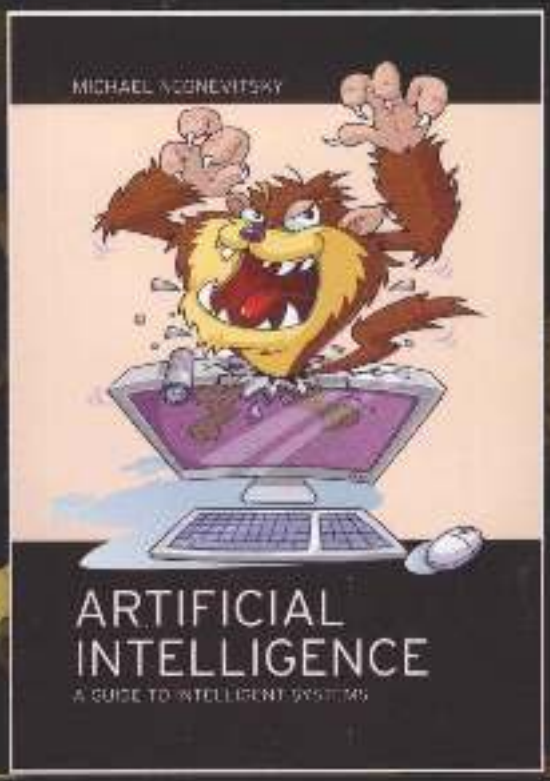
计 算 机 科 学 丛 书

原书第3版

人工智能
智能系统指南

（澳）Michael Negnevitsky 著 陈薇 等译

Artificial Intelligence
A Guide to Intelligent Systems Third Edition



机械工业出版社
China Machine Press

人工智能 智能系统指南（原书第3版）

Artificial Intelligence A Guide to Intelligent Systems Third Edition

人工智能经常被人们认为是计算机科学中一门高度复杂甚至令人生畏的学科。长期以来人工智能方面的书籍往往包含复杂的矩阵代数和微分方程。本书基于作者多年面向微积分知识甚微的学生授课时所用的讲义，假定读者没有编程经验，以简单易懂的方式介绍了智能系统的基础知识。

本书目前已经被国际上多所大学（例如，德国的马格德堡大学、日本的广岛大学、美国的波士顿大学和罗切斯特理工学院等）采纳为教材。

如果您正在寻找关于人工智能或智能系统设计课程的浅显易懂的入门级教材，如果您不是计算机科学领域的专业人员而又正在寻找介绍基于知识系统最新技术发展的自学指南，本书将是您的最佳选择。

本书的主要内容： 基于规则的专家系统、不确定性管理技术、模糊专家系统、基于框架的专家系统、人工神经网络、进化计算、混合智能系统、知识工程、数据挖掘。

本版的新增内容： 与上一版相比，本版进行了全面更新，以反映人工智能领域的最新进展。其中新增了“数据挖掘与知识发现”一章和“自组织神经网络”及聚类相关内容，同时补充了4个新的案例研究。

作者简介



Michael Negnevitsky 澳大利亚塔斯马尼亚大学电气工程和计算机科学系教授。他的许多研究课题都涉及人工智能和软计算。他一直致力于电气工程、过程控制和环境工程中智能系统的开发和应用，发表了300多篇论文，著有2本专著，并获得了4项发明专利。



影印版
书号：978-7-111-35822-0
定价：49.00元

客服热线：(010) 88378991, 88361066
购书热线：(010) 68326294, 88379649, 68995259
投稿热线：(010) 88379604
读者信箱：hzjsj@hzbook.com

华章网站 http://www.hzbook.com

网上购书：www.china-pub.com

上架指南：计算机/人工智能

ISBN 978-7-111-38455-7

9 787111 384557

定价：49.00元

计 [redacted] 丛 书

原书第3版

人工智能
智能系统指南

(澳) Michael Negnevitsky 著 陈薇 等译

Artificial Intelligence
A Guide to Intelligent Systems Third Edition



机械工业出版社
China Machine Press

本书是一本很好的人工智能入门书籍，内容丰富、浅显易懂。作者根据自己多年的教学、实践经验，并结合实际代码、图示、案例等讲解了人工智能的基本知识。

全书共分10章，主要内容包括：基于规则的专家系统、不确定性管理技术、模糊专家系统、基于框架的专家系统、人工神经网络、进化计算、混合智能系统、知识工程、数据挖掘等。另外，本书还提供了一个人工智能相关术语表和包含商业化的人工智能工具的附录。

本书既可以作为计算机科学相关专业本科生的入门教材，也可以作为非计算机科学专业读者的自学参考书。

Michael Negnevitsky: Artificial Intelligence: A Guide to Intelligent Systems, Third Edition (ISBN 978-1-4082-2574-5).

Copyright © 2011 by Pearson Education Limited.

This translation of Artificial Intelligence: A Guide to Intelligent Systems, Third Edition (ISBN 978-1-4082-2574-5) is published by arrangement with Pearson Education Limited.

All rights reserved.

本书中文简体字版由英国 Pearson Education 培生教育出版集团授权出版。

封底无防伪标均为盗版

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：01-2011-4255

图书在版编目(CIP)数据

人工智能：智能系统指南（原书第3版）/（澳）尼格尼维斯基（Negnevitsky, M.）著；陈薇等译. —北京：机械工业出版社，2012.7

（计算机科学丛书）

书名原文：Artificial Intelligence: A Guide to Intelligent Systems, Third Edition

ISBN 978-7-111-38455-7

Ⅰ.人…　Ⅱ.①尼…　②陈…　Ⅲ.人工智能　Ⅳ.TP18

中国版本图书馆CIP数据核字（2012）第104490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：高婧雅

北京市荣盛彩色印刷有限公司印刷

2012年8月第1版第1次印刷

185mm×260mm · 20.75印张

标准书号：ISBN 978-7-111-38455-7

定价：49.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：(010) 88378991；88361066

购书热线：(010) 68326294；88379649；68995259

投稿热线：(010) 88379604

读者信箱：hzjsj@hzbook.com

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，还揭示了学术的源变，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和发展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到积极的推动作用，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公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S. Tanenbaum，Bjarne Stroustrup，Brain W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson 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、研究及珍藏。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，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，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。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

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。随着计算机科学与技术专业学科建设的不断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将步入一个新的阶段，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。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：

华章网站：www. hzbook. com
电子邮件：hzjsj@ hzbook. com
联系电话：(010) 88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037



华章科技图书出版中心

译 者 序
Artificial Intelligence, 3E

人工智能（Artificial Intelligence，AI）是计算机学科的一个分支，被认为是21世纪三大尖端技术（基因工程、纳米科学、人工智能）之一。“人工智能”一词最初是在1956年达特茅斯学院研讨会上提出的。从那以后，研究者们发展了众多理论和原理，人工智能的概念也随之扩展。在经历了辉煌和低谷之后，近30年来，人工智能的研究取得了迅速的发展，在很多领域（例如工程、医疗、财经、商业和管理等）都获得了广泛应用。人工智能的研究逐渐成熟，已成为一个独立的分支，无论在理论和实践上都已自成一个系统。

本书作者Michael Negnevitsky是澳大利亚塔斯马尼亚大学电气工程和计算机科学系教授。他的许多研究课题都涉及人工智能和软计算，一直致力于电气工程、过程控制和环境工程中智能系统的开发和应用。他著有200多篇论文、两本书，并获得了四项发明专利。

本书是一本很好的人工智能入门书籍，内容丰富、浅显易懂、适应面广。美国、德国、日本等多所大学的计算机相关专业都采用本书作为教材或主要教学参考书。与第2版相比，第3版引入了关于“数据挖掘”的新的一章，并展示了智能工具在解决复杂实际问题中的新应用。第3版还扩展了本书的参考文献和参考书目，并更新了附录中的人工智能工具和厂商列表。通过本书的学习，相信读者一定会对人工智能的完整知识体系有全面的了解。

陈薇组织并参与了本书的翻译和审校工作，参加翻译工作的还有闫秋玲、黄威靖、欧高炎、刘璐。由于译者水平有限，译文中疏漏和错误之处在所难免，欢迎广大读者批评指正。

陈 薇

2012年于北京大学

第3版前言
Artificial Intelligence, 3E

这本书的主要目的与第1版相同，即为读者提供一本能实际了解计算机智能领域相关知识的书。它适合作为一个学期课程的入门教程，学生需要具备一些微积分的知识，不要求具备编程的经验。

在涵盖内容方面，本书引入了关于数据挖掘的新的一章，并展示了智能工具在解决复杂实际问题中的新应用。主要的变化如下：

在新的“数据挖掘和知识发现”一章中，我们介绍了大型数据库中知识发现不可或缺的一部分——数据挖掘。涉及将数据转换为知识的主要技术和工具，包括统计方法、数据可视化工具、结构化查询语言、决策树和购物篮分析。同时还展示了几个数据挖掘应用的实例。

在第9章中，增加了采用自组织神经网络进行聚类分析的新的实例。

最后，我们还扩展了本书的参考文献和参考书目，更新了附录中的人工智能工具和厂商列表。

Michael Negnevitsky
2010年9月于澳大利亚
塔斯马尼亚州　霍巴特市

第1版前言

Artificial Intelligence, 3E

"The only way not to succeed is not to try."

——Edward Teller

又是一本人工智能的书……我已经见过很多同类的书，为什么还要理会它？它有什么与众不同之处？

每年，有成百上千本书和博士论文拓展着计算机或人工智能的知识体系。专家系统、人工神经网络、模糊系统以及进化计算是应用于智能系统的主要技术，数百个工具支持着这些技术，数以千计的科学论文不断推进着该学科的发展。本书中的任何章节的内容都可以作为一本书的主题。然而，我想写一本能够阐述智能系统基础的书，更为重要的原因是，我想消除大家对人工智能理论的畏惧心理。

大多数人工智能文献是采用计算机科学的专业术语进行描述的，其中充斥着大量复杂的矩阵代数和微分方程，这当然给人工智能理论带来了令人敬佩的资本，但同时也令非计算机科学的科学家对其敬而远之。不过，这种情况已经有所改变！

个人电脑已经成为我们日常生活中不可或缺的部分，我们将它用于打字机和计算器、日历和通信系统、交互式数据库以及决策支持系统。并且我们还渴望更多。我们希望计算机智能化！我们发现智能系统正快速地走出实验室，而我们也想更好地利用它。

智能系统的原理是什么？它是如何构建的？智能系统的用处是什么？我们该如何选择适当的工具构建智能系统？这些问题都可以在本书中找到答案。

与许多介绍计算机智能的书不同，本书将智能系统背后的奥妙简单明了地展示给读者。它是基于作者多年来给没有多少微积分知识的学生授课时所用的讲义而编写的，读者甚至不用学习任何编程语言就可轻松理解！书中的素材已经经过笔者15年的教学实践的检验，写作中也考虑了学生们提出的典型问题和建议。

本书是一本计算机智能领域的入门书籍，内容包括基于规则的专家系统、模糊专家系统、基于框架的专家系统、人工神经网络、进化计算、混合智能系统、知识工程和数据挖掘。

总体来说，本书可作为计算机科学、计算机信息系统和工程专业的本科生的入门教材。我在教学过程中，会要求学生开发小型的基于规则和基于框架的专家系统，设计一个模糊系统，探究人工神经网络，采用遗传算法求解一个简单的优化问题，并开发混合的神经模糊系统。他们使用一些专家系统的核心程序（XpertRule、Exsys Corvid 和 Visual Rule Studio）、MATLAB 的模糊逻辑工具箱以及 MATLAB 的神经网络工具箱。我们选择这些工具的原因是它们能够便利地演示教学中的原理。然而，本书并不局限于任何特定的工具，书中给出的例子可以轻松地在不同的工具中实现。

本书也适合非计算机科学专业的相关人士自学。对于他们来说，本书提供了进入基于知识

的系统和计算智能的前沿领域的钥匙。事实上，本书面向的专业读者群十分广泛：工程师和科学家、管理人员和商人、医生和律师，也就是所有面临挑战而无法用传统的方法解决问题的人，所有想了解计算机智能领域巨大成就的人。本书将帮助你实际了解智能系统的用途，发现与你的工作密切相关的工具，并最终学会如何使用这些工具。

希望读者能与我共同分享人工智能和软计算学科所带来的乐趣，并从本书中获益。

读者可以访问 http://www.booksites.net/negnevitsky 获得更多的信息。

Michael Negnevitsky
2001 年 2 月于澳大利亚
塔斯马尼亚州 霍巴特市

本书摘要

Artificial Intelligence, 3E

本书一共包含10章。

第1章简要介绍了人工智能的历史，从20世纪中期诞生人工智能的思想并在60年代设立了远大目标并积极实现到20世纪70年代早期理想破灭和资金投入大幅削减；从20世纪70年代第一代专家系统（DENDRAL、MYCIN、PROSPECTOR）的诞生到20世纪八九十年代专家系统技术的成熟和其在不同领域的广泛应用；从20世纪40年代简单的二元神经元模型的提出到20世纪80年代人工神经网络领域的复苏；从20世纪60年代模糊集理论的提出和它的存在被西方忽视到20世纪80年代日本生产出大量模糊用户产品，以及20世纪90年代软计算和文字计算在世界范围内的广泛接受。

第2章提供了基于规则的专家系统概述。作者简要介绍了知识的概念，以及专家用产生式规则表示知识的过程。作者介绍了专家系统开发团队的主要成员和基于规则系统的结构。分析了专家系统的基本特性，并指出专家系统不是万无一失的。然后回顾了前向链接和后向链接推理技术，并讨论了冲突消解策略。最后分析了基于规则的专家系统的优缺点。

第3章展示了专家系统使用的两种不确定性管理技术：贝叶斯推理和确信因子。作者分析了不确定知识的主要来源，并简要回顾了概率理论。作者考虑了可累积论据的贝叶斯方法并开发一个简单的基于贝叶斯方法的专家系统。然后讨论确信因子理论（贝叶斯推理的常用替代方法），并开发一个基于论据推理的专家系统。最后，比较贝叶斯推理和确信因子理论并分析它们的适用范围。

第4章介绍了模糊逻辑并讨论其背后的哲学思想。首先介绍模糊集的概念，考虑如何在计算机里表示一个模糊集并介绍了模糊集的操作。作者还定义了语言变量和模糊限制语（hedge）。然后作者叙述了模糊规则，并解释了经典规则和模糊规则的主要区别。该章主要研究两种模糊推理技术：Mamdani法和Sugeno法，并就它们适宜的应用领域给出建议。最后介绍开发一个模糊专家系统的主要步骤，并通过构建和调试模糊系统的具体过程来阐明其理论。

第5章概述了基于框架的专家系统。介绍了框架的概念，并讨论如何将框架用于知识表达，以及阐明继承是基于框架系统的基本特征，还讨论了方法、守护程序和规则的应用。最后通过一个实例来介绍基于框架的专家系统的开发。

第6章介绍了人工神经网络，并讨论了机器学习的基本思想。叙述作为一个简单的计算单元的感知器的概念，并讨论了感知器的学习规则，还探索了多层神经网络，以及讨论了如何提高反向传播学习算法的计算效率。然后介绍循环神经网络，思考Hopfield网络训练算法和双向联想记忆（BAM）。最后介绍自组织神经网络并探讨Hebbian学习规则和竞争学习。

第7章概述了进化计算，其中包括遗传算法、进化策略和遗传编程。首先介绍了开发一个遗传算法的主要步骤，讨论遗传算法的工作机制，并通过具体的遗传算法应用阐明其理论。然后叙述一个进化策略的基本概念，比较了进化策略和遗传算法之间的不同。最后考虑遗传编程以及它的实际应用。

第8章讨论了结合不同智能技术的混合智能系统。首先介绍一种新型的专家系统——神经专

家系统，它将神经网络和基于规则的专家系统结合起来。然后考虑一个功能上等同于 Mamdani 模糊推理模型的神经－模糊系统，以及一个功能上等同于 Sugeno 模糊推理模型的自适应神经模糊推理系统（ANFIS）。最后讨论进化神经网络和模糊进化系统。

第 9 章讨论了知识工程。首先讨论智能系统可以解决什么样的问题，并介绍知识工程过程的 6 个主要阶段。然后展示了专家系统、模糊系统、神经网络和遗传算法的典型应用。作者演示了如何构建智能系统来解决诊断、选择、预测、分类、聚类和优化问题。最后讨论了混合神经－模糊系统在决策支持和时序预测方面的应用。

第 10 章介绍了数据挖掘的概貌，讨论了将数据转换为知识的主要技术。首先，宽泛地定义了数据挖掘，并解释了在大型数据库中进行数据挖掘和知识发现的过程。介绍了一些统计学方法，包括主成分分析，并讨论了它们的局限性。随后展示了关系数据库中结构化查询语言的应用，介绍了数据仓库和多维数据分析。最后，介绍了最流行的数据挖掘工具——决策树和购物篮分析。

本书还包括一个术语表和一个附录。术语表包含了 300 余条用于专家系统、模糊逻辑、神经网络、进化计算、知识工程以及数据挖掘领域的定义。附录中提供了商业化的人工智能工具的列表。

本书的网站地址为 http://www.booksites.net/negnevitsky。

致　谢

Artificial Intelligence, 3E

在本书的出版过程中，我直接或间接地得到了很多人至诚的帮助。我首先要感谢 Vitaly Faybisovich 博士，他不仅对我在软计算领域的研究提出了许多建设性的意见，还在我过去 30 年间的奋斗生涯中与我建立了真诚的友谊并给予了我大力支持。

我还要感谢许许多多为本书提出评论和有益建议的检阅人，感谢 Pearson Education 的编辑们，特别是 Keith Mansfield、Owen Knight、Liz Johnson 和 Rufus Curnow，他们帮助我完成了本书的出版发行工作。

我要感谢我在澳大利亚塔斯马尼亚大学的本科生和研究生们，特别是我以前的博士生 Tan Loc Le、Quang Ha、Steven Carter 和 Mark Lim，他们对新知识的渴求对我来说既是挑战也是激励。

我想感谢美国波士顿大学的 Stephen Grossberg 教授、德国马德堡大学的 Frank Palis 教授、日本广岛大学的 Hiroshi Sasaki 教授、美国罗切斯特理工学院的 Walter Wolf 教授以及东京工业大学的 Kaoru Hirota 教授，他们为我提供了在学生中对书的内容进行测试的机会。

我还要衷心地感谢 Vivienne Mawson 和 Margaret Eldridge 博士对本书初稿的审读工作。

尽管本书的第 1 版出版了十多年，但是已经有很多人用过它并向我提出了宝贵的意见和建议，由于人数众多，在此不能一一致谢，但我至少要感谢那些提出了特别有益建议的人们：Martin Beck（英国普利茅斯大学）、Mike Brooks（澳大利亚阿德莱德大学）、Genard Catalano（美国哥伦比亚大学）、Warren du Plessis（南非比勒陀利亚大学）、Salah Amin Elewa（埃及美洲大学）、Michael Fang（中国浙江大学）、John Fronckowiak（美国曼达尔学院）、Patrick B. Gibson（加拿大温莎大学）、Lev Goldfarb（加拿大 New Brunswick 大学）、Susan Haller（美国威斯康星大学）、Evor Hines（英国沃里克大学）、Philip Hingston（澳大利亚埃迪斯科文大学）、Sam Hui（美国斯坦福大学）、Yong - Hyuk Kim（韩国光云大学）、David Lee（英国赫特福德郡大学）、Andrew Nunekpeku（加纳大学）、Vasile Palade（英国牛津大学）、Leon Reznik（美国罗切斯特理工学院）、Simon Shiu（中国香港理工大学）、Boris Stilman（美国科罗拉多大学）、Thomas Uthmann（德国美因茨约翰尼斯·古腾堡大学）、Anne Venables（澳大利亚维多利亚大学）、Brigitte Verdonk（比利时安特卫普大学）、Ken Vollmar（美国西南密苏里州立大学）、Kok Wai Wong（新加坡南洋理工大学）以及 Georgios N. Yannakakis（丹麦哥本哈根信息技术大学）。

出版者的话
译者序
第3版前言
第1版前言
本书概要
致谢

第1章 基于知识的智能系统概述 …… 1
1.1 智能机 …… 1
1.2 人工智能的发展历史，从“黑暗时代”到基于知识的系统 …… 3
1.2.1 “黑暗时代”，人工智能的诞生（1943～1956年） …… 3
1.2.2 人工智能的上升期，远大目标积极实现的年代（1956年～20世纪60年代晚期） …… 4
1.2.3 没有履行的诺言，来自现实的冲击（20世纪60年代晚期～20世纪70年代早期） …… 5
1.2.4 专家系统技术，成功的关键因素（20世纪70年代早期～20世纪80年代中期） …… 5
1.2.5 如何使机器学习，神经网络的重生（20世纪80年代中期至今） …… 8
1.2.6 进化计算，在尝试中学习（20世纪70年代早期至今） …… 9
1.2.7 知识工程的新纪元，文字计算（20世纪80年代后期至今） …… 9
1.3 小结 …… 11
复习题 …… 13
参考文献 …… 13
第2章 基于规则的专家系统 …… 16
2.1 知识概述 …… 16
2.2 知识表达技术——规则 …… 16
2.3 专家系统研发团队的主要参与者 …… 18
2.4 基于规则的专家系统的结构 …… 19
2.5 专家系统的基本特征 …… 20
2.6 前向链接和后向链接推理技术 …… 21
2.6.1 前向链接 …… 22
2.6.2 后向链接 …… 23
2.7 MEDIA ADVISOR：基于规则的专家系统实例 …… 25
2.8 冲突消解 …… 29
2.9 基于规则的专家系统的优点和缺点 … 31
2.10 小结 …… 32
复习题 …… 33
参考文献 …… 34
第3章 基于规则的专家系统中的不确定性管理 …… 35
3.1 不确定性简介 …… 35
3.2 概率论基本知识 …… 36
3.3 贝叶斯推理 …… 39
3.4 FORECAST：论据累积的贝叶斯方法 … 41
3.5 贝叶斯方法的偏差 …… 46
3.6 确信因子理论和基于论据的推理 …… 47
3.7 FORECAST：确信因子的应用 …… 51
3.8 贝叶斯推理和确信因子的对比 …… 52
3.9 小结 …… 53
复习题 …… 53
参考文献 …… 54
第4章 模糊专家系统 …… 56
4.1 概述 …… 56
4.2 模糊集 …… 57
4.3 语言变量和模糊限制语 …… 60
4.4 模糊集的操作 …… 63
4.5 模糊规则 …… 66
4.6 模糊推理 …… 68
4.6.1 Mamdani-style 推理 …… 68

4.6.2 Sugeno-style 推理 …… 73
4.7 建立模糊专家系统 …… 75
4.8 小结 …… 82
复习题 …… 83
参考文献 …… 83
参考书目 …… 84
第5章 基于框架的专家系统 …… 86
5.1 框架简介 …… 86
5.2 知识表达技术——框架 …… 87
5.3 基于框架的系统中的继承 …… 91
5.4 方法和守护程序 …… 94
5.5 框架和规则的交互 …… 97
5.6 基于框架的专家系统实例：Buy Smart … 99
5.7 小结 …… 108
复习题 …… 109
参考文献 …… 109
参考书目 …… 110
第6章 人工神经网络 …… 111
6.1 人脑工作机制简介 …… 111
6.2 作为简单计算元素的神经元 …… 113
6.3 感知器 …… 114
6.4 多层神经网络 …… 117
6.5 多层神经网络的加速学习 …… 123
6.6 Hopfield 网络 …… 126
6.7 双向联想记忆 …… 131
6.8 自组织神经网络 …… 133
6.8.1 Hebbian 学习 …… 133
6.8.2 竞争学习 …… 136
6.9 小结 …… 141
复习题 …… 143
参考文献 …… 143
第7章 进化计算 …… 145
7.1 进化是智能的吗 …… 145
7.2 模拟自然进化 …… 145
7.3 遗传算法 …… 146
7.4 遗传算法为什么可行 …… 153
7.5 案例研究：用遗传算法来维护调度 …… 154
7.6 进化策略 …… 160
7.7 遗传编程 …… 161
7.8 小结 …… 167
复习题 …… 167
参考文献 …… 168
参考书目 …… 169
第8章 混合智能系统 …… 170
8.1 概述 …… 170
8.2 神经专家系统 …… 171
8.3 神经－模糊系统 …… 176
8.4 ANFIS …… 182
8.5 进化神经网络 …… 188
8.6 模糊进化系统 …… 192
8.7 小结 …… 195
复习题 …… 196
参考文献 …… 197
第9章 知识工程 …… 198
9.1 知识工程简介 …… 198
9.1.1 问题评估 …… 198
9.1.2 数据和知识获取 …… 199
9.1.3 原型系统开发 …… 200
9.1.4 完整系统开发 …… 201
9.1.5 系统评价和修订 …… 201
9.1.6 系统集成和维护 …… 201
9.2 专家系统可以解决的问题 …… 202
9.3 模糊专家系统可以解决的问题 …… 209
9.4 神经网络可以解决的问题 …… 214
9.5 遗传算法可以解决的问题 …… 226
9.6 混合智能系统可以解决的问题 …… 229
9.7 小结 …… 236
复习题 …… 237
参考文献 …… 239
第10章 数据挖掘和知识发现 …… 241
10.1 数据挖掘简介 …… 241
10.2 统计方法和数据可视化 …… 243
10.3 主成分分析 …… 247
10.4 关系数据库和数据库查询 …… 255
10.5 数据仓库和多维数据分析 …… 258
10.6 决策树 …… 265
10.7 关联规则和购物篮分析 …… 271
10.8 小结 …… 277
复习题 …… 278
参考文献 …… 279
术语表 …… 281
附录 人工智能工具和经销商 …… 295
索引 …… 310

基于知识的智能系统概述

本章讨论何为智能以及机器能否真正实现智能化。

1.1 智能机

哲学家们两千多年来一直致力于理解和解决宇宙的两大问题：人类是如何思考的？人类以外的物体是否也能够思考？至今，这两个问题依旧没有答案。

一部分哲学家接受由计算机领域的科学家提出的计算方法，认可人类所做的任何事情机器都能够做到的观点。另一部分哲学家则非常反对这一观点，认为一些高级复杂的行为（例如爱、创新、道德判断）远超出了任何机器的能力范围。

哲学的特质允许这些争论一直存在。但事实上，工程师和科学家们已经造出了具有“智能”的机器。何为“智能”？下面是来自《柯林斯基础英语词典》的定义：

（1）智能是人类理解和学习事情的能力。

（2）智能是思考和理解事情的能力，而非本能或机械地做事情。

——《柯林斯基础英语词典》2008年版

按照第一条定义，智能是专属于人类的一个品质，但第二条定义更具弹性，指出只要具有思考和理解能力就具有智能，而不限制主体是人还是物。而定义里的“思考”是什么意思呢？再次看词典中的定义：

思考是使用大脑考虑问题或创建新想法的活动。

——《柯林斯基础英语词典》2008年版

为了能够思考，人或物需要拥有像大脑那样的一个器官，依赖它去学习、理解事物，去处理问题和做出决策。因此智能可以定义为“学习和理解事物、处理问题并做出决策的能力”。

计算机能否智能化，即机器能否思考，这一问题源于人工智能的“黑暗时代”（20世纪40年代晚期开始）。作为一门科学，人工智能（Artificial Intelligence，AI）的目标是使机器像人那样具有智能去做事情（Boden，1977）。因此，关于“机器能否思考”这一问题的答案对人工智能这一学科至关重要。答案本身不是“能”或“不能”之类的简单判断，而必定是模糊的。我们从常识中也能得出这个结论：比如一部分人在某些方面比其他人聪明；日常生活中做出的抉择有些是明智的，有些则很愚蠢的；有人可能精通复杂的数学、工程问题，但对于哲学或历史却一窍不通；有人善于赚钱，而有人更善于花钱。虽然人类中的每一个成员都具有学习和理解事物、处理问题并做出决策的能力，但比较起来，这种能力在每一个领域都不均等。所以，如果机器能够思考，一部分机器在某些方面也许比另一部分机器聪明。

英国数学家阿兰·图灵（Alan Turing）在50多年前撰写的论文“计算机器和智能化”（Turing，1950）是关于机器智能方面最早和最有影响的文章之一。图灵的思想经住了时间的考验，到现在为止仍是通用的。

阿兰·图灵在20世纪30年代早期重新发现了中心极限定理，他也由此开始了科学生涯。1937年，他发表了可计算数字的论文，在这篇文章中提出了通用机器的概念。之后在二战期间，他在破解德国军用编码机Enigma的工作中担任了重要角色。第二次世界大战结束后，他设计了“自动计算机器”，并设计了第一个国际象棋比赛程序，后来在曼彻斯特大学的计算机上得以实

戏。图灵关于通用机的理论概念和译码的实践经验使他在人工智能的关键性基础问题方面取得进展。他曾经提出这样的问题：是否存在不需经验的思考？是否存在不需交流的思考？是否存在不使用的语言？是否存在生命之外的智能？显然，这些问题都是人工智能中的基础问题——"机器能否思考？"的不同表达。

图灵并没有对机器和思考进行定义，而是设计了一个游戏——图灵模拟游戏，用游戏阐述他对机器和思考的理解，从而避免了关于定义的异议。图灵认为，与其问"机器能否思考"，不如问"机器能否通过智能行为测试"。他曾经预测，到公元2000年时，计算机能够通过程序与人进行5分钟对话，并有30%的机会使对话的人认为对方不是计算机，而是人。图灵将计算机
2 智能行为定义为在认知任务中达到人类行为水平的能力。换句话说，如果对话者依据所提问题的答案无法判断对方是人还是机器，计算机就通过了测试。

图灵模拟游戏包括两个阶段。在第一阶段，如图1.1所示，一个审讯员和一男一女分别被安排在单独的空间，他们只能通过远程终端之类的中介进行交流。审讯员的目标是通过对另外两个人提问，根据他们的回答来判断谁是男的、谁是女的。游戏规则是男士要尽力让审讯员认为他是女士，而女士则需向审讯员证明她是女士。

游戏的第二个阶段，如图1.2所示，用一台计算机代替男士，计算机的任务和第一阶段中男士的任务一样。对计算机编程时，为了更好地模拟人类，故意让计算机犯错或提供模糊的答案。如果计算机成功"骗过"审讯员的次数和男士成功"骗过"审讯员的次数一致，则认为计算机通过了智能行为测试。

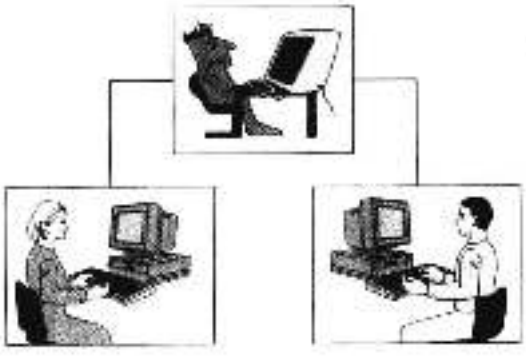


图1.1 图灵模拟游戏：第一阶段

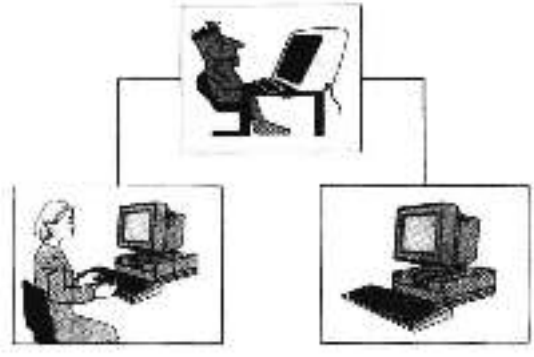


图1.2 图灵模拟游戏：第二阶段

对于人的身体模拟在智能测试中并不重要。因此，图灵测试中，审讯员看不见、碰不到、听不到计算机，因此不会受计算机的外表或声音的干扰。但为了辨别出计算机，审讯员可以问任
3 何问题，甚至是过激的问题。例如，审讯员可以让人和计算机做复杂的数学计算，如果一方提供了正确答案，并且比对方速度快，可以判断这一方就是计算机。因此，计算机必须知道何时故意出错，何时延迟回答。审讯员也可能会问双方关于短篇小说、诗歌、绘画方面的问题，由此可以推断出一方具有的明显的人类专属的情绪特征。当面对这一类问题时，计算机就必须模拟人类对事物的感性理解。

图灵测试很通用，原因是测试具备了两个显著特征：

- 由于使用终端进行人机交互，这一测试让我们对智能有客观的衡量标准，避免了对人类智能本质的争论，并排除了偏向人类的可能性。
- 测试本身独立于实验细节。既可以像前述的进行两阶段测试，也可以变成一个阶段的测试，从一开始就让审讯员对人与机器进行判断。审讯员既可以提任何领域的任何问题，也可以只关注答案本身。

图灵相信到 20 世纪末，能够通过对数字计算机编程来完成测试游戏。尽管现代计算机还没有通过图灵测试，但已经为我们提供了基于知识的系统的评估和验证基础。通过与人类专家行为对比，已经对一些狭窄专业领域的具有智能的程序进行了评估。

我们的大脑具有 10^{18} 位的容量，能够每秒处理 10^{15} 位的信息。到 2020 年，也许一块方糖大小的芯片就足以模拟人脑，也许会有计算机能够通过图灵测试游戏。但是，我们真的想让计算机在做数学计算时，像人一样又慢又会出错吗？实际上，智能机应该能够帮助人类进行决策、搜索信息、控制复杂对象，最终理解单词含义。也许没有必要片面追求使计算机像人一样聪明这一既抽象又艰涩的目标。为了构建一套智能机系统，我们必须去捕捉、组织和使用某个狭窄专业领域的人类专家知识。

1.2　人工智能的发展历史，从“黑暗时代”到基于知识的系统

人工智能成长为一门科学，历经三代研究者的努力。这一节就来介绍每一代研究者对人工智能学科所做的贡献和重大事件。

4

1.2.1　“黑暗时代”，人工智能的诞生（1943 ~ 1956 年）

Warren McCulloch 和 Walter Pitts 在 1943 年发表了人工智能领域的开篇之作。McCulloch 拥有哥伦比亚大学的哲学和医学学位，后来在伊利诺伊大学的精神病学系担任基础研究实验室的主任。他在对中枢神经系统的研究中提出了大脑神经元模型，最早为人工智能做出了主要贡献。

McCulloch 和年轻的数学家 Walter Pitts 合作，提出了人工神经网络模型，这个模型假定每个神经元有两个状态：开、关（McCulloch and Pitts，1943）。并证明他们的神经网络模型实际上等价于图灵机，并且任何可计算函数都可以由相连的神经元网络进行计算。同时揭示了简单的网络结构具有学习功能。

神经元网络模型从理论和实验两个方面激发了对大脑模型的研究。但是，实验结果清晰地表明将神经元限制为两个状态是不正确的。事实上，神经元具有明显的非线性特征，而非简单的二元状态。即使如此，McCulloch 是人工智能领域继阿兰 · 图灵之后的第二个创始人，为神经计算和人工智能网络（Artificial Neural Networks，ANN）奠定了基础。人工神经网络的研究在 20 世纪 70 年代经历了衰退，在 20 世纪 80 年代后期重新复苏。

人工智能的第三个奠基人是聪明的匈牙利籍数学家 John von Neumann。他在 1930 年加入了普林斯顿大学，在数学物理系任教。他是阿兰 · 图灵的同事和朋友。在第二次世界大战期间，von Neumann 在建造原子弹的“曼哈顿项目”中担任重要角色，并成为宾夕法尼亚大学的 ENIAC（Electronic Numerical Integrator and Calculator，电子数字集成计算器）项目的顾问，参与设计了 EDVAC 计算机（Electronic Discrete Variable Automatic Computer，电子离散变量自动计算机）⊖。McCulloch 和 Pitts 所提出的神经网络模型使 von Neumann 受到启发。因此，在 1951 年，当来自普林斯顿大学数学系的两个研究生 Marvin Minsky 和 Dean Edmonds 建造第一台神经网络计算机时，von Neumann 鼓励并资助了他们。

在第一代研究者中，不得不提 Claude Shannon。他毕业于麻省理工学院并于 1941 年加入贝尔实验室。Shannon 继承了阿兰 · 图灵的观点，认为可以使机器智能化。在 1950 年，他发表了一篇关于让机器下象棋的文章，文章指出一盘典型的棋赛大约有 10^{120} 次的移动（Shannon，1950）。即

⊖ EDVAC 是一个存储程序的机器。

使一台新的 von Neumann 式计算机每微秒就能测试一种走法，移动第一步也得花上 3×10^{106} 年。因此，Shannon 论证了在寻找解决方案时使用启发式策略的必要性。

另一位人工智能奠基人 John McCarthy 也来自普林斯顿大学。他从普林斯顿大学毕业后去达
5 特茅斯学院工作。他说服了 Marvin Minsky 和 Claude Shannon 在达特茅斯学院组织一个暑期研讨会。在 1956 年，他们召集了对机器智能、人工神经网络和自动理论感兴趣的研究者，参加由 IBM 赞助的研讨会。尽管研讨会仅有 10 人参加，但这次会议却促成了人工智能学科的诞生。在之后的 20 年里，参加过达特茅斯学院研讨会的学者及他们的学生在人工智能领域一直起主导作用。

1.2.2　人工智能的上升期，远大目标积极实现的年代（1956 年 ~20 世纪 60 年代晚期）

在人工智能学科的发展早期，可以用热情巨大、想法大胆，但成果很少来形容。当时，计算机被用来做常规数学计算才几年时间，而人工智能领域的研究者却已在论证计算机的本领远超过这些。这真是一个有远大目标并积极付诸实现的年代。

John McCarthy 作为达特茅斯学院研讨会的组织者之一，也是“人工智能”这一术语的发明者，从达特茅斯学院转到了麻省理工学院。他设计了高级语言 LISP。LISP 是早期编程语言之一（FORTRAN 仅比它早了两年），到今天还有人在用。在 1958 年，McCarthy 发表了论文“基于常识的程序”，文中提出了一个名为“*建议采纳者*”（Active Taker）的程序，用于搜索一般性问题的解决方案（McCarthy，1958）。McCarthy 论证了设计和生成程序的方法，他以驾车去机场为例，采用这个程序基于几个简单的规则设计了驾驶计划。更重要的是，程序无须变动，就可以接受来自不同专业领域的新的规则，即可以接受新知识。因此，“建议采纳者”是第一个真正集成了知识表达和推理的核心法则的基于知识的系统。

达特茅斯学院研讨会的另一个组织者 Marvin Minsky 也加入了麻省理工学院。他不像 McCarthy 那样专注于形式逻辑，而是建立了一套反逻辑的观点，以处理知识表达和推理。他的框架理论为知识工程做出了主要的贡献（Minsky，1975）。

由 McCulloch 和 Pitts 开创的神经计算和人工神经网络的早期研究得以继续。随着学习方法的改进和提高，Frank Rosenblatt 证明了*感知器收敛理论*，并论证他的学习算法能够调整感知器的连接程度（Rosenblatt，1962）。

GPS（General Problem Solver，通用问题解决方案）是在这个大胆假设年代最具野心的项目之一（Newell and Simon，1961；1972）。卡内基 - 梅隆大学的 Allen Newell 和 Herbert Simon 设计了一个通用程序，用于模拟人类在解决问题时的方法。GPS 也许是第一个尝试将解决问题的技术
6 和数据分开的程序。它使用了现在已被称为*手段-目的分析*（means-ends analysis）的技术。Newell 和 Simon 假定可以用状态去定义问题。手段 - 目的的作用是判断问题的当前状态和期望状态或*目标状态*的差距，并选择操作算子去达到目标状态。如果从当前状态不能直接到达目标状态，就要建立一个更接近目标状态的新状态，重复这一过程直到达到目标状态。操作算子集合决定了解决方案。

但是，GPS 无法解决复杂问题。由于程序建立在形式逻辑的基础上，操作算子可能有无穷多个，导致效率低下。在解决实际问题时，GPS 需要大量运算时间和内存，这导致人们后来放弃 GPS 计划。

总之，在 20 世纪 60 年代，为了模拟复杂的思考过程，人工智能研究者试图研究*通用方法*来解决广泛的问题。他们使用了通用搜索策略以寻找解决方案。通用搜索策略现在被称为*弱方法*，由于使用的是问题域的弱信息，导致程序的性能低下。

不管怎样，这个发展阶段吸引了伟大的科学家针对人工智能各个方面提出创新性的基础理论，例如在知识表达、学习算法、神经计算以及文字计算等诸方面。当时由于计算机性能的限制，这些理论并未实现，但却为20年后的实际应用指明了方向。

值得一提的是，来自加州大学伯克利分校的Lotfi Zadeh教授也在20世纪60年代发表了著名的文章“模糊集”（Zadeh，1965）。现在人们一致认为这篇文章为模糊集理论奠定了基础。20年后，模糊集理论研究者已经构建了成百上千的智能机和智能系统。

人工智能令人鼓舞的时代到1970年结束，大部分政府资助的人工智能项目也被迫取消。人工智能仍是一个相对新的领域，偏于学术性，除了一些游戏之外几乎没有实际应用（Samuel，1959；1967；Greenblatt et al.，1967）。在外界看来，由于不存在能解决实际问题的人工智能系统，所取得的成绩也只是小玩意而已。

1.2.3 没有履行的诺言，来自现实的冲击（20世纪60年代晚期~20世纪70年代早期）

从20世纪50年代中期，人工智能研究者就承诺到20世纪80年代建造出与人类智能相当的全能智能机，到公元2000年，建造出超越人类智能的智能机。但到1970年时，他们已经意识到这些许诺过于乐观。虽然个别智能程序能够在一两个小问题上展示出一定水平的智能，但几乎所有的人工智能项目都对稍宽泛的问题和实际的复杂问题无能为力。 7

在20世纪60年代晚期，人工智能学科存在的问题主要表现在以下几个方面。

- 由于人工智能研究者专注于开发能解决广义问题的一般方法，因此早期的程序只包含很少甚至没有相关问题域的知识。为了解决问题，程序中应用了一种搜索策略，尝试对小步骤进行不同的组合，直到找出真正的解决方案。由于这种思路对于小问题是有效的，因而被误认为可以通过对程序进行扩展来解决更大的问题。事实上，这种想法是错误的。

简单的而易处理的问题在多项式级的时间内就得以解决，就是说，对于一个规模为n的问题，所需的求解时间或求解步骤数是n的多项式函数。而解决棘手而不易处理的问题所需的时间是问题规模的指数级函数。一般认为，多项式级运算时间的算法是高效的，而指数级运算时间的算法则是低效的，原因是后者的运算时间随着问题规模的增大而呈飞速增长。20世纪70年代早期所提出的NP完全问题理论（NP-Completeness）（Cook，1971；Karp，1972）揭示出一大类非确定多项式级问题（Non-deterministic Polynomial problem，NP问题）都是NP完全问题。NP问题是指能够在多项式时间内估算并进行验证的问题（假设有解）。非确定是指不存在特定算法来进行估算。而NP完全问题则是NP问题中最难的问题，即使用速度最快、内存最大的计算机也难以求解。

- 许多人工智能试图解决的问题都太宽泛、太复杂。早期人工智能的一个典型任务就是机器翻译。例如，在前苏联于1957年发射第一颗人造卫星Sputnik之后，美国国家研究委员会就资助针对俄语科学文献的机器翻译项目。最初，项目团队想利用电子词典直接将俄语单词替换为对应的英语单词。不久他们便发现只有了解了文献主题才能正确选择单词。由于任务太难，1966年由美国政府资助的所有翻译项目都被取消。
- 在1971年，英国政府也终止了对人工智能研究的资助。英国科学研究委员会派James Lighthill爵士调研当时的人工智能研究现状，他发现人工智能领域并没有重大甚至显眼的研究成果，因而认为不必保留一个独立的“人工智能”学科。

1.2.4 专家系统技术，成功的关键因素（20世纪70年代早期~20世纪80年代中期）

20世纪70年代最重要的进展，也许就是人们意识到必须对智能机器的问题域进行充分限制。而在此之前，人工智能研究者一直认为，为了对通用的、人类惯用的解决问题的方法进行仿

8 真，必须创造出聪明的搜索算法和推理技术。一种通用的搜索机制可以依赖基本的推理步骤来发现完整的解决方案并可以使用问题域的弱知识。然而，当这种弱方法失败后，研究者们最终意识到，唯一出路是使用大量推理步骤来解决狭窄专业领域的典型问题。

DENDRAL 是当时的一个典型例子（Buchanan et al.，1969），其由斯坦福大学开发，用于化学分析。NASA 资助了这个项目，原因是当时美国要发射一个无人宇宙飞船到火星，需要根据由质谱仪提供的大量光谱数据，由此设计程序以分析火星表面土壤的分子结构。Edward Feigenbaum（曾是 Herbert Simon 的学生）、Bruce Buchanan（计算机科学家）、Joshua Lederberg（遗传学领域的诺贝尔奖获得者）组成团队来解决这一挑战性的问题。

这类问题的传统解决方法都依赖于枚举—测试技术：首先枚举出与质谱图一致的所有可能的分子结构，用每一个分子结构预测质谱，再根据实际的质谱进行检验。但是，由于分子结构会达到千百万个，即使分子数目适中，问题也会迅速复杂化，因此这种方法是行不通的。

更困难的是，没有科学的算法将质谱图匹配到对应的分子结构。不过，像 Lederberg 这样的分析化学家可以使用他们的技巧、经验和特长解决这个问题。通过找出质谱图中的高峰期所具有的明显的模式，可以大大降低可能的分子结构数目，再使用几个可行的方法进一步求解。所以，Feigenbaum 的任务就是在程序设计时将 Lederberg 的专业知识考虑进来，以使程序具有相当于人类专家的水平。这种程序后来被称为*专家系统*。要想理解、采纳 Lederberg 的知识，并用他的术语进行操作，Feigenbaum 需要学习化学和质谱分析的基本知识。当然，他不仅需要使用化学规则，也需要基于其经验和猜测，并使用他自己的经验法则之类的启发式方法。不久 Feigenbaum 就发现项目中的一个主要困难是如何从人类专家那里抽取知识并应用于计算机，他将这一问题称为“知识获取瓶颈”。为了更清楚地表达知识，Lederberg 也需要学习计算方面的基础知识。

Feigenbaum、Buchanan 和 Lederberg 组成团队，最终成功开发出了第一个基于知识的系统——DENDRAL。他们成功的关键在于将所有相关的理论知识从泛化形式映射到极为具体的规则（“烹饪食谱”）（Feigenbaum et al.，1971）。

DENDRAL 的巨大意义可以归纳为如下几点：

- DENDRAL 标志着人工智能领域重要的“范式转移”，即从通用的、知识稀疏型弱方法转
9 移到针对特定领域的知识密集型技术。
- DENDRAL 项目的目标是设计出能达到有经验的化学家水平的计算机程序。通过提取人类专家知识，并表达为高质量的具体规则（经验法则），DENDRAL 团队使用这种启发式策略证明了计算机在特定、可定义的问题领域能达到专家水平。
- DENDRAL 为专家系统开创了新的方法论，即*知识工程*，它包含了对专家知识从获取、分析到用规则表达等一系列技术。

DENDRAL 后来成为化学家们有用的分析工具，并在美国成为商业产品被投放到市场。

Feigenbaum 和来自斯坦福大学的人员接下来承担了医疗诊断方面的一个重大项目，这个项目名为 MYCIN，始于 1972 年，后来成为 Edward Shortliffe 的博士论题（Shortliffe，1976）。MYCIN 是基于规则的专家系统，它的任务是诊断传染性的血液病，它还以一种便捷、用户友好的方式为医生提供治疗建议。

MYCIN 与早期专家系统具有若干相同特点：

- MYCIN 的性能相当于人类专家水平，并高于初级医生的水平。
- MYCIN 包含了 450 条独立的 IF-THEN 形式的规则，这些知识是通过访问特定领域的大量专家获得的。
- 以规则形式表达的知识与推理机制明确地分开了。系统研发员也能够轻易地通过插入、删除规则来操纵知识。例如，斯坦福大学后来研发了一个独立于领域知识的 MYCIN 版本，

称为 EMYCIN（van Melle，1979；van Melle et al.，1981）。它除了传染性血液病知识之外，具有 MYCIN 的一切特征。EMYCIN 促进了诊断方面的各种应用。系统研发员只需将新知识以规则形式添加，就能实现新的应用。

与早期专家系统相比，MYCIN 也有一些新特点。例如，MYCIN 的规则反映了知识本身的不确定性，这里是指医学诊断知识。它在可利用的数据或医生所给的数据上测试规则的条件部分（IF 部分）。在适当时候，MYCIN 使用名为确信因子的不确定性运算推断是否满足条件。考虑不确定性的推理是这个系统的最重要的特征。

另一个被广泛关注的概率系统是 PROSPECTOR，这一专家系统由斯坦福研究院开发，用于矿产勘探（Duda et al.，1979）。这个项目始于 1974 年，在 1983 年结束，有 9 位专家为该系统提 10
供了专业知识。PROSPECTOR 将规则和语义网络合并到一个结构中，以表达专家知识，并使用了一千多条规则来表达大量领域知识。同时拥有一个复杂的支持包，其中包括知识获取系统。

PROSPECTOR 的工作原理是，首先让作为用户方的勘探地质学家输入待检矿床的特征，如地址环境、结构、矿物质类型等。之后，程序将这些特征与矿床模型比较，必要时让用户提供更多信息。最后，系统对待检矿床做出结论。这一系统也能够解释为了得出结论所用到的步骤。

在勘探地质学领域，重要决策常常是在由于知识不完整或模糊而导致不确定性的情况下做出的。为了处理这类知识，PROSPECTOR 使用论据贝叶斯规则在系统中传送不确定性。它的性能达到了专业地质学家的水平，并且在实践中得到了验证。在 1980 年，用它识别出了华盛顿州 Tolman 山附近的一个钼矿床。随后一个采矿公司对这个矿床开采时证实这个矿床价值一亿美元。真是难以想象专家系统能有如此价值。

上面所提到的几个专家系统现在已成为经典之作。20 世纪 70 年代后期，专家系统越来越多地被成功应用，这表明能够将人工智能技术从实验室成功地移植到商业领域。不过，这一时期大部分专家系统都是在功能强大的工作站上，采用特殊的人工智能语言开发的，例如 LISP、PROLOG 以及 OPS。昂贵的硬件和复杂的程序语言意味着专家系统的开发工作只能由个别院校的研究组做，像斯坦福大学、麻省理工学院、斯坦福研究院和卡内基 - 梅隆大学。到了 20 世纪 80 年代，个人电脑和易用的专家系统开发工具框架的出现，才使得普通的研究者和各个领域的工程师都有机会开发专家系统。

1986 年的一份综述报告汇总了大量成功应用于不同领域的专家系统，所应用的领域包括化学、电子、工程、地质、管理、医药、过程控制和军事科学，等等（Waterman，1986）。尽管如此，在 Waterman 发现的将近 200 个专家系统中，大部分还是应用在医学诊断领域。7 年后另一篇综述汇总了 2500 多个开发完毕的专家系统（Durkin，1994）。新兴的应用领域是商业和制造业，占了总应用的 60%。此时专家系统技术已经成熟。

专家系统在任何领域都是成功的关键吗？虽然有了大量的专家系统，并且在不同知识领域得以成功地应用，但高估这一技术的能力是不对的。在技术和社会学两个层面都存在复杂的困难，这些困难包括： 11

- 专家系统局限于非常狭窄的专业领域。例如，虽然 MYCIN 的任务是诊断传染性血液病，却没有人体生理学的知识。如果病人的病不止一个，就不能指望 MYCIN 了。事实上，当病人伴随其他疾病时，使用 MYCIN 开的治疗血液病的处方甚至可能是有害的。
- 由于局限于狭窄的领域，专家系统并不能如用户所愿的那样健壮灵活。并且，专家系统难以识别领域界限。当一项任务不同于传统问题时，专家系统可能在尝试解决时出乎意料地失败。
- 专家系统的解释能力有限。虽然能够显示解决方案中应用的一系列规则，却无法将累积的启发式知识与对问题领域的深层理解关联起来。

- 专家系统难以检验验证。至今都没有开发出通用的技术来分析专家系统的完整性和一致性。启发式规则以抽象形式表达知识，缺乏对领域的基本理解，这使得识别错误的、不完整或不一致的知识的工作非常困难。
- 专家系统几乎不具备从它们的经验中去学习的能力，尤其是第一代专家系统。专家系统的开发相对独立而且开发过程慢。解决一个中等难度的问题要花费 5 ~ 10 人年的时间（Waterman,1986）。而像 DENDRAL、MYCIN 或 PROSPECTOR 之类的复杂系统需要超过 30 人年的开发时间。如果专家系统的完善依赖于它的开发者进一步关注，那么这些巨大的投入将难以评估。

尽管面临这么多的困难，专家系统的研究还是有了很大突破，并在若干重要应用中证明了它的价值。

1.2.5 如何使机器学习，神经网络的重生（20 世纪 80 年代中期至今）

在 20 世纪 80 年代中期，研究者、工程师以及专家们意识到，购买一个推理系统或专家系统框架，再往里填充足够的规则，并不足以构造一个专家系统。应用专家系统技术的幻想破灭，对人工智能项目的资助也严重紧缩，甚至导致人们开始预测人工智能的“冬季”将要到来。人工智能研究者们决定重新审视神经网络。

到 20 世纪 60 年代后期，神经计算所需的基本理论和概念大部分都已经成型（Cowan, 1990）。但是，直到 20 世纪 80 年代中期才出现解决方案，滞后的主要原因在于技术层面：没有
12 个人电脑或功能强大的工作站，所以无法对人工神经网络建模和实验。还有心理学和资金方面的原因。例如，在 1969 年，Minsky 和 Papert 就用数学方法论证了单层感知器具有根本的计算局限性（Minsky and Papert, 1969），他们还认为即使复杂的多层感知器也不见得更好。这显然不会激励人去研究感知器，这种情况导致在 20 世纪 70 年代几乎所有的人工智能研究者都放弃了人工神经网络领域。

在 20 世纪 80 年代，由于需要像大脑那样进行信息处理，以及计算机技术的发展和神经科学的进步，神经网络领域快速复苏了。在几个前沿领域，科学家们在理论和设计上都做了重大贡献。Grossberg 创建了自组织的新理论（adaptive resonance theory，*自适应共振理论*），这为一种新的神经网络奠定了基础（Grossberg, 1980）。Hopfield 引入了具有反馈机制的神经网络，即 Hopfield 网络（Hopfield networks, Hopfield, 1982），在 20 世纪 80 年代引起了广泛关注。Kohonen 发表了一篇关于自组织映射的文章（Kohonen, 1982）。Barto、Sutton 和 Anderson 发表了关于*增强学习*（reinforcement learning）和在控制领域的应用的文章（Barto et al., 1983）。但真正的突破是在 1986 年，Rumelhart 和 McClelland 在《并行分布式处理：对认知微结构的探索》一文中，重新提出了*反向传播学习算法*（back- propagation learning algorithm）（Rumelhart and McClelland, 1986），这一算法由 Bryson 和 Ho 在 1969 年首次提出（Bryson and Ho, 1969）。Parker（Parker, 1987）和 LeCun（LeCun, 1988）在同一时间也提出了反向传播学习算法，从此这一算法成为用于训练多层感知器的最流行的技术。在 1988 年，Broomhead 和 Lowe 提出了使用径向基函数设计*多层前馈网络*的程序，这是多层感知器的替代（Broomhead and Lowe, 1988）。

从 McCulloch 和 Pitts 提出的早期模型，发展为根植于神经科学、心理学、数学和工程的交叉学科，人工神经网络经过了一个漫长的历程，并在理论和应用两方面还会继续发展。无论如何，对于 20 世纪 80 年代神经网络的重生，Hopfield 的文章（Hopfield, 1982）、Rumelhart 和 McClelland 的书（Rumelhart and McClelland, 1986）具有最重要、最具影响的意义。

1.2.6　进化计算，在尝试中学习（20 世纪 70 年代早期至今）

自然界的智能是进化的产物。因此，通过模拟生物进化，我们有希望找出将现有的系统推向高水平智能的方法。自然界是在尝试中学习的，并没有外界告诉生物系统如何适应具体环境，这是物竞天择的过程。适应性最强的物种更有机会繁衍，并将基因遗传给下一代。 13

人工智能中的进化方法是基于自然选择和遗传的计算模型。进化计算的工作流程包括模拟由个体组成的种群、评估个体性能、产生下一代种群，这一过程需要迭代若干次。

进化计算主要包括 3 个主要技术：遗传算法、进化策略以及遗传编程。

遗传算法是 John Holland 在 20 世纪 70 年代早期提出的（Holland，1975）。他设计了一个包括选择、交叉、变异 3 个遗传操作的算法，用于操纵人工“染色体”（表达为二进制字符串）。模式定理是遗传算法的坚实的理论基础（Holland，1975；Goldberg，1989）。

在 Holland 的遗传算法之前，早在 20 世纪 60 年代早期，两个来自柏林科技大学的学生 Ingo Rechenberg 和 Hans-Paul Schwefel 提出了一个新的名为进化策略（evolutionary strategies）的优化方法（Rechenberg，1965）。进化策略的具体任务是解决工程设计中的参数优化问题。Rechenberg 和 Schwefel 建议，像自然界变异那样对参数随机变化。事实上，可将进化策略方法视为工程师直觉的替代物。进化策略使用一种数值优化过程，类似于蒙特·卡洛搜索方法。

遗传算法和进化策略都可以解决广泛的问题。这两个方法为之前根本无法解决的高度复杂的非线性搜索和优化问题提供了健壮、可靠的解决方案（Holland，1995；Schwefel，1995）。

遗传编程代表了将学习式的遗传模型在程序设计中的应用，它的目的并不是推出问题的编码表示，而是推出用以解决问题的计算机代码。就是说，遗传编程的目的是产生解决问题的计算机程序。

20 世纪 90 年代 John Koza 的工作极大促进了人们对遗传编程的兴趣（Koza，1992；1994），他使用遗传算子来操纵表示 LISP 程序的符号码。遗传编程为计算机科学领域的主要问题提供了解决方案，使计算机不需通过精确编程就能解决问题。

遗传算法、进化策略和遗传编程标志着人工智能学科的快速成长，并且具有巨大的发展空间。

1.2.7　知识工程的新纪元，文字计算（20 世纪 80 年代后期至今）

与基于符号推理的系统相比，神经网络的技术提供了与现实世界更加自然的交互。神经网络能够学习、适应问题环境的变化，能在规则未知的情形下构建模式，并能处理模糊、不完整的信息。但是神经网络的解释功能差，像黑匣子一样工作。用现有的技术来训练神经网络很耗时，而且频繁的重新训练还会导致严重的困难。 14

尽管在一些特殊的环境，尤其是缺少知识的环境中，人工神经网络能比专家系统更好地解决问题，但这两个技术发展到现在已不再彼此竞争，而是互补。

传统的专家系统很适用于具有精确输入和逻辑输出的封闭式系统的应用。这类系统使用表达为规则形式的专家知识，必要时还可以与用户交互以构建特定的事实。这类系统的主要缺点在于人类专家并不总能用规则表达知识或逐条解释推理，从而会妨碍专家系统积累必需的知识，最终导致失败。为了克服这个缺陷，可使用神经计算从大数据集中抽取隐含的知识以获取专家系统的规则（Zahedi，1993；Medsker and Leibowitz，1994）。在传统的基于规则的专家系统中也可以使用人工神经网络来校正规则（Omlin and Giles，1996）。换句话说，当得到的知识不完整时，可以用神经网络精炼知识，当知识与给定的数据不一致时，可使用神经网络修改规则。

另一个非常重要的技术是*模糊逻辑*，用于处理模糊、不精确、不确定的知识和数据。传统的专家系统中，用于处理不精确数据的大部分方法都是基于概率论思想，如 MYCIN 就引入了确信因子，PROSPECTOR 则使用了贝叶斯规则来传送不确定性。不过，专家们并不总是使用概率值，

而会使用“常常、一般地、有时、偶尔、极少”之类的术语。模糊逻辑正是关于模糊值使用的方法，模糊值用于理解单词含义、人类推理、制定决策。模糊逻辑以能精确反映专家对复杂难题的理解的形式，对人类知识进行编码和应用，从而为打破传统专家系统的计算瓶颈提供了一个可行的方法。

模糊逻辑的核心在于语言变量这一概念。语言变量的值是文字，而不是数值。模糊系统使用 IF-THEN 规则来体现人类知识，这点与专家系统类似，只是模糊系统的规则是模糊的，例如：

IF speed is high THEN stopping_distance is long

IF speed is low THEN stopping_distance is short

模糊逻辑或*模糊集理论*是由加州大学伯克利分校电子工程系主任 Lotfi Zadeh 教授在 1965 年提出的（Zadeh，1965），它提供了一种采用文字进行计算的方法。然而，模糊集理论被技术团体接受的过程却很困难和漫长，部分原因在于具有争议的名字“模糊”，这个名字似乎太随意而难以被认真对待。模糊理论虽然在西方被忽视，却在东方的日本被接纳了。自 1987 年起，模糊理
15 论被成功地应用在日本人设计的洗碗机、洗衣机、空调、电视机、复印机和汽车上。

模糊产品的出现使得这一看似新颖而在 30 年前就被提出的技术受到极大关注，成百上千的书籍和学术论文相继诞生。一些经典著述包括：《Fuzzy Sets，Neural Networks and Soft Computing》（Yager and Zadeh，eds，1994）；《The Fuzzy Systems Handbook》（Cox，1999）；《Fuzzy Engineering》（Kosko，1997）；《Expert Systems and Fuzzy Systems》（Negoita，1985）；以及畅销书《Fuzzy Thinking》（Kosko，1993），这本书对模糊逻辑进行了普及性介绍。

模糊逻辑主要应用在控制工程领域。不过，模糊控制系统仅使用了模糊逻辑知识表达的一小部分功能。模糊逻辑在基于知识的系统和决策支持系统中的应用优势可归纳如下（Cox，1999；Pedrycz and Gomide，2007；Turban el al.，2010）：

- *提高了计算能力*。基于规则的模糊系统比传统专家系统运算快，需要的规则也少。模糊专家系统通过合并规则使系统功能更强。Lotfi Zadeh 认为用不了几年，多数专家系统都将使用模糊逻辑来解决高度非线性问题和计算困难的问题。
- *改善了认知模型*。模糊系统允许以能反映专家思考复杂问题的形式对知识编码。专家在思考复杂问题时，会用到不精确术语，例如高和低、快和慢、轻和重，也用使用术语常常和几乎不、通常和很少、频繁和偶尔。在建立传统的规则时，需要对术语定义严格的界限，这就把专业知识分解成了知识片段。当传统的专家系统面对高度复杂问题时，知识片段将导致系统性能低下。相反地，模糊专家系统通过对不精确信息建模，就能以更接近专家思考的方式捕捉专业知识，从而改善了系统的认知建模。
- *能够表达多个专家*。传统的专家系统建立在非常狭窄的、被明确定义的专业知识领域，系统性能完全依赖于能否正确选择专家。尽管一般的策略是只选定一个专家，但是当构造更复杂的专家系统或专业知识不能被较好地定义时，就需要多个专家。多个专家能够扩展领域，综合经验，并且避免了对顶级专家的依赖，顶级专家一般费用昂贵且难以联系到。不过，多个专家极难达成一致意见，他们的意见常常不同甚至有冲突。在经济和管理领域，当没有简单方案而又需考虑各种具有冲突的观点时尤其如此。模糊专家系统有助于表达意
16 见相左的多个专家的专业知识。

尽管模糊系统允许以更自然的方式表达专家知识，但是它们仍依赖来自专家提炼的规则，因而系统有聪明的，也有笨拙的。有些专家能够提供非常聪明的模糊规则，有些专家只会猜测或提供错误的规则。所以，所有的规则必须经过测试、调整，这一过程可能会漫长又乏味。例如，日立公司的工程师们用了好几年时间才测试、调整了仙台地铁系统上的 54 条模糊规则。

借助模糊逻辑开发工具，能够轻松地构造出一个简单的模糊系统，但之后很可能要花费几

天、几个星期甚至几个月来测试新规则并调整系统。怎样加快这个过程呢？或者，怎样自动产生好的模糊规则呢？

近年来，一些基于神经网络技术的方法已被用于在数值数据中寻找模糊规则。自适应或神经－模糊系统能够发现新的模糊规则，或者基于提供的数据来调整已有的规则。换言之，即是输入数据－输出规则，或者输入经验－输出规律。

那么知识工程将向什么方向发展呢？

专家系统、神经系统和模糊系统目前已经成熟，并被广泛应用于多个领域处理不同问题，主要在工程、医药、金融、商业和管理等领域。各个技术在处理人类知识的不确定性和模糊性上都各有方法，在知识工程领域都占有一席之地。这 3 个技术之间不再竞争，而是互相配合。在专家系统中融入模糊逻辑和神经计算，能够提高基于知识系统的自适应性、鲁棒性、容错能力和运算速度。此外，利用文字进行计算使得系统更加“人性化”。在构造智能系统时，使用现有理论而非提出新理论已成为惯例。应用目标也不再限于简单的小问题，而是实在的现实问题。

1.3　小结

我们处在知识革新的时代，在这个时代，一个国家的实力不是由军队的士兵数量来决定，而是由这个国家所拥有的知识来决定。科学、医药、工程和商业一方面推动着国家朝更高质量的生活发展，一方面也需要高素质的人。当前我们已开始使用智能机器从知识渊博的人身上获取经验知识，并让机器像人那样进行推理。

在第一台计算机发明之前，对智能机器的渴求仅仅是一个渺茫的梦。虽然早期的计算机按照特定的算法能够有效操作大型数据库，却不能对提供的信息进行推理，因而引发了计算机能否思考的问题。阿兰·图灵将计算机的智能行为定义为在认知任务中达到人类同等水平的能力。图灵测试为基于知识的系统提供了检验和验证的基础。 17

1956 年，在达特茅斯学院召开的暑期研讨会将 10 个对机器智能感兴趣的研究者聚到了一起，一门新学科——人工智能诞生了。

自从 20 世纪 50 年代早期，人工智能技术从个别研究者的好奇开始，发展为帮助人类决策的有价值的工具。从 20 世纪 60 年代关于人工智能的伟大幻想和期待，到 20 世纪 70 年代早期对人工智能的幻灭和资助削减；从 20 世纪 70 年代的第一代专家系统的开发，如 DENDRAL、MYCIN 和 PROSPECTOR，到 20 世纪八九十年代专家系统的成熟以及在不同领域的广泛应用；从 20 世纪 40 年代所提的简单的二进制神经元模型，到 20 世纪 80 年代人工神经网络的戏剧性复活；从 20 世纪 60 年代所提出的模糊集理论在西方被忽视，到 20 世纪 80 年代由日本提供的大量的“模糊型”消费品，以及 20 世纪 90 年代软计算和用文字计算被全世界接受；这些事件使我们看到了人工智能的历史性发展周期。

专家系统的发展创立了知识工程，它是一种构建智能系统的过程。如今知识工程不仅涉及专家系统，还涉及神经网络和模糊逻辑。知识工程仍然是一门艺术，而非工程，但人们已经尝试使用神经网络技术从数值数据中自动抽取规则。

表 1.1 总结了人工智能和知识工程发展历程中的关键事件，涵盖范围从 1943 年 McCulloch 和 Pitts 在人工智能上的初次尝试，到近年来的用文字计算的基于知识的系统中将专家系统、模糊逻辑和神经计算进行合并，从而各取所长。

本章的主要内容包括：

- 智能是指学习和理解问题、解决问题、制定决策的能力。
- 人工智能是使机器像人那样用智能做事的一门学科。
- 如果机器在某些认知任务中能达到人类同等水平，它就是智能机器。在建造智能机器时，

我们必须捕捉、组织和使用特定问题领域的人类专家知识。

- 必须严格限制智能机器的问题域，对此的确认标志着人工智能从泛化的、知识稀疏的弱方法向具体领域的知识密集型方法的重大“范式转移”。这一转移促成了专家系统的发展，从而使计算机程序的性能在狭窄问题领域达到人类专家水平。专家系统以特定规则的形式使用人类知识
18 和经验，主要特征表现在对知识和推理机制的明确划分。专家系统更能解释推理过程。

表 1.1　人工智能和知识工程发展历程中主要事件一览表

时　期	主 要 事 件
人工智能的诞生 (1943 ~ 1956 年)	McCulloch and Pitts, *A Logical Calculus of the Ideas Immanent in Nervous Activity*, 1943 Turing, *Computing Machinery and Intelligence*, 1950 电子数据集成器和计算器项目 (von Neumann) Shannon, *Programming a Computer for Playing Chess*, 1950 在达特茅斯学院举行的机器智能、人工神经网络和自动化理论暑期研讨会, 1956
人工智能的崛起 (1956 年 ~ 20 世纪 60 年代晚期)	*LISP* (McCarthy) 通用问题解决方案 (GPR) 项目 (Newell and Simon) Newell and Simon, *Human Problem Solving*, 1972 Minsky, *A Framework for Representing Knowledge*, 1975
人工智能的幻灭 (20 世纪 60 年代晚期 ~ 20 世纪 70 年代早期)	Cook, *The Complexity of Theorem Proving Procedures*, 1971 Karp, *Reducibility Among Combinatorial Problems*, 1972 Lighthill 报告, 1971
专家系统的发明 (20 世纪 70 年代早期 ~ 20 世纪 80 年代中期)	DENDRAL (Feigenbaum, Buchanan and Lederberg, 斯坦福大学) MYCIN (Feigenbaum and Shortliffe, 斯坦福大学) PROSPECTOR (斯坦福大学研究院) PROLOG ——一种逻辑编程语言 (Colmerauer, Roussel and Kowalski, France) EMYCIN (斯坦福大学) Waterman, *A Guide to Expert Systems*, 1986
人工神经网络的重生 (20 世纪 80 年代 中期至今) 19	Hopfield, *Neural Networks and Physical Systems with Emergent Collective Computational Abilities*, 1982 Kohonen, *Self-Organized Formation of Topologically Correct Feature Maps*, 1982 Rumelhart 和 McClelland, *Parallel Distributed Processing*, 1986 首届 IEEE 神经网络国际会议 Haykin, *Neural Networks*, 1994 神经网络, MATLAB 应用工具箱 (The MathWorks, Inc.)
进化计算 (20 世纪 70 年代早期至今)	Rechenberg, *Evolutionsstrategien - Optimierung Technischer Systeme Nach Prinzipien der Biologischen Information*, 1973 Holland, *Adaptation in Natural and Artificial Systems*, 1975 Koza, *Genetic Programming: On the Programming of the Computers by Means of Natural Selection*, 1992 Schwefel, *Evolution and Optimum Seeking*, 1995 Fogel, *Evolutionary Computation-Towards a New Philosophy of Machine Intelligence*, 1995
用文字计算 (20 世纪 80 年代 晚期至今)	Zadeh, *Fuzzy Sets*, 1965 Zadeh, *Fuzzy Algorithms*, 1969 Mamdani, *Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis*, 1977 Sugeno, *Fuzzy Theory*, 1983 日本的“模糊型”消费品 (洗碗机、洗衣机、空调、电视机、复印机) 仙台地铁系统 (日立公司, 日本), 1986 Negoita, *Expert Systems and Fuzzy Systems*, 1985 首届 IEEE 神经系统国际会议, 1992, Kosko, *Fuzzy Thinking*, 1993 Yager and Zadeh, *Fuzzy Sets, Neural Networks and Soft Computing*, 1994 Cox, *The Fuzzy Systems Handbook*, 1994 Zadeh, *Computing with Words-A Paradigm Shift*, 1996 模糊逻辑, MATLAB 工具箱 (The MathWorks, Inc.) 神经网络, MATLAB 工具箱 (The MathWorks, Inc.) 伯克利软计算倡议组织 (BISC) http: //www-bisc. cs. berkeley. edu

- 构建智能机器面临的主要问题是知识获取瓶颈，即从人类专家那里获取知识，这也是知识工程的主要问题。
- 专家在思考时会使用不精确的术语，如常常和几乎不、通常和很少、频繁和偶尔，也会使用语言变量，如高和低、快和慢、轻和重。模糊逻辑或模糊集理论为用文字进行计算提供了方法。模糊逻辑的重心是使用模糊值，以捕捉和理解文字含义、人类的推理和决策的制定，并为传统专家系统在计算上的负担问题提供了突破性的方法。 [20]
- 专家系统既不能通过经验学习、也不能通过经验自我完善，多由个人开发，后续发展也需要巨大的投入，即使构建中型专家系统也需要 5 ~ 10 人年。机器学习则大大加快了专家系统的开发进程，通过加入新规则或校正已有规则，也提高了知识的质量。
- 人工神经网络启发于生物学上的神经网络，基于历史事件进行学习，并能自动生成规则，由此避免了冗长、昂贵的知识获取过程，以及相应的验证、调整过程。
- 通过对专家系统、人工神经网络分别与模糊逻辑的集成，提高了基于知识的系统的自适应性、容错能力和运算速度。

复习题

1.1　何为智能？什么是机器的智能行为？
1.2　描述人工智能中的图灵测试，并从现代的角度证明它的有效性。
1.3　如何定义人工智能这门科学？人工智能何时诞生？
1.4　什么是*弱方法*？列举 20 世纪 70 年代早期 AI 幻灭的主要原因。
1.5　何为专家系统？弱方法和专家系统技术的主要区别是什么？
1.6　列举 DENDRAL、MYCIN 和 PROSPECTOR 这些早期专家系统的共同特征。
1.7　专家系统的局限性有哪些？
1.8　专家系统和人工神经网络之间的区别是什么？
1.9　人工神经网络在 20 世纪 80 年代复活的原因是什么？
1.10　模糊逻辑的建立前提是什么？模糊集理论是何时出现的？
1.11　在基于知识的系统中应用模糊逻辑的主要优势是什么？
1.12　合并专家系统、模糊逻辑和神经计算的系统会有哪些优势？ [21]

参考文献

Barto, A.G., Sutton, R.S. and Anderson C.W. (1983). Neurolike adaptive elements that can solve difficult learning control problems, *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13, 834–846.

Boden, M.A. (1977). *Artificial Intelligence and Natural Man*. Basic Books, New York.

Broomhead, D.S. and Lowe, D. (1988). Multivariable functional interpolation and adaptive networks, *Complex Systems*, 2, 321–355.

Bryson, A.E. and Ho, Y.-C. (1969). *Applied Optimal Control*. Blaisdell, New York.

Buchanan, B.G., Sutherland, G.L. and Feigenbaum, E.A. (1969). Heuristic DENDRAL: a program for generating explanatory hypotheses in organic chemistry, *Machine Intelligence 4*, B. Meltzer, D. Michie and M. Swann, eds, Edinburgh University Press, Edinburgh, Scotland, pp. 209–254.

Cook, S.A. (1971). The complexity of theorem proving procedures, *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, New York, pp. 151–158.

Cowan, J.D. (1990). Neural networks: the early days, *Advances in Neural Information Processing Systems 2*, D.S. Tourefzky, ed., Morgan Kaufman, San Mateo, CA, pp. 828–842.

Cox, E. (1999). *The Fuzzy Systems Handbook: A Practitioner's Guide to Building, Using, and Maintaining Fuzzy Systems*, 2nd edn. Academic Press, San Diego, CA.

Duda, R., Gaschnig, J. and Hart, P. (1979). Model design in the PROSPECTOR

consultant system for mineral exploration, *Expert Systems in the Microelectronic Age*, D. Michie, ed., Edinburgh University Press, Edinburgh, Scotland, pp. 153–167.

Durkin, J. (1994). *Expert Systems: Design and Development*. Prentice Hall, Englewood Cliffs, NJ.

Feigenbaum, E.A., Buchanan, B.G. and Lederberg, J. (1971). On generality and problem solving: a case study using the DENDRAL program, *Machine Intelligence 6*, B. Meltzer and D. Michie, eds, Edinburgh University Press, Edinburgh, Scotland, pp. 165–190.

Fogel, D.B. (1995). *Evolutionary Computation – Towards a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimisation and Machine Learning*. Addison-Wesley, Reading, MA.

Greenblatt, R.D., Eastlake, D.E. and Crocker, S.D. (1967). The Greenblatt Chess Program, *Proceedings of the Fall Joint Computer Conference*, pp. 801–810.

Grossberg, S. (1980). How does a brain build a cognitive code? *Psychological Review*, 87, 1–51.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

Holland, J.H. (1995). *Hidden Order: How Adaptation Builds Complexity*. Perseus Books, New York.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences of the USA*, 79, 2554–2558.

Karp, R.M. (1972). Reducibility among combinatorial problems, *Complexity of Computer Computations*, R.E. Miller and J.W. Thatcher, eds, Plenum, New York, pp. 85–103.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps, *Biological Cybernetics*, 43, 59–69.

Kosko, B. (1993). *Fuzzy Thinking: The New Science of Fuzzy Logic*. Hyperion, New York.

22

Kosko, B. (1997). *Fuzzy Engineering*. Prentice Hall, Upper Saddle River, NJ.

Koza, J.R. (1992). *Genetic Programming: On the Programming of the Computers by Means of Natural Selection*. MIT Press, Cambridge, MA.

Koza, J.R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA.

LeCun, Y. (1988). A theoretical framework for back-propagation, *Proceedings of the 1988 Connectionist Models Summer School*, D. Touretzky, G. Hilton and T. Sejnowski, eds, Morgan Kaufmann, San Mateo, CA, pp. 21–28.

Lighthill, J. (1973). Artificial intelligence: a general survey, *Artificial Intelligence: A Paper Symposium*. J. Lighthill, N.S. Sutherland, R.M. Needham, H.C. Longuest-Higgins and D. Michie, eds, Science Research Council of Great Britain, London.

McCarthy, J. (1958). Programs with common sense, *Proceedings of the Symposium on Mechanisation of Thought Processes*, vol. 1, London, pp. 77–84.

McCulloch, W.S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, 5, 115–137.

Medsker, L. and Leibowitz, J. (1994). *Design and Development of Expert Systems and Neural Computing*. Macmillan, New York.

Minsky, M.L. (1975). A framework for representing knowledge, *The Psychology of Computer Vision*, P. Winston, ed., McGraw-Hill, New York, pp. 211–277.

Minsky, M.L. and Papert, S.A. (1969). *Perceptrons*. MIT Press, Cambridge, MA.

Negoita, C.V. (1985). *Expert Systems and Fuzzy Systems*. Benjamin/Cummings, Menlo Park, CA.

Newell, A. and Simon, H.A. (1961). GPS, a program that simulates human thought, *Lernende Automatten*, H. Billing, ed., R. Oldenbourg, Munich, pp. 109–124.

Newell, A. and Simon, H.A. (1972). *Human Problem Solving*. Prentice Hall, Englewood Cliffs, NJ.

Omlin, C.W. and Giles, C.L. (1996). Rule revision with recurrent neural networks, *IEEE Transactions on Knowledge and Data Engineering*, 8(1), 183–188.

Parker, D.B. (1987). Optimal algorithms for adaptive networks: second order back propagation, second order direct propagation, and second order Hebbian learning, *Proceedings of the IEEE 1st International Conference on Neural Networks*, San Diego,

CA, vol. 2, pp. 593–600.
Pedrycz, W. and Gomide, F. (2007). *Fuzzy Systems Engineering: Toward Human-Centric Computing*. John Wiley, Hoboken, NJ.
Rechenberg, I. (1965). *Cybernetic Solution Path of an Experimental Problem*. Ministry of Aviation, Royal Aircraft Establishment, Library Translation No. 1122, August.
Rechenberg, I. (1973). *Evolutionsstrategien – Optimierung Technischer Systeme Nach Prinzipien der Biologischen Information*. Friedrich Frommann Verlag (Günther Holzboog K.G.), Stuttgart–Bad Cannstatt.
Rosenblatt, F. (1962). *Principles of Neurodynamics*. Spartan, Chicago.
Rumelhart, D.E. and McClelland, J.L., eds (1986). *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, 2 vols. MIT Press, Cambridge, MA.
Samuel, A.L. (1959). Some studies in machine learning using the game of checkers, *IBM Journal of Research and Development*, 3(3), 210–229.
Samuel, A.L. (1967). Some studies in machine learning using the game of checkers II – recent progress, *IBM Journal of Research and Development*, 11(6), 601–617.
Schwefel, H.-P. (1995). *Evolution and Optimum Seeking*. John Wiley, New York.
Shannon, C.E. (1950). Programming a computer for playing chess, *Philosophical Magazine*, 41(4), 256–275.
Shortliffe, E.H. (1976). *MYCIN: Computer-Based Medical Consultations*. Elsevier Press, New York.
Turban, E., Sharda, R. and Delen, D. (2010). *Decision Support and Business Intelligent Systems*, 9th edn. Prentice Hall, Englewood Cliffs, NJ.
Turing, A.M. (1950). Computing machinery and intelligence, *Mind*, 59, 433–460.
van Melle, W. (1979). A domain independent production-rule system for consultation programs, *Proceedings of the IJCAI 6*, pp. 923–925.
van Melle, W., Shortliffe, E.H. and Buchanan B.G. (1981). EMYCIN: a domain-independent system that aids in constructing knowledge-based consultation programs, *Machine Intelligence, Infotech State of the Art Report 9*, no. 3.
Waterman, D.A. (1986). *A Guide to Expert Systems*. Addison-Wesley, Reading, MA.
Yager, R.R. and Zadeh, L.A., eds (1994). *Fuzzy Sets, Neural Networks and Soft Computing*. Van Nostrand Reinhold, New York.
Zadeh, L. (1965). Fuzzy sets, *Information and Control*, 8(3), 338–353.
Zahedi, F. (1993). *Intelligent Systems for Business: Expert Systems with Neural Networks*. Wadsworth, Belmont, CA.

23

第2章
Artificial Intelligence, 3E

基于规则的专家系统

本章介绍构建知识系统的常用选择：基于规则的专家系统。

2.1 知识概述

在20世纪70年代，人们最终意识到，要用机器解决智能问题，就必须先知道方案。也就是说，人必须具备特定领域的知识，知道如何做。

知识是什么

知识是对于一个主题或一个领域在理论或实践上的理解，也是所有已知的总和，显然，知识就是力量。那些拥有知识的人称为专家，他们是所在组织中重要的人物。任何一个成功运营的公司都有若干一流的专家，没有他们公司就无法运营下去。

谁可称为专家

任何一个人如果在特定领域有深厚的知识（表现在事实和规则两方面）和强大的实践经验，都被认为是领域专家。领域的范围有所限制。例如，发电机专家也许对变压器仅有泛泛的了解，人寿保险专家或许对不动产保险政策的了解也是有限的。总之，专家能熟练地做别人做不了的事情。

专家如何思考

人类的思维活动是内在的、复杂的，用算法表达不了。好在多数专家能以规则的形式表达知识，从而解决问题。举一个简单的例子，假设你遇到一个外星人，他想过马路。你是否帮得了他？因为有多年过马路的经历，这一方面你是专家。所以你有能力就此教外星人。你会怎么做呢？

你可以告诉外星人，当交通灯变绿时，可以安全地过马路，当交通灯变红时，就必须停止。这是基本规则。可以把你的知识简单地表达为如下句子：

IF the 'traffic light' is green
THEN the action is go

IF the 'traffic light' is red
THEN the action is stop

这些用IF-THEN表达的句子称为产生式规则或规则。在人工智能学科中，规则是最常用的知识表达的方式，可以定义为IF-THEN语句，IF部分是给定的信息或事实，THEN部分则是相应的行为。规则用于描述如何解决问题，创建简单，也易于理解。

2.2 知识表达技术——规则

任何规则都包括两部分：IF部分和THEN部分。IF部分是前项（前提或条件），THEN部分是后项（结论或行为）。

规则的基本语法是：

IF ＜前项＞
THEN ＜后项＞

一般而言，规则可以有多个前项，这些前项用关键词AND（合取）、OR（析取）或者AND、OR混合使用。不过，最好在同一个规则中避免混合使用合取和析取。

```
IF      <前项 1>
AND     <前项 2>
          ⋮
AND     <前项 n>
THEN    <结论>

IF      <前项 1>
OR      <前项 2>
          ⋮
OR      <前项 n>
THEN    <结论>
```

26

规则的结论部分也可以由多个从句组成：

```
IF      <前项>
THEN    <结论 1>
        <结论 2>
          ⋮
        <结论 m>
```

规则的前项部分包括两部分：对象（语言对象）和值。以过马路为例，语言对象是“交通灯”，可以取值“绿”或“红”。对象和值用操作符连接。操作符识别对象并进行赋值。像 is、are、is not、are not 等操作符为语言对象赋予符号值。在专家系统中，也可以使用数学操作符定义数值型对象，并赋予数值。例如：

```
IF      'age of the customer' < 18
AND     'cash withdrawal' > 1000
THEN    'signature of the parent' is required
```

规则的后项部分与前项部分类似，也通过操作符连接对象和值。操作符为语言对象赋值。以过马路为例，如果“交通灯”的值是“绿”，第一条规则会将语言对象的行为赋值为“前进”。后项中也可以用数值型对象和简单的算术表达式：

```
IF      'taxable income' > 16283
THEN    'Medicare levy' = 'taxable income' * 1.5 / 100
```

关系、建议、指示、策略和启发式方法也可以用规则表达：

关系

```
IF      the 'fuel tank' is empty
THEN    the car is dead
```

建议

```
IF      the season is autumn
AND     the sky is cloudy
AND     the forecast is drizzle
THEN    the advice is 'take an umbrella'
```

指示

```
IF      the car is dead
AND     the 'fuel tank' is empty
THEN    the action is 'refuel the car'
```

27

策略

```
IF      the car is dead
THEN    the action is 'check the fuel tank';
        step1 is complete

IF      step1 is complete
AND     the 'fuel tank' is full
THEN    the action is 'check the battery';
        step2 is complete
```

- 启发式方法

```
IF    the spill is liquid
AND   the spill pH < 6
AND   the spill smell is vinegar
THEN  the spill material is 'acetic acid'
```

2.3 专家系统研发团队的主要参与者

当人类专家提供了知识后，我们就能够把知识输入计算机，期待着在特定领域的计算机能够做一个智能的助手，或者像专家那样解决问题，并希望计算机能够整合新知识，并以易读、易懂的方式表达知识，或用自然语言而非人工编程语言来处理简单的句子。最后，我们还希望计算机解释得出一个结论的原因。换言之，我们需要构建专家系统——能在狭窄问题领域具有专家水平的计算机程序。

基于规则的系统是最常见的专家系统。已有大量的系统被成功应用于商业、工程、医药、地理、电力系统以及采矿，也有很多公司生产和销售用于开发基于规则的专家系统的软件，即个人电脑上的专家系统框架。

为开发基于规则的系统的专家系统框架越来越普遍，其主要优势在于系统开发员可以把注意力放在知识本身，而不是学习一门编程语言。

什么是专家系统框架

专家系统框架（expert system shell）是不含知识的专家系统。用户的全部任务就是以规则形式加入知识，并提供解决问题的相关数据。

现在看一下构建专家系统需要哪些人员，这些人员需要具备什么能力？

一般地，专家系统开发团队需要 5 个成员：领域专家、知识工程师、程序员、项目经理和终端用户。专家系统能否开发成功完全取决于成员间能否很好地合作。图 2.1 总结了开发团队中成员的基本关系。

| 28 |

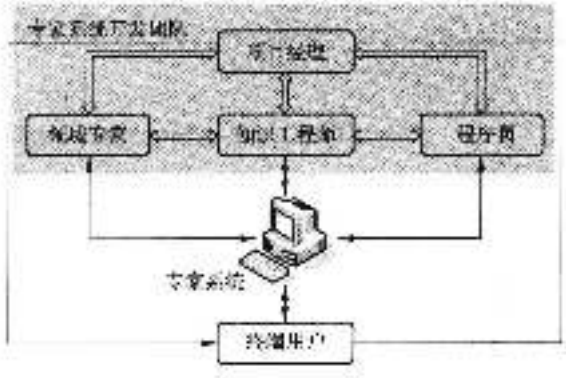


图 2.1 专家系统开发团队的主要成员

领域专家是在特定区域或领域具有渊博知识和丰富经验的人。他们在特定领域具有最好的经验，这些经验会被输入到专家系统中。所以，专家必须能够传达知识，乐意加入开发团队，并在项目上有足够的时间。在专家系统开发团队中，领域专家是最重要的成员。

知识工程师是善于设计、构建、测试专家系统的人，负责为系统选择合适的任务，并就如何解决特定问题和领域专家进行互动。通过互动，知识工程师会明确专家如何解决事实和规则，以及怎样在专家系统中表达。继而选择特定的开发软件或专家系统框架，或选择编程语言以将知识编码（有时知识工程师亲自编码知识）。之后，知识工程师负责测试、修改专家系统，并将系统集成到工作平台。可以看出，知识工程师要负责从系统的最初设计到最终运行的整个流程。项目完成后，也许还要参与系统的维护。

程序员负责编程，用计算机能接受的术语描述领域知识。他不仅需要精通人工智能语言的符号编程，像 LISP、PROLOG 和 OPS5 等，也要具备不同类型专家系统框架的应用经验。此外，

| 29 |

程序员还要会常用编程语言，如 C、Pascal、FORTRAN 和 Basic。如果使用专家系统框架进行开发，知识工程师就能够将知识编码到系统中，就不需要程序员了。当不能使用专家系统框架时，

就需要程序员设计知识和数据表达结构（知识库和数据库）、控制结构（推理引擎），以及对话结构（用户界面）。程序员可能也要参与测试专家系统。

项目经理是开发团队的领导，负责项目的进程，确保按计划进行，并与专家、知识工程师、程序员、终端用户进行互动。

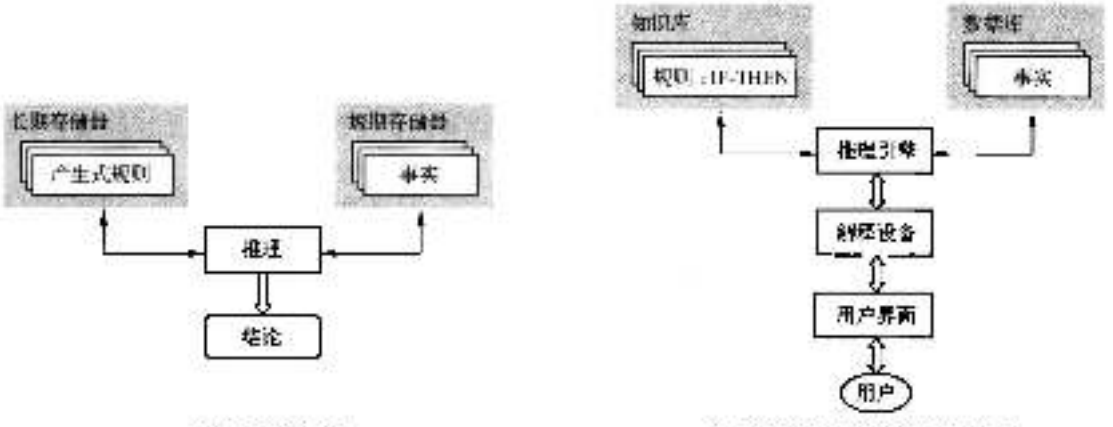
终端用户，也称为用户，是使用专家系统的人，可能是一个判定火星表面上境的化学结构的分析化学家（Feigenbaum et al., 1971），或诊断传染性血液病的初级医生（Shortliffe, 1976），或勘探新矿床的勘探地质学家（Duda et al., 1979），或紧急情况下需要建议的电力系统操作员（Negnevitsky, 2008）。这些终端用户各有所需，所以系统最终能否被认可依赖于用户的满意度。用户的满意度不仅包括系统功能，还包括操作是否舒适。所以系统中用户界面的设计至关重要。因此，终端用户的建议至关重要。

当聚齐5个成员时，就可以开发系统了。不过，现在的许多专家系统是在个人电脑上使用专家系统框架开发的，不需要程序员，或许也不需要知识工程师。对于微型专家系统，项目经理、知识工程师、程序员甚至专家可能是同一个人。在开发大型专家系统时，就需要上述所有的成员了。

2.4 基于规则的专家系统的结构

在20世纪70年代早期，来自卡内基－梅隆大学的Newell和Simon提出了产生式系统模型，为现代的基于规则的专家系统奠定了基础（Newell and Simon, 1972）。产生式模型的思路是，对于一个用相关信息表达了的指定问题，人类运用知识（用产生式规则表达）能够解决。产生式规则存储在长期存储器中，问题相关的信息或事实存储在短期存储器中。图2.2是产生式系统模型和基于规则的专家系统的基本结构。

30



a) 产生式系统模型　　b) 基于规则的专家系统的基本结构

图2.2 产生式系统模型和基于规则的专家系统的基本结构

基于规则的专家系统由5个部分组成：知识库、数据库、推理引擎、解释设备和用户界面。

知识库包含解决问题相关的领域知识。在基于规则的专家系统中，知识用一组规则来表达。每一条规则表达一个关系、建议、指示、策略或启发式方法，具有IF（条件）THEN（行为）结构。当规则的条件被满足时，触发规则，继而执行行为。

数据库包含一组事实，用于匹配存储在知识库中的IF（条件）部分。

推理引擎执行推理，专家系统由此找到解决方案。推理引擎链接知识库中的规则和数据库中的事实。

31

用户使用解释设备查看专家系统怎样得出解决方案的过程，以及为什么需要特定事实。专家系统必须能够解释推理并证明所给的建议、分析或结论。

用户界面是实现用户（查询问题解决方案）和专家系统之间交流的途径，这一途径必须有实际意义并尽可能地友好。

这5个部分对于任何基于规则的专家系统来说都是不可或缺的，它们共同构成了专家系统的核心。除此之外也会有其他附加组件。

外部接口允许专家系统融合外部数据文件和以常用语言编写的程序，如C，Pascal，FORTRAN和Basic。图2.3是基于规则的专家系统的完整结构。

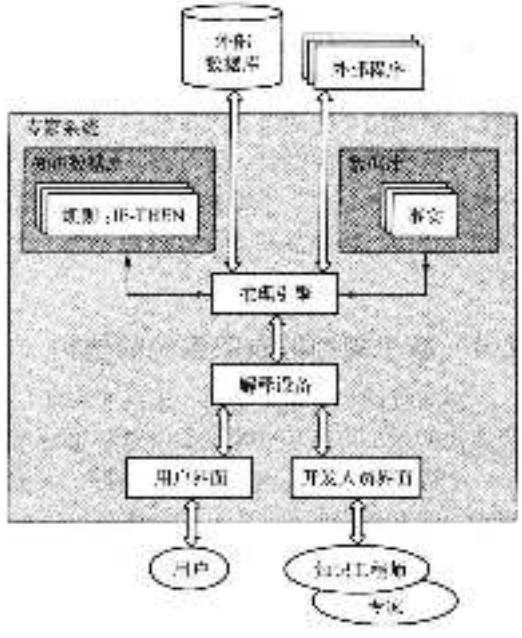


图2.3 基于规则的专家系统的完整结构

开发者接口一般包括知识库编辑器、调试工具，以及输入/输出设备。

任何专家系统框架都会提供简单的文本编辑器，用于输入和调整规则、检查格式和拼写。许
32 多专家系统也包括记录设备，用于监测知识工程师或专家所做的修改。如果规则被改动，编辑器将自动存储改动日期和做改动的人，以便后面参考。当有多个知识工程师和专家都有权进入知识库并做修改时，记录设备尤其重要。

调试工具一般包含跟踪设备和断点包。跟踪设备提供程序执行过程中被触发的规则列表。使用断点包则能提前告诉系统哪里该中断，以备知识工程师或专家即时查看数据库中的当前值。

多数专家系统还提供输入/输出设备，例如连接时知识获取器，以使运行中的专家系统获取数据库之外的必需信息。当知识工程师或专家输入所需信息后，系统接着往下运行。

总之，开发者接口、知识获取设备使得领域专家能够直接将知识输入专家系统，以减少打扰知识工程师的次数。

2.5 专家系统的基本特征

专家系统用于在狭窄的特定领域以专家级的水平解决问题。因此，专家系统的最主要特征就在于高质量的性能。无论系统能多快地解决问题，如果结果错误，用户是不会满意的。另一方面，解决速度也很重要。最精确的决策或诊断如果来的太慢，也许也会无用，例如在病人死亡或核反应堆爆炸等紧急情况时。专家使用经验法则或启发式方法，通过自己的经验和对问题的理解来寻找问题的解决捷径。专家系统也应该像人类专家那样，运用启发式方法来引导推理并降低解决方案的搜索空间。

拥有解释功能是专家系统独具的特征，专家系统通过这一功能回顾推理过程，并对其决策进行解释。所谓解释是跟踪整个解决过程中被触发的规则。这当然是一个简单的说法。事实上，由于系统不具备对底领域的最起码的理解，真正的、像人那样的解释还不现实。尽管一系列被触发的规则并不能证明一个结论，我们仍然可以为每一条规则，至少为知识库中的高级规则，关联上用文本表达的恰当的基本规律。这也许是目前的解释功能所能达到的程度。不过，解释功能对
33 有些专家系统来说不是必需的。例如，为专家设计的科学系统不需要提供人就解释，因为结论对

专家而言是自解释的，有简单的规则－跟踪机制就足够了。另一方面，用于决策支持的专家系统则需要完整的、有意义的解释，毕竟错误结论的代价很高。

在解决问题时，专家系统使用符号推理。符号被用来表示不同类型的知识，如事实、概念和规则。构建专家系统的目的是处理知识，因此能够轻易地处理定性数据，这点不同于处理数值数据的传统程序。

传统程序通过算法处理数据，换言之，它是定义好的逐步的操作序列。一个算法总按同一顺序执行同样的操作，每次都得到精确解。传统程序不会出错，只是程序员有时出错。专家系统则与传统程序不同，因为它并不按照特定的步骤序列执行，允许不精确的推理，也能够处理不完整、不确定的模糊数据。

专家系统会出错吗

再聪明的专家也是人，也会出错。对于要达到人类专家水平的专家系统，也应该容许出错。尽管我们知道专家的判断有时错误，但是我们仍然信赖专家。同样，尽管在多数情况下专家系统所得的结论都是可信赖的，有时依然会出错，我们要认识到这一点。

这意味着传统程序优于专家系统吗

理论上，传统程序提供的答案总是正确的。但要记住传统程序仅能处理数据完整、准确的问题。当数据不完整或包含错误时，传统程序要么无法应对，要么给出错误的答案。相比之下，专家系统能够识别出不完整或者模糊的信息，在这种情况下仍能给出合理的结论。

专家系统有别于传统程序的另一个特征是，*知识与处理过程相分离*（知识库和推理引擎是两部分）。而传统程序中知识和处理知识的控制结构是混合在一起的，这使得我们难以理解和修改程序代码，毕竟对代码的修改既影响知识，又影响知识的处理。在专家系统中，知识与处理机制被明确地划分开，在构建和维护专家系统时就容易得多。如果使用专家系统框架，知识工程师或者专家只需往知识库中添加知识即可。每一条新规则会带来新知识，专家系统也随之更聪明。通过改变、删减规则，能轻易地修改系统。

34

专家系统的上述特征使专家系统有别于传统程序和人类专家。表2.1是三者之间的对比。

表2.1 专家系统、传统程序及人类专家的对比

人类专家	专家系统	传统程序
在狭窄领域，以经验法则或启发式方法的知识形式来解决问题	在狭窄领域，以规则形式表达知识，使用符号推理解决问题	处理数据，使用由一系列操作组成的算法，来解决一般的数值问题
在人脑中，知识以可编译的形式存在	提供知识与处理过程明确分离的机制	知识与处理知识的控制结构没有分离
能够解释推理过程并提供细节	在解决问题过程中跟踪被触发的规则，能够解释怎样得出的结论，以及为什么需要特定数据的原因	对怎样得出结论和需要数据的原因都不做解释
使用不精确的推理，能处理不完整、不确定和模糊的信息	允许不精确的推理，能处理不完整、不确定和模糊的数据	仅能在数据完整、确定的情况下处理问题
当信息不完整或模糊时，可能会出错	当数据不完整或模糊时，可能会出错	当数据不完整或模糊时，要么无能为力，要么出错
通过多年的学习和实践训练，可以提高解决问题的能力。不过这一过程缓慢、低效、代价高	通过向知识库加入新规则或调整老规则，可以提高解决问题的能力。当需要新知识时，很容易实现调整	通过改变程序代码可以提高解决问题的能力。由于对代码的改变将同时影响知识和处理过程，改变起来很困难

2.6 前向链接和后向链接推理技术

在基于规则的专家系统中，领域知识用一组IF-THEN产生式规则来表示，数据用当时情境

35 下的一组事实来表示。推理引擎将存储在知识库中的规则与数据库中的事实相匹配，如果一条规则的 IF（条件）部分与事实匹配，即触发该规则，执行 THEN（行为）部分。被触发的规则可能由于添加新事实而改动事实集合，如图 2.4 所示。数据库和知识库中的字母表示状态或概念。

通过匹配规则的 IF 部分和事实，形成推理链。推理链表明了专家系统如何运用规则得出结论。下面通过一个简单的例子来解释链式推理技术。

假设数据库最初包含 5 个事实：A、B、C、D 和 E，知识库最初包含 3 条规则：

Rule 1: IF Y is true
AND D is true
THEN Z is true

Rule 2: IF X is true
AND B is true
AND E is true
THEN Y is true

Rule 3: IF A is true
THEN X is true

图 2.5 中的推理链表明了专家系统运用规则最终推出事实 Z 的过程。规则 3 最早被激活，由事实 A 推出了新事实 X。之后规则 2 被激活，由初始事实 B 和 E，加上事实 X，共同推出事实 Y。最后，规则 1 被激活，由初始事实 D 和刚确定的事实 Y 推出事实 Z。

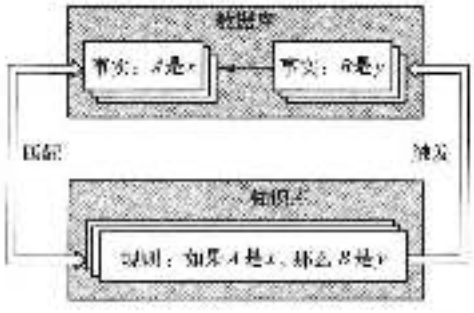


图 2.4 推理引擎中的匹配－触发循环过程

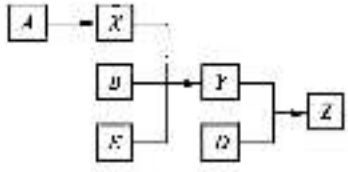


图 2.5 推理链的例子

36 专家系统能够用推理链来解释怎样得出的结论，这是解释设备的必要部分。

推理引擎要决定何时激活哪条规则。选择规则时，有两个主要方法：前向链接和后向链接（Waterman and Hayes-Roth, 1978）。

2.6.1 前向链接

上述例子正是运用了前向链接。现在来具体地分析这一技术。我们先将规则重写为如下形式：

Rule 1: $Y \& D \rightarrow Z$

Rule 2: $X \& B \& E \rightarrow Y$

Rule 3: $A \rightarrow X$

箭头表明规则的 IF 部分和 THEN 部分。另外，再添加两条规则：

Rule 4: $C \rightarrow L$

Rule 5: $L \& M \rightarrow N$

图 2.6 显示了这一组规则按前向链接的运作流程。

前项链接是数据驱动（data-driven）的推理技术。从已知数据开始展开推理。每一次只执行顶端的一条规则。当有规则被触发时，就有新事实加入数据库。任何规则只能被执行一次。当没

有规则可触发时，匹配－触发循环终止。

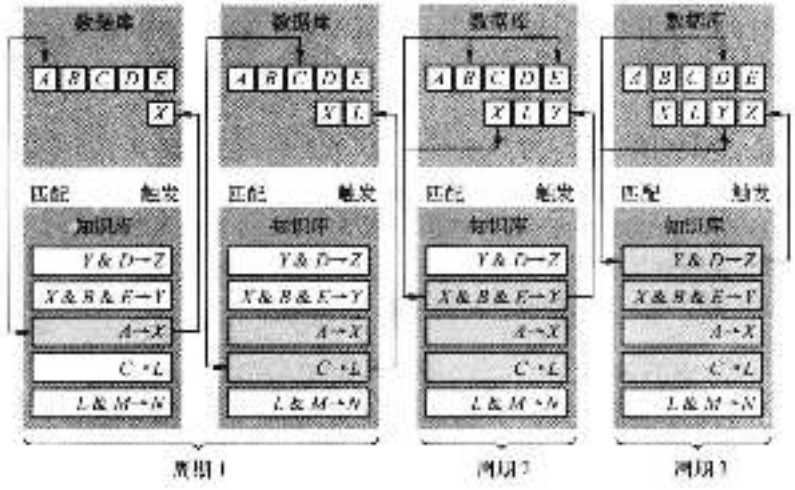


图2.6 前向链接

第一轮中，只有两个规则，规则3：$A \rightarrow X$ 和规则4：$C \rightarrow L$，与数据库中的事实匹配。规则3：$A \rightarrow X$ 处于顶端，就先被触发。规则3的IF部分与数据库中的事实 A 相匹配，执行THEN部分后，将新事实 X 加入数据库。之后，规则4：$C \rightarrow L$ 被触发，新事实 L 加入数据库。

第二轮中，由于数据库中已有事实 B、E 和 X，规则2：$X \& B \& E \rightarrow Y$ 最先被触发，产生事实 Y，并被放入数据库。这导致第三轮中规则1：$Y \& D \rightarrow Z$ 被执行，生成新事实 Z。至此，匹配－激活循环终止。由于规则5：$L \& M \rightarrow N$ 不匹配所有数据库中的事实，规则5无法被触发。

前向链接是搜集信息并推出信息的技术。不过，在前向链接中，许多被触发的规则也许与问题目标无关。在上面的例子中，目标是推出事实 Z。我们仅有5条规则，其中4条规则被触发。规则4：$C \rightarrow L$ 虽与事实 Z 无关，也同样被触发了。一个真正的基于规则的专家系统可能有成百上千条规则，许多规则被触发后虽然能推出有效的新事实，但可能与目标无关。所以，如果只需推出一个特定的事实，前向链接推导技术或许效率很低。

这种情况下，后向链接技术就比较合适。

2.6.2 后向链接

后向链接是目标驱动的推理技术。在后向链接中，专家系统有目标（一个假设的答案），推理引擎的任务是找出证明目标的论据。首先，在知识库中搜寻含有目标的规则，即THEN部分包含的目标规则。如果找到这种规则，在数据库中也有匹配的事实，就触发规则并证明目标。不过这种情况很少见。所以，推理引擎就暂不考虑这类规则（将规则压栈），要建立新目标，即子目标，以证明压栈规则的IF部分。接下来，再次查找知识库中能证明子目标的规则。推理引擎不断将规则压栈，直到知识库中的所有规则都不能证明子目标。

图2.7以上述例子为例，显示了后向链接的工作原理。

在第1个周期，推理引擎尝试推出事实 Z。通过查找知识库来寻找THEN部分包含事实 Z 的规则。推理引擎找到了规则1：$Y \& D \rightarrow Z$，将它压栈。规则1的IF部分包含事实 Y 和 D，所以下面是确立事实 Y 和 D。

在第2个周期，推理引擎安装子目标：事实 Y，并试图确立它。通过检查知识库，发现其中

没有事实 Y，因此需要寻找 THEN 部分包含事实 Y 的规则。推理引擎发现规则 2：$X \& B \& E \rightarrow Y$ 满足条件，将它压栈。规则 2 的 IF 部分包括事实 X、B 和 E，接下来同样需要建立这些事实。

在第 3 个周期，推理引擎安装新的子目标：证实事实 X。推理引擎通过查找数据库，发现不包含 A，就去查找能推出 X 的规则。找到了规则 3：$A \rightarrow X$，将它压栈。接下来需要证实事实 A。

在第 4 个周期，推理引擎发现 A 在数据库中。规则 3：$A \rightarrow X$ 被触发，推出了新事实 X。

在第 5 个周期，推理引擎要证实子目标 Y。再次试图执行规则 2：$X \& B \& E \rightarrow Y$。由于事实 X、B 和 E 都在数据库中，规则 2 被触发，推出了新事实 Y，将 Y 加入数据库。

在第 6 个周期，系统转向规则 1：$Y \& D \rightarrow Z$，尝试证实初始目标 Z。由于规则 1 的 IF 部分与数据库中的事实相匹配，规则 1 被执行，初始目标得以证实。

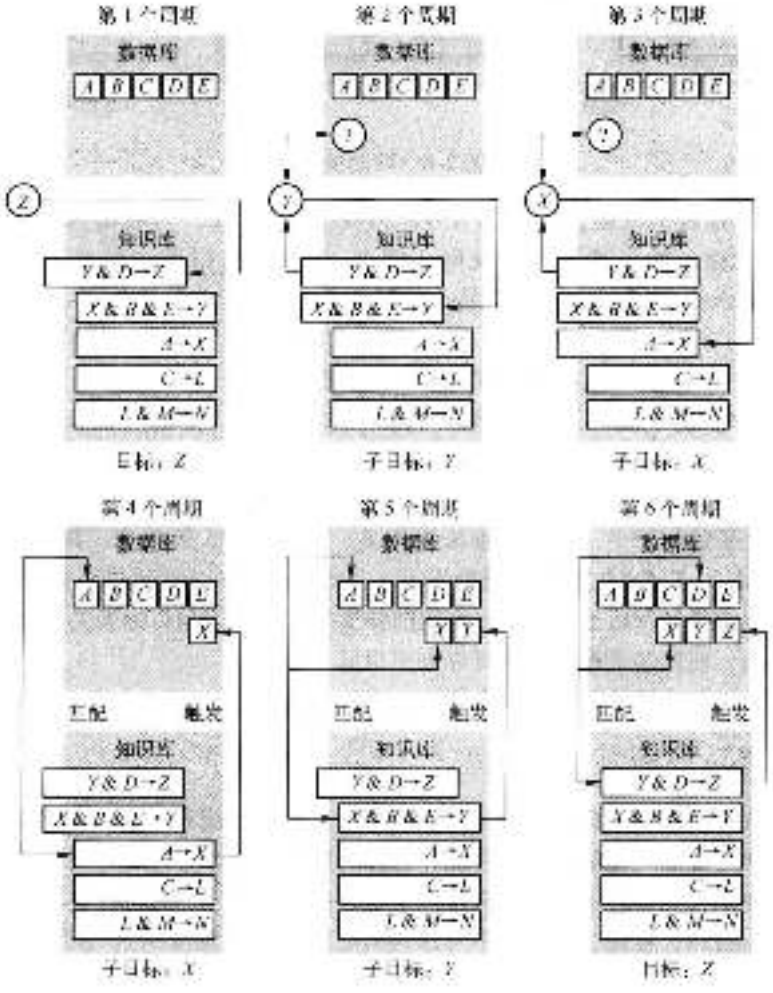


图 2.7 后向链接

现在来对比图 2.6 和图 2.7，可以看出，前向链接中触发了 4 条规则，而后向链接中只触发了 3 条规则。这个简单的例子说明了当需要证实一个特定事实时（例子中是 Z），后向链接更为有效。前向链接中，在推理过程开始前就确定了数据，不需用户额外输入。在后向链接中，确立目标后，只需要那些支持推理的数据，有时也需要用户输入一些数

据库中没有的事实。

如何选择前向链接和后向链接

答案是分析领域专家是如何解决问题的。如果专家事先需要搜集信息，无论结论是什么都试着去推理，这时应选择前向链接推理技术。如果专家从一个假设的结论触发，尝试着找出支持结论的论据，则应选择后向链接推理技术。

前向链接对于设计能够分析和解释的专家系统来说，是理所当然的方法。例如 DENDRAL，这款专家系统基于大量质谱数据来确定未知土壤的分子结构（Feigenbaum et al.，1971），就使用了前向链接技术。多数使用后向链接的专家系统多用于诊断性工作。例如，诊断传染性血液病的医用专家系统 MYCIN（Shortliffe，1976），就使用了后向链接技术。

能否将前项、后向链接相结合

许多专家系统框架同时使用了前项、后向链接推理技术，所以知识工程师就不必二选一。不过，基本的推理机制一般采用后向链接。只有当确立新事实时，才使用前向链接以最大程度地利用新数据。

39
|
40

2.7 MEDIA ADVISOR：基于规则的专家系统实例

接下来通过一个简单的基于规则的专家系统进一步说明上述概念。我们选择专家系统框架 Leonardo 来构建一个名为 MEDIA ADVISOR 决策支持系统，用于为实习生推荐平台，该平台根据实习生的工作制订培训计划。例如，如果实习生是一个负责维护水压系统的机械技术员，车间就是合适的平台，在那里可以学习水压组件的工作原理、如何检测水压故障，以及如何对水压系统做简单修复。如果实习生是评估投保单的员工，培训计划中就需包括特定相关任务的讲座，以及实习生评估实际申请的教程。对于复杂任务，实习生可能会出错，培训计划就应包括对实习生绩效的反馈。

知识库

```
/* MEDIA ADVISOR: a demonstration rule-based expert system

Rule: 1
if      the environment is papers
or      the environment is manuals
or      the environment is documents
or      the environment is textbooks
then    the stimulus_situation is verbal

Rule: 2
if      the environment is pictures
or      the environment is illustrations
or      the environment is photographs
or      the environment is diagrams
then    the stimulus_situation is visual

Rule: 3
if      the environment is machines
or      the environment is buildings
or      the environment is tools
then    the stimulus_situation is 'physical object'

Rule: 4
if      the environment is numbers
or      the environment is formulas
or      the environment is 'computer programs'
then    the stimulus_situation is symbolic
```

41

```
Rule: 5
if      the job is lecturing
or      the job is advising
or      the job is counselling
then    the stimulus_response is oral

Rule: 6
if      the job is building
or      the job is repairing
or      the job is troubleshooting
then    the stimulus_response is 'hands-on'

Rule: 7
if      the job is writing
or      the job is typing
or      the job is drawing
then    the stimulus_response is documented

Rule: 8
if      the job is evaluating
or      the job is reasoning
or      the job is investigating
then    the stimulus_response is analytical

Rule: 9
if      the stimulus_situation is 'physical object'
and     the stimulus_response is 'hands-on'
and     feedback is required
then    medium is workshop

Rule: 10
if      the stimulus_situation is symbolic
and     the stimulus_response is analytical
and     feedback is required
then    medium is 'lecture - tutorial'

Rule: 11
if      the stimulus_situation is visual
and     the stimulus_response is documented
and     feedback is not required
then    medium is videocassette

Rule: 12
if      the stimulus_situation is visual
and     the stimulus_response is oral
and     feedback is required
then    medium is 'lecture - tutorial'

Rule: 13
if      the stimulus_situation is verbal
and     the stimulus_response is analytical
and     feedback is required
then    medium is 'lecture - tutorial'

Rule: 14
if      the stimulus_situation is verbal
and     the stimulus_response is oral
and     feedback is required
then    medium is 'role-play exercises'

/* The SEEK directive sets up the goal of the rule set

seek medium
```

42

对象

MEDIA ADVISOR 使用6个语言对象：*environment*、*stimulus_situation*、*job*、*stimulus_response*、*feedback* 和 *medium*。各个对象都可以在各自值域内取一个值（例如，*environment* 可取的值有

papers、*manuals*、*documents*、*textbooks*、*pictures*、*illustrations*、*photographs*、*diagrams*、*machines*、*buildings*、*tools*、*numbers*、*formulas*、*computer programs*)。对象和它的值构成事实(例如,如果 *environment* is *machines* 和那么 *job* is *repairing*)。所有的事实都存在数据库中。

表 2.2 MEDIA ADVISOR 语言对象及其可取值

对 象	可 取 值	对 象	可 取 值
environment	papers manuals documents textbooks pictures illustrations photographs diagrams machines buildings tools numbers formulas computer programs	job	lecturing advising counselling building repairing troubleshooting writing typing drawing evaluating reasoning investigating
		stimulus_response	oral hands-on documented analytical
stimulus_situation	verbal visual physical object symbolic	feedback	required not required

43

选项

基于规则的专家系统的最终目标是根据数据找出问题的解决方案。在 MEDIA ADVISOR 中,解决方案是从 4 个选项中选择一个平台。

```
medium is workshop
medium is 'lecture - tutorial'
medium is videocassette
medium is 'role-play exercises'
```

对话

在如下的对话中,系统要求用户输入所需数据以解决问题(问题有环境、工作和反馈三方面)。根据用户提供的答案(如下面箭头所示),专家系统应用规则来推出 *stimulus_situation* is *physical object*,以及 *the stimulus_response* is *hands-on* 等结论,再根据规则 9 选择平台。

What sort of environment is a trainee dealing with on the job?
⇒ **machines**

```
Rule: 3
if    the environment is machines
or    the environment is buildings
or    the environment is tools
then  the stimulus_situation is 'physical object'
```

In what way is a trainee expected to act or respond on the job?
⇒ **repairing**

```
Rule: 6
if    the job is building
or    the job is repairing
or    the job is troubleshooting
then  the stimulus_response is 'hands-on'
```

Is feedback on the trainee's progress required during training?
⇒ **required**

Rule: 9
if the stimulus_situation is 'physical object'
and the stimulus_response is 'hands-on'
and feedback is required
44 then medium is workshop

推理技术

Leonardo 系统中标准的推理技术是后向链接技术，偶尔使用前向链接，它在处理可获取的信息时最有效。Leonardo 用户也能关掉后向链接或前向链接，这样便能分别研究这两种技术。

前向链接是数据驱动的推理技术，需要预先提供数据。假设：

the environment is **machines**
'environment' instantiated by user input to **'machines'**

the job is **repairing**
'job' instantiated by user input to **'repairing'**

feedback is **required**
'feedback' instantiated by user input to **'required'**

将会发生下述过程：

Rule: 3 fires **'stimulus_situation'** instantiated by Rule: 3 to **'physical object'**
Rule: 6 fires **'stimulus_response'** instantiated by Rule: 6 to **'hands-on'**
Rule: 9 fires **'medium'** instantiated by Rule: 9 to **'workshop'**
No rules fire stop

后向链接是目标驱动的推理技术，需要预先假定结果（目标）。假设目标是“‘medium’ is ‘workshop’”，确立以下目标：

Pass 1	
Trying Rule: 9	Need to find object **'stimulus_situation'**
Rule: 9 stacked	Object **'stimulus_situation'** sought as **'physical object'**
Pass 2	
Trying Rule: 3	Need to find object **'environment'**
Rule: 3 stacked	Object **'environment'** sought as **'machines'**
ask environment	
⇒**machines**	**'environment'** instantiated by user input to **'machines'**
Trying Rule: 3	**'stimulus_situation'** instantiated by Rule: 3 to **'physical object'**
Pass 3	
Trying Rule: 9	Need to find object **'stimulus_response'**
45 Rule: 9 stacked	Object **'stimulus_response'** sought as **'hands-on'**
Pass 4	
Trying Rule: 6	Need to find object **'job'**
Rule: 6 stacked	Object **'job'** sought as **'building'**
ask job	
⇒ **repairing**	**'job'** instantiated by user input to **'repairing'**
Trying Rule: 6	**'stimulus_response'** instantiated by Rule: 6 to **'hands-on'**
Pass 5	
Trying Rule: 9	Need to find object **'feedback'**
Rule: 9 stacked	Object **'feedback'** sought as **'required'**
ask feedback	
⇒ **required**	**'feedback'** instantiated by user input to **'required'**

Trying Rule: 9 **'medium'** instantiated by Rule: 9 to **'workshop'**

medium is workshop

用树形图来映射与专家系统的咨询会话很直观。图 2.8 是 MEDIA ADVISOR 的树形图。目标是根结点，当系统启动时，推理引擎搜寻目标的值。

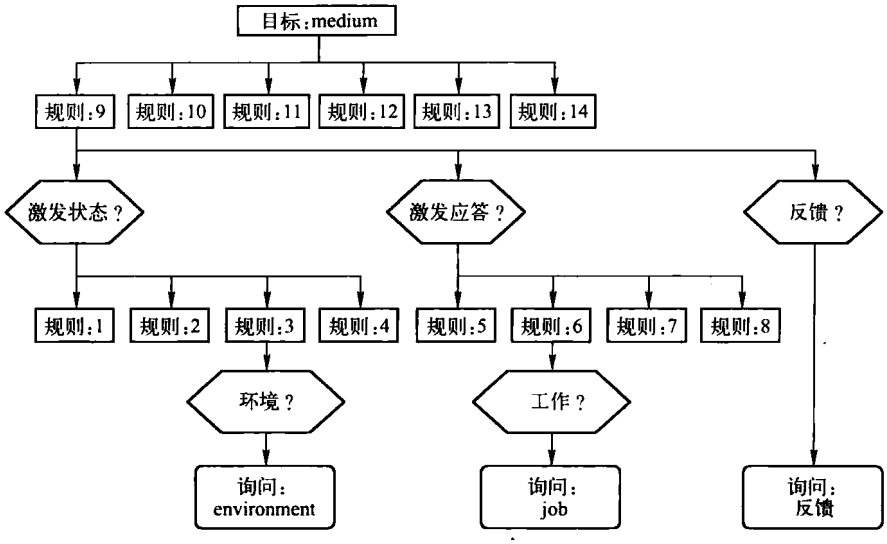


图 2.8 MEDIA ADVISOR 的树形图

46

MEDIA ADVISOR 能处理任何情况吗

当我们使用专家系统一段时间后，会发现可选项不能涵盖所有情况。例如，可能出现下面的对话。

What sort of environment is a trainee dealing with on the job?
⇒**illustrations**

In what way is a trainee expected to act or respond on the job?
⇒**drawing**

Is feedback on the trainee's progress required during training?
⇒**required**

I am unable to draw any conclusions on the basis of the data.

所以，MEDIA ADVISOR 在现阶段不能处理上面这些情况。幸运的是，通过添加更多规则能轻易地扩展专家系统，直到最终符合用户的需求。

2.8 冲突消解

在本章的开始，我们举过过马路的例子，那是两条规则，现在我们增加第三条规则，便得到了下面的规则集：

Rule 1:
IF the 'traffic light' is green
THEN the action is go

Rule 2:
IF the 'traffic light' is red
THEN the action is stop

Rule 3:
IF the 'traffic light' is red
THEN the action is go

将会发生什么

推理引擎将规则的 IF（条件）部分与数据库中的数据相比较，当条件满足时触发规则。由于一条规则被触发可能影响到其他规则，推理引擎每次只允许触发一条规则。过马路的例子有两条规则，规则 2 和规则 3，它们的 IF 部分相同。当它们的条件被满足时，两条规则都符合触发的条件，从而组成冲突集。当多个规则在一个周期内都满足触发条件时，需要从中选择一条规
47 则，这个方法称为*冲突消解*（conflict resolution）。

当交通灯变红时，该执行哪条规则

如果用前向链接，那么两条规则都会被触发。规则 2 由于顺序上的优先，先被触发，执行 THEN 部分，语言对象“action”的值是“stop”。“交通灯变红”这一事实仍在数据库中，规则 3 由于满足条件也被触发，相应的对象“action”的新值是“go”。通过这个简单的例子，可以看出当使用前向链接时，规则的顺序很关键。

怎样消解冲突

消解冲突的显而易见的策略是确立目标，当目标实现后不再触发规则。在过马路的例子中，目标是为语言对象“action”确立值。当专家系统为“action”赋值后，目标实现，系统终止。因此，当交通灯变红时，触发规则 2，“action”的值是“stop”，至此系统终止。在这个例子中，专家系统的结论是对的。但如果调换两个规则，结论就错了。这说明知识库中的规则顺序仍然至关重要。

存在其他消解办法吗

还有其他办法（Shirai and Tsuji，1985；Brachman and Levesque，2004；Giarratano and Riley，2004）。

- 触发优先权最高的规则。在简单的应用场景中，合理地排列知识库中的规则就确定了优先权。这种策略一般适于规则数在 100 条左右的系统。但在某些应用中，必须按照重要程度处理数据。例如，在一个医疗咨询中（Durkin，1994），按照如下方式设置优先权：

```
Goal 1. Prescription is? Prescription

RULE 1 Meningitis Prescription1
(Priority 100)
IF      Infection is Meningitis
AND     The Patient is a Child
THEN    Prescription is Number_1
AND     Drug Recommendation is Ampicillin
AND     Drug Recommendation is Gentamicin
AND     Display Meningitis Prescription1

RULE 2 Meningitis Prescription2
(Priority 90)
IF      Infection is Meningitis
AND     The Patient is an Adult
THEN    Prescription is Number_2
AND     Drug Recommendation is Penicillin
48 AND  Display Meningitis Prescription2
```

- 激发最具体的规则。这个方法也称为*最长匹配策略*，其依据的假设是具体规则比一般规则处理更多的信息。例如：

```
Rule 1:
IF      the season is autumn
AND     the sky is cloudy
AND     the forecast is rain
THEN    the advice is 'stay home'

Rule 2:
IF      the season is autumn
THEN    the advice is 'take an umbrella'
```

如果“季节是秋季、多云、预报有雨”这些条件满足，由于规则1的前项比规则2更具体，规则1被触发。如果仅满足“季节是秋季”，规则2被触发。

- 触发那些在数据库中最近加入的数据的规则。这个方法依于数据库中每个事实附带的时间标签。专家系统优先触发冲突集中最新加入数据的规则。例如：

```
Rule 1:
IF      the forecast is rain    {08:16 PM 11/25/96}
THEN    the advice is 'take an umbrella'
Rule 2:
IF      the weather is wet      {10:18 AM 11/26/96}
THEN    the advice is 'stay home'
```

假定这两条规则的IF部分都与数据库中的事实匹配。由于“weather is wet”这一事实进入数据库的时间晚于“forecast is rain”，规则2被触发。这个方法尤为适于数据库不断更新的实时应用环境。

以上的冲突消解方法比较简单，也易于实施。多数情况下，这些方法足够用了。但当程序越来越庞大、复杂时，知识工程师管理和检查知识库中的规则的任务就会逐渐困难。必须由专家系统分担任务，并理解系统本身的行为。

为了提高系统性能，我们需要为系统提供关于其所拥有的知识的知识，也就是*元知识*(metaknowledge)。

元知识可以简单地定义为关于*知识的知识*，是在专家系统中使用和控制领域知识的知识(Waterman，1986)。在基于规则的专家系统中，用元规则（metarule）来表示元知识。元规则决定专家系统中具体任务的规则使用策略。 49

元知识的起源是什么

知识工程师将领域专家的知识传达到专家系统，学习如何使用问题相关的规则，逐渐在头脑里形成新知识体，即关于专家系统行为的知识。这个新知识，即元知识，很大程度上独立于领域。例如：

```
Metarule 1:
Rules supplied by experts have higher priorities than rules supplied by
novices.

Metarule 2:
Rules governing the rescue of human lives have higher priorities than rules
concerned with clearing overloads on power system equipment.
```

专家系统会理解和使用元规则吗

一些专家系统为元规则提供了单独的引擎。但大多数专家系统都不能区分规则和元规则。所以必须对现有知识库中的元规则设定最高权限。当一条元规则触发时，能改变其他规则优先权的一些重要信息会被添加到数据库。

2.9 基于规则的专家系统的优点和缺点

当构建基于知识的系统时，基于规则的专家系统被公认为最好的选择。

什么特征使得基于规则的专家系统被知识工程师青睐

这些特征是：

- *自然语言表达*。专家通常会使用这样的表达来解释解决问题的过程：“在什么—什么情况下，我如何—如何做。”这样的表达可以被很自然地表达为IF-THEN产生式规则。
- *统一结构*。产生式规则具有统一的IF-THEN结构。每一条规则都是一个独立的知识。产生式规则的语法使得规则具有自释性。

- *知识与处理的相分离*。基于规则的专家系统的结构为知识库和推理引擎提供了有效的分离机制。因此，能够使用同一个专家系统框架开发不同的应用，系统本身也容易扩展。在不
50 干扰控制结构的同时通过添加一些规则，还能使系统更聪明。
- *处理不完整、不确定的知识*。大多数基于规则的专家系统都能表达和推理不完整、不确定的知识。例如：

```
IF      season is autumn
AND     sky is 'cloudy'
AND     wind is low
THEN    forecast is clear     {cf 0.1};
        forecast is drizzle   {cf 1.0};
        forecast is rain      {cf 0.9}
```

这条规则就表达了下面这条句子的不确定性。

“如果是秋季，看似在下毛毛雨，那么今天可能又很潮湿。”

这条规则用数值表达不确定性，称为*确信因子* {cf 0.1}。专家系统使用确信因子来确立规则结论的可信度或者可信水平。我们将在第3章具体讨论这一话题。

这些特征使得专家系统在现实问题的知识表达上非常适用。

基于规则的专家系统就完美无缺了吗

其有3个主要的缺点：

- *规则之间的关系不透明*。尽管单条规则都比较简单，也是自释性的，大量规则间的逻辑关系却可能不透明。在基于规则的系统中，难以观察单条规则如何对整个策略起作用，原因在于基于规则的专家系统缺乏分层的知识表达。
- *低效的搜索策略*。推理引擎在每个周期中搜索所有的规则。当规则很多时（多于100条规则），系统速度会很慢。基于规则的大型系统可能就不适用于实时应用。
- *没有学习能力*。一般的基于规则的专家系统都不具备从经验中学习的能力。人类专家知道何时打破规则，而专家系统并不能自动修改知识库，例如调整规则、添加规则。修改和维护系统的任务仍然由知识工程师来做。

2.10 小结

本章介绍了基于规则的专家系统，简要讨论了何为知识，专家如何以产生式规则形式
51 表达知识。我们分析了专家系统开发团队的主要成员，描述了基于规则的系统的结构，分析了专家系统的主要特征，并指出专家系统也有出错的时候。接下来回顾了前向链接、后向链接推理技术，对冲突消解策略进行了讨论。最后，我们分析了基于规则的专家系统的优缺点。

本章的主要内容为：

- 知识是对主题的理论和实践两方面的理解，是当前所知的总和。
- 专家是在特定领域具有深厚知识（以事实和规则的形式体现）和丰富经验的人，能够做他人做不到的事情。
- 专家通常以产生式规则形式表达知识。
- 产生式规则用IF（前提）THEN（结论）语句表达，它是最常用的知识表达方式。关系、建议、指示、策略以及启发式方法都可以用规则表达。
- 专家系统是在狭窄问题领域具有专家水平的计算机程序。基于规则的专家系统是最普及的专家系统。
- 开发基于规则的专家系统时，框架逐渐成为常用的选择。专家系统框架是没有知识的专家系统骨架。当针对应用构建系统时，用户只需以规则形式添加知识，并提供相关数据。因

此，专家系统框架大大降低了系统的开发时间。

- 专家系统团队由领域专家、知识工程师、程序员、项目经理以及终端用户组成。知识工程师设计、构建和测试专家系统，并从领域专家那里获取知识，确定推理方法，以及选择开发软件。当使用专家系统框架开发小型系统时，项目经理、知识工程师、程序员甚至领域专家都可能是同一个人。
- 基于规则的专家系统包括5个基本组成部分：知识库、数据库、推理引擎、解释设备以及用户界面。知识库包含用一系列规则表示的知识。数据库包含一系列事实，用于和规则的IF部分相匹配。推理引擎链接规则和事实，执行推理，以使系统得出解决方案。解释设备帮助用户查询系统如何得出一个结论，以及为什么需要某一个事实。用户界面则是用户和专家系统的互动途径。 52
- 专家系统将知识库和推理引擎分隔开，以分离知识和对知识的处理，从而使专家系统的构建和维护更加容易。当使用专家系统框架时，知识工程师或者专家只需将知识录入知识库。每添加一条新规则，都会扩充知识，也使专家系统更聪明。
- 专家系统在问题-解决会话中跟踪被触发的规则，从而具备一定的解释能力。
- 专家系统能够处理不完整、不确定的数据，也允许不精确的推理，这不同于传统程序。不过正如人类专家那样，当信息不完整或模糊时，专家系统也有出错的时候。
- 进行搜索和推理有两个主要方法：前向链接和后向链接推理技术。前向链接是数据驱动的：从已知数据出发进行推理，直到没有规则可以被激活。后向链接是目标驱动的：预先设定目标，推理引擎负责寻找证明目标的论据。
- 如果在一个循环内有多条规则满足触发条件，推理引擎必须决定该触发哪条规则，这种行为称为冲突消解。
- 基于规则的专家系统的优点是：能够有自然的知识表达方式，有统一的结构，知识和对知识的处理相分离，能处理不完整、不确定的知识。
- 基于规则的专家系统的缺点是：规则之间的关系不透明，搜索策略低效，不具备学习能力。

复习题

2.1 知识是什么？解释为什么在特定领域的有限范围内专家有丰富的知识。什么是启发式方法？

2.2 产生式规则是什么？举例说明产生式规则的两个必要部分。

2.3 列举并描述专家系统开发团队的5个主要成员。知识工程师的任务是什么？

2.4 什么是专家系统框架？解释为什么专家系统框架会大大降低系统的开发时间。

2.5 什么是产生式系统模型？列举并描述专家系统的5个基本组成部分。

2.6 专家系统的主要特征是什么？专家系统与传统程序的区别在哪里？ 53

2.7 专家系统会出错吗？为什么？

2.8 描述前向链接推理技术并举例。

2.9 描述后向链接推理技术并举例。

2.10 列举前向链接推理技术适合解决的问题。为什么后向链接适合解决诊断问题？

2.11 什么是规则冲突集？怎么解决冲突？列举并描述基本的冲突消解方法。

2.12 列举基于规则的专家系统的优缺点。

参考文献

Brachman, R.J. and Levesque, H.J. (2004). *Knowledge Representation and Reasoning.* Elsevier, San Francisco.

Duda, R., Gaschnig, J. and Hart, P. (1979). Model design in the PROSPECTOR consultant system for mineral exploration, *Expert Systems in the Microelectronic Age*, D. Michie, ed., Edinburgh University Press, Edinburgh, Scotland, pp. 153–167.

Durkin, J. (1994). *Expert Systems: Design and Development.* Prentice Hall, Englewood Cliffs, NJ.

Feigenbaum, E.A., Buchanan, B.G. and Lederberg, J. (1971). On generality and problem solving: a case study using the DENDRAL program, *Machine Intelligence 6*, B. Meltzer and D. Michie, eds, Edinburgh University Press, Edinburgh, Scotland, pp. 165–190.

Giarratano, J. and Riley, G. (2004). *Expert Systems: Principles and Programming*, 4th edn. Thomson/PWS Publishing Company, Boston, MA.

Negnevitsky, M. (2008). Computational intelligence approach to crisis management in power systems, *International Journal of Automation and Control*, 2(2/3), 247–273.

Newell, A. and Simon, H.A. (1972). *Human Problem Solving.* Prentice Hall, Englewood Cliffs, NJ.

Shirai, Y. and Tsuji, J. (1985). *Artificial Intelligence: Concepts, Technologies and Applications.* John Wiley, New York.

Shortliffe, E.H. (1976). *MYCIN: Computer-Based Medical Consultations.* Elsevier Press, New York.

Waterman, D.A. (1986). *A Guide to Expert Systems.* Addison-Wesley, Reading, MA.

Waterman, D.A. and Hayes-Roth, F. (1978). An overview of pattern-directed inference systems, *Pattern-Directed Inference Systems*, D.A. Waterman and F. Hayes-Roth, eds, Academic Press, New York.

54

基于规则的专家系统中的不确定性管理

本章主要介绍不确定性管理范例、贝叶斯推理和确信因子，讨论它们的优缺点，并以实例阐述这些理论。

3.1 不确定性简介

不完美是可供人类专家使用的信息的共同特点之一。信息可能是不完整的、不一致的、不确定的，也许会同时有这3个方面。也就是说，这样的信息总是不适合于解决问题。然而，专家却能应对这些缺陷，并通常做出正确的判断和决定。相应地，专家系统也应能够处理不确定因素，并给出有效的结论。

在专家系统中，不确定性指什么

不确定性是指，不具有使我们得出完美可信结论所需的准确的知识（Stephanou and Sage, 1987）。传统逻辑只允许精确推理，总假设存在完善知识，并总能使用排中律：

IF *A* is true
THEN *A* is not false

和

IF *B* is false
THEN *B* is not true

然而，用专家系统来解决的大多数实际问题并不能提供精确的知识，而是提供不精确、不完整甚至不可测的数据。

专家系统中不确定性知识的来源是什么

我们一般将来源分为4种：弱暗示、不精确的语言、未知数据，以及合并不同专家观点时的困难（Bonissone and Tong, 1985）。下面具体分析这4种来源。

- *弱暗示。*基于规则的专家系统常面临弱暗示和模糊联系的问题。在确立规则的IF（条件）和THEN（行为）两部分的具体关联时，领域专家和知识工程师会感到棘手，或者难以进行。因此，处理模糊联系就成了专家系统的任务，例如用数值型确信因子表达关联度。
- *不精确的语言。*自然语言是天生模糊、不精确的。我们在描述事实时，会用到“常常、有时、频繁地、几乎不”之类的术语。因此，难以用精确的IF－THEN产生式规则表达知识。如果将事实含义定量化，就可以用于专家系统。1944年，Ray Simpson让355个高中生或大学生对20个类似“常常”的术语表达为1～100范围内的数值（Simpson, 1944）。在1968年，Milton Hakel重做了这个实验（Hakel, 1968）。结果如表3.1所示。通过量化术语含义，使专家系统能建立规则的IF（条件）和事实之间的恰当匹配。
- *未知数据。*当数据不完整或缺失时，唯一的方法是将值设为“未知”，以使推理继续下去。
- *融合不同专家的观点。*大型专家系统通常会融合多个专家的知识和经验。例如，在开发勘探专家系统PROSPECTOR时，有9个专家参与（Duda et al., 1979）。不过，专家们的结论很少一致。观点不同导致规则冲突。为了解决冲突，知识工程师需要为每位专家分

配一个权重，并按权重综合各方结论。事实上，即使只有一个领域专家，他的经验水平也在变化。另外，不存在确定权重的系统性方法。

总之，由于任何实际领域都包含不确定知识，专家系统应能够管理不确定性，能够处理不完整、不一致甚至缺失的数据。在基于规则的专家系统中，已有许多数值型和非数值型方法来处理不确定性（Bhatnagar and Kanal，1986）。本章将讨论最通用的不确定性管理范例：贝叶斯推理和
56 确信因子。下面首先回顾一下经典概率论的基本原理。

表3.1 模糊术语、不精确术语在时间频率范围上的量化

Ray Simpson (1944)		Milton Hakel (1968)	
术语	均值	术语	均值
Always	99	Always	100
Very often	88	Very often	87
Usually	85	Usually	79
Often	78	Often	74
Generally	78	Rather often	74
Frequently	73	Frequently	72
Rather often	65	Generally	72
About as often as not	50	About as often as not	50
Now and then	20	Now and then	34
Sometimes	20	Sometimes	29
Occasionally	20	Occasionally	28
Once in a while	15	Once in a while	22
Not often	13	Not often	16
Usually not	10	Usually not	16
Seldom	10	Seldom	9
Hardly ever	7	Hardly ever	8
Very seldom	6	Very seldom	7
Rarely	5	Rarely	5
Almost never	3	Almost never	2
Never	0	Never	0

3.2 概率论基本知识

概率这一基本概念在我们的日常生活中扮演着重要角色，例如要下雨的概率，得到提升的可能性，澳大利亚板球队在下一个赛季获胜的几率，中了Tattslotto彩票百万美元的概率等。

概率的概念有着很长的历史，可回溯到上千年之前，那时口语中已经出现了诸如“probably”、“likely”、“maybe”、“perhaps”、“possibly”之类的词（Good，1959）。不过，概率的数学理论是17世纪才形成的。

如何定义概率

事件的概率是该事件发生所占的比例（Good，1959）。概率也可以定义为可能性的科学度量。Feller（1966；1968）和Fine（1973）写的两本著名教材中对现代概率理论进行了详细分析。本章仅研究在专家系统中与不确定性相关的基本概率概念。

概率可表示为0（不可能发生）到1（必然发生）范围内的数值型指标。大部分事件的概率

指标都严格限制在0~1之间，这意味着每一个事件至少有两个结果：有利的结果或成功，不利 57
的结果或失败。

成功、失败的概率计算公式可表示为：

$$P(\text{成功}) = \frac{\text{成功次数}}{\text{可能结果的总数}} \tag{3.1}$$

$$P(\text{失败}) = \frac{\text{失败次数}}{\text{可能结果的总数}} \tag{3.2}$$

因此，如果用 s 表示成功次数，f 表示失败次数，则有：

$$P(\text{成功}) = p = \frac{s}{s+f} \tag{3.3}$$

$$P(\text{失败}) = q = \frac{f}{s+f} \tag{3.4}$$

和

$$p + q = 1 \tag{3.5}$$

现在，我们来看经典的抛硬币和掷骰子的例子。在抛硬币时，正面朝上和反面朝上的概率是一样的。在一次投掷中，$s=f=1$，因此正面（或反面）朝上的概率是0.5。

再以掷骰子为例，分析单次投掷时得到6点的概率。假设我们的目标是6点，由于单次投掷时，仅有一种情况是6点，而有5种情况不是6点，所以 $s=1$，$f=5$。得到6点的概率是

$$p = \frac{1}{1+5} = 0.1666$$

结果不是6点的概率是

$$q = \frac{5}{1+5} = 0.8333$$

到现在为止，我们所观注的事件都是独立、互斥的（事件不可能同时发生）。在掷骰子的例子中，得到6点与得到1点这两个事件就是互斥的，因为不可能在一次投掷中同时得到6点和1点。同时，如果事件不独立，可能影响其他事件发生的可能。如果在掷骰子时，知道1点不会出现，我们 58
来分析单次投掷中得到6点的概率。虽然有5种情况不会出现6点，由于1点不会出现，就排除了1种情况，所以，

$$p = \frac{1}{1+(5-1)}$$

若 A、B 是真实世界中的事件。假设有 A、B 两个不互斥的事件，一个事件的发生条件依赖于另一个事件的发生。在事件 B 发生的基础上事件 A 发生的概率称为条件概率，用数学形式表示为 $p(A|B)$，其中的竖线“|”表示后面事件已发生。完整的概率表达式可理解为“在事件 B 已发生的情况下，事件 A 发生的条件概率”。

$$p(A|B) = \frac{A \text{ 和 } B \text{ 都发生的次数}}{B \text{ 发生的次数}} \tag{3.6}$$

A 和 B 都发生的概率，称为 A 和 B 的联合概率，它的数学表达形式是 $p(A \cap B)$。事件 B 发生的次数，或者 B 发生的概率，用 $p(B)$ 表示。那么有：

$$p(A|B) = \frac{p(A \cap B)}{p(B)} \tag{3.7}$$

类似地，在事件 A 发生的前提下，事件 B 发生的条件概率是

$$p(B|A)=\frac{p(B\cap A)}{p(A)} \tag{3.8}$$

因此，

$$p(B\cap A)=p(B|A)\times p(A) \tag{3.9}$$

由于联合概率具有可交换性，则有：

$$p(A\cap B)=p(B\cap A)$$

所以，

59

$$p(A\cap B)=p(B|A)\times p(A) \tag{3.10}$$

将公式（3.10）代入公式（3.7），则有：

$$p(A|B)=\frac{p(B|A)\times p(A)}{p(B)} \tag{3.11}$$

其中：

$p(A|B)$是在事件B发生时，事件A发生的条件概率；

$p(B|A)$是在事件A发生时事件B发生的条件概率；

$p(A)$是事件A发生的概率；

$p(B)$是事件B发生的概率。

公式（3.11）就是贝叶斯规则，是18世纪英国数学家Thomas Bayes首先提出了该规则，以他的名字命名的。

前面所讲的条件概率的概念都假定事件A依赖事件B。贝叶斯规则可以扩展为事件A依赖多个互斥事件B_1，B_2，…，B_n的情况。可将公式（3.7）扩展为以下情况：

$$\begin{aligned}p(A\cap B_1)&=p(A|B_1)\times p(B_1)\\p(A\cap B_2)&=p(A|B_2)\times p(B_2)\\&\vdots\\p(A\cap B_n)&=p(A|B_n)\times p(B_n)\end{aligned}$$

或合并为：

$$\sum_{i=1}^{n}p(A\cap B_i)=\sum_{i=1}^{n}p(A|B_i)\times p(B_i) \tag{3.12}$$

如果公式（3.12）包含了图3.1中的所有事件B_i，则有公式

$$\sum_{i=1}^{n}p(A\cap B_i)=p(A) \tag{3.13}$$

因此，公式（3.12）变为下面形式：

60

$$p(A)=\sum_{i=1}^{n}p(A|B_i)\times p(B_i) \tag{3.14}$$

如果事件A的发生依赖于两个互斥事件，即B发生或B不发生，公式（3.14）则变为下面形式：

$$p(A)=p(A|B)\times p(B)+p(A|\neg B)\times p(\neg B) \tag{3.15}$$

其中，¬表示逻辑运算符NOT。

同样的，

$$p(B)=p(B|A)\times p(A)+p(B|\neg A)\times p(\neg A) \tag{3.16}$$

将公式（3.16）代入公式（3.11），得到下面的贝叶斯规则公式：

$$p(A|B)=\frac{p(B|A)\times p(A)}{p(B|A)\times p(A)+p(B|\neg A)\times p(\neg A)} \tag{3.17}$$

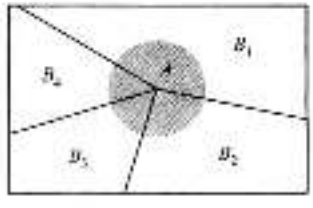


图3.1 联合概率

公式（3.17）为专家系统中的不确定性管理提供了概率论的应用背景。

3.3 贝叶斯推理

有了公式（3.17），我们就可以暂时放下基本的概率理论，将注意力转移到专家系统上。假设知识库中的所有规则都用下面形式表示：

IF *E* is true
THEN *H* is true {with probability *p*}

规则的含义是如果事件 E 发生，则事件 H 发生的概率是 p。

如果事件 E 已发生，但不知道事件 H 是否也发生，怎样计算 H 发生的概率

我们用公式（3.17）就能回答这个问题。我们现在使用符号 H 和 E，而不是 A 和 B。对于专家系统，H 表示假设，E 表示支持假设的论据。我们用 H 和 E 重写公式（3.17）：

$$p(H|E)=\frac{p(E|H)\times p(H)}{p(E|H)\times p(H)+p(E|\neg H)\times p(\neg H)} \tag{3.18}$$

其中：

$p(H)$ 是假设 H 为真的先验概率；

$p(E|H)$ 是假设 H 为真时导致论据 E 的概率；

$p(\neg H)$ 是假设 H 为假的先验概率；

$p(E|\neg H)$ 是在假设 H 为假时发现证据 E 的概率。

公式（3.18）表明在假设 H 的先验概率 $p(H)$ 必须在检验证据前定义。在专家系统中，解决问题时需用到的概率由专家提供，专家定义 $p(H)$ 和 $p(\neg H)$ 这两个先验概率，以及条件概率 $p(E|H)$ 和 $p(E|\neg H)$。用户提供论据 E 的有关信息，由专家系统计算 $p(H|E)$。$p(H|E)$ 称为后验概率。

如果专家根据单个论据 E，提出了多个假设 H_1、H_2、…、H_m，而非一个假设，或给定多个论据 E_1、E_2、…、E_n，专家也提供了多个假设

可以将公式（3.18）推广到包含多个假设 H_1、H_2、…、H_m 和多个论据 E_1、E_2、…、E_n 的场合，假设和论据都必须是互斥且完备的。

对于单个论据 E 和多个假设 H_1、H_2、…、H_m 的情况，有下面公式：

$$p(H_i|E)=\frac{p(E|H_i)\times p(H_i)}{\sum_{k=1}^{m}p(E|H_k)\times p(H_k)} \tag{3.19}$$

对于多个论据 E_1、E_2、…、E_n 和多个假设 H_1、H_2、…、H_m 的情况，则有：

$$p(H_i|E_1E_2\cdots E_n)=\frac{p(E_1E_2\cdots E_n|H_i)\times p(H_i)}{\sum_{k=1}^{m}p(E_1E_2\cdots E_n|H_k)\times p(H_k)} \tag{3.20}$$

在应用公式（3.20）时，需要论据和假设所有可能组合的条件概率。这对于专家来说是个巨大的负担，实际上不可能完成。所以，在专家系统中，可忽略微小的论据，并假设论据间条件独立（Ng and Abramson，1990）。因此我们得到了下面的公式

$$p(H_i|E_1E_2\cdots E_n)=\frac{p(E_1|H_i)\times p(E_2|H_i)\times\cdots\times p(E_n|H_i)\times p(H_i)}{\sum_{k=1}^{m}p(E_1|H_k)\times p(E_2|H_k)\times\cdots\times p(E_n|H_k)\times p(H_k)} \tag{3.21}$$

专家系统怎样计算所有的后验概率，并对真假设按可能性排名

考虑一个简单的例子。假设有3个条件独立的论据 E_1、E_2、E_3，让专家构造3个互斥且完备

的假设 H_1、H_2、H_3，并提供了这些假设的先验概率 $p(H_1)$、$p(H_2)$、$p(H_3)$，同时对于每一个假设，也确定了每一个论据的条件概率。表 3.2 为专家提供的先验概率和条件概率。

表 3.2 先验概率和条件概率

概率	假设		
	$i=1$	$i=2$	$i=3$
$p(H_i)$	0.40	0.35	0.25
$p(E_1 \mid H_i)$	0.3	0.8	0.5
$p(E_2 \mid H_i)$	0.9	0.0	0.7
$p(E_3 \mid H_i)$	0.6	0.7	0.9

63

假定我们首先看到了论据 E_3，专家系统按照公式（3.19）计算所有假设的后验概率：

$$p(H_i \mid E_3) = \frac{p(E_3 \mid H_i) \times p(H_i)}{\sum_{k=1}^{3} p(E_3 \mid H_k) \times p(H_k)}, \quad i = 1,2,3$$

因此，

$$p(H_1 \mid E_3) = \frac{0.6 \times 0.40}{0.6 \times 0.40 + 0.7 \times 0.35 + 0.9 \times 0.25} = 0.34$$

$$p(H_2 \mid E_3) = \frac{0.7 \times 0.35}{0.6 \times 0.40 + 0.7 \times 0.35 + 0.9 \times 0.25} = 0.34$$

$$p(H_3 \mid E_3) = \frac{0.9 \times 0.25}{0.6 \times 0.40 + 0.7 \times 0.35 + 0.9 \times 0.25} = 0.32$$

可以看出，当观察到论据E_3后，假设H_1的可信度降低，和H_2的可信度相同。假设H_3的可信度增加，几乎和H_1、H_2的可信度相等。

现在假设又得到了论据E_1，用公式（3.21）计算后验概率：

$$p(H_i \mid E_1E_3) = \frac{p(E_1 \mid H_i) \times p(E_3 \mid H_i) \times p(H_i)}{\sum_{k=1}^{3} p(E_1 \mid H_k) \times p(E_3 \mid H_k) \times p(H_k)}, \quad i = 1,2,3$$

所以有，

$$p(H_1 \mid E_1E_3) = \frac{0.3 \times 0.6 \times 0.40}{0.3 \times 0.6 \times 0.40 + 0.8 \times 0.7 \times 0.35 + 0.5 \times 0.9 \times 0.25} = 0.19$$

$$p(H_2 \mid E_1E_3) = \frac{0.8 \times 0.7 \times 0.35}{0.3 \times 0.6 \times 0.40 + 0.8 \times 0.7 \times 0.35 + 0.5 \times 0.9 \times 0.25} = 0.52$$

$$p(H_3 \mid E_1E_3) = \frac{0.5 \times 0.9 \times 0.25}{0.3 \times 0.6 \times 0.40 + 0.8 \times 0.7 \times 0.35 + 0.5 \times 0.9 \times 0.25} = 0.29$$

现在假设H_2的可能性最大，而H_1的可能性急剧下降。

假设又观察到了论据E_2，专家系统就可以计算所有假设最终的后验概率。

$$p(H_i \mid E_1E_2E_3) = \frac{p(E_1 \mid H_i) \times p(E_2 \mid H_i) \times p(E_3 \mid H_i) \times p(H_i)}{\sum_{k=1}^{3} p(E_1 \mid H_k) \times p(E_2 \mid H_k) \times p(E_3 \mid H_k) \times p(H_k)}, \quad i = 1,2,3$$

因此，

$$p(H_1 \mid E_1E_2E_3) = \frac{0.3 \times 0.9 \times 0.6 \times 0.40}{0.3 \times 0.9 \times 0.6 \times 0.40 + 0.8 \times 0.0 \times 0.7 \times 0.35 + 0.5 \times 0.7 \times 0.9 \times 0.25} = 0.45$$

$$p(H_2 \mid E_1E_2E_3) = \frac{0.8 \times 0.0 \times 0.7 \times 0.35}{0.3 \times 0.9 \times 0.6 \times 0.40 + 0.8 \times 0.0 \times 0.7 \times 0.35 + 0.5 \times 0.7 \times 0.9 \times 0.25}$$
$$= 0$$

$$p(H_3 \mid E_1E_2E_3) = \frac{0.5 \times 0.7 \times 0.9 \times 0.25}{0.3 \times 0.9 \times 0.6 \times 0.40 + 0.8 \times 0.0 \times 0.7 \times 0.35 + 0.5 \times 0.7 \times 0.9 \times 0.25}$$
$$= 0.55$$

尽管专家按可能性对3个假设的最初排名是 H_1、H_2、H_3，在观察到3个论据（E_1、E_2、E_3）后，只需考虑 H_1、H_3，可以放弃 H_2。通过计算我们认为 H_3 的可能性要大于 H_1。 64

用于勘探的专家系统 PROSPECTOR 是第一个使用贝叶斯规则计算 $p(H \mid E)$ 并在系统中传送不确定性的专家系统（Duda et al.，1979）。下面用一个简单的例子来解释专家系统中的贝叶斯推理。

3.4 FORECAST：论据累积的贝叶斯方法

现在我们开发一个解决实际问题（如天气预报）的专家系统。系统功能是预测明天是否下雨，它需要一些实际数据，这可以从气象局得到。

表3.3总结了伦敦在1982年3月的气象数据，包括每天的最低、最高温度，降雨量和日照时间。如果一天的降雨量为零，则说明一天不下雨。

专家系统所能提供的结果有两个：tomorrow is rain（明天下雨）、tomorrow is dry（明天不下雨），以及这两个结果的可能性。也就是说，专家系统必须给出“明天下雨”、“明天不下雨”这两个假设的条件概率。

在应用贝叶斯规则（公式（3.18））前，我们必须提供这两个假设的先验概率。

首先根据提供的数据写下两条基本规则，用它们预报明天的天气。

```
Rule: 1
IF      today is rain
THEN    tomorrow is rain

Rule: 2
IF      today is dry
THEN    tomorrow is dry
```

有了这两条规则，我们顶多犯10次错误，每次晴天后是雨天，雨天后是晴天。所以两个假设的先验概率都是0.5。接下来以下面形式重写规则：

```
Rule: 1
IF      today is rain {LS 2.5 LN .6}
THEN    tomorrow is rain {prior .5}

Rule: 2
IF      today is dry {LS 1.6 LN .4}
THEN    tomorrow is dry {prior .5}
```
65

表3.3 1982年3月伦敦天气总结

日	最低温度（℃）	最高温度（℃）	降雨量（mm）	日照（h）	实际天气	预报天气
1	9.4	11.0	17.5	3.2	下雨	—
2	4.2	12.5	4.1	6.2	下雨	下雨
3	7.6	11.2	7.7	1.1	下雨	下雨
4	5.7	10.5	0.0	4.3	晴天	下雨①
5	3.0	12.0	0.0	9.5	晴天	晴天
6	4.4	9.6	0.0	3.5	晴天	晴天
7	4.8	9.4	4.6	10.1	下雨	下雨

（续）

日	最低温度（℃）	最高温度（℃）	降雨量（mm）	日照（h）	实际天气	预报天气
8	1.8	9.2	5.5	7.8	下雨	下雨
9	2.4	10.2	4.8	4.1	下雨	下雨
10	5.5	12.7	4.2	3.8	下雨	下雨
11	3.7	10.9	4.4	9.2	下雨	下雨
12	5.9	10.0	4.8	7.1	下雨	下雨
13	3.0	11.9	0.0	8.3	晴天	下雨①
14	5.4	12.1	4.8	1.8	下雨	晴天①
15	8.8	9.1	8.8	0.0	下雨	下雨
16	2.4	8.4	3.0	3.1	下雨	下雨
17	4.3	10.8	0.0	4.3	晴天	晴天
18	3.4	11.1	4.2	6.6	下雨	下雨
19	4.4	8.4	5.4	0.7	下雨	下雨
20	5.1	7.9	3.0	0.1	下雨	下雨
21	4.4	7.3	0.0	0.0	晴天	晴天
22	5.6	14.0	0.0	6.8	晴天	晴天
23	5.7	14.0	0.0	8.8	晴天	晴天
24	2.9	13.9	0.0	9.5	晴天	晴天
25	5.8	16.4	0.0	10.3	晴天	晴天
26	3.9	17.0	0.0	9.9	晴天	晴天
27	3.8	18.3	0.0	8.3	晴天	晴天
28	5.8	15.4	3.2	7.0	下雨	晴天①
29	6.7	8.8	0.0	4.2	晴天	晴天
30	4.5	9.6	4.8	8.8	下雨	下雨
31	4.6	9.6	3.2	4.2	下雨	下雨

① 表示预报错误。

LS 的值是在论据 E 存在时，专家估计假设 H 的可能性，称为*充分性似然值*，定义为 $p(E|H)$ 与 $p(E|\neg H)$ 的比值。

$$LS = \frac{p(E|H)}{p(E|\neg H)} \tag{3.22}$$

在我们的例子中，*LS* 是在明天下雨的情况下，今天下雨的概率，除以如果明天不下雨，今天下雨的可能性，

66

$$LS = \frac{p(\text{今天下雨}|\text{明天下雨})}{p(\text{今天下雨}|\text{明天不下雨})}$$

LN 则是在没有论据 E 时，假设 H 不被信任的度量，也称为*必要性似然值*，计算公式是

$$LN = \frac{p(\neg E|H)}{p(\neg E|\neg H)} \tag{3.23}$$

在我们的例子中，*LN* 表示在明天下雨的情况下今天不下雨的概率，除以如果明天不下雨，今天不下雨的概率，

$$LN = \frac{p(\text{今天不下雨}|\text{明天下雨})}{p(\text{今天不下雨}|\text{明天不下雨})}$$

注意，从 *LS* 的计算公式不能推出 *LN*，领域专家必须分别提供这两个值。

领域专家如何确定充分性似然值和必要性似然值呢？需要他们处理条件概率吗

为了确定 *LS* 和 *LN* 的值，专家不必给出精确的条件概率值，而是直接确定似然值。*LS* 值大（*LS*≫1）就表明在论据存在时规则强烈支持假设，*LN* 值小（0 < *LN* < 1）表明在论据不存在时规则强烈反对假设。

由于从 *LS* 和 *LN* 的似然值可以轻易地算出条件概率，就可以使用贝叶斯规则来传送论据。

现在继续看伦敦天气的例子。规则 1 说明，如果今天下雨，明天也下雨的可能性就很大（*LS* = 2.5）。即使今天不下雨，明天也可能有雨（*LN* = 0.6）。

规则 2 则考虑了不下雨的情况。如果今天不下雨，明天也很有可能不下雨（*LS* = 1.6），而在今天下雨的情况下，明天下雨的概率要高于在今天不下雨的情况下，明天也不下雨的概率。为什么呢？*LS* 和 *LN* 的值通常由领域专家给出。在天气预报的例子中，*LS* 和 *LN* 的值可以由气象局发布的统计信息来确认。规则 2 也给出了在今天下雨的情况下明天不下雨的可能性（*LN* = 0.4）。

专家系统怎样得到明天不下雨或下雨的全部的概率

基于规则的专家系统将后项的先验概率 $p(H)$ 转换为先验几率。

$$O(H) = \frac{p(H)}{1 - p(H)} \tag{3.24}$$

67

仅在第一次调整后项的不确定性时，才使用先验概率。为了计算后验几率，当规则的前项（即论据）为真时，使用 *LS* 更新先验几率。当规则的前项假时，使用 *LN* 更新先验几率。

$$O(H|E) = LS \times O(H) \tag{3.25}$$

和

$$O(H|\neg E) = LN \times O(H) \tag{3.26}$$

再使用后验几率来恢复后验概率：

$$p(H|E) = \frac{O(H|E)}{1 + O(H|E)} \tag{3.27}$$

和

$$p(H|\neg E) = \frac{O(H|\neg E)}{1 + O(H|\neg E)} \tag{3.28}$$

用伦敦天气的例子解释这一方法的用法。假设用户指出今天下雨，规则 1 被触发，明天下雨的先验概率转换为先验几率：

$$O(\text{明天下雨}) = \frac{0.5}{1 - 0.5} = 1.0$$

今天下雨的论据将几率增加了 2.5 倍，所以明天下雨的概率从 0.5 增加到 0.71：

$$O(\text{明天下雨}|\text{今天下雨}) = 2.5 \times 1.0 = 2.5$$

$$p(\text{明天下雨}|\text{今天下雨}) = \frac{2.5}{1 + 2.5} = 0.71$$

规则 2 也被触发。明天不下雨的先验概率转换为先验几率，但今天下雨的论据将几率减小到 0.4。明天不下雨的概率就从 0.5 变为 0.29：

$$O(\text{明天不下雨}) = \frac{0.5}{1 - 0.5} = 1.0$$

$$O(\text{明天不下雨}|\text{今天下雨}) = 0.4 \times 1.0 = 0.4$$

$$p(\text{明天不下雨}|\text{今天下雨}) = \frac{0.4}{1 + 0.4} = 0.29$$

68

结论是，如果今天下雨，明天下雨的可能性是71%，不下雨的可能性是29%。

如果用户输入今天不下雨。同样的方式可计算出明天不下雨的可能性是62%，下雨的可能性是38%。

研究了论据的贝叶斯规则的基本原理后，我们可以向专家系统中添加新知识。首先我们需要确定天气发生变化的条件。通过对表3.3中的数据进行分析，我们建立了如下知识库（使用了专家系统框架 Leonardo）。

知识库

```
/* FORECAST: BAYESIAN ACCUMULATION OF EVIDENCE

control bayes

Rule: 1
if      today is rain {LS 2.5 LN .6}
then    tomorrow is rain {prior .5}

Rule: 2
if      today is dry {LS 1.6 LN .4}
then    tomorrow is dry {prior .5}

Rule: 3
if      today is rain
and     rainfall is low {LS 10 LN 1}
then    tomorrow is dry {prior .5}

Rule: 4
if      today is rain
and     rainfall is low
and     temperature is cold {LS 1.5 LN 1}
then    tomorrow is dry {prior .5}

Rule: 5
if      today is dry
and     temperature is warm {LS 2 LN .9}
then    tomorrow is rain {prior .5}

Rule: 6
if      today is dry
and     temperature is warm
and     sky is overcast {LS 5 LN 1}
then    tomorrow is rain {prior .5}

/* The SEEK directive sets up the goal of the rule set

seek tomorrow
```

69

对话

根据用户提供的信息，专家系统判断明天是否不下雨。用户提供的内容用箭头标出。如果降雨量少于4.1mm，则为低降雨量。如果日平均气温等于或低于7.0℃，则认为天气寒冷，如果高于7.0℃，则认为天气暖和。如果日照时间低于4.6h，则认为是阴天。

What is the weather today?
⇒ **rain**

```
Rule: 1
if      today is rain {LS 2.5 LN .6}
then    tomorrow is rain {prior .5}
```

$$O(\textit{tomorrow is rain}) = \frac{0.5}{1-0.5} = 1.0$$

$$O(\textit{tomorrow is rain} \mid \textit{today is rain}) = 2.5 \times 1.0 = 2.5$$

$$p(\textit{tomorrow is rain} \mid \textit{today is rain}) = \frac{2.5}{1+2.5} = 0.71$$

```
tomorrow is rain    {0.71}
```

Rule: 2
if today is dry {LS 1.6 LN .4}
then tomorrow is dry {prior .5}

$$O(\textit{tomorrow is dry}) = \frac{0.5}{1-0.5} = 1.0$$

$$O(\textit{tomorrow is dry} \mid \textit{today is rain}) = 0.4 \times 1.0 = 0.4$$

$$p(\textit{tomorrow is dry} \mid \textit{today is rain}) = \frac{0.4}{1+0.4} = 0.29$$

tomorrow is rain {0.71}
dry {0.29}

What is the rainfall today?
⇒ **low**

Rule: 3
if today is rain
and rainfall is low {LS 10 LN 1}
then tomorrow is dry {prior .5}

$$O(\textit{tomorrow is dry}) = \frac{0.29}{1-0.29} = 0.41$$

$$O(\textit{tomorrow is dry} \mid \textit{today is rain} \cap \textit{rainfall is low}) = 10 \times 0.41 = 4.1$$

70

$$p(\textit{tomorrow is dry} \mid \textit{today is rain} \cap \textit{rainfall is low}) = \frac{4.1}{1+4.1} = 0.80$$

tomorrow is dry {0.80}
rain {0.71}

What is the temperature today?
⇒ **cold**

Rule: 4
if today is rain
and rainfall is low
and temperature is cold {LS 1.5 LN 1}
then tomorrow is dry {prior .5}

$$O(\textit{tomorrow is dry}) = \frac{0.80}{1-0.80} = 4$$

$$O(\textit{tomorrow is dry} \mid \textit{today is rain} \cap \textit{rainfall is low} \cap \textit{temperature is cold}) = 1.50 \times 4 = 6$$

$$p(\textit{tomorrow is dry} \mid \textit{today is rain} \cap \textit{rainfall is low} \cap \textit{temperature is cold}) = \frac{6}{1+6} = 0.86$$

tomorrow is dry {0.86}
rain {0.71}

Rule: 5
if today is dry
and temperature is warm {LS 2 LN .9}
then tomorrow is rain {prior .5}

$$O(\textit{tomorrow is rain}) = \frac{0.71}{1-0.71} = 2.45$$

$$O(\textit{tomorrow is rain} \mid \textit{today is not dry} \cap \textit{temperature is not warm}) = 0.9 \times 2.45 = 2.21$$

$$p(\textit{tomorrow is rain} \mid \textit{today is not dry} \cap \textit{temperature is not warm}) = \frac{2.21}{1+2.21} = 0.69$$

tomorrow is dry {0.86}
rain {0.69}

What is the cloud cover today?
⇒ **overcast**

Rule: 6
if today is dry
and temperature is warm
and sky is overcast {LS 5 LN 1}
71 then tomorrow is rain {prior .5}

$$O(\textit{tomorrow is rain}) = \frac{0.69}{1-0.69} = 2.23$$

$$O(\textit{tomorrow is rain} \mid \textit{today is not dry} \cap \textit{temperature is not warm} \cap \textit{sky is overcast}) = 1.0 \times 2.23 = 2.23$$

$$p(\textit{tomorrow is rain} \mid \textit{today is not dry} \cap \textit{temperature is not warm} \cap \textit{sky is overcast}) = \frac{2.23}{1+2.23} = 0.69$$

tomorrow is dry {0.86}
rain {0.69}

这说明两个假设“明天不下雨”、“明天下雨”都极有可能发生，但前者的可能性更高。

从表3.3可以看出专家系统仅犯过4次错误，成功率是86%，比同样预测伦敦天气的Naylor（1987）结果还要好。

3.5 贝叶斯方法的偏差

贝叶斯推理框架需要概率值作为初始输入。对这些值的估计常常需要人的评判。不过心理研究表明，人给出的概率值要么与贝叶斯规则不一致，要么做得很糟糕（Burns and Pearl，1981；Tversky and Kahneman，1982）。这说明条件概率可能与专家提供的先验概率不一致。例如，一辆汽车不能启动，在按下启动按钮时发出奇怪的噪声。如果汽车发出奇怪的噪声，故障出在启动按钮上的条件概率可表达为

IF the symptom is 'odd noises'
THEN the starter is bad {with probability 0.7}

而如果汽车发出奇怪的噪声，故障不在启动按钮上的条件概率是

$$p(\text{starter is not bad} \mid \text{odd noises}) = p(\text{starter is good} \mid \text{odd noises}) = 1 - 0.7 = 0.3$$

所以，我们有另外一条规则

IF the symptom is 'odd noises'
72 THEN the starter is good {with probability 0.3}

领域专家在处理条件概率时并不轻松，经常否定隐含概率的存在（在汽车例子中隐含概率是0.3）。

在本例中，通过可用的统计信息和经验学习，可以得到下面两条规则：

IF the starter is bad
THEN the symptom is 'odd noises' {with probability 0.85}

IF the starter is bad
THEN the symptom is not 'odd noises' {with probability 0.15}

要使用贝叶斯规则，还必须有先验概率，即当汽车不能启动时，启动按钮有故障的概率。这需要专家来提供。假设专家认为这一概率是5%，我们应用贝叶斯规则得到下面的式子：

$$p(\text{starter is bad} \mid \text{odd noises}) = \frac{0.85 \times 0.05}{0.85 \times 0.05 + 0.15 \times 0.95} = 0.23$$

这个值远低于最初由专家提供的估计值0.7。

为什么会不一致？专家错了吗

最明显的原因是专家在估计条件概率和先验概率时做了不同的假设。我们试着从后验概率p(starter is bad | odd noises)倒推先验概率p(starter is bad)。在本例中，可以假设

$$p(\text{starter is good}) = 1 - p(\text{starter is bad})$$

根据公式（3.18），我们得出

$$p(H)=\frac{p(H|E)\times p(E|\neg H)}{p(H|E)\times p(E|\neg H)+p(E|H)[1-p(H|E)]}$$

其中：

$p(H)=p$(starter is bad)；

$p(H|E)=p$(starter is bad | odd noises)；

$p(E|H)=p$(odd noises | starter is bad)；

$p(E|\neg H)=p$(odd noises | starter is good)。

如果我们使用专家提供的 p(starter is bad | odd noises) 的值 0.7 作为正确值，p(starter is bad) 的先验概率应该是 73

$$p(H)=\frac{0.7\times 0.15}{0.7\times 0.15+0.85\times(1-0.7)}=0.29$$

这个值几乎是专家提供的值5%的6倍。因此专家应该对先验概率和条件概率分别进行不同的估计。

事实上，由专家提供的先验概率也会与充分性似然值 *LS*、必要性似然值 *LN* 不一致。已有若干方法处理这一问题（Duda et al.，1976）。最常用的技术是分段线性插值模型（Duda et al.，1979），首次应用在 PROSPECTOR 中。

要使用贝叶斯规则，我们必须满足多个假定条件，如针对假设和逆假设，论据间都应是条件独立的。由于很少有实际问题满足这些假定条件，使用贝叶斯推理的系统并不多，用于矿产勘探的专家系统 PROSPECTOR 是其中之一（Duda et al.，1979）。

3.6 确信因子理论和基于论据的推理

确信因子理论是常用的替代贝叶斯推理的方法，这一理论的基本原理首次出现在 MYCIN 中，这是一个用于诊断、治疗传染性血液病和脑膜炎的专家系统（Shortliffe and Buchanan，1975）。MYCIN 的开发者发现，医学专家通常用既不符合逻辑一致性或数学一致性的方式表达术语的可信度。此外，不存在问题领域相关的可信赖的统计数据。所以，MYCIN 团队不能使用传统概率方法。于是，他们决定用确信因子（certainty factor，cf）来衡量专家对术语的可信度。确信因子的最大值是 +1.0（真），最小值是 -1.0（假）。正值表示可信度，负值表示不可信度。例如，如果专家说一个论据几乎是真的，将对 *cf* 值赋值 0.8。表 3.4 是 MYCIN 系统中的一些不确定性术语（Durkin，1994）。

表 3.4 不确定性术语及其解释

术语	确信因子
Definitely not	-1.0
Almost certainly not	-0.8
Probably not	-0.6
Maybe not	-0.4
Unknown	-0.2 ~ +0.2
Maybe	+0.4
Probably	+0.6
Almost certainly	+0.8
Definitely	+1.0

在使用确信因子的专家系统中，知识库中规则的语法是：

IF <evidence>
THEN <hypothesis> {*cf*}

其中，*cf* 是论据 *E* 发生时假设 *H* 的可信度。

可信度理论基于两个函数：可信度的度量 *MB*（*H*, *E*），不可信度的度量 *MD*（*H*, *E*）
74 （Shortliffe and Buchanan, 1975）。这两个函数分别表示在论据 *E* 出现时，假设 *H* 的可信度的增加程度和不可信度的增加程度。

可信度和不可信度可通过先验概率、条件概率计算（Ng and Abramson, 1990）：

$$MB(H,E)=\begin{cases}1 & \text{若 } p(H)=1\\ \dfrac{max[p(H|E),p(H)]-p(H)}{max[1,0]-p(H)} & \text{否则}\end{cases} \tag{3.29}$$

$$MD(H,E)=\begin{cases}1 & \text{若 } p(H)=1\\ \dfrac{min[p(H|E),p(H)]-p(H)}{min[1,0]-p(H)} & \text{否则}\end{cases} \tag{3.30}$$

其中：

$p(H)$ 是假设 *H* 为真的先验概率；

$p(H|E)$ 是给定论据 *E* 时假设 *H* 为真的概率。

$MB(H,E)$ 和 $MD(H,E)$ 的取值范围都介于 0 ~ 1 之间。假设 *H* 的可信度和不可信度都依赖论据 *E* 的类型。有些事实可能会提高可信度，而有些事实可能会提高不可信度。

如何确定假设的可信度和不可信度的总体情况

使用下面的公式将可信度和不可信度综合为一个数值，即确信因子：

$$cf=\frac{MB(H,E)-MD(H,E)}{1-min[MB(H,E),MD(H,E)]} \tag{3.31}$$

75

因此，*cf* 表达了假设 *H* 的总体可信度，它在 MYCIN 中的范围是 -1 ~ +1。

可以通过一个例子来说明 MYCIN 中的方法。考虑下面一条简单的规则：

IF *A* is *X*
THEN *B* is *Y*

专家通常不能保证这个规则绝对有效。在一些情况下，即使规则的 IF 部分条件成立，即对象 *A* 的值是 *X*，对象 *B* 也可能有不同的 *Z* 值。也就是说，我们讨论的是准统计的不确定性。

当 *A* 取值 *X* 时，专家通常会将 *B* 的每一个可能值关联上一个确信因子。因此，规则应该像下面的形式：

IF *A* is *X*
THEN *B* is *Y* {*cf* 0.7};
B is *Z* {*cf* 0.2}

另外 10% 哪去了

这个规则的意思是，假定 *A* 的值是 *X*，则 *B* 取 *Y* 值的概率是 70%，取 *Z* 值的概率是 20%，取其他值的概率是另外 10%。就是专家保留了对象 *B* 取 *Y* 和 *Z* 之外其他取值的可能性。所以，对象 *B* 被赋予了多个值。

规则中的确信因子通过推理链来传送。确信因子的传送还包括当规则前项中的论据不确定时，确立规则后项的净确信度。单个规则前项的净确信度 $cf(H,E)$，可以由规则前项的确信因子 $cf(E)$ 乘以规则的确信因子 *cf* 得到

$$cf(H,E)=cf(E)\times cf \tag{3.32}$$

例如，

IF the sky is clear
THEN the forecast is sunny {*cf* 0.8}

"sky is clear" 的当前确信因子是0.5，那么

$$cf(H,E) = 0.5 \times 0.8 = 0.4$$

按照表3.4，这个结果意味着 "可能晴天"。 76

对于有多个前项的规则，专家系统如何计算确信因子

对于合取规则，例如：

IF <evidence E_1>
AND <evidence E_2>
⋮
AND <evidence E_n>
THEN <hypothesis H> {*cf*}

后项的净确信度，即假设 H 的确信度用以下公式计算：

$$cf(H,E_1 \cap E_2 \cap \cdots \cap E_n) = min[cf(E_1),cf(E_2),\cdots,cf(E_n)] \times cf \tag{3.33}$$

例如，

IF sky is clear
AND the forecast is sunny
THEN the action is 'wear sunglasses' {*cf* 0.8}

"sky is clear" 的确信度是0.9，"the forecast is sunny" 的确信度是0.7，那么

$$cf(H,E_1 \cap E_2) = min[0.9,0.7] \times 0.8 = 0.7 \times 0.8 = 0.56$$

根据表3.4，结论可以解释为 "或许今天适合戴太阳镜"。

对于析取规则，如

IF <evidence E_1>
OR <evidence E_2>
⋮
OR <evidence E_n>
THEN <hypothesis H> {*cf*}

假设 H 的确信度通过下式计算：

$$cf(H,E_1 \cup E_2 \cup \cdots \cup E_n) = max[cf(E_1),cf(E_2),\cdots,cf(E_n)] \times cf \tag{3.34}$$ 77

例如，

IF sky is overcast
OR the forecast is rain
THEN the action is 'take an umbrella' {*cf* 0.9}

"sky is overcast" 的确信度是0.6，"forecast is rain" 的确信度是0.8，则

$$cf(H,E_1 \cup E_2) = max[0.6,0.8] \times 0.9 = 0.8 \times 0.9 = 0.72$$

结论可解释为 "今天几乎肯定是要带上雨伞"。

有时两个或更多的规则会影响到同一个假设，这种情况下专家系统怎么办

当同一个结论要由两个或多个规则共同推出时，必须合并它们的确信因子，以得出假设的确信因子。假设知识库中有如下规则：

Rule 1: IF A is X
THEN C is Z {*cf* 0.8}

Rule 2: IF B is Y
THEN C is Z {*cf* 0.6}

如果规则1和规则2都被触发了，对象 C 取 Z 值的确信度如何计算？按照常识，如果有来自不同来源（规则1和规则2）的两个论据（A is X 和 B is Y），都支持同一个假设（C is Z），假设的可信度应该提高，应该比只有一个论据时更强。

我们使用下面的公式来计算结合后的确信因子（Durkin，1994）：

$$cf(cf_1,cf_2)=\begin{cases} cf_1+cf_2\times(1-cf_1) & \text{若 } cf_1>0,\ cf_2>0 \\ \dfrac{cf_1+cf_2}{1-min[\,|cf_1|,|cf_2|\,]} & \text{若 } cf_1<0 \text{ 或者 } cf_2<0 \\ cf_1+cf_2\times(1+cf_1) & \text{若 } cf_1<0 \text{ 或者 } cf_2<0 \end{cases} \tag{3.35}$$

其中：

cf_1是规则1对假设H的确信度；

cf_2是规则2对假设H的确信度；

78 $|cf_1|$和$|cf_2|$分别是cf_1和cf_2的绝对值。

因此，我们假设

$$cf(E_1)=cf(E_2)=1.0$$

由公式（3.32）可得

$$cf_1(H,E_1)=cf(E_1)\times cf_1=1.0\times0.8=0.8$$
$$cf_2(H,E_2)=cf(E_2)\times cf_2=1.0\times0.6=0.6$$

由公式（3.35）可得

$$cf(cf_1,cf_2)=cf_1(H,E_1)+cf_2(H,E_2)\times[1-cf_1(H,E_1)]$$
$$=0.8+0.6\times(1-0.8)=0.92$$

这个例子表明假设的可信度增加，也验证了我们的预期。

接下来考虑规则的确信因子是负值的情况。假设

$$cf(E_1)=1,\ cf(E_2)=-1.0$$

那么

$$cf_1(H,E_1)=1.0\times0.8=0.8$$
$$cf_2(H,E_2)=-1.0\times0.6=-0.6$$

由公式（3.35）可得

$$cf(cf_1,cf_2)=\frac{cf_1(H,E_1)+cf_2(H,E_2)}{1-min[\,|cf_1(H,E_1)|,|cf_2(H,E_2)|\,]}=\frac{0.8-0.6}{1-min[0.8,0.6]}=0.5$$

该例说明了当一条规则支持假设而另一条规则反对假设的情况下，如何计算确信因子，即净可信度。

如果规则的确信因子都是负值呢？假设

$$cf(E_1)=cf(E_2)=-1.0$$

那么

$$cf_1(H,E_1)=-1.0\times0.8=-0.8$$
79 $$cf_2(H,E_2)=-1.0\times0.6=-0.6$$

由公式（3.35）可得：

$$cf(cf_1,cf_2)=cf_1(H,E_1)+cf_2(H,E_2)\times[1+cf_1(H,E_1)]$$
$$=-0.8-0.6\times(1-0.8)=-0.92$$

这个例子中假设的不可信度有了增加。

确信因子理论是贝叶斯推理的实用的替代方法。合并确信因子的启发式方法不同于合并概率的方法。确定理论不是“纯粹数学”的方法，而是模拟人类专家思考过程的方法。

下面通过3.4节介绍的专家系统FORECAST，来说明基于论据的推理以及确信因子在一组规则中传送的方法。

3.7 FORECAST：确信因子的应用

这个专家系统的任务是预测明天是否下雨，即为多值对象 tomorrow 建立确信因子。为了简化任务，我们仍使用 3.4 节的相关规则。

知识库

```
/* FORECAST: AN APPLICATION OF CERTAINTY FACTORS

control cf

control 'threshold 0.01'

Rule: 1
if      today is rain
then    tomorrow is rain {cf 0.5}

Rule: 2
if      today is dry
then    tomorrow is dry {cf 0.5}

Rule: 3
if      today is rain
and     rainfall is low
then    tomorrow is dry {cf 0.6}

Rule: 4
if      today is rain
and     rainfall is low
and     temperature is cold
then    tomorrow is dry {cf 0.7}

Rule: 5
if      today is dry
and     temperature is warm
then    tomorrow is rain {cf 0.65}

Rule: 6
if      today is dry
and     temperature is warm
and     sky is overcast
then    tomorrow is rain {cf 0.55}

seek tomorrow
```

对话

为了应用基于确信因子的不精确推理技术，专家系统让用户既输入对象的值，又输入与值关联的确信度。例如，如果使用 Leonardo 规定的 0 ~ 1 的范围，可能会出现下面的对话：

What is the weather today?
⇒ **rain**

```
Rule: 1
if      today is rain
then    tomorrow is rain {cf 0.5}
```

cf(tomorrow is rain, today is rain) = cf(today is rain) × cf = 1.0 × 0.5 = 0.5

```
tomorrow is rain    {0.50}
```

What is the rainfall today?
⇒ **low**

To what degree do you believe the rainfall is low? Enter a numeric certainty between 0 and 1.0 inclusive.
⇒ **0.8**

```
Rule: 3
if      today is rain
and     rainfall is low
then    tomorrow is dry {cf 0.6}
```

cf(tomorrow is dry, today is rain ∩ rainfall is low)
$= min[cf(\text{today is rain}), cf(\text{rainfall is low})] \times cf = min[1, 0.8] \times 0.6 = 0.48$

tomorrow is rain {0.50}
81 dry {0.48}

What is the temperature today?
⇒ **cold**

To what degree do you believe the temperature is cold? Enter a numeric certainty between 0 and 1.0 inclusive.
⇒ **0.9**

Rule: 4
if today is rain
and rainfall is low
and temperature is cold
then tomorrow is dry {cf 0.7}

cf(tomorrow is dry, today is rain ∩ rainfall is low ∩ temperature is cold)
$= min[cf(\text{today is rain}), cf(\text{rainfall is low}), cf(\text{temperature is cold})] \times cf$
$= min[1, 0.8, 0.9] \times 0.7 = 0.56$

tomorrow is dry {0.56}
rain {0.50}

$$cf(cf_{\text{Rule:3}}, cf_{\text{Rule:4}}) = cf_{\text{Rule:3}} + cf_{\text{Rule:4}} \times (1 - cf_{\text{Rule:3}})$$
$$= 0.48 + 0.56 \times (1 - 0.48) = 0.77$$

tomorrow is dry {0.77}
rain {0.50}

3.8 贝叶斯推理和确信因子的对比

在上一节，我们介绍了专家系统中的两个流行的不确定性管理技术。现在我们比较这两个技术，分析它们都适合哪些问题。

概率理论是处理不精确知识和随机数据的最古老、最好的技术，很适用于预测和计划相关的问题，因为可以获得统计数据，并编写准确的概率语句。

专家系统 PROSPECTOR 是一个帮助勘探地质学家搜寻矿床的专家系统，它使用了贝叶斯技术。这个专家系统开发得很成功，如通过使用地质、地球物理、地球化学数据，PROSPECTOR 预测出了华盛顿州 Tolman 山附近的一个钼矿床（Campbell et al.，1982）。PROSPECTOR 团队能
82 利用已知矿床的有效数据和统计信息，也定义了每一个事件的概率。PROSPECTOR 团队也假定了论据之间具有条件独立性，这也是使用贝叶斯规则的前提。

但是，在许多应用领域，并不总能得到可用的统计信息，或总能假定论据间满足条件独立性。所以，许多研究者发现贝叶斯方法对他们的工作并不适合。例如，在 MYCIN 中，由于医学领域通常不提供需要的数据，Shortliffe 和 Buchanan 就不能使用传统的概率方法（Shortliffe and Buchanan，1975）。

尽管确信因子方法缺乏概率论那样的数学正确性，在一些领域如在诊断领域、尤其是医药领域，它还是能够胜过贝叶斯推理方法的。在像 MYCIN 这样的诊断专家系统中，专家根据自己的知识或直觉判断来确定规则和确信因子。在概率未知、或难以获取或代价昂贵的场合，可以使用确信因子。论据推理机制可以增量管理获得的论据、合取式假设、析取式假设，以及可信度不同的论据。另外，确信因子方法对基于规则的专家系统中的控制流能提供更好的解释。

贝叶斯方法和确信因子虽彼此不同，它们却有着共同问题：找到能够量化个人的、主观的、定性的信息的专家。人类很容易有偏差，所以选择哪一个不确定性管理技术很大程度上依赖于

现有的领域专家。

当存在可靠的统计数据，知识工程师能够主持大局，也有能参与严密的决策分析对话的专家时，贝叶斯方法应该是最适合的。这3个条件缺少任何一个，使用贝叶斯方法就显得太武断，也会由于偏差而不能得到有价值的结果。还需提及的是，贝叶斯信念传播的复杂度是指数级的，不适用于大型知识库。

虽然缺乏形式基础，但是确信因子技术为专家系统处理不确定性提供了一个简单的方法，在许多应用中的结果也都是可以接受的。

3.9 小结

本章介绍了用于专家系统的两个不确定性管理技术：贝叶斯推理和确信因子。分析了不确定知识的主要来源，简要回顾了概率理论。我们讨论了可累积论据的贝叶斯方法，并基于贝叶斯方法开发了一个简单的专家系统。之后介绍了确信因子理论（贝叶斯推理的常用替代方法），并基于论据推理构建了一个专家系统。最后，将贝叶斯推理和确信因子进行了对比，分析了它们的适用范围。 83

本章的主要内容如下：

- 不确定性是由于缺乏帮助我们得出完美可靠结论的精确知识。专家系统中的不确定知识的主要来源是：弱暗示、不精确的语言、数据缺失、综合不同专家的观点。
- 概率理论为专家系统中的不确定性管理提供了一个精确的、数学校正的方法。在给定论据的前提下，我们可使用贝叶斯规则计算出一个假设的概率。
- 用于矿床勘探的专家系统 PROSPECTOR 是第一个成功应用贝叶斯规则传送不确定性的系统。
- 要使用贝叶斯方法，专家需要提供假设 H 的先验概率、充分性似然值 LS（衡量在给定论据 E 时假设的可信度）、必要性似然值（衡量在给定论据 E 时假设的不可信度）。贝叶斯方法使用下面的规则形式：

IF *E* is true {*LS, LN*}
THEN *H* is true {*prior probability*}

- 要想使用贝叶斯方法，论据间必须满足条件独立，还应该有可靠的统计数据，并为每个假设定义先验概率。在实际问题上，往往不能满足这些要求，所以只有少数系统可以使用贝叶斯推理。
- 确信因子理论是贝叶斯方法的常用替代方法。这一理论的基本原理首次出现在诊断医疗专家系统 MYCIN 中。
- 确信因子理论为专家系统中的不确定管理提供了一个判断方法。使用这一方法时，需要专家提供确信因子 *cf* 的值，用来表示给定论据 E 时假设 H 的可信度水平。确信因子方法使用如下规则：

IF *E* is true
THEN *H* is true {*cf*}

- 当概率未知或不易获得时，会使用确信因子。确定理论可以增量获得的管理论据、合取式假设、析取式假设，以及具有不同可信度的论据。 84
- 贝叶斯推理和确定理论面临的共同问题是：需要一个能够量化主观信息和定性信息的专家。

复习题

1. 什么是不确定性？什么情况下知识可以不精确，数据可以不完整、不一致？举例说明不精确的知识。

2. 什么是概率？用数学方式描述在事件 *B* 发生时事件 *A* 发生的条件概率。什么是贝叶斯规则？
3. 什么是贝叶斯推理？专家系统怎样对可能的真假设进行排序？举例说明。
4. 为什么 PROSPECTOR 团队能够将贝叶斯方法作为不确定性管理技术？使用贝叶斯推理必须要具备哪些条件？
5. 什么是充分性似然值和必要性似然值？专家如何确定 *LS* 和 *LN* 的值？
6. 什么是先验概率？举例说明基于贝叶斯推理的专家系统的规则表达方式。
7. 基于规则的专家系统如何使用贝叶斯方法传送不确定性？
8. 条件概率为什么也许会与专家提供的先验概率不一致？举例说明。
9. 为什么确信因子理论可被看做贝叶斯推理的实用替代方法？什么是可信度和不可信度的度量？定义一个确信因子。
10. 专家系统如何确定“与”、“或”规则的净确定性？举例说明。
11. 专家系统如何合并影响同一个假设的多个规则的确信因子？举例说明。
12. 请比较贝叶斯推理和确信因子。它们分别适合什么样的应用场景？为什么？这两个方法的共同问题是什么？

参考文献

Bhatnagar, R.K. and Kanal, L.N. (1986). Handling uncertain information: a review of numeric and non-numeric methods, *Uncertainty in AI*, L.N. Kanal and J.F. Lemmer, eds, Elsevier North-Holland, New York, pp. 3–26.

Bonissone, P.P. and Tong, R.M. (1985). Reasoning with uncertainty in expert systems, *International Journal on Man–Machine Studies*, 22(3), 241–250.

85

Burns, M. and Pearl, J. (1981). Causal and diagnostic inferences: a comparison of validity, *Organizational Behaviour and Human Performance*, 28, 379–394.

Campbell, A.N., Hollister, V.F., Duda, R.O. and Hart, P.E. (1982). Recognition of a hidden mineral deposit by an artificial intelligence program, *Science*, 217(3), 927–929.

Duda, R.O., Hart, P.E. and Nilsson, N.L. (1976). Subjective Bayesian methods for a rule-based inference system, *Proceedings of the National Computer Conference (AFIPS)*, vol. 45, pp. 1075–1082.

Duda, R.O., Gaschnig, J. and Hart, P.E. (1979). Model design in the PROSPECTOR consultant system for mineral exploration, *Expert Systems in the Microelectronic Age*, D. Michie, ed., Edinburgh University Press, Edinburgh, Scotland, pp. 153–167.

Durkin, J. (1994). *Expert Systems: Design and Development*. Prentice Hall, Englewood Cliffs, NJ.

Feller, W. (1966). *An Introduction to Probability Theory and its Applications*, vol. 2, John Wiley, New York.

Feller, W. (1968). *An Introduction to Probability Theory and its Applications*, vol. 1, 3rd edn. John Wiley, New York.

Fine, T.L. (1973). *Theories of Probability: An Examination of Foundations*. Academic Press, New York.

Firebaugh, M.W. (1989). *Artificial Intelligence: A Knowledge-based Approach*. PWS-KENT Publishing Company, Boston, MA.

Good, I.J. (1959). Kinds of probability, *Science*, 129(3347), 443–447.

Hakel, M.D. (1968). How often is often? *American Psychologist*, no. 23, 533–534.

Naylor, C. (1987). *Build Your Own Expert System*. Sigma Press, Wilmslow, Cheshire.

Ng, K.-C. and Abramson, B. (1990). Uncertainty management in expert systems, *IEEE Expert*, 5(2), 29–47.

Shortliffe, E.H. and Buchanan, B.G. (1975). A model of inexact reasoning in medicine, *Mathematical Biosciences*, 23(3/4), 351–379.

Simpson, R. (1944). The specific meanings of certain terms indicating differing degrees of frequency, *The Quarterly Journal of Speech*, no. 30, 328–330.

Stephanou, H.E. and Sage, A.P. (1987). Perspectives on imperfect information processing, *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17(5), 780–798.

Tversky, A. and Kahneman, D. (1982). Causal schemes in judgements under uncertainty, *Judgements Under Uncertainty: Heuristics and Biases*, D. Kahneman, P. Slovic and A. Tversky, eds, Cambridge University Press, New York.

86

第4章
Artificial Intelligence, 3E

模糊专家系统

本章描述模糊集理论，以及如何建立模糊专家系统，并通过实例来说明模糊集理论。

4.1 概述

专家解决问题时通常会用到常识，但他们也会用到含糊和模棱两可的语言。例如，专家可能会说："电源变压器已经微超载，但还能再坚持一会儿"。其他的专家能够没有任何困难地理解和解释这句话，因为他们都听到过以这种方式描述的问题。但是，知识工程师要让计算机达到相同的理解水平就十分困难。那么怎样在计算机中表达专家使用的含糊和模棱两可的语言描述知识的呢？能做得到吗？

通过研究模糊集理论（或模糊逻辑），本章将尝试回答这些问题。本章首先回顾模糊逻辑的哲学概念，研究它的机制并考虑如何在模糊专家系统中使用模糊逻辑。

本章先从浅显的，但是最基础和本质的命题开始："模糊逻辑并不是说逻辑本身是模糊的，而是指用来描述模糊的逻辑。"模糊逻辑是模糊集的理论，模糊集能够校准含糊的知识。模糊逻辑的基本思想是任何事情都允许有一定的程度。温度、高度、速度、距离和美丽——所有这些都可以在某个范围内浮动。发动机运转起来真的很热。Tom 是个很高的家伙。电车跑得不快。高性能发动机需要快速的动力装置和精确控制。霍巴特和墨尔本之间的路程很短。悉尼是个美丽的城市。这种可变化的尺度很难把某类的成员和非成员区别开。比如，一个小山丘变大到什么时候可以被称为山峰？

布尔或传统的逻辑使用明显的区别，它迫使我们在成员和非成员之间划出明显的界限。例如，可以说："电动交通工具的最大行程很短"，这个判断是基于行程在 300km 以下就是短，超过 300km 才是长这个规定做出的。按照这个标准，只有行程超过 301km（或者 300km 零 500m，甚至是 300km 零 1m）的电动交通工具才可被认为是长距离的。类似地，如果我们以 180cm 为界限，那么就说 Tom 很高，因为其身高为 181cm，David 很矮，因为其身高为 179cm，但是 David 真的很矮吗？这种分界可以这么武断吗？模糊逻辑可以避免发生这样武断的判断。

模糊逻辑能反映人类是怎么样思考的，它尝试模拟人类的预感、决策制定和常识。结果，它促进了新的、更加人性化的和智能的系统的产生。

模糊或多值逻辑是波兰的逻辑学家和哲学家 Jan Lukasiewicz 在 20 世纪 30 年代引入的（Lukasiewicz，1930）。他研究"高"、"老了"和"热"这样的模糊语言的数学表示。当时经典的逻辑操作仅使用两个值 1（为真）和 0（为假），Lukasiewicz 引入了将逻辑真值扩展到 0 和 1 之间所有的实数。使用该范围内的一个数值来表示某个命题为真或为假的可能性。例如，身高为 181cm 的男人确实是高的可能性取值为 0.86，这个人应该是很高。不精确推理技术产生的工作通常称为可能性理论。

1973 年，哲学家 Max Black 发表了论文"Vagueness: an exercise in logical analysis"（Black，1937）。在论文中，他讨论了指示程度的连续区。他说，设想将无数的椅子排成一行，在一端是齐本德尔式的椅子，挨着它的是类似齐本德尔式的，但看上去和第一把椅子几乎没有差别，随后的椅子越来越不像椅子，最后是一根圆木。椅子什么时候变成了圆木？椅子这个概念让我们无法在椅子和非椅子之间画出明显的界限。Max Black 也定义如果连续区是离散的，那么可以为每个元素分配一个数值。这个数值代表一种程度。问题是它是什么的程度？Black 用数值来显示"认为一排'椅子'中的某一个元素能够称为椅子的人的百分比"。换句话说，即接受模糊的概率。

但是，Black 最重要的贡献是在该论文的附录中。在附录中他定义了第一个简单的模糊集，并概述了模糊集操作的基本思想。

1965 年，加州大学伯克利分校电子工程系主任 Lotfi Zadeh 教授发表了著名的论文 "Fuzzy sets"。事实上，Zadeh 是重新发现了模糊论，确定并研究该理论，捍卫它并为它而战。

Zadeh 将可能性理论扩展到数学逻辑的形式系统中，更重要的是，他引入了新概念以应用自然语言的术语。这种表达和操作模糊术语的新逻辑称为模糊逻辑，Zadeh 也成为 "模糊逻辑之父"。

为什么模糊

就像 Zadeh 所说，术语是具体、直接和可叙述的，我们都知道这是什么意思。但是在西方，许多人都抵制 "模糊" 这个词，因为它经常用在贬义的场合。 88

为什么要有逻辑

模糊依赖模糊集理论，模糊逻辑只是该理论的一小部分。但是 Zadeh 在更广泛的意义上使用 "模糊逻辑" 这个术语（Zadeh，1965）：

模糊逻辑的定义是：基于隶属度而不是经典二值逻辑中清晰隶属关系的知识表达的一组数学原理。

和二值的布尔逻辑不同，模糊逻辑是多值的。它处理隶属的程度和可信的程度。模糊逻辑使用介于 0（完全为假）和 1（完全为真）之间的逻辑值的连续区间。与非黑即白不同，它使用颜色的色谱，可以接受同时部分为真和部分为假的事物。如图 4.1 所示，模糊逻辑为布尔逻辑增加了一定范围的逻辑值。经典的二值逻辑可以看做是多值模糊逻辑的特例。



图 4.1 布尔逻辑和模糊逻辑的逻辑值范围

4.2 模糊集

集的概念是数学中的基本概念。但是，我们的语言是集的最大表达。例如，汽车指的是汽车的集合。当我们说到一辆车的时候，实际上是指所有汽车的某一个例。

假设 X 是经典（清晰）的集，x 是一个元素，那么，元素 x 要么属于 X（$x \in X$），要么不属于 X（$x \notin X$）。也就是说，经典的集理论划分了严格的界限，集中的每个成员赋值为 1，不在集中的每个成员赋值为 0。这就是二分法原理。下面我们就来探讨这个原理。

考虑下面经典的逻辑悖论。

（1）毕达哥拉斯的学校（公元前 400 年）

问题：当特里克岛的哲学家断言 "所有的特里克岛人都说谎" 时，这句话是真的吗？

布尔逻辑：这个断言自相矛盾。

模糊逻辑：哲学家可能有、也可能没有说真话。

（2）拉塞尔悖论

村子里的理发师只给那些不能给自己理发的人理发。 89

问题：谁给理发师理发？

布尔逻辑：这个断言自相矛盾。

模糊逻辑：理发师可以给、也可以不给自己理发。

清晰集理论由仅使用两个值（真或假）中的一个值的逻辑支配。这个逻辑不能表达含糊的概念，因此在悖论中无法给出答案。模糊集理论的基本观点是属于模糊集的元素具有确定的隶

属度。因此，命题既不是真也不是假，而是在任何程度上部分为真（或部分为假），其中程度可以取［0，1］区间的实数。

模糊集理论的经典例子是高个子的男人。模糊集“高个子男人”中的元素是全体男性，但他们的隶属度取决于他们的身高，如表4.1所示。例如，给身高为205cm的Mark赋予程度1，给身高为152cm高的Peter赋予程度0，所有身高在152～205cm之间的人取0～1之间的值。他们的高是带有程度的。显然，不同的人对某个人是否可称为高有不同的看法。候选人隶属度见表4.1。

表4.1 “高个子男人”的隶属度

姓名	身高（cm）	隶属度	
		清晰的	模糊的
Chris	208	1	1.00
Mark	205	1	1.00
John	198	1	0.98
Tom	181	1	0.82
David	179	0	0.78
Mike	172	0	0.24
Bob	167	0	0.15
Steven	158	0	0.06
Bill	155	0	0.01
Peter	152	0	0.00

可以看到，清晰集问了一个问题，“那个男人高吗?”，并且在180cm处画了一条线，比180cm高的人就是高个子男人，比它低的就是矮个子男人。相比之下，模糊集的问题是，“这个男人有多高?”，回答是模糊集中的部分成员资格，例如，Tom高的程度是0.82。

模糊集提供跨越边界时平稳过渡的能力，如图4.2所示。

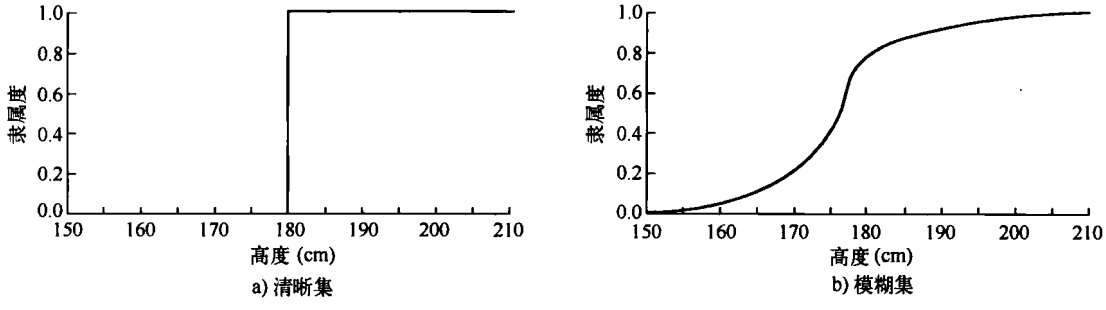


图4.2 “高个子男人”的清晰集和模糊集

考虑一下其他的集合例如“很矮的男人”、“矮男人”、“中等身高的男人”和“很高的男人”。

图4.2的横轴表示论域——变量所有可能取值的范围，在本例中变量是身高。按照这种表示方法，男性的身高应该包含全体男性的身高。但是，通常还有考虑的余地，因为论域在不同的上下文中有不同的含义。例如，“高个子男人”可能是全体人类或哺乳动物，甚至是全体动物身高论域中的一部分。

图4.2的纵轴表示模糊集中的隶属度。在本例中，“高个子男人”的模糊集将身高值映射到相应的隶属值。在图4.2中，David的身高为179cm，仅比Tom低2cm，不再突然地成为不高的人（或矮的人，在清晰集中他是矮的人）。在这里，David和其他人按照身高的不断降低，逐渐地从“高个子男人”的集合中移出。

什么是模糊集

模糊集可以简单地定义为具有模糊边界的集合。

假设 X 为论域，其中的元素可记为 x。在经典的集合论中，X 的清晰集 A 定义为函数 $f_A(x)$，称为 A 的特征函数：

$$f_A(x): X \rightarrow \{0,1\} \tag{4.1}$$

其中：

$$f_A(x) = \begin{cases} 1, & \text{若 } x \in A \\ 0, & \text{若 } x \notin A \end{cases}$$

90
91

该集合将 X 的论域映射到两个元素。对于论域 X 的任何元素 x，如果 x 是集合 A 中的元素，特征函数 $f_A(x)$ 等于 1，如果 x 不是 A 中的元素，则特征函数 $f_A(x)$ 等于 0。

在模糊论中，论域 X 的模糊集 A 定义为函数 $\mu_A(x)$，称为集合 A 的隶属函数：

$$\mu_A(x): X \rightarrow [0,1] \tag{4.2}$$

其中：

如果 x 完全在集合 A 中，则 $\mu_A(x) = 1$。

如果 x 不在集合 A 中，则 $\mu_A(x) = 0$。

如果 x 部分在集合 A 中，则 $0 < \mu_A(x) < 1$。

该集合允许使用可能选择的连续取值。对于论域 X 中的任何元素 x，隶属函数 $\mu_A(x)$ 等于 x 是集合 A 中元素的程度，该程度的取值为 0 - 1，表示隶属度，也称作集合 A 中元素 x 的隶属值。

在计算机中如何表达模糊集

首先必须定义隶属函数。这时可以应用从知识获取中学到的一些方法。例如，形成模糊集的最实用的方法是依赖某一位专家的知识，询问专家这些元素是否属于给定的集合。另一个有用的方法是从多位专家那里获取知识。最近还出现了一种形成模糊集的新技术，其基于人工神经网络，可以学习可用的系统运作数据并自动生成模糊集。

现在回到“高个子男人”的例子。获取了男性身高的知识后，以产生高个子男人的模糊集。用类似的方式，我们也可以得到矮个子男人和中等身高男人的模糊集。这些模糊集合清晰集如图 4.3 所示。我们讨论的（男性身高）论域包含 3 个集：short men（矮个子男人）、average men

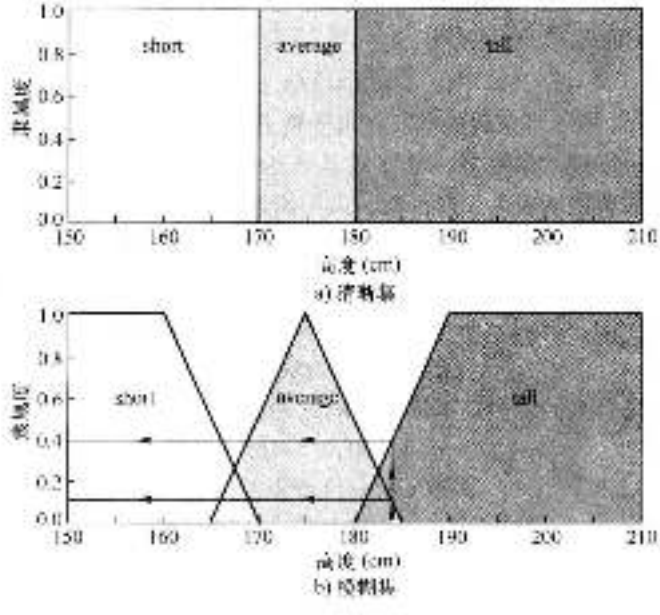


图 4.3 矮个子、中等身高和高个子男人的清晰集和模糊集

(中等身高男人) 和 tall men (高个子男人)。如你所见，在模糊逻辑中，身高为 184cm 的男人是 average men 集的成员，隶属度为0.1，同时他也是 tall men 集的成员，隶属度为 0.4。也就是说，身高为 184cm 的男人可以部分地属于多个集。

假设 X 的论域（也称作参考超集）是包含 5 个元素的清晰集 $X=\{x_1, x_2, x_3, x_4, x_5\}$。设 A 为 X 的清晰子集，仅包含两个元素，即 $A=\{x_2, x_3\}$。子集 A 可以表示为 $A=\{(x_1, 0), (x_2, 1), (x_3, 1), (x_4, 0), (x_5, 0)\}$，即为 $\{(x, \mu_A(x))\}$ 对的集合，其中 $\mu_A(x)$ 为子集 A 中元素 x 的隶属函数。

问题是 $\mu_A(x)$ 只能取两个值，非 0 即 1，还是取 0 和 1 之间的任何数都可以。这是 Lotfi Zadeh 在 1965 年提出的模糊集中最基本的问题（Zadeh，1965）。

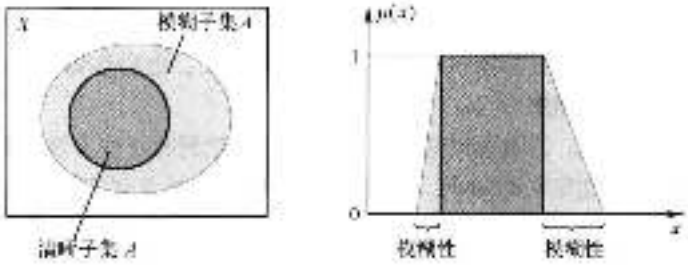
如果 X 是参考超集，A 是 X 的子集，那么当且仅当

$$A=\{x, \mu_A(x)\} \quad x\in X, \mu_A(x): X\rightarrow[0,1] \tag{4.3}$$

A 称为 X 的模糊子集。

在特例中，用 $X\rightarrow\{0, 1\}$ 代替 $X\rightarrow[0, 1]$，模糊子集 A 就变成了清晰子集 A。

模糊子集和清晰子集如图 4.4 所示。



92
93

图 4.4 X 的清晰子集和模糊子集的表示

有限参考超集 X 的模糊子集 A 可表示为

$$A=\{x_1, \mu_A(x_1)\}, \{x_2, \mu_A(x_2)\}, \cdots, \{x_n, \mu_A(x_n)\} \tag{4.4}$$

但是，A 表示为下面的式子更简捷：

$$A=\{\mu_A(x_1)/x_1\}, \{\mu_A(x_2)/x_2\}, \cdots, \{\mu_A(x_n)/x_n\}, \tag{4.5}$$

其中，分割符 "/" 用于将隶属值和其在水平轴上的坐标关联起来。

要在计算机中表示连续的模糊集，需要将其表达为函数，然后将集合中的元素映射为它们的隶属度。可以使用的典型函数为 S 形函数、高斯函数和 pi。这些函数可表示模糊集中的真实数据，但是这样增加了计算的时间。因此，实际上大多数应用都使用线性拟合函数，其类似于图 4.3使用的函数。例如，图 4.3 中的高个子男人的模糊集可以表示为拟合向量：

$$tall\ men=(0/180, 0.5/185, 1/190) \text{或者}$$

$$tall\ men=(0/180, 1/190)$$

矮个子和中等身高男人的模糊集可以用相同方式表示：

$$short\ men=(1/160, 0.5/165, 0/170) \text{或者}$$

$$short\ men=(1/160, 0/170)$$

$$average\ men=(0/165, 1/175, 0/185)$$

4.3 语言变量和模糊限制语

模糊集理论源自语言变量的概念。语言变量是模糊变量。例如，"John 很高" 这句话意味着

语言变量 John 取值为语言值“高”。在模糊专家系统中，语言变量在模糊规则中使用。例如：

```
IF      wind is strong
THEN    sailing is good

IF      project_duration is long
THEN    completion_risk is high

IF      speed is slow
THEN    stopping_distance is short
```

语言变量的可能值的范围表示变量的论域。例如，语言变量“速度”的全域为 0 – 220km/h，包含的模糊子集有 very slow、slow、medium、fast 和 very fast。每个模糊子集还表示相应语言变量的语言值。 | 94

模糊变量带有模糊集限制语概念，称作模糊限制语。模糊限制语是可以修改模糊集形状的术语，包含 very、somewhat、quite、more or less 和 lightly 这样的副词。模糊限制语可以修饰动词、形容词、副词甚至于整个句子。模糊限制语可用作：

- 通用修饰符，例如 very、quite 或 extremely。
- 真值，例如 quite true 或 mostly false。
- 概率，例如 likely 或 not very likely。
- 量词，例如 most、several 或 few。
- 可能性，例如 almost impossible 或 quite possible。

模糊限制语可作为它们自己的操作。例如，very 执行集中操作并创建一个新的子集。从 tall men 集合中派生一个 very tall men 子集，extremely 的作用与此相同，只是程度更高。

和集中相反的操作是扩张。它将集扩展。more or less 执行扩张。例如，more or less tall men 这个集合的范围比 tall men 集合的范围更大。

模糊限制语利于操作，也可将连续空间分解成模糊区间。例如，可以使用下面的模糊限制语描述温度：very cold、moderately cold、slightly cold、neutral、slightly hot、moderately hot 和 very hot。很明显，这些模糊集是相互重叠的。模糊限制语言在助于反映人们的思维，因为人们通常无法区分 slightly hot 和 moderately hot。

模糊限制语的应用如图 4.5 所示。先前在图 4.3 中显示的模糊集用模糊限制语 very 做了数学上的改进。例如，一个身高为 185cm 的男性，他属于 tall men 集，隶属度是 0.5，但他同时也是 very tall men 集的成员，隶属度是 0.15，这就更加合理。

下面考虑在实际应用中经常用到的模糊限制语。

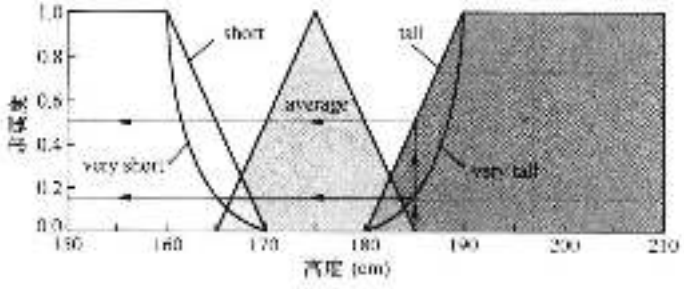


图 4.5 带有模糊限制语“very”的模糊集 | 95

- very（很），即上文提到的集中操作，将一个集合缩小，降低模糊元素的隶属度。这个操作可用数学中的平方来表示：

$$\mu_A^{very}(x) = [\mu_A(x)]^2 \tag{4.6}$$

因此，如果 Tom 在 tall men 集中的隶属度是 0.86，那么他在 very tall men 集中的隶属度是 0.7396。

- extremely（非常），和“very”的作用相同，但是程度更深。该操作可以表示成函数的 $\mu_A(x)$ 三次方：

$$\mu_A^{extremely}(x) = [\mu_A(x)]^3 \tag{4.7}$$

如果 Tom 在 tall men 集中的隶属度是 0.86，那么他在 very tall men 集中的隶属度是 0.7396，在 extremely tall men 集中的隶属度是 0.6361。

- very very（太），是集中操作的扩展。它可以表达成集中操作的平方。

$$\mu_A^{very\ very}(x) = [\mu_A^{very}(x)]^2 = [\mu_A(x)]^4 \tag{4.8}$$

例如，如果 Tom 在 tall men 集中的隶属度是 0.86，在 very tall men 集中的隶属度是 0.7396，那么他在 very very tall men 集中的隶属度是 0.5470。

- more or less（或多或少），属于扩展操作，它将集扩展并增加模糊元素的隶属度。该操作可以表达为：

$$\mu_A^{more\ or\ less}(x) = \sqrt{\mu_A(x)} \tag{4.9}$$

因此，如果 Tom 在 tall men 集中的隶属度是 0.86，那么他在 more or less tall men 集中的隶属度是 0.9274。

- indeed（的确），是增强操作，强化整个句子的含义。通过增加大于 0.5 的隶属度和减少小于 0.5 的隶属度来达到这一目的。Indeed 模糊限制语可以通过以下两种方式给出：

$$\mu_A^{indeed}(x) = 2[\mu_A(x)]^2 \qquad 若\ 0 \leqslant \mu_A(x) \leqslant 0.5 \tag{4.10}$$

或者：

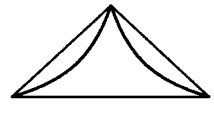
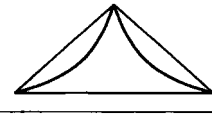
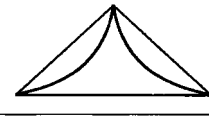
$$\mu_A^{indeed}(x) = 1 - 2[1 - \mu_A(x)]^2 \qquad 若\ 0.5 < \mu_A(x) \leqslant 1 \tag{4.11}$$

如果 Tom 在 tall men 集中的隶属度是 0.86，那么他在 indeed tall men 集中的隶属度是 0.9608。相比之下，如果 Mike 在 tall men 集中的隶属度是 0.24，那么他在 indeed tall men 集中的隶属度是 0.1152。

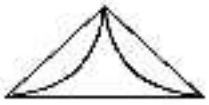
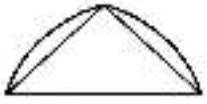
96

模糊限制语的数学和图例的表示如表 4.2 所示。

表 4.2 模糊逻辑中模糊限制语的表示

模糊限制语	数学表示	图例表示
a little	$[\mu_A(x)]^{1.3}$	
slightly	$[\mu_A(x)]^{1.7}$	
cery	$[\mu_A(x)]^2$	
extremely	$[\mu_A(x)]^3$	

（续）

模糊限制词	数学表示	图例表示
very very	$[\mu_A(x)]^4$	
more or less	$\sqrt{\mu_A(x)}$	
somewhat	$\sqrt{\mu_A(x)}$	
indeed	$2[\mu_A(x)]^2$ 若 $0 \leqslant \mu_A \leqslant 0.5$ $1-2[1-\mu_A(x)]^2$ 若 $0.5 \leqslant \mu_A \leqslant 1$	

4.4 模糊集的操作

在19世纪后期，由 Georg Cantor 开发的经典的集合论描述了清晰集是如何相互作用的，这些相互作用称为操作（operation）。

我们来看4个操作：补、包含、交和并。这些操作如图4.6所示。下面比较经典集合模糊集中的操作。

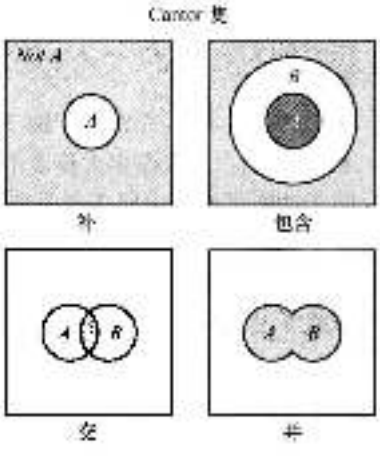


图4.6 经典的集操作

补

- 清晰集：谁不属于集？
- 模糊集：元素不属于集的程度是多少？

集的补集是集的相反操作。例如，有一个 tall men 集，它的补集是 NOT tall men 集。当我们从论域中移除 tall men 集后，就得到补集。如果 A 是模糊集，其补集 $\neg A$ 为：

$$\mu_{\neg A}(x) = 1 - \mu_A(x) \tag{4.12}$$

例如，如果有一个 tall men 模糊集，那么可以很容易得到 NOT tall men 模糊集：

$$tall\ men = (0/180,\ 0.25/182.5,\ 0.5/185,\ 0.75/187.5,\ 1/190)$$

$$NOT\ tall\ men = (1/180,\ 0.75/182.5,\ 0.5/185,\ 0.25/187.5,\ 0/190)$$

包含

- 清晰集：哪一些集包含在其他集中？
- 模糊集：哪一些集属于其他集？

类似于中国盒子或俄罗斯套娃，一个集可以包含另一个集。较小的集称作子集。例如，tall men 集包含所有高个子男人，因此 very tall men 集是 tall men 集的子集。但是 tall men 集是 men 集的子集。在清晰集中，子集中所有元素都属于更大的集，其隶属度为1。但是在模糊集中，每个元素属于子集的程度比属于更大的集的程度低。模糊子集的元素在子集中的隶属值比在更大的

集中的隶属值小。

$$tall\ men = (0/180,\ 0.25/182.5,\ 0.50/185,\ 0.75/187.5,\ 1/190)$$

$$very\ tall\ men = (0/180,\ 0.06/182.5,\ 0.25/185,\ 0.56/187.5,\ 1/190)$$

交

- 清晰集：哪些元素同时属于两个集？
- 模糊集：元素同时属于两个集的程度是多少？

在经典的集合论中，两个集的“交”操作包含两个集中都有的元素。例如，有 tall men 和 fat men 两个集，交就是两个集重叠的部分，即 Tom 属于相交的部分，因为他又高又胖。但是在模糊集中，元素可能是部分地属于两个集，对于两个集而言隶属度也不同，因此，模糊“交”操作中的元素在每个集中的隶属度都比较低。

论域 X 上创建模糊集 A 和 B 的交的模糊操作为：

$$\mu_{A\cap B}(x) = min[\mu_A(x), \mu_B(x)] = \mu_A(x) \cap \mu_B(x), \quad 其中\ x \in X \tag{4.13}$$

例如，tall men 和 average men 模糊集为：

$$tall\ men = (0/165,\ 0/175,\ 0.0/180,\ 0.25/182.5,\ 0.5/185,\ 1/190)$$

$$average\ men = (0/165,\ 1/175,\ 0.5/180,\ 0.25/182.5,\ 0.0/185,\ 0/190)$$

根据公式（4.13），这两个集的交是

$$tall\ men \cap average\ men = (0/165,\ 0/175,\ 0/180,\ 0.25/182.5,\ 0/185,\ 0/190)$$

或

$$tall\ men \cap average\ men = (0/180,\ 0.25/182.5,\ 0/185)$$

99 结果如图 4.3 所示。

并

- 清晰集：哪些元素属于两个集中的任意一个？
- 模糊集：元素在多大程度上属于两个集中的任意一个？

两个清晰集的并由属于两个集的所有元素组成。例如，tall men 和 fat men 的并包含所有高或胖的人。例如，Tom 在并集中，因为他高，他是否胖无关紧要。在模糊集中，并是交的逆操作，也就是说，并由两个集中隶属值较大的元素组成。

形成 X 的论域上模糊集 A 和 B 并的模糊操作为：

$$\mu_{A\cup B}(x) = max[\mu_A(x), \mu_B(x)] = \mu_A(x) \cup \mu_B(x), \quad 其中\ x \in X \tag{4.14}$$

再次考虑 tall men 和 average men 模糊集：

$$tall\ men = (0/165,\ 0/175,\ 0.0/180,\ 0.25/182.5,\ 0.5/185,\ 1/190)$$

$$average\ men = (0/165,\ 1/175,\ 0.5/180,\ 0.25/182.5,\ 0.0/185,\ 0/190)$$

根据公式（4.14），这两个集的并是：

$$tall\ men \cup average\ men = (0/165,\ 1/175,\ 0.5/180,\ 0.25/182.5,\ 0.5/185,\ 1/190)$$

模糊集的操作如图 4.7 所示。

清晰集和模糊集有相同的性质，清晰集可看做模糊集的特例。模糊集经常使用以下性质：

交换性

$A \cup B = B \cup A$

$A \cap B = B \cap A$

例如：

tall men OR *short men* = *short men* OR *tall men*
tall men AND *short men* = *short men* AND *tall men*

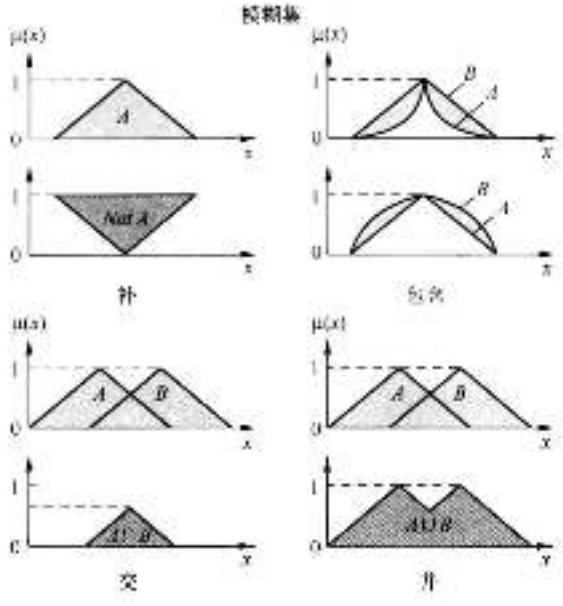


图 4.7 模糊集的操作

结合性

$$A\cup(B\cup C) = (A\cup B)\cup C$$
$$A\cap(B\cap C) = (A\cap B)\cap C$$

例如：

tall men OR (*short men* OR *average men*) = (*tall men* OR *short men*) OR *average men*
tall men AND (*short men* AND *average men*) = (*tall men* AND *short men*) AND *average men*

分配性

$$A\cup(B\cap C) = (A\cup B)\cap(A\cup C)$$
$$A\cap(B\cup C) = (A\cap B)\cup(A\cap C)$$

例如：

tall men OR (*short men* AND *average men*) = (*tall men* OR *short men*) AND (*tall men* OR *average men*)
tall men AND (*short men* OR *average men*) = (*tall men* AND *short men*) OR (*tall men* AND *average men*)

幂等性

$$A\cup A = A$$
$$A\cap A = A$$

例如：

tall men OR *tall men* = *tall men*
tall men AND *tall men* = *tall men*

恒等性

$$A\cup\phi = A$$
$$A\cap X = A$$

$$A \cap \phi = \phi$$
$$A \cup X = X$$

例如：

tall men OR *undefined* = *tall men*
tall men AND *unknown* = *tall men*
tall men AND *undefined* = *undefined*
tall men OR *unknown* = *unknown*

其中，undefined 是空集，即集中元素的隶属度都为 0，unknown 是所有元素的隶属度都为 1 的集。

自乘性

$$\neg(\neg A) = A$$

例如：

NOT (NOT *tall men*) = *tall men*

传递性

若$(A \subset B) \cap (B \subset C)$，则$A \subset C$。

每个集包含它的子集的子集。

例如：

IF (*extremely tall men* ⊂ *very tall men*) AND (*very tall men* ⊂ *tall men*)
THEN (*extremely tall men* ⊂ *tall men*)

德·摩根定律

$$\neg(A \cap B) = \neg A \cup \neg B$$
$$\neg(A \cup B) = \neg A \cap \neg B$$

102

例如：

NOT (*tall men* AND *short men*) = NOT *tall men* OR NOT *short men*
NOT (*tall men* OR *short men*) = NOT *tall men* AND NOT *short men*

使用模糊集操作、它们的性质和模糊限制语，可以很容易地从现有模糊集中得到很多模糊集。例如，现有模糊集 A(tall men)和模糊集 B(short men)，则下面的操作可以生成模糊集 C(not very tall men and not very short men)，甚至模糊集 D(not very very tall and not very very short men)：

$$\mu_C(x) = [1 - \mu_A(x)^2] \cap [1 - \mu_B(x)^2]$$
$$\mu_D(x) = [1 - \mu_A(x)^4] \cap [1 - \mu_B(x)^4]$$

通常，使用模糊操作和模糊限制语可以得到人类自然语言表达的模糊集。

4.5 模糊规则

1973 年，Lotfi Zadeh 发表了他的第二篇有广泛影响力的论文(Zadeh, 1973)。该论文概述了分析复杂系统的新方法，在文中 Zadeh 建议以模糊规则获取人类的知识。

什么是模糊规则

模糊规则可定义为以下形式的条件语句：

IF　　x is A
THEN　y is B

其中，x 和 y 是语言变量；A 和 B 分别是由论域 X 和 Y 上的模糊集定义的语言值。

经典规则和模糊规则之间的区别是什么

经典 IF - THEN 规则使用二值逻辑，例如：

```
Rule: 1
IF      speed is > 100
THEN    stopping_distance is long

Rule: 2
IF      speed is < 40
THEN    stopping_distance is short
```

变量 speed 可取 0 ~ 220km/h 之间的任何数值，语言变量 stopping_distance 可取的值为 long 或 short。换句话说，经典的规则可以用布尔逻辑的非黑即白的语言来描述。但是也可以将上面的规则以模糊的形式描述： 103

```
Rule: 1
IF      speed is fast
THEN    stopping_distance is long

Rule: 2
IF      speed is slow
THEN    stopping_distance is short
```

在本例中，语言变量 speed 的范围（论域）为 0 ~ 220km/h。但这个范围包含模糊集，例如 slow、medium 和 fast。语言变量 stopping_distance 的论域是 0 ~ 300m，并可以包含 short、medium 和 long 这样的模糊集。这样模糊集就可以和模糊规则联系起来。

模糊专家系统合并规则，结果减少了至少 90% 的规则数量。

怎样使用模糊规则推理

模糊推理有两个不同的部分：评估规则的前项（规则的 IF 部分），并将结果应用到后项（规则的 THEN 部分）。

在经典的基于规则的系统中，如果规则的前项为真，那么后项也为真。在模糊系统中，前项是模糊语句，所有的规则在一定程度上被激发，换句话说，规则被部分激发，如果前项在某种程度上为真，那么后项在该程度上也为真。

例如，有两个模糊集 tall men 和 heavy men，分别如图 4.8 所示。

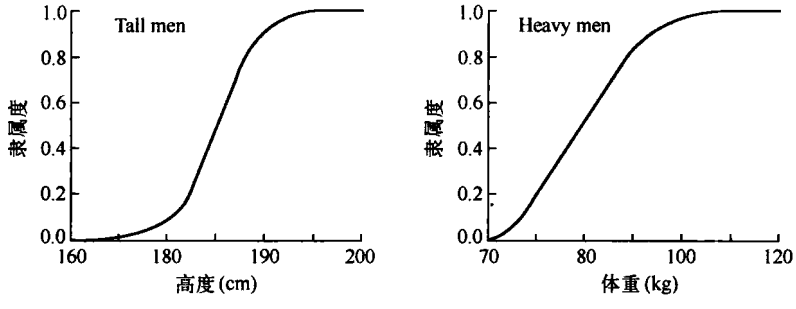


图 4.8 tall men 和 heavy men 模糊集 104

这些模糊集提供了体重评估模型的基础。模型是基于男人的身高和体重之间的关系的，可以用下面的模糊规则来表示：

```
IF      height is tall
THEN    weight is heavy
```

从前项中相应成员为真的程度可以估计后项的输出值或者成员为真的程度（Cox, 1999）。模糊推理的这种形式使用称作单调选择的方法。图 4.9 显示了如何通过男性的身高推导出男性的体重。

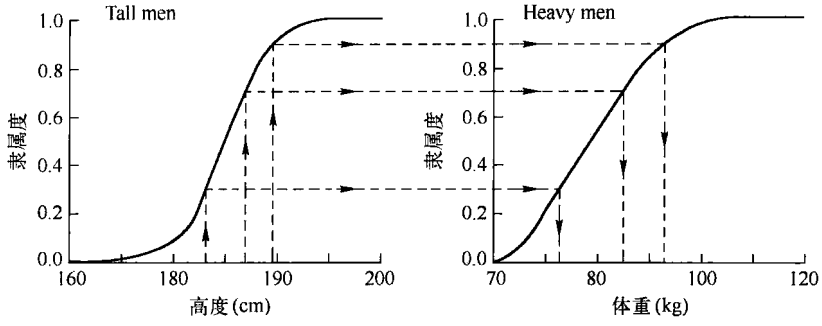


图 4.9 男性体重值的单调选择

模糊规则的前项可以有多个部分吗

模糊规则是产生式规则,因此可以有多个前项,例如:

```
IF      project_duration is long
AND     project_staffing is large
AND     project_funding is inadequate
THEN    risk is high

IF      service is excellent
OR      food is delicious
THEN    tip is generous
```

用前面章节讨论过的模糊集操作,可以同时计算规则前项的各个部分并得到单一的数值。

模糊规则的后项可以有多个部分吗

模糊规则的后项也可以包含多个部分,例如:

```
IF      temperature is hot
THEN    hot_water is reduced;
        cold_water is increased
```

105

本例中,后项的所有部分受前项的影响是相同的。

通常,模糊专家系统会集成描述专家知识的几个规则(相互之间可能会有矛盾)。每个规则的输出是一个模糊集,但通常需要得出一个数值来表示模糊系统的输出。换句话说,我们想要的是精确的而不是模糊的结论。

怎样将输出的模糊集结合并转换成一个单独的数值

为了得到输出变量单一清晰的结果,模糊专家系统首先将所有的输出模糊集聚集到一个模糊输出集中,然后将模糊集的结果逆模糊化为一个单独的数值。下一节将讨论这个完整的过程。

4.6 模糊推理

模糊推理的定义是:使用模糊集理论,将给定输入映射到输出的过程。

4.6.1 Mamdani-style 推理

模糊推理技术中最常用的方法是 Mamdani 方法。1975 年伦敦大学的 Ebrahim Mamdani 教授建立了第一个模糊系统来控制蒸汽机和锅炉(Mamdani and Assilian,1975)。他应用了一套有经验的人类操作员提供的模糊规则。

Mamdani-style 模糊推理过程按 4 个步骤执行:输入变量的模糊化、规则评估、聚合规则的输出以及最终的逆模糊化。

在这里用一个包含3个规则(rule)、两输入一输出的简单例子来说明各部分间是如何协同工作的:

Rule: 1		Rule: 1	
IF	*x* is *A*3	IF	*project_funding* is *adequate*
OR	*y* is *B*1	OR	*project_staffing* is *small*
THEN	*z* is *C*1	THEN	*risk* is *low*
Rule: 2		Rule: 2	
IF	*x* is *A*2	IF	*project_funding* is *marginal*
AND	*y* is *B*2	AND	*project_staffing* is *large*
THEN	*z* is *C*2	THEN	*risk* is *normal*
Rule: 3		Rule: 3	
IF	*x* is *A*1	IF	*project_funding* is *inadequate*
THEN	*z* is *C*3	THEN	*risk* is *high*

其中,x、y 和 z(project funding、project staffing 和 risk)为语言变量;$A1$、$A2$ 和 $A3$(inadequate、marginal 和 adequate)是由论域 X(project funding)上模糊集定义的语言值;$B1$ 和 $B2$(small 和 large)是由 106
论域 Y 上(project staffing)模糊集定义的语言值;$C1$、$C2$ 和 $C3$(low、normal、high)是由论域 Z(risk)上模糊集定义的语言值。

Mamdani-style 模糊推理的基本结构如图4.10所示。

步骤1: 模糊化。

此步骤是取得清晰的输入 $x1$ 和 $y1$(project funding 和 project staffing),确定每个输入属于每个合适模糊集的程度。

什么是清晰的输入? 如何确定

清晰的输入是指位于论域内的数值型的值。在本例中,$x1$ 和 $y1$ 的值都分别位于论域 X 和 Y 中。论域的范围可通过专家的判断来确定。例如,如果需要考虑开发"模糊"项目的风险,可以要求专家给出0~100%之间的值分别表示项目资金和项目员工。换句话说,专家需要回答怎样才是项目资金和项目员工足够的程度。当然,不同的模糊系统使用不同的清晰输入。其中有些输入是可以直接测量的(身高、体重、速度、距离、温度、压力等),而有些输入只能由专家估计。

一旦获得了清晰的输入 $x1$ 和 $y1$,就可以按照合适的语言模糊集进行模糊化。清晰的输入 $x1$(project funding,专家确定为35%)对应于隶属函数 $A1$ 和 $A2$(inadequate 和 marginal),隶属度分为0.5和0.2;清晰的输入 $y1$(project stuffing,专家确定为60%)映射到隶属函数 $B1$ 和 $B2$(small 和 large),隶属度分别为0.1和0.7.按照这种方式,每个输入按照模糊规则的隶属函数进行模糊化。

步骤2: 规则评估。

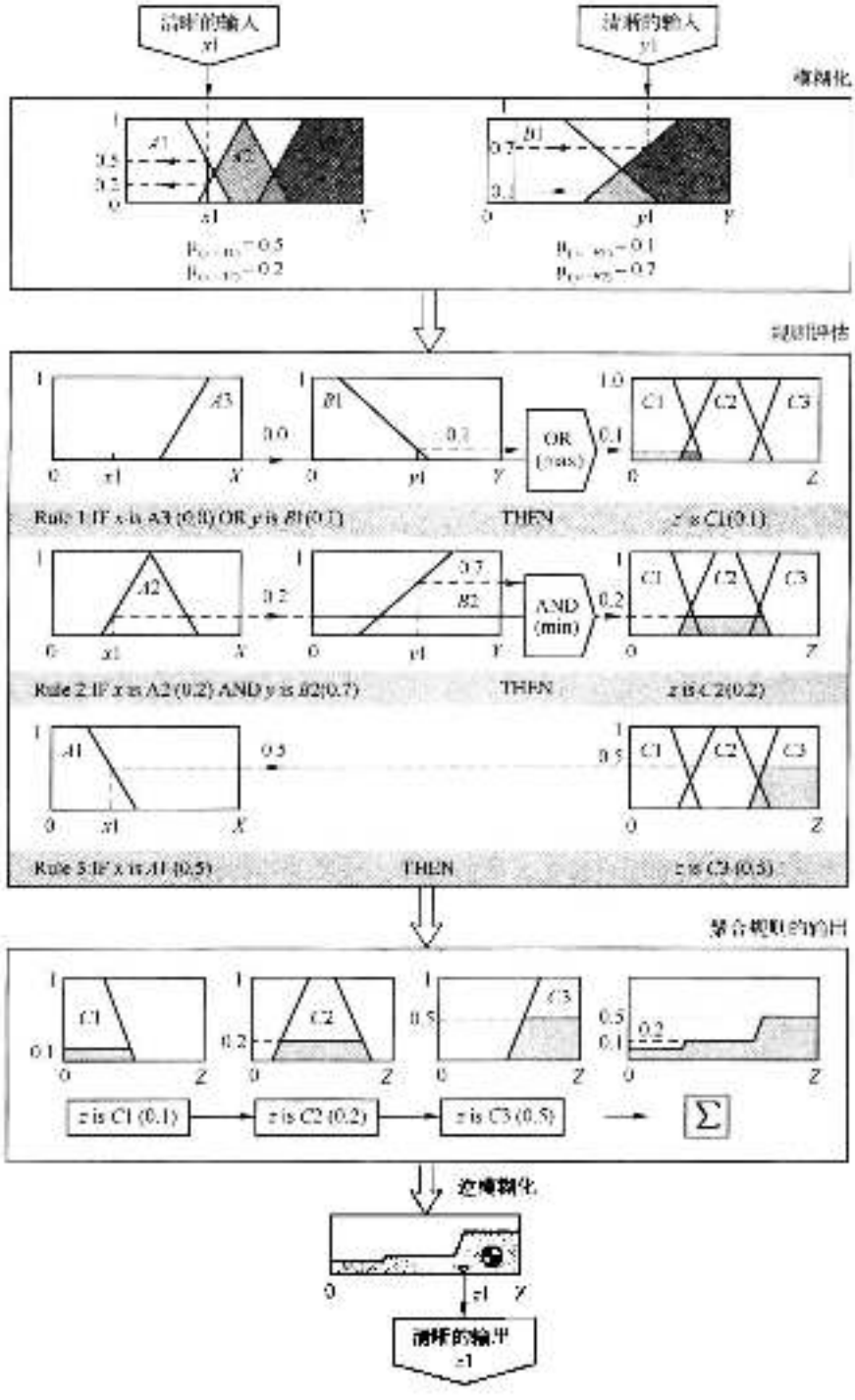
此步骤为取得模糊化后的输入,$\mu_{(x=A1)}=0.5$、$\mu_{(x=A2)}=0.2$、$\mu_{(y=B1)}=0.1$ 以及 $\mu_{(y=B2)}=0.7$,并将它们应用到模糊规则的前项。如果给定的模糊规则有多个前项,则使用模糊操作(AND 或 OR)来得到表示前项评估结果的一个数值。将这个数值(真值)接下来应用在后项隶属函数中。

为了评估规则前项的逻辑和,使用 OR 模糊操作。通常,模糊专家系统使用图4.10(规则1)所示的是经典模糊操作"并"(4.14):

$$\mu_{A\cup B}(x)=max[\mu_A(x),\mu_B(x)] \quad (4.14)$$ 107

但是,如果有必要,可以很容易地制定 OR 操作。例如,MATLAB Fuzzy Logic Toolbox 有两个内置的 OR 方法:max 和概率 OR 方法——probor 。概率 OR 方法也就是代数和,计算方法如下:

$$\mu_{A\cup B}(x)=probor[\mu_A(x),\mu_B(x)]=\mu_A(x)+\mu_B(x)-\mu_A(x)\times\mu_B(x) \quad (4.15)$$



108

图 4.10 Mamdani-style 模糊推理的基本结构

类似地，为了评估规则前项的“并”，应用如图 4.10（规则 2）所示的 AND 模糊操作“交”，见公式（4.13）：

$$\mu_{A\cap B}(x) = min[\mu_A(x), \mu_B(x)]$$

Fuzzy Logic Toolbox 也提供两种 AND 方法：min 和乘积方法 prod，乘积的计算方法是，

$$\mu_{A\cdot B}(x)=prod[\mu_A(x),\mu_B(x)]=\mu_A(x)\times\mu_B(x) \tag{4.16}$$

模糊操作的不同方法会产生不同的结果吗

模糊研究人员建议并应用不同的方法来执行 AND 和 OR 等模糊操作（Cox，1999），当然，不同的方法可能导致不同的结果。利用大多数模糊包也可以定制 AND 和 OR 等模糊操作，用户需要自己做出选择。

下面再看以下规则：

Rule: 1
IF x is A3 (0.0)
OR y is B1 (0.1)
THEN z is C1 (0.1)

$$\mu_{C1}(z)=max[\mu_{A3}(x),\mu_{B1}(y)]=max[0.0,0.1]=0.1$$

or

$$\mu_{C1}(z)=probor[\mu_{A3}(x),\mu_{B1}(y)]=0.0+0.1-0.0\times0.1=0.1$$

Rule: 2
IF x is A2 (0.2)
AND y is B2 (0.7)
THEN z is C2 (0.2)

$$\mu_{C2}(z)=min[\mu_{A2}(x),\mu_{B2}(y)]=min[0.2,0.7]=0.2$$

or

$$\mu_{C2}(z)=prod[\mu_{A2}(x),\mu_{B2}(y)]=0.2\times0.7=0.14$$

因此，规则 2 的表达如图 4.11 所示。

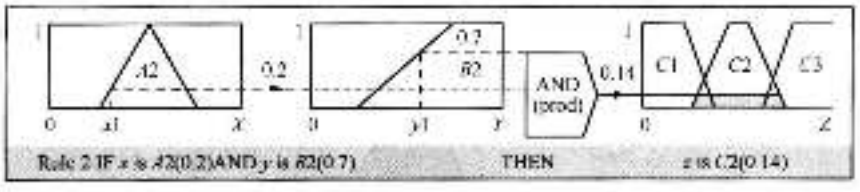


图4.11 AND 的产生式模糊操作

现在前项评估的结果可以应用到后项的隶属函数中。换句话说，后项隶属函数被剪切或缩放到规则前项的真值水平。

"剪切或缩放"是什么意思

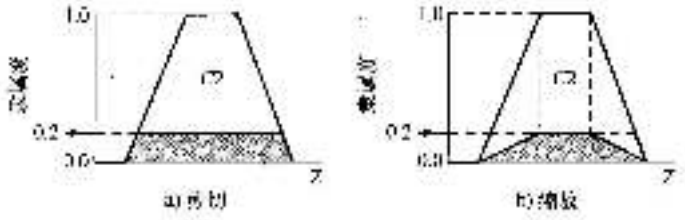
将规则后项和规则前项的真值关联起来最常见的方法是简单地裁减后项隶属函数，使之与前项的真值水准一致，这种方法也叫做剪切或最小相关性。由于隶属函数的顶层被分段，剪切后的模糊集损失了一部分信息。但是，剪切仍然是最常用的，因为它的数学复杂性低、运算速度快、并且能产生易于进模糊化的聚合输出表面。

剪切是最常使用的方法，而缩放或相关性产生式提供了保持模糊集原始形状的更好的方法。规则后项的原始隶属函数通过将其所有隶属度乘以规则前项的真值来调整。这种方法损失的信息较少，在模糊专家系统中非常有用。

剪切和缩放隶属函数如图 4.12 所示。

步骤 3：聚合规则的输出。

聚合是所有规则输出进行单一化的过程。换句话说，我们取之前经过剪切和缩放的所有规则后项的隶属函数并将它们合并到一个模糊集中。因此，聚合过程的输入是已经剪切或缩放后的后项隶属函数的列表，输出是每个输出变量分别有一个模糊集。图 4.10 显示了每个规则的输出是如何集成到代表整个模糊输出的单一的模糊集的。



a)剪切 b)缩放

图4.12 剪切和缩放的隶属函数

109
110

步骤4：逆模糊化。

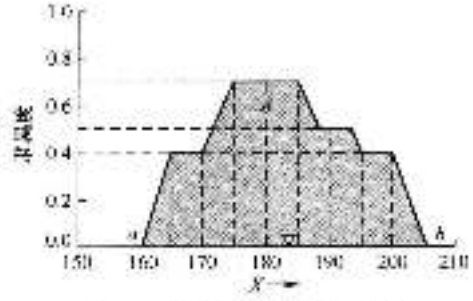
模糊推理过程的最后一个步骤是逆模糊化。模糊化可以帮助我们评估规则，但是模糊系统的最终输出必须是一个清晰的值。逆模糊化过程的输入是集成的模糊输出集，该输出是单一的数值

如何将聚合模糊集逆模糊化

有几种逆模糊化的方法（Cox，1999），但最常用的是质心技术（centroid technique）。这种技术寻找一个点，这个点所在的垂直线能够将聚合集分割成两个相等的部分。这个重力的质心（Centre of Gravity，COG）的数学表示为：

$$COG = \frac{\int_a^b \mu_A(x)\times x\,dx}{\int_a^b \mu_A(x)\,dx} \tag{4.17}$$

如图4.13所示，质心逆模糊化方法在[a，b]区间找到表示模糊集A的重力质心的点。



111

图4.13 逆模糊化的质心方法

从理论上说，COG是在聚合输出隶属函数的连续空间点上计算，但实际上，可以通过在如图4.13所示的样本点上计算COG来得到COG的合理估值。在这种情况下，使用下面的公式：

$$COG = \frac{\sum_{x=a}^{b} \mu_A(x)x}{\sum_{x=a}^{b} \mu_A(x)} \tag{4.18}$$

现在计算本例中的重力质心，方法如图4.14所示。

$$COG = \frac{(0+10+20)\times 0.1+(30+40+50+60)\times 0.2+(70+80+90+100)\times 0.5}{0.1+0.1+0.1+0.2+0.2+0.2+0.2+0.5+0.5+0.5+0.5}$$

$$=67.4$$

因此，逆模糊化的结果表示，z1的清晰输出为67.4，意思是该“模糊”项目的风险为67.4%。

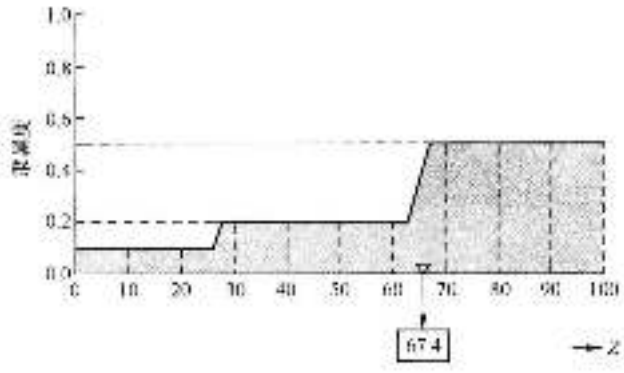


图 4.14 变量模糊集的逆模糊化

4.6.2 Sugeno-style 推理

如前所述，Mamdani-style 推理需要通过整合连续变化的函数找到二维形状的质心，通常这个过程计算的效率不高。

是否可以缩短模糊推理的时间

我们使用只有一个尖峰的单态函数作为规则后项的隶属函数。这种方法由 Michio Sugeno（被誉为日本的 Zadeh）在 1985 年首次提出（Sugeno，1985）。单态模式，更确切地说是模糊单态模式，是带有隶属函数的模糊集，该隶属函数在论域的某个点上为 1，在其他点上为 0。

Sugeno-style 模糊推理和 Mamdani 方法很相似。Sugeno 仅改变了规则的后项。他使用输入变量（而不是模糊集）的数学函数。Sugeno-style 模糊规则的格式为：

IF x is A
AND y is B
THEN z is $f(x,y)$

其中，x、y 和 z 是语言学变量；A 和 B 分别是论域 X 和 Y 上的模糊集；$f(x,y)$ 是数学函数。 112

最常用的单态 Sugeno 模糊模型应用以下形式的模糊规则：

IF x is A
AND y is B
THEN z is k

其中 k 是常数。

在这种情况下，每个模糊规则的输出是常数。换句话说，所有的后项隶属函数由单态尖峰来表示。图 4.15 显示了零阶 Sugeno 模型的模糊推理过程。我们比较一下图 4.15 和图 4.10。Sugeno 和 Mamdani 两种方法的相似性是显而易见的，仅有的差别是 Sugeno 方法的后项是单态的。

清晰的输出结果是怎样得到的

如图 4.15 所示，聚合操作就是将所有的单态模式包含在一起，下面来寻找这些单态模式的加权平均值（Weight Average，WA）：

$$\mathrm{WA}=\frac{\mu(k1)\times k1+\mu(k2)\times k2+\mu(k3)\times k3}{\mu(k1)+\mu(k2)+\mu(k3)}=\frac{0.1\times 20+0.2\times 50+0.5\times 80}{0.1+0.2+0.5}=65$$

因此，对于我们的问题，零阶 Sugeno 系统就足够了。幸运的是，单态输出函数通常能够满足问题的需要。

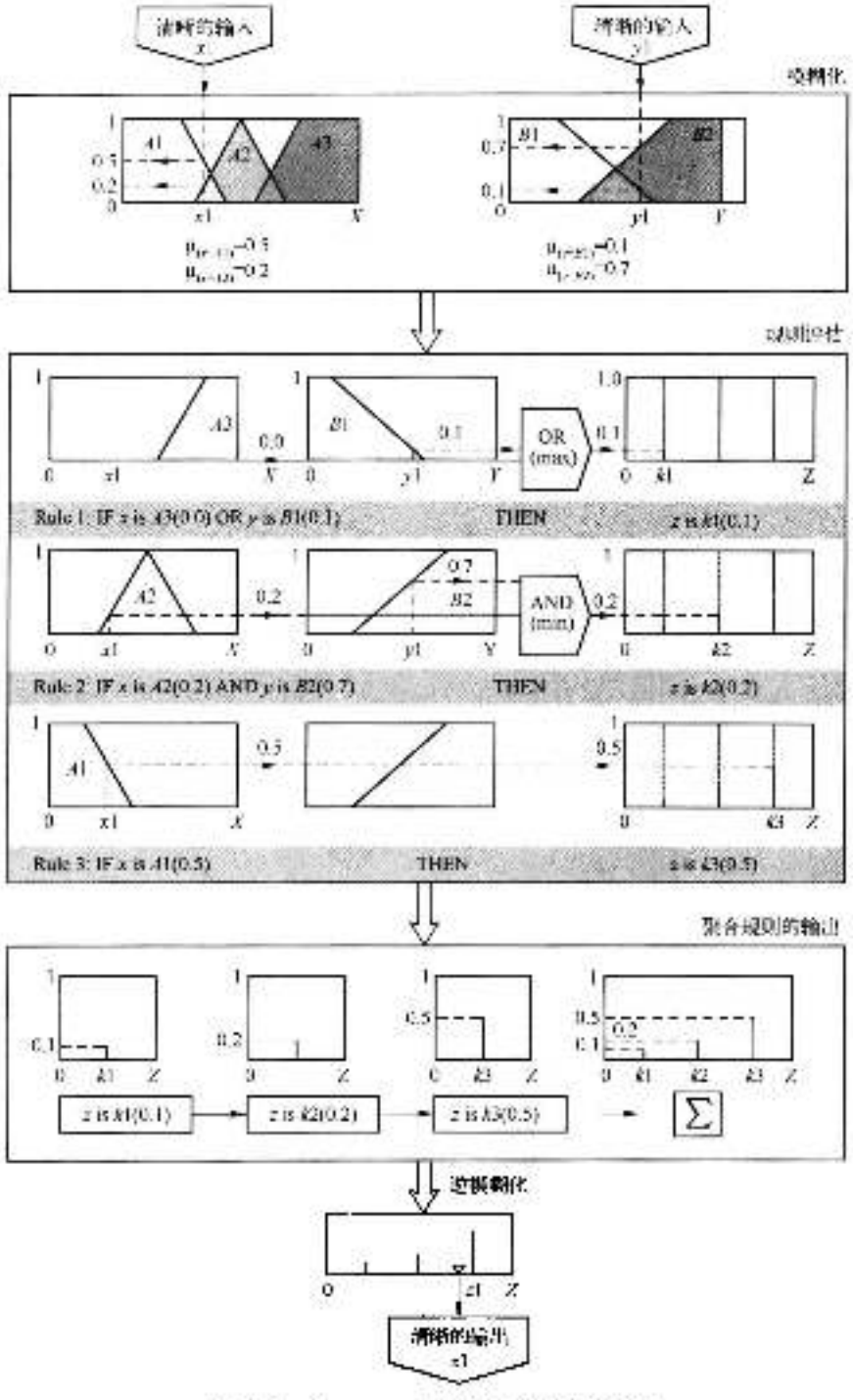


图 4.15 Sugeno-style 模糊推理的基本结构

用 Mamdani 法还是 Sugeno 法

在获取专家知识时常使用 Mamdani 方法，因为这种方法可以用更直接、更接近人类的方式来描述专家的意见，但是 Mamdani 模糊推理的计算量大。另一方面，Sugeno 方法的计算效率高，并能够与优化算法和自适应技术协同工作，这使得该方法在控制问题，尤其是动态非线性系统中很有吸引力。

4.7 建立模糊专家系统

为了说明如何设计模糊专家系统，下面考虑一个备件服务中心的例子（Turksen et al.，1992）。

服务中心保存备件并修复损坏的备件。客户拿来一个损坏的备件，换走一个相同型号的备件。损坏的备件被修好后放置在架子上作为备件，如果架子上有所需的备件，那么客户从架子上拿走备件并离开服务中心。如果架子上没有备件，客户就必须等待，直到拿到需要的备件为止。我们的目标是向备件服务中心的经理建议使客户满意的决策方针。

113 ~ 114

开发模糊专家系统的典型过程的步骤如下：

1）指定问题并定义语言变量。

2）定义模糊集。

3）抽取并构造模糊规则。

4）对模糊集、模糊规则和过程进行编码以在专家系统中执行模糊推理。

5）评估并调试系统。

步骤 1：指定问题并定义语言变量。

建立任何专家系统的第一个也是最重要的步骤是指定问题，需要按照知识工程的方式来描述问题。换句话说，要确定问题的输入和输出变量及范围。

在我们的问题中，有 4 个主要的语言变量：平均等待时间（平均延迟）m、服务中心的修理利用因子 ρ、服务员人数 s 以及初始备件数量 n。

客户的平均等待时间 m 是评价服务中心性能最重要的标准。服务的实际平均延迟不能超过客户可以接受的限度。

服务中心的修理利用因子 ρ 是客户到达率 λ 除以离开率 μ。λ 和 μ 的量级分别表示零件损坏（单位时间内的损坏）和修复（单位时间内的修复）的比率。显然，修复率应该和服务员人数 s 成正比。要提高服务中心的生产率，经理应当使修理利用率因子的值尽可能高。

服务员的人数 s 和初始备件数量 n 直接影响客户的平均等候时间，对中心的性能有重要的影响。如果增加 s 和 n，就可以减少平均延迟时间，但同时增加了雇佣新服务人员的成本，增加配件的数量，就要扩展服务中心的库存能力以容纳新增配件。

首先确定备件的初始数量 n，给定客户平均延迟时间 m、服务员的人数 s 和修理利用因子 ρ。因此，在这里考虑的决策模型中，有 3 个输入：m、s 和 ρ，一个输出 n。换句话说，服务中心的经理应该决定在客户可接受的范围内维持平均延迟时间所需的配件数量。

接下来确定语言变量的范围。假设表 4.3 是得到的结果，其中 m、s 和 n 已经标准化，即除以相应的最大量级基本数值，得到的范围 [0，1]。

表 4.3 语言变量及其范围

语言变量：平均延迟（m）		
语言值	符号	值范围（标准化）
Very Short	VS	[0，0.3]
Short	S	[0.1，0.5]
Medium	M	[0.4，0.7]
语言变量：服务员人数（s）		
语言值	符号	值范围（标准化）
Small	S	[0，0.35]
Medium	M	[0.30，0.70]
Large	L	[0.60，1]

（续）

语言值	语言变量：修理利用因子（ρ） 符号	值范围（标准化）
Low	L	[0, 0.6]
Medium	M	[0.4, 0.8]
High	H	[0.6, 1]
	语言变量：备件数（s）	
语言值	符号	值范围（标准化）
Very Small	VS	[0, 0.30]
Small	S	[0, 0.40]
Rather Small	RS	[0.25, 0.45]
Medium	M	[0.30, 0.70]
Rather Large	RL	[0.55, 0.75]
Large	L	[0.60, 1]
Very Large	VL	[0.70, 1]

注意：对于客户平均延迟 m，这里仅考虑了3个语言值：Very Short、Short 和 Medium，因为其他的取值，例如 Long 和 Very Long 是不符合实际情况的。服务中心的经理不可能容忍客户等待的时间超过 Medium。

实际上，所有的语言变量，语言值及其范围通常是由领域专家来选择的。

步骤2：确定模糊集。

模糊集可以有不同的形状，通常三角形和四边形就足以表达专家的知识了，同时也能最大地简化计算的过程。

415
/
116

图4.16～图4.19显示了例子中用到的所有语言变量的模糊集。你可能已经注意到，这里的一个关键点是在相邻的模糊集间保持足够的重叠，以便模糊系统能够平滑地响应。

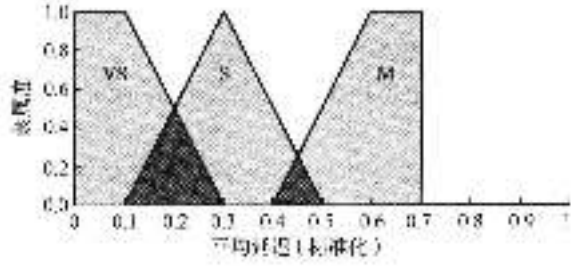


图4.16 平均延迟 m 的模糊集

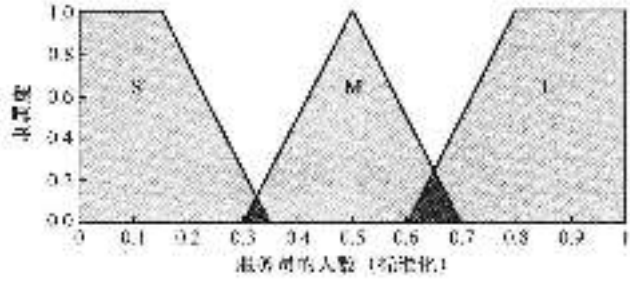


图4.17 服务员人数 s 的模糊集

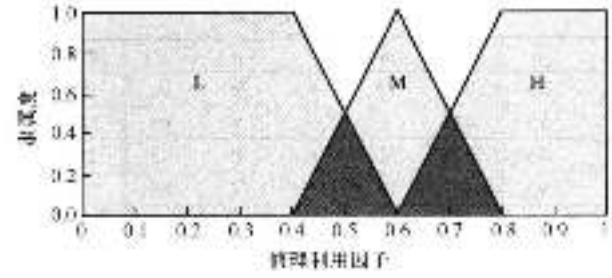


图 4.18 修理利用因子 ρ 的模糊集

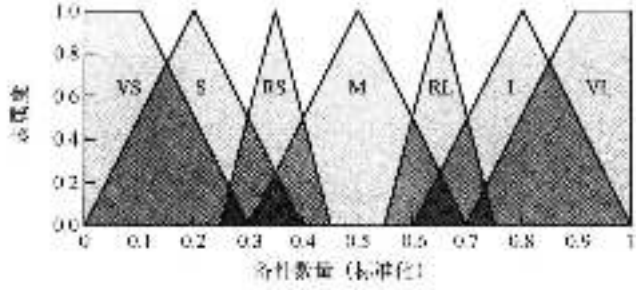


图 4.19 备件数量 n 的模糊集

[117]

步骤 3：抽取并构造模糊规则。

下面要获得模糊规则。要完成这个任务，就需要向专家咨询如何使用前面定义过的模糊语言变量来解决问题。

所需要的知识也可以从其他渠道收集得到，这些渠道包括书本、计算机数据库、流程图和观察到的人类行为。在本例中，可以应用研究论文（Turksen et al., 1992）中提供的规则。

在本例中，有 3 个输入变量和一个输出变量，用矩阵形式来表示模糊规则通常十分便利。一个二对一的系统（两个输入和一个输出）可以描述为输入变量的 $M \times N$ 矩阵。横轴为一个输入变量的语言值，纵轴为另一个输入变量的语言值，行和列的交集为输出变量的语言值。对于三对一的系统（三个输入和一个输出），可以用 $M \times N \times K$ 的立方体表示。这种表现方法称为模糊关联记忆（Fuzzy Associative Memory，FAM）。

首先，使用修理利用因子 ρ 和备件数量 n 之间最基本的关系，假设其他输入变量是固定的。这个关系可以用下面的形式来表达：如果 ρ 增加，那么 n 不应该减少。因此可写出下面 3 条规则：

1. If utilisation_factor is L) then (number_of_spares is S)
2. If utilisation_factor is M) then (number_of_spares is M)
3. If utilisation_factor is H) then (number_of_spares is L)

现在用 3 × 3 的 FAM，它用矩阵形式表示刚下的规则，结果如图 4.20 所示。

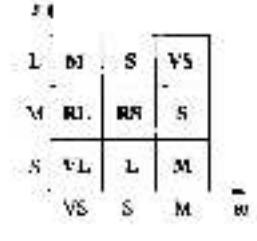


图 4.20 正方形 FAM 表示

同时，详细分析服务中心的操作（带有“专家特征”）（Turksen et al., 1992），可以导出描述专家系统中使用的所有变量之间复杂的复杂关系的 27 条规则。表 4.4 给出了这些规则，图 4.21 显示了立方体（3 × 3 × 3）FAM。

[118]

表 4.4 规则表

规则	m	s	ρ	n	规则	m	s	ρ	n	规则	m	s	ρ	n
1	VS	S	L	VS	10	VS	S	M	S	19	VS	S	H	VL
2	S	S	L	VS	11	S	S	M	VS	20	S	S	H	L
3	M	S	L	VS	12	M	S	M	VS	21	M	S	H	M
4	VS	M	L	VS	13	VS	M	M	RS	22	VS	M	H	M
5	S	M	L	VS	14	S	M	M	S	23	S	M	H	M
6	M	M	L	VS	15	M	M	M	VS	24	M	M	H	S
7	VS	L	L	S	16	VS	L	M	M	25	VS	L	H	RL
8	S	L	L	S	17	S	L	M	RS	26	S	L	H	M
9	M	L	L	VS	18	M	L	M	S	27	M	L	H	RS

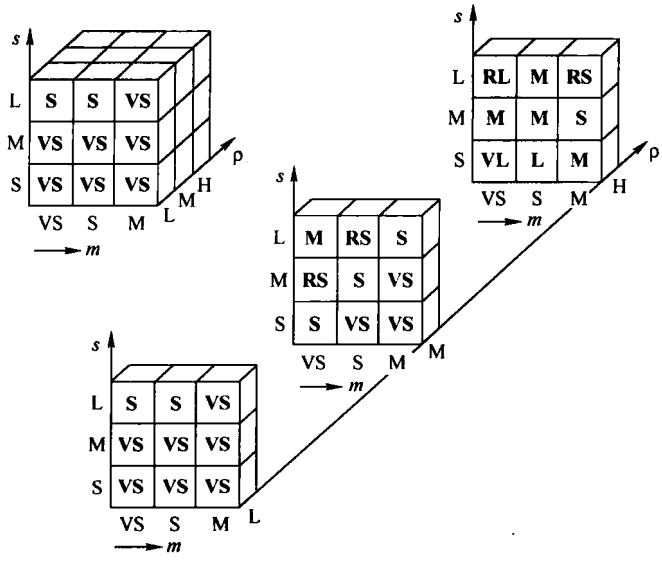


图 4.21 立方体 FAM 和分层的立方体 FAM 表示

首先开发 12 个（3 +3 ×3）规则，接下来得到 27 个（3 ×3 ×3）规则。如果实现了这两个方案，就可以比较结果。只有系统的性能才能告诉我们哪个方案比较好。

Rule Base1

1. If (utilisation_factor is L) then (number_of_spares is S)
2. If (utilisation_factor is M) then (number_of_spares is M)
3. If (utilisation_factor is H) then (number_of_spares is L)
4. If (mean_delay is VS) and (number_of_servers is S) then (number_of_spares is VL)
5. If (mean_delay is S) and (number_of_servers is S) then (number_of_spares is L)

119

6. If (mean_delay is M) and (number_of_servers is S) then (number_of_spares is M)
7. If (mean_delay is VS) and (number_of_servers is M) then (number_of_spares is RL)
8. If (mean_delay is S) and (number_of_servers is M) then (number_of_spares is RS)
9. If (mean_delay is M) and (number_of_servers is M) then (number_of_spares is S)
10. If (mean_delay is VS) and (number_of_servers is L) then (number_of_spares is M)
11. If (mean_delay is S) and (number_of_servers is L) then (number_of_spares is S)
12. If (mean_delay is M) and (number_of_servers is L) then (number_of_spares is VS)

Rule Base2

1. If (mean_delay is VS) and (number_of_servers is S) and (utilisation_factor is L) then (number_of_spares is VS)
2. If (mean_delay is S) and (number_of_servers is S) and (utilisation_factor is L) then (number_of_spares is VS)
3. If (mean_delay is M) and (number_of_servers is S) and (utilisation_factor is L) then (number_of_spares is VS)
4. If (mean_delay is VS) and (number_of_servers is M) and (utilisation_factor is L) then (number_of_spares is VS)
5. If (mean_delay is S) and (number_of_servers is M) and (utilisation_factor is L) then (number_of_spares is VS)
6. If (mean_delay is M) and (number_of_servers is M) and (utilisation_factor is L) then (number_of_spares is VS)
7. If (mean_delay is VS) and (number_of_servers is L) and (utilisation_factor is L) then (number_of_spares is S)
8. If (mean_delay is S) and (number_of_servers is L) and (utilisation_factor is L) then (number_of_spares is S)
9. If (mean_delay is M) and (number_of_servers is L) and (utilisation_factor is L) then (number_of_spares is VS)
10. If (mean_delay is VS) and (number_of_servers is S) and (utilisation_factor is M) then (number_of_spares is S)
11. If (mean_delay is S) and (number_of_servers is S) and (utilisation_factor is M) then (number_of_spares is VS)
12. If (mean_delay is M) and (number_of_servers is S) and (utilisation_factor is M) then (number_of_spares is VS)
13. If (mean_delay is VS) and (number_of_servers is M) and (utilisation_factor is M) then (number_of_spares is RS)
14. If (mean_delay is S) and (number_of_servers is M) and (utilisation_factor is M) then (number_of_spares is S)
15. If (mean_delay is M) and (number_of_servers is M) and (utilisation_factor is M) then (number_of_spares is VS)
16. If (mean_delay is VS) and (number_of_servers is L) and (utilisation_factor is M) then (number_of_spares is M)
17. If (mean_delay is S) and (number_of_servers is L) and (utilisation_factor is M) then (number_of_spares is RS)
18. If (mean_delay is M) and (number_of_servers is L) and (utilisation_factor is M) then (number_of_spares is S)
19. If (mean_delay is VS) and (number_of_servers is S) and (utilisation_factor is H) then (number_of_spares is VL)
20. If (mean_delay is S) and (number_of_servers is S) and (utilisation_factor is H) then (number_of_spares is L)
21. If (mean_delay is M) and (number_of_servers is S) and (utilisation_factor is H) then (number_of_spares is M)
22. If (mean_delay is VS) and (number_of_servers is M) and (utilisation_factor is H) then (number_of_spares is M)
23. If (mean_delay is S) and (number_of_servers is M) and (utilisation_factor is H) then (number_of_spares is M)
24. If (mean_delay is M) and (number_of_servers is M) and (utilisation_factor is H) then (number_of_spares is S)
25. If (mean_delay is VS) and (number_of_servers is L) and (utilisation_factor is H) then (number_of_spares is RL)
26. If (mean_delay is S) and (number_of_servers is L) and (utilisation_factor is H) then (number_of_spares is M)
27. If (mean_delay is M) and (number_of_servers is L) and (utilisation_factor is H) then (number_of_spares is RS)

120

步骤 4：对模糊集、模糊规则和过程编码以便在专家系统中执行模糊推理。

定义模糊集和模糊规则后，就要对它们编码，建立实际的专家系统，为了达到这个目标，有两个选择：用 C 或 Pascal 这样的编程语言来建立系统，或者用 MathWorks 的 MATLAB Fuzzy Logic Toolbox 或 Fuzzy System Engineering 的 Fuzzy Knowledge Builder 这样的模糊逻辑开发工具来建立系统。

大多数经验丰富的模糊系统构建者喜欢用 C/C++ 编程语言（Li and Gupta, 1995; Cox, 1999），因为这种语言有更大的灵活性。但是，为了快速地开发并得到模糊专家系统的原型，最好的选择是模糊逻辑开发工具。这种工具会为创建和测试专家系统提供完善的环境。例如，MATLAB Fuzzy Logic Toolbox 有 5 个集成的图形编辑器：模糊推理系统编辑器、规则编辑器、隶属函数编辑器、模糊推理查看器和输出界面查看器。这些功能使得设计模糊系统变得更加容易。对于没有丰富的模糊专家系统构建经验的初学者而言，这也是个更好的选择。在选定了模糊逻辑开发工具后，知识工程师仅仅需要将模糊逻辑按照类似于英语的语法来编码，并定义图形化的隶属函数即可。

为了建立本例的模糊专家系统，我们选择最常用的工具之一———MATLAB Fuzzy Logic Toolbox。它提供了处理模糊规则和用户图形界面的系统架构。对于构建模糊系统的新手而言，这个工具很容易控制，并易于使用。

步骤5：评估并调试系统。

最后也是最艰苦的工作是评估和调整系统。我们要看模糊系统是否满足开始指定的需求。
121 一些测试情形取决于平均延迟、服务员的人数和修理利用因子。Fuzzy Logic Toolbox 可以产生表面来帮助我们分析系统的性能。图4.22为两个输入和一个输出的系统三维图。

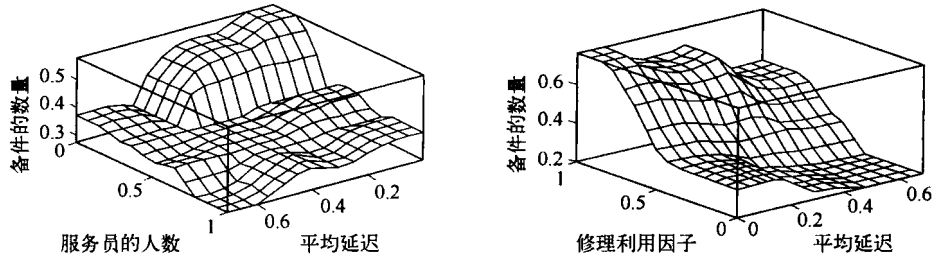


图4.22 规则库1的三维图

但是我们的系统有3个输入和一个输出。可以超越三维空间吗？如果超越了三维空间，那么在显示结果时会遇到困难。幸运的是，Fuzzy Logic Toolbox 有专门的功能：它可以根据任意两个输入，以及将第三个输入设成常数来绘制三维输出表面。因此，我们就可以在两个三维图上观察3个输入一个输出的系统的性能。

虽然模糊系统运转良好，但我们还想应用规则库2来改进它。结果如图4.23所示。比较一下图4.22和图4.23，就可以看到改进之处。

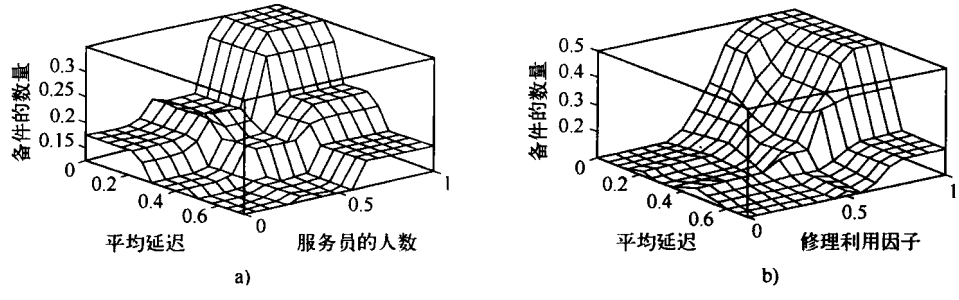


图4.23 规则库2的三维图

即使是这样，专家可能对系统的性能还是不满意。为了提高性能，专家可能建议在服务员人数的论域上增加附加的集合（Rather Small and Rather Large）来改进系统的性能（如图4.24所示），并依照图4.25所示的FAM扩展规则库。模糊系统可以很容易地进行修改和扩展让我们能
122 够听从专家的建议并很快获得如图4.26所示的结果。

通常，调试模糊专家系统花费时间和精力远远超过决定模糊集和构建模糊规则所花费的时间和精力。一般情况下，合理的解决方案可以从第一组模糊集和模糊规则中获得。这是模糊逻辑公认的优势；但是，改进系统更像是艺术而不是工程。

调试模糊专家系统要按照以下次序执行一系列的操作：

1）回顾模型的输入变量和输出变量，如果有必要重新定义变量的范围，要特别注意变量的单位，在同一领域中使用的变量必须用论域上的相同单位加以度量。

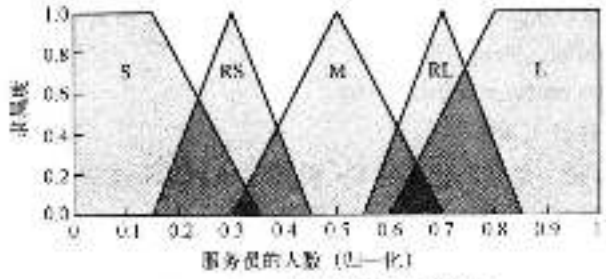


图 4.24 更改服务员人数 x 的模糊集

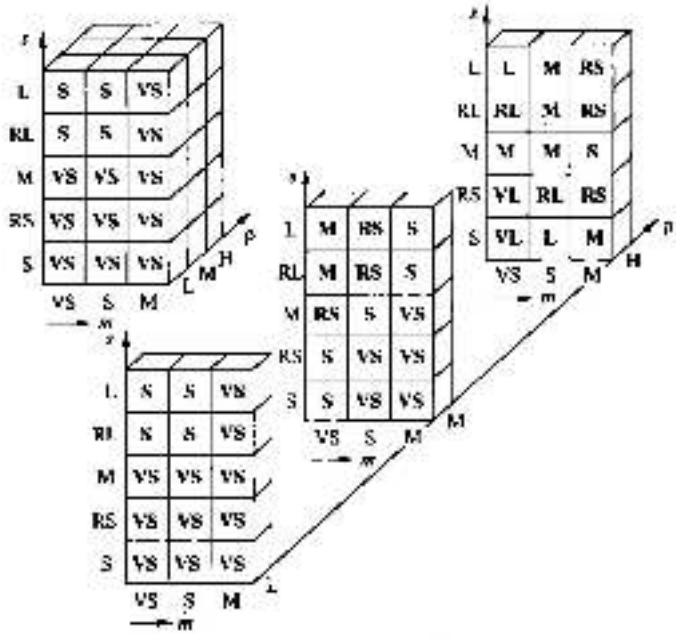


图 4.25 规则库 3 的立方体 FAM

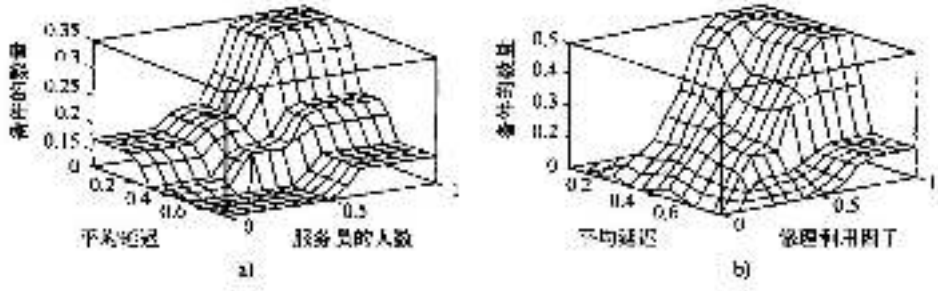


图 4.26 规则库 3 的三维图

2）回顾模糊集，如果必要的话可以在论域上定义附加的集合。使用更广泛的模糊集可能会导致模糊系统粗糙执行。

3）相邻集合之间要有足够的重叠。虽然没有精确的方法确定合适的重叠程度，但这里建议二对二和四对四的模糊系统应该有 25% – 50% 的重叠（Cox，1999）

4）回顾现有的规则，如果有必要则在规则库中加入新的规则。

5）检查规则库以便写规则限制语来捕捉系统的不正常行为。

6）调节规则执行的权重。大多数模糊逻辑工具允许通过改变权重乘数来控制规则的重要性。

在 Fuzzy Logic Toolbox 中，所有的规则都有默认的权重（1.0），但是用户可以通过调节权重来降低规则的强制性。例如，指定

If (utilisation_factor is H) then (number_of_spares is L) (0.6)

那么规则的强制性降低了 40%。

7）修订模糊集的形状。在大多数情况下，模糊系统对形状近似是高度宽容的。因此即使模糊集的形状没有精确的定义，系统也可以运行良好。

逆模糊化的方法怎么样？是否应该尝试不同的技术来调整系统

质心技术可以得到一致的结论。这种是一种对整个模糊区域的高度和宽度以及备件的尖峰敏感的很好的平衡方法。因此，除非你有充足的理由相信你的模糊系统在其他逆模糊化方法中将运转良好，否则推荐利用质心技术。

123 ~ 124

4.8 小结

本章介绍了模糊逻辑并探讨了其背后的哲学思想，介绍了模糊集的概念以及如何在计算机中表示模糊集，并检查了模糊集的操作。我们还定义了语言变量和模糊限制语，随后介绍了模糊规则并解释了经典规则和模糊规则的区别。本章主要研究了两种模糊推理技术——Mamdani 和 Sugeno，并给出了它们的适用情况。最后介绍开发模糊专家系统的步骤，并详细说明了建立和调整模糊系统的实际步骤。

本章的主要内容有：

- 模糊逻辑是描述模糊的逻辑。模糊逻辑试图对人类的语言、决策和常识建模，这样会促使出现更加智能化的机器。
- 模糊逻辑是在 20 世纪 20 年代由 Jan Lukasiewicz 提出的，在 20 世纪 30 年代由 Max Black 进行了仔细的研究，并在 20 世纪 60 年代由 Lotfi Zadeh 重新发现，并扩展至数学逻辑的正式系统中。
- 模糊逻辑是基于隶属度的知识表达的数学原理的集合，而不是基于经典二值逻辑中的清晰的隶属关系。和二值的布尔逻辑不同，模糊逻辑是多值的。
- 模糊集是有模糊边界的集合，例如，对于男性身高而言的 short、average 或 tall。为了在计算机中表达模糊集，我们用函数来表示集合，并将集合中的元素映射到隶属度。模糊专家系统中经典的隶属函数是三角函数和梯形函数。
- 语言变量用来描述有含糊或模糊取值的术语或概念。这些值用模糊集表示。
- 模糊限制语是模糊集的限定词，用来更改模糊集的形状。模糊限制语包含 very、somewhat、quite、more or less 及 slightly 这样的副词。模糊限制语执行数学集中的操作来减少模糊元素的隶属度（例如，very tall men），或通过增加隶属度来扩展（例如，more or less tall men，即有点高的男人）。通过增加隶属度在 0.5 以上的值来加强，或减少隶属度在 0.5 以下的值来减弱（例如，indeed tall men）。
- 模糊集之间可以相互作用。这些关系称为操作。模糊集间的主要操作有：补、包含、交和并。
- 模糊规则用来获取人类的知识。模糊规则是以下形式的条件语句：

IF　　x is A
THEN　y is B

125

其中，x 和 y 是语言变量，A 和 B 是模糊集决定的语言值。

- 模糊推理是用模糊集理论将给定的输入映射到输出的过程。模糊推理包含 4 个步骤：输入变量模糊化、评估规则、聚合规则的输出以及逆模糊化。
- 有两种模糊推理技术：Mamdani 和 Sugeno。Mamdani 方法在模糊专家系统中应用广泛，因

为这种方法能够用模糊规则来获得专家的知识。但是 Mamdani 的模糊推理的计算量非常大。

- 为了改进模糊推理的计算效率，Sugeno 使用有尖峰的单态函数来作为规则后项的隶属函数。Sugeno 方法可以和优化及自适应技术协同作用，因此在控制，尤其是动态非线性系统中非常适用。
- 建立模糊系统是一个迭代的过程，包括定义模糊集和模糊规则、评估和调试系统，以适应具体的需求。
- 调试是在建立模糊系统中最单调和费力的过程，通常涉及调试现有的模糊集和模糊规则。

复习题

1. 什么是模糊逻辑？哪些人创建了模糊逻辑？为什么说模糊逻辑促进了更加智能机器的出现？
2. 什么是模糊集和隶属函数？清晰集和模糊集之间有什么不同？定义男性体重论域上可能的模糊集。
3. 定义语言变量和语言值，并给出例子。如何在模糊规则中使用语言变量？给出模糊规则的例子。
4. 什么是模糊限制语？模糊限制语如何更改现有的模糊集？给出执行集中、扩展和增强操作的模糊限制语。给出合适的数字表示和图形表示。
5. 定义模糊集的主要操作，并举例说明。如何应用模糊集操作、性质和模糊限制语，以及从现有模糊集中获得大量模糊集？
6. 什么是模糊规则？经典规则和模糊规则之间的区别是什么？举例说明。
7. 定义模糊推理。模糊推理过程的主要步骤有哪些？
8. 如何评估模糊规则的多个前项？举例说明。执行 AND 和 OR 模糊操作的不同方法会得到不同的结果吗？为什么？ 126
9. 什么是模糊集的剪切？什么是模糊集的缩放？哪种方法最好地保持了模糊集的原始形状？为什么？举例说明。
10. 什么是逆模糊化？最常用的逆模糊化方法是什么？如何通过数学方式和图形方式确定模糊系统的最终输出？
11. Mamdani 和 Sugeno 模糊推理方法有什么不同？什么是单态？
12. 开发模糊专家系统的主要步骤有哪些？该过程中最费力和枯燥的步骤是哪个？为什么？

参考文献

Chang, A.M. and Hall, L.O. (1992). The validation of fuzzy knowledge-based systems, *Fuzzy Logic for the Management of Uncertainty*, L.A. Zadeh and J. Kacprzyk, eds, John Wiley, New York, pp. 589–604.

Black, M. (1937). Vagueness: An exercise in logical analysis, *Philosophy of Science*, 4, 427–455.

Cox, E. (1999). *The Fuzzy Systems Handbook: A Practitioner's Guide to Building, Using, and Maintaining Fuzzy Systems*, 2nd edn. Academic Press, San Diego, CA.

Li, H. and Gupta, M. (1995). *Fuzzy Logic and Intelligent Systems*. Kluwer Academic Publishers, Boston, MA.

Lukasiewicz, J. (1930). Philosophical remarks on many-valued systems of propositional logic. Reprinted in *Selected Works*, L. Borkowski, ed., Studies in Logic and the Foundations of Mathematics, North-Holland, Amsterdam, 1970, pp. 153–179.

Mamdani, E.H. and Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller, *International Journal of Man–Machine Studies*, 7(1), 1–13.
Sugeno, M. (1985). *Industrial Applications of Fuzzy Control*. North-Holland, Amsterdam.
Turksen, I.B., Tian, Y. and Berg, M. (1992). A fuzzy expert system for a service centre of spare parts, *Expert Systems with Applications*, 5, 447–464.
Zadeh, L. (1965). Fuzzy sets, *Information and Control*, 8(3), 338–353.
Zadeh, L. (1973). Outline of a new approach to the analysis of complex systems and decision processes, *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(1), 28–44.

参考书目

Arfi, B. (2010). *Linguistic Fuzzy-Logic Methods in Social Sciences*. Springer-Verlag, Berlin.
Bergmann, M. (2008). *An Introduction to Many-Valued and Fuzzy Logic: Semantics, Algebras, and Derivation Systems*. Cambridge University Press, New York.
Bojadziev, G. and Bojadziev, M. (2007). *Fuzzy Logic for Business, Finance, and Management*. World Scientific, Singapore.
Buckley, J.J. and Eslami, E. (2002). *An Introduction to Fuzzy Logic and Fuzzy Sets*. Physica-Verlag, Heidelberg.
Carlsson, C., Fedrizzi, M. and Fuller, R. (2004). *Fuzzy Logic in Management*, Kluwer Academic Publishers, Norwell, MA.
Engelbrecht, A.P. (2007). *Computational Intelligence: An Introduction*, 2nd edn. John Wiley, Chichester.

127

Galindo, J., Urrutia, A. and Piattini, M. (2006). *Fuzzy Databases: Modeling, Design and Implementation*. Idea Group Publishing, Hershey, PA.
Hájek, P. (1998). *Metamathematics of Fuzzy Logic*. Kluwer Academic Publishers, Dordrecht.
Jantzen, J. (2007). *Foundations of Fuzzy Control*. John Wiley, Chichester.
Klir, G.J. and Yuan, B. (1996). *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers by Lotfi A. Zadeh*, Advances in Fuzzy Systems – Applications and Theory, vol. 6. World Scientific, Singapore.
Kosko, B. (1992). *Fuzzy Associative Memory Systems, Fuzzy Expert Systems*, A. Kandel, ed., CRC Press, Boca Raton, FL, pp. 135–164.
Kosko, B. (1993). *Fuzzy Thinking: The New Science of Fuzzy Logic*. Hyperion, New York.
Kosko, B. (1997). *Fuzzy Engineering*. Prentice Hall, Upper Saddle River, NJ.
Kosko, B. (2000). *Heaven in a Chip: Fuzzy Visions of Society and Science in the Digital Age*. Random House/Three Rivers Press, New York.
Lee, C.C. (1990). Fuzzy logic in control systems: fuzzy logic controller, Part I, *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2), 404–418.
Lee, C.C. (1990). Fuzzy logic in control systems: fuzzy logic controller, Part II, *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2), 419–435.
Lilly, J.H. (2010). *Fuzzy Control and Identification*. John Wiley, Chichester.
Mamdani, E.H. (1974). Application of fuzzy algorithms for control of simple dynamic plant, *Proceedings of the Institute of Electrical Engineers*, 121(12), 1585–1588.
Margaliot, M. and Langholz, G. (2004). Fuzzy control of a benchmark problem: a computing with words approach, *IEEE Transactions on Fuzzy Systems*, 12(2), 230–235.
Mendel, J. and Wu, D. (2010). *Perceptual Computing: Aiding People in Making Subjective Judgments*. John Wiley, Hoboken, NJ.
Mukaidono, M. (2001). *Fuzzy Logic for Beginners*. World Scientific, Singapore.
Negoita, C.V. (1985). *Expert Systems and Fuzzy Systems*. Benjamin/Cummings, Menlo Park, CA.
Negoita, C.V. (2000). *Fuzzy Sets*. New Falcon Publications, Tempe, AZ.
Nguyen, H.T. and Walker, E.A. (2005). *A First Course in Fuzzy Logic*, 3rd edn. Chapman & Hall/CRC Press, London.
Pedrycz, W. and Gomide, F. (2007). *Fuzzy Systems Engineering: Toward Human-Centric Computing*. John Wiley, Hoboken, NJ.

Ross, T. (2010). *Fuzzy Logic with Engineering Applications*, 3rd edn. John Wiley, Chichester.

Siler, W. and Buckley, J.J. (2004). *Fuzzy Expert Systems and Fuzzy Reasoning*. John Wiley, Hoboken, NJ.

Tanaka, K. (1996). *An Introduction to Fuzzy Logic for Practical Applications*. Springer-Verlag, New York.

Terano, T., Asai, K. and Sugeno, M. (1992). *Fuzzy Systems Theory and its Applications*. Academic Press, London.

Von Altrock, C. (1995). *Fuzzy Logic and Neuro Fuzzy Applications Explained*. Prentice Hall, Upper Saddle River, NJ.

Yager, R.R. and Filev, D.P. (1994). *Essentials of Fuzzy Modeling and Control*. John Wiley, New York.

Yager, R.R. and Zadeh, L.A. (1992). *An Introduction to Fuzzy Logic Applications in Intelligent Systems*. Kluwer Academic Publishers, Boston, MA.

Yager, R.R. and Zadeh, L.A. (1994). *Fuzzy Sets, Neural Networks and Soft Computing*. Van Nostrand Reinhold, New York.

128

Zadeh, L.A. (1975). The concept of a linguistic variable and its applications to approximate reasoning, Part I, *Information Sciences*, 8, 199–249.

Zadeh, L.A. (1975). The concept of a linguistic variable and its applications to approximate reasoning, Part II, *Information Sciences*, 8, 301–357.

Zadeh, L.A. (1975). The concept of a linguistic variable and its applications to approximate reasoning, Part III, *Information Sciences*, 9, 43–80.

Zadeh, L.A. (1999). From computing with numbers to computing with words – from manipulation of measurements to manipulation of perceptions, *IEEE Transactions on Circuits and Systems Science*, 45, 105–119.

Zadeh, L.A. (2004). Precisiated natural language (PNL), *AI Magazine*, 25(3), 74–91.

Zadeh, L.A. (2005). Towards a generalised theory of uncertainty (GTU): an outline, *Information Sciences – Informatics and Computer Science*, 172(1–2), 1–40.

Zadeh, L.A. (2009). Toward extended fuzzy logic – a first step, *Fuzzy Sets and Systems*, 160(21), 3175–3181.

Zimmermann, H.-J. (2001). *Fuzzy Set Theory and its Applications*, 4th edn. Kluwer Academic Publishers, Boston, MA.

129

第5章
Artificial Intelligence, 3E

基于框架的专家系统

本章介绍专家系统中表达知识的一种常用方法：基于框架的方法，介绍如何开发基于框架的专家系统，并通过实例来说明。

5.1 框架简介

计算机中的知识可以通过几种技术来表示。在前面几章中介绍了规则。本章主要用框架（frame）来表达知识。

什么是框架

框架是带有关于某个对象和概念的典型知识的数据结构。框架由 Marvin Minsky 于20世纪70年代提出（Minsky, 1975），用来在基于框架的专家系统中获得和表达知识。图5.1所示的登机证即为表达航班乘客知识的框架。两个框架的结构相同。

QANTAS BOARDING PASS	
Carrier:	QANTAS AIRWAYS
Name:	MR N BLACK
Flight:	QF 612
Date:	29DEC
Seat:	23A
From:	HOBART
To:	MELBOURNE
Boarding:	0620
Gate:	2

a)

AIR NEW ZEALAND BOARDING PASS	
Carrier:	AIR NEW ZEALAND
Name:	MRS J WHITE
Flight:	NZ 0198
Date:	23NOV
Seat:	27K
From:	MELBOURNE
To:	CHRISTCHURCH
Boarding:	1815
Gate:	4

b)

图5.1 登机证框架

每个框架都有自己的名称和相关的属性（或槽）的集合。*Name*、*weight*、*height* 和 *age* 是 *person* 框架的槽，*model*、*processor*、*memory* 和 *price* 是 *computer* 框架的槽。每个属性（或槽）有具体的取值。例如，在图5.1a中，*Carrier*（运输公司）的槽值是 QANTAS AIRWAYS，*Gate* 的槽值为2。在某些情况下，槽值可能不是一个值，而是由一段程序来决定它的取值。

在专家系统中，框架主要用于和产生式规则连接。

为什么有必要使用框架

框架为实现知识的结构化和简化表达提供了一种自然的方法。在单一实体中，框架包含与某个对象或概念有关的所有必需的知识。框架提供了一种将知识组织成槽以描述对象的不同属性和特征的方法。

前面已经说过，真实世界中的很多问题都可以很自然地用 IF-THEN 产生式规则来表示。但是，使用系统的搜索技术基于规则的专家系统会用到散布在整个知识库中的行为。这样可能会搜索到与给定问题无关的知识，因此这种搜索可能会耗费太多的时间。例如，假如仅仅搜索有关

Qantas 的飞行员的知识，就希望避免搜索到有关 Air New Zealand 或 British Airways 的乘客的知识。在这种情况下，就需要框架以便在单一的结构中收集相关的行为。

基本上，框架是专家系统的面向对象编程的一种应用。

什么是面向对象的编程

面向对象的编程可以定义成以对象作为分析、设计和实现基础的编程方法。在面向对象的编程中，对象被定义成一个有清晰边界的概念、抽象或事物，对所有要处理的问题而言是有意义的（Rumbaugh et al.，1991）。所有的对象都有特征，可以彼此清晰地区分开来。例如，Michael Black、Audi 5000 Turbo、HP Pavilion P6555A 等就是对象的例子。

对象在单一实体中将数据结构及其行为结合在一起。这一点和传统的编程截然不同，在传统的编程中，数据结构和程序行为间的关系是隐含或模糊的。

面向对象的编程提供了一种很自然地在计算机中描述真实世界的方式，也具有数据依赖性问题，这一点是传统编程固有的特点（Weisfeld，2008）。当程序员用面向对象的编程语言创建一个对象时，首先要指定对象的名称，然后确定一系列描述对象特征的属性，最后编写指定对象行为的程序。

知识工程师将对象称为框架，框架这个由 Minsky 引入的术语已经成为 AI 的术语，现在对象和框架是同义词。 131 ~ 132

5.2 知识表达技术——框架

框架的概念通过槽的集合定义。每个槽描述了框架的某个属性或操作。在很多方面，框架与传统的包含了实体相关信息的“记录”类似。槽用来存储值，槽可以包含默认值，或指向其他框架，一系列规则或过程的指针，利用指针可以得到相应的槽值。通常，槽可能包含如下几种信息。

1）框架的名称。

2）框架之间的关系。IBM Aptiva S35 框架可能是 *Computer* 类的一个成员，而 *Computer* 类可以属于 *Hardware* 类。

3）槽值。槽的值可能是符号值、数字值或布尔值。例如，在图 5.1 所示的框架中，槽的 *Name* 具有符号值，槽的 *Gate* 为数字值。槽值可以在创建框架时指定，也可以在专家系统会话时指定。

4）默认槽值。在没有发现相反的迹象时，可以将默认值设为真。例如，汽车框架有 4 个轮子，椅子框架有 4 条腿，这些均可以作为相应槽值的默认值。

5）槽值的范围。槽值的范围决定了某个对象或概念是否符合框架定义的原型需求。例如，计算机的成本可以设置在 750 ~ 1500 美元。

6）程序信息。一个槽可以附加一个程序（自包含的任意的计算机编码段），在需要槽值或槽值发生改变时执行该程序。通常有两种类型的程序可以附加到槽上：

（a）槽中加入新信息时执行 WHEN CHANGED 程序。

（b）当解决问题需要信息但还没有指定槽值的时候执行 WHEN CHANGED 程序。

这种附加的程序也称作守护程序。

基于框架的专家系统通过应用侧面来扩展槽值的结构。

什么是侧面

侧面（facet）是为框架的属性提供知识扩展的一种方法。侧面用来建立属性值，控制最终用户的查询，并告知推理引擎如何处理这些属性。

通常，基于框架的专家系统允许将值、提示和推理侧面附加在属性上。值侧面指定属性的默 133

认值和初始值。*提示侧面*（prompt facet）允许在与专家系统的会话中终端用户可以在线输入属性值。最后，推理侧面用于在指定的属性值发生改变时停止推理过程。

将问题恰当地分解为框架、槽和侧面的标准是什么

将问题恰当地分解为框架，将框架分解为槽和侧面取决于问题的本质和知识工程师的判断，因此没有可以预定义的“正确”的表示。

图5.2显示了描述计算机的框架，顶端的框架表示 *Computer* 类，下面的框架描述 *HP Pavilion P6555A* 和 *Acer Aspire M3910* 两个实例。这里用到了两种属性：用于符号信息的字符串类型［Str］和用于数字的数字型［N］。注意，*Desktop* 类中的默认和初始值侧面附加在 Optical Drive、Keyboard、Warranty 和 Stock 槽上，而属性名称、类型、默认和初始值是由实例继承的。

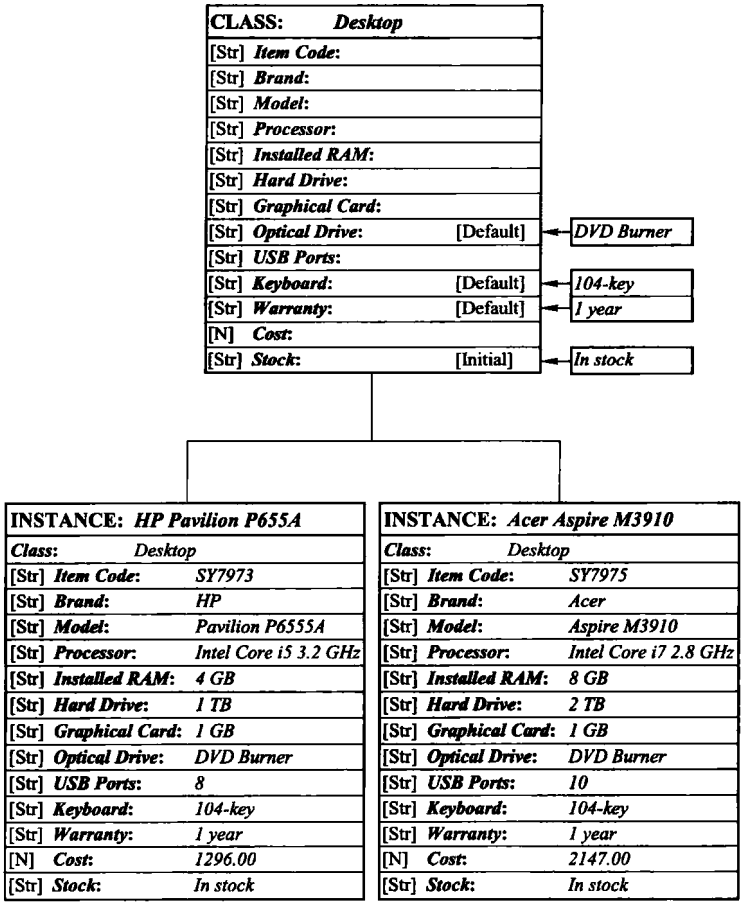


图5.2 *Desktop* 类和实例

什么是类和实例

“框架”这个词的意思很含糊。框架可以指某个对象，如计算机 *HP Pavilion P6555A*，也可以指一组相似的对象。为了更精确一些，可以用*实例框架*（instance-frame）来指某个对象，用*类框架*（class-frame）指一组相似的对象。

类框架描述一组有共同属性的对象，*animal*、*person*、*car* 和 *computer* 都是类框架。在人工智能中，通常用缩写词“类”（class）而不是术语“类框架”。

基于框架的系统中的每一个框架都“知道”它的类。换句话说，框架的类是框架的隐含属性。例如，图5.2的实例在槽 *Class* 中标识了它们的类。

如果对象是基于框架系统的基础，那和类有什么关系

将对象成组放入类中使我们可以用抽象的形式来表示问题。Minsky将框架描述为“表示固定状态的数据结构”。通常，人们很少关心严格地、详尽地定义每个对象的属性，而更关心整个类的显而易见的属性。例如，以鸟为例，鸟会飞吗？答案当然是会。几乎所有的鸟都会飞，因此我们就认为飞的能力是鸟类的一个基本特征，尽管也存在像鸵鸟这样不会飞的鸟。换句话说，和鸵鸟相比，鹰是鸟类更好的成员，因为鹰是鸟类更典型的代表。

基于框架的系统支持类的继承。继承的基本原理是，类框架的属性表示对于类中所有的对象都为真的事物。但是，在实例框架的槽中也可以填写每个实例所特有的实际数据。

来看一个简单的框架结构，如图5.3所示。*Passenger car* 类有几个适合所有汽车的属性，但这个类有太多的不同，以至于几乎没有什么可以填写的公共属性，即使对 *Engine type*、*Drivetrain type* 和 *Transmission type* 等属性加入一些限制也是如此。注意，这些属性是声明为复合的 [C]。复合的属性可以假设成在一组符号值中仅取一个值，例如，属性 Engine type 可以假设在 In-line 4 cylinder 或 V6 中选择一个，但不能同时取这两个值。

134 ~ 135

Mazda 类通过“is-a”关系链接到其超集 *Passenger car*，*Mazda* 类继承了超类的所有属性，并且声明了属性 *Country of manufacture*，附加默认值为 Japan。*Mazda 626* 类引入了3个属性：*Model*、*Colour* 和 *Owner*。最后实例框架 *Mazda DR – 1216* 从 *Mazda* 框架中继承了生产国家的属性，和 *Mazda 626* 一样，为所有复合的属性设置了单一的取值。

实例框架可以覆盖从类框架中继承的属性吗

实例框架可以覆盖或者可以违背继承来的一些属性值。例如，*Mazda 626* 类的平均耗油量是每加仑22英里，但是实例 *Mazda DR – 1216* 的性能可能没那么好，因为它已经跑了很多里程。*Mazda DR – 1216* 框架保留了 *Mazda 626* 类的实例，利用层次结构中上层的属性，即使它和类中的标准值不同。

这种层次结构中框架之间的关系构成了专门化的过程。层次结构顶端的类框架表示一些通用的概念，下面的类框架为一些有限制的概念和更接近实际例子的实例。

对象如何关联到基于框架的系统？关系“is-a”是唯一可用的关系吗

通常，对象之间有3种关系：泛化、聚合和关联。

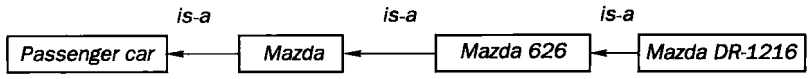
泛化（generalisation）表明超类和子类间的“a-kind-of”或“is-a”关系。例如，汽车是一种交通工具，换句话说，Car 类是更通用的超类 *Vehicle* 的子类。每个子类从其超类中继承所有的特征。

聚合（aggregation）表明是“a-part-of”或“part-whole”关系，几个代表组成部分的子类都为代表整体的超类的一部分。例如，引擎类是汽车类的一部分。

关联（association）描述无关类之间的一些语义关系，例如，Black 先生有房子、车子和计算机。*House*、*Car*、*Computer* 3个类之间是相互独立的，但是它们都通过语义的关联链接到框架 Mr Black 上。

与泛化和聚合关系不同，关联作为动词出现并双向继承。

计算机拥有 Black 先生吗？当然，双向关联的名称可以在一个方向上解读，如“Black 先生拥有一台计算机”，也可以反向解读。拥有的反义词为属于，因为反方向解读应为“计算机属于 Black 先生”。事实上，两个方向上的意义是等价的，是指相同的关联。



a) 汽车框架的关系

CLASS:	*Passenger car*
[C]	*Engine type* In-line 4 cylinder: V6:
[N]	*Horsepower*:
[C]	*Drivetrain type* Rear wheel drive Front wheel drive: Four wheel drive:
[C]	*Transmission type* 5-speed manual: 4-speed automatic:
[N]	*Fuel consumption (mpg)*:
[N]	*Seating capacity*:

Class:	*Mazda*
Superclass:	*Passenger car*
[C]	*Engine type* In-line 4 cylinder: V6:
[N]	*Horsepower*:
[C]	*Drivetrain type* Rear wheel drive Front wheel drive: Four wheel drive:
[C]	*Transmission type* 5-speed manual: 4-speed automatic:
[N]	*Fuel consumption (mpg)*:
[N]	*Seating capacity*:
[Str]	*Country of manufacture*: Japan

CLASS	*Mazda 626*	
Superclass:	*Mazda*	
[C]	*Engine type* In-line 4 cylinder: V6:	
[N]	*Horsepower*:	125
[C]	*Drivetrain type* Rear wheel drive Front wheel drive: Four wheel drive:	
[C]	*Transmission type* 5-speed manual: 4-speed automatic:	
[N]	*Fuel consumption (mpg)*:	22
[N]	*Seating capacity*:	5
[Str]	*Country of manufacture*:	Japan
[Str]	*Model*:	
[C]	*Colour* Glacier White: Sage Green Metallic: Slate Blue Metallic: Black Onyx Clearcoat:	
[Str]	*Owner*:	

INSTANCE	*Mazda DR-1216*	
Class:	*Mazda*	
[C]	*Engine type* In-line 4 cylinder: V6:	 TRUE FALSE
[N]	*Horsepower*:	125
[C]	*Drivetrain type* Rear wheel drive Front wheel drive: Four wheel drive:	 FALSE TRUE FALSE
[C]	*Transmission type* 5-speed manual: 4-speed automatic:	 FALSE TRUE
[N]	*Fuel consumption (mpg)*:	28
[N]	*Seating capacity*:	5
[Str]	*Country of manufacture*:	Japan
[Str]	*Model*:	DX
[C]	*Colour* Glacier White: Sage Green Metallic: Slate Blue Metallic: Black Onyx Clearcoat:	 FALSE TRUE FALSE FALSE
[Str]	*Owner*:	Mr Black

b) 汽车框架和它们的槽

136
~
137

图 5.3 简单框架结构中槽值的继承

图5.4显示了不同对象间的3种关系。

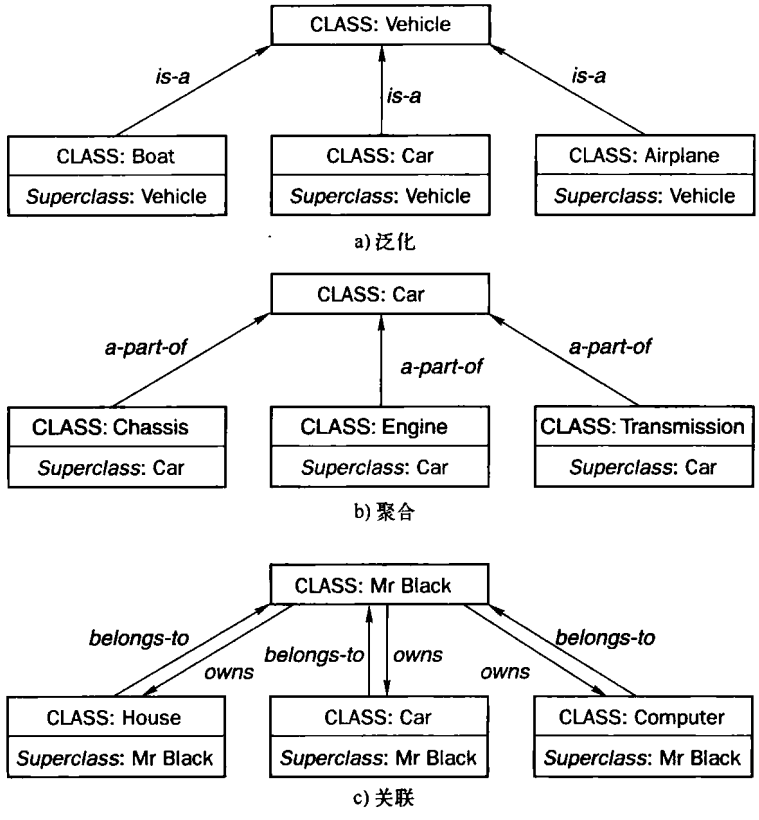


图5.4 对象间的3种关系

5.3 基于框架的系统中的继承

继承是基于框架的系统的基本特征。继承的定义是：类框架的所有特征由实例框架确定的过程。

继承的常规用法是向所有的实例框架强加默认的属性。可以创建一个类框架，其包含一些对象或概念的通用属性，然后再获得一些实例框架而不需要为类级别的特征编码。

基于框架系统的层次结构可视作颠倒的树。最高层的抽象位于树的根部。其下的分支的抽象级别较低，最底层的叶结点为实例框架。每个框架都从高层的所有相关框架继承特征。

图5.5为无污染（Zero-Emission，ZE）交通工具框架的层次结构。根结点 *ZE Vehicle* 有3个分支：*Electric vehicle*、*Solar vehicle* 和 *Muscle vehicle*。接下来沿着其中的一个分支 *Electric vehicle* 往下走，该分支又被细分为 *Car*、*Motorcycle* 和 *Scooter*。接下来 Car 又细分为 *Sedan*、*Van* 和 *Truck*，最底层的叶结点为实例框架 Ford Ecostar。实例 Ford Ecostar 从其父框架中继承所有的特征。

实际上，实例 *Ford Ecostar* 仅有一个父类——类框架 *Van*。此外，在图5.5中，除了根框架 ZE vehicle 外的所有框架仅有一个父类。在这种类型的结构中，每个框架从其父类、祖父类和曾祖父类等继承知识。

框架可以有超过一个的父类吗

在很多问题中，表达与不同世界关联的对象是很自然的。例如，我们希望创建一个人力太阳能电动汽车类。用这种车爬山的时候，人们可以通过踏踏板来启动电力驱动系统，太阳能电池板

会为电力系统充电。因此，框架 Muscle-Solar-Electric 应该结合 3 个类（*Muscle vehicle*、*Solar vehicle* 和 *Electric vehicle*）的特殊的属性。多个父类继承的唯一要求是所有父类的属性必须唯一指定。

138
~
139

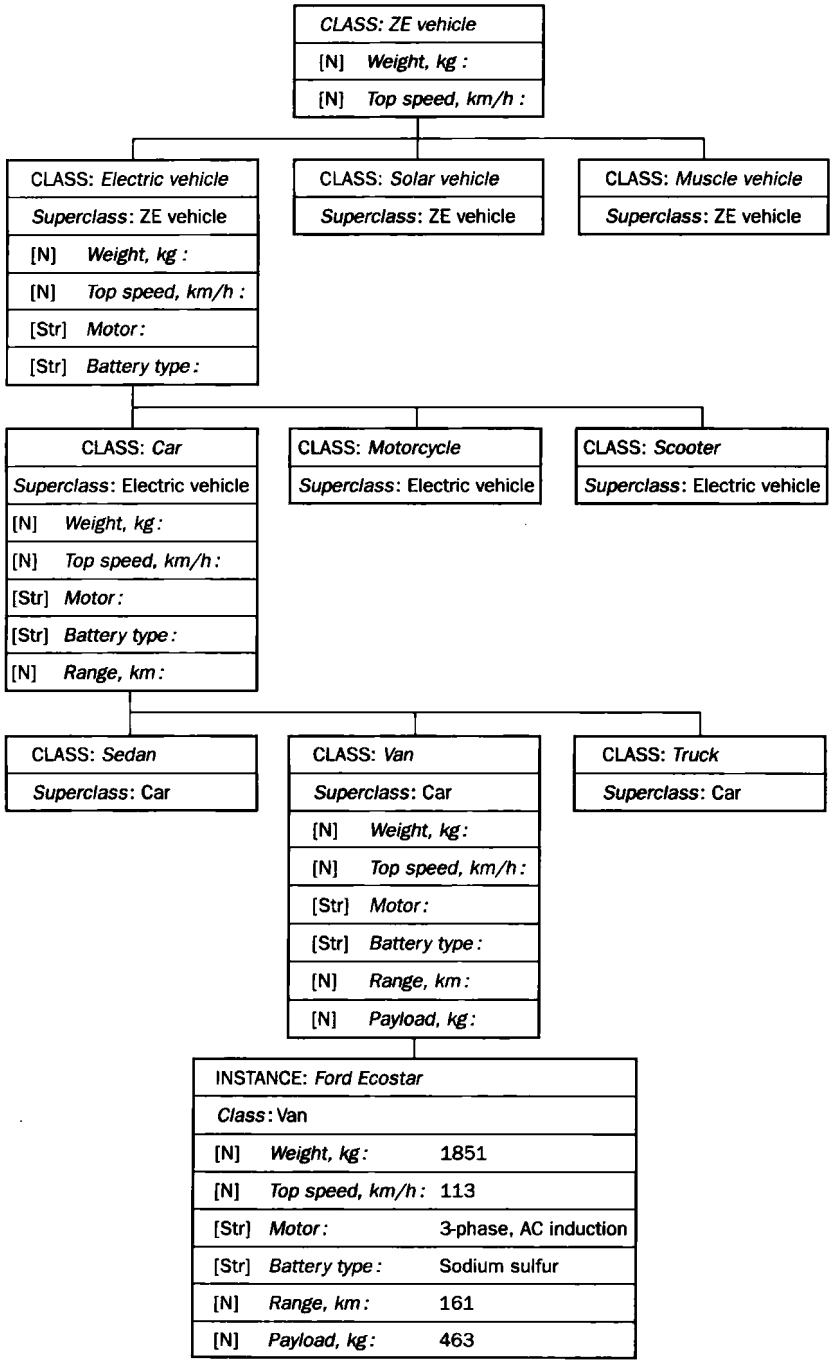


图 5.5 无污染交通工具结构中的单父类继承

在基于框架的系统中，几个类可以使用相同的属性名。但在多重继承中，所有的父类必须有唯一的属性名。例如，要创建父类为 *Muscle vehicle*、*Solar vehicle* 和 *Electric vehicle* 的子类 Muscle-Solar-

Electric，那么需要在父类中剔除 *Weight*、*Top speed* 这样的属性，只有这样才能创建子类。换句话说，要创建多重继承的子类，必须重新考虑系统的整个结构，如图 5.6 所示。

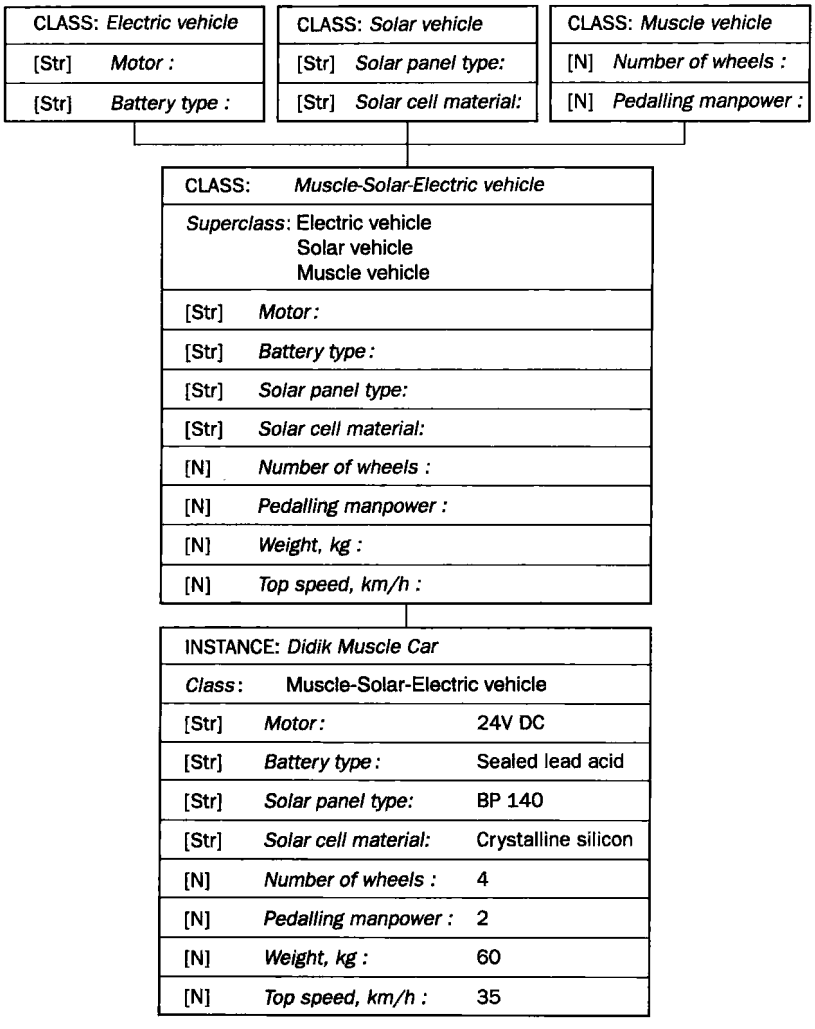


图 5.6 多重继承

在基于框架的系统中，继承意味着代码重用，同时知识工程师的工作就是将相似的类归类，重用公共的代码。继承最重要的优点是概念上的简化，通过在专家系统中减少独立和具体特征的数量可以达到这一目的。

基于框架的专家系统有缺点吗

和上面的例子一样，基于框架的专家系统中很多简化的想法其实在执行阶段都丢失了。Brachman 和 Levesque（2004）证明，如果允许无限制地覆盖继承的属性，就不可能表达确定的语句（例如“所有的正方形都是等边的长方形”）或不确定的普遍的条件（例如“Kasimir Malevich 的所有绘画作品中的正方形都是黑色、红色和白色的”）。通常，基于框架的系统不能区别*本质属性*（实例必须包含的属性，用来区别其是否为该类的成员）和*附加属性*（该类的所有实例只是刚好有这样的属性）。实例继承所有的*典型*属性，并因为这些属性可以在框架层次结构中被覆写，因此使用多重继承时构建组合概念变得难以实现。

看上去这样破坏了整个框架知识表达的思想。然而，框架提供了一种有力的工具，可以结合陈述性知识和过程性知识，虽然这样使知识工程师很难就系统的层次结构和继承途径做出决策。因此，所谓的“典型”特征并不总是实用，因为它使我们得到难以预料的结果。因此，虽然可以使用框架来表达“鸵鸟是一种鸟”这个事实，但鸵鸟并不是典型的鸟类，而在该意义上鹰是
140 ~ 141
鸟类。基于框架的专家系统，例如，在 Level5 Object 中，没有提供阻止创建不一致结构的安全保障，但是这样的系统与传统的程序语言相比能提供更加适合人类推理方式的数据和控制结构。此外，为了结合知识表达的两种技术——规则和框架的优势，现代基于框架的专家系统会在与框架的信息进行交互时使用规则。

5.4 方法和守护程序

前面讨论过，框架提供了一种组织知识的结构化和简洁的方法。但是我们期望专家系统能成为一个聪明的助手——不仅能存储知识，而且可以验证和操纵这些知识。要在框架中增加动作，需要用到方法和守护程序。

什么是方法和守护程序

方法是和框架属性相关的过程，在必要时执行（Durkin，1994）。例如，在 Level5 Object 中，电子数据表程序中的方法用一系列类似于宏的命令来表示。为某个属性编写方法是为了决定属性值或在属性值发生变化时执行一系列动作。

大多数基于框架的专家系统有两种方法：WHEN CHANGED 和 WHEN NEEDED。

通常，*守护程序*的结构为 IF-THEN。它在守护程序 IF 部分的属性值改变时执行。在这种情况下，守护程序和方法是很相似的，两个术语通常作为同义词。但是，需要书写复杂过程时使用方法更加合适，而守护程序的使用仅限于 IF-THEN 语句中。

下面分析一下 WHEN CHANGED 方法。WHEN CHANGED 的属性值发生变化时立即执行。要理解 WHEN CHANGED 如何工作，这里用一个改编自 Sterling 和 Shapiro（1994）的简单例子来说明。这里使用专家系统框架 Level5 Object，因为它提供了大多数基于框架的专家系统和面向对象的编程语言的常用特征。

在评估小商业投资人的贷款申请书时，专家系统应该辅助信贷员。贷款申请书被分为 3 类：“发放贷款”（Give credit）、“拒绝贷款”（Deny credit）和“和上级协商”（Consult a superior），具体如何分类取决于业务担保和金融等级及银行对该笔贷款的预期收益。当信贷员为贷款预期收益的级别定性时，专家系统比较业务担保和贷款申请金额，基于业务资本净值、上年销售增长、销售毛利和销售短期债务评估金融等级，最后确定该贷款申请书的类别。

142 期望专家系统能够提供商业投资的细节、评估系统用户（信贷员）选择的业务的贷款申请。图 5.7 表示申请选择的输入显示。专家系统基于所选的业务，改变显示的数据。

图 5.8 所示的 Action Data 类用来控制输入的显示。用户可以移动到列表中的后一个、前一个、第一个和最后一个申请并检查业务数据。这里 WHEN CHANGED 方法允许我们查看申请列表审批贷款。注意，图 5.8 中的所有属性都声明为简单的[S]，简单属性可以假设值为 TRUE 或 FALSE。下面分析属性 Goto Next 中的 WHEN CHANGED 方法。

这种方法如何工作

在 Level5 Object 中，任何方法都从保留字 WHEN CHANGED 或 WHEN NEEDED 开始，后面紧跟着保留字 BEGIN 和要执行的一系列命令，保留字 END 表示方法结束。要引用方法中的某个属性，必须指定类名和属性名，语法为：

<attribute name> OF <class name>

例如，语句 *Goto Next OF Action Data* 引用 *Action Data* 类的 *Goto Next* 属性。

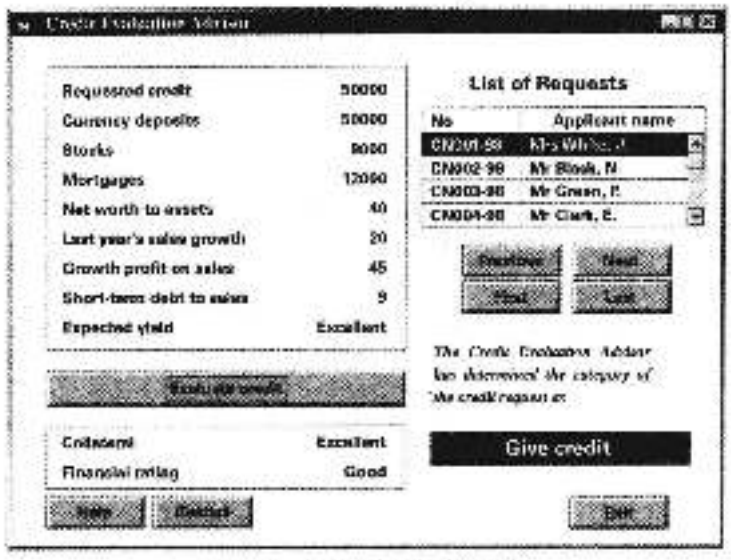


图 5.7 申请选择的输入显示

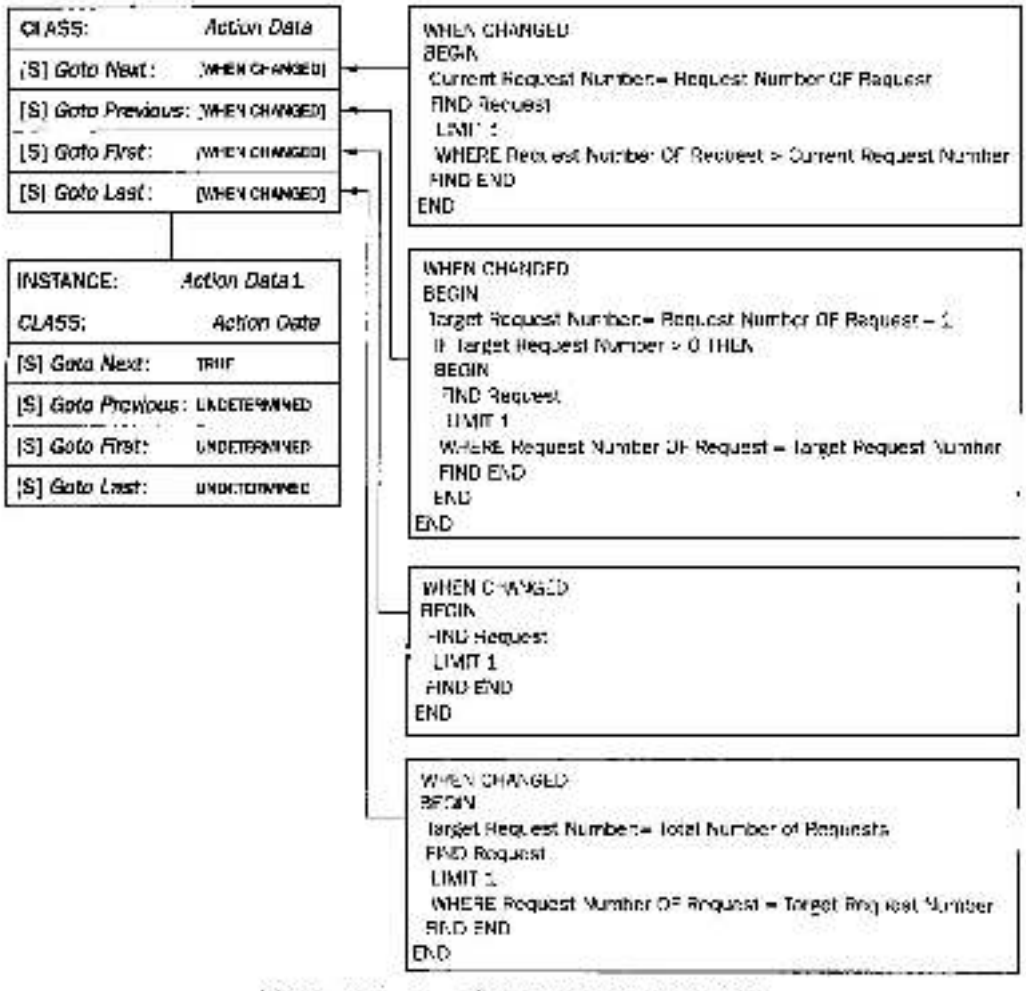


图 5.8 *Action Data* 类和 WHEN CHANGED 方法

输入显示中的按钮 Next 隶属于 *Action Data* 类的属性 *Goto Next*。若运行时选择了该按钮，属性 *Goto Next* 获得值 TRUE，激发了附加在该属性上的 WHEN CHANGED 方法使其执行。该方法的第一个命令将当前选择的 *Request* 类的实例的编号赋予用作参考点的 *Current Request Number* 属性。命令 FIND 使用存储在 *Current Request Number* 中的编号确定列表中的下一个申请。LIMIT 1 命令告诉 Level5 Object 寻找第一个和搜索条件匹配的实例。通过 WHERE 子句

WHERE Request Number OF Request > Current Request Number

确定号码大于 *Current Request Number* 的值的 *Request* 类的实例。申请表按升序来排列，确保可以搜索到正确的实例。例如，如果当前实例号码为 6，则 FIND 命令会找到号码为 7 的实例。

143
~
144

现在分析图 5.9 中的 *Request* 类和实例。实例 Request 1 和 Request 2 有着与 *Request* 类相同的

CLASS: *Request*	
[Str]	*Applicant's name*:
[Str]	*Application no.*:
[N]	*Requested credit*:
[N]	*Currency deposits*:
[N]	*Stocks*:
[N]	*Mortgages*:
[N]	*Net worth to assets*:
[N]	*Last year's sales growth*:
[N]	*Gross profits on sales*:
[N]	*Short-term debt to sales*:
[C]	*Expected yield*: Excellent: Reasonable: Poor:
[N]	*Request Number*:

INSTANCE: **Request 1**		
CLASS: *Request*		
[Str]	*Applicant's name*:	Mrs White, J.
[Str]	*Application no.*:	CN001-98
[N]	*Requested credit*:	50000
[N]	*Currency deposits*:	50000
[N]	*Stocks*:	9000
[N]	*Mortgages*:	12000
[N]	*Net worth to assets*:	40
[N]	*Last year's sales growth*:	20
[N]	*Gross profits on sales*:	45
[N]	*Short-term debt to sales*:	9
[C]	*Expected yield*: Excellent: TRUE Reasonable: FALSE Poor: FALSE	
[N]	*Request Number*:	1

INSTANCE: **Request 2**		
CLASS: *Request*		
[Str]	*Applicant's name*:	Mr Black, N.
[Str]	*Application no.*:	CN002-98
[N]	*Requested credit*:	75000
[N]	*Currency deposits*:	45000
[N]	*Stocks*:	10000
[N]	*Mortgages*:	20000
[N]	*Net worth to assets*:	45
[N]	*Last year's sales growth*:	25
[N]	*Gross profits on sales*:	35
[N]	*Short-term debt to sales*:	10
[C]	*Expected yield*: Excellent: FALSE Reasonable: TRUE Poor: FALSE	
[N]	*Request Number*:	2

145

图 5.9 *Request* 类及其实例

属性。但每个实例的属性值不同。为了在输入显示中显示属性的取值，需要创建取值框（显示数据的项），并将取值框绑定在对应的属性上。运行该应用程序时，取值框显示 *Request* 类当前所选 Request 类的实例属性值，并且 WHEN CHANGED 方法激发相应的动作。

什么时候使用 WHEN NEEDED 方法

在很多应用程序中，属性有初始值或默认值。但是，在有些应用程序中，WHEN NEEDED 方法仅在需要时才用于获得属性值。换句话说，只有需要使用与某个属性相关的信息来解决问题时才执行 WHEN NEEDED 方法，但属性值是不确定的。在后面讨论信用评估的规则的例子时会再探讨这种方法。

5.5 框架和规则的交互

大多数基于框架的专家系统允许用一系列的规则来评估框架中的信息。

基于规则的专家系统中使用的规则和基于框架的专家系统中使用的规则有什么不同

每个规则都具有 IF-THEN 的结构，同时每个规则将在其 IF 部分提供相关的信息或事实与 THEN 部分的某些动作相关联。在这种情况下，基于规则的专家系统中的规则和基于框架的专家系统中的规则没有什么不同。但在基于框架的专家系统中，规则通常使用模式匹配子句。这些子句包含用于在所有实例框架中找到匹配条件的变量。

在基于框架的专家系统中推理引擎如何工作？是什么激发了规则

再次比较一下基于规则和基于框架的专家系统。在基于规则的专家系统中，推理引擎将包含在知识库中的规则和数据库中给定的数据链接起来。如果目标已经设置，或者换句话说，当专家系统收到指令去确定某个具体对象的取值时，推理引擎搜索知识库，寻找在后项（THEN 部分）中含有目标的规则。如果找到这样的规则，其前项（IF 部分）和数据库中的数据能够匹配，则激发该规则，同时指定的对象（即目标）获得取值。如果没有找到得到能用于目标的值的规则，系统就向用户询问，要求用户提供该值。

在基于框架的专家系统中，推理引擎也会为目标或者指定的属性进行搜索，直到得到属性的值为止。

在基于规则的专家系统中，目标为规则库而定义的。在基于框架的系统中，规则只是辅助的角色，框架是知识的主要来源，方法和守护程序都是为了在框架中增加动作。因此，可以预期，基于框架的系统的目标可以在方法中建立也可以在守护程序中建立。现在我们回到信用评估的例子中。 146

假设需要评估用户选择的贷款申请书。当用户按下输入显示界面上的 Evaluate Credit 按钮后，专家系统就应开始评估。该按钮附属于如图 5.10 所示的 *Credit Evaluation* 类的 *Evaluate Credit* 属性。而 *Evaluate Credit* 属性有附属的 WHEN CHANGED 方法。当运行时按下了 Evaluate Credit 按钮，属性 *Evaluate Credit* 接收到新值 TRUE。这个改变激发了 WHEN CHANGED 方法进行执行。PURSUE 命令告诉 Level5 Object 建立 *Credit Evaluation* 类的属性 *Evaluation* 的值。图 5.11 显示了确定属性值的一系列简单规则。

在实例中，推理引擎是如何工作的

对于目标 Evaluation OF Credit Evaluation，推理引擎找到后项包含搜索所需的目标规则，并按照其在规则库出现的顺序逐个进行分析。也就是说，推理引擎从 RULE 9 开始，试图证实属性 *Evaluation* 是否接收到 *Give credit* 的值。这可以通过分析每条规则前项的有效性来实现。换句话说，推理引擎尝试首先确定属性 *Collateral* 的值是否为 Excellent，其次确定属性 *Financial rating* 的值是否为 Excellent。要确定 Collateral OF Credit Evaluation 的值是否为 Excellent，推理引擎分析 RULE 1 和 RULE 2 并确定 Financial rating OF Credit Evaluation 是否为 Excellent，如 RULE 8 所示。

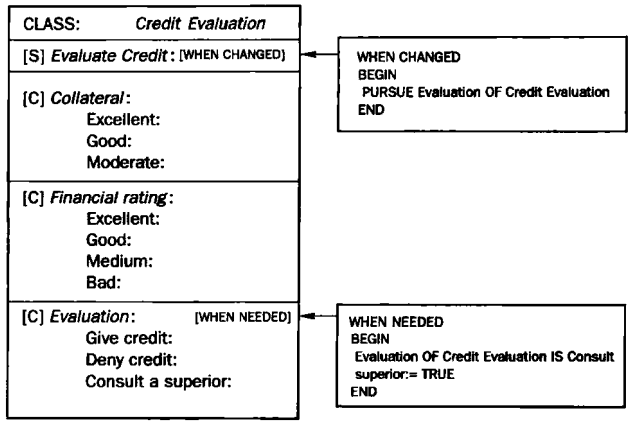


图 5.10 *Credit Evaluation* 类、WHEN CHANGED 和 WHEN NEEDED 方法

RULE 1
IF Currency deposits OF Request >= Requested credit OF Request
THEN Collateral OF Credit Evaluation IS Excellent

RULE 2
IF Currency deposits OF Request >= Requested credit OF Request * 0.7
AND (Currency deposits OF Request + Stocks OF Request) >= Requested credit OF Request
THEN Collateral OF Credit Evaluation IS Excellent

RULE 3
IF (Currency deposits OF Request + Stocks OF Request) > Requested credit OF Request * 0.6
AND (Currency deposits OF Request + Stocks OF Request) < Requested credit OF Request * 0.7
AND (Currency deposits OF Request + Stocks OF Request + Mortgages OF Request) >= Requested credit OF Request
THEN Collateral OF Credit Evaluation IS Good

RULE 4
IF (Currency deposits OF Request + Stocks OF Request + Mortgages OF Request) <= Requested credit OF Request
THEN Collateral OF Credit Evaluation IS Moderate

RULE 5
IF Net worth to assets OF Request * 5 + Last year's sales growth OF Request + Gross profits on sales OF Request * 5 + Short term debt to sales OF Request * 2 <= -500
THEN Financial rating OF Credit Evaluation IS Bad

RULE 6
IF Net worth to assets OF Request * 5 + Last year's sales growth OF Request + Gross Profits on sales OF Request * 5 + Short term debt to sales OF Request * 2 >= -500
AND Net worth to assets OF Request * 5 + Last year's sales growth OF Request + Gross profits on sales OF Request * 5 + Short term debt to sales OF Request * 2 <= 150
THEN Financial rating OF Credit Evaluation IS Medium

RULE 7
IF Net worth to assets OF Request * 5 + Last year's sales growth OF Request + Gross Profits on sales OF Request * 5 + Short term debt to sales OF Request * 2 >= 150
AND Net worth to assets OF Request * 5 + Last year's sales growth OF Request + Gross profits on sales OF Request * 5 + Short term debt to sales OF Request * 2 <= 1000
THEN Financial rating OF Credit Evaluation IS Good

RULE 8
IF Net worth to assets OF Request * 5 + Last year's sales growth OF Request + Gross Profits on sales OF Request * 5 + Short term debt to sales OF Request * 2 > 1000
THEN Financial rating OF Credit Evaluation IS Excellent

RULE 9
IF Collateral OF Credit Evaluation IS Excellent
AND Financial rating OF Credit Evaluation IS Excellent
THEN Evaluation OF Credit Evaluation IS Give Credit

RULE 10
IF Collateral OF Credit Evaluation IS Excellent
AND Financial rating OF Credit Evaluation IS Good
THEN Evaluation OF Credit Evaluation IS Give Credit

RULE 11
IF Collateral OF Credit Evaluation IS Moderate
AND Financial rating OF Credit Evaluation IS Medium
THEN Evaluation OF Credit Evaluation IS Deny Credit

RULE 12
IF Collateral OF Credit Evaluation IS Moderate
AND Financial rating OF Credit Evaluation IS Bad
THEN Evaluation OF Credit Evaluation IS Deny Credit

图 5.11 信用评估的规则

如果所有的规则前项都有效，则推理引擎得到结论：Evaluation OF Credit Evaluation 是 Give credit。但是，如果前项的任何部分是无效的，那么该结论就是无效的。在这种情况下，推理引擎会执行下一个规则 RULE 10，它可以确定属性 Evaluation 的取值。

如果 Collateral OF Credit Evaluation 是 Good 怎么办

基于信用评估的一系列规则，推理引擎在有些情况下不能确定属性 Evaluation 的值。在业务担保取值为 Good，同时业务的金融等级为 Excellent 或者 Good 时就会出现这种情况。实际上，如果看一下图 5.10 就会发现，这种情况在规则库中没有体现。但是也不能完全依赖于一系列规则，可以使用 WHEN NEEDED 方法来确定属性的取值。

147 ~ 148

图 5.10 所示的 WHEN NEEDED 方法附加在属性 Evaluation 上。当推理引擎需要确定 Evaluation 的取值时执行该方法。当 WHEN NEEDED 方法执行时，属性 Evaluation 得到值 Consult a superior。

推理引擎如何得知在哪里、按照什么顺序来获取属性的值

在本例中，如果首先执行 WHEN NEEDED 方法，则属性 Evaluation 总是获得值 Consult a superior，同时不会激发任何规则。因此，推理引擎仅在无法从规则库中得到属性值时才使用 WHEN NEEDED 方法。换句话说，要首先确定属性值的搜索顺序。例如，可以通过附加在属性上的 SEARCH ORDER 实现，该属性告知推理引擎哪里、以何顺序获得属性的值。

在 Level5 Object 中，每个属性都可以指定搜索顺序，在评估信用的例子中，为 Evaluation 取值设置的搜索顺序是 RULES、WHEN NEEDED。因此推理引擎会先从规则库中开始搜索。

5.6 基于框架的专家系统实例：Buy Smart

本节展示了一个简单的专家系统的例子来说明上面的讨论内容。这个基于框架的专家系统名为 Buy Smart，用于为购房者提供建议。

我们将回顾开发基于框架的专家系统的主要步骤，并展示如何通过方法和守护程序来使用框架。实例中使用了专家系统框架 Level5 Object。

开发基于规则的专家系统和基于框架的专家系统的主要步骤有什么不同吗

开发这两种系统的主要步骤基本相同。首先，知识工程师要大致理解问题和完整的知识结构，接着确定使用什么样的专家系统工具来开发原型系统。然后知识工程师创建知识库，通过运行一些测试问题来测试知识库。最后，扩展、测试和修正专家系统，直到它能够满足用户的需要为止。

设计基于规则的专家系统和基于框架的专家系统的主要区别是在系统中如何理解和表达知识。

在基于规则的专家系统中，用一系列的规则来表达解决问题需要的领域知识。每个规则都获得问题的某些启发式方法，同时每次加入新的规则都会增加新的知识，系统由此变得更加聪明。基于规则的专家系统可以通过改变、增加或减少规则来轻松地加以修改。 149

在基于框架的专家系统中，问题是按照不同的方式分析的。这里，首先确定知识的完整的分层框架。然后确定类及其属性，同时各层框架之间的关系也要确定下来。基于框架的专家系统的体系结构不仅要提供问题的自然描述，也要允许通过方法和守护程序为框架增加动作。

基于框架的专家系统的开发过程主要包含下面几个步骤：

1）指定问题并定义系统的范围。

2）定义类及属性。

3）定义实例。

4）设计显示。

5）定义 WHEN CHANGED 和 WHEN NEEDED 方法及守护程序。

6）定义规则。

7）评价并扩展系统。

步骤 1：指定问题并定义系统的范围。

在 Buy Smart 的例子中，先收集本地区的一些代售房地产的信息。要确定相关的细节，例如房产类型、位置、卧室和浴室的数量，当然还有价格。应该提供关于房产的简短描述和精美照片。

可以预料有些房产会售出，新的房源会出现。因此，要建立易于修改并能从专家系统访问的数据库。Level5 Object 允许访问、修改、删除数据库 dBASE Ⅲ中的数据，并执行一些其他动作。

可以在数据库中存储房产的描述和照片吗

房产的描述和照片应该分开存储，描述应存储为文本文件（*.txt），照片存储为位图文件（*bmp）。接下来如果建立包含文本框和图片框的显示界面，就需要通过将文本文件读取到文本框中，将位图文件读取到图片框中来让用户查看房产描述和照片。

现在使用 dBASE Ⅲ或 Microsoft Excel 建立外部数据库文件 house. dbf，如表 5. 1 所示。

表 5. 1 数据库 house. dbf 的属性

区域	城市	价格	类型	卧室
Central Suburbs	New Town	164 000	House	3
Central Suburbs	Taroona	150 000	House	3
Southern Suburbs	Kingston	225 000	Townhouse	4
Central Suburbs	North Hobart	127 000	House	3
Northern Suburbs	West Moonah	89 500	Unit	2
Central Suburbs	Taroona	110 000	House	3
Central Suburbs	Lenah Valley	145 000	House	3
Eastern Shore	Old Beach	79 500	Unit	2
Central Suburbs	South Hobart	140 000	House	3
Central Suburbs	South Hobart	115 000	House	3
Eastern Shore	Cambridge	94 500	Unit	2
Northern Suburbs	Glenorchy	228 000	Townhouse	4
—	—	—	—	—
—	—	—	—	—
—	—	—	—	—

浴室	结构	电话	图片文件	文本文件
1	Weatherboard	(03) 6226 4212	house01. bmp	house01. txt
1	Brick	(03) 6226 1416	house02. bmp	house02. txt
2	Brick	(02) 6229 4200	house03. bmp	house03txt
1	Brick	(03) 6226 8620	house04. bmp	house04. txt
1	Weatherboard	(03) 6225 4666	house05. bmp	house05. txt
1	Brick	(03) 6229 5316	house06. bmp	house06. txt
1	Brick	(03) 6278 2317	house07. bmp	house07. txt
1	Brick	(03) 6249 7298	house08. bmp	house08. txt
1	Brick	(03) 6228 5460	house09. bmp	house09. txt
1	Brick	(03) 6227 8937	house10. bmp	house10. txt
1	Brick	(03) 6248 1459	house11. bmp	house11. txt
2	Weatherboard	(03) 6271 6347	house12. bmp	house12. txt
—	—	—	—	—
—	—	—	—	—
—	—	—	—	—

接下来就要列出能够想到的所有查询：

- 你想在房产上最多花多少钱？
- 你最喜欢什么类型的房产？
- 你最喜欢生活在哪个地区？
- 你想要几个卧室？
- 你想要几个浴室？

这些问题确定后，专家系统就要提供一系列合适的房源。

步骤2：定义类及其属性。

这里要确定解决问题的主要类。首先从通用的或概念上的类开始。例如，分析房产的概念并描述对大多数房产都适合的通用特征。可以通过位置（Area）、价格（Price）、类型（Type）、卧室（Bedroom）和浴室（Bathroom）的数量、结构（Construction）、图片（Pictfile）和描述（Textfile）等特点来介绍每一个房源。还需要列出详细的联系方法，例如房子的地址（Address）或电话号码（Phone）。因此，Property 类如图 5.12 所示。注意，图中增加了实例号（*Instance Number*）属性，这个属性不是用来描述房产的，而是辅助 Level5 Object 访问外部数据库的。

150 ~ 151

CLASS: **Property**
[Str] *Area*:
[Str] *Suburb*:
[N] *Price*:
[Str] *Type*:
[N] *Bedrooms*:
[N] *Bathrooms*:
[Str] *Construction*:
[Str] *Phone*:
[Str] *Pictfile*:
[Str] *Textfile*:
[N] *Instance Number*:

INSTANCE: **Property 1**	
CLASS: *Property*	
[Str] *Area*:	Central Suburbs
[Str] *Suburb*:	New Town
[N] *Price*:	164000
[Str] *Type*:	House
[N] *Bedrooms*:	3
[N] *Bathrooms*:	1
[Str] *Construction*:	Weatherboard
[Str] *Phone*:	(03) 6226 4212
[Str] *Pictfile*:	house01.bmp
[Str] *Textfile*:	house01.txt
[N] *Instance Number*:	1

INSTANCE: **Property 2**	
CLASS: *Property*	
[Str] *Area*:	Central Suburbs
[Str] *Suburb*:	Taroona
[N] *Price*:	150000
[Str] *Type*:	House
[N] *Bedrooms*:	3
[N] *Bathrooms*:	1
[Str] *Construction*:	Brick
[Str] *Phone*:	(03) 6226 1416
[Str] *Pictfile*:	house02.bmp
[Str] *Textfile*:	house02.txt
[N] *Instance Number*:	2

图 5.12 *Property* 类及其实例

步骤3：定义实例。

一旦确定了 Property 类框架，就可以很容易地通过使用存储在 dBASE Ⅲ数据库中的数据创建
152 它的实例。对于大多数类似于 Level5 Object 的基于框架的专家系统而言，需要告诉系统我们想要创建一个新实例。例如，可以使用下面的代码来创建 Property 类的新实例：

```
MAKE Property
    WITH Area := area OF dB3 HOUSE 1
    WITH Suburb := suburb OF dB3 HOUSE 1
    WITH Price := price OF dB3 HOUSE 1
    WITH Type := type OF dB3 HOUSE 1
    WITH Bedrooms := bedrooms OF dB3 HOUSE 1
    WITH Bathrooms := bathrooms OF dB3 HOUSE 1
    WITH Construction := construct OF dB3 HOUSE 1
    WITH Phone := phone OF dB3 HOUSE 1
    WITH Picfile := picfile OF dB3 HOUSE 1
    WITH Txtfile := txtfile OF dB3 HOUSE 1
    WITH Instance Number := Current Instance Number
```

这里，类 dB3 HOUSE 1 用来描述外部数据库文件 house. dbf 的结构。如表 5.1 所示，房产数据库的每一行表示 Property 类的一个实例，每一列表示一个属性。新创建的实例框架从数据库的当前记录获取值。图 5.12 显示了从外部数据库创建的实例。这些实例链接到 Property 类中，它们继承该类的所有属性。

步骤4：设计显示。

一旦确定了主要的类和属性，就可以为应用程序设计主要的界面了。需要在每个应用开始时给出应用程序标题显示（Application Title Display），告诉用户一些适用的信息。显示界面可以包含应用程序标题、问题的通用描述、有代表性的图片和版权信息。图 5.13 为应用程序标题显示的例子。

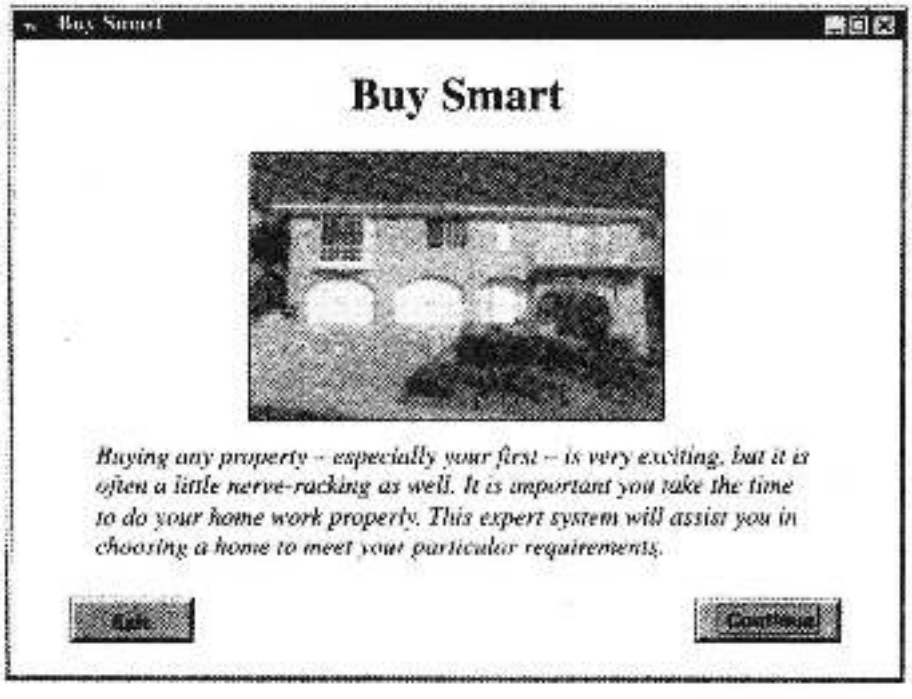


图 5.13 应用程序标题显示

可以想到的第二个显示界面是查询显示（Query Display）。这个界面应该允许用户通过回答专家系统提出的问题表明我们的要求，查询显示看上去如图 5.14 所示。在这里，要求用户选择他（她）在寻找房产时最重要的要求。基于这些选择，专家系统接下来可以给出合适的房产的完整列表。

最后应该设计房产信息显示（Property Information Display）界面。该显示界面可以提供一系列适合的房产，并可以移到列表中的后一个、前一个、第一个和最后一个房产，还能够看到房产的图片和描述。该界面如图 5.15 所示。注意，要在显示界面中包含文本框和图片框。

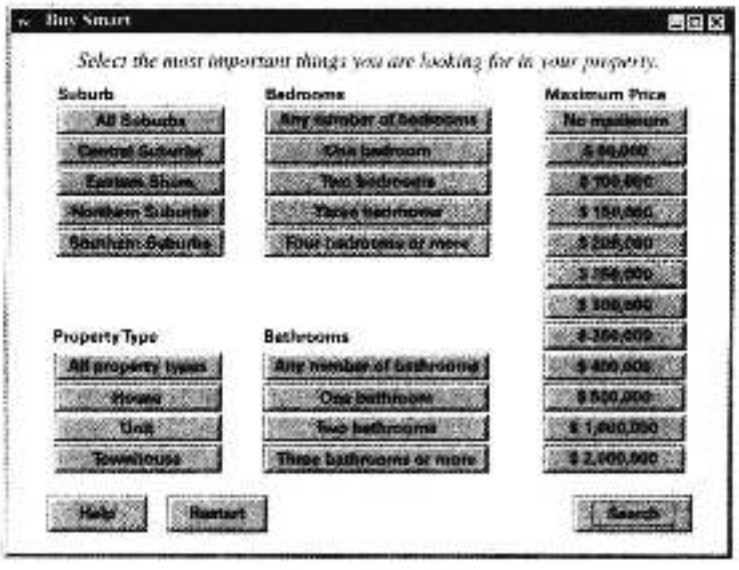


图 5.14 查询显示

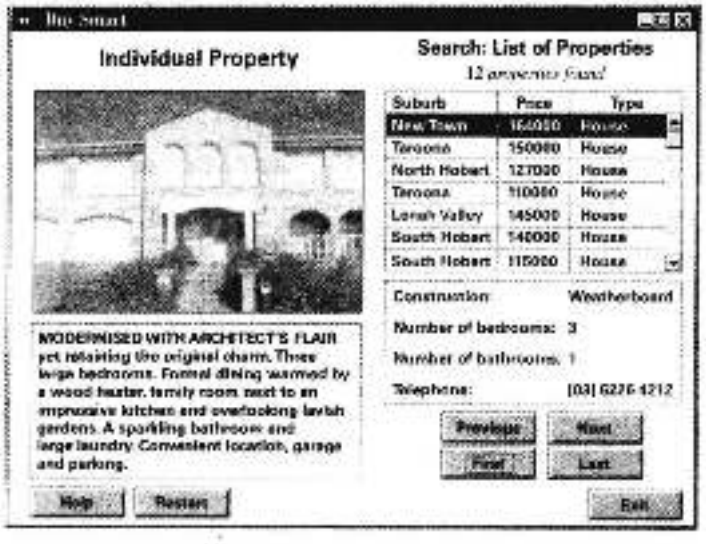


图 5.15 房产信息显示

如何将这些显示链接起来

Level5 Object 允许我们通过在应用程序标题显示界面上放置 Continue 按钮将其链接到查询显示界面上，并在查询显示界面上放置 Search 按钮将其链接到房产信息显示界面。在运行应用程

序时，单击 Continue 或 Search 按钮就会出现新的显示界面。

现在我们来实现这些显示界面。

步骤5：定义 WHEN CHANGED 和 WHEN NEEDED 方法及守护程序。

到目前为止，已经创建了主要的类和相关属性，也确定了类实例，建立了从外部数据库中创建实例的机制，最后设计了为客户显示信息的静态界面。接下来必须研究一种方法来实现这个应用程序。有两种方式可以实现这个任务，第一个方式是使用 WHEN CHANGED 和 WHEN NEEDED 方法及守护程序；第二个方式则与模式匹配规则有关。在基于框架的系统中，通常首先考虑方法和守护程序的应用。

现在要确定的是什么时候创建 Property 类的实例。有两个解决方案。第一个方案是当用户单击应用程序标题显示界面的 Continue 按钮时，创建所有的实例，然后再根据用户的喜好（当用户选择查询显示页面上的按钮时），一步步删除用户不合适的实例。

155 第二个方法是用户在查询显示页面上完成所有的选择后，仅创建相关的实例。这种方法说明了去除 Property 类中不合适的实例的必要性，但可能会增加系统设计的复杂性。

在本设计中，推荐使用第一种方法。该方法会给我们提供使用守护程序而非规则的机会。不过也可以使用其他方法。

现在创建另一个类 *Action Data*，如图 5.16 所示。*Load Properties* 属性附加了 WHEN CHANGED 方法，允许创建 Property 类的所有实例。

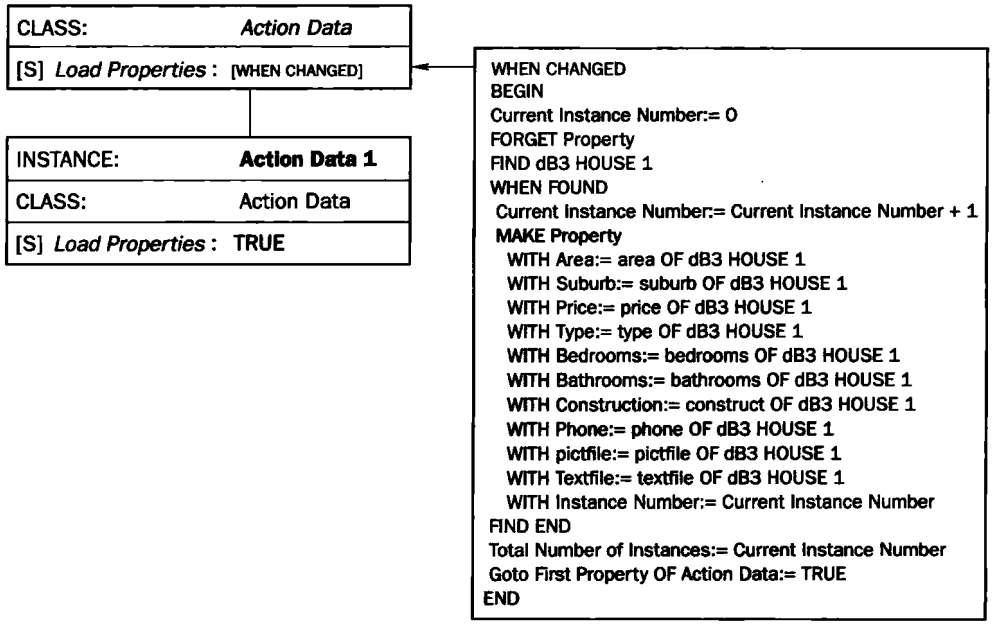


图 5.16 *Load Properties* 属性的 WHEN CHANGED 方法

如何使用这种方法

为了让这种方法工作，应将应用程序标题显示界面的 Continue 按钮和 *Load Properties* 属性链接起来。当运行时按下这个按钮时，*Load Properties* 属性获取值 TRUE，导致 WHEN CHANGED 方法执行，并创建 Property 类的所有实例。创建的实例数量和外部数据库中记录的数量相等。

接下来出现查询显示界面（记住已将应用程序标题显示界面的 Continue 按钮和 Load Properties

属性链接起来），用户需要通过选择合适的按钮找到最合心意的房产特征。本例中，每个按钮都和一个守护程序链接起来，去除 Properties 类中不合适的实例。一系列的守护程序如图 5.17 所示。

这里的守护程序如何工作

直到发生了某些事件，守护程序才会执行。在本例中，这意味着仅在用户按下相应按钮时才激发守护程序。 [156]

```
DEMON 1
IF selected OF Central Suburbs pushbutton
THEN FIND Property
   WHERE Area OF Property <> "Central Suburbs"
   WHEN FOUND
     FORGET CURRENT Property
 FIND END

DEMON 2
IF selected OF Eastern Shore pushbutton
THEN FIND Property
   WHERE Area OF Property <> "Eastern Shore"
   WHEN FOUND
     FORGET CURRENT Property
 FIND END
        :
DEMON 5
IF selected OF House pushbutton
THEN FIND Property
   WHERE Type OF Property <> "House"
   WHEN FOUND
     FORGET CURRENT Property
 FIND END
        :
DEMON 9
IF selected OF One bedroom pushbutton
THEN FIND Property
   WHERE Bedrooms OF Property <> 1
   WHEN FOUND
     FORGET CURRENT Property
 FIND END
        :
DEMON 12
IF selected OF One bathroom pushbutton
THEN FIND Property
   WHERE Bathrooms OF Property <> 1
   WHEN FOUND
     FORGET CURRENT Property
 FIND END
        :
DEMON 15
IF selected OF $50000 pushbutton
THEN FIND Property
   WHERE Price OF Property > 50000
   WHEN FOUND
     FORGET CURRENT Property
 FIND END
```

图 5.17 查询显示界面的守护程序

例如，DEMON 1 和 Central Suburbs 按钮相关。当用户单击查询显示界面中的 Central Suburbs 按钮时，激发 DEMON 1。然后守护程序发出第一个命令告诉 Level5 Object 去寻找 Property 类。WHERE 子句

```
WHERE Area OF Property <> "Central Suburbs"
```

找到 Property 类所有的实例，但这些实例都和用户的选择不匹配。系统搜索属性 *Area* 的值和 Central Suburbs 不等的所有的实例。然后 FORGET CURRENT 命令从应用程序中去除 Property 类的当前实例。 [157]

一旦选择了房产的特征，用户单击查询显示界面中的 Search 按钮，获得具有这些特征的房产列表。该列表显示在房产信息显示界面中（回忆一下，Search 按钮已经附加到房产信息显示界面里了）。

能够看到房产的图片和描述吗

首先再为图 5.18 所示的 Action Data 类创建两个属性 *Load Instance Number* 和 *Goto First Property*。同时也将查询显示页面的 Search 按钮附加在属性 *Load Instance Number* 上。当运行中单击 Search 按钮时，属性 *Load Instance Number* 会接收到值 TRUE，激发 WHEN CHANGED 方法执行。这种方法可以确定 Property 类中实例的总数。它还将属性 *Goto First Property* 的值设为 TRUE，随后激发了其 WHEN CHANGED 方法执行。

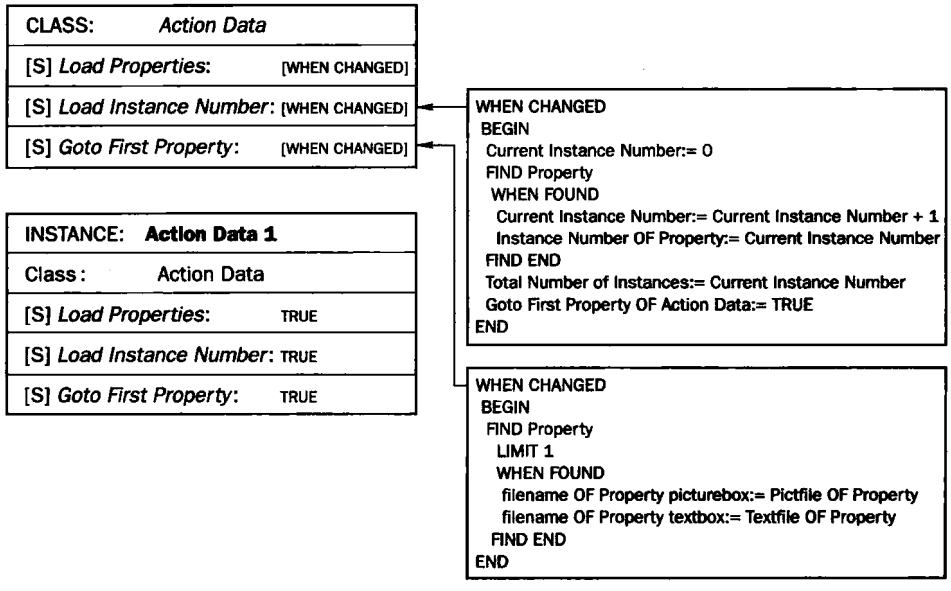


图 5.18 属性 Load Instance Number 和 Goto First Property 的 WHEN CHANGED 方法

将方法附加在属性 *Goto First Property* 上可以保证在用户进入房产信息显示页面时总是处于第一处房产的位置上。它还将 *Pictfile* 属性的值加载到界面的图片框，并把属性 Textfile 的值加载到
158 文本框。因此，用户就可以看到房产的图片和描述，如图 5.15 所示。

步骤 6：定义规则。

在设计基于框架的专家系统时，最重要和最困难的决策是使用规则还是使用方法和守护程序进行管理。制定这个决策主要取决于设计者自己的喜好。本例使用方法和守护程序，因为这种方法在表示过程中功能强大，而且简单。另一方面，在前面的信用评估例子中，已经使用了规则。通常，规则处理过程性的知识不是很有效。

步骤 7：评价并扩展系统。

现在已经完成了 Buy Smart 专家系统的最初设计，接下来的任务就是评价它。我们要确保系统的性能和预期一致。换句话说，要运行一个测试用例。

1）通过单击应用程序标题显示页面的 Continue 按钮来开始测试。Action Data 类的属性 Load Propertise 接收到值 TRUE，执行附加在 Load Propertise 上的方法 WHEN CHANGED，并创建 Property 类的所有实例。

2）出现查询界面，用户做出选择，例如：

```
⇒ Central Suburbs
DEMON 1
IF selected OF Central Suburbs pushbutton
THEN FIND Property
  WHERE Area OF Property <> "Central Suburbs"
  WHEN FOUND
    FORGET CURRENT Property
 FIND END

⇒ House
DEMON 5
IF selected OF House pushbutton
THEN FIND Property
  WHERE Type OF Property <> "House"
  WHEN FOUND
    FORGET CURRENT Property
 FIND END

⇒ Three bedrooms
DEMON 10
IF selected OF Three bedroom pushbutton
THEN FIND Property
  WHERE Bedrooms OF Property <> 3
  WHEN FOUND
    FORGET CURRENT Property
 FIND END

⇒ One bathroom
DEMON 12
IF selected OF One bathroom pushbutton
THEN FIND Property
  WHERE Bathrooms OF Property <> 1
  WHEN FOUND
    FORGET CURRENT Property
 FIND END

⇒ $ 200,000
DEMON 18
IF selected OF $200,000 pushbutton
THEN FIND Property
  WHERE Price OF Property > 200000
  WHEN FOUND
    FORGET CURRENT Property
 FIND END
```

159

守护程序将不符合选择的 Property 实例去除。

3）单击 Search 按钮，*Action Data* 类的 *Load Instance Number* 属性获得取值 TRUE，附加在 *Load Instance Number* 上的 WHEN CHANGED 方法被执行，以确定 Property 类中剩余的实例数量，并设定属性 *Goto First Property* 的值为 TRUE。然后激发附加在 *Goto First Property* 属性上的 WHEN CHANGED 方法，找到第一个 Property 实例，指定 Property Picturebox 的 *filename* 属性取值为 house01. bmp，Property textbox 的 *filename* 属性取值为 house01. txt（前面已在属性信息显示界面中创建了 Property Picturebox 和 Property textbox）。

4）出现属性信息显示界面。如图 5. 15 所示，可以分析 12 个满足要求的房产。注意，我们首先从房产列表的第一个房产开始，房产的图片显示在图片框中，房产的描述显示在文本框中。但是，不能通过界面中的按钮来移动后一个、前一个、第一个和最后一个。要想实现移动，需要创建 *Action Data* 类的附加属性，并将 WHEN CHANGED 方法附加新建的属性上面，如图 5. 19 所示。

现在，Busy Smart 专家系统已经可以进行扩展了，可以将新的房源添加到外部数据库中了。

160

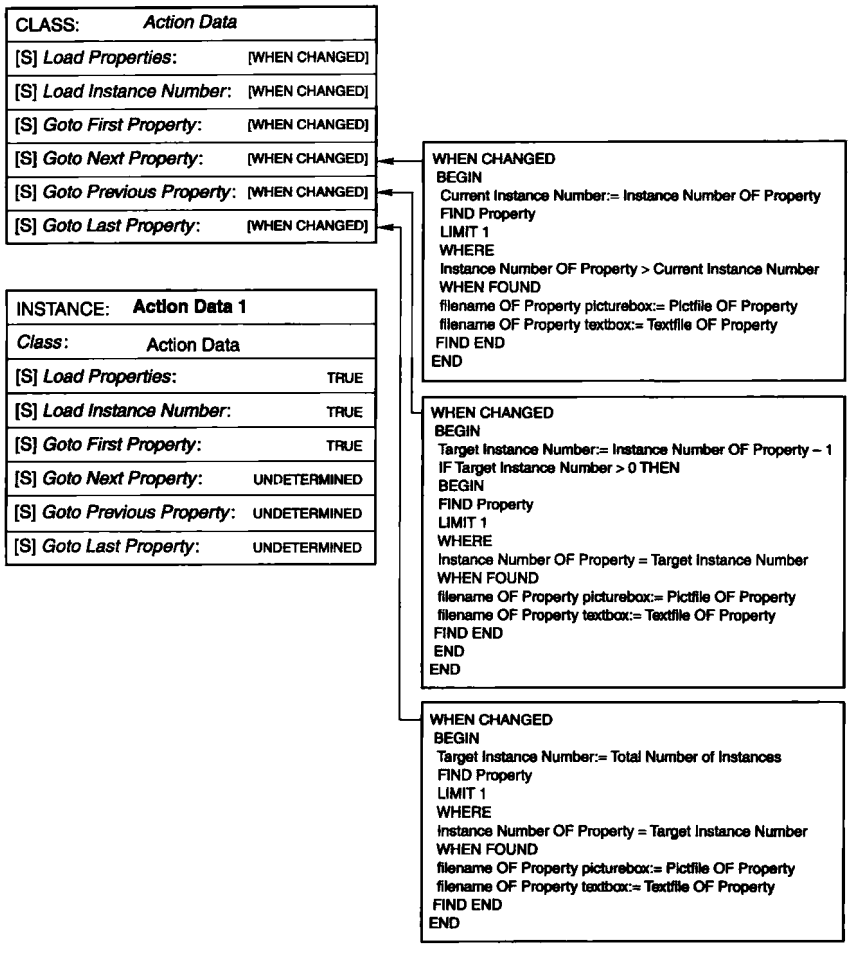


图 5.19 属性 *Goto Next Property*、*Goto Previous Property* 和 *Goto Last Property* 的 WHEN CHANGED 方法

5.7 小结

本章概要描述了基于框架的专家系统。我们介绍了框架的概念，讨论了如何用框架来表达知识，并且发现基于框架的专家系统的本质是继承。本章分析了方法、守护程序和规则的应用。
[161] 最后通过例子来说明如何开发基于框架的专家系统。

本章的主要内容有：

- 框架是带有关于某个对象或概念的典型知识的数据结构。
- 框架用于在基于框架的系统中表达知识。框架包含给定对象的知识，其中包含名称和属性（也叫做槽）的集合。*name*、*weight*、*height* 和 *age* 都是 *Person* 框架的属性，*model*、*processor*、*memory* 和 *price* 是 Computer 框架的属性。
- 属性用来保存值。属性可以包含默认值或指向其他框架、一系列规则或过程的指针，通过这个指针可以得到属性值。
- 基于框架的系统可以通过应用侧面来扩展属性值的结构。侧面用来创建属性值，控制终端用户查询，以及告诉推理引擎如何处理属性。
- 一个框架可以指一组相似的对象，也可以指某一个对象。类框架描述一组有共同

属性的对象。*Animal*、*Person*、*Car* 和 *Computer* 都是类框架。实例可以描述某一个对象。

- 基于框架的系统支持类的继承，即实例框架默认类框架的所有属性。继承的基本思想是类框架中的属性对类中所有对象都为真，但实例框架中的槽的值为实际数据，对每个实例都是唯一的。
- 框架可以从多个父类继承属性，得到多于一个父类的属性。
- 框架间的交互是通过方法和守护程序实现的。方法是和框架属性相关的过程，在需要的时候执行。大多数基于框架的专家系统使用两种方法：WHEN CHANGED 和 WHEN NEEDED。WHEN CHANGED 方法在有新的信息加入槽时使用，WHEN NEEDED 方法在解决问题需要该信息，但没有指定槽值时执行。
- 守护程序和方法类似，这两个术语作为同义词使用。但是，如果需要编写复杂的过程，方法更加适用。另一方面，守护程序一般局限于 IF-THEN 语句。
- 在基于框架的专家系统中，规则经常使用模式匹配子句。这些子句中包含用来在所有的实例框架中定位匹配条件的变量。 [162]
- 虽然框架是结合陈述性知识和过程性知识的强大工具，但是知识工程师还是很难在系统的分层结构和继承路径中作出决策。

复习题

1. 什么是框架？什么是类和实例？举例说明。
2. 为对象 Student（学生）设计类框架，确定它的属性并为该类定义一些实例。
3. 什么是侧面？给出不同侧面的例子。
4. 在将问题分解为框架、槽和侧面时的标准是什么？通过例子来证明你的答案。
5. 如何将对象关联到基于框架的系统？什么是“a-kind-of”和“a-part-of”关系？举例说明。
6. 定义基于框架系统的继承。为什么说继承是基于框架系统的本质特征？
7. 框架可以从多个父类中继承属性吗？举例说明。
8. 什么是方法？在基于框架的专家系统中最常见的方法是什么？
9. 什么是守护程序？守护程序和方法之间有什么不同？
10. 基于规则的专家系统的规则和基于框架的专家系统的规则有什么不同？
11. 开发基于框架的专家系统的主要步骤是什么？
12. 列举基于框架的专家系统的优点。开发基于框架专家系统的难点是什么？

参考文献

Blaha, M. and Rumbaugh, J. (2004). *Object-Oriented Modeling and Design with UML*, 2nd edn. Prentice Hall, Englewood Cliffs, NJ.

Brachman, R.J. and Levesque, H.J. (2004). *Knowledge Representation and Reasoning*. Elsevier, Amsterdam.

Durkin, J. (1994). *Expert Systems: Design and Development*. Prentice Hall, Englewood Cliffs, NJ.

Minsky, M. (1975). A framework for representing knowledge, *The Psychology of Computer Vision*, P. Winston, ed., McGraw-Hill, New York, pp. 211–277.

Sterling, L. and Shapiro E. (1994). *The Art of Prolog: Advanced Programming Techniques*, 2nd edn. MIT Press, Cambridge, MA.

Weisfeld, M. (2008). *The Object-Oriented Thought Process*, 3rd edn. Addison-Wesley, Upper Saddle River, NJ. [163]

参考书目

Aikens, J.S. (1984). A representation scheme using both frames and rules, *Rule-Based Expert Systems*, B.G. Buchanan and E.H. Shortliffe, eds, Addison-Wesley, Reading, MA, pp. 424–440.

Alpert, S.R., Woyak, S.W., Shrobe, H.J. and Arrowood, L.F. (1990). Guest Editors Introduction: Object-oriented programming in AI, *IEEE Expert*, 5(6), 6–7.

Booch G., Maksimchuk, R.A., Engel, M.W., Young, B.J., Conallen, J. and Houston, K.A. (2007). *Object-Oriented Analysis and Design with Applications*, 3rd edn. Addison-Wesley, Reading, MA.

Coppin, B. (2004). *Artificial Intelligence Illuminated*. Jones and Bartlett, Sudbury, MA.

Fikes, R. and Kehler, T. (1985). The role of frame-based representation in reasoning, *Communications of the ACM*, 28(9), 904–920.

Goldstein, I. and Papert, S. (1977). Artificial intelligence, language, and the study of knowledge, *Cognitive Science*, 1(1), 84–123.

Jackson, P. (1999). *Introduction to Expert Systems*, 3rd edn. Addison-Wesley, Harlow.

Keogh, J. and Giannini, M. (2004). *OOP Demystified*. McGraw-Hill, New York.

Kordon, A.K. (2010). *Applying Computational Intelligence: How to Create Value*. Springer-Verlag, Berlin.

Levesque, H.J. and Brachman, R.J. (1985). A fundamental trade-off in knowledge representation and reasoning, *Readings in Knowledge Representation*, R.J. Brachman and H.J. Levesque, eds, Morgan Kaufmann, Los Altos, CA.

Luger, G.F. (2008). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, 6th edn. Addison-Wesley, Harlow.

Poo, D., Kiong, D. and Ashok, S. (2008). *Object-Oriented Programming and Java*, 2nd edn. Springer-Verlag, London.

Rosson, M.B. and Alpert, S.R. (1990). The cognitive consequences of object-oriented design, *Human–Computer Interaction*, 5(4), 345–379.

Stefik, M.J. (1995). *Introduction to Knowledge Systems*. Morgan Kaufmann, San Francisco.

Stefik, M.J. and Bobrow, D.G. (1986). Object-oriented programming: themes and variations, *AI Magazine*, 6(4), 40–62.

Touretzky, D.S. (1986). *The Mathematics of Inheritance Systems*. Morgan Kaufmann, Los Altos, CA.

Waterman, D.A. (1986). *A Guide to Expert Systems*. Addison-Wesley, Reading, MA.

Winston, P.H. (1977). Representing knowledge in frames, Chapter 7 of *Artificial Intelligence*, Addison-Wesley, Reading, MA, pp. 181–187.

Wu, C.T. (2009). *An Introduction to Object-Oriented Programming with Java*, 5th edn. McGraw-Hill, New York.

人工神经网络

本章介绍人脑的工作机制以及如何建立和训练人工神经网络。

6.1 人脑工作机制简介

“计算机什么都没有证明，”1997 年 5 月，愤怒的国际象棋冠军加里·卡斯帕罗夫（Garry Kasparov）在纽约败给计算机时说，“如果我们是在进行一盘真正的比赛，我会把深蓝撕成碎片。”

尽管卡斯帕罗夫对他在六局比赛中的失败轻描淡写，但是，事实上卡斯帕罗夫，世界上最伟大的国际象棋大师被计算机击败，标志着智能机器探索的转折点。

这台被称为深蓝的 IBM 超级计算机，能够每秒钟分析 2 亿个位置，看上去展示出了智能。卡斯帕罗夫甚至一度控告这台机器在作弊。

在比赛中有很多很多的发现，例如计算机有时走棋就非常像人类棋手。

它深刻地理解了位置因素。这是一项非常杰出的科学成就。

从传统意义上讲，要想在国际象棋比赛中击败人类专家，计算机应该有策略而不是简单地每秒钟计算大量的“后续”步骤。国际象棋对弈程序必须能够利用经验来提高它们的表现，换句话说，机器必须有学习的能力。

什么是机器学习

通常，机器学习涉及自适应机制，这种机制使计算机通过经验、实例和类比进行学习。智能系统的学习能力能够逐步改善系统的性能。机器学习的机制是自适应系统的基础。机器学习最常见的方法是人工神经网络和遗传算法。本章主要介绍神经网络。

什么是神经网络

神经网络可以定义为基于人脑的推理模型。大脑由密集的相互连接的神经细胞或基本信息处理单元（也叫神经元）组成。人脑大约有 100 亿个神经元，之间连接形成的突触有 60 万亿个（Shepherd and Koch，1990）。通过同时使用多个神经元，大脑执行任务的速度远远超过现在最快的计算机。

尽管每个神经元的结构都很简单，但这么多神经元结合起来，处理能力还是非常强大的。一个神经元包含一个细胞体（soma）、一些树突（dendrite）和一根很长的轴突（axon）。如果树突伸向细胞体周围的网络中，轴突就向树突和其他神经元的细胞体伸展开。图 6.1 是神经网络的示意图。

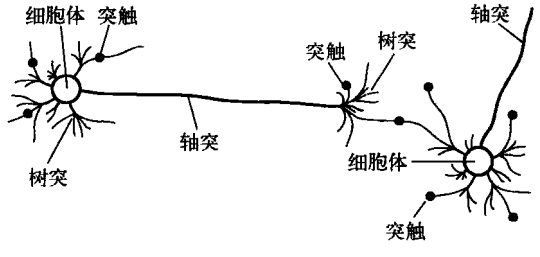


图 6.1 生物神经网络

信号在神经元之间的传递是通过复杂的电化学反应实现的。从突触中释放出的化学物质导致细胞体电压发生变化。当电压达到阈值，就会通过轴突向下传送一个电脉冲（动作电位）。脉冲传播并最终到达突触，使突触的电压增加或减少。但是，最有意思的发现是神经网络表现出一定的可塑性。为了和刺激的模式相适应，神经元在连接处的强度有长期的变化。神经元也可以与其他神经元形成新的连接。甚至整个神经元的集合都可以从一个地方搬到另一个地方。这些机制就是大脑学习的基础。

人脑可以看做是一个高度复杂、非线性、并行的信息处理系统。神经网络中信息的存储和处理在整个网络中是同时进行的，而不是在某个位置上同时进行。换句话说，在神经网络中，数据及其处理是全局的而不是局部的。

正是因为具有这种可塑性，“正确答案”的神经元之间的连接才会被强化，而“错误答案”的连接则会被弱化。因此，神经网络能够通过经验学习。

学习是生物神经网络的基础和本质特征。这种简单、自然的方式使得在计算机中模拟生物
166 神经网络的尝试成为可能。

虽然现在的人工神经网络（ANN）和人脑相似的程度就像纸飞机和超音速飞机相似的程度，但它发展得很快。ANN 有“学习”的能力，即 ANN 可以用经验来改进自身的性能。在接触了足够的实例之后，ANN 可以推广应用到新情况。它们可以识别手写体字母，识别人们说话中的单词，发现飞机上的爆炸物。此外，ANN 还可以分析人类专家难以识别的模式。例如，Chase Manhattan 银行用神经网络来检查被盗信用卡的使用信息，并发现最可疑的消费是 40 ~ 80 美元的女鞋。

人工神经网络如何模拟大脑

人工神经网络包含很多简单但高度互联的处理器，也称作神经元，这与大脑中的生物神经网络相似。神经元之间通过具有权重的链接将信号从一个神经元传递到另一个神经元。每个神经元通过链接收到几个输入信号，输出信号却顶多有一个。输出信号通过神经元的输出链接传送（和生物网络中的轴突类似）。输出链接又分出几个分支传递相同的信号（信号不能在分支间以任何方式分割）。输出分支在网络中其他神经元的输入链接处中止。图 6.2 为典型 ANN 的框架。表 6.1 给出了生物神经网络和人工神经网络间的对比（Medsker and Liebowitz, 1994）。

表 6.1 生物网络和人工神经网络间的对比

生物神经网络	人工神经网络
细胞体	神经元
树突	输入
轴突	输出
突触	权重

人工神经网络如何“学习”

神经元通过链接相连，每个链接都有数值权重。权重是 ANN 用于长期记忆的基本方式。它们表达每个神经元输入的强度或重要性。人工神经网络通过不断调整这些权重进行“学习”。

神经网络懂得如何调整权重吗

如图 6.2 所示，典型的 ANN 是层次结构的，网络中的神经元排列在这些层中。与外部环境链接的神经元形成输入层和输出层。通过调节权重使网络的输入/输出行为与外部环境一致。

每个神经元是信息处理的基本单位。在给定输入和权重时，可以计算神经元的激活水平（activation level）。

要建立人工神经网络，必须首先确定要用到多少神经元，以及如何连接神经元以形成网络。换句话说，必须首先选择网络的架构，然后决定使用什么样的学习算法。最后是训练神经网络，即初始化网络的权重，并通过一系列的实例训练改变权重的值。

下面首先介绍 ANN 的基本构建元素——神经元。

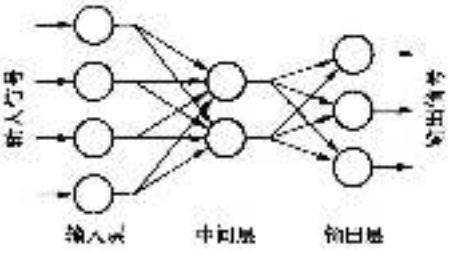


图 6.2 典型人工神经网络的结构

6.2 作为简单计算元素的神经元

神经元接收来自输入链接的一些信号，计算新的激活水平，并将其作为输出信号通过输出链接进行传送。输入信号可以是原始数据或其他神经元的输出。输出信号可以是问题的最终解决方案，也可以是其他神经元的输入信号。图 6.3 为典型的神经元。

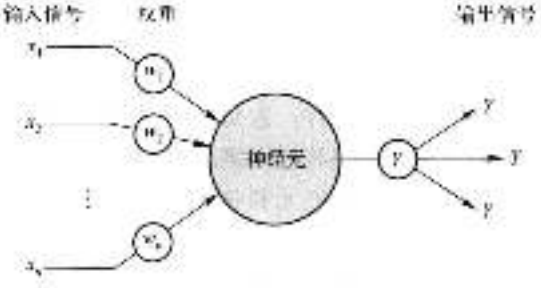


图 6.3 神经元示意图

167
168

神经元如何确定输出

1943 年，Warren McCulloch 和 Walter Pitts 提出了一种非常简单的思想，这种思想现在依旧是大多数人工神经网络的基础。

神经元计算带权重的输入信号之和，并将此结果和阈值 θ 比较。如果网络的净输入比阈值低，神经元输出 −1；如果网络净输入大于或等于阈值，则神经元激活并输出 +1（McCulloch and Pitts，1943）。

换句话说，神经元使用下面的转换或激活函数：

$$X = \sum_{i=1}^{n} x_i w_i \tag{6.1}$$

$$Y = \begin{cases} +1, & \text{若 } X \geqslant \theta \\ -1, & \text{若 } X < \theta \end{cases}$$

其中，X 是神经元的净权重输入，x_i 是输入 i 的值，w_i 是输入 i 的权重，n 是神经元输入的数目，Y 是神经元的输出。

这种激活函数称为符号函数。

因此，符号激活函数的神经元的实际输出可以表示为：

$$Y = \text{sign}\left[\sum_{i=1}^{n} x_i w_i - \theta\right] \tag{6.2}$$

符号函数是神经元使用的唯一激活函数吗

在已经测试的激活函数中，仅有几个在实际中得到了应用。4 个常用的激活函数（阶跃函数、符号函数、S 形函数和线性函数）如图 6.4 所示。

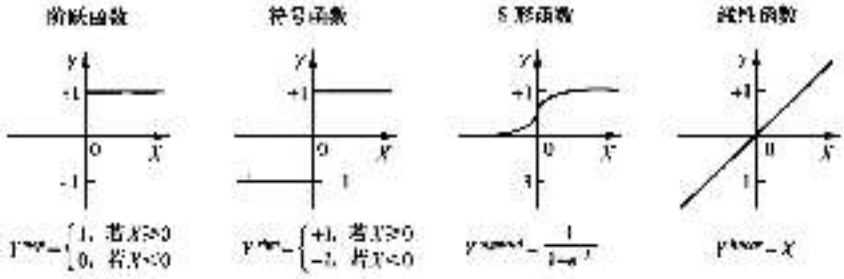


图6.4 神经元的激活函数

阶跃和符号激活函数（也称为硬限幅函数），常用于分类和模式识别的决策制定神经元。

S形函数可以将输入（变化范围在负无穷到正无穷之间的值）转换成范围在0～1之间的适当的值。使用该函数的神经元用于反向传播的网络。

线性激活函数的输出和神经元的权重输入一致。使用线性函数的神经元一般用于线性近似。

单个神经元可以学习吗

1958年，Frank Rosenblatt提出了一种训练算法，该算法给出了第一个训练简单的ANN的过程：感知器（perceptron）（Rosenblatt，1958）。感知器是神经网络最简单的形式。它由一个可调整突触权重的神经元和硬限幅器组成。图6.5所示为单层双输入感知器。

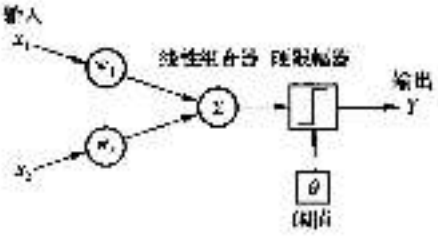


图6.5 单层双输入感知器

6.3 感知器

Rosenblatt感知器的操作是基于McCulloch和Pitts的神经元模型。这个模型由一个线性组合器后接一个硬限幅器组成。输入的权重之和被施加于硬限幅器，若其输入为正，则输出为+1；若其输入为负，则输出为-1。感知器的作用是将输入分类，也就是将输入x_1，x_2，…，x_n分为两类，即A_1和A_2。因此一个基本的感知器，用超平面将n维空间划分为两个决策区域。超平面由线性分割函数定义：

$$\sum_{i=1}^{n} x_i w_i - \theta = 0 \tag{6.3}$$

169
170

在有两个输入x_1和x_2时，决策边界为图6.6a中所示的粗直线。点1在边界线的上方，属于A_1类，点2在边界线的下方，属于A_2类。阈值θ用来改变决策边界。

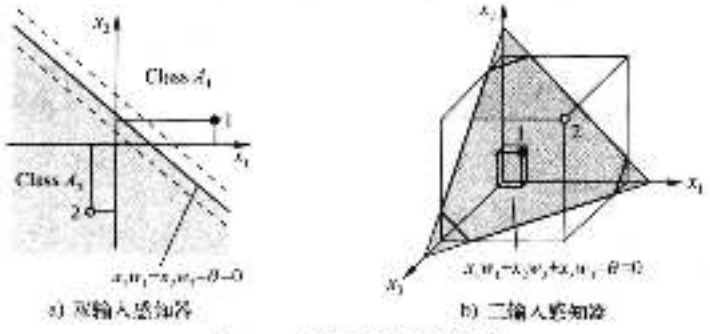


图6.6 感知器的线性分割

有三个输入的超平面还是可见的，图6.6b所示为三输入感知器的三个维度。其中分割平面可以定义为：

$$x_1w_1 + x_2w_2 + x_3w_3 - \theta = 0$$

感知器如何学习分类任务

通过细微地调节权重来减少感知器的期望输出和实际输出之间的差别，可以完成这一任务。初始权重可以随意赋值，通常在范围 [-0.5, 0.5] 内，然后通过训练实例进行调整。对于感知器，权重调整的过程非常简单。如果在迭代 p 中，实际输出为 $Y(p)$，期望输出为 $Y_d(p)$，那么误差为：

$$e(p) = Y_d(p) - Y(p) \quad \text{其中 } p = 1,2,3,\cdots \tag{6.4}$$

这里迭代 p 表示提供给感知器的第 p 个训练实例。

如果误差 $e(p)$ 为正，就需要增加感知器的输出 $Y(p)$；如果 $e(p)$ 为负，就减少感知器的输出 $Y(p)$。考虑到每个感知器对总输入 $X(p)$ 的贡献为 $x_i(p) \times w_i(p)$，可以得知，如果输入值 $x_i(p)$ 为正，那么增加其权重 $w_i(p)$ 可以增加感知器的输出 $Y(p)$ 的值；如果输入值 $x_i(p)$ 为负，那么增加其权重 $w_i(p)$ 可以减少输出 $Y(p)$ 的值。因此，可以建立下面的感知器学习规则：

$$w_i(p+1) = w_i(p) + \alpha \times x_i(p) \times e(p) \tag{6.5}$$

其中，α 是学习速率，是一个小于 1 的正常数。

感知器学习规则首先由 Rosenblatt 在 1960 年提出（Rosenblatt, 1960）。使用这个规则可以得出用于分类任务的感知器训练算法。 171

步骤 1：初始化。

设置权重 w_1，w_2，…，w_n 和阈值 θ 的初值，取值范围为 [-0.5, 0.5]。

步骤 2：激活。

通过用输入 $x_1(p)$，$x_2(p)$，…，$x_n(p)$ 以及期望输出 $Y_d(p)$ 来激活感知器。在迭代 $p = 1$ 上计算实际输出为：

$$Y(p) = step\left[\sum_{i=1}^{n} x_i(p)w_i(p) - \theta\right] \tag{6.6}$$

其中，n 为感知器输入的数量，$step$ 为阶跃激活函数。

步骤 3：权重训练。

修改感知器的权重：

$$w_i(p+1) = w_i(p) + \Delta w_i(p) \tag{6.7}$$

其中，$\Delta w_i(p)$ 为迭代 p 上的权重校正。通过 delta 规则计算权重校正：

$$\Delta w_i(p) = \alpha \times x_i(p) \times e(p) \tag{6.8}$$

步骤 4：迭代。

迭代 p 加 1，回到步骤 2，重复以上过程直至收敛。

可以训练感知器执行类似于 AND、OR 或 Exclusive - OR 的逻辑操作吗

表 6.2 为 AND（与）、OR（或）和 Exclusive - OR（异或）的真值表。表中列出了两个变量 x_1 和 x_2 所有可能的组合和操作的结果。感知器必须要经过训练才能分类不同的输入模式。

表 6.2 基本逻辑操作的真值表

输入变量		与	或	异或
x_1	x_2	$x_1 \cap x_2$	$x_1 \cup x_2$	$x_1 \oplus x_2$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

首先考虑 AND 操作。初始化步骤完成后，感知器由表示一个周期的 4 个输入模式的序列激活。每次激活后修改感知器的权重。一直重复这个过程，直到所有的权重收敛到统一的一组取值 172
为止。结果如表 6.3 所示。

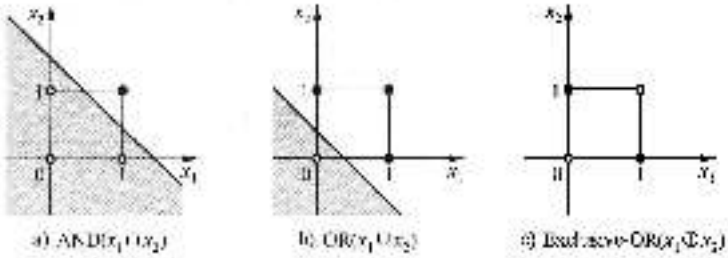
表 6.3 感知器学习的例子：逻辑操作 AND

周期	输入		期望输出 Y_d	初始权重		实际输出 Y	误差 e	最终权重	
	x_1	x_2		w_1	w_2			w_1	w_2
1	0	0	0	0.3	−0.1	0	0	0.3	−0.1
	0	1	0	0.3	−0.1	0	0	0.3	−0.1
	1	0	0	0.3	−0.1	1	−1	0.2	−0.1
	1	1	1	0.2	−0.1	0	1	0.3	0.0
2	0	0	0	0.3	0.0	0	0	0.3	0.0
	0	1	0	0.3	0.0	0	0	0.3	0.0
	1	0	0	0.3	0.0	1	−1	0.2	0.0
	1	1	1	0.2	0.0	1	0	0.2	0.0
3	0	0	0	0.2	0.0	0	0	0.2	0.0
	0	1	0	0.2	0.0	0	0	0.2	0.0
	1	0	0	0.2	0.0	1	1	0.1	0.0
	1	1	1	0.1	0.0	0	1	0.2	0.1
4	0	0	0	0.2	0.1	0	0	0.2	0.1
	0	1	0	0.2	0.1	0	0	0.2	0.1
	1	0	0	0.2	0.1	1	−1	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1
5	0	0	0	0.1	0.1	0	0	0.1	0.1
	0	1	0	0.1	0.1	0	0	0.1	0.1
	1	0	0	0.1	0.1	0	0	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1

注：阈值：$\theta=0.2$；学习速率：$\alpha=0.1$。

类似地，感知器可以学习 OR 操作。但是单层的感知器不能通过训练来执行 Exclusive－OR 操作。

可以通过一个简单的几何例子理解其中的原因。图 6.7 为有两个输入的 AND、OR、Exclusive－OR 二维图。当函数输出为 1 时输入空间的点用黑色表示，当函数输出为 0 时输入空间的点用白色表示。



173 图 6.7 基本逻辑操作的二维图

在图 6.7a 和图 6.7b 中，可以画一条线，让所有的黑点在一边，白点在另一边，但图 6.7c 中黑点和白点就不能用一条线分割。感知器仅仅能够表达这种能用一条线将所有黑点和所有白

点分割开的函数。这种函数也称作线性分割函数。因此，感知器可以学习的操作有 AND 和 OR，但不能学习 Exclusive - OR 操作。

为什么感知器仅仅能学习线性分割函数

可以从式（6.1）中推导出感知器为什么只能学习线性分割函数。仅当总的权重输入 X 大于或等于阈值 θ 时，感知器的输出 Y 才为 1。这就说明整个输入空间通过 $X=\theta$ 所定义的边界分成了两部分。例如，操作 AND 的分割线由下面的公式定义：

$$x_1 w_1 + x_2 w_2 = \theta$$

如果按表 6.3 给权重 w_1、w_2 和阈值 θ 赋值，就得到了一条可能的分割线：

$$0.1x_1 + 0.1x_2 = 0.2$$

或

$$x_1 + x_2 = 2$$

因此，输出为 0 的区域在边界线的下方：

$$x_1 + x_2 - 2 < 0$$

输出为 1 的区域在边界线的上方：

$$x_1 + x_2 - 2 \geqslant 0$$

感知器只能学习线性分割函数，这确实是一个坏消息，因为实际上没有多少这样的函数。

用 S 形或线性元素取代硬限幅器会更好吗

不管感知器使用的激活函数是什么，单层感知器用相同的方法做决策（Shynk，1990；Shynk and Bershad，1992）。这就意味着不管是用硬限幅激活函数还是软限幅激活函数，单层感知器仅能分类线性分割模式。

感知器的计算限制在 Minsky 和 Papert 所著的 *Perceptrons* 中有相关的数学分析（Minsky and Papert，1969）。这本书中证明了 Ronsenblatt 感知器不能根据在局部学习到的实例进行全局推广。同时，Minsky 和 Papert 总结出单层感知器的限制对于多层神经网络也适用。这样的结论当然不鼓 174
励对人工神经网络更进一步的研究。

怎样处理不能线性分割的问题

要处理这样的问题就需要多层神经网络。实际上，历史已经证明了 Rosenblatt 感知器可以通过改进神经网络的形式来克服，例如，用反向传播算法训练的多层感知器。

6.4 多层神经网络

多层感知器是有一个或多个隐含层的前馈神经网络。通常，网络包含由源神经元组成的一个输入层，由计算神经元组成的至少一个中间层或隐含层，以及由计算神经元组成的一个输出层。输入信号一层一层地前向传递。有两个隐含层的多层感知器如图 6.8 所示。

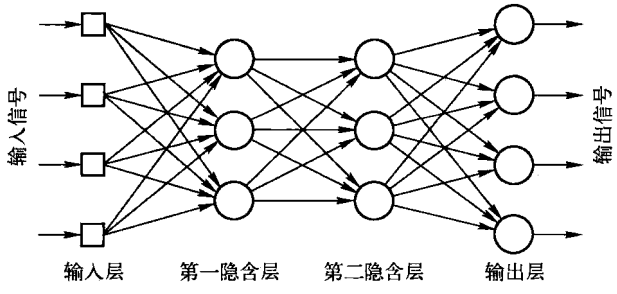


图 6.8 有两个隐含层的多层感知器

为什么需要隐含层

多层神经网络的每一层都有其特定的功能。输入层接收来自外部世界的输入信号，重新将信号发送给隐含层的所有神经元。实际上，输入层很少会包含计算神经元，因此不处理输入模式。输出层从隐含层接收输出信号，或者刺激模式，并为整个网络建立输出模式。

隐含层的神经元发现特征；神经元的权重表示了隐含在输入模式中的特征。输出层根据这些特征确定输出模式。

利用一个隐含层，我们可以表示输入信号的任何连续函数；利用两个隐含层甚至可以表示
175 不连续的函数。

为什么多层网络中的中间层叫做隐含层？这个层隐藏了什么

隐含层“隐含”了其期望的输出值。隐含层的神经元不能通过网络的输入/输出行为来分析。没有明显的方式可以了解隐含层的期望输出值。换句话说，隐含层的期望输出完全由自己决定。

神经网络可以包含2个以上的隐含层吗

商用 ANN 一般有三层或四层，包含一到两个隐含层。每层有 10 ~ 1000 个神经元。实验神经网络可能有 5 层甚至 6 层，包含 3 或 4 个隐含层，有数百万个神经元，但大多数实际应用仅有 3 层，因为每增加一层，计算量将呈指数级上升。

多层神经网络如何学习

有上百万种学习算法可供选择，但最常用的是反向传播方法。这种方法在 1969 年被首次提出（Bryson and Ho，1969），但是由于其对计算的要求过分苛刻而被人们忽略。20 世纪 80 年代中期，这种算法才被重新发现。

多层网络的学习过程和感知器一样。要给网络提供输入模式的训练集。网络计算其输出模式，如果有错，也就是说，实际输出和期望输出不一致，就调节权重来减小误差。

在感知器中，每个输入仅有一个权重和一个输出。但在多层网络中，有多个权重，每个权重对一个以上的输出都有贡献。

是否可以估计误差，并在有作用的权重中间分配

在反向传播神经网络中，学习算法有两个阶段。首先将训练输入模式提供给网络的输入层。然后输入模式在网络中一层一层地传递，直到输出层产生输出模式为止。如果输出模式和网络预期的输出模式不同，则计算误差，然后从网络的输出层反向传播回输入层。在传递误差时，调整权重的值。

和其他神经网络一样，反向传播神经网络是由神经元的连接（网络架构）、神经元使用的激活函数和指定用于调整权重的学习算法（或学习规则）组成。

通常，反向传播网络是多层网络，有 3 层或 4 层，层与层之间充分连接，也就是说，每一层
176 的每个神经元和相邻的前一层中其他神经元都有连接。

神经元确定输出的方式和 Rosenblatt 感知器类似。首先计算净权重输入：

$$X = \sum_{i=1}^{n} x_i w_i - \theta$$

其中，n 是输入的个数，θ 是神经元的阈值。

接下来，输入值通过激活函数传递。和感知器不同的是，反向传播网络中的神经元使用 S 形激活函数。

$$Y^{sigmoid} = \frac{1}{1 + e^{-X}} \tag{6.9}$$

该函数的导数容易计算。它保证神经元的输出在 0 到 1 之间。

反向传播网络中使用的学习规则是什么

要推导反向传播的学习规则，我们考虑图6.9中的3层网络。下标 i、j 和 k 分别表示输入层、隐含层和输出层中的神经元。

输入信号 x_1，x_2，…，x_n 从网络的左侧传递到右侧，而误差信号 e_1，e_2，…，e_l 从右到左传递。符号 w_{ij} 指的是输入层的神经元 i 和隐含层的神经元 j 间连接的权重，符号 w_{jk} 指的是隐含层的神经元 j 和输出层的神经元 k 间连接的权重。

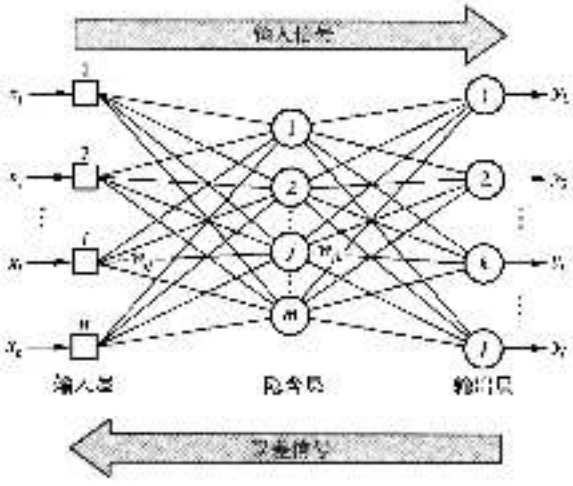


图6.9 3层反向传播神经网络

为了传送误差信号，从输出层开始向后传到隐含层。神经元 k 的第 p 次迭代的误差信号定义为：

$$e_k(p) = y_{d,k}(p) - y_k(p) \qquad (6.10)$$

其中，$y_{d,k}(p)$ 是神经元 k 的第 p 次迭代的期望输出。

位于输出层的神经元 k 具有自己的预期值。因此，可以用简单方法来修改权重 w_{jk} 的值。实际上，修改输出层权重的规则和公式（6.7）的感知器学习规则类似：

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p) \qquad (6.11)$$

其中，$\Delta w_{jk}(p)$ 是权重校正。

确定了感知器的权重校正后，使用输入信号 x_j。但是在多层网络中，输出层神经元的输入和输入层神经元的输入是不同的。

不能使用输入信号 x_j，应该使用什么来替代

我们使用隐含层神经元 j 的输出 y_j 而不是输入 x_j。多层网络权重校正的计算方法（Fu，1994）为：

$$\Delta w_{jk}(p) = \alpha \times y_j(p) \times \delta_k(p) \qquad (6.12)$$

其中，$\delta_k(p)$ 是输出层第 p 次迭代时神经元 k 的误差梯度。

什么是误差梯度

误差梯度的定义为激活函数的导数和神经元输出误差的乘积。

因此，对于输出层神经元 k，有

$$\delta_k(p) = \frac{\partial y_k(p)}{\partial X_k(p)} \times e_k(p) \qquad (6.13)$$

其中，$y_k(p)$ 是神经元 k 在第 p 次迭代的输出，$X_k(p)$ 为同次迭代中神经元 k 的净权重输入。

对于S形激活函数，式（6.13）可表示为：

$$\delta_k(p) = \frac{\partial\left[\dfrac{1}{1+\exp[-X_k(p)]}\right]}{\partial X_k(p)} \times e_k(p) = \frac{\exp[-X_k(p)]}{[1+\exp[-X_k(p)]]^2} \times e_k(p)$$

因此，可以得到：

$$\delta_k(p) = y_k(p) \times [1 - y_k(p)] \times e_k(p) \qquad (6.14)$$

其中，

$$y_k(p) = \frac{1}{1+\exp[-X_k(p)]}$$

如何确定隐含层中神经元的权重校正

要计算隐含层的权重校正，可以使用和输出层相同的公式：

$$\Delta w_{ij}(p) = \alpha \times x_i(p) \times \delta_j(p) \tag{6.15}$$

其中，$\delta_j(p)$为隐含层神经元j的误差梯度。

$$\delta_j(p) = y_j(p) \times [1 - y_j(p)] \times \sum_{k=1}^{l} \delta_k(p) w_{jk}(p)$$

其中，l是输出层神经元的个数。

$$y_j(p) = \frac{1}{1 + e^{-X_j(p)}}$$

$$X_j(p) = \sum_{i=1}^{n} x_i(p) \times w_{ij}(p) - \theta_j$$

n为输入层神经元的个数。

现在可以导出反向传播的训练算法。

步骤1：初始化。

用很小范围内均匀分布的随机数设置网络的权重和阈值（Haykin，2008）：

$$\left(-\frac{2.4}{F_i}, +\frac{2.4}{F_i} \right)$$

其中，F_i是网络中神经元i的输入总数。权重的初值要逐个神经元来设置。

步骤2：激活。

通过应用输入$x_1(p)$，$x_2(p)$，…，$x_n(p)$和期望的输出$y_{d,1}(p)$，$y_{d,2}(p)$，…，$y_{d,n}(p)$来激活反向传播神经网络。

（a）计算隐含层神经元的实际输出：

$$y_j(p) = sigmoid\left[\sum_{i=1}^{n} x_i(p) \times w_{ij}(p) - \theta_j \right]$$

179 其中，n是隐含层神经元j的输入个数，*sigmoid*为S形激活函数。

（b）计算输出层神经元的实际输出：

$$y_k(p) = sigmoid\left[\sum_{j=1}^{m} x_{jk}(p) \times w_{jk}(p) - \theta_k \right]$$

其中，m为输出层神经元k的输入个数。

步骤3：权重训练。

修改反向传播网络中的权重（反向传播网络反向传播与输出神经元相关的误差）。

（a）计算输出层神经元的误差梯度：

$$\delta_k(p) = y_k(p) \times [1 - y_k(p)] \times e_k(p)$$

其中：

$$e_k(p) = y_{d,k}(p) - y_k(p)$$

计算权重的校正：

$$\Delta w_{jk}(p) = \alpha \times y_j(p) \times \delta_k(p)$$

更新输出神经元的权重：

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p)$$

（b）计算隐含层神经元的误差梯度：

$$\delta_j(p) = y_j(p) \times [1 - y_j(p)] \times \sum_{k=1}^{l} \delta_k(p) \times w_{jk}(p)$$

计算权重的校正：

$$\Delta w_{ij}(p) = \alpha \times x_i(p) \times \delta_j(p)$$

更新隐含层神经元的权重：

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$$

步骤4：迭代。

迭代 p 加 1，回到步骤2，重复上述过程直到满足误差要求为止。

举个例子，考虑如图 6.10 所示的三层反向传播网络，假设网络需要执行的逻辑操作是 Exclusive - OR。回忆一下，单层的感知器不能进行这样的操作。这里使用三层的网络。

输入层中的神经元 1 和 2 分别接收输入 x_1 和 x_2，然后并不做任何处理，而是在隐含层中重新分配给神经元：$x_{13} = x_{14} = x_1$ 和 $x_{23} = x_{24} = x_2$。

隐含层或输出层的某个神经元的阈值可以用其权重 θ 来表示，θ 与一个等于 -1 的固定输入相连。

权重或阈值的初值可随意设置如下：

$w_{13} = 0.5$，$w_{14} = 0.9$，$w_{23} = 0.4$，$w_{24} = 1.0$，$w_{35} = -1.2$，$w_{45} = 1.1$，$\theta_3 = 0.8$，$\theta_4 = -0.1$ 以及 $\theta_5 = 0.3$

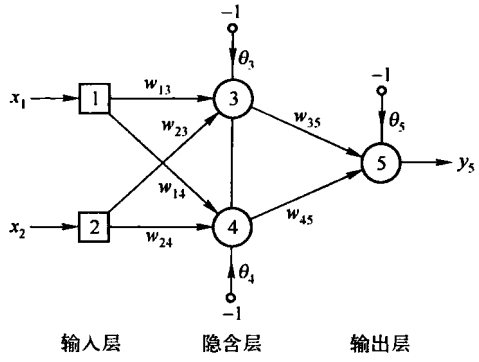


图 6.10 解决 Exclusive - OR 操作的三层网络

考虑训练集的输入 x_1 和 x_2 都为 1，期望输出 $y_{d,5}$ 为 0。隐含层中的神经元 3 和 4 的实际输出为：

$$y_3 = sigmoid(x_1 w_{13} + x_2 w_{23} - \theta_3) = 1/[1 + e^{-(1\times0.5 + 1\times0.4 - 1\times0.8)}] = 0.5250$$

$$y_4 = sigmoid(x_1 w_{14} + x_2 w_{24} - \theta_4) = 1/[1 + e^{-(1\times0.9 + 1\times1.0 + 1\times0.1)}] = 0.8808$$

现在可以确定输出层神经元 5 的实际输出为：

$$y_5 = sigmoid(y_3 w_{35} + y_4 w_{45} - \theta_5) = 1/[1 + e^{-(-0.5250\times1.2 + 0.8808\times1.1 - 1\times0.3)}] = 0.5097$$

因此，得到误差：

$$e = y_{d,5} - y_5 = 0 - 0.5097 = -0.5097$$

下一步是权重训练。要更新网络中的权重和阈值，需要从输出层反向传播误差 e 到输入层。

首先，计算输出层神经元 5 的误差梯度：

$$\delta_5 = y_5[1 - y_5]e = 0.5097 \times (1 - 0.5097) \times (-0.5097) = -0.1274$$

接下来，假设学习速率参数 α 为 0.1，确定权重的校正值：

$$\Delta w_{35} = \alpha \times y_3 \times \delta_5 = 0.1 \times 0.5250 \times (-0.1274) = -0.0067$$

$$\Delta w_{45} = \alpha \times y_4 \times \delta_5 = 0.1 \times 0.8808 \times (-0.1274) = -0.0112$$

$$\Delta \theta_5 = \alpha \times (-1) \times \delta_5 = 0.1 \times (-1) \times (-0.1274) = 0.0127$$

下面计算隐含层中神经元 3 和 4 的误差梯度：

$$\delta_3 = y_3(1 - y_3) \times \delta_5 \times w_{35} = 0.5250 \times (1 - 0.5250) \times (-0.1274) \times (-1.2) = 0.0381$$

$$\delta_4 = y_4(1 - y_4) \times \delta_5 \times w_{45} = 0.8808 \times (1 - 0.8808) \times (-0.1274) \times 1.1 = -0.0147$$

确定权重的校正值：

$$\Delta w_{13} = \alpha \times x_1 \times \delta_3 = 0.1 \times 1 \times 0.0381 = 0.0038$$

$$\Delta w_{23} = \alpha \times x_2 \times \delta_3 = 0.1 \times 1 \times 0.0381 = 0.0038$$

$$\Delta \theta_3 = \alpha \times (-1) \times \delta_3 = 0.1 \times (-1) \times 0.0381 = -0.0038$$

$$\Delta w_{14} = \alpha \times x_1 \times \delta_4 = 0.1 \times 1 \times (-0.0147) = -0.0015$$

$$\Delta w_{24} = \alpha \times x_2 \times \delta_4 = 0.1 \times 1 \times (-0.0147) = -0.0015$$

$$\Delta \theta_4 = \alpha \times (-1) \times \delta_4 = 0.1 \times (-1) \times (-0.0147) = 0.0015$$

最后，更新网络中所有的权重和阈值：

$$w_{13} = w_{13} + \Delta w_{13} = 0.5 + 0.0038 = 0.5038$$
$$w_{14} = w_{14} + \Delta w_{14} = 0.9 - 0.0015 = 0.8985$$
$$w_{23} = w_{23} + \Delta w_{23} = 0.4 + 0.0038 = 0.4038$$
$$w_{24} = w_{24} + \Delta w_{24} = 1.0 - 0.0015 = 0.9985$$
$$w_{35} = w_{35} + \Delta w_{35} = -1.2 - 0.0067 = -1.2067$$
$$w_{45} = w_{45} + \Delta w_{45} = 1.1 - 0.0112 = 1.0888$$
$$\theta_3 = \theta_3 + \Delta \theta_3 = 0.8 - 0.0038 = 0.7962$$
$$\theta_4 = \theta_4 + \Delta \theta_4 = -0.1 + 0.0015 = -0.0985$$
$$\theta_5 = \theta_5 + \Delta \theta_5 = 0.3 + 0.0127 = 0.3127$$

重复训练过程，直至误差的平方和小于0.001为止。

为什么要取误差的平方和

*误差的平方和*是衡量网络性能的很有用的指标。反向传播的训练算法试图把这个标准最小
182 化。如果误差传递通过全部训练集后，误差的平方和（或周期）达到足够小，就认为网络是收敛的。在本例中，误差的平方和足够小是指其值小于0.001。图6.11为学习曲线，纵轴为误差的平方和，横轴为训练使用的周期数。学习曲线可以显示学习的速度。

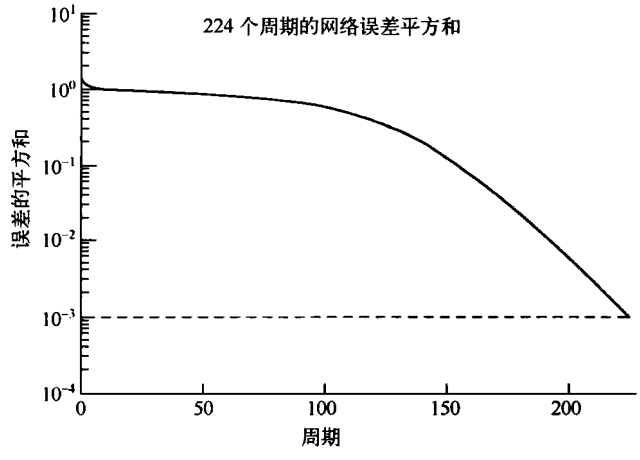


图6.11 Exclusive－OR操作的学习曲线

训练网络执行Exclusive－OR操作使用了224个周期或896次迭代。下面是满足误差标准的权重和阈值的最终取值：

$$w_{13} = 4.7621, w_{14} = 6.3917, w_{23} = 4.7618, w_{24} = 6.3917, w_{35} = -10.3788,$$
$$w_{45} = 9.7691, \theta_3 = 7.3061, \theta_4 = 2.8441 \text{ 以及} \theta_5 = 4.5589$$

网络解决了这个问题！现在可以测试网络，把所有的训练集放到输入层并计算网络输出。结果如表6.4所示。

表6.4 三层网络学习的最终结果：逻辑操作Exclusive－OR

输入		期望输出 y_d	实际输出 y_5	误差 e	误差的平方和
x_1	x_2				
1	1	0	0.0155	−0.0155	0.0010
0	1	1	0.9849	0.0151	
1	0	1	0.9849	0.0151	
183 0	0	0	0.0175	−0.0175	

初始化时权重和阈值都是任意设置的，这是否意味着同一个网络会找到不同的解决方案

不同的初始条件下网络将获得不同的权重和阈值，但是通过不同次数的迭代，总能找到解决方法。例如，如果重新训练网络，可以得到如下的解决方案：

$$w_{13}=-6.3041, w_{14}=-5.7896, w_{23}=6.2288, w_{24}=6.0088, w_{35}=9.6657,$$

$$w_{45}=-9.4242, \theta_3=3.3858, \theta_4=-2.8976 \text{ 以及 } \theta_5=-4.4859$$

现在能否画出多层网络 Exclusive – OR 操作的决策边界

要画出应用 S 形激活函数的神经元的决策边界是很困难的。但是可以通过使用符号函数的 McCulloch 和 Pitts 模型表示隐含层和输出层的每个神经元。也可以训练如图 6.12 所示的网络来执行 Exclusive – OR 操作（Touretzky and Pomerleau，1989；Haykin，2008）。

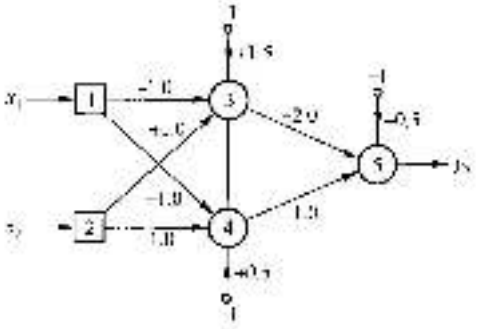
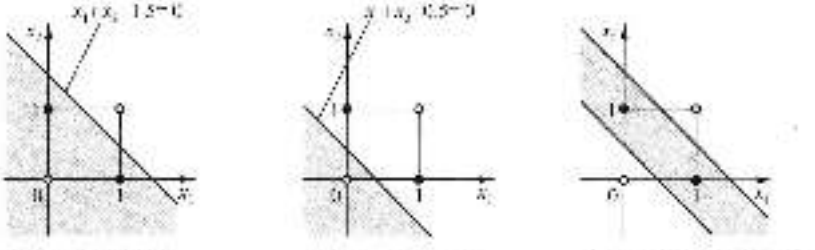


图 6.12 解决 Exclusive – OR 操作的 McCulloch – Pitts 模型表示的网络

隐含层中神经元 3 和 4 构建的决策边界的位置分别如图 6.13a 和图 6.13b 所示。输出层中的神经元 5 将两个隐含层神经元的决策边界进行线性合并，如图 6.13c 所示。图 6.12 所示的网络确实将黑点和白点分割开了，因此解决了 Exclusive – OR 问题。

反向传播的学习方法是机器学习的好方法吗

虽然反向传播的学习方法得到如此广泛的应用，但是它并不能避免所有问题。例如，反向传播学习算法似乎不能在生物领域发挥作用（Stork，1989）。生物神经元不会反向调整它们之间的连接和突触的强度，因此不能将反向传播学习看做模拟人脑学习的过程。



a) 图 6.12 中隐含层神经元 3 的决策边界

b) 图 6.12 中隐含层神经元 4 的决策边界

c) 完整的二层神经网络的决策边界

图 6.13 多层网络中 Exclusive-OR 操作的决策边界

另一个显而易见的问题是计算量巨大，因而训练速度很慢。事实上，纯粹的反向传播算法在实际中很少应用。

有一些方法可以改善反向传播算法的效率（Jacobs，1988；Stubbs，1990；Caudill，1991），下面将讨论其中的几个方法。

6.5 多层神经网络的加速学习

如果用双曲线正切来表示 S 形激活函数，通常多层神经网络的学习速率会加快。

$$Y^{tanh}=\frac{2a}{1+e^{-bX}}-a \tag{6.16}$$

其中 a 和 b 都是常数。a 和 b 的合适取值为 $a=1.716$ 和 $b=0.667$（Guyon，1991）。

我们也可以通过在公式（6.12）的 delta 规则中加入动量常数来加速训练（Rumelhart et al.，1986）。

$$\Delta w_{jk}(p) = \beta \times \Delta w_{jk}(p-1) + \alpha \times y_j(p) \times \delta_k(p) \tag{6.17}$$

其中 β 是正数（$0 \leqslant \beta < 1$），称作动量常数。通常动量常数设为 0.95。

公式（6.17）也称作通用 delta 规则，在特殊情况下，当 $\beta = 0$ 时，就得到了公式（6.12）的 delta 规则。

为什么需要动量常数

184 ~ 185 根据 Watrous（1987）和 Jacobs（1988）所做的研究，在反向传播算法中包含动量常数会对训练产生稳定的效果。换句话说，包含动量常数使得在垂直向下的方向上有了加速下降的趋势，当学习曲面出现峰和谷时，才减缓这种趋势。

图 6.14 为 Exclusive－OR 操作带有动量常数的学习。和纯反向传播算法相比，周期数从 224 减少到了 126。

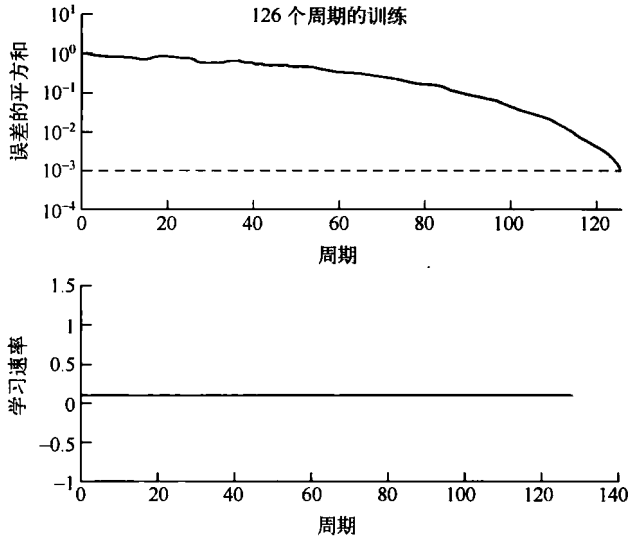


图 6.14 带有动量常数的学习

在 delta 规则和通用 delta 规则中，我们使用一个较小的常数来作为学习速率参数 α，是否可以通过增加这个值来提高训练速度

加快反向传播学习收敛速度的最有效的方法是在训练过程中调节学习速率参数。若学习速率参数 α 小，则在一次次的迭代过程中对网络权重的改变也很小，从而生成平滑的学习曲线。另一方面，如果通过增加学习速率参数 α 来加快训练过程，则权重的改变过大可能引起不稳定，网络也因此变得振荡。

为了加速收敛且避免出现不稳定的危险，可以应用两个启发式方法（Jacobs，1988）。

- **启发式方法 1**：如果在临近的几个周期中，误差平方和变化的代数符号相同，则应增加学习速率参数 α 的值。
- **启发式方法 2**：如果在临近的几个周期中，误差平方和变化的代数符号不同，则应减少学习速率参数 α 的值。

186

为了配合学习速率的改变，需要对反向传播算法做一些改变。首先，需要按初始学习速率参数计算网络的输出和误差。如果本周期中误差的平方和大于前一次值的预先指定的倍数（通常为 1.04），这时学习速率参数要减少（通常乘以 0.7），并且计算新的权重和阈值。如果误差小

于前一次，则增加学习速率（通常乘以1.05）。

图6.15为使用自适应学习速率的反向传播训练的示例。这个图说明使用自适应速度确实减少了迭代次数。

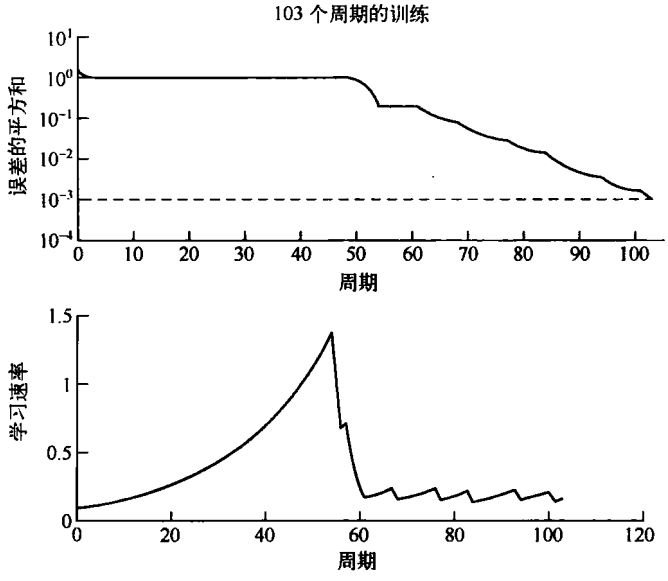


图6.15 使用自适应学习速率学习

自适应学习速率可以和动量常数一起使用，图6.16显示了同时使用这两种技术的优势。

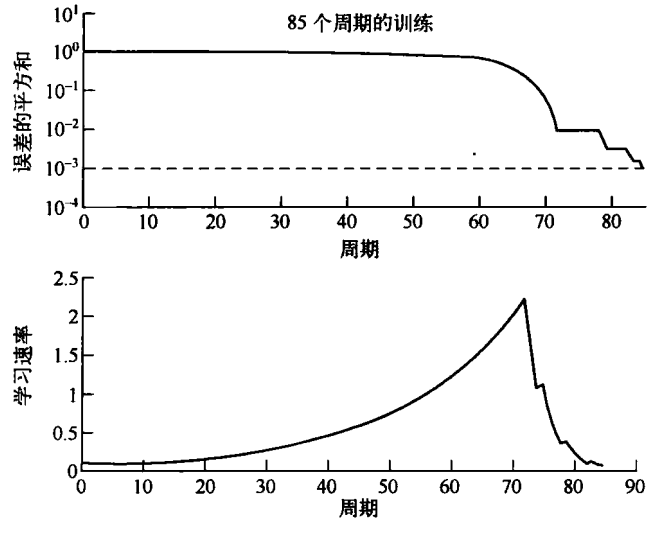


图6.16 使用动量常数和自适应学习速率

同时使用动量常数和自适应学习速率显著提高了多层反向传播神经网络的性能，并使网络变得振荡的机会减到最小。

神经网络是用来模拟大脑的。但是，大脑的记忆通过联想才能运转。例如，我们可以在100～200ms内从不熟悉的环境中找到熟悉的面孔。在听到音乐的一些片段时，我们也能够回忆起完整的感受，包括声音和场景。大脑可以将片段组合在一起。

神经网络可以模拟人类记忆的联想功能吗

用反向传播算法训练的多层神经网络可以用于解决模式识别问题。但是，我们在前面提到，这样的网络本身并不具备智能。为了模拟人类记忆的联想功能，需要一个不同类型的网络：循环神经网络（recurrent neural network）。

6.6 Hopfield 网络

循环神经网络带有从输出到输入的反馈回路。这种回路的出现对于网络的学习性能具有深远的影响。

循环网络如何学习

使用新的输入后，网络计算输出，并反馈，以此调节输入。之后重新计算输出，重复这个过程，直到输出变成常数为止。

输出一定能变成常数吗

连续的迭代不一定能使输出的改变越来越小，相反，可能导致混乱。在这种情况下，网络的输出永远不会变成常数，可以说网络是不稳定的。

187
~
188

20 世纪六七十年代，很多研究者对循环神经网络的稳定性产生了兴趣，但是没有人能够预测什么样的网络是稳定的，一些研究人员对找到通用的解决方案表示悲观。当 John Hopfield 提出在动态稳定网络中存储信息的物理原则（Hopfield，1982）时，这个问题才在 1982 年得到解决。

图 6.17 所示为包含 n 个神经元的单层 Hopfield 网络。每个神经元的输出都反馈到其他所有神经元的输入上（在 Hopfield 网络中没有自反馈）。

Hopfield 网络通常用带有符号激活函数的 McCulloch 和 Pitts 神经元作为其计算元素。

这种函数如何工作

它和图 6.4 所示的符号函数的工作方式一样。如果神经元的权重输入小于 0，则输出为 -1；如果权重输入大于 0，则输出为 +1。然而，如果神经元权重输入等于 0，则输出保持不变，换句话说，神经元保持以前的状态，无论是 -1 还是 +1。

$$Y^{sign}=\begin{cases}+1, 若\ X>0\\ -1, 若\ X<0\\ Y, \quad 若\ X=0\end{cases} \tag{6.18}$$

符号激活函数可以用饱和线性函数来代替，它在 [-1，1] 区间可作为纯线性函数，在 [-1，1] 区间之外可作为符号函数。饱和线性函数如图 6.18 所示。

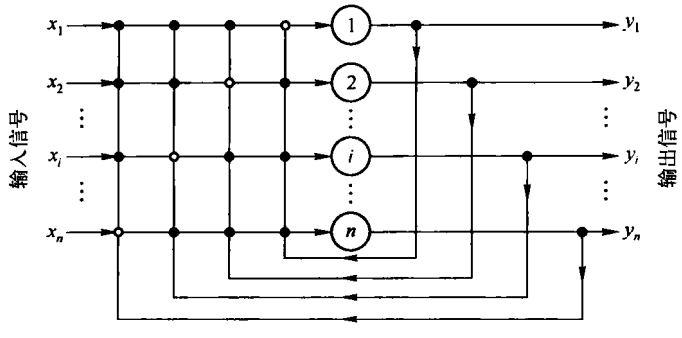
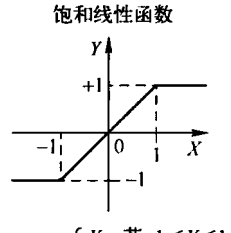


图 6.17 单层 n 个神经元 Hopfield 神经网络



$$Y^{satlin}=\begin{cases}X, 若 -1<X<1\\ +1, 若 X\geqslant 1\\ -1, 若 X\leqslant -1\end{cases}$$

图 6.18 饱和线性激活函数

网络的当前状态由当前所有神经元的输出 y_1，y_2，…，y_n 确定。因此，对于单层 n 个神经元的网络，状态可由下面的状态向量定义：

$$\boldsymbol{Y}=\begin{bmatrix}Y_1\\Y_2\\\cdots\\Y_n\end{bmatrix} \tag{6.19}$$

在 Hopfield 网络中，神经元之间的突触权重通常以下面的矩阵形式表示：

$$\boldsymbol{W}=\sum_{s=1}^{M}\boldsymbol{Y}_s\boldsymbol{Y}_s^{\mathrm{T}}-M\boldsymbol{I} \tag{6.20}$$

其中，M 是网络记忆的状态数目，$\boldsymbol{Y}_s$ 是 n 维的二元向量，$\boldsymbol{I}$ 是 $n\times n$ 的单位矩阵，上角标 T 表示矩阵转置。

Hopfield 网络的操作方法可以用几何学来解释。图 6.19 显示了一个三神经元网络，用三维空间的立方体表示。通常，有 n 个神经元的网络有 2^n 个可能的状态，与 n 维超立方体有关。在图 6.19 中，每个状态用一个顶点来表示。当使用新的输入向量时，网络从一个状态顶点移动到下一个状态顶点，直到网络稳定为止。

189 · 190

用什么可以确定稳定的状态顶点

稳定的状态顶点由权重矩阵 $\boldsymbol{W}$，当前输入向量 $\boldsymbol{X}$ 和矩阵阈值 θ 确定。如果输入向量有部分错误或不完善的地方，那么在几次迭代后，初始状态会收敛到稳定的状态顶点。

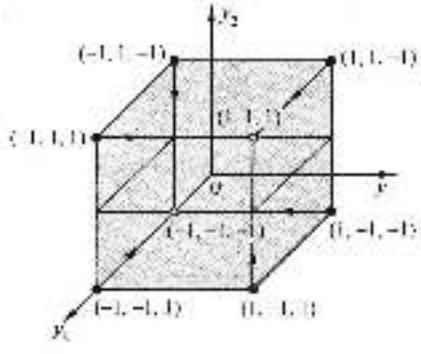


图 6.19 三神经元 Hopfield 网络所有可能状态的立方体表示

例如，假设网络需要记住两个相反的状态 (1, 1, 1) 和 (−1, −1, −1)，那么

$$\boldsymbol{Y}_1=\begin{bmatrix}1\\1\\1\end{bmatrix}\quad \boldsymbol{Y}_2=\begin{bmatrix}-1\\-1\\-1\end{bmatrix}$$

其中，$\boldsymbol{Y}_1$、$\boldsymbol{Y}_2$ 是三维向量。

我们也可以用行来表示这些向量，即转置向量：

$$\boldsymbol{Y}_1^{\mathrm{T}}=[1\quad 1\quad 1]\quad \boldsymbol{Y}_2^{\mathrm{T}}=[-1\quad -1\quad -1]$$

3×3 的单位矩阵 $\boldsymbol{I}$ 为：

$$\boldsymbol{I}=\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$$

因此可以确定权重矩阵为：

$$\boldsymbol{W}=\boldsymbol{Y}_1\boldsymbol{Y}_1^{\mathrm{T}}+\boldsymbol{Y}_2\boldsymbol{Y}_2^{\mathrm{T}}-2\boldsymbol{I}$$

或

$$\boldsymbol{W}=\begin{bmatrix}1\\1\\1\end{bmatrix}[1\quad 1\quad 1]+\begin{bmatrix}-1\\-1\\-1\end{bmatrix}[-1\quad -1\quad -1]-2\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}=\begin{bmatrix}0&2&2\\2&0&2\\2&2&0\end{bmatrix}$$

接下来用输入向量序列 $\boldsymbol{X}_1$ 和 $\boldsymbol{X}_2$ 来测试网络，$\boldsymbol{X}_1$ 和 $\boldsymbol{X}_2$ 分别和输出（或目标）向量 $\boldsymbol{Y}_1$ 和 $\boldsymbol{Y}_2$ 相等。我们希望看一下网络能否识别相似的模式。

如何测试 Hopfield 网络

首先用输入向量 $\boldsymbol{X}$ 来激活网络，然后计算实际的输出向量 $\boldsymbol{Y}$，最后将结果和初始输入向量 $\boldsymbol{X}$ 比较。

$$\boldsymbol{Y}_m = sign(\boldsymbol{W}\boldsymbol{X}_m - \boldsymbol{\theta}), \qquad m = 1,2,\cdots,M \tag{6.21}$$

191 其中，θ 为阈值矩阵。

本例中假设所有的阈值都为0。因此，

$$\boldsymbol{Y}_1 = sign\left\{\begin{bmatrix}0 & 2 & 2\\ 2 & 0 & 2\\ 2 & 2 & 0\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix} - \begin{bmatrix}0\\0\\0\end{bmatrix}\right\} = \begin{bmatrix}1\\1\\1\end{bmatrix}$$

以及

$$\boldsymbol{Y}_2 = sign\left\{\begin{bmatrix}0 & 2 & 2\\ 2 & 0 & 2\\ 2 & 2 & 0\end{bmatrix}\begin{bmatrix}-1\\-1\\-1\end{bmatrix} - \begin{bmatrix}0\\0\\0\end{bmatrix}\right\} = \begin{bmatrix}-1\\-1\\-1\end{bmatrix}$$

可以看到，$\boldsymbol{Y}_1 = \boldsymbol{X}_1$，$\boldsymbol{Y}_2 = \boldsymbol{X}_2$。因此，状态（1，1，1）和（-1，-1，-1）是稳定的。

其他状态怎么样

网络中有3个神经元，有8种可能的状态。剩下的6个状态都是不稳定的。但是，稳定的状态（也称作基本记忆）可以吸引其周围的状态。如表6.5所示，基本记忆（1，1，1）吸引了不稳定状态(-1，1，1)，(1，-1，1)，(1，1，-1)。这些不稳定的状态与基本记忆（1，1，1）相比，只在一个位置上有误差。另外，基本记忆（-1，-1，-1）吸引了不稳定状态（-1，-1，1）、(-1，1，-1)、(1，-1，-1)。这再次说明这些不稳定状态和基本记忆相比，在一个位置上有误差。因此，Hopfield网络确定可以作为*误差校正网络*。现在总结一下Hopfield网络的训练算法。

表6.5 3个神经元的Hopfield网络的操作

可能的状态			迭代	输入			输出			基本记忆		
				x_1	x_2	x_3	y_1	y_2	y_3			
1	1	1	0	1	1	1	1	1	1	1	1	1
-1	1	1	0	-1	1	1	1	1	1			
			1	1	1	1	1	1	1	1	1	1
1	-1	1	0	1	-1	1	1	1	1			
			1	1	1	1	1	1	1	1	1	1
1	1	-1	0	1	1	-1	1	1	1			
			1	1	1	1	1	1	1	1	1	1
-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	0	-1	-1	1	-1	-1	-1			
			1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	0	-1	1	-1	-1	-1	-1			
			1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	0	1	-1	-1	-1	-1	-1			
192			1	-1	-1	-1	-1	-1	-1	-1	-1	-1

步骤1： 存储。

n 个神经元的Hopfield网络需要存储 M 个基本记忆 $\boldsymbol{Y}_1$，$\boldsymbol{Y}_2$，…，$\boldsymbol{Y}_M$。神经元 i 到神经元 j 的突触权重计算方法是：

$$w_{ij} = \begin{cases} \sum_{m=1}^{M} y_{m,i}\, y_{m,j}, & i \neq j \\ 0, & i = j \end{cases} \tag{6.22}$$

其中，$y_{m,i}$和$y_{m,j}$是基本记忆$\boldsymbol{Y}_m$的第 i 个和第 j 个元素。神经元间的突触权重的矩阵表示为：

$$W = \sum_{m=1}^{M} Y_m Y_m^{\mathrm{T}} - MI$$

如果权重矩阵是对称的，其主对角线值为0，那么 Hopfield 网络可以存储一系列基本记忆（Cohen and Grossberg，1983），即

$$W = \begin{bmatrix} 0 & w_{12} & \cdots & w_{1i} & \cdots & w_{1n} \\ w_{21} & 0 & \cdots & w_{2i} & \cdots & w_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ w_{i1} & w_{i2} & \cdots & 0 & \cdots & w_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{ni} & \cdots & 0 \end{bmatrix} \tag{6.23}$$

其中，$w_{ij} = w_{ji}$。

一旦计算好权重，它们就保持不变。

步骤2：测试。

需要确定 Hopfield 网络有能力唤起所有的基本记忆。换句话说，网络必须在将基本记忆Y_m作为输入时记住它。即

$$x_{m,i} = y_{m,i}, \qquad i = 1,2,\cdots,n; \qquad m = 1,2,\cdots,M$$

$$y_{m,i} = sign\left(\sum_{j=1}^{n} w_{ij} x_{m,j} - \theta_i\right)$$

193

其中，$y_{m,i}$是实际输出向量Y_m的第i个元素，$x_{m,j}$是输入向量X_m的j个元素。以矩阵表示为：

$$X_m = Y_m, \qquad m = 1,2,\cdots,M$$

$$Y_m = sign(WX_m - \theta)$$

如果网络可以完全地记住所有基本记忆，我们就可以执行下一步。

步骤3：检索。

将一个位置的n维向量（探针向量）X输入网络并检索稳定状态。探针向量一般情况下表示混乱或不完善的基本记忆，也就是说：

$$X \neq Y_m, \qquad m = 1,2,\cdots,M$$

（a）通过下面的设置来初始化 Hopfield 网络的搜索算法：

$$x_j(0) = x_j \qquad j = 1,2,\cdots,n$$

并计算每个神经元的初始状态：

$$y_i(0) = sign\left(\sum_{j=1}^{n} w_{ij} x_j(0) - \theta_i\right), \qquad i = 1,2,\cdots,n$$

其中，$x_j(0)$是迭代次数$p=0$时探针向量X的第j个元素，$y_i(0)$是迭代次数$p=0$时神经元i的状态。

以矩阵形式，迭代次数$p=0$时的状态向量为：

$$Y(0) = sign[WX(0) - \theta]$$

（b）根据以下规则，更新状态向量的元素$Y(p)$：

$$y_i(p+1) = sign\left(\sum_{j=1}^{n} w_{ij} x_j(p) - \theta_i\right)$$

用于更新的神经元是异步选择的，也就是说，是随机的，并且每次选择一个。

重复迭代过程，直到状态向量不再变化为止，也就是达到稳定状态为止。稳定条件可以定义为：

$$y_i(p+1) = sign\left(\sum_{j=1}^{n} w_{ij} y_j(p) - \theta_i\right), \qquad i = 1,2,\cdots,n \tag{6.24}$$

或者以矩阵式表示为：

194

$$\boldsymbol{Y}(p+1)=sign[\boldsymbol{W}\boldsymbol{Y}(p)-\boldsymbol{\theta}] \tag{6.25}$$

如果检索是异步的，则 Hopfield 网络通常都能够收敛（Haykin，2008）。但是，这个稳定状态不一定必须是一个基本记忆，如果这个状态就是基本记忆，那么也不一定是最接近的那个。

例如，假设要在有五神经元 Hopfield 网络中存储 3 个基本记忆：

$$\boldsymbol{X}_1=(+1,+1,+1,+1,+1)$$
$$\boldsymbol{X}_2=(+1,-1,+1,-1,+1)$$
$$\boldsymbol{X}_3=(-1,+1,-1,+1,-1)$$

权重矩阵按公式（6.20）构建：

$$\boldsymbol{W}=\begin{bmatrix} 0 & -1 & 3 & -1 & 3 \\ -1 & 0 & -1 & 3 & -1 \\ 3 & -1 & 0 & -1 & 3 \\ -1 & 3 & -1 & 0 & -1 \\ 3 & -1 & 3 & -1 & 0 \end{bmatrix}$$

假设探针向量为：

$$\boldsymbol{X}=(+1,+1,-1,+1,+1)$$

如果将探针向量和基本记忆$\boldsymbol{X}_1$进行比较，就会发现两个向量仅在一位上有差别。因此，我们预测探针向量$\boldsymbol{X}$会收敛到基本记忆$\boldsymbol{X}_1$。但是，在使用上面描述的 Hopfield 网络训练算法时，得到了不同的结果。网络通过记忆产生的结果是记忆$\boldsymbol{X}_3$，是错误的记忆。

这个例子暴露了 Hopfield 网络固有的问题。

另一个问题是*存储容量*，或网络可以存储和正确检索的基本记忆的最大数量。Hopfield 用实验方法（Hopfield，1982）指出最多（M_{max}）能够存储在 n 神经元循环网络中的基本记忆个数是：

$$M_{max}=0.15n \tag{6.26}$$

我们还可以定义 Hopfield 网络中几乎全部基本记忆都可以完美检索的存储容量为（Amit，1989）：

195

$$M_{max}=\frac{n}{2\ln n} \tag{6.27}$$

如果想让所有的基本记忆都可以完美检索，会发生什么

可以看到，要想完美地检索所有的基本记忆，记忆数量会减半（Amit，1989）：

$$M_{max}=\frac{n}{4\ln n} \tag{6.28}$$

正如我们所看到的，Hopfield 网络的存储容量保持得较小，以便进行基本记忆检索。这就是 Hopfield 网络的主要限制。

严格地说，Hopfield 网络表现出记忆的*自动联想*功能。换句话说，Hopfield 可以检索混乱或不完整的记忆，但不能将它与不同的记忆联系起来。

相反，人类的记忆本质上是联想式的。一件事情可能提醒我们另一件事，另一件事再提醒另一件事，依此类推。我们用一连串的相关记忆来回想起忘掉的记忆。例如，如果我们想不起来把雨伞放在哪里了，就会试图回忆最后在什么地方拿着雨伞，在做什么事情，与谁在交谈。我们试图建立起联想链，从而回忆起忘掉的事情。

为什么 Hopfield 网络不能这么做

Hopfield 网络是单层网络，因此输出模式与输入模式所用的神经元是相同的。要将一个记忆和其他记忆关联起来，需要一个循环神经网络，其能够接收一个神经元集合中的输入模式并且在另一个神经元集合上产生一个相关的、但与其不同的输出模式。实际上，需要双层循环网络，

双向联想记忆。

6.7 双向联想记忆

双向联想记忆（Bidirectional Associative Memory，BAM）首先由 Bart Kosko 提出，是异质联想网络（Kosko，1987；1988）。它将集合 A 的模式和另一个集合 B 的模式关联起来，反之亦然。和 Hopfield 网络一样，尽管输入含糊或不完善，但 BAM 可以归纳并产生正确的输出。图 6.20 为基本的 BAM 架构。它包含两个完全连接的层：输入层和输出层。 196

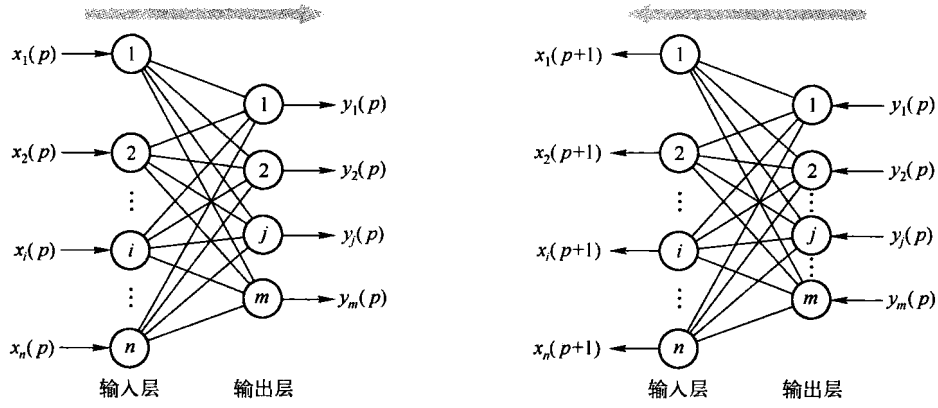


图 6.20 BAM 操作

BAM 如何工作

如图 6.20a 所示，输入向量 $\boldsymbol{X}(p)$ 作用到转置权重矩阵 $\boldsymbol{W}^{\mathrm{T}}$，产生输出向量 $\boldsymbol{Y}(p)$。然后输出向量 $\boldsymbol{Y}(p)$ 作用到权重矩阵 $\boldsymbol{W}$ 产生新的输入向量 $\boldsymbol{X}(p+1)$，如图 6.20b 所示。重复这个过程，直到输入和输出向量都不再变化，换句话说，BAM 达到了稳定状态。

BAM 方法的基本思想是存储模式对，当将集合 A 中的 n 维向量 $\boldsymbol{X}$ 作为输入时，BAM 方法回忆起集合 B 的 m 维向量 $\boldsymbol{Y}$，如果输入为 $\boldsymbol{Y}$，BAM 方法就回忆起 $\boldsymbol{X}$。

要开发 BAM，需要为想要存储的每个模式对创建一个相关矩阵。相关矩阵是一个由输入向量 $\boldsymbol{X}$ 乘以输出向量的转置 $\boldsymbol{Y}^{\mathrm{T}}$ 而得到的矩阵。BAM 权重矩阵是所有相关矩阵的和，即

$$\boldsymbol{W} = \sum_{m=1}^{M} \boldsymbol{X}_m \boldsymbol{Y}_m^{\mathrm{T}} \tag{6.29}$$

其中，M 是存储在 BAM 中的模式对的数量。

和 Hopfield 网络一样，BAM 一般使用带符号激活函数的 McCulloch 和 Pitts 神经元。

BAM 训练算法可以表示如下：

步骤 1：存储。

BAM 需要存储 M 对模式。例如，希望存储 4 对模式：

$$\text{集合 } A: \boldsymbol{X}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad \boldsymbol{X}_2 = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \quad \boldsymbol{X}_3 = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \quad \boldsymbol{X}_4 = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

197

集合 B：$\boldsymbol{Y}_1=\begin{bmatrix}1\\1\\1\end{bmatrix}$ $\boldsymbol{Y}_2=\begin{bmatrix}-1\\-1\\-1\end{bmatrix}$ $\boldsymbol{Y}_3=\begin{bmatrix}1\\-1\\1\end{bmatrix}$ $\boldsymbol{Y}_4=\begin{bmatrix}-1\\1\\-1\end{bmatrix}$

本例中，BAM 输入层必须有 6 个神经元，输出层有 3 个神经元。

权重矩阵的定义是：

$$\boldsymbol{W}=\sum_{m=1}^{4}\boldsymbol{X}_m\boldsymbol{Y}_m^{\mathrm{T}}$$

或者

$$\boldsymbol{W}=\begin{bmatrix}1\\1\\1\\1\\1\\1\end{bmatrix}\begin{bmatrix}1&1&1\end{bmatrix}+\begin{bmatrix}-1\\-1\\-1\\-1\\-1\\-1\end{bmatrix}\begin{bmatrix}-1&-1&-1\end{bmatrix}+\begin{bmatrix}1\\1\\-1\\-1\\1\\1\end{bmatrix}\begin{bmatrix}1&-1&1\end{bmatrix}+\begin{bmatrix}-1\\-1\\1\\1\\-1\\-1\end{bmatrix}\begin{bmatrix}-1&1&-1\end{bmatrix}=\begin{bmatrix}4&0&4\\4&0&4\\0&4&0\\0&4&0\\4&0&4\\4&0&4\end{bmatrix}$$

步骤 2：测试。

BAM 应该能接收集合 A 的任何向量并检索集合 B 的相关向量，以及接收集合 B 的任何向量并检索集合 A 的相关向量。因此，首先需要确定当输入为$\boldsymbol{X}_m$时，BAM 能回忆$\boldsymbol{Y}_m$，即

$$\boldsymbol{Y}_m=sign(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X}_m),\quad m=1,2,\cdots,M \tag{6.30}$$

例如，

$$\boldsymbol{Y}_1=sign(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X}_1)=sign\left\{\begin{bmatrix}4&4&0&0&4&4\\0&0&4&4&0&0\\4&4&0&0&4&4\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\\1\\1\end{bmatrix}\right\}=\begin{bmatrix}1\\1\\1\end{bmatrix}$$

接下来，需要确定当输入为$\boldsymbol{Y}_m$时，BAM 能回忆$\boldsymbol{X}_m$，即

198

$$\boldsymbol{X}_m=sign(\boldsymbol{W}\boldsymbol{Y}_m),\quad m=1,2,\cdots,M \tag{6.31}$$

例如，

$$\boldsymbol{X}_3=sign(\boldsymbol{W}\boldsymbol{Y}_3)=sign\left\{\begin{bmatrix}4&0&4\\4&0&4\\0&4&0\\0&4&0\\4&0&4\\4&0&4\end{bmatrix}\begin{bmatrix}1\\-1\\1\end{bmatrix}\right\}=\begin{bmatrix}1\\1\\-1\\-1\\1\\1\end{bmatrix}$$

在本例中，所有的向量都被准确无误地回忆起来，可以进行下一步。

步骤 3：检索。

未知向量（探针向量）$\boldsymbol{X}$ 输入到 BAM 中，检索存储的关联记忆。探针向量可以表示存储在 BAM 中来自集合 A（或集合 B）的含糊、不完整的模式。即

$$\boldsymbol{X}\neq\boldsymbol{X}_m,\qquad m=1,2,\cdots,M$$

(a) 通过以下设置来初始化 BAM 检索算法：

$$\boldsymbol{X}(0)=\boldsymbol{X},p=0$$

并计算在第 p 次迭代时 BAM 的输出：

$$\boldsymbol{Y}(p) = sign[\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X}(p)]$$

（b）更新输入向量 $\boldsymbol{X}(p)$：

$$\boldsymbol{X}(p+1) = sign\ [\boldsymbol{W}\boldsymbol{Y}(p)]$$

重复迭代直到达到平衡为止，进一步迭代时，输入向量和输出向量都不再改变。输入和输出模式可以成为一个关联对。

BAM 是无条件稳定的（Kosko，1992）。这就是说任何相关的知识都可以学习，没有不稳定的危险。之所以有这个重要的特性，是因为 BAM 使用前向和后向中的权重矩阵间的转置关系。

回到例子中，假设使用向量 $\boldsymbol{X}$ 作为探针向量。它和集合 A 的模式 $\boldsymbol{X}_1$ 相比有一个误差：

$$\boldsymbol{X} = (-1, +1, +1, +1, +1, +1)$$

该探针向量作为 BAM 的输入，产生了集合 B 的输出向量 $\boldsymbol{Y}_1$。将向量 $\boldsymbol{Y}_1$ 作为输入来检索集合 A 中的向量 $\boldsymbol{X}_1$。因此，BAM 确实能够进行误差校正。 199

BAM 和 Hopfield 网络之间的关系密切。如果 BAM 的权重矩阵是正方形且对称的，则有 $\boldsymbol{W} = \boldsymbol{W}^{\mathrm{T}}$。在本例中，输入和输出层大小相同，BAM 就可以简化成自相关的 Hopfield 网络。因此，Hopfield 网络可以被看做是 BAM 的特例。

Hopfield 网络在存储能力上的限制也可以扩展到 BAM。通常，BAM 存储的关联的最大数目不能超过较小层中的神经元数目。另外，更严重的问题是不正确的收敛。BAM 不会总是产生最接近的关联。事实上，一个稳定的关联可能只是轻微地关联到初始输入向量上。

BAM 仍有很多可以深入研究的主题。尽管它有缺点和限制，但 BAM 仍是最有用的人工神经网络之一。

神经网络在没有“老师”时可以学习吗

神经网络最主要的特征就是具有从环境中学习的能力，通过学习改善性能。到目前为止，我们介绍的都是有监督的或主动的学习——跟随外部的“老师”或指导员学习，具体表现形式就是为网络提供一系列的训练集。但是也存在另一种类型的学习：无监督学习。

和有监督的学习相比，无监督学习或自组织学习不需要外部的老师。在训练期间，神经网络接收到许多不同的输入模式，发现这些模式的重要特点，学习如何将输入数据分成合适的类别。无监督学习倾向于模拟大脑的神经生物组织。

无监督学习算法的目标是快速学习。事实上，自组织神经网络比反向传播网络的学习速度快，因此可以应用于实时环境。

6.8 自组织神经网络

自组织神经网络在处理预料之外和有变化的条件时是非常有效的。这一节里，我们介绍 Hebbian 学习和竞争学习，这些都是基于自组织网络的。

6.8.1 Hebbian 学习

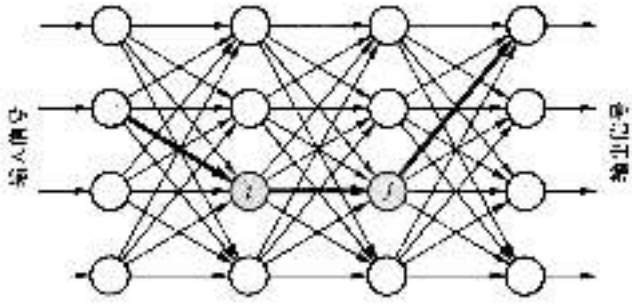
1949 年，神经生理学家 Donald Hebb 在生物学学习方面提出了一个关键思想，就是著名的 Hebb 法则（Hebb，1949）。Hebb 法则提出，如果神经元 i 距神经元 j 足够近且能够刺激神经元 j，并重复这样的活动，则两个神经元之间的突触连接就会加强，神经元 j 对来自神经元 i 的刺激就会格外敏感。 200

用下面两条规则来表示 Hebb 法则（Stent，1973）：

1）如果连接两侧的两个神经元同时被激活，那么连接的权重就会增加。

2）如果连接两侧的两个神经元不是同时被激活，那么连接的权重就会减少。

Hebb 法则是无监督学习的基础。这里的学习是无环境反馈的局部现象。图 6.21 为神经网络中的 Hebbian 学习。



[201] 图 6.21 神经网络中的 Hebbian 学习

使用 Hebb 法则，可以用下面的形式表达第 p 次迭代时权重 w_{ij} 的调整：

$$\Delta w_{ij}(p) = F[y_j(p), x_i(p)] \tag{6.32}$$

其中，$F[y_j(p), x_i(p)]$ 是前向突触活动和后向突触活动的函数。

作为特例，我们可以按下面的方式表示 Hebb 法则（Haykin，2008）：

$$\Delta w_{ij}(p) = \alpha y_j(p) x_i(p) \tag{6.33}$$

其中，α 是学习速率参数。

该公式作为活动产生式规则，它显示了一对神经元间突触连接的权重改变和输入、输出信号的产生是如何关联的。

Hebbian 学习说明权重只能增加。换句话说，Hebb 法则允许增加连接强度，但不提供连接强度减少的方法。因此，重复使用输入信号可能导致权重 w_{ij} 饱和。要解决这个问题，可以对突触权重的增长加以限制。要达到此目的，可以在公式（6.33）的 Hebb 法则中加入非线性遗忘因子（Kohonen，1989）：

$$\Delta w_{ij}(p) = \alpha y_j(p) x_i(p) - \phi y_j(p) w_{ij}(p) \tag{6.34}$$

其中，ϕ 是遗忘因子。

什么是遗忘因子

遗忘因子 ϕ 指在单次学习循环中权重的衰退。它的范围一般介于 0 和 1 之间。如果遗忘因子为 0，则神经网络仅能够增加突触权重的强度，因此权重可能增长到无穷大。另外，如果遗忘因子接近 1，网络就几乎记不住它要学习的内容。因此，应该选择一个很小的遗忘因子，通常介于 0.01 和 0.1 之间，在学习中仅允许微小的遗忘，同时限制权重的增加。

公式（6.34）可以写成通用活动产生式规则的形式：

$$\Delta w_{ij}(p) = \phi y_j(p)[\lambda x_i(p) - w_{ij}(p)] \tag{6.35}$$

其中，$\lambda = \alpha/\phi$。

通用活动积规则的含义是，如果在迭代次数为 p 时前向突触激活值（神经元的输入）$x_i(p)$ 小于 $w_{ij}(p)/\lambda$，则在迭代次数为 $p+1$ 时，修改后的突触权重 $w_{ij}(p+1)$ 会减少，其减少的数量与迭代次数为 p 时的后突触激活值（神经元的输出）$y_j(p)$ 成正比。如果 $x_i(p)$ 大于 $w_{ij}(p)/\lambda$，则在迭代次数为 $p+1$ 时，修改后的突触权重 $w_{ij}(p+1)$ 会增加，结果与神经元 j 的输出 $y_j(p)$ 成正比。也就是说，可以按与 $w_{ij}(p)/\lambda$ 相同的变量来确定需要改变的突触权重的激活平衡点。这种方法可以解决突触权重无限增加的问题。

下面是通用 Hebbian 学习算法。

步骤 1：初始化。

设置突触权重和阈值的初始值为 [0，1] 区间的小的随机值，同时给学习速率参数 α 和遗忘因子 ϕ 设置一个小的正数。

步骤2：激活。

计算迭代次数为 p 时神经元的输出：

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

其中，n 为神经元输入的个数，θ_j 是神经元 j 的阈值。

步骤3：学习。

更新网络中的权重：

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$$

其中，$\Delta w_{ij}(p)$ 是迭代次数为 p 时的权重的校正。

通过下面的通用活动积规则计算权重的校正：

$$\Delta w_{ij}(p) = \phi y_j(p)[\lambda x_i(p) - w_{ij}(p)] \quad 202$$

步骤4：迭代。

迭代次数 p 加1，回到步骤2继续执行，直到突触权重达到稳定状态值为止。

图6.22a显示了一个单层、有5个计算神经元的全连接反馈网络，我们通过这个图来解释Hebbian学习。每个神经元都使用带符号激活函数的McCulloch和Pitts模型表示。网络用下面输入向量集的通用活动积规则训练：

$$\boldsymbol{X}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \boldsymbol{X}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \boldsymbol{X}_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \boldsymbol{X}_4 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \boldsymbol{X}_5 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

其中，输入向量 $\boldsymbol{X}_1$ 是空向量。注意，向量 $\boldsymbol{X}_3$ 的第4个信号 x_4 和向量 $\boldsymbol{X}_4$ 的第3个信号 x_3 的值为1，而 $\boldsymbol{X}_2$ 和 $\boldsymbol{X}_5$ 中的信号 x_2 和 x_5 均为1。

本例中，初始权重矩阵表示为5×5的单位矩阵 $\boldsymbol{I}$。因此，在初始状态下，输入层的每个神经元和输出层相同位置的神经元相连，突触权重为1，与其他神经元间的突触权重为0。阈值是0到1之间的随机数，学习速率参数 α 和遗忘因子 ϕ 的取值分别为0.1和0.02。

训练后（如图6.22b所示），权重矩阵与初始单位矩阵 $\boldsymbol{I}$ 有所不同。输入层神经元2和输出层神经元5以及输入层神经元5和输出层神经元2之间的权重从0增加到2.0204。神经网络学习到了新的关联。同时，输入层神经元1和输出层神经元1之间的权重变为0。神经网络忽略了这个关联。

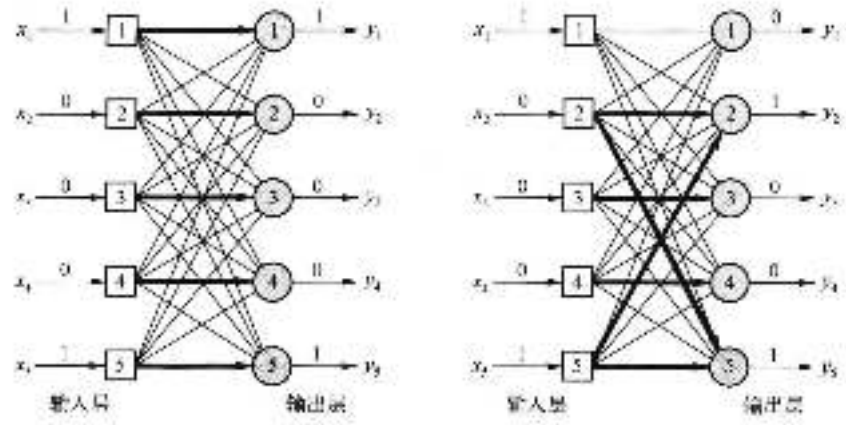
接下来测试网络。测试的输入向量（或探针向量）定义为：

$$\boldsymbol{X} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

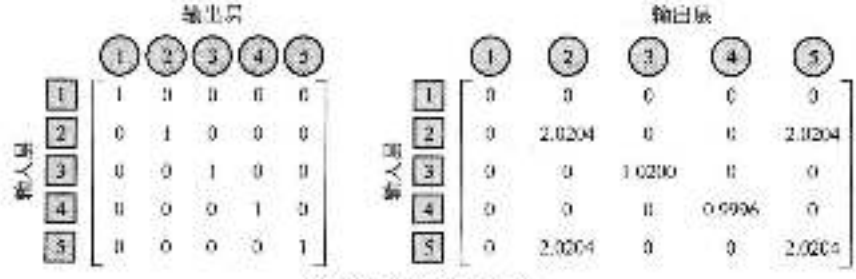
将探针向量输入给网络，得到：

$$\boldsymbol{Y} = sign\ (\boldsymbol{WX} - \boldsymbol{\theta})$$

$$\boldsymbol{Y} = sign\left\{ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2.0204 & 0 & 0 & 2.0204 \\ 0 & 0 & 1.0200 & 0 & 0 \\ 0 & 0 & 0 & 0.9996 & 0 \\ 0 & 2.0204 & 0 & 0 & 2.0204 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0.4940 \\ 0.2661 \\ 0.0907 \\ 0.9478 \\ 0.0737 \end{bmatrix} \right\} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$



a) 网络的初始和最终状态



b) 初始和最终的权重矩阵

图 6.22 单层神经网络的无监督 Hebbian 学习

意无疑问，网络的输入 x_5 和输出 y_3、y_5 关联起来了，因为在训练中 x_3、x_5 是一对。但网络不能将输入 x_3 和输出 y_1 连接起来，因为训练中单位输入 x_1 没有出现，因此网络将它忽略掉了。

203
!
204

因此，神经网络确实能够通过学习来发现经常同时出现的关联刺激因素。更重要的是，网络可以在没有“老师”的情况下自己学习。

6.8.2 竞争学习

另一种常见的无监督学习是竞争学习。在竞争学习中，神经元要通过竞争才能被激活。在 Hebbian 学习中，几个输出神经元可以同时被激活，而在竞争学习中，任意时刻仅有一个输出神经元被激活。在竞争中胜出的神经元叫做“胜者通吃”神经元。

竞争学习的基本思想在 20 世纪 70 年代初期就被提出了（Grossberg，1972；von der Malsburg，1973；Fukushima，1975）。但是直到 20 世纪 80 年代后期，这种方法才引起注意，当时 Teuvo Kohonen 提出了一种叫做自组织映射（Kohonen，1989）的特殊类型的人工神经网络，这种映射是基于竞争学习的。

什么是自组织特征映射

人脑是由大脑皮层控制的，大脑皮层是由数十亿个神经元和数千亿的突触组成的，结构非常复杂。大脑皮层既不统一也不均匀，它包括许多区域，这些区域通过其厚度和其中所包含的神经元的类型加以区分，不同的区域负责控制人类不同的活动（如运动、视觉、听觉和体觉等），因此它和不同的感官输入是相连接的。我们可以说，每个感官输入都会映射到大脑皮层的相应区域；换句话说，人类大脑皮层是一个自组织计算映射。

可以模拟自组织映射吗

Kohonen 提出了拓扑映射的构成原理（Kohonen，1990）。该原理表明，拓扑映射中输出神经元的空间位置和输入模式的某个特征相对应。Kohonen 提出了如图 6.23 所示的特征映射模型（Kohonen，1982）。这个模型包含了大脑中自组织映射的主要特征，并且可以很容易地在计算机中表达出来。

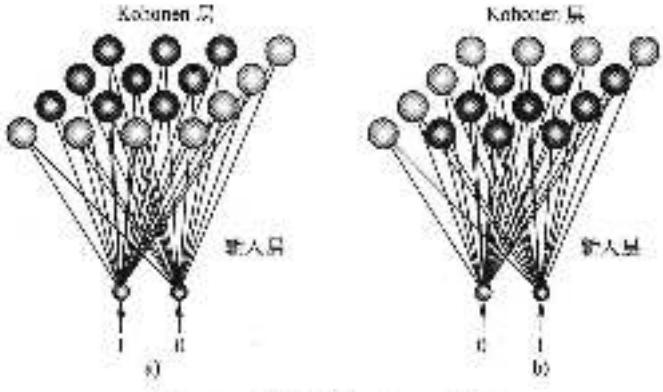


图 6.23 特征映射 Kohonen 模型

[265]

Kohonen 模型提供了拓扑映射，这种拓扑映射将固定数目的输入模式从输入层放置到高维输出层（亦称 Kohonen 层）。在图 6.23 中，Kohonen 层包含由 4×4 神经元组成的二维网络，每个神经元有两个输入。胜出的神经元用黑色表示，周围的神经元用灰色表示。胜出神经元周围的神经元在物理上是最接近胜出神经元的。

"物理上最接近"是多近

物理上最接近是多近，这是由网络设计人员决定的。胜出神经元可能在任何一边一个、两个甚至三个位置包含神经元。例如，图 6.23 中的胜出神经元周围有一个神经元。通常，Kohonen 网络训练开始时，胜出神经元周围有大量神经元。随着训练过程不断进行，周围神经元的数量会逐渐减少。

Kohonen 网络包含单层的计算神经元，但有两种不同类型的连接。一种是输入层神经元到输出层神经元的前向连接，另一种是输出层神经元间的横向连接，如图 6.24 所示。横向连接用来在神经元之间建立竞争。输出层中激活水平最高的神经元才会胜出（即为胜者通吃神经元）。该神经元是唯一产生输出信号的神经元，其他神经元的活动会在竞争中被压制。

当一个输入模式在网络上出现时，Kohonen 层上的每个神经元都会接收到该输入模式的完整拷贝，并且会被通过输入层和 Kohonen 层之间的突触连接权重上的路径所修正。横向反馈连接根据与获胜神经元之间的距离产生刺激或抑制的效应。实现这个功能需要使用墨西哥帽子函数（Mexican hat function），该函数描述了 Kohonen 层神经元间的突触权重。

墨西哥帽子函数是什么

图 6.25 所示的墨西哥帽子函数表示的是胜者通吃神经元之间的距离和 Kohonen 层内的连接的强度之间的关系。根据这个函数，最近的邻域（短程的横向刺激区）有最强烈的刺激效应；最远的邻域（抑制的区域）有轻微的抑制效应；很远的邻域（抑制区域的周围）有很微弱的刺激效应，因此通常被忽略。

在 Kohonen 网络中，神经元通过将权重从未激活的连接变为激活的连接来学习。只有胜出的

神经元和周围神经元可以学习。如果一个神经元不能响应给定的输入模式，这个神经元就不会学习。

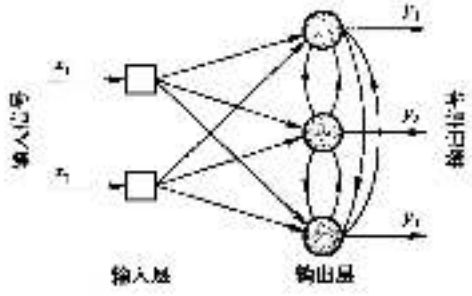


图 6.24 Kohonen 网络的架构

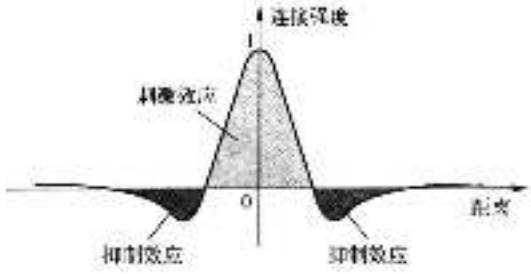


图 6.25 边缘连接的墨西哥帽子函数

胜者通吃神经元 j 的输出信号 y_j 的值为 1，其余的神经元（竞争中失败的神经元）的输出信号的值为 0。

标准的竞争学习规则（Haykin，2008）定义突触权重 w_{ij} 的改变量 Δw_{ij} 为：

$$\Delta w_{ij}=\begin{cases}\alpha(x_i-w_{ij}), & \text{如果神经元 } j \text{ 在竞争中胜出}\\ 0, & \text{如果神经元 } j \text{ 在竞争中失败}\end{cases} \tag{6.36}$$

其中，x_i 为输入信号，α 为学习速率参数（学习速率参数的取值范围为 0～1）。

竞争学习规则的影响在于使胜出神经元 j 的突触权重向量 $\boldsymbol{W}_j$ 向输入模式 $\boldsymbol{X}$ 靠拢。匹配准则等于向量间的最小欧几里得距离。

什么是欧几里得距离

$n\times 1$ 的向量 $\boldsymbol{X}$ 与 $\boldsymbol{W}_j$ 间的欧几里得距离为：

$$d=\|\boldsymbol{X}-\boldsymbol{W}_j\|=\left[\sum_{i=1}^{n}(x_i-w_{ij})^2\right]^{1/2} \tag{6.37}$$

206
207

其中，x_i 和 w_{ij} 分别为向量 $\boldsymbol{X}$ 和 $\boldsymbol{W}_j$ 的第 i 个元素。

向量 $\boldsymbol{X}$ 和 $\boldsymbol{W}_j$ 的相似程度是欧几里得距离 d 的倒数。在图 6.26 中，向量 $\boldsymbol{X}$ 和 $\boldsymbol{W}_j$ 的欧几里得距离是这两个向量顶点间的距离。图 6.26 清楚地表明，欧几里得距离越小，向量 $\boldsymbol{X}$ 和 $\boldsymbol{W}_j$ 就越相似。

为了确定与输入向量 $\boldsymbol{X}$ 最匹配的胜出神经元 $j_{\boldsymbol{X}}$，可以使用如下条件（Haykin，2008）：

$$j_{\boldsymbol{X}}=\min_j\|\boldsymbol{X}-\boldsymbol{W}_j\|,\quad j=1,2,\cdots,m \tag{6.38}$$

其中，m 是 Kohonen 层神经元的数量。

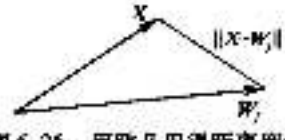


图 6.26 用欧几里得距离度量向量 $\boldsymbol{X}$ 和 $\boldsymbol{W}_j$ 的相似性

例如，假设要将二维输入向量 $\boldsymbol{X}$ 输入到三神经元 Kohonen 网络中：

$$\boldsymbol{X}=\begin{bmatrix}0.52\\0.12\end{bmatrix}$$

初始权重向量 $\boldsymbol{W}_j$ 为：

$$\boldsymbol{W}_1=\begin{bmatrix}0.27\\0.81\end{bmatrix}\quad \boldsymbol{W}_2=\begin{bmatrix}0.42\\0.70\end{bmatrix}\quad \boldsymbol{W}_3=\begin{bmatrix}0.43\\0.21\end{bmatrix}$$

用最小欧几里得距离来确定胜出（最匹配）的神经元 $j_{\boldsymbol{X}}$：

$$d_1=\sqrt{(x_1-w_{11})^2+(x_2-w_{21})^2}=\sqrt{(0.52-0.27)^2+(0.12-0.81)^2}=0.73$$

$$d_2 = \sqrt{(x_1 - w_{12})^2 + (x_2 - w_{22})^2} = \sqrt{(0.52 - 0.42)^2 + (0.12 - 0.70)^2} = 0.59$$

$$d_3 = \sqrt{(x_1 - w_{13})^2 + (x_2 - w_{23})^2} = \sqrt{(0.52 - 0.43)^2 + (0.12 - 0.21)^2} = 0.13$$

因此，神经元3胜出，根据公式（6.36）的竞争学习规则，对权重向量 $\boldsymbol{W}_3$ 进行更新。假设学习速率参数 α 为0.1，则可以得到：

$$\Delta w_{13} = \alpha(x_1 - w_{13}) = 0.1(0.52 - 0.43) = 0.01$$

$$\Delta w_{23} = \alpha(x_2 - w_{23}) = 0.1(0.12 - 0.21) = -0.01$$

208

在迭代次数为 $p+1$ 时，权重向量 $\boldsymbol{W}_3$ 更新为：

$$\boldsymbol{W}_3(p+1) = \boldsymbol{W}_3(p) + \Delta\boldsymbol{W}_3(p) = \begin{bmatrix} 0.43 \\ 0.21 \end{bmatrix} + \begin{bmatrix} 0.01 \\ -0.01 \end{bmatrix} = \begin{bmatrix} 0.44 \\ 0.20 \end{bmatrix}$$

通过每次迭代，胜出的神经元3的权重向量 $\boldsymbol{W}_3$ 和输入向量 $\boldsymbol{X}$ 会越来越近。

下面总结一下竞争学习算法（Kohonen，1989）。

步骤1：初始化。

用介于0～1之间的随机数对突触权重进行初始化设置，将学习速率参数 α 设为一个小的正数。

步骤2：激活和相似匹配。

用输入向量 $\boldsymbol{X}$ 激活Kohonen网络，在迭代次数为 p 时，用最小欧几里得距离找到胜出（最佳匹配）的神经元 j_X：

$$j_X(p) = \min_j \| \boldsymbol{X} - \boldsymbol{W}_j(p) \| = \left\{ \sum_{i=1}^{n} [x_i - w_{ij}(p)]^2 \right\}^{1/2}, \quad j = 1, 2, \cdots, m$$

其中，n 为输入层的神经元数量，m 为输出层（即Kohonen层）的神经元数量。

步骤3：学习。

更新突触权重：

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$$

其中，$\Delta w_{ij}(p)$ 是迭代次数为 p 时的权重校正量。

权重校正量由竞争学习规则确定：

$$\Delta w_{ij}(p) = \begin{cases} \alpha[x_i - w_{ij}(p)], & j \in \Lambda_j(p) \\ 0, & j \notin \Lambda_j(p) \end{cases} \tag{6.39}$$

其中，α 是学习速率参数，$\Lambda_j(p)$ 是胜出神经元 j_X 在迭代次数为 p 时的邻域函数。

邻域函数 Λ_j 通常有一个常数幅值。这说明所有位于拓扑邻域的神经元同时被激活，神经元间的关系和到胜出神经元 j_X 的距离是无关的。邻域函数的简单形式如图6.27所示。

209

矩形邻域函数 Λ_j 有二值特征，因此可以用以下方式表示神经元输出：

$$y_j = \begin{cases} 1, & j \in \Lambda_j(p) \\ 0, & j \notin \Lambda_j(p) \end{cases} \tag{6.40}$$

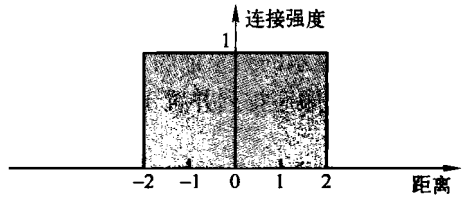


图6.27 矩形邻域函数

步骤4：迭代。

迭代次数 p 加1，回到步骤2继续进行，直到满足最小欧几里得距离，或在特征映射中没有显著改变为止。

可以使用由100个神经元排列成10行10列的二维网格构成的Kohonen网络来说明竞争学

习。这个网络用来对二维输入向量进行分类，换句话说，网络中的每个神经元仅对邻近的输入向量作出响应。

网络用1000个二维输入向量进行训练，输入向量是在 $-1 \sim +1$ 的正方形范围内随机产生的。初始突触权重向量也设为 $-1 \sim +1$ 间的随机值，学习速率参数 α 的取值为0.1。

图6.28显示了网络学习过程的不同阶段。每个神经元用一个黑点来表示，坐标分别为权重 w_{1j} 和 w_{2j}。图6.28a为随机分布在正方形区域的初始突触权重。图6.28b、图6.28c和图6.28d分别表示经过100、1000和10 000次迭代后输入空间中权重向量的情况。

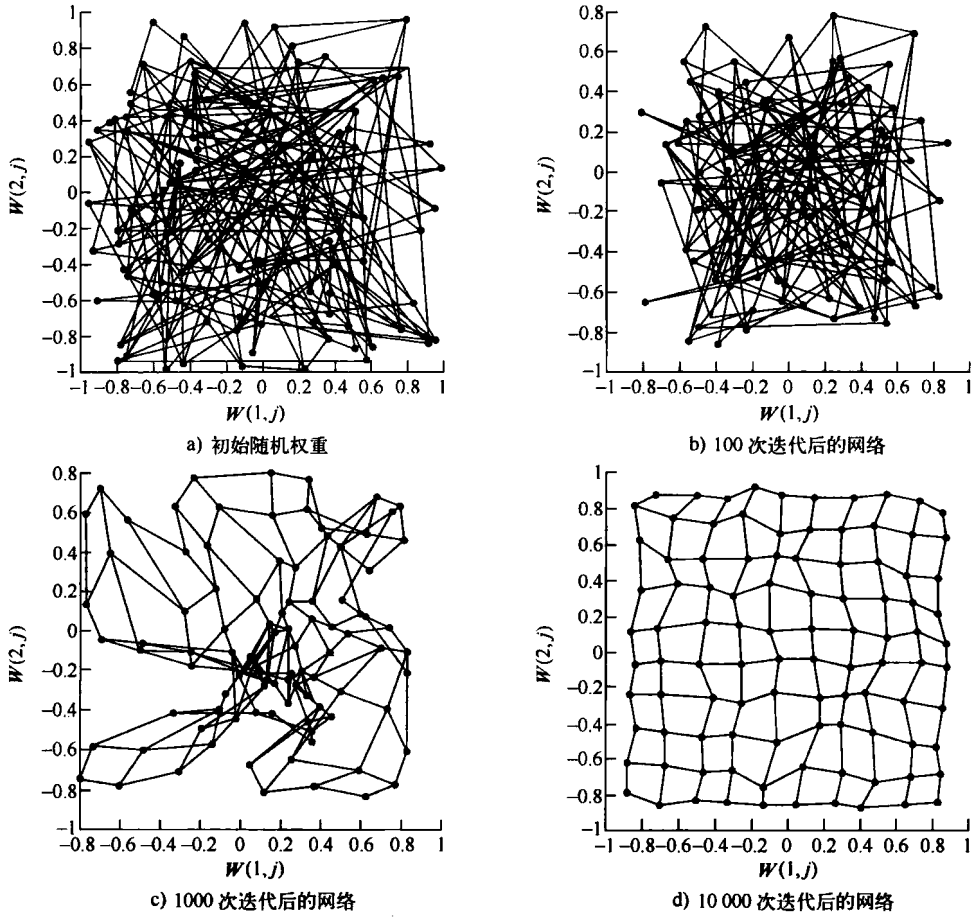


图6.28 Kohonen网络中的竞争学习

图6.28显示的结果证明了Kohonen自组织网络无监督学习的特征。在学习过程的最后，神经元按正确的顺序排列并置于整个输入空间中。每个神经元在自己的输入空间中都能够识别输入向量。

通过使用下面的输入向量来测试网络，以展示神经元是如何响应的：

$$\boldsymbol{X}_1 = \begin{bmatrix} 0.2 \\ 0.9 \end{bmatrix} \quad \boldsymbol{X}_2 = \begin{bmatrix} 0.6 \\ -0.2 \end{bmatrix} \quad \boldsymbol{X}_3 = \begin{bmatrix} -0.7 \\ -0.8 \end{bmatrix}$$

如图6.29所示，神经元6响应输入向量 $\boldsymbol{X}_1$，神经元69响应输入向量 $\boldsymbol{X}_2$，神经元92响应输入向量 $\boldsymbol{X}_3$。因此，图6.29显示的输入空间的特征映射是拓扑排序的，网格中神经元的空间位置与输入模式的某个特征相对应。

210 ~ 211

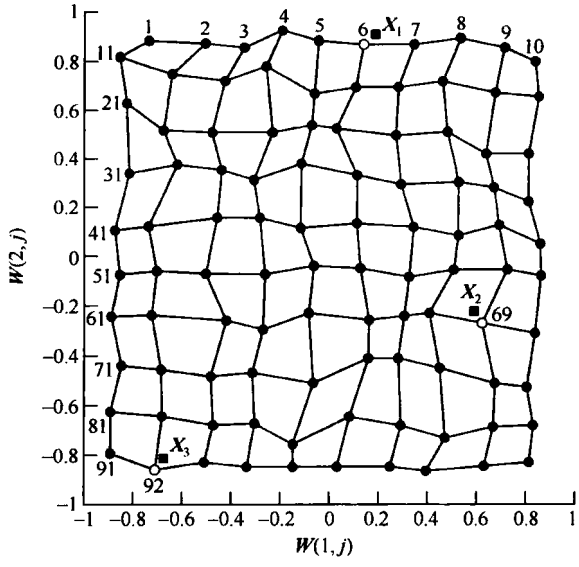


图 6.29 输入空间显示的拓扑排序特征映射

6.9 小结

在本章中，我们介绍了人工神经网络并讨论了与机器学习有关的基本思想；阐述了作为简单计算单元的感知器的概念，并讨论了感知器的学习规则；还探索了多层神经网络，并探讨了如何提高反向传播学习算法的计算效率；接下来介绍了循环神经网络，考虑了 Hopfield 网络训练算法和双向联想记忆（BAM）；最后，我们介绍了自组织神经网络，并探索了 Hebbian 学习规则和竞争学习。

本章中最重要的内容是：

- 机器学习涉及适应性机制，使得计算机能够从经验、实例、类比中学习。随着时间的推移，学习能力可以提升智能系统的表现性能。机器学习最常用的方法之一就是人工神经网络。
- 人工神经网络由一些非常简单并高度互联的，被称作神经元的处理器组成，这和人脑中的生物神经元类似。神经元之间通过带权重的链接相连，信号在这些链接上从一个神经元传递到另一个神经元。每个链接都有一个与之相关的数值权重。权重是人工神经网络中长期记忆的基本概念。它们表达了每个神经元输入的强度或重要性。神经网络通过不断调整这些权重来学习。 212
- 20 世纪 40 年代，Warren McCulloch 和 Walter Pitts 提出了一个简单的神经元模型，它现在依旧是大多数人工神经网络的基础。神经元计算输入信号的权重总和，并将结果和阈值比较。如果网络的净输入小于阈值，则神经元的输出为 -1。如果网络的净输入大于或等于阈值，则神经元被激活，且输出为 +1。
- Frank Rosenblatt 推荐了一种叫做感知器的最简单的神经网络。感知器的操作是基于 McCulloch 和 Pitts 神经模型的。它包含一个带有可调整突触权重的神经元和硬限幅器。感知器通过对权重的微调来减少实际输出和期望输出的差别，从而进行学习。初始权重是随机指定的，然后通过调整权重，得到和训练实例一致的输出。
- 感知器仅仅能够学习线性可分函数，不能在仅学习局部实例的基础上进行全局推广。Rosenblatt 感知器的局限可以通过改进神经网络的形式来克服，例如，用反向传播算法训

练的多层感知器。

- 多层感知器是一个前馈神经网络，含有来源神经元的输入层、至少一个计算神经元的中间层或隐含层和一个计算神经元输出层。输入层从外界接收输入信号，并将信号重新分配给中间层的所有神经元。隐含层检查特征，该层的神经元权重表示输入模式的特征。输出层建立整个网络的输出模式。
- 多层神经网络的学习过程和感知器是一样的。学习算法有两个阶段。首先向网络输入层输入训练模式，网络一层层地传送这个模式，直到输出层产生输出模式为止。如果和期望输出不同，则计算误差，并将误差从输出层传送到输入层。在传送误差时改变权重的值。
- 虽然得到了广泛的应用，但反向传播学习方法也不是没有任何问题。由于计算量巨大，因此训练速度缓慢，纯粹的反向传播算法在实际中很少应用。有几种方法可以改善计算的效率。例如，用双曲正切函数来表示 S 形函数时，多层神经网络的学习速率会提高很多。使用动量常数和自适应学习速率也会显著提升多层反向传播神经网络的性能。
- 当多层反向传播神经网络用于模式识别问题时，可用循环网络来模拟人类的联想记忆。循环网络是指从输出层到输入层有反馈循环的网络。John Hopfield 首次提出了在动态稳定网络中存储信息的理论，并且还提出了一个使用包含符号激活函数的 McCulloch 和 Pitts
213 神经元的单层循环神经网络。
- Hopfield 网络训练算法有两个基本的阶段：存储和检索。在第一个阶段中，网络需要存储一系列由所有神经元的当前输出确定的状态，或称作基本记忆。这是通过计算网络的权重矩阵来实现的。权重一旦被计算出，就会保持不变。在第二个阶段中，未知的、含糊的或不完整的基本记忆会被输入到网络中，然后计算网络输出并进行反馈，以便调整输入。这个过程将会一直重复，直到输出为一个常数为止。要使得基本记忆能够被检索，Hopfield 网络就应该保持较小的存储容量。
- Hopfield 网络表示的是一种自相关的记忆。它可以检索含糊或不完整的记忆，但是不能将一个记忆和其他记忆联系起来。为了克服这个限制，Bart Kosko 提出了双向联想记忆（BAM）。BAM 是一种异质相关网络，它将一个集合的模式和另一个集合的模式关联起来，反之亦然。就像 Hopfield 网络一样，BAM 在输入是含糊的或不完整的情况下，仍能进行推广并产生正确的输出。基本的 BAM 架构包含两个完全互连的层：输入层和输出层。
- BAM 的基本思想是存储模式对。因此，当输入为集合 A 中的 n 维向量 X 时，BAM 便会回忆起集合 B 中的 m 维向量 Y；而当输入为向量 Y 时，BAM 则会回忆起输出向量 X。Hopfield 存储容量的限制也可以扩展到 BAM 中。BAM 中存储的关联的数量不能超过较小层中神经元的数量。另一个问题是不正确的收敛，也就是说，BAM 不总是能产生最接近的关联。
- 与有监督学习（即有“老师”把训练集输入网络中来学习）不同，无监督学习或自组织学习不需要老师。在训练期间，神经网络收到许多不同的输入模式，然后分析它们显著的特点，并学习如何将输入进行分类。
- Hebb 法则由 Donald Hebb 在 20 世纪 40 年代后期提出。这个法则说明，如果神经元 i 和神经元 j 之间足够近，以至 i 可以刺激 j 并且重复地参加 j 的活动，则这两个神经元之间的突触连接就会加强，并且神经元 j 对来自神经元 i 的刺激也更加敏感。该法则是无监督学习的基础。这里的学习是没有来自环境反馈的局部现象。
- 另一个常见的无监督学习是竞争学习。此时，神经元通过相互竞争来激活。输出神经元
214 是竞争中的胜出者，也叫做“胜者通吃”的神经元。虽然在 20 世纪 70 年代早期就提出了竞争学习，但一直被忽略，直到 80 年代后期 Teuvo Kohonen 提出了人工神经网络的一

个特殊类别——自组织特征映射，这种情况才得以改观。他明确阐述了拓扑映射的原理，说明了在拓扑映射中输出神经元的空间位置和输入模式的某个特征相关联。

- Kohonen 网络由单层的计算神经元组成，但存在两种不同类型的连接，即输入层神经元到输出层神经元的前向连接，以及输出层神经元之间的横向连接。其中，后者用于创建神经元间的竞争。在 Kohonen 网络中，神经元通过从非激活状态到激活状态的变化来改变权重进行学习。仅胜出的神经元及其邻居允许学习。如果神经元对给定的输入模式没有响应，那么这个神经元就不会进行学习。

复习题

1. 人工神经网络如何模拟人脑？描述两种主要的学习范例：有监督学习和无监督（自组织）学习。两种范例有何区别？
2. 把感知器作为生物模型时存在什么问题？感知器如何学习？说明感知器学习二值逻辑函数 OR 的学习过程。为什么感知器仅可以学习线性可分函数？
3. 什么是全连接多层感知器？请构建一个有 6 个神经元的输入层、4 个神经元的隐含层和 2 个神经元的输出层的多层感知器。隐含层的用途是什么？隐含了什么？
4. 多层神经网络如何学习？推导反向传播训练算法。说明多层网络学习二值逻辑函数 Exclusive – OR 的过程。
5. 反向传播学习算法的主要问题是什么？在多层神经网络中如何加快学习速度？定义通用 delta 规则。
6. 什么是循环神经网络？它如何学习？构建一个有 6 个神经元的 Hopfield 网络并解释其操作过程。什么是基本记忆？
7. 推导 Hopfield 网络训练算法。说明如何在有 6 个神经元的 Hopfield 网络中存储 3 个基本记忆。
8. delta 规则和 Hebb 法则是两种神经网络的学习方法，比较两种方法的不同之处。
9. 记忆的自相关和异相关类型间的差别是什么？什么是双向联想记忆（BAM）？BAM 如何工作？ 215
10. 推导 BAM 训练算法。BAM 存储容量有什么限制？比较 BAM 和 Hopfield 网络的存储容量。
11. Hebb 法则是怎样表达的？推导活动积规则和通用活动积规则。以往因子的含义是什么？推导通用 Hebb 学习法则。
12. 什么是竞争学习？Hebbian 学习和竞争学习间的差别是什么？描述特征映射 Kohonen 模型。推导竞争学习算法。

参考文献

Amit, D.J. (1989). *Modelling Brain Functions: The World of Attractor Neural Networks*. Cambridge University Press, New York.

Bryson, A.E. and Ho, Y.C. (1969). *Applied Optimal Control*. Blaisdell, New York.

Caudill, M. (1991). Neural network training tips and techniques, *AI Expert*, January, 56–61.

Cohen, M.H. and Grossberg, S. (1983). Absolute stability of global pattern formation and parallel memory storage by competitive networks, *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13, 815–826.

Fu, L.M. (1994). *Neural Networks in Computer Intelligence*. McGraw-Hill, Singapore.

Fukushima, K. (1975). Cognition: a self-organizing multilayered neural network, *Biological Cybernetics*, 20, 121–136.

Grossberg, S. (1972). Neural expectation: cerebellar and retinal analogs of cells fired by learnable or unlearned pattern classes, *Kybernetik*, 10, 49–57.

Guyon, I.P. (1991). Applications of neural networks to character recognition, *International Journal of Pattern Recognition and Artificial Intelligence*, 5, 353–382.

Haykin, S. (2008). *Neural Networks and Learning Machines*, 3rd edn. Prentice Hall, Englewood Cliffs, NJ.

Hebb, D.O. (1949). *The Organisation of Behaviour: A Neuropsychological Theory*. John Wiley, New York.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences of the USA*, 79, 2554–2558.

Jacobs, R.A. (1988). Increased rates of convergence through learning rate adaptation, *Neural Networks*, 1, 295–307.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps, *Biological Cybernetics*, 43, 59–69.

Kohonen, T. (1989). *Self-Organization and Associative Memory*, 3rd edn. Springer-Verlag, Berlin, Heidelberg.

Kohonen, T. (1990). The self-organizing map, *Proceedings of the IEEE*, 78, 1464–1480.

Kosko, B. (1987). Adaptive bidirectional associative memories, *Applied Optics*, 26(23), 4947–4960.

Kosko, B. (1988). Bidirectional associative memories, *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-18, 49–60.

Kosko, B. (1992). *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Prentice Hall, Englewood Cliffs, NJ.

216

McCulloch, W.S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, 5, 115–137.

Medsker, L.R. and Liebowitz, J. (1994). *Design and Development of Expert Systems and Neural Computing*. Macmillan College Publishing, New York.

Minsky, M.L. and Papert, S.A. (1969). *Perceptrons*. MIT Press, Cambridge, MA.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain, *Psychological Review*, 65, 386–408.

Rosenblatt, F. (1960). Perceptron simulation experiments, *Proceedings of the Institute of Radio Engineers*, 48, 301–309.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986). Learning representations by back-propagating errors, *Nature* (*London*), 323, 533–536.

Shepherd, G.M. and Koch, C. (1990). Introduction to synaptic circuits, *The Synaptic Organisation of the Brain*, G.M. Shepherd, ed., Oxford University Press, New York, pp. 3–31.

Shynk, J.J. (1990). Performance surfaces of a single-layer perceptron, *IEEE Transactions on Neural Networks*, 1, 268–274.

Shynk, J.J. and Bershad, N.J. (1992). Stationary points and performance surfaces of a perceptron learning algorithm for a nonstationary data model, *Proceedings of the International Joint Conference on Neural Networks*, Baltimore, MD, vol. 2, pp. 133–139.

Stent, G.S. (1973). A physiological mechanism for Hebb's postulate of learning, *Proceedings of the National Academy of Sciences of the USA*, 70, 997–1001.

Stork, D. (1989). Is backpropagation biologically plausible? *Proceedings of the International Joint Conference on Neural Networks*, Washington, DC, vol. 2, pp. 241–246.

Stubbs, D.F. (1990). Six ways to improve back-propagation results, *Journal of Neural Network Computing*, Spring, 64–67.

Touretzky, D.S. and Pomerlean, D.A. (1989). What is hidden in the hidden layers? *Byte*, 14, 227–233.

Von der Malsburg, C. (1973). Self-organisation of orientation sensitive cells in the striate cortex, *Kybernetik*, 14, 85–100.

Watrous, R.L. (1987). Learning algorithms for connectionist networks: applied gradient methods of nonlinear optimisation, *Proceedings of the First IEEE International Conference on Neural Networks*, San Diego, CA, vol. 2, pp. 619–627.

217

进化计算

本章介绍进化计算，包括遗传算法、进化策略和遗传编程以及它们在学习中的应用。

7.1 进化是智能的吗

智能可以定义为系统调整自身行为来适应不断变化的环境的能力。根据 Alan Turing 的观点（Turing，1950），系统的形态或外观和智能是没有关系的。但是，根据常识，我们知道智能行为的迹象在人类社会中是很容易观察到的。我们本身就是进化的产物，因此通过模拟进化的过程，我们可以期待创建出智能的行为。进化计算就是在计算机上模拟进化过程。模拟的结果是基于简单规则的一系列最优化算法。最优化迭代改进解决方案的质量，直到找到最理想的，至少是最可行的解决方案。

但进化真的是智能的吗？我们可以考虑一个单组织生物体的行为，归纳推理出其环境中不为人知的方面（Fogel et al.，1966）。如果经过连续几代的繁衍，这种生物体存活了下来，就可以说这种生物体有学习的能力，可以预知环境的变化。从人类的角度来看，进化是一个非常缓慢而曲折的过程，但在计算机中模拟进化不会花上几十亿年！

机器学习的进化方法基于自然选择和遗传的计算模型。我们称之为*进化计算*，它是结合了*遗传算法*、*进化策略*和*遗传编程*的术语。所有这些技术都是通过使用*选择*、*突变*和*繁殖*过程来模拟进化的。

7.2 模拟自然进化

1858 年 7 月 1 日，查尔斯·达尔文在伦敦林奈学会发表了他的进化理论。这一天标志着生物学革命的开始。达尔文的经典进化理论，魏斯曼的自然选择理论以及孟德尔的遗传学概念一起构成了现在的新达尔文主义（Mayr，1988；Gould and Keeton，1996）。 219

*新达尔文主义*的基础是繁殖、突变、竞争和选择过程。繁殖能力是生命最本质的特征。突变能力保证了生物体能在不断变化的环境中繁殖。因为自然界的有限空间限制了物种的扩张，竞争和选择过程通常发生在自然界。

如果在计算机中模拟进化的过程，那么在自然界的生命中，什么可以通过进化来优化呢？进化是一个维护或增加种群在特定环境中生存和繁殖能力的过程（Conner and Hartl，2004）。这种能力也称为进化适应性。虽然不能对适应性进行直接测量，但可以通过环境中群体的生态学和功能形态学对其进行估计（Hoffman，1989）。进化适应性也可以看做对群体预见环境变化的能力的度量（Atmar，1994）。因此，适应性（或预见变化），并充分做出反应，对这一功能的定量度量，可以被看做是自然生命具有的可以优化的品质。

我们可以使用*适应性拓扑*的概念来说明适应性（Wright，1932）。用地形图来表示一个给定的环境，每个山峰代表物种的适应性极值。在进化过程中，给定种群的每个物种沿着斜坡向山峰前进。随着时间的推移，环境条件不断发生变化。因此，物种必须不断地调整它们的路线。最后，只有最适应的物种才能到达山峰。

适应性拓扑是一个连续函数。它模拟的环境或自然拓扑不是静态的。随着时间的推移，拓扑形状发生改变，所有的物种要不断地经历选择。进化的目标就是产生适应性增加的后代。

但是如何繁殖适应性不断增加的个体呢？Michalewicz（1996）基于兔子种群作了一个简单的

解释。有一些兔子跑得比较快。由于它们在逃避狐狸的追捕中存活下来并且继续繁殖的机会更大，因此可以说这些兔子在适应性上具有优势。当然，一些跑得慢的兔子也可以存活下来。因此，跑得慢的兔子和快的兔子、跑得慢的兔子之间以及跑得快的兔子之间都可以繁殖。换句话说，繁殖使这些兔子的基因相混合。如果双亲都有较强的适应性，那么在基因混合后，遗传给下一代良好适应性的机会就很大。随着时间的推进，兔子这个种群能跑得更快，以适应狐狸的威胁。但是，环境条件的改变同时对肥胖和聪明的兔子也有利。为了达到最优的生存，兔子种群的
220 遗传结构也相应地变化。同时，跑得快的兔子和聪明的兔子也导致狐狸变得更快更聪明。自然进化是持续的、没有终点的过程。

是否可以在计算机中模拟自然进化的过程

现在已经有了几种进化计算的方法，它们都是模拟自然进化的。这些方法通常都是先创建某个个体的种群，然后评估其适应性，最后通过基因操作产生新的种群，再将这个过程重复一定的次数。但可以用不同的方法来实现进化计算。我们可以从遗传算法（GA）开始，因为大多数其他的进化算法都可以看做是 GA 的变体。

在 20 世纪 70 年代早期，进化计算的创始人之一 John Holland 提出了遗传算法的概念（Holland, 1975）。他的目标是让计算机完全模拟自然界。作为计算机科学家，Holland 的遗传算法用一系列程序步骤来表示将人造“染色体”的一个种群进化到另一个种群的过程。该算法使用“自然”选择机制和遗传学的交叉、突变机制。每个染色体包含许多的“基因”，每个基因用 0 或 1 表示，如图 7.1 所示。

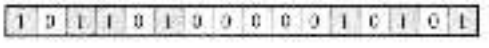


图 7.1 人工染色体的 16 位二进制数字串

虽然没有外界告知该如何做，自然界本身就具有适应环境并且学习的能力。也就是说，自然界是盲目地寻找优秀染色体的。遗传算法也这么做。将遗传算法和问题联系在一起的两种机制是：编码和评估。

在 Holland 的工作中，用 0 和 1 的数字串来表示染色体，进而实现对其编码。虽然人们也发明了很多其他的编码技术（Goldberg and Sastry, 2010），但没有一种技术能够适用于所有的问题。目前，位串仍是最通用的技术。

评估函数用来评估染色体的表现性能或适应性（遗传算法中的评估函数和自然进化中环境的作用相同）。遗传算法通过测量染色体个体的适应性来完成繁殖。在繁殖时，交叉操作交换两个染色体中的一部分，突变操作改变染色体上某个随机位置的基因值。因此，在经过数次连续的繁殖后，适应性较弱的染色体就会灭绝，而适应性最强的染色体逐渐统治了种群。这是一个简单的方法，但即使是最简易的繁殖机制也能显示出高度复杂的行为并能够解决复杂的问题。

221 现在我们讨论遗传算法的一些细节。

7.3 遗传算法

让我们从定义开始：遗传算法是基于生物进化机制的随机搜索算法。假定有一个定义清晰的要解决的问题，并用二进制数字串表示候选的解决方案，则基本遗传算法如图 7.2 所示。遗传算法的主要步骤有（Davis, 1991; Mitchell, 1996）：

步骤 1：用固定长度的染色体表示问题变量域，选择染色体种群数量为 N，交叉概率为 p_c，突变概率为 p_m。

步骤 2：定义适应性函数来衡量问题域上单个染色体的性能或适应性。适应性函数是在繁殖

过程中选择配对染色体的基础。

步骤3：随机产生一个大小为 N 的染色体的种群。

$$x_1, x_2, \cdots, x_N$$

步骤4：计算每个染色体的适应性：

$$f(x_1), f(x_2), \cdots, f(x_N)$$

步骤5：在当前种群中选择一对染色体。双亲染色体被选择的概率和其适应性有关。适应性高的染色体被选中的概率高于适应性低的染色体。

步骤6：通过执行遗传操作——交叉和突变产生一对后代染色体。

步骤7：将后代染色体放入新种群中。

步骤8：重复步骤5，直到新染色体种群的大小等于初始种群的大小 N 为止。

步骤9：用新（后代）染色体种群取代初始（双亲）染色体种群。

步骤10：回到步骤4，重复这个过程直到满足终止条件为止。

我们看到，遗传算法是一个迭代过程。每次迭代称为一代。简单遗传算法的典型迭代次数在50～500代之间（Mitchell，1996）。某一代的全部集合称为run。在run的最后，我们期望找到一个或多个高适应性的染色体。

遗传算法中有传统的终止条件吗

由于遗传算法使用随机搜索方法，因此在超级染色体出现之前，种群的适应性可能在几代中保持稳定。这时使用传统的终止条件可能会出现问题。一个常用的方法是在指定遗传代数后终止遗传算法，并检查种群中的最优的染色体。如果没有得到满意的解决方案，遗传算法会重新启动。

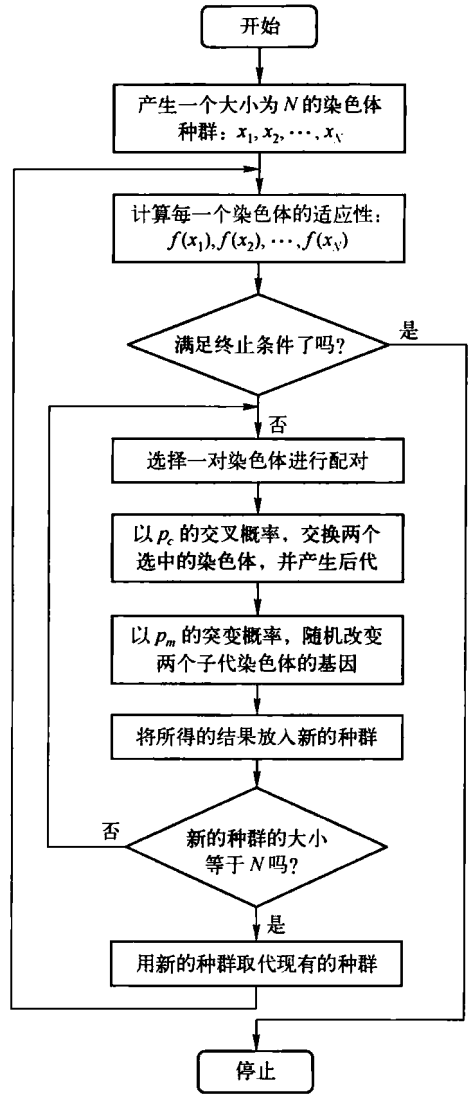


图7.2 基本遗传算法

222

可以通过一个简单的例子来理解遗传算法是如何工作的。假设要寻找函数（$15x - x^2$）在 x 的范围为0～15时的最大值。为方便起见，假设 x 仅取整数值。因此，染色体只要用4个基因就可以构建了，其编码方式如表7.1所示。

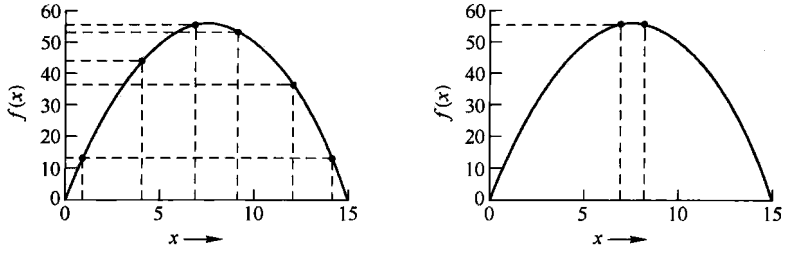
表7.1 编码方式

整 数	二进制编码	整 数	二进制编码	整 数	二进制编码
1	0001	6	0110	11	1011
2	0010	7	0111	12	1100
3	0011	8	1000	13	1101
4	0100	9	1001	14	1110
5	0101	10	1010	15	1111

假设染色体种群的大小 N 为6，交叉概率 p_c 为0.7，突变概率 p_m 为0.001（交叉概率和突变概率的取值为遗传算法的典型值）。本例中的适应性函数为：

$$f(x) = 15x - x^2$$

遗传算法用随机产生的0和1填充6个4位的数字串来创建染色体的初始种群。初始种群看上去如表7.2所示，适应性函数中染色体的初始位置如图7.3a所示。



a) 染色体初始位置　　b) 染色体最终位置

图7.3　适应性函数和染色体位置

在实际问题中，一个种群中一般会有数千个染色体。

接下来计算每个染色体的适应性，其结果如表7.2所示。初始种群的平均适应性为36。为了改善适应性，初始种群会通过选择、交叉和突变这样的遗传操作而有所改变。

在自然选择中，只有适应性最佳的个体才能存活、繁殖，并将基因传给下一代。遗传算法使用此类的方法，但不同的是，从上一代到下一代，染色体种群的大小保持不变。

表7.2　染色体随机产生的初始种群

染色体编号	染色体串	解码后的整数	染色体适应性	适应性比率（%）
*X*1	1 1 0 0	12	36	16.5
*X*2	0 1 0 0	4	44	20.2
*X*3	0 0 0 1	1	14	6.4
*X*4	1 1 1 0	14	14	6.4
*X*5	0 1 1 1	7	56	25.7
*X*6	1 0 0 1	9	54	24.8

如何在改进种群平均适应性的同时保持种群的大小不变

表7.2的最后一列为单个染色体适应性和种群总适应性的比值。这个比值决定了染色体被选中进行配对的概率。因此，染色体*X*5和*X*6被选中的概率最高，而染色体*X*3和*X*4被选中的概率比较低。结果，染色体的平均适应性随着遗传的进行逐渐提高。

最常用的染色体选择技术是轮盘选择（Goldberg，1989；Davis，1991）。图7.4显示了本例的轮盘。如图所示，轮盘上的每一片都代表一个染色体，每片的面积等于该染色体的适应性比值，如表7.2所示。例如，染色体*X*5和*X*6（适应性最高的染色体）面积最大，而染色体*X*3和*X*4（适应性最低）在轮盘上只占据很小的一片。为了选择一个染色体用于配对，在[0，100]之间产生一个随机数，轮盘上区间段正好包含这个随机数的染色体会被选中。就像轮盘上有一个指针，每个染色体在轮盘上都有一个和自身适应性相对应的片，轮盘旋转后，指针在某一片上停住，那个片对应的染色体就被选中了。

223
~
225

在本例中，初始种群有6个染色体。因此，为了保证下一代中有相同数量的染色体，轮盘应旋转6次，第一对可能选择*X*6和*X*2为双亲，第二对可能选择*X*1和*X*5，最后一对可能是*X*2和*X*5。

一旦选好双亲，就执行交叉操作。

交叉操作如何执行

首先，交叉操作随机选择交叉点（这个交叉点就是亲代染色体的“断裂”点），并交换染色

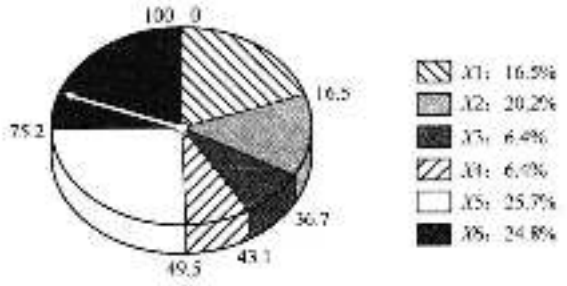


图 7.4 轮盘选择

体交叉点后的部分，从而产生两个新的子代染色体。例如，染色体 X6 和 X2 在第二个基因出交换彼此交叉点后的部分，产生两个后代，如图 7.5 所示。

如果两个染色体没有交叉，那么就克隆自己，子代是亲代染色体的精确副本。例如，亲代染色体 X2 和 X5 有可能没有交叉，那么创建的子代就是它们自身的副本，如图 7.5 所示。

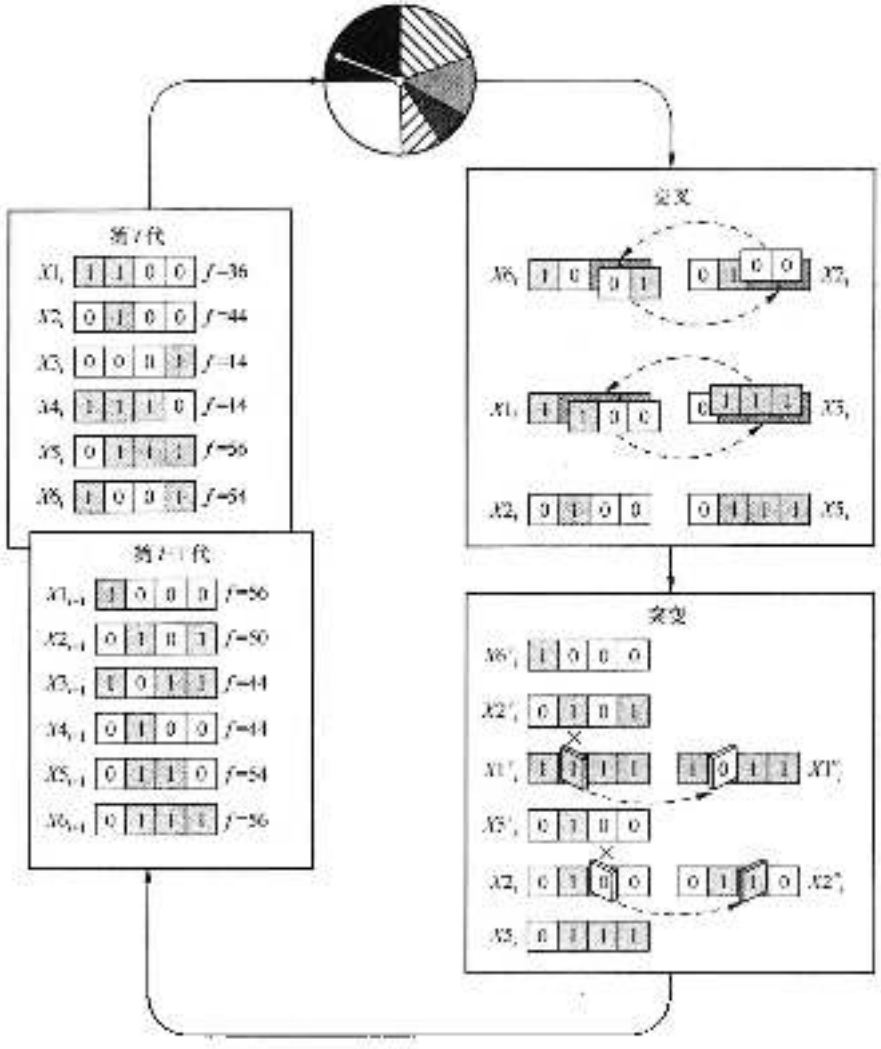


图 7.5 GA 循环

交叉概率为0.7时一般可以得到不错的效果。在完成选择和交叉后，染色体种群的平均适应性得到改善，即从36增加到42。

突变代表什么

突变在自然界中很少发生，它表示基因发生了改变。突变可能导致适应性显著提高，但大多数情况下会产生有害的结果。

那究竟为什么要使用突变呢？Holland是把突变当做一个后台操作（Holland，1975）提出的。它的作用是确保搜索算法不会陷入局部最优值。选择和交叉操作在得到类似的解决方案后可能停滞。在这种情况下，所有的染色体一样，因此种群的平均适应性不可能得到提高。但是，解决方案还可能进一步优化，或者还有更合适的局部最优值，却因为搜索算法不能再向下进行而无法得到更好的结果。突变等同于随机搜索，它避免了遗传多样性的丧失。

突变操作如何工作

突变操作就是随机选择染色体中的某个基因并反转其值。例如，在图7.5中，*X*1′在第二位基因处突变，*X*2在第三位基因处突变。突变可以以某种可能性发生在染色体的任何一个基因上。在自然界中，突变的概率非常小。因此，在遗传算法中也应保持很小的取值，一般在0.001到0.01之间。

遗传算法确保种群的适应性能够不断地得到改善，在繁殖一定代数后（通常有几百代），种群进化到接近最优的情况。在本例中，最终的种群中仅包含染色体 0 1 1 1 和 1 0 0 0。图7.3b显示了染色体在适应性函数中的最终位置。

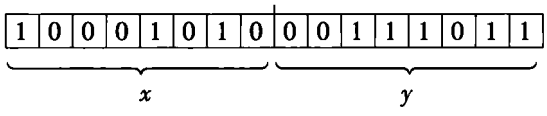
在本例中，问题仅有一个变量，因此很容易表示。假设现在需要找到有两个变量的“峰”函数的最大值：

226 ~ 227

$$f(x,y) = (1-x)^2 e^{-x^2-(y+1)^2} - (x - x^3 - y^3) e^{-x^2-y^2}$$

其中，x，y的取值范围是$-3 \sim 3$。

首先是将问题的变量表示为染色体。换句话说，将参数x和y表示为连接起来的二进制字符串。



其中，每个参数都用8位的二进制位来表示。

然后选择染色体种群的大小（例如为6）并随机产生初始种群。

接下来计算每个染色体的适应性。这通过两个步骤来完成。第一步，将染色体解码，将其转换成两个实数x和y，其取值范围在$-3 \sim 3$之间。第二步，将解码后的x、y的值带入到“峰”函数。

解码如何进行

首先，将16位的染色体分割成两个8位的数字串：

1 0 0 0 1 0 1 0 和 0 0 1 1 1 0 1 1

然后将这两个串的二进制转换成十进制数：

$$(10001010)_2 = 1\times2^7 + 0\times2^6 + 0\times2^5 + 0\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$$
$$= (138)_{10}$$

和

$$(00111011)_2 = 0\times2^7 + 0\times2^6 + 1\times2^5 + 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$$
$$= (59)_{10}$$

8位二进制表示的整数值的范围是$0 \sim (2^8 - 1)$，将其映射到实际范围是$-3 \sim 3$的参数x和y：

$$\frac{6}{256-1} = 0.023\,529\,4$$

为了得到x和y的实际值，将其十进制的值乘以0.023 529 4并减去3：

$$x = (138)_{10} \times 0.023\,529\,4 - 3 = 0.247\,058\,8$$

和

$$y = (59)_{10} \times 0.023\,529\,4 - 3 = -1.611\,764\,7$$

228

必要的时候还可以用其他的解码方法，例如格雷码（Caruana and Schaffer，1988）。

将解码后的 x 和 y 值作为数学函数的输入，遗传算法便会计算每个染色体的适应性。

为了找到“峰”函数的最大值，我们指定交叉概率为 0.7，突变概率为 0.001。前面提到过，在遗传算法中通常要指定代数。假设预期的代数为 100，即遗传算法会在 6 个染色体繁殖 100 代后停下来。

图 7.6a 显示了染色体在表面上的初始位置和“峰”函数的轮廓图。每个染色体用一个球表示。最初的种群由随机产生的个体组成，它们彼此不同且是异质的。但从第一代开始，交叉操作开始将最优的染色体的特征重组，种群开始向包含最高点的峰值处收敛，如图 7.6b 所示。直到最后一代，遗传算法用突变在峰值周围搜索，产生多样性。图 7.6c 显示了最终的染色体情况。但是种群汇聚在位于“峰”函数的局部最大值的染色体上。

但我们要找的是全局最大值。如何确保找到最优解决方案呢？遗传算法最严重的问题就是必须关注解的质量，尤其是是否找到最优解。一种方法就是比较用不同的突变率得到的结果。本例假设将突变比率增加为 0.01，并重新运行遗传算法。种群现在可能收敛到图 7.6d 所示的位置。但要确保得到稳定的结果，就要增加染色体种群的数量。

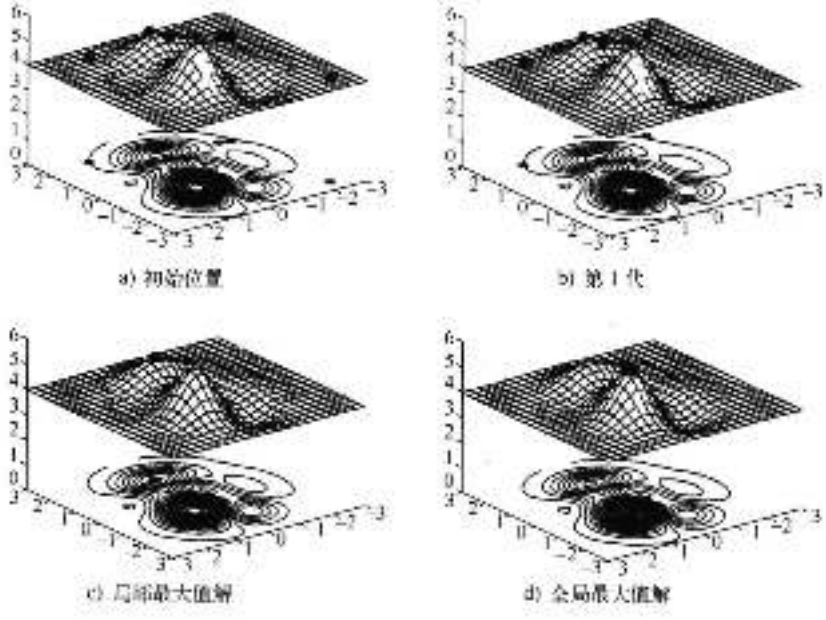


图 7.6 染色体在“峰”函数的表面轮廓图上的位置

图 7.6 所示的一类数学函数是用来展现遗传算法性能的一种方便的媒介。但是，现实中的问题对应的适应性函数可能无法简单地用图来表示。我们可以用性能图作替代。

什么是性能图

由于遗传算法是随机的，每次执行的性能都不同。因此，用曲线表示整个染色体种群的平均性能和用曲线表示种群中最优染色体的性能，都是检验遗传算法在给定遗传代数上的行为的有效方式。

图 7.7a 和图 7.7b 显示了 100 代中适应性函数的最优值和平均值。性能图的 x 轴为遗传的代数，并在运行中某个点处评估，y 轴为该点处适应性函数的值。

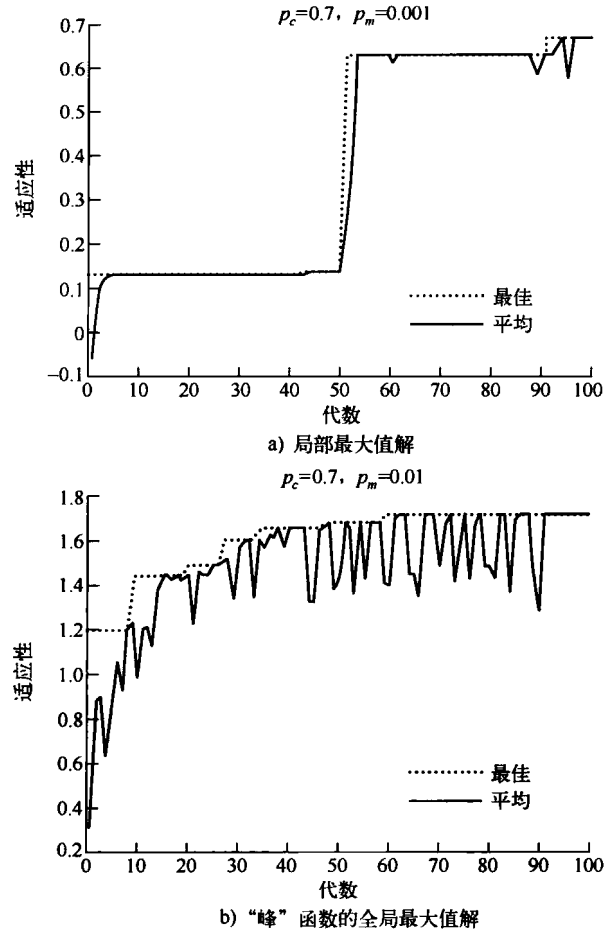


图 7.7 6 个染色体遗传 100 代的性能图

平均性能不稳定的原因是突变。突变操作允许遗传算法用随机的方式扩展范围。突变可能导致种群适应性得到显著改善，但更有可能降低适应性。为了在确保多样性的同时，减少变异带来的损害，我们可以增加染色体的数量。图 7.8 显示了 60 个染色体遗传 20 代的性能图。可以看

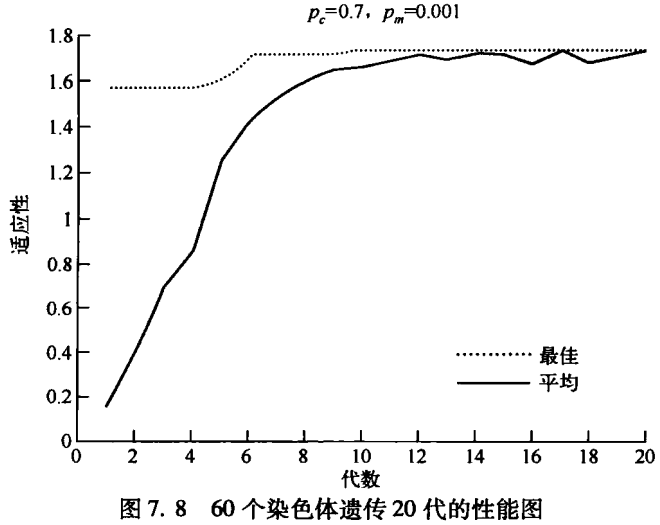


图 7.8 60 个染色体遗传 20 代的性能图

出，在运行开始阶段，平均曲线快速上升，在种群收敛到接近最优解时，上升变得缓慢，最后阶段几乎是平的。

7.4 遗传算法为什么可行

遗传算法有坚实的理论基础（Holland，1975；Goldberg，1989；Rawlins，1991；Whitley，1993）。这些理论基础构建在模式定理之上。

John Holland 引入了模式（shema）这个符号（Holland，1975），该符号来源于希腊语的“形式”（form）。模式是包含 0、1 和星号的位串，星号表示值可能是 0 也可能是 1。0 和 1 代表模式的固定位置，星号代表“通配符”。例如，模式 1 * * 0 表示一组 4 位的串的集合。集合中的每个串都以 1 开始，以 0 结束。这些串叫做模式的实例。

模式和染色体之间有什么关系

很简单，只要模式固定位置和染色体的相应位置匹配，染色体就可以和模式相匹配。例如，模式 H

1	*	*	0

和下面的一组 4 位染色体匹配：

1	1	1	0

1	1	0	0

1	0	1	0

1	0	0	0

每个染色体以 1 开始，以 0 结束。这些染色体叫做模式 H 的实例。

模式中定义的位（非星号）的个数称为秩。例如，模式 H 有两个定义的位，因此秩为 2。

简单地说，遗传算法运行时便会操作图式 schemata（schemata 是模式 schema 的复数形式）。如果遗传算法使用某种技术使繁殖的概率和染色体适应性成正比，那么基于模式定理（Holland，1975），就可以预测在遗传的下一代就会出现给定模式。换句话说，可以用给定模式的实例数量的增加或者减少来描述遗传算法的行为（Goldberg，1989）。

假设染色体的第 i 代中至少包含模式 H 的一个实例。令 $m_H(i)$ 为第 i 代中模式 H 实例的个数，$\hat{f}_H(i)$ 为这些实例的平均适应性。我们想要计算的是下一代中实例的个数 $m_H(i+1)$。由于繁殖的概率和染色体适应性成正比，因此，可以很容易地计算出下一代中染色体 x 的后代预期出现的数量：

229 ~ 232

$$m_x(i+1)=\frac{f_x(i)}{\hat{f}(i)} \tag{7.1}$$

其中，$f_x(i)$ 是染色体 x 的适应性，$\hat{f}(i)$ 是染色体第 i 代的平均适应性。

然后假设染色体 x 是模式 H 的实例，于是可以得到：

$$m_H(i+1)=\frac{\sum_{x=1}^{x=m_H(i)} f_x(i)}{\hat{f}(i)},x\in H \tag{7.2}$$

再定义：

$$\hat{f}_H(i)=\frac{\sum_{x=1}^{x=m_H(i)} f_x(i)}{m_H(i)}$$

就可以得到：

$$m_H(i+1)=\frac{\hat{f}_H(i)}{\hat{f}(i)}m_H(i) \tag{7.3}$$

因此，适应性高于平均值的模式在下一代染色体中出现的概率更高，而适应性低于平均值的模式在下一代染色体中出现的概率更低。

交叉和突变的效果怎么样

交叉和突变都可以创建或破坏模式的实例。这里我们仅考虑破坏的情况，即减少模式 H 的实例的个数的情况。首先将交叉操作导致的破坏量化。仅在至少一个后代也是模式实例的情况下模式免于被破坏。这种情况就是因为交叉操作没有发生在模式的定义长度上而造成的。

什么是模式的定义长度

模式中有定义的位的最远距离称为定义长度。例如，[*|*|*|*|1|0|1|1]的定义长度是3，[*|0|*|1|*|1|0|*]的定义长度是5，[1|*|*|*|*|*|*|0]的定义长度是7。

如果交叉发生在定义长度上，则模式 H 就会被破坏，并且会创建不是 H 的实例的后代（如
233 果两个相同的染色体交叉，即使交叉点在定义长度上，模式 H 也不会被破坏）。

因此，模式 H 在交叉后可以幸存下来的概率为：

$$P_H^{(c)}=1-p_c\left(\frac{l_d}{l-1}\right) \tag{7.4}$$

其中，p_c 是交叉概率，l 和 l_d 分别是模式 H 的长度和定义长度。

很明显，短模式交叉后幸存的概率要高于长模式的幸存的概率。

现在分析突变的破坏效果。假设 p_m 是模式 H 上任意位的突变概率，n 为模式 H 的秩。那么 $(1-p_m)$ 就是某位不会突变的概率，因此，模式 H 突变后幸存的概率为：

$$P_H^{(m)}=(1-p_m)^n \tag{7.5}$$

同样很明显，突变后短模式幸存的概率要大于长模式的幸存的概率。

下面考虑交叉和突变操作的破坏效果，修正公式（7.3）为：

$$m_H(i+1)=\frac{\hat{f}_H(i)}{\hat{f}(i)}m_H(i)\left[1-p_c\left(\frac{l_d}{l-1}\right)\right](1-p_m)^n \tag{7.6}$$

这个公式描述了模式从这一代到下一代的增长。这就是模式定理。因为公式（7.6）仅考虑了交叉和突变操作的破坏性，因此实际给出的是下一代中模式 H 的实例数量的下限。

尽管交叉被当做是遗传算法的一个主要优点，但是目前还没有足够的理论基础可以支持这一观点：仅仅是因为交叉操作能够将部分解决方法结合起来，遗传算法就优于其他搜索或优化算法。

遗传算法是一个功能强大的工具，但需要巧妙地使用。例如，将问题编码成位串可能会改变所研究问题的本质。换句话说，编码表达所代表的问题有可能不同于最初要解决的问题。

234 为了证明上面讨论的观点，下面一节将讨论一个应用遗传算法解决资源计划问题的例子。

7.5 案例研究：用遗传算法来维护调度

遗传算法最成功的应用领域之一是资源调度的问题。资源调度问题通常很复杂且难以解决。一般情况下都是结合使用搜索算法和启发式方法加以解决。

为什么调度问题这么难

首先调度问题属于 NP 完全问题。这种问题难于管理且不可能用组合搜索技术来解决。另

外，单独使用启发式方法也不能保证得到最优解。

其次，调度问题涉及有限资源的竞争，因此，有诸多限制而导致问题难于解决。遗传算法能够成功的关键在于能够定义一个适应性函数，这个函数可以整合所有的限制。

我们这里要讨论的是维护现代电力系统的调度。这个任务有诸多限制和不确定性，例如电力设备的故障，以及导致的被迫中止运转和获得备用件时的时间延迟。调度计划经常要在短时间内修订。人类专家经常手工处理这个维护计划，但不能保证产生的是最优或者接近最优的解决方案。

开发遗传算法的典型过程包含下面几个步骤：

1）确定问题，定义约束条件和最优标准。

2）用染色体来表示问题域。

3）定义适应性函数来评估染色体的性能。

4）构造遗传操作。

5）运行遗传算法，并调整参数。

步骤1：确定问题，定义约束条件和最优标准。

这是开发遗传算法最重要的步骤。如果这个步骤不正确、不完善，就不可能得到可行的计划。

电力系统的各个组成部分需要在它们整个生命周期中不间断地工作，因此需要定期维修。制订维护计划的目的就是找到在给定时间周期内（通常为一年）电力设备的耗损情况，确保电力系统处于最安全的状态。

电力系统的任何损耗都与安全性的丧失相关联。安全边界由系统的净储备定义。净储备定义为系统设备的总容量减去计划损耗的能量，再减去维护期间预测的最大负载量。例如，假设总容量为150 MW，周期内预期最大负载量为100 MW时，周期内计划维护一个20 MW的设备，那 235
么净储备就是30 MW。维护计划必须确保在任何维修期间有充足的净储备以安全地供应电力。

假设在4个等长的时间间隔内有7个电力设备。在时间间隔内预期的最大负载分别为80 MW、90 MW、65 MW和70 MW。表7.3为设备容量和维护需求。

表7.3 电力设备和维护需求

设备编号	设备容量（MW）	一年中维护设备所需的时间段数量
1	20	2
2	15	2
3	35	1
4	40	1
5	15	1
6	15	1
7	10	1

该问题的限制条件如下：

- 任何设备的维护从间隔的起始开始，在本间隔尾端或相邻间隔处结束。维护开始后不能异常中止，或者早于计划的时间而结束。
- 电力系统的净储备在任何时间间隔上必须大于或等于零。

最优标准是指在任何维护期间内净储备的值达到最大。

步骤2：用染色体来表示问题域。

调度问题在本质上是排序问题，它需要我们以一定的顺序列出任务的次序。一个完整的调度可能由一系列相互重叠的任务组成，但不是说所有的次序都合理，因为某些次序会违反系统的限制。我们的任务是将完整的调度表示为固定长度的染色体。

一个显而易见的编码方案是给每个设备制定一个二进制编号，染色体就是这些二进制编号的串。但是把设备按次序排列还不是调度。因为某些设备要同时维护，因此要把维护需要的时间放入调度中。因为，不是把设备按顺序列成串，而是应该为每个设备构建维护计划。设备计划可
236 以很容易地用4位字符串来表示，每一位就是维护的一个时间间隔。如果该设备的维护是在某个时间间隔内进行，相应的位就是1，否则为0。例如，串0100表示的计划是在第二个时间间隔内维护设备。因此，问题的完整维护计划可用28位的染色体来表示。

但是，交叉和突变操作可以很容易地创建这样的二进制字符串：某些设备可能被维护多次，而其他一些设备根本不被维护。另外，可能会使得维护的次数超出了真正所需的维护次数。

更好的方法是改变染色体的语法。前面讨论过，染色体是由基因作基本单元组成的集合。一般情况下，每个基因用一位来表示，并且不能被分成更小的元素。在本问题中，我们采用相同的概念，但是用4位来表示一个基因。也就是说，我们的染色体中，不可分割的最小单位为4位的串。这种表示方式保证了交叉和突变操作根据遗传算法的理论可以进行。接下来就是为每个设备产生基因池：

设备1：1100 0110 0011
设备2：1100 0110 0011
设备3：1000 0100 0010 0001
设备4：1000 0100 0010 0001
设备5：1000 0100 0010 0001
设备6：1000 0100 0010 0001
设备7：1000 0100 0010 0001

遗传算法通过从相应的池中随机选择基因来填满有7个基因的染色体，并以此来创建初始染色体种群。图7.9显示了一个染色体的样本。

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

图7.9 调度问题中的一个染色体

步骤3：定义适应性函数来评估染色体的性能。

染色体的评估是遗传算法中一个关键的部分，因为要根据它们的适应性来选择染色体进行匹配。适应性函数必须捕捉是什么使维护计划对用户而言是好或是坏。本例中，我们使用比较简
237 单的函数，关注每个时间间隔中的限制因素和净储备。

染色体的评估从每个时间间隔上计划维护的设备容量的总和开始，从图7.9我们可以得到：

时间间隔1：$0\times20+0\times15+0\times35+1\times40+0\times15+0\times15+1\times10=50$

时间间隔2：$1\times20+0\times15+0\times35+0\times40+1\times15+0\times15+0\times10=35$

时间间隔3：$1\times20+1\times15+0\times35+0\times40+0\times15+1\times15+0\times10=50$

时间间隔4：$0\times20+1\times15+1\times35+0\times40+0\times15+0\times15+0\times10=50$

然后将这些值从电力系统的总容量中减去（本例为150MW）：

时间间隔1：$150-50=100$

时间间隔2：150 − 35 = 115

时间间隔3：150 − 50 = 100

时间间隔4：150 − 50 = 100

最后，减去每个时间间隔内预期的最大负载，便可得到各自的净储备：

时间间隔1：100 − 80 = 20

时间间隔2：115 − 90 = 25

时间间隔3：100 − 65 = 35

时间间隔4：100 − 70 = 30

由于结果都为正数，这个染色体没有违反任何限制，因此是一个合理的计划。染色体的适应性由净储备的最低值确定，本例是20。

但是，如果任一时间间隔上的净储备是负值，那么计划就不合适，适应性函数要归为0。

在运行初期，随机创建的初始群体可能全部为不合适的调度。在这种情况下，染色体适应性取值保持不变，选择过程要按照实际适应性的值来进行。

步骤4：构造遗传操作。

构造遗传操作具有很大挑战性，我们必须通过实验来设置交叉和突变的取值，以便使其正确地工作。染色体应该以对问题来说合法的方式进行断裂。因为我们已经改变了染色体的语法，所以可用经典的方式来进行遗传操作。染色体中的每个基因是一个4位的不可分割的串，这些串由某个设备的可能的维护计划组成。因此，任何基因随机的突变或来自亲代染色体的许多基因 238
的重组都可能导致设备维护计划的改变，但不能创造“非自然”的染色体。

图7.10a为运行遗传算法时交叉操作的例子。在用竖线表示的随机选择的位置截断亲代，并在这之后交换亲代基因，从而得到子代。图7.10b为突变的例子。突变运算在染色体中随机选择一个4位的基因，并用在相应池中随机选择的基因进行替代。图7.10b的例子显示，染色体在第三个基因处发生突变，在设备3的基因池中选择基因[0][0][0][1]来代替原来的基因。

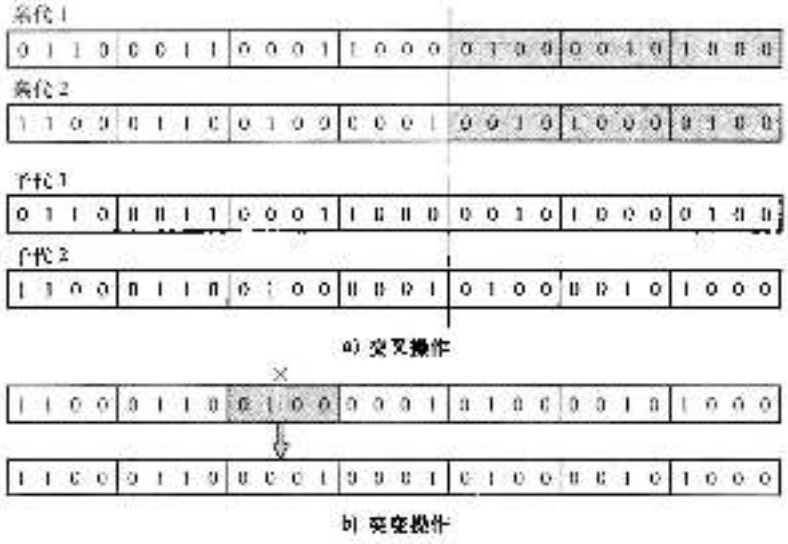


图7.10 调度问题的遗传操作

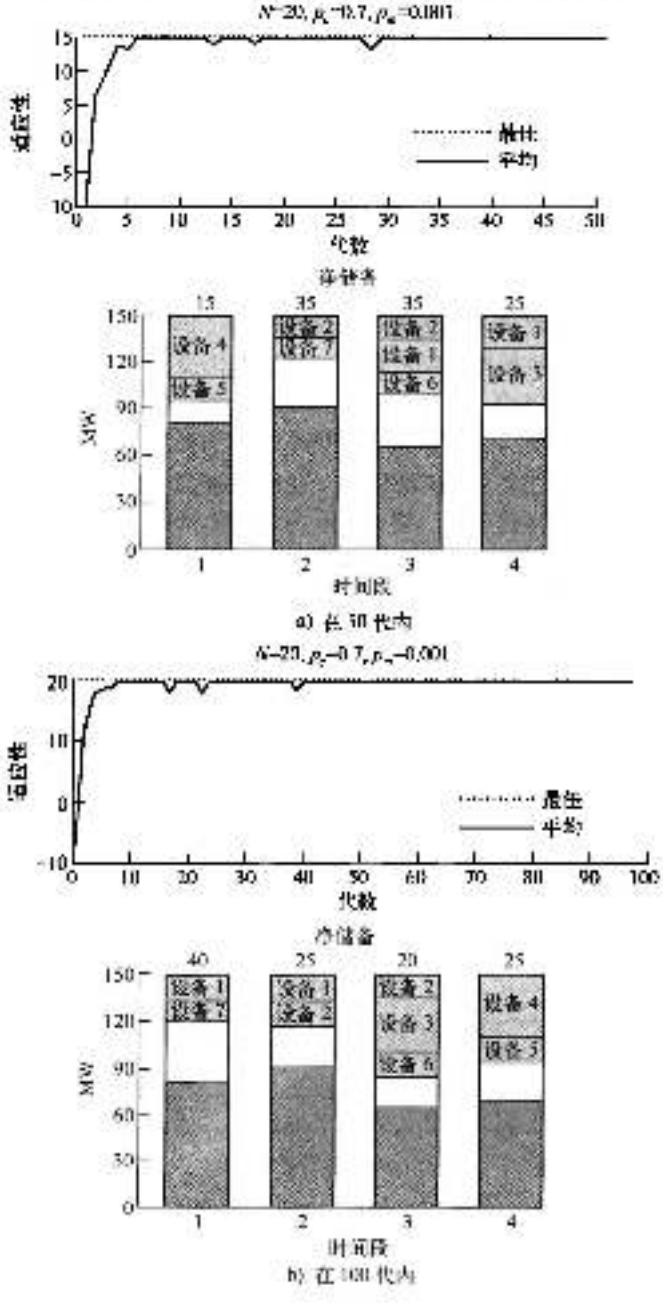
步骤5：运行遗传算法并调整参数。

现在运行遗传算法。首先必须选择用于运行的种群大小和遗传代数。根据常识，种群越大结果越好，但是运行速度会相对较慢。但实际上，种群大小的选择取决于要解决的问题，特别是问

题的编码方案（Goldberg，1989）。遗传算法只可以运行有限的代数来获得解。我们可以选择一个非常大的种群只运行一次，或者选择较小的种群而运行多次。任何情况下，都只有实验能给我们答案。

239

图7.11a为20个染色体50代产生的性能图和最优计划。可以看到，最优调度的净储备的最小值为15 MW。将遗传代数增加到100，比较一下结果（如图7.11b所示）。这次得到的最优调度的净储备的最小值是20 MW。但是，在两种情况中，最佳的个体都出现在第1代中。遗传代数的增加也不会影响最终的解决方案。这就提示我们应该增加种群的大小。

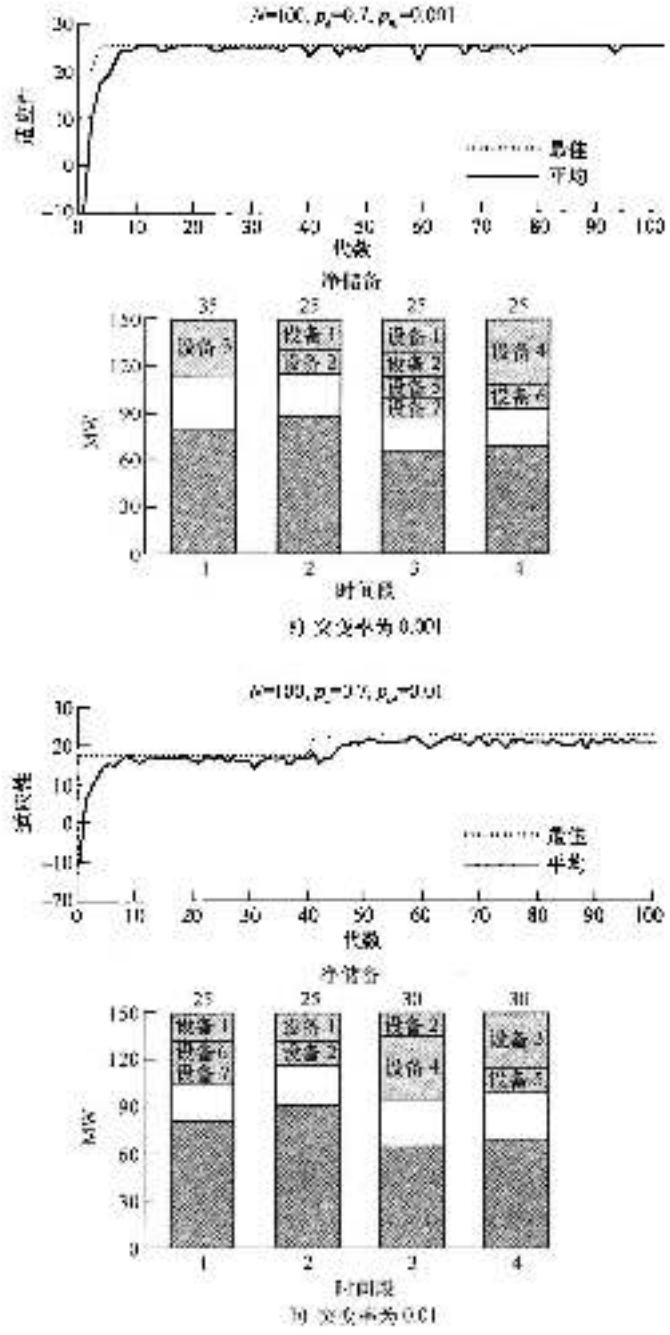


a) 在50代内

b) 在100代内

图7.11 有20个染色体的种群性能图和最优调度计划

图 7.12a 为进化到 100 代的性能图和这种情况下的最优调度计划。冷储备的最小值增加到了 25 MW。为了得到最好调度，我们必须比较在不同突变率下的结果。因此，将突变率提高到 0.01，并再次运行遗传算法，图 7.12b 展示了这种结果。网络最小的冷储备依然是 25 MW。现在我们可以很自信地说最优解已经找到了。



a) 突变率为 0.001

b) 突变率为 0.01

图 7.12 有 100 个染色体的种群性能图和最佳调度计划

7.6 进化策略

另一个模拟自然进化的方法是20世纪60年代初在德国提出的。和遗传算法不同，这种称为进化策略的方法是用于解决工程技术优化问题的。

1963年，柏林工业大学的两个学生 Ingo Rechenberg 和 Hans - Paul Schwefel 致力于研究流体的最佳形状。在工作中，他们使用了流体工程学院的风洞。由于这是一个艰苦且需要依靠直觉的工作，他们决定按照自然突变的例子来随机改变形状的参数，结果便产生了进化策略（Rechenberg，1965；Schwefel，1981）。

进化策略是可替代工程师直觉的一种方法。直到最近，当没有分析对象的函数可用，传统的优化方法不存在，工程师必须依靠他们的直觉时，进化策略才被用于技术优化问题中。

和遗传算法不同，进化策略仅用到突变操作。

如何实现进化策略

进化策略最简单的形式是（1 + 1）进化策略。每一代中，采用正态分布的变异使单个亲代产生一个子代。（1 + 1）进化策略的实现方法如下：

240
~
242

步骤1：选择表示问题的 N 个参数，然后确定每个参数可行的范围。

$$\{x_{1min}, x_{1max}\}, \{x_{2min}, x_{2max}\}, \cdots, \{x_{Nmin}, x_{Nmax}\}$$

定义每个参数的标准差和需要优化的函数。

步骤2：在每个参数各自可行的范围内随机选择初始值。这些参数值的集合就是亲代参数的初始种群。

$$x_1, x_2, \cdots, x_N$$

步骤3：计算和亲代参数相关联的解。

$$X = f(x_1, x_2, \cdots, x_N)$$

步骤4：通过给每个参数加上均值为0，方差为 δ 的正态分布随机变量 a，获得一个新的子代参数。

$$x'_i = x_i + a(0, \delta), \quad i = 1, 2, \cdots, N \qquad (7.7)$$

均值为0的正态分布的突变反映了进化的自然过程，即较小的变化发生的概率大于较大的变化发生的概率。

步骤5：计算和后代参数相关联的解。

$$X' = f(x'_1, x'_2, \cdots, x'_N)$$

步骤6：比较子代参数对应的解和亲代参数对应的解。如果子代的解比较好，就用子代代替亲代。否则，保留亲代参数。

步骤7：回到步骤4，重复这个过程，直到得到满意的解，或者达到了制定的遗传代数为止。

（1 + 1）进化策略的流程图如图7.13所示。

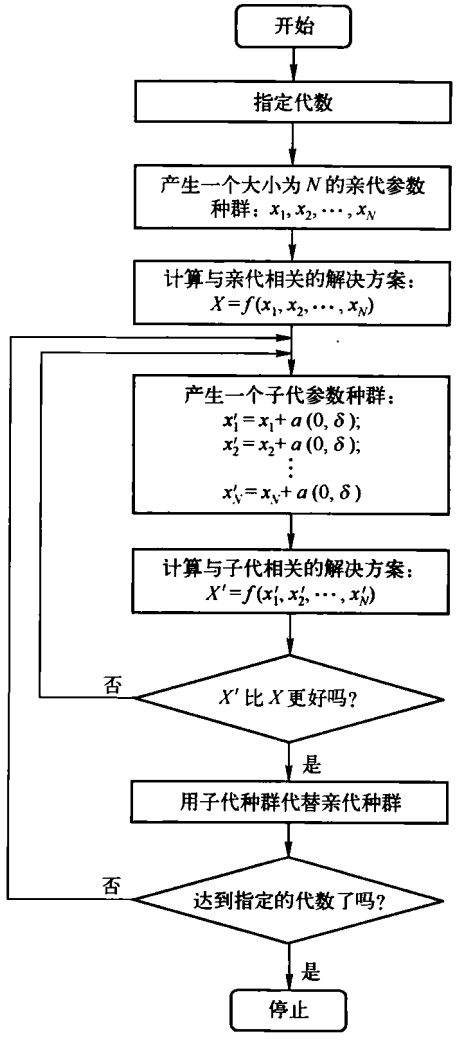


图7.13 （1 + 1）进化策略的流程图

产生新的解时，为什么要同时改变所有的参数

进化策略反映了染色体的本质。实际上，单个的基因可能会同时影响到生物体的几个特征。另一方面，生物体的单个特征也可能由几个基因同时确定。自然选择作用在一组基因而不是单个基因上。

进化策略可以解决很多受限和不受限的非线性优化问题，且由进化策略得到的结果通常比很多传统、高度复杂的非线性优化技术得到的结果要好（Schwefel，1995）。通过实验还验证了进化策略的最简单版本，即使用单个亲代得到单个子代的方法来进行搜索最为有效。 243

遗传算法和进化策略有何区别

遗传算法和进化策略的本质区别在于前者同时使用交叉操作和突变操作，而后者仅使用突变操作。另外，在使用进化策略时，不需要用编码的形式来表示问题。

哪种方法更好

进化策略使用纯粹的数值优化计算过程，这和蒙特·卡洛搜索类似。遗传算法是更一般的应用，但应用遗传算法最困难的部分是对问题进行编码。一般来说，要回答哪种方法更好，需要进行实验，这是完全取决于应用的。

7.7 遗传编程

计算机科学的一个核心问题就是如何能让计算机在没有明确编程的情况下知道如何解决问题。遗传编程提供了解决这个问题的方法，即通过自然选择的方法来使计算机程序进化。实际上遗传编程是传统遗传算法的扩展，但遗传编程的目的不仅仅是用位串来表示问题，而是要编写解决问题的代码。换句话说，遗传编程创建作为解决方案的计算机程序，而遗传算法创建表示解决方案的二进制串。

遗传编程是进化计算领域的最新发展成果。20 世纪 90 年代，John Koza 对遗传编程的发展起了很大的作用（Koza，1992；1994）。

遗传编程如何工作

根据 Koza 的理论，遗传编程搜索可能的计算机程序空间，来获得一个非常适合解决问题的计算机程序（Koza，1992）。

任何的计算机程序都是作用到值（参数）的一系列操作（函数）。但不同的编程语言可能有不同的语句、操作及语法限制。由于遗传编程是用遗传操作来操纵程序，因此，编程语言应该允许计算机程序可以像数据一样被操作，并且新创建的数据可以作为程序执行。基于这些原因，LISP 被选作遗传编程的主要语言（Koza，1992）。

什么是 LISP

LISP，或者说是*列表处理语言*（List Processor），是最古老的高级编程语言之一（FORTRAN 仅比 LISP 早出现两年），它由 John McCarthy 在 20 世纪 50 年代末完成，是人工智能的标准语言之一。

LISP 有高度面向符号的结构。其基本数据结构是*原子*和*表*。原子是 LISP 语法中不可分割的最小元素。数字 21，符号 X 和字符串“This is a string”都是 LISP 原子的例子。表是原子和/或其他表组成的对象。LISP 表可以写为圆括号中的项目的有序集合。例如： 244 / 245

$$(-(*\,AB)\,C)$$

此表要求调用减函数（-）处理两个自变量，也就是表（* AB）和原子 C。首先，LISP 对原子 A 和 B 使用乘函数（*），得到列表（* AB）的结果，然后用减函数（-）计算整个（-(* AB) C)的结果。

原子和表都称作*符号表达式*或 S *表达式*。在 LISP 中，所有的数据和结构都是 S 表达式。因此，LISP 可以像操作数据一样操作程序。也就是说，LISP 程序可以修改它们自己，甚至编写出其他的 LISP 程序。LISP 的这个重要的特点对于遗传编程而言非常有吸引力。

任何 LISP 的 S 表达式都可以表达成一棵用结点标记的有根的、具有有序分支的树。图 7.14 为 S 表达式（ -(* AB) C）的树。这棵树有 5 个结点，每个结点表示一个函数或者终端。树的两个内部结点用（ - ）或（ * ）标识。注意，树的根是出现在 S 表达式中左端圆括号内的函数。这棵树的 3 个外部的结点也叫做叶结点，它们用终端 A、B 和 C 标识。在图形化表示中，分支是有序的，因为函数的参数顺序直接影响最终结果。

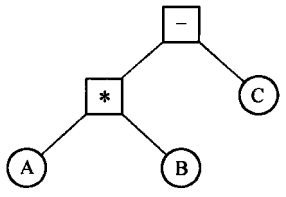


图 7.14 LISP S 表达式（ -(* AB) C) 的图形化表示

如何使用遗传编程来解决问题

在使用遗传编程来解决问题之前，必须先执行 5 个预备步骤（Koza，1994）：

1）确定终端集合。

2）选择基本函数集。

3）定义适应性函数。

4）确定控制运行的参数。

5）选择指定运行结果的方法。

可以用勾股定理来说明这些预备步骤，并证明遗传编程的潜力。勾股定理说明，直角三角形的斜边 c 和两个直角边 a、b 有如下关系：

$$c = \sqrt{a^2 + b^2}$$

遗传编程的目的是找到和这个函数匹配的程序。为了度量还未被发现的计算机程序的性能，我们在这里使用不同的*适应性案例*。勾股定理的适应性案例用表 7.4 所示的直角三角形表示。这些适应性案例是在变量 a 和 b 的取值范围内随机选择的。

表 7.4 勾股定理的适应性案例

边 a	边 b	斜边 c	边 a	边 b	斜边 c
3	5	5.830 952	12	10	15.620 499
8	14	16.124 515	21	6	21.840 330
18	2	18.110 770	7	4	8.062 258
32	11	33.837 849	16	24	28.844 410
4	3	5.000 000	2	9	9.219 545

步骤 1：确定终端集合。

找到与计算机程序的输入相对应的终端。本例中有两个输入，a 和 b。

步骤 2：选择基本函数集。

函数可以用标准算数操作、标准编程操作、标准数学函数、逻辑函数或特定领域的函数来表示。本例中使用标准算数函数 + 、 - 、 * 、/，以及一个数学函数 sqrt。

终端和基本函数一起构成了构造块，遗传编程利用这些构造块构建解决问题的计算机程序。

步骤 3：定义适应性函数。

适应性函数用来评估某个计算机程序解决问题的能力。适应性函数的选择取决于要解决的问题。每个问题的适应性函数可能有很大不同。本例中，计算机程序的适应性可以通过程序产生的实际结果和适应性案例给出的结果间的误差来衡量。一般情况下，只有一个案例时不测量误差，而是要计算一组适应性案例的绝对误差的总和。总和越接近于 0，计算机程序就越好。

步骤4：确定控制运行的参数。

为了控制运行，遗传编程使用的主要参数和遗传算法一样。它包含种群大小和最大遗传代数。

步骤5：选择指定运行结果的方法。

通常在遗传编程中指定目前最好的遗传程序的结果作为运行结果。

一旦这5个步骤执行完毕，就可以开始运行了。遗传编程的运行从随机选择产生的计算机程序初始种群开始。每个程序由 +、−、*、/和 sqrt 以及终端结点 a、b 组成。

在最初的种群中，所有计算及程序的适应性都很差，但某些个体的适应性要比其他个体好。就像适应性较强的染色体被选中进行繁殖的概率要更高一样，适应性较好的计算机程序通过复制自己进入下一代的概率也更高。

交叉操作可以操作计算机程序吗

在遗传编程中，交叉操作针对的是两个基于适应性而被选择的计算机程序。这些程序有不同的尺寸和形状。两个子代程序是两个亲代程序任意部分的组合。例如，有如下两个 LISP 的 S 表达式：

$$(/(-(sqrt(+(*\ a\ a)(-a\ b)))a)(*\ a\ b))$$

它相当于：

$$\frac{\sqrt{a^2+(a-b)}-a}{ab}$$

而

$$(+(-(sqrt(-(*\ b\ b)a))b)(sqrt(/\ a\ b)))$$

相当于

$$\left(\sqrt{b^2-a}-b\right)+\sqrt{\frac{a}{b}}$$

这两个S表达式都可以用带有有序分支的、用结点标记的有根树表示，如图7.15a所示。树的内部结点和函数对应，外部结点和终端对应。

内部或外部的点都可以作为交叉点。假设第一个亲代的交叉点在函数（*）处，第二个 248
亲代的交叉点是函数 sqrt。那么，就可以得到根在刚才的交叉点上的两个交叉片段，如图7.15a所示。交叉操作通过交换两个亲代的片段得到两个子代。因此，第一个子代是第一个亲代在交叉点上插入了第二个亲代的片段来创建的。同样，第二个子代是第二个亲代在交叉点上插入了第一个亲代的片段创建的。两个子代来自于两个亲代的交叉，结果如图7.15b所示，这两个子代是：

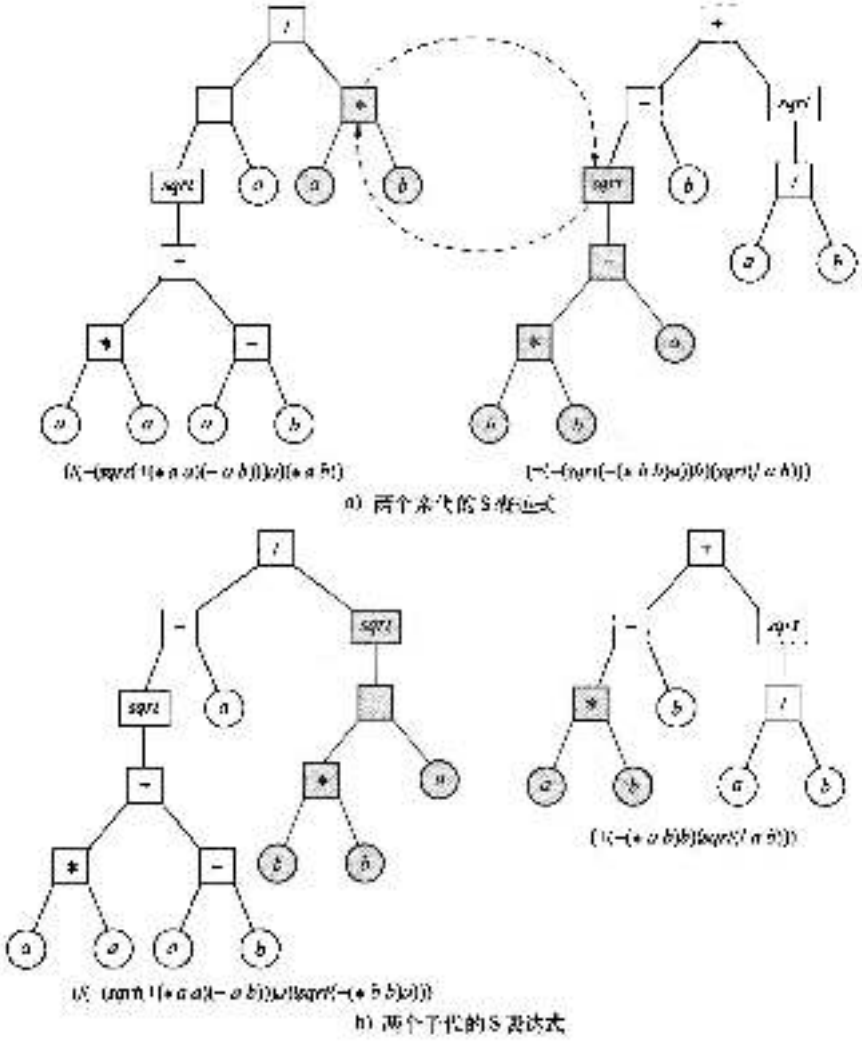
$$\frac{\sqrt{a^2+(a-b)}-a}{\sqrt{b^2-a}} \quad 和 \quad (ab-b)+\sqrt{\frac{a}{b}}$$

不管交叉点怎么选择，交叉操作都能产生有效的子代计算机程序。

遗传编程中要用到突变操作吗

突变操作可以任意改变 LISP 的 S 表达式中的函数或终端。在突变中，函数仅能用函数取代，终端也仅能用终端取代。图7.16解释了遗传编程中突变的基本概念。

总之，遗传编程通过执行下述步骤来创建计算机程序（Koza，1994）：



a) 两个亲代的S表达式

b) 两个子代的S表达式

图 7.15 遗传编程中的交叉

步骤1：指定运行的最大遗传代数以及克隆、交叉和突变的概率。注意，克隆、交叉和突变三者概率之和必须为1。

步骤2：产生长度为 N、由随机选择的终端和函数组成的计算机程序的初始种群

步骤3：执行种群中的每个计算机程序，用合适的适应性函数计算其适应性，指定最好的个体作为运行的结果。

步骤4：用指定的概率选择遗传操作，以执行克隆、交叉和突变。

步骤5：如果选择的是克隆操作，则从现有种群中选择一个计算机程序，复制该程序后将其放入下一代种群中；如果选择的是交叉操作，则从现有种群中选择一对计算机程序，创建一对后代程序，放入下一代种群中；如果选择的是突变操作，则从现有种群中选择一个计算机程序，执行突变操作并将突变的结果放入下一代种群中。所有的程序都按照其适应性的概率进行选择（适应性越高，选中的概率就越大）。

249
250

步骤6：重复执行步骤4，直到新种群的计算机程序数量和初始种群一样多（等于 N）为止。

步骤7：用新的（子代）种群取代当前的（子代）种群。

步骤8：回到步骤3，重复执行，直到满足终止条件为止。

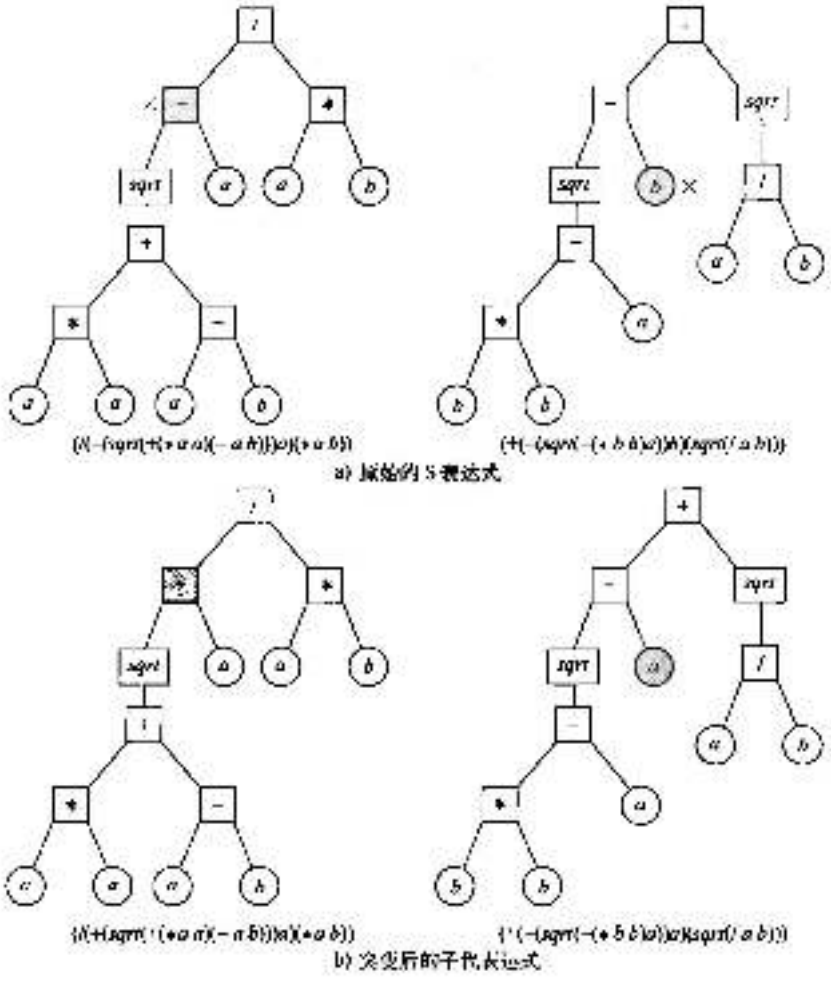


图 7.16 遗传编程中的突变

图 7.17 是以上步骤的遗传编程的流程表示。

251

现在回到勾股定理。图 7.18 显示了 500 个计算机程序的种群中最好的 S 表达式的适应性。可以看到，在随机产生的初始群体中，即使最好的 S 表达式也有很差的适应性。但适应性提高得很快，在第 4 代中就产生了正确的 S 表达式。这个简单的例子证明了遗传编程为计算机程序的进化提供了通用和便宜的方法。

在勾股定理的例子中，使用了 LISP 的 S 表达式，但是没有理由将遗传编程限制在 LISP S 表达式中。其实在 C、C++、Pascal、FORTRAN、Mathematica、Smalltalk 和其他的编程语言中，都可以实现 S 表达式并且更广泛地使用。

和遗传算法相比，遗传编程的优点是什么

遗传编程和遗传算法使用相同的进化方法。但是遗传编程不再用位串来表示编码方法，而是用完整的计算机程序解决具体的问题。遗传算法最基本的困难在于问题的表达，也就是固定长度的编码表达效果不佳，限制了遗传算法的能力，甚至导致错误的解。

固定长度的编码相当不实际。由于不能提供长度的动态变化，这样的编码会经常导致相当大的冗余，从而降低了遗传搜索的效率。相反，遗传编程使用长度可变的高层组件，其大小和复杂性可以在繁殖中改变。遗传编程在很多不同场合中都可以使用（Koza，1994），并且应用范围还可以更广。

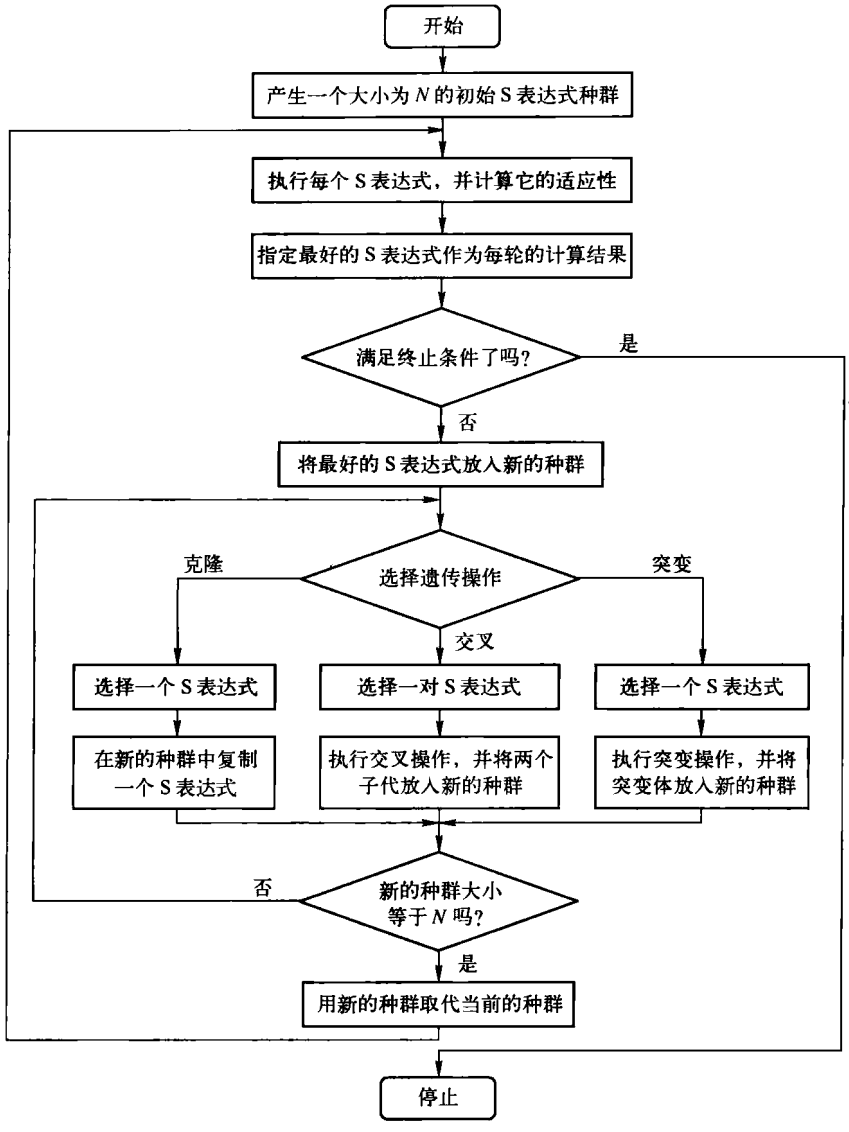


图 7.17 遗传编程的流程图

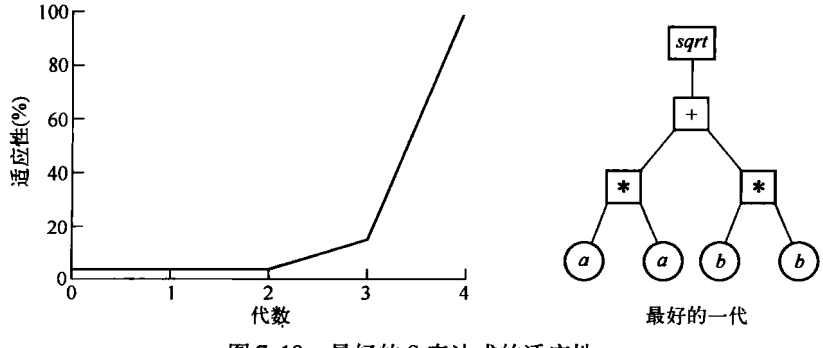


图 7.18 最好的 S 表达式的适应性

还有什么困难

252 ~ 253

尽管有许多成功的应用，但没有任何证据表明遗传编程可以扩展到那些需要大量计算机程序的复杂问题中。即使能够扩展，也需要大量的计算机运行时间。

7.8 小结

本章主要介绍进化计算，其中包括遗传算法、进化策略和遗传编程。我们首先介绍了开发遗传算法的主要步骤，讨论了遗传算法的工作机制，并通过实际应用解释了遗传算法的理论。接下来介绍了进化策略的基本概念，比较了进化策略和遗传算法之间的不同。最后介绍遗传编程及其在解决实际问题中的应用。

本章的主要内容有：

- 以自然选择和遗传的计算模型为基础的，获取人工智能的方法，称作进化计算。进化计算包含了遗传算法、进化策略和遗传编程。
- 进化计算所有方法的工作方式大致如下：创建包含个体的种群，计算适应性，通过遗传操作产生新的种群，并将该过程重复一定的次数。
- 遗传算法由 John Holland 在 20 世纪 70 年代初创建。Holland 的遗传算法是将一代人工“染色体”移到下一代的一系列步骤，它使用的是“自然”选择和交叉、突变等遗传技术。每个染色体由几个基因组成，每个基因用 0 或 1 表示。
- 遗传算法使用单个染色体适应性值来实现繁殖。进行繁殖时，交叉操作交换两个单独染色体的一部分，突变操作则改变染色体上随机选中的某个基因的值。在几次成功繁殖之后，适应性低的染色体就会灭绝，而适应性高的染色体则会在各代中占主导地位。
- 遗传算法的工作原理是，发现和重组好的图式——即候选方案中好的组件。遗传算法不需要知道问题领域的任何知识，但需要适应性函数评估解决方案的适应性。
- 用遗传算法来解决问题，涉及定义约束条件和最优标准，将问题解决方案编码为染色体、定义适应性函数来评估染色体的性能，创建合适的交叉和突变操作。
- 遗传算法是一个功能强大的工具。但是把问题用位串来表示可能会改变问题的本质。这样通常很危险，因为这样做会导致问题变得和我们想解决的问题不同。 254
- 进化策略是代替工程师直觉的一种方法。Ingo Rechenberg 和 Hans-Paul Schwefel 在 20 世纪 60 年代初对其进行了研究。进化策略在没有分析对象函数、没有传统的优化方法且仅仅依赖于工程师直觉的时候使用。
- 进化策略是纯粹的数值优化过程，它和蒙特·卡洛搜索方法类似。与遗传算法不同，进化策略仅使用突变操作。另外，它也不需要把问题用编码的方式表示。
- 遗传编程是进化计算领域的最新发展成果。20 世纪 90 年代，John Koza 对其发展起了很大的作用。遗传编程使用和遗传算法相同的进化机制。但是遗传编程的目的不再是用位串来表示问题，而是要编写解决问题的完整的计算机程序。
- 用遗传编程来解决问题包括确定自变量集合、选择函数集合、定义评估计算机程序性能的适应性函数，以及选择指定运行结果的方法。
- 由于遗传编程用遗传操作来操作程序，编程语言应该允许计算机程序可以像数据一样进行操作，并且新创建的数据应该像程序一样执行。基于这些原因，LISP 被选作遗传编程的主要语言。

复习题

1. 为什么遗传算法被称作遗传？谁是遗传算法“之父”？
2. 遗传算法的主要步骤是什么？画出执行这些步骤的流程图。它的终止条件是什么？
3. 什么是轮盘选择技术？它如何工作？举例说明。
4. 交叉操作如何工作？用固定长度位串为例说明，再用 LISP S 表达式为例说明。

5. 什么是突变？为什么需要突变？突变操作如何工作？用固定长度位串为例说明，再用 LISP S 表达式为例说明。
6. 遗传算法为什么可行？什么是模式？举例说明模式及其实例。解释模式和染色体之间的关系。什么是模式定理？
7. 用一个真实的问题来描述开发遗传算法的过程。遗传算法的难点在哪里？
255 8. 什么是进化策略？如何实现它？进化策略和遗传算法间有什么差别？
9. 画出（1 + 1）进化策略的流程图。为什么在产生新的解时要同时改变所有的参数？
10. 什么是遗传编程？如何使用遗传编程？为什么 LISP 是遗传编程的主要语言。
11. 什么是 LISP S 表达式？举例说明，使用具有序分支的带标记点的有根树来表示 S 表达式。在树中标出终端和函数。
12. 遗传编程的主要步骤是什么？画出执行步骤的流程图。遗传编程的优点是什么？

参考文献

Atmar, W. (1994). Notes on the simulation of evolution, *IEEE Transactions on Neural Networks*, 5(1), 130–148.

Caruana, R.A. and Schaffer, J.D. (1988). Representation and hidden bias: gray vs. binary coding for genetic algorithms, *Proceedings of the Fifth International Conference on Machine Learning*, J. Laird, ed., Morgan Kaufmann, San Mateo, CA.

Conner, J.K. and Hartl, L.D. (2004). *A Primer of Ecological Genetics*. Sinauer Associates, Sunderland, MA.

Davis, L. (1991). *Handbook on Genetic Algorithms*. Van Nostrand Reinhold, New York.

Fogel, L.J., Owens, A.J. and Walsh, M.J. (1966). *Artificial Intelligence Through Simulated Evolution*. Morgan Kaufmann, Los Altos, CA.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimisation and Machine Learning*. Addison-Wesley, Reading, MA.

Goldberg, D. and Sastry, K. (2010). *Genetic Algorithms: The Design of Innovation*, 2nd edn. Springer-Verlag, Berlin.

Gould, J.L. and Keeton, W.T. (1996). *Biological Science*, 6th edn. R.S. Means, New York.

Hoffman, A. (1989). *Arguments on Evolution: A Paleontologist's Perspective*. Oxford University Press, New York.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

Koza, J.R. (1992). *Genetic Programming: On the Programming of the Computers by Means of Natural Selection*. MIT Press, Cambridge, MA.

Koza, J.R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA.

Mayr, E. (1988). *Towards a New Philosophy of Biology: Observations of an Evolutionist*. Belknap Press, Cambridge, MA.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolutionary Programs*, 3rd edn. Springer-Verlag, New York.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA.

Rawlins, G. (1991). *Foundations of Genetic Algorithms*. Morgan Kaufmann, San Francisco.

Rechenberg, I. (1965). *Cybernetic Solution Path of an Experimental Problem*. Ministry of Aviation, Royal Aircraft Establishment, Library Translation No. 1122, August.

Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models*. John Wiley, Chichester.

256 Schwefel, H.-P. (1995). *Evolution and Optimum Seeking*. John Wiley, New York.

Turing, A.M. (1950). Computing machinery and intelligence, *Mind*, 59, 433–460.

Whitley, L.D. (1993). *Foundations of Genetic Algorithms 2*. Morgan Kaufmann, San Francisco.

Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding, and selection in evolution, *Proceedings of the 6th International Congress on Genetics*, Ithaca, NY, vol. 1, pp. 356–366.

参考书目

Affenzeller, M., Winkler, S., Wagner, S. and Beham, A. (2009). *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*. Chapman & Hall/CRC Press, Boca Raton, FL.

Arnold, D.V. and Beyer, H.-G. (2002). *Noisy Optimization with Evolution Strategies*. Kluwer Academic Publishers, Boston, MA.

Banzhaf, W., Harding, S., Langdon, W.B. and Wilson, G. (2008). Accelerating genetic programming through graphics processing units, *Genetic Programming Theory and Practice VI*, R. Riolo, T. Soule, and B. Worzel, eds, Springer-Verlag, Berlin, pp. 1–19.

Beyer, H.-G. (2001). *The Theory of Evolution Strategies*. Springer-Verlag, Heidelberg.

Brameier, M. and Banzhaf, W. (2007). *Linear Genetic Programming*, Springer-Verlag, New York.

Cantu-Paz, E. (2000). *Designing Efficient Parallel Genetic Algorithms*. Kluwer Academic Publishers, Boston, MA.

Cevallos, F. (2009). *A Genetic Algorithm Approach to Bus Transfers Synchronization: Minimizing Transfer Times in a Public Transit System*. VDM-Verlag, Saarbrücken.

Christian, J. (2001). *Illustrating Evolutionary Computation with Mathematica*. Morgan Kaufmann, San Francisco.

Coello Coello, C.A., Lamont, G.B. and van Veldhuizen, D.A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd edn. Springer-Verlag, Berlin.

Crosby, J.L. (1973). *Computer Simulation in Genetics*. John Wiley, London.

Eiben, A.E. and Smith, J.E. (2003). *Introduction to Evolutionary Computing*. Springer-Verlag, Berlin.

Floreano, D. and Mattiussi, C. (2008). *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. MIT Press, Cambridge, MA.

Fogel, D.B. (2005). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, 3rd edn. Wiley–IEEE Press, Hoboken, NJ.

Ghanea-Hercock, R. (2003). *Applied Evolutionary Algorithms in Java*. Springer-Verlag, New York.

Haupt, R.L. and Haupt, S.E. (1998). *Practical Genetic Algorithms*, 2nd edn. John Wiley, New York.

Haupt, R.L. and Werner, D.H. (2007). *Genetic Algorithms in Electromagnetics*. Wiley–IEEE Press, Hoboken, NJ.

Haupt, S.E., Pasini, A. and Marzban, C. (2009). *Artificial Intelligence Methods in the Environmental Sciences*. Springer Science+Business Media, Berlin.

Jamshidi, M., Coelho, L.S., Krohling, R.A. and Fleming, P.J. (2003). *Robust Control Systems with Genetic Algorithms*. CRC Press, London.

Kolhe, M. (2008). *Wind Energy Forecasting by Using Artificial Neural Network – Genetic Algorithm*, VDM-Verlag, Saarbrücken.

Koza, J.R., Bennett III, F.H., Andre, D. and Keane, M.A. (1999). *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann, San Francisco. 257

Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J. and Lanza, G. (2005). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Springer-Verlag, Berlin.

Langdon, W.B. (1998). *Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming!* Kluwer Academic Publishers, Amsterdam.

Langdon, W.B. and Poli, R. (2010). *Foundations of Genetic Programming*. Springer-Verlag, Berlin.

Lowen, R. (2008). *Foundations of Generic Optimization Applications of Fuzzy Control, Genetic Algorithms and Neural Networks*. Springer-Verlag, Berlin.

Pappa, G.L. and Freitas, A. (2009). *Automating the Design of Data Mining Algorithms: An Evolutionary Computation Approach*, Springer-Verlag, Berlin.

Poli, R., Langdon, W.B. and McPhee, N.F. (2008). *A Field Guide to Genetic Programming*. Lulu Enterprises Inc.

Sivanandam, S.N. and Deepa, S.N. (2009). *Introduction to Genetic Algorithms*. Springer-Verlag, Berlin.

Steeb, W.-H. (2008). *The Nonlinear Workbook: Chaos, Fractals, Cellular Automata, Neural Networks, Genetic Algorithms, Gene Expression Programming, Support Vector Machine, Wavelets, Hidden Markov Models, Fuzzy Logic with C++, Java and SymbolicC++ Programs*, 4th edn. World Scientific, Singapore. 258

第8章
Artificial Intelligence, 3E

混合智能系统

本章主要介绍专家系统、模糊逻辑、神经网络和进化计算的结合，并讨论混合智能系统的出现。

8.1 概述

前面几章介绍了几种智能技术，包括概率推理、模糊逻辑、神经网络和进化计算。我们讨论了这几种技术的优点和缺点，并注意到在真实世界的应用中，不仅需要获取不同来源的知识，而且要结合不同的智能技术。对这种结合的需求导致了混合智能系统的出现。

混合智能系统是结合了至少两种智能技术的系统。例如，在混合神经－模糊系统中结合了神经网络和模糊系统。

概率推理、模糊逻辑、神经网络和进化计算的结合构成了软计算（Soft Computing，SC）的核心，软计算是一种在不确定和不精确的环境中建立能够进行推理和学习的混合智能系统的方法。

软计算的潜力首先是由模糊逻辑之父 Lotfi Zadeh 实现的。1991 年3 月，他创建了伯克利软计算倡议组织（Berkeley Initiative in Soft Computing）。该组织包括学生、教授、企业和政府组织的雇员，还有其他对软计算感兴趣的个人。该组织的快速成长表明，软计算对科学技术的影响在未来几年会快速发展。

“软”计算是什么意思

传统的计算或“硬”计算使用清晰的取值或数值，而软计算处理的是软取值或模糊集。软计算能够以反映人类思维的方式处理不确定、不精确和不完整的信息。在真实世界中，人类通常使用由单词而不是数字表示的软数据。人类的感觉器官处理软信息，大脑在不确定和不精确的环境中使用软联想和软推理。人类具有不使用数字进行推理和决策的非凡能力。人类使用单词和软计算来试图在制定决策时对单词的感觉进行建模。

可以用单词来成功地解决复杂问题吗

单词从本质上说不如数字精确，但是精确通常会造成成本过高，所以我们在可以容忍不精确的地方使用单词。同样，可以在容忍不确定和不精确的地方使用软计算，以便达到更容易处理、更健壮和降低解决方案的成本（Zadeh，1996）。

在可用的数据不太精确时，我们也使用单词。这在复杂的问题中经常出现。使用“硬”计算不能得到任何解，而软计算却还能够得到很好的解。

软计算和人工智能的区别是什么

传统的人工智能试图用符号术语表达人类的知识。它的基础是符号操作的严格的理论及精确的推理机制，包括前向和后向的链接。传统的人工智能最成功的例子就是专家系统。但只有在获得清晰的知识，并在知识库中表达的时候，专家系统才是好的。这就极大限制了这种系统的实际应用。

但在过去的几年中，人工智能的领域快速扩展，涵盖了人工神经网络、遗传算法，甚至是模糊集理论（Russell and Norvig，2009）。这就使得现代人工智能和软计算的边界变得模糊、不可捉摸。然而，本章的目的不是去争论什么时候其中一种技术变成另一种技术的一部分，而是让读者

理解构建混合智能系统的主要原则。

在混合系统中应该结合哪些内容

据说，Lotfi Zadeh 说过好的混血儿应该有“英国人的风度、德国人的严谨、法国人的厨艺、瑞士人的理财能力和意大利人的浪漫”。但是“英国人的厨艺、德国人的风度、法国人的严谨、意大利人的理财能力和瑞士人的浪漫”就会变得糟糕了。同样，混合智能系统可能是好的，也可能是糟糕的——这取决于构成混合系统的是什么样的组件。因此，我们的目标是选择正确的组件构建一个好的混合系统。

每个组件都有自己的优点和缺点。概率推理主要关注不确定性，模糊逻辑主要处理不精确性，神经网络主要用来学习，进化计算主要用来优化。表 8.1 是不同智能技术的比较。好的混合系统可以将这些技术的优点集中在一起。它们的协同作用使混合系统可以容纳常识，抽取来自原始数据的知识，采用人类那样的推理机制，处理不确定性和不精确性，以及学习去适应快速变化和未知环境。 260

表 8.1 专家系统（ES）、模糊系统（FS）、神经网络（NN）和遗传算法（GA）的比较

	ES	FS	NN	GA
知识表示	○	●	□	■
不确定性的容忍度	○	●	●	●
不精确性的容忍度	□	●	●	●
适应性	□	■	●	●
学习能力	□	□	●	●
解释能力	●	●	□	■
知识发现和数据挖掘	□	■	●	○
可维护性	□	○	●	○

注：表中用作评级的术语是：□差，■更差，○更好和●好。

8.2 神经专家系统

作为智能技术，专家系统和神经网络拥有共同的目标。它们都尝试模拟人类的智能，并最终创建出智能机器。但是二者达到目标的途径是不同的。专家系统依赖逻辑推理和决策树，关注点在于对人类推理的建模；神经网络依赖并行的数据处理，它主要是模拟人脑。专家系统把大脑看做是一个黑盒，而神经网络观察它的结构和功能，尤其是学习的能力。这些区别表现在专家系统和神经网络使用的知识表达和数据处理技术上。

基于规则的专家系统是通过观察或者访问人类专家，得到 IF - THEN 产生式规则来表述知识的。这个过程称作知识获取，它是困难并且昂贵的。另外，一旦这些规则被存入知识库，专家系统本身就不能再改变它了。专家系统不能通过经验学习，也不能适应新环境。只有人类能手动地增加、修改或是删除其中的规则来改变知识库。

在神经网络中，知识存储为神经元之间突触的权重。当把训练集数据输入到网络时，这些知识便在学习阶段获得了。网络一层一层地传递输入的数据，直到产生输出数据为止。如果该输出数据和预计的输出不同，则会计算误差，并在网络中反向传播回输入端。在传递误差时还会修改突触权重。和专家系统不同，神经网络可以在没有人类干涉的情况下学习。

但是，在专家系统中，知识可以分解为单个的规则，用户可以看到并理解系统使用的知识块。相反，在神经网络中，不能把单个的突触权重看做离散的知识片。在这里，知识嵌入到整个网络中，它不能够分解成单个的块，且突触权重的任何改变都可能导致不可预测的结果，神经网 261

络对于用户而言实际上是个黑盒子。

专家系统不能学习，但它可以解释它是如何得到解决方案的。神经网络可以学习，但其行为像个黑盒子。因此，结合这两种技术的优点，我们可以创建功能更强大、有效的专家系统。结合了神经网络和基于规则的专家系统的混合系统叫做神经专家系统（neural expert system）（或连通式专家系统，connectionist expert system）。神经网络的学习能力、归纳能力、健壮性和信息并行处理能力，使得它成为构建新型专家系统的"正确"组件。

图8.1为神经专家系统的基本结构。和基于规则的专家系统不同，神经专家系统的知识库用训练过的神经网络来表示。

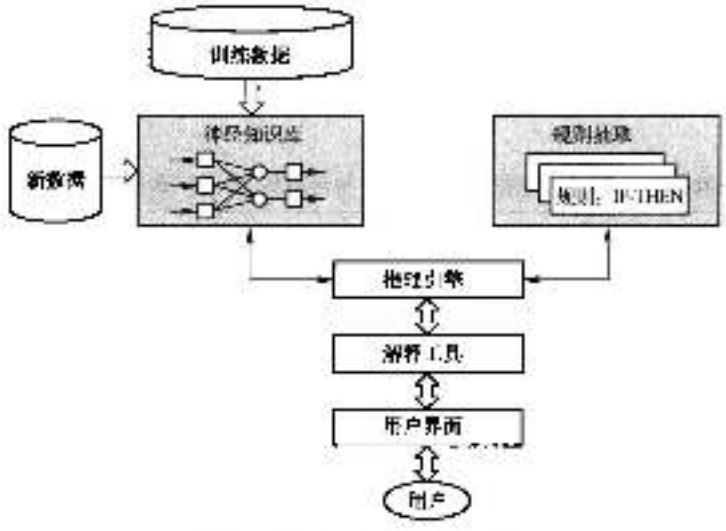


图8.1 神经专家系统的基本结构

神经专家系统可以抽取神经网络中的IF-THEN规则，从而允许它判断和解释其结论。

神经专家系统的核心是推理引擎。它控制系统的信息流，并在神经知识库中发起推理。神经推理引擎也能进行近似推理。

什么是近似推理

在基于规则的专家系统中，推理引擎把每条规则的条件部分和数据库中给定的数据进行比较。如果规则的IF部分和数据库中的数据相匹配，则激发规则并执行其THEN部分。在基于规则的专家系统中，需要精确的匹配，因此，推理引擎不能够处理噪声数据或不完整的数据。

[262]

神经专家系统用训练过的神经网络取代知识库，神经网络能够归纳。换句话说，新的输入数据不一定要和网络训练中使用的数据精确匹配，这就使得神经专家系统能够处理噪声数据和不完整数据。这种能力称为近似推理。

规则抽取单元检查神经知识库，并产生蕴含在训练过的神经网络中的规则。

解释工具向用户解释神经专家系统是如何使用新的输入数据得到某个结论的。

用户界面提供了用户和神经专家系统交互的途径。

神经专家系统如何抽取规则并证明其推理

网络中的神经元通过链接相连，每个链接有一个数值权重。训练过的神经网络的权重决定了相关神经元输入的强度或重要性。这个特征用于抽取规则（Gallant，1993；Nikolopoulos，1997；Sestito and Dillon，1991）。

我们通过一个简单的例子来说明神经专家系统是如何工作的。这个例子是分类问题。要分类的对象属于鸟、飞机和滑翔机之一。用于解决该问题的神经网络如图 8.2 所示。这是一个三层网络，第一层和第二层是完全连接的，每个神经元上标记了其所代表的概念。

第一层是*输入层*。输入层的神经元简单地将外部信号传送给下一层。第二层是*连接层*。这一层的神经元使用的符号激活函数为：

$$Y^{sign} = \begin{cases} +1, 若\ X \geqslant 0 \\ -1, 若\ X < 0 \end{cases} \tag{8.1}$$

其中，X 为神经元的净权重输入，

$$X = \sum_{i=1}^{n} x_i w_i$$

x_i 和 w_i 分别为输入 i 的值和它的权重。n 为输入神经元的数量。

第三层是输出层。在本例中，每个输出神经元接收一个来自于连接神经元的输入。第二层和第三层之间的权重设置为单位值。

你可能已经注意到，IF - THEN 规则可以很自然地映射到三层神经网络中，其中第三层（非连接层）为规则的结果部分。此外，给定规则的强度或确信因子，可以和各自连接及非连接的 263
神经元的权重联系起来（Fu，1993；Kasabov，2007）。稍后，我们将讨论如何将规则映射到神经网络中，不过现在我们要先回到我们的例子中。

神经知识库用一个训练实例集进行训练。图 8.2 显示了第一层和第二层之间的实际数值权重。假设现在将输入层的每个输入设置成 +1（真），或 -1（假），或 0（未知），就可以对任何输出神经元的动作进行语义解释。例如，如果对象有 Wings（+1），Beak（+1）和 Feathers（+1），但没有 Engine（-1），则可以得出对象为 Bird（+1）的结论：

$$\begin{aligned} X_{Rule1} &= 1 \times (-0.8) + 0 \times (-0.2) + 1 \times 2.2 + 1 \times 2.8 + (-1) \times (-1.1) \\ &= 5.3 > 0 \\ Y_{Rule1} &= Y_{Bird} = +1 \end{aligned}$$

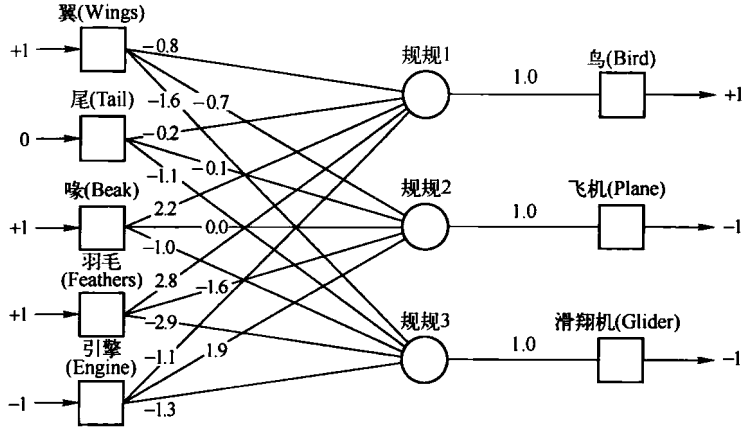


图 8.2　神经知识库

同样，可以得到该对象不是 Plane 的结论：

$$\begin{aligned} X_{Rule2} &= 1 \times (-0.7) + 0 \times (-0.1) + 1 \times 0.0 + 1 \times (-1.6) + (-1) \times 1.9 \\ &= -4.2 < 0 \\ Y_{Rule2} &= Y_{Plane} = -1 \end{aligned}$$

对象也不是 Glider：

$$X_{Rule3} = 1 \times (-0.6) + 0 \times (-1.1) + 1 \times (-1.0) + 1 \times (-2.9) + (-1) \times (-1.3)$$
$$= -4.2 < 0$$

264

$$Y_{Rule3} = Y_{Glider} = -1$$

现在，给每个输入神经元附加上相应的问题：

神经元：翅膀（Wings）　　问题：对象有翅膀吗？
神经元：尾巴（Tail）　　问题：对象有尾巴吗？
神经元：喙（Beak）　　问题：对象有喙吗？
神经元：羽毛（Feathers）　　问题：对象有羽毛吗？
神经元：引擎（Engine）　　问题：对象有引擎吗？

我们可以让系统提示用户为输入变量赋初始值。系统的目的首先是要得到最重要的信息，然后是尽快地得到结论。

系统如何知道最重要的信息是什么，以及是否有足够的信息得到结论

某个输入神经元的重要性由该输入的权重的绝对值确定。例如，对名为“规则 1”的神经元，输入 Feathers 的重要性比输入 Wings 的重要性高。因此，可以为系统建立下面的对话：

```
PURSUING:
> Bird
ENTER INITIAL VALUE FOR THE INPUT FEATHERS:
> +1
```

现在的任务是看一下得到的信息是否足够做出结论。为此，我们可以使用下面的启发式规则（Gallant，1993）：如果神经元已知的净权重输入大于未知输入的权重的绝对值的和，则可以进行推理。

该启发式规则的数学表达式为：

$$\sum_{i=1}^{n} x_i w_i > \sum_{j=1}^{n} |w_j| \tag{8.2}$$

其中，$i \in$ KNOWN，$j \notin$ KNOWN，并且 n 为神经元输入的数量。

本例中，如果输入 Feathers 变成已知，则可以得到：

```
KNOWN = 1 × 2.8 = 2.8
UNKNOWN = |-0.8| + |-0.2| + |2.2| + |-1.1| = 4.3
KNOWN < UNKNOWN
```

因此，还不能进行神经元规则 1 的推导。还要求用户提供下一个最重要的输入的值，输入 Beak：

```
ENTER INITIAL VALUE FOR THE INPUT BEAK:
> +1
```

265

现在有：

```
KNOWN = 1 × 2.8 + 1 × 2.2 = 5.0
UNKNOWN = |-0.8| + |-0.2| + |-1.1| = 2.1
KNOWN > UNKNOWN
```

因此，根据启发式规则（8.2），便可以得出下面的推理：

```
CONCLUDE: BIRD IS TRUE
```

其中，KNOWN 给出了神经元“规则 1”的净权重输入，UNKNOWN 说明引起净输入改变的未知输入的最坏可能。本例中，净权重输入的改变不会超过 ±2.1。因此，无论未知输入是什么，神经元规则 1 的输出都应该大于 0，所以可以得到 Bird 必为真的推理。

现在检查如何抽取单个的规则来证明推理。在这里，我们使用一个只关注和问题中的神经元直接关联的神经元的简单算法（Gallant，1988）。再次考虑图 8.2 所示的例子，判断 Bird 为真的推理。因为第一层中的所有神经元都直接和神经元“规则 1”相连，所以可以预期抽取的规则涉及所有的 5 个神经元：Wings、Tail、Beak、Feathers 和 Engine。

首先，确定所有起作用的输入和每个输入贡献的大小（Gallant，1993）。如果输入 i 不会在相反方向上移动净权重输入，就认为其是有作用的。贡献的大小由起作用的输入 i 的权重的绝对值 $|w_i|$ 确定。

然后，将所有有贡献的输入根据其大小按降序排列。在本例中，对推理“Bird 为真”有贡献的输入列表如下：

输入：Feathers　大小：2.8
输入：Beak　大小：2.2
输入：Engine　大小：1.1
输入：Tail　大小：0.2

通过列表可以创建规则，其条件部分用贡献最大的输入来表示：

IF　*Feathers* is true
THEN　*Bird* is true

接下来就要证明该规则。换句话说，要确保规则能够通过有效性测试。可以用启发式规则（8.2）来证明：

KNOWN = 1 × 2.8 = 2.8
UNKNOWN = |−0.8| + |−0.2| + |2.2| + |−1.1| = 4.3
KNOWN < UNKNOWN

266

规则现在还是无效的，因此需要增加贡献次大的输入作为该规则条件部分的一个子句：

IF　*Feathers* is true
AND　*Beak* is true
THEN　*Bird* is true

现在得到：

KNOWN = 1 × 2.8 + 1 × 2.2 = 5.0
UNKNOWN = |−0.8| + |−0.2| + |−1.1| = 2.1
KNOWN > UNKNOWN

该规则通过了有效性测试。这也是最大通用规则，也就是说，去除了所有导致无效的条件子句。

类似地，可以得到证明推理 Plane 为假和 Glider 为假的规则：

IF　*Engine* is false
AND　*Feathers* is true
THEN　*Plane* is false

IF　*Feathers* is true
AND　*Wings* is true
THEN　*Glider* is false

该例也说明了即使在数据不完整的情况下，神经专家系统也可以得到有用的结论（例如，在本例中 Tail 是未知的）。

在本例中，假设神经专家系统拥有正确的经过训练的神经知识库。然而，在真实世界中，训练数据通常是不充分的。我们也假设对问题领域没有任何先验知识。实际上，我们通常会有一些不完善的相关知识。我们能否用领域知识来确定神经知识库的初始结构，然后用给定的训练数据集来训练它，最后将已训练的神经网络解释为 IF - THEN 规则集？

前面提到过，表达知识领域的 IF - THEN 规则集可以映射到多层神经网络中。图 8.3 显示了一组映射到 5 层神经网络的规则集。合取层和析取层的权重表明了规则的强度，因此可以认为它是相关规则的确信因子。

在建立起神经网络知识库的初始结构后，就可以根据给定的训练数据集来训练网络了。这可以使用反向传播算法这样的合适的训练算法来完成。训练阶段完成时，检查神经网络知识库，提取并改进（如果需要）最初的 IF - THEN 规则集。因此，神经专家系统可以使用 IF-THEN 规则表示的领域知识，也可以使用数值数据集。实际上，神经专家系统提供了神经网络和基于规则

267

规则1：
IF *a*1 AND *a*3 THEN *b*1(0.8)

规则2：
IF *a*1 AND *a*4 THEN *b*1(0.2)

规则3：
IF *a*2 AND *a*5 THEN *b*2(−0.1)

规则4：
IF *a*3 AND *a*4 THEN *b*3(0.9)

规则5：
IF *a*5 THEN *b*3 (0.6)

规则6：
IF *a*1 AND *b*3 THEN *c*1(0.7)

规则7：
IF *b*2 THEN *c*1(0.1)

规则8：
IF *b*2 AND *b*3 THEN *c*2(0.9)

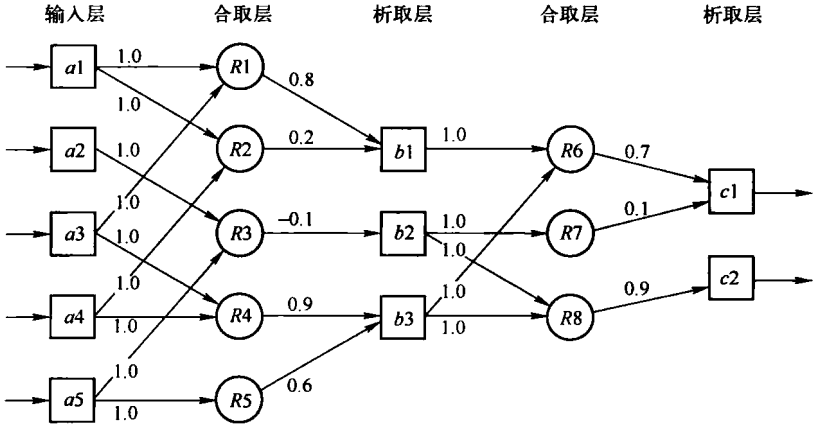


图 8.3 多层知识库的例子

的系统之间的双向链接。

遗憾的是，神经专家系统仍旧受到布尔逻辑的限制，要想表达连续输入变量，可能导致规则的无限增长。这就严重限制了神经专家系统的应用领域。克服这个问题的一般方法是使用模糊逻辑。

8.3 神经－模糊系统

268

模糊逻辑和神经网络是构建智能系统的两种相互补充的工具。神经网络是低级的计算结构，处理原始数据时性能良好，模糊逻辑使用从领域专家那里获取的语言信息进行高级推理。但是模糊系统没有学习的能力，并且不能在新的环境中调整自己。另外，神经网络虽然可以学习，但它们对于用户而言是不透明的。将神经网络和模糊系统融合进一个集成的系统，为构建智能系统提供了有希望的方法。集成的神经模糊系统可以将神经网络的并行计算和学习能力与模糊系统的类似于人类的知识表达方式和解释能力结合起来。这样，神经网络变得更透明，而模糊系统也有了学习能力。

实际上，神经－模糊系统是功能相当于模糊推理模型的神经网络。可以训练它来开发 IF－THEN 模糊规则以及确定该系统输入和输出变量的隶属函数。专家的知识可以很容易地整合到神经－模糊系统的结构中。同时，这种组合结构避免了需要大量计算的模糊推理。

神经－模糊系统看上去是什么样

神经－模糊系统的结构和多层神经网络很相似。通常，神经－模糊系统有输入层和输出层以及 3 个隐含层，隐含层用来表示隶属函数和模糊规则。

图 8.4 为 Mamdani 模糊推理模型，图 8.5 为该模型相应的神经－模糊系统。为了简单起见，

我们假设模糊系统有两个输入 x1 和 x2，一个输出 y。输入 x1 用模糊集 A_1、A_2、A_3 表示，输入 x2 用模糊集 B_1、B_2、B_3 表示，输出 y 用模糊集 C_1、C_2 来表示。

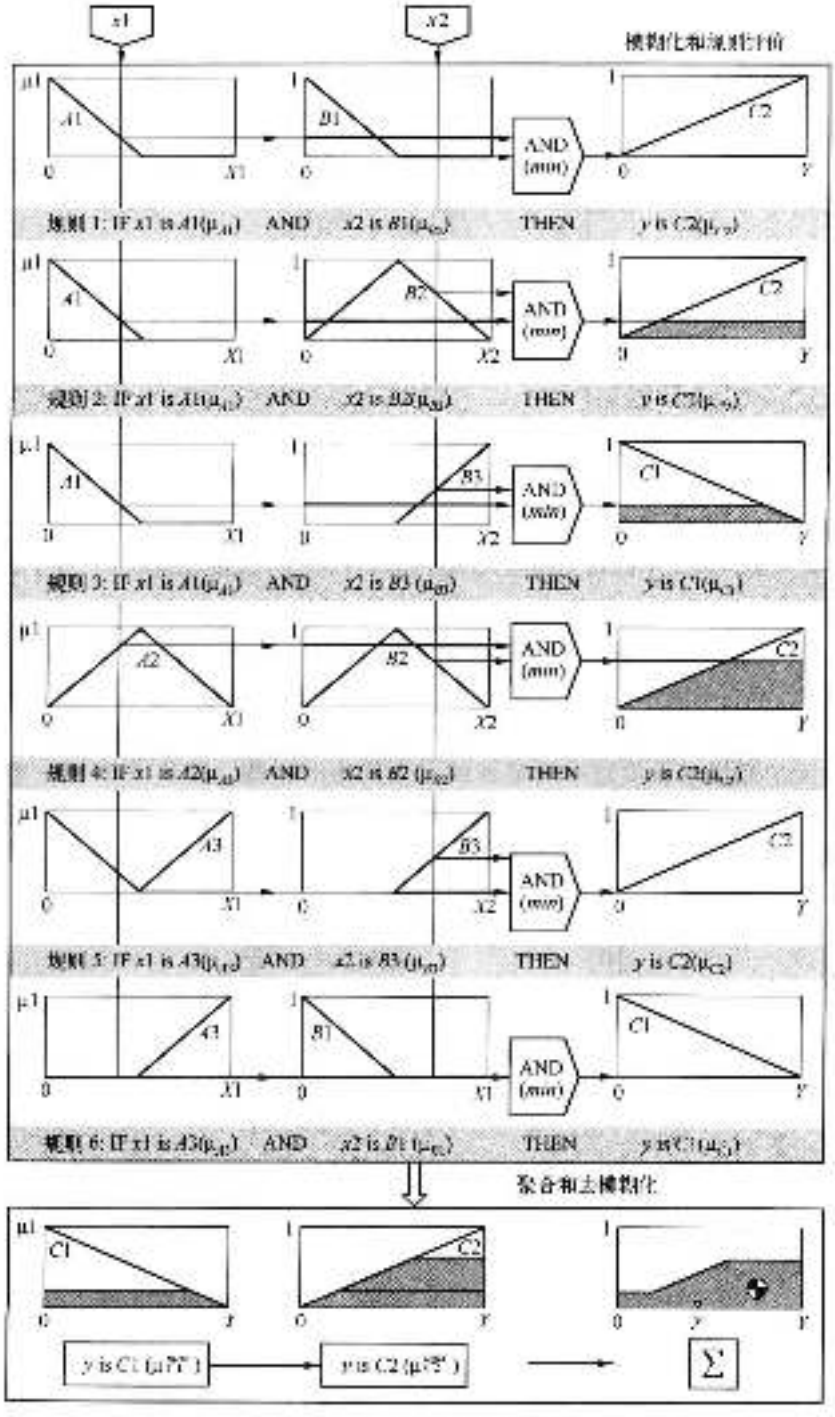


图 8.4 Mamdani 模糊推理系统

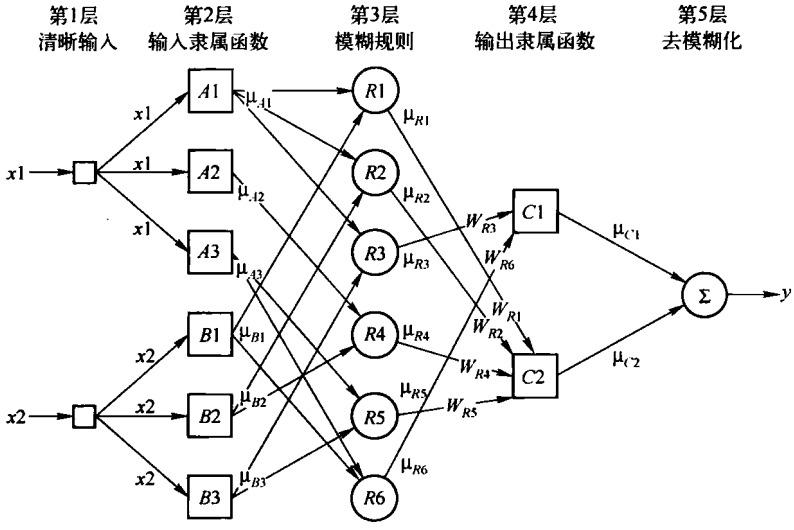


图 8.5 神经 - 模糊等价系统

神经 - 模糊系统的每一层都和模糊推理过程的某个步骤关联。

第 1 层是输入层，该层中的每个神经元将外部清晰的信号直接传递给下一层，即

$$y_i^{(1)} = x_i^{(1)} \tag{8.3}$$

其中，$x_i^{(1)}$、$y_i^{(1)}$ 分别是第 1 层神经元 i 的输入和输出。

第 2 层是输入成员层或模糊化层。该层的神经元表示模糊规则的前项使用的模糊集。模糊化神经元接收到清晰的输入后，确定该输入属于神经元模糊集的程度，如下所述。

成员神经元的激活函数设置成指定神经元模糊集的函数。图 8.4 的例子中使用三角形集合。因此，第 2 层的神经元的激活函数设置为三角形隶属函数（虽然模糊化神经元可能有模糊系统通常使用的隶属函数）。三角形隶属函数可以使用下面的两个参数 {a, b} 来指定：

$$y_i^{(2)} = \begin{cases} 0, & \text{若 } x_i^{(2)} \leqslant a - \dfrac{b}{2} \\ 1 - \dfrac{2\left|x_i^{(2)} - a\right|}{b}, & \text{若 } a - \dfrac{b}{2} < x_i^{(2)} < a + \dfrac{b}{2} \\ 0, & \text{若 } x_i^{(2)} \geqslant a + \dfrac{b}{2} \end{cases} \tag{8.4}$$

其中，参数 a 和 b 分别控制三角形的中心和宽度，$x_i^{(2)}$ 和 $y_i^{(2)}$ 分别是第 2 层模糊化神经元 i 的输入和输出。

图 8.6 描述了三角形函数及参数 a 和 b 的变化产生的效果。可以看出，模糊化神经元的输出不仅取决于输入，还受三角形激活函数的中心 a 和宽度 b 的限制。神经元输入可以保持不变，但参数 a 和 b 的改变可以使输出发生变化。换句话说，神经 - 模糊系统中模糊化神经元的参数 a 和 b 的作用如同神经网络中的突触权重。

第 3 层是模糊规则层。该层的每个神经元都与一个模糊规则相对应。模糊规则神经元接收来自于模糊化神经元的输入，表示模糊层是模糊规则的前项。例如，神经元 $R1$ 和规则 1 对应，它接收来自神经元 $A1$ 和 $B1$ 的输入。

269
~
271

在模糊系统中，如果给定的规则有多个前项，那么模糊操作得出一个数值，该数值描述评估

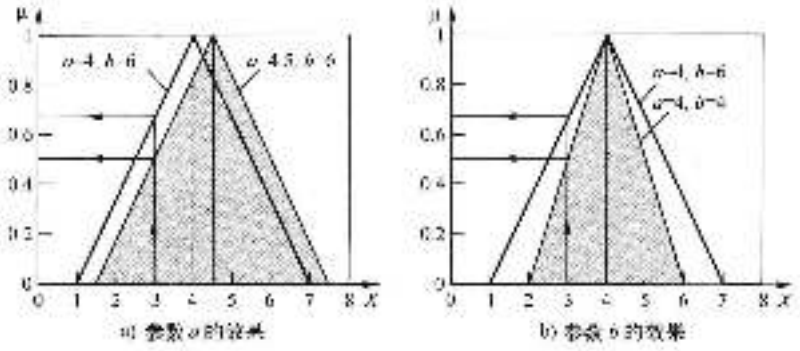


图 8.6 模糊神经元的三角形激活函数

前项的结果。规则前项的连接用模糊操作"交"来评估。使用相同的模糊操作可以合并模糊规则神经元的多个输入。在模糊－神经系统中，"交"可以用乘积操作实现。因此，第3层的神经元 i 的输出为：

$$y_i^{(3)} = x_{i1}^{(3)} \times x_{i2}^{(3)} \times \cdots \times x_{ik}^{(3)} \tag{8.5}$$

其中，$x_{i1}^{(3)}$，$x_{i2}^{(3)}$，…，$x_{ik}^{(3)}$ 为第3层模糊规则神经元 i 的输入，$y_i^{(3)}$ 为输出。例如：

$$y_{R1}^{(3)} = \mu_{A1} \times \mu_{B1} = \mu_{R1}$$

μ_{R1} 的值表示模糊规则神经元 $R1$ 的激活强度。

第3层和第4层之间的权重表示相应模糊规则的可信度（即确信因子）的归一化程度。这些权重可以在训练神经－模糊系统的时候调修。

什么是模糊规则可信度的归一化程度

神经－模糊系统中的不同规则可能和不同的可信度有关。在图8.4中，专家可能将可信的程度和每个模糊 IF-THEN 规则关联起来，并将权重设置在［0，1］区间内。在训练过程中，权重可能发生改变。要将权重值保持在指定的范围内，就要在每次迭代中，通过将各自的权重值除以最高的权重，这就是权重的归一化。

第4层是输出成员层。该层中的神经元表示模糊规则后项中使用的模糊集。输出成员神经元接收相应模糊规则神经元的输入，并通过模糊操作"并"将这些输入合并。该操作可以通过概率论的 OR 操作（即代数和）实现，即 272

$$y_i^{(4)} = x_{i1}^{(4)} \oplus x_{i2}^{(4)} \oplus \cdots \oplus x_{ik}^{(4)} \tag{8.6}$$

其中，$x_{i1}^{(4)}$，$x_{i2}^{(4)}$，…，$x_{ik}^{(4)}$ 为输入，$y_i^{(4)}$ 是第4层的输出成员神经元 i 的输出。例如，

$$y_{B1}^{(4)} = \mu_{R2} \oplus \mu_{R5} = \mu_{B1}$$

μ_{R2} 的值表示模糊规则神经元 $R3$ 和 $R6$ 的综合激发强度。实际上输出成员层的神经元激发强度的合并方式和图8.4所示的模糊规则真值的合并方式相同。

在 Mamdani 模糊系统中，输出模糊集与相应模糊规则的真值求剪切。在神经－模糊系统中，剪切输出成员神经元的激活函数。例如，神经元 $C1$ 的隶属函数用综合激发强度 μ_{C1} 表示。

第5层是去模糊化层。该层中的每个神经元表示神经－模糊系统的一个输出。使用类成的激发强度削除输出模糊集，并将其合成为一个单一的模糊集。

神经－模糊系统的输出是清晰的，因此合并输出模糊集必须进行去模糊化。神经－模糊系统可以使用标准的去模糊化方法，其中包括质心技术。在本例中，我们使用的是和－乘积合成方法（Jang et al.，1997），这种方法为 Mamdani 类推理提供了计算捷径。

和－乘积合成方法计算出清晰的输出，并将结果作为所有输出隶属函数质心的权重平均值。

例如，剪切后的模糊集 $C1$ 和 $C2$ 的质心平均权重的计算方法是：

$$y=\frac{\mu_{C1}\times a_{C1}\times b_{C1}+\mu_{C2}\times a_{C2}\times b_{C2}}{\mu_{C1}\times b_{C1}+\mu_{C2}\times b_{C2}} \tag{8.7}$$

其中，a_{C1}和a_{C2}为中心，b_{C1}和b_{C2}分别是模糊集 $C1$ 和 $C2$ 的宽度。

神经－模糊系统如何学习

神经－模糊系统本质上是多层神经网络，因此，它可以使用专用于神经网络的标准学习算法，其中包括反向传播算法（Lin and Lee，1996；Nauck et al.，1997；Von Altrock，1997；Kasabov，2007）。在对系统应用训练所需的输入/输出例子时，反向传播算法会计算系统的输出并和
273 该训练例子的期望输出作比较。它们之间的差（也称作误差）由输出层反向传播到网络的输入层。在误差传递时修改神经元激活函数。为了确定必要的修改，反向传播算法会区分不同神经元的激活函数。

下面我们通过一个简单的例子来说明神经－模糊系统是如何工作的。图 8.7 为三维输入/输出空间 $X1\times X2\times Y$ 中 100 个训练模式的分布情况。这里的每个训练模式用 3 个变量来确定：两个输入变量 $x1$ 和 $x2$ 及一个输出变量 y。输入和输出变量用两个语言变量表示：small（S）和 large（L）。

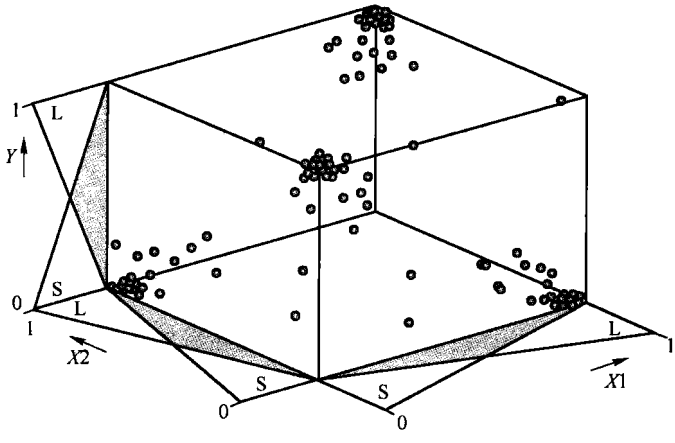
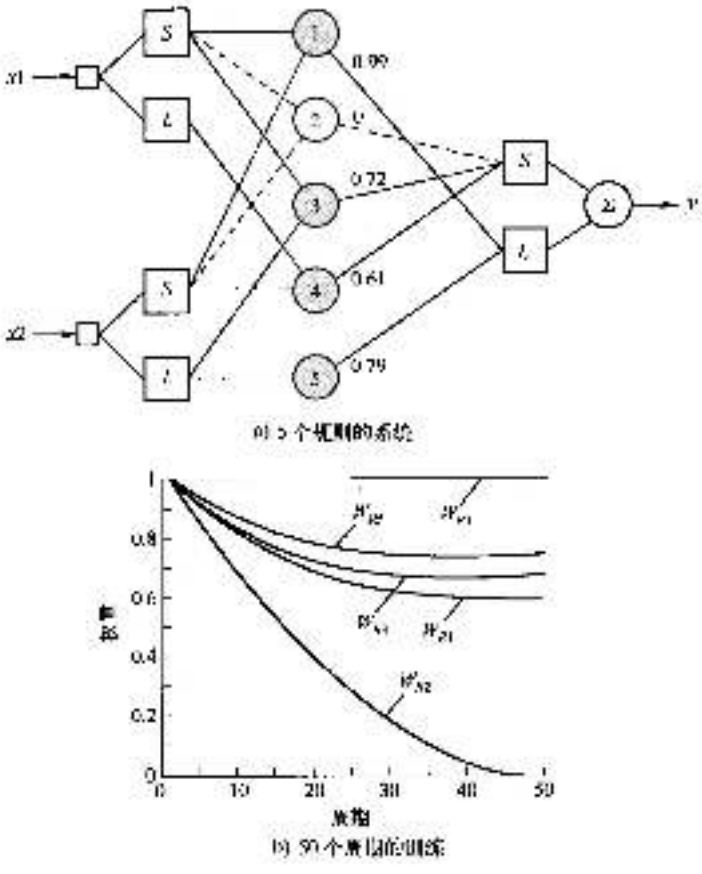


图 8.7 三维输入/输出空间的训练模式

图 8.7 中的数据集用来训练图 8.8a 中的 5 个规则的神经－模糊系统。假设系统结构中的模糊 IF－THEN 规则是由领域专家提供的。先前的或现有的知识可以显著加快系统训练的速度。此外，如果训练数据的质量不佳，专家的知识就是得到解决方案的唯一途径。但是，专家偶尔也会犯错。因此神经－模糊系统中的某些规则可能是错误的或是多余的（例如在图 8.8a 中，规则 1 或规则 2 可能是错误的，因为它们的 IF 部分完全相同，而 THEN 部分则不同）。因此，神经－模糊系统应该能够识别这些不好的规则。

在图 8.8a 中，第 3 层和第 4 层间的初始权重被设置为单位值。在训练中，神经－模糊系统使用反向传播算法调节权重并修改输入和输出隶属函数。训练过程会持续到误差的平方和小于 0.001 为止。在图 8.8b 可以看到，权重 W_{R2}降低为 0，而其他的权重还很高。这表明，规则 2 是完全错误的，因此可以在对神经－模糊系统没有任何危害的情况下将它去除。此系统还有 4 条规则，你可能已经注意到了，这些规则表明了异或操作（XOR）的
274 行为。

本例使用的训练数据中包含很多和 XOR 操作不一致的“坏”模式。但是，神经－模糊系统



a) 5个规则的系统

b) 50个规则的训练

图8.8 进行异或操作的5个规则的神经－模糊系统

依然有能力识别这些错误的规则。

在XOR例子中，专家给出了5条模糊规则，其中一条是错误的。另外，我们也不能确定“专家”是否遗漏了一些规则。我们该如何减少对专家知识的依赖呢？神经－模糊系统能否直接从数值数据中抽取规则呢？

给定输入和输出的语言值，神经－模糊系统就可以自动产生完备的模糊IF-THEN规则集。图8.9展示了为XOR例子创建的系统。该系统包含$2^2 \times 2 = 8$条规则。由于本例系统不包含专家的知识，因此我们将第3层和第4层的初始权重设为0.5。训练后，可以排除确信因子小于一个很小的数（例如0.1）的所有规则。结果，得到了完全相同的表示XOR操作的、有4条模糊IF－THEN规则的规则集。这个简单的例子证明了神经－模糊系统确实可以直接从数值数据中抽取模糊规则。 275

模糊逻辑和神经网络的结合是设计智能系统的很有力的工具。人类专家可以把领域知识以语言变量和模糊规则的形式放入神经模糊系统中。如果可以得到有代表性的例子的集合，神经模糊系统可以自动将其转换成一组强壮的模糊IF－THEN规则，这样在构建智能系统时就可以减少对专家知识的依赖。

到目前为止，我们讨论了实现Mamdani模糊推理模型的神经－模糊系统。但是，Sugeno模型才是目前基于数据的模糊建模的最常用的选择。

Roger Jang提出了和Sugeno模糊推理模型功能一致的神经网络（Jang，1993）。他称之为自适应的神经－模糊推理系统（Adaptive Neuro-Fuzzy Inference Syetem，ANFIS）。

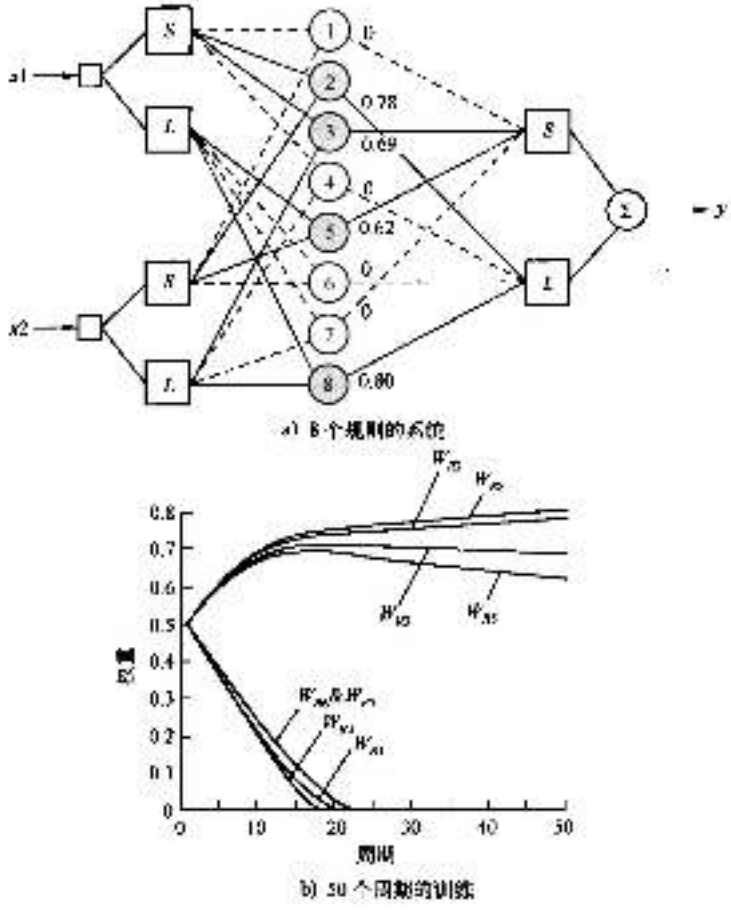


图 8.9 用于异或操作的带有 8 个规则的神经－模糊系统

8.4 ANFIS

Sugeno 模糊模型是一种用来在给定的输入/输出数据集中产生模糊规则的系统方法。一个典型的 Sugeno 模糊规则可以表达成以下形式：

IF x_1 is A_1
AND x_2 is A_2
⋮
AND x_m is A_m
THEN $y = f(x_1, x_2, \cdots, x_m)$

其中，x_1，x_2，…，x_m 为输入变量，A_1，A_2，…，A_m 为模糊集；y 要么是常数，要么是输入变量的线性函数。如果 y 是常数，我们将得到零阶 Sugeno 模糊模型，其中规则的结果由一个单件决定。如果 y 是一阶多项式函数，即

$$y = k_0 + k_1 x_1 + k_2 x_2 + \cdots + k_m x_m$$

就可以得到一阶 Sugeno 模糊模型。

Jang 的 ANFIS 通常用 6 层的前馈神经网络来表示。图 8.10 为一阶 Sugeno 模糊模型对应的 ANFIS 体系结构。为了简单起见，我们假设 ANFIS 有两个输入 x1 和 x2，一个输出 y。每个输入用两个模糊集来表示，输出用一阶多项式表示。ANFIS 实现如下的 4 个规则：

规则 1:
IF　$x1$ is $A1$
AND　$x2$ is $B1$
THEN　$y = f_1 = k_{10} + k_{11}x1 + k_{12}x2$

规则 2:
IF　$x1$ is $A2$
AND　$x2$ is $B2$
THEN　$y = f_2 = k_{20} + k_{21}x1 + k_{22}x2$

规则 3:
IF　$x1$ is $A2$
AND　$x2$ is $B1$
THEN　$y = f_3 = k_{30} + k_{31}x1 + k_{32}x2$

规则 4:
IF　$x1$ is $A1$
AND　$x2$ is $B2$
THEN　$y = f_4 = k_{40} + k_{41}x1 + k_{42}x2$

其中，$x1$ 和 $x2$ 为输入变量，$A1$ 和 $A2$ 是论域 $X1$ 上的模糊集；$B1$ 和 $B2$ 是论域 $X2$ 上的模糊集；k_{i0}、k_{i1}、k_{i2}为规则 i 指定的参数集。

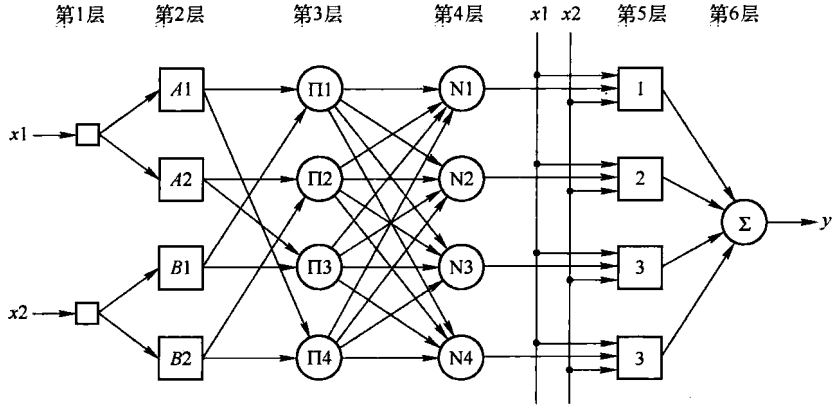


图 8.10　自适应神经 - 模糊推理系统（ANFIS）

我们现在来讨论 Jang 的 ANFIS 中每一层的作用。

第 1 层是输入层。该层的神经元简单地将外部清晰的信号传送给第 2 层，即

$$y_i^{(1)} = x_i^{(1)} \tag{8.8}$$

其中，$x_i^{(1)}$ 是第 1 层中的输入神经元 i 的输入，$y_i^{(1)}$ 是第 1 层中输入神经元 i 的输出。

第 2 层是模糊化层。该层中的神经元执行模糊化操作。在 Jang 的模型中，模糊化神经元有一个*钟形激活函数*（bell activation function）。

钟形激活函数为规则的钟形形状，其定义为：

$$y_i^{(2)} = \frac{1}{1 + \left(\dfrac{x_i^{(2)} - a_i}{c_i}\right)^{2b_i}} \tag{8.9}$$

其中，$x_i^{(2)}$ 是第 2 层中的神经元 i 的输入，$y_i^{(2)}$ 是第二层输入神经元的输出。a_i、b_i、c_i 分别为控制神经元 i 的钟形激活函数的中心、宽度和斜率的参数。

第 3 层是规则层。该层中的每个神经元和 Sugeno 类型的单个模糊规则相对应。规则神经元从各自的模糊化神经元接收输入，并计算它表示的规则激发强度。在 ANFIS 中，规则前项的连接由操作“乘积”来评估。因此，第 3 层的神经元的输出可表示成：

$$y_i^{(3)} = \prod_{j=1}^{k} x_{ji}^{(3)} \tag{8.10}$$

其中，$x_{ji}^{(3)}$和 $y_i^{(3)}$ 分别为第 3 层规则神经元 i 的输入和输出。例如，

$$y_{\Pi 1}^{(3)} = \mu_{A1} \times \mu_{B1} = \mu_1$$

其中，μ_1 的取值代表规则 1 的激发强度或真值。

276
278

第 4 层是归一化层。该层的每个神经元接收来自规则层的所有神经元的输入，并计算给定规则的*归一化激发强度*。

归一化强度是给定规则的激发强度和所有规则激发强度的总和的比值。它表示给定规则对最终结果的贡献。

因此，第 4 层神经元 i 的输出为：

$$y_i^{(4)} = \frac{x_{ii}^{(4)}}{\sum_{j=1}^{n} x_{ji}^{(4)}} = \frac{\mu_i}{\sum_{j=1}^{n} \mu_j} = \bar{\mu}_i \tag{8.11}$$

其中，$x_{ji}^{(4)}$ 是第 4 层神经元 i 从第 3 层的神经元 j 处接收的输入，n 为规则神经元的总数。例如，

$$y_{N1}^{(4)} = \frac{\mu_1}{\mu_1 + \mu_2 + \mu_3 + \mu_4} = \bar{\mu}_1$$

第 5 层是去模糊化层。该层中的每个神经元均连接到各自的归一化神经元上，同时接收初始输入 $x1$ 和 $x2$。去模糊化神经元计算给定规则的带权重的后项值，

$$y_i^{(5)} = x_i^{(5)}[k_{i0} + k_{i1}x1 + k_{i2}x2] = \bar{\mu}_i[k_{i0} + k_{i1}x1 + k_{i2}x2] \tag{8.12}$$

其中，$x_i^{(5)}$ 和 $y_i^{(5)}$ 分别为第 5 层去模糊化神经元 i 的输入和输出，k_{i0}、k_{i1}、k_{i2} 是规则 i 的后项参数的集合。

第 6 层表示为一个总和神经元。该神经元计算所有去模糊化神经元输出的总和，并产生最后的 ANFIS 输出 y：

$$y = \sum_{i=1}^{n} x_i^{(6)} = \sum_{i=1}^{n} \bar{\mu}_i[k_{i0} + k_{i1}x1 + k_{i2}x2] \tag{8.13}$$

因此，图 8.10 所示的 ANFIS 在功能上却是相当于一阶 Sugeno 模糊模型。

但是，以多项式形式指定规则后项非常困难，甚至是不可能的。因此，在 ANFIS 处理问题
279 时，没有必要给出规则后项参数的任何先验知识。ANFIS 会学习这些参数并调节隶属函数。

ANFIS 如何学习

ANFIS 使用混合学习算法，这种算法融合了最小二乘估计和梯度下降法（Jang，1993）。首先，将初始激活函数指定给每个成员神经元。设置连接到输入 x_i 的神经元的函数中心，以便 x_i 的域被平均分割，且各自函数的宽度和斜率设置成允许足够的重叠。

在 ANFIS 训练算法中，每个周期由前向传递和反向传播组成。在前向传递中，输入模式的训练集（输入向量）出现在 ANFIS 中，神经元的输出要一层一层地计算，规则后项参数由最小二乘估计表示。在 Sugeno 类模糊推理中，输出 y 为线性函数。因此，给定隶属参数的值和 P 输入/输出模式的训练集，就可以通过后项参数建立 P 线性方程：

$$\begin{cases} y_d(1) = \bar{\mu}_1(1)f_1(1) + \bar{\mu}_2(1)f_2(1) + \cdots + \bar{\mu}_n(1)f_n(1) \\ y_d(2) = \bar{\mu}_1(2)f_1(2) + \bar{\mu}_2(2)f_2(2) + \cdots + \bar{\mu}_n(2)f_n(2) \\ \vdots \\ y_d(p) = \bar{\mu}_1(p)f_1(p) + \bar{\mu}_2(p)f_2(p) + \cdots + \bar{\mu}_n(p)f_n(p) \\ \vdots \\ y_d(P) = \bar{\mu}_1(P)f_1(P) + \bar{\mu}_2(P)f_2(P) + \cdots + \bar{\mu}_n(P)f_n(P) \end{cases} \tag{8.14}$$

或者

$$
\begin{cases}
y_d(1)=\bar{\mu}_1(1)[k_{10}+k_{11}x_1(1)+k_{12}x_2(1)+\cdots+k_{1m}x_m(1)] \\
\quad +\bar{\mu}_2(1)[k_{20}+k_{21}x_1(1)+k_{22}x_2(1)+\cdots+k_{2m}x_m(1)]+\cdots \\
\quad +\bar{\mu}_n(1)[k_{n0}+k_{n1}x_1(1)+k_{n2}x_2(1)+\cdots+k_{nm}x_m(1)] \\
y_d(2)=\bar{\mu}_1(2)[k_{10}+k_{11}x_1(2)+k_{12}x_2(2)+\cdots+k_{1m}x_m(2)] \\
\quad +\bar{\mu}_2(2)[k_{20}+k_{21}x_1(2)+k_{22}x_2(2)+\cdots+k_{2m}x_m(2)]+\cdots \\
\quad +\bar{\mu}_n(2)[k_{n0}+k_{n1}x_1(2)+k_{n2}x_2(2)+\cdots+k_{nm}x_m(2)] \\
\qquad \vdots \\
y_d(p)=\bar{\mu}_1(p)[k_{10}+k_{11}x_1(p)+k_{12}x_2(p)+\cdots+k_{1m}x_m(p)] \\
\quad +\bar{\mu}_2(p)[k_{20}+k_{21}x_1(p)+k_{22}x_2(p)+\cdots+k_{2m}x_m(p)]+\cdots \\
\quad +\bar{\mu}_n(p)[k_{n0}+k_{n1}x_1(p)+k_{n2}x_2(p)+\cdots+k_{nm}x_m(p)] \\
\qquad \vdots \\
y_d(P)=\bar{\mu}_1(P)[k_{10}+k_{11}x_1(P)+k_{12}x_2(P)+\cdots+k_{1m}x_m(P)] \\
\quad +\bar{\mu}_2(P)[k_{20}+k_{21}x_1(P)+k_{22}x_2(P)+\cdots+k_{2m}x_m(P)]+\cdots \\
\quad +\bar{\mu}_n(P)[k_{n0}+k_{n1}x_1(P)+k_{n2}x_2(P)+\cdots+k_{nm}x_m(P)]
\end{cases}
\tag{8.15}
$$

280

其中，m 是输入变量的个数，n 是规则层中神经元的个数。当输入 $x_1(p)$，…，$x_m(p)$ 时，$y_d(p)$ 为 ANFIS 的期望总输出。

若使用矩阵形式，则有

$$y_d = \boldsymbol{A}\boldsymbol{k} \tag{8.16}$$

其中，y_d 是 $P\times 1$ 的期望输出向量，

$$y_d=\begin{bmatrix} y_d(1) \\ y_d(2) \\ \vdots \\ y_d(p) \\ \vdots \\ y_d(P) \end{bmatrix}$$

$\boldsymbol{A}$ 是 $P\times n(1+m)$ 矩阵：

$$\boldsymbol{A}=\begin{bmatrix}
\bar{\mu}_1(1) & \bar{\mu}_1(1)x_1(1) & \cdots & \bar{\mu}_1(1)x_m(1) & \cdots & \bar{\mu}_n(1) & \bar{\mu}_n(1)x_1(1) & \cdots & \bar{\mu}_n(1)x_m(1) \\
\bar{\mu}_1(2) & \bar{\mu}_1(2)x_1(2) & \cdots & \bar{\mu}_1(2)x_m(2) & \cdots & \bar{\mu}_n(2) & \bar{\mu}_n(2)x_1(2) & \cdots & \bar{\mu}_n(2)x_m(2) \\
\vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\
\bar{\mu}_1(p) & \bar{\mu}_1(p)x_1(p) & \cdots & \bar{\mu}_1(p)x_m(p) & \cdots & \bar{\mu}_n(p) & \bar{\mu}_n(p)x_1(p) & \cdots & \bar{\mu}_n(p)x_m(p) \\
\vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\
\bar{\mu}_1(P) & \bar{\mu}_1(P)x_1(P) & \cdots & \bar{\mu}_1(P)x_m(P) & \cdots & \bar{\mu}_n(P) & \bar{\mu}_n(P)x_1(P) & \cdots & \bar{\mu}_n(P)x_m(P)
\end{bmatrix}$$

k 为未知后项参数的 $n(1+m)\times 1$ 维向量：

$$\boldsymbol{k}=[k_{10}k_{11}k_{12}\cdots k_{1m}k_{20}k_{21}k_{22}\cdots k_{2m}\cdots k_{n0}k_{n1}k_{n2}\cdots k_{nm}]^{\mathrm{T}}$$

通常训练过程中的输入/输出模式 P 的数量要大于后项参数 $n(1+m)$ 的数量。这就意味着我们在处理一个超定的问题，而公式（8.16）的精确解可能并不存在。实际上，我们应该找到 $\boldsymbol{k}$ 的最小二乘估计 $\boldsymbol{k}^*$，它能使误差平方 $\|\boldsymbol{A}\boldsymbol{k}-\boldsymbol{y}_d\|^2$ 最小，这可以用下面的伪逆矩阵技术实现：

$$\boldsymbol{k}^*=(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y}_d \tag{8.17}$$

其中，A^T 是 A 的转置，在 A^TA 是非奇异的条件下，$(A^TA)^{-1}A^T$ 是 A 的伪逆矩阵。

一旦规则的后项参数确定下来，我们就可以计算一个实际的网络输出向量 y，并确定误差向量 e，

281

$$e = y_d - y \tag{8.18}$$

在后向传递过程中，我们使用的是反向传播算法。误差信号向后传递，同时根据链式法则更新前项参数。

例如，要修正神经元 $A1$ 使用的钟形激活函数的参数 a，我们使用如下的链式法则：

$$\Delta a = -\alpha \frac{\partial E}{\partial a} = -\alpha \frac{\partial E}{\partial e} \times \frac{\partial e}{\partial y} \times \frac{\partial y}{\partial (\bar{\mu}_i f_i)} \times \frac{\partial (\bar{\mu}_i f_i)}{\partial \bar{\mu}_i} \times \frac{\partial \bar{\mu}_i}{\partial \mu_i} \times \frac{\partial \mu_i}{\partial \mu_{A1}} \times \frac{\partial \mu_{A1}}{\partial a} \tag{8.19}$$

其中，α 是学习速率，E 为 ANFIS 输出神经元的误差平方的瞬时值，即

$$E = \frac{1}{2}e^2 = \frac{1}{2}(y_d - y)^2 \tag{8.20}$$

因此，有

$$\Delta a = -\alpha (y_d - y)(-1) f_i \times \frac{\bar{\mu}_i (1 - \bar{\mu}_i)}{\mu_i} \times \frac{\mu_i}{\mu_{A1}} \times \frac{\partial \mu_{A1}}{\partial a} \tag{8.21}$$

或者

$$\Delta a = \alpha (y_d - y) f_i \bar{\mu}_i (1 - \bar{\mu}_i) \times \frac{1}{\mu_{A1}} \times \frac{\partial \mu_{A1}}{\partial a} \tag{8.22}$$

其中，

$$\frac{\partial \mu_{A1}}{\partial a} = -\frac{1}{\left[1 + \left(\frac{x1 - a}{c}\right)^{2b}\right]^2} \times \frac{1}{c^{2b}} \times 2b \times (x1 - a)^{2b-1} \times (-1)$$

$$= \mu_{A1}^2 \times \frac{2b}{c} \times \left(\frac{x1 - a}{c}\right)^{2b-1}$$

同样，我们可以对参数 b 和 c 进行修正。

在 Jang 提出的 ANFIS 训练算法中，前项参数和后项参数都要优化。在前向传递过程中，前项参数保持固定，调整后项参数。在后项传递中，后项参数保持固定而调整前项参数。但是，在某些情况下，输入/输出数据集比较小，隶属函数可由人类专家描述。在这样的情况下，这些隶

282

属函数在训练全过程中保持固定，只有后项参数要调整（Jang et al.，1997）。

现在我们来证明 ANFIS 在函数近似方面的应用。在本例中，ANFIS 用于理解下面公式所定义的非线性函数的轨迹。

$$y = \frac{\cos(2x1)}{e^{x2}}$$

首先选择合适的 ANFIS 框架。一个 ANFIS 必须有两个输入 $x1$ 和 $x2$，及一个输出 y。

通过选择产生“满意”性能的最小数量的隶属函数，来确定赋给每个输入的隶属函数的数量。因此，试验性的研究从赋给每个输入变量的两个隶属函数开始。

要构建 ANFIS，我们需要选择编程语言，例如 C/C++，或神经-模糊开发工具。我们选择最常见的工具——MATLAB Fuzzy Logic Toolbox，它提供了构建神经-模糊推理系统的系统框架，并能基于赋给每个输入变量的隶属函数的数量来自动定义规则。因此，在本例中，ANFIS 用 4 个规则来定义，并且实际上有如图 8.10 所示的结构。

ANFIS 训练数据包含 101 个训练例子。它用 101×3 的矩阵 $[x1\ x2\ y_d]$ 表示，其中 $x1$ 和 $x2$ 是输入向量，y_d 是所期望的输出向量。第一个输入向量 $x1$ 从 0 开始，每次递增 0.1，到 10 结束。第二个输入向量 $x2$ 用 $x1$ 的每个分量的正弦值创建。最后，期望输出向量 y_d 的每个分量由函数方程决定。

函数的实际轨迹和 ANFIS 从 1 和 100 个训练周期的输出如图 8.11 所示。注意图 8.11a 表示第一次用最小二乘法计算规则后项参数后的结果。可以看到，即使在训练了 100 个周期后，ANFIS 的性能仍不能令人满意。

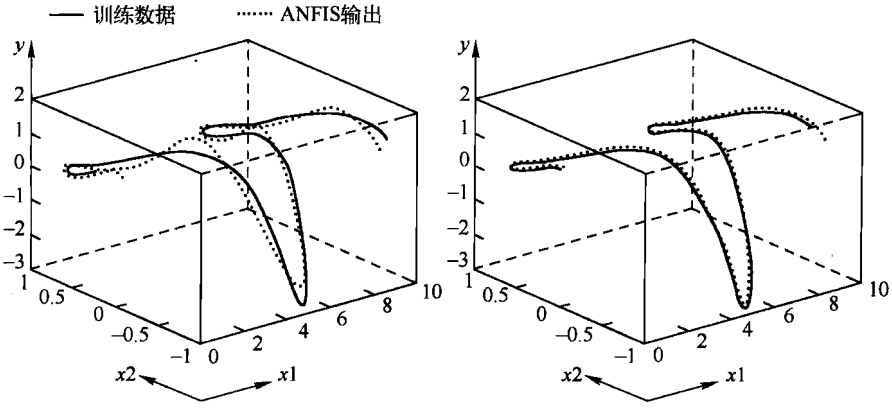


图 8.11　赋给每个输入两个隶属函数的 ANFIS 的学习

283

通过增加训练周期数，我们可以改进 ANFIS 的性能，但是把每个输入变量的隶属函数增加到 3 个可以得到更好的结果。在本例中，ANFIS 模型有 9 个规则，如图 8.12 所示。

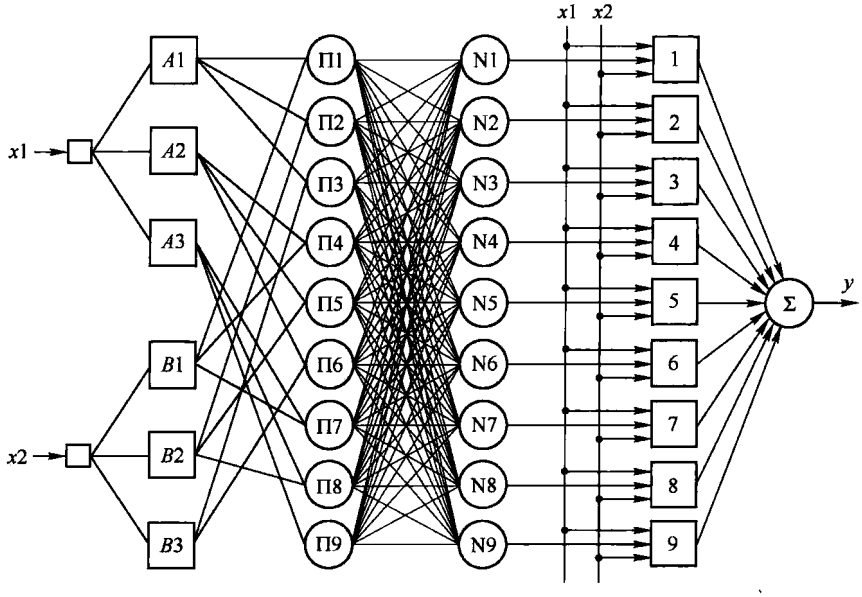


图 8.12　有 9 个规则的 ANFIS 模型

图 8.13 显示 ANFIS 的性能明显改善，即使只经过一个周期的训练，其输出曲线也和期望的轨迹精确吻合。图 8.14 显示了训练前后的隶属函数。

ANFIS 系统具有归纳和快速收敛的能力。这对在线学习是非常重要的。因此，Jang 的模型和其变体的应用非常广，尤其在自适应控制当中。

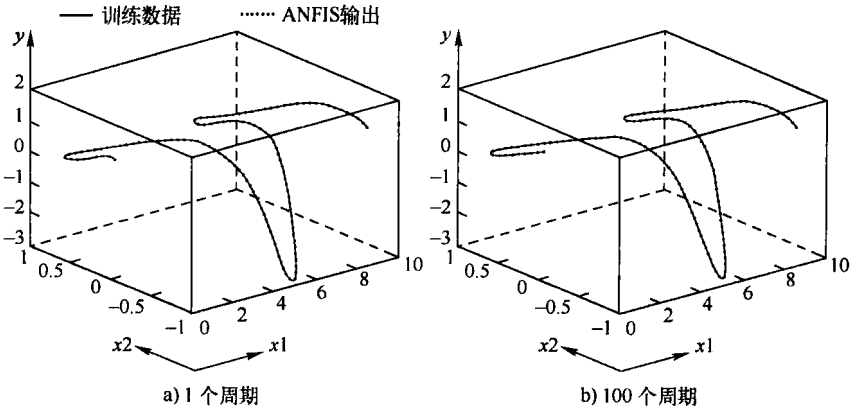


图 8.13 赋给每个输入 3 个隶属函数的 ANFIS 的学习

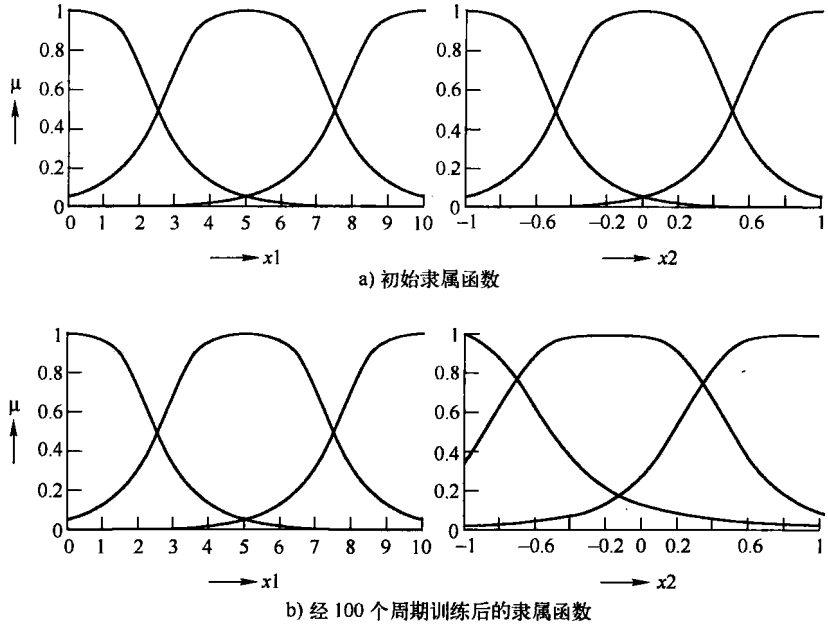


图 8.14 ANFIS 中初始时刻与结束时刻的隶属函数

8.5 进化神经网络

虽然神经网络可以用来解决各种问题，但这种方法还是有一些限制的。其中最普遍的问题是和神经网络训练有关。反向传播学习算法是经常使用的，因为这种方法灵活并且容易用数学方法处理（假设神经元的传递函数可以区分），但这种方法也有些严重的缺点：它不能保证得到最优的解决方案。在实际应用中，反向传播算法可能收敛到局部最优权重，因此，神经网络通常不能为问题找到最合理的解决方案。

另一个困难与为神经网络选择最佳拓扑有关。对于某个问题的“正确”网络架构通常由启发式的方法确定，而且设计神经网络拓扑更是一门艺术而不是工程。

遗传算法是有效的优化技术，可以指导权重优化和拓扑选择。

首先，我们考虑一下进化式权重优化技术的基本概念（Skabar，2003；Abraham，2004；Cas-tellano et al.，2007）。要使用遗传算法，首先要将问题域表示成染色体。例如，要找到如图 8.15 所示的多层反馈神经网络的权重的优化集。

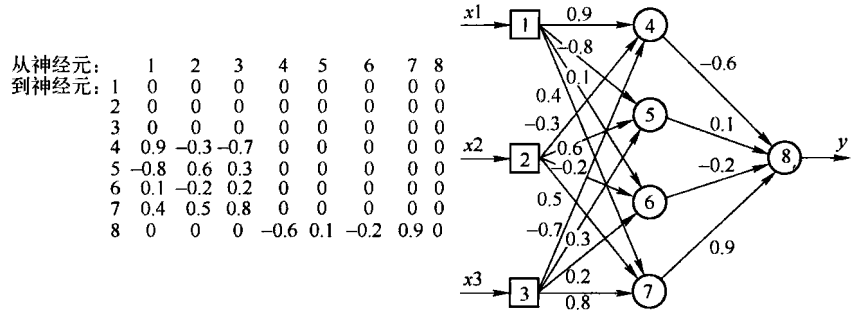


图 8.15　染色体的权重集的编码

网络初始权重是在一个很小的范围（如 [−1，1]）内随机选择的。权重集可以用正方形矩阵来表示，矩阵中的实数表示从一个神经元到另一个神经元链接的权重，0 表示这两个神经元之间没有关系。图 8.15 中的神经元之间有 16 个权重链接。由于染色体是基因的集合，因此可以用 16 位基因的染色体表示，每个基因表示网络中的一个权重链接。因此，我们将矩阵中的行串起来并忽略 0，就可以得到染色体。

另外，每一行表示一组指向一个神经元的所有输入权重链接。该组可以被看做是网络的构造块（Montana and Davis，1989），因此应该放在一起以便向下一代传递遗传物质。为了达到这个目标，就要将每个基因与给定神经元的所有输入权重，而非单个权重，对应起来，如图 8.15 所示。

第二步是定义适应性函数来评估染色体性能。该函数必须评估给定神经网络的性能。这里，我们可以使用误差平方和的倒数，一个相当简单的函数。要评估给定染色体的适应性，染色体中包含的每个权重都应指定给网络中相应的链接。然后将例子的训练集输入到网络中，并计算误差的平方和。和越小，染色体越适合。换句话说，遗传算法应尝试找到可以使误差平方和最小的权重集。

第三步是选择遗传操作——交叉和突变。交叉操作需要两个亲代染色体，并用这两个亲代染色体的遗传物质创建一个子代染色体。子代染色体的每个基因是随机从父代染色体中选择的。图 8.16 显示了交叉操作的应用。

突变操作随机选择染色体中的一个基因，并给该基因上的每个权重添加一个 −1 ~ 1 之间的随机值。图 8.16b 为突变的例子。

现在可以准备应用遗传算法了。当然，还需要定义种群大小，即不同权重的网络数量、交叉和突变的概率以及遗传的代数。前面假设网络的结构是固定的，进化学习仅用来优化给定网络的权重。但是，网络的结构（即神经元的数量以及它们之间的相互连接）通常决定了应用是成功还是失败。通常，网络架构是通过反复实验和误差来决定的。因此，针对某个应用自动设计架构有很大的需求。遗传算法能够帮助我们选择网络的架构。

寻找合适的网络架构的基本方法是在可能的架构种群中进行遗传搜索（Melin and Castillo，2005；Mendivil et al.，2008）。当然，我们必须首先选择将网络架构编码为染色体的方法。

有很多不同的方法来编码网络架构。关键是确定表达网络需要多少信息。网络架构的参数越多，计算成本就越大。这里用一个简单直接的编码方法来进行说明（Miller et al.，1989）。虽

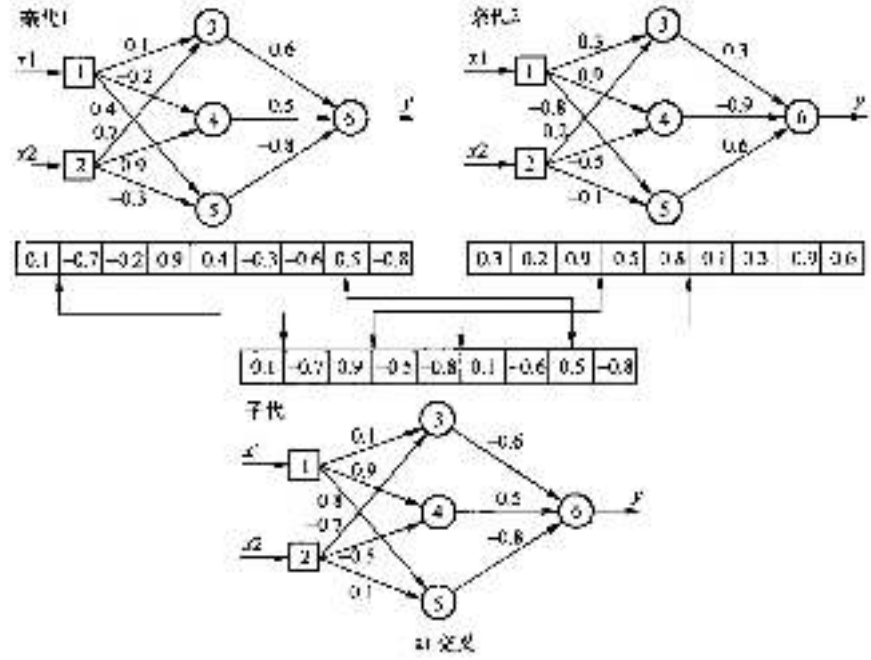


图 8.16 神经网络中权重优化的遗传操作

然直接编码方法是一种受限制的技术，它仅能用在神经元个数固定的前馈网络中，但它能说明网络拓扑结构是如何演化的。

神经网络的连接拓扑可以用正方形连通矩阵来表示，如图 8.17 所示。矩阵的每个项定义了一个神经元（列）和另一个神经元（行）之间的连接的类型，其中 0 表示没有连接，1 表示在学习中权重可变的连接。要将连通矩阵转为染色体，仅需要按行将矩阵重新连成串即可，如图 8.17 所示。

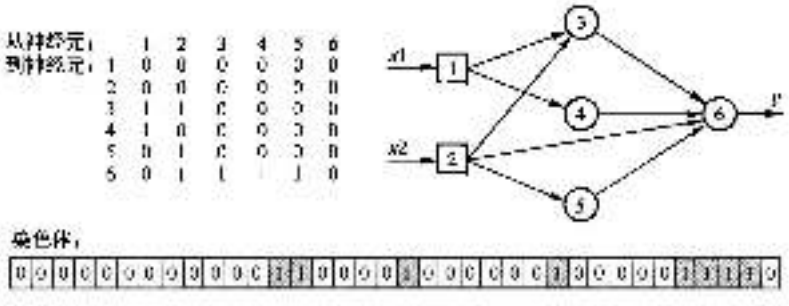


图 8.17 对网络拓扑结构的直接编码

287
288

给定一组训练实例和一个可以代表网络框架的二进制串，一个基本的遗传算法可以通过以下的步骤来描述：

步骤1：选择染色体种群的大小、交叉和突变的概率，并定义训练周期数。

步骤2：定义适应性函数来度量单个染色体的性能或适应性。通常，网络的适应性不仅取决于其精确性，还要取决于其学习速度、大小和复杂度。但是，网络性能还是比其大小更重要，因此，还是可以通过计算误差平方和的倒数来定义适应性函数。

步骤3：随机产生染色体的初始种群。

步骤4：将单个染色体解码成神经网络。由于假定为前馈网络，因此应忽略染色体中指定的所有反馈连接。用小的随机数，如[-1，1]区间的随机数来设置网络的初始权重。在指定的训练周期数下，用一组实例训练集和反向传播算法训练网络。计算误差平方和，确定网络的适应性。

步骤5：重复步骤4，直到群体中所有的染色体都被考虑为止。

步骤6：选择用于配对的一对染色体。选中的概率与其适应性成正比。

步骤7：用基因的交叉和突变操作创建一对子代染色体。交叉操作随机选择一个行索引，简单地交换双亲染色体中相应的行，并创建两个子代染色体。突变操作用很低的概率（如0.005）反转染色体中的1或2位（bit）。

步骤8：将子代染色体放入新的种群中。

步骤9：重复步骤6，直到新的染色体种群的大小和初始种群的大小一致为止，然后用新的种群（子代）取代初始染色体种群（亲代）。

步骤10：回到步骤4，重复这个过程，直到达到指定的迭代次数为止。

神经网络拓扑的进化过程如图8.18所示。

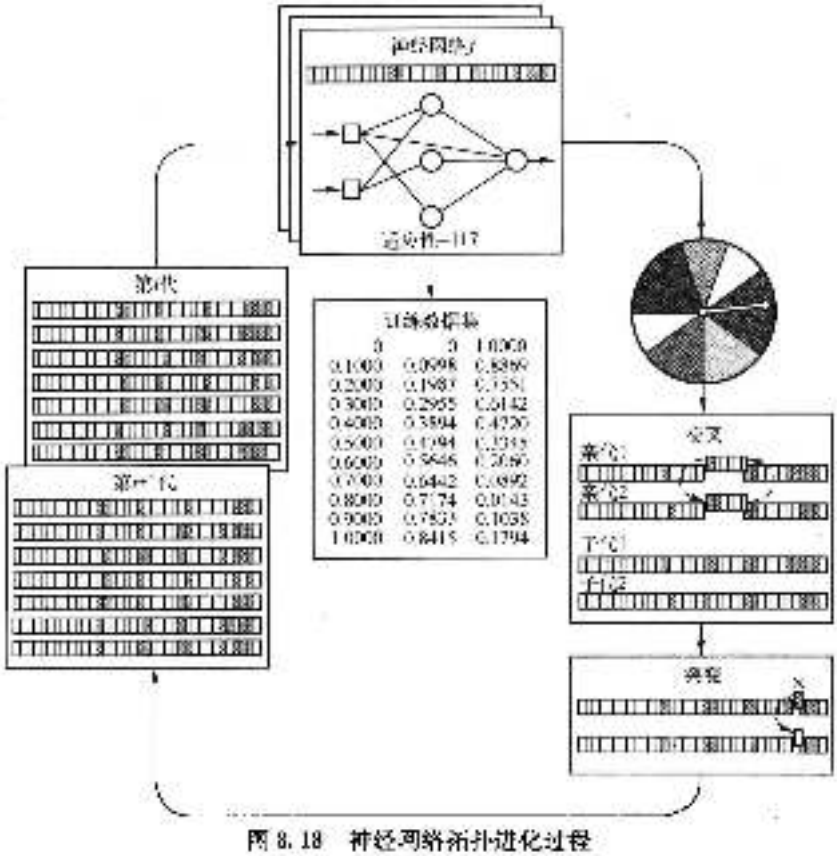


图8.18 神经网络拓扑进化过程

除了神经网络训练和拓扑选择之外，进化计算还可以用于优化传递函数和选择合适的输入变量。从具有复杂或未知函数关系的大量可能的输入变量中找到关键输入，是目前进化神经网络的一个很有潜力的研究领域。神经网络系统上新的进化计算的研究主题可以在（Affenzeller et al.，2009；Melin and Castillo，2005）中找到。

289
290

8.6 模糊进化系统

进化计算也可以用来设计模糊系统，特别是产生模糊规则和调整模糊集的隶属函数。本节主要介绍用遗传算法来为分类问题选择合适的模糊 IF－THEN 规则集（Ishibuchi et al.，1995）。

要应用遗传算法，需要有一个可行的解决方案的种群。在本例中，是模糊 IF－THEN 规则的集合。对于分类问题，模糊 IF－THEN 规则集是从数值数据中产生的（Ishibuchi et al.，1992）。首先，我们应对输入空间使用网格类型的模糊分区。

图 8.19 显示了将二维输入空间模糊分区成 3×3 的模糊子空间的例子。黑色和白色的点分别表示类 1 和类 2 的训练模式。网格类型的模糊分区可以看做是规则表。输入变量 $x1$ 的语言变量（$A1$、$A2$ 和 $A3$）形成水平轴，输入变量 $x2$ 的语言变量（$B1$、$B2$ 和 $B3$）形成垂直轴。行和列的交集就是规则后项。

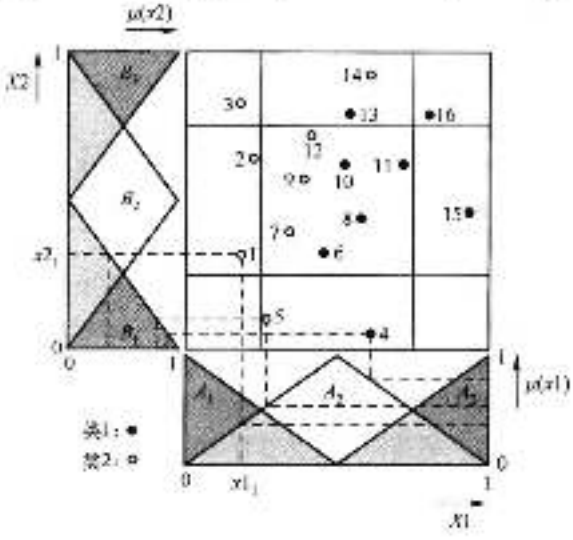


图 8.19 由 3×3 模糊网格组成的模糊分区

在规则表中，每个模糊子空间仅有一个模糊 IF－THEN 规则，因此在 $K \times K$ 个网格中生成的规则的总数也是 $K \times K$ 个。和 $K \times K$ 的模糊分区相对应的模糊规则可以被如下方法表示：

Rule R_{ij}：

IF $x1_p$ is A_i $i = 1, 2, \cdots, K$

AND $x2_p$ is B_j $j = 1, 2, \cdots, K$

THEN $x_p \in C_n$ $|CF_{A_iB_j}^{C_n}|$ $x_p = (x1_p, x2_p)$, $p = 1, 2, \cdots, P$

其中，K 是每个轴上模糊间隔的数量，x_p 是输入空间 $X1 \times X2$ 上的训练模式，P 是训练模式的数量，C_n 是规则后项（本例中是类 1 或类 2）。$CF_{A_iB_j}^{C_n}$ 是模糊子空间 A_iB_j 属于类 C_n 的确信因子或是似然值。

要确定规则后项和确信因子，可以使用下面的过程：

步骤 1：将输入空间分割成 $K \times K$ 的模糊子空间，并计算每个模糊子空间中每类训练模式的强度。

291

给定模糊子空间中的每个类用其训练模式来表示。训练模式越多，类就越强。换句话说，在给定的模糊子空间中，如果某一类模式出现的次数比别的类的模式要多，那么规则后项就越确定。模糊子空间 A_iB_j 中类 C_n 的强度定义为：

$$\beta_{A_iB_j}^{C_n} = \sum_{\substack{p=1 \\ x_p \in C_n}}^{P} \mu_{A_i}(x1_p) \times \mu_{B_j}(x2_p), \quad x_p = (x1_p, x2_p) \tag{8.23}$$

其中，$\mu_{A_i}(x1_p)$ 和 $\mu_{B_j}(x2_p)$ 分别是训练模式 x_p 在模糊集 A_i 和 B_j 中的隶属度。

例如，在图 8.19 中，模糊子空间A_2B_1中类 1 和类 2 的强度分别为：

$$\beta_{A_2B_1}^{Class1} = \mu_{A_2}(x_4) \times \mu_{B_1}(x_4) + \mu_{A_2}(x_6) \times \mu_{B_1}(x_6) + \mu_{A_2}(x_8) \times \mu_{B_1}(x_8) + \mu_{A_2}(x_{15}) \times \mu_{B_1}(x_{15})$$
$$= 0.75 \times 0.89 + 0.92 \times 0.34 + 0.87 \times 0.12 + 0.11 \times 0.09 = 1.09$$
$$\beta_{A_2B_1}^{Class2} = \mu_{A_2}(x_1) \times \mu_{B_1}(x_1) + \mu_{A_2}(x_5) \times \mu_{B_1}(x_5) + \mu_{A_2}(x_7) \times \mu_{B_1}(x_7)$$
$$= 0.42 \times 0.38 + 0.54 \times 0.81 + 0.65 \times 0.21 = 0.73$$

步骤 2：确定每个模糊子空间的规则后项和确信因子。由于规则后项是由强度最高的类来确定的，因此，我们需要找到类C_m，使其满足：

$$\beta_{A_iB_j}^{C_m} = max\left[\beta_{A_iB_j}^{C_1}, \beta_{A_iB_j}^{C_2}, \cdots, \beta_{A_iB_j}^{C_N}\right] \tag{8.24}$$

如果某个类取得了最大值，规则后项就确定为C_m。例如，在模糊子空间A_2B_1中，规则后项就是类 1。

接下来是计算确信因子：

$$CF_{A_iB_j}^{C_m} = \frac{\beta_{A_iB_j}^{C_m} - \beta_{A_iB_j}}{\sum_{n=1}^{N} \beta_{A_iB_j}^{C_n}} \tag{8.25}$$

其中

$$\beta_{A_iB_j} = \frac{\sum_{\substack{n=1 \\ n \neq m}}^{N} \beta_{A_iB_j}^{C_n}}{N-1} \tag{8.26}$$

例如，在模糊子空间A_2B_1中，相应的规则后项的确信因子的计算方法为：

$$CF_{A_2B_1}^{Class2} = \frac{1.09 - 0.73}{1.09 + 0.73} = 0.20$$

如何解释这里的确信因子

下面解释公式（8.25）中的确信因子。如果模糊子空间A_iB_j的所有训练模式都属于相同的类C_m，则确信因子最大，且该子空间中任何新的模式都是属于类C_m的。但是，如果训练模式属于不同的类且这些类的强度相同，则确信因子最小且不能确定新模式属于类C_m。

这就意味着模糊子空间A_2B_1中的模式易导致误分类。如果模糊子空间没有任何训练模式，就根本不能确定规则后项。实际上，如果模糊分区太粗糙，则很多模式就会被误分类。另外，如果模糊分区非常精细，则很多模糊规则就不可能得到，因为缺乏相应模糊子空间的训练集。因此，选择模糊网格的密度对于将输入模式正确分类是至关重要的。

同时，如图 8.19 所示，训练模式没有必要均匀地分布于输入空间中。因此，很难为模糊网格选择合适的密度。为了克服这个难点，可以使用多个模糊规则表（Ishibuchi et al.，1992）。图 8.20 给出了一个例子，表的数量取决于分类问题的复杂程度。

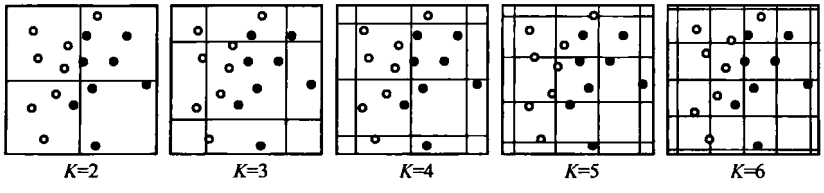


图 8.20 多层模糊规则表

多个模糊规则表的每个模糊子空间产生模糊 IF－THEN 规则，因此完整的规则集可表示为：

$$S_{ALL} = \sum_{K=2}^{L} S_K, K = 2,3,\cdots,L \tag{8.27}$$

其中，S_K是模糊规则表 K 对应的规则集。

如图 8.20 所示，多个模糊规则表产生的规则集S_{ALL}包含$2^2+3^2+4^2+5^2+6^2=90$ 条规则。

一旦产生了S_{ALL}规则集，就可以按下面的步骤给新的模式 $x=(x1,x2)$分类：

步骤 1：在多个模糊规则表的每个模糊子空间中，计算每个类的新模式的兼容程度。

292
~
293

$$\alpha^{C_n}_{K|A_iB_j|}=\mu_{K|A_i|}(x1)\times\mu_{K|B_j|}(x2)\times CF^{C_n}_{K|A_iB_j|}$$
$$n=1,2,\cdots,N;\ K=2,3,\cdots,L;\ i=1,2,\cdots,K;\ j=1,2,\cdots,K \tag{8.28}$$

步骤 2：定义每个类的新模式的最大兼容程度。

$$\begin{aligned}\alpha^{C_n}=max\ [&\alpha^{C_n}_{1|A_1B_1|},\ \alpha^{C_n}_{1|A_1B_2|},\ \alpha^{C_n}_{1|A_2B_1|},\ \alpha^{C_n}_{1|A_2B_2|},\\ &\alpha^{C_n}_{2|A_1B_1|},\ \cdots,\ \alpha^{C_n}_{2|A_1B_k|},\ \alpha^{C_n}_{2|A_2B_1|},\ \cdots,\ \alpha^{C_n}_{2|A_2B_k|},\ \cdots,\ \alpha^{C_n}_{2|A_kB_1|},\ \cdots,\ \alpha^{C_n}_{2|A_kB_k|},\ \cdots,\\ &\alpha^{C_n}_{L|A_1B_1|},\ \cdots,\ \alpha^{C_n}_{L|A_1B_k|},\ \alpha^{C_n}_{L|A_2B_1|},\ \cdots,\ \alpha^{C_n}_{L|A_2B_k|},\ \cdots,\ \alpha^{C_n}_{L|A_kB_1|},\ \cdots,\ \alpha^{C_n}_{L|A_kB_k|}\ ]\\ &n=1,\ 2,\ \cdots,\ N\end{aligned} \tag{8.29}$$

步骤 3：用有最高兼容程度的新模式定义类C_m，即

$$\alpha^{C_m}=max\ [\alpha^{C_1},\ \alpha^{C_2},\ \cdots,\ \alpha^{C_n}\ ] \tag{8.30}$$

为类C_m指定模式 $x=(x1,x2)$。

精确模式分类所需的多个模糊规则表的数目会非常大。因此，完整的规则集S_{ALL}可能非常庞大，同时S_{ALL}中的规则的分类能力也不同，因此仅需要选出精确分类潜力高的规则，这样我们可以大幅减小规则集的大小。

选择模糊 IF－THEN 规则的问题可以看做是有两个目标的复合优化问题。首先，最重要的目标是使正确分类的模式的数量达到最大，其次是使规则的数量达到最小（Ishibuchi et al.，1995）。可以应用遗传算法来解决这个问题。

在遗传算法中，每个可行的解决方案被当做一个个体，因此，我们需要将可行的模糊 IF－THEN 规则集表示成固定长度的染色体。染色体中的每个基因应该表示S_{ALL}中的一个模糊规则，假如S_{ALL}定义为，

$$S_{ALL}=2^2+3^2+4^2+5^2+6^2$$

则染色体可以表示成一个 90 位的字符串。字符串中的每一位可以是 3 个值中的一个：1，－1 或 0。

我们的目标是从规则集S_{ALL}中选出合适的规则组成模糊规则压缩集 S。如果某个特定的规则属于集合 S，那么染色体上对应的位为 1；但是如果不属于集合 S，那么这个位就应该是－1。哑规则（dummy rule）用 0 表示。

什么是哑规则

如果规则的后项不能确定，那么这个规则就是哑规则。当模糊子空间没有相应的训练模式的时候，这种情况就是比较正常的情况。哑规则不会影响分类系统的性能，因此可以排除在规则
294
集 S 以外。

如何确定模糊规则是否属于规则集 S

在初始种群中，这个决定是按 50% 的机会做出的。换句话说，在出现于初始种群的每个染色体中，每个模糊规则接收值 1 的概率为 0.5。

选择模糊 IF－THEN 规则的基本遗传算法包含以下几个步骤（Ishibuchi et al.，1995）：

步骤 1：随机产生染色体的初始种群。种群大小可以相对比较小，例如 10 或者 20 个染色体。染色体上的每个基因对应规则集S_{All}中一个特定的模糊 IF－THEN 规则。哑规则对应的基因接收的值为 0，其他基因随机被指定为 1 或者－1。

步骤 2：为当前种群中的每个染色体计算性能或者适应性。选择模糊逻辑的问题有两个目

标：使模式分类的精确度最大而规则数量最小。适应性函数必须能够适应这两个目标，可以在适应性函数中引入两个权重w_p和w_N。

$$f(S) = w_P \frac{P_s}{P_{ALL}} - w_N \frac{N_S}{N_{ALL}} \tag{8.31}$$

其中，P_s是成功分类的模式的数量，P_{ALL}是分类系统处理的总的模式的数量，N_S和N_{ALL}是分别出现在集合S和集合S_{ALL}中的模糊 IF - THEN 规则。

分类精度比规则集的大小更为重要，可以直接反映到指定的权重上，例如：

$0 < w_N \leqslant w_P$

w_N和w_P的典型的值分别是 1 和 10，因此有下列式子，

$$f(S) = 10 \frac{P_s}{P_{ALL}} - \frac{N_S}{N_{ALL}} \tag{8.32}$$

步骤 3：选择一对染色体配对。亲代染色体依据它们的适应性按照概率来选取，适应性高的染色体有更高的概率被选取。

步骤 4：用标准的交叉操作来产生一对子代染色体。亲代染色体在随机选取的交叉点上进行交换。

步骤 5：在产生的子代染色体上对每一个基因执行突变操作。突变概率一般保持得很低，如 0.01。突变操作将突变基因的值乘以 -1。 295

步骤 6：将产生的子代染色体放入到新的种群中。

步骤 7：重复步骤 3，直到新的种群的大小和初始种群的大小相同，然后用新的（子代）种群替换原来的（亲代）染色体。

步骤 8：回到步骤 2，重复这一过程，直到达到了指定的代数（通常是数百）为止。

上面的算法可以显著地减小为正确分类所需的模糊 IF - THEN 规则的数目。实际上，一些计算机模拟（Ishibuchi et al，1995）证明了规则数量可以减小到初始生成的规则数量的 2%，这样的删减给模糊分类系统留下了少数几条非常重要的规则。这些规则可以由人类专家来仔细验证。这使得我们可以使用模糊进化系统作为知识获取的工具，以便在复杂数据库中发现新知识。

8.7 小结

本章中，我们考察了综合多种不同智能技术的混合智能系统。首先，我们介绍了一种新型的专家系统，即神经专家系统，它综合了神经网络和基于规则的专家系统。然后，我们考察了一个功能上等同于 Mamdani 模糊推理模型的神经 - 模糊系统，以及一个功能上等同于 Sugeno 模糊推理模型的自适应神经 - 模糊推理系统（ANFIS）。最后，我们介绍了进化神经网络和模糊进化系统。

本章的主要内容有：

- 混合智能系统是至少结合两种智能技术的系统。例如，结合神经网络和模糊系统就产生了混合神经 - 模糊系统。
- 概率推理、模糊集理论、神经网络和进化计算组成了软计算的核心。软计算是构建在不确定和不精确环境中能进行推理和学习的混合系统的方法。
- 专家系统和神经网络都试图模拟人类的智能，但使用了不同的方法。专家系统依赖 IF - THEN 规则和逻辑推理，神经网络使用并行数据处理。专家系统不能学习，但能够解释其推理，而神经网络能学习，但它的行为像是一个黑盒。这些特点使它们能很好地构建混合智能系统，称为神经专家系统或连通式专家系统。
- 神经专家系统使用经过训练的神经网络代替知识库。和传统的基于规则的专家系统不同，

296

神经专家系统可以处理噪声数据和不完整数据。领域知识可以被用来构建初始的神经知识库。经过训练，神经知识库就可以表达成 IF - THEN 产生式规则的集合。

- 对应于 Mamdani 模糊推理模型的神经 - 模糊系统，可以用 5 层的前馈神经网络来表示，这 5 层包括：输入层、模糊化层、模糊规则层、输出成员层和去模糊化层。
- 神经 - 模糊系统可以使用神经网络上标准的学习算法，包括反向传播算法。专家的知识表示成语言变量和模糊规则，这些可以被包含进神经 - 模糊系统的结构中。有了一组有代表性的可用例子之后，神经 - 模糊系统可以自动将它转化成模糊 IF - THEN 规则。
- 自适应神经 - 模糊推理系统（ANFIS）对应于一阶 Sugeno 模糊模型。ANFIS 可以表达成包含 6 个层的神经网络，分别是：输入层、模糊化层、模糊规则层、归一化层、去模糊化层和总结层。
- ANFIS 使用了混合最小二乘估算法和梯度下降法的学习算法。在向前传递的过程中会出现一组训练用的输入模式，神经元的输出会一层一层地被计算，规则后项参数用最小二乘估算法表示。在向后传送中，误差信号后向传递，规则前项参数根据链式法则更新。
- 遗传算法可以有效地优化权重和神经网络的拓扑结构。
- 进化计算可以被用在复杂的分类问题中以选择合适的模糊规则集。当使用多个模糊规则表根据数值数据产生完整的模糊 IF - THEN 规则后，可以用遗传算法选择数量相对较少但分类能力高的模糊规则。

复习题

1. 什么是混合智能系统？请给出一个例子。什么构成了软计算的核心？“硬”计算和“软”计算的区别是什么？
2. 为什么神经专家系统能够近似推理？画出一个三分类问题的神经网络知识库，假设一个对象可以被分类为苹果，或者橘子，或者柠檬。
3. 为什么模糊系统和神经网络被认为是在构建混合智能系统时本质上互补的两种工具？画出一个对应于 Sugeno 模糊推理模型，实现 AND 操作的神经 - 模糊系统。假设系统有两个输入和一个输出，每个变量用两个语言变量值表示：small 和 large。

297

4. 描述神经 - 模糊系统中每一层的功能。模糊化在系统中是怎么做的？一个模糊规则神经元怎样将它的多个输入合并？去模糊化在神经 - 模糊系统中是怎么做的？
5. 神经 - 模糊系统如何学习？系统中的参数是怎样学习或者是调整的？神经 - 模糊系统怎样识别人类专家给出的错误规则的？请举一个例子。
6. 描述 ANFIS 中每一层的功能。在 Jang 的模型中，模糊化神经元用的是什么样的激活函数？什么是模糊规则激活强度的归一化？
7. ANFIS 如何学习？描述它采用的混合的学习算法。这个算法的优点是什么？
8. 如果我们想要实现一个零阶的 Sugeno 模糊模型，应该怎样改变图 8.10 中所示的 ANFIS 结构？
9. 对应于 Mamdani 模糊推理模型的神经 - 模糊系统和 ANFIS 的区别是什么？
10. 神经网络的一组权重应该怎样编码进染色体？请举个例子。描述优化神经网络权重所用到的遗传操作。
11. 神经网络的拓扑结构如何编码进染色体？请举个例子。写出用于优化神经网络拓扑结构的基本遗传算法的主要步骤。
12. 什么是网格模糊分区？请举个例子。为什么复杂的模式分类问题中需要多个模糊规则表？描述一个用于选择模糊 IF - THEN 规则的遗传算法。

参考文献

Abraham, A. (2004). Meta learning evolutionary artificial neural networks, *Neurocomputing*, 56, 1–38.

Affenzeller, M., Winkler, S., Wagner, S. and Beham, A. (2009). *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*. Chapman & Hall/CRC Press, Boca Raton, FL.

Castellano, G., Castiello, C., Fanelli, A.M. and Jain, L. (2007). Evolutionary neuro-fuzzy systems and applications, *Advances in Evolutionary Computing for System Design*, L.C. Jain, V. Palade and D. Srinivasan, eds, Springer-Verlag, Berlin, pp. 11–46.

Fu, L.M. (1993). Knowledge-based connectionism for revising domain theories, *IEEE Transactions on Systems, Man, and Cybernetics*, 23(1), 173–182.

Gallant, S.I. (1988). Connectionist expert systems, *Communications of the ACM*, 31(2), 152–169.

Gallant, S.I. (1993). *Neural Network Learning and Expert Systems*. MIT Press, Cambridge, MA.

Ishibuchi, H., Nozaki, K. and Tanaka, H. (1992). Distributed representation of fuzzy rules and its application to pattern classification, *IEEE Transactions on Fuzzy Systems*, 3(3), 260–270.

298

Ishibuchi, H., Nozaki, K., Yamamoto, N. and Tanaka, H. (1995). Selecting fuzzy If-Then rules for classification problems using genetic algorithms, *Fuzzy Sets and Systems*, 52, 21–32.

Jang, J.-S.R. (1993). ANFIS: Adaptive Network-based Fuzzy Inference Systems, *IEEE Transactions on Systems, Man, and Cybernetics*, 23(3), 665–685.

Jang, J.-S.R., Sun, C.-T. and Mizutani, E. (1997). *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice Hall, Englewood Cliffs, NJ.

Kasabov, N. (2007). *Evolving Connectionist Systems: The Knowledge Engineering Approach*. Springer-Verlag, Berlin.

Lin, C.T. and Lee, G. (1996). *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Prentice Hall, Englewood Cliffs, NJ.

Melin, P. and Castillo, O. (2005). *Hybrid Intelligent Systems for Pattern Recognition Using Soft Computing: An Evolutionary Approach for Neural Networks and Fuzzy Systems*. Springer-Verlag, Berlin.

Mendivil, S.G., Castillo, O. and Melin, P. (2008). Optimization of artificial neural network architectures for time series prediction using parallel genetic algorithms, *Soft Computing for Hybrid Intelligent Systems*, O. Castillo, P. Melin J. Kacprzyk and W. Pedrycz, eds, Springer-Verlag, Berlin, pp. 387–400.

Miller, G.F., Todd, P.M. and Hedge, S.U. (1989). Designing neural networks using genetic algorithms, *Proceedings of the Third International Conference on Genetic Algorithms*, J.D. Schaffer, ed., Morgan Kaufmann, San Mateo, CA, pp. 379–384.

Montana, D.J. and Davis, L. (1989). Training feedforward networks using genetic algorithms, *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, San Mateo, CA, pp. 762–767.

Nauck, D., Klawonn, F. and Kruse, R. (1997). *Foundations of Neuro-Fuzzy Systems*. John Wiley, Chichester.

Nikolopoulos, C. (1997). *Expert Systems: Introduction to First and Second Generation and Hybrid Knowledge Based Systems*. Marcel Dekker, New York.

Russell, S.J. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*, 3rd edn. Prentice Hall, Upper Saddle River, NJ.

Sestito, S. and Dillon, T. (1991). Using single layered neural networks for the extraction of conjunctive rules, *Journal of Applied Intelligence*, 1, 157–173.

Skabar, A. (2003). A GA-based neural network weight optimization technique for semi-supervised classifier learning, *Design and Application of Hybrid Intelligent Systems*, A. Abraham, M. Koppen and K. Franke, eds, IOS Press, Amsterdam, pp. 139–146.

Von Altrock, C. (1997). *Fuzzy Logic and NeuroFuzzy Applications in Business and Finance*. Prentice Hall, Upper Saddle River, NJ.

Zadeh, L. (1996). Computing with words – a paradigm shift, *Proceedings of the First International Conference on Fuzzy Logic and Management of Complexity*, Sydney, Australia, 15–18 January, vol. 1, pp. 3–10.

299

第9章
Artificial Intelligence, 3E

知识工程

本章讨论如何选择正确的工具和构建一个智能系统。

9.1 知识工程简介

选择正确的工具对于构建智能系统而言无疑是最关键的部分。到目前为止，读者已经熟悉了基于规则和基于框架的专家系统、模糊系统、人工神经系统、遗传算法、混合神经模糊－系统以及模糊进化系统。利用这些工具虽然可以很好地处理很多问题，但针对特定问题选择最适合的工具仍然很困难。戴维斯法则（Davis's law）提到："每个工具都有其最适合的任务"（Davis and King, 1977）。但是，假设每一个问题都会有一个完美适合于它的工具则未免太过于乐观。在本章中，我们将讨论为给定的任务选择合适工具的基本原则，考虑构建智能系统的主要步骤并讨论如何将数据转化为知识。

智能系统的构建过程从理解问题域开始。我们首先要评估问题，确定可用的数据及解决问题需要什么信息。一旦理解了问题，就可以选择合适的工具并利用这个工具开发系统了。构建基于知识的智能系统的过程为*知识工程*（knowledge engineering）。它有6个基本步骤（Waterman, 1986; Durkin, 1994）：

1）问题评估。
2）数据和知识获取。
3）原型系统开发。
4）完整系统开发。
5）系统评价和修订。
6）系统集成和维护。

知识工程的过程如图9.1所示。知识工程，尽管名为工程，但它更像一门艺术，并且开发智能系统的实际过程不可能像图9.1那样清晰和整洁。虽然每个阶段是按顺序显示的，但实际可能通常是重叠的。由于知识工程的过程本身高度迭代，因此任何时候我们都可能参与到任何开发活动中。下面我们详细说明知识工程的每个阶段。

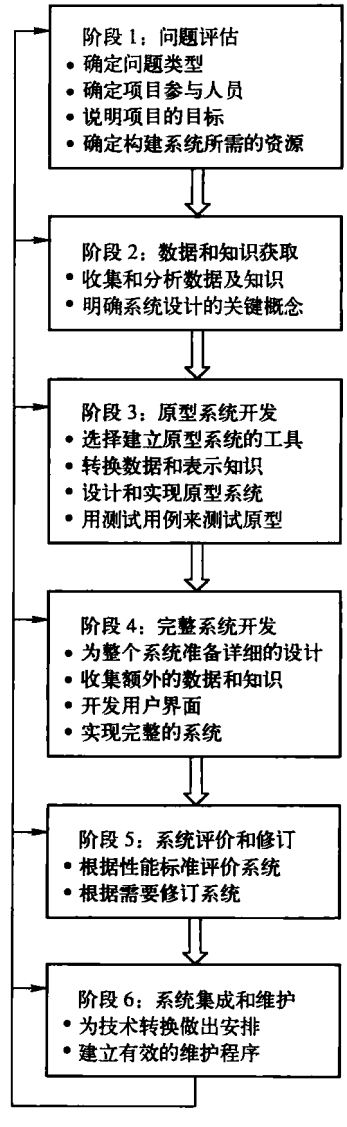


图9.1 知识工程的过程

9.1.1 问题评估

在本阶段中，我们要确定问题的特征，确定项目参与人员，详细说明项目的目标并确定构建系统所需要的资源。

为了刻画问题，我们需要确定问题类型、输入和输出

变量及其交互以及解决方案的形式和内容。

第一步是确定问题类型。智能系统要处理的典型问题如表9.1所示。它们包含诊断、选择、预测、分类、聚类、优化和控制。

表9.1 智能系统要处理的典型问题

问题类型	描 述
诊断	从对象的行为来推理故障和提出解决方案
选择	从可能的选项中提出最好的选项
预测	根据目标过去的行为预测它将来的行为
分类	将某个对象指定到定义好的类
聚类	把不同类的对象分成同类的子集
优化	提高解决方案的质量直到找到优化方案
控制	控制目标的行为来满足实时的需求

构建智能系统时，问题类型影响着我们对工具的选择。例如，假设要开发一个检测电路故障，并在诊断过程中引导用户的系统。很明显这个问题属于诊断问题。这类问题的领域知识通常用产生式规则表示，因此最好选择基于规则的专家系统。

当然，构建工具的选择还取决于解决方案的形式和内容。例如，用于诊断的系统通常需要解释设备——使系统能够证明其解决方案的手段。解释设备是任何专家系统所必需的组件，但在神经网络中就无法使用。另一方面，神经网络对于分类和聚类等结果比理解系统的推理过程更加重要的问题来说，是一个比较好的选择。

问题评估的下一步是确定项目参与人员。任何知识工程都有两个关键参与人员，知识工程师（可以设计、构建和测试智能系统的人）和领域专家（在特定的范围和领域中，有能力解决问题的知识渊博的人）。

302 ~ 303

然后需要详细说明项目的目标。例如取得竞争力优势、改善决策质量、减少人工成本、改进产品和服务的质量等。

最后，确定构建系统所需的资源。这些资源通常包含计算机设备、开发软件、知识和数据源（人类专家、教科书、手册、网址、数据库和例子），当然还有资金。

9.1.2 数据和知识获取

本阶段通过收集、分析数据和知识，获取对问题域的进一步了解，并使系统设计的关键概念更加明确。

智能系统的数据常常通过不同的途径收集，因此数据类型也会不同。然而，用以构建智能系统的特定工具需要特定类型的数据。有些工具可以处理连续变量，而有些工具则需要将变量分割到一些区域中，或者被标准化到某个小的范围中，如0～1中。有些工具可以处理字符（文本）数据，而另一些工具只能处理数值型数据。有些工具可以容忍不精确和有噪声的数据，而另一些工具则要求良好定义、整洁的数据。因此，这些数据必须转换或重新梳理为能被特殊工具使用的数据。但是，无论我们选择什么工具，在转换数据之前，我们必须先解决3个重要问题(Berry and Linoff, 2004)。

第一个问题是数据不兼容性。通常我们所要分析的数据使用EBCDIC编码存储文本和压缩十进制形式存储数值，但我们用来构建智能系统的工具要求使用ASCII编码存储文本和单精度或双精度浮点数形式存储数值。这个问题通常是通过使用数据转换工具，自动生成数据转换所要求

的编码来解决的。

第二个问题是数据不一致性。在通常情况下，相同的内容在不同的数据库中可能表示为不同的形式。如果这些不同不能被及时发现和解决，我们可能发现诸如在分析碳酸饮料的销售模式时居然没有包含可口可乐，仅仅是因为它存储于另一个独立的数据库中之类的问题。

第三个问题是数据缺失。实际的数据记录通常包含空白字段。有时我们可能会丢弃这些不完整的记录，但通常我们希望从这些不完整的数据中能够推断出一些有用的信息。很多情况下，我们可以简单地用最普通或平均数值来填充空白字段。在另一些情况下，那些没有被填充的空白字段本身其实也能提供给我们一些有用的信息。例如，在职位申请表单中，商务电话号码栏的
304 空白可能表示这个申请人尚未被录用。

系统构建工具的选择还依赖于已获得的数据。例如，我们可以考虑一个根据房地产的特征来评估其市场价值的问题。这个问题可以使用专家系统或者神经网络技术来解决。因此，在选择使用工具之前，我们首先应该调查现有数据。如果可以获得整个地区当前的房屋销售价格，我们就可以优先使用前期销量数据构建神经网络而不是使用有经验的鉴定者的知识开发一个专家系统来解决该问题。

数据获取任务和知识获取任务是密切相关的。实际上，我们在收集数据的同时也获取了一些关于问题域的知识。

知识获取过程有哪些阶段

通常我们是从查阅文档，阅读与问题域相关的书籍、论文和手册开始的。熟悉了问题之后，我们可以通过采访领域专家来收集更多的知识。然后，我们研究和分析已获得的知识，并且重复整个过程。知识获取本质上是一个迭代的过程。

在很多的采访过程中，专家会被要求提供4～5个经典案例，用以描述他个人如何解决每个案例，并且仔细思考每个解决方案背后的推理过程（Russell and Norvig，2009）。但是，从人类专家抽取信息是一个困难的过程，这通常被称为“知识获取瓶颈”。绝大多数情况下，专家很少意识到他们拥有什么样的知识和他们用什么样的策略方法解决问题，或者无法将其表达出来。专家还可能提供一些不相关、不完全和不一致的信息。

理解问题域是构建智能系统的关键。Donald Michie（1982）给出了一个经典的例子。奶酪工厂有一位将要退休的经验丰富的检查员。工厂经理打算用一台“智能机器”来接替该检查员的工作。检查员检查奶酪时，会将手指按在奶酪上来检查是否“感觉良好”。因此，我们假设机器也要通过同样的方法——检测奶酪的表面张力。这样一来，这台机器实际上是无用的。最后，他们发现检查员实际上是依靠奶酪的味道而不是奶酪的表面张力来判断奶酪好坏的，他把手指放在奶酪上的目的是为了戳破表面让味道飘出来。

知识工程第二阶段所获得的数据和知识使我们可以在最抽象的、概念化的层次来描述解决问题的策略，并选择一个工具来构建原型系统。但是，在评价原型系统之前，最好不要对问题进行详细的分析。

9.1.3 原型系统开发

这个过程实际包含了创造一个智能系统，更精确地来说是指小型智能系统，并用一些测试
305 用例进行测试。

什么是原型系统

原型系统定义为最终系统的缩小版本。它用来测试我们对问题的理解程度，即确保完成该任务的问题解决策略、构建系统工具选择、展示已获得数据技术和知识是正确的、足够的。很多

情况下，它也提供了一个说服持有怀疑态度的人的机会，也提供了在系统发展中积极鼓励雇佣更多领域专家的机会。

在选择好工具、转换好数据、获取适合该工具的数据后，我们就可以设计并实施系统的原型版本了。一旦构建好系统，我们应通过各测试用例进行测试（通常是和领域专家一起）检查原型的性能。在测试系统时，领域专家起到了积极作用，因此他们会更多地参与到系统的开发之中。

什么是测试用例

测试用例是指在过去已经成功解决，并且输入数据和输出结果已知的问题。在测试中，使用相同的输入数据，并将系统输出数据与原有测试用例作对比。

如果我们选错了系统构建工具，该怎么办

我们应该将该原型系统丢弃并从原型阶段重新开始，任何把选择的错误工具强加给其不适合解决的问题的尝试都会导致系统开发延误更长的时间。原型阶段的主要目标是更好地理解问题，因此，使用新工具重新开始，并没有浪费时间与资金。

9.1.4 完整系统开发

一旦对原型系统的功能感到满意，我们就可以确定开发一个完整系统实际所需要的内容。首先要为完整系统开发制订计划、日程安排和资金预算，并明确定义系统性能的标准。

这个阶段的主要工作通常与向系统添加数据和知识相关。例如，如果我们开发了一个诊断系统，就需要提供处理某种情况的更多的规则；如果要开发预测系统，我们需要收集额外的历史数据使预测更加准确。

接下来的任务是开发用户界面——向用户传达信息的方式。用户界面应该足够简单，使用户容易地获取到想要的信息。有些系统可能需要解释其推理过程、证明其建议、分析和结论的正确性，有些系统则需要用图表来展示结果。

智能系统的开发过程实际上是一个进化的过程。在项目进行过程中，收集新的知识并添加到系统后，系统的性能将得到提升，原型系统逐渐演化为最终系统。 306

9.1.5 系统评价和修订

与传统的计算机程序不同，智能系统主要用来解决那些没有明确定义“正确”和“错误”解决方案的问题。评价智能系统实际上是确保系统执行了让用户满意的预订任务。一个正式的系统评价往往是完成用户选择的测试用例。系统的性能将会与原型系统开发的最后阶段所达成一致的性能标准进行比较。

评价阶段往往会发现系统的局限和缺点，因此需要重复相关的开发阶段以使系统得到修订。

9.1.6 系统集成和维护

这是系统开发的最后一个阶段。该阶段包括将系统集成到将要实际运行的环境中，并建立一个有效的维护程序。

这里“集成”的意思是将新的智能系统与组织内部已有的系统进行连接，并安排技术转换。我们必须保证用户知道如何来使用和维护系统。智能系统是基于知识的系统，而知识是随着时间在演化的，因此必须保证系统能够进行修改。

系统由谁来维护

一旦系统集成到工作环境中，知识工程师就会离开项目，系统便交由用户使用。因此，使用

系统的组织应该安排一位内部的专家来负责维护和修改系统。

应该使用什么工具

到目前为止，我们应该明确的是并不存在一个适合所有任务的工具。专家系统、神经网络、模糊系统和遗传算法各有其用武之地，也都有很多的应用。仅仅是 20 年前，为了应用智能系统（更准确地说是专家系统），人们必须首先找到一个“好”问题，即有可能成功的问题。知识工程平台昂贵、动力只提供风险高，开发一个中等规模的专家系统的开销往往在 250 000 美元到 500 000 美元不等（Simon，1987）。经典的专家系统像 DENDRAL 和 MYCIN 花费 20－40 人年才完成。幸运的是，近几年来这种状况得到了翻天覆地的变化。如今，大多数的智能系统都能在数月（而不是数年）内就可以完成。我们可以使用商业化的专家系统框架、模糊系统、神经网络
307 和进化计算工具包，在标准的 PC 上运行应用。最重要的是，采用新的智能技术将会变为以问题为驱动，而不是像过去那样以好奇心为驱动。

在接下来的几节中，我们将讨论解决特定问题的不同工具的应用。

9.2 专家系统可以解决的问题

案例 1：诊断专家系统

我想要开发一个能够帮助我解决 Mac 电脑故障的智能系统。专家系统可以解决这样的问题吗？

在选择专家系统最初的候选问题时，有一个非常有用的测试方法，称为电话规则（Firebaugh，1988）。电话规则的内容是“10－30 分钟之内跟内部专家通电话能够解决的任何问题都可以开发成专家系统。”

诊断和故障排除问题（计算机诊断也是其中之一）始终是专家系统技术擅长解决的问题。你可能已经知道，医疗诊断是专家系统最早的应用领域之一。从那以后，诊断专家系统又有了很多新的应用，特别是在工程和制造业领域。

诊断专家系统相对来说是比较容易进行开发的。大多数的诊断问题有一个有限的解决方案的列表，并且只涉及很少量的良好形式化的知识，人类专家通常只需要很少的时间（例如一个小时）就能解决这样的问题。

要开发一个计算机诊断系统，我们需要获取计算机故障排除的知识。我们可以寻找并咨询硬件专家，但是对于一个小型的专家系统而言，使用故障排除手册也是较好的方法。故障排除手册提供了排除各种故障的步骤。实际上，手册中包含的知识非常简练，几乎可以直接用在专家系统上，因此我们可以不咨询相关专家，这也就避免了“知识获取的瓶颈”。

计算机手册通常包括故障排除部分，它考虑了在系统启动、计算机/外围设备（硬盘、键盘、显示器、打印机）、磁盘驱动器（软盘、光盘）、文件、网络和文件共享中存在的问题。本案例中，我们主要考虑 Mac 系统的启动问题。然而，一旦专家系统原型开发完成，就可以轻松扩展专家系统。

图 9.2 显示了 Macintosh 计算机的故障排除过程。可以看到，此处故障是通过一系列可视的检查或测试来发现的。首先收集键和（系统尚未启动）的信息，根据其作出推断，然
308 后收集其他的信息（电源良好、电线没有问题），最后确定导致故障的原因。在本质上，这个过程是数据驱动的推理，而推理是通过前向链接的推理技术来实现。专家系统首先应该要求用户来选择某个任务，一旦任务确定，系统就可以向用户询问其他的信息，直至找到问题。

阶段1：系统启动	
问题	动作
1.1.系统不启动	• 检查电源线(以及电源未端) • 检查电源板 • 检查显示器亮度 • 检查电话接头 • 检查键盘连接线 • 检查引线连接
1.2.系统启动然后冻结	• 重启Mac • 解开所有的SCSI设备，重启Mac • 按下Shift键后重启（关掉外围设备） • 移除所有的外设，然后再一个一个接上去。每接上一个外设重启一次Mac。如果使用的是System 7.5或更高版本，应使用外设管理器
1.3.带有问题标志的系统启动	• 重启Mac • 解开所有的SCSI设备，重启Mac • 按下Shift键后重启（关掉外围设备） • 杀死PRAM。如果起作用，分别开启虚拟内存和告诉缓存（每次重启计算机），尝试告诉缓存变小或者降低虚拟内存 • 重启磁盘工具，如果磁盘驱动器图标出现，检查是否存在两个系统管理工具和（或者）两个定位程序。如果磁盘驱动器图标没有出现，则调用AV修复 • 运行磁盘急救、MacCheck，或者诺顿的磁盘医生工具 • 重新安装系统
1.4.带有Sad Mac的系统启动	• 启动磁盘工具，重新安装系统 • 运行磁盘急救，如果磁盘驱动器图标没有出现，则调用AV修复磁盘医生工具
1.5.系统启动时带有错误的“音乐”	• 检查电缆

图9.2 Macintosh计算机系统启动的故障排除

下面我们来开发一个通用的规则结构。在每一条规则中，要包含一个子句来标识当前的任务。由于我们的原型系统局限在Mac系统的启动问题上，所有规则的第一个子句都标识为这个任务。例如： 309

```
Rule: 1
if      task is 'system start-up'
then    ask problem

Rule: 2
if      task is 'system start-up'
and     problem is 'system does not start'
then    ask 'test power cords'

Rule: 3
if      task is 'system start-up'
and     problem is 'system does not start'
and     'test power cords' is ok
then    ask 'test Powerstrip'
```

其他的规则都跟在这个结构的后面。图9.3显示了在Mac系统尚未启动（使用Leonardo码）情况下进行故障排除的一些规则。

```
/* Mac 故障排除专家系统

询问任务

/************************************************
/* 小节 1. 系统启动
/************************************************


Rule:1
if    task is 'system start-up'
then ask problem

/************************************************
/* 小节 1.1 系统未启动
/************************************************


Rule:1.1
if    task is 'system start-up'
and  problem is 'system does not start'
then ask 'test power cords'

Rule:1.2
if    task is 'system start-up'
and  problem is 'system does not start'
and  'test power cords' is ok
then ask 'test Powerstrip'

Rule:1.3
if    task is 'system start-up'
and  problem is 'system does not start'
and  'test power cords' is not ok
then troubleshooting is done

Rule:1.4
if    task is 'system start-up'
and  problem is 'system does not start'
and  'test Powerstrip' is ok
then ask 'test screen brightness'

Rule:1.5
if    task is 'system start-up'
and  problem is 'system does not start'
and  'test Powerstrip' is not ok
then troubleshooting is done

Rule:1.6
if    task is 'system start-up'
and  problem is 'system does not start'
and  'test screen brightness' is ok
then ask 'test phonet connectors'

Rule:1.7
if    task is 'system start-up'
and  problem is 'system does not start'
and  'test screen brightness' is not ok
then troubleshooting is done

Rule:1.8
if    task is 'system start-up'
and  problem is 'system does not start'
and  test 'phonet connectors' is ok
then ask 'test keyboard connectors'

Rule:1.9
if    task is 'system start-up'
and  problem is 'system does not start'
and  'test phonet connectors' is not ok
then troubleshooting is done

Rule:1.10
if    task is 'system start-up'
and  problem is 'system does not start'
and  'test keyboard connectors' is ok
then ask 'test pin connectors'

Rule:1.11
if    task is 'system start-up'
and  problem is 'system does not start'
and  'test keyboard connectors' is not ok
then troubleshooting is done

Rule:1.12
if    task is 'system start-up'
and  problem is 'system does not start'
and  'test pins connectors' is ok
then troubleshooting is 'Call AV Repair'

/************************************************
/* SEEK 指令创建目标

查找故障排除
```

图9.3 Mac故障排除专家系统原型的规则

接下来我们开始构建原型系统，也即利用一个专家系统开发工具实现最初的一些规则。

如何选择专家系统开发工具

通常，我们应该使工具的功能和问题的特征相匹配。这些工具不仅包括高级编程语言，如LISP、PROLOG、OPS、C和Java，也包括专家系统框架。高级编程语言提供了更大的灵活性使得其可以满足任意项目需求，但是同时也要求很高的编程能力。另一方面，框架虽然没有高级编程语言的高灵活性，但是它提供了内置的推理引擎、解释工具和用户界面。使用框架不需要任何编程能力，只需要将自然语言（英语）编写的规则录入框架的知识库，因此框架对于快速地构建

原型系统特别有用。

如何选择框架

本书附录中提供了一些目前市场上可用的商业专家系统的详细介绍。这些介绍将有助于你选择合适的工具。然而，因特网现在正快速成为最有价值的信息源，很多供应商都有自己的网址，因此可以通过其网址来评价他们的产品。

通常在选择专家系统框架时，要考虑框架表示知识的形式（规则或框架），所使用的推理机制（前向链接或后向链接），是否支持不精确推理及此时所使用的推理技术（贝叶斯推理、确信因子或模糊逻辑），框架是否拥有支持使用外部数据和程序的开放架构，以及用户和专家系统交互的方式（图形用户界面或超链接）。

如今，花费不到50美元就可以购买一个专家系统框架并在PC或Mac上运行。你也可以获得一个免费的专家系统（如CLIPS）。但是，你必须清楚地了解授权义务，特别是你是否有分发授权以便在智能系统开发完成时使终端用户能够使用系统。

选择系统开发工具时，一个需要考虑的重要方面是提供工具的公司的稳定性。

公司稳定性的指标是什么

公司稳定性有一些重要的指标，例如公司的成立年份、员工数量、总收入、智能系统产品的总收入、已售产品的数量。类似的指标也可以表示特定产品的稳定性。产品是什么时候正式发布的？已经发布了多少个版本？已经安装了多少套？这些在产品的开发阶段都是很重要的问题。

然而，评价产品和供应商稳定性的最好的方法可能就是得到它的用户列表，成功的应用和安装缺点。我们仅仅需要和使用产品的用户通上几分钟的电话，就可以清楚产品及其供应商的优势和缺点。

案例2：分类专家系统

我想要开发一个能够帮助我将帆船分类的智能系统。专家系统可以解决这样的问题吗？

这是一个经典的分类问题（识别帆船意味着将其归到定义好的类中）。前面已经讨论过，对于分类问题，专家系统和神经网络都可以很好地处理。如果你决定构建一个专家系统，首先应该收集一些关于桅杆结构和不同类型帆船的设计匠的信息。例如，图9.4显示了8种类型的帆船，每种类型的帆船可以由其发计图来唯一标识。

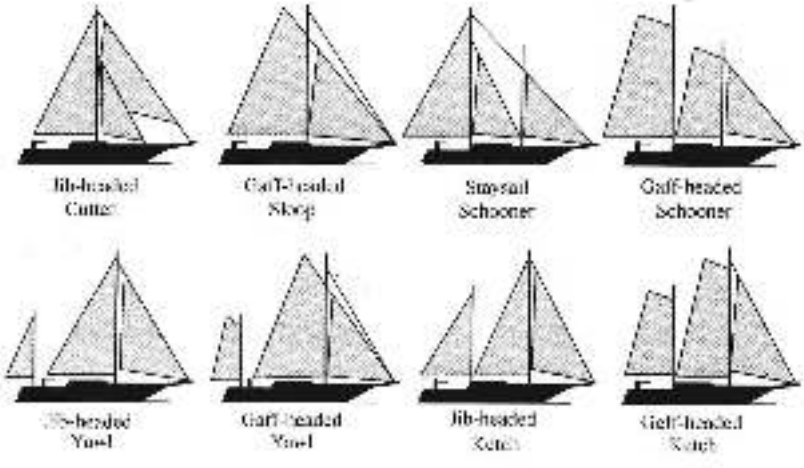


图9.4 8种类型的帆船

图9.5则显示了用于将帆船分类的一系列规则（Leonardo码）。系统在与用户对话阶段获取帆船的桅杆数量、桅杆位置和主帆结构等信息，然后将帆船确定为图9.4中的一类。

```
                帆船分类专家系统：标记 1
Rule:1   if    'the number of masts' is one
         and   'the shape of the mainsail' is triangular
         then  boat is 'Jib-headed Cutter'
Rule:2   if    'the number of masts' is one
         and   'the shape of the mainsail' is quadrilateral
         then  boat is 'Gaff-headed Sloop'
Rule:3   if    'the number of masts' is two
         and   'the main mast position' is 'forward of the short mast'
         and   'the short mast position' is 'forward of the helm'
         and   'the shape of the mainsail' is triangular
         then  boat is 'Jib-headed Ketch'
Rule:4   if    'the number of masts' is two
         and   'the main mast position' is 'forward of the short mast'
         and   'the short mast position' is 'forward of the helm'
         and   'the shape of the mainsail' is quadrilateral
         then  boat is 'Gaff-headed Ketch'
Rule:5   if    'the number of masts' is two
         and   'the main mast position' is 'forward of the short mast'
         and   'the short mast position' is 'aft the helm'
         and   'the shape of the mainsail' is triangular
         then  boat is 'Jib-headed Yawl'
Rule:6   if    'the number of masts' is two
         and   'the main mast position' is 'forward of the short mast'
         and   'the short mast position' is 'aft the helm'
         and   'the shape of the mainsail' is quadrilateral
         then  boat is 'Gaff-headed Yawl'
Rule:7   if    'the number of masts' is two
         and   'the main mast position' is 'aft the short mast'
         and   'the shape of the mainsail' is quadrilateral
         then  boat is 'Gaff-headed Schooner'
Rule:8   if    'the number of masts' is two
         and   'the main mast position' is 'aft the short mast'
         and   'the shape of the mainsail' is 'triangular with two foresails'
         then  boat is 'Staysail Schooner'
         /*************************************************************************
         /* SEEK 指令创建目标
         查找船只
```

图9.5 帆船分类专家系统的规则

毫无疑问，在天空湛蓝和海面平静的情况下，系统可以帮助我们识别帆船。但是实际情况却往往不是如此，在大风或者大雾的海面，看清楚主帆和桅杆位置会变得困难甚至不可能。尽管在解决真实世界的分类问题时会经常包含这样不确定和不完整的数据，我们还是可以使用专家系
313 统的方法，前提是我们需要处理不确定性。我们应用确信因子理论来解决我们的问题。确信因子理论可以管理增量获取的论据和不同信任度的信息。

图9.6显示了解决帆船分类问题带有确信因子的完整规则集。专家系统需要将帆船分类，也即

```
                帆船分类专家系统：标记2
                控制 cf
Rule:1          if      'the number of masts' is one
                then    boat is 'Jib-headed Cutter'      {cf 0.4};
                        boat is 'Gaff-headed Sloop'      {cf 0.4}

Rule:2          if      'the number of masts' is one
                and     'the shape of the mainsail' is triangular
                then    boat is 'Jib-headed Cutter'      {cf 1.0}

Rule:3          if      'the number of masts' is one
                and     'the shape of the mainsail' is quadrilateral
                then    boat is 'Gaff-headed Sloop'      {cf 1.0}

Rule:4          if      'the number of masts' is two
                then    boat is 'Jib-headed Ketch'       {cf 0.1};
                        boat is 'Gaff-headed Ketch'      {cf 0.1};
                        boat is 'Jib-headed Yawl'        {cf 0.1};
                        boat is'Gaff-headed Yawl'        {cf 0.1};
                        boat is 'Gaff-headed Schooner'   {cf 0.1};
                        boat is ' Staysail Schooner'     {cf 0.1}

Rule:5          if      'the number of masts' is two
                and     'the main mast position' is 'forward of the short mast'
                then    boat is 'Jib-headed Ketch'       {cf 0.2};
                        boat is 'Gaff-headed Ketch'      {cf 0.2};
                        boat is 'Jib-headed Yawl'        {cf 0.2};
                        boat is 'Gaff-headed Yawl'       {cf 0.2}

Rule:6          if      'the number of masts' is two
                and     'the main mast position' is 'aft the short mast'
                then    boat is 'Gaff-headed Schooner'   {cf 0.4};
                        boat is 'Staysail Schooner'      {cf 0.4}

Rule:7          if      'the number of masts' is two
                and     'the short mast position' is 'forward of the helm'
                then    boat is 'Jib-headed Ketch'       {cf 0.4};
                        boat is 'Gaff-headed Ketch'      {cf 0.4}

Rule:8          if      'the number of masts' is two
                and     'the short mast position' is 'aft the helm'
                then    boat is 'Jib-headed Yawl'        {cf 0.2};
                        boat is 'Gaff-headed Yawl'       {cf 0.2};
                        boat is 'Gaff-headed Schooner'   {cf 0.2};
                        boat is 'Staysail Schooner'      {cf 0.2}

Rule:9          if      'the number of masts' is two
                and     'the shape of the mainsail' is triangular
                then    boat is 'Jib-headed Ketch'       {cf 0.4};
                        boat is 'Jib-headed Yawl'        {cf 0.4}

Rule:10         if      'the number of masts' is two
                and     'the shape of the mainsail' is quadrilateral
                then    boat is 'Gaff-headed Ketch'      {cf 0.3};
                        boat is 'Gaff-headed Yawl'       {cf 0.3};
                        boat is 'Gaff-headed Schooner'   {cf 0.3}

Rule:11         if      'the number of masts' is two
                and     'the shape of the mainsail' is 'triangular with two foresails'
                then    boat is 'Staysail Schooner'      {cf 1.0}

                查找船只
```

图9.6 帆船分类专家系统中的不确定性管理

为多值对象帆船建立确信因子。要应用论据推理技术，专家系统会提示用户不仅要输入对象的值，而且要输入对象值的确定性。例如，使用取值范围为0～1的Leonardo码，我们可能遇到下面的对话（用户的回答用箭头表示，注意确信因子在不同规则中的传送）：

What is the number of masts?
⇒ **two**

To what degree do you believe that the number of masts is two? Enter a numeric certainty between 0 and 1.0 inclusive.
⇒ **0.9**

Rule: 4
if 'the number of masts' is two
then boat is 'Jib-headed Ketch' {cf 0.1};
boat is 'Gaff-headed Ketch' {cf 0.1};
boat is 'Jib-headed Yawl' {cf 0.1};
boat is 'Gaff-headed Yawl' {cf 0.1};
boat is 'Gaff-headed Schooner' {cf 0.1};
boat is 'Staysail Schooner' {cf 0.1}

cf (boat is 'Jib-headed Ketch') = cf ('number of masts' is two) × 0.1 = 0.9 × 0.1 = 0.09
cf (boat is 'Gaff-headed Ketch') = 0.9 × 0.1 = 0.09
cf (boat is 'Jib-headed Yawl') = 0.9 × 0.1 = 0.09
cf (boat is 'Gaff-headed Yawl') = 0.9 × 0.1 = 0.09
cf (boat is 'Gaff-headed Schooner') = 0.9 × 0.1 = 0.09
cf (boat is 'Staysail Schooner') = 0.9 × 0.1 = 0.09

boat is Jib-headed Ketch {cf 0.09}
Gaff-headed Ketch {cf 0.09}
Jib-headed Yawl {cf 0.09}
Gaff-headed Yawl {cf 0.09}
Gaff-headed Schooner {cf 0.09}
Staysail Schooner {cf 0.09}

What is the position of the main mast?
⇒ **aft the short mast**

To what degree do you believe that the main mast position is aft the short mast? Enter a numeric certainty between 0 and 1.0 inclusive.
⇒ **0.7**

Rule: 6
if 'the number of masts' is two
and 'the main mast position' is 'aft the short mast'
then boat is 'Gaff-headed Schooner' {cf 0.4};
boat is 'Staysail Schooner' {cf 0.4}

314
~
315

$cf_{Rule:6}$ (boat is 'Gaff-headed Schooner') = min [cf ('the number of masts' is two), cf ('the main mast position' is 'aft the short mast')] × 0.4
= min [0.9, 0.7] × 0.4 = 0.28

$cf_{Rule:6}$ (boat is 'Staysail Schooner') = min [0.9, 0.7] × 0.4 = 0.28

cf (boat is 'Gaff-headed Schooner') = $cf_{Rule:4} + cf_{Rule:6} \times (1 - cf_{Rule:4})$
= 0.09 + 0.28 × (1 − 0.09) = 0.34

cf (boat is 'Staysail Schooner') = 0.09 + 0.28 × (1 − 0.09) = 0.34

boat is Gaff-headed Schooner {cf 0.34}
Staysail Schooner {cf 0.34}
Jib-headed Ketch {cf 0.09}
Gaff-headed Ketch {cf 0.09}
Jib-headed Yawl {cf 0.09}
Gaff-headed Yawl {cf 0.09}

What is the position of the short mast?
⇒ **forward of the helm**

To what degree do you believe that the short mast position is forward of the helm? Enter a numeric certainty between 0 and 1.0 inclusive.
⇒ **0.6**

Rule: 7
if 'the number of masts' is two
and 'the short mast position' is 'forward of the helm'
then boat is 'Jib-headed Ketch' {cf 0.4};
boat is 'Gaff-headed Ketch' {cf 0.4}

$cf_{Rule:7}$ (boat is 'Jib-headed Ketch') = min [cf ('the number of masts' is two), cf ('the short mast position' is 'forward of the helm')] × 0.4
= min [0.9, 0.6] × 0.4 = 0.24

$cf_{Rule:7}$ (boat is 'Gaff-headed Ketch') = min [0.9, 0.6] × 0.4 = 0.24

cf (boat is 'Jib-headed Ketch') = $cf_{Rule:6}$ + $cf_{Rule:7}$ × (1 − $cf_{Rule:6}$)
= 0.09 + 0.24 × (1 − 0.09) = 0.30

cf (boat is 'Gaff-headed Ketch') = 0.09 + 0.24 × (1 − 0.09) = 0.30

boat is Gaff-headed Schooner {cf 0.34}
Staysail Schooner {cf 0.34}
Jib-headed Ketch {cf 0.30}
Gaff-headed Ketch {cf 0.30}
Jib-headed Yawl {cf 0.09}
Gaff-headed Yawl {cf 0.09}

What is the shape of the mainsail?
⇒ **triangular**

To what degree do you believe that the shape of the mainsail is triangular?
numeric certainty between 0 and 1.0 inclusive.
⇒ **0.8**

316

Rule: 9
if 'the number of masts' is two
and 'the shape of the mainsail' is triangular
then boat is 'Jib-headed Ketch' {cf 0.4};
boat is 'Jib-headed Yawl' {cf 0.4}

$cf_{Rule:9}$ (boat is 'Jib-headed Ketch') = min [cf ('the number of masts' is two),
cf ('the shape of the mainsail' is triangular)] × 0.4
= min [0.9, 0.8] × 0.4 = 0.32

$cf_{Rule:9}$ (boat is 'Jib-headed Yawl') = min [0.9, 0.8] × 0.4 = 0.32

cf (boat is 'Jib-headed Ketch') = $cf_{Rule:7}$ + $cf_{Rule:9}$ × (1 − $cf_{Rule:7}$)
= 0.30 + 0.32 × (1 − 0.30) = 0.52

cf (boat is 'Jib-headed Yawl') = 0.09 + 0.32 × (1 − 0.09) = 0.38

boat is Jib-headed Ketch {cf 0.52}
Jib-headed Yawl {cf 0.38}
Gaff-headed Schooner {cf 0.34}
Staysail Schooner {cf 0.34}
Gaff-headed Ketch {cf 0.30}
Gaff-headed Yawl {cf 0.09}

现在可以得出结论，帆船应该属于 Jib - headed Ketch 类型，几乎不太可能是 Gaff - headed Ketch 类型或者 Gaff - headed Yawl 类型。

9.3 模糊专家系统可以解决的问题

首先要确定模糊技术适合于解决什么类型的问题。确定是否使用模糊技术的方法很简单：如果你不能为每种可能的情况定义一个精确的规则集，那就使用模糊逻辑。确信因子和贝叶斯概率主要关心明确定义的事件中的不明确结果，而模糊逻辑主要针对不明确的事件本身。换句话说，如果问题本身就不严密，那么模糊技术是最好的选择。

模糊系统特别适合对人类的决策进行建模。当我们作出重要决定时，往往依赖于常识和使用模糊的词语。例如，是否要把一个术后病人从康复区转到普通病房，医生并没有一个十分明确的阈值。由于术后体温是一个非常重要的指标，所以病人的体温通常是医生决策的关键因素。而患者血压的稳定性，患者的感觉等因素也需要考虑。医生不是通过某一个能够精确表示的参数（体温）来做决定，而是要同时评估几个参数，并且有些参数只能用模糊的术语表示（例如患者想离开康复区的意愿）。

317

虽然大多数情况下模糊技术还只是应用于控制和工程领域，但是其在商务和金融业领域前景更加广阔（Von Altrock，1997）。这些领域的决策通常是基于直觉、常识和经验的，而不是数据的可用性和准确性。商务和金融领域的决策制定非常复杂而且不确定，以至于不能用精确的分析方法。而模糊技术为我们提供了一种处理商业和金融业常用的“软标准”和“模糊数据”的方式。

案例3：决策支持模糊系统

我们要开发一个评估抵押申请的智能系统。模糊专家系统可以解决这样的问题吗？

抵押申请评估是决策支持模糊系统能够成功应用的典型问题（Von Altrock，1997）。

要开发解决这个问题的决策支持模糊系统，我们首先使用模糊术语表达抵押申请评估中的基本概念，然后用合适的模糊工具在原型系统中实现这个概念，最后用选定的测试用例来测试和优化系统。

抵押申请的评估通常基于评估市场价和房产的位置、申请人的资产和收入，以及还款计划，而这些都取决于申请人的收入和银行的利率。

抵押贷款评估的隶属函数和规则从哪里来

要定义隶属函数并构建模糊规则，我们需要有经验的抵押顾问和银行经理的帮助，他们能够列出抵押认可的方针。图9.7～图9.14是本问题中使用的语言变量的模糊集。三角形和梯形的隶属函数可以充分地表示抵押专家的知识。

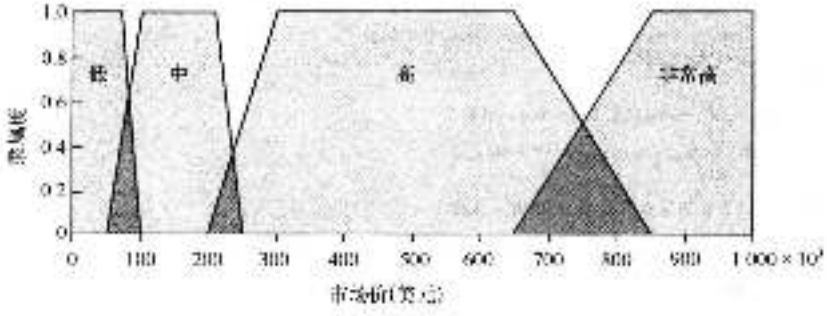


图9.7 语言变量 Market Value 的模糊集

318

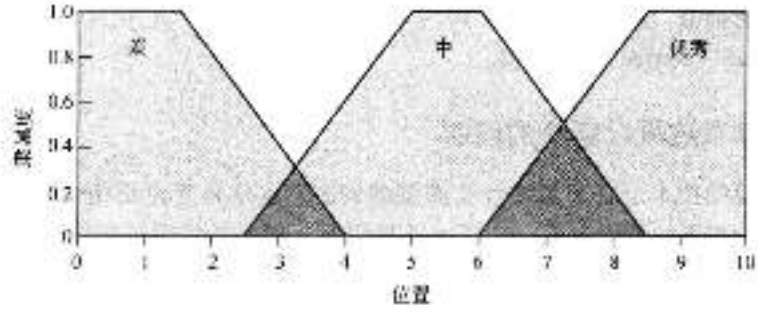


图9.8 语言变量 Location 的模糊集

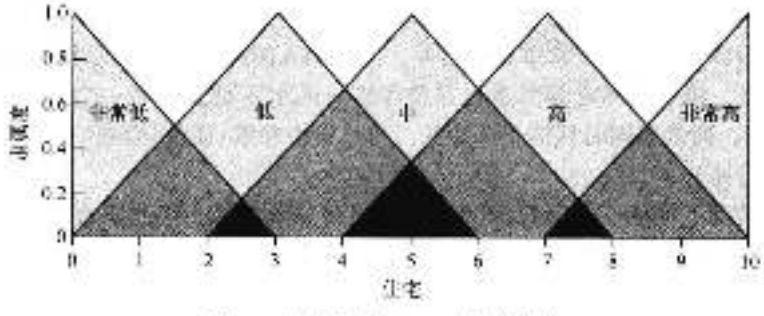


图9.9 语言变量 House 的模糊集

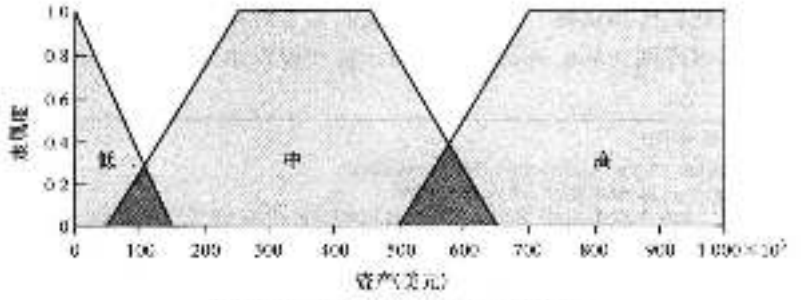


图 9.10 语言变量 Asset 的模糊集

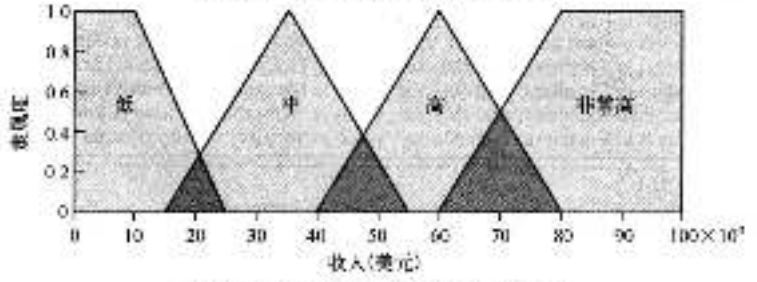


图 9.11 语言变量 Income 的模糊集

312

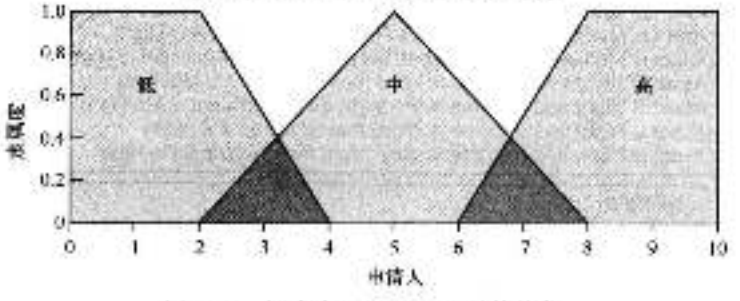


图 9.12 语言变量 Applicant 的模糊集

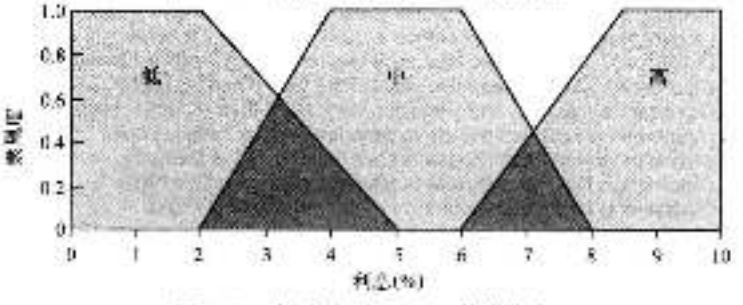


图 9.13 语言变量 Interest 的模糊集

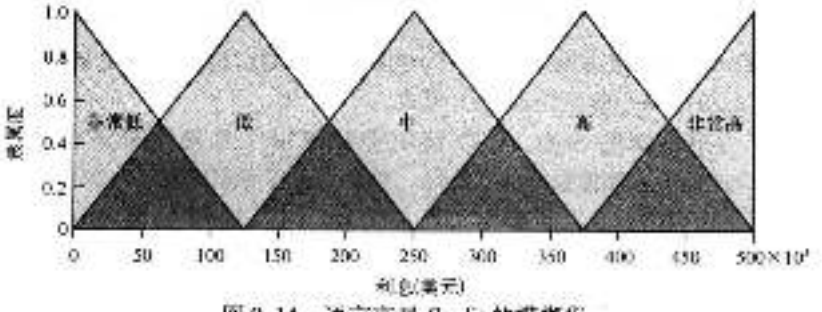


图 9.14 语言变量 Credit 的模糊集

接下来我们需要获取模糊规则。在本例中，我们只是借用了 Von Altrock 在其抵押贷款评估模糊模型中使用的基本规则（Von Altrock，1997）。这些规则如图 9.15 所示。

规则库1：住房评估

1. If (Market_value is Low) then (House is Low)
2. If (Location is Bad) then (House is Low)
3. If (Location is Bad) and (Market_value is Low) then (House is Very_low)
4. If (Location is Bad) and (Market_value is Medium) then (House is Low)
5. If (Location is Bad) and (Market_value is High) then (House is Medium)
6. If (Location is Bad) and (Market_value is Very_ high) then (House is High)
7. If (Location is Fair) and (Market_value is Low) then (House is Low)
8. If (Location is Fair) and (Market_value is Medium) then (House is Medium)
9. If (Location is Fair) and (Market_value is High) then (House is High)
10. If (Location is Fair) and (Market_value is Very_high) then (House is Very_high)
11. If (Location is Excellent) and (Market_value is Low) then (House is Medium)
12. If (Location is Excellent) and (Market_value is Medium) then (House is High)
13. If (Location is Excellent) and (Market_value is High) then (House is Very_high)
14. If (Location is Excellent) and (Market_value is Very_high) then (House is Very_high)

规则库2：申请评估

1. If (Asset is Low) and (Income is Low) then (Applicant is Low)
2. If (Asset is Low) and (Income is Medium) then (Applicant is Low)
3. If (Asset is Low) and (Income is High) then (Applicant is Medium)
4. If (Asset is Low) and (Income is Very_high) then (Applicant is High)
5. If (Asset is Medium) and (Income is Low) then (Applicant is Low)
6. If (Asset is Medium) and (Income is Medium) then (Applicant is Medium)
7. If (Asset is Medium) and (Income is High) then (Applicant is High)
8. If (Asset is Medium) and (Income is Very_high) then (Applicant is High)
9. If (Asset is High) and (Income is Low) then (Applicant is Medium)
10. If (Asset is High) and (Income is Medium) then (Applicant is Medium)
11. If (Asset is High) and (Income is High) then (Applicant is High)
12. If (Asset is High) and (Income is Very_high) then (Applicant is High)

规则库3：信用评估

1. If (Income is Low) and (Interest is Medium) then (Credit is Very_low)
2. If (Income is Low) and (Interest is High) then (Credit is Very_low)
3. If (Income is Medium) and (Interest is High) then (Credit is Low)
4. If (Applicant is Low) then (Credit is Very_low)
5. If (House is Very_low) then (Credit is Very_low)
6. If (Applicant is Medium) and (House is Very_low) then (Credit is Low)
7. If (Applicant is Medium) and (House is Low) then (Credit is Low)
8. If (Applicant is Medium) and (House is Medium) then (Credit is Medium)
9. If (Applicant is Medium) and (House is High) then (Credit is High)
10. If (Applicant is Medium) and (House is Very_high) then (Credit is High)
11. If (Applicant is High) and (House is Very_low) then (Credit is Low)
12. If (Applicant is High) and (House is Low) then (Credit is Medium)
13. If (Applicant is High) and (House is Medium) then (Credit is High)
14. If (Applicant is High) and (House is High) then (Credit is High)
15. If (Applicant is High) and (House is Very_high) then (Credit is Very_high)

图 9.15 抵押贷款评估的规则

模糊系统中使用的所有变量的复杂关系可用层次结构来表示，如图 9.16 所示。

我们使用 MATLAB 的模糊逻辑工具箱来构建系统，它是市场上最常用的模糊工具之一。

开发原型系统的最后一个阶段是评价和测试。

要评价和分析模糊系统的性能，可以使用模糊逻辑工具箱中提供的输出表面查看器。图 9.17 和图 9.18 为抵押贷款评估系统的三维图。最后。抵押专家使用一些测试用例来测试系统。

320
~
321

决策支持模糊系统可能包含几十甚至上百条规则。例如 BMW 银行和 Inform Software 开发的信用风险评估模糊系统中使用了 413 条规则（Güllich，1996）。大型的知识库通常采用类似图 9.16 所示的方式来将其分割成几个模块。

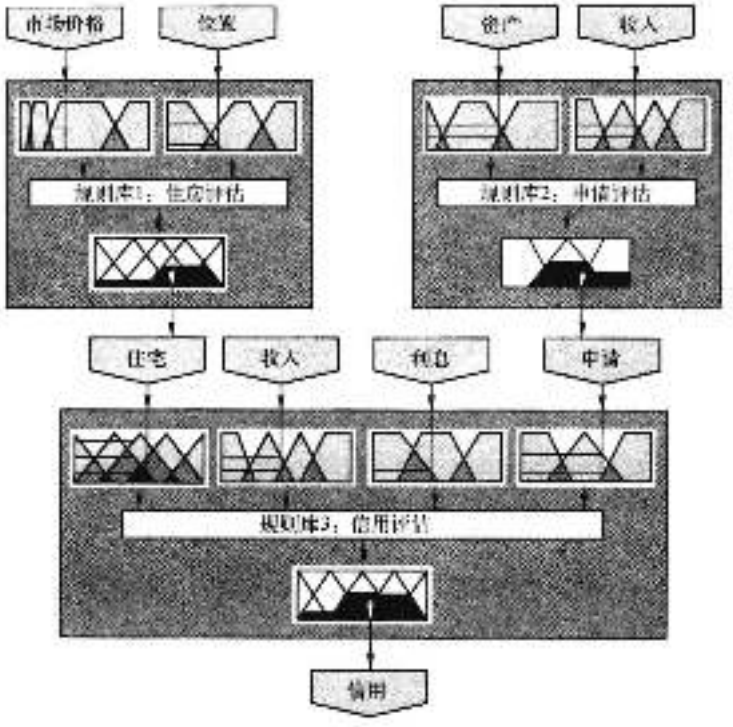


图 9.16 抵押贷款评估的层次模型

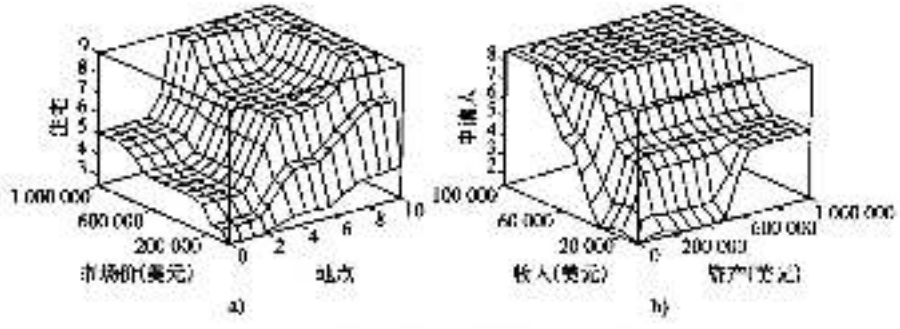


图 9.17 规则库 1 和规则库 2 的三维图

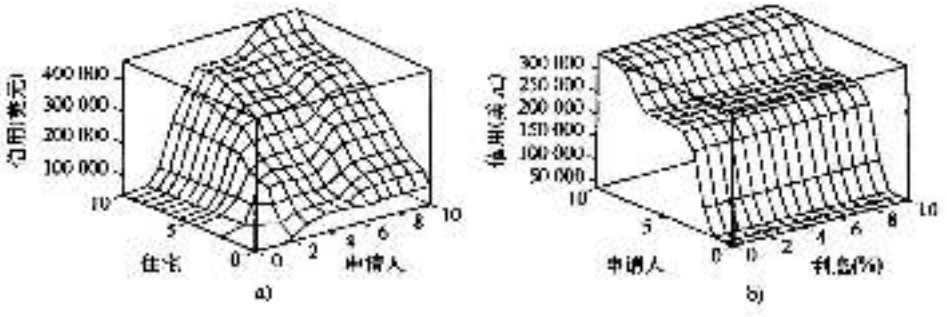


图 9.18 规则库 3 的三维图

尽管通常有大量的规则，决策支持模糊系统可以相对快速地开发、测试和运行。例如，开发和实现一个信用风险评估模糊系统仅需要 2 人年。相比之下，开发 MYCIN 用了 40 人年。

9.4 神经网络可以解决的问题

神经网络是一种功能强大、通用的工具，它已经成功地应用在预测、分类和聚类问题中。神经网络的应用领域非常广泛，从语音和特征识别到交易欺诈检测、从心脏病的医疗诊断到过程控制和机器人技术，从外汇汇率预测到雷达目标识别，等等。神经网络的应用领域仍在快速扩展。

神经网络的普及得益于它非凡的多功能性，处理二值或者连续的数据的能力，以及能在复杂的领域中得到很好的结果的能力。如果输出是连续的，神经网络可以解决预测问题；如果输出为二值的，神经网络可以作为分类器使用。

案例4：字符识别神经网络

我想要开发一个字符识别系统，神经网络可以解决这个问题吗？

识别打印体或手写体字符是神经网络成功应用的经典领域。实际上，光学字符识别（optical character recognition）系统是神经网络最早的商业应用之一。

什么是光学字符识别

计算机使用特殊的软件能够将字符图像转换成文本文件。它使得我们将打印好的文件，以可编辑的形式放入计算机，而不需要重新录入文件。

我们可以用扫描仪得到字符图像。为完成这个任务，我们既可以用光敏的传感器扫过发光的文稿表面，也可以让文稿扫过传感器。扫描仪把每页图像分割为上百个像素大小的小格，每个小格用0（如果小格内有填充）或1（小格内为空）表示。最后得到的点矩阵叫做位图。位图可以用计算机存储、显示和打印，但不能用字处理器编辑。因为点阵的模式不能被计算机识别，而这正是神经网络的工作。

322
!
323

下面我们就通过识别打印字符的多层前馈网络的应用来进行说明。为了简化这个问题，我们限制其仅识别数字0～9，每个数字用图9.19所示的5×9的位图来表示。在商业应用中，为了得到更好的分辨率，通常使用16×16的位图来表示（Zurada，1992）。

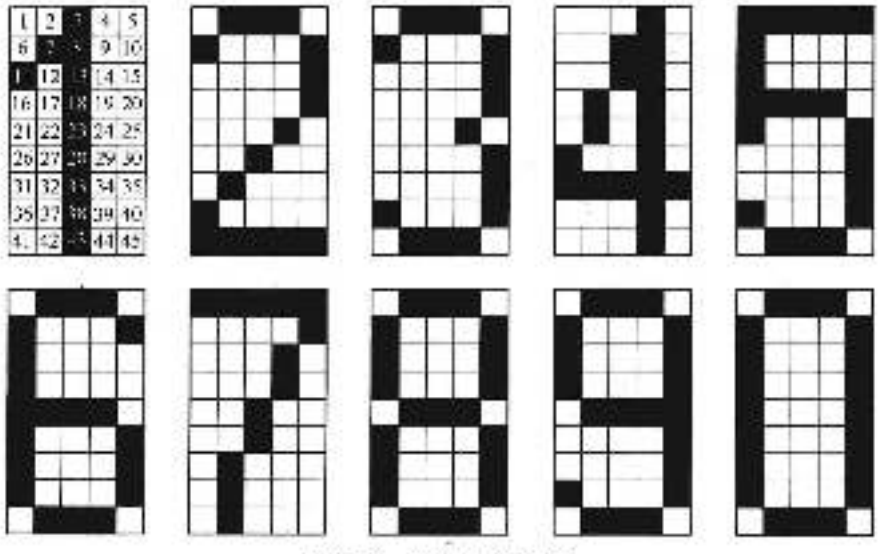


图9.19 数字识别的位图

如何选择用于字符识别的神经网络结构？

神经网络的结构和大小取决于问题的复杂程度。例如，手写字符的识别通常通过相对复杂

的多层结构来执行，它包括3层甚至4层隐含层和数百个神经元（Zurada, 2006; Haykin, 2008）。然而，对于打印字的数字识别问题，包含一个隐含层的三层网络就已经可以提供足够的精确性。

输入层神经元的数目是由位图像素的数目来决定的。在本例中，位图由45个像素组成，因此我们需要45个输入神经元。输出层有10个神经元——每一个神经元用来识别一个数字。

如何确定隐含层神经元的最优数目？

仿真实验表明，隐含层神经元的数目能够影响字符识别的精确度和训练网络的速度。隐含层神经元太少则不能识别复杂的模式，但是隐含层神经元太多又会极大地增加计算负担。 324

另一个问题是过拟合问题。隐含层神经元数目越多，网络识别已经存在的模式的能力越强。但是，如果隐含层神经元的数目太多，则网络很可能只记住所有训练样本。这样，在输入训练集中没有的样本时，网络就无法进行泛化或得到正确的输出。例如，用Helvetica字体训练的过拟合神经网络就无法识别Times New Roman字体的相同字符。

避免过拟合问题的方法是选择使网络产生良好的泛化结果所需要的最少的隐含层的神经元数目。这样，在实验的时候，就可以从隐含层只有2个神经元开始。在本例中，我们会使用2、5、10和20个隐含层神经元数目来测试网络并比较其结果。

字符识别问题的神经网络结构（隐含层有5个神经元），如图9.20所示。隐含层和输出层的神经元使用S形激活函数。神经网络用要素的反向传播算法进行训练。动量常数为0.95。输入和输出训练模式如表9.2所示。表示各个数字的二值输入向量直接输入网络中。

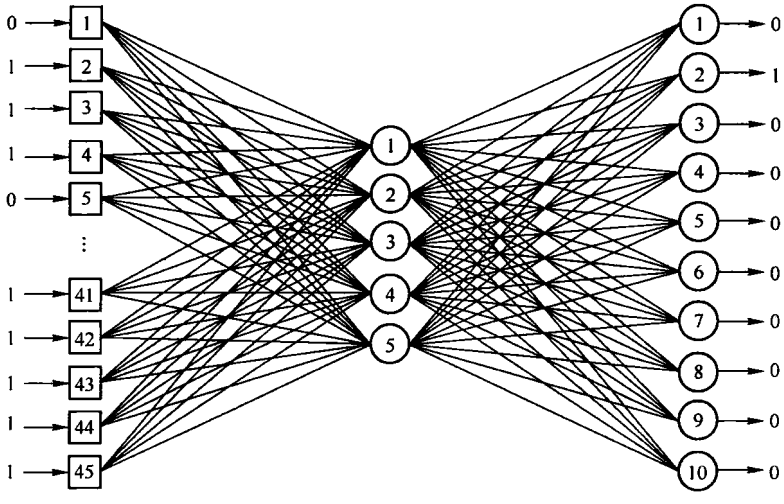


图9.20 打印体数字识别的神经网络

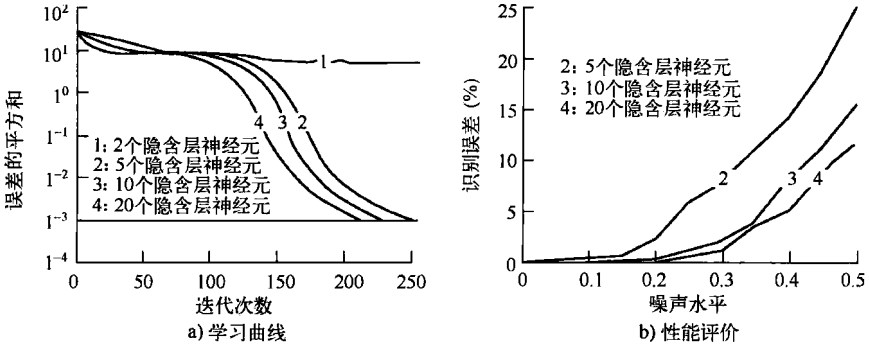
表9.2 数字识别神经网络的输入和预期输出模式

数字	输入模式像素矩阵中的行									预期输出模式
	1	2	3	4	5	6	7	8	9	
1	00100	01100	10100	00100	00100	00100	00100	00100	00100	1000000000
2	01110	10001	00001	00001	00010	00100	01000	100000	111111	0100000000
3	01110	10001	00001	00001	00010	00001	00001	10001	01110	0010000000
4	00010	00110	00110	01010	01010	10010	11111	00010	00010	0001000000
5	11111	10000	10000	11110	10001	00001	00001	10001	01110	0000100000
6	01110	10001	10000	10000	11110	10001	10001	10001	01110	0000010000
7	11111	00001	00010	00010	00100	00100	01000	01000	01000	0000001000
8	01110	10001	10001	10001	01110	10001	10001	10001	01110	0000000100
9	01110	10001	10001	10001	01111	00001	00001	10001	01110	0000000010
0	01110	10001	10001	10001	10001	10001	10001	10001	01110	0000000001

在我们的研究中，神经网络的性能用平方误差和来衡量。图 9.21 显示了实验结果，从图 9.21a 可以看出，隐含层只有 2 个神经元时无法收敛到一个解。然而隐含层神经元为 5、10 和 20 时学习则可以相对较快。实际上，他们都可以在 250 次迭代以内收敛（一次迭代是指遍历一次整个训练样本）。注意，有 20 个隐含层神经元的网络收敛最快。

325
~
326

训练完成以后，我们需要用一个测试样本集来测试网络的性能。



a) 学习曲线　　b) 性能评价

图 9.21　数字识别三层神经网络的训练和性能评价

什么是数字识别的测试样本？它们和用于训练神经网络的样本相同吗？

测试集必须严格独立于训练样本。因此测试数字识别网络时，需要使用一些“噪声”样本——扭曲的输入模式。例如，为了产生扭曲的模式，我们可以从正态分布中抽样出一些小的随机值，并将其加到表示 10 个数字的位图的二值输入向量中。我们使用 1000 个测试样本（每个数字对应 100 个测试样本）评测了打印体数字识别神经网络的性能。评测结果如图 9.21b 所示。

虽然包含 20 个隐含层神经元的神经网络的平均识别误差最小，实验结果却显示 10 个隐含层神经元和 20 个隐含层神经元的差别并不十分显著。在不牺牲识别性能的前提下，两种网络的容纳噪声的能力很相似。基于此，我们可以得出结果，在这里描述的数字识别问题中，使用 10 个隐含层神经元就足够了。

可以提高字符识别神经网络的性能吗？

神经网络的性能与训练它的样本直接相关。因此，我们可以使用数字 0 ~ 9 的一些“噪声”样本作为训练样本来提高数字识别的性能。此时的实验结果如图 9.22 所示。和我们期望的一样，利用了“噪声”样本训练的数字识别网络确实得到了一些性能上的提高。

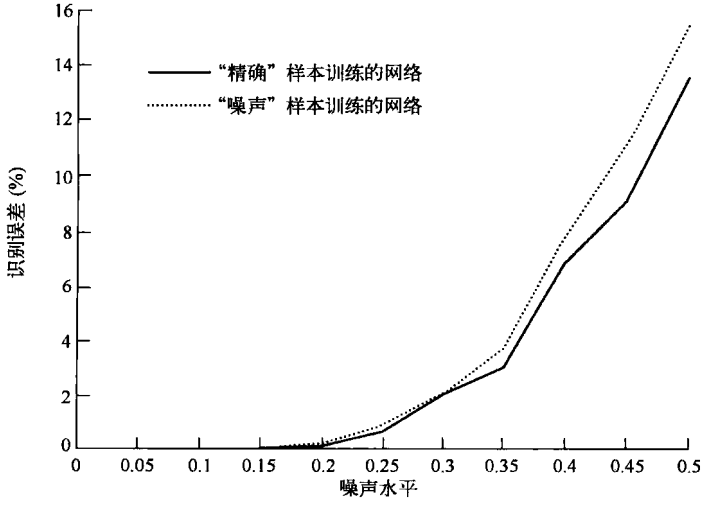


图 9.22　“噪声”样本训练的数字识别网络的性能评测

本研究案例说明了用反向传播算法训练的多层神经网络的一个最常见的应用。现代的字符识别系统能够以很高的精确度处理英文、法文、西班牙文、意大利文、荷兰文以及多种其他语言的不同字体。光学字符识别通常的使用人员有上班人员、律师、保险业务员和记者——实际上是任何希望打印（甚至是手写）文件转载进计算机并转换成可编辑文件的人。手写数字识别系统广泛应用在处理信封的邮政编码方面（LeCun et al.，1990）。

案例5：预测神经网络

我想要开发一个房地产评估的智能系统，神经网络可以解决这个问题吗？

房地产评估是一个根据类似住房的销售价格等知识来预测一个给定房屋的市场价值的问题。前面已经提到过，这个问题既可以使用专家系统来解决，也可以使用神经网络来解决。当然，如果我们选择应用神经网络来解决，我们就无法知道对一个特定房屋的评估是如何获得的，因为神经网络对用户来说本质上是一个黑盒，因而很难从中提取规则。另一方面，一个正确的预测往往比知道如何预测更为重要。

在这个问题中，输入（房屋位置、居住面积、卧室数量、浴室数量、土地尺寸、供热系统类型等）是良好定义的，而且甚至是标准化过的，不同的房产代理可以共享这些房屋市场信息。输出也是良好定义的——我们知道所要预测的目标是什么。更重要的是，我们有很多样本可以用来训练神经网络。这些样本就是最近售出的房屋及其售出价格的特征。 327～328

训练样本的选择是准确预测的关键。训练集应该覆盖输入的整个取值范围。因此，在房地产评估的训练集中，我们应该涵盖大房子和小房子，昂贵的房子和便宜的房子，带车库和不带车库的房子，等等。而且，训练集应该足够大。

如何确定何时训练集的大小才是“足够大”的？

神经网络的泛化能力主要取决于3个因素：训练集的大小、网络的架构和问题的复杂度。一旦网络的架构确定了以后，泛化能力取决于是否有充足的训练集。合适的训练样本数量可以使用Widrow的*拇指规则*来估计。拇指规则指出，为了得到一个较好的泛化能力，我们需要满足以下条件（Widrow and Stearns，1985；Haykin，2008）：

$$N=\frac{n_w}{e} \tag{9.1}$$

其中，N为训练样本数量，n_w是网络中突触权重的数量，e是测试允许的网络误差。

因此，假如我们允许10%的误差，我们需要的训练样本的数量大约是网络中权值数量的10倍。

在解决房地产评估等预测问题时，我们往往需要结合不同类型的输入特征。有一些特征（例如房屋条件和位置，）可以取1（最没吸引力）到10（最具有吸引力）之间的任意值。另一些特征（例如居住面积、土地尺寸和销售价格）则可以根据实际的物理特性来度量——平方米和美元，等等。还有一些特征代表数量（卧室数量、浴室数量等）和分类（供热系统类型）。

当所有输入和输出取值都在0～1时，神经网络的性能表现得很好。因此，所有的数据在用来训练神经网络之前必须进行归一化。

如何归一化数据？

数据可以分为3种主要的类型：连续数据、离散数据和分类数据（Berry and Linoff，2004）。对于不同类型的数据，我们通常采用不同的技术来进行归一化。

连续数据取值介于预先设定的最小值和最大值之间，可以很容易地映射（归一化）到0和1之间：

$$\text{规范值}=\frac{\text{实际值}-\text{最小值}}{\text{最大值}-\text{最小值}} \tag{9.2}$$ 329

例如，在训练样本中，如果房屋的居住面积在59平方米和231平方米之间，我们可以将最小值和最大值分别设定为50平方米和250平方米。小于最小值的值将会被映射到最小值，大于最大值的值将会被映射到最大值。因此，121平方米的居住面积将会被规范为：

$$规范值_{121}=\frac{121-50}{250-50}=0.355$$

这种归一化的方法适合于大多数应用。

卧室数量和浴室数量等离散数据也有最大值和最小值。例如，卧室数量通常为0～4个。离散数据的归一化很简单——我们给每一个可能的取值在0～1之间分配一个相等的区间，如图9.23所示。

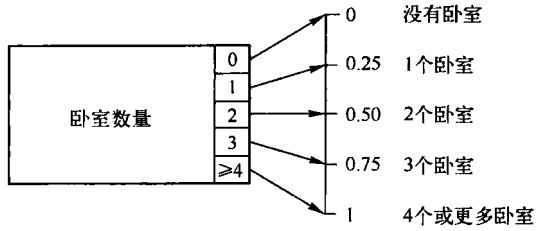


图9.23 离散数据的归一化

现在，神经网络可以将每一个特征（例如卧室数量）当做一个单独的输入来处理。例如，如果卧室数量为3，那么对应的输入值为0.75。

对于拥有具有上十种可能取值的离散特征的大多数应用来说，这种方法已经足够了。然而，如果离散特征的可能具有更多种取值，离散特征应该当做连续特征来处理。

性别和婚姻状态等分类数据可以用1/N编码（Berry and Linoff, 2004）来进行归一化。这种方法意味着将每一个分类值当做一个单独的输入来处理。例如，婚姻状态可以为单身、离异、已婚或丧偶。它将会被标示成4个输入，每一个输入取值为0或1。因此，一个已婚的人士将会被表示成一个输入向量［0 0 1 0］。

现在，我们来构建一个用于房地产评估的前馈神经网络。图9.24表示的是一个利用霍巴特地区最近售出的房屋的特征作为训练样本建立的一个简单模型。

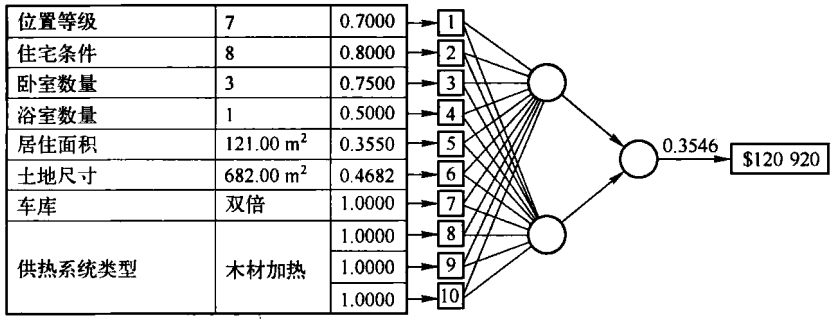


图9.24 用于房地产评估的前馈神经网络

在这个模型中，包含10个神经元的输入层将归一化后的输入值传输给隐含层。除了供热系统类型外，其他的每一个特征都作为一个单独的输入来处理。供热系统类型是一种分类数据，用1/N编码来进行归一化。

隐含层包含2个神经元，输出层则只有一个神经元。隐含层和输出层的神经元都应用了S形激活函数。

房地产评估的神经网络用来确定房屋的价值，因此网络的输出可以使用美元（为度量）来解释。

如何解释神经网络的输出？

在我们的例子中，神经网络的输出表示为0～1之间的连续值。因此，为了解释输出结果，我们可以简单地将归一化连续数据的过程倒转。例如，假设在训练集中，销售价格取值在52 500美元和225 000美元之间，输出值可以这样建立：将50 000美元映射到0，将250 000美元映射到

1。因此，如果神经网络的输出为0.3546，则我们可以计算这个值对应的实际值：

$$实际值_{0.3546} = 0.3546 \times (\$250\,000 - \$50\,000) + \$50\,000 = \$120\,920$$

如何验证结果？

为了验证结果，我们必须使用神经网络尚未遇到的样本集。在训练神经网络之前，所有的可用数据被随机地分到一个训练集和一个测试集中。训练阶段完成以后，网络的泛化能力将由测试集中的样本来测试。

神经网络是不透明的，因此我们看不到网络是如何得到结果的。但是我们仍然需要把握网络输入和其产生的结果之间的关系。尽管当前从神经网络中提取规则的研究最终会带来充足的成果，然而神经元的非线性特性将会给产生简单和可理解的规则造成障碍。幸运的是，理解特定输入对于网络输出的重要性，我们并不需要规则提取。我们可以使用一种称为*灵敏度分析*（sensitivity analysis）的简单技术。

灵敏度分析可以确定模型输出对于特定输入的敏感程度。这种技术可以用来理解不透明模型的内部关系，因此也适用于神经网络。灵敏度分析依次将每一个输入设定为最小值和最大值，然后分别测量网络输出。有些输入的改变对网络输出的影响很小，即网络对这些输入不敏感。另一些输入的改变对网络输出的影响更大，即网络对这些变量敏感。网络输出的变化量代表了网络对相应输入的敏感程度。在很多情况下，灵敏度分析的效果和从训练过的神经网络中提取规则一样。

330
331

案例6：具有竞争学习的分类神经网络

我想要开发一个智能系统用来对一组鸢尾属植物进行分类，并将任意一种鸢尾属植物植物分配到这些类别的一类中。我现在有一个包含多个变量的数据集，因为现在尚未找出唯一或特别的特征，所以还不明确应该如何将其分类。神经网络可以解决这个问题吗？

神经网络可以发现输入模式中的重要特征和学习如何将输入数据分为不同的类。具有竞争学习的神经网络是完成这项任务的合适工具。

竞争学习规则使单层神经网络可以将相似的数据组合成组或*类簇*。这个过程称为聚类。每一个群集用一个单独的输出表示。实际上，聚类可以定义为将输入空间划分为不同区域的过程，每一个区域与一个特定的输出相关（Principe et al.，2000）。

在这个案例中，我们将使用一种包含150个样本的数据集，这些样本取自3种鸢尾属植物类别：山鸢尾、变色鸢尾和维吉尼亚鸢尾（Fisher，1950）。数据集中的每一种植物用4个变量来表示：萼片长度、萼片宽度、花瓣长度和花瓣宽度。萼片长度的取值范围为4.3～7.9 cm，萼片宽度的取值范围为2.0～4.4 cm，花瓣长度的取值范围为1.0～6.9 cm，花瓣宽度的取值范围为0.1～2.5 cm。

在竞争神经网络中，每个输入神经元对应一个单独的输入，每个竞争神经元代表一个单独的聚类。因此，鸢尾属植物分类问题的神经网络中，输入层包含4个神经元，竞争层包含3个神经元。神经网络的架构如图9.25所示。

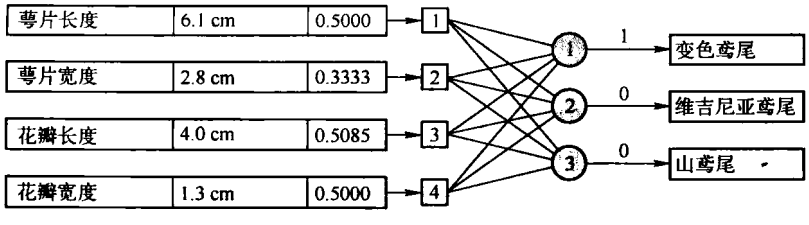


图9.25 鸢尾属植物分类的神经网络

332

然而，在神经网络训练之前，必须先将数据进行归一化，然后将数据划分为训练集和测试集。

鸢尾属植物数据是连续的，取值介于一些最小值和最大值之间。因此，利用公式（9.2）可以很容易地归一化到0～1之间。归一化的数据就可以作为神经网络的输入了。

下一步是从可用数据中生成训练集和测试集。包含150个样本的鸢尾属植物数据被随机地划分为包含100个样本的训练集和包含50个样本的测试集。

接下来就是训练竞争神经网络，并将输入向量分为3类。图9.26显示了学习速率为0.01时的学习过程。图中的黑点表示输入模式，3个球体指示竞争神经元的权重向量。每一个球体的位置是由4维输入空间的神经元的权重决定的。

初始时，竞争神经元的所有权重指定为相同的值0.5，因此只有一个球体出现在输入空间的中心，如图9.26a所示。训练开始以后，权重向量对应于群集中心的位置，因此每个竞争神经元现在只对特定区域的输入数据有响应。

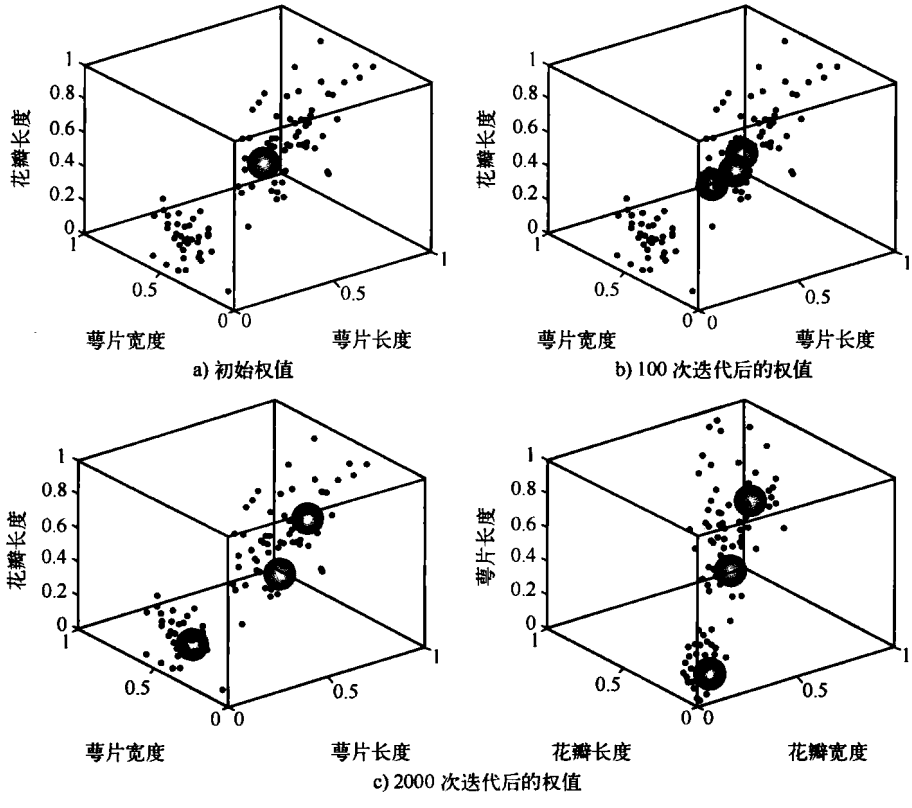


图9.26 鸢尾属植物分类的竞争学习

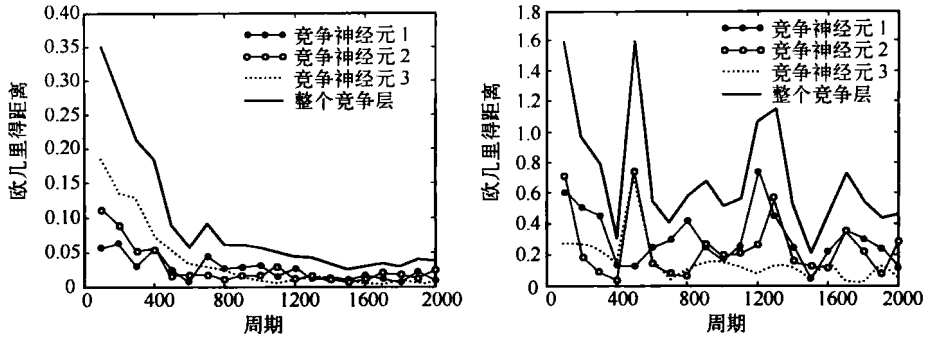
333

如何知道学习过程已经完成?

在竞争神经网络中，没有明显的方法可以得知学习过程是否完成，这一点与利用反向传播算法训练的多层感知器是不一样的。因为我们不知道预期的输出是什么，所以也就无法计算误差平方和——反向传播算法使用的标准。因此，我们应该使用的标准是欧几里得距离。当竞争神经元的权重向量不再发生明显的变化时，我们可以认为网络已经收敛。换句话说，如果输入空间中竞争神经元的运动在连续几个周期中保持稳定，我们就可以假设学习过程已经完成。

图9.27显示了鸢尾属植物分类神经网络的神经元的动态学习过程。网络分别使用两个不同

的学习速率来训练。如图9.27b所示，如果学习速率过快，竞争神经元的行为变得不稳定，网络可能无法收敛。然而，为了加快学习，我们仍然可以给学习速率赋予比较大的初值，但是随着训练的进行逐渐地减小学习速率。



a) 学习速率为0.01　　b) 学习速率为0.5

图9.27　鸢尾属植物分类神经网络的竞争神经元的学习曲线

如何将输出神经元和某个类关联？例如，我们如何确定神经元1代表变色鸢尾类？

竞争神经网络使得我们可以识别输入数据中的聚类。然而，由于聚类是一个无监督过程，我们不能用它来直接标记输出神经元。实际上，聚类只是分类的初步阶段。

在大多数实际应用中，属于同一聚类的数据的分布很稠密，不同的聚类之间有着自然地分界。因此，聚类中心的位置反映了对应聚类与其他聚类相互区分的特征。另一方面，训练获得的竞争神经元权重向量给我们提供了输入空间中这些聚类中心的坐标。因此，一个竞争神经元可以通过它的权重来与一个特定的类关联。表9.3包含了竞争神经元的最终权重、权重的解码值以及对应的鸢尾属植物的类。 334

表9.3　标记竞争神经元

神　经　元	权　　重	鸢尾属植物的尺寸（cm）	鸢尾属植物类别
1	W_{11} 0.4355	萼片长度 5.9	变色鸢尾
	W_{21} 0.3022	萼片宽度 2.7	
	W_{31} 0.5658	花瓣长度 4.4	
	W_{41} 0.5300	花瓣宽度 1.4	
2	W_{12} 0.6514	萼片长度 6.7	维吉尼亚鸢尾
	W_{22} 0.4348	萼片宽度 3.0	
	W_{32} 0.7620	花瓣长度 5.5	
	W_{42} 0.7882	花瓣宽度 2.0	
3	W_{13} 0.2060	萼片长度 5.0	山鸢尾
	W_{23} 0.6056	萼片宽度 3.5	
	W_{33} 0.0940	花瓣长度 1.6	
	W_{43} 0.0799	花瓣宽度 0.3	

如何将权值解码为鸢尾属植物的尺寸？

为了将竞争神经元的权重解码为鸢尾属植物的尺寸，只要使用数据归一化的逆过程。例如：

$$\text{萼片长度 } W_{11} = 0.4355 \times (7.9 - 4.3) + 4.3 = 5.9\,\text{cm}$$

一旦权值被解码，我们就可以让鸢尾属植物专家来标记输出神经元了。

能否不经过专家自动地标记竞争神经元？

我们可以使用测试数据集来自动地标记竞争神经元。神经网络训练完成以后，一组代表相

同类（例如变色鸢尾）的输入样本被输入网络，在竞争中获胜最多的神经元获得对应的类标记。

虽然竞争神经网络只有一层竞争神经元，它却可以对不是线性可分的输入模式进行分类。在分类任务中，竞争神经网络的学习要比利用反向传播算法训练的多层感知器要快很多，但是
335 它提供的精确度却要比后者低。

案例7：自组织神经网络聚类

我想要开发一个智能系统来识别可能倒闭的银行。神经网络可以解决这个问题吗？

2008年，一系列银行的倒闭引发了金融危机。按照任何历史标准，这场危机是20世纪30年代大萧条以来最严重的。危机的直接原因是美国房地产泡沫的破灭。反过来，这引起证券价值和房地产定价陷入低谷，损害了世界各地的金融机构。银行破产和信用缺乏降低了投资者的信心，因此，股市暴跌。2009年，全球经济收缩1.1%，而在发达国家，收缩达3.4%。中央银行和政府进行规模空前的干预后，全球经济开始复苏。然而，全球金融体系仍处于危险之中。

如果我们能够识别出在银行面临偿债能力和资金流危机之前出现的潜在问题，各大银行连锁倒闭的风险将会显著降低。导致银行倒闭的原因有很多，包括高风险、利率波动、管理不善、不合适的会计标准和非存款机构的竞争加剧等。自从危机以来，银行监管机构已越来越关注降低存款保险负债规模。甚至有人建议，最好的监管政策是在银行资本不足之前关闭银行。因此，尽早识别潜在的濒临破产的银行，对于避免另一场巨大的金融危机是至关重要的。在过去30年中，很多工具被开发出来以识别有问题的银行。早期的模型大多依赖于统计技术（Abrams and Huang, 1987; Booth et al., 1989; Espahbodi, 1991），而大多数现代技术则基于模糊逻辑和神经网络（Zhang et al., 1999; Alam et al., 2000; Ozkan - Gunay and Ozkan, 2007）。大多数的这些模型使用二元分类——破产银行和非破产银行。然而，在现实世界中，银行是按照破产可能性排序的。监管机构需要的是一个早期预警系统，能够“标记”潜在的破产银行。一旦这样的银行被识别，就可以针对每一个银行的特定需求制定不同的预案，从而避免了重大的银行失败。

在这个案例研究中，我们使用聚类分析来“标记”潜在的破产银行。

什么是聚类分析？

聚类分析是一种探索性数据分析技术，它将不同的对象划分为组（称为聚类），从而使同一聚类内的两个对象之间的相关度最大化，不同聚类的两个对象之间的相关度最小化。

“聚类分析”一词最早是由Robert Tryon（1939）于70多年前提出。自那以后，聚类分析被
336 成功应用于医学、考古和天文等许多领域。在聚类中，没有预先定义的类——对象仅仅是基于它们之间的相似度而组织到一起。因此，聚类也常常被称为无监督分类。独立变量和非独立变量之间没有区别，当发现聚类时需要用户来解释它们的含义。

使用什么方法来做聚类分析？

我们可以找出3种主要的聚类分析的方法。它们分别基于统计学、模糊逻辑和神经网络。在这个案例研究中，我们将使用一个自组织的神经网络。

我们的目标是用银行的经济数据来将银行聚类。这些数据可以从联邦存款保险公司（FDIC）的人工报表中获得。FDIC是由美国国会创建的独立机构。它负责担保存款、检查和监督金融机构以及管理接管。为了评估一个银行的整体经济状态，监管机构会使用CAMELS（资本充足率、资产、管理、盈利、流动性和市场风险敏感性）评级系统。CAMELS评级已经应用到美国的8500家银行。该系统还被美国政府在2008年的资本化方案中用来选择银行。

在本案例中，我们选择了100家银行并从上一年的FDIC人工报表中获取了这些银行的经济数据。基于CAMELS，我们采用以下5个评级：

1）NITA——收入总额除以资产净额。NITA代表资产收益率。濒临破产的银行NITA非常

低，甚至出现负值。

2）NLLAA——净贷款损失除以调整后资产。调整后资产的计算方法是总资产减去总贷款。濒临破产的银行，通常比健康的银行有更高的 NLLAA 值。

3）NPLTA——不良贷款总额除以总资产。不良贷款包括已超过期限 90 天的贷款和非应计贷款。濒临破产的银行，通常比健康的银行有更高的 NPLTA 值。

4）NLLTL ——净贷款损失除以贷款总额。濒临破产的银行有更高的贷款损失，因为他们往往向高风险的借款人发放贷款。因此濒临破产的银行，通常比健康的银行有更高的 NLLTL 值。

5）NLLPLNI——净贷款损失和贷款损失准备金之和除以收益总额。NLLPLNI 值越高，银行的业绩越差。

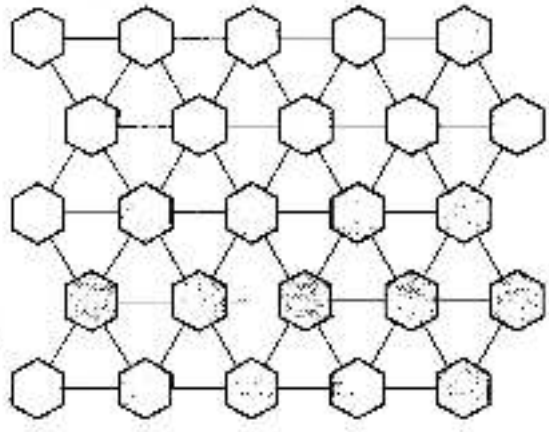


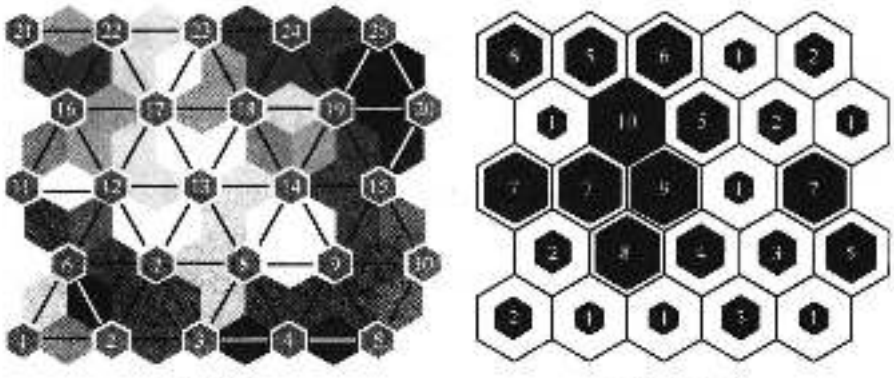
图 9.28 SOM 的结构

初步调查的统计数据显示，多家银行可能会遇到一些财政困难，聚类应该能够帮助我们找出有类似问题的银行集群。

图 9.28 显示了一个自组织神经网络（SOM），这个神经网络的 Kohonen 层是一个 25 个神经元的 5×5 的神经元阵列。注意，Kohonen 层的神经元组织为六边形的模式。

337

输入数据被归一化到 0～1 之间。网络以 0.1 的学习速率训练了 10 000 次迭代。训练完成以后，SOM 形成了一个语义映射，相似的输入向量映射后距离很近，不相似的输入向量映射后距离很远。换句话说，相似的输入向量趋向于映射到 Kohonen 层中相同或者相邻的神经元。这个 SOM 特性可以使用一个权重距离矩阵（又称为 U－矩阵）来可视化。图 9.29 显示了银行经济数据的 U－矩阵和 SOM 样本覆盖图。在 U－矩阵中，六边形代表 Kohonen 层中的神经元。相邻神经元之间区域的颜色表示它们之间的距离——颜色越深，距离越大。SOM 样本覆盖图揭示了 Kohonen 层中的每一个神经元所吸引的输入向量数目。



a) U-矩阵　　b) 样本覆盖图

图 9.29 5×5 的 SOM

通常，SOM 识别出的聚类数目比 Kohonen 层的神经元数目要少，因此被相邻神经元吸引的多个输入向量实际上可能属于同一个聚类。例如在图 9.29a 中，我们可以看到神经元 3 和 8、7 和 8、7 和 12、7 和 13、8 和 9、8 和 13、8 和 14、9 和 14、11 和 12、12 和 13、12 和 16、12 和 17、13 和 14、13 和 17、13 和 18、14 和 19、16 和 17、17 和 18、17 和 22、17 和 23、18 和 19、18 和 23、21 和 22、22 和 23 之间的距离相对较小（神经元之间区域的颜色很浅，因此距离较小）。因

338

此，我们有理由假设神经元3、7、8、9、11、12、13、14、16、17、18、19、21、22和23形成了一个单独的聚类。同时，我们也应该注意到神经元3和7之间的距离要比神经元3和8及神经元7和8之间的距离要大很多。因此，这对于找出与神经元3相关的输入向量和与神经元7相关的输入向量之间的区别是很有用的。表9.4显示了聚类的结果。

表9.4 5×5的SOM的聚类结果

聚类	规模	神经元编号	聚类的财务数据表													
			NITA			NLLAA			NPLTA			NLLTL			NLLPLLNI		
			均值	中位数	标准差	均值	中位数	标准差	均值	中位数	标准差	均值	中位数	标准差	均值	中位数	标准差
A	4	1, 6	0.0369	0.0369	0.0043	-0.1793	-0.1340	0.2516	0.0125	0.0100	0.0055	0.0050	0.0057	0.0036	0.2839	0.2839	0.0164
B	1	2	0.0121	0.0121	0	-0.4954	-0.4954	0	0.0323	0.0323	0	0.0006	0.0006	0	1.1522	1.1522	0
C	75	3,7,8,9,11, 12, 13, 14, 16, 17, 18, 19,21,22,23	0.0101	0.0094	0.0097	-0.0899	-0.0701	0.1646	0.0153	0.0144	0.0102	0.0143	0.0121	0.0093	0.8399	0.6973	0.7252
D	3	4	0.0066	0.0041	0.0064	0.4448	0.4528	0.0672	0.0190	0.0185	0.0058	0.0133	0.0145	0.0068	0.1894	0.1676	0.1617
E	13	5, 10, 15	-0.0006	-0.0010	0.0044	0.0363	0.0357	0.0257	0.0205	0.0166	0.0144	0.0388	0.0376	0.0108	8.0965	7.1786	3.9200
F	1	20	-0.0092	-0.0092	0	0.0089	0.0089	0	0.0215	0.0215	0	0.0055	0.0055	0	9.4091	9.4091	0
G	1	24	-0.0060	-0.0060	0	0.0199	0.0199	0	0.0198	0.0198	0	0.0662	0.0662	0	0.3612	0.3612	0
H	2	25	0.0014	0.0015	0.0019	0.0225	0.0225	0.0048	0.0164	0.0164	0.0029	0.0740	0.0740	0.0052	10.9785	10.9785	1.2720

聚类到底表示什么含义?

解释每一个聚类的含义通常是一个困难的任务。分类问题中的类数目是预先确定的，而在在基于SOM的聚类中，聚类的数目是未知的，而且给每一个聚类赋予一个类标签或解释往往需要一些先验知识和领域知识。

最开始，我们需要一个对比不同聚类的方法。在案例6中我们已经谈到，一个聚类的中心揭示了该聚类与其他聚类不同的特征。因此，确定一个聚类的平均成员，应该能够使我们揭示整个聚类的含义。表9.4包含了本案例中用到的CAMELS评级的平均值，中位值和标准差（SD）。利用这些值，领域专家可以识别出具有相同行为模式或者面临类似问题的银行。

下面我们通过识别出具有负资产回报的问题银行来开始我们的分析。从表9.4中我们可以发现E、F和G三个聚类的NITA值为负。例如，聚类E中的银行的平均净亏损是其总资产的6%。另一方面，健康的银行往往拥有正向的总资产。因此，聚类A中的银行NITA值很高，可以被认为是健康的银行。

按照NLLAA评级，聚类D拥有最高的平均值，其次是聚类E。注意，虽然聚类E中的银行是问题银行，但是他们的NLLAA值比聚类D至少要低12倍（平均3.63%对比44.48%）。这可以看做是一个明确的指示，那就是聚类D中的银行面临严重的贷款问题（虽然他们的资产当前为正值）。聚类A、B和C的NLLAA值为负值，对于健康的银行来说这是很常见的。

聚类B的NPLTA值比较高，其次是聚类E、F和G中的问题银行。这可能指示聚类B中的银行正面临着贷款回收的问题。实际上，处于这种状况的银行比聚类E、F和G中的银行更加糟糕。

339～340

最后，聚类H中的两个银行的NLLTL值和NLLPLLNI值最高，其后是问题银行。因为，高的NLLTL值意味着高的贷款损失，我们可以发现聚类H中的银行倾向于给高风险的客户提供贷款。高的贷款风险也使得这些银行的状况不佳，这点可以从高的NLLPLLNI值中得到反映。

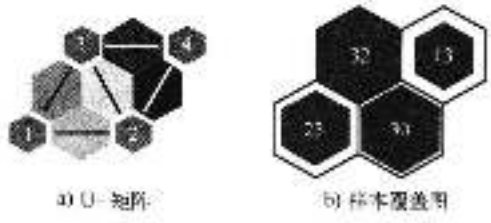
聚类分析的一个重要方面是识别离群点，即那些不属于任何大的聚类的对象。如图9.4所示，有3个银行可以看做是孤立的聚类——聚类B、聚类F和聚类G。这些银行就是离群点，他

们都有自己唯一的财务状况。传统的聚类算法，例如 $K-$means 聚类，不能很好地处理离群点，SOM 却可以轻易地识别。

需要多少个聚类？

在一个多维数据集中，很难确定聚类的数目。实际上，当一个聚类算法产生一个大的聚类时，离群点便会被吸收到这些聚类中。这样不仅会产生较差的聚类结果，而且，更糟糕的是，无法识别唯一的对象。

例如，我们现在使用一个 2×2 的 SOM 来对同一个银行集进行聚类。SOM 经过 1000 次迭代训练。图 9.30 显示了 U－矩阵和 SOM 样本覆盖图。很显然，现在我们只有 4 个聚类。进一步的调查分析显示，基于平均值，与神经元 1，2 和 3 关联的银行可以分类为健康的银行，而与神经元 4 关联的 13 个银行可以分类为问题银行。由 5×5 的 SOM 识别出的拥有高的 NLLPLLNI 值的两个问题银行和两个正常银行，现在都被“健康”聚类吸收。



a) U－矩阵　　b) 样本覆盖图

图 9.30　训练后 2×2 的 SOM

如何测试 SOM 的性能？

为了测试包括 SOM 在内的神经网络，我们需要一个测试集。从 FDIC 的人工报表中我们可以得到一份问题银行的报表，而且可以收集到一个合适的金融陈述数据。银行破产的一些研究指出，一般的银行破产能在实回日期之前 6～12 个月被检测出，有些情况下可以提前到银行破产之前 4 年（Alam et al.，2000；Eichengreen，2002）。虽然偿付能力和流动资产是在接近实回日期时最重要的预测因素，资产质量、收入和管理方法在破产事件接近的时候同样显得尤为重要。为了测试 SOM 的性能，我们选择了去年破产的 10 家银行，并且收集了这些银行破产前一年的金融陈述数据。表 9.5 显示了 CAMELS 评分的平均值、中位值和标准差（SD）。现在我们可以使用 10 个输入向量来查看 SOM 的响应。 [341]

表 8.5　破产银行的财务状况

NITA			NLAA			NPETA			NLETL			NLLPLLNI		
均值	中位数	标准差	均值	中位数	标准差	均值	中位数	标准差	均值	中位数	标准差	均值	中位数	标准差
−0.0025	−0.0046	0.0085	0.0642	0.0630	0.0234	0.0261	0.0273	0.0065	0.0841	0.0859	0.0052	7.3467	6.9561	3.8461

跟预期结果一样，在 2×2 的 SOM 中，10 个输入向量都被神经元 4 所吸引。而在 5×5 的 SOM 中，情况更加复杂。6 个输入向量被神经元 5 吸引，有 2 个输入向量被神经元 10 所吸引，有 1 个输入向量被神经元 20 所吸引，还有两个输入向量被神经元 24 所吸引。在两种情况下，破产银行都被正确地聚类。 [342]

最后，有一点需要我们的注意。虽然 SOM 是一个很强大的聚类工具，每一个聚类的确切含义却不总是很清晰的，通常我们需要一个领域专家来解释聚类结果。同时 SOM 只是神经网络的一种，任何神经网络的性能都取决于用于训练网络的数据的好坏。在本案例中，我们只使用了 5 个财务变量，我们可能需要更多包含银行状况的额外信息的变量（工业界的研究者通常基于 CAMELS 评级系统使用 29 个以上的财务变量）。

Berry 和 Linoff（2004）给出了与聚类相关的一个挑战的例子。某家大型银行决定增加其房屋净值贷款的市场份额。银行分别收集了 5000 个已经拥有房屋净值贷款和 5000 个没有房屋净值贷款的客户数据。这些数据包括房屋价值，amount of credit available，amount of credit granted，客户年龄、婚姻状况、小孩数量和家庭收入等。这些数据被用来训练出一个 SOM。SOM 找出了 5 个

聚类。其中的一个聚类特别有意思。它包含已经拥有房屋净值贷款的客户。这些客户年龄都在40岁以上、已婚且小孩都处于少年晚期。银行于是假设这些客户贷款的目的是为了给小孩提供大学学费。于是，该银行进行了一次促销活动，将房屋净值贷款提供为一种解决大学教育费用的方式。然而，这次促销活动的结果却让人很失望。

进一步的调查分析显示问题还是出现在对SOM识别出的聚类的解释上。于是，银行决定加入更多的客户信息，例如账户类型、存款保险体系、信用卡系统等。重新训练SOM以后才发现，那些在40岁以后且小孩处于大学阶段的客户往往还拥有一个商业账户。因此，银行总结出，那些客户是在他们的小孩离开家上大学以后，父母申请房屋净值贷款是为了开始新的商业活动。于是，这家银行针对这些潜在客户群体进行了一次促销活动，这次促销活动很成功。

9.5 遗传算法可以解决的问题

遗传算法适用于许多的优化问题（Haupt and Haupt, 2004）。优化本质上是寻找一个问题的更好的解决方案的过程。这意味着问题往往有一个以上的解决方案而且不同方案的质量不是相等的。遗传算法生成一系列相互竞争的候选方案，然后让这些方案在自然选择过程中进化——较差的方案趋于灭绝，而较好的方案得以生存并繁殖。通过不断重复这一过程，遗传算法会生成一个最优的方案。

343

案例8：旅行商问题

我想要开发一个智能系统来产生最佳的旅行路线。我打算自驾游，要求游览完西欧和中欧的所有主要城市，然后回家。遗传算法可以解决这个问题吗？

这个问题就是著名的旅行商问题（TSP）。给定固定数目的城市N和每对城市间的旅行费用（或者距离），我们需要找出游览每个城市刚好一次，且最终返回出发点的最便宜的方案（或者最短路径）。

虽然旅行商问题最早在18世纪就已经为人们所知，但是直到20世纪40年晚期到20世纪50年代早期这个问题才被正式地研究，而且被证明是一个典型的NP难问题（Dantzig et al., 1954; Flood, 1955）。这类问题使用经合搜索技术很难解决。TSP问题的搜索空间包括N个城市的所有可能的组合，因此搜索空间的大小是N（城市数目N的连乘）。因为城市数目可能会很大，一次检测一条路径的方法不具有伸缩性。

TSP可以自然地用来表示很多交通和逻辑应用，例如校车路线安排、居家人士的肉类递送、仓库码垛机调度、邮政车路线规划以及很多其他的应用。TSP的一个经典例子是电路板钻孔机器的调度。在这个例子中，孔洞代表城市，旅行费用是将钻头从一个孔洞移动到另一个孔洞所耗费的时间。

随着时间的推移，TSP问题的规模也在显著扩大，从最开始的49个城市的方案（Dantzing et al., 1954），发展到最近的15112个城市的问题（Applegate et al., 2001）。

研究人员使用不同的技术来解决TSP问题。这些技术包括模拟退火（Laarhoven and Aarts, 2009）、离散线性规划（Lawler et al., 1985）、神经网络（Hopfield and Tank, 1985）、分支定界算法（Tschoke et al., 1995）、马尔科夫链（Martin et al., 1991）和遗传算法（Potvin, 1996）。遗传算法特别适合解决TSP问题，因为它能快速地将搜索指向搜索空间的最佳区域。

如何用遗传算法解决TSP问题？

首先，我们需要决定如何表示一条旅行商路线。最自然的路线表示方法是路径表示法（Michalewicz, 1996）。每一个城市被赋予一个字母或者数字名称，经过这些城市的路线表示为一条染色体，然后使用适当的遗传操作来创建新的路线。

假设我们有编号从1~9的9个城市。在一条染色体中，整数的顺序代表了对应的城市被旅行商访问的顺序。例如，下面这条染色体

1	6	5	3	2	8	4	9	7

代表的是如图9.31所示的路线。旅行商从城市1开始，访问所有其他的城市一次，最后返回到出发点。

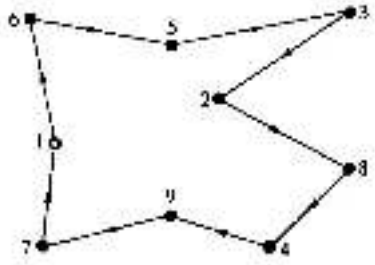
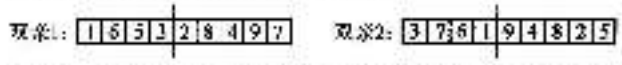


图9.31 一条旅行商路线的例子

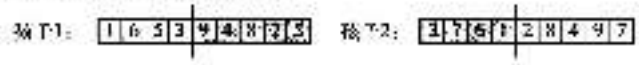
344

TSP问题中如何执行交叉运算？

传统形式的交叉运算不能直接应用于TSP问题中。对两个双亲做简单地部分交换将会产生包含重复和遗漏的非法路线——有的城市可能会被访问两次，有的可能一次都没有访问到。例如，交换以下两条双亲染色体的一部分



将会产生两条路线，其中的一条路线中城市5被访问两次而城市7却没有被访问，另一条路线中城市7被访问了两次而城市5却没有被访问。



显然，只带一个交叉点的交叉运算并不适用于TSP问题。为了克服这个问题，一些包含两个交叉点的交叉运算被提出（Goldberg, 1989）。例如，Goldberg和Lingle（1985）提出了部分映射交叉运算，Davis（1985）提出了顺序交叉运算。然而，这些运算大部分都是从一个亲代中选择部分路线的同时，保留从另一个双亲选择的城市的顺序，从而创建新的后代。图9.32展示了这些交叉运算的执行流程。

首先，在表示两个双亲染色体的字符串中随机选择两个交叉点（用符号"|"标记）。两个交叉点之间的染色体内容定义了互换子段。通过交换两个亲代的互换子段，我们便创建出了两个后代染色体。在图9.32中星号代表是的尚未确定的城市。其次，每一个亲代中的原始城市在后代中都保持其原有的顺序，然后删除另一个亲代中的互换子段。例如，城市1、9、4和8，出现在第二个亲代的互换子段中，在第一个亲代中他们被删除。剩余的城市则被放置到后代中，并保持其原有顺序。最终的结果是，一个后代表示的路线由它的两个亲代各自部分决定。

345

TSP问题中如何执行变异运算？

变异运算一共有两种类型：倒数互换和反转（Michalewicz, 1996）。图9.33a和图9.33b分别展示了这两种变异运算的执行原理。互换运算从染色体中随机选择两个城市，然后简单地交换两个城市的位置。反转运算在染色体字符串中随机选择两个点，然后将这两个点之间的城市的顺序变为逆序。

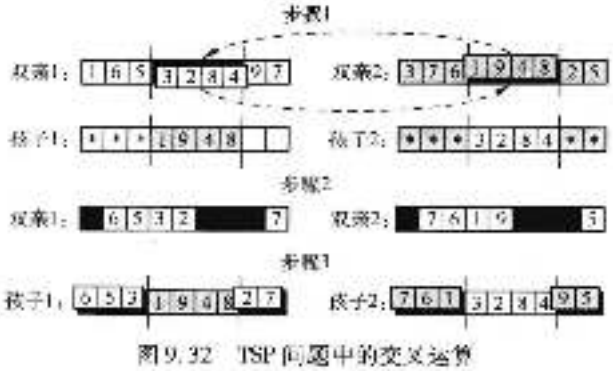


图9.32 TSP问题中的交叉运算



图9.33 TSP问题的变异运算

TSP 问题中如何定义适应性函数?

为 TSP 问题创建遗传运算并不是一件容易的工作，但是定义适应函数却是很直接的——我们需要做的仅仅是估计路线的总长度。每一个单独的染色体的适应性由路线长度的倒数决定。换句话说，路线的长度越短，相应染色体的适应能力越强。

一旦适应函数和遗传运算定义完成，就可以开始实现和运行遗传算法。

作为例子，我们考虑在 1 × 1 单位面积下的 20 个城市的 TSP 问题。首先，我们选择染色体群体的大小和需要运行的遗传代数。开始时我们可能会选择相对较小的群体和少量的遗传代数来检验得到的解决方案。图 9.34 显示了由 20 个染色体运行 100 代之后产生的最佳路线。我们可以看到，这条路线并不是最优路线，显然还可以得到改进。让我们增大群体中染色体的数量并再次运行遗传算法。图 9.35a 显示了运行结果。路线的总长度下降了 20%，这是非常显著的改进。

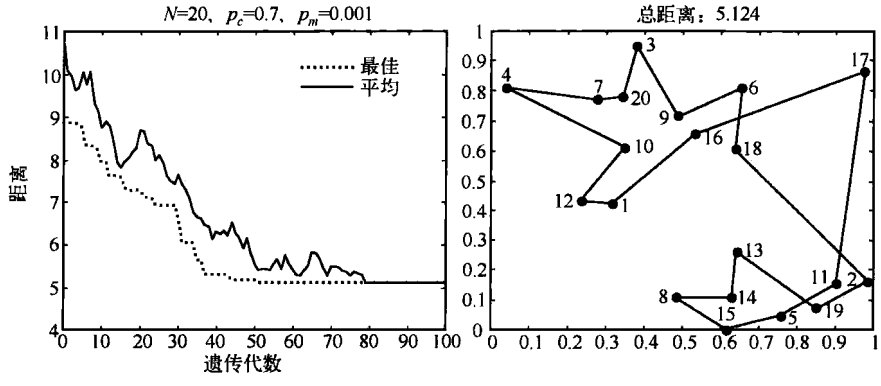
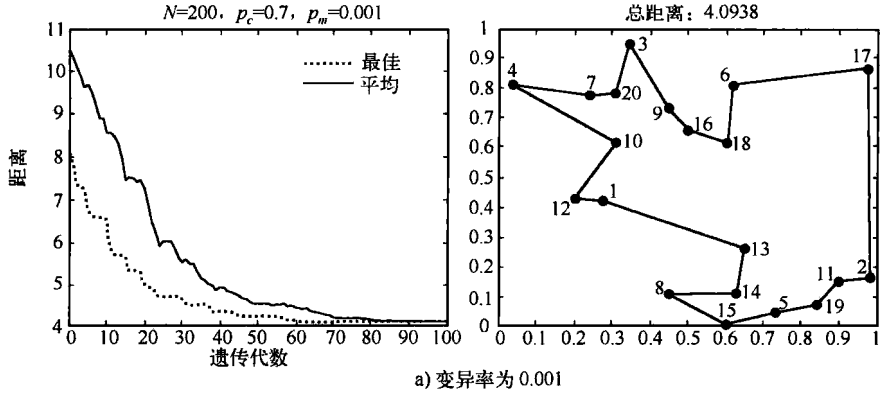
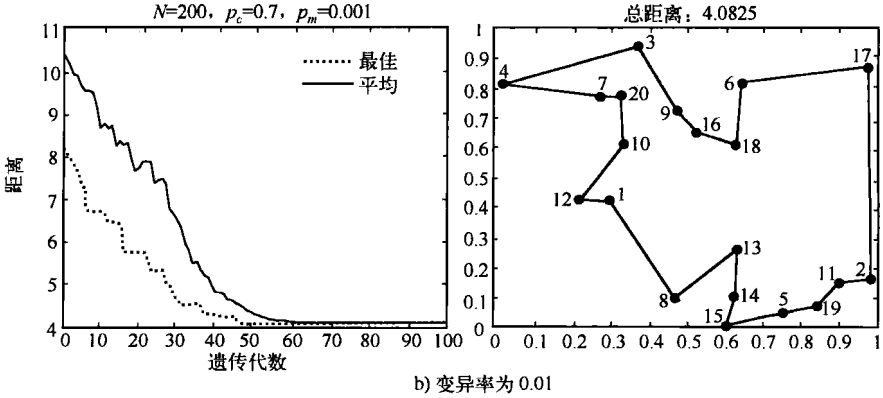


图 9.34 性能图和大小为 20 的染色体群体经过 100 代遗传后生成的最佳旅行商路线



a) 变异率为 0.001



b) 变异率为 0.01

图 9.35 性能图和大小为 200 的染色体群体生成的最佳路线

如何确定遗传算法已经真正找到了最优路线？

事实上，我们无法确定遗传算法何时能够找到最优路线。只有在不同的染色体群体大小上使用不同的交叉率和变异率，通过进一步尝试运行遗传算法，才能获得答案。

例如，我们将变异率提高到0.01。图9.35b显示了运行结果。虽然总距离有少量的下降，但是旅行商路线却和图9.35a中显示得类似。现在我们可能会尝试增大染色体群体的大小，然后重新运行遗传算法。然而，我们几乎不可能获得一个明显更好的方案。我们可以确定这条路线就是最优的吗？当然，我们并不奢望一条最优的路线！然而，经过几次运行以后，我们完全可以确定所获得的路线是一条好的路线。

346 / 347

9.6 混合智能系统可以解决的问题

为了解决现实世界中的复杂问题，我们需要一个综合了专家系统、模糊逻辑、神经网络和进化计算的优点的复杂智能系统。这种系统可以集成类似人类的特定领域的专业知识，并且拥有学习和适应快速变化环境的能力。虽然混合智能系统这个领域还在不断发展，而且大部分混合工具并不是特别有效，神经网络模糊系统却已经成为一种用于很多成功应用的成熟的先进技术。神经网络可以从数据中学习，而模糊逻辑的主要优势在于其对人类决策的建模能力。

案例9：神经模糊决策支持系统

我想开发一个通过心脏图像来诊断心肌灌注的智能系统。我拥有一组心脏图像和临床诊治记录，以及医生的解释。混合系统可以解决这个问题吗？

现代心脏医学中的诊断是基于分析SPECT（单光子发射计算机化断层扫描）图像。通过向病人注射放射性示踪剂，获得两套SPECT图像：一套成像于注射10～15分钟以后应力最大的时候（应力图像），另一套则成像于注射后2～5个小时以后（休息图像）。放射性示踪剂在心脏肌肉中的分布和肌肉灌注成正比。因此，通过对比应力图像和休息图像，心脏病专家通常可以检测心脏功能的异常。

SPECT图像通常表示为一个具有256级灰度的高分辨率黑白图片。图像中的较亮的斑点对应心肌灌注区域，而更暗的斑点则指示可能存在局部贫血。不幸的是，SPECT图像的视觉检查是非常主观的，不同医生的解释往往是不一致的，因而很容易发生错误。显然，能够帮助心脏病专家诊断心脏SPECT图像的智能系统将会有很大的价值。

在这个案例中，我们使用267个心脏病诊断病例。每一个病例包含两个SPECT图像（应力图像和休息图像），且被分为22个区域。区域的亮度，反过来反映了该区域内的灌注，表示为一个0～100之间的整数（Kurgan et al.，2001）。因此，每个心脏病诊断病例表示为44个连续性特征和一个代表整体诊断——正常或异常的二元特征。 348

整个SPECT数据集由55个被划分为正常（阳性样本）的病例和212个被划分为异常（阴性样本）的案例组成。这个数据集被划分为训练集和测试集。训练集包括40个阳性样本和40个阴性样本。测试集包括15个阳性样本和172个阴性样本。

我们可以训练一个反向传播神经网络来将SPECT图像划分为正常和异常吗？

反向传播神经网络确实可以解决SPECT图像分类问题——训练集的大小是足够大的。神经网络在这里可以看做一个分类器。输入层中的神经元数目由应力图像和休息图像中区域的总数目决定。在这个例子中，每个图像被划分为22个区域，所以我们需要44个输入神经元。因为SPECT图像将被分类为正常或异常，我们应该使用2个输出神经元。多次的实验尝试显示隐含层中仅仅使用5～7个神经元时能够获得很好的泛化能力。反向传播网络的学习相对较快，且能收敛到一个方案。

然而，当我们在测试集中测试网络时，我们发现网络的性能相当差——大约25%的正常的心脏病诊断病例被误分类为异常，35%以上的异常病例被误分类为正常；总体的诊断错误超过33%。这意味着训练集中可能缺少某些重要的样本（神经网络性能由训练网络的样本决定）。尽管如此，我们还是可以显著地提高诊断的准确性。

首先，我们需要重新定义问题。为了训练网络，我们使用相同数目的阳性样本和阴性样本。虽然在实际的临床试验中正常和异常的SPECT图像的比例很不相同。异常心脏病病例的误分类比正常的心脏病病例的误分类将会导致更加严重的后果。因此，为了得到一个小的异常的SPECT图像的分类误差，我们可能允许正常图像有相对大的误差。

神经网络将会产生两个输出。第一个输出对应于SPECT图像属于normal（正常）类的可能性，第二个输出对应于图像属于abnormal（异常）类的可能性。例如，如果第一个输出（正常）为0.92且第二个输出（异常）为0.16，相应的SPECT图像被分类为正常，而且我们可以得出对应病例心脏病发作的风险很低的结果。另一方面，如果normal的输出值很低（假设为0.17）而abnormal的输出值要高得多（假设0.51），相应的SPECT图像被分类为异常，而且我们可以得出对应病例心脏病发作的风险很高的结论。然而，如果两个输出值很接近——假设normal输出值为0.51，abnormal输出值为0.49，我们就不能很自信地对图像进行分类。

我们可以使用模糊逻辑来进行医疗诊断中的决策吗？

医生对SPECT图像进行分类时心中并没有精确的阈值。心脏病专家检查诊断图像中所有区域的灌注，并且对比应力图像和休息图像中相应的心肌区域的亮度。实际上，医生通常依靠他们的经验和直觉来检测心肌异常。模糊逻辑为我们提供了一种对心脏病专家评估心脏病发作风险进行建模的手段。

[349]

要建立一个模糊系统，我们首先需要确定输入和输出变量，定义模糊集和构建模糊规则。对于我们的问题，有两个输入（NN output 1和NN output 2）和一个输出（心脏病发作的风险）。输入变量被归一化到区间[0，1]内，输出的变化范围可以从0－100%。图9.36、图9.37和图9.38显示了模糊系统中用到的语言变量的模糊集。模糊规则如图9.39所示。

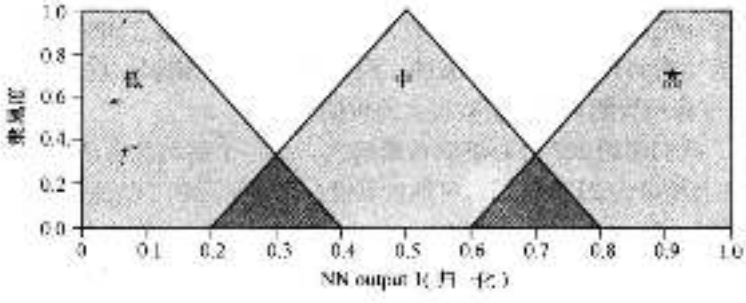


图9.36 神经网络正常输出的模糊集

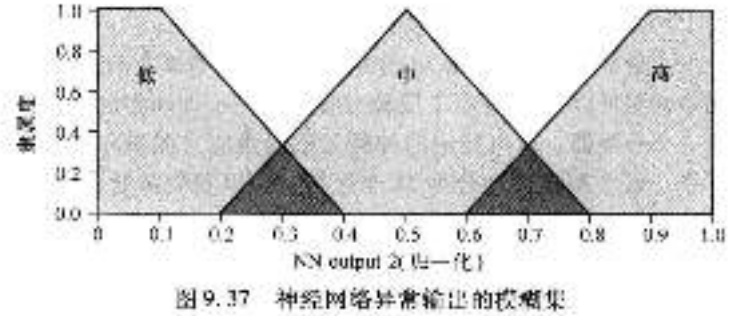


图9.37 神经网络异常输出的模糊集

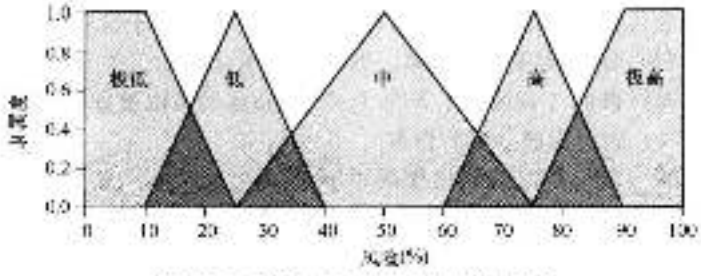


图 9.38 语言变量"风险"的模糊集

```
1. If (NN_output1 is Low) and (NN_output2 is Low) then (Risk is Moderate)
2. If (NN_output1 is Low) and (NN_output2 is Medium) then (Risk is High)
3. If (NN_output1 is Low) and (NN_output2 is High) then (Risk is Very_high)
4. If (NN_output1 is Medium) and (NN_output2 is Low) then (Risk is Low)
5. If (NN_output1 is Medium) and (NN_output2 is Medium) then (Risk is Moderate)
6. If (NN_output1 is Medium) and (NN_output2 is High) then (Risk is High)
7. If (NN_output1 is High) and (NN_output2 is Low) then (Risk is Very_low)
8. If (NN_output1 is High) and (NN_output2 is Medium) then (Risk is Low)
9. If (NN_output1 is High) and (NN_output2 is High) then (Risk is Moderate)
```

图 9.39 心脏疾病风险评估的模糊规则

图 9.40 表示的是用于心脏疾病风险评估的神经模糊决策支持系统的完整结构。为了构建该系统，我们可以使用 MATLAB 神经网络工具包和 MATLAB 模糊逻辑工具箱。系统开发完成以后，我们可以使用图 9.41 所示的三维图来研究和分析它的表现。

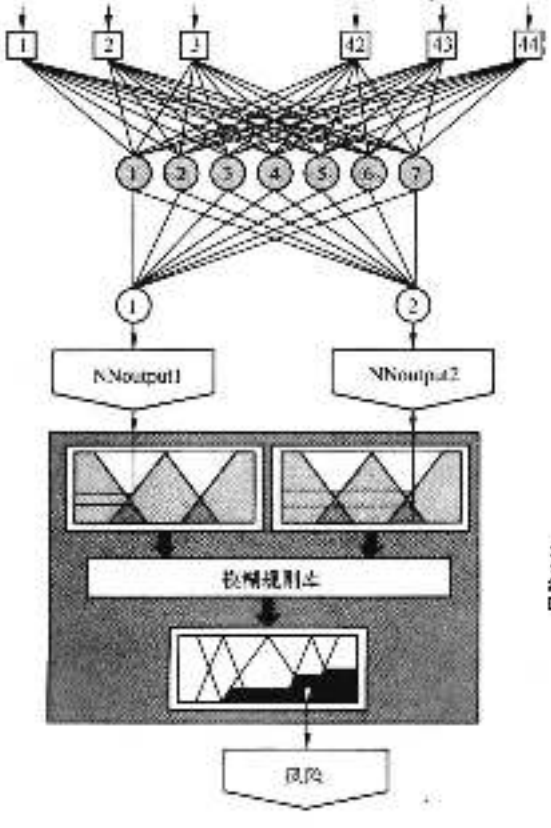


图 9.40 用于心脏疾病风险评估的神经模糊决策支持系统的层次结构

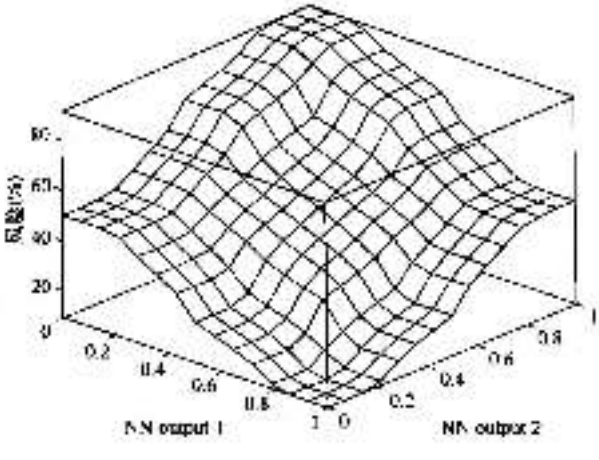


图 9.41 模糊规则库的三维图

系统的输出是一个代表病人心脏病发作风险的明确数值。基于这个数值，心脏病专家现在能以更高的确定性来对心脏病病例进行分类，当风险被量化以后，决策者作出正确决策的可能性更高。例如，如果风险很低（假设低于30%），心脏病病例可以被分类为正常，但是如果风险很高（假设高于50%），病例则被分类为异常。

然而，风险为30% ~50%的心脏病病例既不能被分类为正常，也不能被分类为异常——相反，这样的病例是不确定的。

我们可以利用经验丰富的心脏病专家的知识，来对一些不确定病例进行分类吗?

经验丰富的心脏病专家知道，在正常的心肌中，相同区域内应力最大时的灌注通常比休息时的灌注要高。因此，通过对相应的区域应用以下的一些假设，我们可以使不确定的病例更加确定：

1）在区域 i 内，如果应力时的灌注高于休息时的灌注，那么心脏病发作的风险应该被降低。

2）在区域 i 内，如果应力时的灌注不比休息时的灌注高，那么心脏病发作的风险应该被提高。

这些规则在诊断系统中可以实现如下：

步骤 1：将心脏病病例提供给神经 - 模糊系统来分析。

350 ~ 352

步骤 2：如果系统的输出低于30%，将提供的病例分类为正常并且结束。如果输出高于50%，将病例分类为异常并且结束。否则，转到步骤3。

步骤 3：对于区域1，用应力时的灌注减去休息时的灌注。如果结果为正，将当前的风险乘以0.99来降低风险。否则，将当前风险乘以1.01来提高风险。对所有的22个区域重复这个过程，然后转到步骤4。

步骤 4：如果新的风险值低于30%，将病例分类为正常；如果新的风险值高于50%，将病例分类为异常；否则，将病例分类为不确定。

现在，如果我们将测试集应用到神经 - 模糊系统中，我们会发现诊断的准确性得到了显著提高——整体的诊断误差没有超过5%，而且只有3%的人异常心脏病病例被误分类为正常。虽然我们没有提高系统在正常病例上的诊断性能（仍然有超过30%的正常病例被误分类为异常），而且总病例数量的20%以上被分类为不确定，神经 - 模糊系统在对SPECT图像进行分类时得到的结果甚至比心脏病专家都要好。最重要的是，异常的SPECT图像现在以更高的准确性被识别。

在这个例子中，神经模糊系统拥有一个异质结构——神经网络和模糊系统为独立的部件工作（虽然在解决这个问题时，它们互相协作）。当一个新的病例被提供给诊断系统时，神经网络确定了模糊系统的输入。然后，模糊系统利用预先定义好的模糊集和模糊规则，将输入映射到输出，从而得到心脏病发作的风险。

存在成功的同质结构的神经 - 模糊系统吗?

一个典型的带有同质结构的神经模糊系统的例子是适应神经模糊推理系统（ANFIS）。ANFIS不能被划分为独立不同的部分。实际上，ANFIS是一个可以进行模糊推理的多层神经网络。

案例 10：时序预测

我想开发一个工具，用于预测飞行器在降落到航空母舰过程中的轨迹。我有一个不同飞行员驾驶不同飞行器的降落轨迹的数据库，同时我使用能够提供降落飞行器的实时轨迹的雷达数值数据。我的目标是基于飞机的当前位置，至少提前2秒预测飞行器的轨迹。神经模糊系统能够解决这个问题吗?

飞行器的降落，尤其是降落到航空母舰中，是一个极其复杂的过程。它受到很多因素的影响：飞行甲板的空间限制和运动情况（包括倾斜程度和摇摆程度），飞行条例和燃料负载，持续

机械准备和最关键的时间限制等。航空母舰有大约10英尺的上下颠簸，因此甲板有20英尺的位移。此外，在夜间或者风雨天气下，很难看到接近的飞行器。 353

负责飞机的最后接近和降落的是降落信号官（LSO）。实际上，拒绝降落这一最重要的决定是由LSO而不是飞行员作出的。当飞行器离甲板的距离为1海里（大概相当于60秒的时间）时，需要对其进行仔细的观察和指导。在这段关键的时间内，LSO至少需要提前2秒预测飞行器的轨迹。这类问题就是数学中的时间序列预测问题。

什么是时间序列？

时间序列可以定义为一个观测的集合，其中的每个观测都在一个特定的时间被记录。例如，我们可以通过记录某一个时间间隔（飞行器降落前60秒）内飞行器的位置来得到一个时间序列。现实世界中的时间序列问题往往是非线性的，而且经常存在混乱的行为，因而很难对其进行建模。

飞行器降落轨迹预测主要依靠LSO的经验（所有的LSO都是经过训练的飞行员）。自动预测系统能够使用航空母舰中的雷达提供的飞行器的位置数据，以及之前的飞行员驾驶不同飞行器降落的数据记录（Richards, 2002）。系统使用过去的数据离线训练。当飞行器当前的移动数据在线提供给系统以后，系统要能够预测未来几秒内飞行器的移动。飞行器轨迹预测的时间序列如图9.42所示。

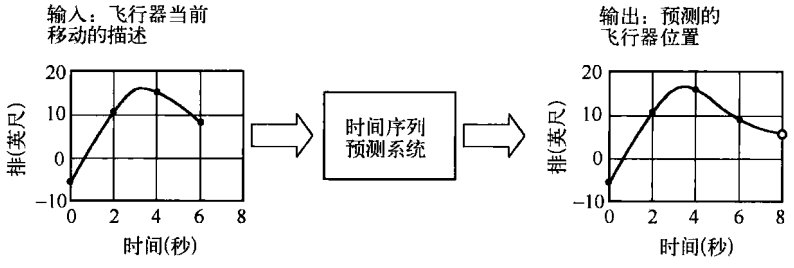


图9.42 飞行器轨迹的在线时间序列预测

为了在线预测飞行器的位置，我们将使用ANFIS。它将从给定的时间序列数据中学习，从而确定使系统最好追踪飞行器轨迹的隶属函数的最佳参数。

ANFIS的输入是什么？

我们使用已知的现有值来预测时间序列的未来值。例如，如果我们要提前2秒预测飞行器的位置，我们需要使用飞行器当前的位置数据以及前2、4、6秒的数据记录。这4个已知的数据组成了一个输入模式——一个4维向量，其形式如下： 354

$$\boldsymbol{x} = [x(t-6)x(t-4)x(t-2)x(t)]$$

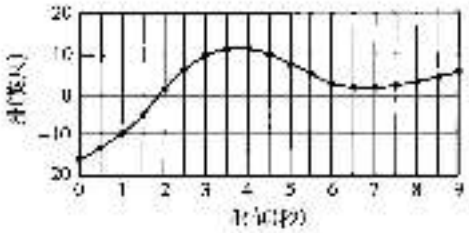
其中，$x(t)$为在时间t处飞行器的位置。

ANFIS的输出对应于预测的飞行器轨迹：飞行器2秒以后的位置$x(t+2)$。

在本案例中，我们使用10个飞行器的降落轨迹——其中5个用来训练，另外5个用来测试。每一个轨迹就是一个时间序列，这个时间序列是在飞行器降落前60秒的时间内每隔半秒记录一次的位置数据。因此，数据集中的每一个轨迹包含121个值。

如何构建用于训练ANFIS的数据集？

考虑图9.43，它显示了一条飞行器轨迹和用轨迹数据每隔2秒采样一次而得到的一个3×5的训练数据集。输入变量x_1、x_2、x_3和x_4分别对应于时间$(t-6)$、$(t-4)$、$(t-2)$和t处飞行器的位置。输出的期望值对应于提前2秒的预测，$x(t+2)$。图9.43中显示的训练数据集构建于t取值为6秒（第一行），6.5秒（第二行）和7.0秒（第三行）。



输入				期望输出
x_1	x_2	x_3	x_4	
17.4	2.1	11.0	3.9	4.2
−12.9	7.5	10.1	2.1	4.9
10.0	9.8	8.2	2.0	5.1

图 9.43 飞行器轨迹和训练 ANFIS 的数据集

在 60 秒的时间间隔上对飞行器降落轨迹使用同样的过程，我们可以得到与 105 × 5 的矩阵表示的 105 个训练样本。因此，用来训练 ANFIS 的整个数据集可表示为一个 525 × 5 的矩阵。

需要给每一个输入变量指定几个隶属函数?

一个实践中用到的方法是选择最少的隶属函数。因此，我们可以从给每个输入变量指定 2 个隶属函数开始。

图 9.44 显示了一条实际的飞行器轨迹和经过 1 和 100 次迭代训练得到的 ANFIS 的输出。如图所示，即使是迭代 100 次，ANFIS 的性能依然不能令人满意。我们也可以看出，增加迭代次数并没有明显改进 ANFIS 的性能。

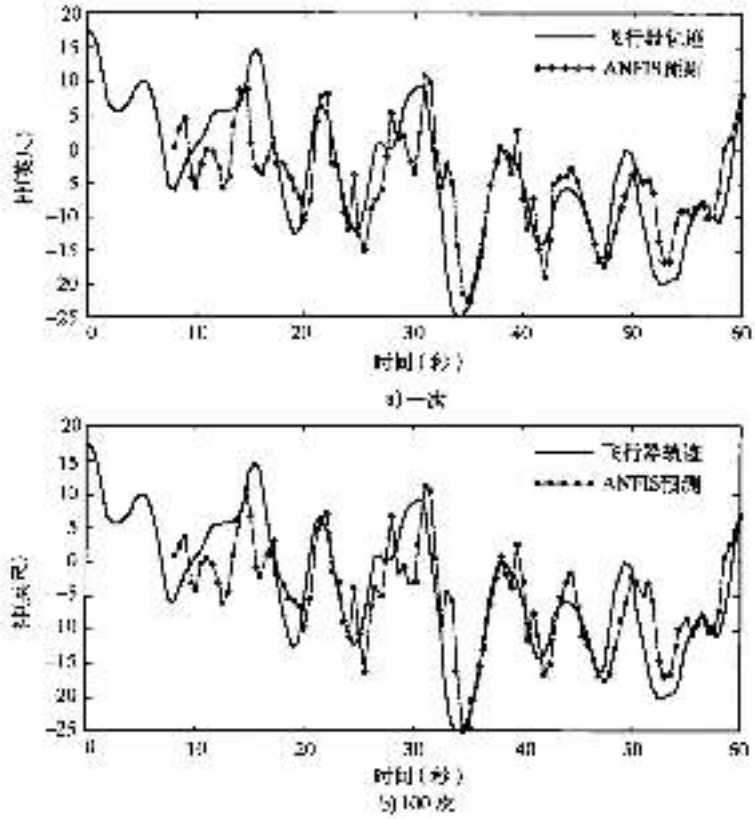


图 9.44 带有 4 个输入变量且每个输入变量指定 2 个隶属函数的 ANFIS 的性能

如何改进 ANFIS 的性能?

通过给每一个输入变量指定 3 个隶属函数可以明显改善 ANFIS 的性能。图 9.45 显示了结果：仅在 1 次迭代训练后，ANFIS 预测的飞行器轨迹就比之前已经迭代 100 次的训练得到的结果还要准确。

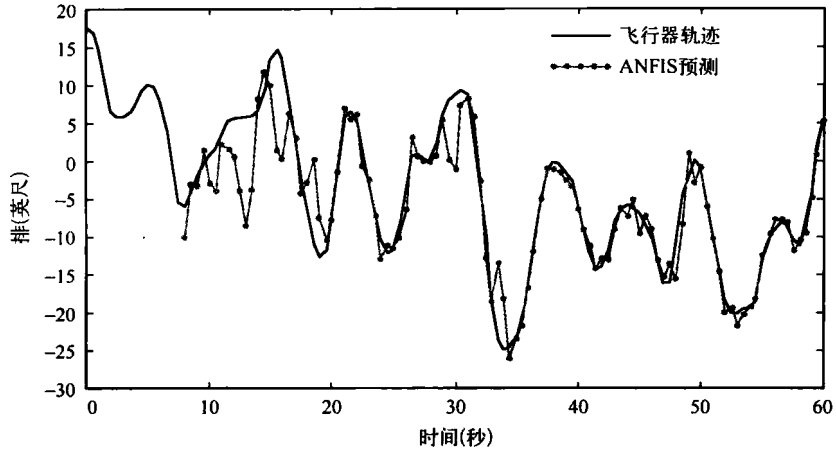


图 9.45 带有 4 个输入变量且每个输入变量指定 3 个隶属函数的 ANFIS 在一次迭代后的性能

另一种改善时间序列预测性能的方法是增加输入变量的个数。例如，考虑一个具有 6 个输入变量的 ANFIS，其中每一个变量分别对应于飞行器在时间 $(t-5)$、$(t-4)$、$(t-3)$、$(t-2)$、$(t-1)$ 和 t 处的飞行位置。ANFIS 的输出还是提前 2 秒的预测。现在，训练数据集用一个 535 × 7 的矩阵表示。

为每个输入变量指定了最小数目的隶属函数以后，我们就可以进行一次迭代训练 ANFIS，并用一个测试轨迹来观察它的性能。如图 9.46 所示，带有 6 个输入变量的 ANFIS 的性能比带有 4 个输入变量的 ANFIS 的性能要好，且仅仅在一次迭代训练后就能提供满意的预测结果。

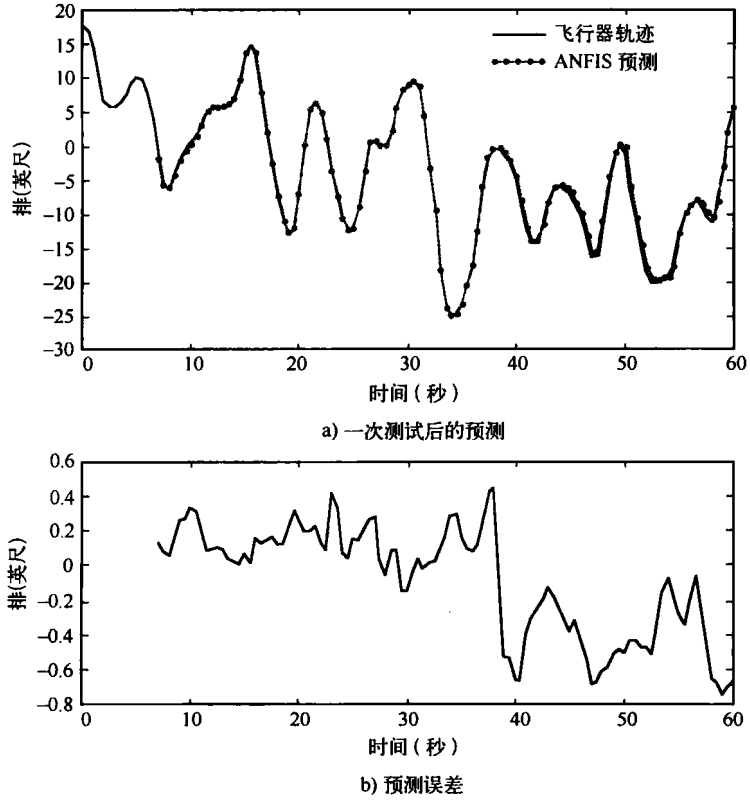


图 9.46 带有 6 个输入变量且每个输入变量指定 2 个隶属函数的 ANFIS 的性能

9.7 小结

355 ~ 357

本章主要研究知识工程。首先，我们讨论了智能系统所能够解决的问题类型，介绍了知识工程过程的6个步骤。其次，我们分析了专家系统、模糊系统、神经网络和遗传算法的一些典型应用。我们演示了如何构建智能系统来解决诊断、选择、预测、分类、聚类和优化问题。最后，我们讨论了混合神经－模糊系统在决策支持和时序预测方面的应用。

本章的主要内容有：

- 知识工程是构建基于知识的智能系统的过程。它主要有6个步骤：问题评估、知识和数据获取、原型系统开发、完整系统开发、系统评价和修订、系统集成和维护。
- 智能系统主要用于诊断、选择、预测、分类、聚类、优化和控制等问题。智能系统的构建工具的选择主要取决于所要解决的问题的类型、数据的可用性、专家意见以及解决方案的形式和内容。
- 理解问题域是构建智能系统的关键。开发原型系统可以帮助我们测试对问题的理解程度并确保我们的问题解决策略、构建工具的选择、数据和知识的表示技术能适合我们将要解决的任务。
- 与传统的计算机程序不同，智能系统主要用来解决那些没有明确定义“正确”和“错误”解决方案的问题。因此，系统通常要通过用户选择的测试用例来进行评价。
- 智能系统擅长于解决诊断和故障排除问题。诊断专家系统相对来说是比较容易进行开发的。大多数的诊断问题有一个有限的解决方案的列表，并且只涉及很少量的良好形式化的知识，人类专家通常只需要很少的时间就能解决这样的问题。
- 解决真实世界中的分类问题时往往会遇到不正确和不完整的数据。专家系统可以通过增量获取的论据和不同信任度的信息来处理这样的数据。
- 模糊系统很适合模拟人类的决策过程。重要的决策通常是基于人们的直觉、常识和经验，而不是基于数据的可用性和精确性。模糊技术提供了一种处理“软标准”和“模糊数据”的方法。虽然决策支持模糊系统可能包含几十甚至上百条规则，但是它却可以很快地进行开发、测试和实现。
- 神经网络是一类已经被成功地用于解决预测、分类和聚类问题的通用工具。神经网络主要应用于语音和字符识别、医疗诊断、过程控制、机器人技术、雷达目标识别、汇率预测和交易欺诈检测等领域，并且其应用领域还在快速扩大。

358

- 多层反向传播神经网络特别善于解决预测和分类问题。但是训练样本的选择也很关键——样本集必须覆盖所有输入值的全部取值范围。神经网络的效果好坏依赖于训练神经网络的数据的好坏。
- 竞争学习神经网络适合于解决聚类和分类问题。在竞争神经网络中，每一个竞争神经元和一个单独的聚类相关联。然而，聚类是一个非监督的过程，所以不能直接用竞争神经元来标识每一个聚类。在大多数实际应用中，由领域专家来负责解释每一个聚类所表示的含义。
- 遗传算法适合解决复杂的优化问题。遗传算法产生一个种群的可行的解决方案，并让这些方案在一个自然选择的过程中进化。为了确保能到找到问题的最优解决方案，遗传算法会以不同的交叉率和突变率在不同大小的染色体种群中运行。
- 解决真实世界中的复杂问题，往往需要一个能够将特定领域的人类专家和学习能力集成的复杂智能系统的应用。虽然混合智能系统还在不断发展，神经－模糊系统已经在决策支持领域找到了很多类型的应用。

复习题

1. 什么是知识工程？描述知识工程的主要步骤。为什么选择合适的工具是构建智能系统的关键步骤？
2. 知识获取过程的步骤有哪些？为什么知识获取常常被称为知识工程的瓶颈？获取的数据如何影响系统构建工具的选择？
3. 什么是原型？什么是测试用例？如何测试智能系统？如果错误选择了系统构建工具，该怎么办？
4. 为什么新兴的智能技术变为以问题为驱动，而不是过去的以好奇心为驱动？
5. 为什么对于诊断和故障排除问题而言，专家系统是很好的选择？什么是电话规则？
6. 开发专家系统时如何选择工具？专家系统框架的优点是什么？构建专家系统时如何选择专家框架？
7. 开发一个基于规则的用于空调诊断的专家系统。空调是一个复杂的设备，有许多区域和组成部件非专业人士无法直接操作。除此之外，在向专业服务人员寻求服务之前，有一些事情可以首先检测。该专家系统应该能够诊断一些诸如你是否有能力自己修复故障的简单问题。因此可以避免打不必要的电话给服务公司。空调系统可能出现的问题有：设备冻结、漏水、不制冷和不运行。详细信息可以参考空调的故障排除手册。你也可以在以下网站中获取一些有用的信息：http://www. accheckup. com/trouble. htm。 359
8. 以下显示了一系列使用贝叶斯论据积累来评估病人的胸痛不适的规则。评价这个系统并且提供一个推理过程的完整追踪。如果病人存在胸痛症状并且其心电图是阳性的，确定病人患有心脏疾病的概率。

```
control bayes

Rule: 1
if    chest_pain is true {LS 0.2 LN 0.01}
then heart_disease is true {prior 0.1}

Rule: 2
if    chest_pain is true
and electrocardiogram is positive {LS 95 LN 5}
then heart_disease is true

/*The SEEK directive sets up the goal of the rule set
seek heart_disease
```

9. 以下显示了使用确信因子来评估胸痛不适的一系列规则。如果病人存在胸痛症状并且其心电图是阳性的，确定病人患有心脏疾病的确定性。提供一个推理过程的完整追踪。

```
control cf

Rule: 1
if    chest_pain is true
then heart_disease is true {cf 0.1}

Rule: 2
if    electrocardiogram is positive
then heart_disease is true  {cf 0.7}

Rule: 3
if    electrocardiogram is negative
then heart_disease is true  {cf 0.05}

/*The SEEK directive sets up the goal of the rule set
seek heart_disease
```

10. 为什么模糊系统尤其适合对人类决策进行建模？为什么模糊技术在商务和金融等领域具有广阔的前景？

11. 开发一个用于直接营销的模糊系统。假设目标群体定义为“处于主要工作年龄并且收入高于平均水平的父亲”。“主要工作年龄”定义为40～55岁之间，平均收入为45000美元。确保你的模糊系统具有将处于年近四十，至少有两个小孩的高收入男性作为目标群体的
360 能力。
12. 开发一个用于检测家庭保险欺诈索赔的模糊专家系统。假设有以下7个输入变量：申请人在过去12个月内提交的索赔数、目前的索赔数额、申请人投保当前保险公司的时间长度、过去12个月内所有账户的平均余额、过去12个月内超出平均余额的索赔数、申请人的年收入、过去4个月内申请人的生活是否发生根本性改变（例如，申请人结婚或者离婚、失去工作或者成为一位父亲）。假设模糊系统的输出为欺诈的可能性（0～1之间的一个数，0表示保险理赔无欺诈，1表示保险理赔为欺诈的可能性非常高）。

使用分层模糊建模方法。首先，评估历史保险（使用前3个输入变量），银行历史（使用第4和第5个输入变量）和索赔状态（使用最后2个输入变量）。然后，使用历史保险，银行历史和索赔状态作为输入，评估欺诈的可能性。

13. 神经网络流行的根本原因是什么？神经网络在哪些领域应用最为成功？解释其原因并举例说明。
14. 在神经网络模型中，使用数据之前为什么要对其进行归一化？如何对数据进行归一化？分别举出对连续型数据和离散型数据进行归一化的例子。什么是1/N编码？
15. 开发一个用于将锯缘青蟹分类为雄性和雌性的反向传播神经网络。使用以下5个测量值（单位为毫米）：额叶的大小，后排宽度，头胸甲长度，头胸甲宽度和体深度。数据集有100个样本，包括50个雄性样本和50个雌性样本。数据集可以在本书的网站中找到：http://www.booksites.net/negnevitsky。
16. 根据从乳房肿块的细针穿刺（FNA）数字图像中得到的一些特征，开发一个用于乳腺癌诊断的反向传播神经网络。数据集包含150个病例。每个病例被诊断为恶性或良性，基于以下10个实值特征：半径、纹理、周长、面积、平整度、压实度、凹度、凹点、对称性和分形维数。数据集位于本书网站：http://www.booksites.net/negnevitsky。解释你所开发的诊断神经网络的困难和局限。
17. 开发一个具有异质结构的神经－模糊系统，用于解决习题16中描述的问题。注意，你首先需要对问题进行重新定义，以使系统的输出为乳腺癌的估计风险。
18. 使用一个自组织神经网络来对污水处理厂数据做聚类分析。数据集包含527个样本。每个样本有38个属性。数据集位于UCI机器学习库网站：
http://archive.ics.uci.edu/ml/datasets/Water+Treatment+Plant
19. 开发一个遗传算法用于城市中急救中心的部位，使得城市中的医疗急救的响应时间最小化。城市被映射到一个7 km×7 km的网格，如图9.47所示。网格中每一个小格上的数字代表该小格区域内的年平均急救次数。

适应函数可以定义为急救率加权距离之和的倒数：

$$f(x,y) = \sum_{n=1}^{49} \lambda_n \sqrt{(x_n - x_{\text{eru}})^2 + (y_n - y_{\text{eru}})^2}$$

361 其中，λ_n 表示小格 n 内的急救率，(x_n, y_n) 表示小格 n 中心点的坐标，$(x_{\text{eru}}, y_{\text{eru}})$ 表示急救中心的坐标。假设急救中心只能位于小格的中心点处。

20. 在习题19描述的问题中，假设在 $x=5$ km 处有一条小河将城市分成了东西两部分。东西两部分在 $x=5$ km，$y=5.5$ km 处有一座桥相连接。如图9.48所示。开发一个遗传算法解决这个问题，找出急救中心的最佳位置并与习题19中得到的位置进行对比。

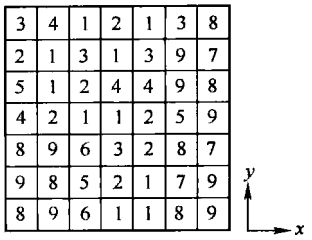


图 9.47 一个 7 km × 7 km 的城市网格

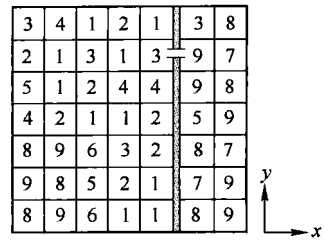


图 9.48 被一条小河划分的 7 km × 7 km 的城市网格

参考文献

Abrams, B.A. and Huang, C.J. (1987). Predicting bank failures: the role of structure in affecting recent failure experiences in the USA, *Applied Economics*, 19, 1291–1302.

Alam, P., Booth, D., Lee, K. and Thordarson, T. (2000). The use of fuzzy clustering algorithm and self-organizing neural networks for identifying potentially failing banks: an experimental study, *Expert Systems with Applications*, 18(3), 185–199.

Applegate, D., Bixby, R., Chvátal, V. and Cook, W. (2001). TSP cuts which do not conform to the template paradigm, *Computational Combinatorial Optimization*, M. Junger and D. Naddef, eds, Springer-Verlag, Berlin, pp. 261–304.

Berry, M. and Linoff, G. (2004). *Data Mining Techniques for Marketing, Sales, and Customer Relationship Management*, 2nd edn. John Wiley, New York.

Booth, D.E., Alam, P., Ahkam, S.N. and Osyk, B. (1989). A robust multivariate procedure for the identification of problem savings and loan institutions, *Decision Sciences*, 20, 320–333.

Dantzig, G., Fulkerson, R. and Johnson, S. (1954). Solution of a large-scale traveling salesman problem, *Operations Research*, 2, 393–410. 362

Davis, L. (1985). Applying adaptive algorithms to epistatic domains, *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, A. Joshi, ed., Morgan Kaufmann, Los Angeles, pp. 162–164.

Davis, R. and King, J. (1977). An overview of production systems, *Machine Intelligence*, 8, 300–322.

Durkin, J. (1994). *Expert Systems: Design and Development*. Prentice Hall, Englewood Cliffs, NJ.

Eichengreen, B. (2002). *Financial Crises and What to Do About Them*, Oxford University Press, Oxford.

Espahbodi, P. (1991). Identification of problem banks and binary choice models, *Journal of Banking and Finance*, 15(1), 53–71.

Firebaugh, M. (1988). *Artificial Intelligence: A Knowledge-Based Approach*. Boyd & Fraser, Boston, MA.

Fisher, R.A. (1950). *Contributions to Mathematical Statistics*. John Wiley, New York.

Flood, M.M. (1955). The traveling salesman problem, *Operations Research*, 4, 61–75.

Goldberg, D.E. (1989). *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Goldberg, D.E. and Lingle, R. (1985). Alleles, loci and the traveling salesman problem, *Proceedings of the 1st International Conference on Genetic Algorithms*, J.J. Grefenstette, ed., Lawrence Erlbaum Associates, Pittsburgh, PA, pp. 154–159.

Gülllich, H.-P. (1996). Fuzzy logic decision support system for credit risk evaluation, *EUFIT Fourth European Congress on Intelligent Techniques and Soft Computing*, Aachen, pp. 2219–2223.

Haupt, R.L. and Haupt, S.E. (2004). *Practical Genetic Algorithms*, 2nd edn. John Wiley, New York.

Haykin, S. (2008). *Neural Networks and Learning Machines*, 3rd edn. Prentice Hall, Englewood Cliffs, NJ.

Hopfield, J.J. and Tank, D.W. (1985). Neural computation of decisions in optimization problems, *Biological Cybernetics*, 52, 141–152.

Kurgan, L.A., Cios, K.J., Tadeusiewicz, R., Ogiela, M. and Goodenday, L. (2001). Knowledge discovery approach to automated cardiac SPECT diagnosis, *Artificial Intelligence in Medicine*, 23(2), 149–169.

Laarhoven, P.J.M. and Aarts, E.H.L. (2009). *Simulated Annealing: Theory and Applications*. Springer Netherlands, Dordrecht.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. and Shmoys, D.B. (1985). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley, Chichester.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D. (1990). Handwritten digit recognition with a back-propagation network, *Advances in Neural Information Processing Systems*, D.S. Touretzky, ed., Morgan Kaufmann, San Mateo, CA, vol. 2, pp. 396–404.

Martin, O., Otto, S.W. and Felten, E.W. (1991). Large-step Markov chains for the traveling salesman problem, *Complex Systems*, 5(3), 299–326.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolutionary Programs*, 3rd edn. Springer-Verlag, New York.

Michie, D. (1982). The state of the art in machine learning, *Introductory Readings in Expert Systems*, Gordon and Breach, New York, pp. 209–229.

Ozkan-Gunay, E.N. and Ozkan, M. (2007). Prediction of bank failures in emerging
363 financial markets: an ANN approach, *Journal of Risk Finance*, 8(5), 465–480.

Potvin, J.V. (1996). Genetic algorithms for the traveling salesman problem, *Annals of Operations Research*, 63, 339–370.

Principe, J.C., Euliano, N.R. and Lefebvre, W.C. (2000). *Neural and Adaptive Systems: Fundamentals Through Simulations*. John Wiley, New York.

Richards, R. (2002). Application of multiple artificial intelligence techniques for an aircraft carrier landing decision support tool, *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'02*, Honolulu, Hawaii.

Russell, S.J. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*, 3rd edn. Prentice Hall, Upper Saddle River, NJ.

Simon, R. (1987). The morning after, *Forbes*, October 19, pp. 164–168.

Tryon, R.C. (1939). *Cluster Analysis*, McGraw-Hill, New York.

Tschoke, S., Lubling, R. and Monien, B. (1995). Solving the traveling salesman problem with a distributed branch-and-bound algorithm on a 1024 processor network, *Proceedings of the 9th IEEE International Parallel Processing Symposium*, Santa Barbara, CA, pp. 182–189.

Von Altrock, C. (1997). *Fuzzy Logic and NeuroFuzzy Applications in Business and Finance*. Prentice Hall, Upper Saddle River, NJ.

Waterman, D.A. (1986). *A Guide to Expert Systems*. Addison-Wesley, Reading, MA.

Widrow, B. and Stearns, S.D. (1985). *Adaptive Signal Processing*. Prentice Hall, Englewood Cliffs, NJ.

Zhang, G., Hu, M.Y., Patuwo, B.E. and Indro, D.C. (1999). Artificial neural networks in bankruptcy prediction: general framework and cross-validation analysis, *European Journal of Operation Research*, 116, 16–32.

364 Zurada, J.M. (2006). *Introduction to Artificial Neural Systems*, 2nd edn. Jaico, Mumbai.

数据挖掘和知识发现

本章主要介绍大型数据库中进行知识发现的一个组成部分：数据挖掘。我们还将讨论将数据转化为知识的主要技术和工具。

10.1 数据挖掘简介

如果你正感觉到信息泛滥，那么你就对了！你和你身边的每个人都处于泛滥的信息海洋中。全世界存储在硬盘上的数据量每年都在翻倍。现如今，数据的度量单位为太字节，拍字节（1000太字节，或者10^{15}字节），甚至是艾字节（1 000 000太字节，或者10^{18}字节）。一太字节相当于两百万本书。整个美国国会图书馆的一千九百万本书只有大约10太字节的信息量。根据IDC白皮书“与经济合同一样，数字宇宙在膨胀”的说法，2008年数字宇宙（以数字形式产生，获取或者复制的信息）大小为487艾字节（Gantz and Reinsel, 2009）。到2012年，数字信息将达到2500艾字节。如果将数字宇宙转化为书籍，所有书籍的厚度将可以来回冥王星50次。而且，这些产生的书籍每年都在翻倍。目前的增长速度相当于火箭速度的20倍。宇宙神5号火箭花费13个月才到达冥王星，但是这些新产生的信息书籍只需要三周的时间。

我们确实生活在一个信息迅速膨胀的宇宙，然而从海量的数据中提取我们所需要的信息仍然存在很多困难。美国宇航局拥有远超过他们分析能力的数据量。人类基因组项目的研究人员需要为30亿个DNA库中的每一个DNA存储和处理上千字节数据来组成人类的基因。每天数艾字节的数据在互联网中流通，我们需要能够帮助我们从中提取有意义的信息和知识的方法。

现在社会也是基于信息的。到目前为止，大部分存储在计算机中的信息都是以事实、观测和测量等原始形式存在。这些事实、观测和测量构成了数据。数据是我们所收集和存储的。随着计算能力代价的持续下降，新增的数据量正在指数级增长。我们的问题是，面对所有这些有价值的 365
资源，我们能做些什么。我们最终需要的是知识。知识能帮助我们做出明智的决策。

传统的数据库能够帮助我们方便地存储和访问数据，但是它并不能很好地对数据进行有意义的分析。对数据进行有意义的分析，这是*数据挖掘*（data mining），又称为*数据库知识发现*（Knowledge Discovery in Database, KDD）。

“数据挖掘”这个词通常和从数据中提取知识有关。数据挖掘也可以定义为对大规模数据的探究和分析以发现有意义的模式和规则（Berry and Linoff, 2004）。数据挖掘的终极目标是发现知识。

数据挖掘可以看成是计算机数据库的自然演化。传统的数据库管理工具帮助用户从数据库中提取特定数据，数据挖掘则协助用户从存储在数据库中的数据中发现隐含的模式（相互关系、趋势、异常和聚类）以及对新数据进行预测。传统的工具往往期望用户拥有一些关于数据库中存在的关系的假设。这些假设会通过一系列数据查询来证明或者反驳。让我们来看一个例子，假设我们已经获取了一些研究高血压的数据。这种数据中通常包含每个人的年龄、性别、体重和身高、体育活动、是否吸烟和饮酒习惯等。通过查询工具，用户可以选择一个可以影响最终输出变量（在我们的例子中是高血压）的特定变量，例如是否吸烟。这里用户的目的是比较吸烟和不吸烟的高血压患者的数量。然而，在选择是否吸烟这个变量的同时，用户已经做出了一个假设：高血压和是否吸烟有紧密的相互关系。

如果选择一个数据挖掘工具，用户不再需要假设数据集中不同变量的相关性（也不需要一次只研究一个关系），就可以决定影响输出变量的最重要的因子。所以，我们现在可以自动识别最显著的风险因子，而不是假设高血压和是否吸烟有相关性。同时，我们也可以比较不同的组（或者称为簇）的高血压人群。数据挖掘不需要任何假设，它能自动发现隐含的关系和模式。

人们经常把数据挖掘和黄金开采进行对比。在提取出黄金之前，需要处理大量的矿石。数据挖掘可以帮助我们从原始的数据中找到“隐含的黄金”——知识。

数据挖掘往往以原始的数据为开始，以提取的知识为结束。将原始的知识转化为知识的整个过程如图 10.1 所示。这个过程包括了 5 个数据转换的步骤。

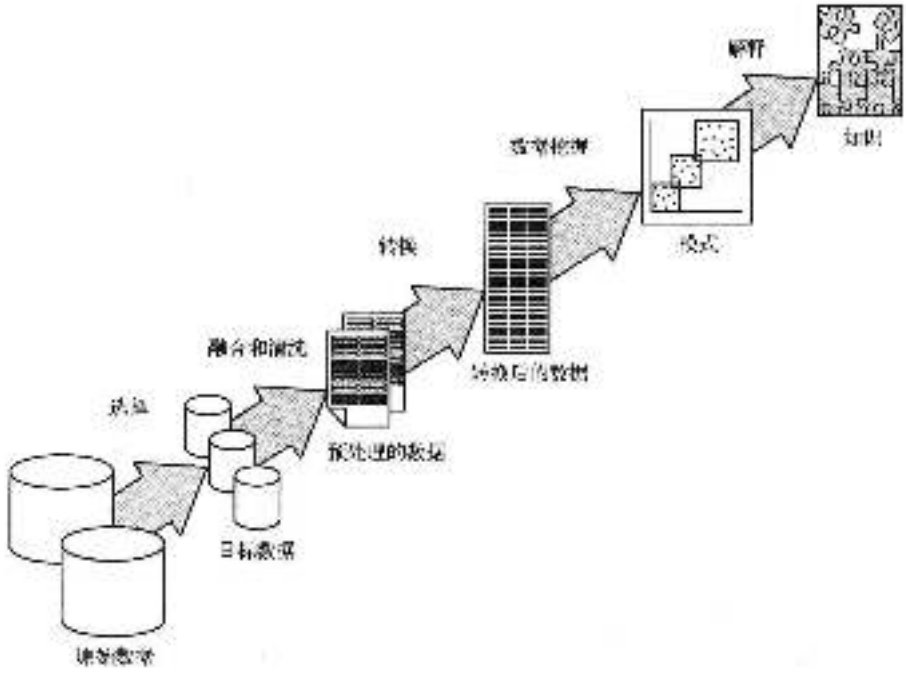


图 10.1 数据挖掘和知识发现的过程

第一个步骤是数据选择。数据选择发生在与分析相关的数据确定以及多个数据源识别之后。数据选择可能是数据挖掘过程中最花费时间的一个步骤。因为我们需要的数据会以文本文件、关系表和电子表格等多种格式分布在许多的数据库中。

第二个步骤是包括数据融合和数据清理（也称为数据清洗）。在这个步骤中，所有相关的数据都从目标数据库中检索出来并集成到一个公共源中。冗余和不一致的数据将会从数据集中删除。

第三个步骤是数据转换，也称为数据整合。在这个步骤中，清洗后的数据将会被转换为适合数据挖掘的形式。

上述数据挖掘过程中的 3 个步骤经常被组合到一起，统称为预处理。预处理的目的是简单地将原始数据转换为适合后续的数据挖掘的格式。

第四个步骤是数据挖掘本身。在这个重要的阶段中，一系列不同的数据挖掘技术被应用，以提取对用户可能有用的信息。

第五个步骤是结果解释。在这个步骤中，发现的知识将会表示为用户可以容易理解和使用的形式。这个最后的步骤经常与将数据挖掘和决策支持系统集成相关联。

如何在实践中应用数据挖掘

虽然数据挖掘仍然是一个很新的、不断发展的领域，但它已经被广泛地应用于银行、金融、市场营销和电信等行业。如今有很多公司都在使用数据挖掘，只不过拒绝谈论它而已。 366 ~ 367

从数据挖掘获得了战略性益处的几个领域是直销、趋势分析和欺诈检测（Abramowicz and Zurada, 2001; Witten and Frank, 2005）。在直销领域，数据挖掘被用来确定最有可能购买某些产品和服务的人。在趋势分析中，数据挖掘被用来确定市场趋势，例如通过对证券市场建模。在欺诈检测中，数据挖掘被用来识别保险理赔、移动电话和信用卡交易中可能的欺诈行为。

在大多数应用中，数据挖掘被用来发现和描述数据中的结构化的模式以做出正确的预测。数据挖掘的一个经典应用是小额商业贷款中的信用分析。银行保存当前和历史的贷款记录。除了保存是否按时偿还等贷款的详细信息之外，银行还会保存一些附加信息，包括当前利率、商业未偿还债务和贷款者的信用历史，等等。为了减少不良贷款的风险，银行管理者可能希望挖掘银行的记录来发现一些有意义的模式。换句话说，银行管理者希望学习到一些可用于新的贷款申请考核的决策标准。此处的目标是建立一些能够区分良好贷款和不良贷款的模式。数据挖掘可以自动发现这些模式，因此可以用来对新的贷款申请作出更准确的风险评估。

什么是数据挖掘技术

数据挖掘是一个实践话题，因此自然会运用可以帮助我们从可用数据中提取更多知识的技术。实际上，数据挖掘经常被视为“什么都包括”的东西——它使用从统计方法和查询工具到复杂的机器学习等技术。大多数在数据挖掘中使用的智能技术已经在本书中讨论过。数据挖掘不是一个单一的方法，而是不同工具和技术的异构组合。这些工具和技术包括：

- 统计方法和数据可视化工具。
- 查询工具。
- 联机分析处理（OLAP）。
- 决策树和关联规则。
- 神经网络和神经 - 模糊系统。

数据挖掘的成功依赖于数据挖掘工具的选择。对数据的初步调查可以帮助我们理解数据的特定特征，从而选择合适的数据分析技术。这个初步的过程称为*数据探索*（data exploration）。数据探索包括汇总统计、可视化、主成分分析、查询工具和 OLAP（Witten and Frank, 2005; Tan et al., 2006）。 368

10.2　统计方法和数据可视化

可视化是一种在数据集中进行模式发现的非常有用的数据探索技术。*数据可视化*（data visualisation）可以定义为一种以图形或表格形式表示信息的方法（Tan et al., 2006）。信息被可视化以后，我们就可以解释和理解它了。在日常生活中，我们经常使用图形、直方图和图表来表示选举结果或证券市场的变化。人们可以很快很容易地吸收可视化的信息，因为经过几百万年的进化，人脑已经拥有很强的模式处理能力。可视化在数据探索的初始阶段是极为重要的。

可视化是一种数据驱动的数据挖掘技术。数据可视化的目的是揭露隐含在数据中的模式、关系和趋势。在数据中发现的模式，最后产生一个假设。

什么软件包可用作数据的信息可视化

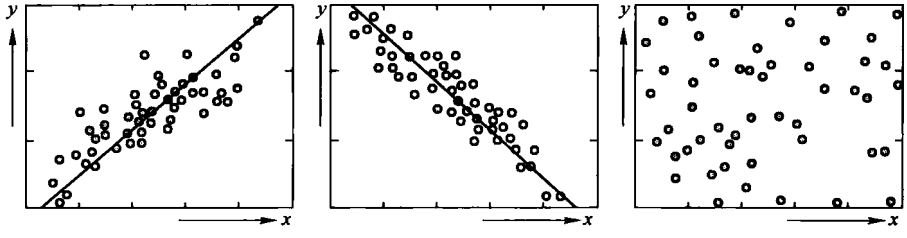
有一些在个人计算机和大型机中都可用的可视化软件包，其中最流行的有统计分析系统（SAS）、Statgraphics、Minitab 和 MATLAB 统计工具包。这些软件包拥有广泛的基本应用和高级的统计分析及可视化功能。

许多图形化的数据探索技术依赖于*汇总统计*。汇总统计可以定义为表示数据不同特征的一

组值，例如均值、中位数、众数、范围和标准差(Montgomery et al.，2001)。需要注意的是，虽然汇总统计提供了关于数据的一些有用的信息，他们的局限性也需要很好地理解。一个经典的例子是平均收入。一个人群的平均收入往往会被误认为表示了这个人群中大部分人的收入。实际上，平均收入要远远高于大部分人的收入，因为存在一些高收入人员，如比尔·盖茨。为了发现这些离群值，我们需要借助于数据可视化技术。

图形化的数据表示方法包括点图、茎叶图、直方图和箱形图（Hand et al.，2001）。然而在统计分析中最常用的可视化技术是散布图。散布图是一个用于表示两个变量之间相关度的二维图形。数据用点的集合来表示。每一个点的横轴位置由一个变量决定，纵轴位置由另外一个变量决定（Utts，2004）。举个例子，为了显示人的身高（第一个变量）和体重（第二个变量）之间的关系，我们首先选择一组人员（样本集），测量其身高和体重，然后将每个人的数据绘制在一个
369 以“身高”为 x 轴和以“体重”为 y 轴的二维坐标系统中。

散布图可以表示不同类型的相关性。如果点是左低右高散布，则称为有一个正相关性；如果点是左高右低散步，则称为有一个负相关性。我们可以绘制一条最佳拟合的直线来表示这种相关性。散布图也可以展示待研究的两个变量之间没有相关性（或称为空相关性）。图 10.2 展示了散布图的例子。



a) 正相关性　　b) 负相关性　　c) 空相关性

图 10.2　散布图

当处理大量的数据点时，散布图尤其有用。在数据的最初观察阶段，散布图常常可以帮助我们揭示变量之间的复杂关系以及发现模式和异常点。

现在让我们更深入地审查一下图 10.2a 和图 10.2b。两个图表均显示大部分的数据点随机散步在一条直线的两旁。这样的直线可以表示成以下的直线方程的形式：

$$y = \beta_0 + \beta_1 x \tag{10.1}$$

其中，x 为自变量（也称为输入变量或回归元），它是被操作的变量；y 为因变量（又称为输出变量或回归子）——它是由自变量操作的观测结果；β_0 和 β_1 经过一个称为回归的过程获得，称为回归因子。

“回归”这个词是在 19 世纪由维多利亚时代的人类学家和优生学家 Francis Galton 爵士引入。他是第一个用统计方法来进行人类种群差异性研究的学者。Galton 的研究表明，高大的祖先的后代的高度在朝着整个种群的身高平均值方向减小（Galton，1877）。随后他的工作被扩展到更一般的统计上下文中。

在数据挖掘中，回归可以被定义为一种寻找能够最佳拟合数据的数学方程的统计技术。回归的最简单形式是线性回归——用一条直线来拟合数据。因此，公式（10.1）决定了由给定 x 来预测 y 的合适的 β_0 和 β_1。高级的回归模型，例如多元回归，可以使用多个输入变量并且允许
370 非线性方程的拟合，包括指数函数、log 函数、二元函数和高斯函数（Seber and Wild，1989）。

公式（10.1）显示了因变量 y 是自变量 x 的线性函数。然而，从图 10.2a 和 10.2b 可以看

出，y 的值并不是精确地落在一条直线上。公式（10.1）仅仅是我们对线性函数能够拟合数据的一个假设。实际上，因变量 y 的每一个实际值由公式（10.1）加上一个随机误差 ε 组成。因此，对于 n 对观测值 (x_1, y_1)，(x_2, y_2)，…，(x_n, y_n)，有：

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \quad i = 1, 2, \ldots, n \tag{10.2}$$

其中，β_0 和 β_1 为未知的回归因子，ε_i 是一个均值为零，方差为 σ^2 的随机误差。

如何确定回归因子

对于线性回归模型，正确的回归因子应该表示了能够“最佳”拟合数据的一条直线。“最佳”拟合的评价标准基于最小二乘法。直线的截距 $\hat{\beta}_0$ 和斜率 $\hat{\beta}_1$ 的最小二乘估计由以下两式确定：

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} y_i x_i - \left[\left(\sum_{i=1}^{n} y_i\right)\left(\sum_{i=1}^{n} x_i\right)\Big/ n\right]}{\sum_{i=1}^{n} x_i^2 - \left[\left(\sum_{i=1}^{n} x_i\right)^2 \Big/ n\right]} \tag{10.3}$$

和

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \tag{10.4}$$

其中，

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}, \quad \bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

因此，拟合的回归直线由下式定义：

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x \tag{10.5}$$

现在，我们可以将公式（10.2）写成以下形式：

$$y_i = \hat{\beta}_0 + \hat{\beta}_1 x_i + e_i, \quad i = 1, 2, \ldots, n \tag{10.6}$$

其中，$e_i = y_i - \hat{y}_i$ 称为*残差*（residual）。残差表示拟合模型和实际数据之间的误差，因此为模型的准确性提供了很好的指示。 371

离群值如何影响线性回归模型

离群值，特别是那些离大部分数据都很远的值，可以很显著地改变回归直线的位置（最小二乘估计被扯拽向离群值）。作为说明，我们考虑表 10.1 中给出的数据（Daniel and Wood, 1999）。表中显示的是对化学工厂的测量值，样本集中包含 20 个样本。自变量 x 表示用滴定法测量的酸值，因变量 y 表示用提取和称重法测量的有机酸含量。此处我们的目标是确定用相对廉价的滴定法获得的测量值多大程度上能够代替用相对昂贵的提取和称重技术获得的测量值。

$$\hat{y} = 35.4583 + 0.3216x$$

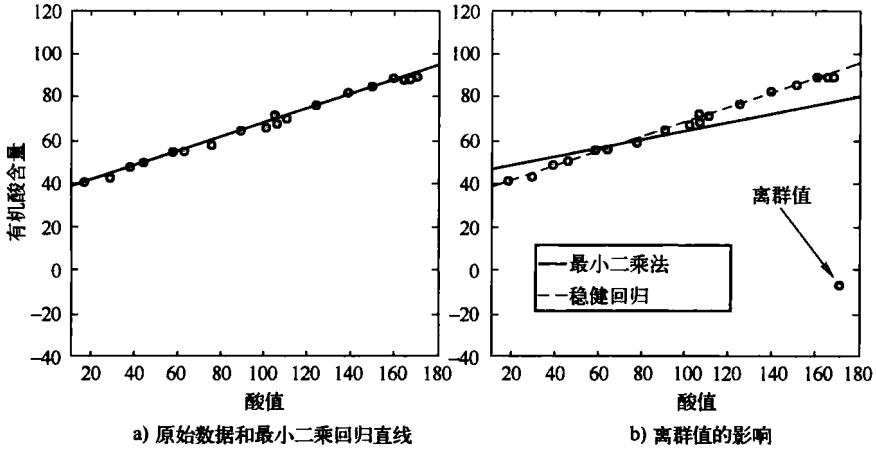
表 10.1　化学工厂的数据（Daniel 和 Wood, 1999）

观测值	自变量：酸值	因变量：有机酸含量
1	123	76
2	109	70
3	62	55
4	104	71
5	57	55
6	37	48
7	44	50
8	100	66

（续）

观测值	自变量：配值	因变量：有机酸含量
9	16	41
10	28	43
11	138	82
12	105	68
13	159	88
14	75	58
15	88	64
16	164	88
17	169	89
18	167	88
19	149	84
20	167	88

图 10.3a 中的散布图显示滴定法测量的酸值和提取称重法测量的有机酸含量之间有很强的统计相关性。因此我们有理由假设一个线性模型，计算最小二乘估计值 $\hat{\beta}_0$ 和 $\hat{\beta}_1$，然后将得到的值代入公式（10.5）中，得到以下的回归直线：



a) 原始数据和最小二乘回归直线　　b) 离群值的影响

图 10.3　离群值对回归直线的影响

利用该回归模型，我们可以通过酸值来预测有机酸含量。例如，假设酸值 $x = 150$，则有机酸含量的估计值为：

$$\hat{y} = 35.4583 + 0.3216.150 = 83.6983$$

虽然这种估计容易受误差影响，但是最小二乘法确实给我们提供了一个合理的预测模型。

现在假设有一个观察值被错误地记录，例如，第 17 行的 y 值被写成了 -8 而不是 89。由图 10.3b 可以看出，这个错误显著地影响了回归直线的位置，现在最小二乘回归直线由下式定义：

$$\hat{y} = 44.0528 + 0.1911x$$

如何解决离群值问题

离群值往往由噪声数据造成，因此统计学家通常希望能够检查数据并删除检测到的离群值。在简单的线性回归模型中，离群值的可视化检测是有可能的。

虽然离群值显著影响着回归模型，我们仍然可以通过一个*鲁棒回归*来减小他们的影响。鲁棒回归使用最小化平方均值来代替最小化二乘法中的最小化平方和（Rousseeuw and Leroy, 2003）。最小平方均值法得到的回归直线如图 10.3b 所示。拟合回归模型由下式表示：

372 ~ 373

$$\hat{y} = 35.3950 + 0.3226x$$

可见此处得到的回归因子的值与从原始数据集中得到的非常接近。鲁棒回归已经被证明能够明显地减小离群值的影响。

然而，离群值的最严重的问题是我们无法确定一个离群值到底是一个错误，还是仅仅是一个不常见的正确值。如果简单地将离群值从数据集中删除，我们可能无意地犯下“把小孩与洗澡水一起倒掉”的错误。在统计回归中，虽然用一条直线来拟合钟形曲线型的数据很常见，但是大多数问题并不能很容易地可视化。一个常用的解决方案是应用不同的学习技术来过滤数据，过滤出来的异常值再由人类专家来校验。

最后，需要注意的是，虽然两个甚至三个变量之间的关系能够进行可视化，高维数据的图形化挖掘仍然是一个挑战。一种处理多维数据的方法是降维。

如何在不丢失重要信息的情况下降低数据维度

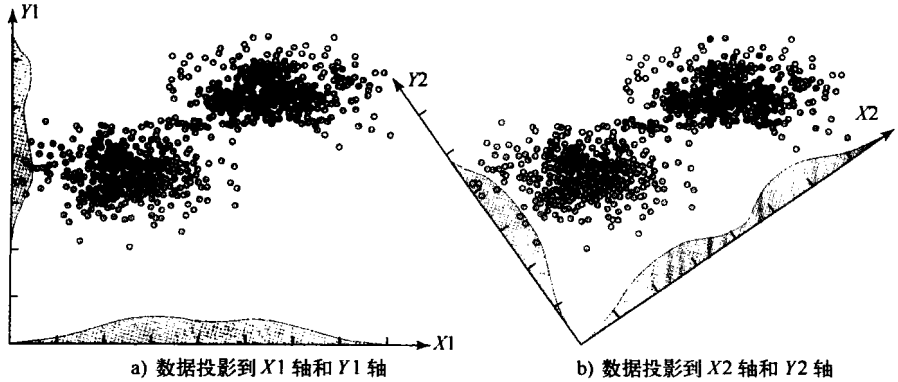
在多维数据中，一个变量与另一个变量高度相关的情况是很常见的。换句话说，由这些变量所提供的信息是明显有冗余的。在两个变量 x 和 y 完全相关的极端情况下，其中的一个变量是冗余的。因为如果已知 x 的值，y 的值能够很容易地确定。将大量相关变量转换为少量不相关变量的数学过程称为*主成分分析*（Principal Component Analysis, PCA）。PCA 能在不明显丢失信息的情况下降低数据维度。PCA 应用于人脸识别和图像压缩领域，它也是数据挖掘领域从大规模多维数据集中进行模式发现时所使用的一种常用技术。

10.3　主成分分析

PCA 是统计学中用做高维数据的信息内容最大化的一种常用方法（Jolliffe, 2002）。它最早是由 Karl Pearson 于一个多世纪以前介绍的（Pearson, 1901）。在数学中，PCA 从数据空间寻找表示最大方差的 m 个正交向量，然后将数据从原来的 n 维空间投影到一个 m 维子空间，其中 $m \ll n$，从而降低数据维度。换句话说，大量的相关变量转换为了少量称为*主成分*的不相关变量。在数据挖掘中，PCA 常常被用作一种数据预处理技术。

作为一个说明，我们考虑图 10.4 所示的二维（二元）数据集。图 10.4a 显示了横轴为 $X1$，纵轴为 $Y1$ 时的数据集。将数据投影到每一个坐标轴，我们得到二维密度图。图 10.4b 显示横轴为 $X2$，纵轴为 $Y2$ 下的同一数据集，$X2$ 和 $Y2$ 通过应用 PCA 得到。将数据投影到 $X2$ 轴之后，我

374



a) 数据投影到 $X1$ 轴和 $Y1$ 轴　　b) 数据投影到 $X2$ 轴和 $Y2$ 轴

图 10.4　PCA 的一个应用

们发现数据集是双峰的。相反，将数据投影到 *Y2* 轴之后，数据集的双峰本质完全被掩盖。群集在高方差方向（*X2* 轴）比低方差方向（*Y2* 轴）更好区分。将数据投影到高方差方向后，能够保留大部分的信息，因而可以降低数据维度。

在上述例子中，数据集的群集结构从一开始就是明显的。然而，在高维数据集中，数据的固有本质往往是隐藏的，需要应用 PCA 才能揭示这些固有本质。

PCA 是如何工作的

首先，我们计算数据在原始坐标下的*协方差矩阵*。其次，确定协方差矩阵的特征值和特征向量。特征向量代表了转换空间的坐标轴，且按照相应特征值的大小排序。每一个特征值显示了相应轴的方差——方差越大，相应轴上保留的信息越多。最后，我们将原始数据投影到由最大特征值对应的特征向量所形成的转换空间中。

现在我们演示 PCA 在一个简单例子上的工作流程。我们将使用一个二维的数据集并显示 PCA 在每一个步骤中的工作内容。

步骤 1：获得数据并减去平均值。

在这个例子中，我们将使用表 10.1 所示的数据集。在 PCA 中，我们需要将每一个数据维上
375 的值减去该维的平均值。我们的例子中，$\bar{x} = 103.05$、$\bar{y} = 68.60$。表 10.2 显示了减去平均值后规范化的数据集。

表 10.2　原始数据和减去平均值后的数据

x	y	$x-\bar{x}$	$y-\bar{y}$
123	76	19.95	7.40
109	70	5.95	1.40
62	55	-41.05	-13.60
104	71	0.95	2.40
57	55	-46.05	-13.60
37	48	-66.05	-20.60
44	50	-59.05	-18.60
100	66	-3.05	-2.60
16	41	-87.05	-27.60
28	43	-75.05	-25.60
138	82	34.95	13.40
105	68	1.95	-0.60
159	88	55.95	19.40
75	58	-28.05	-10.60
88	64	-15.05	-4.60
164	88	60.95	19.40
169	89	65.95	20.40
167	88	63.95	19.40
149	84	45.95	15.40
167	88	63.95	19.40

步骤 2：计算协方差矩阵。

*协方差*用于度量两个变量之间的线性关系。与方差只应用于一个单独的变量不同，协方差需要在两个变量间测量。例如，对于一个三维数据集，我们可以测量 x 和 y、x 和 z、y 和 z 之间的协方差。

方差和协方差是密切相关的。方差的计算公式可以表示为：

$$\text{var}(x) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})}{n} \tag{10.7}$$

将第二个括号的内容替换成 $(y_i - \hat{y})$ 后，我们得到了协方差的计算公式：

$$\text{cov}(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n} \tag{10.8}$$

376

注意，任何变量与它本身的协方差等于这个变量的方差：

$$\text{cov}(x,x) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})}{n} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n} = \text{var}(x) \tag{10.9}$$

一个正的协方差表示两个变量正相关；相反，一个负的协方差表示两个变量负相关。协方差为零，意味着两个变量相互独立，或者不相关。协方差的绝对值能够度量数据冗余度。

通常将给定多维数据中任意两个维度的协方差表示成一个方阵，称为*协方差矩阵*。协方差矩阵相对于主对角线对称。主对角线上的元素表示各变量的方差，其余的元素表示变量之间的协方差。我们的例子中数据是二维的，所以协方差矩阵是一个 2×2 矩阵：

$$C = \begin{bmatrix} \text{cov}(x,x) & \text{cov}(x,y) \\ \text{cov}(y,x) & \text{cov}(y,y) \end{bmatrix} = \begin{bmatrix} \text{var}(x) & \text{cov}(x,y) \\ \text{cov}(y,x) & \text{var}(y) \end{bmatrix}$$

$$= \begin{bmatrix} 2580.7 & 830.0 \\ 830.0 & 268.4 \end{bmatrix}$$

非对角元素都是正数，表明变量 x 和 y 有相同的变化趋势。

步骤 3：计算协方差矩阵的特征值和特征向量。

*特征值和特征向量*是方阵的特征。假设 $\boldsymbol{A}$ 为一个正定对称方阵，它可以用其正交特征向量进行对角化为：

$$\boldsymbol{A} = \boldsymbol{E}\boldsymbol{\Lambda}\boldsymbol{E}^T \tag{10.10}$$

其中，$\boldsymbol{E}$ 是一个 $n \times n$ 正交矩阵，T 表示转置矩阵，$\boldsymbol{\Lambda}$ 是由 $n \times n$ 的特征值定义的对角矩阵：

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \tag{10.11}$$

对应的特征向量为矩阵 $\boldsymbol{E}$ 的列：

$$\boldsymbol{E} = [\boldsymbol{e}_1 \boldsymbol{e}_2 \cdots \boldsymbol{e}_n] \tag{10.12}$$

377

由公式（10.10）可得：

$$\boldsymbol{A}\boldsymbol{E} = \boldsymbol{E}\boldsymbol{\Lambda} \tag{10.13}$$

或者，等价于：

$$\boldsymbol{A}\boldsymbol{e}_i = \lambda_i \boldsymbol{e}_i, \quad i = 1,2,\ldots,n \tag{10.14}$$

其中，$\boldsymbol{e}_i$ 为第 i 个特征值 λ_i 对应的特征向量。

如果将特征值按降序排序，即有：

$$\lambda_1 > \lambda_2 > \ldots > \lambda_i > \ldots > \lambda_n$$

对应的特征向量代表了按照重要程度排序的主成分的方向。因此向量 $\boldsymbol{e}_1$ 代表了多维数据中具有最大方差的轴方向。

如何确定特征值

如果方阵 A 很小（不大于 4×4），我们可以找到其行列式，并将其行列式扩展为一个特征多项式：

$$\det(A-\lambda I)=0 \tag{10.15}$$

通过解特征多项式的根，我们可以确定矩阵 A 的特征值。然而，当多项式的阶高于 4 时，不能用有限序列的算术操作来解，因此通常使用迭代的方法来解（Wilkinson, 1988）。

现在我们来计算之前的协方差矩阵 C 的特征值（我们使用 MATLAB 来计算）。特征值和对应的特征向量按照降序排列：

$$\text{特征值}=\begin{bmatrix}2847.7\\ 1.3\end{bmatrix};\ \text{特征向量}=\begin{bmatrix}-0.9519 & 0.3063\\ -0.3063 & -0.9519\end{bmatrix}$$

因此，

$$\lambda_1=2847.7\quad,\quad e_1=\begin{bmatrix}-0.9519\\ -0.3063\end{bmatrix};\ \lambda_2=1.3\quad,\quad e_2=\begin{bmatrix}0.3063\\ -0.9519\end{bmatrix}$$

图 10.5 显示了在特征向量 e_1 和 e_2 形成的坐标系下规范化的试点工厂数据的平面图。从图中可以看出，其中的一个特征向量 e_1 代表了最佳拟合直线，而另一个特征向量 e_2 代表了另一个不
378 太重要的模式。所有的数据点都处于稍微偏离最佳拟合直线的位置。

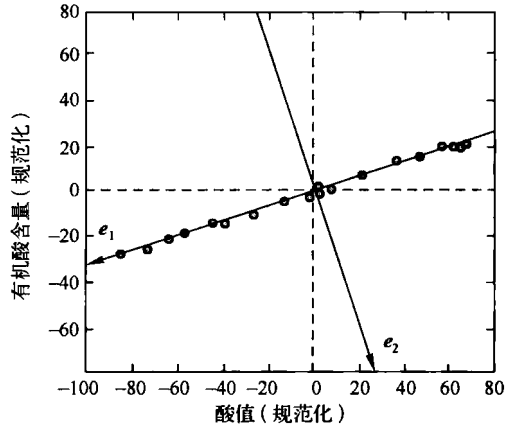


图 10.5　在特征向量 e_1 和 e_2 形成的坐标系下规范化的试点工厂数据图

协方差矩阵的特征向量为我们提供了转换空间的坐标系。现在我们需要用这些特征向量来表示我们的数据。

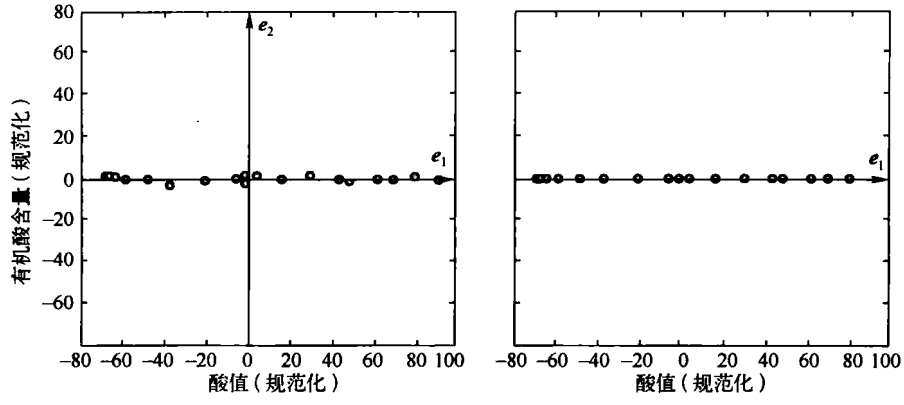
步骤 4：选择成分和生成新的数据集。

一旦确定了一组特征值并将其按降序排列，我们可以忽略那些不太重要的特征值，从而降低了原始数据集的维度。例如，如果原始数据集是 n 维的，我们取得了 n 个特征值，我们可以只保留最大的 m 个特征值。m 个对应的特征向量（eigenvector）形成了*特征向量*（feature vector）矩阵的列。然后，我们将原始数据投影到由特征向量形成的*特征空间*（feature space），从而将原始的 n 维数据集转换成了 m 维数据集。为了将原始数据集投影到特征空间，我们只需要将规范化的原始数据和特征向量相乘。这种数据转换的方法称为*子空间分解*（Oja, 1983）。

如果我们比较我们的例子中获得的特征值，我们发现 $\lambda_1 \gg \lambda_2$。因此第二个特征值可以丢弃，我们的数据集也就变成了一维的。

现在我们来比较一下两种不同的数据转换——保留所有的特征值和只保留一个最大的特征值。前一种情况下的数据平面图如图 10.6a 所示。因为在这种数据转换下没有信息丢失，所以平

面图显示的是规范化的原始数据集在旋转坐标系 e_1 和 e_2 下的情况。实际上，该图和图 10.5 显示 379
的是一样的。在第二种情况下，我们只保留一个最大的特征值，获得的平面图如图 10.6a 所示。可以看出，这实际上是一个一维平面图，所有由第二个特征向量产生的数据方差都被删除了。



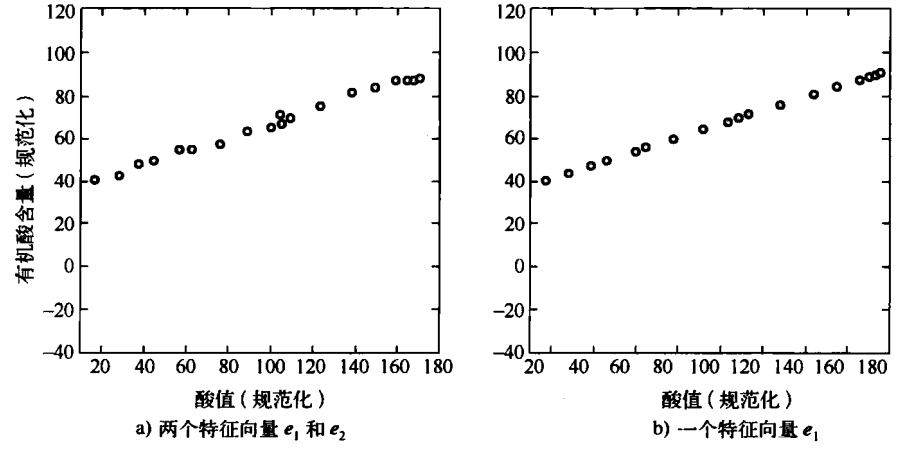
a）保留两个特征向量 e_1 和 e_2　　b）只保留一个特征向量 e_1

图 10.6　规范化的数据投影到特征空间后的平面图

如何恢复原始数据

我们需要记住的是删除了不太重要的主成分之后，会丢失一些信息。因此，除非我们在数据转换的时候保留所有的特征向量，否则我们无法精确恢复原始数据。

通常情况下，原始的 m 维数据集可以通过以下方法重建：首先将转换后的 m 维数据集与 m 维特征向量的转置相乘，然后将每一个数据值加上该维数据的平均值。图 10.7a 显示了通过完整的特征向量重建的数据的平面图。和我们期望的一致，得到是与图 10.3a 一样的平面图。图 10.7b 显示的是降维后的对特征向量的影响。可以看到，沿着主特征向量 e_1 的方差得以保留，而特征向量 e_2 方向的方差则消失。



a) 两个特征向量 e_1 和 e_2　　b) 一个特征向量 e_1

图 10.7　重构后的试点工程数据平面图

案例 11：降维

我想要开发一个不需要分离胆固醇成分，而是直接通过测量血清样本的光谱内容来确定胆固醇水平的智能工具。现在我有一组波长测量方法和间接用脂蛋白成分来测量胆固醇的方法。用什么数据挖掘工具来解决这个问题呢？

当前的胆固醇测量基于离心法将胆固醇分离为高密度脂蛋白（HDL）、低密度脂蛋白（LDL）和极低密度脂蛋白（VLDL）。各种成分中的胆固醇浓度使用分析法来计算出来后，总胆固醇水平即可确定。不幸的是，离心法非常昂贵而且耗费时间。

有学者建议，我们可以采用一个衍生化反应来直接测量胆固醇水平（Purdie 等，1992）。该衍生化反应会产生多种生成物的混合物，这些生成物对应的不同的脂蛋白成分有不同的颜色。然后利用生成的混合物，采用吸收分光技术来分析 HDL、LDL 和 VLDL 成分。从而我们不再需要将胆固醇分离为不同的成分，从而避免了使用昂贵的离心法。

在本案例中，我们测量了 264 个病人的 21 种光谱波长。对某些病人，我们还可使用标准成分分离法获得的 HDL、LDL 和 VLDL 间隔测量胆固醇含量。在给定了 21 种波长的测量值以后，我们可以训练一个反向传播神经网络，使用 HDL、LDL 和 VLDL 成分来预测胆固醇水平。

整个数据集（264 例）被随机地划分为训练集（60%）、验证集（20%）和测试集（20%）。验证集可以用来提前停止训练——用训练集来训练神经网络，直到在一段连续的迭代内，神经网络在验证集上的效果不再提高。这也指示泛化集合达到了峰值，训练可以停止，以防止过拟合。测试数据可以用来对神经网络的效果做完全独立的测试。

380
~
381

对于本案例，我们创建了一个三层神经网络。在隐含层中使用 S 形激活函数，在输出层中使用一个线性激活函数。输入层的神经元数目是由光谱波长的数目决定的，因此我们需要 21 个输入神经元。我们需要预测 3 个值（HDL、LDL 和 VLDL），因此在输出层中需要 3 个神经元。一系列的试验表明，隐含层中只使用 7 个神经元时可以达到很好的泛化。图 10.8 显示了学习曲线图。从图中我们可以看出，验证误差在第 5 步迭代时达到最小值，且在给定数目的后续迭代中不再增加，因此训练可以结束。学习曲线显示我们已经成功地学习到了一个神经网络，而且避免了过拟合问题。

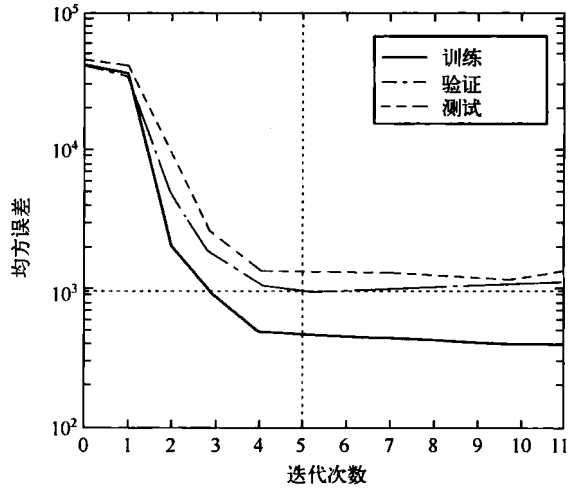


图 10.8　胆固醇水平预测问题的 ANN 学习曲线

现在我们使用测试集来检验学习到的神经网络的性能，并对预期输出和神经网络的输出作回归分析。图 10.9 显示了回归分析的结果。

散布图上的 r 值是 Pearson 相关系数——用于度量两个变量 X（预期输出）和 Y（实际输出）的相关性的系数。Pearson 相关系数取值范围为 +1（正相关）到 −1（负相关）。r 值越大，两个变量之间的相关性就越大，我们也就可以更准确地预测 HDL、LDL 和 VLDL 的值。我们可以看到，前两个输出（HDL 和 LDL）与预期输出十分接近（r 值超过了 0.9），第三个输出则与预期

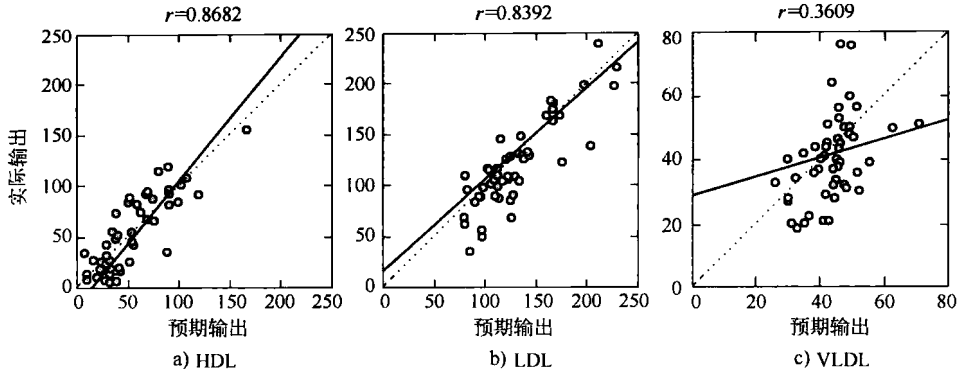


图 10.9　ANN 输出相对于目标值的散布图

输出（VLDL）不是很符合。

当然，我们可以继续通过修改神经网络的结构和增加训练的迭代次数来提高 ANN 的正确性，但是我们发现，无论怎么努力 VLDL 的 r 值都没有明显地提高。

可以使用 ANFIS 来提高胆固醇水平预测的准确性吗?

ANFIS 使用随机混合学习算法，而且其效果已经被证实比传统的反向传播技术要好。然而，它包括了大量需要在训练过程中进行最优化的参数。使用 MATLAB 来调整 Sugeno 型模糊推理系统的参数时，即便是在输入相对较小的情况下，也很容易发生内存溢出。另外需要记住的是，ANFIS 只有在训练样本数目几倍于需要最优化的参数的数目时才能达到很好的泛化。因此，在使用 ANFIS 模型预测胆固醇水平之前，我们需要减小输入量。这个任务由 PCA 来完成。

我们需要对输入数据进行规范化，计算协方差矩阵以及确定特征值和特征向量。表 10.3 包含了降序排列的特征值（主成分）。我们只显示了前 7 个主成分，因为其余的主成分都是微不足道的。现在我们需要确定需要保留多少个主成分来获得输入数据的大部分方差值。

表 10.3　主成分描述的总方差

主成分	1	2	3	4	5	6	7
特征值	0.225947	0.010530	0.000356	0.000175	0.000046	0.000031	0.000021
总方差比（%）	95.2883	4.4406	0.1500	0.0737	0.0193	0.0131	0.0088

一个简单的方法是将相对重要的主成分可视化。用于显示每一个主成分所描述的数据总方差比的折线图称为陡坡图。图 10.10 显示了一个这样的陡坡图。可以看出，陡坡图的图形很像是山的一侧，其中“陡坡”是指山脚下积累的松散岩石斜坡。我们可以忽略山脚上的主成分因为图形上的急剧下降意味着后续的主成分对整个数据的总方差贡献很小。在图 10.10 中，第一个主成分占所有 21 个变量总方差的 95.29%，第二个主成分占 4.44%，第三个主成分占 0.15%，而其余的 18 个主成分只占到总方差的 0.12%。这个图显示出只有前两个主成分代表了输入数据中的有意义的方差——他们占了 99% 以上的方差值。因此，输入向量的维度可以从 21 维降到 2 维。

382
~
383

现在，我们将原始的 21 维数据集转换成 2 维数据集，训练一个 ANFIS 模型，然后将它的效果和 ANN 的效果进行对比。因为输入向量已经变成 2 维，ANFIS 的输入层只需要 2 个神经元。我们应该选择最少的隶属函数，因此我们用一个钟形激活函数给每一个输入变量分配两个模糊神经元。最终的 ANFIS 将会包含 4 条模糊规则。因为 ANFIS 的结构只允许一个输出，我们将使用 3 个独立的模型，每一个使用一种脂蛋白成分来预测胆固醇。

我们的 ANFIS 模型一共有 24 个参数，包括 12 个输入参数（2 个输入；每一个输入有 2 个钟

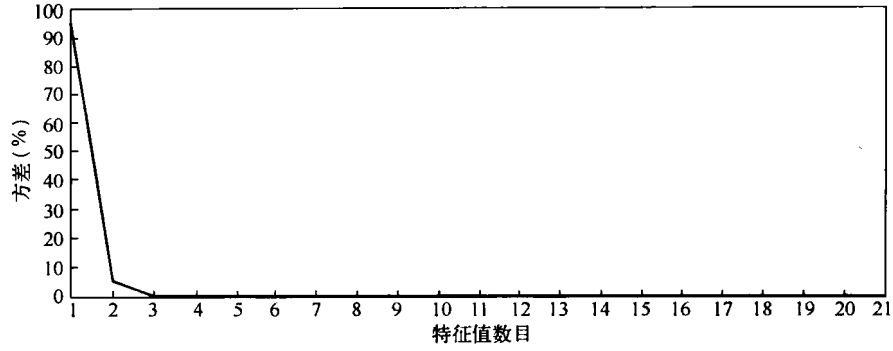


图 10.10　21 维光谱数据集的陡坡图

形激活函数，每一个激活函数有 3 个参数）和 12 个输出参数（4 条规则；每条规则用 3 个参数来描述）。

在该研究中，我们将转换后的数据随机分为两部分：训练集（80%）和测试集（20%）。每一个 ANFIS 模型的参数数量和训练样本数量的比例大约为 1 比 8，因此保证了很好的泛化。

图 10.11 显示了 ANFIS 输出相对于目标输出的散布图。3 个 ANFIS 模型都使用相同的样本数据经过 10 个周期迭代训练，并且使用相同的测试集测试。可以看出，虽然 ANFIS 是在降维后的
384 二维数据集上训练的，但它的效果仍然比 ANN 要好。

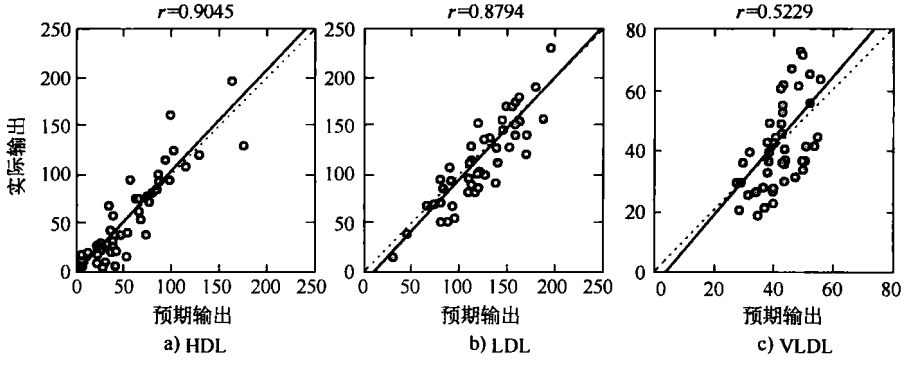


图 10.11　ANFIS 输出相对于目标值的散布图

如何用神经网络实现 PCA?

许多特定的神经网络和学习算法被用来实现 PCA。举个例子，我们可以使用一个两层前馈线性网络，其中输入层的 n 个神经元表示 n 维数据空间，输出层的 m 个神经元提取 m 个主成分。这样的网络称为 PCA 网络，如图 10.12 所示。

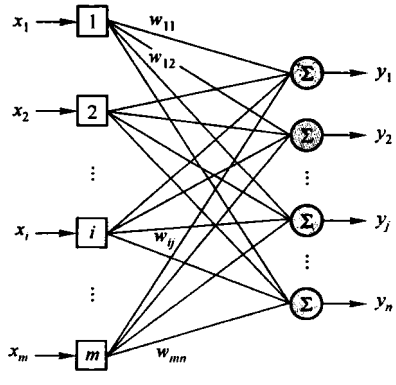


图 10.12　一个 PCA 神经网络

训练一个 PCA 神经网络将会产生一个与最大的特征向量 $\boldsymbol{w}$ 相等的权重向量 $\boldsymbol{e}_1$，因此满足以下条件：

$$\mathbf{A}\boldsymbol{e}_1 = \lambda_1 \boldsymbol{e}_1 \tag{10.16}$$

385 其中，$\boldsymbol{A}$ 是输入自相关矩阵。

PCA 网络的权重使用 Oja 规则来迭代地更新（Oja，1982）：

$$w_{ij}(p+1) = w_{ij}(p)[1 - \phi y_j^2(p)] + \phi x_i(p) y_j(p) \tag{10.17}$$

其中，ϕ 为遗忘因子。

当 PCA 网络收敛到最大的特征向量后，输出向量被从输入向量中减去，训练重新开始。这个过程被不断重复直到找出了 n 个主成分。

在数据挖掘领域，常常将 PCA 当作一种数据预处理技术。它让我们得到一个远远小于原始数据集的新数据集，同时产生一样（或者几乎一样）的分析结果。

PCA 等统计方法往往作为数据探索的第一步，数据库查询和 OLAP 则代表了一些为了处理现代组织积累的大量数据而特别开发的复杂工具。

10.4　关系数据库和数据库查询

很多组织会收集和存储大量的数据，他们需要对这些数据进行高速访问以快速应对市场的变化。

现代组织如何维护大型数据库

数据库的维护，既能使用人工的方式，也能使用计算机化的方式。例如我们可以人工地维护一个索引地址簿，记录朋友的姓名和联系方式——我们简单地添加一个姓名并覆盖已存在的地址和电话信息，如果它们发生了变更。当然我们可以使用个人信息管理工具来代替上述“旧技术”，使我们的个人数据库计算机化。

然而，我们需要处理的是大型数据库，因此也就没有了其他的选择，我们需要将数据库计算机化，而不能简单地由人工维护。换句话说，我们需要一个*数据库管理系统*。

什么是数据库管理系统

我们首先回到数据库的定义上。*数据库*是存储在一起的相关数据的集合。这个定义实际上是非常宽泛的。例如我们可以将组成本章的所有词的集合看做一个数据库。然而，“数据库”一词通常与以下隐含的特性有关（Elmasri and Navathe, 2010）：

- 数据库代表真实世界中的实际事件。
- 数据库是包含特定含义的数据的逻辑集合。
- 数据库是针对特定应用构建的，因此拥有特定的用户群体。

386

数据库必须进行组织和管理，以使用户能够更新数据以及查询和检索他所需要的数据子集。帮助用户创建和维护数据库的一组程序称为*数据库管理系统*（DBMS）。数据库管理系统使得创建、发布、操纵以及在不同的用户和应用之间共享数据库变得可能。*创建*数据库包括定义数据库结构和描述将要存储的数据的类型和约束。*发布*数据库的意思是将数据输入数据库并存储在某些存储媒介中。*操纵*数据库通常包括更新数据库、查询特定数据和产生报表等功能。*共享*数据库暗含着支持多个用户对数据库的并发访问。

要作出明智的决策，数据库用户需要能够对大量的数据进行快速访问。在组织中，数据通常存储为数据表和电子表格的集合。*关系数据库*是一组正式描述的数据表的集合，这些数据表能够在不重组织数据的情况下进行访问和重组。关系数据库模型最早由 IBM 公司的 Ted Codd 于 1970 年在他的论文“A relational model of data for large shared data banks′”介绍（Codd, 1970）。从 20 世纪 80 年代早期开始，该模型在许多商业的数据库管理系统实现，例如 Oracle 和 Rdb（Oracle）、IDS（IBM）以及 SQL Server 和 Access（Microsoft）。

在关系数据库中，每一个表（或称为*关系*）被赋予一个唯一的名称。表又划分为行和列。列（或字段）描述了表的*属性*，行（称为*元组*）则保存这些属性的值。当表创建以后，用户需要正式描述每一列的数据类型。典型的数据类型有：*字符型*（保存一串字母、数字、等长或定长的字符，例如“DC086421”）、*整数型*（保存正数，例如 14）、*十进制数*（保存十进制数，例如 5.65）、*日期*（保存日期，例如“2010 - 10 - 22”）和*时间*（保存时间，例如“11：52：12”）。

关系数据库能够很容易地创建和访问，更重要的是，它能够简单地进行扩展。原始数据库创建以后，我们可以在不修改已有应用的情况下添加新的数据属性。

下面我们来看一个例子。图 10.13 显示了数据库 Department_store 的不完整关系 Item、Customer、Employee 和 Transaction。表 Item 中的列包括 Item_ID、Category、Brand、Model、Description、Price 和 Stock。Item_ID 属性的数据类型为 char(8)，保存长度为 8 的定长字符串。Category 属性的类型为 varchar(15)，保存最大长度为 15 的变长字符串。Price 属性的类型为 decimal(6, 2)，保存一个小数点前最多 6 位、小数点后 2 位的十进制数。Store 属性的类型为 int(4)，保存
387 位数最多为 4 的整数。类似地，我们可以声明其他数据表 Customer、Employee 和 Transaction 中所有属性的数据类型。

Item

Item_ID	Category	Brand	Model	Description	Price	Stock
DC056421	Digital Camera	Canon	Rebel Xsi	12.2 MP/18-55 mm	899.84	121
DC096332	Digital Camera	Nikon	D80	10.2 MP/18-135 mm	1024.84	64
...	...	...	...	...	...	...

Customer

Customer_ID	Name	Age	ZIP_code	State	Address	Income
016745	Smith,Alex	26	4851	NY	186 1st Ave,New York	95000
016749	Green,Jane	19	1419	NY	7 Lake Ave,Rochester	52000
...	...	...	...	...	...	...

Employee

Employee_ID	Name	Age	Department	Job_category	Salary	Commission
0329	Jones,Mary	25	Electronics	Sales	37500	3
037H	Brown,Mark	21	Hardware	Sales	29000	2
...	...	...	...	...	...	...

Transaction

Trans_ID	Item_ID	Payment	Date	Time	Customer_ID	Employee_ID
345728	DC096432	Visa	2008-10-20	11:32:12	016745	032D
...	...	...	...	...	...	...

图 10.13 关系数据库 Department_store 的不完整关系

在关系数据库中，数据表之间相互“关联”，这保证了数据可以进行跨表连接，而不需要在每一个表中重复保存。在我们的例子中，Transaction 表包括 Item_ID、Customer_ID 和 Employee_ID 三个属性，它们分别也是 Item 表、Customer 表和 Employee 表的属性。

然而，仅仅描述关系数据库的数据结构并与关系和数据一起发布并不是高效的数据管理。为了将来进行数据分析，我们还需要能够从数据库中描述、访问和检索数据。

如何从关系数据库中检索数据

关系数据库中的数据可以使用数据库查询来访问。数据库查询使用结构化查询语言（SQL）来进行。SQL 由一些支持对数据进行检索、插入、更新和删除的命令语言组成。SQL 是 ANSI
388 （美国国家标准学会）和 ISO（国际标准组织）访问和操作数据库系统的标准计算机语言。

SQL 的第一个版本称为 SEQUEL，它是由 IBM 公司的 Donald Chamberlin 和 Raymond Boyce 于 20 世纪 70 年代早期开发的（Chamberlin and Boyce，1974）。SEQUEL 是为 IBM 的 R 系统开发，R 是第一个原型关系数据库系统，随后 SQL 被 ANSI 标准化，自此它成为大部分商业数据库管理系统中极为重要的一部分。虽然现在存在许多版本的 SQL，但它们都支持一些主要的关键字（例如 SELECT、FROM、WHERE、UPDATE、DELETE、INSERT、AND、OR 和 NOT）。

在下面的例子中，我们将使用 MySQL 数据库系统（Forta，2005；DuBois，2008）。MySQL 既能以开源产品的方式（最流行的开源数据库系统）获取，也可以购买一个标准的商业许可证。MySQL 程序以服务器端形式运行，提供多用户对多个数据库的访问。

利用 SQL，我们可以查询关系数据库和检索特定的数据子集。假设商店管理员想要检查当前尼康数码相机的库存量。如图 10.14 所示，商店管理员使用一条简单的查询来访问数据库，就可以获得他所需要的数据子集。在这个查询中，出现了两个关键字 SELECT 和 FROM。实际上，大部分的基本 SQL 查询类似于下列查询：

```
SELECT*FROM Item
WHERE Category= 'Digital Camera' AND Brand= 'Nikon'
```

Item_ID	Category	Brand	Model	Description	Price	Stock
DC086527	Digital Camera	Nikon	L18	8.0 MP/3 × Zoom	139.84	186
DC086532	Digital Camera	Nikon	D80	10.2 MP/18–135 mm	1074.84	64
…	…	…	…	…	…	…

图 10.14　一个单表上的 SQL 查询的例子

```
SELECT column_name FROM table_name
```

这个查询从名称为 table_name 的表中查询名称为 column_name 的列的内容。注意，我们可以使用多个列名，也可以使用多个表名。查询结果保存在一个结果表（称为结果集）中。

在图 10.14 中所示的查询中，我们使用符号 * 而不是列名来选择表 Item 的所有列。然而，商店管理员只需要检索尼康数码相机的库存量。为了实现这个目的，需要在 SELECT 语句中添加一个 WHERE 子句。这条查询的返回结果是 Item 表中 Category 属性的值为“Digital Camera”且 Brand 属性的值为“Nikon”的所有记录的列表。注意，选择条件在 WHERE 子句中用关键字 AND、OR 和 NOT 隔开。在这个例子中，关系查询作为一个过滤器，使用户可以通过描述一些约束来检索所选择属性的数据子集。

在大多数情况下，我们需要从两个或多个表中检索数据。假设商店管理员想要查看 VistaQuest 紧凑型数码相机最近的促销信息。这些信息可以从 Item 和 Transaction 两个表中获取。 389
从图 10.13 可以看出，这两个表有一个公共属性 Item_ID，因此我们可以通过匹配 Item_ID 列来从两个表中提取数据。图 10.15 显示了所使用的 SQL 查询和得到的结果集。在这个查询中，FROM 子句用来连接两个表。在使用列时我们应该指明其所在的表，因为不同的表中可能出现相同的列名。我们通过在表名后加一个“.”符号然后再加上列名来实现，如查询中的 SELECT 子句所示。WHERE 子句表明两个表各自的 Item_ID 属性必须匹配。BETWEEN…AND 操作用来查询两个日期范围内的数据，从“2010 - 10 - 23”到当前日期。

上述查询的返回结果是从促销活动开始所售出的 VistaQuest 相机的信息列表。结果表实际上可以包含上百甚至上千的行。但是，这样的表可能没有什么用处——管理员需要的实际是包含售出相机的总量的汇总表。这可以通过使用 SQL 内置的一些聚集函数来完成，这些聚集函数包括 AVG（返回一列的平均值）、COUNT（返回一列包含的行数）、MAX（返回一列的最大值）、MIN（返回一列的最小值）和 SUM（返回一列的和）等。图 10.16 显示的是使用了

```
SELECT Item.Brand,Item.Model,Item.Price,Transaction.Date
FROM Item,Transaction
WHERE Transaction.Item_ID=Item.Item_ID
ADN Item.Brand='VistaQuest'
ADN Transaction.Date between'2010-10-23'and curdate()
```

Brand	Model	Price	Date
VistaQuest	VQ-7024	69.84	2010-10-23
VistaQuest	VQ-5115 Pink	54.84	2010-10-23
VistaQuest	VQ-5015 Silver	54.84	2010-10-23
VistaQuest	VQ-7024	69.84	2010-10-23
VistaQuest	VQ-5115 Blue	54.84	2010-10-24
VistaQuest	VQ-7024	69.84	2010-10-24
…	…	…	…

图 10.15　一个多表上的 SQL 查询的例子

COUNT 和 SUM 函数的 SQL 查询。GROUP BY 子句用来指明 SUM 函数必须对每一个唯一的 Model 值都执行。利用查询的结果集，商店管理员可以通过对比 VistaQuest 相机当前的销售额和促销开始前的销售额，来检查促销活动的收入。

```
SELECT Item.Brand,Item.Model,COUNT(Item.Model),SUM(Item.Price)
FROM Item,Transaction
WHERE Transaction.Item_ID=Item.Item_ID
AND Item.brand='VistaQuest'
AND Transaction.Date between'2010-10-23'and curdate()
GROUP BY Model
```

Brand	Model	COUNT(Item.Model)	SUM(Item.Price)
VistaQuest	VQ-5015 Silver	34	1864.56
VistaQuest	VQ-5115 Blue	42	2303.28
VistaQuest	VQ-5115 Pink	39	2138.76
VistaQuest	VQ-7024	93	6495.12
…	…	…	…

图 10.16　一个带有聚集函数的 SQL 查询的例子

数据库查询和数据挖掘的区别是什么

数据库查询可以看做是数据挖掘的基本形式。查询确实为用户提供了向关系数据库提问和获取
390 满足特定条件的数据子集的机会。例如，通过使用关系查询，我们可以提取一些信息：上个月售出的所有产品列表；以产品类别、品牌和型号分类的总销售额；过去 12 个月消费超过 1000 美元的顾客列表；或者过去一个月销售额超过 10000 美元的员工列表。实际上，我们还能获取更多的信息。

假设我们想开展一个针对性的活动来促销一款新产品。为了使促销活动取得成功，我们需要知道潜在顾客的一些主要特征，如年龄、收入、婚姻状况、房产、债务，等等。当然，我们可以通过数据库获得过去购买了类似产品的顾客列表。利用 SQL 聚集函数，我们可以确定潜在顾客的平均年龄、平均收入和平均债务。如果结果集不是很大，我们还可以识别出一些潜在顾客的公共特征，例如房产。然而，所有的这些信息并不能揭示我们需要找到的实际群体——高收入、高债务的年轻人和低收入、低债务的老年人可能对同一款产品感兴趣。数据挖掘可以自动发现这些隐含的群体。

数据库查询是基于假设的——用户必须提出正确的问题。与查询相反，数据挖掘自动建立一个数据模型；发现数据模式、查询趋势、泛化和执行预测。数据挖掘往往会使用数据库，因为数据库代表了最常用和丰富的数据宝库。

10.5　数据仓库和多维数据分析

关系数据库成熟于 20 世纪 80 年代，它导致了数据管理系统和业务的运作自动化的广泛应用。

然而，随之也出现了一个新的挑战——如何在整个组织或商业中获得一个更广泛的视野。
391 这个挑战在 20 世纪 80 年代晚期到 20 世纪 90 年代早期，当第一个商业*数据仓库*（data warehouses）建立的时候得到了回答。数据被从多个操作型数据库聚集到一起，以给组织管理者提供一个全公司范围的视野。

什么是数据仓库

“数据仓库”一词最早由“数据仓库之父”Bill Inmon 于 20 世纪 90 年代初期提出。他把数据仓库定义为“一个面向主题的、集成的、反应历史变化的、相对稳定的数据集合，用于支持管理决策”（Inmon，1992）。数据仓库是主要为决策支持应用而设计和构建的大型的集成的数据库。

数据仓库的主要特征是什么

数据仓库的一个主要特征是它的规模。数据仓库规模很大——它包括上百万甚至上十亿的

数据记录。然而，这些还不是数据仓库的全部。在数据仓库的定义中，Bill Inmon 描述了数据库的四大主要特征：面向主题、集成性、时变性和稳定性。

*面向主题*代表了数据仓库与关系数据库的根本区别。与关系数据库是为日常商业运作而构建不一样，数据仓库是为了分析特定的主题而开发。举个例子，在银行业中，关系数据库包含了客户存款、贷款和一些其他事务的独立的数据记录；数据仓库将这些数据集中到一起来提供酬金、利润和亏损等特定金融信息。与决策无关的数据并不包括在数据仓库中。

*集成性*意味着数据仓库中的数据来源于多个不同的数据库，包括在线的操作型数据库、关系数据库和普通文件。数据被清洗以后被集中到一个一致性的数据库中。

*时变性*反映出数据仓库中的数据属于一个特定时间内。用户可以生成指定时间区间内的报表，因而可以观察数据随时间的变化。

*稳定性*是指数据库中的数据是稳定和持久的。数据仓库通常包含几年的有价值的数据（相反，一个典型的关系数据库大约只有 60 ~ 90 天的时间跨度）。实际上，数据仓库往往不要求最新的关系数据，也就是说数据仓库中的数据是非实时的。

在商业决策中，往往需要管理者对业务拥有一个更广泛的视野，忽略特定的日常活动的细节。数据仓库使得用户可以查看特定的值随着时间推移的改变情况，从而发现数据中的趋势。查询通常交互式地执行，一个查询的结果能够引出下一个查询。

我们可以把数据仓库看做一个单独的关系数据库系统，这个数据库系统中包含多个操作型数据库系统的数据备份。 392

为什么需要将数据从一个系统复制到另一个系统？为什么我们需要一个单独的数据仓库

操作型数据库管理系统是为业务运作的自动化而设计和构建的。例如，百货公司完成一项交易后，一条新的数据记录被插入到 Transaction 表中，所有相关记录（例如库存、员工和顾客）也随之更新。操作型数据库系统关注的是日常的操作，无需保存历史数据。另一方面，如果我们想用一个操作型数据库系统来做商业决策，我们需要获得很长时间段（通常是 10 年）的数据。如果我们要执行决策支持查询，系统性能将会下降。每当执行一条这样的查询时，系统会变得很缓慢甚至崩溃。

更为严重的是，不同的操作型数据库往往运行在不同的硬件平台和操作系统中。有些应用只是针对特定平台开发。运行在不同机器和不同物理位置上的分布式数据库系统可能是不兼容的。

这些就是我们需要从不同的操作系统复制数据、将数据保存为统一的格式和将数据保存在一个单独的数据库即数据仓库的原因。

如何设计数据仓库

数据仓库是另外一种类型的数据库，需要采用不同的设计方法。数据仓库包含更大规模的数据而且支持复杂的业务类型查询。然而，数据库越大，查询起来也就越困难。

关系数据库基于关系数据模型。这个模型表示了许多关系表之间的交互。关系数据模型特别适合 SQL 查询，它只能提供业务的平面二维视图，不太适用于决策。

另外一种方法是使用*多维数据模型*。关系数据模型研究单个的关系，例如 Item、Customer 或 Transaction，以及它们之间的联系。而多维数据模型将数据看做是一个*数据立方体*(data cube)。

什么是数据立方体

数据立方体提供了一种从多种不同的视角来观察数据的方法。举个例子，像沃尔玛等经营百货商场连锁店的大型公司，其管理者会发现它们在纽约的经营状况要比加州的好。管理者还可以找出本季度经营状况最好的商场。视频游戏的销售是否比手机更好？数据立方体还可以让管理者发现新的趋势。 393

数据立方体可以被定义为一种表示多维聚合数据的方式。数据立方体，或多维数据模型，使

用维度和事实来描述数据（Gray et al.，1996；1998）。

维度表示我们保持我们的记录，并生成数据仓库查询的实体。举个例子，零售商创建一个数据仓库来保持关于 Time（事务发生的时间）、Item（售出的商品）和 Employee（商品出售者）的商场销售记录。如果数据仓库是为拥有多个连锁门店的公司创建，我们还需要添加一个 Store 维度（事务发生地点）。维度可以被定义为一个逻辑上相关的属性的集合。例如，Time 维通常包括 date、week、month、quarter 和 year 等属性，而 Item 维则可能包括 item_ID、model、brand、category 和 department 等属性。每一个维度都有一个与之相关联的表，称为维度表。

数据立方体围绕一个中心的主题而组织，例如一个零售数据仓库往往以销售（sales）作为中心主题。主题用事实来表示，即数据仓库中存储的特定数据。事实用一些度量来表示，这些度量包括销售额和销售量。事实保存在事实表中。事实表是数据仓库中的一张大表——它保存了商场中的大量销售的详细情况。因此，如果我们需要保存一段重要历史时期（数年而不是数天）的销售记录，事实表会变得非常大，可能包含数亿条行记录。幸运的是，事实表中的数据并不必需要最低层的详情，它只需要维护汇总信息，例如产品的每日总销售量。对于决策，我们并不需要知道每一个事务的具体细节。

事实表中包含大量的数据并且被定期更新（通常只是附加数据），而维度表则负责维护查询完成后用于组合数据的一些属性。这些属性虽然会随着时间改变，但是它们却很少被更新。例如，员工或顾客可能会变更他们的地址，或者新的产品类别被添加。这种现象称为渐变维度（Kimball and Ross，2002）。

虽然在几何上立方体是三维的，但是数据仓库中的数据立方体是多维的。数据立方体可以看做是一个多尖的星星，如图 10.17 所示。星星的中心对应事实表，星星的尖则表示维度。每一个维度有一个包含多层抽象的层次结构。例如，Time 维有 5 个层次水平：date、week、month、quarter 和 year。整个星形结构称为星网模型（Han and Kamber，2006）。



图 10.17 数据立方体的星网模型表示

星网模型使用传统的父母－子女层次关系。例如，week 是 month 的子女，month 是 quarter 的子女，等等。父母负责保存子女中包含的数据的汇总和或主卷。星网模型使用户跨越任意数量的维度来浏览数据。用户可以对数据执行上卷操作（例如给 Time 维，从 month 到 quarter 和从 quarter 到 year）或下钻操作（在 Store 维，从 state 到 city 和从 city 到 store_ID）。因此，我们可能执行以下的查询：某个产品（例如 Nintendo 视频游戏）的总销售量；上一周（月或季度）时间内在某一个城市（例如 Trenton）或者州（新泽西州）购买的产品（相机、电话、计算机、电视或者

视频游戏）的类别。我们也可能将这些销售图表与前一年同一时间段进行对比，毕竟销售的增长是业务发展的很好指示。所有的这些查询都是从三个不同维度（Item、Store 和 Time）的不同层次水平来获取数据。

数据立方体模型和魔方很类似，用户可以很轻易地从一个维度组合转换到另一个维度组合。当我们创建数据立方体时，从事实表（数值数据）中得到事实，并且按照 store、quarter、store 和 quarter 或者任意可能的维度组合和它们的层次水平来对数据进行汇总。

数据立方体如何工作

为了理解数据立方体是如何工作的，我们以一个简单的二维立方体的例子来作说明。实际上，二维立方体可以表示为一个单独的表，如图 10.18 所示。这个表中包含电子部门的销售数据。数据表示为 Time 和 Item 两个维度。注意，表中包含一列和一行，分别保存列和行上的总和。这些总和通常被称为“边缘”，因为它们出现在表的“边缘”。然而，对于一个大的数据集，对每一个查询都计算边缘是很低效的。因此，通常数据会被聚合和存储，从而使得那些原本需要复杂处理的高层报表经过简单的选择过程就可以及时生成。

394
395

Time month between MonthName (1,true) and MonthName(3,true)
Item department= 'Electronics'

	Audio	Cameras	Cell phones	Computers	TVs	Video games	Total
Jan	$532 211	$612 765	$227 482	$871 043	$692 015	$314 806	[illegible]
Feb	$529 654	$603 187	$211 285	$854 105	$691 892	$306 021	[illegible]
Mar	$561 290	$613 951	$218 572	$896 202	$712 020	$369 142	[illegible]
Total	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

图 10.18　二维数据立方体的表格表示

现在我们给二维数据立方体添加第三个维度。例如，我们需要添加一个 Store 维来查看位于新泽西州的商城的电子销售情况。图 10.19 显示了表格形式的三维数据立方体。可以看出，这个三维数据立方体包含一系列的表。

Time month between MonthName (1,true) and MonthName (3,true)
Item department= 'Electronics'
Store state = 'NJ'

	Trenton,NJ						
	Audio	Cameras	Cell phones	Computers	TVs	Video games	Total
Jan	$204 518	$213 107	$80 522	$318 582	$214 328	$86 232	[illegible]
Feb	$201 386	$208 425	$76 341	$317 374	$212 230	$82 349	[illegible]
Mar	$206 651	$214 931	$83 711	$323 485	$215 234	$87 027	[illegible]
Total	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

…　　…　　…

	Saddle Brook,NJ						
	Audio	Cameras	Cell phones	Computers	TVs	Video games	Total
Jan	$164 460	$142 097	$64 292	$213 487	$132 348	$63 927	[illegible]
Feb	$161 507	$136 374	$56 348	$243 970	$134 082	$61 402	[illegible]
Mar	$167 934	$143 983	$61 323	$245 092	$134 961	$64 920	[illegible]
Total	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

…　　…　　…

图 10.19　三维数据立方体的表格表示

三维数据立方体也可以表示为如图 10.20 所示的立方体。Time、Item 和 Store 三个维度形成了立方体的坐标系，事实包含在立方体的单元格中。单元格位于从每一维取出一个成员的相交

[396] 处。例如，成员 Jan、Audio 和 Trenton 代表的单元格包含一月份新泽西州特伦顿市的音像销售额（204 518 美元）。聚合数据位于立方体的表面、边和角落。

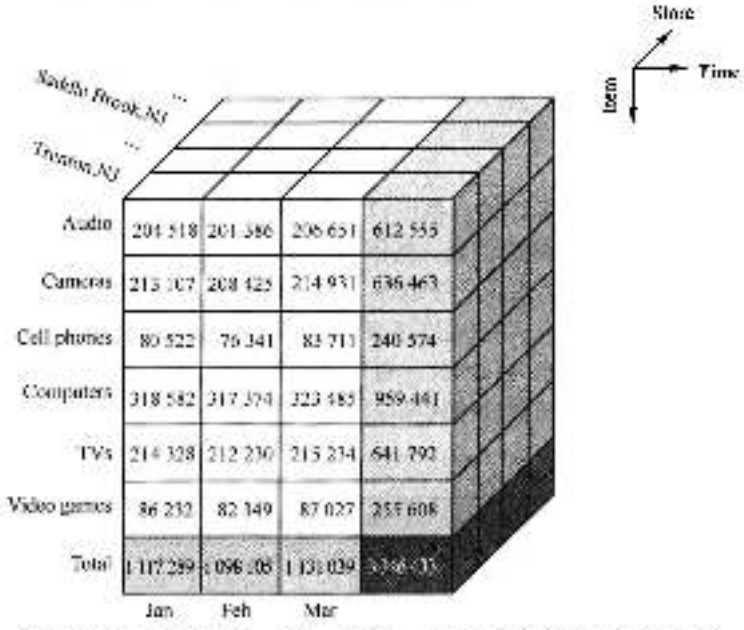


图 10.20 对应于 Time、Item 和 Store 三个维度的三维数据立方体

虽然查看三维以上的事物很困难，但是我们可以将任意的 $n-D$ 维数据立方体显示为一系列的 $(n-1)-D$ 维立方体。因此，如果在我们的数据立方体中添加第四个维度，例如 Customer 维，我们将得到如图 10.21 所示的一系列的三维立方体。然而，我们应该记住，"数据立方体"一词仅仅是一个比喻，它只是用来表达多维数据模型的概念。在现实中，数据立方体表示为电子表格、图表和图形等"标准"格式。

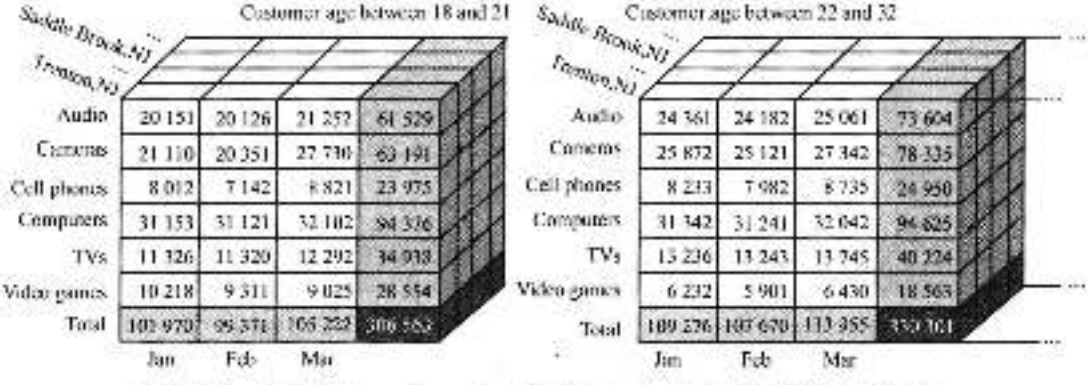


图 10.21 对应于 Time、Item、Store 和 Customer 四个维度的多维数据立方体

因为数据立方体包含聚合形式的数据，它似乎在我们提出一个问题之前就已经知道了答案。例如，如果我们询问年度和城市总销售额，这些数字已经是现有的；如果我们询问季度、类别和供应商的总销售额，这些数字也是现有的。这也是数据立方体如此非凡的原因。你提出的任何问题可以及时（通常是在一秒之内）获得答案。

如何构建和运行自己的数据立方体

支持大规模数据的多维视图和快速访问的软件技术称为联机分析处理（OLAP）。联机分析

处理的需求最早在 Codd 和 Associates 的白皮书中被形式化（Codd et al.，1993）。联机分析处理使我们可以通过构建和运行自己的数据立方体来对数据进行高效的分析。

“联机分析处理”一词的创造是为了和“联机事务处理”（On - Line Transaction Processing，OLTP）一词进行区别，以强调 OLAP 的分析能力。

OLTP 系统可以操作大量涉及多个和复杂关联的关系表，它们能够以光速更新和删除大量的记录。然而，使得 OLTP 系统能够快速和准确地执行事务操作的特别设计却让它在做数据分析时很困难。OLTP 可以使用 SQL 执行简单地查询，然而这些查询只使用了相对较小的数据集。分析处理（按照类别、州和季度来分组统计总销售额）要求我们能够提取更大规模的数据，所以需要运行所谓的“存储过程”——编译成一个单独执行计划的 SQL 声明的组合。这些存储过程可能需要数小时来运行。另外，存储过程还会让 OLTP 系统本身变得很慢甚至停止。

OLTP 系统是为管理操作型数据库而开发，而 OLAP 系统则是为了分析处理数据仓库中的数据而开发。创建数据仓库使得 OLAP 数据库从 OLTP 数据库中分离出来。将数据从操作型数据库复制到数据仓库，使得 OLTP 系统能够在持续高效运行的同时，还保证了 OLAP 系统可以支持商业决策所需要的复杂的查询。OLAP 运行数据立方体使得用户可以从不同的视角来查看数据。

OLAP 如何便于不同视角的数据查看？OLAP 可以执行什么操作

多维查询本质上是一个通过沿着维导航、增加或降低详细程度，从而发现“有趣的”趋势和关系的探索过程。完成这个过程需要多个 OLAP 操作。OLAP 可以在数据立方体中对数据进行“切片”和“切块”，“下钻”和“上卷”。下面我们来看看主要的 OLAP 操作。

397
~
398

- *上卷*（roll up）。上卷操作通过在选定的维的层次结构向上攀，对数据立方体中的数据进行聚集。例如，在图 10.17 所示的数据立方体中，将 Store 维从 city 层上升到 state 层，再到 country 层，我们可以分别按照州和国家来聚集数据。

上卷操作也可以通过降维（即从给定立方体中删除一个或多个维）来完成。例如，如果我们有一个三维立方体，包括 Time、Item 和 Store 三个维（如图 10.20 所示），上卷操作可以通过删除 Item 维来完成。得到的二维立方体表示按 Store 和 Time 汇总的总销售。

- *下钻*（drill down）。下钻是上卷的逆操作。下钻可以通过在给定的数据立方体中沿着选定维的层次结构下降或引入新的维来实现。下钻操作为我们提供了更加详细的数据。
- *切片*（slice）。切片操作通过在数据立方体给定的维的层次结构中选择单一的成员来形成一个子立方体。例如，在图 10.20 所示的三维立方体中，如果在 Store 维的 city 层选择 Trenton 成员，我们将得到一个二维的子立方体。实际上，数据立方体的“切片”类似于图 10.19 所示的交叉表格报表。
- *切块*（dice）。切块操作和切片操作类似，不过它选择两个或更多的成员。切块操作通过限制给定立方体的所有维来定义一个子立方体。例如，通过如下限制图 10.20 所示的三维立方体的两维：

```
Time.month between MonthName (1, true) and MonthName (2, true)
AND (Item.category = 'Audio' or Item.category = 'Cameras')
```

我们得到的是如图 10.22 所示的子立方体（仍然是三维的）。

- *转轴*（pivot）。转轴（又称旋转）操作改变报表或页面的坐标轴方向。例如，我们可以交换如图 10.18 所示的报表的行和列。初始报表以 Item 维的成员为列，以 Time 维的成员为行；转轴操作提供了一个以月份为列和产品为行的报表。

数据立方体可以使用基于 SQL 的查询语言来构建（确切地说是定义）（Bulos and Forsman，2006；Celko，2006）。SQL 语句被扩展以便于 OLAP 查询。例如，Oracle 10g 数据库提供了排名（ranking）、移动窗口聚集（moving - window aggregate）、时期比较（period - over - period

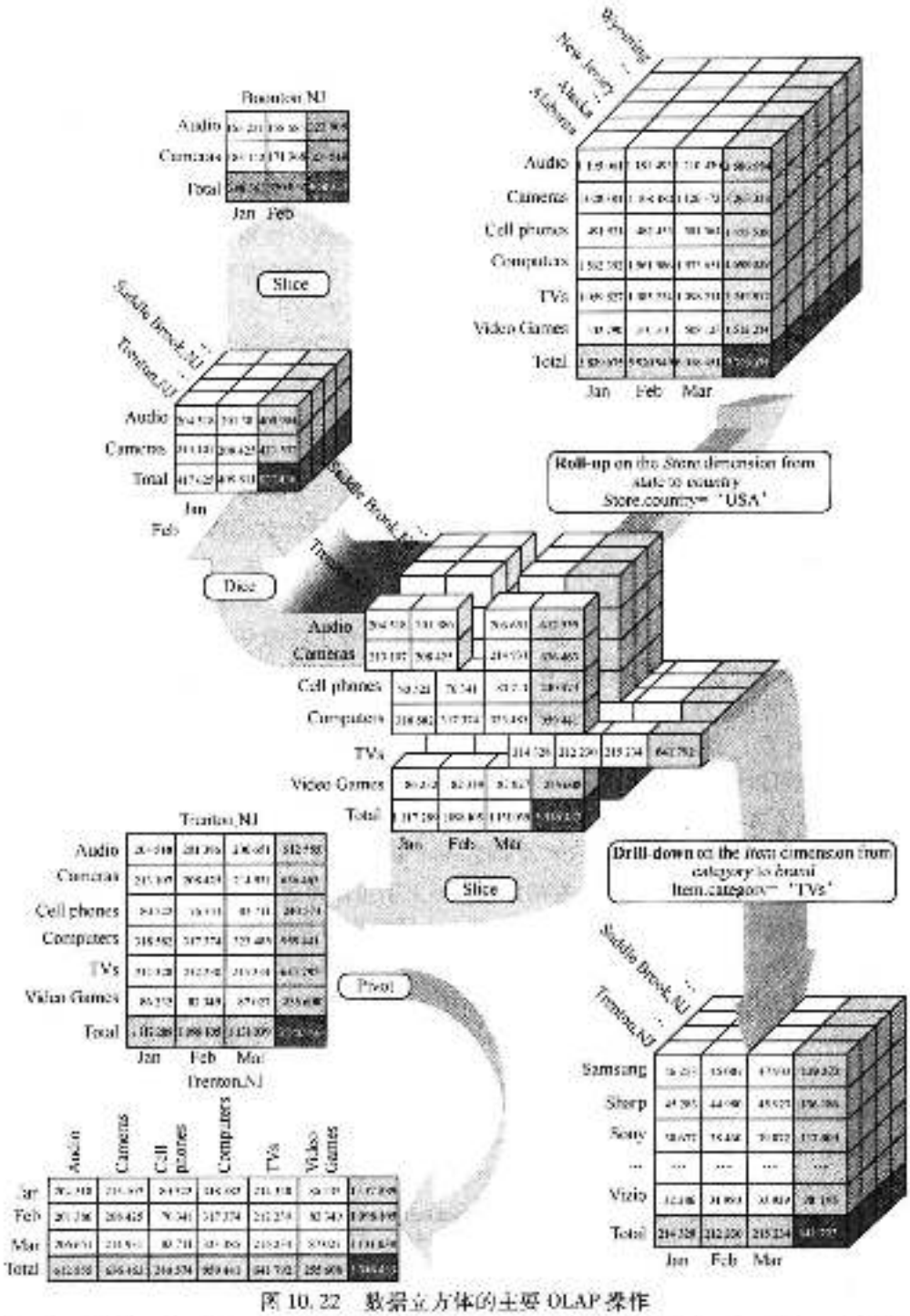


图 10.22 数据立方体的主要 OLAP 操作

399 / 400

comparisons）、统计（statistics）、直方图（histogram）、假设排名（hypothetical rank）和分布（distribution）等函数（Hobbs et al.，2005）。

什么是 OLAP 数据挖掘

OLAP 是一种强大的分析工具。它支持数据仓库中数据的多维视图。OLAP 通过“切片”和“切块”、“下钻”和“上卷”等操作提供交互式用户直接（user-direct）的数据分析。这些操作、

实际上是数据挖掘操作，确实可以从数据中发现新的知识。然而，数据挖掘代表了数据分析的更广阔的视野。它不仅能执行数据汇总和比较任务，还能解决分类、预测、聚类、关联分析和时序分析等复杂的机器学习问题。

OLAP 是一种需要用户主动参与的用户直接（user-direct）的数据分析工具，而数据挖掘通常和从数据中自动提取知识有关，它使于从存储在数据库的数据中发现隐含的模式和自动对新数据进行预测。利用 OLAP 工具，我们往往通过一个假设来开始我们的分析，然后查询数据仓库来证明或反证假设。利用数据挖掘工具，提出正确的问题的压力被转移给计算机。

有意思的是，分类、预测和关联等数据挖掘功能已经被集成到现代的 OLAP 系统中。例如，Microsoft SQL Server 和 Oracle 10g 数据库不仅允许用户访问数据立方体，而且允许用户使用数据挖掘工具（Hobbs et al., 2003; Han and Kamber, 2006）。

虽然数据挖掘经常依赖于神经网络和神经－模糊系统等智能技术，但最流行的工具是决策树和购物篮分析。

10.6　决策树

决策树可以定义为一个推荐过程的示意图。它使用一个类似于树形的结构来描述数据集。决策树特别适合解决分类问题。

图 10.23 显示一个用于识别很可能响应新产品如银行服务的促销活动的家庭的决策树。通常，这个任务是通过确定家庭在过去对类似产品的促销活动进行响应的人口统计特征来完成。家庭用其房产、收入和银行账户类型等来描述。数据库（名称为 Household）中的一个字段用来

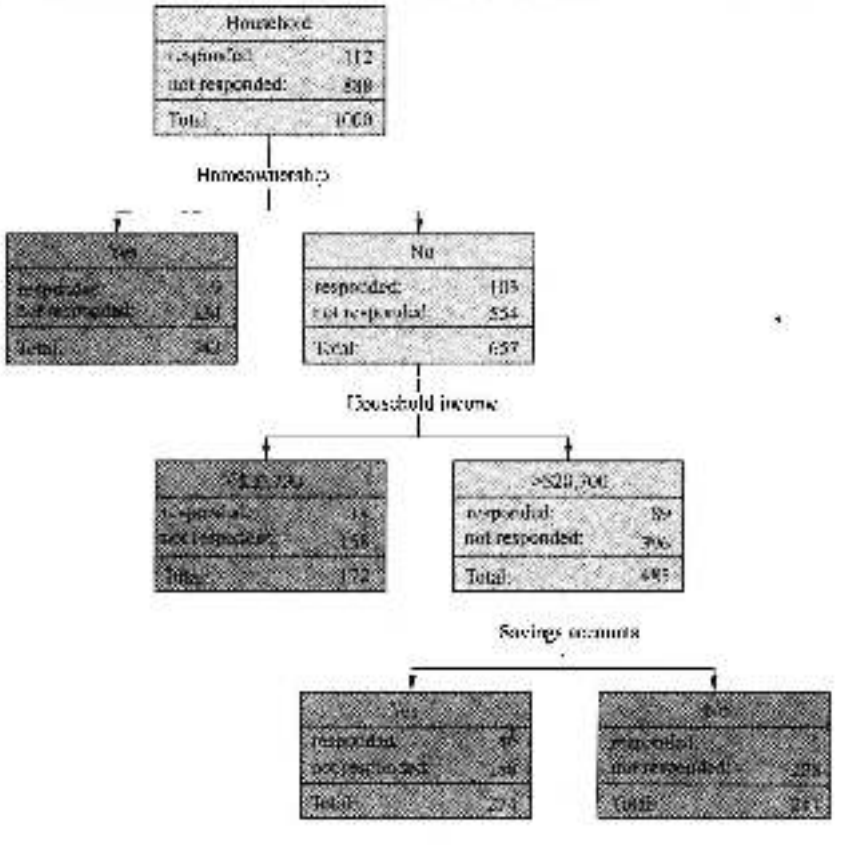


图 10.23　决策树的一个例子

表示家庭是否响应了之前的促销活动。

决策树由结点、分支和叶子组成。图 10.23 中，每一个方框代表一个结点。最顶部的结点称为根结点。决策树总是从根结点开始，然后通过在每一层将数据划分为新的结点来往下增长。根结点包含整个数据集（所有数据记录），子结点包含对应的数据子集。结点之间通过分支来连接。注意，分支最底端的结点称为终端结点，或者叶子。

每一个结点包含该结点所含的数据记录的数目和因变量的值的分布等信息。

什么是因变量

因变量是用户选择的，用来确定研究的目标。在我们的例子中，Household 为因变量，它的取值为 responded 或 not responded。

在根结点下是决策树的下一层。在这一层中，Homeownership 变量被选作因变量的预测因子并且所有的家庭都根据预测因子的取值来进行划分。数据的划分称为分割。实际上，Homeownership 仅仅是数据库中的一个字段。

决策树如何选择一个特定的分割

决策树的一个分割对应于一个具有最大划分能力的预测因子。换句话说，最佳分割将数据划分为一个类占主导的形式。

401
402

在我们的例子中，Homeownership 将那些响应了之前促销活动的家庭从不响应的家庭中最佳分割出来。在图 10.23 中，在所有 11.2% 的响应的家庭中，绝大部分不是房主。

有很多方法可以用来计算预测因子的数据划分能力。其中的一个著名的方法是基于度量不平等程度的基尼系数。

什么是基尼系数

基尼系数用来度量预测因子对父结点数据类别的划分程度。

意大利经济学家基尼提出了一个粗略度量一个国家收入分布的不平等程度的指标。基尼系数的计算如图 10.24 所示。对角线表示财富的绝对平等分布，其上的曲线代表了实际的经济状况。收入分配中总是会有一些不平等。曲线数据将社会人群从最富有到最贫穷排序。基尼系数的计算方法是用曲线和对角线之间的面积除以对角线以下的面积。当财富分配完全平等时，基尼系数取值为 0。当财富完全不平等分布，即由一个人拥有所有的社会财富时，基尼系数取值为 1。

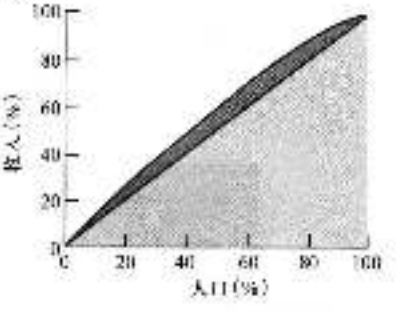


图 10.24　基尼系数的计算

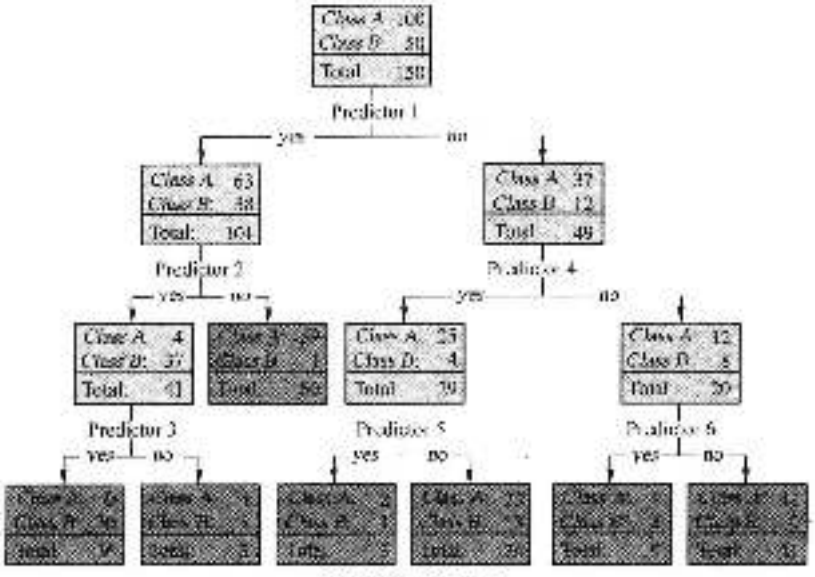
CART（分类和回归树）采用基尼系数来选择分割（Breiman, et al., 1984）。图 10.25 显示了两棵不同的树的对比。假设在根结点中，我们有两个类 A 和 B。决策树力求分离出最大的类，也即将类 A 的数据记录划分到一个单独的结点。这种理想的划分往往很难到达；在大多数情况下，数据库中不存在能够将一个类的数据跟其他类区分开的字段。因此我们需要在多种可能的分割中做选择。

403

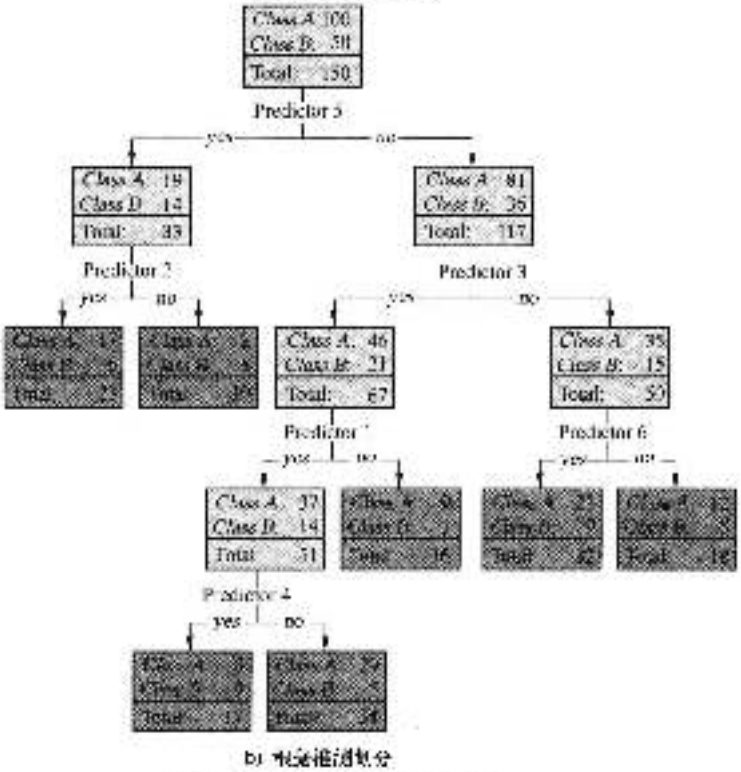
图 10.25a 中显示的决策树是根据基尼系数来选择的分割来自动增长的。在图 10.25b 中，我们根据自己的判断或者推测来选择分割。最后生成的树的比较如图 10.26 中的增益图（也叫做累积增益图）所示。增益图将一个结点类 A 在一个终止结点的累积样本百分比映射到同一个结点的总人口累积百分比。这里的对角线表示的是每一个终止结点包含随机人口样本时的结果。实验结果清楚地显示出用基尼系数分割所建立的树的优势。

如何从决策树中提取规则

从根结点到叶结点的一条路径揭示了一条决策规则。例如，在图 10.25a 所示的决策树中，右下边的叶结点对应的规则可以表示如下：



a) 使用标记系数分割



b) 唯完整测量分

图 10.25　选择一棵最优决策树

```
if      (Predictor 1 = no)
and     (Predictor 4 = no)
and     (Predictor 6 = no)
then    class = Class A
```

案例 12：数据挖掘的决策树

假设有社区健康服务调查的结果，而且想要弄清哪些人有患高血压的高风险。决策树可以用来解决这个问题吗？

404
~
405

决策树可以解决的一个典型任务就是找出可能导致某些输出的条件。这使得决策树成为分析高血压人群的好选择，而社区健康调查为我们提供了需要的数据。

高血压发生在人体的小血管变窄的时候，为了维持血压，心脏的负担加重，尽管人体承受血压升高的时间可以达到数月甚至数年，但最终心脏会出现问题。

血压可以分为最佳、正常和高3种。最佳血压低于120/80，正常血压在120/80～130/85之间，当血压高于140/90时就会被诊断为高血压。图10.27显示了高血压研究使用的一个数据集的例子。

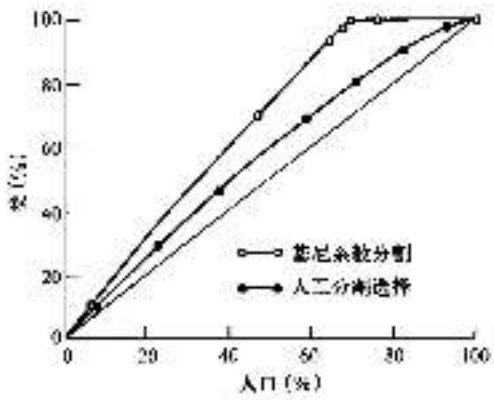


图10.26 类A的增益图

公众健康调查，高血压研究（美国加州）	
性别	☑男性 ☐女性
年龄	☐18~34 ☐35~50 ☑51~64 ☐65及以上
身体	☑白种人 ☐非裔美国人 ☐拉丁美洲 ☐亚洲
婚姻情况	☐已婚 ☐分居 ☑寡妇 ☐丧偶 ☐未婚
家庭收入	☐低于20 700美元 ☐20 701~45 000美元 ☑45 001~75 000美元 ☐75 001美元及以上
喝酒状况	☐不喝酒 ☐偶尔（每个月喝一点） ☑有规律的（每天一次或者两次） ☐喝酒（每天一次以上）
吸烟	☐不吸烟 ☐每天1~10根烟 ☑每天11~20根烟 ☐一天超过一包
咖啡消费取	☐不喝咖啡 ☑一天一杯或两杯 ☐一天三杯以上
饮食取	☐低盐食物 ☑中或度食物 ☐高或度食物
身体活动	☐不活动 ☑一周一次或两次 ☐一周一次以上
身高 体重	☐98 厘米 ☑91 公斤
血压	☐最佳 ☐正常 ☑高
肥胖	☑肥胖 ☐不肥胖

图10.27 一个高血压研究的数据集

决策树的好坏取决于数据。和神经网络和模糊系统不同，决策树不能容忍噪声和污染数据。因此在开始数据挖掘之前，数据必须被清洗。

几乎所有的数据库在某种程度上都是被污染的。在高血压研究中，我可能会发现一些字段（例如 Alcohol Consumption（喝酒状况）或 Smoking（吸烟））值为空或者包含不正确的信息。我们应该检查数据是否有不一致和拼写错误。然而，不管我们怎么努力都不太可能提前清除所有的污染，有些数据中的异常情况只有在数据挖掘过程中才能被发现。

我们也可能想丰富数据。例如，假如有 weight（体重）和 height（身高）变量，我们可以很容易地派生出一个新的变量 obesity。这个变量的计算使用体重系数（BMI）：以公斤为单位的体重除以以米为单位的身高的平方。BMI 等于或高于 27.8 的男士和 BMI 等于或高于 27.3 的女士被分类为“肥胖”的。

一旦高血压研究的数据准备好，我们可以选择一个决策树工具。在我们的研究中，我们使用 Angoss 公司的 KnowledgeSEEKER——一个全面的用于构建分类树的工具。

KnowledgeSEEKER 构建决策树是以创建一个对应于因变量 Blood pressure（血压）的根结点开始的。其次它将所有的应答者分为 3 个类别：optimal（最佳）、normal（正常）和 high（高）。在我们的研究中，319 人（32%）被分为 optimal 类，528 人（53%）被分为 normal 类，153 人（15%）被分为 high 类。

然后 KnowledgeSEEKER 确定每一个变量对 Blood pressure 的影响，并对最重要的变量生成一个排序列表。在我们的研究中，age 位于列表的顶端，KnowledgeSEEKER 按照年龄对应答者进行划分，从而产生了决策树的第二层，如图 10.28 所示。

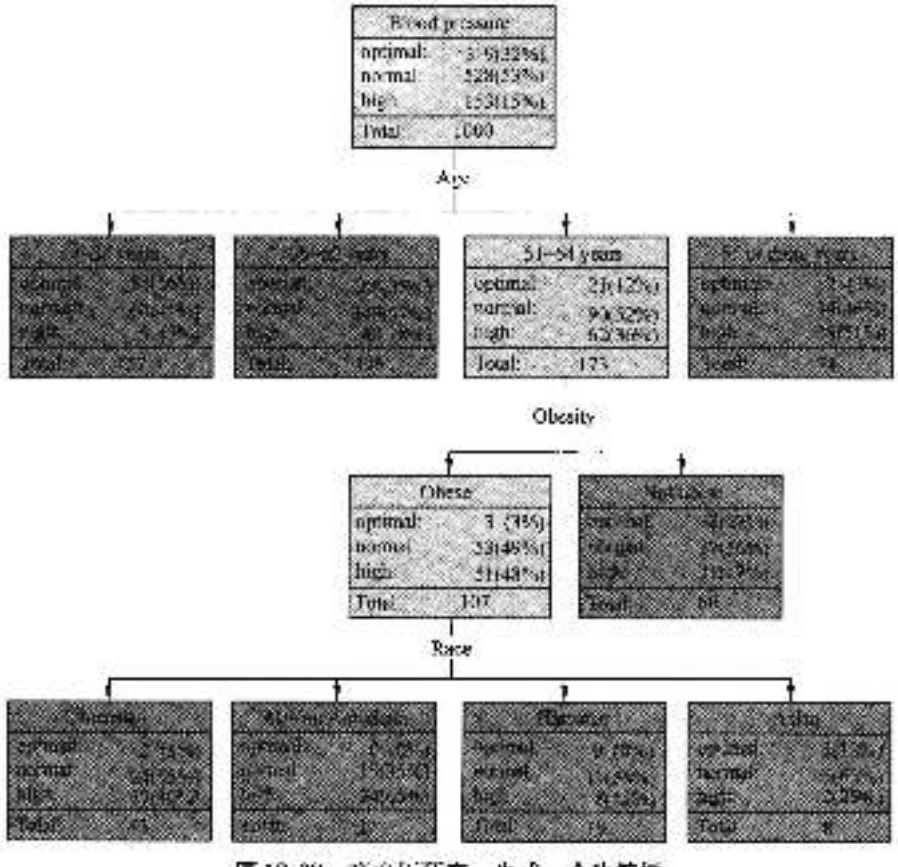


图 10.28　高血压研究：生成一个决策树

可以看出，高血压的风险随着年龄在增长。年龄超过50岁以后，高血压变得很普遍。

我们通过产生新的分割来生长树。例如，我们考虑对应于年龄在51～64岁之间的群体的这个第二层的结点。KnowledgeSEEKER按照Obesity来分割这个群体。这是因为在我们的例子中，Obesity是年龄在51～64岁的人是否患高血压的一个关键指示。如图10.28所示，在这个群体中48%的肥胖个体患有高血压。实际上，在老龄人群中，血压的增长主要是由体重的增长造成。

随着我们一个结点一个结点地生长树，我们会发现非裔美国人患高血压的风险比其他群体要高很多，而且吸烟和酗酒会更加增大患高血压的风险。

如何查看特定的分割

KnowledgeSEEKER等决策树工具允许我们查看任意的分割。图10.29显示的是按照Gender（性别）对35～50岁的人群和51～64岁的人群所作的分割。实验结果显示在51岁之前，男士患高血压的比例比女士高；而在51岁之后，结果刚好相反，女士比男士更有可能患高血压。对于数据挖掘，决策树方法的主要优点是它将解决方案可视化，而且可以跟踪树的每一条路径。决策树发现的关系可以表示为一系列规则，这些规则可以用来开发一个专家系统。

406
/
408

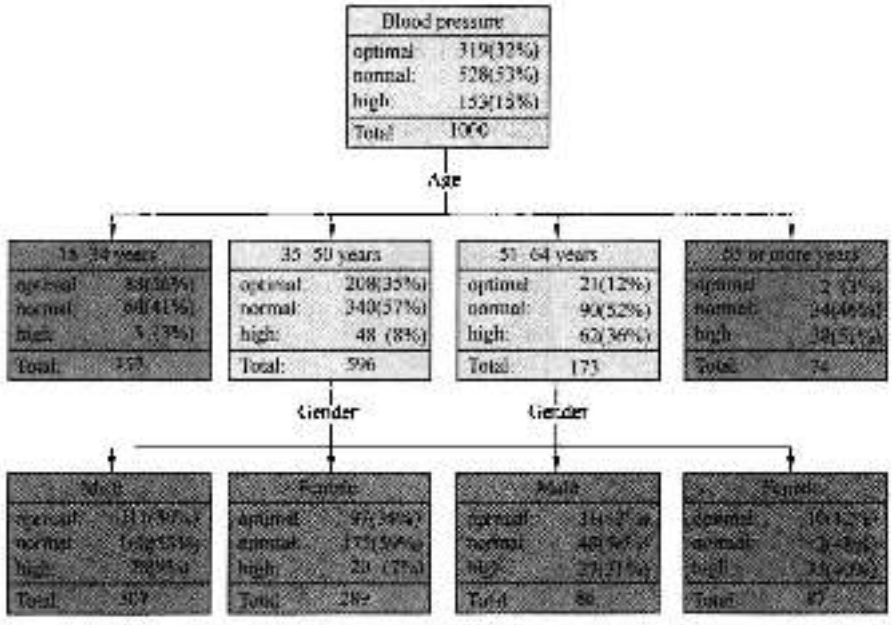


图10.29 高血压研究：关注一个分割

然而，决策树也有一些缺点。连续数据（例如年龄或收入）必须被划分为范围值，从而可能会无意地隐藏一些重要的模式。

另一个常见的问题是丢失数据和不一致数据的处理，决策树只有在处理"干净的"数据时才能产生可靠的输出。

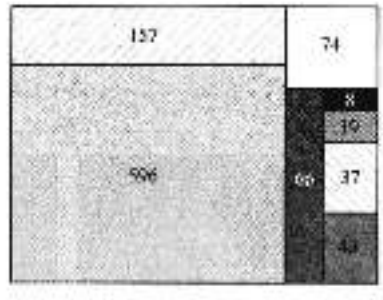


图10.30 高血压研究的解空间

然而，决策树最重要的限制来自它没有一次检查一个以上变量的能力。它将树局限在那些可以将解空间（solution space）划分为连续的矩形的问题上。图10.30所示说明了这一点。高血压研究的解空间首先按age划分为4个矩形，年龄在51～64岁的群体又被划分为超重的和不超重的。最后，肥胖人群按照race来划分。这样的"矩形"划分可能与实际数据的分布不能很好地吻合。在树很大的时候会导致数据碎

片，而且当从根结点传到底端结点的数据量很小的时候会使发现有意义的模式和规则变得很困难。为了最小化数据碎片，通常需要对最低层的结点和叶子进行修剪。 409

尽管有这些限制，决策树已经成为了数据挖掘中最成功的技术之一。产生清晰的规则集的能力使得决策树对业务专家尤其具有吸引力。

与决策树类似，我们可以使用关联规则挖掘以在大型数据库中发现变量之间隐含的关系。

10.7　关联规则和购物篮分析

关联规则表达大型数据库中对象的并发关系（称为关联）。它们陈述了不同的对象如何趋向于组合到一起。关联规则的概念最早在 20 世纪 90 年代初期提出（Agrawal et al.，1993），最初被用在购物篮分析中。

什么购物篮分析

假设我们观察超市中的一位顾客。该顾客有一个篮子装满了各种产品，例如苹果、香蕉、橘子汁、清洁剂、牛奶等。一个篮子只跟一个用户相关，它不能揭示产品之间有用的关联。然而，很多的篮子（确切地说是很多顾客购买的产品信息）确实可能告诉我们哪些产品经常被一起购买。例如，食品店的管理员会发现意大利面和意大利面酱经常被一起购买。这些信息代表了联系共现事件的一条关联规则。购物篮分析回答“如果一位顾客购买了产品 A，他或她有多大可能性也会同时购买产品 B?”和“如果一位顾客同时购买了产品 A 和产品 B，他或她最可能还会买什么其他产品?”这样的问题。

购物篮分析可以被食品店管理员用于地板放置、存货控制、市场营销和广告活动等。如果你使用 amazon. com，你很可能已经对购物篮分析和关联规则很熟悉了。当你购买一本书（例如 Michael Negnevitsky 的《人工智能：智能系统指南》）的时候，amazon. com 会向你推荐一些跟这本书经常一起购买的产品。这些推荐基于一条简单的关联规则：“购买了产品 A 的顾客很可能也会购买产品 B。”

购物篮分析如何工作

购物篮分析总是从事务开始。每一个事务包含一次购买的一个或一组产品。注意，我们通常不关注产品的数量和价格。表 10.4 代表了 7 个事务的集合。从表我们可以看出，面酱和意大利面很可能会被一起购买，因此我们可以写出下面一条规则：“如果一位顾客购买了面酱，那么这位顾客也会购买意大利面。”这条规则可以重写为更一般的形式：

$\{Sauce\} \rightarrow \{Spaghetti\}[support = 57\%, confidence = 80\%]$ 410

表 10.4　食品交易

事务	项
1	Beef，Milk，Sauce，Spaghetti
2	Sauce，Spaghetti
3	Bread，Sauce，Spaghetti
4	Cheese，Detergent，Sauce
5	Bread，Detergent，Milk
6	Beef，Cheese，Chicken，Milk
7	Milk，Sauce，Spaghetti

这条规则说明有 57% 的顾客同时购买了面酱和意大利面，80% 购买了面酱的顾客也会同时购买意大利面。支持度和置信度用于选择有效的关联规则。

如何定义一条关联规则的支持度和置信度

我们首先用数学术语来定义关联规则挖掘问题。假设 $I = \{I_1, I_2, \cdots, I_m\}$ 为项的集合，$T =$

$\{T_1, T_2, \cdots, T_n\}$ 为事务的集合，其中 $T_i \subseteq I$。则一条关联规则可以写成：

$$X \rightarrow Y$$

其中，X 和 Y 为项的集合，称为项集，$X \subset I$，$Y \subset I$ 且 $X \cap Y = \phi$。

我们称事务 T_i 包含项集 X 如果 X 是 T_i 的一个子集，即 $X \subset T_i$。例如，事务 {Bread, Sauce, Spaghetti} 包含7个项集：{Bread}、{Sauce}、{Spaghetti}、{Bread, Sauce}、{Bread, Spaghetti}、{Sauce, Spaghetti} 和 {Bread, Sauce, Spaghetti} 本身。

关联规则 $X \rightarrow Y$ 的支持度（support）定义为 X 和 Y 两个项集的并在事务集 $\boldsymbol{T}$ 中出现的概率：

$$support\ (X \rightarrow Y) = p_{X \cup Y} = \frac{n_{X \cup Y}}{n} \tag{10.18}$$

其中，$n_{X \cup Y}$ 为事务集 $\boldsymbol{T}$ 中包含 X 和 Y 两个项集的并的事务数目；n 是事务集 $\boldsymbol{T}$ 中的总事务数目。

如果支持度很低，规则只会偶尔出现。在我们的例子中，Sauce 和 Spaghetti 在7个事务中共同出现了4次，所以这条规则的支持度是57%。

411 关联规则 $X \rightarrow Y$ 的置信度（confidence）定义为给定 X 出现在某一个事务后，Y 也出现在同一个事务中的条件概率：

$$confidence\ (X \rightarrow Y) = p\ (Y \mid X) = \frac{n_{X \cup Y}}{n_X} \tag{10.19}$$

其中，n_X 表示为事务集 $\boldsymbol{T}$ 中包含项集 X 的事务数目。

置信度是规则准确率的一个度量。在我们的例子中，在5位购买了 Sauce 的顾客中，有4位也同时购买了 Spaghetti。因此，这条规则的置信度是80%。通常我们会选择那些具有更高置信度和更低支持度的关联规则。

关联规则挖掘的目标是发现那些支持度和置信度分别高于或等于用户指定的最小支持度和最小置信度的关联规则。例如，如果我们指定最小支持度 = 25% 且最小置信度 = 75%，我们将会得到下列有效的规则：

Rule 1: {*Sauce*} → {*Spaghetti*} [*support* = 57%, *confidence* = 80%]
Rule 2: {*Spaghetti*} → {*Sauce*} [*support* = 57%, *confidence* = 100%]
Rule 3: {*Beef*} → {*Milk*} [*support* = 29%, *confidence* = 100%]

显然，我们可以生成更多的关联规则。通常情况下，规则数随着项数指数增长。假设项数为 m，我们可以计算所有可能的关联规则数 R（Tan et al., 2006）：

$$R = 3^m - 2^{m+1} + 1 \tag{10.20}$$

因此，仅仅我们的例子中所使用的8个项就能生成6050条规则。考虑到典型的超市的库存中一般有大约20000种不同的项，挖掘关联规则将会出现很大的问题。实际上，我们不需要生成一个完整的规则集，因为大多数规则具有很低的支持度和置信度。我们需要的是一个能够将关联规则数目降低到一个可操作水平的鲁棒算法。Apriori 算法是这类算法的一个很好的例子（Berry and Linoff, 2004; Tan et al., 2006）。它已经被应用在大多数关联规则挖掘的商业产品中。

什么是 Apriori 算法

Apriori 算法是最先用来解决关联规则数目指数级增长问题的算法之一。然而，在讨论 Apriori
412 算法之前，我们首先需要考虑频繁（也称为大）项集的概念。

一个有 m 个项的数据集可以生成 $2^m - 1$ 个项集。在大多数实践应用中，m 往往很大，因而项集的搜索空间会变得不可操作。所以，我们需要一个修剪搜索空间的策略。这个策略基于频繁项集的概念。一个项集被称为是频繁的，如果它在事务集中出现的次数超过了一个指定的最小支持度。如果一个项集是频繁的，那么它的所有子集必然也是频繁的。这个性质称为 Apriori 性质。相反，如果一个项集是不频繁的，那么它的所有超集必然都是不频繁的。

为了说明频繁项集的概念，我们来考虑图10.31所示的栅格。这里，四层的栅格是从项集

{A,B,C,D} 构建（空项集不包含在内）。频繁 1－项集位于栅格的最上层，最大的频繁－4 项集位于栅格的最底层。图中的线表示项集－子项集关系。如果一个项集是频繁的，那么在任意包含该项集的路径上，位于该项集上方的所有子项集都是频繁的。例如，如果项集 {A,B,C} 是频繁的，那么它的所有子集 {A,B}、{A,C}、{B,C}、{A}、{B} 和 {C} 必然也都是频繁的。这称为支持度的单调性（或向上封闭性）。

然而，如果一个项集是不频繁的，那么在任意包含该项集的路径上，位于该项集下方的所有超集都是不频繁的。这称为支持度的反单调性（或向下封闭性）。如图 10.31 所示，一旦某个项集被发现是不频繁的，那么该项集下的整个子图都可以被剪枝。这被称为基于支持度的剪枝。Apriori 算法利用基于支持度的剪枝来控制项集的搜索空间的增长。

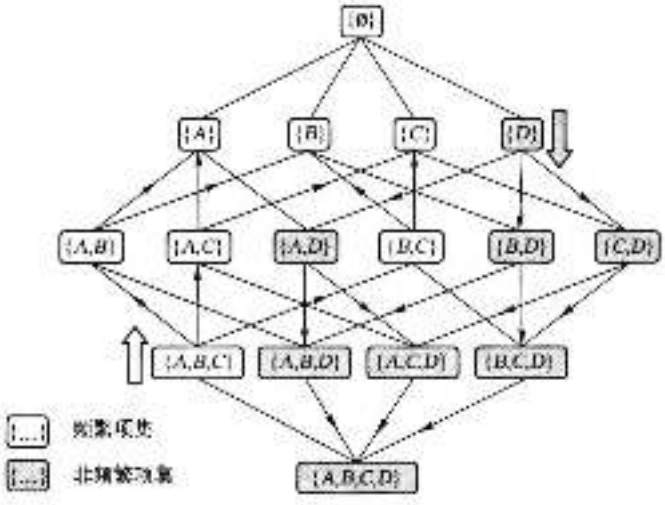


图 10.31　项集 {A,B,C,D} 的栅格

关联规则挖掘分为两个步骤。在第一步中，Apriori 算法找出所有的频繁项集；在第二步中，它从第一步找出的频繁项集中生成高置信度的规则。 413

为了演示 Apriori 算法的运行机制，我们参考表 10.4，计算所有非空项集的支持度。结果如表 10.5 所示。所有子集中的项以字母顺序（也称为字典顺序）保存。

表 10.5　表 10.4 中包含的所有非空项集的支持度

项集	支持度	项集	支持度
{Beef}	29%	{Chicken, Milk}	14%
{Bread}	29%	{Detergent, Milk}	14%
{Cheese}	29%	{Detergent, Sauce}	14%
{Chicken}	14%	{Milk, Sauce}	29%
{Detergent}	29%	{Milk, Spaghetti}	29%
{Milk}	57%	{Sauce, Spaghetti}	57%
{Sauce}	71%		
{Spaghetti}	57%	{Beef, Cheese, Chicken}	14%
		{Beef, Cheese, Milk}	14%
{Beef, Cheese}	14%	{Beef, Chicken, Milk}	14%
{Beef, Chicken}	14%	{Beef, Milk, Sauce}	14%
{Beef, Milk}	29%	{Beef, Milk, Spaghetti}	14%

（续）

项集	支持度	项集	支持度
{Beef, Sauce}	14%	{Beef, Sauce, Spaghetti}	14%
{Beef, Spaghetti}	14%	{Bread, Detergent, Milk}	14%
{Bread, Detergent}	14%	{Bread, Sauce, Spaghetti}	14%
{Bread, Milk}	14%	{Cheese, Chicken, Milk}	14%
{Bread, Sauce}	14%	{Cheese, Detergent, Sauce}	14%
{Bread, Spaghetti}	14%	{Milk, Sauce, Spaghetti}	29%
{Cheese, Chicken}	14%		
{Cheese, Detergent}	14%	{Beef, Cheese, Chicken, Milk}	14%
{Cheese, Milk}	14%	{Beef, Milk, Sauce, Spaghetti}	14%
{Cheese, Sauce}	14%		

生成频繁项集需要多次遍历事务数据集 T。在第一次遍历中，Apriori 算法产生所有的频繁 1 - 项集：{Beef}、{Bread}、{Cheese}、{Chicken}、{Detergent}、{Milk}、{Sauce} 和 {Spaghetti}。对所有的频繁 1 - 项集进行支持度计数后，项集 {Chicken} 被丢弃，因为它的支持度低于指定的 25% 的最小支持度。

在第二次遍历中，Apriori 算法利用种子集（前一步发现的频繁项集的集合）来生成候选频繁 2 - 项集。每一个候选项集通过连接种子集中的两个项集产生。在我们的例子中，生成了下列候选频繁 2 - 项集：

{*Beef, Bread*}, {*Beef, Cheese*}, {*Beef, Detergent*}, {*Beef, Milk*},
{*Beef, Sauce*}, {*Beef, Spaghetti*}, {*Bread, Cheese*}, {*Bread, Detergent*},
{*Bread, Milk*}, {*Bread, Sauce*}, {*Bread, Spaghetti*}, {*Cheese, Detergent*},
{*Cheese, Milk*}, {*Cheese, Sauce*}, {*Cheese, Spaghetti*},
{*Detergent, Milk*}, {*Detergent, Sauce*}, {*Detergent, Spaghetti*},
414
{*Milk, Sauce*}, {*Milk, Spaghetti*}, {*Sauce, Spaghetti*}

我们一共得到 21 条候选项集；然而，其中只有 4 条项集是频繁的：

{*Beef, Milk*}, {*Milk, Sauce*}, {*Milk, Spaghetti*}, {*Sauce, Spaghetti*}

在第三步和所有后续步骤中，Apriori 算法连接两个项集，如果这两个项集只有一个项不同。例如，{*Beef*，*Milk*} 会和其他包含 *Beef* 或 *Milk* 的频繁 2 - 项集进行连接。结果，我们将会得到下列候选频繁 3 - 项集：

{*Beef, Milk, Sauce*}, {*Beef, Milk, Spaghetti*}, {*Milk, Sauce, Spaghetti*}

并不是所有上述候选频繁 3 - 项集都是频繁的。实际上，一个候选项集只有当它的所有子集出现在之前步骤的种子集中时才是频繁的。因为上述第一个项集的子集 {*Beef*，*Sauce*} 和第二个项集的子集 {*Beef*，*Spaghetti*} 都不在种子集中，这两个候选项集都会被丢弃。因此，在第三个步之后，只剩下一个频繁 3 - 项集：

{*Milk, Sauce, Spaghetti*}

注意，对事务数据的遍历次数等于最大项集的大小。在实践中，我们将这个大小限制在一些少数项上，因为很长的关联规则很难被理解和使用。

Apriori 算法如何生成关联规则

找出所有的频繁项集以后，关联规则的生成就很直接了。

任何一个至少包含两个项的项集 Y 都可以划分成两个非空的子集 X 和 $(Y-X)$。如果项集 X 和子集 $(Y-X)$ 的支持度的比率大于或等于用户指定的最小置信度，那么就可以产生一条规则 X $(X \subset Y)$：

$$\text{IF } \frac{support\ (Y)}{support\ (X)} \geqslant minimum\ confidence, \text{ THEN } X \rightarrow (Y \rightarrow X) \tag{10.21}$$

注意，因为项集 Y 和 X 的支持度在频繁项集生成阶段已经被发现，因此不需要再一次遍历事务数据集。

为了说明关联规则是怎样生成的，我们考虑一个频繁 2 - 项集 $Y = \{Beef, Milk\}$，假设最小置信度是 75%。频繁项集 Y 包含两个非空的子集：X_1 $\{Beef\}$ 和 X_2 $\{Milk\}$。对于子集 X_1，我们得到

$$\frac{support(Y)}{support(X1)} = \frac{support(\{Beef, Milk\})}{support(\{Milk\})} = \frac{29}{29} = 1$$

415

这表示关联规则

$$X1 \rightarrow (Y - X1) \text{ 或者 } \{Beef\} \rightarrow \{Milk\}$$

的置信度是 100%，与我们采用公式（10.19）计算的结果一样。类似地，对于第二个子集 X_2，有

$$\frac{support(Y)}{support(X2)} = \frac{support(\{Beef, Milk\})}{support(\{Milk\})} = \frac{29}{57} = 0.5$$

这表示关联规则

$$X2 \rightarrow (Y - X2) \text{ or } \{Milk\} \rightarrow \{Beef\}$$

只有 50%，因此它不是一个有效（高置信度）的规则，需要被丢弃。

一个完整的关联规则集合如表 10.6 所示。让我们来检查这些规则。我们发现，跟支持度不同，置信度并不满足单调性；也就是说，规则 $X \rightarrow Y$ 的置信度可能大于、等于和小于另一个规则 $x \rightarrow y$ 的置信度，其中 $x \subseteq X$ 且 $y \subseteq Y$。例如，{Milk, Sauce} → {Spaghetti} 是一条高置信度的规则，但是 {Milk} → {Spaghetti} 置信度却很低；{Sauce} → {Milk, Spaghetti} 是一条低置信度规则，但是 {Sauce} → {Spaghetti} 置信度却很高。

416

表 10.6　从食品交易事务中生成的关联规则

大的项集	关联规则	置信度
{Beef, Milk}	{Beef} → {Milk}	100%
	{Milk} → {Beef}	50%
{Milk, Sauce}	{Milk} → {Sauce}	50%
	{Sauce} → {Milk}	40%
{Milk, Spaghetti}	{Milk} → {Spaghetti}	50%
	{Spaghetti} → {Milk}	50%
{Sauce, Spaghetti}	{Sauce} → {Spaghetti}	80%
	{Spaghetti} → {Sauce}	100%
{Milk, Sauce, Spaghetti}	{Milk, Sauce} → {Spaghetti}	100%
	{Milk, Spaghetti} → {Sauce}	100%
	{Sauce, Spaghetti} → {Milk}	50%
	{Milk} → {Sauce, Spaghetti}	50%
	{Sauce} → {Milk, Spaghetti}	40%
	{Spaghetti} → {Milk, Sauce}	50%

另一方面，从同一频繁集中生成的规则确实满足置信度的反单调性；也就是说，如果规则 $X \rightarrow (Y - X)$ 不满足置信度的要求，那么任意规则 $x \rightarrow (Y - x)$，其中 x 是 X 的子集，也不满足置信度要求。例如，如果规则 {Sauce, Spaghetti} → {Milk} 置信度很低，那么规则 {Sauce} → {Milk} 和 {Spaghetti} → {Milk} 的置信度也很低。实际上，这也是从同一频繁项集中生成的任意关联规则集所满足的一般规律（Tan et al., 2006）。将规则的置信度表示为

$$confidence\ (X \rightarrow \{Y - X\}) = \frac{support\ (Y)}{support\ (X)}$$

和

$$confidence\ (x \rightarrow |Y-x|) = \frac{support\ (Y)}{support\ (x)}$$

因为 x 是 X 的子集，则 $support\ (x) \geq support\ (X)$，因此有

$$confidence\ (x \rightarrow |Y-x|) \leq confidence\ (X \rightarrow |Y-X|)$$

一般情况下，每一个 k - 项集可以生成 $2^k - 2$ 条关联规则。例如，一个 4 - 项集 {A, B, C, D} 可以产生 14 条关联规则。这些规则可以表示成图 10.32 所示的栅格。可以看出，栅格有 5 个 (k + 1) 层。第 0 层只有一条规则，规则的前项是一个 4 - 项集，后项为空。第 1 层包含 1 - 项集后项的规则，第 2 层是 2 - 项集后项，第 3 层是 3 - 项集后项。最后一层，也即第 4 层，只包含一条单独的前项为空，后项为 4 - 项集的规则。

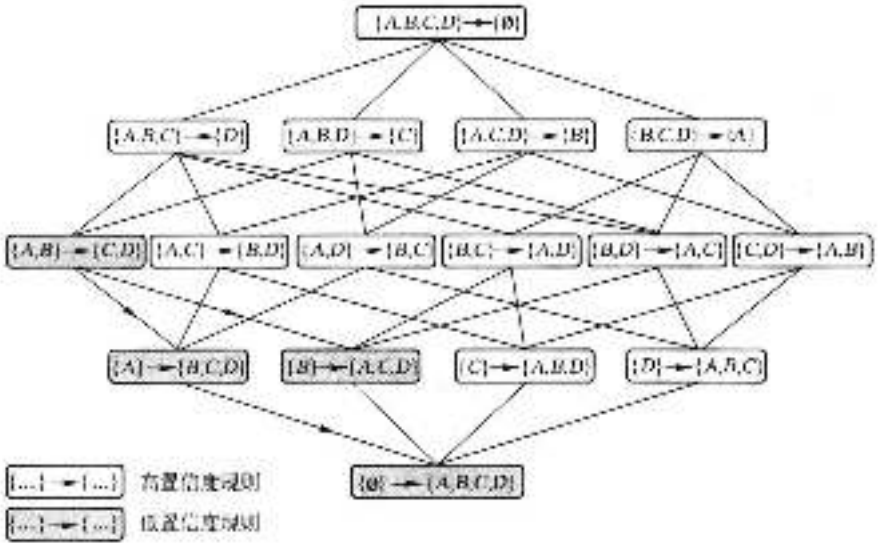


图 10.32 由项集 {A, B, C, D} 生成的关联规则栅格

根据置信度的反单调性质，如果栅格中的一条规则置信度很低，那么我们就可以删除由这条规则生成的整个子图。例如，如果我们发现规则 {A, B} → {C, D} 置信度很低，所有后件中包含 {C, D} 的规则置信度都很低，可以被丢弃，如图 10.32 所示。

Apriori 算法首先生成后件中包含 1 - 项集的规则。然后计算每一条规则的置信度，置信度低的规则被删除。高置信度的规则被用来产生新的候选规则。这种生成 - 测试过程是很直接和高效的。

最后需要注意一点，购物篮分析的主要困难是我们会发现很多实际价值很低的规则。因此，最重要的是要选择正确的项集。

超市中的所有项遵循分类学，根据产品代码形成的层次分类。分类学总是从最一般的产品开始，然后沿着详细层次上升。因此，通过使用更高层次的分类，我们可以显著地减少频繁项集的数目，如用“冷冻蔬菜”来代替“冷冻豌豆”和“冷冻胡萝卜”等。同时，一般化的项集拥有在常规事务中一样多的支持度。因此，什么是恰当的一般化层次？这依赖于项产生可行性结果的重要程度。一个常用的方法是，除了我们感兴趣的项以外，对其他所有项都进行一般化处理。

购物篮分析是一种有用的间接数据挖掘的技术，特别是在零售业中。它的吸引力来自其清晰的结果表示方式：关联规则。如今，关联规则还被用于 Web 使用挖掘等很多其他领域（Liu, 2007; Markov and Larose, 2007）。

10.8　小结

本章我们对数据挖掘作了一个概述，以及讨论了将数据转换为知识的主要技术。首先，我们笼统地定义了数据挖掘并解释了在大型数据库中进行数据挖掘和知识发现的过程。我们介绍了一些统计学方法，包括主成分分析，同时讨论了它们的限制。然后我们展示了 SQL 语言在关系数据库中的应用，以及介绍了数据仓库和多维数据分析。最后，我们讨论了最流行的数据挖掘工具——决策树和购物篮分析。

本章的主要内容有：

- 现代社会是基于信息的。大多数信息以事实、观察和测量等原始形式存在，构成了我们收集和存储的数据。随着计算能力代价持续下降，累积的数据量在以指数级增长。传统数据库并不是为了执行有意义的分析而设计，而这些正是数据挖掘来执行的。 417 ~ 418
- 数据挖掘就是从原始数据进行知识提取。它还可以被定义为对大规模数据的探索和分析，以发现有意义的模式和规则。数据挖掘的终极目标是发现知识。
- 数据挖掘是一个实践话题，经常被视为“什么都包括”的东西，包括从统计方法和查询工具到复杂的机器学习等技术。数据挖掘不是一个单一的方法，而是不同工具和技术的异构组合。这些工具和技术包括：统计方法和数据可视化工具、查询工具、联机分析处理、决策树、关联规则、神经网络和神经 - 模糊系统。
- 虽然数据挖掘仍然是一个很新的、不断发展的领域，它已经拥有很多不同的应用。在直销领域，数据挖掘被用来确定最有可能购买某些产品和服务的人。在趋势分析中，数据挖掘被用来确定市场中的趋势，例如通过对证券市场建模。在欺诈检测中，数据挖掘被用来识别保险理赔、移动电话和信用卡交易中最有可能的欺诈行为。
- 数据挖掘的成功经常依赖于数据挖掘工具的选择。对数据的初步调查可以帮助我们理解数据的特定特征，从而选择合适的数据分析技术。这个初步的过程称为数据探索，包括汇总统计、可视化、主成分分析、查询工具和 OLAP。
- 数据可视化可以定义为一种以图形或表格形式表示信息的方法。信息被可视化以后，我们就可以解释和理解它了。可视化在数据探索的初始阶段是极为重要的。图形化的数据表示方法包括点图、茎叶图、直方图、箱形图和散布图。
- 虽然数据可视化具有很明显的吸引力，高维数据上的图形挖掘却不能够很轻易地完成。一种处理多维数据的方法是降维。
- 主成分分析能在不明显丢失信息的情况下降低数据维度。PCA 是这一种将大量相关变量转换为少量不相关变量的数学过程。在数据挖掘中，PCA 常常被用作一种数据预处理技术。它让我们得到一个远远小于原始数据集的新数据集，同时产生一样（或者几乎一样）的分析结果。
- PCA 等统计方法往往作为数据探索的第一步，数据库查询和 OLAP 则代表了一些为了处理现代组织积累的大量数据而特别开发的复杂工具。 419
- 数据库查询可以看做是数据挖掘的基本形式。查询确实为用户提供了向关系数据库提问和获取满足特定条件的数据子集的机会。然而，数据库查询是基于假设的——用户必须提出正确的问题。
- OLAP 使我们可以对存储在数据仓库中的数据进行分析。OLAP 系统通过运行一个数据立方体来支持复杂的多维查询。用户可以增长或降低详细层次，并且可以从多种不同的视角来观察数据。OLAP 是一种需要用户主动参与的用户直接（user-direct）的数据分析工具。

- 数据挖掘最流行的工具之一是决策树，一个可以使用一种类似树的结构来描述数据集的工具。决策树特别适合解决分类问题——追踪树中的任意路径是很容易的。产生清晰的规则集的能力使得决策树对业务专家尤其具有吸引力。
- 另一种数据挖掘的流行工具是购物篮分析。它能回答“如果一位顾客购买了产品 A，他或她有多大可能性也会同时购买产品 B?”和“如果一位顾客同时购买了产品 A 和产品 B，他或她最可能还会买什么其他产品?”这样的问题。购物篮分析的吸引力来自它使用关联规则这种清晰的结果表示方式。购物篮分析经常被用作一种间接数据挖掘的技术，特别是在零售业中。

复习题

1. 什么是数据挖掘？描述数据挖掘和知识发现的总过程。举例说明数据挖掘如何应用于实践？什么是数据挖掘工具？
2. 什么是数据探索？给出数据可视化的定义。举一些日常生活中图形数据表示方法的例子。
3. 什么是散布图？如何定义线性回归模型和确定回归因子？解释离群值是如何影响线性回归模型的。什么是鲁棒回归？
4. 什么是主成分分析？举例说明 PCA 是如何工作的？为了捕获给定数据集中包含的大部分信息，如何决定需要保留的主成分数目？
5. 什么是数据库管理系统？什么是关系数据库？定义关系数据库 Travel_agency 中的表。
6. 什么是查询？给出一些基本 SQL 查询的例子。数据库查询和数据挖掘的区别是什么？
7. 什么是数据仓库？数据仓库的主要特征是什么？为什么我们需要将数据从操作型数据库管理系统复制到数据仓库中？ 420
8. 什么是数据立方体和星网模型？举一个三维数据立方体的例子并将其转换为表格形式。
9. 什么是联机分析处理？OLAP 如何实现不同角度的数据视图？OLAP 支持一些什么操作？
10. 什么是决策树？什么是因变量和预测因子？什么是基尼系数？决策树是如何选择预测因子的？
11. 数据挖掘中，决策树方法的优点和缺点是什么？为什么决策树对业务专家特别具有吸引力？
12. 什么是购物篮分析？什么是关联规则？举一些关联规则的例子。给出支持度和置信度的定义。
13. 什么是频繁项集？为项集 {A, B, C, D, E} 构造一个项集栅格，并解释支持度的单调性和反单调性概念。
14. 考虑表 10.7 所示的 5 个事务。利用 Apriori 算法找出频繁项集并生成关联规则。假设最小支持度是 60%，最小置信度是 80%。

表 10.7　一个购物篮事务的例子

事务	项
1	Bread, Cheese, Orange Juice, Sugar
2	Bread, Milk
3	Beef, Cheese, Milk, Orange Juice
4	Bread, Cheese, Milk, Orange Juice
5	Beef, Bread, Cheese, Milk

参考文献

Abramowicz, W. and Zurada, J. (2001). *Knowledge Discovery for Business Information Systems*. Kluwer Academic Publishers, Norwell, MA.

Agrawal, R., Imielinski, T. and Swami, A. (1993). Mining association rules between sets of items in large databases, *Proceedings of the 1993 International Conference on Management of Data (SIGMOD 93)*, pp. 207–216.

Berry, M. and Linoff, G. (2004). *Data Mining Techniques for Marketing, Sales, and Customer Relationship Management*, 2nd edn. John Wiley, New York.

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984). *Classification and Regression Trees*, Wadsworth, Belmont, CA.

Bulos, D. and Forsman, S. (2006). *Olap Database Design: Delivering on the Promise of the Data Warehouse*, Elsevier, Amsterdam.

Celko, J. (2006). *Joe Celko's Analytics and OLAP in SQL*, The Morgan Kaufmann Series in Data Management Systems, Elsevier, San Francisco.

Chamberlin, D.D. and Boyce, R.F. (1974). SEQUEL: A Structured English Query Language, *Proceedings of the 1974 ACM SIGFIDET Workshop on Data Description, Access and Control*, Ann Arbor, MI, pp. 249–264.

421

Codd, E.F. (1970). A relational model of data for large shared data banks, Communications of the ACM, 13(6), 377–387.

Codd, E.F., Codd, S.B. and Salley, C.T. (1993). Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate. Technical Report, E.F. Codd and Associates.

Daniel, C. and Wood, F.S. (1999). *Fitting Equations to Data: Computer Analysis of Multifactor Data*, 2nd edn. John Wiley, New York.

DuBois, P. (2008). *MySQL: Developer's Library*, 4th edn. Addison-Wesley Professional, Reading, MA.

Elmasri, R. and Navathe, S.B. (2010). *Fundamentals of Database Systems*, 6th edn. Pearson Education, Boston, MA.

Forta, B. (2005). *MySQL: Crash Course*. Sams Publishing, Indianapolis.

Galton, F. (1877). Typical laws of heredity, *Nature*, 15.

Gantz, J. and Reinsel, D. (2009). As the economy contracts, the digital universe expands, *An IDC White Paper* – sponsored by EMC Corporation, IDC, May.

Gray, J., Bosworth, A., Layman, A. and Pirahesh, H. (1996). Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub-total, *Proceedings of the 12th International Conference on Data Engineering (ICDE)*, New Orleans, pp. 152–159.

Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F. and Pirahesh, H. (1998). Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub-totals, *Readings in Database Systems*, 3rd edn. Morgan Kaufmann, San Francisco, pp. 555–567.

Han, J. and Kamber, M. (2006). *Data Mining: Concepts and Techniques*, 2nd edn. Elsevier, Amsterdam.

Hand, D., Mannila, H. and Smyth, P. (2001). *Principles of Data Mining*. MIT Press, Cambridge, MA.

Hobbs, L., Hillson, S., Lawande, S. and Smith, P. (2005). *Oracle Database 10g Data Warehousing*. Elsevier, Amsterdam.

Inmon, W.H. (1992). *Building the Data Warehouse*. John Wiley, New York.

Jolliffe, I.T. (2002). *Principal Component Analysis*, 2nd edn. Springer-Verlag, New York.

Kimball, R. and Ross, M. (2002). *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*, 2nd edn. John Wiley, New York.

Liu, B. (2007). *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Springer-Verlag, Berlin.

Markov, Z. and Larose, D.T. (2007). *Data Mining the Web: Uncovering Patterns in Web Content, Structure, and Usage*. John Wiley, Hoboken, NJ.

422

Montgomery, D.C., Runger, G.C. and Hubele, N.F. (2001). *Engineering Statistics*, 2nd edn. John Wiley, New York.

Oja, E. (1982). A simplified neural model as a principal component analyzer, *Journal of Mathematical Biology*, 15, 239–245.

Oja, E. (1983). *Subspace Methods of Pattern Recognition*. Research Studies Press, Letchworth, Herts.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space, *Philosophical Magazine*, 2, 559–572.

Purdie, N., Lucas, E.A. and Talley, M. (1992). Direct measure of total cholesterol and its distribution among major serum lipoproteins, *Clinical Chemistry*, 38(9),1645–1646.

Rousseeuw, R.J. and Leroy, A.M. (2003). *Robust Regression and Outlier Detection*. John Wiley, Hoboken, NJ.

Seber, G.A.F. and Wild, C.J. (1989). *Nonlinear Regression*. John Wiley, New York.

Tan, P.N., Steinbach, M. and Kumar, V. (2006). *Introduction to Data Mining*. Addison-Wesley, Boston, MA.

Utts, J.M. (2004). *Seeing Through Statistics*, 3rd edn. Duxbury Press, Pacific Grove, CA.

Wilkinson, J.H. (1988). *The Algebraic Eigenvalue Problem*. Oxford University Press, New York.

Witten, I.H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edn. Elsevier, Amsterdam.

423

术 语 表

本术语表使用如下缩写形式：

es = expert systems（专家系统）

fl = fuzzy logic（模糊逻辑）

nn = neural networks（神经网络）

ec = evolutionary computation（进化计算）

dm = data mining（数据挖掘）

ke = knowledge engineering（知识工程）

Action potential（动作电位） 生物神经元通过长距离的传递也不会减弱强度的输出信号，也称作神经脉冲。当出现动作电位时，神经元就“发出一个脉冲”。[nn]

Activation function（激活函数） 将神经元的净输入映射到输出的数学函数。通常使用的激活函数有：阶跃、信号、线性和S形函数。参见Transfer function（传递函数）。[nn]

Adaptive learning rate（自适应学习速率） 根据训练中的误差的变化而调节的学习速度。如果当前周期误差大于上一周期值的预先指定倍数，学习速度将会降低。但是如果小于前一次的值，学习速度则会增加。使用自适应学习速度使多层感知器的学习速率加快。[nn]

Aggregate set（聚合集） 通过聚合获得的模糊集。[fl]

Aggregation（聚合） 模糊推理的第三步；把已切割或缩放的所有模糊规则后项的隶属函数合并进一个模糊集以用于每个输出变量的过程。[fl]

Algorithm（算法） 用于解决问题的指令的集合。

AND（逻辑操作符） 在产生式规则中使用时，表示所有用AND连接的前项为真时，规则后项才为真。[es]

Antecedent（前项）：规则中IF部分的条件语句，也称为前提。[es]

A-part-of：连接代表组件的子类和代表整体的父类的弧（也称作“部分－整体”）。例如，engine is a-part-of a car。[es]

Approximate reasoning（近似推理） 不需要产生式规则的IF部分与数据库中的数据精确匹配的推理。[es]

Apriori algorithm（Apriori算法） 一种关联规则学习算法。它在大型事物数据库上运行（例如顾客购买的产品）。[dm]

Arc（弧） 在表示相邻结点连接特点的语义网络中，结点之间带方向标识的标识的连接。最常见的arc就是is-a和a-part-of。[es]

Architecture（体系结构） 参见拓扑Topology。[nn]

Artificial Intelligence（AI）人工智能：计算机的一个领域，该领域关注智能机器的开发，如果机器的行为在人类身上也能观察到，就认为该机器是智能的。

Artificial Neural Network（ANN）人工神经网络：一个信息处理范例，它受人脑的结构和功能的启发而产生。ANN包括大量称作神经元的简单而高度互连的处理器，它类似于大脑的生物神经元。神经元通过把信号从一个神经元传递到另一个神经元的有权重的链接相连。在一个生物神经网络中，学习涉及调整突触，ANN通过不断调整权重来学习。这些权重存储了

解决某个问题所需的知识。[nn]

Assertion（断言） 在推理中得出的事实。[es]

Association rule（关联规则） 表示大型数据库中属性同时出现关系（关联）的规则。它表示不同属性如何同时出现。关联规则用于购物篮分析 [dm]。

Associative memory（关联记忆） 记忆的一个类型，允许我们把事物关联起来，例如，当我们听到一段音乐时，我们能回想起一个完整的感觉体验，包括声音和情景。我们甚至能在不熟悉的环境中认出熟悉的脸。当输入一个相似的模式时，相关 ANN 会想起最为接近的已存储的训练模式。Hopfield 网络就是一个相关 ANN 的例子。[nn]

Aattribute（属性） 对象的特性。例如，“计算机”对象可能有“模型”、“处理器”、“内存”
426 和“成本”等属性。[es]

Axon（轴突） 生物神经元的一根很长的枝，它能从细胞中输出信号（动作电位）。一个轴突可能长达一米。在 ANN 中，轴突由神经元输出来模拟。[nn]

Back-propagation（反向传播） 参见反向传播算法。[nn]

Back-propagation algorithm（反向传播算法） 有监督学习最常用的方法。该算法有两个阶段。首先，向输入层输入一个训练输入模式。网络逐层传送输入模式直到输出层产生输出模式为止。如果输出模式与预期输出不一致，则计算误差，然后从输出层向输入层传送误差。在传送误差时调整权重的值。参见反向传播。[nn]

Backward chaining（后向链接） 一个推理技术，从假设结论（一个目标）开始向后推理，将规则库中的规则和数据库中的事实相匹配直到目标或被证实为正确或被证实为错误。也被称作目标驱动推理。[es]

Bayesianreasoning（贝叶斯推理） 专家系统中不确定性管理的一种统计方法。其基于论据的贝叶斯规则在系统中传送不确定性。[es]

Bayesian rule（贝叶斯规则） 一种统计方法，根据新的论据来更新与事实相关的概率。[es]

Bidirectional associative memory（BAM）双向联想记忆：模拟关联记忆特征的一种神经网络，由 Bart Kosko 在 20 世纪 80 年代提出。BAM 把不同集合中的模式相关起来，反之亦然。它的基本体系结构包括两个完全连接的层——输入层和输出层。[nn]

Bit（位） 二进制数字。信息的最小单位。存储在计算机里的数据由位组成。[ke]

Bit map（位图） 对行、列的点组成的图的描述。位图可以在计算机中存储、显示和打印。光扫描仪可以把纸上的文本和图片转换成位图。扫描仪把每英寸图像分成上百个像素，并把每个像素表示成 0 或者 1。[ke]

Black box（黑盒） 对用户来说是不透明的模型，尽管模型能得出正确的结果，但并不知道它的内部联系。神经网络就是黑盒的一个例子。要理解黑盒的输入和输出间的关联，可以使用灵
427 敏度分析。[ke]

Boolean logic（布尔逻辑） 基于布尔代数的逻辑系统，以 George Boole 的名字命名。它处理两个值：“真”和“假”。布尔条件的真假经常用 0 和 1 表示，0 代表“假”，1 代表“真”。

Branch（分支） 决策树中结点间的连接。[dm]

Building block（构造块） 给染色体高度适应性的一组基因。根据构造块假设，在一个染色体上结合几个构造块可以找到最佳解决方案。

Byte（字节） 一组 8 位二进制数字，代表在现代计算机中信息的最小可寻址项。一字节信息相当于一个单词中的一个字母。1G 相当于 1 000 000 000（2^{30}或 1 073 741 824）。[ke]

C 一个多用途的编程语言，由 Bell 实验室在 UNIX 操作系统上开发。

C ++ C 语言的面向对象的扩展。

CART（Classification and Regression Trees）分类和回归树：使用决策树的数据挖掘工具。CART 提供一组能够将新数据用于预测结果的规则。CART 通过生成二元分割来给数据记录分段。[dm]

Categorical data（分类数据） 适合少量离散类别的数据。例如，性别（男或女）或者婚姻状况（单身、离异、已婚或者丧偶）。[ke]

Centroid technique（质心技术） 找到称作质心或者重心的点的逆模糊化方法，而在该点的垂线能将聚合集分割成两个相等的部分。[fl]

Certainty factor（确信因子） 赋给事实或者规则的数字，表示事实或者规则有效的可信度或者置信水平。参见置信因子。[es]

Certainty theory（确信理论） 在基于不精确推理的专家系统中，管理不确定性的推理。它使用确信因子表示观察到某个事件的假设的可信度。[es]

Child（子代） 参见后代（Offspring）[ec] 428

Child（子结点） 在决策树中，子结点就是由树中的上一层结点通过分割产生的结点。一个子结点包含其父结点中数据的子集。[dm]

Chromosome（染色体） 代表一个个体的基因串。[ec]

Class（类） 有共同属性的对象集。animal、person、car 和 computer 都是类。[es]

Class-frame（类框架） 代表一个类的框架。[es]

Clipping（剪切） 把模糊规则的后项和规则前项的真值相联系的常用方法。该方法基于在前项真值水平上裁剪后项隶属函数。因为从隶属函数的顶部分段，被剪切的模糊集会丢失一些信息。[fl]

Cloning（克隆） 生成一个其亲代的完全副本的后代。[ec]

Clustering（聚类） 把不同种类的对象划分成同类子集的过程。聚类算法可用于查找在某些方面相似的类。例如，保险公司可以用聚类算法根据客户的年龄、资产、收入和预先的要求对他们分组。[ke]

Coding（编码） 将信息从一种表现方案转换到另一种方案的过程。[ec]

Cognitive science（认知科学） 研究如何获得和使用知识的交叉学科。涉及的学科包括人工智能、心理学、语言学、哲学、神经科学和教育学。也就是，关于用计算模拟智能行为的智能和智能系统的研究。

Common sense（常识） 如何解决现实世界中的问题的普遍知识，通常通过实践经验获得。[ke]

Competitive learning（竞争学习） 无监督的学习，神经元之间彼此竞争使得只有一个神经元能够响应某个输入模式。在竞争中获胜的神经元称为胜者通吃神经元。Kohonen 自组织特征图是竞争学习的 ANN 的一个例子。[nn]

Complement（补） 在经典的集合论中，集合 A 的补集就是不是 A 的成员元素的集合。在模糊集理论中，集合的补集就是它的反集。[fl] 429

Confidence factor（置信因子） 参见确信因子（Certainty factor）。[es]

Conflict（冲突） 两个或者多个产生式规则与数据库中的数据匹配，但是只能有一个规则在给定的循环中被激发的状态。[es]

Conflict resolution（冲突消解） 当有多个规则可以在给定的循环中被激发时，选择其中一个被激发的产生式规则的方法。[es]

Conjunction（合取） 逻辑操作符 AND，能连接两个产生式规则中的前项。[es]

Connection（连接） 神经元之间传递信号的链接，也称作突触，它通常与决定传递信号的强度

的权重相关联。[nn]

Consequent（后项） 规则中 IF 部分的结论或者动作。[es]

Continuous data（连续数据） 在某个间隔中有无限个可能取值的数据。例如，长度、重量、家庭收入和房子居住面积都是连续数据。连续数据通常都是可度量的，但并不一定是整数。[ke]

Convergence（收敛） 当误差达到预设的阈值，表明网络已经学习了任务，即 ANN 网络是收敛的。[nn]

Convergence（收敛） 种群中的个体有达到一致的趋势。当已获得一个解决方案时，称遗传算法收敛。[ec]

Crossover（交叉） 通过交换两个现有染色体的部分来产生新染色体的繁殖操作。[es]

Crossover probability（交叉概率） 介于 0 ~ 1 的数，表示两个染色体交叉的可能性。[ec]

Darwinism（达尔文主义） 查尔斯·达尔文的学说，指出通过自然选择得到进化，同时伴随遗传特性的随机改变。

Data（数据） 事实、测量数据或者观测数据。也可以是事实、测量数据或者观测数据的符号表
430 示。数据就是我们收集和保存的东西。

Data cleaning（数据清洗） 探测和纠正明显的误差并替换数据库中丢失的数据的过程，也称作数据清洁（Data cleansing）。[dm]

Data cleansing：参见数据清洗。[dm]

Data cube（数据立方体） 表示多维聚集数据。它提供一种从不同角度观察数据的方法。数据立方体模型类似于 Rubik 立方体，用户能够容易地将一个维度组合转换为另一个。[dm]

Data-driven reasoning：参见前项链接（Forward chaining）。[es]

Data mining（数据挖掘） 数据中知识的提取，也就是为了发掘有用的模式和规则，对大量数据的研究和分析。数据挖掘的最终目标是发现知识。[dm]

Data record（数据记录） 与单个对象的属性相对应的一系列值。一个数据记录是数据库中的一行。参见记录。[dm]

Datavisualization（数据可视化） 数据的图形化表示，能帮助用户理解数据的结构和数据中包含的信息的含义。参见可视化。[dm]

Data warehouse（数据仓库） 一个大型的数据库，包含了上百万条，甚至上亿条的数据记录，用于支持组织的决策制定。它是结构化的，能快速在线查询和管理摘要。[dm]

Database（数据库） 结构化数据的集合。数据库是专家系统的基本组成部分。[es]

Database management system（DBMS）（数据库管理系统） 实现创建和维护数据库的一组程序。[dm]

Decision-support system（决策支持系统） 一个交互的基于计算机的系统，用来帮助个人或组织在特定领域进行决策。[es]

Decision tree（决策树） 一个数据集的图形化表示，以树状结构描述数据。一个决策树包括结点、分支和叶子。树通常开始于根结点，通过在每层上分裂数据形成新结点而往下生长。决策树特别适合解决经典的问题，它的主要优点是数据可视化。[dm]

431 Deductive reasoning（演绎推理） 从一般到特殊的推理。[es]

Defuzzification（逆模糊化） 模糊推理的最后一步，将结合的模糊规则的输出转化为清晰（数字化）的值的过程。逆模糊化的输入是聚合集，输出是单个数字。[fl]

Degree of membership（隶属度） 0 ~ 1 之间的数值，表示元素属于某个集合的程度。参见隶属值（Membership value）。[fl]

Delta rule（Delta 规则） 在训练过程中更新感知器权重的过程，Delta 规则通过将神经元输入乘以误差和学习速率的输出来决定权重的校正值。[nn]

Demon（守护程序） 与一个槽相关的过程，当槽值发生变化或被请求时会执行，守护程序通常拥有一个 IF - THEN 结构，它和方法为同义词。[es]

DENDRAL：20 世纪 60 年代末，在 Stanford 大学开发的基于规则的专家系统，它基于广谱仪提供的广谱数据来分析化学产品。DENDRAL 标志着人工智能领域重要的“范式转换”：从通用的知识稀少的方法转向针对特定领域的知识密集型的技术。[es]

Dendrite（树突） 生物神经元的一个分支，在细胞的不同部分之间传递信息。树突通常负责细胞输入的功能，尽管很多树突也有输出的功能。在 ANN 中，神经元输入模拟树突。[nn]

Deterministic model（确定性模型） 一个数学模型，它能假定对象之间准确的关系（不识别随机变量）。给定一个数据输入集，确定性模型决定了它的输出也是完全确定的。[es]

Discrete data（离散数据） 只能取有限个不同值的数据。离散数据通常（并非必须）是数字。例如家庭中孩子的数量，卧室的数量，帆船桅杆数量等都是离散数据。[ke]

Disjunction（析取） 逻辑操作符 OR，连接产生式规则的两个前项。[es]

Domain（域） 相对狭窄的问题范围。例如，在医学领域诊断血液病。专家系统就是关注特定域的。[es]

Domain expert（领域专家） 参见 Expert。[es] 432

EMYCIN 即 Empty MYCIN，20 世纪 70 年代后期在斯坦福大学开发的一个专家系统。除了传染性血液病的知识外，它拥有 MYCIN 系统的所有特征。EMYCIN 用来开发诊断专家系统。[es]

End-user（终端用户） 参见 User。[es]

Epoch（周期） 在训练中，把整个训练集输入给 ANN 的过程。[nn]

Error（误差） 在有监督学习的 ANN 中，实际输出和预期输出之间的差。[nn]

Euclidean distance（欧几里得距离） 空间中两点之间最短的距离。在笛卡儿坐标中，两点 (x_1, y_1) 和 (x_2, y_2) 之间的欧几里得距离，由毕达哥拉斯定理 $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ 决定。

Evolution（进化） 一个活的生物获得区别于其他生物的特征的一系列基因变化。[ec]

Evolution strategy（进化策略） 与蒙特·卡洛搜索类似的数据优化过程。与遗传算法不同，进化策略只使用突变操作，并不要求以编码形式描述一个问题。当没有任何可用的分析对象函数，也没有传统的优化方法存在时，进化策略用来解决技术优化问题。[ec]

Evolutionary computation（进化计算） 在计算机上模拟进化的计算模型。进化计算的领域包括遗传算法、进化策略和遗传编程。[ec]

Exhaustive search（穷举搜索） 一种问题解决技术，即测试每一种可能的方法，直到找到一种可用的方法为止。[es]

Expert（专家） 在某个领域有渊博的知识（以事实和规则的形式表示）和丰富经验的人。参见领域专家。[es]

Expert system（专家系统） 在狭窄的领域中能够以人类专家水平执行的计算机程序。专家系统有 5 个基本的组成部分：知识库、数据库、推理引擎、解释工具和用户界面。[es]

Expert system shell（专家系统框架） 去除知识的骨架专家系统。参见框架。[es] 433

Explanation facility（解释工具） 专家系统的一个基本组成部分。它有助于用户查询专家系统是如何得到某个结论以及为什么需要特定的事实来得到结论。[es]

Facet（侧面） 为框架的属性提供知识扩展的一种方法。侧面用来建立属性值，控制用户查询，

并告知推理引擎如何处理属性。[es]

Fact（事实） 具有正确或错误的属性的语句。[es]

Feedback neural network（反馈神经网络） ANN 的一种拓扑，它的神经元带有从输出到输入的反馈回路。反馈神经网络的一个例子就是 Hopfield 网络。参见循环网络。[nn]

Feedforward neural network（前馈神经网络） ANN 的一种拓扑，它的某一层的神经元和下一层的神经元连接在一起。输入信号一层一层向前传递。前馈神经网络的一个例子是多层感知器。[nn]

Field（字段） 数据库中为某个属性分配的空间。（在电子表单中，字段叫做单元。）例如，纳税申报表格包括很多字段：你的姓名、地址、纳税文件号、应征税的收入等。数据库中的每个字段都有一个名称，叫做字段名。[dm]

Firing a rule（激发规则） 执行产生式规则的过程，或者更准确地说，就是当 IF 部分为真的时候，执行 THEN 部分。[es]

Fitness（适应性） 活的生物体在特殊环境中生存和繁殖的能力。也就是一个与染色体相关的值，它把一个相对量赋给该染色体。[ec]

Fitness function（适应性函数） 一个用来计算染色体的适应性的数学函数。[ec]

Frame（框架） 带有某个对象的典型知识的数据结构。框架用来在基于框架的专家系统中表示知识。[es]

Frame-based expert system（基于框架的专家系统） 一种专家系统，其中框架表示主要的知识源，方法和守护程序用来向框架中添加动作。在基于框架的专家系统中，产生式规则起辅助作用。[es]

434

Fuzzification（模糊化） 模糊推理的第一步，把清晰（数字）输入映射到确定每个输入属于合适模糊集的程度的过程。[fl]

Fuzzy expert system（模糊专家系统） 使用模糊逻辑而不是布尔逻辑的专家系统。模糊专家系统是由模糊规则和有关的数据推理的隶属函数组成的。与传统的专家系统使用符号推理不同，模糊专家系统面向数值处理。[fl]

Fuzzy inference（模糊推理） 基于模糊逻辑的推理过程。模糊推理包含 4 个步骤：输入变量的模糊化、规则评估、规则输出的聚合以及逆模糊化。[fl]

Fuzzy logic（模糊逻辑） 用来描述那些不能简单地用二进制的术语“真”和“假”来描述的情况的逻辑系统。这个概念由 Lotfi Zadeh 于 1965 年提出。与布尔逻辑不同，模糊逻辑是多值的，它用来处理局部真的概念（在“完全正确”和“完全错误”之间的真）。参见模糊集理论。[fl]

Fuzzy rule（模糊规则） 下列形式的条件语句——IF x is A THEN y is B，其中 x 和 y 是语言变量，A 和 B 是由模糊规则决定的语言变量。[fl]

Fuzzy set（模糊集） 有模糊边界的集合，如用“矮”、“平均”或者“高”来描述人的身高。为了在计算机上表示模糊集，我们使用函数来表示它，然后再把集合的元素映射到其隶属度上。[fl]

Fuzzy set theory（模糊集理论） 参见模糊逻辑。[fl]

Fuzzy singleton（模糊单态模式） 一种具有隶属函数的模糊集，在论域中的某一点函数值为 1，在任何其他情况下则为零。参见单一事例。[fl]

Fuzzy variable（模糊变量） 能取语言值的量。例如，模糊变量“温度”有“热”、“中等”和“冷”这样的值。[fl]

Gene（基因） 染色体的一个基本单元，它控制着生物体的某个特征的发展。在 Holland 的染色体中，基因是用 0 或者 1 来表示的。[ec]

General Problem Solver（通用问题解决方案，GPS） 早期人工智能系统，它试图模拟人类解决问题的方法。通用问题解决方案首先试图把问题的解决技术与数据分开。然而，程序是基于通用搜索机制的。该方法现在被认为是弱方法，它运用了问题域中的弱信息，这导致在解 435
决现实世界的问题时，程序性能低下。[es]

Generalisation（泛化） ANN 利用没有训练的数据产生正确结果的能力。[nn]

Generation（代） 遗传算法的一次迭代。[ec]

Genetic algorithm（遗传算法） 受达尔文进化理论启发而得到的一种进化计算。遗传算法产生许多可能用染色体编码的解决方案，估计它们的适应性，同时应用遗传操作（如交叉、突变）创建新的种群。在许多代重复这个过程，遗传算法为解决问题提供了合适的解决方案。[ec]

Genetic operator（遗传操作） 遗传算法或遗传编程中的一种操作，它对染色体进行操作，从而产生新的个体。遗传操作包括交叉和突变。[ec]

Genetic programming（遗传编程） 遗传算法在计算机程序中的应用。当程序设计语言允许程序像数据一样被操作以及新产生的数据像程序一样被执行时，遗传编程最容易实现。这也许是 LISP 作为遗传编程最主要的语言的原因之一。[ec]

Global minimum（全局最小化） 所有输入参数在其整个范围内的最小函数值。在训练中，ANN 的权重会进行调整以寻找误差函数的全局最小值。[nn]

Global optimization（全局最优化） 在整个搜索空间中寻找最佳的值。[ec]

Goal（目标） 专家系统试图证明的一个假设。[es]

Goal – driven reasoning（目标驱动推理） 参见后向链接。[es]

Hard – limit activation function（硬限幅激活函数） 用阶跃函数和符号函数表示的一种激活函数，参见硬限幅函数。[nn]

Hard limiter（硬限幅函数） 参见硬限幅激活函数。[nn]

Hebb's Law（Hebb 法则） 该学习法则由 Donald Hebb 于 20 世纪 40 年代后期提出。它提出如果 436
神经元 i 和神经元 j 足够近并能够不断地激活神经元 j，那么两个神经元之间的突触就会加强，神经元 j 对来自神经元 i 的刺激就会格外敏感。这个法则为无监督学习奠定了基础。[nn]

Hebbian learning（Hebbian 学习） 把一对神经元之间的突触权重的变化与产生输入和输出信号关联起来的无监督学习。[nn]

Hedge（模糊限制语） 可以修改模糊集形状的模糊集术语。它包含 very、somewhat、quite、more or less 和 slightly 这样的副词。它们通过减少模糊元素的隶属度的集中（如 Very tall men），通过增加模糊元素的隶属度来扩张（如 more or less tall men），通过增加隶属度到 0.5 以上，或减少隶属度到 0.5 以下的增强（如 indeed tall men）。[fl]

Heuristic（启发式方法） 一种能应用于复杂问题的策略，它通常（但不总是）能得出正确的解决方案。从多年的经验总结中得出的启发式方法经常用来根据判断将复杂的问题简化为简单的操作。启发式方法经常被表达为经验法则。[es]

Heuristic search（启发式搜索） 应用启发式方法来指导推理，从而减少搜索空间的搜索技术。[es]

Hidden layer（隐含层） 介于输入层和输出层之间的神经元层。之所以叫做“隐含”层，是因为这一层中的神经元不能通过神经网络的输入/输出行为来观察到。没有明显的方法可以知道隐含层的预期输出应该是什么。[nn]

Hidden neuron（隐含神经元） 隐含层中的神经元。[nn]

Hopfield network（Hopfield 网络） 单层反馈神经网络。在 Hopfield 网络中，每个神经元的输出反馈给其他所有神经元的输入（其中没有自反馈）。Hopfield 网络经常用带有符号激活函数的 McCulloch 和 Pitts 神经元。Hopfield 网络试图模拟关联记忆的特征。[nn]

Hybrid system（混合系统） 至少结合了两种智能技术的系统。例如，结合了神经网络和模糊系统的混合神经 - 模糊系统。[ke]

Hypothesis（假设） 有待证明的语句，也指专家系统中使用后向链接的目标。[es]

[437] **Individual（个体）** 种群中的单个成员。[ec]

Inductive reasoning（归纳推理） 从特殊到一般的推理。[es]

Inference chain（推理链） 指出专家系统如何应用规则库中的规则来得到结论的一系列步骤。[es]

Inference engine（推理引擎） 专家系统的一个基本组成部分，它执行推理，由此专家系统得到解决方案。推理引擎将规则库中的规则和数据库中的事实匹配起来。参见解释器。[es]

Inference technique（推理技术） 在专家系统中推理引擎用来直接搜索和推理的技术。它有两种主要的技术：前向链接和后向链接。[es]

Inheritance（继承） 类框架的所有特征由实例框架接受的过程。继承是基于框架系统的一个基本特征。继承的常规用法是把默认的属性强加到所有的实例框架上。[es]

Initialization（初始化） 训练算法的第一步，设置权重和阈值的初始值。[nn]

Input layer（输入层） ANN 中神经元的第一层。输入层从外界接收输入信号，然后重新将信号发送到下一层的神经元。输入层很少包含计算神经元，因此不处理输入模式。[nn]

Input neuron（输入神经元） 输入层中的神经元。[nn]

Instance（实例） 类的一个具体对象。如 computer 类可能有 IBM Aptiva S35 和 IBM Aptiva S9C 两个实例。在基于框架的专家系统中，类的所有特征被它的实例所继承。[es]

Instance（实例） 模式的一个成员。例如，染色体 1110 和 1010 是模式 1 * *0 的一个实例。[ec]

Instance - frame（实例框架） 表示实例的框架。[es]

Instantiation（实例化） 把特定值分配给一个变量的过程。例如，August 是对象 month 的一个实例化。[es]

Intelligence（智能） 为解决问题和制定决策而学习与理解的能力。如果一个机器在某些认知的
[438] 任务中能够达到人类的水平，就认为它是智能的。

Interpreter（解释器） 参见推理引擎。[es]

Intersection（交） 在经典集合论中，两个集合的交包含两个集合所共有的元素。例如，tall men 和 fat men 的交包含既高又胖的男人。在模糊集理论中，一个元素可能部分地属于两个集合，则交集是两个集合中隶属值最低的元素。[fl]

Is - a 在基于框架的专家系统中，把超类和子类联合在一起的一个弧。例如，如果 car is - a vehicle，那么 car 是更通用的超类 vehicle 的子类。每个子类从其超类中继承所有的特性。[es]

Knowledge（知识） 对某个主题在理论上或实践上的理解。知识帮助我们做出正式的决定。

Knowledge acquisition（知识获取） 获取、研究和组织知识的过程，因此它可以用于基于知识的系统中。[ke]

Knowledge base（知识库） 专家系统的一个基本组成部分，它包含特定领域的知识。[es]

Knowledge - base system（基于知识的系统） 一个用存储的知识来解决某个领域的问题的系统。对基于知识的系统的评估通常是比较它的性能与人类专家的性能。

Knowledge engineer（知识工程师） 设计、构建并测试基于知识的系统的人。知识工程师从领域专家那里获得知识，然后建立推理方法和选择开发软件。[ke]

Knowledge engineering（知识工程） 建立基于知识的系统的过程。它包括 6 个主要的步骤：评估问题、获取数据和知识、开发原型系统、开发完整系统、评估并修改系统、整合和维护系统。[ke]

Knowledge representation（知识表达） 组织知识以存入基于知识的系统的过程。在人工智能中，产生式规则是知识表达最常用的类型。[ke]

Kohonen self – organising feature maps（Kohonen 自组织特征映射） Teuvo Kohonen 于 20 世纪 80 年代晚期提出的具备竞争学习的一种特殊的 ANN。Kohonen 映射由一个有两种类型的连接的单层计算神经元组成。这两个连接是输入层的神经元到输出层的神经元的前向连接，以及输出层的神经元之间的横向连接。横向连接用来在神经元之间产生竞争。神经元通过 439
将它的权重从非活动连接变为活动连接来学习。只有获胜的神经元和它的邻近神经元可以学习。[nn]

Layer（层） 有特殊功能，并且是作为一个整体处理的一组神经元。例如，多层感知器至少有 3 层：输入层、输出层以及一个或多个隐含层。[nn]

Leaf（叶结点） 决策树中最低端的结点，它没有子结点。参见终端结点。[dm]

Learning（学习） 在 ANN 中，为了达到网络的某些预期行为而调整权重的过程。参见训练。[nn]

Learning rate（学习速率） 小于单位 1 的正数，它控制 ANN 中的迭代从一次到下一次的权重的变化量。学习速度直接影响网络训练的速率。[nn]

Learning rule（学习规则） 在训练 ANN 时，修改权重的过程。[nn]

Linear activation function（线性激活函数） 产生的输出和神经元的净输入相等的激活函数。具有线性激活函数的神经元一般用在线性近似中。[nn]

Linguistic value（语言值） 模糊变量能接受的语言元素。例如，模糊变量 income 可以接受 very low、low、medium、high 和 very high 这样的语言。语言值通过隶属函数来定义。[fl]

Linguistic variable（语言变量） 有语言元素值的变量，如单词和短语。在模糊逻辑中，术语语言变量和模糊变量是同义的。[fl]

LISP（LISt Processor） 最早的高级编程语言之一。LISP 由 John McCarthy 于 20 世纪 50 年代后期发明，已成为人工智能的标准语言。

Local minimum（局部最小化） 输入参数在其有限的取值范围内的最小函数值。如果在训练中遇到局部最小化，将永远不会达到 ANN 的预期行为。去除局部最小化的常用方法就是利用随机权重并继续训练。[nn]

Machine learning（机器学习） 使计算机能够通过经验、例子和类比进行学习的自适应机制。学习能力可以改善智能系统的性能。机器学习是自适应系统的基础。机器学习最常见的方法是人工神经网络和遗传算法。 440

Market basket analysis（购物篮分析） 一种数据挖掘技术，通过分析两个或多个项在上下文中同时出现的次数来发现项间的统计关系。购物篮分析回答这样的问题，例如“如果一个用户购买了 A，他购买 B 的可能有多大?”和“如果用户购买了 A 和 B，她可能还会购买什么物品?”。

Massaging data（修改数据） 在应用于 ANN 输入层之前修改数据的过程。[nn]

McCulloch 和 Pitts neuron model（McCulloch 和 Pitts 神经元模型） 由 Warren McCulloch 和 Walter Pitts 于 1943 年提出的神经元模型，至今仍是大多数人工神经网络的基础。这个模型由一个硬限幅器后跟一个线性组合器组成。净输入被应用于限幅器，当输入是正数，输出就是 +1，当输入是负数，输出就是 -1。[nn]

Membership function（隶属函数） 在论域中定义模糊集的数学函数。在模糊专家系统中使用的典型隶属函数有三角形函数和梯形函数。[fl]

Membership value（隶属值） 见隶属度。[fl]

Metaknowledge（元知识） 关于知识的知识。元知识是在专家系统中使用和控制领域知识的知

识。[es]

Metarule（元规则） 描述元知识的规则。元规则决定了在专家系统中使用特殊任务规则的策略。[es]

Method（方法） 方法是和框架属性相关的过程。方法决定属性值或在属性值发生变化时执行一系列动作。大多数基于框架的专家系统有两种方法：WHEN CHANGED 和 WHEN NEEDED。方法和守护程序通常作为同义词使用。[es]

Momentum constant（要素常数） delta 规则中小于单位 1 的正数。要素的使用加快了在多层感知器里的学习速度，有助于避免遇到局部最小化。[nn]

Multilayer perceptron（多层感知器） ANN 最普通的拓扑，其中感知器被连接起来成为单独的层。多层感知器有输入层、至少一个隐含层以及输出层。操作多层感知器最常用的方法是反向传播。[nn]

441

Multiple inheritance（多重继承） 一个对象或者框架能够从多重超类中继承信息的能力。[es]

Mutation（突变） 随机改变染色体中的基因值的遗传操作。[ec]

Mutation probability（突变概率） 揭示在单个基因中发生突变可能性的 0 ~ 1 之间的数值。[ec]

MYCIN MYCIN 是一个于 20 世纪 70 年代提出的，用以诊断传染性血液疾病的一个典型的基于规则的专家系统，这个系统使用确信因子来管理与医学诊断知识有关的不确定性。[es]

Natural selection（自然选择） 适应性最强的个体进行配对和繁殖的机会更大，从而把它们的遗传物质传递给下一代的过程。[ec]

Neural computing（神经计算） 模拟人脑的计算方法，它依靠连接大量简单的处理器来产生复杂的行为。神经计算能够在专门的硬件或使用称为人工神经网络的软件上实现，人工神经网络在传统计算机上模拟人脑的结构和功能。

Neural network（神经网络） 把处理元素（称为神经元）连接在一起形成网络的系统。生物学神经网络的基本的和本质的特征是学习的能力。人工神经网络也有这种能力，它们没有被编程，但是可以通过重复地调整权重来学习。[nn]

Neuron（神经元） 能够处理信息的细胞。一个典型的神经元有许多输入（树突）和一个输出（轴突）。人脑大概有10^{12}个神经元。同时它也是 ANN 的一个基本处理元素，用于计算输入信号的权重和，并把结果通过激活函数产生一个输出。[nn]

Node（结点） 决策树上的一个决策点。[dm]

Noise（噪声） 影响传输信号的一个随机外部干扰。噪声数据包含与数据的采集、测量和解释的方法有关的误差。[dm]

NOT 表示语句的反面的一个逻辑操作。[es]

Object（对象） 一个概念、抽象或事物，能够被单独选择和操作，并且它对问题而言是有意义的。所有对象都有特征，可以被清晰地区分开来。Michael Black、Audi 5000 Turbo 和 IBM

442

Aptiva S35 就是对象的例子。在面向对象的编程中，对象是一个自包含的实体，它包括数据和处理数据的过程。[es]

Object – oriented programming（面向对象编程） 把对象作为分析、设计和实现的基础的一种编程方法。[es]

Offspring（后代） 通过繁殖产生的个体。参见子代。[ec]

On – line analytical processing（OLAP）（联机分析处理） 支持多维视图和快速访问大量数据的软件技术，OLAP 能够实现数据立方体上的“切面、切片”和“上卷、下钻”操作。[dm]

Operational database（操作型数据库） 用于企业日常操作的数据库。操作数据库中的数据会进行有规律地更新。[dm]

OPS 起源于 LISP，用于开发基于规则的专家系统的一种高级编程语言。[es]

Optimisation（优化） 利用特殊的对象函数来迭代改进问题的解决方案的过程。[ec]

OR 一个逻辑操作符，当运用在产生式规则中时，它意味着如果和 OR 连在一起的任何一个前项为真的话，那么规则的后项也为真。[es]

Outlier（离群值） 到其他数据的距离值。[dm]

Output layer（输出层） ANN 中神经元的最后一层。输出层产生整个网络的输出模式。[nn]

Output neuron（输出神经元） 输出层中的神经元。[nn]

Overfitting（过拟合） ANN 能记住所有的训练例子，但是不能进行推广的现象。如果隐含神经元的数量太大，就会发生过拟合。防止过拟合的可行办法就是选择能产生最佳推广的隐含神经元的最小数量。参见过训练。[nn]

Over－training（过训练） 参见过拟合。[nn]

Parallel processing（并行处理） 同时处理多个任务的计算技术。人脑就是并行信息处理系统的一个例子，它在整个生物神经网络中（而不是在特殊的地方）同时存储和处理信息。[nn] 443

Parent（亲代） 能产生一个或者多个个体（后代或子代）的个体。[ec]

Parent（父结点） 在决策树中，父结点是把它的数据分离到该树的下一层结点的结点。父结点包含整个数据集，而子结点只包含整个数据集的子集。[dm]

Pattern recognition（模式识别） 计算机可识别的视频或音频模式。模式识别包括把模式转化为数字信号，然后将这些数字信号与存储在内存中的模式进行比较。人工神经网络成功应用于模式识别，特别是声音和特征的识别、雷达目标的辨别和遥控领域更为常用。[nn]

Perceptron（感知器） 由 Frank Rosenblatt 提出的最简单的神经网络形式。感知器的操作基于 McCulloch 和 Pitts 的神经模型。它由一个带有调整突触权重的神经元和一个限幅器组成。感知器通过细微地调整权重来减少实际输出和预期输出的差别来进行学习。初始权重是随机设置的，接着更新初始值来获取和训练例子一致的输出值。[nn]

Performance（性能） 适应性的统计评估。[ec]

Performance graph（性能图） 揭示整个种群的平均性能和种群中最优秀的个体经过制定代数后性能的图。[ec]

Pixel（像素） 图片元素。图像中的一个点。计算机显示器是通过将屏幕分成成千上万（或几百万个）按行和列组织的像素来展示图片的。像素之间靠得很近，因此它们看起来像一个图像。[ke]

Population（种群） 在一起生活的一群个体。[ec]

Premise（前提） 参见前项。[es]

Principal component analysis（PCA）（主成分分析） 将高维相关变量转换为低维不相关变量的数学过程。[dm]

Probability（概率） 某个事件可能发生的定量的描述。概率在数学上定义为一个在 0（绝对不可能）和 1（绝对肯定）之间的一个数。[es] 444

Procedure（过程） 计算机代码中自包含的一段代码。[es]

Production（产生式） 认知心理学家用来描述规则的常用术语。[es]

Production rule（产生式规则） IF（前项）－THEN（后项）形式的语句。如果前项为真，那么后项也为真。参见规则。[es]

PROLOG 20 世纪 70 年代后期由 Marseilles 大学开发的一种高级编程语言，它是逻辑编程的实用工具，也是人工智能的常用语言。

PROSPECTOR 20 世纪 70 年代后期由斯坦福研究所提出的用于矿藏勘探的专家系统。为了描述知识，PROSPECTOR 使用包括产生式规则和语义网络的组合结构。[es]

Query tool（查询工具） 允许用户创建并向数据库提出特定问题的软件。查询工具提供了从数据库获取所需信息的方法。[dm]

Reasoning（推理） 得出结论或通过观察结果、事实或假设进行推导的过程。[es]

Record（记录） 参见数据记录。[dm]

Recurrent network（循环网络） 参见反馈网络。[nn]

Regression analysis（回归分析） 发现最符合数据的数学公式的统计技术。[dm]

Relational database（关系数据库） 以表形式表示的数据集合，可以访问和重装数据，而无需重新组织表本身。[dm]

Reproduction（繁殖） 亲代创建后代的过程。[ec]

Root（根） 参见根结点。[dm]

Root node（根结点） 决策树最顶端的结点。树总是从根结点开始向下生长，并在每一层分割数据以产生新的结点。根结点包含整个数据集（所有的数据记录）。子结点包含该集的子
445 集。参见根。[dm]

Roulette wheel selection（轮盘选择） 从种群中选出一个个体作为亲代，其概率等于它的适应性除以种群总的适应性的方法。[ec]

Rule（规则） 参见产生式规则。[es]

Rule base（规则库） 包含一组产生式规则的知识库。[es]

Rule – based expert system（基于规则的专家系统） 其知识库包含一系列产生式规则的专家系统。[es]

Rule evaluation（规则评估） 模糊推理的第二步。把模糊输入应用于模糊规则的前项，以及决定每个规则的前项的真值的过程。如果给定的规则有多个前项，则使用模糊操作（交或者并）来得到一个数值，该数值代表前项评估的结果。

Rule – of – thumb（经验法则） 表达启发式方法的规则。[es]

Scaling（缩放） 把模糊规则的后项和前项规则的真值联系在一起的方法。它是基于规则后项的初始隶属函数乘以规则前项的真值来调整初始隶属函数的。缩放用来保存模糊集的原始形状。[fl]

Scatter plot（散点图） 表示两个变量相关度的二维图。数据表示为点的集合。每个点的水平坐标由一个变量决定，垂直坐标由另一个变量决定。[dm]

Schema（模式） 模式是包含 0、1 和星号的位串，其中星号表示 0 或者 1。例如，模式 1 * * 0 表示 4 位串，每个串开始于 1，结束于 0。[ec]

Schema theorem（模式定理） 下一代中给定的模式的实例的数量与这一代中的模式的适应性以及染色体的平均适应性关联起来的理论。这个理论表明，高于平均适应性的模式容易在下一代中频繁发生。[ec]

Search（搜索） 通过检查解决问题的一系列可能的解决方案来找出可以接受的解决方案的过程。[es]

446 **Search space（搜索空间）** 给定问题的所有可能的解决方案的集合。[es]

Selection（选择） 根据适应性来选择用于繁殖的亲代的过程。[ec]

Self – organised learning（自组织学习） 参见无监督学习。[nn]

Semantic network（语义网络） 通过一个由带标记的结点和弧构成的图来表达知识的方法，图中的结点表示对象，弧表示这些对象之间的关系。[es]

Sensitivity analysis（灵敏度分析） 确定模型的输出对特定输入的敏感程度的技术。该技术可以

用来理解不透明模型的内部关系，因此可以应用在神经网络中。通过将每个输入设置成最小值，然后再设置成最大值，并测量网络的输出来执行灵敏度分析。[ke]

Set（集） 元素或者说成员的集合。

Set theory（集合论） 对对象的类或者集合的研究。集合是数学的基本单元。经典集合理论并不承认模糊集，模糊集的元素在某种程度上可以属于很多个集合。经典集合理论是二元的：元素要么属于一个集合，要么不属于一个集合。也就是说，经典集合理论为集合中的每个元素所赋的值为1，为不属于那个集合的所有元素赋值为0。

Shell（框架） 参见专家系统框架。[es]

Sigmoid activation function（S形激活函数） 把一个在正无穷大和负无穷大之间的任意输入值转变为0～1之间的一个合理值的激活函数。有这个功能的神经元被用在多层感知器中。[nn]

Sign activation function（符号激活函数） 如果输入是正的，那么它的输出是+1，如果输入是负的，那么它的输出是-1的一个硬限幅激活函数。[nn]

Singleton（单态模式） 参见模糊单态模式。[fl]

Slot（槽） 基于框架系统的一个框架组成部分，它描述框架的某个属性。例如，框架computer有一个用于model属性的槽。[es]

Soma（细胞） 生物神经元的载体。[nn]

Step activation function（阶跃激活函数） 如果输入是正的，那么它的输出是+1，如果输入是负的，那么它的输出是-1的一个硬限幅激活函数。[nn] 447

Structured Query Language（SQL）（结构化查询语言） 一种实现检索、插入、更新和删除数据的命令语言。SQL的第一个版本被称为SEQUEL，由IBM的Donald Chamberlin和Raymond Boyce于20世纪70年代早期创建。[dm]

Summary statistics（摘要统计） 表示数据特性的一组值，例如均值、中值、模和标准差等。[dm]

Supervised learning（有监督学习） 需要一个能提供一系列训练例子给ANN教师的学习类型。每个例子都包含输入模式和由网络产生的预期输出模式。这个网络决定了实际的输出，并把它与训练例子的预期输出进行比较。如果网络得到的输出和训练例子的预期输出不同，那么网络权重将被修改。监督学习的最常用的方法是反向传播。[nn]

Survival of the fittest（适者生存） 只有适应性最强的个体才能够生存下来，并把它们的基因传给下一代的理论。[ec]

Symbol（符号） 描述一些对象的字符或字符串。[es]

Symbolic reasoning（符号推理）：用符号来推理。[es]

Synapse（突触） 生物神经网络中两个神经元的化学媒介连接，因此一个细胞的状态会影响另一个细胞的状态。虽然还有许多其他的组织，但是突触经常出现在轴突和树突之间。参见连接。[nn]

Synaptic weight（突触权重） 参见权重。[nn]

Terminal node（终端结点） 参见叶子。[dm]

Test set（测试集） 用来测试一个ANN的推广能力的数据集。测试数据集是严格独立于训练集的，它包括在网络中从来没有出现的一些例子。一旦训练完成，数据集马上用来验证网络。[nn]

Threshold（阈值） 在神经元的输出产生之前必须超出的特定值。例如，在McCulloch和Pitts神

经模型中，如果净输入小于阈值，那么神经元的输出就是 -1；如果净输入大于或者等于阈
448 值，那么神经元被激活，输出为 +1。参见极限值。[nn]

Threshold value（极限值） 见阈值。[nn]

Topology（拓扑） 神经网络的一个结构，指神经网络的层数、每一层神经元的个数以及神经元之间的连接。参见体系结构。[nn]

Toy problem（玩具问题） 人造问题，如游戏。也指一个复杂问题的不现实采用。[es]

Training（训练） 参见学习。[nn]

Training set（训练集） 用于训练 ANN 的数据集。[nn]

Transfer function（传递函数） 参见激活函数。[nn]

Truth value（真值） 一般而言，术语真值与隶属值是同义词。真值反映模糊语句的真实性。例如，模糊命题 x is A (0.7) 表示元素 x 是模糊集 A 的隶属度是 0.7。这个数字表示命题的真实性。[fl]

Turing test（图灵测试） 用于决定一个机器是否能够通过智能行为的测试。图灵把计算机的智能行为定义为在某个认知任务中能达到人类水平的能力。在测试当中，某人通过神经介质（如远程终端）来询问某人或某物。如果提问者不能将人和机器区分开，那么该计算机就能通过这个测试。

Union（并） 在经典的集合理论中，两个集合的并由属于两个集合的所有元素组成。例如，tall men 和 fat men 的并包含所有高或胖的男人。在模糊集理论中，并是交的逆操作，也就是说，并集由两个集合中隶属值较大的元素组成。[fl]

Universe of discourse（论域） 应用到所给变量中所有可能值的范围。[fl]

Unsupervised learning（无监督学习） 不需要额外教师的一种学习类型。在学习中，ANN 接收一些不同的输入模式，发现这些模式中的重要特征，并学会怎样将输入数据放入合适的类中。参见自组织学习。[nn]

User（用户） 使用基于知识的系统的人。例如，用户有可能是确定分子结构的分析化学家，诊
449 断传染性血液疾病的医生，一位试图发现新的矿藏的地质勘探学家，或者在紧急情况下寻找建议的电力系统操作员。参见终端用户。[es]

User interface（用户界面） 用户和机器之间交流的方式。[es]

Visualization（可视化） 参见数据可视化。[dm]

Weight（权重） 在 ANN 中与两个神经元的连接有关的值。这个值决定了连接的强度，揭示了一个神经元有多少输出被传送给了另一个神经元的输入。参见突触权重。[nn]

WHEN CHANGED method（WHEN CHANGED 方法） 在基于框架的专家系统中，与框架的槽有关的过程。当新的信息被放入槽中时，WHEN CHANGED 方法被执行。[es]

WHEN NEEDED method（WHEN NEEDED 方法） 在基于框架的专家系统中，与框架的槽有关的过程。当解决问题所需的信息被放入槽中时，WHEN NEEDED 方法被执行，但是槽的值
450 不会被指定。[es]

人工智能工具和经销商

Expert system shells（专家系统框架）

ACQUIRE

一个用于获取知识和开发专家系统的工具。知识通过产生式规则和基于模式的动作表来表示。ACQUIRE 在建立专家系统时不需要进行特殊的训练。领域专家不需要知识工程师的帮助就能够创建一个知识库并且开发应用程序。

Acquired Intelligence Inc.
205 - 1095 McKenzie Avenue
Victoria, BC, Canada, V8P 2L5
电话：+1 (250) 479-8646
传真：+1 (250) 479-0764

http://www.aiinc.ca/acquire/acquire.shtml

Exsys Corvid

一种基于 Java 的通用专家系统构建工具。通过构建逻辑图表和规则来描述决策支持步骤。Exsys Corvid 可以通过 Corvid Servlet 运行时程序（能够为用户接口动态构建 HTML 页面）在网络上发送，或者通过 Java Applet 运行时程序在客户端展示。系统可以通过标准的 ODBC/JDBC 接口和 SQL 命令访问数据库。Exsys Corvid 也可以作为独立的应用运行，可以被集成到其他信息结构中。

EXSYS
6301 Indian School Rd. NE, Suite 700
Albuquerque, NM 87110, USA
电话：+1 (505) 888-9494
传真：+1 (505) 888-9509

http://www.exsys.com/

FICO Blaze Advisor

一种用于开发基于规则的面向对象的专家系统的成熟工具。可以采用多种方法创建和管理规则，包括：决策树、记分卡、决策表、公式构建器、图形化决策流和用户模板。Blaze Advisor 监控从用户定义事件中捕获的商务性能。该系统有一个开放的结构，能够容易地和其他计算环境集成，并能够接收来自多种数据库、XML 文档、Java 对象、.NET/COM 对象和 COBOL copybooks 的输入。

Fair Isaac Corporation
901 Marquette Avenue, Suite 3200
Minneapolis, MN 55402, USA
电话：+1 (612) 758-5200
电话：+1 (415) 472-2211
传真：+1 (612) 758 5201

http://www.fico.com/en/Products/Pages/default.aspx

Flex

一种基于框架的专家系统工具包。支持带有继承的基于框架的推理、基于规则的编程和数

据驱动的过程。Flex 有类似英语的自己的知识规格语言（KSL）。Flex 的主要结构是用于组织对象的框架和带有槽的实例，用于存放数据的默认值和当前值，为槽值增加功能的守护程序和限制，表达知识和专门技术的规则和联系，定义必要的过程的函数和动作，以及对终端用户相交互的问题的解答。KSL 支持数学表达式、布尔表达式和条件表达式。

Logic Programming Associates Ltd
Studio 30, Royal Victoria Patriotic Building
Trinity Road, London SW18 3SX, UK
电话：+44 (0)20 8871-2016
传真：+44 (0)20 8874-0449

http://www.lpa.co.uk/flx.htm

G2 Rule Engine Platform

一种用于开发和在线展示智能系统的面向对象的图形化环境。对象被组织为包含多继承的层次化类。开发人员可以通过图形化的表示和连接对象来建模一个应用。专家知识通过规则表示。G2 可以有效地实时运行，事件具有时间戳标记。G2 可以在 Microsoft Windows、Red Hat Linux、Linux Enterprise，Sun Solaris SPARC、IBM AIX、HP－UX 和 Compaq Tru64 UNIX 上运行。

Gensym Corporation
6011 West Courtyard Drive, Suite 300
Austin, TX 78730, USA
电话：+1 (512) 377-9700
e-mail: info@gensym.com

http://www.gensym.com

Intellix

一种结合了神经网络和专家系统技术而开发的综合性工具。这个工具提供了一个不需要任何编程技巧的友好的用户环境。领域知识由产生式规则和例子来表示。这个系统使用了神经网络中的模式匹配和规则解释相结合的技术，它能够进行实时学习。

452

Intellix Denmark
Nikolaj Plads 32, 2
DK-1067 Copenhagen K, Denmark
电话：+45 3314-8100
传真：+45 3314-8130
e-mail: info@intellix.com

http://www.intellix.com/products/designer/designer.html

JESS

JESS（Java Expert System Shell）是能够从 Sandia 国家实验室免费下载的工具（包括其完整的 Java 源代码）。JESS 是从 CLIPS（C Language Integrated Production System）中得到启发的，但是却发展成了一个拥有自己特点的完整工具。JESS 语言与 CLIPS 兼容，JESS 脚本和 CLIPS 脚本互相通用。JESS 为 CLIPS 增加了许多特性，包括后向链接以及操作和直接推理 Java 对象的能力。尽管用 Java 实现，但是 JESS 比 CLIPS 的运行速度要快。

Sandia National Laboratories, California
PO Box 969
Livermore, CA 94551-0969, USA
e-mail: ejfried@sandia.gov

http://herzberg.ca.sandia.gov/jess

Vanguard knowledge Automation System

一款基于互联网集成应用的综合工具。它包括 Vanguard 工作站、Vanguard 服务器、互联网

开发插件和内容管理系统（CMS）。CMS 是一款供普通人使用的基于互联网专家系统的开发工具。它使用可视化编辑工具和可视化逻辑展示，帮助商务人士通过互联网浏览器将它们的专业知识和合作自动化。Vanguard 工作站使用互联网开发插件，允许用户通过网络环境控制模型的样式和行为。

Vanguard Software Corporation
1100 Crescent Green
Cary, NC 27518, USA
电话：+1 (919) 859-4101
传真：+1 (919) 851-9457
e-mail: info@vanguardsw.com

http://www.vanguardsw.com/products/knowledge-automation-system/

VisiRule

一款通过绘制决策逻辑来开发和发布商务规则系统的基于规则的图形化工具。商务逻辑通过一组图标和链接来定义和表示。后台逻辑推理引擎具有高度的一致性和正确性。VisiRule 非常适用于构建普通的合规系统、金融决策支持系统和验证系统。它支持多链接图。VisiRule 图可以 453
在 Windows 环境中发布，或通过 LPA 智能服务工具包内嵌在 Java、VB、Delphi、C/C ++ 、. NET 和 C#中，也可以发布到互联网。

Logic Programming Associates Ltd
Studio 30, Royal Victoria Patriotic Building
Trinity Road, London SW18 3SX, UK
电话：+44 (0)20 8871-2016
传真：+44 (0)20 8874-0449

http://www.lpa.co.uk/

Visual Rules

Visual Rules 是基于图形化建模方法。它在统一建模语言（UML）中补充了适用于商务人士的商务逻辑。能够自动产生标准代码。

Innovations
Software Technology Corp.
161 N. Clark Street
Chicago, Illinois 60601, USA
电话：+1 (312) 523-2176
传真：+1 (312) 268-6286
e-mail: info@innovations-software.com

http://www.innovations-software.com

XMaster

这个系统由两个基本的包组成：XMaster Developer 和 XMaster User。有了 XMaster Developer，用户就可以很容易地通过建立可能的假设列表和证据项目列表来创建知识库。接着建立证据项和假设之间的关系。XMaster 也能使用户将不确定的或近似的关系整合到知识库中。它使用贝叶斯推理来管理不确定性。

Chris Naylor Research Limited
14 Castle Gardens
Scarborough, North Yorkshire
YO11 1QU, UK
电话：+44 (1)723 354590
e-mail: ChrisNaylor@ChrisNaylor.co.uk

http://www.chrisnaylor.co.uk/

XpertRule

一种用来开发基于规则的专家系统的工具。领域知识通过决策树、例子、真值表和异常树来表示。决策树是表示知识的主要方法。例子将结果与属性关联起来。真值表是例子的延伸，它表示涵盖了各种可能组合的例子集。例如，真值树和异常树，XpertRule 能自动生成决策树。Xpert-
454 tRule 也运用了模糊推理，能够集成明确推理和 GA 优化。

XpertRule Software
Innovation Forum, Innovation Park
51 Frederick Road
Salford M6 6FP, UK
电话：+44 (0)870 60-60-870
传真：+44 (0)870 60-40-156
e-mail: info@xpertrule.com

http://www.xpertrule.com/

Fuzzy logic tools（模糊逻辑工具）

FLINT

FLINT（Fuzzy Logic INferencing Toolkit）是一个通用的模糊逻辑推理系统，它使模糊规则可用于复杂的编程环境中。FLINT 支持模糊变量、模糊限定符和模糊修饰符（语言模糊限制语）概念。它用简单的、完整的语法来表达模糊规则。而且，它们被放入矩阵中，这通常叫做模糊关联记忆（FAM）。FLINT 为在所有支持 LPA 的硬件和软件平台上构建模糊专家系统和制定决策应用的程序员提供一个综合的工具集。

Logic Programming Associates Ltd
Studio 30, Royal Victoria Patriotic Building
Trinity Road, London SW18 3SX, UK
电话：+44 (0)20 8871-2016
传真：+44 (0)20 8874-0449
e-mail: tech_team@lpa.co.uk

http://www.lpa.co.uk/fln.htm

Fuzzy Control Manager

Fuzzy Control Manager（FCM）提供了允许用户在开发、调试和优化模糊系统时显示相关数据的图形用户界面（GUI），还提供了可以单击规则的编辑器和隶属函数的图形化编辑器。它使用户在 C 编译器中生成源代码或二进制代码。

TransferTech GmbH
Eichendorffstr. 1
D-38110 Braunschweig, Germany
电话：+49 5307-490-9160
传真：+49 5307-490-9161
e-mail: info@transfertech.de

http://www.transfertech.de/wwwe/fcm/fcme_gen.htm

Fuzzy Query

Fuzzy Query 是一个基于 Win32 的应用程序。它让用户使用强大并且语义灵活的 SQL 来查询数据库，SQL 是从数据库检索信息最流行的方法。它提供的信息不受布尔逻辑的严密制约。用户
455 不仅能看见最符合标准的选项，而且也能观察到不能遗漏的选项。Fuzzy Query 返回的每一个记录都显示按照符合特定标准的隶属度来排序的数据。

Fuzzy Systems Solutions
Sonalysts Inc.
215 Parkway North
Waterford, CT 06385, USA
电话：+1 (860) 526-8091
传真：+1 (860) 447-8883
e-mail: FuzzyQuery@Sonalysts.com

http://fuzzy.sonalysts.com/

FuzzyJ Toolkit

FuzzyJ Toolkit 是一个 Java 类的集合，它提供了处理模糊推理的能力。对于在 Java 环境中开发模糊逻辑，它是非常有用的。它是根据早期扩展 CLIPS 专家系统框架得到 Fuzzy CLIPS 的经验而得到的。它可以单独地用来创建模糊规则和进行推理，也可以和 JESS（来自 Sandia 国家实验室的 Java 专家系统框架）一起使用。FuzzyJ 是可以免费下载的。

Integrated Reasoning Group
NRC Institute for Information Technology
1200 Montreal Road, Building M-50
Ottawa, ON, Canada, K1A 0R6
电话：+1 (613) 993-9101
传真：+1 (613) 952-9907
e-mail: info@nrc-cnrc.gc.ca

http://www.nrc-cnrc.gc.ca/eng/ibp/iit/past-projects/fuzzyj-toolkit.html

FuzzyTECH

FuzzyTECH 是先进的用户模糊逻辑和神经模糊解决方案的软件开发工具族。它提供了两个基本的产品：用于技术应用的 Editions 和用于财务与商业应用的 Business。树形查看方式能让用户以结构化方式访问正在设计的模糊逻辑系统的所有组件，就像 Windows Explorer 能够让用户浏览他们的个人计算机结构那样。Editor 和 Analyser 窗口允许以图形化的方式设计模糊系统的每一个组件。

Inform Software Corporation
525 West Monroe Street, Suite 2360
Chicago, IL 60661, USA
电话：+1 (312) 575-0578
传真：+1 (312) 575-0581
e-mail: office@informusa.com

INFORM GmbH
Pascalstrasse 23
D-52076 Aachen, Germany
电话：+49 2408-9456-5000
传真：+49 2408-9456-5001
e-mail: hotline@inform-ac.com

http://www.fuzzytech.com/

jFuzzyLogic

用 Java 编写的一款模糊逻辑工具。它采用模糊逻辑语言（FCL）规范（IEC 1131p7）。参数优化算法包括 Derivate、梯度下降和 Jump。隶属函数可以使用连续的隶属函数（GenBell、Sigmoidal、Trapetzoidal、Gaussian、PieceWiseLinear、Triangular、Cosing 和 Dsigm）、离散的隶属函数 456
（Singleton、GenericSingleton）和用户自定义函数。jFuzzyLogic 可以免费下载。

Pablo Cingolani
e-mail: pcingola@users.sourceforge.net

http://jfuzzylogic.sourceforge.net/html/index.html

Mathematica Fuzzy Logic Package

这个包描述了内建函数，这些函数为定义输入和输出、创建模糊集、操纵和合并模糊集及关系、运用模糊推理函数以及整合逆模糊化路径提供了方便。有经验的模糊逻辑设计者发现用这个包来研究、模拟、测试和可视化非常复杂的系统是很简答的。Fuzzy Logic 要求 Mathematica

5.0－5.2，能用于 Windows、Mac OS、Linux x86 和 san Solaris 平台。

Wolfram Research, Inc.
100 Trade Center Drive
Champaign, IL 61820-7237, USA
电话：+1 (217) 398-0700
传真：+1 (217) 398-0747

http://www.wolfram.com/products/applications/fuzzylogic/

MATLAB Fuzzy Logic Toolbox

具有简单的单击界面，能引导用户完成模糊设计的每个步骤（从设定到诊断）。它以最新的模糊逻辑方法（如模糊聚类和自适应模糊学习）提供了内嵌支持。工具箱的交互式图形界面能让用户看到并精确调整系统行为。

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098, USA
电话：+1 (508) 647-7000
传真：+1 (508) 647-7001

http://www.mathworks.com/products/fuzzylogic/

rFLASH

rFLASH（Rigel 的 Fuzzy Logic Application Software Helper）是一种代码生成器，它用 MCS－51 汇编语言生成一套子程序和表来实现模糊逻辑控制（FLC）应用程序。它生成的代码可以在 8051 系列微处理器上运行。rFLASH 软件包括一个代码生成器和一个仿真器。作为一个代码生成器，它可以直接从一个高级控制任务描述文件（CTDF）中创建 FLC 代码。而作为一个仿真器，它可以用给定的输入在 PC 上产生输出。这种仿真器可以测试集中输入和定义好的术语或定义好的规则。

Rigel Corporation
PO Box 90040
Gainesville, FL 32607, USA
电话：+1 (352) 384-3766
e-mail: techsupport@rigelcorp.com

457 http://www.rigelcorp.com/flash.htm

Right Rule

一个实现模糊逻辑推理引擎的独立库。该库可以以 C ++ 类库的形式嵌入系统，以 C#模块实现桌面/服务器端应用。模糊逻辑推理引擎被设计为微软 . NET Micro 框架，力图在较短的开发周期内灵活地嵌入控制系统。

BLUEdev Ltd
Ap. Melaxrinou 24
Patras 26442, Greece
电话：+30 2610-4546-00
传真：+30 2610-4546-01
e-mail: support@bluedev.eu

http://www.bluedev.eu/t2c/

Neural network tools（神经网络工具）

BrainMaker

这是一个用于商业和市场预测，股票、期货、商品的销售预测、模式识别以及医疗诊断等任何用户需要专业分析的领域的神经网络软件。用户不需要任何特殊的编程或计算机技巧。

BrainMaker 使用反向传播算法，可以在 Mac 和 PC 上运行。

California Scientific
4011 Seaport
West Sacramento, CA 95691, USA
电话：+1 (916) 372-6800
USA toll free: 1-800-284-8112
e-mail: sales@calsci.com

http://www.calsci.com/BrainIndex.html

EasyNN – plus

EasyNN – plus 是一个用于 Microsoft Windows 的神经网络软件系统。它能巩固利用导入的文件生成多层神经网络。数值数据、文本或者图像可以用来创建神经网络。神经网络可以被训练、验证和查询。所有神经网络生成的或使用的图标、图像和输入/输出数据都可以被显示出来。图像、表格和网络图表都会及时地更新，因此用户可以看到它们是怎么工作的。神经网络可以用来对数据进行分析、预报、预测、分类和时间序列投影。

Neural Planner Software Ltd
18 Seymour Road
Cheadle Hulme
Cheshire, SK8 6LR, UK
电话：+44 7817-680-689
传真：+44 8700-518-079
e-mail: support@easynn.com

http://www.easynn.com

458

G2 NeurOn – line

G2 NeruOn – line 构建在 Gensym 的 G2 实时规则引擎平台，其结合神经网络和基于规则的技术来实时管理变量。它的神经网络模型像软传感器一样工作，分析在线操作数据和预测、估计重要的过程变量。神经网络可以分析历史数据和实时数据。G2 NeurOn – line 在预测时充分考虑变量，这些决策能够为过程生成有效的设置。

Gensym Corporation
6011 West Courtyard Drive, Suite 300
Austin, TX 78730, USA
电话：+1 (512) 377-9700
e-mail: info@gensym.com

http://www.gensym.com

MATLAB Neural Network Toolbox

MATLAB Neural Network Toolbox 是 MATLAB 中一个完整的神经网络工作环境。它是一个模块化的、开放的、可扩展的设计，对许多已经证明的网络范例提供了全面的支持，如有反向传播学习功能的多层传感器、循环网络、竞争层和自组织映射等。这个工具有用于设计和管理网络的图形化用户界面。

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098, USA
电话：+1 (508) 647-7000
传真：+1 (508) 647-7001

http://www.mathworks.com/products/neuralnet/

NeuNet Pro

一款使用神经网络实现模式识别、数据挖掘、建模和预测的通用软件工具。它功能强大、简

单易用，具有图形化开发环境。NeuNet Pro 可以直接从 MDB 数据库文件中读取数据，最多可包含 255 个数据段。它可以提供包含混淆矩阵、散列图和时序图的综合图形化报告。

CorMac Technologies Inc.
28 North Cumberland Street
Thunder Bay, ON, Canada, P7A 4K9
电话：+1 (807) 345-7114
传真：+1 (613) 822-5625
e-mail: douglas@cormactech.com

http://www.cormactech.com/

NeuralWorks Predict

NeuralWorks Predict 是一个集成的、先进的工具，它可用于创建、部署预测和分类应用程序。Predict 把神经网络技术和一般的算法、统计学和模糊逻辑结合在一起，为各种问题自动寻找最
459 优或者接近最优的解决方案。Predict 不需要任何神经网络方面的先验知识。对于高级用户，Predict 还允许他们直接访问所有关键的训练和网络参数。在 Microsoft Windows 环境中，Predict 既可以作为 Microsoft Excel 插件以利用 Excel 的数据处理能力，也可以作为一个命令行程序运行。在 UNIX 和 Linux 环境中，Predict 作为一个命令行程序。

NeuralWare
230 East Main Street, Suite 200
Carnegie, PA 15106-2700, USA
电话：+1 (412) 278-6280
传真：+1 (412) 278-6289
e-mail: info@neuralware.com

http://www.neuralware.com/products.jsp

NeuroCoM

NeuroCoM（Neuro Control Manager）是一个用于开发和测试神经网络的高性能工具。它是一个面向窗口的图形化用户界面，它对于神经网络训练及其分析都非常便利。这个界面有助于神经网络体系结构、传递函数和学习过程的可视化。NeuroCoM 能够用 C 语言生成源代码。

TransferTech GmbH
Eichendorffstr 1
D-38110 Braunschweig, Germany
电话：+49 (05307) 490-91-60
传真：+49 (05307) 490-91-61
电话：+49 (531) 890-255
传真：+49 (531) 890-355
e-mail: info@transfertech.de

http://www.transfertech.de/wwwe/ncm/ncme_scr.htm

NeuroModel

NeuroModel 生成的模型能够充分适应制造业的需要。模型从制造性数据中产生，并用来分析制造性数据。NeuroModel 能够处理 255 个连续变量，因此可对带有几百个变量复杂的过程进行可视化和建模。

atlan-tec Systems GmbH
CEO: Dipl.-Ing. Thomas Froese
Hanns-Martin-Schleyer-Str. 18 a
Haus 3/ 1. Obergeschoss
47877 Willich-Münchheide II, Germany
电话：+49 2154-9248-0
传真：+49 2154-9248-100
e-mail: info@atlan-tec.com

http://www.atlan-tec.com/neuromodel_en.htm

NeuroShell 2

NeuroShell 2 有功能强大的神经网络架构，它的用户界面采用的是 Microsoft Windows 的图标，有丰富的工具和流行的选项，给用户提供了基础的神经网络实验环境。它适用于从事理论研究 460
的用户或者那些对典型神经网络范例（如反向传播）感兴趣的用户。对解决实际问题感兴趣的用户应该考虑 NeuroShell Predictor、NeuroShell Classifier 及 GeneHunter。

Ward Systems Group, Inc.
Executive Park West
5 Hillcrest Drive
Frederick, MD 21703, USA
电话：+1 (301) 662-7950
传真：+1 (301) 663-9920
e-mail: sales@wardsystems.com

http://www.wardsystems.com/neuroshell2.asp

NeuroSolutions

这个模块将模块化的基于图标的网络设计界面和学习过程的实现（如循环反向传播和通过时间反向传播）相结合。其他特征包括图形用户界面和 C ++ 源代码的生成。NeuroSolutions 有 3 个级别：Educator（面向要学习神经网络的初学者）、User（通过许多用于静态模式识别应用的神经模型扩展了 Educator 级别）、Consultants（为动态模式识别、时间序列预测和进程控制提供增强的模型）。

NeuroDimension, Inc.
3701 NW 40th Terrace, Suite 1
Gainesville, FL 32606, USA
电话：+1 (352) 377-5144
USA toll free: 1-800-634-3327
传真：+1 (352) 377-9009
e-mail: info@nd.com

http://www.neurosolutions.com/products/ns

STATISTICA Automated Neural Networks

STATISTICA Automated Neural Networks（SANN）是一款市场上广泛应用性能最好的神经网络产品。它一步一步指导用户建立神经网络，包括数据选择、标称值编码、比例设置、正规化和缺失值替换。SANN 能够集成不同的网络和网络结构。它能够在 Microsoft Excel 表单中自动运行神经网络分析，或者在用户用 C、C ++ 、C#、Java 等语言开发的应用中集成神经网络过程。

StatSoft, Inc.
2300 East 14th Street
Tulsa, OK 74104, USA
电话：+1 (918) 749-1119
传真：+1 (918) 749-2217
e-mail: info@statsoft.com

http://www.statsoft.com/products/statistica-automated-neural-networks 461

Trajan 6. 0 Professional

一款可被用于数据挖掘、推理、建模和预测的成熟的神经网络仿真工具包。Trajan 的一个主要特征是智能问题解决器，能够流水线式地处理整个神经设计过程。智能问题解决器能够完成模型类型和复杂度的选择、变量选择和训练过程选择。Trajan 支持多层感知器、Kohonen 网络、Radial 基本函数、线性模型、概率和一般化的回归神经网络。Trajan 算法包括反向传播、Levenburg - Marquardt 和共轭梯度下降算法。还包括错误标定、自动交叉检验和一组停止条件。

Trajan Software Ltd
The Old Rectory, Low Toynton,
Horncastle, Lincs, LN9 6JU, UK
电话：+44 1507 524-077
传真：+44 8700 515-931
e-mail: sales@trajan-software.demon.co.uk

http://www.trajan-software.demon.co.uk/

Evolutionary computation tools（进化计算工具）

Evolutionary Optimizer

Evolutionary Optimizer（EVO）是一个优化系统特性的基本工具，其中系统特性由数值参数确定。这个系统提供了图形化的友好的用户界面，它不需要任何编程知识。

TransferTech GmbH
Eichendorffstr. 1
D-38110 Braunschweig, Germany
电话：+49 5307-490-9160
传真：+49 5307-490-9161
e-mail: info@transfertech.de

http://www.transfertech.de/wwwe/evo/evoe_gen.htm

Evolver

Evolver 是 Microsoft Excel 的一个优化插件。Evolver 使用遗传算法来快速地解决金融、分配、计划安排、资源分配、制造、预算、工程等领域内复杂的优化问题。事实上，任何一个可以用 Excel 建模的问题，Evolver 都可以解决。它不需要任何编程或遗传算法理论方面的知识，它带有解释详尽的参考手册、几个例子，以及免费的、无限的技术支持。

Palisade Corporation
798 Cascadilla Street
Ithaca, NY 14850, USA
电话：+1 (607) 277-8000
462 USA/Canada toll free: 1-800-432-7475
传真：+1 (607) 277-8001
e-mail: sales@palisade.com

http://www.palisade.com/evolver/

GEATbx

GEATbx（Genetic and Evolutionary Algorithm Toolbox）和 MATLAB 的结合使用是进化算法在 MATLAB 上的综合的实现。大量的操作被完全嵌入到这个环境中，这个环境构成一个可适用于解决各种问题的功能强大的优化工具。

Hartmut Pohlheim
Charlottenstr. 23, 13156 Berlin, Germany
电话：+49 700 7645-4346
传真：+49 700 7645-4346
e-mail: support@geatbx.com

http://www.geatbx.com/

GeneHunter

它是解决优化问题的功能强大的解决方案。它包括一个 Excel 插件（该插件允许用户在 Excel 电子表格中运行优化问题）以及遗传算法函数的动态链接库（它可以从 Microsoft Visual Basic 或者 C 等程序语言来调用）。

Ward Systems Group, Inc.
Executive Park West
5 Hillcrest Drive
Frederick, MD 21703, USA
电话：+1 (301) 662-7950
传真：+1 (301) 663-9920
e-mail: sales@wardsystems.com

http://www.wardsystems.com/genehunter.asp

GenSheet

将遗传算法实现为快速用 C 编码的动态链接库。GenSheet 支持用于二进制、整数、实数和置换表示的遗传运算，还包括用于限制的非线性优化、遗传分类器、工作时间安排和计算最小方差组合的特殊命令。GenSheet 需要使用 Microsoft Excel 软件。所有的 GenSheet 命令都被分配在容易使用的 Excel 菜单栏中。GenSheet 提供了交互式的帮助和教程。

Inductive Solutions, Inc.
380 Rector Place, Suite 4A
New York, NY 10280, USA
电话：+1 (212) 945-0630
传真：+1 (212) 945-0367
e-mail: roy@inductive.com

http://www.inductive.com/softgen.htm

463

MATLAB Global Optimization Toolbox

一款解决多目标最大化和最小化全局优化问题的工具包。可以通过调整初始人口和适应度比例或定义亲代选择、交叉和变异函数来创建用户自定义的遗传算法。通过定义投票、查找和其他函数来创建用户自定义模式搜索方法。该工具包括设计和管理遗传算法提供了图形化用户接口。

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098, USA
电话：+1 (508) 647-7000
传真：+1 (508) 647-7001

http://www.mathworks.com/products/global-optimization/

OptiGen Library

OptiGen 库是一个面向对象的程序接口。它提供单/多对象遗传算法、增强的属性选择算法和其他优化策略（例如模拟退火）的通用面向对象接口。OptiGen 库支持所有的 .NET 语言、Visual Studio C ++ 和 ActiveX（COM）语言，包括 Microsoft Office 产品中的 VBA。

NeuroDimension, Inc.
3701 NW 40th Terrace, Suite 1
Gainesville, FL 32606, USA
电话：+1 (352) 377-5144
USA toll free: 1-800-634-3327
传真：+1 (352) 377-9009
e-mail: info@nd.com

http://www.nd.com/genetic/

xl bit

一款用于 Microsoft Excel 的遗传算法插件。它不需要任何编程知识。系统运行最多 500 个染色体、多适应度函数、四类变量（二元、整数、非整数和排列）、3 种交叉方法，并提供图形化

用户接口。

XLPert Enterprise
e-mail: sales@xlpert.com

http://www.xlpert.com/

Data mining（数据挖掘）

Angoss KnowledgeSEEKER

一款功能强大的数据挖掘工具，提供数据剖析、增强的数据可视化和决策树功能。它被广泛应用于市场、销售和风险分析。KnowledgeSEEKER 能够接受来自几乎所有数据源的数据：统计文
464 件、文件服务器和 SAS、SPSS、ODBC 等驱动的数据库。它可以在数据仓库中进行运算，避免了数据在软件间移动造成的反应时间下降。

Angoss Software Corporation
111 George Street, Suite 200
Toronto, ON, Canada, M5A 2N4
电话：+1 (416) 593-1122
传真：+1 (416) 593-5077

http://www.angoss.com/

BLIASoft Knowledge Discovery

BLIASoft Knowledge Discovery 是一个决策支持工具。它从数据中提取模型。训练数据可以是定量或定性数据。BLIASoft Knowledge Discovery 可以处理不完整、不确定甚至矛盾数据。它使用简便，不需要任何特殊技能。

BLIA Solutions
1, rue du Pont Guilhemery
31 000 Toulouse, France
e-mail: support@bliasolutions.com

http://www.bliasoft.com

FICO Model Builder for Decision Trees

一款图形化开发工具，能够结合人类的专业知识进行数据驱动分析，迅速创建决策树。用户可以根据性能测试来选择能实现最有效划分的变量。它能够迅速比较策略，无需额外编写代码。

Fair Isaac Corporation
901 Marquette Avenue, Suite 3200
Minneapolis, MN 55402, USA
电话：+1 (612) 758 5200
传真：+1 (612) 758 5201

http://www.fico.com/en/Products/Pages/default.aspx

IBM DB2 Data Warehouse Edition

DB2 数据仓库编辑器（DWE）是一组商业智能平台产品集。用户可以构建完整的数据仓库解决方案，包括高可扩展的关系数据库、数据访问能力、商业智能分析和前端分析工具。DB2 数据仓库编辑器包括如下产品：DB2 通用数据库工作组服务器、DB2 Cube 视图、DB2 智能挖掘器和 DB2 Office 连接器。

IBM Corporation
1 New Orchard Road
Armonk, NY 10504-1722, USA
电话：+1 877-426-6006
传真：+1 800-314-1092

465 http://www-01.ibm.com/software/data/iminer/

Investigator II

Investigator II 为商业性能分析、预测和监控提供了简单易用的工具。它将数据转换为支持决策的报告和图表。它允许用户监控、报告和比较关于产品和市场任何方面的组合，并对过去、现在和未来的性能进行分析和预测。Investigator II 支持 Bayesian 分类器、基于事例的推理、决策树、C – means 聚类、数据混合、K means 聚类、Kohonen 聚类、神经网络和主成分分析。

Solutions4Planning Ltd
Garrick House
138 London Road
Gloucester GL1 3PL, UK
电话：+44 1276-485-711

http://www.solutions4planning.com/investigator-2.html

KXEN Analytic Framework

为分类、回归、属性重要性、聚类、预测和购物篮分析提供解决方案。它发现个体集的自然组或行为，提供相应的每个分组的描述，生成事物数据的规则模式以发现事件的关联性，使用时间序列数据预测下一阶段的结果来发现有意义的模式和趋势。不同于传统的数据挖掘工具，KXEN Analytic Framework 能够处理高维的输入属性（超过 10000）。

KXEN, Inc.
201 Mission Street, Suite 1950
San Francisco, CA 94105-1831, USA
电话：+1 (415) 904-4160
传真：+1 (415) 904-9041
e-mail: sales-us@kxen.com

http://www.kxen.com/

Oracle Data Mining

Oracle Data Mining（ODM）是可用于 Oracle 数据库 11g 企业版的收费工具。它为用户提供构建和发布预测分析的应用工具。应用可以使用 SQL 和 Java API 建立，自动挖掘 Oracle 数据并实时展示结果。因为数据、模型和结果都存储在 Oracle 数据库中，数据不用移动，安全性得到了最大保障，并能够最小化信息延迟。ODM 模型可以嵌入 SQL 查询和应用中。ODM 可以实现分类、回归分析、异常点发现、聚类、购物篮分析和特征提取。

Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065, USA
电话：+1 (650) 506-7000 or +1 800-392-2999
e-mail: oraclesales_us@oracle.com

http://www.oracle.com/technology/products/bi/odm/index.html 466

Partek Discovery Suite

Partek Discovery Suite 集成了数据分析方法和可视化功能，帮助统计研究者迅速实现模式分析和识别。它对数据的行和列没有限制，因此能够很容易实现分析、编辑和处理大数据集。同时能够有效地为可视化、分析和建模提供数据降维功能。

Partek Incorporated
12747 Olive Blvd., Suite 205
St. Louis, MO 63141, USA
电话：+1 (314) 878-2329
传真：+1 (314) 275-8453
e-mail: inquire@partek.com

http://www.partek.com/partekds

Portrait Customer Analytics

一款简单易用的预测分析工具，通过增加高效的3维数据可视化功能扩展了现有的市场和分析环境。简单易用的接口将数据转换为可视化信息。Portrait Customer Analytics 为流失/损耗、活动响应、交叉销售/追加销售、风险/违约、用户生命值和收益率提供快速的自动建模。预设的建模技术包括决策树、回归、记分卡和聚类。

Portrait Software
The Smith Centre
The Fairmile
Henley-on-Thames
RG9 6AB, UK
电话：+44 1491-416-600
传真：+44 1491-416-601

http://www.portraitsoftware.com/

SAS Enterprise Miner

SAS 使用一组技术对数据进行收集、分类、分析和解释，实现对其中模式、异常、关键变量和关系的发现。SAS Enterprise Miner 通过一组成熟的预测和建模算法，包括购物篮分析、决策树、梯度 boosting、最小角回归样条、神经网络、线性和 logistic 回归、部分最小平方回归等。

SAS Institute Inc.
100 SAS Campus Drive
Cary, NC 27513-2414, USA
电话：+1 (919) 677-8000
传真：+1 (919) 677-4444

467

http://www.sas.com/technologies/analytics/datamining/miner/

TIBCO Spotfire Miner

数据挖掘和可视化编程工具，实现从收集的数据中发现模式、趋势和隐含关系。它为统计和商业分析提供简单易用的接口。Spotfire Miner 的可视化工作流图记录应用的每一步，使结果易于理解。Spotfire Miner 提供数据处理结点来支持数据挖掘过程的每一步，包括数据访问（文件、数据库）、开发和转换、数据清洗、建模、logistic 回归和线性回归、分类和回归树、神经网络、naive 贝叶斯、K 均值聚类、主成分分析、关联规则和 Cox 回归、模型评估、数据导出。

TIBCO Spotfire Main Office
212 Elm Street
Somerville, MA 02144, USA
电话：+1 (617) 702-1600
传真：+1 (617) 702-1700
e-mail: mds@tibco.com

http://spotfire.tibco.com/products/data-mining-applications.aspx

Vanguard Decision Tree

Vanguard Decision Tree 是一组集成应用，包括 Vanguard 工作室和决策树分析插件模块。帮助用户实现经典的决策树分析和 Markov 仿真。

Vanguard Software Corporation
1100 Crescent Green
Cary, NC 27518, USA
电话：+1 (919) 859-4101
传真：+1 (919) 851-9457
e-mail: info@vanguardsw.com

http://www.vanguardsw.com/products/decision-tree-suite/

Viscovery SOMine

Viscovery SOMine 是实现数据挖掘、可视化聚类分析、统计剖析、基于自组织映射的分类和工作流量环境中经典统计方法的一款桌面应用。它在 1996 年首次发布，Viscovery SOMine 成为基于 SOM 的标准商业数据挖掘工具。它为数据挖掘、类定义、剖析和分段、统计功能（Ward 聚类、组剖析、描述统计）提供建模工具。Viscovery SOMine 使用标准的软件，不需要特殊的用户化、安装、硬件框架和系统维护。

Viscovery Software GmbH
Kupelwiesergasse 27
A-1130 Vienna, Austria
电话：+43 1 532-05-70
传真：+43 1 532-05-70-33
e-mail: inquire@partek.com

http://www.eudaptics.co.at/ 468

VisiRex

可视化规则提取（VisiRex）是一款进行数据挖掘、数据预测和知识发现的通用软件工具，它从数据库中发现有用的模式和规则，生成综合的统计报告，构建有颜色显示的决策树（可调整数据剪枝），分割数据组成类簇。VisiRex 也允许用户使用部分数据构建预测模型，并使用其他数据测试模型。

CorMac Technologies Inc.
28 North Cumberland Street
Thunder Bay, ON, Canada, P7A 4K9
电话：+1 (807) 345-7114
传真：+1 (613) 822-5625
e-mail: douglas@cormactech.com

http://cormactech.com/visirex/ 469

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

accelerated learning（加速学习），185－188
accidental property（偶有属性），140
action（动作），参见 rule，consequent
action potential（动作电位），166
activation function（激活函数），169
  bell（钟形激活函数），278
  hard limit（硬限幅函数），169
  hyperbolic tangent（双曲正切），185
  linear（线性函数），169－170
  saturated linear（饱和线性函数），189
  sigmoid（S形函数），169－170，177
  sign（符号函数），169，189
  step（阶跃函数），169
activation level（激活水平），168
activity balance point（活动平衡点），202
activity product rule（活动积规则），201
  generalised（一般化的），202
adaptive learning rate（自适应学习速率），186－188
adaptive neuro-fuzzy inference system（自适应神经模糊推理系统），277，353－354，383－384
  architecture（结构），277－279
  defuzzificztion layer（逆模糊化层），279
  fuzzification layer（模糊化层），278
  input layer（输入层），278
  normalisation layer（归一化层），279
  rule layer（规则层），278－279
  summation neuron（总和神经元），279
  learning（学习），280－282
adaptive topology（自适应拓扑），220
Advice Taker（意见接受人），6
Affenzeller，M.，289
aggregate fuzzy set（聚合模糊集），110－111
aggregation（聚合），110，137
AI（人工智能）参见 artificial intelligence
AI "winter"（人工智能"胜者"），12
algebraic sum（代数和），参见 probabilistic OR
algorithm（算法），34
a-kind-of（一种），参见 is－a
AND（与），172
AND product（与乘积），109
Anderson C.，13
ANFIS（自适应的神经模糊推理系统），参见 adaptive neuro-fuzzy inference system
ANN（人工神经网络），参见 artificial neural network
antecedent（前项），参见 rule，antecedent
a-part-of（部分），137－8
approximate reasoning（近似推理），262－263
Apriori algorithm（Apriori 算法），412－418
Apriori principle（Apriori 原理），413
artificial intelligence（人工智能），2，6，260
  foundations（基础），5－6
  history（历史），5－17，19－20
  paradigm shift（范式转换），9
artificial neural network（人工神经网络），12－13，165，167
ASCII code（ASCII 码），304
Association（关联），137－138，410
association rule（关联规则），410－418
  confidence（置信度），411－412
  support（支持度），411－412
associative memory（关联记忆），187
associativity（关联性），100－101
attribute（属性），131
Automatic Computing Engine（自动计算引擎），2
axon（轴突），166

B

back-propagation（反向传播），参照 back-propagation algorithm
back-propagation algorithm（反向传播算法），13，179－180
backward chaining（后向链接），38－40
BAM（双向联想记忆），参见 bidirectional associative memory
Barto，A.，13
BASIC（BASIC 语言），30，32

Bayes, T., 60
Bayesian reasoning (贝叶斯推理), 61 - 65, 65 - 72
Bayesian rule (贝叶斯规则), 60 - 61, 73
bell activation function (钟形激活函数), 278
belongs-to (属于), 137 - 138
Berkeley Initiative in Soft Computing (伯克利软计算倡议组织), 20, 259
bidirectional associative memory (双向联想记忆), 196 - 200
  architecture (结构), 196 - 197
  convergence (收敛), 200
  stability (稳定性), 199
  storage capacity (存储能力), 200
  training algorithm (训练算法), 197 - 199
binary logic (二值逻辑), 89
bit map (位图), 323
Black, M., 88, 125
book-keeping facilities (记录设备), 33
Boole, G. 428
Boolean logic (布尔逻辑), 87 - 88
Boyce, R. 389, 448
Brain (大脑), 2, 4, 166, 187
Branch (分支), 401
break package (断点包), 33
Broomhead, D., 13
Bryson, A. (贝叶斯), 13
Buchanan, B. 9, 19, 83

C

C (C 语言), 30, 32, 121, 253, 283, 310
C ++ (C ++语言) 121, 253, 283
Cantor, G., 97
Castillo, O 289
CART (分类和回归树), 403
categorical data (分类数据), 330
Central Limit Theorem (中心极限定理), 2
centre of gravity (重心), 111 - 112
centroid technique (质心技术), 111
cerebral cortex (大脑皮层), 205
certainty factor (确信因子), 10, 74 - 75, 77 - 78, 291 - 293
certainty factors theory (确信因子理论), 74 - 80
Chamberlin, D., 389, 448
character recognition (符号识别), 323 - 328
characteristic function, 91
child chromosome (子代染色体), 222 - 223
child node (子结点), 401
chromosome (染色体), 221, 232, 237
  decoding (解码), 228
  encoding (编码), 221, 236 - 237, 344 - 345
  evaluation (估值), 221, 237 - 238, 346 - 347
  offspring (后代), 222 - 223
  parent (父母), 222 - 223
  population (人群), 224
    average fitness (平均健康), 224
    size (规模), 224, 239
class-frame (类框架), 134 - 135
classification (分类), 303, 312 - 317, 332 - 335
clipped membership function (剪切隶属函数、剪切隶属度), 110
clipping (剪切), 109, 110
CLIPS (CLIPS 语言), 311
Cloning (克隆), 226
cluster (聚类), 332, 336
  analysis (分析), 336 - 337
clustering (聚类), 303, 332, 336 - 343
Codd, T., 387, 397
COG (重心), 参见 centre of gravity
Colmerauer, A. 19
common sense (常识), 87
commutativity (交换性), 100
competitive learning (竞争学习), 209 - 212, 332 - 335
competitive learning rule (竞争学习规则), 207, 209
competitive neural network (竞争神经网络), 205, 332 - 335
complement (补), 98
concentration (集中), 95 - 96
conclusion (结论), 参见 rule、consequent
condition (条件), 参见 rule、contecedent
conditional probability (条件概率), 59
conflict resolution (冲突消解), 47 - 49
conflict set (冲突集), 47
conjunction (合取), 26, 77, 109
connectionist expert system (连通式专家系统), 参见 neural expert system
consequent (后项), 参见 rule、consequent
containment (控制), 98 - 99
continuous data (连续数据), 329 - 330
control (控制), 303
convergence (收敛), 183, 195, 200, 229
Cook, S., 19

correlation minimum（最小相关性），参见 clipping
correlation product（相关性产生式），参见 scaling
cortex（皮层），参见 cerebral cortex
covariance matrix（协方差矩阵），375 - 377
Cox，E.，20
crisp set（清晰集合），89 - 91
crossover（交叉），221，226，248 - 250，345 - 346
  probability（概率），224

D

Dartmouth College workshop（达特茅斯学院研讨会），6，19
Darwin，C.，219，220，430，436
data（数据），304 - 305，365 - 367
  categorical（分类），330
  continuous（连续），329 - 330
  discrete（离散），330
  incompatible（不相容的），304
  incomplete（不完整的），304
data acquisition（数据采集），304
database（数据库），31，386 - 387
  operational（操作），350
  relational（关系），387 - 388
data base management system（数据库管理系统），386 - 387
data cleaning（数据清洗），304，366 - 367
data consolidation（数据统一），参见 data transformation
data cube（数据立方体），393 - 398
  dimension（维度），394 - 395
  fact（事实），394 - 395
  starnet model，（星形网模型），394 - 395
data-driven reasoning（数据驱动推理），参见 forward chaining
data exploration（数据探索），368
data fusing（数据融合），366 - 367
data mining（数据挖掘），365 - 368
data selection（数据选择），366
data transformation（数据转换），367
data visualisation（数据可视化），369
data warehouse（数据仓库），391 - 393
Davis，L.，345
Davis's law（戴维斯法则），301
dBASE III，150，152
DBMS（数据库管理系统），参见 database management system
De Morgan's Laws（德·摩根定律），102 - 103
debugging aids（调试辅助程序），33
decision support system（决策支持系统），318 - 323，348 - 353
decision tree（决策树），401 - 403
  branch（分支），401
  child node（子结点），401
  dependent variable（因变量），401 - 402
  leaf（叶结点），401
  parent node（父结点），401
  root node（根结点），401
  split（分割），402，403 - 405
Deep Blue（深蓝），165
defuzzification（逆模糊化），111 - 112，273
  centroid technique（质心技术），111
  sum-product composition（和 - 积合成），273
defuzzification layer（逆模糊化层），273，279
degree of confidence（置信度），参见 certainty factor
degree of membership（隶属度），92
delta rule（delta 规则），172
  generalised（广义），185
demon（守护程序），133，142
DENDRAL，9 - 10，12，19，40，307
dendrite（树突），166
dependent variable（因变量），401 - 402
developer interface（开发者接口），32 - 33
  debugging aids（调试辅助程序），33
  input/output facilities（输入/输出工具），33
  knowledge base editor（知识库编辑器），32 - 33
diagnosis（诊断），303，308 - 312，348 - 353
dilation（扩张），95 - 96
discrete data（离散数据），330
disjunction（析取），26，77，107
distributivity（分布），101
domain expert（领域专家），参见 expert
dummy rule（伪规则），294

E

EBCDIC code（EBCDIC 编码），304
Edmonds，D.，5
eigenvalue（特征值），375，377 - 380
eigenvector（特征向量），375，377 - 380
Electronic Numerical Integrator and Calculator（电子数字积分器和计算器），5，19
EMYCIN，10，19
end-user（终端用户），参见 user

Enigma, 2
epoch (周期), 172, 183
error gradient (误差梯度), 178
essential property (实质属性), 140
Euclidean distance (欧几里得距离), 207 - 208, 334
evidential reasoning (论据推理), 74 - 80, 315 - 317
evolution (进化), 219 - 221
evolution strategy (进化策略), 14, 242 - 244
(1 + 1) - evolution strategy((1 + 1) - 进化策略), 242 - 244
evolutionary computation (进化计算), 14, 219
evolutionary fitness (进化适应), 220
evolutionary neural network (进化神经网络), 285 - 287
exclusive-OR (异), 172, 180 - 181, 184 - 185, 275 - 276
exhaustive search (穷举搜索), 51
expert (专家), 25, 29, 56, 67, 73, 87
expert system (专家系统), 8 - 12, 28, 33 - 35
frame-based (基于框架的), 149 - 161
fuzzy (模糊), 113 - 124, 317 - 323
neural (神经), 262 - 268
rule-based (基于规则的), 30 - 33, 41 - 47, 50 - 51, 308 - 317
expert system shell (专家系统框架), 28, 310 - 312
explanation facilities (解释工具), 32, 263
how (如何), 32
why (为什么), 32
external interface (外部接口), 32

F

facet (侧面), 133 - 134
inference facet (推理侧面), 134
prompt facet (提示侧面), 134
search order facet (查找侧面), 149
value facet (值侧面), 133
fact (事实), 31
FAM (模糊关联记忆), 参见 fuzzy associative memory
feature space (特征空间), 379
feature vector (特征向量), 379
feedback neural network (反馈神经网络), 188 - 189
feedforward neural network (前馈神经网络), 175
Feigenbaum E., 9, 10, 19
Feller, W., 57
Fine, T., 57
firing a rule (激发规则), 31, 36, 146
first-order Sugeno fuzzy model (一阶 Sugeno 模糊模型), 277
fit-vector (适应向量), 94
fitness (适应性), 221
fitness function (适应性函数), 222, 237 - 238, 346
Fogel, D., 20
forgetting factor (遗忘因子), 201 - 202, 386
FORTRAN (FORTRAN 语言), 6, 30, 32, 245, 253
forward chaining (前向链接), 37 - 38, 309
frame (框架), 131 - 132, 133 - 138, 140
class (类), 134 - 135
instance (实例), 134 - 135
frame-based expert system (基于框架的专家系统), 134 - 137, 137 - 138, 138 - 142, 146, 149 - 161
fully connected neural network (全连接神经网络), 176
fundamental memory (基础记忆), 192 - 193
fuzzification (模糊化), 107
fuzzification layer (模糊化层), 269, 278
fuzzy associative memory (模糊关联记忆), 118
fuzzy evolutionary system (模糊进化系统), 290 - 296
fuzzy expert system (模糊专家系统), 113 - 124, 317 - 322
fuzzy grid (模糊网格), 290 - 293
fuzzy inference (模糊推理), 106 - 113
Mamdani, 106 - 112, 113
Sugeno, 112 - 113, 114
Fuzzy Knowledge Builder (模糊知识构建器), 121
fuzzy logic (模糊逻辑), 15 - 17, 87 - 89
fuzzy reasoning (模糊推理), 104 - 105
fuzzy rule (模糊规则), 15, 103 - 106, 112
fuzzy rule layer (模糊规则层), 271 - 272, 278 - 279
fuzzy rule table (模糊规则表), 119, 291
multiple (多), 293
fuzzy set (模糊集), 89 - 94
clipping (剪切), 110
scaling (缩放), 110
fuzzy set theory (模糊集理论), 参见 fuzzy logic
fuzzy singleton (模糊单态模式), 112
fuzzy thinking (模糊思想), 87 - 89
fuzzy variable (模糊变量), 94 - 95

G

gain chart (收益图), 405

Galton, F., 370
gene (基因), 221, 237
General Problem Solver (通用问题解决器), 6, 19
generalisation (泛化), 137, 325, 329, 381 - 384
generalised activity product rule (通用活动积规则), 202
generalised delta rule (通用 delta 规则), 185
generation (代), 222
genetic algorithm (遗传算法), 14, 222 - 223, 295 - 296, 343 - 348
  convergence (收敛), 229
  performance (性能), 229 - 231, 346 - 347
genetic operator (遗传操作符), 221, 226, 345 - 346
  cloning (克隆), 226
  crossover (交叉), 221, 226, 248 - 250, 345 - 346
  mutation (变异), 221, 226, 250 - 251, 346
genetic programming (遗传编程), 14, 245 - 253
genetics (遗传学), 220
Gini, C., 403
Gini coefficient (基尼系数), 403
global optimum (全局优化), 229 - 230
goal (目标), 38, 146
goal-driven reasoning (目标驱动推理), 参见 backward chaining
Goldberg, D., 345
GPS (通用问题解决器), 参见 General Problem Solver
Gray coding (格雷编码), 229
grid-type fuzzy partition (栅格模糊划分), 290 - 291
Grossberg, S., 13

H

Hakel, M., 56
hard limiter (硬限幅函数), 169
Haykin, S., 19
Hebb, D., 200, 214, 436
Hebb's Law (Hebb 法则), 200 - 201
Hebbian learning (Hebbian 学习), 200 - 204
hedge (模糊限制语), 95 - 97
  a little (一点), 97
  extremely (极其), 96, 97
  indeed (的确), 96, 97
  more or less (或多或少), 96, 97
  slightly (稍稍), 97
  somewhat (稍微), 97
  very (非常), 96, 97
  very very (极), 96, 97
heuristic (启发式的), 28, 33
hidden layer (隐含层), 175 - 176
Ho, Y. - C., 13
Holland, J., 14, 20, 221, 226, 232, 254, 435
Hopfield, J., 13, 19, 189, 213
Hopfield network (Hopfield 网络), 188 - 196
  architecture (结构), 189
  convergence (收敛), 195
  fundamental memory (基础记忆), 192 - 193, 196
  storage capacity (存储能力), 195 - 196
  training algorithm (训练算法), 193 - 194
Human Genome Project (人类基因组计划), 365
human expert (人类专家), 参见 expert
hybrid intelligent system (混合智能系统), 259 - 260, 348 - 357
hyperbolic tangent (双曲正切), 185

I

idempotency (等幂), 101 - 102
identity (同一性), 102
IF-THEN rule (IF-THEN 规则), 参见 production rule
Inmon, W. H., 392
inference chain (推理链), 36 - 37
inference engine (推理引擎), 31, 146 - 147, 262
inheritance (继承), 134, 138 - 142
  multiple (多), 140 - 141
  one-parent (单亲), 139 - 140
input layer (输入层), 175, 269, 278
input/output facilities (输入/输出工具), 33
instance-frame (例子 - 框架), 134 - 135
intelligence (智能), 1 - 2, 219
intelligent behaviour test (智能行为测试), 参见 Turing test
intelligent machine (智能机), 4, 18, 165
  character recognition (符号识别), 323 - 328
  classification (分类), 303, 312 - 317, 332 - 335
  clustering (聚类), 303, 332, 336 - 343
  control (控制), 303
  decision support (决策支持), 318 - 323, 348 - 353
  diagnosis (诊断), 303, 308 - 312, 348 - 353
  optimisation (优化), 303, 344 - 348
  prediction (预测), 303, 328 - 332, 353 - 357

selection（选择），303
troubleshooting（解决问题），308－312
intensification（激烈化），96
intersection（交），99，109
inversion（转置），346
involution（回转），102
is-a，136，137－138
iteration（迭代），172，183，222

J

Jacobs，R.，185
Jang，R.，277，282
Java，310
joint probability（联合概率），59

K

Karp，R.，19
Kasparov，G.，165
Knowledge（知识），25，365－366
knowledge acquisition（知识获取），9，305
knowledge acquisition bottleneck（知识收集瓶颈），9，305
knowledge base（知识库），31，41－43，69，80－81，262，263
knowledge base editor（知识库编辑器），32－33
knowledge discovery（知识发现），366
knowledge engineer（知识工程师），29
knowledge engineering（知识工程），10，301－302
complete system development（完全系统开发），306－307
data and knowledge acquisition（数据和知识收集），304－305
evaluation and revision（评估和修订），307
integration and maintenance（集成和维护），307
problem assessment（问题评估），303－304
prototype development（原型开发），306
knowledge representation（知识表达），26，50，103－104，131－132，133－138
KnowledgeSEEKER（知识搜索器），406，408
Kohonen，T.，13，19，205，215，439
Kohonen layer（Kohonen 层），206，337－338
Kohonen network（Kohonen 网络），206
architecture（结构），206，338
training algorithm（训练算法），209－211
Kosko，B.，20，196，214，427
Kowalski，R.，19
Koza，J.，14，20，245，255

L

law of the excluded middle（排中律），55
leaf（叶结点），401
learning（学习），165
accelerated（加速），185－188
competitive（竞争），209－212，332－335
Hebbian，200－204
supervised（有监督的），171－172，179－180
unsupervised（无监督的），200－203，209－212
learning rate（学习速率），171
adaptive（自适应的），186－188
LeCun，Y.，13
Lederberg，J.，9，19
Leonardo，参见 Leonardo expert system shell
Leonardo expert system shell（Leonardo 专家系统框架），41，69，310，313，315
Level，5 Object，141，149，152，155
lift chart（收益图），参见 gain chart
Lighthill，J.，8，19
Lighthill report（Lighthill 报告），8，19
likelihood of necessity（必要似然），67
likelihood of sufficiency（充分似然），66
linear activation function（线性激活函数），169－170
linear fit function（线性适应函数），94
linearly separable function（线性可分函数），170，173－174
Lingle，R.，345
linguistic value（语言值），94－95
linguistic variable（语言变量），94－95
LISP，6，11，14，19，30，245－246，310
atom（原子），245－246
list（列表），245－246
S-expression（S 表达式），246，253
List Processor（表处理器），参见 LISP
local optimum（局部优化），229－230
logical operation（逻辑操作），172
AND（与），172
exclusive-OR（异或），172，180－181，184－185，275－276
NOT（非），61
OR（或），172
Lowe，D.，13
Lukasiewicz，J.，88，125

M

machine learning（机器学习），165，219
Malevich，K.，140
Mamdani，E.，20，106
Mamdani fuzzy inference（Mamdani 模糊推理），106－112，113
Manhattan Project（曼哈顿项目），5
market basket analysis（购物篮分析），410
massaging data（调理数据），329－330
Mathematica（数据），253
MATLAB Fuzzy Logic Toolbox（MATLAB 模糊逻辑工具箱），20，109，121－122，283，320，352
MATLAB Neural Network Toolbox（MATLAB 神经网络工具箱），19，352
MATLAB Statistics Toolbox（MATLAB 统计工具箱），369
McCarthy，J.，5，6，19，245，440
McClelland，J.，13，19
McCulloch，W.，5，6，13，19，169，213，441
McCulloch and Pitts neuron model（McCulloch 和 Pitts 神经元模型），5，169
means-ends analysis（手段－目的分析），6－7
measure of belief（可信度），75
measure of disbelief（不可信度），75
Melin，P.，289
membership function（隶属函数），92
  trapezoid（梯形），116－117
  triangle（三角形），116－117，271
membership value（隶属值），参见 degree of membership
Mendel，G.，220
metaknowledge（元知识），49
metarule（元规则），50
method（方法），142
  WHEN CHANGED（改变时），133，142
  WHEN NEEDED（需要时），133，142，146
Mexican hat function（墨西哥帽子函数），206－207
Michalewicz，Z.，220
Michie，D.，305
Microsoft Excel（微软表格处理软件），150
Minitab（质量管理统计软件），369
Minsky，M.，5，6，13，19，131，132，134，174
momentum constant（动量常数），185－186
Monte Carlo search（蒙特·卡洛查找），14，245
multidimensional data model（多维数据模型），参见 data cube
multilayer perceptron（多层感知器），175
  architecture（结构），175－176
    hidden layer（隐藏层），175－176
    input layer（输入层），175
    output layer（输出层），175
  convergence（收敛），183
  learning（学习），179－180
  accelerated learning（加速学习），185－188
multiple antecedents（多重前项），26，77，105
multiple consequents（多重后项），27，105
multiple evidences（多重论据），62
multiple fuzzy rule tables（多重模糊规则表），293
multiple hypotheses（多重假定），62
multiple inheritance（多重继承），140－141
multi-valued logic（多值逻辑），89
mutation（突变），221，226，250－251，346
    probability（概率），226
MYCIN，10，12，15，19，40，74，76，83，84，307
MySQL，389

N

NASA（美国国家航空航天局），9，365
Naylor，C.，72
Negoita，C.，20
neo-Darwinism（新达尔文主义），220
net certainty（净确信度），76－77
neural computing（神经计算），5，6，12，167－170
neural expert system（神经专家系统），262－268
neural knowledge base（神经知识库），262－263
neural network（神经网络），12－13，166－168
  artificial（人工智能），12－13，167
    biological（生物学的），166
    competitive（竞争的），205，332－335
    evolutionary（进化），285－287
    feedback（反馈），188－189
    feedforward（前馈），175
    fully connected（全连接），176
    recurrent（循环），188－189
  self-organising（自组织），200－201，205
neuro-fuzzy system（神经－模糊系统），268－276
  architecture（结构），269－273
    defuzzification layer（逆模糊层），273
    fuzzification layer（模糊层），269
    fuzzy rule layer（模糊规则层），271－272

input layer（输入层），269
output membership layer（输出成员层），272－273
learning（学习），273－276
neuron（神经元），166－168
artificial（人工的），168
binary model（二元模型），169
biological（生物的），166
Newell，A. 6，7，19，30
NOT（非），61
NP-complete problem（NP 完全问题），8，235
NP-hard problem（NP 难问题），336
normalisation layer（归一化层），279
normalised firing strength（归一化实施强度），279
numerical object（数字对象），27

O

object（对象），27，132
numerical（数字），27
symbolic（符号），27
object-oriented programming（面向对象编程），132
odds（几率），67－68
posterior（后验），68
prior（先验），67
offspring chromosome（后代染色体），参见 child chromosome
Oja's rule，386
OLAP（联机分析处理），参见 on-line analytical processing
OLTP（联机事务处理），参见 on-line transaction processing
one-parent inheritence（单亲继承），138－140
on-line analytical processing（联机分析处理），397－401
on-line transaction processing（联机事务处理），398
operational database（操作型数据库），350
operations of fuzzy sets（模糊集操作），98－100
complement（补），98
containment（包含），98－99
intersection（交），99，109
union（并），100，107
OPS，11，30，310
optical character recognition（光学字符识别），323－324
optimisation（优化），303，344－348
OR（或），172
ordered crossover（顺序交叉），345
outlier（外层），341，372－374
output layer（输出层），175
output membership layer（输出成员层），272－273
overfitting（过拟合），325，381－382

P

Papert，S.，13，174
paradox of logic（逻辑悖论），89－90
Pythagorean School（毕达哥拉斯学派），89
Russell'sParadox（罗素悖论），89
parent chromosome（亲代染色体），222－223
parent node（父结点），401
Parker，D.，13
partially mapped crossover（部分映射交叉），345
part-whole（部分－全部），参见 a－part－of
Pascal（Pascal 语言），30，32，121，253
PCA（主成分分析），参见 principal component analysis
PCA neural network（PCA 神经网络），385－386
Pearson，K.，374
Pearson's correlation coefficient（Pearson 相关系数）383
Perceptron（感知器），6，170－172
convergence theorem（收敛理论），6
learning rule（学习规则），171－172
Phone Call Rule（电话规则），308
Pitts，W.，5，6，13，19，169，213，441
plasticity（可塑性），166
possibility theory（可能性理论），88
posterior odds（后验几率），68
posterior probability（后验概率），62
prediction（预测），303，328－332，353－357
premise（前提），参见 rule、antecedent
principal component analysis（主成分分析），374－385
principle of dichotomy（二分法原理），89
principle of topographic map formation（拓扑映射的构成原理），205
prior odds（先验几率），67
prior probability（先验概率），62
probabilistic OR（概率论 或），109，273
probability（概率），57－61
conditional（条件），59
joint（联合），59
posterior（后验），62

prior（先验），62
probability theory（概率论），57－61
procedure（程序），133
production model（产生式模型），30－31
production rule（产生式规则），26－28
programmer（程序员），29－30
PROLOG（PROLOG 语言），11，19，30，310
PROSPECTOR，10－11，12，15，19，56，65，74，82，84
prototype（原型），306
Pythagorean School（毕达哥拉斯学派），89
Pythagorean Theorem（勾股定律），247，253

Q

query tool（查询工具），388，390－391

R

reasoning（推理），31
approximate（近似），262－263
Bayesian（贝叶斯），61－65，65－72
data-driven（数据驱动），37－38，309
evidential（证据），74－80，315－317
fuzzy（模糊），104－105
goal-driven（目标驱动），38－40
symbolic（象征性的），34
Rechenberg，I.，14，20，242，255
reciprocal exchange（倒数互换），346
recurrent neural network（循环神经网络），188－189
reference super set（参考超集），参见 universe of discourse
regression（回归），370
linear（线性），370－373
robust（鲁棒），373－374
regression coefficient（回归系数），370－371
reinforcement learning（增强学习），13
relation（关系），387
reproduction（再生），221
probability（概率），225－226
root node（根结点），401
Rosenblatt，F.，6，170，171，213，444
roulette wheel selection（轮盘选择），225
Roussel，P.，19
rule（规则），26－28
antecedent（前项），26
consequent（后项），26
rule-based expert system（基于规则的专家系统），30－33，41－47，50－51，308－317
rule evaluation（规则评估），107－110
rule extraction（规则提取），263－268
rule-of-thumb（拇指规则），10，33
rule table（规则表），参见 fuzzy rule table
Rumelhart，D.，13，19
run（运行），222
run-time knowledge acquisition（运行时知识获取），33
Russell's Paradox（罗素悖论），89

S

sample hit plot（样图），338，341
saturated linear function（饱和线性函数），189
scaled membership function（缩放隶属度），110
scaling（缩放），110
scatter plot（分散图），369－370，382，385
schema（模式），232－234
defining length（定义长度），233
instance（实例），232
Schema Theorem（模式定理），14，232－234
Schwefel，H.－P.，14，20，242，255
scree plot（石子堆），383－384
seed set（种子集），414－415
selection（选择），303
self-organising map（自组织映射），13，205－206，337－343
sample hit plot（样图），338，341
U-matrix（统一距离矩阵），338，341
self-organising neural network（自组织神经网络），200－201，205
semantic network（语义网络），11
Sendai subway system（仙台地铁系统），17，20
sensitivity analysis（灵敏度分析），331－332
SEQUEL，389
set（集），89
crisp（清晰），89－91
fuzzy（模糊），89－94
S-expression（S 表达式），参见 LISP，S-expression
Shannon，C.，5，19
Shapiro，E.，142
shell（框架），参见 expert system shell
Shortliffe，E.，10，19，83
sigmoid activation function（S 形激活函数），169－170，177
sign activation function（符号激活函数），169，189

Simon, H., 6, 7, 19, 30
Simpson, R., 56
Single Proton Emission Computed Tomography (单光子发射计算机化断层扫描), 348
singleton (单态模式), 参见 fuzzy singleton
slot (槽), 131, 133
slot value (槽值), 133
  Boolean (布尔), 133
  default (默认), 133
  numeric (数值), 133
  symbolic (象征性的), 133
Smalltalk (Smalltalk 语言), 253
soft computing (软计算), 259 - 260
soma (细胞), 166
SPECT (单光子发射计算机化断层扫描), 参见 Single Proton Emission Computed Tomography
split (分裂), 402, 403 - 405
Sputnik (人造地球卫星), 8
SQL (结构化查询语言), 参见 Structured Query Language
SQL query (结构化查询语言查询), 389 - 391, 399
stacking a rule (叠加规则), 38 - 40
starnet model (星形网罗模型), 394 - 395
Statgraphics, 369
statistics (统计), 369
Statistical Analysis System (统计分析系统), 369
step activation function (阶跃激活函数), 169
Sterling, L., 142
strength of belief (可信度), 75
strength of disbelief (不可信度), 75
Structured Query Language (结构化查询语言), 388 - 391
subspace decomposition (子图分解), 379
Sugeno, M., 20, 112
Sugeno fuzzy inference (Sugeno 模糊推理), 112 - 113, 114
Sugeno fuzzy model (Sugeno 模糊模型), 112 - 114, 277
  first-order (一阶), 277
  zero-order (零阶), 114, 277
Sugeno fuzzy rule (Sugeno 模糊规则), 112, 114, 277
sum of squared errors (误差平方和), 182 - 183
summation neuron (神经元总和), 279
summary statistics (统计摘要), 369
sum-product composition (和 - 积合成), 273
supervised learning (有监督的学习), 171 - 172, 179 - 180
survival of the fittest (适者生存), 220
Sutton, R., 13
symbolic object (符号对象), 27
symbolic reasoning (符号推理), 34
synapse (突触), 166
synaptic weight (突触权重), 参见 weight

T

terminal node (终端结点), 参见 decision tree, leaf
test set (测试集), 326, 335
theory of evolution (进化论), 220
theory of natural selection (自然选择学说), 220
thinking (思考), 1
threshold (阈值), 169 - 170, 181
threshold value (阈值), 参见 threshold
time series (时间序列), 354
topographic map formation (拓扑映射的构成), 205
toy problem (玩具问题), 8
tracing facilities (跟踪工具), 33
training set (训练集), 173, 327, 329
transfer function (转换函数), 参见 activation function
transitivity (传递性), 102
travelling salesman problem (旅行商问题), 344
troubleshooting (解决纷争), 308 - 312
truth value (真实值), 107
Tryon, R., 336
TSP (旅行商问题), 参见 travelling salesman problem
tuple (元组), 387
Turing, A. (图灵), 2, 17, 19, 219, 449
Turing Imitation Game (图灵模拟游戏), 2 - 4
Turing test (图灵测试), 2 - 4

U

U-matrix (U - 矩阵), 338, 341
uncertainty (不确定性), 55 - 56
union (并), 100, 107
universal machine (通用机器), 2
universe of discourse (论域), 90 - 91
unsupervised learning (无监督学习), 200 - 203, 209 - 212
user (用户), 30
user interface (用户接口), 32, 263

V

vagueness（含糊），88
validation error（有效误差），382
validation set（有效集），381
visualisation（可视化），参见 data visualisation
von Neumann，J.，5，19

W

Waterman，D.，11，19
Watrous，R.，185
weak method（弱方法），7
weight（权重），167－168
  distance matrix（距离矩阵），参见 U－matrix
Weismann，A.，220
WHEN CHANGED method（WHEN CHANGED 方法），133，142
WHEN NEEDED method（WHEN NEEDED 方法），133，142，146
Widrow's rule（Widrow 拇指规则），329
winner-takes-all neuron（“胜者通吃”神经元），205－206

X

XOR（异），参见 logical operation，exclusive－OR

Y

Yager，R.，20

Z

Zadeh，L.，7，15，16，20，88－89，92，103，125，259，260，435
zero-order Sugeno fuzzy model（零阶 Sugeno 模糊模型），113，277