_i \tag{3-145}$$

137

其中$\beta$表示用于输出层的学习率参数。只有连接隐藏层中的获胜神经元与输出层神经元之间的权值被更新。

**训练对传网络的算法**

**步骤1** 在聚类层中选择神经元数量为$N$，初始化权值为一区间内的随机值，这个区间的范围为输入向量分量的方差。

**步骤2** 提交一个输入模式$\boldsymbol{x} \in \Re^{n\times 1}$和相应的输出目标向量$\boldsymbol{y} \in \Re^{m\times 1}$。

**步骤3** 按照如下公式计算输入向量与聚类单元的连接权值之间的距离

$$z_j = \left[\sum_{k=1}^{n}(x_k - w_{jk})^2\right]^{1/2} \qquad j = 1, 2, \cdots, N$$

**步骤4** 按照如下公式计算聚类单元的输出

$$z_i = \begin{cases} 1 & \text{如果}i\text{为对任何}j\text{满足}z_i \leqslant z_j\text{的最小整数} \\ 0 & \text{其他} \end{cases}$$

**步骤5** 按照如下公式更新Kohonen层权值

$$\boldsymbol{w}_j(k+1) = [1-\alpha(k)]\,\boldsymbol{w}_j(k) + \alpha(k)\,\boldsymbol{x}$$

其中学习率$\alpha$在训练期间逐渐减少。一个可能是按照如下公式逐渐减少$\alpha$

$$\alpha(k) = \alpha(0)\exp\left(-\frac{k}{k_0}\right)$$

其中$k_0$是适当选择的时间常数。

**步骤6** 按照如下公式更新输出层权值

$$u_{ji}(k+1) = u_{ji}(k) + \beta(k)\,[-u_{ji}(k) + y_j]\,z_i \quad j = 1, 2, \cdots, m;\ i = 1, 2, \cdots, N$$

其中$z_i$是获胜的聚类单元。

**步骤7** 如果网络收敛则停止，否则，回到步骤2。 □

**例3.3** 考虑设计一个神经网络在区间[0, 4]内逼近如下函数的问题：

$$y = \frac{1}{x+1}$$

为了完成该任务，可以使用在隐藏层有20个神经元的仅有前向的对传网络。在网络训练期间，Kohonen层学习率设置为$\alpha_0 = 0.95$，按如下指数函数逐渐降低

$$\alpha(k) = \alpha_0 \exp\left(-\frac{k}{10}\right)$$

训练网络50个回合，近似结果如图3-8所示。

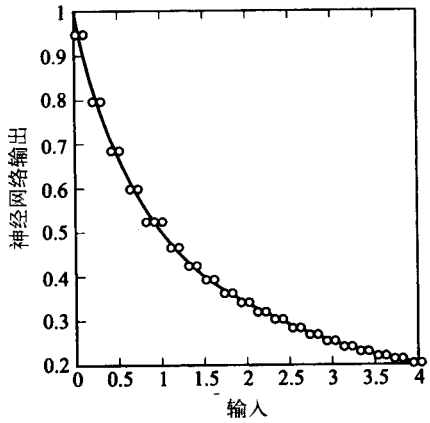



图3-8 利用仅有前向的对传神经网络对函数$y = 1/(x+1)$的逼近

全对传

仅有前向对传网络训练后提供仅一个方向的映射，全对传网络设计成学习双向映射。通过监督训练过程，网络自适应地构建一个查询表逼近已有的输入/输出训练对：$\boldsymbol{x} \in \Re^{n\times 1}$，$\boldsymbol{y} \in \Re^{m\times 1}$之间的映射。这表示仅有前向对传网络的推广。在全对传网络训练后，如果$\boldsymbol{x}$已知，可以用来重新构建相应的$\boldsymbol{y}^*$向量，反之亦然。图3-9所示全对传网络的结构。为了简单起见，图中省略了单独的权值。

138

全对传网络有四个权值集合。连接$x$和$y$输入层到聚类层的权值使用Kohonen自组织学习规则来训练，连接聚类层到两个输出层的权值使用Grossberg学习规则来训练。在算法的基本形式中，仅允许与获胜神经元相连的权值进行学习。

**训练全对传网络的算法**

**步骤1** 选择聚类层神经元的数量$N$，初始化网络权值。初始化网络权值为随机值，当：

(a) 初始化$\boldsymbol{W}$和$\boldsymbol{V}$为有界随机值，其边界为输入$\boldsymbol{x}$的分量的最大值与最小值。

(b) 初始化$\boldsymbol{U}$和$\boldsymbol{T}$为有界随机值，其边界为输入$\boldsymbol{y}$的分量的最大值与最小值。

**步骤2** 按照如下公式计算输入对$\boldsymbol{x}, \boldsymbol{y}$与聚类层单元之间的距离

$$z_i = \sum_{k=1}^{n}(x_k - w_{i,k})^2 + \sum_{k=1}^{m}(y_k - u_{i,k})^2$$

**步骤3** 按照如下公式设置聚类层神经元的激活函数

$$z_i = \begin{cases} 1 & \text{如果}i\text{为对任何}j\text{满足}z_i \leqslant z_j\text{的最小整数} \\ 0 & \text{其他} \end{cases}$$

**步骤4** 按照如下公式更新Kohonen层的权值

$$\boldsymbol{w}_i(k+1) = [1 - \alpha_x(k)]\,\boldsymbol{w}_i(k) + \alpha_x(k)\,\boldsymbol{x}$$

和

$$\boldsymbol{u}_i(k+1) = [1-\alpha_y(k)]\,\boldsymbol{u}_i(k) + \alpha_y(k)\,\boldsymbol{y}$$

其中$\alpha_x$和$\alpha_y$是学习率参数，通常随训练过程逐渐减小，并且$i$是获胜神经元的指标，即，激活函数置为1的那个神经元。

**步骤5** 按照如下公式更新Grossberg层的权值

$$v_{ji}(k+1) = v_{ji}(k) + \beta_x(k)[-v_{ji}(k) + x_j]\,z_i \quad j = 1, 2, \cdots, n;\ i = 1, 2, \cdots, N$$

和

$$t_{ji}(k+1) = t_{ji}(k) + \beta_y(k)\,[-t_{ji}(k) + y_j]\,z_i \quad j = 1, 2, \cdots, m;\ i = 1, 2, \cdots, N$$

**步骤6** 如果网络以收敛则停止；否则，回到步骤2。 □

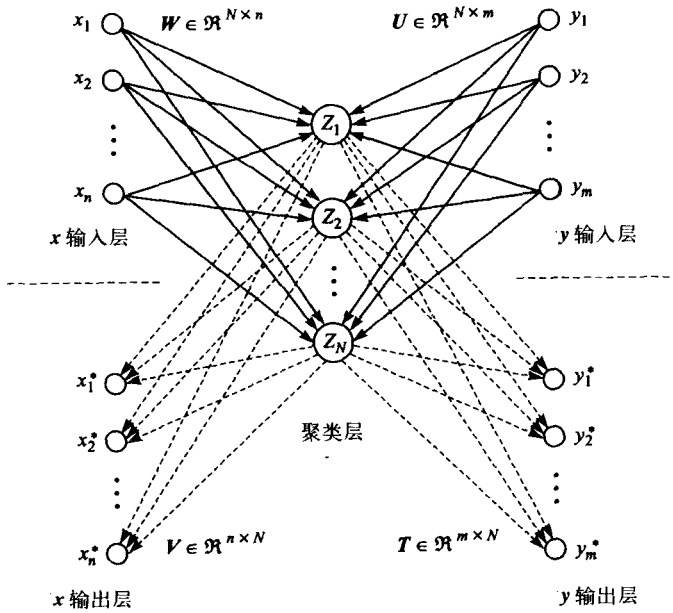



图3-9 全对传网络结构

## 3.6 径向基函数神经网络

在实践中，神经网络监督训练可以看作一个曲线拟合过程。提交训练对给网络，每个训练对由一个输入空间的向量和一个期望的网络响应组成。通过一个定义好的学习算法，网络执行它的权值调整，使相对于某个最优化准则最小化实际与期望响应之间的误差。一旦被训练后，网络实现输出向量空间中的插值，这称为泛化性能。在前面几节中，我们看见反向传播和对传网络可以训练来实现一个输入和输出向量空间之间的非线性映射。在这一节中，我们给出另一个网络，它能够完成同样的任务。这就是径向基函数网络（radial basis function neural network, RBF NN）。

139 ~ 140

RBF NN的结构如图3-10所示。网络由三层构成：一个输入层，单个非线性处理神经元层和一个输出层。RBF NN的输出按如下公式计算

$$y_i = f_i(\boldsymbol{x}) = \sum_{k=1}^{N} w_{ik}\phi_k(\boldsymbol{x}, \boldsymbol{c}_k) = \sum_{k=1}^{N} w_{ik}\phi_k(\|\boldsymbol{x} - \boldsymbol{c}_k\|_2), \quad i = 1,2,\cdots,m \tag{3-146}$$

其中$\boldsymbol{x} \in \Re^{n\times 1}$是一个输入向量，$\phi_k(\cdot)$是一个从$\Re^+$（所有正实数的集合）到$\Re$的函数。$\|\cdot\|_2$表示欧几里得范数，$w_{ik}$是输出层权值，$N$是隐藏层的神经元数目，并且$\boldsymbol{c}_k \in \Re^{n\times 1}$是输入向量空间

的RBF中心。对于隐藏层的每个神经元，计算它相关的中心和网络输入之间的欧几里得距离。隐藏层神经元的输出是距离的一个非线性函数。最后，计算网络输出为一个隐藏层输出的加权和。假设$\phi_k(\cdot)$的函数形式已经给出，一些典型选择如下[26, 47]：

1. $\phi(x)=x$　　线性函数
2. $\phi(x)=x^3$　　立方近似
3. $\phi(x)=x^2\ln x$　　薄板样条函数
4. $\phi(x)=\exp(-x^2/\sigma^2)$　　高斯函数
5. $\phi(x)=\sqrt{x^2+\sigma^2}$　　多二次函数
6. $\phi(x)=\dfrac{1}{\sqrt{x^2+\sigma^2}}$　　逆多二次函数

141

其中参数$\sigma$控制RBF的“宽度”，并且通常称为扩展参数。在实际应用中，最广泛应用的RBF是高斯RBF。

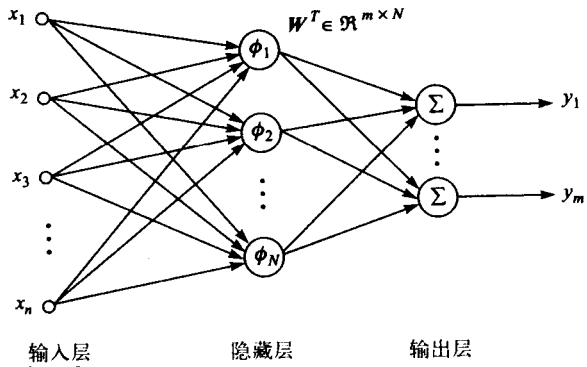



图3-10　RBF NN结构

定义中心$c_k$为那些假定实现输入向量空间的适当采样的点。它们通常选择为输入数据的一个子集。从图3-10中我们看见在输出层的神经元之间没有相互作用。基于这个原因，不失一般性，可以考虑单输出的RBF NN。有多于一个输出的网络可以当作几个单输出网络共享一个隐藏层的叠加。

### 3.6.1　训练具有固定中心的RBF NN

由式（3-146）可知，支配RBF NN映射属性的参数有两组：输出层权值$w_{ik}$和径向基函数中心$c_k$。RBF NN训练的最简单形式是有固定的中心。具体地，它们通常随机地选作输入数据集合的子集。该方法由Broomhead and Lowe[48]最早提议。方法的内在原因小结如下。从输入数据集中随机选择足够数目的中心将依据训练数据的概率密度函数进行分布，因而提供输入空间的适当采样。定性地，该方法当作一个“灵敏的”方法[29]。然而，很难确定足够的中心是多少，才能达到输入空间的适当取样。通用的方法就是选择数目相对大的输入向量为中心。这样可以保证有适当的输入空间取样。在网络训练以后，一些中心可能根据系统化的方式去除而不引起网络映射性能的显著退化。

一旦中心选定，训练数据集的输入向量对应的网络输出可以计算为

$$\hat{y}(q)=\sum_{k=1}^{N} w_{ik}\phi(\boldsymbol{x}(q),c_k),\quad q=1,2,\cdots,Q \tag{3-147}$$

其中$Q$是训练对的总数。以向量矩阵形式整理式（3-147），有

$$\begin{bmatrix}\hat{y}(1)\\ \hat{y}(2)\\ \vdots\\ \hat{y}(Q)\end{bmatrix}=\begin{bmatrix}\phi(\boldsymbol{x}(1),\boldsymbol{c}_1) & \phi(\boldsymbol{x}(1),\boldsymbol{c}_2) & \cdots & \phi(\boldsymbol{x}(1),\boldsymbol{c}_N)\\ \phi(\boldsymbol{x}(2),\boldsymbol{c}_1) & \phi(\boldsymbol{x}(2),\boldsymbol{c}_2) & \cdots & \phi(\boldsymbol{x}(2),\boldsymbol{c}_N)\\ \cdots & \cdots & \cdots & \cdots\\ \phi(\boldsymbol{x}(Q),\boldsymbol{c}_1) & \phi(\boldsymbol{x}(Q),\boldsymbol{c}_2) & \cdots & \phi(\boldsymbol{x}(Q),\boldsymbol{c}_N)\end{bmatrix}\begin{bmatrix}w_1\\ w_2\\ \vdots\\ w_N\end{bmatrix} \tag{3-148}$$

或

$$\hat{\boldsymbol{y}}=\boldsymbol{\Phi}\boldsymbol{w} \tag{3-149}$$

142

其中 $\hat{\boldsymbol{y}}\in\Re^{Q\times 1}$ 是实际网络输出向量，$\boldsymbol{w}\in\Re^{N\times 1}$是输出层权值向量，并且$\boldsymbol{\Phi}\in\Re^{Q\times N}$是RBF隐藏层实现的非线性映射的矩阵。因为中心固定，隐藏层实现的映射也固定。所以，网络训练任务是决定网络输出层权值的适当取值，以便网络映射性能在某种意义下最优。一个通常使用的最优准则是实际与期望的网络输出之间的均方误差。换句话说，权值的最优设置是最小化性能度量

$$\boldsymbol{J}(\boldsymbol{w})=\frac{1}{2}\sum_{q=1}^{Q}[y_d(q)-\hat{y}(q)]^2=\frac{1}{2}(\boldsymbol{y}_d-\hat{\boldsymbol{y}})^T(\boldsymbol{y}_d-\hat{\boldsymbol{y}}) \tag{3-150}$$

其中$\boldsymbol{y}_d\in\Re^{Q\times 1}$表示期望网络输出向量。把式（3-149）代入式（3-150）给出

$$J(\boldsymbol{w})=\frac{1}{2}(\boldsymbol{y}_d-\boldsymbol{\Phi}\boldsymbol{w})^T(\boldsymbol{y}_d-\boldsymbol{\Phi}\boldsymbol{w})=\frac{1}{2}(\boldsymbol{y}_d^T\boldsymbol{y}_d-2\boldsymbol{y}_d^T\boldsymbol{\Phi}\boldsymbol{w}+\boldsymbol{w}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{w}) \tag{3-151}$$

可由下式取得最小化性能度量$\boldsymbol{J}(\boldsymbol{w})$

$$\frac{\partial J(\boldsymbol{w})}{\partial(\boldsymbol{w})}=0 \tag{3-152}$$

或

$$-\boldsymbol{\Phi}^T\boldsymbol{y}_d+\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{w}=0 \tag{3-153}$$

求解$\boldsymbol{w}$，我们有

$$\boldsymbol{w}=(\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}_d=\boldsymbol{\Phi}^+\boldsymbol{y}_d \tag{3-154}$$

其中$\boldsymbol{\Phi}^+$表示非线性映射矩阵$\boldsymbol{\Phi}$的伪逆（参看A.2.7节）。

从式（3-154）我们看到在固定网络中心情况下，网络训练问题有一个“封闭形式”的解。这实际上意味着和反向传播网络乃至对传网络比较，RBF NN可以快速训练。它实现非线性多维插值，使用线性最小二乘算法来训练的事实使RBF NN对于各类信号处理的应用有非常大的吸引力[29，47，49-52]。

需要指出另外一个重点。依赖于RBF NN方程组（3-153）的大小，它可以是不定的，也可以为超定的，或可以有唯一的解。如果中心的数目大于或等于训练模式的数目，期望与实际的网络输出的误差可以变得任意小。事实上，如果使用式（3-154），误差将总是等于0。考虑下面的例子。

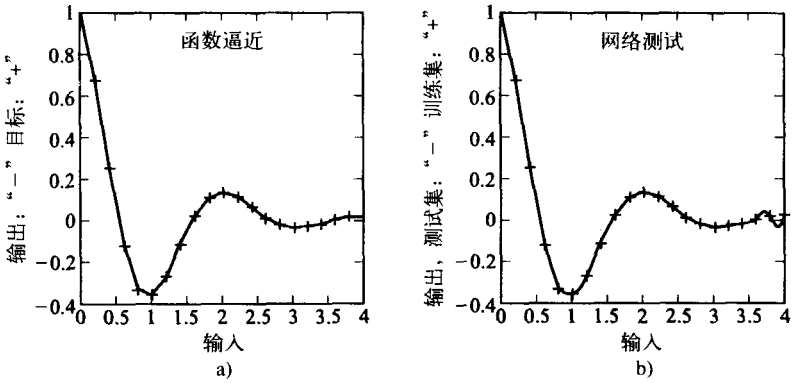
**例3.4** 训练RBF NN在区间[0, 4]内逼近非线性函数

$$y=e^{-x}\sin(3x)$$

143

它和我们使用MLP NN的例3.2中检验的任务一样。为了保证合理的比较，区间[0, 4]取样为间隔为0.2的21个点，使用MATLAB函数`trainrbfe`训练网络，隐藏单元数是21，中心对应于输入训练向量。因此，式（3-153）中方程组有唯一解。MATLAB函数`trainrbfe`使用高斯

RBF，扩展参数σ设为0.2。在训练以后，画出网络对训练模式的响应，如图3-11a所示。为了测试网络的泛化能力，同样的区域取样为401个点，点之间的间隔为0.01。这样产生测试数据集。网络对测试数据集的响应在图3-11b中给出。




a)　b)

图3-11　RBF神经网络响应a)对于训练数据；b)对于测试数据

例3.4的结果表示使用RBF NN的数据过适应的情况。在例3.2中，过适应整个毁坏了MLP NN的性能。图3-11b表明，即使当RBF NN过适应训练数据时，它至少保持可接受的性能。

*设置扩展参数*

在高斯RBF情形下，扩展参数σ通常按照如下简单启发式关系设置[29]

$$\sigma = \frac{d_{\max}}{\sqrt{K}} \tag{3-155}$$

其中$d_{\max}$是选择的中心之间最大的欧几里得距离，$K$是中心的数目。使用式（3-155），网络隐藏层的一个神经元的RBF可以写成

144

$$\phi(\boldsymbol{x},\boldsymbol{c}_k) = \exp\left(-\frac{K}{d_{\max}^2}\|\boldsymbol{x}-\boldsymbol{c}_k\|^2\right) \tag{3-156}$$

**具有固定中心的RBF NN的训练算法**

**步骤1**　为RBF函数选择中心。从输入向量集中选择中心。选择足够数量的中心以确保输入向量空间的适当取样。

**步骤2**　按照式（3-155）计算RBF函数的扩展参数σ。

**步骤3**　初始化网络输出层权值为一些小的随机值。

**步骤4**　按照式（3-149）计算神经网络的输出。

**步骤5**　使用式（3-154）求解网络权值。　□

上面描述的算法通常称为批量训练算法。矩阵Φ的伪逆在一步内得到，这意味着所有的训练数据必须事先可用。对于实时处理的情形，计算伪逆可以用一种迭代的数值程序，如最速下降、递归最小二乘、共轭梯度、牛顿方法以及7.2节给出的各种方法。

### 3.6.2　用随机梯度方法训练RBF NN

在前面小节中描述了用于训练RBF NN的过程，其中网络可调参数仅为输出层的权值。像我们看见的一样，该方法导致一个非常简单的训练算法。然而，为了实现输入的恰当取样，必须从输入数据集合中选择较大数目的中心。这产生相对大的网络，甚至对于一个简单问题

亦是如此。

用于RBF NN的随机梯度方法允许调整所有的三组网络参数（即，权值、RBF中心的位置、RBF的宽度）。所以，在隐藏层中每个处理单元的中心位置及扩展参数与输出层权值一起经历监督训练过程。开发基于随机梯度的监督训练算法的第一步是按如下公式定义瞬时误差代价函数

$$J(n)=\frac{1}{2}|e(n)|^2=\frac{1}{2}\left[y_d(n)-\sum_{k=1}^{N}w_k(n)\phi\{\boldsymbol{x}(n),\boldsymbol{c}_k(n)\}\right]^2 \tag{3-157}$$

如果RBF选择高斯型，式（3-157）变成

$$J(n)=\frac{1}{2}\left[y_d(n)-\sum_{k=1}^{N}w_k(n)\exp\left(-\frac{\|\boldsymbol{x}(n)-\boldsymbol{c}_k(n)\|_2^2}{\sigma_k^2(n)}\right)\right]^2 \tag{3-158}$$

网络参数更新方程如下

$$\boldsymbol{w}(n+1)=\boldsymbol{w}(n)-\mu_w\frac{\partial}{\partial\boldsymbol{w}}J(n)\Big|_{\boldsymbol{w}=\boldsymbol{w}(n)} \tag{3-159}$$ 145

$$\boldsymbol{c}_k(n+1)=\boldsymbol{c}_k(n)-\mu_c\frac{\partial}{\partial\boldsymbol{c}_k}J(n)\Big|_{\boldsymbol{c}_k=\boldsymbol{c}_k(n)} \tag{3-160}$$

$$\sigma_k(n+1)=\sigma_k(n)-\mu_\sigma\frac{\partial}{\partial\sigma_k}J(n)\Big|_{\sigma_k=\sigma_k(n)} \tag{3-161}$$

**训练RBF NN的基于随机梯度方法**

**步骤1** 选择RBF函数中心，从输入向量集合选择中心。

**步骤2** 按照式（3-155）计算RBF函数的扩展参数的初值。

**步骤3** 初始化网络输出层权值为一些小的随机值。

**步骤4** 提交一个输入向量，并且按下式计算网络输出

$$\hat{y}(n)=\sum_{k=1}^{N}w_k\phi\{\boldsymbol{x}(n),\boldsymbol{c}_k,\sigma_k\}$$

**步骤5** 按照下式更新网络参数

$$\boldsymbol{w}(n+1)=\boldsymbol{w}(n)+\mu_w e(n)\,\Psi(n)$$

$$\boldsymbol{c}_k(n+1)=\boldsymbol{c}_k(n)+\mu_c\frac{e(n)w_k(n)}{\sigma_k^2(n)}\phi\{\boldsymbol{x}(n),\boldsymbol{c}_k(n),\sigma_k\}[\boldsymbol{x}(n)-\boldsymbol{c}_k(n)]$$

$$\sigma_k(n+1)=\sigma_k(n)+\mu_\sigma\frac{e(n)w_k(n)}{\sigma_k^3(n)}\phi\{\boldsymbol{x}(n),\boldsymbol{c}_k(n),\sigma_k\}\|\boldsymbol{x}(n)-\boldsymbol{c}_k(n)\|^2$$

其中

$$\Psi(n)=[\phi\{\boldsymbol{x}(n),\boldsymbol{c}_1,\sigma_1\},\ \phi\{\boldsymbol{x}(n),\boldsymbol{c}_2,\sigma_2\},\ \cdots,\ \phi\{\boldsymbol{x}(n),\boldsymbol{c}_N,$$

$$e(n)=\hat{y}(n)-y_d(n)$$

$y_d(n)$是期望网络输出，$\mu_w$、$\mu_c$和$\mu_\sigma$是适当的学习率参数。

**步骤6** 如果网络已经收敛则停止，否则，回到步骤4。 □

具有更新隐藏层处理单元的中心位置及扩展参数的能力极大地提高RBF NN的性能。对于给定大小的隐藏层，与随机梯度方法一起训练的RBF NN超过一个固定中心的网络。然而，得到这样的代价增加训练算法的复杂度，这增加了训练网络需要的时间。观察上面的更新方程

组，我们注意如下[29]：

1. 瞬时误差代价函数$\boldsymbol{J}(n)$对输出层的权值是凸的。然而，对于隐藏层单元的RBF的中心位
146 置与扩展参数这并不一定是对的。这使得训练算法易于陷入局部最小值。

2. 通常，学习率参数$\mu_w$、$\mu_c$和$\mu_\sigma$设置成不同值。

3. 学习规则仍然没有反向传播复杂。因为RBF NN仅有一组可校正权值（在输出层）。误差的反向传播并不需要。

### 3.6.3 正交最小二乘

我们已经看见RBF NN设计的主要挑战是中心的选择。按随机方式选择，甚至使用随机梯度算法修改，通常导致一个相对大的网络。正交最小二乘（Orthogonal Least-Square, OLS）方法提供了用于中心选择的系统方法，显著地压缩RBF NN的大小。在介绍用于选择RBF NN中心的OLS方法之前，回顾格拉姆-施密特正交化过程的基础。

格拉姆-施密特正交化

格拉姆-施密特正交化[29, 50]用于将矩阵$\boldsymbol{M} \in \Re^{n\times m}$的分解成两个矩阵的积的一个过程，按照如下方式进行

$$\boldsymbol{M} = \boldsymbol{WA} \tag{3-162}$$

其中$\boldsymbol{A} \in \Re^{m\times m}$是一个上三角形式矩阵

$$A = \begin{bmatrix} 1 & \alpha_{12} & \cdots & \alpha_{1m} \\ 0 & 1 & \cdots & \alpha_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{3-163}$$

并且$\boldsymbol{W} \in \Re^{n\times m}$是一个有$m$个相互正交向量的矩阵，如下给出

$$\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} = \mathrm{diag}(h_1, h_2, \cdots, h_m) \tag{3-164}$$

矩阵的格拉姆-施密特正交化是建立在线性向量空间理论的基本结果基础之上的，即**正交分解定理**。该定理可以陈述如下。

**定理3.1**　任何向量$\boldsymbol{m} \in \Re^{n\times 1}$在子空间$\boldsymbol{Y} \in \Re^{n\times 1}$上可以唯一地分解成相互正交的两部分。一部分平行于子空间$\boldsymbol{Y}$（即，位于其中），另一部分与它垂直。即，

$$\boldsymbol{m} = \hat{\boldsymbol{m}} + \boldsymbol{e} \tag{3-165}$$

有$\hat{\boldsymbol{m}} \in \boldsymbol{Y}$并且$\boldsymbol{e} \perp \boldsymbol{Y}$。分量$\hat{\boldsymbol{m}}$叫作$\boldsymbol{m}$在子空间$\boldsymbol{Y}$的正交投影。

在我们的例子中，矩阵$\boldsymbol{M}$的列可以看作$n$维向量空间上的向量。使用正交化得到由矩阵$\boldsymbol{M}$
147 的列张成的子空间的正交基向量集合。

**格拉姆-施密特正交化算法**

**步骤1**　设置第一个基向量等于矩阵$\boldsymbol{M}$的第一列

$$\boldsymbol{w}_1 = \boldsymbol{m}_1$$

**步骤k**　抽取第$k$个基向量，以便它与前面$k-1$个向量正交

$$\alpha_{ik} = \boldsymbol{m}_k^T\boldsymbol{w}_i \quad 1 \leqslant i \leqslant k-1$$

$$\boldsymbol{w}_k = \boldsymbol{m}_k - \sum_{i=1}^{k-1}\alpha_{ik}\boldsymbol{w}_i$$

重复步骤$k$直到$k = m$。 □

在每一步$k$中，子空间$\boldsymbol{Y}^{(k-1)}=[\boldsymbol{w}_1\ \boldsymbol{w}_2\ \cdots\ \boldsymbol{w}_{k-1}]$用向量$\boldsymbol{w}_k$来扩展，表示$\boldsymbol{m}_k$与子空间$\boldsymbol{Y}^{(k-1)}$正交的一部分。显然，$\boldsymbol{W}=\boldsymbol{Y}^{(m)}$。矩阵$\boldsymbol{A}$的元素表示向量$\boldsymbol{m}_k$在$\boldsymbol{W}$列形成的坐标系中的坐标，$1\leqslant k\leqslant m$。

正交最小二乘回归

OLS方法起源于线性回归模型。RBF NN实现的映射可以看作如下形式的回归模型

$$\begin{bmatrix} y_d(1) \\ y_d(2) \\ \vdots \\ y_d(Q) \end{bmatrix} = \begin{bmatrix} \phi(\boldsymbol{x}_1,\boldsymbol{c}_1,\sigma_1) & \phi(\boldsymbol{x}_1,\boldsymbol{c}_2,\sigma_2) & \cdots & \phi(\boldsymbol{x}_1,\boldsymbol{c}_N,\sigma_N) \\ \phi(\boldsymbol{x}_2,\boldsymbol{c}_1,\sigma_1) & \phi(\boldsymbol{x}_2,\boldsymbol{c}_2,\sigma_2) & \cdots & \phi(\boldsymbol{x}_2,\boldsymbol{c}_N,\sigma_N) \\ \cdots & \cdots & \cdots & \cdots \\ \phi(\boldsymbol{x}_Q,\boldsymbol{c}_1,\sigma_1) & \phi(\boldsymbol{x}_Q,\boldsymbol{c}_2,\sigma_2) & \cdots & \phi(\boldsymbol{x}_Q,\boldsymbol{c}_N,\sigma_N) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_Q \end{bmatrix} \tag{3-166}$$

或

$$\boldsymbol{y}_d=\boldsymbol{\Phi}\boldsymbol{w}+\boldsymbol{e} \tag{3-167}$$

其中$\boldsymbol{y}_d\in\Re^{Q\times 1}$是期望的网络输出向量，$\boldsymbol{\Phi}\in\Re^{Q\times N}$可以看作一个线性回归矩阵，该矩阵每列向量$\boldsymbol{\phi}\in\Re^{Q\times 1}$是一个回归向量或回归量（regressor），$\boldsymbol{w}\in\Re^{N\times 1}$是权值向量，$\boldsymbol{e}\in\Re^{N\times 1}$是期望的与实际的网络输出之间的误差向量。

RBF NN的中心从输入模式集合中选择。正如在式（3-166）中看到，总共有$\boldsymbol{Q}$个候选。使用所有的$\boldsymbol{Q}$个输入模式作为中心将产生一个无误差映射的网络。然而，在大多数情况下，使用来自输入向量空间的所有$\boldsymbol{Q}$个候选的网络是非常大的。OLS回归的任务是执行$N<Q$个中心的系统选择，使得在最小降低网络性能情况下显著地削减网络的大小。 148

从式（3-166）我们看见，RBF NN的中心与回归矩阵$\Phi$的回归量之间一一对应。在OLS回归的每一步，使期望输出的方差增加最大的方式来选择一个新的中心。假设我们已选择$N<Q$个中心。最小二乘得到的权值解为

$$\hat{\boldsymbol{w}}=\boldsymbol{\Phi}^{+}\boldsymbol{y}_d \tag{3-168}$$

网络实际输出为

$$\hat{\boldsymbol{y}}=\boldsymbol{\Phi}\hat{\boldsymbol{w}}=[\boldsymbol{\phi}_1,\boldsymbol{\phi}_2,\cdots,\boldsymbol{\phi}_N]\hat{\boldsymbol{w}} \tag{3-169}$$

其中$\hat{\boldsymbol{y}}$表示$\boldsymbol{y}_d$在回归矩阵$\Phi$的列$\boldsymbol{\phi}_i$所扩张成的向量空间内的那部分。

通过使用格拉姆-施密特正交方式，回归矩阵能分解为

$$\boldsymbol{\Phi}=\boldsymbol{B}\boldsymbol{A}=[\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_N]\begin{bmatrix} 1 & a_{11} & a_{12} & \cdots & a_{1N} \\ 0 & 1 & a_{23} & \cdots & a_{2N} \\ & & \cdots & & \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{3-170}$$

其中$\boldsymbol{A}\in\Re^{N\times N}$是一个主对角线元为1的上三角矩阵，$\boldsymbol{B}\in\Re^{Q\times N}$是具相互正交列$\boldsymbol{b}_i$的矩阵，使得

$$\boldsymbol{B}^T\boldsymbol{B}=\boldsymbol{H}=\mathrm{diag}(h_1,h_2,\cdots,h_N) \tag{3-171}$$

矩阵$\boldsymbol{H}\in\Re^{N\times N}$是对角阵，其元素$h_i$由下面给出

$$h_i=\boldsymbol{b}_i^T\boldsymbol{b}_i=\sum_{k=1}^{N}b_{ik}^2 \tag{3-172}$$

正交基向量集$\boldsymbol{b}_i$所扩张成的向量空间与回归矩阵$\Phi$的列扩张成的向量空间相同。把式（3-170）代入式（3-167），我们得到

$$\boldsymbol{y}_d=\boldsymbol{B}\boldsymbol{A}\boldsymbol{w}+\boldsymbol{e}=\boldsymbol{B}\boldsymbol{g}+\boldsymbol{e} \tag{3-173}$$

其中$\boldsymbol{g} = \boldsymbol{A}\boldsymbol{w}$。在式（3-173）中，期望输出向量$\boldsymbol{y}_d$表示成矩阵$\boldsymbol{B}$的相互正交列的线性组合。坐标向量$\boldsymbol{g}$的最小二乘解为

$$\hat{\boldsymbol{g}} = (\boldsymbol{B}^T\boldsymbol{B})^{-1}\boldsymbol{B}^T\boldsymbol{y}_d = \boldsymbol{B}^+\boldsymbol{y}_d = \boldsymbol{H}^{-1}\boldsymbol{B}^T\boldsymbol{y}_d \tag{3-174}$$

向量$\hat{\boldsymbol{g}}$的第$i$个坐标是

$$g_i = \frac{\boldsymbol{b}_i^T\boldsymbol{y}_d}{\boldsymbol{b}_i^T\boldsymbol{b}_i} \tag{3-175}$$

149 并且，像我们期望的那样，它是向量$\boldsymbol{y}_d$在列$\boldsymbol{b}_i$方向的规范化投影。因为格拉姆−施密特正交化保证式（3-173）中的逼近误差与$\boldsymbol{Bg}$之间的正交，我们有

$$\boldsymbol{y}_d^T\boldsymbol{y}_d = \boldsymbol{g}^T\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{g} + \boldsymbol{e}^T\boldsymbol{e} = \boldsymbol{g}^T\boldsymbol{H}\boldsymbol{g} + \boldsymbol{e}^T\boldsymbol{e} = \sum_{i=1}^{N} h_i g_i^2 + \boldsymbol{e}^T\boldsymbol{e} \tag{3-176}$$

给出式（3-176）的物理解释是相对容易的。项$\boldsymbol{y}_d^T\boldsymbol{y}_d$表示期望输出向量的总能量。项$\sum_{i=1}^{N} h_i g_i^2$表示回归所解释的能量部分，而$\boldsymbol{e}^T\boldsymbol{e}$是回归误差能量。式（3-176）右边总和中的每一项表示由于包含第$i$个回归向量能量的增长。由于存在回归向量$\boldsymbol{g}_i$的分量与RBF中心$\boldsymbol{c}_i$之间的一一对应，在求和中的每一项反映每个RBF中心的贡献。可以定义由于包含第$p$个RBF中心的误差缩减率（err）为

$$[\text{err}]_p = \frac{h_p g_p^2}{\boldsymbol{y}_d^T\boldsymbol{y}_d} \tag{3-177}$$

式（3-177）中的误差缩减率为前向回归方式的RBF中心选择提供一个简单和有效的标准。在前向回归的每一步中，选择一个RBF中心使误差缩减率最大。传统的格拉姆−施密特正交化过程可以合并到前向回归设计，整个过程可以小结为如下算法。

**训练RBF网络的正交最小二乘算法**

**步骤1**　$k = 1$。对于$1 \leqslant i \leqslant Q$,设置

$$\boldsymbol{b}_1^{(i)} = \boldsymbol{\phi}_i$$

计算第$i$个中心的误差缩减率为

$$[\text{err}]_1^i = \frac{\left(\boldsymbol{b}_1^{(i)T}\boldsymbol{y}_d\right)^2}{\boldsymbol{b}_1^{(i)T}\boldsymbol{b}_1^{(i)} \cdot \boldsymbol{y}_d^T\boldsymbol{y}_d}$$

寻找

$$[\text{err}]_1^{i_1} = \max\{[err]_1^i,\ 1 \leqslant i \leqslant Q\}$$

选择

$$\boldsymbol{b}_1 = \boldsymbol{\phi}_{i_1}$$

并且中心$c_1 = c_{i_1}$。

**步骤k**　$k \geqslant 2$。对于$1 \leqslant i \leqslant Q, i \neq i_1, i \neq i_2, \cdots, i \neq i_{k-1}$计算

150

$$a_{jk}^{(i)} = \frac{\boldsymbol{b}_j^T\boldsymbol{\phi}_i}{\boldsymbol{b}_j^T\boldsymbol{b}_j} \quad 1 \leqslant j \leqslant k-1$$

设

$$\boldsymbol{b}_k^{(i)} = \boldsymbol{\phi}_i - \sum_{j=1}^{k-1} a_{jk}^{(i)}\boldsymbol{b}_j$$

计算

$$[\text{err}]_k^{(i)} = \frac{(\boldsymbol{b}_k^{(i)T}\boldsymbol{y}_d)^2}{\boldsymbol{b}_k^{(i)T}\boldsymbol{b}_k^{(i)} \cdot \boldsymbol{y}_d^T\boldsymbol{y}_d}$$

寻找

$$[\text{err}]_k^{i_k} = \max\{[\text{err}]_k^i, 1 \leqslant i \leqslant Q, i \neq i_1, i \neq i_2, \cdots, i \neq i_{k-1}\}$$

选择

$$\boldsymbol{b}_k = \boldsymbol{b}_k^{(i_k)}$$

并且中心 $\boldsymbol{c}_k = \boldsymbol{c}_{i_k}$ 。

**步骤k + 1** 重复步骤k，当

$$1 - \sum_{j=1}^{N_1} [\text{err}]_j < \rho$$

在步骤$N_1$停止回归，其中$0 < \rho < 1$是选择的容许值。 □

上面过程的几何学解释可以陈述如下。在第$k$步，由选择的回归量扩张成的空间维数通过引入一个另外的基向量从$k-1$增加到$k$。最新添加的向量最大化期望网络输出的能量（即，最大化误差缩减率）。在回归中包括的每个向量对应于输入数据点集合中的一个中心。当回归已经包含足够部分的期望网络输出的能量时，新向量的增加（和新中心的选择）停止。因而，OLS方法产生一个相对小的网络。

容许参数$\rho$对于平衡网络的精度和复杂度是很重要的。如果$\rho$设置得太高（即，靠近1），结果网络将以高精度逼近映射，但它导致很大数目的中心。而且，得到的精度很可能导致过拟合。另一方面，设置$\rho$很小将导致网络的相对差的建模性质。但是，网络规模将极大缩减。设置$\rho$的合理方法是 $\rho = 1 - \sigma_n^2/\sigma_d^2$ ，其中 $\sigma_n^2$ 是度量噪声功率的某种估计，$\sigma_d^2$是目标信号的总功率。

本节介绍的OLS算法形式应用于有单一输出的RBF NN。多输出网络的扩展是相对直接的。

**例3.5** 考虑设计一个RBF NN逼近映射$z = f(x, y)$的任务。

$$z = \cos(3x)\sin(2y)$$

考虑区域为：$-1 \leqslant x \leqslant 1$并且$-1 \leqslant y \leqslant 1$。为了完成这个任务，使用一个在定义的输入空间中分布的121个中心的RBF NN，如图3-12中用+号表示。使用扩展参数设置为0.3的高斯RBF函数，训练网络实现所有121个中心的精确映射。利用OLS算法执行具有$\rho = 0.99$期望精度的网络规模缩减。网络保留28个中心，在图3-12中用“o”表示。由于RBF NN映射的性能的最小限度的降低，网络的规模已经缩减了$1 - 28/121 \approx 77\%$。

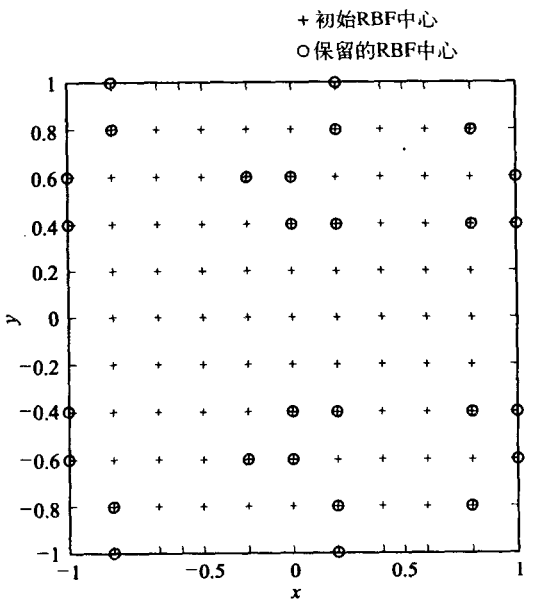



图3-12 用于设计在$-1 \leqslant x \leqslant 1$和$-1 \leqslant y \leqslant 1$区域内的映射$z = \cos(3x)\sin(2y)$的RBF中心。应用OLS算法后，缩减网络仅保留28个中心

## 习题

3.1 考虑3.2节中描述的联想记忆网络。通常，关键向量和记忆向量的维数并不一定相同。考虑下面的输入关键向量

$$x_1 = [1\ 0\ 0\ 0]^T, x_2 = [0\ 1\ 0\ 0]^T, x_3 = [0\ 0\ 0\ 1]^T$$

输出联想向量

$$y_1 = [2\ -2\ 3]^T \quad y_2 = [1\ 1\ -2]^T \quad y_3 = [-2\ 4\ 2]^T$$

(a) 使用外积规则来计算（矩形）记忆矩阵 $\hat{M}$。

(b) 得到的记忆联想完全吗？可以通过使用每个关键模式为网络输入来决定，也就是说，

$$\hat{y}_i = \hat{M}x_i$$

152　并且把估计 $\hat{y}_i$ 与记忆向量进行比较。

3.2　在某些情况下，网络的输入模式可以变得扭曲或模糊（masked）。例如在例3.1中，式（3-15）给出的第一个关键向量由于某些原因可能有第一个元素为零输出，导致初始关键向量$x_1$的一个模糊版本$x_{1m}$，即，

$$x_{1m} = [0\ \ 0.7778\ \ 0.5329]^T$$

使用式（3-19）的记忆矩阵计算$x_{1m}$的记忆响应。然后计算该响应与式（3-15）中原始的关键向量之间的欧几里得距离。你得出什么结论呢？当式（3-24）中的第一个关键向量的第一个元素有零输出时执行同样的分析，也就是说，模糊向量是

$$x_{1m} = [0\ \ -0.9779\ \ -0.1629]^T$$

而现在使用式（3-28）的记忆矩阵。从该结果中得到什么结论呢？此外，把这些结果同该问题第一部分的那些进行比较，如果有，可以得到什么结论？

M　3.3　再次考虑问题3.1，使用式（3-37）的误差修正递归算法计算记忆矩阵，写一个MATLAB函数实现递归算法，在自变量列表中包含学习率参数$\mu$，迭代总次数$N$，和一个容许值tol，它定义什么时候达到收敛。具体地，当下列不等式成立时你的程序里的“停止标准”数学上应该满足

$$\|Y - \hat{M}X\|_2 \leqslant tol$$

其中$\|\cdot\|_2$是矩阵 $Y - \hat{M}X$ 的最大奇异值（参见A.2.13节），$Y$包含所有存储模式作为列向量，$X$包含所有关键模式作为列向量，$\hat{M}$是一个矩形记忆矩阵。MATLAB函数作为一个“m文件”（m-file）以如下形式开始

```
function  M = corrmm(X,Y,mu,N,tol)
```

把你的结果与问题3.1得到的结果相比较。用不同的学习率参数$\mu$和容许值tol来试验。tol的合理值是$10^{-8}$。

M　3.4　异联想记忆神经网络可以用来检测噪声存在下的已知序列。作为说明，设计一个异联想网络存储下面的向量对：

$$x(1) = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]^T \qquad y(1) = [1\ \ -1]^T$$
$$x(2) = [1\ -1\ -1\ 1\ \ 1\ \ -1\ -1\ 1]^T \qquad y(2) = [1\ \ 1]^T$$
$$x(3) = [1\ -1\ 1\ \ -1\ -1\ 1\ \ -1\ 1]^T \qquad y(3) = [-1\ \ 1]^T$$

使用图3-13中的结构和式（3-7）的学习规则。

(a) 当输入数据被零均值高斯噪声损坏时，测试网络的性能。用不同的噪声功率进行实验。

(b) 存储一个附加联想向量对，定义如下

153

$$x(4) = [-1\ 1\ 1\ -1\ 1\ 1\ 1\ 1]^T \quad y(4) = [-1\ \ -1]^T$$

当输入数据被零均值且标准偏差为0.3的白色噪声损坏时，测试网络性能。解释对于部分

(a) 的性能退化。

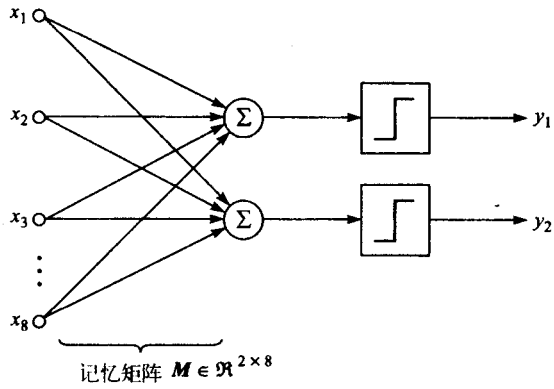



图3-13 问题3.4的异联想记忆神经网络

3.5 使用反向传播训练的MLP NN的最普通应用之一是非线性函数映射的逼近。写一个计算机程序，并且设计一个带有一个隐藏层的MLP NN，由反向传播训练来实现如下映射：

1. $f(x)=\dfrac{1}{x}$ $x\in(0.1, 1)$

2. $f(x, y) = x^2 + y^2$ $x\in(-1, 1)$并且$y\in(-1, 1)$

3. $f(x, y) =\sin(\pi x)\cos(\pi x)$ $x\in(-2, 2)$并且$y\in(-2, 2)$

4. $f(x,y,z)=\dfrac{x^2}{2}+\dfrac{y^2}{3}+\dfrac{z^2}{4}$ $x\in(-2, 2), y\in(-2, 2), z\in(-2, 2)$

对于上面的每个例子，执行如下操作：

(a) 产生三个输入模式的独立集合：

训练集合：200个模式

测试集合：100个模式

检验集合：50个模式

对于每个集合，使用被逼进函数的解析表达式来产生目标值。

(b) 在网络训练过程中，使用训练数据集合修改权值。在每次训练回合结束时利用测试数据集合监测网络的泛化能力并阻止过适应。最后，使用检验集合验证网络训练后的全面性能。

(c) 对隐藏层不同数量的神经元进行试验。

(d) 比较当权值初始化为小的随机数和Nguyen-Widrow初始化过程时网络的收敛速度（参考3.3.2节）。

154

3.6 写一个计算机程序实现带有动量更新的反向传播学习算法。使用你的程序训练一个在隐藏层有10个神经元的MLP NN来实现非线性映射，该映射定义如下：

$$f(x_1,x_2)=\begin{cases}+1 & 若\ x_1^2+x_2^2\leqslant 1\\ -1 & 若\ x_1^2+x_2^2>1\end{cases}$$

其中$-2<x_1<2$和$-2<x_2<2$

(a) 作为训练集合使用441个数据点，定义为：

$$\boldsymbol{x}=(x_i, x_j)$$

其中$x_i=-2+i\cdot 0.2$ $i=0, 1,\cdots, 20$

$x_j = -2 + j \cdot 0.2 \quad j = 0, 1, \cdots, 20$

(b) 使用以下定义的遗忘因子的值试验：

$$\alpha \in \{0, 0.2, 0.4, 0.6, 0.8, 0.9, 0.99\}$$

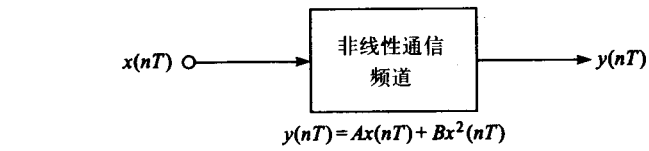
对于遗忘因子的每个值，训练网络50次，并记录需要用于网络收敛的回合数的平均值。实现非线性映射具有如下设定的精度

$$\frac{1}{441}\sum_{i=0}^{40}\sum_{j=0}^{40}[f_{NN}(x_i, x_j) - f(x_i, x_j)]^2 \leqslant 0.1$$

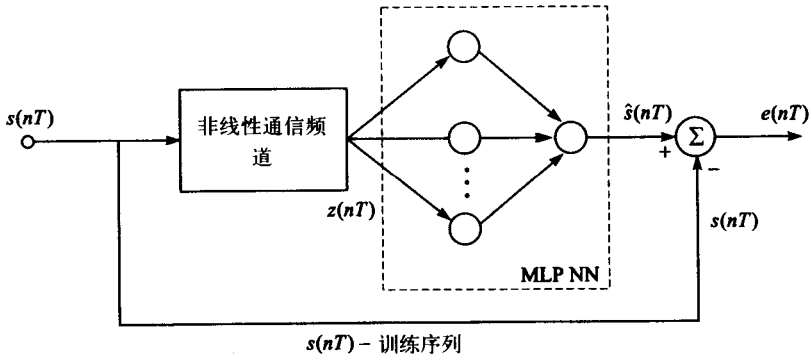
初始化权值为小的随机数。

(c) 使用Nguyen-Widrow初始化程序重复试验（参考3.3.2节）。

3.7 反向传播训练的MLP NN可以成功应用到无记忆的通信频道均衡问题。考虑图3-14a描绘的情形。频道引入的非线性失真可能导致信号传输质量的退化。如图3-14b所示，可以使用MLP NN作为消除非线性的自适应滤波器。假定通信频道的非线性输入/输出关系可以近似如下：




a)




b)

图3-14　使用MLP NN使非线性频道均衡

$$y(nT) = Ax(nT) + Bx^2(nT)$$

其中$A = 1, B = 0.2$。

(a) 设计并训练具有1个输入，7个隐藏神经元和1个输出神经元的MLP NN来实现通信频道的均衡。设置$T = 1$，使用如下信号作为训练序列

$$s(nT) = s(n) = 2\sin\left(\frac{2\pi n}{20}\right) \quad n = 1, 2\cdots$$

(b) 使用如下测试信号测试均衡器的性能

$$s_1(nT) = s_1(n) = 0.8\sin\left(\frac{2\pi n}{10}\right) + 0.25\cos\left(\frac{2\pi n}{25}\right)$$ 并且

155　$s_2(nT) = s_2(n)$是一个满足零均值和单位方差的正态分布的随机数序列。

(c) 假定频道输入/输出关系近似如下，重复试验，

$$y(nT) = Ax(nT) + Bx^2(nT) + Cx^3(nT)$$

其中$A = 1, B = 0.3, C = -0.2$。对MLP NN隐藏层神经元的不同数目进行试验。

3.8 多层感知器神经网络可以用来实现数据有损压缩。考虑图3-15所示的神经网络结构。该网络包含两层：一个隐藏层和一个输出层。隐藏层神经元的数目比输入向量的维数小，并且它实现由输入数据压缩到一个较低维数向量空间。用作输入模式的向量作为目标模式。所以，训练网络输出层从低维数表示中重构输入数据。

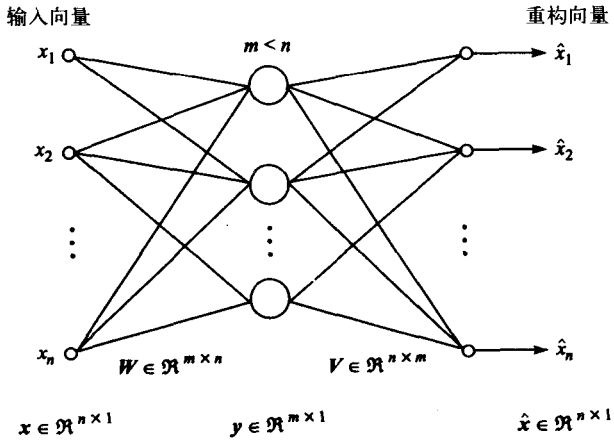



图3-15 使用MLP NN压缩数据

网络训练以后，存储隐藏层的输出代替输入数据以实现压缩。并且，为了重新构造数据，输出层权值也需要存储。让$\Psi_1(\cdot)$和$\Psi_2(\cdot)$分别是在隐藏层和输出层的神经元实现的非线性映射。在网络训练以后，压缩步骤完成如下

$$y = \Psi_1(Wx)$$

原始信号的重构实现如下

$$\hat{x} = \Psi_2(Vy)$$

156

写一个使用由反向传播训练的MLP NN来实现图像压缩的计算机程序。为了产生输入/目标向量$x$，把图像分割成$8 \times 8$像素块，重新排列每块的元素到一个64维向量中。对隐藏层神经元的不同数目进行试验。

3.9 伪随机序列有时称为伪噪声（PN）序列，因为它们有类似噪声的性质，广泛用在通信系统中，例如，在扩频系统中。这些二进制序列拥有许多有趣的性质，且可以使用一个线性移位寄存器（LSR）产生。使用长度$n = 5$的LSR（例如图3-16）可以产生有六个不同最大长度的PN序列。通常，一个最大长度序列长度为$N = 2^n - 1$，其中$2^{n-1} - 1$个为0，其余位是1。如果用于产生序列LSR上的抽头位置与本原多项式（primitive polynomial）相关，则PN序列将是最大长度。六个本原多项式是：

$$g_1(x) = x^5 + x^3 + 1 \qquad g_4(x) = x^5 + x^4 + x^3 + x^2 + 1$$
$$g_2(x) = x^5 + x^2 + 1 \qquad g_5(x) = x^5 + x^4 + x^3 + x + 1$$
$$g_3(x) = x^5 + x^3 + x^2 + x + 1 \qquad g_6(x) = x^5 + x^4 + x^2 + x + 1$$

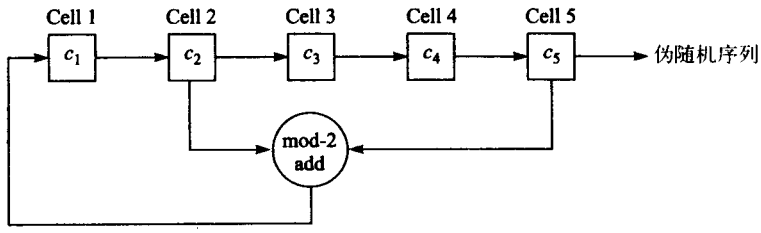
六个本原多项式的向量表示如下

$$g_1 \rightarrow [0\ 1\ 0\ 0\ 1] \qquad g_4 \rightarrow [1\ 1\ 1\ 0\ 1]$$

$g_2 \rightarrow [0\ 0\ 1\ 0\ 1]$　　$g_5 \rightarrow [1\ 1\ 0\ 1\ 1]$

157

$g_3 \rightarrow [0\ 1\ 1\ 1\ 1]$　　$g_6 \rightarrow [1\ 0\ 1\ 1\ 1]$




$c_i$s：移动寄存器初始条件（不能全为零）

图3-16　配置线性移位寄存器以利用本原多项式$g_1(x) = x^5 + x^3 + 1$产生最大长度的伪随机序列

图3-17显示一个MATLAB函数lprs，对给定的本原多项式系数向量和一个初始条件向量将产生一个最大长度的伪随机序列。该初始条件向量可以是1和0组成的但不全是0的任何恰当长度的序列（典型地，所有值选择为1）。

```
function z=lprs(n,ppc,b)
% z:     output binary sequence
% n:     number of tubes in shift register
%        (or length of shift register)
% ppc:   primitive polynomial coefficients
%        (n-dimensional row vector)
% b:     initialization vector
%        (n-dimensional row vector)
M=zeros(n,n);
M(:,1)=ppc';
for i=1:n-1
    M(i,i+1)=1;
end
z(1)=b(1);
for k=1:2^n-2
    b=b*M;
    for i=1:n
       if b(i)/2 == fix(b(i)/2)
          b(i)=0;
       else
         b(i)=1;
       end
    end
z(k+1)=b(1);
end
```

图3-17　产生最大长度的伪随机序列的MATLAB函数，该序列被给定一个本原多项式系数向量和一个初始化条件向量

(a) 生成6 PN序列（长度：$N = 2^n - 1|_{n=5} = 31$），使用上面给定的六个本原多项式的向量表达式和图3-17的MATLAB函数。

(b) 使用部分（a）中的六个向量作为前馈多层感知器的训练输入，令目标向量为本原多项式系数的相关向量表示。使用MATLAB神经网络工具箱中的trainbp。一个隐藏层足够实现该映射。用隐藏层中不同数量的神经元和激活函数试验。当网络训练后，提交六个输入并观察输出。神经网络是否能正确分类输入？

158

(c) 添加一个随机噪声到用于训练网络的六个输入向量，也就是，随机“跳转”向量中的一些位。使用这些作为在本问题部分（b）中你训练的网络的输入。观察网络的误差修正能力。输入向量如何被噪声损坏的例子显示在下面（图3-18）MATLAB函数errors中。用不同位误差率（BER）和隐藏层不同数目的神经元评价网络的误

差修正能力。在Ham et al.[53]写的论文中，使用两个无监督神经网络分类伪随机序列。

```
function ZC=errors (Z,BER)
% Z:     matrix containing input
%        patterns in columns
% BER:   bit-error-rate given in a percentage,
%        for example, BER=30 means that on the
%        average 30% of the bits will be
%        ''toggled''
% ZC:    Corrupted output matrix
[nr,nc]=size(Z);
RN=rand(size(Z));
ZC=Z;
T=1-BER/100;
for i=1:nr
     for j=1:nc
         if RN(i, j)>=T
            if ZC(i, j)==1
               ZC(i, j)=0;
            else
               ZC(i, j)=+1;
            end
         end
     end
end
```

图3-18 产生位误差的MATLAB函数errors

(d) 对生成PN序列的不同初始条件进行试验。记住序列是周期的，也就是说，对每$N = 2^n - 1$个位，它们将重复二值模式。试对六个不同最大长度序列的每一个，尝试不同的初始条件，且用导出的模式训练网络。评论结果。

3.10 考虑圆周上的点相对于直角坐标系四个象限的分类问题，该圆定义为：

$$x^2 + y^2 = 1$$

换句话说，

$$\text{点}z=(x, y)\text{分类为}\begin{cases}1 & \text{若}x>0\text{且}y>0\\ 2 & \text{若}x<0\text{且}y>0\\ 3 & \text{若}x<0\text{且}y<0\\ 4 & \text{若}x>0\text{且}y<0\end{cases}$$

159

(a) 使用在隐藏层含有36个神经元的仅有前向对传的神经网络来完成分类。

(b) 3.5节中表明如何使用欧几里得距离计算对传的Kohonen层的距离。使用距离的点积度量重复上面陈述的分类问题，距离的点积度量为

$$\text{dist}(\boldsymbol{z}, \boldsymbol{w}_j) = 1 - \frac{xw_{j1} + yw_{j2}}{\sqrt{x^2 + y^2}\sqrt{w_{j1}^2 + w_{j2}^2}}$$

(c) 使用欧几里得距离和距离的点积度量测试对于来自如下集合的输入网络的分类性能

$$\{z=(x, y):\ x^2 + y^2 < 1\}$$

解释结果的任何差异。

3.11 使用式（3-158）～式（3-161）导出用于训练RBF神经网络的随机梯度方法的学习规则。

3.12 设计一个RBF NN逼近如下定义的映射：

$$f(x_1, x_2) = \begin{cases}+1 & \text{若 } x_1^2 + x_2^2 \leqslant 1\\ -1 & \text{若 } x_1^2 + x_2^2 > 1\end{cases}$$

区域为 $-2 < x_1 < 2$, $-2 < x_2 < 2$

作为一个训练集合，使用如下定义的441个数据点

$$\boldsymbol{x} = (x_i, x_j)$$

其中

$$x_i = -2 + i \cdot 0.2 \qquad i = 0, 1, \cdots, 20$$
$$x_j = -2 + j \cdot 0.2 \qquad j = 0, 1, \cdots, 20$$

(a) 使用训练集合的所有点为RB函数的中心，实现RBF NN的设计。

(b) 使用从输入数据中随机选择的150个中心实现RBF NN的设计。把该网络的性能和部分（a）设计的网络性能相比较。

(c) 随机选择150个中心并且使用随机梯度方法实现RBF NN的设计。把该网络的性能相对于部分（a）和（b）的网络性能做比较。

(d) 使用整个数据集合作为RB函数的中心设计一个RBF NN。应用正交最小二乘过程压缩网络规模。描绘出通过OLS回归选择的中心，并且把它们同部分（c）中得到中心相比较。

3.13　使用一个RBF神经网络重复问题3.7。讨论使用两种神经网络结构的优点和缺点。

3.14　经常将RBF神经网络用于非线性动态系统的辨识。考虑图3-19描绘的情形。非线性设备实现输入序列$x(k)$与输出序列$y(k)$之间的映射。基于以前的输入和输出序列值训练神经网络，以预报非线性系统的输出。训练后，神经网络辨识非线性系统。

160

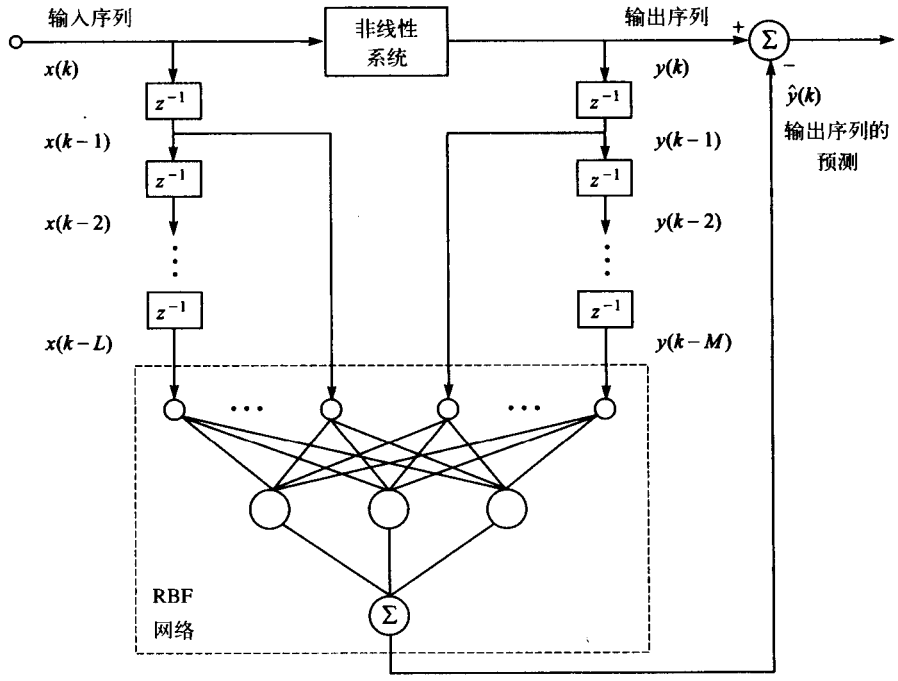



图3-19　对非线性系统辨识采用RBF神经网络

让我们假定图3-19中的非线性设备可以用如下的输入/输出差分方程来表征

$$y(k) = \frac{1}{1 + y(k-1)^2} + 0.25x(k) - 0.3x(k-1)$$

(a) 设计一个RBF神经网络执行非线性设备的辨识。

(b) 实现批学习方法。对于一个输入序列，使用零均值和单位标准偏差的白色噪声。

(c) 使用如下测试信号测试辨识的结果：

$$x_1(k)=\sin\left(\frac{2\pi k}{10}\right) \qquad x_2(k)=4\sin\left(\frac{2\pi k}{10}\right)$$

解释结果。

(d) 使用式（3-159）～式（3-161）的学习规则重新解这个问题。 161

## 参考文献


1. T. J. Teyler, "Memory: Electrophysiological Analogs," in *Learning and Memory: A Biological View*, eds. J. L. Martinez, Jr., and R. S. Kesner, Orlando, FL: Academic Press, 1986, pp. 237–65.
2. T. Kohonen, *Self-Organization and Associative Memory*, 3rd ed., Berlin: Springer-Verlag, 1989.
3. T. Kohonen, *Associative Memory: A System-Theoretical Approach*, Berlin: Springer-Verlag, 1977.
4. K. Nakano, "Association—A Model of Associative Memory," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-2, 1972, pp. 380–8.
5. J. A. Anderson, "A Simple Neural Network Generating an Interactive Memory," *Mathematical Bioscience*, vol. 14, 1972, pp. 197–220. Reprinted in 1988, Anderson and Rosenfeld, pp. 181–92. J. A. Anderson and E. Rosenfeld, eds., *Neurocomputing: Foundations of Research*, Cambridge, MA: M.I.T. Press, 1988.
6. S. Amari, "Learning Patterns and Pattern Sequences by Self-Organizing Nets of Threshold Elements," *IEEE Transactions on Computers*, vol. C-21, 1972, pp. 1197–1206.
7. S. Amari and K. Maginu, "Statistical Neurodynamics of Associative Memory," *Neural Networks*, vol. 2, 1988, pp. 63–73.
8. B. Kosko, "Bidirectional Associative Memories," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-18, 1988, pp. 49–60.
9. S. Amari, "Characteristics of Sparseiy Encoded Associative Memories," *Neural Networks*, vol. 3, 1989, pp. 451–7.
10. M. Hassoun, ed., *Associative Neural Memories: Theory and Implementation*, Oxford, England: Oxford University Press, 1993.
11. J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," in *Proceedings of National Academy of Science, USA*, vol. 79, 1982, pp. 2554–8.
12. T. Kohonen, "Correlation Matrix Memories," *IEEE Transactions on Computers*, vol. C-21, 1972, pp. 353–9. Reprinted in 1988, Anderson and Rosenfeld, pp. 174–80.
13. A. D. Whalen, *Detection of Signals in Noise*, New York, Academic Press, 1971.
14. C. W. Therrien, *Discrete Random Signals and Statistical Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1992.
15. J. A. Anderson, "Cognitive and Psychological Computation with Neural Models," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, 1983, pp. 799–815.
16. J. A. Anderson and G. L. Murphy, "Concepts in Connectionist Models," *Neural Networks for Computing*, ed. J. S. Denker, New York: American Institute of Physics, 1986, pp. 17–22.
17. P. J. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," Ph.D. thesis, Cambridge, MA: Harvard University, 1974.

18. P. J. Werbos, "Backpropagation through Time: What It Does and How to Do It," *Proceedings of the IEEE*, vol. 78, 1990, pp. 1550–60.
19. P. J. Werbos, *The Roots of Backpropagation*, New York: Wiley, 1994.
20. D. B. Parker, *Learning-Logic*, Invention Report: S81-64, File 1, Stanford, CA: Office of Technology Licensing, Stanford University, October 1982.
21. D. B. Parker, "Optimal Algorithms for Adaptive Networks: Second Order Back Propagation, Second Order Direct Propagation, and Second Order Hebbian Learning," in *Proceedings of the First IEEE International Conference On Neural* 162 *Networks*, June 1987, vol. 2, San Diego, CA, pp. 593–600.
22. D. B. Parker, "Learning-Logic: Casting the Cortex of the Human Brain in Silicon," Technical Report TR-47, Cambridge, MA: Center for Computational Research in Economics and Management Sciences, M.I.T., 1985.
23. Y. LeCun, "Une procedure d'apprentissage pour reseau à seuil asymmetrique," in *In Cognitiva 85: A la Frontiere de l'Intelligence Artificielle des Sciences de la Connaissance des Neurosciences*, CESTA, vol. 85, Paris, 1985, pp. 599–604.
24. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing*, vol. 1, chap. 8, eds. D. E. Rumelhart and J. L. McClelland, Cambridge, MA: M.I.T. Press, 1986, pp. 318–62.
25. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-Propagating Errors," *Nature*, vol. 323, 1986, pp. 533–6.
26. A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, New York: Wiley, 1993.
27. D. Nguyen and B. Widrow, "Improving the Learning Speed of the 2-Layer Neural Networks by Choosing Initial Values of Adaptive Weights," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 3, San Diego, CA, 1990, pp. 21–26.
28. K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Networks Are Universal Approximators," *Neural Networks*, vol. 2, no. 5, 1989, pp. 359–66.
29. S. Haykin, *Adaptive Filter Theory*, 3rd ed., Upper Saddle River, NJ: Prentice-Hall, 1996.
30. F. M. Ham, "Detection and Classification of Biological Substances Using Infrared Absorption Spectroscopy and Hybrid Artificial Network," *Journal of Artificial Neural Networks*, vol. 1, no. 1, 1994, pp. 101–4.
31. K. S. Narendra, "Neural Networks for Identification and Control," *Series of Lectures Presented at Yale University*, December 1994.
32. T. P. Vogl, J. K. Mangis, A. K. Zigler, W. T. Zink, and D. L. Alkon, "Accelerating the Convergence of the Backpropagation Method," *Biological Cybernetics*, vol. 59, 1988, pp. 257–63.
33. C. Darken and J. Moody, "Towards Faster Stochastic Gradient Search," in *Advances in Neural Information Processing Systems*, vol. 4, San Mateo, CA: Morgan Kaufmann, 1991, pp. 1009–16.
34. C. Darken, J. Chang, and J. Moody, "Learning Rate Schedules for Faster Stochastic Gradient Search," in *Proceedings of Neural Network Signal Processing, 1992 IEE Workshop*, New York: IEEE Press, 1992.
35. M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*, Boston, MA: PWS Publishing Company, 1996.
36. R. Battit, "First and Second Order Methods of Learning: Between the Steepest Descent and Newton's Method," *Neural Computation*, vol. 4, no. 2, 1991, pp. 141–66.
37. M. F. Moller, "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning," PB-339 Reprint, Computer Science Department, University of Aaurhus, Denmark, 1990.

38. E. Barnard and R. Cole, "A Neural-Net Training Program Based on Conjugate Gradient Optimization," Beaverton, OR: Oregon Graduate Institute, Report CSE 89-014, 1988.
39. R. Rosario, "A Conjugate-Gradient Based Algorithm for Training of the Feed-Forward Neural Networks," Ph.D. dissertation, Melbourne: Florida Institute of Technology, September 1989.
40. R. S. Scalero, "A Fast New Algorithm for Training of the Feed Forward Neural Networks," Ph.D. dissertation, Melbourne: Florida Institute of Technology, September 1989.
41. R. S. Scalero and N. Tepedelenlioglu, "A Fast Training Algorithm for Neural Networks," in *Proceedings of International Joint Conference on Neural Networks*, Washington, 1990.
42. A. Rezgui, "The Effect of an Adaptive Slope of the Activation Function on the Performance of the Backpropagation Algorithm," Ph.D. dissertation, Florida Institute of Technology, Melbourne, August 1990.
43. A. Rezgui and N. Tepedelenlioglu, "The Effect of the Slope of the Activation Function on the Performance of the Backpropagation Algorithm," in *Proceedings of International Joint Conference on Neural Networks*, vol. 1, Washington, January 1990, pp. 707–710.
44. R. Hecht-Nielsen, "Counterpropagation Networks," in *Proceedings of International Conference on Neural Network*, vol. 2, New York: IEEE, 1987, pp. 19–32.
45. R. Hecht-Nielsen, "Counterpropagation Networks," *Applied Optics*, vol. 26, 1987, pp. 4979–84.
46. R. Hecht-Nielsen, *Neurocomputing*, Reading, MA: Addison-Wesley, 1990.
47. S. Chen, S. A. Billings, C. F. N. Cowan, and P. M. Grant, "Practical Identification of NARMAX Models Using Radial Basis Functions," *International Journal of Control*, vol. 52, 1990, pp. 1327–50.
48. D. S. Broomhead and D. Lowe, "Multivariable Functional Interpolation and Adaptive Networks," *Complex Systems*, vol. 2, 1988, pp. 269–303.
49. E. S. Chen, S. Chen, and B. Mulgrew, "Efficient Computational Schemes for the Orthogonal Least Squares Algorithm," *IEEE Transactions on Signal Processing*, vol. 43, 1995, pp. 373–6.
50. S. Chen, C. F. Cowan, and P. M. Grant, "Orthogonal Least Squares Algorithm for Radial Basis Function Networks," *IEEE Transactions on Neural Networks*, vol. 2, pp. 302–9, 1991.
51. S. Chen, B. Mulgrew, and P. M. Grant, "A Clustering Technique for Digital Communications Channel Equalization Using Radial Basis Function Neural Networks," *IEEE Transactions on Neural Networks*, vol. 4, July 1993, pp. 570–9.
52. S. Chen, S. A. Billings, C. F. Cowan, and P. M. Grant, "Non-linear Systems Identification Using Radial Basis Functions," *International Journal of Control*, vol. 21, 1990, pp. 2513–39.
53. F. M. Ham, L. V. Fausett, and Byoungho Cho, "Classification of Maximal Length Pseudo-random Sequences Using ART and SOM," in *Proceedings of Artificial Neural Networks in Engineering 1993*, November 14–17, 1993, pp. 851–6.

163

164

# 第4章 自组织网络

## 4.1 概述

虽然有许多不同类型的自组织神经网络，但它们具有共同特征。这就是具有如下三种能力：评估提交给网络的输入模式，基于输入聚合集之间网络自身的相似性组织网络自身去学习，以及把输入数据分类（或聚类）成相似的模式组。因此，这类神经网络的学习无“教师”，即无监督学习。在第9章讨论神经网络的主成分分析（PCA）神经网络时将回顾这类自组织的思想。将无教师学习的相同思想应用于（自适应地）从输入数据抽取主成分信息。

通常，自组织（或无监督）学习涉及响应一系列输入模式而频繁修改网络突触权值。权值修改依照一系列的学习规则。这些模式重复应用于网络之后，出现一些有意义的构型(configuration)。基本上，为了响应输入模式，从网络内许多原始随机的局部相互作用中涌现全局顺序。该全局顺序最终导致某种形式的一致行为。然而，对于自组织学习而言，为了执行有意义的信息处理功能，提交给网络的输入模式必须存在冗余。从输入模式的这种冗余中呈现出顺序和结构，以及因此而来的使神经网络能吸收作为知识的信息。例如，在PCA情形
165 下，从大量提交给网络的随机输入模式中，神经元权值收敛于将输入映射到输入数据的主成分的映射算子。换言之，从大量（冗余）输入数据中，网络学习输入模式固有的潜在特征，映射（矩阵），即网络的突触权值包含这个信息。

特殊的自组织神经网络基于竞争学习。在竞争学习网络中输出神经元之间竞争，决出获胜者。这章将学习这类网络。给出三类基本的神经网络：Kohonen自组织映射（SOM）、学习向量量化（LVQ）和自适应共振理论（ART）网络。虽然LVQ学习不是非监督的，恰好不属于这类网络。然而，从某种意义上来说，由于LVQ同Kohonen SOM有密切的联系而确实属于这类神经网络。

## 4.2 Kohonen自组织映射

由Kohonen提出的自组织映射是非监督的、竞争学习的聚类网络，在其中一次仅有一个神经元（或一组中仅有一个神经元）“激活”[1]。自组织映射（SOM）是模仿在大脑中发生的某些映射的人工系统。例如，在视觉系统中，存在几类视觉空间到可视皮层的表面的拓扑映射。这类自组织神经网络的基本思想是（从原始事件空间）输入被自适应单元的简单网络接受。信号表示以一种使得响应保持原始事件相同的拓扑排序的方式（自动）映射为一系列输出。因此，网络能获得可观察事件属性的正确拓扑映射的自动形成。换言之，SOM以拓扑有序的方式将（任意维）输入模式变换为一维或二维的特征映射。图4-1表示通常的特征映射结构（从输入向量集到二维映射）。图4-2表示一个生物学激发的映射网络，它模仿从视网膜到皮层的映射（这类结构的研究通常少于图4-1表示的常用结构）。

我们将研究用于图4-1表示结构的Kohonen训练算法。即使输出神经元之间没有侧向连接，但是与输入最佳匹配的神经元邻域内的神经元（即获胜神经元）被修改，使与以前相比它们更像获胜单元那样响应。

神经元不是以相互独立的方式而是以拓扑相关的方式学习，对于形成有序映射来说是至

关重要的。在生理学激发的神经网络模型中，空间邻近的神经元使用侧向反馈连接和其他侧向交互作用获得相关学习。使用侧向连接的早期特征映射结构由Willshaw and von der Malsburg[3]开发视网膜映射问题而提出。他们运用图4-2所示的输出神经元之间具有（墨西哥帽形式）侧向连接的结构。 166

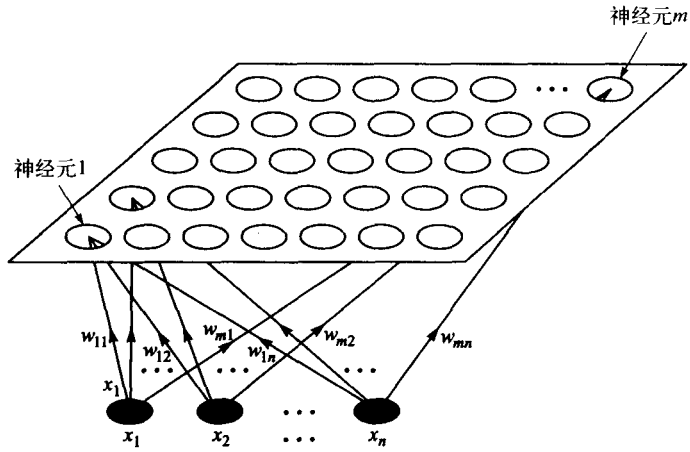



图4-1 自组织映射—常用的特征映射结构。输入与输出神经元全连接，但仅显示了少数连接

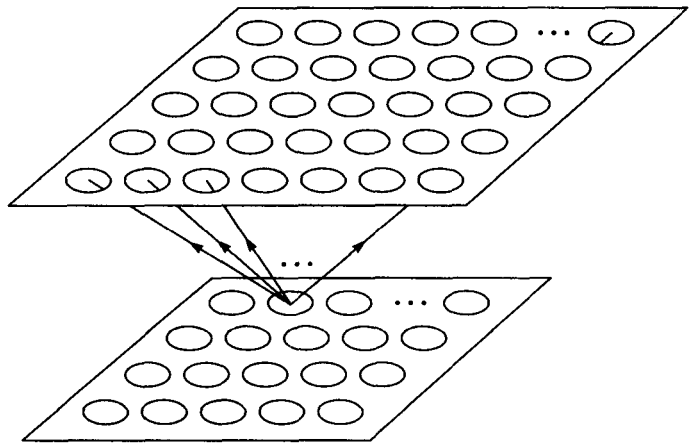


图4-2 自组织映射—生物学激发的映射（能模仿从视网膜到皮层的映射）。每一输入与所有输出神经元全连接；但图中仅给出了少数连接

图4-1所示的网络输入可写成向量形式：

$$\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T \tag{4-1}$$

167

二维（2-D）阵列中神经元$i$的突触权值向量由下式给出：

$$\boldsymbol{w}_i = [w_{i1}, w_{i2}, \cdots, w_{in}]^T \quad i = 1,2,\cdots,m \tag{4-2}$$

其中$m$是2-D阵列中的输出神经元总数。输入向量$\boldsymbol{x}$与突触权值向量$\boldsymbol{w}_i$的最佳匹配由下式确定：

$$q(\boldsymbol{x}) = \min_{\forall i} \| \boldsymbol{x} - \boldsymbol{w}_i \|_2 \quad i = 1,2,\cdots,m \tag{4-3}$$

其中$q(\boldsymbol{x})$表示输出神经元阵列的索引，特别指定为*获胜神经元*，$\| \cdot \|_2$是$L_2$或欧几里得范数。因此，通过运用式（4-3），连续输入空间映射到神经元的离散阵列。网络的响应也可以是最邻

近输入的输出神经元突触权值向量，而不是它在输出神经元阵列中的索引位置。Kohonen算法的下一步是更新与获胜神经元相联系的突触权值向量和获胜神经元的确定邻域内的神经元的突触权值向量。学习规则可表示为

$$\boldsymbol{w}_i(k+1)=\boldsymbol{w}_i(k)+\eta_{qi}(k)[\boldsymbol{x}(k)-\boldsymbol{w}_i(k)] \tag{4-4}$$

其中

$$\eta_{qi}(k)=\begin{cases}\mu(k) & \text{在}N_q\text{（获胜神经元}q\text{的邻域集合）内，其中}0<\mu(k)<1\text{，学习率参数应随时间下降}\\ 0 & \text{在}N_q\text{外}\end{cases} \tag{4-5}$$

是标量核函数（或邻域函数）。更具体地，由式（4-4）和式（4-5），学习规则可写为

$$\boldsymbol{w}_i(k+1)=\begin{cases}\boldsymbol{w}_i(k)+\mu(k)[\boldsymbol{x}(k)-\boldsymbol{w}_i(k)] & \text{如果}\quad i\in N_q(k)\\ \boldsymbol{w}_i(k) & \text{如果}\quad i\notin N_q(k)\end{cases} \tag{4-6}$$

其中$0<\mu(k)<1$（学习率参数）。注意式（4-6）中的$N_q$通常看成是离散时间指标$k$的函数，即$N_q(k)$。在训练开始时，令邻域集$N_q(k)$相对大一些，然后随着时间单调地缩小[4]，已经证明是有利的（如图4-3所示）。

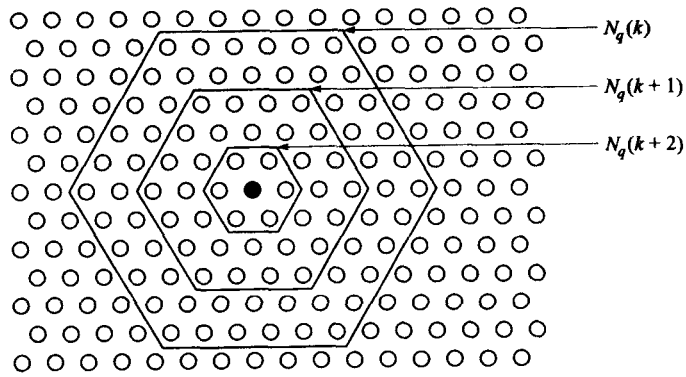



图4-3 拓扑邻域的例子，表示邻域的单调缩小

从生物侧向相互作用是钟形曲线[4]的角度来说，邻域函数$\eta_{qi}$的定义可以更一般化。为了将其与核函数合并，分别用向量$\boldsymbol{r}_q$和$\boldsymbol{r}_i$表示神经元$q$和$i$的坐标。因此，式（4-4）中$\eta_{qi}$的一种典型选择为

$$\eta_{qi}=\eta_0\exp(-\|\boldsymbol{r}_i-\boldsymbol{r}_q\|^2/\sigma^2) \tag{4-7}$$

其中的$\eta_0=\eta_0(k)$和$\sigma=\sigma(k)$选择为适合的时间下降函数。根据Hertz et al.[6]，式（4-4）中的
168 学习规则把与获胜单元相关的权值向量$\boldsymbol{w}_q$拖向$\boldsymbol{x}$，同时也把那些最接近单元的$\boldsymbol{w}_i$随$\boldsymbol{w}_q$一起拖动。在输入空间可以将弹性网络想象成想尽可能接近网络输入。弹性网络具有输出阵列的拓扑，网络的点可看成具有权值作为坐标。

在学习过程期间，有两个分离但关联的阶段，即排序阶段和收敛阶段。在初始学习过程，即排序阶段，学习率参数应设置得接近单位值，然后渐渐地下降（但不允许小于0.1）。在学习过程的这段期间完成权值向量的拓扑排序。收敛阶段（学习过程的第二阶段）一般是网络学习最长的部分。在这阶段为了完成映射的精细调整，剩下的迭代是必需的。学习率参数应该长时间保持相对小的值。例如，学习率参数应该接近（或小于）0.01[4]。

SOM算法的小结

步骤1 初始化网络权值向量$w_i$[⊖]初始化学习率参数，定义拓扑邻域函数，初始化参数，设置$k=0$。

步骤2 检查停止条件，[⊖]如果失败，继续，如果成功，退出。

步骤3 对于每个训练向量$x$，执行步骤4—步骤7。

步骤4 计算和输入最好匹配权值向量

$$q(x)=\min_{i}\|x-w_i\|,$$

步骤5 对于给定邻域$i\in N_q(k)$的所有单元（$q$是获胜神经元），按如下公式更新权值向量

$$w_i(k+1)=\begin{cases}w_i(k)+\mu(k)[x(k)-w_i(k)] & \text{如果 } i\in N_q(k)\\ w_i(k) & \text{如果 } i\notin N_q(k)\end{cases}$$

其中$0<\mu(k)<1$（学习率参数）。

步骤6 调整学习率参数。

步骤7 适当缩减拓扑邻域$N_q(k)$。

步骤8 设置$k\leftarrow k+1$，然后转到步骤2。 □ 169

例4.1 在单位正方形内均匀产生1 000个二维向量，参看图4-4。这些向量映射到一个5×5平面阵列的神经元。使用MATLAB神经网络工具箱中的函数trainsm完成模拟。在训练期间，邻域和学习率参数均逐渐减小。图4-5显示随网络学习表示输入的分布时神经网络训练的结果。图4-5b中在800次迭代后映射开始“展开”。在图4-5c框架中，在1 600次迭代后，映射继续展开，最后在10 000次迭代之后。图4-5d框架显示出映射本质上展开，在图中神经元的统计分布逼近输入向量的分布，它是输入向量的均匀分布的结果。这将不是对其他输入分布的情形。图4-6显示一个可能偶尔发生的异常，也就是说，映射的扭曲。当映射的不同部分拟合输入空间的分离部分的拓扑时，可能出现这种情形。这个例子与前一个例子的唯一不同是一个不同的初始化权值矩阵。在4 000次迭代后，映射仍然没有展开，并且它将不大可能展开。所以，最好是简单地重启动一个具有不同初始权值集合的训练过程。

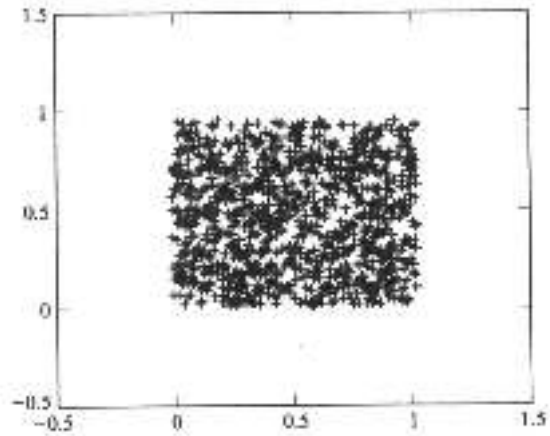



图4-4 例4.1中使用的1 000个二维随机向量的分布。在两个维上向量是从[0，1]区间内的均匀分布抽取的

例4.2 Kohonen SOM的一个非常重要的特征称为拓扑排序性质，这个性质使SOM能够从高维的输入空间中形成一个抽象的二维表示。这里说明这个属性。产生两个分别具有1 000个三维高斯向量集。两个数据集的方差$\sigma^2=0.1$，而第一个数据集合的中心在（0，0，0）（类别1），第二个数据集合的中心在（5，5，5）（类别2）。图4-7显示两个高斯云。我们训练一个正方形内有25个神经元的SOM映射。这两个三维向量集合提交给网络5 000次，也就是5 000个训练回合。 170

⊖ 实现网络权值的初始化既可以随机初始化，也可以选择反映输入数据的某些先验知识的一组权值。也就是，输出聚类的可能分布的信息。

⊖ 例如，停止条件可以基于指定的迭代总次数或者监视权值的变化。

如前例，使用MATLAB神经网络工具箱中的函数trainsm完成模拟，在训练期间邻域和学习率参数（初始设为1）均是逐渐减少的。在5 000个训练回合之后的二维映射显示在图4-8a中（这是训练时MATLAB产生的拓扑网格）。在图4-8b中画出三维的权值向量，并且两个类显示的轮廓通过计算每个权值向量均值来决定。从图4-8a及图4-8b中，我们发现SOM能适当地分类二维高斯向量。

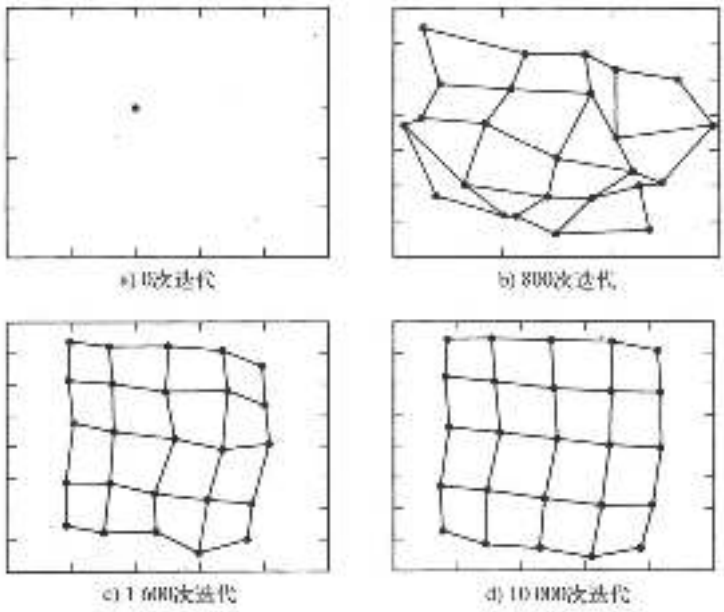



图4-5　从一个单位正方形到一个5×5神经元阵列的均匀随机向量的映射。初始学习率参数设置为$\alpha(0)=1$

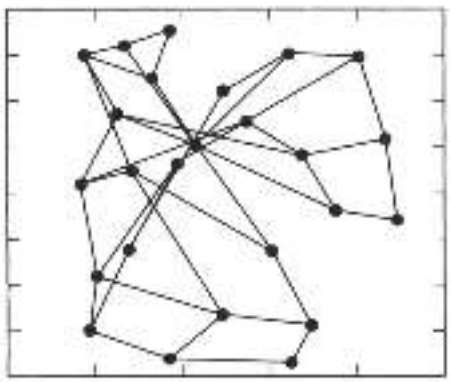


图4-6　经过4 000次迭代后扭曲的特征映射

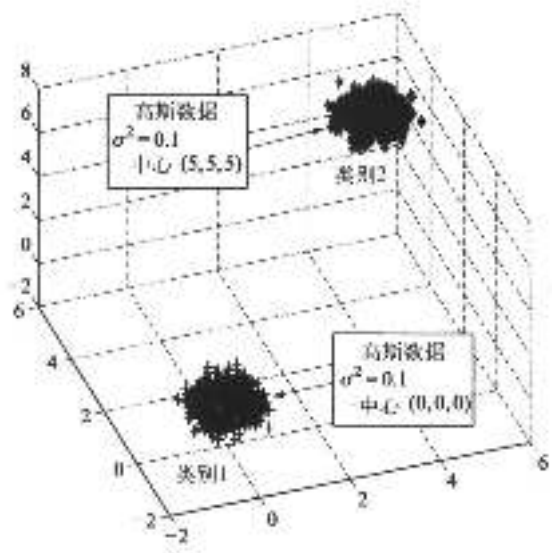
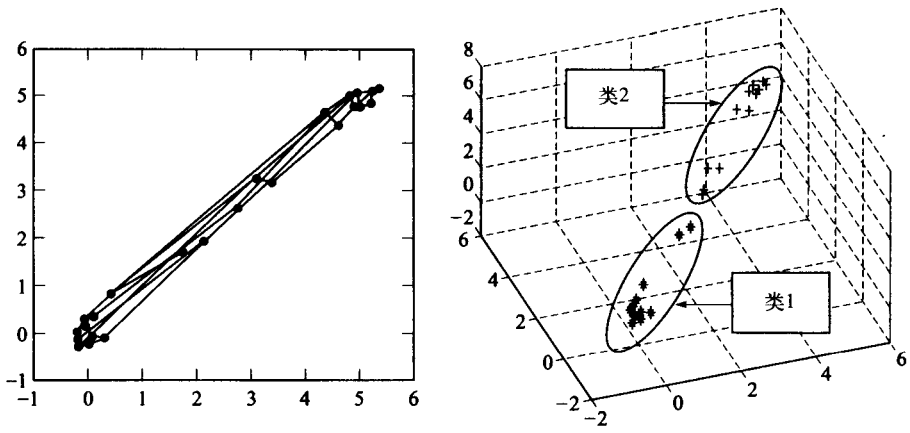



图4-7　三维高斯向量的两个集合，二者方差均为$\sigma^2=0.1$和如图所示的中心




a) b)

图4-8 a)两个三维高斯向量集合到一个5×5神经元阵列的Kohonen SOM网络映射；b) SOM的三维权值。通过计算每个权值向量的均值来决定这两个类

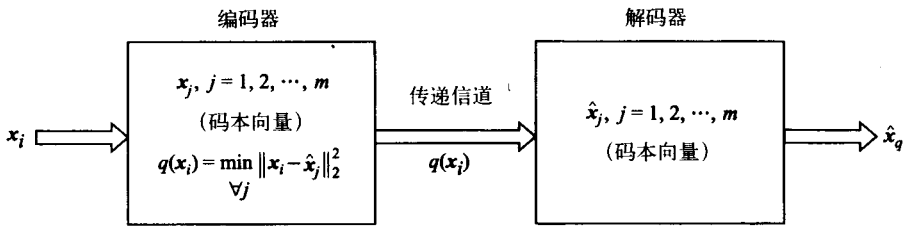
171
~
172

## 4.3 学习向量的量化

向量的量化[7-10]是用于语音和图像数据压缩的技术。基本思想是用较小的原型集合表示输入向量，提供给输入空间$\mathscr{X}$一个好的近似，其中向量$\boldsymbol{x}_i(i = 1, 2, \cdots, N)$构成输入空间。无需给出输入空间概率模型的先验知识，假定可以提供数据的一个很长的训练序列（或一个相对大的输入训练向量集合，即$N \gg 1$）。目标是开发量化向量的“码本”，然后，使用这些向量编码任何输入向量。为了开发出一个可靠的码本，使用大集合的训练向量根据预先确定的聚类数目形成组，并且聚类$j$用它的特定中心$\hat{\boldsymbol{x}}_j$（即再造或重构向量）来表示。中心聚类是基于给定的失真度量。虽然可运用几个不同的失真度量，然而，失真度量应该是易处理的、利用样本数据易计算和主观上是有意义的[11]。这类失真度量之一是基于向量的$L_2$（欧几里得）范数的，即：

$$d(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \| \boldsymbol{x} - \hat{\boldsymbol{x}}^2 \|_2^2 = (\boldsymbol{x} - \hat{\boldsymbol{x}})^T (\boldsymbol{x} - \hat{\boldsymbol{x}}) \tag{4-8}$$

称为平方误差失真。一旦产生码本，将储存在“传送者”和“接受者”两处。输入向量$\boldsymbol{x}_i$的量化按以下步骤进行：(1) 传送输入给向量量化器，并且与码本向量$\hat{\boldsymbol{x}}_j$ $(j = 1, 2, \cdots, m)$比较，最后选择最小失真的码本向量（即根据式 (4-8) 得到最小距离）。(2) 选定的向量$\hat{\boldsymbol{x}}_q$代表$\boldsymbol{x}_i$，索引$q$（与输入向量所属的恰当类型相关）传递给接受者，选择与输入向量$\boldsymbol{x}_i$的描述一样的合适的重构向量$\hat{\boldsymbol{x}}_q$。图4-9解释了向量量化过程。




$\boldsymbol{x}_i \in \mathscr{X}$（输入空间）

$q(\boldsymbol{x}_i) = q$（与“最靠近”输入$\boldsymbol{x}_i$的重生向量$\hat{\boldsymbol{x}}_q$相关的索引）

图4-9 向量量化器

另外一个失真度量是马哈拉诺比斯（Mahalanobis）失真[12]，定义为：

$$d_M(\boldsymbol{x},\hat{\boldsymbol{x}}) = (\boldsymbol{x}-\hat{\boldsymbol{x}})^T \boldsymbol{C}_x^{-1}(\boldsymbol{x}-\hat{\boldsymbol{x}}) \tag{4-9}$$

其中权值矩阵$\boldsymbol{C}_x$输入的协方差矩阵，即 $\boldsymbol{C}_x = \mathrm{E}\{(\boldsymbol{x}-\bar{\boldsymbol{x}})(\boldsymbol{x}-\bar{\boldsymbol{x}})^T\}$，其中 $\bar{\boldsymbol{x}} = \mathrm{E}\{\boldsymbol{x}\}$（参见A.7.6节）。另一类更复杂的Itakura and Saito失真度量[13, 14]定义如下：

$$d_{IS}(\boldsymbol{x},\hat{\boldsymbol{x}}) = (\boldsymbol{x}-\hat{\boldsymbol{x}})^T \boldsymbol{R}(\boldsymbol{x})(\boldsymbol{x}-\hat{\boldsymbol{x}}) \tag{4-10}$$

其中对于每个$\boldsymbol{x}$，$\boldsymbol{R}(\boldsymbol{x})$是正定对称矩阵。

当用式（4-8）定义的欧几里得距离（失真）度量来确定输入向量$\boldsymbol{x}_i$属于的区域（类、组、分类或聚类）时，量化器称为*Voronoi*量化器[15]。Voronoi量化器将输入空间划分
173 成各种各样的Voronoi单元[8]，每一单元由重生向量 $\hat{\boldsymbol{x}}_j$ 表示。第$q$个Voronoi单元包括输入空间$\mathscr{X}$中在欧几里得意义下与重生向量 $\hat{\boldsymbol{x}}_q$ 靠近而远离其他重生向量 $\hat{\boldsymbol{x}}_j (j \neq q)$ 的那些点。图4-10显示将输入空间$\mathscr{X}$划分成与四个重生向量相联系的四个单元的例子。

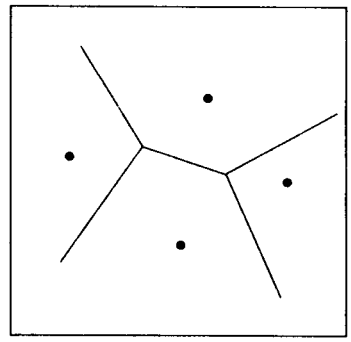


图4-10　具有四个重生向量表示四个单元的Voronoi量化器所实现的输入空间$\mathscr{X}$的划分

Kohonen开发了学习向量量化[16]，并且在[17]中总结了这个算法的三种版本。学习向量量化是在向量量化基础上能将输入向量分类的监督学习技术。这里的LVQ形式是LVQ1，是Kohonen的学习向量量化的第一种版本[16]。LVQ1训练过程开始于随机地自“标定”训练集合选择一个输入向量（以及该向量的正确类别，因此为监督学习）。每类或分类中实际存在几个重生（原型）向量。LVQ1与Kohonen SOM相似，即使LVQ1是一个监督网络，而Kohonen SOM是非监督的。

给定一个输入向量$\boldsymbol{x}_i$到网络，LVQ1的“输出神经元”（即，类或分类）认为是根据下式的一个“获胜者”

174
$$\min_{\forall j} d(\boldsymbol{x}_i,\boldsymbol{w}_j) = \min_{\forall j} \| \boldsymbol{x}_i - \boldsymbol{w}_j \|_2^2 \tag{4-11}$$

它与我们建立的Kohonen SOM“获胜规则”本质上是一样的。式（4-11）中唯一不同在于采用了欧几里得范数的平方。在式（4-11）中突触权值向量$\boldsymbol{w}_j$代替了图4-8和图4-9中表示的重生向量 $\hat{\boldsymbol{x}}_j$。LVQ1与Kohonen SOM的主要不同在于如何更新权值向量。我们用$\{\boldsymbol{x}_i\}(i = 1, 2, \cdots, N)$表示输入向量集，$\{\boldsymbol{w}_j\}(j = 1, 2, \cdots, m)$表示网络突触权值向量（Voronoi向量）。我们用 $\boldsymbol{C}_{w_j}$ 表示与（权值）Voronoi向量$\boldsymbol{w}_j$相关的分类，$\boldsymbol{C}_{x_i}$ 是网络输入向量$\boldsymbol{x}_i$的类标签，权值向量$\boldsymbol{w}_j$以下面方式调整：

1. 如果与权值向量相关的类与输入向量的类标签相同，即 $\boldsymbol{C}_{w_j} = \boldsymbol{C}_{x_i}$，则

$$\boldsymbol{w}_j(k+1) = \boldsymbol{w}_j(k) + \mu(k)[\boldsymbol{x}_i - \boldsymbol{w}_j(k)] \tag{4-12}$$

其中$0 < \mu(k) < 1$（学习率参数）。

2. 但，如果 $\boldsymbol{C}_{w_j} \neq \boldsymbol{C}_{x_j}$，则

$$\boldsymbol{w}_j(k+1) = \boldsymbol{w}_j(k) - \mu(k)[\boldsymbol{x}_i - \boldsymbol{w}_j(k)] \tag{4-13}$$

并且其他权值向量不调整。

所以，如果类是正确的，则式（4-12）中关于修改权值向量的更新规则是标准的。换句话说，根据式（4-12），如果输入向量与权值向量的类标签一致，则权值向量$\boldsymbol{w}_j$沿着输入向量$\boldsymbol{x}_i$方向移动。然而，如果类不正确，根据式（4-13），则权值向量$\boldsymbol{w}_j$沿着远离$\boldsymbol{x}_i$的方向移动。学习率

参数$\mu(k)$是关于离散时间指标$k$单调下降的（例如，随时间线性下降，起始点为0.01或0.02[17]。很多时候使用0.1为初值）。LVQ的收敛性已在Baras and LaVigna[18]中研究，他们的方法以随机逼近理论为基础。有几种方法可初始化权值，如，训练向量集合中的前$m$（总的类别数）个向量可用来初始化$m$个权值向量，即$\boldsymbol{w}_j(0)(j = 1, 2, \cdots, m)$。另一种方法是随机初始化权值向量（在输入向量的动态范围内）。终止条件可以基于期望的训练回合的总量，或者基于监控权值向量的收敛。另一类终止条件可以基于直接监控学习率参数，当学习率参数充分小时，终止训练。下面给出的例子将以迭代总次数（预定义）为终止条件。基本的LVQ1算法小结如下： 175

**LVQ1算法**

**步骤1** 初始化所有权值向量$\boldsymbol{w}_j(0)$，初始化学习率参数$\mu(0)$，并且设置$k = 0$。

**步骤2** 检查终止条件。如果失败，继续；如正确，退出。

**步骤3** 对每个训练向量$\boldsymbol{x}_i$执行步骤4与步骤5：

**步骤4** 决定权值向量指标$(j = q)$，以便$\min_{\forall j} \|\boldsymbol{x}_i - \boldsymbol{w}_j(k)\|_2^2$[使用在式（4-11）中给出欧几里得距离的平方]。[注意：$\boldsymbol{w}_q(k)$将为最小化范数平方的权值向量。]

**步骤5** 恰当更新权值向量$\boldsymbol{w}_q(k)$如下：

$$\text{如果 } \boldsymbol{C}_{w_q} = \boldsymbol{C}_{x_i} \quad \text{则 } \boldsymbol{w}_q(k+1) = \boldsymbol{w}_q(k) + \mu(k)[\boldsymbol{x}_i - \boldsymbol{w}_q(k)]$$

$$\text{如果 } \boldsymbol{C}_{w_q} \neq \boldsymbol{C}_{x_i} \quad \text{则 } \boldsymbol{w}_q(k+1) = \boldsymbol{w}_q(k) - \mu(k)[\boldsymbol{x}_i - \boldsymbol{w}_q(k)]$$

**步骤6** 用$k + 1$代替$k$，降低学习率参数，[⊖] 再回到步骤2。 □

LVQ1的神经结构如图4-11所示。实际上该结构除了没有拓扑结构外基本上与Kohonen SOM映射的结构一样。

Kohonen改进了LVQ1并且称为新版本LVQ2[17]。LVQ2算法基于光滑的移动决策边界逼近贝叶斯（Bayes）极限。LVQ2版本接着修改，导致LVQ2.1[17]，最终发展为LVQ3[4]。这些后来的LVQ版本共同具有获胜神经元的权值向量和“次获胜（runner up）”神经元的权值向量都被更新。

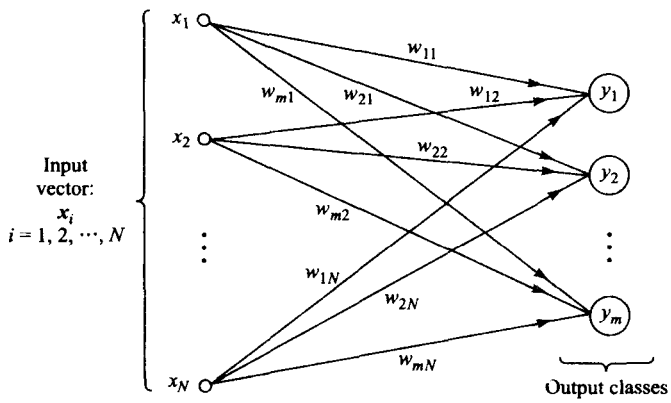



图4-11 学习向量量化的神经结构。输出类$y_j(j = 1, 2, \cdots, m)$有与输出类的突触权值向量相关的类标签，即$\boldsymbol{C}_{w_j}(j = 1,2,\cdots,m)$[⊜] 176

**例4.3** 我们给出了一个非常简单的、7个四维向量分配到两个类型的例子。7个向量和相

⊖ 学习率参数$\mu$可以随$k$（离散时间指标）缩减如下：对于$k > 0$，$\mu(k) = \mu(k-1)/(k+1)$。

⊜ 注意遵循第2章（参考2.2节）建立的网络权值的下标的习惯。但是，在上面给出的LVQ1算法中，考虑了权值向量的转置。

关的类型如下所示：

| 向量 | 类型 ($C_{x_i}$) |
|---|---|
| $x_1 = [1, 0, 0, 1]^T$ | → 1 |
| $x_2 = [0, 1, 1, 0]^T$ | → 2 |
| $x_3 = [0, 0, 0, 1]^T$ | → 2 |
| $x_4 = [1, 0, 0, 0]^T$ | → 1 |
| $x_5 = [1, 1, 1, 0]^T$ | → 1 |
| $x_6 = [0, 1, 1, 1]^T$ | → 2 |
| $x_7 = [1, 1, 1, 1]^T$ | → 1 |

如上所述，网络初始化的一个方法是使用来自上面集合的前两个向量$x_1$和$x_2$分别初始化$w_1$和$w_2$。对于$w_1$和$w_2$的相关类型，有 $C_{w_1} = 1$和 $C_{w_2} = 2$。使用剩余5个向量进行训练（第一个训练回合）。经历一个训练回合的细节。初始化学习率参数到$\mu(k = 1) = \mu(1) = 0.1$，随每个训练回合$k$来减少它，例如，$\mu(2) = \mu(1) / 2$，$\mu(3) = \mu(2) / 3$，等等。⊖

训练回合$1(k = 1)$：

1. 初始化权值：

$$w_1 = [1, 0, 0, 1]^T \text{（类别 } C_{w_1} = 1\text{）并且}$$
$$w_2 = [0, 1, 1, 0]^T \text{（类别 } C_{w_2} = 2\text{）}$$

2. 对于类别 $C_{x_3} = 2$ 的输入向量$x_3 = [0, 0, 0, 1]^T$，检查：

$$\| x_3 - w_2 \|_2^2 = 3$$
$$\| x_3 - w_1 \|_2^2 = 1 \Rightarrow \text{最小值} \Rightarrow q = 1$$

由于 $C_{x_3} \neq C_{w_1}$，移动权值向量$w_1$远离$x_3$，也就是使用式（4-13）：

$$w_1 = [1, 0, 0, 1]^T - 0.1([0, 0, 0, 1]^T - [1, 0, 0, 1]^T)$$
$$= [1.1, 0, 0, 1]^T$$

177

3. 对于类型 $C_{x_4} = 1$的输入向量$x_4 = [1, 0, 0, 0]^T$，检查：

$$\| x_4 - w_2 \|_2^2 = 3$$
$$\| x_4 - w_1 \|_2^2 = 1.01 \Rightarrow \text{最小值} \Rightarrow q = 1$$

由于 $C_{x_4} = C_{w_1}$，沿$x_4$方向移动权值向量$w_1$，即使用式（4-12）：

$$w_1 = [1.1, 0, 0, 1]^T + 0.1([1, 0, 0, 0]^T - [1.1, 0, 0, 1]^T) = [1.09, 0, 0, 0.9]^T$$

4. 对于类型 $C_{x_5} = 1$的输入向量$x_5 = [1, 1, 1, 0]^T$，检查：

$$\| x_5 - w_1 \|_2^2 = 2.8181$$
$$\| x_5 - w_2 \|_2^2 = 1 \Rightarrow \text{最小值} \Rightarrow q = 2$$

由于 $C_{x_5} \neq C_{w_2}$，移动权值向量$w_2$远离$x_5$，也就是使用式（4-13）：

$$w_2 = [0, 1, 1, 0]^T - 0.1([1, 1, 1, 0]^T - [0, 1, 1, 0]^T)$$
$$= [-0.1, 1, 1, 0]^T$$

5. 对于类型 $C_{x_6} = 2$的输入向量$x_6 = [0, 1, 1, 1]^T$，检查：

⊖ 注意这个例子中，我们采用初始离散时间指标为$k = 1$替代$k = 0$。因此，学习率参数现在根据$\mu(k) = \mu(k-1)/k$，$k > 1$调整。

$$\| x_6 - w_1 \|_2^2 = 3.1981$$

$$\| x_6 - w_2 \|_2^2 = 1.01 \Rightarrow \text{最小值} \Rightarrow q = 2$$

由于 $C_{x_6} = C_{w_2}$，沿$x_6$方向移动权值向量$w_2$，也就是使用式（4-12）：

$$w_2 = [-0.1, 1, 1, 0]^T + 0.1([0, 1, 1, 1]^T - [-0.1, 1, 1, 0]^T) = [-0.09, 1, 1, 0.1]^T$$

6. 对于类型 $C_{x_6} = 1$的输入向量$x_7 = [1, 1, 1, 1]^T$，检查：

$$\| x_7 - w_1 \|_2^2 = 2.0181 \quad \| x_7 - w_2 \|_2^2 = 1.9981 \Rightarrow \text{最小值} \Rightarrow q = 2$$

由于 $C_{x_7} \neq C_{w_2}$，移动权值向量$w_2$远离$x_7$，也就是说，使用式（4-13）：

$$w_2 = [-0.09, 1, 1, 0.1]^T - 0.1([1, 1, 1, 1]^T - [-0.09, 1, 1, 0.1]^T) = [-0.199, 1, 1, 0.01]^T$$

训练回合1的结束

表4-1 用于LVQ1的MATLAB函数

```
function W = lvq1 (X,CX,m,mu,maxiter)
%
% W =       LVQ1 (X,CX,mu,maxiter) computes the weight
%           matrix for learning vector quantization 1 (LVQ1)
%
% X:        is the matrix of inputs, i.e., each column
%           vector is an input
% CX:       is a row vector of scalar ''classes'' associated
%           with the column vectors of X
% m:        number of different classes
% mu:       initial learning rate parameter
% maxiter: maximum number of training epochs (iterations)
%
N=size (X,2);
% Initialize the weight vectors with the first nc vectors
% from the training set (Note: must have training vectors
% arranged so first nc vectors have the full set of classes
% to be represented) .
W=X(:,1:m) ;
CW=CX(1:m) : % classes for weight vectors
snorm=zeros (1,m);
niter=1;
while niter <= maxiter
   if niter == 1
      for i=m+1:N
         for j=1:m
            snorm (1,j)=norm(X(:,i)-W(:,j))^2;
         end
         [mind,index]=min(snorm) ;
         if CX(i)==CW(index)
           W(:,index)=W(:,index)+mu*(X(:,i)-W(:,index)) ;
         else
           W(:,index)=W(:,index)-mu*(X(:,i)-W(:,index)) ;
         end
      end
   else
      for i=1:N
         for j=1:m
            snorm(1,j)=norm(X(:,i)-W(:,j))^2;
         end
         [mind,index]=min(snorm) ;
         if CX(i)==CW(index)
            W(:,index)=W(:,index)+(mu/niter)*(X(:,i)-W(:,index));
         else
            W(:,index)=W(:,index)-(mu/niter)*(X(:,i)-W(:,index));
         end
      end
   end
   niter=niter+1;
end
```

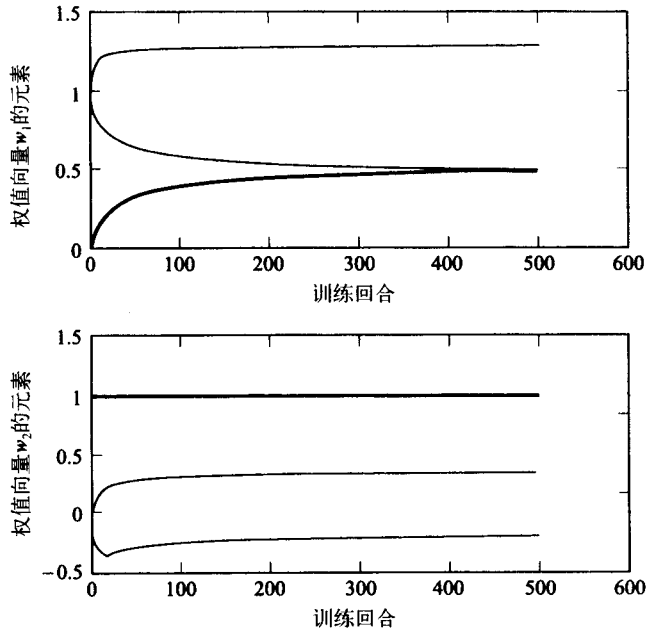



图4-12　两个权值向量的元素作为训练回合函数的收敛图

根据上面给出的算法，置$k = 2$，$\mu(2) = \mu(1) / 2 = 0.05$，转到步骤2，并且检查停止条件。在下一回合的训练中以第一次训练的向量$x_1$开始，继续下去。在这个例子中，使用表4-1的MATLAB函数，训练回合数置为500。

在500次训练回合之后最终的权值为

$$w_1 = \begin{bmatrix} 1.2996 \\ 0.4952 \\ 0.4952 \\ 0.4848 \end{bmatrix} \text{和} \; w_2 = \begin{bmatrix} -0.1881 \\ 1 \\ 1 \\ 0.3603 \end{bmatrix}$$

图4-12显示了两个权值向量的元素作为训练回合函数的收敛图。最后，若我们计算每个输入向量和上面显示的计算权值向量$w_1$和$w_2$的最小距离，这将指出每个输入向量$x_i$属于哪个类。结果如下：

178 ~ 179

$$\min\{\|x_1 - w_1\|_2^2, \|x_1 - w_2\|_2^2\} \Rightarrow x_1\text{最靠近}w_1 \Rightarrow \text{类1}$$
$$\min\{\|x_2 - w_1\|_2^2, \|x_2 - w_2\|_2^2\} \Rightarrow x_2\text{最靠近}w_2 \Rightarrow \text{类2}$$
$$\min\{\|x_3 - w_1\|_2^2, \|x_3 - w_2\|_2^2\} \Rightarrow x_3\text{最靠近}w_2 \Rightarrow \text{类2}$$
$$\min\{\|x_4 - w_1\|_2^2, \|x_4 - w_2\|_2^2\} \Rightarrow x_4\text{最靠近}w_1 \Rightarrow \text{类1}$$
$$\min\{\|x_5 - w_1\|_2^2, \|x_5 - w_2\|_2^2\} \Rightarrow x_5\text{最靠近}w_1 \Rightarrow \text{类1}$$
$$\min\{\|x_6 - w_1\|_2^2, \|x_6 - w_2\|_2^2\} \Rightarrow x_6\text{最靠近}w_2 \Rightarrow \text{类2}$$
$$\min\{\|x_7 - w_1\|_2^2, \|x_7 - w_2\|_2^2\} \Rightarrow x_7\text{最靠近}w_1 \Rightarrow \text{类1}$$

这些结果准确对应于每个输入向量确定的类。

自组织映射和LVQ

通过结合Kohonen SOM和LVQ，可发展一个自适应模式分类系统。模式识别中使用的$K$均值聚类[19]可以用来替代Kohonen SOM。我们仅考虑后者。处理中的第一步使用SOM选择

一个包含有关输入数据的相关信息的比较小的特征集合，输入数据可以在第二步由LVQ网络分类。所以，LVQ网络将起到一个实际分类器的作用，使用选择自输入数据的特征（由SOM产生），把它们指派给每个类或分类。这个混合的自适应模式分类系统组合了无监督网络（SOM）和监督网络（LVQ），如图4-13所示。 180

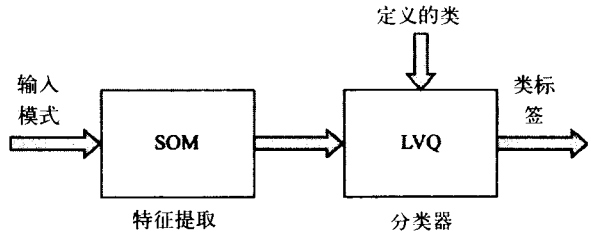



图4-13 使用Kohonen SOM和LVQ的自适应模式分类系统

**例4.4** 图4-13中的混合的自适应模式分类网络的用途。使用例4.2的结果，也就是使用Kohonen SOM产生的三维权值向量[显示在图4-8b，并且在图4-14a中重复]。输入25个SOM权值向量到有两个确定类（类1和类2，如图4-14a所示）的LVQ网络。修改表4-1所示的MATLAB LVQ1程序，随机地初始化两个三维权值。初始学习率参数设置为$\mu(1) = 0.1$，训练回合总数为1000。最后，两个三维权值向量显示在图4-14b中，两个权值向量表示为：

$$\boldsymbol{w}_1 = \begin{bmatrix} 0.3232 \\ 0.3770 \\ 0.3113 \end{bmatrix} \text{和} \ \boldsymbol{w}_2 = \begin{bmatrix} 4.2837 \\ 4.2825 \\ 4.2575 \end{bmatrix}$$

它们相当接近于图4-7中的高斯云的中心。将图4-14a中的SOM结果与图4-14b使用LVQ结果相比较，容易地发现通过使用LVQ分类模式和指派它们到两个不同的类所获得的提高。

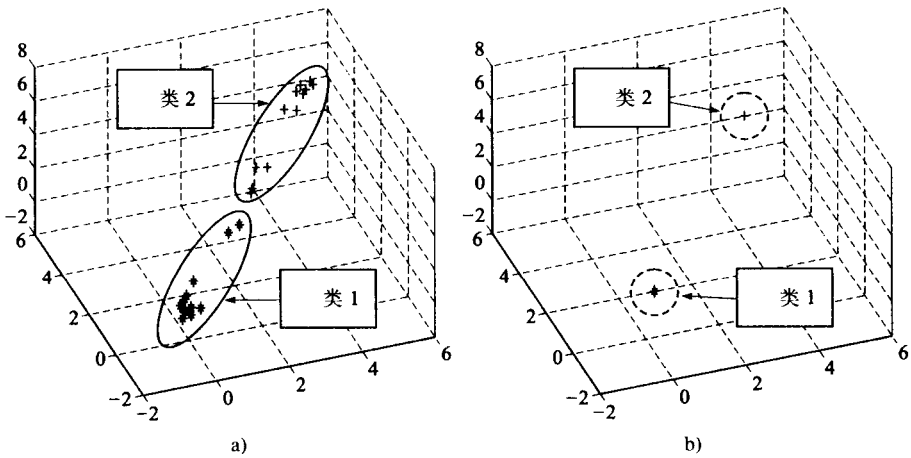



图4-14 a)例4.2带有两个定义类别的SOM三维权值向量；b)1000个训练回合后LVQ1网络的最终权值向量 181

## 4.4 自适应共振理论（ART）神经网络

由竞争学习网络形成的聚类（分类）并不能保证稳定。例如，即使连续提交相同的输入向量集合给网络，获胜单元也会继续变化。防止这种情况的一个途径是逐渐缩减学习率直至

零，然后冻结学习所得的分类。然而，这样做时，获得稳定性的代价是损失可塑性(plasticity)，或网络对新数据的反应能力（即，网络不能学习新类别）。这一问题通常称为Grossberg稳定性/可塑性困境（stability/plasticity dilemma）。由Carpenter和Grossberg发展的自适应共振理论（Adaptive Resonance theory, ART）[20]通过仅当输入与网络储存的类别原型足够相似时才接受和采用原型，以此来克服稳定性/可塑性困境。当输入模式和任何存在的类别原型都不够相似时，使用以前没有执行的输出（聚类）单元以输入模式作为原型形成一个新的类别。因此，这类网络可以本质上由自身产生新的聚类。如果没有剩余这样的未执行单元，那么一个新的输入不产生响应（即，一种外部拒绝的形式）。足够相似的意思依赖一个警戒参数（vigilance parameter）。如果警戒参数很大，相似条件变得非常严格，会形成许多良好划分的类别。另一方面，一个小的警戒参数给出一个粗的分类。在网络训练中，提交每个训练模式给网络几次。第一次提交一个模式给网络时，它可能放置于一个聚类单元；而当它在后面再提交时，它可能放置在不同的聚类单元。这归因于表示第一次聚类的网络权值的变化，假如它已经学习了其他的输入模式。一个稳定的网络将不返回模式到以前的聚类。

图4-15显示ART网络的一个基本特征。稳定性和可塑性能够使用增益控制单元$G_1$和$G_2$以及方向子系统M单元的警戒参数$\rho$来取得。在注意子系统内有一个称为特征表示域$F_1$的处理单元层，同时有一个称为类别表示域$F_2$的输出单元层。在这些域中有短期记忆（short-term memory, STM）的痕迹，因为它们仅存在于一个输入向量的一个相关简单应用中。在$F_1$与$F_2$之间自底向上和自顶向下的连接相关的权值称为长期记忆痕迹，因为它们编码那些在一个扩展期内保留为网络一部分的信息。兴奋的信号由+符号，而抑制信号由−符号来表示。在学习过程中，一旦一个聚类单元被选中，在一个扩展期内保持自底向上和自顶向下的信号。在此期间发生权值改变。这是
182 共振条件。ART网络的每个单元能够从三个源中接受信号。输入单元能够接受来自输入向量的信号，和来自$F_2$神经元或者增益控制单元$G_1$的自顶向下的信号。类似地，$F_2$单元能够接受来自$F_1$单元、M单元或增益控制单元$G_2$的信号。$F_1$或$F_2$单元必须接受两个兴奋的信号才能为“开(on)”。由于有三个可能的信号源，这称为2/3规则。辅助单元M（方向子系统）控制警戒匹配。

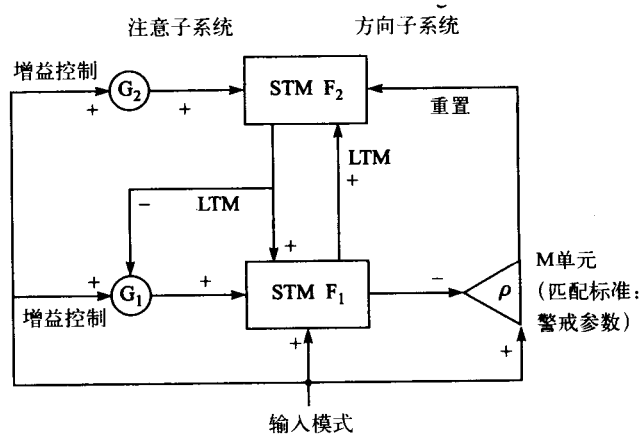



图4-15　ART体系结构的基本特征

网络神经元活跃水平的变化以及网络权值的变化实际上由耦合的微分方程决定。然而，在实践中可以假定神经元的活跃水平通常比权值改变得更快。所以，该过程可以看为简化方式。特别是它应用于训练ART网络的方式中。ART网络中有两类简化学习方法。它们区别在于所作的假设及性能特征。在快速学习情况下，假定在共振期间权值更新执行更快（相对于一个输入模式提交给网络的持续时间）。由此，网络权值在每次试验中达到平衡。相反地，慢

学习情况下，网络权值改变比在快学习方式更慢。因此，权值在任何特定试验中达不到平衡。在这两类基本学习模式之间存在折中。在慢学习情形下，达到平衡权值需要更多（与快学习比较）的训练回合。但是，在慢学习方式下所需计算更少。

无监督ART网络有三种基本类型：ART1（二值输入向量）、ART2（连续值输入向量）和模糊ART（二值和连续值的输入向量）。我们仅给出ART1网络的细节。 183

### 4.4.1 ART1

ART1设计为聚类二值向量[20]，图4-16显示了基本结构。这里给出的描述ART1网络运行的解释是基于结构单元的离散时间事件[21]，与微分方程相反[20]。网络的权值平衡点能容易地决定，而无需求助于微分方程的迭代解。$F_1$层（特征表示域）和$F_2$层（类别表示域）在两个方向上是全连接的，输出节点（神经元）之间也在两个方向上全连接，以便实现胜者全得子系统。然而，为了简单起见，图4-16中并没显示出这些连接。增益控制单元$G$为$F_1$层提供控制信号，有两个状态：如果工作的输出层没有单元是“开”的，则$G=1$；否则$G=0$。因此，增益$G$能够通过一个阈值逻辑单元（threshold logic unit, TLU）实现（参考2.2节），其中增益计算为：

$$G = T_{\text{binary}}\left(\sum_{i=1}^{n} x_i - n\sum_{j=1}^{m} z_j - 0.5\right) \tag{4-14}$$

其中

$$T_{\text{binary}}(\lambda) = \begin{cases} 1 & \text{如果 } \lambda > 0 \\ 0 & \text{如果 } \lambda \leqslant 0 \end{cases} \tag{4-15}$$

假定$\boldsymbol{x} \neq 0$，若输出单元是“开”状态，式（4-14）中括号内的量将是负的（$G=0$）；如果没有输出单元处于开状态，括弧内的值将是正的（$G=1$）。

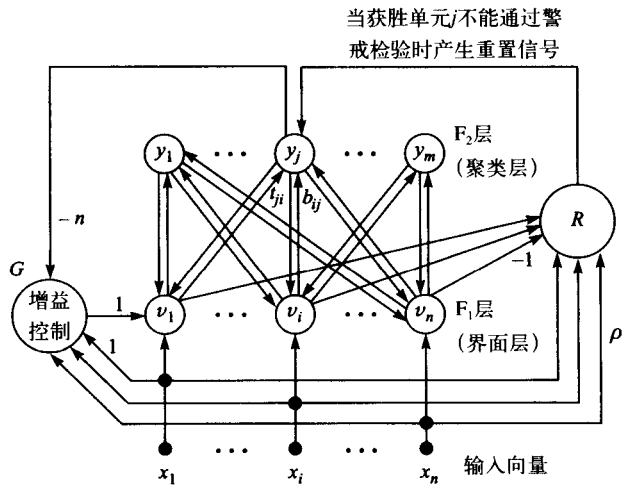



图4-16 ART1网络的结构，其中$t_{ji}$是自顶向下的权值（都初始化为1），并且$b_{ij}$是自底向上的权值 184

在$F_1$层中，单元的$v_i$状态定义如下：

$$v_i = T_{\text{binary}}\left(x_i + G + \sum_{j=1}^{m} t_{ji} z_j - 1.5\right) \tag{4-16}$$

如果增益$G=1$，意味着没有输出单元为开，并且式（4-16）的总和将是0。所以，如果输出向量第$i$个元素$x_i$是1，则$v_i=1$。如果$x_i=0$，则$v_i=0$。另一方面，当$G=0$时，由于在输出中仅一个输出单元能够是开的状态，$\sum_{j=1}^{m}t_{ji}z_j=t_{ji}$（即，单元$j$是获胜输出单元且$z_j=1$），对于$v_i=1$，$x_i$和$t_{ji}$必须是1。否则，若$x_i$或$t_{ji}$（或两者均）是0，则$v_i=0$。总之，从上面的讨论可知，为了$v_i=1$，式（4-16）括号内三项中至少两项必须呈现（即，为1）。这就是2/3规则。

在图4-15中的$F_2$层（输出层）是胜者全得的竞争层。在输出层$F_2$中，当输入向量第一次提交给网络时所有单元均断开，且$G=1$。因此，从前面的讨论知输入向量在$F_1$层复制，即，$v_i=x_i$（$i=1,2,\cdots,n$）。从$F_1$层到输出单元$j$的自底向上的权值可根据下式计算：

$$b_{ij}(k)=\frac{t_{ji}(k)}{\tau+\sum_{i=1}^{n}t_{ji}(k)} \tag{4-17}$$

其中$\tau=0.5$，$k$是离散时间指标，自顶向下的权值向量$\boldsymbol{t}_j$（原型向量）的第$i$元素$t_{ji}$是从$F_2$层第$j$单元到$F_1$层第$i$单元的连接权值。

接着建立决定在输出中“获胜”聚类的度量，从而确定输入$\boldsymbol{x}$与权值向量$\boldsymbol{t}_j$之间的匹配程度。这需要计算两个相似度量。两个度量是必需的，因为ART1结构中考虑的是二值而不是双极值向量[22]。第一个相似度量计算为

$$\sigma_1=\frac{\sum_{i=1}^{n}t_{ji}x_i}{\tau+\sum_{i=1}^{n}t_{ji}} \tag{4-18}$$

在可能的输出单元中，根据式（4-18）计算出的最大值能够发现最接近当前输入向量$\boldsymbol{x}$的原型向量$\boldsymbol{t}_j$。对于一个较小值$\tau$，式（4-18）中$\sigma_1$近似于输入向量$\boldsymbol{x}$中为1的分量与自顶向下权值向量$\boldsymbol{t}_j$中为1的分量的重叠个数和$\boldsymbol{t}_j$中为1的分量个数的比值。当网络输出的两个聚类具有与输入$\boldsymbol{x}$的
185 分量为1的重叠个数相同的原型向量，在原型向量中具有更少分量为1的那个聚类选择为获胜者。网络输入$\boldsymbol{x}$与所有的存在聚类相比较，以便减少相似性，除非所有存在的聚类不是足够相似，并不形成新的聚类。对于由式（4-18）的相似度量决定的网络输出的第$j$个获胜聚类，$G=0$。并且，第二个相似度量用于决定输入$\boldsymbol{x}$与$\boldsymbol{t}_j$之间的匹配是否足够好，它按如下计算

$$\sigma_2=\frac{\sum_{i=1}^{n}t_{ji}x_i}{\sum_{i=1}^{n}x_i} \tag{4-19}$$

然后比较这个值与阈值$\rho$（警戒参数）相比较，其中$0\leqslant\rho\leqslant1$。相似度量$\sigma_2$与极限参数$\rho$的比较在网络中通过图4-16中的重置单元按如下式子执行

$$R=T_{\text{binary}}\left(\rho\sum_{i=1}^{n}x_i-\sum_{i=1}^{n}v_i\right) \tag{4-20}$$

然而，$F_1$层的输出$v_i$的总和是$\sum_{i=1}^{n}v_i=\sum_{i=1}^{n}t_{ji}x_i$，式（4-20）能够改写成

$$R=T_{\text{binary}}\left[\left(\frac{\rho}{\sigma_2}-1\right)\sum_{i=1}^{n}v_i\right] \tag{4-21}$$

所以，从式（4-21）我们看到当$\alpha_2 \geqslant \rho$时，$R=0$；而当$\alpha_2 < \rho$时，$R=1$。当$R=0$时，网络中共振条件存在，自顶向下的权值$t_{ij}$按照如下式子更新

$$t_{ij}(k+1) = t_{ij}(k) + u(k)z_{ij}(k)\left[v_j(k) - t_{ij}(k)\right] \tag{4-22}$$

其中$0 \leqslant u(k) \leqslant 1$。当重置的值是$R=1$时，$F_2$层所有输出单元重置成断开状态，并且在当前输入向量进行处理时，当前竞争单元将丧失竞争能力。当获胜单元重置时，增益$G=1$，选择另外一个获胜单元，继续这个过程，如果在学习聚类中没有模式与输入足够相似，它们将逐个丧失竞争能力而选择的获胜单元将是一个没有使用的单元。

例4.5 介绍警戒参数如何影响ART1网络的聚类。图4-17显示的字母用作ART1网络的9个输入。每个字母转化成一个（列）向量表示，该向量是通过“堆栈”7×5阵列中的每个字母图像的连续的行来得到的。假定黑色像素为1，并且白色像素为0。[⊖]在类别表示域，即$F_2$（竞争）层节点的最大数量，假定为输入模式的数$m=9$。自底向上和自顶向下的权值分别如下初始化： 186

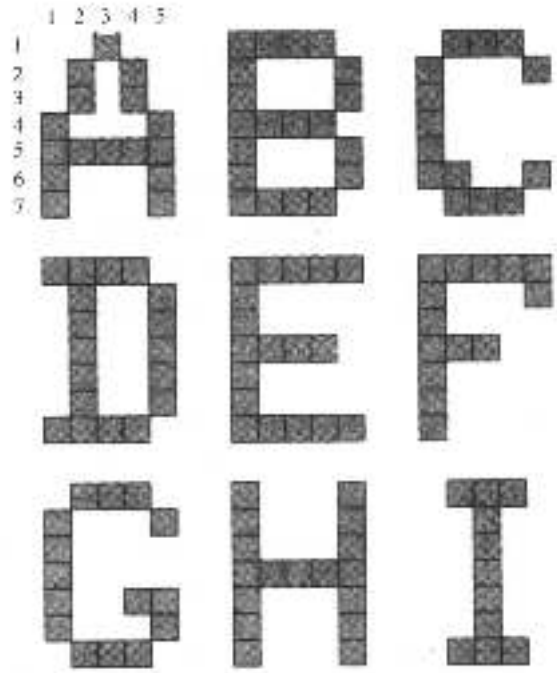



图4-17 用作输入例4.5中的ART1网络的字母

$$b_{ij} = \frac{L}{L-1+n} \quad （其中L>1，典型地，L=2）\quad t_{ij}=1$$

在$\rho=0.3$处设置警戒参数处生成四个类别或聚类。这些显示在表4-2中。如果在$\rho=0.7$处设置警戒参数，将形成两个多余的类。六个类显示在表4-3中。

正如表4-2及表4-3显示的结果，当警戒参数相当地小，和一个警戒参数更高设置时相比，聚类显得更粗糙，而后者使得聚类变得更精细。增加警戒参数到$\rho=0.95$，每个模式（字母）放置到自己的聚类中，允许模式回忆。

⊖ 为了在MATLAB中显示二值字母图像，首先使其“镜像(toggled)”，然后乘以63，使用colormap gray和image函数显示字母图像。在MATLAB:0=黑色，1=白色，乘以63来放大图像，产生6位的灰度范围。

表4-2　对于$\rho = 0.3$的ART1聚类结果

| 类　别 | 模式（字母） | 类　别 | 模式（字母） |
|---|---|---|---|
| 1 | A, B, C, G | 3 | H |
| 187 2 | D, E, F | 4 | I |

表4-3　对于$\rho = 0.7$的ART1聚类结果

| 类　别 | 模式（字母） | 类　别 | 模式（字母） |
|---|---|---|---|
| 1 | A | 4 | E,F |
| 2 | B, C, G | 5 | H |
| 3 | D | 6 | I |

### 4.4.2　模糊ART和模糊ARTMAP

ART1有几个与它相关的问题，例如，低效的存储能力（对于二进制输入向量可能的最大学习聚类数是$2^n$）和输入中的噪声灵敏性，以及其他一些例子。Carpenter和Grossberg[23]发展了用于模拟（或连续）输入的ART2以克服ART1的一些问题。模糊ART[24]有响应模拟或二值输入模式的任意序列的学习类别的能力。对于模拟和二值输入模式学习的泛化都是通过使用模糊集合论的MIN(^)算符（参看A.8节）替代ART1神经网络结构的相交（∩）算符来实现的。在二值情况下，MIN算符退化为相交算符，从而在响应二值输入模式时，模糊ART退化为ART1。在模糊ART中，输入向量按照一个导致对称理论的*补码编码*过程规范化，在对称理论中模糊集合论[25]的MIN(^)算符与MAX(v)算符执行互补作用[24]。补码编码在保持振幅信息的同时完成输入模式的规范化。

模糊ARTMAP[26]是ARTMAP（Adaptive Resonance Theory Mapping，自适应共振理论映射）[27]神经网络结构的推广。ARTMAP也称为预测ART网络，对于给定的输入模式（向量）流它能够快速且稳定地在线识别学习和假设检验。图4-18给出了模糊ARTMAP结构。从图中可看出，该结构由两个模糊ART模块构成。在网络的监督训练过程中，$ART_a$接收到输入模式$\{\boldsymbol{a}^{(p)}\}$流，$ART_b$接收输出模式$\{\boldsymbol{b}^{(p)}\}$（其中$\boldsymbol{b}^{(p)}$是给定$\boldsymbol{a}^{(p)}$的正确响应）。$ART_a$和$ART_b$模块通过联想学习网络和一个确保自动系统实时运行的内部控制器连接在一起。设计控制器使得它创建
188 满足精度标准所必需最小数量的$ART_a$识别分类（或“隐藏单元”）。这个过程是通过实现一个极小极大学习规则来完成的，极小极大学习规则能够使ARTMAP系统快速、有效且准确地学习，因为该系统结合最小化预测误差和最大化预测的泛化性能。这种过程在样例学习（trial-by-trial）基础上使用局部水平操作，将自动成功预测分类的大小。这个过程通过将$ART_a$的警戒参数$\rho_a$增加修正$ART_b$预测误差所必须的最少数量来执行。警戒参数$\rho_a$的较低值将导致形成较大分类数（较低$\rho_a$值将导致较宽的推广和更高的编码压缩）。当发生$ART_b$预测错误时，$ART_a$根据利用*匹配—跟踪*（match-tracking）机制[27]初始化假设检验所必须的最小数量来增加$\rho_a$。匹配跟踪给出了用来修正预测误差所需最小值的推广。可以通过运用输入数据集的不同序列的多次训练网络来获得预测性能的提高。这种“投票策略”是基于观察到ARTMAP快速学习通常对于确定训练集的不同顺序导致不同自适应权值和识别分类（甚至是对于不同刺激产生相似预测性能）。模糊ARTMAP的性能用来与嵌套广义范例（nested generalized exemplar, NGE）[28, 29]和模糊最小最大分类器（FMMC）[30]的性能相比较。

侧向启动自适应共振理论（laterally primed adaptive resonance theory, LAPART）[31]能完成响应输入向量为静态数据和顺序数据的识别分类和多维映射的增量式（incremental）监

督学习。在[31]中，Healy等人通过验证从以往经验推断出的模式对来强调LAPART在识别系列相似模式中的应用。LAPART结构由相互连接的ART网络构成，这种相互连接使LAPART能从一种模式类中推断出另一种模式类，以便形成一个预测序列。LAPART基于当前模式的识别预测下一种模式类，然后当新数据可以利用时检验该预测。模糊LAPART[32，33]是广义的LAPART结构，网络可以有模拟的和二值的输入。为了噪声输入集的有效编码，在模糊LAPART中融合具有快速执行和慢速重编码选项的慢速学习。Carpenter[34]已经发展了一类用于学习、识别和预测具有任意分布式编码表示的ART模型。这些分布式网络综合了胜者全得的ART网络的快速稳定学习的能力和多层感知器的抗噪声和压缩编码的能力。

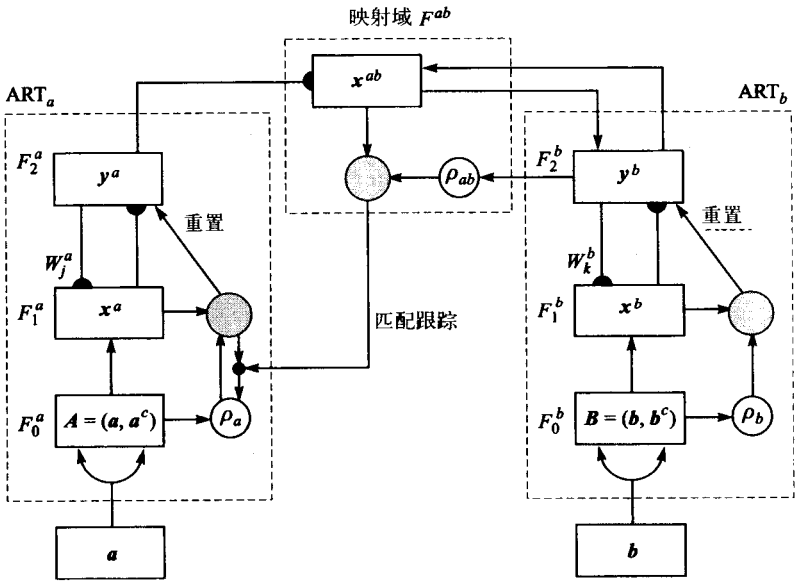



图4-18 模糊ARTMAP结构

## 习题

4.1 生成2000个随机向量$x\in\Re^{2\times1}$的例子，该随机向量的概率密度函数定义如下：

$$p_x(x)=\frac{1}{2}\frac{1}{2\pi\det(\boldsymbol{Q}_1)}\exp\left[-\frac{1}{2}(x-x_1)^T\boldsymbol{Q}_1^{-1}(x-x_1)\right]$$
$$+\frac{1}{2}\frac{1}{2\pi\det(\boldsymbol{Q}_2)}\exp\left[-\frac{1}{2}(x-x_2)^T\boldsymbol{Q}_2^{-1}(x-x_2)\right]$$

其中

$$\boldsymbol{Q}_1=\begin{bmatrix}0.1^2 & 0\\ 0 & 0.4^2\end{bmatrix}\quad x_1=[1\ \ 1]^T$$

$$\boldsymbol{Q}_2=\begin{bmatrix}0.5^2 & 0\\ 0 & 0.3^2\end{bmatrix}\quad x_2=[-1\ \ -1]^T$$

设计有100个神经元的Kohonen SOM，神经元组织在一个正方形和一个六角形网格中实现输入向量样例的分类。画出神经元的最后位置，把它们和输入向量样例的散列图比较。

对于该问题写一个计算机程序来补充Kohonen SOM。

NJ 4.2 生成1000个二维向量$\boldsymbol{X} = [\boldsymbol{x}_1\ \boldsymbol{x}_2\ \cdots\ \boldsymbol{x}_{1000}] \in \Re^{2\times 1000}$，两个分量都选自一个均匀分布。第一分量必须是零均值且方差为3($\sigma^2 = 3$)；第二个分量必须是零均值且有单位方差($\sigma^2 = 1$)[⊖]。

189 ~ 190

(a) 使用如下函数在$x-y$平面内画出随机向量：

```
plot(X(1,:),X(2,:),'+'), axis([-4 4 -4 4])
```

(b) 设计有15个神经元的Kohonen SOM，神经元排列成3×5矩形网格。

(c) 网络收敛需要多少训练回合？

提示：在解这个问题时可以使用MATLAB神经网络工具箱中的`initsm`，`nbman`和`trainsm`这三个函数。

NJ 4.3 (a) 生成两个高斯二维随机向量集。它们都有方差$\sigma^2 = 0.5$。但是，第一个集的中心在(5, 0)，第二个集的中心在(0, 5)。在MATLAB内使用如下方式来生成数据：

```
X1N=sqrt(0.5)*randn(2,1000);
X1N(1,:)=X1N(1,:)+5*ones(1,1000);
X2N=sqrt(0.5)*randn(2,1000);
X2N(2,:)=X2N(2,:)+5*ones(1,1000);
```

提交这些随机向量给一个Kohonen SOM，在确定网络已经达到收敛后观察映射的拓扑排序。假定SOM的梯格结构是有36个神经元的正方形。从这些结果你能推断出什么？

(b) 与部分（a）的问题相似，生成的随机向量取自具有均匀分布的随机数。这两个具有1000个二维向量的集合都有方差$\sigma^2 = 0.5$，并且中心分别设置在(5, 0)和(0, 5)。提交这些向量给一个Kohonen SOM，在确定网络已经达到收敛后观察映射的拓扑排序。再假定SOM的梯格结构是有36个神经元的正方形。从这些结果你能推断出什么？注意：留意你怎样产生随机向量的。参考A.7.4节中均匀分布的细节。

NJ 4.4 与例4.2相似，这个问题探索Kohonen SOM的拓扑排序属性；也就是，一个高维输入空间（三维数据）映射到一个抽象的二维表示。产生三个三维的高斯随机向量集。所有三个向量集都具有方差$\sigma^2 = 0.1$。第一个向量集的中心在(6, 0, 0)，第二个中心在(0, 6, 0)，第三个在(0, 0, 6)。设计一个带有64个神经元的二维正方形梯格结构的Kohonen SOM映射。训练回合的合理数量是多少，才能确保网络收敛？最后的二维映射看起来像什么（画出结果）？你能够推断出什么？

NJ 4.5 与问题4.4相似，但包括附加的高斯数据集。特别地，产生4个随机的高斯向量集，每个都有方差$\sigma^2 = 0.1$。设置这些相继的高斯向量集的中心分别为(7, 0, 7)，(7, 7, 7)，(0, 7, 7)，(0, 0, 7)。下面的MATLAB命令将产生数据：

191

```
%variance = 0.1, centered at (7 0 7)
X1=sqrt(0.1)*randn(3,1000);
X1=detrend(X1')';
X1(1,:)=X1(1,:)+7*ones(1,1000);
X1(3,:)=X1(3,:)+7*ones(1,1000);
%variance = 0.1, centered at (7 7 7)
X2=sqrt(0.1)*randn(3,1000);
X2=detrend(X2')';
X2(1,:)=X2(1,:)+7*ones(1,1000);
X2(2,:)=X2(2,:)+7*ones(1,1000);
X2(3,:)=X2(3,:)+7*ones(1,1000);
%variance = 0.1, centered at (0 7 7)
```

⊖ 参考A.7.4节关于均匀分布的说明。

```
X3=sqrt(0.1)*randn(3,1000);
X3=detrend(X3')';
X3(2,:)=X3(2,:)+7*ones(1,1000);
X3(3,:)=X3(3,:)+7*ones(1,1000);
%variance = 0.1, centered at (0 0 7)
X4=sqrt(0.1)*randn(3,1000);
X4=detrend(X4')';
X4(3,:)=X4(3,:)+7*ones(1,1000);
```

设计一个带有64个神经元的二维正方形梯格结构的Kohonen SOM映射。训练回合的合理数量是多少，才能确保网络收敛？最后的映射看起来像什么（画出结果）？你能够推断出什么？

4.6 判别两个重叠的二维高斯分布模式。即，有两类高斯分布模式，标记为类别1和类别2。事件的两个集合设计为 $C_{x_1}$ 和 $C_{x_2}$，在MATLAB中按如下方式产生两个向量集合。

```
X1=sqrt(0.1)*randn(2,1000);
X2(1,:)=2*randn(1,1000)+5*ones(1,1000);
X2(2,:)=2*randn(1,1000);
```

这将产生两个向量集合。（1）在集合 $C_{x_1}$ 内，对于类别1，是均值为 $\bar{x}_1 = [0 \quad 0]^T$ 和方差为 $\sigma_1^2 = 0.1$的二维向量集。因此，该类别 $p_x(x|C_{x_1})$ 的条件概率密度函数（参看A.7.4节）有均值 $\bar{x}_1$ 和方差 $\sigma_1^2$。（2）在集合 $C_{x_2}$ 内，对于类别2，是一个均值为 $\bar{x}_2 = [5 \quad 0]^T$ 和方差 $\sigma_2^2 = 4$的二维向量集。从而，该类别 $p_x(x|C_{x_2})$ 的条件概率密度函数有均值 $\bar{x}_2$ 和方差 $\sigma_2^2$。

(a) 用不同的符号画出两个随机向量集合，并且确定存在多少重叠的量。

(b) 使用在一个正方形内有25个神经元的Kohonen SOM，充分训练SOM，并且画出权值向量。你能从*xy*平面的构成确定什么？ 192

4.7 在例子4.3中，使用7个四维二值向量决定最终权值向量

$$w_1 = \begin{bmatrix} 1.2996 \\ 0.4952 \\ 0.4952 \\ 0.4848 \end{bmatrix} \text{和} \; w_2 = \begin{bmatrix} -0.1881 \\ 1 \\ 1 \\ 0.3603 \end{bmatrix}$$

（在500个训练回合后）使用表4-1中的LVQ1程序。给定下面四个在训练过程中没有使用过的向量，决定每个向量属于的类别

$$x_{\text{test 1}} = [1, 0, 1, 0]^T \qquad x_{\text{test 2}} = [0, 0, 1, 0]^T$$
$$x_{\text{test 3}} = [0, 1, 0, 0]^T \qquad x_{\text{test 4}} = [1, 1, 0, 1]^T$$

4.8 考虑图4-19描述的分类问题。

(a) 设计一个LVQ网络实现分类。

(b) 按照如下方法测试网络设置的分类边界：

- 在比图4-19中网格更细的网格上生成测试输入点。
- 提交测试输入点给网络。
- 让网络实现分类。
- 产生显示不同分类区域的图。

4.9 给定下面带有相关类型的输入向量集合：

| 向　量 | | 类　型 |
|---|---|---|
| $x_1 = [0, 0, 1]^T$ | （对应于1的二值表示） | 1（奇数） |

| | | |
|---|---|---|
| $x_3 = [0, 1, 0]^T$ | （对应于2的二值表示） | 2（偶数） |
| $x_4 = [0, 1, 1]^T$ | （对应于3的二值表示） | 1（奇数） |
| $x_5 = [1, 0, 0]^T$ | （对应于4的二值表示） | 2（偶数） |
| $x_6 = [1, 0, 1]^T$ | （对应于5的二值表示） | 1（奇数） |

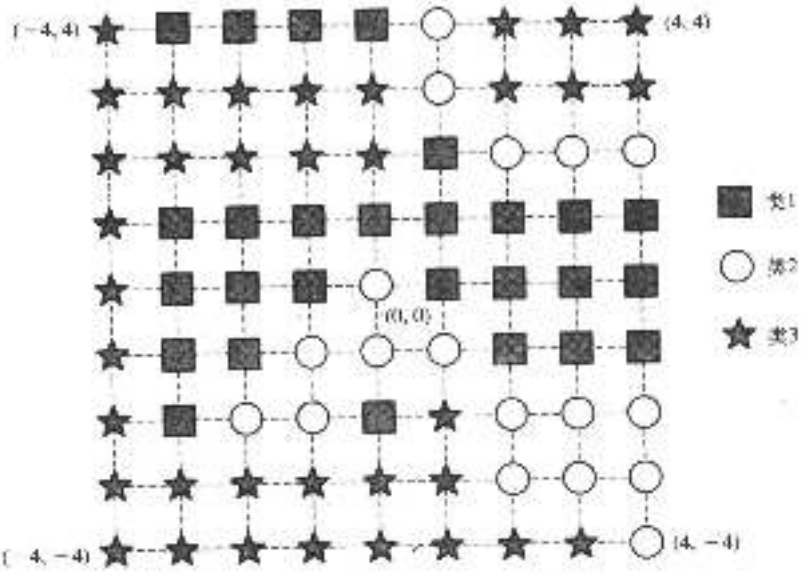



图4-19　非线性可分离性类型的LVQ分类

| 向　量 | | 类　型 |
|---|---|---|
| $x_6 = [1, 1, 0]^T$ | （对应于6的二值表示） | 2（偶数） |
| $x_7 = [1, 1, 1]^T$ | （对应于7的二值表示） | 1（奇数） |

利用上面给出的数据训练一个LVQ网络，你能得到什么结论？

4.10　在完成问题4.3的部分（a）后，使用图4-13中的混合自适应模式分类系统来微调聚类，利用SOM的权值向量作为LVQ网络的输入。对用于训练LVQ网络的SOM权值向量开发一种方法确定恰当分类数。从你的结果能得到什么结论？对于问题4.3的部分（b）重复该过程。

提示：你可以直接使用表4-1的LVQ1程序，或以随机初始化权值向量来修改它。

4.11　利用图4-13所示的混合自适应模式分类系统执行问题4.10中描述的同样过程，使用问题4.5得到的SOM输出结果。

4.12　重做问题4.11，但使用问题4.6的SOM输出结果。

4.13　(a) 在4.4.1节中介绍了ART1网络学习的细节。基于该节的讨论，写一个用于训练ART1网络的算法步骤。

(b) 根据本问题部分（a）中你开发的算法，写一个MATLAB函数实现用于ART1学习的算法。

4.14　本问题是问题4.5的扩展。

(a) 使用例4.5罗列的方法生成字母表的26个字母。例如，字母L的（列）向量表示在MATLAB中可得到如下所示：

```
Lletters= [1 0 0 0 0 ...
           1 0 0 0 0 ...
           1 0 0 0 0 ...
           1 0 0 0 0 ...
           1 0 0 0 0 ...
           1 0 0 0 1 ...
           1 1 1 1 1]';
```

193
|
194

注意：参考有关在MATLAB中描绘字母图像的4.4.1节中的脚注。

(b) 使用在问题4.13（b）中的警戒参数设置为$\rho = 0.3$的ART1 MATLAB函数来分类字母表的26个字母。形成多少个聚类？哪个字母分类到哪个聚类中？增加警戒参数为$\rho = 0.7$。这怎样影响结果？使得所有26个字母有自己的聚类单元的警戒参数最小值是多少？

4.15 图4-20显示9个不同图像，能够通过使用例4.5中描述的相同程序来向量化。如例4.5中，假定黑色像素是1，白色像素是0。

注意：参考有关在MATLAB中描绘字母图像的4.4.1节中的脚注。

(a) 使用问题4.13（b）中你写的ART1 MATLAB函数，用警戒参数的不同值试验，并且讨论结果，你能得到什么结论？

(b) 让警戒参数足够大，使得在ART1网络的输出中创建9个分离的聚类。选择第一个和第五个图案，在图像中锁定10个随机选择的位。提交这些损坏的图像给训练好的网络，网络还能恰当分类图像吗？

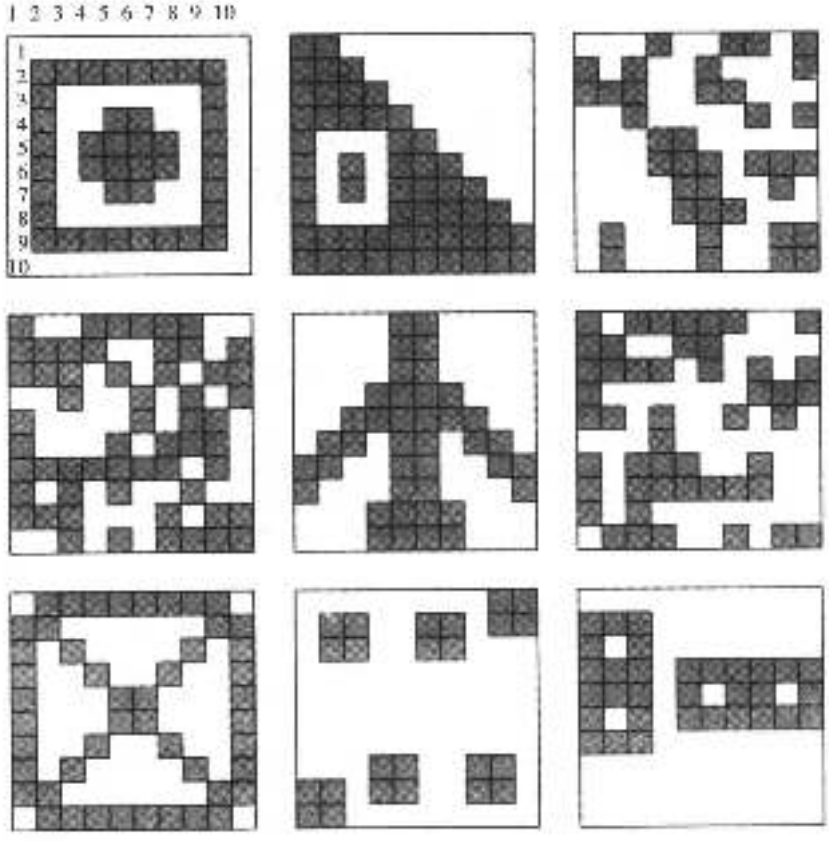



图4-20 在问题4.15中使用ART1神经网络的聚类图像

## 参考文献


1. T. Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps," *Biological Cybernetics*, vol. 43, 1982, pp. 59–69. Reprinted in 1988, Anderson and Rosenfeld [2], pp. 511–21.
2. J. A. Anderson and E. Rosenfeld, eds., *Neurocomputing: Foundations of Research*, Cambridge, MA: M.I.T. Press, 1988.
3. D. J. Willshaw and C. von der Malsburg, "How Patterned Neural Connections Can Be Set Up by Self-Organization," *Proceedings of the Royal Society of London, Series B*, vol. 194, 1976, pp. 431–45.
4. T. Kohonen, "The Self-Organizing Map," *IEEE Proceedings*, vol. 78, 1990, pp. 1464–80.
5. T. Kohonen, *Self-Organization and Associative Memory*, 3rd ed., Berlin: Springer-Verlag, 1989.
6. J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Redwood City, CA: Addison-Wesley, 1991.
7. Y. Linde, A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, vol. COM-28, 1980, pp. 84–95.
8. R. M. Gray, "Vector Quantization," *IEEE Acoustics, Speech, and Signal Processing Magazine*, vol. 1, 1984, pp. 4–29.
9. N. M. Nasrabadi and R. A. King, "Image Coding Using Vector Quantization: A Review," *IEEE Transactions on Communications*, vol. 36, 1988, pp. 957–71.
10. S. P. Luttrell, "Hierarchical Vector Quantization," *IEE Proceedings (London)*, vol. 136, Part I, 1989, pp. 405–13.
11. A. Buzo, A. H. Gray, R. M. Gray, and J. D. Markel, "Speech Coding Based upon Vector Quantization," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-28, 1980, pp. 562–74.
12. P. C. Mahalanobis, "On the Generalized Distance in Statistics," *Proceedings of Indian National Institute of Science (Calcutta)*, 1936, pp. 49–55.
13. F. Itakura and S. Saito, "Analysis Synthesis Telephony Based upon Maximum Likelihood Method," *Reports of the 6th International Congress on Acoustics*, ed. Y. Kohasi, Tokyo, Japan, 1968, C-5-5, pp. C17–20.
14. F. Itakura, "Maximum Prediction Residual Principle Applied to Speech Recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, 1975, pp. 67–72.
15. A. Gersho, "On the Structure of Vector Quantizers," *IEEE Transactions on Information Theory*, vol. IT-28, 1982, pp. 157–66.
16. T. Kohonen, "Learning Vector Quantization for Pattern Recognition," *Technical Report TKK-F-A601*, Helsinki University of Technology, Finland, 1986.
17. T. Kohonen, "Improved Versions of Learning Vector Quantization," *Proceedings of the International Joint Conference on Neural Networks*, San Diego, CA, vol. 1, 1990, pp. 545–50.

195
↓
196

18. J. S. Baras and A. LaVigna, "Convergence of Kohonen's Learning Vector Quantization," *International Joint Conference on Neural Networks*, San Diego, CA, vol. 3, 1990, pp. 17–20.
19. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, New York: Wiley, 1973.
20. G. A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," *Computer Vision, Graphics, and Image Processing*, vol. 37, 1987, pp. 54–115.
21. B. Moore, "ART1 and Pattern Clustering," *Proceedings 1988 Connectionist Models Summer School, Pittsburgh*, eds., D. Touretsky, G. Hinton, and T. Sejnowski, San Mateo, CA: Morgan Kaufmann, 1988, pp. 174–85.

22. N. K. Bose and P. Liang, *Neural Networks Fundamentals with Graphs, Algorithms, and Applications*, New York: McGraw-Hill, 1996.
23. G. A. Carpenter and S. Grossberg, "ART2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns," *Applied Optics*, vol. 26, 1987, pp. 4919–30.
24. G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System," *Neural Networks*, vol. 4, 1991, pp. 759–71.
25. L. A. Zadeh, "Fuzzy Sets," *Information and Control*, vol. 8, 1965, pp. 338–53.
26. G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps," *IEEE Transactions on Neural Networks*, vol. 3, 1992, pp. 698–713.
27. G. A. Carpenter, S. Grossberg, and J. H. Reynolds, "ARTMAP: Supervised Real-Time Learning and Classification of Non-Stationary Data by a Self-Organizing Neural Network," *Neural Networks*, vol. 4, 1991, pp. 565–88.
28. S. L. Salzberg, "Learning with Generalized Nested Exemplars," Ph.D. thesis, Technical Report TR-14-89, Cambridge, MA: Department of Computer Science, Harvard University, 1989.
29. S. L. Salzberg, "A Nearest Hyperrectangular Learning Method," *Machine Learning*, vol. 6, 1991, pp. 251–76.
30. P. Simpson, "Fuzzy Min-Max Classification with Neural Networks," *Heuristics: Journal of Knowledge Engineering*, vol. 4, 1991, pp. 1–9.
31. M. J. Healy, T. P. Caudell, and S. D. G. Smith, "A Neural Architecture for Pattern Sequence Verification through Inferencing," *IEEE Transactions on Neural Networks*, vol. 4, 1993, pp. 9–20.
32. G. Han, "A Fuzzy Neural Architecture with Binary and Analog Inputs for Supervised Learning through Inferencing," Ph.D. thesis, Melbourne, FL: Electrical Engineering Program, Florida Institute of Technology, 1993.
33. F. M. Ham, G. Han, and L. V. Fausett, "Fuzzy LAPART: A Neural Architecture for Supervised Learning through Inferencing for Stable Category Recognition," *Journal of Artificial Neural Networks*, vol. 2, 1995, pp. 241–64.
34. G. A. Carpenter, "Distributed Learning, Recognition, and Prediction by ART and ARTMAP Networks," *Neural Networks*, vol. 10, 1997, pp. 1473–94.

197

# 第5章 递归网络和时间前馈网络

## 5.1 概述

本章涉及递归神经网络。通常，递归网络视为拓扑结构上含有闭合回路的网络。虽然本章考虑的时间前馈网络也是递归网络，但我们将对时间前馈网络和非多层前馈网络加以区分。因此本章包括两部分：继5.2节概述递归神经网络之后，本章第一部分将介绍离散时间霍普菲尔德网络（5.3节）、模拟退火（5.4节）和玻尔兹曼机（5.5节）。继5.6节概述时间前馈网络之后，第二部分将介绍简单递归网络（simple recurrent network, SRN）（也称Elman网）（5.7节）、时延网络（5.8节）和分布式时滞前馈神经网络（5.9节）。

## 5.2 递归神经网络概述

先前研究的前馈网络完成从输入空间到输出空间的固定权值映射。因为网络有固定权值，从而任意神经元的状态仅由该单元的输入决定，而不是由初始状态和过去的状态决定。由于网络不具备动态性，网络神经元对初始状态和过去状态的无关限制了这些网络。为了使初始
198 状态和过去状态能够介入一系列处理，*递归神经网络*利用反馈方法。由于递归神经网络也以使用非线性处理单元为其特征，因此，这样的网络是非线性动态系统。递归网络的另一重要特征是对个别设备故障的相对不灵敏性（容错）。霍普菲尔德称这一特性为*软故障设备*[1]。由于递归网络有反馈路径，它们是顺序的而非组合的；即网络能演示时间性行为。这些网络可能是全连接的。换言之，网络神经元之间所有可能的连接都是允许的。而且，递归神经网络的连接权值可以是对称或不对称的。在对称情况下，连接权值$w_{ij} = w_{ji}\ \forall i, j$，而在非对称情况下，$w_{ij} \neq w_{ji}\ \forall i, j$。

在对称情况下，网络总是收敛于稳定点吸引子（稳定平衡点或状态）。但是这些网络不能提供模式的时间顺序。在非对称情况下，网络的动态性能除了显示出稳定状态外，还显示出极限环和混沌[2]，并且如果恰当选择权值，网络就能产生并储存时空模式。这些类型的网络在高级智能系统中（如符号推理）起着重要作用。递归神经网络的发展多次被视为是受到统计力学概念的启发[3, 4]。

## 5.3 霍普菲尔德联想记忆

在霍普菲尔德的经典论文[1]中，讲述到由大量简单元素（神经元）构成的物理系统能展示出*集体涌现特性*。简单说，系统的集体特性不能从单元素出现，但是能从系统的局部单元相互作用中出现。他所描述的具有集体特性的模型可以产生按内容寻址的存储器，该存储器能根据部分信息正确地产生完整存储。他还描述了其他的集体涌现特性，如相似性识别、泛化、分类、纠错和时序记忆的能力。

我们讨论标准离散时间霍普菲尔德神经网络。由于具有反馈连接，故认为该标准离散时间网络是递归网络，而霍普菲尔德网络也可以看成是非线性*联想记忆*（参看3.2节）或*内容可寻址存储器*。打算利用该网络执行数据储存和检索的功能。但是，该网络在动态稳定的环境中储存信息。对有噪声（或不完整）的储存模式输入的情况，网络将检索储存在存储器中的

模式。内容可寻址存储器即使对于提交的不完整的或有错误的输入数据，也能可靠地检索存储器模式，在这个意义上内容可寻址存储是纠错存储器。

作为内容可寻址储存器（CAM），霍普菲尔德网络的基本特性是执行映射功能。然而，这是动态映射。在讨论霍普菲尔德网络的细节之前需要解释一些基本概念。吸引子是系统从初始条件开始随时间演变到的状态。每一吸引子有与之关联的初始条件集。初始条件集开始演变到特定的吸引子终止。特定吸引子的初始条件集称为吸引盆。在状态空间中，如果吸引子是单点，则称为固定点。吸引子可能有更复杂的结构，如极限环。 199

原型状态（原型记忆）$\phi_h$可以由动态系统的固定（稳定）点$\sigma_h$表示。因此$\phi_h$ ($h = 1, 2, \cdots, r$)可以映射到网络的稳定点$\sigma_h$上。该映射可表示为 $\phi_h \rightleftarrows \sigma_h$，其中向前的方向（由左到右）代表编码过程，向后的方向（由右到左）代表解码过程。因此，霍普菲尔德网络是异步非线性动态系统，系统的相空间由稳定点构成，这些稳定点是网络原型状态（或原型记忆）。这就是霍普菲尔德在论文中讨论的集体出现特性[1]。图5-1解释了霍普菲尔德联想记忆完成的状态空间编码/解码过程。在回忆过程中，传送模式给网络。假定该模式包含属于网络原型记忆之一的信息。输入模式也可仅包含与原型记忆有关的部分信息。对输入的响应为相空间的开始点，如果开始点“接近”记忆检索的稳定点，则动态系统将随时间演变并且收敛于该记忆状态（即系统状态的相空间流收敛于该记忆状态）。从而动态系统产生适当记忆。因此，霍普菲尔德神经网络执行在动态稳定环境中储存信息的功能。

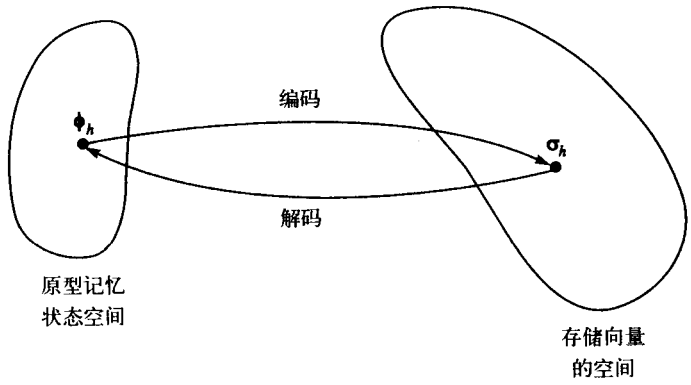



图5-1 用霍普菲尔德相关记忆神经网络执行编码/解码的过程

离散时间霍普菲尔德网络的神经元在架构上采用了McCulloch-Pitts模型[5]，我们仅考虑对称硬限制激活函数（参照2.3节）。因此，网络任意时间的状态只能是+1或－1。运用图2-12所示的神经元离散时间霍普菲尔德模型，可以构建图5-2所示的霍普菲尔德神经网络。 200

对于图5-2中的每一神经元，线性组合器的输出可表示为

$$v_i = \sum_{j=1}^{n} w_{ij}x_j - \theta_i = \boldsymbol{w}_i^T\boldsymbol{x} - \theta_i \quad \text{对于} \quad i = 1, 2, \cdots, n \quad (n\text{个神经元}) \tag{5-1}$$

其中$\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T$是网络的状态（参照A.2.12节），$\theta_i$是外部应用阈值。对于$i = 1, 2, \cdots, n$，每一线性组合器输出传送给对称硬限制激活函数和单元延迟元素。单元延迟输出$x_i$ ($i = 1, 2, \cdots, n$)是反馈给神经元的输入。但例外的是神经元的输出不反馈给自己。因此，图5-2中$i = j$时$w_{ij} = 0$。每一神经元的状态可表示为

$$x_j = \operatorname{sgn}(v_i) = \begin{cases} +1 & \text{对于} v_i > 0 \\ -1 & \text{对于} v_i < 0 \end{cases} \tag{5-2}$$

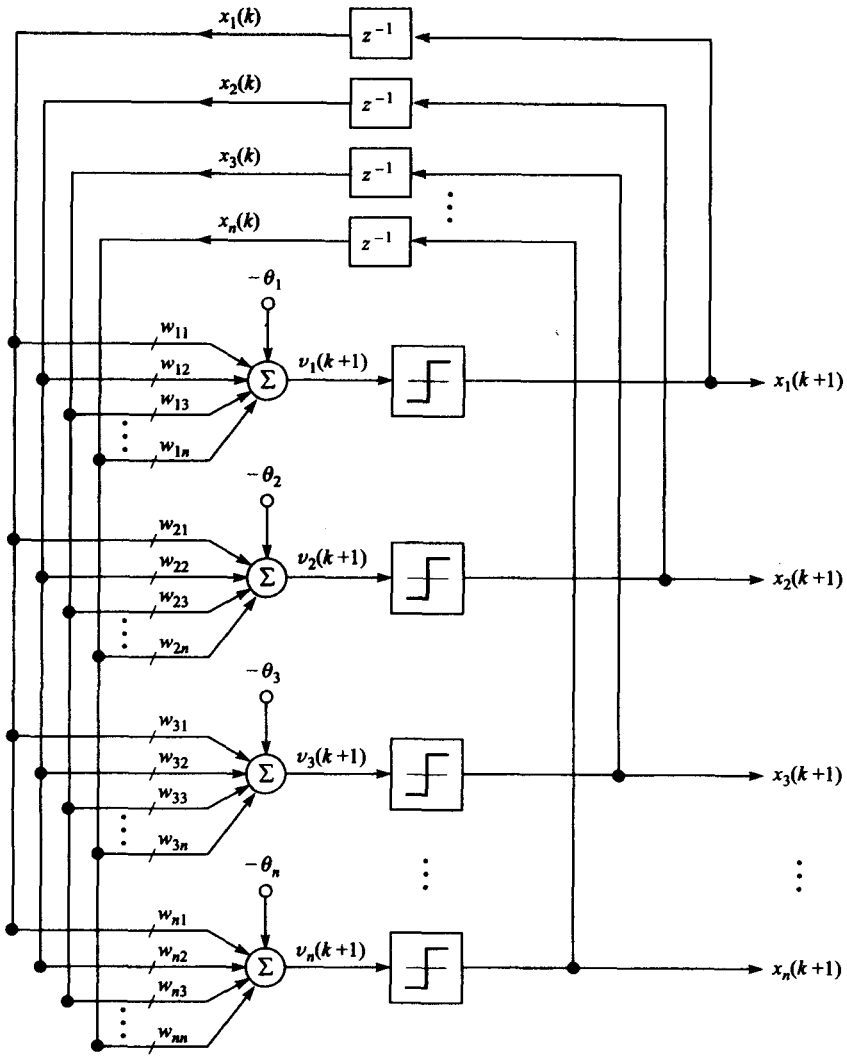



图5-2　离散时间霍普菲尔德神经网络（$z^{-1}$ ■单位延迟）。假设$w_{ij} = w_{ji}$（对称权值矩阵），对于$i = j$有$w_{ij} = 0$（也就是，自身没有神经元输出反馈，或没有“自我闭环”）

其中$j = 1, 2, \cdots, n$，$\mathrm{sgn}(\cdot)$是符号函数（参照2.3节）。如果$v_i = 0$，则将$x_j$的值定义为它的先前状态。可以将阈值向量写为$\boldsymbol{\theta} = [\theta_1, \theta_2, \cdots, \theta_n]^T$。因此，式（5-1）的向量矩阵形式为

$$\boldsymbol{v} = \boldsymbol{W}\boldsymbol{x} - \boldsymbol{\theta} \tag{5-3}$$

其中网络权值矩阵为

$$\boldsymbol{W} = \begin{bmatrix} 0 & w_{12} & w_{13} & w_{14} & \cdots & w_{1n} \\ w_{21} & 0 & w_{23} & w_{24} & \cdots & w_{2n} \\ w_{31} & w_{32} & 0 & w_{34} & \cdots & w_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ w_{n-11} & w_{n-12} & w_{n-13} & \cdots & 0 & w_{n-1n} \\ w_{n1} & w_{n2} & w_{n3} & \cdots & w_{nn-1} & 0 \end{bmatrix} \tag{5-4}$$

式（5-4）中每一行是图5-2所示的每一神经元的相关权值向量。网络输出的向量矩阵形式可写为

$$\boldsymbol{x}(k+1) = \text{sgn}[\boldsymbol{W}\boldsymbol{x}(k) - \boldsymbol{\theta}] \tag{5-5}$$

或标量形式为

$$x_i(k+1) = \text{sgn}\left(\sum_{j=1}^{n} w_{ij}x_j - \theta_i\right) \tag{5-6}$$

其中$i = 1, 2, \cdots, n$。有两个与霍普菲尔德网络相关的基本操作阶段：*储存阶段*和*回忆阶段*。

在储存阶段，联想记忆（内容可寻址记忆）是根据3.2.2节关于相关矩阵记忆的外积规则建立的。假定$r$个原型记忆的集为$\{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \cdots, \boldsymbol{\phi}_r\}$，网络权值矩阵根据下式计算

$$\boldsymbol{W} = \frac{1}{n}\sum_{h=1}^{r}\boldsymbol{\phi}_h\boldsymbol{\phi}_h^T - \frac{r}{n}\boldsymbol{I} \tag{5-7}$$

201
202

其中比例因子$1/n$是为了方便，第二项$(r/n)\boldsymbol{I}$为从原型记忆向量的外积和中减去，为满足$w_{ij} = 0(i = j)$。如3.2.2节所述，式（5-7）是Hebb学习的一种形式（参照2.8.2节）。CAM（内容可寻址记忆器）建立之后，回忆阶段包括传送测试输入向量$\boldsymbol{x}' \in \Re^{n\times 1}$给网络，初始化具有未知输入值的网络状态$\boldsymbol{x}(k)$，即$\boldsymbol{x}(k)|_{k=0} = \boldsymbol{x}(0) = \boldsymbol{x}'$。运用表达式（5-6），状态向量$\boldsymbol{x}(k)$元素随机地（异步地）一次更新一个，直到向量元素没有明显变化为止。当满足这个条件时，这个稳定（平衡）状态（或系统的相空间固定点）$\boldsymbol{x}_e$就是网络输出。

离散时间霍普菲尔德神经网络有很多种，但是基于上面的讨论，其中一种离散时间霍普菲尔德神经网络的算法的操作细节概括如下：

**离散时间霍普菲尔德网络训练算法**

**步骤1** 给定原型记忆集$\{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \cdots, \boldsymbol{\phi}_r\}$，运用式（5-7）的外积规则，网络的突触权值可根据下式计算：

$$w_{ij} = \begin{cases} \dfrac{1}{n}\displaystyle\sum_{h=1}^{r}\phi_{hi}\phi_{hj} & i \neq j \\ 0 & i = j \end{cases} \tag{5-8}$$

**步骤2** 用$\boldsymbol{x}'$表示给定的未知输入向量，在$k = 0$时设置网络的状态$\boldsymbol{x}(k)$初始化霍普菲尔德网络为$\boldsymbol{x}'$，即

$$\boldsymbol{x}(0) = \boldsymbol{x}' \tag{5-9}$$

**步骤3** 网络的状态元素$\boldsymbol{x}(k)$根据式（5-6）进行异步更新，即：

$$x_i(k+1) = \text{sgn}\left(\sum_{j=1}^{n} w_{ij}x_j - \theta_i\right) \tag{5-10}$$

这种替代过程一直进行到状态向量元素不再改变为止。当满足这一条件时，网络输出平衡状态，即式（5-10）产生

$$\boldsymbol{x} = \boldsymbol{x}_e \tag{5-11}$$

□

与霍普菲尔德网络相关的主要问题之一是可能产生*伪平衡状态*[⊖]（或伪吸引子）[4, 6]。这样的稳定平衡状态（或者系统相空间的固定点）并不是原型记忆设计集的一部分。在霍普菲尔德网络中存在伪吸引子的原因有几点：（1）伪吸引子产生于奇数个模式的线性组合[6]；（2）对于存储在内容可寻址存储器中的大量原型记忆，能量函数存在局部最小值。这些局部

⊖ 伪平衡状态也指镜旋转状态[7, 8]。

最小值与网络结构的任何原型记忆部分无关。(3) 由霍普菲尔德网络定义的对称能量函数
203 $\mathscr{E}(x)$也能产生伪吸引子。在Li et al.[9]中，分析了与霍普菲尔德神经网络基本结构相同的一类网络。这篇论文中所谈及的设计方法是在一阶线性常微分方程组基础上，它的状态空间定义在闭式超立方体上。当这些方程组的解存在于超立方的边界上时，称系统为饱和状态。前面讲述的方法比著名的霍普菲尔德模型[9]更易于分析、综合和实施。设计方法的一个主要优点是使伪吸引子数量达到最小。如MATLAB神经网络工具箱[10]有建立在该方法基础上的函数newhop。函数newhop利用所有输入目标向量（网络记忆的模式，仅以双极形式）来设计网络。

因为霍普菲尔德网络具有对称权值且不具有神经元自环，可以定义能量函数，或李雅普诺夫函数（参照A.4节）[11]。同样因为对稳定平衡状态的收敛流有CAM的相关特征，因此，希望有一个方法来执行收敛分析。确保状态空间流稳定的数学条件是$w_{ij} = w_{ji}(i \neq j)$，并且$w_{ij} = 0$ $(i = j)$（权值对称，且不存在神经元自环）。收敛证明来自所选择的适当的能量函数，该能量函数总是随任何状态变化（单调）下降的。离散时间霍普菲尔德神经网络的能量函数由[11, 12]可写为

$$\mathscr{E} = -\frac{1}{2}\sum_{\substack{i=1\\ i\neq j}}^{n}\sum_{j=1}^{n} w_{ij}x_i x_j - \sum_{i=1}^{n} x_i' x_i + \sum_{i=1}^{n}\theta_i x_i \tag{5-12}$$

其中$x$是网络的状态，$x'$是传送给网络的外部应用输入，$\boldsymbol{W} = [w_{ij}]$是突触权值矩阵，$\boldsymbol{\theta}$是阈值向量。由于第$i$个神经元$x_i$的状态改变表示为$\Delta x_i$，因此，能量函数$\mathscr{E}$的变量$\Delta\mathscr{E}$表示为

$$\Delta\mathscr{E} = -\left(\sum_{\substack{j=1\\ j\neq i}}^{n} w_{ij}x_j + x_i' - \theta_i\right)\Delta x_i \tag{5-13}$$

仅当一次仅有一个神经元改变其状态时，关系式（5-13）有效。根据霍普菲尔德网络的运算，网络任意状态下的符号变化由式（5-13）括号内项的相同符号变化完成。因此，能量函数的改变$\Delta\mathscr{E}$总是负的，从而能量总是减少的（这暗指渐近稳定系统），最终达到不再随着时间明显变化的系统稳定状态（参照A.4节）。换言之，霍普菲尔德网络的运算导致能量函数单调递
204 减，并且网络状态一直变化，直到能量带达到局部极小。相空间的吸引子（吸引盆）与能量带的局部极小点相关。这些吸引子符合网络的指定存储。然而，正如前面所提及的，也存在伪吸引子。对于没有外部应用输入时，能量函数可定义为

$$\mathscr{E} = -\frac{1}{2}\sum_{\substack{i=1\\ i\neq j}}^{n}\sum_{j=1}^{n} w_{ij}x_i x_j \tag{5-14}$$

能量变为

$$\Delta\mathscr{E} = -\left(\sum_{\substack{j=1\\ j\neq i}}^{n} w_{ij}x_j\right)\Delta x_i \tag{5-15}$$

霍普菲尔德网络的存储容量由霍普菲尔德根据实验确定[1]。合理精确地存储和回忆的双极模式的极大值近似为

$$P_s \approx 0.15n \tag{5-16}$$

其中$n$是网络中神经元的个数。然而，最大限度的精确回忆大部分原型记忆的需要使网络所需

的极大存储容量[4]为

$$P_s = \frac{n}{2\ln n} \tag{5-17}$$

此外，如果要使所有的原型记忆数$r$都能够完美回忆，极大存储容量将小于式（5-17）。在这种情况下，要求右边获得$nr$位（假定99%概率），并且极大存储容量由[4]给定如下

$$P_s = \frac{n}{4\ln n} \tag{5-18}$$

**例5.1** 考虑一个经典的例子，它阐明了霍普菲尔德网络的运算。图5-3给出了带有固定权值的3个神经元。每个神经元的阈值假设为0。网络可能有8种可能的双极状态。权值矩阵能够直接从图5-3的状态图中得出如下

$$\boldsymbol{W} = \begin{bmatrix} 0 & -2/3 & 2/3 \\ -2/3 & 0 & -2/3 \\ 2/3 & -2/3 & 0 \end{bmatrix} \tag{5-19}$$

它满足对称条件和零对角线元素的要求。稳定性条件要求下式成立

$$\boldsymbol{x} = \operatorname{sgn}(\boldsymbol{W}\boldsymbol{x} - \boldsymbol{\theta}) \tag{5-20}$$

然而，8种可能双极向量仅有2种满足稳定性条件，即，[−1, 1, −1]和[1, −1, 1]。当它们提交给网络时，剩余的状态将转换到稳定状态。图5-3表明了这点。式（5-19）的权值矩阵由两个稳定向量组成（即原型记忆）。也就是说，使用式（5-7）能够写出下式 205

$$\begin{aligned} \boldsymbol{W} &= \frac{1}{3}\left(\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}[-1,1,-1] + \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}[1,-1,1]\right) - \frac{2}{3}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & -2/3 & 2/3 \\ -2/3 & 0 & -2/3 \\ 2/3 & -2/3 & 0 \end{bmatrix} \end{aligned} \tag{5-21}$$

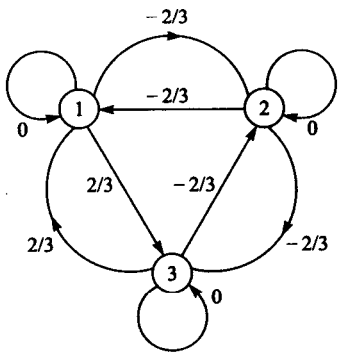



图5-3 带有固定权值的3神经元霍普菲尔德网络的状态图

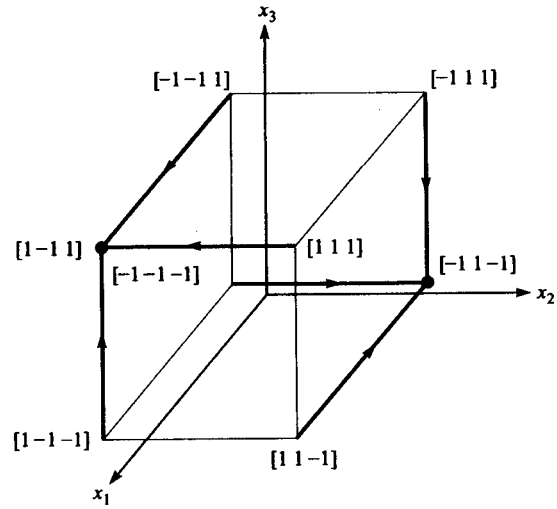



图5-4 三神经元霍普菲尔德网络的状态流。立方体两角的点表明两种稳定状态

另一种方法是观察霍普菲尔德网络纠错能力。从图5-4中，我们发现，如果网络提交[-1,-1,1]，[1,-1,-1]或[1,1,1]时，网络收敛到[1,-1,1]。在上述二种情况下，每个向量中有一位误差，通过网络"修正"产生原型向量[1,-1,1]。对于另一原型向量[-1,1,-1]有类似的情形，参看图5-4。

使用式（5-14）中带有图5-3所示的突触权值网络的表达式，能量函数可以写成

$$\mathscr{E} = \frac{2}{3}(x_1x_2 - x_1x_3 + x_2x_3) \tag{5-22}$$

对于系统的8种可能状态，两种稳定原型记忆产生式（5-22）中能量函数的最小值。表5-1给出了能量函数计算出的八个值，它与能量函数$\Delta\mathscr{E}$一起改变，其中，在网络中使用式（5-15）来改变单个神经元的能量函数$\Delta\mathscr{E}$。

**例5.2**　阐明霍普菲尔德网络识别字符的能力。图5-5显示MATLAB中产生的5个字符，每个字符由一个$12\times12$双极数字阵列组成，A + 1是黑色的，-1是白色的，霍普菲尔德网络需要$N = 144$个带有$N^2 = 20736$个突触权值的神经元（虽然这些权值中的144个与权值矩阵的对角线元素相关的权值是0）。网络中每个神经元的阈值假设为0，每个字符向量化成单一图案。也就是说，如果霍普菲尔德考虑一个矩阵$X_k \in \mathfrak{R}^{12\times12}$ $(k = 1, 2, \cdots, 5)$作为每个字符的矩阵形式，那么字符的向量形式为：

206

$$\boldsymbol{\phi}_k = \mathrm{vec}(X_k) \in \mathfrak{R}^{144\times1} \tag{5-23}$$

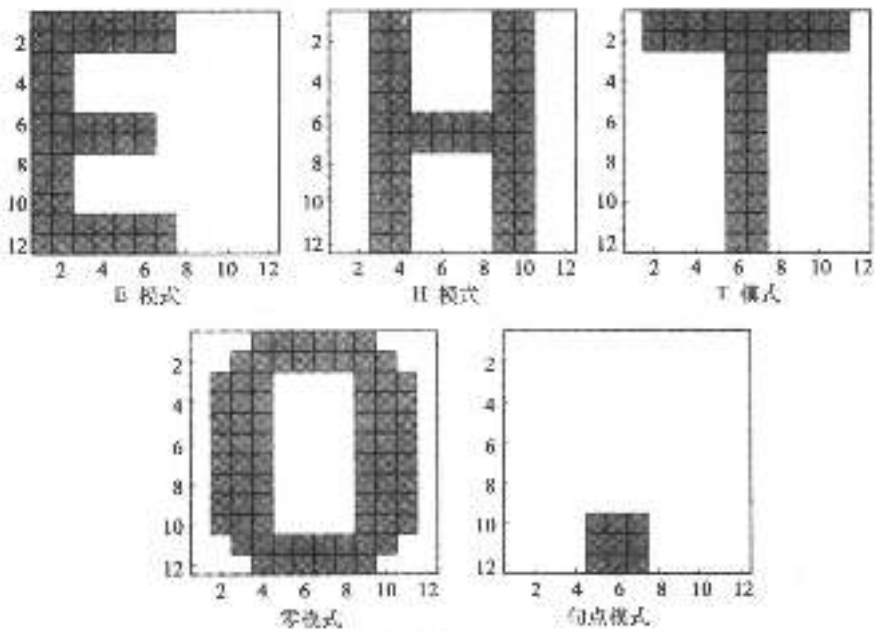



图5-5　通过霍普菲尔德神经网络识别的5个字符

其中操作$\mathrm{vec}(\cdot)$形成一个来自矩阵的向量，该向量是通过把该矩阵的列"堆栈"形成的（参照A.2.17节）。这些字符按照此种方式创建，可以很好地保证网络性能[13]。通过观察向量$\boldsymbol{\phi}_k$ $(k = 1, 2, \cdots, 5)$的内积可以实现这一点。越是邻近的向量越有可能相互正交，识别的效果也会更好。突触权值矩阵通过使用5个来自式（5-23）的原型向量和式（5-7）中的表达式来创建。在图5-6中，权值矩阵$W \in \mathfrak{R}^{144\times144}$表示一个灰度图。

表5-1 例5.1的能量函数值

| 网络状态 | | | 能量函数 $E$ | 相对于初始状态(1, 1, 1)能量函数的改变$\Delta E$ |
|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | | |
| 1 | 1 | 1 | 2/3 | |
| −1 | 1 | 1 | 2/3 | 0 |
| 1 | 1 | 1 | −2 | −8/3 |
| −1 | −1 | 1 | 2/3 | 8/3 |
| 1 | 1 | −1 | 2/3 | 8 |
| −1 | 1 | −1 | 2 | 8/3 |
| 1 | −1 | −1 | 2/3 | 8/3 |
| −1 | −1 | −1 | 2/3 | 0 |

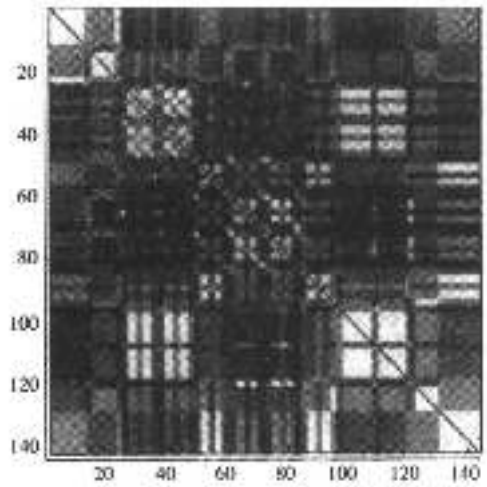



图5-6 例5.2的突触权值矩阵表示为一个灰度图。权值较小的元素对应的像素颜色深，权值较大的元素对应的像素颜色浅。权值矩阵的零对角线在图像中是非常明显的

207
|
208

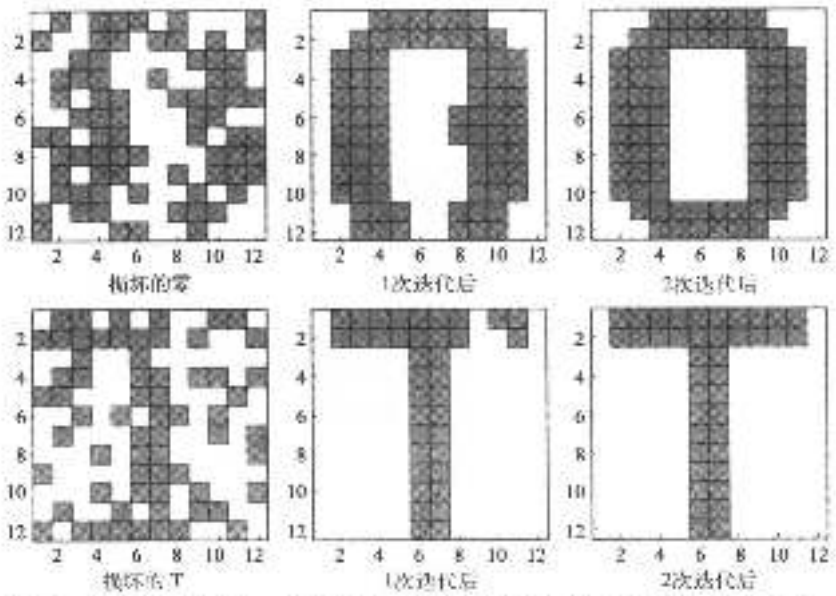


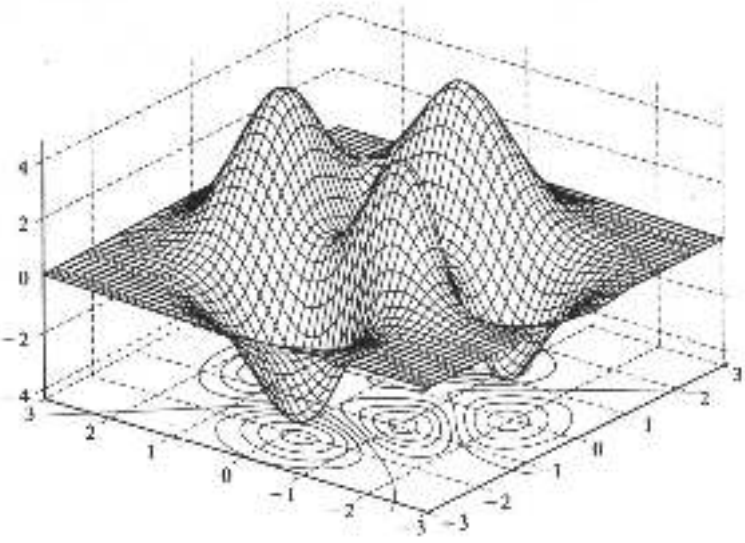

图5-7 例5.2的模拟结果。在两种情况下两个字符在两次迭代后被正确地识别

霍普菲尔德网络的存取首先要保证5个原型向量（字符）能够从CAM中回忆。应用式（5-20）的稳定性条件($\theta = 0$)，在每种情况下，字符能够回忆或由存储在霍普菲尔德网络中的信息正确地恢复。为了阐明霍普菲尔德网络的纠错能力，图5-5中的每个字符被噪声破坏。也就是说，每个字符随机“扭曲”一些“位”（也就是，一个随机极性变化）。位误差率（BER）是30%（或一位扭曲的概率是0.3）。所以，在图案中平均有30%的位的符号发生变化（也就是，从一个黑色像素到一个白色像素的扭曲，反之亦然）。图5-7给出噪声损坏的两个字符的重构结果。在这两种情况下，通过两次迭代，使（5-20）式的阈值都等于0，这样字符能够完成重构（或被回忆）。剩余字符仅通过一次迭代就能够完全重构。当位误差率超过30%时，结果是零乱的。大多数情况下，字符不能够完全重构，或识别了错误字符。

## 5.4 模拟退火

在5.3节中，我们发现霍普菲尔德神经网络局部能够在网络达到局部最小值时恢复存储模
209 式。然而，大多数情况下要求网络达到全局最小值，例如最优化问题。对于霍普菲尔德网络，使用梯度下降规则来检索网络的存储模式，最终收敛于一个局部最小值，然后保持不变（这样术语陷入一个局部最小值）。如果对于网络的全局能量相关的目标（代价）函数求一个全局最小值，可以在路径搜索中仅使用局部信息，局部随机性必须考虑增加到梯度下降方法中，以增加搜索全局最小值的概率。这就涉及下面将要讨论的模拟退火算法。

模拟退火最初由Kirkpatrick et al.[14]提出，可以用来解决组合最优化问题，或NP完全（非确定多项式时间完全）问题。模拟退火[14-19]不同于标准迭代改善最优化方法（梯度下降法）。在该方法中，搜索全局最小值时随机性被考虑进去。这使得系统“跃”出局部最小值（见图5-8），并且继续搜索全局最小值。若没有“跳出”这个局部最小值，那么这个局部最小值将是迭代改善搜索方法的最后结果。因此，算法并不一定非要粘滞于局部最小值，而是可以跳出局部最小值到达全局最小值，在它们之间至少要跨越（或征服）一个山脊。模拟退火的“跳出”操作是算法中温度参数变化的结果，并且在非零温度下从局部最小值跳出来的转化总是切实可行的。模拟退火的另一个特征是它展示一个自适应分治性质，系统状态的大致粗略特征将在较高的温度下表现出来，而系统状态的细节特征在较低的温度下表现。




210 图5-8　含有相多个极小值和极大值的两个变量的函数的示意图

模拟退火算法将一个含有大量系统变量的代价函数最优化问题和一个在不同有限温度的热平衡内具有多自由度的物理系统之间做了一个类比，就像在统计力学[20]中发现的一样。给定一个描述，计算系统能量并找到它的低温状态，这是一个与组合最优化问题类似的最优化问题。在固体物理学中[21]，退火这个名词涉及在加热一个固体超过熔点变成液体状态，然后（慢慢地）降低温度以便在晶格内所有微粒重排序的物理过程。当它冷却时，液态金属微粒寻找最小能量配置（或状态）。冷却阶段缓慢进行（与淬火相反），最终在晶格上瑕疵最小化。晶格的无瑕疵状态满足结构的全局最小能量状态。模拟退火过程由两个阶段组成，首先在有效高温下融化系统，使其最优化，然后缓慢降低温度直到系统冻结（也就是说，没有进一步的变化发生）。

在统计力学中一个能量函数$\mathscr{E}(\boldsymbol{x})$可以定义为，度量在一个给定状态$\boldsymbol{x}$下，原子集合（或一个物理系统）的热能，其中$\boldsymbol{x} \in \Phi$（所有可能点的集合）。一个物理学的基本结论是在热平衡下，每个状态$\boldsymbol{x}$的发生概率如下

$$\Pr(\boldsymbol{x}) = \frac{1}{Z} e^{-\mathscr{E}(\boldsymbol{x})/k_B T} \tag{5-24}$$

其中$k_B$是玻尔兹曼常数[$k_B = 1.3806 \times 10^{-23}$J/K（焦耳/开氏温标)]，$T$是温度，$Z$是分割函数。分割函数定义为

$$Z = \mathrm{Tr}(e^{-\mathscr{E}(\boldsymbol{x})/k_B T}) \tag{5-25}$$

其中Tr表示在采样系统[14]中的原子所有可能配置种类的总数。分割函数将限制式（5-24）中的$\Pr(\boldsymbol{x})$在0到1之间，式（5-24）可以写成

$$\Pr(\boldsymbol{x}) = \frac{e^{-\mathscr{E}(\boldsymbol{x})/k_B T}}{\mathrm{Tr}(e^{-\mathscr{E}(\boldsymbol{x})/k_B T})} \tag{5-26}$$

即所谓的玻尔兹曼-吉布斯（Boltzmann-Gibbs）分布。定义一个与随机状态变化相关的概率$\Pr(\boldsymbol{x} \rightarrow \boldsymbol{x}_p)$的集合，即从任意状态$\boldsymbol{x}$到某个紊乱状态$\boldsymbol{x}_p$的变化。变化概率集合$\Pr(\boldsymbol{x} \rightarrow \boldsymbol{x}_p)$可能无法保证达到热平衡，但既然我们关注热平衡，并且不限制系统的循环或混乱行为，就需要一个充分条件$\Pr(\boldsymbol{x} \rightarrow \boldsymbol{x}_p)$来确保使其达到热平衡。确保$\Pr(\boldsymbol{x} \rightarrow \boldsymbol{x}_p)$可以到达热平衡的充分条件是从$\boldsymbol{x}$到$\boldsymbol{x}_p$的变化概率平均等于从$\boldsymbol{x}_p$到$\boldsymbol{x}$的变化概率。可以写作

$$\Pr(\boldsymbol{x})\Pr(\boldsymbol{x} \rightarrow \boldsymbol{x}_p) = \Pr(\boldsymbol{x}_p)\Pr(\boldsymbol{x}_p \rightarrow \boldsymbol{x}) \tag{5-27}$$

211

如果式（5-27）成立，系统将依照玻尔兹曼-吉布斯分布达到平衡。重排序式（5-27）并使用式（5-26），得到如下式子

$$\frac{\Pr(\boldsymbol{x} \rightarrow \boldsymbol{x}_p)}{\Pr(\boldsymbol{x}_p \rightarrow \boldsymbol{x})} = \frac{e^{-\mathscr{E}(\boldsymbol{x}_p)/k_B T}}{e^{-\mathscr{E}(\boldsymbol{x})/k_B T}} = e^{-[\mathscr{E}(\boldsymbol{x}_p) - \mathscr{E}(\boldsymbol{x})]/k_B T} = e^{-\Delta\mathscr{E}/k_B T} \tag{5-28}$$

其中$\Delta\mathscr{E} = \mathscr{E}(\boldsymbol{x}_p) - \mathscr{E}(\boldsymbol{x})$，即，能量的变化。

米特罗波利斯（Metropolis）算法[22]，是一种蒙特卡罗（Monte Carlo）技术，满足式（5-27）的条件，使用下面的变化概率

$$\Pr(\boldsymbol{x} \rightarrow \boldsymbol{x}_p) = \begin{cases} 1 & \text{对于 } \Delta\mathscr{E} < 0 \\ e^{-\Delta\mathscr{E}/k_B T} & \text{对于 } \Delta\mathscr{E} \geqslant 0 \end{cases} \tag{5-29}$$

对于模拟在一个给定温度下的原子集合演化达到平衡，米特罗波利斯算法提出了一个简单有效的方法。在算法的每个后继步骤中，温度慢慢降低，一个原子发生一个小的随机波动（置换），在能量$\mathscr{E}$中引起的变化用$\Delta\mathscr{E}$表示，随后计算。从式（5-29）中，如果能量变化是负的，

即$\Delta \mathscr{E} < 0$（降低能量），那么允许置换，并且置换原子的配置当作初始化条件来开始下一个步骤。如果$\Delta \mathscr{E} \geqslant 0$（从一个较低能量状态变化到一个较高的能量状态），配置是否被接受将依靠来自式（5-29）的概率，也就是

$$\Pr(\Delta \mathscr{E}) \triangleq e^{-\Delta \mathscr{E}/k_B T} \tag{5-30}$$

通过使用区间[0, 1]的均匀分布随机数来实现算法的随机选择部分（参见A.7.4节）。从该分布中选择一个数与$\Pr(\Delta \mathscr{E})$比较，如果该数小于$\Pr(\Delta \mathscr{E})$，新的（混乱）配置保留；如果大于或等于$\Pr(\Delta \mathscr{E})$，当前配置不变并用于下一步。通过多次重复这个基本步骤，模拟温度$T$下与热浴相关的原子热运动。假设温度下降得足够慢，在每个选定的温度下，原子集合都能够达到热平衡。如果式（5-27）中的条件满足，系统将按照玻尔兹曼-吉布斯分布变化并达到平衡。

基于模拟退火的全局搜索算法（源自统计力学的米特罗波利斯算法）有四个基本部分：（1）简洁的系统配置描述，（2）一个目标函数或代价函数（包含必要的权衡），（3）一个探测过程，或一个配置中系统组成单元的“移动”或重新安排的随机发生器，（4）温度退火调度和定义系统演变[14]的时间周期。基本的思想是用大部分时间“下山”算法代替始终下山算法[23]。

基于模拟退火的全局搜索方法对于给定的全局多元代价（目标）函数$f(\boldsymbol{x})$ $(\boldsymbol{x} \in \Phi)$求其全
212 局最优化解，步骤如下：

**基于模拟退火的全局搜索算法**

**步骤1**　初始化向量$\boldsymbol{x}$为一个集合$\Phi$内的随机点。

**步骤2**　为参数$T$选择一个退火（冷却）进度表，初始化$T$为一个足够大的数。

**步骤3**　计算$\boldsymbol{x}_p = \boldsymbol{x} + \Delta \boldsymbol{x}$（其中$\Delta \boldsymbol{x}$是一个系统状态下建议的变化）。

**步骤4**　计算在代价（能量）函数中的改变$\Delta f = f(\boldsymbol{x}_p) - f(\boldsymbol{x})$。

**步骤5**　使用与米特罗波利斯算法相关的式（5-29）来决定是用$\boldsymbol{x}_p$当作系统新的状态还是保持当前状态$\boldsymbol{x}$。对于模拟退火最优化算法等式（5-29）做如下改变

$$\Pr(\boldsymbol{x} \to \boldsymbol{x}_p) = \begin{cases} 1 & \text{对于 } \Delta f < 0 \\ e^{-\Delta f/T} & \text{对于 } \Delta f \geqslant 0 \end{cases} \tag{5-31}$$

其中用$T$代替$K_B T$。在$\Delta f \geqslant 0$的情况下，随机数$\eta$从[0, 1]范围内的均匀分布中选择。如果$\Pr(\boldsymbol{x} \to \boldsymbol{x}_p) > \eta$，混乱状态$\boldsymbol{x}_p$作为新状态（或搜索点）来使用；否则，状态（或搜索点）保持为$\boldsymbol{x}$。

**步骤6**　重复步骤3～步骤5直到系统达到平衡，即，所接受到的变化数量微不足道时，当前搜索的点是（或接近）局部最小值。通常步骤3～步骤5按预先规定的次数执行。

**步骤7**　在步骤2中温度$T$按照退火进度表更新，重复步骤3～步骤6。当温度$T$达到零（系统冻结）或一个规定的很小的数（正数）时，该过程停止。 □

该算法的相关性能依赖于温度参数$T$的进度表选取。如果系统冷却得太快，收敛可能发生得太早，可能导致一个局部最小值作为“解决方案”。另一方面，如果$T$的进度设置得太慢，算法将花费大量的计算时间才能收敛。按照Geman和Geman[24]的观点，如果温度参数如下变化

$$T(k) = \frac{T(0)}{\log(1+k)} \qquad k = 1, 2, \cdots \tag{5-32}$$

其中$k$表明搜索的第$k$次迭代，$T(k)$是第$k$次的温度，$T(0)$是初始温度（一个足够大的正常量），模拟退火算法可以保证收敛（概率为1），当$k \to \infty$时$f(\boldsymbol{x})$达到一个全局最小值。该算法产生一个马尔可夫（Markov）链[25]，按照最小能量配置[24]的均匀分布收敛。式（5-32）给定的温度进度表非常慢，致使它无法使用。对加速模拟退火搜索算法的兴趣与工作一直在持续。例如，Szu[26]的工作。使用一个次优的方法可以加速收敛，但在性能上要付出相应的代价。也

就是说，算法不再保证一定能够收敛到全局最小值（概率为1）。然而，在只需求出接近最优的解时，算法的次优解决方案有许多实际的应用。温度进度表的一个次优方法由[14]给定如下

$$T(k) = \alpha T(k-1) \qquad k = 1, 2, \cdots \tag{5-33}$$

213

其中衰减因子$\alpha$应该足够小并且接近单位值，一般，$0.8 \leqslant \alpha \leqslant 0.99$。等式（5-33）给出温度调度中的指数缩减。为了使这个温度调度算法更加可行，在每个限定的温度[14]变换尝试应在限定的次数内。例如，在每个温度下尝试足够变化，以使系统接收10次状态变化。如果期望接收的数在三个连续温度下仍未达到，那么系统定义为*冻结*和停止退火。

**例5.3** 为了阐明模拟退火作用于最优化问题，考虑著名的“旅行商”问题。该问题按如下方式描述。在一个推销旅程中推销员需要访问给定的城市集合$N$。他想依次访问每个城市一次，并且以一个所需旅行最少的顺序进行（也就是*距离*最短）。最优化任务是决定推销员在旅程中依次访问的城市的最优顺序。模拟退火方法在该问题应用中的第一步是定义所有可能方案的状态空间。在这个例子中，方案是旅程中的一个城市序列表。在该方法中，假定城市的任意顺序是可行的，所以，如果要访问$N$个城市，不同序列的可能数量是$N!$。旅行商问题的候选方案数量与城市数量的阶乘是成比例的。即使一些不能生成最优解的方案能够很容易地被排除，但对于如此多的更优方案来说，问题的规模还是太大。现在假设有一个城市的初始序列。需要指明状态混乱的本质，即，用于在问题配置空间“旅行”的算法。对于该例，通过在当前解决方案中交换两个城市的位置来得到一个新的解决方案（即一个新序列）。最后，需要指明建议方案适当量化的代价函数。这里，代价函数用作表示推销员旅行的总距离。

图5-9a显示了20个需要访问的城市（随机选择）的位置。在模拟退火方法中可以从任意初始序列开始。图5-9b显示出一个这样的序列（随机生成的模拟）。正如显示的一样，该序列包含推销员旅程路程和，并可以明显看出当前的方案远离了最优路径。在算法的每一步中，通过交换随机选择的两个城市的位置得到一个新的序列。通过使用基于模拟退火的全局搜索算法的米特罗波利斯标准，即式（5-31），新提议的序列要么被接受，要么被拒绝。在给定温度下混乱的数量设置为$N(N-1)/2$。温度按照式（5-33）中给定的带有参数$\alpha = 0.95$的指数速度冷却。最终由最优化算法得到的序列显示在图5-9c中。如图所示，该访问城市序列的总旅行距离比图5-9b中的初始条件少得多。最后，图5-9d显示了按照最优化算法状态空间轨迹计算出的代价函数值（即旅行总距离）。

214

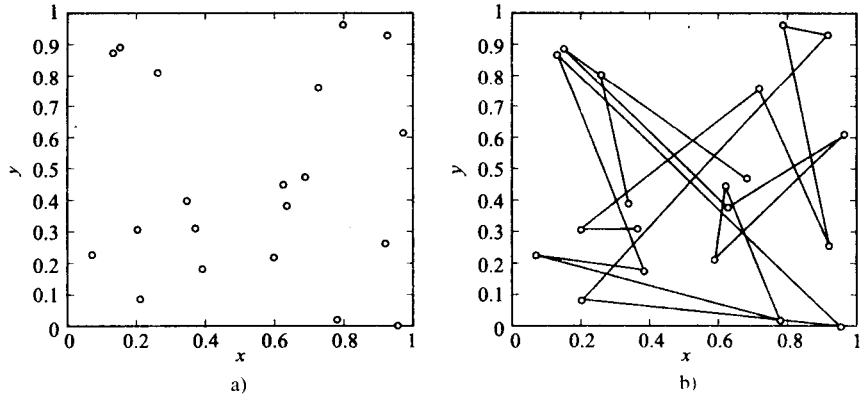
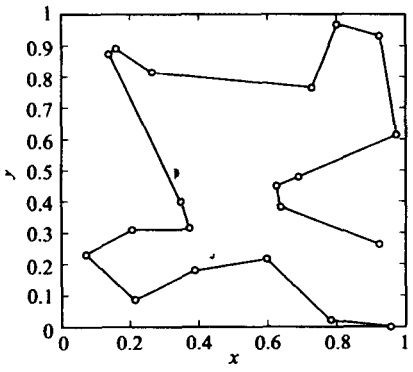
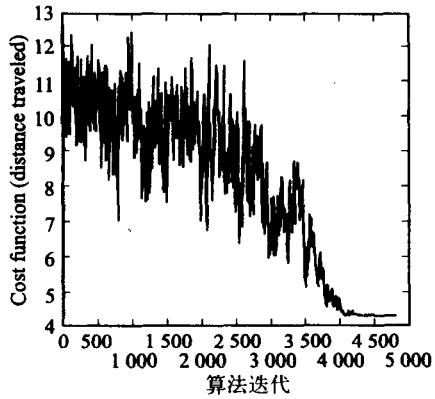



图5-9 使用模拟退火的旅行商问题解决方案。a) 20个城市的位置；b) 初始（条件）解决方案；c) 通过模拟退火得到的解决方案；d) 退火过程中的代价函数轨迹




c)




d)

图5-9 （续）

## 5.5 玻尔兹曼机

玻尔兹曼机是建立在模拟退火（参见5.4节）和使用随机神经元[27-29]基础上，并行的约束满足网络。玻尔兹曼机能够学习集合[28]所示的一组模式的潜在约束特性。由于该网络运用扩展的内部反馈和随机神经元，故认为是随机递归网络。可以把该网络看成是霍普菲尔德网
215 络的延伸（参见5.3节）。这两种网络的主要不同在于玻尔兹曼机可以有隐藏神经元，而霍普菲尔德网络不允许存在隐藏神经元。玻尔兹曼机器和霍普菲尔德网络还有其他两个主要区别：（1）如上所述，玻尔兹曼机器使用依照概率分布“点火”的随机神经元，而霍普菲尔德网络的神经元建立在具有确定性点火机制的McCulloch-Pitts神经元模型上；（2）霍普菲尔德网络采用无监督操作，而玻尔兹曼机器可以在监督模式下训练（也可以在无监督模式下操作）。玻尔兹曼机器和霍普菲尔德网络的共同点是：（1）所有突触权值连接对称；（2）不存在自反馈；（3）处理单元有双极状态；（4）可随机选择神经元，并且一次只更新一个。

玻尔兹曼机器非常适合于涉及大量“弱”（或“软”）限制的约束满足任务[28, 30]。约束满足搜索一般[31]运用满足任何解决方案的“强”约束限制。在涉及博弈和迷宫问题领域时，目标准则通常有强限制特性，因此，强限制是规则（不要混淆合法游戏与好游戏）。然而，对于一些实际问题，准则并不是全有或全无，很多情况下，最好的解决方案并不满足约束限制[32]。因此在这种情况下，最优解通常是实际上尽可能（而不是正好）满足弱限制的“最好”解。下面先给出随机神经元的概述。

随机神经元

我们希望用数学上易处理的方式来解决神经网络中突触噪声的影响。最常用的方法是运用概率方法来命令神经元点火。假定$q$神经元根据概率规则点火，即根据概率$\Pr(v_q)$，神经元的点火由神经元活动水平值 $v_q = \sum_{\substack{j=1 \\ j\neq q}}^{n} w_{qj}x_j$ 决定（参见2.2节）。从而输出神经元$y_q$遵从概率规则

$$y_q = \begin{cases} 1 & \text{概率为}\Pr(v_q) \\ -1 & \text{概率为}1-\Pr(v_q) \end{cases} \tag{5-34}$$

若$v_q = 0$，则$y_q = \pm 1$的概率各为1/2。概率函数$\Pr(v)$的典型选择是

$$\Pr(v) = \frac{1}{1+e^{-2v/T}} \tag{5-35}$$

式（5-35）是S形函数（参见2.3节），$T$是用来控制不确定相关神经元点火的类似温度参数（伪温度）。对于无噪声情况，即温度参数$T$接近0时，式（5-34）中概率规则还原为式（5-2）表示的霍普菲尔德网络的确定性规则。图5-10描绘出了随机神经元点火的概率分布，无噪声极限（$T \to 0$）导致McCulloch-Pitts神经元的二值激活函数。

像霍普菲尔德网络一样，玻尔兹曼机器有对称突触权值结构（即$w_{ij} = w_{ji}$），不允许自反馈（因此，$w_{ij} = 0,\ \forall i = j$）。这类神经元的系统状态概率来源于统计力学的玻尔兹曼-吉布斯分布，因此，这类神经网络称为玻尔兹曼机器。玻尔兹曼机器中的神经元划分为两类：可见和隐藏。对于玻尔兹曼机器中的隐藏神经元存在与前馈网络相似的情况，即在不能从训练模式知道隐藏单元表示什么的情况下确定正确连接到隐藏神经元的问题。图5-11描绘了这两类神经元的区别。如果$n_v$表示可见神经元的个数，$n_h$表示隐藏神经元的个数，则网络连接的总数为$(n_v + n_h)(n_v + n_h - 1)$。在无监督模式下，输入输出神经元没有区别；仅有可见神经元与“外部环境”存在直接的相互作用，并且钳住环境状态。隐藏神经元可以自由操作，并且在环境输入中达到说明基本限制的目的。这种无监督学习过程可用于概率分布建模，此种分布用与具有恰当概率的可见神经元上的同环境相关的夹钳模式说明。如果网络能学习训练恰当的概率分布，则网络能执行所谓的模式完成。在监督模式下，玻尔兹曼机器定义了输入和输出神经元（参看图5-11），并且执行联想功能。训练的监督模式要对每一输入模式提供概率修正响应模式。

216

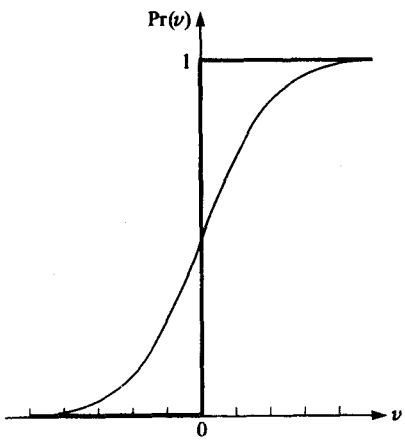



图5-10 随机神经元点火的概率分布函数（S形）和用粗线表示的McCulloch-Pitts神经元激活函数

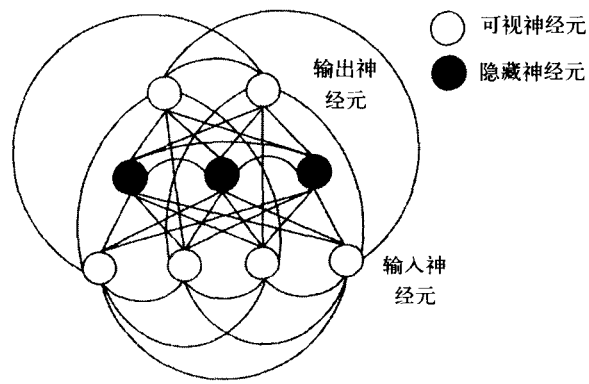



图5-11 表示可见和隐藏神经元的玻尔兹曼机器网络的例子

下式给出全局网络配置的能量

$$\mathscr{E} = -\frac{1}{2}\sum_{\substack{i \\ i \neq j}}\sum_{j} w_{ij}x_i x_j + \sum_i \theta_i x_i \tag{5-36}$$

其中$x_i$相当于第$i$个神经元输出（状态），$\theta_i$是第$i$个神经元阈值，$w_{ij}$是神经元$i$和$j$的连接突触权值。如果神经元$i$是激活状态，则状态$x_i$为+1，反之若神经元$i$不是激活状态，则状态$x_i$为$-1$。式（5-36）的最小值是网络的稳定状态。能量函数式（5-36）的向量矩阵形式为

217

$$\mathscr{E} = -\frac{1}{2}\boldsymbol{x}^T\boldsymbol{W}\boldsymbol{x} + \boldsymbol{x}^T\theta \tag{5-37}$$

其中对称权值矩阵$W$具有0对角元素。

由于玻尔兹曼机器具有隐藏单元，因此，玻尔兹曼机比霍普菲尔德网络需要更新更多神经元，即需要更新与隐藏神经元相关的权值。尽管这样，更新过程相当简单直接。一般而言，在玻尔兹曼机器中，交替存在着与学习周期相关的两个阶段：正向阶段和负向阶段。然后是突触权值调整。特别地，神经元$i$是随机选择的，通过从正向状态$x_i$到负向状态$-x_i$的状态转换函数表示输出。状态转换函数为

$$\Pr(x_i \to -x_i) = \frac{1}{1+e^{-\Delta_i \mathscr{E}/T}} \tag{5-38}$$

其中$\Delta\mathscr{E}_i$是神经元$i$相关转换的能量变化，$T$是伪温度。当T→0时，等式（5-38）逼近应用在霍普菲尔德网络中的且与确定的McCulloch-Pitts神经元模型相关的“阶梯函数”。因此，霍普菲尔德网络是当玻尔兹曼机器的温度为0、没有隐藏神经元的特殊情况。将式（5-38）与
218 式（5-35）中的概率函数$\Pr(v)$比较，可发现它们形式不同，但表示是相同的。为了说明这一点，观察式（5-36）中的能量函数的变化（假定网络不包含外界偏置，因此$\theta_i = 0 \forall i$）。再假定神经元$i$的状态改变由正到负，即$x_i \to -x_i$，能量函数的变化表示为

$$\Delta\mathscr{E} = \underbrace{-\Delta x_i}_{-[x_i-(-x_i)]} \underbrace{\sum_{j=1}^{n} w_{ij}x_j}_{v_i} = -2x_i v_i \tag{5-39}$$

其中$v_i$是神经元$i$的活动水平。将式（5-39）中的能量变化$\Delta\mathscr{E}$代入式（5-38）中，得

$$\Pr(x_i \to -x_i) = \frac{1}{1+e^{2x_i v_i/T}} \tag{5-40}$$

因此，当神经元$i$的初始状态$x_i = -1$时，由式（5-40）给出的神经元将转换或扭曲到相反状态（即$x_i \to 1$）的概率为

$$\Pr(x_i \to 1) = \frac{1}{1+e^{-2v_i/T}} \tag{5-41}$$

式（5-41）与式（5-35）一致。若神经元$i$的初始状态是$x_i = 1$，则神经元将反转到相反状态（$x_i \to -1$）的概率为

$$\Pr(x_i \to -1) = \frac{1}{1+e^{2v_i/T}} \tag{5-42}$$

显然，式（5-42）可改写为

$$\Pr(x_i \to -1) = 1 - \frac{1}{1+e^{-2v_i/T}} \tag{5-43}$$

式（5-41）与式（5-43）一起正好是式（5-34）描述的关于一般随机神经元的概率规则。在玻尔兹曼机器中总共有$n = n_v + n_h$个（可见和隐藏）神经元。若每一神经元呈现双极状态$x_i = \pm 1$的任一种，网络全局状态的总数为$2^n$。

在玻尔兹曼机器中，运用模拟退火搜寻能量地图，以确定能量函数的全局最小值。从而，网络根据米特罗波利斯动力学演变。在霍普菲尔德网络中，由于网络的局部最小能量点用来储存信息，因此需要定位这些局部最小点。对于玻尔兹曼机器，给定当前输入，执行约束满足任务时，希望“避开”局部最小而搜索全局最小。因此，在玻尔兹曼学习过程中，首先在高温下运行网络，然后，慢慢降低温度直到网络达到热平衡。在这个过程中网络实现能量最小化。降温过程遵循退火进度（即“温度”参数$T$的冷却计划）完成（参见5.4节）。在高温时，

完成全局状态空间的粗略搜索，这一过程中微弱的能量变化被忽略，网络迅速接近热平衡，并且达到较好的极小值。降低温度使网络完成能量分布的精确搜索，现在需要网络对微弱的能量变化做出响应。在精细调整过程中，网络在能量的粗略搜索到的极小值附近找出更好的极小值。因此，当退火过程结束时，网络将定位在与一系列满足弱限制的且最接近可行解决方案的极小值处（不是最好，而是尽可能好）。 219

玻尔兹曼机器学习规则不是起源于此，但在[4, 27]中按部就班地按照算法中的步骤得以实现。可见神经元的状态用$\alpha$标示，隐藏神经元用$\beta$标示。之前，假定可见神经元的个数为$n_v$，隐藏神经元的个数为$n_h$，因此，$\alpha$取值从1到$2^{n_v}$、$\beta$取值从1到$2^{n_h}$。用$\alpha$和$\beta$定义的网络全局状态共有$2^{n_v+n_h}$种可能状态。找到全局状态中的每一种的概率$\Pr_{\alpha\beta}$由Boltzmann-Gibbs分布给出（参见5.4节）。$\Pr_\alpha$是在状态$\alpha$独立于$\beta$情况下找到可见神经元的概率。概率$\Pr_{\alpha\beta}$可在给出可见状态$\alpha$条件下，根据隐藏神经元$\beta$的条件概率$\Pr_{\beta|\alpha}$（参照A.7.1节）给出

$$\Pr_{\alpha\beta} = \Pr_{\beta|\alpha}\Pr_\alpha \tag{5-44}$$

在“自由运行系统”中，$\Pr_\alpha$实际上是在状态$\alpha$时找到可视单元的概率，由网络的突触权值$w_{ij}$决定。这些状态的期望概率由集合$R_\alpha$表示

在操作上，学习算法由四层嵌套循环[4]实现：

**玻尔兹曼机的学习算法**

**循环1** 在最外层循环中，网络的突触权值根据下式更新多次，以确保收敛

$$\Delta w_{ij} = \mu\beta\left[\overline{\langle x_i x_j\rangle}_{\text{clamped}} - \langle x_i x_j\rangle_{\text{free}}\right] \tag{5-45}$$

其中$\mu > 0$，并且

$$\overline{\langle x_i x_j\rangle}_{\text{clamped}} = \sum_\alpha\sum_\beta R_\alpha P_{\beta|\alpha} x_{i|\alpha\beta} x_{j|\alpha\beta} \qquad i,j = 1,2,\cdots,n; i \neq j \tag{5-46}$$

它是神经元$j$和$i$的状态之间的关系，以固定到环境的可见神经元为条件（所有的可能状态取平均）。所以$\overline{\langle x_i x_j\rangle}_{\text{clamped}}$，是$\langle x_i x_j\rangle$当可见单元在状态$\alpha$中钳制时的值，按照期望概率$R_\alpha$计算$\alpha$的平均值。在式（5-46）中，$x_{i|\alpha\beta}$表示神经元$i$的状态，其中可见神经元在状态$\alpha$中和隐藏神经元在状态$\beta$中。$\alpha$的范围是$[1, 2^{n_v}]$（对于可见神经元），$\beta$的范围是$[1, 2^{n_h}]$（对于隐藏神经元），$n = n_v + n_h$是网络中神经元的总数量。式（5-45）的第一项本质上是带有可见的钳制单元的Hebb项。式（5-45）中第二项符合带有自由运行的Hebb不学习。当自由系统单元/单元相关性$\langle x_i x_j\rangle$等于钳制的单元时，该过程收敛。 220

**循环2** 对于循环1中的每次迭代必须在松开状态下，与钳制在期望模式中的可视单元计算。为了运行玻尔兹曼机器，对于某个正温度$T > 0$来说，系统必须处于热平衡状态。系统状态$\boldsymbol{x}$波动，通过计算$x_i x_j$的时间平均度量相关性$\langle x_i x_j\rangle$。为了得到对于计算式（5-45）的突触权值更新规则的所有必要信息，这个过程必须依次执行，先用含有钳制的每个状态$\alpha$下可视神经元，再将神经元松开一次。在每种情形中，在得到一个平均值之前系统必须重复上述步骤直至达到热平衡。

**循环3** 对于循环2中的每个平均值，对一个足够大的初始温度$T(0)$，使温度逐渐地降低，必须使用一个模拟退火温度进度表$\{T(k)\}$来实现热平衡。

**循环4** 在循环3的每个温度，许多神经元必须抽样出来，按照式（5-34）的规则更新

$$x_i = \begin{cases} 1 & \text{概率是 } \Pr(v_i) \\ -1 & \text{概率是 } 1-\Pr(v_i) \end{cases} \tag{5-47}$$

其中

$$\Pr(v_i) = \frac{1}{1+e^{-2v_i/T}} \tag{5-48}$$

$v_i$是神经元$i$的活动水平，也就是，

$$v_i = \sum_{\substack{j=1 \\ j\neq i}}^{n} w_{ij}x_j \tag{5-49}$$

□

运用这个学习策略更新玻尔兹曼机器的突触权值[参见式（5-45）]需要两平均值之间有差异，且两平均值都能波动。因此，处理一个平衡性较差的系统或运用较短平均时间能缩小更新周期，但是将发生较差的权值更新$\{\Delta w_{ij}\}$，并且最终需要更多的更新周期。虽然玻尔兹曼机器非常慢，但是在解决复杂问题方面非常有效。由于玻尔兹曼机器计算量大，并且存在模拟退火过程，因此已经研究了其他不同的学习算法。例如，运用统计力学的均场近似值[33]，通过用确定性的、模拟神经元代替玻尔兹曼机器的随机双极状态神经元来减少过多的计算时间[34]。

## 5.6 时间前馈网络概述

在神经网络中（连续或离散）时间元素是学习过程中非常重要的因素。许多认知函数（如语言、视觉、电机控制）的正确操作依赖于时间。当时间元素包括在神经网络中时，网络可以执行那些没有时间就不可行的任务。如，网络能跟踪与非静态过程相关的输入数据的统计变化。下一节将讨论有关时间过程的神经网络数量选择。在每种情况，考虑静态神经网络结构，并且引入网络的时间属性是为了将动态属性并入结构中。特别地，将短期记忆并入网
221 络是为了时间延迟。时间延迟允许网络变为动态网络。用空间依赖性代替与网络输入有关的时间依赖性。因此，我们的讨论实际上涉及静态网络的时空敏感性。我们仅就所关注的网络相关的时间依赖性发表意见。

下面是时间网络的最常见类型的简明概括。

1. 时间延迟神经网络（TDNN）是每一输出层神经元都有时间延迟的前馈多层神经网络。TDNN已经应用在语言识别问题中[35-37]。

2. 有限冲击响应（FIR）前馈多层网络是TDNN的推广[38-40]。它有FIR数字过滤器代替TDNN结构中的每一权值。这是分布式时滞前馈神经网络（DTLFNN）的基础。

3. 简单递归网络（SRN）或Elman网络[41]是单隐藏层前馈网络，它具有从隐藏层神经元输出到网络输入的反馈连接。

4. 实时递归神经网络（RTRNN）有两层，能实时学习[42]。除霍普菲尔德网络没有隐藏神经元外，其余同霍普菲尔德网络相似。

5. 管状递归神经网络（PRNN）[43]是模块结构，其中每一模块接收网络输入的不同延迟给予。每一模块是同一个输出神经元完全连接的递归网络。这类网络除了两个前馈连接之外还有反馈（即递归）连接。这类网络是为了自适应预测非静态信号（如语音信号）而研究的。

6. 非线性自回归滑动平均（NARMA）神经网络用于非线性系统的控制和辨识（参见10.6.2节）[44-46]。

## 5.7 简单递归网络

简单递归网络（通常称为Elman网络）[41]是单隐藏层前馈神经网络。它具有从隐藏层神经元的输出到网络输入的反馈连接。该网络结构同Jordan[47]提出的结构相似。最初研究SRN是为了学习时变模式或时间序列，尤其是字符串。基本的SRN结构如图5-12所示。图中网络的上部分包含上下文单元。这些单元的作用是在前时间步复制隐藏层输出信号。

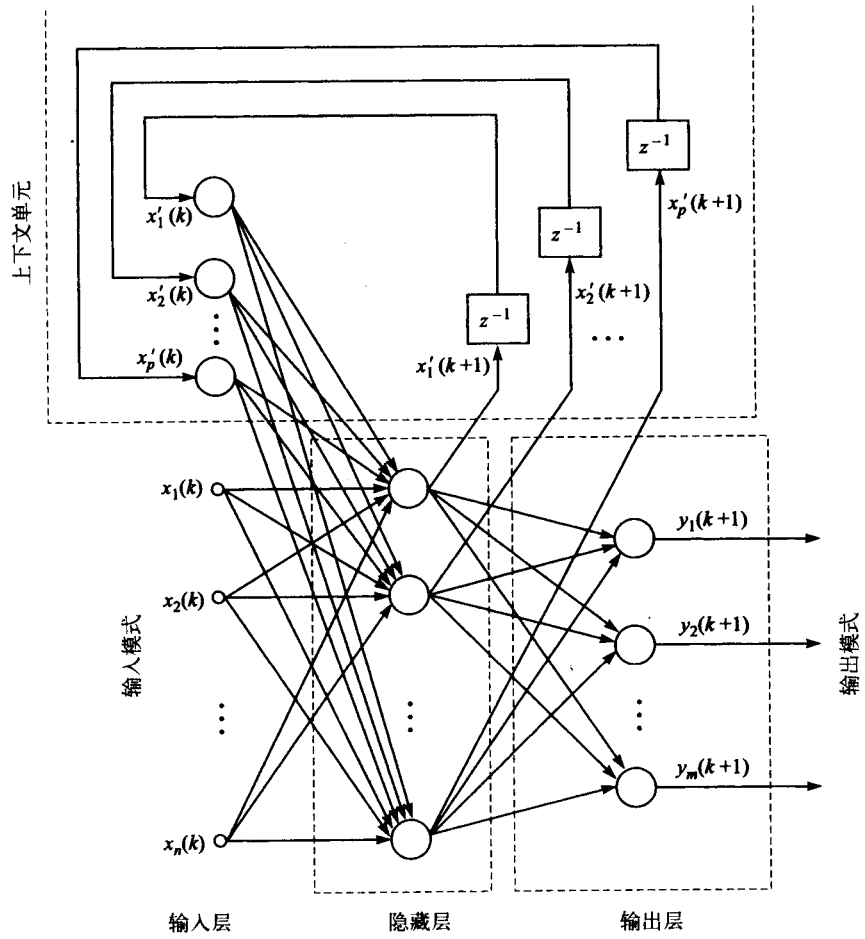



图5-12 SRN或Elman神经结构（其中$z^{-1}$是单位时间延迟）

这些上下文单元的目的是什么？实际上回答很简单。上下文单元的目的是处理输入模式不一致。换言之，可能发生的模式冲突导致从单一输入模式产生多种输出。从而导致标准反向传播网络的复杂情形。在SRN中，网络的输入（即图5-12中的$\boldsymbol{x}(k)$）是前时间步隐藏层输出的扩大，即$\boldsymbol{x}'(k)$。因此，SRN中提供的反馈或$\boldsymbol{x}'(k)$基本是对当前输入$\boldsymbol{x}(k)$建立上下文。这样可以在网络内部提供一种机制来区别发生在不同时间，但本质上相同的模式。上下文单元的权值固定。但是，其他网络权值可以在监督训练模式下使用具有动量的误差反向传播算法调节（参考3.3.3节）。

**例5.4** 设计一个SRN来检测次声信号的最高振幅（参照例10.8）。图5-13显示两个模拟次声信号（正弦波）。它可能是两个火山爆发的稳态次声信号。两个信号均有一个50mHz基本频

222
~
223

率，第一个信号有2.5$\mu$巴（声压单位）最高振幅，第二个信号有1$\mu$巴（声压单位）最高振幅。样本频率假定是$f_s$ = 1Hz。两个模拟次声事件的样本连接形成长204的输入向量。特别地，来自每个信号的输入样本改变（每个复制两次）。也就是说，在MATLAB中设计成包含输入抽样的向量是INPUTS，定义为

```
INPUTS=[signal1 signal2 signal1 signal2];
```

其中INPUTS $\in \Re^{1\times 204}$，signal1包含来自图5-13的第一个信号抽样，signal2包含图5-13中的第二个信号抽样。与每个信号相关的目标值反映各自的最高振幅。所以，对于signal1的每个抽样（总共51个），相关目标值是2.5（这是使用在MATLAB中的向量T1），对于signal2，相关的51个值总是1（在MATLAB中的向量T2）。这些也连接来对应地协调向量INPUTS中的值。也就是，在MATLAB中向量TARGETS包含如下目标值：

```
T1=2.5*ones (1,51) ;
T2=ones (1,51) ;
TARGETS=[T1  T2  T1  T2];
```

其中TARGETS $\in \Re^{1\times 204}$。

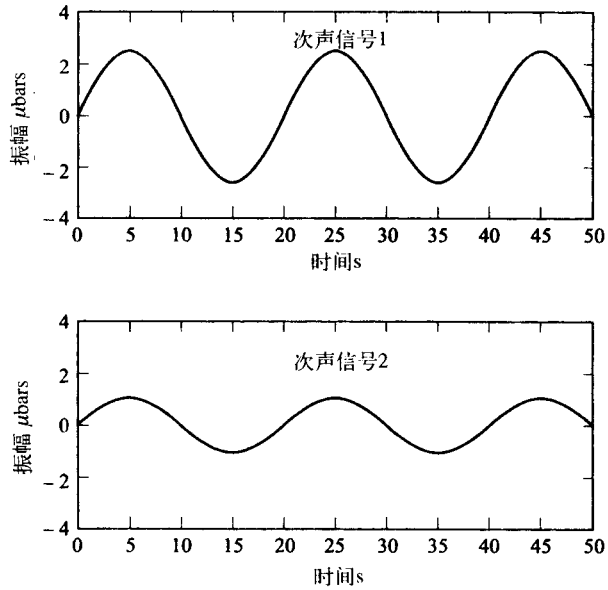



图5-13　两个模拟次声信号，均有50mHz的频率

MATLAB神经网络工具箱[10]用来执行模拟。主要的目标是SRN（Elman网络）作为最高振幅探测器来执行，即，使用训练数据{INPUTS, TARGETS}学习两个最高振幅之间的区别。
224 在MATLAB中，SRN（作为Elman网络被提及）在隐藏层神经元中有tansig激活函数（2.3节讨论的双曲线正切函数$f_{hts}$），在输出神经元中有purelin（线性）激活函数。对于这个问题，Elman神经网络有单一输入，单一输出神经元，带有15个隐藏（递归）神经元。网络顺序处理输入。也就是，在向量INPUTS和TARGETS中的“时间”抽样是顺序处理的。

在MATLAB中Elman网络的初始化使用函数initelm来执行。该函数接受作为输入：(1) 输入数据（INPUTS），(2) 隐藏（递归）tansig神经元的数量，(3) 目标输出（TARGETS）。对于第三个变量，可以选择给定输出purelin神经元数量代替目标输出到初始化网络。initelm

函数的输出是两个初始权值矩阵和用于网络的两层的偏置。MATLAB函数trainelm用来训练Elman网络。对于多层前馈网络该函数使用具有动量反向传播学习规则（参见3.3.3节）。函数trainelm接收为初始权值矩阵的输入和来自initelm函数的偏置，该函数带有输入、目标值和一个可选择训练参数向量TP。在训练以后，MATLAB函数trainelm的输出是最终网络权值矩阵和偏置。下面是MATLAB命令，用来初始化和训练网络：

```
[W1,B1,W2,B2]=initelm (INPUTS,15,TARGETS) ;
TP=[10 5000 2 0.001 1.05 0.7 0.95 1.04];
[W1,B1,W2,B2]=trainelm (W1,B1,W2,B2,INPUTS,TARGETS,TP) ;
```

在训练参数向量TP中，所有显示的值是默认值，除非第一个值10（两个显示之间的回合数）、第二个值5000（训练的最大回合数）、第三个值2（总平方和误差目标）。

在2609个训练回合后，满足误差目标，如图5-14显示。将该网络内实际目标值和由网络产生的估计一起绘出，图5-15显示出训练Elman网络后的结果。用INPUTS中输入数据完成训练后，MATLAB函数simuelm产生网络输出。也就是 225

```
OUTPUTS=simuelm (INPUTS,W1,B1,W2,B2) ;
```

正如从图5-15中能看到的，Elman网络在检测两个信号的正确最高振幅方面具有相当好的性能。

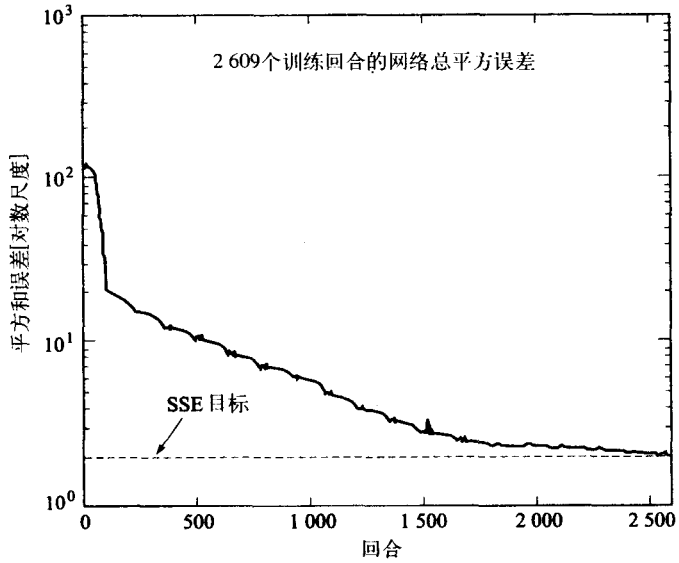



图5-14 训练期间的网络总平方误差

对于任何神经网络来说，其重要属性是推广能力。假设提交两个相同频率但不同最高振幅的次声信号到训练网络中。这是两个该神经网络以前没见过的信号。第一个信号有0.75的最高振幅，第二个信号的最高振幅是1.75。再使用MATLAB函数simuelm，网络输出能够使用“新”测试输入数据来产生。这些结论在图5-16中给出。从图中显而易见网络没有推广。为了提高网络的推广能力，需要更多的训练输入来更进一步训练网络。也就是，输入更多的有不同最高振幅的训练信号。

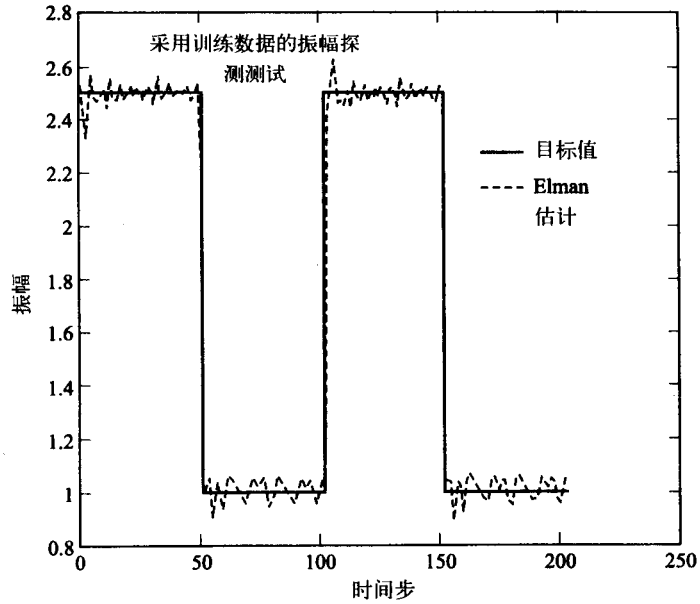



图5-15　对于例5.4使用训练数据的最高振幅探测结果

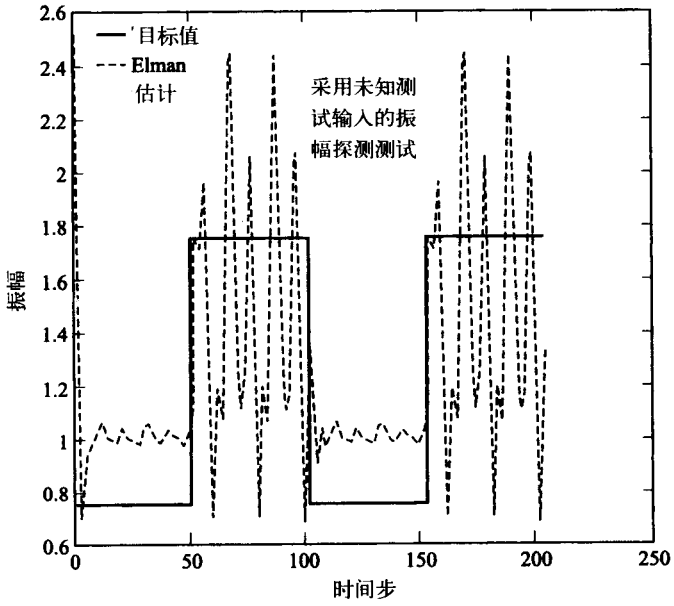



图5-16　对于例5.4使用测试数据的最高振幅探测结果

## 5.8　时延神经网络

时延神经网络运用时间延迟来执行时间处理。它实际上是网络输入按时间相继延迟的前馈神经网络。图5-17显示了对输入向量的每一元素具有多重延迟的单一神经元。这是前馈TDNN（不失一般性，忽略模型中的可能的偏置项）的神经元“构建模块”。当输入向量$\boldsymbol{x}(k)$按时间发展时（$k$是离散时间指数），在神经元中计算出过去的$p$个值。建立的输入时间序列表达式为

$$X = \{x(0), x(1), \cdots, x(m)\} \tag{5-50}$$

因此，矩阵$X$由传送给网络的时间灵敏性输入（列）向量序列$x(k)(k = 0, 1, \cdots, m)$组成。在神经元结构中输入的过去值利用图5-17所示的时间延迟确定($p < m$)。单一神经元需要的权值总数为$(p + 1)n$。

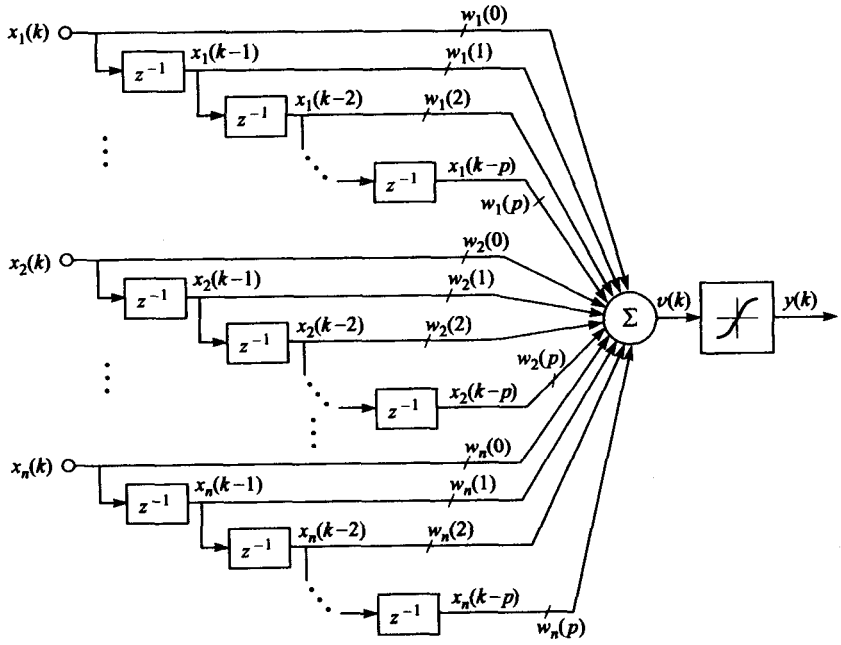



图5-17 具有$n$输入和每一输入$p$个延迟的基本TDNN神经元（$k$是离散时间指数）

单一神经元模型可扩展为多层结构。如前讲述，TDNN结构合并具有时间延迟的前馈多层网络。TDNN的典型结构是仅在网络输入层延迟的分层结构。在层间合并延迟是可能的。TDNN可使用改进的标准反向传播算法来训练（参见3.3.1节）。基本上，若假定训练集由无数与目标（输出）数据相一致的移位输入构成，则在训练期间，网络能够学习输入模式的显著特征。每一隐藏单元不仅观察输入而且是输入的$p$个延迟。图5-18显示了用于语音识别，尤其是音素识别[36]的三层TDNN的例子。该网络是TDNN在网络输入层和隐藏层运用时间延迟的例子。在图5-18中，输入是由信号的功能谱（参照10.10.2节）计算得到的16个melscale系数。

## 5.9 分布式时滞前馈神经网络

分布式时滞前馈神经网络（DTLFNN）从某种意义上来说是时间元素的分布贯穿整个网络。DTLFNN的基本构成块是图5-19所示的简单非线性神经元过滤器。有意思的是，实际上这与图5-17描述的TDNN神经元结构是一样的。对于图5-17的多个输入的神经元，神经元的每一输入$x_i(k)(k = 1, 2, \cdots, n)$用有限脉冲响应过滤器过滤（由于我们考虑的是离散时间过程）[48]。特别地，参看图5-17，线性组合器的输出可表示为

226 ~ 228

$$v(k) = v_1(k) + v_2(k) + \cdots + v_n(k) = \sum_{i=1}^{n} v_i(k) \tag{5-51}$$

其中

$$v_i(k) = w_i(0)x_i(k) + w_i(1)x_i(k-1) + w_i(2)x_i(k-2) + \cdots + w_i(p)x_i(k-p)$$
$$= \sum_{r=0}^{p} w_i(r)x_i(k-r) \tag{5-52}$$

其中$i = 1, 2, \cdots, n$，式（5-52）中是卷积和，在域$z$中，可由式（5-52）得出

$$V_i(z) = w_i(0)X_i(z) + w_i(1)z^{-1}X_i(z) + w_i(2)z^{-2}X_i(z) + \cdots + w_i(p)z^{-p}X_i(z) \tag{5-53}$$

或作为传递函数有

$$H_i(z) = \frac{V_i(z)}{X_i(z)} = w_i(0) + w_i(1)z^{-1} + w_i(2)z^{-2} + \cdots + w_i(p)z^{-p} \tag{5-54}$$

或

$$H_i(z) = \frac{V_i(z)}{X_i(z)} = \frac{w_i(0)z^p + w_i(1)z^{p-1} + w_i(2)z^{p-2} + \cdots + w_i(p)}{z^p} \tag{5-55}$$

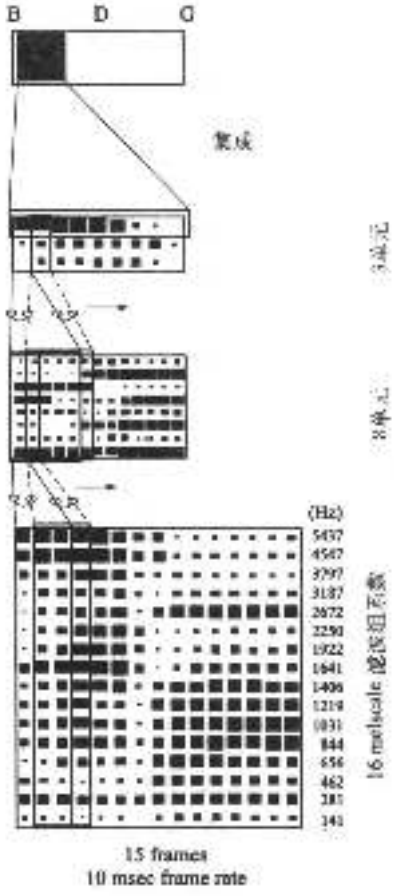



图5-18 Waibel et al.[36]使用的用于音素识别的（三）层TDNN结构

在式（5-54）或式（5-55）中，$H_i(z)$是FIR过滤器的传递函数。因此，图5-19所示的每一FIR过滤器块是由式（5-54）或式（5-55）给出的特定FIR过滤器的传递函数$H_i(z)$，$H_i(z)$中的权值实际上是分子多项式的系数，且规定传递函数的零点存在于复的$z$平面。图5-19中对于网络第$q$神经元的线性组合器的输出为

$$v_q(k) = s_{q1}(k) + s_{q2}(k) + \cdots + s_{qn}(k) = \sum_{i=1}^{n} s_{qi}(k) \tag{5-56}$$

对于第$q$神经元，比较式（5-56）和式（5-51），可发现某些单个过滤的输入为$s_{qi}(k) = v_{qi}(k)$。尤其，图5-19中的时间域中，每一过滤的输入由卷积和表示为

$$s_{ji}(k) = \sum_{r=0}^{p} w_{ji}(r) x_i(k-r) \tag{5-57}$$

$i = 1, 2, \cdots, n$，$j = 1, 2, \cdots, q$，$r = 0, 1, \cdots, p$，其中$p$是延迟的总数。因此，网络中第$j$神经元的输出表示为

$$y_j(k) = f[v_j(k)] = f\left[\sum_{i=1}^{n} s_{ji}(k)\right] = f\left[\sum_{i=1}^{n}\sum_{r=0}^{p} w_{ji}(r) x_i(k-r)\right] \tag{5-58}$$

DTLFNN由图5-19所示形式的神经元层构成，神经元输出形式由式（5-58）给出。

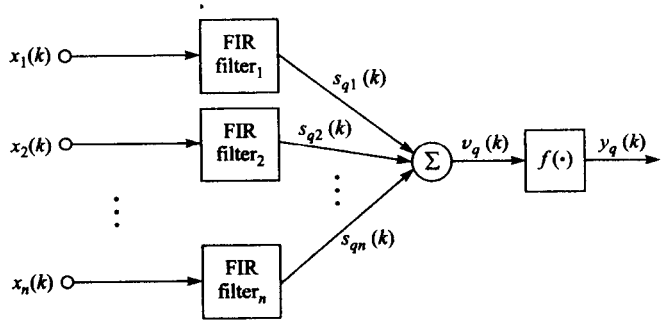



图5-19 非线性神经元过滤器

229 ~ 230

DTLFNN用监督学习算法训练，特别是时间反向传播算法[38]。这一训练算法是标准反向传播训练算法的时间推广。从[38]中可知，推广的时间反向传播算法可总结如下：

根据下式更新适当的网络权值（列）向量

$$\boldsymbol{w}_{ji}^{(s)}(k+1) = \boldsymbol{w}_{ji}^{(s)}(k) - \mu^{(s)} \delta_j^{(s+1)}(k) \boldsymbol{x}_i^{(s)}(k) \tag{5-59}$$

其中

$$\delta_j^{(s)}(k) = \begin{cases} -e_j(k) f'[v_j(k)] & \text{对于输出层的神经元} j \\ f'[v_j(k)] \sum_{h=1}^{q_{s+1}} \Delta_h^{s+1}(k) \boldsymbol{w}_{hj}^{s+1} & \text{对} s \text{隐藏层的神经元} j \end{cases} \tag{5-60}$$

式（5-60）中，$e_j(k)$是即时误差，且

$$\Delta_h^s(k) = [\delta_h^s(k) \quad \delta_h^s(k+1) \quad \cdots \quad \delta_h^s(k+p)]$$

## 习题

5.1 考虑由两个带有零阈值的神经元组成的简单霍普菲尔德神经网络。网络突触权值矩阵如下

$$\boldsymbol{W} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

网络存在四种可能的状态，这四种状态给定如下

$$x_1=\begin{bmatrix}1\\1\end{bmatrix}\quad x_2=\begin{bmatrix}-1\\1\end{bmatrix}\quad x_3=\begin{bmatrix}-1\\-1\end{bmatrix}\quad x_4=\begin{bmatrix}1\\-1\end{bmatrix}$$

这些显示在图5-20中，其中$x_2$与$x_4$（黑“点”）是稳定平衡状态，而$x_1$与$x_3$不是（白“点”）。

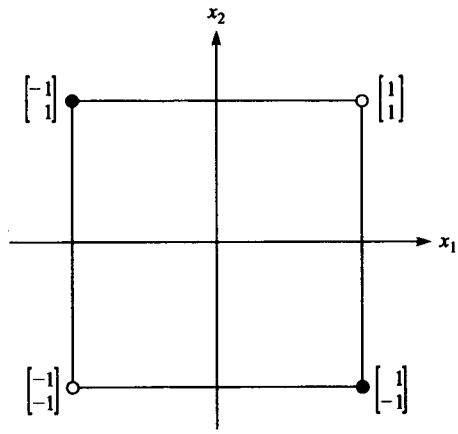



图5-20　两个神经元霍普菲尔德网络的状态

(a) 为了表示这种情形，提交四个上面的输入到霍普菲尔德网络并且观察输出。也就是使用式（5-10）中异步更新表达式。你能够推断出什么？

(b) 使用式（5-14）中的能量函数，能够得出与部分（a）中同样的结论吗？

5.2　假定下面5个双极原型记忆：

$$\phi_1=[-1,-1,1,1,1,1,1,1,1,1,-1,-1,1,1,-1,1,1]^T$$
$$\phi_2=[-1,1,1,-1,-1,-1,1,-1,1,1,1,1,-1,1,1,1]^T$$
$$\phi_3=[1,1,1,-1,1,1,1,1,1,1,1,-1,-1,-1,-1,-1]^T$$
$$\phi_4=[-1,1,1,-1,-1,1,1,1,-1,-1,-1,1,1,-1,1,-1]^T$$
$$\phi_5=[1,-1,-1,1,1,1,1,1,1,1,1,-1,-1,-1,-1,1]^T$$

(a) 用这5个原型记忆，建立一个16个神经元的霍普菲尔德内容可寻址存储器。

231 (b) 使用原型记忆$\{\phi_1,\phi_2,\phi_3,\phi_4,\phi_5\}$中的每个作为到部分（a）中设计的霍普菲尔德网络的输入。也就是说，$x_1=\phi_1$，$x_2=\phi_2$，…，显示出能够在某一时间步内使用式（5-10）的异步更新表达式来回忆原型向量。

(c) 图5-21给出了根据用户提供的位误差率（BER）值，MATLAB函数产生一个输入向量噪声。在图5-21中的MATLAB函数noise在输出中产生不仅仅是输入向量的破坏副本，也在初始输入向量中作索引，该向量内的元素已经被“拴牢”。使用每个输入，即$x_1$，$x_2$，…，通过实验，决定BER的一个合理值以便损坏的原型：（1）总能够恢复；（2）能够在95%的时间内恢复；（3）能够在90%的时间内恢复。

5.3　问题5.2的一个变化。除“句点”被图5-22中的字母M代替外，图5-5中同样的5个字符作为发展霍普菲尔德网络的原型来使用。

(a) 用字母M替代句点建立144个神经元的霍普菲尔德神经网络来重复执行习题5.2的步骤。显示突触权值矩阵作为与图5-6中显示图像相似的灰度图。

(b) 证明每个字符在某时间步内能够使用式（5-10）的异步更新表达式来逐步回忆，该表

达式具有部分（a）开发的突触权值矩阵。每个到网络的输入将是字符图像的向量化形式；而作为“被改造的”字符图像用来显示网络输出。使用MATLAB函数reshape来实现：

```
function [XC,I] = noise (X,BER)
% [XC,I] = NOISE(X,BER) generates a noisy
% version of the input vector X, i.e., XC.
% INPUTS:
% X:    Uncorrupted input vector, can only have
%       bipolar values, i.e., [-1,1].
% BER:  bit-error-rate given in percentage,
%       for example, BER=20 means that on the
%       average 20% of the elements in the
%       vector will be ''toggled'' (or the polarity
%       changed) .
% OUTPUTS:
% XC:   Corrupted version of the input vector X
% I:    Indices in the original input vector X
%       where the elements were ''toggled.''
%

R=rand(size(X));
XC=X;
T=1-BER/100;
  for i=1:length(X)
       if R(i)>=T
          if X(i)==1
            XC(i)=-1;
          else
            XC(i)=+1;
          end
       end
  end
I=find([X-XC]);
```

图5-21 产生带有定义位误差率输入向量的噪声的MATLAB函数noise

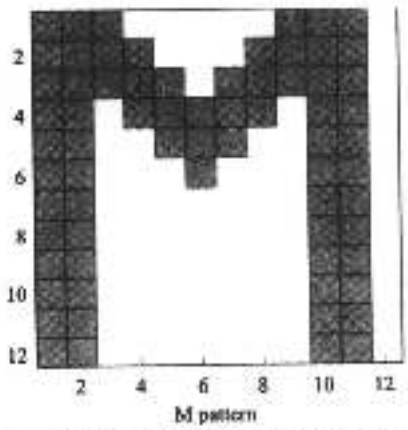



图5-22 替代问题5.2中图5-5的“句点”字符的字母M

(c) 在每个字符图像中引入了30%的错误。也就是，在字符图像中30%的像素有一个极性改变，也就是说，由黑色像素变为白色像素，或相反。使用图5-23中的MATLAB函数noise2来“破坏”每个图像。注意：MATLAB函数类似于问题3.9（图3-18）中的errors函数和问题5.2（图5-21）中的noise函数，事实上，函数noise2是问题5.2中noise的二维表达形式。使用式（5-10）中的异步更新表达式，使用noise2产生被破坏的输入，需要多少次迭代来回忆存储在霍普菲尔德网络记忆中

232
↓
233

的正确字符？

```
function [XC,I,J] = noise2(X,BER)
% [XC,I,J] = NOISE2(X,BER) generates a noisy
% version of the input matrix X, i.e., XC.
% INPUTS:
% X:     Uncorrupted input matrix, can only have
%        bipolar values, i.e., [-1,1].
% BER:   bit-error-rate given in percentage,
%        for example, BER=20 means that on the
%        average 20% of the elements in the
%        matrix will be ''toggled'' (or the polarity
%        changed).
% OUTPUTS:
% XC:    Corrupted version of the input matrix X
% I:     vector of row indices in the original input
%        matrix X where the elements were ''toggled.''
% J:     vector of column indices in the original input
%        matrix X where the elements were ''toggled.''

[nr,nc]=size(X);
N=rand(size(X));
XC=X;
T=1-BER/100;
      for i=1:nr
         for j=1:nc
            if N(i,j)>=T
              if X(i,j)==1
                XC(i,j)=-1;
              else
                XC(i,j)=+1;
              end
            end
         end
      end
[I,J]=find([X-XC]);
```

图5-23　产生带有已定义的位误差率输入向量的噪声的MATLAB函数noise2

5.4　考虑由存储如下两个原型记忆的5个神经元组成的霍普菲尔德网络：

$$\boldsymbol{\phi}_1 = [1, 1, -1, 1, 1]^T \qquad \boldsymbol{\phi}_2 = [1, -1, 1, -1, 1]^T$$

(a) 使用$\boldsymbol{\phi}_1$和$\boldsymbol{\phi}_2$建立霍普菲尔德网络，并且观察突触权值矩阵。

(b) 使用式（5-10）的异步更新表达式，证明两个原型记忆能够经过一次迭代的记忆来恢复。

(c) 考虑两个向量：

$$\boldsymbol{x}_3 = [1, 1, -1, -1, -1]^T \qquad \boldsymbol{x}_4 = [-1, 1, -1, 1, -1]^T$$

234　其中没有用来建立霍普菲尔德网络，仍存储在一个内容可寻址存储器中的向量模式可以当作一个网络的基本记忆。两个向量中任一个（或全部）能够隐含地存储在由部分（a）开发的霍普菲尔德记忆吗？给出详细分析来证明结论。

M　5.5　考虑一个如下定义的最优化问题：$-40 \leqslant x_1 \leqslant 40$和$-40 \leqslant x_2 \leqslant 40$，$x_1$和$x_2$，求下面函数的最小值，函数由$-40 \leqslant x_1 \leqslant 40$和$-40 \leqslant x_2 \leqslant 40$定义在矩形内，且$x_1$, $x_2$都为整数：

$$f(x_1, x_2) = (x_1 - 30)^2 + (x_2 - 20)^2 + 40\sin^2(x_1 x_2) + \exp\left[-\frac{(x_1 - 10)^2 + (x_2 - 10)^2}{100}\right]$$

使用5.4节的模拟退火算法来完成最优化任务。函数$f(x_1, x_2)$可以看作一个能量函数。变量$x_1$和$x_2$的整型值可最优化调用，该算法步骤3需要如下修改

$$\boldsymbol{x}_p = \boldsymbol{x} + \Delta\boldsymbol{x}$$

其中$\Delta x$是一个从下面集合随机选择的向量

$$S=\left\{\begin{bmatrix}1\\0\end{bmatrix},\begin{bmatrix}1\\1\end{bmatrix},\begin{bmatrix}0\\1\end{bmatrix},\begin{bmatrix}-1\\1\end{bmatrix},\begin{bmatrix}-1\\0\end{bmatrix},\begin{bmatrix}-1\\-1\end{bmatrix},\begin{bmatrix}0\\-1\end{bmatrix},\begin{bmatrix}1\\-1\end{bmatrix}\right\}$$

使用式（5-33）中的次优“冷却进度表”，该进度表的渐缩因子设置为$\alpha = 0.99$。对不同起始点执行最优化，并且记录对于发现算法的解决方案的迭代次数。

5.6 模拟退火经常用来解决多维组合最优化问题。在这些问题中最优化目标是决定一个$n$维向量，该向量最小化下列基本限制条件下的能量函数$\mathscr{E}(x)$

$$x_k = 1 \quad 或 \quad x_k = 0$$

也就是说，

$$x_k \in \{0, 1\}其中k = 1, 2, \cdots, n$$

常用的一些限制增加到最优化问题。向量$x$的所有元素要么是1要么是0，作为一个二值向量。一个大小为$n$的由所有二值向量组成的集合当作构造空间，并且组合最优化问题能够当作搜索使能量函数能够达到全局最小的某点（也就是，向量$x^*$）的构造空间。

作为简单组合最优化问题的一个例子，考虑能量函数的最小化，该能量函数定义为：

$$\mathscr{E}(x) = 12x_1 + 14x_2 + 22x_3 + 38x_4 + 15x_5 + 13x_6 + 17x_7 + 28x_8 + 4x_9$$

符合

$$x_i \in \{0, 1\} \qquad i = 1, 2, \cdots, 9 \tag{c-1}$$

$$x_1 + x_2 + 2x_3 + 6x_4 + 7x_5 + 8x_6 + 9x_7 + 3x_8 + 11x_9 \leqslant 35 \tag{c-2}$$

235

写一个使用模拟退火方法来解决上面最优化问题的计算机程序。算法应该满足以下条件：

(a) 随机初始化起始点。起始点必须满足限制式（c-1）和式（c-2）。

(b) 设置初始温度并且选择冷却进度表。

(c) 通过在构形空间随机选择向量分量和通过从0到1或从1到0改变它的值来执行转换。

(d) 接受或拒绝在米特罗波利斯标准基础上的变化（参考5.4节）。

(e) 继续构形空间搜索直到

- 到达全局最小值或
- 温度降到预先设定的较小值之下，并且在构造空间内没有明显的变化接受或
- 超过迭代最大次数。

5.7 绝大多数神经网络训练算法建立在局部搜索技术基础之上（最速下降、共轭梯度、牛顿方法等等）。所有建立在局部搜索基础之上的算法有收敛到一个误差实现表面的局部最小值的趋向，因此，提供一个次优的问题解决方案。用于帮助训练网络的算法从一个局部最小值中跳离的技术之一是增加人工高频率噪声到能量代价函数。例如，自适应噪音项能够按照如下算法加入能量函数中

$$\tilde{\mathscr{E}}(w, N) = \mathscr{E}(w) + T(k)N^T x$$

其中$\tilde{\mathscr{E}}(w, N)$是一个扰动能量函数，$w$是网络权值向量，$N$是一个每个分量为一个随机白噪声过程产生的噪声向量，$T(k)$是一个决定能量函数扰动规模的参数。通常$T(k)$的规模随训练过程减小。

(a) 证明一个用于校正调整在扰动代价函数基础之上的权值的最速下降学习规则有如下形式

$$w(k+1) = w(k) + \mu\left[\frac{\partial \mathscr{E}(w)}{\partial w} + T(k)N\right]$$

(b) 考虑求如下定义的能量函数的全局最小值问题

$$\mathscr{E}(w) = \cos^2(3w) + 0.4|w+1| \qquad |w| < 10$$

函数的曲线图如图5-24所示。正如我们看到的，函数有几个局部最小值。应用带有最速下降的扰动能量函数技术求给定函数的全局最小值。

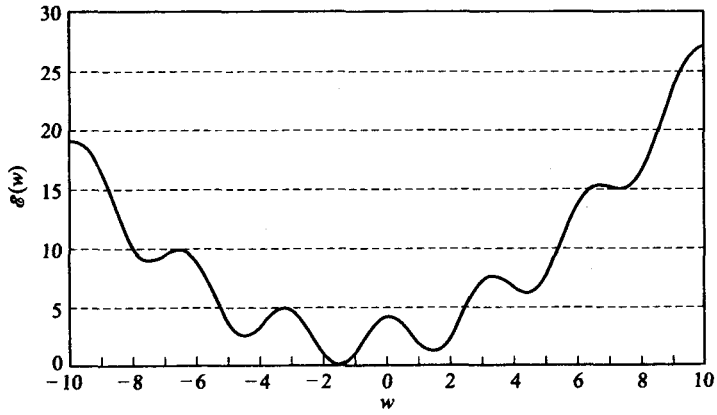



图5-24　问题5.7中最小化的能量函数曲线图

5.8　设计一个简单递归网络，即一个Elman网络，它能够学习探测到三个信号的最高振幅。这些信号与问题5.4中的相似。特别地，这是三个正弦信号，均带有30mHz的频率（模拟次声信号），从0到100秒时间上采样，频率均为1Hz。第一个信号振幅最高为5，第二个信号最高振幅为2.5，第三个信号最高振幅为1。下面的MATLAB命令将产生适当的信号和相关的目标值：

训练信号：

```
t=[0:100];
signal1=5*sin (2*Pi*0.03*t) ;
signal2=2.5*sin (2*Pi*0.03*t) ;
signal3=sin (2*Pi*0.03*t) ;
```

目标值：

```
T1=5*ones (1,101) ;
T2=2.5*ones (1,101) ;
T3=ones (1,101) ;
```

与例5.4一样，复制信号两次，因此每个回合SRN依次训练，该顺序通过重复每个波形两次形成（明显地相关目标值被适当地重复）。

(a) 使用MATLAB神经网络工具箱[10]函数initelm、trainelm和simuelm来初始化、训练和测试网络。用递归（隐藏）神经元数量和训练参数进行测试。需要多少训练回合才能得到合理结果呢？也就是说，使用神经网络工具箱中的MATLAB函数simuelm，使用训练数据作为网络输入，并且得到信号最高振幅的估计。对于相同的曲线图，画出其信号的目标值和信号最高振幅的估计值，你能够推断出什么？

(b) 产生三个30mHz正弦测试信号和相关目标值，与训练数据相似，第一个信号的最高

振幅是5.5，第二个信号的最高振幅是3，第三个信号的最高振幅是1.75。这是部分（a）中训练的SRN以前未见过的三个信号。使用MATLAB函数simuelm，并将信号（以前的复制形式）输入到训练的SRN中。在同一曲线图上，画出信号的目标值和信号的最高振幅的估计。你能推断出什么？SRN进行了推广吗？如果结果不令人满意，做什么来改进结果？ 236~237

5.9 设计一个SRN来实现频率分离。

(a) 在MATLAB中产生两个单位振幅的正弦信号；第一个信号有频率70mHz，第二个信号有频率30mHz。在从0～50秒周期内两个信号均假定以1Hz抽样。

(b) 选择频率目标值的适当描述。使用MATLAB神经网络工具箱中的Elman网络函数，训练SRN，使其能够达到使用15个递归（隐藏）层神经元的精度的合理水平。用各种训练参数测试，你能从得到的结果中推断出什么？

(c) 用（1）不同数量的递归层神经元，（2）不同表示的频率目标值和（3）训练参数来测试。与部分（b）中得到的结论比，有任何改进吗？

(d) 在完成频率估计的SRN训练后，如果结果令人满意，那么使用该网络，在20～60mHz范围内，用不同频率的一致振幅的正弦信号测试。针对网络识别其他频率的能力你能够推断出什么？

## 参考文献


1. J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proceedings of the National Academy of Sciences, USA*, vol. 79, 1982, pp. 2554–8. Reprinted in 1988, Anderson and Rosenfeld [23], pp. 460-4.
2. J. E. Slotine and W. Li, *Applied Nonlinear Control*, Englewood Cliffs, NJ: Prentice-Hall, 1991.
3. S. Haykin, *Adaptive Filter Theory*, 3rd ed., Upper Saddle River, NJ: Prentice-Hall, 1996.
4. J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Redwood City, CA: Addison-Wesley, 1991.
5. W. S. McCulloch and W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, vol. 5, 1943, pp. 115–33. Reprinted in 1988, Anderson and Rosenfeld [23], pp. 18–27.
6. D. J. Amit, *Model Brain Function: The World of Attractor Neural Networks*, New York: Cambridge University Press, 1989.
7. D. J. Amit, H. Gutfreund, and H. Sompolinsky, "Spin Glass Models of Neural Networks," *Physical Review A*, vol. 32, 1985, pp. 1007–18.
8. D. J. Amit, H. Gutfreund, and H. Sompolinsky, "Storing Infinite Numbers of Patterns in a Spin-Glass Model of Neural Networks," *Physical Review Letters*, vol. 55, 1985, pp. 1530–3.
9. J. H. Li, A. N. Michel, and W. Porod, "Analysis and Synthesis of a Class of Neural Networks: Linear Systems Operating on a Closed Hypercube," *IEEE Transactions on Circuits and Systems*, vol. 36, 1989, pp. 1405–22.
10. H. Demuth and M. Beale, *Neural Network Toolbox—For Use with MATLAB*, version 3, Natick, MA: The Mathworks, Inc., 1998.
11. J. J. Hopfield, "Neurons with Graded Response Have Collective Computational Properties like Those of Two-State Neurons," *Proceedings of the National Academy of Sciences*, vol. 81, 1984, pp. 3088–92. Reprinted in 1988, Anderson and Rosenfeld [23], pp. 579–83. 238
12. C. M. Marcus and R. M. Westervelt, "Dynamics of Iterated-Map Neural Networks," *Physical Review A*, vol. 40, 1989, pp. 501–4.

13. R. P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE Acoustics, Speech, and Signal Processing Magazine*, April 1987, pp. 4–22.
14. S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, 1983, pp. 671–80. Reprinted in 1988, Anderson and Rosenfeld [23], pp. 554–67.
15. R. A. Rutenbar, "Simulated Annealing Algorithms: An Overview," *IEEE Circuits and Devices Magazine*, January 1989, pp. 19–26.
16. P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*, Boston: Kluwer Academic Publishers, 1988.
17. B. Hajek, "A Tutorial Survey of Theory and Applications of Simulated Annealing," *Procedings of the 24th IEEE Conference on Decision and Control*, vol. 2, 1985, pp. 755–60.
18. R. V. V. Vidal, ed., *Applied Simulated Annealing*, New York: Springer-Verlag, 1988.
19. B. Müller and J. Reinhardt, *Neural Networks: An Introduction*, Berlin: Springer-Verlag, 1990.
20. E. Shrödinger, *Statistical Thermodynamics*, London: Cambridge University Press, 1946.
21. K. Binder, ed., *The Monte Carlo Method in Condensed Matter Physics*, Topics in Applied Physics, vol. 71, New York: Springer-Verlag, 1992.
22. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," *Journal of Chemical Physics*, vol. 21, 1953, pp. 1087–92.
23. J. A. Anderson and E. Rosenfeld, eds., *Neurocomputing: Foundations of Research*, Cambridge, MA: M.I.T. Press, 1988.
24. S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Baysian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, 1984, pp. 721–41.
25. D. L. Isaacson and R. W. Madsen, *Markov Chains: Theory and Applications*, New York: Wiley, 1976.
26. H. Szu, "Fast Simulated Annealing," in *Neural Networks for Computing*, Snowbird 1986, ed. J. S. Denker, New York: American Institute of Physics, 1986, pp. 420–5.
27. G. E. Hinton and T. J. Sejnowski, " Optimal Perceptual Inference," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington, DC, New York: IEEE, 1983, pp. 448–53.
28. D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A Learning Algorithm for Boltzmann Machines," *Cognitive Science*, vol. 9, 1985, pp. 147–69. Reprinted in 1988, Anderson and Rosenfeld [23], pp. 638–49.
29. G. E. Hinton and T. J. Sejnowski, "Learning and Relearning in Boltzmann Machines," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, eds. D. E. Rumelhart and J. L. McClelland, and the PDP Research Group, vol. 1: *Foundations*, Cambridge, MA: M.I.T. Press, 1986, pp. 282–317.
30. G. E. Hinton, T. J. Sejnowski, and D. H. Ackley, "Boltzmann Machines: Constraint Satisfaction Networks that Learn," Technical Report: CMU-CS-84-119, Pittsburgh, PA: Carnegie-Mellon University, May 1984.
31. P. H. Winston, *Artificial Intelligence*, 2nd ed., Reading, MA: Addison-Wesley, 1984.
32. G. E. Hinton, "Relaxation and Its Role in Vision," Ph.D thesis, University of Edinburgh, Scotland, 1977.
33. R. J. Glauber, "Time-Dependent Statistics of the Ising Model," *Journal of*

239

*Mathematical Physics*, vol. 4, 1963, pp. 294–307.
34. C. Peterson and J. R. Anderson, "A Mean Field Theory Learning Algorithm for Neural Networks," *Complex Systems*, vol. 1, 1987, pp. 995–1019.
35. K. J. Lang and G. E. Hinton, "The Development of the Time-Delay Neural Network Architecture for Speech Recognition," Technical Report CMU-CS-88-152, Pittsburgh, PA: Carnegie-Mellon University, 1988.
36. A. Waibel, T. Hanazama, G. Hinton, K. Shikano, and K. L. Lang, "Phoneme Recognition Using Time Delay Neural Networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-37, 1989, pp. 328–39.
37. A. Waibel, H. Sawai, and K. Shikano, "Modularity and Scaling in Large Phonemic Neural Networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-37, 1989, pp. 1888–98.
38. E. A. Wan, "Temporal Backpropagation for FIR Neural Networks," *IEEE International Joint Conference on Neural Networks*, vol. 1, San Diego, CA, 1990, pp. 575–80.
39. E. A. Wan, "Temporal Backpropagation: An Efficient Algorithm for Finite Impulse Response Neural Networks," in *Proceedings of the 1990 Connectionist Models Summer Schools*, eds. D. S. Touretzky, J. L. Elman, T. J. Sejnowski, and G. E. Hinton, San Mateo, CA: Morgan Kaufmann, 1990, pp. 131–40.
40. E. A. Wan, "Time Series Prediction by Using a Connectionist Network with Internal Time Delays," in *Time Series Prediction: Forecasting the Future and Understanding the Past*, eds. A. S. Weigend and N. A. Gershenfeld, Reading, MA: Addison-Wesley, 1994, pp. 195–217.
41. J. L. Elman, "Finding Structure in Time," *Cognitive Science*, vol. 14, 1990, pp. 179–211.
42. R. J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," *Neural Computation*, vol. 1, 1989, pp. 270–80.
43. S. Haykin and L. Li, "Nonlinear Adaptive Prediction of Nonstationary Signals," *IEEE Transactions on Signal Processing*, vol. 37, 1995, pp. 526–35.
44. K. S. Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Transactions on Neural Networks*, vol. 1, 1990, pp. 4–27.
45. A. V. Levin and K. S. Narendra, "Control of Nonlinear Dynamical Systems Using Neural Networks, II: Observability and Identification," Technical Report 9116, New Haven, CT: Center for Systems Science, Yale Univesity, 1992.
46. K. S. Narendra, *Neural Networks for Identification and Control*, NIPS-95, Tutorial Program, Denver, CO, 1995, pp. 1–46.
47. M. I. Jordan, "Serial Order: A Parallel Distributed Processing Approach," Institute for Cognitive Science Report 8604, San Diego: University of California, 1986.
48. J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 3rd ed., Upper Saddle River, NJ: Prentice-Hall, 1996.

# 第二部分　神经计算的应用

## 第6章　用神经网络解决最优化问题

### 6.1　概述

本章的主要目的是演示神经网络解决约束最优化的问题。一般而言，约束最优化问题假设一些目标代价函数的最小化是受限于加在独立变量上的各种各样的约束。在数学上，约束最优化问题可以用公式表达如下：

最小化

$$f(x_1, x_2, \cdots, x_n) \tag{6-1}$$

受限于

$$r_i(x_1, x_2, \cdots, x_n) = 0 \quad i = 1, 2, \cdots, m \tag{6-2}$$

在各种各样的科学与工程领域中，包括信号处理、回归分析、统计、运筹学等等，都经常遇到这些类型的最优化问题。由于它们在实际应用中的重要性，已被广泛研究，并且已经找到了许多数值方法。这些方法可以在参考文献[1-10]中找到。然而，在这些方法中，大多数需要大量的计算，而且并不适合于需要实时或近似实时最优化结果的应用。神经网络方法为解决约束最优化问题提供了一个不同的发展方向，通过使用具备高度并行计算能力的相对简单的神经网络体系结构，即使相对复杂的最优化问题也可以实时解决。

241
243

本章介绍几种解决一些重要类型的约束最优化问题的神经网络算法：

1. 线性规划，
2. 二次规划，
3. 非线性连续约束最优化问题。

### 6.2　解决线性规划问题的神经网络

线性规划（linear programming，LP）是约束最优化问题的最简单形式。线性规划假设目标函数和约束方程是独立变量的线性组合。根据式（6-1）和式（6-2），对于LP情形，可以把一般约束最优化问题的方程组改写如下：

最小化

$$f(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{n} c_i x_i \tag{6-3}$$

受限于

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \tag{6-4}$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \tag{6-5}$$

$$\vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \tag{6-6}$$

和

$$x_1 \geqslant 0, x_2 \geqslant 0, \cdots, x_n \geqslant 0 \tag{6-7}$$

其中$m < n$, $a_{ij}$, $b_i$, $c_i$表示实常量；$x_i$是独立变量，其值待定以使得目标函数最小化。因为有多种常用符号，LP问题可以多种形式陈述。式（6-3）～式（6-7）的形式通常称为LP问题的标准形式。标准形式可以用更紧凑的向量矩阵符号，改写如下：

最小化

$$f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} \tag{6-8}$$

受限于

244

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{6-9}$$

和

$$\boldsymbol{x} > 0 \tag{6-10}$$

其中$\boldsymbol{x}, \boldsymbol{c} \in \Re^{n\times 1}$, $\boldsymbol{A} \in \Re^{m\times n}$, $\boldsymbol{b} \in \Re^{m\times 1}$。

当用公式表达LP问题时，发现这些约束可以由线性等式和不等式的混合来表达。此外，独立变量不一定需要满足式（6-10）。然而，可以看出无论初始的公式表达是怎样的，每个LP问题都可转化为标准形式。向标准形式的转化可以执行如下：

1. 目标函数$f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x}$的最大化可以用$f(\boldsymbol{x}) = -\boldsymbol{c}^T\boldsymbol{x}$的最小化来代替。

2. 不等式约束形式如

$$\sum_{j=1}^{n} a_{ij}x_j \leqslant \mathrm{b}_i \tag{6-11}$$

可以写作

$$\sum_{j=1}^{n} a_{ij}x_j + x_{n+1} = b_i \tag{6-12}$$

其中$x_{n+1} > 0$是一个新变量，通常称为剩余变量。

3. 不等式约束形式如

$$\sum_{j=1}^{n} a_{ij}x_j \geqslant b_i \tag{6-13}$$

可以写作

$$\sum_{j=1}^{n} a_{ij}x_j - x_{n+1} = b_i \tag{6-14}$$

其中$x_{n+1} > 0$是一个新变量，通常称为松弛变量。

4. 如果约束$x_i \geqslant 0$不适用，变量$x_i$可以用两个新变量替代，使得

$$x_i = x_i^{(1)} - x_i^{(2)} \tag{6-15}$$

且

$$x_i^{(1)} \geqslant 0 \qquad x_i^{(2)} \geqslant 0$$

使用式（6-11）～式（6-15），可以把每个LP问题转化为标准形式。然而，可以看出每个转化得到的等式都引入一个附加的变量，由此增加了问题的维数。鉴于这个原因，在神经网

245 络方法中用LP问题的原始形式考虑问题可能比把它转化为标准形式更好一些。

LP问题的解

解决LP问题可以看作是在向量空间中搜索独立变量 $\tilde{x}$ 的最优向量。最优向量需要满足所有约束，同时使得目标函数最小化。式（6-9）中给出了约束方程组在由独立变量向量构成的$n$维向量空间中确定的$m$个超平面。式（6-9）中的超平面和式（6-10）中的非负约束形成了一个多维多边形，通常称为可行域。每个满足式（6-9）和式（6-10）的独立可变向量都称为可行解，LP问题的任务是找到一个能最小化目标函数的可行向量。对LP问题[1, 4]的几何解释进一步分析揭示出，最优解向量常常在多维多边形的一个顶点上，它有$n-m$个元素等于零，其余变量取非零正值。

一般来说，LP问题的解有四种可能的情况：

1. 唯一解。只有一个解满足所有的约束，且目标函数在可行域里达到最小值。

2. 非唯一解。存在几个可行解使得目标函数达到最小值。

3. 一个无界的解。目标函数在可行域无界，达到$-\infty$。

4. 无可行解。式（6-9）和式（6-10）中约束的限制性太强，可行解的集合为空集。

尽管理论上存在可能性，但第3、4种情况在工程与科学应用中极少出现。而且，这两种情况很容易察觉，在对LP问题的进一步考虑中将假设它有至少一个可行解。

LP问题的对偶形式

由式（6-8）～式（6-10）中公式表达的LP问题通常称为原始LP问题。对于每一个原始LP问题都有另一个与之相关的称为对偶的LP问题。对偶LP问题有如下形式：

最大化

$$g(\boldsymbol{y}) = \boldsymbol{b}^T\boldsymbol{y} \tag{6-16}$$

受限于

$$\boldsymbol{A}^T\boldsymbol{y} \leqslant \boldsymbol{c} \tag{6-17}$$

其中$\boldsymbol{A} \in \Re^{m\times n}$, $\boldsymbol{b} \in \Re^{m\times 1}$, $\boldsymbol{c} \in \Re^{n\times 1}$与式（6-8）～式（6-10）中的对应变量有相同的取值，$\boldsymbol{y} \in \Re^{m\times 1}$是一个对偶独立变量。注意在对偶问题中，独立变量$\boldsymbol{y}$并不一定是非负的。 246

原始LP问题和对偶LP问题之间的关系可以用如下定理表述。为了使内容简短，定理的证明省略了。有兴趣的读者可以在Gass[1]中找到证明。

**定理6.1** 考虑由式（6-8）～式（6-10）定义的LP问题，和式（6-16）、式（6-17）中它的对偶形式。如下陈述中必有一个为真：

（a）原始问题和对偶问题都分别有最优解 $\tilde{\boldsymbol{x}}$ 和 $\tilde{\boldsymbol{y}}$，且满足等式：

$$\boldsymbol{c}^T\tilde{\boldsymbol{x}} = \boldsymbol{b}^T\tilde{\boldsymbol{y}} \tag{6-18}$$

换句话说，在最优解这一点上，LP问题和对偶LP问题的目标函数的值相等。

（b）其中一个问题是无界的，这种情况下另一个问题是不可行的。

（c）两个问题都是不可行的。

此外，可以证明对于任何一对原始LP问题和对偶LP问题的可行解，对偶间隙（duality gap）满足

$$\boldsymbol{c}^T\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{y} \geqslant 0 \tag{6-19}$$

由于对偶间隙是一个在最优解这一点减少到零的非负量，可用来度量解决LP最优化问题的程度。

### 6.2.1 解决LP问题标准形式的神经网络

为了解决LP问题，神经网络实现的第一步是定义一个在无约束情况下可以被最优化的能量函数。为了完成这一步，式（6-9）中的线性约束和式（6-10）中的非负约束以某种方便的方式附加到目标函数中。通常，这些约束合并成惩罚项，即当违反了这些约束时，会增加能量函数的值。两个可以用拉格朗日（Lagrange）乘子法（参见A.6.2节）导出的能量函数定义为[2]

$$E_1(\boldsymbol{x}) = L_1(\boldsymbol{x},\lambda) = \boldsymbol{c}^T\boldsymbol{x} + \frac{K}{2}(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b})^T(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}) + \lambda^T(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}) \tag{6-20}$$

$$E_2(\boldsymbol{x}) = L_2(\boldsymbol{x},\lambda) = \boldsymbol{c}^T\boldsymbol{x} + \frac{K}{2}(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b})^T(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}) + \lambda^T(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}) - \alpha\lambda^T\lambda \tag{6-21}$$

其中$K$, $\alpha \geqslant 0$, $\boldsymbol{\lambda} \in \Re^{m\times 1}$, $\boldsymbol{x} \geqslant 0$。式（6-21）右边最后一项称为正则化项。这一项提高了在病态约束系统下拉格朗日乘子法的稳定性[2]。

247 应用离散时间最速下降法，计算式（6-21）中能量函数对$\boldsymbol{x}$的梯度，得到

$$\begin{aligned}\nabla_x E_2 &= \frac{\partial E_2(\boldsymbol{x},\lambda)}{\partial \boldsymbol{x}} \\ &= \frac{\partial}{\partial \boldsymbol{x}}\left[\boldsymbol{c}^T\boldsymbol{x} + \frac{K}{2}(\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{b} - \boldsymbol{b}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^T\boldsymbol{b}) + \lambda^T(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}) - \alpha\lambda^T\lambda\right]\end{aligned} \tag{6-22}$$

和

$$\frac{\partial E_2(\boldsymbol{x},\lambda)}{\partial \boldsymbol{x}} = \boldsymbol{c} + K\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}) + \boldsymbol{A}^T\lambda = \boldsymbol{c} + \boldsymbol{A}^T(K\boldsymbol{r}+\lambda) \tag{6-23}$$

其中$\boldsymbol{r} \in \Re^{m\times 1}$定义为$\boldsymbol{r} = \boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$。用同样的方式

$$\frac{\partial E_2}{\partial \lambda} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} - \alpha\lambda = \boldsymbol{r} - \alpha\lambda \tag{6-24}$$

基于式（6-23）和式（6-24）用最速法，一组更新方程式可以用如下公式表达

$$x_i(k+1) = \begin{cases} x_i(k) - \mu(k)\left\{c_i + \sum_{j=1}^{m} a_{ji}[Kr_j(k) + \lambda_j(k)]\right\} & \text{如果 } x_i(k+1) > 0 \\ 0 & \text{如果 } x_i(k+1) \leqslant 0 \end{cases} \tag{6-25}$$

和

$$\boldsymbol{\lambda}(k+1) = \boldsymbol{\lambda}(k) + \upsilon(k)[\boldsymbol{r}(k) - \alpha\boldsymbol{\lambda}(k)] \tag{6-26}$$

其中$\boldsymbol{r}(k) = \boldsymbol{A}\boldsymbol{x}(k) - \boldsymbol{b}$, $K$, $\alpha \geqslant 0$, $\mu(k)$, $\upsilon(k) > 0$是学习率参数。注意式（6-25）中独立变化向量的更新方程组确保所有分量都保持非负。这个过程的神经网络体系结构实现如图6-1所示。

**例6.1** 欲解决如下LP问题：

最大化

$$f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} = x_1 + x_2 \tag{6-27}$$

受限于

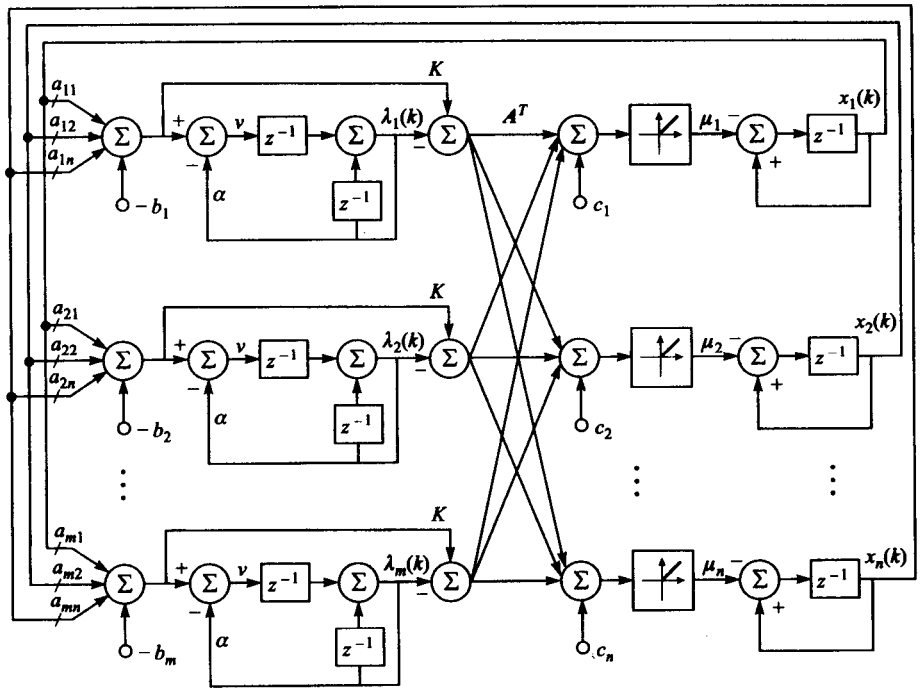
$$-2x_1 + x_2 \leqslant 3 \tag{6-28}$$

$$x_1 + 3x_2 \leqslant 16 \tag{6-29}$$

$$4x_1 + x_2 \leqslant 20 \tag{6-30}$$

$$x_1, x_2 \geqslant 0 \tag{6-31}$$

248 由问题陈述可以看出这个LP问题并不是标准形式。通过添加另外的变量$x_3$, $x_4$, $x_5$，这个LP问题可以转化为如下标准形式：




$z^{-1}$：延迟元素

图6-1 解决标准形式的LP问题的离散时间神经网络，方程（6-25）和方程（6-26）的实现

最大化

$$f(\boldsymbol{x}) = x_1 + x_2 + 0x_3 + 0x_4 + 0x_5 \tag{6-32}$$

受限于

$$-2x_1 + x_2 + x_3 = 3 \tag{6-33}$$

$$x_1 + 3x_2 + x_4 = 16 \tag{6-34}$$

$$4x_1 + x_2 + x_5 = 20 \tag{6-35}$$

为了解决这个LP问题，模拟图6-1中的神经网络。选择$\mu = 0.01$, $\eta = 0.01$, $K = 0$, $\alpha = 0$作为神经网络的参数。对$\boldsymbol{x}$和$\boldsymbol{\lambda}$都假设零初始条件。图6-2分别显示了五个独立变量的轨迹。可以看出网络在大约3000步迭代内收敛。给出的LP的解是$\tilde{\boldsymbol{x}} = [4.0042 \quad 3.9953]^T$，与精确解$\tilde{\boldsymbol{x}}^* = [4 \quad 4]^T$的误差在学习率参数的精度以内。注意神经网络方法也提供了其余变量的解。

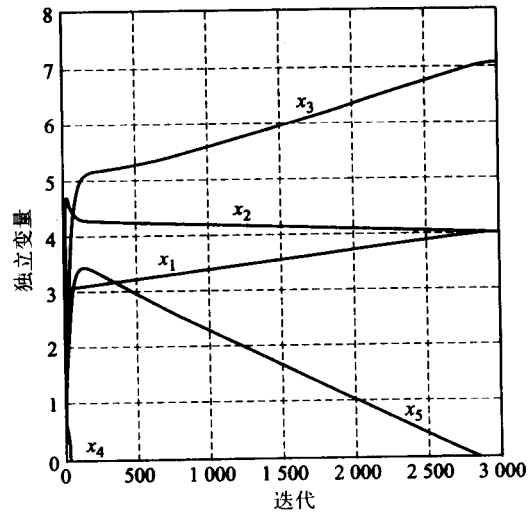



图6-2 例6.1中LP问题的神经网络解的独立变量轨迹

249

### 6.2.2 解决LP问题非标准形式的神经网络

前一节演示了使用神经网络解决标准形式的LP问题。从例6.1中可以看出这种方法的基本折衷。如果要解决的LP问题是非标准

形式的，将它转化成标准形式会增加问题的维数，从而增加问题的复杂性。这一节将说明，在计算上，用原始形式考虑LP问题比用等价的标准形式更有利。

具有不等式约束的LP问题的神经网络体系结构

考虑一个LP问题，它的所有约束都用不等式的形式来表达：

最小化

$$f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} \tag{6-36}$$

受限于

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n - b_i \leqslant 0 \quad \text{对于} \quad i = 1, 2, \cdots, m \tag{6-37}$$

和

$$x_1 \geqslant 0, x_2 \geqslant 0, \cdots, x_n \geqslant 0 \tag{6-38}$$

通常把式（6-37）的约束简写成如下标记

250

$$r_i(\boldsymbol{x}) = a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n - b_i, \quad \text{对于} \quad i = 1, 2, \cdots, m \tag{6-39}$$

与前一节对标准形式的LP问题采用的方法相似，寻找一个能够用神经网络容易求解的合适的能量函数。在这种情况下，需要构造一个能够惩罚违反每个不等式约束条件的能量函数。设计出这个能量函数后，可采用最速下降法或任何其他无约束最优化技术来解决这个LP问题。

考虑一个如下定义的能量函数

$$E(\boldsymbol{x}, K) = \boldsymbol{c}^T\boldsymbol{x} + K\sum_{i=1}^{m}\Phi[r_i(\boldsymbol{x})] \tag{6-40}$$

其中

$$\Phi[r_i(\boldsymbol{x})] \begin{matrix} = 0 & \text{如果} \ r_i(\boldsymbol{x}) \leqslant 0 \\ > 0 & \text{如果} \ r_i(\boldsymbol{x}) > 0 \end{matrix} \tag{6-41}$$

且

$$x_1 \geqslant 0, x_2 \geqslant 0, \cdots, x_n \geqslant 0 \tag{6-42}$$

这个函数由两项组成。式（6-40）中右边第一项是待最小化的LP问题的目标函数。第二项是对违反约束的惩罚。函数$\boldsymbol{\Phi}(\boldsymbol{v})$可以选用任何具有式（6-41）所述特性的分段可微函数。正参数$K$控制式（6-40）～式（6-42）中无约束最优化问题接近式（6-36）～式（6-38）中原始LP问题。很容易看出当$K$趋于正无穷时，这两个问题成为等价的。因此，参数$K$通常选取一个足够大的正数。应用最速下降技术，有下式：

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) - \mu\frac{\partial E(\boldsymbol{x})}{\partial \boldsymbol{x}} \tag{6-43}$$

对式（6-40）中能量函数求偏导数得到

$$\frac{\partial E(\boldsymbol{x})}{\partial(\boldsymbol{x})} = \frac{\partial}{\partial \boldsymbol{x}}\left\{\boldsymbol{c}^T\boldsymbol{x} + K\sum_{i=1}^{m}\Phi[r_i(\boldsymbol{x})]\right\} = \boldsymbol{c} + K\sum_{i=1}^{m}\Psi[r_i(\boldsymbol{x})]\frac{\partial}{\partial \boldsymbol{x}}[r_i(\boldsymbol{x})] \tag{6-44}$$

其中 $\Psi(\upsilon) = \dfrac{d\Phi(\upsilon)}{d\upsilon} = \Phi'(\upsilon)$。结合式（6-39）和式（6-44），有：

$$\frac{\partial E(\boldsymbol{x})}{\partial \boldsymbol{x}} = \boldsymbol{c} + K\sum_{i=1}^{m}\Psi[r_i(\boldsymbol{x})]\begin{bmatrix} a_{i1} \\ a_{i2} \\ \vdots \\ a_{in} \end{bmatrix} \tag{6-45}$$

考虑式（6-38）中的非负约束，并把式（6-45）代入式（6-43），获得一组更新方程式 [251]

$$x_j(k+1)=\begin{cases}x_j(k)-\mu_j\left\{c_j+K\sum_{i=1}^{m}\Psi[r_i(\boldsymbol{x})]a_{ij}\right\} & \text{如果 } x_j(k+1)\geqslant 0\\ 0 & \text{如果 } x_j(k+1)<0\end{cases} \tag{6-46}$$

基于式（6-46），构造如图6-3所示的相应神经网络体系结构。为了简洁，函数$\Phi(\upsilon)$通常选用

$$\Phi(\upsilon)=\begin{cases}\frac{1}{2}\upsilon^2 & \text{对于 } \upsilon>0\\ 0 & \text{对于 } \upsilon\leqslant 0\end{cases} \tag{6-47}$$

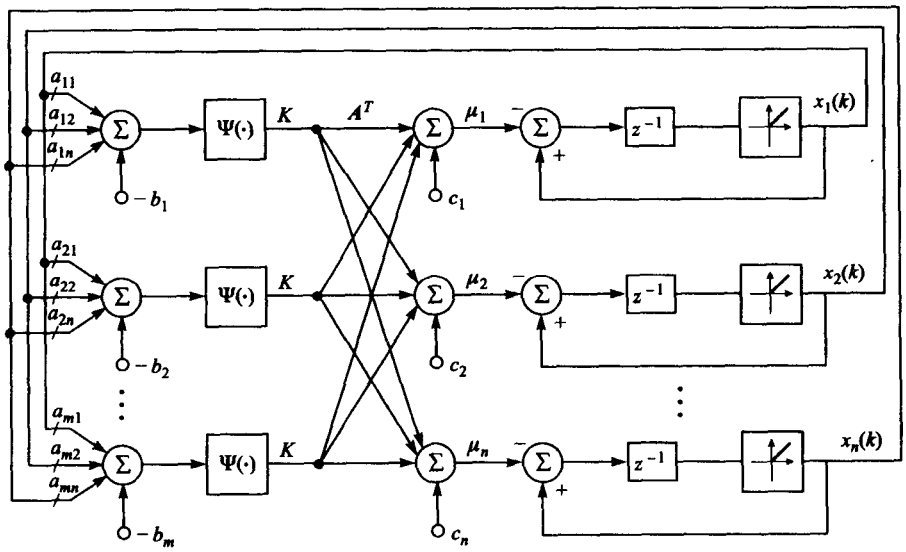



图6-3　求解非标准形式LP问题的离散时间神经网络，方程（6-46）的实现

另一个可用来解决带有不等式约束的LP问题的能量函数可定义为[2]

$$E(\boldsymbol{x})=\boldsymbol{c}^T\boldsymbol{x}+K\sum_{i=1}^{m}\max\{0,r_i(\boldsymbol{x})\} \tag{6-48}$$

其中$K>0$。式（6-48）对$\boldsymbol{x}$的梯度给定如下：

$$\frac{\partial E(\boldsymbol{x})}{\partial(\boldsymbol{x})}=c+K\sum_{i=1}^{m}S_i[a_{i1},a_{i2},\cdots,a_{in}]^T \tag{6-49}$$

因此，采用式（6-43），离散时间学习为

$$x_j(k+1)=x_j(k)-\mu_j\left(c_j+K\sum_{i=1}^{m}S_ia_{ij}\right) \tag{6-50}$$ [252]

其中

$$S_i=\begin{cases}1 & \text{如果 } r_i(\boldsymbol{x})>0\\ 0 & \text{如果 } r_i(\boldsymbol{x})\leqslant 0\end{cases} \tag{6-51}$$

当然必须满足式（6-42）中的约束，并且在每个离散时间步$k$有

$$x_j(k)=\max\{x_j(k),0\} \tag{6-52}$$

实现式（6-50）和式（6-51）的神经网络体系结构如图6-4所示。注意它与图6-3中的网络有很相似的结构。

**例6.2**　图6-3和图6-4中的神经网络体系结构用来解决例6.1的LP问题。图6-3中网络的参数选用$\mu_0 = 0.005, K = 5$。第二个网络的参数也是$\mu_0 = 0.005, K = 5$。独立变量的轨迹如图6-5所示。两种情况的初始条件设为$\boldsymbol{x}_0 = [1\ \ 2]^T$。在训练过程中，学习率按照如下进度减少

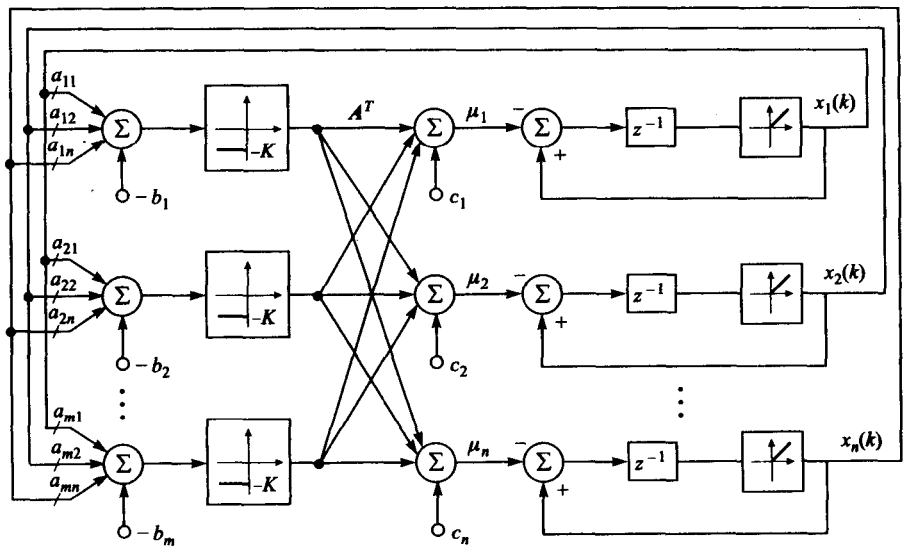



图6-4　求解非标准形式LP问题的离散时间神经网络，方程（6-50）和方程（6-51）的实现

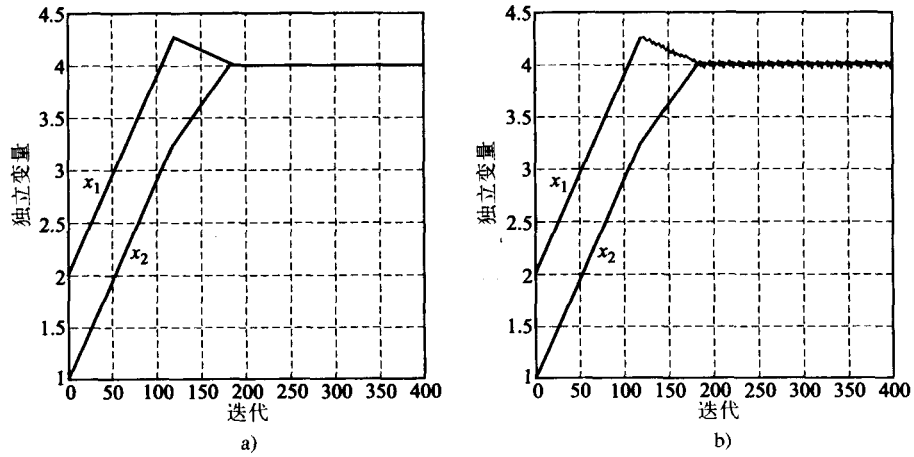



图6-5　例6.1中LP问题的神经网络解的独立变量轨迹：a）使用图6-3所示的网络结构；b）使用图6-4所示的网络结构

$$\mu = \frac{\mu_0}{\log(1+k)} \tag{6-53}$$

注意两个网络都比例6.1中使用的网络收敛快得多，在例6.1中把LP问题转化成了标准形式。从第一个网络获得的解是$\tilde{\boldsymbol{x}}_1^T = [4.0127\ \ 4.0218]$，第二个网络的解是$\tilde{\boldsymbol{x}}_2^T = [4.0060\ \ 3.9976]$。两个解的精度都不比学习率参数的精度差。

具有混合约束的LP问题的神经网络

前几节专门介绍使用神经网络体系结构解决两种重要情形下的LP问题：全部约束都是等式约束（LP问题的标准形式），和全部约束都是不等式约束。一般来说，LP问题可以用两种约束来表达。通过增加剩余或松弛变量，任何LP问题都可以转化成标准形式。然而，不这么做往往更有益。换句话说，在线性规划的神经网络方法中，处理问题的原始形式可以获得明显优势。

考虑如下具有混合约束的LP问题：

最小化

$$f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} = \sum_{i=1}^{n} c_i x_i \tag{6-54}$$

受限于

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \tag{6-55}$$

$$\vdots$$

$$a_{p1}x_1 + a_{p2}x_2 + \cdots + a_{pn}x_n = b_p \tag{6-56}$$

$$a_{p+1,1}x_1 + a_{p+1,2}x_2 + \cdots + a_{p+1,n}x_n \leqslant b_{p+1} \tag{6-57}$$

$$\vdots \tag{6-58}$$

$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leqslant b_m \tag{6-59}$$

254

和

$$x_1, x_2, \cdots, x_n \geqslant 0 \tag{6-60}$$

这个$m$约束集可以分成两个子集。第一个子集由$p$个等式约束组成，第二个子集有$m-p$个不等式约束。由于这两个子集不相交，可以用式（6-20）、式（6-21）、式（6-40）和式（6-48）的能量函数的复合作为能量函数。依赖于复合能量函数的形成，可以推导出几种不同的学习算法和相应的神经网络体系结构。例如，用式（6-21）和式（6-40），可以如下表达能量函数

$$E(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} + \frac{K_1}{2}(\boldsymbol{A}_p\boldsymbol{x} - \boldsymbol{b}_p)^T(\boldsymbol{A}_p\boldsymbol{x} - \boldsymbol{b}_p) + \boldsymbol{\lambda}_p^T(\boldsymbol{A}_p\boldsymbol{x} - \boldsymbol{b}_p) - \alpha\boldsymbol{\lambda}_p^T\boldsymbol{\lambda}_p + \frac{K_2}{2}\sum_{i=p+1}^{m}\Phi[r_i(\boldsymbol{x})] \tag{6-61}$$

其中

$$K_1, K_2, \alpha \geqslant 0 \tag{6-62}$$

$$\boldsymbol{A}_p = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{p1} & a_{p2} & \cdots & a_{pn} \end{bmatrix} \qquad \boldsymbol{b}_p = [b_1, b_2, \cdots, b_p]^T \tag{6-63}$$

$$\boldsymbol{\lambda}_p = [\lambda_1, \lambda_2, \cdots, \lambda_p]^T \tag{6-64}$$

函数$\Phi(v)$是满足式（6-41）的分段可微函数。

式（6-61）中的能量函数由三种不同类型的项组成。第一种是

$$T_1 = \boldsymbol{c}^T\boldsymbol{x} \tag{6-65}$$

这是一个待最小化的LP目标函数。第二种

$$T_2 = \frac{K_1}{2}(\boldsymbol{A}_p\boldsymbol{x} - \boldsymbol{b}_p)^T(\boldsymbol{A}_p\boldsymbol{x} - \boldsymbol{b}_p) + \boldsymbol{\lambda}_p^T(\boldsymbol{A}_p\boldsymbol{x} - \boldsymbol{b}_p) - \alpha\boldsymbol{\lambda}_p^T\boldsymbol{\lambda}_p \tag{6-66}$$

对每一个违反等式约束的惩罚。最后，第三种

$$T_3 = \frac{K_2}{2}\sum_{i=p+1}^{m}\Phi[r_i(\boldsymbol{x})] \tag{6-67}$$

对违反不等式约束的惩罚。应用最速下降法，更新方程式可以写作

255

$$\boldsymbol{\lambda}_p(k+1) = \boldsymbol{\lambda}_p(k) + \upsilon(k)[\boldsymbol{r}_p(k) - \alpha\boldsymbol{\lambda}_p(k)] \tag{6-68}$$

和

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) - \mu(k)\left\{\boldsymbol{c} + \boldsymbol{A}_p^T[K_1\boldsymbol{r}_p(k) - \lambda_p(k)] + K_2\sum_{i=p+1}^{m}\Psi[r_i(\boldsymbol{x})]\cdot\begin{bmatrix}a_{i1}\\ a_{i2}\\ \vdots\\ a_{in}\end{bmatrix}\right\} \tag{6-69}$$

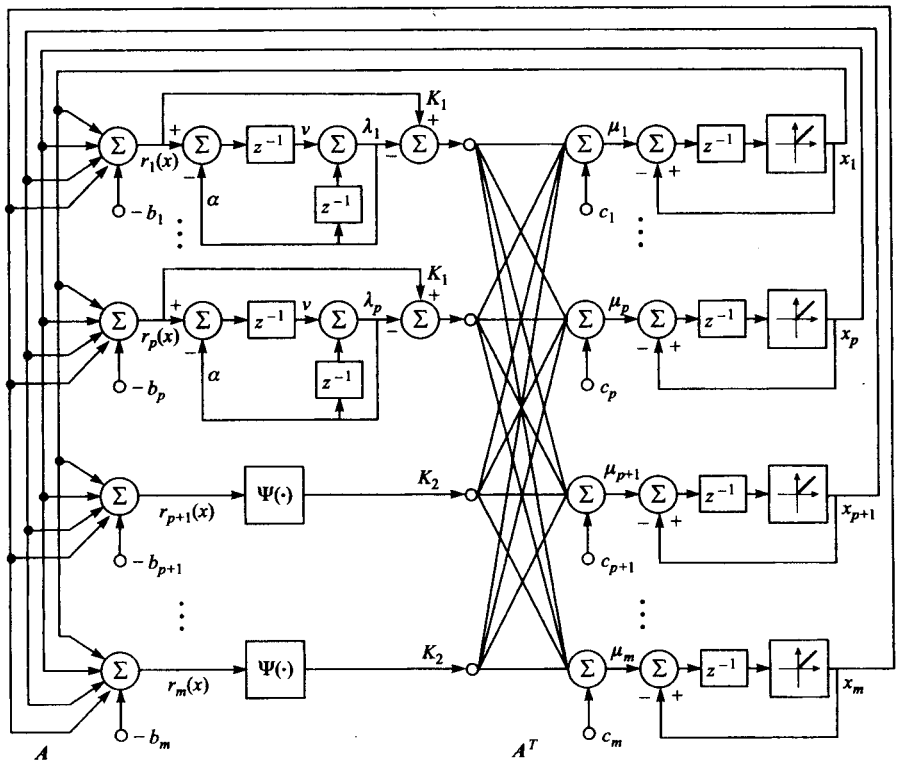
其中

$$\boldsymbol{r}_p = \boldsymbol{A}_p\boldsymbol{x} - \boldsymbol{b}_p \tag{6-70}$$

$$\Psi(\upsilon) = \frac{\mathrm{d}\Phi(\upsilon)}{\mathrm{d}\upsilon} \tag{6-71}$$

且$\mu(k)$, $\upsilon(k) > 0$是学习率参数。

这个过程的神经网络体系结构实现如图6-6所示。注意，其中强制约束独立变量为正。




256　图6-6　解决非标准LP问题的离散时间神经网络，方程（6-68）和方程（6-69）的实现

## 6.3　解决二次规划问题的神经网络

二次规划（quadratic programming, QP）问题是非线性最优化问题的特殊情况，是在线性约束条件下最优化一个二次目标函数。然而，由于它在日常实践中的重要性和频繁出现，被

当作一类单独的问题。本节考虑解决根据线性约束的形式定义的三种类型的QP问题的神经网络方法。在考虑QP问题的各种形式之前，先看看一般的二次型。

二次型

二次型是所有QP问题中都要最优化的目标函数的一部分。本节定义二次型，并简要介绍它与QP问题有关的性质。

考虑向量$\boldsymbol{x} \in \Re^{n\times 1}$。定义一个函数为

$$f(\boldsymbol{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_i x_j \tag{6-72}$$

称作向量$\boldsymbol{x}$的二次型。系数$q_{ij}$常常排列成矩阵，式（6-72）可以用更紧凑的形式重写为

$$f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} \tag{6-73}$$

其中$\boldsymbol{Q} \in \Re^{n\times n}$。对于每一个$\boldsymbol{x}$和$\boldsymbol{Q}$，乘积$\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}$是一个标量，有

$$\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} = (\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x})^T = \boldsymbol{x}^T\boldsymbol{Q}^T\boldsymbol{x} \tag{6-74}$$

因此，

$$\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} = \frac{1}{2}(\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}^T\boldsymbol{x}) = \boldsymbol{x}^T\frac{\boldsymbol{Q}+\boldsymbol{Q}^T}{2}\boldsymbol{x} = \boldsymbol{x}^T\overline{\boldsymbol{Q}}\boldsymbol{x} \tag{6-75}$$

因此，矩阵$\overline{\boldsymbol{Q}}$是一个对称矩阵，其元素为

$$\overline{q}_{ij} = \frac{1}{2}(q_{ij} + q_{ji}) \tag{6-76}$$

其中$q_{ij}$和$q_{ji}$是$\boldsymbol{Q}$的元素。由式（6-75）和式（6-76），很明显，对于每一个矩阵$\boldsymbol{Q}$，可以用对称系数矩阵$\overline{\boldsymbol{Q}}$来构造一个等价的二次型。由于这个原因，为不失一般性，可以假设$\boldsymbol{Q}$是一个实对称矩阵。

二次型称为正定的（参考A.2.6节），如果对于每一个非零的$\boldsymbol{x} \in \Re^{n\times 1}$，

$$\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} > 0 \tag{6-77}$$

成立。二次型称为半正定的，如果对于所有非零$\boldsymbol{x} \in \Re^{n\times 1}$， 257

$$\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} \geqslant 0 \tag{6-78}$$

成立，并且至少存在一个向量$\boldsymbol{x} \neq \boldsymbol{0}$使得$\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} = 0$。通过把式（6-77）和式（6-78）中不等号适当反向，还可以定义负定和半负定形式。

与非线性规划相关的一个基本数学概念是凸函数（参见A.3.1节定义A.11）。定义在$\Re^{n\times 1}$中一个凸集$\boldsymbol{D}$上的函数$f(\boldsymbol{x})$，如果对于$\boldsymbol{D}$中任意两个点$\boldsymbol{x}_1$、$\boldsymbol{x}_2$与任意$0 \leqslant \lambda \leqslant 1$，

$$f[\lambda\boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2] \leqslant \lambda f(\boldsymbol{x}_1) + (1-\lambda)f(\boldsymbol{x}_2) \tag{6-79}$$

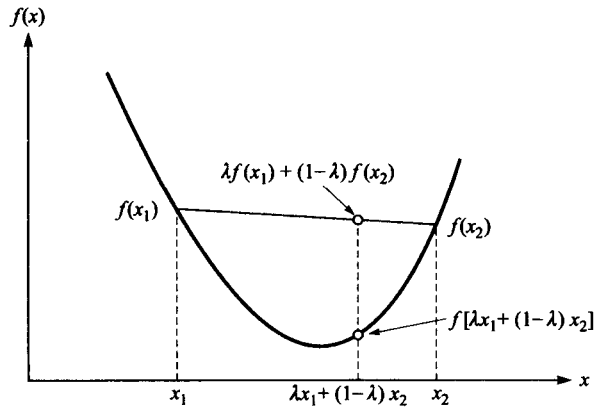
则$f(\boldsymbol{x})$称为凸函数。如果$-f(\boldsymbol{x})$是凸函数，则$f(\boldsymbol{x})$称为凹函数。当$\boldsymbol{x} \in \Re$时，凸函数的概念如图6-7所示。如果函数是凸的，它在区间$(\boldsymbol{x}_1, \boldsymbol{x}_2)$上任意一点的函数值小于连接点$f(\boldsymbol{x}_1)$和$f(\boldsymbol{x}_2)$的直线上的相应值。下面的定理介绍了凸函数的一个重要性质。为了简洁，定理的证明省略，证明可以在Wilde and Beightler[3]中找到。

**定理6.2** 如果$f(\boldsymbol{x})$在凸集$\boldsymbol{D}$中是凸的，那么$f(\boldsymbol{x})$最多只有一个局部最小点。如果存在这样一个最小点，它就是全局最小点并且在凸集$\boldsymbol{D}$上获得。

定理6.2在最优化理论中有深远的重要性。它解决了许多基于某种梯度技术的迭代算法中常见的一个问题——逃离误差（能量）曲面的局部最小点。本质上，这个定理说明，如果能

量函数是凸的，则局部最小点问题就不存在，并且基于梯度的最小化过程会确保终止于全局最小点。

式（6-73）中的二次型是非线性规划中使用的许多能量函数的一个常见部分。除了许多非线性规划问题本来就包含二次型的事实外，下面的定理揭示了这种形式常见的原因。




258

图6-7　凸函数的例子

**定理6.3**　让$\boldsymbol{x}\in\Re^{n\times 1}$，$f(\boldsymbol{x})=\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}$是定义在凸集$\boldsymbol{D}\subset\Re^{n\times 1}$上的一个二次型。如果矩阵$\boldsymbol{Q}$是半定的，则二次型$f(\boldsymbol{x})$在集合$\boldsymbol{D}$上是凸的。

**证明**：基于式（6-79）的凸函数定义，希望证明对于$0\leqslant\lambda\leqslant 1$和所有$\boldsymbol{x}_1,\boldsymbol{x}_2\in\boldsymbol{D}$，有

$$f[\lambda\boldsymbol{x}_1+(1-\lambda)\boldsymbol{x}_2]-\lambda f(\boldsymbol{x}_1)-(1-\lambda)f(\boldsymbol{x}_2)\leqslant 0 \tag{6-80}$$

由于$\boldsymbol{Q}$是对称矩阵，有$\boldsymbol{x}_1^T\boldsymbol{Q}\boldsymbol{x}_2=\boldsymbol{x}_2^T\boldsymbol{Q}\boldsymbol{x}_1$。通过把表达式写成二次型的形式，式（6-80）的左边可以重写成

$$\begin{aligned}
&[\lambda\boldsymbol{x}_1+(1-\lambda)\boldsymbol{x}_2]^T\boldsymbol{Q}[\lambda\boldsymbol{x}_1+(1-\lambda)\boldsymbol{x}_2]-\lambda\boldsymbol{x}_1^T\boldsymbol{Q}\boldsymbol{x}_1-(1-\lambda)\boldsymbol{x}_2^T\boldsymbol{Q}\boldsymbol{x}_2\\
&=\lambda^2\boldsymbol{x}_1^T\boldsymbol{Q}\boldsymbol{x}_1+(1-\lambda)^2\boldsymbol{x}_2^T\boldsymbol{Q}\boldsymbol{x}_2+2\lambda(1-\lambda)\boldsymbol{x}_1^T\boldsymbol{Q}\boldsymbol{x}_2-\lambda\boldsymbol{x}_1^T\boldsymbol{Q}\boldsymbol{x}_1-(1-\lambda)\boldsymbol{x}_2^T\boldsymbol{Q}\boldsymbol{x}_2\\
&=(\lambda^2-\lambda)\boldsymbol{x}_1^T\boldsymbol{Q}\boldsymbol{x}_1+(1-\lambda)[(1-\lambda)\boldsymbol{x}_2^T\boldsymbol{Q}\boldsymbol{x}_2-\boldsymbol{x}_2^T\boldsymbol{Q}\boldsymbol{x}_2]+2\lambda(1-\lambda)\boldsymbol{x}_1^T\boldsymbol{Q}\boldsymbol{x}_2\\
&=\lambda(\lambda-1)\boldsymbol{x}_1^T\boldsymbol{Q}\boldsymbol{x}_1+\lambda(\lambda-1)\boldsymbol{x}_2^T\boldsymbol{Q}\boldsymbol{x}_2-2\lambda(\lambda-1)\boldsymbol{x}_1^T\boldsymbol{Q}\boldsymbol{x}_2\\
&=\lambda(\lambda-1)[\boldsymbol{x}_1^T\boldsymbol{Q}\boldsymbol{x}_1+\boldsymbol{x}_2^T\boldsymbol{Q}\boldsymbol{x}_2-2\boldsymbol{x}_1^T\boldsymbol{Q}\boldsymbol{x}_2]\\
&=\lambda(\lambda-1)[\boldsymbol{x}_1-\boldsymbol{x}_2)^T\boldsymbol{Q}(\boldsymbol{x}_1-\boldsymbol{x}_2)]
\end{aligned} \tag{6-81}$$

由于$\boldsymbol{Q}$是一个半定矩阵，有

$$(\boldsymbol{x}_1-\boldsymbol{x}_2)^T\boldsymbol{Q}(\boldsymbol{x}_1-\boldsymbol{x}_2)\geqslant 0 \tag{6-82}$$

由于$0\leqslant\lambda\leqslant 1$，则

$$\lambda(\lambda-1)\leqslant 0 \tag{6-83}$$

所以，式（6-80）的不等式成立，证毕。

定理6.3说明半定二次型是一个凸函数。因此，根据定理6.2，可以用基于梯度的迭代技术轻易找到唯一极小点。然而，如果矩阵$\boldsymbol{Q}$是不定的，有可能这个二次型会有多个局部极小点。从实际情况来看，这个限制很少碰到，因为大多数二次规划问题可以用半正定的$\boldsymbol{Q}$表达为二次型的形式。鉴于此，在后面的节中，假设$\boldsymbol{Q}$是对称半正定矩阵。如果不符合这一点，下面描述的一些神经网络算法将不能收敛到全局最小点。

用于标准形式QP问题的神经网络

与LP问题的情况相似，QP问题也可以表达为标准形式。QP问题的标准形式如下：

最小化

$$f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} \tag{6-84}$$

受限于

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{6-85}$$

259

和

$$x_1, x_2, \cdots, x_n \geqslant 0 \tag{6-86}$$

其中$\boldsymbol{x} \in \Re^{n\times 1}$, $\boldsymbol{c} \in \Re^{n\times 1}$, $\boldsymbol{Q} \in \Re^{n\times n}$, $\boldsymbol{A} \in \Re^{m\times n}$, $\boldsymbol{b} \in \Re^{m\times 1}$, $m < n$，并且假设$\boldsymbol{Q}$是对称半正定矩阵。

为了使用神经网络方法，需要定义一个简便的能量函数。使用增广的拉格朗日乘子法（参见A.6.2节），可以如[2]定义一个能量函数

$$E(\boldsymbol{x},\boldsymbol{\lambda}) = \boldsymbol{c}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) + \frac{K}{2}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \tag{6-87}$$

其中$\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \cdots, \lambda_m]^T \in \Re^{m\times 1}$，$k \geqslant 0$是一个惩罚参数。

应用梯度法，可以得到网络更新方程为

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) - \mu\nabla_x E(\boldsymbol{x}, \boldsymbol{\lambda}) \tag{6-88}$$

和

$$\boldsymbol{\lambda}(k+1) = \boldsymbol{\lambda}(k) + \eta\nabla_\lambda E(\boldsymbol{x}, \boldsymbol{\lambda}) \tag{6-89}$$

其中$\mu, \eta > 0$是学习率参数。在确定了式（6-88）和式（6-89）的梯度之后，学习规则是

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) - \mu\{\boldsymbol{c} + \boldsymbol{Q}\boldsymbol{x}(k) + \boldsymbol{A}^T\boldsymbol{\lambda}(k) + K\boldsymbol{A}^T[\boldsymbol{A}\boldsymbol{x}(k) - \boldsymbol{b}]\} \tag{6-90}$$

和

$$\boldsymbol{\lambda}(k+1) = \boldsymbol{\lambda}(k) + \eta(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \tag{6-91}$$

这个过程的神经网络体系结构的实现如图6-8所示。

作为最后一个注释，考虑与式（6-87）定义的能量函数相关的黑塞矩阵（参见A.3.5节）

$$\boldsymbol{H} = \frac{\partial^2 E(\boldsymbol{x},\lambda)}{\partial \boldsymbol{x}^2} = \boldsymbol{Q} + K\boldsymbol{A}^T\boldsymbol{A} \tag{6-92}$$

由式（6-92）看出，这个能量函数的黑塞矩阵是半正定的，如果$\boldsymbol{Q}$是半正定的，则对$K = 0$也成立。由于这正是大多数QP问题的情形，图6-8中的网络确保收敛到唯一的全局最小点。然而，即使在$\boldsymbol{Q}$不是半正定的情况下，如果参数$K$设成一个足够大的正值，黑塞矩阵就可以强制成为正定的。由于这个原因，当$\boldsymbol{Q}$的一些特征值为相对较小的正数或$\boldsymbol{Q}$不是正定的时候，惩罚项$(K/2)(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$趋向于提高网络的收敛性。

**例6.3** 考虑一个QP问题如下：

最大化

$$f(\boldsymbol{x}) = x_1 + x_2 - x_1^2 - 3x_2^2 \tag{6-93}$$

260

受限于

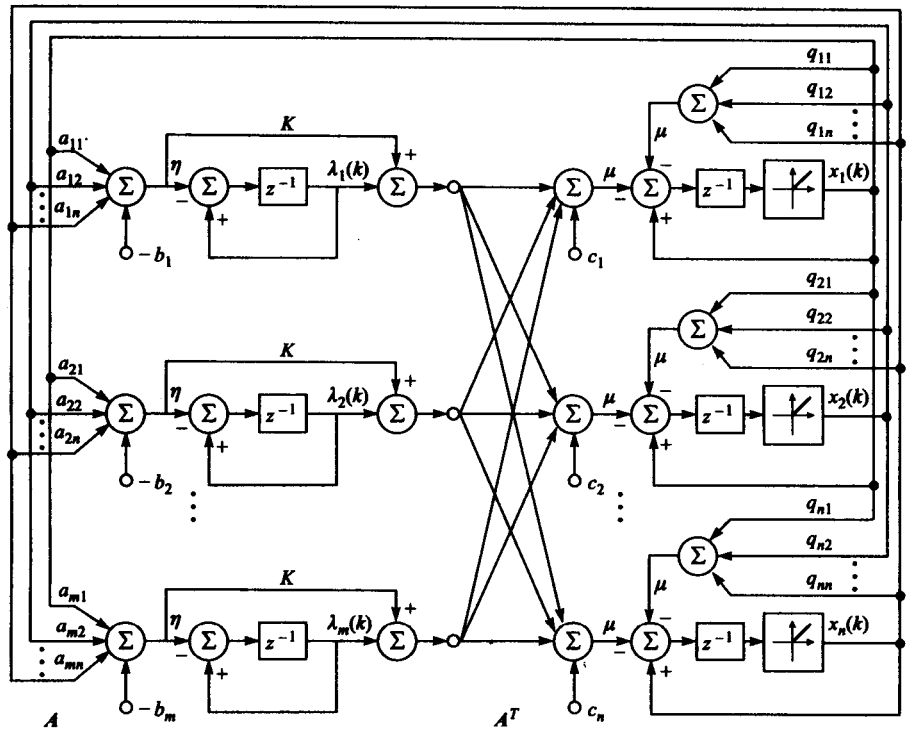



图6-8　解决标准QP问题的离散时间神经网络，方程（6-88）和方程（6-89）的实现

$$-2x_1 + x_2 \leqslant 3 \tag{6-94}$$

$$x_1 + 3x_2 \leqslant 16 \tag{6-95}$$

$$4x_1 + x_2 \leqslant 20 \tag{6-96}$$

$$x_1, x_2 \geqslant 0 \tag{6-97}$$

由于问题不是标准形式的，可以通过加一些额外变量进行变换。问题的等价的标准形式如下：

最小化

$$f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} = -x_1 - x_2 + x_1^2 + 3x_2^2 \tag{6-98}$$

受限于

$$-2x_1 + x_2 + x_3 = 3 \tag{6-99}$$

261

$$x_1 + 3x_2 + x_4 = 16 \tag{6-100}$$

$$4x_1 + x_2 + x_5 = 20 \tag{6-101}$$

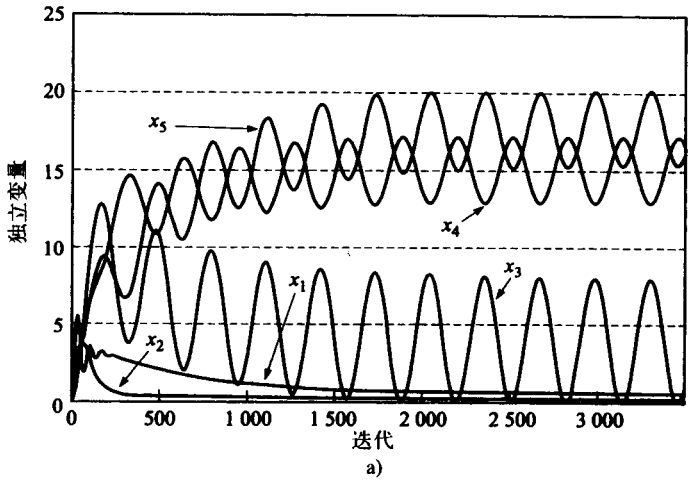
$$x_1, x_2, \cdots, x_5 \geqslant 0 \tag{6-102}$$

图6-8所示的神经网络用来解决这个QP问题。选择学习率$\mu = 0.01$和$\eta = 0.01$。图6-9a、b分别显示两个不同参数$K$值下的独立变量$x_i\ (i = 1, 2, \cdots, 5)$的轨迹。可以看出$K$取非零值显著地提高了网络的收敛性。QP问题的一个精确解是$x_1 = 0.5$和$x_2 = 0.1667$。在两种情况下，网络都收敛到了实际情况下学习率精度中的值。

*具有不等式约束的QP问题的神经网络*

在前一节中，考察了解决标准形式的QP问题的神经网络方法。如同例6.3中演示的，即使问题不是标准形式的，也总是可以通过增加新的独立变量转换成标准形式。作为转换而付出

的代价是问题维数的增加和相应神经网络大小的增加。本节专注于解决具有不等式约束的QP问题的神经网络方法。



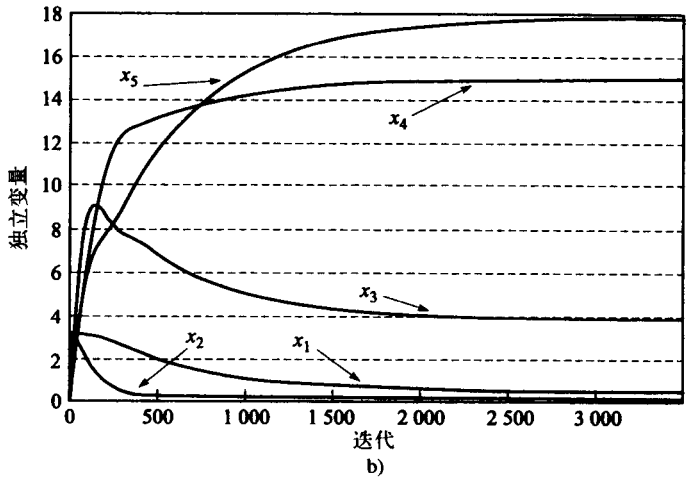




图6-9 例6.3中QP问题的独立变量的轨迹。a) $K=0$，网络收敛到$x_1=0.5176$和$x_2=0.1642$；b) $K=1$，网络收敛到$x_1=0.5069$和$x_2=0.1673$

具有不等式约束的QP问题形式如下：

最小化

$$f(\boldsymbol{x})=\boldsymbol{c}^T\boldsymbol{x}+\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} \tag{6-103}$$

受限于

$$\boldsymbol{A}\boldsymbol{x}\leqslant\boldsymbol{b} \tag{6-104}$$

和

$$\boldsymbol{x}\geqslant 0 \tag{6-105}$$

其中$\boldsymbol{x}, \boldsymbol{c}\in\Re^{n\times 1}, \boldsymbol{A}\in\Re^{m\times n}, \boldsymbol{b}\in\Re^{m\times 1}, \boldsymbol{Q}\in\Re^{n\times n}$是对称半正定矩阵。

为了解决式（6-103）～式（6-105）定义的QP问题，可以扩充具有不等式约束的LP问题中使用的方法。可以定义

$$r_i(\boldsymbol{x}) = \sum_{j=1}^{n} a_{ij}x_j - b_i \tag{6-106}$$

并且用公式表达两个不同形式的能量函数。第一种形式可以写成

$$E_1(\boldsymbol{x},K) = \boldsymbol{c}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + k\sum_{i=1}^{m}\Phi[r_i(\boldsymbol{x})] \tag{6-107}$$

其中$\Phi(v)$是任意满足下列性质的分段可微函数：

$$\Phi[r_i(\boldsymbol{x})]\begin{cases}=0 & \text{如果 } r_i(\boldsymbol{x})\leqslant 0\\ >0 & \text{如果 } r_i(\boldsymbol{x})>0\end{cases} \tag{6-108}$$

式（6-107）右边的求和项是惩罚项，只要有一个不等式约束被违反，惩罚项就会增加能量函数的值。QP问题的能量函数的第二种形式可以写成：

$$E_2(\boldsymbol{x},K) = \boldsymbol{c}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + K\sum_{i=1}^{m}\max\{r_i(\boldsymbol{x}),0\} \tag{6-109}$$

262
~
263

可以看出，这两种形式都是通过把约束附加到QP问题的目标函数中来产生能量函数。参数$K>0$通常称为惩罚参数，在两种情况下都需要选一个相当大的惩罚参数来确保最小化过程满足所有约束。通过采用最速下降方法，可以计算能量函数的梯度，并且在离散时间步内确立学习。

式（6-107）对$\boldsymbol{x}$的梯度给定如下

$$\frac{\partial E_1(\boldsymbol{x},K)}{\partial \boldsymbol{x}} = \boldsymbol{c} + \boldsymbol{Q}\boldsymbol{x} + K\sum_{i=1}^{m}\Psi[r_i(\boldsymbol{x})]\begin{bmatrix}a_{i1}\\ a_{i2}\\ \vdots\\ a_{in}\end{bmatrix} \tag{6-110}$$

注意式（6-105）中的非负约束，现在可以写出更新方程如下：

$$x_j(k+1) = \begin{cases}x_j(k) - \mu_j\left\{c_j + \sum_{i=1}^{n}q_{ji}x_i + K\sum_{i=1}^{m}\Psi[r_i(\boldsymbol{x})]a_{ij}\right\} & \text{如果 } x_j(k+1)\geqslant 0\\ 0 & \text{如果 } x_j(k+1)<0\end{cases} \tag{6-111}$$

其中$j=1,2,\cdots,n$，$\Psi(v)=\mathrm{d}\Phi(v)/\mathrm{d}v$，$\mu_j$是学习率参数。

式（6-109）的能量函数的梯度如下

$$\frac{\partial E_2(\boldsymbol{x},K)}{\partial \boldsymbol{x}} = \boldsymbol{c} + \boldsymbol{Q}\boldsymbol{x} + K\sum_{i=1}^{m}S_i\begin{bmatrix}a_{i1}\\ a_{i2}\\ \vdots\\ a_{in}\end{bmatrix} \tag{6-112}$$

其中

$$S_i = \begin{cases}0 & \text{如果 } r_i(\boldsymbol{x})\leqslant 0\\ 1 & \text{如果 } r_i(\boldsymbol{x})>0\end{cases} \tag{6-113}$$

再次注意式（6-105）中的非负约束，可以看出，更新方程变成

$$x_j(k+1) = \begin{cases}x_j(k) - \mu_j\dfrac{\partial E_2(\boldsymbol{x},K)}{\partial \boldsymbol{x}} & \text{如果 } x_j(k+1)\geqslant 0\\ 0 & \text{如果 } x_j(k+1)<0\end{cases} \tag{6-114}$$

基于两个不同能量函数的神经网络体系结构如图6-10和图6-11。可以看出，这两个网络实际上有相同的结构，仅有的显著区别是第一层中对约束的非线性处理。把这两个神经网络体系结构与图6-8中的网络相比较显示出，用神经网络解决有不等式约束的QP问题的最大好处来自它不需要添加松弛变量或剩余变量就可以解决原始问题的能力。 264

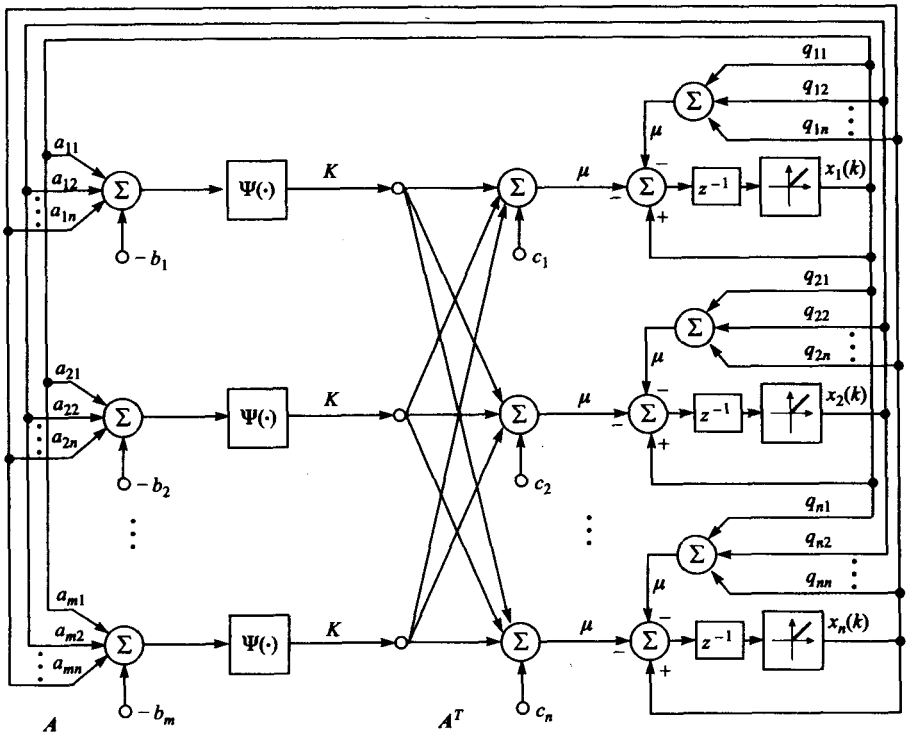



图6-10 解决有不等式约束的QP问题的离散时间神经网络，方程（6-114）的实现

具有混合约束的QP问题的神经网络

前面几节介绍了几种可以用来解决标准形式或所有约束都由不等式形式给出的QP问题的神经网络体系结构。一般来说，QP问题可以用等式约束和不等式约束来表达。如同已经讨论的，解决具有混合约束的QP问题的一种方法是把其转换成标准形式的等价问题。然而，这种方法增加了问题的维数，因此，在很多情况下，这么做在计算上并不高效。另一种选择是直接法，构造一个增广的能量函数，对每个约束违反引入一个惩罚，并且使用某种梯度技术来进行最优化。

具有混合约束的QP问题有如下的一般形式：

最小化

$$f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} \tag{6-115}$$ 265

受限于

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \tag{6-116}$$

$$\vdots$$

$$a_{p1}x_1 + a_{p2}x_2 + \cdots + a_{pn}x_n = b_p \tag{6-117}$$

$$a_{p+1,1}x_1 + a_{p+1,2}x_2 + \cdots + a_{p+1,n}x_n \leqslant b_{p+1} \tag{6-118}$$

$$\vdots \tag{6-119}$$

$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leqslant b_m \tag{6-120}$$

和

$$x_1, x_2, \cdots, x_n \geqslant 0 \tag{6-121}$$

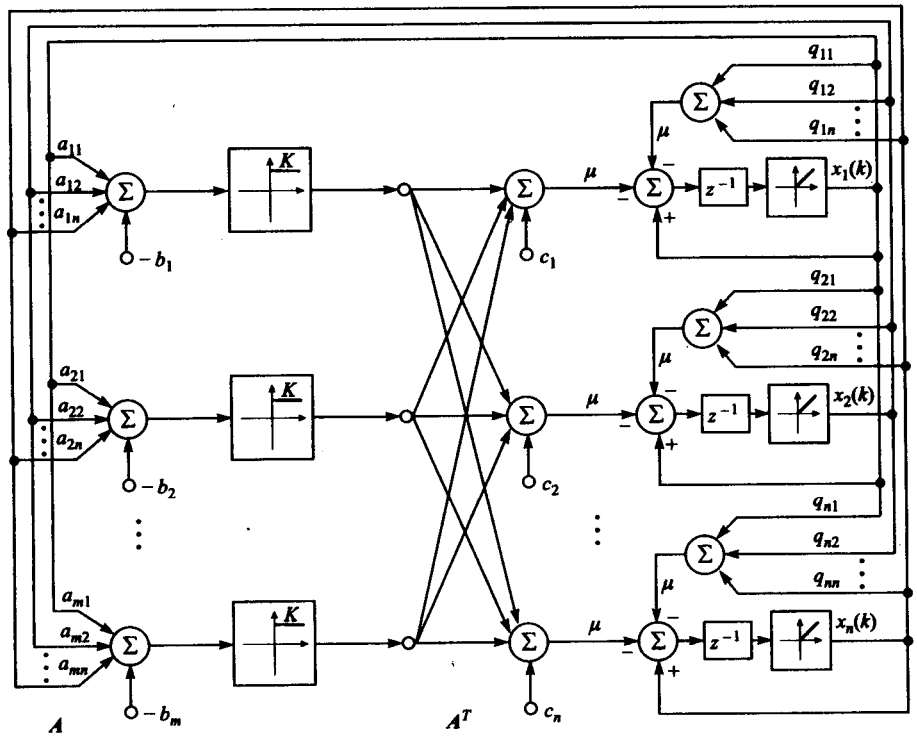



图6-11　解决具有不等式约束的QP问题的离散时间神经网络，方程（6-102）和方程（6-103）的实现

其中$\boldsymbol{c} \in \Re^{n\times 1}$, $\boldsymbol{Q} \in \Re^{n\times n}$是对称半正定矩阵。

266　概念上，式（6-115）～式（6-121）中指定的QP问题可以看作式（6-84）～式（6-86）和式（6-103）～式（6-105）中的QP问题的推广，并且能量函数可以写成式（6-87）、式（6-107）和式（6-109）中定义的能量函数的复合。比如，如果定义

$$\boldsymbol{A}_p = \begin{bmatrix} a_{11} & a_{12} & & a_{1n} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ a_{p1} & a_{p2} & \cdots & a_{pn} \end{bmatrix} \tag{6-122}$$

$$\boldsymbol{b}_p = [b_1 \quad b_2 \quad \cdots \quad b_p]^T \tag{6-123}$$

和

$$\boldsymbol{\lambda}_p = [\lambda_1 \quad \lambda_2 \quad \cdots \quad \lambda_p]^T \tag{6-124}$$

能量函数可以用式（6-87）和式（6-107）表达成

$$\begin{aligned} E(\boldsymbol{x}, \boldsymbol{\lambda}_p, K_1, K_2) = {} & \boldsymbol{c}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{\lambda}_p^T(\boldsymbol{A}_p\boldsymbol{x} - \boldsymbol{b}_p) \\ & + \frac{K_1}{2}(\boldsymbol{A}_p\boldsymbol{x} - \boldsymbol{b}_p)^T(\boldsymbol{A}_p\boldsymbol{x} - \boldsymbol{b}_p) + K_2\sum_{i=p+1}^{m}\Phi[r_i(\boldsymbol{x})] \end{aligned} \tag{6-125}$$

使用最速下降梯度法，更新方程可以写成

$$\boldsymbol{x}(k+1)=\boldsymbol{x}(k)-\mu\frac{\partial E(\boldsymbol{x},K_1,K_2)}{\partial \boldsymbol{x}} \tag{6-126}$$

和

$$\lambda_p(k+1)=\lambda_p(k)+\eta\frac{\partial E(\boldsymbol{x},\lambda_p,K_1,K_2)}{\partial \lambda_p} \tag{6-127}$$

偏导数可计算为

$$\frac{\partial E(\boldsymbol{x},\lambda_p,K_1,K_2)}{\partial \boldsymbol{x}}=\boldsymbol{c}+\boldsymbol{Q}\boldsymbol{x}+\boldsymbol{A}_p^T\lambda_p+K_1\boldsymbol{A}_p^T(\boldsymbol{A}_p\boldsymbol{x}-\boldsymbol{b}_p)\ +K_2\sum_{i=p+1}^{m}\Psi[r_i(\boldsymbol{x})]\begin{bmatrix}a_{i1}\\a_{i2}\\\vdots\\a_{in}\end{bmatrix} \tag{6-128}$$

和

$$\frac{\partial E(\boldsymbol{x},\lambda_p K_1,K_2)}{\partial \lambda_p}=\boldsymbol{A}_p\boldsymbol{x}-\boldsymbol{b}_p \tag{6-129}$$

考虑到式（6-110）中的非负要求，标量形式的更新方程组可以写成 267

$$x_j(k+1)=\begin{cases}x_j(k)-\mu\left\{c_j+\sum_{i=1}^{n}q_{ji}x_i(k)\right.\\\qquad\qquad+\sum_{i=1}^{p}a_{ij}[K_1r_i(x)+\lambda_i(k)]\\\qquad\qquad\left.+K_2\sum_{i=p+1}^{m}\Psi[r_i(\boldsymbol{x})]a_{ij}\right\} & \text{如果 } x_j(k+1)\geqslant 0\\0 & \text{如果 } x_j(k+1)<0\end{cases} \tag{6-130}$$

和

$$\lambda_j(k+1)=\lambda_j(k)+\eta\left(\sum_{i=1}^{n}a_{ji}x_i-b_{pj}\right) \tag{6-131}$$

## 6.4　解决非线性连续约束最优化问题的神经网络

在本章前面几节中，介绍了两种重要形式的约束最优化问题——LP问题和QP问题。现在把注意力回到式（6-1）和式（6-2）中描述的约束最优化问题的一般情形中去。通常，为了强调目标函数和约束都可能是非线性的，这种类型的问题称为非线性规划（nonlinear programming, NP）。

NP问题有大量的实际应用，因此，在理论上和实践上得到了广泛的研究。由于这个原因，容易找到许多极好的参考文献，而且大量的非线性规划算法已经或正在开发。这一节的主要目标是举几个例子来说明神经网络是如何用作一个计算上高效、且相对简单的工具来实现一些著名的非线性规划技术，包括惩罚函数法，障碍函数法，普通拉格朗日乘子法，和增广拉格朗日乘子法。

在专注于NP算法的神经实现之前，回顾一下这个问题的定义。基于约束的形式，可以定义三种不同形式的NP最小化问题[2]：

**NP1（具有等式约束的NP问题）**

最小化

268

$$f(\boldsymbol{x}) = f(x_1, x_2, \cdots, x_n) \tag{6-132}$$

受限于

$$h_i(\boldsymbol{x}) = 0 \quad \text{其中} i = 1, 2, \cdots, m \tag{6-133}$$

**NP2（具有不等式约束的NP问题）**

最小化

$$f(\boldsymbol{x}) = f(x_1, x_2, \cdots, x_n) \tag{6-134}$$

受限于

$$g_i(\boldsymbol{x}) \leqslant 0 \quad \text{其中} i = 1, 2, \cdots, m \tag{6-135}$$

**NP3（具有混合约束的NP问题）**

最小化

$$f(\boldsymbol{x}) = f(x_1, x_2, \cdots, x_n) \tag{6-136}$$

受限于

$$h_i(\boldsymbol{x}) = 0 \quad \text{其中} i = 1, 2, \cdots, p \tag{6-137}$$

和

$$g_i(\boldsymbol{x}) \leqslant 0 \quad \text{其中} i = p+1, p+2, \cdots, m \tag{6-138}$$

其中$\boldsymbol{x} \in \Re^{n\times 1}$是独立变量向量，$f(\boldsymbol{x}):\Re^{n\times 1}\to\Re$是目标函数，函数$h_i(\boldsymbol{x})$, $g_i(\boldsymbol{x}):\Re^{n\times 1}\to\Re$代表约束。为了简化算法的推导，假设目标函数和约束都是独立变量的光滑可微函数。

使用额外变量，问题NP2和NP3中的不等式约束可以转化成等式约束。相似地，每个等式约束可以根据

$$h_i(\boldsymbol{x}) = 0 \Leftrightarrow h_i(\boldsymbol{x}) \leqslant 0 \quad \text{且} \quad h_i(\boldsymbol{x}) \geqslant 0 \tag{6-139}$$

转化成一对不等式约束。因此，每个NP问题可以转换成NP1或NP2形式。然而，从计算的角度更倾向于用NP问题的原始形式考虑问题。最后，注意问题的NP1或NP2形式可以看作是更一般的NP3形式的特殊情况。

### 6.4.1 罚函数NP方法的神经网络

使用罚函数的方法尝试把NP问题转换成一个等价的无约束最优化问题，或转换成一系列的约束最优化问题。这个转换是通过修正目标函数，使得目标函数包含一些项来惩罚每个对约束的违反实现的。一般来说，修正的目标函数有如下的形式：

269

$$f_A(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{p} K_i^{(1)} \Phi_i^{(1)}[h_i(\boldsymbol{x})] + \sum_{i=p+1}^{m} K_i^{(2)} \Phi_i^{(2)}[g_i(\boldsymbol{x})] \tag{6-140}$$

函数$\Phi_i^{(1)}, \Phi_i^{(2)}$称作罚函数，当独立变量向量违反某个约束时，或者说当它在可行域外时，罚函数会增加修正的目标函数$f_A(\boldsymbol{x})$的值。通常至少选择满足下列条件的一阶可微函数作为罚函数：

1. 对于等式约束

$$\begin{matrix} \Phi_i^{(1)} > 0 & \text{对于 } h_i(\boldsymbol{x}) \neq 0 \\ = 0 & \text{对于 } h_i(\boldsymbol{x}) = 0 \end{matrix} \tag{6-141}$$

2. 对于不等式约束

$$\begin{matrix} \Phi_i^{(2)} > 0 & \text{对于 } g_i(\boldsymbol{x}) > 0 \\ = 0 & \text{对于 } g_i(\boldsymbol{x}) \leqslant 0 \end{matrix} \tag{6-142}$$

下面给出了一些通常用作罚函数的函数：对于等式约束的情形

1. $\Phi_i^{(1)}(\upsilon)=\frac{1}{2}\upsilon^2$ (6-143)

2. $\Phi_i^{(1)}(\upsilon)=\frac{1}{\rho}|\upsilon|^{\rho}\quad \rho>0$ (6-144)

3. $\Phi_i^{(1)}(\upsilon)=\cosh\upsilon-1$ (6-145)

4. $\Phi_i^{(1)}(\upsilon)=\ln\left|\frac{1}{2}(e^{\upsilon}+e^{-\upsilon})\right|$ (6-146)

对于不等式约束的情形

1. $\Phi_i^{(2)}(\upsilon)=\max\{0,\upsilon\}$ (6-147)

2. $\Phi_i^{(2)}(\upsilon)=(\max\{0,\upsilon\})^2$ (6-148)

比如，NP3问题的典型修正目标函数可以写成

$$f_A(\boldsymbol{x})=f(\boldsymbol{x})+\sum_{i=1}^{p}\frac{K_i^{(1)}}{\rho_1}|h_i(\boldsymbol{x})|^{\rho_1}+\sum_{i=p+1}^{m}K_i^{(2)}\max\{0,g_i(\boldsymbol{x})\}^{\rho_2} \tag{6-149}$$

其中$\rho_1$, $\rho_2\geqslant 0$。参数$K_i^{(1)},K_i^{(2)}\geqslant 0$通常称为罚参数或罚乘数，式（6-149）中假设每个罚函数都有一个相关的独立的罚参数。在实践中却很少这样，通常整个惩罚项只有一个参数乘数，即，

$$\begin{aligned}f_A(\boldsymbol{x})&=f(\boldsymbol{x})+K\left[\sum_{i=1}^{p}\frac{1}{\rho_1}|h_i(\boldsymbol{x})|^{\rho_1}+\sum_{i=p+1}^{m}\max\{0,g_i(\boldsymbol{x})\}^{\rho_2}\right]\\&=f(\boldsymbol{x})+KP(\boldsymbol{x})\end{aligned} \tag{6-150}$$

其中$P(\boldsymbol{x})$表示惩罚项。 270

在罚函数的实际应用中，有两个需要注意的基本问题。首先，必须意识到式（6-150）仅仅表示一个从式（6-136）～式（6-138）的原始问题近似。这个近似的接近程度如何？第二个问题是设计一个能够以适时方式成功解决无约束问题且在计算上高效的神经网络算法。

由式（6-150）中增广目标函数的形式，很明显，解都在一个罚函数$P(\boldsymbol{x})$的值很小的区域内。事实上，如果$K$向无穷大增长，无约束问题的解将限制到原始NP问题的可行域中。记住，如果一个点在可行域中，则它满足所有的约束且罚函数等于零。在极限的情况下，当$K\to\infty$，这两个问题变成等价的。简单地说，这两个问题的等效性可以总结成如下的定理。

**定理6.4** 考虑一个NP问题如下：

最小化

$$f(x)=f(x_1,x_2,\cdots,x_n) \tag{6-151}$$

受限于

$$\boldsymbol{x}\in S,\quad S\subset\Re^{n\times 1} \tag{6-152}$$

其中$S$是一个由一系列等式或不等式定义的约束集（即可行域）。定义无约束最优化问题序列如下：

最小化

$$q(K_j,\boldsymbol{x})=f(\boldsymbol{x})+K_jP(\boldsymbol{x}) \tag{6-153}$$

其中$P$是一个罚函数，满足

$$P(x)\geqslant 0 \quad 对于\ x\in \Re^{n\times 1} \tag{6-154}$$

$$P(x)=0 \quad 当且仅当\ x\in S \tag{6-155}$$

$K_j, j=1,2,\cdots$，是一个实数序列，满足

$$K_j>0 \qquad \forall j \tag{6-156}$$

$$K_{j+1}>K_j \qquad \forall j \tag{6-157}$$

$$K_j\to\infty \text{as} j\to\infty \text{以任意方式} \tag{6-158}$$

令$\{\tilde{x}_j\}, j=1,2,\cdots$，是式（6-153）给出的无约束最优化问题的解序列。序列$\{\tilde{x}_j\}$的极限点就是式（6-151）和式（6-152）中的NP问题的解。

为了简洁，省略定理6.4的证明。有兴趣的读者可以在Luenberger[4]中找到更多的细节。定理6.4本质上概括了罚函数法的方法论。可以看出，由式（6-153）～式（6-158）产生了无
271 约束最优化问题序列，而其序列的解收敛到原始NP问题的解。从神经网络的观点看，求解这个无约束最优化问题的序列明显是不可接受的。因此，惩罚方法的实际实现通常采用如下两种方式：

1. 罚参数$K$是时变的，它随着网络的训练过程而增加。

2. 罚参数$K$要选择一个足够大的正数，确保无约束问题是对原NP问题的一个接近的近似。

一旦修正目标函数确定，则任何一种梯度技术都可用来执行最小化任务。为了简化，将演示最速下降法的使用。然而，共轭梯度法、牛顿法和拟牛顿法都可以提供显著加快的收敛速率[4]，虽然代价是增加了计算复杂性。

应用最速下降方法，可以依照下式产生更新方程

$$x(k+1)=x(k)-\mu\frac{\partial f_A(x)}{\partial x} \tag{6-159}$$

其中$\mu>0$是学习率参数，式（6-159）右边的梯度项依赖于罚函数的选择。例如，当能量函数的形式如式（6-150），设$\rho_1=2$和$\rho_2=1$，则

$$\frac{\partial f_A(x)}{\partial x}=\frac{\partial f(x)}{\partial x}+K\left[\sum_{i=1}^{p}\frac{\partial h_i(x)}{\partial x}h_i(x)+\sum_{i=p+1}^{m}\frac{\partial}{\partial x}\max\{0,g_i(x)\}\right] \tag{6-160}$$

把式（6-160）代入式（6-159），有如下的学习规则

$$x(k+1)=x(k)-\mu\left[\frac{\partial f(x)}{\partial x}+K\sum_{i=1}^{p}\frac{\partial h_i(x)}{\partial x}h_i(x)+K\sum_{i=p+1}^{m}\frac{\partial}{\partial x}\max\{0,g_i(x)\}\right] \tag{6-161}$$

这个过程的神经网络体系结构实现如图6-11所示。注意，仅呈现了神经网络的一部分，它是用来计算由独立变量向量的一个分量。同样注意，这个网络符合NP3问题的一般情况。即它适应等式和不等式约束。由图6-12中的网络，通过除去负责不等式约束（NP1的情况）或等式约束（NP2的情况）的部分，可以导出NP1和NP2问题的适当网络。

**例6.4**　考虑如下的NP问题：

最小化

272

$$f(x)=\exp[(x_1-1.5)^2+x_2^2] \tag{6-162}$$

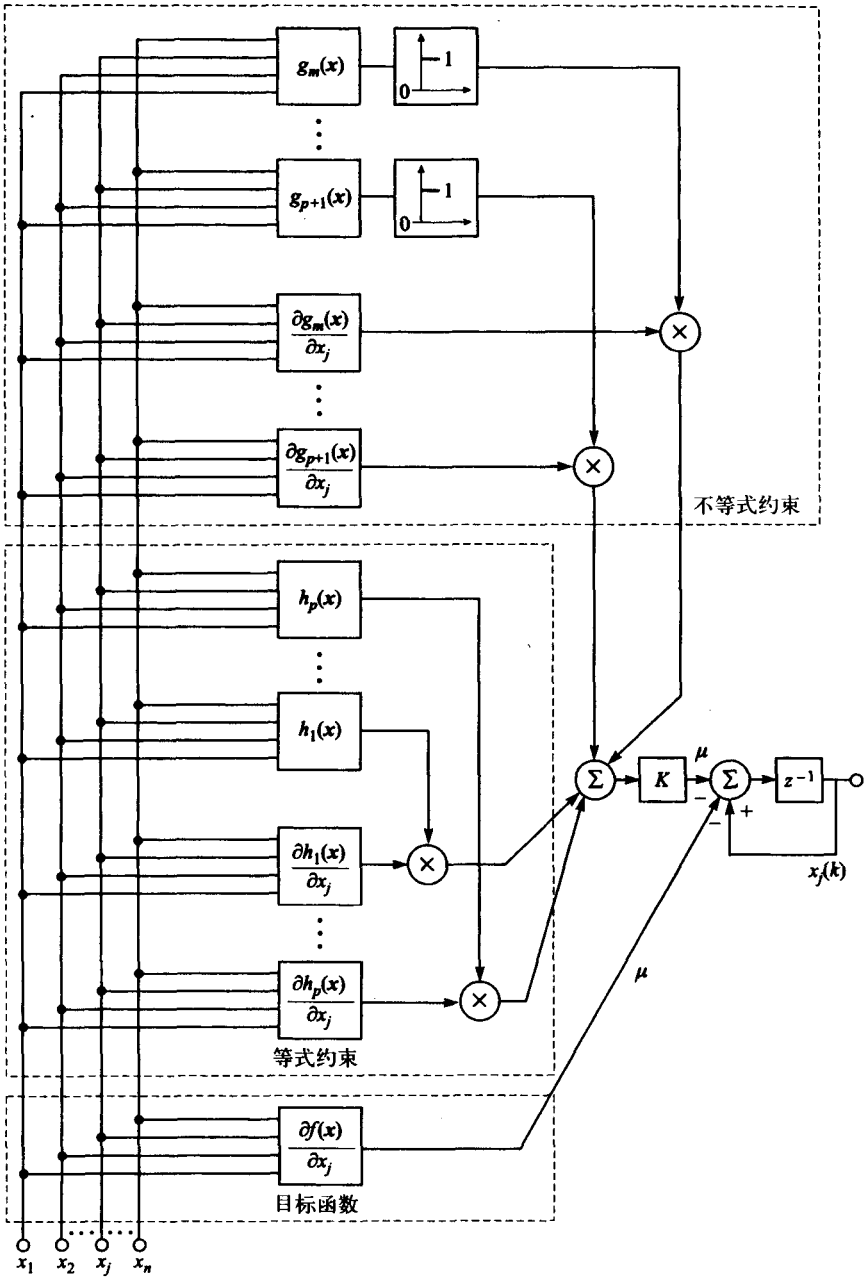



图6-12 NP3问题离散时间网络的罚函数法的实现，等式（6-161）的实现

273

受限于

$$x_1^2 + x_2^2 - 1 \leqslant 0 \tag{6-163}$$

上述问题明显是一个具有不等式约束的NP问题（即NP2）。修正的罚函数可以写成：

$$f_A(x) = \exp[(x_1 - 1.5)^2 + x_2^2] + \frac{K}{2}\max\{0, x_1^2 + x_2^2 - 1\} \tag{6-164}$$

通过使用最速下降法，更新方程可以写成：

$$x_1(k+1)=x_1(k)-\mu\{2(x_1-1.5)\exp[(x_1-1.5)^2+x_2^2]+K[\mathrm{sgn}(x_1^2+x_2^2-1)+1]x_1\} \tag{6-165}$$

和

$$x_2(k+1)=x_2(k)-\mu\{2x_2\exp[(x_1-1.5)^2+x_2^2]+K[\mathrm{sgn}(x_1^2+x_2^2-1)+1]x_2\} \tag{6-166}$$

其中$\mathrm{sgn}(v)=v/|v|$是符号函数。图6-12所示的神经网络体系结构常常用来确定NP问题的解。网络的参数通常选作$K=5$, $\mu=0.01$，设初始解为$\boldsymbol{x}=[0\ \ 1.5]^T$。这个网络在大约1900步迭代内收敛，解的轨迹如图6-13所示。

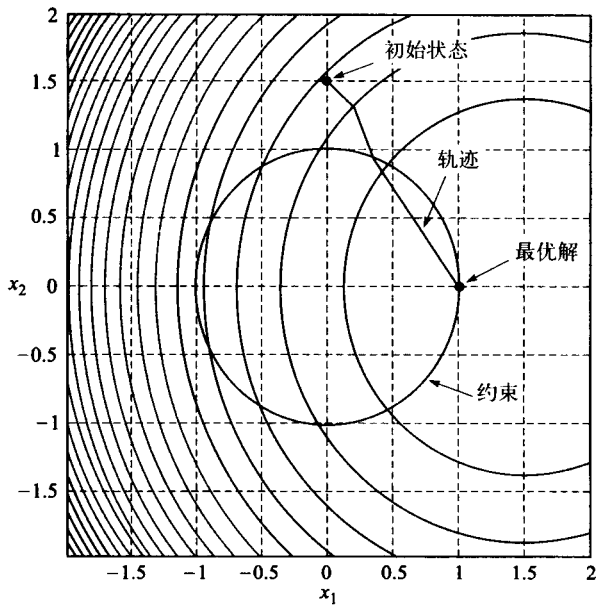



图6-13　例6.4中NP问题解的轨迹

274

使用罚函数在计算上的困难

如同早先讨论的，只要罚参数的值足够大，无约束最优化问题式（6-150）的解可以任意接近原始NP问题的解。然而，如果对$K$选择一个非常大的值，罚函数的黑塞矩阵可能呈现病态。当NP问题的解在约束条件构成的超曲面上时（参见例6.4），一个很大的罚常数可能会导致算法在边界的振荡。如果检测到这种现象，算法必须把学习率参数$\mu$减到一个很小的值以获得收敛

由罚函数法推导出的更新方程基于某类梯度学习法，比如，最速下降法，共轭梯度法，牛顿法等等。因此，它们都受到一个对所有梯度法都存在的共同问题的困扰——局部最小值。为了防止算法的搜索落到修正罚函数$f_A(\boldsymbol{x})$的局部最小值，可以使用随机退火法。可以推出基于这种方法的更新方程如下

$$\boldsymbol{x}(k+1)=\boldsymbol{x}(k)-\mu\frac{\partial f_A(\boldsymbol{x})}{\partial \boldsymbol{x}}+\sigma^2\boldsymbol{n} \tag{6-167}$$

其中$\boldsymbol{n}\in\Re^{n\times 1}$是随机产生的具有零均值和单位方差的白噪声向量，即

$$E\{\boldsymbol{n}^T\boldsymbol{n}\}=1 \tag{6-168}$$

$\sigma^2$是在网络训练过程中向零递减的参数。大的$\sigma^2$值可以使算法脱离局部最小值，而小的$\sigma^2$值“微调”以达到最优解。

### 6.4.2 障碍函数NP方法的神经网络

与罚函数法相似，障碍法用于把一个约束NP问题转换成一个等价的无约束问题或一系列无约束问题。为了完成转换，用可以防止解偏离可行域的障碍函数来增广NP问题的原始代价函数。为了阐明这一点，考虑式（6-134）和式（6-135）给出的NP2类型的非线性规划问题。障碍函数法如下转换问题：

最小化

$$f_A(\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{1}{K}\mathrm{B}(\boldsymbol{x}) \tag{6-169}$$

其中$K>0$，是用来控制障碍函数的参数，$\mathrm{B}(\boldsymbol{x})$是任何有如下性质的函数： 275

1. $\boldsymbol{x}\in S$时$B(\boldsymbol{x})=0$，其中$S\subset\Re^{n\times 1}$是由式（6-135）中不等式条件确定的可行域。

2. 当向量$\boldsymbol{x}$接近可行域的边界时$B(\boldsymbol{x})\to+\infty$。

理想状态下，障碍函数应该在可行域内接近边界的地方趋向无穷大，在可行域内的其他任何地方都为零。实际应用中，最常用的障碍函数如下：

1. $$B(\boldsymbol{x}) = \sum_{i=1}^{m}\frac{1}{g_i(\boldsymbol{x})} \tag{6-170}$$

2. $$B(\boldsymbol{x}) = -\sum_{i=1}^{m}\ln[g_i(\boldsymbol{x})] \tag{6-171}$$

由于式（6-170）和式（6-171）中的障碍函数在可行域内为非零值，所以参数$K$应该取一个相当大的值来最小化障碍项在除了边界以外的其他地方时对NP代价函数的影响。然而，$K$取大的值会导致问题呈现病态，通常是在训练过程中加大$K$值。注意式（6-170）和式（6-171）的两个障碍函数在可行域的边界上都是不连续的。这点给它们在NP问题中的使用加上了两个严格的限制。第一，因为函数在边界上不连续，不能用于有等式约束的NP问题。第二，对于许多有不等式约束的NP问题，解正好就在边界上（参见例6.4）。障碍函数的一些缺点可以通过应用Cichocki和Unbehauen[2]与Nash和Sofer[9]中介绍的混合惩罚法来去除。

与罚函数法不同，障碍函数不允许试验点$\boldsymbol{x}$脱离可行域。这对于最优化可能在到达最优点$\tilde{\boldsymbol{x}}$之前就中止（比如代价函数低于某个预先确定的值时）的NP问题，可能是有用的。在构造出修正惩罚函数后，任何一种无约束最优化方法都可以用来确定式（6-169）的最优解。与罚函数法相似，可以证明式（6-169）的解在$K\to\infty$[7]时收敛到原始NP2问题的解。

### 6.4.3 普通拉格朗日乘子NP方法的神经网络

与罚函数法和障碍函数法相似，拉格朗日乘子法通过把约束合并到一个修正目标函数中来处理约束。在拉格朗日乘子法的神经网络实现过程中，首先考虑NP1形式的NP问题（即，有等式约束的NP问题）。同样的方法可以扩展到有不等式约束的NP问题。 276

NP1问题的拉格朗日乘子

NP问题拉格朗日乘子方法的应用（参见A.6.2节）要求把NP问题转换成一个无约束最优化问题。这个无约束问题是通过把约束乘以一个作为比例因子的拉格朗日乘子加到目标函数中形成的。新的目标函数称为拉格朗日算子，其形式如下：

$$L(\boldsymbol{x},\lambda) = f(\boldsymbol{x}) + \sum_{i=1}^{m}\lambda_i h_i(\boldsymbol{x}) \tag{6-172}$$

其中$\boldsymbol{x} \in \Re^{n\times 1}$, $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \cdots, \lambda_m]^T \in \Re^{m\times 1}$。

式（6-172）中的无约束问题总共有$n + m$个未知数，其维数比原始约束问题要高，即最优化操作发生在$\Re^{n+m}$空间中。在最小点，无约束最优化问题的目标函数必须满足稳定条件。对式（6-172）中的拉格朗日算子，稳定条件可以写作

$$\frac{\partial L(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \boldsymbol{x}} = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} + \sum_{i=1}^{m} \lambda_i \frac{\partial h_i(\boldsymbol{x})}{\partial \boldsymbol{x}} = \mathbf{0} \tag{6-173}$$

和

$$\frac{\partial L(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} = [h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \cdots, h_m(\boldsymbol{x})]^T = \mathbf{0} \tag{6-174}$$

其中$\mathbf{0}$表示恰当维数的零（或空）向量。等式（6-173）和式（6-174）形成了由$n + m$个未知数构成的$n + m$个方程的系统，需要求解独立变量的最优值和拉格朗日乘数，即$(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{\lambda}})$。注意式（6-174）确保最优解满足所有的约束，换句话说，在最优点$\tilde{\boldsymbol{x}}$

$$L(\tilde{\boldsymbol{x}}, \boldsymbol{\lambda}) = f(\tilde{\boldsymbol{x}}) \tag{6-175}$$

神经网络方法通过采用某种基于梯度的技术的迭代方式来解决式（6-173）和式（6-174）的方程组系统。对于最速下降法，有如下两个学习规则[2]

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) - \mu_x \left[ \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} + \sum_{i=1}^{m} \lambda_i \frac{\partial h_i(\boldsymbol{x})}{\partial \boldsymbol{x}} \right] \tag{6-176}$$

和

277

$$\boldsymbol{\lambda}(k+1) = \boldsymbol{\lambda}(k) + \mu_\lambda [h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \cdots, h_m(\boldsymbol{x})]^T \tag{6-177}$$

标量形式如下

$$x_j(k+1) = x_j(k) - \mu_x \left[ \frac{\partial f(\boldsymbol{x}(k))}{\partial x_j} + \sum_{i=1}^{m} \lambda_i(k) \frac{\partial h_i(\boldsymbol{x}(k))}{\partial x_j} \right] \tag{6-178}$$

和

$$\lambda_j(k+1) = \lambda_j(k) + \mu_\lambda h_j(\boldsymbol{x}(k)) \tag{6-179}$$

其中$\mu_x, \mu_\lambda > 0$是学习率参数。

拉格朗日乘子法的实际实现中有几个问题。在6.3节已经看到，如果目标函数不是凸的，就会有多个局部极小点，并且最小化过程很容易陷入其中某点。在使用拉格朗日乘子方法的情况下，即使原始代价函数$f(\boldsymbol{x})$是凸的，也丝毫不能确保式（6-172）中拉格朗日算子也是凸的。此外，在一些情况下，式（6-178）和式（6-179）中的迭代方法可能会在一个局部极小点附近振荡。为了防止振荡，强制算法收敛，可以在式（6-177）中附加一个阻尼因子

$$\boldsymbol{\lambda}(k+1) = \boldsymbol{\lambda}(k) + \mu_\lambda \left[ \frac{\partial L(\boldsymbol{x}(k), \boldsymbol{\lambda}(k))}{\partial \boldsymbol{\lambda}} - \alpha \boldsymbol{\lambda}(k) \right] \tag{6-180}$$

其中$0 \leqslant \alpha \leqslant 1$，称为阻尼参数。

这个过程的神经网络体系结构如图6-14所示。

NP2问题的拉格朗日乘子方法

在NP2问题中，约束是以不等式的形式给出的。通过加上适当的额外变量，式（6-135）中的不等式可以转换成等式，可以使用上一节中介绍的方法。这个方法通过下面的例子来示范。

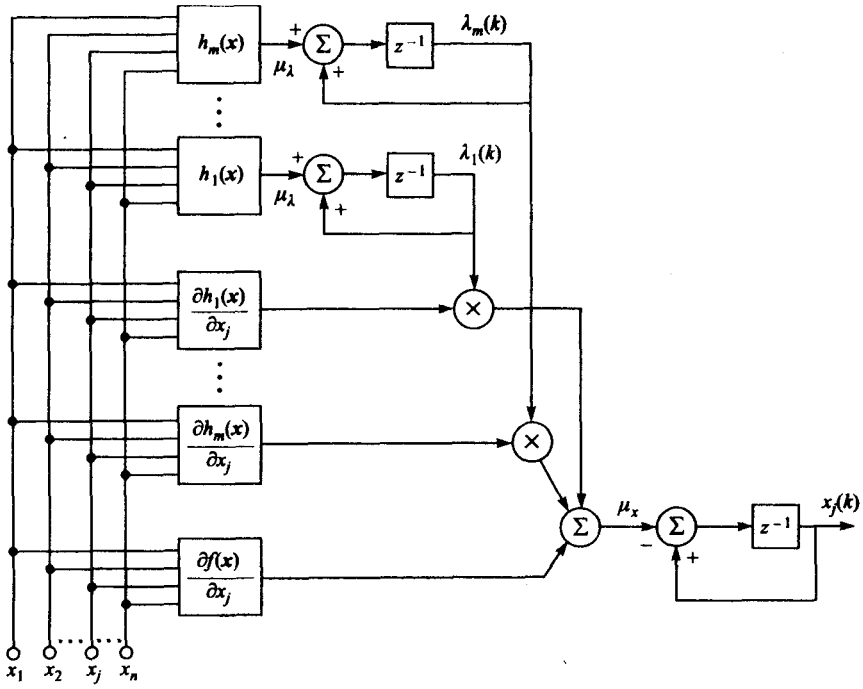



图6-14 拉格朗日乘子神经网络的离散时间实现。方程（6-178）和方程（6-179）表示这个学习规则

**例6.5** 考虑如下的NP2问题：

最小化

$$f(\boldsymbol{x}) = x_1^2 - 1.4x_1 + x_2^2 \tag{6-181}$$

受限于

$$x_1^2 + x_2^2 \leqslant 1 \tag{6-182}$$

通过加一个额外变量$\theta^2$，原始问题可以转换如下：

最小化

$$f(\boldsymbol{x}) = x_1^2 - 1.4x_1 + x_2^2 \tag{6-183}$$

受限于

$$x_1^2 + x_2^2 + \theta^2 = 1 \tag{6-184}$$

或

$$x_1^2 + x_2^2 + \theta^2 - 1 = 0 \tag{6-185}$$

278

上述问题的拉格朗日公式可以表示为

$$L(\boldsymbol{x}, \lambda, \theta) = x_1^2 - 1.4x_1 + x_2^2 + \lambda(x_1^2 + x_2^2 + \theta^2 - 1) \tag{6-186}$$

应用稳定条件，有

$$\frac{\partial L(\boldsymbol{x},\lambda,\theta)}{\partial x_1} = 2(1+\lambda)x_1 - 1.4 = 0 \tag{6-187}$$

$$\frac{\partial L(\boldsymbol{x},\lambda,\theta)}{\partial x_2} = 2(1+\lambda)x_2 = 0 \tag{6-188}$$

$$\frac{\partial L(\boldsymbol{x},\lambda,\theta)}{\partial \lambda} = x_1^2 + x_2^2 + \theta^2 - 1 = 0 \tag{6-189}$$

$$\frac{\partial L(\boldsymbol{x},\lambda,\theta)}{\partial \theta} = 2\lambda\theta = 0 \tag{6-190}$$

由式（6-190），或者$\lambda = 0$，或者$\theta = 0$；或者$\lambda = 0$和$\theta = 0$。把$\lambda = 0$代入式（6-187）～式（6-189）中，得到解$x_1 = 0.7, x_2 = 0, \theta^2 = 0.57$。可以求出代价函数的值为$f(x_1, x_2) = -0.49$。如果把$\theta = 0$代入式（6-187）～式（6-189），系统的解就是$x_1 = 1, x_2 = 0, \lambda = -0.3$。在第2个解的点，代价函数的值为$f(x_1, x_2) = -0.4$。最后，$\lambda = 0$，$\theta = 0$代入方程无解。比较代价函数在两个稳定点
279 的值，看到，在$x_1 = 0.7, x_2 = 0$达到最小化。而第2个解是拉格朗日算子的局部最小点。

一般情况下，对于NP2问题，拉格朗日函数如下

$$L(\boldsymbol{x},\lambda,\theta) = f(\boldsymbol{x}) + \sum_{j=1}^{m} \lambda_j [g_j(\boldsymbol{x}) + \theta_j^2] \tag{6-191}$$

应用稳定条件，得到

$$\frac{\partial L(\boldsymbol{x},\lambda,\theta)}{\partial x_i} = \frac{\partial f(\boldsymbol{x})}{\partial x_i} + \sum_{j=1}^{m} \lambda_j \frac{\partial g_j(\boldsymbol{x})}{\partial x_i} = 0 \quad i = 1,2,\cdots,n \tag{6-192}$$

$$\frac{\partial L(\boldsymbol{x},\lambda,\theta)}{\partial \lambda_i} = g_i(\boldsymbol{x}) - \theta_i^2 = 0 \quad i = 1,2,\cdots,m \tag{6-193}$$

$$\frac{\partial L(\boldsymbol{x},\lambda,\theta)}{\partial \theta_i} = 2\lambda_i\theta_i = 0 \quad i = 1,2,\cdots,m \tag{6-194}$$

从式（6-194），可以看到三种不同情况：或者$\lambda_i = 0$，或者$\theta_i = 0$，或者两者全部为零。

1. 如果$\lambda_i = 0$，$\theta_i \neq 0$，那么约束$g_i(\boldsymbol{x}) \leqslant 0$是冗余的，可以忽略。换句话说，约束$g_i(\boldsymbol{x}) \leqslant 0$的存在并不改变最优化的结果。

2. 如果$\lambda_i \neq 0$，$\theta_i = 0$，则最优解在可行域的边界上，或在曲线$g_i(\boldsymbol{x}) = 0$上。

3. 如果$\lambda_i = 0$，$\theta_i = 0$，则由$g_i(\boldsymbol{x}) = 0$给出的可行域的边界通过代价函数的全局极小点。

神经网络方法是用一种梯度技术和迭代的方式求解式（6-192）～式（6-194）中的方程组系统。与这种方法相关的几个问题已经在前一节中概述了。

从不等式约束到等式约束的转换增加了非线性规划问题的维数。对于每一个转换得到的等式，引入一个附加的变量。尽管这在小规模的情况下不成问题，但在有大量不等式的情况下，会显著地增加计算负担。因为这个原因，在Golub和Van Loan[2]和Press等[13]中，常规的拉格朗日乘子法根据如下的等式扩展到不等式约束

$$x_j(k+1) = x_j(k) - \mu\left[\frac{\partial f(\boldsymbol{x})}{\partial x_j} + \sum_{i=1}^{m} \lambda_i \frac{\partial g_i(\boldsymbol{x})}{\partial x_j}\right] \quad j = 1,2,\cdots,n \tag{6-195}$$

和

$$\lambda_i(k+1) = \max\{0, \lambda_i(k) + \eta g_i(\boldsymbol{x})\} \quad i = 1,2,\cdots,m \tag{6-196}$$

其中$\eta > 0$表示学习率参数。

280 这个过程的神经网络体系结构实现如图6-15所示。从图中可以看出，它与图6-14中的体系结构非常类似。唯一的区别是求取式（6-196）中max函数的非线性元素。

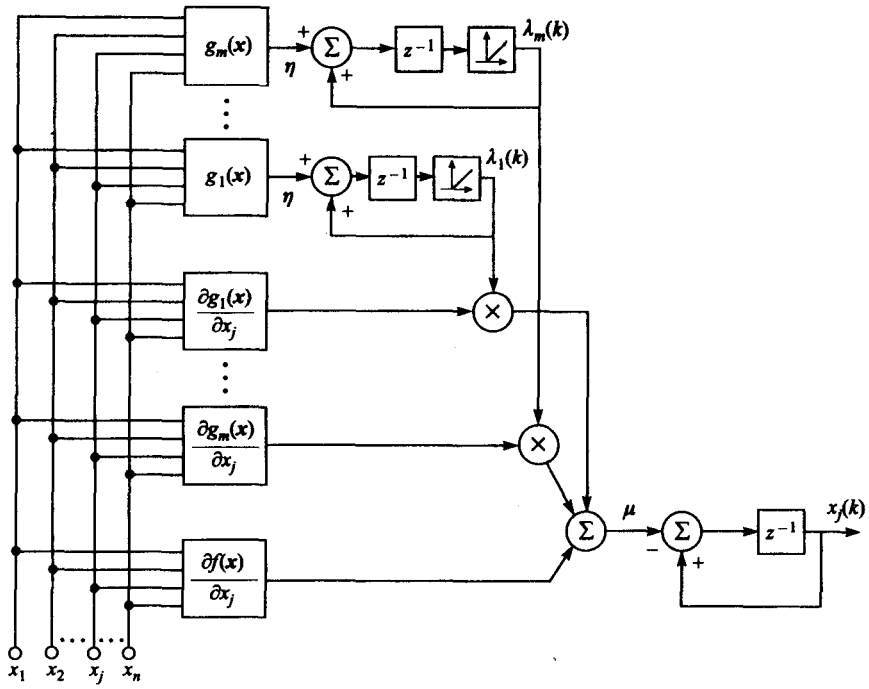



图6-15 具有不等式约束的NP问题的拉格朗日乘子神经网络的离散时间实现。方程（6-195）和方程（6-196）给出了学习规则

### 6.4.4 增广拉格朗日乘子方法的神经网络

增广拉格朗日乘子方法是解决NP问题的最有效的一般性方法之一。它由Hestens[14]和Powell[15]分别独立地提出。Gill等人[17]提供了关于对这些类型的最优化问题进行广泛研究的其他参考。增广的拉格朗日乘子法对式（6-132）～式（6-138）中的所有三种类型的NP问题都是可行的。简单起见，首先介绍它在具有等式约束的NP问题中的使用，再讨论它对具有不等式或混合约束的NP问题的扩展。

*NP1问题的增广拉格朗日方法*

根据前一节，与式（6-132）和式（6-133）中定义的约束NP问题相关的拉格朗日函数如下 281

$$L(\boldsymbol{x},\boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i h_i(\boldsymbol{x}) \tag{6-197}$$

增广拉格朗日算子是由式（6-197）增加了额外惩罚项得到的。增广拉格朗日算子的最常用形式在[17, 18]中给出

$$L_A(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{k}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i h_i(\boldsymbol{x}) + \sum_{i=1}^{m} k_i h_i^2(\boldsymbol{x}) \tag{6-198}$$

其中$\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \cdots, \lambda_m]^T \in \Re^{m\times 1}$表示拉格朗日乘子向量，$\boldsymbol{k} = [k_1, k_2, \cdots, k_m]^T \in \Re^{m\times 1}$是正惩罚参数向量。

可以看出式（6-198）增加了二次惩罚项，也就增加了拉格朗日函数的黑塞矩阵的正定性[16, 17]。此外，如果$\boldsymbol{k}$中的系数足够大，就可以强制拉格朗日函数的黑塞矩阵的所有特征值都大于零，从神经网络实现的角度看，这是很重要的，因为是用迭代的方式搜索解。已经提到

过（参见定理6.2），如果函数在某个集合上是凸的，则可以确保基于梯度的搜索方法收敛到局部极小点。黑塞矩阵正定的事实确保存在解的一个领域使得函数是凸的，由此，如果最优化过程的起始点选择得当，算法一定收敛到全局极小点。在有了增广拉格朗日算子后，最优解即是无约束极小点。这可以用任何一种无约束最优化技术实现。最简单地，可以使用最速下降法，拉格朗日算子的最小化转换成一个差分方程形式的系统

$$\boldsymbol{x}(k+1)=\boldsymbol{x}(k)-\mu_x\frac{\partial L(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{k})}{\partial \boldsymbol{x}} \tag{6-199}$$

和

$$\boldsymbol{\lambda}(k+1)=\boldsymbol{\lambda}(k)+\mu_\lambda\frac{\partial L(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{k})}{\partial \boldsymbol{\lambda}} \tag{6-200}$$

求出式（6-199）和式（6-200）中梯度的值，产生如下的更新方程

$$x_j(k+1)=x_j(k)-\mu_x\left\{\frac{\partial f(\boldsymbol{x}(k))}{\partial x_j}+\sum_{i=1}^{m}[\lambda_i(k)+2k_ih_i(\boldsymbol{x}(k))]\frac{\partial h_i(\boldsymbol{x}(k))}{\partial x_j}\right\} \tag{6-201}$$

和

$$\lambda_j(k+1)=\lambda_j(k)+\mu_\lambda h_j(\boldsymbol{x}(k)) \tag{6-202}$$

282 其中$\mu_x$, $\mu_\lambda>0$表示学习率参数。

式（6-201）和式（6-202）的神经网络实现如图6-16所示。这个网络由两个独立模块组成。第一个模块执行解的更新，第二个模块更新拉格朗日乘子。早期的增广拉格朗日方法建议对解$\boldsymbol{x}$和拉格朗日乘子使用不同更新率[15, 17]。而且，在这些早期版本中，在每次更新拉格朗日乘子之前，都要对$\boldsymbol{x}$执行一次完整的无约束最小化。这个过程证明是非常低效的，因为解的精确性依赖于拉格朗日乘子估计的精确性，在拉格朗日乘子收敛到最优值$\boldsymbol{\lambda}=\tilde{\boldsymbol{\lambda}}$之前，算法无法收敛到最优值$\boldsymbol{x}=\tilde{\boldsymbol{x}}$。基于这些观察，一些不同的更新策略被提出[2, 15-18]。式（6-201）和式（6-202）中提出的算法是一种极端情况，解的估计和拉格朗日乘子的估计在每一步迭代中都被计算。本质上，这个算法在每个迭代步都提出了不同的最优化问题，并且与称作基于QP的投影拉格朗日方法[17]联系紧密。

增广拉格朗日乘子方法有几个重要的性质应当考虑[17]：

1. *局部最小点性质*。与惩罚函数法相似，增广拉格朗日乘子法确保收敛到增广拉格朗日算子的局部最小点。增广拉格朗日算子的局部最小点只在惩罚参数$\boldsymbol{k}$足够大的情况下才收敛到目标函数的约束最小点。

2. *惩罚参数的选择*。一般来说，需要选择使增广拉格朗日算子的黑塞矩阵正定的惩罚参数。如果惩罚参数的值太小，算法将不能收敛，或是虽然收敛到增广拉格朗日算子的局部最小点，但却不能最小化目标函数。另一方面，如果参数选择得过大，算法会在解的附近表现出振荡行为。

3. *拉格朗日乘子的收敛性*。对于式（6-201）和式（6-202）中的算法，要找到最优解必须使$\boldsymbol{x}$和$\boldsymbol{\lambda}$都收敛到最优值$\tilde{\boldsymbol{x}}$和$\tilde{\boldsymbol{\lambda}}$。在某些情况下，增广拉格朗日算子对乘子的值很敏感，并且，在获得收敛前需要相当数量的迭代步骤[17]。

*NP2问题的增广拉格朗日方法*

前一节的增广拉格朗日乘子法可以扩展到具有不等式约束的NP问题。为了实现这点，增广拉格朗日算子必须改成

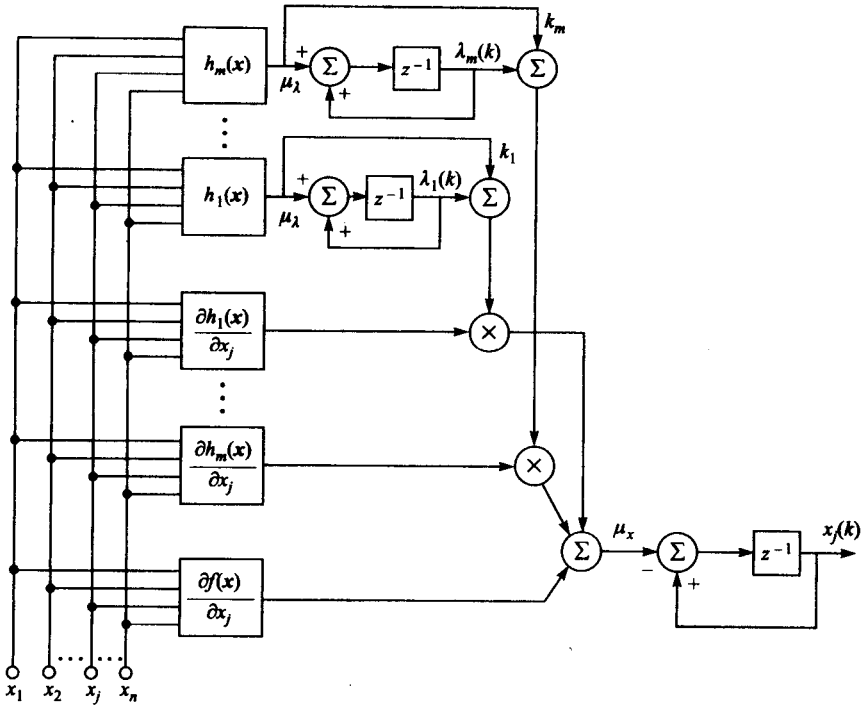



图6-16　增广拉格朗日乘子法的离散时间神经网络实现。学习规则由式（6-201）和式（6-202）给出

$$L(\boldsymbol{x},\lambda)=f(\boldsymbol{x})+\sum_{i=1}^{m}\lambda_i\max\{0,g_i(\boldsymbol{x})\}+\sum_{i=1}^{m}\frac{K_i}{2}\max\{0,g_i(\boldsymbol{x})\}^2 \tag{6-203}$$

其中$\lambda_i$, $i=1,2,\cdots,m$是拉格朗日乘子，$K_i$, $i=1,2,\cdots,m$是惩罚参数。如同在式（6-203）中看到的，任何对约束的违反都会增加拉格朗日算子的值。即只认为被违反的约束是有效的。式（6-203）可以写成一种更紧凑的形式

$$L(\boldsymbol{x},\lambda)=f(\boldsymbol{x})+\sum_{i=1}^{m}S_i\left[\lambda_i g_i(\boldsymbol{x})+\frac{K_i}{2}g_i(\boldsymbol{x})^2\right] \tag{6-204}$$

其中

$$S_i=\begin{cases}0 & \text{如果 } g_i(\boldsymbol{x})\leqslant 0\\ 1 & \text{如果 } g_i(\boldsymbol{x})>0\end{cases} \tag{6-205}$$

为了推导出相应的神经网络，更新方程可以根据下式获得

$$\boldsymbol{x}(k+1)=\boldsymbol{x}(k)-\mu_x\frac{\partial L(\boldsymbol{x},\lambda)}{\partial \boldsymbol{x}} \tag{6-206}$$

和

$$\lambda(k+1)=\lambda(k)+\mu_\lambda\frac{\partial L(\boldsymbol{x},\lambda)}{\partial \lambda} \tag{6-207}$$

283
~
284

把式（6-204）的适当梯度代入式（6-206）和式（6-207）后，得到

$$x_j(k+1)=x_j(k)-\mu_x\left\{\frac{\partial f(\boldsymbol{x}(k))}{\partial x_j}+\sum_{i=1}^{m}S_i[\lambda_i+K_i g_i(\boldsymbol{x}(k))]\frac{\partial g_i(\boldsymbol{x}(k))}{\partial x_j}\right\} \tag{6-208}$$

和

$$\lambda_j(k+1) = \lambda_j(k) + \mu_\lambda S_j g_j(\boldsymbol{x}(k)) \tag{6-209}$$

这个过程的神经网络结构实现如图6-17所示。

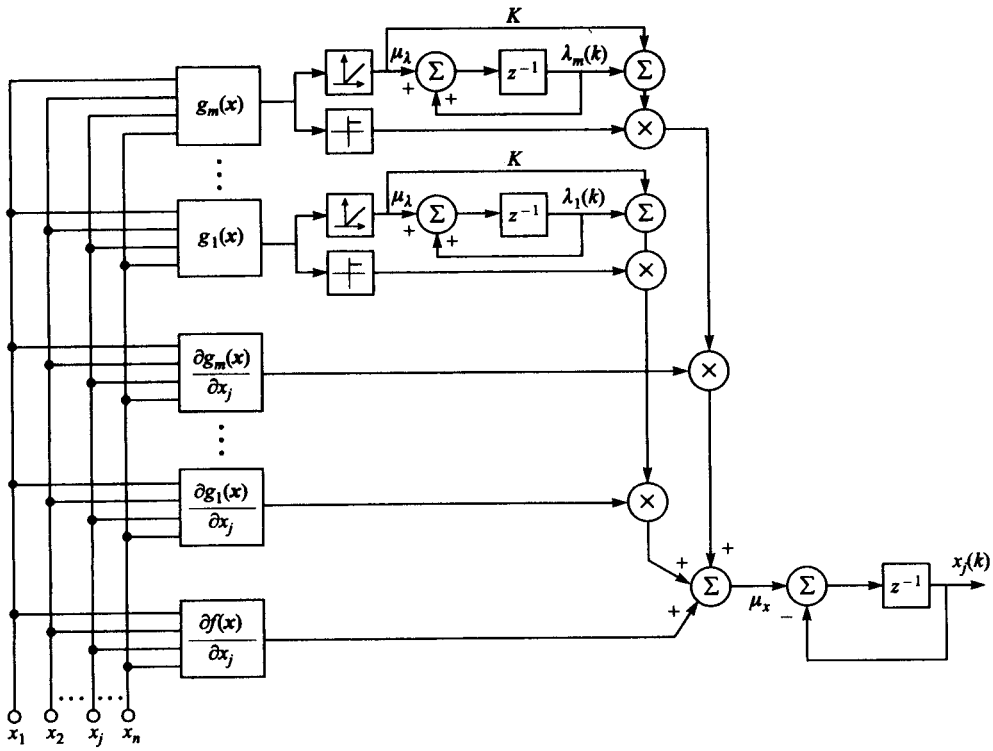



图6-17　用增广拉格朗日乘子法解决有不等式约束的NP问题的神经网络的离散时间实现。方程（6-208）和方程（6-209）表示两个学习规则

**例6.6**　用增广拉格朗日乘子法考虑例6.4中的问题。对于式（6-133）和式（6-134）中的问题，增广拉格朗日算子写作

285

$$L(\boldsymbol{x},\lambda) = \exp[(x_1-1.5)^2 + x_2^2] + \lambda \max\{0, x_1^2 + x_2^2 - 1\} + \frac{K}{2}\max\{0, x_1^2 + x_2^2 - 1\}^2 \tag{6-210}$$

相应的导数计算如下

$$\begin{aligned}\frac{\partial L(\boldsymbol{x},\lambda)}{\partial x_1} &= 2(x_1-1.5)\exp[(x_1-1.5)^2 + x_2^2] \\ &\quad +2x_1[\operatorname{sgn}(x_1^2 + x_2^2 - 1) + 1][\lambda + K(x_1^2 + x_2^2 - 1)]\end{aligned} \tag{6-211}$$

$$\begin{aligned}\frac{\partial L(\boldsymbol{x},\lambda)}{\partial x_2} &= 2x_2\exp[(x_1-1.5)^2 + x_2^2] \\ &\quad +2x_2[\operatorname{sgn}(x_1^2 + x_2^2 - 1) + 1][\lambda + K(x_1^2 + x_2^2 - 1)]\end{aligned} \tag{6-212}$$

和

$$\frac{\partial L(\boldsymbol{x},\lambda)}{\partial \lambda} = [\operatorname{sgn}(x_1^2 + x_2^2 - 1) + 1](x_1^2 + x_2^2 - 1) \tag{6-213}$$

图6-17中的网络可以用来执行最优化任务。网络的参数可以设置成例6.4中的值，即$K$ =5, $\mu$ = 0.01。初始条件选择$x_1(0)$ = 0, $x_2(0)$ = 0和$\lambda(0)$ = 1。在大约700步迭代后网络收敛到解$\tilde{x}$ = [1.008 8, 0.000 2]$^T$。

## 习题

6.1 写一个MATLAB程序，实现如图6-1所示神经网络方法。用求解如下LP问题测试程序：

(a) 最大化 $f(\boldsymbol{x}) = 2x_1 + 4x_2$

受限于 $x_1 - x_2 \leqslant 1$

$3x_1 + 2x_2 \leqslant 12$

$2x_1 + 3x_2 \leqslant 3$

$-2x_1 + 3x_2 \leqslant 9$

$x_1, x_2 \geqslant 0$

(b) 最大化 $f(\boldsymbol{x}) = x_1 + 2x_2$

受限于 $-x_1 + x_2 \leqslant 1$

$-x_1 + x_2 \geqslant -1$

$x_1 + x_2 \leqslant 4$

$x_1, x_2 \geqslant 0$

(c) 最小化 $f(\boldsymbol{x}) = -5x_1 - x_2$

受限于 $x_1 - x_2 \leqslant 2$

$x_1 + 2x_2 \leqslant 8$

$x_1, x_2 \geqslant 0$

(d) 最大化 $f(\boldsymbol{x}) = 10x_1 + 19x_2 + 9x_3$

受限于 $2x_1 + 3x_2 + 2x_3 \leqslant 10$

$x_1, x_2, x_3 \geqslant 0$

对于每个问题：

- 将问题转换成标准形式。
- 使用神经网络方法解决最优化问题。
- 用作图的方法来解决问题，并且把结果与用神经网络获得的结果进行比较。

6.2 写一个MATLAB程序实现图6-3所示神经网络方法。

(a) 使用神经网络方法解决问题6.1。

(b) 作图解决问题，并验证神经网络获得的结果。

(c) 从复杂性和收敛速度的角度来比较图6-1和图6-3中网络的性能。

6.3 重复问题6.2，使用图6-4给出的神经网络。

6.4 考虑具有不等式约束的LP问题，定义如下

$$\text{最小化 } f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x}$$
$$\text{受限于 } \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}$$

它的对偶问题，公式为

$$\text{最大化 } g(\boldsymbol{\lambda}) = \boldsymbol{b}^T\boldsymbol{\lambda}$$
$$\text{受限于 } \boldsymbol{A}^T\boldsymbol{\lambda} = \boldsymbol{c} \quad \boldsymbol{\lambda} \geqslant 0$$

其中$\boldsymbol{A} \in \Re^{m\times n}$, $\boldsymbol{x}, \boldsymbol{c} \in \Re^{n\times 1}$, $\boldsymbol{b}, \boldsymbol{\lambda} \in \Re^{m\times 1}$。考虑对偶定理（参见6.1节定理6.1），能量函数

定义为

$$E(\boldsymbol{x},\lambda)=\frac{1}{2}(\boldsymbol{c}^T\boldsymbol{x}-\boldsymbol{b}^T\lambda)^2+\frac{K_1}{2}(\boldsymbol{A}^T\lambda-\boldsymbol{c})^T(\boldsymbol{A}^T\lambda-\boldsymbol{c})+\frac{K_2}{2}\|\Phi[\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}]\|_2^2$$

其中$\upsilon\in\Re^{n\times 1}$

$$\Phi(\upsilon)=\begin{bmatrix}\max\{0,\upsilon_1\}\\ \max\{0,\upsilon_2\}\\ \vdots\\ \max\{0,\upsilon_n\}\end{bmatrix}$$

对偶定理可以用来推导一个迭代过程，同时解决原来的LP问题和对偶LP问题。

(a) 用梯度方法和上面定义的能量函数，推导$\boldsymbol{x}$和$\boldsymbol{\lambda}$的更新方程。注意$\boldsymbol{x}$应该依照最速下降来更新，而$\boldsymbol{\lambda}$则应依照最速上升来更新。

(b) 设计一个神经网络来实现部分（a）的方程。

(c) 写一个MATLAB程序来模仿部分（b）中的神经网络。用如下的LP问题来测试代码

$$\boldsymbol{b}=[-3\ \ -16\ \ -20]^T\quad \boldsymbol{A}=\begin{bmatrix}2 & 1\\ -1 & -3\\ -4 & -1\end{bmatrix}\quad \boldsymbol{c}=[-1\ \ -1]^T$$

6.5　使用牛顿梯度方法（参见A.5.3节）需要计算黑塞矩阵的逆。考虑式（6-20）中具有不等式约束的LP问题定义的能量函数。

(a) 证明式（6-20）中函数的黑塞矩阵可以这样计算

$$\boldsymbol{H}=\nabla_x^2E_1(\boldsymbol{x})=\boldsymbol{K}\boldsymbol{A}^T\boldsymbol{A}$$

287

(b) 证明式（6-20）中函数的黑塞矩阵必须是奇异的。

(c) Levenberg-Marquardt算法（参见3.4.4节）可以用来提供快速收敛，即使黑塞矩阵是奇异的。这个方法的更新方程如下

$$\boldsymbol{x}(k)=\boldsymbol{x}(k)-(\mu\boldsymbol{I}+\boldsymbol{H})^{-1}\boldsymbol{g}$$

其中$E(\boldsymbol{x})$是要最小化的能量函数，$\boldsymbol{H}=\nabla_x^2E(\boldsymbol{x})$，$\boldsymbol{g}=\nabla_xE$，$\boldsymbol{I}$是相应维数的单位矩阵，并且$\mu>0$。

- 推导式（6-20）的能量函数的Levenberg-Marquardt更新方程。
- 写一个MATLAB程序来实现Levenberg-Marquardt方法。
- 用问题6.1的LP问题测试你程序的性能。

6.6　方程（6-61）给出了有混合约束的LP问题的能量函数，此函数用式（6-21）和式（6-40）构成一种可能组合。

(a) 用式（6-20）和式（6-48）构造一个另能量函数。

(b) 基于部分（a）的能量函数和最速下降方法，推导更新方程组。

(c) 设计一个神经网络实现部分（b）中的方程组。

(d) 写一个MATLAB程序实现部分（c）中的方程组，并且用问题6.1中的LP问题测试你的程序。

6.7　式（6-90）中有等式约束的QP问题的更新方程组是由最速下降法推导得到的。证明用如下方法可以得到另外一组方程

(a) 牛顿法

$$\boldsymbol{x}(k+1)=\boldsymbol{x}(k)-(\boldsymbol{Q}+k\boldsymbol{A}^T\boldsymbol{A})^{-1}\{\boldsymbol{c}+\boldsymbol{Q}\boldsymbol{x}(k)+\boldsymbol{A}^T\boldsymbol{\lambda}(k)+\boldsymbol{K}\boldsymbol{A}^T[\boldsymbol{A}\boldsymbol{x}(k)-\boldsymbol{b}]\}$$

(b) Levenberg-Marquardt法

$$\boldsymbol{x}(k+1)=\boldsymbol{x}(k)-(\mu\boldsymbol{I}+\boldsymbol{Q}+k\boldsymbol{A}^T\boldsymbol{A})^{-1}\{c+\boldsymbol{Q}\boldsymbol{x}(k)+\boldsymbol{A}^T\boldsymbol{\lambda}(k)+K\boldsymbol{A}^T[\boldsymbol{A}\boldsymbol{x}(k)-b]\}$$

其中$\mu>0$。

6.8　为了演示惩罚函数法的使用，考虑如下NP问题：

$$\begin{aligned}&\text{最小化}\quad f(\boldsymbol{x})=-x_1x_2\\&\text{受限于}\quad h(\boldsymbol{x})=3x_1+x_2-7=0\end{aligned}$$

其精确解是$x_1=7/6$和$x_2=7/2$。

(a) 以如下形式构造能量函数

$$f_A(\boldsymbol{x})=f(\boldsymbol{x})+Kh(\boldsymbol{x})^2\quad K>0$$

并且证明无约束最小点在下面点处得到

$$x_1=\frac{14K}{12K-1},\quad x_2=\frac{42K}{12K-1}$$

288

(b) 求能量函数的黑塞矩阵，并且确定它的条件数（是$K$的函数）。条件数定义成

$$\rho=\frac{\lambda_{\max}}{\lambda_{\min}}$$

其中$\lambda_{max}$和$\lambda_{min}$分别是矩阵最大和最小的特征值。

(c) 从解精度的观点和梯度搜索过程稳定性的角度来讨论参数$K$的影响。

(d) 写一个MATLAB程序来实现图6-12中的神经网络，并用它来解决上述的NP问题。用不同的$K$值做试验。

6.9　考虑一个NP问题，定义如下

$$\begin{aligned}&\text{最小化}\quad f(\boldsymbol{x})=e^{x_1}-x_1x_2+x_2^2\\&\text{受限于}\quad x_1^2+x_2^2=4\\&\qquad\quad 2x_1+x_2\leqslant 2\end{aligned}$$

这是一个NP3形式的问题。即，既有等式约束又有不等式约束。处理这个问题的一种方法是使用混合惩罚障碍法形成一个增广代价函数。例如，混合惩罚障碍函数可以写作

$$f_A(\boldsymbol{x})=f(\boldsymbol{x})+K\sum_{i=1}^{p}|h_i(\boldsymbol{x})|^{\rho_1}+\frac{1}{K}\sum_{i=p+1}^{m}\frac{1}{|g_i(\boldsymbol{x})|^{\rho_2}}$$

其中$h_i(\boldsymbol{x})$是等式约束，$g_i(\boldsymbol{x})$是不等式约束，$\rho_1,\rho_2$是正数。

(a) 给出上述NP问题的混合惩罚障碍增广代价函数。

(b) 使用最速下降梯度法和部分（a）中的代价函数导出更新方程组。

(c) 写一个MATLAB程序实现部分（b）中给出的方程组，用这个程序解上述NP问题。确保迭代过程的初始点落在与不等式约束相关联的可行域中。

(d) 用不同的参数$K$值做试验，讨论结果作为$K$的函数的收敛性和精确性。设计一个更新方案在迭代过程中自动增加$K$，相应地修改部分（c）中的MATLAB代码。

6.10　考虑下述NP问题：

$$\begin{aligned}&\text{最小化}\quad f(\boldsymbol{x})=x_1^2+x_2^2\\&\text{受限于}\quad x_1+x_2=1\\&\qquad\quad x_1+x_2=2\end{aligned}$$

显然，这个问题的约束有冲突的设置并且无法解决。然而，如果应用惩罚方法，可以

289

如下构造一个增广代价函数

$$f_A(\boldsymbol{x}) = x_1^2 + x_2^2 + K[(x_1 + x_2 - 1)^2 + (x_1 + x_2 - 2)^2]$$

(a) 忽略这个问题没有解的事实，推导增广代价函数的无约束最优化问题的更新方程组。

(b) 写MATLAB代码实现部分（a）中的方程组并且执行无约束最优化。

(c) 解释结果。

6.11　考虑如下具有不等式约束的QP问题：

$$\text{最小化}\quad f(x, y) = x^2 - xy + y^2 - 3x$$
$$\text{受限于}\quad x + y \leqslant 4 \quad x, y \geqslant 0$$

(a) 用向量矩阵形式重新描述问题。

(b) 写一个MATLAB程序实现图6-8中的神经网络。把上述QP问题转化为标准形式，并且使用你的MATLAB程序解决问题。

(c) 写一个MATLAB程序实现图6-10中的神经网络。通过解决上述的QP问题来测试你的程序。

(d) 使用图6-11中所示的神经网络方法来重复部分（c）。

6.12　使用增广拉格朗日乘子法来解决下述NP问题

$$\begin{aligned}\text{最小化}\quad & f(\boldsymbol{x}) = x_1^3 + x_2^2\\ \text{受限于}\quad & x_1^2 + x_2^2 - 10 = 0\\ & 1 - x_1 \leqslant 0\\ & 1 - x_2 \leqslant 0\end{aligned}$$

## 参考文献


1. S. I. Gass, *Linear Programming, Methods and Applications*, 5th ed., New York: McGraw-Hill, 1984.
2. A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, Chichester, England: Wiley, 1993.
3. D. J. Wilde and C. S. Beightler, *Foundation of Optimization*, Englewood Cliffs, NJ: Prentice-Hall, 1967.
4. D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd ed., Reading, MA: Addison-Wesley, 1989.
5. D. M. Himmelblau, *Applied Nonlinear Programming*, New York: McGraw-Hill, 1972.
6. W. I. Zangwill, *Nonlinear Programming, A Unified Approach*, Englewood Cliffs, NJ: Prentice-Hall, 1969.
7. M. S. Bazaraa and C. M. Shetty, *Nonlinear Programming, Theory and Applications*, New York: Wiley, 1979.
8. N. K. Kwak and M. C. Schniederjans, *Mathematical Programming*, Malabar, FL: Krieger, 1987.

290

9. S. G. Nash and A. Sofer, *Linear and Nonlinear Programming*, New York: McGraw-Hill, 1996.
10. A. Ravindran, D. T. Philips, and J. J. Solberg, *Operations Research, Principles and Practice*, New York: Wiley, 1987.
11. D. P. Gaver and G. L. Thompson, *Programming and Probability Models in Operations Research*, Belmont, CA: Wadsworth, 1973.
12. J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, New York: Academic Press, 1970.

13. K. J. Arrow, L. Hurwicz, and H. Uzawa, *Studies in Linear and Nonlinear Programming*, Stanford, CA: Stanford University Press, 1958.
14. M. R. Hestens, "Multiplier Gradient Methods," *Journal of Optimization Theory and Applications*, vol. 4, 1969, pp. 303–20.
15. M. J. D. Powell, "A Method for Nonlinear Constraints in Minimization Problems," In *Optimization*, ed. R. Fletcher, London: Academic, 1969, pp. 283–98.
16. R. Fletcher, *Practical Methods of Optimization*, Chichester, England: Wiley, 1987.
17. P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, London: Academic, 1981.

# 第7章 用神经网络解决矩阵代数问题

## 7.1 概述

本章将介绍矩阵代数中的几个重要概念。目的不仅是为了拓宽读者关于矩阵代数概念的知识，而且是为了采用结构神经网络的神经计算方法的方式来表达问题。就是说，以一种能够利用结构神经网络解决问题的方式来表达特定的问题，这个神经计算方法有可能设计出高效且健壮的算法，特别是服务于实时（或联机）应用的算法。神经网络的并行计算特性能够以相对直接的方式实现用于这类问题的大规模并行学习算法。即将给出的数学概念并不是新的，但是，在一些情形中给出的神经计算算法是比较独特的，并有实际的应用。而且，本章采用了统一的方法导出学习规则。

结构神经网络首先由Wang和Mendel[1]提出，是为特殊的矩阵代数应用量身定做的神经体系结构。例如，确定矩阵的逆可以有很多种方法[2]。但是，如果需要重复计算矩阵的逆，应该使用更有效的方法，利用算法所具有的并行结构。本章将给出解决矩阵代数问题的多种方法，比如，矩阵求逆，LU分解，QR分解，舒尔（Schur）分解，对称特征值问题，奇异值分
292 解，求解代数矩阵李雅普诺夫方程和求解代数矩阵里卡蒂（Riccati）方程。

将介绍的算法与类神经元自适应信号处理系统有关。所有学习算法都基于第3章所讨论的误差反向传播算法。此外，绝大部分本章介绍的用于解决矩阵代数问题的神经网络都由线性处理单元组成。有观点认为线性网络只能处理线性函数，并且一个多层线性网络总是可以通过调整权值[3]变换为只有一层线性处理单元的体系结构，从这种观点看，线性神经元是没有什么意义的。然而，在本章会看到线性多层神经网络对许多计算任务非常有用。因此，在本章，将采用线性神经网络（结构神经网络）来解决相对较大类的矩阵代数问题。

本章中用神经网络来解决特定矩阵代数问题的基本方法包含四个阶段。这些方法与参考文献[1, 4, 5, 6]中的方法相似，具体如下：

1. 第一阶段是为待解决的特定类型的问题构造合适的误差代价函数。该误差代价函数基于定义的误差变量，这些误差变量一般通过能解决这个特定的问题的函数网络来表达。这样，问题基本上是由结构多层神经网络来表达。

2. 第二阶段是最优化阶段，使用第一阶段定义的误差代价函数为这个结构神经网络推导一个合适的学习规则。一般都采用批量形式（或向量矩阵形式）来推导学习规则。一旦推导出学习规则的向量矩阵形式，就能相对直接地表达成标量形式。

3. 第三阶段是用第二阶段设计的学习规则来训练神经网络，使之符合所期望的模式，即输入/输出信号配对。因此，实质上网络是使相关联的误差代价函数取最小值的最优化过程。就是说，训练阶段是根据推导的学习规则来调整网络的突触权值，以最小化相关的误差代价函数。对于本章介绍的很多学习规则，都可以利用式（2-36）的学习率参数的搜索然后收敛调度策略来提高收敛速度。

4. 第四阶段也就是最后一个阶段实际上是应用阶段。结构神经网络可以针对某个特定的输入集合产生合适的输出信号，从而解决特定的问题。

## 7.2 矩阵的逆和伪逆

计算矩阵的逆可能是线性代数中最重要的一个问题[2]。已有很多种方法用来计算矩阵的逆或伪逆。然而，对于某些情况一些实时应用（比如：自适应信号处理，机器人技术和自动控制）所必需的在线计算，这些方法却常常不具有可操作性。因此，目标是设计能够实时实现的计算矩阵逆（或伪逆）的方法。实现这个目标的一个途径是使用类神经元处理器，应用神经计算技术来求矩阵逆。 293

方法1：矩阵求逆

第一种方法可能是最直接的途径，假设待求逆的矩阵为$\boldsymbol{A} \in \Re^{n\times n}$[6]。显然矩阵$\boldsymbol{A}$是方阵，还假设$\boldsymbol{A}$非奇异。稍后会发现其实$\boldsymbol{A}$是方阵且非奇异的假设实际并不需要。将计算矩阵$\boldsymbol{A}$的逆$\boldsymbol{C} = \boldsymbol{A}^{-1}$，可以写为：

$$\boldsymbol{AC} = \boldsymbol{CA} = \boldsymbol{I} \tag{7-1}$$

$\boldsymbol{I} \in \Re^{n\times n}$是$n \times n$的单位矩阵。如同许多能用来计算$\boldsymbol{A}$的逆的神经计算方法一样，将加进一个确定的误差代价函数（或能量函数）来控制算法（或学习规则）去训练神经网络是必需的。方法1使用线性最小二乘法，其中的误差代价函数定义为：

$$\mathscr{E}(\boldsymbol{C}) = \frac{1}{2}\text{trace}(\boldsymbol{EE}^T) \tag{7-2}$$

由式（7-1）推得误差矩阵$\boldsymbol{E} \in \Re^{n\times n}$为：

$$\boldsymbol{E} = \boldsymbol{AC} - \boldsymbol{I} \tag{7-3}$$

注意，由式（7-1），$\boldsymbol{E}$也可以定义为$\boldsymbol{E} = \boldsymbol{CA} - \boldsymbol{I}$。然而，尽管这将导致学习规则变化，但是结果（即$\boldsymbol{A}^{-1}$）却是相同的。

一个简单的基于最速下降法的连续时间学习规则可以推导出如下的矩阵微分方程组：

$$\frac{\mathrm{d}\boldsymbol{C}(t)}{\mathrm{d}t} = -\mu\nabla_{\boldsymbol{C}}\mathscr{E}(\boldsymbol{C}) = -\mu\frac{\partial\mathscr{E}(\boldsymbol{C})}{\partial\boldsymbol{C}} \tag{7-4}$$

其中$\mu > 0$是学习率参数，必须选取足够小的$\mu$来确定保收敛到逆$\boldsymbol{C}$。式（7-4）的离散时间形式可以写成如下的差分方程组：

$$\boldsymbol{C}(k+1) = \boldsymbol{C}(k) - \mu\nabla_{\boldsymbol{C}}\mathscr{E}(\boldsymbol{C}) \tag{7-5}$$

$k$是离散时间索引，即$k = 0, 1, 2, \cdots$。因此，式（7-2）中误差代价函数的梯度必须计算。如下： 294

$$\begin{aligned}\frac{\partial\mathscr{E}(\boldsymbol{C})}{\partial\boldsymbol{C}} &= \frac{\partial}{\partial\boldsymbol{C}}\left[\frac{1}{2}\text{trace}(\boldsymbol{EE}^T)\right] = \frac{\partial}{\partial\boldsymbol{C}}\left\{\frac{1}{2}\text{trace}[(\boldsymbol{AC}-\boldsymbol{I})(\boldsymbol{C}^T\boldsymbol{A}^T-\boldsymbol{I})]\right\}\\ &= \frac{\partial}{\partial\boldsymbol{C}}\left[\frac{1}{2}\text{trace}(\boldsymbol{ACC}^T\boldsymbol{A}^T - \boldsymbol{AC} - \boldsymbol{C}^T\boldsymbol{A}^T + \boldsymbol{I})\right]\\ &= \boldsymbol{A}^T\boldsymbol{AC} - \boldsymbol{A}^T = \boldsymbol{A}^T\underbrace{(\boldsymbol{AC}-\boldsymbol{I})}_{\boldsymbol{E}} = \boldsymbol{A}^T\boldsymbol{E}\end{aligned} \tag{7-6}$$

上面的推导中用到了两个标量对矩阵求导的一般结果，即：

$$\frac{\partial}{\partial\boldsymbol{A}}\text{trace}(\boldsymbol{BAC}) = \boldsymbol{B}^T\boldsymbol{C}^T \tag{7-7}$$

和

$$\frac{\partial}{\partial \boldsymbol{A}}\text{trace}(\boldsymbol{B}\boldsymbol{A}^T\boldsymbol{C})=\boldsymbol{C}\boldsymbol{B} \tag{7-8}$$

以及合适的链式规则（chain rule，参见A.3.4.2节）。因此，使用式（7-6）的结果，式（7-5）的离散时间学习规则如下：

$$\boldsymbol{C}(k+1)=\boldsymbol{C}(k)+\mu\boldsymbol{A}^T[\boldsymbol{I}-\boldsymbol{A}\boldsymbol{C}(k)] \tag{7-9}$$

如果用误差矩阵$\boldsymbol{E}$来表达，则上述学习规则的另一种形式如下：

$$\boldsymbol{C}(k+1)=\boldsymbol{C}(k)-\mu\boldsymbol{A}^T\boldsymbol{E}(k) \tag{7-10}$$

同样，如果误差矩阵$\boldsymbol{E}$定义成$\boldsymbol{E}=\boldsymbol{C}\boldsymbol{A}-\boldsymbol{I}$，则相应的学习规则如下：

$$\boldsymbol{C}(k+1)=\boldsymbol{C}(k)+\mu[\boldsymbol{I}-\boldsymbol{C}(k)\boldsymbol{A}]\boldsymbol{A}^T \tag{7-11}$$

或用误差矩阵$\boldsymbol{E}$来表达，则学习规则如下：

$$\boldsymbol{C}(k+1)=\boldsymbol{C}(k)-\mu\boldsymbol{E}(k)\boldsymbol{A}^T \tag{7-12}$$

如前所述，式（7-9）和式（7-11），或式（7-10）和式（7-12），将产生相同的结果。

基于式（7-9）或式（7-11）所表示的学习规则可以马上得到两个结论：（1）计算$\boldsymbol{A}$的逆不需要除法运算，只要乘法和加法（或减法）。因此，无论矩阵$\boldsymbol{A}$是否奇异，迭代过程都会产生一个解。事实上，$\boldsymbol{A}$可以推广到$\boldsymbol{A}\in\Re^{m\times n}$，当$m=n$时，是一个完全确定系统。对于一个由代数方程组$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$, $\boldsymbol{x}\in\Re^{n\times 1}$, $\boldsymbol{b}\in\Re^{m\times 1}$表示的系统，当$m>n$时，是一个超定组（overdetermined system）；当$m<n$时，是一个欠定方程组（underdetermined system）。此外，式（7-9）和式（7-11），或式（7-10）和式（7-12）都可以用来推广计算矩阵$\boldsymbol{A}\in\Re^{m\times n}$的伪逆。对此稍后再详细解释。（2）从式（7-10）或式（7-12）可以看出，用来训练神经网络的学习规则是基于误
295 差反向传播的。这在大多数用以解决矩阵代数问题的结构神经网络中是具有典型性的。

当计算矩阵$\boldsymbol{A}\in\Re^{m\times n}$的伪逆时，定义误差矩阵成：

$$\boldsymbol{E}=\boldsymbol{A}\boldsymbol{C}\boldsymbol{A}-\boldsymbol{A} \tag{7-13}$$

其中$\boldsymbol{E}\in\Re^{m\times n}$, $\boldsymbol{C}=\boldsymbol{A}^+\in\Re^{n\times m}$，并且$\boldsymbol{A}^+\equiv\boldsymbol{A}$的伪逆。$\boldsymbol{A}$的伪逆（或Moore-Penrose广义逆）（参见A.2.7节）可以写作：

$$\boldsymbol{C}=\boldsymbol{A}^+=(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T,\quad \text{for } m>n \tag{7-14}$$

或

$$\boldsymbol{C}=\boldsymbol{A}^+=\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1},\quad \text{for } m<n \tag{7-15}$$

如果误差代价函数仍写作：

$$\mathscr{E}(\boldsymbol{C})=\frac{1}{2}\text{trace}(\boldsymbol{E}\boldsymbol{E}^T) \tag{7-16}$$

其中的误差矩阵$\boldsymbol{E}$采用式（7-13）中的定义，则$\mathscr{E}(\boldsymbol{C})$的梯度计算如下：

$$\nabla_{\boldsymbol{C}}\mathscr{E}(\boldsymbol{C})=\frac{\partial\mathscr{E}(\boldsymbol{C})}{\partial\boldsymbol{C}}=\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{A}\boldsymbol{A}^T-\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{A}^T \tag{7-17}$$

欲使式（7-17）中的梯度最小（即在全局极小点处，误差代价函数逼近零），就有下式成立：

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{A}\boldsymbol{A}^T-\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{A}^T=\boldsymbol{A}^T\boldsymbol{A}(\boldsymbol{C}\boldsymbol{A}-\boldsymbol{I})\boldsymbol{A}^T=\boldsymbol{0} \tag{7-18}$$

或

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{A}\boldsymbol{A}^T-\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{A}^T=\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{C}-\boldsymbol{I})\boldsymbol{A}\boldsymbol{A}^T=\boldsymbol{0} \tag{7-19}$$

如果式（7-18）的两端都左乘$(\boldsymbol{A}^T\boldsymbol{A})^{-1}$（假设这个逆存在），则基于最速下降法（即式（7-5））的学习规则与式（7-11）相同。另一方面，如果式（7-19）两端右乘$(\boldsymbol{A}\boldsymbol{A}^T)^{-1}$（同样假设这个逆存在），则导出的学习规则与式（7-9）相同。因此，式（7-9）和式（7-11）中给出的学习规则，虽然是在$\boldsymbol{A}$为方阵且$\boldsymbol{A}$非奇异的假设条件下导出的，却对在一般情形下对于求$\boldsymbol{A}$的伪逆同样有效。但是为了在一般情形下表示误差，必须使用式（7-13）的表达式。这种计算一个矩阵的伪逆的递归方法与奇异值分解（SVD）（参见A.2.14节）的方法相似。

由式（7-9），计算矩阵广义逆的学习规则的向量矩阵形式如下：

$$\boldsymbol{C}(k+1)=\boldsymbol{C}(k)+\mu\boldsymbol{A}^T[\boldsymbol{I}-\boldsymbol{A}\boldsymbol{C}(k)]=\boldsymbol{C}(k)+\mu\Delta\boldsymbol{C}(k) \tag{7-20}$$

这个学习规则的标量形式可以如下导出。首先把式（7-20）中的更新项$\Delta\boldsymbol{C}(k)=\boldsymbol{A}^T[\boldsymbol{I}-\boldsymbol{A}\boldsymbol{C}(k)]$写成：

$$\Delta\boldsymbol{C}(k)=\boldsymbol{A}^T-\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{C}(k) \tag{7-21}$$

296

然后，把式（7-21）中的“固定”项$\boldsymbol{W}\equiv\boldsymbol{A}^T\boldsymbol{A}$写成标量形式：

$$w_{ih}=\sum_{q=1}^{m}a_{qi}a_{qh} \tag{7-22}$$

其中$i=1,2,\cdots,n$，$h=1,2,\cdots,n$。因此，用式（7-22）和式（7-9）中的学习规则的标量形式可以写成：

$$c_{ij}(k+1)=c_{ij}(k)+\mu[a_{ii}-\sum_{h=1}^{n}w_{ih}c_{hj}(k)] \tag{7-23}$$

其中$j=1,2,\cdots,m$。最后做点补充说明，可以证明式（7-4）的连续时间学习规则可以用一个非线性矩阵微分方程代替，而不影响最终结果和解的精确性[6, 7]。这个非线性可以是一个合适的函数$\xi(\cdot)$，它的自变量就是式（7-4）中的梯度，即$\xi[\nabla_{\boldsymbol{C}}\mathscr{E}(\boldsymbol{C})]$。

*方法2：矩阵求逆*

到目前为止，我们介绍了一种递归计算矩阵伪逆的学习规则。然而，相关的神经网络体系结构还没有给出。现在给出另一种计算方阵伪逆的神经网络方法。这个方法是由Wang和Mendel在1992[1]首次提出的，也被Cichocki和Unbehauen提出过[6]。给定一个矩阵$\boldsymbol{A}\in\Re^{n\times n}$，求它的逆（或，如果$\boldsymbol{A}$非奇异就求它的伪逆），即求$\boldsymbol{C}=\boldsymbol{A}^{-1}$。这种神经网络方法是一种恒等映射方法，它可以由图7-1的方框图来描述。由图7-1可以看出，这个神经网络在作线性变换$\boldsymbol{u}=\boldsymbol{A}\boldsymbol{x}$和$\boldsymbol{y}=\boldsymbol{C}\boldsymbol{u}$。因此，经过训练后

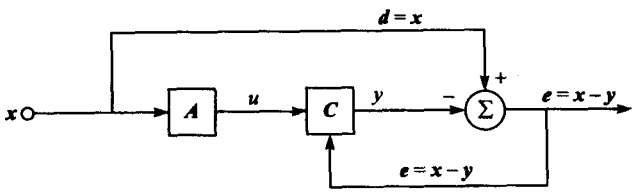



图7-1 使用恒等映射方法求矩阵逆的结构化神经网络方框图

$$\boldsymbol{y}=\boldsymbol{C}\boldsymbol{u}=\boldsymbol{C}\boldsymbol{A}\boldsymbol{x}=\boldsymbol{I}\boldsymbol{x}=\boldsymbol{x} \tag{7-24}$$

第一个问题：向量$\boldsymbol{x}$是什么？它是外部兴奋输入信号（或从图7-1来说，是期望信号$\boldsymbol{d}=\boldsymbol{x}$）。这样，神经网络经过训练后，输出信号$\boldsymbol{y}$将等于输入信号$\boldsymbol{x}$。因此，图7-1中的映射误差$\boldsymbol{e}=\boldsymbol{x}-\boldsymbol{y}$将为0。可用作外部兴奋信号的最简单输入是一组线性无关的二值向量，如$\boldsymbol{x}_i=[1,0,0,\cdots,0]^T$，

297 $x_2=[0,1,0,\cdots,0]^T,\cdots,x_n=[0,0,0,\cdots,1]^T$。然而，为了在训练中获得更快的收敛速度，最好使用一组线性无关的双极值向量，$x_1=[1,-1,-1,\cdots,-1]^T$, $x_2=[-1,1,-1,\cdots,-1]^T$, $\cdots$, $x_n=[-1,-1,-1,\cdots,1]^T$[6]。这种求矩阵逆的神经网络方法与经典的基于增广矩阵求逆的Gauss-Jordan消去法[8]非常相似。

$$\boldsymbol{A}_{\text{aug}}=\{\boldsymbol{A}\vdots\boldsymbol{I}_{n\times n}\} \tag{7-25}$$

Gauss-Jordan消去法只是对增广矩阵$\boldsymbol{A}_{\text{aug}}$进行初等行（列）变换，直到$\boldsymbol{A}$变成单位矩阵，原来的单位矩阵变成$\boldsymbol{A}$的逆，即$\{\boldsymbol{A}\vdots\boldsymbol{I}_{n\times n}\}\sim\cdots\sim\{\boldsymbol{I}_{n\times n}\vdots\boldsymbol{A}^{-1}\}$。学习规则的向量矩阵形式可以从误差代价函数推出

$$\mathscr{E}(\boldsymbol{C})=\frac{1}{2}\|\boldsymbol{e}\|_2^2=\frac{1}{2}\boldsymbol{e}^T\boldsymbol{e} \tag{7-26}$$

其中$\boldsymbol{e}=\boldsymbol{x}-\boldsymbol{y}$（映射误差），如图7-1所示。目标是调整$\boldsymbol{C}$来最小化式（7-26）中的$\mathscr{E}(\boldsymbol{C})$。式（7-26）可写成：

$$\begin{aligned}\mathscr{E}(\boldsymbol{C})&=\frac{1}{2}\boldsymbol{e}^T\boldsymbol{e}=\frac{1}{2}(\boldsymbol{x}^T-\boldsymbol{y}^T)(\boldsymbol{x}-\boldsymbol{y})=\frac{1}{2}(\boldsymbol{x}^T\boldsymbol{x}-\boldsymbol{x}^T\underbrace{\boldsymbol{y}}_{\boldsymbol{C}\boldsymbol{u}}-\boldsymbol{y}^T\boldsymbol{x}+\boldsymbol{y}^T\boldsymbol{y})\\&=\frac{1}{2}(\boldsymbol{x}^T\boldsymbol{x}-\boldsymbol{x}^T\boldsymbol{C}\boldsymbol{u}-\boldsymbol{u}^T\boldsymbol{C}^T\boldsymbol{x}+\boldsymbol{u}^T\boldsymbol{C}^T\boldsymbol{C}\boldsymbol{u})\end{aligned} \tag{7-27}$$

基于最速下降法的连续时间学习规则可以写成一组矩阵微分方程如下：

$$\frac{\mathrm{d}\boldsymbol{C}(t)}{\mathrm{d}t}=-\mu\nabla_{\boldsymbol{C}}\mathscr{E}(\boldsymbol{C}) \tag{7-28}$$

计算式（7-28）中必需的梯度$\nabla_{\boldsymbol{C}}\mathscr{E}(\boldsymbol{C})$需要使用式（7-7）和式（7-8）中的一般结果。因此，计算式（7-27）的梯度得到：

$$\nabla_{\boldsymbol{C}}\mathscr{E}(\boldsymbol{C})=-\boldsymbol{x}\boldsymbol{u}^T+\boldsymbol{C}\boldsymbol{u}\boldsymbol{u}^T=-(\boldsymbol{x}-\underbrace{\boldsymbol{C}\boldsymbol{u}}_{\boldsymbol{y}})\boldsymbol{u}^T=-(\boldsymbol{x}-\boldsymbol{y})\boldsymbol{u}^T \tag{7-29}$$

且连续时间学习规则可以写成

$$\frac{\mathrm{d}\boldsymbol{C}(t)}{\mathrm{d}t}=\mu[\boldsymbol{x}(t)-\boldsymbol{y}(t)]\boldsymbol{u}^T(t) \tag{7-30}$$

其中$\mu>0$是学习率参数。学习规则的离散时间形式如下：

$$\boldsymbol{C}(k+1)=\boldsymbol{C}(k)+\mu[\boldsymbol{x}(k)-\boldsymbol{y}(k)]\boldsymbol{u}^T(k) \tag{7-31}$$

其中

$$\boldsymbol{u}(k)=\boldsymbol{A}\boldsymbol{x}(k)\quad\text{和}\quad\boldsymbol{y}(k)=\boldsymbol{C}(k)\boldsymbol{u}(k) \tag{7-32}$$

298 从图7-1中的方框图可知。式（7-31）和式（7-32）中的学习规则的标量形式可分别写成

$$c_{ij}(k+1)=c_{ij}(k)+\mu[x_{ip}-y_{ip}(k)]u_{jp} \tag{7-33}$$

$$u_{ip}=\sum_{h=1}^{n}a_{ih}x_{hp}\quad\text{和}\quad y_{ip}(k)=\sum_{h=1}^{n}C_{ih}(k)u_{hp} \tag{7-34}$$

其中$i,j=1,2,\cdots,n$, $k=0,1,2,\cdots$, $p=1,2,\cdots,n$（即前述的$n$个激励双极值输入信号）。在每个训练回合，这$n$个激励输入信号都要提交一次。在每个训练回合，这$n$个线性无关的双极值向量都可随机打乱提交给神经网络的顺序。图7-2展示了用映射误差法求矩阵逆的神经网络体系结构。

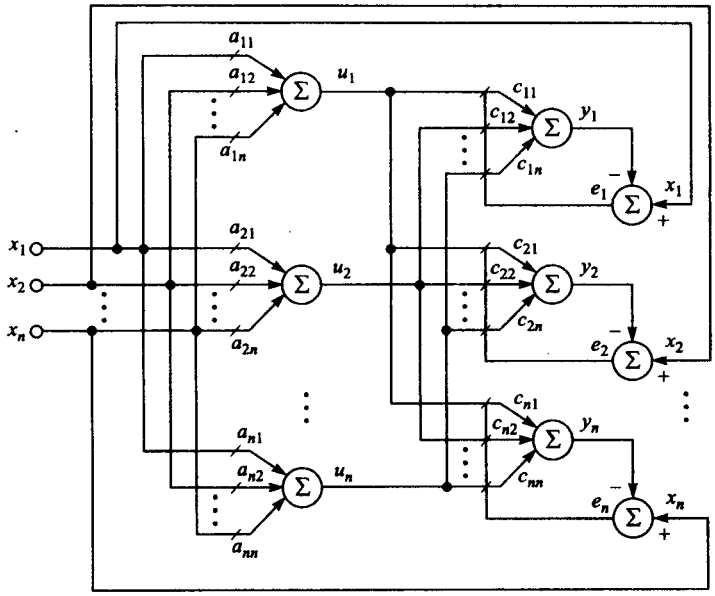



图7-2 使用映射误差方法求矩阵逆的神经网络体系结构。第一层由固定权值的线性神经元组成$(a_{ij}, i, j = 1, 2, \cdots n)$，权值由矩阵$A$的行给出。第二层由自适应权值的线性神经元构成$(c_{ij}, i, j = 1, 2, \cdots, n)$，权值由$A^{-1}$的行给出

**例7.1** 下面的例子演示了两种神经网络方法计算非奇异矩阵逆的能力。给定矩阵

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 & 2 \\ 3 & 2 & 3 \\ 1 & 1 & 1 \end{bmatrix} \tag{7-35}$$

用两种神经网络方法计算的结果，产生四位小数，矩阵逆$\boldsymbol{C}^{NN}$为：

$$\boldsymbol{C}^{NN} = \boldsymbol{A}^{-1} = \begin{bmatrix} -1.000\,0 & 1.000\,0 & -1.000\,0 \\ 0.000\,0 & -1.000\,0 & 3.000\,0 \\ 1.000\,0 & 0.000\,0 & -1.000\,0 \end{bmatrix} \tag{7-36}$$

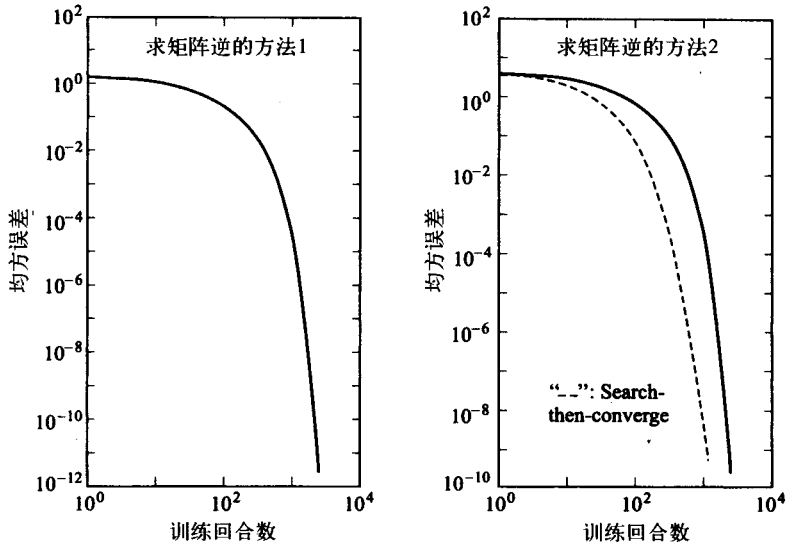
299

**表7-1 例7.1神经网络训练细节**

| 求矩阵逆的方法 | 学习率参数 | 训练回合数 | $\|\boldsymbol{C}^{M} - \boldsymbol{C}^{NN}\|_2$ |
|---|---|---|---|
| 1 | $\mu = 0.065$ | 2 500 $[\boldsymbol{C}(k=0) = \boldsymbol{0}]$ | $8.004\,8 \times 10^{-6}$ |
| 2 | $\mu = 0.015$ | 2 500 $[\boldsymbol{C}(k=0) = \boldsymbol{0}]$ | $4.798\,6 \times 10^{-5}$ |
| 2 | $\mu_0 = 0.06$ | 1 200 $[\boldsymbol{C}(k=0) = \boldsymbol{0}]$ | $5.914\,9 \times 10^{-5}$ |
| （使用搜索然后收敛调度） | $\tau = 500$ | | |

与式（7-36）所示同样的结果可以用MATLAB中inv函数获得，记作$\boldsymbol{C}^{M}$。表7-1总结了两个用来计算矩阵逆的神经网络的训练结果。表7-1中出现的范数$\|\cdot\|_2$是矩阵的谱范数（参见A.2.13节）。如A.2.13节所述，谱范数可以作为矩阵最大的奇异值计算得到。如表7-1所示，各有一个固定的学习率参数用于两种方法。式（2-36）给出的学习率参数的搜索然后收敛调度策略也用于方法2。如果产生四位小数，（调度策略对方法1无效）获得与式（7-36）相同的结果。如表7-1所示，使用学习率参数的搜索然后收敛调度策略提高了方法2算法的收敛速度。训练回合数减少了一半以上。图7-3显示了两种神经网络（包括应用搜索然后收敛调度策略的

方法2）训练来计算式（7-35）中$A$的逆的过程中的均方误差。




300

图7-3　用于求矩阵逆的两种神经网络在训练期间的均方误差

## 7.3 LU分解

LU分解（或因子分解）法是最重要的矩阵因子分解法之一[8]。原因是，对于联立线性代数方程组表示的系统，即$\boldsymbol{Ax}=\boldsymbol{b}$（试图计算解$\boldsymbol{x}$），如果矩阵$\boldsymbol{A}\in\Re^{n\times n}$能分解成下三角矩阵$\boldsymbol{L}\in\Re^{n\times n}$和上三角矩阵$\boldsymbol{U}\in\Re^{n\times n}$的乘积，那么问题就可以轻易地解决。即：

$$\boldsymbol{A}=\boldsymbol{LU} \tag{7-37}$$

下三角矩阵$\boldsymbol{L}$常常取单位下三角矩阵，因为对角线元素都是1。在矩阵$\boldsymbol{A}$被分解成$\boldsymbol{L}$和$\boldsymbol{U}$之后，两个三角系统可以用反向代入法[8]求解，即，

$$\begin{aligned}&\text{先求}\boldsymbol{y}\text{：}\boldsymbol{Ly}=\boldsymbol{b}\\&\text{再求}\boldsymbol{x}\text{：}\boldsymbol{Ux}=\boldsymbol{y}\end{aligned} \tag{7-38}$$

由式（7-37），显然可以把$\boldsymbol{A}$的LU分解写成

$$\boldsymbol{A}=\boldsymbol{L}\underbrace{\boldsymbol{D}\boldsymbol{D}^{-1}}_{I}\boldsymbol{U}=(\underbrace{\boldsymbol{LD}}_{\hat{L}})(\underbrace{\boldsymbol{D}^{-1}\boldsymbol{U}}_{\hat{U}})=\hat{\boldsymbol{L}}\hat{\boldsymbol{U}} \tag{7-39}$$

其中$\boldsymbol{D}\in\Re^{n\times n}$是一个非奇异对角矩阵，假设$\boldsymbol{L}$只是一个下三角矩阵而非单位下三角矩阵。由此可以推断矩阵$\boldsymbol{A}$的LU分解并不一定是唯一的。然而，如果给矩阵$\boldsymbol{A}$加上特定的条件，那么唯一的LU分解的确存在[2, 9]。如下陈述（来自Golub和Van Loan[2]）：

给定矩阵$\boldsymbol{A}\in\Re^{n\times n}$，如果对于$k=1,2,\cdots,n-1$，$\det(\boldsymbol{A}(1{:}k,1{:}k))\neq 0$，那么就存在$\boldsymbol{A}$的LU因子分解，其中$\boldsymbol{L}\in\Re^{n\times n}$是单位下三角，$\boldsymbol{U}\in\Re^{n\times n}$是上三角。如果$\boldsymbol{A}$的LU因子分解存在，且有$\rho(\boldsymbol{A})=n$成立，那么这个LU因子分解唯一，并且$\det(\boldsymbol{A})=\boldsymbol{U}$的对角线元素的乘积。

证明可以在Golub和Van Loan[2]中找到。

如果假设存在一个对角矩阵$\boldsymbol{D}\in\Re^{n\times n}$，其中$\rho(\boldsymbol{D})=n$，那么可以写出矩阵$\boldsymbol{A}$的LDU因子分解为

$$\boldsymbol{A}=\boldsymbol{LDU} \tag{7-40}$$

这样，式（7-40）可以写成

$$L(DU) = A \tag{7-41}$$

和

$$(LD)U = A$$

以上两式都是$A$的可能因子分解。

特殊情况下，如果$A$是对称的（即$A = A^T$），那么因子分解如下[2, 9] 301

$$A = LDL^T \tag{7-42}$$

此外，如果$A > 0$（即$A$是正定的），那么$D$的对角元素都是非零正值，则因子分解可进一步简化。就是说，如果假设$D = \Delta^2$，则$\Delta$称作$D$的平方根（$\Delta = \sqrt{D}$），且$A$的$LDL^T$因子分解可以写作

$$A = LDL^T = L\Delta^2 L^T = (L\Delta)(\Delta L^T) = L_c L_c^T \tag{7-43}$$

其中$L_c$是下三角矩阵。式（7-43）中$A$的分解称为Cholesky因子分解[2, 8, 9]。在信号处理中矩阵的Cholesky因子分解有很多应用；一个特别重要的应用就是谱因子分解中自相关矩阵的因子分解[10]。在控制理论中，矩阵的Cholesky因子分解用于李雅普诺夫稳定性理论[11]，也用于平方根卡尔曼（Kalman）滤波[12]。

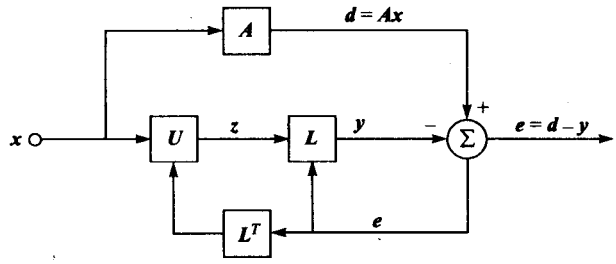



图7-4 用于LU分解的结构化神经网络的方框图。注意该神经网络由误差反向传播训练

一般而言，矩阵$A$可以是长方形的，即$A \in \Re^{m\times n}$。这将使LU分解中的$L \in \Re^{m\times m}$且$U \in \Re^{m\times n}$。然而，讨论仅限于$A$是方阵的情况。已经提出了许多计算矩阵LU分解的数值方法。最基本的方法是基于高斯消去法及其变体，比如Crout和Doolittle[2, 9]。其他数值方法可以在Press[13]或[14]中找到。这里关心的是设计一种计算矩阵LU分解的神经计算方法，并且希望在实时应用背景下使用此方法。图7-4是执行LU分解的结构神经网络的方框图，图7-5显示了双层神经网络体系结构。

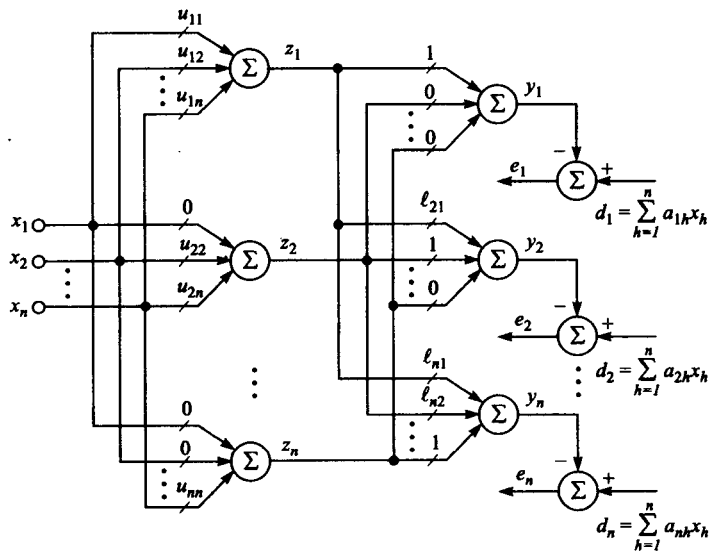



图7-5 一个通过误差反向传播训练的实现LU分解的两层神经网络体系结构

从图7-4开始，对于一个给定的外部激励输入信号$\boldsymbol{x}$，期望信号（或目标信号）由$\boldsymbol{d}=\boldsymbol{A}\boldsymbol{x}$给出。这些输入信号将视为前面7.2节所述的线性无关双极值向量。并且，由图7-4，可以把向量误差信号$\boldsymbol{e}$写作

$$\boldsymbol{e}=\boldsymbol{d}-\underbrace{\boldsymbol{y}}_{\boldsymbol{Lz}}=\boldsymbol{A}\boldsymbol{x}-\boldsymbol{L}\underbrace{\boldsymbol{z}}_{\boldsymbol{Ux}}=\boldsymbol{A}\boldsymbol{x}-\boldsymbol{L}\boldsymbol{U}\boldsymbol{x}=(\boldsymbol{A}-\boldsymbol{L}\boldsymbol{U})\boldsymbol{x} \tag{7-44}$$

因此，由式（7-44）可以看出，对于一个合适的兴奋信号$\boldsymbol{x}$，在神经网络已经学习过两个三角。

矩阵的$\boldsymbol{L}$和$\boldsymbol{U}$的元素之后，由于$\boldsymbol{A}=\boldsymbol{L}\boldsymbol{U}$，所以误差向量将是0。现在用式（7-44）可以定义误差代价函数，它是分解中矩阵$\boldsymbol{L}$和$\boldsymbol{U}$的函数，如下

$$\begin{aligned}\mathscr{E}(\boldsymbol{L},\boldsymbol{U})&=\frac{1}{2}\|\boldsymbol{e}\|_2^2=\frac{1}{2}\boldsymbol{e}^T\boldsymbol{e}=\frac{1}{2}[(\boldsymbol{A}-\boldsymbol{L}\boldsymbol{U})\boldsymbol{x}]^T[(\boldsymbol{A}-\boldsymbol{L}\boldsymbol{U})\boldsymbol{x}]\\&=\frac{1}{2}(\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}-\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{L}\boldsymbol{U}\boldsymbol{x}-\boldsymbol{x}^T\boldsymbol{U}^T\boldsymbol{L}^T\boldsymbol{A}\boldsymbol{x}+\boldsymbol{x}^T\boldsymbol{U}^T\boldsymbol{L}^T\boldsymbol{L}\boldsymbol{U}\boldsymbol{x})\end{aligned} \tag{7-45}$$

由式（7-45）可以推出两个基于最速下降梯度法的学习规则（用于计算$\boldsymbol{L}$和$\boldsymbol{U}$）。离散时间形式的学习规则是

$$\boldsymbol{L}(k+1)=\boldsymbol{L}(k)-\mu\nabla_{\boldsymbol{L}}\mathscr{E}(\boldsymbol{L},\boldsymbol{U}) \tag{7-46}$$

和

$$\boldsymbol{U}(k+1)=\boldsymbol{U}(k)-\mu\nabla_{\boldsymbol{U}}\mathscr{E}(\boldsymbol{L},\boldsymbol{U}) \tag{7-47}$$

其中$\mu>0$，是学习率参数。因此，我们必须由式（7-45）给出的误差代价分别计算式（7-46）和式（7-47）中的两个梯度项$\nabla_{\boldsymbol{L}}\mathscr{E}(\boldsymbol{L},\boldsymbol{U})$和$\nabla_{\boldsymbol{U}}\mathscr{E}(\boldsymbol{L},\boldsymbol{U})$。式（7-45）关于矩阵$\boldsymbol{L}$的梯度可以用式（7-7）、式（7-8）中的一般性结果和合适的链式规则来求出。结果是

302
ι
303

$$\nabla_{\boldsymbol{L}}\mathscr{E}(\boldsymbol{L},\boldsymbol{U})=-\boldsymbol{A}\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{U}^T+\boldsymbol{L}\boldsymbol{U}\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{U}^T=\underbrace{(-\boldsymbol{A}\boldsymbol{x}+\boldsymbol{L}\boldsymbol{U}\boldsymbol{x})}_{-\boldsymbol{e}}\underbrace{\boldsymbol{x}^T\boldsymbol{U}^T}_{\boldsymbol{z}^T}=-\boldsymbol{e}\boldsymbol{z}^T \tag{7-48}$$

因此，由式（7-46）和式（7-48），$\boldsymbol{L}$的学习规则是

$$\boldsymbol{L}(k+1)=\boldsymbol{L}(k)+\mu\boldsymbol{e}(k)\boldsymbol{z}^T(k) \tag{7-49}$$

其中

$$\boldsymbol{e}(k)=[\boldsymbol{A}-\boldsymbol{L}(k)\boldsymbol{U}(k)]\boldsymbol{x}(k) \tag{7-50}$$

由式（7-44）和图7-4中的方框图可知

$$\boldsymbol{z}(k)=\boldsymbol{U}(k)\boldsymbol{x}(k) \tag{7-51}$$

式（7-45）中误差代价函数关于矩阵$\boldsymbol{U}$的梯度是

$$\nabla_{\boldsymbol{U}}\mathscr{E}(\boldsymbol{L},\boldsymbol{U})=-\boldsymbol{L}^T\boldsymbol{A}\boldsymbol{x}\boldsymbol{x}^T+\boldsymbol{L}^T\boldsymbol{L}\boldsymbol{U}\boldsymbol{x}\boldsymbol{x}^T=\boldsymbol{L}^T\underbrace{(-\boldsymbol{A}\boldsymbol{x}+\boldsymbol{L}\boldsymbol{U}\boldsymbol{x})}_{-\boldsymbol{e}}\boldsymbol{x}^T=-\boldsymbol{L}^T\boldsymbol{e}\boldsymbol{x}^T \tag{7-52}$$

因此，由式（7-47）和式（7-52），计算$\boldsymbol{U}$的离散时间学习规则是

$$\boldsymbol{U}(k+1)=\boldsymbol{U}(k)+\mu\boldsymbol{L}^T(k+1)\boldsymbol{e}(k)\boldsymbol{x}^T(k) \tag{7-53}$$

其中$\boldsymbol{e}(k)$由式（7-50）给出。式（7-49）～式（7-51）和式（7-53）给出的学习规则的向量矩阵形式并不能求出矩阵$\boldsymbol{A}$的LU分解，因为没有对$\boldsymbol{L}$和$\boldsymbol{U}$加限制。就是说，没有限制$\boldsymbol{L}$是对角线元素均为单位元素的下三角，也没有限制$\boldsymbol{U}$是上三角。因此，现在给出式（7-49）～式（7-51）和式（7-53）给出的学习规则的标量形式，并且给两个矩阵的元素加上合适的限制。单位下三角矩阵$\boldsymbol{L}$的学习规则的标量形式是

$$\ell_{ij}(k+1)=\ell_{ij}(k)+\mu e_{ip}(k)z_{jp}(k) \tag{7-54}$$

其中$k = 0, 1, 2, \cdots, p = 1, 2, \cdots, n$（即前述的$n$个双极值激励输入信号）。对$\boldsymbol{L}$的元素$\ell_{ij}(\forall i, j = 1, 2, \cdots, n)$的限制是：（1）如果$i = j$，则$\ell_{ij} = 1$；（2）如果$i < j$，则$\ell_{ij} = 0$；（3）如果$i > j$，则$\boldsymbol{L}$中剩下的元素（连接权值）根据式（7-54）更新。式（7-54）中$e_{ip}$和$z_{ip}$的标量形式可以这样计算

$$e_{ip} = d_{ip} - y_{ip} \tag{7-55}$$

其中

$$d_{ip} = \sum_{h=1}^{n} a_{ih} x_{hp} \tag{7-56}$$

$$y_{ip} = \sum_{h=1}^{n} \ell_{ih} z_{hp} \tag{7-57}$$

304

和

$$z_{ip} = \sum_{h=1}^{n} u_{ih} x_{hp} \tag{7-58}$$

其中$i = 1, 2, \cdots, n$。

上三角矩阵$\boldsymbol{U}$的学习规则的标量形式是

$$u_{ij}(k+1) = u_{ij}(k) + \mu \left[ \sum_{h=i}^{n} \ell_{hi}(k+1) e_{hp}(k) \right] x_{jp} \tag{7-59}$$

其中$k = 0, 1, 2, \cdots, p = 1, 2, \cdots, n$。对$\boldsymbol{U}$的元素$u_{ij}(\forall i, j = 1, 2, \cdots, n)$的限制是：（1）如果$i > j$，则$u_{ij} = 0$；（2）如果$i \leqslant j$，则$\boldsymbol{U}$中剩下的元素（连接权值）根据式（7-59）更新。上面给出的加在$\boldsymbol{L}$和$\boldsymbol{U}$上的限制将确保学习规则产生正确的LU分解。观察式（7-54）和式（7-59）中学习规则的标量形式，现在两个学习规则之间存在对偶关系。在式（7-59）$\boldsymbol{U}$的学习规则中，先计算下三角矩阵$\boldsymbol{L}$的元素的更新值，然后更新$\boldsymbol{U}$的元素，求和。

## 7.4 QR因子分解

QR因子分解（或正交三角化）是另一种非常重要的矩阵分解，在工程与科学领域广泛地使用。这种方法在计算矩阵的全部特征向量上有广泛的应用[2, 9]。QR因子分解还有许多别的应用，特别是在信号处理领域[12, 15, 16]。比如，平方根自适应滤波方法[12, 16]和最小二乘格型滤波器[12]的基础就是QR因子分解。

矩阵$\boldsymbol{A} \in \Re^{m \times n}$（假设$m \geqslant n$）的QR因子分解可以写作[2, 6, 12, 17, 18]

$$\boldsymbol{A} = \boldsymbol{Q} \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{bmatrix} \tag{7-60}$$

其中$\boldsymbol{Q} \in \Re^{m \times m}$是正交的（即$\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$），$\boldsymbol{R} \in \Re^{n \times n}$是一个上三角矩阵，$\boldsymbol{0} \in \Re^{(m-n) \times n}$是一个零矩阵。然而，将研究$\boldsymbol{A}$是方阵的特殊情况，即$\boldsymbol{A} \in \Re^{n \times n}$。在这种情况下，$\boldsymbol{A}$的QR因子分解可以简单地写成

$$\boldsymbol{A} = \boldsymbol{QR} \tag{7-61}$$

其中$\boldsymbol{Q} \in \Re^{n \times n}$是正交矩阵（即$\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{QQ}^T = \boldsymbol{I}$），并且$\boldsymbol{R} \in \Re^{n \times n}$是一个上三角矩阵。已经开发了很多算法可以把$\boldsymbol{A}$因子分解成矩阵$\boldsymbol{Q}$和$\boldsymbol{R}$，比如，改进的Gram-Schmidt，快速Givens，block Householder和Hessenberg方法[2]。然而，本节用来计算方阵的QR因子分解的神经网络方法是源于一种特殊应用的在线计算需求。目的是开发一种在矩阵$\boldsymbol{A}$的一些元素随时间缓慢变化的情况下，可以在线修正$\boldsymbol{Q}$和$\boldsymbol{R}$的神经计算方法。使用任一种标准的QR因子分解数值方法[2, 9]，

305

根据每一个普通矩阵$A$的标准值，考虑神经网络的初始权值由该值确定。已经提出两种神经网络体系结构用以解决QR因子分解问题。这些神经计算方法都是基于Wang和Mendel[4]的开创性工作和Cichocki和Unbehauen[6]提供的素材。

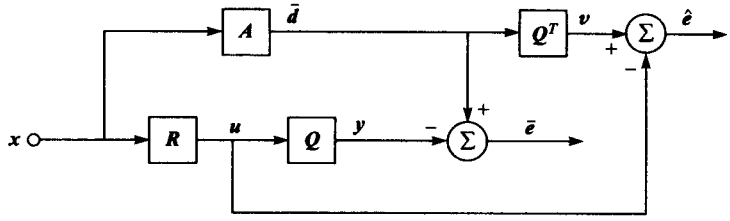



图7-6　QR因子分解的（方法1）结构化神经网络的方框图

方法1：QR因子分解

图7-6显示了第一种执行QR因子分解的结构神经网络的方框图。如图7-6所示，定义了两个误差向量$\bar{e}$和$\hat{e}$。误差向量$\bar{e}$可表示作

$$\bar{e} = \bar{d} - y = Ax - Qu = Ax - QRx = (A - QR)x \tag{7-62}$$

从式（7-62）可以看出，在神经网络学习因子分解（$Q$和$R$）时，对于一个合适的外部激励输入信号$x$，误差向量$\bar{e}$将会逼近零。为了推导训练神经网络的学习规则，我们把误差向量$\bar{e}$定义作

$$\bar{e} = Ax - Qu \tag{7-63}$$

式中不作代换$u = Rx$，而是保留$u$作为一个辅助变量。同样，由图7-6中的方框图，误差向量$\hat{e}$可以表示作

$$\hat{e} = v - u = Q^T\bar{d} - u = Q^TAx - Rx = (Q^TA - R)x \tag{7-64}$$

在式（7-64）中，在神经网络学习QR因子分解时，对于一个合适的外部激励输入信号$x$，误差向量$\hat{e}$将逼近零。这在式（7-64）中并不明显。然而，由$A$的QR因子分解，即$A = QR$，如果两边左乘$Q^T$，得到$Q^TA = \underbrace{Q^TQ}_{I}R = R$，就可以看出结果。由式（7-64）把误差向量定义为

306

$$\hat{e} = Q^T\bar{d} - u \tag{7-65}$$

其中$u$如同式（7-63）中保留作为一个辅助变量。可用的最好的兴奋输入信号是7.2节介绍的线性无关双极性向量组。可以定义一个总误差代价函数为

$$\mathscr{E}(\bar{e},\hat{e}) = \mathscr{E}[\bar{e}(Q,R),\hat{e}(Q,R)] = \frac{1}{2}\|\bar{e}\|_2^2 + \frac{v}{2}\|\hat{e}\|_2^2 \tag{7-66}$$

其中$v > 0$是惩罚参数，它可以使代价函数的两项之间保持适当的比例。一般$v = 1$。最小化式（7-66）中的第一项，确保矩阵$A$被分解成两个矩阵$Q$和$R$。最小化式（7-66）中的第二项，促使矩阵$Q$正交化（即$Q^TQ = I$）。这里的目标是基于梯度最速下降法推导出两个$Q$和$R$的学习规则。如下两个矩阵微分方程可以推出结果

$$\frac{dQ(t)}{dt} = -\mu\nabla_Q\mathscr{E}(\bar{e},\hat{e}) = -\mu\frac{\partial\mathscr{E}(\bar{e},\hat{e})}{\partial Q} \tag{7-67}$$

和

$$\frac{\mathrm{d}\boldsymbol{R}(t)}{\mathrm{d}t}=-\mu\nabla_{\boldsymbol{R}}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}})=-\mu\frac{\partial\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}})}{\partial\boldsymbol{R}} \tag{7-68}$$

其中$\mu>0$，是学习率参数。通过逼近式（7-67）和式（7-68）中的导数，可以用矩阵差分方程写出学习规则的离散时间形式，分别如下

$$\boldsymbol{Q}(k+1)=\boldsymbol{Q}(k)-\mu\nabla_{\boldsymbol{Q}}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}}) \tag{7-69}$$

和

$$\boldsymbol{R}(k+1)=\boldsymbol{R}(k)-\mu\nabla_{\boldsymbol{R}}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}}) \tag{7-70}$$

因此，必须计算式（7-69）和式（7-70）中的两个梯度项。为了计算梯度$\nabla_{\boldsymbol{Q}}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}})$，可以用式（7-63）和式（7-65）中的两个误差向量的表达式，把式（7-66）中的总误差代价函数写成

$$\begin{aligned}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}})&=\frac{1}{2}(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{Q}\boldsymbol{u})^T(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{Q}\boldsymbol{u})+\frac{v}{2}(\boldsymbol{Q}^T\bar{\boldsymbol{d}}-\boldsymbol{u})^T(\boldsymbol{Q}^T\bar{\boldsymbol{d}}-\boldsymbol{u})\\&=\frac{1}{2}(\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}-\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{Q}\boldsymbol{u}-\boldsymbol{u}^T\boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{x}+\boldsymbol{u}^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{u})\\&\quad+\frac{v}{2}(\bar{\boldsymbol{d}}^T\boldsymbol{Q}\boldsymbol{Q}^T\bar{\boldsymbol{d}}-\bar{\boldsymbol{d}}^T\boldsymbol{Q}\boldsymbol{u}-\boldsymbol{u}^T\boldsymbol{Q}^T\bar{\boldsymbol{d}}+\boldsymbol{u}^T\boldsymbol{u})\end{aligned} \tag{7-71}$$

通过使用式（7-7）和式（7-8）中的两个一般结果和合适的链式规则（参见A.3.4.2节），可以由式（7-71）得到梯度$\nabla_{\boldsymbol{Q}}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}})$

$$\begin{aligned}\nabla_{\boldsymbol{Q}}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}})&=-\boldsymbol{A}\boldsymbol{x}\boldsymbol{u}^T+\boldsymbol{Q}\boldsymbol{u}\boldsymbol{u}^T+v\bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^T\boldsymbol{Q}-v\bar{\boldsymbol{d}}\boldsymbol{u}^T\\&=-\underbrace{(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{Q}\boldsymbol{u})}_{\bar{\boldsymbol{e}}}\boldsymbol{u}^T+v\bar{\boldsymbol{d}}\underbrace{(\boldsymbol{Q}^T\bar{\boldsymbol{d}}-\boldsymbol{u})}_{\hat{\boldsymbol{e}}}^T=-\bar{\boldsymbol{e}}\boldsymbol{u}^T+v\bar{\boldsymbol{d}}\hat{\boldsymbol{e}}^T\end{aligned} \tag{7-72}$$

307

因此，由式（7-69）和式（7-72），$\boldsymbol{Q}$的离散时间学习规则的批量（向量矩阵）形式可以写作

$$\boldsymbol{Q}(k+1)=\boldsymbol{Q}(k)+\mu[\bar{\boldsymbol{e}}(k)\boldsymbol{u}^T(k)-v\bar{\boldsymbol{d}}(k)\hat{\boldsymbol{e}}^T(k)] \tag{7-73}$$

其中$\mu>0.\ v>0$，$\bar{\boldsymbol{e}}(k)$和$\hat{\boldsymbol{e}}(k)$分别由式（7-63）和式（7-65）给出，$\boldsymbol{u}(k)=\boldsymbol{R}\boldsymbol{x}$(k)。式（7-73）中$\boldsymbol{Q}$的学习规则的标量形式可以写作

$$q_{ij}(k+1)=q_{ij}(k)+\mu[\bar{e}_i(k)u_j(k)-v\bar{d}_i(k)\hat{e}_j(k)] \tag{7-74}$$

其中$i,j=1,2,\cdots,n$。标量形式的$\bar{\boldsymbol{e}},\hat{\boldsymbol{e}},\bar{\boldsymbol{d}}$和$\boldsymbol{u}$可以这样计算

$$\bar{e}_i=\sum_{h=1}^{n}a_{ih}x_{hp}-\sum_{h=1}^{n}q_{ih}u_{hp} \tag{7-75}$$

其中

$$u_{ip}=\sum_{h=1}^{n}r_{ih}x_{hp} \tag{7-76}$$

和

$$\hat{e}_i=\sum_{h=1}^{n}q_{hi}\bar{d}_{hp}-u_{ip} \tag{7-77}$$

其中

$$\bar{d}_{ip}=\sum_{h=1}^{n}a_{ih}x_{hp} \tag{7-78}$$

其中$i=1,2,\cdots,n$。

现在来推导$\boldsymbol{R}$的学习规则，用$\boldsymbol{u}=\boldsymbol{Rx}$（由图7-6的方框图）来代换式（7-63）和式（7-65）$\bar{\boldsymbol{e}}$和$\hat{\boldsymbol{e}}$的表达式中的辅助变量$\boldsymbol{u}$，即$\bar{\boldsymbol{e}}=\boldsymbol{Ax}-\boldsymbol{QRx}$和$\hat{\boldsymbol{e}}=\boldsymbol{Q}^T\bar{\boldsymbol{d}}-\boldsymbol{Rx}$。总误差代价函数现在可以写成

$$\begin{aligned}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}})&=\frac{1}{2}(\boldsymbol{Ax}-\boldsymbol{QRx})^T(\boldsymbol{Ax}-\boldsymbol{QRx})+\frac{v}{2}(\boldsymbol{Q}^T\bar{\boldsymbol{d}}-\boldsymbol{Rx})^T(\boldsymbol{Q}^T\bar{\boldsymbol{d}}-\boldsymbol{Rx})\\&=\frac{1}{2}(\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{Ax}-\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{QRx}-\boldsymbol{x}^T\boldsymbol{R}^T\boldsymbol{Q}^T\boldsymbol{Ax}+\boldsymbol{x}^T\boldsymbol{R}^T\boldsymbol{Q}^T\boldsymbol{QRx})\\&\quad+\frac{v}{2}(\bar{\boldsymbol{d}}^T\boldsymbol{QQ}^T\bar{\boldsymbol{d}}-\bar{\boldsymbol{d}}^T\boldsymbol{QRx}-\boldsymbol{x}^T\boldsymbol{R}^T\boldsymbol{Q}^T\bar{\boldsymbol{d}}+\boldsymbol{x}^T\boldsymbol{R}^T\boldsymbol{Rx})\end{aligned}\tag{7-79}$$

式（7-79）中误差代价函数关于$\boldsymbol{R}$的梯度是

$$\begin{aligned}\nabla_{\boldsymbol{R}}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}})&=-\boldsymbol{Q}^T\boldsymbol{Axx}^T+\boldsymbol{Q}^T\boldsymbol{Q}\underbrace{\boldsymbol{Rx}}_{\boldsymbol{u}}\boldsymbol{x}^T-v\boldsymbol{Q}^T\bar{\boldsymbol{d}}\boldsymbol{x}^T+v\underbrace{\boldsymbol{Rx}}_{\boldsymbol{u}}\boldsymbol{x}^T\\&=-\boldsymbol{Q}^T(\underbrace{\boldsymbol{Ax}-\boldsymbol{Qu}}_{\bar{\boldsymbol{e}}})\boldsymbol{x}^T-v(\underbrace{\boldsymbol{Q}^T\bar{\boldsymbol{d}}-\boldsymbol{u}}_{\hat{\boldsymbol{e}}})\boldsymbol{x}^T\\&=-\boldsymbol{Q}^T\bar{\boldsymbol{e}}\boldsymbol{x}^T-v\hat{\boldsymbol{e}}\boldsymbol{x}^T=-(\boldsymbol{Q}^T\bar{\boldsymbol{e}}+v\hat{\boldsymbol{e}})\boldsymbol{x}^T\end{aligned}\tag{7-80}$$

308

因此，由式（7-70）和式（7-80），$\boldsymbol{R}$的离散时间学习规则的向量矩阵形式可以写成

$$\boldsymbol{R}(k+1)=\boldsymbol{R}(k)+\mu[\boldsymbol{Q}^T(k+1)\,\bar{\boldsymbol{e}}(k)+v\,\hat{\boldsymbol{e}}(k)]\boldsymbol{x}^T(k)\tag{7-81}$$

其中$\mu>0$, $v>0$。然而，因为$\boldsymbol{R}$必须是上三角矩阵，所以必须给$\boldsymbol{R}$加上附加的限制。因此，对于$\boldsymbol{R}$矩阵的元素$r_{ij}(i,j=1,2,\cdots,n)$，当$i\leqslant j$时，由式（7-81）中的学习规则决定$r_{ij}$；$i>j$时，$r_{ij}=0$。式（7-81）中学习规则的标量形式可以写作，当$i\leqslant j$时

$$r_{ij}(k+1)=r_{ij}(k)+\mu\left[\sum_{h=1}^{n}q_{hi}(k+1)\bar{e}_h(k)+v\hat{e}_i(k)\right]x_j(k)\tag{7-82}$$

当$i>j$时，$r_{ij}=0$。其中$i,j=1,2,\cdots,n$。两个误差项的标量形式已经在式（7-75）～式（7.78）中给出。

方法2：QR因子分解

图7-7中的方框图总结另一种可以自适应地执行QR因子分解的神经计算方法。相应的多层结构化神经网络体系结构如图7-8所示。由图7-7中结构化神经网络的方框图，可以看出这种体系结构与QR因子分解的方法1非常相似。实际上，这种学习规则与方法1的学习规则很相似。然而，作为一种可以用误差反向传播法训练的多层神经网络，这种体系结构更容易实现。

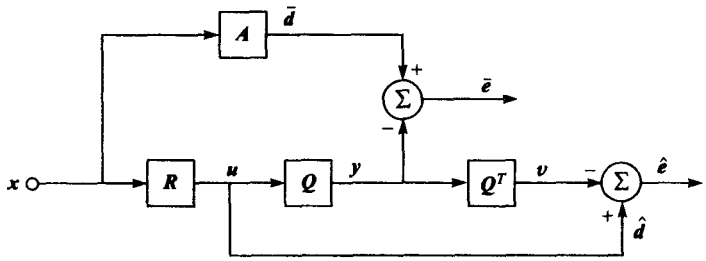



图7-7　QR因子分解（方法2）的另一种结构

总误差代价函数定义成与方法1完全相同，见式（7-66）。由图7-7，两个误差向量写作

$$\bar{\boldsymbol{e}}=\boldsymbol{Ax}-\boldsymbol{Qu}\tag{7-83}$$

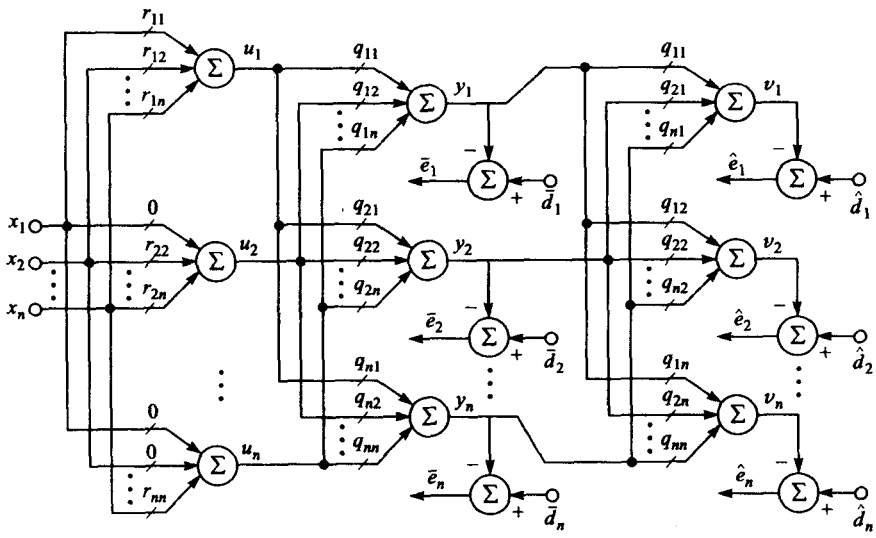


图7-8 QR因子分解（方法2）的多层结构化神经网络

和

$$\hat{\boldsymbol{e}} = \boldsymbol{u} - \boldsymbol{Q}^T \boldsymbol{y} \tag{7-84}$$

为了推导正交矩阵$\boldsymbol{Q}$的离散时间学习规则，使用与方法1式（7-69）中相同的最速下降梯度形式。在式（7-84）中，没有对$\boldsymbol{y}$作代换，因为这个向量被当作常量。理由是，与$\boldsymbol{R}$的收敛速率相比，神经网络学习正交矩阵$\boldsymbol{Q}$的速度相对要快一些。把式（7-83）和式（7-84）代入式（7-66），并计算总误差代价函数对$\boldsymbol{Q}$的梯度可推出$\boldsymbol{Q}$的向量矩阵形式的离散时间学习规则如下

$$\boldsymbol{Q}(k+1) = \boldsymbol{Q}(k) + \mu[\bar{\boldsymbol{e}}(k)\boldsymbol{u}^T(k) + \nu\boldsymbol{y}(k)\hat{\boldsymbol{e}}^T(k)] \tag{7-85}$$

其中$\mu > 0$，$\nu > 0$，$\bar{\boldsymbol{e}}(k)$和$\hat{\boldsymbol{e}}(k)$分别由式（7-83）和式（7-84）给出，$\boldsymbol{u}(k) = \boldsymbol{R}\boldsymbol{x}(k)$，并且$\boldsymbol{y}(k) = \boldsymbol{Q}\boldsymbol{u}(k)$。

$\boldsymbol{R}$的离散时间学习规则可以由$\boldsymbol{Q}$的总误差代价函数推出，由图7-7中方框图，其中$\boldsymbol{u}$被$\boldsymbol{R}\boldsymbol{x}$代换。对$\boldsymbol{R}$采用与方法1式（7-70）中相同的最速下降梯度形式。从而导出$\boldsymbol{R}$的向量矩阵形式的离散时间学习规则

$$\boldsymbol{R}(k+1) = \boldsymbol{R}(k) + \mu[\boldsymbol{Q}^T(k+1)\bar{\boldsymbol{e}}(k) - \nu\hat{\boldsymbol{e}}(k)]\boldsymbol{x}^T(k) \tag{7-86}$$

其中$\mu > 0$, $\nu > 0$。加在$\boldsymbol{R}$的元素上的合适限制与方法1中的限制相同。

$\boldsymbol{Q}$和$\boldsymbol{R}$的学习规则的标量形式可写作

$$\begin{aligned} q_{ij}(k+1) &= q_{ij}(k) + \mu[\bar{e}_i(k)u_j(k) + \nu y_i(k)\hat{e}_j(k)] \\ r_{ij}(k+1) &= r_{ij}(k) + \mu\left[\sum_{h=1}^{n} q_{hi}(k+1)\bar{e}_h(k) - \nu\hat{e}_i(k)\right]x_j(k) \end{aligned} \tag{7-87}$$

对于$\boldsymbol{R}$矩阵的元素$r_{ij}(i, j = 1, 2, \cdots, n)$，当$i \leqslant j$时，根据式（7-87）中的学习规则自适应地决定$r_{ij}$；$i > j$时，$r_{ij} = 0$。$\bar{\boldsymbol{e}}, \hat{\boldsymbol{e}}, \boldsymbol{y}$和$\boldsymbol{u}$的标量形式可以用与方法1中相似的方式来计算。 309 ~ 310

## 7.5 舒尔分解

舒尔（Schur）分解[2]是另一种矩阵因子分解方法，它把矩阵$\boldsymbol{A} \in \Re^{n\times n}$分解成一个正交矩阵$\boldsymbol{Q} \in \Re^{n\times n}$和一个上三角矩阵$\boldsymbol{R} \in \Re^{n\times n}$，从而$\boldsymbol{A}$可以写成

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}\boldsymbol{Q}^T \tag{7-88}$$

沿着上三角矩阵$\boldsymbol{R}$的对角线向下的元素是矩阵$\boldsymbol{A}$的特征值。由式（7-88）矩阵$\boldsymbol{R}$可写作

$$\boldsymbol{R}=\boldsymbol{D}+\boldsymbol{N}=\boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{Q} \tag{7-89}$$

其中$\boldsymbol{D}\in\Re^{n\times n}$是一个对角线上为$\boldsymbol{A}$的特征值的对角矩阵，$\boldsymbol{N}\in\Re^{n\times n}$是严格的上三角[2]。

Wang和Mendel[4]首先提出了一种对方阵进行舒尔分解的结构化神经网络。这里给出的方法与他们提出的方法相似。图7-9展示了舒尔分解的网络方框图。由该图可写出两个误差向量

$$\bar{\boldsymbol{e}}=\boldsymbol{A}\boldsymbol{x}-\boldsymbol{Q}\boldsymbol{R}\boldsymbol{b} \tag{7-90}$$

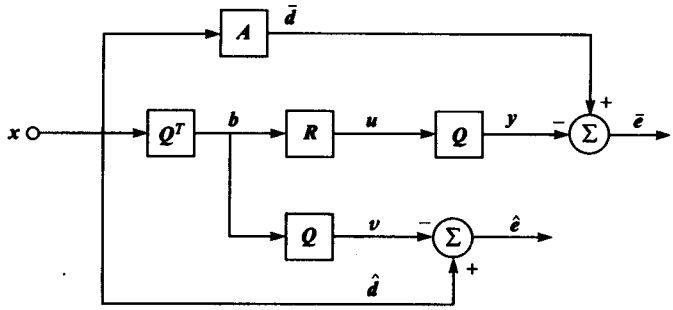



图7-9　结构化神经网络方框图，通过误差反向传播训练实现舒尔分解

其中

$$\boldsymbol{b}=\boldsymbol{Q}^T\boldsymbol{x} \tag{7-91}$$

和

$$\hat{\boldsymbol{e}}=\boldsymbol{x}-\boldsymbol{Q}\boldsymbol{b} \tag{7-92}$$

由式（7-90）和式（7-91）明显可以看出，当神经网络学习$\boldsymbol{A}$的舒尔分解（即$\boldsymbol{Q}$和$\boldsymbol{R}$）时，对于一个合适的外部激励输入信号，误差向量$\bar{\boldsymbol{e}}$将收敛到零。此外，由式（7-91）和式（7-92）可以看出，当神经网络学习舒尔分解中的正交矩阵$\boldsymbol{Q}$时，对于一个合适的外部激励输入信号，误差向量$\hat{\boldsymbol{e}}$将收敛到零。

311　使用式（7-90）和式（7-92）中的两个误差向量，可以定义一个如QR因子分解式（7-66）的总误差代价函数。训练神经网络的离散时间学习规则在形式上与QR因子分解完全一样，已经在式（7-69）和式（7-70）中给出。总误差代价函数给出如下：

$$\begin{aligned}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}})&=\frac{1}{2}\|\bar{\boldsymbol{e}}\|_2^2+\frac{v}{2}\|\hat{\boldsymbol{e}}\|_2^2=\frac{1}{2}(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{Q}\boldsymbol{R}\boldsymbol{b})^T(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{Q}\boldsymbol{R}\boldsymbol{b})+\frac{v}{2}(\boldsymbol{x}-\boldsymbol{Q}\boldsymbol{b})^T(\boldsymbol{x}-\boldsymbol{Q}\boldsymbol{b})\\&=\frac{1}{2}(\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}-\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{Q}\boldsymbol{R}\boldsymbol{b}-\boldsymbol{b}^T\boldsymbol{R}^T\boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}^T\boldsymbol{R}^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{R}\boldsymbol{b})\\&\quad+\frac{v}{2}(\boldsymbol{x}^T\boldsymbol{x}-\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{b}-\boldsymbol{b}^T\boldsymbol{Q}^T\boldsymbol{x}+\boldsymbol{b}^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{b})\end{aligned} \tag{7-93}$$

其中$v>0$是惩罚参数。由式（7-93）以及式（7-7）和式（7-8）中给出的标量对矩阵微分的一般性结果，可以计算式（7-69）和式（7-70）中的梯度项。如下

$$\begin{aligned}\nabla_{\boldsymbol{Q}}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}})&=-\boldsymbol{A}\boldsymbol{x}\boldsymbol{b}^T\boldsymbol{R}^T+\boldsymbol{Q}\boldsymbol{R}\boldsymbol{b}\boldsymbol{b}^T\boldsymbol{R}^T-v\boldsymbol{x}\boldsymbol{b}^T+v\boldsymbol{Q}\boldsymbol{b}\boldsymbol{b}^T\\&=-\underbrace{(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{Q}\boldsymbol{R}\boldsymbol{b})}_{\bar{\boldsymbol{e}}}\boldsymbol{b}^T\boldsymbol{R}^T-v\underbrace{(\boldsymbol{x}-\boldsymbol{Q}\boldsymbol{b})}_{\hat{\boldsymbol{e}}}\boldsymbol{b}^T\\&=-\bar{\boldsymbol{e}}\underbrace{\boldsymbol{b}^T\boldsymbol{R}^T}_{\boldsymbol{u}^T}-v\hat{\boldsymbol{e}}\boldsymbol{b}^T=-\bar{\boldsymbol{e}}\boldsymbol{u}^T-v\hat{\boldsymbol{e}}\boldsymbol{b}^T\end{aligned} \tag{7-94}$$

和

$$\nabla_R \mathscr{E}(\bar{e},\hat{e}) = -\boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{x}\boldsymbol{b}^T + \boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{R}\boldsymbol{b}\boldsymbol{b}^T = -\boldsymbol{Q}^T(\underbrace{\boldsymbol{A}\boldsymbol{x}-\boldsymbol{Q}\boldsymbol{R}\boldsymbol{b}}_{\bar{e}})\boldsymbol{b}^T = -\boldsymbol{Q}^T\bar{e}\boldsymbol{b}^T \tag{7-95}$$

因此，$\boldsymbol{Q}$和$\boldsymbol{R}$的两个学习规则分别如下

$$\boldsymbol{Q}(k+1) = \boldsymbol{Q}(k) + \mu[\bar{e}(k)\boldsymbol{u}^T(k) + \nu\hat{e}(k)\boldsymbol{b}^T(k)] \tag{7-96}$$

和

$$\boldsymbol{R}(k+1) = \boldsymbol{R}(k) + \mu\boldsymbol{Q}^T(k+1)\bar{e}(k)\boldsymbol{b}^T(k) \tag{7-97}$$

其中$\mu>0, \nu>0$（通常，$\nu=1$），$\boldsymbol{u}=\boldsymbol{R}\boldsymbol{b}, \boldsymbol{b}=\boldsymbol{Q}^T\boldsymbol{x}$，$\bar{e}$和$\hat{e}$分别由式（7-90）和式（7-92）给出。最后，因为$\boldsymbol{R}$限制为一个上三角，当$i \leqslant j$时，式（7-97）中的学习规则应用于$\boldsymbol{R}$的$r_{ij}$个元素（连接权值）的自适应，当$i>j$时，$r_{ij}=0$（$i,j=1,2,\cdots,n$）。$\boldsymbol{Q}$和$\boldsymbol{R}$的学习规则的标量形式分别给出如下

$$q_{ij}(k+1) = q_{ij}(k) + \mu[\bar{e}_{ip}(k)u_j(k) + \nu\hat{e}_{ip}(k)b_{jp}(k)] \tag{7-98}$$

其中$i, j = 1, 2, \cdots, n$和

$$r_{ij}(k+1) = r_{ij}(k) + \mu\left[\sum_{h=1}^{n} q_{hi}(k+1)\bar{e}_{hp}(k)\right]b_{jp}(k) \tag{7-99}$$

312

其中$i \leqslant j$；当$i>j$时，$r_{ij}=0$，且对于前述的$n$个双极性兴奋输入向量，$p = 1, 2, \cdots, n$。辅助变量的标量形式如下

$$u_i = \sum_{h=i}^{n} r_{ih}b_{hp} \tag{7-100}$$

和

$$b_{ip} = \sum_{h=1}^{n} q_{hi}x_{hp} \tag{7-101}$$

其中$i = 1, 2, \cdots, n$。两个误差项的标量形式如下

$$\bar{e}_{ip} = \sum_{h=1}^{n} a_{ih}x_{hp} - \sum_{h=1}^{n} q_{ih}u_h \tag{7-102}$$

和

$$\hat{e}_{ip} = x_{ip} - \sum_{h=1}^{n} q_{ih}b_{hp} \tag{7-103}$$

其中$i = 1, 2, \cdots, n$。对于$\boldsymbol{A}$为对称矩阵（即$\boldsymbol{A}^T = \boldsymbol{A}$）的特殊情况，最好的外部激励输入信号是让$\boldsymbol{x}$为一个随机向量，就是说，是零均值高斯白噪声单位方差，而不是双极值向量。

## 7.6 谱因子分解——特征值分解（EVD）（对称特征值问题）

如果一个方阵$\boldsymbol{A} \in \Re^{n\times n}$有不同的特征值$\lambda_1, \lambda_2, \cdots, \lambda_n$和相应的特征向量$\upsilon_1, \upsilon_2, \cdots, \upsilon_n$（列向量），则可以构造一个相似（非奇异）变换

$$\boldsymbol{V} = [\upsilon_1, \upsilon_2, \cdots, \upsilon_n] \tag{7-104}$$

矩阵$\boldsymbol{A}$可写作

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1} \tag{7-105}$$

其中$\boldsymbol{\Lambda}$是一个对角矩阵，其对角线上是$\boldsymbol{A}$的特征值[2, 8, 17, 18]，即

$$\boldsymbol{\Lambda} = \text{diag}[\lambda_1, \lambda_2, \cdots, \lambda_n] \tag{7-106}$$

在特定情形下，如果$\boldsymbol{A}$有非相异（重复）特征值，矩阵仍然可以对角化[11]。然而，仅限于讨论$\boldsymbol{A}$有相异特征值的情况。

313 式（7-105）中的结果可以从标准特征值问题中获得（参见A.2.9节）

$$(\lambda_i \boldsymbol{I} - \boldsymbol{A})\upsilon_i = \boldsymbol{0} \tag{7-107}$$

其中$\lambda_i (i = 1, 2, \cdots, n)$是$\boldsymbol{A}$的特征值，$\upsilon_i (i = 1, 2, \cdots, n)$是$\boldsymbol{A}$的特征向量。把式（7-107）写作

$$\lambda_i \upsilon_i = \boldsymbol{A}\upsilon_i \tag{7-108}$$

其中$i = 1, 2, \cdots, n$，有

$$[\lambda_1\upsilon_1, \lambda_2\upsilon_2, \cdots, \lambda_n\upsilon_n] = [\boldsymbol{A}\upsilon_1, \boldsymbol{A}\upsilon_2, \cdots, \boldsymbol{A}\upsilon_n] \tag{7-109}$$

或

$$\underbrace{[\upsilon_1, \upsilon_2, \cdots, \upsilon_n]}_{\boldsymbol{V}}\underbrace{\text{diag}[\lambda_1, \lambda_2, \cdots, \lambda_n]}_{\boldsymbol{\Lambda}} = \boldsymbol{A}\underbrace{[\upsilon_1, \upsilon_2, \cdots, \upsilon_n]}_{\boldsymbol{V}} \tag{7-110}$$

或

$$\boldsymbol{V\Lambda} = \boldsymbol{AV} \tag{7-111}$$

现在式（7-111）右乘$\boldsymbol{V}^{-1}$，可得到

$$\boldsymbol{A} = \boldsymbol{V\Lambda V}^{-1} \tag{7-112}$$

而这正好是式（7-105）所陈述的。如果把$\boldsymbol{A}$限制为对称矩阵（即$\boldsymbol{A}^T = \boldsymbol{A}$），这就引出对称特征值问题，并且$\boldsymbol{V}$还是正交的，即

$$\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{VV}^T = \boldsymbol{I} \tag{7-113}$$

且

$$\boldsymbol{V}^{-1} = \boldsymbol{V}^T \tag{7-114}$$

因此，每个有相异特征值的实对称矩阵可以通过一个正交相似变换对角化。另一种陈述对称矩阵$\boldsymbol{A}$的特征向量的正交性质的方式是

$$\upsilon_i^T\upsilon_j = \begin{Bmatrix} 1 & i = j \\ 0 & i \neq j \\ & \forall i, j \end{Bmatrix} = \delta_{ij} \text{ (Kronecker delta)} \tag{7-115}$$

由式（7-112）和式（7-114）可得

$$\begin{aligned} \boldsymbol{A} = \boldsymbol{V\Lambda V}^T &= [\upsilon_1, \upsilon_2, \cdots, \upsilon_n]\text{diag}[\lambda_1, \lambda_2, \cdots, \lambda_n][\upsilon_1, \upsilon_2, \cdots, \upsilon_n]^T \\ &= \lambda_1\upsilon_1\upsilon_1^T + \lambda_2\upsilon_2\upsilon_2^T + \cdots + \lambda_n\upsilon_n\upsilon_n^T = \sum_{i=1}^{n}\lambda_i\upsilon_i\upsilon_i^T \end{aligned} \tag{7-116}$$

这就是$\boldsymbol{A}$的特征值分解。由式（7-116）明显看出$\boldsymbol{A}$的秩就是$n$个矩阵秩的和。然而，这些都是秩为1的外积矩阵（由$\boldsymbol{A}$的特征值构成），并且每个外积矩阵都乘以一个相应的特征值。因

314 此，如果所有特征值都非零，则$\boldsymbol{A}$是满秩。然而，如果$\boldsymbol{A}$的任何一个特征值为零，则$\boldsymbol{A}$秩亏损。

## 7.7 对称特征值问题的神经网络方法

最早的解对称特征值问题的神经网络方法可以在Wang和Mendel[4]的论文中找到。这个领域的其他工作可以在[6, 19]中找到。这里采用的神经计算方法与前一节所述的舒尔分解结构神

经网络很相似。就是说，如图7-10所示的对称特征值问题的结构神经网络与如图7-9所示用于舒尔分解的网络有相同的形式。实际上，对于矩阵$A \in \Re^{n\times n}$的舒尔分解，如果$A$对称（即$A^T = A$），由式（7-88）可得

$$A = QRQ^T = A^T = QR^TQ^T \Rightarrow R = R^T \tag{7-117}$$

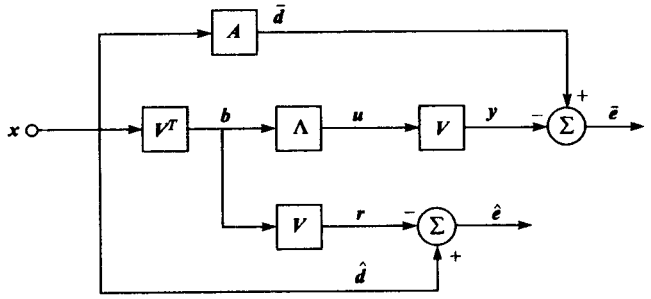



图7-10　通过误差反向传播训练的用于解决对称特征值问题的结构化神经网络方框图

由式（7-117）可以推断矩阵$R$不能是上三角矩阵，而必须是对角线上为$A$的特征值的对角矩阵，并且$Q$的列向量为$A$的相关特征向量。因此，对于对称矩阵$A$，$R = \Lambda$, $Q = V$，并且$A$的舒尔分解变成$A = V\Lambda V^T$，这正是式（7-116）给出的特征值分解。详细的神经网络体系结构如图7-11所示。

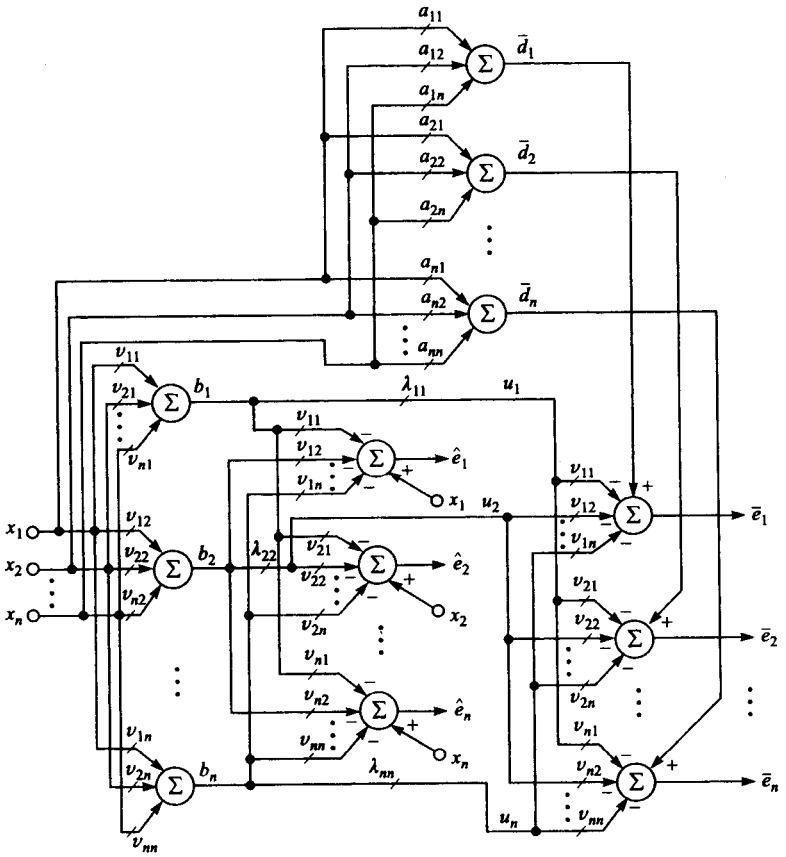



图7-11　解决对称特征值问题的详细神经网络体系结构

因此，由上述讨论，可以期望对称特征值问题中的$V$和$\Lambda$学习规则与舒尔分解中$Q$和$R$的学习规则有相同的基本形式。唯一的区别就是对矩阵$\Lambda$的元素的限制，因为在对称特征值问题中，$\Lambda$必须是对角矩阵。总误差代价函数的形式与舒尔分解神经计算方法中式（7-93）相同。因此，离散时间学习规则的向量矩阵形式如下

$$\boldsymbol{V}(k+1)=\boldsymbol{V}(k)+\mu[\bar{\boldsymbol{e}}(k)\boldsymbol{u}^T(k)+v\hat{\boldsymbol{e}}(k)\boldsymbol{b}^T(k)] \tag{7-118}$$

和

$$\Lambda(k+1)=\Lambda(k)+\mu\boldsymbol{V}^T(k+1)\bar{\boldsymbol{e}}(k)\boldsymbol{b}^T(k) \tag{7-119}$$

其中$\mu>0$, $\upsilon>0$（通常$\upsilon=1$），其中，由图7-10中的方框图

$$\bar{\boldsymbol{e}}=\boldsymbol{A}\boldsymbol{x}-\boldsymbol{V}\boldsymbol{u} \tag{7-120}$$

和

$$\boldsymbol{u}=\Lambda\boldsymbol{b} \tag{7-121}$$

其中

$$\boldsymbol{b}=\boldsymbol{V}^T\boldsymbol{x} \tag{7-122}$$

和

315
~
316

$$\hat{\boldsymbol{e}}=\boldsymbol{x}-\boldsymbol{V}\boldsymbol{b} \tag{7-123}$$

最好的外部激励输入$\boldsymbol{x}$是随机向量，即零均值高斯白噪声单位方差。式（7-118）和式（7-119）中的学习规则的标量形式分别如下

$$\upsilon_{ij}(k+1)=\upsilon_{ij}(k)+\mu[\bar{e}_i(k)u_j(k)+v\hat{e}_i(k)b_j(k)] \tag{7-124}$$

和

$$\lambda_{ij}(k+1)=\lambda_{ij}(k)+\mu\left[\sum_{h=1}^{n}\upsilon_{hi}(k+1)\bar{e}_h(k)\right]b_j(k) \tag{7-125}$$

其中$i, j=1, 2, \cdots, n$。

式（7-125）中的学习规则仅适用于$i=j$的情况，对于$i\neq j$，$\lambda_{ij}=0$，因为$\Lambda$是对角矩阵。基于舒尔分解的结果，确定辅助变量和误差项的标量形式是简单的。在Cichocki和Unbehauen[6]中，给出了图7-10中神经网络方框图的一种经济的实现。注意在图7-10中$V$被两个误差项的计算所共享。充分利用这个事实，设计出一种使用分时（多路复用）技术的简单结构网络。这个基于分时方法的简化神经网络具有由连接权值矩阵$V$描述的输出层计算单元，这些单元由两个计算渠道分时计算$\bar{\boldsymbol{e}}$和$\hat{\boldsymbol{e}}$得到。

最小/最大特征值问题

常常需要求取实对称矩阵$\boldsymbol{A}\in\Re^{n\times n}(\boldsymbol{A}^T=\boldsymbol{A})$的极值（即最小或最大）实特征值和相应特征向量。找$\boldsymbol{A}$的最小/最大特征值对应于获取函数$\mathscr{R}(\upsilon)$（Rayleigh商[2, 20, 21]）的最小值。$\mathscr{R}(\upsilon)$定义为

$$\mathscr{R}(\upsilon)=\pm\frac{\langle\boldsymbol{A}\upsilon,\upsilon\rangle}{\langle\upsilon,\upsilon\rangle}=\pm\frac{\upsilon^T\boldsymbol{A}\upsilon}{\upsilon^T\upsilon} \tag{7-126}$$

其中假设$\upsilon\in\Re^{n\times 1}$不会恒为零。式（7-126）中的瑞利（Rayleigh）商是能最小化$\|\boldsymbol{A}\upsilon-\lambda\upsilon\|_2^2$的唯一数。如果$\upsilon$为$\boldsymbol{A}$的一个特征向量，那么瑞利商$\mathscr{R}(\upsilon)$就是$\boldsymbol{A}$的相应特征值极值，如果取+号就是最小特征值，如果取－号就是最大特征值。

求取$\boldsymbol{A}$的最小/最大特征值的神经计算方法可以用带约束的最优化问题来表达

$$\text{最小化}\ \pm\frac{1}{2}\upsilon_i^T\boldsymbol{A}\upsilon_i\ \text{对于}\ i=1,2,\cdots,n \tag{7-127}$$

约束条件是

$$(\boldsymbol{A}-\lambda_i\boldsymbol{I})\boldsymbol{v}_i=\boldsymbol{0} \tag{7-128}$$

和

$$\boldsymbol{v}_i^T\boldsymbol{v}_i-1=0 \tag{7-129}$$

总误差代价函数可以用惩罚法（penalty method）[22, 23]来表示

$$\mathscr{E}(\boldsymbol{v}_i,\lambda_i)=\frac{1}{2}\left[\pm\alpha\boldsymbol{v}_i^T\boldsymbol{A}\boldsymbol{v}_i+\boldsymbol{e}_i^T\boldsymbol{e}_i+\frac{\beta}{2}(\boldsymbol{v}_i^T\boldsymbol{v}_i-1)^2\right] \tag{7-130}$$ 317

其中

$$\boldsymbol{e}_i=\boldsymbol{A}\boldsymbol{v}_i-\lambda_i\boldsymbol{v}_i \tag{7-131}$$

由标准特征值问题，$\alpha>0$, $\beta>0$是惩罚参数，$i=1, 2, \cdots, n$。通过最小化式（7-130）中的总误差代价函数，可以推导出求取最小/最大特征值$\lambda_i$和相应特征向量$\boldsymbol{v}_i$的两个学习规则。就是说，通过使用最速下降梯度法，离散时间学习规则（向量矩阵形式）如下

$$\lambda_i(k+1)=\lambda_i(k)+\mu\boldsymbol{v}_i^T(k)\boldsymbol{e}_i(k) \tag{7-132}$$

和

$$\boldsymbol{v}_i(k+1)=\boldsymbol{v}_i(k)-\mu\{\pm\alpha\boldsymbol{A}\boldsymbol{v}_i(k)+\boldsymbol{A}\boldsymbol{e}_i(k)-\lambda_i(k)\boldsymbol{e}_i(k)+\beta[\boldsymbol{v}_i^T(k)\boldsymbol{v}_i(k)-1]\boldsymbol{v}_i(k)\} \tag{7-133}$$

其中$\mu>0$，惩罚参数$\alpha>0$, $\beta>0$可以用一种独特的方式来调整。然而，还是建议在神经网络训练过程中逐渐减小惩罚参数$\alpha$（特别是在训练的最后阶段）[6]。再说一次，如果取+号对应的是最小特征值，那么取－号就产生最大特征值。式（7-133）中求特征向量的离散时间学习规则的标量形式可以写作

$$\begin{aligned}v_{ji}(k+1)=v_{ji}(k)-\mu\Bigg\{&\pm\alpha\sum_{h=1}^{n}a_{ih}v_{hi}(k)+\sum_{h=1}^{n}a_{ih}e_{hi}(k)\\&-\lambda_i(k)e_{ji}(k)+\beta\left[\sum_{h=1}^{n}v_{hi}^2(k)-1\right]v_{ji}(k)\Bigg\}\end{aligned} \tag{7-134}$$

其中$j=1, 2, \cdots, n$。

**例7.2** 求取下面矩阵的特征值和特征向量

$$\boldsymbol{A}=\begin{bmatrix}1 & 1 & \frac{1}{2} & -1\\ 1 & -5 & \frac{1}{2} & -3\\ \frac{1}{2} & \frac{1}{2} & -4 & \frac{1}{2}\\ -1 & -3 & \frac{1}{2} & -5\end{bmatrix} \tag{7-135}$$

通过使用MATLAB中的eig函数可以获得$\boldsymbol{A}$的特征值和特征向量如下（取四位小数）：

$$\Lambda^M=\mathrm{diag}[-3.9408 \quad -2.5376 \quad 1.6005 \quad -8.1221] \tag{7-136}$$

$$V^M=\begin{bmatrix}0.1245 & -0.3524 & -0.9275 & -0.0097\\ -0.1840 & 0.6436 & -0.2620 & -0.6952\\ -0.9734 & -0.1285 & -0.0836 & 0.1702\\ -0.0558 & -0.6671 & 0.2532 & -0.6983\end{bmatrix} \tag{7-137}$$

318 用本节介绍的神经网络方法，设学习率$\mu = 0.00255$, $\nu = 1$, $\boldsymbol{V}(k=0) = \boldsymbol{\Lambda}(k=0) = \boldsymbol{I}$，经过65 000步训练，特征值和特征向量为

$$\Lambda^{NN} = \text{diag}[1.6005, -3.9408, -8.1221, -2.5376] \tag{7-138}$$

$$V^{NN} = \begin{bmatrix} -0.9275 & 0.1245 & -0.0097 & -0.3524 \\ -0.2620 & -0.1840 & -0.6952 & 0.6436 \\ -0.0836 & -0.9734 & 0.1702 & -0.1285 \\ 0.2532 & -0.0558 & -0.6983 & -0.6671 \end{bmatrix} \tag{7-139}$$

用神经网络方法计算特征值和特征向量的误差可以通过计算矩阵$\boldsymbol{A}$与式（7-116）给出的$\boldsymbol{A}$的特征值分解$\boldsymbol{V\Lambda V}^T$的差值的模来量化。可以这样计算

$$\text{误差} = \|\boldsymbol{A} - \boldsymbol{V\Lambda V}^T\|_2 = 1.0969 \times 10^{-6} \tag{7-140}$$

其中矩阵的$\|\cdot\|_2$范数是$\boldsymbol{A} - \boldsymbol{V\Lambda V}^T$的最大奇异值。这些结果与MATLAB的结果是一样的，除了特征值/特征向量的顺序和一些特征向量符号相反。图7-12显示了计算特征值和特征向量的神经网络学习过程中均方误差的变化。

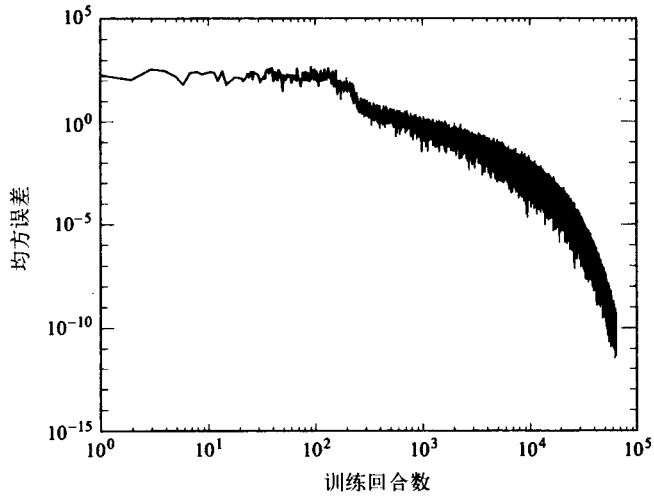



图7-12　在训练期间计算例7.2中对称矩阵的特征值和特征向量的均方误差

**例7.3**　如果仅想计算$\boldsymbol{A}$的最大特征值（和相应的特征向量），可以使用本节推导的神经计算方法。欲求如下矩阵的最大特征值和特征向量

$$\boldsymbol{A} = \begin{bmatrix} 0.45 & 0.55 \\ 0.55 & 0.45 \end{bmatrix} \tag{7-141}$$

使用MATLAB计算所得的特征值和特征向量如下

$$\Lambda^M = \text{diag}[-0.1, 1] \quad V^M = \begin{bmatrix} 0.7071 & 0.7071 \\ -0.7071 & 0.7071 \end{bmatrix} \tag{7-142}$$

319

为了寻找式（7-141）中矩阵的最大特征值和特征向量，神经网络的训练过程按如下方式进行：（1）用$\mu = 2 \times 10^{-4}$和$\alpha = \beta = 0.5$进行5000个训练回合，然后使用来自训练过程第一部分的突触权值；（2）用$\mu = 2 \times 10^{-4}$，$\beta = 0.5$进行5000个附加的训练回合，开始时用$\alpha = 0.5$，$\alpha$每个训练步减少0.1%。神经网络训练过程的第二部分多重复8次，即用来自前面训练过程的最终突触权值作为下一阶段的初始权值。因此，在总共50 000步训练后，神经网络收敛到$\boldsymbol{A}$的

最大特征值如下

$$\lambda_{\max}^{NN} = 1.0000 \tag{7-143}$$

相应的特征向量是

$$\upsilon_{\max}^{NN} = [0.7071, 0.7071]^T \tag{7-144}$$

这与式（7-141）和式（7-142）中给出的MATLAB的结果是相符的。

## 7.8 奇异值分解

矩阵的奇异值分解（SVD）[2, 8, 9, 17, 18, 20, 24]（参见A.2.14节）是最重要的矩阵分解方法之一。它在信号处理、控制理论、数据的参数建模等方面都有许多应用，特别是解决大多数线性最小二乘问题的上佳方法，尤其是对于病态条件矩阵[2]。如同在A.2.14节所解释的，SVD的基本目的是把一个矩阵$\boldsymbol{A} \in \Re^{m\times n}$分解成两个正交的矩阵$\boldsymbol{U} \in \Re^{m\times m}(\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{I})$和$\boldsymbol{V} \in \Re^{n\times n}(\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I})$和一个伪对角矩阵$\boldsymbol{S} \in \Re^{m\times n}$，即$\boldsymbol{S}$ = pseudodiag $(\sigma_1, \sigma_2, \cdots, \sigma_p)$，其中$p = \min(m, n)$，非负实数$\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_p \geqslant 0$称为$\boldsymbol{A}$的奇异值。因此$\boldsymbol{A}$可以写作

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T \tag{7-145}$$

相反，$\boldsymbol{S}$可写作

$$\boldsymbol{S} = \boldsymbol{U}^T\boldsymbol{A}\boldsymbol{V} \tag{7-146}$$

如果$\boldsymbol{A}$的秩是$r$，即$\rho(\boldsymbol{A}) = r$（这也是最小奇异值的索引），则$\boldsymbol{A}$的SVD可以写作

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T = \boldsymbol{U}_r\boldsymbol{S}_r\boldsymbol{V}_r^T = \sum_{i=1}^{r}\sigma_i\boldsymbol{u}_i\upsilon_i^T \tag{7-147}$$

其中，$\boldsymbol{u}_i(i = 1, 2, \cdots, r)$是$\boldsymbol{U}$的前$r$列；$\boldsymbol{v}_i(i = 1, 2, \cdots, r)$是$\boldsymbol{V}$的前$r$列。$\sigma_i(i = 1, 2, \cdots, r)$是伪对角矩阵$\boldsymbol{S}$的前$r$个奇异值（所有其余奇异值都是零）。因此，$\boldsymbol{U}_r \in \Re^{m\times r}$，$\boldsymbol{V}_r \in \Re^{n\times r}$和$\boldsymbol{S}_r \in \Re^{r\times r}$。

我们的目标是用神经计算方法计算两个正交矩阵$\boldsymbol{U}$和$\boldsymbol{V}$以及伪对角矩阵$\boldsymbol{S}$[1, 4, 6, 25, 26]。不失一般性，假设$m \geqslant n$，$\boldsymbol{U} \in \Re^{m\times n}(\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I})$，$\boldsymbol{V} \in \Re^{n\times n}(\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I})$，$\boldsymbol{S} = \mathrm{diag}[\sigma_1, \sigma_2, \cdots, \sigma_n] \in \Re^{n\times n}$，其中奇异值按$\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_n \geqslant 0$的顺序排列。采用的方法可以从图7-13所示的结构神经网络的方框图看出。总误差代价函数定义为 320

$$\mathscr{E}(\bar{\boldsymbol{e}}, \hat{\boldsymbol{e}}, \tilde{\boldsymbol{e}}) = \frac{1}{2}\|\bar{\boldsymbol{e}}\|_2^2 + \frac{\nu_1}{2}\|\hat{\boldsymbol{e}}\|_2^2 + \frac{\nu_2}{2}\|\tilde{\boldsymbol{e}}\|_2^2 \tag{7-148}$$

其中$\nu_1 > 0$，$\nu_2 > 0$是惩罚参数，通常，$0 < \nu_1, \nu_2 \leqslant 10$。最小化式（7-148）中的第一项可以把矩阵$\boldsymbol{A}$因子分解成三个矩阵$\boldsymbol{U}$，$\boldsymbol{S}$和$\boldsymbol{V}$。最小化式（7-148）中的第二项确保矩阵$\boldsymbol{V}$（其列为右奇异向量）是正交的，最小化第三项得到一个正交矩阵$\boldsymbol{U}$（其列为左奇异向量）。由最速下降梯度最优化方法推出的$\boldsymbol{V}$，$\boldsymbol{S}$和$\boldsymbol{U}$的三个离散时间学习规则的向量矩阵形式是

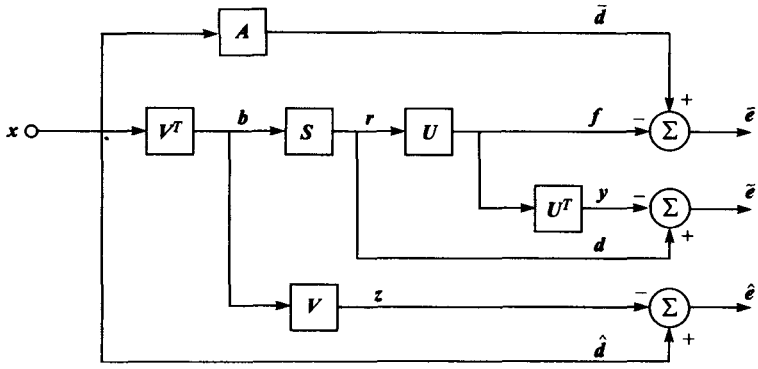
$$\boldsymbol{V}(k+1) = \boldsymbol{V}(k) - \mu_1\nabla_{\boldsymbol{V}}\mathscr{E}(\bar{\boldsymbol{e}}, \hat{\boldsymbol{e}}, \tilde{\boldsymbol{e}}) \tag{7-149}$$

$$\boldsymbol{S}(k+1) = \boldsymbol{S}(k) - \mu_2\nabla_{\boldsymbol{S}}\mathscr{E}(\bar{\boldsymbol{e}}, \hat{\boldsymbol{e}}, \tilde{\boldsymbol{e}}) \tag{7-150}$$

$$\boldsymbol{U}(k+1) = \boldsymbol{U}(k) - \mu_3\nabla_{\boldsymbol{U}}\mathscr{E}(\bar{\boldsymbol{e}}, \hat{\boldsymbol{e}}, \tilde{\boldsymbol{e}}) \tag{7-151}$$

其中$\mu_1 > 0$，$\mu_2 > 0$和$\mu_3 > 0$是三个独立的学习率参数。由图7-13中的方框图，误差向量的三个表达式写作

$$\begin{aligned}\bar{\boldsymbol{e}} &= \bar{\boldsymbol{g}} - \boldsymbol{f} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{U}\underbrace{\boldsymbol{r}}_{\boldsymbol{S}\boldsymbol{b}} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{U}\boldsymbol{S}\underbrace{\boldsymbol{b}}_{\boldsymbol{V}^T\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{x} \\ &= (\boldsymbol{A} - \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T)\boldsymbol{x}, \quad (\bar{\boldsymbol{e}} \in \Re^{m\times 1})\end{aligned} \tag{7-152}$$




321

图7-13　用于计算矩阵SVD的结构化的神经网络体系结构

$$\hat{e}=\underbrace{\hat{d}}_{x}-\underbrace{z}_{Vb}=x-V\underbrace{b}_{V^Tx}=x-VV^Tx=(I_{n\times n}-VV^T)x\quad(\hat{e}\in\Re^{n\times 1}) \tag{7-153}$$

$$\begin{aligned}\tilde{e}&=\underbrace{\tilde{d}}_{r}-\underbrace{y}_{U^Tf}=r-U^T\underbrace{f}_{Ur}=r-U^TUr=(I_{n\times n}-U^TU)\underbrace{r}_{SV^Tx}\\&=(I_{n\times n}-U^TU)SV^Tx\quad(\tilde{e}\in\Re^{n\times 1})\end{aligned} \tag{7-154}$$

因此，从式（7-152），式（7-153）和式（7-154）可以看出，对于一个合适的外部激励输入$x\in\Re^{n\times 1}$，当$U$的列都收敛到$A$的左奇异向量，$S$的对角线元素收敛到$A$的奇异值，且$V$的列都收敛到$A$的右奇异向量的时候，三个误差向量将收敛到零，并且神经网络也被正确地训练。

为了推导出式（7-149），式（7-150）和式（7-151）中的三个离散时间学习规则，分别计算三个矩阵$V$，$S$和$U$，必须求出三个梯度$\nabla_V\mathscr{E}(\bar{e},\hat{e},\tilde{e})$，$\nabla_S\mathscr{E}(\bar{e},\hat{e},\tilde{e})$和$\nabla_U\mathscr{E}(\bar{e},\hat{e},\tilde{e})$。为了先推得梯度$\nabla_V\mathscr{E}(\bar{e},\hat{e},\tilde{e})$，可以把式（7-148）中的总误差代价函数写成

$$\begin{aligned}\mathscr{E}(\bar{e},\hat{e},\tilde{e})&=\frac{1}{2}\|\bar{e}\|_2^2+\frac{v_1}{2}\|\hat{e}\|_2^2+\frac{v_2}{2}\|\tilde{e}\|_2^2=\frac{1}{2}\bar{e}^T\bar{e}+\frac{v_1}{2}\hat{e}^T\hat{e}+\frac{v_2}{2}\tilde{e}^T\tilde{e}\\&=\frac{1}{2}(\underbrace{x^TA^T-x^TVS^TU^T}_{\bar{e}^T})(\underbrace{Ax-USV^Tx}_{\bar{e}})\\&+\frac{v_1}{2}(\underbrace{x^T-x^TVV^T}_{\hat{e}^T})(\underbrace{x-VV^Tx}_{\hat{e}})+\frac{v_2}{2}(\underbrace{r^T-f^TU}_{\tilde{e}^T})(\underbrace{r-U^Tf}_{\tilde{e}})\\&=\frac{1}{2}(x^TA^TAx-x^TA^TUSV^Tx-x^TVS^TU^TAx+x^TVS^TU^TUSV^Tx)\\&+\frac{v_1}{2}(x^Tx-2x^TVV^Tx+x^TVV^TVV^Tx)+\frac{v_2}{2}(r^Tr-r^TU^Tf-f^TUr+f^TUU^Tf)\end{aligned} \tag{7-155}$$

通过使用式（7-7）和式（7-8）中给出的标量对矩阵微分的一般结果与合适的链式规则，可以计算式（7-155）对矩阵$V$的梯度。结果是

$$\nabla_V\mathscr{E}(\bar{e},\hat{e},\tilde{e})=\frac{\partial\mathscr{E}(\bar{e},\hat{e},\tilde{e})}{\partial V}=-x\bar{e}^TUS-v_1(x\hat{e}^TV+\hat{e}b^T) \tag{7-156}$$

由图7-13中的方框图，其中

322

$$b=V^Tx \tag{7-157}$$

因此，由式（7-149）以及式（7-156）中的梯度结果，$V$的离散时间学习规则的向量矩阵形式是

$$\boldsymbol{V}(k+1)=\boldsymbol{V}(k)+\mu_1\{\boldsymbol{x}(k)\bar{\boldsymbol{e}}^T(k)\boldsymbol{U}(k)\boldsymbol{S}(k)+\nu_1[\boldsymbol{x}(k)\hat{\boldsymbol{e}}^T(k)\boldsymbol{V}(k)+\hat{\boldsymbol{e}}(k)\boldsymbol{b}^T(k)]\} \tag{7-158}$$

其中

$$\bar{\boldsymbol{e}}=\boldsymbol{A}\boldsymbol{x}-\boldsymbol{U}\boldsymbol{r} \tag{7-159}$$

$$\hat{\boldsymbol{e}}=\boldsymbol{x}-\boldsymbol{V}\boldsymbol{b} \tag{7-160}$$

和

$$\boldsymbol{r}=\boldsymbol{S}\boldsymbol{b} \tag{7-161}$$

网络的外部激励输入$\boldsymbol{x}$应该使用如前所述的线性无关双极性向量组。

下一步可以计算总误差代价函数对矩阵$\boldsymbol{S}$的梯度，这个梯度是式（7-150）的学习规则所需要的。现在式（7-148）中的总误差代价函数可以写作

$$\begin{aligned}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}},\tilde{\boldsymbol{e}})&=\frac{1}{2}(\underbrace{\boldsymbol{x}^T\boldsymbol{A}^T-\boldsymbol{x}^T\boldsymbol{V}\boldsymbol{S}^T\boldsymbol{U}^T}_{\bar{\boldsymbol{e}}^T})(\underbrace{\boldsymbol{A}\boldsymbol{x}-\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{x}}_{\bar{\boldsymbol{e}}})\\&+\frac{\nu_1}{2}(\underbrace{\boldsymbol{x}^T-\boldsymbol{b}^T\boldsymbol{V}^T}_{\hat{\boldsymbol{e}}^T})(\underbrace{\boldsymbol{x}-\boldsymbol{V}\boldsymbol{b}}_{\hat{\boldsymbol{e}}})+\frac{\nu_2}{2}(\underbrace{\boldsymbol{r}^T-\boldsymbol{f}^T\boldsymbol{U}}_{\tilde{\boldsymbol{e}}^T})(\underbrace{\boldsymbol{r}-\boldsymbol{U}^T\boldsymbol{f}}_{\tilde{\boldsymbol{e}}})\\&=\frac{1}{2}(\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}-\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{x}-\boldsymbol{x}^T\boldsymbol{V}\boldsymbol{S}^T\boldsymbol{U}^T\boldsymbol{A}\boldsymbol{x}+\boldsymbol{x}^T\boldsymbol{V}\boldsymbol{S}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{x})\\&+\frac{\nu_1}{2}(\boldsymbol{x}^T\boldsymbol{x}-\boldsymbol{x}^T\boldsymbol{V}\boldsymbol{b}-\boldsymbol{b}^T\boldsymbol{V}^T\boldsymbol{x}+\boldsymbol{b}^T\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{b})\\&+\frac{\nu_2}{2}(\boldsymbol{r}^T\boldsymbol{r}-\boldsymbol{r}^T\boldsymbol{U}^T\boldsymbol{f}-\boldsymbol{f}^T\boldsymbol{U}\boldsymbol{r}+\boldsymbol{f}^T\boldsymbol{U}\boldsymbol{U}^T\boldsymbol{f})\end{aligned} \tag{7-162}$$

式（7-162）对$\boldsymbol{S}$的梯度为

$$\nabla_{\boldsymbol{S}}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}},\tilde{\boldsymbol{e}})=\frac{\partial\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}},\tilde{\boldsymbol{e}})}{\partial\boldsymbol{S}}=-\boldsymbol{U}^T\bar{\boldsymbol{e}}\boldsymbol{b}^T \tag{7-163}$$

因此，由式（7-150）以及式（7-163）中的梯度结果，$\boldsymbol{S}$的离散时间学习规则的向量矩阵形式是

$$\boldsymbol{S}(k+1)=\boldsymbol{S}(k)+\mu_2\boldsymbol{U}^T(k)\bar{\boldsymbol{e}}(k)\boldsymbol{b}^T(k) \tag{7-164}$$

这个式子仅适用于调整$\boldsymbol{S}$的对角线元素。$\boldsymbol{S}$中的所有其他元素都为零，因为$\boldsymbol{S}$应该是一个对角矩阵。

最后，可以计算总误差代价函数对矩阵$\boldsymbol{U}$的梯度。式（7-148）中的总误差代价函数现在可以写作 323

$$\begin{aligned}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}},\tilde{\boldsymbol{e}})&=\frac{1}{2}(\underbrace{\boldsymbol{x}^T\boldsymbol{A}^T-\boldsymbol{x}^T\boldsymbol{V}\boldsymbol{S}^T\boldsymbol{U}^T}_{\bar{\boldsymbol{e}}^T})(\underbrace{\boldsymbol{A}\boldsymbol{x}-\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{x}}_{\bar{\boldsymbol{e}}})\\&+\frac{\nu_1}{2}(\underbrace{\boldsymbol{x}^T-\boldsymbol{b}^T\boldsymbol{V}^T}_{\hat{\boldsymbol{e}}^T})(\underbrace{\boldsymbol{x}-\boldsymbol{V}\boldsymbol{b}}_{\hat{\boldsymbol{e}}})+\frac{\nu_2}{2}(\underbrace{\boldsymbol{x}^T\boldsymbol{V}\boldsymbol{S}^T-\boldsymbol{x}^T\boldsymbol{V}\boldsymbol{S}^T\boldsymbol{U}^T\boldsymbol{U}}_{\tilde{\boldsymbol{e}}^T})(\underbrace{\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{x}-\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{x}}_{\tilde{\boldsymbol{e}}})\\&=\frac{1}{2}(\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}-\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{x}-\boldsymbol{x}^T\boldsymbol{V}\boldsymbol{S}^T\boldsymbol{U}^T\boldsymbol{A}\boldsymbol{x}+\boldsymbol{x}^T\boldsymbol{V}\boldsymbol{S}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{x})\\&+\frac{\nu_1}{2}(\boldsymbol{x}^T\boldsymbol{x}-\boldsymbol{x}^T\boldsymbol{V}\boldsymbol{b}-\boldsymbol{b}^T\boldsymbol{V}^T\boldsymbol{x}+\boldsymbol{b}^T\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{b})\\&+\frac{\nu_2}{2}(\boldsymbol{x}^T\boldsymbol{V}\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{x}-2\boldsymbol{x}^T\boldsymbol{V}\boldsymbol{S}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{x}+\boldsymbol{x}^T\boldsymbol{V}\boldsymbol{S}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{x})\end{aligned} \tag{7-165}$$

式（7-165）对$U$的梯度是

$$\nabla_U \mathscr{E}(\bar{e},\hat{e},\tilde{e}) = \frac{\partial \mathscr{E}(\bar{e},\hat{e},\tilde{e})}{\partial U} = -\bar{e}r^T - v_2(U\tilde{e}r^T + f\tilde{e}^T) \tag{7-166}$$

其中

$$\tilde{e} = r - U^T f \tag{7-167}$$

和

$$f = Ur$$

因此，由式（7-151）以及式（7-166）中的梯度结果，$U$的离散时间学习规则的向量矩阵形式是

$$U(k+1) = U(k) + \mu_3\{\bar{e}(k)r^T(k) + v_2[U(k)\tilde{e}(k)r^T(k) + f(k)\tilde{e}^T(k)]\} \tag{7-168}$$

学习规则的标量形式是

$$\begin{aligned} v_{ij}(k+1) = v_{ij}(k) + \mu_1 \Bigg( & x_i(k)\left\{\left[\sum_{h=1}^{m} \bar{e}_h(k)u_{hj}(k)\right]\sigma_j(k)\right\} \\ & + v_1\left[x_i(k)\sum_{h=1}^{n}\hat{e}_h(k)v_{hj}(k) + \hat{e}_i(k)b_j(k)\right]\Bigg) \end{aligned} \tag{7-169}$$

其中$i, j = 1, 2, \cdots, n$，

324

$$\sigma_i(k+1) = \sigma_i(k) + \mu_2\left\{\left[\sum_{h=1}^{m} \bar{e}_h(k)u_{hi}(k)\right]b_i(k)\right\} \tag{7-170}$$

其中$i = 1, 2, \cdots, n$，并且

$$\begin{aligned} u_{qj}(k+1) = u_{qj}(k) + \mu_3 \Bigg( & \bar{e}_q(k)r_j(k) \\ & + v_2\left\{\left[\sum_{h=1}^{n} u_{qh}(k)\tilde{e}_h(k)\right]r_j(k) + f_q(k)\tilde{e}_j(k)\right\}\Bigg) \end{aligned} \tag{7-171}$$

其中$q = 1, 2, \cdots, m$，$j = 1, 2, \cdots, n$。三个误差项也可以写成标量形式

$$\bar{e}_q = \sum_{h=1}^{n} a_{qh}x_h - \sum_{h=1}^{n} u_{qh}r_h \quad q = 1,2,\cdots,m \tag{7-172}$$

$$\hat{e}_i = x_i - \sum_{h=1}^{n} v_{ih}b_h \quad i = 1,2,\cdots,n \tag{7-173}$$

$$\tilde{e}_i = r_i - \sum_{h=1}^{m} u_{hi}f_h \quad i = 1,2,\cdots,n \tag{7-174}$$

其中

$$r_i = \sigma_i b_i \quad \mathrm{i} = 1, 2, \cdots, n \tag{7-175}$$

$$b_i = \sum_{h=1}^{n} v_{hi}x_h \quad i = 1,2,\cdots,n \tag{7-176}$$

$$f_q = \sum_{h=1}^{n} u_{qh}r_h \quad q = 1,2,\cdots,m \tag{7-177}$$

和前面的解决特定矩阵代数问题的神经计算方法一样，对于$A$包含随时间缓慢变化的元素

的情况，SVD神经网络最适合。在这种情况下，网络的初始权值和整个网络都将由与$A$的SVD相对应的值确定，这些SVD值是由诸如LAPACK[27]等强健壮性数值算法确定的。在需要的时候，神经网络会更新适当的突触连接权值以反映矩阵$A$的元素的相应变化。

前面的结构神经网络是一种不唯一的常规体系结构。根据$A$的特殊属性，比如，是否是方阵，是否对称等等，可以设计更加简化的神经网络和学习规则。Cichocki和Unbehauen[6]的书中以方框图的形式给出了一些特殊的SVD体系结构。 325

## 7.9 求解代数李雅普诺夫方程的神经计算方法

代数李雅普诺夫方程在很多应用中都扮演着非常重要的角色（参见A.7.8节），特别是在控制理论中[28, 29]。此方程是称为西尔维斯特（Sylvester）方程的线性矩阵方程的特殊形式。西尔维斯特方程如下

$$\boldsymbol{AX}+\boldsymbol{XB}=-\boldsymbol{C} \tag{7-178}$$

其中假设已知$\boldsymbol{A}\in\Re^{n\times n}$, $\boldsymbol{B}\in\Re^{m\times m}$和$\boldsymbol{C}\in\Re^{n\times m}$，并且寻找唯一解$\boldsymbol{X}\in\Re^{n\times m}$[29]。如果$\boldsymbol{B}=\boldsymbol{A}^T$，那么$\boldsymbol{C}\in\Re^{n\times n}$，此时的方程称作李雅普诺夫方程，如下

$$\boldsymbol{AX}+\boldsymbol{XA}^T=-\boldsymbol{C} \tag{7-179}$$

其中，当且仅当$\lambda_i(\boldsymbol{A})+\lambda_j(\boldsymbol{A})\neq 0$，$\forall i,j=1,2,\cdots,n$（参见A.2.17节）时，式（7-179）有唯一解。对于式（7-179）中的李雅普诺夫方程，解$\boldsymbol{X}\in\Re^{n\times n}$和$\boldsymbol{C}\in\Re^{n\times n}$都是对称半正定矩阵，即$\boldsymbol{X}^T=\boldsymbol{X}, \boldsymbol{X}\geqslant 0$，且$\boldsymbol{C}^T=\boldsymbol{C}, \boldsymbol{C}\geqslant 0$。

神经计算方法可以用来解式（7-179）中的李雅普诺夫方程[6]。然而，当$A$矩阵中的部分（或全部）元素随时间缓慢变化时，用结构神经网络解李雅普诺夫方程的真正优点才能看出来。在这种情况下，网络的初始突触权值（即解$X$）可以置成通过用强健壮的数值方法解李雅普诺夫方程获得的值。图7-14展示了解决代数矩阵李雅普诺夫方程的结构神经网络体系结构。由方框图，误差项可写作

$$\boldsymbol{e}=\boldsymbol{d}-\boldsymbol{y}=-\boldsymbol{Cu}-\boldsymbol{AXu}-\boldsymbol{Xz} \tag{7-180}$$

其中

$$\mathrm{z}=\boldsymbol{A}^T\boldsymbol{u} \tag{7-181}$$

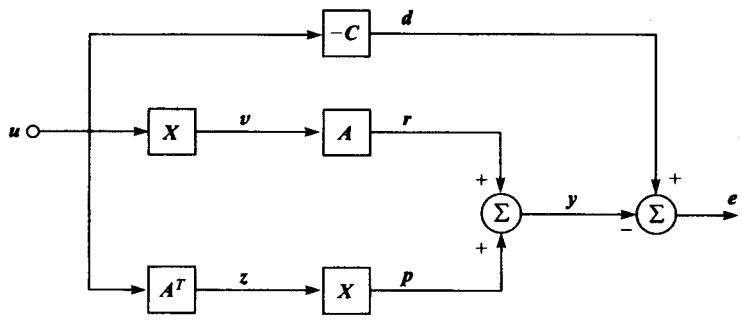



图7-14　用于代数矩阵李雅普诺夫方程求解的结构化神经网络体系结构。使用误差反向传播训练这个网络 326

把式（7-181）代入式（7-180），得

$$\boldsymbol{e}=\boldsymbol{d}-\boldsymbol{y}=-\boldsymbol{Cu}-\boldsymbol{AXu}-\boldsymbol{XA}^T\boldsymbol{u}=(-\boldsymbol{C}-\boldsymbol{AX}-\boldsymbol{XA}^T)\boldsymbol{u} \tag{7-182}$$

因此，由式（7-182）看出，如果给出一个合适的外部激励输入信号$\boldsymbol{u}$，当$\boldsymbol{X}$逼近式（7-179）中李雅普诺夫方程的解时，误差向量$\boldsymbol{e}$将趋于零。连续时间神经网络学习规则的基础是最速下降梯度法，由它推出了一组矩阵微分方程

$$\frac{\mathrm{d}\boldsymbol{X}(t)}{\mathrm{d}t}=-\mu\nabla_{\boldsymbol{X}}\mathscr{E}(\boldsymbol{X}) \tag{7-183}$$

其中，误差代价函数$\mathscr{E}(\boldsymbol{X})$是通过式（7-180）中误差向量$\boldsymbol{e}$定义的。这样，均方误差代价函数可以写作

$$\begin{aligned}\mathscr{E}(\boldsymbol{X})&=\frac{1}{2}\|\boldsymbol{e}\|_2^2=\frac{1}{2}(\boldsymbol{Cu}+\boldsymbol{AXu}+\boldsymbol{Xz})^T(\boldsymbol{Cu}+\boldsymbol{AXu}+\boldsymbol{Xz})\\&=\frac{1}{2}(\boldsymbol{u}^T\boldsymbol{C}^T\boldsymbol{Cu}+\boldsymbol{u}^T\boldsymbol{C}^T\boldsymbol{AXu}+\boldsymbol{u}^T\boldsymbol{C}^T\boldsymbol{Xz}+\boldsymbol{u}^T\boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{Cu}+\boldsymbol{u}^T\boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{AXu}\\&\quad+\boldsymbol{u}^T\boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{Xz}+\boldsymbol{z}^T\boldsymbol{X}^T\boldsymbol{Cu}+\boldsymbol{z}^T\boldsymbol{X}^T\boldsymbol{AXu}+\boldsymbol{z}^T\boldsymbol{X}^T\boldsymbol{Xz})\end{aligned} \tag{7-184}$$

为了计算式（7-183）中的梯度，即$\nabla_{\boldsymbol{X}}\mathscr{E}(\boldsymbol{X})$，需要以前用过的式（7-7）和式（7-8）中的两个一般结果，以及合适的链式规则（参见A.3.4.2节）。执行梯度计算得到

$$\begin{aligned}\nabla_{\boldsymbol{X}}\mathscr{E}(\boldsymbol{X})&=\boldsymbol{A}^T\boldsymbol{Cuu}^T+\boldsymbol{A}^T\boldsymbol{AXuu}^T+\boldsymbol{A}^T\boldsymbol{Xzu}^T+\boldsymbol{Cuz}^T+\boldsymbol{AXuz}^T+\boldsymbol{Xzz}^T\\&=\boldsymbol{A}^T(\underbrace{\boldsymbol{Cu}+\boldsymbol{AXu}+\boldsymbol{Xz}}_{-\boldsymbol{e}})\boldsymbol{u}^T+(\underbrace{\boldsymbol{Cu}+\boldsymbol{AXu}+\boldsymbol{Xz}}_{-\boldsymbol{e}})\boldsymbol{z}^T=-\boldsymbol{A}^T\boldsymbol{eu}^T-\boldsymbol{ez}^T\end{aligned} \tag{7-185}$$

因此，由式（7-183）以及式（7-185）中的梯度，可以写出连续时间学习规则

$$\frac{\mathrm{d}\boldsymbol{X}(t)}{\mathrm{d}t}=\mu[\boldsymbol{A}^T\boldsymbol{e}(t)\boldsymbol{u}^T(t)+\boldsymbol{e}(t)\boldsymbol{z}^T(t)] \tag{7-186}$$

其中$\mu>0$是学习率参数。学习规则的离散时间形式是

$$\boldsymbol{X}(k+1)=\boldsymbol{X}(k)+\mu[\boldsymbol{A}^T\boldsymbol{e}(k)\boldsymbol{u}^T(k)+\boldsymbol{e}(k)\boldsymbol{z}^T(k)] \tag{7-187}$$

如前所述，李雅普诺夫方程的解$\boldsymbol{X}$是对称的。然而，在式（7-187）中，方程右边的第二项是不对称的。这倒是不会影响学习规则计算出式（7-179）的半正定对称解。然而，相比每一步训练都强制确保对称性的式（7-187）的修改版本，这种形式的学习规则收敛得较慢。有
327 两种实施强制对称性的不同方法，相比式（7-187）它们需要相同数量的附加运算。第一种修改方法使用上面的式（7-187），然后每步训练执行一个附加运算步骤来强制确保对称性，即

$$\boldsymbol{X}(k+1)\leftarrow\frac{\boldsymbol{X}(k+1)+\boldsymbol{X}^T(k+1)}{2} \tag{7-188}$$

第二种方法首先定义$\Delta\boldsymbol{X}(k)=\boldsymbol{A}^T\boldsymbol{e}(k)\boldsymbol{u}^T(k)+\boldsymbol{e}(k)\boldsymbol{z}^T(k)$，然后把式（7-187）的学习规则重写作

$$\boldsymbol{X}(k+1)=\boldsymbol{X}(k)+\frac{\mu}{2}[\Delta\boldsymbol{X}(k)+\Delta\boldsymbol{X}^T(k)] \tag{7-189}$$

对于式（7-187）的学习规则，采用式（7-188）或式（7-189）中的方法将导致更快的收敛到解$\boldsymbol{X}$。

式（7-187）中学习规则的标量形式可写作

$$x_{ij}(k+1)=x_{ij}(k)+\mu\{e_i(k)z_j(k)+[\sum_{h=1}^{n}a_{hi}e_h(k)]u_j(k)\} \tag{7-190}$$

其中$i,j=1,2,\cdots,n$，并且

$$e_i=-\sum_{h=1}^{n}c_{ih}u_h-\sum_{h=1}^{n}a_{ih}v_h-\sum_{h=1}^{n}x_{ih}z_h \tag{7-191}$$

其中

$$v_i = \sum_{h=1}^{n} x_{ih} u_h \tag{7-192}$$

和

$$z_i = \sum_{h=1}^{n} a_{hi} u_h \tag{7-193}$$

其中$i = 1, 2, \cdots, n$。可以修改方程（7-190）借助于对称性来得到更快的收敛。修改后的学习规则为

$$\begin{aligned} x_{ij}(k+1) = x_{ij}(k) + \frac{\mu}{2}\{&e_i(k)z_j(k) + e_j(k)z_j(k) \\ &+[\sum_{h=1}^{n} a_{hi}e_h(k)]u_j(k) + [\sum_{h=1}^{n} a_{hj}e_h(k)]u_i(k)\} \end{aligned} \tag{7-194}$$

其中$i, j = 1, 2, \cdots, n$。

在Cichocki和Unbehauen[6]中，外部激励信号采用正弦信号，即$u_l(t) = \sin(l\omega_0 t)$其中$l = 1, 2, \cdots, n$。然而，线性无关双极性向量集是更好的选择，因为通常可以获得收敛速度的提高。 328

## 7.10 求解代数里卡蒂方程的神经计算方法

代数矩阵里卡蒂（Riccati）方程在最优控制和最优估计理论中都扮演重要的角色[28, 29]。此方程的标准形式为

$$\boldsymbol{A}^T\boldsymbol{X} + \boldsymbol{X}\boldsymbol{A} - \boldsymbol{X}\boldsymbol{R}\boldsymbol{X} + \boldsymbol{Q} = \boldsymbol{0} \tag{7-195}$$

这是非线性矩阵方程，因为左边第三项是非线性的。给定$\boldsymbol{A} \in \Re^{n\times n}, \boldsymbol{R} \in \Re^{n\times n}(\boldsymbol{R} > 0, \boldsymbol{R} = \boldsymbol{R}^T)$和$\boldsymbol{Q} \in \Re^{n\times n}(\boldsymbol{Q} \geqslant 0, \boldsymbol{Q} = \boldsymbol{Q}^T)$，寻找式（7-195）的解$\boldsymbol{X} \in \Re^{n\times n}(\boldsymbol{X} \geqslant 0, \boldsymbol{X} = \boldsymbol{X}^T)$。

代数矩阵里卡蒂方程的结构神经网络解法最早是由Ham和Collins[30, 31]提出的。图7-15以方框图形式显示了解决代数里卡蒂方程的结构化神经网络的体系结构。由方框图可写出误差向量$\boldsymbol{e}$如下

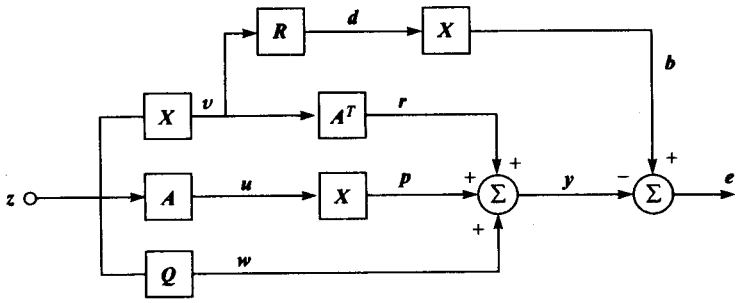



图7-15 用于代数矩阵里卡蒂方程求解的结构化网络。使用误差反向传播训练这个网络

$$\boldsymbol{e} = \boldsymbol{b} - \boldsymbol{y} = \boldsymbol{X}\boldsymbol{R}\boldsymbol{v} - \boldsymbol{A}^T\boldsymbol{X}\boldsymbol{z} - \boldsymbol{X}\boldsymbol{A}\boldsymbol{z} - \boldsymbol{Q}\boldsymbol{z} \tag{7-196}$$

其中

$$\boldsymbol{v} = \boldsymbol{X}\boldsymbol{z} \tag{7-197}$$

把式（7-197）代入式（7-196），得到

$$\boldsymbol{e} = (\boldsymbol{XRX} - \boldsymbol{A}^T\boldsymbol{X} - \boldsymbol{XA} - \boldsymbol{Q})\boldsymbol{z} \tag{7-198}$$

因此，由式（7-198）可见，对于合适的外部激励信号$\boldsymbol{z}$，当$\boldsymbol{X}$逼近式（7-195）给出的里卡蒂方程的解时，误差向量$\boldsymbol{e}$将趋于零。通过利用最速下降最优化方法和把连续时间学习规则
329 定义成如下一组矩阵微分方程，可以设计一个训练神经网络的学习规则

$$\frac{\mathrm{d}\boldsymbol{X}(t)}{\mathrm{d}t} = -\mu\nabla_{\boldsymbol{X}}\mathscr{E}(\boldsymbol{X}) \tag{7-199}$$

其中均方误差代价函数$\mathscr{E}(\boldsymbol{X})$，定义成

$$\mathscr{E}(\boldsymbol{X}) = \frac{1}{2}\|\boldsymbol{e}\|_2^2 \tag{7-200}$$

把式（7-196）代入式（7-200），得到

$$\begin{aligned}\mathscr{E}(\boldsymbol{X}) = \frac{1}{2}(&\boldsymbol{v}^T\boldsymbol{R}^T\boldsymbol{X}^T\boldsymbol{XRv} - \boldsymbol{v}^T\boldsymbol{R}^T\boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{Xz} - \boldsymbol{v}^T\boldsymbol{R}^T\boldsymbol{X}^T\boldsymbol{XAz} - \boldsymbol{v}^T\boldsymbol{R}^T\boldsymbol{X}^T\boldsymbol{Qz}\\ &-\boldsymbol{z}^T\boldsymbol{X}^T\boldsymbol{AXRv} + \boldsymbol{z}^T\boldsymbol{X}^T\boldsymbol{AA}^T\boldsymbol{Xz} + \boldsymbol{z}^T\boldsymbol{X}^T\boldsymbol{AXAz} + \boldsymbol{z}^T\boldsymbol{X}^T\boldsymbol{AQz}\\ &-\boldsymbol{z}^T\boldsymbol{A}^T\boldsymbol{X}^T\boldsymbol{XRv} + \boldsymbol{z}^T\boldsymbol{A}^T\boldsymbol{X}^T\boldsymbol{A}^T\boldsymbol{Xz} + \boldsymbol{z}^T\boldsymbol{A}^T\boldsymbol{X}^T\boldsymbol{XAz} + \boldsymbol{z}^T\boldsymbol{A}^T\boldsymbol{X}^T\boldsymbol{Qz}\\ &-\boldsymbol{z}^T\boldsymbol{Q}^T\boldsymbol{XRv} + \boldsymbol{z}^T\boldsymbol{Q}^T\boldsymbol{A}^T\boldsymbol{Xz} + \boldsymbol{z}^T\boldsymbol{Q}^T\boldsymbol{XAz} + \boldsymbol{z}^T\boldsymbol{Q}^T\boldsymbol{Qz})\end{aligned} \tag{7-201}$$

为了计算式（7-199）中的梯度，即$\nabla_{\boldsymbol{X}}\mathscr{E}(\boldsymbol{X})$，必须使用式（7-7）和式（7-8）给出的一般结果，和合适的链式规则（参见A.3.4.2节）。计算式（7-201）对$\boldsymbol{X}$的梯度，得到

$$\begin{aligned}\nabla_{\boldsymbol{X}}\mathscr{E}(\boldsymbol{X}) &= [\underbrace{\boldsymbol{XRv} - \boldsymbol{A}^T\boldsymbol{Xz} - \boldsymbol{XAz} - \boldsymbol{Qz}}_{\boldsymbol{e}}]\boldsymbol{v}^T\boldsymbol{R}^T\\ &\quad -[\underbrace{\boldsymbol{XRv} - \boldsymbol{A}^T\boldsymbol{Xz} - \boldsymbol{XAz} - \boldsymbol{Qz}}_{\boldsymbol{e}}]\boldsymbol{z}^T\boldsymbol{A}^T\\ &\quad -\boldsymbol{A}[\underbrace{\boldsymbol{XRv} - \boldsymbol{A}^T\boldsymbol{Xz} - \boldsymbol{XAz} - \boldsymbol{Qz}}_{\boldsymbol{e}}]\boldsymbol{z}^T\\ &= \boldsymbol{ev}^T\boldsymbol{R} - \boldsymbol{ez}^T\boldsymbol{A}^T - \boldsymbol{Aez}^T\end{aligned} \tag{7-202}$$

把式（7-202）中的结果代入式（7-199），得到连续时间学习规则如下

$$\frac{\mathrm{d}\boldsymbol{X}(t)}{\mathrm{d}t} = \mu[\boldsymbol{Ae}(t)\boldsymbol{z}^T(t) + \boldsymbol{e}(t)\boldsymbol{z}^T(t)\boldsymbol{A}^T - \boldsymbol{e}(t)\boldsymbol{v}^T(t)\boldsymbol{R}] \tag{7-203}$$

其中$\mu > 0$是学习率参数。学习规则的离散时间形式是

$$\boldsymbol{X}(k+1) = \boldsymbol{X}(k) + \mu[\boldsymbol{Ae}(k)\boldsymbol{z}^T(k) + \boldsymbol{e}(k)\boldsymbol{z}^T(k)\boldsymbol{A}^T - \boldsymbol{e}(k)\boldsymbol{v}^T(k)\boldsymbol{R}] \tag{7-204}$$

其中$k$是离散时间索引，即$k = 0, 1, 2\cdots$。和解李雅普诺夫方程的神经计算方法一样，式（7-204）中的更新项，即

$$\Delta\boldsymbol{X}(k) = \boldsymbol{Ae}(k)\boldsymbol{z}^T(k) + \boldsymbol{e}(k)\boldsymbol{z}^T(k)\boldsymbol{A}^T - \boldsymbol{e}(k)\boldsymbol{v}^T(k)\boldsymbol{R} \tag{7-205}$$

是不对称的，尽管如此，即使式（7-205）中的$\Delta\boldsymbol{X}(k)$不对称，式（7-204）中的学习规则仍然能收敛到半正定对称解。然而，相对于每一步训练都强制确保对称性的式（7-204）的修改版本，这种形式的学习规则收敛得较慢。有两种实施强制对称性的不同方法，相比式（7-204），
330 它们需要相同数量的附加运算。第一种修改方法使用上面的式（7-204），然后每步训练执行一个附加运算来强制确保对称性，即

$$\boldsymbol{X}(k+1) \leftarrow \frac{\boldsymbol{X}(k+1)+\boldsymbol{X}^T(k+1)}{2} \tag{7-206}$$

第二种方法是使用式（7-205）中的更新表达式来重写式（7-204）中的学习规则如下

$$\boldsymbol{X}(k+1)=\boldsymbol{X}(k)+\frac{\mu}{2}[\Delta\boldsymbol{X}(k)+\Delta\boldsymbol{X}^T(k)] \tag{7-207}$$

相比式（7-204）的学习规则，采用式（7-206）或式（7-207）中的方法将更快地收敛到解$\boldsymbol{X}$。

学习规则的标量形式（带有合适的项强制确保对称性）如下

$$\begin{aligned} x_{ij}(k+1)=x_{ij}(k)+\frac{\mu}{2}\Bigg\{&\left[\sum_{h=1}^{n}a_{ih}e_h(k)\right]z_j(k)+z_i(k)\left[\sum_{h=1}^{n}e_h(k)a_{hi}\right]\\ &+e_i(k)\left[\sum_{h=1}^{n}z_h(k)a_{hi}\right]+\left[\sum_{h=1}^{n}a_{ih}z_h(k)\right]e_j(k)\\ &-e_i(k)\left[\sum_{h=1}^{n}\upsilon_h(k)r_{hj}\right]-\left[\sum_{h=1}^{n}r_{ih}\upsilon_h(k)\right]e_j(k)\Bigg\}\end{aligned} \tag{7-208}$$

其中

$$\begin{aligned} e_i(k)=&\left[\sum_{h=1}^{n}x_{ih}(k)r_{hj}\right]\upsilon_i(k)-\left[\sum_{h=1}^{n}a_{hi}x_{hj}(k)\right]z_i(k)\\ &-\left[\sum_{h=1}^{n}x_{ih}(k)a_{hj}\right]z_i(k)-\left[\sum_{h=1}^{n}q_{ih}z_h(k)\right]\end{aligned} \tag{7-209}$$

和

$$\upsilon_i(k)=\sum_{h=1}^{n}x_{ih}(k)z_h(k) \tag{7-210}$$

其中$i, j = 1, 2, \cdots, n$。外部激励向量输入信号$z$应该采用$n$个线性无关双极值向量的集合（如前所述）。图7-16显示了解代数矩阵里卡蒂方程的多层神经网络体系结构。这个网络用误差反向传播法来训练。

**例7.4** 该例子演示了神经计算方法在如下条件下解式（7-195）中代数矩阵里卡蒂方程的能力： 331

$$\boldsymbol{A}=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -6 & -11 & -6\end{bmatrix}\quad \boldsymbol{Q}=\begin{bmatrix}1 & 0 & 0\\ 0 & 0.25 & 0\\ 0 & 0 & 0.5\end{bmatrix}\quad \boldsymbol{R}=\begin{bmatrix}10 & 0 & 0\\ 0 & 3.3333 & 0\\ 0 & 0 & 14.2857\end{bmatrix} \tag{7-211}$$

MATLAB控制系统工具箱[32]的`lqr2`函数首先用来解稳态代数矩阵里卡蒂方程。`lqr2`函数用舒尔分解方法解决线性二次调节器问题。因此，这个函数以矩阵$\boldsymbol{A}$, $\boldsymbol{Q}$ $\tilde{\boldsymbol{R}}$和$\boldsymbol{B}$为输入，其中$\tilde{\boldsymbol{R}}=\boldsymbol{R}^{-1}$（如上所述），而矩阵$\boldsymbol{R}$由$\boldsymbol{B}\tilde{\boldsymbol{R}}^{-1}\boldsymbol{B}^T$内部计算求得$\boldsymbol{B}=\boldsymbol{I}_{n\times n}$（单位矩阵）。在MATLAB中用舒尔分解方法产生的里卡蒂方程的解$\boldsymbol{X}^M$如下

$$\boldsymbol{X}^M=\begin{bmatrix}0.3324 & 0.1094 & -0.0123\\ 0.1094 & 0.3790 & -0.0059\\ -0.0123 & -0.0059 & 0.0388\end{bmatrix} \tag{7-212}$$

使用学习率参数$\mu = 0.00275$, $N = 500$（或$N \times n = 1\ 500$步迭代，其中$n = 3$），并使用初始条件$\boldsymbol{X}^{NN}(k = 0) = \boldsymbol{0}$（零矩阵），则神经网络解$\boldsymbol{X}^{NN}$产生的结果与MATLAB解一样（只取四位小数），如式（7-212）所示。外部激励输入信号采用$n$个线性无关的双极值向量集合。误差向量$\boldsymbol{e}$的三个元素分别被绘制在图7-17a中。通过如下这样与用MATLAB中`lqr2`函数得到的解相比较，解里卡蒂方程的神经网络计算方法的精确度可以量化为

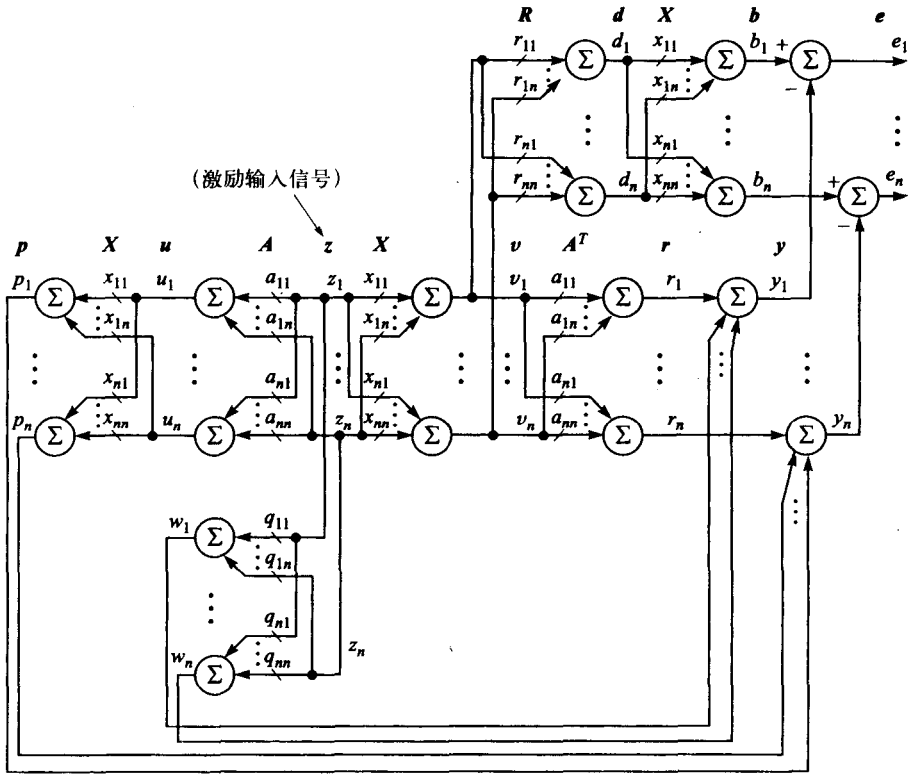



图7-16　用于求解代数矩阵里卡蒂方程的多层神经网络体系结构。元素$a_{ij}$, $q_{ij}$和$r_{ij}$，其中$i, j = 1, 2, \cdots, n$是固定的，而$x_{ij}$是自适应的，其中$i, j = 1, 2, \cdots, n$

$$\|\boldsymbol{X}^{NN} - \boldsymbol{X}^{M}\|_2 = 9.5081 \times 10^{-6} \tag{7-213}$$

现在来做个比较，使用正弦信号作为神经网络的同一个学习率参数的外部激励输入（即$\mu = 0.00275$），并使用与双极性输入同样数量的总训练步数（即$N' = N \times n = 1\ 500$）。正弦输入的一般形式以如下的离散时间形式给出

332
~
333

$$z_l(kT) = \sin(kl\omega_0 T) \tag{7-214}$$

其中$l = 1, 2, 3$，$\omega_0 = 500\text{rad/s}$，$T = 5 \times 10^{-4}\text{s}$（采样周期），$k = 0, 1, \cdots, (N' - 1)$。正弦输入时的误差向量$\boldsymbol{e}$的三个元素分别绘制在图7-17b中。图7-17b显示对于正弦输入，神经网络在1 500步训练之后仍然没有收敛，即

$$\|\boldsymbol{X}^{NN} - \boldsymbol{X}^{M}\|_2 = 0.0371 \tag{7-215}$$

正弦输入的神经网络还需要3 500步训练才能获得与双极性向量输入时相当的精度水平，见式（7-213）。

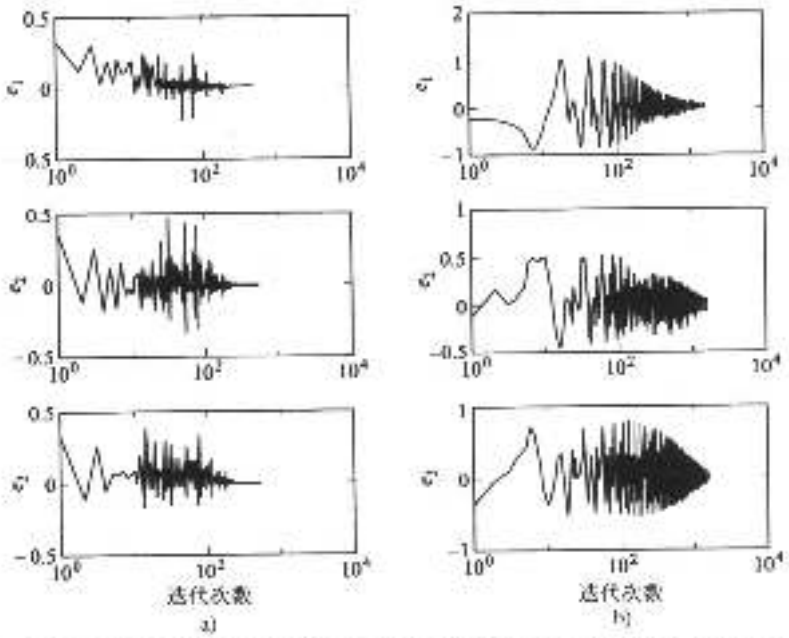



图7-17 a) 作为双极向量的外部激励输入的神经网络学习均方误差，b) 作为正弦信号的外部激励输入的神经网络学习均方误差

## 习题

7.1 对于合适维数的矩阵$A$，$B$和$C$，求证

(a) $\dfrac{\partial}{\partial A}\text{trace}(BAC)=B^TC^T$

(b) $\dfrac{\partial}{\partial A}\text{trace}(BA^TC)=CB$

7.2 写一个电脑程序来实现在式（7-9）中以向量矩阵（批量）形式给出的神经网络学习规则，即计算矩阵伪逆的方法1，并计算以下矩阵的伪逆：

(a) $A=\begin{bmatrix}-1 & 2 & 3\\ 4 & 5 & 6\\ 4 & -1 & -1\end{bmatrix}$　　(b) $A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}$

(c) $A=\begin{bmatrix}-1 & 2 & 3 & 5\\ 2 & -4 & 6 & 3\\ 2 & -3 & 1 & 7\end{bmatrix}$　　(d) $A=\begin{bmatrix}-1 & 2 & 3\\ 2 & -4 & 6\\ 2 & -3 & 1\\ 3 & 5 & -7\end{bmatrix}$

在每种情况下根据式（7-16）的表达式绘出误差代价函数，定义一个终止训练过程的停止标准。把你的结果与用MATLAB中pinv函数计算得到的伪逆相比较。

7.3 重复计算问题7.2（a）和（b），用方法2计算矩阵的伪逆。实现式（7-31）中给出的神经网络学习规则的向量矩阵（批量）形式。使用双极性向量作为结构神经网络的外部激励输入。在两种情况下，根据式（7-26）的表达式绘出误差代价函数，定义一个终止训练过程的停止标准。重复这个问题，但是如同式（2-36）那样使用搜索然后收敛调度来调

整学习率参数，而不是使用一个固定的学习率参数。

7.4　使用图7-4所示的结构神经网络，计算下列矩阵的LU分解：

(a) $\boldsymbol{A}=\begin{bmatrix}1 & 2 & -7\\ -2 & 4 & 5\\ -1 & -1 & 4\end{bmatrix}$　　(b) $\boldsymbol{A}=\begin{bmatrix}0 & 0 & 4\\ 3 & 2 & -6\\ -1 & 2 & -1\end{bmatrix}$

334　使用式（7-49）和式（7-53）学习规则的向量矩阵（批量）形式。采用式（7-45）表达式绘出误差代价函数来监视训练过程，并且定义一个终止训练过程的停止标准。

7.5　使用计算矩阵QR因子分解方法1的结构神经网络（图7-6），求下面矩阵的因子分解

$$\boldsymbol{A}=\begin{bmatrix}1 & 4 & -3\\ 4 & 5 & 2\\ 5 & -3 & 2\end{bmatrix}$$

实现式（7-73）和式（7-81）的学习规则的向量矩阵形式。对于两种不同的外部激励输入，用总的训练回合数与训练时间相比较。第一种输入使用双极性向量来训练神经网络。第二种输入使用离散时间的正弦函数来重新训练神经网络。正弦输入的离散时间形式应该采用$x_l(kT)=\sin(kl\omega_0 T)$，其中$l=1,2,3$，$\omega_0=500\text{rad/s}$，$T=1\times10^{-4}\text{s}$（采样周期）和$k=0,1,\cdots,(N-1)$，其中$N$是训练总回合数。定义一个终止训练过程的停止标准。

7.6　图7-18以方框图的形式显示了一个计算矩阵$\boldsymbol{A}\in\Re^{n\times n}$的舒尔分解的结构神经网络（参见7.5节）。和7.5节一样，定义两个误差向量$\bar{\boldsymbol{e}}=\boldsymbol{Ax}-\boldsymbol{QRb}$和$\hat{\boldsymbol{e}}=\boldsymbol{x}-\boldsymbol{Qb}$，定义总误差代价函数$\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}})=1/2\|\bar{\boldsymbol{e}}\|_2^2+\nu/2\|\hat{\boldsymbol{e}}\|_2^2$，定义离散时间学习规则的向量矩阵形式为$\boldsymbol{Q}(k+1)=\boldsymbol{Q}(k)-\mu\nabla_{\boldsymbol{Q}}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}})$和$\boldsymbol{R}(k+1)=\boldsymbol{R}(k)-\mu\nabla_{\boldsymbol{R}}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}})$。通过计算必要的梯度项$\nabla_{\boldsymbol{Q}}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}})$和$\nabla_{\boldsymbol{R}}\mathscr{E}(\bar{\boldsymbol{e}},\hat{\boldsymbol{e}})$来推导求$\boldsymbol{Q}$和$\boldsymbol{R}$的两条学习规则。

7.7　在7.4节提到的计算矩阵$\boldsymbol{A}\in\Re^{n\times n}$的QR因子分解的方法2中，两个误差向量定义成$\bar{\boldsymbol{e}}=\boldsymbol{Ax}-\boldsymbol{Qu}$和$\hat{\boldsymbol{e}}=\boldsymbol{u}-\boldsymbol{Q}^T\boldsymbol{y}$。基于之前方法1使用的相同总误差代价函数和学习规则的最速下降梯度形式推导计算$\boldsymbol{Q}$和$\boldsymbol{R}$的两个学习规则的离散时间向量矩阵形式。提示：当推导$\boldsymbol{R}$的学习规则时，在计算梯度前，把$\boldsymbol{u}=\boldsymbol{Rx}$代入总误差代价函数。

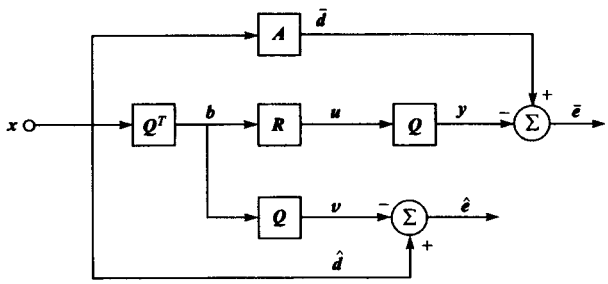



图7-18　7.5节中用于舒尔分解的结构化神经网络方框图

335

7.8　对于图7-9（在图7-18中重复）中的方框图给出的舒尔分解结构神经网络，写一个计算机程序实现式（7-96）和式（7-97）相应学习规则的向量矩阵形式。设$\nu=1$，并确保对$\boldsymbol{R}$矩阵加上上三角限制。使用双极值向量作为网络的外部激励输入，计算下列矩阵的舒尔分解（即计算$\boldsymbol{Q}$和$\boldsymbol{R}$）

(a) $\boldsymbol{A}=\begin{bmatrix}0.7562 & 0.3750 & -2.3775\\ 0.4005 & 1.1252 & -0.2738\\ -1.3414 & 0.7286 & -0.3229\end{bmatrix}$　(b) $\boldsymbol{A}=\begin{bmatrix}-1 & -4 & 0\\ -4 & 5 & 0\\ 0 & 0 & 0\end{bmatrix}$

通过用式（7-93）的表达式绘出总误差代价函数来监视训练过程中的均方误差，并定义

一个终止训练过程的停止标准。

7.9 写计算机程序实现式（7-118）和式（7-119）给出的计算对称矩阵特征值和特征向量的结构神经网络（图7-11）学习规则的向量矩阵形式。使用高斯随机向量作为网络的外部激励输入，计算下列矩阵的特征值和特征向量：

(a) $\boldsymbol{A}=\begin{bmatrix}1 & 0.5 & -1\\ 0.5 & 2 & 1\\ -1 & 1 & 3\end{bmatrix}$ (b) $\boldsymbol{A}=\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}$ (c) $\boldsymbol{A}=\begin{bmatrix}-1 & -1\\ -1 & -3\end{bmatrix}$

(d) $\boldsymbol{A}=\begin{bmatrix}-1.2384 & 1.1546 & -0.4880 & 0.4191\\ 1.1546 & -0.6638 & -0.8822 & 0.0437\\ -0.4880 & -0.8822 & 1.0290 & -0.3905\\ 0.4191 & 0.0437 & -0.3905 & -0.1419\end{bmatrix}$ (e) $\boldsymbol{A}=\begin{bmatrix}1 & -1 & 1\\ -1 & 1 & -1\\ 1 & -1 & 1\end{bmatrix}$

把你的结果与MATLAB中的特征值程序，即eig函数作比较。

7.10 所有的实方阵$\boldsymbol{A}\in\Re^{n\times n}$都可以因子分解成$\boldsymbol{A}=\boldsymbol{QH}$，其中$\boldsymbol{Q}\in\Re^{n\times n}$是正交的($\boldsymbol{QQ}^T=\boldsymbol{Q}^T\boldsymbol{Q}=\boldsymbol{I}$)，$\boldsymbol{H}\in\Re^{n\times n}$是对称($\boldsymbol{H}^T=\boldsymbol{H}$)半正定($\boldsymbol{H}>0$)矩阵。如果$\boldsymbol{A}$是可逆的[$\rho(\boldsymbol{A})=n$]，则$\boldsymbol{H}$是正定的($\boldsymbol{H}>0$)。这称为$\boldsymbol{A}$的极分解[8]。

证明上述陈述。

7.11 对于式（7-158）、式（7-164）和式（7-168）的计算矩阵奇异值分解的结构神经网络（图7-13），写一个计算机程序实现它的学习规则的向量矩阵形式。通过计算下列矩阵的SVD来测试你的程序。

$$\boldsymbol{A}=\begin{bmatrix}1 & 0 & 0\\ -1 & 2 & -1\\ 0 & 0 & 3\end{bmatrix}$$

7.12 在Cichocki和Unbehauen[6]中，给出了8个计算方阵$\boldsymbol{A}\in\Re^{n\times n}$的奇异值分解的结构神经网络体系结构。其中之一如图7-19中的方框图所示。基于最速下降梯度法推导三个计算SVD离散时间学习规则，使得$\boldsymbol{U}^T\boldsymbol{A}=\boldsymbol{SV}^T$成立，其中$\boldsymbol{V}^T\boldsymbol{V}=\boldsymbol{I}$和$\boldsymbol{UU}^T=\boldsymbol{I}$。把总误差代价函数定义成与式（7-148）相似。

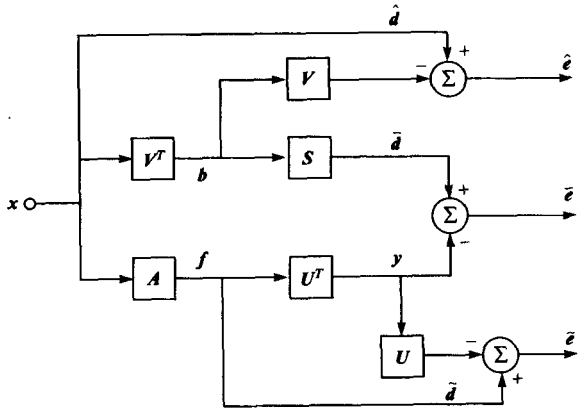



图7-19 计算方阵SVD的另一个结构化神经网络体系结构

7.13 用图7-13（7.8节）中方框图给出的相同结构神经网络来计算矩阵$\boldsymbol{A}\in\Re^{m\times n}$, $m\geqslant n$的SVD，定义三个误差向量为$\bar{\boldsymbol{e}}=(\boldsymbol{A}-\boldsymbol{USV}^T)\boldsymbol{x}$, $\hat{\boldsymbol{e}}=(\boldsymbol{I}-\boldsymbol{VV}^T)\boldsymbol{x}$和$\tilde{\boldsymbol{e}}=(\boldsymbol{I}-\boldsymbol{U}^T\boldsymbol{U})\boldsymbol{r}$，其中$\boldsymbol{U}\in\Re^{m\times n}$ ($\boldsymbol{U}^T\boldsymbol{U}=\boldsymbol{I}$), $\boldsymbol{V}\in\Re^{n\times n}$ ($\boldsymbol{VV}^T=\boldsymbol{I}$), $\boldsymbol{S}=\text{diag}[\sigma_1,\sigma_2,\cdots,\sigma_n]\in\Re^{n\times n}$。使用上述三个误差向量的表达式，式（7-148）给出的总误差代价函数和分别由式（7-149）、式（7-150）和式（7-151）给出的求$\boldsymbol{V}$, $\boldsymbol{S}$和$\boldsymbol{U}$的三个学习规则的离散时间向量矩阵形式，推导三个新的学习规则。把这些$\boldsymbol{V}$, $\boldsymbol{S}$和$\boldsymbol{U}$的新的学习规则与7.8节推导的式（7-158）、式（7-164）和式（7-168）中的学习规则分别比较。

7.14　对于问题7.10讨论的矩阵的极分解，推导使用结构神经网络方法计算$\boldsymbol{Q}$和$\boldsymbol{H}$的离散时间学习规则（用向量矩阵形式）。计算$\boldsymbol{Q}$和$\boldsymbol{H}$的学习规则应基于最速下降梯度法，且应具有一般的离散时间形式$\boldsymbol{Q}(k+1)=\boldsymbol{Q}(k)-\mu\nabla_{\boldsymbol{Q}}\mathscr{E}(\bar{e},\hat{e})$和$\boldsymbol{H}(k+1)=\boldsymbol{H}(k)-\mu\nabla_{\boldsymbol{H}}\mathscr{E}(\bar{e},\hat{e})$，其中$\bar{e}$和$\hat{e}$是合理定义的误差向量。通过构造含有外部激励输入和两个误差向量$\bar{e}$和$\hat{e}$的合理方框图来设计结构神经网络。应该使用的总误差代价函数是$\mathscr{E}(\bar{e},\hat{e})=\mathscr{E}[\bar{e}(\boldsymbol{Q},\boldsymbol{H}),\hat{e}(\boldsymbol{Q},\boldsymbol{H})]=1/2\|\bar{e}\|_2^2+\nu/2\|\hat{e}\|_2^2$。通过一系列计算机仿真来确定一组将产生最好的收敛的外部激励信号。尝试二值向量，双极值向量，正弦输入信号和高斯白噪声。通过寻找下列矩阵的极分解来测试你的计算机程序。

337

(a) $\boldsymbol{A}=\begin{bmatrix}1&2&3\\5&6&7\\7&8&9\end{bmatrix}$　　(b) $\boldsymbol{A}=\begin{bmatrix}0&1&0\\0&0&1\\-6&-11&-6\end{bmatrix}$

提示：此结构神经网络应该是7.4节中方法1的QR因子分解网络的修改版本。

7.15　设计一个能对对称矩阵$\boldsymbol{A}\in\Re^{n\times n}(\boldsymbol{A}^T=\boldsymbol{A})$进行下列因子分解的结构神经网络

(a) $\boldsymbol{A}=\boldsymbol{QTQ}^T$　　(b) $\boldsymbol{A}=\boldsymbol{LDL}^T$

其中$\boldsymbol{Q}\in\Re^{n\times n}$是正交矩阵，$\boldsymbol{T}\in\Re^{n\times n}$是对称三对角矩阵，$\boldsymbol{L}\in\Re^{n\times n}$是下三角矩阵，$\boldsymbol{D}\in\Re^{n\times n}$是对角矩阵。用问题7.9（a）、（c）、（e）中的矩阵来测试你的计算机程序（使用学习规则的向量矩阵形式）。

7.16　写一个计算机程序实现7.9节中讨论的解李雅普诺夫方程的学习规则的离散时间向量矩阵形式。用下列矩阵测试你的计算机程序：

(a) $\boldsymbol{A}=\begin{bmatrix}0&1&0\\0&0&1\\-1&-2&-2\end{bmatrix}$, $\boldsymbol{C}=\begin{bmatrix}2&1&0\\1&1&1\\0&1&1\end{bmatrix}$

(b) $\boldsymbol{A}=\begin{bmatrix}1&2&3\\6&5&4\\7&0&8\end{bmatrix}$, $\boldsymbol{C}=\begin{bmatrix}7&2&4\\2&5&1\\4&1&6\end{bmatrix}$

使用例7.4描述的双极性向量和正弦函数作为网络的外部激励输入，并比较两组结果。

7.17　写一个计算机程序实现7.10节中讨论的解里卡蒂方程的学习规则的离散时间向量矩阵形式。用下列矩阵测试你的计算机程序：

$$\boldsymbol{A}=\begin{bmatrix}0&1&0\\0&0&1\\-6&-11&-6\end{bmatrix}\quad\boldsymbol{Q}=\begin{bmatrix}2&0&0\\0&0.25&0\\0&0&0.5\end{bmatrix}$$

$$\boldsymbol{R}=\begin{bmatrix}0.1&0&0\\0&0.3&0\\0&0&0.07\end{bmatrix}^{-1}$$

把结果与解决线性二次正则问题的MATLAB控制系统工具箱函数lqr2的结果比较。这个函数用舒尔分解来解里卡蒂方程。如例7.4中所讨论的，必须向MATLAB函数lqr2提供矩阵$\boldsymbol{B}$和$\boldsymbol{R}$。可以在MATLAB函数lqr2中把加权矩阵$\boldsymbol{R}$定义成$\tilde{\boldsymbol{R}}$，然后用两种方法比较结果，在lqr2 MATLAB函数中令$\boldsymbol{B}=\boldsymbol{I}_{n\times n}$，$\tilde{\boldsymbol{R}}=R^{-1}$。使用双极性向量作为网

络的外部激励输入。还使用例7.4所述的正弦函数作为外部激励输入，比较两个结果。 338

7.18 重复问题7.17，要求计算机程序每次计算里卡蒂方程解$\boldsymbol{X}$的一列。

7.19 在数字控制理论中，离散时间李雅普诺夫方程以$\boldsymbol{\Phi X \Phi}^T + \boldsymbol{C} = \boldsymbol{X}$形式给出，其中矩阵$\boldsymbol{\Phi} \in \Re^{n \times n}$, $\boldsymbol{C} \in \Re^{n \times n}$已知，$\boldsymbol{X} \in \Re^{n \times n}$是方程的待求解。设计一个合适的结构神经网络来计算离散时间李雅普诺夫方程的解。确定网络应使用的最好的外部激励输入信号。

7.20 设计一个结构神经网络来解7.9节中给出的西尔维斯特方程，即$\boldsymbol{AX} + \boldsymbol{XB} = -\boldsymbol{C}$，其中假设已知$\boldsymbol{A} \in \Re^{n \times n}$, $\boldsymbol{B} \in \Re^{m \times m}$, $\boldsymbol{C} \in \Re^{n \times m}$，寻找解$\boldsymbol{X} \in \Re^{n \times m}$。

7.21 对于有相异特征值（即没有重复的特征值）矩阵$\boldsymbol{A} \in \Re^{n \times n}$，存在一个相似变换（由$\boldsymbol{A}$的相应特征向量组成）能够把$\boldsymbol{A}$对角化（参见7.6节）。然而，如果$\boldsymbol{A}$有重复的特征值，$\boldsymbol{A}$就不与对角矩阵相似，除非$\boldsymbol{A}$有一个独立的特征向量完全集合。如果这些特征向量不独立，则$\boldsymbol{A}$称为亏损的（defective）[2]。对这些矩阵，广义特征值问题用于确定方程$\boldsymbol{A}\upsilon_i = \lambda_i \boldsymbol{B}\upsilon_i$, $i = 1, 2, \cdots, n$的非平凡解。除了$\boldsymbol{A}$以外，矩阵$\boldsymbol{B} \in \Re^{n \times n}$给定，$\lambda_i \in \Re$是广义特征值，$\upsilon_i \in \Re^{n \times 1}$是相应的广义右特征向量，其中$i = 1, 2, \cdots, n$。很容易看出$\lambda$是特征方程$\det(\boldsymbol{A} - \lambda \boldsymbol{B}) = 0$的根。（注意：如果$\boldsymbol{B} = \boldsymbol{I}_{n \times n}$，则特征方程是求解标准特征值问题。）矩阵$\boldsymbol{A} - \lambda \boldsymbol{B}$称作矩阵束（matrix pencil）[20]。QZ算法[2, 20]是解决广义特征值问题的标准方法。与有相异特征值的情况类似，广义特征值和特征向量满足因子分解$\boldsymbol{AV} = \boldsymbol{BVA}$，其中$\boldsymbol{A} = \text{diag}[\lambda_1, \lambda_2, \cdots, \lambda_n]$，$\boldsymbol{V} = [\upsilon_1, \upsilon_2, \cdots, \upsilon_n]$。设计一个神经网络解决广义特征值问题，即计算$\lambda_i$和$\upsilon_i$。

## 参考文献


1. L.-X. Wang and J. M. Mendel, "Structured Trainable Networks for Matrix Algebra," *Proceedings of the International Joint Conference on Neural Networks*, San Diego, CA, vol. 2, 1990, pp. 125–32.
2. G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Baltimore, MD: Johns Hopkins, 1996.
3. P. Baldi and K. Hornik, "Neural Networks and Principal Component Analysis: Learning from Examples without Local Minima," *Neural Networks*, vol. 2, 1989, pp. 53–8.
4. L.-X. Wang and J. M. Mendel, "Parallel Structured Networks for Solving a Wide Variety of Matrix Algebra Problems," *Journal of Parallel and Distributed Computing*, vol. 14, 1992, pp. 236–47.
5. M. M. Polycarpou and P. A. Ioannou, "Learning and Convergence Analysis of Neural-Type Structured Networks," *IEEE Transactions on Neural Networks*, vol. 3, 1992, pp. 39–50.
6. A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, New York: Wiley, 1993. 339
7. J.-S. Jong, S.-Y. Lee and S.-Y. Skin, "An Optimization Network for Matrix Inversion," *Neural Information Processing Systems*, ed. D. Z. Anderson, College Park, MD: American Institute of Physics, 1988, pp. 397–401.
8. G. Strang, *Introduction to Linear Algebra*, Wellesley, MA: Wellesley-Cambridge Press, 1993.
9. D. S. Watkins, *Fundamentals of Matrix Computations*, New York: Wiley, 1991.
10. S. J. Orfanidis, *Optimum Signal Processing: An Introduction*, 2nd ed., New York: McGraw-Hill, 1988.
11. C. T. Chen, *Linear System Theory and Design*, New York: Holt, Rinehart and Winston, 1984.
12. S. Haykin, *Adaptive Filter Theory*, 3rd ed., Upper Saddle River, NJ: Prentice-Hall, 1996.

13. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed., New York: Cambridge University Press, 1992.
14. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P Flannery, *Numerical Recipes in Fortran: The Art of Scientific Computing*, 2nd ed., New York: Cambridge University Press, 1992.
15. C. W. Therrien, *Discrete Random Signals and Statistical Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1992.
16. B. D. O. Anderson and J. B. Moore, *Optimal Filtering*, Englewood Cliffs, NJ: Prentice-Hall, 1979.
17. G. W. Stewart, *Introduction to Matrix Computations*, New York: Academic, 1973.
18. G. Strang, *Linear Algebra and Its Applications*, 2nd ed., New York: Academic, 1980.
19. A. Cichocki and R. Unbehauen, "Neural Networks for Computing Eigenvalues and Eigenvectors," *Biological Cybernetics*, vol. 68, 1992, pp. 155–64.
20. B. N. Datta, *Numerical Linear Algebra and Applications*, Pacific Grove, CA: Brooks/Cole, 1995.
21. R. S. Varga, *Matrix Iterative Analysis*, Englewood Cliffs, NJ: Prentice-Hall, 1962.
22. S. G. Nash and A. Sofer, *Linear and Nonlinear Programming*, New York: McGraw-Hill, 1996.
23. D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd ed., Reading, MA: Addison-Wesley, 1984.
24. N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 1996.
25. H. Bourlard and Y. Camp, "Auto-Association by Multilayer Perceptrons and Singular Value Decomposition," *Biological Cybernetics*, vol. 59, 1988, pp. 291–4.
26. J. A. Sirat, "A Fast Algorithm for Principal Component Analysis and Singular Value Decomposition," *International Journal of Neural Systems*, vol. 2, nos. 1 and 2, 1991, pp. 147–55.
27. E. Anderson, Z. Bai, C. H. Bischof, J. W. Demmel, J. J. Dongarra, J. J. Du Croz, A. Greenbaum, S. J. Hammarling, A. McKenney, S. Ostrouchov, and D. C. Sorensen, *LAPACK Users' Guide, Release 2.0*, 2nd ed., Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 1995.
28. H. Kwakernaak and R. Sivan, *Linear Optimal Control Systems*, New York: Wiley-Interscience, 1972.
29. K. Zhou, J. C. Doyle, and K. Glover, *Robust and Optimal Control*, Upper Saddle River, NJ: Prentice-Hall, 1996.
30. F. M. Ham and E. Collins, "A Neural Network Architecture for Solving the Algebraic Matrix Riccati Equation," *Proceedings of the SPIE International Conference on Aerospace/Sensing and Controls (Applications and Science of Artificial Neural Networks II)*, eds. S. K. Rogers and D. W. Ruck, Orlando, FL, SPIE vol. 2760, 1996, pp. 294–301.
31. F. M. Ham and E. Collins, "A Neurocomputing Approach for Solving the Algebraic Matrix Riccati Equation," *Proceedings of the IEEE International Conference on Neural Networks*, Washington, vol. 1, 1996, pp. 617–22.
32. A. Grace, A. J. Laub, J. N. Little, and C. M. Thompson, *Control System Toolbox—For Use with MATLAB*, Natick, MA: The Mathworks, Inc., 1993.

340

341

# 第8章 使用神经网络求解线性代数方程组

## 8.1 概述

在科学与工程中遇到的许多问题都要求解线性代数方程组，比如，在信号处理和机器人学问题上。原则上讲，求解方程组与矩阵求逆（或伪逆）是等效的（参考A.2.7节）。本章打算扩展第7章的概念以求解方程组，主要兴趣在于实时和在线处理策略的研究上。并不打算将本章中提出的方法与离线求解方程组的标准数值方法相竞争[1]。已经存在许多出色的数值方法。通常，时间约束对离线求解方程组并不重要，一个好的解却是极为重要的。然而，如果必须重复在线或实时地求解方程组，如果求解这些方程的时间限制比一台典型的数字计算机所能提供的更苛刻，那么就必须探索其他方法了。一种可能的求解方法是使用脉动（systolic）（或波阵面（wavefront））阵列[2-5]。另一种方式是使用模拟人工神经网络，那是因为它们具有内在的并行结构[6-8]。

我们的目标是为线性方程组的在线求解开发各种学习策略。这里的基本原理与第7章中利用神经网络求解特定矩阵代数问题是相似的。对于那些当特定的系统参数随时间缓慢变化时必须重复求解的系统，使用神经计算方法求解线性方程组具有计算优势（正是由于参数随时间的缓慢变化所以方程组必须重复性的处理）。由于系统参数的变化小，“新”求解方法同 342
“旧”的方法相差不多。因此，几乎没有“学习”步骤需要开发以便于形成新的求解方法。这里，我们将不关注这些系统的最终神经体系结构的硬件实现深层问题。很明显这是一个非常重要的课题，在实现一个特殊的系统时必须关注，因此，我们把它留给了其他的相关主题的优秀资源[比如，6-8]。

## 8.2 联立线性代数方程组

考虑一个带有常数系数的线性代数方程组

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{aligned} \tag{8-1}$$

已知系数$a_{ij}$以及$b_i$ ($i = 1, 2, \cdots, m$; $j = 1, 2, \cdots, n$) 的情况下，求解未知量$x_1, x_2, \cdots, x_n$[9-24]。方程组（8-1）可以写成一种更方便（紧凑）的形式，也就是向量矩阵的形式，如下

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{8-2}$$

其中假设$\boldsymbol{A} \in \Re^{m \times n}$, $\boldsymbol{x} \in \Re^{n \times 1}$, $\boldsymbol{b} \in \Re^{m \times 1}$，并且

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \tag{8-3}$$

$$\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T \tag{8-4}$$

$$\boldsymbol{b} = [b_1, b_2, \cdots, b_m]^T \tag{8-5}$$

这里会出现三种情况：(1) 如果$m > n$，这种情况很普遍（方程的个数多于未知量），这种方程组称作超定方程组[25]。(2) 如果$n > m$（未知量的个数多于方程数），此时方程组称作欠定方程组[25]。(3) 当$m = n$，方程的数目与未知量的数目一致（也就是确定方程组）。矩阵$\boldsymbol{A}$有时称作数量矩阵，向量$\boldsymbol{b}$有时称作观察向量[25]。

假设最简单的例子，构建一组线性方程，考虑$xy$平面的两个直线方程，例如，

343

$$\begin{aligned} y &= -2x + 2 \\ y &= x \end{aligned} \tag{8-6}$$

问题是：是否存在$xy$平面的一个公共点在两条直线上。或者等效地，是否存在$xy$平面的两条直线的交点？对这个问题的回答是建立一组代数方程，然后求解。可以把方程（8-6）重写为如下形式

$$\begin{aligned} 2x + y &= 2 \\ -x + y &= 0 \end{aligned} \tag{8-7}$$

或者写成向量矩阵的形式

$$\begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \tag{8-8}$$

这是一个完全的确定性系统。即矩阵$\boldsymbol{A}$是一个方阵。解这个代数方程组（需要求矩阵$\boldsymbol{A}$的逆），将会得到两条直线在$xy$平面的交点如下

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{-1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix} \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{2}{3} \\ \frac{2}{3} \end{bmatrix} \tag{8-9}$$

而此结果显示当在$xy$平面绘制两条直线时，可以看到相应的交点如图8-1所示。由于两条直线有一个交点，所以方阵$\boldsymbol{A}$的逆存在。假如两条直线之间不存在交点，（即，它们是一对平行线）方阵$\boldsymbol{A}$就是奇异的。

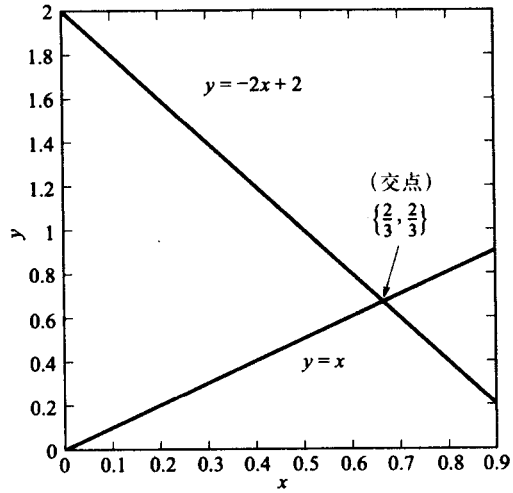



图8-1　两条相交的直线

我们真正感兴趣的系统类型比这里举例的情况复杂得多。我们将要研究大规模的超定、欠定、病态系统，以及具有不确定性的系统。 344

## 8.3 线性方程组的最小二乘解

假设在式（8-2）中的一般线性方程组中，当$m \geqslant n$（超定系统），将首先考虑满秩最小二乘问题[25]。换句话说，假定矩阵$\boldsymbol{A}$满秩⊖，其秩为$\rho(\boldsymbol{A}) = n$，求解$\boldsymbol{x}$。为了达到目的，尽力在保证选取合适的$p$情况下，最小化标量代价函数

$$\mathscr{E}(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p \tag{8-10}$$

基于不同的$p$选择（即，不同的范数），就会产生不同的最优结果。如果使用1范数（$L_1$范数）和∞范数（$L_\infty$范数）（或Chebyshev范数）（参照A.2.13节），式（8-10）的最小化是困难的，因为对于这些$p$值，函数$\mathscr{E}(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p$不可微[26]。当$p = 1$时，称作*最小绝对偏差问题*。对于$p = 2$，称作*线性最小二乘问题*。而当$p = \infty$时，称作*最小最大问题*。式（8-10）中所用的范数类型很大程度上取决于数据误差的分布和[18-20]中将要提及的应用类型。对于当误差是双指数分布（或拉普拉斯（Laplace）分布）[27, 28]时，可以用$L_1$范数。当误差分布具有明显的剧烈的跃迁时，例如均匀分布，那么应该用$L_\infty$范数（或Chebyshev范数）。但是，如果误差分布是高斯分布（正态分布），此时，最好使用欧几里得范数（或$L_2$范数）。使用欧几里得范数，得到最小二乘问题

$$\min_{\forall \boldsymbol{x}} \mathscr{E}(\boldsymbol{x}) = \min_{\forall \boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 \tag{8-11}$$

相比其他情况更易于处理。如果$\boldsymbol{A}$是满秩的，式（8-2）就会有唯一解[25]。一般情况下，标量代价函数写作

$$\mathscr{E}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 = \frac{1}{2}\|\boldsymbol{e}\|_2^2 \tag{8-12}$$

其中，

$$\boldsymbol{e} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \tag{8-13}$$

是解的误差向量（参照A.2.7节）。计算式（8-12）对于向量$\boldsymbol{x}$的梯度，并设结果等于0，也就是$\nabla_{\boldsymbol{x}} \mathscr{E}(\boldsymbol{x}) = 0$，可以推导出以下正规方程组：

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^T\boldsymbol{b} = \boldsymbol{0} \tag{8-14}$$

直接由式（8-14）可解出$\boldsymbol{x}$：

$$\boldsymbol{x} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b} \tag{8-15}$$ 345

其中

$$\boldsymbol{A}^+ = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T \tag{8-16}$$

定义为$\boldsymbol{A}$的*伪逆*（参见A.2.7节）。式（8-15）的解是式（8-2）问题的“批量”解。如果$m<n$（欠定问题）和$\rho(\boldsymbol{A}) = m$（即$\boldsymbol{A}$为满秩），方程（8-2）的解为

$$\boldsymbol{x} = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1}\boldsymbol{b} \tag{8-17}$$

⊖ 当我们用神经计算方法求解线性方程组时，这个限制实际上并不需要。这一点的详细讨论会在适当时间指出。

其中

$$A^{+} = A^{T}(AA^{T})^{-1} \tag{8-18}$$

为$A$的伪逆。

## 8.4 求解线性方程组的最小二乘神经计算方法

首先假定需要求取式（8-2）的解，但不是基于批处理方法而是基于一种能够“学习”求解的方法。一种直接的方法是使用最速梯度下降的方法。将沿用第7章中的方法。连续时间（模拟）的学习规则给定如下

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = -\mu\nabla_{x}\mathscr{E}(\boldsymbol{x}) \tag{8-19}$$

其中$\boldsymbol{\mu}\in\Re^{n\times n}(\boldsymbol{\mu}=[\mu_{ij}], i, j = 1, 2, \cdots, n)$是一个正定矩阵，一般情况下选择为对角形矩阵。也就是，$\mu_{ij} = \mu_j\delta_{ij}$，其中$\mu_j > 0$，是Kronecker（克罗内克）Δ（参照A.2.8节）并且$\boldsymbol{x}(0) = \boldsymbol{x}_0$。式（8-19）的梯度已经计算过，是式（8-14）的左侧。将以上求得的梯度代入式（8-19），从而得到解如下方程组的学习规则：

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = -\mu(\boldsymbol{A}^{T}\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^{T}\boldsymbol{b}) = -\mu\boldsymbol{A}^{T}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \tag{8-20}$$

在A.2.7节，定义求解误差为

$$\boldsymbol{e} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \tag{8-21}$$

将式（8-21）代入式（8-20），得到式（8-2）中求解线性方程组的连续时间学习规则

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = -\mu\boldsymbol{A}^{T}\boldsymbol{e} \tag{8-22}$$

其中要求初始条件$\boldsymbol{x}(0) = \boldsymbol{x}_0$。方程（8-21）和式（8-22）一起足够求解任何方程组，不论方程组是完全确定的（$m = n$），超定的（$m > n$），和欠定的（$m < n$，或者$\boldsymbol{A}$是不满秩的，甚至$\boldsymbol{A}$是

346 病态的。很多时候使用正规化的矩阵方程$\hat{\boldsymbol{A}}\boldsymbol{x} = \hat{\boldsymbol{b}}$是很方便的，其中有$\hat{a}_{ij} = \dfrac{a_{ij}}{\|\boldsymbol{a}_i^{T}\|_2}$，且$\hat{b}_i = \dfrac{b_i}{\|\boldsymbol{a}_i^{T}\|_2}$（$i = 1, 2, \cdots, m$，$j = 1, 2, \cdots, n$）。因此，$\boldsymbol{a}_i$是矩阵$\boldsymbol{A}$的第$i$行，$\|\cdot\|_2$是$L_2$范数（或欧几里得范数）。

通过与第7章中开发结构化神经网络相似的过程，可以形成一个神经体系结构以实现用于求解线性方程组的最小二乘神经计算方法。图8-2是一个微分系统的方框图，图中显示了神经计算方法的模拟系统实现的基础结构，而图8-3显示神经网络架构的细节。这是一个三层网络结构，输出带有一系列的积分器。积分器的初始条件是与式（8-22）的连续时间学习规则相关的初始条件。反馈连接把观察向量的元素一起“输入”给网络。如果积分器（可用的放大器）在饱和区域工作，那么反馈信号必须经过一个S形非线性过程。这样就为输出的反馈信号提供一个自然的饱和特征。

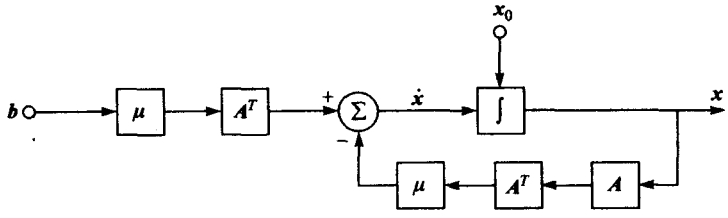



图8-2　微分实现求解线性方程组的最小二乘神经计算方法的方框图

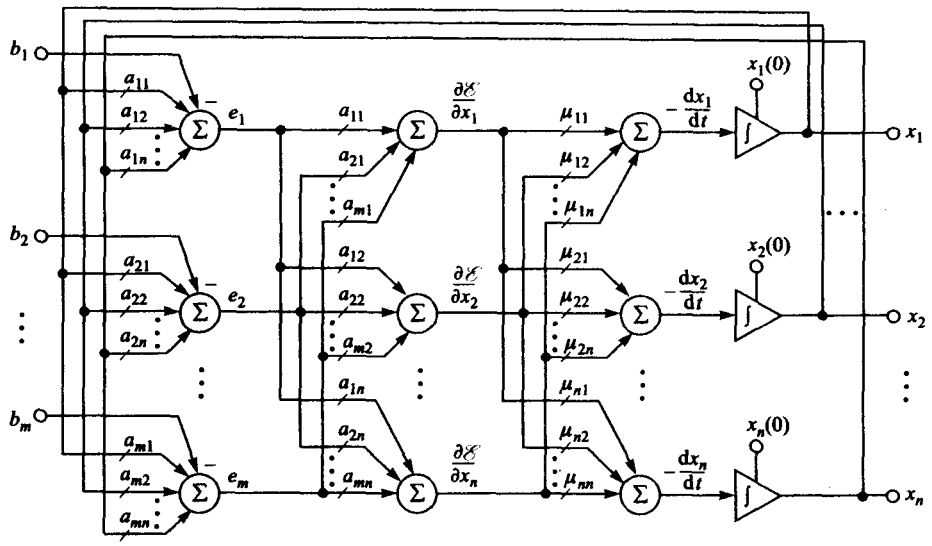



图8-3　（以最小二乘方法）求解线性方程组的神经网络体系结构图

考虑式（8-21）定义的误差向量，这个误差向量的公式只适用于完全确定性的情况（即，$m = n$）。为了对于学习过程中的任何实例估计误差，必须改写误差表达式为

$$\boldsymbol{e} = \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^T\boldsymbol{b} \tag{8-23}$$

注意式（8-23）的右侧等于零时，结果就是式（8-14）的正规方程形式。在误差代价函数

$$\mathscr{E}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{e}\|_2^2 = \frac{1}{2}\boldsymbol{e}^T\boldsymbol{e} \tag{8-24}$$

中使用式（8-23），并计算式（8-24）[⊖]中对于$\boldsymbol{x}$的梯度值，就从式（8-19）得到新的连续时间学习规则如下：

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = -\mu\boldsymbol{e} \tag{8-25}$$

其中，要以$\boldsymbol{x}(0) = \boldsymbol{x}_0$作为初始条件。因此，式（8-23）和式（8-25）一同给出了求解联立线性代数方程组的一般学习规则。此外，没有必要作出“满秩”的假设，也就是，同8.3节中的例子一样，$\rho(\boldsymbol{A}) = \min(m, n)$。这一点从学习规则本身看是显然的。正如所看到的，这里不使用除法，只用加法和乘法。由式（8-23）和式（8-25）中的学习规则所得到的结果同使用奇异值分解（SVD）（参照A.2.14节）所确定的线性方程组的解是一致的。

考虑到稳定性的原因，$\mu$矩阵的元素选择是很关键的。显然要确保式（8-25）中的向量微分方程是稳定的。$\mu$中元素的选择也决定了收敛到平衡状态（即，$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$的解）的速度。推导学习规则稳定性的条件，也就是式（8-25）中的向量微分方程，实际上是很直接的。从A.4节中可以看到，如果合适的选择李雅普诺夫（能量）函数得到的全导数是负定的，由此证明动态系统的渐近稳定性。此时，李雅普诺夫或能量函数成为式（8-24）的误差代价函数，动态系统也可以作为式（8-25）的连续时间学习规则。因此，能量函数的全导数可以根据式

---

⊖　式（8-24）对于$\boldsymbol{x}$求梯度得到$\nabla_{\boldsymbol{x}}\mathscr{E}(\boldsymbol{x}) = \boldsymbol{A}^T\boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^T\boldsymbol{b}) = \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{e}$。但是，应用与7.2节使用的同样原理，所用梯度为$\nabla_{\boldsymbol{x}}\mathscr{E}(\boldsymbol{x}) = \boldsymbol{e}$。见式（7-18）和式（7-19）的有关说明。

347～348 (A.197) 进行计算，并由

$$\mathscr{E}(\boldsymbol{x}) = \nabla_x^T \mathscr{E}(\boldsymbol{x}) \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} \tag{8-26}$$

给出。如果把式（8-19）中的连续时间学习规则的一般形式代入式（8-26），得到

$$\mathscr{E}(\boldsymbol{x}) = -\nabla_x^T \mathscr{E}(\boldsymbol{x}) \boldsymbol{\mu} \nabla_x \mathscr{E}(\boldsymbol{x}) \tag{8-27}$$

如同以前在式（8-27），解释得那样使用$\nabla_x \mathscr{E}(\boldsymbol{x}) = \boldsymbol{e}$，建立如下等式：

$$\mathscr{E}(\boldsymbol{x}) = -\boldsymbol{e}^T \boldsymbol{\mu} \boldsymbol{e} \tag{8-28}$$

回忆曾经假定矩阵$\boldsymbol{\mu}$是正定的，也就是说$\boldsymbol{\mu} > 0$；因此，有

$$\mathscr{E}(\boldsymbol{x}) = -\boldsymbol{e}^T \boldsymbol{\mu} \boldsymbol{e} < 0 \tag{8-29}$$

（假定$\boldsymbol{e} \neq 0$），这说明，能量函数的全导数总是负定的。因此，由A.4节，式（8-25）是渐近稳定的。

学习规则的标量形式能够直接由式（8-25）和式（8-23）写作

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = -\sum_{h=1}^{n} \mu_{ih} e_h \qquad x_i(0) = x_{i0} \tag{8-30}$$

$i = 1, 2, \cdots, n$。再一次指出，$\mu_{ih}$参数的选择必须仔细，以确保微分方程的稳定性以及向平衡状态（即，方程$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$的解）收敛的合理速度。式（8-30）中的$e_h$项是式（8-23）中的误差向量的单个元素。把误差向量中的标量元素通过定义矩阵给出

$$\boldsymbol{W} \triangleq \boldsymbol{A}^T \boldsymbol{A} \qquad \boldsymbol{W} \in \Re^{n \times n} \tag{8-31}$$

利用标量形式，式（8-31）可以写作

$$w_{hr} = \sum_{q=1}^{m} a_{qh} a_{qr} \tag{8-32}$$

其中，$h = 1, 2, \cdots, n$以及$r = 1, 2, \cdots, n$。利用式（8-32），式（8-23）中的误差向量元素就可以写作如下形式：

$$e_h = \sum_{r=1}^{n} w_{hr} x_r - \sum_{j=1}^{m} a_{jh} b_j \tag{8-33}$$

**例8.1** 求解下列方程组（欠定组）：

$$\begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 5 \end{bmatrix} \tag{8-34}$$

式（8-25）和式（8-23）中的学习规则的离散时间形式经常用作求解式（8-34），即

349 $$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) - \boldsymbol{\mu} \boldsymbol{e}(k) \tag{8-35}$$

其中

$$\boldsymbol{e}(k) = \boldsymbol{A}^T \boldsymbol{A} \boldsymbol{x}(k) - \boldsymbol{A}^T \boldsymbol{b} \tag{8-36}$$

具有标量$\mu = \mu_j = 0.003$ ($j = 1, 2, 3, 4$)，$k$是离散时间指数。初始条件设为$\boldsymbol{x}(0) = [0, 0, 0, 0]^T$，执行2500次迭代训练。由式（8-35）和式（8-36）中的学习规则产生的解如下

$$\boldsymbol{x} = [2.5000, 1.333, 0.1667, -1.0000]^T \tag{8-37}$$

这个解同使用奇异值分解（SVD）（参考A.2.14节）得到的解一致，保留四位小数。图8-4 中描述了代价函数的值随迭代次数的变化情况。

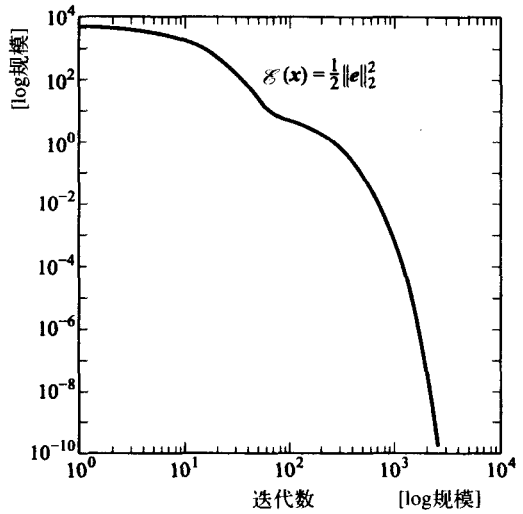



图8-4　求解式（8-34）中线性方程组的离散时间学习规则（见式（8-35）、式（8-36））收敛图

*求解线性方程组和矩阵伪逆之间的关系*

正如在本章概述中描述的一样，求解线性方程组和计算矩阵的逆（或伪逆）是等价的。这容易从式(8-35)和式（8-36）的学习规则中看到。考虑矩阵$\boldsymbol{A} \in \Re^{m \times n}$，计算它的伪逆，假定一系列的（共$m$个）$m$维$\boldsymbol{b}$向量如下

$$\begin{aligned} \boldsymbol{b}_1 &= [1,0,0,0,\cdots,0]^T = \boldsymbol{e}_1 \quad & \boldsymbol{b}_2 &= [0,1,0,0,\cdots,0]^T = \boldsymbol{e}_2 \\ \boldsymbol{b}_3 &= [0,0,1,0,\cdots,0]^T = \boldsymbol{e}_3 \quad & \boldsymbol{b}_m &= [0,0,0,0,\cdots,1]^T = \boldsymbol{e}_m \end{aligned} \tag{8-38}$$

换句话说，观察向量集共同组成一个$m \times m$的单位矩阵

$$[\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_m] = [\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_m] = \boldsymbol{I} \in \Re^{m \times m} \tag{8-39}$$

350

因此，式（8-36）中的向量$\boldsymbol{b}$可以用式（8-39）给出的单位矩阵替换，向量$\boldsymbol{x}$用矩阵$\boldsymbol{C} \in \Re^{n \times m}$替换，误差向量$\boldsymbol{e}$用误差矩阵$\boldsymbol{E} \in \Re^{n \times m}$替代。这样，导出误差矩阵的表达式为

$$\boldsymbol{E}(k) = \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{C}(k) - \boldsymbol{A}^T \tag{8-40}$$

式（8-35）的表达式变成（对于标量学习率参数$\mu$）

$$\boldsymbol{C}(k+1) = \boldsymbol{C}(k) - \mu\boldsymbol{E}(k) \tag{8-41}$$

将式（8-40）代入式（8-41），再假定标量学习率参数$\mu > 0$，从而有

$$\boldsymbol{C}(k+1) = \boldsymbol{C}(k) + \mu\boldsymbol{A}^T[\boldsymbol{I} - \boldsymbol{A}\boldsymbol{C}(k)] \tag{8-42}$$

这就是在式（7-20）中的决定矩阵伪逆的学习规则。即使式（8-40）中的误差矩阵与式（7-13）中的不同，仍然可以看到在所使用的误差代价函数的形式方面，二者是等价的。

## 8.5 求解线性方程组的共轭梯度学习规则

与求解方程组的任何数值方法有关的收敛问题都很重要[15, 29]。现在进一步研究前一节所提及的神经计算方法。这里使用共轭梯度的方法[30-32]（参见A.5.5节）代替最速下降的方法，加快收敛的速度。按照在A.5.5节所描述的具有重启动能力的Fletcher-Reeves共轭梯度算法（即，每经过$n$次迭代产生一个最速下降阶段）。因此，打算求解$\boldsymbol{Ax}=\boldsymbol{b}$的超定或者完全确定性方程组，其中假定$\boldsymbol{A}\in\Re^{m\times n}(m\geqslant n)$。

为了将共轭梯度方法整合到前一节提及的离散时间学习规则，首先要推导出迭代$\boldsymbol{x}_k$的更新表达式中$\alpha_k$的表达式。在Fletcher-Reeves共轭梯度算法的第4步中，求解的更新如下

$$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\alpha_k\boldsymbol{d}_k \tag{8-43}$$

其中

$$\alpha_k=\min_{\alpha\geqslant 0}\mathscr{E}(x_k+\alpha\boldsymbol{d}_k) \tag{8-44}$$

其中向量$\boldsymbol{d}_k$是当前方向向量，$\mathscr{E}(\bullet)$是要最小化的目标函数。此处目标函数如下

$$\mathscr{E}(\boldsymbol{x})=\frac{1}{2}\|\boldsymbol{e}\|_2^2=\frac{1}{2}\boldsymbol{e}^T\boldsymbol{e} \tag{8-45}$$

351 其中，对于求解$\boldsymbol{Ax}=\boldsymbol{b}$的解误差由下式给出

$$\boldsymbol{e}=\boldsymbol{Ax}-\boldsymbol{b} \tag{8-46}$$

由此，

$$\begin{aligned}\mathscr{E}(\boldsymbol{x}_k+\alpha\boldsymbol{d}_k)&=\frac{1}{2}[\boldsymbol{A}(\boldsymbol{x}_k+\alpha\boldsymbol{d}_k)-\boldsymbol{b}]^T[\boldsymbol{A}(\boldsymbol{x}_k+\alpha\boldsymbol{d}_k)-\boldsymbol{b}]\\&=\frac{1}{2}(\boldsymbol{x}_k^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}_k+\alpha\boldsymbol{x}_k^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{d}_k-\boldsymbol{x}_k^T\boldsymbol{A}^T\boldsymbol{b}+\alpha\boldsymbol{d}_k^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}_k)\\&\quad+\alpha^2\boldsymbol{d}_k^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{d}_k-\alpha\boldsymbol{d}_k^T\boldsymbol{A}^T\boldsymbol{b}-\boldsymbol{b}^T\boldsymbol{A}\boldsymbol{x}_k-\alpha\boldsymbol{b}^T\boldsymbol{A}\boldsymbol{d}_k+\boldsymbol{b}^T\boldsymbol{b}\end{aligned} \tag{8-47}$$

必须计算式（8-47）关于$\alpha$的梯度，然后令结果等于0。

$$\nabla_\alpha\mathscr{E}(\boldsymbol{x}_k+\alpha\boldsymbol{d}_k)=\frac{\partial\mathscr{E}(\boldsymbol{x}_k+\alpha\boldsymbol{d}_k)}{\partial\alpha}=\boldsymbol{d}_k^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}_k+\alpha\boldsymbol{d}_k^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{d}_k-\boldsymbol{d}_k^T\boldsymbol{A}^T\boldsymbol{b}=0 \tag{8-48}$$

从式（8-48）求解$\alpha$，同时令$\alpha\to\alpha_k$，得到

$$\alpha_k=\frac{\boldsymbol{d}_k^T\boldsymbol{A}^T\boldsymbol{b}-\boldsymbol{d}_k^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}_k}{\boldsymbol{d}_k^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{d}_k}=\frac{\boldsymbol{d}_k^T\overbrace{[\boldsymbol{A}^T\boldsymbol{b}-\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}_k]}^{-\boldsymbol{g}_k}}{\boldsymbol{d}_k^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{d}_k}=\frac{-\boldsymbol{d}_k^T\boldsymbol{g}_k}{\boldsymbol{d}_k^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{d}_k} \tag{8-49}$$

或

$$\alpha_k=\frac{-\boldsymbol{g}_k^T\boldsymbol{d}_k}{\boldsymbol{d}_k^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{d}_k} \tag{8-50}$$

其中$\boldsymbol{g}_k$是式（8-45）的梯度，

$$\boldsymbol{g}_k=\nabla_x\mathscr{E}(\boldsymbol{x}_k)=\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}_k-\boldsymbol{A}^T\boldsymbol{b} \tag{8-51}$$

因此，求解$\boldsymbol{Ax}=\mathbf{b}$的Fletcher-Reeves共轭梯度算法（附带重新启动）的步骤可以总结如下：

**求解$Ax = b$的Fletcher-Reeves 共轭梯度算法**（附带重新启动）

**步骤1**　置$x_0$。

**步骤2**　计算$g_k|_{k=0} = g_0 = A^TAx_0 - A^Tb$。

**步骤3**　令$d_0 = -g_0$。

**步骤4**　计算$x_{k+1} = x_k + \alpha_k d_k$，其中$\alpha_k = -g_k^T d_k / (d_k^T A^T A d_k)$。

**步骤5**　计算$g_{k+1} = A^TAx_{k+1} - A^Tb$。

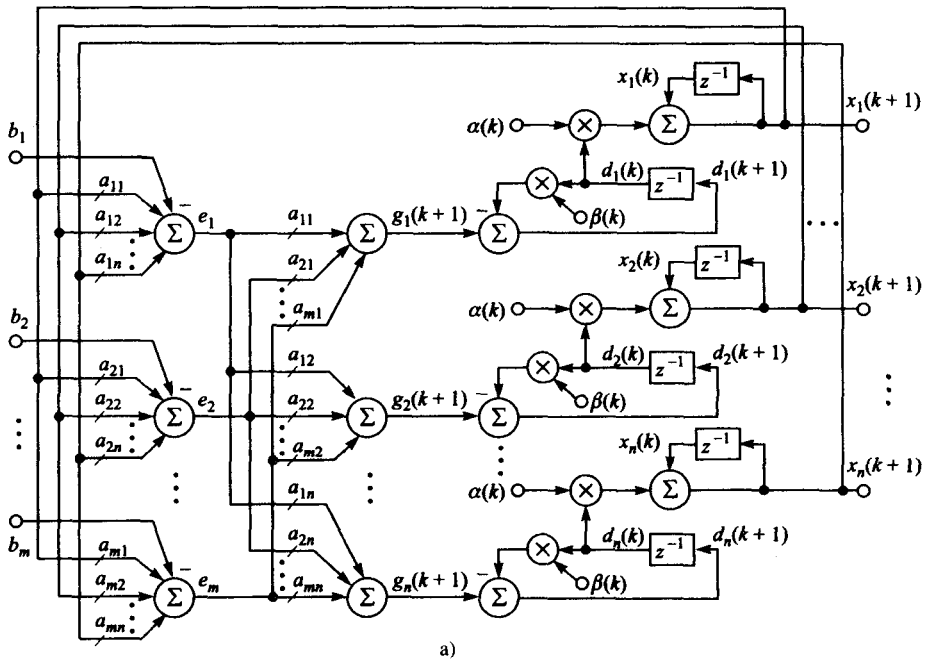
**步骤6**　计算$d_{k+1} = -g_{k+1} + \beta_k d_k$，其中$\beta_k = g_{k+1}^T g_{k+1}/(g_k^T g_k)$。

**步骤4～6**　随$k = 0, 1, \cdots, n-1$变化执行。

**步骤7**　用$x_n$替换$x_0$，然后回到步骤1。

**步骤8**　继续以上步骤直到达到收敛，终止的标准是$\|d_k\| < \varepsilon$（其中$\varepsilon$是一个预先确定的足够小的数）。 □

图8-5描述了用共轭梯度算法求解线性方程组的离散时间神经网络体系结构。 352



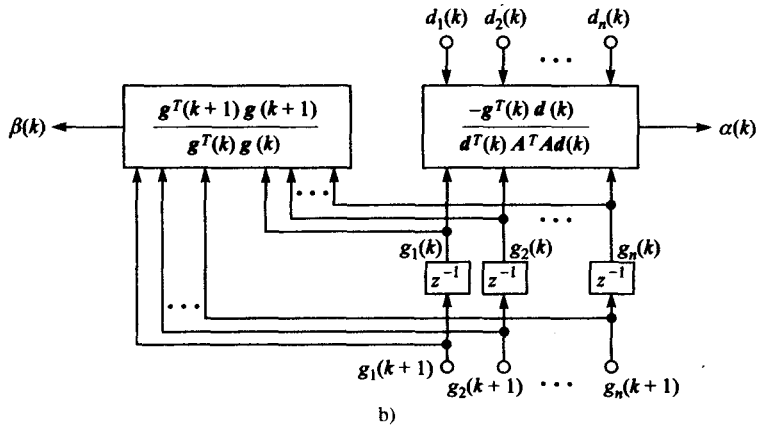




图8-5　a) 用共轭梯度算法求解线性代数方程组的离散时间神经网络体系结构；b) 用于计算$\alpha_k$和$\beta_k$的部分神经体系结构

**例8.2**　求解以下方程组

$$\begin{bmatrix} 3 & 5 & 1 & 4 \\ 3 & -8 & -2 & 11 \\ -1 & 0 & 0 & -7 \\ -10 & -3 & -2 & -5 \\ 1 & -5 & -2 & 5 \\ 8 & -6 & 2 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -7 \\ 2 \\ 8 \\ 4 \\ 10 \\ 3 \end{bmatrix} \tag{8-52}$$

353

对于这种超定的方程组，采用式（8-35）和式（8-36）给出的离散时间最速下降方法以及上面刚提到的离散时间共轭梯度方法都可以求解$\boldsymbol{x}$。使用最速下降梯度方法，单个学习率参数设置为$\mu = 0.0055$以及一系列的随机数据初始化$\boldsymbol{x}$，学习规则在1 900次迭代后收敛到解。解如下

$$\boldsymbol{x} = [1.8725, 0.0935, -7.4699, -1.5274]^T \tag{8-53}$$

这与使用奇异值分解（保留四位小数）的结果一致。收敛图如图8-6a所示，为解向量中的每个元素的情况。使用上面提到的共轭梯度学习规则（使用同最速下降学习规则相同的初始条件），四次迭代后就可以得到式（8-53）的解！因此，基于共轭梯度的学习规则比基于最速下降的方法在训练步骤上少475次！对于基于共轭梯度的学习规则，图8-6b的收敛图反映了解向量中的每一个元素的情况。

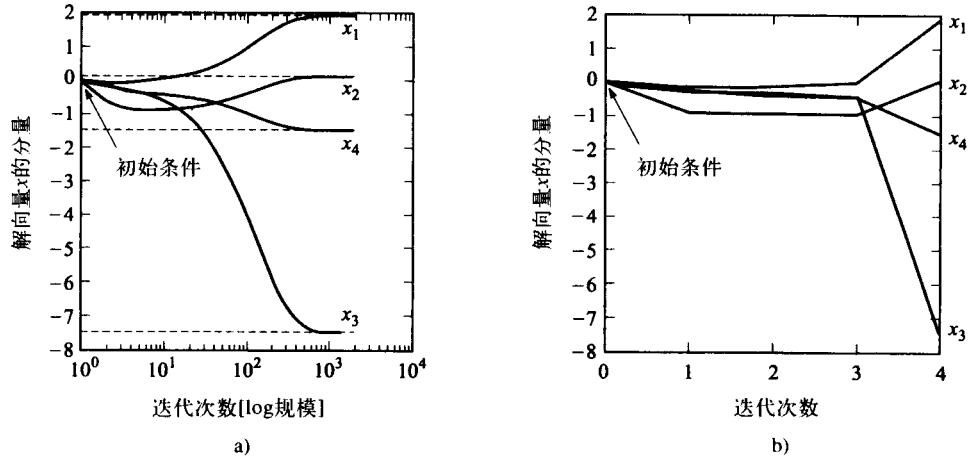



图8-6　a) 使用基于最速下降学习规则求解式（8-52）问题的收敛图；b) 使用基于共轭梯度学习规则求解式（8-52）问题的收敛图

## 8.6　求解受噪声侵扰的线性方程组的广义鲁棒方法

很多时候数据中的误差分布并不是高斯的（或者正态的）。在这些情况中，例如，假如脉冲（“尖锐”）噪声存在，那么前一节涉及的基于$L_2$范数的标准最小二乘性能准则对于$\boldsymbol{Ax} = \boldsymbol{b}$

354 的解将只能产生一个很差的估计值。因此，在非常大的误差（称作出格点，亦称离群值）、脉冲噪声或者有色噪声出现的情况下，当系统误差不遵循高斯分布时，应该采用另一种不同的方法。为了减缓非高斯噪声的影响同时提供解向量$\boldsymbol{x}$的鲁棒性估计，我们要采用一种更加一般的误差代价函数公式。只有当所有的误差属于观察向量$\boldsymbol{b}$并且满足高斯分布，（即，满足零均值以及等变化的无关联误差）普通的最小二乘问题才是最优的。现在考虑误差存在于数量矩

阵$\boldsymbol{A}$，并且是非高斯分布的情况。

这里所采用的方法比迭代加权的最小二乘方法（或者鲁棒最小二乘准则）更有一般性[8, 13, 19, 20]。一般情况下，在数据中想要的出格点加权（即，严重的误差）少于较小的误差。因此，这就需要一个“增长”缓于二次的加权函数。这里所指的二次函数就是在8.4节用于求解$\boldsymbol{Ax}=\boldsymbol{b}$问题的最小二乘方法。二次函数的“尾部”施加在大的（或严重的）系统误差的加权往往太大。除非这些误差缩小，否则它们对$\boldsymbol{Ax}=\boldsymbol{b}$的解的影响将是彻底压倒性的，同时，对于解向量$\boldsymbol{x}$的估计将会非常差。因此，正如前面描述的，需要一个增加少于二次的加权函数。

对于解$\boldsymbol{Ax}=\boldsymbol{b}$的问题（$\boldsymbol{A}\in\Re^{m\times n}$，$\boldsymbol{b}\in\Re^{m\times 1}$），现在可以描述成如下一般化的最小化问题。确定如下向量

$$\boldsymbol{x}\in\Re^{n\times 1} \tag{8-54}$$

从而最小化加权统计误差代价函数[33-35]

$$\mathscr{E}(\boldsymbol{x})=\mathbf{1}^T\boldsymbol{S}\mathrm{E}\{f(\boldsymbol{e})\} \tag{8-55}$$

其中，$\mathbf{1}=[1\ \ 1\ \ \cdots\ \ 1]^T$，$\boldsymbol{S}\in\Re^{m\times m}$（$\boldsymbol{S}^T=\boldsymbol{S}$和$\boldsymbol{S}>0$），$\mathrm{E}\{\cdot\}$是期望（值）算子，$f\{\cdot\}$是一个凸的加权函数[⊖]，以及

$$\boldsymbol{e}=\boldsymbol{Ax}-\boldsymbol{b} \tag{8-56}$$

因为只有瞬时误差会被处理，所以期望值算子可以忽略[33]。因此，实际的误差代价函数可以写作如下形式

$$\mathscr{E}(\boldsymbol{x})=\mathbf{1}^T\boldsymbol{S}f(\boldsymbol{e}) \tag{8-57}$$

其中，正定对称加权矩阵$\boldsymbol{S}$一般情况下选择单位矩阵。为求解$\boldsymbol{Ax}=\boldsymbol{b}$采用最速下降梯度方法来开发学习策略，使用与8.4节相同的形式

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}=-\mu\nabla_x\mathscr{E}(\boldsymbol{x}) \tag{8-58}$$

其中，$\boldsymbol{\mu}\in\Re^{n\times n}$（$\boldsymbol{\mu}=[\mu_{ij}]$，其中$i, j=1, 2, \cdots, n$）是一个典型的要对角化的正定矩阵，也就是，$\mu_{ij}=\mu_j\delta_{ij}$，其中$\mu_j>0$。因此，必须定义式（8-58）中的梯度如下 355

$$\nabla_x\mathscr{E}(\boldsymbol{x})=\frac{\partial\mathscr{E}(\boldsymbol{x})}{\partial\boldsymbol{x}}=\frac{\partial}{\partial\boldsymbol{x}}\mathbf{1}^T\boldsymbol{S}f(\boldsymbol{e})=\frac{\partial\boldsymbol{e}}{\partial\boldsymbol{x}}\boldsymbol{S}g(\boldsymbol{e}) \tag{8-59}$$

其中，$g(t)=\mathrm{d}f(t)/\mathrm{d}t$且$t$是一个哑元。需要对式（8-59）中的偏导数$\partial\boldsymbol{e}/\partial\boldsymbol{x}$说明一下。我们可以把式（8-56）中的误差向量写作如下形式

$$\boldsymbol{e}=\boldsymbol{Ax}-\boldsymbol{b}=\begin{bmatrix}\boldsymbol{a}_1\\ \boldsymbol{a}_2\\ \vdots\\ \boldsymbol{a}_m\end{bmatrix}\boldsymbol{x}-\boldsymbol{b} \tag{8-60}$$

其中，$\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_m$是$\boldsymbol{A}$的行向量，也就是，$\boldsymbol{a}_j\in\Re^{1\times n}$（$j=1, 2, \cdots, m$）。因此，式（8-60）可以写作如下形式

$$\boldsymbol{e}=\begin{bmatrix}\boldsymbol{a}_1\boldsymbol{x}\\ \boldsymbol{a}_2\boldsymbol{x}\\ \vdots\\ \boldsymbol{a}_m\boldsymbol{x}\end{bmatrix}_{m\times 1}-\boldsymbol{b} \tag{8-61}$$

⊖ 实际上，$f\{\cdot\}$为一个近似凸函数。

式（8-61）对于向量$x$求偏导数如下（利用A.3.4.1节的结果）

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}}=\frac{\partial}{\partial \boldsymbol{x}}\begin{bmatrix}\boldsymbol{a}_1\boldsymbol{x}\\ \boldsymbol{a}_2\boldsymbol{x}\\ \vdots\\ \boldsymbol{a}_m\boldsymbol{x}\end{bmatrix}=\begin{bmatrix}\frac{\partial}{\partial \boldsymbol{x}}\boldsymbol{a}_1\boldsymbol{x}\\ \frac{\partial}{\partial \boldsymbol{x}}\boldsymbol{a}_2\boldsymbol{x}\\ \vdots\\ \frac{\partial}{\partial \boldsymbol{x}}\boldsymbol{a}_m\boldsymbol{x}\end{bmatrix}=[\boldsymbol{a}_1^T,\boldsymbol{a}_2^T,\cdots,\boldsymbol{a}_m^T]_{n\times m}=\boldsymbol{A}^T \tag{8-62}$$

因此，通过使用式（8-62）的结果，式（8-59）的梯度$\nabla_x \mathscr{E}(\boldsymbol{x})$可以写作

$$\nabla_x \mathscr{E}(\boldsymbol{x})=\frac{\partial \mathscr{E}(\boldsymbol{x})}{\partial \boldsymbol{x}}=\frac{\partial}{\partial \boldsymbol{x}}\mathbf{1}^T\boldsymbol{S}f(\boldsymbol{e})=\boldsymbol{A}^T\boldsymbol{S}g(\boldsymbol{e}) \tag{8-63}$$

再利用式（8-63），式（8-58）的鲁棒性学习规则可以写作如下形式

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}=-\mu\boldsymbol{A}^T\boldsymbol{S}g(\boldsymbol{e}) \tag{8-64}$$

为了表明这是一个普遍的结果，假使令加权矩阵$\boldsymbol{S}=\boldsymbol{I}$，并且令加权函数取作二次形式，也就是，$f(t)=1/2t^2[g(t)=\mathrm{d}f(t)/\mathrm{d}t=t$,即线性函数]，则式（8-64）变化成

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}=-\mu\boldsymbol{A}^T\boldsymbol{e} \tag{8-65}$$

这也就是式（8-22）为求问题的首个最小二乘解获得的结果。图8-7描述了用于求解线性方程组的广义鲁棒性方法的神经网络体系结构的细节。在图中假定加权矩阵为单位矩阵，即
356
$\boldsymbol{S}=\boldsymbol{I}_{m\times m}$。

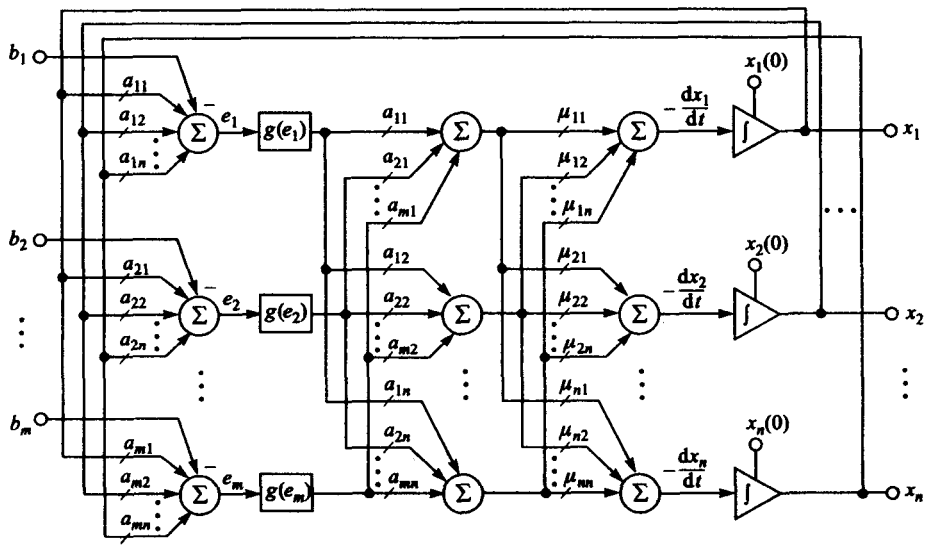



图8-7　求解线性方程组的广义鲁棒性方法的神经网络体系结构图。假定$\boldsymbol{S}=\boldsymbol{I}_{m\times m}$同时$g(t)=\mathrm{d}f(t)/\mathrm{d}t$，且$t$是一个虚变量，其中$f(\cdot)$是选择的非线性加权函数

我们需要讨论一下加权函数$f(\cdot)$的选择问题，即在式（8-57）中出现的误差代价函数。正如前面所描述的，此函数必须是凸的或者近似凸的（参照A.3.1节定义 A.11）。尽管加权函数的选择存在着很多的可能，但是表9-2中总结了其中四个最广泛的应用，图9-21给出了图示。

图9-22中描述了相关联的导数。所使用的函数类型以及限界参数$\beta$都依赖于具体数据。因此，通常可以使用试错的过程来决定使用“最好”的函数以及“最好”的参数$\beta$值。有趣的是，可以设置使用一个最普通的加权函，即logistic函数，见下式。

$$f_L(t) = \beta^2 \ln[\cosh(t/\beta)] \tag{8-66}$$

图8-8显示对于限界参数$\beta$三个的值的加权函数以及与二次函数进行对比的坐标图。正如在图8-8中所看到的，随着限界参数$\beta$值的增加，曲线的形状接近二次函数。因此，对于较大的$\beta$值，学习算法的鲁棒形性能会减小。观察这幅图，我们发现误差大约位于$-2/3 \sim 2/3$之间，所有的曲线采用了相同的（二次）加权。在此范围以外的误差，依赖于限界函数的值，加权将小于二次。但是，选择参数$\beta$值时必须小心仔细。当$\beta$值太小时，有价值的信息会丢弃。相反，当$\beta$值太大时，出格点以及噪声强烈干扰数据的同时，减退解向量的估计（见图8-8）。图中显示的 357
$\beta$值介于$0.7 \sim 1$之间的部分显示了一种可能的最优区域，其中有较好的出格点出现及噪声拒绝发生。图8-9描述了以上各自的导数函数。

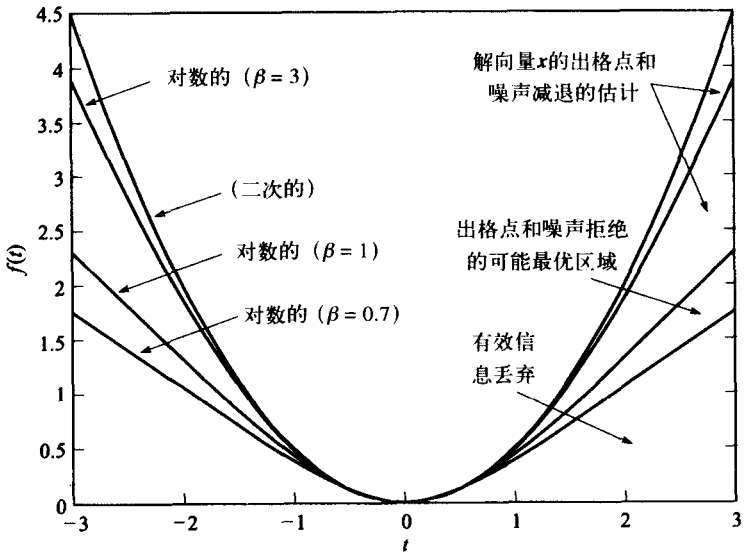



图8-8　对于三个不同的限界参数$\beta$值的对数加权函数对比二次函数$f_Q(t) = 1/2t^2$，其中$t$是一个虚变量

把鲁棒性神经计算学习规则总结为如下步骤。采用离散时间向量矩阵形式的学习规则。

**求解具有噪声的方程组的鲁棒学习算法**

**步骤1**　选取合适的矩阵$\mu$和$S$。

**步骤2**　选取一个合适的加权函数$f(\cdot)$，同时计算导数$g(\cdot)$。再选择一个合理的限界参数$\beta$值。

**步骤3**　令$k = 0$（$k$是离散时间指数），并且选择一个初始点$x(0)$。

**步骤4**　根据下式，计算解误差。

$$e(k) = Ax(k) - b \tag{8-67}$$

**步骤5**　根据下式，更新解向量的估计值。

$$x(k+1) = x(k) - \mu A^T S g[e(k)] \tag{8-68}$$

**步骤6**　收敛完成则停止；否则，令$k \leftarrow k+1$，返回步骤4。 □

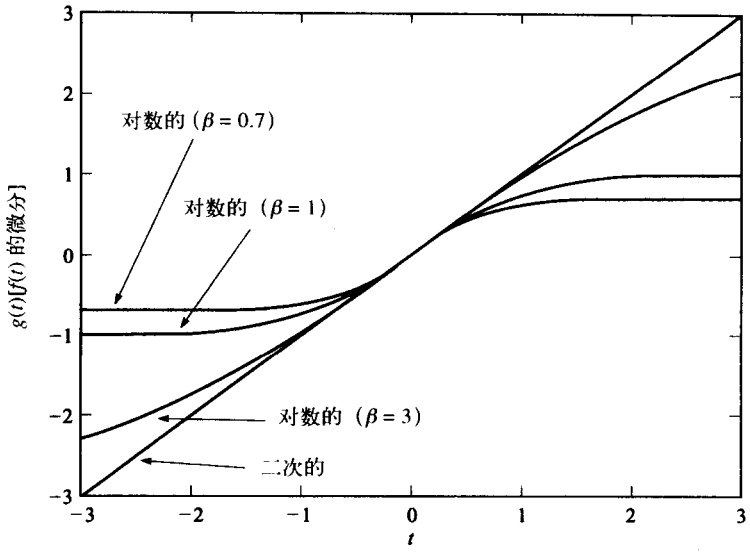



图8-9　图8-8中的加权函数的导数

**例8.3**　这里示例式（8-67）和式（8-68）中的离散时间鲁棒学习规则是如何减轻脉冲噪声的影响的。要求找到一个包含10个方程、10个未知数的系统$\boldsymbol{Ax}=\boldsymbol{b}$的解$\boldsymbol{x}\in\Re^{10\times 1}$，其中

$$\boldsymbol{A}=\begin{bmatrix}
0.1417 & -1.0075 & -0.7732 & -0.9113 & 0.4871 & -1.6601 & -1.3186 & 0.4035 & -0.6255 & -1.0275\\
-1.2650 & 1.3090 & -0.1451 & -0.2353 & -1.4658 & 0.4176 & 1.1449 & 2.0468 & 0.8100 & -0.2398\\
-0.2704 & -0.5756 & 0.2369 & 0.3593 & -0.6807 & 0.4390 & 0.4493 & -1.0552 & -1.0041 & -0.3516\\
-0.5650 & -2.1317 & -2.3882 & -1.5344 & 0.1802 & 1.3482 & 1.1635 & -1.4374 & -1.1220 & -1.0924\\
1.8503 & -0.5428 & 2.4866 & 0.9782 & 0.2633 & 0.4738 & -1.0853 & 1.1041 & 0.2045 & -1.1709\\
0.8016 & 0.0237 & 0.3418 & -0.4727 & 1.6016 & -0.3046 & 1.9539 & -1.3561 & 0.8812 & -0.5963\\
1.0093 & -0.0254 & 0.0430 & -0.9378 & -1.4734 & 0.2955 & 0.2372 & 0.3510 & 0.5705 & 0.5723\\
-0.2946 & -1.6822 & 0.1412 & -0.8454 & 1.1526 & -0.7930 & 0.1700 & -1.9511 & -1.3950 & -0.8927\\
0.5661 & -0.0328 & 0.6679 & 1.4221 & -0.3615 & 1.5595 & 0.1963 & 0.5752 & 0.4470 & -0.2336\\
0.1112 & 0.9667 & -0.7838 & -0.4777 & -0.8919 & 2.6388 & -0.2533 & -1.5569 & -0.1884 & -0.4363
\end{bmatrix} \tag{8-69}$$

以及

$$\boldsymbol{b}=[-0.7800,\ -1.1331,\ 0.9693,\ -0.0026,\ 1.8051,\ 0.3402,\ 0.2821,\ 1.4632,\ 0.1786,\ 0.5643]^T \tag{8-70}$$

式（8-69）中的矩阵$\boldsymbol{A}$认为是“真值”矩阵，因为这个矩阵实际上使用$\boldsymbol{A}_\varepsilon=\boldsymbol{A}+\Delta\boldsymbol{A}$，它受到了噪声的干扰。因此，实际上求解的是如下系统

$$\boldsymbol{A}_\varepsilon\boldsymbol{x}_\varepsilon=\boldsymbol{b} \tag{8-71}$$

358
~
359

其中式（8-69）中的矩阵$\boldsymbol{A}_\varepsilon$的一些元素受到了脉冲噪声的干扰。特别是，可以从区间为[0, 0.25]的均匀分布中选取随机数据，这些值加到式（8-69）中矩阵$\boldsymbol{A}$的随机的选项中，形成受干扰的阵列$\boldsymbol{A}_\varepsilon$。因此，这个均匀分布可以作为脉冲噪声（或者出格点）的模型。脉冲噪声所干扰的阵列元素以0.15的概率随机选取；也就是，式（8-69）中的矩阵$\boldsymbol{A}$以平均15%的概率选取

的元素将受到方差为$\sigma^2 = (0.15)\ (0.25)^2/12 = 7.8125 \times 10^{-4}$的脉冲噪声的干扰（参照A.7.4节）。事实上，$\Delta a_{ij}$的值以及$\boldsymbol{A}$受到干扰的阵列项都在表8-1中列出。

因此，目标是求解式（8-71）中的$\boldsymbol{x}_c$，并且使用一种方法使获得的解尽可能接近真实的解，也就是，令$\boldsymbol{x}_c \cong \boldsymbol{x}$。利用MATLAB求解式（8-71）中的$\boldsymbol{x}_c$，得到

$$\boldsymbol{x}_c^M = [0.4295, 0.2243, 0.9805, -0.7051, 0.4362, 0.5609, 0.3072, 0.2087, -1.4279, 0.5178]^T \tag{8-72}$$

因为$\boldsymbol{A}_c$是带有良好条件数的满秩矩阵（参见A.2.15节），式（8-72）的解可以以简明的方式使用MATLAB内建的函数inv，即$\boldsymbol{x}_c^M = inv(\boldsymbol{A}_c)*\boldsymbol{b}$计算。真实的解（即，“真值”系统$\boldsymbol{A}_x^M = \boldsymbol{b}$的解）$\boldsymbol{x}^M$通过相同的方式计算得到，即$\boldsymbol{x}^M = inv(\boldsymbol{A})*\boldsymbol{b}$，从而得到如下式

$$\boldsymbol{x}^M = [0.3177, 0.0486, 0.9002, -0.4987, 0.2073, 0.4242, 0.2029, -0.0008, -1.0085, 0.2863]^T \tag{8-73}$$

上面提到的离散时间形式的鲁棒神经计算学习规则可以用于决定鲁棒解。总共需要420次的迭代才能达到收敛，使用限界参数$\beta = 0.7$的对数加权函数。这是使用试错法确定的最优值。使用标量学习率参数，通过试错法确定最优值使$\mu = 0.01$。权值矩阵$\boldsymbol{S} = \boldsymbol{I}$，$\boldsymbol{x}_c^N(0)$的初始条件设置为0均值和单位方差高斯随机数据。通过鲁棒神经计算方法推导出的结果$\boldsymbol{x}_c^N$如下

$$\boldsymbol{x}_c^N = [0.2663, 0.0306, 0.9194, -0.5053, 0.1882, 0.4092, 0.1678, -0.0553, -0.8985, 0.2828]^T \tag{8-74}$$

360

(1) 使用MATLAB求解$\boldsymbol{A}_c\boldsymbol{x}_c = \boldsymbol{b}$的解和真值系统$\boldsymbol{A}\boldsymbol{x}^M = \boldsymbol{b}$的解；(2) 使用鲁棒神经计算方法求解$\boldsymbol{A}_c\boldsymbol{x}_c = \boldsymbol{b}$的解和真值系统$\boldsymbol{A}\boldsymbol{x}^M = \boldsymbol{b}$的解，估计的标准误差（SEE）通过计算得到结果。使用式（8-72）和式（8-73）的结果，MATLAB求解的SEE值如下

$$\mathrm{SEE}_M = \left[\frac{\sum_{i=1}^{10}(x_{ci}^M - x_i^M)^2}{10}\right]^{1/2} = 0.2116 \tag{8-75}$$

使用式（8-73）和式（8-74），鲁棒神经计算求解的SEE值如下

$$\mathrm{SEE}_N = \left[\frac{\sum_{i=1}^{10}(x_{ci}^N - x_i^M)^2}{10}\right]^{1/2} = 0.0450 \tag{8-76}$$

可见，（使用MATLAB的）$\mathrm{SEE}_M$值大约比（使用鲁棒神经计算方法）的$\mathrm{SEE}_N$值大4.7倍。使用鲁棒神经计算方法较之使用MATLAB方法，方程组解的改善是显而易见的，可以在图8-10中看到。

可以观察到在矩阵$\boldsymbol{A}$中，由于扰动解的敏感性。即表8-1中给出扰动，利用在A.2.15节方程（A.115）所提出的关系，方程（A.115）如下：

$$\frac{\|\Delta x\|}{\|x + \Delta x\|} \leqslant \mathrm{cond}(\boldsymbol{A})\frac{\|\Delta A\|}{\|A\|} \tag{8-77}$$

其中，式（A.107）给出的$\boldsymbol{A}$的条件数可以表示为$\mathrm{cond}(\boldsymbol{A})$，也就是$\boldsymbol{A}$最大奇异值与最小值的比率。考虑这个问题得到的结果，式（8-77）右边计算的结果为1.8494。左边使用MATLAB方

法得出的结果是0.3105，而使用神经计算方法的结果是0.0940。因此，不等式是满足的，这些结果再一次说明了鲁棒神经计算方法可以得到更好的结果。

**表8-1　使用均匀分布随机产生的扰动作为脉冲噪声干扰例8.3中矩阵$A$随机选择的元素**

| 阵列索引 | $\Delta a_{ij}$ | 阵列索引 | $\Delta a_{ij}$ |
|---|---|---|---|
| (1, 3) | 0.0564 | (5, 1) | 0.0104 |
| (1, 10) | 0.2393 | (6, 8) | 0.2120 |
| (3, 6) | 0.2084 | (9, 3) | 0.0817 |
| (4, 1) | 0.0837 | (9, 7) | 0.2353 |
| (4, 3) | 0.0137 | | |

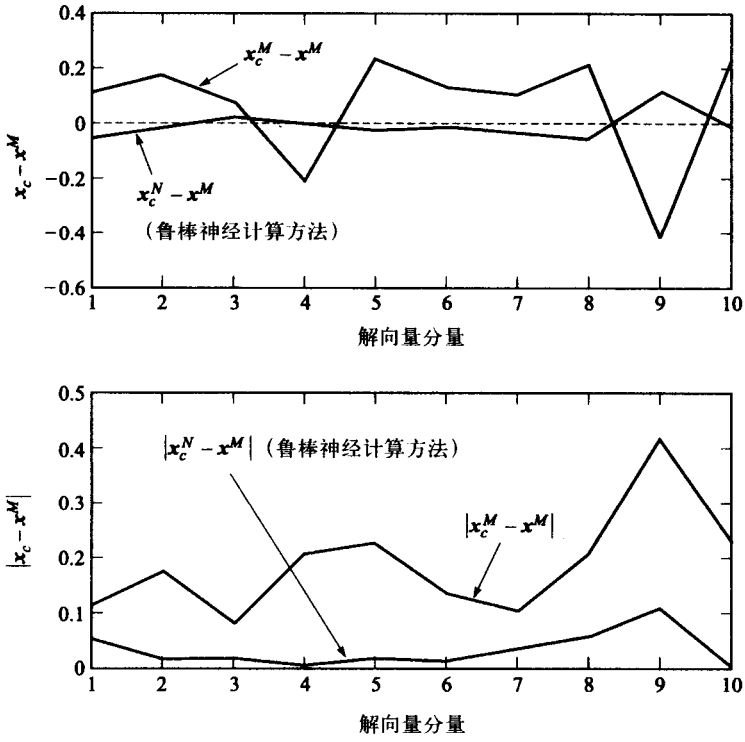



图8-10　相对于真值系统解的受干扰系统的MATLAB求解的剩余误差与相对于真值系统解的受干扰系统的鲁棒神经计算求解的剩余误差对比图

## 8.7　带病态确定数值秩的不适定问题的正则化方法

寻求下式的解

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{8-78}$$

其中，$\boldsymbol{A}\in\Re^{m\times n}$，$\boldsymbol{x}\in\Re^{n\times 1}$，$\boldsymbol{b}\in\Re^{m\times 1}$，且$m\geqslant n$，矩阵$\boldsymbol{A}$可能是病态的，且拥有病态确定数值秩[25]。也就是，$\boldsymbol{A}$的奇异值在频谱中没有确定间隙的情况下向0衰减。因此，讨论的焦点是病
361 态的最小二乘问题。这类问题在许多不同的情况下出现，例如，第一类弗雷德霍姆（Fredholm）积分方程组的数值解，这是不适定问题的经典例子。求解病态问题有几种方法可

以选择，例如，截断的QR（因子分解）正则化[38]，截断的奇异值分解（SVD）正则化[39-41]，以及阻尼最小二乘或者Tikhonov正则化[42-53]。我们将主要讨论Tikhonov正则化方法。

最初在式（8-12）中，定义的标量代价（能量）函数，为式(8-78)的最小二乘解创造了条件，如下

$$\mathscr{E}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \tag{8-79}$$

计算相对于$\boldsymbol{x}$的式（8-79）的梯度，令结果等于0，导出式（8-14）中的正规方程，重复如下

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^T\boldsymbol{b} = \boldsymbol{0} \tag{8-80}$$

将采用与式（8-15）稍微不同的方式求式（8-80）的解。式（8-15）采用了Moore-Penrose逆进行求解。这里采用$\boldsymbol{A}$的奇异值分解（SVD）（参照A.2.14节），也就是

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T \tag{8-81}$$ 362

在A.2.14节中，通过式（8-80）中的正规方程用$\boldsymbol{A}$的奇异值分解（SVD），求解式（8-78）最小二乘解，可以写作

$$\boldsymbol{x} = \boldsymbol{V}\boldsymbol{S}^+\boldsymbol{U}^T\boldsymbol{b} = \sum_{i=1}^{n}\frac{\upsilon_i\boldsymbol{u}_i^T}{\sigma_i}\boldsymbol{b} \tag{8-82}$$

如果$\boldsymbol{A}$是不满秩的，也就是，$\rho(\boldsymbol{A}) < n$，此时式（8-82）中的求和极限一定会影响到矩阵$\boldsymbol{A}$的实际的秩。假如令$\rho(\boldsymbol{A}) = k \leqslant n$，此时式（8-82）可以写作如下形式

$$\boldsymbol{x}_k = \boldsymbol{V}\boldsymbol{S}^+\boldsymbol{U}^T\boldsymbol{b} = \sum_{i=1}^{k}\frac{\upsilon_i\boldsymbol{u}_i^T}{\sigma_i}\boldsymbol{b} \tag{8-83}$$

考虑方程（8-83）采用了截断的奇异值分解（TSVD）方法，$k$作为截断参数（作用同Tikhonov正则化方法中的λ相似[39]）。当矩阵$\boldsymbol{A}$具有病态的确定数值秩时，该方法的问题就会出现了。也就是，矩阵$\boldsymbol{A}$的奇异值在频谱中没有定义间隙的情况下向0衰减。

现在定义一个正则化能量函数，写作

$$\mathscr{E}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \frac{\lambda^2}{2}\|\boldsymbol{x}\|_2^2 \tag{8-84}$$

其中$\lambda > 0$是定义为正则化参数的自由参数。它控制着正则化求解过程中的“平滑度”。我们寻求问题$\min_{\boldsymbol{x}\in\mathbb{R}^n}\mathscr{E}(\boldsymbol{x})$的解$\boldsymbol{x} = \boldsymbol{x}_\lambda$。能量函数中的额外项伴随着标准的误差项一起最小化了，也就是，$1/2\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$。第二个项$(\lambda^2/2)\|\boldsymbol{x}\|_2^2$称作平滑度约束，也指稳定器能量。计算式（8-84）关于$\boldsymbol{x}$的梯度，然后将结果置为零，就可以推导出“修正的”正规方程组如下

$$(\boldsymbol{A}^T\boldsymbol{A} + \lambda^2\boldsymbol{I})\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b} \tag{8-85}$$

式（8-85）中的基本结果就是正则化矩阵$\tilde{\boldsymbol{A}} = \boldsymbol{A}^T\boldsymbol{A} - \lambda^2\boldsymbol{I}$的条件，也就是，对应于$\boldsymbol{A}^T\boldsymbol{A}$的条件数$\tilde{\boldsymbol{A}}$的条件数。可以很直接地表示矩阵$\tilde{\boldsymbol{A}}$的条件数如下

$$\mathrm{cond}(\tilde{\boldsymbol{A}}) = \frac{\sigma_{\max}^2 + \lambda^2}{\sigma_{\min}^2 + \lambda^2} \tag{8-86}$$

其中，$\sigma_{\max}$和$\sigma_{\min}$分别是矩阵$\boldsymbol{A}$的最大和最小奇异值。例如，如果矩阵$\boldsymbol{A}$的最大和最小奇异值分别是$\sigma_{\max} = 5$以及$\sigma_{\min} = 0.1$，那么$\boldsymbol{A}^T\boldsymbol{A}$的条件数就是2 500。如果正则化参数设置为$\lambda = 1/\sqrt{2}$，则式

363 (8-86) 中的矩阵$\tilde{A}$的条件数就是50。因此，正则化参数通过因数50改善了（减小）条件数。我们打算再次利用奇异值分解，确定式（8-85）的解。将式（8-81）代入式（8-85），从而得到

$$(\boldsymbol{V}\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{V}^T + \lambda^2\boldsymbol{I})\boldsymbol{x} = \boldsymbol{V}\boldsymbol{S}^T\boldsymbol{U}^T\boldsymbol{b} \tag{8-87}$$

通过对式（8-87）两边首先左乘$\boldsymbol{V}^T$，然后再乘$(\boldsymbol{S}^T\boldsymbol{S})^{-1}$（假定此时矩阵$\boldsymbol{A}$是满秩的，从而矩阵$\boldsymbol{A}$的逆存在），最终再乘$\boldsymbol{V}$，得到

$$[\boldsymbol{I} + \lambda^2\boldsymbol{V}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{V}^T]\boldsymbol{x} = \boldsymbol{V}\boldsymbol{S}^+\boldsymbol{U}^T\boldsymbol{b} \tag{8-88}$$

其中，$\boldsymbol{S}^+$是矩阵$\boldsymbol{S}$的转置矩阵，伪对角线上为非零奇异值的倒数（参照A.2.14节）。式（8-88）左侧可以写作如下

$$\begin{aligned}[\boldsymbol{I} + \lambda^2\boldsymbol{V}(\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{V}^T]\boldsymbol{x} &= \left(\sum_{i=1}^{n}\boldsymbol{v}_i\boldsymbol{v}_i^T + \lambda^2\sum_{i=1}^{n}\frac{\boldsymbol{v}_i\boldsymbol{v}_i^T}{\sigma_i^2}\right)\boldsymbol{x} \\ &= \sum_{i=1}^{n}\boldsymbol{v}_i\left(1+\frac{\lambda^2}{\sigma_i^2}\right)\boldsymbol{v}_i^T\boldsymbol{x} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{x}\end{aligned} \tag{8-89}$$

其中

$$\boldsymbol{\Sigma} = \begin{bmatrix} 1+\dfrac{\lambda^2}{\sigma_1^2} & 0 & 0 & \cdots & 0 \\ 0 & 1+\dfrac{\lambda^2}{\sigma_2^2} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & 1+\dfrac{\lambda^2}{\sigma_n^2} \end{bmatrix}_{n\times n} \tag{8-90}$$

因此，利用式（8-89），方程（8-88）可以写作

$$\boldsymbol{V}\sum\boldsymbol{V}^T\boldsymbol{x} = \boldsymbol{V}\boldsymbol{S}^+\boldsymbol{U}^T\boldsymbol{b} \tag{8-91}$$

通过对式（8-91）两边首先左乘$\boldsymbol{V}^T$，再乘$\sum^{-1}$，最后乘$\boldsymbol{V}$，有

$$\boldsymbol{x}_\lambda = \boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{S}^+\boldsymbol{U}^T\boldsymbol{b} = \boldsymbol{V}\begin{bmatrix} \dfrac{1}{\sigma_1+\dfrac{\lambda^2}{\sigma_1}} & 0 & 0 & \cdots \\ 0 & \dfrac{1}{\sigma_2+\dfrac{\lambda^2}{\sigma_2}} & 0 & \cdots \\ \vdots & & \ddots & \end{bmatrix}_{n\times n}\boldsymbol{U}^T\boldsymbol{b} \tag{8-92}$$

或者

$$\boldsymbol{x}_\lambda = \sum_{i=1}^{n}\frac{\boldsymbol{v}_i\boldsymbol{u}_i^T\boldsymbol{b}}{\sigma_i + \lambda^2/\sigma_i} \tag{8-93}$$

364 或

$$\boldsymbol{x}_\lambda = \sum_{i=1}^{n}\left(\frac{\sigma_i^2}{\sigma_i^2+\lambda^2}\right)\frac{\boldsymbol{v}_i\boldsymbol{u}_i^T}{\sigma_i}\boldsymbol{b} \tag{8-94}$$

如果定义

$$\beta_i \triangleq \boldsymbol{u}_i^T \boldsymbol{b} \tag{8-95}$$

那么式（8-94）也可以写作如下

$$\boldsymbol{x}_\lambda = \sum_{i=1}^{n} \left( \frac{\beta_i}{\sigma_i + \lambda^2 / \sigma_i} \right) \upsilon_i \tag{8-96}$$

从式（8-94）或者式（8-96）的结果中可以观察到：

1. 如果正则化参数置为0，也就是$\lambda = 0$，那么式（8-94）可以归纳出式（8-82）中的结果，即$\boldsymbol{x}_0 = \sum_{i=1}^{n} [(\upsilon_i \boldsymbol{u}_i^T / \sigma_i)\, \boldsymbol{b}]$。

2. 对于正则化参数$\lambda > 0$在已定义范围内的值，利用式（8-94）可以推导出一系列的解。理想的情况是定义一个标准，从而可以从一系列容许的解中选择一个合适的解。一种方法可以用于合适解的选择，就是$L$曲线方法[44, 46, 50, 52]，也就是Tikhonov参数$\lambda$的选择。其他的方法可以在[53-55]找到。

3. 如果$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$中的$\boldsymbol{b}$没有受到干扰，$\boldsymbol{A}$是病态的，且具有病态确定数值秩，那么它就满足离散皮卡（Picard）条件（DPC）[45, 48, 53]，当且仅当$|\boldsymbol{u}_i^T \boldsymbol{b}|$较之奇异值$\sigma_i$（其中$i = 1, 2, \cdots, n$）（平均）衰减到0更快，也就是序列

$$\frac{|\boldsymbol{u}_1^T \boldsymbol{b}|}{\sigma_1}, \frac{|\boldsymbol{u}_2^T \boldsymbol{b}|}{\sigma_2}, \cdots, \frac{|\boldsymbol{u}_n^T \boldsymbol{b}|}{\sigma_n} \tag{8-97}$$

是（对大部分情况而言）单调减少。$\boldsymbol{u}_i$是矩阵$\boldsymbol{A}$的左奇异向量。

4. 正则化方法在$\lambda$的范围内对可能解施加了一个弱的平滑度约束。对比式（8-94）和式（8-82），发现正则化参数$\lambda$的作用是缓冲或者过滤掉奇异值之和小于$\lambda$近似值的项。在实际应用中，正则化参数将满足以下不等式

$$\sigma_n \leqslant \lambda \leqslant \sigma_1 \tag{8-98}$$

可以定义过滤因数如下

$$f_i(\lambda) = \frac{\sigma_i^2}{\sigma_i^2 + \lambda^2} \tag{8-99}$$

其中$i = 1, 2, \cdots, n$。因此，使用式（8-99），方程（8-94）可以写作如下形式

$$\boldsymbol{x}_\lambda = \sum_{i=1}^{n} f_i(\lambda) \frac{\upsilon_i \boldsymbol{u}_i^T}{\sigma_i} \boldsymbol{b} \tag{8-100}$$

365

当选取合适的Tikhonov参数时，式（8-99）中的过滤因数可以认为趋近于0，即，来自较小的$\sigma_i$的解$\boldsymbol{x}_\lambda$的作用$(\beta_i / \sigma_i)\, \boldsymbol{\upsilon}_i$（其中$\beta_i = \boldsymbol{u}_i^T \boldsymbol{b}$）被过滤掉了。当DPC不成立时，这将具有消除解中误差污染的效果。当$\sigma_i < \lambda$时，较小的奇异值的过滤功能开始起作用。另一种方式可以写作如下

$$\left( \frac{\sigma_i^2}{\sigma_i^2 + \lambda_2} \right) \frac{\upsilon_i \boldsymbol{u}_i^T}{\sigma_i} \boldsymbol{b}$$

在式（8-94）中有

$$\frac{1/\sigma_i}{1 + (\lambda / \sigma_i)^2} \beta_i \upsilon_i$$

并且考虑当某个奇异值$\sigma_i$小于$\lambda$很多的情况。此时，当$\sigma_i \to 0$，可以得到（使用l'Hôpital规则）

$$\lim_{\sigma_i \to 0} \frac{1/\sigma_i}{1+(\lambda/\sigma_i)^2}\beta_i \upsilon_i = \lim_{\sigma_i \to 0} \frac{\sigma_i}{2\lambda^2}\beta_i \upsilon_i = 0 \quad \text{对于} \sigma_i \ll \lambda \tag{8-101}$$

因此，在式（8-94）总和中相关项将接近于**0**，假设在奇异值分解（SVD）总和中的项具有合适的过滤或者自然截断，这表明对于一个真实的物理系统，需要其解具有连续性。

Tikhonov正则化方法的一种变化包括形式如下式的能量函数

$$\mathscr{E}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2 + \frac{\lambda^2}{2}\|\boldsymbol{L}\boldsymbol{x}\|_2^2 \tag{8-102}$$

其中$\boldsymbol{L}$是正则矩阵。再次求问题 $\min_{x\in\Re^n}\mathscr{E}(\boldsymbol{x})$ 的解$\boldsymbol{x} = \boldsymbol{x}_\lambda$。正则化矩阵可以采用导数算子的离散近似值，一般情况下$\boldsymbol{L}_p \in \Re^{(n-p)\times n}$，其中$p$是导数的阶数，并且$\rho(\boldsymbol{L}_p) = n-p$。例如，一阶导数矩阵如下

$$\boldsymbol{L}_1 = \begin{bmatrix} -1 & 1 & 0 & 0 & & \cdots & 0 \\ 0 & -1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & -1 & 1 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{bmatrix}_{(n-1)\times n} \tag{8-103}$$

366 且二阶导数矩阵可以写作

$$\boldsymbol{L}_2 = \begin{bmatrix} 1 & -2 & 1 & 0 & & & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & & \cdots & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 0 & 1 & -2 & 1 \end{bmatrix}_{(n-2)\times n} \tag{8-104}$$

对于$\boldsymbol{x}$最小化式（8-102）（即，计算式（8-102）关于$\boldsymbol{x}$的梯度并令结果等于0）将会导出另一系列经修正的正规方程如下

$$(\boldsymbol{A}^T\boldsymbol{A} + \lambda^2\boldsymbol{L}^T\boldsymbol{L})\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b} \tag{8-105}$$

正如前面求解$\boldsymbol{L} = \boldsymbol{I}_{n\times n}$，在只使用$\boldsymbol{A}$的奇异值分解的情况下，打算对不同的$\lambda$值求解式（8-105）。但是，这不可能有结果。存在两种设计好的方法求解这个问题：转化成标准的正则化问题以及使用广义的奇异值分解（GSVD）[25]。关于这两种方法的综述请见Varah[38]和Hansen[46]的相关论文。

**例8.4**　在矩阵$\boldsymbol{A}$是病态时，对比使用截断的奇异值分解（TSVD）方法与Tikhonov正则化方法求解$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$的结果。令矩阵$\boldsymbol{A}$是一个$20\times 20$希尔伯特（Hilbert）矩阵（参照A.2.19节），其中元素由式$a_{ij} = 1/(i+j-1)$给出，$i,j = 1,2,\cdots,n(n=20)$。因此，

$$\boldsymbol{A} = \begin{bmatrix} 1/1 & 1/2 & \cdots & 1/n \\ 1/2 & 1/3 & \cdots & 1/(n+1) \\ \cdots & \cdots & \cdots & \cdots \\ 1/n & 1/(n+1) & \cdots & 1/(2n+1) \end{bmatrix} \tag{8-106}$$

为了评估由以上两种方法得到的结果的质量，做一个有意义的对比，我们设计一个已知的（精确的）求解方案

$$x_e(j) = \sqrt{0.5j} \quad j = 1,2,\cdots,20 \tag{8-107}$$

因此，利用式（8-106）的矩阵$A$以及式（8-107）中的$x_e$，向量$b$可以写作$b = Ax_e$。（使用MATLAB得到的）矩阵$A$的条件数是$\text{cond}(A) = \sigma_{\max}(A)/\sigma_{\min}(A) = 1.0675 \times 10^{19}$，同时求得秩为$\rho(A) = 13$。

首先使用TSVD方法求解。回忆在正则化方法中，试图寻求解$x$，从而使$\|x\|_2$与$\|Ax - b\|_2$一同相对较小。图8-11a描述了$x_k$相对于SVD截断参数$k$（见式（8-83）（$k = 1, 2, \cdots, 20$））的$L_2$范数，即$\|x_k\|_2$的图。在图中看到曲线是相对平直的，直到（包括）$k = 14$；在$k = 14$后曲线迅速上升。在$k = 14$处的转角把曲线分成了两部分。在图8-11a中，曲线左侧的部分（通常地相当于信号）相对平直，而右边的部分（相当于噪声）非常陡峭。曲线平滑的部分符合$\|x_k\|_2$的值极其接近于$x_e$的$L_2$范数$\|x_e\|_2$。因此，可能尝试使用一个$k = 14$的截断参数。实际上最小的相对误差出现在$k = 11$处。在图8-11b中，以相对误差和截断参数为坐标绘制。因此，$k = 11$是使用TSVD的截断参数最好的值，同时，相对误差如下给出 367

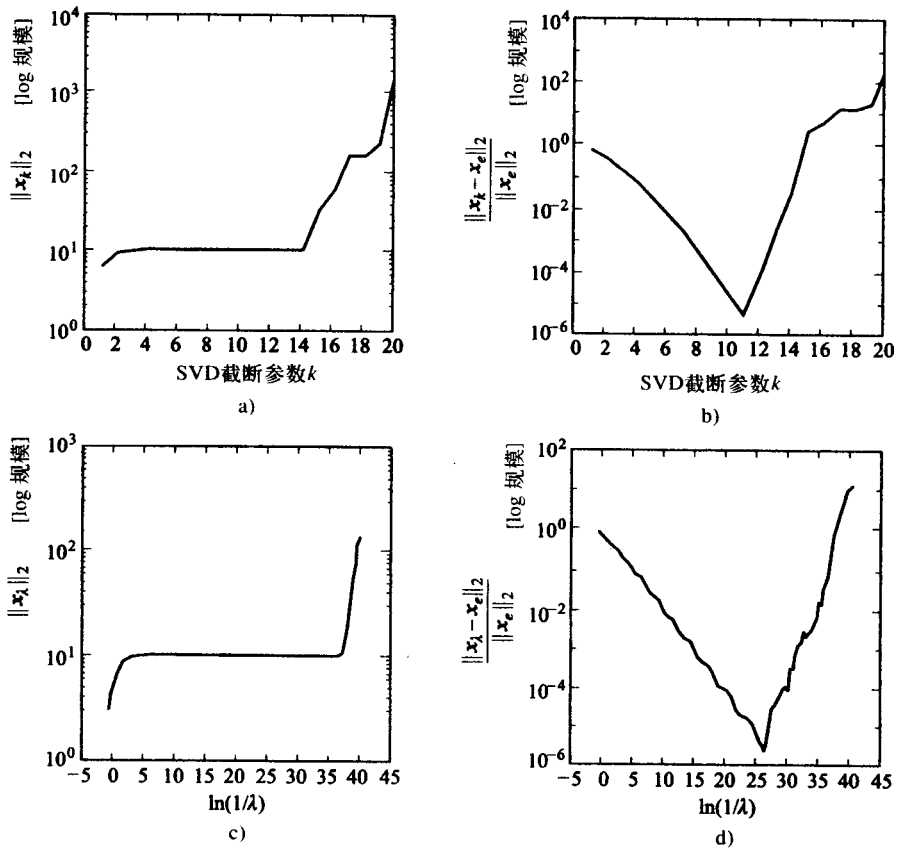



图8-11　a) $x_k$的$L_2$范数（$\|x_k\|_2$）对比TSVD方法的SVD截断参数$k$（$k = 1, 2, \cdots, 20$）；b) 对比TSVD方法截断参数$k$的相对误差；c) $x_\lambda$的$L_2$范数（$\|x_\lambda\|_2$）对比Tikhonov正则化方法中的ln(1/λ)（λ为正则化参数）；d) 对比Tikhonov正则化方法中的ln(1/λ)的相对误差

$$\left.\frac{\|x_k - x_e\|_2}{\|x_e\|_2}\right|_{k=11} = 4.735\ 258\ 615\ 542\ 855 \times 10^{-6} \tag{8-108}$$

对于Tikhonov正则化方法，必须对正则化参数λ生成值的范围。为$\lambda_1 \sim \lambda_\ell$之间的λ赋值。令 $\lambda_1 = \sigma_{\max}(A)$以及$\lambda_\ell < \max\{\sigma_{\min}(A), \sigma_{\max}(A) \cdot \varepsilon\}$，其中ε是机器舍入修整单元。在MATLAB中，它被选取为$eps = 2^{-52} = 2.220\ 446\ 049\ 250\ 313 \times 10^{-16}$。使用MATLAB内建函数svd计算的矩阵$A$的奇异值如表8-2所示。要求一个$\ell = 100$个点的“网格”，λ的两个端点为$\lambda_1 = \sigma_1 = 1.907\ 134\ 720\ 407\ 253$以及$\lambda_\ell = \sigma_1 \cdot eps/100 = 4.234\ 689\ 755\ 316\ 386 \times 10^{-18}$。余下的98（$\ell - 2$）个点，也就是$\lambda_2 \sim \lambda_{\ell-1}$可以从下式中得到[50，53]

368

$$\lambda_q = \lambda_1 \left(\frac{\lambda_\ell}{\lambda_1}\right)^{(q-1)/(\ell-1)} \qquad q = 1,2,\cdots,\ell \tag{8-109}$$

对于这100个λ值，利用式（8-94）可以产生100个“解”。图8-11c描述了$\boldsymbol{x}_\lambda$的$L_2$范数（即$\|\boldsymbol{x}_\lambda\|_2$）与ln(1/λ)。注意这条曲线同应用TSVD方法的图8-11a很相似。同样可以发现，曲线的平滑部分对应的$\|\boldsymbol{x}_\lambda\|_2$的值同$\boldsymbol{x}_e$的$L_2$范数（$\|\boldsymbol{x}_e\|_2$）很接近。图8-11d描述了对比Tikhonov正则化方法中的ln(1/λ)的相对误差。图中显示的明确定义的最小值对应于最优的正则化参数$\lambda_0 = 3.245\ 947\ 737\ 964\ 136 \times 10^{-12}$以及相对误差

$$\left.\frac{\|\boldsymbol{x}_\lambda - \boldsymbol{x}_e\|_2}{\|\boldsymbol{x}_e\|_2}\right|_{\lambda=\lambda_0} = 2.673\ \ 028\ \ 585\ \ 143\ \ 722 \times 10^{-6} \tag{8-110}$$

这一结果比式（8-108）的TSVD方法的相对误差好一些。注意最优正则化参数位于奇异值11～12之间，见表8-2。因此，式（8-94）中全部的过滤项在这一值后会出现，因为奇异值12～20比$\lambda_0$小。

**表8-2　使用MATLAB（5.1版）内建函数svd计算的矩阵$A$的奇异值**

| $i$ | $\sigma_i$ | $i$ | $\sigma_i$ |
|---|---|---|---|
| 1 | 1.907 134 720 407 253 | 11① | $2.192\ 890\ 048\ 019\ 410 \times 10^{-11}$ |
| 2 | $4.870\ 384\ 065\ 720\ 490 \times 10^{-1}$ | 12 | $6.740\ 801\ 127\ 335\ 105 \times 10^{-13}$ |
| 3 | $7.559\ 582\ 130\ 544\ 090 \times 10^{-2}$ | 13 | $1.738\ 400\ 044\ 807\ 469 \times 10^{-14}$ |
| 4 | $8.961\ 128\ 614\ 856\ 439 \times 10^{-3}$ | 14 | $3.740\ 758\ 559\ 585\ 443 \times 10^{-16}$ |
| 5 | $8.676\ 711\ 091\ 714\ 799 \times 10^{-4}$ | 15 | $1.704\ 893\ 499\ 875\ 155 \times 10^{-17}$ |
| 6 | $7.033\ 431\ 473\ 193\ 232 \times 10^{-5}$ | 16 | $1.464\ 577\ 158\ 596\ 144 \times 10^{-17}$ |
| 7 | $4.830\ 510\ 048\ 804\ 696 \times 10^{-6}$ | 17 | $9.694\ 815\ 980\ 744\ 144 \times 10^{-18}$ |
| 8 | $2.827\ 652\ 055\ 224\ 344 \times 10^{-7}$ | 18 | $8.115\ 826\ 272\ 543\ 485 \times 10^{-18}$ |
| 9 | $1.413\ 954\ 758\ 555\ 360 \times 10^{-8}$ | 19 | $1.760\ 134\ 198\ 986\ 154 \times 10^{-18}$ |
| 10 | $6.036\ 095\ 327\ 608\ 074 \times 10^{-10}$ | 20 | $1.786\ 569\ 618\ 132\ 934 \times 10^{-19}$ |

注：①为TSVD方法保留的奇异值的最优数。

如果使用MATLAB内建的函数pinv计算矩阵$A$的伪逆，再计算$\boldsymbol{Ax} = \boldsymbol{b}$的解，也就是$\boldsymbol{x}^M = pinv(\boldsymbol{A})\boldsymbol{b}$，此时，结果的相对误差就是

$$\frac{\|\boldsymbol{x}^M - \boldsymbol{x}_e\|_2}{\|\boldsymbol{x}_e\|_2} = 2.330\ \ 150\ \ 590\ \ 018\ \ 773 \times 10^{-3} \tag{8-111}$$

369 这一结果相对于使用TSVD方法以及Tikhonov正则化方法的相对误差要差3个数量级。pinv计算矩阵$A$的伪逆实际使用的方法是SVD。用作截断奇异值的默认的公差值（小于公差值）可以通过max(size(A))*norm(A)*eps进行计算。因此，这样的问题默认的公差值是

$20*\sigma_1*eps = 8.469\,379\,510\,632\,772 \times 10^{-15}$。在表8-2中，这一默认的公差值分布在奇异值13～14之间。因此，在式（8-83）中总共使用13项去计算解$x^M$。通过MATLAB的rank函数，应用相同的默认公差值基于矩阵$A$的SVD确定$A$的秩。

## 8.8　求解线性方程组的离散时间迭代方法中的矩阵分裂

存在四种基本的迭代离散时间方法来解如下形式的线性方程组

$$Ax = b \tag{8-112}$$

其中矩阵$A \in \Re^{n\times n}$是基于矩阵分裂[25，29，56-58]。当且仅当矩阵$A$是非奇异的，解向量$x$存在并且是唯一的。所有的方法都具有基本形式

$$Mx(k+1) = Nx(k) + b \tag{8-113}$$

其中$k$是离散时间指数。可以把矩阵$A$表达成矩阵的和

$$A = D - E - F \tag{8-114}$$

其中，$D \in \Re^{n\times n}$是对角矩阵，也就是，$d = \mathrm{diag}[a_{11}, a_{22}, \cdots, a_{nn}]$，$E \in \Re^{n\times n}$与$F \in \Re^{n\times n}$分别是严格的下三角矩阵和上三角矩阵。矩阵$E$和$F$的元素是矩阵$A$相应元素的负数，分别对应于矩阵$A$主对角线的下上部分。矩阵$D$的全部对角元素假设为非零。

**雅可比迭代方法**

将式（8-114）代入式（8-112），有

$$(D - E - F)x = b \tag{8-115}$$

分裂成

$$Dx = (E + F)x + b \tag{8-116}$$

式（8-116）的迭代方案可以写作如下

$$Dx(k+1) = (E + F)x(k) + b \tag{8-117}$$

因此，根据式（8-113），$M = D$以及$N = E + F$。因为矩阵$D$的对角元素非零，可以把式（8-117）写作

$$x(k+1) = D^{-1}(E + F)x(k) + D^{-1}b \tag{8-118}$$

其中$k \geqslant 0$，$x(0)$为已知初始条件向量。这就是向量矩阵形式的雅可比迭代方法[56]，可以把以下矩阵 370

$$B = D^{-1}(E + F) \tag{8-119}$$

称作雅可比矩阵。式（8-118）的标量形式可以写作

$$x_i(k+1) = -\sum_{\substack{j=1 \\ j\neq i}}^{n}\left(\frac{a_{ij}}{a_{ii}}\right)x_j(k) + \frac{b_i}{a_{ii}} \tag{8-120}$$

其中，$1 \leqslant i \leqslant n$，$k \geqslant 0$，已知的初始条件为$x_i(0)$。从式（8-120）中看到，一般情况下，当计算向量$x(k+1)$的元素时，所有的向量$x(k)$的元素必须保存。但是，使用解向量$x$的元素$x_i$的最近估计$x_i(k+1)$，进行随后的计算似乎是有道理的。这样就导出了这一类中的第二种方法。

高斯－赛德尔迭代法

从式（8-115）$(D - E - F)x = b$开始，对表达式重排（即，拆分）会有

$$(\boldsymbol{D}-\boldsymbol{E})\boldsymbol{x}=\boldsymbol{F}\boldsymbol{x}+\boldsymbol{b} \tag{8-121}$$

其中，$\boldsymbol{D}-\boldsymbol{E}$是一个非奇异下三角矩阵。从式（8-121）可以得到迭代方案如下

$$(\boldsymbol{D}-\boldsymbol{E})\boldsymbol{x}(k+1)=\boldsymbol{F}\boldsymbol{x}(k)+\boldsymbol{b} \tag{8-122}$$

或者

$$\boldsymbol{x}(k+1)=(\boldsymbol{D}-\boldsymbol{E})^{-1}\boldsymbol{F}\boldsymbol{x}(k)+(\boldsymbol{D}-\boldsymbol{E})^{-1}\boldsymbol{b} \tag{8-123}$$

其中，$k\geqslant 0$，且已知的初始条件向量为$\boldsymbol{x}(0)$。对比式（8-122）与式（8-113）可得，$\boldsymbol{M}=\boldsymbol{D}-\boldsymbol{E}$以及$\boldsymbol{N}=\boldsymbol{F}$。这就是向量矩阵形式的高斯－赛德尔迭代方法[56]，矩阵$\boldsymbol{C}=(\boldsymbol{D}-\boldsymbol{E})^{-1}\boldsymbol{F}$称作高斯－赛德尔矩阵。高斯－赛德尔迭代方法的标量形式可以写成$\boldsymbol{D}\boldsymbol{x}(k+1)=\boldsymbol{E}\boldsymbol{x}(k+1)+\boldsymbol{F}\boldsymbol{x}(k)+\boldsymbol{b}$以及下式

$$x_i(k+1)=-\frac{1}{a_{ii}}\sum_{j=1}^{i-1}a_{ij}x_j(k+1)-\frac{1}{a_{ii}}\sum_{j=i+1}^{n}a_{ij}x_j(k)+\frac{b_i}{a_{ii}} \tag{8-124}$$

其中，$1\leqslant i\leqslant n$，$k\geqslant 0$，初始条件为$x_i(0)$。

逐次超松弛迭代法

在逐次超松弛（SOR）迭代法的情况中，可以把矩阵$\boldsymbol{A}$分裂成

$$\boldsymbol{A}=\boldsymbol{M}_\omega-\boldsymbol{N}_\omega=\boldsymbol{D}-\boldsymbol{E}-\boldsymbol{F} \tag{8-125}$$

其中

371

$$\boldsymbol{M}_\omega=\frac{1}{\omega}(\boldsymbol{D}-\omega\boldsymbol{E}) \tag{8-126}$$

以及

$$\boldsymbol{N}_\omega=\frac{1}{\omega}[(1-\omega)\boldsymbol{D}+\omega\boldsymbol{F}] \tag{8-127}$$

参数$\omega$称作松弛因数。因此，将式（8-126）以及式（8-127）代入式（8-113）可得

$$\frac{1}{\omega}(\boldsymbol{D}-\omega\boldsymbol{E})\boldsymbol{x}(k+1)=\frac{1}{\omega}[(1-\omega)\boldsymbol{D}+\omega\boldsymbol{F}]\boldsymbol{x}(k)+\boldsymbol{b} \tag{8-128}$$

式（8-128）两边同乘以$\omega$，然后，两边同时左乘以$\boldsymbol{D}^{-1}$可得

$$(\boldsymbol{I}-\omega\boldsymbol{D}^{-1}\boldsymbol{E})\boldsymbol{x}(k+1)=[(1-\omega)\boldsymbol{I}+\omega\boldsymbol{D}^{-1}\boldsymbol{F}]\boldsymbol{x}(k)+\omega\boldsymbol{D}^{-1}\boldsymbol{b} \tag{8-129}$$

现在定义$\boldsymbol{L}\triangleq\boldsymbol{D}^{-1}\boldsymbol{E}$（严格下三角矩阵）以及$\boldsymbol{U}\triangleq\boldsymbol{D}^{-1}\boldsymbol{F}$（严格上三角矩阵），然后将它们代入式（8-129）可得

$$(\boldsymbol{I}-\omega\boldsymbol{L})\boldsymbol{x}(k+1)=[(1-\omega)\boldsymbol{I}+\omega\boldsymbol{U}]\boldsymbol{x}(k)+\omega\boldsymbol{D}^{-1}\boldsymbol{b}$$

两边同时左乘以$(\boldsymbol{I}-\omega\boldsymbol{L})^{-1}$，推导出

$$\boldsymbol{x}(k+1)=(\boldsymbol{I}-\omega\boldsymbol{L})^{-1}[(1-\omega)\boldsymbol{I}+\omega\boldsymbol{U}]\boldsymbol{x}(k)+\omega(\boldsymbol{I}-\omega\boldsymbol{L})^{-1}\boldsymbol{D}^{-1}\boldsymbol{b} \tag{8-130}$$

其中$k\geqslant 0$，且已知的初始条件向量为$\boldsymbol{x}(0)$。方程（8-130）即是向量矩阵形式的$SOR$迭代方法[56]。矩阵

$$\mathscr{L}_\omega=(\boldsymbol{I}-\omega\boldsymbol{L})^{-1}[(1-\omega)\boldsymbol{I}+\omega\boldsymbol{U}] \tag{8-131}$$

称作逐次松弛矩阵。如果松弛因数分布于$0\leqslant\omega\leqslant 1$的范围，此时称作低松弛。但是，如果$\omega>1$，就称作超松弛。注意，如果松弛因数置为$\omega=1$，那么式（8-130）就可以转化成式（8-123）

中向量矩阵形式的高斯-赛德尔方法。标量形式的SOR迭代方法可以从式（8-128）推导出，写作$\boldsymbol{D}\boldsymbol{x}(k+1)=\omega\boldsymbol{E}\boldsymbol{x}(k+1)+(1-\omega)\boldsymbol{D}\boldsymbol{x}(k)+\omega\boldsymbol{F}\boldsymbol{x}(k)+\omega\boldsymbol{b}$。标量形式的SOR迭代方法可以写作

$$x_i(k+1)=\frac{\omega}{a_{ii}}\left[b_i-\sum_{j=1}^{i-1}a_{ij}x_j(k+1)-\sum_{j=i+1}^{n}a_{ij}x_j(k)\right]+(1-\omega)x_i(k) \tag{8-132}$$

其中，$1\leqslant i\leqslant n$，$k\geqslant 0$，同时已知的初始条件为$x_i(0)$。图8-12描述了利用SOR迭代方法实现的神经网络体系结构。

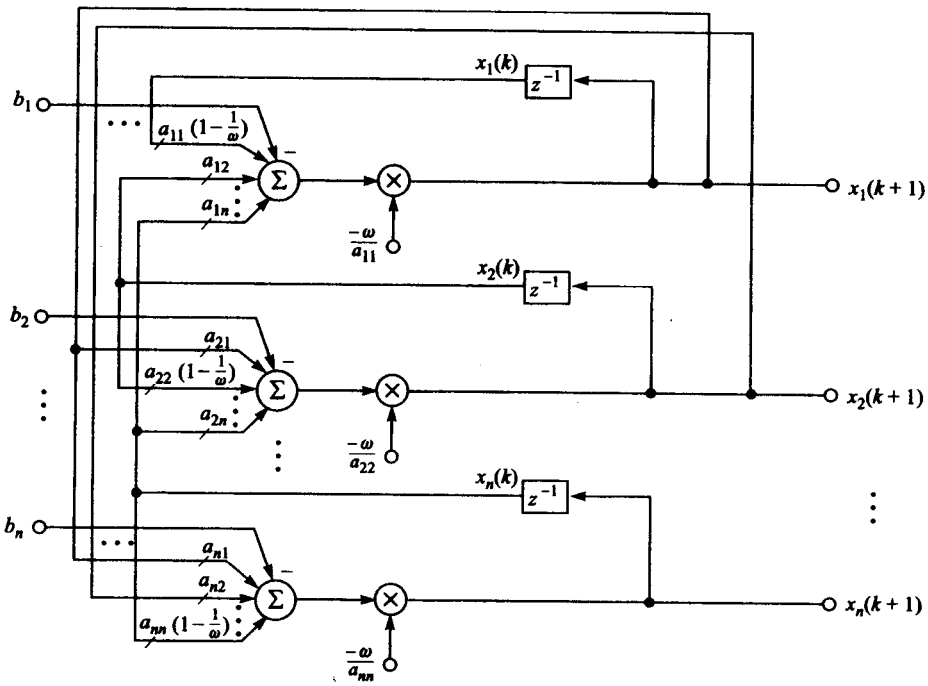



图8-12　使用SOR迭代方法的线性神经网络体系结构

使用式（8-131）中的逐次松弛矩阵的定义，方程（8-130）可以写作

$$\boldsymbol{x}(k+1)=\mathscr{L}(\omega)\boldsymbol{x}(k)+\boldsymbol{R}\boldsymbol{b} \tag{8-133}$$

其中$\boldsymbol{R}\triangleq\omega(\boldsymbol{I}-\omega\boldsymbol{L})^{-1}\boldsymbol{D}^{-1}$。现在定义误差向量如下

$$\boldsymbol{e}(k)=\boldsymbol{x}(k)-\boldsymbol{x}\qquad k\geqslant 0 \tag{8-134}$$

其中，$\boldsymbol{x}$是式（8-112）唯一的向量解。对于这个误差，从式（8-133）可以导出一个齐次误差差分方程

$$\boldsymbol{e}(k+1)=\mathscr{L}(\omega)\boldsymbol{e}(k) \tag{8-135}$$

可以选择松弛因数$\omega$来最小化$\sigma_r[\mathscr{L}(\omega)]$，从而使得$\boldsymbol{x}(k)$尽可能快地[58]收敛于$\boldsymbol{x}$，其中$\sigma_r(\cdot)$是$\mathscr{L}(\omega)$的谱半径（参见A.2.13节）。松弛因数的最优值称为$\omega^o$。$\omega^o$的计算一般是很困难的，除了在简单情况下。通常情况下，通过尝试$\omega$的不同值观察对收敛速度的影响，从而近似确定$\omega^o$。即使考虑计算$\omega^o$的问题，这样的努力还是值得的，因为这一结果极大地提高了$\boldsymbol{x}(k)$收敛到$\boldsymbol{x}$的速度。

理查森迭代方法

另一种可以考虑用于求解方程组的迭代技术是理查森（Richardson）迭代方法[57]。基本思想就是迭代，直到解的离散时间近似值的一阶导数的负数接近于0，也就是，

372 ~ 373

$$-\frac{\boldsymbol{x}(k+1)-\boldsymbol{x}(k)}{\beta}=\boldsymbol{A}\boldsymbol{x}(k)-\boldsymbol{b} \tag{8-136}$$

其中$\boldsymbol{e}(k)=\boldsymbol{A}\boldsymbol{x}(k)-\boldsymbol{b}$以及$k\geqslant 0$。经过整理后上式如下

$$\boldsymbol{x}(k+1)=\boldsymbol{x}(k)-\beta(k)[\boldsymbol{A}\boldsymbol{x}(k)-\boldsymbol{b}] \tag{8-137}$$

其中$\beta=\beta(k)$，$\boldsymbol{x}(0)$是已知的初始条件向量，最优迭代参数可以如下[8]确定

$$\beta(k)=\frac{\boldsymbol{e}^T(k)\boldsymbol{e}(k)}{\boldsymbol{e}^T(k)\boldsymbol{A}\boldsymbol{e}(k)} \tag{8-138}$$

可以从式（8-137）中的向量矩阵形式推导出标量形式的理查森迭代方法如下

$$x_i(k+1)=x_i(k)-\beta(k)\left[\sum_{j=1}^{n}a_{ij}x_j(k)-b_i\right] \tag{8-139}$$

其中$1\leqslant i\leqslant n$，$k\geqslant 0$，已知初始条件为$x_i(0)$。如果选择式（8-139）中的$\beta(k)=1/a_{ii}$，可以把迭代表达式写作如下形式

$$x_i(k+1)=x_i(k)-\frac{1}{a_{ii}}\left[\sum_{j=1}^{n}a_{ij}x_j(k)-b_i\right] \tag{8-140}$$

也可依次写作

$$x_i(K+1)=-\sum_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{a_{ij}}{a_{ii}}\right)x_j(k)+\frac{b_i}{a_{ii}} \tag{8-141}$$

其中$1\leqslant i\leqslant n$，$k\geqslant 0$，已知初始条件为$x_i(0)$。实际上这正是式（8-120）中给出的雅可比迭代方法。

**例8.5**　对比本节所提到的求解$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$形式的线性方程组的四种方法中的三种的性能。在这个例子中系统如下

$$\begin{bmatrix}6&5&5&6\\5&9&4&3\\5&4&10&5\\6&3&5&7\end{bmatrix}\boldsymbol{x}=\begin{bmatrix}55\\47\\63\\55\end{bmatrix} \tag{8-142}$$

因为矩阵$\boldsymbol{A}$的条件数相对较小[cond($\boldsymbol{A}$) = 394.874 2]，因此，式（8-142）中的方程组可以使用MATLAB中的`inv`函数求解。结果如下

$$\boldsymbol{x}^M=inv(\boldsymbol{A})^*\boldsymbol{b}=\begin{bmatrix}1.0000\\2.0000\\3.0000\\4.0000\end{bmatrix} \tag{8-143}$$

逐次超松弛迭代法需要确定松弛参数$\omega$的最优值。根据本节提及的SOR方法的计算过程，必须求解作为$\omega$的函数的$\sigma_r$ [$\mathscr{L}(\omega)$]的最小值。图8-13a描述了作为$\omega$的函数的$\sigma_r$ [$\mathscr{L}(\omega)$]的图，$\omega$值的分布范围如下：$0\leqslant\omega\leqslant 2.5$，且$\Delta\omega=1/10\,000$。定义好的最小值导出了由$\omega^o=1.721\,5$给出的松弛参

数的最优值。三种方法都使用同样的初始条件：$x(0) = [0.011\ 8,\ 0.031\ 5,\ 0.144\ 4,\ -0.035\ 1]^T$。这些初始条件通过从均值0以及方差为0.01的高斯分布中选定4个随机数来产生。利用由图8-13a确定的松弛参数的最优值，使SOR算法得以进行，同时图8-13b反映了收敛的情况。整个算法进行了115次迭代才收敛。迭代终止的准则确定为：当绝对误差小于$10^{-7}$（也就是，$\|x^{SOR} - x^M\|2 < 10^{-7}$）达到收敛。图8-13c、d分别描述了使用理查森以及高斯−赛德尔方法解式 374
(8-142) 的收敛图。为了合理地对比所有的结果，这些方法都使用共同的结束准则。表8-3总结了仿真结果。从表8-3中，我们发现SOR迭代方法得到了最好的结果，也就是最快地达到收敛。对比SOR结果与次优的结果（高斯−赛德尔方法，$\omega = 1$），我们发现SOR方法收敛速度快约10倍。

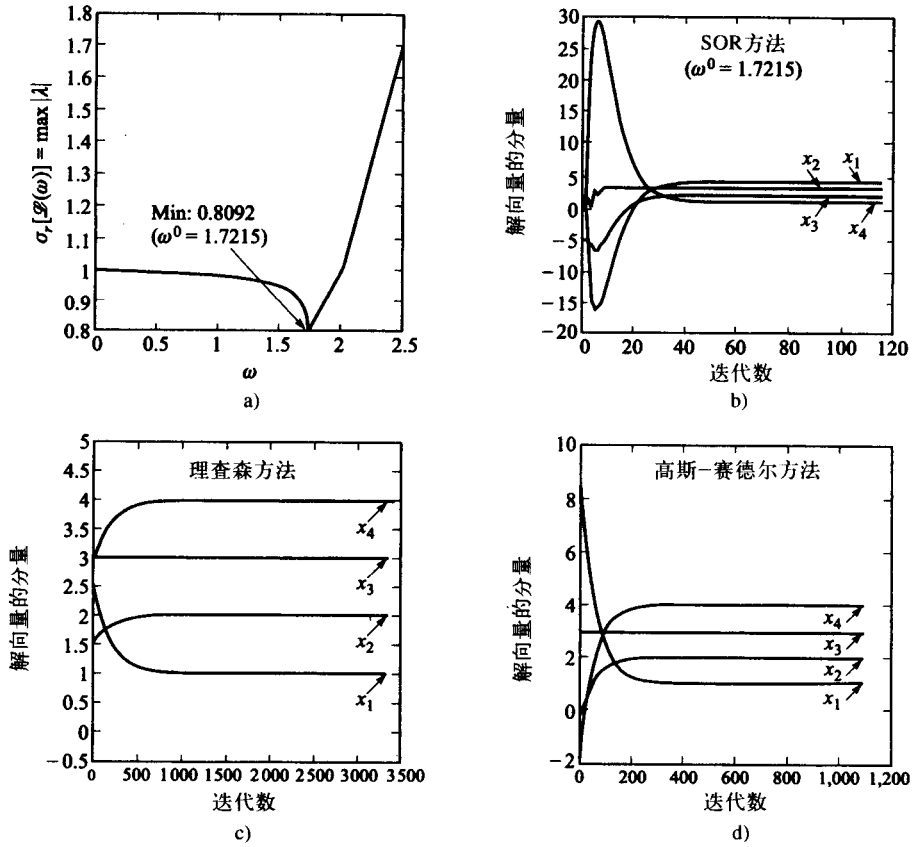



图8-13　a) 逐次松弛矩阵$\sigma_r[\mathscr{L}(\omega)]$谱半径对松弛因数$\omega$，确定$\omega$最优值的图。图中，最小值发生在$\omega^0 = 1.7215$；b) 使用SOR方法求解式（8-142）的收敛图；c) 使用理查森方法求解式（8-142）的收敛图；d) 使用高斯−赛德尔方法求解式（8-142）的收敛图

**表8-3　使用三种方法求解式（8-142）的仿真结果对比（MATLAB 5.1）**

| 方　法 | 绝对误差$\|x - x^M\|_2$ | 相对误差$\dfrac{\|x - x^M\|_2}{\|x^M\|_2}$ | 收敛所需的迭代次数 |
|---|---|---|---|
| 逐次超松弛（$\omega^0 = 1.7215$） | $2.1999 \times 10^{-8}$ | $4.0165 \times 10^{-9}$ | 115 |
| 高斯−赛德尔 | $5.3028 \times 10^{-8}$ | $9.681\,6 \times 10^{-9}$ | 1 100 |
| 理查森 | $2.7619 \times 10^{-8}$ | $5.042\,6 \times 10^{-9}$ | 3 400 |
| 雅可比 | 发散 | 发散 | 发散 |

## 8.9 总体最小二乘问题

375 如果误差被限制在$Ax = b$系统的$b$向量，那么一般的最小二乘方法就适宜用于求解系统。然而，如果误差不仅发生在数据矩阵$A$中，而且包括观察向量$b$，那么就可以利用总体最小二乘（TLS）[25, 59-62]法推导出解$x$。一般情况下假定数据矩阵$A$并未被误差干扰是不现实的。更现实的做法是假设$A$确实包含误差，因为更多时候矩阵$A$中的数据是由已经遭受噪声干扰的测量数据组成的。这些测量噪声可能是由负责收集数据的实际设备本身产生的（即，测量值）。建模误差以及量化误差也有可能干扰数据矩阵。TLS方法的主要原则可以通过首先重新用公式形式表示标准最小二乘问题来引入[25]。

最小二乘问题的一种不同观点

考虑如下式的包含$m$个线性方程的超定组

$$Ax \approx b \tag{8-144}$$

有$n$个未知量，即$A \in \Re^{m\times n}$，$b \in \Re^{m\times 1}$，$x \in \Re^{n\times 1}$，且$m > n$，最小二乘问题求

$$\underset{\tilde{b}\in\Re^{m\times 1}}{\text{Minimize}} \| b - \tilde{b} \|_2 \tag{8-145}$$

$$\text{受限于 } \tilde{b} \in \mathcal{R}(A) \tag{8-146}$$

其中$\mathcal{R}(A)$是矩阵$A$的范围[⊖]。向量$\tilde{b}$由下式给出

$$\tilde{b} = b - \tilde{e} \quad (\tilde{e} \perp A\tilde{x}) \tag{8-147}$$

其中

$$\tilde{e} = b - A\tilde{x} \tag{8-148}$$

以及

376 $$\tilde{x}\text{求解最小二乘问题} \Leftrightarrow A^T b - A^T A\tilde{x} = 0 \tag{8-149}$$

其中$\tilde{b}$是$b$在$\mathcal{R}(A)$上的正交投影。因此，从式（8-147）中我们看到

$$\tilde{e} = b - \tilde{b} \tag{8-150}$$

同样，由式（8-149），假定$\rho(A) = n$，我们发现$\tilde{x} = (A^T A)^{-1} A^T b$（满足 $\underset{x\in\Re^{n\times 1}}{\text{minimize}} \| Ax - b \|_2$ 的唯一解），等式两边左乘以矩阵$A$可以得到

$$A\tilde{x} = A(A^T A)^{-1} A^T b \tag{8-151}$$

由式（8-148）和式（8-150）$A\tilde{x} = \tilde{b}$，以及式（8-151）可以写作$\tilde{b} = A(A^T A)^{-1} A^T b$。因此有，

$$\tilde{b} = P_A b \tag{8-152}$$

以及

$$P_A = A(A^T A)^{-1} A^T \tag{8-153}$$

是在$\mathcal{R}(A)$上的正交投影算子。因此，$\tilde{b}$是b在$\mathcal{R}(A)$上的正交投影。如果$\rho(A) < n$，那么最小二乘问题 $\underset{x\in\Re^{n\times 1}}{\text{minimize}} \| Ax - b \|_2$ 有无确定数目的解。但是，为了稳定性和最小灵敏度，拥有最小$L_2$范数的唯一解是从最小值的集合中选出来的[59]

---

⊖ 矩阵$A$的范围定义为：$\mathcal{R}(A) = \{y \in \Re^{m\times 1}: y = Ax \text{对于某些} x \in \Re^{n\times 1}\}$。

$$\mathscr{X} = \{\boldsymbol{x} \in \Re^{n\times 1} : \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 = \min\} \tag{8-154}$$

这个解通过$\tilde{\boldsymbol{x}}$表示（在满秩的情况下，只存在一种最小二乘解，它必须拥有最小的$L_2$范数），如A.2.14节中指出的，$\tilde{\boldsymbol{x}}$可以经过SVD求得。

在式（8-145）和式（8-146）中，一旦发现一个最小化的$\tilde{\boldsymbol{b}}$，那么任何$x$满足

$$\boldsymbol{A}\boldsymbol{x} = \tilde{\boldsymbol{b}} \tag{8-155}$$

称作最小二乘解

$$\Delta\tilde{\boldsymbol{b}} = \boldsymbol{b} - \tilde{\boldsymbol{b}} \tag{8-156}$$

称作相应的最小二乘修正。满足方程式（8-145）和式（8-146），只须$\tilde{\boldsymbol{b}}$是$\boldsymbol{b}$在$\mathscr{R}(\boldsymbol{A})$上的正交投影。因此，最小二乘问题涉及到最小量$\Delta\tilde{\boldsymbol{b}}$干扰观察向量$\boldsymbol{b}$，因此

$$\tilde{\boldsymbol{b}} = \boldsymbol{b} - \Delta\tilde{\boldsymbol{b}} \tag{8-157}$$

可以通过数据矩阵$\boldsymbol{A}$的列进行估测。正如前面描述的，标准最小二乘问题中的基本假设是误差只发生在观察向量$\boldsymbol{b}$，并且矩阵$\boldsymbol{A}$假设为完全已知的。但是，这通常不是一个实际的假设。因此，现在计划同时考虑$\boldsymbol{b}$和$\boldsymbol{A}$的误差，并考虑总体最小二乘问题。 377

基本的总体最小二乘问题

考虑如式（8-144）中的超定组，总体最小二乘问题寻求

$$\underset{[\hat{\boldsymbol{A}}|\hat{\boldsymbol{b}}]\in\Re^{m\times(n+1)}}{\text{Minimize}} \|[\boldsymbol{A}|\boldsymbol{b}] - [\hat{\boldsymbol{A}}|\hat{\boldsymbol{b}}]\|_F \tag{8-158}$$

$$\text{受限于 } \hat{\boldsymbol{b}} \in \mathscr{R}(\hat{\boldsymbol{A}}) \tag{8-159}$$

其中$[\boldsymbol{A}|\boldsymbol{b}]_{m\times(n+1)}$是一个增广矩阵，也就是，列向量$\boldsymbol{b}$“加到”矩阵$\boldsymbol{A}$的“尾部”，$\|\cdot\|_F$是弗罗贝尼乌斯（Frobenius）范数（参照A.2.13节）。一旦找到最小的$[\hat{\boldsymbol{A}}|\hat{\boldsymbol{b}}]$值，那么任意向量$\boldsymbol{x}$都满足

$$\hat{\boldsymbol{A}}\boldsymbol{x} = \hat{\boldsymbol{b}} \tag{8-160}$$

称作TLS“解”，$[\Delta\hat{\boldsymbol{A}}|\Delta\hat{\boldsymbol{b}}] = [\boldsymbol{A}|\boldsymbol{b}] - [\hat{\boldsymbol{A}}|\hat{\boldsymbol{b}}]$是相关的TLS修正。我们使用$\hat{\boldsymbol{x}}$表示TLS方法。另一种看待这个问题的方式是找到向量$\hat{\boldsymbol{x}}$，它寻求

$$\text{最小化 } [\|\Delta\hat{\boldsymbol{A}}\|_F^2 + \|\Delta\hat{\boldsymbol{b}}\|_F^2] \tag{8-161}$$

$$\text{受限于 } (\boldsymbol{A} - \Delta\hat{\boldsymbol{A}})\hat{\boldsymbol{x}} = (\boldsymbol{b} - \Delta\hat{\boldsymbol{b}}) \tag{8-162}$$

奇异值分解（参照A.2.14节）是求解TLS问题的标准方法。方程（8-144）可以写作如下形式

$$[\boldsymbol{A}|\boldsymbol{b}][\boldsymbol{x}^T|-1]^T = \boldsymbol{0} \tag{8-163}$$

$[\boldsymbol{A}|\boldsymbol{b}]$最好的$n$秩TLS近似值$[\hat{\boldsymbol{A}}|\hat{\boldsymbol{b}}]$最小化了方差偏差，有以下形式

$$[\hat{\boldsymbol{A}}|\hat{\boldsymbol{b}}] = \boldsymbol{U}\hat{\boldsymbol{S}}\boldsymbol{V}^T \tag{8-164}$$

其中$\hat{\boldsymbol{S}} = \text{diag}[\sigma_1, \sigma_2, \cdots, \sigma_n, 0]$。最化的TLS修正如下

$$\sigma_{n+1} = \min_{\rho([\hat{\boldsymbol{A}}|\hat{\boldsymbol{b}}])=n} \|[\boldsymbol{A}|\boldsymbol{b}] - [\hat{\boldsymbol{A}}|\hat{\boldsymbol{b}}]\|_F \tag{8-165}$$

相对应的TLS修正矩阵

$$[\Delta\hat{\boldsymbol{A}}|\Delta\hat{\boldsymbol{b}}] = [\boldsymbol{A}|\boldsymbol{b}] - [\hat{\boldsymbol{A}}|\hat{\boldsymbol{b}}] = \sigma_{n+1}\boldsymbol{u}_{n+1}\boldsymbol{v}_{n+1}^T \tag{8-166}$$

求解式（8-158）和式（8-159）中的TLS问题，且

$$\hat{\boldsymbol{x}}=\frac{1}{v_{n+1,n+1}}[v_{1,n+1},v_{2,n+1},\cdots,v_{n,n+1}]^T \tag{8-167}$$

[378] 是式（8-162）的唯一解。

## 8.10 求解线性方程组的$L_\infty$范数（最小最大）神经网络

本节打算介绍一种求解$\boldsymbol{Ax}=\boldsymbol{b}$($\boldsymbol{A}\in\Re^{m\times n}$, $\boldsymbol{x}\in\Re^{n\times 1}$, $\boldsymbol{b}\in\Re^{m\times 1}$的神经网络体系结构，基于误差的$L_\infty$范数（或者Chebyshev范数）

$$e_i(\boldsymbol{x})=\sum_{j=1}^{n}a_{ij}x_j-b_i \quad i=1,2,\cdots,m \tag{8-168}$$

这也称作最小最大问题，可以通过公式表示如下：

最小化如下所示的能量函数，求解向量$\boldsymbol{x}$

$$\mathscr{E}(\boldsymbol{x})=\max_{1<i<m}\{|e_i(\boldsymbol{x})|\} \tag{8-169}$$

其中$e_i(\boldsymbol{x})$由式（8-168）给出。可以把它写得更简洁一些，如下

$$\min_{x\in\Re^n}\ \max_{1<i<m}\{|e_i(\boldsymbol{x})|\} \tag{8-170}$$

这一最大最小最优化问题可以改写成有带不等式约束的线性规划问题，如下

$$\begin{aligned}&\text{最小化}\quad x_0\\&\text{受限于}\quad |e_i(\boldsymbol{x})|\leqslant x_0\qquad \text{对于}i=1,2,\cdots,m\\&\text{且}\quad x_0\geqslant 0\end{aligned} \tag{8-171}$$

$x_0(x_0^o)$的最优值必须满足式（8-170）

$$x_0^o=\mathscr{E}(\boldsymbol{x}^o)=\min_{x\in\Re^n}\max_{1<i<m}\{|e_i(\boldsymbol{x})|\} \tag{8-172}$$

式（8-171）中的线性规划问题可以写成一种更加易于求解的形式，即

$$\begin{aligned}&\text{最小化}\quad x_0\\&\text{受限于}\quad \begin{cases}f_{i1}(\hat{\boldsymbol{x}})=x_0+e_i(\boldsymbol{x})\geqslant 0\\f_{i2}(\hat{\boldsymbol{x}})=x_0-e_i(\boldsymbol{x})\geqslant 0\end{cases}\\&\text{且}\quad x_0\geqslant 0\end{aligned} \tag{8-173}$$

其中$\hat{\boldsymbol{x}}=[x_0,x_1,x_2,\cdots,x_n]^T$。求解式（8-173）的线性规划问题的一种方法是，首先用公式表示一个基于二次惩罚函数方法[30]的能量函数，也就是，

$$\mathscr{E}(\hat{\boldsymbol{x}})=\alpha_1 x_0+\frac{\alpha_2}{2}\sum_{i=1}^{m}\left(\{[f_{i1}(\hat{\boldsymbol{x}})]_{\min}\}^2+\{[f_{i2}(\hat{\boldsymbol{x}})]_{\min}\}^2\right) \tag{8-174}$$

其中$[\xi]_{\min}=\min[0,\xi]$以及$\alpha_1$，$\alpha_2>0$[8]。使用这一能量函数，可以形成包含两个微分方程的最速下降梯度系统，这两个方程组成神经网络连续时间学习规则。这两个微分方程的基本

[379] 形式

$$\frac{\mathrm{d}x_0}{\mathrm{d}t}=-\mu_0\nabla_{x0}\mathscr{E}(\hat{\boldsymbol{x}}) \tag{8-175}$$

$$\frac{\mathrm{d}x_j}{\mathrm{d}t}=-\mu_j\nabla_{xj}\mathscr{E}(\hat{\boldsymbol{x}}) \tag{8-176}$$

其中$j = 1, 2, \cdots, n$。式（8-175）中的梯度可以计算为

$$\begin{aligned}\nabla_{x0}\mathscr{E}(\hat{\boldsymbol{x}}) &= \frac{\partial \mathscr{E}(\hat{\boldsymbol{x}})}{\partial x_0} \\ &= \alpha_1 + \frac{\alpha_2}{2}\sum_{i=1}^{m}\left(\frac{\partial}{\partial x_0}\{[f_{i1}(\hat{\boldsymbol{x}})]_{\min}\}^2 + \frac{\partial}{\partial x_0}\{[f_{i2}(\hat{\boldsymbol{x}})]_{\min}\}^2\right) \\ &= \alpha_1 + \alpha_2\sum_{i=1}^{m}\{[x_0 + e_i(\boldsymbol{x})]T^1_{\text{binary},i} + [x_0 - e_i(\boldsymbol{x})]T^2_{\text{binary},i}\}\end{aligned} \tag{8-177}$$

其中

$$T^1_{\text{binary},i} \triangleq T^1_{\text{binary},i}[f_{i1}(\hat{\boldsymbol{x}})] = \begin{cases}0 & \text{如果} x_0 + e_i(\boldsymbol{x}) \geqslant 0 \\ 1 & \text{其他}\end{cases} \tag{8-178}$$

以及

$$T^2_{\text{binary},i} \triangleq T^2_{\text{binary},i}[f_{i2}(\hat{\boldsymbol{x}})] = \begin{cases}0 & \text{如果} x_0 - e_i(\boldsymbol{x}) \geqslant 0 \\ 1 & \text{其他}\end{cases} \tag{8-179}$$

式（8-176）中的梯度可以如下计算

$$\begin{aligned}\nabla_{xj}\mathscr{E}(\hat{\boldsymbol{x}}) &= \frac{\partial \mathscr{E}(\hat{\boldsymbol{x}})}{\partial x_j} \\ &= \frac{\alpha_2}{2}\sum_{i=1}^{m}\left(\frac{\partial}{\partial x_j}\{[f_{i1}(\hat{\boldsymbol{x}})]_{\min}\}^2 + \frac{\partial}{\partial x_j}\{[f_{i2}(\hat{\boldsymbol{x}})]_{\min}\}^2\right) \\ &= \alpha_2\sum_{i=1}^{m}(\alpha_{ij}\{[x_0 + e_i(\boldsymbol{x})]T^1_{\text{binary},i} + [x_0 - e_i(\boldsymbol{x})]T^2_{\text{binary},i}\})\end{aligned} \tag{8-180}$$

其中$T^1_{\text{binary},i}$以及$T^2_{\text{binary},i}$分别由式（8-178）和式（8-179）定义。设置这两个梯度值为0，式（8-177）和式(8-180)可以写作

$$\nabla_{x_0}\mathscr{E}(\hat{\boldsymbol{x}}) = \frac{\alpha_1}{\alpha_2} + \sum_{i=1}^{m}\{[x_0 + e_i(\boldsymbol{x})]T^1_{\text{binary},i} + [x_0 - e_i(\boldsymbol{x})]T^2_{\text{binary},i}\} \tag{8-181}$$

$$\nabla_{x_0}\mathscr{E}(\hat{\boldsymbol{x}}) = \sum_{i=1}^{m} a_{ij}\{[x_0 + e_i(\boldsymbol{x})]T^1_{\text{binary},i} + [x_0 - e_i(\boldsymbol{x})]T^2_{\text{binary},i}\} \tag{8-182}$$

使用式（8-181）和式（8-182），方程式（8-175）和式（8-176）可以写作

$$\frac{\mathrm{d}x_0}{\mathrm{d}t} = -\mu_0\left(\frac{\alpha_1}{\alpha_2} + \sum_{i=1}^{m}\{[x_0 + e_i(\boldsymbol{x})]T^1_{\text{binary},i} + [x_0 - e_i(\boldsymbol{x})]T^2_{\text{binary},i}\}\right) \tag{8-183}$$

$$\frac{\mathrm{d}x_j}{\mathrm{d}t} = -\mu_j\sum_{i=1}^{m}(a_{ij}\{[x_0 + e_i(\boldsymbol{x})]T^1_{\text{binary},i} + [x_0 - e_i(\boldsymbol{x})]T^2_{\text{binary},i}\}) \tag{8-184}$$

380

其中$j = 1, 2, \cdots, n$, $\mu_0 > 0$以及$\mu_j > 0$。图8-14描述了使用最小最大二次惩罚函数方法求解线性方程组的神经网络体系结构。

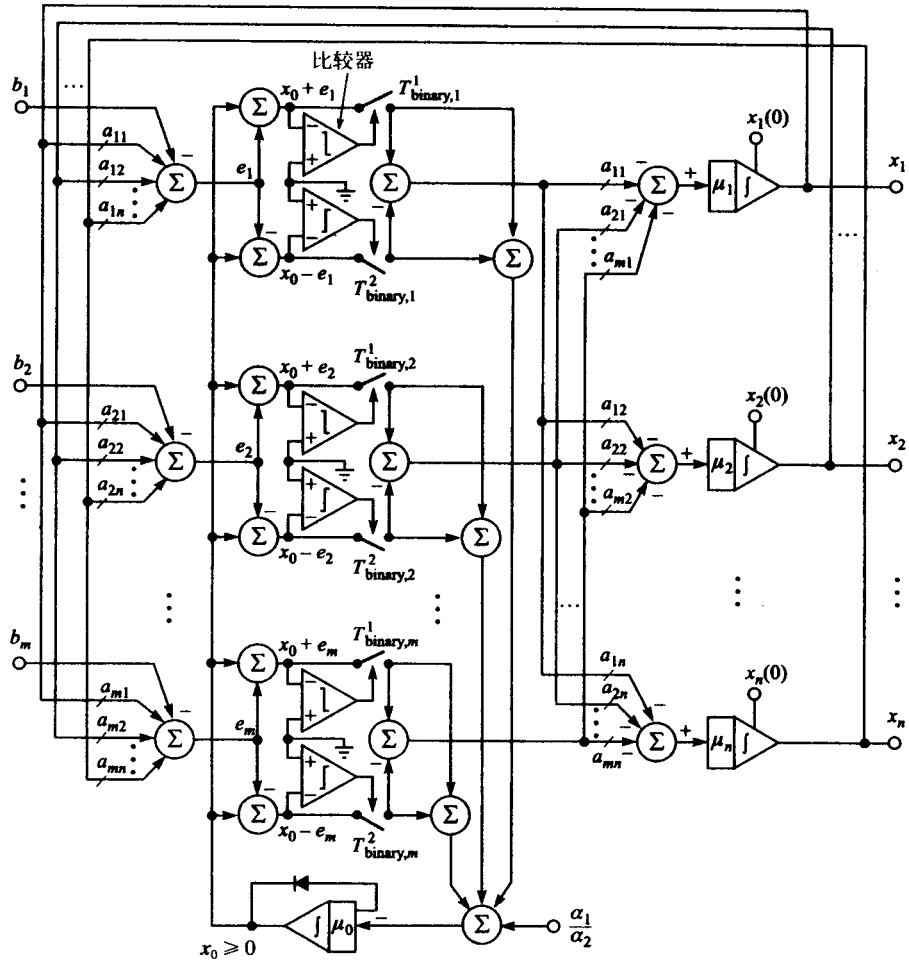



图8-14　使用最小最大二次惩罚函数方法求解线性方程组的神经网络体系结构

## 8.11　求解线性方程的$L_1$范数(最小绝对偏差)神经网络

$L_1$范数有时候称作*最小绝对偏差*范数。对于求解线性（以及非线性）方程组，这一范数
381 为$L_\infty$范数（Chebyshev范数）和$L_2$范数（最小二乘范数）提供了一种非常有用的替代。$L_1$范数和$L_\infty$范数已在信号处理领域有着广泛的应用[10, 12, 17, 21, 23, 24, 63]。$L_1$范数求解线性方程组拥有的许多特征是$L_2$范数最小二乘求解同一系统所不具备的。例如，$L_1$范数求解倾向于鲁棒，也就是这些解对于数据中相对较大的误差敏感性较差。线性方程组超定组的$L_1$范数解一般都存在（但是，并不要求唯一），而对于相同系统的最小二乘（$L_2$范数）解对于满秩系统是唯一的。$L_1$范数问题相当于线性规划问题，反过来，线性规划问题可以使用公式表示成$L_1$范数问题。

$L_1$范数（最小绝对偏差）问题的基本形式可以描述成以下形式。考虑线性代数方程组

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{8-185}$$

($\boldsymbol{A} \in \Re^{m \times n}$, $\boldsymbol{x} \in \Re^{n \times 1}$, $\boldsymbol{b} \in \Re^{m \times 1}$)，包含以下的约束

$$\delta_i = \boldsymbol{c}_i^T \boldsymbol{x} \quad \text{对于 } i = 1, 2, \cdots, p$$

$$\delta_i \leqslant \boldsymbol{c}_i^T \boldsymbol{x} \quad \text{对于 } i = p+1, p+2, \cdots, q \tag{8-186}$$

所要求的解向量$\boldsymbol{x}$要能使如下能量函数最小化

$$\mathscr{E}(\boldsymbol{x}) = \| \boldsymbol{e}(\boldsymbol{x}) \|_1 = \sum_{i=1}^{m} | e_i(\boldsymbol{x}) | \tag{8-187}$$

其中误差向量$\boldsymbol{e}(\boldsymbol{x})$由式$\boldsymbol{e}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$给出，或者利用标量形式

$$e_i(\boldsymbol{x}) = \sum_{j=1}^{n} a_{ij} x_j - b_i \tag{8-188}$$

因此，使用式（8-188），由式（8-187）给出的能量函数可以写作如下形式

$$\mathscr{E}(\boldsymbol{x}) = \sum_{i=1}^{m} \left| \sum_{j=1}^{n} a_{ij} x_j - b_i \right| \tag{8-189}$$

受限于式（8-186）的约束条件。约束值包含在向量$\boldsymbol{\delta} = [\delta_1, \delta_2, \cdots, \delta_q]^T$中，且$\boldsymbol{c}_i = [c_{i1}, c_{i2}, \cdots, c_{in}]^T$是约束向量。正如在这一节介绍中所述，$L_1$范数问题相当于线性规划问题。

现在介绍一种实现求线性方程组$L_1$范数的神经网络体系结构[8]。首先，考虑以下非约束问题。确定解向量$\boldsymbol{x}$，最小化能量函数

$$\mathscr{E}(\boldsymbol{x}) = \| \boldsymbol{e}(\boldsymbol{x}) \|_i = \sum_{i=1}^{m} | e_i(\boldsymbol{x}) \tag{8-190}$$

382

其中，

$$e_i(\boldsymbol{x}) = \sum_{j=1}^{n} a_{ij} x_j - b_i \tag{8-191}$$

现在修改这一非约束问题，使之包含辅助性的不等式及等式约束，形成一个线性规划问题，也就是，

$$\text{最小化} \sum_{i=1}^{m} \sigma_i \tag{8-192}$$

$$\text{受限于} -\sigma_i \leqslant e_i(\boldsymbol{x}) \leqslant \sigma_i \qquad i = 1, 2, \cdots, m \tag{8-193}$$

一种可以求解这一问题的方法是拉格朗日乘子技术（参照A.6.2节）。采用这种方法实际上是对标准拉格朗日函数的增大，使之包括正则化项。因此，增广的拉格朗日（能量）函数可以由下式给出

$$\begin{aligned} \mathscr{L}(\boldsymbol{x}, \lambda, \sigma) = \mathscr{E}(x) = \sum_{i=1}^{m} \Bigg[ & \sigma_i + \frac{\alpha_i}{2} \left( \{ [e_i(\boldsymbol{x}) + \sigma_i]_{\min} \}^2 + \{ [e_i(\boldsymbol{x}) - \sigma_i]_{\min} \}^2 \right) \\ & + \lambda_i \left\{ [e_i(\boldsymbol{x}) + \sigma_i]_{\min} + [e_i(\boldsymbol{x}) - \sigma_i]_{\min} - \frac{\sigma}{2} \lambda_i^2 \right\} \Bigg] \end{aligned} \tag{8-194}$$

其中

$[\xi]_{\min} = \min[0, \xi]$

$[\xi]_{\max} = \max[0, \xi]$

$\alpha_i > 0$ = 罚参数

$\lambda_i$ = 拉格朗日乘子

$\sigma \geqslant 0$=正则化（和稳定性）参数

使用式（8-194）中的能量函数，最速下降梯度系统可以由三个构成神经网络连续时间学习规则的微分方程组成。这三个微分方程分别是

$$\frac{\mathrm{d}x}{\mathrm{d}t} = -\mu_j \sum_{i=1}^{m} \alpha_{ij}(\lambda_i + \alpha_i \gamma_i) \tag{8-195}$$

$$\frac{\mathrm{d}\sigma_i}{\mathrm{d}t} = -\mu_0(1 - |\lambda_i + \alpha_i \gamma_i|) \tag{8-196}$$

$$\frac{\mathrm{d}\lambda_i}{\mathrm{d}t} = \beta_i(\gamma_i - \sigma\lambda_i) \tag{8-197}$$

其中$j = 1, 2, \cdots, n$以及$i = 1, 2, \cdots, m$, 以及

$$\mu_j > 0 \qquad \beta_i > 0 \tag{8-198}$$

$$\gamma_i = \begin{cases} e_i(\boldsymbol{x}) + \sigma_i & \text{对于 } e_i(\boldsymbol{x}) < -\sigma_i \\ 0 & \text{对于 } -\sigma_i \leqslant e_i(\boldsymbol{x}) \leqslant \sigma_i \\ e_i(\boldsymbol{x}) - \sigma_i & \text{对于 } e_i(\boldsymbol{x}) > -\sigma_i \end{cases} \tag{8-199}$$

图8-15描述的神经网络结构实现了基于$L_1$范数（最小绝对偏差范数）求解线性方程组的连续时间（模拟）过程。

383

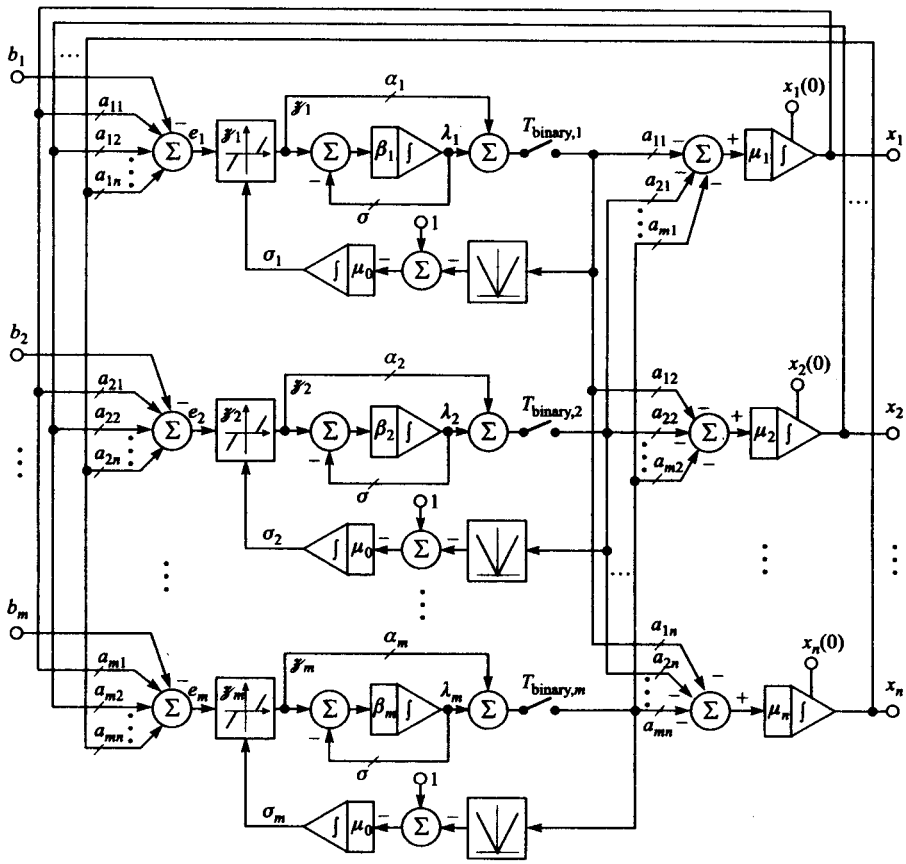



图8-15　使用$L_1$（最小绝对偏差）范数求解线性方程组的人工神经网络体系结构

## 习题

8.1　考虑以下线性方程组的超定组（$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$）

$$\text{(a)}\begin{bmatrix}0&1&0\\2&3&0\\2&0&0\\0&2&1\\1&-1&1\\1&5&6\\5&4&4\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}=\begin{bmatrix}-2\\-8\\1\\1\\-6\\6\\6\end{bmatrix}\qquad\text{(b)}\begin{bmatrix}1&0\\0&1\\1&1\\-1&1\\2&-1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}=\begin{bmatrix}1\\0\\0\\1\\-1\end{bmatrix}$$

请使用在8.4节中提到的标准最小二乘神经计算方法求解以上各$\boldsymbol{x}$。求解终止的准则是当 384
$\|\boldsymbol{x}^N-\boldsymbol{x}^M\|_2<10^{-8}$，其中$\boldsymbol{x}^N$是神经网络解，$\boldsymbol{x}^M$是MATLAB解（使用pinv内建函数）。使用一组合适的随机数值，初始化你的网络。最优化学习率参数以及在每一种情况训练步骤的数目，使得求解在最小数目的迭代后完成。

8.2　考虑以下线性方程欠定组

$$\text{(a)}\begin{bmatrix}22&5&4&0&4&0&4\\1&17&10&3&9&7&3\\10&6&0&11&7&6&15\\6&6&0&19&6&3&2\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\\x_4\\x_5\\x_6\\x_7\end{bmatrix}=\begin{bmatrix}1\\3\\14\\4\end{bmatrix}$$

$$\text{(b)}\begin{bmatrix}6&2&4&-9&-12&2&-12&0&1\\8&-10&1&8&-22&0&-11&-11&7\\9&-7&-6&6&10&-10&-15&-13&-12\\-10&11&-6&-8&-5&-9&1&-3&-5\\2&-1&4&-3&3&-4&-12&10&-3\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\\x_4\\x_5\\x_6\\x_7\\x_8\\x_9\end{bmatrix}=\begin{bmatrix}-12\\-13\\9\\0\\-6\end{bmatrix}$$

使用在8.4节提出的标准最小二乘神经计算方法求解每个$\boldsymbol{x}$。使用与问题8.1相同的终止准则。

8.3　在A.2.9节中关于一个矩阵迹的特征，可以有$\text{trace}(\boldsymbol{AB})=\text{trace}(\boldsymbol{BA})$，其中$\boldsymbol{A}\in\Re^{n\times m}$以及$\boldsymbol{B}\in\Re^{m\times n}$。使用这个特征以及矩阵迹的其他特征，证明以下误差代价函数是相同的。第一个误差代价函数是

$$\mathscr{E}_1(\boldsymbol{C})=\frac{1}{2}\text{trace}(\boldsymbol{E}_1\boldsymbol{E}_1^T)$$

其中误差矩阵$\boldsymbol{E}_1$与方程式（7-13）中提出的矩阵伪逆的计算相关，也就是，

$$E_1 = ACA - A$$

其中$A \in \Re^{m \times n}$，$C \in \Re^{n \times m}$即是矩阵$A$的伪逆。第二个误差代价函数是

$$\mathscr{E}_2(C) = \frac{1}{2}\text{trace}(E_2^T E_2)$$

其中误差矩阵$E_2$即式（8-40）所示，也就是，

$$E_2 = A^T AC - A^T$$

385

这个误差矩阵同样与矩阵伪逆的计算相关。然而，它是从求解线性方程组的学习规则中发展而来的。请证明：$\mathscr{E}_1(C) = \mathscr{E}_2(C)$。

8.4　编写一段计算机程序实现在8.5节中提出的求解方程组的共轭梯度神经计算方法，也就是，带有重启选项的Fletcher-Reeves共轭梯度算法。用你自己的程序，求解以下矩阵方程的解$x$

$$\begin{bmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 11 & 8 \\ 7 & 5 & 8 & 11 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 22 \\ 23 \\ 33 \\ 31 \end{bmatrix}$$

使用一组适当的随机数初始化网络。那么求解收敛所必需的最小训练步数目是多少？为你的程序确定一个合理的终止准则。尝试使用你在问题8.1中编制的实现最小二乘神经计算方法的计算机程序，求解以上矩阵方程。解释结果。

8.5　考虑以下矩阵方程

$$\begin{bmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 11 & 8 \\ 7 & 5 & 8 & 11 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 32.1 \\ 22.9 \\ 32.9 \\ 31.1 \end{bmatrix}$$

这与问题8.4考虑的系统相同，除了向量$b$中的元素已经被一个大小为$\pm 0.1$的量所扰动。首先检查矩阵$A$的条件数。基于这一条件数，你能看到对比问题8.4中求得的解向量$x$的元素的强烈变化吗？使用你的共轭梯度程序，确定矩阵方程的解。对比问题8.4中你的解，是否存在强烈变化？

8.6　一个复线性矩阵方程如下

$$(A + jC)x = b + jd$$

其中向量$x$可以写作$x = x_R + jx_I (j = \sqrt{-1})$。将这一复向量代入以上复线性矩阵方程，推导出两个新的方程

$$Ax_R - Cx_I = b$$

$$j(Cx_R - Ax_I) = jd$$

可以将两者写成一个实的向量矩阵方程如下

$$\begin{bmatrix} A & -C \\ C & A \end{bmatrix} \begin{bmatrix} x_R \\ x_I \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}$$

386

使用这一转化以及在8.4节提出的标准最小二乘神经计算方法，求解下式

$$\begin{bmatrix} 1+j2 & 5-j7 & 3-j3 \\ 3-j5 & 1+j5 & j2 \\ 3 & 3-j3 & 3+j3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}=\begin{bmatrix} 1+j2 \\ 1-j2 \\ 1+j2 \end{bmatrix}$$

使用标准方法，通过在MATLAB中计算求解，从而对照检查你的结果。

8.7　编写一段计算机程序实现在8.6节提出的求解噪声干扰的线性方程组的广义鲁棒方法。假定非线性加权函数是logistic函数，也就是，$f_L(t)=\beta^2\ln[\cosh(t/\beta)]$。一个标准线性矩阵方程如下所示

$$\begin{bmatrix} -1 & -4 & 5 & 6 & 10 \\ -1 & -1 & -9 & -2 & -16 \\ 5 & 15 & 0 & -21 & -1 \\ -6 & -6 & -6 & 1 & -7 \\ -1 & -13 & 5 & 16 & -10 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}=\begin{bmatrix} -12 \\ 3 \\ -4 \\ 1 \\ -4 \end{bmatrix}$$ 标准系统

使用MATLAB求解这一标准系统，即$x^M$。假定数据矩阵$A$已经被脉冲噪声干扰，同时给出的结果系统如下

$$\begin{bmatrix} -1 & -4 & 5.236\ 1 & 6 & 10 \\ -1 & -0.917\ 8 & -9 & -1.977\ 0 & -16 \\ 5 & 15 & 0.178\ 1 & -21 & -1 \\ -6 & -6 & -6 & 1 & -7 \\ -1 & -12.824\ 1 & 5 & 16 & -10 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}=\begin{bmatrix} -12 \\ 3 \\ -4 \\ 1 \\ -4 \end{bmatrix}$$ 受扰系统

矩阵$A$的部分元素受扰破坏起因于一些随机选择的15%的元素被一个来自方差为（0.15）*（0.005）的均匀分布的随机数所干扰。再一次使用MATLAB，确定方程组的解。使用你的计算机程序寻求鲁棒解，并使之与MATLAB的解进行对比。分别使用式（8-75）和式（8-76），计算MATLAB求解和鲁棒神经计算求解的标准误差估计（SEE）。按照例8.3的图8-10，以相同的方式绘制你的结果图。

提示：尝试不同的$\beta$参数值以及学习率参数值。同样，使用来自方差为25的均值为0的高斯分布的随机数作为鲁棒神经网络的初始权值。

8.8　Kahan矩阵是上三角矩阵（参照A.2.19节）。它可以生成作为带有病态确定数值秩的病态矩阵。以下介绍使用MATLAB函数产生Kahan矩阵（注：角度$\alpha$以度计量）

387

```
function A=kahan (n,alpha)
alpha=alpha*pi/180;
TMP1=zeros (n,n);
TMP2=eye (n,n);
TMP1 (1,1)=1;
for k=2:n
TMP1 (k,k)=(sin(alpha))^(k-1);
end
for k=1:n
  for j=1:n
     if j>k
        TMP2 (k, j)=cos(alpha);
     end
  end
end
A=TMP1*TMP2;
```

使用这一函数，生成一个10×10的角度$\alpha = 1.0°$的Kahan矩阵。这将作为数据矩阵$\boldsymbol{A}$。“精确解”向量$\boldsymbol{x}_e$的元素由下式给出

$$x_{ei} = e^{2i} \qquad \text{其中} i = 1, 2, \cdots, 10$$

从你生成的Kahan矩阵$\boldsymbol{A}$以及精确的解向量$\boldsymbol{x}_e$可以产生观察向量$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}_e$。那么数据矩阵$\boldsymbol{A}$的条件数是多少？使用神经计算方法求解$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$，导出最优解，也就是，对于相对误差的最优解（参照例8.4）。

8.9　使用MATLAB的hilb函数，产生一个维数为15的希尔伯特（Hilbert）矩阵。精确解向量$\boldsymbol{x}_e$的元素由下式给出

$$x_{ei} = e^{i/2} \qquad \text{其中} i = 1, 2, \cdots, 15$$

从希尔伯特矩阵$\boldsymbol{A}$以及精确解向量$\boldsymbol{x}_e$，可以产生观察向量$\boldsymbol{b}$如下$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}_e$。那么数据矩阵$\boldsymbol{A}$的条件数是多少？使用8.7节提出的截断奇异值分解（TSVD）以及Tikhonov正则化方法，寻求$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$的考虑相对误差的最优解（参照例8.4）。确定最佳的正则化参数。

8.10　分别使用（a）高斯-赛德尔，(b)雅可比，(c) 理查森以及(d) 逐次超松弛（SOR）基于矩阵分裂的迭代算法，求下式的解

$$\begin{bmatrix} 4 & 3 & 2 & 5 & 2 \\ 3 & 7 & 4 & 4 & 3 \\ 2 & 4 & 11 & 7 & 4 \\ 5 & 4 & 7 & 16 & 7 \\ 2 & 3 & 4 & 7 & 11 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} -34 \\ -2 \\ -3 \\ -123 \\ -113 \end{bmatrix}$$

388

对于每一种迭代方法，基于相对误差标准$\|\boldsymbol{x}^N - \boldsymbol{x}^M\|_2 / \|\boldsymbol{x}^M\|_2 < 10^{-8}$最小化实现收敛的迭代数目，其中$\boldsymbol{x}^N$是任意的一个从(a)～(d)的迭代解，$\boldsymbol{x}^M$为MATLAB解。对于每一种方法使用相同的初始随机权值。

8.11　确定以下直线的参数（$a, b$）（斜率以及纵轴交点）

$$y(x) = ax + b$$

直线满足数据点$\{x_i, y_i\} = \{(0, 1), (2, 2), (3, 3), (4, 7), (6, 9)\}$。这个问题可以用公式表示成超定线性方程组，如下

$$\begin{bmatrix} 0 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \\ 6 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 7 \\ 9 \end{bmatrix}$$

应用最小二乘神经计算方法求解这一方程组。在同一张图上，使用你求得的点（$a, b$)的值，画出数据点以及最小二乘直线。

8.12　对于问题8.11应用$L_1$范数（参照8.11节）以及$L_\infty$范数（参照8.10节）标准。

8.13　求解以下多项式的参数（$a, b, c, d$)

$$y(x) = ax^3 + bx^2 + cx + d$$

以上多项式满足数据点$\{x_i, y_i\} = \{(0, 1), (1, 3), (2, 4), (4, 5), (5, 8), (7, 9), (10, 11)\}$。应用最小二乘神经计算方法，在同一张图上使用你求到的值（$a, b, c, d$）画出最终的最

小二乘多项式以及满足观察的数据点。

提示：使用MATLAB的polyval函数计算你所确定的多项式$y = ax^3 + bx^2 + cx + d$的系数 $(a, b, c, d)$。

8.14　要求使用以下6次多项式对于$\{x_i\} = \{0, 0.1, 0.2, \cdots, 10\}$逼近函数$f(x) = e^{-0.1x}\cos x$

$$y(x) = ax^6 + bx^5 + cx^4 + dx^3 + ex^2 + fx + g$$

确定多项式$y(x)$的系数，也就是 $(a, b, c, d, e, f, g)$，使得多项式在考虑最小二乘时尽可能最好的拟合函数$f(x)$。实验使用不同次多项式确定什么次可以得到最好的拟合。 389

提示：使用在8.5节提出的共轭梯度学习规则。以下所示的MATLAB函数可以用于从所有的数据点中生成数据矩阵$A$以及观察向量$b$。

```
function [A,b]=points (x,y,n)
% [A,b]=points(x,y,n)
% Sets up: Ax=b
% n: degree of the polynomial
% x: vector of "x" points
% y: vector of "y" points
    k=length(x);
    for i=1:k
        for j=1:n+1
            A(i,j)=x(i)^(j-1);
        end
    end
    A=rot90(A',-1);
    b=y;
```

8.15　假设标准线性矩阵方程$Ax = b$，其中数据矩阵可以由MATLAB产生如下

```
A=round(10*randn(10,10));
```

产生如下形式观察向量

```
b=round(5*randn(10,1))
```

受干扰的数据矩阵$A_c$可以使用以下介绍的MATLAB函数uniferr产生。这函数可以用来在MATLAB中随机使用脉冲噪声来干扰数据矩阵$A$中的一些元素。函数的两个自变量元素必须是已知的，它们分别是标准数据矩阵$A$以及$A$中受干扰的元素的（平均）百分率。在函数自变量中的第三个元素是可以选择的。对于目前的问题使用默认值。受干扰的元素百分比使用15%。

使用MATLAB的pinv函数（求取矩阵伪逆）求解标准系统以及受干扰系统的最小二乘解。使用8.6节提出的广义鲁棒方法求解方程组，并确定受干扰系统的鲁棒解。假定非线性加权函数是logistic函数$f_L(t) = \beta^2\ln[\cosh(t/\beta)]$。用方差为25的零均值高斯分布的随机数作为鲁棒神经网络一个初始权值。尝试不同的$\beta$值以及学习率参数。分别使用式（8-75）以及式（8-76），为MATLAB的解和鲁棒神经计算的解计算标准误差估计。按照例8.3的图8-10相同的方式画出你的计算结果。 390

```
function [AC,N]=uniferr(A,per,s)
% [AC,N]=uniferr(A,per,s)
% A:      input matrix
% AC:     corrupted "output matrix"
% N:      noise matrix, random elements from
%         uniform distribution on the interval
%         (0.0,1.0), that is, if s=1
```

```
% per:   percentage of elements to be corrupted
% s:     interval scale factor, i.e.,
%        (0.0,s), the default value is
%        s=0.2449 resulting in a
%        variance of (per/100)*0.005
  if nargin>2
     scale=s;
  else
     scale=0.2449;
     end
  [nr,nc]=size(A);
  N=zeros(size(A));
  for i=1:nr
  for j=1:nc
  if (rand <= per/100)
  N(i,j)=scale*rand;
     end
   end
  end
  AC=A+N;
```

## 参考文献


1. E. Anderson, Z. Bai, C. H. Bischof, J. W. Demmel, J. J. Dongarra, J. J. Du Croz, A. Greenbaum, S. J. Hammarling, A. McKenney, S. Ostrouchov, and D. C. Sorenson, *LAPACK Users' Guide, Release 2.0*, 2nd ed., Philadelpha: Society for Industrial and Applied Mathematics, 1995.
2. G.-J. Li and B. W. Wah, "The Design of Optimal Systolic Arrays," *IEEE Transactions on Computers*, vol. C-34, 1985, pp. 66–77.
3. S. Y. Kung, *VLSI Array Processors*, Englewood Cliffs, NJ: Prentice-Hall, 1988.
4. C. Lehmann, M. Viredaz, and F. Blayo, "A Generic Systolic Array Building Block for Neural Networks with On-Chip Learning," *IEEE Transactions on Neural Networks*, vol. 4, 1993, pp. 400–7.
5. V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms*, Redwood City, CA: Benjamin/Cummings, 1994.
6. A. Cichocki and R. Unbehauen, "Neural Networks for Solving Systems of Linear Equations and Related Problems," *IEEE Transactions on Circuits and Systems*, vol. CAS-39, 1992, pp. 124–38.
7. A. Cichocki and R. Unbehauen, "Neural Networks for Solving Systems of Linear Equations—Part II: Minimax and Least Absolute Value Problems," *IEEE Transactions on Circuits and Systems*, vol. CAS-39, 1992, pp. 619–33.

391

8. A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, New York: Wiley, 1993.
9. S. A. Ruzinsky and E. T. Olsen, "$L_1$ and $L_\infty$ Minimization via a Variant of Karmarkar's Algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, 1989, pp. 245–53.
10. J. W. Bandler, W. Kellerman, and K. Madsen, "A Nonlinear $L_1$ Optimization Algorithm for Design, Modeling, and Diagnosis of Networks," *IEEE Transactions on Circuits and Systems*, vol. CAS-34, 1987, pp. 174–81.
11. I. Barrodale and C. A. Zala, "$L_1$ and $L_\infty$ Curve Fitting and Linear Programming Algorithms and Applications," in *Numerical Algorithms*, eds. J. L. Mohamed and J. E. Walsh, Oxford: Oxford University Press, 1986, pp. 220–38.
12. N. N. Abdelmalek, "Chebychev and $L_1$ Solution of Overdetermined Systems of Linear Equations with Bounded Variables," *Numerical Functional Analysis and Optimization*, vol. 8, 1986, pp. 399–418.

13. D. P. O'Leary, "Robust Regression Computation Using Iteratively Reweighted Least-Squares," *SIAM Journal on Matrix Analysis and Applications*, vol. 11, 1990, pp. 466–80.
14. R. H. Bartels, A. R. Conn, and Y. Li, "Primal Methods Are Better than Dual Methods for Solving Overdetermined Systems in the $\ell_\infty$ Sense?" *SIAM Journal on Numerical Analysis*, vol. 26, 1989, pp. 693–726.
15. A. Dax, "The Convergence of Linear Stationary Iterative Processes for Solving Singular Unstructured Systems of Linear Equations," *SIAM Review*, vol. 32, 1990, pp. 611–25.
16. A. Dax, "A Row Relaxation Method for Large $L_1$ Problems," *Linear Algebra and Its Applications*, vols. 154–156, 1991, pp. 793–818.
17. J. Schroeder, R. Yarlagadda, and J. Hershey, "$L_p$ Normed Minimization with Applications to Linear Predictive Modeling for Sinusoidal Frequency Estimation," *Signal Processing*, vol. 24, 1991, pp. 193–216.
18. P. J. Huber, *Robust Statistical Procedures*, 2nd ed., Philadelphia: Society for Industrial and Applied Mathematics, 1996.
19. P. J. Huber, *Robust Statistics*, New York: Wiley, 1981.
20. F. R. Hampel, E. N. Ronchetti, P. Rousser, and W. A. Stachel, *Robust Statistics—The Approach Based on Influence Functions*, Philadelphia: Wiley, 1987.
21. C. A. Zala, I. Barrodale, and J. S. Kennedy, "High-Resolution Signal and Noise Field Estimation Using the $L_1$ Norm (Least Absolute Values)," *IEEE Transactions on Oceanic Engineering*, vol. OE-12, 1987, pp. 253–64.
22. P. Bloomfield and W. L. Steiger, *Least Absolute Deviations: Theory, Applications, and Algorithms*, Boston: Birkhäuser, 1983.
23. R. K. Ward, "An On-line Adaptation for Discrete $\ell_1$ Linear Estimation," *IEEE Transactions on Automatic Control*, vol. AC-29, 1984, pp. 67–71.
24. S. Levy, C. Walker, T. J. Ulrych, and P. K. Fullagar, "A Linear Programming Approach to the Estimation of the Power Spectra of Harmonic Processes," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-30, 1982, pp. 675–9.
25. G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Baltimore, MD: Johns Hopkins University Press, 1996.
26. Y. Zhang, "A Primal-Dual Interior Point Approach for Computing the $L_1$ and $L_\infty$ Solutions of Overdetermined Linear Systems," *Journal of Optimization Theory and Applications*, vol. 77, 1993, pp. 323–41.
27. A. M. Mood, F. A. Graybill, and D. C. Boes, *Introduction to the Theory of Statistics*, 3rd ed., New York: McGraw-Hill, 1974.
28. M. Abramowitz and C. A. Stegun (eds.), *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, 9th printing, New York: Dover, 1972.

392

29. N. Higham, *Accuracy and Stability of Numerical Algorithms*, Philadelphia: Society for Industrial and Applied Mathematics, 1996.
30. S. G. Nash and A. Sofer, *Linear and Nonlinear Programming*, New York: McGraw-Hill, 1996.
31. D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd ed., Reading, MA: Addison-Wesley, 1984.
32. M. Avriel, *Nonlinear Programming: Analysis and Methods*, Englewood Cliffs, NJ: Prentice-Hall, 1976.
33. F. M. Ham and T. M. McDowall, "Robust Learning in a Partial Least-Squares Neural Network," *Journal of Nonlinear Analysis, Theory, Methods & Applications*, Proceedings of the *Second World Congress of Nonlinear Analysts 1996*, July 10–17, Athens, Greece, vol. 30, no. 5, 1997, pp. 2903–14.
34. T. M. McDowall and F. M. Ham, "Robust Partial Least-Squares: A Modular

Neural Network Approach," *Applications and Science of Artificial Neural Networks III*, ed. S. Rogers, Proceedings of SPIE, vol. 3077, 1997, pp. 344–55.
35. T. M. McDowall and F. M. Ham, "Robust Learning in a Generalized Partial Least Squares Regression Modular Neural Network," *Journal of Neural, Parallel and Scientific Computations*, vol. 6, 1998, pp. 391–415.
36. T. M. McDowall, "A Robust Partial Least Squares Neural Network," Ph.D. dissertation, Melbourne: Florida Institute of Technology, May 1997.
37. F. M. Ham and T. M. McDowall, "Inverse Partial Least Squares: A Robust Neural Network Approach," *Applications and Science of Computational Intelligence*, eds. S. K. Rogers, D. B. Fogel, J. C. Bezdek, and B. Bosacchi, Proceedings of SPIE, vol. 3390, 1998, pp. 36–47.
38. J. M. Varah, "A Practical Examination of Some Numerical Methods for Linear Discrete Ill-Posed Problems," *SIAM Review*, vol. 21, 1979, pp. 100–11.
39. P. C. Hansen, "Truncated Singular Value Decomposition Solutions to Discrete Ill-Posed Problems with Ill-Determined Numerical Rank," *SIAM Journal of Scientific Statistical Computing*, vol. 11, 1990, pp. 503–18.
40. P. C. Hansen, "The Truncated SVD as a Method for Regularization," *BIT*, vol. 27, 1987, pp. 534–53.
41. P. C. Hansen, T. Sekii, and H. Shibahashi, "The Modified Truncated SVD Method for Regularization in General Form," *SIAM Journal of Scientific Statistical Computing*, vol. 13, 1992, pp. 1142–50.
42. A. N. Tikhonov, "Solution of Incorrectly Formulated Problems and the Regularization Method," *Dokl. Akad. Nauk. SSSR*, vol. 151, 1963, pp. 501–4 (in Russian). [*Soviet Math. Dokl.*, vol. 4, 1963, pp. 1035–8 (in English).]
43. A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*, New York: Wiley, 1977.
44. A. N. Tikhonov and A. V. Goncharsky, *Ill-Posed Problems in the Natural Sciences*, Moscow: MIR Publishers, 1987.
45. J. M. Varah, "Pitfalls in the Numerical Solution of Linear Ill-Posed Problems," *SIAM Journal of Scientific Statistical Computing*, vol. 4, 1983, pp. 164–76.
46. P. C. Hansen, "Analysis of Discrete Ill-Posed Problems by Means of the *L*-Curve," *SIAM Review*, vol. 34, 1992, pp. 561–80.
47. P. C. Hansen, "Test Matrices for Regularization Methods," *SIAM Journal of Scientific Computing*, vol. 16, 1995, pp. 506–12.
48. P. C. Hansen and D. O'Leary, "The Use of the *L*-Curve in the Regularization of Discrete Ill-Posed Problems," *SIAM Journal of Scientific Computing*, vol. 14, 1993, pp. 1487–1503.

393

49. P. C. Hansen, "Regularization Tools: A MATLAB Package for Analysis and Solution of Discrete Ill-Posed Problems," *Numerical Algorithms*, vol. 6, 1994, pp. 1–35.
50. P. C. Hansen, "Regularization Tools: A MATLAB Package for Analysis and Solution of Discrete Ill-Posed Problems," version 2.0 for MATLAB 4.0, Technical Report UNI•C-92-03, Danish Computing Center for Research and Education, Technical University of Denmark, June 1992 (revised June 1993). [Available in PostScript via Netlib (netlib@research.att.com) from the director NUMERALGO.)
51. M. Hanke and P. C. Hansen, "Regularization Methods for Large-Scale Problems," *Surveys on Mathematics for Industry*, vol. 3, 1993, pp. 253–315.
52. M. Hanke, "Limitations of the *L*-Curve Method in Ill-Posed Problems," *BIT*, vol. 36, 1996, pp. 287–301.
53. L. Wu, "Regularization Methods and Algorithms for Least Squares and Kronecker Product Least Squares Problems," Ph.D. dissertation, Melbourne:

Florida Institute of Technology, May 1997.
54. G. H. Golub, M. T. Heath, and G. Wahba, "Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter," *Technometrics*, vol. 21, 1979, pp. 215–23.
55. V. A. Morozov, *Methods for Solving Incorrectly Posed Problems*, New York: Springer-Verlag, 1984.
56. R. S. Varga, *Matrix Iterative Analysis*, Englewood Cliffs, NJ: Prentice-Hall, 1962.
57. D. M. Young, *Iterative Solution of Large Linear Systems*, New York: Academic, 1971.
58. K. E. Atkinson, *An Introduction to Numerical Analysis*, 2nd ed., New York: Wiley, 1989.
59. S. Van Huffel and J. Vandewalle, *The Total Least Squares Problem: Computational Aspects and Analysis*, Philadelphia: Society for Industrial and Applied Mathematics (*Frontiers in Applied Mathematics*, vol. 9), 1991.
60. G. H. Golub and C. F. Van Loan, "An Analysis of the Total Least Squares Problem," *SIAM Journal of Numerical Analysis*, vol. 17, 1980, pp. 883–93.
61. Y. Nievergelt, "Total Least Squares: State-of-the-Art Regression in Numerical Analysis," *SIAM Review*, vol. 36, 1994, pp. 258–64.
62. B. De Moor and J. Vandewalle, "A Unifying Theorem for Linear and Total Linear Least Squares," *IEEE Transactions on Automatic Control*, vol. 35, 1990, pp. 563–6.
63. S. A. Ruzinsky and E. T. Olsen, "$L_1$ and $L_\infty$ Minimization via a Variant of Karmarkar's Algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-37, 1989, pp. 245–53.

394

# 第9章 使用神经网络的统计方法

## 9.1 概述

本章的主要目的是提出两种在解决科技工程问题中广泛应用且非常重要的统计分析以及建模方法，同时讲述如何应用神经计算算法来实现这些方法。特别地，主成分分析（PCA）[1-5] 首先提出。PCA已经在许多工程以及科学应用中得到了广泛的应用，它是主成分回归方法的基础（PCR）[4, 6-8]。PCR是一种基于因子分析的统计建模方法。考虑一系列经验数据（即，训练数据），PCR是一种统计校正模型，可以基于保留最佳数量的PCR因子来获得模型。假设对于校正模型有一个未知的输入，那么，这一统计校正模型就可以用来预测（或估计）输出。

第二种方法称作部分最小二乘回归（PLSR）方法[7-14]。这是另一种基于因子分析的建立统计模型方法。PCR和PLSR之间的区别主要体现在产生统计校正模型的经验数据的表达方式上。对于PCR方法，在产生校正模型的过程中，唯一使用的是测量数据（即，独立变量块）；而对于PLSR方法，测量数据和目标数据（或者相关变量块）都是必需的。一般情况下，PLSR方法比PCR方法会产生更好的估计预测，这是因为在产生统计校正模型时，前者使用了更多的信息。

395 PCR方法和PLSR方法是组成化学统计学[8,14]领域的诸多方法中的两种。化学统计学是应用于化学领域的，基于数学逻辑的统计和数学方法应用的学科。最近，化学统计学中的一些方法（即，PCR以及PLSR方法）才被应用于工程学科以及其他科技领域[9]。在PCR和PLSR方法中，在产生校正模型的过程中，选择必要的训练数据主要特征（因子分析）的能力可以产生更好的预测性能，例如，比经典最小二乘（CLS）方法[7]。也就是说，当合理地进行因子分析时，对于模型的产生只保留数据的恰当特征。例如，保留下的数据若与噪声相关，将会降低预测的性能，则放弃它们[9]。PCA，PCR和PLSR都可在各种神经网络结构中实现。因此，在这一章中，我们将介绍一些PCA，PCR和PLSR的神经网络方法。

## 9.2 主成分分析

主成分分析在许多工程以及科技领域中有着广泛应用，如数字图像处理[15]中著名的Hotelling转换[5]和通信理论中的Karhunen-Loève变换[16]。PCA应用包括数据压缩编码（译码），模式识别，图像处理，自适应波束形成，降阶控制器设计，以及高分辨率谱分析（面向频率估计）等等，这里只提及了其中的一部分。正如我们将要看到的，PCA同特征值分解（EVD）是直接相关的（参照7.6节）。

一般情况下，PCA是一种统计方法，它可以用于确定最优线性变换矩阵$\boldsymbol{W}\in\Re^{m\times n}$ $(m<n)$，如考虑输入向量$\boldsymbol{x}\in\Re^{n\times 1}$，一般认为$\boldsymbol{x}$来源于零均值宽平稳随机过程，$\boldsymbol{x}$中的数据可以根据下式进行压缩

$$\boldsymbol{y}=\boldsymbol{W}\boldsymbol{x} \tag{9-1}$$

其中，$\boldsymbol{y}\in\Re^{m\times 1}$。因此，PCA方法通过变换矩阵$\boldsymbol{W}$，将原始的$n$维向量空间的输入数据投影到$m$维输出空间，其中典型的$m\ll n$。维数减缩因此可以通过PCA执行，其中$\boldsymbol{y}$包含（保持）了大

部分驻留在输入向量$\boldsymbol{x}$中的必要信息。也就是说，PCA可以将大量的相关输入数据转化成一系列的统计去相关分量（或者特征）。分量通常按照方差递减来排序。

举例如下，令$\boldsymbol{x}$是一个零均值随机输入向量，即$\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T$，以及协方差矩阵（或者相关矩阵，因为$\boldsymbol{x}$被看作零均值）、$\boldsymbol{C}_x = \mathrm{E}[\boldsymbol{x}\boldsymbol{x}^T]$，其中$\boldsymbol{C}_x \in \Re^{n\times n}$且$\mathrm{E}[\cdot]$是期望算子（参照A.7.4节）。同样，向量$[\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m]$是一个标准正交集，也就是，单位长度的正交，即$\boldsymbol{w}_i^T\boldsymbol{w}_j = \delta_{ij}$（$\forall i$以及$j$），其中$\delta_{ij}$是克罗内克Δ，并且每个向量的单位长度可以通过$L_2$范数表示为$\|\boldsymbol{w}_i\|_2^2 = \boldsymbol{w}_i^T\boldsymbol{w}_i = 1$。向量$[\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m]$称作协方差矩阵$\boldsymbol{C}_x$的前$m$个特征向量，这样$\boldsymbol{w}_1 = [w_{11}, w_{12}, \cdots, w_{1n}]^T$对应于矩阵$\boldsymbol{C}_x$最大的特征值（$\lambda_1$），$\boldsymbol{w}_2 = [w_{21}, w_{22}, \cdots, w_{2n}]^T$对应于次大的特征值（$\lambda_2$），以此类推。因此，对于标准的特征值问题，矩阵方程可以写作 396

$$\boldsymbol{C}_x\boldsymbol{w}_j = \lambda_j\boldsymbol{w}_j \qquad \text{对于} j = 1, 2, \cdots, n \tag{9-2}$$

并且其满足$\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n \geqslant 0$，同时$\|\boldsymbol{w}i\|_2^2 = 1$。矩阵$\boldsymbol{C}_x$的前$m$个特征向量$[\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m]$称作主特征向量。这就是$n$维向量空间的方向向量，对于它们，输入数据具有最大的方差（或者说最大信息量）。因此，对于一个已知的输入向量$\boldsymbol{x}$，向量$\boldsymbol{y}$中$m$个主成分可以通过式（9-1）的变换来定义，其中变换矩阵$\boldsymbol{W}$如下

$$\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m]^T \tag{9-3}$$

换句话说，输入数据空间的$m$维主成分子空间可以定义为由输入协方差矩阵$\boldsymbol{C}_x$的$m$维主特征向量形成的子空间。

如果用式（9-2）表示每一个特征值与特征向量，即，

$$\boldsymbol{C}_x[\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_n] = [\lambda_1\boldsymbol{w}_1, \lambda_2\boldsymbol{w}_2, \cdots, \lambda_n\boldsymbol{w}_n] \tag{9-4}$$

$$\boldsymbol{C}_x\boldsymbol{W}^T = \boldsymbol{W}^T\mathrm{diag}[\lambda_1, \lambda_2, \cdots, \lambda_n] = \boldsymbol{W}^T\boldsymbol{\Lambda}$$

因此，

$$\boldsymbol{C}_x\boldsymbol{W}^T = \boldsymbol{W}^T\boldsymbol{\Lambda} \tag{9-5}$$

利用$\boldsymbol{W}$（其中$\boldsymbol{W}\boldsymbol{W}^T = \boldsymbol{I}$，$m \leqslant n$，因为$\boldsymbol{W}$的行向量是正交的）左乘式（9-5）的两边，可以得到

$$\boldsymbol{\Lambda} = \boldsymbol{W}\boldsymbol{C}_x\boldsymbol{W}^T \tag{9-6}$$

同样，如果利用$\boldsymbol{W}$（其中$\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}$，$m = n$）右乘式（9-5），又可以得到

$$\boldsymbol{C}_x = \boldsymbol{W}^T\boldsymbol{\Lambda}\boldsymbol{W} = \sum_{i=1}^{n}\lambda_i\boldsymbol{w}_i\boldsymbol{w}_i^T \tag{9-7}$$

也就是矩阵$\boldsymbol{C}_x$（参照7.6节）的特征值分解（EVD），或者谱因子分解。因此，在式(9-6)中$\boldsymbol{\Lambda} \in \Re^{m\times m}$代表式（9-1）中输出向量$\boldsymbol{y}$的协方差矩阵。因为对角矩阵$\boldsymbol{\Lambda}$的对角元素是非负的，输出向量$\boldsymbol{y}$的元素是不相关的，同时方差等于协方差矩阵$\boldsymbol{C}_x$的特征值。

通过再次考虑式（9-1）中的线性变换，可以很好地发现以上性质。我们假定关注的是第$j$个主成分，$y_j$且所有感兴趣的主成分的次序是：第一个主成分关系到输入数据的最大方差，第二个主成分关系到输入数据的次大方差，等等（如同以上陈述）。因此，令$y_j$作为数据输入向量$\boldsymbol{x}$分量的线性集合，也就是，

$$y_j = w_{1j}x_1 + w_{2j}x_2 + \cdots + w_{nj}x_n = \boldsymbol{w}_j^T\boldsymbol{x} \tag{9-8}$$ 397

其中$\boldsymbol{w}_j \in \Re^{n\times 1}$。因为$\boldsymbol{x}$认为是一个零均值的随机向量，所以$y_j$是一个零均值的随机变量，其相关的方差如下所示

$$\sigma_{yj}^2 = \mathrm{E}[y_j^2] = \mathrm{E}[\boldsymbol{w}_j^T\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{w}_j] = \boldsymbol{w}_j^T\mathrm{E}[\boldsymbol{x}\boldsymbol{x}^T]\boldsymbol{w}_j = \boldsymbol{w}_j^T\boldsymbol{C}_x\boldsymbol{w}_j = \sum_{i=1}^{n}\sum_{h=1}^{n}w_{ij}w_{hj}c_{xxih} \tag{9-9}$$

我们要求$\boldsymbol{w}_j^T\boldsymbol{w}_j=1$。因此，我们的目的是在满足$\boldsymbol{w}_j^T\boldsymbol{w}_j=1$的约束下，如同式（9-9）所提出的，最大化与$y_j$相关的方差。这一问题可以通过拉格朗日乘子法（参照A.6.2节）进行计算，也就是，

$$\mathscr{L}(\boldsymbol{w}_j)=\sigma_{yj}^2-\lambda_j(\boldsymbol{w}_j^T\boldsymbol{w}_j-1) \tag{9-10}$$

极值可以通过求解式（9-10）相对于$\boldsymbol{w}_j$的偏导数，令其结果等于零来得到（参照A.3.4.1节）。也就是

$$\begin{aligned}\frac{\partial\mathscr{L}(\boldsymbol{w}_j)}{\partial\boldsymbol{w}_j}&=\frac{\partial}{\partial\boldsymbol{w}_j}[\sigma_{yj}^2-\lambda_j(\boldsymbol{w}_j^T\boldsymbol{w}_j-1)]\\&=\frac{\partial}{\partial\boldsymbol{w}_j}[\boldsymbol{w}_j^T\boldsymbol{C}_x\boldsymbol{w}_j-\lambda_j(\boldsymbol{w}_j^T\boldsymbol{w}_j-1)]\\&=2\boldsymbol{C}_x\boldsymbol{w}_j-2\lambda_j\boldsymbol{w}_j=\boldsymbol{0}\end{aligned} \tag{9-11}$$

或者

$$(\boldsymbol{C}_x-\lambda_j\boldsymbol{I})\boldsymbol{w}_j=\boldsymbol{0} \tag{9-12}$$

这就是式（9-2）中提出的初始特征值问题。进一步地，在当且仅当

$$|\boldsymbol{C}_x-\lambda_j\boldsymbol{I}|=0 \tag{9-13}$$

时，式（9-12）拥有一个非平凡的解，其中式（9-13）中，$\lambda_j$（一个方差）$(j=1,2,\cdots,n)$是矩阵$\boldsymbol{C}_x$的奇异值，且式（9-13）中的$\boldsymbol{w}_j(j=1,2,\cdots,n)$是与之相对应的（主）特征向量。此外，如果用$\boldsymbol{w}_j^T$同时左乘以式（9-12）两边，可以得到

$$\boldsymbol{w}_j^T(\boldsymbol{C}_x-\lambda_j\boldsymbol{I})\boldsymbol{w}_j=\underbrace{\boldsymbol{w}_j^T\boldsymbol{C}_x\boldsymbol{w}_j}_{\sigma_{yj}^2}-\lambda_j\underbrace{\boldsymbol{w}_j^T\boldsymbol{w}_j}_{1}=\sigma_{yj}^2-\lambda_j=0 \tag{9-14}$$

或者

$$\lambda_j=\sigma_{yj}^2 \tag{9-15}$$

因此，在式（9-15）中对于$j=1$，$\lambda_1=\sigma_{y1}^2$，是协方差矩阵$\boldsymbol{C}_x$的最大特征值，或者向量$\boldsymbol{x}$的输入数据向量分量的线性集合的最大方差，以及相应的特征向量$\boldsymbol{w}_1$表示与最大方差相对应的向量空间方向。

398 现在只要保留前$m$个$\boldsymbol{C}_x$特征向量就可以形成式（9-3）中的变换矩阵$\boldsymbol{W}$，就可以写出式（9-1）中的$\boldsymbol{y}=\boldsymbol{W}\boldsymbol{x}$，向量$\boldsymbol{y}$中的分量就是前$m$个主成分。如果$m$经过仔细选择，输入向量$\boldsymbol{x}$中绝大部分信息内容会包含在向量$\boldsymbol{y}$中。进一步，向量$\boldsymbol{x}$的线性最小二乘估计可以写成

$$\hat{\boldsymbol{x}}=\boldsymbol{C}_x\boldsymbol{W}^T(\boldsymbol{W}\boldsymbol{C}_x\boldsymbol{W}^T)^{-1}\boldsymbol{y} \tag{9-16}$$

以上可以通过均方误差函数（见下式）来推导

$$J(\boldsymbol{x})=\mathrm{E}\{(\boldsymbol{x}-\hat{\boldsymbol{x}})^T(\boldsymbol{x}-\hat{\boldsymbol{x}})\} \tag{9-17}$$

当矩阵$\boldsymbol{W}$的行向量对应于$\boldsymbol{C}_x$的前$m$个主特征向量时，方程（9-17）对应于向量$\boldsymbol{x}$的最小化。我们还可以对应于$m$个主特征向量$[\boldsymbol{w}_1,\boldsymbol{w}_2,\cdots,\boldsymbol{w}_m]$写出向量$\boldsymbol{x}$的近似值，如下

$$\begin{aligned}\hat{\boldsymbol{x}}&=\sum_{h=1}^{m}(\boldsymbol{x}^T\boldsymbol{w}_h)\boldsymbol{w}_h=\sum_{h=1}^{m}\boldsymbol{w}_h(\boldsymbol{x}^T\boldsymbol{w}_h)=\sum_{h=1}^{m}\boldsymbol{w}_h(\boldsymbol{w}_h^T\boldsymbol{x})\\&=(\boldsymbol{w}_1\boldsymbol{w}_1^T+\boldsymbol{w}_2\boldsymbol{w}_2^T+\cdots+\boldsymbol{w}_m\boldsymbol{w}_m^T)\boldsymbol{x}\\&=(\underbrace{[\boldsymbol{w}_1,\boldsymbol{w}_2,\cdots,\boldsymbol{w}_m]}_{\boldsymbol{W}^T}\underbrace{[\boldsymbol{w}_1,\boldsymbol{w}_2,\cdots,\boldsymbol{w}_m]^T}_{\boldsymbol{W}})\boldsymbol{x}=\boldsymbol{W}^T\underbrace{\boldsymbol{W}\boldsymbol{x}}_{\boldsymbol{y}}=\boldsymbol{W}^T\boldsymbol{y}\end{aligned} \tag{9-18}$$

同样，式（9-7）中协方差矩阵$C_x$的近似值可以写作$C_x = W^T \Lambda W$，其中$W \in \Re^{m \times n}$，$\Lambda \in \Re^{m \times m}$，且矩阵$C_x$前$m$个特征值为$\Lambda$的沿对角递减排列的元素。此外，从式（9-6）可以得到$\Lambda = WC_x W^T$，正如前面所提到的，$\Lambda$是输出向量$y$的协方差矩阵。因为$\Lambda$是具有非负元素的对角矩阵，所以$y$的分量是不相关的，且其方差等于矩阵$C_x$的$m$个特征值。

在实际中，协方差矩阵$C_x$和向量$x$的概率分布通常是未知的。但是，典型地，可以收集大量的采样数据向量$x(k)$ ($k = 1, 2, \cdots, N$)。从这$N$个采样向量中，也就是$X = [x(1), x(2), \cdots, x(N)]$，可以对$N$个可能的采样向量依照时间平均[17]，得到一个协方差矩阵的估计，如下

$$C_x \approx \hat{C}_x = \frac{1}{N}\sum_{k=1}^{N} x(k)x^T(k) = \frac{1}{N}XX^T \tag{9-19}$$

总的来说，PCA的目标是要确定与输入的数据相关的$m$个正交的主特征向量集$W = [w_1, w_2, \cdots, w_m]^T \in \Re^{m \times n} (m < n)$。进一步说，这些正交向量应该生成一个输入数据空间，以便于尽可能多地说明输入数据的方差。因此，向量$x$中的信息压缩到包含在向量$y$中的输出（主成分）中。但是，如同在式（9-19）中所描述的，通常对于估计$C_x$必要的信息$x(k)$ ($k = 1, 2, \cdots, N$)，不能集体获得，而可以每次得到一个向量测量。因此，在一般的情况下，必须采用自适应法来求取与未知的协方差矩阵$C_x$相关的主成分。因而，自适应求取算法将在下一步进行讨论，这种方法基于几种神经网络结构，可以通过Hebb学习(规则)来进行训练（参照2.8.2节）。 399

## 9.3　神经网络自适应主成分估计的学习算法

在过去的几年里，许多神经网络方法在协方差矩阵的自适应主成分提取应用中得到了长足的发展[18-95]。以下提出的是四种最著名的自适应主成分提取方法。这些方法之间并不是互不相干的。事实上，它们可以系统化地追溯到单一神经元使用Hebb学习作为主成分分析器的Oja[18]的最初公式。因此，首先给出的将是Oja的单一神经元Hebb学习主成分分析器。单一神经元的情况被扩展到对于若干主成分的估计。用于多重主特征向量求取的单层神经网络结构是由Oja和Karhunen所提出的[20]。用于将单一主成分的情况扩展到若干个主成分估计的典型启发式讨论被关于学习规则的数学推导所取代。从这一对称子空间学习规则可以直接推导出其他两个学习规则，也就是，广义Hebb算法（GHA）[21]以及随机梯度上升（SGA）算法[26]。最后，Kung等[30]提出了APEX（自适应主成分提取）算法。基本上存在两种PCA学习规则：*重估计算法*以及*去相关算法*[40]。下面将研究的前四者是重估计算法，而最后一个（APEX）是去相关算法。

### 9.3.1　第一主成分估计——Oja的正规化Hebb学习规则

建立一个简单的（单一神经元）的神经网络能够求取第一主成分。在1982年，Oja[18]为这个目标提出了单一线性处理单元，见图9-1，从图中可以得到如下的表达式：

$$y_1 = \sum_{j=1}^{n} w_{1j}x_j = x^T w_1 = w_1^T x \tag{9-20}$$

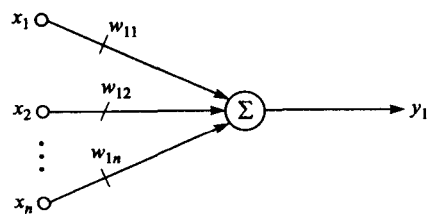


图9-1　第一主成分估计的Oja神经模型

误差表达式可以写成：$e = x - \hat{x}$，其中在单一分量的情

况下，对于$x$的估计从式（9-18）中可以得到$\hat{x}=w_1y_1$。定义如下形式的误差代价（李雅普诺夫）函数

$$\begin{aligned}L(w_1)&=\frac{1}{2}\|e\|_2^2=\frac{1}{2}\|x-\hat{x}\|_2^2=\frac{1}{2}(x-w_1y_1)^T(x-w_1y_1)\\&=\frac{1}{2}(x^Tx-2y_1w_1^Tx+y_1^2w_1^Tw_1)\end{aligned}\tag{9-21}$$

其中$\|\cdot\|_2$是$L_2$（欧几里得）范数，计算相对于权值向量$w_1$的梯度如下

$$\nabla_{w1}L(w_1)=\frac{\partial L(w_1)}{\partial w_1}=\frac{1}{2}\frac{\partial}{\partial w_1}(x^Tx-2y_1w_1^Tx+y_1^2w_1^Tw_1)=-y_1x+y_1^2w_1\tag{9-22}$$

其中要使用A.3.4.1节的结果。

用于提取第一主成分的连续时间学习规则可以通过最速下降梯度方法作为[64, 65]如下形式的向量微分方程来表达

$$\frac{\mathrm{d}w_1(t)}{\mathrm{d}t}=-\mu\nabla_{w1}L(w_1)\tag{9-23}$$

其中学习率参数$\mu>0$。利用式（9-22）中的梯度结果，连续时间学习规则可以写成如下形式

$$\frac{\mathrm{d}w_1(t)}{\mathrm{d}t}=\mu[y_1(t)x(t)-y_1^2(t)w_1(t)]\tag{9-24}$$

以及如下形式的离散时间学习规则

$$w_1(k+1)=w_1(k)+\mu[y_1(k)x(k)-y_1^2(k)w_1(k)]\tag{9-25}$$

其中$k$是时间索引。自适应学习规则称作正规化Hebb或者Oja的规则[18]，其中权值向量的变化可以写作

$$\Delta w_1(k)=w_1(k+1)-w_1(k)=\mu[y_1(k)x(k)-y_1^2(k)w_1(k)]\tag{9-26}$$

根据式（9-25），标量学习规则可以写成

$$w_{1j}(k+1)=w_{1j}(k)+\mu[y_1(k)x_j(k)-y_1^2(k)w_{1j}(k)]\quad\text{对于 } j=1,2,\cdots,n\tag{9-27}$$

已经证明Oja的学习规则收敛于权值向量$w_1$（即，协方差矩阵$C_x$[18，20，39]的第一主特征向量），具有以下性质：对于PCA分析算法的收敛特性的细节，请参阅[94,95]。

1. 权值向量$w_1$具有单位长度，也就是说，$\|w_1\|_2^2=\sum_{h=1}^{n}w_{1h}^2=w_1^Tw_1=1$。

2. 权值向量$w_1$是协方差矩阵$C_x$的特征向量之一。

3. 权值向量$w_1$最大化了输出$y_1=w_1^Tx$的方差。因此，对于0均值输入，$w_1$是第一主特征向量，$y_1$是第一主成分。

400
l
401

位于式（9-26）右侧的第一项是标准Hebb共生参数（参照2.8.2节），右侧第二项是一个活跃衰减或者遗忘项，它防止突触权值向量$w_1$在训练的过程中变得无界。

重点讨论学习率参数$\mu$是因为它对于由式（9-25）的算法收敛性起着直接的作用。如果$\mu$的值太大，学习规则将不收敛。也就是说，学习算法会在数值上表现不稳定。另一方面，如果$\mu$值太小，收敛将会变得极其缓慢。典型地，学习率参数看成是随时间变化的，也就是$\mu=\mu(k)$，因此应该在一开始设置一个较大的值，然后逐渐减小直到得到希望的精确度。实际上，为了确保算法在数值上的稳定性，学习率必须满足不等式$0<\mu(k)<1/(1.2\lambda_1)$，其中$\lambda_1$是协方

差矩阵$C_x$的最大特征值[20]。使用一个良好的初始值应该基于输入$x(k)$，也就是，$\mu(k) = 1/[2x^T(k)x(k)]$。这样将会确保初始的收敛性，因为这样步长将会相对较大。然后，学习率逐渐减小，以便于精细调整权值更新，从而得到一个精确的突触权值向量$w_1$[20，38，62]估计。很多时候，在数值梯度搜索算法中，遗忘因子[2]或者泄露因子被引入，作为学习率的一种表达。在Cichocki和Unbehauen[38]的著作中，给出了一种如下的可变学习率

$$\mu_i(k) = \frac{1}{\dfrac{\gamma}{\mu_i(k-1)} + y_i^2(k)} \quad 和 \quad \mu_i(0) = \frac{1}{y_i^2(0)} 对于 \quad i = 1, 2, \cdots, m \tag{9-28}$$

其中$\gamma$是在范围$0 \leqslant \gamma \leqslant 1$中的遗忘因子。通常，遗忘因子在范围$0.9 \leqslant \gamma \leqslant 1$。对于估计第一主成分，在式（9-28）中$i = 1$。因此，式（9-28）经常表示成求取$m$个主成分的各个学习率或者可以用作求取$m$个主成分的$\mu$的单一值，见式（9-48）。

**例9.1**　求取第一分量的Oja学习规则要生成1 000个零均值高斯随机三维向量，其中$x$分量具有单位方差，$y$分量和$z$分量的方差为0.002。在随机选择初始权值向量的情况下，完成式（9-25）所给出的离散时间学习规则，网络在726次迭代后收敛（即，没有必要输入所有的1 000个训练向量）。收敛是通过计算权值向量在更新之前和之后的差的$L_2$范数来确定的。学习率参数根据式（9-28）自适应校正，其中$\gamma = 1$。

在图9-2中，可以在三维环境中观察权值向量的收敛路径，其中圆圈部分代表初始随机权值向量，而最终的权值向量以星号标记。通过神经网络计算的最终权值向量为$w_1^{NN} = [-1.000\ 0, -0.000\ 2, -0.001\ 6]^T$。利用MATLAB，根据式（9-19）计算的协方差矩阵$C_x$的对应于最大特征值（$\lambda_1^M = 0.984\ 3$）的特征向量是$w_1^M = [1.000\ 0, -0.000\ 2, 0.001\ 1]^T$。注意，$w_1^{NN}$与$w_1^M$基本相同，只是相差一个负号，这种情况经常发生，因为符号的不确定与特征向量相关。如图9-2 402
中所描述，在收敛到第一主向量之后，所示的方向（图中的实线）代表最大方差的方向。最后，方差（即特征值）可以通过$X$中的数据来计算，其中包含1 000个样本向量，权值向量通过神经网络$w_1^{NN}$来计算，也就是，

$$\lambda_1^{NN} = \mathrm{var}(w_1^{NN\,T} X) = 0.985\ 2 \tag{9-29}$$

以上计算的结果与使用MATLAB计算的特征值一致。

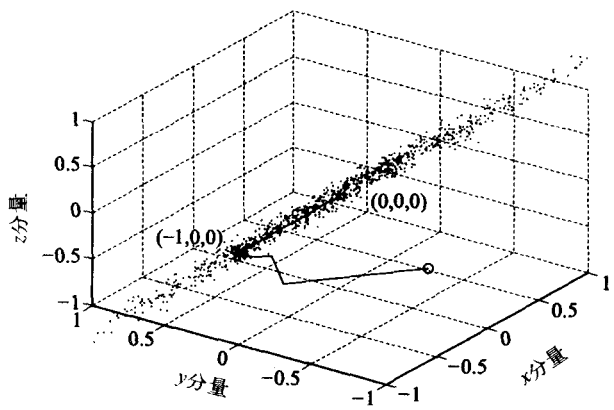



图9-2　例9.1中的样本向量和协方差矩阵$C_x$的第一主特征向量（权值向量）的收敛路径，通过神经网络自适应计算（遗忘因子$\gamma = 1$）

在我们继续估计其他若干主成分之前，需要讨论一下学习规则[18, 40]的正规化问题。为

了展示上面提出的Oja学习规则是正规化的Hebb学习，可以从如下的典型形式的Hebb学习（离散时间标量形式）开始

$$w_j(k+1) = w_j(k) + \mu(k)y(k)x_j(k) \tag{9-30}$$

其中$j = 1, 2, \cdots, n$。这种基本形式的学习规则导致权触突触$w_j(k)(j = 1, 2, \cdots, n)$的无限增长，这将使权值向量$\boldsymbol{w}$无法收敛。这一点可以通过在自适应变化突触权值的学习规则中加入一些正规化（或者饱和）的形式来克服。正规化所引起的神经突触之间的竞争将会产生稳定性。利用式（9-30），可以将合理的正规化形式写成

$$w_j(k+1)=\frac{w_j(k)+\gamma\,\mu(k)y(k)x_j(k)}{\left\{\sum_{j=1}^{n}[w_j(k)+\gamma\,\mu(k)y(k)x_j(k)]^2\right\}^{1/2}} \tag{9-31}$$

403 其中式（9-31）分母的总和会扩展到整个与神经元相关的突触系列，可塑性系数$\gamma$将保持Oja[18]的方式。因此，式（9-31）表示可以通过截断表达式的幂级数展开来简化正规化学习规则。首先，再次将突触权值向量处理成单位长度，即$\|\boldsymbol{w}(k)\|_2=\left[\sum_{j=1}^{n}w_j^2(k)\right]^{1/2}=1$。第二，将式（9-31）展开成为关于$\gamma=0$的幂级数如下

$$w_j(k+1)=\frac{w_j(k+1)\big|_{\gamma=0}}{0!}+\frac{\left[\dfrac{\mathrm{d}w_j(k+1)}{\mathrm{d}\gamma}\right]_{\gamma=0}}{1!}\gamma+O(\gamma^2) \tag{9-32}$$

假定$\gamma\ll1$，式（9-32）中只有前两项保留。$O(\gamma^2)$项表示$\gamma$的高阶作用，所以忽略。利用式（9-31），式（9-32）的第一项可以写成如下形式。

$$\frac{w_j(k+1)\big|_{\gamma=0}}{0!}=\frac{w_j(k)}{\left[\sum_{j=1}^{n}w_j^2(k)\right]^{1/2}}=\underbrace{\frac{w_j(k)}{\|\boldsymbol{w}(k)\|_2}}_{1}=w_j(k) \tag{9-33}$$

对于式（9-32）中的第二项，可以在式（9-31）中定义如下

$$\alpha = w_j(k) + \gamma\mu(k)y(k)x_j(k) \tag{9-34}$$

同时

$$\beta=\left\{\sum_{j=1}^{n}[w_j(k)+\gamma\mu(k)y(k)x_j(k)]^2\right\}^{1/2} \tag{9-35}$$

因此有，

$$\frac{\left[\dfrac{\mathrm{d}w_j(k+1)}{\mathrm{d}\gamma}\right]_{\gamma=0}}{1!}\gamma=\frac{\left(\beta\dfrac{\mathrm{d}\alpha}{\mathrm{d}\gamma}\right)_{\gamma=0}-\left(\alpha\dfrac{\mathrm{d}\beta}{\mathrm{d}\gamma}\right)_{\gamma=0}}{\beta^2\big|_{\gamma=0}}=\gamma\mu(k)[y(k)x_j(k)-w_j(k)y^2(k)] \tag{9-36}$$

将式（9-33）以及式（9-36）的结果带入式（9-32），其中$\gamma = 1$[$\mu(k)$足够小]，同时忽略$O(\gamma^2)$项，可以得到

$$w_j(k+1) = w_j(k) + \mu(k)[y(k)x_j(k) - y^2(k)w_j(k)] \quad \text{对于 } j = 1, 2, \cdots, n \tag{9-37}$$

这是用于估计式（9-27）中的第一主特征向量的Oja离散时间标量形式的学习规则。式（9-31）中的正规化同样可以通过先假定可塑性系数$\gamma = 1$来推导出，然后，假定$\mu(k) \ll 1$，可以重新得到式（9-37）中的近似结果。

### 9.3.2 多个主成分估计——对称子空间学习规则

一种若干（$m$）个主成分自适应估计情况的典型方式是启发式地扩展用于估计第一主成分 404
的Oja结果。以上观点是基于通过权值矩阵$\boldsymbol{W} \in \Re^{m \times n}$来取代式（9-25）中的权值向量，用向量$\boldsymbol{y} \in \Re^{m \times 1}$取代标量$y_1$而提出的。但是，这不是我们要采用的方法。我们将更加严谨并利用与以往求取单一主成分情况下的学习规则相似的设置来导出这一学习规则。

图9-3所描述的首先是由Karhunen与Oja [20, 26, 62]一起提出的单层线性神经网络，这也是我们进行推导的基础。式（9-1）的原始PCA映射$\boldsymbol{y} = \boldsymbol{Wx}$如图9-3所示，可以估计若干（$m$）个主成分。我们要定义误差向量为$\boldsymbol{e} = \boldsymbol{x} - \hat{\boldsymbol{x}} \in \Re^{n \times 1}$，其中正如式（9-18）给出的$\hat{\boldsymbol{x}} = \boldsymbol{W}^T\boldsymbol{y} = \boldsymbol{W}^T\boldsymbol{Wx}$。同样我们定义误差代价（李雅普诺夫）函数如下

$$\begin{aligned} L(\boldsymbol{W}) &= \frac{1}{2}\|\boldsymbol{e}\|_2^2 = \frac{1}{2}\boldsymbol{e}^T\boldsymbol{e} = \frac{1}{2}(\boldsymbol{x} - \hat{\boldsymbol{x}})^T(\boldsymbol{x} - \hat{\boldsymbol{x}}) = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{W}^T\boldsymbol{Wx})^T(\boldsymbol{x} - \boldsymbol{W}^T\boldsymbol{Wx}) \\ &= \frac{1}{2}(\boldsymbol{x}^T\boldsymbol{x} - 2\boldsymbol{x}^T\boldsymbol{W}^T\boldsymbol{Wx} + \boldsymbol{x}^T\boldsymbol{W}^T\boldsymbol{WW}^T\boldsymbol{Wx}) \end{aligned} \tag{9-38}$$

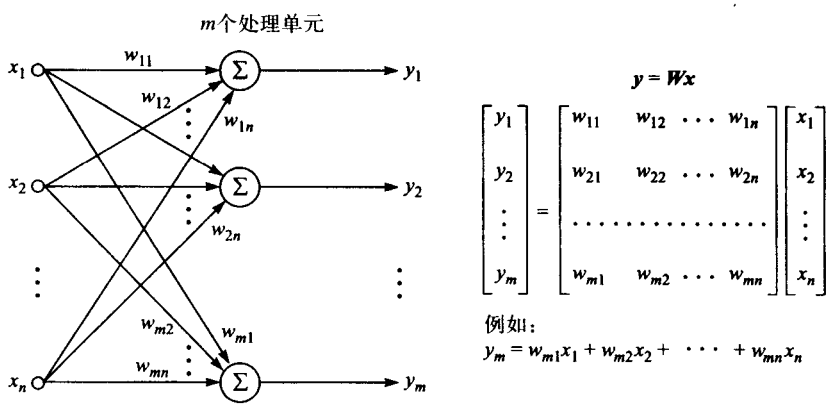



图9-3　求取多（$m$）个主成分的Karhunen和Oja的单层线性神经网络

自适应估计$m$个主成分（或者主特征向量）的连续时间学习规则可以使用最速下降梯度方法[64，65]，表达成矩阵微分方程的形式如下

$$\frac{\mathrm{d}\boldsymbol{W}(t)}{\mathrm{d}t} = -\mu\nabla_{\boldsymbol{W}}\boldsymbol{L}(\boldsymbol{W}) \tag{9-39}$$

其中$\mu$是学习率参数。因此，式（9-38）的梯度$\boldsymbol{L}(\boldsymbol{W})$一定要计算。为了完成这一点，我们需要A.3.4.2节中所提出的对标量求矩阵微分的两个一般结果，也就是，

$$\frac{\partial}{\partial \boldsymbol{A}}\mathrm{trace}(\boldsymbol{BAC}) = \boldsymbol{B}^T\boldsymbol{C}^T \tag{9-40}$$ 405

以及

$$\frac{\partial}{\partial \boldsymbol{A}}\operatorname{trace}(\boldsymbol{B}\boldsymbol{A}^T\boldsymbol{C})=\boldsymbol{C}\boldsymbol{B} \tag{9-41}$$

通过使用式（9-40）以及式（9-41）中的一般结果以及适当的链式规则，梯度$\boldsymbol{L}(\boldsymbol{W})$可以写成如下形式

$$\nabla_W L(\boldsymbol{W})=\frac{\partial L(\boldsymbol{W})}{\partial \boldsymbol{W}}=-\boldsymbol{W}\boldsymbol{x}\boldsymbol{x}^T+\boldsymbol{W}\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{W}^T\boldsymbol{W}-\boldsymbol{W}\boldsymbol{x}\boldsymbol{x}^T+\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{x}\boldsymbol{x}^T \tag{9-42}$$

但是，式（9-42）中最右边的两项会非常快地逼近0，因为最右边的项$\boldsymbol{W}\boldsymbol{W}^T\to\boldsymbol{I}\in\Re^{m\times m}$，也就是，

$$-\boldsymbol{W}\boldsymbol{x}\boldsymbol{x}^T+\underbrace{\boldsymbol{W}\boldsymbol{W}^T}_{\boldsymbol{I}}\boldsymbol{W}\boldsymbol{x}\boldsymbol{x}^T=0 \tag{9-43}$$

因此，使用这一近似，式（9-42）中的梯度可以带入到式（9-39），从而得到如下形式的连续时间学习规则

$$\frac{\mathrm{d}\boldsymbol{W}(t)}{\mathrm{d}t}=\mu\boldsymbol{W}(t)\boldsymbol{x}(t)\boldsymbol{x}^T(t)[\boldsymbol{I}-\boldsymbol{W}^T(t)\boldsymbol{W}(t)] \tag{9-44}$$

离散时间的学习规则，可以写成如下形式

$$\boldsymbol{W}(k+1)=\boldsymbol{W}(k)+\mu(k)\boldsymbol{W}(k)\boldsymbol{x}(k)\boldsymbol{x}^T(k)[\boldsymbol{I}-\boldsymbol{W}^T(k)\boldsymbol{W}(k)] \tag{9-45}$$

其中，我们认为学习参数是自适应的，即$\mu(k)$。式（9-45）中的离散时间批量学习规则称作Karhunen-Oja对称子空间学习规则[20, 41, 62]。式（9-45）中的学习规则同样可以利用已定义的误差向量$\boldsymbol{e}=\boldsymbol{x}-\hat{\boldsymbol{x}}=\boldsymbol{x}-\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{x}$以及$\boldsymbol{y}=\boldsymbol{W}\boldsymbol{x}$，写成如下形式

$$\boldsymbol{W}(k+1)=\boldsymbol{W}(k)+\mu(k)\boldsymbol{y}(k)\boldsymbol{e}^T(k) \tag{9-46}$$

这说明了它是基于误差反向传播的学习算法。在式（9-45）中权值更新表达式的第一项，即$\boldsymbol{W}\boldsymbol{x}\boldsymbol{x}^T$，就是直接遵从输入数据的标准Hebb项。第二项$\boldsymbol{W}\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{W}^T\boldsymbol{W}$是非线性衰减项，它确保突触权值矩阵$\boldsymbol{W}$接近于正交[20，41，62]。

标量离散时间学习规则直接从式（9-45），可以写成如下

$$w_{ij}(k+1)=w_{ij}(k)+\mu(k)y_i(k)\left[x_j(k)-\sum_{h=1}^{m}w_{hj}(k)y_h(k)\right] \tag{9-47}$$

其中$i=1,2,\cdots,m$；$j=1,2,\cdots,n$以及$y_i(k)=\boldsymbol{w}_i^T(k)\boldsymbol{x}(k)$。在式（9-47）中，如果$i$，$m=1$，学习规则可以化简成式（9-27）的Oja单一处理单元学习规则。

这一对称子空间学习规则的一个有趣方面是权值矩阵$\boldsymbol{W}$的行向量并没有收敛到协方差矩
406 阵的实际主特征向量，却收敛到了$\boldsymbol{C}_x$前$m$个主特征向量的线性组合。因此，神经网络学习由$m$个主成分所张成的空间。但是，最终的特征向量$\boldsymbol{w}_i$（其中$i=1,2,\cdots,m$）并没有在实际的主特征向量的方向[37, 39]。此外，式（9-45）或式（9-47）中的学习规则可以导出权值向量$\boldsymbol{w}_i$，其中$i=1,2,\cdots,m$，其所张成的子空间与$\boldsymbol{C}_x$的前$m$个主向量的一致。但是，每次实验都是不同的。也就是，在训练神经网络之前，最初设置的条件和用于训练的实际数据采样将决定通过网络提取的正交向量的最终结果。同样，和使用Karhunen-Oja对称子空间学习规则求取的特征向量相关的特征值（方差），倾向于以均匀的方式分布。因此，计算所得方差并不是协方差矩阵$\boldsymbol{C}_x$实际的特征值。但是前者的和与$\boldsymbol{C}_x$的实际特征值的和是相等的。

在前一节，我们提出了一种用于求取$m$个主成分的带有用户自定义遗忘因子（$\gamma$）的自适应改变学习率$\mu(k)$的方法，如式（9-28）所示。这一自适应方法可以用于式（9-45）或式（9-47）中的对称子空间学习规则的学习参数。另一种只需计算单一学习参数，且可以用于利用

式（9-45）求取所有的$m$个主成分的可选的方式由下式给出

$$\mu(k)=\frac{1}{\dfrac{\gamma}{\mu(k-1)}+\|\boldsymbol{y}(k)\|_2^2}\text{ 和 }\mu(0)=\frac{1}{\|\boldsymbol{y}(0)\|_2^2}\quad\text{对于}\quad k=1,2,3,\cdots,\text{和}0\leqslant\gamma\leqslant 1 \tag{9-48}$$

因此，在学习规则中，学习率参数可以为$\mu(k)$ 或者$\mu_i(k)$，这取决于所用的更新方式。在式（9-48）中，$\|\boldsymbol{y}\|_2^2$可以用$\|\boldsymbol{y}\|_\infty$来代替（即，$\max|y_i|$，其中$i=1,2,\cdots,m$）。

接下来的两个PCA学习算法可以直接通过式（9-47）的标量对称子空间学习规则推导出来。在下一节，式（9-47)中将首先推导标量学习规则，同时广义的Hebb算法可以直接由这一结果推导出。然后，可以从GHA学习规则中推导出Oja的随机梯度上升算法。我们需要指出，此时所有学习规则的标量形式都是可以在并行神经结构中实际完成的形式。在特殊神经网络发展中，这些算法的批量形式典型地用于分析和操作仿真的目的。

### 9.3.3　多个主成分估计——广义Hebb算法

在1989年，Sanger[21，36]提出了一种学习规则，自适应提取协方差矩阵$\boldsymbol{C}_x$的前$m$个主特征向量。这一学习规则称作广义Hebb算法，它可以从式（9-45）中Karhunen-Oja离散时间学习规则推导而得到。我们将首先推导式（9-47），这要将式（9-45）的学习规则的批量向量矩阵形式写成 407

$$\begin{bmatrix}\boldsymbol{w}_1^T(k+1)\\ \boldsymbol{w}_2^T(k+1)\\ \vdots\\ \boldsymbol{w}_m^T(k+1)\end{bmatrix}=\begin{bmatrix}\boldsymbol{w}_1^T(k)\\ \boldsymbol{w}_2^T(k)\\ \vdots\\ \boldsymbol{w}_m^T(k)\end{bmatrix}+\mu(k)\underbrace{\begin{bmatrix}\boldsymbol{w}_1^T(k)\boldsymbol{x}(k)\\ \boldsymbol{w}_2^T(k)\boldsymbol{x}(k)\\ \vdots\\ \boldsymbol{w}_m^T(k)\boldsymbol{x}(k)\end{bmatrix}}_{\boldsymbol{y}(k)}$$

$$\cdot\left\{\boldsymbol{x}^T(k)-\frac{[\boldsymbol{x}^T(k)\boldsymbol{w}_1(k),\boldsymbol{x}^T(k)\boldsymbol{w}_2(k),\cdots,\boldsymbol{x}^T(k)\boldsymbol{w}_m(k)]}{\boldsymbol{y}^T(k)=[y_1(k),y_2(k),\cdots,y_m(k)]}\begin{bmatrix}\boldsymbol{w}_1^T(k)\\ \boldsymbol{w}_2^T(k)\\ \vdots\\ \boldsymbol{w}_m^T(k)\end{bmatrix}\right\}$$

$$=\begin{bmatrix}\boldsymbol{w}_1^T(k)\\ \boldsymbol{w}_2^T(k)\\ \vdots\\ \boldsymbol{w}_m^T(k)\end{bmatrix}+\mu(k)\begin{bmatrix}y_1(k)\\ y_2(k)\\ \vdots\\ y_m(k)\end{bmatrix}$$

$$\cdot\left\{\boldsymbol{x}^T(k)-\underbrace{[y_1(k),y_2(k),\cdots,y_m(k)]\begin{bmatrix}w_{11}(k) & w_{12}(k) & \cdots & w_{1n}(k)\\ w_{21}(k) & w_{22}(k) & \cdots & w_{2n}(k)\\ \cdots & \cdots & \cdots & \cdots\\ w_{m1}(k) & w_{m2}(k) & \cdots & w_{mn}(k)\end{bmatrix}}_{\left[\sum_{h=1}^{m}w_{h1}(k)y_h(k),\sum_{h=1}^{m}w_{h2}(k)y_h(k),\cdots,\sum_{h=1}^{m}w_{hn}(k)y_h(k)\right]}\right\} \tag{9-49}$$

因此，由式（9-49）可以将标量离散时间形式的Karhunen-Oja的对称子空间学习规则写成如下形式

$$w_{ij}(k+1)=w_{ij}(k)+\mu(k)y_i(k)\left[x_j(k)-\sum_{h=1}^{m}w_{hj}(k)y_h(k)\right] \tag{9-50}$$

其中，$i=1,2,\cdots,m$；$j=1,2,\cdots,n$，这就是由式（9-47）所提出的表达式。由式（9-49）中的表达式，我们可以将学习规则写成

$$\begin{bmatrix}\boldsymbol{w}_1^T(k+1)\\ \boldsymbol{w}_2^T(k+1)\\ \vdots\\ \boldsymbol{w}_m^T(k+1)\end{bmatrix}=\begin{bmatrix}\boldsymbol{w}_1^T(k)\\ \boldsymbol{w}_2^T(k)\\ \vdots\\ \boldsymbol{w}_m^T(k)\end{bmatrix}+\mu(k)\left\{\boldsymbol{y}(k)\boldsymbol{x}^T(k)-\boldsymbol{y}(k)\boldsymbol{y}^T(k)\begin{bmatrix}\boldsymbol{w}_1^T(k)\\ \boldsymbol{w}_2^T(k)\\ \vdots\\ \boldsymbol{w}_m^T(k)\end{bmatrix}\right\} \tag{9-51}$$

Sanger学习算法可以通过式（9-51）利用“打破”与权值更新项中的矩阵$\boldsymbol{y}(k)\boldsymbol{y}^T(k)$相关的对称
408 来推导得到。这一点可以只保持对称矩阵$\boldsymbol{y}(k)\boldsymbol{y}^T(k)$的下三角部分来完成，也就是，应用运算符LT{$\boldsymbol{y}(k)\boldsymbol{y}^T(k)$}，其中LT{•}选择矩阵下三角部分，包括矩阵的对角线元素[21]。对式（9-51）中的矩阵$\boldsymbol{y}(k)\boldsymbol{y}^T(k)$应用这一运算符，可以得到

$$\begin{bmatrix}\boldsymbol{w}_1^T(k+1)\\ \boldsymbol{w}_2^T(k+1)\\ \vdots\\ \boldsymbol{w}_m^T(k+1)\end{bmatrix}=\begin{bmatrix}\boldsymbol{w}_1^T(k)\\ \boldsymbol{w}_2^T(k)\\ \vdots\\ \boldsymbol{w}_m^T(k)\end{bmatrix}+\mu(k)\left\{\boldsymbol{y}(k)\boldsymbol{x}^T(k)-\mathrm{LT}\{\boldsymbol{y}(k)\boldsymbol{y}^T(k)\}\begin{bmatrix}\boldsymbol{w}_1^T(k)\\ \boldsymbol{w}_2^T(k)\\ \vdots\\ \boldsymbol{w}_m^T(k)\end{bmatrix}\right\}=\begin{bmatrix}\boldsymbol{w}_1^T(k)\\ \boldsymbol{w}_2^T(k)\\ \vdots\\ \boldsymbol{w}_m^T(k)\end{bmatrix}$$

$$+\mu(k)\left\{\boldsymbol{y}(k)\boldsymbol{x}^T(k)-\mathrm{LT}\begin{bmatrix}y_1(k)y_1(k) & y_1(k)y_2(k) & \cdots & y_1(k)y_m(k)\\ y_2(k)y_1(k) & y_2(k)y_2(k) & \cdots & y_2(k)y_m(k)\\ \cdots & \cdots & \cdots & \cdots\\ y_m(k)y_1(k) & y_m(k)y_2(k) & \cdots & y_m(k)y_m(k)\end{bmatrix}\begin{bmatrix}\boldsymbol{w}_1^T(k)\\ \boldsymbol{w}_2^T(k)\\ \vdots\\ \boldsymbol{w}_m^T(k)\end{bmatrix}\right\}$$

$$=\begin{bmatrix}\boldsymbol{w}_1^T(k)\\ \boldsymbol{w}_2^T(k)\\ \vdots\\ \boldsymbol{w}_m^T(k)\end{bmatrix}$$

$$+\mu(k)\left\{\boldsymbol{y}(k)\boldsymbol{x}^T(k)-\underbrace{\begin{bmatrix}y_1(k)y_1(k) & 0 & \cdots & 0\\ y_2(k)y_1(k) & y_2(k)y_2(k) & \cdots & 0\\ \cdots & \cdots & \cdots & \cdots\\ y_m(k)y_1(k) & y_m(k)y_2(k) & \cdots & y_m(k)y_m(k)\end{bmatrix}\begin{bmatrix}\boldsymbol{w}_1^T(k)\\ \boldsymbol{w}_2^T(k)\\ \vdots\\ \boldsymbol{w}_m^T(k)\end{bmatrix}}_{\begin{bmatrix}y_1(k)[y_1(k)\boldsymbol{w}_1^T(k)]\\ y_2(k)[y_1(k)\boldsymbol{w}_1^T(k)+y_2(k)\boldsymbol{w}_2^T(k)]\\ y_3(k)[y_1(k)\boldsymbol{w}_1^T(k)+y_2(k)\boldsymbol{w}_2^T(k)+y_3(k)\boldsymbol{w}_3^T(k)]\\ \vdots\\ y_m(k)[y_1(k)\boldsymbol{w}_1^T(k)+y_2(k)\boldsymbol{w}_2^T(k)+y_3(k)\boldsymbol{w}_3^T(k)+\cdots+y_m(k)\boldsymbol{w}_m^T(k)]\end{bmatrix}}\right\} \tag{9-52}$$

因此，式（9-52）中Sanger的离散时间标量PCA学习规则可以写成如下形式

$$w_{ij}(k+1)=w_{ij}(k)+\mu(k)y_i(k)\left[x_j(k)-\sum_{h=1}^{i}w_{hj}(k)y_h(k)\right] \tag{9-53}$$

其中$i = 1, 2, \cdots, m$以及$j = 1, 2, \cdots, n$。

将式（9-53）中Sanger的GHA学习规则与式（9-50）中Karhunen与Oja的对称子空间学习规则对比，我们发现唯一的不同就是求和极限。通过分解对称的$\boldsymbol{y}(k)\boldsymbol{y}^T(k)$，即只保留式（9-52）中的矩阵的下三角部分，随着$i$从1→$m$，式（9-53）中求和的项数也依次增加，正如式（9-52）所述。对于式（9-50）的对称子空间学习规则，随着$i$从1→$m$，求和中项的数目是常数，即$m$ 409
（前$m$个主成分）。在两种学习规则中，在求和极限中的不同说明了Karhunen-Oja对称子空间学习规则并没有提取实际的主特征向量，而是作为代替线性组合张成与$m$个实际的主特征向量相同的子空间。而式（9-53）中Sanger的PCA学习规则（GHA）提取协方差矩阵$\boldsymbol{C}_x$[21，36]的$m$个实际的主特征向量。此外，已经证明了使用Sanger的GHA学习规则训练的神经网络将从任何的随机权值的初始集合依照特征值递减的次序收敛到协方差矩阵$\boldsymbol{C}_x$的特征值和特征向量[39]。同Karhunen-Oja对称子空间学习规则的情况一样，Sanger的GHA可以根据式（9-28）或式（9-48）及时调整自适应学习参数。因此，在学习规则式（9-53）中，学习率参数依赖所采用的更新方式，可取为$\mu(k)$或者$\mu_i(k)$。

式（9-53）中的GHA学习规则也可以写成一种递归向量的形式，如下

$$\boldsymbol{w}_i(k+1) = \boldsymbol{w}_i(k) + \mu_i(k)y_i(k)\tilde{\boldsymbol{x}}_i(k) \tag{9-54}$$

其中

$$\tilde{\boldsymbol{x}}_i(k) = \tilde{\boldsymbol{x}}_{i-1}(k) - \boldsymbol{w}_i(k)y_i(k) \tag{9-55}$$

同时

$$\tilde{\boldsymbol{x}}_0(k) = \boldsymbol{x}(k) \quad \text{对于} i = 1, 2, \cdots, m \tag{9-56}$$

### 9.3.4　多个主成分估计——随机梯度上升算法

另一种与Sanger的GHA算法紧密相关的PCA神经网络学习算法是由Oja[26]提出的随机梯度上升（SGA）算法。SGA学习算法可以从GHA直接推导。式（9-53）中的GHA拥有相对于$i$的求和极限。在SGA算法中，求和极限看作是分解式（9-53）中的求和极限得到的$i-1$，如下

$$\sum_{h=1}^{i} w_{hj}(k)y_h(k) \Rightarrow \sum_{h=1}^{i-1} w_{hj}(k)y_h(k) + w_{ij}(k)y_i(k) \tag{9-57}$$

如果式（9-57）的右边求和通过标量$\alpha$来加权，那么Oja的离散时间标量SGA算法可以写成如下形式

$$w_{ij}(k+1) = w_{ij}(k) + \mu(k)y_i(k)\left[x_j(k) - w_{ij}(k)y_i(k) - \alpha\sum_{h=1}^{i-1} w_{hj}(k)y_h(k)\right] \tag{9-58}$$

其中$i = 1, 2, \cdots, m$；$j = 1, 2, \cdots, n$且$\alpha > 1$（一般取为$\alpha = 2$）。SGA算法相比GHA算法的一个主要优点是在求取较少的支配分量时的行为。即当求取相比最初的一些特征向量具有较小支配的主特征向量时，在SGA算法的收敛速度比GHA算法好。SGA算法的另一个优点是它可以简 410
单地扩展用于提取次分量[26]。

### 9.3.5　多个主成分估计——自适应主成分提取算法

PCA的自适应主成分提取（APEX）算法基于反Hebb学习，且认为是去相关类型的算法[40]。首先由Kung et al.[25, 30, 81]提出的APEX，应用一种新类型的PCA称作约束PCA（CPCA）[66]。以并行方式[67]逐次从输入模式中求取协方差矩阵$\boldsymbol{C}_x$的主特征向量，来并行估

计它们。由Bannour和Azimi-Sadjadi[68]开发的相关方法是基于压缩变换[96, 97]，和输出结点之间的侧向连接一样，它使得他们的方法本质上为串行的。图9-4中描述了线性APEX模型。这一结构与前面的PCA网络的主要不同在于网络输出处额外的侧向连接，如图9-4所示。这些侧向连接产生了拓扑布局的反馈和突触权值的正交化。APEX结构的另一特征是允许随模型的大小而生长或者收缩，而不用重新训练已有单元。也就是，如果要提取一个或者更多的附加分量，附加神经元的大约数目可以简单地与以前的模型相关联。

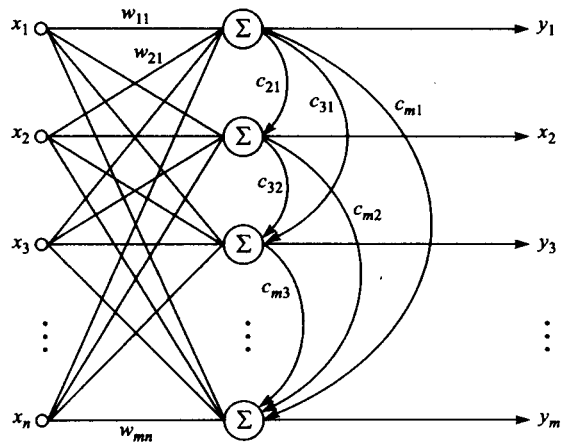



图9-4　包含线性处理单元的并行APEX模型

对于图9-4中的模型，假定$\boldsymbol{x}\in\Re^{n\times 1}$，$\boldsymbol{y}\in\Re^{m\times 1}$，$\boldsymbol{C}\in\Re^{m\times(m-1)}$，以及$\boldsymbol{W}\in\Re^{m\times n}$。因此，两种不同类型的突触权值用于APEX模型，前馈连接权值$\boldsymbol{W}$和侧向连接权值$\boldsymbol{C}$。

411　1. 前馈连接权值$\boldsymbol{w}_i(k)=[w_{i1}(k)\ w_{i2}(k)\ \cdots\ w_{in}(k)]^T$（$i=1,2,\cdots,m$）（其中$m$是希望提取的主成分数），是与从输入到线性神经元分枝相关的突触。前馈突触权值的适应性是与标准的Hebb学习相一致的，其中连接组成了兴奋信号，因此只要自我放大。

2. 侧向连接权值由单个神经元输出到所有的连续神经元所组成的连接分支相关联，其中$i=1,2,\cdots,m-1$，因此考虑网络接受的反馈（见图9-4）。侧向连接权值的适应是根据反Hebb学习来执行的。

反馈（或者侧向）连接权值矩阵可以写成

$$\boldsymbol{C}(k)=\begin{bmatrix}0 & 0 & 0 & 0 & 0 & \cdots & 0\\ c_{21}(k) & 0 & 0 & 0 & 0 & \cdots & 0\\ c_{31}(k) & c_{32}(k) & 0 & 0 & 0 & \cdots & 0\\ c_{41}(k) & c_{42}(k) & c_{43}(k) & 0 & 0 & \cdots & 0\\ c_{51}(k) & c_{52}(k) & c_{53}(k) & c_{54}(k) & 0 & \cdots & 0\\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots\\ c_{m1}(k) & c_{m2}(k) & c_{m3}(k) & c_{m4}(k) & c_{m5}(k) & \cdots & c_{mm-1}(k)\end{bmatrix}=\begin{bmatrix}\boldsymbol{c}_1(k)\\ \boldsymbol{c}_2(k)\\ \boldsymbol{c}_3(k)\\ \boldsymbol{c}_4(k)\\ \boldsymbol{c}_5(k)\\ \vdots\\ \boldsymbol{c}_m(k)\end{bmatrix} \tag{9-59}$$

其中，$\boldsymbol{c}_i\in\Re^{1\times(m-1)}$（$i=1,2,\cdots,m$）就是矩阵$\boldsymbol{C}$的行向量。式（9-59）中所描述的矩阵$\boldsymbol{C}$的第一行的所有分量均为0，因为并没有与第一神经元相关的侧向连接。此外，第一主特征向量（$\boldsymbol{w}_1$）的提取是根据Oja的单一神经元学习规则来执行的，如式（9-25）（参照9.3.1节）。式（9-59）的侧向权值用作连续地减掉主成分信息，如根据下式训练每一个连续的神经元（见图9-4）

$$y_i(k)=\boldsymbol{w}_i^T(k)\boldsymbol{x}(k)-\boldsymbol{c}_i(k)\boldsymbol{y}(k) \tag{9-60}$$

其中$i=1,2,\cdots,m$，且$\boldsymbol{y}(k)=\boldsymbol{W}(k)\boldsymbol{x}(k)$。侧向连接权值根据如下的反Hebb学习规则来更新

$$\boldsymbol{c}_i(k+1)=\boldsymbol{c}_i(k)+\mu[y_i(k)\boldsymbol{y}^T(k)-y_i^2(k)\boldsymbol{c}_i(k)] \tag{9-61}$$

412　其中$i=2,3,\cdots,m$（$i$从2开始，因为侧向连接权值矩阵$\boldsymbol{C}$的第一行向量，即$\boldsymbol{c}_1$假定为包含全部0值分量）。前馈连接权值根据下式的标准Hebb学习规则来更新

$$\boldsymbol{w}_i(k+1)=\boldsymbol{w}_i(k)+\mu[y_i(k)\boldsymbol{x}(k)-y_i^2(k)\boldsymbol{w}_i(k)] \tag{9-62}$$

其中$i = 1, 2, \cdots, m$。学习率参数在两个更新方程（即式（9-61）与式（9-62））中假定是相同的。在Kung et al.[30]，提出了两个不同的自适应主成分求取的结构，即并行APEX与顺序APEX。假定以下是一个并行APEX模型的算法。这一模型同神经类型的学习更加协调，它就是并行处理。

**并行APEX算法小结**

**步骤1**　选择学习率参数$\mu > 0$。可以是一个固定值，即可以为$\mu$选择一个足够小的值，或者可以通过使用Kung和Diamantaras[69]，Kung et al.[30]以及Haykin[40]所提出的方法来提前计算（估计）。此外，$\mu$还可以使用式（9-28）的方法迭代计算，在Kung et al.[30]中也提及。

**步骤2**　随机初始化$\boldsymbol{W}$以及$\boldsymbol{C}$，同时令$k = 1$。

**步骤3**　对于$i = 1$，计算

$$y_1(k) = \boldsymbol{w}_1^T(k)\boldsymbol{x}(k) \tag{9-63}$$

其中$\boldsymbol{w}_1^T$是前馈连接权值矩阵$\boldsymbol{W}$的第一行向量，根据Oja的学习规则更新单一神经元模型的第一突触权值向量，也就是

$$\boldsymbol{w}_1(k+1) = \boldsymbol{w}_1(k) + \mu[y_1(k)\boldsymbol{x}(k) - y_1^2(k)\boldsymbol{w}_1(k)] \quad (\boldsymbol{w}_1 \in \Re^{n\times 1}) \tag{9-64}$$

**步骤4**　对于$i = 1, 2, \cdots, m$，计算

$$\boldsymbol{y}(k) = \boldsymbol{W}(k)\boldsymbol{x}(k) \tag{9-65}$$

重点注意算法的这一步$\boldsymbol{W}(k) = [\boldsymbol{w}_1(k)\ \boldsymbol{w}_2(k) \cdots \boldsymbol{w}_{m-1}(k)]^T$，因此$\mathbf{y} \in \Re^{(m-1)\times 1}$。接下来从式（9-60）计算

$$y_i(k) = \boldsymbol{w}_i^T(k)\boldsymbol{x}(k) - \boldsymbol{c}_i(k)\boldsymbol{y}(k) \tag{9-66}$$

再根据式（9-62）计算前馈连接权值的更新，也就是，

$$w_i(k+1) = \boldsymbol{w}_i(k) + \mu[y_i(k)\boldsymbol{x}(k) - y_i^2(k)\boldsymbol{w}_i(k)] \tag{9-67}$$

最后，根据式（9-61）计算侧向连接权值的更新，也就是，

$$\boldsymbol{c}_i(k+1) = \boldsymbol{c}_i(k) + \mu[y_i(k)\boldsymbol{y}^T(k) - y_i^2(k)\boldsymbol{c}_i(k)] \tag{9-68}$$

**步骤5**　以1为步长增加$k$值，也就是，$k \leftarrow k + 1$[下一输入使用$\boldsymbol{x}(k)$]。

**步骤6**　如果$\boldsymbol{W}$中的权值收敛，停止；否则，回到步骤3。

对于上面提到的并行APEX学习算法，协方差矩阵$\boldsymbol{C}_x$的主特征向量按照特征值递减的顺序求取。同样，如果全部$N$个零均值输入向量已经测量并且保存，也就是，$\boldsymbol{X} = \{\boldsymbol{x}(k)\} \in \Re^{n\times N}$ $(k = 1, 2, \cdots, N)$，以上提及的算法可以用于离线情况。也就是，每个训练回合将包含全部$N$ 413
个样本$[\boldsymbol{x}(k), k = 1, 2, \cdots, N]$，对于总共$M$个训练回合，这一过程可以从步骤2～步骤6重复，直到$\boldsymbol{W}$中的权值收敛（或者全部$NM$个重复出现的$\boldsymbol{x}$个输入向量）。对于离线训练过程，在第一训练回合之后，在步骤2，$\boldsymbol{W}$以及$\boldsymbol{C}$设置为前面训练回合的值。

已经提出的各种PCA神经网络可以看作正交特征的检测器，这是统计模式识别的一个基本问题。这一过程与哺乳动物的大脑是相似的，可以编码包含在输入模式中的大量相互独立的数据。例如，当人类看到一幅复杂的画面时，数据空间转换为特征空间。但是，转换设计成便于用简化数目的有效特征来表示数据集，这些特征中包含了大部分的数据固有的信息内容，即维数缩减，PCA可以完成[40]得足够精确。PCA可能是在多元分析中最早且最知名的技术。它首先由Pearson[98]提出的，他利用PCA在生物设置中重新计算线性回归分析[4]。

**例9.2**　用PCA解决图像编码问题[21, 40, 99-104]。用于图像编码的技术相对简单，与

Sanger[21]所使用的方法相似，同时也在Haykin[40]中提过。图9-5是一般图像分解成8×8的非重叠的区块布局图。本例中使用的数字化图像由240×352的8位像素组成。图中的每一个像素都是介于组成256级灰度[0, 255]中的一个整数值，其中255代表白色，0代表黑色，如图9-5所描述，图像中1320个8×8的区块中的每一个，即$\Omega_j$（$i=1, 2, \cdots, 30$以及$j=1, 2, \cdots, 44$），由下式

414

$$\boldsymbol{x}_q = \text{vec}(\Omega_0^T) = [(\omega_{11}, \omega_{12}, \cdots, \omega_{18})(\omega_{21}, \omega_{22}, \cdots, \omega_{28})\cdots(\omega_{81}, \omega_{82}, \cdots, \omega_{88})]^T \tag{9-69}$$

生成训练向量（$q=1, 2, \cdots, 1\,320$），因此，1320个$\boldsymbol{x}_q$向量中的每一个都是64×1维的（见A.2.17节）。协方差矩阵$\boldsymbol{C}_x$的近似值可以根据式（9-19）来计算，也就是，

$$\boldsymbol{C}_x = E[\boldsymbol{x}\boldsymbol{x}^T] \approx \frac{1}{N}\sum_{q=1}^{N}\boldsymbol{x}_q\boldsymbol{x}_q^T \quad \text{对于} N=1\,320 \tag{9-70}$$

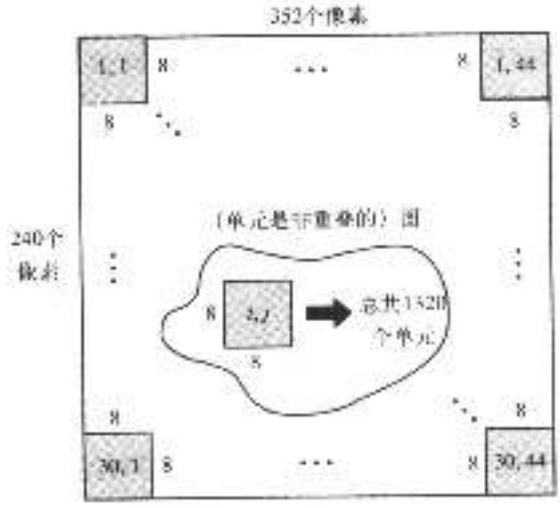



图9-5　非重叠的图像编码处理，其中每8位像素对应一个256等级灰度范围

两种用于提取$\boldsymbol{C}_x$的主特征向量的PCA方法如下：直接特征值分解（EVD）方法（参照7.6节以及9.2节）和神经网络方法，也就是使用Oja的对称子空间学习算法。无论使用什么方法，一旦提取了$\boldsymbol{C}_x$的主特征向量，这些向量集就可以用于图像单元的编码。也就是，可使用16∶1的编码率来观察对图像的编码效果。对于一个16∶1编码率，每一图像单元只有64个主特征向量中的前四个用于估计每一单元中的图像特征。因此，转换矩阵如同式（9-3）中的$\boldsymbol{W}=[\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m]^T$，其中$m=4$。一个单元图像特征的估计是根据式（9-18）来确定的，也就是，

$$\hat{\boldsymbol{x}}_q = \boldsymbol{W}^T\boldsymbol{W}\boldsymbol{x}_q \tag{9-71}$$

其中$q=1, 2, \cdots, 1320$，也就是图像的每一个单元。在图像编码步骤完成以后，图像必须重构。这是通过简单的执行“逆向量”操作来实现的，也就是式（9-69）中过程的逆操作。当对每一个连续的单元完成逆操作之后，用近似来重建图像的每一单元，也就是，$\hat{\boldsymbol{x}}_q \Rightarrow \hat{\Omega}_{ij}$。

图9-6a所示图像编码之前原始的老虎的图像，图9-6b中则是在每一个图像单元使用了16∶1（即，使用EVD方法的前四个主特征向量）的编码过程之后，然后如上所述重建的老虎图像。对比图9-6中的两幅图像，对应于图9-6a中的原始图像，可以发现图9-6b中老虎的编码图像是可辨识的。

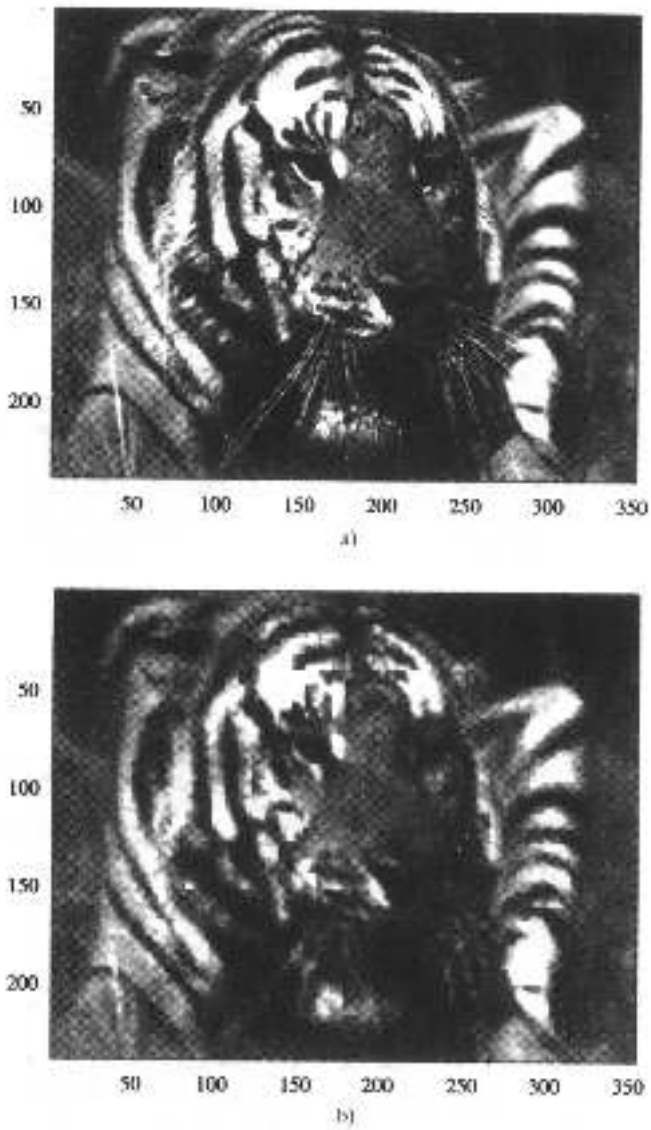



图9-6 a) 用于编码处理的240×352的8位像素的原始老虎图像，b) 使用16∶1的编码率编码a) 中的图像。即，EVD方法中只有4个主特征向量保留用于编码

图9-7描述的是前四个用于编码图9-6b中的老虎图像的8×8的掩码（主特征向量）。使用Sanger的GHA方法也可以获得相同的结果，其中使用64个输入，4个神经元单层神经网络（如图9-3所示）并使用式（9-53）中的学习规则进行训练。作为以上描述结果的对比，使用相同的单层神经网络结构但使用Karhunen-Oja的对称子空间学习规则进行训练。对于老虎图片的编码结果如图9-8所示。对比使用EVD方法（或Sanger GHA）的图9-6b和使用Karhunen-Oja的对称子空间学习算法编码相同图的图9-8，我们可以发现两幅编码图片具有相似的质量。图9-9描述了使用Karhunen-Oja的对称子空间学习规则编码图9-6b中老虎图像的前四个8×8的掩码。尽管在图9-7和图9-9中掩码有很大的不同，但在图9-6b和图9-8中显示的编码图像非常相似。即使Karhunen-Oja的对称子空间学习规则并没有推导出协方差矩阵$\boldsymbol{C}_x$的实际主特征向量，被

415 ~ 417

提取的突触网络权值还是可以与主特征向量张成相同子空间的主特征向量线性集合。

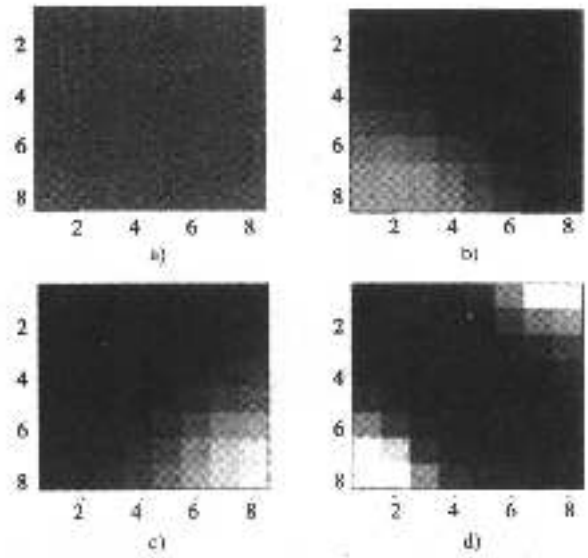



图9-7　a)–d) 利用EVD方法对图9-6a中的老虎图像提取的四个8×8掩码。这些是式(9-70)所给出的协方差矩阵的前4个主特征向量

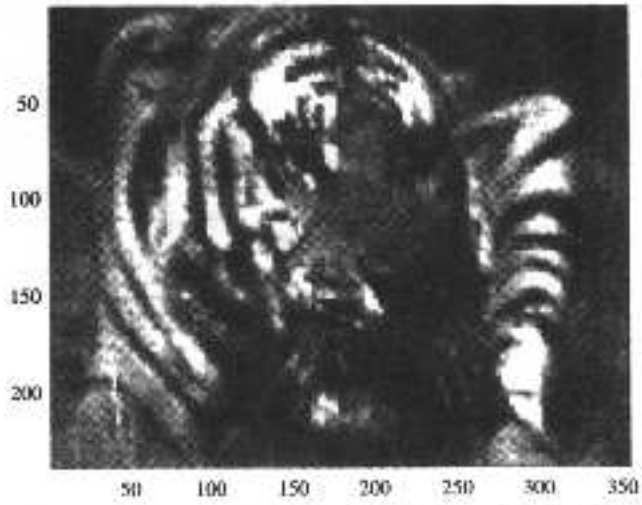



图9-8　图9-6a（240×352的8位像素）中的图像使用16∶1的编码率编码，即在使用Karhunen-Oja的对称子空间学习算法中只有4个主特征向量保留用于编码

为了说明使用Oja的对称子空间学习规则训练的神经网络的泛化能力，将图9-9中描述的老虎掩码用于编码原神经网络中未见过的不同的图像，结果如图9-10所描述，其中图9-10a是原始水果图像而图9-10b是图9-10a中的水果图像使用Karhunen-Oja对称子空间学习规则按16∶1的编码率（见图9-9）的老虎掩码编码的结果。显而易见，老虎与水果图像是统计相似的，因为相对较好的编码性能，在图9-10b中是很明显的，当图9-9中的老虎掩码应用于水果图像的编码[如图9-10a所示]，所产生的编码水果图像如图9-10b所示。

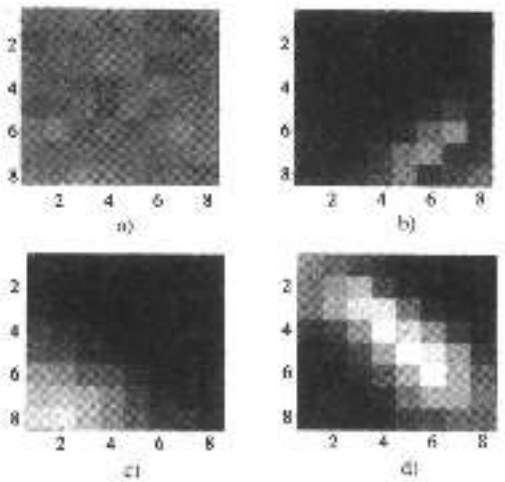



图9-9 a)–d) 图9-6a所描述的老虎图像使用Karhunen-Oja的对称子空间学习算法训练的神经网络学习的四个8×8的编码

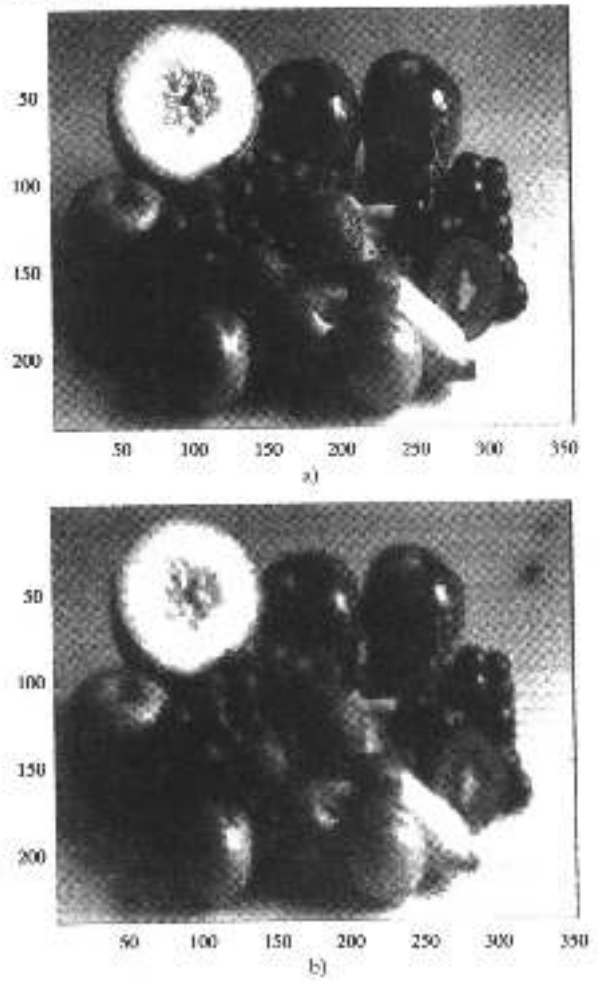



图9-10 a)用于编码处理的拥有240×352个8位像素的原始水果图像。b) 图a)中的图像是利用Karhunen-Oja对称子空间学习规则的神经网络学习的（图9-9）老虎掩码来进行的编码率16∶1编码的结果

### 9.3.6 非线性主成分分析（NLPCA）和鲁棒PCA

在前面部分提到的标准线性PCA起源于对于几个不同的信号表示问题的最佳解决方案，总结如下：

1. 在正交约束条件下，即$\boldsymbol{W}\boldsymbol{W}^T = \boldsymbol{I} \in \Re^{m \times m}$时线性（变换后）方差$E\{[\boldsymbol{w}_i^T\boldsymbol{x}]^2\}$或者线性网络输出的最大化。

2. 均方表达误差$E\{\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2\}$的最大化，其中输入数据$\boldsymbol{x}$的近似值可以通过较低维的线性子空间$\hat{\boldsymbol{x}} = \boldsymbol{W}^T\boldsymbol{W}\boldsymbol{x}$给出。

3. 对于在正交化变换$\boldsymbol{W}\boldsymbol{W}^T = \boldsymbol{I}$之后的不同神经元，对线性网络的输出$y_i = \boldsymbol{w}_i^T\boldsymbol{x}$去相关性问题，其中$i = 1, 2, \cdots, m$。

4. 表示熵最小化问题。

418
~
419

与相关的假设以及条件约束相关联的最佳PCA解的来源可以在[41]中找到。Karhunen和Joutsensalo[41]证明线性PCA作为这些不同信息表示问题的最佳解，事实上是因为这个解基于输入仅为二阶统计量。线性PCA神经网络以及相关的学习算法在一些条件[41]下，由于以下限制而失去了吸引力：

1. 标准PCA网络有能力实现只有线性输入/输出的映射。

2. 可以通过知名的数值方法来有效地计算标准PCA中所需的特征向量。梯度类型的神经PCA学习算法典型地收敛速度较慢。对于大型问题，为了达到较好的精确度需要额外数量的迭代。

3. 主成分使用数据协方差或者相关性唯一地定义。这些输入的二阶统计量完全可以通过高斯数据以及稳态线性处理操作来表示。

4. 线性PCA网络不能经常从子信号的线性混合中分离出独立的子信号。

如果PCA类型的网络包含非线性，那么这一设置将变得更加适合于神经实现，(1) 输入/输出映射一般情况下是非线性的，这正是我们一般情况下使用神经网络的一个主要判断依据，因此数据的处理也更加有效；(2) 神经算法相对于经典的PCA方法更加具有竞争力；(3) 使用非线性隐含的也将高阶统计引入了计算当中；(4) 标准PCA网络的输出一般是不相关的但也不是独立的，后一点在某些情况下会更加期望；PCA网络附加非线性因素会增加输出的独立性，因此，输入信号某些时候有必要同它们的混合相分离（Karhunen以及Joutsensalo[41]）。线性PCA的推广可以使标准神经PCA算法产生鲁棒性以及非线性扩展。通过考虑最优化问题的泛化而推导出的结果学习算法，导致标准PCA可以分成两类[41]：*鲁棒PCA算法*[70-73, 105]以及*非线性PCA（NLPCA）算法*[42-47, 52, 60, 61, 63, 106, 107]。

定义鲁棒PCA，以便于优化的标准增长速度少于二次方（即，非二次方最优化标准），约束条件与标准PCA解是一致的。非二次方最优化标准将非线性引入梯度算法。这使得结果对有色噪声和数据出格点更加具有鲁棒性[7, 12]。因此，源于非二次方最优化标准的梯度算法比其线性相对的部分（即，二次方性能标准），变得更加具有鲁棒性。更特殊地，二次方性能标准十分注重来源于强烈的噪声元素或者出格点的大误差值。相反地，非二次方标准在强噪声和出格点情况下会是一个更好的选择，因为它比二次方函数增加得更加缓慢，且受较大误差干扰的影响更小。同样，非二次方最优化标准允许考虑输入相关联的较高阶的统计[43, 107-110]。如果只使用二阶统计，一些重要的问题就不能够适当解决。典型地，神经网络的权值向量，即展开的基向量，要求相互正交。鲁棒PCA问题的设置一般会导致稍微非线性的

420

学习算法，其中非线性特性在网络结构中只会出现在已经选择的位置上[41]。也就是，至少鲁棒PCA网络的一些神经元输出将仍然拥有线性反应$y_i = \boldsymbol{w}_i^T\boldsymbol{x}$, $i = 1, 2, \cdots, m$，其中$\boldsymbol{w}_i$是第$i$个神

经元的权值向量。

相反，在非线性PCA结构中，网络神经元的所有输出都包含非线性的特性[即$g(y_i) = g(\boldsymbol{w}_i^T \boldsymbol{x})$，其中$g(\cdot)$是已选择的非线性特性]，最优化标准仍然可以公式化成一个二次方函数，如线性PCA中的例子。较之上面描述的前两种，更加普遍的第三种非线性PCA神经网络是基于非二次方最优化标准的，正如鲁棒PCA算法，且非线性特性在所有的输出神经元中都出现，这正如非线性PCA算法中的情况。在Karhunen和Joutsensalo[41]中，在方差最大化以及均方误差最小化的问题中，可能最重要的单个结果就是知名的Sanger[21]的广义Hebb算法（参照9.3.3节）的推导，还有它的鲁棒性以及非线性的副本。提取数据的统计独立特征的相关方法可以在Parra et al.[111]处找到，这是非线性的独立成分分析[112-114]（参照10.8节）以及偶对的非线性独立成分分析[115]。

NLPCA网络的一种对称结构如图9-11所示，其中反馈连接（以虚线标出）在网络经过训练以后就被移除了。因此，网络此时是严格的前馈结构。另一个NLPCA结构是图9-12所示的分层结构。同对称结构一样，反馈连接也在网络训练后被移除。包含非线性特性的PCA网络存在不足如下：(1) 许多学习规则的数学分析通常比线性PCA网络固有的困难更多。因此，带 421
有非线性特性的网络特征通常很难理解。(2) 非线性学习算法更加复杂，且可能偶尔导致收敛到一个局部最小值。(3) 给网络增加非线性特性并不一定能避免一些问题。因此，不能随意地将非线性特性引入PCA网络。其他NLPCA的结构，例如，递归推广和伴有侧向连接[45]的非线性PCA结构。

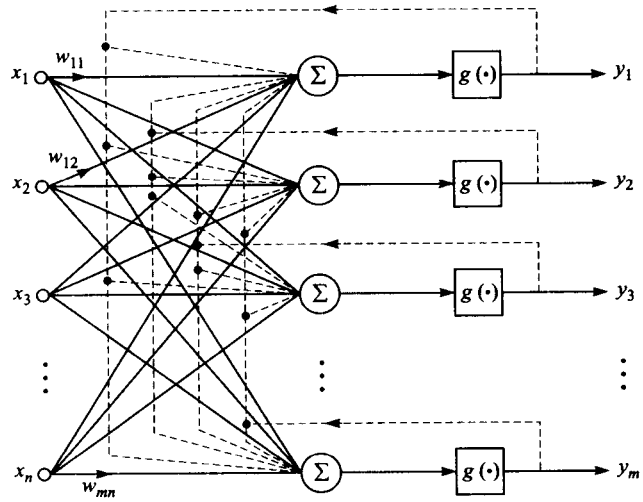



图9-11　对称非线性PCA神经网络结构。其中反馈连接（虚线）只在训练阶段是必要的

如前所述，鲁棒PCA神经网络将非线性引入到了相关的梯度算法中，后者拥有“鲁棒化”关于冲击和有色噪声以及数据出格点的神经网络的作用。也就是，公式化非二次方最优化标准倾向于通过对输入数据向量较少的作用来抵制带来相对较大的误差的出格点的影响。在1992[58]线性PCA神经网络的非线性扩张已进行了探索性地讨论，但是直到1993年才在数学基础上实际提出了鲁棒PCA类型的神经网络[42, 70-73]。1993年以后，许多关于鲁棒PCA的研究结果 [41, 43-48, 74]得到报道。这三种类型的鲁棒PCA神经子空间学习算法要归功于Karhunen和Joutsensalo[41, 43, 44, 48, 70, 71, 74, 75]。

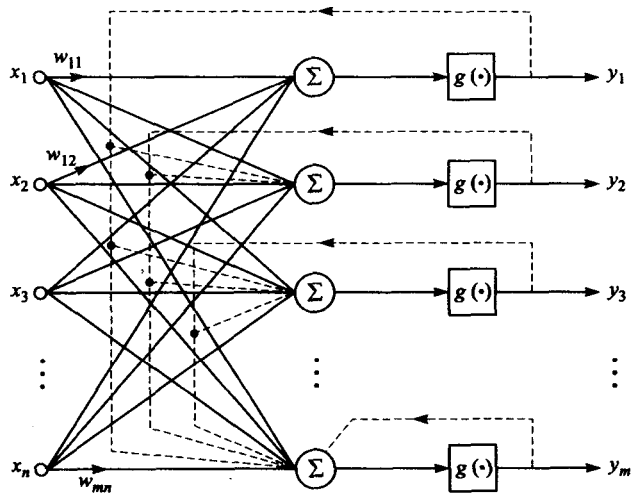



图9-12　分层的非线性PCA神经网络结构。其中反馈连接（虚线）只在训练阶段是必要的

为了使提出的每种情况直接符合Karhunen和Joutsensalo的要求，现在把突触权值矩阵定义成$\boldsymbol{W}\in\Re^{n\times m}$，同时零均值随机向量的输入为$\boldsymbol{x}\in\Re^{n\times 1}$，输出为$\boldsymbol{y}\in\Re^{m\times 1}$。三个鲁棒PCA子空间离
422 散时间学习算法可以由方差最大化问题或者最小化表示误差标准推导而出，并可以写成如下形式

$$\boldsymbol{W}(k+1)=\boldsymbol{W}(k)+\mu(k)[\boldsymbol{I}-\boldsymbol{W}(k)\boldsymbol{W}^T(k)]\boldsymbol{x}(k)g[\boldsymbol{x}^T(k)\boldsymbol{W}(k)] \tag{9-72}$$

$$\boldsymbol{W}(k+1)=\boldsymbol{W}(k)+\mu(k)\{\boldsymbol{x}(k)g[\boldsymbol{e}^T(k)]\boldsymbol{W}(k)+g[\boldsymbol{e}(k)]\boldsymbol{x}^T(k)\boldsymbol{W}(k)\} \tag{9-73}$$

其中

$$\boldsymbol{e}(k)=\boldsymbol{x}(k)-\hat{\boldsymbol{x}}(k)=\boldsymbol{x}(k)-\boldsymbol{W}(k)\boldsymbol{W}^T(k)\boldsymbol{x}(k)=[\boldsymbol{I}-\boldsymbol{W}(k)\boldsymbol{W}^T(k)]\boldsymbol{x}(k) \tag{9-74}$$

以及

$$\boldsymbol{W}(k+1)=\boldsymbol{W}(k)+\mu(k)g[\boldsymbol{e}(k)]\boldsymbol{x}^T(k)\boldsymbol{W}(k) \tag{9-75}$$

在式（9-72）、式（9-73）以及式（9-75），$\mu(k)>0$是控制学习率的增益参数。学习参数可以根据式（9-28）和式（9-48）自适应改变每一个步骤。这些鲁棒PCA学习算法可以由两个不同的性能标准推导。例如，首先式（9-72）中的学习规则可以通过与方差最大化问题相关的性能标准来推导，其中方差最大化问题可以修改成如下形式（其中对于每一个$w_i$, $i=1,2,\cdots,m$）

$$J_1(\boldsymbol{w}_i)=\mathrm{E}\{f[\boldsymbol{x}^T\boldsymbol{w}_i]\,|\,\boldsymbol{w}_i\}+\frac{1}{2}\sum_{j=1}^{m}\lambda_{ij}(\boldsymbol{w}_i^T\boldsymbol{w}_j-\delta_{ij}) \tag{9-76}$$

它将被最大化。只考虑瞬时输入，从而，期望运算符从式（9-76）中撤销，计算梯度。在式（9-76）中，$\delta_{ij}$是克罗内克Δ，右边的第一个项（即，条件期望）要被最大化（此时神经元权值向量是正交的），同时求和要求拉格朗日乘子$\lambda_{ij}=\lambda_{ji}$满足必要的正交的约束（即，$\boldsymbol{w}_i^T\boldsymbol{w}_j=\delta_{ij}$）。假定函数$f(t)$是偶函数，非负，可连续求导（几乎是处处），且$f(t)\leqslant t^2/2$，其中对于较大的$|t|$严格不等。此外，唯一的最小值发生在$t=0$时，如果$|t_2|>|t_1|$，则$f(t_2)\geqslant f(t_1)$。在式（9-76）中的标准比二次的增长长得慢，至少对于较大值是这样，同时标准可以以向量矩阵的形式写成更加简洁如下

$$J_1(\boldsymbol{W})=\mathbf{1}^T\mathrm{E}\{f[\boldsymbol{W}^T\boldsymbol{x}]\,|\,\boldsymbol{W}\}+\frac{1}{2}\mathrm{trace}[\Lambda(\boldsymbol{W}^T\boldsymbol{W}-\boldsymbol{I})] \tag{9-77}$$

其中全1向量$\mathbf{1}^T = [1, \cdots, 1]$拥有合适的维数，且矩阵$\Lambda$的元素是$\lambda_{ij}$。式（9-72）中的鲁棒PCA学习规则可以通过式（9-77）推导得到，且$g(t)$是$f(t)$的导数，即$\mathrm{d}f(t)/\mathrm{d}t$。注意，如果$g(t) = t$，这是导致式（9-45）（参照9.3.2节中）Karhunen-Oja的线性对称子空间学习规则的一个特例。然而，根据式（9-45）中的突触权值矩阵定义的方式做出合适的解释，也就是$\boldsymbol{W} \in \Re^{m \times n}$。因此，式（9-77）的标准是方差最大化问题的一个推广，导出式（9-72）的细节可在[41, 43, 60]中找到。

式（9-73）和式（9-75）中的鲁棒PCA学习规则可以由表达误差[41, 70, 71]的推广而推得，即，求最小化的相关性能标准以向量矩阵形式表示如下 423

$$J_2(\boldsymbol{W}) = \mathbf{1}^T \mathrm{E}\{f[\boldsymbol{e}]\} = \mathbf{1}^T \mathrm{E}\{f[\boldsymbol{x} - \hat{\boldsymbol{x}}]\} = \mathbf{1}^T \mathrm{E}\{f[\boldsymbol{x} - \boldsymbol{W}^T \boldsymbol{W}\boldsymbol{x}] | \boldsymbol{W}\} \tag{9-78}$$

函数$f(t)$必须满足前面给出的类似条件。由式（9-78）中广义表达误差标准，导出式（9-73）和式（9-75）的细节可在[43, 75]中找到。注意，在式（9-73）中，对于每一个权值向量，更新表达式中右边的第一项，也就是$\boldsymbol{x}(k)\boldsymbol{g}[\boldsymbol{e}^T(k)]\boldsymbol{W}(k)$只与数据输入向量成比例。因此，它可以消去，从而直接导出式（9-75）中的简化学习规则[70]。同样，在式(9-73)中的鲁棒PCA学习规则中，如果$g(t) = t$，此时这一学习规则可以独立地最小化均方误差，这是Xu[76]和Russo[77]早期研发的标准PCA子空间学习规则的特殊情况。

非线性的选择是任意的（考虑它是偶函数，非负，连续可微）。但是，我们将只提及Karhunen和Joutsensalo[43]用到的三种更常见的。这些是二次函数$f_1(t) = t^2/2$，线性标准$f_2(t) = |t|$，以及标准$f_3(t) = \ln[1/2(\mathrm{e}^t + \mathrm{e}^{-t})]$。图9-13分别给出了这三种函数的坐标图。每一种函数对于$t$的相应导数可以以如$g_1(t) = t$，$g_2(t) = \mathrm{sgn}(t)$相近的形式分别计算，[其中$\mathrm{sgn}(t)$表示正负号函数，或者$t$的符号]，以及$g_3(t) = \tanh t$。每一非线性的导数如图9-14所示。

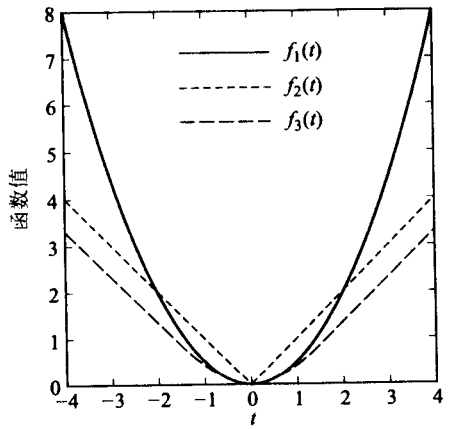



图9-13　用于式（9-76）和式（9-77）的性能标准的典型非线性（$f_i(t)$），其中$f_1(t) = t^2/2$，$f_2(t) = |t|$以及$f_3(t) = \ln[1/2(e^t + e^{-t})]$

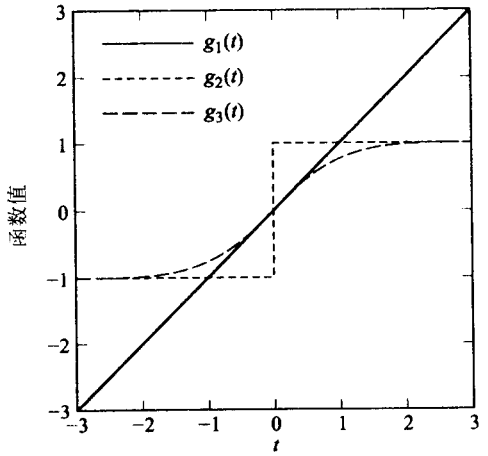



图9-14　图9-13中的三个非线性函数的导数（$g_i(t)$），其中$g_1(t) = t$，$g_2(t) = \mathrm{sgn}(t)$，以及$g_3(t) = \tanh t$

**例9.3**　对比式（9-72）中的鲁棒PCA子空间学习规则的性能和式（9-45）中的Karhunen-Oja对称子空间学习规则的性能。500个三维零均值随机高斯向量产生在$x$，$y$，$z$方向的方差分别为$\sigma_x^2 = 5$，$\sigma_y^2 = 3$，$\sigma_z^2 = 0.2$。理论上，相关的协方差矩阵的对角线上的特征值与三个分量的方差是相等的。使用Karhunen-Oja的对称子空间学习规则和鲁棒PCA子空间学习规则时，首先遇到的问题是计算前面两个张成$xy$平面中子空间的主特征向量。然后，通过使用参考坐标系中 424

的$z$轴，计算共面的$x$，$y$向量（被两个不同的神经网络学习）的法分量的角度的余弦。这个角度是500个三维向量中的300次独立试验中的平均值；即，500个三维向量是每次实验中随机产生的。此外，对于每一次试验500个随机向量要求输入到两个神经网络两次，也就是，每次实验两个训练回合。使用Karhunen-Oja的对称子空间学习规则，这个角度的余弦是$\cos(\angle_{KO}) = 0.9993$，而使用鲁棒PCA子空间学习规则时，这个角度的余弦是$\cos(\angle_{ROB}) = 0.9743$。对于这两个神经网络，都使用了式（9-48）中提出的自适应学习参数方法，其中遗忘因子赋值为0.95。

但是，这并没有说明鲁棒PCA子空间学习规则的鲁棒性。因此，进行另一个相似的试验，其中产生的随机向量有0.9概率来自相同的高斯分布，而0.1概率来自于间隔 $[-10, 10]$之间均匀分布并被冲击噪声所破坏的高斯随机向量。虽然数据的方差是根据$\sigma_x^2 = 5 + (20^2)(0.1)/12$，$\sigma_y^2 = 3 + (20^2)(0.1)/12$，$\sigma_z^2 = 0.2 + (20^2)(0.1)/12$来改变的，但是理论上，主子空间是保持不变的，即$xy$平面。在训练两个神经网络之后（使用这个问题的第一部分中的相同方法，不计冲击噪声），在两个结论中都要求计算法线（为了提取子空间而计算的）和参考坐标框架中的$z$轴之间的角度的余弦。Karhunen-Oja对称子空间神经网络的结果推导出$\cos(\angle_{KO}) = 0.9317$，而使用鲁棒PCA子空间学习规则时，这个角度的余弦是$\cos(\angle_{ROB}) = 0.9865$。因此，对于冲击噪声
425 破坏数据的情况，鲁棒PCA子空间神经网络可以推导出比Karhunen-Oja对称子空间神经网络更好的结果。图9-15描述了当有冲击噪声破坏数据时，鲁棒PCA子空间神经网络的结果。

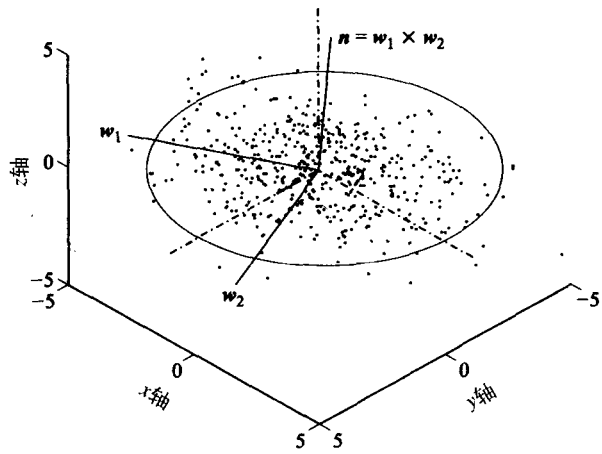



图9-15　当有10%来自于间隔 $[-10, 10]$之间均匀分布的高斯随机数据被冲击噪声所破坏时，鲁棒PCA子空间神经网络的结果。向量$w_1$与$w_2$是由神经网络学习的，计算的角度余弦给出为$\cos(\angle_{ROB}) = 0.9865$，与之对比的Karhunen-Oja对称子空间神经网络的结果中，$\cos(\angle_{KO}) = 0.9317$。“点划线”代表参考坐标系

## 9.4　主成分回归

很多种数据回归方法组成了化学统计学[7, 8, 11, 14]领域。化学统计学方法广泛地应用于分析化学中的光谱数据[116, 117]的定量分析。这里介绍两种多元回归方法，基于PCA的主成分回归（PCR）[4, 6-8]和部分最小二乘回归（PLSR）[7-14]，将在下一节中提到。基本问题是校准模型以适应在特殊系统中采集的经验数据。目的是利用PCR或者PLSR开发双线性模型[7]，通过将数据的特征空间压缩成只保持数据中的基本信息的简化的数据空间，这样校准模型关于它的预示性能得到最优化。也就是说，PCR和PLSR都认为是基于方法[7, 8, 10-13]的因子分析（或者秩化简），它们可以通过训练过程推导出一种校正模型，然后，其可用于在假定

模型测试输入时预测某些量。近来，Ham以及Kostanic[9]已经将PLSR十分成功地应用于工程信号的处理问题。PCR已经同样应用于频率估计的超分辨率算法当中，即，多元信号分类（MUSIC）和最小范数方法[2]。 426

经典的最小二乘（CLS）

经典的最小二乘（CLS）[4, 6, 7, 12]在PCR之前简单地讨论过，下面将在这两种方法之间作出对比，并在下一节中同样将CLS与PLSR进行对比。假定存在线性关系，即向量$\boldsymbol{h}\in\Re^{n\times 1}$，在以如下矩阵形式给出的一组自变量（独立变量块）之间，其中$\boldsymbol{X}(k)\in\Re^{n\times m}$

$$\boldsymbol{X}(k)=\begin{bmatrix} x_{11}(k) & x_{12}(k) & \cdots & x_{1m}(k) \\ x_{21}(k) & x_{22}(k) & \cdots & x_{2m}(k) \\ \cdots & \cdots & \cdots & \cdots \\ x_{n1}(k) & x_{n2}(k) & \cdots & x_{nm}(k) \end{bmatrix} \tag{9-79}$$

$k$是时间索引，同时假定如$\boldsymbol{y}(k)\in\Re^{m\times 1}$向量形式的依赖变量（依赖变量块），由此，我们可以写出

$$\boldsymbol{y}(k)=\boldsymbol{X}^T(k)\boldsymbol{h} \tag{9-80}$$

矩阵$\boldsymbol{X}(k)$ 中的每一个列向量是由宽平稳过程的时间序列组成，同时$\boldsymbol{X}(k)$ 中的列向量不一定要线性独立。但是，式（9-79）中的$\boldsymbol{X}(k)$ 的列向量假定受到高斯白噪声$\boldsymbol{N}(k)\in\Re^{n\times m}$的影响；即，$\boldsymbol{N}(k)$ 的每一列是零均值白序列，同时$\boldsymbol{N}(k)$的协方差矩阵可以使用式（9-19）中的表达式通过$\boldsymbol{C}_N\approx(1/m)\boldsymbol{N}(k)\boldsymbol{N}^T(k)\in\Re^{n\times n}$来近似。因此，式（9-79）中的$\boldsymbol{X}(k)$ 的测量可以写成如下

$$\boldsymbol{Z}(k)=\boldsymbol{X}(k)+\boldsymbol{N}(k) \tag{9-81}$$

其中$\boldsymbol{Z}(k)\in\Re^{n\times m}$。$\boldsymbol{X}(k)$ 列中的信息是未知的。但是，$\boldsymbol{Z}(k)$ 包含$\boldsymbol{X}(k)$ 的噪声测量。因变量$\boldsymbol{y}(k)$的估计利用$\boldsymbol{Z}(k)$ 可以写成

$$\hat{\boldsymbol{y}}(k)=\boldsymbol{Z}^T(k)\boldsymbol{g} \tag{9-82}$$

其中$\boldsymbol{g}\in\Re^{n\times 1}$，$\hat{\boldsymbol{y}}(k)\in\Re^{m\times 1}$。因此，目标是找到式（9-82）中的线性关系$\boldsymbol{g}$，使得测量的变量$\boldsymbol{Z}(k)$ 和因变量$\hat{\boldsymbol{y}}(k)$ 的估计相关联。为了实现这一点，可以采用经典的最小二乘方法，通过首先定义如下形式的误差变量

$$\boldsymbol{e}(k)=\boldsymbol{y}(k)-\hat{\boldsymbol{y}}(k) \tag{9-83}$$

其中$\boldsymbol{e}(k)\in\Re^{m\times 1}$。使用式（9-83），二次方性能测量可以定义为

$$\boldsymbol{J}_g=\frac{1}{m}\boldsymbol{e}^T(k)\boldsymbol{W}\boldsymbol{e}(k) \tag{9-84}$$

它将关于$\boldsymbol{g}$最小化。但是，式（9-84）中的正定对称权值矩阵$\boldsymbol{W}\in\Re^{m\times m}$，可以看作$m\times m$的单位矩阵，即$\boldsymbol{W}=\boldsymbol{I}_m$，因为所有的误差认为是平等加权的。因此，式（9-84）可以写作

$$J_g=\frac{1}{m}\boldsymbol{e}^T(k)\boldsymbol{e}(k)=\frac{1}{m}[\boldsymbol{y}(k)-\hat{\boldsymbol{y}}(k)]^T[\boldsymbol{y}(k)-\hat{\boldsymbol{y}}(k)] \tag{9-85}$$ 427

将式（9-80）、式（9-81）以及式（9-82）代入式（9-85）得到

$$\begin{aligned} J_g&=\frac{1}{m}[\boldsymbol{y}(k)-\hat{\boldsymbol{y}}(k)]^T[\boldsymbol{y}(k)-\hat{\boldsymbol{y}}(k)] \\ &=\boldsymbol{h}^T\left[\frac{1}{m}\boldsymbol{X}(k)\boldsymbol{X}^T(k)\right]\boldsymbol{h}-2\boldsymbol{g}^T\left[\frac{1}{m}\boldsymbol{X}(k)\boldsymbol{X}^T(k)\right]\boldsymbol{h}+\boldsymbol{g}^T\left[\frac{1}{m}\boldsymbol{X}(k)\boldsymbol{X}^T(k)\right]\boldsymbol{g}+\boldsymbol{g}^T\left[\frac{1}{m}\boldsymbol{N}(k)\boldsymbol{N}(k)^T\right]\boldsymbol{g} \\ &=\boldsymbol{h}^T\boldsymbol{C}_X\boldsymbol{h}-2\boldsymbol{g}^T\boldsymbol{C}_X\boldsymbol{h}+\boldsymbol{g}^T\boldsymbol{C}_X\boldsymbol{g}+\boldsymbol{g}^T\boldsymbol{C}_N\boldsymbol{g} \end{aligned} \tag{9-86}$$

其中$\boldsymbol{C}_x \approx (1/m)\boldsymbol{X}(k)\boldsymbol{X}^T(k) \in \Re^{n\times n}$是对$\boldsymbol{X}(k)$的协方差矩阵的近似，并假定$\boldsymbol{N}(k)$和$\boldsymbol{X}(k)$是不相关的，即，

$$\mathrm{E}[\boldsymbol{n}(k)\boldsymbol{x}^T(k)] = \mathrm{E}[\boldsymbol{x}(k)\boldsymbol{n}^T(k)] = \boldsymbol{0} \tag{9-87}$$

对式（9-86）求取关于$\boldsymbol{g}$的偏导数，并令结果等于0，如下

$$\begin{aligned}\frac{\partial \boldsymbol{J}_g}{\partial \boldsymbol{g}} &= \frac{\partial}{\partial \boldsymbol{g}}(\boldsymbol{h}^T\boldsymbol{C}_X\boldsymbol{h} - 2\boldsymbol{g}^T\boldsymbol{C}_X\boldsymbol{h} + \boldsymbol{g}^T\boldsymbol{C}_X\boldsymbol{g} + \boldsymbol{g}^T\boldsymbol{C}_{Ng}) \\ &= -2\boldsymbol{C}_X\boldsymbol{h} + 2\boldsymbol{C}_X\boldsymbol{g} + 2\boldsymbol{C}_N\boldsymbol{g} = 0\end{aligned} \tag{9-88}$$

从式（9-88）中求解出$\boldsymbol{g}$如下

$$\boldsymbol{g} = (\boldsymbol{C}_X + \boldsymbol{C}_N)^{-1}\boldsymbol{C}_X\boldsymbol{h} \tag{9-89}$$

为了从式（9-89）中确定$\boldsymbol{g}$，需要了解关于未受损的测量矩阵$\boldsymbol{C}_X$和测量噪声$\boldsymbol{C}_N$的协方差矩阵，还有介于未受损的测量$\boldsymbol{X}(k)$和因变量$\boldsymbol{y}(k)$之间的线性关系$\boldsymbol{h}$。然而，这些量通常是未知的，但是用CLS方法可以推导出$\boldsymbol{g}$，仅使用包含在受损测量$\boldsymbol{Z}(k)$和如下形式的与式(9-89)直接相联系的因变量$\boldsymbol{y}(k)$[4, 118]中的信息。

$$\boldsymbol{g} = [\boldsymbol{Z}(k)\boldsymbol{Z}^T(k)]^{-1}\boldsymbol{Z}(k)\boldsymbol{y}(k) \tag{9-90}$$

这可以首先利用式（9-81）以及假设$\boldsymbol{N}(k)$和$\boldsymbol{X}(k)$之间是不相关的，来写出协方差矩阵$\boldsymbol{Z}(k)$的近似如下

$$\boldsymbol{C}_Z \approx \frac{1}{m}\boldsymbol{Z}(k)\boldsymbol{Z}^T(k) = \frac{1}{m}\boldsymbol{X}(k)\boldsymbol{X}^T(k) + \frac{1}{m}\boldsymbol{N}(k)\boldsymbol{N}^T(k) = \boldsymbol{C}_X + \boldsymbol{C}_N \tag{9-91}$$

式（9-91）的逆给出了式（9-89）中表达式的第一部分。接下来，使用因变量$\boldsymbol{y}(k)$随同测量数据$\boldsymbol{Z}(k)$中的信息，我们可以得到

$$\frac{1}{m}\boldsymbol{Z}(k)\boldsymbol{y}(k) = \left[\frac{1}{m}\boldsymbol{X}(k)\boldsymbol{X}^T(k)\right]\boldsymbol{h} = \boldsymbol{C}_X\boldsymbol{h} \tag{9-92}$$

利用式（9-80）、式（9-81），再一次假设$\boldsymbol{N}(k)$和$\boldsymbol{X}(k)$是不相关的。式（9-92）中的表达式给出了式（9-89）中的第二部分。因此，从式（9-91）和式（9-92）的结果中，可以发现式（9-89）
428 和式（9-90）之间的关系。

有一点十分重要，就是实现式（9-90）中的CLS的结果，使用在受损测量$\boldsymbol{Z}(k)$和因变量$\boldsymbol{y}(k)$中所有可能的信息。但是，存在许多很不如意的情况，会导致数据[7, 118]的过适应。也就是，在展开式（9-90）中模型$\boldsymbol{g}$的过程中，模型参数可能不只基于与经验数据$\boldsymbol{Z}(k)$和$\boldsymbol{y}(k)$相关的必要的因果关系特征，而且包含数据中不希望的效果。这将导致不良预测性能的模型。这些不希望的效果可能与测量噪声或者任何其他与分析者丝毫不感兴趣的因果关系现象的额外数据特征相关。广义上讲，这些效果可以看作是噪声，因为现象的本质不会认为是先验已知的。因此，利用回归方法（该方法有能力在可以使模型性能降低的数据中，没有模糊效果的先验知识的情况下，作出严格的选择），推导出带有改良预测性能的校准模型，只需要使用经验数据的这些特征。主成分回归是一种可以通过与这种回归技术相关的因子分析能力，做出严格选择的方法。典型地，经过PCR展开的结果校正模型较之CLS模型具有更佳的预测性能。PCR的细节将在下面介绍，在下一节将提出部分最小二乘回归的细节。典型地，PLSR将推导出较之PCR[7, 10]更好的预测性能。这里性能加强的根本原因将在下一节进行解释（参照9.5节）。

主成分回归

在介绍PCR之前，我们定义数据矩阵为$\boldsymbol{A}\in\Re^{m\times n}$，如同前面$\boldsymbol{Z}(k)$的定义，其中$\boldsymbol{A}$的每一个行向量都是一个特殊的测量（即，总计$m$个测量），每个测量拥有相同数量的分量（即，每个测量中总共有$n$个分量）。对于自变量块$\boldsymbol{A}$的用法与在文献中的发现更加一致，因为$\boldsymbol{A}$特别地涉及到光谱吸收数据[119]，即使这并不是必需的情况。事实上，矩阵$\boldsymbol{A}$实际可以包含任何测量数据。因此，时间索引$k$被丢弃了。但是，如果包含在$\boldsymbol{A}$中的数据是光谱数据，那么这就暗示每一个测量由$n$个光谱频率或者波数[119]组成。另一方面，如果$\boldsymbol{A}$中的每一行是时间域的测量，那么每一行就有$n$个时间采样组成，其中$k$是时间索引。同样，因变量块，前面提及的$\boldsymbol{y}(k)$，现在指定为$\boldsymbol{c}\in\Re^{m\times 1}$。这样，再一次与文献中出现的大部分符号更加地一致，因为$\boldsymbol{c}$向量中的每一个元素通常对应一个中心值，专门为每一个用分光光度计[7]收集的样例谱而测量。然而，$\boldsymbol{c}$的分量可以是可独立确定的任意目标值，它直接与对应的自变量块中行测量相关，即$\boldsymbol{A}$矩阵。 429
我们将只讨论单分量情形[7]，即矩阵$\boldsymbol{A}$的被测数据中，只有一个感兴趣分量的情况。这就是$\boldsymbol{c}$中的目标值只形成列向量而不是矩阵的原因。同样，这一符号将随后用于下一节关于PLSR的讨论。

正如前面对于CLS的描述，这一方法的主要问题关系到保留所有包含在经验数据中以及用于开发校准模型的信息；见式（9-90）。至少还有两种问题关系到这种开发校准模型的方法：(1) 如果数据矩阵$\boldsymbol{A}$中存在共线性[7]（即，线性依赖，其中一个独立变量可以精确地或者近似地表示成与另一个变量的线性组合），那么$\boldsymbol{A}^T\boldsymbol{A}$将是病态的[96]。此外，对于没有测量噪声的情况，存在精确的重复测量，$(\boldsymbol{A}^T\boldsymbol{A})^{-1}$可能不存在。(2) 即使$\boldsymbol{A}$不是共线性的，但是，如果$n$非常的大（样本分量以及频率数），那么计算$\boldsymbol{A}^T\boldsymbol{A}$的逆需大量的计算。因此，我们十分希望拥有一种方法（或者一些方法）对于共线性不十分敏感，不需要很大量计算，同时可以在经验数据中只使用相干信息，这样校准模型并不包括不希望的效果，因此，当同CLS进行对比时，从中发现其加强的预测性能。PCR和PLSR都是这样的方法。即，它们基于因子分析（秩简化）[7]。

在PCR的情况中，我们打算开发一种校正模型，它可以预测浓度$\hat{\boldsymbol{c}}$，假设测试输入到在开发中没有用到的模型。此外，我们希望在模型开发中使用相同的数据的情况下，PCR校正模型与CLS校正模型相比，拥有更好的预测能力。因此，我们假定对于开发校正模型是可行的自变量块$\boldsymbol{A}$拥有以下形式

$$\boldsymbol{A}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \tag{9-93}$$

其中$\boldsymbol{A}$中的每一行对应于一个可能被噪声破坏的测量。与每一个测量相关的目标值（因变量块）在列向量中给出如下

$$\boldsymbol{c}=[c_1, c_2, \cdots, c_m]^T \tag{9-94}$$

如果假设在自变量块和因变量块$\{\boldsymbol{A},\boldsymbol{c}\}$之间存在线性关系，即

$$\boldsymbol{b}_f\in\Re^{n\times 1} \tag{9-95}$$

我们可以写出

$$\boldsymbol{c}=\boldsymbol{A}\boldsymbol{b}_f+\boldsymbol{e}_c \tag{9-96}$$

其中$\boldsymbol{e}_c\in\Re^{m\times 1}$是误差向量，说明了所有误差，包括测量噪声。式（9-96）的CLS解可以通过误

430 差代价函数（李雅普诺夫函数）对于$\boldsymbol{b}_f$最小化来得到，即

$$L(\boldsymbol{b}_f)=\frac{1}{2}\|\boldsymbol{e}_c\|_2^2=\frac{1}{2}\|\boldsymbol{c}-\boldsymbol{A}\boldsymbol{b}_f\|_2^2 \tag{9-97}$$

它可以导出结果

$$\hat{\boldsymbol{b}}_{f\text{-CLS}}=(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{c} \tag{9-98}$$

其中$\hat{\boldsymbol{b}}_{f\,\text{CLS}}\in\Re^{n\times 1}$是最终的CLS校正模型，假定$m\geqslant n$（即，较之每一个测量中的分量，有更多或者相等数量的测量存在）。这被认为是超定的情况[2, 96]。但是，如果$m<n$（欠定的情况[2, 96]），那么CLS校正模型可按照式$\hat{\boldsymbol{b}}_{f\,\text{CLS}}=\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1}\boldsymbol{c}$来进行计算。这一模型现在可以用于预测因变量，假定一组没有用于开发校正模型的独立数据如$\{\boldsymbol{A}_{test},\boldsymbol{c}_{test}\}$，其中$\boldsymbol{A}_{test}\in\Re^{p\times n}$以及$\boldsymbol{c}_{test}\in\Re^{p\times 1}$，即

$$\hat{\boldsymbol{c}}_{\text{testCLS}}=\boldsymbol{A}_{\text{test}}\hat{\boldsymbol{b}}_{f\,\text{CLS}} \tag{9-99}$$

但是，正如前面所提到的，存在一些与CLS方法相关的基本问题。主要问题是$\boldsymbol{A}$中经验数据的所有特征用于开发式（9-98）中的校正模型，包括可能的噪声以及其他的不希望的模糊效应。注意到，为便利起见，对于用于开发校正模型（即，训练阶段）矩阵$\boldsymbol{A}$，将下标“train”省略，这将成为暗指。这也应用于式（9-98）中的因变量块$\boldsymbol{c}$。

然而，PCR（以及PLSR）基于随后的经验数据表示。即，自变量块写成分解的形式，如下

$$\boldsymbol{A}=\boldsymbol{T}\boldsymbol{B}+\boldsymbol{E}_A \tag{9-100}$$

在式（9-100）中，$\boldsymbol{B}\in\Re^{h\times n}$包含装载向量（或者装载谱），行向量表示$h$个向量的新PCA基组（其中$h$是保留的因子数目，将在随后解释）。对于PCA的情况，$\boldsymbol{B}$的行向量是$\boldsymbol{A}^T\boldsymbol{A}$（协方差矩阵）的（主）特征向量，是正交的，假定$m\geqslant n$（超定情况）。因此，$\boldsymbol{B}$的行向量可以使用9.3.2～9.3.6节中提到的任何方法或者EVD方法来得到。对于$m$个测量中的每个，$\boldsymbol{T}\in\Re^{m\times h}$是在$h$个PCA装载向量的新坐标系下的强度（评分或者隐式变量）矩阵。最后，$\boldsymbol{E}_A\in\Re^{m\times n}$是测量（或者谱）残余矩阵，并不适应最优化模型。因此，在式（9-100）中，矩阵$\boldsymbol{A}$可以写成$\boldsymbol{T}$列向量和$\boldsymbol{B}$的行向量的外积，加上测量残余矩阵$\boldsymbol{E}_A$的线性组合。也就是，$\boldsymbol{A}=\sum_{q=1}^{h}\boldsymbol{t}_q\boldsymbol{b}_q+\boldsymbol{E}_A$，其中$\boldsymbol{t}_q$是$\boldsymbol{T}$的强度列向量，$\boldsymbol{b}_q$是$\boldsymbol{B}$的装载行向量（$q=1,2,\cdots,h$）。

PCR实际上由两个基本的步骤组成：PCA（数据压缩），即主特征向量的最优数目选择，以及使用PCA结果建立的校正模型，即回归步骤。一般情况下，$h<m$、$n$，存在简化数目的
431 强度。但是，为了最优化校正模型，PCR（或者PLSR）的因子优化数目必须确定。这就是为人们所知的PCR和PLSR共同拥有的因子分析能力。当因子的最优化数目被选择后，就导出最小化的数据的压缩，引发噪声以及其他模糊效果。数据中的噪声典型地分布于所有的装载向量（大部分是高阶向量），而未受破坏的测量内容一般集中于前面的极少数。

PCR（以及PLSR）的第二个一般的关系涉及到新坐标系中使用强度（或者评分）矩阵$\boldsymbol{T}$，并可写成与式（9-96）相似的关系。也就是，我们现在得到的

$$\boldsymbol{c}=\boldsymbol{T}\upsilon+\boldsymbol{e}_c \tag{9-101}$$

其中$\upsilon\in\Re^{h\times 1}$是回归系数（或者内联关系）[7, 10]。从式（9-101）中，很明显并不存在与CLS相关的问题。对于PCR方法以及$m\geqslant n$，且假定在式（9-100）中残余（$\boldsymbol{E}_A$）足够小，可以对式（9-100）用$\boldsymbol{B}^T$右乘式子两边得到如下：

$$\boldsymbol{T}\boldsymbol{B}\boldsymbol{B}^T=\boldsymbol{A}\boldsymbol{B}^T \tag{9-102}$$

然而，因为矩阵$\boldsymbol{B}$的行向量（包含装载向量）是正交的，即$\boldsymbol{B}\boldsymbol{B}^T=\boldsymbol{I}_h$，方程（9-102）可以写成如下形式

$$\boldsymbol{T} = \boldsymbol{A}\boldsymbol{B}^T \tag{9-103}$$

将式（9-103）代入式（9-101）推导出

$$\boldsymbol{c} = \boldsymbol{A}\boldsymbol{B}^T\upsilon + \boldsymbol{e}_c \tag{9-104}$$

如果定义误差代价函数如下

$$L(\upsilon) = \frac{1}{2}\| \boldsymbol{e}_c \|_2^2 \tag{9-105}$$

对于回归系数向量$\upsilon$最小化上式，这将导出最小二乘解

$$\upsilon = (\boldsymbol{B}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{B}^T)^{-1}\boldsymbol{B}\boldsymbol{A}^T\boldsymbol{c} \tag{9-106}$$

其中$\upsilon$可以看作是因子空间中的校正模型。但是，如果对于CLS问题对比式（9-104）中的模型同式（9-96）中的原始模型，可以发现

$$\hat{\boldsymbol{b}}_{f\,\mathrm{PCR}} = \boldsymbol{B}^T\upsilon = \boldsymbol{B}^T(\boldsymbol{B}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{B}^T)^{-1}\boldsymbol{B}\boldsymbol{A}^T\boldsymbol{c} \tag{9-107}$$

对于CLS模型$\hat{\boldsymbol{b}}_{f\,\mathrm{CLS}}$，比较式（9-107）中的项$(\boldsymbol{B}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{B}^T)^{-1} \in \Re^{h\times h}$与式（9-98）中相关项，也就是，$(\boldsymbol{A}^T\boldsymbol{A})^{-1} \in \Re^{n\times n}$，对于$h < n$计算式（9-107）中的校正模型$\hat{\boldsymbol{b}}_{f\,\mathrm{PCR}} \in \Re^{n\times 1}$，相对于CLS，PCR方法具有较小的计算强度。

因此，如果选择PCR因子（$h$）的最优数目，同时保持根据式（9-107）计算PCR校正模型，那么，一般结果模型比式（9-98）中使用CLS开发的校正模型具有较好的预测性能。如同CLS 432
的情况，PCR校正模型$\hat{\boldsymbol{b}}_{f\,\mathrm{PCR}}$可以用于预测自变量，假定一系列没有用于开发校正模型的独立测试数据，并考虑$\{\boldsymbol{A}_{\mathrm{test}}, \boldsymbol{c}_{\mathrm{test}}\}$，其中$\boldsymbol{A}_{\mathrm{test}} \in \Re^{p\times n}$以及$\boldsymbol{c}_{\mathrm{test}} \in \Re^{p\times 1}$，即有

$$\hat{\boldsymbol{c}}_{\mathrm{testPCR}} = \boldsymbol{A}_{\mathrm{test}}\hat{\boldsymbol{b}}_{f\,\mathrm{PCR}} \tag{9-108}$$

当$m < n$（欠定情况），$\boldsymbol{B}$的行向量（包含装载向量）也是$\boldsymbol{A}^T\boldsymbol{A}$的（主）特征向量，式（9-107）中的表达式还可以用于计算PCR校正模型$\hat{\boldsymbol{b}}_{f\,\mathrm{PCR}}$。但是，必须注意选择主特征向量，因为对于欠定情况，$\boldsymbol{A}^T\boldsymbol{A}$的特征向量并没有张成$\boldsymbol{A}$的完整行空间。因此，使用奇异值分解（参照A.2.14节）来计算$\boldsymbol{A}^T\boldsymbol{A}$的特征向量可能是最佳的。

有两个问题仍然没有得到解答。因子的最佳数目怎样选择以及校正模型的预测性能怎样可以定量的估计？首先回答第二个问题，因为第一个问题的答案依赖于第二个答案，也就是，将演示定量估计校正模型性能的方法。我们已经讨论了相对于PCR，CLS的预测性能。然而，我们并没有提出一种用于定量的确定性能的方法。一种普遍应用于确定校正模型性能的方法，例如对于PCR，可以预测未知的浓度，或者因变量$\boldsymbol{c}$，其中假定一系列没有用于计算校正模型的测试数据，这就是*标准预测误差*（SEP）[7]。使用一系列独立的测试数据$\{\boldsymbol{A}_{\mathrm{test}}, \boldsymbol{c}_{\mathrm{test}}\}$，SEP可以定义成

$$\mathrm{SEP} = \left[\sum_{i=1}^{m_{\mathrm{test}}}(c_{i\,\mathrm{test}} - \hat{c}_{i\,\mathrm{test}})^2 / m_{\mathrm{test}}\right]^{1/2} \tag{9-109}$$

其中$\boldsymbol{c}_{i\,\mathrm{test}}$是测试数据有兴趣的引用（实际）纯分量，$\hat{\boldsymbol{c}}_{i\,\mathrm{test}}$是$\boldsymbol{c}_{i\,\mathrm{test}}$的PCR（或者PLS）预测（估计），同时$m_{\mathrm{test}}$是测试测量的总数。为了完整性，我们同样定义一个相似的性能测量用来计算用于开发校正模型的训练数据$\{\boldsymbol{A}_{\mathrm{train}}, \boldsymbol{c}_{\mathrm{train}}\}$，进行计算。这称作*标准误差校正*（SEC）[7]，为训练数据$\{\boldsymbol{A}_{\mathrm{train}}, \boldsymbol{c}_{\mathrm{train}}\}$可以定义成

$$\mathrm{SEC} = \left[\sum_{i=1}^{m_{\mathrm{train}}}\frac{(c_{i\,\mathrm{train}} - \hat{c}_{i\,\mathrm{train}})^2}{m_{\mathrm{train}} - h - 1}\right]^{1/2} \tag{9-110}$$

其中是$c_{i\,\text{train}}$是训练数据有兴趣的引用（实际）纯分量，$\hat{c}_{i\,\text{train}}$是$c_{i\,\text{train}}$的PLC预测（估计），$m_{\text{train}}$是训练测量的总数，同时$h$是PCR（或者PLSR）因子数目，包括在式（9-110）的分母中，因为对于训练的数据，引了附加因子使惩罚必须置于校正模型的预测性能中。因此，式（9-110）实际上表示一个对于训练数据加权的性能测量。换句话说，通过在式（9-110）的分母中引入
433 $h$，正如更多的PCR（或者PLSR）因子附加到模型，当逆推训练数据的时候，模型的预测性能必须能够说明这一增加，因此，最小化数据的过适应[7, 118]。

最优因子数目的选择

现在可以提出最优化PCR因子的选择问题了。这里提出的方法也可以以相似的方式应用于PLSR。式（9-109）中的SEP性能测量可以用于确定PCR因子$h^o$的最优数目，以继续PCR校正的开发，即由式（9-107）给出的$\hat{\boldsymbol{b}}_{f\,\text{PCR}}$。在第一种方法中，它可能是应用最普遍的，校正模型$\hat{\boldsymbol{b}}_{f\,\text{PCR}}$（$h = 1, 2, \cdots, q$）是使用训练数据集$\{\boldsymbol{A}_{\text{train}}, \boldsymbol{c}_{\text{train}}\}$作为PCR因子所产生的。假定拥有$q$个校正模型集，对于$h = 1, 2, \cdots, q$，$\hat{\boldsymbol{b}}_{f\,\text{PCR}}$的测试数据集$\{\boldsymbol{A}_{\text{test}}, \boldsymbol{c}_{\text{test}}\}$用于估计每一个校正模型的预测性能。使用式（9-109）中的关系，对于校正模型推导出的$\boldsymbol{c}_{\text{test}}$的每一个预测，计算SEP，即$\hat{\boldsymbol{c}}_{\text{test}}$对于$h = 1, 2, \cdots, q$，使用$\boldsymbol{A}_{\text{test}}$。通过把SEP值当作PCR因子$q$的数量函数来观察，以此确定$h^o$的选择。典型地，与观察到的最小SEP相关的因子$h$的数目将指出PCR因子的最优数目，即$h^o$，如下所示

$$h^o = \{h : \text{SEP}_{\min} = \min\{\text{SEP}(h)\} \forall h = 1, 2, \cdots, h^o, \cdots, q\} \tag{9-111}$$

这种选择最优数目PCR因子$h^o$的方法称作独立检验。但是，在使用式（9-111）必须注意，因为绝对最小值可能导致这样一个结果：它可能允许保留与潜在噪声[7]相关的附加因子。因此，必须小心地估计每一情况，以协调实际保留的PCR因子的最优数目。在某些情况下，式（9-111）的绝对最小值并不是最佳选择，很多时候，一个小于绝对最小值的因子可以给出最佳的整体性能[7]。

另一种可以用于选择保留的PCR（或PLSR）因子数目的方法称作交叉确认[7, 8]。交叉确认经常称作每次保留一个的分析方法，因为先从数据集$\{\boldsymbol{A}_{\text{train}}, \boldsymbol{c}_{\text{train}}\}$保留一个测量，同时基于保留的测量开发PCR模型，然后，被保留的测量用于测试以推导出$\boldsymbol{c}_{\text{train}}$的一个预测，即$\hat{\boldsymbol{c}}_{\text{train}}$。重复这一过程，直到所有的测量都被保留并用于预测，然后根据式（9-109），对全部$m_{\text{train}}$个测量计算SEP。在测量的数目稀少，即没有足够的可行测量的时候，经常使用交叉确认。在这种情况下，没有足够的数据形成训练集和测试集。使用这种方法得到的结果并不同使用独立检验方法得到的结果一样好。理想的情况是，希望拥有一种统计表达的测量集，这样就可以
434 形成分离的训练集和测试集[4-7, 120, 121]。

## 9.5 部分最小二乘回归

部分最小二乘回归与PCR相似，是另一种基于因子分析的方法。但是，PLSR比PCR具有一个主要的优势。在PLSR的“压缩”阶段，目标值（因变量块）用于自变量块的附加，而在PCR中，只有自变量块用于基于PCA的压缩步骤中。在压缩步骤中，同时使用自变量块$\boldsymbol{A} \in \Re^{m \times n}$和因变量块$\boldsymbol{c} \in \Re^{m \times 1}$，使用PLSR的预测性能结果可以得到一个显著的效果，典型地较之CLS和PCR可以得到更好的性能。

PLSR最初由Wold[122, 123]提出，是作为经济学以及社会科学中的数据分析问题的实际解决方案。这一基本问题与PCR的情况是一致的，即，为了使校正模型拟合经验数据，假定一系列的测试数据作为训练阶段之后的校正模型的输入，使用这一方法来预测某些量。PLSR经常涉及到抽象因子分析（降秩）的内容，因为系统中重要的量并不能总是利用数学建模过

程[124]来直接确定。然而，如果解释合理，构成PLSR的因子分析方法的数学基础可以推导出非常有力的、与物理设置中某些关键特征相关的信息，并从物理设置中提取数据用于PLSR的建模过程 [12]。此外，PLSR认为是全谱技术，对应于逆最小二乘（ILS）方法[12]。因此，使用来自于PLS过程的剩余误差的有效出格点技术是可行的。出格点认为是不具有校正采样的样本。因此，并行估计的输出必须仔细核查[7, 12]。

当提出PCR的时候，假定限制于单分量情形[7]的讨论，这一点与PLSR是相同的。这里我们将提出PLSR1校正算法（单分量情形）和两个不同的预测算法。在9.6节将要提出一种PLSR神经网络的实现。正如前面提到的，为PCR所提出的基础模型也是PLSR的基础，即式（9-100）和式（9-101）。但是，我们将看到在PLSR1校正算法的陈述中，其遵循Haaland和Thomas[12]提出的方法，$\boldsymbol{A}$与$\boldsymbol{c}$都用于压缩阶段。存在许多PLSR算法的不同方法和PLSR[7-13]的基本规则的各种表示。下面给出的解释可能是最直接的，因此更容易理解。同样，遵循Haaland和Thomas[12]提出的方法的两种预测方法也是如此。

下面给出的PLSR1校正算法中，7个主要步骤中的每一步都是由CLS方法推导出的。对于校正算法的一般解释，数据$\{\boldsymbol{A}, \boldsymbol{c}\}$用作训练数据，即最终的开发校正模型的$\{\boldsymbol{A}_{\text{train}}, \boldsymbol{c}_{\text{train}}\}$。但是，435
在PLS1校正算法解释中，“train”下标省略了，以避免混淆。

**PLSR1校正算法**

**步骤1：数据的平均中心以及方差规整。**第一步是数据的预处理，即数据的平均中心以及方差规整[7, 10]。我们将不给出数据的这种预处理的基本原因详尽的解释。然而，会解释一些典型的需要数据预处理情况。例如，如果收集的数据（测量），即$\boldsymbol{A}$的行向量包含与数据相关的偏置，那么数据的平均中心是可行的。这一过程也会对于因变量即$\boldsymbol{c}$执行。这里基本完成消除对于数据的非零截距需求，通常导致校正模型复杂度的下降。也就是用于建模数据[12]所需PLSR因子数目的减少。如果收集的数据使用不同的单元进行测量，那么方差规整就是可行的。平均中心包括计算$\boldsymbol{A}$每一列的平均值和从相应列的每一个元素中减去特殊列的均值。如果$\boldsymbol{A}$是平均中心的，$\boldsymbol{c}$也应该是平均中心的。方差规整包括计算$\boldsymbol{A}$中每一列的标准偏差，然后通过相关的标准偏差值除以各自列中的每一个元素。对于因变量$\boldsymbol{c}$也执行相同的处理。然而，这通常只对于多分量的情况是必要的。关于平均中心和方差规整是统计学家中一直持续的讨论。许多人坚持数据应该经常进行预处理，而另外一些人坚持数据不应该进行平均中心和方差规整[121]。我们认为，基于上面的原因，数据预处理是必要的。然而，如果没有足够的强制性原因，进行数据的平均中心和方差规整的处理不应该任意执行。

首先，索引$h$（PLS因子的数目）初始化为1。

**步骤2：构成权值装载向量$\hat{\boldsymbol{w}}_h \in \Re^{n\times 1}$。**这一步实际上是一个CLS校正，使用的模型有如下形式

模型：
$$\boldsymbol{A} = \boldsymbol{c}\boldsymbol{w}_h^T + \boldsymbol{E}_A \tag{9-112}$$

其中最小二乘解如下

最小二乘解：
$$\hat{\boldsymbol{w}}_h = \boldsymbol{A}^T\boldsymbol{c}/\boldsymbol{c}^T\boldsymbol{c} \tag{9-113}$$

然后正规化$\hat{\boldsymbol{w}}_h$，即

$$\hat{\boldsymbol{w}}_h \leftarrow \frac{\hat{\boldsymbol{w}}_h}{\|\hat{\boldsymbol{w}}_h\|_2}$$

在式（9-113）中，对于每个$h$增量，每个向量$\hat{\boldsymbol{w}}_h$是与矩阵$\boldsymbol{A}$中行元素的加权平均成比例的权值向量，其中平均权值与$\boldsymbol{c}$中的元素是成比例的。每一个权值向量$\hat{\boldsymbol{w}}_h$正规化的同时，构建成相互 436

正交的。因此，$\hat{w}_h$ 向量是正交的。这一步与使用PCR方法[4, 6-8]有着很大的不同，因为因变量$c$中用到的信息附加到$A$，以形成PLSR中的权值向量。在PCR中，只有矩阵$A$中的信息在这一步使用。式（9-112）中的矩阵$E_A \in \Re^{m \times n}$包含与$A$相联系的剩余误差。

**步骤3：生成计分（隐式变量）向量$\hat{t}_h \in \Re^{m \times 1}$。**在这一步，$A$现在相对于隐式变量或者计分可以写成

模型：
$$A = t_h \hat{w}_h^T + E_A \tag{9-114}$$

其中最小二乘解可由下式给出

最小二乘解：
$$\hat{t}_h = A\hat{w}_h / \hat{w}_h^T \hat{w}_h = A\hat{w}_h \tag{9-115}$$

这一步CLS也是如此，其中如式（9-115）中所述，通过基于$\hat{w}_h$回归$A$得到$t_h$的最小二乘估计$\hat{t}_h$。$\hat{t}_h$的各元素说明了在数据矩阵$A$中的每一个行中包含多少个$\hat{w}_h$。对于新的PLS坐标系统，向量$\hat{t}_1$表示$A$的行数据中第一权值装载向量的强度（或者数量）。因为$\hat{w}_1$是一阶尝试，表示从$A$的行向量中的干扰（噪声）数据得到的未受干扰的数据，$\hat{t}_1$表示一阶尝试确定在$A$的每一个关联行向量中纯分量值（即，包含在$c$中信息）的数目。因此，在PLSR方法中，每一个$\hat{t}_h$向量与$A$和$c$相关，而不是只是与$A$相关，如PCR中的情况。

**步骤4：建立计分向量$\hat{t}_h$与$c$的元素的关系。**在这一步中，表示新的PLSR坐标系中的强度的计分向量$\hat{t}_h$（或者与矩阵$A$的每一行中包含的纯元素值关联的主特征联系的隐式向量）关联着向量$c$的元素，其中通过使用线性最小二乘回归。在PLSR中，因为与逆最小二乘方法[12, 125]以及PCR相反，在估计每一个权值向量之后，得到计分$\hat{t}_h$和$c$向量（或者$c$剩余误差）的元素之间的独立关联。$\hat{t}_h$和$c$之间的关系建模成

模型：
$$c = \upsilon_h \hat{t}_h + e_c \tag{9-116}$$

最小二乘解有如下形式

最小二乘解
$$\hat{\upsilon}_h = \hat{t}_h^T c / \hat{t}_h^T \hat{t}_h \tag{9-117}$$

其中对于每一个$h$增量式（9-117），提出一个用于关联$\hat{t}_h$与$c$中元素的标量回归系数（内在关系）$\upsilon_h \in \Re$的估计。式（9-116）中的向量$e_c \in \Re^{m \times 1}$包含与$c$相关的PLSR剩余误差。式（9-117）中的关系与ILS解相似，因为平方的$c$误差之和最小化了。然而，在这一步，与ILS相似的解是一个元素接一个元素建立的。

437 **步骤5：$\hat{b}_h \in \Re^{n \times 1}$的泛化，$A$的PLS装载向量。**为了去除共线性，正交$\hat{t}_h$向量（即，线性依赖）是可取的。通过基于隐式变量$\hat{t}_h$为矩阵$A$组成一个新模型来得到正交向量$\hat{t}_h$。新模型有如下形式

模型：
$$A = \hat{t}_h b_h^T + E_A \tag{9-118}$$

其中最小二乘解有如下形式

最小二乘解：
$$\hat{b}_h = A^T \hat{t}_h / \hat{t}_h^T \hat{t}_h \tag{9-119}$$

向量$\hat{b}_h$（对于$h = 1, 2, \cdots$）是PLSR的装载向量。算法中的这一步以及下一步确保$\hat{t}_h$个向量将是相互正交的。最小二乘回归对于矩阵$A$中的每一行的所有样本同时进行，如式（9-119）。与PCR中的第一PCA装载向量不同，由式（9-119）所确定的第一PLSR装载向量$\hat{b}_1$并没有归因于矩阵$A$行中的最大方差。但是，当同时与估计$c$的$\hat{t}_h$相关联时，这并不表示一种尝试说明在矩阵$A$中有同样多的变化。与PCA也不一样，$\hat{b}_h$个向量并不是相互正交的。此外，既然$\hat{t}_1$是$c$向量的一阶近似，与向量$\hat{b}_1$的最大正元素相关的，矩阵$A$中的列元素倾向于指示矩阵$A$中的这些列元素，它们表明了由于特殊的装载向量，矩阵对于$c$中元素的最大依赖性。但是，直接与向量

$c$相关的向量$\hat{w}_1$将表明这一倾向比$\hat{b}_1$要好一些，因此，对于从PLSR1分析中提取的信息，$\hat{w}_1$将比$\hat{b}_1$更加有用。

**步骤6：矩阵$A$与$c$的剩余误差计算。**计分向量$\hat{t}_h$和装载向量$\hat{b}_h$的积是PLSR对于矩阵$A$的近似。矩阵$A$的剩余误差$E_A$通过从$A$的行向量中的测量中减去矩阵$A$的行向量PLSR的近似来计算，如下

$A$剩余误差：
$$E_A = A - \hat{t}_h \hat{b}_h^T \tag{9-120}$$

$c$剩余误差：
$$e_c = c - \hat{v}_h \hat{t}_h \tag{9-121}$$

类似地，已经通过PLSR建模的向量$c$中的信息成分可以通过消除得到$c$中的剩余误差，即$e_c$，如式（9-121）中所示。在式（9-121）中$\hat{v}_h \hat{t}_h$的积表示基于矩阵$A$信息$c$，其PLSR估计为$\hat{c}$。

**步骤7：增量$h$，在步骤2中以$E_A$代替$A$并以$e_c$代替$c$，然后继续直到期望的装载向量的数目（或者PLSR因子$h^o$的最佳数目）。**

**PLSR1校正算法小结**

**步骤1** 数据预处理：平均中心$A \in \Re^{m \times n}$以及$c \in \Re^{m \times 1}$，如果必要，还有方差规整$A$。令指数$h$等于1（其中$h$是PLS因子的数目）。

438

**步骤2** 组成权值装载向量$\hat{w}_h \in \Re^{n \times 1}$

模型：$A = cw_h^T + E_A$（$E_A \in \Re^{m \times n}$包含$A$的剩余误差）

最小二乘解：$\hat{w}_h = \dfrac{A^T c}{c^T c}$

正规化$\hat{w}_h$，即$\hat{w}_h \leftarrow \dfrac{\hat{w}_h}{\| \hat{w}_h \|_2}$

**步骤3** 计分（隐式变量）向量$\hat{t}_h \in \Re^{m \times 1}$的生成

模型：$A = \hat{t}_h w_h^T + E_A$

最小二乘解：$\hat{t}_h = \dfrac{A\hat{w}_h}{\hat{w}_h^T \hat{w}_h} = A\hat{w}_h$

**步骤4** 将计分向量$\hat{t}_h$与$c$的元素相联系。

模型：$c = v_h \hat{t}_h + e_c$（$e_c \in \Re^{m \times 1}$包含剩余量c）

最小二乘解：$\hat{v}_h = \dfrac{\hat{t}_h^T c}{\hat{t}_h^T \hat{t}_h}$

其中$v_h \in \Re$是将计分向量$\hat{t}_h$与$c$的元素相联系的标量回归系数（内在关系）。

**步骤5** $A$的装载向量$\hat{b}_h \in \Re^{n \times 1}$的生成

模型：$A = \hat{t}_h b_h^T + E_A$

最小二乘解：$\hat{b}_h = \dfrac{A^T \hat{t}_h}{\hat{t}_h^T \hat{t}_h}$

**步骤6** $A$以及$c$中的剩余误差的计算

$A$的剩余误差：$E_A = A - \hat{t}_h \hat{b}_h^T$

$c$的剩余误差：$e_c = c - \hat{v}_h \hat{t}_h$

**步骤7** 增量$h$，在步骤2中以$E_A$代替$A$，以$e_c$代替c，继续直到期望的装载向量的数目，或者PLS因子$h^o$的最佳数目，其中$h^o = m$（其中$m < n$）或者$h^o = n$（其中$m \geqslant n$）。

PLSR1预测算法

**预测算法1**。存在两种不同方法用于预测。即考虑一个未知的测量，利用从PLSR1校正阶段抽取的信息来预测期望的因变量的分量。这里提出的第一个预测算法符合Haaland 和Thomas[12]对方法2的解释。随后的过程很容易理解，分解成两部分：校正模型开发以及使用校正模型的预测。这种方法的不足之一是它并没有按照矩阵$\boldsymbol{A}$的剩余误差的确定。因此，当获
439 得预测时，没有关于适合数据校正模型的质量的可行诊断信息。

根据上面提出的PLSR1校正算法，校正模型$\hat{\boldsymbol{b}}_{f\,\mathrm{PLSR}}\in\Re^{n\times 1}$可以由权值装载向量$\hat{\boldsymbol{w}}_h\,(h=1,2,\cdots,q)$、$\boldsymbol{A}$的装载向量$\hat{\boldsymbol{b}}_h\,(h=1,2,\cdots,q)$，以及内在关系$\hat{\upsilon}_h\,(h=1,2,\cdots,q)$组成。如果生成所有的PLSR因子，那么$q=m$，其中$m<n$，以及$q=n$，其中$m\geqslant n$。理想地，只使用最优数目的PLSR因子，即$q=h^o$。最优数目的PLSR因子可以产生一个校正模型$\hat{\boldsymbol{b}}_{f\,\mathrm{PLSR}}$，它可以基于独立测试数据推导出最小预测性能误差，因此不允许模型过适应数据。确定$h^o$的过程同前面提到的PCR方法是一致的。为了产生校正模型，我们首先构造矩阵

$$\hat{\boldsymbol{W}}^T=[\hat{\boldsymbol{w}}_1,\hat{\boldsymbol{w}}_2,\cdots,\hat{\boldsymbol{w}}_q] \tag{9-122}$$

以及

$$\hat{\boldsymbol{B}}^T=[\hat{\boldsymbol{b}}_1,\hat{\boldsymbol{b}}_2,\cdots,\hat{\boldsymbol{b}}_q] \tag{9-123}$$

以及

$$\hat{\upsilon}^T=[\hat{\upsilon}_1,\hat{\upsilon}_2,\cdots,\hat{\upsilon}_q] \tag{9-124}$$

其中$\hat{\boldsymbol{W}}\in\Re^{q\times n}$，$\hat{\boldsymbol{B}}\in\Re^{q\times n}$，以及$\hat{\upsilon}\in\Re^{q\times 1}$。从式（9-122）、式（9-123）、式（9-124）以及$q=h^o$，最终的最优校正模型（或者最终校正系数）$\hat{\boldsymbol{b}}_{f\,\mathrm{PLSR}}$可以得到如下形式

$$\hat{\boldsymbol{b}}_{f\,\mathrm{PLSR}}=\hat{\boldsymbol{W}}^T(\hat{\boldsymbol{B}}\hat{\boldsymbol{W}}^T)^{-1}\hat{\upsilon} \tag{9-125}$$

**PLSR1预测方法1小结**

**步骤1**　从PLSR1校正阶段，组成以下矩阵（对于最优或者期望数目的PLSR因子$h=h^o$）：

$\hat{\boldsymbol{W}}^T=[\hat{\boldsymbol{w}}_1,\hat{\boldsymbol{w}}_2,\cdots,\hat{\boldsymbol{w}}_q]$其中$\hat{\boldsymbol{W}}\in\Re^{h^o\times n}$

$\hat{\boldsymbol{B}}^T=[\hat{\boldsymbol{b}}_1,\hat{\boldsymbol{b}}_2,\cdots,\hat{\boldsymbol{b}}_q]$其中$\hat{\boldsymbol{B}}\in\Re^{h^o\times n}$

$\hat{\upsilon}^T=[\hat{\upsilon}_1,\hat{\upsilon}_2,\cdots,\hat{\upsilon}_1]$其中$\hat{\upsilon}\in\Re^{h^0\times 1}$

**步骤2**　计算最终回归系数，或者最优校正模型$\hat{\boldsymbol{b}}_{f\mathrm{PLSR}}$

$$\hat{\boldsymbol{b}}_{f\,\mathrm{PLSR}}=\hat{\boldsymbol{W}}^T(\hat{\boldsymbol{B}}\hat{\boldsymbol{W}}^T)^{-1}\hat{\upsilon}$$

[注意：$\rho(\hat{\boldsymbol{B}}\hat{\boldsymbol{W}}^T)=h^o$，其中$h^o\ll n,m$]

**步骤3**　考虑一组测量$\boldsymbol{A}_{\mathrm{test}}$（不用于开发校正模型$\hat{\boldsymbol{b}}_{f\mathrm{PLSR}}$），估计输出（或者因变量）

$$\hat{\boldsymbol{c}}_{\mathrm{test}}=\boldsymbol{A}_{\mathrm{test}}\hat{\boldsymbol{b}}_{f\,\mathrm{PLSR}}$$

或者，假如训练数据是平均中心的（其中依赖参考数据的均值由$\bar{\boldsymbol{c}}_{\mathrm{train}}$给出）

440
$$\hat{\boldsymbol{c}}_{\mathrm{test}}=\boldsymbol{A}_{\mathrm{test}}\hat{\boldsymbol{b}}_{f\mathrm{PLSR}}+\bar{\boldsymbol{c}}_{\mathrm{train}}$$

式（9-125）提出的校正模型（向量）由训练数据$\{\boldsymbol{A}_{\mathrm{train}},\boldsymbol{c}_{\mathrm{train}}\}$开发得到，现在可以用于预测。即，假定另外一组没有用于校正模型开发的测量$\boldsymbol{A}_{\mathrm{test}}$，并通过将矩阵$\boldsymbol{A}_{\mathrm{test}}$投影到$\hat{\boldsymbol{b}}_{f\,\mathrm{PLSR}}$，对于测试数据可以得到$\boldsymbol{c}$的估计如下

$$\hat{\boldsymbol{c}}_{\mathrm{test}}=\boldsymbol{A}_{\mathrm{test}}\hat{\boldsymbol{b}}_{f\mathrm{PLSR}} \tag{9-126}$$

如果数据是平均中心的，其中依赖参考数据的均值如$\bar{\boldsymbol{c}}_{\mathrm{train}}$，那么

$$\hat{\boldsymbol{c}}_{\text{test}} = \boldsymbol{A}_{\text{test}}\hat{\boldsymbol{b}}_{f\,\text{PLSR}} + \bar{c}_{\text{train}} \tag{9-127}$$

显然，将式（9-125）矩阵的逆应用于计算校正模型。以前对于CLS的描述，其数据矩阵很大，这是使用这一方法的一个缺点。但是，对于PLSR的情况，式（9-125）的矩阵必须计算其逆，即$\hat{\boldsymbol{B}}\hat{\boldsymbol{W}}^T$只是$q \times q$，或者$h^o \times h^o$，在维数上，其中特别是$h^o << n, m$，这一点与PCR相似。

**预测算法2**。预测的第二种方法并不需要计算矩阵的逆，是9.6节提出的PLSNET-P神经网络结构的基础。如前所述，前面方法的不足是它不允许矩阵$\boldsymbol{A}$剩余误差的确定。然而，随后的方法允许计算剩余误差并有益于在神经结构中实现。也就是说，矩阵的逆不需要得到预测。算法的提出符合Haaland和Thomas[12]对方法1的解释。这一算法是递归的，分5步。

考虑未知的测量或者观察$\boldsymbol{a}_i \in \Re^{n \times 1}$，$(i = 1, 2, \cdots, m)$（其中$m$是测试输入的数目），另一种（递归）预测算法可以总结如下。（注意：为了方便，$\boldsymbol{a}_i$看作是一个列向量。）

**PLSR1预测方法2小结**

**步骤1**　平均中心以及方差规整。如果矩阵$\boldsymbol{A}$在校正阶段是平均中心以及方差规整，$\boldsymbol{a}$（注意$i$下标为了方便被忽略了，其中$i$表示特殊测量）使用校正数据也是平均中心和方差规整的，令$h = 1$。令$c_h = 0$以及$\boldsymbol{e}_{a0} = \boldsymbol{a}$。

**步骤2**　测量强度$\hat{t}_h$的计算

$$\hat{t}_h = \hat{\boldsymbol{w}}_h^T \boldsymbol{a} \tag{9-128}$$

利用校正阶段的权值装载向量$\hat{\boldsymbol{w}}_h$。这与校正阶段[见PLSR1校正算法的步骤3中的式（9-115）]是相同的步骤。

**步骤3**　因变量$c$的更新估计的计算

$$c_h = c_{h-1} + \hat{\upsilon}_h \hat{t}_h \tag{9-129}$$

其中PLSR1校正阶段中的$\hat{\upsilon}_h$是标量回归系数（对于内在关系），假如数据是平均中心的，$c_0$是在PLSR1校正训练阶段中使用的目标值的平均。注意，式（9-129）相似于式（9-116）的模型。 441

**步骤4**　$\boldsymbol{a}$的剩余误差计算

$$\boldsymbol{e}_{ah} = \boldsymbol{e}_{ah-1} - \hat{\boldsymbol{b}}_h \hat{t}_h \tag{9-130}$$

其中$\hat{\boldsymbol{b}}_h$是PLSR1校正阶段的装载向量。注意，式（9-130）同计算矩阵$\boldsymbol{A}$的剩余误差的校正算法具有相同的表达式[见PLSR1校正算法中的式（9-120）]。

**步骤5**　回到步骤2。增量$h$（即，$h \leftarrow h + 1$），为$\boldsymbol{a}$代换$\boldsymbol{e}_{ah}$，直到$h = h^o$[其中$h^o$是PLSR因子的期望（或最优）数，或者装载向量或评分]。当$h = h^o$，那么预测$\hat{c} = c_{h^o}$。

正如我们所看到的，这个预测算法实际上是PLSR1校正算法的子类。在下一节PLSNET校正和预测神经结构介绍之后，这一点将会很明显。

## 9.6　部分最小二乘回归的神经网络方法

在9.5节中介绍的PLSR1校正算法以及PLSR1预测方法2是递归算法，它们可以在神经网络结构中以一种相对简单的方式实现。用于实现PLSR的神经网络结构称作部分最小二乘网络（PLSNET），它最初由Ham以及Kostanic[126, 127]提出。PLSNET实际上是由两个分离但耦合的神经网络组成，基于前面提出的PLSR1校正算法的PLSNET-C（对于PLSR校正），以及基于前面提出的预测方法2的PLSNET-P（对于PLSR1预测）。PLSNET-C是一种自适应模块的线性神经网络，根据3个标准的Hebb学习规则来训练，允许PLSR权值装载向量、回归系数以及装载向量的求取。PLSNET-C是一种在训练阶段中使用期望响应（输出或者因变量）信息的监督

训练神经网络。由PLSNET-C求取的校正信息用于预测。也就是，PLSNET-P使用由PLSNET-C求取的3个系列的突触权值来预测由测试输入给出的输出（或者因变量）。将首先介绍PLSNET-C，接下来介绍PLSNET-P。然后再进行两个网络之间耦合的解释。

PLSNET-校正（PLSNET-C）

PLSR校正阶段（即PLSNET-C）中的PLSNET神经结构如图9-16所示。图9-16中的每一个神经元都是一个线性处理器。也就是，每个神经元的激活函数是线性的。图中显示的各种阶
442 段与保留的PLSR因子的数目相关。也就是，在PLSNET-C阶段中，PLSR因子的最优数目是根据前面解释的独立校验方法来确定的。PLSNET-C需要的每一个附加因子只是简单的通过给网络加入一个阶段来实现。例如，在PLSNET-C对于具体数目的因子的初始训练之后，希望增加另外一个因子，这一点可以通过简单的给网络增加另一阶段来实现。结果网络并非完全地重新训练，只利用以前的训练阶段中已经为上面阶段设置的突触权值来训练增加的附加级。对于每一PLSR因子，这一阶段是PLSNET-C结构的自适应模块特性。在图9-16中给出的结构与图9-4（参照9.3.5节）中的PCA的APEX网络相似，但有着明显的不同，即在训练过程中PLSNET-C包含目标，或者因变量，信息，而APEX则不然。因此，PLSNET-C是在一个监督模型中被训练的，而APEX是一个无监督训练神经网络。

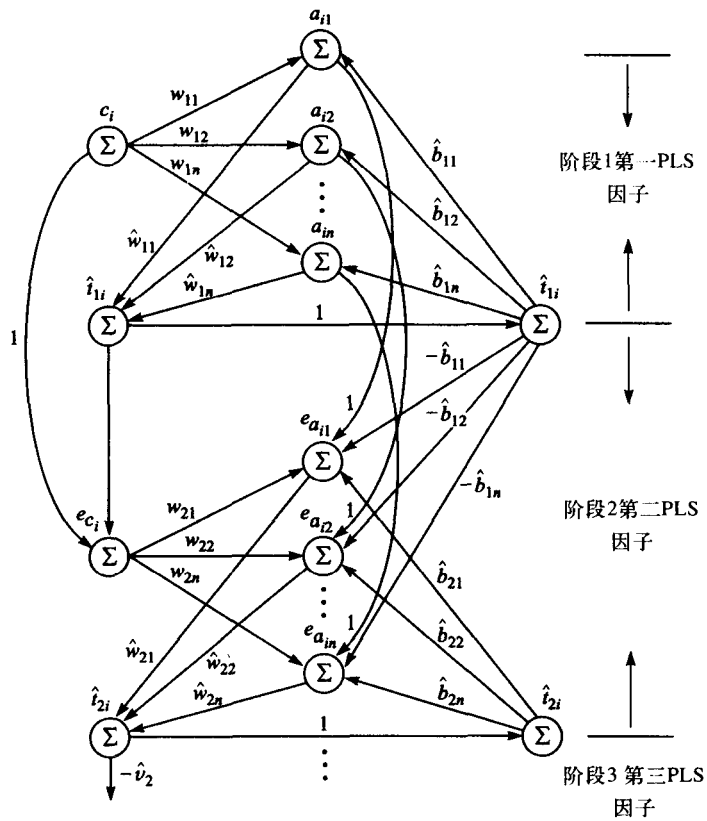



图9-16　PLSNET-C结构，对于自适应抽取PLSR权值装载向量$\hat{w}_h$、回归系数$\hat{v}_h$以及装载向量$\hat{b}_h(h=1,2,\cdots,h^o)$其中$h^o$是PLSR因子的最优数目

如前所述，PLSNET-C根据3个标准的Hebb学习规则训练，求取PLSR权值装载向量$\hat{w}_h$、
443 回归系数$\hat{v}_h$以及装载向量$\hat{b}_h$($h=1,2,\cdots,h^o$)，其中$h^o$是PLSR因子的最优数目。三个Hebb学

习规则中的每一个都可以由图9-16中描述的PLSNET-C结构来推导得到。也就是，图9-17描述了在图9-16中网络结构的每一部分，分别与求解PLSR权值装载向量、回归系数以及装载向量相关。

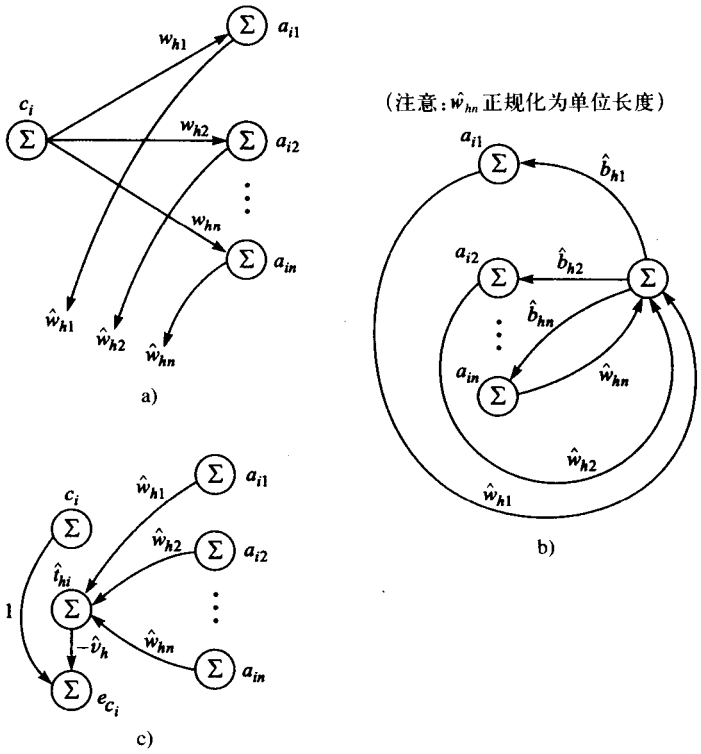



图9-17　a)在图9-16中为了求取PLS权值装载向量$\hat{\boldsymbol{w}}_h$的PLSNET-C结构的子网络；b) 在图9-16中为了求取PLS装载向量$\hat{\boldsymbol{b}}_h$的PLSNET-C结构的子网络；c) 在图9-16中为了求取PLS标量回归系数$\hat{\upsilon}_h$的PLSNET-C结构的子网络

我们将首先集中精力推导求取PLSR权值转载向量的学习规则，其他两个回归系数以及装载向量的学习规则能以相似的方式推导。图9-17a描述了对于负责求取PLSR权值装载向量的PLSNET-C网络（对于单阶段）的一部分，对于单一测量或者观察底层基础模型如下所示

$$\boldsymbol{a}_i = c_i \boldsymbol{w}_h + \boldsymbol{e}_{a_i}^{w} \tag{9-131}$$

其中$\boldsymbol{a}_i \in \Re^{n\times 1}$（为了方便$\boldsymbol{a}_i$看作一个列向量），且误差向量$\boldsymbol{e}_{a_i}^{w} \in \Re^{n\times 1}(i = 1, 2, \cdots, m)$。这与在式(9-112）所提出的模型结构是相同的。

444

构造PLSNET-C以达到一次只处理一个测量。为了推导出第一个学习规则，误差代价函数（李雅普诺夫函数）可以写作

$$L_w(\boldsymbol{w}_h) = \frac{1}{2} \| \boldsymbol{e}_{a_i}^{w} \|_2^2 \tag{9-132}$$

其中$\boldsymbol{e}_{a_i}^{w}$是同式（9-131）的测量相关的误差。式（9-132）关于权值向量$\boldsymbol{w}_h$的梯度可以以如下计算

$$\nabla L_w(\boldsymbol{w}_h) = \frac{\partial L_w(\boldsymbol{w}_h)}{\partial \boldsymbol{w}_h} = -c_i \boldsymbol{a}_i + c_i^2 \hat{\boldsymbol{w}}_h \tag{9-133}$$

因此，使用Amari[64]的结论，PLSR权值装载向量的离散时间学习规则有如下形式

$$\hat{\boldsymbol{w}}_h(k+1)-\hat{\boldsymbol{w}}_h(k)=-\mu_w\frac{\partial L_w(\boldsymbol{w}_h)}{\partial \boldsymbol{w}_h}=\mu_w[c_i\boldsymbol{a}_i-c_i^2\hat{\boldsymbol{w}}_h(k)] \tag{9-134}$$

或者

$$\hat{\boldsymbol{w}}_h(k+1)=\hat{\boldsymbol{w}}_h(k)+\mu_w c_i[\boldsymbol{a}_i-\boldsymbol{c}_i\hat{\boldsymbol{w}}_h(k)] \tag{9-135}$$

其中$k$是离散时间索引且$\mu_w>0$是学习参数。在图9-17a中描述的PLS权值装载向量的正规化是不必要的。但是，它已经包括其中了，那么PLSNET-C可以直接与经典的PLSR1校正算法相联系。

另外两种求取PLSR装载向量（$\hat{\boldsymbol{b}}_h$）和标量回归系数（$\hat{\upsilon}_h$）的学习规则可用相似的方式来推导，至于权值装载向量分别通过参考图9-17b和图9-17c。对于3个学习规则中每一个，其结果总结在表9-1中。

在表9-1中计算3个系列的权值，即$\{\hat{\boldsymbol{w}}_h, \hat{\boldsymbol{b}}_h, \hat{\upsilon}_h\}$（其中$h=1, 2, \cdots, h^o$）的3个学习规则中，可以发现标准的Hebb共存项（这是在3个表达式中每一个的右手边的第一项）。每个表达式右手边的第二项表示一个活性衰退，它实质上阻止了各自的权值$\{\hat{\boldsymbol{w}}_h, \hat{\boldsymbol{b}}_h, \hat{\upsilon}_h\}$在训练过程中成为无界。对于学习第一主特征向量（参照9.3.1节），这三个学习规则与式（9-25）所提出的Oja的正规化Hebb学习规则有着相同的形式。它可以描述[126]成如果学习率$\mu_w$满足不等式

$$0<\mu_w<\left(\sum_{i=1}^{m}c_i^2\right)^{-1} \tag{9-136}$$

其中$c_i$，对于$i=1, 2, \cdots, m$，是训练目标值，同样也有

$$\mu_w\geqslant\mu_b\geqslant\mu_\upsilon \tag{9-137}$$

445 那么PLSNET-C保证收敛。其他PLSR神经网络实现方式，例如Holcomb和Morari[128]提出了一个前馈线性神经网络结构。同样，非线性PLSR神经网络结构已经由Qin和McAvoy[129]以及Malthouse[130]提出。但是，这里提出的PLSR神经网络与这结构主要在PLSNET-C自适应模块特性方面有着显著的不同。

**表9-1　PLSNET-C学习规则小结**

| PLS信息 | 误差函数　　误差代价函数 | PLSNET-C学习规则 |
|---|---|---|
| $\hat{\boldsymbol{w}}_h\in\Re^{n\times 1}$（权值装载向量） | $\boldsymbol{e}_{a_i}^w=\boldsymbol{a}_i-c_i\boldsymbol{w}_h$　$L_w(w_h)=\frac{1}{2}\Vert e_{a_i}^w\Vert_2^2$ | $\hat{\boldsymbol{w}}_h(k+1)=\hat{\boldsymbol{w}}_h(k)+\mu_w c_i[\boldsymbol{a}_i-c_i\hat{\boldsymbol{w}}_h(k)]$ |
| $\hat{\boldsymbol{b}}_h\in\Re^{n\times 1}$（装载向量） | $\boldsymbol{e}_{a_i}^b=\boldsymbol{a}_i-\hat{t}_{hi}\boldsymbol{w}_h$　$L_b(\boldsymbol{b}_h)=\frac{1}{2}\Vert e_{a_i}^b\Vert_2^2$ | $\hat{\boldsymbol{b}}_h(k+1)=\hat{\boldsymbol{b}}_h(k)+\mu_b\hat{t}_{hi}[\boldsymbol{a}_i-\hat{t}_{hi}\hat{\boldsymbol{b}}_h(k)]$ |
| $\hat{\upsilon}_h\in\Re$（标量回归系数或内在关系） | $e_{c_i}^\upsilon=\boldsymbol{c}_i-\upsilon_h\hat{t}_{hi}$　$L_\upsilon(\upsilon_h)=\frac{1}{2}\Vert e_{c_i}^\upsilon\Vert_2^2$ | $\hat{\upsilon}_h(k+1)=\hat{\upsilon}_h(k)+\mu_\upsilon\hat{t}_{hi}[c_i-\hat{t}_{hi}\hat{\upsilon}_h(k)]$ |
| I. 其中学习规则参数必须满足$\mu_w\geqslant\mu_b\geqslant\mu_\upsilon$<br>II. $h=1, 2, \cdots, h^o$(PLS因子的最优化数)<br>III. $c_i$是训练目标值<br>IV. $\boldsymbol{a}_i$是训练输入模式<br>V. $\hat{\boldsymbol{t}}_{hi}=\hat{\boldsymbol{w}}_h^T\boldsymbol{a}_i$（测量强度）<br>VI. $i=1, 2, \cdots, m$（测量数） | | |

PLSNET预测（PLSNET-P）

图9-18描述了PLSNET-P结构。这一结构基于前面的PLSR1预测算法（方法2）。这一结构

实际上是PLSNET-C结构的一个子网络。操作中的主要不同是PLSNET-P并不使用目标数据，但是为自变量$\hat{c}$来预测（或者估计）值，假定网络有一个测试测量输入$\boldsymbol{a}_{\text{test}\,i}(i = 1, 2, \cdots)$。

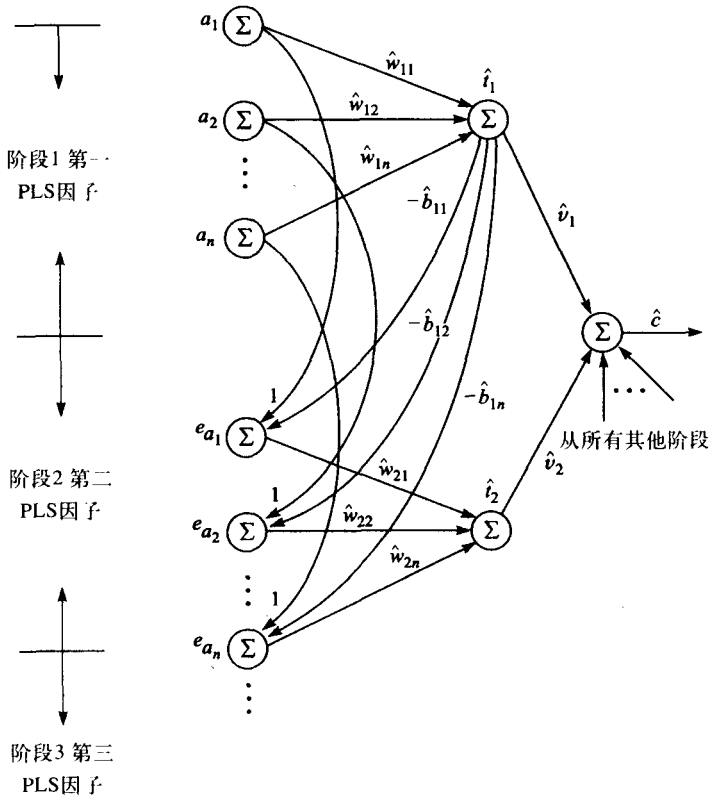



图9-18 用于因变量（$\hat{c}$）的预测（或者估计）的PLSNET=P结构，假定输入是一个独立测试测量或者观察

因此，PLSNET-P神经网络是不用训练的，但是它的权值是根据从训练PLSNET-C神经结构中提取的信息来设置的，也就是$\{\hat{\boldsymbol{w}}_h, \hat{\boldsymbol{b}}_h, \hat{\upsilon}_h\}$其中$h = 1, 2, \cdots, h^o$。这一点可以在图9-19中发现，通过3组突触权值$\{\hat{\boldsymbol{w}}_h, \hat{\boldsymbol{b}}_h, \hat{\upsilon}_h\}$描述了耦合。PLSNET-C与PLSNET-P之间的相似性主要包括与输入测量相关的误差剩余的计算，同时两者必须计算PLSR分数（或者隐式变量）。如图9-18所示，测试输入测量误差$\boldsymbol{a}_{\text{test}}$提交给阶段1（注意“test”下标已从图9-18中的测试输入测量样本中除去以避免混淆）。图9-19所描述的PLSNET-P输出是考虑一个特殊的测试输入时，一个与因变量（或响应变量）$\hat{c}$相联系的预测（或者估计）。

**例9.4** 对比PLSNET以及CLS方法的预测性能。这个例子中的数据由一组与中心目标值相关的200个模拟近红外（NIR）谱组成，其中对于谱中感兴趣的分量，中心目标值分布于2.7～500 mg/dL（毫克每分升）之间。这200个谱是从基兴趣分量[图9-20a所描述的作为最低幅度谱]以及一个基模糊分量[图9-20a所描述的较高幅度谱]产生的。两种分量都是作为各自附带的两个吸收频带的高斯函数而产生的，如图9-20a所示。模糊分量认为是模拟NIR水吸收，后者控制着NIR兴趣分量[131]的吸收。因此，200个模拟NIR谱是通过感兴趣的模糊分量3次较大量增加模糊分量来得到的，同时带有$\sigma^2 = 9$的相对方差的0均值高斯噪声在附加伴有随机振幅模糊的分量模拟分光光度计，即基线方差[7]的非理想特性之前被附加到感兴趣的分量当中。图9-20b描述了200个模拟NIR谱中的5个。因为水吸收的支配效果，所有的谱看起来似乎是一样的。

446～447

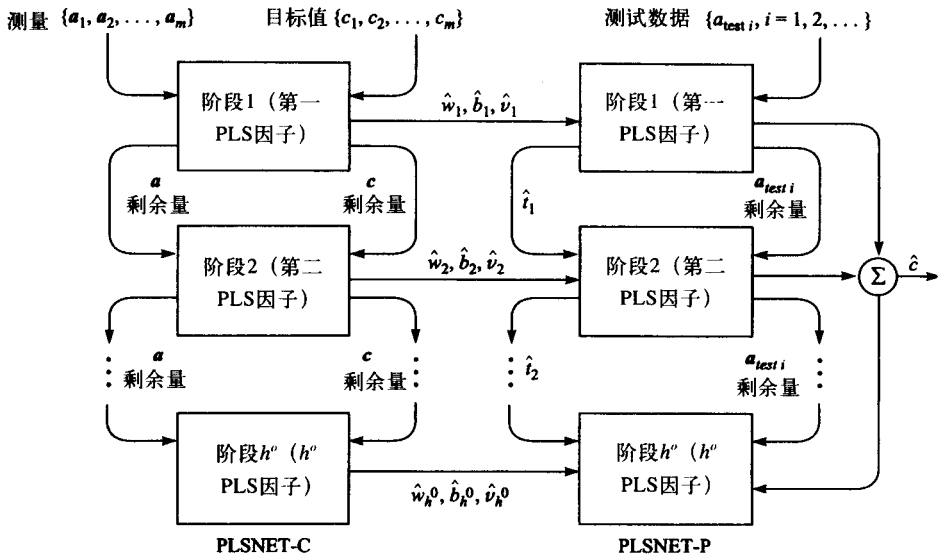



图9-19　PLSNET-C以及PLSNET-P通过突触权值$\{\hat{w}_h, \hat{b}_h, \hat{v}_h\}$其中$h = 1, 2, \cdots, h^o$耦合，在训练期间由PLSNET-C求取

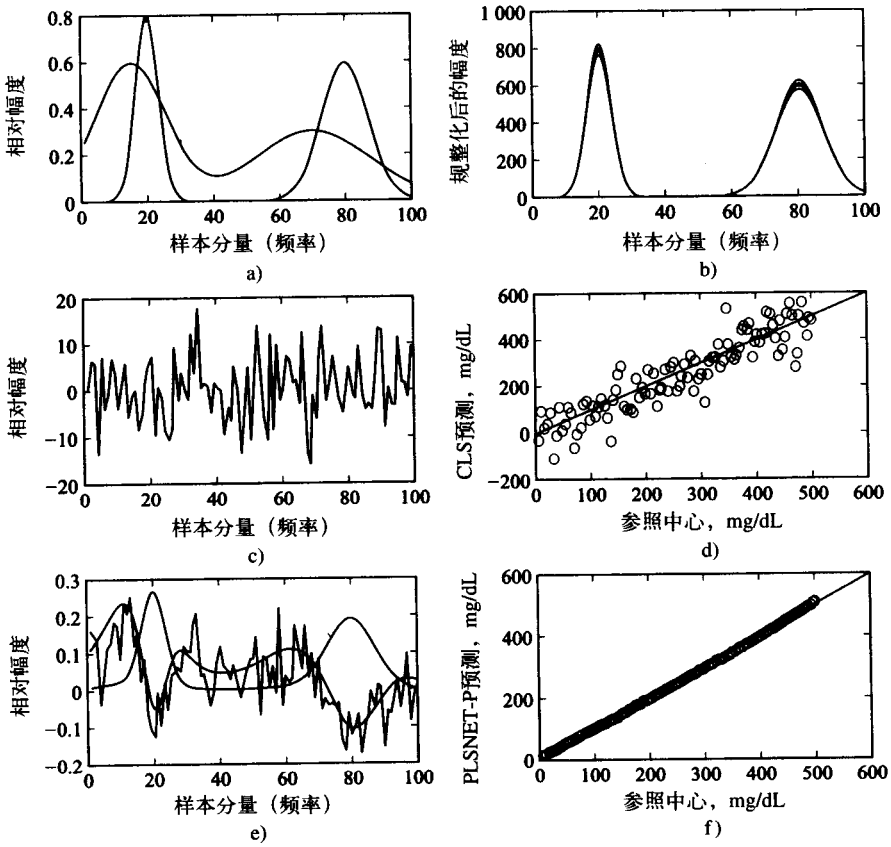



图9-20　a) 用于产生200个模拟NIR谱的基本谱分量；b) 5个代表模拟NIR谱；c) CLS校正模型；d) CLS浓度预测；e) 由PLSNET-C求取的前3个PLS权值装载向量；f) PLSNET-P中心预测

在理想条件下，根据知名的Lambert-Beer定律[119]在一个含水的解中吸收量与感兴趣的分析物（分量）浓度之间存在一个线性关系。但是，在许多情况下存在着一个近线性（稍微的非线性）关系。因此，这广泛分布的200个参考浓度值由下式产生

$$c_i = 10^3 \tan^{-1}\left(10^{-4}\sum_{j=1}^{n=100} a_{ij}^{p}\right) \tag{9-138}$$

其中$i = 1, 2, \cdots, m = 200$，模拟的纯分量NIR吸收谱包含在$\boldsymbol{A}^p \in \Re^{m\times n}$中（$m$个谱以及$n$个谱分量或者频率）。

因此，200个模拟合成谱包含200个感兴趣的谱，它们是确定的并与其相关的参考值（即，*参考浓度*）[132]相联系，其中存在介于纯分量吸收谱和相关的参考浓度之间的非线性关系，参考中心高度支配模糊分量，即水吸收、随机噪声以及基线变化。200个复合谱以及以升序排列相关的参考浓度值划分成两个相等的数据集，$\{\boldsymbol{A}_{\text{train}}, \boldsymbol{c}_{\text{train}}\}$以及$\{\boldsymbol{A}_{\text{test}}, \boldsymbol{c}_{\text{test}}\}$，每一组包含100个谱以及中心值。训练数据集看作奇数样本，测试集看作偶数样本。一个CLS模型（$\hat{\boldsymbol{b}}_{f\,\text{CLS}}$）是从$\hat{\boldsymbol{b}}_{f\,\text{CLS}} = (\boldsymbol{A}^T_{\text{train}}\boldsymbol{A}_{\text{trian}})^{-1}\boldsymbol{A}^T_{\text{train}}\boldsymbol{c}_{\text{train}}$中发展来的。图9-20c描述了CLS校正模型向量分量作为样本分量（或模拟频率）的函数。如同在图9-20c中看到的，校正模型是非常不稳定的。因此，可以预料基于这一模型的预测效果相对较差。将测试数据吸收矩阵$\boldsymbol{A}_{\text{test}}$投影到CLS校正模型$\hat{\boldsymbol{b}}_{f\,\text{CLS}}$，相关的测试浓度（$\boldsymbol{c}_{\text{test}}$）可以得到预测，即$\hat{\boldsymbol{c}}_{\text{testCLS}} = \boldsymbol{A}_{\text{test}}\hat{\boldsymbol{b}}_{f\,\text{CLS}}$。利用式（9-109）的表达式，SEP计算得到67.08mg/dL，CLS测试预测如图9-20d所示。

PLSNET-C使用$\{\boldsymbol{A}_{\text{train}}, \boldsymbol{c}_{\text{train}}\}$来训练，有3个阶段（对应于3个PLSR因子），从式（9-136）中的学习率参数$\mu_w$的计算值是$1.1205\times10^{-8}$，$\mu_b = \mu_v = 0.05\mu_w$，满足不等式（9-137）。在6000个训练回合之后PLSNET-C收敛，在图9-20e中描述了3个权值装载向量。观察图9-20e，明显可以发现只有前两个PLSR因子应该保留，以使用PLSNET-P来预测。图9-20e描述了第三个权值装载向量（对比前两个信号相对不稳定的信号）与噪声相关，因此不应该保留。只有前两个PLSR因子的保留已经使用独立确认和交叉确认方法证明了。使用测试集$\{\boldsymbol{A}_{\text{test}}, \boldsymbol{c}_{\text{test}}\}$，再使用PLSNET-P产生浓度预测，其中由PLSNET-C求取的突触权值系列$\{\hat{\boldsymbol{w}}_h, \hat{\boldsymbol{b}}_h, \hat{v}_h\}$，$h = 1, 2, (h^o)$，（即，前两个PLSR因子）。使用式（9-109）的表达式，SEP计算为5.16mg/dL，这大约比使用CLS方法的SEP计算小13倍。图9-20f描述了PLSNET-P预测。将图9-20a所描述的模糊分量与图9-20e所描述的第一PLSR权值装载向量（即，与模糊向量相似的向量）进行对比是很有趣的。这是不一致的，事实上，利用PLSR（或者PLSNET-C）从数据中求取信息是典型的，因为$\hat{\boldsymbol{w}}_1$是对模糊分量的一阶近似。在PLSNET-C中从阶段1到阶段2，与第一权值装载向量$\hat{\boldsymbol{w}}_1$相关的计分（或者隐式变量）向量$\hat{\boldsymbol{t}}_1$以及装载向量$\hat{\boldsymbol{b}}_1$用于产生第一*谱残留*，见图9-16。因此，对于模糊分量的近似值从谱数据中剔除，这一过程将持续到PLSNET-C的下一阶段。PLSNET相比CLS能够为测试数据更好地预测浓度，这是由于PLSR因子分析能力。也就是说，在PLSNET中，100个中只有2个因子为预测而保留，而CLS使用所有包括与噪声相关的100个因子，这将增大CLS相对于PLSNET的预测误差。

448
~
449

## 9.7 鲁棒PLSR：一种神经网络方法

如9.6节所述，非线性PLSR神经网络已经得到开发[129, 130]。这些网络包括典型的非线性激活函数，因此，执行训练过程可以完成一个非线性映射。但是，这里提出的鲁棒PLSR方法是基于与前面描述相同的*线性*PLSNET结构，而学习规则是基于Ham和McDowall[133-137]开发的非二次代价函数。也就是说，使用统计误差代价函数产生的比二次方要少，后者允许

考虑与输入相关的高阶统计。当经验数据包含冲击和有色噪声以及出格点时，PLSR的泛化用来鲁棒结果。由Karhunen和Joutsensalo[41]（对于鲁棒PCA）以及Cichocki和Unbehauen[38]（对于鲁棒的求解线性方程组）开发的鲁棒性技术可以直接应用于PLSNET。在加权函数假设为二次方的（即，均方误差标准）且加权矩阵假定为单位矩阵，9.6节的PLSNET学习规则可以复原的情况下，从某种意义上，用于鲁棒PLSR神经网络的学习规则的结果是一般的。后面提出的几个非线性加权函数可以用于进行鲁棒PLSR。有一些与在9.3.6节提出的对于鲁棒PCA的情况是一致的。

我们首先对权值装载向量$\hat{\boldsymbol{w}}_h$ ($h = 1, 2, \cdots, h^o$。其中$h^o$是PLSR因子的最佳数目）推导鲁棒
450 PLSNET学习规则。对于权值装载向量统计误差代价函数可以写成如下形式

$$\bar{L}_w(\boldsymbol{w}_h) = \mathbf{1}^T \boldsymbol{S}_w \mathrm{E}\{f(\boldsymbol{e}_{a_i}^w)\} \tag{9-139}$$

其中

$$\boldsymbol{e}_{a_i}^w = \boldsymbol{a}_i - c_i \boldsymbol{w}_h \tag{9-140}$$

从式（9-131）中，$\boldsymbol{e}_{a_i}^w \in \Re^{n\times 1}$，其中$\boldsymbol{a}_i \in \Re^{n\times 1}$是单一测量或者观察，$i = 1, 2, \cdots, m$（其中$m$是测量或者观察的总数），$c_i \in \Re$是与测量或者观察相关的目标值，$\boldsymbol{S}_w \in \Re^{n\times n}$是正定的（$\boldsymbol{S}_w > 0$）、对称（$\boldsymbol{S}_w^T = \boldsymbol{S}_w$）加权矩阵，在式（9-139）中$f(\cdot)$是一个合适的非线性加权函数（凸函数或者接近凸函数）以及$\mathbf{1}^T = [1, \cdots, 1] \in \Re^{1\times n}$单位向量。权值矩阵$\boldsymbol{S}_w$是引入来允许不同地为误差向量$\boldsymbol{e}_{a_i}^w$元素加权的。采用一种最速下降方法推导PLSR权值装载向量的离散时间鲁棒学习规则，即

$$\hat{\boldsymbol{w}}_h(k+1) = \hat{\boldsymbol{w}}_h(k) - \mu_w \nabla_w \bar{L}_w(\boldsymbol{w}_h) \tag{9-141}$$

我们发现必须计算梯度$\nabla_w \bar{L}_w(\boldsymbol{w}_h)$。当式（9-139）的梯度相对于$\boldsymbol{w}_h$计算时，只考虑瞬时值。因此，期望运算符$\mathrm{E}\{\cdot\}$被丢掉，结果是

$$\nabla_w \bar{L}_w(\boldsymbol{w}_h) = \frac{\partial \bar{L}_w(\boldsymbol{w}_h)}{\partial \boldsymbol{w}_h} = -c_i \boldsymbol{S}_w g(\boldsymbol{e}_{a_i}^w) = -c_i \boldsymbol{S}_w g(\boldsymbol{a}_i - c_i \hat{\boldsymbol{w}}_h) \tag{9-142}$$

其中$g(t) = \mathrm{d}f(t)/\mathrm{d}t$。使用式（9-141）以及式（9-142），权值装载向量的瞬时鲁棒PLSNET学习规则可以得到如下

$$\hat{\boldsymbol{w}}_h(k+1) = \hat{\boldsymbol{w}}_h(k) + \mu_w \boldsymbol{S}_w c_i g[\boldsymbol{a}_i - c_i \hat{\boldsymbol{w}}_h(k)] \tag{9-143}$$

其中$\mu_w > 0$是学习率参数，且对于线性PLSNET的情况，可以根据式（9-136）来设置。

在我们讨论对于装载向量和回归系数的两个瞬时鲁棒学习规则之前，从式（9-143）我们观察发现：（1）如果$\boldsymbol{S}_w = \boldsymbol{I}_{n\times n}$以及$f(t) = 1/2t^2$（用于线性PLSNET的二次方加权函数，即，均方误差标准），那么对于线性PLSNET的情况，式（9-143）与式（9-135）是相等的。（2）将权矩阵$\boldsymbol{S}_w$包含在式（9-143）中，学习规则可以看作是加权最小二乘[38]。（3）如果$\boldsymbol{S}_w = \mathrm{diag}[1/\sigma_i^2]$，其中对于$i = 1, 2, \cdots, n$，$\sigma_i^2$是测量矩阵列向量的方差，即$\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_m]^T \in \Re^{m\times n}$，如9.4节[见式（9-93）]所描述的，这就是9.5节对于PLSR1校正算法所讨论的方差规整处理。在这种情况下，矩阵$\boldsymbol{S}_w$将唯一一次应用于完成方差规整。但是，如果不是所有的测量数据，即$\boldsymbol{A}$认为是先验的，那么这一过程将不会执行。（4）如果$\boldsymbol{S}_w = \boldsymbol{I}$，在式（9-143）中项$c_i g(\boldsymbol{a}_i - c_i \hat{\boldsymbol{w}}_h)$是式（9-139）中的统计误差代价函数梯度的负值。但是，如果$\boldsymbol{S}_w$是与$\bar{L}_w(\boldsymbol{w}_h) = \mathbf{1}^T \mathrm{E}\{f(\boldsymbol{e}_{a_i}^w)\}$相关的黑塞矩阵（参照A.3.5节）的逆，即$\boldsymbol{H}_w = \nabla_w^2 \bar{L}_w(\boldsymbol{w}_h)$，同时如前所述，梯度将是$\nabla_w \bar{L}_w(\boldsymbol{w}_h) = -c_i g(\boldsymbol{a}_i - c_i \hat{\boldsymbol{w}}_h)$，那么式（9-143）可以看作是牛顿方法（参照A.5.3节）。在这种情况下，式（9-143）
451 可以写成

$$\hat{\boldsymbol{w}}_h(k+1)=\hat{\boldsymbol{w}}_h(k)+\mu_w \boldsymbol{H}_w^{-1}\nabla w\bar{L}_w(\boldsymbol{w}_h) \tag{9-144}$$

装载向量$\hat{\boldsymbol{b}}_h\in\Re^{n\times 1}$和标量回归系数$\hat{\upsilon}_h\in\Re$（$h=1,2,\cdots,h^o$）的瞬时鲁棒PLSNET学习规则可以使用与推导权值装载向量学习规则的相似过程来处理。装载向量的统计误差代价函数可以写成

$$\bar{L}_b(\boldsymbol{b}_h)=\mathbf{1}^T\boldsymbol{S}_b\mathrm{E}\{f(\boldsymbol{e}_{a_i}^b)\} \tag{9-145}$$

其中$\boldsymbol{S}_b\in\Re^{n\times n}$，$\boldsymbol{S}_b>0$，$\boldsymbol{S}_b^T=\boldsymbol{S}_b$，$\boldsymbol{e}_{a_i}^b\in\Re^{n\times 1}$可以写成

$$\boldsymbol{e}_{a_i}^b=\boldsymbol{a}_i-\hat{t}_{hi}\boldsymbol{b}_h \tag{9-146}$$

（见表9-1），以及

$$\hat{t}_{hi}=\hat{\boldsymbol{w}}_h^T\boldsymbol{a}_i \tag{9-147}$$

就是在9.5节中解释的测量强度。使用式（9-145），对于装载向量的瞬时鲁棒PLSNET学习规则有如下形式

$$\hat{\boldsymbol{b}}_h(k+1)=\hat{\boldsymbol{b}}_h(k)+\mu_b\boldsymbol{S}_b\hat{\boldsymbol{t}}_{hi}g[\boldsymbol{a}_i-\hat{t}_{hi}\hat{\boldsymbol{b}}_h(k)] \tag{9-148}$$

其中$\mu_b>0$以及$\mu_b\leqslant\mu_w$。标量回归系数的统计误差代价函数可以写成

$$\bar{L}_\upsilon(\upsilon_h)=s_\upsilon\mathrm{E}\{f(e_{c_i}^\upsilon)\} \tag{9-149}$$

其中$s_\upsilon>0$，以及$e_{c_i}^\upsilon\in\Re$可以写成

$$e_{c_i}^\upsilon=c_i-\upsilon_h\hat{\boldsymbol{t}}_{hi} \tag{9-150}$$

（见表9-1）。利用式（9-149），回归系数的瞬时鲁棒PLSNET学习规则有如下形式

$$\hat{\upsilon}_h(k+1)=\hat{\upsilon}_h(k)+\mu_\upsilon s_\upsilon\hat{\boldsymbol{t}}_{hi}g[c_i-\hat{\boldsymbol{t}}_{hi}\hat{\upsilon}_h(k)] \tag{9-151}$$

其中$\mu_\upsilon>0$，$\mu_\upsilon\leqslant\mu_b$。因此，线性PLSNET的不等式（9-137）就应用于鲁棒PLSNET。

在我们提出鲁棒PLSNET算法之前，需要讨论一下加权函数$f(\cdot)$的选择问题。有几个加权函数适合于鲁棒PLSNET校正。在这里提出的加权函数集并不包含在内；但是讨论的特殊加权函数是一般用于鲁棒处理[38, 41, 138-140]的非线性函数。特别是，M估计量[139, 140]加权函数已经应用于使用神经网络[138]的鲁棒主成分估计，同时也被Chen和Jain[141]应用于对于鲁棒函数近似的反向传播神经网络。表9-2描述了加权函数$f(t)$以及导数$g(t)=\mathrm{d}f(t)/\mathrm{d}t$。图9-21描述了表9-2中的每一个$\beta=1$的加权函数，以及图9-22描述了每一个加权函数的导数。从图9-21中可以发现对于每一个用于鲁棒PLSNET（$\beta=1$）的加权函数，函数提出的特殊的加权（$t\in[-1,1]$）实际是二次的。但是，在$t$的这个范围以外，随$t$增长加权函数低于二次的产生较少的。因此，错误的训练数据（出格点）重要性显著地降低，或者基本上彻底滤出（见图9-22），假设训练PLSNET校正网络的鲁棒性，并在冲击或者有色噪声存在时改善相对于二次加权情况的性能。用于鲁棒PLSNET的加权函数的选择依赖于训练数据的特征。因此，特殊的加权函数一般靠经验选择。但是，在调节$\beta$值的时候需要注意。例如，如果$\beta$值对于对数函数设置得太小（见表9-2），由函数提供的加权

452

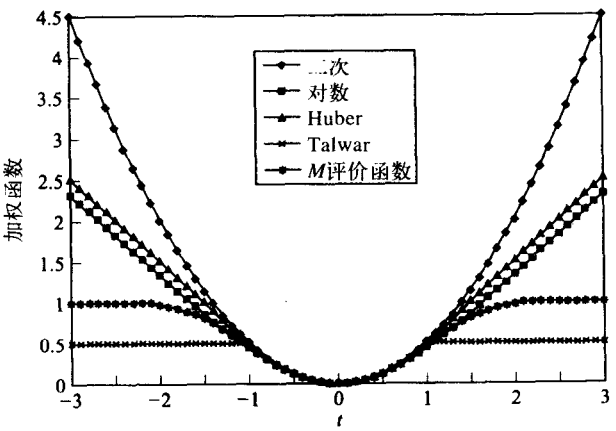



图9-21　用于鲁棒PLSNET（$\beta=1$）的加权函数

会彻底地（或几乎）拒绝所有的训练输入。因此，神经网路根本就不会训练。相反，如果$\beta$设置得过高，加权会造成二次的情形，鲁棒性会消失。$\beta$值的典型范围是在$0.7 \leqslant \beta \leqslant 8$之间选择。

**表9-2　鲁棒PLSNET的加权函数**

| 加权函数$f(t)(\beta > 0)$ | 微分 $g(t) = \dfrac{\mathrm{d}f(t)}{\mathrm{d}t}$ |
|---|---|
| 对数函数 $f_L(t) = \beta^2 \ln\cosh\dfrac{t}{\beta}$ | $g_L(t) = \beta\tanh\dfrac{t}{\beta}$ |
| Huber函数 $f_H(t) = \begin{cases} \dfrac{t^2}{2} & \text{对于} \lvert t\rvert \leqslant \beta \\ \beta\lvert t\rvert - \dfrac{\beta^2}{2} & \text{对于} \lvert t\rvert > \beta \end{cases}$ | $g_H(t) = \begin{cases} -\beta & \text{对于} t < -\beta \\ t & \text{对于} \lvert t\rvert \leqslant \beta \\ \beta & \text{对于} t > \beta \end{cases}$ |
| Talwar函数 $f_T(t) = \begin{cases} \dfrac{t^2}{2} & \text{对于} \lvert t\rvert \leqslant \beta \\ \dfrac{\beta^2}{2} & \text{对于} \lvert t\rvert > \beta \end{cases}$ | $g_T(t) = \begin{cases} 0 & \text{对于} t < -\beta \\ t & \text{对于} \lvert t\rvert \leqslant \beta \\ 0 & \text{对于} t > \beta \end{cases}$ |
| $M$评价函数 $f_M(t) = \dfrac{1 - \mathrm{e}^{-\beta t^2}}{1 + \mathrm{e}^{-\beta t^2}}$ | $g_M(t) = \dfrac{4\beta t\mathrm{e}^{-\beta t^2}}{(1 + \mathrm{e}^{-\beta t^2})^2}$ |

基于逆建模的鲁棒PLSR另一个公式已经有所发展[137]。这一方法相比前向建模方法，其主要优点是较少的计算需求。

鲁棒PLSNET前向建模校正处理可以总结成下面的算法。

**鲁棒PLSNET校正算法**

训练数据集由测量/目标值对组成：

$$\{\boldsymbol{a}_i, c_i\} \quad i = 1, 2, \cdots, m$$

**步骤1**　选择$h^o$（最优因子数目），设置$h = 1$。

**步骤2**　计算学习率参数：

$$\mu_w = \eta\left(\sum_{i=1}^{m} c_i^2\right)^{-1} \quad \eta < 1 \quad \mu_b = \frac{\mu_w}{\alpha} \quad \mu_v = \frac{\mu_b}{\alpha} \qquad (\text{一般}\alpha = 2)$$

**步骤3**　为权值矩阵$\boldsymbol{S}_w$、$\boldsymbol{S}_b$以及$s_v$设置相应的数值。

**步骤4**　选择一个适当的加权函数$f(t)$，同时设置$\beta$值。

**步骤5**　初始化权值$\hat{\boldsymbol{w}}_h$, $\hat{\boldsymbol{b}}_h$，以及$\hat{v}_h$。

**步骤6**　置$i = 1$。

**步骤7**　(a) $\hat{\boldsymbol{w}}_h \leftarrow \hat{\boldsymbol{w}}_h + \mu_w \boldsymbol{S}_w c_i\, g(\boldsymbol{a}_i - c_i\hat{\boldsymbol{w}}_h)$

(b) $\dfrac{\hat{\boldsymbol{w}}_h \leftarrow \hat{\boldsymbol{w}}_h}{\lVert\hat{\boldsymbol{w}}_h\rVert_2}$

(c) $\hat{t}_{hi} = \hat{\boldsymbol{w}}_h^T \boldsymbol{a}_i$

(d) $\hat{\boldsymbol{b}}_h \leftarrow \hat{\boldsymbol{b}}_h + \mu_b \boldsymbol{S}_b \hat{t}_{hi}\, g(\boldsymbol{a}_i - \hat{t}_{hi}\hat{\boldsymbol{b}}_h)$

(e) $\hat{v}_h \leftarrow \hat{v}_h + \mu_v s_v \hat{t}_{hi}\, g(c_i - \hat{t}_{hi}\hat{v}_h)$

如果$i = m$，转到步骤8，否则$i \leftarrow i + 1$，然后转到（a）。

**步骤8**　如果$h = h^o$，转到步骤9，否则$h \leftarrow h + 1$，然后转到步骤6。

**步骤9**　如果达到收敛，停止；否则，置$h = 1$，然后转到步骤6。　□

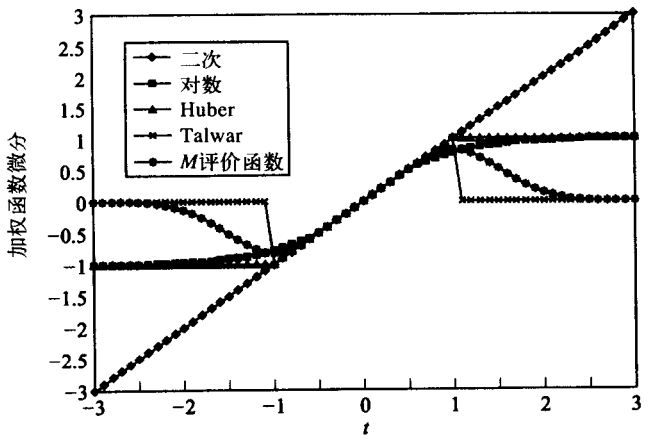



图9-22 图9-21中加权函数的导数（$\beta = 1$）

使用独立测数据集，通过计算SEP来确定收敛，见式（9-109）。

在前面描述的鲁棒PLSNET校正算法中，如果权值矩阵选择为黑塞矩阵的逆，那么可以使用的最好加权函数是对数函数。对于比截止参数$\beta$值幅度大的值，Huber和Talwar加权函数推出二阶导数为零。$M$估计量加权函数导出一个复杂的二阶导数的表达式。即使牛顿的方法提供了自身自适应学习率，标准学习率参数$\{\mu_w, \mu_b, \mu_v\}$应该包括在学习规则中，因为一次只有一个测量在进行。也就是说，牛顿方法典型地应用于“批量-建模”操作中。但是，既然PLSNET一次只处理一个测量，一个完全解适于一次测量是不理想的。这与在3.4.1节应用于前馈多层感知器的反向传播共轭梯度方法的情况是相同的原理。

## 习题

9.1 假设式（9-25）以离散时间向量形式，设计一个单节点神经网络来完成Oja正规化Hebb学习规则，从而估计第一主特征向量。从一个零平均，单位方差，正态（高斯）分布产生两组独立的5000个随机数据。标定第一序列的值以便于方差为5（即，序列中的数乘以）。使用式（9-19）的表达式，从随机数的两个序列中估计协方差矩阵。计算协方差矩阵的特征值以及特征向量，利用标准特征值程序，例如MATLAB的`eig`函数。使用Oja离散时间形式的正规化Hebb学习规则，计算第一主特征向量。将初始突触权值向量元素置为随机值。使用适当的固定学习率参数的估计以及式（9-28）给出的自适应算法。对于收敛必需的训练回合的总数来对比两个结果。对比神经网络结果与使用标准特征值程序计算的特征向量（与最大特征值相对应）。估计与使用Oja的正规Hebb学习规则估计的第一主特征向量相关的特征值。提示：如例9-1中的证明，使用MATLAB中的`var`（方差）函数，见式（9-29）。

453 ~ 455

9.2 考虑由下式随机差分方程给出的一个宽平稳一阶离散时间马尔可夫过程

$$x(k) = \alpha x(k-1) + w(k)$$

其中$\alpha = 0.9$, $x, w \in \Re^{3\times 1}$，$w$是带有如下协方差矩阵的零平均高斯白噪声。

$$C_v = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}$$

(a) 计算协方差矩阵$\boldsymbol{C}_x = \mathrm{E}\{\boldsymbol{x}\boldsymbol{x}^T\}$。

(b) 计算$\boldsymbol{C}_x$的理论特征值以及特征向量。

(c) 建构一个神经网络以实现在式（9-45）中提出的向量-矩阵形式的Karhunen-Oja的对称子空间学习规则的离散时间形式。使用适当固定的学习率参数以及式（9-48）的自适应算法。使用你开发的网络求取两维主子空间。求解介于“Oja的子空间”以及理论子空间之间的角度，即由部分(b) 计算的前两个特征向量张成的子空间。

9.3　建构一个神经网络来实现Sanger的广义Hebb算法。式（9-53）给出了标量学习规则的离散时间形式。使用你开发的网络为在问题9.1产生的数据求取两维主子空间[即，前（也只有）两个主特征向量]。使用适当固定的学习率参数以及式（9-48）给出的自适应算法。求解介于“Sanger的子空间”以及理论子空间之间的角度。

9.4　在问题9.2(c) 以及问题9.3中使用自适应学习率参数以及带有遗忘因子（$\gamma$）的试验。观察对于收敛速度的影响。

9.5　考虑信号$\boldsymbol{x}(k) = \boldsymbol{s}(k) + \boldsymbol{w}(k), \quad k = 0, 1, 2, \cdots$

其中

$$\boldsymbol{s} = \left[\sin\left(\frac{2\pi}{20}1\right), \sin\left(\frac{2\pi}{20}2\right), \cdots, \sin\left(\frac{2\pi}{20}20\right)\right]^T \in \Re^{20\times 1}$$

以及

$$\boldsymbol{w} \in \Re^{20\times 1}$$

是包含如下协方差的零平均高斯白噪声。

$$\boldsymbol{C}_w = \mathrm{diag}(\sigma_w^2, \sigma_w^2, \cdots, \sigma_w^2) \in \Re^{20\times 20} \qquad \text{其中} \sigma_w^2 = 0.1$$

(a) 使用Oja单节点学习规则，求取$\boldsymbol{C}_x$的第一主特征向量。使用式（9-28）中给出的学习率参数的自适应算法。

456　(b) 使用信号$\boldsymbol{s}$来对比估计的第一主特征向量，然后解释发生了什么。

9.6　非线性迭代局部最小二乘（NIPALS）方法是一种迭代的求取正定矩阵$\boldsymbol{C}_x$的有不同特征值的特征向量方法。

(a) 令$\boldsymbol{e}_1$是与$\boldsymbol{C}_x$的最大特征值相对应的特征向量。

考虑如下迭代过程

$$\boldsymbol{w}(k+1) = \boldsymbol{C}_x\boldsymbol{w}(k)$$

$$\boldsymbol{w}(k+1) = \frac{\boldsymbol{w}(k+1)}{\|\boldsymbol{w}(k+1)\|_2}$$

其中$k = 0, 1, 2, \cdots$，$\boldsymbol{w}$是一个合适维数的随机向量。

证明 $\lim_{k\to\infty} \boldsymbol{w}(k) = \boldsymbol{e}_1$

(b) 证明以下可以用于求取第二主特征向量的降阶技术：

$$\boldsymbol{C}_x^{(D_1)} = (\boldsymbol{I} - \boldsymbol{e}_1\boldsymbol{e}_1^T)\boldsymbol{C}_x(\boldsymbol{I} - \boldsymbol{e}_1\boldsymbol{e}_1^T)$$

其中$D_1$是$\boldsymbol{C}_x$的第一次降阶的设计器。这称作$\boldsymbol{C}_x$的降阶变换。

注意：部分(a) 与部分(b) 一起组成了NIPALS方法，以及部分(b) 中的降阶过程的延续来求取高阶主特征向量。

9.7　建构一个神经网络用以实现式（9-58）中的标量形式Oja随机梯度上升（SGA）学习规则的离散时间形式。使用式（9-48）给出的学习率参数的自适应算法。

(a) 利用零平均单位方差正态分布，产生3组独立的1000个随机数。规定第二序列的值范围以便方差为0.01。规定第三序列的值范围以便其方差为0.001。

(b) 使用Oja的SGA算法估计所有与在部分(a)产生的随机数据相关的协方差矩阵的主特征向量。假设随机初始突触权值。这一过程可能需要几个训练回合，即几个1000个随机向量的表达。对于这一训练神经网络的方法，初始学习率参数可以对于每一个训练回合重新计算，其中前面训练回合应用的突触权值作为下一训练回合的初始权值。

(c) 使用Sanger的GHA重复部分(b)。对比你得到的结果，特别是收敛必需的训练时间总量。

9.8 在9.3.2节中，Karhunen-Oja对称子空间学习规则是从式（9-38）的均方误差标准推导而来。结果离散时间向量矩阵学习规则在式（9-45）中提出。均方误差标准可以修改以包含一个正定对称加权矩阵$\boldsymbol{S}\in\Re^{n\times n}$，即$L(\boldsymbol{W})=1/2\boldsymbol{e}^T\boldsymbol{S}\boldsymbol{e}$。

(a) 使用$L(\boldsymbol{W})=1/2\boldsymbol{e}^T\boldsymbol{S}\boldsymbol{e}$作为误差代价函数推导向量矩阵形式的离散时间加权对称子空间学习规则。

(b) 从部分(a)推导的结果，令$\boldsymbol{S}=\boldsymbol{I}$，同时利用合理的假定证明式（9-45）中的学习规则 457
的结果。因此，加权对称子空间学习规则是一般的情况。

9.9 如同9.3.6节所述，标准线性PCA可以起源于一个线性变换方差$E\{[\boldsymbol{w}_i^T\boldsymbol{x}]^2\}$最大化的优化解$(i=1,2,\cdots,m)$，或者正交约束即$\boldsymbol{W}\boldsymbol{W}^T=\boldsymbol{I}\in\Re^{m\times m}$条件下的，最大化线性网络的输出。使用方差最大化方法推导求取第一主特征向量$\boldsymbol{w}_1$的离散时间学习规则。你所推导的学习规则与式（9-25）的结果应该是一致的，是使用代价函数的表达误差公式的最小化来得到的。提示：将（要求最大化的）代价函数公式化如$L(\boldsymbol{w}_1)=\sigma_{y_1}^2-\lambda_1(\boldsymbol{w}_1^T\boldsymbol{w}_1-1)$，其中$y_1=\boldsymbol{w}_1^T\boldsymbol{x}$，$\sigma_{y1}^2=\boldsymbol{w}_1^T\boldsymbol{C}_x\boldsymbol{w}_1$以及$\boldsymbol{C}_x=E[\boldsymbol{x}\boldsymbol{x}^T]$。这一代价函数的公式化是基于约束最优化问题的拉格朗日乘子方法(参照A.6.2节)，其中拉格朗日乘子是与第一主特征值$\lambda_1$相关的，同时约束条件是$\boldsymbol{w}_1^T\boldsymbol{w}_1=1$。同样，因为第一主特征向量是使用学习规则来估计的，因此只考虑输入的瞬时值。

9.10 设计一个APEX神经网络来实现估计前馈以及侧向权值的两个学习规则。使用在9.3.5节概述的APEX算法。利用固定学习率参数，基于在问题9.7(a)中产生的数据来测试你的神经网络。如同Oja的SGA以及Sanger的GHA方法，这一神经网络同样典型地需要训练输入向量的重复提交。将你的结果与问题9.7中得到的结果相对比。可以得到什么样的结论呢？

9.11 可以发现网络经过训练以后，Sanger的GHA方法以及Kung以及Diamantaras的APEX算法可以得到相同的稳态点。请证明之。

9.12 (a)在9.5节中提出了PLSR1校正算法。编写一个计算机程序，适宜在MATLAB中，来实现校正过程（为数据的平均中心以及方差规整，忽略算法中的第一步）。PLSR因子的数目应该是你的程序中的一个变量。

(b) 写出两个附加的计算机程序来分别实现两个在9.5节中提出的PLSR1预测算法，即*预测方法1*以及*预测方法2*。

9.13 常常存在几个感兴趣的分量包含于由一个特殊过程产生的测量数据中。如例9.4所述，对于合成近红外数据，由神经过程（PLSNET）实现的单一分量PLSR1算法能够获取满足于单一目标值（参考中心）的信息，同时推导出较之CLS方法更好的预测性能。当希望增加兴趣分量（几组目标值）而使用PLSR时，PLSR1方法是不能直接应用的。多分量PLSR方法已经有所发展[7, 10, 13]。但是一般情况下，当使用这些方法来开发一个可

458 以预测多分量的单一模型时，性能的牺牲是可以预知的。作为一个选择，可以使用PLSR1算法，但是要为每个分量单独使用（即，不同组的目标值）。换句话说，对于每一个感兴趣的分量，存在目标值（参考值），PLSR1算法可以独立地使用，对于每一个连续的兴趣分量产生一组独立的权值装载向量、装载向量以及回归系数。

(a) 为PLSR设计一个神经网络（即，PLSNET）。即，为校正阶段开发PLSNET-C，用于预测的PLSNET-P。为它们分别编写MATLAB函数。

(b) 编写一个可以执行CLS的计算机程序。

(c) 在McGraw-Hill站点，在http://www.mhhe.com/engcs/electrical/ham下载为这个问题指定的数据。你将发现两组的数据：(1) 与例9.4中使用相同的数据（这指定为P9_13_Date_Set_1），即训练以及测试数据；(2) 另一个数据集（P9_13_Data_Set_2）由使用与两组目标值相联系的依赖变量块来训练的独立变量块矩阵，以及另一个等价测试数据集组成。在第二个数据集中的测量数据与第一个数据集中的是相同的，除了一个附加的兴趣分量是新引入的。因此，在训练以及测试数据集中存在着第二个目标值列。

(d) 使用在部分(a) 开发的PLSNET，重复例9.4中描述的结果，使用只包含数据中单分量的第一数据集。你将得到一组权值装载向量、装载向量以及回归系数。绘制出测试数据的预测回归结果，计算SEP（与例9.4中的数值相等）。问题9.12的PLSR1校正以及预测计算机程序可以用于核实必须保留的PLSR因子的数目。

(e) 使用你的部分(b) 的CLS计算机程序开发CLS校正模型，并且产生对测试数据的预测。在一个独立的图上描绘回归结果，计算SEP。

(f) 使用包含两个感兴趣得分量以及PLSNET的第二数据集，选择合理数目的PLS因子，然后产生合适两组权值装载向量、装载向量以及回归系数。将开发两个校正模型。PLSR1校正模型以及问题9.12的预测计算机程序可以再一次用于核实需要保留的PLSR因子的数目。为两个分量产生预测，使用与特殊的感兴趣分量相关的测试数据以及合适的权值装载向量、装载向量以及回归系数。为独立的分量分别绘制回归结果。为分量分别计算SEP。

(g) 利用部分(b) 的CLS计算机程序开发两个用于两个不同分量的CLS校正模型，对两个测试数据集生成预测。在分别的图上画出预测的回归结果，并且对两个分量计算SEP。

(h) 对比由在部分(f) 以及部分(g) 得到的性能结果。你可以得到什么结论呢？

459 (i) 将在部分(f)中使用PLSNET得到的第一分量的预测性能与在部分(d)中得到的相对比。你会得到什么结论呢？

## 参考文献


1. D. F. Morrison, *Multivariate Statistical Methods*, 3rd ed., New York: McGraw-Hill, 1990.
2. S. Haykin, *Adaptive Filter Theory*, 2nd ed., Englewood Cliffs, NJ: Prentice-Hall, 1991.
3. I. T. Jolliffe, *Principal Component Analysis*, New York: Springer-Verlag, 1986.
4. N. Draper and H. Smith, *Applied Regression Analysis*, 2nd ed., New York: Wiley-Interscience, 1981.

5. H. Hotelling, "Analysis of a Complex of Statistical Variables into Principal Components," *Journal of Educational Psychology*, vol. 24, 1933, pp. 498–520.
6. K. V. Mardia, J. T. Kent, and J. M. Bibby, *Multivariate Analysis*, New York: Academic, 1979.
7. H. Martens and T. Naes, *Multivariate Calibration*, New York: Wiley, 1989.
8. B. R. Kowalski, ed., *Chemometrics—Mathematics and Statistics in Chemistry*, NATO ASI Series, Series C: Mathematical and Physical Sciences, vol. 138, Boston: D. Reidel, 1984.
9. F. M. Ham and I. Kostanic, "Partial Least-Squares: Theoretical Issues and Engineering Applications in Signal Processing," *Journal of Mathematical Problems in Engineering*, vol. 2, no. 1, 1996, pp. 63–93.
10. P. Geladi and B.R. Kowalski, "Partial Least-Squares Regression: A Tutorial," *Analytica Chimica Acta,* no. 185, 1986, pp. 1–17.
11. K. R. Beebe and B. R. Kowalski, "An Introduction to Multivariate Calibration and Analysis," *Analytical Chemistry*, vol. 59, no. 417, 1987, pp. 1007A–17A.
12. D.M. Haaland and E. V. Thomas, "Partial Least-Squares Methods for Spectral Analyses. 1. Relation to Other Quantitative Calibration Methods and the Extraction of Qualitative Information," *Analytical Chemistry*, vol. 60, no. 11, 1988, pp. 1193–202.
13. A. Lorber, L. E. Wagen, and B. R. Kowalski, "A Theoretical Foundation for the PLS Algorithm," *Journal of Chemometrics*, vol. 1, 1987, pp. 19–31.
14. S. D. Brown, "Chemometrics," *Analytical Chemistry*, vol. 62, 1990, pp. 84R–101R.
15. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Reading, MA: Addison-Wesley, 1992.
16. A. D. Whalen, *Detection of Signal in Noise*, New York: Academic, 1971.
17. G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis—Forcasting and Control*, 3rd ed., Englewood Cliffs, NJ: Prentice-Hall, 1994.
18. E. Oja, "A Simplified Neuron Model as a Principal Component Analyzer," *Journal of Mathematical Biology*, vol. 15, 1982, pp. 267–73.
19. P. Baldi and K. Hornik, "Neural Networks and Principal Component Analysis: Learning from Examples Without Local Minima," *Neural Networks*, vol. 2, 1989, pp. 53–8.
20. E. Oja and J. Karhunen, "On Stochastic Approximation of the Eigenvectors and Eigenvalues of the Expectation of a Random Matrix," *Journal of Mathematical Analysis and Applications*, vol. 104, 1985, pp. 69–84.
21. T. D. Sanger, "Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network," *Neural Networks*, vol. 2, 1989, pp. 459–73.
22. P. Földiak, "Forming Sparse Representations by Local Anti-Hebbian Learning," *Biological Cybernetics*, vol. 64, 1990, pp. 165–70.
23. P. Földiak, "Adaptive Network for Optimal Linear Feature Extraction," in *Proceedings of the International Joint Conference on Neural Networks*, Washington, DC, vol. 1, 1989, pp. 401–5.
24. F. M. Silva and L. B. Almeida, "A Distributed Decorrelation Algorithm," in *Neural Networks Advances and Applications*, ed. E. Gelenbe, Amsterdam: North-Holland, 1991, pp. 145–63.
25. S. Y. Kung, K. I. Diamantaras, and J. S. Taur, "Neural Networks for Extracting Pure/Constrained/Oriented Principal Components," in *SVD and Signal Processing*, ed. R. J. Vaccaro, Amsterdam: Elsevier Science, 1991, pp. 57–81.
26. E. Oja, "Principal Components, Minor Components and Linear Neural Networks," *Neural Networks*, vol. 5, 1992, pp. 927–35.
27. P. F. Baldi and K. Hornik, "Learning in Linear Neural Networks: A Survey,"

460

*IEEE Transactions on Neural Networks*, vol. 6, 1995, pp. 837–58.
28. L.-H. Chen and S. Chang, "An Adaptive Learning Algorithm for Principal Component Analysis," *IEEE Transactions on Neural Networks*, vol. 6, 1995, pp. 1255–63.
29. E. Oja, "Neural Networks, Principal Components, and Subspaces," *International Journal of Neural Systems*, vol. 1, 1989, pp. 61–8.
30. S. Y. Kung, K. I. Diamantaras, and J. S. Taur, "Adaptive Principal Component Extraction (APEX) and Applications," *IEEE Transactions on Signal Processing*, vol. 42, 1994, pp. 1202–17.
31. J. Rubner and P. Tavan, "A Self-Organizing Network for Principal Component Analysis," *Europhysics Letters*, vol. 10, no. 7, 1989, pp. 693–8.
32. R. Lenz and M. Österberg, "Computing the Karhunen-Loève Expansion with a Parallel, Unsupervised Filter System," *Neural Computation*, vol. 4, 1992, pp. 382–92.
33. R. H. White, "Competitive Hebbian Learning: Algorithm and Demonstrations," *Neural Networks*, vol. 5, 1992, pp. 261–75.
34. J. Karhunen and J. Joutsensalo, "Tracking of Sinusoidal Frequencies by Neural Network Learning Algorithms," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Toronto, Canada, 1991, pp. 69–84.
35. K. Hornik and C.-M. Kuan, "Convergence Analysis of Local Feature Extraction Algorithms," *Neural Networks*, vol. 5, 1992, pp. 229–40.
36. T. D. Sanger, "An Optimality Principle for Unsupervised Learning," in *Advances in Neural Information Processing Systems*, ed. D.S. Touretzky, Palo Alto, CA: Morgan Kaufmann, 1989, pp. 11–19.
37. A. Krogh and J. A. Hertz, "Hebbian Learning of Principal Components," in *Parallel Processing in Neural Systems and Computers*, eds. R. Eckmiller, G. Hartmann, and G. Hauske, Amsterdam: North-Holland, 1990, pp. 183–6.
38. A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, New York: Wiley, 1993.
39. J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Redwood City , CA: Addison-Wesley, 1991.
40. S. Haykin, *Neural Networks—A Comprehensive Foundation*, New York: Macmillan, 1994.
41. J. Karhunen and J. Joutsensalo, "Generalizations of Principal Component Analysis, Optimization Problems, and Neural Networks," *Neural Networks*, vol. 8, 1995, pp. 549–62.
42. E. Oja and J. Karhunen, "Nonlinear PCA: Algorithms and Applications," Technical Report: A18 (Otaniemi 1993), Helsinki University of Technology
461 (Laboratory of Computer and Information Sciences), Espoo, Finland, 1993.
43. J. Karhunen and J. Joutsensalo, "Representation and Separation of Signals Using Nonlinear PCA Type Learning," *Neural Networks*, vol. 7, 1994, pp. 113–27.
44. J. Joutsensalo, "Nonlinear Data Compression and Representation by Combining Self-Organizing Map and Subspace Rule," *Proceedings of IEEE International Conference on Neural Networks '94*, Orlando, FL, vol. 2, 1994, pp. 637–40.
45. F. Palmieri, "Hebbian Learning and Self-Association in Nonlinear Neural Networks," *Proceedings of IEEE International Conference on Neural Networks '94*, Orlando, FL, vol. 2, 1994, pp. 1258–63.
46. A. Sudjianto and M. H. Hassoun, "Nonlinear Hebbian Rule: A Statistical Interpretation," *Proceedings of IEEE International Conference on Neural Networks '94*, Orlando, FL, vol. 2, 1994, pp. 1247–52.

47. L. Xu, "Theories for Unsupervised Learning: PCA and Its Nonlinear Extensions," *Proceedings of IEEE International Conference on Neural Networks '94*, Orlando, FL, vol. 2, 1994, pp. 1252–7.
48. J. Karhunen, "Optimization Criteria and Nonlinear PCA Neural Networks," *Proceedings of IEEE International Conference on Neural Networks '94*, Orlando, FL, vol. 2, 1994, pp. 1241–6.
49. S. Y. Kung, *Digital Neural Networks*, Englewood Cliffs, NJ: Prentice-Hall, 1993.
50. R. Linsker, "Designing a Sensory Processing System: What Can Be Learned from Principal Component Analysis?" in the *Proceedings of the International Joint Conference on Neural Networks*, Washington, DC, vol. 2, 1990, pp. 291–7.
51. H. Köhnel and P. Taven, "The Anti-Hebbian Rule Derived from Information Theory," in *Parallel Processing in Neural Systems and Computers*, eds. R. Eckmiller, G. Hartmann, and G. Hauske, Amsterdam: North-Holland, 1990, pp. 187–90.
52. A. Carlson, "Anti-Hebbian Learning in a Non-Linear Neural Network," *Biological Cybernetics*, vol. 64, 1990, pp. 171–6.
53. L. Xu, E. Oja, and C. Y. Suen, "Modified Hebbian Learning for Curve and Surface Fitting," *Neural Networks*, vol. 5, 1992, pp. 441–57.
54. S. Becker, "Unsupervised Learning Procedures for Neural Networks," *International Journal of Neural Systems*, vol. 2, 1991, pp. 17–33.
55. J. A. Sirat, A Fast Neural Algorithm for Principal Component Analysis and Singular Value Decomposition," *International Journal of Neural Systems*, vol. 2, 1991, pp. 147–55.
56. Y. Chauvin, "Constrained Hebbian Learning: Gradient Descent to Global Minima in an *N*-Dimesional Landscape," *International Journal of Neural Systems*, vol. 2, 1991, pp. 35–46.
57. E. Oja, H. Ogawa, and J. Wangviwattana, "Principal Components Analysis by Homogeneous Neural Networks, Part I: The Weighted Subspace Criterion," *IEICE Transactions on Information and Systems* (Japan), vol. E75-D, 1992, pp. 366–75.
58. E. Oja, H. Ogawa, and J. Wangviwattana, "Principal Components Analysis by Homogeneous Neural Networks, Part II: Analysis and Extensions of the Learning Rule," *IEICE Transactions on Information and Systems* (Japan), vol. E75-D, 1992, pp. 376–82.
59. L. Xu, A. Krzyzak, and E. Oja, "Neural-Net Method for Dual Subspace Pattern Recognition," in *Proceedings of the International Joint Conference on Neural Networks*, Seattle, WA, vol. 2, 1991, pp. 379–84.
60. J. Karhunen and J. Joutsensalo, "Learning of Sinusoidal Frequencies by Nonlinear Constrained Hebbian Algorithms," in *Neural Networks for Signal Processing II*, eds. S. Y. Kung, F. Fallside, J. A. Sorenson, and C. A. Kamm, New York: IEEE Press, 1992, pp. 39–48. 462
61. J. Joutsensalo and J. Karhunen, "A Nonlinear Extension of the Generalized Hebbian Learning," *Neural Processing Letters*, vol. 2, 1995, pp. 5–8.
62. J. Karhunen and J. Joutsensalo, "Frequency Estimation by a Hebbian Subspace Learning Algorithm," in *Artificial Neural Networks*, eds. T. Kohonen et al., Amsterdam: North-Holland, 1991, pp. 1637–40.
63. J. Karhunen and J. Joutsensalo, "Nonlinear Hebbian Algorithms for Sinusoidal Frequency Estimation," in *Artificial Neural Networks*, eds. I. Aleksander and J. Taylor, Amsterdam: North-Holland, 1992, pp. 1199–202.
64. S. Amari, "Mathematical Theory of Neural Learning," *New Generation Computing*, vol. 8, 1991, pp. 281–94.
65. D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd ed., Reading, MA: Addison-Wesley, 1984.

66. S. Y. Kung, "Adaptive Principal Component Analysis Via an Orthogonal Learning Network," in *Proceedings of the International Symposium on Circuits and Systems*, New Orleans, 1990, pp. 719–22.
67. K. I. Diamantaras, "Principal Component Learning Networks and Applications," *Ph.D. dissertation*, Princeton, NJ: Princeton University, 1992.
68. S. Bannour and M. R. Azimi-Sadjadi, "Principal Component Extraction Using Recursive Least Squares Method," in *Proceedings of the International Joint Conference on Neural Networks*, Singapore, 1991, pp. 2110–5.
69. S. Y. Kung and K. I. Diamantaras, "A Neural Network Learning Algorithm for Adaptive Principal Component Extraction (APEX)," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Albuquerque, NM, 1990, vol. 2, pp. 861–4.
70. J. Karhunen and J. Joutsensalo, "Learning of Robust Principal Component Subspace," in *Proceedings of the International Joint Conference On Neural Networks*, Nagoya, Japan, October 1993, pp. 2409–12.
71. J. Karhunen and J. Joutsensalo, "Nonlinear Generalizations of Principal Component Learning Algorithms," in *Proceedings of the International Joint Conference On Neural Networks*, Nagoya, Japan, October 1993, pp. 2599–602.
72. A. Cichocki and R. Unbehauen, "Robust Estimation of Principal Components by Using Neural Network Learning Algorithms," *Electronic Letters*, vol. 29, 1993, pp. 1869–70.
73. L. Xu and A. Yuille, "Self-Organizing Rules for Robust Principal Component Analysis," in *Advances in Neural Information Processing Systems*, vol. 5, eds. S. J. Hanson, J. D. Cowan, and C. L. Giles, San Mateo, CA: Morgan Kaufmann, 1993, pp. 467–74.
74. J. Joutsensalo, "Conventional and Neural Methods for Estimating Principal Component Type Subspaces with Application to Sinusoidal Frequency Estimation," *Ph.D. dissertation*, Helsinki, Finland: Helsinki, University of Technology, 1994.
75. J. Karhunen and J. Joutsensalo, "Representation and Separation of Signals Using Nonlinear PCA Type Learning," Technical Report A17 (Otaniemi 1993), Helsinki University of Technology (Laboratory of Computer and Information Sciences), Espoo, Finland, 1993.
76. L. Xu, "Least MSE Reconstruction for Self-Organization," *Proceedings of the International Joint Conference On Neural Networks*, Part 2, Singapore, November 1991, pp. 2368–73.
77. L. E. Russo, "An Outer Product Neural Network for Extracting Principal
463 Components from a Time Series," in *Neural Networks for Signal Processing (Proceedings of the 1991 IEEE Workshop)*, eds. B. H. Juang, S. Y. Kung, and C. A. Kamm, New York: IEEE Press, 1991, pp. 161–70.
78. F. L. Luo, R. Unbehauen, and A. Cichocki, "A Minor Component Analysis," *Neural Networks*, vol. 10, 1997, pp. 291–7.
79. K. Matsuoka and M. Kawamoto, "A Neural Network that Self-Organizes to Perform Three Operations Related to Principal Component Analysis," *Neural Networks*, vol. 7, 1994, pp. 753–65.
80. F. M. Ham and I. Kim, "Extension of the Generalized Hebbian Algorithm for Principal Component Extraction," in *Proceedings of Applications and Science of Neural Networks, Fuzzy Systems and Evolutionary Computation*, eds. J. C. Bezdek, B. Bosacchi, and D. B. Fogel, San Diego: SPIE, July 19–24, 1998, vol. 3455, pp. 274–85.
81. K. I. Diamantaras and S. Y. Kung, *Principal Component Neural Networks—Theory and Applications*, New York: Wiley-Interscience, 1996.
82. F. Peper and H. Noda, "A Symmetric Linear Neural Network that Learns

Principal Components and Their Variances," *IEEE Transactions on Neural Networks*, vol. 7, 1996, pp. 1042–6.
83. W. Skarbek, A. Cichocki and W. Kasprzak, "Principal Subspace Analysis for Incomplete Image Data in One Learning Epoch," *Neural Network World*, vol. 6, 1996, pp. 375–81.
84. F. L. Luo, R. Unbehauen and Y. D. Li, "A Principal Component Analysis Algorithm with Invariant Norm," *Neurocomputing*, vol. 8, 1995, pp. 213–21.
85. K. I. Diamantaras and S. Y. Kung, "Cross-Correlation Neural Network Models," *IEEE Transactions on Signal Processing*, vol. 42, 1994, pp. 3218–23.
86. D. Obradovic and G. Deco, "An Information Theory Based Learning Paradigm for Linear Feature Extraction," *Neurocomputing*, vol. 12, 1996, pp. 203–21.
87. M. D. Plumbley, "Lyapunov Functions for Convergence of Principal Component Algorithms," *Neural Networks*, vol. 8, 1995, pp. 11–23.
88. J. Dehaene, M. Moonen, and J. Vandewalle, "An Improved Stochastic Gradient Algorithm for Principal Component Analysis and Subspace Tracking," *IEEE Transactions on Signal Processing*, vol. 45, 1997, pp. 2582–6.
89. W. Y. Yan, U. Helmke, and J. B. Moore, " Global Analysis of Oja's Flow for Neural Networks," *IEEE Transactions on Neural Networks*, vol. 5, 1994, pp. 674–83.
90. Q. Zhang and Y. W. Leung, "Energy Function for the One Unit Oja Algorithm," *IEEE Transactions on Neural Networks*, vol. 6, 1995, pp. 1291–3.
91. T. Chen, Y. Hua and W. Y. Yan, "Global Convergence of Oja's Subspace Algorithm for Principal Component Extraction," *IEEE Transactions on Neural Networks*, vol. 9, 1998, pp. 58–67.
92. S. Bannour and M. R. Azami-Sadjadi, "Principal Component Extraction Using Recursive Least Squares Learning," *IEEE Transactions on Neural Networks*, vol. 6, 1995, pp. 457–69.
93. F. Peper and H. Noda, "A Symmetrical Linear Neural Network that Learns Principal Components and Their Variances, *IEEE Transactions on Neural Networks*, vol. 7, 1996, pp. 1042–7.
94. V. Solo and X. Kong, "Performance Analysis of Adaptive Eigenanalysis Algorithms," *IEEE Transactions on Signal Processing*, vol. 46, 1998, pp. 636–46.
95. C. Chatterjee, V. P. Roychowdhury, and E. K. P. Chong, "On Relative Convergence Properties of Principal Component Analysis Algorithms," *IEEE Transactions on Neural Networks*, vol. 9, 1998, pp. 319–29.
96. G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Baltimore, MD: Johns Hopkins University Press, 1996. 464
97. J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, London: Oxford University Press, 1965.
98. K. Pearson, "On Lines and Planes of Closest Fit to Systems of Points in Space," *Philosophical Magazine*, ser. 6, vol. 2, 1901, pp. 559–72.
99. L. E. Russo and E. C. Real, "Image Compression Using an Outer Product Neural Network," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, 1992, pp. 377–80.
100. G. W. Cottrell and P. Munro, "Principal Components Analysis of Images via Back Propagation," in *SPIE Visual Communications and Image Processing*, vol. 1001, 1988, pp. 1070–7.
101. E. Oja and T. Kohonen, "A Subspace Learning Algorithm as a Formalism for Pattern Recognition and Neural Networks," in *Proceedings of IEEE Neural Network Conference*, vol. 1, 1988, pp. 270–84.
102. T. Y. Young and T. W. Calvert, *Classification, Estimation, and Pattern Recognition*, New York: American Elsevier, 1974.
103. P. A. Devijver and J. Kittler, *Pattern Recognition, A Statistical Approach*,

Englewood Cliffs, NJ: Prentice-Hall, 1982.
104. A. K. Jain, *Fundamentals of Digital Image Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1989.
105. L. Xu and A. Yuille, "Robust PCA Learning Rules Based on Statistical Physics Approach," *Proceedings of the International Joint Conference on Neural Networks*, Part 1, Baltimore, MD, June 1992, pp. 812–17.
106. M. A. Kramer, "Nonlinear Principal Component Analysis Using Autoassociative Neural Networks," *AIChE Journal*, vol. 37, 1991, pp. 233–43.
107. B. Porat, *Digital Processing of Random Signals*, Englewood Cliffs, NJ: Prentice-Hall, 1994.
108. B. Picinbono, *Random Signals and Systems*, Englewood Cliffs, NJ: Prentice-Hall, 1993.
109. C. W. Therrien, *Discrete Random Signals and Statistical Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1992.
110. C. L. Nikias and J. M. Mendel, "Signal Processing with Higher-Order Spectra," *IEEE Signal Processing Magazine*, vol. 10, 1993, pp. 10–37.
111. L. Parra, G. Deco, and S. Miesbach, "Redundancy Reduction with Information-Preserving Nonlinear Maps," *Network*, vol. 1, 1995, pp. 61–72.
112. P. Comon, "Independent Component Analysis: A New Concept?" *Signal Processing*, vol. 36, 1994, pp. 287–314.
113. E. Oja, "The Nonlinear PCA Learning Rule in Independent Component Analysis," *Neurocomputing*, vol. 17, 1997, pp. 25–45.
114. J. Karhunen, E. Oja, L. Wang, R. Vigario, and J. Joutsensalo, "A Class of Neural Networks for Independent Component Analysis," *IEEE Transactions on Neural Networks*, vol. 8, 1997, pp. 486–504.
115. L. Parra, "Symplectic Nonlinear Independent Component Analysis," in *Advances in Neural Information Processing Systems*, vol. 8, eds. D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Cambridge, MA: MIT Press, 1996, pp. 437–43.
116. C. N. Banwell, *Fundamentals of Molecular Spectroscopy*, New York: McGraw-Hill, 1966.
117. B. P. Straughtan and S. Walker, eds., *Spectroscopy*, vol. 2, New York: Wiley, 1976.
118. P. Williams and K. Norris, eds., *Near-Infrared Technology*, St. Paul, MN:
465 American Association of Cereal Chemists, 1987.
119. M. Avram and G. H. Mateescu, *Infrared Spectroscopy*, New York: Wiley-Interscience, 1972.
120. E. R. Malinowski, "Determination of the Number of Factors and the Experimental Error in a Data Matrix," *Analytical Chemistry*, vol. 49, 1977, pp. 612-7.
121. E. R. Malinowski, *Factor Analysis in Chemistry*, 2nd. ed., New York: Wiley, 1991.
122. H. Wold, "Soft Modelling: The Basic Design and Some Extensions," *Systems Under Direct Observation, Causality-Structure-Prediction*, eds. K. G. Joreskog and H. Wold, Amsterdam: North-Holland, 1981.
123. H. Wold, in *Food Research and Drug Analysis*, eds. H. Martens and H. Russwurm, London: Applied Science Publ., 1983.
124. E. R. Malinowski, "Theory of Error in Factor Analysis," *Analytical Chemistry*, vol. 49, 1977, pp. 606–12.
125. D. M. Haaland, "Classical Versus Inverse Least-Squares Methods in Quantitative Spectral Analyses," *Spectroscopy*, vol. 2, 1987, pp. 56–7.
126. F. M. Ham and I. Kostanic, "A Neural Network Architecture for Partial Least-

Squares Regression (PLSNET) with Supervised Adaptive Modular Hebbian Learning," *Neural, Parallel, Scientific Computations*, vol. 6, 1998, pp. 35–72.
127. F. M. Ham and I. Kostanic, "A Partial Least-Squares Regression Neural NETwork (PLSNET) with Supervised Adaptive Modular Learning," in *Applications and Science of Artificial Neural Networks II*, eds. S. K. Rogers and D. W. Ruck, Proceedings of SPIE, vol. 2760, 1996, pp. 139–50.
128. T. R. Holcomb and M. Morari, "PLS/Neural Networks," *Computers in Chemical Engineering*, vol. 16, 1992, pp. 393–411.
129. S. J. Qin and T. J. McAvoy, "Nonlinear PLS Modeling Using Neural Networks," *Computers in Chemical Engineering*, vol. 16, 1992, pp. 379–91.
130. E. C. Malthouse, "Nonlinear Partial Least Squares", *Ph.D. dissertation*, Evanston, IL: Northwestern University, 1995.
131. F. M. Ham, I. Kostanic, G. M. Cohen, and B. R. Gooch, "Determination of Glucose Concentrations in an Aqueous Matrix from NIR Spectra Using Optimal Time-Domain Filtering and Partial Least-Squares Regression," *IEEE Transactions on Biomedical Engineering*, vol. 44, 1997, pp. 475–85.
132. F. M. Ham, G. M. Cohen, I. Kostanic, and B. R. Gooch, "Multivariate Determination of Glucose Concentrations from Optimally Filtered Frequency-Warped NIR Spectra of Human Blood Serum," *Journal of Physiological Measurement*, vol. 17, 1996, pp. 1–20.
133. F. M. Ham and T. M. McDowall, "Robust Learning in a Partial Least-Squares Neural Network," *Journal of Nonlinear Analysis, Theory, Methods & Applications*, Proceedings of the Second World Congress of Nonlinear Analysts 1996, July 10–17, Athens Greece, vol. 30, 1997, pp. 2903–14.
134. T. M. McDowall, *A Robust Partial Least-Squares Neural Network, Ph.D. dissertation*, Melbourne: Florida Institute of Technology, May 1997.
135. T. M. McDowall and F. M. Ham, "Robust Partial Least-Squares Regression: A Modular Neural Network Approach," in *Applications and Science of Artificial Neural Networks III*, ed. S. Rogers, Proceedings of SPIE, vol. 3077, 1997, pp. 344–55.
136. T. M. McDowall and F. M. Ham, "Robust Learning in Generalized Partial Least-Squares Regression Modular Neural Network", *Neural Parallel and Scientific Computations*, vol. 6, 1998, pp. 391–415.
137. F. M. Ham and T. M. McDowall, "Inverse Model Formulation of Partial Least-Squares Regression: A Robust Neural Network Approach," (Invited Paper), in *Applications and Science of Computational Intelligence*, eds. S. K. Rogers, D. B. Fogel, J. C. Bezdek, and B. Bosacchi, Proceedings of SPIE, vol. 3390, 1998, pp. 36–47.
138. C. Wang, H.-C. Wu, and J. C. Principe, "A Cost Function for Robust Estimation of PCA" in *Applications and Science of Artificial Neural Networks II*, eds. S.K. Rogers and D. W. Ruck, Orlando, FL, Proceedings of SPIE, vol. 2760, 1996, pp. 120–7.
139. P. J. Huber, *Robust Statistics*, New York: Wiley, 1981.
140. F. R. Hampel, P. J. Rousseeuw, E. M. Ronchetti, and W. A. Stahel, *Robust Statistics—The Approach Based on Influence Functions*, New York: Wiley, 1986.
141. D. S. Chen and R. C. Jain, "A Robust Back Propagation Learning Algorithm for Function Approximation," *IEEE Transactions on Neural Networks*, vol. 5, 1994, pp. 467–79.

466

467

# 第10章　使用神经网络进行辨识、控制和估计

## 10.1　概述

本章是关于不同类型的神经网络在信号处理、辨识、分类、控制和估计问题中的应用。特别是，分别运用自回归滑动平均（ARMA）和非线性ARMA（NARMA）模型对线性系统和非线性系统的参数辨识进行了讨论。对非线性系统的控制和随后的盲源分离问题（即，运用独立成分分析法（ICA）对未知源信号进行分离）也进行了讨论。然后，讨论了用部分最小二乘回归法进行频谱估计，在本章的最后一节给出了运用神经网络解决两类重要问题的实例研究。

## 10.2　线性系统的表示法

本章我们只讨论离散线性时不变动态系统的表示法。用两种基本的方法来描述动态系统：输入/输出（或传递函数）法和状态空间法。着重讨论单输入单输出（SISO）的情况。由于自然现象非常复杂，在某些情况下，现有的科学知识不能充分解释我们关心的一些动态系统
468 （设备）。然而，可以根据实验数据构建一个系统模型，也就是说，可以用适当的输入激发设备，对反应进行测量。通过这些输入/输出数据，可以构建一个模型对未知的设备参数进行估计，这称为*系统辨识* [1-4] 。我们关注进行*参数辨识*的模型，而不是*非参数模型*的辨识[1-4]。

在用输入/输出（传递函数）法描述离散时间线性非时变动态系统时，严格正常有理传递函数形式假定如下：

$$H(z)=\frac{Y(z)}{U(z)}=\frac{b_1z^{n-1}+b_2z^{n-2}+b_3z^{n-3}+\cdots+b_n}{z^n+a_1z^{n-1}+a_2z^{n-2}+\cdots+a_n} \tag{10-1}$$

其中$z$为复变量。当用状态空间表示动态系统时，状态变量的规范形式或可控的正则形式[⊖]是设定的（详见A.2.12节），状态方程形式如下：

$$\dot{\boldsymbol{x}}(t)=\begin{bmatrix}0&1&0&0&\cdots&0\\0&0&1&0&\cdots&0\\\cdots&\cdots&\cdots&\cdots&\cdots&\cdots\\0&0&0&\cdots&1&0\\0&0&0&\cdots&0&1\\-a_n&-a_{n-1}&-a_{n-2}&\cdots&-a_2&-a_1\end{bmatrix}\boldsymbol{x}(t)+\begin{bmatrix}0\\0\\0\\\vdots\\0\\1\end{bmatrix}u(t) \tag{10-2}$$

输出（测量）方程由下式给出：

$$y(t)=[b_n,b_{n-1},b_{n-2},\cdots,b_2,b_1]\boldsymbol{x}(t) \tag{10-3}$$

参数辨识的目的是估计与所关注的系统相关的参数向量（$\boldsymbol{\theta}$），也就是式（10-1）或式（10-2）

⊖ 也可以假定可观察的正则形式[5]（这是可控的正则形式的对偶）。

和式（10-3）中出现的系统参数。

$$\boldsymbol{\theta} = [a_1, a_2, \cdots, a_n, b_1, b_2, \cdots, b_n]^T \tag{10-4}$$

## 10.3 自回归滑动平均模型

从式（10-1）的传递函数可以得到一个如下的时域差分方程：

$$\begin{aligned} &y(k) + a_1 y(k-1) + a_2 y(k-2) + \cdots + a_n y(k-n) \\ &-b_1 u(k-1) - b_2 u(k-2) - b_3 u(k-3) - \cdots - b_n u(k-n) \\ &= \varepsilon(k; a_1, a_2, \cdots, a_n, b_1, b_2, b_3, \cdots, b_n) \end{aligned} \tag{10-5}$$

其中$\varepsilon(k; a_1, a_2, \cdots, a_n, b_1, b_2, b_3, \cdots, b_n)$是一个误差项，当参数向量

$$\boldsymbol{\theta} = [a_1, a_2, \cdots, a_n, b_1, b_2, \cdots, b_n]^T \tag{10-6}$$

469

包含实际的或真实的设备参数时为0[2]，$k$是离散时间索引。方程（10-5）也可写为：

$$y(k) = \phi^T(k)\theta + \varepsilon(k; \theta) \tag{10-7}$$

其中

$$\begin{aligned} \phi^T(k) = [&-y(k-1), -y(k-2), \cdots \\ &-y(k-n), u(k-1), u(k-2), u(k-3), \cdots, u(k-n)] \end{aligned} \tag{10-8}$$

对于离散时间索引$k = n, n+1, n+2, \cdots, N$，$n$是设定的系统维数，$N$是数据集$\{\boldsymbol{u}(k), \boldsymbol{y}(k)\}$中用到的样本总数，由式（10-7）可推出如下方程组：

$$\boldsymbol{y}(N) = \Phi(N)\theta + \varepsilon(N, \theta) \tag{10-9}$$

其中

$$y^T(N) = [y(n), y(n+1), y(n+2), \cdots, y(N)] \tag{10-10}$$

并且

$$\Phi^T(N) = [\phi(n), \phi(n+1), \phi(n+2), \cdots, \phi(N)] \tag{10-11}$$

$$\varepsilon^T(N; \theta) = [\varepsilon(n; \theta), \varepsilon(n+1; \theta), \varepsilon(n+2; \theta), \cdots, \varepsilon(N; \theta)] \tag{10-12}$$

$$\begin{aligned} &\Phi(N) \in \Re^{(N-n+1)\times 2n} && \text{ARMA数据矩阵} \\ &\boldsymbol{y}(k) \in \Re^{N-n+1} && \text{输出向量} \\ &\varepsilon(N, \theta) \in \Re^{N-n+1} && \text{误差向量} \\ &\theta \in \Re^{2n} && \text{系统参数向量} \end{aligned} \tag{10-13}$$

## 10.4 用ARMA模型的线性系统辨识

利用传统最小二乘（CLS）法解方程（10-9）可求得参数向量$\boldsymbol{\theta}$（参见9.4节）。首先，性能度量定义为

$$J(\theta) = \|\varepsilon(N, \theta)\|_2^2 \tag{10-14}$$

其中由式（10-7）得$\boldsymbol{\varepsilon} = \boldsymbol{y} - \boldsymbol{\Phi\theta}$。然后，将其对$\boldsymbol{\theta}$进行最小化，即

$$\min_{\theta \in \Re^{2N}} J(\theta) = \min_{\theta \in \Re^{2N}} \|\varepsilon(N, \theta)\|_2^2 \tag{10-15}$$

其中$N$表示数据集$\{u(k), y(k)\}$中所用的样本总数，取

$$N > 2n \tag{10-16}$$

对$\boldsymbol{\theta}$最小化$\|\boldsymbol{\varepsilon}(N, \boldsymbol{\theta})\|_2^2$可得到CLS的结果：

$$\hat{\boldsymbol{\theta}}_{\mathrm{CLS}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y} \tag{10-17}$$

470 其中参数向量$\hat{\boldsymbol{\theta}}_{\mathrm{CLS}}$由式（10-6）给出。

现在自然有一个问题要问，存在唯一的解吗？或者说式（10-17）的解$\hat{\boldsymbol{\theta}}_{\mathrm{CLS}}$是唯一解吗？回答这个问题要看以下两点：(1) 如何选取参数向量$\boldsymbol{\theta}$；(2) 用什么类型的输入信号$\{u(k)\}$去激发系统得到输出序列$\{y(k)\}$。必须选择一个典范型的模型或ARMA模型，才能得到唯一的参数集。参数向量$\theta$称为可辨识的，如果有且只有唯一的值使$J(\boldsymbol{\theta})$最小，如果存在两个参数向量$\boldsymbol{\theta}_1$、$\boldsymbol{\theta}_2$，使$J(\boldsymbol{\theta}_1) = J(\boldsymbol{\theta}_2)$，则说它们是等价的[2]。选择正确的输入信号的问题可以用使输入必须具有$n$阶持续激发的方式提出[1, 2]。简单地说，输入信号必须能充分地“激发”系统，使系统的输出有足够的信息对它的属性进行辨识（或者说能够对参数向量$\boldsymbol{\theta}$进行正确的估计）。如果输入信号的离散频谱在区间$0 \leqslant \omega < \omega_s/2$中至少存在$n$个非零点，则它就是一个$n$阶持续激发，其中$\omega_s$是采样频率。在公式（10-17）的CLS解中，矩阵$\boldsymbol{\Phi}^T\boldsymbol{\Phi} \in \Re^{2n \times 2n}$必须满秩，即$\rho(\boldsymbol{\Phi}^T\boldsymbol{\Phi}) = 2n$。Franklin et al. [2]提出，如果矩阵$\boldsymbol{\Phi}^T\boldsymbol{\Phi}$的右下区分量（部分）$(n \times n$，只依赖于输入向量$\{u(k)\})$是非奇异矩阵，则称输入信号是$n$阶持续激发。举例说明，假设$N = 6$，$n = 3$，输入信号是一个常数，即$u(k) = c$，得到下式：

$$\boldsymbol{\Phi}^T\boldsymbol{\Phi} = \begin{bmatrix} \sum_{i=1}^{4} y(i+1)y(i+1) & \sum_{i=1}^{4} y(i+1)y(i) & \sum_{i=1}^{4} y(i+1)y(i-1) & -c\sum_{i=1}^{4} y(i+1) & -c\sum_{i=1}^{4} y(i+1) & -c\sum_{i=1}^{4} y(i+1) \\ \sum_{i=1}^{4} y(i)y(i+1) & \sum_{i=1}^{4} y(i)y(i) & \sum_{i=1}^{4} y(i)y(i-1) & -c\sum_{i=1}^{4} y(i) & -c\sum_{i=1}^{4} y(i) & -c\sum_{i=1}^{4} y(i) \\ \sum_{i=1}^{4} y(i-1)y(i+1) & \sum_{i=1}^{4} y(i-1)y(i) & \sum_{i=1}^{4} y(i-1)y(i-1) & -c\sum_{i=1}^{4} y(i-1) & -c\sum_{i=1}^{4} y(i-1) & -c\sum_{i=1}^{4} y(i-1) \\ -c\sum_{i=1}^{4} y(i+1) & -c\sum_{i=1}^{4} y(i) & -c\sum_{i=1}^{4} y(i-1) & 4c^2 & 4c^2 & 4c^2 \\ -c\sum_{i=1}^{4} y(i+1) & -c\sum_{i=1}^{4} y(i) & -c\sum_{i=1}^{4} y(i-1) & 4c^2 & 4c^2 & 4c^2 \\ -c\sum_{i=1}^{4} y(i+1) & -c\sum_{i=1}^{4} y(i) & -c\sum_{i=1}^{4} y(i-1) & 4c^2 & 4c^2 & 4c^2 \end{bmatrix} \tag{10-18}$$

从式（10-18）中我们看到右下角的（3×3）矩阵区域是奇异矩阵。因而常量输入信号$u\{k\} = c$不是$n$阶持续激发，关于参数$\boldsymbol{\theta}$的最小化$\|\boldsymbol{\varepsilon}(N, \boldsymbol{\theta})\|_2^2$不存在唯一解。

在系统辨识中，高斯白噪声和线性调频信号这两种很重要的信号可以用来进行激发动态系统[6]。也有其他类型的信号可利用，但这两种信号在系统辨识中很典型可适当地激发模拟系统。图10-1a显示典型的线性调频信号，图10-1b显示了与快速傅里叶变换（FFT）相关的幅度。在图10-1b中我们看到幅度响应包含一个大的频段，所以，保证了兴奋输入信号是$n$阶持
471 续激发。

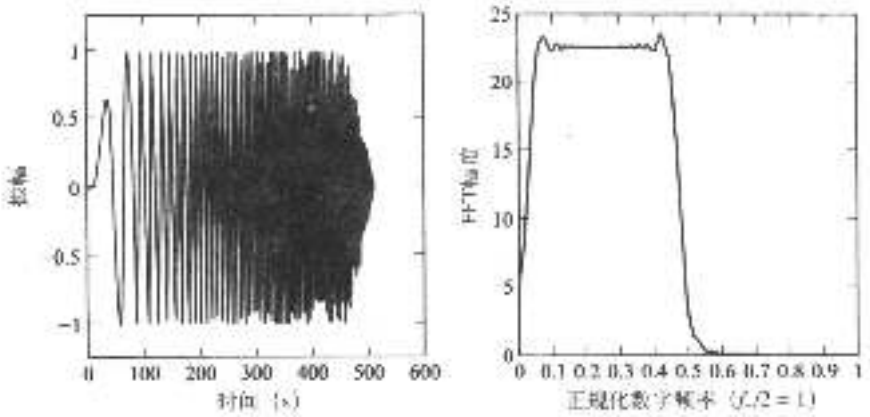



图10-1　a) 线性调频信号；b) 线性调频信号的快速傅里叶转换幅频图（相应于半采样频率点的单位正规化的频率。即$f/f2=1$）

另一个需要讨论的重点是系统维数的选择，即系统的阶$n$。在参数系统辨识中这是一个更具挑战性的问题。系统的阶数是一个未知量，必须在开始时选择，而且要尽可能地按照所估计结果去调整。因此，选择系统的阶数$n$可能是一个迭代过程。经验知识常能有助于得到系统模型的阶数范围。特殊应用也有助于系统模型阶数的确定。人们已经提出几种方法来帮助选择系统阶数$n$，其中的三种方法描述如下，它们都基于对初始数据分析［1］：

1. 检验系统传递函数的频谱估计。这种方法是从输入/输出数据$\{u(k), y(k)\}$中寻找传递函数的非参数估计。

2. 测试样本协方差矩阵的秩。假设采用ARMA模型，从输入输出数据$\{u(k), y(k)\}$对样本协方差进行估计。应用式（10-8）中给出的$\boldsymbol{\phi}(k)$，样本协方差矩阵可写为如下形式：

$$\boldsymbol{C}_z(N)=\frac{1}{N}\sum_{k=1}^{N}\boldsymbol{\phi}_z(k)\boldsymbol{\phi}_z^{\mathrm{T}}(k)=\frac{1}{N}\boldsymbol{\Phi}_z^{\mathrm{T}}(N)\boldsymbol{\Phi}_z(N) \tag{10-19}$$

其中$\hat{n}$是实际（或真正）系统维数$n$的估计值。当$\hat{n}\geqslant n$［假设$u(k)$是持续激发］时，式（10-19）中的样本协方差矩阵是非奇异矩阵；当$\hat{n}\geqslant n+1$时，它是奇异矩阵。很明显应用$|\boldsymbol{C}_z(N)|=\det|\boldsymbol{C}_z(N)|$能够测试系统模型的估计阶数$\hat{n}$。

3. 关联变量。阶数判定问题也可以看作是讨论在模型结构中是否包括另一个变量，即在式（10-5）中附加一项$y(k-n-1)$［或者一个可能的扰动变量$v(k)$的附加作用］，并且确定当明确输出$y(k)$时，这个附加变量起到了什么作用。这可以通过$y(k)$和$v(k)$的关系计算出来。但是，$y(k)$和$v(k)$之间有关系，因此，$y(k)$和$u(k;\hat{\boldsymbol{\theta}}_n)$之间的关系可以计算。其中$\varepsilon(k;\hat{\boldsymbol{\theta}}_n)=y(k)-\hat{y}(k;\hat{\boldsymbol{\theta}}_n)$构成余数。这就是正规相关或部分相关[7]。

还有另一个方法也可用来判定系统的阶数，这个方法就是部分最小二乘回归（PLSR）（详见9.5节），或是PLSNET（详见9.6节）。PLSR或PLSNET都能用来估计参数向量，同时系统阶数$n$也可以通过因子分析得到。这是下一节的主题。

在本节讨论中，提到多种方法可以用来估计系统参数向量[1-4]。一个非常重要的方法包括最小二乘递归加权算法（RWLS）[1, 2]，因为所有从输入/输出数据$\{u(k), y(k)\}$得到的可用数据同时用来估计参数$\boldsymbol{\theta}$。式（10-17）的结果指出了要解。RWLS方法的两个主要优点是：(1) RWLS是一个迭代算法，自适应地估计参数向量。(2) 因为在递归算法中加权函数$w(k)=\alpha\gamma^{N-k}$，过

去数据（观察）和现在数据相比显得不重要，因此，加权函数起到了滤波器的作用。

## 10.5　应用PLSNET进行线性系统的参数系统辨识

部分最小二乘法用来进行参数系统辨识[8]，同使用PLSNET法一样[9]（详见9.6节），也利用ARMA模型。首先根据输入/输出数据集$\{\boldsymbol{u}(k), \boldsymbol{y}(k)\}$建立训练和测试集如下：

$$\text{训练数据集} = \{\boldsymbol{\Phi}_{\text{train}}(N), \boldsymbol{y}_{\text{train}}(\text{N})\} \tag{10-20}$$

$$\text{测试数据集} = \{\boldsymbol{\Phi}_{\text{test}}(N), \boldsymbol{y}_{\text{test}}(\text{N})\} \tag{10-21}$$

训练数据集和测试数据集选自集合$\{\boldsymbol{u}(k), \boldsymbol{y}(k)\}$的不同部分。在式（10-20）和式（10-21）中的两个数据集生成时，设定系统的阶数$n$超过指定的值，也就是说，如果$\bar{n}$是系统的实际维数，那么设定的系统维数是$n > \bar{n}$。运用PLSNET来处理参数系统辨识的目的就是估计系统维数$\hat{n}$和参数向量$\hat{\boldsymbol{\theta}}$。

473

为选择最优的PLSR因子数，需使用独立确认方法（详见9.4节）。当神经网络执行PLSR时，数据矩阵（独立变量块）由下式给出

$$\boldsymbol{A} = \boldsymbol{\Phi}(N) \tag{10-22}$$

观察向量、目标值（因变量块）是

$$\boldsymbol{c} = \boldsymbol{y}(N) \tag{10-23}$$

分别利用式（10-20）和式（10-21）中的训练数据集和测试数据集，执行公式（9-111）的因子分析。在确定了最优的因子数时，就能估计出系统的阶数：

$$\hat{n} = \frac{h^o}{2} \tag{10-24}$$

$h^o$是由因子分析得到的最优因子数[10]。利用PLSR因子分析法能得到系统最小实现的阶数[5]，即一个系统模型总是可控制的和可观察的（详见A.2.12节）。在标准预测误差与PLSR的因子数的图上选择第一最小即为此情形。在最优的因子数取定后，取$N > 2\hat{n}$，可以由数据集$\{\boldsymbol{u}(k), \boldsymbol{y}(k)\}$得到一个新的数据集如下：

$$\text{最后数据集} = \{\Phi_f(N; \hat{n}), \boldsymbol{y}_f(N; \hat{n})\} \tag{10-25}$$

用于生成$\boldsymbol{\Phi}_f(N; \hat{n})$和$\boldsymbol{y}_f(N; \hat{n})$的样本数不必一定要和生成公式（10-20）中集合$\{\boldsymbol{\Phi}_{\text{train}}(N), \boldsymbol{y}_{\text{train}}(N)\}$和公式（10-21）中集合$\{\boldsymbol{\Phi}_{\text{test}}(N), \boldsymbol{y}_{\text{test}}(N)\}$的数目一样，但必须是$N > 2\hat{n}$。参数向量$\hat{\boldsymbol{\theta}}$的估计值也可以使用9.5节的预测方法1来获得。根据PLSNET-C（详见9.6节），利用式（10-25）表示的最后数据集，PLSR得到权值装载向量$\{\hat{\boldsymbol{w}}_1, \hat{\boldsymbol{w}}_2, \cdots, \hat{\boldsymbol{w}}_{h^o}\}$、装载向量$\{\hat{\boldsymbol{b}}_1, \hat{\boldsymbol{b}}_2, \cdots, \hat{\boldsymbol{b}}_{h^o}\}$和回归系数$\{\hat{\upsilon}_1, \hat{\upsilon}_2, \cdots, \hat{\upsilon}_{h^o}\}$。利用这些信息，可以构成下面的矩阵：

$$\hat{\boldsymbol{W}}^T = [\hat{\boldsymbol{w}}_1, \hat{\boldsymbol{w}}_2, \cdots, \hat{\boldsymbol{w}}_{h^o}] \tag{10-26}$$

$$\hat{\boldsymbol{B}}^T = [\hat{\boldsymbol{b}}_1, \boldsymbol{b}_2, \cdots, \hat{\boldsymbol{b}}_{h^o}] \tag{10-27}$$

$$\hat{\upsilon}^T = [\hat{\upsilon}_1, \hat{\upsilon}_2, \cdots, \hat{\upsilon}_{h^o}] \tag{10-28}$$

其中$\hat{\boldsymbol{W}} \in \Re^{h^o \times h^o}$，$\hat{\boldsymbol{B}} \in \Re^{h^o \times h^o}$，$\hat{\upsilon} \in \Re^{h^o \times 1}$，$h^o = 2\hat{n}$。从式（10-26）、式（10-27）和式（10-28）可以得出最优的PLSR校正模型（最后校正系数）$\hat{\boldsymbol{b}}_{f\,\text{PLSR}}$，表示为下式：

$$\hat{\theta} = \hat{\boldsymbol{b}}_{f\,\text{PLSR}} = \hat{\boldsymbol{W}}^T(\hat{\boldsymbol{B}}\hat{\boldsymbol{W}}^T)^{-1}\hat{\upsilon} \tag{10-29}$$

因此，参数向量估计值$\hat{\boldsymbol{\theta}}$就是PLSR校正模型$\hat{\boldsymbol{b}}_{f\,\mathrm{PLSR}}$。通常PLSR校正模型也用来预测。这样，最后的校正模型将预测系统的输出（响应）$\{y(k)\}$。为了估计已建参数模型的性能，如式（10-25）所示将生成另外一个数据集合，它利用数据集合$\{\boldsymbol{u}(k), \boldsymbol{y}(k)\}$中没有用过的部分生成$\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{b}}_{f\,\mathrm{PLSR}}$，即： 474

$$\text{最后（用来测试的）数据集} = \{\boldsymbol{\Phi}_{f\,\mathrm{test}}(N; \hat{n}), \boldsymbol{y}_{f\,\mathrm{test}}(N; \hat{n})\} \tag{10-30}$$

其中$N > 2\hat{n}$。利用$\boldsymbol{\Phi}_{f\,\mathrm{test}}(N; \hat{n})$和式（10-29），按下式可以得到$\boldsymbol{y}_{f\,\mathrm{test}}$的估计值：

$$\hat{\boldsymbol{y}}_{f\,\mathrm{test}} = \boldsymbol{\Phi}_{f\,\mathrm{test}}\hat{\boldsymbol{\theta}} = \boldsymbol{\Phi}_{f\,\mathrm{test}}\hat{\boldsymbol{b}}_{f\,\mathrm{PLSR}} \tag{10-31}$$

这个值可以和实际输出值$\boldsymbol{y}_{f\,\mathrm{test}}$进行比较。

**例10.1**　说明怎样运用PLSNET进行参数系统辨识。实际离散时间系统的传递函数由以下二阶函数给出：

$$H(z) = \frac{z - 0.1}{z^2 - 0.5999z + 0.05} \tag{10-32}$$

其中采样周期$T_s = 2\pi/1\,000\mathrm{s}$，$\bar{n} = 2$（系统的实际阶数）。因此，实际的参数向量为

$$\boldsymbol{\theta}^T = [-0.5999, 0.05, 1, -0.1] \tag{10-33}$$

仿真的输入/输出经验数据$\{\boldsymbol{u}(k), \boldsymbol{y}(k)\}$在MATLAB中用零平均（zero-mean）单位方差高斯白噪声函数作为输入生成，利用由公式（10-33）得到的参数，用MATLAB函数`dlsim`生成输出数据。

输入和输出序列都有1024个样本；每个序列的前100个样本用来生成训练数据，如式（10-20）所示，其后的100个样本用来组成测试数据集，如式（10-21）所示，系统的阶数设为$n = 2$。ARMA数据矩阵由式（10-11）给出。利用训练和测试数据集，由直接独立检验因子分析法可得到系统的阶数$\hat{n} = 1$（见图10-2），又由图10-2看出最优的因子数$h^o = 2$。这表明开始的二阶系统是一个可简化系统。因此，传递函数中存在极值消去或零点消去。实际上，初始系统的极值点为：

$$\text{极值}_{2\text{阶系统}} = [0.1, 0.4999] \tag{10-34}$$

零点为

$$\text{零}_{2\text{阶系统}} = [0.1] \tag{10-35}$$

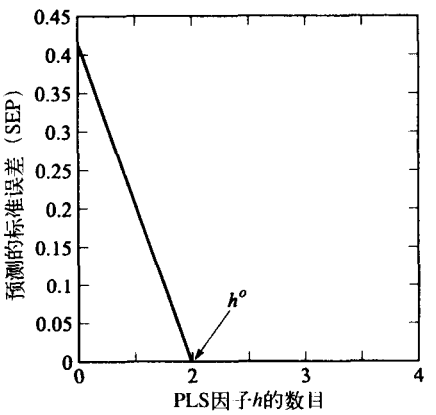



图10-2　由PLSR1选择理想的PLSR因数；$h^o$为2，即系统的阶数$\hat{n} = 1$（由系统的最小实现得出）

因此，在0.1点处极值和零可以消去。如果这个极值/零点被消去，则得到的系统是一个不可简化（可控和可观察）系统。

为了求得系统最小实现的参数，最后的数据集（系统维数采用$\hat{n} = 1$）必须首先生成，如式（10-25）。对最后的数据集运用PLSNET-C，经过3000个训练回合后，计算PLS的权值装载向量、装载向量和回归系数，即$\{\hat{\boldsymbol{w}}_h, \hat{\boldsymbol{b}}_h, \hat{v}_h\}$，$h = 1, 2$。用于PLSNET-C的学习率参数按下式选取：

$$\mu_w = \frac{0.1}{\sum_{j=1}^{N-n+1=100} y_{fj}^2} \tag{10-36a}$$

$$\mu_b = \mu_v = (0.05)\mu_w \tag{10-36b}$$

根据集合$\{\hat{\boldsymbol{w}}_h, \hat{\boldsymbol{b}}_h, \hat{v}_h\}$，在式（10-26）～式（10-28）中给出的三个矩阵$\hat{\boldsymbol{W}}$, $\hat{\boldsymbol{B}}$, $\hat{\boldsymbol{v}}$都可以生成，降

475

阶（最小实现）系统的参数向量由下式给出：

$$\hat{\theta} = \hat{\boldsymbol{b}}_{f\,\mathrm{PLSR}} = \hat{\boldsymbol{W}}^T(\hat{\boldsymbol{B}}\hat{\boldsymbol{W}}^T)^{-1}\hat{\upsilon} = \begin{bmatrix} -0.4999 \\ 1.0 \end{bmatrix} \tag{10-37}$$

这个结果是由式（10-34）～式（10-35）得来。因此，最小实现系统的传递函数由下式给出：

$$H_{mr}(z) = \frac{1}{z - 0.4999} \tag{10-38}$$

利用PLSNET-C的结果$\{\hat{\boldsymbol{w}}_h, \hat{\boldsymbol{b}}_h, \hat{\upsilon}_h\}$做一个简单的检测，图10-3显示了初始（实际）系统的前50次系统输出样本和把PLSNET-P用于一阶系统的系统响应图像。从图上可以明显看出两类响应本质上是一致的。结果相同是由于它们都采用了式（10-31）给出的方法，也就是说，都利用了式（10-37）给出的运算参数向量。

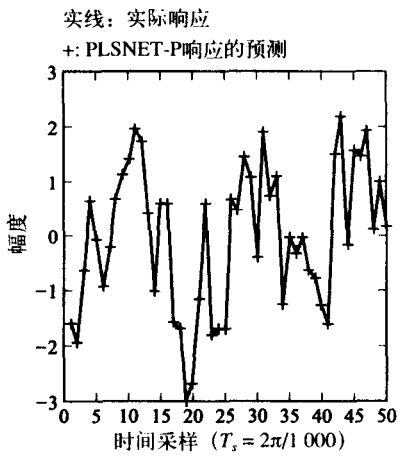



图10-3　实际系统（实线）的输出响应与运用PLSNET-P的降阶（最小实现）系统，它的参数使用PLSNET-C进行估计（图示为离散样本点）

用PLSNET-C和9.5节中给出的PLSR1校准算法比较权值装载向量、装载向量和回归系数，会得到令人有趣的结果。用于PLSNET-C的三个向量由下式给出：

$$\hat{\boldsymbol{W}}_{\mathrm{PLSNET-C}} = [\hat{\boldsymbol{w}}_1, \hat{\boldsymbol{w}}_2]_{\mathrm{PLSNET-C}} = \begin{bmatrix} -0.4815 & 0.9431 \\ 0.8764 & 0.3324 \end{bmatrix}$$

$$\hat{\boldsymbol{B}}_{\mathrm{PLSNET-C}} = [\hat{\boldsymbol{b}}_1, \hat{\boldsymbol{b}}_2]_{\mathrm{PLSNET-C}} = \begin{bmatrix} -0.5331 & 0.8900 \\ 0.8471 & 0.4948 \end{bmatrix}$$

$$\hat{\upsilon}_{\mathrm{PLSNET-C}} = \begin{bmatrix} \hat{\upsilon}_1 \\ \hat{\upsilon}_2 \end{bmatrix}_{\mathrm{PLSNET-C}} = \begin{bmatrix} 1.1136 \\ 0.0500 \end{bmatrix}$$

用PLSR1校准算法的三个向量由下式给出：

$$\hat{\boldsymbol{W}}_{\mathrm{PLSR1}} = [\hat{\boldsymbol{w}}_1, \hat{\boldsymbol{w}}_2]_{\mathrm{PLSR1}} = \begin{bmatrix} -0.4854 & 0.8743 \\ 0.8743 & 0.4854 \end{bmatrix}$$

$$\hat{\boldsymbol{B}}_{\mathrm{PLSR1}} = [\hat{\boldsymbol{b}}_1, \hat{\boldsymbol{b}}_2]_{\mathrm{PLSR1}} = \begin{bmatrix} -0.5329 & 0.8743 \\ 0.8479 & 0.4854 \end{bmatrix}$$

$$\hat{\upsilon}_{\mathrm{PLSR1}} = \begin{bmatrix} \hat{\upsilon}_1 \\ \hat{\upsilon}_2 \end{bmatrix}_{\mathrm{PLSR1}} = \begin{bmatrix} 1.1143 \\ 0.0483 \end{bmatrix}$$

对比数组中的值可以看出，两种方式抽取的PLS信息几乎相同。

## 10.6 非线性系统的表示法

一般来说，任何系统都可以看作一个算子进行两个空间之间的某种映射[11]，图10-4从概念上给以说明。包含系统定义域的空间通常称为输入空间U。类似地，包含系统映射结果的空间称为输出空间Y。在多数实际应用中，空间U和Y是向量空间，分别是$\Re^{m\times 1}$和$\Re^{n\times 1}$的子集。系统的输入和输出都可以定义为时间变量的函数，这样的系统称为动态系统。换句话说，如图10-4所示的动态系统在时刻$t$接收到一个输入$\boldsymbol{u}(t)\subset U$，产生一个输出$\boldsymbol{y}(t)\subset Y$。在许多动态系统中，输出不仅取决于当前的输入变量值，而且依赖于系统过去的输入和输出值。这些系统就是通常说的有记忆的系统。 476 ~ 477

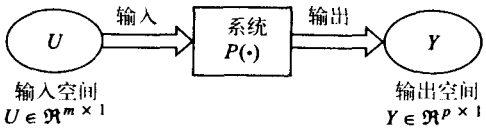



图10-4 动态系统的简单模型

在动态系统学习的有关学科中，准确的系统表示法是一个非常重要的问题，从本质上说，这个问题就是利用数学工具寻找一个简便的方法为系统建模。模型的性能可以从不同的方面来判定，包括准确性、简易性、可计算性和现实有效性。就建立准确的（用数学可处理的）模型来说，它本身是一个非常复杂的科学问题。幸运的是，在实际应用中，我们在建模时很少对系统的每个方面都考虑，这就允许我们建立一个相对简单的模型，它只在实际应用的方面是准确的。

根据向量空间U和Y的映射性质，系统可以在广义上分为两类：线性和非线性。如果图10-4中的映射对于任意的$A, B\in\Re$和$\boldsymbol{u}_1(t), \boldsymbol{u}_2(t)\subset U$都能满足下式：

$$P\{A\boldsymbol{u}_1(t)+B\boldsymbol{u}_2(t)\}=AP\{\boldsymbol{u}_1(t)\}+BP\{\boldsymbol{u}_2(t)\} \tag{10-39}$$

那么系统就是线性的。反之，如果式（10-39）不成立，系统就是非线性的。线性系统理论在许多应用中已经是一门发展很完善的学科。它主要基于线性代数、复变理论、线性算子理论和数学以及科学技术的其他一些领域。但是，非线性系统的表示、设计、辨识和控制却是很具挑战性的工作，还需要进一步研究。在本节中，我们提出两种常用的非线性动态系统的表示法。

### 10.6.1 非线性输入-状态-输出表示法

很多类型的非线性动态系统可由下式表达：

$$\frac{d\boldsymbol{x}(t)}{dt}=\dot{\boldsymbol{x}}(t)=f[\boldsymbol{x}(t),\boldsymbol{u}(t)] \tag{10-40}$$

$$\boldsymbol{y}(t)=g[\boldsymbol{x}(t),\boldsymbol{u}(t)] \tag{10-41}$$ 478

其中

$$\text{状态向量 } \boldsymbol{x}(t)=[x_1(t),u_2(t),\cdots,x_n(t)]^T\in\Re^{n\times 1}$$

$$\text{输入向量 } \boldsymbol{u}(t)=[u_1(t),u_2(t),\cdots,u_m(t)]^T\in\Re^{m\times 1}$$

$$\text{输出向量 } \boldsymbol{y}(t)=[y_1(t),y_2(t),\cdots,y_p(t)]^T\in\Re^{p\times 1}$$

$$t=\text{连续时间变量}$$

函数$f(\cdot)$和$g(\cdot)$表示由$f:\Re^{(n+m)\times 1}\to\Re^{n\times 1}$和$g:\Re^{(n+m)\times 1}\to\Re^{p\times 1}$定义的非线性映射。如果函数

$g[\boldsymbol{x}(t), \boldsymbol{u}(t)]$中明确地包含$\boldsymbol{u}(t)$，我们说这个系统的输入和输出是直接连接。但对于大多数系统来说不是直接连接，通常函数$g(\cdot)$假设只是一个系统的状态函数。

式（10-40）和式（10-41）通常认定为非线系统的输入−状态−输出（input-state-output，ISO）模型。很明显，式（10-40）和式（10-41）中的表达式可用来描述连续动态系统。如果是离散时间系统，则式（10-40）中的微分方程式需要转换为对应的差分方程式。因此，在一个离散时间系统中，ISO表示法的形式如下式：

$$\boldsymbol{x}(k+1) = f[\boldsymbol{x}(k), \boldsymbol{u}(k)] \tag{10-42a}$$

$$\boldsymbol{y}(k) = g[\boldsymbol{x}(k), \boldsymbol{u}(k)] \tag{10-42b}$$

如果映射函数$f(\cdot)$和$g(\cdot)$是线性的，式（10-42a）和式（10-42b）可化简变换为线性系统ISO表示的通常形式如下：

连续型：

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t) \tag{10-43}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}(t)\boldsymbol{x}(t) + \boldsymbol{D}(t)\boldsymbol{u}(t) \tag{10-44}$$

离散型：

$$\boldsymbol{x}(k+1) = \boldsymbol{A}(k)\boldsymbol{x}(k) + \boldsymbol{B}(k)\boldsymbol{u}(k) \tag{10-45}$$

$$\boldsymbol{y}(k) = \boldsymbol{C}(k)\boldsymbol{x}(k) + \boldsymbol{D}(k)\boldsymbol{u}(k) \tag{10-46}$$

其中，$\boldsymbol{A}\in\Re^{n\times n}$，$\boldsymbol{B}\in\Re^{n\times m}$，$\boldsymbol{C}\in\Re^{p\times n}$，$\boldsymbol{D}\in\Re^{p\times m}$。当时间是常量时，系统变为最简单的形式，$\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$和$\boldsymbol{D}$是常实数矩阵（详见A.2.12节）。

### 10.6.2 非线性ARMA

以ISO形式建立非线性系统表示需要了解关于系统内部运行过程的充足知识。显而易见，为了说明式（10-42a）和式（10-42b）中的离散时间状态表示必须了解系统的状态。虽然多数情况下这些状态是现成的，但仅根据系统本身的输入输出建立一个非线性系统表示也是非常有用的。Leontaritis和Billings [12] 介绍了另外一种方法来建立非线性动态系统模型：外
479 输入非线性自回归滑动平均（NARMAX）模型。NARMAX模型可以认为是非常成功的ARMAX模型在非线性动态系统情形下的一个自然扩展。为了阐明这一点，考虑一个线性动态系统的ARMAX模型如下：

$$\boldsymbol{y}(k) = \sum_{i=1}^{n_y} \theta_{yi}\boldsymbol{y}(k-i) + \sum_{i=1}^{n_u} \theta_{ui}\boldsymbol{u}(k-i) + \sum_{i=1}^{n_e} \theta_{ei}\boldsymbol{e}(k-i) + \boldsymbol{e}(k) \tag{10-47}$$

其中$\boldsymbol{y}(k)\in\Re^{p\times 1}$、$\boldsymbol{u}(k)\in\Re^{m\times 1}$和$\boldsymbol{e}(k)\in\Re^{p\times 1}$分别是线性系统在离散时间的时刻$k$的输出向量、输入向量和测量（或近似）误差向量。同样有$\boldsymbol{\theta}_{yi}\in\Re^{r\times p}$、$\boldsymbol{\theta}_{ui}\in\Re^{p\times m}$和$\boldsymbol{\theta}_{ei}\in\Re^{p\times p}$。如果假定系统无噪声，则ARMAX模型就简化为在10.3节和10.4节中所讨论的比较简单的ARMA模型。

在适当的假定下，一大类非线性系统都可以用下面的非线性离散时间差分方程表示[12]：

$$\begin{aligned}\boldsymbol{y}(k) = f[&\boldsymbol{y}(k-1),\cdots,\boldsymbol{y}(k-n_y),\boldsymbol{u}(k-1),\cdots,\boldsymbol{u}(k-n_u),\\ &\boldsymbol{e}(k-1),\cdots,\boldsymbol{e}(k-n_e)] + \boldsymbol{e}(k)\end{aligned} \tag{10-48}$$

其中$\boldsymbol{y}(k)\in\Re^{p\times 1}$、$\boldsymbol{u}(k)\in\Re^{m\times 1}$和$\boldsymbol{e}(k)\in\Re^{p\times 1}$和式（10-47）中意义一样，$f(\cdot)$是一个非线性映射，定义为$f:\Re^{(n_y\cdot p+n_u\cdot m+n_e\cdot p)\times 1}\to\Re^{p\times 1}$。式（10-48）中的矩阵方程可以分解为关于标量$p$的方程如下：

$$\begin{aligned} y_i(k) = f_i[&y_1(k-1),\cdots,y_1(k-n_y),y_2(k-1),\cdots,y_2(k-n_y),\cdots,y_p(k-1),\cdots,y_p(k-n_y),\\ &u_1(k-1),\cdots,u_1(k-n_u),u_2(k-1),\cdots,u_2(k-n_u),\cdots,u_m(k-1),\cdots,u_m(k-n_u),\\ &e_1(k-1),\cdots,e_1(k-n_e),e_2(k-1),\cdots,e_2(k-n_e),\cdots,e_p(k-1),\cdots,e_p(k-n_e)]\\ &+e_i(k) \end{aligned} \tag{10-49}$$

其中$i = 1, 2, \cdots, p$。一个非常重要的特例是，当系统的输出假定为没有误差时，NARMAX模型就简化为简单的外输入非线性自回归（NARX）模型，可以写成如下形式：

$$\boldsymbol{y}(k) = f[\boldsymbol{y}(k-1),\boldsymbol{y}(k-2),\cdots,\boldsymbol{y}(k-n_y),\boldsymbol{u}(k-1),\boldsymbol{u}(k-2),\cdots,\boldsymbol{u}(k-n_u)]+\boldsymbol{e}(k) \tag{10-50}$$

或标量形式如下：

$$\begin{aligned} y_i(k) = f_i[&y_1(k-1),\cdots,y_1(k-n_y),y_2(k-1),\cdots,y_2(k-n_y),\cdots,y_p(k-1),\cdots,\\ &y_p(k-n_y),u_1(k-1),\cdots,u_1(k-n_u),u_2(k-1),\cdots,u_2(k-n_u),\cdots,u_m(k-1),\cdots,\\ &u_m(k-n_u)]+e_i(k) \end{aligned} \tag{10-51}$$

以后可以看到，NARMAX模型的NARX形式对于非线性系统辨识是非常重要的。 480

式（10-51）中给出的NARX形式是非常常见的，它包含的非线性动态系统范围很广。然而，在实际应用中，非线性函数$f(\cdot)$几乎是不可知的，因此，在系统辨识过程中需要建立它的近似式。解决这个问题的一种方法就是根据著名的斯通-魏尔斯特拉斯（Stone-Weierstrass）定理[13]：任何函数都能用多项式以任意要求的精度逼近。把式（10-51）展开成$l$次多项式，就得到下式：

$$\begin{aligned} y_i(k) = \theta_0^{(i)} &+ \sum_{i_1=1}^{n}\theta_{i_1}^{(i)}x_{i_1}(k) + \sum_{i_1=1}^{n}\sum_{i_2=i_1}^{n}\theta_{i_1i_2}^{(i)}x_{i_1}(k)x_{i_2}(k)\\ &+\sum_{i_1=1}^{n}\sum_{i_2=i_1}^{n}\cdots\sum_{i_l=i_{l-1}}^{n}\theta_{i_1i_2\cdots i_l}^{(i)}x_{i_1}(k)x_{i_2}(k)\cdots x_{i_l}(k)+e_i(k) \end{aligned} \tag{10-52}$$

其中$n = pn_y + mn_u$，

$$x_1(k) = y_1(k-1), x_2(k) = y_1(k-2),\cdots, x_{pn_y}(k) = y_p(k-n_y)$$

和

$$x_{pn_y+1}(k) = u_1(k_1-1), x_{pn_y+2}(k) = u_1(k-2),\cdots, x_n(k) = u_1(k-n_u)$$

式（10-52）中的多项式扩展把式（10-51）中确定未知函数的问题转变成一个参数估计的问题。为了说明以上过程，举例如下：

**例10.2** 按照式（10-50），一个单输入单输出系统的NARX模型表示为下式：

$$y(k) = f[y(k-1), y(k-2), u(k-1)] \tag{10-53}$$

我们为式（10-53）中的非线性函数建立一个近似多项表达式，假设它能用2次多项式准确逼近。那么，按照式（10-52）我们能得到下式：

$$\begin{aligned} y(k) = \theta_0 &+ \theta_1y(k-1)+\theta_2y(k-2)+\theta_3u(k-1)\\ &+\theta_4y^2(k-1)+\theta_5y(k-1)y(k-2)+\theta_6y(k-1)u(k-1)\\ &+\theta_7y^2(k-2)+\theta_8y(k-2)u(k-1)+\theta_9u^2(k-1) \end{aligned} \tag{10-54}$$

从例10-2的简单非线性系统能看到NARX模型存在一个明显的问题。由于非线性函数$f(\cdot)$

的结构是未知的，在分析时多项式的所有项都需要包括。一般来说，这意味着对于一个具有$p$个输出和$m$个输入且输出和输入的最大延迟数分别为$n_y$和$n_u$，它的$l$次非线性多项式逼近，我们需要进行估计的总项数为

481

$$N = p[1 + n + n(n-1) + \cdots + n(n-1)(n-2)\cdots(n-l+1)] = p\sum_{j=0}^{l} j!\binom{n}{j} \tag{10-55}$$

在式（10-55）中，$n = mn_u + pn_y$。举例来说，由$n_u = n_y = 4$给出的最大延迟的单输入单输出系统的4次多项式逼近需要估计

$$N = \sum_{j=0}^{4} j!\binom{8}{j} = 2\,081 \tag{10-56}$$

个系数。而且，除去高阶非线性系统，大多数系数的值都非常小，也就是说式（10-52）中大多数项是多余的，应该删除。如果只从系统辨识角度看，系数的数量多不会造成难题。大多数情况下，辨识过程是非线性设计和自适应控制的第一个步骤。但从实用的角度看，系数的数量多对系统的表示会造成非常大的困难。对这个问题已经提出几种解决方案，这里只讨论其中的两种。

*应用正交分解的线性回归*

对式（10-52）仔细检验后，可以看出它还能写为更简单的形式如下：

$$\boldsymbol{x}^T(k)\theta^{(i)} = y_i(k) \tag{10-57}$$

其中

$$\boldsymbol{x}(k) = [1, x_1(k), \cdots, x_n(k), x_1^2(k), \cdots, x_n^2(k), x_1^l(k), \cdots, x_n^l(k)]^T \tag{10-58}$$

和

$$\theta^{(i)} = \left[\theta_0^{(i)}, \theta_1^{(i)}, \cdots, \theta_n^{(i)}, \theta_{11}^{(i)}, \cdots, \theta_{nn}^{(i)}, \cdots, \underbrace{\theta_{11\cdots1}^{(i)}}_{l}, \cdots, \underbrace{\theta_{nn\cdots n}^{(i)}}_{l}\right]^T \tag{10-59}$$

设$M$为连续的时间间隔，方程组能写为如下形式：

$$[\boldsymbol{x}(k), \boldsymbol{x}(k-1), \cdots, \boldsymbol{x}(k-M)]^T\theta^{(i)} = [\boldsymbol{y}_i(k), \boldsymbol{y}_i(k-1), \cdots, \boldsymbol{y}_i(k-M)] \tag{10-60}$$

等式（10-60）能写为更简约的形式如下：

$$\boldsymbol{P}\boldsymbol{\theta}^{(i)} = \boldsymbol{Y}_i \tag{10-61}$$

其中$\boldsymbol{P} \in \Re^{M+1\times N}$，$\boldsymbol{\theta}^{(i)} \in \Re^{N\times 1}$，$\boldsymbol{Y}_i \in \Re^{M+1\times 1}$。$N$是每个输出要估计的系统参数总数，由下式给出：

482

$$N^{(i)} = \sum_{j=0}^{l} j!\binom{n}{j} \tag{10-62}$$

和

$$n = mn_u + pn_y \tag{10-63}$$

因此，关于NARX模型多项式扩展中的参数估计问题就转变为式（10-61）中的线性回归问题。在文献[14]中，Billings等验证了一种可靠的回归技术，它能得到关于NARX模型多项式近似的参数。参考文献[15]中给出了一种稍加改动的算法，它运用了投影。文献[16]中Chen等人给出了一篇优秀的论文，对NARX非线性系统建模的正交最小二乘法和应用进行了全面的调查研究。

*神经网络方法*

把神经网络用于建立NARX模型的非线性映射近似是多项式扩展中参数估计的另一可选方案。任何能模拟非线性映射近似的神经网络都可用来学习函数$f(\cdot)$。在参考文献[11, 17, 18]中论证了多层感知器网络和递归神经网络都能完成这个任务，在参考文献[19, 20]中使用径向基函数神经网络也达到了这一目的。

作为最后的定论，Narendra和Parthasarathy[11]证明，如果非线性系统中存在已知数据，非线系统建模最好是组合NARMAX和ARMAX。在参考文献[11]中特别指出，一个非线性系统能辨识为下面模型之一。

**模型1**. 系统的输出非线性依赖于以前的$n_u$个输入值，线性依赖于以前的$n_y$个输出值，它可写为如下形式：

$$\boldsymbol{y}(k)=\sum_{i=1}^{n_y}\alpha_i\boldsymbol{y}(k-i)+f_u[\boldsymbol{u}(k-1),\boldsymbol{u}(k-1),\cdots,\boldsymbol{u}(k-n_u)]+\boldsymbol{e}(k) \tag{10-64}$$

其中，$\alpha_i\in\Re, f_u:\Re^{m\times n_u}\to\Re^{p\times 1}$。

**模型2**. 系统的输出线性依赖于$n_u$个以前的输入值、非线性依赖于$n_y$个以前的输出值，它可写为如下形式：

$$\boldsymbol{y}(k)=f_y[\boldsymbol{y}(k-1),\boldsymbol{y}(k-2),\cdots,\boldsymbol{y}(k-n_y)]+\sum_{i=1}^{n_u}\boldsymbol{B}_i\boldsymbol{u}(k-i)+\boldsymbol{e}(k) \tag{10-65}$$

其中 $\boldsymbol{B}_i\in\Re^{p\times m}$ 并且 $f_y:\Re^{p\times n_y}\to\Re^{p\times 1}$。

**模型3**. 系统的输出非线性依赖于以前的输入和输出值，但非线性映射是可分离的，它可写为如下形式：

483

$$\begin{aligned}\boldsymbol{y}(k)=&f_y[\boldsymbol{y}(k-1),\boldsymbol{y}(k-2),\cdots,\boldsymbol{y}(k-n_y)]\\&+f_u[\boldsymbol{u}(k-1),\boldsymbol{u}(k-2),\cdots,\boldsymbol{u}(k-n_u)]+\boldsymbol{e}(k)\end{aligned} \tag{10-66}$$

其中 $f_y:\Re^{p\times n_y}\to\Re^{p\times 1}$ 并且 $f_u:\Re^{m\times n_u}\to\Re^{p\times 1}$。

**模型4**. 系统的输出与以前的输入和输出值之间的依赖性用不可分离的非线性形式表示。因此，系统的输出可写为如下形式：

$$\boldsymbol{y}(k)=f[\boldsymbol{y}(k-1),\cdots,\boldsymbol{y}(k-n_y),\boldsymbol{u}(k-1),\cdots,\boldsymbol{u}(k-n_u)]+\boldsymbol{e}(k) \tag{10-67}$$

其中 $f:\Re^{(n_y p+n_u m)\times 1}\to\Re^{p\times 1}$。很明显，模型4是以前章节中讨论的NARX模型的常用情形，模型1～3都来源于模型4，都是非线性函数$f(\cdot)$的一种特殊形式。

## 10.7 非线性动态系统的辨识和控制

辨识和控制是数学系统理论的两大基础工作。对于线性系统来说，已经有大量的数学工具可供应用，然而对于非线性系统来说，辨识和控制仍很困难。由于缺乏通用的理论来建立具有可控性、可观察性和稳定性等性质的非线性系统，迫使我们逐个地分析研究非线性系统辨识和控制的基本事例。最近，应用神经网络来处理非线性系统的辨识和控制已经取得了可喜的成果 [11, 17-21] 。从实践和理论两方面看，神经控制将是神经网络应用方向上一个很有发展前景的领域。

在本节中，我们给出应用神经网络进行非线性系统辨识和控制的一些简单方法。首先我们讨论系统辨识的问题，然后我们讨论一些非线性控制设计的方法。为了讨论起来简单，我们只针对单输入单输出（SISO）系统。但这些方法都可以直接扩展应用于多输入多输出（MIMO）系统。

### 10.7.1 非线性系统的辨识

如图10-5所示为非线性系统辨识的一般问题。辨识模型的参数是自适应地进行估计，所以实际系统输出和模型的输出之间的差别是最小的。理想状态下，即使没有动态系统的先验
484 知识，这样的辨识过程也能得到非线性系统的精确模型。实质上，在辨识过程中模型的结构和参数都在进行调整。但即使能这样做，产生的模型也仅能对辨识过程中用到的输入序列有效，如果在识别过程中不用一个特殊的输入序列，系统和它的模型就不一定能达到期望的精确度的结果。

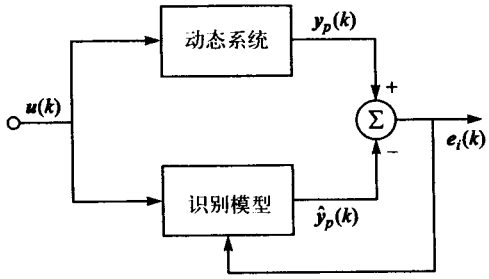



图10-5　系统辨识过程的一般模型

一种用于非线性系统辨识的最有用的方法基于NARMAX建模。图10-6a和10-6b给出了采用NARMAX建模方法来实现图10-5所示识别方案的两种不同实现。从图上可以看出，两种配置完全相似。事实上，非线性识别系统的结构也完全相同，不同点仅是系统和辨识器之间的连接。图10-6a的结构是通常所说的*并行配置*，图10-6b的结构称为*并行－串行配置*。

*并行配置*

图10-6a和图10-6b所示的辨识模型中都有一个重要的设定是：对辨识过程所用的所有输入序列，非线性系统都是有界输入有界输出（BIBO）稳定的。在图10-6a的并行配置中把辨识器的过去值作为NARMAX模型的输出延迟，参看式（10-48），并行配置应用NARMAX模型的最常用形式。由于每个输出样本都有一个预测误差反馈给NARMAX模型，即使系统是BIBO稳定的，也不能保证辨识模型是稳定的。使并行辨识过程具有稳定性的确切条件尚未可知，即使线性系统也是这样[11]，所以并行配置在实践中很少应用。

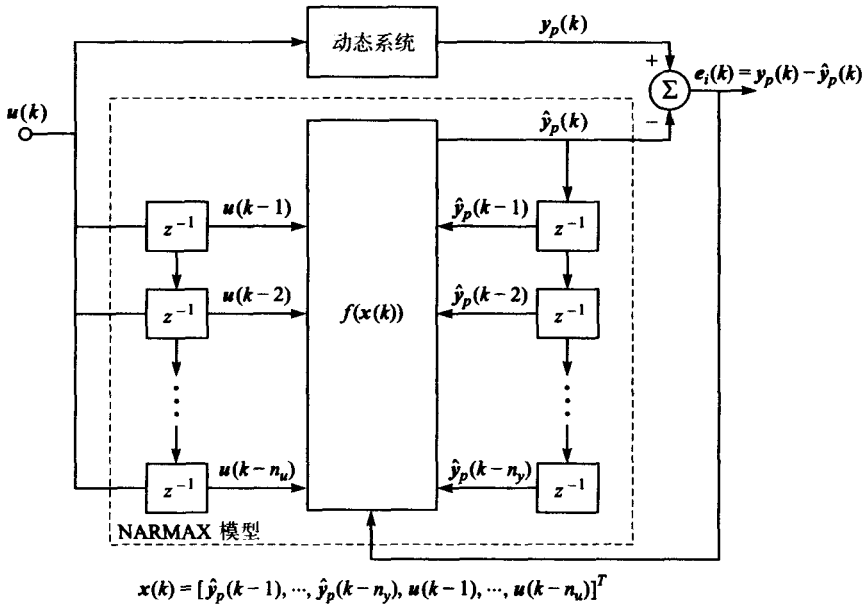
*并行－串行配置*

不同于并行配置模型，在并行−串行配置模型中，实际系统输出的过去值在辨识过程中应用。由于系统设定为BIBO稳定的，所以，辨识过程中用到的所有信号也是有界的。因此，用
485 实际输出来消去计算误差，则NARMAX模型就简化为更易于控制的NARX模型（见前一节讨论）。基于以上的考虑，把并行−串行配置作为非线性系统辨识的可选方式。

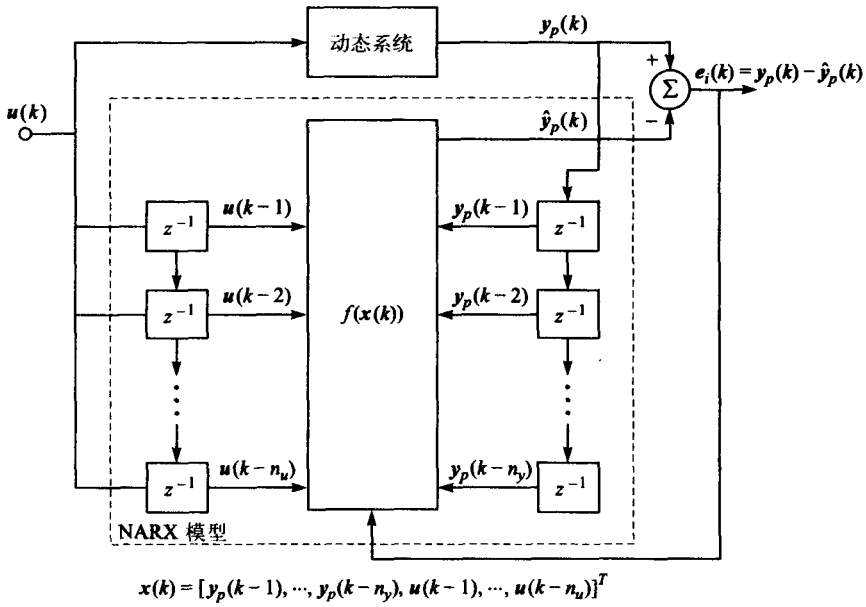
图10-6b中并行−串行配置辨识器的最重要部分是非线性映射。对于SISO系统，由下式给出映射方式：

$$y(k)=f[y(k-1),\cdots,y(k-n_y),u(k-1),\cdots,u(k-n_u)] \tag{10-68}$$

由于神经网络结构的全局逼近的性质，在实现式（10-68）的映射时，主要选择神经网络。不同的网络结构都可用来处理式（10-68）的映射[11-15, 18-20]。选择哪一类型的神经网络主要决定于辨识问题的性质。在此，我们只讨论两类最流行的网络结构：多层感知器网络（MLP NN）和径向基神经网络（RBF NN）。




a)




b)

图10-6 a) 系统辨识的并行配置；b) 系统辨识的并行−串行配置

应用MLP NN辨识非线性系统

图10-7给出了应用MLPNN逼近并行-串行NARX辨识模型中的非线性映射。用于网络训练的学习算法是通用的反向传播。用于调整网络权值的是两种不同的常用方法，第一种称为模式学习[18]，按照这种方法，网络的权值随每一次的输入而更新。第二种方法也是一种模式学习的可选方案，称批量学习。采用这种方法时，网络要得到一系列的输入后才更新网络权值。在第3章中（详见3.3.4节）讨论了这两种方法以及它们的折中方法。关于这两种学习方法的更详细讨论参考文献[18, 21]。从实际应用看，仿真模拟表明这两种学习方法在学习率参数很小的情况下，它们的性能是不相上下的[18]。但随着学习率参数的增加，批量学习表现为收敛不一致，而模式学习的收敛保持相对一致。运用MLP NN来模拟式（10-68）中的映射关系的最大优点是它的简便性和适于在线应用。另一方面，反向传播学习带来了辨识过程中收敛速度的问题，一般来说，第3章中提到的任何一种加速反向传播学习的方法都能增加它的收敛速度。但通常是以增加学习过程中的计算复杂度为代价的。

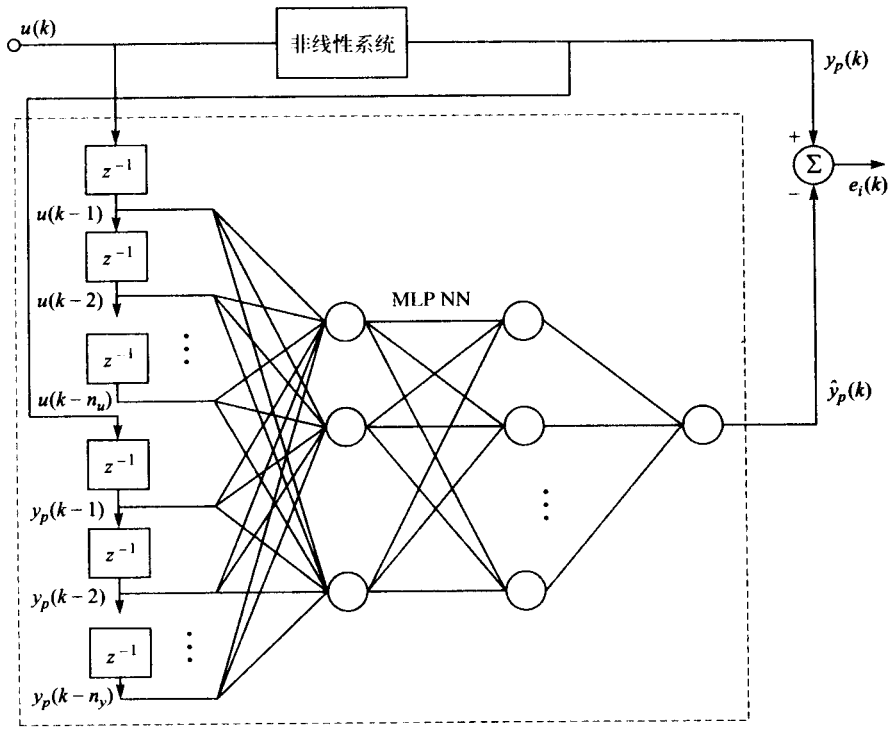



图10-7　运用MLP NN实现并行-串行NARX系统辨识

**例10.3**　假设一个SISO非线性系统的系统辨识问题由下面的ISO表达式给出：

状态方程：

$$x_1(k+1)=\frac{x_2(k)}{x_1(k)+4} \tag{10-69}$$

$$x_2(k+1)=\tanh\{x_1(k)+[1+x_2(k)]u(k)\} \tag{10-70}$$

输出方程：

$$y(k)=2x_2(k) \tag{10-71}$$

用于图10-7中给出的并行－串行配置来完成辨识，NARX模型由下式给出：

$$y(k)=f[u(k-1),\cdots,u(k-4),y(k-1),\cdots,y(k-4)] \tag{10-72}$$

建立的多层感知器网络（MLP NN）有两个隐藏层，每层有10个神经元。在辨识过程中，区间[－0.5, 0.5]中均匀分布的噪声序列作为系统的输入。用反向传播作为训练神经网络的辨识器。图10-8a、b给出了用方波形作为输入时测试系统和NARX模型的输出图像。图10-8a中方波输入的峰值是0.2，从图可以看出，NARX模型准确地模拟了非线性系统的表现，这种结果也是意料之中的，因为波形的幅度在训练网络的随机噪声范围之内。图10-8b显示了当方波峰值为2时，系统和NARX模型的输出图像。从图上可以看出，实际非线性系统与模型的输出之间有很大不同。这是因为输入的波形幅度超出了建立NARX模型时的输入模式范围。因此，在非线性系统辨识过程中，模型只表现出系统局部的属性。一般来说，当超出了建模时的输入范围时，网络的精确度是不可信赖的。

486 ~ 488

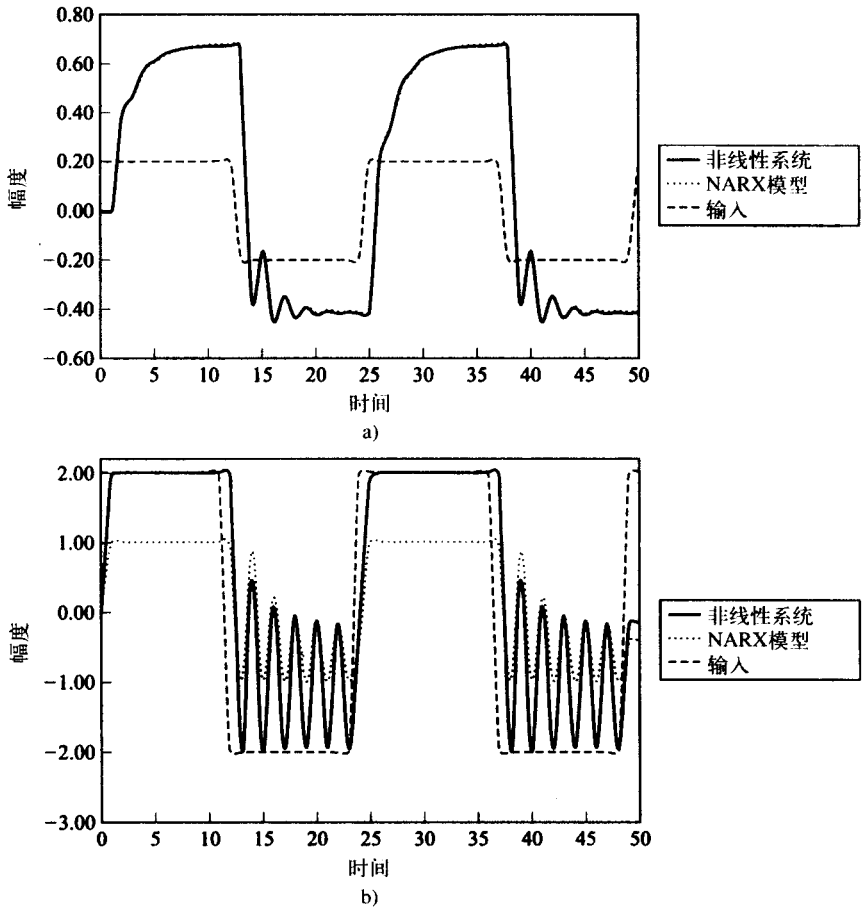



图10-8　a) 例10.2系统辨识过程的仿真，方波输入峰值为0.2时的响应；b) 方波输入峰值为2时的响应

*应用RBF NN辨识非线性系统*

图10-9为运用RBF NN进行非线性系统辨识过程的描述。从图上可以看出，辨识器的结构和图10-7中MLP NN的结构非常相似。可用几种不同的方法训练图10-9中的RBF NN，当在线

辨识是最看重的性能时，可用第3章（详见3.6.2节）中介绍的随机梯度方法来生成RBF NN的
489 三个参数集：权值、中心和扩展参数集。这样虽然很直接，但这种方法不能发挥RBF NN的最好特性，即线性“非参数”结构。在3.6.1节说明了一旦网络的中心和扩展参数选定，网络的权值就可作为一个线性回归问题的解，也就是用“封闭的形式”。而且，选择该网络的中心的最好方式是正交最小二乘（OLS）法（详见3.6.3节），这样能使RBF NN的辨识器相对较小。不使用随机梯度方法的代价是既不是固定中心的方法，也没有适于在线辨识的OLS回归法。另外，这些方法的计算复杂度要比随机梯度法高出许多。

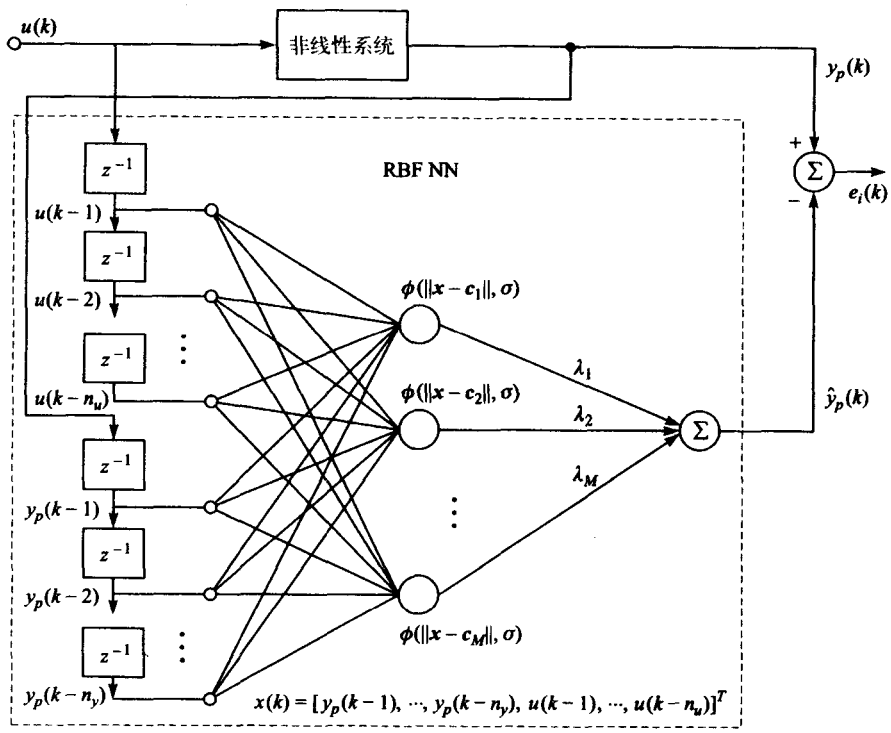



图10-9　运用RBF NN实现并行-串行NARX系统

**例10.4**　假定动态系统和例10.3中的相同。现在使用RBF NN来模拟NARX辨识器中的非线性映射。图10-10a和图10-10b给出了应用RBF NN时系统辨识过程的结果，其中RBF NN的隐藏层有30个神经元，使用高斯径向基函数，扩展参数设为1。网络用OLS前向回归方式训练（详见3.6.3节），从图10-10a可以看出NARX模型在方波的峰值是0.2时，可以有效地预测出非
490 线性系统的输出；从图10-10b能看出在输入方波峰值为2时，应用RBF NN的NARX模型不能模拟非线性系统的过程。虽然这也能用例10.3的原因来解释，但我们看到这两个网络操作的方式是完全不同的。在例10.3中的MLP NN达到了饱和，而用RBF NN时，当输入数量为2时，网络只产生0输出，因为这超出了RBF NN的映射区域。

最后注意一点，用于非线性系统辨识的形式与第10.6节描述的模型4相对应，这是NARX模型常用的形式，绝大多数非线性动态系统都是用这种形式来建模。然而，许多实例又能充分地表明，有时用NARX模型中较简单的形式来构建非线性动态系统的模型是正确的。

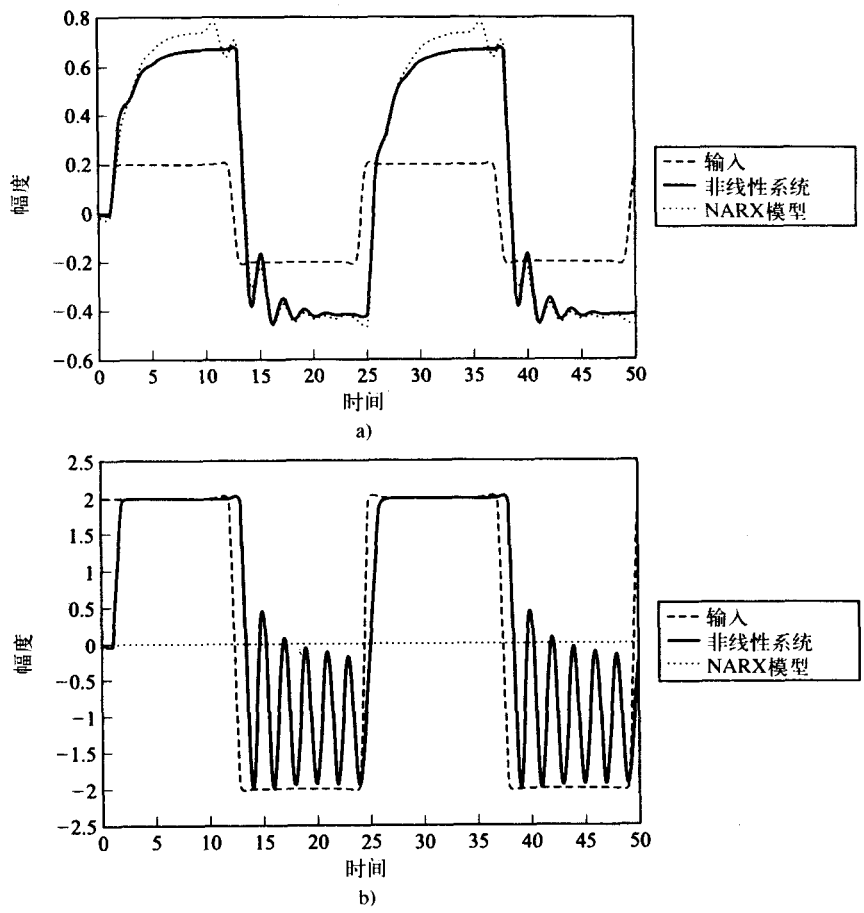



图10-10　a) 例10.3系统辨识过程的仿真。方波输入峰值为0.2时的响应；b) 方波输入峰值为2时的响应　491

### 10.7.2　非线性控制

非线性系统控制的主要目的是把动态系统的输出维持在特定的界限内。我们来看图10-11描述的情况，非线性系统有外部输入$r(t)$。假设系统的期望输出是知道的，由时间函数$y_d(t)$给出，那么非线性控制系统的目标就是：生成它的输入信号$u(t)$，使系统的实际输出$y_p(t)$和期望输出$y_d(t)$的差别保持在一个特定的界限。根据期望输出的性质，我们把控制问题分为两类：

1. 如果$y_d(t)$是常数，控制问题就是通常说的规范问题。

2. 如果$y_d(t)$是非常数的时间函数，控制问题就指跟踪问题。

在多数情况下，动态系统预先是未知的，控制器的设计必须采用自适应的方式。控制算法必须利用非线性系统的输入和输出，根据它们的变化来改变控制器和输入信号$u(t)$的内部参数，这样才能实现控制的目的。

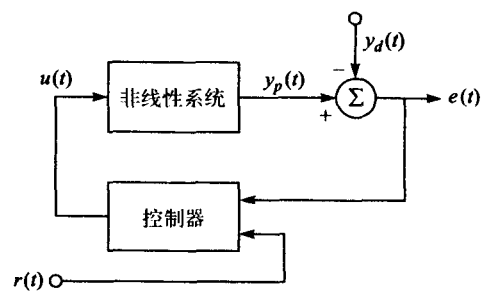



图10-11　一般的非线性控制问题

*自适应参考控制模型*（Model reference adaptive control, MRAC）

对图10-11所示的自适应控制系统，假定非线性系统的期望输出是已知的。通常被控的非线性系统

的期望输出指定一个参考模型。参考模型的用途由图10-12说明。这个模型是一个已知的动态系统，对给定的输入信号$r(t)$产生一个期望的输出。控制设计的目的就是对于给定的输入把参考模型和非线性系统的输出之间的差别最小化。通常参考模型既可选择线性的，也可选择非线性的。但从
[492] 控制器设计的角度看，采用线性模型有明显的好处，因为这样能利用线性系统理论中有效的工具。

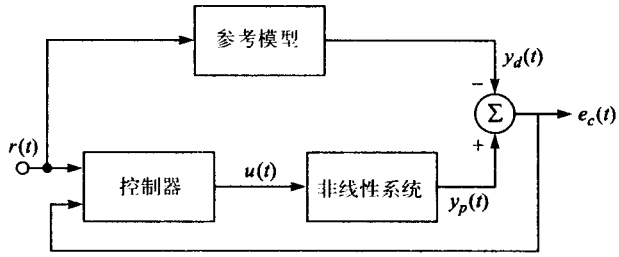



图10-12　MRAC框图

直接和间接控制

习惯上，动态系统自适应控制使用两种不同的方式：直接和间接控制。图10-12显示了运用直接控制的一个系统。根据系统实际输出和期望输出的差别，用直接的方式调整控制器自身的参数。另一方面，在间接控制中，系统实际输出和期望输出的差别在系统的辨识过程中运用。控制器的设计是以在辨识过程中建立的系统模型为基础。图10-13给出了一个间接控制的系统的框图。在开始时，图10-12中的直接自适应控制器结构看起来更吸引人，因为在这个结构中不存在系统辨识过程。但是，不存在基于输入$r(t)$和输出误差$e_c(t)$的控制器参数调整方法。因为基于误差输出和控制器输出的动态系统是未知的。而如果使用间接控制，非线性系统和参考模型的误差就能通过已知的动态系统模型反向传播，并更新控制器的参数。

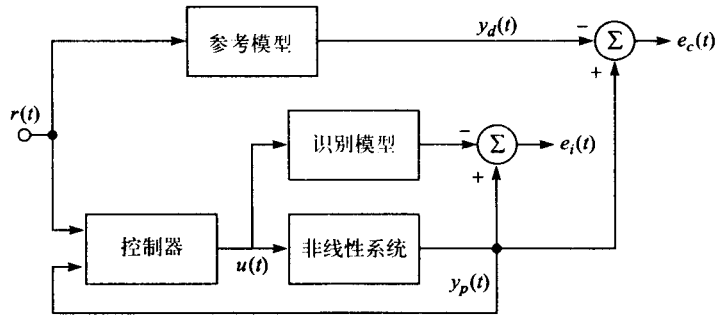



图10-13　间接自适应控制方式的框图

[493] 直接和间接控制在线性系统中运用非常成功，关于这两种方法的详细叙述见参考文献[22]。辨识模型的结构和控制器的设计本质上是一样的，两者最大的区别是辨识模型中的线性增益在控制器设计中由非线神经网络映射代替。

神经网络进行间接控制

正如以前提到的，非线性系统的直接自适应控制非常难实现，目前，对于直接自适应控制还没有有效的方法[11]。用于非线性系统控制的大多数神经网络训练算法使用误差信号的反向传播，误差信号能反映出实际系统和期望系统的响应差值。当前，对于未知的非线性动态系统，还没有通用的理论支持误差信号的反向传播。在这些理论出现之前，用于非线性系统自适应控制的主要方法还是间接控制。

间接控制的第一步就是非线性系统辨识。一个常用的方法是把非线性系统辨识为第10.6节中描述的四种非线性系统模型之一。一旦辨识模型能足够准确地模拟非线性系统，它就能在MRAC控制器中用来训练过程。因为动态神经网络系统辨识模型已知，误差信号能反向传播和调整神经网络控制器的权值。图10-14给出一个间接神经网络控制器的框图。

图10-14中所示的间接控制使用的神经网络常用类型是MLP NN和RBF NN。一般来说，可以使用实现非线性映射的任何一种神经网络。Narendra和Parthasarathy[11]也讨论了递归网络的运用（详见第5章）。

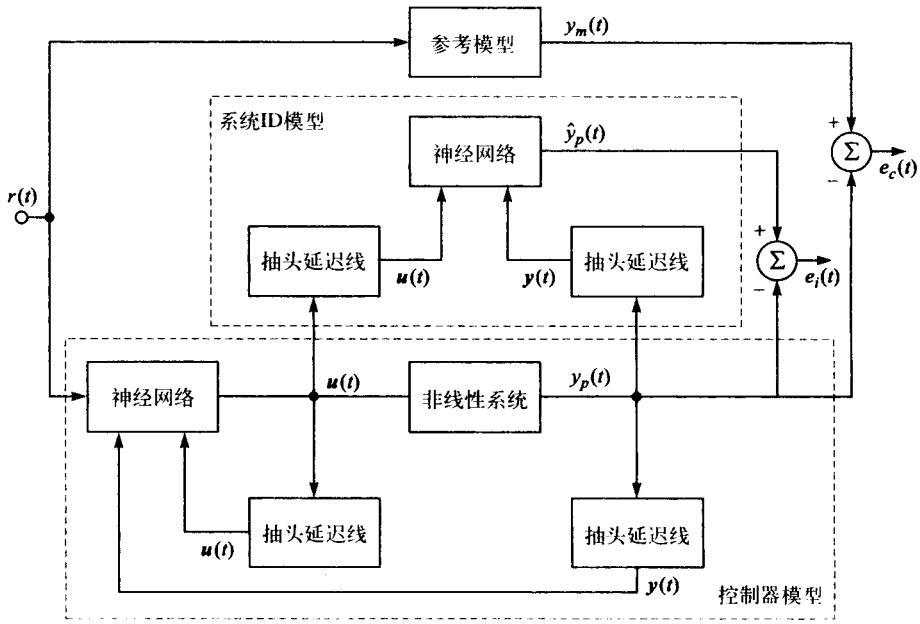



图10-14　运用神经网络的间接控制器的框图

494

确保图10-14中所示的控制过程稳定是一件非常困难的事，因为系统是非线性的，不能应用线性系统中的可控性和可观察性的概念。而且控制器的设计必须认真权衡估计，多数情况下建立在试凑基础上。图10-14中神经网络的选择和控制器设计中所用的方法依赖于对系统运行状态的了解。例如，如果系统在BIBO下是稳定的，系统辨识就能脱机进行，可以运用10.7节中讨论的方法。

如果对动态系统的了解是有限的，辨识和控制都必须在线操作，而且必须同时进行。在下面这些情况中，如表示复杂非线性系统的整个结构和系统的单个组成部分的稳定性不能确保整个系统的稳定。在系统辨识和控制器模型中要特别注意权值的调整率。一般，在参考文献[11]中的仿真报告表明，对于在线控制的稳定性和效率来说，辨识过程必须足够准确。这个要求可由下面两种方法完成：

1. 系统网络辨识时的权值更新率要比控制器网络时的权值更新率大。

2. 在辨识误差值低于一个特定值前，网络控制器不进行权值更新。

这两种方法更适应于实例依赖。

现在我们复习两个间接非线性控制的例子，它们都使用基于神经网络的系统辨识和控制器结构方式，目的是说明控制器设计的过程和阐明平时碰到的一些问题。

**例10.5　对非线性动态系统模型2进行MRAC控制。**

设定一个由下面差分方程给出的离散非线性系统：

$$y_p(k+1)=\frac{y_p(k)+y_p(k-1)}{1+y_p^2(k)+2y_p^2(k-1)}+u(k) \tag{10-73}$$

输入/输出系统表示对应于10.6节讨论的模型2。在控制器设计的过程中，假定输出方程式如下：

$$y_p(k+1)=f(y_p(k),y_p(k-1))+u(k) \tag{10-74}$$

其中函数$f(\cdot)$是未知的。指定系统期望的参考模型由下式给出：

495

$$y_m(k+1)=0.5y_m(k)+0.3y_m(k-1)+r(k) \tag{10-75}$$

其中$r(k)$是参考模型的输入。可以看出，这个参考模型是稳定的线性动态系统，在任意时刻$k$，模型和实际系统的输出误差由下式给出：

$$e_c(k+1)=y_p(k+1)-y_m(k+1) \tag{10-76}$$

把式（10-74）和式（10-75）代入式（10-76），可以得到下式：

$$e_c(k+1)=f(y_p(k),y_p(k-1))+u(k)-0.5y_m(k)-0.3y_m(k-1)-r(k) \tag{10-77}$$

控制器设计的目的就是最小化式（10-77）中的误差值。把式（10-77）的右边设为0，求解控制输入，我们得到

$$u(k)=0.5y_m(k)+0.3y_m(k)+r(k)-f(y_p(k),y_p(k-1)) \tag{10-78}$$

然而，由于非线性函数$f(\cdot)$是未知的，必须使用神经网络近似值。按下式生成对系统的控制输入

$$u(k)=0.5y_m(k)+0.3y_m(k-1)+r(k)-N(y_p(k),y_p(k-1)) \tag{10-79}$$

公式（10-79）表示出了非线性控制器应用的映射。把式（10-79）代入式（10-74），控制系统在时刻样本$k$的输出由下式给出：

$$\begin{aligned}y_p(k+1)=&f(y_p(k),y_p(k+1))-N(y_p(k),y_p(k-1))\\&+0.5y_p(k)+0.3y_p(k-1)+r(k)\end{aligned} \tag{10-80}$$

从式（10-80）中我们看出，非线性系统和参考模型的输出之间的差别直接依赖于神经网络近似的精度。

间接控制器设计的第一步是系统辨识过程。假设系统是BIBO稳定，以离线方式进行系统辨识。从式（10-74）能得到下式：

$$f_y(y_p(k),y_p(k-1))=y_p(k+1)-u(k) \tag{10-81}$$

给系统输入然后记录它的输出，利用式（10-81）能够生成神经网络的训练输入模式。简单地说，对神经网络按下式表示的映射进行训练

$$[y_p(k),y_p(k-1)]^T\rightarrow y_p(k+1)-u(k) \tag{10-82}$$

在本例中我们训练一个RBF NN网络，它的每个隐藏层有48个神经元，扩展参数$\sigma=1$。输入信号为标准方差为2的正态分布随机数。一旦系统辨识正确，就可把式（10-79）作为非线性控制器。图10-15a显示了测试的输入信号图像：

$$y_t(t)=\frac{1}{2}[\sin(10\pi t)+\sin(25\pi t+0.5)] \tag{10-83}$$

图10-15b给出了在输入信号为式（10-83）时，参考模型和被控非线性系统的输出图像的

496 对比。从图看出，两者的差别是可以忽略的。

**例10.6 模型4的MRAC控制。**设由下面的差分方程给出的非线性系统

$$y_p(k+1)=\frac{1+y_p(k)+y_p(k-1)}{1+y_p^2(k)+2y_p^2(k-1)}\tan^{-1}(u^3(k)) \tag{10-84}$$

假定我们需要设计一个神经网络控制器，使被控系统的所有活动能用下面给出的参考模型方程来描述

$$y_m(k+1)=0.5y_m(k)+0.3y_m(k-1)+r(k) \tag{10-85}$$

神经网络控制器的结构由图10-16给出。

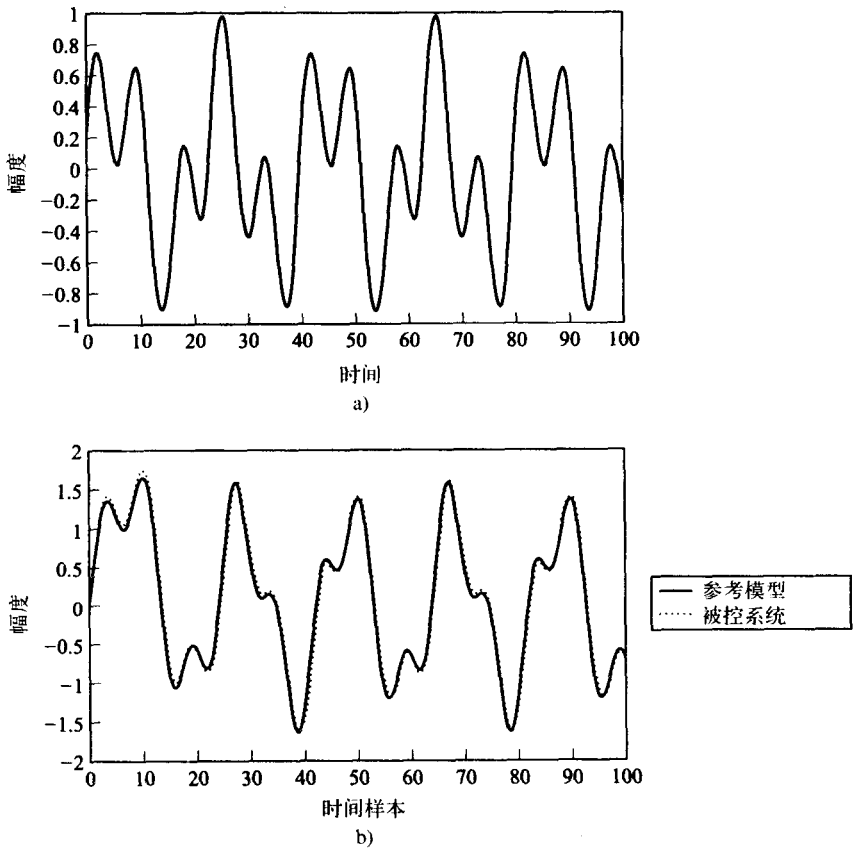



图10-15 a) 式（10-83）给出的测试信号；b) 对测试信号的响应

控制器设计的开始阶段需要进行系统辨识，也就是说，网络$NN_I$需要训练并模拟非线性系统的活动。与系统对应的NARX模型写为下式：

$$y_p(k+1)=f[u(k),y_p(k),y_p(k-1)]=f(\boldsymbol{x}) \tag{10-86}$$

其中$\boldsymbol{x}=[u(k),y_p(k),y_p(k-1)]^T$。利用第10.7.1节描述的方法，使用一个有两个隐藏层、每层30个神经元的MLP NN进行系统辨识。使用反向传播算法训练网络，以区间[-4, 4]中均匀分布的噪声作为输入序列。 497

只要系统被辨识，控制器设计就简化为训练一个神经网络$NN_C$，由$NN_C$所执行的NARX映射由下式给出：

$$u(k)=g[r(k),y_p(k),y_p(k-1)]\approx g[r(k),\hat{y}_p(k),\hat{y}_p(k-1)] \tag{10-87}$$

或

$$u(k) = \mathrm{NN_C}(y) \tag{10-88}$$

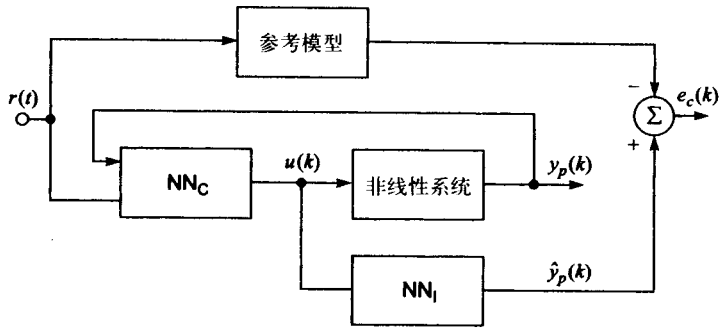



图10-16　例10.6中神经网络控制器的结构

其中$\boldsymbol{y} = [r(k), \hat{y}_p(k), \hat{y}_p(k-1)]^T$。在本例中，使用具有1000个中心的RBF NN作为$\mathrm{NN_C}$，其中$\boldsymbol{c}_i = [c_{i1}, c_{i2}, c_{i3}] \in \Re^{3\times 1}$，且$c_i \in [-1, 1]$，由$\mathrm{NN_C}$执行的映射函数为下式：

$$\mathrm{NN_C}(y) = \sum_{i=1}^{N_c} \lambda_i \phi(\|y - c_i\|) \tag{10-89}$$

其中

$$\phi(\upsilon) = \exp\left(-\frac{\upsilon^2}{\sigma^2}\right) \tag{10-90}$$

$\lambda_i$是网络的权值，扩展参数$\sigma = 0.2$。

被控系统和式（10-85）给出的参考模型间的误差由下式给出：

$$e_c(k+1) = y_p(k+1) - y_m(k+1) \approx \hat{y}_p(k+1) - y_m(k+1) \tag{10-91}$$

代价函数为下式：

$$J(k+1) = \frac{1}{2}e_c^2(k+1) = \frac{1}{2}[\hat{y}_p(k+1) - y_m(k+1)]^2 \tag{10-92}$$

498 网络$\mathrm{NN_C}$被训练以最小化式（10-92）中的函数。应用最速下降方法。权值按下式调整：

$$\lambda_i(k+1) = \lambda_i(k) - \mu \left.\frac{\partial J(k+1)}{\partial \lambda_i}\right|_{\lambda_i = \lambda_i(k)} \tag{10-93}$$

式（10-93）的梯度可由下式计算

$$\begin{aligned}\frac{\partial J(k+1)}{\partial \lambda_i} &= \frac{\partial}{\partial \lambda_i}\left\{\frac{1}{2}[\hat{y}_p(k+1) - y_m(k+1)]^2\right\} \\ &= e_c(k+1)\frac{\partial \hat{y}(k+1)}{\partial \lambda_i} \\ &= e_c(k+1)\frac{\partial \mathrm{NN_I}}{\partial u(k)}\frac{\partial u(k)}{\partial \lambda_i}\end{aligned} \tag{10-94}$$

运用式（10-88）和式（10-89），我们得到：

$$\frac{\partial u(k)}{\partial \lambda_i} = \phi\left(\|\boldsymbol{y} - \boldsymbol{c}_i\|\right) \tag{10-95}$$

网络输出对信号$u(k)$的导数由下式近似：

$$\frac{\partial \mathrm{NN}_1}{\partial u(k)} \approx \frac{\mathrm{NN}_1(u(k)+\varepsilon) - \mathrm{NN}_1(u(k))}{\varepsilon} \tag{10-96}$$

其中$\varepsilon$是比学习率参数小得多的数。

最后，应用式（10-94）、式（10-95）和式（10-96），我们能得到如下的学习规则，用来调整RBF NN控制器$\mathrm{NN}_c$的权值：

$$\lambda_i(k+1) = \lambda_i(k) - \mu e_c(k+1)\frac{\mathrm{NN}_1(u(k)+\varepsilon) - \mathrm{NN}_1(u(k))}{\varepsilon}\phi\left(\|\boldsymbol{y} - \boldsymbol{c}_i\|\right) \tag{10-97}$$

式（10-97）用来确定网络控制器的权值。图10-17给出了参考模型和被控非线性系统的输出之间的对比，测试信号是：

$$s(n) = \frac{1}{4}[\sin(10\pi n) + \sin(20\pi n)] \tag{10-98}$$

从图可以看出差别较小。

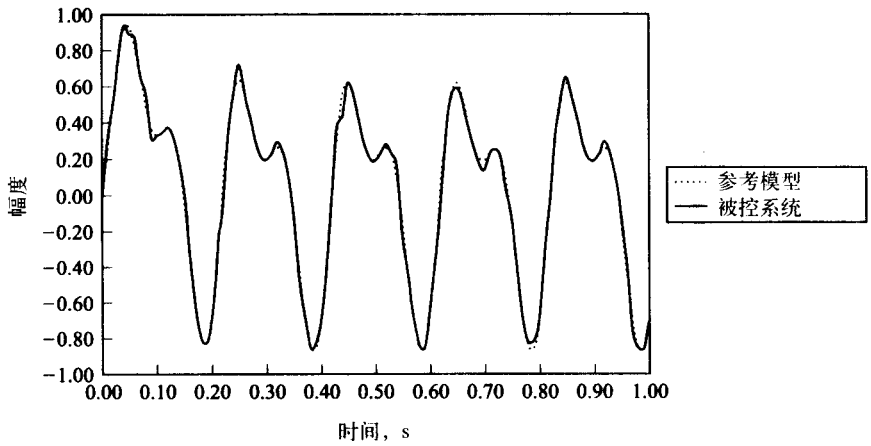



图10-17 在式（10-98）所示的测试信号下，被控系统和参考模型的输出之间的对照图 499

## 10.8 独立成分分析：未知源信号的盲分离

本节的主要目标是给出进行独立成分分析（independent-component analysis, ICA）的一种神经网络方法，然后介绍用于ICA的快速固定点算法（fast fixed-point algorithm, FFPA）。首先介绍ICA的基本思想，然后给出几个例子，说明利用ICA进行信号处理和图像处理的问题。

### 10.8.1 独立成分分析的概述

ICA可以看作是主成分分析（PCA）的扩展（详见9.2节），主要用于线性混合或未知源信号分离问题中的未知源信号分离[23-26]，也用于特征值的提取。从一组可观察的（可计算的）噪声信号中分离源信号不必一定要知道传输信道的特性。未知源信号分离技术可应用于阵列处理、医学信号处理、通信、语音处理、图像处理和许多其他领域。一般来说，存在两类未知源分离问题：瞬时混合和卷积（convolutive）混合。我们只关注瞬时混合的问题。

PCA和ICA的主要区别是与标准PCA相关的非关联属性的替代，在ICA中，数据向量的线性扩展系数必须相互独立，或者尽可能独立。这就是说，高阶统计[27, 28]必须用于ICA扩展的确定。在标准PCA中，二阶统计只提供去相关。在非高斯处理、非最小相问题、有色噪声或非线性处理中高阶统计很有用[27]。因此，不必惊奇，当ICA应用于神经网络时，在学习阶段必须使用非线性，即使最后的输入/输出是线性映射[29]。

ICA的基本概念

假定存在$q$个零均值、宽平稳源信号$s_1(k), s_2(k), \cdots, s_q(k)$，其中$k = 1, 2, \cdots$（离散时间的索引或图像的像素），它们都是标量，而且对于每一个样本值$k$都相互独立。独立的条件（见A.7.1节）可明确定义为：源信号的联合概率密度等于各信号的边缘概率密度的乘积，即：

500

$$\mathcal{A}[s_1(k), s_2(k), \cdots, s_q(k)] = \mathcal{A}[s_1(k)]\mathcal{A}[s_2(k)]\cdots\mathcal{A}[s_q(k)] = \prod_{i=1}^{q}\mathcal{A}[s_i(k)] \tag{10-99}$$

各个源信号假定都是未知的（不可观察的）。然而，我们能得到一组$h$个未知信号的噪声线性混合$x_1(k), x_2(k), \cdots, x_h(k)$。这些可计算的信号由下式给出：

$$x_j(k) = \sum_{i=1}^{q} s_i(k)a_{ij} + n_j(k) \tag{10-100}$$

其中$j = 1, 2, \cdots, h$，元素$a_{ij}$设为未知，$n_j(k)$是加性度量噪声。现在我们定义向量$\boldsymbol{x}(k) = [x_1(k), x_2(k), \cdots, x_h(k)]^T$，$\boldsymbol{x}(k)\in\Re^{h\times 1}$，$\boldsymbol{s}(k) = [s_1(k), s_2(k), \cdots, s_q(k)]^T$，$\boldsymbol{s}(k)\in\Re^{q\times 1}$（源向量包含有$q$个独立成分），$\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_q]$，$\boldsymbol{A}\in\Re^{h\times q}$（混合矩阵），其中$\boldsymbol{A}$的列向量是ICA扩展的基本向量。等式（10-100）可写为如下的向量矩阵形式：

$$\boldsymbol{x}(k) = \mathbf{A}\boldsymbol{s}(k) + \boldsymbol{n}(k) = \sum_{i=1}^{q} s_i(k)\boldsymbol{a}_i + \boldsymbol{n}(k) \tag{10-101}$$

这就是所提到的ICA扩展。假定混合矩阵$\boldsymbol{A}$中行数至少等于列数（$h \geqslant q$），且矩阵是列满秩，即：$\rho(\boldsymbol{A}) = q$（即，源信号的混合体都是不同的）。

与ICA相关的不明确点

ICA的使用中存在几个不明确的地方：

1. 已分离信号（独立成分）的幅度不能确定。因为$\boldsymbol{s}$和$\boldsymbol{A}$是未知的，见式（10-101）。当用一个标量去除$\boldsymbol{A}$中相应的列（如$\boldsymbol{a}_i$）时，源信号中的这个标量乘数都会不起作用。

2. 已分离信号的符号不明确，即独立成分乘以－1不影响模型。

3. 独立成分的阶数不能确定。这也是因为$\boldsymbol{s}$和$\boldsymbol{A}$是未知的。任一个独立成分可定义为“第一”个。为了说明这点，假定$\boldsymbol{P} > 0$（$\boldsymbol{P}\in\Re^{q\times q}$）是一个置换矩阵，那么ICA模型就可写为下式（无噪声的情况）：

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{P}^{-1}\boldsymbol{P}\boldsymbol{s} = (\boldsymbol{A}\boldsymbol{P}^{-1})(\boldsymbol{P}\boldsymbol{s}) = \tilde{\boldsymbol{A}}\tilde{\boldsymbol{s}} \tag{10-102}$$

其中$\tilde{\boldsymbol{A}}$是“新的”未知混合矩阵，向量$\tilde{\boldsymbol{s}}$包含重新排列的独立成分变量。

### 10.8.2　用神经网络进行独立成分分析

这一节讨论用ICA进行未知源分离适合于神经网络方法，它最初是由Karhunen et al.[29]

501 提出的。图10-18给出了用于源信号分离（或估计独立成分）和估计ICA扩展的基础向量[即式（10-101）的混合矩阵$\boldsymbol{A}$的列向量估计]的基本神经网络结构。

预漂白过程

分离过程之前的漂白过程（即预漂白）是一个很关键的步骤。这个过程把观察到的信号

的方差规格化为1。一般来说，使用经过预漂白的输入信号的分离算法有更好的稳定性，收敛也较快。但是，如果混合矩阵$\boldsymbol{A}$是病态的，或者一些源信号比另一些信号相对较弱，那么漂白数据也会使分离问题更复杂[30, 31]。应用下面的转换漂白输入向量$\boldsymbol{x}(k)$：

$$\upsilon(k) = \boldsymbol{V}\boldsymbol{x}(k) \tag{10-103}$$

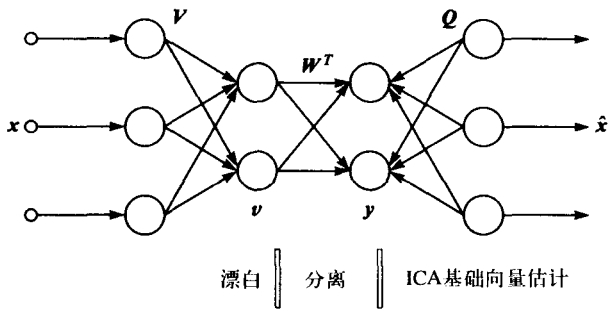



图10-18　ICA网络。网络的三层分别漂白、分离、估计基础向量。需要确定的权值矩阵是$\boldsymbol{V}$，$\boldsymbol{W}^T$和$\boldsymbol{Q}$

其中$\upsilon(k)$是第$k$个被漂白的向量，$\boldsymbol{V}$是漂白矩阵。漂白矩阵可由两种方法确定：批处理或神经元学习。对于批处理方法，如果使用PCA确定漂白矩阵，它由下式给出：

$$\boldsymbol{V} = \boldsymbol{D}^{-1/2}\boldsymbol{E}^T \tag{10-104}$$

其中$\boldsymbol{V}\in\Re^{q\times h}$，$\boldsymbol{D} = \mathrm{diag}[\lambda_1, \lambda_2, \cdots, \lambda_q]\in\Re^{q\times q}$，$\boldsymbol{E} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_q]\in\Re^{h\times q}$，$\lambda_i$是协方差矩阵$\boldsymbol{C}_x = \mathrm{E}\{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\}\in\Re^{h\times h}$的第$i$个大特征值，$\boldsymbol{c}_i$是伴随（主）特征向量，$i = 1, 2, \cdots, q$。因而，式（10-103）的转换中其实包含两个步骤，即压缩和漂白。压缩步骤中包括给$q$（源信号的数目）选择适当的值。因而，如果式（10-101）中的噪声项$\boldsymbol{n}(k)$假定为协方差矩阵$\mathrm{E}\{\boldsymbol{n}(k)\boldsymbol{n}^T(k)\} = \sigma^2\boldsymbol{I}_h$的零平均高斯白噪声，那么前面提到的可用于漂白的PCA也能用于选择（即估计）源信号$q$的数目（或独立成分数）可以被恢复。在噪声协方差矩阵中，$\sigma^2$是噪声向量$\boldsymbol{n}(k)$的各分量的联合方差。设定噪声向量与源信号$\boldsymbol{s}_i(k)$是没有关联的，$i = 1, 2, \cdots, q$。在这些设定下，数据向量$\boldsymbol{x}(k)$的协方差矩阵由下式给出： 502

$$\mathrm{E}\{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\} = \sum_{i=1}^{q}\mathrm{E}\{s_i^2(k)\}\boldsymbol{a}_i\boldsymbol{a}_i^T + \sigma^2\boldsymbol{I}_h \tag{10-105}$$

$q$是式（10-104）中协方差矩阵的最大特征值，即：$\lambda_1, \lambda_2, \cdots, \lambda_q$是源信号的平方值$\mathrm{E}\{s_i^2(k)\}$加上噪声的平方值$\sigma^2$的线性组合。因而，剩余$h-q$个特征值只表示相应的噪声（理论上说这些特征值等于$\sigma^2$）。如果信噪比（signal-to-noise）率足够大，那么这$q$个最大信号特征值将明显地大于剩余的噪声特征值。实际应用中，输入协方差矩阵的特征值决定于有效数据向量的协方差矩阵的时间平均，由下式给出：

$$\boldsymbol{C}_x \cong \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i(k)\boldsymbol{x}_i^T(k) \tag{10-106}$$

其中$N$是输入向量的总数。

学习漂白矩阵的随机近似算法由下式给出：

$$\boldsymbol{V}(k+1) = \boldsymbol{V}(k) - \mu(k)[\upsilon(k)\upsilon^T(k) - \boldsymbol{I}]\boldsymbol{V}(k) \tag{10-107}$$

上式中的学习因子参数要按照式（9-48）所述进行调整，即：

$$\mu(k)=\frac{1}{\dfrac{\gamma}{\mu(k-1)}+\|\upsilon(k)\|_2^2}\qquad 0<\gamma\leqslant 1.0 \tag{10-108}$$

其中$\gamma$是遗忘因子。当漂白（正交）转换$\boldsymbol{V}$应用于如式（10-103）的输入时，输出的漂白结果$\upsilon(k)$将得到漂白条件，即：

$$\mathrm{E}\{\upsilon(k)\upsilon^T(k)\}=\boldsymbol{I}_q \tag{10-109}$$

其中$\upsilon(k)$在式（10-103）中已定义。

*分离过程*

分离过程可采用多种不同的方法进行[26, 30, 32]。近似对比函数、分离矩阵最大化等方法已经发展起来[26]。但是，对比函数典型地需要利用估计的高阶统计数据进行大量的批处理计算，这导致自适应分离算法非常复杂。后面将会看到，利用数据的峭度（四阶积累）已经足够了。在第10.8.3节中给出神经元学习方法的另外一种方式，用在这里收敛将更快。另外一
503 类分离方法是利用神经网络进行源信号的分离[31]。在图10-18中，网络结构的第二阶段用于漂白信号$\upsilon$的分离。线性分离转换由下式给出：

$$\boldsymbol{y}(k)=\boldsymbol{W}^T\upsilon(k) \tag{10-110}$$

其中$\boldsymbol{W}\in\Re^{q\times q}$（$\boldsymbol{W}^T\boldsymbol{W}=\boldsymbol{I}_q$）是分离矩阵。因此，分离信号是第二阶段的输出，即$\hat{\boldsymbol{s}}(k)=\boldsymbol{y}(k)$。我们可以观察到一个有趣的事情，一旦估计出了源信号$\boldsymbol{s}(k)$，那么$\boldsymbol{A}$的伪逆$\boldsymbol{A}^+$一定也“暗暗地(blindly)”被确定[见式（10-101）]。

用来确定分离矩阵的一个非常直接的神经元学习方法基于下式的非线性PCA子空间学习规则[33-37]（详见9.3.6节）：

$$\boldsymbol{W}(k+1)=\boldsymbol{W}(k)+\mu(k)\{\upsilon(k)-\boldsymbol{W}(k)g[\boldsymbol{y}(k)]\}g[\boldsymbol{y}^T(k)] \tag{10-111}$$

其中$\upsilon(k)$是式（10-103）的预漂白输入向量，函数$g(\cdot)$是选定的合适的非线性函数，为了确保信号稳定地分离，它通常是奇函数。学习率参数$\mu(k)$要按照式（10-108）自适应方法调整，用$\boldsymbol{y}(k)$代替$\upsilon(k)$。为了好的收敛性，初始权值矩阵$\boldsymbol{W}(0)$最好用一组正交向量作为列向量。通常，非线性函数$g(\cdot)$选为下式：

$$g(t)=\beta\tanh(t/\beta) \tag{10-112}$$

其中$g(t)=\mathrm{d}f(t)/\mathrm{d}t$，logistic函数$f(t)=\beta^2\ln[\cosh(t/\beta)]$（详见9.7节）。对于式（10-111）的学习规则用于非线性时，这个函数的选取不是任意的，它是由需要确定ICA扩展高阶统计(Higher-order statistics)的事实所激发的。这也能从另一个未知信号分离的神经学习规则中观察到，这个学习规则称为*双梯度算法*[29, 36, 37]，由下式给出：

$$\boldsymbol{W}(k+1)=\boldsymbol{W}(k)+\mu(k)\upsilon(k)g[\boldsymbol{y}^T(k)]+\gamma(k)\boldsymbol{W}(k)[\boldsymbol{I}-\boldsymbol{W}^T(k)\boldsymbol{W}(k)] \tag{10-113}$$

其中$\gamma(k)$是另一个*增益参数*，一般约是0.5或1。这是一个随机梯度算法，在权值矩阵$\boldsymbol{W}$为正交的约束下，用来最大化或最小化性能指标如下：

$$J(\boldsymbol{W})=\sum_{i=1}^{q}\mathrm{E}\{f(y_i)\} \tag{10-114}$$

式（10-114）的正交约束条件在式（10-113）的学习规则中是以附加方式实现的。只要在式（10-114）中选取适宜的函数$f(\cdot)$，性能标准就可包含输出的四阶统计（四阶积累）之和，即*峭度*[27]。因此，标准或者以*负峭度*（峰谷值）最小化源信号，或者以*正峭度*（峰顶值）最大化源信号，具有负峭度的源信号通常称为*亚高斯信号*，具有正峭度的源信号常指*超高斯信号*。

在式（10-114）中期望算子将会被删掉，因为我们只考虑瞬时值。现在把logistic函数$f(t)=\ln[\cosh(t)]$ ($\beta=1$) 写成泰勒级数的展开形式： 504

$$f(t)=\ln[\cosh(t)]=\frac{t^2}{2}-\frac{t^4}{12}+\frac{t^6}{45}-\cdots \tag{10-115}$$

由于漂白，二阶项$t^2/2$是平均常数。非线性由$g(t)=\mathrm{d}f(t)/\mathrm{d}t=\tanh(t)=t-t^3/3+2t^5/15-\cdots$给出，如果数据是预漂白，那么三阶项将起主导作用（一个奇函数）。

*估计ICA的基础向量*

这是图10-18的最后一个阶段。本节给出两种方法估计ICA基础向量，或估计式（10-101）所示的混合矩阵$\boldsymbol{A}$的列向量。第一个方法是批处理方式，此时矩阵$\boldsymbol{A}$的估计$\hat{\boldsymbol{A}}$由下式给出：

$$\hat{\boldsymbol{A}}=\boldsymbol{E}\boldsymbol{D}^{1/2}\boldsymbol{W} \tag{10-116}$$

其中$\boldsymbol{D}$是式（10-104）所示的特征值矩阵，$\boldsymbol{E}$是式（10-104）所示的关联特征向量的列，$\boldsymbol{W}$是分离矩阵。第二个方法是用神经方式来估计ICA基础向量。从图10-18的最后阶段可给出观察数据的估计值如下：

$$\hat{\boldsymbol{x}}=\boldsymbol{Q}\boldsymbol{y} \tag{10-117}$$

当$\boldsymbol{n}=\boldsymbol{0}$（或$\boldsymbol{x}=\boldsymbol{A}\boldsymbol{s}$），把式（10-117）和式（10-101）进行比较，可以看出由于$\boldsymbol{y}=\hat{\boldsymbol{s}}$，所以$\boldsymbol{Q}=\hat{\boldsymbol{A}}$，因此，矩阵$\boldsymbol{Q}$的列向量就是ICA基础向量，即$\boldsymbol{A}$的列向量。神经学习算法可由下面的误差性能测量表达式转变而来：

$$J(\boldsymbol{Q})=\frac{1}{2}\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|_2^2=\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{Q}\boldsymbol{y}\|_2^2 \tag{10-118}$$

当采用最速下降方法$\boldsymbol{Q}(k+1)=\boldsymbol{Q}(k)-\mu\nabla_{\boldsymbol{Q}}J(\boldsymbol{Q})$时，用于估计ICA基础向量的神经学习规则变为下式：

$$\boldsymbol{Q}(k+1)=\boldsymbol{Q}(k)+\mu(k)[\boldsymbol{x}(k)-\boldsymbol{Q}(k)\boldsymbol{y}(k)]\boldsymbol{y}^T(k) \tag{10-119}$$

其中$\mu>0$是学习率参数，它在按式（10-108）的学习过程中，可用$\boldsymbol{Q}(k)\boldsymbol{y}(k)$替换$v(k)$进行调整。

**例10.7** 第一个例子分离三个正弦信号，它们的频率分别是$f_1=500$Hz、$f_2=600$Hz、$f_3=1000$Hz，采样频率是$f_s=10$kHz。初始信号如图10-19所示。使用的混合矩阵与Karhunen et al.[29]在第一个例子中用的一样，形如下式：

$$\boldsymbol{A}=\begin{bmatrix}0.0891 & 0.3906 & -0.3408\\ -0.8909 & -0.6509 & 0.8519\\ 0.4454 & 0.6509 & -0.3976\end{bmatrix} \tag{10-120}$$

因此，由$\boldsymbol{x}(k)=\boldsymbol{A}\boldsymbol{s}(k)$可得到三个“观察到的”信号，$k=1,2,\cdots,100$，如图10-20所示。这是一组三条曲线瞬间混合图像。首先，用式（10-103）给出的批处理漂白过程对观察信号进行预漂白。在本例中不必使用压缩，因此$h=q=3$。使用式（10-111）非线性PCA子空间学习规则进行分离，式（10-108）中的遗忘因子$\gamma=0.9$，$\beta=1$[用于式（10-112）表示的非线性]，为了收敛训练100回合。根据零均值、单位方差的高斯分布选择一组随机初始权值。然后，权值矩阵的列向量正交化。已分离信号如图10-21所示。因为我们已经知道正确的结果，所以能根据已知（实际）源信号计算每一个分离信号的相关系数，这些相关系数如图10-21所示。注意，分离信号与实际源信号的关联非常紧密，负相关系数表明ICA分离过程的输出有一个180°的相位平移。还要注意，输出信号的顺序不同于图10-19所示的初始信号。 505

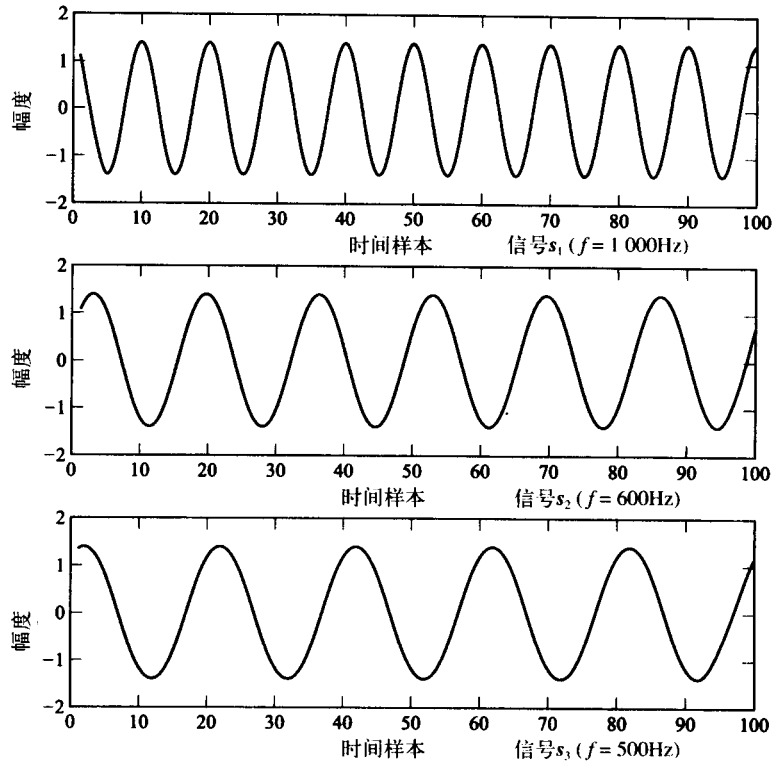



图10-19　三个初始正弦源信号

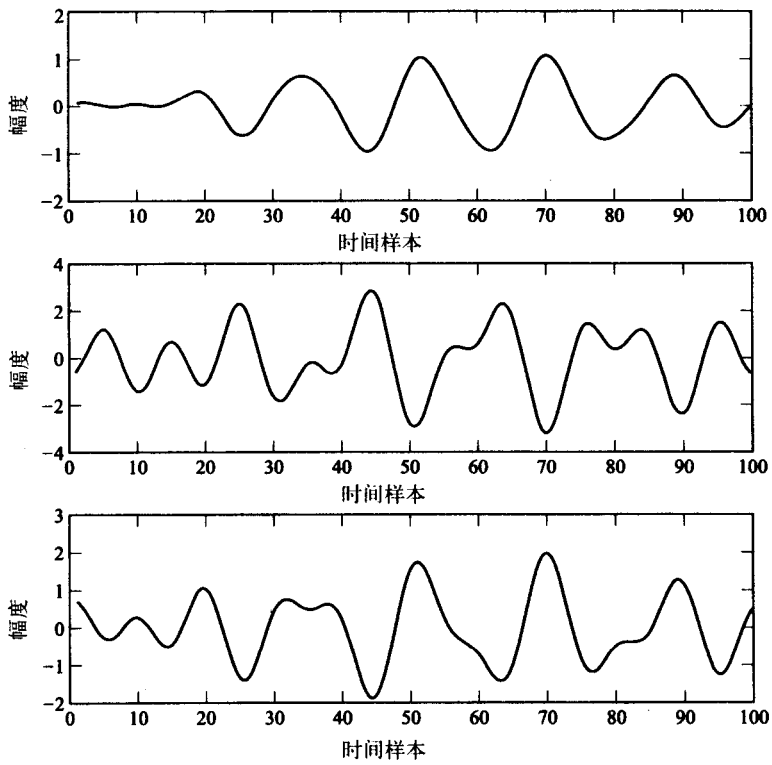



图10-20　“观察到的”混合信号

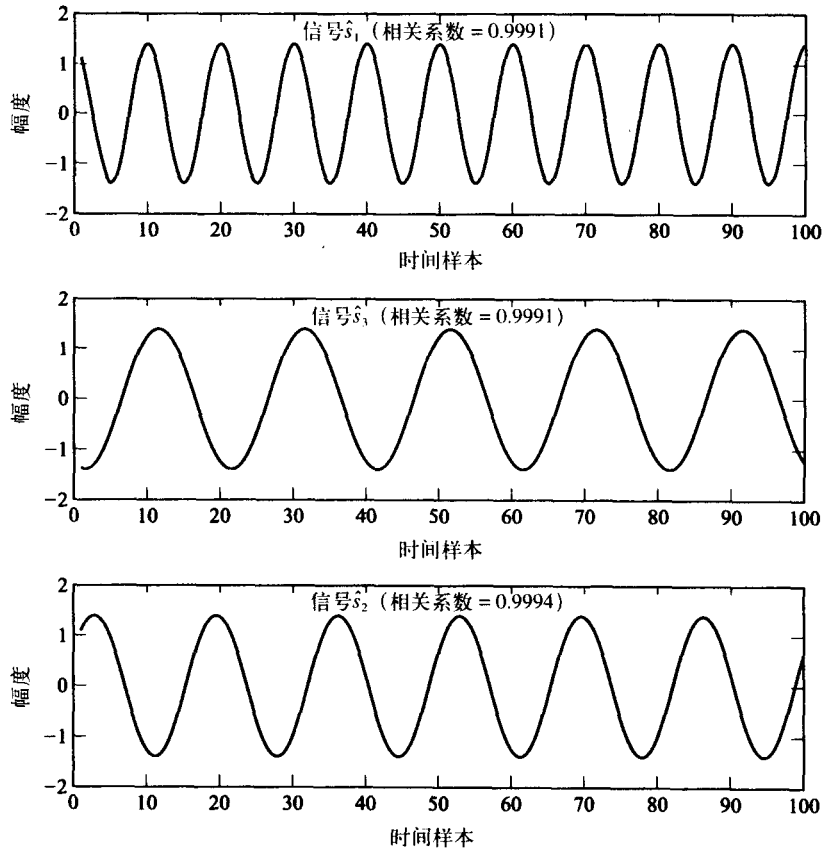



图10-21 使用非线性PCA子空间学习规则分离的正弦源信号

在这个问题中，我们还想估计ICA的基础向量[即式（10-120）给出的混合矩阵$A$的列向量]。利用式（10-116）的批处理方法，我们能得到矩阵$A$的估计值为：

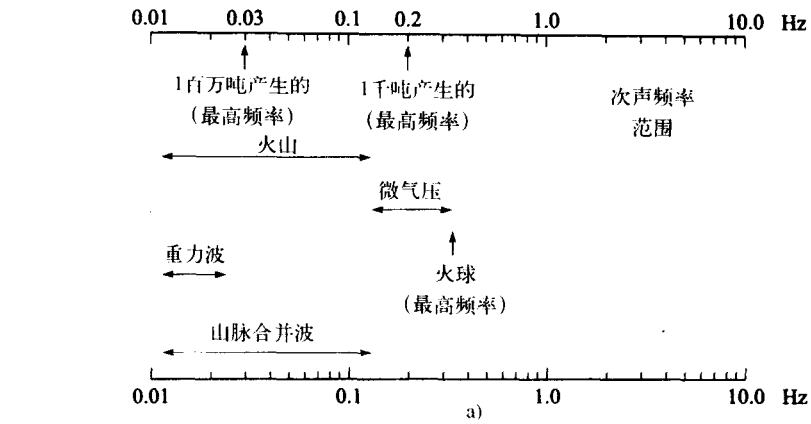
$$\hat{A}=\begin{bmatrix}0.1101 & 0.3478 & 0.3807\\ -0.9372 & -0.8376 & -0.6109\\ 0.4749 & 0.3985 & 0.6323\end{bmatrix} \tag{10-121}$$

把这些结果同式（10-120）中的实际混合矩阵相比较，我们能看出矩阵$A$的列向量估计值不准确，但是，它们之间比较相近。注意矩阵$\hat{A}$的列向量的顺序不同于初始混合矩阵。接下来使用神经学习方法估计ICA的基础向量。选择一组随机权值作为初始权值矩阵$Q$，式（10-108）中的自适应学习率参数中的遗忘因子$\gamma=0.9$，在训练5个回合后，网络收敛，混合矩阵的估计值由下式给出：

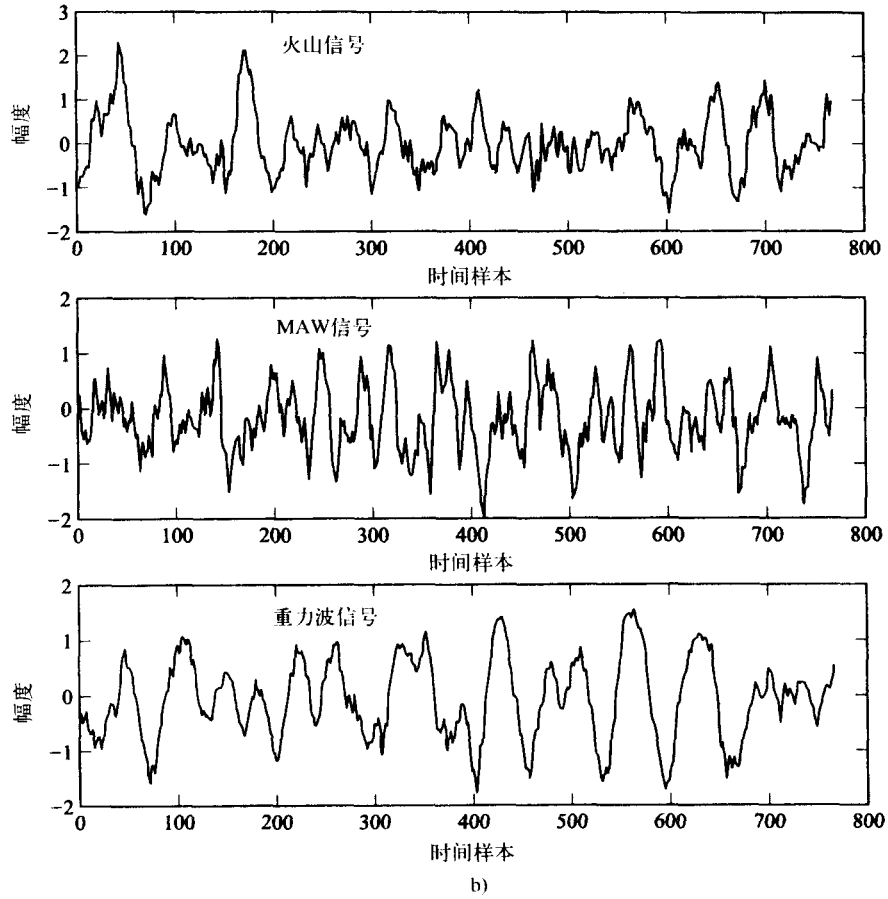
$$\hat{A}=\begin{bmatrix}0.1095 & 0.3461 & 0.3788\\ -0.9323 & -0.8333 & -0.6079\\ 0.4724 & 0.3964 & 0.6292\end{bmatrix} \tag{10-122}$$

同样地，式（10-122）中的混合矩阵估计值不准确，但它也与式（10-120）中所示的实际混合矩阵值相近。

**例10.8** 本例分离三个不同的瞬间混合次声信号[38-41]，见图10-22a。这些次声信号分别记录为三个独立的事件，如图10-22b所示。这三个事件分别是：(1) Galunggung Java火山爆




a)




b)

图10-22　a) 次声频率范围；b) 三种初始次声源信号

发次声信号[40]；(2) 新西兰发起的山组合（mountain-associated）波[41]；（3）内大气层重力次声波[38]，重力次声波是由大气中的温度倒置造成的。这些事件都是用一个大的四传感次声阵列（F阵列）从1981～1983年间在南极洲的Windless Bight记录的。图10-23给出了用于收集数据的F阵列的几何图。次声信号都是用标准采样频率1 Hz采样得到的。次声波是亚音频声波[38]，频率范围一般为$0.01 < f < 10$Hz，许多自然和人为现象都能产生这种波。见图10-22a。

次声传感的波段一般为0.01～10Hz。次声波来自于火山爆发、山组合波、极光波、地震、流星、雪崩、恶劣气候、矿山爆破、高速航空、重力波、微气压和核爆炸[42]。也发现大象（可能其他动物）用次声波进行沟通[43]。为确保遵守“全面核试验禁止条约（CTBT）”禁止核爆炸，国际监督机构（IMS）建立了一个收集全球次声的、地震的、水底传音的和放射性核的数据系统[42]。对于次声网络，现在专家建议频率从0.02～5.0Hz的范围用于传感。 506～509

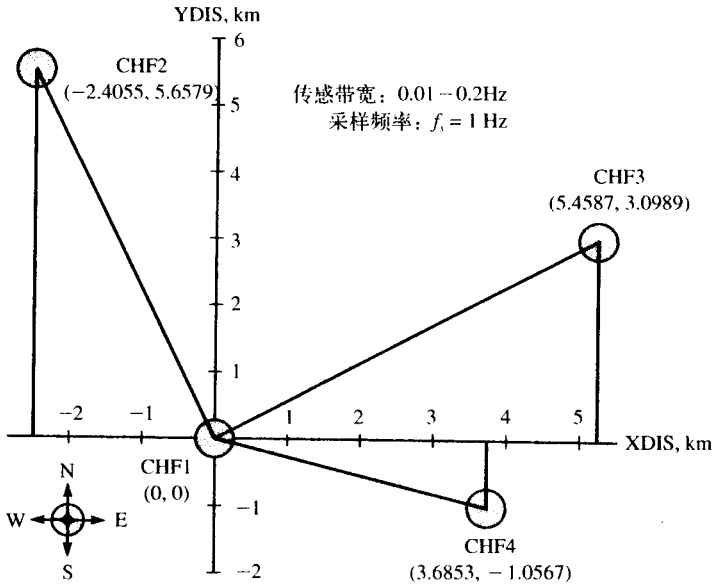



图10-23　南极洲Windless Bight次声传感F阵列

三种信号用下面的随机混合矩阵进行人为混合。

$$\boldsymbol{A}=\begin{bmatrix}0.3050 & 0.9708 & 0.4983\\ 0.8744 & 0.9901 & 0.2140\\ 0.0150 & 0.7889 & 0.6435\\ 0.7680 & 0.4387 & 0.3200\end{bmatrix} \tag{10-123}$$

式（10-123）中的数据随机选取自区间[0, 1]的均匀分布中。因此，由$\boldsymbol{x}(k)=\boldsymbol{A}\boldsymbol{s}(k)$可得到四个观察的混合信号，$k=1, 2, \cdots, 768$，如图10-24所示。这些信号的均值已被删除为零。由式（10-106）给出的观察数据协方差矩阵的特征值分别是$\lambda_1=2.1346$、$\lambda_2=0.1976$、$\lambda_3=0.0434$和$\lambda_4=-3.7772\times10^{-16}$。第四个特征值远比前三个小得多。因此，只有前三个最大的需要保留，从式（10-104）的漂白矩阵$\boldsymbol{V}\in\Re^{3\times4}$可知观察数据既需要漂白也需要压缩。因此$h=4$，$q=3$（重新得到源信号的数目）。和前面的例子一样，使用式（10-111）中的非线性PCA子空间学习规则进行分离。根据零均值、单位方差的高斯分布选择一组随机初始权值，然后，权值矩阵的列向量正交化。在式（10-108）中的遗忘因子$\gamma=0.9$，$\beta=1$[用于式（10-111）表示的非线性]，为了收敛训练250回合。最后被分离的信号如图10-25所示。因为我们已经知道源信号，所以能根据已知（实际）源信号计算每一个分离信号的相关系数，这些相关系数如图10-25所示。分离信号与实际源信号的关联非常紧密，负相关系数表明ICA分离过程的输出改变了符号。不同于前一个问题，分离信号的输出顺序与图10-22b所示的初始源信号相同。运用ICA对四信道的单个事件的阵列信号（即火山爆发[44]）进行了分离，在ICA过程中使用的次声信号是缠绕混合。结果非常有启发，表明在主火山次声信号中“隐藏”着微气压信号。 510

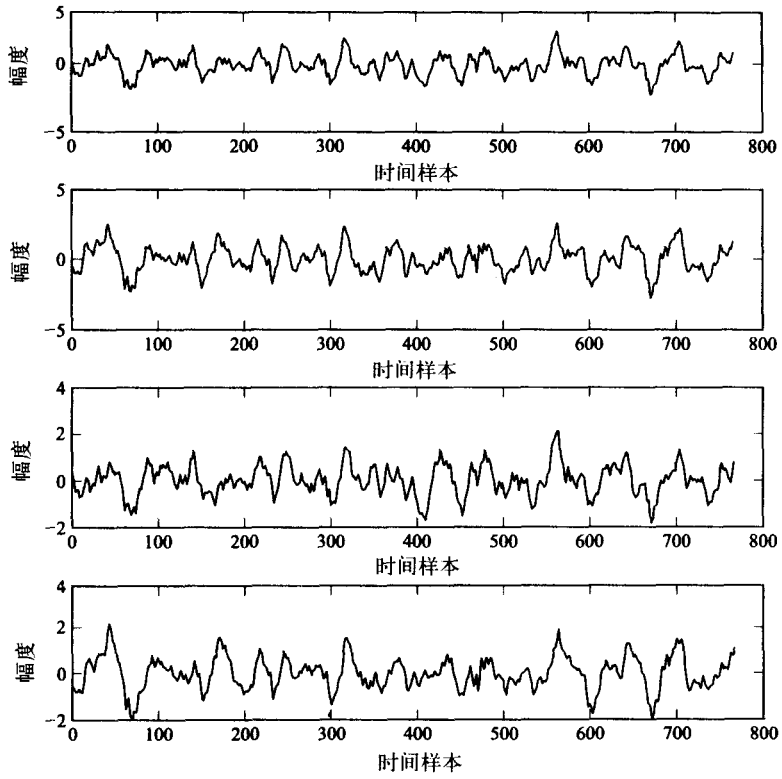



图10-24　“观察的”混合次声信号

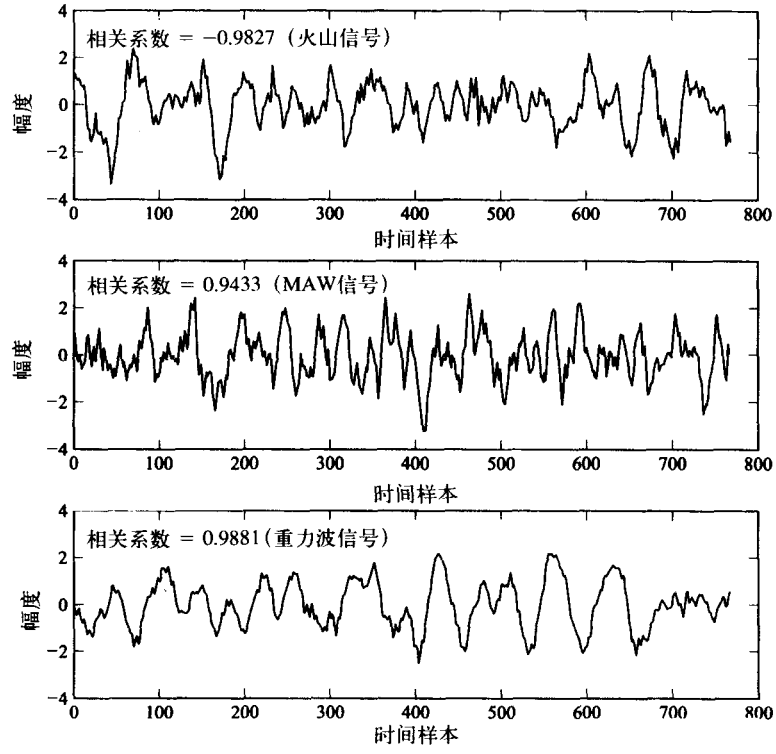



图10-25　应用非线性PCA子空间学习规则分离次声源信号

### 10.8.3 用于ICA的快速固定点算法

应用式（10-111）的非线性PCA子空间学习规则的ICA进行源信号盲分离至少有一个缺点，就是收敛相对较慢。而只要数据结构允许，应用快速固定点算法（FFPA）的ICA却能很快收敛到最精确解。FFPA不依赖任何用户指定的参数，它能处理所有的非高斯独立成分，而不必考虑它的概率分布（实际上有一个信号可以是高斯独立成分）。它的收敛速度是立方级的，与基于梯度的算法相比，FFPA的速度要快10～100倍[45]。

我们用稍有不同的方式来处理10.8.2节的问题。ICA的基础线性关系取为下式：

511
~
512

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s} \tag{10-124}$$

除不计噪声项和时间依赖外，它与式（10-101）给出的关系相同。假定$\boldsymbol{s}\in\Re^{q\times 1}$（有$q$个独立成分的向量）是零均值单位方差，并且元素都相互统计独立。当$h\geqslant q$时，观察的向量$\boldsymbol{x}\in\Re^{h\times 1}$有$h$个计算的变量，（混合矩阵）$\boldsymbol{A}\in\Re^{h\times q}$是满秩。正如10.8.2节所述，对$\boldsymbol{x}$中计算的数据预漂白通常能提高某些ICA算法的稳定性和收敛性。转换矩阵$\boldsymbol{V}$可用标准的PCA方法求得，这样观察到的数据就线性转换成一个向量：

$$\upsilon = \boldsymbol{V}\boldsymbol{x} \tag{10-125}$$

向量$\upsilon$的元素是互不关联的，而且有单位方差。因此，关联矩阵（或因$\upsilon$是零平均也称协方差矩阵）是单位矩阵或恒等矩阵，即$\mathrm{E}[\upsilon\upsilon^T] = \boldsymbol{I}$。如10.8.2节所讨论，在这个过程（即数据压缩）中向量$\upsilon$的维数也下降为$q$（独立成分的数目）。因此，PCA预漂白过程实际上有两个目标：规格化数据和确定独立成分的数目[⊖]。把式（10-124）代入式（10-125），能得到下式：

$$\upsilon = \boldsymbol{V}\boldsymbol{A}\boldsymbol{s} = \boldsymbol{B}\boldsymbol{s} \tag{10-126}$$

其中$\boldsymbol{B} = \boldsymbol{V}\boldsymbol{A}$是一个正交矩阵。通过前面的假定，由式（10-126）我们写出下式：

$$\underbrace{\mathrm{E}[\upsilon\upsilon^T]}_{I} = \mathrm{E}[\boldsymbol{B}\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{B}^T] = \boldsymbol{B}\underbrace{\mathrm{E}[\boldsymbol{s}\boldsymbol{s}^T]}_{I}\boldsymbol{B}^T = \boldsymbol{B}\boldsymbol{B}^T = \boldsymbol{I} \tag{10-127}$$

因而，问题就简化为确定正交矩阵$\boldsymbol{B}\in\Re^{q\times q}$，由式（10-126）可知，它能用来进行独立成分信号分离，即：

$$\hat{\boldsymbol{s}} = \boldsymbol{B}^T\upsilon \tag{10-128}$$

所以，混合矩阵$\boldsymbol{A}$的伪逆矩阵$\boldsymbol{A}^+ = \boldsymbol{B}^T\boldsymbol{V}$。

用FFPA进行ICA是基于一种高效固定点迭代法，求得观察变量线性组合的峭度的局部极值。一般，一个零均值随机变量$x$的峭度（或四阶积累）[27, 28]由下式给出：

$$\mathrm{kurt}(x) = \mathrm{E}[x^4] - 3(\mathrm{E}[x^2])^2 \tag{10-129}$$

对于两个独立随机变量$x_1$和$x_2$，等式$\mathrm{kurt}(x_1 + x_2) = \mathrm{kurt}(x_1) + \mathrm{kurt}(x_2)$成立。另外，对于零均值随机变量$x$和标量$\alpha$，等式$\mathrm{kurt}(\alpha x) = \alpha^4\mathrm{kurt}(x)$成立。漂白的观察变量的线性组合能写为$\boldsymbol{w}^T\upsilon$，而且这个线性组合可被搜索，因为它包含最大或最小峭度，其中权值向量$\boldsymbol{w}$是有界的，即$\|\boldsymbol{w}\|_2 = 1$。降阶FFPA方法基于这一思想，通过这种算法求得的每一个向量$\boldsymbol{w}_i (i = 1, 2, \cdots, q)$都是正交矩阵$\boldsymbol{B}$的列向量。从参考文献[45]，估计一个独立成分的算法如下：

513

---

⊖ 一个标准的PCA方法不总是计算独立分量数的最好方法。在一些情况下其他方法更适合，例如称为Akaike信息理论标准（AIC）的最大似然估计的扩展和编码理论的最小描述长度（MDL）[46]。

**用快速固定点算法进行ICA（一个分量）**

**步骤1**　漂白观察到的数据$\boldsymbol{x}$得到向量$\boldsymbol{v}$。

**步骤2**　随机设定初始权值向量$\boldsymbol{w}(0)$（注意省略了下标，因为开始就只关注于找一个独立成分），并把它规格化为单位长度，即：

$$\boldsymbol{w}(0) \leftarrow \frac{\boldsymbol{w}(0)}{\|\boldsymbol{w}(0)\|_2}$$

设定$j=1$。

**步骤3**　使$\boldsymbol{w}(j)=\mathrm{E}[\upsilon(\boldsymbol{w}^T(j-1)\upsilon)^3]-3\boldsymbol{w}(j-1)$，利用一个相对多个数目的向量$\upsilon$估计期望因子。

**步骤4**　规格化$\boldsymbol{w}(j)$为单位长度：

$$\boldsymbol{w}(j) \leftarrow \frac{\boldsymbol{w}(j)}{\|\boldsymbol{w}(j)\|_2}$$

**步骤5**[⊖]　如果$|\boldsymbol{w}^T(j)\boldsymbol{w}(j-1)|$不接近于1，那么令$j \rightarrow j+1$，然后返回步骤3；否则，输出向量$\boldsymbol{w}(j)$。

**步骤6**　利用$\boldsymbol{w}(j)$，由下式得到其中一个分离源信号：

$$s(k)=\boldsymbol{w}^T(j)\upsilon(k) \qquad k=1,2,\cdots$$

□

要估计$q$个独立成分，只需把上面的算法运行$q$次。但是，为了确保每次估计不同的独立成分，在上面给出的循环算法中要包含一个正交投影。基本思想就是：如果目前找到的$\boldsymbol{w}(j)$正好投影到与预先找到的矩阵$\boldsymbol{B}$的列向量正交的空间，那么就一个接一个地对独立成分进行估计。因此，我们定义矩阵$\tilde{\boldsymbol{B}}$的列向量就是预先计算的矩阵$\boldsymbol{B}$的列向量。投影操作加在上面步骤4的开头，即步骤4变为：

步骤4：令$\boldsymbol{w}(j) \leftarrow \boldsymbol{w}(j)-\tilde{\boldsymbol{B}}\tilde{\boldsymbol{B}}^T\boldsymbol{w}(j)$，然后

$$\boldsymbol{w}(j) \leftarrow \frac{\boldsymbol{w}(j)}{\|\boldsymbol{w}(j)\|_2}$$

514　在开始迭代前，初始随机向量也要按这种方式投影。

**例10.9**　应用上面介绍的降阶FFPA进行ICA，对混合数字图像进行分离。图10-26显示了初始图像。每个数字图像有$243\times351$ (= 85 293) 个像素，每6位像素有64级灰度。除了图10-26a是正峭度以外，所有图像的计算峭度都是负的。图10-26a是人工生成的嵌花式瓷砖图案，图10-26d是均匀分布的噪声图像，图10-26e是二元周期检验图案，其余的图像都是自然景象。这六幅图像用一个非正交满秩$6\times6$混合矩阵$\boldsymbol{A}$人为地混合在一起，这个矩阵$\boldsymbol{A}$在MATLAB中用区间[0, 1]上的随机均匀分布生成。特别地，如果我们定义图10-26a～f分别为数组$\boldsymbol{P}_1$, $\boldsymbol{P}_2$, …, $\boldsymbol{P}_6$，使用式（10-124）的ICA扩充法，执行下面的过程，就能得到一组“观察信号”：

生成每一个源信号：

$$\begin{aligned} \boldsymbol{s}_1&=\mathrm{vec}(\boldsymbol{P}_1) \quad & \boldsymbol{s}_2&=\mathrm{vec}(\boldsymbol{P}_2) \\ \boldsymbol{s}_3&=\mathrm{vec}(\boldsymbol{P}_3) \quad & \boldsymbol{s}_4&=\mathrm{vec}(\boldsymbol{P}_4) \\ \boldsymbol{s}_5&=\mathrm{vec}(\boldsymbol{P}_5) \quad & \boldsymbol{s}_6&=\mathrm{vec}(\boldsymbol{P}_6) \end{aligned} \tag{10-130}$$

组成源信号矩阵：

$$\boldsymbol{S}=[s_1, s_2, s_3\ s_4, s_5, s_6] \tag{10-131}$$

⊖　一个替代停止标准基于检查$\min\{\|\boldsymbol{w}(j)-\boldsymbol{w}(j-1)\|_2, \|\boldsymbol{w}(j)+\boldsymbol{w}(j-1)\|_2\}<\varepsilon$，其中$\varepsilon=10^{-4}$为一个合理的值。

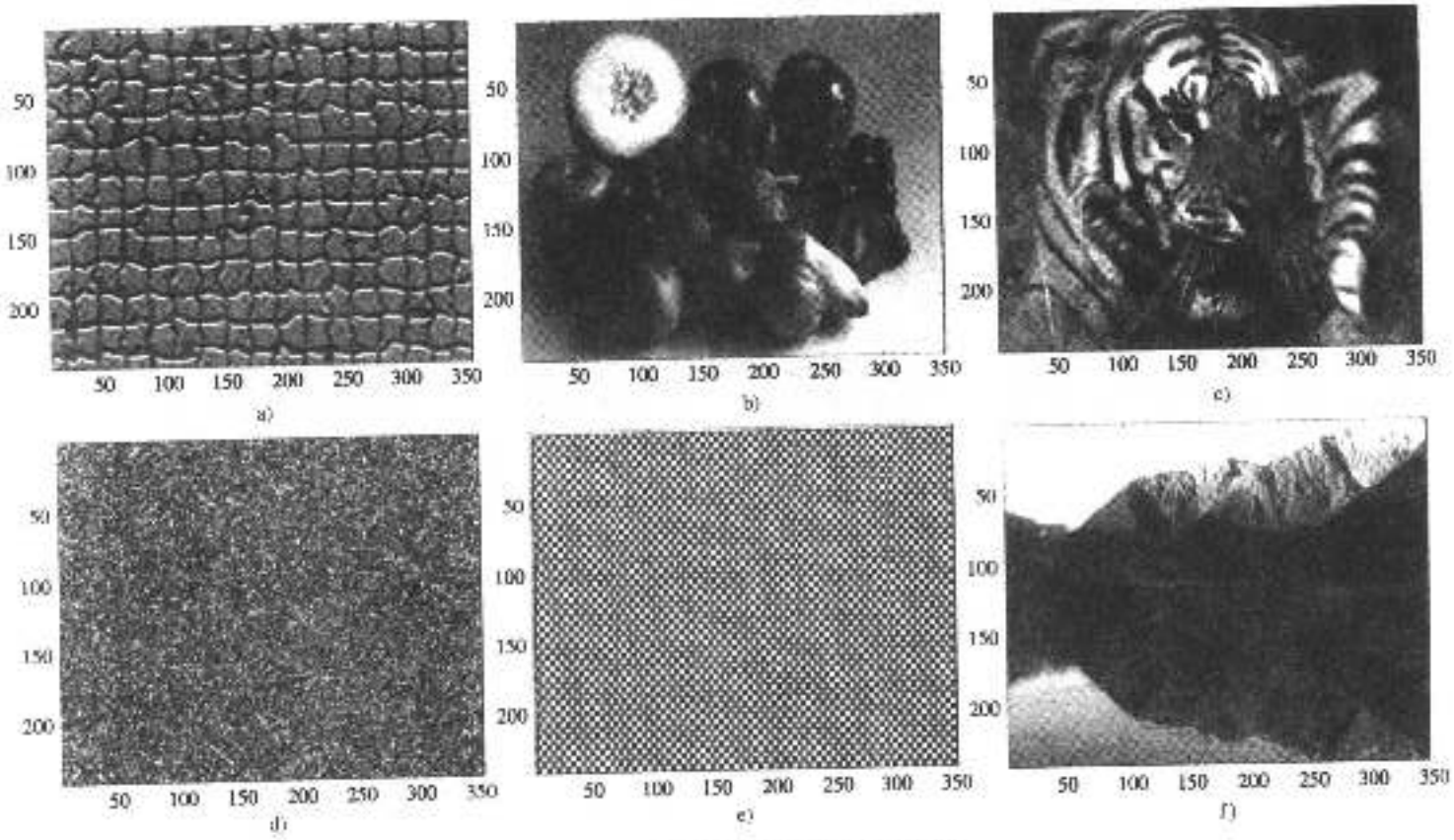



图10.26 例10.9中所用的六个初始源数字图像

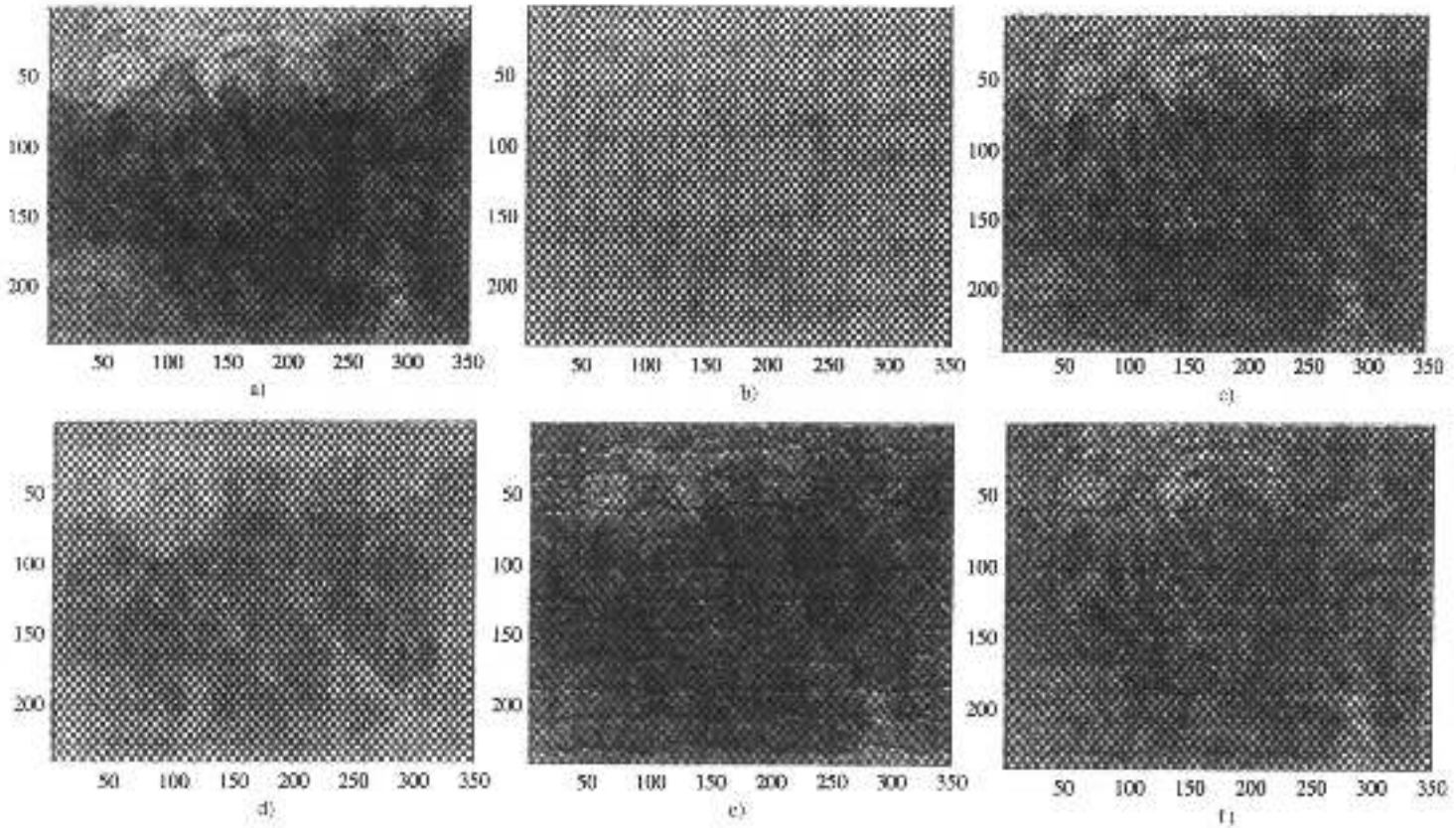



图10-27 混合图像，其中每幅图像的向量转置组成一个观察信号，即$x_i$。这些图像是图10-26中源图像的线性混合

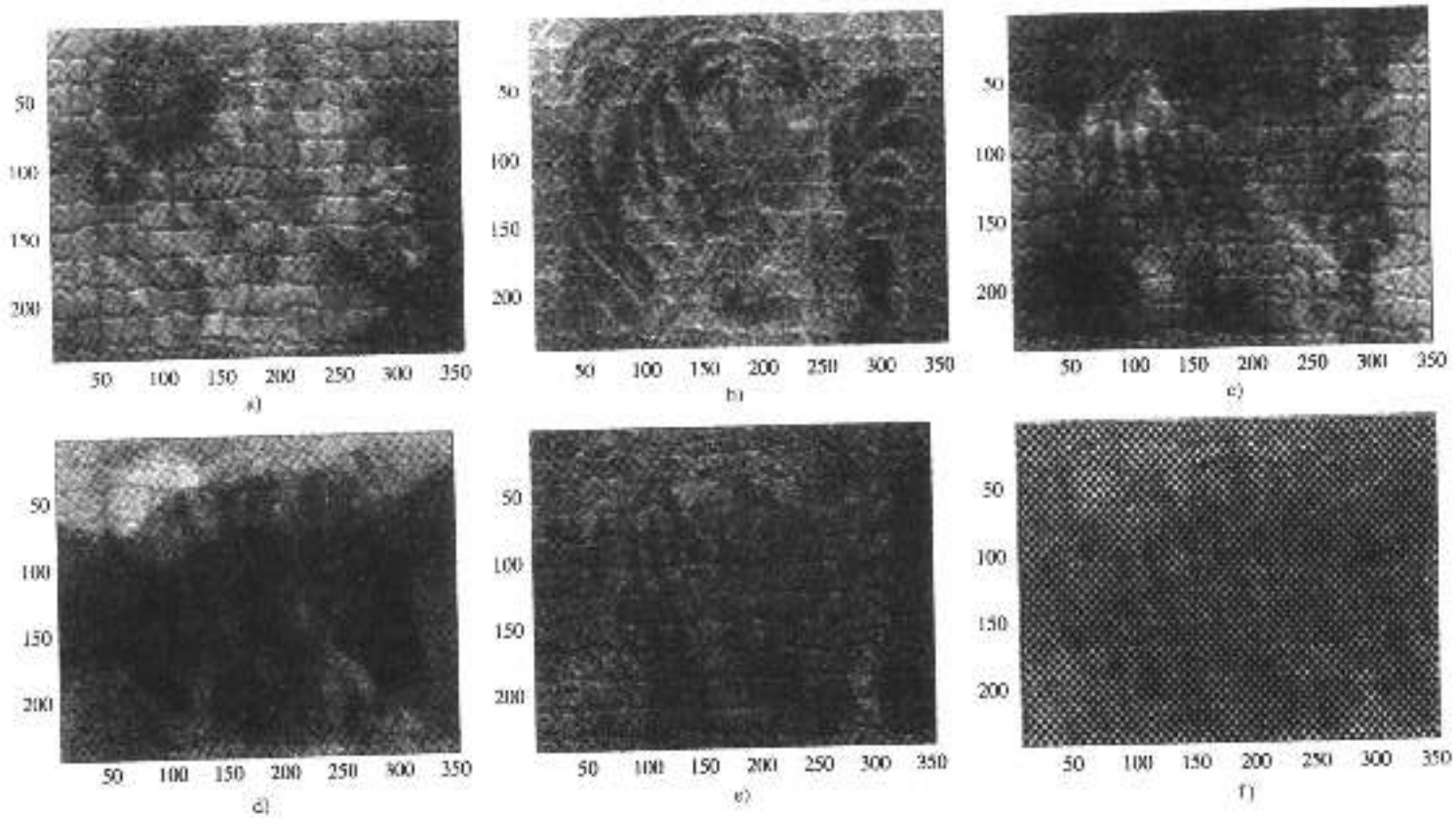



图10-28 PCA预漂白图像

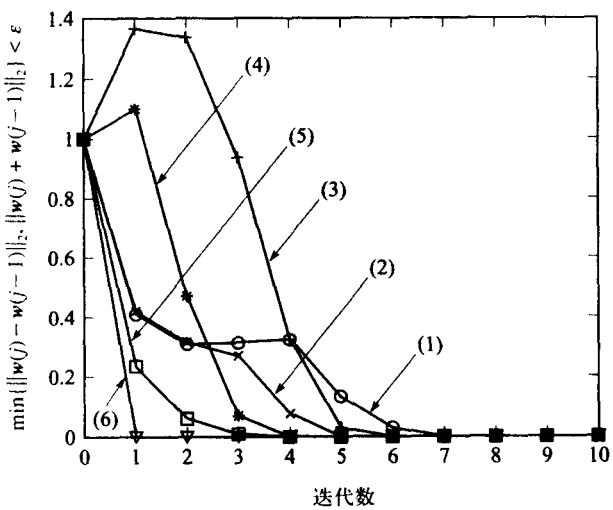
其中$S \in \Re^{85\,293 \times 6}$。

组成观察信号矩阵：

$$X = AS^T \tag{10-132}$$

其中$X \in \Re^{6 \times 85\,293}$。因此，矩阵$X$的行就是每幅混合图像的向量形式。现在取消对每个行的vec操作，则每幅混合图像都是由$X$的行有效抽取而来，假定合适的列长是243（像素）（这是图10-26中初始图像的行数）。图10-27给出了所有六个混合图像。它们的均值就是每幅图减去它的下一幅，结果就是经过预漂白的图像，图10-28显示了这些预漂白的图像。

一个能实现上面讨论的降阶FFPA的MATLAB m-file函数，它的停止参数设为$\varepsilon = 10^{-4}$（使用另一个停止标准）。利用六个预漂白的输入向量$v_1, v_2, \cdots, v_6$对六个独立成分图像进行抽取，能较快地达到收敛。对于每一个抽取的独立成分，图10-29给出了它们的收敛曲线。图10-30显示了抽取的独立成分图像。对于这些图像有几点需要讨论：第一，因为源信号的振幅信息不能保留，所以图10-30中每个抽取独立成分图像的输出灰度级都调整为64级灰度。第二，注意到输出图像的顺序与图10-26中的初始源图像不同。第三，注意到图10-30a、d、f分别是图10-26e、b、c的“负”图像。这是由于ICA的含糊性造成的（如10.8.1节所解释的）。图10-31显示了消除符号含糊的三幅图像的“反转”图。第四，注意到图10-30c、b不是图10-26f、b的准确复制，原因是，至少部分原因是图10-26中的原始图像的相互独立没有经过测试。总之，把图10-30中的三幅负图像反转之后，使用降阶FFPA分离最后得到的独立成分图像还是非常好的。图10-32给出实现降阶FFPA的MATLAB函数`ffpica.m`。在FFPA中对分离很必要的非线性是三次方非线性，在图10-32中可以看到（从结尾数第5行代码）。在这段程序中，一个独立成分的最大循环次数设为`max_iteration = 1000`。还要注意在图10-32中函数对数据进行了必要的预漂白。因为$q = h$，即独立成分数目等于观察信号数目，所以，没有对数据进行“压缩”。如果有必要估计独立成分的数目，那么必须修改图10-32中所示的降阶FFPA程序。




| IC号 | 符号 | 迭代数($j$) |
|---|---|---|
| (1) | ○ | 10 |
| (2) | × | 6 |
| (3) | + | 6 |
| (4) | * | 5 |
| (5) | □ | 7 |
| (6) | ▽ | 1 |

图10-29　用降阶FFPA抽取每个独立成分的收敛曲线

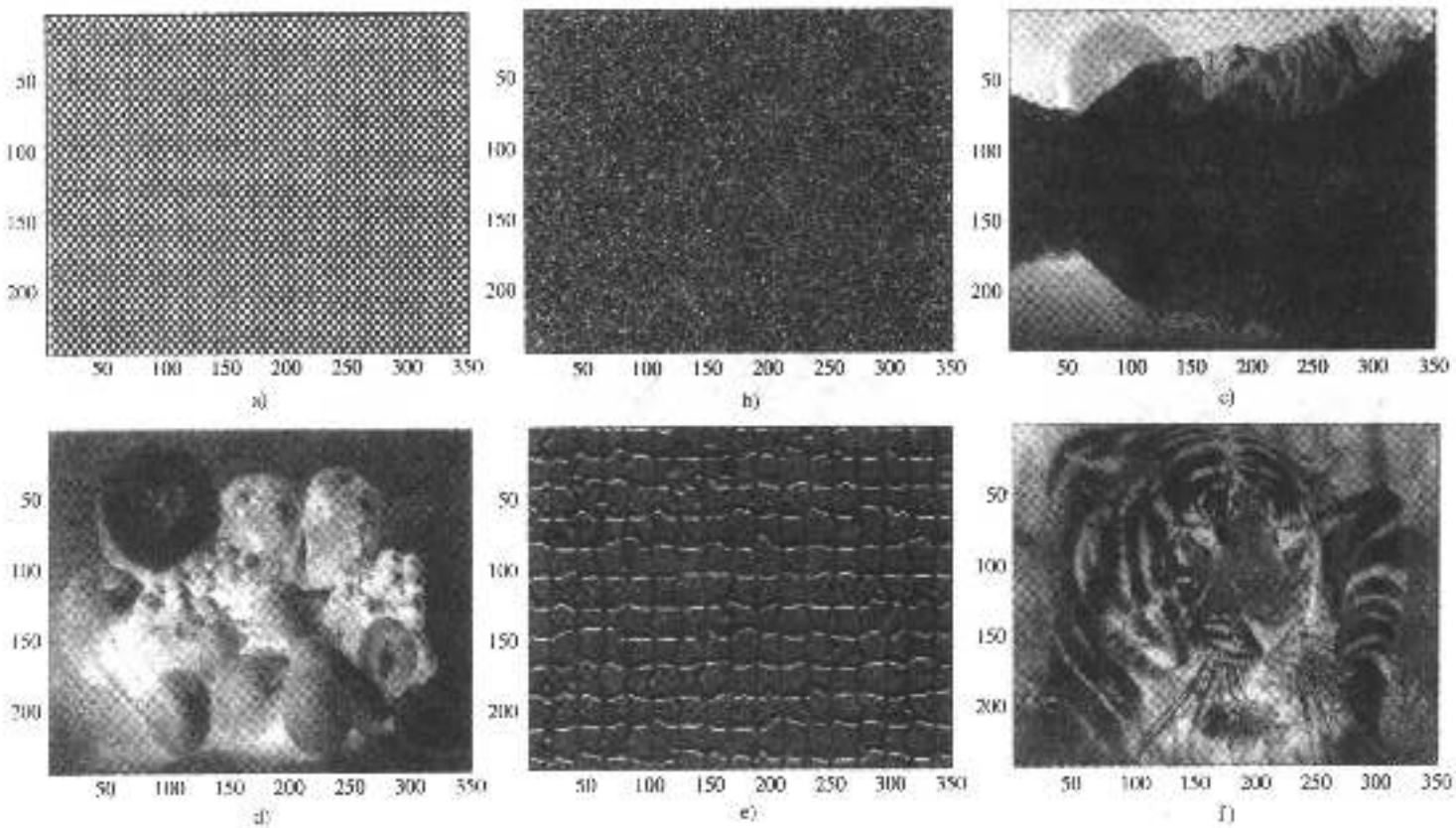



图10-30 用降阶FFPA抽取每个独立成分得到的图像

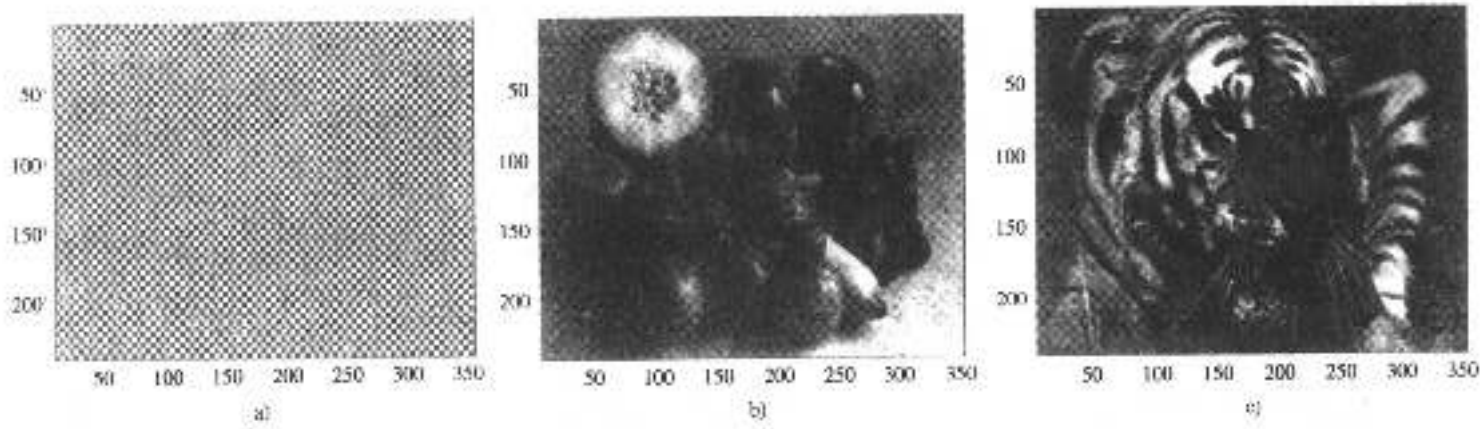



图10-31 图10-30a、d，经过反转后的分离图像。这是必要的过程，因为ICA的符号不明确

```
function [output] = ffpica(mixed_signal, max_iteration, epsilon)

% Fast Fixed-Point Independent Component Analysis
% (Deflation Method)
X = mixed_signal;
[num_IC, num_sample] = size(X);

% Remove the mean of row vectors
meanX = mean (X')';
X = X - meanX * ones (1,size (X, 2));

% PCA prewhitening of the data
covX = cov(X', 1); % covariance matrix of X
[E, D] = eig(covX);
whitening_matrix = inv (sqrt (D)) * E';
whitened_X = whitening matrix * X; % whitened signals

% Calculate the ICA using the fixed point algorithm
B = zeros (num_IC);
for i = 1:num_IC
   w = rand(num_IC, 1) - 0.5; % Initialize the weight vector
   w = w - B * B' * w;
   w = w / norm(w);
   w_old = zeros(size(w));
   for j = 1 : max_iteration
      w = w - B * B' * w;
      w = w / norm(w);
      if norm(w - w_old) < epsilon | norm(w + w_old) < epsilon

         B(:, i) = w;
         W(i,:) = w' * whitening_matrix;
         break;
       end
       w_old = w;
       u = whitened_X' * w;
       w = (whitened_X * (u .^ 3)) / num_sample - 3 * w;
       w = w / norm(w);
    end
end
output = W * mixed_signal;
```

图10-32 执行降阶FFPA的MATLAB函数ffpica.m

## 10.9 可加噪声中的正弦曲线的谱估计

对给定随机信号的功率谱估计是信号处理的一个基本问题。如果所记录信号的跨度相对较长，则运用著名的传统傅里叶分析可得到信号频谱的准确图像。换句话说，只要记录的信号足够长，利用离散傅里叶变换很容易得到信号的频率表示形式。但是，在许多实际应用中，需要估计时间跨度短的信号的频谱。在这种情况下，傅里叶分析就不再准确了，我们必须寻找另外的方法。如果信号不稳定、或需要对频谱快速估计、或者收集长的信号序列受到限制，那么很自然就提出了对时间跨度短的信号频谱估计问题。

在本节中，我们检验PLSR（见9.5节）或PLSNET（见9.6节）算法在特定频谱估计问题中的应用，在附加噪声下对正弦信号进行估计。虽然这个问题有一定的局限性，但它又常常需要进行分析，因为它在实践中很重要。应用表明：估计加性噪声下正弦频率与估计平面电磁波的来向（DOA）之间是等同的[47,48]。而DOA问题是雷达信号处理的基础，因此，被广泛研究[49,50]。虽然非常有趣，但对这两个问题完全分析超出了本书的范围，在本节，我们只限于讨论频谱估计的一个方法，并且用这个方法替代经典的傅里叶分析是可行的。

515 ~ 522

### 10.9.1 问题描述

考虑一个信号有$K$个正弦分量，受加性噪声干扰。设定信号是以采样频率$f_s$进行采样，整

个过程中样本总数为$N$。信号的第$i$个样本可表示为下式：

$$u(i)=\sum_{k=1}^{K}a_k\cos\left(2\pi\frac{f_k}{f_s}i+\theta_k\right)+\vartheta(i) \tag{10-133}$$

其中$a_k$、$f_k$和$\theta_k$分别是未知正弦分量的振幅、频率和相位，$\vartheta(i)$是加性噪声分量。谱估计就是要确定式（10-133）给出的复合正弦信号的未知参数。如果样本数较多，则各正弦分量相互正交，在这种情况下，由信号的傅里叶变换就能得到功率谱估计的准确值：

$$S(\omega)=|U(\omega)|^2=\left|\sum_{i=1}^{N}u(i)e^{-j\omega i}\right|^2 \tag{10-134}$$

$S(\omega)$是功率谱估计值，$\omega$是正规化的频率。但是，如果样本数较少，式（10-134）所提供的功率谱估计方法就不能得到足够分辨率，我们需要寻找另外的方法。还要注意式（10-134）是问题的一般方法，它没利用式（10-133）信号的特殊形式的优势。

虽然式（10-133）看起来还不太明显，但对未知正弦参数的估计可以简化为对其频率的估计。一旦频率知道了，它的振幅和相位就很容易确定。为了更明确，我们重写式（10-133）如下：

$$\begin{aligned}u(i)&=\sum_{k=1}^{K}a_k\frac{e^{j[2\pi(f_k/f_s)i+\theta_k]}+e^{-j[2\pi(f_k/f_s)i+\theta_k]}}{2}+\vartheta(i)\\&=\sum_{k=1}^{K}\left(\frac{1}{2}a_ke^{j\theta_k}\right)e^{j2\pi(f_k/f_s)i}+\left(\frac{1}{2}a_ke^{-j\theta_k}\right)e^{-j2\pi(f_k/f_s)i}+\vartheta(i)\end{aligned} \tag{10-135}$$

或

$$u(i)=\sum_{k=1}^{K}\left(A_ke^{j2\pi(f_k/f_s)i}+A_k^*e^{-j2\pi(f_k/f_s)i}\right)+\vartheta(i) \tag{10-136}$$

523 如果正弦信号的频率知道了，每个有效样本都可写为式（10-136）。它的向量矩阵形式如下：

$$\begin{bmatrix}u(1)\\u(2)\\\vdots\\u(N)\end{bmatrix}=\begin{bmatrix}e^{j2\pi(f_1/f_s)} & \cdots & e^{j2\pi(f_K/f_s)} & e^{-j2\pi(f_1/f_s)} & \cdots & e^{-j2\pi(f_K/f_s)}\\e^{j2\pi(f_1/f_s)2} & \cdots & e^{j2\pi(f_K/f_s)2} & e^{-j2\pi(f_1/f_s)2} & \cdots & e^{-j2\pi(f_K/f_s)2}\\\vdots & & & & & \vdots\\e^{j2\pi(f_1/f_s)N} & \cdots & e^{j2\pi(f_K/f_s)N} & e^{-j2\pi(f_1/f_s)N} & \cdots & e^{-j2\pi(f_K/f_s)N}\end{bmatrix}\begin{bmatrix}A_1\\\vdots\\A_K\\A_1^*\\\vdots\\A_K^*\end{bmatrix}+\begin{bmatrix}\vartheta(1)\\\vartheta(2)\\\vdots\\\vartheta(N)\end{bmatrix} \tag{10-137}$$

或者写为一个更简洁的形式如下：

$$\boldsymbol{U}=\boldsymbol{\Phi a}+\upsilon \tag{10-138}$$

其中$\boldsymbol{U}\in\Re^{N\times 1}$，$\boldsymbol{\Phi}\in C^{N\times 2K}$，$\boldsymbol{a}\in C^{2k\times 1}$，$\upsilon\in\Re^{N\times 1}$。假定$N\geqslant 2K$，系数集合$\{A_k\}$可以作为式（10-138）线性方程组的一个解。例如，式（10-138）中系统的最小二乘解可由下式给出（详见8.3节）：

$$\boldsymbol{a}=(\boldsymbol{\Phi}^H\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^H\boldsymbol{U} \tag{10-139}$$

式（10-139）的解假定矩阵$\boldsymbol{\Phi}$满秩，即：

$$\text{rank}(\boldsymbol{\Phi})=2K \tag{10-140}$$

为了说明这总是正确的，我们定义：

$$r_k = e^{j2\pi(f_k/f_s)} \tag{10-141}$$

和

$$r_{-k} = e^{-j2\pi(f_k/f_s)} \tag{10-142}$$

现在矩阵$\Phi$可重写为下式：

$$\Phi = \begin{bmatrix} r_1 & \cdots & r_K & r_{-1} & \cdots & r_{-K} \\ r_1^2 & \cdots & r_K^2 & r_{-1}^2 & \cdots & r_{-K}^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ r_1^N & \cdots & r_K^N & r_{-1}^N & \cdots & r_{-K}^N \end{bmatrix} \tag{10-143}$$

从式（10-143）我们看出，矩阵$\Phi$的形式是范德蒙德（Vandermonde）矩阵（详见A.2.19节），由于对任意的$i$和$j$来说，$r_i \neq r_j$，所以矩阵$\Phi$是满秩。

### 10.9.2　频率估计问题的PLSR解

假定正弦振幅和相位可按照式（10-139）估计，我们再看频率估计问题。为了得到这个问题的PLSR解，设定过滤器结构由图10-33给出。过滤器可以执行单步预测式（10-133）中的时间序列$\{u(i)\}$的函数。它的输出是第$i$个样本信号值和从它前面$M$个样本得到的预测值之间的差值，即：

524

$$e(i) = u(i) - \hat{u}(i) = u(i) - \sum_{m=1}^{M} u(i-m)\, w_m \tag{10-144}$$

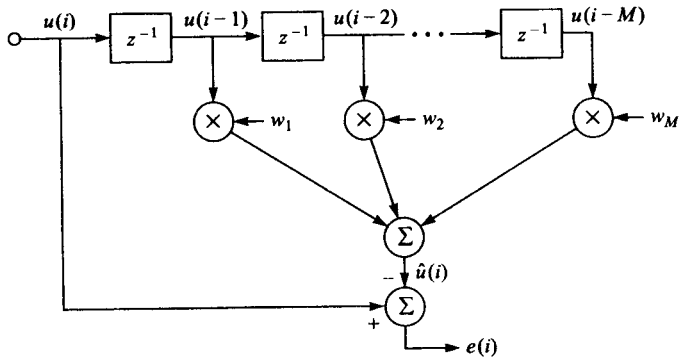



图10-33　对时间序列进行单步预测的横向过滤器

对式（10-144）进行$z$转换，我们得到下式：

$$E(z) = U(z)\left[1 - \sum_{m=1}^{M} w_m z^{-m}\right] = \frac{U(z)}{z^M}\left[z^M - \sum_{m=1}^{M} w_m z^{M-m}\right] \tag{10-145}$$

或

$$E(z) = U(z)H(z) \tag{10-146}$$

其中$H(z)$代表图10-33中预测过滤器总的转换函数。此时，假定对所有的$i = 1, 2, \cdots, N$，都有$\vartheta(i)=0$，

且假定我们能确定过滤器的系数，使预测值$\hat{u}(i)$与信号值$u(i)$之间的误差为0。这样得到下式：

$$U(z)H(z) \equiv 0 \tag{10-147}$$

或

$$U(e^{j\omega})H(e^{j\omega}) = 0, \quad \forall\omega \tag{10-148}$$

由于输入信号是正弦的叠加，因此频谱由离散分量组成，如图10-34所示。

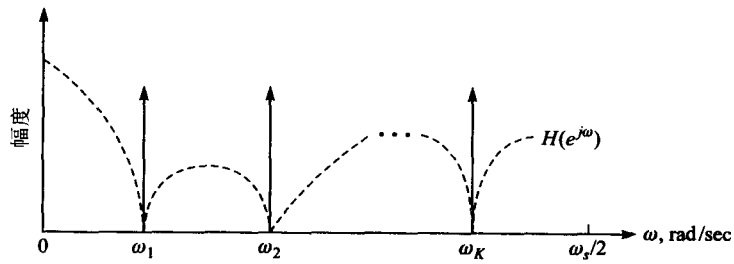



图10-34　单步预测过滤器的转换函数

为了使过滤器的输出为0对所有的$i$成立，过滤器转换函数的零值点必须准确地定位于正弦
525 信号的频率上。因此，估计频率的问题简化为寻找过滤器转换函数单步预测的零值点。讨论到现在，我们一直假定系统无噪声。但在实际应用中，出现的附加噪声还是对零点的位置多少有些影响。然而，除非噪声功率谱有跳动，我们希望它的影响尽量小。

式（10-147）表明为了估计正弦频率，需要寻找预测过滤器的选择权值。$i = M+1, M+2, \cdots, N$时，重写式（10-144）得到：

$$u(M+1) = \sum_{m=1}^{M} u(M+1-m)w_m + e(M+1) \tag{10-149}$$

$$u(M+2) = \sum_{m=1}^{M} u(M+2-m)w_m + e(M+2) \tag{10-150}$$

$$\vdots$$

$$u(N) = \sum_{m=1}^{M} u(N-m)w_m + e(N) \tag{10-151}$$

或者写为向量矩阵形式如下：

$$\boldsymbol{c} = \boldsymbol{A}\boldsymbol{w} + \boldsymbol{e} \tag{10-152}$$

其中：

$$\boldsymbol{c} = [u(M+1), u(M+2), \cdots, u(N)]^T \tag{10-153}$$

$$\boldsymbol{A} = \begin{bmatrix} u(M) & u(M-1) & \cdots & u(1) \\ u(M+1) & u(M) & \cdots & u(2) \\ \cdots & \cdots & \cdots & \cdots \\ u(N-1) & u(N-2) & \cdots & u(N-M) \end{bmatrix} \tag{10-154}$$

$$\boldsymbol{w} = [w_1, w_2, \cdots, w_M]^T \tag{10-155}$$

和
$$\boldsymbol{e}=[e(M+1),e(M+2),\cdots,e(N)]^T \tag{10-156}$$
估计式（10-152）中的选择权值和在第9章中的处理是相同的（详见9.5节和9.6节）。使用PLSR方法来处理选择权值问题如下：

526

$$\boldsymbol{w}=\hat{\boldsymbol{W}}^T\left(\hat{\boldsymbol{B}}\hat{\boldsymbol{W}}^T\right)^{-1}\hat{\upsilon} \tag{10-157}$$
其中$\hat{\boldsymbol{W}}$、$\hat{\boldsymbol{B}}$和$\hat{\upsilon}$是PLSR模型中相应的矩阵。⊖

一旦找到预测过滤器的选择权值，也就找到了过滤器转换函数零值点处的正弦信号的频率。但是，如前所述，由于附加噪声的出现会影响零值点的位置，所以习惯上定义准频谱函数如下：
$$S_c(\omega)=\frac{1}{\left|H(e^{j\omega})\right|^2}=\frac{1}{\left|1-\sum_{i=1}^{M}w_i e^{j\omega}\right|^2} \tag{10-158}$$
正弦频率可以根据$\omega$的值定位，此时式（10-149）中的函数存在峰值。

**例10.10**　假定输入信号由如下的三个正弦分量叠加：
$$u(i)=\cos[2\pi(0.3)i]+\cos\left[2\pi(0.35)i+\frac{\pi}{4}\right]+\cos\left[2\pi(0.1)i+\frac{\pi}{3}\right]+\vartheta(i) \tag{10-159}$$
其中$\vartheta(i)$是附加噪声分量。假定噪声是零均值高斯噪声，标准偏差是0.4，可以得到近似的信噪比（signal-to-noise ratio, SNR）如下：
$$\frac{S}{N}=10\log\left(\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{0.4^2}\right)=9.7\text{dB} \tag{10-160}$$
在信号处理中，设定有30个样本信号。单步预测过滤器的结构如图10-33所示，有10个抽头-延迟（tap-delay）元素。图10-35a显示了在20条曲线上运用PLSR谱估计算法实验得到的转换函数的零点位置，用10个PLSR因子估计过滤器的权值。过滤器转换函数的零点可以分成两组。第一组中的元素是靠近单位圆的零点。这些零点与式（10-159）所给出的正弦分量的频率相对应，由于存在三个正弦分量，图10-35a中的图像表明每半个单位圆有三个零点聚集点。第二组由单位圆内的随机位置零点组成。这些零点的位置由附加噪声的实际样本所决定，因而，因实验不同而不同。其中一条实验曲线的准谱图像由图10-35b所示。我们看到准谱函数的峰值点在各个正弦分量的频率处出现。

PLSR模型的维

PLSR模型的一个重要部分就是决定因子数目，直观地说，随着PLSR因子的增加，我们期望单步预测过滤器更准确。但是，预测精度的提高可能造成数据过适应。在信号谱估计中数据过适应会造成伪峰值。图10-36显示了在不同PLSR因子数下，例10.10的问题估计转换函数的零点聚集图像。当PLSR因子的数目比单独正弦分量的数目少时，估计算法不能很好地进行谱估计。例如，在一个PLSR因子的情况下，在单位圆附近只有一个零点聚集位置，在PLSR因子数为2时，也是相同的情形。在PLSR因子数为3时，注意到形成3个聚集与3个正弦分量的频率相对应。更多地增加PLSR因子会增加算法的求解方案，但这也造成数据过适应，当我们用10个PLSR因子时，从图10-36d可以清楚地看到这一情况。在这种情况下，注意到有一些不在三个聚集中的零点在单位圆附近、甚至就在单位圆上。在准谱估计中这会造成伪谱

527

⊖ 注意，PLSNET（参见9.6节）也可用于计算$\{\hat{\boldsymbol{W}},\hat{\boldsymbol{B}},\hat{\upsilon}\}$。

峰值，会看作是信号频谱的附加分量。

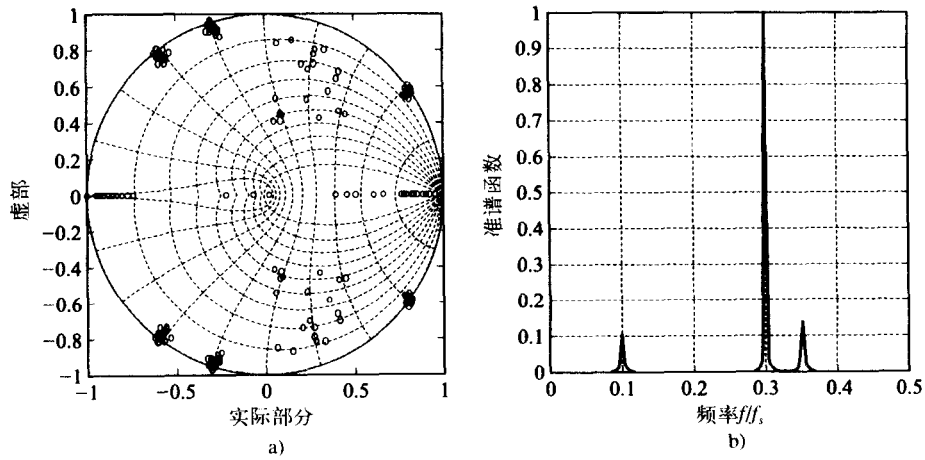



图10-35　应用PLSR进行谱估计。a) 例10.10中单步预测过滤器的转换函数的零点位置。图上显示了20个独立实验得到的图像；b) 1个实验中得到的规范化准谱图像。规范化使图像的最大值是1

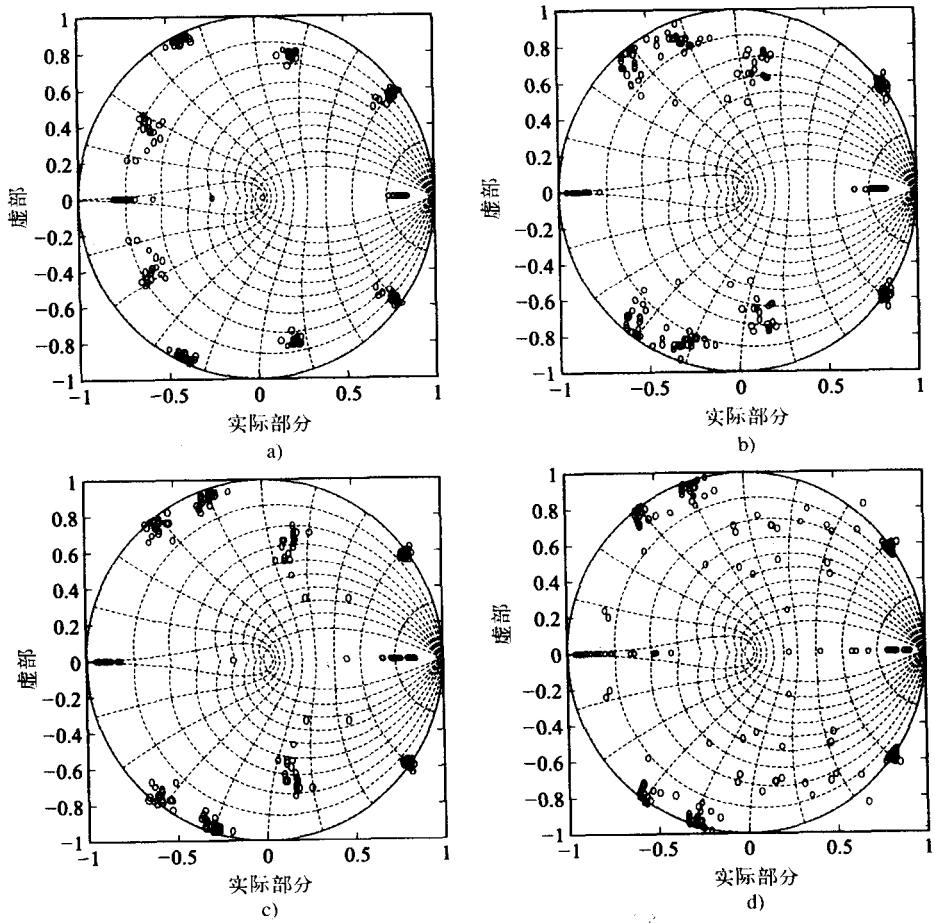



图10-36　单步预测器转换函数的零点位置数是PLS因子的数目的函数。a) 1个PLS因子；b) 2个PLS因子；c) 3个PLS因子；d) 10个PLS因子

从9.5节关于PLSR的讨论中，我们知道当所有的回归因子都使用时，PLS就与CLS估计等价。图10-36中的实验结果表明，同对应的CLS相比，减少因子的数目能增加估计的鲁棒性。在许多谱估计算法中这是一项常用的技术，通常指谱估计的特征分析算法。关于这些算法的详细介绍和更多的谱估计技术可参考文献[47-50]。

## 10.10 其他案例分析

本节重点介绍两个关于神经网络应用的实例。这些例子说明运用神经计算方法可以处理两类重要问题。

### 10.10.1 从近红外谱模拟数据估计葡萄糖浓度

528 ~ 529

本例运用人工近红外（NIR）数据模仿简单水矩阵中不同的葡萄糖浓度[51, 52]，使用的数据与9.6节中例9.4数据基本一样（见图9-20）。区别仅是本例中可加噪声与以前所使用的不同（即用不同的蒙特卡罗游程生成数据）。数据是怎样生成的和数据表示什么的详细介绍见例9.4。图10-37和图10-38给出了用来生成数据的MATLAB m-file。

```
%
% Generates Synthetic Near-Infrared (NIR) Data
%

% Spectrum of the component of interest (could be NIR spectrum of
  glucose)
      Spi = .6*gaussd(30,100,15) + .3*gaussd(50,100,70);
% GAUSSD generates a Gaussian distribution
% Spectrum of obscuring component (NIR spectrum of water)
      Spo = .8*gaussd(10,100,20) + .6*gaussd(20,100,80);
      A = zeros(200,100);
      for i = 1:200
          A(i,:) = i*Spi;
      end
% Concentrations (could be glucose concentrations)
      p = ones(100,1);
      C1 = 1000*atan(.0001*A*p);
% Addition of the zero-mean Gaussian noise
      An = A + randn(200,100);
% Addition of the obscuring component
      An1 = zeros(200,100);
      for i =1:200
          An1(i:) = An(i,:) + (1000+30*randn)*Spo;
      end
% Form the Training and Test Data
% Training Spectra (each row is a NIR spectrum)
      TRAIN = An1 (1:2:200,:);
% Training Concentrations (target values)
      TRAINC = C1(1:2:200,:);
% Test Spectra (each row is a NIR spectrum)
      TEST = An1(2:2:200,:);
% Test Concentrations (target values)
      TESTC = C1(2:2:200,:);
clear p i A An An1 C1 Spi Spo
```

图10-37 生成人工NIR数据的MATLAB m-file。gaussd函数如图10-38所示

本例的目的是比较下面三种方法的性能：(1) CLSR方法（详见9.4节）；(2) 用反向传播（BP）训练的标准MLP NN（详见3.3.1节）；(3) PLS神经网络（详见9.6节）。下面详细介绍每种方法。

CLSR

根据式（9-98）生成CLS校准模型$\hat{\boldsymbol{b}}_{fCLS}$，即：

530

$$\hat{\boldsymbol{b}}_{f\,\mathrm{CLS}} = (\boldsymbol{A}_{\mathrm{train}}^{T}\boldsymbol{A}_{\mathrm{train}})^{-1}\boldsymbol{A}_{\mathrm{train}}^{T}\boldsymbol{c}_{\mathrm{train}} \tag{10-161}$$

```
function out=gaussd(d,np,c)

% out=gaussed(d,np,c)
% generates a Gaussian distribution
% out = output distribution
% d   = distance from center
% np  = number of points out contains
% c   = peak center

d = d/2;
for k = 1:np
   out(k) = exp(-((k-c)/d)^2);
end
```

图10-38　生成高斯曲线的MATLAB函数

图10-39显示了用于建立CLS校准模型的100个训练光谱中的25个。运用这个模型和测试数据，另外100个光谱（看起来与图10-39所示数据相同），即预测（估计）的葡萄糖浓度由下式给出：

$$\hat{\boldsymbol{c}}_{\mathrm{test\,CLS}} = \boldsymbol{A}_{\mathrm{test}}\hat{\boldsymbol{b}}_{f\,\mathrm{CLS}} \tag{10-162}$$

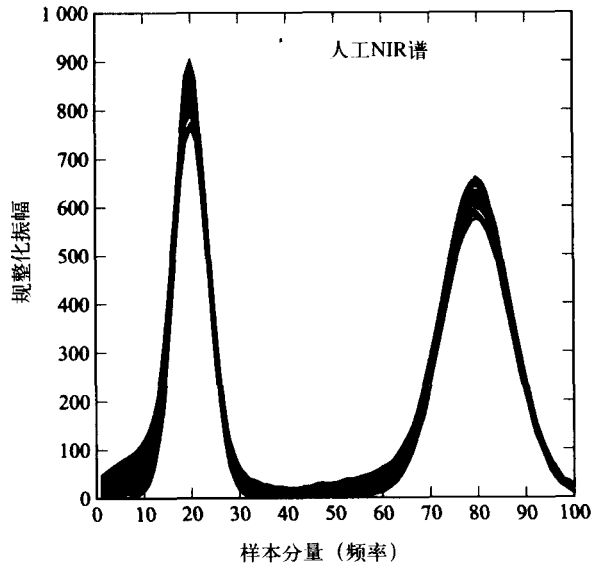



图10-39　用于三种方法的训练数据。这是全部训练数据其中的25个谱

531 式（10-161）中校准模型的性能通过计算预测值的标准误差来评价，图10-40显示了CLS的预测结果，由CLS方法得到的SEP = 67.1mg/dL。对于无干扰的葡萄糖监控系统来说，这个值与实际可接受的值相比有很大的误差[51]。

使用反向传播训练的MLP NN

MLP NN使用的网络结构如图10-41所示，使用MATLAB神经网络工具箱（第2版）训练MLP NN。用函数`initff`设置初始突触权值和偏置，网络中所有三层的激活函数都是`tansig`。MATLAB中的激活函数`tansig`是2.3节中提到的双曲正切*S*形激活函数。使用函数`trainbpx`训练网络，该网络所有三层的激活函数都是`tansig`。这个MLP NN训练函数由神

经网络工具箱提供，可自适应调整学习率参数。每个训练回合训练函数trainbpx都对输入（共100个）随机修改。初始学习率参数设为$10^{-4}$，动量设为0.9，平方误差和（SSE）设为$10^{-2}$。训练数据（输入数据即100个光谱和目标值即葡萄糖浓度）依照100个光谱振幅的最大值（max_input = 925.3427）按比例缩放。由于激活函数的限制必须这样处理训练数据。网络经过2920个训练回合达到要求的误差目标。在训练结束后，MATLAB神经网络工具箱中的simff函数用来预测测试光谱的测试浓度。在预测结束后，最重要的一步就是依照光谱振幅的最大值（即前面缩放训练数据时使用的max_input = 925.3427）按比例对测试浓度进行缩放调整。用MLP NN得到的SEP = 9.6mg/dl，图10-40b给出了用MLP NN预测的结果。对比图10-40a和图10-40b，我们看出回归结果有很大的区别。MLP NN能有效地从提供给它的输入光谱（见图10-39，葡萄糖光谱“隐含于”信号）中“学习”到“葡萄糖谱属性”的本质特点，不同的葡萄糖浓度对应于光谱振幅的相应谱信息。现在我们不禁要问：这些结果还能再提高吗？

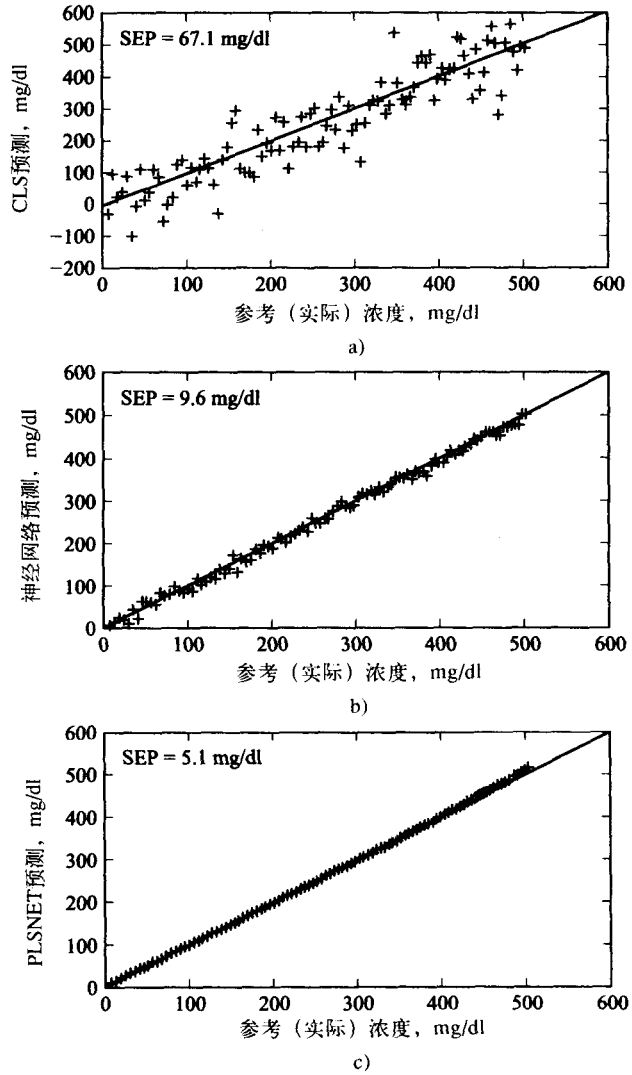



图10-40　a) 使用CLS预测葡萄糖浓度；b) 使用反向传播训练的MLP NN（100/30/1层网络）预测葡萄糖浓度；c) 使用PLSNET（因子数为2）预测葡萄糖浓度

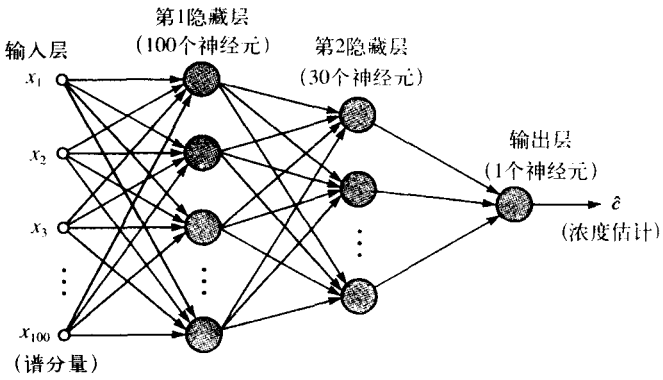



图10-41　用BP训练的三层MLP NN

532
~
533

PLSNET

PLSNET-C（详见9.6节）用来提取训练数据（可能100个）的前两个初始因数。例9.4中应用的因数分析法已经证明两个因数是最优的。图9-16给出了两个（因数）阶段的神经网络结构，初始权值设定为零平均、单位方差的随机正态分布数。学习率参数$\mu_w$、$\mu_b$和$\mu_v$分别设为：

$$\mu_w = \frac{0.1}{\sum_{j=1}^{100} c_{\text{train}\,j}^2} \tag{10-163}$$

和

$$\mu_b = \mu_v = 0.05\mu_w \tag{10-164}$$

训练10 000个回合后，网络收敛到图10-42所示的权值装载向量。这些值看起来与图9-20e所示相似。第一个权值装载向量与图9-20a所示的水光谱有相似的性质。由测试光谱预测测试浓度时，使用PLSNET-P与提取权值装载向量$\{\hat{\boldsymbol{w}}_1, \hat{\boldsymbol{w}}_2\}$、装载向量$\{\hat{\boldsymbol{b}}_1, \hat{\boldsymbol{b}}_2\}$和回归系数$\{\hat{v}_1, \hat{v}_2\}$。使用PLSNET-P得到SEP = 5.1 mg/dl，图10-40c显示了预测结果与实际葡萄糖浓度。对比三种方法可以清楚地看出，PLSNET比CLS回归或用BP训练的MLP NN能更好地预测葡萄糖浓度。

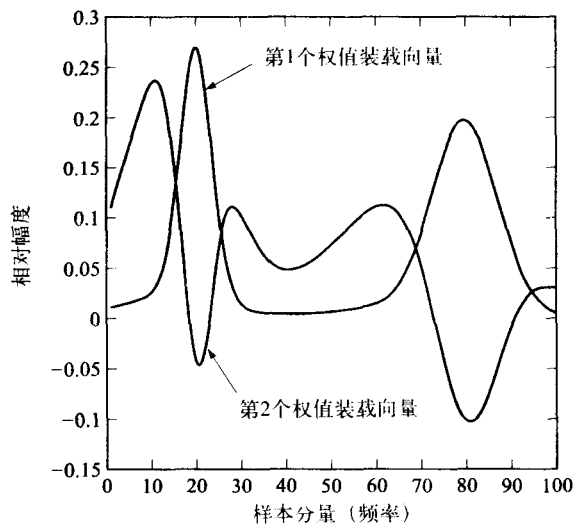



图10-42　由PLSNET-C从训练数据中提取的前两个初始权值装载向量

### 10.10.2 使用次声数据进行事件分类

本例中将应用次声数据分类自然事件，如火山活动和山组合波（详见10.8.2节）。 534
图10-43a显示了1982年乔吐Galunggung火山爆发的四信道信号（在波束形成之前）的一组记录，这些信号是在南极洲的Windless Bight用次声传感阵列（F阵列）记录的。这些次声信号是以标准采样频率1Hz采样得到的。图10-43b显示了在波束形成后的同一组信号。用波束形成处理信号是为了补偿信号在阵列传感器之间的时间延迟（见图10-23）。时间延迟补偿为的是使不同信道的四个信号能排列在任意的时间同步引用点上（图10-23中的CHF 1）。在应用波束形成前，用适当数目的零振幅时间样本"叠加"原始时域信号，为的是确保在信号调整过程中不会丢失信息。

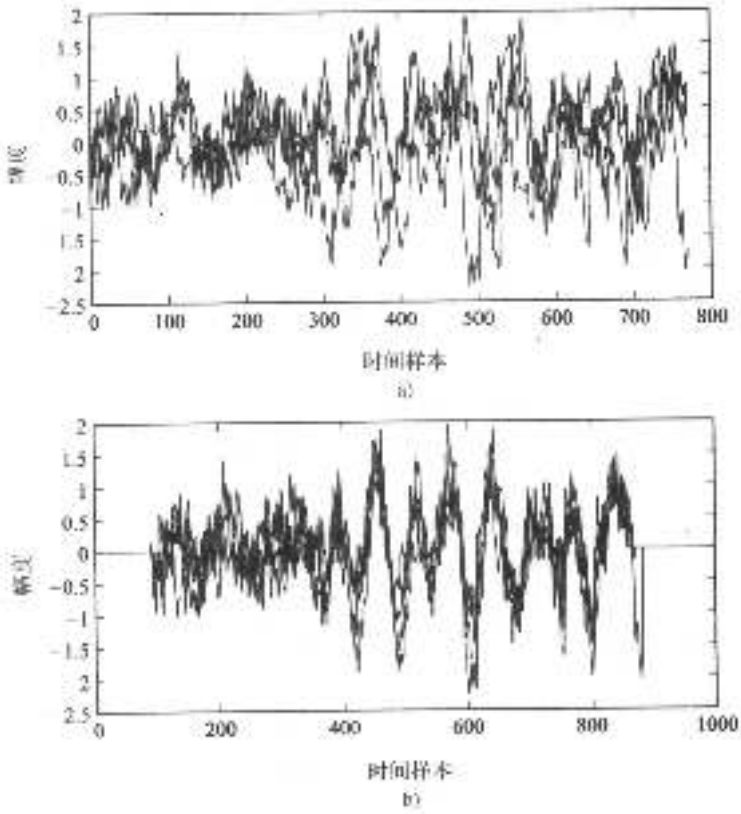



图10-43 a) 1982年乔吐Galunggung火山爆发的波束形成之前，来自南极洲Windless Bight的次声传感器的四信道信号集；b) 经波束形成后的a) 中的四个信号

使用一组包含152个次声信号的数据来训练和测试用BP训练的MLP NN[53-55]。这152个 535
信号由28个火山事件（VOL）和10个山组合波（MAW）[56, 57]组成。在这些火山事件中，6个来源于墨西哥的El Chichon火山爆发，22个来源于乔吐Galunggung火山爆发，两组火山次声数据都记录于1982年。1983年记录于南极洲的Windless Bight的MAW对应于新西兰山脉的方位波段。对于这两种事件（VOL和MAW）的每一个，都包含来自F阵列传感器的四个信号。152个信号分为一个训练数据集[76个信号：56个VOL（12个来自El Chichon和44个来自Galunggung）和20个MAW信号]和一个测试数据集（另一半数据）。当数据分类用作训练和测试时，四信道信号是合在一起使用的。

如图10-43b所示，经波束形成后的“原始”次声数据要用语音辨识系统中常用的方法进行预处理[58]。虽然这两种现象是完全不同的，但它们的性质有相似点，例如，虽然在信道（或传播媒介）和频率范围明显不同，但声源上又有些相同点。如在火山爆发和人类语音之间就有相似点。为了产生特征向量来进行训练和测试，用下面的步骤对原始信号进行预处理。

数据预处理步骤

1. 从信号中消除（减去）均值。

2. 对信号进行汉明窗口处理。

3. 计算信号的功率谱密度（PSD）。

4. 对PSD应用唛耳频率规模变换[59]。这是通常用于语音信号的信号倒谱自适应调整[47]，特别应用于语音辨识。对于前1步的PSD $S(k)$（$k$是离散频率），按照下式进行改变：

$$S_m(k) = \alpha \ln[\beta S(k)]$$

其中$\alpha = 1125$，$\beta = 0.0016$。这些都应用于唛耳频率语音数据的标准值（经验值）。

5. 对于前1步给出的$S_m(k)$，进行离散反余弦变换[47]得到：

$$x_m(n) = \frac{1}{N}\sum_{k=0}^{N-1} S_m(k)\cos\left(\frac{2\pi kn}{N}\right) \quad n = 0,1,2,\cdots,N-1$$

其中$N$是时域样本的总数，$x_m \in \Re^{1\times n}$。

6. 对序列$x_m(n)$求导，得到$x'_m(n)$。

536　7. 把导数序列$x'_m(n)$和倒频序列$x_m(n)$连接起来，组成扩展序列$x^a_m = [x'_m(n)|x_m(n)]$。

8. 对序列$x^a_m$的每个元素取绝对值得到$x^a_{m,\text{abs}} = |x^a_m|$。

9. 再对$x^a_m$取对数得到$x^a_{m,\text{abs,ln}} = \ln(x^a_{m,\text{abs}})$。

10. 最后，对得到的整个数据集关于最大振幅按比例缩放。

应用PLSR（详见9.5节）对特征空间进行预分析以确定多频倒频系数和关联导数系数的最佳组合。最佳组合是15个倒频导数与25个倒频系数。图10-44a显示了所选的两个火山爆发的特征表示向量，图10-44b显示了所选的MAW的特征表示向量。

由于信号经过了预处理，形成了训练和测试数据集，即$\{A_{\text{train}}, c_{\text{train}}\}$和$\{A_{\text{test}}, c_{\text{test}}\}$。矩阵$A_{\text{train}}$和$A_{\text{test}}$的每一行都包含40个特征向量，矩阵$c_{\text{train}}$和$c_{\text{test}}$由神经网络企图分类的特定事件的合适目标向量组成。二元目标向量建立如下：

$$[1, 0]^T = \text{火山} \quad \text{且} \quad [0, 1]^T = \textbf{MAW}$$

使用三种不同的事件分类器，并对它们的性能进行了比较。这三种分类器分别是：用BP训练的MLP NN、PLSR和RBF NN。

MLP NN是一个40/80/2层网络，用二元S形（对数$S$形）激活函数。每训练一次随机化输入特征向量，目标误差设为SSE = 0.1，初始学习率参数设为0.0001，动量调整参数设为0.9，使用MATLAB神经网络工具箱函数`trainbpx`[60]训练MLP NN。通过随机调整输入向量改变网络。网络的初始权值和偏置用MATLAB神经网络工具箱函数`initff`设置，用函数`simuff`仿真网络进行测试。

因为网络使用的目标向量是二元，不是标量，所以不能使用9.5节的PLSR算法。而是使用PLSR算法的一种更普遍的形式，它允许多分量目标。使用MATLAB中化学统计学工具`pls`函数[61]和函数`plspred`进行阶段（phase）测试。另外，函数`plspress`用来评价PLSR分类器的性能。这个MATLAB函数计算预测残留方误差和（PRESS），可以看作是9.4节提到的标准预测误差的一般形式。由于用多分量目标代替了标量，所以现在必须使用PRESS。对于所有

的目标值来说，它是预测方差的加权和。总共用15个PLS因数保留来进行预测是比较适宜的。

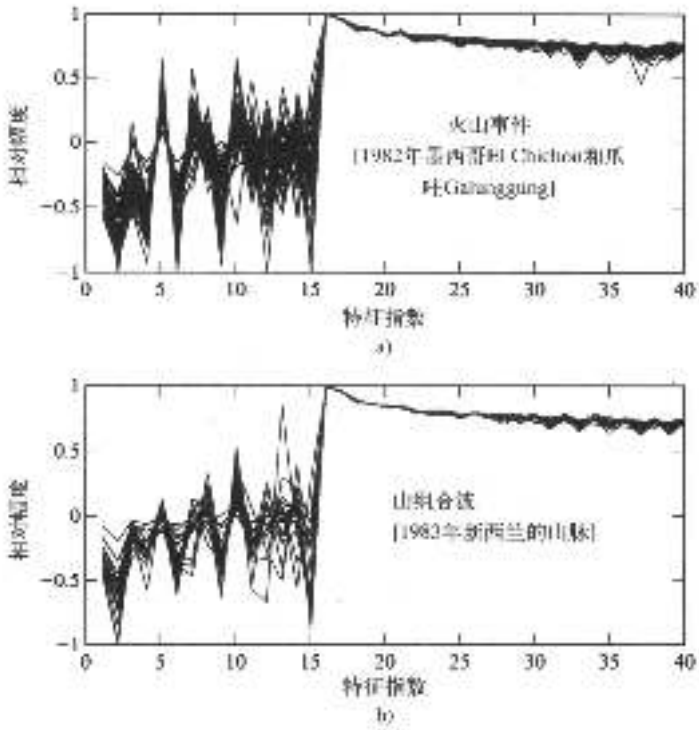



图10-44　a) 典型的火山爆发次声特征向量；b) 典型的MAW次声特征向量。两个特征向量数据集都由15个倒频导数和25个倒频系数组成

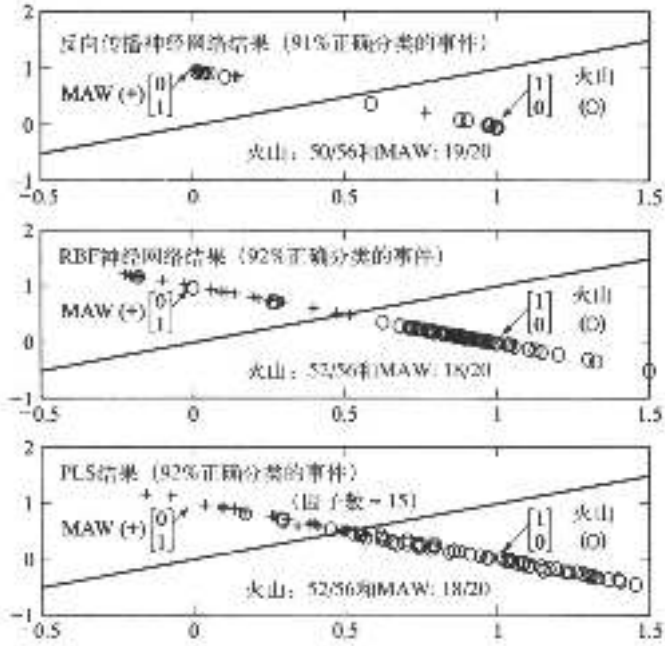



图10-45　用BP训练的MLP NN，PLSR和RBF NN的分类结果

RBF NN由54个隐神经元组成，高斯径向基函数的扩展参数等于0.775，目标SSE设为0.1。使用MATLAB神经网络工具箱函数solverb设计网络，使用函数simurb进行阶段测试。

图10-45显示了使用三类分类器的结果。用三种分类器对事件分类的准确率基本上是一样，但用BP训练的MLP NN比另外两种方法显示出了更强的鲁棒性，也就是说，用这种网络形成的聚集要比用RBF NN和PLSR“更紧凑”。

## 习题

10.1　假设有一个二阶离散时间系统，它的$z$变换传递函数由下式给出：

$$H(z)=\frac{Y(z)}{U(z)}=\frac{z-0.1}{z^2-0.8999+0.08}$$

采样周期$T_s=2\pi/1\ 000$s，系统的实际（真实）的维数为$\bar{n}=2$。因此，实际的参数向量$\boldsymbol{\theta}=[-0.8999, 0.08, 1, -0.1]^T$。

(a) 设系统的输入序列$\boldsymbol{u}(k)$（1024个样本）是零均值、单位方差的高斯噪声（即1024个样本取自零均值、单位方差的高斯分布）。在此情况下，用MATLAB函数dlsim生成模拟系统数据$\{\boldsymbol{u}(k), \boldsymbol{y}(k)\}$。

(b) 若系统维数（过）指定为$n=2$，由部分(a)中生成的数据组成用于训练和测试的ARMA数据矩阵，用前100个样本训练数据$\{\boldsymbol{\Phi}_{\text{train}}(N), \boldsymbol{y}_{\text{train}}(N)\}$，用随后的100个样本组成测试数据$\{\boldsymbol{\Phi}_{\text{test}}(N), \boldsymbol{y}_{\text{test}}(N)\}$。

(c) 利用PLSR（详见9.5节）确定最佳因子数目，即利用由部分(b)得到的训练和测试数据，进行独立检验因子分析找到系统的“真实”维数。运用9.5节给出的PLSR1校准算法和PLSR1预测算法（方法2）。

(d) 生成最后的测试数据集，即，最后数据集$=\{\boldsymbol{\Phi}_f(N;\hat{n}), \boldsymbol{y}_f(N;\hat{n})\}$，其中$\hat{n}$是根据部分(c)的结果确定的系统维数的估计值。从这个最后数据集确定用于PLSR1预测算法（方法1）的参数向量。

(e) 应用训练数据画出初始系统的输出图像。在同一幅图上画出PLSR1预测算法（方法2）应用训练数据和由部分(c)确定的系统维数得出的离散时间样本。

537～539

(f) 使用PLSNET-P代替PLSR1校正算法重新对以上部分做一次，确定系统的权值装载向量、装载向量和回归系数。确定参数向量并把得到的结果同前面得到的参数向量进行比较。

(g) 如果系统是可简化的，就只应用PLSR1校验算法和PLSR1预测算法（方法1），对部分(b)生成的初始测试数据集确定参数向量。

10.2　设定传递函数由下式给出，重复问题10.1的所有步骤。

$$H(z)=\frac{Y(z)}{U(z)}=\frac{1}{z^2-z+0.5}$$

在部分(b)假定系统维数（过）指定为$n=4$。

10.3　一个非线性系统由下面的差分方程给出：

$$y_p(k)=0.2y_p(k)+0.5y_p(k-1)+y_p(k-2)+f[u(k)]$$

其中

$$f(u)=4\sin(4u)+\frac{1}{1+u^2}$$

可以看出，动态系统与10.6.2节描述的模型1相对应。

(a) 证明系统是稳定的，而且对于任意的有界输入序列都有相应的有界输出序列（BIBO稳定标准）。

(b) 假定函数$f(u)$是未知的，用下面的串行-并行模型设计一个神经网络辨识器：

$$\hat{y}(k+1) = 0.2y_p(k) + 0.5y_p(k-1) + y_p(k-2) + N[u(k)]$$

其中$N[u(k)]$是由SISO的RBF NN进行的映射。在隐藏层中取不同的神经元数目进行实验，在辨识过程中，用区间[−2, 2]中均匀随机分布的输入。以模型训练区间以外的输入测试模型的性能，并评价模型。

(c) 使用MLP NN重新做部分(b)。

10.4　考虑一个多路信号传播的非线性传输信道的例子。假设由输入/输出方程表示的信道能近似如下：

$$y(k) = 2\arctan\{0.5[x(k) - 0.1x(k-1) + 0.4x(k-3)]\} + v(k)$$

其中$v(k)$是信道携带的白噪声。图10-46给出了一个用于补偿非线性信道的非线性均衡器的例子。

(a) 假定$v(k)$是零均值，标准方差$\sigma = 0.1$的高斯白噪声。要求生成随机双极输入序列，然后把上面的方程用于非线性信道，生成相应的输出序列。

(b) 对于图10-46的均衡器结构图，训练神经网络来实现均衡过程，用RBF或MLP NN都可以。用不同结构、不同规模的网络进行实验。如果要求均衡器在多路变换的数量和延迟随时间改变情况下能正常工作，应该选取哪种神经网络结构？ 540

(c) 求出用于均衡器结构的最小延迟选择数？并加以证明。

(d) 如果数据位速率为100kbits/s，期望的最大多路延迟是50$\mu$s，估计实现均衡功能的非线性均衡器所需的选择数。

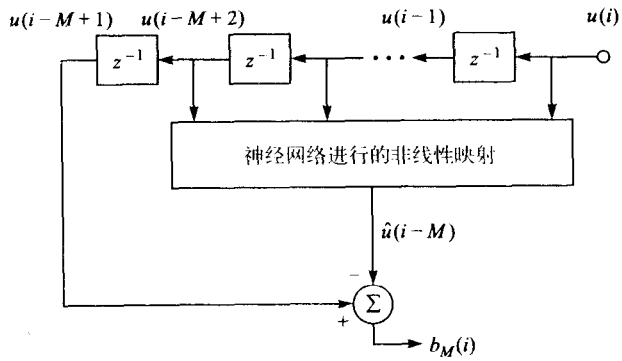



图10-46　用于神经网络的非线性均衡

10.5　本题我们准备盲分离两个信号。第一个是零均值方波信号，第二个是区间[−1, 1]中均匀分布的噪声信号。这些信号可用下面的MATLAB程序代码产生。

```
% Generate Two Signals
t=0:0.001:0.199;
% Deterministic Square Wave
s1=square(2*pi*30*t);
% Impulse Noise in the Interval [-1, 1]
s2=2*(rand(1,200)-0.5*ones(1,200));
% Plot Figures
```

```
subplot(2,1,1), plot(t,s1), axis([0 0.2 -1.5 1.5])
title('Square Wave'), xlabel('Time (sec)')
ylabel('Amplitude')
subplot(2,1,2), plot (t,s2), axis([0 0.2 -1.5 1.5])
title('Impulse Noise'), xlabel('Time (sec)')
ylabel'Amplitude')
```

信号s1、s2是源信号。应用混合矩阵

$$A=\begin{bmatrix}0.2258 & 0.3686\\ -0.1013 & 0.1264\\ -0.1416 & -0.2588\\ -0.2147 & 0.4781\end{bmatrix}$$

生成四个可观察信号$X = AS$，其中$X\in\Re^{4\times 200}$（每行一个可观察信号），$S$ = [s1; s2]。

541 假定只有这4个可观察信号是有效信息，执行下面的步骤：

(a) 使用式（10-116）的批处理方法预漂白可观察的数据。为了确定源信号的数目，很有必要对可观察数据进行PCA（即使数目很明显也要进行这一步骤）。在这个过程中，必须确认$X$中的四个可观察信号的平均值已删除，漂白过程的输出按比例调整。

(b) 使用部分(a) 的结果，运用式（10-111）的学习规则确定分离权值矩阵$W$，其中非线性函数$g(\cdot)$是logistic函数的导数。要用$y(k)$代替$v(k)$，按照式（10-108）调整学习率参数。使用下面的权值矩阵初始化你所建的网络：

$$W(0)=\begin{bmatrix}0.9762 & -0.2171\\ 0.2171 & 0.9762\end{bmatrix}$$

(c) 由部分(b) 得到的独立成分（即$y = W^T v$）是最初的源信号（$y = \hat{s}$）。计算两个分离信号和两个初始源信号之间的相关系数（计算四个数字）。这些系数中的其中两个要比另外两个大一些。

10.6 重新做一次问题10.5，但现在需要分离四个不同的源信号：三个确定信号（方波、锯齿波和正弦波）和一个区间[ − 1, 1]之间的均匀分布噪声信号。这些信号可由下面的MATLAB代码产生：

```
% Generate Four Signals
t = 0:0.001:0.199;
s1 = square(2*pi*30*t);
% Sine Wave
s2=sin(2*pi*45*t);
% Sawtooth Waveform
s3=sawtooth(2*pi*50*t);
% Impulse Noise in the Interval [-1,1]
s4=2*(rand(1,200)-0.5*ones(1,200));
% Plot Figures
subplot(4,1,1), plot(t,s1), axis([0 0.2 -1.5 1.5])
title('Square Wave'), xlabel('Time (Sec)')
ylabel('Amplitude')
subplot(4,1,2), plot(t,s2), axis([0 0.2 -1.5 1.5])
title('Sine Wave'), xlabel('Time (Sec)')
ylabel('Amplitude')
subplot(4,1,3), plot(t,s3), axis([0 0.2 -1.5 1.5])
title('Sawtooth Waveform'), xlabel('Time (Sec)')
ylabel('Amplitude')
subplot(4,1,4), plot(t,s4), axis([0 0.2 -1.5 1.5])
title('Impulse Noise'), xlabel('Time (Sec)')
ylabel('Amplitude')
```

542

信号s1、s2、s3和s4是源信号。利用混合矩阵

$$\boldsymbol{A}=\begin{bmatrix} 0.4501 & -0.4815 & -0.3237 & -0.1471 \\ -0.2689 & 0.3214 & -0.0943 & 0.3132 \\ 0.1068 & -0.0553 & 0.4355 & -0.4901 \\ -0.0140 & 0.1154 & 0.4169 & -0.3611 \\ 0.3913 & 0.2919 & -0.0897 & -0.2972 \\ 0.2621 & 0.4218 & 0.3936 & -0.3013 \\ -0.0435 & 0.2382 & -0.4421 & 0.1038 \end{bmatrix}$$

生成七个可观察的信号$\boldsymbol{X}=\boldsymbol{AS}$，其中$\boldsymbol{X}\in\Re^{7\times 200}$（每一行是一个可观察信号），$\boldsymbol{S}$ = [s1; s2;s3;s4]。假定可用信息只有四个可观察信号。使用下面的权值矩阵初始化你的网络：

$$\boldsymbol{W}(0)=\begin{bmatrix} 0.5628 & -0.0889 & -0.7278 & -0.3816 \\ -0.1329 & 0.1824 & -0.5475 & 0.8058 \\ 0.6855 & 0.6051 & 0.3456 & 0.2109 \\ 0.4424 & -0.7698 & 0.2259 & 0.4008 \end{bmatrix}$$

10.7 用第10.8.3节给出的ICA的快速固定点算法来解问题10.6，把所得结果与应用神经网络方法（即式（10-111）给出的（Karhunen-Oja）非线性PCA子空间学习规则）的结果相比较。在收敛速度和结果的精确度上两者相比怎样？你能得出什么样的结论？

10.8 设计一个自适应网络（即一个单神经元处理器），用来估计带干扰信号$e(t)$的周期信号$y(t)$（即$z(t)=y(t)+e(t)$）的参数$\{\alpha_1,\alpha_2,\alpha_3\}$和$\{\beta_1,\beta_2,\beta_3\}$。因此，$z(t)$是一个可观察的信号，或是一个可观察的噪声信号

$$y(t)=\alpha_1\sin(\omega t)+\beta_1\cos(\omega t)+\alpha_2\sin(5\omega t)+\beta_2\cos(5\omega t)$$
$$+\alpha_3\sin(7\omega t)+\beta_3\cos(7\omega t)$$

其中$\omega=2\pi f\big|_{f=1\text{Hz}}$。噪声为零均值高斯白噪声，方差$\sigma^2=0.1$。在仿真函数时，假定采样频率$f_s=1\ 000\text{Hz}$，生成1025个数据点。测试你的自适应网络的性能，参数设定为$\alpha_1=\beta_1=1$，$\alpha_2=0.5$，$\beta_2=1/\alpha_2$，$\alpha_3=0.25$，$\beta_3=1/\alpha_3$。

10.9 对于频率估计问题，在10.9节中我们用PLS方法处理线性单步预测过滤器的选择权值（见式(10-157)和式(10-158)），在第9章中，我们用PCR方法来处理同一问题（见9.4节）。

(a) 用PCR对估计频率公式化。

(b) 写一个MATLAB程序，实现你的基于PCR的谱估计算法。把PCR因子数作为其中程序的输入参数。

(c) 使用例10.10给出的谱估计问题测试你的程序。

(d) 把你的结果和使用PLSR所得结果进行比较。

543

10.10 在频率估计问题中，单步预测过滤器的长度是很重要的。假定给出一个四个正弦信号叠加的信号如下：

$$u(i)=\cos[2\pi(0.1i)]+\cos\left[2\pi(0.2i)+\frac{\pi}{4}\right]+\cos[2\pi(0.38i)]+\cos[2\pi(0.4i)]+\vartheta(i)$$

其中$\vartheta(i)$是附加噪声信号的。假设噪声是零均值高斯噪声，标准偏差为0.35。假定在

处理中有$N = 40$个可用的样本信号。

(a) 处理这个问题的预测过滤器的最小长度是多少?

(b) 写一个MATLAB程序，用CLS处理实现单步预测过滤器的系数问题（见式（9-98）和式（9-99））。允许过滤器的阶数（延迟线上的选择数目）是其中一个输入参数。

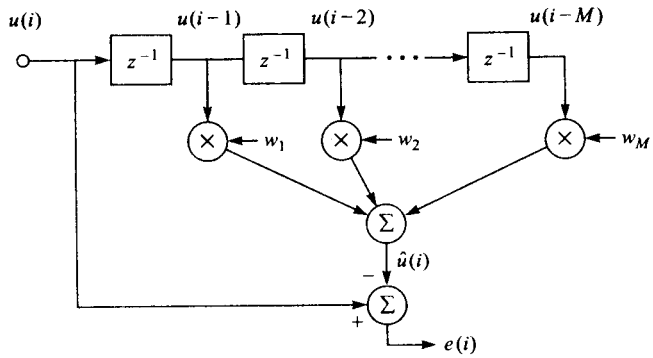
(c) 用你所写的CLS程序处理上面所给信号的正弦频率估计问题，用预测过滤器的不同的阶数进行实验。

(d) Lang和McClellan[62]建议过滤器的阶数$M = N/3$。但Tuffs和Kumaresan[63]经过实验确定过滤器的阶数$M = 3N/4$。在你的实验中分别运用这些研究者的建议值，并对它们的结果进行比较。

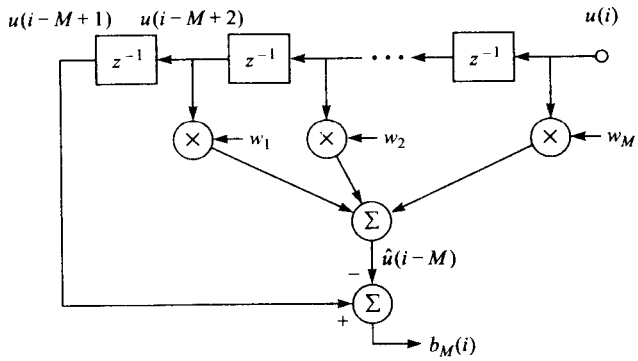
M 10.11　在介绍谱估计问题时我们曾用到单步预测方法。这种方法能得到如式（10-152）的回归方程组。通常，谱估计问题也能用图10-47所示的前向-后向线性预测器来阐明，此时，回归模型依照如下等式来说明：

前向预测误差：

$$f_M(i) = u(i) - \sum_{k=1}^{M} w_k u(i-k)$$




a)




b)

图10-47　a) 单步前向预测过滤器；b) 单步后向预测过滤器

后向预测误差：

$$b_M(i) = u(i-M) - \sum_{k=1}^{M} w_k u(i-M+k)$$

最优标准设置为：

$$J(\boldsymbol{w}) = \sum_{i=M+1}^{N} \left[ |f_M(i)|^2 + |b_M(i)|^2 \right]$$

回归模型形式为：

$$\boldsymbol{c} = \boldsymbol{A}\boldsymbol{w} + \boldsymbol{e}$$

其中

$$\boldsymbol{A} = \begin{bmatrix} u(M) & u(M-1) & \cdots & u(1) \\ u(M+1) & u(M) & \cdots & u(2) \\ \cdots & \cdots & \cdots & \cdots \\ u(N-1) & u(N-2) & \cdots & u(N-M) \\ \cdots & \cdots & \cdots & \cdots \\ u(2) & u(3) & \cdots & u(M+1) \\ u(3) & u(4) & \cdots & u(M+2) \\ \cdots & \cdots & \cdots & \cdots \\ u(N-M+1) & u(N-M+2) & \cdots & u(N) \end{bmatrix}$$

$$\boldsymbol{c} = [u(M+1), u(M+2), \cdots, u(N) | u(1), u(2), \cdots, u(N-1)]^T$$

和

$$\boldsymbol{e} = [f_M(M+1), \cdots, f_M(N) | b_M(M+1), \cdots, b_M(N)]^T$$

如果求得过滤器的系数，那么可利用式（10-158）估计频谱。

(a) 使用问题10.10中所定义的信号$u(i)$，阐明前向－后向预测问题。

(b) 使用PLSR技术解决这一问题。

(c) 进行实验，以确定前向－后向预测过滤器的最佳长度。

544
~
545

## 参考文献


1. L. Ljung, *System Identification: Theory for the User*, Englewood Cliffs, NJ: Prentice-Hall, 1987.
2. G. F. Franklin, J. D. Powell, and M. L. Workman, 2nd ed., *Digital Control of Dynamical Systems*, 2nd ed., Reading, MA: Addison-Wesley, 1990.
3. T. Söderström and P. Stoica, *System Identification*, New York: Prentice-Hall, 1989.
4. J.-N. Juang, *Applied System Identification*, Englewood Cliffs, NJ: Prentice-Hall, 1994.
5. C.-T. Chen, *Linear System Theory and Design*, 2nd ed., New York: Oxford University Press, 1999.
6. P. L. Bogler, *Radar Principles with Applications to Tracking Systems*, New York: Wiley, 1990.
7. N. R. Draper and H. Smith, *Applied Regression Analysis*, 2nd ed., New York: Wiley, 1981.

8. A. M. Qabazard, "System Identification Using Partial Least Squares," Ph.D. Dissertation, Melbourne, FL: Florida Institute of Technology, 1995.
9. F. M. Ham and I. Kostanic, "A Partial Least-Squares Regression Neural NETwork (PLSNET) with Supervised Adaptive Modular Learning," *Applications and Science of Artificial Neural Networks II*, eds. S. K. Rogers and D. W. Ruck, Proceedings of SPIE, vol. 2760, 1996, pp. 139–50.
10. F. M. Ham and I. Kostanic, "Partial Least-Squares: Theoretical Issues and Engineering Applications in Signal Processing," *Mathematical Problems in Engineering*, vol. 2, 1995, pp. 63–93.
11. K. S. Narendra, and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, 1990, pp. 4-27.
12. I. J. Leontaritis and S. A. Billings, "Input-Output Parametric Models for Non-Linear Systems. Part I: Deterministic Non-Linear Systems, Part II: Stochastic Non-Linear Systems," *International Journal of Control*, vol. 41, 1987, pp. 303–44.
13. H. L Royden, *Real Analysis*, New York: Macmillan, 1964.
14. S. A. Billings, S. Chen and M. J. Korenberg, "Identification of MIMO Non-linear Systems Using Forward-Regression Orthogonal Estimator," *International Journal of Control*, vol. 49, 1989, pp. 2157–89.
15. A. Desrochers and S. Mohseni, "On Determing the Structure of Non-linear Systems," *International Journal of Control*, vol. 40, 1984, pp. 923–938.
16. S. Chen, S. A. Billings, and W. Luo, "Orthogonal Least-Squares Methods and Their Application to Non-Linear System Identification," *International Journal of Control*, vol. 50, 1989, pp. 1873–86.
17. K. S. Narendra and K. Parthasarathy, "Gradient Methods for Optimization of Dynamic Systems Containing Neural Networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, 1991, pp. 252–62.
546
18. S. Z. Quin, H. T Su, and T. J. McAvoy, "Comparison of Four Neural Net Learning Methods for Dynamic System Identification," *IEEE Transactions on Neural Networks*, vol. 3, no. 1, 1992, pp. 122–30.
19. S. Chen, S. A. Billings, C. F. N. Cowan, and P. M. Grant, "Practical Identification of NARMAX Models Using Radial Basis Functions," *International Journal of Control*, vol. 52, no. 6, 1990, pp. 1327–50.
20. S. Chen, S. A. Billings, C. F. N. Cowan and P. M. Grant, "Non-linear System Identification Using Radial Basis Functions," *International Journal of Systems Science*, vol. 21, no. 12, 1990, pp. 2513–39.
21. P. J. Werbos, "Generalization of Backpropagation with Application to Recurrent Gas Market Model," *Neural Networks*, vol. 1, no. 4, 1988, pp. 270–80.
22. K. S. Narendra and A. M. Annaswamy, *Stable Adaptive Systems*, Englewood Cliffs, NJ: Prentice Hall, 1989.
23. C. Jutten and J. Herault, "Independent Component Analysis (INCA) versus Principal," in *Signal Processing IV: Theories and Applications*, Proc. *EUSIPCO-88*, eds. J. Lacoume, et al., Amsterdam, The Netherlands: Elsevier, 1988, pp. 643–6.
24. C. Jutten and J. Herault, "Blind Separation of Sources, Part I: An Adaptive Algorithm Based on Neuromimetic Archicture," *Signal Processing*, vol. 24, 1991, pp. 1–10.
25. P. Comon, "Independent Component Analysis," in *Proceedings of the International Signal Processing Workshop on Higher-Order Statistics*, Chamrousse, France, 1991, pp. 111–20. [Republished in *Higher-Order Statistics*, ed. J. L. Lacoume, Amsterdam, The Netherlands: Elsevier, 1992, pp. 29–38.]
26. P. Comon, "Independent Component Analysis—A New Concept?" *Signal Processing*, vol. 36, 1994, pp. 287–314.

27. J. M. Mendel, "Tutorial on Higher-Order Statistics (Spectra) in Signal Processing and System Theory: Theoretical Results and Applications," *Proceedings of the IEEE*, vol. 79, 1991, pp. 278–305.
28. C. L. Nikias and J. M. Mendel, "Signal Processing with Higher-Order Spectra," *IEEE Signal Processing Magazine.*, vol. 10, 1993, pp. 10–37.
29. J. Karhunen, E. Oja, L. Wang, R. Vigario, and J. Joutsensalo, "A Class of Neural Networks for Independent Component Analysis," *IEEE Transactions on Neural Networks*, vol. 8, 1997, pp. 486–504.
30. J.-F. Carduso and B. H. Laheld, "Equivariant Adaptive Source Separation," *IEEE Transactions on Signal Processing*, vol. 44, 1996, pp. 3017–30.
31. J. Karhunen, "Neural Approaches to Independent Component Analysis and Source Separation," In *Proceedings of 4th European Symposium on Artificial Neural Networks, ESANN'96*, Bruges, Belgium, April 1996, pp. 249–66.
32. A. Hyvärinen, "Fast and Robust Fixed-Point Algorithms for Independent Component Analysis," *IEEE Transactions on Neural Networks*, vol. 10, 1999, pp. 626–34.
33. E. Oja, H. Ogawa, and J. Wangviwattana, "Learning in Nonlinear Constrained Networks," in *Artificial Neural Networks Proceedings of ICANN-91*, eds. T. Kohonen et al., Amsterdam, The Netherlands: North-Holland, 1991, pp. 385–90.
34. J. Karhunen and J. Joutsensalo, "Representation and Separation of Signals Using Nonlinear PCA Type Learning," *Neural Networks*, vol. 7, 1994, pp. 113–27.
35. J. Karhunen and J. Joutsensalo, "Generalizations of Principal Component Analysis, Optimization Problems, and Neural Networks," *Neural Networks*, vol. 8, 1995, pp. 549–62.
36. L. Wang and E. Oja, "A Unified Neural Bigradient Algorithm for Robust PCA and MCA," International Journal of Neural Systems., vol. 7, 1996, pp. 53–67. 547
37. L. Wang, J. Karhunen, and E. Oja, "A Bigradient Optimization Approach for Robust PCA, MCA, and Source Separation," In *Proceedings of 1995 IEEE International Conference on Neural Networks*, Perth, Australia, November 1995, pp. 1684–9.
38. V. N. Valentina, *Microseismic and Infrasonic Waves*, Research Reports in Physics, New York: Springer Verlag, 1992.
39. A. D. Pierce, *Acoustics: An Introduction to Its Physical Principles and Applications*, Swickley, PA, Acoustical Society of America, 1989.
40. C. R. Wilson and R. B. Forbes, "Infrasonic Waves from Alaskan Volcanic Eruptions," *Journal of Geophysical Research.* vol. 74, 1969, pp. 1812–36.
41. A. J. Bedard, "Infrasound Originating near Mountain Regions in Colorado," *Journal of Applied Meteorology*, vol. 17, 1978, p. 1014.
42. National Research Council, *Comprehensive Nuclear Test Ban Treaty Monitoring*, Washington: National Academy Press, 1997.
43. K. B. Payne, *Silent Thunder: In the Presence of Elephants*, New York: Simon & Schuster, 1998.
44. F. M. Ham, N. A. Faour, and J. C. Wheeler, "Infrasound Signal Separation Using Independent Component Analysis," in *Proceedings of the 21st Seismic Research Symposium: Technologies for Monitoring the Comprehensive Nuclear-Test-Ban Treaty*, Las Vegas, NV, Sept. 21–24, 1999, vol. 2, pp. 133–40.
45. A. Hyvärinen and E. Oja, "A Fast Fixed-Point Algorithm for Independent Component Analysis," *Neural Computation*, vol. 9, 1997, pp. 1483–92.
46. G. Deco and D. Obradovic, *An Information-Theoretic Approach to Neural Computing*, New York: Springer-Verlag, 1996.
47. J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 3rd ed., Upper Saddle River, NJ: Prentice-Hall, 1996.

48. S. Haykin, *Adaptive Filter Theory*, 2nd ed., Upper Saddle River, NJ: Prentice-Hall, 1991.
49. *Proceedings of the IEEE—Special Issue on Spectral Estimation*, vol. 70. no. 9, 1982.
50. K. Hamid and M. Viberg, "Two Decades of Array Signal Processing Research, the Parametric Approach," *IEEE Signal Processing Magazine*, vol. 13, no. 4, 1996, pp. 67–94.
51. F. M. Ham, I. Kostanic, G. M. Cohen, and B. R. Gooch, "Determination of Glucose Concentrations in an Aqueous Matrix from NIR Spectra Using Optimal Time-Domain Filtering and Partial Least-Squares Regression," *IEEE Transactions on Biomedical Engineering*, vol. 44, 1997, pp. 475–85.
52. F. M. Ham and I. Kostanic, "Partial Least-Squares: Theoretical Issues and Engineering Applications in Signal Processing," *Journal of Mathematical Problems in Engineering*, vol. 2, 1996, pp. 63–93.
53. F. M. Ham, T. A. Leeney, H. M. Canady, and J. C. Wheeler, "Discrimination of Volcano Activity and Mountain Associated Waves Using Infrasonic Data and a Backpropagation Neural Network," *SPIE Conference on Applications and Science of Computational Intelligence II*, eds. K. L. Priddy, P. E. Keller, D. B. Fogel, and J. C. Bezdek, vol. 3722, April 1999, pp. 344–56.
54. F. M. Ham, T. A. Leeney, H. M. Canady, and J. C. Wheeler, "Volcano Event Classification Using Infrasonic Data and a Backpropagation Neural Network" (abstract), *EOS Transactions (Supplement), AGU*, vol. 79, no. 45, 1998, p. F620.
55. F. M. Ham, T. A. Leeney, H. M. Canady, and J. C. Wheeler, "An Infrasonic Event Neural Network Classifier," *International Joint Conference on Neural Networks*, Washington, DC, July 10–16, 1999, vol. 6, pp. 3768–73.
548
56. C. R. Wilson, J. V. Olson, and R. Richards, "Library of Typical Infrasonic Signals," Report Prepared for ENSCO, Inc., Melbourne, FL (Subcontract no. 269343-2360.009), vols. 1–4, 1996.
57. K. D. Hutchenson, "Acquisition of Historical Infrasonic Data," Final Technical Report, Contract no. F08650-95-D-0033, ARS-97-012, ENSCO, Inc., Melbourne, FL, 1997.
58. R. J. Mammone, X. Zhang, and R. P. Ramachandran, "Robust Speaker Recognition: A Feature-Based Approach," *IEEE Signal Processing Magazine*, vol. 13, 1996, pp. 58–71.
59. S. B. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, 1980, pp. 357–66.
60. H. Demuth and M. Beale, *Neural Network Toolbox—For Use with MATLAB*, Natick, MA: MathWorks, 1995.
61. R. Kramer, *Chemometrics Toolbox—For Use with MATLAB*, Natick, MA: MathWorks, 1993.
62. S. W. Lang and J. H. McClellan, "Frequency Estimation with Maximum Entropy Spectral Estimation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-28, 1980, pp. 716–24.
63. D. W. Tuffs and R. Kumaresan, "Estimation of Frequencies of Multiple Sinusoids: Making Linear Perform Like Maximum Likelihood," *Proceedings of IEEE*, vol. 70, 1982, pp. 975–89.
549

# 附录A 神经计算的数学基础

## A.1 引言

为了正确分析和设计人工神经网络，需要有坚实的数学基础。这个附录有两个目的：介绍在研究神经网络时所需要用到的线性代数、非线性规划、一般系统理论和随机过程领域的一些必要的数学结论；介绍一些经过选择的数学定义并且约定了全书中经常使用的符号。目的是要介绍神经计算的必备的数学背景知识，而不关注数学表达式的严格“定理证明”。通过本附录提供的资料，读者可以在掌握了研究神经网络的必备数学技能知识的基础上进行工作。

## A.2 线性代数

### A.2.1 域和向量空间

域

**定义A.1** 一个域$\mathcal{F}$包含一组元素和两个运算，即加法和乘法。在$\mathcal{F}$中定义的这两个运算需满足如下条件：

1. 对于$\mathcal{F}$中的每对元素$\alpha$和$\beta$，对应的元素$\alpha+\beta$也属于$\mathcal{F}$，称之为$\alpha$与$\beta$的和；元素$\alpha\cdot\beta$也属于$\mathcal{F}$，称之为$\alpha$与$\beta$的积。 550

2. 加法和乘法都满足交换律，即对于$\mathcal{F}$中的任意$\alpha$和$\beta$，有$\alpha+\beta=\beta+\alpha$和$\alpha\cdot\beta=\beta\cdot\alpha$。

3. 加法和乘法都满足结合律，即对于$\mathcal{F}$中任意的$\alpha$，$\beta$，$\gamma$，有$(\alpha+\beta)+\gamma=\alpha+(\beta+\gamma)$和$(\alpha\cdot\beta)\cdot\gamma=\alpha\cdot(\beta\cdot\gamma)$。

4. 乘法对加法满足分配律，即对于$\mathcal{F}$中任意的$\alpha$，$\beta$，$\gamma$，有$\alpha\cdot(\beta+\gamma)=(\alpha\cdot\beta)+(\alpha\cdot\gamma)$。

5. $\mathcal{F}$中包含两个特殊的元素，一个用0表示，一个用1表示，对于$\mathcal{F}$中任意元素$\alpha$，满足$\alpha+0=\alpha$和$1\cdot\alpha=\alpha$。

6. 对于$\mathcal{F}$中的每一个元素$\alpha$，存在一个元素$\beta\in\mathcal{F}$，使得$\alpha+\beta=0$。这个元素$\beta$称为$\alpha$的负元素。

7. 对于$\mathcal{F}$中的每一个非零元素$\alpha$，存在一个元素$\gamma$，使得$\alpha\cdot\gamma=1$，这个元素$\gamma$称为$\alpha$的逆元素。

所有的实数及其加法和乘法运算构成一个域，称为实数域$\mathfrak{R}$。复数及其加法和乘法运算也构成一个域，即复数域$\mathcal{C}$。对于复数集合，一个复数$\sigma=\alpha+j\beta$（其中$j=\sqrt{-1}$）的乘法逆运算可以写作$1/\sigma=\overline{\sigma}/|\sigma|^2$，其中$\overline{\sigma}=\alpha-j\beta$（称为$\sigma$的共轭复数），$|\sigma|=\sqrt{\alpha^2+\beta^2}$（称为$\alpha$的数量或模）。与复数$\sigma$对应的辐角定义为$\angle\sigma=\arg(\sigma)=\tan^{-1}(\beta/\alpha)$。

对于通常意义下的加法和乘法，集合$\{0,1\}$是无法形成域的。这很容易看出，因为$1+1=2$不属于集合$\{0,1\}$。然而，我们可以这样定义运算：

$0+0=0$ $1+0=1$ $1+1=0$ $0\cdot1=0$ $0\cdot0=0$和$1\cdot1=1$

这样，集合$\{0,1\}$就构成一个域（即，集合$\{0,1\}$与定义的加法和乘法满足上面列出的域的七个条件）。这个域称为二进制数域。

对于所有形如$\begin{bmatrix} w & -z \\ z & w \end{bmatrix}$的$2\times2$的矩阵构成的集合，其中$w$和$z$是任意实数，与矩阵加法和

乘法的标准定义构成域。在这种情况下，域中的元素0和1分别是零矩阵 $\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$ 和单位矩阵 $\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$。但是，所有形式的$2\times 2$矩阵的集合并不能构成域，例如，在一些情况下乘法矩阵的逆并不存在。

正实数集合并不能构成域，因为正实数的负元素在该域中不存在。整数集合不能构成域，
551 因为乘法的逆运算不存在。除了A.1定义中的第七条性质，满足其他六条性质的元素构成的集合称为环。多项式集合也无法形成域，因为乘法的逆运算也不存在。

向量空间

向量空间有一个简单的几何解释。例如，在普通的二维几何平面，如果定义一个参考点为原点，这样平面内每一个点都可以看作一个向量。也就是说，在平面内所有由原点指向任意一个点的“箭头”（向量），有各自的方向和数量。每一个向量可以进行收缩和扩大，并且任两个向量可以相加。但是两个向量不能相乘（也就是，没有定义向量乘法）。这个平面称为向量空间（或线性空间，或线性向量空间）。向量空间总是定义一个特殊的域，包括数乘（向量的收缩和扩大）和向量加法。

**定义A.2**　一个域$\mathcal{F}$中的向量（线性）空间记做（$\mathcal{X}$，$\mathcal{F}$），包含元素集$\mathcal{X}$，称为向量（任意长度），一个域$\mathcal{F}$，和两种运算，即*数乘*和*向量加法*。在$\mathcal{X}$和$\mathcal{F}$上定义的两种运算必须满足以下条件：

1. 对于向量集$\mathcal{X}$中的每一对向量$\boldsymbol{x}_1$和$\boldsymbol{x}_2$，相应地$\boldsymbol{x}_1+\boldsymbol{x}_2$也在$\mathcal{X}$中，称为$\boldsymbol{x}_1$和$\boldsymbol{x}_2$的和。

2. 向量加法满足交换律，即对于$\mathcal{X}$中的任意$\boldsymbol{x}_1$，$\boldsymbol{x}_2$，有$\boldsymbol{x}_1+\boldsymbol{x}_2=\boldsymbol{x}_2+\boldsymbol{x}_1$。

3. 向量加法满足结合律，即对于$\mathcal{X}$中的$\boldsymbol{x}_1$，$\boldsymbol{x}_2$，$\boldsymbol{x}_3$，有$(\boldsymbol{x}_1+\boldsymbol{x}_2)+\boldsymbol{x}_3=\boldsymbol{x}_1+(\boldsymbol{x}_2+\boldsymbol{x}_3)$。

4. 向量集合$\mathcal{X}$包含一个向量表示为$\mathbf{0}$，对于$\mathcal{X}$中的每一个$\boldsymbol{x}$满足$\mathbf{0}+\boldsymbol{x}=\boldsymbol{x}$。向量$\mathbf{0}$称为零向量或向量空间中的原点。

5. 对于$\mathcal{X}$中的每一个$\boldsymbol{x}$，存在一个向量$\boldsymbol{y}=-\boldsymbol{x}$属于$\mathcal{X}$，使得$\boldsymbol{x}+\boldsymbol{y}=\mathbf{0}$。

6. 对于$\mathcal{F}$中的每一个$\alpha$和$\mathcal{X}$中的每一个$\boldsymbol{x}$，相应地向量$\alpha\boldsymbol{x}$也属于$\mathcal{X}$，称作$\alpha$和$\boldsymbol{x}$的数乘。

7. 数乘满足结合律，即对于$\mathcal{F}$中的任意$\alpha$，$\beta$和$\mathcal{X}$中的任意$\boldsymbol{x}$，有$\alpha(\beta\boldsymbol{x})=(\alpha\beta)\boldsymbol{x}$。

8. 数乘对于向量加法满足分配律，即对于$\mathcal{F}$中任意$\alpha$和$\mathcal{X}$中的任意$\boldsymbol{x}_1$，$\boldsymbol{x}_2$，有$\alpha(\boldsymbol{x}_1+\boldsymbol{x}_2)=\alpha\boldsymbol{x}_1+\alpha\boldsymbol{x}_2$。

9. 数乘对于标量加法满足分配律，即对于$\mathcal{F}$中的任意$\alpha$，$\beta$和$\mathcal{X}$中的任意$\boldsymbol{x}$，有$(\alpha+\beta)\boldsymbol{x}=\alpha\boldsymbol{x}+\beta\boldsymbol{x}$。

10. 对于$\mathcal{X}$中的任意$\boldsymbol{x}$，有$1\boldsymbol{x}=\boldsymbol{x}$，其中1是$\mathcal{F}$中的单位元素1。

一般而言，一个域形成本身上一个向量空间。也就是说向量集中的某向量所包含的元素就是域中的元素；如果向量空间所定义（的数乘和向量加法）的域是同一个域，那么它满足
552 定义A.2中的10个条件。$(\Re,\Re)$和$(\mathscr{C},\mathscr{C})$是两个非常重要的向量空间。很显然，$(\mathscr{C},\Re)$是一个向量空间，但是$(\Re,\mathscr{C})$不是，因为不产生数乘，通常，向量的元素是实数域。$(\Re(s),\Re(s))$和$(\Re(s),\Re)$也是向量空间，其中$\Re(s)$表示包含实系数和独立变量$s$的有理函数域。但是，$(\Re,\Re(s))$不是向量空间。同样，定义在区间$(-\infty,\infty)$实值分段连续函数形成实数域的向量空间。加法和乘法按照通常方式定义。这种向量空间特称为*函数空间*。

我们一般定义向量空间的维数，即向量的长度（维数）。例如，$(\Re^n,\Re)$表示实数域上的一个向量空间，向量的长度为$n$。这样，这个向量空间就是一个$n$*维实向量空间*。$(\mathscr{C}^n,\mathscr{C})$是$n$*维复向量空间*，而$(\Re^n(s),\Re(s))$是$n$*维有理向量空间*。在这本书中，用$\mathcal{F}^n$表示$\mathcal{F}$域上的$n$维向量

空间，以简化向量空间的标示。我们还将区分向量空间的列向量和行向量。例如，$\Re^{n\times 1}$是一个$n$维实向量空间的列向量，$\Re^{1\times n}$是一个$n$维实向量空间的行向量。

另一个重要的向量空间基于多项式集合$\Re_n\{s\}$，即$s$的次数不大于$n$的（实系数）多项式。$(\Re_n\{s\}, \Re)$是一个向量空间，其中向量加法定义为：

$$\sum_{i=0}^{n}\alpha_i s^i + \sum_{i=0}^{n}\beta_i s^i = \sum_{i=0}^{n}(\alpha_i+\beta_i)s^i$$

数乘定义为：

$$\gamma\left(\sum_{i=0}^{n}\alpha_i s^i\right) = \sum_{i=0}^{n}\gamma\alpha_i s^i$$

其中$\alpha_i, \beta_i, \gamma \in \Re$。

**定义A.3** 假设$(\mathscr{X}, \mathscr{F})$是一个向量空间，$\mathscr{Y}$是$\mathscr{X}$的子集，那么在向量空间$(\mathscr{X}, \mathscr{F})$的运算下，$(\mathscr{Y}, \mathscr{F})$是$(\mathscr{X}, \mathscr{F})$的一个子空间，$\mathscr{Y}$形成域$\mathscr{F}$的一个向量空间。

### A.2.2 矩阵的表示和运算

*矩阵和向量*

一个矩阵是一组元素，一般元素就是数值。然而，矩阵的元素也可以是函数。大多数情况下，矩阵是长方形的。特殊情况下，可能为方阵、向量（行或列向量）以及标量。假设$\boldsymbol{A} \in \Re^{n\times m}$表示所有的长方形矩阵为实（数）元素，$n\times m$维，其中$n$为矩阵的行数，$m$为矩阵的列数，即 553

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}$$

这个矩阵的维数简单地记做$n\times m$。$\boldsymbol{A} \in \mathscr{C}^{n\times m}$表示所有矩形的复数矩阵。矩阵$\boldsymbol{A}$也可以记做$\boldsymbol{A} = [a_{ij}]_{n\times m}$。这样，$a_{ij}$表示位于矩阵第$i$行第$j$列的元素。$\boldsymbol{A} \in \Re^{n\times m}$在特殊情况下，包含如下几种情况：(1) 如果$m = n$，$\boldsymbol{A} \in \Re^{n\times n}$表示所有的实数方阵；(2) 如果$m = 1$，$\boldsymbol{x} \in \Re^{n\times 1}$表示所有$n$个实数的列向量；(3) $\boldsymbol{y} \in \Re^{1\times n}$表示所有的$n$个实数的行向量。如果定义矩阵$\boldsymbol{A} \in \Re^{n\times m}$的列为$\boldsymbol{a}_j$，其中$j = 1, 2, \cdots, m$，矩阵$\boldsymbol{A}$可以记成$\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_m]$。

*矩阵（向量）加法和减法*

若$\boldsymbol{A} \in \Re^{n\times m}$，$\boldsymbol{B} \in \Re^{n\times m}$，那么$\boldsymbol{C} = \boldsymbol{A} \pm \boldsymbol{B}$，其中$\boldsymbol{C} \in \Re^{n\times m}$。这样，两个矩阵必须有相同的行数和列数才能进行加法或减法运算。如果$\boldsymbol{x} \in \Re^{n\times 1}$，$\boldsymbol{y} \in \Re^{n\times 1}$，那么两个列向量相加或相减就会有：$z = \boldsymbol{x} \pm \boldsymbol{y}$，其中$z \in \Re^{n\times 1}$。对于两个相同长度的行向量，它们的和以及差的结果是显然的。

*矩阵乘法*

若$\boldsymbol{A} \in \Re^{n\times m}$，其元素记作$a_{ij}$，$\boldsymbol{B} \in \Re^{m\times p}$，其元素记作$b_{jk}$，$\boldsymbol{A}$乘以$\boldsymbol{B}$，有：$\boldsymbol{C} = \boldsymbol{AB} \in \Re^{n\times p}$，元素记作$c_{ik}$ $(\Re^{n\times p} \leftarrow \Re^{n\times m} \times \Re^{m\times p})$。$\boldsymbol{C}$中的每一个元素可以写作

$$c_{ik} = \sum_{j=1}^{m} a_{ij}b_{jk} \quad \text{或} \quad \boldsymbol{AB} = \left[\sum_{j=1}^{m} a_{ij}b_{jk}\right]_{n\times p}$$

从这个结论中可以看出只有当矩阵$\boldsymbol{B}$的行数和矩阵$\boldsymbol{A}$的列数相同时，矩阵才能相乘。这样，$\boldsymbol{BA}$没有意义，因为$\boldsymbol{B}$有$p$列而$\boldsymbol{A}$有$n$行，很明显不匹配。因此，矩阵乘法不具可交换性。但是，矩阵乘法满足结合律和分配律。矩阵乘法的特殊情况就是一个矩阵与一个向量相乘，即，对于

554 $\boldsymbol{A} \in \Re^{n\times m}$，$\boldsymbol{x} \in \Re^{m\times 1}$，$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} \in \Re^{n\times 1}$具有元素$b_i$ ($\Re^{n\times 1} \leftarrow \Re^{n\times m} \times \Re^{m\times 1}$)。

**转置矩阵**

矩阵$\boldsymbol{A} \in \Re^{n\times m}$的转置记作$\boldsymbol{A}^T$。要转置矩阵$\boldsymbol{A} = [a_{ij}]_{n\times m}$，交换原矩阵的行与列，即

$$\boldsymbol{A}^T = [a_{ij}]_{n\times m}^T = [a_{ji}]_{m\times n} \in \Re^{m\times n} \tag{A-1}$$

以下是转置矩阵的一些重要属性：

1. $(\boldsymbol{A}^T)^T = \boldsymbol{A}$
2. 若$\boldsymbol{A} \in \Re^{n\times m}$, $\boldsymbol{B} \in \Re^{m\times p}$, $(\boldsymbol{A}\boldsymbol{B})^T = \boldsymbol{B}^T\boldsymbol{A}^T \in \Re^{p\times n}$
3. $(\boldsymbol{A} + \boldsymbol{B})^T = \boldsymbol{A}^T + \boldsymbol{B}^T$

**对角矩阵、对称矩阵和单位矩阵**

对角矩阵是只有对角元素的方阵，即，（除了对角元素外）其他元素均为零。例如，若$\boldsymbol{A} \in \Re^{n\times n}$为对角矩阵，可以记作

$$\boldsymbol{A} = \text{diag}[a_{11}, a_{22}, \cdots, a_{nn}]$$

其中

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ 0 & \cdots & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{bmatrix}$$

如果$\boldsymbol{A}^T = \boldsymbol{A}$，那么$\boldsymbol{A}$称作对称矩阵。显然对角矩阵是对称的。如果对角矩阵对角线上的元素都是单位元素，那么该矩阵称作单位矩阵。例如，一个$n \times n$单位矩阵$\boldsymbol{I}_n$对角线上有$n$个单位元素，其他非对角线上的元素为零，即

$$\boldsymbol{I}_n = \begin{bmatrix} 1 & & \boldsymbol{0} \\ & 1 & \\ & \ddots & \\ \boldsymbol{0} & & 1 \end{bmatrix} = \text{diag}[1, 1, \cdots, 1] = [\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_n] \tag{A-2}$$

其中$\boldsymbol{I}_n$的第$j$列为$\boldsymbol{e}_j$；例如，如果$j = 2$，那么$\boldsymbol{e}_2 = [0, 1, 0, \cdots, 0]^T$。

### A.2.3 内积和外积

假设两个$n$维列向量$\boldsymbol{x} \in \Re^{n\times 1}$, $\boldsymbol{y} \in \Re^{n\times 1}$。这两个向量的内积为

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^T\boldsymbol{y} = \boldsymbol{y}^T\boldsymbol{x} = \langle \boldsymbol{y}, \boldsymbol{x} \rangle = \sum_{i=1}^{n} y_i x_i \tag{A-3}$$

555 如果$\boldsymbol{x}$和$\boldsymbol{y}$含有复数元素，即$\boldsymbol{x} \in \mathbb{C}^{n\times 1}$, $\boldsymbol{y} \in \mathbb{C}^{n\times 1}$，则$\boldsymbol{x}$与$\boldsymbol{y}$的内积为：

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \bar{\boldsymbol{x}}^T\boldsymbol{y} = \boldsymbol{x}^*\boldsymbol{y} = \sum_{i=1}^{n} \bar{x}_i y_i \tag{A-4}$$

其中 $\boldsymbol{x}^* = \bar{\boldsymbol{x}}^T$ 称为向量$\boldsymbol{x}$的复共轭转置。假设两个$n$维列向量是随时间变化的元素$\boldsymbol{x}(t)$和$\boldsymbol{y}(t)$，这样对于实连续函数的向量空间，在区间$t_1 < t < t_2$上，其$\boldsymbol{x}$与$\boldsymbol{y}$的内积为:

$$\langle \boldsymbol{x}(t), \boldsymbol{y}(t) \rangle = \frac{1}{t_2 - t_1}\int_{t_1}^{t_2} \boldsymbol{x}^T(t)\boldsymbol{y}(t)\mathrm{d}t = \frac{1}{t_2 - t_1}\int_{t_1}^{t_2}\left[\sum_{i=1}^{n} x_i(t)y_i(t)\right]\mathrm{d}t \tag{A-5}$$

两个向量$\boldsymbol{x} \in \Re^{n\times 1}$和$\boldsymbol{y} \in \Re^{n\times 1}$的外积产生一个秩为1的$n \times n$维矩阵（见A.2.5节），即

$$\boldsymbol{A}=\boldsymbol{x}\boldsymbol{y}^T=\begin{bmatrix}x_1\\x_2\\\vdots\\x_n\end{bmatrix}\begin{bmatrix}y_1 & y_2 & \cdots & y_n\end{bmatrix}=\begin{bmatrix}x_1y_1 & x_1y_2 & \cdots & x_1y_n\\x_2y_1 & x_2y_2 & \cdots & x_2y_n\\\cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots\\x_ny_1 & x_ny_2 & \cdots & x_ny_n\end{bmatrix} \tag{A-6}$$

### A.2.4 向量的线性无关

**定义A.4** 假设$\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_m$是$\Re^{n\times 1}$中向量的集合，$\alpha_1, \alpha_2, \cdots, \alpha_m$是$\Re$中的数集。由向量和数形成一个线性组合：

$$\boldsymbol{b}=\alpha_1\boldsymbol{a}_1+\alpha_2\boldsymbol{a}_2+\cdots+\alpha_m\boldsymbol{a}_m=\sum_{i=1}^{m}\alpha_i\boldsymbol{a}_i$$

如果线性组合等于零，即$\boldsymbol{b}=\boldsymbol{0}$，只有对于每一个$\alpha_i=0$ $(i=1, 2, \cdots, m)$，向量集合$\{\boldsymbol{a}_i\}$才是线性无关的。然而，如果存在一个$\alpha_i\neq 0 \ni \boldsymbol{b}=\boldsymbol{0}$，那么向量集合$\{\boldsymbol{a}_i\}$称作线性相关。

**定义A.4.1** 假设$\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_m$是$\Re^{n\times 1}$中的向量集合，$\alpha_1, \alpha_2, \cdots, \alpha_m$是$\Re$中的数集。$\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_m\in\Re^{n\times 1}$的所有线性组合构成的集合叫做由$\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_m$生成的子空间，记作

$$\text{span}\{\boldsymbol{a}_1,\boldsymbol{a}_2,\cdots,\boldsymbol{a}_m\}\triangleq\{\boldsymbol{a}=\alpha_1\boldsymbol{a}_1+\alpha_2\boldsymbol{a}_2+\cdots+\alpha_m\boldsymbol{a}_m:\ \alpha_i\in\Re\}_{1\leqslant i\leqslant m}$$

如果由向量集合$\{\boldsymbol{a}_i\}$形成$n\times m$的矩阵，即$\boldsymbol{A}=[\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_m]\in\Re^{n\times m}$，那么当且仅当$\boldsymbol{A}^T\boldsymbol{A}$是非奇异矩阵时（参照A.2.7节），或等价地$\boldsymbol{A}^T\boldsymbol{A}$是满秩的（参照A.2.5节），矩阵$\boldsymbol{A}$的列线性无关。$\boldsymbol{A}\in\Re^{n\times m}$的行是线性无关的，当且仅当$\boldsymbol{A}\boldsymbol{A}^T$是非奇异的（或$\boldsymbol{A}\boldsymbol{A}^T$是满秩的）。 556

### A.2.5 矩阵的秩和线性无关

**定义A.5** 矩阵$\boldsymbol{A}$的秩是指最大线性无关列数，或者是最大线性无关行数。矩阵的秩用$\rho(\boldsymbol{A})$表示。如果$\rho(\boldsymbol{A})=\min\{n, m\}$，则矩阵$\boldsymbol{A}\in\Re^{n\times m}$满秩。矩阵$\boldsymbol{A}$的秩也可以定义为包含在矩阵$\boldsymbol{A}$中的最大非奇异子矩阵（方阵）的维数（见A.2.7节）。

假设$\boldsymbol{A}\in\Re^{n\times m}$，$\min\{n, m\}=m$，且$\rho(\boldsymbol{A})<m$，那么称矩阵$\boldsymbol{A}$为秩亏损。矩阵的秩有时称作矩阵的本质维数。下面列出矩阵$\boldsymbol{A}\in\Re^{n\times m}$的一些重要特性：

1. $\rho(\boldsymbol{A}^T)=\rho(\boldsymbol{A})$
2. $\rho(\boldsymbol{A}^T\boldsymbol{A})=\rho(\boldsymbol{A})$
3. $\rho(\boldsymbol{A}\boldsymbol{A}^T)=\rho(\boldsymbol{A})$
4. 若$\boldsymbol{A}\in\Re^{n\times m}$, $\boldsymbol{B}\in\Re^{m\times p}$，则$\rho(\boldsymbol{A})+\rho(\boldsymbol{B})\leqslant m+p(\boldsymbol{A}\boldsymbol{B})$
5. 若$\boldsymbol{A}\in\Re^{n\times m}$, $\boldsymbol{B}\in\Re^{m\times p}$，则$\rho(\boldsymbol{A})+\rho(\boldsymbol{B})-m\leqslant p(\boldsymbol{A}\boldsymbol{B})\leqslant\min\{\rho(\boldsymbol{A}), \rho(\boldsymbol{B})\}$（西尔维斯特不等式）

### A.2.6 矩阵的确定性

**定义A.6** 一个对称矩阵$\boldsymbol{A}\in\Re^{n\times n}$，如果$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}>0$，$\forall\boldsymbol{x}\in\Re^{n\times 1}$（除了$\boldsymbol{x}=\boldsymbol{0}$），则称$\boldsymbol{A}$为正定矩阵。若$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}<0$，则称$\boldsymbol{A}$为负定矩阵。若$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}\geqslant 0$，则称$\boldsymbol{A}$为半正定矩阵（或非负定）。若$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}\leqslant 0$，则称$\boldsymbol{A}$为半负定矩阵（或非正定）。

对于对称矩阵$\boldsymbol{A}\in\Re^{n\times n}$（即，$\boldsymbol{A}^T=\boldsymbol{A}$），也可以说如果$\boldsymbol{A}$是正定的，那么$\boldsymbol{A}$的特征值（参照A.2.9节）都是正实数。如果$\boldsymbol{A}$是负定的，那么$\boldsymbol{A}$的特征值全是负实数。如果$\boldsymbol{A}$是半正定的，$\boldsymbol{A}$的某些特征值可以为零（但不全是零），其余的必须为正实数。如果$\boldsymbol{A}$是半负定的，$\boldsymbol{A}$的某些特征值可以为零（但不全是零），其余的必须为负实数。如果一个对称矩阵$\boldsymbol{A}\in\Re^{n\times n}$既有正特征值，

又有负特征值，那么这个矩阵$A$是不确定的。对于任意矩阵$A$，$A^TA$和$AA^T$是半正定矩阵。

若$A \in \Re^{n\times n}$为对称矩阵，简单记作：

1. $A>0$，$A$正定。
2. $A<0$，$A$负定。
3. $A\geqslant 0$，$A$半正定（非负定）。
4. $A\leqslant 0$，$A$半负定（非正定）。

### A.2.7 矩阵的逆和伪逆

*矩阵的逆*

557 假设矩阵$A \in \Re^{n\times n}$，$\rho(A)=n$，那么$A$有逆存在，或者说$A$非奇异，或者说$A$的列（行）是线性无关的。$A$的逆记作$A^{-1}$，且$AA^{-1}=A^{-1}A=I_n$。若$\rho(A)<n$，则$A$是秩亏损的，$A$称作奇异的。正如我们所看到的，矩阵的逆与A.2.1节讨论域的定义中数的倒数（乘法逆）有着相似记号，是其推广。

**定义A.7** 非奇异矩阵$A \in \Re^{n\times n}$的逆可以表示成：

$$A^{-1}=\frac{\text{adj}(A)}{|A|}=\frac{[\text{cof}(A)]^T}{|A|} \tag{A-7}$$

其中adj $\triangleq$伴随，cof $\triangleq$余因子，$|A|$为$A$的行列式（下面将做解释）。矩阵$A \in \Re^{n\times n}$的行列式也可以记作$\det(A)$。若$|A|=0$，则$A$是奇异的（即，$A$的逆不存在）。也可以说如果$A$的行（或列）是线性相关的，或者说$A$至少有一个零特征值，那么$A$是秩亏损的，即$\rho(A)<n$。

下面是关于矩阵的逆的一些重要特性：假设$A \in \Re^{n\times n}$，$B \in \Re^{n\times n}$，$C \in \Re^{m\times m}$，$u \in \Re^{n\times 1}$，$v \in \Re^{n\times 1}$：

1. $(A^{-1})^{-1}=A$
2. $(AB)^{-1}=B^{-1}A^{-1}$
3. $(A^T)^{-1}=(A^{-1})^T=A^{-T}$
4. $(A+uv^T)^{-1}=A^{-1}-\dfrac{(A^{-1}u)(v^TA^{-1})}{1+v^TA^{-1}u}$
5. $(C+DBE)^{-1}=C^{-1}-C^{-1}D(EC^{-1}D+B^{-1})^{-1}EC^{-1}$
6. $(C-DB^{-1}E)^{-1}=C^{-1}+C^{-1}D(B-EC^{-1}D)^{-1}EC^{-1}$

矩阵$A \in \Re^{n\times n}$的行列式在很多领域都有重要的作用。我们从上面看了怎样用矩阵的行列式定义矩阵的逆。一个$2\times 2$的矩阵$A \in \Re^{2\times 2}$的行列式可以这样计算：

$$|A|=\begin{vmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{vmatrix}=a_{11}a_{22}-a_{12}a_{21}$$

$3\times 3$的矩阵$A \in \Re^{3\times 3}$的行列式可以这样计算：

$$|A|=\begin{vmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{vmatrix}=a_{11}\begin{vmatrix}a_{22} & a_{23}\\ a_{32} & a_{33}\end{vmatrix}-a_{12}\begin{vmatrix}a_{21} & a_{23}\\ a_{31} & a_{33}\end{vmatrix}+a_{13}\begin{vmatrix}a_{21} & a_{22}\\ a_{31} & a_{32}\end{vmatrix}$$

假设$A \in \Re^{n\times n}$，$B \in \Re^{n\times n}$，如下是行列式的一些性质：

1. 若$A$的任意一行（或一列）的全部元素为零，那么$|A|=0$。
2. $|A^T|=|A|$

558 3. $|AB|=|BA|=|A||B|$

4. $$\begin{vmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ a_{21} & a_{22} & 0 & \cdots & 0 \\ a_{31} & a_{32} & a_{33} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{vmatrix} = \begin{vmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & a_{2n} \\ 0 & 0 & a_{33} & \cdots & a_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & a_{nn} \end{vmatrix} = a_{11}a_{22}a_{33}\cdots a_{nn} = \prod_{i=1}^{n} a_{ii}$$

5. 若矩阵$A$任意交换两行或两列得到矩阵$B$，则$|A| = -|B|$。

6. 若矩阵$A$的一行或列的每个元素乘以数$k \in \Re$得到矩阵$B$，则$|A| = \frac{1}{k}|B|$。

7. $|kA| = k^n|A|$。

8. 若矩阵$A$的某一行（或列）的常数倍加到另一行（或列）得到矩阵$B$，则$|A| = |B|$。

9. 若$A$的两行（或列）相等，则$|A| = 0$［即，如果$A$的行或列线性相关，或$\rho(A) < n$，那么$|A| = 0$］。

10. 若$\lambda_1, \lambda_2, \cdots, \lambda_n$是$A \in \Re^{n\times n}$的特征值（参照A.2.9节），则$|A| = \prod_{i=1}^{n}\lambda_i$。这样，如果$A$的特征值有零，那么$|A| = 0$，$A$是奇异的。

11. 若$\rho(A) = n$（或所有的列或行线性无关，或$|A| \neq 0$，或$A$非奇异，或$A$的逆存在），则$|A^{-1}| = 1/|A|$。

12. 假设$A \in \Re^{n\times n}$, $B \in \Re^{n\times m}$, $C \in \Re^{m\times n}$, $D \in \Re^{m\times m}$，如果$A$和$D$是可逆的，那么$\det(A)\det(D - CA^{-1}B) = \det(D)\det(A - BD^{-1}C)$。

*矩阵伪逆*

对于一个奇异方阵或矩形矩阵，可以计算它的广义逆。这对于解形如$Ax = b$（其中$A \in \Re^{m\times n}$, $x \in \Re^{n\times 1}$, $b \in \Re^{m\times 1}$）的联立线性代数方程组很有用。$A$的穆尔-彭罗斯（Moore-Penrose）广义逆（或伪逆）记作$A^+$，有以下性质：

1. $A^+ = (A^TA)^{-1}A^T$
2. $A^+AA^+ = A^+$
3. $AA^+A = A$
4. $(AA^+)^T = AA^+$
5. $(A^+A)^T = A^+A$

若$m = n$，则$A$是方阵；若$\rho(A) = n$，则$A^+ = A^{-1}$。然而，若假设$m > n$（超定情况），定义一个误差向量$e \in \Re^{m\times 1}$，使$e = Ax - b$，同样定义一个误差函数如下：

$$\mathscr{E}(x) = \frac{1}{2}\|e\|_2^2 = \frac{1}{2}\|Ax - b\|_2^2 = \frac{1}{2}(Ax - b)^T(Ax - b) \tag{A-8}$$

对$x$最小化$\mathscr{E}(x)$（即，$\partial\mathscr{E}(x)/\partial x = 0$）可以得到*正规方程组*：

$$A^TAx - A^Tb = 0 \tag{A-9}$$

559

求解正规方程组，若$A^TA$是非奇异的［即，$\rho(A) = n$］，得到$Ax = b$的*最小二乘解*$x^*$

$$x^* = (A^TA)^{-1}A^Tb \tag{A-10}$$

定义伪逆矩阵，即穆尔-彭罗斯广义逆矩阵为$A^+ = (A^TA)^{-1}A^T$，因此，$x^* = A^+b$。若$m < n$（欠定的情况），则伪逆矩阵为$A^+ = A^T(AA^T)^{-1}$，其中$AA^T$是非奇异［即，$\rho(A) = m$］。下面是与伪逆矩阵相关的重要性质：

1. 若$\alpha \neq 0$, $(\alpha A)^+ = \alpha^{-1}A^+$
2. $(A^+)^+ = A$

3. $(\boldsymbol{A}^+)^T = (\boldsymbol{A}^T)^+$
4. $\boldsymbol{A}^+ = (\boldsymbol{A}^T\boldsymbol{A})^+\boldsymbol{A}^T = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^+$
5. $\boldsymbol{A}\boldsymbol{A}^T(\boldsymbol{A}^+)^T = \boldsymbol{A}$
6. $\boldsymbol{A}^+\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{A}^T$
7. $(\boldsymbol{A}^+)^T\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{A}$
8. $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{A}^+ = \boldsymbol{A}^T$
9. $\rho(\boldsymbol{A}^+) = \rho(\boldsymbol{A}) = \rho(\boldsymbol{A}^T)$

以这种方式定义矩阵的伪逆的问题之一是矩阵必须是满秩的，即，当$m \geqslant n$, $\rho(\boldsymbol{A}) = n$。当$m < n$, $\rho(\boldsymbol{A}) = m$。这个问题可以通过定义伪逆矩阵的奇异值分解（SVD）来解决（见A.2.14节）。

### A.2.8 正交矩阵、酉矩阵和共轭向量

*正交矩阵和酉矩阵*

假设非零向量集合$\{\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_n\}$, $\boldsymbol{q}_i \in \Re^{n\times 1}$, $i = 1, 2, \cdots, n$。如果向量集合$\{\boldsymbol{q}_i\}$的每一个向量满足$\boldsymbol{q}_i^T\boldsymbol{q}_j = 0 (i \neq j)$，则是正交的。如果向量集是正交的，那么

$$\boldsymbol{q}_i^T\boldsymbol{q}_j = \begin{cases} 0 & \text{对} \quad i \neq j \\ 1 & \text{对} \quad i = j \end{cases} = \delta_{ij} \tag{A-11}$$

其中$\delta_{ij}$是克罗内克记号Δ。若定义一个方阵$\boldsymbol{Q} \triangleq \{\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_n\}$，则$\boldsymbol{Q} \in \Re^{n\times n}$称为正交矩阵，$\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{I}_n$。复数构成的正交矩阵称作酉矩阵。因此，若$\boldsymbol{Q} \in \mathscr{C}^{n\times n}$是酉矩阵，则$\boldsymbol{Q}^*\boldsymbol{Q} = \boldsymbol{I}_n$，其中$\boldsymbol{Q}^*$是$\boldsymbol{Q}$的复数共轭转置矩阵，即$\boldsymbol{Q}^* = \overline{\boldsymbol{Q}}^T$。$\boldsymbol{Q}$的复数共轭转置矩阵也可写作$\boldsymbol{Q}^H$，即$\boldsymbol{Q}$的埃尔米特（Hermitian）转置。若$\boldsymbol{Q}\boldsymbol{Q}^H = \boldsymbol{Q}^H\boldsymbol{Q}$[1]，则矩阵$\boldsymbol{Q} \in \mathscr{C}^{n\times n}$是正规的。

正交矩阵的一个重要属性是对内积无影响。例如，假设$\boldsymbol{Q} \in \Re^{n\times n}$是正交矩阵，$\boldsymbol{x}, \boldsymbol{y} \in \Re^{n\times 1}$，
560 那么

$$\langle \boldsymbol{Q}\boldsymbol{x}, \boldsymbol{Q}\boldsymbol{y} \rangle = (\boldsymbol{Q}\boldsymbol{x})^T\boldsymbol{Q}\boldsymbol{y} = \boldsymbol{x}^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{y} = \boldsymbol{x}^T\boldsymbol{y} = \langle \boldsymbol{x}, \boldsymbol{y} \rangle$$

以及

$$\|\boldsymbol{Q}\boldsymbol{x}\|_2 = \left[(\boldsymbol{Q}\boldsymbol{x})^T\boldsymbol{Q}\boldsymbol{x}\right]^{1/2} = (\boldsymbol{x}^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{x})^{1/2} = (\boldsymbol{x}^T\boldsymbol{x})^{1/2} = \|\boldsymbol{x}\|_2 = \langle \boldsymbol{x}, \boldsymbol{x} \rangle^{1/2}$$

*共轭向量*

**定义A.8** 假设$\boldsymbol{Q} \in \Re^{n\times n}$是对称矩阵，两个向量$\boldsymbol{d}_1 \in \Re^{n\times 1}$和$\boldsymbol{d}_2 \in \Re^{n\times 1}$，若$\boldsymbol{d}_1^T\boldsymbol{Q}\boldsymbol{d}_2 = 0$，那么$\boldsymbol{d}_1$和$\boldsymbol{d}_2$是关于矩阵$\boldsymbol{Q}$（或$\boldsymbol{Q}$正交）共轭的。

.（非零）向量集合$\{\boldsymbol{d}_0, \boldsymbol{d}_1, \cdots, \boldsymbol{d}_{n-1}\}$, $\boldsymbol{d}_i \in \Re^{n\times 1}$, $i = 1, 2, \cdots, n$，若

$$\boldsymbol{d}_i^T\boldsymbol{Q}\boldsymbol{d}_j = 0 \quad \text{其中} i \neq j \tag{A-12}$$

称为$\boldsymbol{Q}$正交集。

从定义可以看出，若$\boldsymbol{Q} = \boldsymbol{I}_n$，则共轭的概念相当于正交的意思。既然这样，若$\boldsymbol{Q} \in \Re^{n\times n}$是正定的（$\boldsymbol{Q} > 0$），非零向量集$\{\boldsymbol{d}_0, \boldsymbol{d}_1, \cdots, \boldsymbol{d}_{n-1}\}$是$\boldsymbol{Q}$正交，那么这些向量是线性无关的。

### A.2.9 特征值和特征向量

假设矩阵$\boldsymbol{A} \in \Re^{n\times n}$，对一个标量$\lambda$和一个非零向量$\boldsymbol{v}$，若

$$\boldsymbol{A}\boldsymbol{v} = \lambda\boldsymbol{v} \tag{A-13}$$

则$\lambda$是$\boldsymbol{A}$的一个特征值，$\boldsymbol{v}$是对应的特征向量。对于$\boldsymbol{A}$的所有特征值和特征向量，标准的特征值问题如下：

$$(\lambda_i \boldsymbol{I} - \boldsymbol{A})\boldsymbol{v}_i = \boldsymbol{0} \quad 或 \quad (\boldsymbol{A} - \lambda_i \boldsymbol{I})\boldsymbol{v}_i = \boldsymbol{0} \quad 其中 i = 1, 2, \cdots, n \tag{A-14}$$

当且仅当

$$|\lambda \boldsymbol{I} - \boldsymbol{A}| = 0 \tag{A-15}$$

时，这个方程组有一个解，上式称为$\boldsymbol{A}$的特征方程。

多项式$|\lambda \boldsymbol{I} - \boldsymbol{A}|$的根就是特征值；即$\{\lambda_i\}$, $i = 1, 2, \cdots, n$，倘若特征值是不同的，对于每一个特征值和其相应的特征向量满足$(\lambda_i \boldsymbol{I} - \boldsymbol{A})\upsilon_i = \boldsymbol{0}$。$\boldsymbol{A}$的非零特征值对应的特征向量总是线性无关的。

一般，特征值可以相同也可以不同，既可以是实数也可以是复数。然而，由于$\boldsymbol{A}$是实矩阵，若有一个复数特征值，那么一定存在复数共轭对。矩阵特征值的集合有时也称为矩阵的谱，记作$\sigma(\boldsymbol{A})$；一个特定的特征值记作$\lambda(\boldsymbol{A})$，即$\lambda_i(\boldsymbol{A})$, $i = 1, 2, \cdots, n$。对于$\boldsymbol{A}$的不同特征值，如果构造一个矩阵$\boldsymbol{V} = [\upsilon_1, \upsilon_2, \cdots, \upsilon_n] \in \Re^{n \times n}$，其中的列为特征向量，那么，若构造 561

$$\boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} = \boldsymbol{\Lambda} \tag{A-16}$$

矩阵$\boldsymbol{\Lambda} = \mathrm{diag}[\lambda_1, \lambda_2, \cdots, \lambda_n]$，$\boldsymbol{A}$的特征值在对角线上，称$\boldsymbol{A}$被对角化。矩阵$\boldsymbol{V}$称作相似矩阵变换（参照A.2.10节）。对于非奇异矩阵$\boldsymbol{A} \in \Re^{n \times n}$，即$\rho(\boldsymbol{A}) = n$，所有特征值都是非零的，从式（A-16）可以很容易看出$\boldsymbol{A}^{-1}$的特征值是$\boldsymbol{A}$的特征值的倒数，即$\boldsymbol{\Lambda}^{-1} = \mathrm{diag}[1/\lambda_1, 1/\lambda_2, \cdots, 1/\lambda_n]$。

对于含有相同特征值的矩阵$\boldsymbol{A} \in \Re^{n \times n}$，$\boldsymbol{A}$不一定是可对角化的。为了简单又不失一般性，假定$\boldsymbol{A}$只有一组相同的特征值。$m$为特征值的重数，则有$m \leqslant n$，并且，若$m < n$，那么剩余的$n - m$个特征值是不同的。$\boldsymbol{A}$的零化度即$v(\boldsymbol{A})$表示为，

$$v(\boldsymbol{A}) = n - \rho(\boldsymbol{A}) \tag{A-17}$$

假设$\boldsymbol{A}$的一个特征值有$m$重，即$\lambda_k (k = 1, 2, \cdots, m)$，或$\lambda_1 = \lambda_2 = \cdots = \lambda_m = \lambda$。若$v(\lambda \boldsymbol{I} - \boldsymbol{A}) = m$，则存在$m$个线性无关的特征向量与这$m$个特征值相关。这有时称作完全退化。与特征值$\lambda_k = \lambda (k = 1, 2, \cdots, m)$相应的$m$个线性无关的特征向量可以取自下列矩阵的非零列

$$\frac{1}{m-1}\left\{\frac{\mathrm{d}^{m-1}}{\mathrm{d}\lambda^{m-1}}[\mathrm{adj}(\lambda \boldsymbol{I} - \boldsymbol{A})]\right\}\Bigg|_{\lambda = \lambda_k}$$

剩余的不同特征值对应的剩余的$n - m$个特征向量取自$(\lambda_j \boldsymbol{I} - \boldsymbol{A})\upsilon_j = \boldsymbol{0} (j = m + 1, m + 2, \cdots, n)$。可以用$n$个线性无关的特征向量的集合构成相似转换矩阵$\boldsymbol{V}$（前$m$列对应于$m$重特征值相应的特征向量，剩余$n - m$列对应不同的特征值相应的特征向量）。使用转换矩阵$\boldsymbol{V}$可得到：

$$\boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} = \mathrm{diag}[\lambda, \lambda, \cdots, \lambda, \lambda_{m+1}, \lambda_{m+2}, \cdots, \lambda_n] \tag{A-18}$$

可以看出$\boldsymbol{A}$仍是可对角化的。然而，若$v(\lambda \boldsymbol{I} - \boldsymbol{A}) = 1$（有时称作简单退化），尽管有$m$重，但只有一个特征向量与$\lambda_1 = \lambda_2 = \cdots = \lambda_m = \lambda$对应，可以通过标准表达式$(\lambda \boldsymbol{I} - \boldsymbol{A})\upsilon = \boldsymbol{0}$求得。然而，若有$m - 1$个其他向量与$m$重特征值对应，则称为广义特征向量，所有$m$个向量是线性无关的。这$m - 1$个广义特征向量可由以下求出： 562

$$\begin{aligned}
(\lambda \boldsymbol{I} - \boldsymbol{A})\boldsymbol{v}_2 &= -\boldsymbol{v}_1 \\
(\lambda \boldsymbol{I} - \boldsymbol{A})\boldsymbol{v}_3 &= -\boldsymbol{v}_2 \\
(\lambda \boldsymbol{I} - \boldsymbol{A})\boldsymbol{v}_4 &= -\boldsymbol{v}_3 \\
&\vdots \\
(\lambda \boldsymbol{I} - \boldsymbol{A})\boldsymbol{v}_{m-1} &= -\boldsymbol{v}_{m-2} \\
(\lambda \boldsymbol{I} - \boldsymbol{A})\boldsymbol{v}_m &= -\boldsymbol{v}_{m-1}
\end{aligned} \tag{A-19}$$

其中$v_1$由求解标准表达式$(\lambda \boldsymbol{I}-\boldsymbol{A})v_1=\boldsymbol{0}$求得。剩余的不同特征值对应的剩余的$n-m$个特征向量可由方程$(\lambda_j \boldsymbol{I}-\boldsymbol{A})v_j=\boldsymbol{0}(j=m+1, m+2, \cdots, n)$求得。由$n$个线性无关的特征向量集合可以得到相似变换矩阵$\boldsymbol{V}$（$\boldsymbol{V}$的前$m$列是与$m$重特征值相对应的广义特征向量，剩余$n-m$列是与不同特征值相对应的特征向量）。利用变换矩阵$\boldsymbol{V}$，可以得到:

$$\boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V}=\begin{bmatrix}\lambda & 1 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0\\ 0 & \lambda & 1 & 0 & 0 & 0 & 0 & \cdots & 0\\ 0 & 0 & \lambda & 1 & 0 & 0 & 0 & \cdots & 0\\ & \vdots & & \ddots & 1 & & & \vdots & \\ 0 & 0 & \cdots & 0 & \lambda & 0 & 0 & \cdots & 0\\ 0 & 0 & 0 & \cdots & 0 & \lambda_{m+1} & 0 & \cdots & 0\\ 0 & 0 & 0 & \cdots & 0 & 0 & \lambda_{m+2} & 0 & 0\\ & & \vdots & & & 0 & 0 & \ddots & \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & \lambda_n\end{bmatrix} \tag{A-20}$$

在A.2.11节讨论关于$1<v(\lambda_k \boldsymbol{I}-\boldsymbol{A})<m$的情况，即，当$(\lambda_k \boldsymbol{I}-\boldsymbol{A})$的零度在1（简单退化）和重数$m$（完全退化）之间。

方阵的迹定义为对角元素之和。例如，$\boldsymbol{A}\in\Re^{n\times n}$的迹为：

$$\text{trace}(\boldsymbol{A})=\text{tr}(\boldsymbol{A})=\sum_{i=1}^{n}a_{ii}$$

关于迹的性质总结如下：

1. $\text{trace}(\boldsymbol{A})=\text{trace}(\boldsymbol{A}^T)$

2. 若$\boldsymbol{A}\in\Re^{n\times m}, \boldsymbol{B}\in\Re^{m\times n}$，则

$$\text{trace}(\boldsymbol{AB})=\text{trace}(\boldsymbol{BA})=\text{trace}(\boldsymbol{A}^T\boldsymbol{B}^T)=\text{trace}(\boldsymbol{B}^T\boldsymbol{A}^T)$$
$$\text{trace}(\boldsymbol{AA}^T)=\text{trace}(\boldsymbol{A}^T\boldsymbol{A})$$

3. 若$\boldsymbol{A}, \boldsymbol{B}\in\Re^{n\times n}$，$\alpha, \beta\in\Re$，则

563
$$\text{trace}(\alpha\boldsymbol{A}+\beta\boldsymbol{B})=\alpha\,\text{trace}(\boldsymbol{A})+\beta\,\text{trace}(\boldsymbol{B})$$

4. $\text{trace}(\boldsymbol{A})=\Sigma_{i=1}^{n}a_{ii}=\Sigma_{i=1}^{n}\lambda_i$，其中，$\lambda_i(i=1, 2, \cdots, n)$为$\boldsymbol{A}$的特征值。

若令$\Delta(\lambda)=|\lambda\boldsymbol{I}-\boldsymbol{A}|=\lambda^n+\alpha_1\lambda^{n-1}+\alpha_2\lambda^{n-2}+\cdots+\alpha_{n-1}\lambda+\alpha_n$为矩阵$\boldsymbol{A}\in\Re^{n\times n}$的特征多项式，则$\Delta(\boldsymbol{A})=\boldsymbol{A}^n+\alpha_1\boldsymbol{A}^{n-1}+\alpha_2\boldsymbol{A}^{n-2}+\cdots+\alpha_{n-1}\boldsymbol{A}+\alpha_n\boldsymbol{I}=\boldsymbol{0}$。换句话说，每个方阵满足自己的特征方程。这就是著名的凯莱-哈密顿定理。

特征值和特征向量的一些其他属性:

1. 假设$\boldsymbol{x}$是矩阵$\boldsymbol{A}$对应于特征值$\lambda$的特征向量，同时$\boldsymbol{A}$是可逆的，那么$\boldsymbol{x}$是$\boldsymbol{A}^{-1}$对应于特征值$1/\lambda$的特征向量。

2. 若$\boldsymbol{x}$为$\boldsymbol{A}$的特征向量，则$k\boldsymbol{x}$也为$\boldsymbol{A}$的特征向量（其中$k$是非零常量），其中$\boldsymbol{x}$与$k\boldsymbol{x}$对应同一个特征值。

3. 矩阵和它的转置矩阵有相同的特征值。

4. 上三角和下三角矩阵主对角线上的元素是该矩阵的特征值。

5. 若$\boldsymbol{x}$是矩阵$\boldsymbol{A}$的特征值$\lambda$对应的特征向量，则对于任一标量$\alpha$, $\boldsymbol{x}$是矩阵$\boldsymbol{A}-\alpha\boldsymbol{I}$的特征值$\lambda-\alpha$对应的特征向量。

### A.2.10 相似变换

假设矩阵$\boldsymbol{A} \in \Re^{n\times n}, \boldsymbol{P} \in \Re^{n\times n}$，其中$\rho(\boldsymbol{P}) = n$，$\overline{\boldsymbol{A}} = \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}$，则$\boldsymbol{A}$和$\overline{\boldsymbol{A}}$有相同的特征值（即$\boldsymbol{A}$与$\overline{\boldsymbol{A}}$是相似矩阵），或者等价地，它们有相同的特征方程组，即，$\Delta(\lambda) = |\lambda\boldsymbol{I} - \boldsymbol{A}| = 0$，$\overline{\Delta}(\lambda) = |\lambda\boldsymbol{I} - \overline{\boldsymbol{A}}| = 0$。可由如下证明：

$$\overline{\Delta}(\lambda) = |\lambda\boldsymbol{I} - \overline{\boldsymbol{A}}| = |\lambda\boldsymbol{I} - \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}| = |\boldsymbol{P}^{-1}\lambda\boldsymbol{I}\boldsymbol{P} - \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}| = \boldsymbol{P}^{-1}|\lambda\boldsymbol{I} - \boldsymbol{A}|\boldsymbol{P} = 0$$

且

$\overline{\Delta}(\lambda) = \boldsymbol{P}^{-1}|\lambda\boldsymbol{I} - \boldsymbol{A}|\boldsymbol{P} = 0$ 左乘$\boldsymbol{P}$，右乘$\boldsymbol{P}^{-1}$：

$$\underbrace{\boldsymbol{P}\boldsymbol{P}^{-1}}_{\boldsymbol{I}}|\lambda\boldsymbol{I} - \boldsymbol{A}|\underbrace{\boldsymbol{P}\boldsymbol{P}^{-1}}_{\boldsymbol{I}} = 0 \quad \Rightarrow \quad \Delta(\lambda) = |\lambda\boldsymbol{I} - \boldsymbol{A}| = 0 \quad \Rightarrow \quad \overline{\Delta}(\lambda) = \Delta(\lambda)$$

在A.2.9节介绍了一个非常重要的相似变换，即$\boldsymbol{V} = [\upsilon_1, \upsilon_2, \cdots, \upsilon_n]$，其中$\upsilon_i (i = 1, 2, \cdots, n)$是当$\boldsymbol{A} \in \Re^{n\times n}$有不同的特征值时的特征向量。在式（A-16）中$\boldsymbol{\Lambda} = \mathrm{diag}[\lambda_1, \lambda_2, \cdots, \lambda_n]$体现了这个结论，相似变换即对角化$\boldsymbol{A}$。若$\boldsymbol{A} \in \Re^{n\times n}$是对称的，$\boldsymbol{A}^T = \boldsymbol{A}$，则相似变换矩阵$\boldsymbol{P}$有性质$\boldsymbol{P}^{-1} = \boldsymbol{P}^T$，当$\boldsymbol{P} = \boldsymbol{V} = [\upsilon_1, \upsilon_2, \cdots, \upsilon_n]$（$\boldsymbol{A}$的特征向量）时，$\boldsymbol{V}^T\boldsymbol{A}\boldsymbol{V} = \boldsymbol{\Lambda} = \mathrm{diag}[\lambda_1, \lambda_2, \cdots, \lambda_n]$。这样，$\boldsymbol{A}$是正规的，$\boldsymbol{V}$是正交的（参照A.2.8节）。同样，也可以用正交相似变换矩阵$\boldsymbol{V}$，把$\boldsymbol{A}$表示成如下形式： 564

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T = \sum_{i=1}^{n}\lambda_i\upsilon_i\upsilon_i^T \tag{A-21}$$

称作$\boldsymbol{A}$的特征值（谱）分解（EVD）。对于对称矩阵$\boldsymbol{A}$，最小和最大特征值分别满足：

$$\lambda_{\min}(\boldsymbol{A}) = \min_{\upsilon\neq 0}\frac{\upsilon^T\boldsymbol{A}\upsilon}{\upsilon^T\upsilon} \quad 和 \quad \lambda_{\max}(\boldsymbol{A}) = \max_{\upsilon\neq 0}\frac{\upsilon^T\boldsymbol{A}\upsilon}{\upsilon^T\upsilon} \tag{A-22}$$

还有其他的相似变换矩阵使$\boldsymbol{A}$转变成特殊的形式。在下一节，我们讨论这样一种变换矩阵，使其变换成若当（Jordan）标准形。

若两个矩阵通过一个相似变换矩阵相关，那它们是相似的。若$\boldsymbol{A}, \boldsymbol{B} \in \Re^{n\times n}$是相似的，则：

1. $|\boldsymbol{A}| = |\boldsymbol{B}|$
2. $\mathrm{trace}(\boldsymbol{A}) = \overline{\left[\mathrm{trace}(\overline{\boldsymbol{A}})\right]}$（其中$\overline{\boldsymbol{A}}$是$\boldsymbol{A}$的复共轭）。
3. $\boldsymbol{A}^k = \boldsymbol{B}^k$是相似的（$k \geqslant 1$）。
4. $(\boldsymbol{A}^k)^T = (\boldsymbol{B}^k)^T$是相似的（$k \geqslant 1$）。
5. $\boldsymbol{A}$是可逆的$\Leftrightarrow\boldsymbol{B}$是可逆的$\Leftrightarrow\boldsymbol{A}^{-k}$和$\boldsymbol{B}^{-k}$是相似的（$k \geqslant 1$）。
6. $\boldsymbol{A}$是对合的，斜对合的，幂等的，三幂等的，零幂的$\Leftrightarrow\boldsymbol{B}$是对合的，斜对合的，幂等的，三幂的，幂零的（参见A.2.18节）。

### A.2.11 若当标准形

每一个矩阵本身映射到$n$维复向量空间，即$\boldsymbol{A}:(\mathscr{C}^n, \mathscr{C}) \rightarrow (\mathscr{C}^n, \mathscr{C})$，有一个若当标准形（或若当形表示法）。式（A-16）中的矩阵$\boldsymbol{\Lambda} = \mathrm{diag}[\lambda_1, \lambda_2, \cdots, \lambda_n]$就是若当标准形，它是由相似变换矩阵$\boldsymbol{V} = [\upsilon_1, \upsilon_2, \cdots, \upsilon_n]$得来的，其中，$\boldsymbol{A} \in \Re^{n\times n}$有$n$个不同的特征值，$\upsilon_i (i = 1, 2, \cdots, n)$是$\boldsymbol{A}$的特征向量。在这种情况下，$\boldsymbol{\Lambda}$有$n$个1阶的若当块。这其实是矩阵若当标准形的一种特例。

若当标准形可以看作是对角矩阵块，就像式（A-20）所表示的那样。在这种情况下，矩阵$\boldsymbol{A} \in \Re^{n\times n}$有一个特征值$\lambda$在$\nu(\lambda\boldsymbol{I} - \boldsymbol{A}) = 1$重复了$m$次（即，零度化为1或简单退化），剩余$n-m$个特征值是各异的。因此，$\boldsymbol{A}$的若当标准形即式（A-20）$\boldsymbol{A}_J = \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V}$，有一个$m$阶的若当

块（第1块）与$m$重的特征值$\lambda$相关；剩余的$n-m$个若当块均为1阶的，对应不同的特征值。在这种情况下，$\nu(\lambda \boldsymbol{I}-\boldsymbol{A})=1$的零度化表明有$m$个若当块与$m$重的特征值$\lambda$相对应。

对于完全退化的情况，即当$\nu(\lambda \boldsymbol{I}-\boldsymbol{A})=m$（重数），式（A-18）的若当标准形$\boldsymbol{A}_J=\boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V}=\operatorname{diag}[\lambda, \lambda, \cdots, \lambda, \lambda_{m+1}, \lambda_{m+2}, \cdots, \lambda_n]$是对角矩阵，即$\boldsymbol{A}$是可对角化的。对于这种情况，零度化$\nu(\lambda \boldsymbol{I}-\boldsymbol{A})=m$表明有$m$个若当块与$m$重特征值$\lambda$相对应。

565

因此，假设矩阵$\boldsymbol{A} \in \Re^{n \times n}$有$m$重特征值$\lambda$，剩余$n-m$个特征值是各异的。$\nu(\lambda \boldsymbol{I}-\boldsymbol{A})=q$的零度化表明与多个特征值相关的若当块的数量。若$q=1$，我们说它是简单退化的，若$q=m$，则称它是完全退化的。当$1<q<m$，则需要更多的信息才能决定若当块的结构[2]。例如，假定$\boldsymbol{A} \in \Re^{4 \times 4}$，特征值为$\lambda_1=\lambda_2=\lambda_3=\lambda_4=\lambda$，$m=4$。若$\nu(\lambda \boldsymbol{I}-\boldsymbol{A})=2$，这表明$\boldsymbol{A}$的若当标准形有两个若当块。这可能是两个2阶若当块，或一个3阶的若当块和一个1阶的若当块，即

$$\boldsymbol{A}_J=\begin{bmatrix}\lambda & 1 & 0 & 0\\ 0 & \lambda & 0 & 0\\ 0 & 0 & \lambda & 1\\ 0 & 0 & 0 & \lambda\end{bmatrix} \tag{A-23}$$

或

$$\boldsymbol{A}_J=\begin{bmatrix}\lambda & 1 & 0 & 0\\ 0 & \lambda & 1 & 0\\ 0 & 0 & \lambda & 0\\ 0 & 0 & 0 & \lambda\end{bmatrix} \tag{A-24}$$

或

$$\boldsymbol{A}_J=\begin{bmatrix}\lambda & 0 & 0 & 0\\ 0 & \lambda & 1 & 0\\ 0 & 0 & \lambda & 1\\ 0 & 0 & 0 & \lambda\end{bmatrix} \tag{A-25}$$

只是把式（A-24）的两个若当块重排序。矩阵$\boldsymbol{A}:(\mathscr{C}^n, \mathscr{C}) \rightarrow (\mathscr{C}^n, \mathscr{C})$的若当标准形是否唯一取决于若当块的阶。如果计算矩阵$\boldsymbol{A}$的广义特征向量，对于$\boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V}=\boldsymbol{A}_J$，若当标准形总是成立的。

在A.2.9节我们看到，有不同特征值的矩阵是可以对角化的。这些矩阵称作非退化的。如果矩阵没有完整的特征向量集称作退化的。因此，对有重复特征值且不能对角化的矩阵，但可以求出它的若当标准形（即，块对角矩阵），这些矩阵称作是退化的。

### A.2.12　动态系统的状态空间描述

动态系统的状态空间模型在控制系统设计分析，信号处理以及其他许多领域都非常重要。

566

对于时间不变、线性、时间连续的系统，状态空间模型可以如下表示：

$$\dot{\boldsymbol{x}}(t)=\boldsymbol{A}\boldsymbol{x}(t)+\boldsymbol{B}\boldsymbol{u}(t) \quad \text{其中 } \dot{\boldsymbol{x}}(t) \triangleq \frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} \tag{A-26}$$

和

$$\boldsymbol{y}(t)=\boldsymbol{C}\boldsymbol{x}(t)+\boldsymbol{D}\boldsymbol{u}(t) \tag{A-27}$$

其中式（A-26）是状态方程，式（A-27）是系统输出方程，系统的动态方程经常是指这两个方程。在式（A-26）中，状态向量

$$\boldsymbol{x}(t) = [x_1(t), x_2(t), \cdots, x_n(t)]^T \in \Re^{n\times 1} \tag{A-28}$$

包含描述系统内在行为的系统状态变量，且

$$\boldsymbol{u}(t) = [u_1(t), u_2(t), \cdots, u_p(t)]^T \in \Re^{p\times 1} \tag{A-29}$$

是（控制）系统输入的向量，$\boldsymbol{A} \in \Re^{n\times n}$, $\boldsymbol{B} \in \Re^{n\times p}$。在式（A-27）的输出方程常作为测量方程或观察，其中

$$\boldsymbol{y}(t) = [y_1(t), y_2(t), \cdots, y_q(t)]^T \in \Re^{q\times 1} \tag{A-30}$$

$\boldsymbol{C} \in \Re^{q\times n}$和$\boldsymbol{D} \in \Re^{q\times p}$。对于初始条件集合$\boldsymbol{x}(t_0) = \boldsymbol{x}_0$和定义的系统输入$\boldsymbol{u}(t)$，用式（A-26）求出$\boldsymbol{x}(t)$，然后代入式（A-27）的输出方程，可以求出系统的输出$\boldsymbol{y}(t)$。式（A-26）的状态方程可以如下解出：

$$\boldsymbol{x}(t) = \phi(t-t_0)\boldsymbol{x}_0 + \int_{t_0}^{t} \phi(t-\tau)\boldsymbol{B}\boldsymbol{u}(\tau)\mathrm{d}\tau \quad t \geqslant t_0 \tag{A-31}$$

在式（A-31）中，$\phi(t-t_0)$称作状态转换矩阵。对于线性时间不变的系统，状态转换矩阵可以写作

$$\phi(t-t_0) = e^{A(t-t_0)} \tag{A-32}$$

即矩阵指数函数。状态转换矩阵可以用拉普拉斯变换表达，即假设$t \geqslant 0$，$f(t)$的拉普拉斯变换可以写作：

$$F(s) = \mathscr{L}\{f(t)\} = \int_0^{\infty} f(t)e^{-st}\mathrm{d}t \tag{A-33}$$

逆拉普拉斯变换为：

$$f(t) = \mathscr{L}^{-1}\{F(s)\} = \frac{1}{2\pi j}\int_{c-j\infty}^{c+j\infty} F(s)e^{st}\mathrm{d}s$$

其中积分为在复平面内，从$c-j\infty$到$c+j\infty$通过路径$s = c + j\omega$求出，其中$c$是任意实数，使路径$s = c + j\omega$位于$F(s)$的收敛域[3]。因此，

$$\phi(t-t_0) = \mathscr{L}^{-1}\{\Phi(s)\}\big|_{t\leftarrow t-t_0} \tag{A-34}$$

567

其中，

$$\boldsymbol{\Phi}(s) = (s\boldsymbol{I} - \boldsymbol{A})^{-1} \tag{A-35}$$

称作预解矩阵。用式（A-31）代入式（A-27）中的输出方程，得：

$$\boldsymbol{y}(t) = \underbrace{\boldsymbol{C}\phi(t-t_0)\boldsymbol{x}_0}_{\text{自然响应}} + \underbrace{\boldsymbol{C}\int_{t_0}^{t}\phi(t-\tau)\boldsymbol{B}\boldsymbol{u}(\tau)\mathrm{d}\tau + \boldsymbol{D}\boldsymbol{u}(t)}_{\text{驱动响应}} \quad t \geqslant t_0 \tag{A-36}$$

若式（A-36）初始条件向量设置为零，即$\boldsymbol{x}_0 = \boldsymbol{0}$，则输出可以写作：

$$\boldsymbol{y}(t) = \int_{t_0}^{t}\boldsymbol{C}\phi(t-\tau)\boldsymbol{B}\boldsymbol{u}(\tau)\mathrm{d}\tau + \boldsymbol{D}\boldsymbol{u}(t) = \int_{t_0}^{t}[\boldsymbol{C}\phi(t-\tau)\boldsymbol{B} + \boldsymbol{D}\delta(t-\tau)]\boldsymbol{u}(\tau)\mathrm{d}\tau \quad t \geqslant t_0 \tag{A-37}$$

其中

$$\boldsymbol{h}(t-\tau)=\begin{cases}\boldsymbol{C}\phi(t-\tau)\boldsymbol{B}+\boldsymbol{D}\delta(t-\tau) & \text{对于 } t\geqslant\tau\\ \boldsymbol{0} & \text{对于 } t<\tau\end{cases} \tag{A-38}$$

$\boldsymbol{h}(t-\tau)$是系统脉冲响应矩阵。现在输出以系统脉冲响应矩阵写作：

$$\boldsymbol{y}(t)=\int_{t_0}^{t}\boldsymbol{h}(t-\tau)\boldsymbol{u}(\tau)\mathrm{d}\tau \quad t\geqslant t_0 \tag{A-39}$$

它表明对于零初始条件，系统的输出是系统输入与系统脉冲响应的卷积。如果对式（A-39）两边进行拉普拉斯变换，可以获得

$$\boldsymbol{Y}(s)=\boldsymbol{H}(s)\boldsymbol{U}(s) \tag{A-40}$$

其中

$$\boldsymbol{H}(s)=\begin{bmatrix}H_{11}(s) & H_{12}(s) & \cdots & H_{1p}(s)\\ H_{21}(s) & H_{22}(s) & \cdots & H_{2p}(s)\\ \cdots & \cdots & \cdots & \cdots\\ H_{q1}(s) & H_{q2}(s) & \cdots & H_{qp}(s)\end{bmatrix} \tag{A-41}$$

是系统的转换函数矩阵。因此，系统转换函数矩阵是脉冲响应矩阵的拉普拉斯变换形式，即

568

$$\boldsymbol{H}(s)=\boldsymbol{C}(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}+\boldsymbol{D}=\frac{1}{\det(s\boldsymbol{I}-\boldsymbol{A})}\boldsymbol{C}\left[\mathrm{cof}(s\boldsymbol{I}-\boldsymbol{A})\right]^{T}\boldsymbol{B}+\boldsymbol{D} \tag{A-42}$$

**定义A.9**　正常有理矩阵$\boldsymbol{H}(s)$的特征多项式定义为$\boldsymbol{H}(s)$的所有子式的最小公分母。$\boldsymbol{H}(s)$的度（也称作McMillan度或Smith-McMillan度），用$\deg[\boldsymbol{H}(s)]$表示，定义为$\boldsymbol{H}(s)$的特征多项式的次数。

若$\boldsymbol{H}(s)$是一个标量，即特征多项式$\boldsymbol{H}(s)$简化为$\boldsymbol{H}(s)$的分母。为确定特征多项式和转换函数矩阵的度，假定如下：

$$\boldsymbol{H}(s)=\begin{bmatrix}\dfrac{2s+1}{s+2} & \dfrac{4s}{s+2}\\ \dfrac{s+1}{s+2} & \dfrac{2s+1}{s+2}\end{bmatrix}$$

首先确定1阶子式。它们是矩阵的各个元素，即$\dfrac{2s+1}{s+2}$、$\dfrac{4s}{s+2}$、$\dfrac{s+1}{s+2}$和$\dfrac{2s+1}{s+2}$。$\boldsymbol{H}(s)$的二阶子式是$1/(s+2)^2$。因此，$\boldsymbol{H}(s)$的特征多项式$\Delta(s)=(s+2)^2$，$\boldsymbol{H}(s)$的度是$\deg[\boldsymbol{H}(s)]=2$。

在式（A-26）和式（A-27）给出了时间不变、线性、连续时间系统的状态空间模型，可以描述如下：

1. 如果$\boldsymbol{A}$的所有特征值的实部是严格的负数，那么动态系统是（渐近）稳定的。同样，对于一个单输入单输出的（SISO）系统，如果系统的极点严格地位于左半$s$平面，那么系统是（渐近）稳定的。系统的极点是分母多项式（特征多项式）$D(s)$的根，其中$D(s)$是变换函数$Y(s)/U(s)=H(s)=N(s)/D(s)$的分母多项式［分子多项式$N(s)$的根称作系统的零度］。如果$\boldsymbol{A}$有（非重复）特征值，其实部是0，其他特征值的实部均为负，那么这个系统称作李雅普诺夫稳定（或边缘稳定）。

2. 如果$\{\boldsymbol{A},\boldsymbol{B}\}$对是可控制的，那么动态系统是可控制的，或者

$$\rho(\boldsymbol{S})=\rho[\boldsymbol{B},\boldsymbol{AB},\boldsymbol{A}^2\boldsymbol{B},\cdots,\boldsymbol{A}^{(n-1)}\boldsymbol{B}]=n \tag{A-43}$$

其中$\boldsymbol{S}=[\boldsymbol{B},\boldsymbol{AB},\boldsymbol{A}^2\boldsymbol{B},\cdots,\boldsymbol{A}^{(n-1)}\boldsymbol{B}]\in\Re^{n\times np}$是可控性矩阵。

3. 如果$\{\boldsymbol{A}, \boldsymbol{C}\}$对是可观测的，那么动态系统是可观测的，或者

$$\rho(\boldsymbol{L}) = \rho[\boldsymbol{C}^T, \boldsymbol{A}^T\boldsymbol{C}^T, (\boldsymbol{A}^2)^T\boldsymbol{C}^T, \cdots, (\boldsymbol{A}^{(n-1)})^T\boldsymbol{C}^T]^T = n \tag{A-44}$$

其中$\boldsymbol{L} = [\boldsymbol{C}^T, \boldsymbol{A}^T\boldsymbol{C}^T, (\boldsymbol{A}^2)^T\boldsymbol{C}^T, \cdots, (\boldsymbol{A}^{(n-1)})^T\boldsymbol{C}^T]^T \in \Re^{n\times nq}$是可观察性矩阵。

4. 如果式（A-26）和式（A-27）描述一个开环系统，使用状态变量反馈控制，闭环反馈控制可以应用到系统中，控制规则如下：

$$\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{x}(t) + \boldsymbol{r}(t) \tag{A-45}$$

569

其中$\boldsymbol{K} \in \Re^{p\times n}$是闭环系统增益矩阵，$\boldsymbol{r}(t) \in \Re^{p\times 1}$是系统基准命令输入。如果$\boldsymbol{r}(t) \neq \boldsymbol{0}$，那么闭环系统是一个跟踪系统。然而，如果$\boldsymbol{r}(t) = \boldsymbol{0}$，那么闭环系统是校准器。状态变量反馈控制可以用作极点定位设计。把式（A-45）代入式（A-26）可以得到闭环系统状态方程:

$$\dot{\boldsymbol{x}}(t) = (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{r}(t) \tag{A-46}$$

设计目的是确定增益矩阵$\boldsymbol{K}$，这样，闭环系统的特征值（极点）在期望的位置，其中

$$|\lambda\boldsymbol{I} - \boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}| = 0 \tag{A-47}$$

是闭环特征方程。对于单输入系统，如果系统是可控制的，那么闭环系统的极点可以任意放置，且增益矩阵是唯一的。

5. 假设一个SISO系统，即$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t)$，$\boldsymbol{A} \in \Re^{n\times n}$，（系统设置）有不同的特征值，那么以$\boldsymbol{A}$的特征向量构造一个相似（等价）转换，即$\boldsymbol{P} = [\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_n] \in \Re^{n\times n}$（前面提及的矩阵$\boldsymbol{V}$）［回顾：$\rho(\boldsymbol{V}) = n$］，利用$\boldsymbol{P}$，定义一个新的状态向量$\bar{\boldsymbol{x}}(t) = \boldsymbol{P}^{-1}\boldsymbol{x}(t)$。把$\boldsymbol{x}(t) = \boldsymbol{P}\bar{\boldsymbol{x}}(t)$代入$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t)$，得到一个等价状态方程：

$$\dot{\bar{\boldsymbol{x}}}(t) = \underbrace{\boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}}_{\Lambda}\bar{\boldsymbol{x}}(t) + \underbrace{\boldsymbol{P}^{-1}\boldsymbol{B}}_{\Lambda}\boldsymbol{u}(t) = \Lambda\bar{\boldsymbol{x}}(t) + \Gamma\boldsymbol{u}(t) \tag{A-48}$$

和等价输出方程

$$\boldsymbol{y}(t) = \underbrace{\boldsymbol{C}\boldsymbol{P}}_{\bar{\boldsymbol{C}}}\bar{\boldsymbol{x}}(t) + \boldsymbol{D}\boldsymbol{u}(t) = \bar{\boldsymbol{C}}\bar{\boldsymbol{x}}(t) + \boldsymbol{D}\boldsymbol{u}(t) \tag{A-49}$$

在式（A-48），$\Lambda = \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}$是一个对角矩阵，$\boldsymbol{A}$的特征值在对角线上（参照A.2.9节）。因此，系统的动态性完全解耦。注意这两个等价的系统有相同的特征值，两个系统的输入和输出是相同的。

6. 如果像第5项那样假设SISO系统，$\{\boldsymbol{A}, \boldsymbol{B}\}$对是可控制的，那么存在一个相似变换如下

$$\bar{\boldsymbol{x}}(t) = \boldsymbol{Q}\boldsymbol{x}(t) = \begin{bmatrix} \tilde{\boldsymbol{Q}} \\ \tilde{\boldsymbol{Q}}\boldsymbol{A} \\ \vdots \\ \tilde{\boldsymbol{Q}}\boldsymbol{A}^{n-1} \end{bmatrix}\boldsymbol{x}(t) \tag{A-50}$$

其中

$$\tilde{\boldsymbol{Q}} = [0 \quad \cdots \quad 0 \quad 1][\boldsymbol{B}, \boldsymbol{A}\boldsymbol{B}, \boldsymbol{A}^2\boldsymbol{B}, \cdots, \boldsymbol{A}^{(n-1)}\boldsymbol{B}]^{-1} \tag{A-51}$$

$\boldsymbol{S} = [\boldsymbol{B}, \boldsymbol{A}\boldsymbol{B}, \boldsymbol{A}^2\boldsymbol{B}, \cdots, \boldsymbol{A}^{(n-1)}\boldsymbol{B}]$是系统可控性矩阵。等价系统可以用式（A-50）的转换形式：

570

$$\dot{\bar{\boldsymbol{x}}}(t) = \underbrace{\boldsymbol{Q}\boldsymbol{A}\boldsymbol{Q}^{-1}}_{\bar{\boldsymbol{A}}}\bar{\boldsymbol{x}}(t) + \underbrace{\boldsymbol{Q}\boldsymbol{B}}_{\bar{\boldsymbol{B}}}\boldsymbol{u}(t) \tag{A-52}$$

$$\boldsymbol{y}(t) = \underbrace{\boldsymbol{C}\boldsymbol{Q}^{-1}}_{\bar{\boldsymbol{C}}}\bar{\boldsymbol{x}}(t) + \underbrace{\boldsymbol{D}}_{\bar{\boldsymbol{D}}}\boldsymbol{u}(t) \tag{A-53}$$

其中

$$\overline{\boldsymbol{A}}=\boldsymbol{QAQ}^{-1}=\begin{bmatrix}0 & 1 & 0 & 0 & \cdots & 0\\ 0 & 0 & 1 & 0 & \cdots & 0\\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots\\ 0 & 0 & 0 & \cdots & 1 & 0\\ 0 & 0 & 0 & \cdots & 0 & 1\\ -\alpha_n & -\alpha_{n-1} & -\alpha_{n-2} & \cdots & -\alpha_2 & -\alpha_1\end{bmatrix} \tag{A-54}$$

$$\overline{\boldsymbol{B}}=\boldsymbol{QB}=[0,\cdots,0,1]^T \tag{A-55}$$

$$\overline{\boldsymbol{C}}=\boldsymbol{Q}^{-1}\boldsymbol{C}=[\bar{c}_1,\bar{c}_2,\cdots,\bar{c}_n] \tag{A-56}$$

$$\overline{\boldsymbol{D}}=\boldsymbol{D} \tag{A-57}$$

从式（A-52）和式（A-53），等价系统可以表示为：

$$\dot{\bar{\boldsymbol{x}}}(t)=\overline{\boldsymbol{A}}\bar{\boldsymbol{x}}(t)+\overline{\boldsymbol{B}}u(t) \tag{A-58}$$

$$y(t)=\overline{\boldsymbol{C}}\bar{\boldsymbol{x}}(t)+\overline{\boldsymbol{D}}u(t) \tag{A-59}$$

它是式（A-26）和式（A-27）的相位变量典范型，对于一个SISO系统，或是可控典范型。式（A-54）中的矩阵称作伴随矩阵（或弗罗贝尼乌斯矩阵）。式（A-54）中 $\overline{\boldsymbol{A}}$ 的特征方程如下：

$$\Delta(\lambda)=|\lambda\boldsymbol{I}-\overline{\boldsymbol{A}}|=\lambda^n+\alpha_1\lambda^{n-1}+\alpha_2\lambda^{n-2}+\cdots+\alpha_{n-1}\lambda+\alpha_n=0 \tag{A-60}$$

7. 在第6项中可控制典范型的双重典范型是可观测典范型，记作：

$$\dot{\check{\boldsymbol{x}}}(t)=\check{\boldsymbol{A}}\check{\boldsymbol{x}}(t)+\check{\boldsymbol{B}}u(t) \tag{A-61}$$

$$y(t)=\check{\boldsymbol{C}}\check{\boldsymbol{x}}(t)+\check{\boldsymbol{D}}u(t) \tag{A-62}$$

其中 $\check{\boldsymbol{A}}=\overline{\boldsymbol{A}}^T,\check{\boldsymbol{B}}=\overline{\boldsymbol{C}}^T$ [来自式（A-56）]，$\check{\boldsymbol{C}}=\overline{\boldsymbol{B}}^T$ [来自式（A-55）]，$\check{\boldsymbol{D}}=\boldsymbol{D}$。

### A.2.13 向量和矩阵的范数

向量范数的概念是向量长度的推广。如果假定任一向量 $\boldsymbol{x}\in\Re^{n\times 1}$（或 $\boldsymbol{x}\in\mathscr{C}^{n\times 1}$）和任一标量 $\alpha\in\Re$（或 $\alpha\in\mathscr{C}$），那么 $\boldsymbol{x}$ 的任一实函数可记作 $\|\boldsymbol{x}\|$，如果满足下列性质，可以定义为向量范数：

571

1. $\|\boldsymbol{x}\|\geqslant 0$ 且 $\|\boldsymbol{x}\|=0\Leftrightarrow\boldsymbol{x}=\boldsymbol{0}$
2. $\|\alpha\boldsymbol{x}\|=|\alpha|\,\|\boldsymbol{x}\|$
3. $\|\boldsymbol{x}_1+\boldsymbol{x}_2\|\leqslant\|\boldsymbol{x}_1\|+\|\boldsymbol{x}_2\|$。（著名的三角不等式）

向量 $\boldsymbol{x}=[x_1,x_2,\cdots,x_n]^T$ 的 $L_p$ 范数（或 $p$ 范数）定义为

$$\|\boldsymbol{x}\|_p\triangleq\left[\sum_{i=1}^{n}|x_i|^p\right]^{1/p} \tag{A-63}$$

其中 $p\in\Re^+$。然而，$p$ 通常取正整数值1, 2, …, ∞，相应的范数分别为1, 2, …，无穷范数。$L_p$ 范数的一个经典结论是赫尔德（Holder）不等式：对于 $\boldsymbol{x},\boldsymbol{y}\in\Re^{n\times 1}$，有

$$|\langle\boldsymbol{x},\boldsymbol{y}\rangle|=|\boldsymbol{x}^T\boldsymbol{y}|\leqslant\|\boldsymbol{x}\|_p\|\boldsymbol{y}\|_q\quad\frac{1}{p}+\frac{1}{q}=1 \tag{A-64}$$

式（A-63）中1范数或 $L_1$ 范数（或绝对值范数）定义如下：

$$\|\boldsymbol{x}\|_1 \triangleq \sum_{i=1}^{n} |x_i| \tag{A-65}$$

式（A-63）的2范数或$L_2$范数（或欧几里得范数）定义如下：

$$\|\boldsymbol{x}\|_2 \triangleq \left[\sum_{i=1}^{n} x_i^2\right]^{1/2} = (\boldsymbol{x}^T\boldsymbol{x})^{1/2} = \langle \boldsymbol{x}, \boldsymbol{x}\rangle^{1/2} \tag{A-66}$$

用欧几里得范数，柯西—施瓦茨不等式可以表示如下：

$$|\langle \boldsymbol{x}, \boldsymbol{y}\rangle| = |\boldsymbol{x}^T\boldsymbol{y}| \leqslant \|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2 \tag{A-67}$$

对于$\boldsymbol{x}, \boldsymbol{y} \in \Re^{n\times 1}$上式是式（A-64）赫尔德不等式的特例。无穷范数或$L_\infty$范数（或切比雪夫（Chebyshev）范数）定义为：

$$\|\boldsymbol{x}\|_\infty = \max(|x_1|, |x_2|, \cdots, |x_n|) \tag{A-68}$$

负无穷范数或$L_{-\infty}$范数定义为：

$$\|\boldsymbol{x}\|_{-\infty} = \min(|x_1|, |x_2|, \cdots, |x_n|) \tag{A-69}$$

内积生成范数定义为

$$\|\boldsymbol{x}\|_W = \langle \boldsymbol{x}, \boldsymbol{x}\rangle_W^{1/2} = [(\boldsymbol{W}\boldsymbol{x})\cdot(\overline{\boldsymbol{W}\boldsymbol{x}})]^{1/2} = [(\boldsymbol{W}\boldsymbol{x})^T(\overline{\boldsymbol{W}\boldsymbol{x}})]^{1/2} = [\boldsymbol{x}^T\boldsymbol{W}^T\overline{\boldsymbol{W}}\overline{\boldsymbol{x}}]^{1/2} \tag{A-70}$$

相关范数是加权欧几里得范数，形式如下：

$$\|\boldsymbol{x}\|_{2,Q} = (\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x})^{1/2} \tag{A-71}$$

其中$\boldsymbol{Q} \in \Re^{n\times n}$, $\boldsymbol{Q}^T = \boldsymbol{Q}$, $\boldsymbol{Q} > 0$。

单位向量是范数等于单位值的向量。把一个非零向量标准化是通过用它的范数去除向量的每一个元素。因此，标准化向量是单位向量。当正交向量集的每一个向量都是单位长度时，称为规范正交。两个向量$\boldsymbol{x}, \boldsymbol{y} \in \Re^{n\times 1}$之间的距离是$\|\boldsymbol{x}-\boldsymbol{y}\|$（取决于所选范数的类型）。 572

方阵$\boldsymbol{A} \in \Re^{n\times n}$（$\boldsymbol{A} \in \mathscr{C}^{n\times n}$）的范数记作$\|\boldsymbol{A}\|$，是一个实值函数，必须满足下列条件：

1. $\|\boldsymbol{A}\| \geqslant 0$
2. $\|\boldsymbol{A}\| = 0 \Leftrightarrow \boldsymbol{A} = \boldsymbol{0}$
3. 对任一标量$\alpha$，$\|\alpha\boldsymbol{A}\| = |\alpha|\,\|\boldsymbol{A}\|$
4. 假设$\boldsymbol{A}, \boldsymbol{B} \in \Re^{n\times n}$（$\boldsymbol{A}, \boldsymbol{B} \in \mathscr{C}^{n\times n}$），$\|\boldsymbol{A}+\boldsymbol{B}\| \leqslant \|\boldsymbol{A}\| + \|\boldsymbol{B}\|$（三角不等式）
5. 假设$\boldsymbol{A}, \boldsymbol{B} \in \Re^{n\times n}$（$\boldsymbol{A}, \boldsymbol{B} \in \mathscr{C}^{n\times n}$），$\|\boldsymbol{A}\boldsymbol{B}\| \leqslant \|\boldsymbol{A}\|\,\|\boldsymbol{B}\|$（一致性条件）

一个重要的范数是矩阵$\boldsymbol{A} \in \Re^{n\times n}$（$\boldsymbol{A} \in \mathscr{C}^{n\times n}$）($\boldsymbol{A} = [a_{ij}]$, $i, j = 1, 2, \cdots, n$)的弗罗贝尼乌斯（Frobenius）范数，为：

$$\|\boldsymbol{A}\|_F = \left(\sum_{i=1}^{n}\sum_{j=1}^{n} |a_{ij}|^2\right)^{1/2} \tag{A-72}$$

可以看出弗罗贝尼乌斯范数可由$\boldsymbol{A}$的非零奇异值的平方和开方得到（参照A.2.14节）。

导出矩阵范数可由向量范数导出。每一个向量$p$范数导出（或生成）矩阵$p$范数：

$$\|\boldsymbol{A}\|_p = \sup_{\boldsymbol{x}\neq\boldsymbol{0}} \frac{\|\boldsymbol{A}\boldsymbol{x}\|_p}{\|\boldsymbol{x}\|_p} = \sup_{\|\boldsymbol{x}\|_p=1} \|\boldsymbol{A}\boldsymbol{x}\|_p \tag{A-73}$$

其中“sup”意思是上确界，$\|\boldsymbol{A}\boldsymbol{x}\|_p$的最小上界。由式（A-73）可以得：

$$\|\boldsymbol{A}\boldsymbol{x}\|_p \leqslant \|\boldsymbol{A}\|_p\|\boldsymbol{x}\|_p \tag{A-74}$$

如果一个向量范数与矩阵范数是相容的，那么它必须保持式（A-74）成立。导出范数总是与生成它们的向量范数相容。相容性对导出范数没有约束。例如，即使欧几里得范数没有导出弗罗贝尼乌斯矩阵范数，弗罗贝尼乌斯范数也是与欧几里得向量范数相容的。

一些有用的导出范数如下：

1.$L_1$矩阵范数（由$L_1$向量范数导出）

$$\|A\|_1 = \max_{j=1,2,\cdots,n}\left(\sum_{i=1}^{n}|a_{ij}|\right) \tag{A-75}$$

这是列绝对值和的最大值。

2. $L_\infty$矩阵范数（由$L_\infty$向量范数导出）：

573

$$\|A\|_\infty = \max_{i=1,2,\cdots,n}\left(\sum_{j=1}^{n}|a_{ij}|\right) \tag{A-76}$$

这是行绝对值和的最大值。

3. 谱范数（由欧几里得范数导出）：

$$\|A\|_2 = [\lambda_{\max}(A^*A)]^{1/2} \tag{A-77}$$

是$A^*A = \bar{A}^TA$的最大特征值的平方根。可当作$A$的最大奇异值计算（参照A.2.14节）。

方阵$A \in \Re^{n\times n}$($A \in \mathscr{C}^{n\times n}$)的谱半径记作$\sigma_r(A)$，是$A$的任一特征值的最大绝对值。这样，假设$A$的特征值为$\lambda_i$，$i = 1, 2, \cdots, n$，$A$的谱半径为：

$$\sigma_r(A) = \max_{1\leqslant i\leqslant n}|\lambda_i| \tag{A-78}$$

对于任一矩阵范数：

$$\sigma_r(A) \leqslant \|A\| \tag{A-79}$$

限定了$A$的特征值的界。矩阵$A$的谱半径的等价表达可写作：

$$\sigma_r(A) = \lim_{m\to\infty}\|A^m\|^{1/m} \tag{A-80}$$

Gerschgorin定理[1]可以提供估计矩阵$A$的谱半径的方法。

### A.2.14　奇异值分解

假设$A \in \Re^{m\times n}$，实正交矩阵

$$U = [u_1, u_2, \cdots, u_m] \in \Re^{m\times m} \tag{A-81}$$

$$V = [v_1, v_2, \cdots, v_n] \in \Re^{n\times n} \tag{A-82}$$

存在，得到：

$$U^TAV = \text{pseudodiag}[\sigma_1, \sigma_2, \cdots, \sigma_p] = S \tag{A-83}$$

其中$S \in \Re^{m\times n}$，

$$p = \min\{m, n\} \tag{A-84}$$

以及

$$\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_p \geqslant 0 \tag{A-85}$$

因为矩阵$U$和$V$是正交的，式（A-83）可写作

$$A = USV^T = \sum_{i=1}^{r}\sigma_i u_i v_i^T \tag{A-86}$$

称作矩阵$A$的奇异值分解，其中$\rho(A)=r$（最小奇异值的指数）。若

$$\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_r > \sigma_{r+1} = \sigma_{r+2} = \cdots = \sigma_p = 0 \tag{A-87}$$

574

那么可以定义：

$$\boldsymbol{U}_r \triangleq \boldsymbol{U}(:,1:r) \tag{A-88}$$

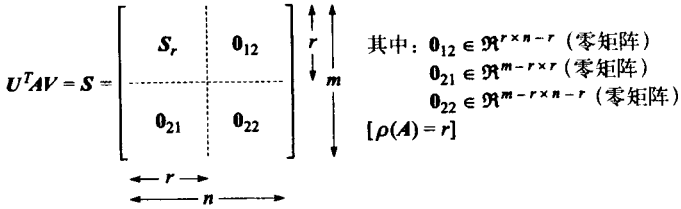
$$\boldsymbol{V}_r \triangleq \boldsymbol{V}(:,1:r) \tag{A-89}$$

$$\boldsymbol{S}_r \triangleq \boldsymbol{S}(1:r,1:r) \tag{A-90}$$

其中$\boldsymbol{U}_r \in \Re^{m\times r}$, $\boldsymbol{V}_r \in \Re^{n\times r}$, $\boldsymbol{S}_r \in \Re^{r\times r}$。式（A-88）、式（A-89）和式（A-90）中所用的符号来自MATLAB的引用数组的方法。例如，在式（A-88）中，$\boldsymbol{U}(:, 1:r)$表示选取所有的行，这样第1个“:”和“1:$r$”表示取第1～$r$列。因此，式（A-86）可以写作

$$\boldsymbol{A} = \boldsymbol{U}_r \boldsymbol{S}_r \boldsymbol{V}_r^T = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T \tag{A-91}$$

从图A-1可以直接看到$\boldsymbol{S}_r$和$\boldsymbol{S}$之间的关系。




图A-1 矩阵$\boldsymbol{A} \in \Re^{m\times n}$的SVD中，$\boldsymbol{S}_r$和$\boldsymbol{S}$的关系

现在可以很容易明白MATLAB是如何使用矩阵的SVD计算它的秩了。即，矩阵$\boldsymbol{A} \in \Re^{m\times n}$的非零奇异值数目就是矩阵的秩。根据式（A-86）有：

$$\boldsymbol{A}\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i \tag{A-92}$$

$$\boldsymbol{A}^T\boldsymbol{u}_i = \sigma_i \boldsymbol{v}_i \tag{A-93}$$

其中$i = 1, 2, \cdots, \min\{m, n\}$, $\boldsymbol{u}_i \in \Re^{m\times 1}$（左奇异向量），$\boldsymbol{v}_i \in \Re^{n\times 1}$（右奇异向量）。这与标准特征值问题相似（参照A.2.9节）。

当$\boldsymbol{A} \in \Re^{n\times n}$, $\boldsymbol{A}^T = \boldsymbol{A}$, $\boldsymbol{A} \geqslant 0$，SVD产生一个特例。在这种情况下，奇异值是$\boldsymbol{A}$的特征值，即：

$$\boldsymbol{S} = \begin{bmatrix} \lambda_1 & & & \boldsymbol{0} \\ & \lambda_2 & & \\ & & \ddots & \\ \boldsymbol{0} & & & \lambda_n \end{bmatrix} = \operatorname{diag}[\lambda_1, \lambda_2, \cdots, \lambda_n] = \boldsymbol{\Lambda} \in \Re^{n\times n} \tag{A-94}$$

其中$\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n \geqslant 0$是$\boldsymbol{A}$的实特征值（因为$\boldsymbol{A}$是对称的），相应的特征向量集合$\boldsymbol{v}_i$是正交的（因为$\boldsymbol{A}$是对称的），$i = 1, 2, \cdots, n$。此外，$\boldsymbol{U} = \boldsymbol{V}$；即，左右奇异向量是相同的。为了不失一般性，假设$\rho(\boldsymbol{A}) = n$。$\boldsymbol{A}$的SVD已给定，即$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$，可以得到：

575

$$\boldsymbol{A}^T\boldsymbol{A} = (\boldsymbol{V}\boldsymbol{S}^T\boldsymbol{U}^T)(\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T) = \boldsymbol{V}\boldsymbol{S}^T\underbrace{\boldsymbol{U}^T\boldsymbol{U}}_{\boldsymbol{I}}\boldsymbol{S}\boldsymbol{V}^T = \boldsymbol{V}\underbrace{\boldsymbol{S}^T\boldsymbol{S}}_{\boldsymbol{S}^2}\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{S}^2\boldsymbol{V}^T \tag{A-95}$$

然而，由式（A-94），$\boldsymbol{S}^2 = \boldsymbol{\Lambda}^2$，式（A-95）可以写作：

$$\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}^2\boldsymbol{V}^T = \boldsymbol{A}\boldsymbol{A}^T \quad \text{因为} \boldsymbol{U} = \boldsymbol{V} \tag{A-96}$$

这是$\boldsymbol{A}^T\boldsymbol{A}$（或$\boldsymbol{A}\boldsymbol{A}^T$）的特征值的分解（参照A.2.10节）。现在很容易明白如何用矩阵的SVD计算

谱范数（参照A.2.13节）。即，$\boldsymbol{A}^T\boldsymbol{A}$的最大奇异值是$\lambda_1^2$，$\boldsymbol{A}$的谱范数是$\sqrt{\lambda_1^2}=\lambda_1$。然而，这可由$\boldsymbol{A}$的SVD求出；即，$\boldsymbol{A}$的最大奇异值是$\sigma_1=\lambda_1$[见式（A-94）]。

矩阵$\boldsymbol{A}$的弗罗贝尼乌斯范数［见A.2.13节，方程（A-72)］由它的SVD决定。弗罗贝尼乌斯范数的另一种形式为：

$$\|\boldsymbol{A}\|_F=[\text{trace}(\boldsymbol{A}^T\boldsymbol{A})]^{1/2}=[\text{trace}(\boldsymbol{A}\boldsymbol{A}^T)]^{1/2} \tag{A-97}$$

如果根据SVD来表示$\boldsymbol{A}$，即$\boldsymbol{A}=\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$，那么式（A-97）可以写作：

$$\begin{aligned}\|\boldsymbol{A}\|_F&=[\text{trace}(\boldsymbol{A}^T\boldsymbol{A})]^{1/2}=[\text{trace}(\boldsymbol{A}\boldsymbol{A}^T)]^{1/2}=[\text{trace}(\boldsymbol{S}^T\boldsymbol{S})]^{1/2}=[\text{trace}(\boldsymbol{S}\boldsymbol{S}^T)]^{1/2}\\&=\left\{\text{trace}\begin{bmatrix}\sigma_1^2 & & \mathbf{0}\\ & \sigma_2^2 & \\ & \ddots & \\ \mathbf{0} & & \sigma_p^2\end{bmatrix}\right\}^{1/2}=\left(\sum_{i=1}^{p}\sigma_i^2\right)^{1/2}\end{aligned} \tag{A-98}$$

（回忆$p=\min\{m,n\}$）

重新回顾一下A.2.7节中矩阵伪逆的问题，这涉及解决联立线性代数方程组$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$的问题，其中$\boldsymbol{A}\in\Re^{m\times n}$，$\boldsymbol{x}\in\Re^{n\times 1}$，$\boldsymbol{b}\in\Re^{m\times 1}$。如果根据式（A-91）以矩阵的SVD来表示$\boldsymbol{A}$，即，$\boldsymbol{A}=\boldsymbol{U}_r\boldsymbol{S}_r\boldsymbol{V}_r^T$，其中$r=\rho(\boldsymbol{A})$，式（A-8）的误差函数可以写作：

$$\mathscr{E}(\boldsymbol{x})=\frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2=\frac{1}{2}\boldsymbol{x}^T\boldsymbol{V}_r\boldsymbol{S}_r^T\boldsymbol{S}_r\boldsymbol{V}_r^T\boldsymbol{x}-\boldsymbol{x}^T\boldsymbol{V}_r\boldsymbol{S}_r^T\boldsymbol{U}_r^T\boldsymbol{b}+\frac{1}{2}\boldsymbol{b}^T\boldsymbol{b} \tag{A-99}$$

对式（A-99）的$\mathscr{E}(\boldsymbol{x})$求向量$\boldsymbol{x}$的偏导，令其等于零，得：

$$\boldsymbol{V}_r\boldsymbol{S}_r^T\boldsymbol{S}_r\boldsymbol{V}_r^T\boldsymbol{x}-\boldsymbol{V}_r\boldsymbol{S}_r^T\boldsymbol{V}_r^T\boldsymbol{b}=\boldsymbol{0} \tag{A-100}$$

式（A-100）两边左乘$\boldsymbol{V}_r^T$得$\boldsymbol{S}_r^T\boldsymbol{S}_r\boldsymbol{V}_r^T\boldsymbol{x}=\boldsymbol{S}_r^T\boldsymbol{U}_r^T\boldsymbol{b}$，两边再左乘$(\boldsymbol{S}_r^T\boldsymbol{S}_r)^{-1}$得：

576

$$\boldsymbol{V}_r^T\boldsymbol{x}=(\boldsymbol{S}_r^T\boldsymbol{S}_r)^{-1}\boldsymbol{S}_r^T\boldsymbol{U}_r^T\boldsymbol{b}=\boldsymbol{S}_r^{-1}\boldsymbol{U}_r^T\boldsymbol{b} \tag{A-101}$$

现在对于$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$，可以通过式（A-101）两边左乘$\boldsymbol{V}_r$，求最小二乘解$\boldsymbol{x}^*$：

$$\boldsymbol{x}^*=\boldsymbol{V}_r\boldsymbol{S}_r^{-1}\boldsymbol{U}_r^T\boldsymbol{b} \tag{A-102}$$

由式（A-102）的$\boldsymbol{A}$的伪逆为：

$$\boldsymbol{A}^+=\boldsymbol{V}_r\boldsymbol{S}_r^{-1}\boldsymbol{U}_r^T \tag{A-103}$$

然而，式（A-103）中的$\boldsymbol{A}$的伪逆可以用$\boldsymbol{A}$的SVD表示：

$$\boldsymbol{A}^+=\boldsymbol{V}\boldsymbol{S}^+\boldsymbol{U}^T \tag{A-104}$$

其中

$$\boldsymbol{S}^+=\begin{bmatrix}\boldsymbol{S}_r^{-1} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}^T=\begin{bmatrix}\boldsymbol{S}_r^{-1} & \boldsymbol{0}_{r\times m-r}\\ \boldsymbol{0}_{n-r\times r} & \boldsymbol{0}_{n-r\times m-r}\end{bmatrix}\in\Re^{n\times m}$$

因此，$\boldsymbol{S}^+$可以只通过计算非零奇异值的倒数得到（取矩阵的转置），式（A-102）中$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$的线性最小二乘解可以写作：

$$\boldsymbol{x}^*=\boldsymbol{V}\boldsymbol{S}^+\boldsymbol{U}^T\boldsymbol{b}=\boldsymbol{A}^+\boldsymbol{b} \tag{A-105}$$

### A.2.15　矩阵条件数

矩阵$\boldsymbol{A}\in\Re^{m\times n}$的条件数定义如下：

$$\mathrm{cond}_p(\boldsymbol{A}) \triangleq \|\boldsymbol{A}\|_p \|\boldsymbol{A}^+\|_p \tag{A-106}$$

其中$p$与矩阵范数相关，$\boldsymbol{A}^+$是矩阵的伪逆。小条件数矩阵称作良态矩阵，大条件数矩阵称作病态矩阵。在许多应用中，用到矩阵$\boldsymbol{A} \in \Re^{m\times n}$的$L_2$范数条件数。因此，条件数可以记作：

$$\mathrm{cond}_2(\boldsymbol{A}) = \|\boldsymbol{A}\|_2 \|\boldsymbol{A}^+\|_2 = \frac{\sigma_1}{\sigma_p} \tag{A-107}$$

即，$\boldsymbol{A}$的最大奇异值与最小奇异值的比，其中$p = \min\{m, n\}$。当函数cond用于计算矩阵的条件数时，这就是由MATLAB生成的条件数。

矩阵的条件数在分析联立线性代数方程组$\boldsymbol{Ax} = \boldsymbol{b}$的解的灵敏性时自然出现。为了简便，假设$\boldsymbol{A} \in \Re^{n\times n}$和$\rho(\boldsymbol{A}) = n$，如果$\boldsymbol{b}$中存在舍入误差，即$\boldsymbol{b} + \Delta\boldsymbol{b}$，那么解改为$\boldsymbol{x} + \Delta\boldsymbol{x}$。因此，可以得到：$\boldsymbol{A}(\boldsymbol{x} + \Delta\boldsymbol{x}) = (\boldsymbol{b} + \Delta\boldsymbol{b})$，减去$\boldsymbol{Ax} = \boldsymbol{b}$得：

$$\boldsymbol{A}\Delta\boldsymbol{x} = \Delta\boldsymbol{b} \tag{A-108}$$

577

求解$\Delta\boldsymbol{x}$得：

$$\Delta\boldsymbol{x} = \boldsymbol{A}^{-1}\Delta\boldsymbol{b} \tag{A-109}$$

用式（A-74）中矩阵范数的性质，可以写作：

$$\|\Delta\boldsymbol{x}\| \leqslant \|\boldsymbol{A}^{-1}\| \, \|\Delta\boldsymbol{b}\| \tag{A-110}$$

式（A-110）两边除以$\|\boldsymbol{x}\|$，右边乘以再除以$\|\boldsymbol{b}\|$得：

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leqslant \frac{\|\boldsymbol{b}\|\|\boldsymbol{A}^{-1}\|\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{x}\|\|\boldsymbol{b}\|} \tag{A-111}$$

然而，$\|\boldsymbol{b}\| = \|\boldsymbol{Ax}\|$。因此，式（A-111）可以写作：

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leqslant \left[\frac{\|\boldsymbol{Ax}\|}{\|\boldsymbol{x}\|}\right]\|\boldsymbol{A}^{-1}\|\frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|} \tag{A-112}$$

此外，由式（A-74），也可以写作$\|\boldsymbol{A}\| \, \|\boldsymbol{x}\| \geqslant \|\boldsymbol{Ax}\|$，两边除以$\|\boldsymbol{x}\|$得$\|\boldsymbol{A}\| \geqslant \|\boldsymbol{Ax}\|/\|\boldsymbol{x}\|$。利用此结果，式（A-112）可以写作：

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \quad \underbrace{\|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|}_{\mathrm{cond}(\boldsymbol{A})}\frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|} \tag{A-113}$$

或

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \quad \mathrm{cond}(\boldsymbol{A})\frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|} \tag{A-114}$$

因此，在式（A-114）中的相对解误差$\dfrac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|}$小于（或等于）$\boldsymbol{A}$的条件数，即，cond($\boldsymbol{A}$)乘以相对问题误差，其中右边是上界。如果问题误差记作$\Delta\boldsymbol{A}$（即，$\boldsymbol{A}$中的误差代替$\boldsymbol{b}$），那么式（A-114）可以变换成：

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x} + \Delta\boldsymbol{x}\|} \leqslant \mathrm{cond}(\boldsymbol{A})\frac{\|\Delta\boldsymbol{A}\|}{\|\boldsymbol{A}\|} \tag{A-115}$$

若$\boldsymbol{A}$和$\boldsymbol{b}$中都存在微扰，即分别为$\Delta\boldsymbol{A}$和$\Delta\boldsymbol{b}$，则如下不等式成立：

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leqslant \mathrm{cond}(\boldsymbol{A})\left(\frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|} + \frac{\|\Delta\boldsymbol{A}\|}{\|\boldsymbol{A}\|}\right) \tag{A-116}$$

### A.2.16 分块矩阵运算

很多矩阵运算可以通过对矩阵进行分块适当简化。下面给出一些有用的分块矩阵运算。 578

矩阵行列式

假设$\boldsymbol{A}\in\Re^{n\times n}$, $\boldsymbol{B}\in\Re^{n\times m}$, $\boldsymbol{C}\in\Re^{m\times n}$, $\boldsymbol{D}\in\Re^{m\times m}$, $\boldsymbol{M}\in\Re^{(n+m)\times(n+m)}$分块为：

$$\boldsymbol{M}=\begin{bmatrix}\boldsymbol{A} & \boldsymbol{B}\\ \boldsymbol{C} & \boldsymbol{D}\end{bmatrix} \tag{A-117}$$

那么若$\det(\boldsymbol{A})\neq 0$，则

$$\det(\boldsymbol{M})=\det(\boldsymbol{A})\det(\boldsymbol{D}-\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}) \tag{A-118}$$

假设$\boldsymbol{A}\in\Re^{m\times m}$, $\boldsymbol{B}\in\Re^{m\times p}$, $\boldsymbol{C}\in\Re^{p\times m}$, $\boldsymbol{D}\in\Re^{p\times p}$, $\boldsymbol{M}\in\Re^{(m+p)\times(m+p)}$分块为：

$$\boldsymbol{M}=\begin{bmatrix}\boldsymbol{A} & \boldsymbol{B}\\ \boldsymbol{C} & \boldsymbol{D}\end{bmatrix} \tag{A-119}$$

若$\det(\boldsymbol{D})\neq 0$，则

$$\det(\boldsymbol{M})=\det(\boldsymbol{D})\det(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}) \tag{A-120}$$

对于$\boldsymbol{A}$和$\boldsymbol{B}$方阵，

$$\det\begin{bmatrix}\boldsymbol{A} & \boldsymbol{B}\\ \boldsymbol{0} & \boldsymbol{C}\end{bmatrix}=\det(\boldsymbol{A})\det(\boldsymbol{C})\quad 或\quad \det\begin{bmatrix}\boldsymbol{A} & \boldsymbol{0}\\ \boldsymbol{B} & \boldsymbol{C}\end{bmatrix}=\det(\boldsymbol{A})\det(\boldsymbol{C}) \tag{A-121}$$

假设$\boldsymbol{B}\in\Re^{n\times m}$, $\boldsymbol{C}\in\Re^{m\times n}$，那么

$$\det\begin{bmatrix}\boldsymbol{I}_n & \boldsymbol{B}\\ -\boldsymbol{C} & \boldsymbol{I}_m\end{bmatrix}=\det(\boldsymbol{I}_m-\boldsymbol{C}\boldsymbol{B})=\det(\boldsymbol{I}_n-\boldsymbol{B}\boldsymbol{C}) \tag{A-122}$$

分块矩阵的逆

假设$\boldsymbol{B}\in\Re^{n\times m}$和$\boldsymbol{C}\in\Re^{m\times n}$，那么

$$\begin{bmatrix}\boldsymbol{I}_n & \boldsymbol{B}_{n\times m}\\ \boldsymbol{0}_{m\times n} & \boldsymbol{I}_m\end{bmatrix}^{-1}=\begin{bmatrix}\boldsymbol{I}_n & -\boldsymbol{B}_{n\times m}\\ \boldsymbol{0}_{m\times n} & \boldsymbol{I}_m\end{bmatrix}\in\Re^{(n+m)\times(n+m)} \tag{A-123}$$

和

$$\begin{bmatrix}\boldsymbol{I}_n & \boldsymbol{0}_{n\times m}\\ \boldsymbol{C}_{m\times n} & \boldsymbol{I}_m\end{bmatrix}^{-1}=\begin{bmatrix}\boldsymbol{I}_n & \boldsymbol{0}_{n\times m}\\ -\boldsymbol{C}_{m\times n} & \boldsymbol{I}_m\end{bmatrix}\in\Re^{(n+m)\times(n+m)} \tag{A-124}$$

假设$\boldsymbol{A},\boldsymbol{B}\in\Re^{n\times n}$和$\rho(\boldsymbol{A})=\rho(\boldsymbol{B})=n$，那么

$$\begin{bmatrix}\boldsymbol{A} & \boldsymbol{0}\\ \boldsymbol{P} & \boldsymbol{B}\end{bmatrix}^{-1}=\begin{bmatrix}\boldsymbol{A}^{-1} & \boldsymbol{0}\\ -\boldsymbol{B}^{-1}\boldsymbol{P}\boldsymbol{A}^{-1} & \boldsymbol{B}^{-1}\end{bmatrix} \tag{A-125}$$

和

$$\begin{bmatrix}\boldsymbol{A} & \boldsymbol{Q}\\ \boldsymbol{0} & \boldsymbol{B}\end{bmatrix}^{-1}=\begin{bmatrix}\boldsymbol{A}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{Q}\boldsymbol{B}^{-1}\\ \boldsymbol{0} & \boldsymbol{B}^{-1}\end{bmatrix} \tag{A-126}$$

579

假设$\boldsymbol{A}\in\Re^{n\times n}$, $\boldsymbol{B}\in\Re^{n\times m}$, $\boldsymbol{C}\in\Re^{m\times n}$, $\boldsymbol{D}\in\Re^{m\times m}$，那么

$$\begin{bmatrix}\boldsymbol{A} & \boldsymbol{B}\\ \boldsymbol{C} & \boldsymbol{D}\end{bmatrix}^{-1}=\begin{bmatrix}\boldsymbol{A}^{-1}+\boldsymbol{E}_A\Delta_A^{-1}\boldsymbol{F}_A & -\boldsymbol{E}_A\Delta_A^{-1}\\ -\Delta_A^{-1}\boldsymbol{F}_A & \Delta_A^{-1}\end{bmatrix} \tag{A-127}$$

如果$\Delta_A=\boldsymbol{D}-\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}$（称为$\boldsymbol{A}$的舒尔补）和$\boldsymbol{A}$是可逆的，其中$\boldsymbol{E}_A=\boldsymbol{A}^{-1}\boldsymbol{B}$, $\boldsymbol{F}_A=\boldsymbol{C}\boldsymbol{A}^{-1}$。同样，如果$\boldsymbol{D}$和$\Delta_D=\boldsymbol{A}-\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}$（称为$\boldsymbol{D}$的舒尔补）是可逆的，那么

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^{-1} = \begin{bmatrix} \Delta_D^{-1} & -\Delta_D^{-1}\boldsymbol{F}_D \\ -\boldsymbol{E}_D\Delta_D^{-1} & \boldsymbol{E}_D\Delta_D^{-1}\boldsymbol{F}_D + \boldsymbol{D}^{-1} \end{bmatrix} \tag{A-128}$$

其中$\boldsymbol{E}_D = \boldsymbol{D}^{-1}\boldsymbol{C}, \boldsymbol{F}_D = \boldsymbol{B}\boldsymbol{D}^{-1}$。此外，如果$\boldsymbol{A}$，$\Delta_A = \boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}, \Delta_D = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}$都是可逆的，那么

$$\Delta_D^{-1} = \boldsymbol{A}^{-1} + \boldsymbol{E}_A\Delta_A^{-1}\boldsymbol{F}_A \tag{A-129}$$

分块矩阵的秩

假设$\boldsymbol{A} \in \Re^{n\times m}, \boldsymbol{B} \in \Re^{p\times q}, \boldsymbol{C} \in \Re^{p\times m}$，那么

$$\rho(\boldsymbol{A}) + \rho(\boldsymbol{B}) = \rho\begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B} \end{bmatrix} \leqslant \rho\begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{B} \end{bmatrix} \tag{A-130}$$

和

$$\rho(\boldsymbol{A}) + \rho(\boldsymbol{B}) = \rho\begin{bmatrix} \boldsymbol{0} & \boldsymbol{A} \\ \boldsymbol{B} & \boldsymbol{0} \end{bmatrix} \leqslant \rho\begin{bmatrix} \boldsymbol{0} & \boldsymbol{A} \\ \boldsymbol{B} & \boldsymbol{C} \end{bmatrix} \tag{A-131}$$

假设$\boldsymbol{A} \in \Re^{n\times n}, \boldsymbol{B} \in \Re^{n\times m}, \boldsymbol{C} \in \Re^{p\times n}, \boldsymbol{D} \in \Re^{p\times m}$，若$\boldsymbol{A}$是可逆的，那么

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{C}\boldsymbol{A}^{-1} & \boldsymbol{I} \end{bmatrix}\begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B} \end{bmatrix}\begin{bmatrix} \boldsymbol{I} & \boldsymbol{A}^{-1}\boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{A-132}$$

和

$$\rho\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} = n + \rho(\boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}) \tag{A-133}$$

若$p = m$，见式（A-118）。

假设$\boldsymbol{A} \in \Re^{n\times m}, \boldsymbol{B} \in \Re^{n\times p}, \boldsymbol{C} \in \Re^{p\times m}, \boldsymbol{D} \in \Re^{p\times p}$，若$\boldsymbol{D}$是可逆的，那么

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{B}\boldsymbol{D}^{-1} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}\begin{bmatrix} \boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D} \end{bmatrix}\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{D}^{-1}\boldsymbol{C} & \boldsymbol{I} \end{bmatrix} \tag{A-134}$$

和

$$\rho\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} = p + \rho(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}) \tag{A-135}$$

若$n = m$，见式（A-120）。 580

假设$\boldsymbol{A} \in \Re^{n\times m}, \boldsymbol{B} \in \Re^{p\times m}$，那么

$$\rho\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{B} \end{bmatrix} \leqslant \rho(\boldsymbol{A}) + \rho(\boldsymbol{B}) \tag{A-136}$$

若$\boldsymbol{A} \in \Re^{n\times m}, \boldsymbol{B} \in \Re^{n\times p}$，那么

$$\rho[\boldsymbol{A}, \boldsymbol{B}] \leqslant \rho(\boldsymbol{A}) + \rho(\boldsymbol{B}) \tag{A-137}$$

### A.2.17 克罗内克积与和

克罗内克积与和[4, 5]在系统理论的问题中十分有用。假设矩阵$\boldsymbol{A} \in \Re^{p\times q}, \boldsymbol{B} \in \Re^{m\times n}$，克罗内克积定义为：

$$\boldsymbol{A}\oplus\boldsymbol{B}\triangleq\begin{bmatrix}a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \cdots & a_{1q}\boldsymbol{B}\\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \cdots & a_{2q}\boldsymbol{B}\\ \cdots & \cdots & \cdots & \cdots\\ a_{p1}\boldsymbol{B} & a_{p2}\boldsymbol{B} & \cdots & a_{pq}\boldsymbol{B}\end{bmatrix}=\boldsymbol{C}\in\Re^{pm\times qn} \tag{A-138}$$

假设矩阵$\boldsymbol{N}\in\Re^{n\times n},\boldsymbol{M}\in\Re^{m\times m}$，克罗内克和定义为：

$$\boldsymbol{N}\oplus\boldsymbol{M}\triangleq\boldsymbol{N}\otimes\boldsymbol{I}_m+\boldsymbol{I}_n\otimes\boldsymbol{M}=\boldsymbol{P}\in\Re^{nm\times nm} \tag{A-139}$$

克罗内克和本质上允许通常不适合求和的矩阵“相加”。假定$\boldsymbol{A}\in\Re^{p\times q}$，定义一个矩阵的重要的向量值函数如下[4, 5]：

$$\mathrm{vec}(\boldsymbol{A})\triangleq\Big[\underbrace{(a_{11},a_{21},\cdots,a_{p1})}_{\boldsymbol{A}\text{的第1列}},\underbrace{(a_{12},a_{22},\cdots,a_{p2})}_{\boldsymbol{A}\text{的第2列}},\cdots,\underbrace{(a_{1q},a_{2q},\cdots,a_{pq})}_{\boldsymbol{A}\text{的第}q\text{列}}\Big] \tag{A-140}$$

其中$\mathrm{vec}(\boldsymbol{A})\in\Re^{pq\times 1}$。矩阵$\boldsymbol{A}$的第$k$行可以写作：$\boldsymbol{A}_{k:}$，第$k$列记作$\boldsymbol{A}_{:k}$。因此，式（A-140）可以写作：

$$\mathrm{vec}(\boldsymbol{A})\triangleq\begin{bmatrix}\boldsymbol{A}_{:1}\\ \boldsymbol{A}_{:2}\\ \vdots\\ \boldsymbol{A}_{:q}\end{bmatrix}\in\Re^{pq\times 1} \tag{A-141}$$

上面的vec运算符可以修改为只选择方阵主对角线上的元素。即，方阵$\boldsymbol{N}\in\Re^{n\times n}$的向量值函数可以定义为：

$$\mathrm{vecd}(\boldsymbol{N})\triangleq[n_{11},n_{22},\cdots,n_{nn}]^T\in\Re^{n\times 1} \tag{A-142}$$

581 同样，假设$\boldsymbol{C}\in\Re^{p\times q},\boldsymbol{D}\in\Re^{m\times q}$（注意矩阵有相同的列数），*Khatri-Rao*积定义如下：

$$\boldsymbol{D}\odot\boldsymbol{C}\triangleq[\boldsymbol{D}_{:1}\otimes\boldsymbol{C}_{:1},\boldsymbol{D}_{:2}\otimes\boldsymbol{C}_{:2},\cdots,\boldsymbol{D}_{:q}\otimes\boldsymbol{C}_{:q}]\in\Re^{mp\times q} \tag{A-143}$$

一些有用的性质包括克罗内克积，克罗内克和，vec运算符和Khatri-Rao积如下：

1. $(\boldsymbol{A}\otimes\boldsymbol{B})^T=(\boldsymbol{A}^T\otimes\boldsymbol{B}^T)$
2. $(\boldsymbol{A}\otimes\boldsymbol{B})\otimes\boldsymbol{C}=\boldsymbol{A}\otimes(\boldsymbol{B}\otimes\boldsymbol{C})$
3. $(\boldsymbol{A}\otimes\boldsymbol{B})(\boldsymbol{D}\otimes\boldsymbol{G})=\boldsymbol{AD}\otimes\boldsymbol{BG}$
4. $(\alpha\boldsymbol{A})\otimes\boldsymbol{B}=\boldsymbol{A}\otimes(\alpha\boldsymbol{B})$，其中$\alpha\in\Re$
5. $(\boldsymbol{A}+\boldsymbol{H})\otimes(\boldsymbol{B}+\boldsymbol{R})=\boldsymbol{A}\otimes\boldsymbol{B}+\boldsymbol{A}\otimes\boldsymbol{R}+\boldsymbol{H}\otimes\boldsymbol{B}+\boldsymbol{H}\otimes\boldsymbol{R}$
6. $(\boldsymbol{A}+\boldsymbol{H})\otimes\boldsymbol{B}=\boldsymbol{A}\otimes\boldsymbol{B}+\boldsymbol{H}\otimes\boldsymbol{B}$
7. $\boldsymbol{A}\otimes(\boldsymbol{B}+\boldsymbol{R})=\boldsymbol{A}\otimes\boldsymbol{B}+\boldsymbol{A}\otimes\boldsymbol{R}$
8. $(\boldsymbol{N}\otimes\boldsymbol{M})^{-1}=\boldsymbol{N}^{-1}\otimes\boldsymbol{M}^{-1}$，如果$\boldsymbol{N}$和$\boldsymbol{M}$可逆
9. $\det(\boldsymbol{N}\otimes\boldsymbol{M})=(\det\boldsymbol{N})^m(\det\boldsymbol{M})^n=\det(\boldsymbol{M}\otimes\boldsymbol{N})$
10. $\mathrm{trace}(\boldsymbol{N}\otimes\boldsymbol{M})=\mathrm{trace}(\boldsymbol{N})\mathrm{trace}(\boldsymbol{M})$
11. $(\boldsymbol{I}_m\otimes\boldsymbol{N})(\boldsymbol{M}\otimes\boldsymbol{I}_n)=(\boldsymbol{M}\otimes\boldsymbol{I}_n)(\boldsymbol{I}_m\otimes\boldsymbol{N})$
12. $f(\boldsymbol{N}\otimes\boldsymbol{I}_m)=f(\boldsymbol{N})\otimes\boldsymbol{I}_m$，其中$f(\cdot)$是一个分析函数
13. $f(\boldsymbol{I}_m\otimes\boldsymbol{N})=\boldsymbol{I}_m\otimes f(\boldsymbol{N})$
14. $\mathrm{e}^{\boldsymbol{N}\otimes\boldsymbol{M}}=\mathrm{e}^{\boldsymbol{N}}\otimes\mathrm{e}^{\boldsymbol{M}}$
15. $(\boldsymbol{I}_p\otimes z)\boldsymbol{A}=\boldsymbol{A}\otimes z$

16. $\boldsymbol{A}(\boldsymbol{I}_q \otimes \boldsymbol{z}^T) = \boldsymbol{A} \otimes \boldsymbol{z}^T$

17. $\text{vec}(\boldsymbol{A} + \boldsymbol{H}) = \text{vec}(\boldsymbol{A}) + \text{vec}(\boldsymbol{H})$

18. $\text{vec}(\boldsymbol{ADB}) = (\boldsymbol{B}^T \otimes \boldsymbol{A})\text{vec}(\boldsymbol{D})$

19. $\text{vec}(\boldsymbol{AD}) = (\boldsymbol{I}_s \otimes \boldsymbol{A})\text{vec}(\boldsymbol{D}) = (\boldsymbol{D}^T \otimes \boldsymbol{I}_p)\text{vec}(\boldsymbol{A}) = (\boldsymbol{D}^T \otimes \boldsymbol{A})\text{vec}(\boldsymbol{I}_q)$

20. $\text{vec}(\boldsymbol{AD}) = \sum_{k=1}^{q} \boldsymbol{D}_{:k}^T \otimes \boldsymbol{A}_{:k}$

21. $\boldsymbol{A}^{[2]} = \boldsymbol{A} \otimes \boldsymbol{A}$（克罗内克平方）

22. $\boldsymbol{A}^{[k+1]} = \boldsymbol{A} \otimes \boldsymbol{A}^{[k]}$

23. $(\boldsymbol{AD})^{[k]} = \boldsymbol{A}^{[k]}\boldsymbol{D}^{[k]}$

24. $\boldsymbol{M} \otimes \boldsymbol{A} = (\boldsymbol{U}_1 \otimes \boldsymbol{U}_2)(\boldsymbol{S}_1 \otimes \boldsymbol{S}_2)(\boldsymbol{V}_1 \otimes \boldsymbol{V}_2)^T$，其中$\boldsymbol{M}$和$\boldsymbol{A}$有SVD
$\boldsymbol{M} = \boldsymbol{U}_1\boldsymbol{S}_1\boldsymbol{V}_1^T$ 和 $\boldsymbol{A} = \boldsymbol{U}_2\boldsymbol{S}_2\boldsymbol{V}_2^T$

25. $\text{trace}(\boldsymbol{ADW}) = [\text{vec}(\boldsymbol{A}^T)]^T(\boldsymbol{I}_p \otimes \boldsymbol{D})\text{vec}(\boldsymbol{W})$

26. $\text{trace}(\boldsymbol{A}^T\boldsymbol{H}) = [\text{vec}(\boldsymbol{A})]^T\text{vec}(\boldsymbol{H})$

27. $\boldsymbol{A} \odot (\boldsymbol{D}^T \odot \boldsymbol{F}^T) = (\boldsymbol{A} \odot \boldsymbol{D}^T) \odot \boldsymbol{F}^T$

28. $(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{F} \odot \boldsymbol{G}) = \boldsymbol{AF} \odot \boldsymbol{BG}$

29. $\text{vec}(\boldsymbol{AVD}) = (\boldsymbol{D}^T \odot \boldsymbol{A})\text{vecd}(\boldsymbol{V})$，如果$\boldsymbol{V} \in \Re^{q \times q}$是对角阵

30. $\boldsymbol{N} \oplus \boldsymbol{N} = \boldsymbol{A} \otimes \boldsymbol{I}_n + \boldsymbol{I}_n \otimes \boldsymbol{A}$

31. 假若$\lambda_i$, $i = 1, 2, \cdots, n$是$\boldsymbol{N}$的特征值相应于特征向量$\boldsymbol{z}_i$; $\beta_j$, $j = 1, 2, \cdots, m$是$\boldsymbol{M}$的特征值相应于特征向量$\boldsymbol{y}_j$，那么：$\boldsymbol{\lambda}(\boldsymbol{N} \otimes \boldsymbol{M}) = \boldsymbol{\lambda}(\boldsymbol{N}) \otimes \boldsymbol{\lambda}(\boldsymbol{M})$，其中$\boldsymbol{N}$和$\boldsymbol{M}$是非奇异的。即，两个非奇异矩阵的克罗内克积的特征值由各自矩阵的特征值（包含在特征向量$\boldsymbol{\lambda}(\boldsymbol{N})$和$\boldsymbol{\lambda}(\boldsymbol{M})$）的克罗内克积给出，其中向量的维数为$\boldsymbol{\lambda}(\boldsymbol{N} \otimes \boldsymbol{M})_{nm \times 1}$、$\boldsymbol{\lambda}(\boldsymbol{N})_{n \times 1}$和$\boldsymbol{\lambda}(\boldsymbol{M})_{m \times 1}$。另一种可以选择的表达方式是$\boldsymbol{N} \otimes \boldsymbol{M}$的特征值为$mn$个$\lambda_i\beta_j$，其中$i = 1, 2, \cdots, n$, $j = 1, 2, \cdots, m$。 582

32. $\boldsymbol{N} \oplus \boldsymbol{M} = (\boldsymbol{N} \otimes \boldsymbol{I}_m) + (\boldsymbol{I}_n \otimes \boldsymbol{M})$的特征值为$mn$个$\lambda_i + \beta_j$，其中$i = 1, 2, \cdots, n$, $j = 1, 2, \cdots, m$，（见性质31）。

33. $\boldsymbol{N} \otimes \boldsymbol{M}$和$\boldsymbol{N} \oplus \boldsymbol{M}$特征向量分别相应于特征值$\lambda_i\beta_j$和$\lambda_i + \beta_j$，它们是$\boldsymbol{z}_i \otimes \boldsymbol{y}_j$, $i = 1, 2, \cdots, n$, $j = 1, 2, \cdots, m$（见性质31）。

上面性质中用到的矩阵的维数和向量如下：

| | | | |
|---|---|---|---|
| $\boldsymbol{A} \in \Re^{p \times q}$ | $\boldsymbol{D} \in \Re^{q \times s}$ | $\boldsymbol{H} \in \Re^{p \times q}$ | $\boldsymbol{R} \in \Re^{s \times t}$ |
| $\boldsymbol{B} \in \Re^{s \times t}$ | $\boldsymbol{F} \in \Re^{q \times u}$ | $\boldsymbol{M} \in \Re^{m \times m}$ | $\boldsymbol{W} \in \Re^{s \times p}$ |
| $\boldsymbol{C} \in \Re^{r \times 1}$ | $\boldsymbol{G} \in \Re^{t \times u}$ | $\boldsymbol{N} \in \Re^{n \times n}$ | $\boldsymbol{y} \in \Re^{m \times 1}$ |
| | | | $\boldsymbol{z} \in \Re^{n \times 1}$ |

著名的李雅普诺夫方程[6]为：

$$\boldsymbol{AX} + \boldsymbol{XA}^T = -\boldsymbol{C} \tag{A-144}$$

其中$\boldsymbol{A} \in \Re^{n \times n}$，$\boldsymbol{C} \in \Re^{n \times n}$（$\boldsymbol{C}^T = \boldsymbol{C}$且$\boldsymbol{C} \geqslant 0$），解$\boldsymbol{X} \in \Re^{n \times n}$（$\boldsymbol{X}^T = \boldsymbol{X}$且$\boldsymbol{X} \geqslant 0$）。用线性代数的标准方法，这个方程不能直接解出$\boldsymbol{X}$。然而，如果先在式（A-144）两边构造矩阵向量，

$$\text{vec}(\boldsymbol{AX} + \boldsymbol{XA}^T) = -\text{vec}(\boldsymbol{C}) \tag{A-145}$$

应用性质19和式（A-139），可以把式（A-145）写作：

$$(\boldsymbol{A} \oplus \boldsymbol{A})\text{vec}(\boldsymbol{X}) = -\text{vec}(\boldsymbol{C}) \tag{A-146}$$

为解出$\text{vec}(\boldsymbol{X})$，$\boldsymbol{A} \oplus \boldsymbol{A}$必须是可逆的，即，

$$\text{vec}(\boldsymbol{X}) = -(\boldsymbol{A} \oplus \boldsymbol{A})^{-1}\text{vec}(\boldsymbol{C}) \tag{A-147}$$

所以，$A \oplus A$不能有零特征值。因此，使用性质32，可以得如下结论：式（A-144）的唯一解存在的充分必要条件是$\lambda_i(A) + \lambda_j(A) \neq 0$，其中$i, j = 1, 2, \cdots, n$。因此，如果$A$是渐近稳定（参照A.2.12节），式（A-144）存在唯一解。然而，若$A$是边缘稳定，不存在唯一解。同样，若$A$是不稳定的，有关于复平面$j\omega$轴对称的复共轭对（或严格的实数）特征值，则式（A-144）没有唯一解。根据上面相同的方法，很简单就可证明：若$\lambda_i(A) + \lambda_j(A) \neq 0, i, j = 1, 2, \cdots, n$，（其中$\bar{\lambda}$是$\lambda$的复共轭），$A \in \mathscr{C}^{n \times n}, C \in \mathscr{C}^{n \times n}$，式（A-144）有唯一解$X \in \mathscr{C}^{n \times n}$。

### A.2.18　实数和复数方阵的重要性质小结

583

下面是对实数和复数方阵的重要性质小结。

实矩阵

如果$A \in \Re^{n \times n}(A = [a_{ij}]$，对于$i, j = 1, 2, \cdots, n)$，那么：

1. $A$为对称$\Leftrightarrow A^T = A$
2. $A$为斜对称$\Leftrightarrow A^T = -A$
3. $A$为对角$\Leftrightarrow a_{ij} = 0, \forall i \neq j$
4. $A$为正规$\Leftrightarrow AA^T = A^TA$
5. $A$为正定$\Leftrightarrow A^T = A$且$x^TAx > 0, x \in \Re^{n \times 1}$且$x \neq 0$
6. $A$为半正定（非负定）$\Leftrightarrow A^T = A$且$x^TAx \geqslant 0, x \in \Re^{n \times 1}$且$x \neq 0$
7. $A$为正交$\Leftrightarrow A^TA = AA^T = I$
8. $A$为对合$\Leftrightarrow A^2 = I$
9. $A$为斜对合$\Leftrightarrow A^2 = -I$
10. $A$为幂等$\Leftrightarrow A^2 = A$
11. $A$为三幂等$\Leftrightarrow A^3 = A$
12. $A$为幂零$\Leftrightarrow A^k = 0$，其中某个$k > 0$
13. $A$为上三角$\Leftrightarrow a^{ij} = 0, \forall i > j$
14. $A$为下三角$\Leftrightarrow a^{ij} = 0, \forall i < j$

如果$A \in \Re^{2n \times 2n}, J_n \in \Re^{2n \times 2n}$定义为

$$J_n \triangleq \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}$$

则：

1. $A$是哈密顿的$\Leftrightarrow J^{-1}A^TJ = -A \Leftrightarrow (JA)^T = JA$
2. $A$是偶对的$\Leftrightarrow J^{-1}A^TJ = A^{-1} \Leftrightarrow A^TJA = J$

如果$A, B \in \Re^{n \times n}$是合同的，则：

1. $\rho(A) = \rho(B)$
2. $A^T$和$B^T$全等
3. $A$可逆$\Leftrightarrow B$可逆$\Leftrightarrow A^{-1}$和$B^{-1}$全等
4. $A$对称、斜对称、正定、半正定$\Leftrightarrow B$对称、斜对称、正定、半正定。

如果$A, B \in \Re^{n \times n}$是正交相似，那么$A$是对称的、斜对称的、正定的、半正定的、正规的、正交的、对合的、斜对合的、幂等的、三幂等的和幂零的当且仅当$B$是对称的、斜对称的、正定的、半正定的、正则的、正交的、对合的、斜对合的、幂等的、三幂等的和幂零的。

复矩阵

如果$A \in \mathscr{C}^{n \times n}$，那么

1. $A$是正规的$\Leftrightarrow AA^* = A^*A$
2. $A$是埃尔米特的$\Leftrightarrow A^* = A$
3. $A$是斜埃尔米特的$\Leftrightarrow A^* = -A$
4. $A$是正定的$\Leftrightarrow A^* = A$和$x^*Ax > 0$，对于$x \in \mathscr{C}^{n\times 1}$
5. $A$是半正定的$\Leftrightarrow A^* = A$和$x^*Ax \geqslant 0$，对于$x \in \mathscr{C}^{n\times 1}$ 584
6. $A$是单元的$\Leftrightarrow A^*A = AA^* = I$

### A.2.19 模式化矩阵和特殊矩阵

循环矩阵

循环矩阵是一个方阵。每一行从第二个元素开始，把上一行的每个元素向右移一列，第一行第一个元素最终移动到最后一行最后一个元素得到该方阵。循环矩阵的一般形式如下：

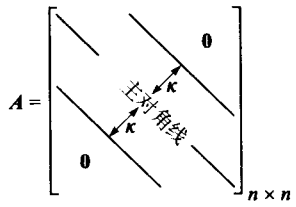
$$A = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & \cdots & a_n \\ a_n & a_1 & a_2 & a_3 & \cdots & a_{n-1} \\ a_{n-1} & a_n & a_1 & a_2 & \cdots & a_{n-2} \\ a_{n-2} & a_{n-1} & a_n & a_1 & \cdots & a_{n-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_2 & a_3 & a_4 & a_5 & \cdots & a_1 \end{bmatrix} \tag{A-148}$$

循环矩阵的性质如下：

1. 循环矩阵$A_{n\times n}$的特征值如下：$\lambda_i = a_1 + a_2 r_i + a_3 r_i^2 + \cdots + a_n r_i^{n-1}$，其中$i = 1, 2, \cdots, n$，$[a_1, a_2, a_3, \cdots, a_n]$是$A$的第一行，$r_i$是$r^n = 1$的一个不同解。相应的特征向量为$v_i = [1, r_i, r_i^2, \cdots, r_i^{n-1}]^T$。
2. 如果$A$和$B$是阶数相同的循环矩阵，$\alpha, \beta$是两个数量，那么，$\alpha A + \beta B$也是循环矩阵。
3. 若循环矩阵是非奇异的，则它的逆是循环矩阵。
4. 阶数相同的循环矩阵$A$和$B$的积也是循环矩阵，积满足交换律，即$AB = BA$。

带状矩阵

包含元素$[a_{ij}]$的方阵$A_{n\times n}$，$i, j = 1, 2, \cdots, n$，当$|i-j| > \kappa$，$\kappa$是非负整数$0 \leqslant \kappa \leqslant n-1$，如果$a_{ij} = 0$，称矩阵$A$为宽度为$2\kappa + 1$的带状矩阵。带状矩阵的一般形式形如图A-2。带状矩阵的所有非零元素全部位于主对角线和$\kappa$条上对角线，$\kappa$条下对角线。$\kappa = 0$的带状矩阵是（严格）对角矩阵。$n \times n$维的宽度为$2\kappa + 1$的带状矩阵的和、积和转置矩阵仍旧是同样宽度的带状矩阵。



图A-2 带状矩阵的一般形式 585

特普利茨（Toeplitz）矩阵是带状矩阵，每个对角线由相同的元素构成。然而，不同的对角线上含有不同的元素。每一个非零的循环矩阵是一个满宽度的特普利茨矩阵。特普利茨矩阵的一个例子，也是循环矩阵：

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 2 & 3 & 4 \\ 4 & 5 & 1 & 2 & 3 \\ 3 & 4 & 5 & 1 & 2 \\ 2 & 3 & 4 & 5 & 1 \end{bmatrix}$$

三对角矩阵

三对角矩阵是一个宽度只有3的带状矩阵（$\kappa = 1$）。因此，非零元素只在主对角线、上一条对角线和下一条对角线（矩阵中的所有其他元素为零）。如果我们假设一个$n \times n$阶三对角特普利茨矩阵，即$\boldsymbol{T}_{n\times n}$，主对角线上的元素为$\alpha$，上一条对角线的元素为$\beta$，下一条对角线的元素为$\gamma$，$\boldsymbol{T}$的特征值为：

$$\lambda_k = \alpha + 2\sqrt{\beta\gamma}\cos\left(\frac{k\pi}{n+1}\right) \tag{A-149}$$

其中$k = 1, 2, \cdots, n$。

海森伯格（Hessenberg）型

若方阵的下对角线以下的元素均为零，该矩阵拥有海森伯格型。每一个实方阵$\boldsymbol{A} \in \Re^{n\times n}$与海森伯格型的矩阵相合。具有海森伯格型的矩阵的一个例子：

$$\boldsymbol{A} = \begin{bmatrix} 2 & -4 & 0 & 0 & 0 \\ 3 & 2 & -4 & 0 & 0 \\ 0 & 3 & 2 & -4 & 0 \\ 0 & 0 & 3 & 2 & -4 \\ 0 & 0 & 0 & 3 & 2 \end{bmatrix}$$

这个矩阵也是一个宽度为3($\kappa = 1$)的带状矩阵、三对角阵和特普利茨矩阵。

希尔伯特矩阵

希尔伯特矩阵是对称的方阵$\boldsymbol{H} \in \Re^{n\times n}(\boldsymbol{H} = [h_{ij}], i, j = 1, 2, \cdots, n)$，其元素为$h_{ij} = 1/(i + j - 1)$。这是一个典型的病态矩阵的例子。希尔伯特矩阵是柯西矩阵的一个特例[7]。

汉克尔（Hankel）矩阵

汉克尔矩阵$\boldsymbol{H} = [h_{ij}], i, j = 1, 2, \cdots, n$是对称的，在反对角线上有相同的元素。它的基本形
586 式：汉克尔矩阵的元素完全由定义的向量$\upsilon = [\upsilon_k]$决定，$k = 1, 2, \cdots, n$，即

$$h_{ij} = \upsilon_{i+j-1} \tag{A-150}$$

由于向量$\upsilon$的标定指数比其长度要大（即>$n$，式（A-150)），所以$\boldsymbol{H}$的元素设置为0。

考虑一个适当的有理变换函数（参照A.2.12节）：

$$H(s) = \frac{\beta_0 s^n + \beta_1 s^{n-1} + \beta_2 s^{n-2} + \cdots + \beta_{n-2} s^2 + \beta_{n-1} s^1 + \beta_n}{s^n + \alpha_1 s^{n-1} + \alpha_2 s^{n-2} + \cdots + \alpha_{n-2} s^2 + \alpha_{n-1} s^1 + \alpha_n} \tag{A-151}$$

扩展到变量$s$的无限幂级数：

$$H(s) = h(0) + h(1)s^{-1} + h(2)s^{-2} + h(3)s^{-3} + \cdots \tag{A-152}$$

式（A-152）的系数$h(i), i = 0, 1, 2, 3, \cdots$称作马尔可夫参数，可以递归地写作：

$$\begin{aligned} h(0) &= \beta_0 \\ h(1) &= -\alpha_1 h(0) + \beta_1 \\ h(2) &= -\alpha_1 h(1) - \alpha_2 h(0) + \beta_2 \\ &\vdots \\ h(n) &= -\alpha_1 h(n-1) - \alpha_2 h(n-2) - \cdots - \alpha_n h(0) + \beta_n \\ h(n+i) &= -\alpha_1 h(n+i-1) - \alpha_2 h(n+i-2) - \cdots - \alpha_n h(i) \end{aligned} \tag{A-153}$$

其中$i = 1, 2, 3, \cdots$。矩阵$\boldsymbol{H}(\alpha, \beta)$（$\alpha \times \beta$阶汉克尔矩阵）可以由马尔可夫参数$h(i), i = 1, 2, 3 \cdots$构造

如下:

$$\boldsymbol{H}(\alpha,\beta)=\begin{bmatrix} h(1) & h(2) & h(3) & \cdots & h(\beta) \\ h(2) & h(3) & h(4) & \cdots & h(\beta+1) \\ h(3) & h(4) & h(5) & \cdots & h(\beta+2) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ h(\alpha) & h(\alpha+1) & h(\alpha+2) & \cdots & h(\alpha+\beta-1) \end{bmatrix} \tag{A-154}$$

[注意$h(0)$不能直接用于构造$\boldsymbol{H}(\alpha, \beta)$]。式（A-154）中的汉克尔矩阵用于控制理论找不可约简的式（A-151）给出的转换实现函数。

范德蒙德矩阵

范德蒙德矩阵是方阵$\boldsymbol{V}_{n\times n}$，它的列是伴随矩阵的特征向量或弗罗贝尼乌斯矩阵（参照A.2.12节）。范德蒙德矩阵的一般形式：

$$\boldsymbol{V}=\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ \lambda_1 & \lambda_2 & \lambda_3 & \cdots & \lambda_n \\ \lambda_1^2 & \lambda_2^2 & \lambda_3^2 & \cdots & \lambda_n^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \lambda_1^{n-1} & \lambda_2^{n-1} & \lambda_3^{n-1} & \cdots & \lambda_n^{n-1} \end{bmatrix} \tag{A-155}$$

其中$\lambda_i$，$i=1, 2, \cdots, n$是伴随矩阵的$n$个特征值。范德蒙德矩阵的行列式由下式给出： 587

$$\det(\boldsymbol{V})=(\lambda_2-\lambda_1)(\lambda_3-\lambda_2)(\lambda_3-\lambda_1)(\lambda_4-\lambda_3)(\lambda_4-\lambda_2)(\lambda_4-\lambda_1)\cdots(\lambda_n-\lambda_1) \tag{A-156}$$

如果伴随矩阵的特征值是不同的，我们从（A-156）看出范德蒙德矩阵的行列式是非零的。

阿达马（Hadamard）矩阵

阿达马矩阵[8]是一个方阵$\boldsymbol{H}\in\Re^{n\times n}(\boldsymbol{H}=[h_{ij}], i, j=1, 2, \cdots, n)$，其元素$h_{ij}=\pm 1$，$\boldsymbol{H}$的行之间是相互正交的。对于特定的$n$才存在阿达马矩阵。具体说，它们存在的必要条件是$n>2$，那么$n$必须是4的倍数。对于阿达马矩阵，我们有$\boldsymbol{H}^T\boldsymbol{H}=\boldsymbol{H}\boldsymbol{H}^T=n\boldsymbol{I}$。因此，它满足：$\boldsymbol{H}^{-1}=n^{-1}\boldsymbol{H}^T$。

惯性矩阵

如果存在非奇异矩阵$\boldsymbol{Q}$，满足：

$$\boldsymbol{A}=\boldsymbol{Q}\boldsymbol{B}\boldsymbol{Q}^T(\boldsymbol{A}=\boldsymbol{Q}\boldsymbol{B}\boldsymbol{Q}^H) \tag{A-157}$$

那么，方阵$\boldsymbol{A}$和相同维数的方阵$\boldsymbol{B}$是合同的（埃尔米特合同或合取的）。每一个$n\times n$的秩为$r$的埃尔米特矩阵与唯一的一个分块矩阵合同：

$$\boldsymbol{U}=\begin{bmatrix} \boldsymbol{I}_p & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_q & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \tag{A-158}$$

即为惯性矩阵。

惯性的西尔维斯特定律：两个埃尔米特矩阵是合同的当且仅当相合于相同的惯性矩阵，而且它们都有$p$个正特征值，$q$个负特征值，$\gamma=n-p-q$个零特征值（$p$是$\boldsymbol{A}$的指数，$s=p-q$是$\boldsymbol{A}$的符号差）。

帕斯卡（Pascal）矩阵

帕斯卡矩阵是正定对称测试矩阵，元素为整数，由帕斯卡三角构成。帕斯卡矩阵的有趣的性质是：它的逆元素也是整数。帕斯卡矩阵$\boldsymbol{P}_n\in\Re^{n\times n}$的元素定义如下:

$$p_{ij}=\frac{(i+j-2)!}{(i-1)!(j-1)!}=\binom{i+j-2}{j-1} \tag{A-159}$$

在MATLAB中，函数pascal(n)将生成一个$n$维（方）帕斯卡阵，其元素由式（A-159）所定义。MATLAB中的pascal(n,1)函数将生成一个数和一个转置的楚列斯基因子$\boldsymbol{S}$（下三
588 角矩阵），它是对合的，即它是其本身的逆（$\boldsymbol{S}^2=\boldsymbol{I}$）。

Kahan矩阵

一个*Kahan*矩阵是一个$n$阶参数为θ，上三角矩阵$\boldsymbol{K}_n(\theta)\in\Re^{n\times n}$，定义如下：

$$\boldsymbol{K}_n(\theta)=\mathrm{diag}[1,s,s^2,\cdots,s^{n-1}]\begin{vmatrix}1 & c & c & \cdots & c\\ 0 & 1 & c & \cdots & c\\ 0 & 0 & 1 & \cdots & c\\ \cdots & \cdots & \cdots & \cdots & \cdots\\ 0 & 0 & 0 & \cdots & 1\end{vmatrix} \tag{A-160}$$

其中$s=\sin(\theta)$, $c=\cos(\theta)$。对于不同阶的矩阵和不同的角θ，可以改变条件构成不同的矩阵。一般，角度θ越小，矩阵的病态程度越高。

## A.3 多变量分析的原理

### A.3.1 集合和函数

集合

考虑$\sigma$是一个数或集合$\sum$的元素，即$\sigma\in\sum$。如果元素不属于集合$\sum$，我们记作$\sigma\notin\sum$。一般我们把集合的元素写在一对括号里，例如：$\sum=\{-1,0,1\}$，或正整数集合$\Omega=\{1,2,3,\cdots,n,\cdots\}$。我们可以根据特定性质定义一个集合；例如，我们可以说集合$A$是所有$\sum$中的拥有性质$P$的元素$\sigma$，即，

$$A=\{\sigma\in\sum:P(\sigma)\} \tag{A-161}$$

或可以简写作：

$$A=\{\sigma:P(\sigma)\} \tag{A-162}$$

其中暗指在$\sum$集合。虽然我们一般认为集合具有一些元素，但是有（且只有）一个集合不含任何元素，称为空集∅。若集合$A$的每个元素$\sigma$在集合$B$中，即$\sigma\in A\Rightarrow\sigma\in B$，则称$A$是$B$的子集（或$A$包含于$B$），写作$A\subset B$。若$A\subset B$且$B\subset A$，则$B=A$。此外，对于任一集合$A$，空集∅的元素也是$A$的元素（因为空集里没有任何元素），写作$\varnothing\subset A$。因此，空集是任一集合的子集。集
589 合$C$的元素或者属于$A$或者属于$B$，称为$A$与$B$的并集，记作：

$$A\cup B=C=\{\sigma:\sigma\in A\ \vee\ \sigma\in B\} \tag{A-163}$$

若$A$和$B$是$C$的子集，我们定义它们的交集为既属于$A$又属于$B$的元素的集合，写作：

$$A\cap B=C=\{\sigma:\sigma\in A\ \wedge\ \sigma\in B\} \tag{A-164}$$

若$A$是$B$的子集，则$A$的补集$\tilde{A}$（相对于$B$），是不属于$A$的元素的集合。写作：

$$\tilde{A}=\{\sigma\in B:\sigma\notin A\} \tag{A-165}$$

它满足：$\tilde{\tilde{A}}=A, A\cup\tilde{A}=B, A\cap\tilde{A}=\varnothing$。德摩根定律为：

$$\widetilde{A\cup B}=\tilde{A}\cap\tilde{B}\quad\text{和}\quad\widetilde{A\cap B}=\tilde{A}\cup\tilde{B} \tag{A-166}$$

若$A$是实数集$B$的子集，即$A \subset B$，则集合$B$中最小的元素，又大于或等于$A$中的所有元素，称为$A$的最小上界或上确界（sup）。我们把$A$的上确界记作：

$$\sup A \quad 或 \quad \sup_{\sigma \in A} \sigma \quad 或 \quad \sup\{\sigma : \sigma \in A\} \tag{A-167}$$

相反，实数集$A$的最大下界也称为$A$的下确界（inf），是$A$的下界中最大的，记作：

$$\inf A \quad 或 \quad \inf_{\sigma \in A} \sigma \quad 或 \quad \inf\{\sigma : \sigma \in A\} \tag{A-168}$$

注意：$\inf_{\sigma \in A} \sigma = -\sup_{\sigma \in A} -\sigma$。

函数

函数本质上讲是一个约束设定规则。即，从集合$X$到集合$Y$的函数$f$是一种规则，其对于集合$X$中的每一个$x$都有$Y$中的唯一的一个$f(x)$与之对应。（笛卡儿积）$X \times Y$中的有序对$\langle x, f(x)\rangle$的集合$\mathfrak{G}$称作函数$f$的图。大多数情况下函数定义为图。此外，名词映射经常用作函数的同义词。表示$X$到$Y$的函数$f$为：$f: X \to Y$。集合$X$称作$f$的定义域，$f$取的值的集合称作$f$的值域。函数的值域一般小于$Y$。然而，如果$f$的值域就是$Y$，那么说$f$是映射到$Y$上的一个函数（或$f$是满射的）。

如果假设$A \subset X$，那么在$A$的函数$f$下的像定义为$Y$中元素的集合，即对于$x \in A$，有$y = f(x)$。像记作$f[A]$。这样，当且仅当$Y = f[X]$，$f$的值域为$f[X]$，$f$是映射到$Y$的。现在，若$B \subset Y$，我们定义$B$的逆像$f^{-1}[B]$为当$f(x)$在$B$中时$x \in X$的集合。注意，当且仅当$Y$的每个非空子集的逆像非空时$f$映射到$Y$。函数$f: X \to Y$称作一对一或内射，或单一映射，如果$f(x_1) = f(x_2)$只在$x_1 = x_2$时成立。如果 590
函数是一对一的从$X$映射到$Y$，函数称为$X$与$Y$一一对应的（或双射）。在这种情况下，有一个函数$g: Y \to X$，对任意$x$和$y$有$g(f(x)) = x$和$f(g(y)) = y$。函数$g$称作函数$f$的逆，记作$f^{-1}$。若$f: X \to Y$，$g: Y \to Z$，我们将定义一个新函数$h: X \to Z$，写作$h(x) = g(f(x))$。函数$h$称为$g$与$f$的复合函数记作$g \circ f$。

在神经计算的研究中多个变量的函数是非常重要的。如果假设向量$\boldsymbol{x} \in \Re^{n \times 1}$（其中$\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T$），对于$\mathscr{X}$特定向量集$\mathscr{X}$，函数$f(\boldsymbol{x})$的最小值写作（参照A.5节）：

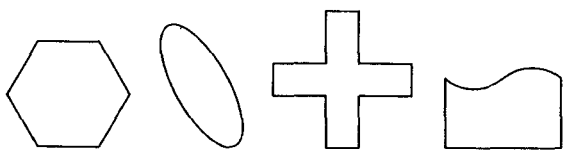
$$\min_{\boldsymbol{x} \in \mathscr{X}} f(\boldsymbol{x})$$

下面定义凸集和凸函数。

**定义A.10** 集合$\Sigma$是凸集，当任意元素$\boldsymbol{x}, \boldsymbol{y} \in \Sigma$，

$$\beta \boldsymbol{x} + (1-\beta)\boldsymbol{y} \in \Sigma \quad \forall 0 \leqslant \beta \leqslant 1$$

定义A.10简单地说如果$\boldsymbol{x}$, $\boldsymbol{y}$属于$\Sigma$，那么连接$\boldsymbol{x}$和$\boldsymbol{y}$的线段也在$\Sigma$中。每一个由线性约束系统定义的集合都是凸集。图A-3给出凸集和非凸集的例子。

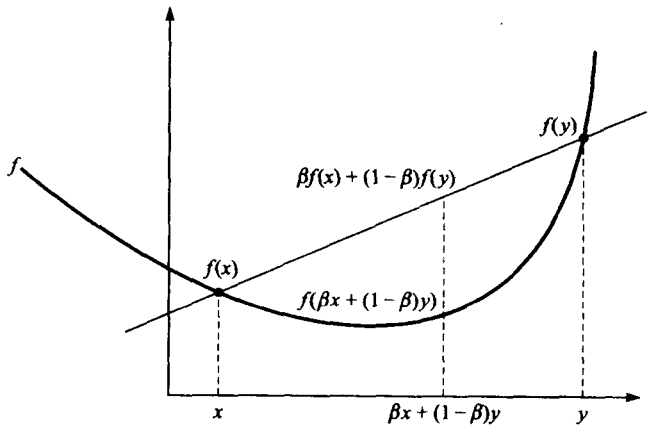



图A-3 a)和b)为凸集示例；c)和d)为非凸集示例

**定义A.11** 对每一个$\boldsymbol{x}, \boldsymbol{y} \in \Sigma$，$\forall \beta, (0 \leqslant \beta \leqslant 1)$，定义在凸集$\Sigma$上的函数$f$是一个凸函数，写作：$f(\beta \boldsymbol{x} + (1-\beta)\boldsymbol{y}) \leqslant \beta f(\boldsymbol{x}) + (1-\beta) f(\boldsymbol{y})$。此外，对于$\forall \beta, (0 < \beta < 1)$且$\boldsymbol{x} \neq \boldsymbol{y}$，函数$f$是严格凸函数，即：$f(\beta \boldsymbol{x} + (1-\beta)\boldsymbol{y}) < \beta f(\boldsymbol{x}) + (1-\beta) f(\boldsymbol{y})$。

图A-4给出了一个凸函数的例子。从几何上看，这意味着一个函数是凸函数，如果在它的图上画连接任意两点的直线均在图之上（没有在图像之下的）。直观上讲，函数的图像是碗状

的。注意，若函数是凸函数，它的负函数是凹的。




图A-4　凸函数示例

设$\boldsymbol{F}(\boldsymbol{x})$是$n$维向量$\boldsymbol{x}$的函数构成的$m$维向量，即$\boldsymbol{F}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})]^T$。当$\beta$为某个常量满足：

$$\|\boldsymbol{F}(\boldsymbol{x}) - \boldsymbol{F}(\boldsymbol{y})\| \leqslant \beta\|\boldsymbol{x} - \boldsymbol{y}\| \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \Sigma \tag{A-169}$$

$\boldsymbol{F}$在开集[⊖]$\Sigma$是利普希茨（Lipschitz）连续的。

### A.3.2　二次型

一个二次型写为：

$$q = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij} x_i x_j \tag{A-170}$$

$\boldsymbol{x} \in \Re^{n\times 1}$（其中$\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T$），在$p_{ij}$中，$i, j = 1, 2, \cdots, n$是实系数，$q \in \Re$。式（A-170）可以简写作向量矩阵形式：

$$q = \boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} \tag{A-171}$$

其中，$\boldsymbol{P} \in \Re^{n\times n}(\boldsymbol{P} = [p_{ij}])$（$\boldsymbol{P}$通常假定为对称的，$\boldsymbol{P}^T = \boldsymbol{P}$）。式（A-171）的二次表达式可以写作：

$$q = \text{trace}(\boldsymbol{P}\boldsymbol{x}\boldsymbol{x}^T) = \text{trace}(\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{P}) \tag{A-172}$$

对于$\boldsymbol{x} \in \mathcal{C}^{n\times 1}$和$\boldsymbol{P} \in \mathcal{C}^{n\times n}$，复二次型可以写作：

$$q = \boldsymbol{x}^H\boldsymbol{P}\boldsymbol{x} \tag{A-173}$$

591
ι
592

其中$\boldsymbol{P}$假定为埃尔米特，$\boldsymbol{P}^H = \boldsymbol{P}$。二次表达式可以写作$q = \langle \boldsymbol{P}\boldsymbol{x}, \boldsymbol{x}\rangle$（参照A.2.3节），称作欧几里得内积，若$\boldsymbol{P}^H = \boldsymbol{P}$，其为实数。关于二次型，参照A.2.6节对称矩阵的确定性限定。二次型的一个特例是$\boldsymbol{P}$是对角矩阵。在这种情况下，没有叉积项，即$P_{ij} = 0, \forall i \neq j$。

### A.3.3　链式法则

若$q(x) = u(x)v(x)$，则

---

⊖　开集此处是指一个开球面，其中心为$\boldsymbol{x}$，半径$\varepsilon > 0$，即$S(\boldsymbol{x}, \varepsilon)$。因此，式（A-169）也可以写作$\|\boldsymbol{F}(\boldsymbol{x}) - \boldsymbol{F}(\boldsymbol{y})\| \leqslant \beta\|\boldsymbol{x} - \boldsymbol{y}\|$，$\forall \boldsymbol{y} \in S(\boldsymbol{x}, \varepsilon)$。函数$\boldsymbol{F}$也可称为$\beta$秩在$\boldsymbol{x}$点是局部利普希茨连续的。

$$\frac{\mathrm{d}}{\mathrm{d}x}q(x)=\frac{\mathrm{d}}{\mathrm{d}x}(uv)=u\frac{\mathrm{d}v}{\mathrm{d}x}+v\frac{\mathrm{d}u}{\mathrm{d}x} \tag{A-174}$$

若$q(x)=(u(x)/v(x))(v\neq 0)$，则

$$\frac{\mathrm{d}}{\mathrm{d}x}q(x)=\frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{u}{v}\right)=\frac{v\left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)-u\left(\frac{\mathrm{d}v}{\mathrm{d}x}\right)}{v^2} \tag{A-175}$$

若$q(x)=(c/u(x))(c\in\Re)$，则

$$\frac{\mathrm{d}}{\mathrm{d}x}q(x)=\frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{c}{u}\right)=c\frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{1}{u}\right)=-\frac{c}{u^2}\frac{\mathrm{d}}{\mathrm{d}x}(u) \tag{A-176}$$

若$q(x)=u^n(x)$，则

$$\frac{\mathrm{d}}{\mathrm{d}x}q(x)=\frac{\mathrm{d}}{\mathrm{d}x}(u^n)=nu^{n-1}\frac{\mathrm{d}}{\mathrm{d}x}(u) \tag{A-177}$$

假设$y=f(u)$, $u=g(x)$。可以把$y$写作一个函数的函数$y=f(g(x))$。若$y$是$u$的可微函数，$u$是$x$的可微函数，则$y=f(g(x))$是$x$的可微函数。写作：

$$\frac{\mathrm{d}y}{\mathrm{d}x}=\frac{\mathrm{d}y}{\mathrm{d}u}\frac{\mathrm{d}u}{\mathrm{d}x} \tag{A-178}$$

若$z=f(x,y)$是变量$x$和$y$的连续函数且有连续的偏微分$\partial z/\partial x$和$\partial z/\partial y$，$x$，$y$是变量$t$的可微函数，$x=g(t)$, $y=h(t)$，那么$z$也是变量$t$的函数$\mathrm{d}z/\mathrm{d}t$（称为$z$对于$t$的全微分）如下：

$$\frac{\mathrm{d}z}{\mathrm{d}t}=\frac{\partial z}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t}+\frac{\partial z}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t} \tag{A-179}$$

若$z=f(x,y)$是变量$x$和$y$的连续函数，且有连续的偏微分$\partial z/\partial x$和$\partial z/\partial y$，$x$和$y$是独立变量$r$和$s$的连续函数$x=g(r,s)$和$y=h(r,s)$，那么$z$是$r$和$s$的函数且：

$$\frac{\partial z}{\partial r}=\frac{\partial z}{\partial x}\frac{\partial x}{\partial r}+\frac{\partial z}{\partial y}\frac{\partial y}{\partial r}\text{和}\frac{\partial z}{\partial s}=\frac{\partial z}{\partial x}\frac{\partial x}{\partial s}+\frac{\partial z}{\partial y}\frac{\partial y}{\partial s} \tag{A-180}$$

假设

$$f(\boldsymbol{x})=g(\boldsymbol{x})h(\boldsymbol{x}) \tag{A-181}$$ 593

其中$g$和$h$是向量$\boldsymbol{x}\in\Re^{n\times 1}(\boldsymbol{x}=[x_1,x_2,\cdots,x_n]^T)$的连续可微标量函数。那么，

$$\nabla_x f(\boldsymbol{x})=\nabla_x g(\boldsymbol{x})h(\boldsymbol{x})+\nabla_x h(\boldsymbol{x})g(\boldsymbol{x}) \tag{A-182}$$

其中

$$\nabla_x f(\boldsymbol{x})\triangleq\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}=\left[\frac{\partial f}{\partial x_1},\frac{\partial f}{\partial x_2},\cdots,\frac{\partial f}{\partial x_n}\right]^T \tag{A-183}$$

是$f$对应于$\boldsymbol{x}$的梯度。

### A.3.4 矩阵微积分

#### A.3.4.1 关于向量的标量函数微分

假设$\boldsymbol{x},\boldsymbol{y}\in\Re^{n\times 1}$, $\boldsymbol{P}\in\Re^{n\times n}$，且$\boldsymbol{A}\in\Re^{m\times n}$，则

1. $\dfrac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x}^T\boldsymbol{y})=\boldsymbol{y}$

2. $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{y}^T\boldsymbol{x}) = \boldsymbol{y}$

3. $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x}^T\boldsymbol{x}) = 2\boldsymbol{x}$

4. $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{P}\boldsymbol{x}) = \boldsymbol{P}^T$

5. $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{y}) = \boldsymbol{P}\boldsymbol{y}$

6. $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{y}^T\boldsymbol{P}\boldsymbol{x}) = (\boldsymbol{y}^T\boldsymbol{P})^T = \boldsymbol{P}^T\boldsymbol{y}$

7. $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{y}) = \frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{y}^T\boldsymbol{P}\boldsymbol{x}) = \boldsymbol{P}\boldsymbol{y}$, 对于 $\boldsymbol{P}^T = \boldsymbol{P}$

8. $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x}) = \boldsymbol{P}\boldsymbol{x} + \boldsymbol{P}^T\boldsymbol{x}$

9. $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x}) = 2\boldsymbol{P}\boldsymbol{x}$, 对于 $\boldsymbol{P}^T = \boldsymbol{P}$

10. $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x}-\boldsymbol{y})^T\boldsymbol{P}(\boldsymbol{x}-\boldsymbol{y}) = 2\boldsymbol{P}(\boldsymbol{x}-\boldsymbol{y})$, 对于 $\boldsymbol{P}^T = \boldsymbol{P}$

11. $\frac{\partial}{\partial \boldsymbol{x}^T}(\boldsymbol{A}\boldsymbol{x}) = \boldsymbol{A}$

12. $\frac{\partial}{\partial \boldsymbol{x}}\|\boldsymbol{x}\|_2 = \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_2}$

13. $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x}) = \boldsymbol{P}\boldsymbol{x} + \mathrm{vec}(\boldsymbol{x}^T\boldsymbol{P}) = \boldsymbol{P}\boldsymbol{x} + \boldsymbol{P}^T\boldsymbol{x}$

594 14. $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{A}\boldsymbol{x}) = (\boldsymbol{I}_n \otimes \boldsymbol{A})\mathrm{vec}(\boldsymbol{I}_n) = \mathrm{vec}(\boldsymbol{A})$

15. $\frac{\partial}{\partial \boldsymbol{x}^T}(\boldsymbol{x}\otimes\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{x}^T}(\boldsymbol{x}\odot\boldsymbol{x}) = \boldsymbol{I}_n\otimes\boldsymbol{x} + \boldsymbol{x}\otimes\boldsymbol{I}_n$

16. $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x}^T\odot\boldsymbol{A}) = \frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{A}\odot\boldsymbol{x}^T) = \boldsymbol{I}_n\odot\boldsymbol{A}$

A.3.4.2　关于矩阵的标量函数微分

对于适当维数的矩阵$\boldsymbol{A}$、$\boldsymbol{B}$和$\boldsymbol{C}$，可以得到

1. $\frac{\partial}{\partial \boldsymbol{A}}\mathrm{trace}(\boldsymbol{A}) = \boldsymbol{I}$

2. $\frac{\partial}{\partial \boldsymbol{A}}\mathrm{trace}(\boldsymbol{B}\boldsymbol{A}\boldsymbol{C}) = \boldsymbol{B}^T\boldsymbol{C}^T$

3. $\frac{\partial}{\partial \boldsymbol{A}}\mathrm{trace}(\boldsymbol{B}\boldsymbol{A}^T\boldsymbol{C}) = \boldsymbol{C}\boldsymbol{B}$

4. $\frac{\partial}{\partial \boldsymbol{A}}\mathrm{trace}(\boldsymbol{A}\boldsymbol{B}\boldsymbol{A}^T) = \boldsymbol{A}\boldsymbol{B}^T + \boldsymbol{A}\boldsymbol{B}$

5. $\frac{\partial}{\partial \boldsymbol{A}}\mathrm{trace}(\boldsymbol{A}\boldsymbol{B}\boldsymbol{A}) = \boldsymbol{A}^T\boldsymbol{B}^T + \boldsymbol{B}^T\boldsymbol{A}^T$

6. $\frac{\partial}{\partial \boldsymbol{A}}\mathrm{trace}(\boldsymbol{B}\boldsymbol{A}\boldsymbol{C}\boldsymbol{A}) = \boldsymbol{B}^T\boldsymbol{A}^T\boldsymbol{C}^T + \boldsymbol{C}^T\boldsymbol{A}^T\boldsymbol{B}^T$

7. $\frac{\partial}{\partial \boldsymbol{A}}\text{trace}(\boldsymbol{BACA}^T)=\boldsymbol{B}^T\boldsymbol{AC}^T+\boldsymbol{BAC}$

8. $\frac{\partial}{\partial \boldsymbol{A}}\text{trace}(\boldsymbol{A}^T\boldsymbol{A})=2\boldsymbol{A}$

9. $\frac{\partial}{\partial \boldsymbol{A}}\text{trace}(\boldsymbol{BA}^T\boldsymbol{AC})=\boldsymbol{ACB}+\boldsymbol{AB}^T\boldsymbol{C}^T$

10. $\frac{\partial}{\partial \boldsymbol{A}}\text{trace}(\boldsymbol{BAA}^T\boldsymbol{C})=\boldsymbol{B}^T\boldsymbol{C}^T\boldsymbol{A}+\boldsymbol{CBA}$

11. $\frac{\partial}{\partial \boldsymbol{A}}\text{trace}(\boldsymbol{BA}^T\boldsymbol{AB}^T)=2\boldsymbol{AB}^T\boldsymbol{B}$

12. $\frac{\partial}{\partial \boldsymbol{A}}\text{trace}(\boldsymbol{B}^T\boldsymbol{AA}^T\boldsymbol{B})=2\boldsymbol{BB}^T\boldsymbol{A}$

13. $$\frac{\partial}{\partial \boldsymbol{A}}\text{trace}\{\boldsymbol{B}(\boldsymbol{A}^T\boldsymbol{A})^2\boldsymbol{B}^T\}=\frac{\partial}{\partial \boldsymbol{A}}\text{trace}(\boldsymbol{BA}^T\boldsymbol{AA}^T\boldsymbol{AB}^T)$$
$$=2\boldsymbol{AA}^T\boldsymbol{AB}^T\boldsymbol{B}+2\boldsymbol{AB}^T\boldsymbol{BA}^T\boldsymbol{A}$$

14. $\frac{\partial}{\partial \boldsymbol{A}}\text{trace}(e^{\boldsymbol{A}})=e^{\boldsymbol{A}}$

15. $\frac{\partial}{\partial \boldsymbol{A}}|\boldsymbol{BAC}|=|\boldsymbol{BAC}|(\boldsymbol{A}^{-1})^T$

16. $\frac{\partial}{\partial \boldsymbol{A}}\text{trace}(\boldsymbol{A}^k)=k(\boldsymbol{A}^{k-1})^T$

595

17. $\frac{\partial}{\partial \boldsymbol{A}}\text{trace}(\boldsymbol{BA}^k)=\left(\sum_{i=0}^{k-1}\boldsymbol{A}^i\boldsymbol{BA}^{k-i-1}\right)^T$

18. $\frac{\partial}{\partial \boldsymbol{A}}\text{trace}(\boldsymbol{BA}^{-1}\boldsymbol{C})=-(\boldsymbol{A}^{-1}\boldsymbol{CBA}^{-1})^T$

19. $\frac{\partial}{\partial \boldsymbol{A}}\log|\boldsymbol{A}|=(\boldsymbol{A}^T)^{-1}$

20. $\frac{\partial}{\partial \boldsymbol{A}}|\boldsymbol{A}^T|=\frac{\partial}{\partial \boldsymbol{A}}|\boldsymbol{A}|=|\boldsymbol{A}|(\boldsymbol{A}^T)^{-1}$

21. $\frac{\partial}{\partial \boldsymbol{A}}|\boldsymbol{A}^k|=k|\boldsymbol{A}^k|(\boldsymbol{A}^T)^{-1}$

我们总结这一节，对标量$\tau\in\Re$（或$\tau\in\mathbb{C}$）的矩阵$\boldsymbol{A}(\tau)\in\Re^{m\times n}$［或$\boldsymbol{A}(\tau)\in\mathbb{C}^{m\times n}$］求微分

$$\frac{\mathrm{d}\boldsymbol{A}(\tau)}{\mathrm{d}\tau}\triangleq\left[\frac{\mathrm{d}a_{ij}(\tau)}{d\tau}\right]$$

其中$\alpha_{ij}(\tau)$, $i=1,2,\cdots,m$, $j=1,2,\cdots,n$，为$\boldsymbol{A}(\tau)$的元素。

### A.3.5 黑塞矩阵

若$f(\boldsymbol{x})$是向量$\boldsymbol{x}\in\Re^{n\times 1}$的实标量函数，在A.3.3节已定义$f(\boldsymbol{x})$关于$\boldsymbol{x}$的梯度，记作：$\nabla_{\boldsymbol{x}}f(\boldsymbol{x})=\partial f(\boldsymbol{x})/\partial \boldsymbol{x}$, $f:\Re^n\to\Re$。这里假定$f(\boldsymbol{x})$是$C^1$类的，即，$f(\boldsymbol{x})\in C^1$。若$f(\boldsymbol{x})$是$C^2$类的，对应于$\boldsymbol{x}$的黑塞矩阵$f(\boldsymbol{x})$定义为：

$$\nabla_x^2 f(\boldsymbol{x}) = \frac{\partial^2 f(\boldsymbol{x})}{\partial \boldsymbol{x} \partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_1} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_2} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_3} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_2} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_3} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_n} \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_3 \partial x_1} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_3 \partial x_2} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_3 \partial x_3} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_3 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_n \partial x_1} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_n \partial x_2} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_n \partial x_3} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_n \partial x_n} \end{bmatrix} = \left[ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j} \right] \tag{A-184}$$

其中$i, j = 1, 2, \cdots, n$。黑塞矩阵是对称的，即

$$\left[ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j} \right] = \left[ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_j \partial x_i} \right]$$

596 $f(\boldsymbol{x})$ 对于$x$的黑塞矩阵也记作：$\nabla_x^2 f(\boldsymbol{x}) = \boldsymbol{H}(\boldsymbol{x})$。

### A.3.6 雅可比矩阵

若$f(\boldsymbol{x})$ 是向量$\boldsymbol{x} \in \Re^{n \times 1}$的实向量函数，其中$f: \Re^n \to \Re^m$，即:

$$\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})]^T \tag{A-185}$$

$f(\boldsymbol{x})$ 的一阶微分包含每一个$f_i(\boldsymbol{x})$ 的微分，$i = 1, 2, \cdots, m$，分别如下

$$\nabla \boldsymbol{f}(\boldsymbol{x}) = [\nabla f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, \nabla f_m(\boldsymbol{x})] \tag{A-186}$$

其中

$$\nabla f_i(\boldsymbol{x}) = \left[ \frac{\partial f_i}{\partial x_1}, \frac{\partial f_i}{\partial x_2}, \cdots, \frac{\partial f_i}{\partial x_n} \right]^T \tag{A-187}$$

各自的梯度$\nabla f_1(\boldsymbol{x})$, $\nabla f_2(\boldsymbol{x})$, $\cdots$, $\nabla f_m(\boldsymbol{x})$构成了$n \times m$矩阵$\nabla \boldsymbol{f}(\boldsymbol{x})$ 的列；它的转置$m \times n$矩阵$\nabla^T \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{J}(\boldsymbol{x})$称作雅可比矩阵。因此，雅可比矩阵可以写作：

$$\boldsymbol{J}(\boldsymbol{x}) = \left[ \left. \frac{\partial f_i(\boldsymbol{x})}{\partial x_j} \right|_{ij} \right] \in \Re^{m \times n} \tag{A-188}$$

其中$i = 1, 2, \cdots, m, j = 1, 2, \cdots, n$。具体一些，雅可比矩阵可以写作：

$$\boldsymbol{J}(\boldsymbol{x}) = \begin{bmatrix} \nabla^T f_1(\boldsymbol{x}) \\ \nabla^T f_2(\boldsymbol{x}) \\ \vdots \\ \nabla^T f_m(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} = \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} \tag{A-189}$$

假设两个实向量函数$\boldsymbol{f}(\boldsymbol{x})$和$\boldsymbol{g}(\boldsymbol{x})$, $\boldsymbol{x} \in \Re^{n \times 1}$，其中$\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})]^T$, $\boldsymbol{g}(\boldsymbol{x}) = [g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \cdots, g_m(\boldsymbol{x})]^T$。我们可以（用适当的链式法则）写作：

$$\frac{\partial}{\partial \boldsymbol{x}} (\boldsymbol{f}^T \boldsymbol{g}) = \left( \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \right)^T \boldsymbol{g} + \left( \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}} \right)^T \boldsymbol{f} \tag{A-190}$$

其中$\partial f/\partial \boldsymbol{x} = \nabla^T f(\boldsymbol{x}) = \boldsymbol{J}_f(\boldsymbol{x})$为$f$关于$\boldsymbol{x}$的雅可比式，$\partial \boldsymbol{g}/\partial \boldsymbol{x} = \nabla^T \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{J}_g(\boldsymbol{x})$为$\boldsymbol{g}$关于$\boldsymbol{x}$的雅可比式。式（A-190）的结果为$n$维列向量。

### A.3.7 泰勒级数展开式

给定实向量$\boldsymbol{x} \in \Re^{n\times 1}$的实值标量函数$f(\boldsymbol{x})$，可以写出其在点$\boldsymbol{x}_k$的多变量泰勒级数展开式为： 597

$$f(\boldsymbol{x}) = f(\boldsymbol{x}_k + \Delta\boldsymbol{x}) = f(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_k} + \Delta\boldsymbol{x}^T \nabla f(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_k} + \frac{1}{2}\Delta\boldsymbol{x}^T \nabla^2 f(\boldsymbol{x})\Delta\boldsymbol{x}|_{\boldsymbol{x}=\boldsymbol{x}_k} + \text{高阶项} \tag{A-191}$$

其中$\nabla f(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_k} = \nabla f(\boldsymbol{x}_k)$是$f(\boldsymbol{x})$在$\boldsymbol{x}_k$处求得的关于$\boldsymbol{x}$的梯度，$\nabla^2 f(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_k} = \nabla^2 f(\boldsymbol{x}_k)$是$f(\boldsymbol{x})$关于$\boldsymbol{x}$在$\boldsymbol{x}_k$处求得的黑塞矩阵。在式（A-191）中（忽略高阶项，考虑到黑塞矩阵的对称性）$f(\boldsymbol{x})$在$\boldsymbol{x} = \boldsymbol{x}_k + \Delta\boldsymbol{x}$处的关于$\boldsymbol{x}$的偏导数为：

$$\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} \cong \nabla f(\boldsymbol{x}_k) + \nabla^2 f(\boldsymbol{x}_k)\Delta\boldsymbol{x} \tag{A-192}$$

设$f(\boldsymbol{x}), \boldsymbol{x} \in \Re^{n\times 1}$为$C^1$类的实值标量函数，则泰勒定理（或均值定理）描述如下：

$$f(\boldsymbol{x}) = f(\boldsymbol{x}_k + \Delta\boldsymbol{x}) = f(\boldsymbol{x}_k) + \Delta\boldsymbol{x}^T \nabla f(\boldsymbol{x}_k + \beta\Delta\boldsymbol{x}) \tag{A-193}$$

其中$\beta$是标量（$0 \leqslant \beta \leqslant 1$）。此外，若$f(\boldsymbol{x})$是$C^2$类的，则存在标量$\beta(0 \leqslant \beta \leqslant 1)$，使得：

$$f(\boldsymbol{x}) = f(\boldsymbol{x}_k + \Delta\boldsymbol{x}) = f(\boldsymbol{x}_k) + \Delta\boldsymbol{x}^T \nabla f(\boldsymbol{x}_k) + \frac{1}{2}\Delta\boldsymbol{x}^T \nabla^2 f(\boldsymbol{x}_k + \beta\Delta\boldsymbol{x})\Delta\boldsymbol{x} \tag{A-194}$$

## A.4 李雅普诺夫直接法

假设齐次线性非时变系统描述如下（参照A.2.12节）：

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) \tag{A-195}$$

其中$\boldsymbol{x} \in \Re^{n\times 1}, \boldsymbol{A} \in \Re^{n\times n}$，可以定义李雅普诺夫函数如下：

**定义A.12** 非时变李雅普诺夫函数记作$V(\boldsymbol{x})$，是状态向量$\boldsymbol{x}$的标量函数，满足如下条件$\forall t \geqslant t_0$且所有的$\boldsymbol{x}$在原点附近：

1. 函数$V(\boldsymbol{x})$和关于变量$x_1, x_2, \cdots, x_n$的一阶偏微分即$\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T$存在且连续。
2. $V(\boldsymbol{0}) = 0$。
3. 若$\boldsymbol{x} \neq \boldsymbol{0}$，则$V(\boldsymbol{x}) > 0$（正定）。

当

$$\dot{V}(\boldsymbol{x}) < 0 \text{ 负定} \tag{A-196}$$ 598

式（A-195）中系统的平衡状态$\boldsymbol{x}_e$是渐近稳定的，或等效矩阵$\boldsymbol{A}$有严格负实部的特征值。其中$\dot{V}(\boldsymbol{x})$是李雅普诺夫函数的全微分（参照A.3.3节），

$$\dot{V}(\boldsymbol{x}) = \frac{\partial V}{\partial x_1}\frac{\mathrm{d}x_1}{\mathrm{d}t} + \frac{\partial V}{\partial x_2}\frac{\mathrm{d}x_2}{\mathrm{d}t} + \cdots + \frac{\partial V}{\partial x_n}\frac{\mathrm{d}x_n}{\mathrm{d}t} = \underbrace{\left[\frac{\partial V}{\partial x_1}, \frac{\partial V}{\partial x_2}, \cdots, \frac{\partial V}{\partial x_n}\right]}_{\nabla_x^T V(x)} \underbrace{\begin{bmatrix} \frac{\mathrm{d}x_1}{\mathrm{d}t} \\ \frac{\mathrm{d}x_2}{\mathrm{d}t} \\ \vdots \\ \frac{\mathrm{d}x_n}{\mathrm{d}t} \end{bmatrix}}_{\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}} = \nabla_x^T V(\boldsymbol{x})\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} \tag{A-197}$$

在非时变系统，$\boldsymbol{x}_e=\mathbf{0}$总是 $\dot{\boldsymbol{x}}(t)=\boldsymbol{A}\boldsymbol{x}(t)$的一种平衡状态，其中平衡状态为 $\dot{\boldsymbol{x}}=\mathbf{0}$（或$\boldsymbol{A}\boldsymbol{x}=\mathbf{0}$）的解，并具有性质 $\forall t\geqslant 0, \boldsymbol{x}_e=\mathrm{e}^{\boldsymbol{A}t}\boldsymbol{x}_e$。若

$$\dot{\boldsymbol{V}}(\boldsymbol{x})\leqslant 0 \quad \text{半负定} \tag{A-198}$$

这种平衡状态称为李雅普诺夫稳定性（$L$稳定性）。在式（A-195）中考虑对于$\boldsymbol{A}$的特征值的关系中的$L$稳定性也有与类似的声明。即，$\boldsymbol{A}$有一些特征值其实部为零，没有重复特征值。李雅普诺夫函数通常称作能量函数。然而，在大多数情况下，李雅普诺夫函数没有动态系统的任何物理意义，或表示动态系统的能量。

对于线性非时变的情况，二次型可以代替李雅普诺夫函数为：

$$\boldsymbol{V}(\boldsymbol{x})=\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} \tag{A-199}$$

其中$\boldsymbol{P}\in\Re^{n\times n}, \boldsymbol{P}^T=\boldsymbol{P}$，通过适当的链式法则，$V(\boldsymbol{x})$的微分可以写作：

$$\dot{\boldsymbol{V}}(\boldsymbol{x})=\dot{\boldsymbol{x}}^T\boldsymbol{P}\boldsymbol{x}+\boldsymbol{x}^T\boldsymbol{P}\dot{\boldsymbol{x}} \tag{A-200}$$

利用式（A-195），方程（A-200）可以重新写作：

$$\dot{\boldsymbol{V}}(\boldsymbol{x})=\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{P}\boldsymbol{x}+\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{A}\boldsymbol{x}=\boldsymbol{x}^T(\boldsymbol{A}^T\boldsymbol{P}+\boldsymbol{P}\boldsymbol{A})\boldsymbol{x} \tag{A-201}$$

其中要求满足 $\dot{V}(\boldsymbol{x})<0$（负定）以保证系统的平衡状态是渐近稳定的。在式（A-201）我们让

$$\boldsymbol{A}^T\boldsymbol{P}+\boldsymbol{P}\boldsymbol{A}=-\boldsymbol{Q} \tag{A-202}$$

为了保证系统的平衡状态是渐近稳定的，$\boldsymbol{Q}>0$，即$\boldsymbol{Q}$必须是正定对称的（$\boldsymbol{Q}^T=\boldsymbol{Q}$）。因此，确定一个系统（它的齐次状态方程为 $\dot{\boldsymbol{x}}(t)=\boldsymbol{A}\boldsymbol{x}(t)$）是否有渐近稳定平衡状态的一个标准测试是，
599 首先要选择一个正定矩阵$\boldsymbol{Q}$，并解（A-202）（李雅普诺夫方程）求出$\boldsymbol{P}$。若$\boldsymbol{P}>0$（即正定），式（A-199）的二次函数是李雅普诺夫函数，系统平衡状态是渐近稳定的。

更一般的情况：

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t}=\boldsymbol{f}(\boldsymbol{x},t) \tag{A-203}$$

其中我们假设非线性时变动态系统，时变李雅普诺夫函数定义如下：

**定义A.13**　时变李雅普诺夫函数记作$\boldsymbol{V}(\boldsymbol{x}, t)$，是状态向量$\boldsymbol{x}\in\Re^{n\times 1}(\boldsymbol{x}=[x_1, x_2, \cdots, x_n]^T)$的标量函数，时间$t$满足下列条件：对于$t\geqslant t_0$，以及原点附近的所有$\boldsymbol{x}$（平衡状态是零状态）：

1. 函数$V(\boldsymbol{x}, t)$、对状态变量$x_1, x_2, \cdots, x_n$的一阶偏导数和$t$存在且连续。

2. $V(\mathbf{0}, t)=0$。

3. 对于$\boldsymbol{x}\neq 0, t\geqslant t_0, V(x, t)\geqslant\alpha\|\boldsymbol{x}\|>0$，其中$\alpha(0)=0$ [$\alpha(t)$是$t$的连续的非增标量函数]。

如果对于系统 $\dot{\boldsymbol{x}}(t)=\boldsymbol{f}(\boldsymbol{x},t)$（其中$\boldsymbol{f}(\mathbf{0}, t)=\mathbf{0}, \boldsymbol{x}\neq 0$，$\dot{V}(\boldsymbol{x}, t)<0$）可以找到时变李雅普诺夫函数，那么状态$\boldsymbol{x}=\mathbf{0}$是渐近稳定的。

考虑线性非时变数字系统用差分方程描述如下：

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k) \tag{A-204}$$

其中$\boldsymbol{x}\in\Re^{n\times 1}$，$\boldsymbol{A}\in\Re^{n\times n}$。若我们定义李雅普诺夫函数为$V(\boldsymbol{x})=\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x}$（其中$\boldsymbol{P}\in\Re^{n\times n}$，$\boldsymbol{P}>0, \boldsymbol{P}^T=\boldsymbol{P}$），对$V(\boldsymbol{x})$进行差分运算：

$$\Delta V[\boldsymbol{x}(k)]=V[\boldsymbol{x}(k+1)]-V[\boldsymbol{x}(k)] \tag{A-205}$$

则式（A-205）可以写作：

$$\Delta V[\boldsymbol{x}(k)]=\boldsymbol{x}^T(k+1)\boldsymbol{P}\boldsymbol{x}(k+1)-\boldsymbol{x}^T(k)\boldsymbol{P}\boldsymbol{x}(k) \tag{A-206}$$

把式（A-204）代入式（A-206），我们得到：

$$\Delta V[\boldsymbol{x}(k)] = \boldsymbol{x}^T(k)\boldsymbol{A}^T\boldsymbol{P}\boldsymbol{A}\boldsymbol{x}(k) - \boldsymbol{x}^T(k)\boldsymbol{P}\boldsymbol{x}(k) = \boldsymbol{x}^T(k)[\boldsymbol{A}^T\boldsymbol{P}\boldsymbol{A} - \boldsymbol{P}]\boldsymbol{x}(k) \tag{A-207}$$

平衡状态$\boldsymbol{x}_e = \boldsymbol{0}$是渐近稳定当且仅当式（A-207）中$\boldsymbol{A}^T\boldsymbol{P}\boldsymbol{A} - \boldsymbol{P} < 0$（负定）。即，若我们定义一个矩阵$\boldsymbol{Q} \in \Re^{n\times n}, \boldsymbol{Q} > 0, \boldsymbol{Q}^T = \boldsymbol{Q}$，那么

$$\boldsymbol{A}^T\boldsymbol{P}\boldsymbol{A} - \boldsymbol{P} = -\boldsymbol{Q} \tag{A-208}$$

如果式（A-208）的解$\boldsymbol{P}$是正定的，那么，式（A-207）$\Delta V[\boldsymbol{x}(k)] = -\boldsymbol{x}^T(k)\boldsymbol{Q}\boldsymbol{x}(k) < 0$，且式（A-204）的平衡状态$\boldsymbol{x}_e = \boldsymbol{0}$是渐近稳定的。 600

## A.5 无约束最优化方法

### A.5.1 极值的充分必要条件

一个无约束最优化问题可以描述如下：找一个向量$\boldsymbol{x} \in \Re^{n\times 1}$，使得实值标量函数$\mathscr{E} = \mathscr{E}(\boldsymbol{x})$达到最小值，这个函数称作代价函数或能量函数（或目标函数）。因此，我们可以描述无约束最优化问题为：

$$\underset{\boldsymbol{x}}{\text{minimize}}\,\mathscr{E}(\boldsymbol{x}) \tag{A-209}$$

其中对于设计向量$\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T$的元素没有任何强制性约束。不失一般性，一个无约束最优化问题等价的描述就是求出相同代价函数的负值的最大值：$\underset{\boldsymbol{x}}{\text{maximize}} -\mathscr{E}(\boldsymbol{x})$。

我们让$\boldsymbol{x}^*$为$\mathscr{E}(\boldsymbol{x})$的全局最小点，使得

$$\mathscr{E}(\boldsymbol{x}^*) \leqslant \mathscr{E}(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \Re^{n\times 1} \tag{A-210}$$

若$\mathscr{E}(\boldsymbol{x}^*) < \mathscr{E}(\boldsymbol{x}), \forall \boldsymbol{x} \in \Re^{n\times 1}$，则$\boldsymbol{x}^*$为严格全局最小点。并不是所有的函数有有穷的全局最小点。此外，即使对于一个函数全局最小值存在，也不能保证函数有严格的全局最小点。理想地，我们希望能够找到任一函数的全局最小点，但这是不现实的。很多最优化方法基于在特定点上的特殊函数的信息，这样，信息在特定点的邻域内有效。如果没有其他的信息是可用的，对此问题或做额外的假设，那么不能保证找到全局的解（或存在一个全局的解）。一个重要的例外就是函数$\mathscr{E}$是凸函数（如线性规划问题）。若我们加上约束考虑此问题，可行点集$\Sigma$可以定义为一组约束，这个集合也可以为凸集。注意，对于没有约束的问题，集合$\Sigma$可以为$\Re^n$。

如果在特定的函数中找不到全局最小点，那么求其次，求比周围点中更好的解。因此，我们要找函数$\mathscr{E}$的局部最小点，即，一个点满足：

$$\mathscr{E}(\boldsymbol{x}^*) \leqslant \mathscr{E}(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \Re^{n\times 1} \ni \|\boldsymbol{x} - \boldsymbol{x}^*\| < \varepsilon \tag{A-211}$$

其中$\varepsilon$是依赖于$\boldsymbol{x}^*$的小正数。若点$\boldsymbol{x}^*$（局部解）是严格局部最小点，则

$$\mathscr{E}(\boldsymbol{x}^*) < \mathscr{E}(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \Re^{n\times 1} \ni \boldsymbol{x} \neq \boldsymbol{x}^* \text{且} \|\boldsymbol{x} - \boldsymbol{x}^*\| < \varepsilon \tag{A-212}$$ 601

严格局部最小点在很多情况下可以通过计算$\mathscr{E}$对于$\boldsymbol{x}$，在$\boldsymbol{x} = \boldsymbol{x}^*$的一阶和二阶微分得到。因此，严格局部最小点可以通过计算代价函数的$\mathscr{E}$的一阶和二阶微分的算法求得。很多算法只允许在平稳点微分，特别是只计算一阶微分。对于无约束情况，$\mathscr{E}$的一阶微分在平稳点处等于0。即使特定问题要求全局解，很多情况下局部解如果能够使代价（目标）函数产生一个可接受的缩减量，也是令人满意的。

假设$\mathscr{E}(\boldsymbol{x})$的一阶和二阶导数存在，$\nabla_{\boldsymbol{x}}\mathscr{E}(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}^*} = \dfrac{\partial}{\partial \boldsymbol{x}}\mathscr{E}(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}^*} = \boldsymbol{0}$，向量$\boldsymbol{x}^*$是$\mathscr{E}(\boldsymbol{x})$的严格

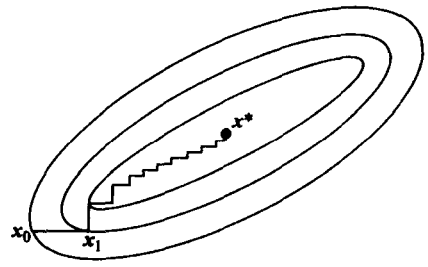
局部最小点（即，$\mathscr{E}(\boldsymbol{x})$对于$\boldsymbol{x}$的梯度在$\boldsymbol{x}^*$点的值为零），黑塞矩阵$\nabla^2\mathscr{E}(\boldsymbol{x})$在点$\boldsymbol{x}^*$是正定的，即$\boldsymbol{x}^T\nabla^2\mathscr{E}(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}^*}\boldsymbol{x}>0$，$\forall\boldsymbol{x}\in\Re^{n\times1}$，（除了$\boldsymbol{x}\neq\boldsymbol{0}$）。黑塞矩阵总是对称的（参照A.3.5节）。因此，一个点$\boldsymbol{x}^*$是严格局部最小点的充分必要条件为$\mathscr{E}(\boldsymbol{x}^*)<\mathscr{E}(\boldsymbol{x})$，$0<\|\boldsymbol{x}-\boldsymbol{x}^*\|<\varepsilon(\varepsilon>0)$的所有$\boldsymbol{x}$，若$\nabla_{\boldsymbol{x}}\mathscr{E}(\boldsymbol{x}^*)=0$，黑塞矩阵$\nabla^2\mathscr{E}(\boldsymbol{x}^*)$是对称正定的。

### A.5.2 最速下降法

最速下降法（常指梯度方法）是一种求多变量函数最小值的最古老以及最广泛使用的数值优化技术。它是其他方法比较的基准。许多其他方法是由最速下降的方法衍变而来的，以提高新算法的收敛性。若我们假定$\mathscr{E}$（能量函数）是$\boldsymbol{x}=[x_1,x_2,\cdots,x_n]^T(\boldsymbol{x}\in\Re^{n\times1})$多元函数，在$\Re^n$上有连续的偏微分，则$\mathscr{E}$关于向量$\boldsymbol{x}$的梯度由$\nabla_{\boldsymbol{x}}\mathscr{E}(\boldsymbol{x})\in\Re^{n\times1}$求得，有时写作$\boldsymbol{g}\triangleq\nabla_{\boldsymbol{x}}\mathscr{E}(\boldsymbol{x})$。在离散时间情况下，最速下降方法可以定义如下：

$$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k-\alpha_k\boldsymbol{g}_k \tag{A-213}$$

其中，$k$是离散时间的指标，$\boldsymbol{x}_k=\boldsymbol{x}(k)$，$\boldsymbol{g}_k\triangleq\nabla_{\boldsymbol{x}}\mathscr{E}(\boldsymbol{x}_k)$，$\alpha_k$是非负标量使$\mathscr{E}(\boldsymbol{x}_k-\alpha\boldsymbol{g}_k)$最小。最速下降从点$\boldsymbol{x}_k$沿着“负梯度”的方向$-\boldsymbol{g}_k$下降到这条线上的最小点搜索，最小点用$\boldsymbol{x}_{k+1}$表示。图A-5说明了最速下降的迭代过程。



图A-5　最速下降迭代过程

离散时间最速下降方法是非约束优化动态梯度系统的特例。上面提到，很多梯度下降方法由最速下降方法（和牛顿方法）发展而来。这些方法把最
602 优化（最小化）问题转化为求一阶微分方程：

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t}=-\mu(\boldsymbol{x},t)\nabla_{\boldsymbol{x}}\mathscr{E}(\boldsymbol{x}) \tag{A-214}$$

初始条件为$\boldsymbol{x}(t_0=0)=\boldsymbol{x}_0$,其中$\mu(\boldsymbol{x},t)$是对称正定矩阵即学习矩阵。为了找到向量$\boldsymbol{x}^*$，使得代价函数$\mathscr{E}(\boldsymbol{x})$最小，在式（A-214）中向量矩阵的普通微分方程（与初始条件一起）必须可解。首先考虑找到的$\boldsymbol{x}^*$的稳定性。式（A-214）的微分方程的稳定性可以通过对代价或能量（李雅普诺夫）函数对时间求导解决（参见A.4节）：

$$\frac{\mathrm{d}\mathscr{E}}{\mathrm{d}t}=\frac{\partial\mathscr{E}}{\partial\boldsymbol{x}}\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}=\nabla_{\boldsymbol{x}}^T\mathscr{E}(\boldsymbol{x})\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}=-\nabla_{\boldsymbol{x}}^T\mathscr{E}(\boldsymbol{x})\mu(\boldsymbol{x},t)\nabla_{\boldsymbol{x}}\mathscr{E}(\boldsymbol{x})<0 \tag{A-215}$$

因此，根据式（A-215）中的二次表达式，稳定性条件是学习矩阵必须是对称正定的。这保证能量函数$\mathscr{E}(\boldsymbol{x})$随时间减少，当$t\to\infty$时，收敛到稳定的局部最小点（平衡点）。换句话说，能量函数的局部最小点由梯度方法解轨迹$\boldsymbol{x}^*=\lim_{t\to\infty}\boldsymbol{x}(t)$求得。式（A-214）的学习矩阵中的项表明了收敛到最小点的速度。

学习矩阵（或开发的算法）的不同选择可以产生不同的梯度方法。当学习矩阵用单位矩阵乘以标量$\mu$（学习率参数）时，可以产生最简单形式。由式（A-214），微分方程的结果如下：

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t}=-\mu\nabla_{\boldsymbol{x}}\mathscr{E}(\boldsymbol{x})=-\mu\boldsymbol{g}(\boldsymbol{x}) \tag{A-216}$$

603 初始条件为$\boldsymbol{x}(0)=\boldsymbol{x}_0$。随着时间的变化，$\boldsymbol{x}(t)$的轨线沿着下降率最陡的方向移动，这个方向称最速下降。在式（A-216），根据最速下降，若学习率参数是正数（$\mu>0$），则学习规则的连续时间形式是收敛的。在离散时间形式中，式（A-216）最速下降连续时间学习规则变成：

$$\underbrace{\frac{\boldsymbol{x}_{(k+1)T_s}-\boldsymbol{x}_{kT_s}}{T_s}}_{\approx\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t}}=-\tilde{\mu}\nabla_{\boldsymbol{x}}\mathscr{E}(\boldsymbol{x}_k)=-\tilde{\mu}\boldsymbol{g}_k$$

或

$$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k-\mu\boldsymbol{g}_k \tag{A-217}$$

其中 $\boldsymbol{x}_k=\boldsymbol{x}_{kT_s}$，$T_s$是采样周期，$\mu=T_s\tilde{\mu}$，$\boldsymbol{x}(0)=\boldsymbol{x}_0$是初始条件向量。当$0\leqslant\mu\leqslant\mu_{\max}$时，式（A-217）保证收敛性（稳定性）。学习率参数（或积分步长大小）在时间$\mu_k$也能改变，例如，根据与退火调度表相似的算法（2.5.1节）作自适应调整。

### A.5.3 牛顿法

牛顿法涉及能量（或目标）函数$\mathscr{E}(\boldsymbol{x})$的局部近似值，以二次函数的形式到达极小值。这个$\mathscr{E}(\boldsymbol{x})$的二次近似值是对当前点$\boldsymbol{x}_k$局部的，精确地最小化。在标量情况下，我们的目标是在点$\boldsymbol{x}_k$对于$\boldsymbol{x}$最小化$\mathscr{E}(\boldsymbol{x})$，可用$\{\mathscr{E}(\boldsymbol{x}_k),\ \mathscr{E}'(\boldsymbol{x}_k),\ \mathscr{E}''(\boldsymbol{x}_k)\}$［例$\mathscr{E}'(\boldsymbol{x}_k)$是$\mathscr{E}(\boldsymbol{x}_k)$对于$\boldsymbol{x}$的一阶导数］。然后，可以构建一个二次函数$q$，在点$\boldsymbol{x}_k$，$\mathscr{E}$存在二次导数，即：

$$q(x)=\mathscr{E}(x_k)+\mathscr{E}'(x_k)(x-x_k)+\frac{1}{2}\mathscr{E}''(x_k)(x-x_k)^2 \tag{A-218}$$

$\mathscr{E}$的最小点$\boldsymbol{x}_{k+1}$的估计可以通过让$q$的导数成为0所求得的点计算，由式（A-218）得到：

$$\dot{q}(x)=\frac{\mathrm{d}q(x)}{\mathrm{d}x}=\mathscr{E}'(x_k)+\mathscr{E}''(x_k)(x-x_k)\Big|_{x=x_{k+1}}=0$$

或

$$\mathscr{E}'(\boldsymbol{x}_k)+\mathscr{E}''(\boldsymbol{x}_k)\boldsymbol{x}_{k+1}-\mathscr{E}''(\boldsymbol{x}_k)\boldsymbol{x}_k=0$$

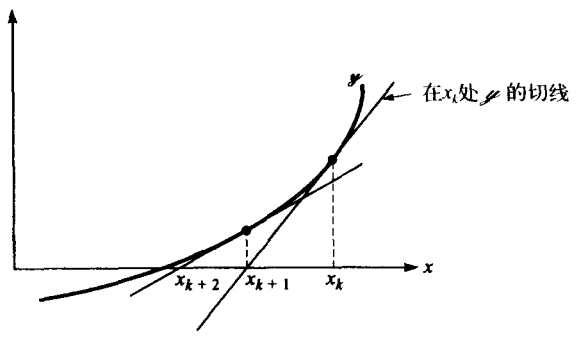
可以由经典牛顿法求出$\boldsymbol{x}_{k+1}$：

$$x_{k+1}=x_k-\frac{\mathscr{E}'(x_k)}{\mathscr{E}''(x_k)} \tag{A-219}$$

注意式（A-219）的结果不依赖于$\mathscr{E}(\boldsymbol{x}_k)$。一般而言，牛顿法可以看作迭代解形如$\varphi(x)=0$方程组的技术。当应用于最小化问题时，我们令$\varphi(x)=\mathscr{E}'(\boldsymbol{x})$。因此，牛顿法可以写作： 604

$$x_{k+1}=x_k-\frac{\varphi(x_k)}{\varphi'(x_k)} \tag{A-220}$$

迭代求出$\varphi(x)=0$的根。图A-6给出图解。




图A-6 用牛顿法迭代确定函数$\varphi$的根

对于多变量情况，即，$\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T$，标量函数$\mathscr{E}(\boldsymbol{x})$在局部由二次函数近似，这个近似函数精确最小化。因此，在点$\boldsymbol{x}_k$附近我们可以用截取泰勒级数展开式近似函数$\mathscr{E}$：

$$\mathscr{E}(\boldsymbol{x}) \cong \mathscr{E}(\boldsymbol{x}_k) + \nabla_x^T \mathscr{E}(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_k)^T \boldsymbol{H}_k (\boldsymbol{x} - \boldsymbol{x}_k) \tag{A-221}$$

其中$\nabla_x \mathscr{E} \in \Re^{n\times 1}[\boldsymbol{g}_k = \nabla_x \mathscr{E}(\boldsymbol{x}_k)]$是$\mathscr{E}$关于向量$\boldsymbol{x}$的梯度，$\boldsymbol{H} = \nabla^2 \mathscr{E} \in \Re^{n\times n}[\boldsymbol{H}_k = \boldsymbol{H}(\boldsymbol{x}_k) = \nabla^2 \mathscr{E}(\boldsymbol{x}_k)]$是黑塞矩阵（参照A.3.5节）。截取泰勒级数展开式（A-221）在点$\boldsymbol{x}_{k+1}$达到最小值需满足：

$$\nabla_x \mathscr{E}(\boldsymbol{x})\big|_{x=x_{k+1}} = 0$$

得到

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - H_k^{-1} \nabla_x \mathscr{E}(\boldsymbol{x}_k) \tag{A-222}$$

或

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - H_k^{-1} \boldsymbol{g}_k \tag{A-223}$$

从式（A-222）或式（A-223）看出，二阶充分必要条件要求黑塞矩阵在最小点$\boldsymbol{x} = \boldsymbol{x}^*$是正定的。在这种情况下目标函数是纯粹的二次式，即

605

$$\mathscr{E}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{b}^T \boldsymbol{x} + a \tag{A-224}$$

梯度为$\nabla_x \mathscr{E}(\boldsymbol{x}) = \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{b}$，黑塞矩阵是常量，即$\nabla_x^2 \mathscr{E}(\boldsymbol{x}) = \boldsymbol{H} = \boldsymbol{Q}$。因此，把这些结果与式（A-222）中的经典牛顿法比较，我们看出从任意初始点$\boldsymbol{x}_0$开始，二次函数的极小值一步就可到达。

### A.5.4 改进的牛顿法和拟牛顿法

*改进的牛顿法*

在理想的情况下，牛顿法的收敛速度是二次的，而最速下降（最简单的牛顿法）是线性收敛的。然而，在那些远离“解”的点，对黑塞矩阵的改进必须保证这个矩阵的正定性和下降。首先引入一个*搜索参数*$\alpha$，

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k (\boldsymbol{H}_k^{-1} \boldsymbol{g}_k) \tag{A-225}$$

其中$\alpha_k > 0$用来最小化目标函数$\mathscr{E}$。在接近解处我们期望$\alpha_k \cong 1$。然而，这个参数可以避免在实际的目标函数中非二次项导致的*增量*目标函数。方程（A-225）称作*有限步牛顿公式*。

基本牛顿法第二个改进包括阻止黑塞矩阵变为病态矩阵，即趋近于奇异的。可以通过很多方式实现。一种方法是对黑塞矩阵$\boldsymbol{H}$进行*LDU分解*（参照7.3节），即$\boldsymbol{H} = \boldsymbol{LDU} = \boldsymbol{LDL}^T$，$\boldsymbol{H}$是对称的。$\boldsymbol{D} \in \Re^{n\times n}$的对角元素表明$\boldsymbol{H}$的定性，若$\boldsymbol{H}$趋向于奇异矩阵，当$k$增加时，$\boldsymbol{D}$至少有一个对角元素有趋于零。$\boldsymbol{D}$的这个零元素（或几乎为零，或非正）可以被一个小正数替换，以保证当条件数增长时，改进的黑塞矩阵$\boldsymbol{H}$是正定的。可以通过每次迭代时执行下面的步骤实现：

**改进的牛顿法**

**步骤1**　选定初始解向量$\boldsymbol{x}_0$及收敛容差$\varepsilon$。

**步骤2**　k = 0, 1, 2, …，计算$\boldsymbol{g}_k = \nabla_x \mathscr{E}[\boldsymbol{x}_k]$（其中$\mathscr{E}$是目标函数）。如$\|\boldsymbol{g}_k\| < \varepsilon$，则停止计算。

**步骤3**　计算$\boldsymbol{H} = \boldsymbol{LDL}^T$。

**步骤4**　如有必要修改$\boldsymbol{D}$的对角元素，$\boldsymbol{D} \leftarrow \boldsymbol{D}_m$。

**步骤5**　从$(LD_mL^T)\boldsymbol{d}_k = -\boldsymbol{g}_k$计算搜索方向。

**步骤6**　执行线性搜索的确定$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$新解估计，其中$\alpha_k$选定如下：

$$\min_{\alpha \geqslant 0} \mathscr{E}(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$$

**步骤7** 回到步骤2。 □ 606

作为选择，一个对角矩阵可以加到$\boldsymbol{H}$，即用$\varepsilon_k \boldsymbol{I} + \boldsymbol{H}_k$取代$\boldsymbol{H}$，其中$\varepsilon_k$是最小的非负常量，使得矩阵$\varepsilon_k \boldsymbol{I} + \boldsymbol{H}_k$的特征值大于或等于$\delta > 0$（其中$\delta$是根据$\boldsymbol{H}$合理的条件数确定的）。然后定义方向向量

$$\boldsymbol{d}_k = -(\varepsilon_k \boldsymbol{I} + \boldsymbol{H}_k)\boldsymbol{g}_k \tag{A-226}$$

根据

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k \tag{A-227}$$

迭代，其中$\alpha \geqslant 0$使得$\mathscr{E}(\boldsymbol{x}_k - \alpha \boldsymbol{d}_k)$达到最小值。我们很容易从式（A-226）和式（A-227）看出最速下降方法是牛顿法的特例。特别是式（A-226），若每次迭代$\varepsilon_k \boldsymbol{I} + \boldsymbol{H}_k = \boldsymbol{I}_n$，则方向向量总是取$\mathscr{E}$的梯度的负值，即$\boldsymbol{d}_k = -\boldsymbol{g}_k$。式（A-227）的迭代表达式归约为最速下降式（A-213）。此外，黑塞矩阵$\boldsymbol{H}_k$加上$\varepsilon_k \boldsymbol{I}$等价于前面使用LDU分解改进的牛顿法。

拟牛顿法

很多情况下只有目标函数$\mathscr{E}$的梯度是可用的，而不是黑塞矩阵。在这种情况下，所谓的拟牛顿法（也叫做变度量法）可以使用。拟牛顿法的基本思想：在式（A-223）中黑塞矩阵的求逆的过程近似于梯度下降优化过程实际求逆的每一步。现今最流行的拟牛顿法之一就是Broyden-Fletcher-Goldfarb-Shanno（BFGS）算法。算法的细节在以下[9]中给出。

**Broyden-Fletcher-Goldfarb-Shanno 算法**

**步骤1** 选定初始解向量$x_0$及初始黑塞近似值$\boldsymbol{B}_0$（$\boldsymbol{B}_0 = \boldsymbol{I}$）。

**步骤2** $k = 0, 1, 2, \cdots$，若$\boldsymbol{x}_k$是最优的（在某种意义上），则停止。

**步骤3** 否则计算目标函数$\mathscr{E}$的梯度，即，$\boldsymbol{g}_k = \nabla \boldsymbol{x}\, \mathscr{E}(\boldsymbol{x}_k)$，然后对于$\boldsymbol{d}_k$解$\boldsymbol{B}_k \boldsymbol{d}_k = -\boldsymbol{g}_k$。

**步骤4** 执行线性搜索的确定$x_{k+1} = x_k + \alpha_k \boldsymbol{d}_k$，其中$\alpha_k$选定如下：

$$\min_{\alpha \geqslant 0} \mathscr{E}(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$$

**步骤5** 计算$\delta_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$及$\boldsymbol{y}_k = \boldsymbol{g}_{k+1} - \boldsymbol{g}_k$。

**步骤6** 计算

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}_k - \frac{(\boldsymbol{B}_k \delta_k)(\boldsymbol{B}_k \delta_k)^T}{\delta_k^T \boldsymbol{B}_k \delta_k} + \frac{\boldsymbol{y}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \delta_k}$$

其中，$\boldsymbol{B}_k$是黑塞矩阵$\nabla_x^2 \mathscr{E}(\boldsymbol{x}_k)$的当前估计值。

**步骤7** 回到步骤2。 □

上面的步骤6用来修正黑塞矩阵的估计值，看作秩2公式。秩2修正公式保证黑塞矩阵近似值是对称正定的。 607

### A.5.5 共轭梯度法

共轭梯度方法原来是用来解决

$$\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{b} \tag{A-228}$$

对于$\boldsymbol{x} \in \Re^{n \times 1}, \boldsymbol{Q} \in \Re^{n \times n}, (\boldsymbol{Q}^T = \boldsymbol{Q}, \boldsymbol{Q} > 0), \boldsymbol{b} \in \Re^{n \times 1}$。求解式（A-228）等价于求标量函数[10]

$$\mathscr{E}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{Q}\boldsymbol{x} - \boldsymbol{x}^T \boldsymbol{b} \tag{A-229}$$

的最小值。在共轭梯度法中，有一个方向向量集$\{\boldsymbol{d}_0, \boldsymbol{d}_1, \cdots, \boldsymbol{d}_{n-1}\}$与矩阵$\boldsymbol{Q}$共轭（参照A.2.8节定义A.8），即$\boldsymbol{d}_i^T\boldsymbol{Q}\boldsymbol{d}_j = 0$，$i \neq j$。在迭代过程的第$k$次迭代，把目标函数当前计算的负梯度向量和以前方向向量线性组合生成的共轭方向向量。共轭梯度法的优点在于：（1）用非常简单的公式确定新的方向向量；（2）这使得共轭梯度法稍微比最速下降法复杂；（3）因为方向向量是基于计算的梯度，所以过程中的每一步求解都保持很好的统一性。对于纯二次式的情况，这是不重要的；但共轭梯度法的通用性对于非二次式问题非常重要。共轭梯度算法总结如下：

**共轭梯度法**

**步骤1**　以任意$\boldsymbol{x}_0 \in \Re^{n\times 1}$开始。定义初始方向向量为

$$\boldsymbol{d}_0 = -\boldsymbol{g}_0 = -\Delta_{\boldsymbol{x}}\mathscr{E}(\boldsymbol{x}_k)|_{k=0} = \boldsymbol{b} - \boldsymbol{Q}\boldsymbol{x}_0$$

**步骤2**　$\alpha_k = -\dfrac{\boldsymbol{g}_k^T\boldsymbol{d}_k}{\boldsymbol{d}_k^T\boldsymbol{Q}\boldsymbol{d}_k}$，其中，$\boldsymbol{g}_k = \boldsymbol{Q}\boldsymbol{x}_k - \boldsymbol{b}$

**步骤3**　$x_{k+1} = \boldsymbol{x}_k + \alpha_k\boldsymbol{d}_k$

**步骤4**　$\boldsymbol{d}_{k+1} = -\boldsymbol{g}_{k+1} + \beta_k\boldsymbol{d}_k$，其中 $\beta_k = \dfrac{\boldsymbol{g}_{k+1}^T\boldsymbol{Q}\boldsymbol{d}_k}{\boldsymbol{d}_k^T\boldsymbol{Q}\boldsymbol{d}_k}$，

$\beta_k$的另一个形式是

$$\beta_k = \frac{\boldsymbol{g}_{k+1}^T\boldsymbol{g}_{k+1}}{\boldsymbol{g}_k^T\boldsymbol{g}_k}$$

**步骤5**　回到步骤2。　□

这个算法在有限步内收敛，二次式问题收敛在$n$步内完成。注意，在共轭梯度算法中第一步与最速下降算法是等同的。在上面算法的步骤2，参数$\alpha_k$是给定的。这个参数可由 $\alpha_k = \min_{\alpha\geqslant 0}\mathscr{E}$

608　$(\boldsymbol{x}_k + \alpha\boldsymbol{d}_k)$确定。即，对于纯二次式（A-229）的情况 $\mathscr{E}(\boldsymbol{x}) = \dfrac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{b}$。因此，

$$\begin{aligned}\mathscr{E}(\boldsymbol{x}_k + \alpha\boldsymbol{d}_k) &= \frac{1}{2}(\boldsymbol{x}_k + \alpha\boldsymbol{d}_k)^T\boldsymbol{Q}(\boldsymbol{x}_k + \alpha\boldsymbol{d}_k) - (\boldsymbol{x}_k + \alpha\boldsymbol{d}_k)^T\boldsymbol{b} \\ &= \frac{1}{2}(\boldsymbol{x}_k^T\boldsymbol{Q}\boldsymbol{x}_k + 2\alpha\boldsymbol{d}_k^T\boldsymbol{Q}\boldsymbol{x}_k + \alpha^2\boldsymbol{d}_k^T\boldsymbol{Q}\boldsymbol{d}_k) - \boldsymbol{x}_k^T\boldsymbol{b} - \alpha\boldsymbol{d}_k^T\boldsymbol{b}\end{aligned} \tag{A-230}$$

计算式（A-230）对参数$\alpha$的梯度，并令结果为0：

$$\begin{aligned}\frac{\partial\mathscr{E}(\boldsymbol{x}_k + \alpha\boldsymbol{d}_k)}{\partial\alpha} &= \boldsymbol{d}_k^T\boldsymbol{Q}\boldsymbol{x}_k - \boldsymbol{d}_k^T\boldsymbol{b} + \alpha\boldsymbol{d}_k^T\boldsymbol{Q}\boldsymbol{d}_k = \boldsymbol{d}_k^T\underbrace{(\boldsymbol{Q}\boldsymbol{x}_k - \boldsymbol{b})}_{\boldsymbol{g}_k} + \alpha\boldsymbol{d}_k^T\boldsymbol{Q}\boldsymbol{d}_k \\ &= \boldsymbol{d}_k^T\boldsymbol{g}_k + \alpha\boldsymbol{d}_k^T\boldsymbol{Q}\boldsymbol{d}_k = \boldsymbol{g}_k^T\boldsymbol{d}_k + \alpha\boldsymbol{d}_k^T\boldsymbol{Q}\boldsymbol{d}_k = 0\end{aligned} \tag{A-231}$$

对于$\alpha = \alpha_k$，式（A-231）中的解为：

$$\alpha_k = -\frac{\boldsymbol{g}_k^T\boldsymbol{d}_k}{\boldsymbol{d}_k^T\boldsymbol{Q}\boldsymbol{d}_k} \tag{A-232}$$

$\alpha_k$的表达式在共轭梯度算法的步骤2给出。

现在让我们把结果推广到非二次问题。假设在解点的附近，问题近似为二次方程。有几种方法可以办到。但是，我们只介绍一种方法，基于线性搜索算法，和这个算法的两个变异。

Fletcher-Reeves共轭梯度算法（带重启）

为求$\mathscr{E}(\boldsymbol{x})$的最小值，其中$\boldsymbol{x} \in \Re^{n\times 1}$，$\mathscr{E}$不一定是二次函数。

**Fletcher-Reeves 共轭梯度算法（带重启）**

**步骤1** 设$\boldsymbol{x}_0$。

**步骤2** 计算

$$\boldsymbol{g}_0 = \nabla_x \mathscr{E}(\boldsymbol{x}_0) = \left.\frac{\partial \mathscr{E}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_0}$$

**步骤3** 设$\boldsymbol{d}_0 = -\boldsymbol{g}_0$

**步骤4** 计算

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k, \quad \text{其中} \alpha_k = \min_{\alpha \geqslant 0} \mathscr{E}(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$$

**步骤5** 计算$\boldsymbol{g}_{k+1} = \Delta_x \mathscr{E}(\boldsymbol{x}_{k+1})$

**步骤6** 计算

$$\boldsymbol{d}_{k+1} = -\boldsymbol{g}_{k+1} + \beta_k \boldsymbol{d}_k, \quad \text{其中}, \quad \beta_k = \frac{\boldsymbol{g}_{k+1}^T \boldsymbol{g}_{k+1}}{\boldsymbol{g}_k^T \boldsymbol{g}_k}$$

执行步骤4至6时$k = 0, 1, \cdots, n-1$。

**步骤7** 以$\boldsymbol{x}_n$替代$\boldsymbol{x}_0$，回到步骤1。

**步骤8** 继续直到达到收敛。终止标准应是$\|\boldsymbol{d}_k\| < \varepsilon$（其中$\varepsilon$是一个适当的预先确定的小数）。 □ 609

在上面的步骤6计算$\beta_k$时用到了*Fletcher-Reeves*公式。其他两个计算$\beta_k$的方法如下：

Polak-Ribiere法 $\beta_k = \dfrac{(\boldsymbol{g}_{k+1} - \boldsymbol{g}_k)^T \boldsymbol{g}_{k+1}}{\boldsymbol{g}_k^T \boldsymbol{g}_k}$

Hestenes-Stiefel法 $\beta_k = \dfrac{(\boldsymbol{g}_{k+1} - \boldsymbol{g}_k)^T \boldsymbol{g}_{k+1}}{\boldsymbol{d}_k^T (\boldsymbol{g}_{k+1} - \boldsymbol{g}_k)}$

共轭梯度法是最速下降法和拟牛顿法的折衷。上面算法的重启特性（在步骤7）对于目标函数不是二次式的情况非常重要。在每$n$重迭代（或非下降搜索方向生成）后，在最速下降方向的搜索重启Fletcher-Reeves共轭梯度算法。纯最速下降每执行$n$步称作一个“间隔步”。算法重启的特性对于全局收敛非常重要，因为通常不能保证生成的方向$\boldsymbol{d}_k$就是下降方向。注意，在上面的步骤4，$\alpha_k$必须来自特定的目标函数，这在某些情况下很难做到。

## A.6 约束非线性规划

### A.6.1 库恩-塔克条件

库思-塔克条件是不等式约束的最优化问题的必要条件。我们让$\boldsymbol{x}^*$为标量函数$\mathscr{E}$的局部极小值（最小点）限于约束$g_j(\boldsymbol{x}) \geqslant 0$，其中$j = 1, 2, \cdots, m, \boldsymbol{x} \in \Re^{n\times 1}$。因此，希望

最小化 $\mathscr{E}(\boldsymbol{x})$ (A-233)

受限于 $g_j(x) \geqslant 0 \quad j = 1, 2, \cdots, m$ (A-234)

我们可以构建一个由下式给出的拉格朗日函数[9-11]：

$$\mathscr{L}(\boldsymbol{x}, \lambda) = \mathscr{E}(\boldsymbol{x}) - \sum_{j=1}^{m} \lambda_j g_j(\boldsymbol{x}) = \mathscr{E}(\boldsymbol{x}) - \lambda^T \boldsymbol{g}(\boldsymbol{x}) \tag{A-235}$$

其中$\boldsymbol{g} \in \Re^{m\times 1}$是约束向量，$\boldsymbol{\lambda} \in \Re^{m\times 1}$是拉格朗日乘子向量。再令$\boldsymbol{x}^*$在式（A-233）和式（A-234）为问题的局部最小点,它也是约束的正则点[⊖]。那么，存在一个拉格朗日乘子向量 610 $\lambda^*$，满足如下条件（称作库恩-塔克条件）：

1. $\nabla_x \mathscr{L}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) = \boldsymbol{0}$，即

$$\nabla_x \mathscr{E}(\boldsymbol{x}^*) - \sum_{j=1}^{m} \lambda_j^* \nabla_x g_j(\boldsymbol{x}^*) = \frac{\partial \mathscr{E}(\boldsymbol{x}^*)}{\partial x_i} - \sum_{j=1}^{m} \lambda_j^* \frac{\partial g_j(\boldsymbol{x}^*)}{\partial x_i} = \boldsymbol{0} \tag{A-236}$$

对于$i = 1, 2, \cdots, n$

2. $$\lambda_j^* g_j(\boldsymbol{x}^*) = 0 \quad j=1,2, \cdots, m \tag{A-237}$$

3. $$\lambda_j^* \geqslant 0 \quad j=1, 2, \cdots, m \tag{A-238}$$

4. $$g_j(\boldsymbol{x}^*) \geqslant 0 \quad j = 1, 2, \cdots, m \tag{A-239}$$

假设式（A-236）～式（A-239），$\mathscr{E}(\boldsymbol{x})$和$g_j(\boldsymbol{x})$有连续的一阶偏微分。点$\boldsymbol{x} \in \Sigma \subset \Re^{n\times 1}$满足所有约束称为可行性。若可行点的集合$\Sigma$非空，最优化问题称作相容的。若一个可行点$\boldsymbol{x}^*$是标量函数$\mathscr{E}(\boldsymbol{x})$在可行点集$\Sigma$上的局部最小点，它就是局部最小点。这个条件在式（A-237）中，即$\boldsymbol{\lambda}^{*T}\boldsymbol{g}(\boldsymbol{x}^*) = 0$，称作互补松弛条件。既然向量$\boldsymbol{\lambda}^*$和$\boldsymbol{g}(\boldsymbol{x}^*)$均为非负，它暗示对于每个$j$，$\lambda_j^* g_j(\boldsymbol{x}^*) = 0$，这意味着约束无效或相应的拉格朗日乘子为0。特别地，任何一个无效约束都有拉格朗日乘子等于0。若有关的拉格朗日乘子与有效约束均为正，那么严格的互补性存在。否则，若有效约束相应的拉格朗日乘子为0，约束是退化的[9]。

### A.6.2　拉格朗日乘子法

我们处理约束优化问题，考虑如下非线性规划问题：

最小化　$$\mathscr{E}(\boldsymbol{x}) \tag{A-240a}$$

受限于　$$g_j(\boldsymbol{x}) \leqslant c_j \quad j = 1, 2, \cdots, m_1 \tag{A-240b}$$

$$g_j(\boldsymbol{x}) \geqslant c_j \quad j = m_1+1, \cdots, m_2 (m_1 \leqslant m_2) \tag{A-240c}$$

$$g_j(\boldsymbol{x}) = c_j \quad j = m_2+1, \cdots, m (m_2 \leqslant m) \tag{A-240d}$$

其中$\boldsymbol{x} \in \Re^{n\times 1}$，假设$\mathscr{E}(\boldsymbol{x})$和$g_j(\boldsymbol{x})$，$j = 1, 2, \cdots, m$有连续的一阶偏导数。对于非线性规划问题，我们可以定义拉格朗日函数：

$$\mathscr{L}(\boldsymbol{x}, \lambda) = \mathscr{E}(\boldsymbol{x}) - \sum_{j=1}^{m} \lambda_j [g_j(\boldsymbol{x}) - c_j] \tag{A-241}$$

若我们假设$\boldsymbol{x}^*$是正则点和该问题的局部最小点，那么至少存在一个非零向量$\boldsymbol{\lambda}^* \in \Re^{m\times 1}$，使

611

1. $$\lambda_j^* \leqslant 0 \quad j = 1, 2, \cdots, m_1 \tag{A-242a}$$

2. $$\lambda_j^* \geqslant 0 \quad j = m_1+1, \cdots, m_2 \tag{A-242b}$$

3. $$\text{任意符号的}\lambda_j^* \quad j = m_2+1, \cdots, m \tag{A-242c}$$

4. $$\lambda_j^* = 0 \quad j \in \boldsymbol{J}_0 \tag{A-242d}$$

其中$\boldsymbol{J}_0$是$j$下标的集合，$j = 1, 2, \cdots, m_2$，不等式在$\boldsymbol{x}^*$处为严格不等。

5. $$\lambda_j^* [g_j(\boldsymbol{x}^*) - c_j] = 0 \quad j = 1, 2, \cdots, m_2 \tag{A-242e}$$

---

⊖ $\boldsymbol{x}^*$的局部最小点称作约束的正则点，如果梯度向量$\nabla_x g_j(\boldsymbol{x}^*)$，$j \in \boldsymbol{J}$是线性独立的，其中$\boldsymbol{J}$是在$\boldsymbol{x}^*$对应有效的不等式约束的指标集，即，$\boldsymbol{J} = \{j: 1 \leqslant j \leqslant m, g_j(\boldsymbol{x}^*) = 0\}$（即，所有指标在$\boldsymbol{x}^*$对应等式约束）。

6. $$\nabla_x \mathscr{E}(\boldsymbol{x}^*) - \sum_{j=1}^{m} \lambda_j^* \nabla_x g_j(\boldsymbol{x}^*) = \mathbf{0} \tag{A-242f}$$

对于目标函数$\mathscr{E}(\boldsymbol{x})$最小化的特例，须满足等值约束$g_j(\boldsymbol{x}) = c_j$，$j = 1, 2, \cdots, m$，这些条件归结于下面的关系：

$$\nabla_x \mathscr{E}(\boldsymbol{x}^*) - \sum_{j=1}^{m} \lambda_j^* \nabla_x g_j(\boldsymbol{x}^*) = \mathbf{0} \tag{A-243}$$

其他条件是冗余的。

拉格朗日乘子法在一个等值约束条件下的一个例子，求可以装入椭球体的最大体积：

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \tag{A-244}$$

假设盒子的每一条边是平行于直角坐标系轴，盒子的8个角的每一个均在椭球体上。令第1个角（八分之一）的坐标为$(x, y, z)$，因此，盒子的维度为$2x, 2y, 2z$，体积为$V = 8xyz$。这是在式（A-244）约束下希望确定的$V$的最大值；因此，它的拉格朗日函数可以写作：

$$\mathscr{L}(x, y, z, \lambda) = 8xyz - \lambda\left(\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1\right) \tag{A-245}$$

式（A-243）的必要条件为：

$$\frac{\partial \mathscr{L}}{\partial x} = 8yx - 2\lambda\frac{x}{a^2} = 0 \tag{A-246a}$$

$$\frac{\partial \mathscr{L}}{\partial y} = 8xz - 2\lambda\frac{y}{b^2} = 0 \tag{A-246b}$$

$$\frac{\partial \mathscr{L}}{\partial z} = 8xy - 2\lambda\frac{z}{c^2} = 0 \tag{A-246c}$$

式（A-246a）～式（A-246c）中每个方程除以2，式（A-246a）乘以$x$，式（A-246b）乘以$y$，式（A-246c）乘以$z$，再相加，得：

612

$$12xyz - \lambda\underbrace{\left(\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2}\right)}_{1} = 0 \Rightarrow \lambda = 12xyz \tag{A-247}$$

现在将式（A-247）的结果代入必要条件式（A-246a）～式（A-246c），得：

$$yz(a^2 - 3x^2) = 0 \qquad xz(b^2 - 3y^2) = 0 \qquad xy(c^2 - 3z^2) = 0 \tag{A-248}$$

既然要求最大体积，从式（A-248）中的三个表达式我们可以求得正值$x, y, z$。因此，

$$x = \frac{a}{\sqrt{3}} \qquad y = \frac{b}{\sqrt{3}} \qquad z = \frac{c}{\sqrt{3}} \tag{A-249}$$

和

$$\lambda = 12xyz = \frac{4abc}{\sqrt{3}} \tag{A-250}$$

因此，可以装进椭球体的最大体积的盒子的维度为$(2a/\sqrt{3},\ 2b/\sqrt{3},\ 2c/\sqrt{3})$，最大体积为

$$V_{\max} = \frac{8abc}{3\sqrt{3}} \tag{A-251}$$

## A.7 随机变量和随机过程

### A.7.1 随机变量

随机变量的表达始于概率的讨论。有多种方法定义概率。四个最常用的为：(1) 公理(测量)；(2) 相对频率（冯·米泽斯分布 (Von Mises)）；(3) 以有利因素占可供选择的总数的比率为优先权的定义（经典方法）；(4) 可信度度量（归纳推理）。然而，这两个最有用的定义是相对频率方法和公理方法。相对频率方法试图把概率和物理意义联系起来。因此，这种方法可以把有关概率概念和现实世界联系起来。公理方法把一个事件的概率看作是一个满足特定假设的数，否则未定义。这个数没有必要一定要涉及现实世界的任何东西，也没有必要与假定中出现的数学结构相关。

*相对频率方法*

在概率的相对频率方法中，概率与特定事件的出现频率密切相关。一个事件发生或不发
613 生是偶然的。例如，抛硬币结果可能为正面也可能为反面，每一个都是事件。试验和试验的结果对于更加精确理解这个概念十分重要。抛硬币，掷骰子，抓扑克牌，观测电压大于零(小于零) ……所有这些均是试验的例子。在列举的每个试验中，出现的结果是有限的。这是离散概率的特例。如果试验在一个可能的连续值的范围内观测电压，那么有无限种结果（这是连续概率的例子）。在抛硬币的试验中，我们期望，对于相对大量的试验次数，一半次数是正面，一半次数是反面。因此，我们可以对两个事件之一的每个事件的概率赋值为1/2。一般地，若一个试验执行$N$次，我们期望事件$A$发生$N_A$次，则我们假定$A$的概率为$\Pr(A)$，即，

$$\Pr(A) = \frac{N_A}{N} \tag{A-252}$$

数目$N_A$不是在$N$次试验中$A$发生的实际次数，只是我们假设基于这个试验的直觉的数。如果某个试验可能有的结果为：$A$, $B$, $C$, …, $M$（在任何一次试验中只有一种结果发生），那么，可能事件称作互斥的。如果事件$A$期望在$N$次试验中发生$N_A$次，事件$B$期望发生$N_\beta$次，依此类推，那么，

$$N_A + N_B + N_C + \cdots + N_M = N \tag{A-253}$$

两边均除以$N$：

$$\frac{N_A}{N} + \frac{N_B}{N} + \frac{N_C}{N} + \cdots + \frac{N_M}{N} = 1 \tag{A-254}$$

由式（A-252），我们可以把式（A-254）写作：

$$\Pr(A) + \Pr(B) + \Pr(C) + \cdots + \Pr(M) = 1 \tag{A-255}$$

四个重要的声明总结如下:

1. $0 \leqslant \Pr(A) \leqslant 1$。
2. $\Pr(A) + \Pr(B) + \Pr(C) + \cdots + \Pr(M) = 1$，假设为互斥事件的全集。
3. 一个不可能事件表示为：$\Pr(A) = 0$。
4. 一个必然事件表示为：$\Pr(A) = 1$。

如果有几个事件一次同时发生，那么必须考虑联合概率。例如，$\Pr(A, B)$表示事件$A$和$B$联合发生

的概率。联合概率不一定等于各个（边缘）概率的积，即$\Pr(A, B, \cdots, M) \neq \Pr(A)\Pr(B)\cdots\Pr(M)$。

另一个重要类型的概率是条件概率。例如，$\Pr(A|B)$表示事件$A$在事件$B$已发生条件下发生的概率。一般地，我们可以写作：

$$\Pr(A, B) = \Pr(A|B)\Pr(B) = \Pr(B|A)\Pr(A) \tag{A-256}$$

614

两个随机事件是统计独立的当且仅当：

$$\Pr(A, B) = \Pr(A)\Pr(B) \tag{A-257}$$

公理法

这个方法把概率理论和集合论的概念联系起来。概率空间($\mathscr{S}$)定义为在一次试验中可能发生的所有结果作为元素的集合。这样每一事件被赋予一个数当作这个事件的概率。$\mathscr{S}$的不同子集可以由不同的事件区分。我们把事件$A$的概率记作$\Pr(A)$。被赋予的数必须满足如下3个条件（或公理）：

1. $\Pr(A) \geqslant 0$ (A-258)
2. $\Pr(\mathscr{S}) = 1$ (A-259)
3. 若$AB = \varnothing$，则$\Pr(A + B) = \Pr(A) + \Pr(B)$ (A-260)

其中$AB$是积或交，$\varnothing$表示空集（参见A.3.1节）。

概率的全体可以由这3个公理推出。一些重要的推论也可以由这些公理推出：

- 因为$\mathscr{S}\varnothing = \varnothing$，$\mathscr{S} + \varnothing = \varnothing$（其中$\mathscr{S} + \varnothing$是和或并（参照A.3.1节）），使用式（A-260）它遵循：

$$\Pr(\mathscr{S} + \varnothing) = \Pr(\mathscr{S})，\text{因此}\Pr(\varnothing) = 0 \tag{A-261}$$

- 因为$A\tilde{A} = \varnothing$，其中$\tilde{A}$是$A$的补集（参照A.3.1节），$A + \tilde{A} = \mathscr{S}$，使用式（A-260），它遵循：

$$\Pr(A + \tilde{A}) = \Pr(A) + \Pr(\tilde{A}) = \Pr(\mathscr{S}) = 1 \tag{A-262}$$

- 由式（A-262）和式（A-258），它满足：

$$\Pr(A) = 1 - \Pr(\tilde{A}) \leqslant 1 \tag{A-263}$$

因此，一个事件的概率一定在0与1之间。

- 若$A$和$B$不是互斥事件，则：

$$\Pr(A + B) = \Pr(A) + \Pr(B) - \Pr(AB) \leqslant \Pr(A) + \Pr(B) \tag{A-264}$$

条件概率的一个重要性质是：

$$\Pr(A|B) = \frac{\Pr(AB)}{\Pr(B)} \qquad \Pr(B) > 0 \tag{A-265}$$

其中$\Pr(AB)$是事件$AB$的概率。

独立性

两个事件$A$和$B$是相互独立的当且仅当：

$$\Pr(AB) = \Pr(A)\Pr(B) \tag{A-266}$$

615

随机变量的概念可以总结为如下定义。

**定义A.14** 实随机变量$X$是一个实函数，它的定义域是样本空间$\mathscr{S}$（即，$\mathscr{S} = \{\alpha\}$，一个随机试验的所有可能结果的集合）且

1. 对于任何实数$x \in \Re$，集合$\{X \leqslant x\}$是一个事件。

2. 事件$\{X=+\infty\}$和$\{X=-\infty\}$的概率为零，即$\Pr(X=+\infty)=\Pr(X=-\infty)=0$。

因此，从定义A.14，一个随机变量是直接定义在样本空间$\mathcal{S}$的实值函数；或者可以认为它是随机试验的结果的数量描述。当随机试验的结果是$\alpha$，随机变量$X$有一个值记为$X(\alpha)$。若随机变量在一定范围内（可能是无限的）可以假定为任意值，那么它是一个连续的随机变量。离散随机变量只能假设为可数集合的值。然而，它可以当作连续随机变量使用相同的方法来精确的处理。在下面，如果不特别说明，我们将集中讨论连续随机变量。

### A.7.2 概率分布函数

连续随机变量可以在概率概念的框架内考虑，通过概率分布函数[12]定义事件及其相关概率空间。我们令$X$为一个随机变量，$x$为随机变量所允许的任意值。这样，概率分布函数定义为事件的概率，观察的随机变量$X$小于或等于允许的值$x$，即，

$$\mathcal{P}_X(x)=\Pr(X\leqslant x)^{\ominus} \tag{A-267}$$

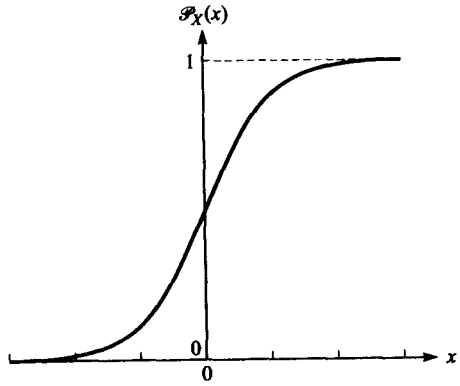
一个概率分布函数本身就是概率。因此，它必须满足A.7.1节中概率的性质。然而，这个函数也是$x$的函数（随机变量$X$的可能值），这样必须定义具有一般性以适合所有的$x$值。概率分布函数的性质为：

1. $0\leqslant \mathcal{P}_X(x)\leqslant 1 \quad -\infty<x<\infty$
2. $\mathcal{P}_X(-\infty)=0 \quad \mathcal{P}_X(\infty)=1$
3. 当$x$增加，$\mathcal{P}_X$非递减
4. $\Pr(x_1<X\leqslant x_2)=\mathcal{P}_X(x_2)-\mathcal{P}_X(x_1)$

616 概率分布函数也可以用来表示事件的概率，观测的随机变量$X$大于但不等于$x$。这个事件即为概率为$\mathcal{P}_X(x)$的事件的补集，即：

$$\Pr(X>x)=1-\mathcal{P}_X(x) \tag{A-268}$$

一个典型的概率分布函数在图A-7给出。




图A-7　典型概率分布函数

### A.7.3 概率密度函数

概率密度函数对于单随机变量的概率模型更方便。（边缘）概率密度函数是概率分布函数

---

⊖ 符号$\mathcal{P}_X(x)$，通常在数学文献中用来替代$F_X(x)$，表示概率分布函数。类似地，$\mathcal{p}_X(x)$也用来替代$f_X(x)$表示概率密度函数。

的微分（当微分存在的时候），即，

$$p_X(x) = \frac{\mathrm{d}\mathscr{P}_x(x)}{\mathrm{d}x} \tag{A-269}$$

概率密度函数的一般性质是：

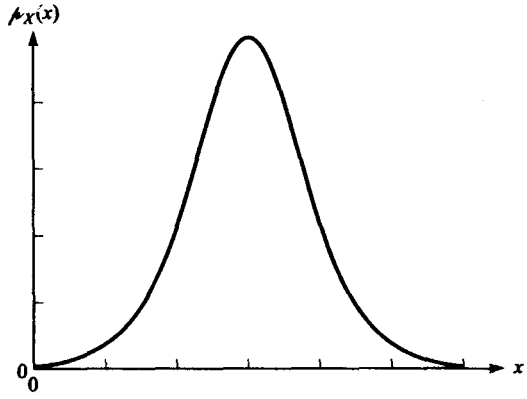
1. $p_X(x) \geqslant 0 \quad -\infty < x < \infty$

2. $\int_{-\infty}^{\infty} p_X(x)\mathrm{d}x = 1$

3. $\mathscr{P}_X(x) = \int_{-\infty}^{x} p_X(u)\mathrm{d}u$

4. $\int_{x_1}^{x_2} p_X(x)\mathrm{d}x = \Pr(x_1 < X \leqslant x_2)$

图A-8给出了典型的概率密度函数。




图A-8 （与图A-7的概率分布函数相关的）概率密度函数

### A.7.4 期望值、均值和矩

求取时间函数平均值的概念对于工程师和科学家是再熟悉不过了。时间平均值对于时间随机函数也很重要。然而，它们对于单随机变量没有意义（被定义为即时的时间函数的值）。 617
在一个随机变量的情况下，在可以假设的随机变量可能值的范围内积分求得平均值十分必要。这种操作称作*总体均值*，结果称作*均值*。随机变量$X$的均值为：

$$\mathrm{E}(X) = \overline{X} = \int_{-\infty}^{\infty} x p(x)\,\mathrm{d}x \tag{A-270}$$

其中$p(x) = p_X(x)$（随机变量的概率密度——下标$X$将省略），$\mathrm{E}[X]$读作随机变量$X$的*期望*（或$X$的*期望值*）。$x$的函数的期望值，即$f(x)$可以类似地由下式求得：

$$\mathrm{E}[f(X)] = \int_{-\infty}^{\infty} f(x)p(x)\,\mathrm{d}x \tag{A-271}$$

一个特别重要的函数是$f(x) = x^n$。这个函数导出随机变量的一般矩，即：

$$\mathrm{E}[X^n] = \overline{X^n} = \int_{-\infty}^{\infty} x^n p(x)\,\mathrm{d}x \tag{A-272}$$

当式（A-272）中的$n = 1$时，可求出前面讨论的平均值；当$n = 2$时，可求得均方值：

$$\mathrm{E}[X^2]=\overline{X^2}=\int_{-\infty}^{\infty}x^2\rho(x)\,\mathrm{d}x \tag{A-273}$$

随机变量的中心矩也非常重要，它定义为随机变量和它的均值的差的矩。因此，第$n$阶中心矩定义为：

618

$$\mathrm{E}[(X-\overline{X})^n]=\overline{(X-\overline{X})^n}=\int_{-\infty}^{\infty}(x-\overline{X})^n\rho(x)\,\mathrm{d}x \tag{A-274}$$

从式（A-274）我们看出第一阶中心矩（$n=1$）为0。第二阶中心矩（$n=2$）有一个特殊的名字，*方差*（$\sigma^2$），由式（A-274），它可以写作：

$$\sigma^2=\mathrm{E}[(X-\overline{X})^2]=\overline{(X-\overline{X})^2}=\int_{-\infty}^{\infty}(x-\overline{X})^2\rho(x)\,\mathrm{d}x \tag{A-275}$$

该方差也可以写作：

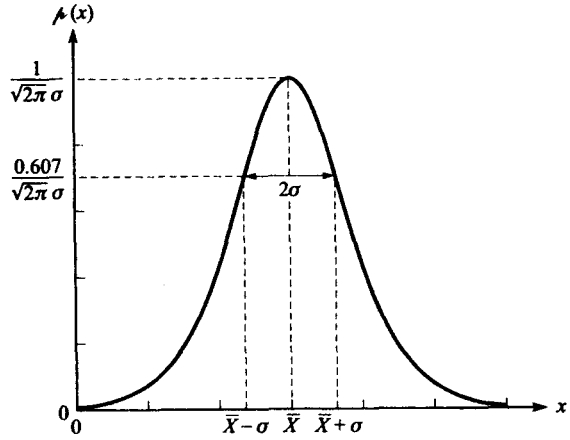
$$\begin{aligned}\sigma^2&=\mathrm{E}[(X-\overline{X})^2]=\mathrm{E}[X^2-2X\overline{X}+\overline{X}^2]\\&=\mathrm{E}[X^2]-2\mathrm{E}[X]\overline{X}+\overline{X}^2=\overline{X^2}-2\overline{X}\,\overline{X}+\overline{X}^2=\overline{X^2}-\overline{X}^2\end{aligned} \tag{A-276}$$

因此，从式（A-276）我们看出随机变量的方差可以表示为均方值与均值的平方之间的差。此外，方差的平方根$\sigma$称作*标准差*。

特别考虑高斯（或正态）密度函数。高斯密度函数的数学表达式为：

$$\rho(x)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left[\frac{-(x-\overline{X})^2}{2\sigma^2}\right]\qquad -\infty<x<\infty \tag{A-277}$$

其中$\overline{X}$是均值，$\sigma^2$是方差。图A-9表示了高斯随机变量的概率密度函数。




图A-9　高斯随机变量的概率密度函数

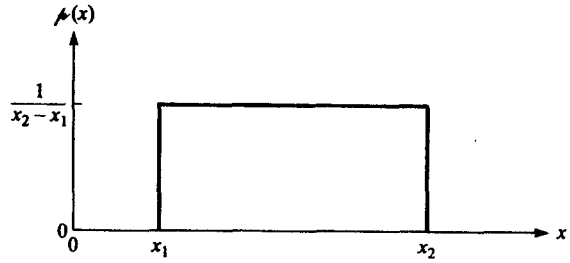
均匀概率密度函数可以写作：

619

$$\rho(x)=\begin{cases}\dfrac{1}{x_2-x_1} & x_1<x\leqslant x_2\\ 0 & \forall\ \text{其他}x\end{cases}$$

见图A-10。

均值可以用式（A-270）计算如下：




图A-10 均匀概率密度函数

$$\overline{X} = \mathrm{E}[X] = \int_{-\infty}^{\infty} x\, p(x)\mathrm{d}x = \int_{x_1}^{x_2} x \frac{1}{x_2 - x_1}\mathrm{d}x = \frac{1}{x_2 - x_1}\frac{x^2}{2}\Big|_{x_1}^{x_2}$$
$$= \frac{1}{x_2 - x_1}\frac{x_2^2 - x_1^2}{2} = \frac{x_1 + x_2}{2} \tag{A-278}$$

均方值可以由式（A-273）计算如下：

$$\overline{X^2} = \mathrm{E}[X^2] = \int_{-\infty}^{\infty} x^2 p(x)\,\mathrm{d}x = \int_{x_1}^{x_2} x^2 \frac{1}{x_2 - x_1}\mathrm{d}x = \frac{1}{x_2 - x_1}\frac{x^3}{3}\Big|_{x_1}^{x_2}$$
$$= \frac{1}{x_2 - x_1}\frac{x_2^3 - x_1^3}{3} = \frac{1}{3}(x_2^2 + x_1 x_2 + x_1^2) \tag{A-279}$$

方差由式（A-276）、式（A-278）和式（A-279）给出如下：

$$\sigma^2 = \overline{X^2} - \overline{X}^2 = \frac{(x_2 - x_1)^2}{12} \tag{A-280}$$

### A.7.5 随机过程

假定随机试验以及其结果$\alpha$构成一个样本空间$\mathscr{S}$，其中$\mathscr{S}$的子集称作事件和这些事件的概率。对于每一个结果$\alpha$我们可以赋予时间函数$X(t, \alpha)$（根据已定义的规则）。对于每一个$\alpha$，$X(t, \alpha)$形成一个函数族，这个族称为随机过程。因此，一个随机过程是两个变量$t$（时间）和$\alpha$（随机试验结果）的函数，$X(t, \alpha)$的四种不同情形描述如下：

1. 时间函数的族（$t$和$\alpha$变量）。
2. 随机变量（$t$固定，$\alpha$是变量）。
3. 单一时间函数（$t$为变量，$\alpha$固定），称作随机过程的实现或采样路径。
4. 单个数（$t$和$\alpha$固定）。

620

通常，符号$X(t)$用于描述随机过程，这样，忽略对$\alpha$的依赖性（这通常可以从上下文得到）。

对于一个特定的$t$, $X(t)$是一个随机变量，该随机变量的分布函数通常依赖于$t$：

$$\mathscr{P}(x;t) = \Pr\{X(t) \leqslant x\} \tag{A-281}$$

假定两个实数$x$, $t$，函数$\mathscr{P}(x;t)$等于事件$\{X(t) \leqslant x\}$的概率，$\{X(t) \leqslant x\}$包含这个过程在特定时间$t$所有不超过$x$的函数$X(t)$的值$\alpha$。函数$\mathscr{P}(x;t)$称作过程$X(t)$的一阶分布。相应的密度函数可以由对$x$进行分布函数的微分得到：

$$p(x;t) = \frac{\partial \mathscr{P}(x;t)}{\partial x} \tag{A-282}$$

在两个不同时间$t_1$和$t_2$，我们可以考虑随机变量$X(t_1)$和$X(t_2)$。其联合分布函数依赖于$t_1$和$t_2$，可以写作：

$$\mathscr{P}(x_1, x_2; t_1, t_2) = \mathscr{P}\{X(t_1) \leqslant x_1, X(t_2) \leqslant x_2\} \tag{A-283}$$

函数$\mathscr{P}(x_1, x_2; t_1, t_2)$称作过程$X(t)$的二阶分布。相应的密度函数为：

$$p(x_1, x_2; t_1, t_2) = \frac{\partial^2 \mathscr{P}(x_1, x_2; t_1, t_2)}{\partial x_1 \, \partial x_2} \tag{A-284}$$

同时，

$$p(x_1; t_1) = \int_{-\infty}^{\infty} p(x_1, x_2; t_1, t_2) \, \mathrm{d}x_2 \tag{A-285}$$

和

$$\mathscr{P}(x_1, \infty; t_1, t_2) = \mathscr{P}(x_1; t_1) \tag{A-286}$$

此外，条件密度由下式给出：

$$p(x_1, t_1 \mid X_2(t_2) = x_2) = \frac{p(x_1, x_2; t_1, t_2)}{p(x_2; t_2)} \tag{A-287}$$

*平稳、非平稳和宽平稳过程*

若一个特定过程的所有边缘和联合密度函数不依赖于时间原点的选择，这个过程称为*平稳的*。因此，所有相应的均值和矩是常数，不依赖于时间。若概率密度函数随时间原点变化，这个过程是*非平稳的*。因此，一个或更多的均值或矩也将依赖于时间。

621 严格意义上讲，平稳过程不存在。然而，在许多物理情况下，在观测的时间段内，过程变化并不大。因此，一个更宽松的要求是过程的均值是一个常数，它的自相关性依赖于时间差$t_2 - t_1$；那么，这个过程称为*宽平稳*（wss）。一般情况下，没有必要去区分平稳过程和宽平稳过程。

*遍历随机过程和非遍历随机过程*

一些平稳随机过程有这样的性质：总体的几乎每一个成员都表现出与总体相同的统计行为。在这些情况下，只通过分析一种典型的样本函数来决定统计行为是可能的。这些过程称为*遍历的*。对于遍历过程，均值和矩可以由时间平均值和总平均值决定。例如，第$n$阶一般矩可以由下式求出：

$$\mathrm{E}[X^n] = \overline{X^n} = \int_{-\infty}^{\infty} x^n p(x) \, \mathrm{d}x = \lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{T} X^n(t) \, \mathrm{d}t \tag{A-288}$$

若一个过程不具有式（A-288）的属性称作非遍历的。

*均值、自相关和自协方差函数*

随机过程$X(t)$的均值$m_X(t)$是$X(t)$期望值，即，

$$m_X(t) = \mathrm{E}[X(t)] = \int_{-\infty}^{\infty} x \, p(x; t) \, \mathrm{d}x \tag{A-289}$$

若$X(t)$是一个随机过程的样本函数，我们假定两个时间$t_1$和$t_2$，考虑随机变量$X(t_1) = X_1$和$X(t_2) = X_2$。那么，*自相关函数*$R_X(t_1, t_2)$可以定义为：

$$R_X(t_1, t_2) = \mathrm{E}[X_1 \, X_2] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_1 x_2 p(x_1, x_2; t_1, t_2) \, \mathrm{d}x_1 \, \mathrm{d}x_2 \tag{A-290}$$

这个定义对于平稳和非平稳的随机过程都是适用的。然而，我们一般对平稳过程感兴趣。因此，式（A-290）可以简化。对于宽平稳过程，所有的总体平均独立于时间原点，因此，

$$R_X(t_1, t_2) = R_X(t_1 + T, t_2 + T) = \mathrm{E}\,[X(t_1 + T)X(t_2 + T)] \tag{A-291}$$

既然等式是独立于所选的时间原点的，我们可以令$T = -t_1$，式（A-291）可以写作：

$$R_X(t_1, t_2) = R_X(0, t_2 - t_1) = \mathrm{E}\,[X(0)X(t_2 - t_1)] \tag{A-292}$$

因为这个表达式只依赖于时间差$t_2 - t_1$，我们令$\tau = t_2 - t_1$，式（A-292）写作：

$$R_X(\tau) = R_X(t_2 - t_1) = \mathrm{E}\,[X(t_1)X(t_1 + \tau)] \tag{A-293}$$

从式（A-293）我们看出，自相关函数仅依赖于$\tau$而不是$t_1$。因为自相关函数不依赖于宽平稳过程求取总体平均值的具体时间$t_1$，我们把式（A-293）写作： 622

$$R_X(\tau) = \mathrm{E}\,[X(t)X(t + \tau)] \tag{A-294}$$

一个时间自相关函数可以定义为特殊的样本函数：

$$\mathscr{R}_x(\tau) = \lim_{T\to\infty} \frac{1}{2T}\int_{-T}^{T} x(t)x(t + \tau)\,\mathrm{d}t \tag{A-295}$$

对于遍历随机过程的特殊情况，$\mathscr{R}_x(\tau)$与每一个$x(t)$相同，且等于$R_X(\tau)$，即：

$$\mathscr{R}_x(\tau) = R_X(\tau) \quad \text{对于一个遍历过程} \tag{A-296}$$

平稳过程的自相关函数的常规性质如下：

1. $R_X(0) = \overline{X^2}$。在自相关函数中，随机过程的均方值可以通过令$\tau = 0$获得。
2. $R_X(\tau) = R_X(-\tau)$。自相关函数是$\tau$的偶函数。
3. $|R_X(\tau)| \leqslant R_X(0)$。自相关函数的最大值总是出现在$\tau = 0$。
4. 若随机过程$X(t)$有均值，那么$R_X(\tau)$将有一个常量成分。
5. 若$X(t)$有一个周期成分，那么$R_X(\tau)$将也有一个相同的周期成分。
6. 若$\{X(t)\}$是遍历的、零均值，无周期成分，那么$\lim_{|\tau|\to\infty} R_X(\tau) = 0$。

当$\tau$变大时，随机变量趋向于统计上独立，因为随时间推移，过去值的作用逐渐“消失”。

随机过程$X(t)$的自协方差函数是随机变量$X(t_1)$和$X(t_2)$的协方差，即，

$$C_X(t_1, t_2) = \mathrm{E}\{[X(t_1) - m_X(t_1)][X(t_2) - m_X(t_2)]\} \tag{A-297}$$

自协方差函数可以直接写作：

$$C_X(t_1, t_2) = R_X(t_1, t_2) - m_X(t_1)m_X(t_2) \tag{A-298}$$

从式（A-298）很容易看出：若$X(t)$是零均值的，自相关函数和自协方差函数对该过程是相同的。

$X(t)$的方差如下：

$$\sigma^2_{X(t)} = \mathrm{E}\{[X(t) - m_X(t)]^2\} = C_X(t, t) = R_X(t, t) - m_X^2(t) \tag{A-299}$$

$X(t)$的相关系数定义为$X(t_1)$和$X(t_2)$的相关系数： 623

$$\rho_X(t_1, t_2) = \frac{C_X(t_1, t_2)}{\sqrt{C_X(t_1, t_1)}\sqrt{C_X(t_2, t_2)}} \tag{A-300}$$

相关系数是随机变量可以预测为另一个的线性函数的程度的度量。

### A.7.6 向量随机过程

假设$x_1(t), x_2(t), \cdots, x_n(t)$是$n$个标量随机过程（可能相互独立），那么

$$\boldsymbol{x}(t) = [x_1(t), x_2(t), \cdots, x_n(t)]^T \tag{A-301}$$

称作向量随机过程。向量随机过程的均值如下

$$\boldsymbol{m}_x(t) = \mathrm{E}[\boldsymbol{x}(t)] \tag{A-302}$$

相关矩阵为：

$$\boldsymbol{R}_x(t_1, t_2) = \mathrm{E}\{\boldsymbol{x}(t_1)\boldsymbol{x}^T(t_2)\} \tag{A-303}$$

以及协方差矩阵为：

$$\boldsymbol{C}_x(t_1, t_2) = \mathrm{E}\{[\boldsymbol{x}(t_1) - \boldsymbol{m}_x(t_1)][\boldsymbol{x}(t_2) - \boldsymbol{m}_x(t_2)]^T\} \tag{A-304}$$

若$\boldsymbol{x}(t)$是宽平稳随机过程，式（A-302）的均值$\boldsymbol{m}_x$是常量，式（A-303）的相关矩阵$\boldsymbol{R}_x(t, t)$对所有的$t$是有限的，式（A-304）的协方差矩阵只依赖于$t_2 - t_1$，即$\boldsymbol{C}_x(t_2 - t_1)$。

高斯（正态）向量随机过程$\boldsymbol{x}$是一个向量随机过程，$n$个随机变量的集合，其联合概率分布是高斯的。相应的对于$\boldsymbol{x}$的概率密度函数由下式给出

$$p_x(\xi) = \frac{1}{(2\pi)^{n/2}[\det(\boldsymbol{C}_x)]^{1/2}} \exp\left[\frac{-(\xi - \boldsymbol{m}_x)^T \boldsymbol{C}_x^{-1}(\xi - \boldsymbol{m}_x)}{2}\right] \tag{A-305}$$

其中$\xi$是$\boldsymbol{x}$所允许的值，并假定协方差矩阵$C_x$的逆存在。

### A.7.7 功率谱密度函数和功率谱密度矩阵

宽平稳随机过程的自相关函数的傅里叶变换称作*功率谱密度*函数，记作$S_x(\omega)$。假定wss随机过程$x(t)$，功率谱密度函数给出如下：

624

$$S_x(\omega) = \int_{-\infty}^{\infty} R_x(\tau) e^{-j\omega\tau} \mathrm{d}\tau \tag{A-306}$$

该随机过程的均方值可以由功率谱密度函数求出：

$$\overline{x^2} = \frac{1}{2\pi}\int_{-\infty}^{\infty} S_x(\omega)\,\mathrm{d}\omega \tag{A-307}$$

在式（A-301）中，若$\boldsymbol{x}(t)$是一个向量值wss随机过程，带有相关矩阵$\boldsymbol{R}_x(\tau)$，功率谱密度矩阵可以写作；

$$\boldsymbol{S}_x(\omega) = \int_{-\infty}^{\infty} \boldsymbol{R}_x(\tau) e^{-j\omega\tau} \mathrm{d}\tau \tag{A-308}$$

功率谱密度矩阵的常用性质如下：

1. $\boldsymbol{S}_x(-\omega) = \boldsymbol{S}_x^T(\omega)$, $\forall\omega$
2. $\boldsymbol{S}_x^*(\omega) = \boldsymbol{S}_x(\omega)$, $\forall\omega$ （星号表示复共轭转置）
3. $S_x(\omega) \geqslant 0$, $\forall\omega$ [即，$S_x(\omega)$是半正定的，或非负定的]

*指数相关噪声例子*

我们将考虑标量wss随机过程$x(t)$，自相关函数为：

$$R_x(\tau) = \sigma^2 e^{-|\tau|/\theta} \qquad \theta > 0 \tag{A-309}$$

功率谱密度函数可以由$\boldsymbol{R}_x(\tau)$进行傅里叶变换求得如式（A-306）：

$$S_x(\omega)=\sigma^2\int_{-\infty}^{\infty}e^{-|\tau|/\theta}e^{-j\omega\tau}\mathrm{d}\tau=\sigma^2\int_{-\infty}^{0}e^{\tau/\theta}e^{-j\omega\tau}\mathrm{d}\tau+\sigma^2\int_{0}^{\infty}e^{-\tau/\theta}e^{-j\omega\tau}\mathrm{d}\tau=\frac{2\sigma^2\theta}{1+\omega^2\theta^2}\quad\theta>0 \tag{A-310}$$

### A.7.8 白噪声驱动的线性系统和谱因子分解

*白噪声驱动的线性系统*

考虑线性非时变系统：

$$\begin{aligned}\dot{\boldsymbol{x}}(t)&=\boldsymbol{A}\boldsymbol{x}(t)+\boldsymbol{B}\boldsymbol{w}(t)\\ \boldsymbol{x}(0)&=\boldsymbol{x}_0\end{aligned} \tag{A-311}$$

其中$\boldsymbol{w}(t)$是常量密度$V$的白噪声，即$\boldsymbol{w}(t)$是功率谱密度矩阵如下的宽平稳过程：

$$\boldsymbol{S}_w(\omega)=\boldsymbol{V} \tag{A-312}$$

假定$\boldsymbol{A}$是渐近稳定的（参照A.2.12节）。式（A-311）中的初始条件向量$\boldsymbol{x}_0$是一个独立于$\boldsymbol{w}(t)$的随机变量，均值为$\boldsymbol{m}_0$, $\boldsymbol{Q}_0=\mathrm{E}[(x_0-\boldsymbol{m}_0)(x_0-\boldsymbol{m}_0)^T]$为方差矩阵。那么，$\boldsymbol{x}(t)$的均值： 625

$$\boldsymbol{m}_x(t)=\phi(t)\boldsymbol{m}_0=e^{\boldsymbol{A}t}\boldsymbol{m}_0 \tag{A-313}$$

其中$\phi(t)=e^{\boldsymbol{A}t}$是状态转换矩阵（参照A.2.12节）。$\boldsymbol{x}(t)$相应的稳定状态方差矩阵可以由稳定状态的代数李雅普诺夫方程：

$$\boldsymbol{A}\boldsymbol{Q}+\boldsymbol{Q}\boldsymbol{A}^T+\boldsymbol{B}\boldsymbol{V}\boldsymbol{B}^T=\boldsymbol{0} \tag{A-314}$$

求得$\boldsymbol{Q}$[13]。稳定状态方差矩阵也可由[13]得出

$$\boldsymbol{Q}=\int_0^{\infty}e^{\boldsymbol{A}\tau}\boldsymbol{B}\boldsymbol{V}\boldsymbol{B}^Te^{\boldsymbol{A}^T\tau}\mathrm{d}\tau \tag{A-315}$$

或从[13]得出

$$\boldsymbol{Q}=\int_{-\infty}^{\infty}(j\omega\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{V}\boldsymbol{B}^T(-j\omega\boldsymbol{I}-\boldsymbol{A}^T)^{-1}\mathrm{d}f \tag{A-316}$$

其中$\omega=2\pi f$ rad/s。

*谱因子分解*

考虑传递函数矩阵为$\boldsymbol{H}(s)$的渐近稳定线性非时变系统，其中$s$是拉普拉斯变量。若对系统的输入是一个wss随机过程$\boldsymbol{u}(t)$的实现，功率谱密度矩阵为$\boldsymbol{S}_u(\omega)$，那么系统的输出是wss随机过程$\boldsymbol{y}(t)$的实现，其功率谱密度矩阵为[13]：

$$\boldsymbol{S}_y(\omega)=\boldsymbol{H}(j\omega)\boldsymbol{S}_u(\omega)\boldsymbol{H}^T(-j\omega) \tag{A-317}$$

其中，$\boldsymbol{H}(j\omega)=\boldsymbol{H}(s)|_{s=j\omega}$是正弦稳定状态传递函数矩阵。对于标量系统，转换函数为$H(s)$，标量wss随机输入$u(t)$，功率谱密度函数为$S_u(\omega)$，输出$y(t)$的功率谱密度函数为：

$$S_y(\omega)=H(j\omega)H(-j\omega)S_u(\omega) \tag{A-318}$$

换句话说，输出的功率谱密度函数与传递函数的大小的平方成比例。式（A-318）的结果可以用拉普拉斯变换表示：

$$S_y(s)=H(s)H(-s)S_u(s) \tag{A-319}$$

这个方法需要用到双边拉普拉斯变换[14]。

许多情况下只给定过程的功率谱密度，需要建立一个随机过程的模型。更具体地讲，给

626 定wss随机过程的功率谱密度函数，当修正过滤器输入白噪声时，输出是wss随机过程（定义了功率谱密度函数）的实现，修正过滤器的特征是什么？答案实际很简单，方法很直接，称作谱因子分解[14-16]。这个方法包括分解这个过程的功率谱密度函数为正时和负时部分。正时部分是需要的修正过滤器，这样，当白噪声进入时，响应是随机过程的实现。这可以从式（A-318）看出。若修正过滤器的输入是零均值，单位方差白噪声，输出的功率谱密度是$S_y(\omega) = H(j\omega)H(-j\omega)$，因为白噪声的谱密度为$S_w(\omega) = 1$。记住白噪声的自相关为$R_w(\tau) = \delta(\tau)$。因此，$H(j\omega)$是所期望的修正过滤器正弦稳定状态传递函数，其一定是从$S_y(\omega)$分解出来的。下面的例子将解释这个过程。

在第一个例子中，过程的功率谱密度是符合式（A-310）中的指数相关噪声的，即：

$$S_y(\omega) = \frac{2\sigma^2\theta}{1+\omega^2\theta^2} \qquad \theta > 0 \tag{A-320}$$

这个函数可以写作（分解）：

$$S_y(\omega) = \underbrace{\frac{\sqrt{2\sigma^2\theta}}{1+j\omega\theta}}_{\text{正时部分}} \underbrace{\frac{\sqrt{2\sigma^2\theta}}{1-j\omega\theta}}_{\text{负时部分}} \tag{A-321}$$

因此，当由白噪声驱动时，修正过滤器的正弦稳定状态传递函数将使它的输出为期望的随机过程。这个传递函数由下式给出

$$H(j\omega) = \frac{\sqrt{2\sigma^2\theta}}{1+j\omega\theta} \tag{A-322}$$

以拉普拉斯变换形式表示的传递函数：

$$H(s) = \frac{\sqrt{2\sigma^2\theta}}{1+\theta s} \tag{A-323}$$

当指数相关噪声过程的功率谱密度是以拉普拉斯变换表示时，式（A-323）的结果可以由式（A-319）直接得出，即：

$$S_y(s) = \frac{2\sigma^2\theta}{1-\theta^2 s^2} \tag{A-324}$$

在第二个例子中，随机过程的功率谱密度函数（以拉普拉斯变换形式）表示如下：

$$S_y(s) = \frac{-s^2+1}{s^2+64} \tag{A-325}$$

627 对分母多项式完全平方，得到根$-2 \pm j2$和$2 \pm j2$的集合，分解如下：

$$S_y(s) = \underbrace{\frac{s+1}{s^2+4s+8}}_{\text{正时部分}} \underbrace{\frac{-s+1}{s^2-4s+8}}_{\text{负时部分}} \tag{A-326}$$

因此，修正过滤器的传递函数为：

$$H(s) = \frac{s+1}{s^2+4s+8} \tag{A-327}$$

## A.8 模糊集合论

一个典型的集合$A$（参照A.3.1节）定义为元素或对象的集合$x \in X$，每个$x$可属于也可不属于集合$A$，$A \subset X$（即，$A$是$X$的子集）。通过对$A$中的每一个元素$x$定义特征函数（或成员函数），一个典型的集合可以表示为有序对（$x$, 0）或（$x$, 1）的集合，其中1表示成员关系，0表示非成员关系。特征函数也称作指示器函数，定义如下：

$$I_A(x) = \begin{cases} 1 & \text{如果 } x \in A \\ 0 & \text{如果 } x \notin A \end{cases} \tag{A-328}$$

两个集合$A$和$B$的交或并的指示器函数可以很容易地以集合$A$的指示器函数和集合$B$的指示器函数表示：

$$I_{A\cap B}(x) = \min[I_A(x), I_B(x)] \tag{A-329}$$

$$I_{A\cup B}(x) = \max[I_A(x), I_B(x)] \tag{A-330}$$

类似地，集合$A$的补集（即，$\tilde{A}$）的指示器函数由下式给出：

$$I_{\tilde{A}}(x) = 1 - I_A(x) \tag{A-331}$$

$A$为$B$的子集的条件表示如下：

$$A \subset B \Leftrightarrow I_A(x) \leqslant I_B(x) \quad \forall x \in X \tag{A-332}$$

不同于上面描述的传统意义上的集合，泛集$X$在一定程度上是模糊集$A$的一个元素。这样，特征函数可以表示一个给定模糊集的元素的成员的度，使其在0和1之间取值。若$X$是一般表示为$\{x\}$的对象集，则$X$中的模糊集$A$定义为如下有序对的集合：

$$A = \{(x, m_A(x)) | x \in X\} \tag{A-333}$$

其中$m_A(x)$表示在$A$中$x$的成员函数，把$X$映射到[0, 1]区间。一般，$X$是有序集合，集合$A$可以仅用成员函数值说明。当$m_A(x)$只取值0和1时，$A$是非模糊的（或易碎的），$m_A(x)$等同于非模糊集的特征函数。 628

Zadeh[17]通过同标准集精确地相似的表达，定义了模糊集$A$和$B$的模糊交集（MIN）和模糊并集（MAX），以及$A$的补集。以类似的方式，Zadeh提出使用$A$和$B$的成员函数定义$A$为$B$的模糊子集，即，

$$A \subset B \Leftrightarrow m_A(x) \leqslant m_B(x) \quad \forall x \in X \tag{A-334}$$

这称为主成员函数。例如，若$A = \{0.3, 0.0, 0.7\}$, $B = \{0.4, 0.7, 0.9\}$，则$A$是$B$的模糊子集，但$B$不是$A$的模糊子集。

## A.9 部分三角恒等式

毕达哥拉斯公式

$$\sin^2\alpha + \cos^2\alpha = 1$$

$$1 + \tan^2\alpha = \sec^2\alpha$$

$$1 + \cot^2\alpha = \csc^2\alpha$$

互反公式

$$\sin\alpha = \frac{1}{\csc\alpha} \qquad \cos\alpha = \frac{1}{\sec\alpha} \qquad \tan\alpha = \frac{1}{\cot\alpha}$$

$$\csc\alpha=\frac{1}{\sin\alpha}\qquad \sec\alpha=\frac{1}{\cos\alpha}\qquad \cot\alpha=\frac{1}{\tan\alpha}$$

商公式

$$\sin\alpha=\frac{\tan\alpha}{\sec\alpha}\qquad \cos\alpha=\frac{\cot\alpha}{\csc\alpha}\qquad \tan\alpha=\frac{\sin\alpha}{\cos\alpha}$$
$$\csc\alpha=\frac{\sec\alpha}{\tan\alpha}\qquad \sec\alpha=\frac{\csc\alpha}{\cot\alpha}\qquad \cot\alpha=\frac{\cos\alpha}{\sin\alpha}$$

629

乘积公式

$$\sin\alpha=\tan\alpha\cos\alpha\qquad \cos\alpha=\cot\alpha\sin\alpha\qquad \tan\alpha=\sin\alpha\sec\alpha$$
$$\cot\alpha=\cos\alpha\csc\alpha\qquad \sec\alpha=\csc\alpha\tan\alpha\qquad \csc\alpha=\sec\alpha\cot\alpha$$

$$\sin\alpha\sin\beta=\frac{1}{2}\cos(\alpha-\beta)-\frac{1}{2}\cos(\alpha+\beta)$$
$$\cos\alpha\cos\beta=\frac{1}{2}\cos(\alpha-\beta)+\frac{1}{2}\cos(\alpha+\beta)$$
$$\sin\alpha\cos\beta=\frac{1}{2}\sin(\alpha+\beta)+\frac{1}{2}\sin(\alpha-\beta)$$
$$\cos\alpha\sin\beta=\frac{1}{2}\sin(\alpha+\beta)-\frac{1}{2}\sin(\alpha-\beta)$$

角的和差公式

$$\sin(\alpha+\beta)=\sin\alpha\cos\beta+\cos\alpha\sin\beta$$
$$\sin(\alpha-\beta)=\sin\alpha\cos\beta-\cos\alpha\sin\beta$$
$$\cos(\alpha+\beta)=\cos\alpha\cos\beta-\sin\alpha\sin\beta$$
$$\cos(\alpha-\beta)=\cos\alpha\cos\beta+\sin\alpha\sin\beta$$
$$\tan(\alpha+\beta)=\frac{\tan\alpha+\tan\beta}{1-\tan\alpha\tan\beta}$$
$$\tan(\alpha-\beta)=\frac{\tan\alpha-\tan\beta}{1+\tan\alpha\tan\beta}$$
$$\sin(\alpha+\beta)\sin(\alpha-\beta)=\sin^2\alpha-\sin^2\beta=\cos^2\beta-\cos^2\alpha$$
$$\cos(\alpha+\beta)\cos(\alpha-\beta)=\cos^2\alpha-\sin^2\beta=\cos^2\beta-\sin^2\alpha$$

630

倍角公式

$$\sin2\alpha=2\sin\alpha\cos\alpha=\frac{2\tan\alpha}{1+\tan^2\alpha}$$
$$\cos2\alpha=\cos^2\alpha-\sin^2\alpha=2\cos^2\alpha-1=1-2\sin^2\alpha=\frac{1-\tan^2\alpha}{1+\tan^2\alpha}$$
$$\tan2\alpha=\frac{2\tan\alpha}{1-\tan^2\alpha}\qquad \cot2\alpha=\frac{\cot^2\alpha-1}{2\cot\alpha}$$

乘方公式

$$\sin^2\alpha=\frac{1}{2}(1-\cos2\alpha)\qquad \sin^3\alpha=\frac{1}{4}(3\sin\alpha-\sin3\alpha)$$

$$\sin^4\alpha=\frac{1}{8}(3-4\cos 2\alpha+\cos 4\alpha)$$

$$\cos^2\alpha=\frac{1}{2}(1+\cos 2\alpha)\qquad \cos^3\alpha=\frac{1}{4}(3\cos\alpha+\cos 3\alpha)$$

$$\cos^4\alpha=\frac{1}{8}(3+4\cos 2\alpha+\cos 4\alpha)$$

$$\tan^2\alpha=\frac{1-\cos 2\alpha}{1+\cos 2\alpha}\qquad \cot^2\alpha=\frac{1+\cos 2\alpha}{1-\cos 2\alpha}$$

半角公式

$$\sin\frac{\alpha}{2}=\pm\sqrt{\frac{1-\cos\alpha}{2}}\qquad \cos\frac{\alpha}{2}=\pm\sqrt{\frac{1+\cos\alpha}{2}}$$

$$\tan\frac{\alpha}{2}=\pm\sqrt{\frac{1-\cos\alpha}{1+\cos\alpha}}=\frac{1-\cos\alpha}{\sin\alpha}=\frac{\sin\alpha}{1+\cos\alpha}$$

$$\cot\frac{\alpha}{2}=\pm\sqrt{\frac{1+\cos\alpha}{1-\cos\alpha}}=\frac{1+\cos\alpha}{\sin\alpha}=\frac{\sin\alpha}{1-\cos\alpha}$$

欧拉公式

$$e^{\pm j\alpha}=\cos\alpha\pm j\sin\alpha\qquad j=\sqrt{-1}$$

$$\sin\alpha=\frac{e^{j\alpha}-e^{-j\alpha}}{2j}\qquad \cos\alpha=\frac{e^{j\alpha}+e^{-j\alpha}}{2}$$

$$\tan\alpha=-j\left(\frac{e^{j\alpha}-e^{-j\alpha}}{e^{j\alpha}+e^{-j\alpha}}\right)=-j\left(\frac{e^{j2\alpha}-1}{e^{j2\alpha}+1}\right)$$

函数和与函数差公式

$$\sin\alpha+\sin\beta=2\sin\frac{1}{2}(\alpha+\beta)\cos\frac{1}{2}(\alpha-\beta)$$

$$\sin\alpha-\sin\beta=2\cos\frac{1}{2}(\alpha+\beta)\sin\frac{1}{2}(\alpha-\beta)$$

$$\cos\alpha+\cos\beta=2\cos\frac{1}{2}(\alpha+\beta)\cos\frac{1}{2}(\alpha-\beta)$$

$$\cos\alpha-\cos\beta=-2\sin\frac{1}{2}(\alpha+\beta)\sin\frac{1}{2}(\alpha-\beta)$$

$$\tan\alpha+\tan\beta=\frac{\sin(\alpha+\beta)}{\cos\alpha\cos\beta}\qquad \tan\alpha-\tan\beta=\frac{\sin(\alpha-\beta)}{\cos\alpha\cos\beta}$$

631

## 参考文献


1. G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Balitmore, MD: Johns Hopkins University Press, 1996.
2. C. T. Chen, *Linear System Theory and Design*, New York: Holt, Rinehart and Winston, 1984.
3. C. V. Churchill, J. W. Brown, and R. F. Verhey, *Complex Variables and Applications*, 3rd ed., New York: McGraw-Hill, 1976.
4. J. W. Brewer,"Kronecker Products and Matrix Calculus in System Theory,"

*IEEE Transactions on Circuits and Systems*, vol. CAS-25, 1978, pp. 772–81.
5. R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, New York: Cambridge University Press, 1991.
6. K. Zhou, J. C. Doyle, and K. Glover, *Robust and Optimal Control*, Upper Saddle River, NJ: Prentice-Hall, 1996.
7. N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, Philadelphia, PA: Society for Industrial and Applied Mathematics, 1996.
8. W. D. Wallis, "Hadamard Matrices," in *Combinatorial and Graph-Theoretical Problems in Linear Algebra*, eds. R. A. Brualdi, S. Friedland, and V. Klee, vol. 50, New York: Springer-Verlag, 1993, pp. 235–43.
9. S. G. Nash and A. Sofer, *Linear and Nonlinear Programming*, New York: McGraw-Hill, 1996.
10. D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd ed., Reading, MA: Addison-Wesley, 1984.
11. M. Avriel, *Nonlinear Programming: Analysis and Methods*, Englewood Cliffs, NJ: Prentice-Hall, 1976.
12. A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 3rd ed., New York: McGraw-Hill, 1991.
13. H. Kwakernaak and R. Sivan, *Linear Optimal Control Systems*, New York: Wiley-Interscience, 1972.
14. R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, 2nd ed., New York: Wiley, 1992.
15. B. Picinbono, *Random Signals and Systems*, Englewood Cliffs, NJ: Prentice-Hall, 1993.
16. T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms*, Upper Saddle River, NJ: Prentice-Hall, 2000.
632 17. L. A. Zadeh, "Fuzzy Sets," *Information and Control*, vol. 8, 1965, pp. 338–53.

# 主题索引

索引中的页码为英文原书页码，与书中页边标注的页码一致。

## A


Absolute correction（绝对修正），55
Abstract factor analysis（抽象因子分析），435
Accelerated learning backpropagation algorithms（加速学习反向传播算法），120-136
  backpropagation with adaptive slopes of activation functions（具有自适应激活函数斜度的反向传播），129-132
  conjugate gradient backpropagation for the feedforward multilayer perceptron（用于前馈多层感知器的共轭梯度反向传播），120-126
  Levenberg-Marquardt algorithm（L-M算法），132-136
  recursive least-squares-based backpropagation algorithm 66-67（基于最小二乘递归反向传播算法），126-129
Activation , Synchronous（激活，同步的），69
Activation functions（激活函数），
  adaptive slopes of（自适应斜度），129-132
  basic（基本），27-33
  identity function（恒等函数），27
  saturating linear function（饱和线性函数），29
  threshold function（阈值函数），27
Activity level（激活水平），26
Adaline（自适应线性单元），7,44-52
Adaptive computing , in the brain（自适应计算，在人脑中），4
Adaptive linear element(Adaline)（自适应线性单元），7,44-52
  linear error correction rules（线性误差修正规则），50-51
  linear separability（线性可分离性），46-48
  nonlinear weight correction rules（非线性权值修正规则），52
  with nonlinearly transformed inputs（具有非线性变换输入），49-50
Adaptive modular architecture（自适应模块体系结构），443
Adaptive principal-component extraction(APEX) algorithm（自适应主成分提取算法），411-418
Adaptive resonance theory(ART)（自适应共振理论）
  ART1，184-188
  fuzzy（模糊的），188-190
  neural networks（神经网络），182-190
Adaptive resonance theory mapping(ARTMAP),fuzzy（自适应共振理论映射，模糊），188-190
Adaptive slopes of activation functions, backpropagation algorithms with（激活函数的自适应斜度，反向传播算法），129-132
Additive noise, spectrum estimation of sinusoids in（加性噪声、正弦曲线的谱估计），519-528
Affine transformations（仿射变换），26
Algebra. 代数，参见linear algebra , Matrix algebra
Algebraic Lyapunov equation, neurocomputing approach for solving（代数李雅普诺夫方程，神经计算的解决方法），326-328
Algebraic Riccati equation , neurocomputing approach for solving（代数里卡蒂方程，神经计算的解决方法），329-334
Algorithms（算法）
  adaptive principal-component extraction（自适应主成分提取），411-418
  annealing-based global search（基于退火的全局搜索），213
  backpropagation（反向传播），7
    accelerated learning（加速学习），120-136
    learning（学习），106-119
    recursive least-squares-based（基于最小二乘递归），66-67, 126-129
  bigradient（双梯度），504
  Broyden-Fletcher-Goldfarb-Shanno，607
  conjugate gradient（共轭梯度），608-610
  decorrelating（去相关），400, 411
  discrete-time Hopfield network training（离散时间的霍普菲尔德网络训练），203
  fast fixed-point for ICA（用于ICA的快速固定点），512-519
  Fletcher-Reeves conjugate gradient（Fletcher-Reeves共轭梯度），352,609
  generalized Hebbian（广义Hebb），407-410
  learning,for neural network adaptive estimation of principal components（学习，用于神经网络自适应主成分估计），400-426

Levenberg-Marquardt, 132-136
LMS, 36-44
modified LMS (修正的LMS), 68
LQV1, 176
modified Newton method (改进的牛顿法), 606-607
modified relaxation (修正的松弛), 58
Newton's optimization (牛顿最优化), 132
Nguyen and Widrow's initialization (Nguyen初始化和Widrow初始化), 111
parallel APEX (并行的APEX), 411-414
PLSR1 Calibration (PLSR1校准), 436-439
PLSR1 prediction (PLSR1预测), 439-442
Quasi-Newton methods (拟牛顿法), 607
recursive least-squares version of the back propagation (反向传播的递归最小二乘形式), 127-129
reestimation (重新估计), 400
robust PLSNET calibration (鲁棒PLSNET校准), 450-454
robust learning algorithm for solving systems of equations with noise (用于求解具有噪声的方程组的鲁棒学习算法), 358
search-then-converge (搜索然后收敛), 41,115-116
simulated annealing global search (模拟退火全局搜索), 213
standard backpropagation (标准反向传播), 110
stochastic gradient ascent (随机梯度上升), 410-411
training (训练), 123-126
for training counter propagation networks (用于训练对传网络), 138
for training full propagation networks (用于训练完全传播网络), 140
vector matrix form of the backpropagation (反向传播的向量矩阵形式), 119
Analog artificial neural networks (类似人工神经网络), 342
Analysis.参见Independent-component analysis; Multivariable analysis; Principal-component analysis
Angle-sum and angle-difference formulas (角度和与角度差的公式), 630
Annealing (退火)
defined (退火定义), 211
simulated (模拟), 209-215
Annealing-based global search algorithm (基于退火的全局搜索算法), 213
ANN, 参见Artificial neural networks
Anti-Hebbian synaptic activity (反Hebbian突触活动), 71
Antidiagonals (反对角线), 586-587
APEX, 参见Adaptive principal-component extraction algorithm
Application phase (应用阶段), 293
Applications (应用), 243-632
case studies (案例研究), 529-539
estimation of glucose concentrations from synthetic NIR data (来自人工NIR数据的葡萄糖浓度估计), 529-534
event classification using infrasonic data (使用次声数据的事件分类), 534-539
for identification, control, and estimation (用于识别、控制和估计), 468-549
for optimization problems (最优化问题), 243-291
for solving linear algebraic equations (求解线性代数方程组), 342-394
for solving matrix algebra problems (求解矩阵代数问题), 292-341
statistical methods (统计方法), 395-467
Approximation (近似), 40,173
mean-field (平均场), 221
ARMA models (ARMA模型)
linear system identification with (线性系统识别), 470-473
nonlinear (非线性), 479-484
APT. *See* Adaptive resonance theory
APT1 (参看自适应共振理论ART1), 184-188
Artificial neural networks (人工神经网络), 3-5, 25
analog (类似物), 342
Artificial neurons (人工神经元)
basic models of (基本模型), 25-27
Hopfield model of (霍普菲尔德模型), 33-35
nonlinear (非线性), 25-26
ARTMAP. *See* Adaptive resonance theory mapping (参看自适应共振理论映射)
Association (联想), 3
Association area (联想区域), 56
Associative memory (联想记忆), 98,199
Hopfield (霍普菲尔德), 34,199-209
linear distributed, general (线性分布, 一般), 98-99
Associative memory networks (联想记忆网络), 97-106
correlation matrix (相关矩阵), 100-104
error correction approach for (误差修正方法), 104-106
Assumed system dimensions (假设系统维数), 473
Attractors (吸引子), 200
Augmented Lagrange multiplier methods (增广的拉格朗日乘子法), 281-286
Autoassociative memory (自联想记忆), 98
Autocorrelation function (自修正函数), 622
Autoregressive moving average models (自回归滑动平均模型), 469-470

Autovariance function（自方差函数），623
Average. *See* Moving average models（平均，参看滑动平均模型）

B

Backpropagation（反向传播）
  with adaptive slopes of activation functions（具有自适应斜度的激活函数），129-132
  conjugate gradient（共轭梯度），120-126
Backpropagation learning algorithms（反向传播学习算法），7,106-119
  accelerated（加速的），120-136
  batch updating（批量更新），114-115
    with variable learning rate（具有可变学习率），116
  for the feedforward multi-layer perceptron（前馈多层感知器），106-110
  with momentum updating（具有动量项的更新），113-114
  practical issues in using standard（在使用（标准算法）时的实际问题），110-113
  recursive least-squares-based（基于最小二乘递归），66-67,126-129
  search-then-converge method（搜索然后收敛方法），115-116
  vector-matrix form of（向量矩阵形式），117-119
Backpropagation learning rule（反向传播学习规则），36
Band matrices（带状矩阵），585-586
Batch learning（批量学习）488
Batch updating algorithms（批量更新算法），114-115
  case-dependent（范例依赖），115
  with variable learning rate（可变学习率），116
Bell-shaped curve（钟形曲线），168
BFGS. *See* Broyden-Fletcher-Goldfarb-Shanno algorithm（BFGS，参看BFGS算法）
Bigradient algorithm（双梯度算法），504
Binary function（二值函数），27
Binary sigmoid function（二值S形函数），30-31
Biological neural networks（生物神经网络），13-18
Bipolar function（双极函数），27
Bipolar sigmoid function（双极S形函数），31-33
Boltzmann-Gibbs distribution（玻尔兹曼-吉布斯分布），211-212
Boltzmann machine（玻尔兹曼机），215-221
Brain, as a computer（人脑，好像一个计算机），4
Broyden-Fletcher-Goldfarb-Shanno(BFGS) algorithm（BFGS算法），606

C

Calculus. *See* Matrix calculus（微积分，参看矩阵微积分）
Calibration ,of PLSNET（校准），442-446
Calibration algorithms（校准算法），436
Calibration model（校准模型），426
CAM. *See* Content-addressable memory（参看内容可寻址存储器）
Case studies（案例研究），529-539
  estimation of glucose concentrations from synthetic NIR data（来自人工NIR数据的葡萄糖浓度估计），529-534
  event classification using infrasonic data（利用次声数据的事件分类），534-539
Category representation field（类别表示域），182
Cauchy-Schwartz inequality（柯西－施瓦茨不等式），572
Cayley-Hamilton theorem（凯莱－哈密顿定理），564
Cell assemblies（单元集合），69
Centering ,mean（中心，均值），80
Central moments（中心矩），618
Cepstrum, complex（倒谱，复数的），83
Chain rule（链式规则），593-594
Chebyshev norm（切比雪夫范数），572
Chemometrics（化学计量学，化学统计学），395-396,426
Circulant matrices（循环矩阵），585
Class labels（类别标签），175
Classical least-squares(CLS)（经典的最小二乘法），427-429
  regression（回归），530-532
Classifier, maximum-likelihood Gaussian（分类器，最大似然高斯函数），56
CLS. *See* Classical least-squares（CLS, 参看经典的最小二乘法）
Co-occurrence, standard Hebbian（并发，标准Hebb规则），402
Coding process, complement（编码过程，补码），188
Collective emergent properties（集体涌现特性），199
Companion matrix（伴随矩阵），571
Complement coding process（补码过程），188
Complex cepstrum（复倒谱），83
Complex square matrices, summary of important properties for（复数方阵，重要性质小结），583-584
Concentration values（中心值），429
Condition number. *See* Matrix condition number（条件数，参看矩阵条件数）
Conditional probabilities（条件概率），614-615
Configuration network（结构网络），111-112
Conjugate gradient method（共轭梯度方法），608-610

backpropagation for the feedforward multilayer perceptron (前馈多层感知器的反向传播), 120-126
applied to solving normal equations (用于求解正规方程组), 122-123
normal equations for the linear combiner (线性组合器的正规方程), 121-122
training algorithm (训练算法), 123-126
learning rule for solving systems of linear equations (求解线性方程组的学习规则), 351-354
Conjugate vectors ,orthogonal and unitary matrices and (共轭向量、正交矩阵和酉矩阵), 560-561
Conjunctional mechanisms (连接机制), 71
Connection weights (连接权值), 412-413
Constant ,search time (常量, 搜索时间), 41
Constrained nonlinear programming (约束非线性规划), 610-613
Kuhn-Tucker conditions (库恩-塔克条件), 610-611
Lagrange multiplier methods (拉格朗日乘子方法), 611-613
Content-addressable memory(CAM) (按内容可寻址存储器), 34,199
Context units (上下文单元), 222
Continuous derivatives (连续导数), 59
Control (控制)
direct *vs.* indirect (直接与间接), 493-494
neurocomputing applications for (神经计算应用), 468-549
of nonlinear dynamic systems (非线性动力系统), 484-499
autoregressive moving average models (自回归滑动平均模型), 469-470
identification of linear systems with ARMA models (具有ARMA模型的线性系统识别), 470-473
identification of nonlinear systems (非线性系统识别), 484-491
independent-component analysis ,blind separation of unknown source signals (独立成分分析, 未知源信号的盲分离), 500-519
linear system representation (线性系统表示), 468-469
nonlinear control (非线性控制), 492-499
nonlinear system representation (非线性系统表示), 477-484
other case studies (其他情形研究), 529-539
parametric system identification of linear systems using PLSNET (使用PLSNET的线性系统的参数系统识别), 473-477
spectrum estimation of sinusoids in additive noise (加性噪声的正弦谱估计), 519-528
Controllability matrix (可控性矩阵), 569
Controllable canonical form (可控性正则形式), 571
Convergence, speed of (收敛, 速度), 112-113
Convergence phase (收敛阶段), 115,169
Correction, absolute (修正, 绝对的), 55
Correlation (相关)
learning rule (学习规则), 76-77
matrix memory (矩阵存储), 100-104
Correlation coefficient (相关系数), 623-624
Correlation matrix (相关矩阵), 624
Correlational mechanisms (相关机制), 71
Counterpropagation networks (对传网络), 136-140
full (全), 138-140
training (训练), 138
Covariance matrix (协方差矩阵), 624
Cross-validation (交叉确认), 434
Crosstalk (串音), 101

D

Damping parameter (阻尼参数), 278
Data compression (数据压缩), 396, 399
Data matrix (数据矩阵), 343
Data preprocessing (数据预处理), 79-85, 536-539
Fourier transform (傅里叶变换), 81-83
partial least-squares regression (部分最小二乘回归), 84
principal component analysis (主成分分析), 83-84
scaling (规模), 80-81
transformations (变换), 81
wavelets and wavelet transforms (小波及小波变换), 84-85
Dead zone (死区), 57
Decomposition; *See also* Eigenvalue decomposition ;LU decomposition (分解, 参看特征值分解; LU分解)
generalized singular-value (广义奇异值), 367
orthogonal ,linear regression with (正交, 线性回归), 482-483
Schur, 311-313
singular-value (奇异值), 320-325, 574-577
Deconvolution ,homomorphic (解卷积, 同态), 83
Decorrelating algorithms (去相关算法), 400, 411
Definiteness. *See* Matrix definiteness (定性的, 参看矩阵定性)
Degeneracy (退化), 562
Delta rule (Delta规则), 35
Derivatives ,continuous (导数, 连续的), 59
Descent. *See* Steepest descent (下降, 参看最速下降)
Desired probabilities (期望概率), 220
Difference equations (差分方程组), 34

Differential of scalar functions ,with respect to a matrix (标量函数关于矩阵的微分), 595-596
Differentiation of scalar functions ,with respect to a vector (标量函数关于向量的微分), 594-595
Direct control (直接控制), 493-494
Discrete Picard condition(DPC) (离散皮卡条件(DPC)), 365-366
Discrete-time methods, for solving linear algebraic equations, iterative (离散时间方法, 用于求解线性代数方程组, 迭代), 370-375
Distributed information, in neural networks (分布式信息, 在神经网络内), 5
Distributed time-lagged feedforward neural networks(DTLFNN) (分布式时滞前馈神经网络), 222,228-231
Disturbance. *See* Minimal-disturbance principle (扰动, 看最小扰动原理)
Divide-and-conquer characteristics (分治性质), 210
Dominated membership function (优势成员函数), 629
Double-angle formulas (倍角公式), 630
DPC. *See* Discrete Picard condition (DPC。参看离散皮卡条件)
DTLFNN. *See* Distributed time-lagged feedforward neural networks (DTLFNN, 参看分布式时滞前馈神经网络)
Duality gap (对偶间隔), 247
Dynamical systems, state-space description of(动态系统, 状态空间描述), 566-571

E

Eigenvalue decomposition(EVD) (特征值分解), 313-314
Eigenvalue problem (特征值问题, 最大/最小), min/max, 317-320,379
Eigenvalues (特征值), 561-564
  generalized (广义的), 562
  nondistinct (不相异的), 562
Eigenvectors (特征向量), 561-564
  first principal (第一原则), 402-403
Elman network (Elman网络), 222-226
Emergent properties ,collective (涌现性质, 集体的), 199-200
Energy function (能量函数)
  instantaneous (瞬时的), 77
  negative definite (负定), 349
Ensemble averaging (总体平均), 618
Equations (方程组),
  algebraic (代数的),
    linear (线性的), 342-394
    Lyapunov, 326-328
    Riccati, 329-334
  difference (差分), 34
Equilibrium states ,spurious (平衡状态, 伪的), 203
Error correction approach, for correlation matrix memories (误差修正方法, 相关矩阵存储器), 104-106
Error correction rules ,linear (误差修正规则, 线性的), 50-51
Error surface, instantaneous (瞬时, 误差曲面), 39
Error vector (误差向量), 430
Errors, in mapping (误差, 在映射内), 297, 299
Estimation; *See also* Frequency estimation; Spectrum estimation (估计谱估计, 参看频率估计)
  of the first principal component, normalized Hebbian learning rule of Oja (第一主分量, Oja的正规化Hebb学习规则), 400-404
  of glucose concentrations from synthetic NIR data (人工NIR数据的葡萄糖浓度), 529-534
  neural network adaptive ,of principal components (神经网络自适应性, 主成分), 400-426
  neurocomputing applications for (神经计算应用), 468-549
  of several principal components adaptive principal component extraction algorithm (多个主成分提取算法), 411-418
    generalized Hebbian algorithm (广义Hebb算法), 407-410
    stochastic gradient ascent algorithm (随机梯度上升算法), 410-411
    symmetric subspace learning rule (对称子空间学习规则), 404-407
Euclidean norm (欧几里得范数), 572
Euler's formulas (欧拉公式), 631
EVD. *See* Eigenvalue decomposition (EVD, 参看特征值分解)
Event classification using infrasonic data (利用次声数据的事件分类), 534-539
Exactly determined systems (精确确定的系统), 344-347
Excitatory input (兴奋输入), 9, 70, 412
Expectation (期望), 617-620, 618
Extremum ,necessary and sufficient conditions for (极值, 必要充分条件), 601-602

F

Factor analysis (因子分析), 426, 430, 432, 435
  abstract (抽象), 435
Factorization (因子分解)

QR, 305-310
spectral (谱的), 313-314, 625-628
Fail-soft devices (软失败设备), 199
Fast fixed-point algorithm(FFPA) for ICA (快速固定点算法 (FFPA), 用于ICA), 512-519
"Fast" learning ("快速" 学习), 55, 183
Feature representation field (特征表示域), 182
Feedback (反馈), 104, 198
in Hopfield networks (霍普菲尔德网络), 34
Feedforward connection weights (前馈连接权值), 412-413
Feedforward multilayer perceptron (前馈多层感知器), 62-64
conjugate gradient backpropagation algorithms for (共轭梯度反向传播算法), 120-126
Feedforward networks, temporal (前馈网络, 时间的), 221-231
FFPA.*See* Fast fixed-point algorithm (FFPA, 参看快速固定点算法)
Fields (域), 550-552
Filters (滤波器), 365
finite impulse response (有限冲击响应), 37,222,228
linear transversal (线性横波的), 37
Finite impulse response (FIR) filters (有限冲击响应 (FIR) 滤波器), 37, 222, 228
FIR. *See* Finite impulse response filters (FIR, 参看有限冲击响应滤波器)
Fletcher-Reeves conjugate gradient algorithm (Fletcher-Reeves共轭梯度算法), 352,609
Forgetting factor (遗忘因子), 124,402
Fourier transform (傅里叶变换), 81-83
Frequency estimation, PLSR solution to (频率估计, PLSR解决方案), 524-528
Frobenius matrix (弗罗贝尼乌斯矩阵), 571
Full degeneracy (完全退化), 562
Function-sum and function-difference formulas (函数和与函数差公式), 631
Functions, 589-592; *See also* Radial basic function neural networks (函数, 参看径向基函数神经网络)
basic activation (基本激活), 27-33
binary sigmoid (二值S形), 30-31
bipolar sigmoid (双极S形), 31-33
hyperbolic tangent sigmoid (双曲正切S形), 31-33
partition (剖分), 211
power spectral density (功率谱密度), 624-625
probability density (概率密度), 617
probability distribution (概率分布), 616-617
scalar (标量)
with respect to a matrix (关于矩阵), 595-596
with respect to a vector (关于向量), 594-595
scalar kernel (标量核), 168
sets and (集合), 589-592
Fuzzy ART and fuzzy ARTMAP (模糊ART和模糊ARTMAP), 188-190
Fuzzy LAPART (模糊LAPART), 190
Fuzzy set theory (模糊集合论), 628-629

G

Gain matrix (增益矩阵), 570
Gauss-Seidel iterative method (高斯-赛德尔迭代方法), 371
Gaussian classifier, maximum-likelihood (高斯分类器, 最大似然), 56
General linear distributed associative memories (一般线性分布式联想记忆), 98-99
Generalization (泛化)
capability for (能力), 41, 64-5
process of (过程, 步骤, 方法), 45
Generalized eigenvalues (广义特征值), 562
Generalized Hebbian algorithm (广义Hebb算法), 407-410
Generalized LMS learning rule (广义LMS学习规则), 64-69
Generalized perceptron learning rule (广义感知器学习规则), 78-79
Generalized robust approach, for solving systems of linear equations corrupted with noise (广义鲁棒方法, 用于求解噪声损害的线性方程组), 354-361
Generalized singular-value decomposition (GSVD) (广义奇异值分解), 367
Global search algorithm, annealing-based (全局搜索算法, 基于退火), 213
Gradient. *See* Conjugate gradient method (梯度, 参看共轭梯度方法)
Gradient ascent algorithm, stochastic (梯度上升算法, 随机的), 410-411
Gram-Schmidt orthogonalization (格拉姆-施密特正交化), 101, 147-148

H

Hadamard matrices (阿达马矩阵), 588
Half-angle formulas (半角公式), 631
Hamming error (汉明误差), 55
Hankel matrices (汉克尔矩阵), 586-587
Hard limiter function (硬限幅函数), 27
symmetric (对称), 27
Hebbian algorithm, generalized (Hebb算法, 广义的),

407-410
Hebbian co-occurrence, standard (Hebb共生，标准)，402
Hebbian learning (Hebb学习)，69-73
Hebbian synapses (Hebb突触)，69-71
Hessenberg form (海森伯格形式)，586
Hessian matrix (黑塞矩阵)，596
Heteroassociative memory (异联想记忆)，98
Hidden layer (隐藏层)，62
Hidden neurons (隐藏神经元)，215
Hilbert matrices (希尔伯特矩阵)，586
Historical notes (历史注释)
McCulloch-Pitts neuron (McCulloch-Pitts神经元)，9-13
on neurocomputing (神经计算)，6-13
Holder inequality (赫尔德不等式)，572
Homomorphic deconvolution (同态解卷积)，83
Hopfield associative memory (霍普菲尔德联想记忆)，199-209
Hopfield model, of the artificial neuron (霍普菲尔德模型，人工神经元)，33-35
Hopfield networks (霍普菲尔德网络)，19
feedback in (反馈)，34
Hyperbolic tangent sigmoid function (双曲正切S形函数)，31-33

I

ICA, fast fixed-point algorithm for (ICA，快速固定点算法)，512-519
Identification (识别)
of linear systems, with ARMA models (线性系统，具有ARMA模型)，470-473
neurocomputing applications for (神经计算应用)，468-549
of nonlinear dynamic systems (非线性动态系统)，484-499
autoregressive moving average models (自动回归滑动平均模型)，469-470
independent-component analysis (独立成分分析)，500-519
linear system representation (线性系统表示)，468-469
nonlinear control (非线性控制) 492-499
nonlinear system representation (非线性系统表示法)，477-484
other case studies (其他情况研究)，529-539
parametric system identification of linear systems using PLSNET (利用PLSNET的线性系统的参数系统识别)，473-477
spectrum estimation of sinusoids in additive noise (在加性噪声中的正弦谱估计)，519-528
Identity function; 27; *See also* Trigonometric identities (恒等函数，也可参考三角恒等式)
Identity mapping (恒等映射)，297
Ill-posed problems with ill-determined numerical rank, regularization methods for (具有病态确定数值秩的不适定问题，正则化方法)，361-370
Impulse response matrix (冲击响应矩阵)，568
Independent-component analysis (独立分量分析)，500-501
blind separation of unknown source signals (未知源信号的盲分离)，500-519
fast fixed-point algorithm for ICA (用于ICA的快速固定点算法)，512-519
using neural networks (利用神经网络)，501-512
Independent test data (独立测试数据)，431, 433
Independent validation (独立检验)，112, 434
Indirect control (间接控制)，494-499
Induced norms of matrices (矩阵的导出范数)，573
Inequality constraints (不等约束)，263-265
Inertia matrices (惯性矩阵)，588
Information, distributed in neural networks (信息，分布在神经网络中)，5
Inhibitory input (抑制输入)，9
Inhibitory synapses (抑制突触)，70
nonzero (非零)，9
Initialization, of synaptic weights (初始化，突触权值)，110-111,203
Inner products (内积)，555-556
Input layer (输入层)，53
Input space (输入空间)，477
Input-state-output representation, nonlinear (输入状态输出表示，非线性)，478-479
Inputs (输入)
excitatory (兴奋的)，9
inhibitory (抑制的)，9
nonlinearly transformed (非线性变换)，49-50
Instantaneous energy function (瞬时能量函数)，77
Instantaneous error surface (瞬间误差曲面)，39,59
Intensities, matrix of (强度，矩阵)，431
Interactive mechanisms (交互机制)，71
Internal potential (内部位势)，75
Inverse Laplace transform (拉普拉斯逆变换)，567
Inverse of a matrix (矩阵的逆)，293-300,557-560
Inverse of partitioned matrices (分区矩阵的逆)，579-580
Inversion lemma, matrix (求逆引理，矩阵)，126
Iterative methods (迭代方法)，370-375

J

Jacobi iterative method (雅可比迭代方法)，370-371

Jacobian matrix（雅可比矩阵），134,597
Joint probabilities（联合概率），614
Jordan canonical form（若尔当标准形），565-566

K

Kahan matrices（Kahan矩阵），589
Kernel function, scalar（核函数，标量），168
Key input pattern（关键输入模式），98
Kohonen self-organizing map（Kohonen自组织映射），166-172
Kronecker delta（克罗内克Δ），396
Kronecker product and sum（克罗内克积与和），581-583
Kuhn-Tucker conditions（库恩-塔克条件），610-611
Kurtosis（峭度），504
  negative *vs.* positive（负的与正的），504-505

L

$L_1$-norm (least-absolute-deviations)neural network for solving linear equations（求解线性方程组的$L_1$范数（最小绝对偏差）神经网络），381-384
$L_\infty$-norm (minimax)neural network for solving linear equations（求解线性方程组的$L_\infty$范数（最小最大）神经网络），379-381
Lagrange multiplier methods（拉格朗日乘子法），277-281，611-613
  augmented（增广），281-286
  convergence of（收敛），284
LAPART. *See* Laterally primed adaptive resonance theory（LAPART，参看侧向基本自适应共振理论）
Laplace transform（拉普拉斯转换），567
Latent variable vector（隐式变量向量），437
Latent variables（隐式变量），431
Lateral connection weights（侧向连接权值），412
Laterally primed adaptive resonance theory（LAPART）（侧向基本自适应共振理论），189-190
Layers. *See* Multilayer perceptron; individual layers（层，参看多层感知器，各层）
Learning; *See also* Training（学习；也可参看训练）
  by example（例子），4
  "fast" *vs.* "slow"（"快" 相对于 "慢"），55,183
  Hebbian，69-73
  pattern *vs.* batch（单模式相对于集中式），488
  supervised（监督），106,174
  unsupervised（无监督），165
Learning algorithms; *See also* Backpropagation learning algorithms（学习算法；也可参看反向传播学习算法）
  for Boltzmann machine（玻尔兹曼机），220-221
  estimation of several principal components（多个主分量的估计），404-418
    adaptive principal component extraction algorithm（自适应主成分提取算法），411-418
    generalized Hebbian algorithm（广义Hebbian算法），407-410
    stochastic gradient ascent algorithm（随机梯度上升算法），410-411
    symmetric subspace learning rule（对称子空间学习规则），404-407
  estimation of the first principal component, normalized Hebbian learning rule of Oja（第一主成分估计，正规化的Hebb的Oja学习规则），400-404
  Kohonen self-organizing map（Kohonen自组织映射），166-172
  for neural network adaptive estimation of principal components（用于神经网络的主成分自适应估计），400-426
  nonlinear principal component analysis and robust PCA（非线性主成分分析和鲁棒PCA），418-426
Learning process, postulating from a neurobiological viewpoint（学习过程，从神经生理观点的假定），6
Learning rate parameter（学习率参数），40,42
Learning rules; *See also* Oja's learning rule（学习规则；也可参看Oja学习规则）
  backpropagation（反向传播），36
  conjugate gradient（共轭梯度），351-354
  correlation of（相关性），76-77
  generalized（广义）
  LMS（最小均方），64-69
  perceptron（感知器），78-79
  Hebbian，69-73
  least mean-square（LMS）（最小均方），36-44
  Oja's，73-75
  perceptron 感知器
  generalized（广义的），78-79
  standard（标准），77-78
  PLSNET-C，442-446
  potential（潜在，潜能），75-76
  for a single neuron（单个神经元），64-79
  symmetric subspace（对称子空间），404-407
  Widrow-Hoff，35-37
Least mean-square（LMS）learning rule（最小均方学习规则）
  algorithm for（算法），36-44
  generalized（广义的），64-69
Least-squares correction（最小二乘修正），377
Least-squares solution of systems of linear equations（线性方程组的最小二乘解），345-346

neurocomputing approach for (神经计算方法), 346-351
Lemma, matrix inversion (引理, 矩阵逆), 126
Levenberg-Marquardt algorithm (Levenberg-Marquardt算法), 132-136
Limited-step Newton formula (有限步牛顿公式), 606
Limiter function, hard (限制函数), 27
Linear algebra (线性代数), 550-589
eigenvalues and eigenvectors (特征值和特征向量), 561-564
fields and vector spaces (域和向量空间), 550-553
inner and outer products (内积和外积), 555-556
inverse and pseudoinverse of a matrix (矩阵的逆和伪逆), 557-560
Jordan canonical form (若当标准形), 565-566
Kronecker product and sum (克罗内克积与和), 581-583
linear independence of vectors (向量的线性独立), 556
matrix condition number (矩阵条件数), 577-578
matrix definiteness (矩阵的定性), 557
matrix representations and operations (矩阵表示和运算), 553-555
orthogonal and unitary matrices, and conjugate vectors (正交和酉矩阵, 共轭向量), 560-561
partitioned matrix operations (分区矩阵运算), 578-581
patterned and special matrices (模式和特殊矩阵), 585-589
rank of a matrix and linear independence (矩阵的秩和线性独立), 557
similarity transformations (相似变换), 564-565
singular-value decomposition (奇异值分解), 574-577
state-space description of dynamical systems (动态系统的状态空间描述), 566-571
summary of important properties for real and complex square matrices (关于实的和复的方阵的重要特征概述), 583-584
vector and matrix norms (向量和矩阵范数), 571-574
Linear algebraic equations (线性代数方程组), 342-394
conjugate gradient learning rule for solving systems of (求解系统的共轭梯度学习规则), 351-354
corrupted with noise, generalized robust approach for solving systems of (噪声污染, 求解系统的广义鲁棒处理方法), 354-361
$L_1$-norm(least-absolute -deviations)neural network for solving (利用$L_1$范数 (最小绝对偏差) 神经网络求解), 381-384
$L\infty$-norm(minimax)neural network for solving (利用$L_\infty$范数 (最小最大) 神经网络求解), 379-381
least-squares neurocomputing approach for solving systems of(求解系统的最小二乘神经计算方法), 346-351
least-squares solution of systems of (系统的最小二乘解), 345-346
matrix splittings for iterative discrete-time methods for solving (基于矩阵分裂的离散时间迭代方法求解), 370-375
regularization methods for ill-posed problems with ill-determined numerical rank (具有病态确定数值秩的病态问题的正则化方法), 361-370
solving with neural networks (用神经网络求解), 342-394
systems of simultaneous (并行系统), 343-344
total least-squares problem (总最小二乘问题), 375-378
Linear combiner (线性组合器), 45
simple adaptive ,and the LMS algorithm (简单自适应, LMS算法), 36-44
Linear distributed associative memories ,general (线性分布式联想记忆, 一般的), 98-99
Linear error (线性误差), 45
correction rules (修正规则), 50-51
Linear function (线性函数), 27
Linear independence of vectors (向量的线性独立), 556
rank of a matrix and (矩阵的秩), 557
Linear programming(LP)problems, neural networks for (线性规划问题, 神经网络), 244-256
the nonstandard form (非标准形式), 250-256
the standard form (标准形式), 247-250
Linear regression, with orthogonal decomposition (线性回归, 用正交分解), 482-483
Linear separability (线性可分性), 46-48
Linear systems (线性系统)
driven by white noise and spectral factorization (由白噪声和谱因子分解驱动的), 625-628
identification with ARMA models (具有ARMA模型的识别), 470-473
parametric system identification using PLSNET (利用PLSNET进行参数系统识别), 473-477
representation of (表示), 468-469
Linear transversal filters (线性横向滤波器), 37
Lipschitz continuous (利普希茨连续), 591
Loading vectors (装载向量), 431, 437-439
Local mechanisms (局部机制), 70-71
Local minima of networks (网络的局部极小), 209, 258
Local-minimum property (局部极小特征), 283-284
Long-term memory (长期记忆), 98

LP problem. *See* Linear programming problems (LP问题，参看线性规划问题)
LU decomposition (LU分解)，301-305
LVQ,self-organizing map and (LVQ，自组织映射)，180-182
LVQ1，MATLAB function for (LVQ1，MATLAB函数)，179
Lyapunov equation, neurocomputing approach for solving the algebraic (李雅普诺夫方程，求解代数的神经计算方法)，326-328，348
Lyapunov's direct method (李雅普诺夫直接方法)，598-600

M

Madaline (多重自适应线性单元)，7，52-55
Magno ganglion cells (Magno中心细胞)，14
Mahalanobis distortion (马哈拉诺比斯失真)，173
Mapping error (映射误差)，297，299
Mapping networks, 96-152; *See also* Kohonen self-organizing map (映射网络；参看Kohonen自组织映射)
  associative memory networks (联想记忆网络)，97-106
  backpropagation learning algorithms (反向传播学习算法)，106-119
  accelerated (加速)，120-136
  counterpropagation (对传) 136-140
  radial basic function neural networks (径向基函数神经网络)，140-152
Marginal probabilities (边缘概率)，614
Marquardt. *See* Levenberg-Marquardt algorithm (Marquardt，参看Levenberg-Marquardt算法)
Mathematical foundation for neurocomputing (神经计算的数学基础)，550-632
  constrained nonlinear programming (约束非线性规划)，610-613
  fuzzy set theory (模糊集合论)，628-629
  linear algebra (线性代数)，550-589
  Lyapunov's direct method (李雅普诺夫直接方法)，598-600
  principles of multivariable analysis (多变量分析原理)，589-598
    random variables and stochastic processes，613-628 (随机变量和随机过程)
  selected trigonometric identities (部分三角恒等式)，629-631
  unconstrained optimization methods (非约束优化方法)，601-610
Matrices (矩阵)，553-554
  adding and subtracting (加和减)，554
  data (数据)，343
  diagonal (对角)，555
  differential of scalar functions with respect to (关于标量函数的微分)，595-596
  Hessian，596
  identity (单位阵)，555
  induced norms of (导出范数)，573
  of intensities (强度)，431
  inverse and pseudoinverse of (逆和伪逆)，293-300，557-560
  inverse of partitioned (分区逆)，579-580
  Jacobian，134，597
  mixing (混合)，501
  multiplying (乘)，554
  orthogonal, and conjugate vectors (正交，共轭向量)，560-561
  patterned (组成图案的)，585-589
  power spectral density (功率谱密度)，624-625
  rank of (秩)，557
  real and complex square, summary of important properties for (实的和复的方阵，重要性质的概要)，583-584
  special (特殊的，专门的)，585-589
  symmetric (对称)，555
  transposing (转置)，555
  weighting (加权)，66
Matrix algebra (矩阵代数)，292-341
  decomposition (分解)
    eigenvalue (特征值)，313-314
    LU，301-305
    Schur，311-313
    singular-value (奇异值)，320-325
  factorization (因子分解)
    QR，305-310
    spectral (谱)，313-314
  Lyapunov equation, neurocomputing approach for solving (李雅普诺夫方程，神经计算方法用于求解)，326-328
  pseudoinverse of a matrix (矩阵的伪逆)，293-300
  Riccati equation , neurocomputing approach for solving (里卡蒂方程，神经计算用于求解)，329-334
  symmetric eigenvalue problem (对称特征值问题)，313-314
    neural network approach for (神经网络方法)，315-320
Matrix calculus (矩阵微积分) 594-596
  differential of scalar functions with respect to a matrix (标量函数对矩阵求微分)，595-596

differentiation of scalar functions with respect to a vector（标量函数对向量求微分），594-595
Matrix condition number（矩阵条件数），577-578
Matrix definiteness（矩阵的定性），557
Matrix determinant（矩阵行列式），579
Matrix exponential function（矩阵指数函数），567
Matrix inversion（矩阵的逆），126，294-29
Matrix memory, correlation of（矩阵记忆，相关），100-104
Matrix norms（矩阵范数），571-574
Matrix operations, partitioned（矩阵运算，分块），578-581
Matrix representations and operations（矩阵表示和运算），553-555
partitioned（分块），578-581
Matrix splittings , for iterative discrete-time methods for solving linear equations（矩阵分裂，求解线性方程组的离散时间迭代方法），370-375
Maximum-likelihood Gaussian classifier（最大似然高斯分类器），56
Mays's perceptron learning rules（Mays感知器学习规则），57-58
McCulloch-Pitts neuron（McCulloch-Pitts神经元），9-13，200
a two-state device（两个状态的装置），9
Mean centering（平均中心），80
Mean-field approximation（平均场近似），221
Mean square error(MSE)（均方误差），120
Mean values（均值），617-620
Measurement data（测量数据），395
Memorized patterns（存储模式），96-97，100
reconstructing optimally（最优重构），105
Memory（记忆，存储）
associative（联想），98，199
autoassociative（自联想），98
content-addressable（内容可寻址），34，199
heteroassociative（异联想），98
long-term *vs.* short-term（长期与短期），98
Memory networks（记忆网络）
associative（联想的），97-106
correlation matrix（相关矩阵），100-104
error correction approach for correlation matrix（相关矩阵的误差修正方法），104-106
general linear distributed（一般的线性分布），98-99
capability of（能力，容量），97
MIMO. *See* Multiple-input, multiple-output systems（MIMO，参看多输入多输出系统）
Min/max eigenvalue problem（最小/最大特征值问题），317-320, 379
Minima of networks, local（网络的最小值，局部），209
Minimal-disturbance principle（最小扰动原理），51. 55
Minimal realization system parameters（最小实现系统参数），476
Mixed constraints（混合约束），254-256, 265-268
Mixing matrix（混合矩阵），501
MLP. *See* Multilayer perceptron（MLP，参看多层感知器）
Models（模型）
of artificial neurons（人工神经网络），25-27, 33-35
autoregressive moving average（自回归滑动平均），469-470
Hopfield，33-35
linear system identification with ARMA（具有ARMA的线性系统识别），470-473
Modified Newton methods（改进的牛顿法），606-607
Modified relaxation algorithm（修改的松弛算法），58
Moments（矩），617-620
Momentum（动量）
adding parameter for（增加参数），66
backpropagation learning algorithm with updating（具有更新的反向传播学习算法），113-114
Moore-Penrose inverse（Moore-Penrose逆），362
Moving average models, autoregressive（滑动平均模型，自回归），469-470
MSE. *See* Mean square error（MSE，参看均方差误差）
Multilayer perceptron(MLP)（多层感知器）
feedforward（前向反馈），62-64
trained by BP（由BP训练），532-533
Multiple Adaline (Madaline)（多重自适应线性单元），7，52-55
Multiple-input,multiple-output(MIMO) systems（多输入，多数出(MIMO)系统），484
Multiplier methods, Lagrange（乘子法，拉格朗日），611-613
Multivariable analysis（多变量分析）
chain rule（链式规则），593-594
Hessian matrix（黑塞矩阵），596
Jacobian matrix（雅可比矩阵），597
matrix calculus（矩阵微积分学）594-596
differential of scalar functions with respect to a matrix（标量函数关于矩阵的微分），595-596
differentiation of scalar functions with respect to a vector（标量函数关于向量的微分），594-595
principles of（原理，原则）589-598
quadratic forms（二次型），592-593
sets and functions（集合和函数），589-592
Taylor series expansion（泰勒级数展开），597-598

N

NARMA. *See* Nonlinear autoregressive moving average（NARMA，参看非线性自回归滑动平均）

Negative definite energy function（负定能量函数），349
Negative kurtosis（负峭度），504-505
Network configuration（网络配置），111-112
Networks; *See also* Neural networks（网络；也可参看神经网络）
ability to generalize（泛化能力），111-112
associative memory（联想记忆），97-106
correlation matrix（相关矩阵），100-104
error correction approach for correlation matrix（相关矩阵的误差修正方法），104-106
general linear distributed（一般线性分布），98-99
counterpropagation（对传），136-140
Elman，222
feedforward, temporal（前馈，时间的），221-231
local minima of（局部最小），209
mapping（映射），96-152
recurrent（递归），198-221
simple（简单的），222-226
self-organizing（自组织）165-197
adaptive resonance theory neural networks（自适应共振理论神经网络），182-190
Kohonen self-organizing map（Kohonen自组织映射），166-172
learning vector quantization（学习向量量化），173-182
spatiotemporal sensitivity of static（状态的时空灵敏性），222
Neural network approach（神经网络方法）
for partial least-squares regression（部分最小二乘回归），442-450
rebust（鲁棒），450-455
for the symmetric eigenvalue problem（对称特征值问题），315-320
Neural network architectures, examples of（神经网络体系结构，例子），18
Neural networks（神经网络）
adaptive estimation of principal components, learning algorithms for（主成分学习算法的自适应估计），400-426
adaptive resonance theory（自适应共振理论），182-190
artificial（人工的），3-5
for the augmented Lagrange multiplier method（增广拉格朗日乘子法），281-286
for barrier function NP methods（障碍函数的NP方法），275-276
classification of（分类），18-19
distributed information in（分布信息），5
for linear programming problems（线性规划问题），244-256
the nonstandard form（非标准形式），250-256
the standard form（标准形式），247-250
for nonlinear continuous constrained optimization problems（非线性连续约束优化问题），268-286
for NP methods（NP方法）
ordinary Lagrange multiplier（普通拉格朗日乘子），276-281
penalty function（惩罚函数），269-275
for optimization problems（优化问题），243-291
for quadratic programming problems（二次规划问题），257-268
radial basic function（径向基函数），140-152
recurrent（递归），19,198-221
Boltzmann machine（玻尔兹曼机），215-221
Hopfield associative memory（霍普菲尔德联想记忆），199-209
overview of（概览），198-199
simulated annealing（模拟退火），209-215
robust nature of（鲁棒性），5
solving linear algebraic equations with（求解线性代数方程组），342-394
solving matrix algebra problems with（求解矩阵代数问题），292-341
structured（结构），292-293
time-delay（时间延迟），226-228
Neurobiological viewpoint; *See also* Neuroscience（神经生物学的观点；也可参看神经科学）
postulating a learning process from（假定一个学习步骤），6
Neurocomputing（神经计算），3-23
defined（定义），3-6
fundamental concepts of（基本概念），24-95
historical notes on（历史注释），6-13
mathematical foundation for（数学基础），550-632
and neuroscience（神经科学），13-18
Neurocomputing applications（神经计算应用），243-632
for identification , control, and estimation（识别，控制和估计），468-549
for optimization problems（优化问题），243-291
for solving linear algebraic equations（求解线性代数方程组），342-394
for solving matrix algebra problems（解决矩阵代数问题），292-341
statistical methods（统计方法），395-467
Neurocomputing approach（神经计算方法）
for least-squares solution of systems of linear equations（用于线性问题组的最小二乘法），346-351
for solving the algebraic Lyapunov equation（求解代

数李雅普诺夫方程)，326-328
for solving the algebraic Riccati equation（求解代数里卡蒂方程)，329-334
Neurocomputing concepts（神经计算概念)，24-95
Adaline and Madaline，35-55
basic activation functions（基本激活函数)，27-33
basic models of artificial neurons（人工神经元的基本模型)，25-27
data preprocessing（数据预处理)，79-85
feedforward multilayer perceptron（前馈多层感知器)，62-64
Hopfield model of the artificial neuron（人工神经元的霍普菲尔德模型)，33-35
overview of basic learning rules for a single neuron（单个神经元的基本学习规则概况)，64-79
simple perceptron（简单感知器)，56-62
Neurons; *See also* Artificial neuron; single neuron（神经元；也可参看人工神经元；单个神经元)
hidden（隐藏)，215
McCulloch-Pitts，9-13
output of（输出)，56
stochastic（随机的)，215-221
sufficient numbers of（足够数目)，111
threshold of（阈值)，9
winning（获胜)，166,168
Neuroscience（神经科学)，13-18
biological neural networks（生物神经网络)，13-18
Newton's methods（牛顿法)，604-606
limited-step（有限步)，606
modified（改进，修正)，606-607
Newton's optimization algorithm（牛顿优化算法)，132
NLPCA. *See* Nonlinear principal component analysis（NLPCA，参看非线性主成分分析)
No feasible solutions（没有可行解)，246
Noise（噪声)，101
Non-Hebbian synaptic activity（非Hebbian突触活动)，71
Non-parametric models（非参数模型)，469
Nondistinct eigenvalues（非相异特征值)，562
Nonlinear ARMA（非线性ARMA)，479-484
Nonlinear autoregressive moving average(NARMA)（非线性自回归滑动平均)，222
Nonlinear computing ,in the brain（非线性计算，在人脑中)，4
Nonlinear control（非线性控制)，492-499
Nonlinear input-state-output representation（非线性输入状态输出表示)，62, 478-479
Nonlinear models , of artificial neurons（非线性模型，人工神经元)，25-26
Nonlinear principal component analysis(NLPCA), and robust PCA（非线性主成分分析，鲁棒PCA)，418-426
Nonlinear programming ,constrained(非线性规划,约束)，610-613
Kuhn-Tucker conditions（库恩-塔克条件)，610-611
Lagrange multiplier methods（拉格朗日乘子法)，611-613
Nonlinear system representation（非线性系统表示)，477-484
nonlinear ARMA（非线性ARMA)，479-484
nonlinear input-state-output representation（非线性输入状态输出表示)，478-479
Nonlinear systems, identification and control of dynamic（非线性系统，动态系统的识别和控制)，484-499
autoregressive moving average models（自回归滑动平均模型)，469-470
identification of linear systems with ARMA models（具有ARMA模型的线性系统识别)，470-473
identification of nonlinear systems（非线性系统的识别)，484-491
independent-component analysis, blind separation of unknown source signals（独立分量分析，未知信号源的盲分离)，500-519
linear system representation（线性系统表示)，468-469
nonlinear control（非线性控制)，492-499
nonlinear system representation（非线性系统表示)，477-484
other case studies（其他情况研究)，529-539
parametric system identification of linear systems using PLSNET（利用PLSNET的线性系统的参数系统识别)，473-477
spectrum estimation of sinusoids in additive noise（在加性噪声下的正弦谱估计)，519-528
Nonlinear weight correction rules（非线性权值修正规则)，52
Nonlinearly transformed inputs（非线性变换输入)，49-50
Nonstandard form of the linear programming problem, neural networks for（线性规划问题的非标准形式，神经网络)，250-256
Nonstationary processes（非稳态过程)，621-622
Nonunique solutions（不唯一解)，246
Nonzero inhibitory synapses（非零抑制突触)，9
Normal equations（正规方程组)，120
for the linear combiner（线性组合器)，121-122
solving（求解)，122-123
Normalization（正规化)，403

Normalized Hebbian learning rule of Oja (Oja的正规化的Hebb学习规则), 400-404
NP-complete (完全NP), 210
Numerical rank, ill-determined (数值秩, 病态确定的), 361-370

O

Observability matrix (可观察性矩阵), 569
Observable canonical form (可观察典范形), 571
Observation vector (观察向量), 343
Off-line training (离线训练) 414
Oja's learning rule (Oja学习规则), 73-75
  normalized Hebbian (正规化的Hebbian), 400-404
OLS. *See* Orthogonal least squares (OLS, 参看正交最小二乘法;)
Online method (在线方法), 107
Online processing (在线处理), 342
Operations, matrix (运算, 矩阵), 553-555
  partitioned (分区), 578-581
Optimal iteration parameters (最佳迭代参数), 373
Optimization algorithm, Newton's (最优化算法, 牛顿法), 132
Optimization methods (最优化方法)
  neurocomputing applications for (神经计算应用), 243-291
  unconstrained (无约束), 601-610
    conjugate gradient method (共轭梯度方法), 608-610
    modified Newton and quasi-Newton methods (改进的牛顿法和拟牛顿法), 606-607
    necessary and sufficient conditions for an extremum (极值的必要充分条件), 601-602
    Newton's methods (牛顿法), 604-606
    steepest descent (最速下降), 602-604
Ordering phase (排序阶段), 169
Orthogonal decomposition, linear regression with (正交分解, 线性回归), 482-483
Orthogonal least squares(OLS) (正交最小二乘法), 147-152
  algorithm for training an RBF network (训练RBF网络的算法), 150-151
  Gram-Schmidt orthogonalization (格拉姆-施密特正交化), 147-148
  regression (回归), 148-151
Orthogonal matrices, and conjugate vectors (正交矩阵, 共轭向量), 560-561
Orthogonal projector (正交投影), 377
Orthonormality (规范正交的), 573
Outer product rule (外积规则), 100
Outer products (外积), 100,555-556
Outliers (出格点), 354
Output layer (输出层), 62
Output space (输出空间), 477
Overdetermined equations (超定方程组), 343,431
Overfitting (过适应), 112,429
Overrelaxation iterative method, successive (超松弛迭代方法, 逐次), 371-373

P

Parallel computing (并行处理), 292
  in the brain (人脑), 4
Parallel-series configuration (并行-串行配置), 485
Parametric system identification (参数系统识别), 43, 469
  of linear systems, using PLSNET (线性系统, 利用PLSNET), 473-477
Partial least-squares regression(PLSR) (局部最小二乘回归), 84, 435-442
  dimension of (维数), 527-529
  neural network approach for (神经网络方法), 442-450
  solution to frequency estimation (频率估计解), 524-528
Partition function (剖分函数), 211
Partitioned matrices (分区矩阵)
  inverse of (逆), 579-580
  rank of (秩), 580-581
Partitioned matrix operations (分区矩阵运算), 578-581
Parvo ganglion cells (Parvo小细胞), 14
Pascal matrices (帕斯卡矩阵), 588
Pattern completion (模式完全), 217
Pattern learning (模式学习), 488
Patterned matrices (模式化矩阵), 585-589
Patterns (模式)
  memorized (记忆), 96-97
  recognition of (识别), 3
PCA. *See* Principal-component analysis (PCA, 参看主成分分析)
Penalty parameters (惩罚参数), 270-275
  choice of (选择, 精选), 284
  computational difficulties in using (在使用中的计算难度), 275
Perceptron; *See also* Simple perceptron (感知器, 参看简单感知器)
  feedforward multilayer (前馈多层), 62-64
  learning rules (学习规则)

generalized（广义的），78-79
Mays's，57-58
original concept of（原始概念），7
Phase variable canonical form（相位变量正则形式），571
Piecewise linear function（分段线性函数），29
Pipelined recurrent neural network(PRNN)（管状递归神经网络），222
Plasticity（可塑性）
coefficient of（系数），74,403
vs. stability（相对于稳定性），182
PLSNET，533-534
calibration of（校准），442-446
parametric system identification of linear systems using（线性系统的参数系统识别），473-477
PLSR. *See* partial least-squares regression（PLSR，参看部分最小二乘回归）
Positive kurtosis（正的峭度），504-505
Potential learning rule（潜在学习规则），75-76
Power formulas（幂公式），631
Power spectral density functions and matrices（功率谱密度函数和矩阵），624-625
Prediction algorithms（预测算法），439-442
Preprocessing 预处理
data（数据），79-85,536-539
transformation（变换），80
Prewhitening process（预漂白处理），502-503
Principal-component analysis(PCA)（主成分分析），83-84, 396-400
robust（鲁棒），418-426
Principal-component regression（主成分回归），425-434
PRNN, *See* Pipelined recurrent neural network（PRNN，参看管状递归神经网络）
Probabilistic correct response patterns（概率修正响应模式），217
Probabilities, desired（概率，期望），220
Probability density functions（概率密度函数），617
Probability distribution functions（概率分布函数），616-617
Problem statement（问题陈述），523-524
Product , Kronecker（积，克罗内克），581-583
Product formulas（乘积公式），630
Programmed computing（程序的计算），3
Programming .*See* Constrained nonlinear programming（规划，参看约束非线性规划）
Propagation. *See* Backpropagation; Counterpropagation（传播，参看反向传播；对传）
Pseudoinverse of a matrix（矩阵的伪逆），293-300,557-560
Pythagorean formulas（毕达哥拉斯公式），629

Q

QR factorization（QR因子分解），305-310
Quadratic forms（二次型），257-259,592-593
Quantization（量化），56
learning vector（学习向量），173-182
Quasi-Newton methods（准牛顿法），606-607
Quotient formulas（商公式），629

R

Radial basic function(RBF) neural networks（径向基函数神经网络），140-152
orthogonal least squares（正交最小二乘法），147-152
training（训练）
with fixed centers（具有固定中心），142-145
using the stochastic gradient approach（应用随机梯度方法），145-147
Random variables（随机变量），613-616,613-620
axiomatic approach（公理方法），615
expectation,mean values,and moments（期望、均值和矩），617-620
indepencence（独立性），615-616
probability density functions（概率密度函数），617
probability distribution functions（概率分布函数），616-617
relative-frequency approach（相关频率方法），613-615
Randomness（随机性），210
Range scaling（幅度范围），80
Rank of a matrix ,and linear independence（矩阵的秩，线性独立），557
Rank of partitioned matrices（分区矩阵的秩），580-581
RBF NN. *See* Radial basic function neural networks（RBF NN，参看径向基函数神经网络）
Real square matrices , summary of important properties for（实方阵，重要特性的概括），583-584
Real-time processing（实时处理），342
Real-time recurrent neural network(RTRNN)（实时递归神经网络），222
Recall phase（回溯阶段），202
Reciprocal formulas（倒数公式），629
Recurrent neural networks（递归神经网络），19,198-221
Boltzmann machine（玻尔兹曼机），215-221
Hopfield associative memory（霍普菲尔德联想记忆），199-209
overview of（概览），198-199
simple（简单的），222-226
simulated annealing（模拟退火），209-215
Recursive least-squares-based backpropagation(RWLS)

algorithm（基于最小二乘递归反向传播算法），66-67，126-129
derivation of（导数），127-129
matrix inversion lemma（矩阵逆引理），126
Woodbury's identity（伍德伯里恒等式），126
Reestimation algorithms（重估计算法），400
Regression（回归）
partial least-squares（部分最小二乘法），435-442
neural network approach for（神经网络方法），442-450
principal-component（主成分），425-434
Regularization methods（正则化方法）
for ill-posed problems with ill-determined numerical rank（具有不定数值秩的不适定问题），361-370
Tikhonov, 362
Representation（表示）
of linear systems（线性系统），468-469
matrix（矩阵），553-555
nonlinear system（非线性系统），62,477-484
Representation fields（表示域），182
Resolvent matrix（预解矩阵），568
Riccati equation, neurocomputing approach for solving the algebraic（里卡蒂方程，求解代数（里卡蒂方程）的神经计算方法），329-334
Richardson's iterative method（理查森迭代方法），373-375
Right singular vectors（右奇异向量），321
Robust PLSR ,neural network approach to（鲁棒PLSR，神经网络方法），450-455
RTRNN. *See* Real-time recurrent neural network（RTRNN，参看实时递归神经网络）
RWLS. *See* Recursive least-squares-based backpropagation algorithm（RWLS，参看基于最小二乘递归反向传播算法）

S

Saturating linear function（饱和线性函数），29
symmetric（对称的），29
Saturation（饱和），403
Scalar functions（标量函数）
kernel（核），168
preprocessing（预处理），80
with respect to a matrix（关于矩阵），595-596
with respect to a vector（关于向量），594-595
Scaling（尺度，规模），80-81
variance（变化），80
Schmidt. *See* Gram-Schmidt orthogonalization（Schmidt，参看格拉姆-施密特正交化）
Schur decomposition（舒尔分解），311-313
Schwartz. *See* Cauchy-Schwartz inequality（Schwartz，参看柯西-施瓦茨不等式）
Search phase（搜索阶段），115
Search-then-converge method（搜索然后收敛），41，115-116
Search time constant（搜索时间常数），41
Self-amplification（自放大），412
Self-organizing map(SOM)（自组织映射）
Kohonen，166-172
and LVQ，180-182
*Self-organizing networks*（自组织网络），165-197
adaptive resonance theory(ART) neural networks（自适应共振理论神经网络），182-190
Kohonen self-organizing map（Kohonen自组织映射），166-172
learning vector quantization（学习向量量化），173-182
Sensitivity（灵敏度），117
spatiotemporal（时空），222
SEP. *See* standard error of prediction（SEP，参看标准预测误差）
Separability , linear（可分性，线性的），46-48
Separation process（分离处理），503-505
Set theory（集合论）
and functions（函数），589-592
fuzzy（模糊），628-629
Shaping filter（修正滤波器），626
Short-term memory（短期记忆），98，221
Sigmoid functions（S形函数）
binary（二值的），30-31
bipolar（双极性），31-33
hyperbolic tangent（双曲正切），31-33
Signum function（符号函数），28，44，202
Similarity transformations（相似变换），564-565
Simple adaptive linear combiner ,and the LMS algorithm（简单自适应线性组合器，LMS算法），36-44
Simple degeneracy（简单退化），562
*Simple perceptron*（简单感知器），56-62
Mays's perceptron learning rules（Mays感知器学习规则），57-58
with a sigmoid activation function（具有S形激活函数），58-62
Simple recurrent network(SRN)（简单递归网络），222-226
Simulated annealing（模拟退火），209-215
Simultaneous linear algebraic equations, systems of（联立线性代数方程组，系统），343-344
Single-component case（单成分情形），429
Single-input, single-output(SISO) case（单输入单输出

(SISO) 情况), 468, 484
Single neuron, learning rules for (单个神经元, 学习规则), 64-79
Singular-value decomposition (奇异值分解), 320-325, 574-577
  generalized (广义), 367
Singular values (奇异值), 573
Sinusoids in additive noise, spectrum estimation of (加性噪声正弦, 谱估计), 519-528
SISO, *See* Single-input ,single-output case (SISO, 参看单输入单输出情况)
Slab (板), 62
"Slow" learning ("慢" 学习), 55, 183
SOM. *See* Self-organizing map(SOM) (SOM, 参看自组织映射)
SOR. *See* Successive overrelaxation iterative method (SOR, 参看逐次超松弛迭代方法)
Spatiotemporal sensitivity of static networks (静态网络的时空敏感性), 222
Special matrices (特殊矩阵), 585-589
Spectral factorization (谱因子分解)
  eigenvalue decomposition (特征值分解), 313-314
  linear systems driven by (驱动的线性系统), 625-628
Spectral residue (谱残留), 450
Spectrum estimation, of sinusoids in additive noise (谱估计, 加性噪声的正弦), 519-528
  PLSR solution to frequency estimation (频率估计的PLSR解), 524-528
  problem statement (问题陈述), 523-524
Splittings. *See* Matrix splittings (分裂, 参看矩阵分裂)
Spread parameter, setting (扩展参数, 设置), 144-145
Square matrices , real and complex ,summary of important properties for (方阵, 实的和复的, 重要性质小结), 583-584
SRN. *See* Simple recurrent network (SRN, 参看简单递归网络)
Stability, *vs.* plasticity (稳定性相对于可塑性), 182
Standard backpropagation (标准的反向传播)
  ability of network to generalize (网络泛化能力), 111-112
  independent validation (独立检验), 112
  initialization of synaptic weights (突触权值的初始化), 110-111
  network configuration (网络配置), 111-112
  practical issues in using (在使用中的实际问题), 110-113
  speed of convergence (收敛速度), 112-113
Standard error of prediction(SEP) (标准预测误差), 433
  minimum (最小值), 434
Standard form of the linear Programming problem , neural networks for (线性规划问题的标准形式 , 神经网络), 244, 247-250
Standard Hebbian co-occurrence (标准Hebbian共生), 402
Standard perceptron learning rule(标准感知器学习规则), 77-78
State-space description of dynamical systems (动态系统的状态空间描述), 566-571
State transition matrix (状态转移矩阵), 567
Static networks, spatiotemporal sensitivity of (静态网络, 时空敏感性), 222
Stationary processes (稳定过程), 621-622
Statistical learning (统计学习), 55
Statistical methods using neural networks (利用神经网络的统计方法), 395-467
  learning algorithms for neural network adaptive estimation of principal components (神经网络对主成分的自适应估计学习算法), 400-426
  neural network approach for partial least-squares regression (部分最小二乘回归的神经网络方法), 442-450
  partial least-squares regression (部分最小二乘回归), 435-442
  principal-component analysis (主成分分析), 396-400
  principal-component regression (主成分回归), 425-434
  robust PLSR ,a neural network approach (鲁棒PLSR, 神经网络方法), 450-455
Steepest descent (最速下降), 36, 105, 602-604
Stochastic gradient ascent algorithm (随机梯度上升算法), 410-411
Stochastic gradient-based method for training an RBF NN (训练RBF NN的基于随机梯度方法), 146
Stochastic neurons (随机神经元), 215-221
Stochastic processes (随机过程), 620-624, 620-628
  ergodic and nonergodic (遍历和非遍历), 622
  linear systems driven by white noise and spectral factorization (由白噪声和谱因子分解驱动的线性系统), 625-628
  power spectral density functions and matrices (功率谱密度函数和矩阵), 624-625
  vector (向量), 624
Storage phase (存储阶段), 202
Stored patterns (存储模式), 96
Structured neural networks (结构神经网络), 292-293
Sub-Gaussian signals (次高斯信号), 504
Subspace learning rule, symmetric (子空间学习规则, 对称的), 404-407

Successive overrelaxation(SOR) iterative method（逐次超松弛迭代方法），371-373
Sum, Kronecker（和，克罗内克），581-583
Super-Gaussian signals（超高斯信号），504
Supervised learning（监督学习），106，174
Supremum（上确界），573
Symmetric eigenvalue problem（对称特征值问题），313-314
  neural network approach for（神经网络方法），315-320
Symmetric hard limiter function（对称的硬限制函数），27
Symmetric saturating linear function（对称饱和线性函数），29
Symmetric subspace learning rule（对称子空间学习规则），404-407
Synapses. *See* Hebb synapses（突触，参看Hebb突触）
Synaptic weights,initialization of（突触权值，初始化），110-111
Synchronous activation（同步激活），69
System dimensions,assumed（系统维数，假定），473
System identification（系统识别），469
System parameters,minimal realization（系统参数，最小实现），476
System representation; *See also* Nonlinear systems（系统表示；参看非线性系统
  nonlinear（非线性），477-484
Systems of linear equations（线性方程组），343-361
  algebraic , simultaneous（代数的，联立的），343-344
  conjugate gradient learning rule for solving（用于求解的共轭梯度学习规则），351-354
  corrupted with noise ,generalized robust approach for solving（由噪声损坏，用于求解的广义鲁棒方法），354-361
  least-squares solution of（最小二乘解），345-346
  neurocomputing approach for（神经计算方法），346-351

T

Target data（目标数据），395
Target values（目标值），429
Taylor series expansion（泰勒级数展开），597-598
TDNN. *See* Time-delay neural networks（TDNN，参看时延神经网络）
Temporal feedforward networks（时间前馈网络），221-231
  distributed time-lagged feedforward neural networks（分布式时滞前馈神经网络），228-231
  overview of（概述），221-222
  simple recurrent network（简单递归网络），222-226
  time-delay neural networks（时延神经网络），226-228
Threshold , of a neuron（阈值，一个神经元），9
Threshold function（阈值函数），27-28
Tikhonov regularization（Tikhonov正则化），362，368
Time autocorrelation function（时间自相关函数），623
Time-delay neural networks(TDNN)（时延神经网络），33，221-222，226-228
Time-dependent mechanisms（时间依赖机制），70
Time-frequency descriptions（时频描述），85
Time-lagged feedforward neural networks, distributed（时滞前馈神经网络，分布式的），228-231
TLS. *See* Total least-squares problem（TLS，参看总体最小二乘问题）
Toeplitz matrices（特普利茨矩阵），586
Topological ordering property（拓扑排序性质），166，170
Total least-squares(TLS) problem（总体最小二乘问题），375-378
Training（训练）
  algorithm for an RBF NN with fixed centers（用于具有固定中心的RBF NN的算法），145
  conjugate-gradient-based algorithm for training an MLP NN（训练MLP NN的基于共轭梯度的算法），124-125
  off-line（离线），414
  process of（过程），45
  the RBF NN
    with fixed centers（具有固定中心），142-145
    using the stochastic gradient approach（利用随机梯度方法），145-147
Training algorithm（训练算法），123-126
Training data（训练数据），395
Training measurements（训练度量），433
Transfer function matrix（转移函数矩阵），568
Transformation preprocessing（变换预处理），80
Transformations（变换），81
  affine（仿射），26
  similarity（相似性），564-565
Triangular systems（三角系统），301-302
Tridiagonal matrix（三对角矩阵），586
Trigonometric identities（三角恒等式），629-631
Truncation parameter（截断参数），363
Tucker. *See* Kuhn-Tucker conditions（塔克，参看库恩-塔克条件）
Two-state device, McCulloch-Pitts neuron as（双状态设计，McCulloch-Pitts神经元），9

U

Unbounded solutions（无界的解），246
Unconstrained optimization methods（无约束最优化方

法)，601-610
conjugate gradient method（共轭梯度方法)，608-610
modified Newton and quasi-Newton methods（改进的牛顿法及拟牛顿法)，606-607
necessary and sufficient conditions for an extremum（极值的充分必要条件)，601-602
Newton's methods（牛顿法)，604-606
steepest descent（最速下降)，602-604
Underdetermined equations（欠定方程组)，343，431
Uniform probability density function（均匀概率密度函数)，619-620
Unique solutions（唯一解)，246
Unit vector（单位向量)，572
Unitary matrices, and conjugate vectors（酉矩阵，共轭向量)，560-561
Unsupervised learning（无监督学习)，165
Update step, 40; *See also* Momentum updating（更新步，40；参看动量更新）

V

Validation; *See also* Cross-validation independent（确认，参看交叉确认独立性)，112，434
Vandermonde matrices（范德蒙德矩阵)，587-588
Variance-scaling（方差规整)，80，436
Vector-matrix form ,of the backpropagation algorithm（向量矩阵形式，反向传播算法)，117-119
Vector norms（向量范数)，571-574
Vector quantization ,learning（向量量化，学习)，173-182
Vector spaces（向量空间)，552-553
fields and（域)，550-553
Vectors（向量)，553-554
conjugate, orthogonal and unitary matrices and（共轭，正交矩阵和酉矩阵)，560-561
differentiation of scalar functions with respect to（关于标量函数微分)，594-595
error（误差)，430
latent variable（隐式变量)，437
linear independence of（线性无关)，556
loading（装载)，431，437-439
observation（观察，观测)，343
right singular（右奇异)，321
stochastic processes of（随机过程)，624
weight（权值)，202
weight loading（权值装载)，436-437
Vigilance parameter（警戒参量)，182
Voronoi quantizer（Voronoi量化器)，173

W

Wavelets and wavelet transforms（小波和小波变换)，84-85
Weight correction（修正权值）
for errors（误差)，78
rules for nonlinear（非线性规则)，52
Weight loading vector（权值装载向量)，202，436-437
Weighting function（加权函数)，450
Weighting matrix（加权矩阵)，66，450
White noise ,linear systems driven by（白噪声，被驱动的线性系统)，625-628
Wide-sense-stationary processes（宽平稳过程)，622
Widrow-Hoff learning rule（Widrow-Hoff学习规则)，35-37
Winning neuron（获胜神经元)，166，168
Woodbury's identity（伍德伯里恒等式)，126

Z

Zero-mean（零平均)，500