普通高等教育信息技术类系列教材

人工智能基础

主编　柳　林　戚桂美　于海清

本书免费提供

教学大纲

教学课件

科学出版社

普通高等教育信息技术类系列教材

人工智能基础

主　编　柳　林　戚桂美　于海清

副主编　张树钧　王海龙　李艳玲

科学出版社

北　京

内 容 简 介

本书以简明的方式，系统、全面地介绍人工智能基础知识及其拓展应用，目的是用通俗易懂的方法帮助学生构建完整的人工智能知识体系，为后续深入学习人工智能相关知识打下坚实的理论基础。全书共 8 章，主要内容包括人工智能的概述与流派、知识表示、搜索求解、群集智能算法、专家系统、机器学习、神经网络与深度学习、强化学习。每章附有习题，以供读者练习。

本书可作为高等院校的人工智能专业的导论课教材和非人工智能专业的选修课教材，也可作为人工智能技术爱好者学习的入门书籍。

图书在版编目(CIP)数据

人工智能基础 / 柳林，戚桂美，于海清主编. —北京：科学出版社，2022.6
（普通高等教育信息技术类系列教材）
ISBN 978-7-03-072396-3

Ⅰ. ①人… Ⅱ. ①柳… ②戚… ③于… Ⅲ. ①人工智能-高等学校-教材 Ⅳ. ①TP18

中国版本图书馆 CIP 数据核字（2022）第 090098 号

责任编辑：杨 昕 宋 丽/ 责任校对：马英菊
责任印制：吕春珉 / 封面设计：东方人华平面设计部

科学出版社出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com

北京凌奇印刷有限责任公司印刷

科学出版社发行 各地新华书店经销
*
2022 年 6 月第 一 版 开本：787×1092 1/16
2022 年 6 月第一次印刷 印张：12 1/4
字数：290 000

定价：42.00 元
（如有印装质量问题，我社负责调换）
销售部电话 010-62136230 编辑部电话 010-62135397-2032

版权所有，侵权必究

前　言

本书以教育部高等学校计算机基础课程教学指导委员会制定的《大学计算机基础课程教学基本要求》为指导，同时结合普通本科高等学校的实际教学情况进行组织和编写，参考国内外高等院校对计算机基础类课程的优秀教学思路，围绕教育信息化 2.0 时代的数字化教学特征，融入课程思政内容，突出“应用”的特点，大力推进大学计算机课程的教学改革。本书兼顾基础知识、操作技能与应用三个方面的能力培养，突出应用性、实用性、有效性原则。

人工智能是一门以计算机科学为基础，由计算机、心理学、哲学等多学科相互融合组成的交叉学科和新兴学科。它可以用于开发、模拟、延伸和扩展人的智能理论、方法和技术，通过探索人类智能的实质，生产出一种与人类思维、智能相似的智能机器。该领域的研究包括机器人、语言识别、图像识别、自然语言处理和专家系统等。展望未来，人工智能将会引起人类社会的巨大变革，甚至改变人类的生活方式。

编者依托内蒙古师范大学计算机科学技术学院系列教材计划，根据高等教育“必需、够用”原则，按照学生认知过程编排本书，使内容由浅入深、详略得当、易于理解。编者长期从事计算机专业人工智能导论课程教学工作，深刻感受到对于非计算机专业人工智能爱好者和计算机专业学生而言，学习人工智能入门较难，而导论课程又关系到学生未来学习人工智能专业知识的基础和信心。基于此，本书尽量简洁明了地介绍人工智能基础知识，向学生展示并使其了解新一代人工智能的专业知识。

本书由柳林、戚桂美、于海清担任主编，张树钧、王海龙、李艳玲担任副主编。具体编写分工如下：王海龙编写第 1 章，于海清编写第 2 章，张树钧编写第 3 章，柳林编写第 4 章和第 8 章，李艳玲编写第 5 章，戚桂美编写第 6 章和第 7 章。

本书的出版离不开团队多位老师和研究生的支持与帮助，衷心感谢内蒙古师范大学的老师们，研究生赵晔辉、张如琳参与了本书的文字校对及排版工作，在此一并表示感谢。

本书是人工智能教学的阶段性探索成果。限于编者水平，书中难免存在不足，欢迎广大读者提出宝贵意见。

编　者

目　　录

第 1 章　人工智能 …… 1

1.1　人工智能概述 …… 1

1.1.1　人工智能的定义 …… 1

1.1.2　人工智能简史 …… 2

1.1.3　研究领域 …… 6

1.2　人工智能流派 …… 15

1.2.1　符号主义 …… 15

1.2.2　连接主义 …… 15

1.2.3　行为主义 …… 16

本章小结 …… 16

本章习题 …… 16

第 2 章　知识表示 …… 17

2.1　知识表示概述 …… 17

2.2　数理逻辑表示法 …… 17

2.2.1　命题逻辑 …… 17

2.2.2　谓词逻辑 …… 18

2.2.3　谓词逻辑表示法 …… 18

2.3　语义网络表示法 …… 19

2.3.1　语义网络的概念 …… 19

2.3.2　知识的语义网络表示 …… 19

2.3.3　语义网络中常用的语义联系 …… 23

2.3.4　语义网络表示下的推理过程 …… 25

2.3.5　语义网络表示法的特点 …… 25

2.4　过程表示法 …… 26

2.4.1　过程表示法概述 …… 26

2.4.2　过程表示法求解过程 …… 26

2.4.3　过程表示法求解总结 …… 28

2.5　框架表示法 …… 28

2.5.1　框架结构及知识表示 …… 29

2.5.2 基于框架的推理……31
2.5.3 框架表示法的特点……32
2.6 产生式表示法……32
2.6.1 产生式系统的组成……33
2.6.2 产生式系统推理机的推理方式……34
2.6.3 产生式系统的优缺点……36
2.7 状态空间表示法……36
2.7.1 状态空间表示法的描述……37
2.7.2 状态空间表示法示例……38
2.8 问题归约法……39
2.8.1 问题归约描述……39
2.8.2 与图、或图、与或图表示法……39
2.9 面向对象表示法……40
2.9.1 面向对象的基本概念……40
2.9.2 面向对象的知识表示……42
本章小结……43
本章习题……43
第 3 章 搜索求解……45
3.1 搜索概述……45
3.1.1 搜索的基本问题与主要过程……45
3.1.2 搜索策略……45
3.2 状态空间的搜索策略……47
3.2.1 状态空间表示法……47
3.2.2 状态空间的图描述……47
3.3 盲目的图搜索策略……48
3.3.1 回溯策略……48
3.3.2 宽度优先搜索策略……51
3.3.3 深度优先搜索策略……52
3.4 启发式图搜索策略……54
3.4.1 启发式策略……55
3.4.2 启发信息和估价函数……55
3.4.3 A 搜索算法……56
3.4.4 A*搜索算法及其特征分析……59
3.5 综合案例……60
本章小结……63
本章习题……63

第 4 章　群集智能算法 ····· 64

4.1　群集智能概述 ····· 64

4.1.1　群集智能的基本概念 ····· 64

4.1.2　群集智能研究方法的主要优缺点 ····· 64

4.2　蚁群算法 ····· 65

4.2.1　蚁群算法的生物原型 ····· 65

4.2.2　基本蚁群算法的原理 ····· 65

4.2.3　蚁群优化算法的特点及收敛性 ····· 66

4.2.4　基本蚁群算法的数学模型 ····· 67

4.2.5　蚁群算法的参数设置 ····· 68

4.2.6　改进的蚁群算法 ····· 69

4.3　粒子群优化算法 ····· 72

4.3.1　粒子群优化算法的生物原型 ····· 72

4.3.2　标准粒子群优化算法 ····· 72

4.3.3　量子粒子群优化算法 ····· 73

4.4　遗传算法 ····· 74

4.4.1　遗传算法原型 ····· 74

4.4.2　编码 ····· 75

4.4.3　群体设定 ····· 76

4.4.4　适应度函数 ····· 77

4.4.5　选择 ····· 77

4.4.6　交叉 ····· 79

4.4.7　变异 ····· 79

4.4.8　终止条件 ····· 80

本章小结 ····· 80

本章习题 ····· 81

第 5 章　专家系统 ····· 82

5.1　专家系统概述 ····· 82

5.1.1　专家系统的特点 ····· 82

5.1.2　专家系统的类型 ····· 83

5.2　专家系统的结构与工作原理 ····· 85

5.2.1　专家系统的一般结构 ····· 85

5.2.2　专家系统的工作原理 ····· 86

5.3　知识的获取 ····· 87

5.3.1　知识获取的方式 ····· 87

5.3.2　知识获取的步骤 ····· 89

5.4 专家系统的应用 …… 90
5.5 专家系统的局限性 …… 91
本章小结 …… 92
本章习题 …… 92

第 6 章 机器学习 …… 93

6.1 机器学习概述 …… 93
6.1.1 机器学习的定义 …… 93
6.1.2 机器学习的主要策略 …… 93
6.1.3 机器学习系统的基本结构 …… 94
6.2 监督学习 …… 95
6.2.1 k 最近邻算法 …… 95
6.2.2 线性回归 …… 98
6.2.3 决策树 …… 100
6.2.4 支持向量机 …… 105
6.2.5 贝叶斯分类 …… 109
6.2.6 神经网络 …… 112
6.3 无监督学习 …… 115
6.3.1 聚类 …… 116
6.3.2 自编码器 …… 122
6.4 弱监督学习 …… 125
6.4.1 半监督学习 …… 125
6.4.2 迁移学习 …… 129
6.4.3 强化学习 …… 135
本章小结 …… 139
本章习题 …… 139

第 7 章 神经网络与深度学习 …… 141

7.1 神经网络概述 …… 141
7.2 神经元与神经网络 …… 141
7.2.1 生物神经元结构 …… 141
7.2.2 神经元数学模型 …… 142
7.2.3 神经网络的结构 …… 145
7.2.4 神经网络的学习 …… 145
7.3 BP 神经网络 …… 146
7.3.1 BP 神经网络的结构 …… 146
7.3.2 BP 学习算法 …… 147

7.3.3　BP 神经网络的应用 …… 149
7.4　卷积神经网络 …… 149
7.4.1　人脑视觉激励 …… 150
7.4.2　卷积神经网络的结构 …… 150
7.4.3　卷积神经网络的局部连接 …… 150
7.4.4　卷积神经网络的权值 …… 151
7.4.5　卷积神经网络的多卷积核 …… 151
7.4.6　卷积神经网络的池化 …… 152
7.4.7　卷积神经网络应用 …… 153
7.5　循环神经网络 …… 153
7.5.1　循环神经网络的结构 …… 154
7.5.2　循环神经网络的输出 …… 154
7.5.3　梯度消失和梯度爆炸 …… 155
7.5.4　长短时记忆网络 …… 156
7.5.5　循环神经网络的应用 …… 157
7.6　生成对抗网络 …… 158
7.6.1　生成对抗网络的基本原理 …… 158
7.6.2　生成对抗网络的结构 …… 159
7.7　深度学习的应用 …… 159
7.7.1　深度学习在博弈中的应用 …… 159
7.7.2　深度学习在医学影像识别中的应用 …… 160
7.7.3　生成对抗网络在图像处理中的应用 …… 162
7.7.4　生成对抗网络在语言处理中的应用 …… 163
本章小结 …… 165
本章习题 …… 166

第 8 章　强化学习 …… 167

8.1　任务与奖赏 …… 167
8.2　k-摇臂赌博机 …… 169
8.2.1　探索与利用 …… 169
8.2.2　ε-贪心 …… 170
8.2.3　Softmax …… 170
8.3　有模型学习 …… 171
8.3.1　策略评估 …… 172
8.3.2　策略改进 …… 173
8.3.3　策略迭代与值迭代 …… 174

8.4 免模型学习 ····· 175
8.4.1 蒙特卡罗强化学习 ····· 175
8.4.2 时序差分学习 ····· 177
8.5 值函数近似 ····· 178
8.6 模仿学习 ····· 179
8.6.1 直接模仿学习 ····· 179
8.6.2 逆强化学习 ····· 180
本章小结 ····· 181
本章习题 ····· 181
参考文献 ····· 183

第1章　人 工 智 能

1.1　人工智能概述

1.1.1　人工智能的定义

人工智能（artificial intelligence，AI）是用人工的方法在机器（计算机）上实现的智能，或者说是人们使机器具有类似于人的智能。由于人工智能是在机器上实现的，因此，又称为机器智能（machine intelligence）。

关于“人工智能”的含义，早在它被正式命名之前，已由英国数学家艾伦·麦席森·图灵（Alan Mathison Turing）提出。1950 年，他发表了题为《计算机器与智能》（*Computing Machinery and Intelligence*）的论文，文章以“机器能思维吗？”开始，论述并提出了著名的“图灵测试”，形象地指出什么是人工智能以及机器应该达到的智能标准。图灵在这篇论文中指出，不要问机器是否能思维，而是要看它能否通过如下测试：让人与机器分别在两个房间里，他们可以通话，但彼此都看不到对方，如果通过对话，作为人的一方不能分辨对方是人还是机器，那么就可以认为那台机器达到了人类智能的水平。为了进行这个测试，图灵还设计了一个有趣且智能性很强的对话内容，称为“图灵的梦想”。

现在许多人仍把图灵测试作为衡量机器智能的准则，但是也有人认为图灵测试仅仅反映了结果，没有涉及思维过程。即使机器通过了图灵测试，也不能认为机器就拥有智能。针对图灵测试，哲学家约翰·塞尔勒（John Searle）在 1980 年设计了“中文屋”思想实验以说明这一观点。在“中文屋”思想实验中，一个完全不懂中文的人处在一间密闭的屋子里，给他一本中文处理规则的书，他不必理解中文就可以使用这些规则。屋外的测试者通过门缝不断给他写一些有中文语句的纸条。他在书中查找这些中文语句的规则，根据规则将一些中文字符抄在纸条上作为对相应语句的回答，并将纸条递出房间。这样，从屋外的测试者看来，仿佛屋里的人是一个以中文为母语的人，但是屋子里的人实际上并不理解他所处理的中文，也不会在此过程中提高自己对中文的理解。用计算机模拟这个系统，可以通过图灵测试。这说明一个按照规则执行的计算机程序不能真正理解其输入、输出的意义。许多人对塞尔勒的“中文屋”思想实验进行了反驳，但是还没有人能够彻底将其驳倒。

实际上，要使机器达到人类智能的水平是非常困难的，但是，人工智能的研究正朝着这个方向发展，图灵的梦想总有一天会变成现实。特别是在专业领域，人工智能能够充分利用计算机的特点，具有显著的优越性。2014 年，图灵测试举办方——英国雷丁

大学宣称居住在美国的俄罗斯人弗拉基米尔·维西罗夫（Vladimir Veselov）创立的 AI 软件——尤金·古斯特曼（Eugene Goostman）通过了图灵测试。尤金让 33%的测试者相信它是人类。

从科学的角度看，人工智能是一门研究如何构造智能机器（智能计算机）或智能系统，使它能模拟、延伸、扩展人类智能的学科。通俗地说，人工智能就是研究如何使机器具有能听、会说、能看、会写、能思维、会学习、能适应环境变化、能解决面临的各种实际问题等功能的一门新的技术科学。

实际上，人工智能就是运用知识解决问题，造福人类，就像 100 年前的电气化一样。人类现在的绝大部分职业在不久的将来都会被智能设备所取代。

1.1.2 人工智能简史

回顾人工智能的发展历史，可归结为孕育、形成和发展三个阶段。

1．孕育阶段（1956 年之前）

自古以来，人们就一直试图用各种机器来代替人的部分脑力劳动，以提高人们征服自然的能力。其中对人工智能的产生、发展有重大影响的主要研究成果有以下几个方面。

1）早在公元前 384 年到公元前 322 年，伟大的哲学家和思想家亚里士多德就在他的名著《工具论》中提出了形式逻辑的一些主要定律。他提出的三段论至今仍是演绎推理的基本依据。

2）英国哲学家弗朗西斯·培根（Francis Bacon）曾系统地提出了归纳法，还提出了“知识就是力量”的警句。这对于研究人类的思维过程，以及自 20 世纪 70 年代人工智能转向以知识为中心的研究产生了重要影响。

3）德国数学家和哲学家戈特弗里德·威廉·莱布尼茨（Gottfried Wilhelm Leibniz）提出了万能符号和推理计算的思想，他认为可以建立一种通用的符号语言以及在此符号语言上进行推理的演算。这一思想不仅为数理逻辑的产生和发展奠定了基础，而且是现代机器思维设计思想的萌芽。

4）英国逻辑学家乔治·布尔（George Boole）致力于使“思维规律”形式化和实现机械化，并创立了布尔代数。他在《思维规律的研究》（*An Investigation of the Laws of Thought*）一书中首次用符号语言描述了思维活动的基本推理法则。

5）英国数学家图灵在 1936 年提出了一种理想计算机的数学模型，即图灵机，为后来电子数字计算机的问世奠定了理论基础。

6）美国神经学家沃伦·麦卡洛克（Warren McCulloch）与数学家沃尔特·匹茨（Walter Pitts）在 1943 年建成了第一个神经网络模型（M-P 模型），开创了微观人工智能的研究工作，为后来人工神经网络的研究奠定了基础。

7）美国爱荷华州立大学的约翰·文森特·阿塔纳索夫（John Vincent Atanasoff）和他的研究生克利福德·贝瑞（Clifford Berry）在 1937～1941 年间开发了世界上第一台电子计算机——阿塔纳索夫-贝瑞计算机（Atanasoff-Berry computer，ABC）。自此以后，电子计算机迅速发展，为人工智能的研究奠定了物质基础。

由上面的发展过程可以看出，人工智能的产生和发展绝不是偶然的，它是科学技术发展的必然产物。

2．形成阶段（1956～1969 年）

1956 年夏季，由当时达特茅斯学院（Dartmouth College）的年轻数学助教约翰·麦卡锡（John McCarthy）联合他的三位朋友——哈佛大学年轻数学家和神经学家、时任麻省理工学院教授马文·李·明斯基（Marvin Lee Minsky），IBM 公司信息研究中心负责人纳撒尼尔·罗切斯特（Nathaniel Rochester），贝尔实验室信息部数学研究员克劳德·艾尔伍德·香农（Claude Elwood Shannon）——共同发起并邀请普林斯顿大学的莫尔（Moore）、IBM 公司的塞缪尔（Samuel）、麻省理工学院的塞尔夫里奇（Selfridge）和索罗莫夫（Solomonff）以及兰德（RAND）公司和卡内基梅隆大学的艾伦·纽厄尔（Allen Newell）、赫伯特·亚历山大·西蒙（Herbert Alexander Simon）等 10 名年轻学者在美国达特茅斯学院召开了一次为时两个月的学术研讨会，讨论关于机器智能的问题。会上经麦卡锡提议正式采用了“人工智能”这一术语，麦卡锡因而被称为“人工智能之父”。这是一次具有历史意义的重要会议，它标志着人工智能作为一门新兴学科正式诞生。此后，美国形成了多个人工智能研究组织，如纽厄尔和西蒙的 Carnegie RAND 协作组、明斯基和麦卡锡的 MIT 研究组、塞缪尔的 IBM 工程研究组等。

自这次会议之后的 10 多年间，人工智能的研究在机器学习、定理证明、模式识别、问题求解、专家系统及人工智能语言等方面都取得了许多令人瞩目的成就。

1）在机器学习方面，1957 年弗兰克·罗森布拉特（Frank Rosenblatt）研制成功了感知机。这是一种将神经元用于识别的系统，它的学习功能引起了科学家们广泛的兴趣，推动了连接机制的研究，但是很快人们发现了感知机的局限性。

2）在定理证明方面，美籍华人数理逻辑学家王浩于 1958 年在 IBM-704 机器上用 3～5 min 证明了《数学原理》中有关命题演算的全部定理（220 条），并且证明了谓词演算中 150 条定理的 85%；1965 年罗宾逊（Robinson）提出了归结原理，为定理的机器证明做出了突破性的贡献。

3）在模式识别方面，1959 年塞尔夫里奇推出了一个模式识别程序；1965 年罗伯特（Roberts）编制了可分辨积木构造的程序。

4）在问题求解方面，1960 年纽厄尔等通过心理学试验总结出了人们求解问题的思维规律，编制了通用问题求解程序 GPS，可以求解 11 种不同类型的问题。

5）在专家系统方面，美国斯坦福大学的爱德华·费根鲍姆（Edward Albert Feigenbaum）领导的研究小组自 1965 年开始对专家系统 DENDRAL 进行研究，1968 年完成并投入使用。该专家系统能根据质谱仪的实验，通过分析推理决定化合物的分子结构，其分析能力已接近甚至超过有关化学专家的水平，在美、英等国得到了实际的应用。该专家系统的研制成功不仅为人们提供了一个实用的专家系统，而且对知识表示、存储、获取、推理及利用等技术是一次非常有益的探索，为以后专家系统的建造树立了榜样，对人工智能发展产生了深刻的影响，其意义远远超过了系统本身在实用上所创造的价值。

6）在人工智能语言方面，1960 年麦卡锡研制出了人工智能语言 Lisp，成为建造智

能系统的重要工具。

1969 年召开的国际人工智能联合会议（International Joint Conference on Artificial Intelligence，IJCAI）是人工智能发展史上一个重要的里程碑，它标志着人工智能这门新兴学科已经得到了世界的肯定和公认。1970 年创刊的、国际性的《人工智能》（*Artificial Intelligence*）杂志对推动人工智能的发展、促进研究者们之间的交流起到了重要的作用。

3．发展阶段（1969 年以后）

进入 20 世纪 70 年代，许多国家开展了人工智能的研究，涌现出大量的研究成果。例如，1972 年法国马赛大学的科麦瑞尔（Comerauer）提出并实现了逻辑程序设计语言 Prolog，斯坦福大学的爱德华·肖特利夫（Edward Shortliffe）等从 1972 年开始研制用于诊断和治疗感染性疾病的专家系统 MYCIN。

但是，与其他新兴学科的发展一样，人工智能的发展道路也不是平坦的。例如，机器翻译方面的研究并不如人们最初想象得那么容易。当时人们以为只要一部双向词典及一些语法知识就可以实现两种语言文字之间的互译，后来发现机器翻译远非这么简单。实际上，由机器翻译出来的文字有时会出现十分荒谬的错误。例如，把“眼不见，心不烦”的英语“Out of sight，out of mind”翻译成俄语时变成了“又瞎又疯”；把“心有余而力不足”的英语句子“The spirit is willing but the flesh is weak”翻译成俄语，再翻译回来时，竟变成了“The wine is good but the meat is spoiled”，即“酒是好的但肉变质了”；当把“光阴似箭”的英语句子“Time flies like an arrow”翻译成日语，再翻译回来时，竟变成了“苍蝇喜欢箭”。由于机器翻译出现的这些问题，1960 年美国政府顾问委员会的一份报告裁定：还不存在通用的科学文本机器翻译，也没有很近的实现前景。因此，英国、美国当时中断了对大部分机器翻译项目的资助。在其他方面，如问题求解、神经网络、机器学习等也都遇到了困难。由此人工智能的研究陷入了困境。

人工智能研究的先驱者们认真反思，总结了前一段研究的经验和教训。1977 年，费根鲍姆在第五届国际人工智能联合会议上提出了“知识工程”的概念，对以知识为基础的智能系统的研究与建造起到了重要的作用。大多数人接受了费根鲍姆关于以知识为中心开展人工智能研究的观点。从此，人工智能的研究又迎来了蓬勃发展的以知识为中心的新时期。这个时期也称为知识应用期。

在这个时期，专家系统的研究在多个领域取得了重大突破，各种不同功能、不同类型的专家系统如雨后春笋般建立起来，产生了巨大的经济效益及社会效益。例如，1976 年，斯坦福大学的杜达等研制的地矿勘探专家系统 PROSPECTOR 拥有 15 种矿藏知识，能够根据岩石标本及地质勘探数据对矿藏资源进行估计和预测，能够对矿床分布、储藏量、品位及开采价值等进行推断，制订合理的开采方案。应用该系统，人们成功地找到了价值超亿美元的钼矿。1972～1976 年，费根鲍姆等研制的专家系统 MYCIN 能识别 51 种病菌，并能够正确处理 23 种抗生素，可协助医生诊断、治疗细菌感染性血液病，为患者提供最佳处方。该系统成功处理了数百病例，并通过了严格的测试，显示出较高的医疗水平。美国 DEC 公司的专家系统 XCON 能够根据用户要求确定计算机的配置，由专家做这项工作一般需要 3 min，而该系统只需要 0.5 min，速度提高了 300 多倍。DEC

公司还建立了另外一些专家系统，由此产生的净收益每年超过 4000 万美元。信用卡认证辅助决策专家系统 American Express 能够防止不应有的损失，每年可节省约 2700 万美元。

专家系统的成功使人们越来越清楚地认识到知识是智能的基础，对人工智能的研究必须以知识为中心来进行。由于对知识的表示、利用及获取等研究取得了较大的进展，特别是对不确定性知识的表示与推理取得了突破，促进了主观贝叶斯（Bayes）理论、确定性理论、证据理论等的建立，为人工智能中的模式识别、自然语言理解等领域的发展提供了支持，解决了许多理论及技术上的问题。

1986 年之后也称为集成发展时期。计算智能（computational intelligence，CI）弥补了人工智能在数学理论和计算上的不足，更新和丰富了人工智能理论框架，使人工智能进入了一个新的发展时期。

人工智能在博弈中的成功应用受到了人们的关注。人们对博弈的研究一直抱有极大的兴趣，早在 1956 年人工智能刚刚作为一门学科问世时，塞缪尔就研制出了跳棋程序。这个程序能从棋谱中学习，也能从下棋实践中提高棋艺，它于 1959 年击败了塞缪尔本人，于 1962 年又击败了美国一个州的冠军。

1991 年 8 月在悉尼召开的第十二届国际人工智能联合会议上，IBM 公司研制的深思（Deep Thought）计算机系统与澳大利亚象棋冠军达瑞尔·约翰森（Darryl Johansen）举行了两场人机对抗赛，结果以 1：1 平局告终。

1996 年 2 月 10～17 日，为了纪念世界上第一台通用电子计算机诞生 50 周年，美国 IBM 公司出巨资邀请国际象棋棋王卡斯帕罗夫与 IBM 公司的“深蓝”计算机系统进行了 6 局人机大战。这场比赛被人们称为“人脑与电脑的世界决战”。参赛双方分别代表了人脑和电脑的世界最高水平。当时的“深蓝”是一台运算速度达每秒 1 亿次的超级计算机。第一盘，“深蓝”就给了卡斯帕罗夫一个下马威，赢了这位世界冠军，引起了世界棋坛极大的震动。但卡斯帕罗夫总结经验，稳扎稳打，在剩下的五盘中赢三盘、平两盘，最后以总比分 4：2 获胜。一年后，即 1997 年 5 月 3～11 日，“深蓝”再次挑战卡斯帕罗夫。这时，“深蓝”已经是一台拥有 32 个处理器和强大并行计算能力的 RS/6000SP/2 超级计算机，运算速度达每秒 2 亿次。计算机里存储了百余年来世界顶尖棋手的棋局。5 月 3 日棋王卡斯帕罗夫首战击败“深蓝”，5 月 4 日“深蓝”扳回一盘，之后双方平三局。双方的决胜局于 5 月 11 日拉开了帷幕，卡斯帕罗夫在这盘比赛中仅仅走了 19 步便放弃了抵抗，比赛用时只有 1 个多小时。“深蓝”最终以 3.5：2.5 的总比分赢得了这场世人瞩目的人机大战。“深蓝”的胜利表明人工智能所达到的成就。尽管它的棋路还远非真正地对人类思维方式的模拟，但它已经向世人说明，计算机能够以人类远远不能企及的速度和准确性，完成属于人类思维的大量任务。“深蓝”精湛的残局战略使观战的国际象棋专家们大为惊讶。卡斯帕罗夫也表示：“这场比赛中有许多新的发现，其中之一就是计算机有时也可以走出人性化的棋步。在一定程度上，我不能不赞扬这台机器，因为它对棋盘走势因素有着深刻的理解，我认为这是一项杰出的科学成就。”

我国自 1978 年开始把“智能模拟”作为国家科学技术发展规划的主要研究课题之一，并在 1981 年成立了中国人工智能学会（Chinese Association for Artificial Intelligence，CAAI），目前在专家系统、模式识别、机器人学及汉语的机器理解等方面都取得了很多研究成果。

1.1.3 研究领域

目前，随着人工智能研究的发展和计算机网络技术的广泛应用，人工智能技术已经应用到越来越多的领域，特别是人工智能产业链也在迅速发展。人工智能的研究是结合具体领域进行的，下面简要介绍几个主要研究领域。

（1）自动定理证明

自动定理证明是人工智能中最先进行研究并得到成功应用的一个研究领域，同时它也对人工智能的发展起到了重要的推动作用。实际上，除了数学定理证明，医疗诊断、信息检索、问题求解等许多非数学领域问题，都可以转化为定理证明问题。

定理证明的实质是证明由前提 P 得到结论 Q 的永真性。但是，直接证明 P→Q 的永真性一般来说是很困难的，通常采用的方法是反证法。在这方面，海伯伦（Herbrand）与罗宾逊先后进行了卓有成效的研究，提出了相应的理论及方法，为自动定理证明奠定了理论基础。其中，罗宾逊提出的归结原理使定理证明得以在计算机上实现，对机器推理做出了重要贡献。我国吴文俊院士提出并实现的几何定理机器证明“吴氏方法”，是机器定理证明领域的一项标志性成果。

（2）博弈

诸如下棋、打牌、战争等一类竞争性的智能活动称为博弈（game playing）。下棋是一个斗智斗策的过程，要求参赛者不仅具有超凡的记忆能力、丰富的下棋经验，而且具有很强的思维能力，能对瞬息万变的随机情况迅速做出反应，及时采取有效的措施。对于人类来说，博弈是一种智能性很强的竞争活动。

人工智能研究博弈的目的并不是为了让计算机与人进行下棋、打牌之类的游戏，而是通过对博弈的研究来检验某些人工智能技术是否能实现对人类智慧的模拟，从而促进人工智能技术的深入研究。正如俄罗斯人工智能学者亚历山大·克隆罗德所说“象棋是人工智能中的果蝇”，将象棋在人工智能研究中的作用类比于果蝇在生物遗传研究中作为实验对象所起的作用。

（3）模式识别

模式识别（pattern recognition）是一门研究对象描述和分类方法的学科。分析和识别的模式可以是信号、图像或者普通数据。

模式是对一个物体或者某些其他感兴趣的实体定量或者结构的描述，而模式类是指具有某些共同属性的模式集合。用机器进行模式识别的主要内容是研究一种自动技术，依靠这种技术，机器可以自动或者以尽可能少的人工干预的方式把模式分配到它们各自的模式类中去。

传统的模式识别方法有统计模式识别与结构模式识别等类型。近年来迅速发展的模糊数学及人工神经网络技术已经应用到模式识别中，形成了模糊模式识别、神经网络模式识别等方法，展示了巨大的发展潜力。

（4）机器视觉

机器视觉（machine vision）或者计算机视觉（computer vision）是用机器代替人眼进行测量和判断，是模式识别研究的一个重要方面。计算机视觉通常分为低层视觉与高层视觉

两类。低层视觉主要执行预处理功能，如边缘检测、移动目标检测、纹理分析、立体造型以及曲面色彩等，主要目的是让看见的对象更加突出。这时还不是理解阶段。高层视觉主要是理解对象，需要掌握与对象相关的知识。机器视觉的前沿课题包括实时图像的并行处理，实时图像的压缩、传输与复原，三维景物的建模识别，动态和时变视觉，等等。

机器视觉系统是指通过图像摄取装置将被摄取的目标转换成图像信号，传送给专用的图像处理系统。图像处理系统根据像素分布和宽度、颜色等信息，将目标转换成数字信号，并对这些信号进行各种运算，抽取目标的特征，进而根据判断的结果来控制现场的设备动作。

机器视觉的主要研究目标是使计算机具有通过二维图像认知三维环境信息的能力，能够感知与处理三维环境中物体的形状、位置、姿态、运动等几何信息。

机器视觉与模式识别存在很大程度的交叉性，两者的主要区别是机器视觉更注重三维视觉信息的处理，而模式识别仅仅关心模式的类别。此外，模式识别还包括听觉等非视觉信息。

在国外，机器视觉的应用相当普及，主要集中在半导体及电子、汽车、冶金、食品饮料、零配件装配及制造等行业。机器视觉系统在质量检测的各个方面已经得到广泛的应用。在国内，机器视觉产品刚刚起步，目前主要集中在制药、印刷包装、食品饮料等行业。随着国内制造业的快速发展，对于产品检测和质量的要求不断提高，各行各业对图像和机器视觉技术的工业自动化需求越来越大。在未来制造业中，图像和机器视觉技术将会有很大的发展空间。

（5）自然语言理解

目前人们使用计算机时，大多是用计算机的高级语言（如 C、Java 等语言）编制程序来告诉计算机“做什么”“怎么做”。这给计算机的应用带来了诸多不便，严重阻碍了计算机应用的进一步推广。如果能让计算机“听懂”或“看懂”人类自身的语言（如汉语、英语等），那么计算机将具有更广泛的用途，特别是将大幅推进机器人技术的发展。自然语言理解（natural language understanding）就是研究如何让计算机理解人类的自然语言，是人工智能中一个重要的研究领域，主要研究能够实现人与计算机之间用自然语言进行通信的理论与方法。具体地说，它要达到如下三个目标。

1）计算机能够正确理解人们用自然语言输入的信息，并能正确回答输入信息中的有关问题。

2）对输入信息，计算机能够产生相应的摘要，能够用不同词语复述输入信息的内容。

3）计算机能够把用某一种自然语言表示的信息自动翻译为用另一种自然语言表示的相同信息。

关于自然语言理解的研究可以追溯到 20 世纪 50 年代初期。当时由于通用计算机的出现，人们开始考虑用计算机把一种语言翻译成另一种语言的可能性，在此之后的 10 多年中，机器翻译一直是自然语言理解中的主要研究课题。起初，主要是进行“词对词”的翻译，当时人们认为翻译工作只要进行“查词典”及简单的“语法分析”就可以了，即对一篇需要翻译的文章，首先通过查词典找出两种语言之间的对应词，然后经过简单的语法分析调整词序就可以实现翻译。出于这一认识，人们把主要精力用在计算机内构造

不同语言对照关系的词典上。但是这种方法并未达到预期的效果，以致闹出了一些阴差阳错、颠三倒四的笑话，正像本书在前面列举的一些例子那样。

进入 20 世纪 70 年代后，一批采用语法-语义分析技术的自然语言理解系统脱颖而出，在语言分析的深度和难度方面都比早期的系统有了长足的进步。这期间，有代表性的系统主要有维诺格拉德（Winograd）于 1972 年研制的 SHRDLU、伍德（Woods）于 1972 年研制的 LUNAR、夏克（Schank）于 1973 年研制的 MARGIE 等。其中，SHRDLU 是一个在“积木世界”中进行英语对话的自然语言理解系统，该系统模拟一个能够操作桌子上摆放的一些玩具积木的机器人手臂，用户通过与计算机进行命令对话来让机器人操作积木块，如拿起、放下等；LUNAR 是一个用来协助地质学家查找、比较和评价阿波罗-11 飞船带回来的月球岩石和土壤标本化学分析数据的系统，是第一个实现了用普通英语与计算机对话的人机接口系统；MARGIE 是夏克根据概念依赖理论建成的一个心理学模型，目的是研究自然语言理解的过程。

进入 20 世纪 80 年代后，更强调知识在自然语言理解中的重要作用，1990 年 8 月在赫尔辛基召开的第十三届国际计算机语言学大会上，首次提出了处理大规模真实文本的战略目标，并组织了“大型语料库在建造自然语言系统中的作用”“词典知识的获取与表示”等专题讲座，预示着语言信息处理的一个新时期的到来。

近年来在自然语言理解的研究中，一个值得注意的事件是语料库语言学（corpus linguistics）的崛起。它认为语言学知识来自语料，人们只有从大规模语料库中获取理解语言的知识，才能真正实现对语言的理解。目前，基于语料库的自然语言理解方法还不成熟，正处于研究阶段，但是它是一个应引起重视的研究方向。

2012 年 11 月，微软公司在中国天津公开演示了全自动的同声传译系统：讲演者用英文演讲，后台的计算机传译系统自动完成语音识别、英-中机器翻译以及中文语音合成，效果非常流畅。

（6）智能信息检索

数据库系统是存储大量信息的计算机系统。随着计算机应用的发展，存储的信息量越来越庞大，研究智能信息检索系统具有重要的理论意义和实际应用价值。

智能信息检索系统应具有下述功能。

1）能够理解自然语言，允许用户使用自然语言提出检索要求和询问。

2）具有推理能力，能够根据数据库存储的事实，推理产生用户要求和询问的答案。

3）系统拥有一定的常识性知识，根据这些常识性知识和专业知识能够演绎、推理出专业知识中没有包含的答案。例如，某单位的人事档案数据库中有下列事实：“张强是采购部工作人员”“李明是采购部经理”。如果系统具有“部门经理是该部门工作人员的领导”这一常识性知识，就可以对询问“谁是张强的领导”演绎推理出答案“李明”。

（7）数据挖掘与知识发现

随着计算机网络的飞速发展，计算机处理的信息量越来越大。数据库中包含的大量信息无法得到充分的利用，造成信息浪费，甚至变成大量的数据垃圾。因此，人们开始考虑以数据库作为新的知识源。数据挖掘（data mining）和知识发现（knowledge discovery）是 20 世纪 90 年代初期崛起的一个活跃的研究领域。

知识发现系统通过各种学习方法，自动处理数据库中大量的原始数据，提炼出具有必然性且有意义的知识，从而揭示蕴含在这些数据背后的内在联系和本质规律，实现知识的自动获取。知识发现是从数据库中发现知识的全过程，而数据挖掘则是这个全过程中一个特定的、关键的步骤。

数据挖掘的目的是从数据库中找出有意义的模式。这些模式可以是一组规则、聚类、决策树、依赖网络或其他方式表示的知识。一个典型的数据挖掘过程可以分成四个阶段，即数据预处理、建模、模型评估及模型应用。数据预处理阶段主要包括数据的理解、属性选择、连续属性离散化、数据中的噪声及丢失值处理、实例选择等。建模包括学习算法的选择、算法参数的确定等。模型评估是进行模型训练和测试，对得到的模型进行评价。在得到满意的模型后，就可以运用此模型对新数据进行解释。

知识获取是人工智能的关键问题之一。因此，知识发现和数据挖掘成为当前人工智能的一个研究热点。

（8）专家系统

专家系统是目前人工智能中最活跃、最有成效的一个研究领域。自 1965 年费根鲍姆等开始研究 DENDRAL 以来，专家系统已获得了迅速的发展，广泛地应用于医疗诊断、地质勘探、石油化工、教学及军事等各个领域，产生了巨大的社会效益和经济效益。

专家系统是一个智能的计算机程序，运用知识和推理步骤来解决只有专家才能解决的困难问题。因此，可以这样定义：专家系统是一种具有特定领域内大量知识与经验的程序系统，它应用人工智能技术、模拟人类专家求解问题的思维过程来求解领域内的各种问题，其水平可以达到甚至超过人类专家的水平。

（9）自动程序设计

自动程序设计是将自然语言描述的程序自动转换成可执行程序的技术。自动程序设计与一般的编译程序不同，编译程序只能把用高级程序设计语言编写的源程序翻译成目标程序，而不能处理自然语言类的高级形式语言。

自动程序设计包括程序综合与程序正确性验证两个方面的内容。程序综合用于实现自动编程，即用户只需告诉计算机要“做什么”，无须说明“怎么做”，计算机就可以自动实现程序的设计。程序正确性验证是要研究出一套理论和方法，通过运用这套理论和方法可以证明程序的正确性。目前常用的验证方法是用一组已知其结果的数据对程序进行测试，如果程序的运行结果与已知结果一致，就认为程序是正确的。这种方法对于简单程序来说未必不可，但对于一个复杂系统来说就很难行得通。因为复杂程序存在着纵横交错的复杂关系，形成难以计数的通路，用于测试的数据即使很多，也难以保证对每一条通路都能进行测试，这就不能保证程序的正确性。程序正确性的验证至今仍是一个比较困难的课题，有待进一步开展研究。自动程序设计是人工智能与软件工程相结合的课题。

（10）机器人

机器人是指可模拟人类行为的机器。人工智能的所有技术几乎可以在它身上得到应用，因此，它可作为人工智能理论、方法、技术的实验场地。反过来，对机器人的研究又可大幅推动人工智能研究的发展。

自 20 世纪 60 年代初研制出尤尼梅特和沃莎特兰这两种机器人以来，机器人的研究已经从低级到高级经历了三代的发展历程。从程序控制机器人（第一代）发展到自适应机器人（第二代），目前正在发展智能机器人，即具有类似于人的智能的机器人。它具有感知环境的能力，配有视觉、听觉、触觉、嗅觉等感觉器官，能够从外部环境中获取有关信息；具有思维能力，能够对感知的信息进行处理，以控制自己的行为；具有作用于环境的行为能力，能够通过传动机构使自己的“手”“脚”等肢体行动起来，正确、灵活地执行思维机构下达的命令。目前研制的机器人大都只具有部分智能，真正的智能机器人还处于研究之中，但是现在已经迅速发展为新兴的高新技术产业。

目前，机器人已经活跃在各种生产线，涉及自动化、金属加工、食品和塑料等诸多行业。在亚马逊机器人物流系统中，机器人取代仓库工人，从早到晚不断地抬起 150 磅（约 68 kg）的重物，分好类，然后装上卡车。柯马（COMAU）公司开发的生产线上分布着 250 个机器人，没有一个工人。每个工位的机器人相互合作，对生产线源头进入的汽车空壳进行焊接、上底板、上螺钉等。

自动驾驶作为轮式机器人的典型应用已经走向实用化。2012 年 3 月 1 日，美国内华达州立法机关允许自动驾驶车辆上路行驶。2012 年 5 月 7 日，内华达州机动车辆管理局批准了美国首个自动驾驶车辆许可证。据专家预测，到 2026 年，无人驾驶汽车将占全美汽车总量的 10%。到 2050 年，大多数货车将实现无人驾驶。无人驾驶汽车拥有巨大潜力，可大幅增加安全性，减少温室气体排放，同时改变交通模式。

（11）组合优化问题

有许多实际问题属于组合优化问题。例如，旅行商问题、生产计划与调度、通信路由调度等都属于这一类问题。

组合优化问题一般是 NP 完全问题。NP 完全问题是指用目前知道的最好的方法求解，问题求解需要花费的时间（或称为问题求解的复杂性）将随问题规模的增大以指数关系增长。至今还不知道对 NP 完全问题是否有花费时间较少的求解方法。例如，可使求解时间随问题规模按多项式关系增长。

随着求解问题规模的增大，问题求解程序的复杂性（用于求解程序运行所需的时间和空间或求解步数）可随问题规模按线性关系、多项式关系或指数关系增长，大多数组合优化问题求解程序面临“组合爆炸”问题。因此，经典的优化方法难以求解大规模组合优化问题，需要研究人工智能求解方法。目前已经提出了许多有效的方法，特别是遗传算法、神经网络方法等。

组合优化问题的求解方法已经应用于生产计划与调度、通信路由调度、交通运输调度、列车组编、空中交通管制和军事指挥自动化等系统。

（12）人工神经网络

人工神经网络是一个用大量简单处理单元经广泛连接而组成的人工网络，用来模拟大脑神经系统的结构和功能。早在 1943 年，麦卡洛克和匹茨就提出了神经元的数学模型（M-P 模型），从此开创了神经科学理论研究的时代。20 世纪 60～70 年代，神经网络的研究也因其自身的局限性陷入低潮。特别是著名人工智能学者明斯基等在 1969 年以批评的观点编写的很有影响力的《感知器》一书，直接导致神经网络的研究进入萧条

时期。到 20 世纪 80 年代，对神经网络的研究取得了突破性进展，特别是鲁梅尔哈特（Rumelhart）和麦克莱兰（McClelland）等于 1985 年提出了多层前向神经网络的反向传播（back propagation，BP）算法，霍普菲尔德（Hopfield）提出了霍普菲尔德神经网络模型，有力地推动了神经网络的研究，由此又使人工神经网络的研究进入了一个新的发展时期，取得了许多研究成果。

现在，神经网络已经成为人工智能中一个极其重要的研究领域。对神经网络模型、算法、理论分析和硬件实现的大量研究，为神经网络走向应用提供了物质基础。神经网络已经在模式识别、图像处理、组合优化、自动控制、信息处理、机器人学习等领域获得日益广泛的应用。

（13）分布式人工智能与多智能体

分布式人工智能（distributed artificial intelligence，DAI）是分布式计算与人工智能结合的结果。分布式人工智能系统以鲁棒性作为控制系统质量的标准，并具有互操作性，即不同的异构系统在快速变化的环境中具有交换信息和协同工作的能力。

分布式人工智能的研究目标是创建一种描述自然系统和社会系统的精确概念模型。分布式人工智能中的智能并非独立存在，只能在团体协作中实现，因而其主要研究问题是各智能体（agent）之间的合作与对话，包括分布式问题求解（distributed problem solving，DPS）和多智能体系统（multi-agent system，MAS）两个领域。分布式问题求解把一个具体的求解问题划分为多个相互合作和知识共享的模块或者节点，多智能体系统研究各智能体之间行为的协调。这两个研究领域都需要研究知识、资源和控制的划分问题。分布式问题求解往往含有一个全局的概念模型、问题和成功标准，而多智能体系统则含有多个局部的概念模型、问题和成功标准。多智能体系统更能够体现人类的社会智能，具有更强的灵活性和适应性，更适合开放和动态的世界环境，成为人工智能领域的研究热点。

（14）智能控制

智能控制就是把人工智能技术引入控制领域，建立智能控制系统。自从国际知名美籍华裔科学家傅京孙在 1965 年首先提出把人工智能的启发式推理规则用于学习控制系统以来，国内外众多的研究者投身于智能控制系统研究，并取得了一些成果。经过 20 年的努力，到 20 世纪 80 年代中叶，智能控制新学科的形成条件已经逐渐成熟。1985 年 8 月，电气与电子工程师协会（Institute of Electrical and Electronics Engineers，IEEE）在美国纽约召开了新一届智能控制学术讨论会。会上集中讨论了智能控制原理和智能控制系统的结构。1987 年 1 月，在美国费城由控制系统学会和计算机学会联合召开了智能控制国际学术讨论会。会议显示出智能控制的长足进展，也说明了高新技术的发展要求重新考虑自动控制科学及其相关领域。这次会议表明，智能控制已作为一门新学科出现在国际科学舞台上。

智能控制具有两个显著的特点。

1）智能控制是同时具有知识表示的非数学广义世界模型和传统数学模型混合表示的控制过程，往往是含有复杂性、不完全性、模糊性或不确定性以及不存在已知算法的过程，并以知识进行推理，用启发来引导求解过程。

2）智能控制的核心在于高层控制，即组织级控制，其任务在于对实际环境或过程进行组织，即决策与规划，以实现广义问题求解。

智能控制系统的智能可归纳为以下几方面。

1）先验智能：有关控制对象及干扰的先验知识，可以从一开始就在控制系统的设计中考虑。

2）反应性智能：在实时监控、辨识及诊断的基础上，对系统及环境变化的正确反应能力。

3）优化智能：包括对系统性能的先验性优化及反应性优化。

4）组织与协调智能：表现为对并行耦合任务或子系统之间的有效管理与协调。

对于智能控制的开发，目前认为有以下途径。

1）基于专家系统的专家智能控制。

2）基于模糊推理和计算的模糊控制。

3）基于人工神经网络的神经网络控制。

4）综合以上三种方法的综合型智能控制。

（15）智能仿真

智能仿真就是将人工智能技术引入仿真领域，建立智能仿真系统。仿真是对动态模型的实验，即行为产生器在规定的实验条件下驱动模型，从而产生模型行为。具体地说，仿真是在三种类型知识（描述性知识、目的性知识及处理知识）的基础上产生另一种形式的知识——结论性知识。因此，可以将仿真看作一个特殊的知识变换器，从这个意义上讲，人工智能与仿真有着密切的关系。

一方面，利用人工智能技术能够对整个仿真过程（包括建模、实验运行及结果分析）进行指导，能够改善仿真模型的描述能力，在仿真模型中引进知识表示，为研究面向目标的建模语言打下基础，提高仿真工具面向用户、面向问题的能力。另一方面，仿真与人工智能相结合可使仿真更有效地用于决策，更好地用于分析、设计及评价知识库系统，从而推动人工智能技术的发展。正是基于这些方面，近年来，将人工智能特别是专家系统与仿真相结合，成为仿真领域中一个十分重要的研究方向，引起了大批仿真专家的关注。

（16）智能计算机辅助设计

智能计算机辅助设计（intelligent computer-aided design，ICAD）就是把人工智能技术引入计算机辅助设计领域，建立智能计算机辅助设计系统。事实上，人工智能几乎可以应用到计算机辅助设计技术的各个方面。从目前发展的趋势来看，具体有以下四个方面。

1）设计自动化。

2）智能交互。

3）智能图形学。

4）自动数据采集。

从具体技术来看，ICAD 技术大致可以分为如下几种方法。

1）规则生成法。

2）约束满足方法。

3）搜索法。

4）知识工程方法。

5）形象思维方法。

（17）智能计算机辅助教学

智能计算机辅助教学（computer assisted instruction，CAI）就是把人工智能技术引入计算机辅助教学领域。智能 CAI 系统的特点是能对学生因材施教地进行指导，为此，CAI 应具备下列智能特征。

1）自动生成各种问题与练习。

2）根据学生的水平和学习情况自动选择与调整教学内容与进度。

3）在理解教学内容的基础上自动解决问题生成解答。

4）具有自然语言生成和理解能力。

5）对教学内容有理解咨询能力。

6）能诊断学生错误，分析原因并采取纠正措施。

7）能评价学生的学习行为。

8）能不断地在教学中改善教学策略。

为了实现上述智能 CAI 系统，一般把整个系统分成专门知识、教导策略和学生模型三个基本模块和一个自然语言的智能接口。

总之，智能 CAI 已是人工智能的一个重要应用领域和研究方向，引起了人工智能界和教育界的极大关注和共同兴趣。特别是 20 世纪 80 年代以来，知识工程、专家系统技术的发展，使得智能 CAI 与专家系统的关系日益密切。近几届国际人工智能联合会议都把智能 CAI 的研究列入议程，甚至还召集了专门的智能教学系统会议。

（18）智能管理与智能决策

智能管理是现代管理科学技术发展的新动向。智能管理是人工智能与管理科学、系统工程、计算机技术及通信技术等多学科、多技术互相结合、互相渗透而产生的一门新技术、新学科。

智能管理就是把人工智能技术引入管理领域，建立智能管理系统，研究如何提高计算机管理系统的智能水平，以及完善智能管理系统的设计理论、方法与实现技术。

智能管理系统是在管理信息系统、办公自动化系统、决策支持系统的功能集成和技术集成的基础上，应用人工智能专家系统、知识工程、模式识别、人工神经网络等方法和技术，进行智能化、集成化、协调化，设计和实现的新一代计算机管理系统。

智能决策就是把人工智能技术引入决策过程，建立智能决策支持系统。智能决策支持系统是在 20 世纪 80 年代初提出来的。它是在决策支持系统基础上集成人工智能、专家系统形成的，其充分发挥了专家系统中知识及知识处理的特长，既可以进行定量分析，又可以进行定性分析，能有效地解决半结构化和非结构化的问题，从而扩大了决策支持系统的范围，提高了决策支持系统的能力。据专家预测，到 2026 年，首台人工智能机器将加入公司董事会，作为决策工具。人工智能可吸取过去的经验，并结合数据进行科学决策。

智能决策支持系统是在传统决策支持系统的基础上发展起来的，由传统决策支持系

统加上相应的智能部件就构成了智能决策支持系统。智能部件可以有多种模式，如专家系统模式、知识库系统模式等。专家系统模式是把专家系统作为智能部件，这是目前比较流行的一种模式。该模式适合处理以知识为主的问题，但它与决策支持系统的接口比较困难。知识库系统模式是以知识库作为智能部件。在这种情况下，决策支持系统就是由模型库、方法库、知识库、数据库组成的四库系统。这种模式接口比较容易实现，其整体性能也较好。

一般来说，智能部件中可以包含如下一些知识。

1）建立决策模型和评价模型的知识。

2）如何形成候选方案的知识。

3）建立评价标准的知识。

4）如何修正候选方案，从而得到更好候选方案的知识。

5）完善数据库，改进对它的操作及维护的知识。

（19）智能多媒体系统

多媒体技术是当前计算机热门的研究领域之一。多媒体计算机系统是指能综合处理文字、图形、图像和声音等多种媒体信息的计算机系统。智能多媒体就是将人工智能技术引入多媒体系统，使其功能和性能得到进一步发展和提高。事实上，多媒体技术与人工智能所研究的机器感知、机器理解等技术不谋而合。所以，智能多媒体实际上是人工智能与多媒体技术的有机结合。人工智能的计算机视听觉、语音识别与理解、语音对译、信息智能压缩等技术运用于多媒体系统，将会使现在的多媒体系统产生质的飞跃。目前，基于视频的动画技术、对环境感知的动画生成、虚拟中文打字机等都已成为热点研究课题。

（20）智能操作系统

智能操作系统就是将人工智能技术引入计算机的操作系统，从质上提高操作系统的性能和效率。

智能操作系统的基本模型以智能机为基础，并能支撑外层的 AI 应用程序，实现多用户的知识处理和并行推理。智能操作系统主要有三大特点：并行性、分布性和智能性。并行性是指能够支持多用户、多进程同时进行逻辑推理和知识处理。分布性是指把计算机的硬件和软件资源分散而又有联系地组织起来，能够支持局域网和远程网处理。智能性体现在三个方面：一是操作系统所处理的对象是知识对象，具有并行推理和知识操作功能，支持智能应用程序的运行；二是操作系统本身的绝大部分程序使用 AI 程序（规则和事实）编制，充分利用硬件并行推理功能；三是其系统管理应具有较高智能程序的自动管理维护功能，如故障的监控分析等，以帮助系统维护人员做出必要的决策。

（21）智能计算机系统

智能计算机系统是指人们正在研制的新一代计算机系统。这种计算机系统从基本元件到体系结构，从处理对象到编程语言，从使用方法到应用范围，同当前的冯·诺依曼型计算机相比，都有质的飞跃和提高，它将全面支持智能应用开发，且自身具有智能。

（22）智能通信

智能通信就是将人工智能技术引入通信领域，建立智能通信系统。智能通信是在通信系统的各个层次和环节上实现智能化。例如，在通信网的构建、网管与网控、转接、

信息传输与转换等环节，都可实现智能化。这样，网络可运行在最佳状态，具有自适应、自组织、自学习、自修复等功能。

（23）智能网络系统

智能网络系统就是将人工智能技术引入计算机网络系统，如在网络构建、网络管理与控制、信息检索与转换、人机接口等环节运用人工智能的技术与成果。研究表明，人工智能的专家系统、模糊技术和神经网络技术可用于网络的连接接纳控制、业务量管制、业务量预测、资源动态分配、业务流量控制、动态路由选择、动态缓冲资源调度等许多方面。

（24）人工生命

1987 年，计算机科学家克里斯·兰顿（Chiris Langton）在美国洛斯阿拉莫斯国家实验室（Los Alamos National Laboratory，LANL）召开的“生成以及模拟生命系统”国际会议上首先提出人工生命（artificial life，AL）的概念。

人工生命是以计算机为研究工具，模拟自然界的生命现象，生成表现自然生命系统行为特点的仿真系统。它主要研究天体生物学、宇宙生物学、自催化系统、分子自装配系统、分子信息处理等的生命自组织和自复制，研究多细胞发育、基因调节网络、自然和人工的形态形成理论，研究生命系统的复杂性，研究进化的模式和方式、人工仿生学、进化博弈、分子进化、免疫系统进化、学习等，研究具有自治性、智能性、反应性、预动性和社会性的智能主体的形式化模型、通信方式、协作策略，研究生物感悟的机器人、自治和自适应机器人、进化机器人、人工脑。

1.2　人工智能流派

人工智能研究中有三大流派：符号主义、连接主义和行为主义。

1.2.1　符号主义

符号主义流派认为人工智能源于数学逻辑，人类认知和思维的基本单元是符号，而认知过程就是在符号表示上的一种运算。

符号主义致力于用某种符号来描述人类的认知过程，并把这种符号输入能够处理符号的计算机中，从而模拟人类的认知过程，实现人工智能。

符号主义的发展大概经历了推理期（20 世纪 50～70 年代）和知识期（20 世纪 70 年代至今）两个阶段。在推理期，人们基于符号知识表示，通过演绎推理技术取得了很大的成就；在知识期，人们基于符号表示，通过获取和利用领域知识来建立专家系统，取得了大量的成果。

1.2.2　连接主义

连接主义通过算法模拟神经元，并把这样一个单元叫作感知机，将多个感知机组成一层网络，多层这样的网络互相连接最终得到神经网络。

这一流派认为人工智能源于仿生学，特别是人脑模型的研究，从神经生理学和认知科学的研究成果出发，把人的智能归结为人脑高层活动的结果，强调智能活动是由大量简单的单元通过复杂的相互连接后并行运行的结果。可以根据需要解决的实际问题来构建神经网络，进而用数据不断训练这一网络，调整连接权重来模拟智能。

20 世纪 60～70 年代，连接主义，尤其是对以感知机（perceptron）为代表的脑模型的研究出现热潮，但受到当时的理论模型、生物原型和技术条件的限制，脑模型研究在 20 世纪 70 年代后期至 80 年代初期落入低潮。直到霍普菲尔德在 1982 年和 1984 年发表了两篇重要论文，提出用硬件模拟神经网络以后，连接主义才又重新受到重视。1986 年，鲁梅尔哈特等提出多层网络中的 BP 算法。进入 21 世纪后，连接主义卷土重来，提出了“深度学习”的概念。

1.2.3 行为主义

行为主义是一种基于感知-行动的行为智能模拟方法。行为主义流派认为，行为是有机体用以适应环境变化的各种身体反应的组合，它的理论目标在于预见和控制行为。

行为主义是 20 世纪末以人工智能新学派的面孔出现的，引起了许多人的兴趣。这一流派的代表作首推布鲁克斯（Brooks）的六足行走机器人，它被看作新一代的“控制论动物”，是一个基于感知-动作模式的模拟昆虫行为的控制系统。

本 章 小 结

人类智能是自然界的四大奥秘之一，很难给出确切的定义，目前有思维理论、知识阈值理论、进化理论等学派。简单地说，智能是知识与智力的总和。知识是一切智能行为的基础，智力是获取知识并应用知识求解问题的能力。

人工智能是用人工的方法在机器（计算机）上实现的智能，人工智能的发展历史可归结为孕育、形成和发展三个阶段。

人工智能的应用领域非常广泛。

人工智能主要分为三大流派：符号主义、连接主义、行为主义。

本 章 习 题

1．什么是人工智能？它在发展过程中经历了哪些阶段？

2．人工智能有哪些主要的研究领域？

3．人工智能的主要流派有哪些？

第2章　知 识 表 示

2.1　知识表示概述

知识是人对自然世界、人类社会以及思维方式与运动规律的认识与掌握，是人的大脑通过思维重新组合和系统化的信息集合。在计算机信息系统中运用知识进行信息交流或智能问题求解时，都需要预先进行知识表示，进而实现知识调用，达到利用知识求解问题的目的，因而知识表示是信息处理系统中必不可少的关键环节。

知识表示是知识的符号化和形式化的过程，是用机器表示知识的可行性、有效性的一般方法，是一种数据结构与控制结构的统一体，既考虑知识的存储，又考虑知识的使用。知识表示可以看作一组描述事物的约定，以便把人类知识表示成机器能处理的数据结构。

知识表示方法有多种，主要包括数理逻辑表示法、语义网络表示法、过程表示法、框架表示法、产生式表示法、状态空间表示法、问题归约法、面向对象表示法等。本章结合简单应用分别做具体介绍和分析。

2.2　数理逻辑表示法

数理逻辑是一种基于命题与谓词分析的形式化的语言及其推理，是人工智能产生和发展的重要基础。数理逻辑表示法包含命题逻辑表示法和谓词逻辑表示法。因命题逻辑的局限性，对于命题逻辑法，本书不做讲述。

2.2.1　命题逻辑

首先来看命题逻辑的概念。

命题：对客观世界的带有真假意义的确定性的陈述句。

1）命题一定是陈述而不是疑问，也不是感叹。

2）命题有确定性，“他晚上可能会来”不是一个命题。

3）命题有真假意义。

命题可以是单一命题，如“煤是白色的”；也可以是复合命题，如“香蕉具有黄色的皮并且内部是白色的”。

在下面的例子中利用命题逻辑表示知识，规定用大写英文字母表示命题中的实体和

事实，配合逻辑符号来表示知识。

例如，有 A、B、C 三位同学参加面试，面试官对三人进行判断，得到如下几个想法：

1）三人中至少录取一人；

2）若录取 A 而不录取 B，则一定录取 C；

3）B、C 要么都录取，要么都不录取；

4）若录取 C，则一定录取 A。

令 P、Q、R 分别表示录取 A、B、C，则老师的想法可以表示如下：

1）三人中至少录取一人：// P∨Q∨R；

2）若录取 A 而不录取 B，则一定录取 C：// P∧¬Q→R；

3）B、C 要么都录取，要么都不录取：// Q ↔R；

4）若录取 C，则一定录取 A：// R→P。

命题逻辑具有局限性。命题逻辑用符号表示命题，通过逻辑符号连接命题得到复合命题，从而实现知识表示。但命题逻辑对命题内部的结构、对不同命题的共同特征缺少描述手段。例如，小张是一个父亲，小李也是一个父亲。用 P 和 Q 表示这两个命题，则整个命题表示为 P∧Q。虽然两个命题非常类似，但是在符号中无法体现。因此，在命题逻辑基础上发展出了谓词逻辑。

2.2.2 谓词逻辑

仔细观察一个命题，会发现简单的命题至少由一个主语和一个谓语组成。主语表示独立存在的某个事物、实体或者概念，谓语刻画主语的性质、状态、关系等属性。命题中，主语往往是具体的、特定的，但是谓词具有通用性，如以下几个命题。

1）小张是一个父亲；小李是一个父亲。

2）煤是黑色的；棉花是白色的。

3）香蕉的味道是甜的；苹果的味道是甜的。

将命题中的“谓词”抽象出来，作为知识的核心，而将主语作为谓词支配的部分，就可以将知识形式化。这就是谓词逻辑。

2.2.3 谓词逻辑表示法

在谓词逻辑中，命题是用形如 $P(x_1, x_2, \cdots, x_n)$ 的谓词来表述的。一个谓词可分为“谓词名”与“个体”两个部分。

以“个体”命题的主语，表示独立存在的事物或某个抽象的概念。在形式描述 $P(x_1, x_2, \cdots, x_n)$ 中，$x_1, x_2, \cdots, x_n$ 是个体，一般用小写字母表示。个体可以是个体常量、变元或函数。“谓词名”表示个体的性质、状态或个体之间的关系。P 是谓词名，一般用大写字母表示，称 P 是一个 n 元谓词。

例如，对于命题“张三是学生”，用谓词可以表示为 student（张三）。其中，student 是谓词名，“张三”是个体，student 刻画了“张三”是个学生这一特征。

在谓词中，个体可以是常量，也可以是变元，还可以是一个函数。例如，对于命题 x>10 可以表示为 more(x,10)，其中 x 是变元。又如，命题“小张的父亲是老师”，可以

表示为 teacher(father(zhang))，其中，father(zhang)是一个函数。

当谓词中的变元都用特定的个体取代时，谓词就具有一个确定的真值——真（T）或假（F）。

2.3　语义网络表示法

语义网络是奎利恩（Quillian）于 1968 年在研究人类联想记忆时提出的一种心理学模型，他认为记忆是由概念之间的联系实现的。随后，奎利恩又把它用作知识表示。1972 年，西蒙在他的自然语言理解系统中也采用了语义网络表示法。1975 年，亨德里克斯（Hendrix）又对全称量词的表示提出了语义网络分区技术。目前，语义网络已经成为人工智能领域应用较多的一种知识表示方法，尤其是在自然语言处理方面的应用。

2.3.1　语义网络的概念

语义网络是一种用实体及其语义关系来表达知识的有向图。一个语义网络主要包括事件以及事件之间的关系。从结构上来看，语义网络一般由一些最基本的语义单元组成。这些最基本的语义单元被称为语义基元，可用三元组来表示：(节点 1, 弧线, 节点 2)。“节点”代表实体，表示各种事物、概念、情况、属性、状态、事件、动作等；“弧线”代表语义关系，表示它所连接的两个实体之间的语义联系。在语义网络中，每一个节点和弧线都必须带有标识，这些标识用来说明它所代表的实体或语义。

2.3.2　知识的语义网络表示

语义网络由下列四个相关部分组成。

1）词法部分，决定表示词汇表中允许有哪些符号，它涉及各个节点和弧线。

2）结构部分，叙述符号排列的约束条件，指定各弧线连接的节点对。

3）语义部分，确定与描述相关的（联想）意义的方法，即确定有关节点的排列及其占有物和对应弧线。

4）过程部分，说明访问过程，这些过程能够用来建立和修正描述，以及回答相关问题。

一个简单的语义网络是一个三元组：(节点 1, 弧线, 节点 2)。它可以用图表示，称为一个基本网元，如图 2.1 所示。

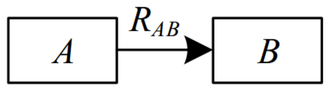


图 2.1　基本网元

图中，A、B 分别代表两个节点；R_{AB} 表示 A 与 B 之间的某种语义联系。基本网元实例如图 2.2 所示。

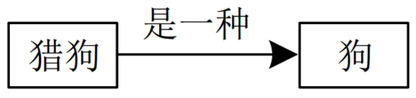


图 2.2　基本网元实例

图中，猎狗与狗之间的语义关系为“是一种”，即猎狗是狗的一种，两者之间存在类属关系。这里，弧线的方向是有意义的，需要根据事物之间的关系确定。例如，在表示类属关系时，箭头所指的节点代表上层概念，而箭尾的节点代表下层概念。

当把多个基本网元用相应语义联系关联在一起时，就可得到一个语义网络。例如，三个基本网元经合并后可得到一个语义网络，如图 2.3 和图 2.4 所示。

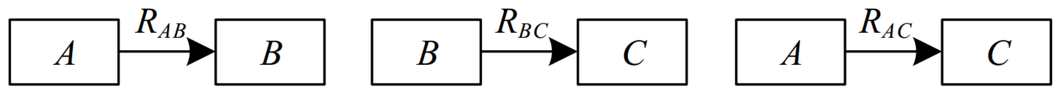


图 2.3　三个基本网元

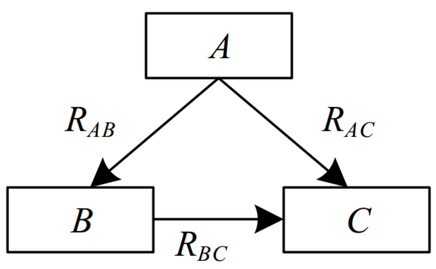


图 2.4　基本网元合并

语义网络的巴克斯-诺尔范式（Backus-Naur form，BNF）描述如下。

<语义网络>::＝<基本网元> | Merge(<基本网元>,…)

<基本网元>::＝<节点><语义联系><节点>

<节点>::＝(<属性—值对>,…)

<属性—值对>::＝(<属性名>:<属性值>)

<语义联系>::＝(<系统预定义语义联系> | <用户自定义语义联系>)

其中，Merge(…)是一个合并过程，它把括号中的所有基本网元关联在一起，即把相同的节点合并为一个，从而构成一个语义网络。

语义网络可以表示事实性的知识，也可以表示有关事实性知识之间的复杂联系。

1．用语义网络表示事实

（1）一般表示

一般表示语义网络举例如图 2.5 所示。

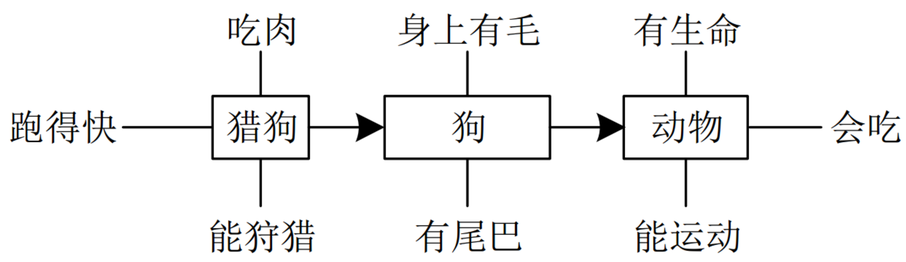


图 2.5　一般表示语义网络举例

该语义网络表示猎狗是一种狗，且进一步指出狗是一种动物，并且指出它们所具有的属性。表示这些只要在图中增加一个节点和一条弧线，并对每个节点附上相应的属性

就可以了。

语义网络的特点如下：

1）语义网络具有属性继承的特性，即下层概念可以继承上层概念的属性，这样就可以在下层概念中只列出它独有的属性。

2）另外，下层概念还可以对其上层概念的属性做进一步的细化、补充、变异，使其能够更准确地反映下层概念的特征。

（2）表示稍复杂系统

在一些稍复杂的事实性知识中，经常会使用“并且”及“或者”这样的连接词（用谓词公式表示时，可用合取符号和析取符号把它们表示出来），语义网络可以通过增设合取节点及析取节点来表示。

例如，与会者有男，有女，有年老的，有年轻的。其语义网络如图 2.6 所示（其中 *A*、*B*、*C*、*D* 分别代表四种不同情况的与会者）。

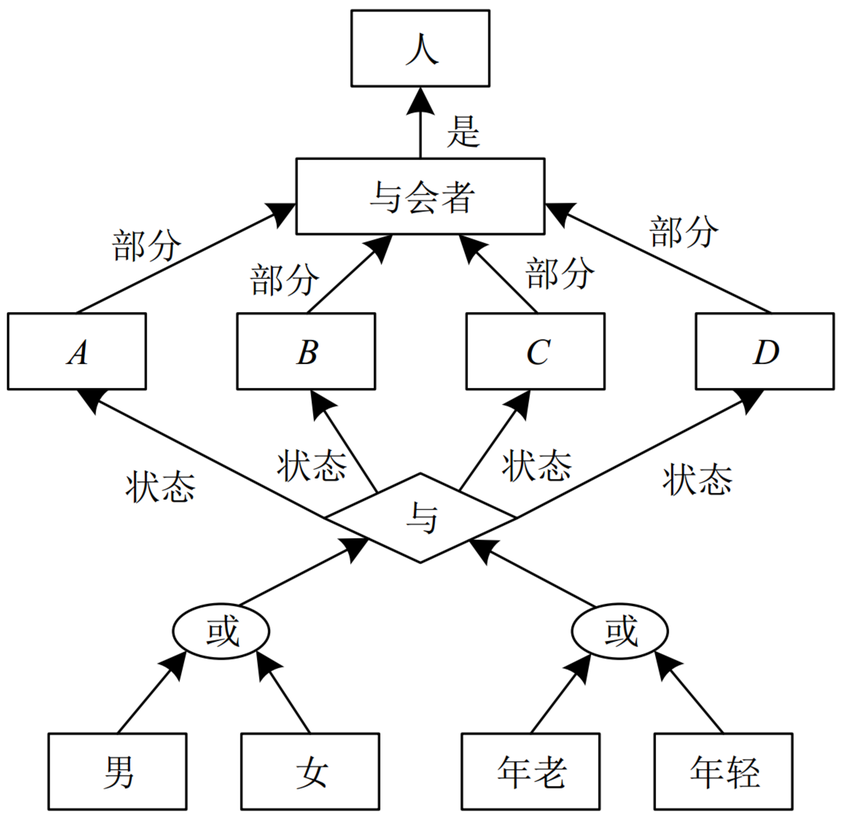


图 2.6　稍复杂系统语义网络

（3）表示某一情况、某一事件或者某个动作

上述例子中的节点都是用来表示一个事物或者一个具体概念的，其实节点还可以表示某一情况、某一事件或者某个动作。此时节点可以有一组向外的弧线，用于指出不同的情况。例如，当用节点表示某一动作时，向外的弧线可用来指出动作的主体及客体。

如图 2.7 所示，有如下事实：张三给李四一本书。可把“张三给李四一本书”作为一个事件，并在语义网络中增设一个“事件”节点。

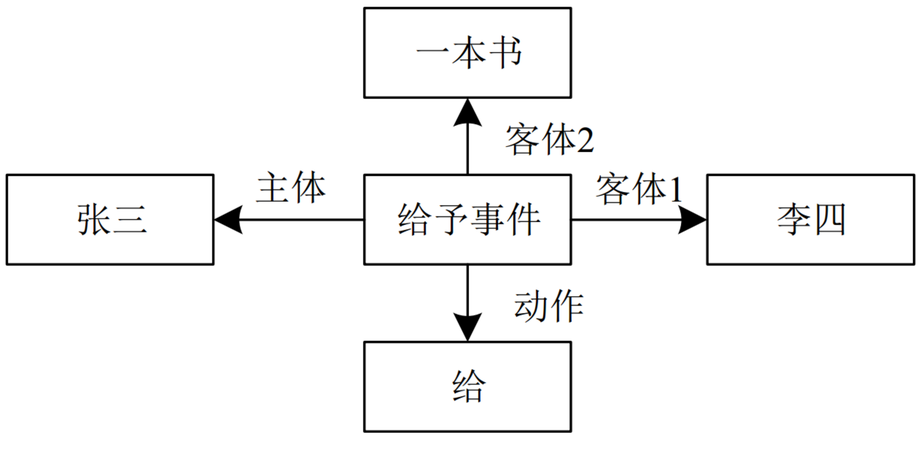


图 2.7　事件语义网络

2. 用语义网络表示有关事实之间的关系

语义网络可以描述事物之间多种复杂的语义关系，下面是常用的几种。

1）分类关系：是指事物之间的类属关系，如“是一种类型的事物”等，如图 2.8 所示。

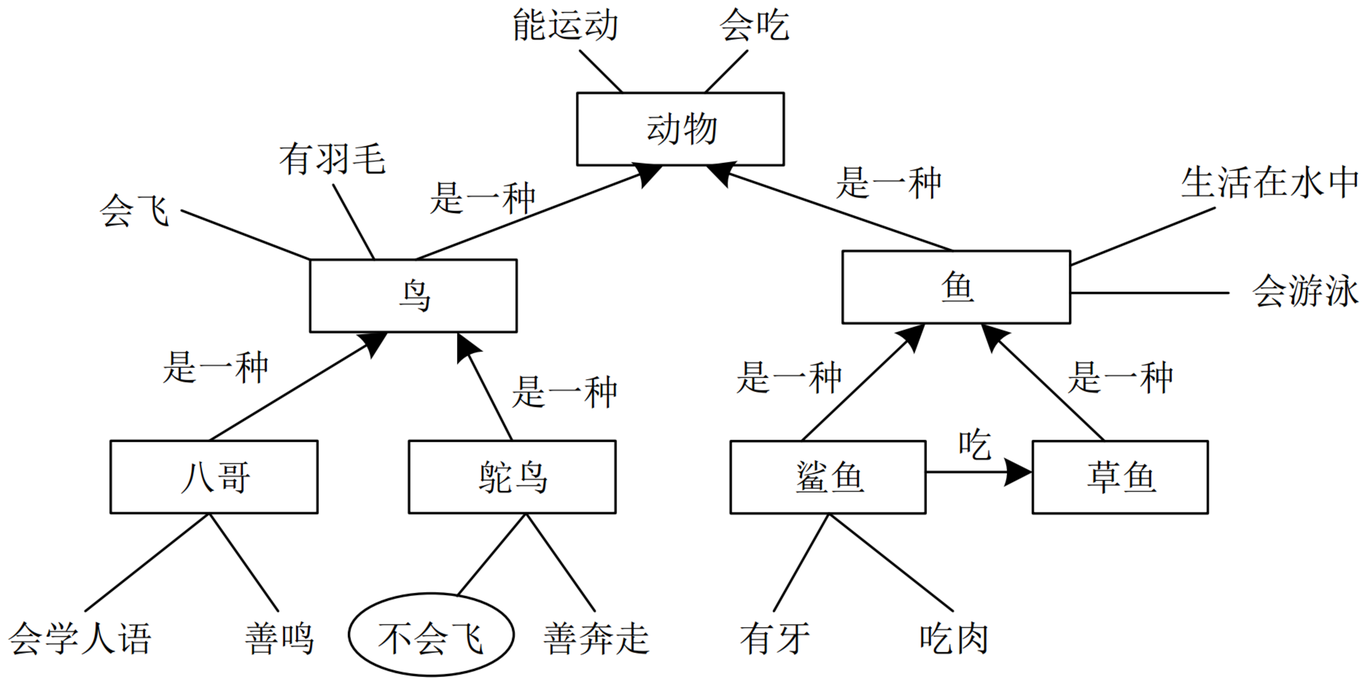


图 2.8　语义网络之分类关系

下层概念节点除可继承、细化、补充上层概念节点的属性之外，还出现了变异的情况：鸟是鸵鸟的上层概念节点，其属性是有羽毛和会飞，但是鸵鸟只是继承了有羽毛这一属性，把鸟的会飞变异为不会飞、善奔走。

2）聚集关系：如果下层概念是其上层概念的一方面或者一个部分，则称它们是聚集关系，如图 2.9 所示。

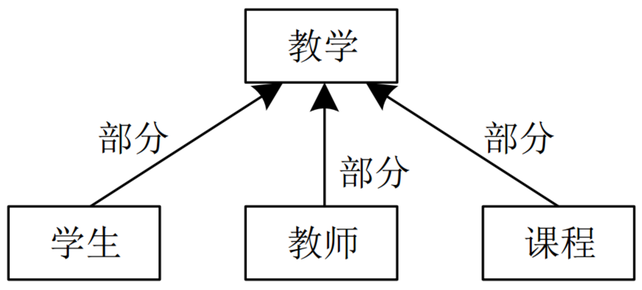


图 2.9　语义网络之聚集关系

3）推论关系：如果一个概念可由另一个概念推出，则称它们之间存在推论关系，如图 2.10 所示。

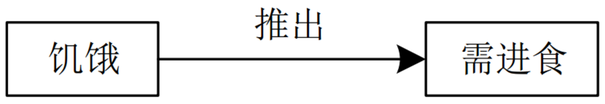


图 2.10　语义网络之推论关系

4）时间关系、位置关系：时间关系是指不同事件在其发生时间方面的先后次序关系；位置关系是指不同事物在位置方面的关系。时间关系和位置关系如图 2.11 所示。

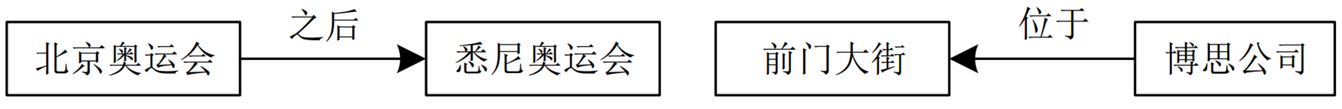


图 2.11　语义网络之时间关系（左）和位置关系（右）

5）多元关系：在语义网络中，一条弧线只能从一个节点指向另一个节点，适合表示一个二元关系。但是在许多情况下需要用一种关系把几个事物联系起来。例如，对于“郑州位于西安和北京之间”这个事实，其语义网络的多元关系如图 2.12 所示。

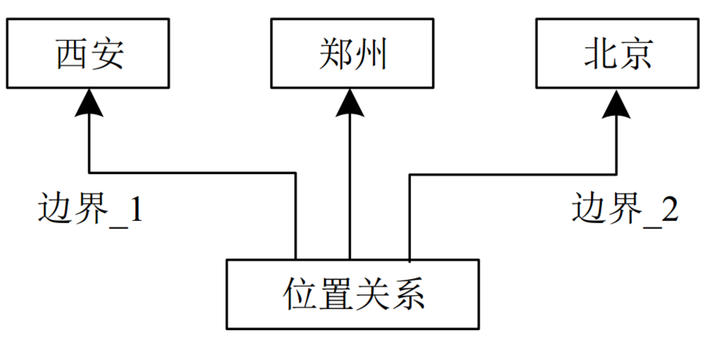


图 2.12　语义网络之多元关系

3．多元语义网络表示

设有如下两个事实，请将其用语义网络描述出来。

事实：张三的自行车是飞鸽牌、黑色、28 型；李四的自行车是金狮牌、红色、26 型。

分析：如果写成两个网络，很容易，但会给知识的利用带来不便，那么如何写成一个网络呢？分析事实发现，它们都是关于自行车的，因此可以把自行车作为一个通用概念用一个节点来表示，而把张三、李四的自行车作为他们的实例，这样就很容易用一个语义网络把它们表示出来。当要寻找有关自行车的信息时，只要找到自行车这个节点就可以了。实现效果如图 2.13 所示。

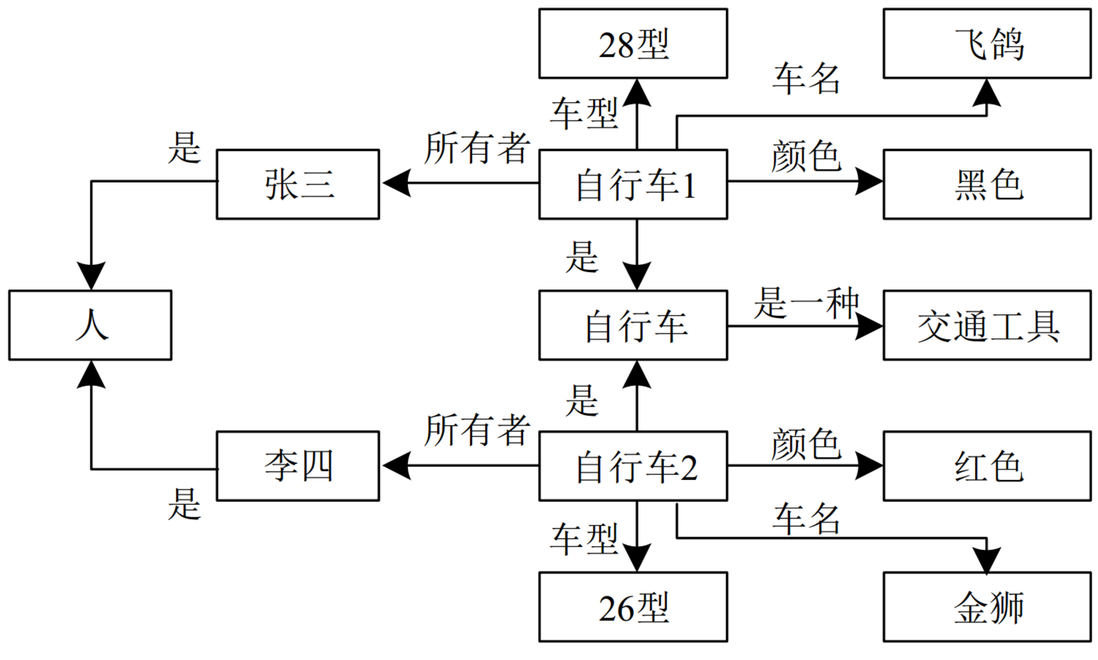


图 2.13　多元语义网络表示

2.3.3　语义网络中常用的语义联系

语义联系反映了节点之间的语义关系。下面列出一些常用的语义联系。

（1）a-member-of 联系

它表示个体与集体之间的关系，它们之间有属性继承权和属性更改权。例如，张三是工会会员，如图 2.14 所示。

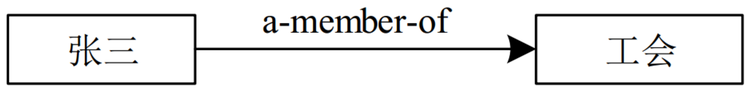


图 2.14　a-member-of 联系

（2）composed-of 联系

它表示构成联系，是一种一对多的联系，被它连接的节点不具有属性继承性。例如，整数由正整数、负整数及零组成，如图 2.15 所示。

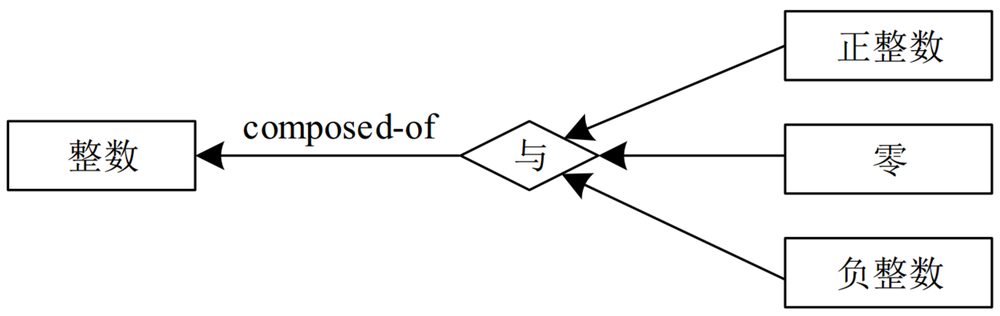


图 2.15　composed-of 联系

（3）have 联系

它表示属性或事物的占有关系，如图 2.16 所示。

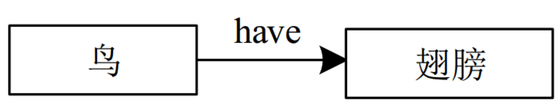


图 2.16　have 联系

（4）before、after、at 联系

它们用来表示事件之间的时间先后关系。其中，before 表示一个事件在另一个事件之前发生，after 表示一个事件在另一个事件之后发生，at 表示某一事件发生的时间，如图 2.17 所示。

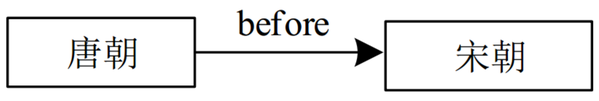


图 2.17　before 联系

（5）located-on 联系

这个联系用来表示事物之间的位置关系，如图 2.18 所示。

图 2.18　located-on 联系

（6）similar-to、near-to 联系

这些语义联系用来表示事物之间相似和接近的联系，如图 2.19 所示。

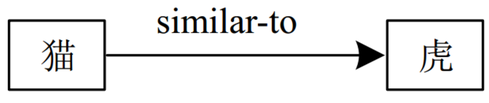


图 2.19　similar-to 联系

此外，语义网络表达知识时，有向弧 AKO（a-kind-of，分类关系）链、ISA（is-a，实例关系）链可以用来表达节点知识的可扩充性，而 AKO、ISA 也可用作语义网络。

2.3.4　语义网络表示下的推理过程

语义网络系统由两部分组成：语义网络组成的知识库；用于求解问题的解释程序——语义网络推理机。

在语义网络中，问题的求解一般是通过匹配实现的，主要过程如下。

1）根据求解问题的要求构造一个网络片段，其中有些节点或弧线的标识是空的，反映待求解的问题。

2）以此网络片段到知识库中寻找可匹配的网络，以找出所需要的信息。当然，这种匹配一般是不完全的，具有不确定性，因此需要解决不确定性匹配问题。

3）当问题的语义网络片段与知识库中的某些语义网络片段匹配时，与询问处匹配的就是问题的解。

例 2.1　设有如下事实：张三是一个学生，他在师范大学主修计算机课程，他入校的时间是 2020 年。求解问题：张三主修什么课程？

解：1）将事实用下列语义网络表示出来放在知识库中，如图 2.20 所示。

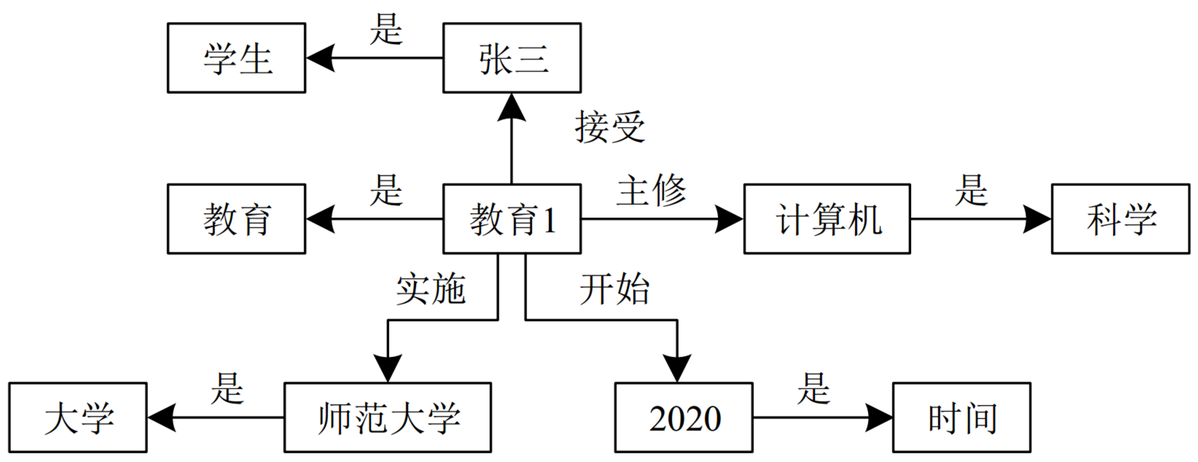


图 2.20　语义网络表示知识库

2）将待求解问题构造一个语义片段（待求解问题：张三主修什么课程？），如图 2.21 所示。

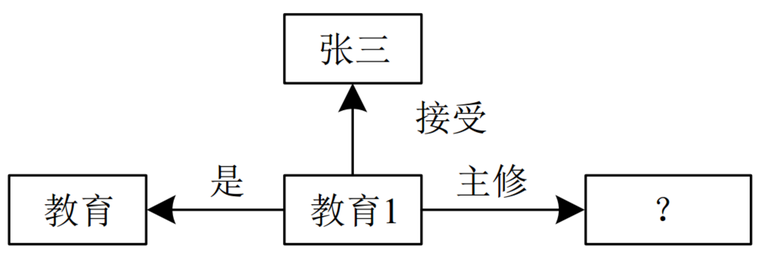


图 2.21　语义网络推理机

3）将其与知识库中的知识网络进行匹配，由主修弧线所指节点可知张三的主修课程是计算机。

4）语义网络中的推理过程主要有两种：一种是继承，另一种是匹配。因此，可以把节点分为概念节点、类节点、实例节点、值节点。

2.3.5　语义网络表示法的特点

语义网络的结构特点和推理方式等决定了其自身的特点，其表示法可以深层次地表示知识，包括实体结构、层次及实体之间的因果关系；其推理的非有规则导致无推理规

律可循；知识表达的自然性直接从语言语句强化而来。

语义网络表示法的特点总结如下。

1）结构性：语义网络把事物的属性以及事物之间的各种语义联系显式地表现出来，是一种结构化的知识表示法。

2）联想性：着重强调事物之间的语义联系，体现了人类思维的联想过程。

3）自然性：是一种直观的知识表示方法，符合人们表达事物之间关系的习惯，而且把自然语言转换成语义网络也较容易。

4）非严格性：语义网络没有公认的形式表示体系，它没有给其节点和弧线赋予确切的含义。在推理过程中有时不能区分物体的“类”和“实例”的特点。因此，通过语义网络实现的推理不能保证其推理结果的正确性。

5）处理上的复杂性：语义网络表示法的推理规则不十分明了，其表达范围受到一定限制，当语义网络中节点个数比较多时，网络结构复杂，推理就难以进行。

2.4 过程表示法

2.4.1 过程表示法概述

在人工智能的发展史中，关于知识的表示方法曾存在两种不同的观点。

一种观点认为知识主要是陈述性的，其表示方法应着重将其静态特性，即事物的属性以及事物之间的关系表示出来，称以这种观点表示知识的方法为陈述式或说明性表示方法。说明性表示方法是一种静态表示方法，其主要特征是把领域内的过程性知识与控制性知识（即问题求解策略）分离开。谓词逻辑、产生式、语义网络、框架、剧本都属于陈述性的。

另一种观点认为知识主要是过程性的，其表示方法应将知识及如何使用这些知识的控制策略均表示为求解问题的过程，称以这种观点表示知识的方法为过程性表示法或过程表示法。

过程性表示方法着重于对知识的利用，它把与问题有关的知识以及如何运用这些知识求解问题的控制策略都表述为一个或多个求解问题的过程，每一个过程编写一段程序，用于完成对一个具体时间或情况的处理。在问题求解过程中，当需要使用某个过程时就调用相应的程序并执行。

2.4.2 过程表示法求解过程

例如，八数码难题，给出一种求解该问题的过程式描述：用一个 3×3 的棋盘表示该问题的一个状态，为叙述方便，用 a～i 来标记这 9 个方格的位置，其摆有 8 个棋子，每个棋子上标有 1～8 中的某一数字，不同棋子上标的数字不相同，如图 2.22 所示。棋盘上还有一个空格，与空格相邻的棋子可以移到空格中。要求解决的问题：给出一个初始状态和一个目标状态，找出一种从初始状态转变成目标状态的移动棋子步数最少的移动步骤。

a	b	c
d	e	f
g	h	i

（a）棋盘

1	2	3
8		4
7	6	5

（b）目标状态

图 2.22　方格阵标准位置标记图

任意给定一个初始状态后，求解该问题的过程如下（图 2.23、图 2.24）。

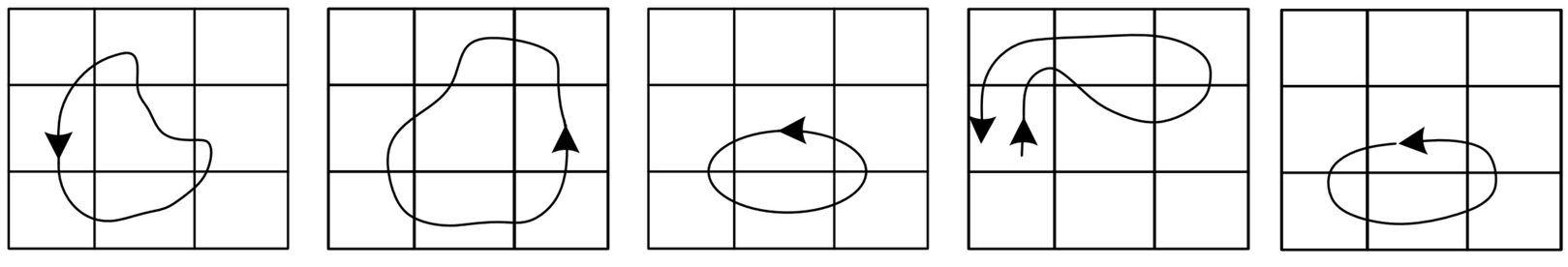


图 2.23　移动方向图

2		1
4	6	5
3	7	8

初始状态

2	1	5
4		6
3	7	8

步骤1）

1	6	5
	8	7
2	4	3

步骤2）

1	2	8
4		6
3	7	5

步骤3）

1	2	8
	3	4
7	5	6

步骤4）

1	2	3
	4	8
7	5	6

步骤5）

1	2	3
7		4
5	6	8

步骤6）

1	2	3
	5	4
6	7	8

步骤7）

1	2	3
	7	4
6	8	5

步骤8）

1	2	3
8		4
7	6	5

步骤9）

图 2.24　方位示意图

1）首先移动棋子，使棋子 1 和空格均不在位置 c 上。

2）依次移动棋子，使空格位置沿图 2.23（a）所示的箭头方向移动，直到棋子 1 位于位置 a 为止。

3）依次移动棋子，使空格位置沿图 2.23（b）所示的箭头方向移动，直到棋子 2 位于位置 b 为止。若这时刚好棋子 3 在位置 c，则转步骤 6）。

4）依次移动棋子，使空格位置沿图 2.23（c）所示的箭头方向移动，直到棋子 3 位于位置 e 为止。这时空格刚好在位置 d。

经过以上 4 步，得到的状态如图 2.23（f）所示。其中“×”表示除空格以外的任何棋子。

5）依次移动棋子，使空格位置沿图 2.23（d）所示的箭头方向移动，直到空格又回到位置 d 为止，此时状态如图 2.23（g）所示。

6）依次移动棋子，使空格位置沿图 2.23（e）所示的箭头方向移动，直到棋子 4 位于位置 f 为止。若这时刚好棋子 5 在位置 i 则转步骤 9）。

7）依次移动棋子，使空格位置沿图 2.23（h）所示的箭头方向移动，直到棋子 5 位于位置 e 为止。这时空格刚好在位置 d。

8）依次移动棋子，使空格位置沿图 2.23（i）所示的箭头方向移动，直到空格又回到位置 d 为止。

9）依次移动棋子，使空格位置沿图 2.23（j）所示的箭头方向移动，直到棋子 6 位于位置 h 为止。若这时棋子 7、棋子 8 分别在位置 g 和位置 d，则问题得解；否则，说明由所给初始状态达不到所要求的目标状态。

2.4.3 过程表示法求解总结

从以上案例可以看出，按照这样的一种过程编写的计算机程序具有非常高的求解效率。知识的修改就是对程序的修改，也是对求解过程的修改。

综上所述，过程表示法求解的优点是表示效率高，利于问题解决，控制系统容易实现。缺点是不易添加新知识，对一个过程的修改可能会波及另一个过程，给系统的维护带来不便。

2.5 框架表示法

1975 年，明斯基根据人们在理解情景、故事时提出的心理学模型，在论文《表征知识的框架》（*A Framework for Representing Knowledge*）中提出了框架理论，尽管只是思想方法而非具体实现，但还是引起了学术界的重视。

框架理论的出发点产生于人们对现实世界各种事物的认识都是以一种类似于框架的结构存储在记忆中的。当遇到新事物时，就从记忆中找出一个合适的框架，并根据新的情况对其加以修改、补充，从而形成对这个新事物的认识，称为实例框架。

框架是知识的基本单位，一组相关框架连接起来便可形成一个框架系统。

框架是由若干节点和关系（统称为“槽”）构成的网络，是语义网络一般化的形式，与后者没有本质的差别。它是表示某一类情景的结构化的一种数据结构，框架的最顶层是固定的一类事物，基于概念的抽象程度表现出自上而下的分层结构。框架由框架名和一些槽组成，每个槽可以拥有若干个侧面，每个侧面又可以拥有若干个值。

2.5.1　框架结构及知识表示

（1）框架基本结构

框架的基本结构如下。

<框架名>

<槽 1> <侧面 11><值 111>…

　　<侧面 12><值 121>…

<槽 2> <侧面 21><值 211>…

　　⋮

<槽 *n*> <侧面 *n*1><值 *n*11>…

　　<侧面 *nm*><值 *nm*1>…

约束：

约束条件 1

约束条件 2

　　⋮

约束条件 *n*

（2）基于框架的基本结构

基于框架的基本结构的实例如下。

框架名：<计算机主机>

主机品牌：联想

生产厂商：北京联想公司

CPU：品牌　Intel

型号：奔腾/933

主板：品牌　QDI

型号：ATXVA5

内存：品牌　现代

型号：SDRAM

容量：512 MB

硬盘：品牌　Seagate

型号：ST320423A

容量：40 GB

（3）框架系统组成

框架系统：多个相互联系的框架组织起来形成框架系统。

基本结构：横向或纵向联系。

横向联系：一个框架的槽值或侧面值是另外一个框架的名字，即对另一个框架的调用。

纵向联系：具有继承关系的框架之间构成纵向联系。

（4）框架表示知识的步骤

1）分析要表示的知识的对象属性，合理设置槽。

2）使用常用的槽名，描述上下层框架之间的联系，如 ISA 槽、AKO 槽、Instance 槽、Part-of 槽。

3）对各层对象的槽和侧面进行合理安排，避免信息描述的重复。

（5）具有继承关系的框架系统实例

框架名 1：<学生>

姓名：单位(姓,名)

性别：范围(男,女)

默认：男

年龄：单位(岁)

住址：单位(楼号,房间号)

电话：单位((区号),手机号)

入学时间：单位(年,月)

学制：单位(年)

框架名 2：<硕士研究生>

继承：<学生>

学籍：<学籍>

研究方向：单位(专业方向名称)

导师姓名：单位(姓,名)

参加课题：范围(国家级,省部级,其他级别)

默认：国家级

学位论文：单位(论文题目)

默认：题目未定

框架名 3：<计算机硕士研究生>

继承：<硕士研究生>

专业：范围(计算机应用技术,计算机软件及理论）

默认：计算机应用技术

使用计算机：单位(计算机型号)

（6）通过框架解决具体问题

例 2.2　2022 年 2 月 4 日，第 24 届冬季奥林匹克运动会在中国的北京和张家口开幕。本届冬季奥林匹克运动会共有 91 个国家和地区参加。本届冬季奥林匹克运动会的主题口号是“一起向未来”。

解：1）确定属性。冬季奥林匹克运动会开幕的时间、地点、参与国家和地区数量、主题口号、届数。

2）冬季奥林匹克运动会的框架如下。

框架名：<冬季奥林匹克运动会>

地点：中国的北京和张家口
时间：2022 年 2 月 4 日
参与国家和地区数量：91
主题口号：一起向未来
届数：24

2.5.2　基于框架的推理

基于框架的推理是通过继承和匹配来实现的。

（1）继承

通过继承关系求得新事物的其他属性。

（2）匹配

1）将待求解问题表示成框架，求解答案的槽值或侧面值为空。

2）在框架系统中寻找与求解问题框架匹配的框架。

3）通过匹配框架的槽值或侧面值推出问题答案。

例 2.3　从前文所建立的学生的框架系统中找出一个满足条件的硕士研究生。条件：女性，年龄 25 岁以下，专业为计算机应用技术，研究方向为人工智能。

解：将上述条件用框架表示出来，可得到如下问题框架：

框架名：<计算机系硕士研究生—*x*>
姓名：
年龄：25
性别：女
专业：计算机应用技术
研究方向：人工智能

已知框架系统中的一个实例框架：

框架名：<计算机系硕士研究生—1>
继承：<硕士研究生>
姓名：柳青
性别：女
年龄：23
研究方向：人工智能

用此问题框架与知识库中的框架匹配，显然“计算机系硕士研究生—1”框架可以匹配。因为“年龄”槽、“性别”槽、“研究方向”槽都符合要求，虽然没有给出“专业”的槽值，但是由继承性可知它取默认值“计算机应用技术”，也完全符合初始问题框架“计算机研究生—*x*”的要求，所以要找的学生有可能是柳青。

具体实现时框架中的继承关系如图 2.25 所示。

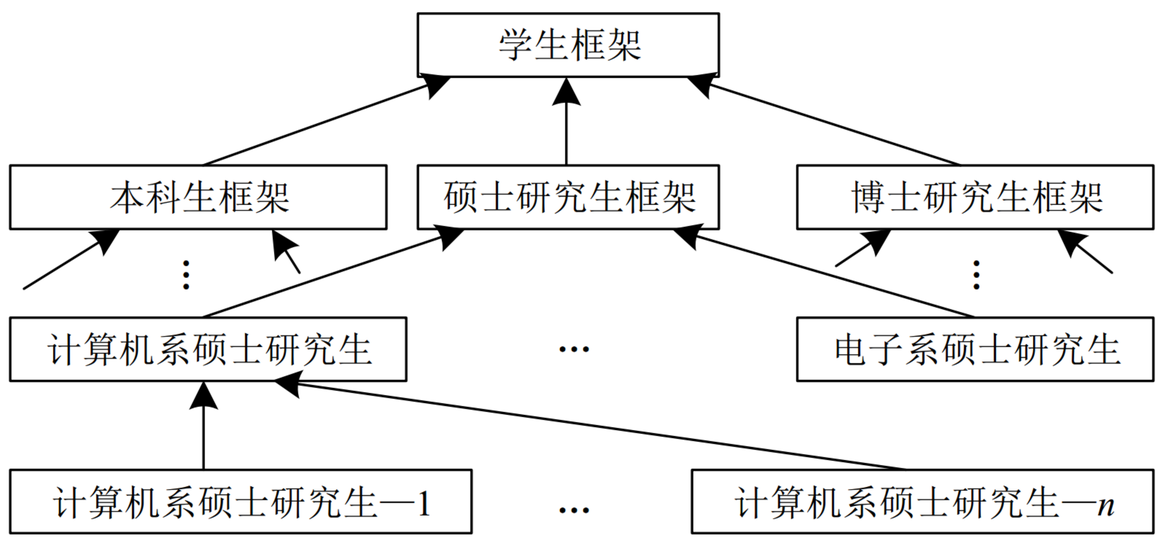


图 2.25　框架继承关系图

2.5.3　框架表示法的特点

综上所述，框架表示法有如下特点。

（1）结构性

框架表示法突出的特点是它善于表达结构性的知识，能够把知识的内容结构关系及知识之间的联系表示出来，因此，它是一种经组织起来的结构化的知识表示方法。这一特点是产生式表示所不具备的，产生式系统中的知识单位是产生式规则，这种知识单位由于太小而难以处理复杂问题，也不能把知识之间的结构关系显式地表示出来。框架表示法的知识单位是框架，而框架是由槽组成的，槽又可分为若干侧面，这样就可以把知识的内部结构显式地表示出来。

（2）继承性

框架表示法通过将槽值设置成另一个框架的名字来实现框架之间的联系，建立起表示复杂知识的框架网络。在框架网络中，下层框架可以继承上层框架的槽值，也可以进行补充和修改，这样不仅减少了知识的冗余，而且较好地保证了知识的一致性。

（3）自然性

框架表示法体现了人们在观察事物时的思维活动。当遇到新事物时，通过从记忆中调用类似事物的框架，并将其中某些细节进行修改、补充，就形成了对新事物的认识，这与人们的认识活动是一致的。

2.6　产生式表示法

产生式系统（production system）由波斯特（Post）于 1943 年提出的产生式规则（production rule）而得名。人们用这种规则对符号进行置换运算。1965 年，纽厄尔和西蒙利用这个原理建立了一个人类的认知模型。1968 年，斯坦福大学利用产生式系统结构设计出第一个专家系统 DENDRAL。

产生式系统用来描述若干个不同的以一个基本概念为基础的系统。这个基本概念就是产生式规则或产生式条件和操作对的概念。在产生式系统中，论域的知识分为两部分：用事实表示静态知识，如事物、事件和它们之间的关系；用产生式规则表示推理过程和

行为。由于这类系统的知识库主要用于存储规则，因此把这类系统称为基于规则的系统（rule-based system）。

2.6.1　产生式系统的组成

产生式系统是指认知心理学程序表征系统的一种。它是为解决某一问题或完成某一作业而按照一定层次联结组成的认知规则系统，由总数据库、产生式规则库和控制策略三部分组成，各部分之间的关系如图 2.26 所示。每一条产生式规则由条件（即当前的状态或情境）和行动两部分组成，其基本规则是“若满足条件 *X*，则实施行动 *Y*”，即当一个产生式中的条件得到满足，便执行该产生式规定的某个行动。

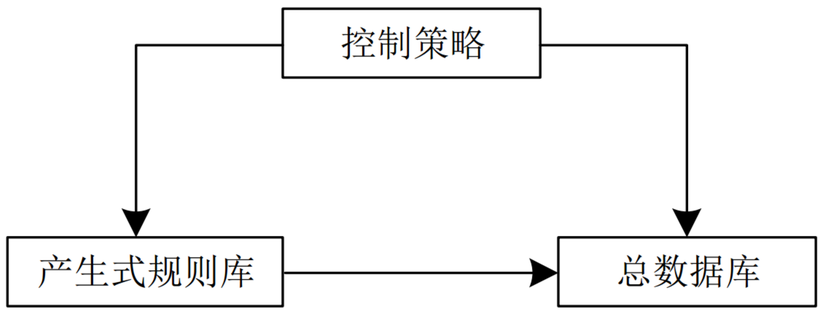


图 2.26　产生式系统的主要组成

（1）总数据库

总数据库又称综合数据库、上下文、黑板等，用于存放求解过程中各种当前信息的数据结构，如问题的初始状态、事实或证据、中间推理结论和最后结果等。其中的数据是产生式规则的处理对象。数据库中的数据根据应用问题的不同，可以是常量、变量、谓词、表结构、图像等。

例如，关于动物世界的产生式系统有如下数据库：

…

（Mammal Dog）

（Eat Dog Meat）

…

从另一个角度看，数据库可视为推理过程中间结果的存储池。随着中间结果的不断加入，数据库描述的问题状态逐步转变为目标状态。

（2）产生式规则库

产生式规则库是某领域知识用规则形式表示的集合，其中包含将问题从初始状态转换到目标状态的所有变换规则。当产生式规则中某条规则的前提与总数据库中的事实相匹配时，该规则库就被激活，并把其结论作为新的事实存入总数据库。

规则的一般形式如下：

条件→行为

或

前提→结论

用一般计算机程序语言表示如下：

```
If…then…
```

其中，左侧确定了该规则可应用的先决条件，右侧描述应用这条规则所采取的行动或得出的结论。当确定规则的前提或条件时，通常采用匹配的方法，即查看全局数据库中是否存在规则的前提或条件所指的情况，若存在则匹配成功，否则认为失败。

（3）控制策略

控制策略或称控制系统，是产生式系统的推理机。它是规则的解释程序，规定了如何选择一条可应用的规则对全局数据库进行操作，即决定了问题求解过程或推理路线。通常情况下，控制策略负责产生式规则前提或条件与全局数据库中数据的匹配，按照一定的策略从匹配成功的规则中选出一条加以执行，并在合适的时候结束产生式系统的运行。其基本的控制流程为匹配、冲突解决和操作。

1）匹配：也称识别。在这一步，把当前数据库与规则的执行条件部分相匹配。如果两者完全匹配，则把这条规则称为触发规则。当按照规则的操作部分执行时，称这条规则为启用规则。在一个循环的匹配阶段，若有多于一条的规则激活，就称引起了一个冲突。

2）冲突解决：决定首先使用哪一条规则。冲突解决的策略分为如下三类。

First：选用首条激活的规则加以执行。

Best：选用已激活规则中最好的加以执行。

All：执行所有激活的规则。

3）操作：就是执行规则的操作部分。经过操作后，当前数据库将被修改，然后其他的规则有可能被使用。

2.6.2 产生式系统推理机的推理方式

产生式系统推理机的推理方式有正向推理、逆向推理和双向推理三种。

（1）正向推理

正向推理是从已知事实出发，通过规则库得出结论。正向推理也称数据驱动方式或自底向上的方式，其推理过程如下。

1）用规则集中规则的前件与数据库中的事实进行匹配，得到匹配的规则集合。

2）从匹配的规则集合中选择一条规则作为使用规则。

3）执行使用规则，将该使用规则后件的执行结果送入数据库。

重复这个过程直至达到目标。

具体地说，如果数据库中含有 A，而规则库中有规则 $A \rightarrow B$，那么这条规则便是匹配规则，进而将后件 B 送入数据库。这样可不断扩大数据库，直至数据库中包含目标，便成功结束。如果有多条匹配规则，则从中选择一条作为使用规则，不同的选择方式直接影响求解效率。其推理过程如图 2.27 所示。

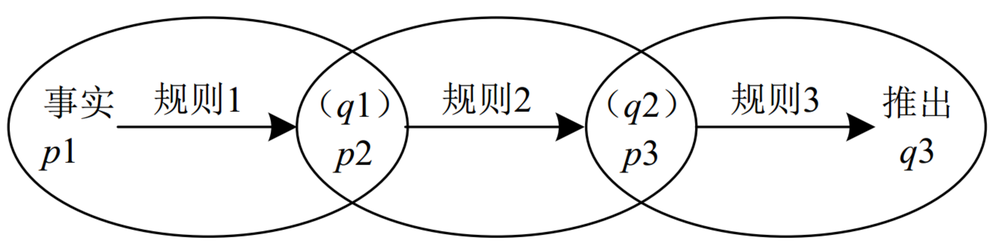


图 2.27　正向推理过程

例如，在动物识别系统 IDENTIFIER 中，包含如下几个规则：

规则 I2　如果　该动物能产乳，
　　那么　它是哺乳动物。

规则 I8　如果　该动物是哺乳动物，它能反刍，
　　那么　它是有蹄动物而且是偶蹄动物。

规则 I9　如果　该动物是有蹄动物，
　　它有长颈，有长腿，
　　它的颜色是黄褐色，它有深色斑点，
　　那么　它是长颈鹿。

这样的例子就是用了正向推理。

（2）逆向推理

逆向推理是从目标出发，逆向使用规则，找到已知事实。逆向推理也称目标驱动方式或自顶向下的方式，其推理过程如下。

1）用规则集中规则的后件与假设的目标事实进行匹配，得到匹配的规则集合。

2）从匹配的规则集合中选择一条规则作为使用规则。

3）将使用规则的前件作为新的假设子目标。

重复这个过程，直至各个子目标均为已知事实后成功结束。

如果目标明确，那么使用逆向推理效率较高。其推理过程如图 2.28 所示。

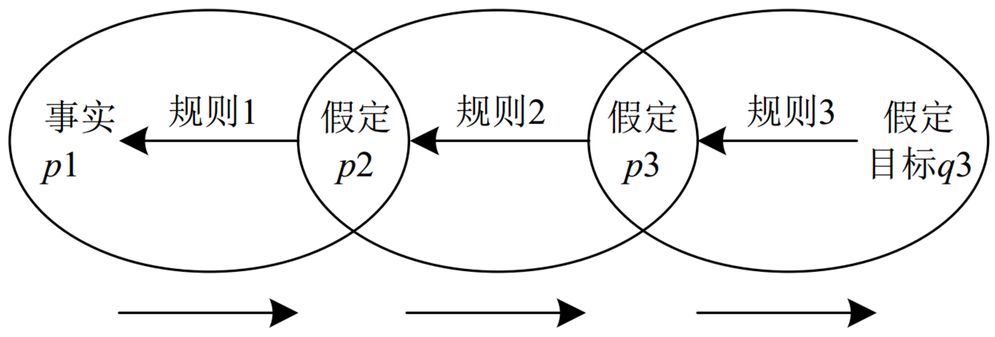


图 2.28　逆向推理过程

从上面的推理过程可以看出，做逆向推理时可以假设一个结论，然后利用规则推导支持假设的事实。

例如，在动物识别系统中，为了识别一个动物，可以进行以下逆向推理。

1）假设这个动物是长颈鹿，为了检验这个假设，根据规则 I9，要求这个动物是长颈、长腿且是有蹄动物。

2）假设全局数据库中已有该动物是长腿、长颈等事实，还要验证“该动物是有蹄动物”。因此，规则 I8 要求该动物是反刍动物且是哺乳动物。

3）验证“该动物是哺乳动物”，根据规则 I2，要求该动物是产乳动物。现在已经知道该动物是产乳动物和反刍动物，即各子目标都是已知事实，所以逆向推理成功，即“该动物是长颈鹿”假设成立。

正向与逆向推理策略的比较如表 2.1 所示。

表 2.1　正向与逆向推理的比较

指标	正向推理	逆向推理
驱动方式	数据驱动	目标驱动
推理方式	从一组数据出发向前推导结论	从可能的解答出发，向后推理验证解答
启动方式	从一个事件启动	由询问关于目标状态的一个问题启动
透明程度	不能解释其推理过程	可解释其推理过程
推理方向	自底向上推理	自顶向下推理
典型系统	CLIPS、OPS	PROLOG

（3）双向推理

双向推理又称混合推理，既自顶向下，又自底向上，从两个方面做推理，直至某个中间界面上两个方向的结果相符后成功结束。不难想象，这种双向推理较正向推理或逆向推理所形成的推理网络更小，因而推理效率更高。

例如，在动物识别系统中，已知某动物具有长腿、长颈、反刍、产乳的特征。为了识别一个动物，可以进行以下双向推理。

1）首先假设这个动物是长颈鹿，为了检验这个假设，根据规则 I9，要求这个动物是长颈、长腿且是有蹄动物。这是由逆向推理得到的中间结论。

2）根据该动物产乳，由规则 I2 可知该动物是哺乳动物；再加上该动物反刍，由规则 I8 可知该动物是有蹄动物而且是偶蹄动物。这是由正向推理得到的中间结论。

3）由 1）和 2）得到的中间结论——“有蹄动物”重合，而且 1）中的另两个中间结论“长颈、长腿”是已知事实，所以假设“这个动物是长颈鹿”是正确的。

2.6.3　产生式系统的优缺点

由于产生式系统具有模拟人类求解问题的思维方式、系统模拟性强、易于修改扩充等优点，因此得到了广泛的应用。可以说，目前很多专家系统采用产生式系统的结构方式来建立。但是，产生式系统也是有缺点的，具体如下。

1）冲突解决的策略选择问题。

2）求解效率低。因为反复执行“匹配→冲突消除→执行”过程，如果规则库较大，则易引起“组合爆炸”，效率较低。

3）不能表示结构性知识。产生式表示的知识有一定的格式，且规则之间不能直接调用，因此具有结构关系或层次关系的知识不易用它表示出来。

2.7　状态空间表示法

状态空间法（state-space method）是一种基于解答空间的问题表示和求解方法，它是以状态和操作符为基础的。在利用状态空间图表示时，从某个初始状态开始，每次加一个操作符，递增地建立起操作符的试验序列，直到达到目标状态为止。由于状态空间法需要扩展过多的节点，容易出现“组合爆炸”，因而只适用于表示比较简单的问题。

状态空间法是现代控制理论中建立在状态变量描述基础上的对控制系统进行分析

和综合的方法。状态变量是能完全描述系统运动的一组变量。如果系统的外输入为已知，那么根据这组变量的现时值就能完全确定系统在未来各时刻的运动状态。通过状态变量描述能建立系统内部状态变量与外部输入变量和输出变量之间的关系。反映状态变量与输入变量之间因果关系的数学描述称为状态方程，而输出变量与状态变量和输入变量之间的变换关系则由量测方程来描述。状态与状态变量描述的概念早就存在于经典动力学和其他一些领域，但是将它系统地应用于控制系统的研究，则是从 1960 年卡尔曼（Kalman）发表的题为《控制系统的一般理论》的论文开始的。状态空间法的引入促成了现代控制理论的建立。

2.7.1　状态空间表示法的描述

状态空间表示法是人工智能中基本的形式化方法，也是讨论问题求解技术的基础。

状态空间表示法是以“状态空间”的形式对问题进行表示。

状态空间表示法的定义如下。

（1）状态

状态是描述问题求解过程中不同时刻状况的数据结构。一般用一组变量的有序集合表示：

$$\boldsymbol{Q}=(q_0,q_1,\cdots,q_n) \tag{2.1}$$

式中，每个元素 $q_i(i=0,1,2,\cdots,n)$为集合的分量，称为状态变量。当给每一个分量以确定的值时，就得到了一个具体的状态。

（2）算符

引起状态中某些分量发生变化，从而使问题由一个状态变为另一个状态的操作称为算符。算符可分为走步、过程、规则、数学算子、运算符号或逻辑符号等。

例如，在产生式系统中，每一条产生式规则就是一个算符；在下棋程序中，一个算符就是一个走步。

（3）状态空间

由表示一个问题的全部状态及一切可用算符构成的集合称为该问题的状态空间。它一般由三部分构成。

1）由问题的所有可能初始状态构成的集合 S。

2）算符集合 F。

3）目标状态集合 G。

用一个三元组表示如下：(S,F,G)。

状态空间的图示形式称为状态空间图。其中，节点表示状态，有向边（弧线）表示算符。

（4）问题的解

从问题的初始状态集 S 出发，经过一系列的算符运算，到达目标状态。由初始状态到目标状态所用算符的序列就构成了问题的一个解。问题的求解过程是一个不断把算符作用于状态的过程。首先，将适用的算符作用于初始状态，以产生新的状态；其次，把一些适用的算符作用于新的状态；这样继续下去，直到产生的状态为目标状态为止。由此即可得到问题的一个解，这个解是从初始状态到目标状态所用算符构成的序列。这时，

可能有多个算符序列都可使问题从初始状态变到目标状态，这就得到了多个解。其中有的使用算符较少，有的使用算符较多，把使用算符最少的解称为最优解。这里只是根据解中使用算符的个数来评价解的优劣，评价解的优劣主要是看使用算符时所付出的代价，只有总代价最小的解才是最优解。

对于任何一个状态，可使用的算符可能不止一个，这样由一个状态所生成的后继状态就可能有多个。当对这些后继状态使用算符生成更进一步的状态时，首先应对哪一个状态进行操作呢？这属于搜索策略的问题，不同的搜索策略其操作的顺序是不同的。

2.7.2 状态空间表示法示例

例 2.4 二阶汉诺塔问题。设有 3 根柱子，它们的编号分别是 1 号、2 号、3 号。在初始情况下，1 号柱子上穿有 A、B 两个圆盘，A 比 B 小，A 位于 B 的上面。要求把这两个圆盘全部移到 3 号柱子上，而且规定每次只能移动一个圆盘，任何时刻都不能使大圆盘位于小圆盘的上面，如图 2.29 所示。

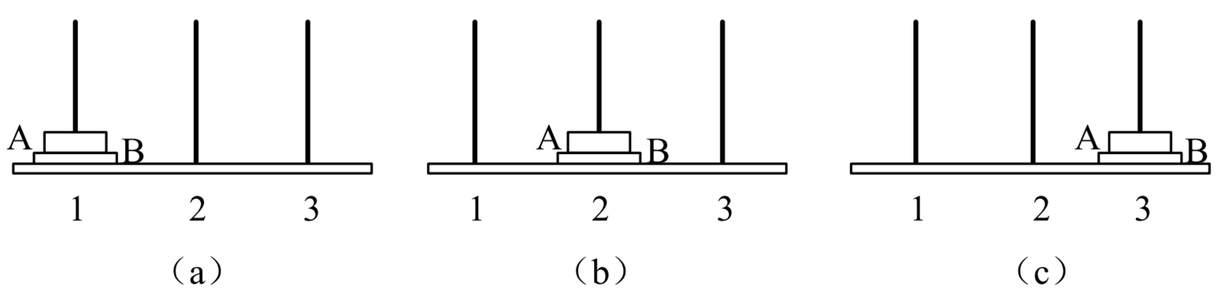


图 2.29 二阶汉诺塔图（1）

解：按照上面给出的步骤，将问题以状态空间的形式表示出来。

第一步：定义问题状态的描述形式。设 $S_k=[S_{k0},S_{k1}]$表示问题状态，其中 S_{k0} 表示圆盘 A 所在的柱子号，S_{k1} 表示圆盘 B 所在的柱子号。

第二步：用所定义的状态描述形式把问题的所有可能的状态都表示出来，并确定问题的初始状态集合描述和目标状态集合描述。本问题所有可能的状态共有 9 种，图 2.30 所示为其中的 3 种。

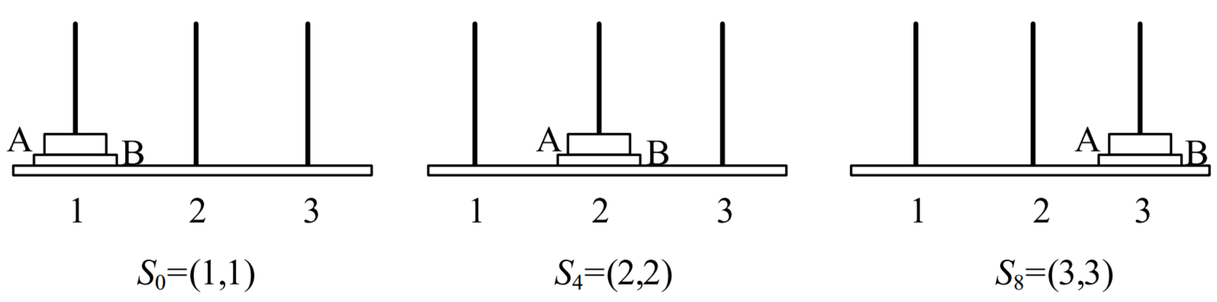


图 2.30 二阶汉诺塔图（2）

第三步：定义一组算符 F。

算符 $A(i,j)$表示把圆盘 A 从第 i 号柱子移到第 j 号柱子上的操作。

算符 $B(i,j)$ 表示把圆盘 B 从第 i 号柱子移到第 j 号柱子上的操作。

这样定义的算符集合 F 中共有 12 个算符，分别如下：

$$A(1,2)\ A(1,3)\ A(2,1)$$
$$A(2,3)\ A(3,1)\ A(3,2)$$

$B(1,2)$ $B(1,3)$ $B(2,1)$

$B(2,3)$ $B(3,1)$ $B(3,2)$

至此，该问题的状态空间（S,F,G）构造完成。这就完成了对问题的状态空间表示。

一种得到解的操作序列：$A(1,3)$，$B(1,2)$，$A(3,2)$。

操作的条件：$B(1,2)$：$S_k=(x,1)\rightarrow x\neq 1$。

操作的结果：$B(1,2)$：$S_k=(x,1)\rightarrow S_k=(x,2)$。

2.8　问题归约法

2.8.1　问题归约描述

问题归约是指从较复杂、难于求解的目标问题出发，进行分解或变换，将它转化为一系列较简单、易于求解的子问题及子问题的子问题，直至归约为一个平凡的易求解的本原问题集合。

问题归约表示的组成部分如下。

1）一个初始问题描述。

2）一套把问题变换为子问题的操作符。

3）一套本原问题描述。

子问题的描述可以采用各种数据结构，如表列、树、字符串、矢量、数组等。

问题归约是不同于状态空间的人工智能中另一种问题描述和求解的方法，问题归约表示也是一个三元组：（S_0,F,G）。

S_0：一个初始问题描述，即要解决的问题（初始问题）。

F：一套把问题变换为子问题的操作符（操作算子集）。

G：一套本原问题描述，其中的每一个问题是不证明的、自然成立的，如公理、已知的实事等（本原问题集）。

2.8.2　与图、或图、与或图表示法

（1）与图

把一个原问题分解为若干个子问题 $P_1,P_2,P_3,\cdots$可用一个与图来表示。$P_1,P_2,P_3,\cdots$对应的子问题节点称为与节点。

例如，设 P 可以分解为三个子问题 P_1、P_2、P_3 的与，则 P 和 P_1、P_2、P_3 之间的关系可用图 2.31 所示的一个与图表示。

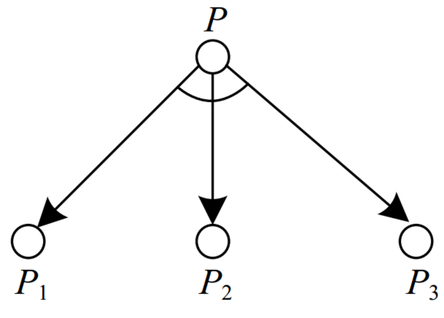


图 2.31　与图

（2）或图

把一个原问题变换为若干个子问题 $P_1,P_2,P_3,\cdots$可用一个或图来表示。子问题 $P_1,P_2,P_3,\cdots$对应的节点称为或节点。

例如，设 P 可以变换为三个子问题 P_1、P_2、P_3 的或，则 P 和 P_1、P_2、P_3 之间的关系可用图 2.32 所示的一个或图表示。

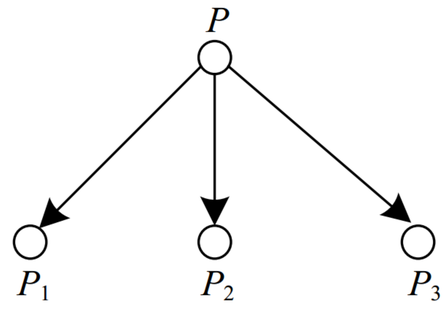


图 2.32　或图

（3）与或图

如果一个原问题既需要通过分解又需要通过变换才能得到其本原问题，则其归约过程可用一个与或图来表示，如图 2.33 所示。在特殊情况下可能不出现任何与节点，则该图变为普通图。

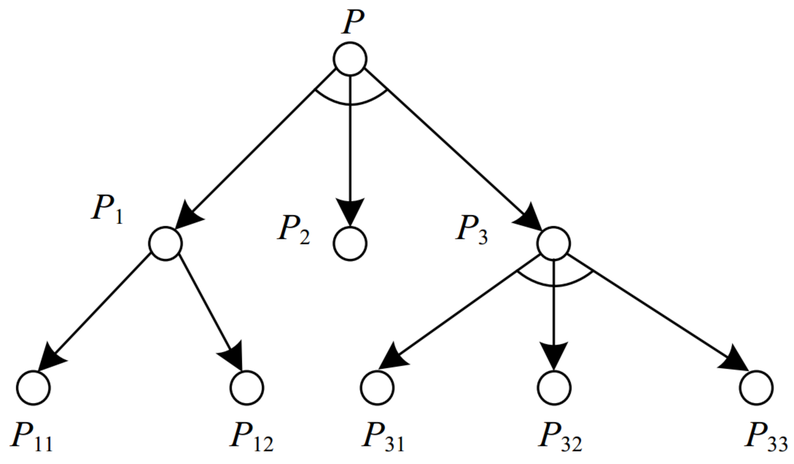


图 2.33　与或图

2.9　面向对象表示法

面向对象表示法强调客观世界中事物与事物之间的关系，因此采用面向对象的知识表示法比较接近人类的思维方式。

面向对象表示法的基本原则如下。

1）一切事物都是对象。

2）任何系统都是由对象构成的，系统本身也是对象。

3）系统的发展和进化过程是由系统的内部对象和外部对象之间（也包括内部对象与内部对象之间）的相互作用完成的。

2.9.1　面向对象的基本概念

对象是由一组数据和与该组数据相关的操作构成的实体。

在面向对象表示中，类和类继承是一组重要概念。

类由一组变量和一组操作组成，它描述了一组具有相同属性和操作的对象。每一个对象属于某一类，每个对象可由相关的类生成。换言之，对象是类的实例。一个类可以通过继承拥有另一个类的全部变量和操作，继承是面向对象表示法的主要推理形式。

由于一个事物的描述集中在一个类中，又体现了类的封装性。继承和封装是面向对象的两大特点。

（1）对象

所谓对象，从广义上来讲是指客观世界中的任何事物。它可以是一个具体的事物，也可以是由多个简单事物组合而成的复杂事物。从问题求解的角度讲，对象是与问题领域有关的客观事物。

按照对象方法学的观点，一个对象的形式可以用如下的四元组表示：

对象::=<ID,DS,MS,MI>

即一个完整的对象包括该对象的标识符（identifier，ID）、数据结构（data structure，DS）、方法集合（method of sets，MS）和消息接口（message interface，MI）。

下面分别对 ID、DS、MS、MI 进行解释。

1）ID：对象的标识符，又称对象名，用以标识一个特定的对象，如人们的身份证号或姓名一样。

2）DS：对象的数据结构，描述对象当前的内部状态或所具有的静态属性，常用一组<属性名,属性值>表示。

3）MS：对象的方法集合，用以说明对象所具有的内部处理方法或对受理消息的操作过程，它反映了对象自身的智能行为。

4）MI：对象的消息接口，是对象接收外部信息和驱动有关内部方法的唯一对外接口。这里的外部信息称为消息。

由于客观事物具有其自然属性及行为，因此与问题有关的对象也有一组数据和一组操作，且不同对象之间的相互作用可通过互传消息来实现。

（2）类

类是一种抽象机制，是对一组相似对象的抽象。具体说就是那些具有相同结构和处理能力的对象用类来描述。

一个类实际上定义了一种对象类型，它描述属于该对象类型的所有对象的性质。例如，黑白电视机、彩色电视机都是具体对象，但是它们有共同属性，于是可把它们抽象成“电视机”，“电视机”是一个类对象。各个类还可以进行进一步抽象，形成超类。例如，对电视机、电冰箱等，可以形成超类“家用电器”。这样类、超类和对象就形成了一个层次结构。该结构还可以包含更多的层次，层次越高越抽象，层次越低越具体。

（3）封装

封装是指一个对象的状态只能由它的私有操作来改变，其他对象的操作不能直接改变它的状态。

当一个对象需要改变另一个对象的状态时，它只能向该对象发送消息，该对象接收消息后根据消息的模式找出相应的操作，并执行操作改变自己的状态。

封装是一种信息隐藏技术，是面向对象方法的重要特征之一。它使对象的用户可以不了解对象行为实现的细节，只需用消息来访问对象，使面向对象的知识系统便于维护和修改。

（4）消息

消息是指在通信双方之间传递的任何书面、口头或代码的内容。

在面向对象的方法中，对对象实施操作的唯一途径就是向对象发送消息，各对象之间的联系只能通过消息发送和接收来进行。同一消息可以送往不同的对象，不同对象对于相同形式的信息可以有不同的解释和不同的反应。一个对象可以接收不同形式、不同内容的多个消息。

（5）继承

继承是指父类所具有的数据和操作可以被子类继承，除非子类对相应数据及操作重新进行了定义，称为对象之间的继承关系。

面向对象的继承关系与框架之间属性的继承关系类似，可以避免信息的冗余。

综上所述，面向对象的基本特征是：封装性、继承性和多态性。所谓多态，是指一个方法表现出多种输出结果（多种形态或者多种语义解释），运算符“+”“-”“*”“/”既可以做整数运算，也可以做实数运算，但是它们的执行代码却全然不同。

2.9.2 面向对象的知识表示

在面向对象的知识表示中，父类、子类及具体对象构成了一个层次结构，而且子类可以继承父类的数据及操作。这种层次结构及继承机制直接支持了分类知识的表示，其表示方法与框架表示法有许多相似之处，知识可以按类以一定层次形式进行组织，类之间通过链实现联系。

用面向对象方法表示知识时需要对类的构成形式进行描述，不同面向对象语言所提供的类的描述形式不同，下面给出一种描述形式：

```
Class<类名>    [：<父类名>]
               [<类变量表>]
               Structrue
               <对象的静态结构描述>
               Method
               <关于对象的操作定义>
               Restraint
               <限制条件>
  EndClass
```

【说明】

Class：类描述的开始标志。

<类名>：该类的名字，它是系统中该类的唯一标识。

<父类名>：是任选项，指出当前定义的类的父类，可以缺省。

<类变量表>：是一组由变量名构成的序列，该类中的所有对象都共享这些变量，对该类对象来说，它们是全局变量。当把这些变量实例化为一组具体的值时，就得到了该类中的一个具体对象，即一个实例。

Structrue：后面的<对象的静态结构描述>用于描述该类对象的构成方式。

Method：后面的<关于对象的操作定义>用于定义对类元素可实施的各种操作。它既可以是一组规则，也可以是为了实现相应操作所需执行的一段程序。

Restraint：后面的<限制条件>指出该类元素应该满足的限制条件，可用包含类变量的谓词构成，不出现时表示没有限制。

EndClass：最后以 EndClass 结束类的描述。

面向对象方法学的主要观点如下。

1）认为世界由各种“对象”组成，任何事物都是对象，是某对象类的元素；复杂的对象可由相对比较简单的对象以某种方法组成；甚至整个世界也可从一些最原始的对象开始，层层组合而成。

2）所有对象被分成各种对象类，每个对象类定义所谓的“方法”，它们实际上可以被视为允许作用于该类对象上的各种操作。

3）对象之间除互相传递消息的联系之外，不再有其他联系，一切局部对象的信息和实现方法等都被封装在相应对象类的定义之中，在外面是看不见的，这便是封装的概念。

4）对象类将按“类”、“子类”与“超类”等概念形成一种层次关系（或树形结构）。在这个层次结构中，上一层对象所具有的一些属性或特征可被下一层对象继承，除非在下一层对象中对相应的属性做了重新描述（这时以新属性值为准），从而避免了描述中信息的冗余，这称为对象类之间的继承关系。

本 章 小 结

知识表示是指将知识载体中的知识因子与知识关联起来，便于人们识别和理解知识。知识表示的本质是将人类知识形式化或模型化。

本章首先给出了知识表示的概念，包含知识表示的基本问题和主要过程，并对知识表示的作用进行了描述，然后分别对数理逻辑表示法、语义网络表示法、过程表示法、框架表示法、产生式表示法、状态空间表示法、问题归约法和面向对象表示法进行了详细介绍。

本 章 习 题

1．设有如下语句，请用相应的谓词公式分别把它们表示出来。

1）有的人喜欢梅花，有的人喜欢菊花，有的人既喜欢梅花又喜欢菊花。

2）有人每天下午都去打篮球。

3）新型计算机速度又快，存储容量又大。

4）不是每个计算机系的学生都喜欢在计算机上编写程序。

5）凡是喜欢编写程序的人都喜欢计算机。

2．用谓词表示法求解农夫、狼、山羊、白菜问题。农夫、狼、山羊、白菜全部在一条河的左岸，现在要把他们送到河的右岸去，农夫有一条船，过河时，除农夫之外，船上至多能载狼、山羊、白菜中的一种。狼要吃山羊，山羊要吃白菜，除非农夫在岸上。拟规划出一个确保安全过河的计划。请写出所用谓词的定义，并给出每个谓词的功能及变量的个体域。

3．用状态空间法求解修道士和野人问题。设在河的左岸有三个野人、三个修道士和一条船，修道士想用这条船把所有的人运到河的右岸，但受以下条件的约束：

1）修道士和野人都会划船。

2）船每次至多可载两个人。

3）无论在河的左岸还是右岸，如果野人数目超过修道士数目，修道士就会被野人吃掉。

假设野人会服从任何一次过河安排，请规划一个确保修道士和野人都能过河，且没有修道士被野人吃掉的安全过河计划。请写出其状态、算符、操作结果。

4．用问题归约法求解三阶汉诺塔问题。设有三根柱子，它们的编号分别是 1 号、2 号、3 号。在初始情况下，1 号柱子上穿有 A、B、C 三个圆盘，A 比 B 小，B 比 C 小，A 位于 B 的上面，B 位于 C 的上面。要求把这三个圆盘全部移到 3 号柱子上，而且规定每次只能移动一个圆盘，任何时刻都不能使大圆盘位于小圆盘的上面。画出其状态空间图节点。

5．用一个语义网络来表示如下事实：苹果树是一种果树，果树又是树的一种，树有根、有叶，且树是一种植物。

6．把下列语句表示成语义网络描述。

1）All man are mortal.

2）Every cloud has a silver lining.

3）All branch managers of DEC participate in a profit-sharing plan.

7．按照“雇佣框架”“雇主框架”“雇员框架”的形式写出一个框架系统的描述。

8．假设一个用以识别虎、斑马、长颈鹿、企鹅、鸵鸟、孔雀等动物的产生式系统，若实现对这些动物的识别，应该如何建立规则库？

9．以一场学术报告为例进行脚本的知识表示描述。

10．假设有以下一段天气预报：北京地区今天白天晴，偏北风 3 级，最高气温 12℃，最低气温-2℃，降水概率 15%。请用框架表示这一知识。

第3章　搜 索 求 解

3.1　搜 索 概 述

在求解一个问题时，涉及两个方面：一是该问题的表示，如果一个问题找不到一个合适的表示方法，就谈不上对它求解；二是选择一种相对合适的求解方法。在人工智能学习中，问题求解的基本方法有搜索法、归约法、归结法、推理法及产生式等。由于绝大多数需要用人工智能方法求解的问题缺乏直接求解的方法，因此，搜索不失为一种求解问题的一般方法。搜索求解的应用非常广泛，如在下棋等游戏软件中的应用。

3.1.1　搜索的基本问题与主要过程

搜索中需要解决的基本问题如下。

1）搜索过程是否一定能够找到一个解。

2）搜索过程是会终止运行或是陷入一个死循环。

3）当搜索过程找到一个解时，找到的是否为最佳解。

4）搜索过程的时间与空间复杂性如何。

搜索的主要过程具体如下。

1）从初始或目的状态出发，并将它作为当前状态。

2）扫描操作算子集，将适用当前状态的一些操作算子作用于当前状态而得到新的状态，并建立指向其父节点的指针。

3）检查所生成的新状态是否满足结束状态，如果满足，则得到问题的一个解，并可沿着有关指针从结束状态反向到达开始状态，给出解答路径；否则，将新状态作为当前状态，返回第 2）步再进行搜索。

3.1.2　搜索策略

搜索策略直接关系到智能系统的性能和运行效率。搜索问题中，主要的工作是找到正确的搜索策略。搜索策略反映了状态空间或问题空间扩展的方法，也决定了状态或问题的访问顺序。由于搜索策略的不同，人工智能中的搜索问题的命名也不同。

1．搜索的方向

（1）正向搜索

从初始状态出发的正向搜索，称为数据驱动。正向搜索是从问题给出的条件——一

个用于状态转换的操作算子集合出发的。搜索过程如下：应用操作算子从给定的条件中产生新条件，再用操作算子从新条件中产生更多的新条件，这个过程一直持续到有一条满足目的要求的路径产生为止。数据驱动就是用问题给定数据中的约束知识指导搜索，使其沿着那些已知是正确的线路前进。

（2）逆向搜索

从目的状态出发的逆向搜索，称为目的驱动。逆向搜索是先从想达到的目的入手，看哪些操作算子能够产生该目的以及应用这些操作算子产生该目的时需要哪些条件，这些条件就成为要达到的新目的，即子目的。逆向搜索就是通过不断产生子目的，直至所产生的子目的需要的条件为问题给定的条件为止。这样就找到了一条从数据到目的的操作算子所组成的链。

（3）双向搜索

结合上述两种方式的搜索称为双向搜索，即从开始状态出发做正向搜索，同时又从目的状态出发做逆向搜索，直到两条路径在中间的某处汇合为止。

（4）盲目搜索与启发式搜索

根据搜索过程中是否运用与问题有关的信息，可以将搜索方法分为盲目搜索（blind search）和启发式搜索（heuristic search）。

盲目搜索是指在对特定问题不具有任何有关信息的条件下，按照固定的步骤（依次或随机调用操作算子）进行的搜索，它能够快速地调用一个操作算子。

启发式搜索是考虑特定问题领域可应用的知识，动态地确定调用操作算子的步骤，优先选择较适合的操作算子，尽量减少不必要的搜索，以求尽快到达结束状态，提高搜索效率。

在盲目搜索中，由于没有可参考的信息，只要能够匹配的操作算子都需要运用，从而搜索出更多的状态，生成较大的状态空间显示图；在启发式搜索中，运用一些启发信息，只采用少量的操作算子，生成较小的状态空间显示图，就能搜索到一个解，但是每使用一个操作算子，需要做更多的计算与判断。启发式搜索一般优于盲目搜索，但是不可过于追求更多的甚至完整的启发信息。

2．人工智能领域中主要的搜索策略

1）求任一解的搜索策略：爬山法（hill climbing）、深度优先法（depth-first search）、限定范围搜索法（beam search）、回溯法（back-tracking）、最好优先法（best-first search）。

2）求最佳解的搜索策略：大英博物馆法（British museum）、宽度优先法（breadth-first search）、分支定界法（branch and bound）、动态规划法（dynamic programming）、最佳图搜索法（A*）。

3）求与或关系解图的搜索法：一般的与或图解搜索法（AO*）、极大极小法（minimax）、α-β 剪枝法（alpha-beta pruning）、启发式剪枝法（heuristic pruning）。

3.2　状态空间的搜索策略

3.2.1　状态空间表示法

状态空间（state space）表示法是知识表示的一种基本方法。

状态是用来表示系统状态、事实等叙述型知识的一组变量或数组，表示如下：

$$Q=[q_1,q_2,\cdots,q_n]^{\mathrm{T}} \tag{3.1}$$

操作是用来表示引起状态变化的过程型知识的一组关系或函数，表示如下：

$$F=\{f_1,f_2,\cdots,f_m\} \tag{3.2}$$

状态空间是利用状态变量和操作符号，表示系统或问题的有关知识的符号体系。状态空间可以用一个四元组表示：

$$(S,O,S_0,G)$$

式中，S 是状态集合，S 中的每一个元素表示一个状态，状态是某种结构的符号或数据；O 是操作算子的集合，利用算子可将一个状态转换为另一个状态；S_0 是包含问题的初始状态，是 S 的非空子集，$S_0\subset S$；G 是包含问题的目的状态，是 S 的非空子集，$G\subset S$，G 可以是若干具体状态，也可以是满足某些性质的路径信息描述。

从 S_0 节点到 G 节点的路径称为求解路径。求解路径上的操作算子序列为状态空间的一个解。例如，操作算子序列 $O_1,O_2,\cdots,O_k$ 使初始状态转换为目标状态：$S_0\xrightarrow{O_1}S_1\xrightarrow{O_2}S_2\xrightarrow{O_3}\cdots\xrightarrow{O_k}G$，则 $O_1,O_2,\cdots,O_k$ 即为状态空间的一个解。当然，解往往不是唯一的。

任何类型的数据结构都可以用来描述状态，如符号、字符串、向量、多维数组、树和表格等。所选用的数据结构形式需要与状态所蕴含的某些特性具有相似性。例如，对于八数码问题，一个 3×3 的阵列便是一个合适的状态描述方式。

3.2.2　状态空间的图描述

状态空间可用有向图来描述，图的节点表示问题的状态，图上的弧线表示状态之间的关系，就是求解问题的步骤。初始状态对应于实际问题的已知信息，是图中的根节点。在问题的状态空间描述中，寻找从一种状态转换为另一种状态的某个操作算子序列就等价于在一个图中寻找某一路径。

图 3.1 所示为用有向图描述的状态空间。图中表示，针对状态 S_0，允许使用操作算子 O_1、O_2 及 O_3，分别使 S_0 转换为 S_1、S_2 及 S_3。这样一步步利用操作算子转换下去，如 $S_{10}\subset G$，则 O_2、O_6 及 O_{10} 就是一个解。

在某些问题中，各种操作算子的执行是有不同费用的。例如，在旅行商问题中，两两城市之间的距离通常是不相等的，那么，在图中只需要给各弧线标注距离或费用即可。

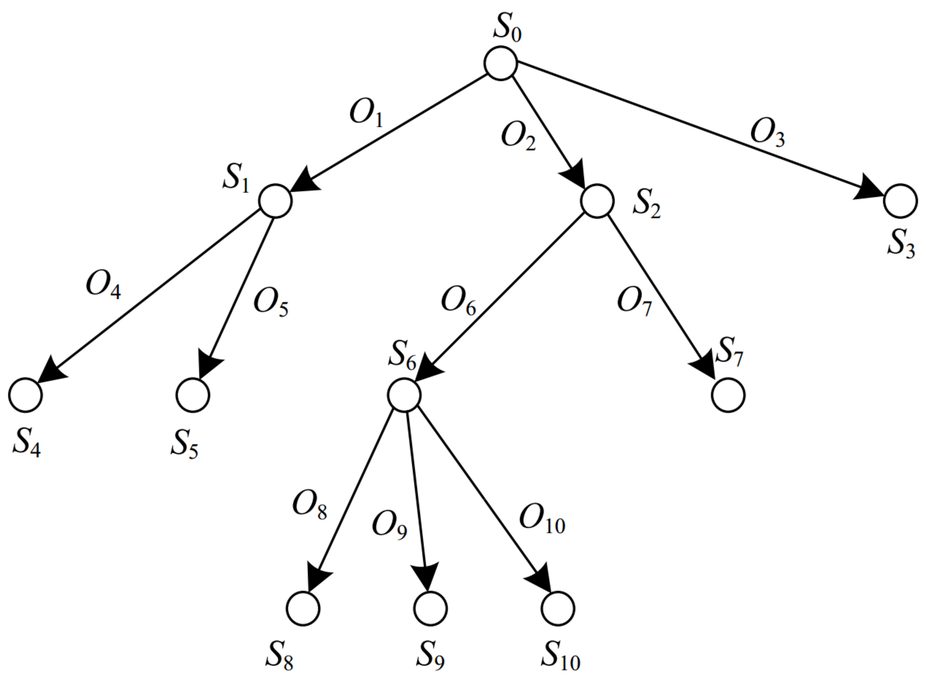


图 3.1　状态空间的有向图描述

3.3　盲目的图搜索策略

3.3.1　回溯策略

不管是正向搜索还是逆向搜索，都是在状态空间图中找到从初始状态到目的状态的路径。路径上弧线的序列对应于解题的步骤。若在选择操作算子求解问题时，能够给出绝对可靠的预测或者绝对正确的选择策略，一次性成功穿过状态空间而达到目的，构造出一条解题路径，那么就不需要所谓的搜索了。但是事实上，不可能给出绝对可靠的预测，求解实际问题时必须尝试多条路径直到找到目标为止。回溯策略是一种系统地尝试状态空间中各种不同路径的技术。

回溯策略的搜索是从初始状态出发，不停地、试探性地寻找路径，直到它到达目的地或“不可解节点”，即“死胡同”为止。

回溯策略是当遇到不可解节点时就回溯到路径中最近的父节点上，查看该节点是否还有其他的子节点未被扩展。若有，则沿着这些子节点继续搜索；如果找到目标，就成功退出搜索，返回解题路径。

可以看出，回溯策略的搜索过程呈现出递归过程的性质，搜索过程在每个节点上的检查遵循递归方式。下面给出递归过程：

```
Step Track(DataList);
Data : = First(DataList);                          *当前状态为状态序列表中第一个状态
if Member(Data, Tail(DataList))
      then return FAIL;                                    *回老路，退回
if Goal(Data)then return NIL;                              *到达目的地，成功返回
if DeadEnd(Data) then return FAIL;                         *到达不合理状态，退回
if Length(DataList)> Bound then return FAIL;               *已到深度限制，退回
Rules: =AppRules(Data);                                    *得出可应用的规则集
```

```
Loop:
     if Null(Rules)then return FAIL;                    *进入死胡同，退回
     R: = First(Rules);                                 *取出第一条可用规则
     Rules: = Tail(Rules);
     Newdata:  = Gen(R,Data);                           *运用规则，生成新状态
     NewDataList: = Cons(Newdata, DataList);
     Path: = Back Track(NewDataList);                   *递归
     If Path: = FAIL then go Loop
     else return Cons(R, Path);
```

说明：变量符号 Data、DataList、Rules、R、Newdata、NewDataList、Path 分别表示当前状态、状态序列表、规则集序列表、当前运用的规则、生成的新状态、生成的新状态序列表、当前解路径表；常量符号 FAIL、NIL、Bound、Loop 分别表示回溯点标记、空表、深度限制值、循环标号；函数 First(y)表示在 y 表中取第一个元素，函数 Tail(x)是取 x 表中除了第一个元素的其余部分，函数 Member(x,y)表示变量 x 是否为 y 表的一个元素，函数 Goal(x)表示变量状态 x 是否为目的状态，函数 DeadEnd(x)表示变量状态 x 是否为不可解路径上的状态，函数 Length(x)是取 x 表的表长度，函数 AppRules(x)是求出在变量状态 x 上可运用的规则集，函数 Null(x)表示 x 表是否为空表，函数 Gen(x,y)是求 x 规则运用在 y 状态后所生成的新状态，函数 Cons(x,y)是将变量状态 x 加在 y 表的前部。

从递归过程可以看出，若当前状态 S 未达到目的的要求，就对它的第一个子状态 S_{child1} 递归调用回溯过程。如果在以 S_{child1} 为根的子图中未找到目标，就对它的兄弟子状态 S_{child2} 递归调用回溯过程。这样重复进行，直到某个子状态的后裔是目的状态或者所有子状态都搜索完为止。若 S 的子状态中没有一个能达到目的，则回溯到 S 的父状态。这样，算法就可以对 S 的兄弟状态进行搜索。算法以这种方式搜索，直到找到目的状态或遍历了状态空间为止。图 3.2 给出了一个状态空间中应用回溯搜索的示意过程，图中虚线箭头的方向表示搜索的轨迹，节点边的数字表示被搜索到的次序。

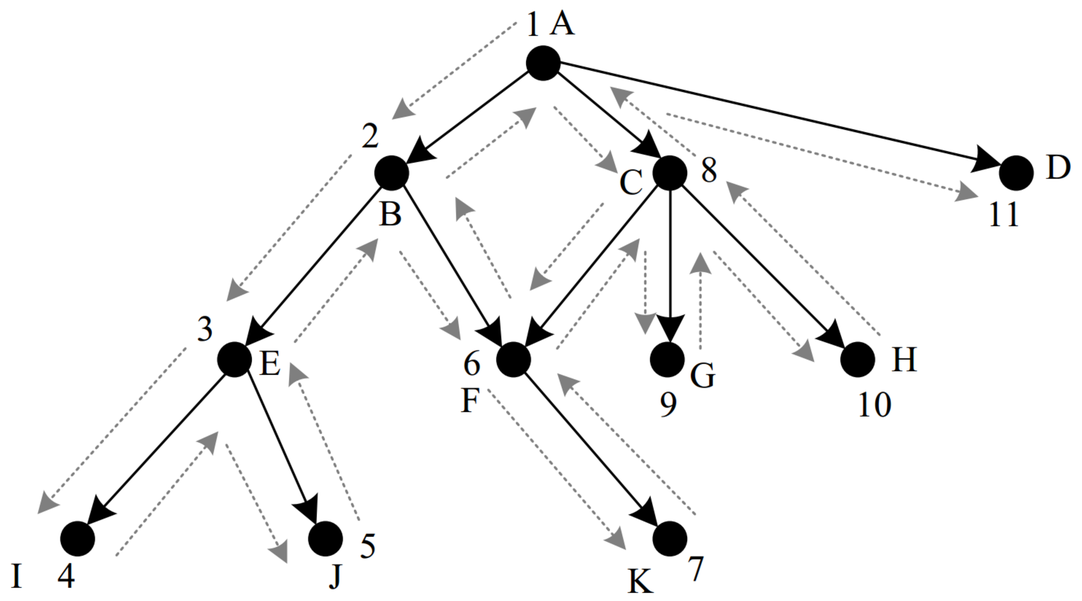


图 3.2　回溯搜索示意图

至此，可以定义一个回溯搜索的算法。

算法用三张表来保存状态空间中的不同性质节点。

（1）路径状态表（path states，PS）

PS 保存当前搜索路径上的状态。如果找到了目标，PS 就是解路径上的状态有序集。

（2）新的路径状态表（new path states，NPS）

NPS 包含了等待搜索的状态，其后裔状态还未被搜索到，即未被生成扩展。

（3）不可解状态表（no solvable states，NSS）

NSS 列出了找不到解路径的状态。如果在搜索中扩展出的状态是它的元素，则可立即将其排除，不必沿该状态继续搜索。

为了避免造成无穷循环搜索，需要检测并删除多次出现的那些状态。具体检测可通过判断每一个新生成的状态是否在 PS、NPS、NSS 三张表中来实现。如果它属于其中一张表，就说明它已被搜索过而不必再考虑。

当前正在被检测的状态记作当前状态（current state，CS）。CS 总是等于最近加入 PS 的状态，是当前正在探寻解题路径的“前锋”。各种合适的推理规则或其他问题求解操作都可应用于 PS。应用后一般得到一些新状态，即 PS 的子状态的有序集，然后再将该集合中的第一个子状态作为当前状态（CS），并加入 PS 中，其余的则按序放入 NPS 中，用于以后的搜索。如果应用后 CS 没有子状态，则要从 PS、NPS 中删除它，同时将其加入 NSS，之后回溯查找 NPS 中表首位置的状态。下面给出回溯过程：

```
Function backtrack :
begin
PS: =[Start];NPS: =[Start];NSS: =[ ];CS: = Start;   *初始化
    while NPS≠[ ] do
      begin
      if CS=目的状态 then return（PS）;                    *成功，返回解题路径
      if CS 没有子状态（不包括 PS、NPS 和 NSS 中已有的状态）
         then
            begin

               while（（PS 非空）and（CS=PS 中的第一个元素））do
                     begin
                        将 CS 加入 NSS                      *标明此状态不可解
                        从 PS 中删除第一个元素 CS           *回溯
                        从 NPS 中删除第一个元素 CS;
                        CS: =NPS 中的第一个元素;
                  end;
                        将 CS 加入 PS;
               end
         else
            begin
                  将 CS 子状态（不包括 PS、NPS 和 NSS 中已有的状态）加入 NPS;
                  CS: = NPS 中的第一个元素;
                  将 CS 加入 PS;
            end
         end;
      return FAIL;
end
```

图 3.2 的回溯轨迹如表 3.1 所示。

表 3.1　图 3.2 回溯轨迹表

序列	CS	PS	NPS	NSS
0	A	[A]	[A]	[ ]
1	B	[BA]	[BCDA]	[ ]
2	E	[EBA]	[EFBDCA]	[ ]
3	I	[IEBA]	[IJEFBCDA]	[ ]
4	J	[JEBA]	[JEFBCDA]	[I]
5	F	[FBA]	[FBCDA]	[EJI]
6	K	[KFBA]	[KFBCDA]	[EJI]
7	C	[CA]	[CDA]	[BFKEJI]
8	G	[GCA]	[GHCDA]	[BFKEJI]

初值：PS：=[A]；NPS：=[A]；NSS：=[ ]；CS：=Start[A]。

上面的搜索过程显示出回溯是状态空间中的一个正向搜索。它将初始条件作为初始状态，对其子状态进行搜索以寻找目的。若将目的作为搜索图的根，即初始状态，则本算法可看作逆向搜索。如果将算法中“成功并返回解路径”的判别条件“CS=目的状态”修改为“搜索路径的性质优劣”，那么算法必须通过检查 PS 中的路径来确定是否到达目的地。

回溯是状态空间搜索的一个基本算法。各种图搜索算法（包括深度优先、宽度优先、最好优先搜索等）都有回溯的思想。

1）用新的路径状态表（NPS）使算法能返回（回溯）到其中任一状态。

2）用不可解状态表（NSS）来避免算法重新搜索无解的路径。

3）在路径状态表（PS）中记录当前搜索路径的状态，当满足目的时可以将它作为结果返回。

4）为避免陷入死循环，必须对新生成的子状态进行检查，看它是否在该三张表中。

3.3.2　宽度优先搜索策略

如果搜索是以接近起始节点的程度依次扩展节点的，那么这种搜索就叫作宽度优先搜索（breadth-first search），如图 3.3 所示。这种搜索是逐层进行的，在对下一层的任一节点进行搜索之前，必须搜索完本层的所有节点。

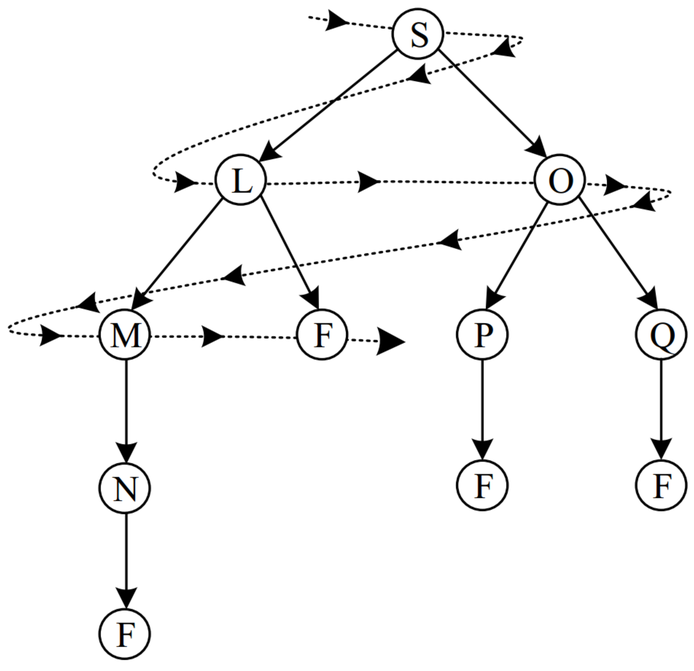


图 3.3　宽度优先搜索示意图

在图搜索过程中涉及的数据结构除了图本身外，还需要两个辅助的数据结构，即存放已访问但未扩展节点的 OPEN 表，以及存放已扩展节点的 CLOSED 表。宽度优先搜索算法如下。

1）把起始节点放到 OPEN 表中（如果该起始节点为一目标节点，则求得一个解）。

2）如果 OPEN 是个空表，则没有解，失败退出；否则继续。

3）把第一个节点（节点 n）从 OPEN 表中移出，并把它放入 CLOSED 的扩展节点表中。

4）扩展节点 n。如果没有后继节点，则转向第 2）步。

5）把 n 的所有后继节点放到 OPEN 表的末端，并提供从这些后继节点回到 n 的指针。

6）如果 n 的任何一个后继节点是个目标节点，则找到一个解，成功退出；否则转向第 2）步。

上述宽度优先算法的流程如图 3.4 所示。

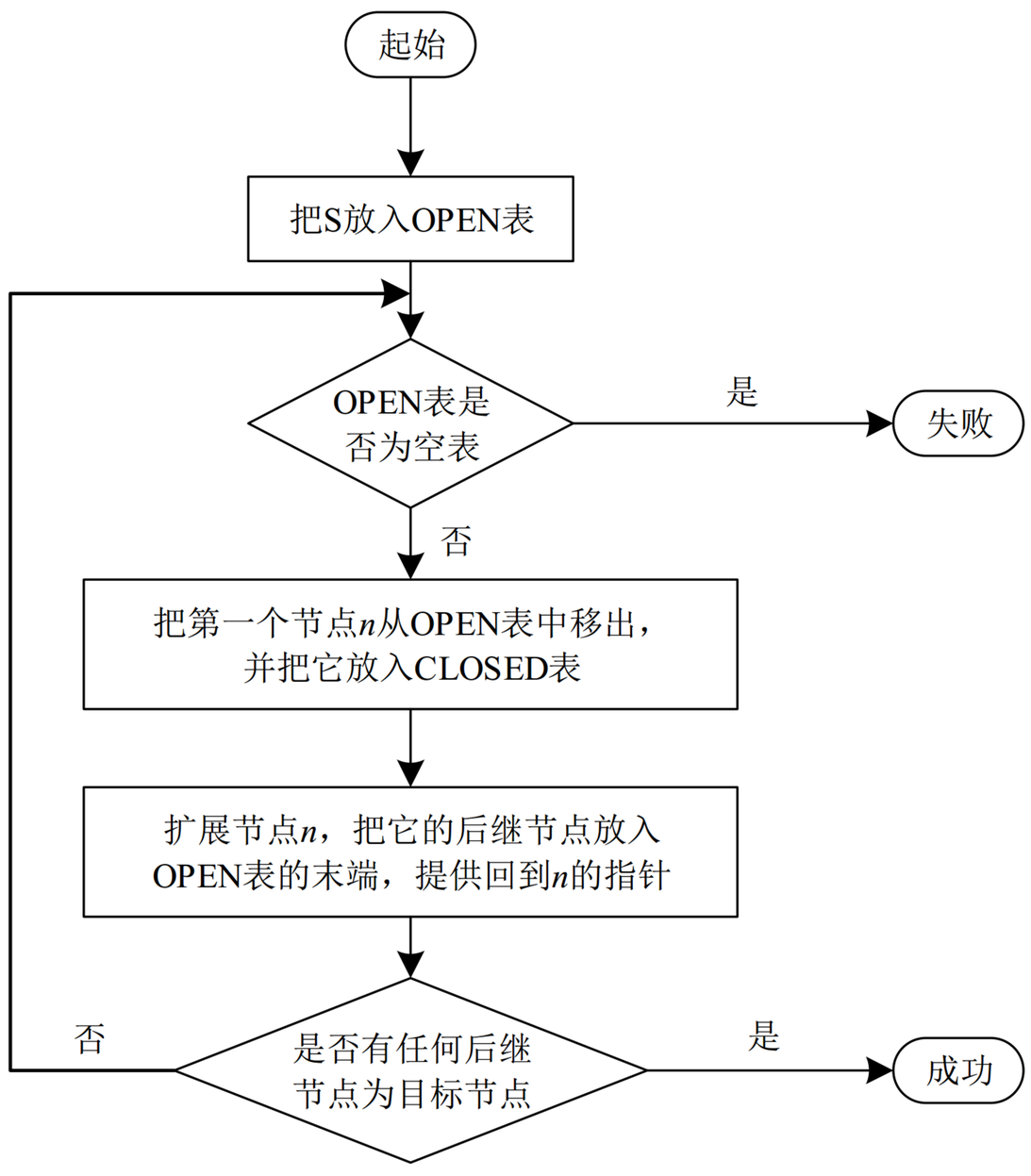


图 3.4　宽度优先算法流程图

3.3.3　深度优先搜索策略

另一种盲目（无信息）搜索叫作深度优先搜索（depth-first search）。在深度优先搜索中，首先扩展最新产生的（即最深的）节点，如图 3.5 所示。深度相等的节点可以任意排列。定义节点的深度如下。

1）起始节点（即根节点）的深度为 0。

2）任何其他节点的深度等于其父节点深度加 1。

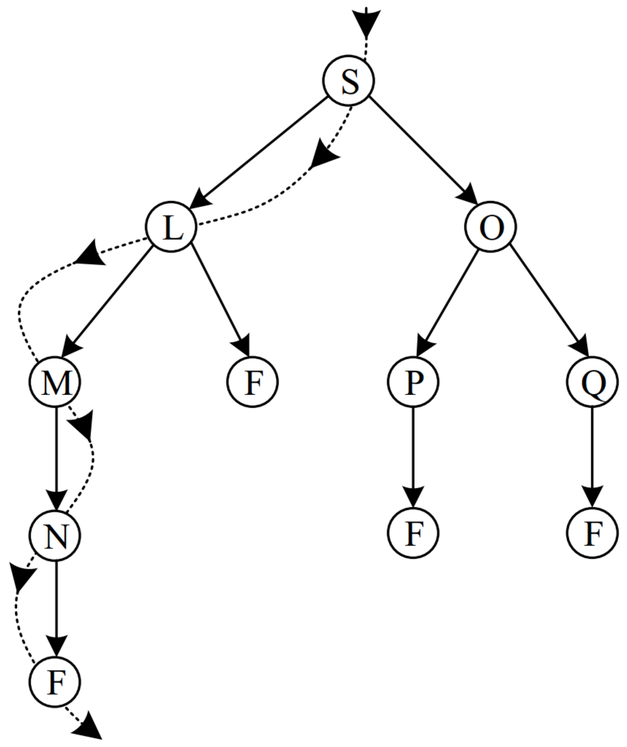


图 3.5　深度优先搜索示意图

首先，扩展最深的节点的结果使得搜索沿着状态空间某条单一的路径从起始节点向下进行；只有当搜索到达一个没有后裔的状态时，它才考虑另一条替代的路径。替代路径与前面已经试过的路径的不同之处仅仅在于改变最后 n 步，而且保持 n 尽可能小。

对于许多问题，其状态空间搜索树的深度可能为无限深，至少要比某个可接受的解答序列的已知深度上限还要深。为了避免考虑太长的路径（防止搜索过程沿着无益的路径扩展下去），往往给出一个节点扩展的最大深度——深度界限。任何节点如果达到了深度界限，那么都将把它们作为没有后继节点处理。需要说明的是，即使应用了深度界限的规定，所求得的解答路径并不一定就是最短的路径。

含有深度界限的深度优先搜索算法如下。

1）把起始节点 S 放到未扩展节点 OPEN 表中。如果此节点为一目标节点，则得到一个解。

2）如果 OPEN 为一个空表，则失败退出；否则继续。

3）把第一个节点（节点 n）从 OPEN 表移到 CLOSED 表。

4）如果节点 n 的深度等于最大深度，则转向第 2）步；否则继续。

5）扩展节点 n，产生其全部后裔，并把它们放入 OPEN 表的前部。如果没有后裔，则转向第 2）步。

6）如果后继节点中有任何一个为目标节点，则求得一个解，成功退出；否则，转向第 2）步。

有界深度优先搜索算法的程序流程图如图 3.6 所示。显然，深度优先算法中节点进出 OPEN 表的顺序是后进先出。OPEN 表是一个栈。

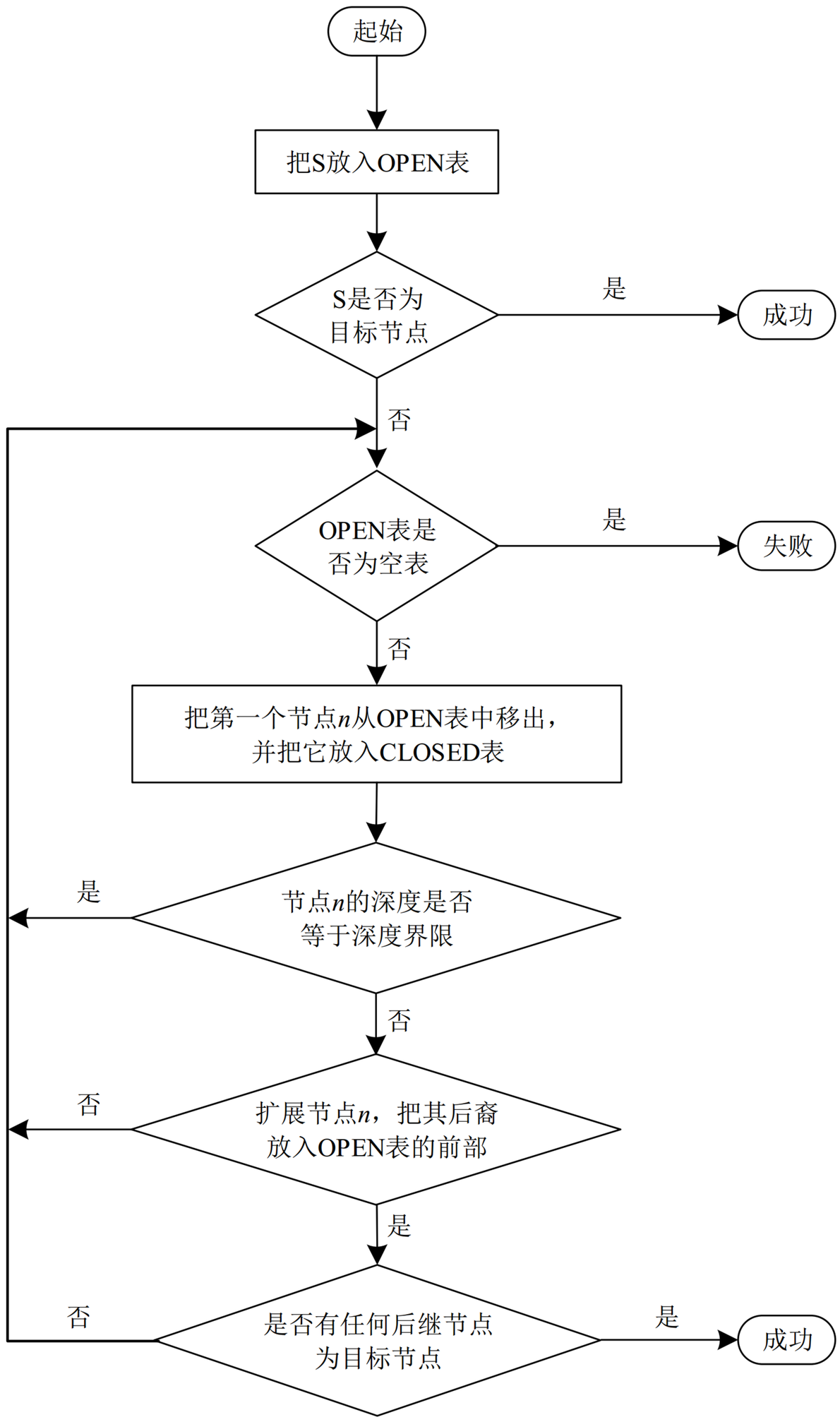


图 3.6　有界深度优先搜索算法流程图

3.4　启发式图搜索策略

上一节所介绍的大部分搜索方法都是盲目搜索方法，其搜索的复杂性往往是很高的。为了提高算法的效率，必须放弃利用纯数学的方法来决定搜索节点的次序，而需要对具体问题做具体分析，利用与问题有关的信息，从中获得启发以引导搜索，进而达到减少搜索量的目的，这就是启发式搜索。

本节首先对启发及启发式策略所涉及的问题进行介绍，然后具体介绍启发式图搜索算法——A 搜索算法及 A*搜索算法。

3.4.1　启发式策略

启发式（heuristic）策略是利用与问题有关的启发信息引导搜索。

在状态空间搜索中，启发式被定义为一系列操作算子，并能从状态空间中选择最有希望到达问题解的路径。

问题求解系统可在两种基本情况下运用启发式策略。

1）由于在问题陈述和数据获取方面存在模糊性，可能会使问题没有一个确定的解，这就要求系统能够运用启发式策略做出最有可能的解释。

2）虽然一个问题可能有确定解，但是其状态空间特别大，搜索中生成扩展的状态数会随着搜索的深度呈指数级增长。穷尽式搜索策略（如宽度优先或深度优先搜索）在一个给定的较实际的时空内很可能得不到最终的解，启发式策略通过引导搜索向最有希望的方向进行来降低搜索复杂度。

但是，启发式策略也是极易出错的。在解决问题的过程中，启发仅仅是下一步将要采取措施的一个猜想，它常常根据经验和直觉来判断。由于启发式搜索只利用特定问题的有限信息，因此很难准确地预测下一步在状态空间中采取的具体的搜索行为。一个启发式搜索可能得到一个次优解，也可能一无所获。这是启发式搜索固有的局限性，而这种局限性不可能由所谓更好的启发式策略或更有效的搜索算法来彻底消除。

在问题求解中，需要用启发式知识来剪枝以减少状态空间的大小，否则只能求解一些小规模问题。因此，启发式搜索策略及算法设计一直是人工智能的核心问题。

启发式搜索通常由两部分组成：启发方法和使用该方法搜索状态空间的算法。

3.4.2　启发信息和估价函数

在解决一个实际问题时，人们常常把一个复杂的实际问题抽象化，保留某些主要因素，忽略大量次要因素，从而将这个实际问题转化成具有明确结构的有限或无限的状态空间问题。这个状态空间中的状态和变换规律都是已知的集合，因此可以找到一个求解该问题的算法。

在具体求解中，启发式搜索能够利用与该问题有关的信息来简化搜索过程，称此类信息为启发信息。然而，在求解问题中能利用的大多不是具有完备性的启发信息，而是非完备的启发信息。其原因如下。

1）大多数情况下，求解问题系统不可能知道与实际问题有关的全部信息，因而无法知道该问题的全部状态空间，也不可能用一套算法来求解所有的问题。这样就只能依靠部分状态空间、一些特殊的经验和有关信息来求解其中的部分问题。

2）有些问题在理论上虽然存在着求解算法，但是在工程实践中，这些算法不是效率太低，就是无法实现。为了提高求解问题的效率，不得不放弃使用这些“完美”的算法，而求助于一些启发信息来进行启发式搜索。

启发信息按照运用的方法不同可分为以下三种。

1）陈述性启发信息。一般被用于更准确、更精练地描述状态，使问题的状态空间缩小，如待求问题的特定状况等属于此类信息。

2）过程性启发信息。一般被用于构造操作算子，使操作算子少而精，如一些规律性知识等属于此类信息。

3）控制性启发信息。它是表示控制策略方面的知识，包括协调整个问题求解过程中所使用的各种处理方法、搜索策略、控制结构等有关的知识。

为了提高搜索效率，需要利用上述三种启发信息作为搜索的辅助性策略。这里主要介绍控制性的启发信息。

利用控制性的启发信息有两种极端的情况：一种是没有任何控制性知识作为搜索的依据，因而搜索的每一步完全是随意的，如随机搜索、宽度搜索、深度搜索等；另一种是有充分的控制知识作为依据，因而搜索的每一步选择都是正确的，但是这是不现实的。一般情况介于二者之间。在搜索过程中需要根据这些启发信息估计各个节点的重要性。

用估价函数（evaluation function）估计待搜索节点的“有希望”程度，并依次给它们排定次序。估价函数可以是任意一种函数，如定义节点 n 为处于最佳路径上的概率，或是节点 n 和目的节点之间的距离或差异等。一般来说，估计一个节点的价值，必须综合考虑两方面的因素：已经付出的代价和将要付出的代价。因此，估价函数定义为从初始节点经过节点 n 到达目的节点的路径的最小代价估计值，其一般形式为

$$f(n)=g(n)+h(n) \tag{3.3}$$

式中，$g(n)$是从初始节点到节点 n 的实际代价，而 $h(n)$是从节点 n 到目的节点的最佳路径的估计代价。因为实际代价 $g(n)$可以根据已生成的搜索树实际计算出来，而估计代价 $h(n)$是对未生成的搜索路径做某种经验性的估计。这种估计来源于对问题解的某些特性的认识，希望依靠这些特性来更快地找到问题的解，因此，主要是 $h(n)$体现了搜索的启发信息。$h(n)$称为启发函数。

$g(n)$的作用一般是不可忽略的，因为它代表了从初始节点经过节点 n 到达目的节点的总代价估值中实际已付出的那一部分。保持 $g(n)$项就保持了搜索的宽度优先成分，$g(n)$的比重越大，越倾向于宽度优先搜索方式。这有利于搜索的完备性，但是会影响搜索的效率。$h(n)$的比重越大，表示启发性能越强。在特殊情况下，如果只希望找到达到目的节点的路径而不关心会付出什么代价，则 $g(n)$的作用可以忽略。另外，当 $h(n)\gg g(n)$时，也可忽略 $g(n)$，这时有 $f(n)=h(n)$，有利于提高搜索的效率，但是影响搜索的完备性。

3.4.3 A 搜索算法

A 搜索算法是一种常见且应用广泛的启发式图搜索算法。

启发式图搜索算法的关键是如何寻找并设计一个与问题有关的 $h(n)$及构造出 $f(n)=g(n)+h(n)$，然后以 $f(n)$的大小来排列待扩展状态的次序，每次选择 $f(n)$值最小者进行扩展。

与宽度优先及深度优先搜索算法一样，启发式图搜索算法使用两张表记录状态信息：在 OPEN 表中保留所有已生成而未扩展的状态；在 CLOSED 表中记录已扩展过的状态。算法中有一步是根据某些启发信息来排列 OPEN 表，它既不同于宽度优先所使用的队列（先进先出），也不同于深度优先所使用的堆栈（先进后出），而是一个按照状态的启发估价函数值的大小来排列的。进入 OPEN 表的状态不是简单地排在队尾（或

队首），而是根据其估值的大小插入表中合适的位置。每次从表中优先取出启发估价函数值最小的状态加以扩展。

A 搜索算法是基于估价函数的一种加权启发式图搜索算法，具体步骤如下。

步骤 1　把附有 $f(S_0)$的初始节点 S_0 放入 OPEN 表。

步骤 2　若 OPEN 表为空，则搜索失败，退出。

步骤 3　移出 OPEN 表中第一个节点 N 放入 CLOSED 表中，并顺序编号 n。

步骤 4　若目标节点上附有 $f(S_0)$的初始节点 $S_0=N$，则搜索成功，结束。

步骤 5　若 N 不可扩展，则转步骤 2。

步骤 6　扩展 N，对任意节点 x 生成一组附有 $f(x)$的子节点，对这组子节点做如下处理。

1）考查是否有已在 OPEN 表或 CLOSED 表中存在的节点。若有，则再考查其中有无 N 的先辈节点，若有则删除，对于其余节点也删除，但是由于它们又被第二次生成，因此需要考虑是否修改已经存在于 OPEN 表或 CLOSED 表中的这些节点及其后裔的返回指针和 $f(x)$的值。修改原则：选择 $f(x)$值小的路径走。

2）为其余子节点配上指向 N 的返回指针后放入 OPEN 表中，并对 OPEN 表按照 $f(x)$值以升序排序，然后转步骤 2。

启发式图搜索的 A 搜索算法描述如下：

```
procedure heuristic search
open: =[start];closed: =[ ];f(s): =g(s)+h(s);     *初始化
while open≠[ ] do
begin
    从 OPEN 表中删除第一个状态，称之为 n;
    if n=目的状态 then return (success);
    生成 n 的所有子状态;
    if n 没有任何子状态 then continue;
    for n 的每个子状态 do
     case 子状态 is not already on OPEN 表 or CLOSED 表;
              begin
               计算该子状态的估价函数值;
               将该子状态加到 OPEN 表中;
              end;
         case 子状态 is already on OPEN 表;
              if 该子状态是沿着一条比在 OPEN 表已有的更短路径而到达
              then 记录更短路径走向及其估价函数值;
case 子状态 is already on CLOSED 表:
              if 该子状态是沿着一条比在 CLOSED 表已有的更短路径而到达
              then
begin
                    将该子状态从 CLOSED 表移到 OPEN 表中;
                    记录更短路径走向及其估价函数值;
     end;
case end;
```

```
                将 n 放入 CLOSED 表中;
                根据估价函数值，从小到大重新排列 OPEN 表;
        end;                                                    *open 表中节点已耗尽
        return(failure);
end
```

从上面的描述可知，在 A 搜索算法中，从 OPEN 表中取出第一个状态，如果该状态满足目的条件，则算法返回到该状态的搜索路径。在这里，每个状态都保留了其父状态的信息，以保证能够返回完整的搜索路径。

如果 OPEN 表的第一个状态不是目的状态，则算法利用与之相匹配的一系列操作算子进行相应的操作来产生它的子状态。如果某个子状态在 OPEN 表（或 CLOSED 表）中出现过，则该状态再一次被发现时，通过刷新它的祖先状态的历史记录，该算法极有可能找到到达目的状态的更短的路径。

接着，用 A 搜索算法计算 OPEN 表中每个状态的估价函数值，按照值的大小重新排序，将值最小的状态放在表头，使其第一个被扩展。

图 3.7 所示是一个启发式图搜索的层次式状态空间，有些状态还用括号标上了相应的估价函数值。标上值的那些状态都是在 A 搜索中实际生成的。在这个图中，A 搜索算法扩展的状态都已显示，可以看出 A 搜索算法无法搜索所有的状态空间。A 搜索算法的目标是尽可能地减小搜索空间而得到解，一般来说，启发信息给得越多，即估价函数值越大，需要搜索处理的状态数就越少。

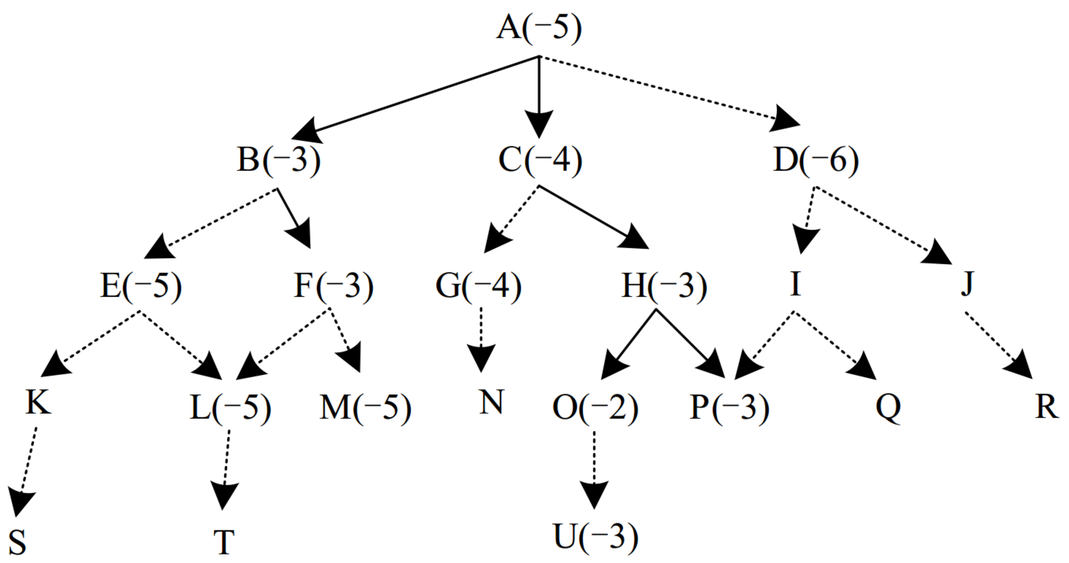


图 3.7　启发式图搜索示意

A 搜索算法总是从 OPEN 表中选取估价函数值最小的状态进行扩展。但是，在图 3.7 中假定状态 P 是目的状态，而到状态 P 的路径上的状态有较小的估价函数值，从而可以看出，启发信息难免会有错误，状态 O 比状态 P 的估价函数值小而先被搜索扩展。然而，A 搜索算法本身具有纠错功能，能够从不理想的状态跳转到正确的状态，即从状态 B 跳到状态 C 来进行搜索。

因此，A 搜索算法并不丢弃其他的生成状态，而是把它们保留到 OPEN 表中。当某一个启发信息将搜索导向错误路径时，算法可以从 OPEN 表中检索出先前生成的“次最好”状态，并且将搜索方向转向状态空间的另一部分。如图 3.7 所示，当算法发现状态 F 的子状态的估价函数值很差时，搜索便转移到状态 C，但是状态 F 的子状态 L 和 M 都保留在 OPEN 表中，以防算法在未来的某一步再一次转向它们。

3.4.4　A*搜索算法及其特征分析

定义 $h^*(n)$为状态 n 到目的状态的最优路径的代价，则当 A 搜索算法的启发函数 $h(n)$小于等于 $h^*(n)$，即满足

$$h(n)\leqslant h^*(n)，\text{对所有节点 } n \tag{3.4}$$

时，A 搜索算法称为 A*搜索算法。

A*搜索算法是由著名的人工智能学者尼尔斯·约翰·尼尔森（Nils John Nilsson）提出的，它是目前最有影响的启发式图搜索算法，也称为最佳图搜索算法。如果某一问题有解，那么利用 A*搜索算法对该问题进行搜索就一定能够搜索到解，并且一定能够搜索到最优解。

A*搜索算法比 A 搜索算法好。它不仅能够得到目标解，还一定能够找到最优解（只要问题有解）。

在一些问题求解中，只要搜索到一个解，就会想得到最优解，关键是提高搜索效率。那么，是否还有更好的启发式策略？在什么意义上称某一启发式策略比另一个好？另外，当通过启发式搜索得到某一状态的路径代价时，是否能够保证在以后的搜索中不会出现到达该状态有更小的路径代价？

针对上面这些问题，下面讨论 A*搜索算法的有关特性。

1．可采纳性

对于可解状态空间图，如果一个搜索算法在有限步内终止，并能够得到最优解，就称该搜索算法是可采纳的。

通过估价函数 $f(n)=g(n)+h(n)$，可归纳一类可采纳性的启发搜索策略的特征。若 n 是状态空间图中的一个状态，$g(n)$是衡量某一状态在图中的深度，$h(n)$是状态 n 到目的状态的代价估计值，那么 $f(n)$就是从起点出发，通过状态 n 到达目标状态的路径的总代价估计值。

定义最优估价函数为

$$f^*(n)=g^*(n)+h^*(n) \tag{3.5}$$

式中，$g^*(n)$为起点到状态 n 的最短路径代价值；$h^*(n)$是状态 n 到目的状态的最短路径代价值。这样，$f^*(n)$就是从起点出发通过状态 n 到达目的状态的最佳路径的总代价值。

尽管在绝大部分实际问题中并不存在 $f^*(n)$这样的先验函数，但是可以将 $f(n)$作为 $f^*(n)$的一个近似估计函数。在 A 及 A*搜索算法中，$g(n)$作为 $g^*(n)$的近似估价，可能两者并不相等，但是有 $g(n)\geqslant g^*(n)$。仅当搜索过程已发现了到达状态 n 的最佳路径时，它们才相等。

同样，可以用 $h(n)$代替 $h^*(n)$作为状态 n 到目的状态的最小代价估计值。虽然在绝大多数情况下无法计算 $h^*(n)$，但是要判别某一 $h(n)$是否大于 $h^*(n)$还是可能的。

宽度优先算法是 A*搜索算法的一个特例，是一个可采纳的搜索算法。该算法相当于 A*算法中取 $h(n)=0$ 和 $f(n)=g(n)$。宽度优先搜索时对某一状态只考虑它同起始状态的

距离代价。这是由于该算法在考虑 n+1 层状态之前，已考查了 n 层中的任意一种状态，因此每个目的状态都是沿着最短的可能路径找到的。但是宽度优先搜索算法的搜索效率太低。

2．单调性

在 A*搜索算法中并不要求 $g(n)=g^*(n)$，这意味着要采纳的启发式算法可能会沿着一条非最佳路径搜索到某一中间状态。如果对启发函数 $h(n)$加上单调性的限制，可以减小比较代价并减少调整路径的工作量，从而减小搜索代价。

下面介绍启发函数单调性的概念。

如果某一启发函数满足如下条件：

1）对所有状态 n_i 和 n_j（其中 n_j 是 n_i 的后裔），满足 $h(n_i)-h(n_j)\leqslant \text{cost}(n_i,n_j)$（其中 $\text{cost}(n_i,n_j)$是从 n_i 和 n_j 的实际代价）；

2）目的状态的启发函数值为 0 或 $h(\text{Goal})=0$，

则称该启发函数 $h(n)$是单调的。

搜索算法的单调性可这样描述：在整个搜索空间都是局部可采纳的。一个状态和任意一个子状态之间的差由该状态与其子状态之间的实际代价所限定。这就是说，启发策略无论在何处都是可采纳的，总是从祖先状态沿着最佳路径到达任一状态。

于是，由于算法总是在第一次发现该点时就已经发现了到达该点状态的最短路径，因此当某一状态被重新搜索时，无须检验新的路径是否更短，这就意味着当某一状态被重新搜索时，可以立即将其从 OPEN 表或 CLOSED 表中删除，无须修改路径的信息。

容易证明单调性启发策略是可采纳的。这意味着单调性策略中的 $h(n)$满足 A*搜索策略的下界要求，算法是可采纳的。

3．信息性

在两个 A*启发策略的 h_1 和 h_2 中，如果对搜索空间中的任一状态 n 都有 $h_1\leqslant h_2(n)$，就称策略 h_1 比 h_2 具有更多的信息性。某一搜索策略的 $h(n)$越大，它所搜索的状态就越少。

如果启发策略 h_2 的信息性比 h_1 多，那么用 h_2 所搜索的状态集合是 h_1 所搜索的状态集合的一个子集。因此，A*算法的信息性越多，它所搜索的状态数就越少。必须注意的是，更多的信息性需要更多的计算时间，从而有可能抵消减少搜索空间所带来的益处。

3.5 综 合 案 例

将宽度优先搜索应用于八数码难题时所生成的搜索树，这个问题就是将初始状态变为如下目标状态的问题，如图 3.8 所示。

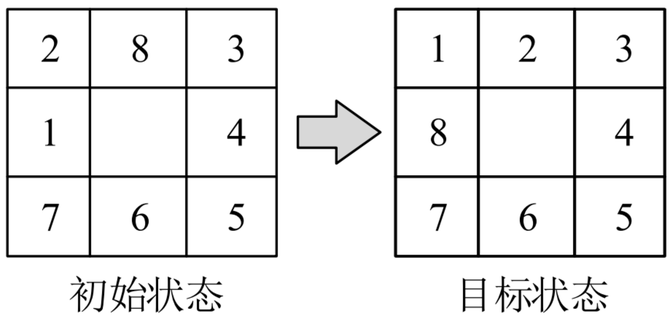


图 3.8　重排八数码难题

搜索树上的所有节点都标记了它们所对应的状态描述，每个节点旁边的数字表示节点扩展的顺序（按顺时针方向移动空格）。图 3.9 中的第 26 个节点是目标节点。

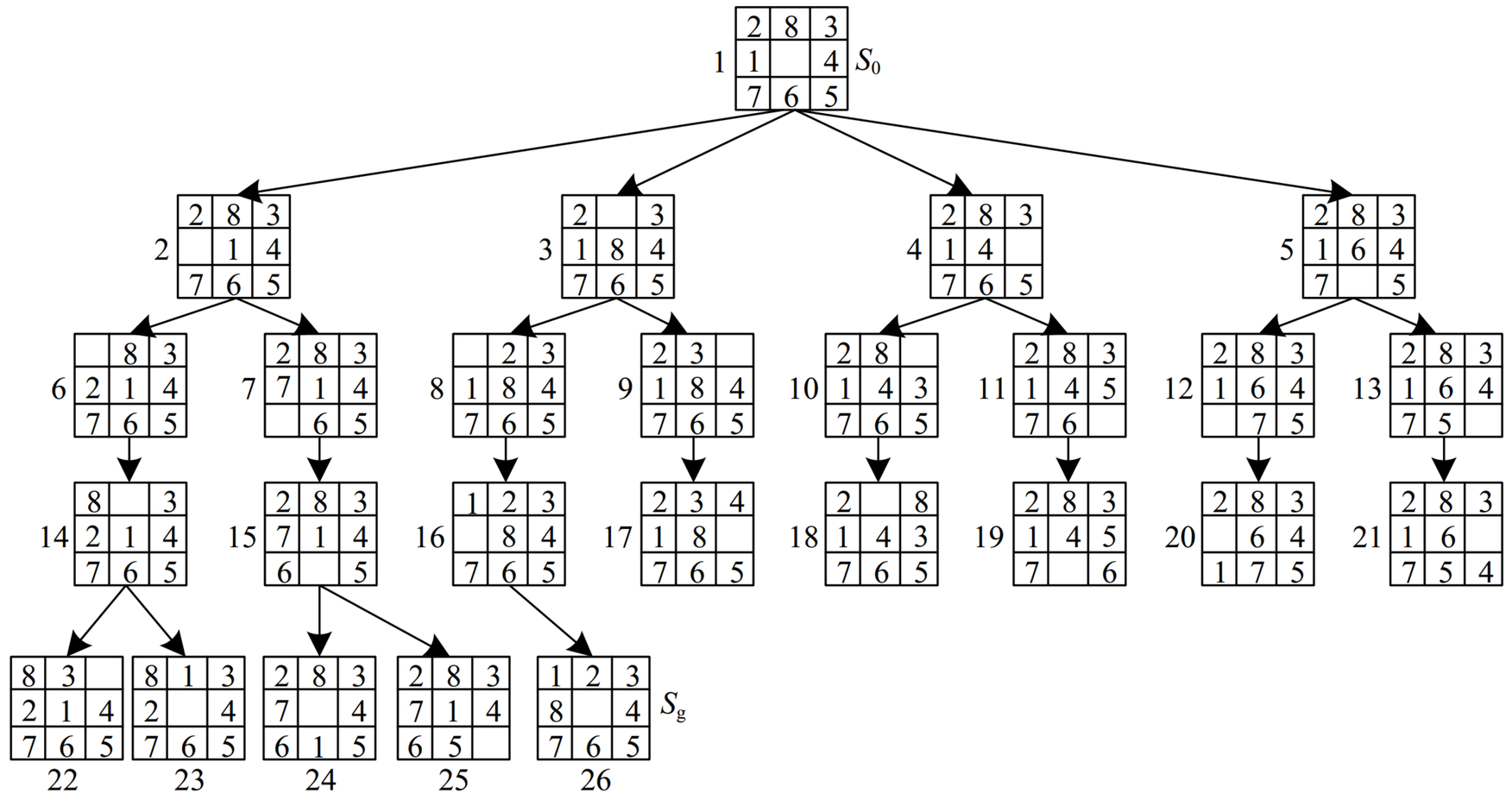


图 3.9　八数码难题的宽度优先搜索树

图 3.10 给出了按深度优先搜索生成的八数码难题搜索树。其中，设置深度界限为 5，粗线条的路径表明含有 5 条应用规则的一个解。从图中可见，深度优先搜索过程首先是沿着一条路径进行下去，直到达到深度界限为止；然后再考虑只有最后一步有差别的相同深度或较浅深度可供选择的路径；接着再考虑最后两步有差别的那些路径；等等。

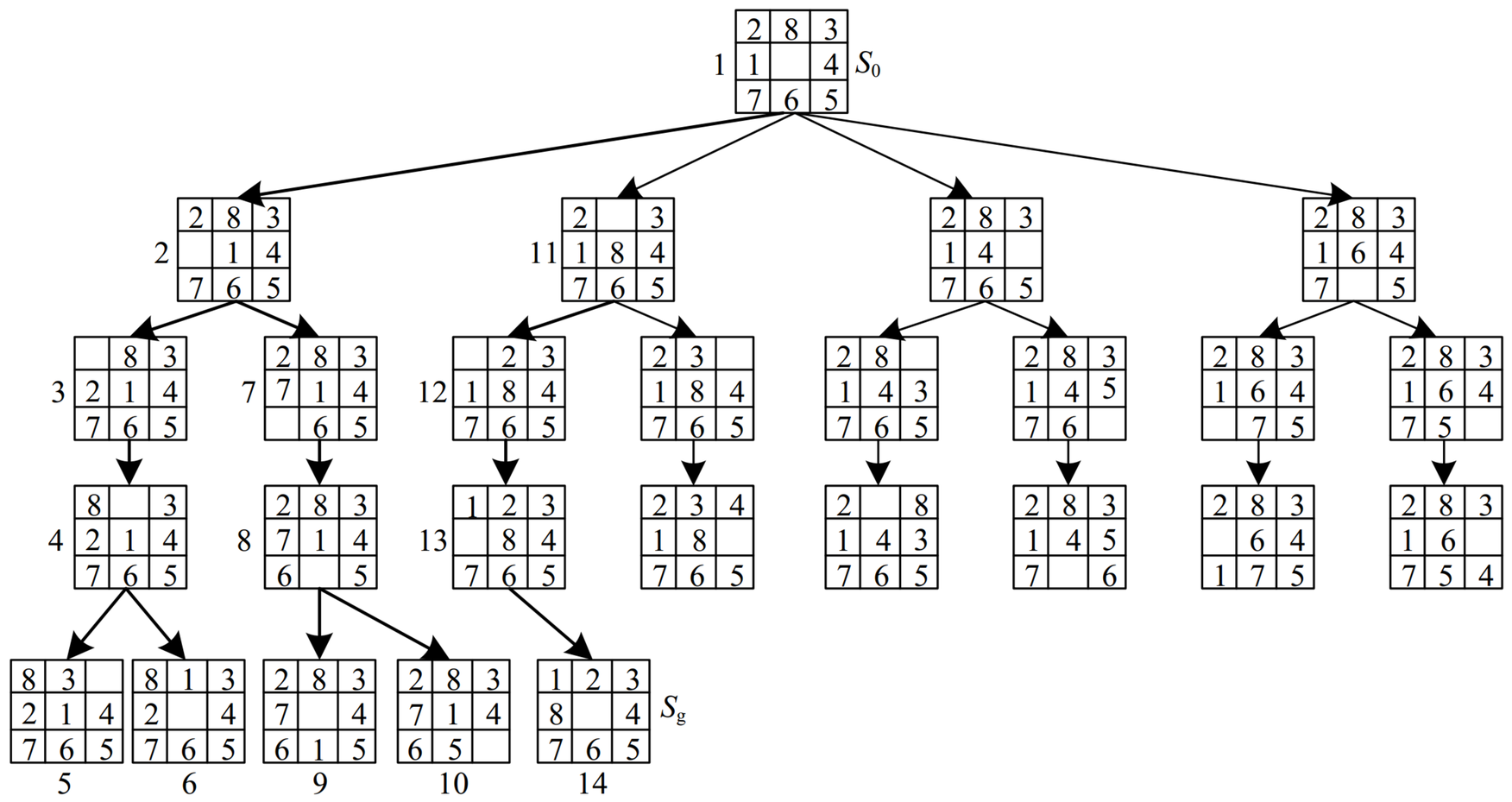


图 3.10　八数码难题的有界深度优先搜索树

下面再次用八数码难题的例子来说明有序搜索是如何应用估价函数排列节点的。这里采用如下简单的估价函数：

$$f(n)=d(n)+W(n) \tag{3.6}$$

式中，$d(n)$是搜索树中节点 n 的深度，这个深度实际就等同于从初始节点到节点 n 所需要进行的操作次数；$W(n)$用来计算节点 n 相对于目标棋局错放的棋子个数，一般来说，错放的棋子数量越少，则越接近于目标状态，因此这个值相当于描述了当前节点 n 与目标节点之间的距离。在这种估价函数定义下，起始节点棋局的 f 值等于 0+3=3。

图 3.11 表示利用这个估价函数把有序搜索应用于八数码难题的结果。图中圆圈内的数字表示该节点的 f 值，不带圈的数字表示节点扩展的顺序。从图中可见，这里所求得的解答路径和用其他搜索方法找到的解答路径相同。不过，估价函数的应用显著地减少了被扩展的节点数，如果只用估价函数 $f(n)=d(n)$，那么就得到宽度优先搜索过程。

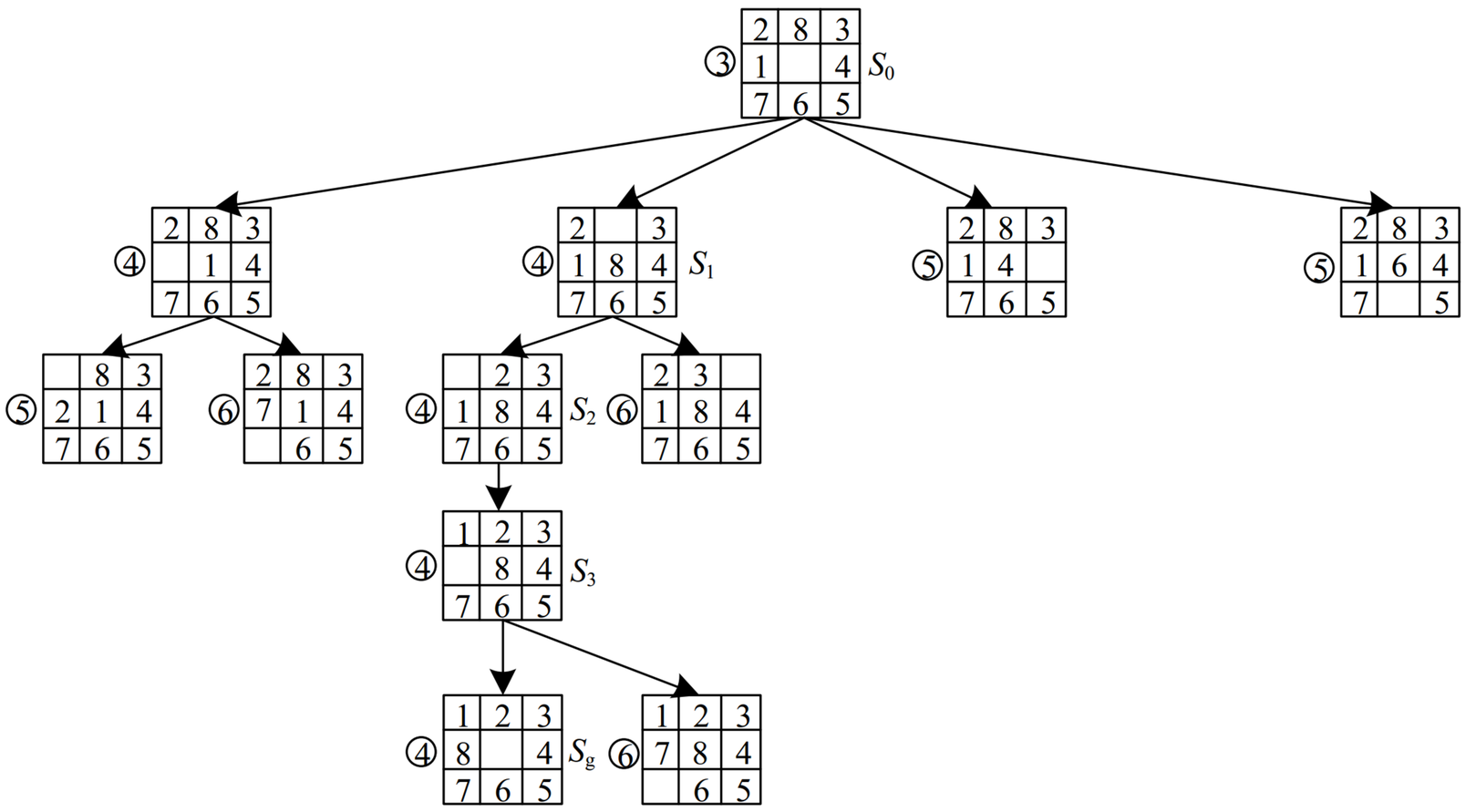


图 3.11　八数码难题的有序搜索树

前面已经提到，A*搜索算法中估价函数的定义是非常重要的，尤其是其中的启发函数 $h(n)$。由于启发信息在算法中是通过$h(n)$体现的，如果在估价函数的定义中恰好令 $h(n)=h^*(n)$，则可以看到搜索树将只扩展出最佳路径，也就是最理想的情况。但是一般情况下必须满足 $h(n)$不超过 $h^*(n)$算法才能保证找到最优解，$h(n)$的这种特性称为可采纳性，即 $h(n)$的定义必须满足可采纳性才能保证算法的最优性。对于同一个问题，如果有两种不同的启发函数定义均能满足可采纳性，且对于所有节点 x 来说，都有 $h_1(x)\leqslant h_2(x)$，则称 $h_2(x)$比 $h_1(x)$占优，采用 h_2 的算法将比采用 h_1 的算法更加高效。在上述问题中，若将 $h(n)$定义为所有棋子距离目标位置的曼哈顿距离之和（与目标位置的水平距离和垂直距离之和），则该定义会比放错的棋子数占优。在这种估价函数定义下，起始节点棋局的 $h(n)$值等于 1+1+2=4，显然该定义也能满足可采纳性要求。利用该函数来计算 f 值，搜索效率更高。读者可以试着画出搜索树，比较两种不同估价函数对算法的影响。

本 章 小 结

从问题表示到问题的解决，有个求解的过程，也就是搜索过程。本章给出了搜索的概念，包含搜索的基本问题和主要过程，对搜索策略进行了总体描述，分别对状态空间的搜索策略、盲目的图搜索策略和启发式图搜索策略进行了详细的讲解。综合案例使用宽度优先搜索和深度优先搜索解决八数码难题，使用估价函数把有序搜索应用于八数码难题的结果，启发读者自行比较在 A*搜索算法中使用两种不同估价函数对算法的影响。

本 章 习 题

1．用状态空间法表示问题时，什么是问题的解？求解过程的本质是什么？什么是最优解？最优解唯一吗？

2．宽度优先搜索与深度优先搜索有何不同？分析深度优先和宽度优先的优缺点。

3．什么是 A*搜索算法？它的估价函数是如何确定的？A*搜索算法与 A 搜索算法的区别是什么？

第 4 章　群集智能算法

4.1　群集智能概述

4.1.1　群集智能的基本概念

在生态系统中，行为模式简单的生物个体，在形成群体后却出现复杂而有序的种群自发行为。例如，鸟群在空中飞行时自动地调整队形；蚁群能够分享信息，协同合作，优化到达食物位置的最短路径；鱼群聚集最密集的地方通常是水中食物集中的地方；大肠杆菌会自动向营养浓度高的区域聚集。在仿生学的基础上，学者们将此类现象概述为：在不存在中央控制机制的条件下，种群中的所有个体都遵循某种特定的行为模式，通过个体间的相互影响与相互作用，在种群整体层面涌现出复杂系统行为。

群集智能自提出以来为解决复杂的组合优化问题提供了新的途径。从该角度可将群集定义为：可以通过合作进行分布式优化的个体集合，该集合由相互之间可以进行通信的个体组成。群集智能则表示行为模式简单的个体在群体层面上涌现出智能的行为。

4.1.2　群集智能研究方法的主要优缺点

与传统优化算法相比，群集智能鲁棒性强，并行性好，实现相对简单，无须中央控制机制。这类特性是普通的个体智能无法比拟的，尤其是对那些没有集中控制且不能提供全局信息的应用场景，群集智能展现出了较好的优化性能。

实现群集智能的五条基本原则如下。

1）接近原则：种群有能力进行简单的空间和时间的计算。

2）质量原则：个体除了能够响应空间和时间等因素外，还可以对信息的优劣做出反应。

3）多样性反应原则：种群需要具有一定广泛的活动范围，避免群体多样性由于环境变化而丧失。

4）稳定性原则：群体应保持一定的稳定性，有对抗环境发生变化的能力，种群的行为模式不能每次都随着环境的变化而改变。

5）适应性原则：群体的行为模式能够通过自适应改变以适应环境变化。

群集智能依赖一定的概率进行随机搜索，这点与传统基于梯度的算法有所不同。与传统方法相比，群集智能突出的优点如下。

1）分布式：种群中个体的搜索行为相对独立，并行性较强，适合网络和分布式系统计算。

2）鲁棒性：由于种群中不存在中央控制机制，个体相互平等，算法不会因为某些个体的失效而使整体相对问题的搜索失效。

3）可扩充性：群集中的个体可以使用多种信息共享方式，便于种群规模与智能个体的扩充。

4）简单性：个体的行为模式相对简单，利于算法实现。

群集智能也存在如下三个方面的缺陷。

1）并不是总能从涌现的群体行为中推导出个体的行为。

2）真实生物个体如此复杂，以至于几乎不可能在一个仿真系统中完成复制。

3）简单的规则产生类似生命的群体行为，并不能保证真实的生态系统一定能够遵循这些简单的规则。

4.2　蚁 群 算 法

4.2.1　蚁群算法的生物原型

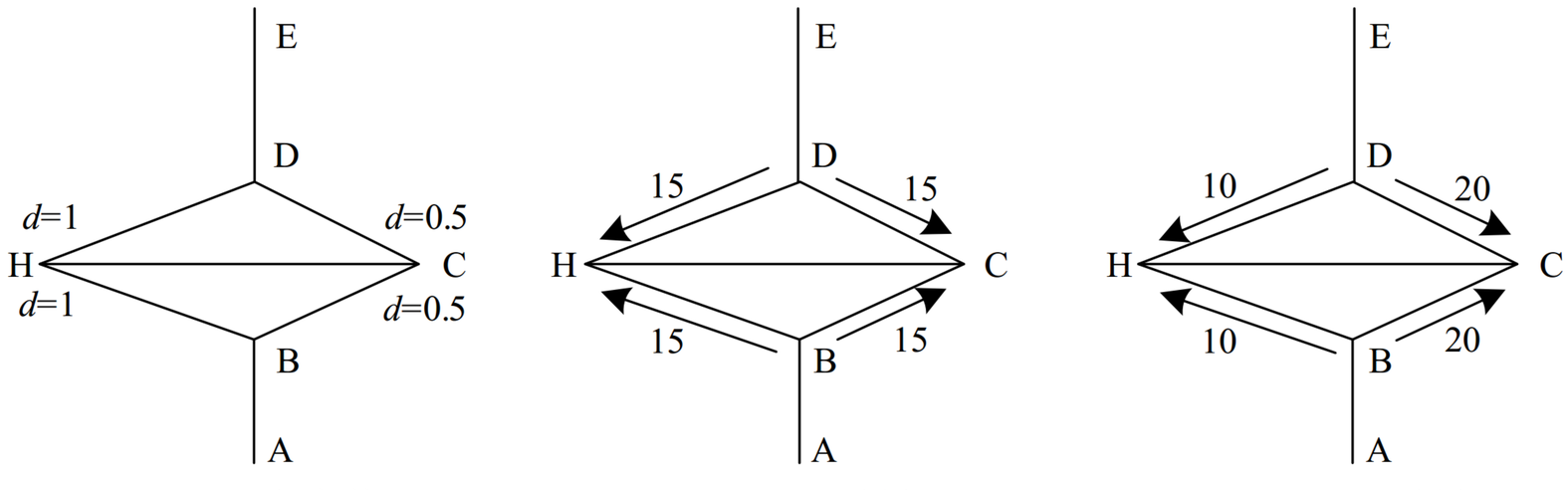
蚂蚁是一种众所周知的群居性小昆虫，单只蚂蚁很难完成复杂的任务，但是蚁群却拥有巨大的能量。人们知道它能够预报暴雨和洪涝气象，还能够毁坏河堤和水坝，引起水患。这个个体甚微的小生灵，作为群体表现出十分独特的生物特征和生命行为。1992 年，意大利学者马可 • 多里戈（Marco Dorigo）在他的博士论文中提出了蚁群算法（ant colony algorithm，ACA）。1999 年，马可 • 多里戈和詹妮 • 迪卡罗（Gianni Di Caro）给出了蚁群算法的一个通用框架，对蚁群算法的发展具有重要意义。同年，马尼佐（Maniezzo）和科洛龙（Colorni）从生物进化和仿生学角度出发，研究蚂蚁寻找路径的自然行为，并用该方法求解 TSP 问题、二次分配问题和作业调度问题，取得较好结果。蚁群算法已显示出它在求解复杂优化问题特别是离散优化问题方面的优势，是一种很有发展前景的计算智能方法。

4.2.2　基本蚁群算法的原理

蚁群算法（又称为人工蚁群算法）是因受到对真实蚁群行为研究的启发而提出的。为了说明人工蚁群系统的原理，先从蚁群搜索食物的过程谈起。像蚂蚁、蜜蜂、飞蛾等群居昆虫，虽然单个昆虫的行为极其简单，但是由每个简单的个体组成的群体却表现出极其复杂的行为。仿生学家经过大量细致观察、研究后发现，蚂蚁个体之间是通过一种称为外激素（pheromone）的物质进行信息传递的。蚂蚁在运动过程中，能够在所经过的路径上留下该种物质，而且蚂蚁在运动过程中能够感知这种物质，并以此指导自己的运动方向。因此，由大量蚂蚁组成的蚁群的集体行为表现出一种信息正反馈现象：某一路径上经过的蚂蚁越多，则后来者选择该路径的概率就越大。蚂蚁通过这种个体之间的信息交流来达到搜索食物的目的。

下面用马可 • 多里戈所举的例子来说明蚁群系统的原理。

如图 4.1 所示，设 A 是蚂蚁的巢穴，E 是食物源，HC 为一障碍物。由于存在障碍物，蚂蚁只能绕经障碍物的 H 端或 C 端由 A 到达 E，或由 E 到达 A。图 4.1（a）中 d 表示两点之间的距离。设每个时间单位有 30 只蚂蚁由 A 到达 B，又有 30 只蚂蚁由 E 到达 D，蚂蚁过后留下的信息素为 1。为便于讨论，设信息素停留的时间为 1。在 t=0 时刻，由于路径 BH、BC、DH、DC 上均无信息存在，位于 B 和 D 的蚂蚁可以随机选择路径。从统计学的角度可以认为各有 15 只蚂蚁以相同的概率选择 BH、BC、DH、DC，如图 4.1（b）所示。经过一个时间单位后，在路径 BCD 上的信息量是路径 BHD 上的信息量的两倍。在 t=1 时刻，将有 20 只蚂蚁由 B 和 D 到达 C，有 10 只蚂蚁由 B 和 D 到达 H，如图 4.1（c）所示。随着时间的推移，蚂蚁将会以越来越大的概率选择路径 BCD，最终完全选择路径 BCD，从而找到由蚁巢到食物源的最短路径。由此可见，蚂蚁个体之间的信息交换是一个正反馈过程。



（a）已知条件　（b）t=0时刻　（c）t=1时刻

图 4.1　蚁群系统示意图

4.2.3　蚁群优化算法的特点及收敛性

蚁群优化算法采用了分布式正反馈并行计算机制，易于与其他方法结合，并具有较强的鲁棒性。

1）其原理是一种正反馈机制或称增强型学习系统，它通过信息素的不断更新达到最终收敛于近似最优路径上。

2）它是一种通用型随机优化方法，人工蚂蚁不是对实际蚂蚁的简单模拟，它融入了人类的智能。

3）它是一种分布式的优化方法，不仅适合目前的串行计算机，而且适合未来的并行计算机。

4）它是一种全局优化方法，不仅可用于求解单目标优化问题，而且可用于求解多目标优化问题。

5）它是一种启发式算法，时间复杂度为 $O(\mathrm{NC}*m*n^2)$。其中，NC 是迭代次数，m 是蚂蚁数量，n 是目的节点数量。

蚁群优化算法具有如下优点。

1）可以快速求解问题。

2）具有全局优化性。

3）可以在有限的时间内获取合理的答案。

蚁群优化算法的收敛性包括值收敛和解收敛。值收敛评估是任意一个最优解至少出现一次的可能性，即至少有一只蚂蚁能够找到最优解的概率；解收敛评估是算法达到某个可以持续生产相同最优解的状态的可能性，即任意一只蚂蚁都可以找到最优解的概率。经科学家验证得出以下结果。

1）对于固定的信息素蒸发系数，只要迭代次数足够多，某只蚂蚁找到最优解的概率接近 1。

2）对于一定量的蚂蚁和足够小的信息素蒸发系数，只要迭代次数足够多，某只蚂蚁找到最优解的概率接近 1。

上述收敛指的是值收敛，此外，科学家也证明了蚁群优化算法的解收敛性。

4.2.4　基本蚁群算法的数学模型

蚁群优化算法的第一个应用是著名的旅行商问题（traveling salesman problem，TSP），马可・多里戈等充分利用了蚁群搜索食物的过程与旅行商问题之间的相似性，通过人工模拟蚂蚁搜索食物的过程，即通过个体之间的信息交流和相互协作，最终找到从蚁穴到食物源的最短路径，来求解旅行商问题。下面用旅行商问题阐明蚁群系统的模型。

设 m 是蚁群中蚂蚁的数量，$d_{xy}(x,y=1,\cdots,n)$表示元素（城市）x 和元素（城市）y 之间的距离。$\eta_{xy}(t)$表示能见度，称为启发信息函数，等于距离的倒数，即 $\eta_{xy}(t)=\dfrac{1}{d_{xy}}$。

$b_x(t)$表示 t 时刻位于城市 x 的蚂蚁的个数，$m=\sum_{x=1}^{n} b_x(t)$。$\tau_{xy}(t)$表示 t 时刻在 xy 连线上残留的信息素，初始时刻，各条路径上的信息素相等，即 $\tau_{xy}(0)=C$(常数)。蚂蚁 $k(k=1,\cdots,m)$ 在运动过程中，根据各条路径上的信息素决定转移方向。$P_{xy}^{k}(t)$ 表示在 t 时刻蚂蚁 k 选择从元素（城市）x 转移到元素（城市）y 的概率。

$P_{xy}^{k}(t)$ 由信息素 $\tau_{xy}(t)$和局部启发信息 $\eta_{xy}(t)$共同决定，已成为随机比例规则（random-proportional rule）：

$$P_{xy}^{k}(t)=\begin{cases}\dfrac{\left|\tau_{xy}^{k}(t)\right|^{\alpha}\left|\eta_{xy}^{k}(t)\right|^{\beta}}{\sum_{y\in \text{allowed}_k(x)}\left|r_{xy}^{k}(t)\right|^{\alpha}\left|\eta_{xy}^{k}(t)\right|^{\beta}}, & \text{如果 } y\in \text{allowed}_k(x)\\ 0, & \text{否则}\end{cases} \tag{4.1}$$

式中，$\text{allowed}_k(x)=\{0,1,\cdots,n-1\}-\text{tabu}_k(x)$，表示蚂蚁 k 在下一步运行选择的城市；$\text{tabu}_k(x)(k=1,2,\cdots,m)$ 记录蚂蚁 k 当前走过的城市；α 是信息素启发式因子，表示轨迹的相对重要性，反映了残留信息素浓度 $r_{xy}(t)$在指导蚁群搜索中的相对重要程度。α 值越大，该蚂蚁越倾向于选择其他蚂蚁经过的路径，该状态转移概率接近于贪婪规则。当 $\alpha=0$ 时，不再考虑信息素水平，算法成为有多重起点的随机贪婪算法。而当期望值启发式因子 $\beta=0$ 时，算法成为纯粹的正反馈的启发式算法。

随着时间的推移，以前留下的信息素逐渐消逝，用参数 $1-\rho$ 表示信息素挥发度（ρ

为信息素残留常数），蚂蚁完成一次循环，各路径上信息素浓度消散规则为

$$\tau_{xy}(t)=\rho\tau_{xy}(t)+\Delta\tau_{xy}(t) \tag{4.2}$$

蚁群的信息素浓度更新规则为

$$\Delta\tau_{xy}(t)=\sum_{k=1}^{m}\Delta\tau_{xy}^{k}(t) \tag{4.3}$$

马可·多里戈给出了$\Delta\tau_{xy}(t)$的三种不同模型。

第一种称为蚂蚁圈系统（ant-cycle system）。单只蚂蚁访问路径上的信息素浓度更新规则为

$$\Delta\tau_{xy}^{k}(t)=\begin{cases}\dfrac{C}{L_k}, & \text{若第}k\text{只蚂蚁在本次循环中的路径从}x\text{到}y\\ 0, & \text{否则}\end{cases} \tag{4.4}$$

式中，$\tau_{xy}(t)$为当前路径上的信息素，$\Delta\tau_{xy}(t)$为路径（x,y）上信息素的增量，$\Delta\tau_{xy}^{k}(t)$为第k只蚂蚁留在路径（x, y）上信息素的增量；C为常数；L_k为优化问题的目标函数值，表示第k只蚂蚁在本次循环中所走路径的长度。根据具体算法的不同，$\Delta\tau_{xy}^{k}(t)$、$\Delta\tau_{xy}(t)$、$\tau_{xy}(t)$及$\tau_{xy}^{k}(t)$的表达形式可以不同，要根据具体问题而定。

第二种称为蚂蚁数量系统（ant-quantity system），更新规则为

$$\Delta\tau_{xy}^{k}(t)=\begin{cases}\dfrac{C}{d_{xy}}, & \text{若第}k\text{只蚂蚁在本次循环中的路径从}x\text{到}y\\ 0, & \text{否则}\end{cases} \tag{4.5}$$

第三种称为蚂蚁密度系统（ant-density system），更新规则为

$$\Delta\tau_{xy}^{k}(t)=\begin{cases}C, & \text{若第}k\text{只蚂蚁在本次循环中的路径从}x\text{到}y\\ 0, & \text{否则}\end{cases} \tag{4.6}$$

第一种模型利用的是整体信息，即蚂蚁完成一个循环后，更新所有路径上的信息，通常作为蚁群优化算法的基本模型。后两种模型利用的是局部信息，每走一步都要更新残留信息素的浓度，而非等到所有蚂蚁完成对所有n个城市的访问以后。

比较上述三种方法，蚂蚁圈系统的效果最好，这是因为它利用的是全局信息C/L_k，而其他两种算法用的是局部信息C/d_{xy}和C。全局信息更新方法很好地保证了残留信息素不至于无限累积，如果路径没有被选中，那么上面的残留信息素会随时间的推移而逐渐减弱，这能使算法“忘记”不好的路径。即使路径经常被访问，也不至于因为$\Delta\tau_{xy}^{k}(t)$的累积而产生$\Delta\tau_{xy}^{k}(t)\gg\eta_{xy}(t)$，使期望值的作用无法体现，这充分体现了算法中全局范围内较短路径（较好解）的生存能力，加强了信息正反馈性能，提高了系统搜索收敛的速度。因而，在蚁群算法中，通常采用蚂蚁圈系统作为基本模型。

4.2.5 蚁群算法的参数设置

从蚁群搜索最短路径的机理不难看出，算法中有关参数的不同选择对蚁群算法的性能有重要的影响，但是其选取的方法和原则目前尚没有理论上的依据，通常都是根据经

验而定。

信息素启发式因子 α 的大小反映了蚁群在路径搜索中随机性因素作用的强度，其值越大，蚂蚁选择以前走过的路径的可能性就越大，搜索的随机性减弱，当 α 过大时会使蚁群的搜索过早陷入局部最优。期望值启发式因子 β 的大小反映了蚁群在路径搜索中先验性、确定性因素作用的强度，其值越大，蚂蚁在某个局部点上选择局部最短路径的可能性越大。虽然搜索的收敛速度得以加快，但蚁群在最优路径的搜索过程中随机性减弱，易于陷入局部最优。蚁群算法的全局寻优性能首先要求蚁群的搜索过程必须具有很强的随机性，而蚁群算法的快速收敛性能又要求蚁群的搜索过程必须具有较高的确定性，因此 α 和 β 对蚁群算法性能的影响和作用是相互配合、密切相关的。

蚁群算法存在收敛速度慢、易于陷入局部最优等缺陷。而信息素挥发度 $1-\rho$ 直接关系到蚁群算法的全局搜索能力及其收敛速度。由于信息素挥发度 $1-\rho$ 的存在，当要处理的问题规模比较大时，会使那些从来未被搜索到的路径（可行解）上的信息量减小到接近于 0，因而降低了算法的全局搜索能力，但是当 $1-\rho$ 过大时，以前搜索过的路径被再次选择的可能性也会过大，这会影响算法的随机性能和全局搜索能力。反之，通过减小信息素挥发度 $1-\rho$，虽然可以提高算法的随机搜索性能和全局搜索能力，但是又会使算法的收敛速度降低。

对于 TSP 问题，单个蚂蚁在一次循环中所经过的路径，表现为问题的可行解集合中的一个解，k 只蚂蚁在一次循环中所经过的路径，则表现为问题的可行解集合中的一个子集。显然，子集越大（即蚂蚁数量越多），越可以提高蚁群算法的全局搜索能力以及算法的稳定性。但是蚂蚁数量增加后，会使大量的曾被搜索过的解（路径）上的信息素量的变化比较平均，信息素正反馈的作用不明显，搜索的随机性虽然得到了加强，但是收敛速度减慢。反之，子集较小（即蚂蚁数量少），特别是当要处理的问题规模比较大时，会使那些从来未被搜索到的解（路径）上的信息素量减小到接近于 0，搜索的随机性减弱，虽然收敛速度加快，但是会使算法的全局性能降低，算法的稳定性较差，容易出现过早停滞现象。

在蚂蚁圈系统模型中，总信息素量 C 为蚂蚁循环一周时释放的在所经过路径上的信息素总量。总信息素量 C 越大，在蚂蚁已经走过的路径上信息素的累积越快，可以加强蚁群搜索时的正反馈性能，有助于算法的快速收敛。由于在蚁群算法中各个算法参数的作用实际上是紧密结合的，其中对算法性能起着主要作用的应该是信息素启发式因子 α、期望值启发式因子 β 和信息素残留常数 ρ 三个参数。总信息素量 C 对算法性能的影响有赖于上述三个参数的配置以及算法模型的选取。例如，在蚂蚁圈系统模型和蚂蚁数量系统模型中，总信息素量 C 对算法性能的影响情况有较大的差异。同样，信息素初始值 γ_0 对算法性能的影响不是很大。

4.2.6　改进的蚁群算法

1．蚂蚁-Q 系统

1995 年，意大利学者卢卡（Luca）、甘巴德拉（Gambardella）、马可·多里戈提出了蚂蚁-Q 系统（ant-Q system）。该算法在蚁群算法的随机比例规则基础上，在解构造

过程中提出了伪随机比例状态迁移规则，从而能够实现解构造过程中知识探索和知识利用的平衡，并引入信息素局部更新过程。信息素局部更新规则引入了强化学习理论中的 Q 学习机制。此外，在信息素的全局更新中采用了精英策略：

$$y=\begin{cases}\arg\max\limits_{y\in \text{allowed}_k(x)}\left\{[\text{HE}(x,u)]^{\alpha}[\text{AQ}(x,u)]^{\beta}\right\}, & \text{如果}q\leqslant q_0\\ Y, & \text{其他}\end{cases} \tag{4.7}$$

式中，α 和 β 是衡量学习 AQ 值得到 HE 值的重要参数。

根据式（4.7）计算概率分布如下：

$$P_k(x,y)=\begin{cases}\dfrac{\text{HE}(x,y)^{\alpha}\cdot\text{AQ}(x,y)^{\beta}}{\sum\limits_{y\in \text{allowed}_k(x)}\left\{\text{HE}(x,u)^{\alpha}\cdot\text{AQ}(x,u)\beta\right\}}, & \text{如果}y\in \text{allowed}_k(x)\\ 0, & \text{其他}\end{cases} \tag{4.8}$$

AQ 值按照如下规则进行更新：

$$\text{AQ}(x,y)\leftarrow(1-\alpha)\text{AQ}(x,y)+\alpha\left(\Delta\text{AQ}(x,y)+\gamma\cdot\max_{y\in \text{allowed}_k(x)}\text{AQ}(x,u)\right) \tag{4.9}$$

$$\Delta\text{AQ}(x,y)=\begin{cases}\dfrac{w}{L_k}, & \text{若第 }k\text{只蚂蚁在本次循环中的路径从}x\text{到}y\\ 0, & \text{其他}\end{cases} \tag{4.10}$$

假设 AQ(x, u)是与边(x, u)相关联的正实值，旨在表明当在城市 x 时，移动到城市 u 的作用有多大。当蚂蚁-Q 系统应用于对称旅行商问题时，AQ(x, u)=AQ(u, x)，当蚂蚁-Q 系统应用于非对称旅行商问题时，AQ(x, u)可能不等于 AQ(u, x)。

假设 HE(x, u)是与边(x, u)相关联的启发式值，该值允许对那些移动更好的进行启发式评估。

假设 k 为一个个体，它的任务是进行一次旅行，参观所有的城市，然后回到起点。与 k 相关联的是有待访问的城市列表 $\text{allowde}_k(x)$，其中，x 是当前城市。

位于城市 x 中的个体 k 使用以下规则移动到城市 u，称为动作选择规则（或状态转换规则）：

2．蚁群系统

1996 年，甘巴德拉和马可·多里戈又在蚂蚁-Q 的基础上提出了一种修正的蚁群算法，称为蚁群系统（ant colony system，ACS）。该算法可以看成蚂蚁-Q 算法的特例。它与蚁蚂圈系统算法的不同之处在于蚂蚁选择城市时遵循的规则，这里使用的是所谓的状态转移规则（state transition rule）：

$$S_k=\begin{cases}\arg\max\left\{\left[\tau\left(\begin{matrix}s\\ r,s\in j_k(r)\end{matrix}\right)\right]^{\alpha}[\eta(r,s)]^{\beta}\right\}, & (q\leqslant q_0)\\ S, & (q>q_0)\end{cases} \tag{4.11}$$

式中，S_k 是序号为 k 的蚂蚁所选中的下一个节点；q 表示一个随机变量；q_0 是一个适当

选定的阈值；$\tau\begin{pmatrix} s \\ r, s \in j_k(r) \end{pmatrix}$为边$(r,s)$上的信息素，$\eta(r,s)$为对应的启发式值。蚂蚁在选择下一座城市之前先进行一次随机试验得 q，若 $q \leqslant q_0$，则选择城市时按第一种情况，即选择当前信息素浓度最大的路径。式（4.11）中的第一种情况表示若自变量不在蚂蚁 k 的禁忌表 $j_k(r)$中，且令花括号中表达式最大，则整个表达式的值为该自变量的值，这称为知识利用，而非随机的方法（其余参数含义同前）。若 $q>q_0$，则按第二种情况随机选择变量 S，即采用基本蚁群算法的选择法，称为知识搜索。S 的概率分布 $P_k(r,s)$与前述蚂蚁圈系统算法中的计算方法相同，参见式（4.8）。从中可以看出，上述算法和以前算法的主要不同在于蚂蚁选择下一座城市之前多进行了一次随机试验，将选择情况分成“利用已知信息”和“探索”两类。

马可·多里戈等讨论不同的蚁群种群初始分布对求解的影响，提出了所谓的精英策略（elitist strategy），以强化精英蚂蚁（发现迄今最好路径的蚂蚁）的影响。结果发现，对精英蚂蚁数而言有一个最优的范围：低于此范围，增加精英蚂蚁数可较早地发现更好的路径；高于此范围，精英蚂蚁会在搜索早期迫使寻优过程始终在次优解附近，导致性能变差。

3．最大-最小蚂蚁系统

最大-最小蚂蚁系统（max-min ant system，MMAS）是德国学者托马斯·施蒂茨勒（Thomas Stutzle）等于 1997 年提出的。该算法在启动时将所有支路上的信息素浓度初始化为最大值 $\tau_{\max}$；为了更好地利用历史信息，每次迭代后按照挥发系数 ρ（$\rho+\rho=1$）降低信息素浓度，只有最佳路径上的支路才允许增加信息素浓度并保持在高水平上，也就是用当前找到的最好解更新信息素来指引蚂蚁向更高质量的解空间搜索的贪婪策略。信息素可按照式（4.12）更新：

$$\tau(x,y)=\rho\cdot\tau(x,y)+\Delta\tau(x,y)^{\text{best}} \tag{4.12}$$

式中，$\Delta\tau(x,y)^{\text{best}}=\dfrac{1}{f(s^{\text{best}})}$，$f(s^{\text{best}})$表示该次迭代中的最优路径或是全局最优路径的代价函数。

为了避免算法过早收敛于局部最优解，将各条路径可能的信息素浓度限制于$[\tau_{\min},\tau_{\max}]$，超出这个范围的值将被强制设置为 $\tau_{\min}$ 或 $\tau_{\max}$，可以有效地避免某条路径上的信息素量远大于其他路径，使得所有蚂蚁都集中在同一条路径上，从而使算法不再扩散。但是，只采用最大、最小信息素浓度的限制，还不足以在较长的运行时间里消除停滞现象，因此，采用了让路径上信息素浓度的增加正比于 $\tau_{\max}$ 和当前浓度 $\tau(x,y)$之差的平滑机制，如式（4.13）所示：

$$\tau'(x,y)=\tau(x,y)+\delta\left(\tau_{\max}(x,y)-\tau(x,y)\right)\quad 0<\delta<1 \tag{4.13}$$

4.3 粒子群优化算法

4.3.1 粒子群优化算法的生物原型

粒子群优化（particle swarm optimization，PSO）算法最初是由肯尼迪（Kennedy）和埃伯哈特（Eberhart）于 1995 年受人工生命研究结果启发，在模拟鸟群觅食过程中的迁徙和群集行为时提出的一种基于群体智能的进化计算技术。鸟群中的每只鸟在初始状态下是处于随机位置向各个随机方向飞行的，但是随着时间的推移，这些初始处于随机状态的鸟通过自组织（self-organization）逐步聚集成一个个小的群落，并且以相同速度朝着相同方向飞行，然后几个小的群落又聚集成大的群落，大的群落可能又分散为一个个小的群落。这些行为和现实中的鸟类飞行的特性是一致的。可以看出，鸟群的同步飞行这个整体的行为只是建立在每只鸟对周围的局部感知上面，而且并不存在一个集中的控制者。也就是说，整个群体组织起来但是没有一个组织者，群体之间相互协调却没有一个协调者。肯尼迪和埃伯哈特从诸如鸟类这样的群居性动物的觅食行为中得到启示，发现鸟类在觅食等搜寻活动中，通过群体成员之间分享关于食物位置的信息，可以大大加快找到食物的速度，即通过合作可以加快发现目标的速度。通常，群体搜寻所获得的利益要大于群体成员之间争夺资源产生的损失。这些简单的经验事实加以提炼，可以用如下规则来说明：当整个群体搜寻某个目标时，对于其中的某个个体，它往往是参照群体中目前处于最优位置的个体和自身曾经到达的最优位置来调整下一步的搜寻。肯尼迪和埃伯哈特把这个模拟群体相互作用的模型经过修改并设计成了一种解决优化问题的通用方法，称为粒子群优化算法。

4.3.2 标准粒子群优化算法

PSO 算法根据对环境的适应度将群体中的个体移动到好的区域，每个个体都被看作 n 维搜索空间中一个没有体积质量的粒子，在搜索空间中以一定的速度飞行。

在 PSO 算法中，在 n 维连续搜索空间中，对粒子群中第 $i(i=1,2,\cdots,m)$ 个粒子，定义 n 维当前位置向量 $\boldsymbol{x}^i(k)=[x_1^i,x_2^i,\cdots,x_n^i]^{\mathrm{T}}$ 表示搜索空间中粒子的当前位置，n 维最优位置向量 $\boldsymbol{P}^i(k)=[P_1^i,P_2^i,\cdots,P_n^i]^{\mathrm{T}}$ 表示该粒子至今所获得的具有最优适应度 $f_P^i(k)$ 的位置，即每个粒子经历过的最优位置（pbest），n 维速度向量 $\boldsymbol{v}^i(k)=[v_1^i,v_2^i,\cdots,v_n^i]^{\mathrm{T}}$ 表示该粒子的搜索方向。

群体经历过的最优位置（gbest）记为 $\boldsymbol{P}^g(k)=[P_1^g,P_2^g,\cdots,P_n^g]^{\mathrm{T}}$，则基本的 PSO 算法为

$$v_j^i(k+1)=\omega(k)v_j^i(k)+\varphi_1\mathrm{rand}(0,a_1)\left(P_j^i(k)-x_j^i(k)\right)+\varphi_2\mathrm{rand}(0,a_2)\left(P_j^g(k)-x_j^i(k)\right) \quad (4.14a)$$

$$x_j^i(k+1)=x_j^i(k)+v_j^i(k+1) \quad (i=1,2,\cdots,m;\ j=1,2,\ldots,n) \quad (4.14b)$$

式中，ω 是惯性权重因子；φ_1、φ_2 是加速度常数，分别控制个体认知分量和群体社会分量相对贡献的学习率，均为非负值；$\mathrm{rand}(0,a_1)$ 和 $\mathrm{rand}(0,a_2)$ 为 $[0,a_1]$、$[0,a_2]$ 范围内的具有均匀分布的随机数，a_1 与 a_2 为相应的控制参数。

式（4.14a）等号右边的第一部分是粒子在前一时刻的速度；第二部分为个体“认知”（cognition）分量，表示粒子本身的思考，将现有的位置和曾经经历过的最优位置相比；第三部分是群体“社会”（social）分量，表示粒子之间的信息共享与相互合作。φ_1 和 φ_2 引入 rand(0,a_1)和 rand(0,a_2)，将增加认知和社会搜索方向的随机性和算法多样性。

基于学习率 φ_1、φ_2，肯尼迪给出以下四种类型的 PSO 模型。

1）若 $\varphi_1>0$，$\varphi_2>0$，则称该算法为 PSO 全模型。

2）若 $\varphi_1>0$，$\varphi_2=0$，则称该算法为 PSO 认知模型。

3）若 $\varphi_1=0$，$\varphi_2>0$，则称该算法为 PSO 社会模型。

4）若 $\varphi_1=0$，$\varphi_2>0$，且 $g\neq i$，则称该算法为 PSO 无私模型。

标准的 PSO 算法分为两个版本：全局版和局部版。上面介绍的是全局版 PSO 算法。局部版与全局版的差别在于，用局部领域内最优邻居的状态代替整体群体的最优状态。全局版的收敛速度比较快，但是容易陷入局部极值点，而局部版搜索到的解可能更优，但是速度较慢。

粒子群优化算法的流程如下。

1）初始化每个粒子，即在允许范围内随机设置每个粒子的初始位置和速度。

2）评价每个粒子的适应度，计算每个粒子的目标函数。

3）设置每个粒子的 P_i。对每个粒子，将其适应度与其经历过的最好位置的 P_i 进行比较，如果优于 P_i，则将其作为该粒子的最好位置 P_i。

4）设置全局最优值 P_g。对每个粒子，将其适应度与群体经历过的最好位置的 P_g 进行比较，如果优于 P_g，则将其作为当前群体的最好位置 P_g。

5）根据式（4.14a）和式（4.14b）更新粒子的速度和位置。

6）检查终止条件。如果未达到设定条件（预设误差或迭代次数），则返回步骤 2）。

4.3.3　量子粒子群优化算法

在经典力学中，粒子通过位置向量 $\boldsymbol{x}_i$ 和速度向量 $\boldsymbol{v}_i$ 来描述粒子的运动轨迹，粒子在牛顿力学确定的轨迹下移动。在量子力学中是没有确定的轨迹的，因为根据不确定性原理，位置向量 $\boldsymbol{x}_i$ 和速度向量 $\boldsymbol{v}_i$ 是不可能同时确定的。因此，在粒子群优化算法中，如果个体加入量子行为，将会丰富算法中种群的多样性，提高算法的全局搜索能力。

在量子粒子群优化（quantum-behaved particle swarm optimization，QPSO）算法中，粒子不再被描述为位置向量 $\boldsymbol{x}_i$ 和速度向量 $\boldsymbol{v}_i$，而是采用波函数（wave-function）来表示。

粒子的波函数为

$$\psi(x)=\frac{1}{\sqrt{L}}\exp\left(-\frac{\|P-x\|}{L}\right) \tag{4.15}$$

式中，$\|P-x\|$表示 $P-x$ 的范数。

其概率密度函数为

$$Q(x)=\psi(x)^2=\frac{1}{L}\exp\left(-2\times\frac{\|P-x\|}{L}\right) \tag{4.16}$$

式中，P 为每个粒子历史的最好位置。参数 L 指出了微粒的搜索空间范围，L 的取值可

通过式（4.17）和式（4.18）确定：

$$L(t+1)=2\alpha\,|P-x(t)| \tag{4.17}$$

$$L(t+1)=2\alpha\left|\text{mbest}-x(t)\right| \tag{4.18}$$

式中，α 为收敛系数，不同的 α 影响算法的收敛速度，一般 α 取值为

$$\alpha=\frac{(1.0\sim 0.5)\times(\text{MAXITER}-T)}{\text{MAXITER}} \tag{4.19}$$

式中，MAXITER 为最大迭代次数；T 为当前迭代次数。

同时针对新算法引入平均最佳位置（mean best position，mbest）为所有微粒的中心：

$$\text{mbest}=\sum_{i=1}^{M}\frac{P_i}{M} \tag{4.20}$$

式中，M 为种群数量；P_i 是第 i 个粒子的最好位置。

通常将所有粒子的中心 mbest 取代每个粒子的最好位置 P，可以有效提高算法的全局搜索能力。

在给定概率分布函数的情况下，利用蒙特卡罗（Monte Carlo）逆变换可以测量粒子的位置，表示如下；

$$x(t)=P\pm\frac{L}{2\ln\frac{1}{u}} \tag{4.21}$$

式中，u 是均匀分布在（0,1）上的随机数，代入参数 L，量子粒子群优化算法的进化方程为

$$x(t+1)=P\pm\alpha\left|\text{mbest}-x(t)\right|\ln u \tag{4.22}$$

量子粒子群优化算法的基本步骤如下。

1）确定种群规模和粒子维数，初始化粒子群体。

2）计算个体历史最优值。根据适应度函数计算每一个微粒的适应度值，通过与个体的历史最优值比较，如果当前值优于个体历史最优值，则把当前值替换为个体最优值，否则不替换。

3）计算群体的历史最优值。计算所有微粒的适应值，并与当前的全局最优值（gbest）比较，若当前值优于全局最优值，则把当前值替换为全局最优值。

4）计算所有粒子的重心。根据式（4.20）更新所有粒子的重心。

5）根据量子粒子群进化方程式（4.22）更新每个粒子的位置，产生新的种群。

6）如果粒子适应度满足收敛条件或者达到最大迭代次数，则算法结束，否则跳转到步骤 2）继续迭代执行。

4.4 遗 传 算 法

4.4.1 遗传算法原型

生命自地球上诞生以来，就开始了漫长的生物进化历程，低级、简单的生物类型逐

渐发展为高级、复杂的生物类型。这一过程已经由古生物学、胚胎学和解剖学等方面的研究所证实。生物进化的原因从古至今有着各种不同的解释，其中被人们广泛接受的是达尔文的自然选择学说。

自然选择学说认为，生物要生存下去，就必须进行生存斗争。生物斗争包括种内斗争、种间斗争以及生物与无机环境之间的斗争。在斗争中，具有有利变异（mutation）的个体更容易存活，并且有更大的概率将有利变异遗传给后代；具有不利变异的个体容易被淘汰，将变异遗传给后代的概率也更小。因此，凡是在生存斗争中获胜的个体都是对环境适应性比较强的个体。达尔文把这种生存斗争中适者生存、不适者淘汰的过程叫作自然选择。达尔文的自然选择学说表明，遗传和变异是决定生物进化的内在因素。遗传是指父代与子代之间在性状上的相似现象。变异是指父代与子代之间以及子代的个体之间在性状上的差异现象。

根据现在细胞学和遗传学的研究得知，遗传物质的主要载体是染色体，染色体主要是由脱氧核糖核酸（DNA）和蛋白质组成，其中 DNA 又是最主要的遗传物质。现代分子水平的遗传学研究进一步证明，基因是有遗传效应的片段，它储存遗传信息，可以准确地复制，也能够发生突变，并可通过控制蛋白质的合成来控制生物的性状。生物体自身通过对基因的复制和交叉（即基因分离、基因自由组合和基因连锁互换）操作使其性状的遗传得到选择和控制。同时，通过基因重组、基因变异和染色体在结构和数目上的变异产生丰富多彩的变异现象。根据达尔文进化论，多种多样的生物之所以能够适应环境而得以生存进化，是与上述遗传和变异现象分不开的。

遗传算法（genetic algorithm，GA）是模拟达尔文的遗传选择和自然淘汰生物进化过程的计算模型，它是由美国密歇根大学的霍兰（Holland）于 1975 年首先提出的。霍兰和他的研究小组围绕遗传算法进行研究的宗旨有两个：一是抽取和解释自然系统的自适应过程；二是设计具有自然系统机理的人工系统。毫无疑问，霍兰的研究无论对自然系统还是对人工系统都是十分有意义的。

为了辅助理解，本节提出一个简单的背包问题，并以该问题作为示例贯穿本节全部内容。具体问题如下：现有一个背包和 4 个物品 $T=\{t_1,t_2,t_3,t_4\}$，4 个物品的物品重量 $W=\{w_1,w_2,w_3,w_4\}$，4 个物品的价值 $V=\{v_1,v_2,v_3,v_4\}$，背包可装载的重量上限为 10，每个物品的价值和重量如表 4.1 所示。如何才能使背包中的物品价值最高？

表 4.1　物品重量、价值表

物品（T）	重量（W）	价值（V）
t_1	2	5
t_2	5	4
t_3	3	2
t_4	4	3

4.4.2　编码

如何将问题的解编码成为染色体是遗传算法使用中的关键问题。在过去的 10 多年

里，已经针对特定的问题提出了各种编码方法，其目的是能够更有效地实现遗传算法。根据采用何种符号作为基因的等位基因，编码方式可分类如下。

1）二进制编码。

2）实数编码。

3）整数或字母排列编码。

4）一般数据结构编码。

其中，实数编码对于函数优化问题最有效。由于实数编码基因型空间中的拓扑结构与其表现空间中的拓扑结构一致，因此很容易从传统优化方法中借鉴好的技巧来形成有效的遗传算法。整数和字母排列编码对于组合优化问题最有效。由于组合优化问题的关键是寻找满足约束项目的最佳排列或组合，因此字母排列编码对于这类问题是最有效的方法。对于更复杂的现实问题，用合适的数据结构来表示基因的等位基因，可以有效抓住问题的本质。在这种情况下，基因可能是 n 维数组或更为复杂的数据结构。

根据编码的结构，编码方式可分为两类：一维编码和多维编码。在大多数实践中采用的编码方式均为一维编码。然而许多实际问题需要多维结构的解，用多维编码方式来表示这些解就很自然。

每当提出一种编码方式，都需要验证能否采用编码建立有效的遗传搜索。评价编码方式的原则有非冗余性（nonredundancy）、合法性（legality）、完备性（completeness）、拉马克性质（Lamarckian property）和因果性（causality）五种。

原则 4.1　非冗余性：从编码到解的映射应该是 1 对 1 的。

1 对 1 映射可确保在产生后代时不会进行无价值的操作，如果是 n 对 1 的映射，遗传算法在搜索过程中会浪费时间，1 对 n 的映射则还需要其他方法在表现空间中从许多可能的解中确定一个。

原则 4.2　合法性：对编码的任意排列都对应着一个解。

该原则确保大多数现有的遗传操作可以被简单地应用于该编码。

原则 4.3　完备性：任意解都对应着一个编码。

该原则确保解空间中任意点对于遗传搜索来说都是可达的。

原则 4.4　拉马克性质：某个基因的等位基因的含义不依赖于其他基因。

关于编码的拉马克性质，考虑的是一个染色体能否将其价值通过常用的遗传操作传递到将来的种群中。

原则 4.5　因果性：由于变异带来的基因型的小扰动在表现型空间中也对应着小扰动。

该原则强调保护领域结构，在变异算子带来有用的新信息的同时，它应该保留在对应表现空间中的领域结构。

本节背包问题编码如下：染色体 X 长度为 4，基因 x_i=1 表示物品 t_i 放入背包中，x_i=0 表示物品 t_i 不放入背包中，如 X=1011 表示物品 t_1、t_3、t_4 放入背包中。

4.4.3　群体设定

初始种群是从解中随机选择出来的，将这些解比喻为染色体或基因，该种群被称为第一代种群。一般来讲，初始群体的设定可采取如下策略。

1）根据问题固有知识，设法把握最优解所占空间在整个问题空间中的分布范围，然后，在此分布范围内设定初始群体。

2）先随机生成一定数目的个体，然后从中挑出最好的个体加到初始群体中。这种过程不断迭代，直到初始群体中个体数达到预先确定的规模。

对于种群的规模不宜太大或太小，种群规模太小会导致遗传算法优化性能较差，易陷入局部最优解，种群规模太大则会导致计算复杂。模式定理表明，若群体规模为 M，则遗传操作可从这 M 个个体中生成和检测 M^3 个模式，并在此基础上不断形成和优化，直到找到最优解。

本节问题中群体规模大小为 4，即群体由 4 个个体组成，每个个体可通过随机方法产生，如 0111、1010、0100、1101。

4.4.4　适应度函数

进化论中的适应度，是表示某一个个体对环境的适应能力，也表示该个体繁殖后代的能力。遗传算法的适应度函数也叫评价函数，是用来判断群体中个体优劣程度的指标，它是根据所求问题的目标函数来进行评估的。

在搜索进化过程中，一般不需要其他外部信息，仅用评估函数来评估个体或解的优劣，并作为以后遗传操作的依据。由于适应度函数需要比较排序，并在此基础上计算选择概率，因此要取正值。由此可见，在不少场合，将目标函数映射成求最大值形式且函数值非负的适应度函数是必要的。

适应度函数的设计主要满足以下条件。

1）单值、连续、非负、最大化。

2）合理、一致性。

3）计算量小。

4）通用性强。

在具体应用中，适应度函数的设计需要结合求解问题本身的要求而定。适应度函数设计直接影响遗传算法的性能。

本节问题中目标函数总取非负值，并以求函数最大值为优化目标，故可直接利用目标函数值作为个体的适应度。适应度函数表示如下：

$$f_k = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{V} = \sum_{i=1}^{n} x_i v_i \tag{4.23}$$

4.4.5　选择

选择操作的目的是从当前群体中选出优良的个体，使它们有机会作为父代为下一代繁殖子孙。判断个体优良与否的准则就是各自的适应度值，即个体适应度越高，其被选择的概率就越大。在过去 20 多年中，学者提出、验证并比较了许多选择方法。通常采用的选择方法如下。

1）轮盘赌选择（roulette wheel selection）。

2）$(\mu+\lambda)$ 选择（$(\mu+\lambda)$ selection）。

3）竞争选择（tournament selection）。

4）排序与比例变换（sorting and scaling）。

5）共享（sharing）。

轮盘赌选择是知名的选择方式，其基本原理是根据每个染色体适应度值的比例来确定该个体的选择概率，因此可以建立一个轮盘赌模型来表示这些概率。选择的过程就是旋转轮盘若干次（次数等于种群规模），每次为新种群选出一个父代个体。

$(\mu+\lambda)$选择是一种确定性选择过程，从父代和子代中选取最好的染色体。该方法禁止从种群中选取相同的染色体，因此许多研究人员倾向于采用该选择方式来处理组合优化问题。

竞争选择方式同时包含随机和确定性的特征。该方法随机选择一些染色体，并从中选出最好的进行繁殖。

在比例选择过程中，个体的选择概率与其适应值大小成正比。这种简单的选择方法存在一些我们不希望的性质。例如，在算法进行的早期，个别超级染色体具有控制选择过程的趋势；在算法进行的晚期（种群已大部分收敛），个体之间的竞争并不激烈，呈现出随机搜索的行为。

排序和比例变换机制的目的就是解决这些问题。比例变换方法将原始目标函数值映射为正实数，根据这些正实数计算每个染色体的生存概率。比例变换具有双重目的：①维持染色体相对适应值比例间的合理差异；②在算法早期通过抑制某些超级染色体过快收敛来达到限制竞争的目的，在算法晚期则加速竞争。排序方法忽略了实际的目标函数值，与之相对应的是采用了染色体的排序来确定其生存概率。这种方法对种群进行从好到差的排序，然后根据每个染色体的排序（而不是原始适应度值）确定其选择概率。

共享方法是一种用于根据个体在某个距离内与其他个体相邻的程度来确定该个体适应值下降多少的方法。由于适应值的下降，在拥挤的峰周围的个体的复制概率受到抑制，而其他个体则易于产生后代。

本节问题选择算法采用轮盘赌选择法，根据与适应度成正比的概率来确定各个个体复制到下一代群体中的数量。各染色体的价值与被选择概率如表 4.2 所示。

表 4.2　各染色体价值与被选择概率

染色体 X	重量（W）	适应度值（f）	选择概率（$P=f/F$）	选择范围
1（0111）	12	9	0.281	0～0.281
2（1010）	5	7	0.219	0.281～0.5
3（0100）	5	4	0.125	0.5～0.625
4（1101）	11	12	0.375	0.625～1
		F=32	1	

确定各染色体选择概率后生成 4 个 0～1 之间的随机数，依据随机数出现在表 4.2 中哪个区域来确定各个体被选中的次数及每次选中的个体。表 4.3 所示为 4 个随机数选中的个体。

表 4.3　产生的随机数与选择的个体

产生的随机数	选择的染色体编号
0.91	4
0.36	2
0.45	2
0.05	1

4.4.6　交叉

在自然界生物进化过程中起核心作用的是生物遗传基因的重组（加上变异）。遗传算法中起核心作用的是遗传操作的交叉算子。交叉是指把两个父代个体的部分结构加以替换重组而生成新个体的操作。通过交叉，遗传算法的搜索能力得以飞跃提高。

交叉算子根据交叉率将种群中的两个个体随机地交换某些基因，产生新的基因组合，期望将有益基因组合在一起。根据编码表示方法的不同，可以有以下算法。

1）实值重组（real valued recombination），包括离散重组（discrete recombination）、中间重组（intermediate recombination）、线性重组（linear recombination）和扩展线性重组（extended linear recombination）。

2）二进制交叉（binary valued crossover），包括单点交叉（single-point crossover）、多点交叉（multiple-point crossover）、均匀交叉（uniform crossover）、洗牌交叉（shuffle crossover）和缩小代理交叉（crossover with reduced surrogate）。

常用的交叉算子为单点交叉。具体的操作是在个体串中随机设定一个交叉点，实行交叉时，该点前或该点后的两个个体的结构进行交换，并生成两个新的个体。

本节问题采用单点交叉的方法，其具体操作过程如下。

1）对群体进行随机配对。本例中染色体 1 与染色体 3 进行配对，染色体 2 与染色体 4 进行配对。

2）随机设置交叉点位置。本例中随机交叉点位置为 3，交叉方式为交叉点及交叉点后一位基因互换。

表 4.4 为本次交叉运算结果。

表 4.4　交叉运算结果

染色体 X	交叉结果
1（0111）	0110
2（1010）	1001
3（1010）	1011
4（1101）	1110

4.4.7　变异

变异算子的基本内容是对群体中个体串的某些基因座上的基因值做变动。依据个体编码表示方法的不同，可以有以下算法。

1）实值变异。

2）二进制变异。

一般来说，变异算子操作的基本步骤如下。

1）对群中所有个体以事先设定的变异概率判断是否进行变异。

2）对进行变异的个体随机选择变异位进行变异。

在遗传算法中引入变异的目的有如下两个。

1）使遗传算法具有局部的随机搜索能力。当遗传算法通过交叉算子已接近最优解邻域时，利用变异算子的局部随机搜索能力可以加速向最优解收敛。显然，此种情况下的变异概率应选取较小值，否则接近最优解的解会因变异而遭到破坏。

2）使遗传算法可以维持群体多样性，防止出现未成熟收敛现象。此时变异概率应取较大值。

遗传算法变异概率的选取一般受种群大小、染色体长度等因素的影响，通常选取的值都比较小，一般选取 0.001～0.1。

本节问题中，采用基本位变异的方法来进行变异运算，其具体操作过程如下。

1）随机确定各个染色体的基因变异位置。

2）依照某一概率将变异点的原有基因值取反。

该问题中，设置总的变异概率为 0.1，每个染色体的变异概率为 0.025。当随机数为 0～0.025 时染色体 1 进行变异，随机数为 0.025～0.05 时染色体 2 进行变异，随机数为 0.05～0.075 时染色体 3 进行变异，随机数为 0.075～0.1 时染色体 4 进行变异。表 4.5 为本次变异运算结果。

表 4.5　变异运算结果

染色体 X	交叉运算结果	变异位	随机数	变异结果	适应度值（f）
1（0111）	0110	2	0.57	0110	6
2（1010）	1001	3	0.86	1001	8
3（1010）	1011	2	0.66	1011	10
4（1101）	1110	4	0.08	1111	14

经过一次迭代，满足总重量 $W \leqslant 10$ 的最好解是染色体 3。实际上 1011 是该问题的最优解。

4.4.8　终止条件

当最优个体的适应度达到给定的阈值，或者最优个体的适应度和群体适应度不再上升时，或者迭代次数达到预设的代数时，算法终止。预设的最大代数一般设置为 100～500 代。

本 章 小 结

群集智能是表示行为模式简单的个体在群体层面上涌现出智能的行为。与传统优化

算法相比，群集智能具有鲁棒性强、并行性好、实现相对简单等诸多优势。同时，群集智能也存在一些缺陷。

从生物进化和仿生学角度出发，由研究蚂蚁寻找食物路径的自然行为，提出了蚁群算法。用该方法求解旅行商问题、分配问题和调度等问题，取得了较好的结果。蚁群算法已显示出它在求解复杂优化问题方面的优势，是一种很有发展前景的群集智能方法。

粒子群优化算法是一种基于群体搜索的算法，它是建立在模拟鸟群社会行为的基础上的。粒子在搜索空间中的位置变化是以个体成功地超越其他个体的社会心理意向为基础的。一个粒子的搜索行为受到群体中其他粒子的搜索行为的影响。由此可见，粒子群优化算法是一种共生合作算法。

遗传算法是模仿生物遗传学和自然选择激励，通过人工方式构造的一类搜索算法，是对生物进化过程的一种数学仿真，也是进化计算的重要形式。本章分析了遗传算法的原理与框架、遗传算法的编码、遗传算法的遗传算子、遗传算法的执行过程和执行实例。

本 章 习 题

1．群集智能算法的基本思想是什么？

2．群集智能算法的主要特点是什么？

3．举例说明蚁群算法的搜索原理，并简要叙述蚁群算法的特点。

4．蚁群算法的寻优过程包含哪几个阶段？寻优的准则有哪些？

5．蚁群算法中的参数如何选择？

6．为什么要对蚁群算法进行改进？常见的改进方法有哪些？

7．举例说明粒子群优化算法的搜索原理，并简要叙述粒子群优化算法的特点。

8．粒子群优化算法的寻优过程包含哪几个阶段？寻优的准则有哪些？

9．举例说明遗传算法的搜索原理，并简要叙述遗传算法的特点。

10．遗传算法的寻优过程包含哪几个阶段？寻优的准则有哪些？

第5章　专 家 系 统

5.1　专家系统概述

5.1.1　专家系统的特点

专家系统（expert system，ES）是人工智能应用研究的主要领域。正如专家系统的先驱费根鲍姆所说：专家系统的力量是从它处理的知识中产生的。这正符合一句名言：知识就是力量。

ES 实质上是一个计算机程序，它能够以人类专家的水平完成特别困难的某一专业领域的任务。在设计专家系统时，知识工程师的任务就是使计算机尽可能模拟人类专家解决某些实际问题的决策和工作过程，即模仿人类专家如何运用他们的知识和经验解决所面临问题的方法、技巧和步骤。

对于 ES，存在着不同的定义。

定义 5.1　专家系统是一个智能计算机程序系统，其内部含有大量的某个领域专家水平的知识与经验，能够利用人类专家的知识和解决问题的方法来处理该领域问题。也就是说，专家系统是一个具有大量的专门知识与经验的程序系统，它应用人工智能技术和计算机技术，根据某领域一个或多个专家提供的知识和经验，进行推理和判断，模拟人类专家的决策过程，以解决需要人类专家处理的复杂问题。简言之，专家系统是一种模拟人类专家解决领域问题的计算机程序系统。

此外，还有其他一些关于 ES 的定义。这里首先给出 ES 技术先行者和开拓者费根鲍姆于 1982 年对人工智能的定义。

定义 5.2　Expert system is “an intelligent computer program that use knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solutions.” That is, an expert system is a computer system that emulates the decision-making ability of a human expert. The term emulate means that the expert system is intended to act in all respects like a human expert.（专家系统是“一种智能计算机程序，它使用知识和推理程序来解决难以解决且需要大量人类专业知识才能解决的问题”。也就是说，专家系统是模拟人类专家决策能力的计算机系统。仿真一词意味着专家系统旨在在所有方面都像人类专家一样行事。）

下面是韦斯（Weiss）和库利柯夫斯基（Kulikowski）对 ES 的界定。

定义 5.3　专家系统使用人类专家推理的计算机模型来处理现实世界中需要专家做出解释的复杂问题，并得出与专家相同的结论。

总体上，ES 具有下列特点。

1）启发性。ES 能够运用专家的知识与经验进行推理、判断和决策。世界上的大部分工作和知识都是非数学性的，只有一小部分人类活动是以数学公式或数字计算为核心的（约占 8%）。即使是化学和物理学科，大部分也是靠推理进行思考的。对于生物学、大部分医学和全部法律，情况也是如此。企业管理的思考几乎全靠符号推理，而不是数值计算。

2）透明性。ES 能够解释本身的推理过程并回答用户提出的问题，以便让用户了解推理过程，提高对 ES 的信赖度。例如，一个医疗诊断 ES 诊断某人患有肺炎，必须用某种抗生素治疗，那么，这一 ES 将会向患者解释为什么他患有肺炎，而且必须用某种抗生素治疗，就像一位医疗专家对患者详细解释病情和治疗方案一样。

3）灵活性。ES 能够不断地增长知识，修改原有知识，不断更新。由于这一特点，ES 获得了十分广泛的应用。

随着 ES 的迅速发展，其应用领域越来越广，解决实际问题的能力越来越强，这也是由 ES 的优良性能及其对国民经济的重大作用决定的。具体地说，ES 的优点包括下列几个方面。

1）ES 能够高效率、准确、周到、迅速和不知疲倦地进行工作。

2）ES 解决实际问题时不受周围环境的影响，也不可能出现遗漏或有所忘记。

3）可以使专家的专长不受时间和空间的限制，以便推广珍贵和稀缺的专家知识与经验。

4）ES 能够促进各领域的发展，它使各领域专家的专业知识和经验得到总结和提炼，能够广泛有力地传播专家的知识、经验和能力。

5）ES 能够汇聚和集成多领域专家的知识和经验以及他们协作解决重大问题的能力，它拥有更渊博的知识、更丰富的经验和更强的工作能力。

6）军事 ES 的水平是一个国家国防现代化和国防实力的重要标志之一。

7）ES 的研制和应用具有巨大的经济效益和社会效益。

8）研究 ES 能够促进整个学科技术的发展。ES 对人工智能的各个领域的发展起到了很大的促进作用，并对科技、经济、国防、教育、社会和人民生活产生深远的影响。

5.1.2　专家系统的类型

按照 ES 的特性及功能分类，ES 可分为 10 类，如表 5.1 所示。

表 5.1　专家系统类型

ES 类型	解决的问题
解释	根据感知数据推理情况描述
诊断	根据观察结果推理系统是否有故障
预测	指导给定情况可能产生的结果

续表

ES 类型	解决的问题
设计	根据给定的要求进行相应的设计
规划	设计动作
控制	控制整个系统的行为
监督	比较观察结果和期望结果
修理	执行计划来实现规定的补救措施
教学	诊断、调整、修改学生行为
调试	建议故障的补救措施

1）解释型 ES。解释型 ES 根据感知数据，经过分析、推理，给出相应的解释。例如，化学结构说明、图像分析、语言理解、信号解释、地质解释、医疗解释等领域的 ES。代表性的解释型 ES 有 DENDRAL（推断化学分子结构的专家系统）、PROSPECTOR（探矿专家系统）等。

2）诊断型 ES。诊断型 ES 能够根据取得的现象、数据或事实推断出系统是否有故障，并能找出产生故障的原因，给出排除故障的方案。这是目前开发、应用得最多的一类 ES。例如，医疗诊断、机械故障诊断、计算机故障诊断等领域的 ES。代表性的诊断 ES 有 MYCIN（血液感染病诊断专家系统）、CASNET（青光眼诊断与治疗专家系统）、PUFF（肺功能诊断系统）、PIP（肾脏病诊断系统）、DART（计算机硬件故障诊断系统）等。

3）预测型 ES。预测型 ES 能够根据过去和现在的信息（数据和经验）推断可能发生和出现的情况。例如，用于天气预报、地震预报、市场预测、人口预测、灾难预测等领域的 ES。

4）设计型 ES。设计型 ES 能够根据给定要求进行相应的设计。例如，用于工程设计、电路设计、建筑及装修设计、服装设计、机械设计及图案设计的 ES。对这类系统一般要求在给定的限制条件下能给出最佳或较佳的设计方案。代表性的设计型 ES 有 XCON（计算机系统配置系统）、KBVLSI（VLSI 电路设计系统）等。

5）规划型 ES。规划型 ES 能够按照给定目标拟定总体规划、行动计划、运筹优化等，适用于机器人动作控制、工程规划、军事规划、城市规划、生产规划等领域。这类 ES 一般要求在一定的约束条件下能够以较小的代价达到给定的目标。代表性的规划型 ES 有 NOAH（机器人规划系统）、SECS（制定有机合成规划的 ES）、TATR（帮助空军制定攻击敌方机场计划的 ES）等。

6）控制型 ES。控制型 ES 能够根据具体情况，控制整个系统的行为，适用于对各种大型设备及系统进行控制。为了实现对控制对象的实时控制，控制型 ES 必须具有能够直接接收来自控制对象的信息，并能够迅速地进行处理，及时地做出判断和采取相应行动的能力。所以控制型 ES 实际上是 ES 技术与实时控制技术相结合的产物。代表性的控制型 ES 是 YES/MVS（帮助监控和控制 MVS 操作系统的 ES）。

7）监督型 ES。监督型 ES 能够完成实时的监控任务，并根据监测到的现象做出相应的分析和处理。这类系统必须能够随时收集任何有意义的信息，并能快速地对得到的

信号进行鉴别、分析和处理。一旦发现异常，能尽快做出反应，如发出警报信号等。代表性的监督型 ES 是 REACTOR（帮助操作人员检测和处理核反应堆事故的 ES）。

8）修理型 ES。修理型 ES 是用于制定排除某类故障的规划并实施排除的一类 ES，要求能够根据故障的特点制订纠错方案，并能实施该方案排除故障；当制订的方案失效或部分失效时，能够及时采取相应的补救措施。

9）教学型 ES。教学型 ES 主要适用于辅助教学，并能够根据学生在学习过程中所产生的问题进行分析、评价，找出错误原因，有针对性地确定教学内容或采取其他有效的教学手段。代表性的教学型 ES 是 GUIDON（讲授有关细菌传染性疾病方面医学知识的计算机辅助教学系统）。

10）调试型 ES。调试型 ES 用于对系统进行调试，能够根据相应的标准检测被检测对象存在的错误，并能够从多种纠错方案中选出适用于当前情况的最佳方案，排除错误。

表 5.1 是根据 ES 的特性及功能对 ES 进行的分类。这种分类往往是不确切的，因为许多 ES 具有不止一种功能，还可以从另外的角度对 ES 进行分类。例如，可以根据 ES 的应用领域进行分类。

5.2　专家系统的结构与工作原理

5.2.1　专家系统的一般结构

由 ES 的定义可知，ES 的主要组成部分是知识库和推理机。实际 ES 的功能和结构可能彼此存在差异，但是完整的 ES 一般应包括人机接口、推理机、知识库、综合数据库、知识获取机构、解释机构六部分。各部分关系如图 5.1 所示。

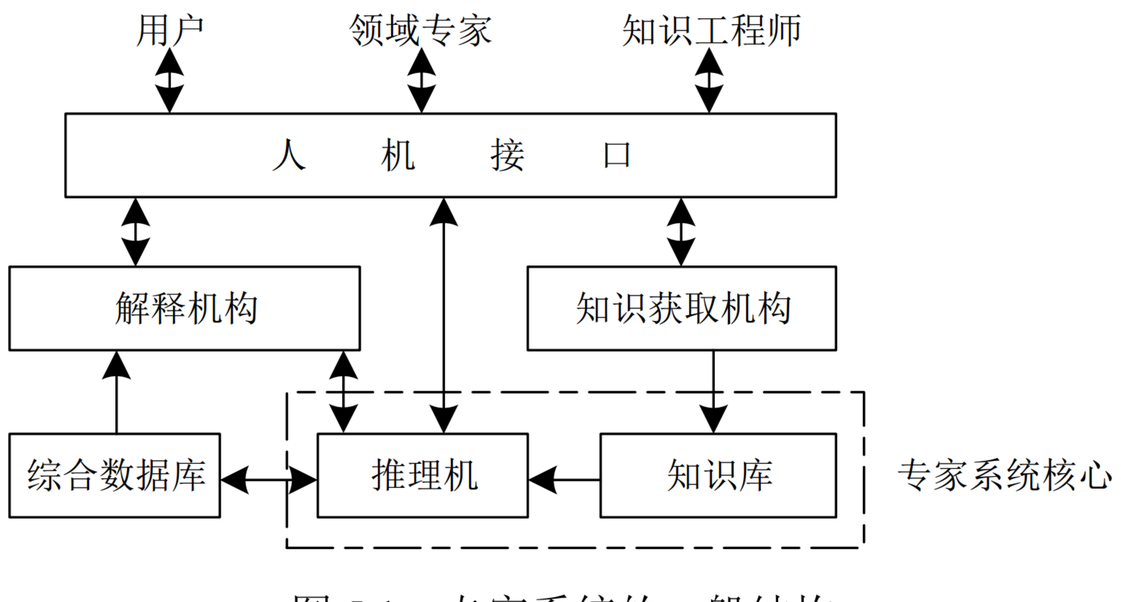


图 5.1　专家系统的一般结构

ES 的核心是知识库和推理机，其工作过程是根据知识库中的知识和用户提供的事实进行推理，不断地由已知的事实推出未知的结论（即中间结果），并将中间结果放到数据库中，作为已知的新事实进行推理，从而把求解的问题由未知状态转换为已知状态。在 ES 的运行过程中，会不断地通过人机接口与用户进行交互，向用户提问，并向用户做出解释。

5.2.2 专家系统的工作原理

（1）知识库

在 ES 中，知识库主要用来存放领域专家提供的专门知识。知识库中的知识源于知识获取机构，同时它又为推理机提供求解问题所需的知识。

建立知识库，首先需要选择合适的知识表达方式。对同一知识，一般可以用多种方式进行表达，但其效果却不同，选择知识表达方式应从能够充分表达领域知识、能够充分有效地进行推理、便于对知识的组织维护和管理、便于理解与实现四个方面进行考虑。

知识库管理系统负责对知识库中的知识进行组织、检索、维护等。ES 中任何其他部分要与知识库发生联系，都必须通过该管理系统来完成。这样可实现对知识库的统一管理和使用。

在进行知识库维护时，还要保证知识库的安全性。必须建立严格的安全保护措施，防止由于操作失误等主观原因使知识库遭到破坏，造成严重的后果。一般知识库的安全保护也可以像数据库系统那样，通过设置口令验证操作者的身份，对不同操作者设置不同的操作权限等技术来实现。

（2）推理机

推理机的功能是模拟领域专家的思维过程，控制并执行对问题的求解。它能够根据当前已知的事实，利用知识库中的知识，按照一定的推理方法和控制策略进行推理，直到得出相应的结论为止。

推理机包括推理方法和控制策略两部分。推理方法有确定性推理和不确定性推理两种。控制策略主要指推理方法的控制及推理规则的选择策略。推理包括正向推理、反向推理和正反向混合推理。

推理机的性能与构造一般与知识的表示方法有关，但是与知识的内容无关，这有利于保证推理机与知识库的独立性，提高 ES 的灵活性。

（3）综合数据库

综合数据库又称动态数据库或黑板，主要用于存放初始事实、问题描述及系统运行过程中得到的中间结果、最终结果等信息。

在开始求解问题时，综合数据库中存放的是用户的初始事实。综合数据库的内容随着推理的进行而变化，推理机根据综合数据库的内容从知识库中选择合适的知识进行推理并将得到的中间结果存放于综合数据库中。综合数据库记录了推理过程中的各种有关信息，又为解释机构提供了回答用户咨询的依据。

综合数据库还必须具有相应的数据库管理系统，负责对数据库中的知识进行检索、维护等。

从计算机技术角度来看，知识库和综合数据库都是数据库。它们所不同的是：知识库的内容在 ES 运行过程中是不改变的，只有知识工程师通过人机接口进行管理；而综合数据库在 ES 运行过程中是动态变化的，不仅可以由用户输入数据，而且推理的中间结果也会改变其内容。

（4）知识获取机构

知识获取是建造和设计 ES 的关键，也是目前建造 ES 的瓶颈。知识获取的基本任务是为 ES 获取知识，建立起健全、完善、有效的知识库，以满足求解领域问题的需要。

知识获取通常是由知识工程师与 ES 中的知识获取机构共同完成的。知识工程师负责从领域专家那里抽取知识，并用适用的方法把知识表达出来；而知识获取机构把知识转换为计算机可存储的内部形式，然后把它们存入知识库。在存储过程中，要对知识进行一致性、完整性的检测。

不同 ES 的知识获取功能与实现方法差别较大，有的系统采用自动获取知识的方法，有的系统采用非自动或半自动的知识获取方法。

（5）人机接口

人机接口是 ES 与领域专家、知识工程师、一般用户之间进行交互的界面，由一组程序及相应的硬件组成，用于完成输入、输出工作。知识获取机构通过人机接口与领域专家及知识工程师进行交互，更新、完善、扩充知识库；推理机通过人机接口与用户交互，在推理过程中，ES 根据需要不断向用户提问，以得到相应的事实数据，在推理结束时会通过人机接口向用户显示结果；解释机构通过人机接口与用户交互，向用户解释推理过程，回答用户问题。

在输入或输出过程中，人机接口需要在内部表示形式与外部表示形式之间进行转换。在输入时，它将领域专家、知识工程师或一般用户输入的信息转换成系统的内部表示形式，然后分别交给相应的机构去处理；输出时，它将系统输出的信息由内部形式转化为人们易于理解的外部形式显示给用户。

在不同的 ES 中，由于硬件、软件环境不同，接口的形式与功能有较大的差别。随着计算机硬件和自然语言理解技术的发展，有的 ES 已经可以用简单的自然语言与用户交互，但是有的 ES 只能通过菜单方式、命令方式或简单的问答方式与用户交互。

（6）解释机构

解释机构回答用户提出的问题，解释系统的推理过程。解释机构由一组程序组成。它跟踪并记录推理过程，当用户提出的询问需要给出解释时，它将根据问题的要求分别做相应的处理，最后将解答用约定的形式通过人机接口输出给用户。

上面讨论的只是 ES 的基本形式。实际上，在具体建造一个 ES 时，随着系统要求的不同，可以在此基础上做适当修改。

5.3　知识的获取

5.3.1　知识获取的方式

按照知识获取的自动化程度划分，知识获取主要有非自动、自动和半自动三种方式。

（1）非自动知识获取

非自动知识获取也称为人工移植。在这种方式中，知识获取分两步进行，首先由知

识工程师从领域专家或有关的技术文献中获取知识，然后再由知识工程师用某种知识编辑软件将知识输入知识库。其工作方式如图 5.2 所示。

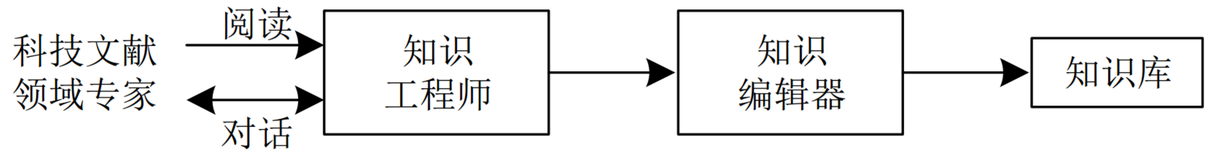


图 5.2　非自动知识获取

非自动方式是 ES 建造中用得较普遍的一种知识获取方式。在非自动获取方式中，知识工程师起着关键的作用。其主要任务如下。

1）与领域专家进行交谈，阅读有关文献，获取 ES 所需要的原始知识。

2）对已获得的原始知识进行分析、归纳、整理，形成用自然语言描述的知识条款，然后返回给领域专家检查。

3）将确定的知识条款用知识表示方法表示出来，用知识编辑器进行编辑输入。

知识编辑器是一种用于知识输入的软件，通常是在建造 ES 时根据实际需要编制的。一般来说，知识编辑器应具有如下主要功能。

1）将用某种模式或语言表示的知识转换成计算机可表示的内部形式并输入知识库。

2）检测输入知识库中的知识的语法错误。

3）检测知识的一致性，报告产生错误的原因，以便进行改正。

（2）自动知识获取

自动知识获取是指系统具有获取知识的能力。它不仅可以直接与领域专家对话，从专家提供的原始信息中学习到 ES 所需的知识，还能从系统自身的运行实践中总结、规划出新的知识，发现知识中可能存在的错误，不断自我完善，建立起性能优良、知识完善的知识库。为达到这一目的，它至少应具备以下能力。

1）具有识别语言、文字、图像的能力。ES 中知识主要来源于领域专家以及有关的科技文献资料、图像等。为了实现知识的自动获取，必须使系统能够与领域专家直接对话，能够阅读有关的科技资料。这就要求系统具有识别语言、文字和图像的能力。只有这样，它才能够直接获得 ES 所需的原始知识，为知识库的建立奠定基础。

2）具有理解、分析、归纳的能力。领域专家提供的知识通常是处理具体问题的实例，不能直接用于知识库。为了将它变为知识库中的知识，必须在理解的基础上进行分析、归纳、提炼、综合，从中抽取出 ES 所需的知识并放入知识库。在非自动知识获取中，这一工作是由知识工程师完成的，而在自动知识获取中，是由系统取代知识工程师完成的。

3）具有从运行实践中学习的能力。在知识库初步建成投入使用后，随着应用向纵深发展，知识库的不完善性会逐渐暴露出来。此时知识的自动获取系统应不断地总结经验教训，从运行实践中学习，产生新的知识，纠正可能存在的错误，不断进行知识库的自我完善。

总之，在自动知识获取系统中，原来需要知识工程师做的工作都由系统取代了，并且还要做得更好。自动获取知识的过程如图 5.3 所示。

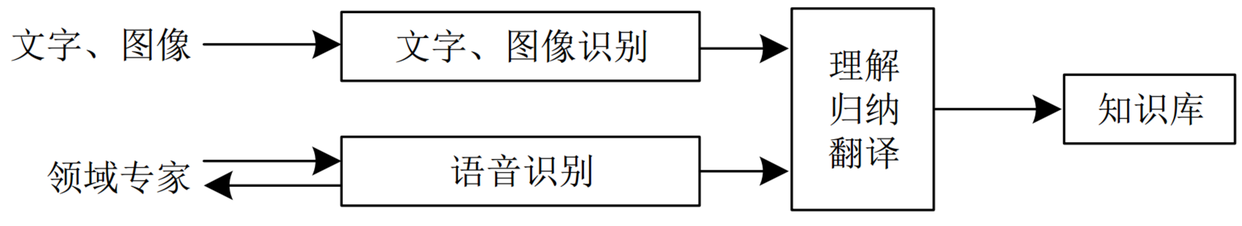


图 5.3　自动知识获取

（3）半自动知识获取

自动知识获取是一种理想的知识获取方式，涉及人工智能的多个领域，目前尚处在研究阶段，如模式识别、自然语言理解、机器学习等，其对硬件亦有较高的要求。这几年在自然语言理解、机器学习方面的研究已取得了较大的进展，在人工神经网络的研究中已提出了多种学习算法。这些都为知识的获取提供了有利条件。因此，在创建知识获取系统时，应充分利用这些成果，逐渐向知识的自动获取过渡，提高其智能程度。事实上，在近些年创建的 ES 中，不同程度地做了这方面的尝试与探讨。在非自动知识获取的基础上增加了部分学习功能，使系统能够从大量事例中归纳出某些知识。由于这样的系统不同于纯粹的非自动知识获取，但是又没有达到完全自动知识获取的程度，因此称为半自动知识获取。

5.3.2　知识获取的步骤

知识获取主要是把用于问题求解的专门知识从某些知识源中提炼出来，并转化为计算机内的表示形式存入知识库。知识源包括专家、书本、相关数据库、实例研究和个人经验等。目前 ES 的知识源主要是领域专家，所以知识获取过程需要知识工程师与领域专家反复交流、共同合作完成，如图 5.4 所示。

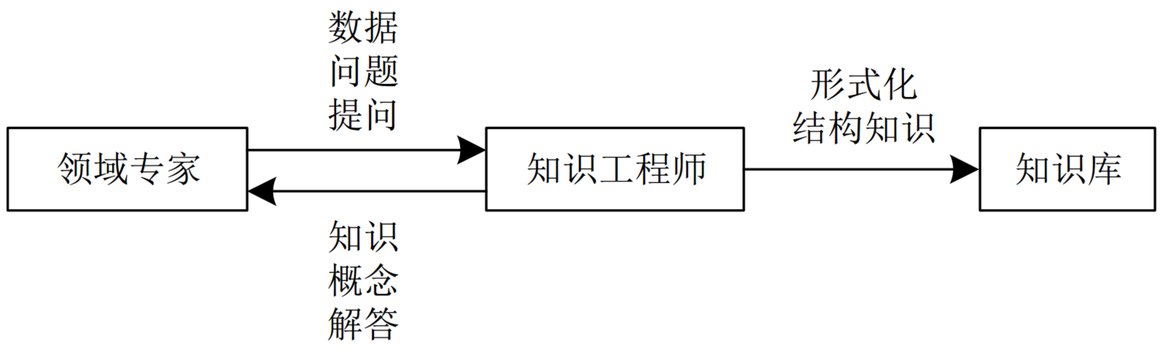


图 5.4　知识获取的过程

知识获取的基本任务是为 ES 获取知识，建立起健全、完善、有效的知识库，以满足求解领域问题的需要。因此，它需要做以下几项工作。

（1）抽取知识

抽取知识是将蕴含于知识源中的知识经识别、理解、筛选、归纳等抽取出来以便用于建立知识库。

知识主要来源于领域专家及相关的专业技术文献，但是知识并不都是以某种现成的形式存在于这些知识源中的。领域专家虽然可以处理领域内的各种困难问题，但是往往缺乏总结能力，不一定能够有条理地说出处理问题的道理和原则；领域专家可以列举大量处理过的实例，但不一定能够建立起相互之间的联系，有时甚至是靠直觉或灵感解决问题。而且，领域专家一般不熟悉 ES 的有关技术。这些都为知识的获取带来了困难。

为了从领域专家处得到有用的知识，需要反复多次地与专家交谈，并有目的地引导交谈的内容，然后通过分析、综合、去粗取精，归纳出可供建立知识库的知识。

知识的另一来源是 ES 自身的运行实践。这就需要从实践中学习，总结出新的知识。一般来说，一个 ES 初步建立后，通过运行会发现知识不够健全，需要补充新的知识。此时除了请领域专家提供进一步的知识外，还可由 ES 根据运行经验从已有的知识或实例中演绎、归纳出新的知识，补充到知识库中。这时 ES 就具有自我学习的能力。

（2）知识的转换

知识的转换是指将知识由一种形式变换为另一种表示形式。

人类专家或科技文献中的知识通常是用自然语言、图形、表格等形式表示的，而知识库中的知识是用计算机能够识别、运用的形式表示的，两者有较大的差距，所以必须将从专家处抽取的知识转换成适合知识库存放的知识。知识转换一般分为两步进行：第一步是将从专家及文献资料处抽取的知识转换为某种知识表示模式，如产生式规则、框架等；第二步是将该模式表示的知识转换为系统可直接利用的内部形式。第一步通常由知识工程师完成，第二步一般通过输入及编译实现。

（3）知识的输入

将某模式表示的知识经编辑、编译送入知识库的过程称为知识的输入。

目前，知识的输入一般是通过两种途径实现的：一种是利用计算机系统提供的编译软件；另一种是用专门编制的知识编辑系统，称为知识编辑器。

（4）知识的检测

知识库的建立是通过知识抽取、转换、输入等环节实现的。这一过程中任何环节上的失误都会造成知识的错误，直接影响 ES 的性能。知识的检测是发现并纠正知识库中的知识可能存在的不一致、不完整的问题，并采取相应的修正措施。

5.4 专家系统的应用

当前专家系统主要的应用领域有医学、地质学、计算机系统、化学、数学、工程、军事等。表 5.2 按应用领域列出了典型专家系统及其功能。

表 5.2 各应用领域中的典型专家系统及其功能

领域	系统	功能
医学	MYCIN	细菌感染性疾病诊断和治疗
	CASNET	青光眼的诊断和治疗
	PIP	肾脏病诊断
	INTERNIST	内科病诊断
	PUFF	肺功能试验结果解释
	ONCOCIN	癌症化学治疗咨询
	VM	人工心肺机监控

续表

领域	系统	功能
地质学	PROSPECTOR	帮助地质学家评估某一地区的矿物储量
	DIPMETER ADVISOR	油井记录分析
	DRILLING ADVISOR	诊断和处理石油井设备的“钻头黏着”问题
	MUD	诊断和处理同钻探泥浆有关的问题
	HYDRO	水源总量咨询
	ELAS	油井记录解释
计算机系统	DART	计算机硬件系统故障诊断
	RI/XCON	配置 VAX 计算机
	YES/MVS	监控和控制 MVS 操作系统
	PTRANS	管理 DEC 计算机系统的建造和配置
	IDT	定位 PDP 计算机中有缺陷的单元
化学	DENDRAL	根据质谱数据来推断化合物的分子结构
	MOLGEN	DNA 分子结构分析和合成
	CRYSALIS	通过电子云密度图推断一个蛋白质的三维结构
	SECS	帮助化学家制定有机合成规划
	SPEX	帮助科学家设计复杂的分子生物学实验
数学	MACSYMA	数学问题求解
	AM	从基本的数学和集合论中发现概念
工程	SACON	帮助工程师发现机构分析问题的分析策略
	DELTA	帮助识别和排除机车故障
	REACTOR	帮助操作人员检测和处理核反应堆事故
军事	AIRPLAN	用于航空母舰周围的空中交通运输计划的安排
	HASP	海洋声呐信号识别和潜艇跟踪
	TATR	帮助空军制订攻击地方机场的计划
	RTC	通过解释雷达图像进行舰船分类

5.5　专家系统的局限性

ES 仍有一些局限与不足，具体如下：

1）知识获取困难。为了构建 ES，需要与人类专家共同探讨对于给定的领域，哪些规则适用，哪些规则不适用。规则，即 ES 需要存储的知识，是很难明确揭示的。人类专家不能明确地将他所用到的规则罗列出来。另外，专家们往往很难确切地表达出他们用到的真正有效的知识。基于这些原因，知识获取成为构建传统 ES 的瓶颈。

2）ES 不具备记忆功能。例如，对于一个医疗诊断系统而言，当输入一个新的患者的症状时，系统调用成百上千条规则进行成百上千次推理，最终做出诊断和治疗。随后需要对有同样症状的其他患者进行诊断时，系统不得不再次调用成百上千条规则进行成百上千次推理。因为传统的 ES 无法记忆以前处理过的症状，所以不得不浪费大量的时间进行重复的劳动，直接导致系统效率的低下。更为严重的是，由于 ES 没有记忆功能，无法记忆本身的错误，因此注定在今后要犯同样的错误，除非人为进行修改。

3）ES 不是鲁棒系统。如果 ES 对于一个问题找不到任何匹配的规则，则系统无法

给出任何结论。系统的知识库局限于存储的规则，如果没有规则可以应用，则系统也无法给出备用选项。因此，ES 是脆弱的。

此外，ES 还有在特殊情况下无法做出创造性的回应、维护成本高等局限性。

本 章 小 结

ES 是基于知识的系统，用于在某种特定的领域中运用领域专家多年积累的经验和专业知识，求解需要专家才能解决的困难问题。

ES 具有专家水平的专业知识，能够进行有效的推理，具有启发性、灵活性、透明性、交互性等特点。

ES 本身是一个程序，但是它与传统程序不同。

ES 一般包括人机接口、推理机、知识库、综合数据库、知识获取结构、解释机构六部分。

ES 的知识获取方式主要有非自动、自动和半自动三种，知识获取的过程可分为知识抽取、知识的转换、知识的输入和知识的检测。

ES 已应用于多个领域并取得长足发展，但其仍有一些局限性。

本 章 习 题

1. 什么是专家系统？它有哪些基本特征？
2. 专家系统由哪几部分组成？各部分的功能和结构是什么？
3. 专家系统和传统程序有何不同与相似之处？
4. 专家系统的主要类型有哪些？
5. 专家系统的知识获取方式有哪些？
6. 专家系统的知识获取过程是什么？
7. 专家系统的主要应用领域有哪些？
8. 专家系统有哪些局限性？

第6章　机 器 学 习

6.1　机器学习概述

6.1.1　机器学习的定义

机器学习是一门交叉性学科，它致力于研究如何通过计算的手段，利用经验来改善系统自身的性能。在计算机系统中，“经验”通常以“数据”的形式存在，因此，机器学习所研究的主要内容，是关于在计算机上从数据中产生“模型”的算法，即“学习算法”。有了学习算法，使用者只要把经验数据提供给它，它就能基于这些数据产生模型；在面对新的情况时，模型会给使用者提供判断。如果说计算机科学是研究关于“算法”的学问，那么类似地，可以说机器学习是研究关于“学习算法”的学问。

6.1.2　机器学习的主要策略

学习是一项复杂的智能活动，学习过程与推理过程是紧密相连的，按照学习中使用推理的多少，机器学习所采用的策略可分为机械学习、类比学习、示例学习、解释学习以及基于神经网络的学习。

（1）机械学习

机械学习也称记忆学习，它是通过记忆和评价外部环境所提供的信息来达到学习目的的。在这种学习方法中，学习环节对外部提供的信息不进行任何变换，只进行简单的记忆。虽然机械学习在方法上看似简单，但是由于计算机的存储容量相当大，检索速度又相当快，记忆精确、无丝毫误差，也能产生人们难以预料的结果。机械学习的典型代表是西蒙的西洋跳棋程序。该程序用极大、极小博弈树搜索来选择走法。学习环节记忆了棋局态势和倒推得出的极大值、极小值。在下棋过程中，只要碰到过去出现过的棋局，就可以直接采用原来的走棋方案。

（2）类比学习

类比学习系统只能得到完成类似任务的有关知识，学习系统必须能够发现当前任务与已知任务的相似点，由此制订完成当前任务的方案，因此，它需要更多的推理。

（3）示例学习

示例学习也称为实例学习，它是一种从具体示例中推导出一般性知识的归纳学习方法，这种学习方法给学习者提供某一概念的一组正例和反例，学习者从这些例子中归纳出一个总的概念描述，并使这个描述适合所有的正例，排除所有的反例。

（4）解释学习

解释学习是一种分析学习方法。这种方法是在领域知识的指导下，通过对单个问题求解例子的分析，构造出求解过程的因果结构，并获取控制知识，便于以后将其用于类似问题的求解。解释学习在获取控制性知识、精华知识、软件重用、计算机辅助设计和计算机辅助教育等方面有较多应用。

（5）基于神经网络的学习

基于神经网络的学习也称为连接学习，是指神经网络的训练过程，其主要表现为连接权值的调整，它利用神经网络解决问题，一般分为训练和工作两个阶段。训练阶段的主要目的是从训练样本中提取隐含知识和规律，并存储在网络中，供工作阶段解决问题使用。

6.1.3 机器学习系统的基本结构

一个学习系统不仅与环境和知识库有关，还应该包含学习和执行两个重要环节，它们之间的关系如图 6.1 所示。

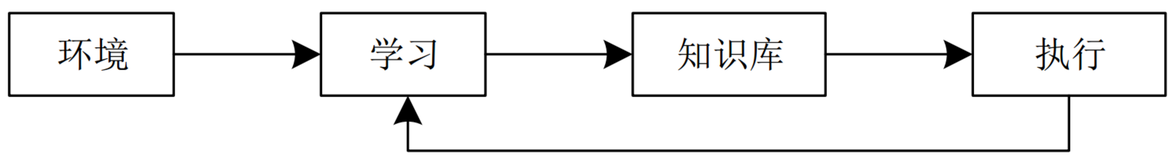


图 6.1 机器学习系统的基本结构

图中给出了学习系统的基本模型，它由四部分组成。其中，环境是以某种形式表达的外界信息集合，它代表外界信息来源；学习环节是将外界信息加工为知识的过程，它先从环境获取外部信息，然后通过对这些信息的分析、综合、类比、归纳等加工形成知识，最后把所形成的知识放入知识库中；知识库是以某种形式表示的知识集合，用来存放学习环节所得到的知识；执行环节是利用知识库中的知识完成某种任务的过程，并把完成任务过程中所获得的一些信息反馈给学习环节，以指导进一步的学习。

（1）环境与学习环节

环境中信息的水平和质量是影响学习系统设计的第一个重要因素。信息的水平是指信息的一般化程度，或者说是指信息适用范围的广泛性。信息的一般化程度又是相对于执行环节而言的。高水平信息的一般化程度比较高，能适应更广泛的问题。低水平信息的一般化程度比较低，只适用个别问题。无论环境中信息的水平是高是低，这些信息与执行环节所需要的信息水平往往是有差距的，学习环节的任务就是缩小这一差距。如果环境提供的是高水平信息，学习环节就要补充遗漏的细节，以便执行环节能够将其用于更具体的情况。如果环境提供的是低水平信息，学习环节就要由这些具体实例归纳出适用一般情况的规则，以便执行环节能够将其用于更广的任务。

（2）知识库环节

知识库的形式和内容是影响学习系统设计的又一重要因素。知识库的形式是与知识表示方式直接相关的。知识的表示有多种形式，如特征向量、一阶逻辑语句、产生式规则、语义网络和框架等。这些表示方式各具特点，在选择表示方式时要兼顾以下几个方

面：表达能力强、易于推理、容易修改知识库、知识表示易于扩展。对于知识库中的内容，其初始知识是非常重要的。学习系统不可能在没有任何知识的情况下凭空获取知识，它总是先利用初始知识去理解环境提供的信息，并依次逐步进行学习。学习系统的学习过程实质上是对原有知识库的扩充和完善过程。

（3）执行环节

执行环节是整个学习系统的核心，它与学习环节之间是相互联系的。学习环节的目的就是要改善执行环节的行为，而执行环节的复杂度、反馈作用以及透明性又会反过来对学习环节产生一定的影响。

1）复杂度。不同复杂度的任务，所需要的知识是不一样的。一般来说，一个任务越复杂，它所需要的知识就会越多。

2）反馈。所有的学习系统都必须有从执行环节到学习环节的反馈信息。这种反馈信息是根据执行环节的执行情况，对学习环节所获知识的评价。学习环节主要根据这些反馈信息来决定是否还需要从环境中进一步获取信息，修改、完善知识库中的知识。目前，学习系统所采用的评价方式主要有两种：一种是由系统自动进行评价，另一种是由人来协助完成评价。由系统自动完成评价，是指将评价时所需要的性能指标直接建立在学习系统中，然后由系统对执行环节得到的结果自动进行评价；由人来协助完成评价，是指由人提出外部执行标准，然后观察执行环节相对这个标准的执行情况，并将比较结果反馈给学习环境。

3）透明性。透明性是指根据系统执行部分的动作效果可以容易地对知识库的规则进行评价。可见，执行环节的透明性应该越高越好。

6.2　监 督 学 习

监督学习（supervised learning）也称为监督训练，是指利用一组已知类别的样本调整分类器的参数，使其达到所要求性能的过程，该过程中有指导者。对于给出的数据集中的每个样本有相应的“正确答案”。

监督学习主要分为回归（regression）、分类（classification）、降维（dimensionality reduction）、结构化预测（structured prediction）、异常检测（anomaly detection）。对于大部分算法来说，只需要稍做修改就可以应用于多种情况，如下文介绍的 k 最近邻（k-nearest neighbor，KNN）算法，不仅可以应用于分类问题，同时也可以应用于回归场景。本节主要介绍算法的原理，对其具体应用场景不做限制。

6.2.1　k 最近邻算法

k 最近邻算法是一种常用的监督学习方法，其工作机制非常简单：给定测试样本，基于某种距离度量找出训练集中与其最靠近的 k 个训练样本，然后基于这 k 个“邻居”的信息进行预测。通常，在分类任务中可使用“投票法”，即选择 k 个样本中出现最多的类别标记作为预测结果；在回归任务中可使用“平均法”，即将 k 个样本的实值输出

标记平均值作为预测结果；还可基于距离远近进行加权平均或加权投票，距离越近的样本权重越大。

KNN 算法可以抽象为三大步骤。

1）算距离：给定测试样本，计算它与训练集中的每个样本的距离。

2）找邻居：圈定距离最近的 k 个训练样本，作为测试样本的近邻。

3）做预测：根据这 k 个训练样本的标记值，推测出测试样本的值。

与其他学习方法相比，KNN 有一个明显的不同之处：它似乎没有显式的训练过程。事实上，它是“懒惰学习（lazy learning）”的著名代表，此类学习技术在训练阶段仅仅是把样本保存起来，训练时间开销为零，待收到测试样本后再进行处理；相应地，那些在训练阶段就对样本进行学习处理的方法，称为“急切学习（eager learning）”。

1．k 值的选取

在 KNN 中，k 是一个重要参数，当 k 取不同值时，分类结果会有显著不同。如果选取较小的 k 值，则意味着整体模型会变得复杂，由于此时预测只考虑了较少的邻居样本，因此容易发生过拟合问题。过拟合，就是模型在训练集上表现非常好，但是在测试集上表现不佳。如果选取较大的 k 值，则相当于用较大邻域中的训练数据进行预测，这时与输入样本较远（不相似）的训练样本也会起到预测作用，使得预测发生错误，k 值的增大意味着整体模型变得简单，适当增大 k 值会降低过拟合风险。

一般怎么选取 k 值呢？在实际应用中，一般将某个范围内的数值作为 k 值的候选，通常采取交叉验证法对这个范围内的每个取值进行效果验证，选择效果最佳的值作为 k 值。也就是说，选取 k 值的关键是实验调参，通过不断地调整 k 的取值来得到一个比较好的结果。

下面通过一个简单的例子来进行说明。如图 6.2 所示，绿色圆要被决定赋予哪个类，是红色三角形还是蓝色三角形？如果 k=3，由于红色三角形所占比例为 2/3，绿色圆将被赋予红色三角形类；如果 k=5，由于蓝色三角形所占比例为 3/5，绿色圆将被赋予蓝色三角形类。

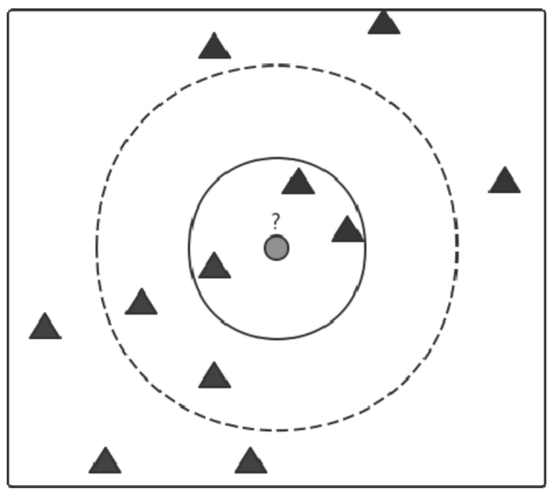


图 6.2　取不同的 k 值，分类结果发生变化

2．k 距离计算方式的选取

除了 k 的取值，若采用不同的距离计算方式，则找出的“近邻”可能有显著差别，从而也会导致分类结果有显著不同。常用的距离计算公式包括欧几里得距离（Euclidean

distance），又称欧式距离、曼哈顿距离（Manhattan distance）等。

在实现算法的过程中，需要根据具体的场景来选择合适的距离计算方式，为了保证每个维度的特征在计算距离的过程中保持同等的重要性，还需要提前对特征进行归一化处理，消除不同特征之间由于不同量纲带来的影响。

例如，用一个人的身高（cm）和鞋码（尺码）作为特征，性别作为预测类别，现在有 5 个训练样本和一个待预测样本（序号 6），分布如表 6.1 所示。

表 6.1　个人信息分布表

序号	身高/cm	鞋码/尺码	标签（性别）
1	179	42	男
2	178	43	男
3	165	36	女
4	177	42	男
5	160	35	女
6	167	43	?

通过上面的样本，很容易看到，第一维身高特征是第二维鞋码的 4 倍左右，那么在进行距离度量的时候，就会更偏向于第一维特征。这样就造成了两个特征并不是等价重要的，最终可能导致距离计算错误，从而导致预测错误。

例如，现在需要预测样本 6 是男性还是女性，选取 k=3 来预测。通过欧式距离计算出样本 6 与每个样本的距离，然后选取最近的 3 个，多数类别就是样本 6 的预测类别，计算如下：

$$D(1,6)=\sqrt{(167-179)^2+(43-42)^2}=\sqrt{145}$$

$$D(2,6)=\sqrt{(167-178)^2+(43-43)^2}=\sqrt{121}$$

$$D(3,6)=\sqrt{(167-165)^2+(43-36)^2}=\sqrt{53}$$

$$D(4,6)=\sqrt{(167-177)^2+(43-42)^2}=\sqrt{101}$$

$$D(5,6)=\sqrt{(167-160)^2+(43-35)^2}=\sqrt{113}$$

由计算得到，最近的三个样本分别是 3、4、5，那么得到最终的预测类别为女性。而实际上，一个女性的鞋码为 43 的可能性要远远小于男性鞋码为 43 的可能性，为什么算法的预测结果为女性呢？是因为两个特征的量纲不同，在这里导致身高的重要性远远大于鞋码，从而主导了模型的预测结果，这是不客观的。应该通过归一化的方式，让每个特征具有同等的重要性。归一化的公式如下：

$$X_{\text{norm}}=\frac{X-X_{\min}}{X_{\max}-X_{\min}} \tag{6.1}$$

式中，$X_{\max}$ 和 $X_{\min}$ 分别为变量 X 的最大值和最小值。

综上所述，KNN 是一种简单、易于理解、容易实现的学习算法，它无须估计参数，也不需要事先训练，就能取得不错的结果。KNN 也存在明显的不足之处，除 k 值的选择和距离计算方式的选择外，KNN 还容易受到样本数量不平衡的影响。同时，为了选

择 k 个邻居，每一个测试样本都需要计算其与所有训练样本的距离，巨大的计算量会影响模型的预估性能。

6.2.2 线性回归

“回归”这个词源于弗朗西斯·高尔顿，他发现高个子父亲的儿子身高会矮一些，而矮个子父亲的儿子身高会高一些（否则高个子家族会越来越高，而矮个子家族会越来越矮）。也就是说，人类的身高都会回归到平均值附近，他将这种现象称为均值回归。

回归大致可以理解为根据数据集 D，拟合出近似的曲线，所以回归也常被称为拟合。如图 6.3 所示，像其右侧图一样拟合出来的直线效果称为线性回归。

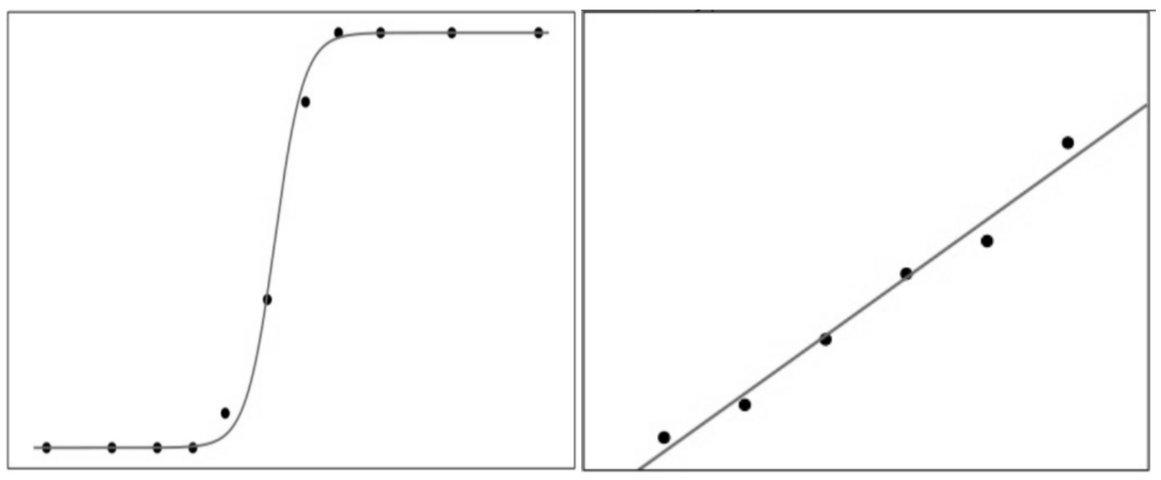


图 6.3 回归曲线图

线性回归分析（linear regression analysis）是确定两种或两种以上变量之间相互依赖的定量关系的一种统计分析方法。需要注意的是，线性模型可以通过基函数的方式画出曲线分类线，“线性”限制的是参数，而不是自变量。

给定数据集 $D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}$，其中 $\boldsymbol{x}_i=(x_{i1},x_{i2},\cdots,x_{id}),y_i\in\mathbf{R}$，线性回归试图学习一个线性模型尽可能准确地预测输出标签，其过程如式（6.2）所定义：

$$Y=b+\boldsymbol{w}_1\boldsymbol{x}_1+\boldsymbol{w}_2\boldsymbol{x}_2+\cdots+\boldsymbol{w}_i\boldsymbol{x}_i+\varepsilon \tag{6.2}$$

式中，Y 为因变量，也称为目标变量；x 为自变量，也称为解释变量；ε 为随机误差。即线性回归试图找到：

$$f(\boldsymbol{x}_i)=\boldsymbol{w}\boldsymbol{x}_i+b,\ \text{使得}f(x_i)\approx y_i \tag{6.3}$$

式中，$\boldsymbol{w}$ 为参数；$\boldsymbol{x}_i$ 为样本 i 的样本值；y_i 为与样本 i 对应的目标值；偏置 b 的存在是为了补偿目标值的平均值（在训练集上的）与基函数值的加权平均值之间的差距。

为了使线性模型更好地拟合数据集，通常使用梯度下降的方法来最小化均方误差函数，即

$$(\boldsymbol{w}^*,b^*)=\arg\min_{(\boldsymbol{w},b)}\sum_{i=1}^{m}(f(\boldsymbol{x}_i)-y_i)^2=\arg\min_{(\boldsymbol{w},b)}\sum_{i=1}^{m}(y_i-\boldsymbol{w}\boldsymbol{x}_i-b)^2 \tag{6.4}$$

可以看出，最小二乘法得到的误差函数是一个凸函数，可以看作一元二次方程，目标是求该方程的最小值。求解 $\boldsymbol{w}$ 和 b 使损失 $E(\boldsymbol{w},b)=\sum_{i=1}^{m}(y_i-\boldsymbol{w}\boldsymbol{x}_i-b)^2$ 最小化的过程，称为线性回归模型的最小二乘参数估计。可将 $E(\boldsymbol{w},b)$ 分别对 $\boldsymbol{w}$ 和 b 求导，得到

$$\frac{\partial E(\boldsymbol{w},b)}{\partial \boldsymbol{w}}=2\left(\boldsymbol{w}\sum_{i=1}^{m}\boldsymbol{x}_i^2-\sum_{i=1}^{m}(y_i-b)\boldsymbol{x}_i\right) \tag{6.5}$$

$$\frac{\partial E(\boldsymbol{w},b)}{\partial b}=2\left(mb-\sum_{i=1}^{m}(y_i-\boldsymbol{w}\boldsymbol{x}_i)\right) \tag{6.6}$$

然后令 $\frac{\partial E(\boldsymbol{w},b)}{\partial \boldsymbol{w}}=0$ 和 $\frac{\partial E(\boldsymbol{w},b)}{\partial b}=0$，可以得到 $\boldsymbol{w}$ 和 b 的最优解：

$$\boldsymbol{w}=\frac{\sum_{i=1}^{m}y_i(\boldsymbol{x}_i-\overline{x})}{\sum_{i=1}^{m}\boldsymbol{x}_i^2-\frac{1}{m}\left(\sum_{i=1}^{m}\boldsymbol{x}_i\right)^2},\quad b=\frac{1}{m}\sum_{i=1}^{m}(y_i-\boldsymbol{w}\boldsymbol{x}_i) \tag{6.7}$$

式中，$\overline{x}=\frac{1}{m}\sum_{i=1}^{m}\boldsymbol{x}_i$ 为 x 的均值。

在历史上，高尔顿总共采集了近千个父亲-孩子身高的数据来拟合。这里从中抽取 6 个作为数据集 D，父亲-孩子身高对应表及分布散点分别如表 6.2、图 6.4 所示。

表 6.2　父亲-孩子身高对应表　　单位：m

父亲身高（x）	孩子身高（y）
1.51	1.63
1.64	1.70
1.60	1.71
1.73	1.72
1.82	1.76
1.87	1.86

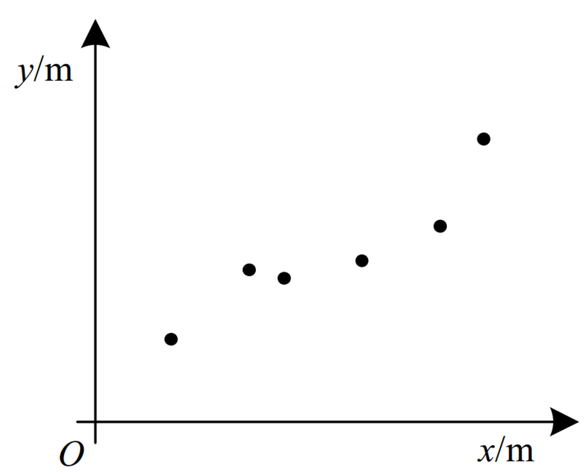


图 6.4　父亲-孩子身高分布散点图

代入线性回归的均方误差后可得

$$E(\boldsymbol{w},b)=\sum_{i=1}^{m}(y_i-\boldsymbol{w}\boldsymbol{x}_i-b)^2=\begin{bmatrix}(1.63-1.51w-b)^2+(1.70-1.64w-b)^2\\+(1.71-1.60w-b)^2+(1.72-1.73w-b)^2\\+(1.76-1.82w-b)^2+(1.86-1.87w-b)^2\end{bmatrix}$$

通过令 $\frac{\partial E(\boldsymbol{w},b)}{\partial \boldsymbol{w}}=0$ 和 $\frac{\partial E(\boldsymbol{w},b)}{\partial b}=0$，直接求解可以得到 $\boldsymbol{w}\approx0.51$ 以及 $b\approx0.86$。得到

的结果与高尔顿得出的结果几乎一样。父亲-孩子身高拟合直线图如图 6.5 所示。

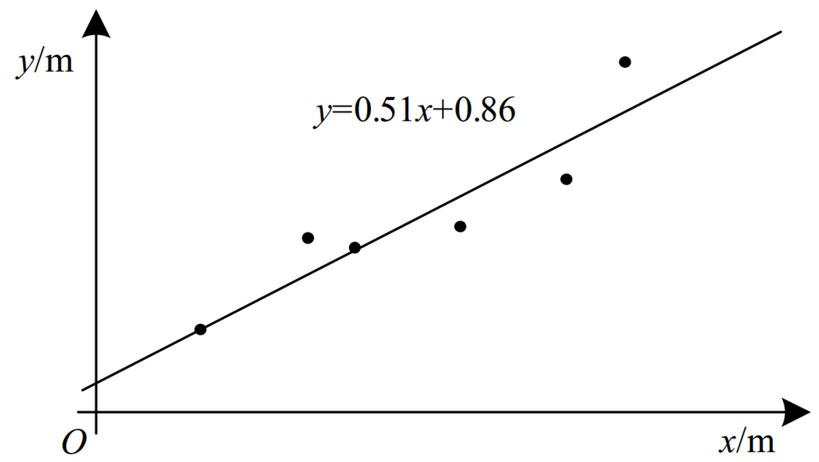


图 6.5　父亲-孩子身高拟合直线图

线性回归模型虽然简单，但是每个输入特征都有对应的权值解释，这在一定程度上可以反映预测输出跟输入特征之间的强弱关系，增强模型的可解释性，因此线性回归在数据足够多的情况下也可以产生很不错的效果，依旧是工业界使用广泛、效果非常好的模型。

6.2.3　决策树

1．基本流程

决策树（decision tree）是一类常见的机器学习方法。以二分类任务为例，希望从给定训练数据集学得一个模型用以对新示例进行分类，这个把样本分类的任务，可看作对“当前样本属于正类吗？”这个问题的“决策”或“判定”过程。顾名思义，决策树是基于树结构来进行决策的，这恰是人类在面临决策问题时一种很自然的处理机制。例如，在相亲场景下，女孩子根据信息对“是否要和男孩子见面？”这样的问题进行决策时，通常会进行一系列的判断或“子决策”：“高不高”“帅不帅”“文化程度怎么样”“收入水平怎么样”等，最后得出是否要见面的决策，这个决策的过程如图 6.6 所示。

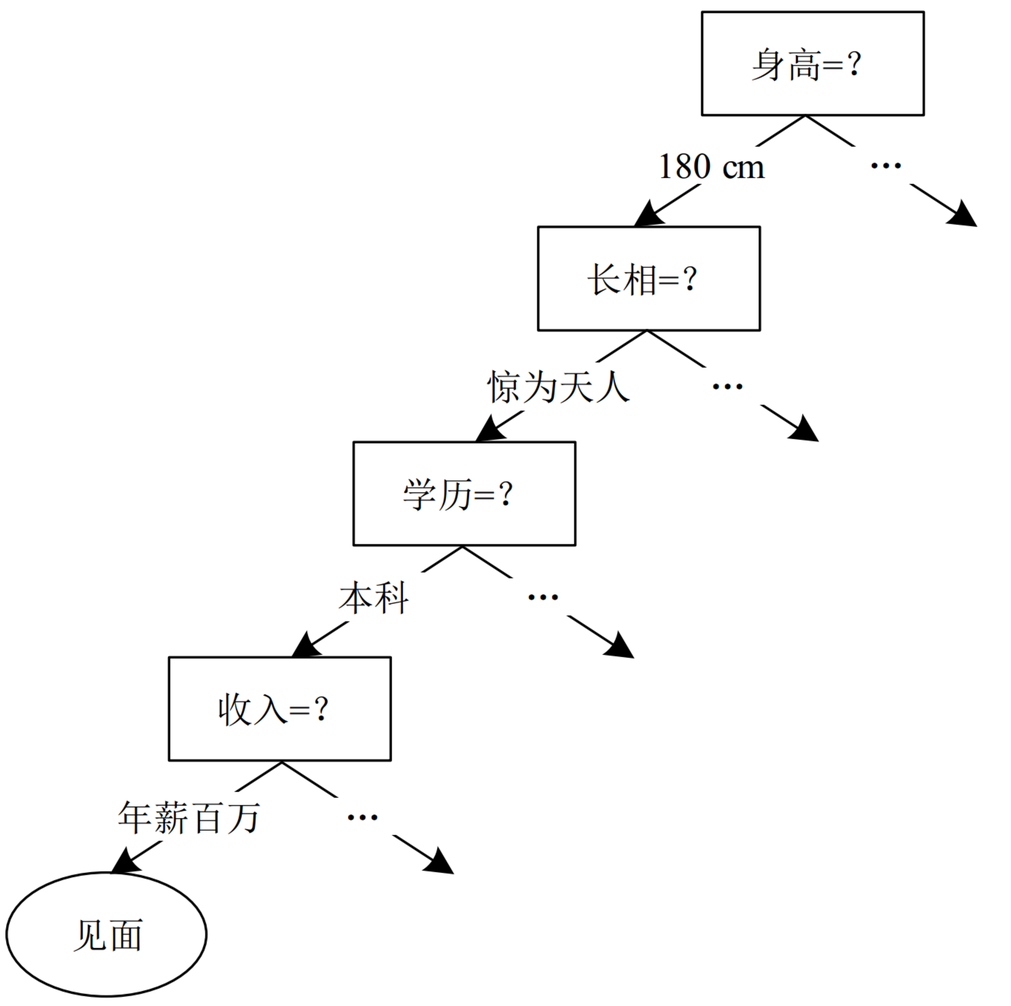


图 6.6　相亲问题的一棵决策树

显然，决策过程的最终结论对应了所希望的判定结果，如“见面”或“不见面”；决策过程中提出的每个判定问题都是对某个属性的“测试”，如“身高=?”“长相=?”；每个测试的结果，或是导出最终结论，或是导出进一步的判定问题，其考虑范围都是在上一次决策结果的限定范围之内。例如，若在“身高=180 cm”之后再判断“长相=?”，则表示仅考虑身高为 180 cm 的男生的范围。

一棵决策树的生成过程主要分为以下三个部分。

1）特征选择：从训练数据众多的特征中选择一个特征作为当前节点的分裂标准，如何选择特征有着很多不同量化评估标准，从而衍生出不同的决策树算法。

2）决策树生成：根据选择的特征评估标准，从上至下递归地生成子节点，直到数据集不可分则停止决策树的生长。对树结构来说，递归结构是容易理解的方式。

3）剪枝：决策树容易过拟合，一般来说需要剪枝，以缩小树结构规模，缓解过拟合。剪枝技术有预剪枝和后剪枝两种。

一般地，一棵决策树包含一个根节点、若干个内部节点和若干个叶节点；叶节点对应于决策结果，其他每个节点对应一个属性测试；每个节点包含的样本集合根据属性测试的结果被划分到子节点中；根节点包含样本全集。从根节点到每个叶节点的路径对应一个判定测试序列。决策树学习的目的是产生一棵泛化能力强(即处理未见示例能力强)的决策树，其基本流程遵循简单且直观的“分而治之（divide and conquer)”策略。决策树学习的基本算法流程如下：

输入：

训练集 $D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}$；

属性集 $A=\{a_1,a_2,a_3,\cdots,a_d\}$；

过程：函数 TreeGenerate(D, A)

1）生成节点 node；

2）if D 中样本全属于同一类别 C then

3）　　将 node 标记为 C 类叶节点；return　　*递归返回，情形 1）

4）end if

5）if $A=\Phi$ or D 中样本在 A 上取值相同 then

6）　　将 node 标记为叶节点，其类别标记为 D 中样本数最多的类；return

*递归返回，情形 2）

7）end if

8）从 A 中选择最优划分属性 a_*；

9）for a_*的每一个值 a_*^v do

10）　　为 node 生成一个分支；令 D_v 表示 D 在 a_*上取值为 a_*^v 的样本子集；

11）　　if D_v 为空 then

12）　　将分支节点标记为叶节点，其类别标记为 D 中样本最多的类；return

*递归返回，情形 3）

13）　　else

14）　　以 TreeGenerate(D_v,$A\backslash\{a_*\}$)为分支节点　　*从 A 中去掉 a_*

15）　end if

16）end for

输出：以 node 为根节点的一棵决策树

决策树的生成是一个递归过程，有三种情形会导致递归返回。

1）当前节点包含的样本全属于同一类别，无须划分。

2）当前属性为空或者所有样本在所有属性上取值相同，无须划分。在这种情况下，将当前节点标记为叶节点，并将其类别设定为该节点所含样本最多的类别。

3）当前节点包含的样本集为空，不能划分，同样把当前节点标记为叶节点，将其类别设定为其父节点所含样本最多的类别。

注意，2）和 3）这两种情形的处理实质不同：情形 2）是利用当前节点的后验分布，而情形 3）则是把父节点的样本分布作为当前节点的先验分布。

从上述决策树学习的过程中可以看出，决策树学习的关键在于如何选择最优划分属性。一般而言，随着划分过程不断进行，希望决策树的分支节点所包含的样本尽可能地属于同一类别，即节点的“纯度”越来越高。

对“纯度”的度量方式并不是唯一的，每一种决策树算法都遵循自己的衡量标准来计算通过不同属性进行分支选择后的分类情况，找到最好的那个当成分裂点，以此类推。

2．常见决策树算法

ID3 算法是最早提出的一种决策树算法，ID3 算法的核心是在决策树各个节点上应用信息增益准则来选择特征，递归构建决策树。

信息熵是度量样本集合纯度常用的一种指标，假定当前样本集合 D 中第 k 类样本所占比例为 $p_k(k=1,2,\cdots)$，则 D 的信息熵定义为

$$\mathrm{Ent}(D)=-\sum_{k=1}^{|y|}p_k\log_2 p_k \tag{6.8}$$

Ent(D)的值越小，则 D 的纯度越高。

假定离散属性 a 有 V 个可能的取值 $\{a^1,a^2,\cdots,a^V\}$，若使用 a 对样本集 D 进行划分，则会产生 V 个分支节点，其中第 $i(i=1,2,\cdots,V)$个分支节点包含了 D 中所有在属性 a 上取值为 a^i 的样本，记为 D^i。根据式（6.8）计算出 D^i 的信息熵，再考虑不同分支上的节点所包含的样本数量不同，给分支节点赋予权重$|D^i|/|D|$，即样本越多影响越大，于是可以计算出用属性 a 对样本集 D 进行划分所获得的信息增益。

$$\mathrm{Gain}(D,a)=\mathrm{Ent}(D)-\sum_{i=1}^{V}\frac{|D^i|}{|D|}\mathrm{Ent}(D^i) \tag{6.9}$$

一般而言，信息增益越大，意味着使用该属性进行划分所获得的“纯度提升”越大。正是如此，ID3 算法使用信息增益为准则来选择划分属性。

以相亲为例，假定有以下数据集（表 6.3），用以学习一棵能够预测是否要见面的决策树。显然$|y|=2$。在决策树学习开始时，根节点包含 D 中的所有样例，其中正例占

$p_1=\frac{8}{17}$，反例占 $p_2=\frac{9}{17}$。于是根据式（6.8）可以计算出根节点的信息熵为

$$\mathrm{Ent}(D)=-\sum_{k=1}^{|y|}p_k\log_2 p_k=-\left(\frac{8}{17}\log_2\frac{8}{17}+\frac{9}{17}\log_2\frac{9}{17}\right)=0.998$$

表 6.3　个人属性统计表

编号	身高/cm	长相	学历	性格	见面
1	180	平平无奇	硕士	沉稳	是
2	180	惊为天人	本科	沉稳	是
3	180	惊为天人	硕士	沉稳	是
4	180	平平无奇	本科	沉稳	是
5	180	不忍直视	硕士	沉稳	是
6	170	平平无奇	硕士	沉稳	是
7	170	惊为天人	硕士	活泼	是
8	170	惊为天人	硕士	沉稳	是
9	170	惊为天人	本科	活泼	否
10	160	平平无奇	高中	沉稳	否
11	160	不忍直视	高中	暴躁	否
12	180	不忍直视	硕士	暴躁	否
13	170	平平无奇	硕士	活泼	否
14	170	不忍直视	本科	活泼	否
15	170	惊为天人	硕士	沉稳	否
16	180	不忍直视	硕士	暴躁	否
17	180	平平无奇	本科	活泼	否

然后，计算出当前属性集合{身高，长相，学历，性格}中每个属性的信息增益。以长相为例，它有 3 个可能的取值：{不忍直视，平平无奇，惊为天人}。若使用该属性对 D 进行划分，则可得到 3 个子集，分别记为 D^1（长相=平平无奇）、D^2（长相=惊为天人）、D^3（长相=不忍直视），根据式（6.8）可以计算出用“长相”划分之后所获得的 3 个分支节点的信息熵为

$$\mathrm{Ent}(D^1)=-\left(\frac{3}{6}\log_2\frac{3}{6}+\frac{3}{6}\log_2\frac{3}{6}\right)=1.000$$

$$\mathrm{Ent}(D^2)=-\left(\frac{4}{6}\log_2\frac{4}{6}+\frac{2}{6}\log_2\frac{2}{6}\right)=0.918$$

$$\mathrm{Ent}(D^3)=-\left(\frac{1}{5}\log_2\frac{1}{5}+\frac{4}{5}\log_2\frac{4}{5}\right)=0.722$$

于是可以计算出属性“长相”的信息增益为

$$\begin{aligned}\mathrm{Gain}(D,\text{长相})&=\mathrm{Ent}(D)-\sum_{i=1}^{3}\frac{|D^i|}{|D|}\mathrm{Ent}(D^i)\\&=0.998-\left(\frac{6}{17}\times1.000+\frac{6}{17}\times0.918+\frac{5}{17}\times0.722\right)\\&=0.109\end{aligned}$$

考虑极端的情况，若一个属性能够把所有的样本都区分开来（如用身份证号码对人

进行区分），那么由每个取值和样本为一一对应的关系，此时计算出来的信息增益会非常高。由此可见，信息增益的缺点在于选择特征时，倾向于选择取值较多的属性，在C4.5 算法中对这一缺点进行了改进。

C4.5 算法不直接使用信息增益，而是使用“增益率”来选择最优划分属性。增益率的定义为

$$\text{Gain}_{\text{ratio}(D,a)} = \frac{\text{Gain}(D,a)}{\text{IV}(a)} \tag{6.10}$$

式中

$$\text{IV}(a) = -\sum_{i=1}^{V} \frac{|D^i|}{|D|} \log_2 \frac{|D^i|}{|D|} \tag{6.11}$$

称为属性 a 的固有值，属性 a 的可能取值数目越多，IV(a)的值通常会越大。

需要注意的是，与信息增益恰恰相反，增益率准则对可取值数目较小的属性有所偏好，因此，C4.5 算法并不是直接选择增益率最大的候选划分属性，而是使用了一个启发式：先从候选划分属性中找出信息增益高于平均水平的属性，再从中选择增益率最高的。

分类与回归树（classification and regression trees，CART）决策树使用基尼指数来选择划分属性。数据集 D 的纯度可用基尼值来度量。

$$\text{Gini}(D) = \sum_{k=1}^{|y|} \sum_{k' \neq k} p_k p_{k'} = 1 - \sum_{k=1}^{|y|} p_k^2 \tag{6.12}$$

直观来说，Gini(D)反映了从数据集 D 中随机抽取两个样本，其类别标记不一致的概率。因此 Gini(D)越小，则数据集 D 的纯度越高。

对属性 a 的基尼指数定义为

$$\text{Gini}_{\text{index}(D,a)} = \sum_{i=1}^{V} \frac{|D^i|}{|D|} \text{Gini}(D^i) \tag{6.13}$$

于是，在候选属性集合 A 中选择使划分后基尼指数最小的属性作为最优划分属性。

3．剪枝处理

剪枝是决策树学习算法优化“过拟合”问题的主要手段。在决策树学习中，为了尽可能正确地分类训练样本，节点划分过程将不断重复，有时会造成决策树分支过多，这时就可能因训练样本学得“太好”，以至于把训练集自身的一些特点当作所有数据都具有的一般性质而导致过拟合。因此，可通过主动去掉一些分支来降低过拟合的风险。

决策树剪枝的基本策略有预剪枝和后剪枝。预剪枝是指在决策树生成过程中，对每个节点在划分前先进行估计，若当前节点的划分不能带来决策树泛化能力的提升，则停止划分并将当前节点标记为叶节点；后剪枝则是先从训练集生成一棵完整的决策树，然后自底向上地对非叶节点进行考查，若将该节点对应的子树替换为叶节点能带来决策树泛化能力的提升，则将该子树替换为叶节点。

后剪枝决策树通常比预剪枝决策树保留了更多的分支。一般情形下，后剪枝决策树

的欠拟合风险很小，泛化性能往往优于预剪枝决策树。但是后剪枝过程是在生成完全决策树之后进行的，并且要自底向上地对树中所有非叶节点进行逐一考查，因此其训练时间开销比未剪枝决策树和预剪枝决策树都要大得多。

6.2.4　支持向量机

1．线性支持向量机

支持向量机（support vector machine，SVM）主要适用于一些线性可分的数据集，是一个典型的二分类算法。其基本模型定义为特征空间上的间隔最大的线性分类器，其学习策略便是间隔最大化，最终可转化为一个凸二次规划问题的求解。给定数据集 $D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}$，其中 $\boldsymbol{x}_i=(x_{i1},x_{i2},\cdots,x_{id})^{\mathrm{T}}, y_i\in\{-1,+1\}$，一个线性分类器的学习目标是基于训练集 D 找到一个超平面，将不同类别的样本分开，如图 6.7 所示。

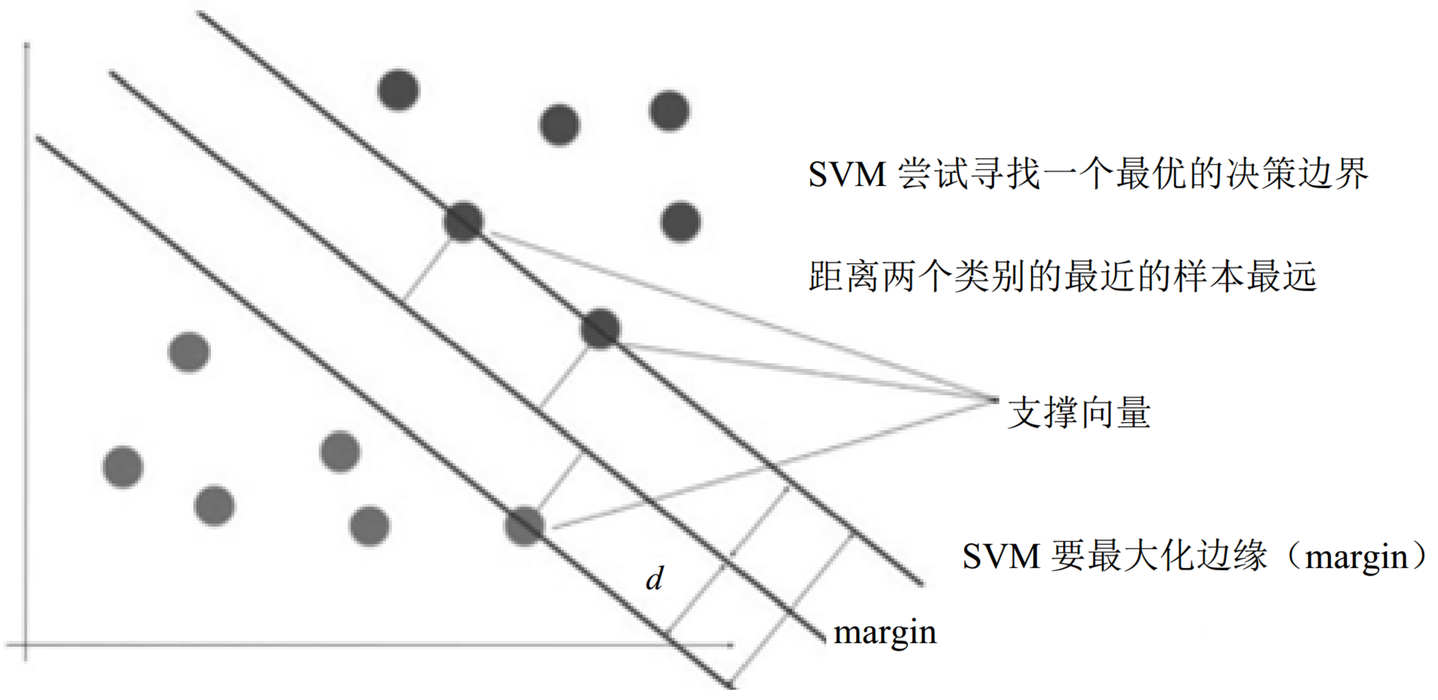


图 6.7　大间隔分类超平面

从直观上讲，这个超平面应该是最适合分开两类数据的直线。而判定“最适合”的标准就是这条直线离直线两边的数据的间隔最大。所以，需要寻找具有最大间隔的超平面。在样本空间中，划分超平面可以通过如下线性方程来描述：

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}=b=0 \tag{6.14}$$

式中，$\boldsymbol{w}=(w_1,w_2,\cdots,w_d)^{\mathrm{T}}$ 为法向量，决定了超平面的方向；b 为位移项，决定了超平面与原点的距离。若超平面 $(\boldsymbol{w},b)$ 可以将训练样本分类正确，那么对于任何 $(x_i,y_i)\in D$，都有

$$\begin{cases}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\geqslant+1, & y_i=+1\\ \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\leqslant-1, & y_i=-1\end{cases} \tag{6.15}$$

如图 6.8 所示，距离超平面最近的几个点使等号成立，这几个点称为支持向量（support vector），每个支持向量到超平面的距离可以写为

$$d=\frac{|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b|}{\|\boldsymbol{w}\|}=\frac{1}{\|\boldsymbol{w}\|}$$

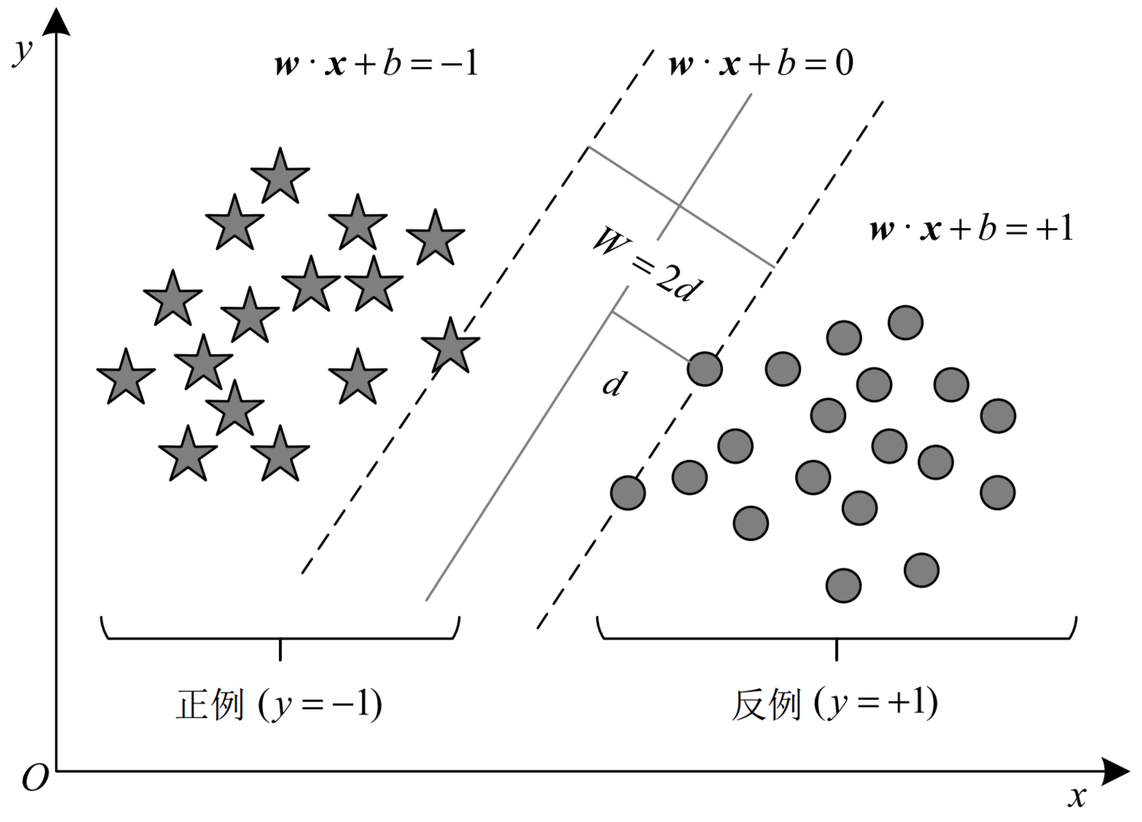


图 6.8　支持向量机与间隔

两个异类支持向量到超平面的距离之和称为间隔，即

$$\gamma = d + d = \frac{1}{\|\boldsymbol{w}\|} + \frac{1}{\|\boldsymbol{w}\|} = \frac{2}{\|\boldsymbol{w}\|}$$

支持向量机的优化目标找到有最大间隔的超平面，也就是找到约束参数（$\boldsymbol{w}$,b），使 γ 最大，即

$$\begin{aligned} &\max_{\boldsymbol{w},b} \frac{2}{\|\boldsymbol{w}\|} \\ &y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geqslant 1 \quad (i = 1, 2, \cdots, m) \end{aligned} \tag{6.16}$$

为了最大化 $\dfrac{2}{\|\boldsymbol{w}\|}$，等价于最小化 $\|\boldsymbol{w}\|^2$，于是优化目标可以重新改写为

$$\begin{aligned} &\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \\ &y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geqslant 1 \quad (i = 1, 2, \cdots, m) \end{aligned} \tag{6.17}$$

这就是支持向量机的基本型，可以看出优化目标下是一个凸二次规划问题，能够直接用现成的优化包求解。对于线性不可分的复杂情况，上述算法存在一定的局限性。由于近似线性可分问题中只有少数样本容易错分，换句话说，只有少数样本不满足约束条件 $y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \leqslant 1$。为此引入了松弛变量 δ，允许在一定程度上可以违反间隔约束，即约束条件可以变为

$$y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \leqslant 1 - \delta_i \quad (i = 1, 2, \cdots, m) \tag{6.18}$$

按照上述概念，这就变成一个软间隔支持向量机的问题。δ_i 变量的取值至关重要，若取值过大，$1-\delta_i$ 则趋近于负无穷大，这对约束样本没有意义；若取值过小，则起不到放松的作用。为此，在模型的基础上加一项惩罚项 $\sum_{i=1}^{m}\delta_i$，当 δ_i 充分大时，训练样本总会满足上述约束条件。在设计算法时为了避免较大的值对目标函数进行惩罚，原始模型修改如下：

$$\begin{aligned}&\min_{\boldsymbol{w},b}\frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{i=1}^{m}\delta_i\\&\begin{cases}y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b)\geqslant 1, & i=1,2,\cdots,m\\ \delta_i\geqslant 0, & i=1,2,\cdots,m\end{cases}\end{aligned}\tag{6.19}$$

式中，$C>0$ 是惩罚参数，用于调节 $\frac{1}{2}\|\boldsymbol{w}\|^2$ 与惩罚项 $\sum_{i=1}^{m}\delta_i$ 之间的权重。目标函数既要最小化 $\frac{1}{2}\|\boldsymbol{w}\|^2$，又要最小化 $\sum_{i=1}^{m}\delta_i$，此优化问题基于结构风险最小化原理，求解方法可通过求解其对偶问题获得。

2．非线性支持向量机

在实际应用中，数据多为线性不可分，但是非线性可分。采用非线性支持向量机算法能够使该问题获得很好的解决。非线性 SVM 首先采用非线性映射将原始空间中输入的数据映射至高维特征空间，再在特征空间构造最优分类超平面。这样就将求解原始空间中数据的非线性分类问题转化为求解高维空间中的线性分类问题，即在高维特征空间中寻找超平面。对于非线性划分问题，需要引入一个适当的变换函数 $\phi(\cdot)$，如图 6.9 所示，假设引进后从原始空间到高维特征空间的变换为 $\boldsymbol{x}\to\phi(\boldsymbol{x})$，那么在特征空间划分超平面所对应的模型可表示为

$$f(\boldsymbol{x})=\boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x})+b\tag{6.20}$$

式中，$\boldsymbol{w}$ 和 b 是模型的超参数，在高维空间的优化目标可以重新改写为

$$\begin{aligned}&\max_{\boldsymbol{w},b}\frac{1}{2}\|\boldsymbol{w}\|^2\\&y_i(\boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x})+b)\geqslant 1\quad(i=1,2,\cdots,m)\end{aligned}\tag{6.21}$$

其对偶问题为

$$\begin{aligned}&\max\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{i=1}^{m}\alpha_i\alpha_j y_i y_j\phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}_j)\\&\sum_{i=1}^{m}a_i y_i=0\quad(a_i\geqslant 0,\ i=1,2,\cdots,m)\end{aligned}\tag{6.22}$$

求解式（6.22）涉及计算 $\phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}_j)$，这是样本 x_i 与 x_j 映射到高维特征空间之间的内积，由于特征空间维度可能很高甚至是无穷维的，因此直接计算 $\phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}_j)$ 通常是困难的。为了避开这个障碍，可以设想这样一个函数

$$K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\langle\phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j)\rangle=\phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}_j)\tag{6.23}$$

即样本 $\boldsymbol{x}_i$ 与 $\boldsymbol{x}_j$ 在高维特征空间之间的内积等于它们在原始样本空间通过 $K(\cdot,\cdot)$ 计算的结果，有了这样的核函数就不再需要直接计算高维特征空间甚至是无穷维特征空间的内积。核函数有多种不同的形式，常用的核函数及其表达式如表 6.4 所示。在 SVM 模型建立的过程中，一旦确定了核函数的形式和参数，其隐含的非线性映射及其对应的高维

特征空间就被确定，因此，核函数形式和参数决定着 SVM 算法的分类性能。当选取的核函数适当时，高斯径向基函数（Gaussian radial basis function，RBF）能够近似于大多数核函数的映射性能。

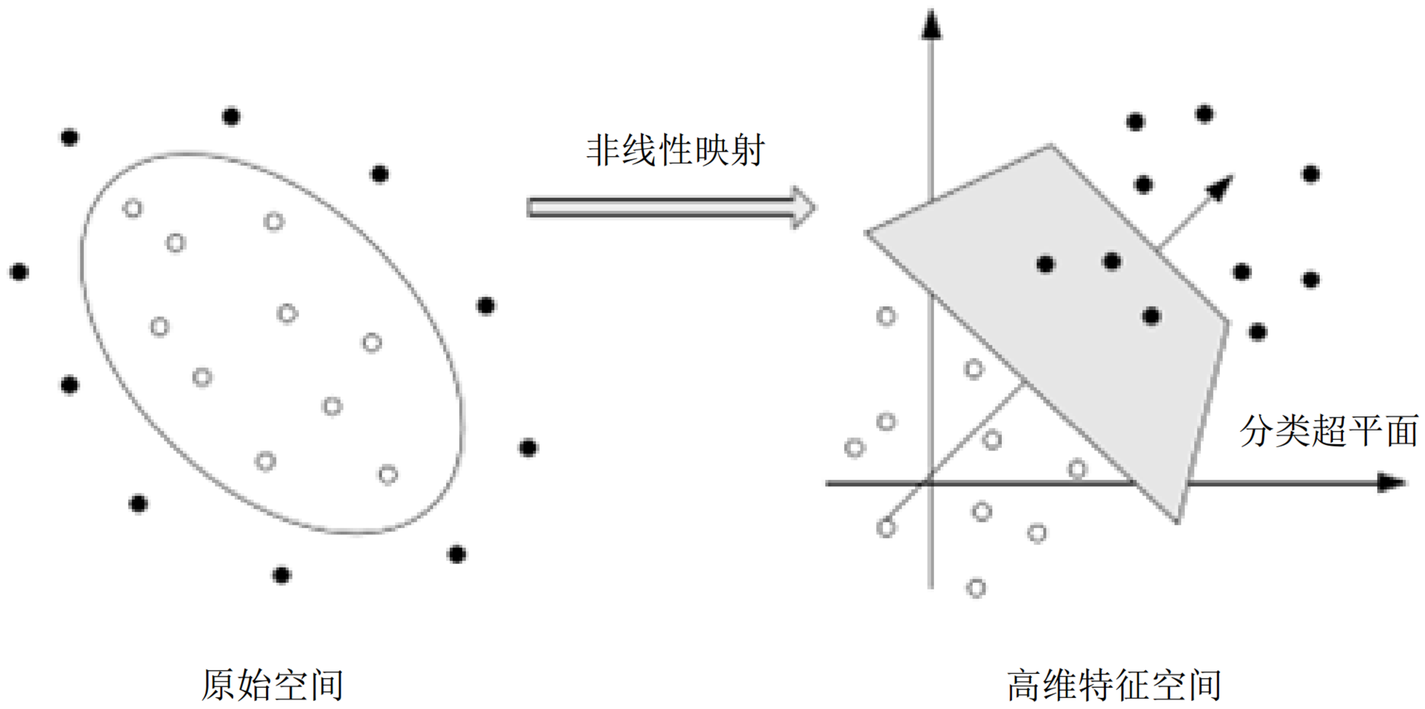


图 6.9　非线性映射示意图

表 6.4　常用核函数

核函数名称	表达式
线性核	$K(\boldsymbol{x}_i,\boldsymbol{x}_j)=<\boldsymbol{x}_i\cdot\boldsymbol{x}_j>$
多项式核	$K(\boldsymbol{x}_i,\boldsymbol{x}_j)=(\gamma<\boldsymbol{x}_i\cdot\boldsymbol{x}_j>+c)^d$
高斯径向基核	$K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\exp(-\Vert\boldsymbol{x}_i-\boldsymbol{x}_j\Vert^2/2\sigma^2)=\exp(-\gamma\Vert\boldsymbol{x}_i-\boldsymbol{x}_j\Vert^2)$
Sigmoid 核	$K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\tanh(\gamma<\boldsymbol{x}_i\cdot\boldsymbol{x}_j>+c)$

支持向量机是针对二分类问题提出的，它不可直接用于解决多类分类问题，但是实际应用中往往涉及多类分类问题，因此必须对其进行扩展以适应多分类的要求。对于多分类问题，有两种解决方案。第一种方案，只使用一个支持向量机实现多值输出。该方案是在标准支持向量机基础上，通过改变二分类支持向量机中的优化目标函数，使其满足多分类的要求。第二种方案，通过组合多个二分类支持向量机来求解，这种方案更常用。对 K 类问题，主要有四种组合方式：一对一法、一对其余法、决策导向无环图和决策树方法。

支持向量机的优点如下。

1）有严格的数学理论支持，可解释性强，不依靠统计方法，从而简化了通常的分类和回归问题。

2）能够找出对任务至关重要的关键样本（即支持向量）。

3）采用核技巧之后，可以处理非线性分类/回归任务。

4）最终决策函数只由少数的支持向量所确定，计算的复杂性取决于支持向量的数目，而不是样本空间的维数，这在某种意义上避免了“维数灾难”。

支持向量机的缺点如下。

1）当采用核技巧时，如果需要存储核矩阵，则空间复杂度为 $O(N^2)$。

2）模型预测时，预测时间与支持向量的个数成正比。当支持向量的数量较大时，预测计算复杂度较高。

因此，支持向量机目前只适合小批量样本的任务，无法适应百万甚至上亿样本的任务。

6.2.5　贝叶斯分类

贝叶斯决策理论（Bayesian decision theory）是概率框架下实施决策的基本方法。对分类任务来说，在所有相关概率都已知的理想情形下，贝叶斯决策理论考虑如何基于这些概率和误判损失来选择最优的类别标记。

1. 贝叶斯定理

贝叶斯定理是由英国数学家托马斯·贝叶斯提出的。

在现实生活中，我们很难知道事情的全貌。贝叶斯则从实际场景出发，提出了一个问题：如果事先不知道袋子里面黑球和白球的比例，那么摸出来的球的颜色，能够判断出袋子里面黑球与白球的比例吗？

正是这样的一个问题，影响了接下来近二百年的统计学理论。这是因为贝叶斯定理与其他统计学推断方法截然不同，它是建立在主观判断的基础上：在我们不了解所有客观事实的情况下，同样可以先估计一个值，然后根据实际结果不断进行修正。

用一个例子来体会贝叶斯定理的思想：假设有一种称为 Z 的疾病，它的发病率是万分之一，即 10 000 个人中会有 1 个人患病。现有一种测试可以检验 1 个人是否患病的准确率是 99.9%，它的误报率是 0.1%，那么现在的问题是，如果一个人被检测出来患有 Z 疾病，那么他实际上患有该病的可能性有多大？

你可能会说，既然查出患病的准确率是 99.9%，那是不是实际上患病的概率也是 99.9%呢？实际上不是的。因为在 10 000 个人中，还存在 0.1%的误查，也就是说有 10 个人没有患病但是检测结果呈阳性。当然，在 10 000 个人中也确实存在一个患病的人，他有 99.9%的概率检查出来。所以可以粗略地估算，检测出患病的这个人实际上是这 11 个人里面的一员，即其实际患病比率为 1/11≈9%。

上面的例子实际上涉及如下贝叶斯定理中的几个概念。

1）先验概率：通过经验来判断事情发生的概率。例如，Z 的发病率是万分之一就是先验概率。再比如，南方的梅雨季是 6～7 月，就是通过往年的气候总结出来的经验，这个时候下雨的概率比其他时间高出很多。

2）后验概率：后验概率是发生结果之后，推测原因的概率。例如，某人查出患有疾病 Z，那么患病的原因可能是 a、b 或 c。患病是因为原因 a 的概率就是后验概率。它属于条件概率的一种。

3）条件概率：事件 A 在另外一个事件 B 已经发生条件下的发生概率，表示为 $P(A|B)$，读作“在 B 发生的条件下 A 发生的概率”。例如，在原因 a 的条件下，患病的概率就是条件概率。条件概率计算公式如下：

$$P(A \mid B) = P(AB) / P(B) \tag{6.24}$$

式中，$P(AB)$表示联合概率，即事件 A 和事件 B 同时发生的概率；$P(B)$表示事件 B 发生的概率。

4）全概率公式：若事件 $A_1, A_2, \cdots, A_n$ 构成一个完备事件组且都有正概率，则对任意一个事件 B 都有

$$P(B)=\sum_{i=1}^{n}P(A_i)P(B\mid A_i)=\sum_{i=1}^{n}P(BA_i) \tag{6.25}$$

贝叶斯定理实际上就是求解后验概率。假设事件 A 表示“阳性”（检测出患有疾病 Z），用事件 B_1 表示“患有疾病 Z”，B_2 表示“没有患疾病 Z”。根据之前的例子，可以得到以下信息：患有疾病 Z 的情况下，检测结果呈阳性的概率为 $P(A|B_1)$=99.9%；没有患疾病 Z，但是检测结果呈阳性的概率为 $P(A\mid B_2)=0.1\%$；另外，患有疾病 Z 的概率为 $P(B_1)=0.01\%$，没有患疾病 Z 的概率为 $P(B_2)=99.99\%$。

根据条件概率公式，检测结果呈阳性，并且患疾病 Z 的概率为

$$P(AB_1)=P(B_1)P(A\mid B_1)=0.01\%\times 99.9\%=0.009\,99\%$$

同样可以求得，检测结果呈阳性，并且没有患疾病 Z 的概率为

$$P(AB_2)=P(B_2)P(A\mid B_2)=99.99\%\times 0.1\%=0.099\,99\%$$

根据全概率公式，检测结果呈阳性的概率为

$$\begin{aligned}P(A)&=P(B_1)P(A\mid B_1)+P(B_2)P(A\mid B_2)\\&=P(AB_1)+P(AB_2)=0.009\,99\%+0.099\,99\%\approx 0.11\%\end{aligned}$$

所以，检测结果呈阳性，且患疾病 Z 的概率为

$$P(B_1\mid A)=\frac{P(B_1)P(A\mid B_1)}{P(A)}=\frac{0.01\%\times 99.9\%}{0.11\%}\approx 9\%$$

检测结果呈阳性，但是没有患疾病 Z 的概率为

$$P(B_2\mid A)=\frac{P(B_2)P(A\mid B_2)}{P(A)}=\frac{99.99\%\times 0.1\%}{0.11\%}\approx 90.9\%$$

由此，可以总结出通用的贝叶斯公式为

$$P(B_i\mid A)=\frac{P(B_i)P(A\mid B_i)}{\sum_{i=1}^{n}P(B_i)P(A\mid B_i)} \tag{6.26}$$

2．朴素贝叶斯算法

在分类场景中，已知样本特征 $x=(x_1,x_2,\cdots,x_n)$，需要预测样本的真实类别 c 的概率 $P(c|x)$，根据贝叶斯公式，可以得到：$P(c\mid x)=\dfrac{P(c)P(x\mid c)}{P(x)}$。其中，对于某一条样本来说，由于特征 x 已经确定，对所有的类别来说 $P(x)$相同，而类别发生的概率 $P(c)$也可以通过已有样本统计而来，计算 $P(c|x)$的关键在于计算 $P(c|x)$。

计算 $P(c|x)$的主要困难在于：类条件概率 $P(c|x)$是属性上的联合概率，难以从有限的训练样本直接估计而得。为了避开这个障碍，朴素贝叶斯分类器采用了“属性条件独立性假设”：对已知类别，假设所有的属性相互独立。换言之，假设每个属性独立地对

分类结果发生影响。

基于属性条件独立假设，预测样本类别概率的公式可以改写为

$$P(c \mid x)=\frac{P(c)P(x \mid c)}{P(x)}=\frac{P(c)}{P(x)}\prod_{i=1}^{n}P(x_i \mid c) \tag{6.27}$$

式中，n 为属性数目；x_i 为 x 在第 i 个属性上的取值。

根据上述分析，朴素贝叶斯算法的分类流程如图 6.10 所示。

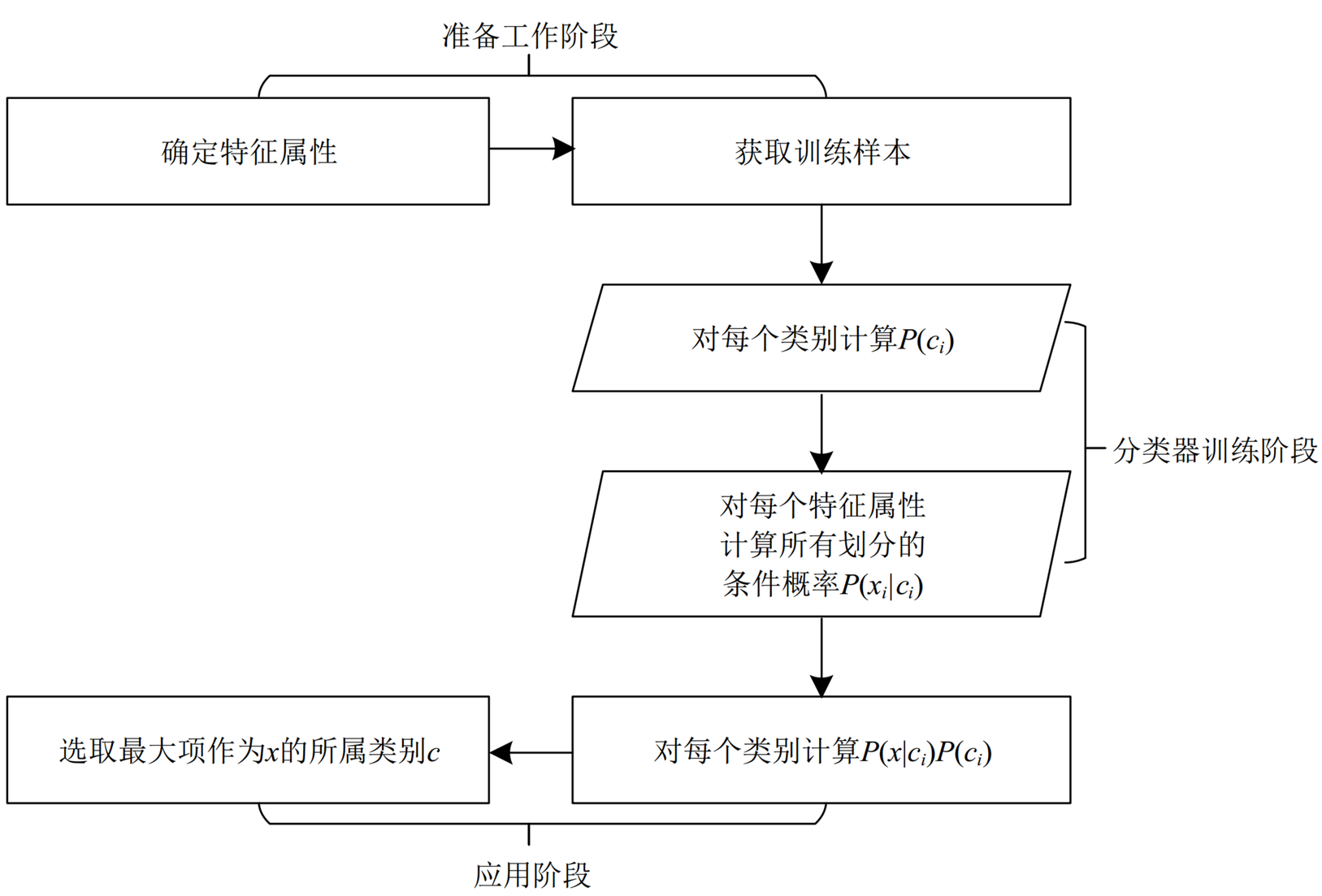


图 6.10　朴素贝叶斯算法的分类流程

可以看到，整个朴素贝叶斯分类分为三个阶段。

1）准备工作阶段。这个阶段的任务是为朴素贝叶斯分类做必要的准备，主要工作是根据具体情况确定特征属性，并对每个特征属性进行适当划分，然后由人工对一部分待分类样本进行分类，形成训练样本集合。这一阶段的输入是所有待分类数据，输出是特征属性和训练样本。这一阶段是整个朴素贝叶斯分类中唯一需要人工完成的阶段，其质量对整个过程有重要影响，分类器的质量很大程度上由特征属性、特征属性划分及训练样本质量决定。

2）分类器训练阶段。这个阶段的任务就是生成分类器，主要工作是计算每个类别在训练样本中的出现频率及每个特征属性划分对每个类别的条件概率估计，并将结果记录。其输入是特征属性和训练样本，输出是分类器。这一阶段是计算阶段，根据前面讨论的公式可以由程序自动计算完成。

3）应用阶段。这个阶段的任务是使用分类器对待分类项进行分类，其输入是分类器和待分类项，输出是待分类项与类别的映射关系。这一阶段也是计算性阶段，由程序完成。

在现实任务中，朴素贝叶斯分类器有多种使用方式。例如，若任务对预测速度要求较高，则对给定训练集，可将朴素贝叶斯分类器涉及的所有概率估值事先计算好并存储

起来，这样在进行预测时只需“查表”即可进行判别；若任务数据更替频繁，则可采用“懒惰学习”方式，事先不进行任何训练，待收到预测请求时，再根据当前数据集进行概率估值；若数据不断增加，则可在现有估值基础上，仅对新增样本的属性值所涉及的概率估值进行计数修正，即可实现增量学习。

6.2.6 神经网络

对神经网络（neural network）方面的研究很早就已经出现，今天，“神经网络”已是一个相当大的多学科交叉的学科领域。各相关学科对神经网络的定义多种多样，本书采用目前使用广泛的一种，即“神经网络是由具有适应性的简单单元组成的广泛并行互连的网络，它的组织能够模拟生物神经系统对真实世界物体所做出的交互反应”。在机器学习中谈论神经网络时指的“神经网络学习”，就是机器学习与神经网络这两个学科领域的交叉部分。

神经网络中基本的成分是神经元（neuron）模型，即上述定义中的“简单单元”。在生物神经网络中，每个神经元与其他神经元相连。如果某神经元“兴奋”时，就会向相连的神经元发送化学物质，从而改变这些神经元内的电位；如果某神经元的电位超过了一个“阈值”，那么该神经元就会被激活（即“兴奋”起来），并向其他神经元发送化学物质。

1．M-P 神经元模型

神经元接收来自 n 个其他神经元传递过来的输入信号，这些输入信号通过带权重的连接进行传递。神经元接收到的总输入值将与神经元的阈值进行比较，然后通过“激活函数”处理以产生神经元的输出。M-P 神经元模型如图 6.11 所示。

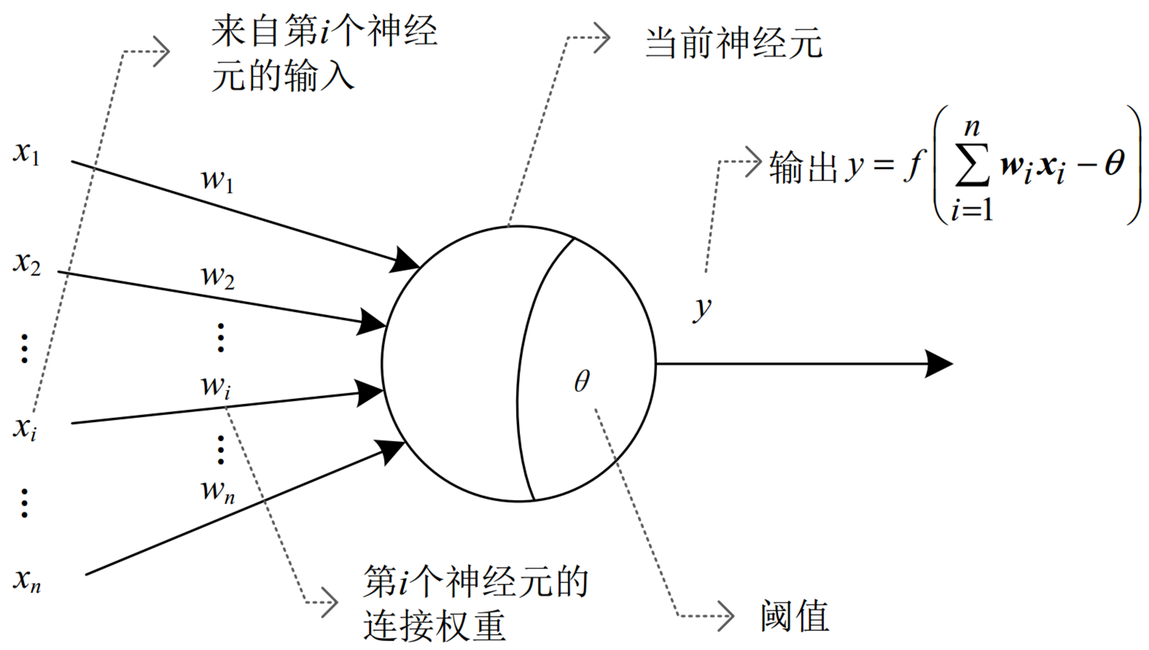
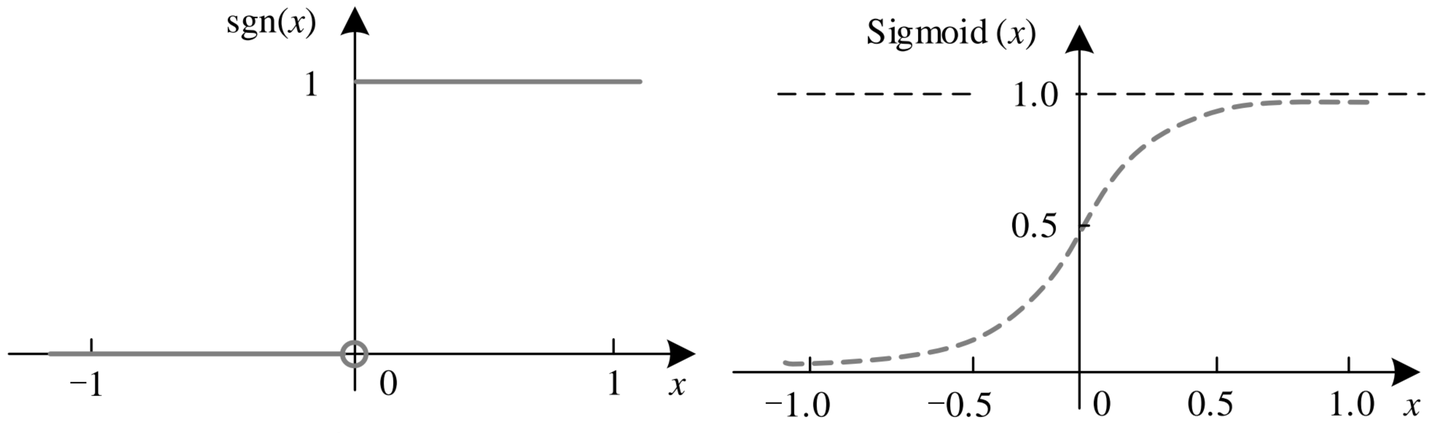


图 6.11 M-P 神经元模型

理想中的激活函数是图 6.12（a）所示的阶跃函数，它将输入值映射为输出值“0”或者“1”，显然“1”对应神经元兴奋，“0”对应神经元抑制。然而，阶跃函数具有不连续、不光滑等数学性质，因此实际常用 sigmoid()函数作为激活函数。典型的 sigmoid()函数如图 6.12（b）所示，它将可能在较大范围内变化的值挤压到（0,1）输出值范围内，因此有时也称为挤压函数。



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

（a）阶跃函数

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

（b）挤压函数

图 6.12　典型的神经元激活函数

将许多个这样的神经元按照一定的层次结构连接起来，就得到了神经网络。

事实上，从计算机科学的角度看，可以先不考虑神经网络是否真的模拟了生物神经网络，只需将一个神经网络视为包含了许多参数的数学模型，这个模型是由若干个函数（如 $y_i=f\left(\sum_i w_i x_i-\theta_j\right)$）相互（嵌套）代入而得。有效的神经网络学习算法大多以数学证明为支撑。

2．感知机与多层神经网络

感知机由两层神经元组成，如图 6.13 所示。输入层接收外界输入信号后传递给输出层，输出层是 M-P 神经元，也称阈值逻辑单元。

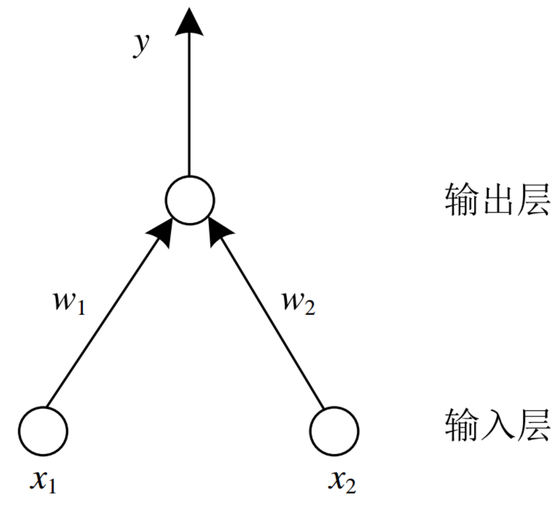


图 6.13　两层输入神经元的感知机网络结构示意图

感知机能容易实现逻辑与、或、非运算。例如，已知 $y=f\left(\sum_i \boldsymbol{w}_i \boldsymbol{x}_i-\theta\right)$，假定 f 是阶跃函数，则有以下几点。

1）与运算：令 $w_1=w_2=1$，$\theta=2$，则 $y=f(1\cdot x_1+1\cdot x_2-2)$，仅在 $x_1=x_2=1$ 时，$y=1$。

2）或运算：令 $w_1=w_2=1$，$\theta=0.5$，则 $y=f(1\cdot x_1+1\cdot x_2-0.5)$，当 $x_1=1$ 或 $x_2=1$ 时，$y=1$。

3）非运算：令 $w_1=-0.6$，$w_2=0$，$\theta=-0.5$，则 $y=f(-0.6\cdot x_1+0\cdot x_2+0.5)$，当 $x_1=1$

时，$y=0$；当 $x_1=0$ 时，$y=1$。

一般地，给定训练数据集，权重 $\boldsymbol{w}_i(i=1,2,\cdots,n)$ 以及阈值 θ 可通过学习得到，阈值 θ 可以看作一个固定输入为-1.0 的“哑节点”所对应的连接权重 $\boldsymbol{w}_{i+1}$。这样，权重和阈值的学习就可以统一为权重的学习。感知机学习规则非常简单，对训练样本（x, y），若当前感知机的输出为 $\hat{y}$，则感知机权重将按下式进行调整：

$$\boldsymbol{w}_i \leftarrow \boldsymbol{w}_i + \Delta w_i \tag{6.28}$$

$$\Delta w_i = \eta(y-\hat{y})\,\boldsymbol{x}_i \tag{6.29}$$

式中，$\eta \in (0,1)$ 称为学习率。从式（6.28）中可以看出：若感知机对训练样例（x, y）预测正确，即 $\hat{y}=y$，则感知机不发生变化；否则将根据错误的程度进行权重调整。

需要注意的是，感知机只有输出层神经元才能进行激活函数处理，即只拥有一层功能神经元，其学习能力非常有限。事实上，上述与、或、非问题都是线性可分的问题。可以证明：若两类模式是线性可分的，即存在一个线性超平面能将它们分开，则感知机的学习过程一定会收敛而求得适当的权重 $\boldsymbol{w}$；否则，感知机学习过程将会发送振荡，$\boldsymbol{w}$ 难以稳定下来，不能求得合适解。例如，感知机不能求解异或这样简单的非线性可分问题，如图 6.14 所示。

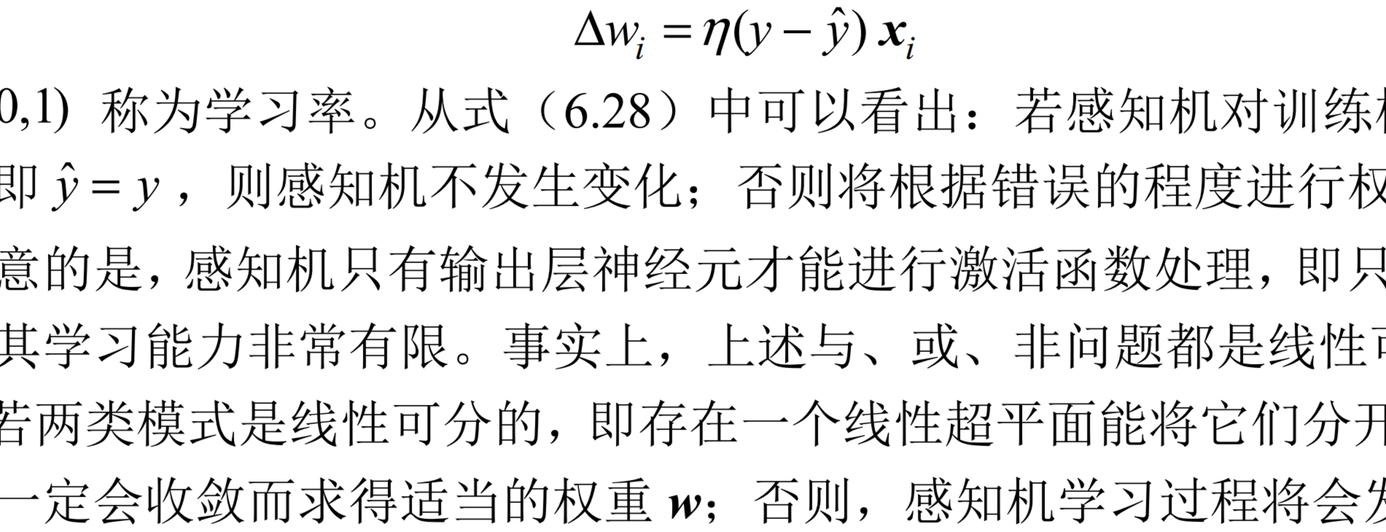
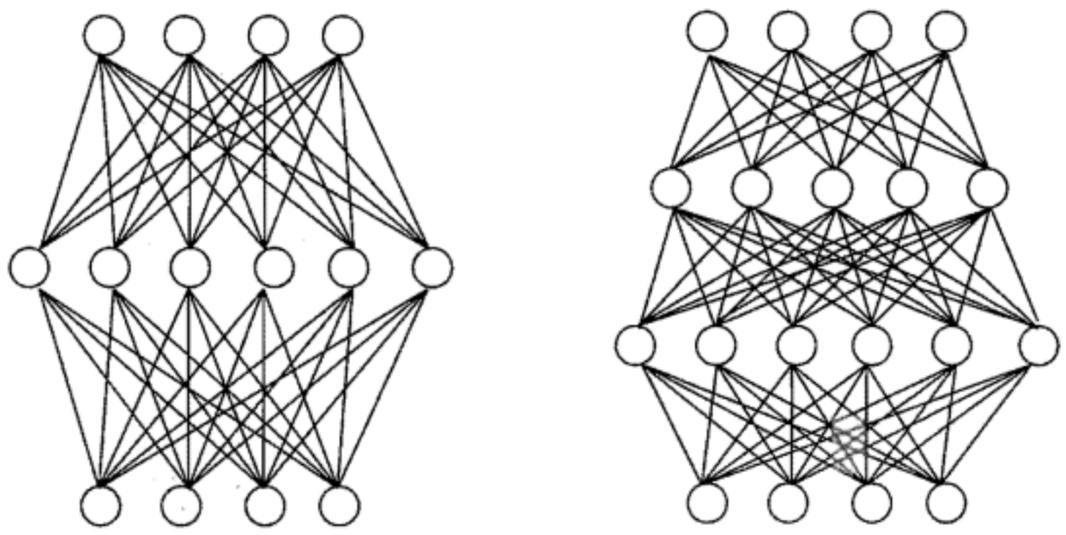


图 6.14　线性可分的与、或、非问题与非线性可分的异或问题

解决非线性可分问题，需要考虑使用多层功能神经元构成神经网络，常见的神经网络是如图 6.15 所示的层级结构，每层神经元与下一层神经元完全互连，神经元之间不存在同层连接，也不存在跨层连接。



（a）单隐含层前馈网络　　（b）双隐含层前馈网络

图 6.15　多层前馈神经网络结构示意图

输出层与输入层之间的神经元层称为隐含层，隐含层和输出层的神经元都是拥有激活函数的功能神经元。这样的神经网络结构通常被称为“多层前馈神经网络”。其中，输入层神经元接收外界输入，隐含层与输出层神经元对信号进行加工，最终结果由输出层神经元输出；换言之，输入层神经元仅接收输入，不进行函数处理，隐含层与输出层包含功能神经元。神经网络的学习过程，就是根据训练数据来调整神经元之间的“连接权”以及每个功能神经元的阈值；换言之，神经网络“学”到的东西，蕴含在连接权与阈值中。理论上，多层神经网络足以拟合任意的非线性函数。

3．典型的深度学习模型

理论上来说，参数越多的模型，其复杂度越高，“容量”越大，这意味着它能完成更复杂的学习任务。但是一般情形下，复杂模型的训练效率低，容易陷入过拟合，因此难以受到人们的青睐。随着云计算、大数据时代的到来，计算能力的大幅提升可缓解训练的低效性，训练数据的大幅增加则可降低过拟合风险，因此，以深度学习（deep learning）为代表的复杂模型开始受到人们的关注。

显然，对于神经网络模型，提高容量的一个简单办法就是增加隐含层数目。隐含层多了，相应的神经元连接权、阈值等参数就会多。模型复杂度也可以通过单纯增加隐含层神经元的数目来实现。单隐含层前馈神经网络已具有强大的学习能力，但是从增加模型复杂度的角度来看，增加隐含层的数目显然比增加隐含层神经元的数目更有效。因为增加隐含层数不仅增加了拥有激活函数的神经元数目，还增加了激活函数嵌套层数，所以典型的深度学习模型就是堆叠了多个隐含层的神经网络。

6.3　无监督学习

无监督学习（unsupervised learning）的数据集和监督学习的不同，没有任何标签，也就是没有“正确的输出结果”。在此过程中没有指导者，只有计算机自己学习。

6.3.1 聚类

聚类（clustering）是按照某个特定标准（如距离）把一个数据集分割成不同的类或簇，使同一个簇内的数据对象的相似性尽可能大，同时不在同一个簇中的数据对象的差异性也尽可能大，即聚类后同一类的数据尽可能聚集到一起，不同类数据尽量分离。

1．聚类和分类的区别

聚类：是指将相似的数据划分到一起，具体划分的时候并不关心这一类的标签，目标就是将相似的数据聚合到一起。聚类是一种无监督学习方法。

分类（classification）：是指将不同的数据划分开，其过程是通过训练数据集获得一个分类器，再通过分类器预测未知数据。分类是一种监督学习方法。

2．聚类的典型过程

一般来说，所有的聚类算法，其过程都可以抽象为以下几个步骤。

1）数据准备：样本清洗、特征标准化等。

2）特征选择：从最初的特征中选择有效的特征，并将其存储到向量中。

3）特征提取：通过对选择的特征进行转换形成新的突出特征。

4）聚类：基于某种距离函数进行相似度度量，将距离满足条件的样本归为同一个簇（cluster）。

5）聚类结果评估：分析聚类结果，如误差平方和（sum of squared error，SSE）等。

3．相似度度量

在聚类的典型过程中，相似度的度量是关键要素，对于数值型数据，通常使用以下集中相似度度量方法计算样本之间的距离或者相似度。

欧式距离：

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

曼哈顿距离：

$$d(x,y)=\sum_{i=1}^{n}\| x_i-y_i \|$$

切比雪夫距离：

$$d(x,y)=\max_{i=1,2,\cdots,n}^{n}\| x_i-y_i \|$$

闵可夫斯基距离：

$$d(x,y)=\left[\sum_{i=1}^{n}(x_i-y_i)^p\right]^{\frac{1}{p}}$$

闵可夫斯基距离就是 L_p 范数（$p\geqslant 1$），而曼哈顿距离、欧式距离、切比雪夫距离分

别对应 p=1,2,∞时的情形。

4．簇之间的相似度度量

除了需要衡量对象之间的距离外，有些聚类算法（如层次聚类）还需要衡量簇之间的距离，假设 C_i 和 C_j 为两个簇，常用以下几种方式定义 C_i 和 C_j 之间的距离。

1）Single-link 定义两个簇之间的距离为两个簇之间距离最近的两个点之间的距离。这种方法会在聚类的过程中产生链式效应，即有可能出现非常大的簇。

2）Complete-link 定义两个簇之间的距离为两个簇之间距离最远的两个点之间的距离，这种方法可以避免链式效应，对异常样本点（不符合数据集的整体分布的噪声点）非常敏感，容易产生不合理的聚类。

3）UPGMA 正好是 Single-link 和 Complete-link 方法的折中，它定义两个簇之间的距离为两个簇之间所有点距离的平均值。

4）WPGMA 方法计算的是两个簇之间两个对象之间距离的加权平均值，加权的目的是使两个簇对距离的计算的影响在同一层次上，而不受簇大小的影响，具体公式与采用的权重方案有关。

5．划分式聚类算法

聚类方法主要可以分为划分式聚类方法（partition-based method）、基于密度的聚类方法（density-based method）、层次化聚类方法（hierarchical method）等。

划分式聚类方法需要事先指定簇类的数目或者聚类中心，通过反复迭代，直至最后达到“簇内的点足够近，簇间的点足够远”的目标。经典的划分式聚类方法有 k 均值聚类（k-means）及其变体 k-means++、bi-k-means、kernel k-means 等。

（1）k-means 算法

经典的 k-means 算法的流程如下。

1）创建 k 个样本作为初始质心（也称作聚类中心，通常是随机选择）。

2）当任意一个样本的簇分配结果发生改变时：

① 对数据集中的每个样本点计算其到每个簇中心的距离；

② 将样本分配到距离其最近的簇；

③ 为所有样本点分配好簇之后，重新计算簇中心（簇中所有样本的均值）。

一般来说，经典 k-means 算法有以下几个特点。

1）需要提前确定 k 值。

2）对初始质心点敏感。

3）对异常数据敏感。

（2）k-means++算法

针对 k-means 中初始质心点选取的优化，提出了 k-means++算法。k-means++算法的流程和 k-means 类似，改变的地方只有初始质心的选取，该部分的算法流程如下。

1）随机选取一个样本点作为初始的聚类中心。

2）当聚类中心数量小于 k，计算每个样本点与当前已有聚类中心的最短距离，用

$D(x)$表示，这个值越大，表示被选取为下一个聚类中心的概率越大，最优使用轮盘法选取下一个聚类中心。

（3）bi-k-means 算法

度量聚类效果的一种指标是 SSE，它表示聚类后的簇离该簇的聚类中心的平方和，SSE 越小，表示聚类效果越好。bi-k-means 是针对 k-means 算法会陷入局部最优的缺陷进行的改进算法。该算法基于 SSE 最小化的原理，首先将所有的数据点视为一个簇，然后将该簇一分为二，之后选择其中一个簇继续进行划分，选择哪一个簇进行划分取决于对其划分是否能最大程度降低 SSE 的值。该算法的流程如下。

1）将所有样本点视为一个簇。

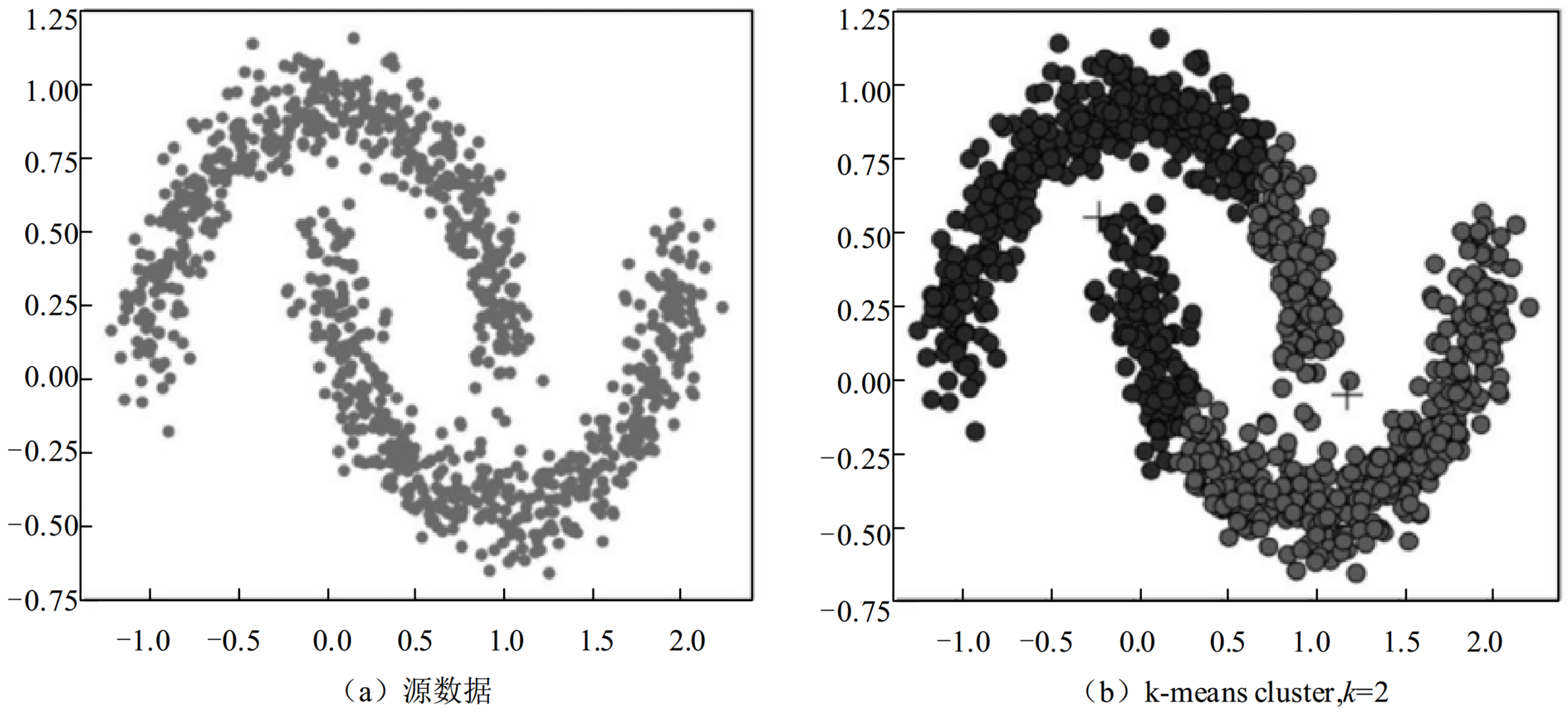
2）当簇的个数小于 k 时，对每一个簇，首先计算总误差，然后在给定的簇上进行 k-means（k=2），并计算将该簇一分为二之后的总误差；选取使误差最小的那个簇进行划分操作。

（4）kernel k-means 算法

常规的 k-means 聚类算法所解决的问题为线性可分的问题，而在工业领域，存在大部分数据集并不是线性可分的情况。在这部分数据集上，常规的 k-means 方法通常表现不佳。kernel k-means 的原理为将数据集通过核函数映射高维空间，从而找到更优的数据边界，达到更好的聚类效果。

6. 基于密度的聚类算法

k-means 算法对于凸性数据具有良好的效果，能够根据距离将数据分为球状类的簇，但是对于非凸形状的数据点，就无能为力了。k-means 算法对环形数据进行聚类的情况如图 6.16 所示。



（a）源数据　　（b）k-means cluster,k=2

图 6.16　k-means 算法在非凸数据集上表现不佳

从图中可以看到，k-means 产生了错误的结果。图中的数据集中，同类数据形状并不规范，但是同类数据之间紧密相连，具有连通性，这个时候就需要用到基于密度的聚

类方法，该方法需要定义两个参数 ε 和 M，分别表示密度的邻域半径和邻域密度阈值。具有噪声的基于密度的聚类方法（density-based spatial clustering of applications with noise，DBSCAN）就是其中的典型。

（1）DBSCAN 算法

考虑集合 $X=\{x^1,x^2,\cdots,x^n\}$，ε 表示定义密度的邻域半径，设聚类的邻域密度阈值为 M，有以下定义。

1）ε 邻域（ε-neighborhood）：$N_\varepsilon(x)=\{y\in X\,|\,d(x,y)<\varepsilon\}$。

2）密度（density）：x 的密度为 $p(x)=|N_\varepsilon(x)|$。

3）核心点（core point）：设 $x\in X$，若 $p(x)\geqslant M$，则称 x 为 X 的核心点，记 X 中所有核心点构成的集合为 X_c，记所有非核心点构成的集合为 X_{nc}。

4）边界点（boundary point）：若 $x\in X_{nc}$，且 $\exists y\in X$，满足 $y\in N_\varepsilon(x)\bigcap X_c$，即 x 的 ε 邻域中存在核心点，则称 x 为 X 的边界点，记 X 中所有的边界点构成的集合为 X_{bd}。

5）噪声点（noise point）：若 x 满足 $x\in X$，$x\notin X_c$，且 $x\notin X_{bd}$，则称 x 为噪声点。

如图 6.17 所示，设 $M=3$，则 A 为核心点，B、C 为边界点，而 N 为噪声点。

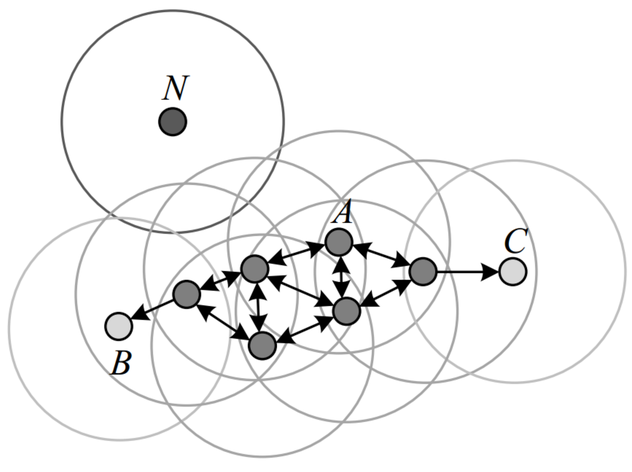


图 6.17　核心点、边界点和噪声点

（2）DBSCAN 算法的流程

1）标记所有对象为 unvisited。

2）当有标记对象时，进行如下操作。

① 随机选取一个 unvisited 对象 p。

② 标记 p 为 visited。

③ 如果 p 的邻域内至少有 M 个样本对象，则：

a．创建一个新的簇 C，并把 p 放入 C 中。

b．设 N 是 p 的 ε 邻域内的集合，对于 N 中的每个样本点 p'，如果 p' 是 unvisited，则标记 p' 为 visited；如果 p' 的 ε 邻域内至少有 M 个样本对象，则把这些样本点添加到 N；如果 p' 还不是任何簇的成员，则把 p' 添加到 C。

c．保存 C。

④ 否则标记 p 为噪声。

使用 DBSCAN 对之前提到过的数据进行聚类，结果如图 6.18 所示。

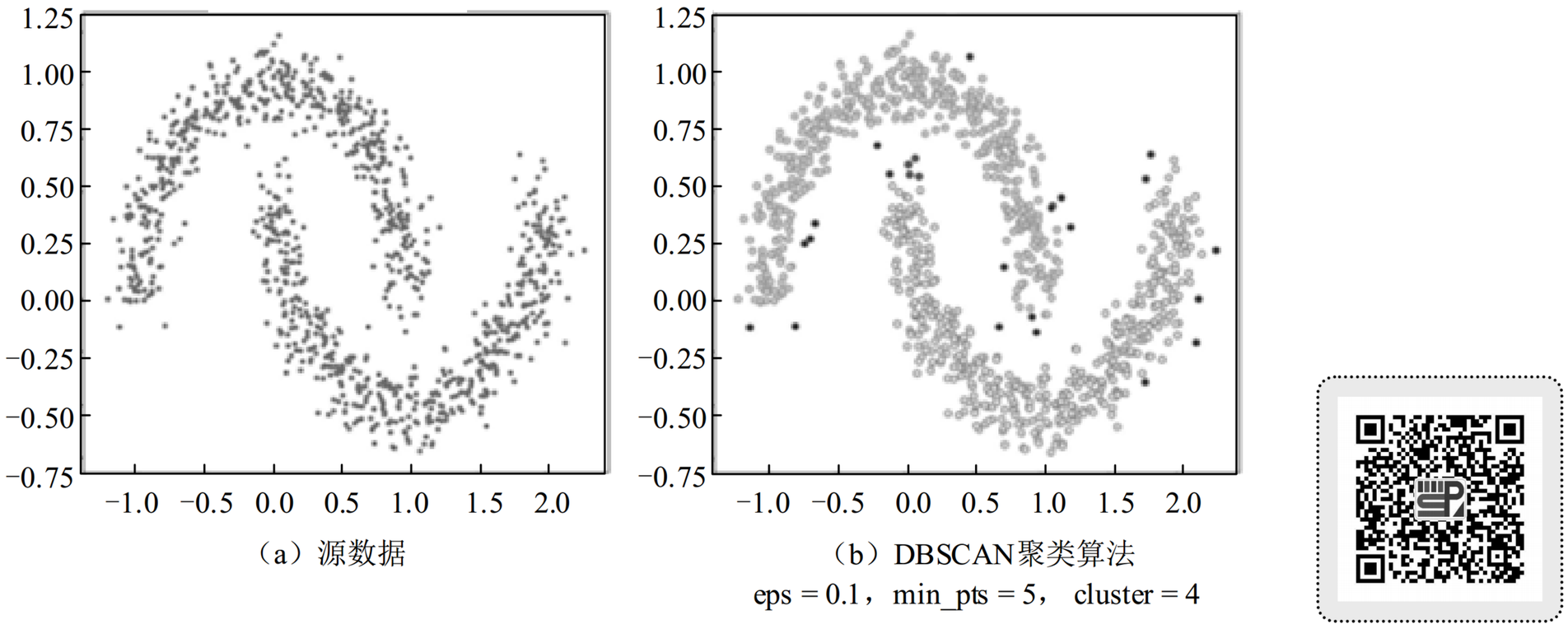


图 6.18　DBSCAN 在非凸数据集上的聚类效果

一般来说，DBSCAN 算法有以下几个特点。

1）需要提前确定 ε 和 M 参数。

2）不需要提前设置聚类的个数。

3）对初值选取敏感，对噪声不敏感。

4）对密度不均的数据聚合效果不好。

7．层次化聚类算法

前面介绍的几种算法确实可以在较小的复杂度内获取较好的结果，但是这几种算法存在一个链式效应的现象。例如，A 与 B 相似，B 与 C 相似，那么在聚类的时候便会将 A、B、C 聚合到一起；但是如果 A 与 C 不相似，就会造成聚类误差，严重的时候这个误差可以一直传递下去。为了降低链式效应，这时候层次聚类（hierarchical clustering）可以发挥作用，如图 6.19 所示。

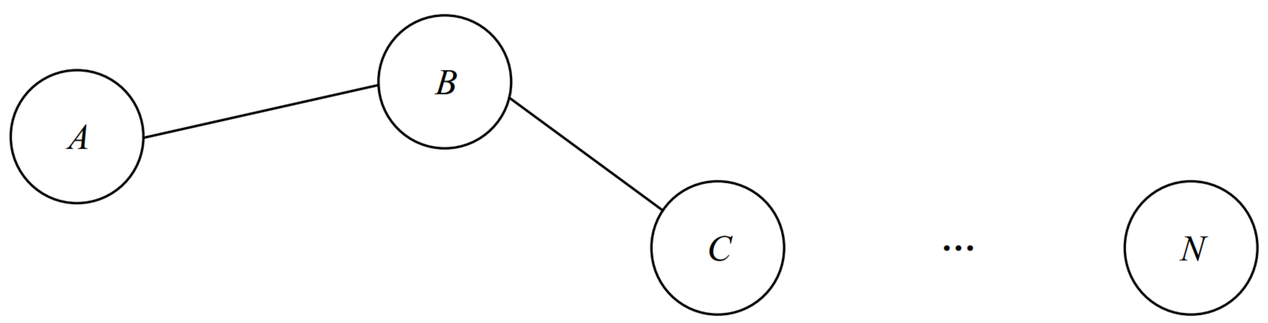


图 6.19　聚类算法的链式效应

（1）层次聚类算法

将数据集划分为一层一层的聚类，后面一层生成的聚类基于前面一层的结果。层次聚类算法一般分为两类，如图 6.20 所示。

1）凝聚（agglomerative）层次聚类：又称自底向上（bottom-up）的层次聚类。每一个对象最开始都是一个单独的簇，每次按照一定的规则将最相近的两个簇合并生成一个新的簇，如此往复，直至最终所有的对象都属于一个簇。这里主要关注此类算法。

2）分拆（divisive）层次聚类：又称自顶向下（top-down）的层次聚类。最开始所

有的对象均属于同一个簇，每次按照一定的规则将某个簇划分为多个簇，如此往复，直至每个对象均是一个单独的簇。

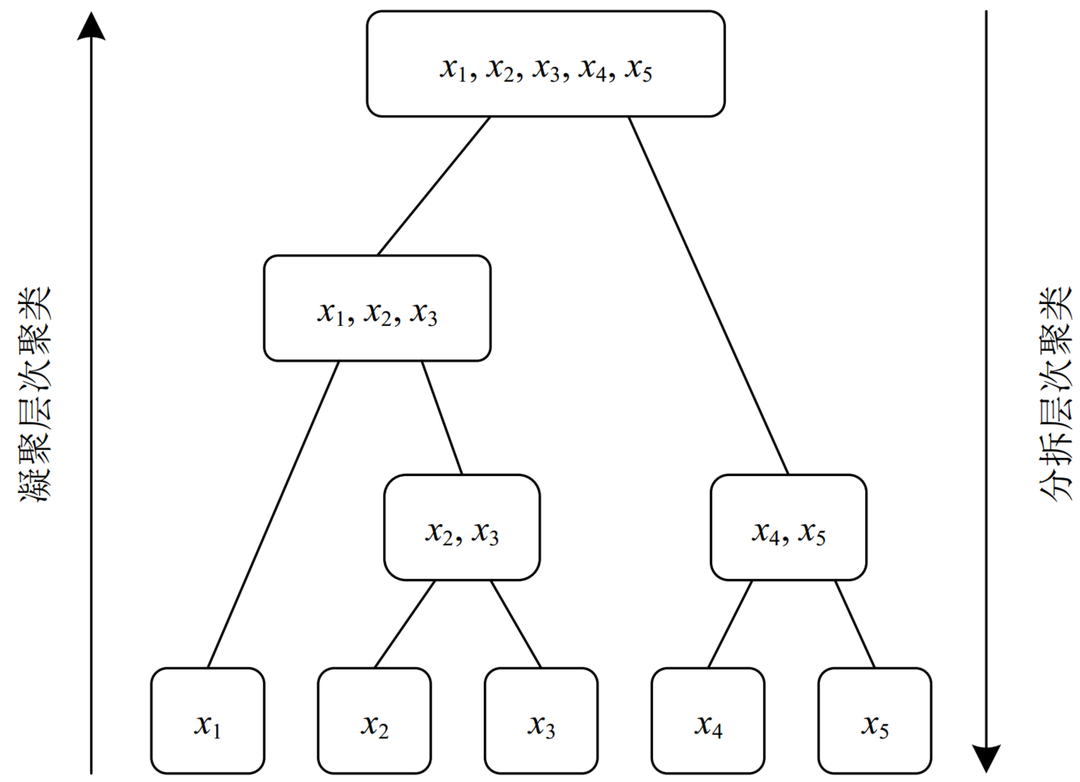


图 6.20　层次聚类的两种类型

需要指出的是，层次聚类算法是一种贪婪算法（greedy algorithm），因为其每一次合并或划分都是基于某种局部最优的选择。

（2）凝聚层次聚类

给定数据集 $X=\{x^1,x^2,\cdots,x^n\}$，凝聚层次聚类最简单的实现方法分为以下几步。

1）初始时每个样本为一个单独的簇，计算距离矩阵 $\boldsymbol{D}$，其中 D_{ij} 为样本 D_i 和 D_j 之间的距离。

2）遍历距离矩阵 $\boldsymbol{D}$，找出其中的最小距离（对角线上的除外），并由此得到拥有最小距离的两个簇的编号。将这两个簇合并为一个新的簇，并依据簇距离度量方法更新距离矩阵 $\boldsymbol{D}$（删除这两个簇对应的行和列，并把由新簇所计算出来的距离向量插入 $\boldsymbol{D}$ 中），存储本次合并的相关信息。

3）重复步骤 2）的过程，直至最终只剩下一个簇。

从以上的描述中可以看到，由于每次合并两个簇时，都需要遍历距离矩阵来搜索最小距离，需要进行合并的轮数也非常多，因此层次聚类算法的效率相对其他聚类算法来说是比较差的。

层次聚类算法中，常用的簇间距离度量方式有如下几种。①单链接（single-linkage）度量法。簇间距离等于两簇中对象之间的最小距离，若用相似度衡量，则是各簇中的任一对象与另一簇中任一对象的最大相似度。②全链接（complete-linkage）度量法。簇间距离等于两簇中对象之间的最大距离。③均链接（average-linkage）度量法。簇间距离等于两簇中对象之间的平均距离。④离差平方和链接（ward-linkage）度量法。簇间距离通过两簇中对象的最短、最长平均离差平方和计算得到。

图 6.21 分别使用了层次聚类中 4 个不同的簇间距离度量方法。可以看到，使用 single-linkage 确实会造成一定的链式效应，而使用 complete-linkage 则完全不会产生这种现象，

使用 average-linkage 和 ward-linkage 则介于两者之间。

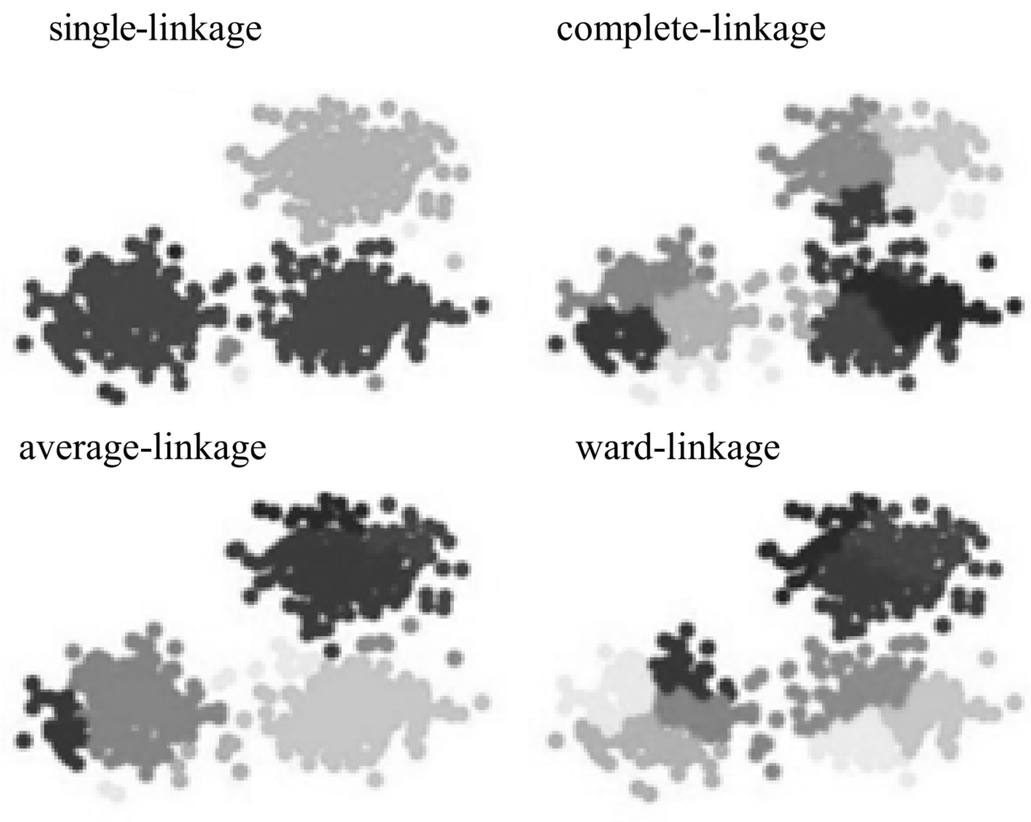


图 6.21　层次聚类中使用不同的簇间距离度量方法

每种聚类算法的优缺点如表 6.5 所示。

表 6.5　聚类算法优缺点对比表

算法	适合的数据类型	抗噪点性能	聚类形状	算法效率
k-means	混合型	较差	球形	高
k-means++	混合型	一般	球形	较高
bi-k-means	混合型	一般	球形	较高
DBSCAN	数值型	较好	任意形状	一般
agglomerative	混合型	较好	任意形状	较差

6.3.2　自编码器

自编码器（autoencoder，AE）是一种用于学习有效特征表示的无监督神经网络模型，它尝试从降维后的编码中生成尽可能接近其原始输入的表达。因此，其输出层的节点数与输入层的节点数相同，并且模型会朝着重构自己的输入目标优化。自编码器是一种典型的无监督学习网络，在网络的预训练过程中不需要标签样本，只在网络的微调阶段需要少量的标签样本进行网络结构微调。

1．自编码器模型

自编码器模型结构如图 6.22 所示，从输入数据得到隐含特征的过程叫作编码，由隐含特征重构输出数据的过程叫作解码。模型中的隐含特征也可被称为隐含层表征或潜在表征，一般来说其维数低于输入样本维数。

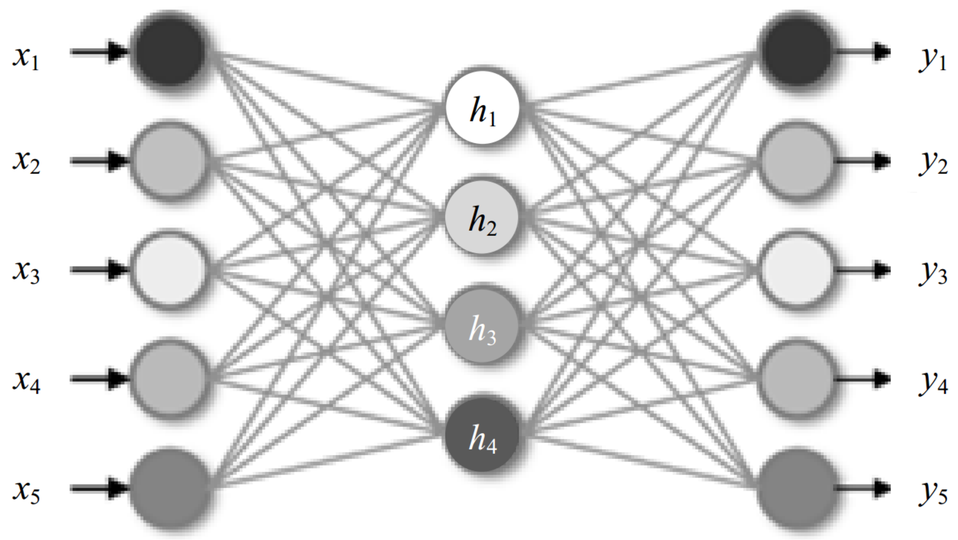


图 6.22　自编码器模型结构图

假设训练集中包含了 m 个训练样本 $\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_i,\cdots,\boldsymbol{x}_m$，每个样本 $\boldsymbol{x}_i \in \mathbf{R}^N$（$\mathbf{R}^N$ 为 N 维实数集），是一个 N 维的向量。对每个向量进行编码可以得到隐含的特征表示 $\boldsymbol{h}_i \in \mathbf{R}^K$：

$$\boldsymbol{h}_i = \boldsymbol{f}_\theta\ (\boldsymbol{x}_i) = \sigma(\boldsymbol{\omega}\boldsymbol{x}_i + b) \tag{6.30}$$

式中，$\theta(\boldsymbol{\omega},b)$ 是参数集合，σ 是隐含层的激活函数，激活函数通过给神经元加入非线性元素处理复杂的问题，可根据实际问题的需要自行选择合适的激活函数。编码之后再进行解码，从 K 维的隐含特征 $\boldsymbol{h}_i$ 重构出原始的 N 维输入特征 $\boldsymbol{x}_i \in \mathbf{R}^N$。

$$\boldsymbol{z}_i = g_\theta(\boldsymbol{h}\boldsymbol{v}) = g(\boldsymbol{\omega}'\boldsymbol{h}_i + b') \tag{6.31}$$

式中，$\boldsymbol{\omega}$和 $\boldsymbol{\omega}'$ 互为转置矩阵，自编码器的学习过程就是通过误差反向传播原理不断最小化误差函数来反复调整参数组 $\theta(\boldsymbol{\omega},b)$ 和 $\theta'(\boldsymbol{\omega}',b')$，其目标函数如下：

$$\begin{aligned}\theta^*,\theta'^* &= \arg\min_{\theta,\theta'} \frac{1}{m}\sum_{i=1}^{m} L(\boldsymbol{x}_i,\boldsymbol{z}_i) \\ &= \arg\min_{\theta,\theta'} \frac{1}{m}\sum_{i=1}^{m} L(\boldsymbol{x}_i, g_\theta(f_\theta(\boldsymbol{x}_i)))\end{aligned} \tag{6.32}$$

可以看出，自编码器是一个典型的三层前馈神经网络，它与传统的前馈神经网络有相似也有不同。

1）自编码器是前馈神经网络的一种，最开始主要用于数据的降维以及特征的抽取，随着技术的不断发展，现在也被用于生成模型中，可用来生成图片等。

2）前馈神经网络是有监督学习，其需要大量的标注数据；自编码器是无监督学习（自对比），数据不需要标注，因此较容易收集。

3）前馈神经网络在训练时主要关注的是输出层的数据以及错误率，而自编码器的应用可能更多地关注中间隐含层的结果。

自编码器具有三大特点。①自编码器是数据相关模型，这代表着自编码器只能对与训练数据模式相似的数据进行压缩重构，对于训练样本概率分布差异较大的测试样本，压缩性能较差，重构误差也会比较大。②自编码器是有损的压缩算法，自编码器的训练目标是最小化样本重构损失，但是无法将其降为 0，压缩得到的潜在表征和重构得到的样本相比于原始样本一定存在信息损失。③自编码器是无监督的学习算法，训练过程不会利用样本的标签信息，仅将指定类的数据输入模型便可以进行训练，最终得到特定的编码器和解码器。

2．深度自编码器

单个自编码器只能提取原始数据的浅层特征，为了得到更加抽象和鲁棒的特征表示，可以通过堆叠多个自编码器来搭建深度网络，逐层传递，从而在高层得到更具有表征能力和鲁棒性能的特征表示。其训练过程的基本结构如图 6.23 所示。

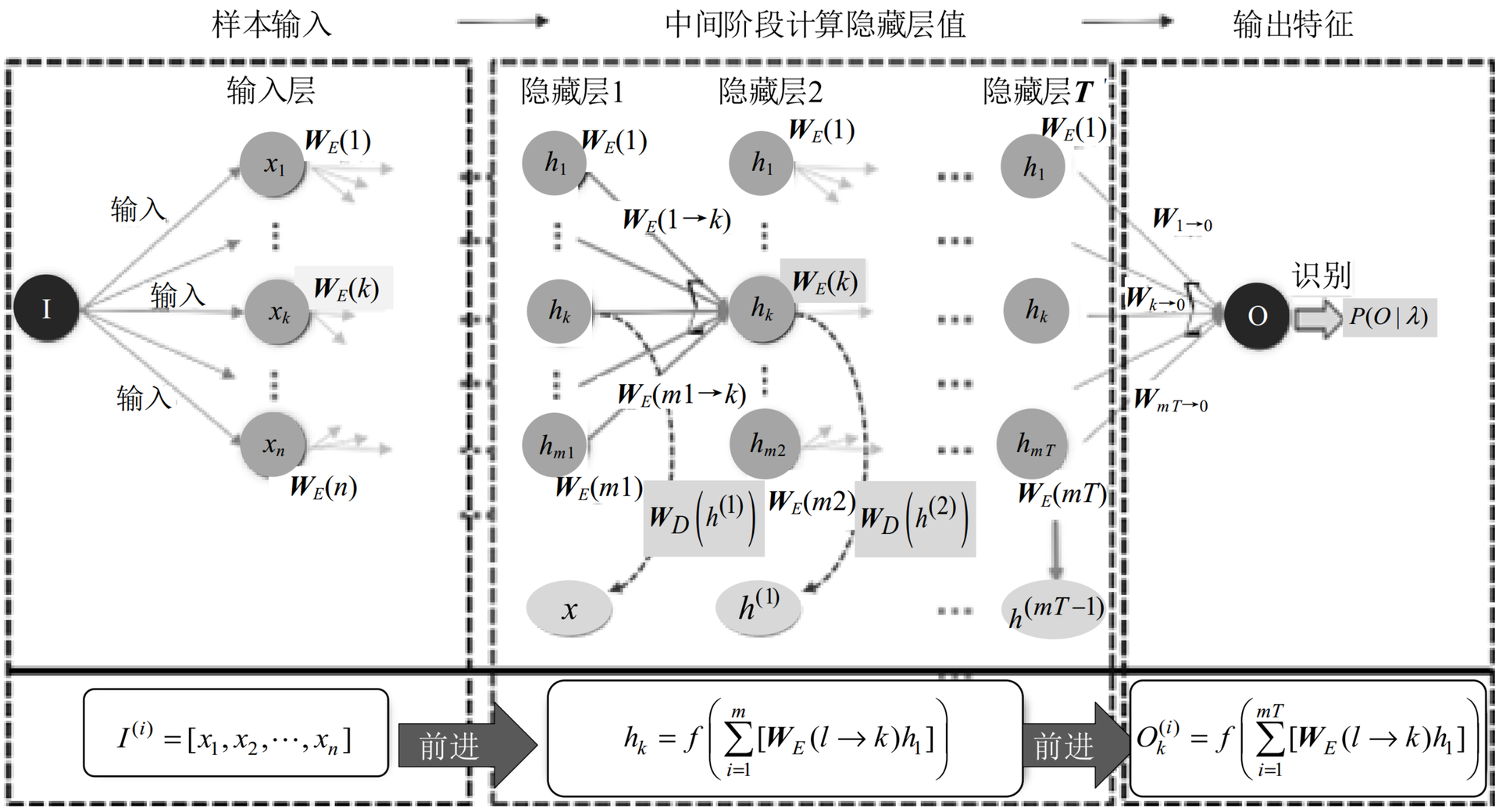


图 6.23　深度自编码器学习框架

通过堆叠自编码器自学习数据抽象特征的过程如下。首先，用无标签的数据训练第一个自编码器，其训练过程与传统神经网络的训练过程相同，随机初始化网络权重和偏差，并通过误差反向传播算法更新模型的参数，模型的输入和输出是相同的。其次，把第一个自编码器隐含层的输出作为第二个自编码器的输入，用无标签的数据使用相同的方式训练第二个自编码器。把第二个自编码器的隐含层输入拿出来训练第三个自编码器，依照这样的规则训练到预先设定的个数。最后，把每个训练好的自编码器的隐含层拿出来堆叠形成深度网络，这样通过一层一层逐层训练堆叠起来的深度网络，除输入层之外，其余各层都是自动提取的特征表示，且层数越高学习到的特征越抽象、鲁棒越有意义。通过这样的堆叠网络可以自动学习数据中隐含的抽象特征表示。

标准自编码器算法中，输入层、隐藏层（即特征层，两者等价）维度存在三种关系：输入维度>特征维度；输入维度=特征维度；输入维度<特征维度。当满足“>”关系时，自编码器强制学习输入数据的低维表示，用于捕捉数据中显著的特征。特别说明的是，当采用线性激活函数时，欠完备自编码器进行线性变换，等同于主成分分析（principal component analysis，PCA）、奇异值分解（singular value decomposition，SVD）等线性降维方法。

根据自编码器的限制条件不同，可以对隐藏层添加正则化，隐藏单元个数，使用卷积和池化、反卷积和反池化操作等，据此自编码器分为用于判别模型的正则自编码器和

收缩自编码器，以及用于生成模型的变分自编码器。当隐藏层的维度小于输入数据的维度时，欠完备的自编码器学习的数据特征是分布显著的，但是此类型的容量过大时，自编码器无法学习有用的信息。同样，隐藏层维度与输入层相等或大于输入的过完备网络类型时，就只是单纯地进行输入到输出的复制而无法挖掘到关于数据分布的有效信息。为解决此类问题，人们提出了正则自编码器。正则自编码器在不限制编码维度和模型容量的前提下使用损失函数，通过向损失函数增加对模型复杂度的惩罚项，使即使是非线性且过完备的模型仍然可以学习关于数据分布的特性（包括数据的稀疏表示以及对包含噪声或不完整数据的鲁棒性）。

6.4　弱监督学习

6.4.1　半监督学习

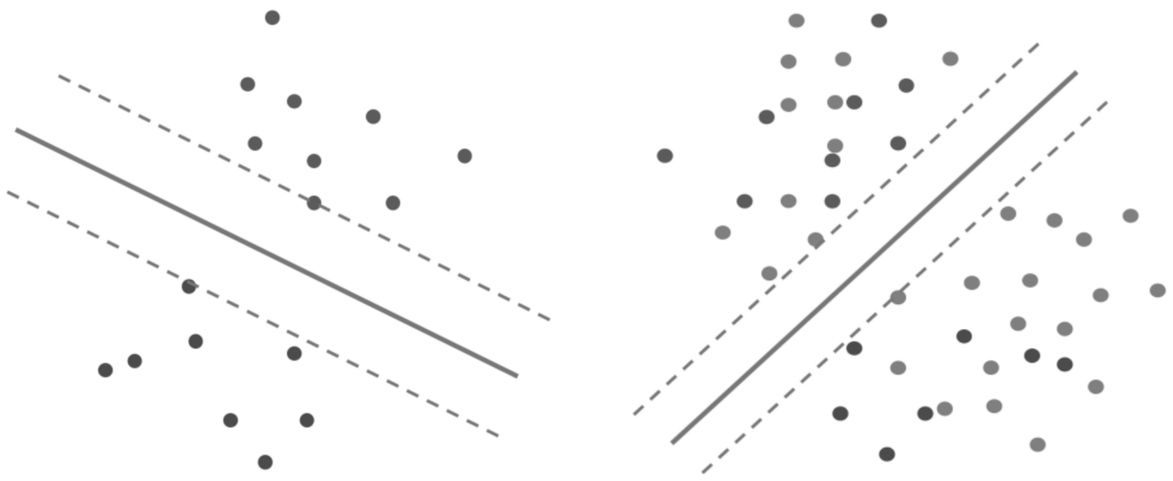
大数据背景下，现实世界中产生的数据通常容量巨大，结构复杂，而且其中绝大多数数据都是未被标记的。传统的机器学习技术一般只利用有标记样本集或者只利用无标记样本集进行学习，而在实际问题中大多数情况是有标记样本与无标记样本并存。例如，对于图像分类任务来说，除了已有的标记数据，因特网上会时刻产生新的但没有标记的同类数据。为了更好地利用这些数据，半监督学习技术应运而生。近年来，半监督学习技术成为机器学习领域一个被广泛研究的热点。

在传统的监督学习中，机器学习通过对大量有标记的（labeled）实例进行学习，从而建立用于预测未来示例标记的模型。这里的“标记（label）”是指示例所对应的输出。在分类问题中标记就是示例的类别，而在回归问题中标记就是示例所对应的实值输出。同时，随着数据收集和存储技术的飞速发展，收集大量未标记的示例已相当容易，而获取大量有标记的示例则相对困难，因为获得这些标记可能需要耗费大量的人力和物力。

例如，在计算机辅助医学图像分析中，可以从医院获得大量的医学图像作为训练实例，但是如果要求医学专家把这些图像中的病灶都标识出来，则往往是不现实的。再如，因特网上绝大多数的图片是没有标记的，通常图片的标记需要人工来进行，花费巨大是一方面，另一方面，在有些场景中对数据打标签是一件几乎办不到的事情，如对未知基因的功能进行标记。事实上，无标记数据大量存在，且其获取成本相对廉价，如何利用无标记数据来辅助提高学习的预测性能是一个非常有价值、可研究的问题，这也是半监督学习的宗旨。例如，在进行 web 网页推荐时，需要用户标记出哪些网页是他感兴趣的，很少会有用户愿意花大量的时间来提供标记，因此有标记的网页示例比较少。这样因特网上就存在着无数的网页，它们都可作为未标记示例来使用。一方面，如果只使用少量的有标记示例，那么利用它们所训练出的学习系统往往很难具有强的泛化能力；另一方面，如果仅使用少量“昂贵的”有标记示例而不利用大量“廉价的”未标记示例，则是对数据资源极大的浪费。因此，在有标记示例较少时，如何利用大量的未标记示例来提高学习性能，已成为当前机器学习研究中备受关注的问题之一。

传统的机器学习主要有监督学习和无监督学习。其中典型的监督学习的经典场景有分类和回归两大类。对于给定的训练集 $\{x_i, y_i\}_{i-1}^{N}$，其目的都是学习一种从样本 x 到标签 y 的映射关系。半监督学习在过去十几年中受到广泛关注，它致力于利用大量无标记样本来辅助相对少量人工标记的样本，从而希望能够训练出比只使用人工标记样本更准确的分类器。在传统的监督学习中，人工标记的样本是非常重要的，被标记的训练样本的数量和质量往往决定着最后得到的分类器的好坏。然而，在具体的应用中，想获得大量的标记样本，其花费通常巨大。例如，在因特网上图片的标记、基因数据的标记等，通常是需要投入巨大的人力，甚至即使投入巨大也没法直接获取样本的标签。相对于稀少的人工标记样本，无标记的样本大量存在，获取大量无标记样本的成本也相对廉价很多。正是在这种现实背景下，半监督学习研究得到发展。在生物信息学领域，基因的数量比较大。例如，人类基因组拥有近 30 000 个基因，尽管研究者们希望根据功能对这些基因分门别类，但是，由于研究者们对其中大部分基因功能的研究还存在空白，因此只能把少量基因归到具体的功能类别上。在这种情况下，如何利用仅有的少量的有标注功能的基因来推测其他基因的功能是一个非常典型和重要的应用场景。

半监督学习的概念从产生到发展的过程中，演化出了多种具体的形式。例如，早期的自学习（self-learning）概念，弗拉基米尔·瓦普尼克（Vladimir Vapnik）提出了传导支持向量机，基于生成模型的半监督学习，以及后来的基于图的半监督学习、多视角学习、多核学习和协同训练等。然而，所有半监督学习算法都强调无标记数据在学习中的重要作用。不同于在传统的监督学习模式中，只有标记的样本参与训练决策函数，而未标记数据只能作为被预测的对象，在半监督学习算法中，除了用标记样本得到决策函数，还需要用无标记样本的分布信息来调整决策面。如图 6.24（a）所示的一个二类分类的例子里，红色和蓝色代表样本的两种标签，直接用监督学习的方法（线性 SVM）训练可以得到一个“最佳决策面”（图 6.24 中蓝色的实线）；但是如果在大量的无标记样本[图 6.24（b）中灰色的点]的情况下，最佳决策面需要重新调整。如果忽略无标记样本的作用，而直接利用图 6.24（a）所示的决策面来对未来的样本进行预测，误分率必然很高。



（a）利用标记数据训练得到的线性分类函数　　（b）利用标记和无标记数据共同训练得到的线性分类函数

图 6.24　线性分类函数

在半监督学习场景，有 n 个样本，其中 l 个标记数据 $L:\{x_i, y_i\}_{i-1}^{l}$，$n-l$ 个未标记数

据 $U:\{x_i,y_i\}_{i-l+1}^{n-l}$。

1．基于生成模型的半监督学习

生成半监督模型是古老的半监督学习方法。它假设在联合分布 $p(x,y)=p(y)p(x\mid y)$，其中 $p(x\mid y)$ 是一个可确定的混合分布，如高斯混合模型。利用大量无标记的数据，可以确定混合模型的分布。理想状态下，对于混合分布的一个构件是只需要一个样本标签的信息就可以确定。生成式模型分类器将未标记实例属于每个类别的概率视为一组缺失参数，然后采用最大期望（expectionmaximization，EM）算法进行标记估计和模型参数估计。此类算法可以看成在少量有标记示例周围进行聚类，是早期直接采用聚类假设的做法。

在概率模型里，高斯过程分类器是一种重要的监督学习分类器。具体地，它们从传统的半监督生成模型出发，对未标记样本提出一种“空类（null category）”的概念，因此可以从概率的角度去解释判别模型里的边界（margin）概念。

2．自学习

自学习是出现在 1960～1970 年的一种较早的半监督学习类型。算法的基本原理如下。

1）在已标记数据集 L 上训练分类器 f。

2）用 f 来对未标记数据集 $\{X_i\}_{i-l+1}^{N}$ 做预测。

3）根据上一步预测的结果，从 U 中筛选出预测可信度最高的子集 U'。

4）$L\leftarrow L+U'$。

5）$U\leftarrow U+U'$。

6）重复上述过程直至分类结束。

自我学习典型的例子是应用在自然语言处理（natural language processing，NLP）上，以及计算机视觉中的目标检测（target detection）。自我学习在这些领域的应用都表现出了很高的性能。

3．传导支持向量机

半监督支持向量机是半监督学习的一个重要分支。通常，对于判别模型，试图学习到 $p(x\mid y)$。然而，对于半监督学习，非标记样本的分布信息 $p(x)$ 如果不能通过共享的参数和 $p(x\mid y)$ 产生关联，那么未标记样本将对分类器的训练无任何贡献。

传导支持向量机又称直推支持向量机（transductive support vector machine，TSVM），它的出现弥补了这一缺陷。传导支持向量机利用非标记数据的分布信息，将传导支持向量机的最大边界思想运用到半监督学习场景中，认为最优的分类决策面应当根据未标记数据的密度进行调整，如图 6.25 所示。不同之处是：标准的支持向量机仅利用标记的数据来寻找可以使数据最大化边界的决策面；而在传导支持向量机中，算法的目标是预测未标记数据的标签，因而需要利用标记数据和未标记数据来共同完成寻找最优决策面的优化过程。寻找到的决策面需要对未标记数据的标签进行预测有最小的泛化误差。从

这个意义上讲，半监督学习场景中的未标记数据将引导决策面的优化。

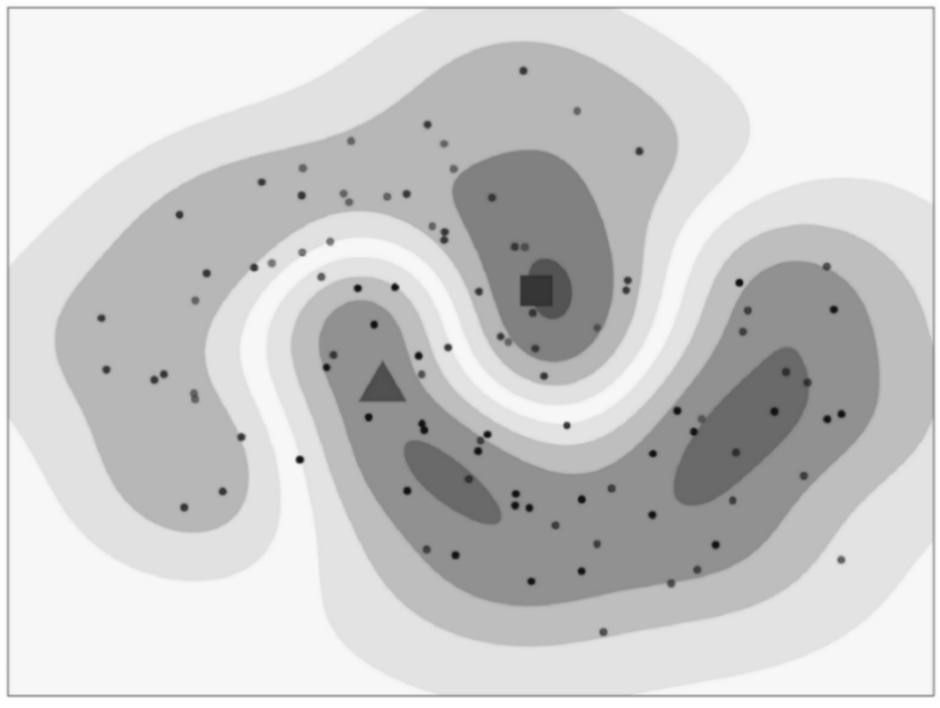


图 6.25　直推支持向量机示意图

注：在 Two-moon 的数据集中：红色为正类样本，其中有一个被标记样本点（图中方形）；蓝色为负类样本，有一个被标记样本点（图中三角形）。训练得到的决策面穿过图中低密度的样本分布的区域。

4．基于图的半监督学习

基于图的半监督学习的基本思想是构造关于所有的样本(包括标记和未标记)的图，其中样本本身被设想为图的节点;根据预先定义的某种度量得到的样本之间的相似或者相近关系是图中的边，即如果在这种度量下两个样本 X_i 和 X_j 被认为是相似的，那么 X_i、X_j 之间存在着一条边，反之则不存在边。在构建完成后，就可以利用各种衍生的算法来推测未标记数据的标签。这些算法共同的一个关于图的基本假设是，节点的标签在整个图上具有平滑度（smoothness），即相似的样本倾向于有相同的标签。

构建图是基于图的半监督学习的一个重要步骤，不同方法构建的图可能在标签预测时表现出完全不同的性能。构建的图可以是全连接的图（无向图），也可以是稀疏的图，但是大多数是稀疏图；其中图的邻接矩阵可以是只含有 **0** 和 **1**（**0** 代表两个样本不相似，**1** 代表相似）二值的矩阵，也可以是带权邻接矩阵。迄今为止，图的构建仍是一个未被广泛研究的问题，已有的构建方法大致可以分为以下两类。

1）利用领域（背景）知识来构造图。当对某个领域数据有很深的理解后，可以由领域专家根据其对数据的理解来直接构造图。

2）利用邻居关系构造图。在不清楚领域知识的情况下，可以利用 KNN 等无监督学习的思想来发现邻居样本，从而构造邻居图。典型的例子是：对于样本 X_i，利用 KNN 算法，如果 X_i 是 X_j 的 k 个最近的邻居之一，则判定 X_i 和 X_j 之间存在一条边；否则不存在边。

流形假设对基于图的算法，主要有两种大规模应用思路。第一，让图的拉普拉斯（Laplacian）矩阵稀疏；第二，求近似解，通常做法是把无标记数据表示成其他样本的线性组合，从而减少变量规模。目前来看，基于图的方法主要应用在图像标注上。随着互联网技术和应用的发展，产生了大量的图片数据，其中绝大多数是未被标注的。考虑人工标注图片的巨大成本，使用机器学习的办法是一种选择。在有少量被标注的图片的

前提下，运用半监督学习的方法，可以充分利用未标注数据的分布来辅助学习过程。在图片标注的大规模应用中，主要问题还是当算法被大规模应用时，节点太多，对于 Laplacian 矩阵来说，其复杂度是节点数的多项式（polynomial）函数。因而，大规模基于图的半监督学习应用的一个思想是尽量减少图的规模。已有试图大规模的方法都是在尽量减少 Laplacian 矩阵的规模，于是产生了很多主干网络（backbone）思想来构建图。

5．多视角学习和协同训练

传统的机器学习算法，如支持向量机、Logistic 回归等，并不区分特征的来源。这些方法通常将多个来源的异构特征当作一个整体来处理。多视角学习和协同训练（co-training）则相反，对于多个来源的特征区分对待，并在多个来源上共同训练模型。

协同训练的原理和多视角学习类似，是多视角学习的一种特例。协同训练假设特征可以分为两个独立的子特征 X_A 和 X_B，根据区分的子特征 X_A 和 X_B，可以将数据分成标记数据和未标记数据，分成 L_A、L_B 和 U_A、U_B 四个不同的集合。协同训练的基本思想如下。

1）在 L_A、L_B 上独立训练各自的分类器 f_A、f_B。

2）利用得到的 f_A、f_B 分类器对未标记数据集合 U_A、U_B 分别进行独立的预测。

3）根据上一步预测的结果，从 U_A、U_B 中筛选出预测可信度最高的子集 U'_A、U'_B。

4）$L_A \leftarrow L_A + U'_A, L_B \leftarrow L_B + U'_B$。

5）$U_A \leftarrow U_A - U'_A, U_B \leftarrow U_B - U'_B$。

6）用新的 L_A、L_B 来校正 f_A、f_B。

7）重复上述过程直至分类结束。

6．多视角学习

多视角学习（multi-view learning）是一种针对多种子特征的学习方法，这种方法兴起于现实世界中数据的多样性。例如，在生物信息学（bioinformatics）领域，同一个问题可能需要多种数据的支持，可能有微阵列基因表达（microarray gene expression）数据、切片的磁共振成像（magnetic resonance imaging，MRI）数据、血液的实验室检测（lab test）数据等。再如，在通过手机进行人的移动行为感知领域，通常需要读取手机中多种传感器的数据（加速度、陀螺仪、指南针等）。

多视角学习又可以细分为多核学习（multiple kernel learning）和子空间学习（subspace learning）。其中，多核学习核矩阵的基本思想是通过搜索一种核矩阵的线性或者非线性的组合来学习多视角的数据，从而最大化一般形式的学习性能。子空间学习是指通过数据学习，寻找一种多个视角的数据共同的隐藏空间表达。反过来，所有观测到的多视角数据都能由隐藏空间表达来生成。例如，对于有多种患者的 ICU 期间的生理指标检测数据，不同指标之间具体数值和生理含义都有很大区别，但是都能联系到一个共同的子空间来描述患者的病况。

6.4.2　迁移学习

机器学习作为人工智能的一个分支，在现实应用过程中起到的作用日益突出，计算

机通过现有的、标签清楚的数据进行训练和学习，挖掘有价值的、实用的知识和信息，并通过一定的技术手段应用在新的实际问题上。但是，传统的机器学习要求学习任务要在给定充分的、标签清楚的训练数据的基础上学习和训练得到决策模型，利用这个决策模型对新数据进行测试与预测。这就要求学习任务之间的分布相同，即领域之间的特征应该相同。但是，在当下这样一个互联网高速发展的时代，每天都会在不同领域、不同地区涌现出大量新的信息和咨询，从传统的新闻，到网页、图片、视频，再到微博、微信等。若还像传统的机器学习那样需要对每个领域的信息都标定清楚，再借助其进行训练，无疑耗费的人力与物力是巨大的，甚至可能由于样本的数量大、维数高、更新快，使得传统机器学习失效。如果完全丢弃这些已知数据又是浪费的。显然，如何合理地利用这些数据就是传统机器学习所面临的主要问题。此时，借助领域和任务之间的知识进行迁移学习（transfer learning，TL）就显得非常必要。在人类不断进化的过程中具备了这样迁移学习的能力。例如，人们学习骑摩托车之前，若已经学会骑自行车，那么骑自行车的本领对骑摩托车显然是有帮助的。人类在先前知识的基础上，可以在不同领域和问题之间进行迁移学习，这种迁移学习的能力可以帮助计算机提高学习的效率。因此，人们也企图借助一定的技术和手段让计算机在挖掘和整理这些已有数据信息的基础上，可以总结出新的知识结构，提高目标领域的学习能力。

从一般意义上讲，迁移学习放松了传统机器学习要求领域之间的分布相同的要求，只要源领域和目标领域之间具有一定的相似关系，迁移学习就可以借助从源领域数据和特征中提取的知识，帮助目标领域的学习。进一步说，迁移学习的目标就是使用源领域学到的知识来帮助目标领域的学习。在实际应用场景中，在目标领域的数据标签较少甚至没有标签的情况下，若有大量的已知相关领域的数据或者知识，合理地利用和借鉴这些数据就是迁移学习的动机，也就回答了何时迁移的问题。

从心理学角度来分析人类认知过程，大致可以分成三个层次。第一层次就是学习者通过记忆、背诵、记录等基本学习方式，让有关的信息、知识、文化、技艺等得到传承。第二层次就是学习者已经进步到“学会如何学习”的程度，学习者在必要的知识和技艺基础上，学会归纳、总结和挖掘知识的内在规律，将实际问题中新的情况纳入已有的思维和行为模式。第三层次是学习者能够达到“融合学习”的程度，学习者可以将他人的、离散的、有意义的生活经验融合成自己特有的行为准则，以适应新形势下不断变化的社会问题。“融合学习”使学习者较前两个阶段具有了更高的适应能力，并成为生活中必不可少的技能核心。从上述理论来分析，迁移学习属于人类学习的第二层次。具体来说，迁移学习从大量标签清楚的源领域数据、数据特征或者某种知识中挑选对目标任务有帮助作用的信息，从而有效提高目标领域的学习能力。迁移学习效果主要取决于领域之间的相似程度，若领域之间在学习内容和方法上无共同之处，迁移学习就不会发生；若领域之间有共同因素，就会产生迁移，且共同因素越多，迁移作用就越大。

例如，在生物课堂上向小学生讲解鸳鸯的相关知识时，若给出图 6.26 所示的几幅图片，孩子们会联想到以前见过的鸭子和天鹅，因为其外形与鸳鸯十分相似，有所不同的只是颜色和大小。将孩子之前了解的关于鸭子的生活习性迁移到鸳鸯的一些生活习性中，显然会更加有助于孩子们对鸳鸯的了解。此外，朱鹭的相关信息对认识鸳鸯的作用

不大；而北极熊和鸳鸯无论是外形还是生活环境都相差很远，因此，关于北极熊的知识对了解鸳鸯没有什么帮助。

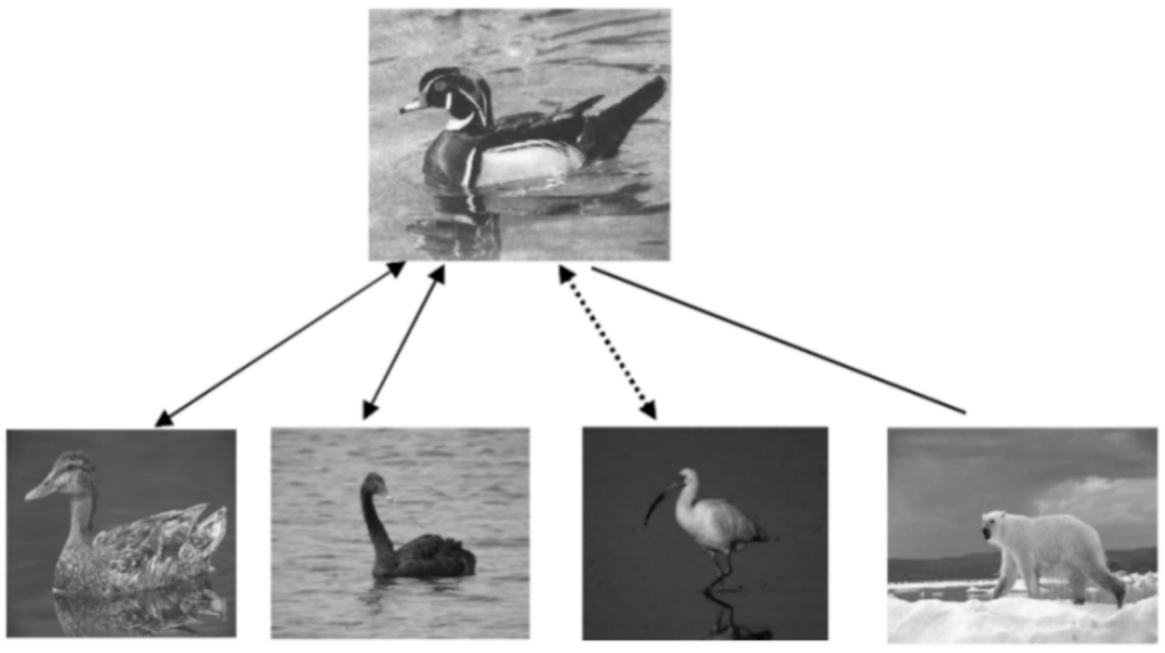
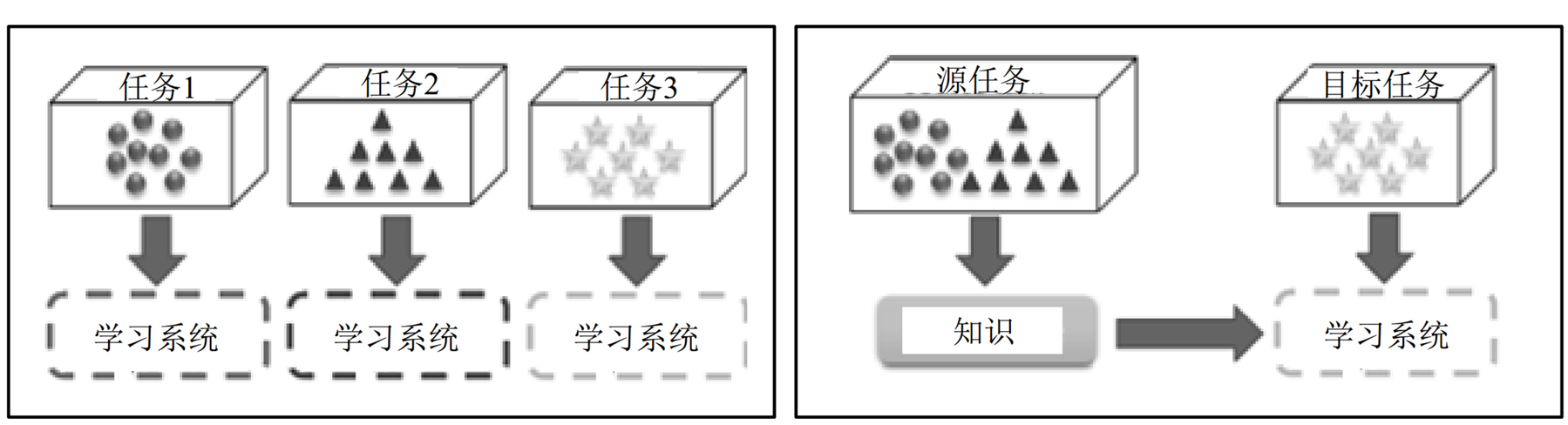


图 6.26　迁移学习示意过程

迁移学习比较有代表性的定义是在 2005 年神经信息处理系统（neural information processing systems，NIPS）的国际会议上提出来的，英文描述如下：Transfer learning emphasizes the transfer of knowledge across domains, tasks, and distributions that are similar but not the same。其中文翻译简单描述为：迁移学习强调的是在不同但是相似的领域、任务和分布之间进行知识的迁移。从本质上讲，迁移学习就是将已有领域的信息和知识运用于不同但相关领域中的一种新的机器学习方法。迁移学习不要求相似领域服从相同的概率分布，其目标是将源领域里面已有的知识和信息，通过一定的技术手段迁移到新领域中，进而解决目标领域标签样本数据较少甚至没有标签的学习问题。迁移学习历经数十年的发展，该领域算法的名称几经转变，与其他领域又多有交叉，定义和界限都比较模糊，造成了迁移学习与其他机器学习算法的界限模糊。如图 6.27 所示，迁移学习的本质是迁移已有数据、任务的知识去完成新的任务，强调迁移知识而非从头开始的训练过程。



（a）传统机器学习方法　　（b）迁移学习方法

图 6.27　迁移学习与传统机器学习的差异

迁移学习的过程可以描述为在源数据域中通过完成源任务获取知识，并将知识迁移到目标数据域中协助完成目标任务。通常，源领域与目标领域的数据分布或任务并不一致。当源领域与目标领域的数据分布、任务均一致时，该学习问题将退化为传统机器学习问题。根据源领域与目标领域数据分布、任务的不同情况，迁移学习分为归纳迁移、

转导迁移和非监督迁移三类，如表 6.6 所示。

表 6.6　不同类别迁移学习的差异

迁移学习类别	相关领域	源领域	目标领域	源领域与目标领域数据分布	源领域与目标领域任务
归纳迁移学习	多任务学习、自学习		少量标注		不一致
转导迁移学习	领域自适应、样本偏置选择	标注		不一致	一致
非监督迁移学习			无标注	不一致	不一致

归纳迁移学习的主要特点是目标任务与源任务不一致。如图 6.28 所示，当源领域有大量的标注数据时，归纳迁移学习类似于多任务学习。两者的主要区别在于：多任务学习联合学习源和目标任务，致力于两者的联合最优；而归纳迁移学习侧重于模型在目标任务上的性能。当源领域仅有未标注数据（没有标注数据）时，归纳迁移学习类似于自学习。

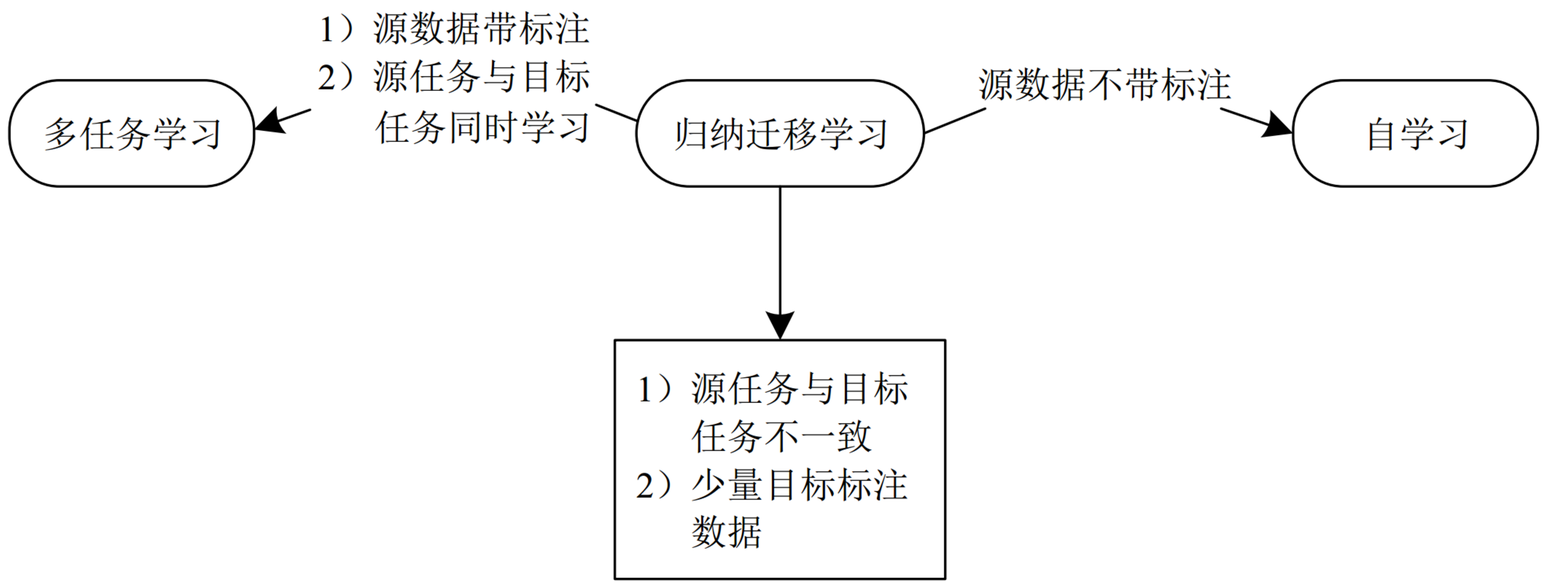


图 6.28　归纳迁移学习及与其他研究领域的差异

转导迁移学习的特点是源任务与目标任务一致，但是两者数据领域不同。传统的转导学习要求目标数据在训练阶段全部可见，学习得到的模型难以保证对未见数据的有效性。随着研究的进展，限制逐步放开，但是仍要求少量未标注的目标数据在测试阶段可见，用以估计目标数据的边际分布。如图 6.29 所示，当源空间与目标空间的特征空间一致而边际分布不同时，转导迁移学习类似于样本选择偏置问题、协方差转变问题。

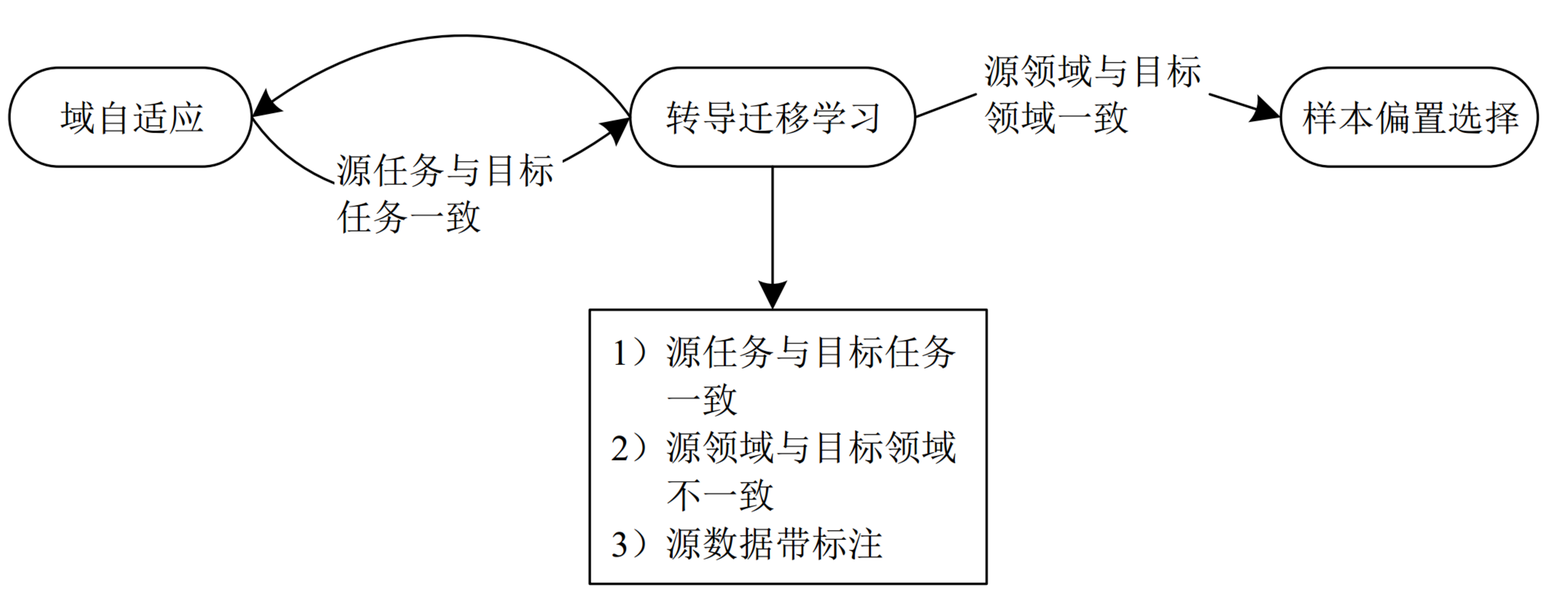


图 6.29　转导迁移学习及与其他研究领域的差异

如图 6.30 所示，无监督迁移学习的特点为目标数据与源数据均未标注，其更关注在目标领域解决非监督问题，如聚类、降维、密度估计等。其主要思想是存在一个共同的特征空间，在该空间下，源领域与目标领域距离最小，因此可以通过源数据辅助目标数据进行无监督的聚类、降维等任务。

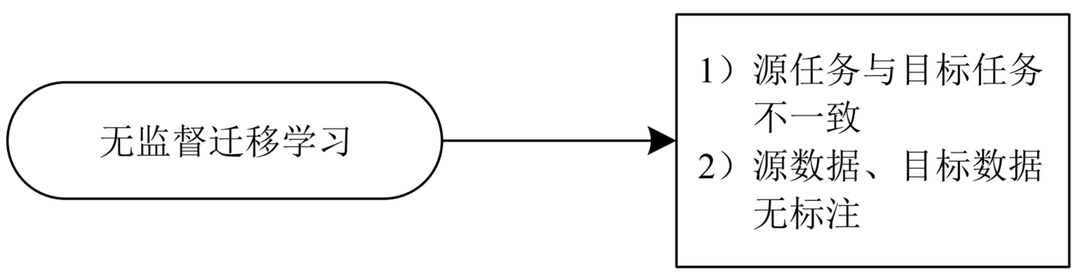


图 6.30　无监督迁移学习图示

从迁移学习提出至今，关于迁移学习的研究成果很丰富，主要为基于实例、特征以及模型的知识迁移途径。

1．基于实例的深度迁移学习

如图 6.31 所示，基于实例的深度迁移学习通过调整源样本权重，选择部分源样本作为目标训练集的补充。权重调整策略是这类方法的主要研究重点，MultiSource TrAd-aBoost 算法为源领域中与目标领域相似的样本（正确分类样本），赋予高权重，并降低源领域中与目标领域不相似样本（分类错误样本）的权重。双权重域自适应（BIW）算法通过在数据和特征两个空间的相似度，为源领域实例分配权重。这类方法通过调节源样本权重，使源领域数据分布与目标领域数据分布尽量一致，从而在数据层面消除领域的差异性。

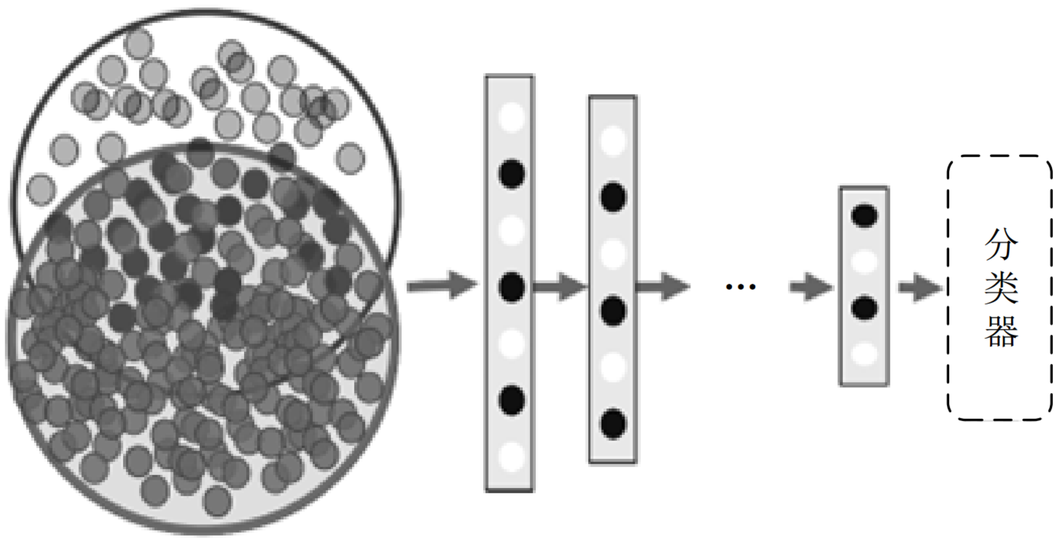


图 6.31　基于实例的深度迁移学习

2．基于特征的深度迁移学习

基于特征的深度迁移学习是指通过学习映射函数将源与目标映射到一个统一的特征空间。在该空间，源和目标领域的差异最小化。其中，在特征空间中度量领域差异的方法是该类方法的研究重点。这类方法假设存在一个领域不变的特征空间，该空间剔除了领域特定属性，保留了源领域与目标领域的共同属性，将迁移问题转化为传统机器学习问题，如图 6.32 所示。

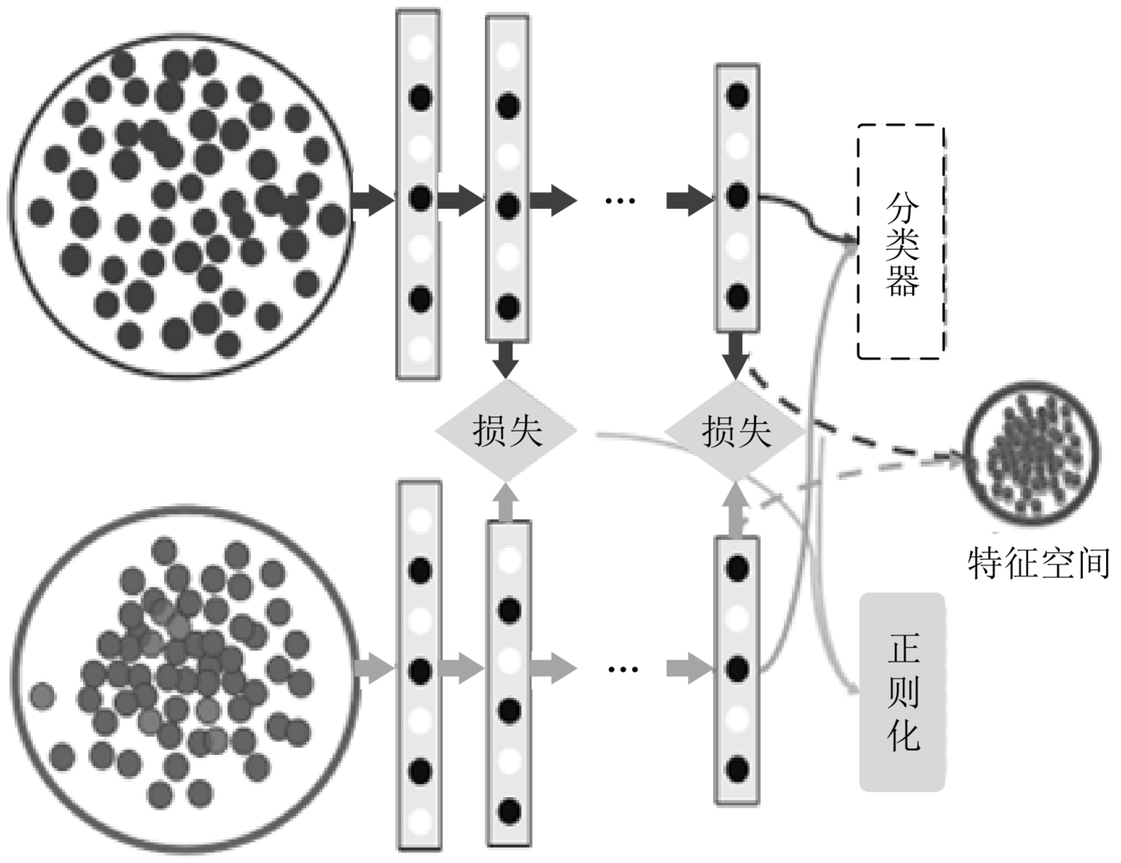


图 6.32　基于特征的深度迁移学习

3．基于模型的深度迁移学习

基于模型的深度迁移学习是指复用在源领域中预先训练好的部分或全部模型，再根据目标与源模型的关系，为源模型分配权重或进行微调。度量、估计不同模型之间的相似度，是这类方法的主要研究难点。根据深度特征的组合特性，低级特征具有高度可复用性，目标模型可以通过复用或微调源模型的部分模型参数，实现知识的迁移，如图 6.33 所示。

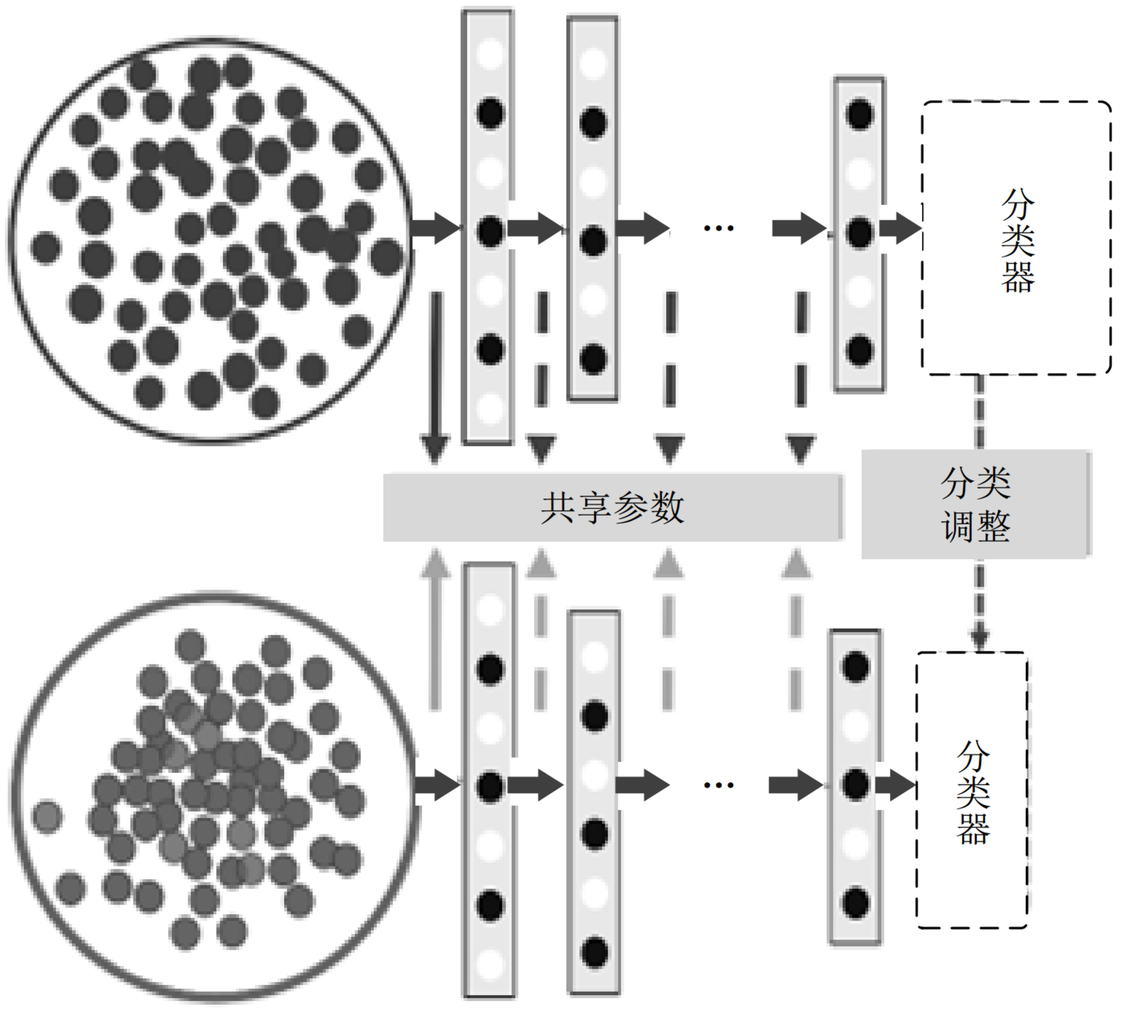


图 6.33　基于模型的深度迁移学习

当前，“大数据”是一个热点问题，这里的“大数据”具有数据量大、数据类型繁多、价值密度低、速度快、时效高等特征。大数据给传统机器学习带来的不仅是因为数据量大而使计算产生存储困难、计算复杂度高的问题，更大的问题是这些数据来自不同

领域，这些不同领域的数据之间存在某些联系，但是不能满足独立同分布的基本假设，因而，传统的机器学习理论和算法不能发挥作用，这给迁移学习带来了机遇。

1）借助特征选择、特征映射、特征降维等方法，通过迁移学习可以将高维空间中大数据投影到低维空间，实现信息的快速学习和预测。

2）大数据使得样本空间有了更多可以借鉴和学习的信息，提供了更多可利用的数据和知识，借助不断更新的迁移技术将在很大程度上提高分类器性能。

3）大数据为机器学习提供了学习更复杂知识的机会，也就是说，随着大数据时代的到来，迁移学习可以进入学习更复杂知识的时代。

6.4.3　强化学习

强化学习属于机器学习中的自监督学习，其与深度学习的融合催生了深度强化学习的出现与发展。强化学习在人工智能领域与深度学习、机器学习的关系如图 6.34 所示。

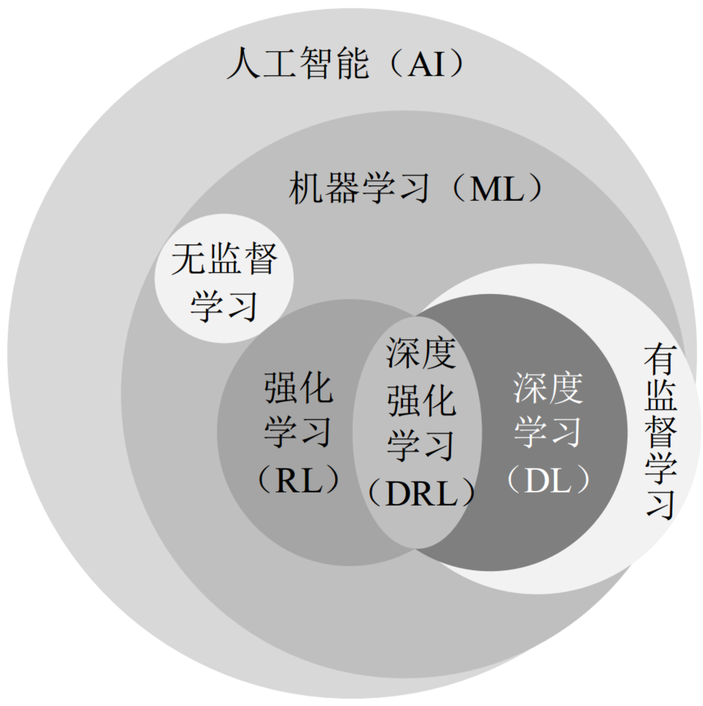


图 6.34　人工智能各方法之间的关系图

强化学习是指智能体在与环境的交互中学习最优行为策略的机器学习问题，其本质是最优序贯决策的学习问题。智能体与环境的交互如图 6.35 所示。智能体获取当前环境的状态，基于当前的状态并依据某策略选择一个动作，该动作作用于环境，环境响应该动作，智能体得到一个奖励。智能体的任务是学习一个策略，能把当前状态映射到最优动作。

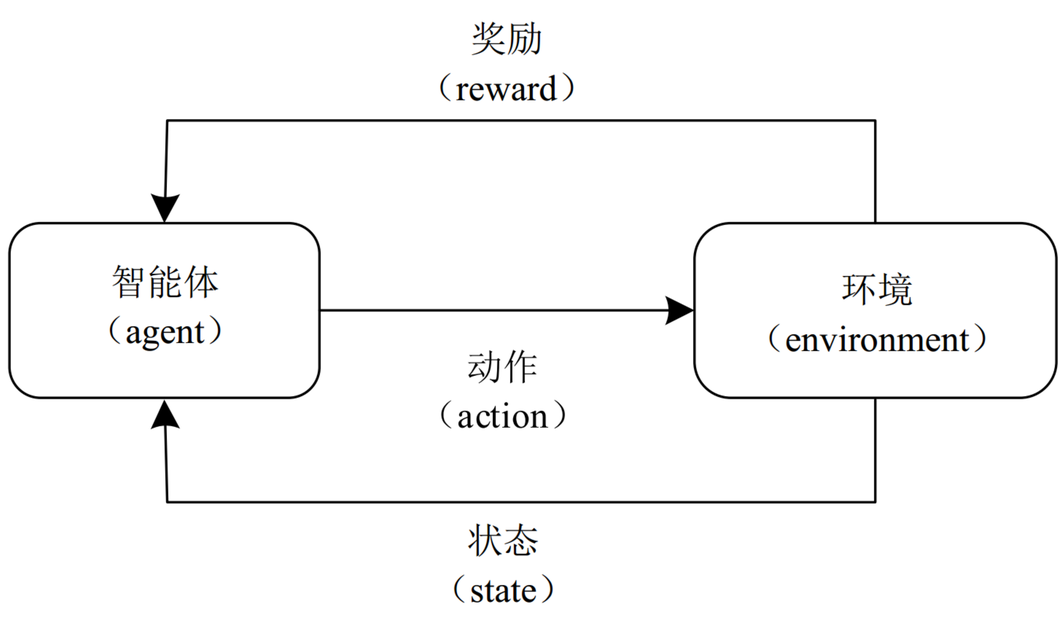


图 6.35　强化学习示意图

马尔可夫决策过程（Markov decision process，MDP）是强化学习问题在数学上的理想化形式，可以表示为一个五元组：

$$\langle S, A(s), P_{ss'}^{a}, R_{ss'}^{a}, V \mid s, s' \in S, a \in A(s) \rangle \tag{6.33}$$

式中，S 是状态集，s, s' 是其中的状态；$A(s)$是在状态 s 下所有可能动作的集合；$P_{ss'}^{a}$ 是在状态 s 时，执行动作 a 后状态变成 s' 的概率，即 $P_{ss'}^{a} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$；$R_{ss'}^{a}$ 是在状态 s 时，执行动作 a 后状态变成 s' 而获得的立即奖励值；V 为价值函数，通常以状态值函数或者状态-动作值函数来表示价值函数。

在 MDP 中，奖励和状态变化仅依赖于当前状态和动作，不依赖于以前的状态和动作。奖励函数可定义为

$$r_t = r(s_t, a_t) \tag{6.34}$$

其含义为在每个离散时间步 t，智能体观察当前状态 s_t 并选择一个动作 a_t 来执行，环境响应此智能体，给出奖励 r_t。

在强化学习中，智能体的任务是学习到一个策略 $\pi: S \rightarrow A$，它基于当前获得的状态 s_t 采取下一步动作 a_t，即 $\pi:(s_t)=a_t$。策略如同智能体的行动指南，它为智能体指明了下一步的行动方向。显然，我们希望智能体能够学到一个最优策略。

价值函数 $V(s)$表示一个状态的价值。在强化学习的具体实现过程中，智能体实际上是根据价值函数来选择下一步的动作。$V^{\pi}(s_t)$是指在策略 π 下状态 s 的价值函数，它表示在策略 π 下从初始状态 s_t 开始获得的累积奖励的期望值。$V^{\pi}(s_t)$ 有多种定义方式，最常用的一种定义方式为折扣累积奖励（discounted cumulative reward），其公式为

$$\begin{aligned} V^{\pi}(s_t) &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots \\ &= \sum_{i=0}^{\infty} \gamma^i r_{t+i} \end{aligned} \tag{6.35}$$

式中，奖励序列 r_{t+i} 是通过从状态 s_t 开始并采用策略 π 来选择动作而生成的；γ 为折扣系数，$0 \leqslant \gamma < 1$，它确定了延迟奖励与当前奖励的相对比例，如果 $\gamma = 0$，则只考虑当前奖励。对所有状态 s，若智能体学习到一个策略 π，使得 $V^{\pi}(s)$为最大，则此策略被称为最优策略，并用 π^* 来表示：

$$\pi^* = \arg\max V^{\pi^*}(s), (\forall s) \tag{6.36}$$

为简化表示，将此最优策略 π^* 的值函数 $V^{\pi^*}(s)$ 记作 $V^*(s)$。

1．Q-learning 强化学习

离轨策略下的时序差分控制算法是强化学习的重要突破之一，这一算法的代表就是 Q 学习（Q-learning）。Q 学习是一种无模型的强化学习方法，学习如何在给定（有限）马尔可夫决策过程中找到最优的动作选择策略。在 Q 学习及与其他相关的技术中，智能体的目标是通过它与环境的交互历史来学习一个最优策略。

在强化学习中，由于不存在监督学习中$\langle s,a \rangle$形式的成对训练数据，因此，智能体无法直接学习最优策略 $\pi^*: S \rightarrow A$。为此，一个容易的做法是，让智能体学习一个定义在状态和动作上的数值评估函数，然后以此评估函数来实现最优策略。显然，在状态 s 下的

最优动作是立即使奖励 $r(s,a)$加上其后继状态的 V^*值（被 γ 折算）最大的动作 a：

$$\pi^* = \arg\max_{\pi}[r(s,a) + \gamma V^*(\delta(s,a))] \tag{6.37}$$

式中，$\delta(s,a)$ 代表在状态 s 下采用动作 a 得到的结果状态。

价值函数表示的是某一状态的价值，一般而言，每个状态下面临多个动作可以选择，每个动作又会产生新的状态。所以可以定义一个状态-动作值函数 $Q(s,a)$，它代表在状态 s 下采用动作 a 得到的奖励。相比而言，状态-动作值函数更直观，其定义为

$$Q(s,a) = r(s,a) + \gamma V^*(\delta(s,a)) \tag{6.38}$$

由式（6.37）和式（6.38）可得

$$\pi^*(s) = \arg\max_{\pi} Q(s,a) \tag{6.39}$$

该式表明，智能体无须进行前瞻性搜索，只需考虑当前状态 s 可选的每个动作 a，并选择使 $Q(s,a)$ 最大的动作，就可以得到最优动作。这是因为状态-动作值函数的定义：在状态 s 下选择动作 a 时得到的折算累积奖励包含了所有需要的现在以及将来的信息。

根据状态-动作值函数 Q 和最大折算累计奖励 V^*之间的关系：

$$V^* = \max_{a'} Q(s,a') \tag{6.40}$$

可得最优状态-动作值函数为

$$Q(s,a) = r(s,a) + \gamma \max_{a'} Q(\delta(s,a),a') \tag{6.41}$$

根据上式的迭代定义，可以得到迭代逼近 Q 学习算法。令 $\hat{Q}$ 表示智能体对实际 Q 函数的估计。在此算法中，智能体使用一个表格来存储 $\hat{Q}(s,a)$，如表 6.7 所示，称其为 Q 表格。每个状态-动作对 $\langle s_i, a_i \rangle$ 都有一个相应的 $\hat{Q}(s_i,a_i)$，即智能体对实际的但未知的 $Q(s_i,a_i)$ 的估计。此表可被初始化为任意值，通常被初始化为 0。

表 6.7　Q 表格

状态	a_1	a_2	a_3	a_4	…
s_1	$\hat{Q}(s_1,a_1)$	$\hat{Q}(s_1,a_2)$	$\hat{Q}(s_1,a_3)$	$\hat{Q}(s_1,a_4)$	…
s_2	$\hat{Q}(s_2,a_1)$	$\hat{Q}(s_2,a_2)$	$\hat{Q}(s_2,a_3)$	$\hat{Q}(s_2,a_4)$	…
s_3	$\hat{Q}(s_3,a_1)$	$\hat{Q}(s_3,a_2)$	$\hat{Q}(s_3,a_3)$	$\hat{Q}(s_3,a_4)$	…
s_4	$\hat{Q}(s_4,a_1)$	$\hat{Q}(s_4,a_2)$	$\hat{Q}(s_4,a_3)$	$\hat{Q}(s_4,a_4)$	…
⋮	⋮	⋮	⋮	⋮	⋮

智能体不断地观察其当前状态 s，选择并执行动作 a，然后获得奖励 $r(s,a)$，环境更新状态为 $s' = \delta(s,a)$，按照下式不断更新 $\hat{Q}(s,a)$，即不断更新 Q 表：

$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a') \tag{6.42}$$

假定被学习的系统可被建模为确定性马尔可夫决策过程，奖励函数 r 有界，并且每个状态-动作对可被无限频繁地访问，则该算法中智能体估计的 $\hat{Q}$ 可收敛到实际 Q 函数。该算法的具体流程如下：

Q 学习算法

Step1：对每个 s、a，初始化表项 $\hat{Q}(s,a)$ 为 0。

Step2：观察当前状态 s。

Step3：重复至收敛。

1）选择一个动作 a 并执行它。

2）接收到立即奖励 r，获得新状态 s'。

3）更新 $\hat{Q}(s,a)$ 表项目：$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')(0 \leqslant \gamma \leqslant 1)$。

4）$s \leftarrow s'$。

2．策略梯度强化学习

Q 学习属于一种基于值迭代的强化学习方法。对于连续空间下的强化学习问题，难以通过值函数来描述每一个状态或动作的好坏。为此，一种较直接的办法是对策略直接进行优化，该方法称为基于策略的强化学习方法。策略梯度强化学习算法的主要思想是将策略 π 参数化表示为 π_θ，这样就可以用梯度等信息寻找最优策略。策略梯度强化学习中，策略可以是确定性的，也可以是随机性的。在确定性策略中，一个状态对应一个动作；而在随机性策略中，相同状态也会对应不同的动作。

下面以随机性策略介绍策略梯度强化学习算法。假设在智能体和环境交互的一个情节（episode）中得到一条采样轨迹 $\tau = (s_0, a_0, s_1, a_1, s_2, a_2, \cdots, s_{T-1}, a_{T-1}, s_T, a_T)$，该轨迹的奖励记为 $\tau(t) = \sum_t^T r(s_t, a_t)$。强化学习的目标是找到一个策略，使得累积奖励的期望最大化，即

$$\pi^* = \arg\max_\pi E_{\tau \sim \pi_\theta(\tau)}[r(\tau)] \tag{6.43}$$

对于采样轨迹 τ，根据概率链式法则可得

$$\pi_\theta(\tau) = p(\tau \mid \theta) = p(s_0) \prod_{t=0}^{T} \pi_\theta(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t) \tag{6.44}$$

式中，$p(s_0)$ 是初始状态分布概率；$p(s_{t+1} \mid s_t, a_t)$ 是状态的转换概率；$\pi_\theta(a_t \mid s_t)$ 是参数为 θ 的策略函数。

对式（6.44）取对数得

$$\log \pi_\theta(\tau) = \log p(\tau \mid \theta) = \log p(s_0) + \sum_{t=0}^{T} [\log \pi_\theta(a_t \mid s_t) + \log p(s_{t+1} \mid s_t, a_t)] \tag{6.45}$$

关于 θ 求导得

$$\begin{aligned} \nabla_\theta \log \pi_\theta(\tau) &= \nabla_\theta \left[\log p(s_0) + \sum_{t=0}^{T} \log \pi_\theta(a_t \mid s_t) + \sum_{t=0}^{T} \log(s_{t+1} \mid s_t, a_t) \right] \\ &= \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \end{aligned} \tag{6.46}$$

记 $J(\theta)=E_{\tau\sim\pi_\theta(\tau)}[r(\tau)]$，则

$$\begin{aligned}\nabla_\theta J(\theta)&=\int_{\tau\sim\pi_\theta(\tau)}\nabla_\theta\pi_\theta(\tau)r(\tau)\mathrm{d}\tau\\&=\int_{\tau\sim\pi_\theta(\tau)}\pi_\theta(\tau)\nabla_\theta\log\pi_\theta(\tau)r(\tau)\mathrm{d}\tau\\&=E_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta\log\pi_\theta(\tau)r(\tau)]\end{aligned}\tag{6.47}$$

根据式（6.46）计算出的 $\nabla_\theta\log\pi_\theta(\tau)$ 可得

$$\begin{aligned}\nabla_\theta J(\theta)&=E_{\tau\sim\pi_\theta(\tau)}\left[\left(\sum_{t=0}^{T}\nabla_\theta\log\pi_\theta(a_t\mid s_t)\right)\left(\sum_{t=0}^{T}r(s_t,a_t)\right)\right]\\&\approx\frac{1}{N}\sum_{i=0}^{T}\left[\left(\sum_{t=0}^{T}\nabla_\theta\log\pi_\theta(a_{i,t}\mid s_{i,t})\right)\left(\sum_{t=0}^{T}r(s_{i,t}\mid a_{i,t})\right)\right]\end{aligned}\tag{6.48}$$

根据式（6.48）即可计算策略梯度，从而更新策略函数的参数。REINFORCE 算法是策略梯度强化学习的典型代表方法，其算法流程如下：

REINFORCE 算法

Step1：给出可微的策略参数化表达。

Step2：初始化参数 θ。

Step3：重复至收敛。

1）根据 $\pi(a_t\mid s_t)$ 采样得到轨迹 τ^t。

2）计算。

$$\nabla_\theta J(\theta)\approx\sum_i\left(\sum_t\nabla_\theta\log\pi_\theta(a_{i,t}\mid s_{i,t})\right)\left(\sum_t r(s_{i,t},a_{i,t})\right)$$

3）$\theta\leftarrow\theta+\alpha\nabla_\theta J(\theta)$。

本 章 小 结

本章对机器学习的基本概念及机器学习分类进行阐述，介绍监督学习算法及其原理，包括 KNN、线性回归、决策树、支持向量机以及神经网络，这些算法主要应用在带有标签的数据场景中，大多简单且有效，可以很方便地借助它们来解决一些实际问题。

在很多实际应用场景中，数据标签不仅获取成本高，而且质量难以保证，本章同时介绍了不需要标签数据就能学习的无监督学习，包括大部分经典的聚类算法，以及弱监督学习，包括半监督学习、迁移学习和强化学习，为后续的课程学习打下基础。

本 章 习 题

1．编程实现 KNN 算法。

2．编程实现线性回归算法，思考除直接求解外，有没有其他方法可以求解。

3．概括三种常见决策树的主要区别。

4．支持向量机中核函数的作用是什么？

5．深度学习的模型有哪些？网络层次加深可能会带来什么负面作用？

6．除了本章介绍的聚类算法外，还有哪些聚类算法？它们的优缺点是什么？

7．迁移学习在 NLP 任务中有哪些应用？

8．思考强化学习应用在机器人领域时，强化学习系统中的几个部分分别对应什么？

9．除了本章介绍的无监督学习算法外，还有哪些场景也属于无监督学习？

第 7 章　神经网络与深度学习

7.1　神经网络概述

人类大脑是一个可以产生意识、思想和情感的器官，其中包含近 860 亿个神经元。神经元是人类大脑中的基本单元，它可以接收其他神经元的信息，也可以给其他神经元发送信息，神经元之间没有物理连接，两个“连接”的神经元之间留有约 20 nm 的间隙，并靠突触进行互联来传递信息，形成一个神经网络。受到人脑神经系统的启发，神经科学家构造了一种模仿人类大脑的数学模型，称为人工神经网络，简称神经网络。神经网络由具有适应性的简单单元组成，它能够模拟人类大脑对真实世界做出的交互反应。

在机器学习领域，神经网络是指由很多人工神经元构成的网络结构模型，这些人工神经元之间的连接权重是可学习的参数。

7.2　神经元与神经网络

神经网络是为模拟人类大脑而设计的一种计算模型，它从结构、实现机理和功能上模拟人脑神经网络。人工神经网络与人类大脑类似，由多个人工神经元互相连接而成，可以对数据之间的复杂关系进行建模；不同节点之间的连接被赋予不同的权重，每个权重代表一个节点对另一个节点的影响大小；每个节点代表一种特定函数，来自其他节点的信息经过其相应的权重计算，输入激活函数中并得到一个新的活性值（兴奋或抑制）。

7.2.1　生物神经元结构

在生物神经网络中，最基本的成分是神经元模型，即上述定义中的“简单单元”。神经元是携带和传输信息的细胞，每个神经元与其他神经元相连。神经元的状态取决于从其他神经细胞接收到的输入信号量，以及突触的强度（抑制或加强），当信号总量超过某个阈值时，细胞体就会兴奋，产生电脉冲，电脉冲沿着轴突并通过突触传递到其他神经元，向相连的神经元发送化学物质，从而改变这些神经元内的电位。如图 7.1 所示，典型的生物神经元结构大致可分为细胞体和细胞突起。

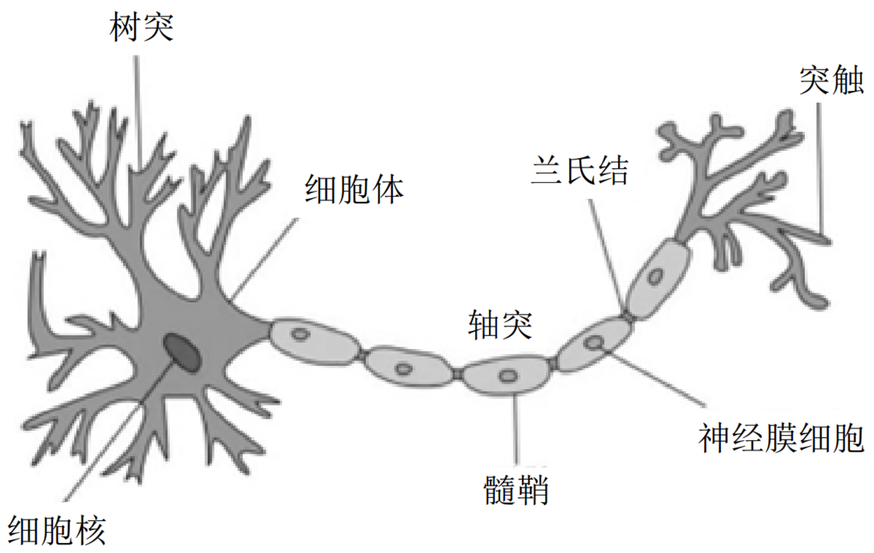


图 7.1　生物神经元结构

1）细胞体中的神经细胞膜上有各种受体和离子通道。细胞膜的受体可与相应的化学物质神经递质结合，引起离子通透性及膜内外电位差发生变化，产生相应的生理活动——兴奋或抑制。

2）细胞突起是由细胞体延伸出来的细长部分，又可分为树突和轴突：①树突可以接受刺激并将兴奋传入细胞体，每个神经元可以有一个或多个树突；②轴突可把自身的兴奋状态从胞体传送到另一个神经元或其他组织，每个神经元只有一个轴突。

7.2.2　神经元数学模型

人工神经元简称神经元，是构成人工神经网络的基本单元，其主要是模拟生物神经元的结构和特性，接收一组输入信号并产生输出。假设一个神经元接收 D 个输入，令向量 $\boldsymbol{x}=[x_1,x_2,\cdots,x_D]$ 表示这组输入，并用净输入 $z\in\mathbf{R}$ 表示一个神经元所获得的输入信号 x 的加权和，用公式表示如下：

$$\begin{aligned} z &= \sum_{d=1}^{D} w_d x_d + b \\ &= \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b \end{aligned} \tag{7.1}$$

式中，$\boldsymbol{w}=[w_1,w_2,\cdots,w_D]\in\mathbf{R}^D$ 是 D 维的权重向量；$b\in\mathbf{R}$ 是偏置。

净输入 z 在经过一个非线性函数 $f(\cdot)$ 后，得到神经元的活性值 a：

$$a = f(z) \tag{7.2}$$

式中，非线性函数 $f(\cdot)$ 称为激活函数（activation function）。图 7.2 所示是典型的人工神经元结构。

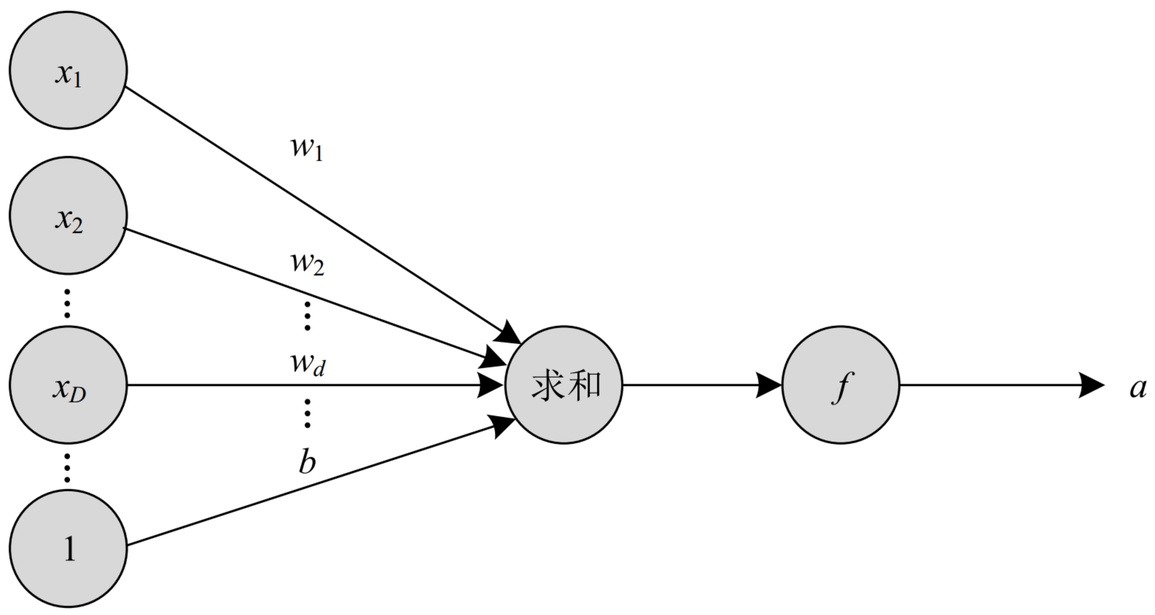


图 7.2　人工神经元结构

激活函数在神经元中具有非常重要的作用，为了增强网络的表示能力和学习能力，激活函数需要具备以下性质。

1）连续并可导的非线性函数；可导的激活函数可以直接利用数值优化的方法来学习网络参数。

2）激活函数及其导函数要尽可能地简单，有利于提高网络计算效率。

3）激活函数的导函数值域要在合适的区间内，否则会影响训练的效率和稳定性。

下面介绍两种在神经网络中常用的激活函数。

1）Sigmoid 型函数是指一类 S 形曲线函数，它为两端饱和函数。常用的 Sigmoid 型函数有 Logistic 函数和 Tanh 函数。

对于函数 $f(x)$：若 $x \to -\infty$，其导数 $f'(x) \to 0$，则称其为左饱和；若 $x \to \infty$，其导数 $f'(x) \to 0$，则称其为右饱和；当同时满足左、右饱和时，称为两端饱和。

Logistic 函数定义为 $\delta(x) = \dfrac{1}{1+\mathrm{e}^{-x}}$，Logistic 函数是一个“挤压”函数，它把实数域的输入“挤压”到(0,1)的输出区间。当输入值在 0 附近时，Sigmoid 型函数近似为线性函数；当输入靠近两端时，对输入进行抑制，输入越小越接近 0，输入越大越接近 1。

Tanh 函数定义为 $\tanh(x) = \dfrac{\mathrm{e}^{x}-\mathrm{e}^{-x}}{\mathrm{e}^{x}+\mathrm{e}^{-x}}$，它可以看作放大并平移的 Logistic 函数：

$$\tanh(x) = 2\sigma(2x) - 1 \tag{7.3}$$

如图 7.3 所示，Tanh 函数的输出是零中心化的，而 Logistic 函数的输出恒大于 0。非零中心化的输出会使其后一层神经元的输入发生偏置偏移，并进一步使梯度下降的收敛速度变慢。

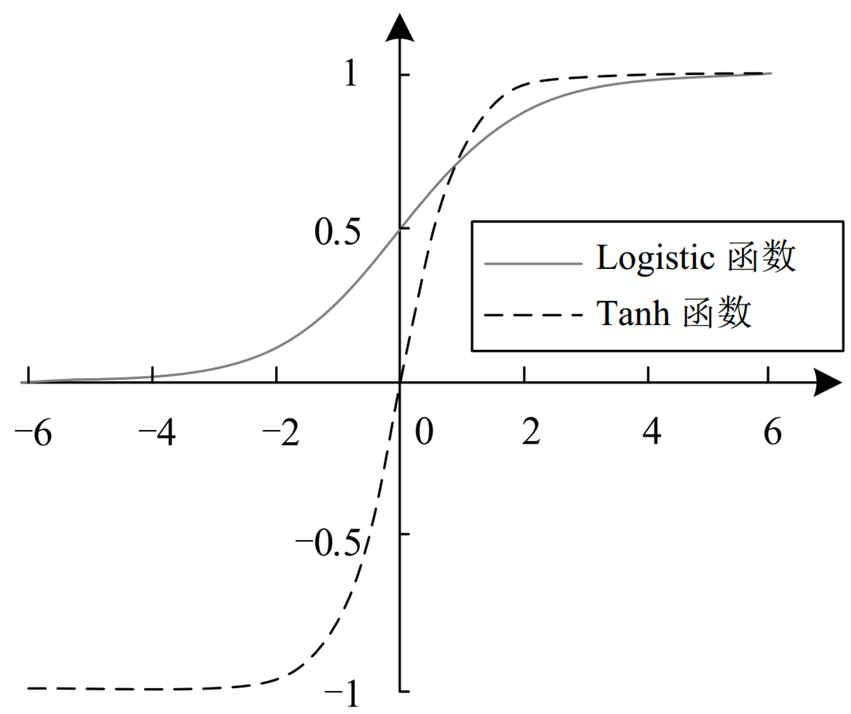


图 7.3　Logistic 函数和 Tanh 函数

2）修正线性单元（rectified linear unit，ReLU）是目前神经网络中经常使用的激活函数，定义为

$$\mathrm{ReLU}(x) = \begin{cases} x, & x \geqslant 0 \\ 0, & x < 0 \end{cases} = \max(0, x) \tag{7.4}$$

采用 ReLU 的神经元只需进行加法、乘法和比较操作，计算上更加高效；在优化方

面，ReLU 函数为左饱和函数，且在 $x>0$ 时导数为 1，在一定程度上缓解了神经网络的梯度消失问题，加速梯度下降的收敛速度。ReLU 神经元在训练时比较容易“死亡”，在训练时，如果参数在一次不恰当的更新后，第一个隐藏层中的某个 ReLU 神经元在所有的训练数据上都不能激活，那么这个神经元自身参数的梯度永远都会是 0，在以后的训练过程中永远不能被激活，这种现象称为死亡 ReLU 问题。为了避免此问题，有几种 ReLU 的变体也被广泛地使用。

① 带泄露的 ReLU（leaky ReLU），$0\leqslant\gamma<1$ 在输入 $x<0$ 时，保持一个很小的斜率，当神经元非激活时也能有一个非零的梯度可以更新参数，避免永远不能被激活。ReLU 定义如下：

$$\text{leaky ReLU}(x)=\begin{cases}x, & x>0\\ \gamma x, & x\leqslant 0\end{cases}=\max(0,x)+\gamma\min(0,x) \tag{7.5}$$

② 带参数的 ReLU（parametric ReLU，PReLU），$0\leqslant\gamma_i<1$ 引入一个可学习的参数，不同神经元拥有不同的参数。PReLU 的定义如下：

$$\text{PReLU}(x)=\begin{cases}x, & x>0\\ \gamma_i x, & x\leqslant 0\end{cases}=\max(0,x)+\gamma_i\min(0,x) \tag{7.6}$$

PReLU 是非饱和函数。如果 $\gamma_i=0$，那么 PReLU 变为 ReLU；如果 γ_i 为一个很小的常数，则 PReLu 可以看作带泄露的 ReLU。PReLU 允许不同神经元拥有不同的参数，也允许一组神经元共享一个参数。

③ 指数线性单元（exponential linear unit，ELU）是一个近似的零中心化的非线性函数，定义如下：

$$\text{ELU}(x)=\begin{cases}x, & x>0\\ \gamma(\mathrm{e}^{x}-1), & x\leqslant 0\end{cases}=\max(0,x)+\min(0,\gamma(\mathrm{e}^{x}-1)) \tag{7.7}$$

式中，$\gamma\geqslant 0$ 是一个超参数，决定 $x\leqslant 0$ 时的饱和曲线，并调整输出均值在 0 附近。

④ Softplus 函数是 ReLU 函数的平滑版本，其定义如下：

$$\text{Softplus}(x)=\lg(1+\mathrm{e}^{x})$$

Softplus 函数的导数是 Logistic 函数。Softplus 函数虽然具有单侧抑制、宽兴奋边界的特性，却没有稀疏激活性。

上述四种激活函数如图 7.4 所示。

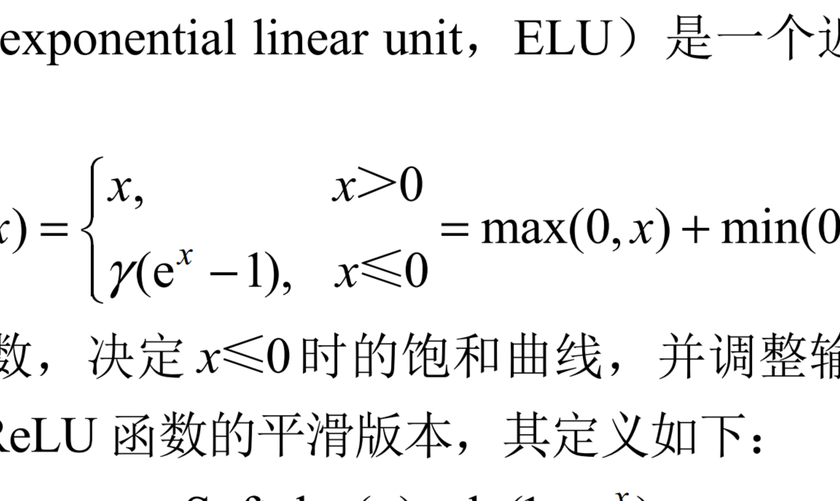


图 7.4　ReLU、leaky ReLU、ELU 和 Softplus 函数

7.2.3　神经网络的结构

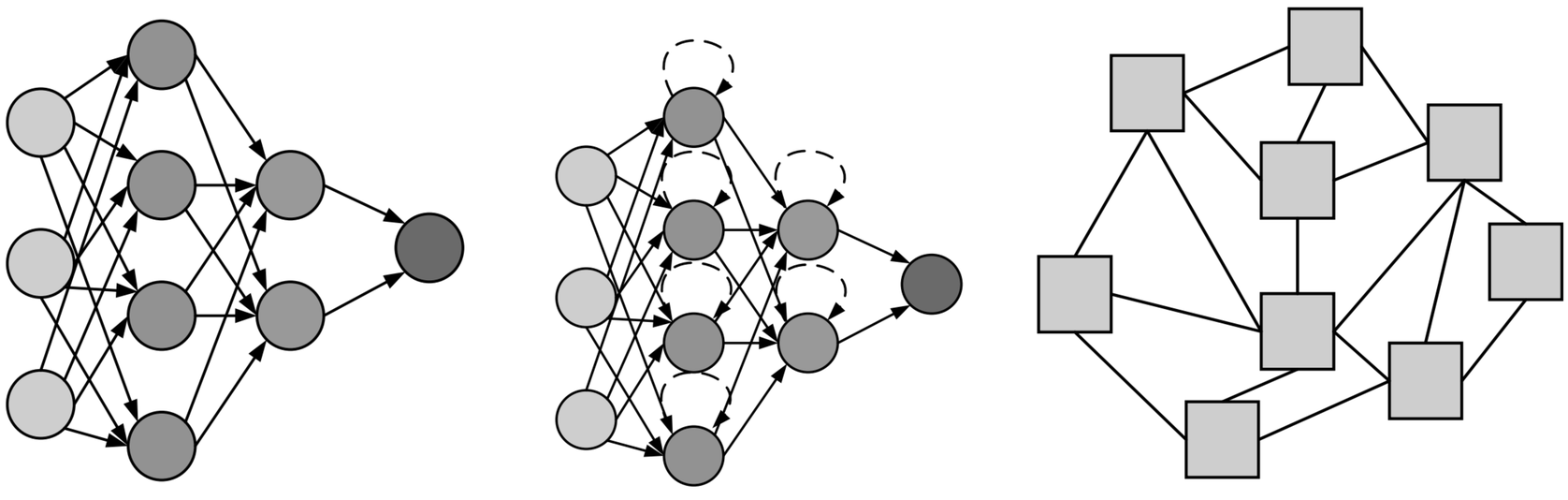
人工神经网络要想模拟人脑的能力，单一的神经元是远远不够的，需要通过很多神经元一起协作来完成复杂的功能，这样通过一定的连接方式或信息传递方式进行协作的神经元可以看作一个网络，就是神经网络。目前常用的神经网络结构有三种：前馈网络、记忆网络和图神经网络。

1）前馈网络中每个神经元按接收信息的先后分为不同的组，每一组可以看作一个神经元，每一层中的神经元接收前一层神经元的输出，并输出到下一层神经元。整个网络中的信息朝一个方向传播，没有反向的信息传播。前馈网络包括全连接前馈网络和卷积神经网络等。

2）记忆网络也称为反馈网络，网络中的神经元不仅可以接收其他神经元的信息，也可以接收自己的历史信息。与前馈网络相比，记忆网络中的神经元具有记忆功能，在不同的时刻具有不同的状态，记忆神经网络中的信息传播可以是单向或双向传递。记忆网络包括循环神经网络（recurrent neural network，RNN）、霍普菲尔德神经网络（Hopfield neural network）、玻尔兹曼机、受限玻尔兹曼机等。

3）图神经网络是定义在图结构数据上的神经网络，图中每个节点由一个或一组神经元构成，节点之间的连接可以是有向的，也可以是无向的，每个节点可以收到来自相邻节点或自身的信息。图神经网络是前馈网络和记忆网络的泛化，包括图卷积网络、图注意力网络、消息传递神经网络等。

上述三种不同的网络结构如图 7.5 所示。



（a）前馈网络　　（b）记忆网络　　（c）图神经网络

图 7.5　前馈网络、记忆网络以及图神经网络的网络结构

7.2.4　神经网络的学习

神经网络的学习过程，就是根据训练数据来调整神经元之间的“连接权”以及每个功能神经元的阈值。如果采用交叉熵损失函数，对于样本(x, y)，其损失函数为

$$L(y, \hat{y}) = -\boldsymbol{y}^{\mathrm{T}} \lg \hat{y} \tag{7.8}$$

式中，$\boldsymbol{y} \in \{0,1\}^{C}$为标签$y$对应的独热（one-hot）向量表示。

一个好的模型应当有一个较小的期望错误，但是由于不知道真实的数据分布和映射

函数，实际上无法计算其期望风险；给定一个训练集，可以计算的是经验风险，即在训练集上的平均损失，在经验风险最小化的基础上再引入参数的正则化来限制模型能力，使其不要过度地最小化经验风险，这个准则就是结构化风险最小化。

给定训练集 $D=\left\{\left(x^{(n)},\hat{y}^{(n)}\right)\right\}_{n=1}^{N}$，将每个样本 $x^{(n)}$ 输入给前馈神经网络，得到网络输出为 $\hat{y}^{(n)}$，其在数据集 D 上的结构化风险函数为

$$R(\boldsymbol{W},\boldsymbol{b})=\frac{1}{N}\sum_{n=1}^{N}L\left(y^{(n)},\hat{y}^{(n)}\right)+\frac{1}{2}\lambda\|\boldsymbol{W}\|_F^2 \tag{7.9}$$

式中，$\boldsymbol{W}$ 和 $\boldsymbol{b}$ 分别表示网络中所有的权重矩阵和偏置向量；$\|\boldsymbol{W}\|_F^2$ 是正则化项，用来防止过拟合；$\lambda>0$ 为超参数，λ 越大，$\boldsymbol{W}$ 越接近于 0。这里的 $\|\boldsymbol{W}\|_F^2$ 一般使用弗罗贝尼乌斯（Frobenius）范数，表示如下：

$$\|\boldsymbol{W}\|_F^2=\sum_{l=1}^{L}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(w_{ij}^{(l)}\right)^2 \tag{7.10}$$

式中，L 为网络层数；m 和 n 分别表示第 l 层权重矩阵的长和宽。

有了学习准则和训练样本，网络参数可以通过梯度下降法来进行学习，在梯度下降法的每次迭代中，第 l 层的 $\boldsymbol{W}^{(l)}$ 和 $\boldsymbol{b}^{(l)}$ 参数更新方式为

$$\boldsymbol{W}^{(l)}\leftarrow\boldsymbol{W}^{(l)}-\alpha\frac{\partial R(\boldsymbol{W},\boldsymbol{b})}{\partial\boldsymbol{W}^{(l)}}=\boldsymbol{W}^{(l)}-\alpha\left(\frac{1}{N}\sum_{n=1}^{N}\left(\frac{\partial L\left(y^{(n)},\hat{y}^{(n)}\right)}{\partial\boldsymbol{W}^{(l)}}\right)+\lambda\boldsymbol{W}^{(l)}\right) \tag{7.11}$$

$$\boldsymbol{b}^{(l)}\leftarrow\boldsymbol{b}^{(l)}-\alpha\frac{\partial R(\boldsymbol{W},\boldsymbol{b})}{\partial\boldsymbol{b}^{(l)}}=\boldsymbol{b}^{(l)}-\alpha\left(\frac{1}{N}\sum_{n=1}^{N}\frac{\partial L\left(y^{(n)},\hat{y}^{(n)}\right)}{\partial\boldsymbol{b}^{(l)}}\right) \tag{7.12}$$

式中，α 为学习率。

7.3 BP 神经网络

多层网络的学习能力比单层网络强得多。训练多层网络时，如果使用梯度下降法，则需要计算损失函数对参数的偏导数；如果通过链式法逐一对每个参数求偏导，则又比较低效。实际上，在训练多层神经网络时经常使用 BP 算法来高效地计算梯度。BP 算法不仅可用于多层前馈神经网络，还可用于其他类型的神经网络。

7.3.1 BP 神经网络的结构

假设一个拥有 d 个输入神经元、l 个输出神经元、q 个隐藏层神经元的多层前馈网络结构，BP 神经网络结构如图 7.6 所示。

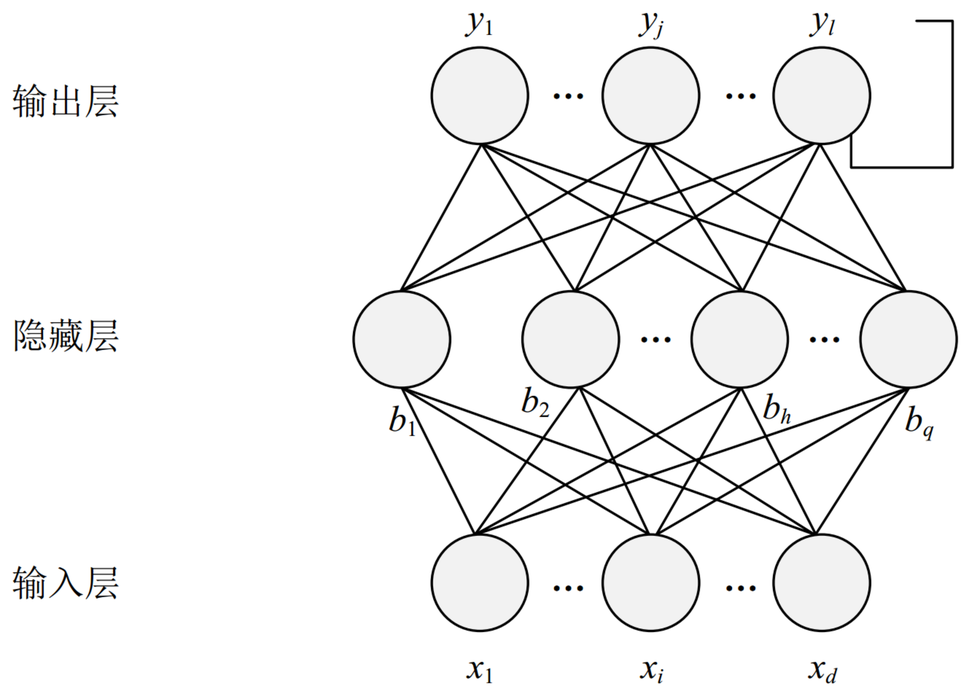


图 7.6　BP 神经网络

7.3.2　BP 学习算法

假设采用梯度下降法进行神经网络参数学习，给定一个样本(x,y)，将其输入神经网络模型中，得到网络输出$\hat{y}$，损失函数为$L(y,\hat{y})$，进行参数学习，需要计算损失函数关于每个参数的导数。

对第l层中的参数$\boldsymbol{W}^{(l)}$和$b^{(l)}$计算偏导数，根据链式法则有

$$\frac{\partial L(y,\hat{y})}{\partial w_{ij}^{(l)}}=\frac{\partial z^{(l)}}{\partial w_{ij}^{(l)}}\frac{\partial L(y,\hat{y})}{\partial z^{(l)}} \tag{7.13}$$

$$\frac{\partial L(y,\hat{y})}{\partial b^{(l)}}=\frac{\partial z^{(l)}}{\partial b^{(l)}}\frac{\partial L(y,\hat{y})}{\partial z^{(l)}} \tag{7.14}$$

式（7.13）和式（7.14）中第二项都是目标函数关于第l层的神经元$z^{(l)}$偏导数，称为误差项。只需计算三个偏导数：$\frac{\partial z^{(l)}}{\partial w_{ij}^{(l)}}$、$\frac{\partial z^{(l)}}{\partial b^{(l)}}$和$\frac{\partial L(y,\hat{y})}{\partial z^{(l)}}$。

1）计算偏导数$\frac{\partial z^{(l)}}{\partial w_{ij}^{(l)}}$。因$z^{(l)}=\boldsymbol{W}^{(l)}a^{(l-1)}+b^{(l)}$，则偏导数

$$\begin{aligned}\frac{\partial z^{(l)}}{\partial w_{ij}^{(l)}}&=\left[\frac{\partial z_1^{(l)}}{\partial w_{ij}^{(l)}},\cdots,\frac{\partial z_i^{(l)}}{\partial w_{ij}^{(l)}},\cdots,\frac{\partial z_{M_l}^{(l)}}{\partial w_{ij}^{(l)}}\right]\\&=\left[0,\cdots,\frac{\partial\left(w_{i:}^{(l)}a^{(l-1)}+b_i^{(l)}\right)}{\partial w_{ij}^{(l)}},\cdots,0\right]\\&=\left[0,\cdots,a_j^{(l-1)},\cdots,0\right]\\&\triangleq l_i\left(a_j^{(l-1)}\right)\in\mathbf{R}^{1\times M_l}\end{aligned} \tag{7.15}$$

式中，$w_{i:}^{(l)}$为权重矩阵$\boldsymbol{W}^{(l)}$的第i行；$l_i\left(a_j^{(l-1)}\right)$表示第$i$个元素为$a_j^{(l-1)}$，其余为 0 的行向量。

2）计算偏导数 $\frac{\partial z^{(l)}}{\partial b^{(l)}}$。因为 $z^{(l)}=\boldsymbol{W}^{(l)}a^{(l-1)}+b^{(l)}$，所以偏导数

$$\frac{\partial z^{(l)}}{\partial b^{(l)}}=I_{M_l}\in\mathbf{R}^{M_l\times M_l} \tag{7.16}$$

为 $M_l\times M_l$ 的单位矩阵。

3）计算偏导数 $\frac{\partial L(y,\hat{y})}{\partial z^{(l)}}$。这个偏导数为第 l 层神经元的误差项，表示第 l 层神经元对最终损失的影响，也反映了最终损失对第 l 层神经元的敏感程度，用 $\delta^{(l)}$ 表示为

$$\delta^{(l)}\triangleq\frac{\partial L(y,\hat{y})}{\partial z^{(l)}}\in\mathbf{R}^{M_l} \tag{7.17}$$

根据 $z^{(l+1)}=\boldsymbol{W}^{(l+1)}a^{(l)}+b^{(l+1)}$ 有

$$\frac{\partial z^{(l+1)}}{\partial a^{(l)}}=\left(\boldsymbol{W}^{(l+1)}\right)^{\mathrm{T}}\in\mathbf{R}^{M_l\times M_{l+1}} \tag{7.18}$$

根据 $a^{(l)}=f_l(z^{(l)})$，其中 $f_l(\cdot)$ 为按位计算的函数：

$$\begin{aligned}\frac{\partial a^{(l)}}{\partial z^{(l)}}&=\frac{\partial f_l(z^{(l)})}{\partial z^{(l)}}\\&=\operatorname{diag}\left(f_l'\left(z^{(l)}\right)\right)\in\mathbf{R}^{M_l\times M_l}\end{aligned} \tag{7.19}$$

因此，根据链式法则，第 l 层的误差项为

$$\begin{aligned}\delta^{(l)}&\triangleq\frac{\partial L(y,\hat{y})}{\partial z^{(l)}}\\&=\frac{\partial a^{(l)}}{\partial z^{(l)}}\frac{\partial z^{(l+1)}}{\partial a^{(l)}}\frac{\partial L(y,\hat{y})}{\partial z^{(l+1)}}\\&=\operatorname{diag}\left(f_l'\left(z^{(l)}\right)\right)\cdot\left(\boldsymbol{W}^{(l+1)}\right)^{\mathrm{T}}\cdot\delta^{(l+1)}\\&=f_l'\left(z^{(l)}\right)\odot\left(\left(\boldsymbol{W}^{(l+1)}\right)^{\mathrm{T}}\cdot\delta^{(l+1)}\right)\in\mathbf{R}^{M_l}\end{aligned} \tag{7.20}$$

式中，$\odot$ 表示向量的每个元素相乘。

从误差项式可以看出，第 l 层的误差项可以通过第 l+1 层的误差项计算得到，这就是误差的反向传播。反向传播算法的含义如下：第 l 层的一个神经元的误差项是所有与该神经元相连的第 l+1 层的神经元的误差项的权重和，然后再乘上该神经元激活函数的梯度。

因此，$L(y,\hat{y})$ 关于第 l 层权重 $\boldsymbol{W}^{(l)}$ 的梯度为

$$\frac{\partial L(y,\hat{y})}{\partial \boldsymbol{W}^{(l)}}=\delta^{(l)}(a^{(l-1)})^{\mathrm{T}}\in\mathbf{R}^{M_l\times M_{l-1}} \tag{7.21}$$

$L(y,\hat{y})$ 关于第 l 层偏置 $\boldsymbol{b}^{(l)}$ 的梯度为

$$\frac{\partial L(y,\hat{y})}{\partial b^{(l)}}=\delta^{(l)}\in\mathbf{R}^{M_l} \tag{7.22}$$

使用误差反向传播算法的前馈神经网络训练过程可以分为如下三步。

1）前馈计算每一层的净输入 $z^{(l)}$ 和激活值 $a^{(l)}$，直到最后一层。

2）反向传播计算每一层的误差项 $\delta^{(l)}$。

3）计算每一层参数的偏导数，并更新参数。

算法 7.1　使用反向传播算法的随机梯度下降训练过程

输入：训练集 $D=\left\{\left(x^{(n)},y^{(n)}\right)\right\}_{n=1}^{N}$，验证集 V，学习率 α，正则化系数 λ，网络层数 L，神经元数量 M_l，$1\leqslant l\leqslant L$。

1）随机初始化 $\boldsymbol{W}$、$\boldsymbol{b}$；

2）repeat

3）　　对训练集 D 中的样本随机重新排序；

4）　　for $n=1\cdots N$ do

5）　　从训练集中选取样本 $\left(x^{(n)},y^{(n)}\right)$；

6）　　前馈计算每一层的净输入 $z^{(l)}$ 和激活值 $a^{(l)}$，直到最后一层；

7）　　反向传播计算每一层的误差 $\delta^{(l)}$；

　　# 计算每一层参数的导数

8）　　$\dfrac{\partial L(y,\hat{y})}{\partial W^{(l)}}=\delta^{(l)}\left(a^{(l-1)}\right)^{\mathrm{T}}$；

9）　　$\dfrac{\partial L(y,\hat{y})}{\partial b^{(l)}}=\delta^{(l)}$；

　　# 更新参数

10）　　$\boldsymbol{W}^{(l)}\leftarrow\boldsymbol{W}^{(l)}-\alpha\left(\delta^{(l)}\left(a^{(l-1)}\right)^{\mathrm{T}}+\lambda\boldsymbol{W}^{(l)}\right)$；

11）　　$\boldsymbol{b}^{(l)}\leftarrow\boldsymbol{b}^{(l)}-\alpha\delta^{(l)}$；

12）　　验证集 V 计算错误率；

13）　　end

14）until 神经网络模型在验证集 V 上的错误率不再下降；

输出：$\boldsymbol{W}$、$\boldsymbol{b}$。

7.3.3　BP 神经网络的应用

BP 算法是迄今为止最成功的神经网络学习算法，在自然语言处理领域和计算机视觉领域的诸多任务中都会用到 BP 算法，它不仅解决了多层神经网络的参数训练问题，同时促进了深度神经网络的发展。BP 算法不仅可用于多层前馈神经网络，还可用于其他类型的神经网络，如训练递归神经网络。

7.4　卷积神经网络

卷积神经网络（convolution neural network，CNN）是一种具有局部连接、权重共

享等特性的深层前馈神经网络。在用全连接前馈网络处理图像时，会存在以下问题：①如果输入图像大小为 28（像素）×28（像素）×3（通道），第一个隐藏层的每个神经元到输入层都有 28（像素）×28（像素）×3（通道）=2352 个互相独立的连接，每个连接对应一个权重参数，随着隐藏层神经元数量的增多，参数的规模也会急剧增加，这会导致神经网络的训练效率非常低且容易出现过拟合问题；②自然图像中的物体具有局部不变的特征，而全连接前馈网络难以提取这些局部不变的特征，需要进行数据增强来提高模型性能。

目前的卷积神经网络一般是由卷积层、汇聚层和全连接层交叉堆叠而成的前馈神经网络。卷积神经网络有三个结构上的特性：局部连接、权重共享以及汇聚。这些特性使卷积神经网络具有一定程度上的平移、缩放和旋转不变性。与全连接前馈神经网络相比，卷积神经网络的可学习参数更少。

7.4.1 人脑视觉激励

卷积神经网络受到生物学上感受野机制的启发而提出，感受野机制主要是指听觉、视觉等神经系统中一些神经元的特性，即神经元只接收其所支配的刺激区域内的信号。在视觉神经系统中，视觉皮层中的视觉细胞的输出依赖于视网膜上的光感受器。视网膜上的光感受器受刺激兴奋时，将神经冲动信号传到视觉皮层，但不是所有视觉皮层中的神经元都会接收这些信号。一个神经元的感受野是指视网膜上的特定区域，只有这个区域内的刺激才能够激活该神经元。

7.4.2 卷积神经网络的结构

典型的卷积网络是由卷积层、汇聚层、全连接层交叉堆叠而成。目前常用的卷积神经网络整体结构如图 7.7 所示，一个卷积块为连续 m 个卷积层和 b 个汇聚层，一个卷积网络中可以堆叠 n 个连续的卷积块，然后在后面接着 k 个全连接层。

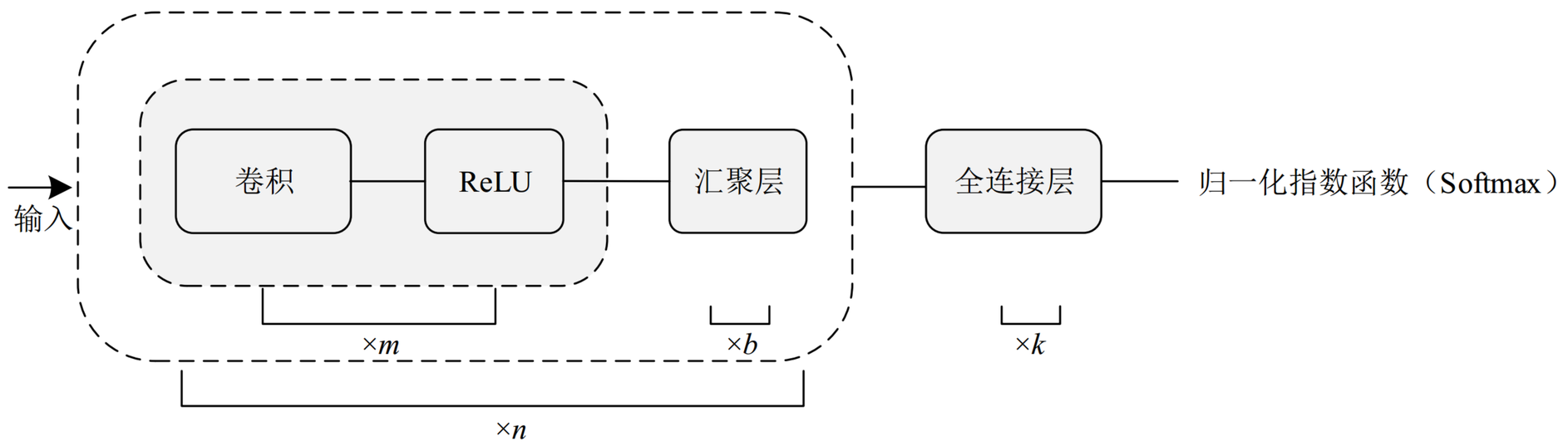


图 7.7 卷积神经网络结构

7.4.3 卷积神经网络的局部连接

卷积神经网络采用卷积层来代替全连接层，第 l 层的净输入 $z^{(l)}$ 为第 $l-1$ 层活性值 $a^{(l-1)}$ 和卷积核 $\boldsymbol{w}^{(l)} \in \mathbf{R}^K$ 的卷积，即

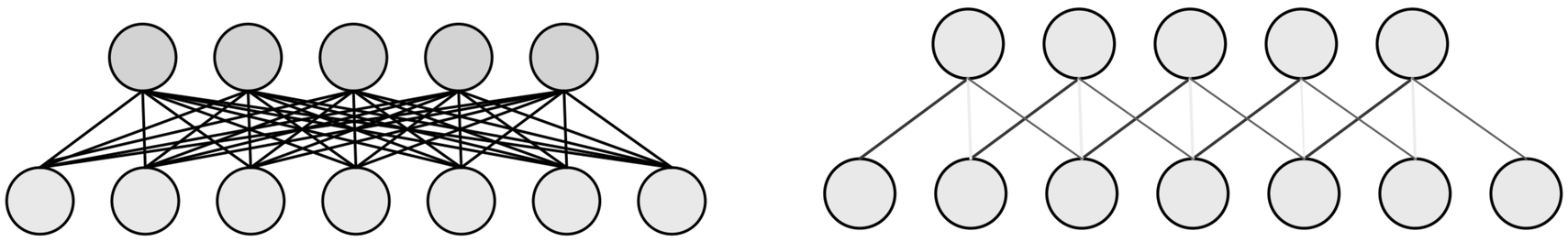
$$z^{(l)} = \boldsymbol{w}^{(l)} \otimes a^{(l-1)} + b^{(l)} \tag{7.23}$$

式中，卷积核 $\boldsymbol{w}^{(l)} \in \mathbf{R}^K$ 为可学习的权重向量；$b^{(l)} \in \mathbf{R}$ 为可学习的偏置。

卷积层有两个重要的性质。

1）局部连接：在卷积层中的每一个神经元都只与前一层中某个局部窗口内的神经元相连，构成一个局部连接网络，卷积层和前一层之间的连接数大大减少，由原来的 $M_l \times M_{l-1}$ 个连接数变为 $M_l \times K$ 连接，K 为卷积核大小。

2）权重共享：作为参数的卷积核 $\boldsymbol{w}^{(l)}$ 对于第 l 层的所有神经元都是相同的，如图 7.8 所示，所有同颜色连接上的权重是相同的。权重共享可以理解为一个卷积核只捕捉输入数据中的一种特定的局部特征，因此，如果要提取多种特征就需要使用多个不同的卷积核。



（a）全连接层　　（b）卷积层

图 7.8　卷积层代替全连接层

7.4.4　卷积神经网络的权值

在卷积神经网络中，主要有两种功能的神经层：卷积层和汇聚层。由于局部连接和权重共享，卷积层的参数有 K 维的权重和 1 维的偏置，共 $K+1$ 个参数，与全连接前馈神经网络类似，卷积神经网络也可以通过误差反向传播算法来进行参数更新。

对第 l 层为卷积层，第 $l-1$ 层的输入特征映射为 $x^{(l-1)} \in \mathbf{R}^{M \times N \times D}$，通过卷积计算得到第 l 层的特征映射净输入 $z^{(l)} \in \mathbf{R}^{M' \times N' \times P}$，第 l 层的第 $p(1 \leqslant p \leqslant P)$ 个特征映射净输入为

$$z^{(l,p)} = \sum_{d=1}^{D} \boldsymbol{w}^{(l,p,d)} \otimes x^{(l-1,d)} + b^{(l,p)} \tag{7.24}$$

式中，$\boldsymbol{w}^{(l,p,d)}$ 和 $b^{(l,p)}$ 为卷积核以及偏置。第 l 层中共有 $p \times d$ 个卷积核和 p 个偏置，可以分别使用链式法则来计算其梯度。

损失函数关于第 L 层的卷积核 $\boldsymbol{w}^{(l,p,d)}$ 和第 p 个偏置 $b^{(l,p)}$ 的偏导数为

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{w}^{(l,p,d)}} &= \frac{\partial L}{\partial z^{(l,p)}} \otimes x^{(l-1,d)} \\
&= \delta^{(l,p)} \otimes x^{(l-1,d)} \frac{\partial L}{\partial b^{(l,p)}} \\
&= \sum\nolimits_{i,j} \left[\delta^{(l,p)}\right]_{i,j}
\end{aligned} \tag{7.25}$$

式中，$\delta^{(l,p)} = \dfrac{\partial L}{\partial z^{(l,p)}}$ 为损失函数关于第 l 层的第 p 个特征映射净输入 $z^{(l,p)}$ 的偏导数。

在卷积网络中，每层参数的梯度依赖其所在层的误差项。

7.4.5　卷积神经网络的多卷积核

在卷积神经网络中，采用卷积层代替全连接层，卷积层的作用是提取一个局部区域

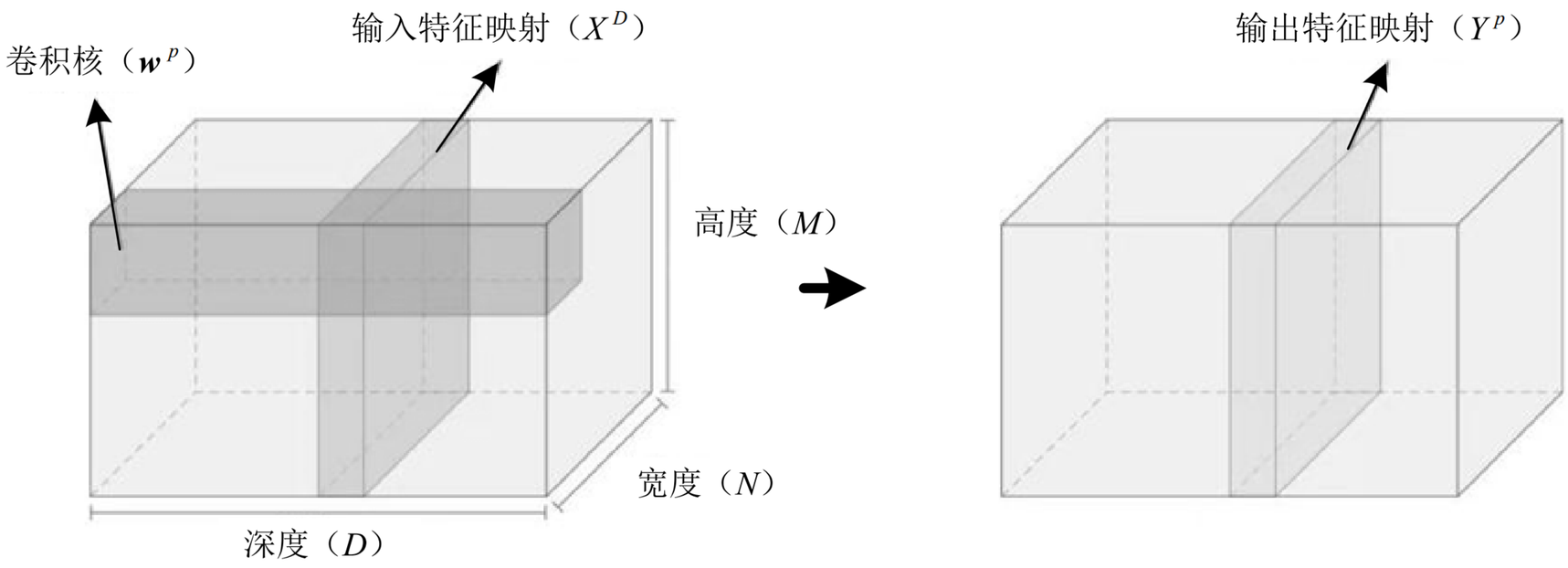
的特征，不同的卷积核相当于不同的特征提取器。图像为二维结构，为了更充分地提取局部信息，使用 D 个 N（宽度）×M（高度）的卷积核对其进行卷积操作，最终获得大小为 $N\times M\times D$ 特征映射。

特征映射为一幅图像经过卷积提取到的特征，每个特征映射可以作为一类抽取的图像特征。为了提高卷积网络的表示能力，可以在每一层使用多个不同的特征映射，以更好地表示图像的特征。如图 7.9 所示，为了计算输出特征映射 Y^p，用卷积核 $\boldsymbol{w}^{p,1},\boldsymbol{w}^{p,2},\cdots,\boldsymbol{w}^{p,D}$ 分别对输入特征映射 $X^1,X^2,\cdots,X^D$ 进行卷积，然后将卷积结果进行相加，并加上一个标量偏置 b^p 得到卷积层的净输入 Z^p，再经过非线性激活函数后得到特征映射 Y^p：

$$\begin{aligned} Z^p &= \boldsymbol{w}^p \otimes X + b^p \\ &= \sum_{d=1}^{D} \boldsymbol{w}^{p,D} \otimes X^D + b^p \end{aligned} \tag{7.26}$$

$$Y^p = f(Z^p) \tag{7.27}$$

式中，$\boldsymbol{w}^p \in \mathbf{R}^{U\times V\times D}$ 为三维卷积核，每个切片矩阵 $\boldsymbol{w}^{p,D} \in \mathbf{R}^{U\times V}$ 为一个二维卷积核。



（a）输入特征映射组 x　　（b）输出特征映射组 y

图 7.9　卷积层三维结构

7.4.6　卷积神经网络的池化

汇聚层（pooling layer）也叫子采样层，其作用是进行特征选择，降低特征数量，从而减少网络参数数量。卷积层虽然可以显著减少网络中连接的数量，但是特征映射组中的神经元个数并没有显著减少，如果后面接分类器，分类器的输入维数依然很高，很容易出现过拟合。为解决此问题，可以在卷积层之后增加汇聚层，从而降低特征维数，避免过拟合。汇聚操作是指对每个区域进行下采样得到一个值，作为这个区域的概括。常用的汇聚函数有两种。

1）最大汇聚（maximun pooling）：对于一个区域，选择这个区域内所有神经元的最大活性值作为这个区域的表示，即 $y_{m,n}^d = \max(x_i), i \in R_{m,n}^d$，其中 x_i 为区域 $R_{m,n}^d$ 内每个神经元的活性值。

2）平均汇聚（mean pooling）：取区域内所有神经元活性值的平均值，即

$y_{m,n}^{d}=\frac{1}{\left|R_{m,n}^{d}\right|}\sum_{i\in R_{m,n}^{d}}x_i$。

图 7.10 所示是使用最大汇聚进行子采样操作的示例。汇聚层不但可以有效地减少神经元数量，还可以使网络对一些小的局部形态变化保持不变性，并拥有更大的感受野。

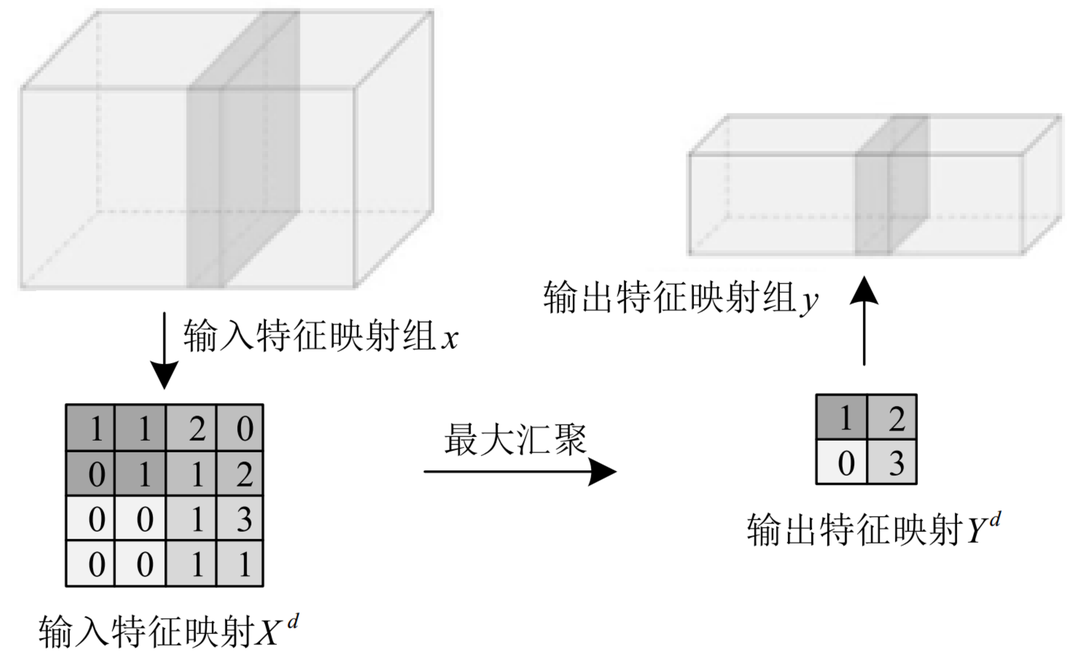


图 7.10　汇聚层中采用最大汇聚进行子采样示例

典型的汇聚层是将每个特征映射划分为 2×2 大小的不重叠区域，然后使用最大汇聚的方式进行下采样。汇聚层也可以看作一个特殊的卷积层，卷积核大小为 $k\times k$，步长为 $s\times s$，卷积核为 max 函数或 mean 函数。过大的采样区域会急剧减少神经元的数量，也会造成过多的信息损失。

7.4.7　卷积神经网络应用

卷积神经网络主要应用在图像和视频分析的各种任务（如图像分类、人脸识别、物体识别、图像分割等）上，其准确率远远超出了其他的神经网络模型。近年来，卷积神经网络也广泛地应用到自然语言处理、推荐系统等领域。

7.5　循环神经网络

在前馈神经网络中，信息的传递是单向的，可以看作一个复杂的函数，网络的输出只依赖当前的输入。但是在实际任务中，当理解一句话的意思时，孤立地理解这句话的每个词是不够的，需要处理这些词连接起来的整个序列；当处理视频的时候，也不能只单独地分析每一帧，而是要分析这些帧连接起来的整个序列。另外，前馈神经网络要求输入和输出的数据维度是固定的，而像文本、语音、视频等时序数据的长度一般是不固定的，前馈神经网络难以处理时序数据，因此需要一种能力更强的模型。

循环神经网络（RNN）是一类具有短期记忆能力的网络。在循环神经网络中，神经元不但可以接收其他神经元的信息，也可以接收自身的信息，形成具有环路的网络结构。与前馈神经网络相比，循环神经网络更加符合生物神经网络的结构，更适合处理时序任务。

7.5.1 循环神经网络的结构

循环神经网络中存在循环结构，指的是神经元的输入是该层神经元的输出。如图 7.11 所示，左侧是 RNN 的结构图，右侧是 RNN 结构按时刻展开的效果。时刻是 RNN 中一个非常重要的概念，不同时刻的记忆在隐藏单元中存储和流动，每一个时刻的隐含层单元有一个输出。在 RNN 中，各隐含层共享参数。

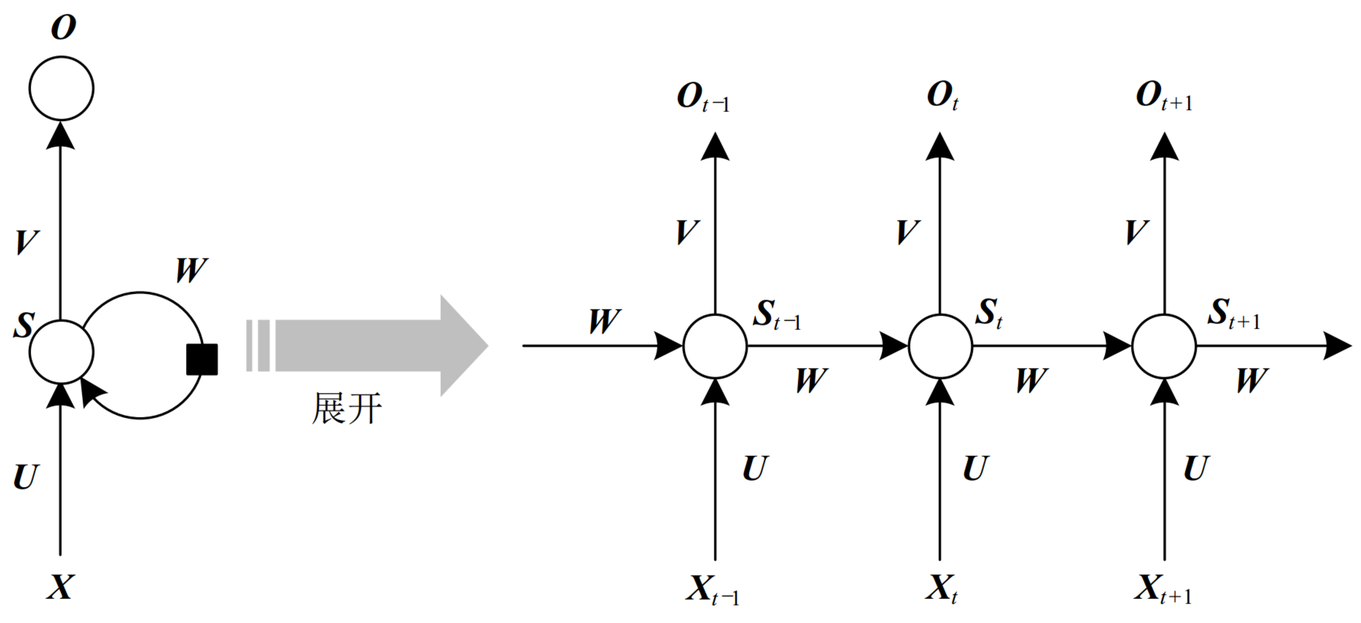


图 7.11　循环神经网络结构和按时刻展开（$\boldsymbol{W}$、$\boldsymbol{U}$、$\boldsymbol{V}$ 为共享参数）

图 7.11 中，$\boldsymbol{X}_t$ 是 t 时刻的输入向量；$\boldsymbol{S}_t$ 是 t 时刻的隐状态，基于上一时刻的隐状态和当前输入得到 $\boldsymbol{S}_t = f(\boldsymbol{U}\boldsymbol{X}_t + \boldsymbol{W}\boldsymbol{S}_{t-1})$；$\boldsymbol{O}_t$ 表示 t 时刻的输出，$\boldsymbol{O}_t = \text{softmax}(\boldsymbol{V}\boldsymbol{S}_t)$。

7.5.2 循环神经网络的输出

记忆在隐藏单元中存储和流动，而输出取自隐含单元以及网络的最终输出。展开后可以看到网络输出的整个过程，即图 7.11 中的 O_{t-1}、O_t、O_{t+1} 展开后的神经网络，每一层负责一个输出。$\boldsymbol{O}_t$ 是 t 时刻神经网络的输出，从神经网络中产生每个时刻的输出信息。例如，在文本处理中，输出的是词汇的概率向量，通过 Softmax 得出。

根据输入、输出的不同，RNN 可以有不同的分类，如图 7.12 所示。

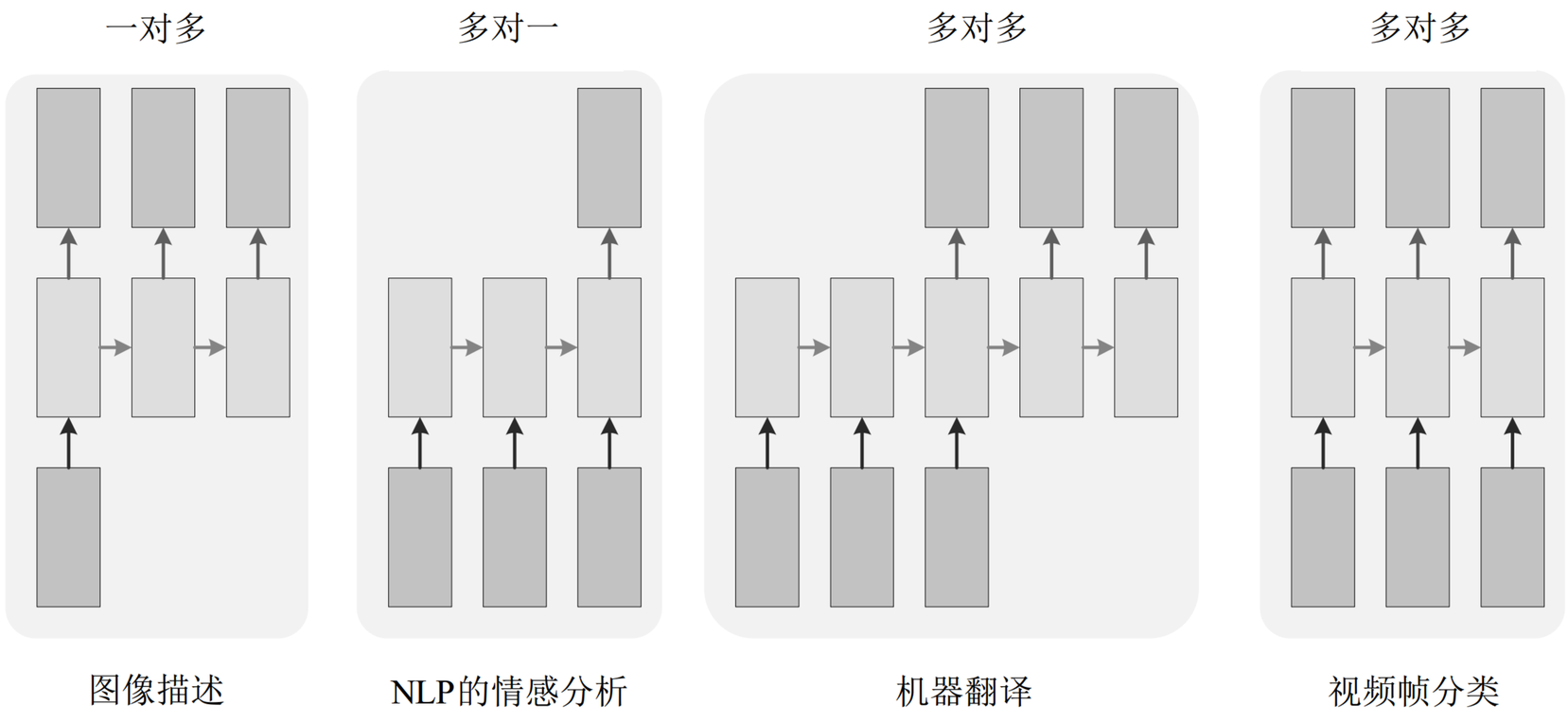


图 7.12　RNN 分类

7.5.3　梯度消失和梯度爆炸

循环神经网络的求解可以采用基于时间的反向传播（back propagation through time，BPTT）算法实现，BPTT 实际上是 BP 算法的简单变种。BPTT 算法将循环神经网络看作一个展开的多层前馈网络，其中“每一层”对应循环网络中的“每个时刻”。这样循环神经网络就可以按照前馈网络中的反向传播算法计算参数梯度。在“展开”的前馈网络中，所有层的参数是共享的，因此参数的真实梯度是所有“展开层”的参数梯度之和。

计算偏导数 $\frac{\partial \mathcal{L}_t}{\partial \boldsymbol{U}}$，首先计算公式 $\frac{\partial \mathcal{L}}{\partial \boldsymbol{U}}=\sum_{t=1}^{T}\frac{\partial \mathcal{L}_t}{\partial \boldsymbol{U}}$ 中 t 时刻损失函数对参数 $\boldsymbol{U}$ 的偏导数 $\frac{\partial \mathcal{L}_t}{\partial \boldsymbol{U}}$。因为参数 $\boldsymbol{U}$ 与隐藏层在每个时刻 $k(1\leqslant k\leqslant t)$ 的净输入 $Z_k=\boldsymbol{U}\boldsymbol{h}_{k-1}+\boldsymbol{W}\boldsymbol{x}_{\boldsymbol{k}}+b$ 有关，因此 t 时刻的损失函数 $\mathcal{L}_t$ 关于参数 u_{ij} 的梯度为

$$\frac{\partial \mathcal{L}_t}{\partial u_{ij}}=\sum_{k=1}^{t}\frac{\partial^{+} Z_k}{\partial u_{ij}}\frac{\partial \mathcal{L}_t}{\partial Z_k} \tag{7.28}$$

式中，$\frac{\partial^{+} Z_k}{\partial u_{ij}}$ 表示“直接”偏导数，即公式 $Z_k=\boldsymbol{U}\boldsymbol{h}_{k-1}+\boldsymbol{W}\boldsymbol{x}_k+b$ 中保持 h_{k-1} 不变，对 u_{ij} 进行求偏导数，得

$$\begin{aligned}\frac{\partial^{+} Z_k}{\partial u_{ij}}&=[0,\cdots,[h_{k-1}]_j,\cdots,0]\\&\triangleq\prod(x_i)([h_{k-1}]_j)\end{aligned} \tag{7.29}$$

式中，$[h_{k-1}]_j$ 为 $k-1$ 时刻隐状态的第 j 维；$\prod_i(x)$ 除了第 i 行值为 x，其余都为 0 的行向量。

定义误差项 $\delta_{t,k}=\frac{\partial \mathcal{L}_t}{\partial Z_k}$ 为第 t 时刻的损失函数对第 k 时刻隐藏神经层的净输入 Z_k 的导数，当 $1\leqslant k<t$ 时，则有

$$\begin{aligned}\delta_{t,k}&=\frac{\partial \mathcal{L}_t}{\partial Z_k}\\&=\frac{\partial h_k}{\partial Z_k}\frac{\partial Z_{k+1}}{\partial h_k}\frac{\partial \mathcal{L}_t}{\partial Z_{k+1}}\\&=\operatorname{diag}(f'(Z_k))\boldsymbol{U}^{\mathrm{T}}\delta_{t,k+1}\end{aligned} \tag{7.30}$$

将式（7.30）和式（7.29）代入式（7.28）得

$$\frac{\partial \mathcal{L}_t}{\partial u_{ij}}=\sum_{k=1}^{t}[\delta_{t,k}]_i[h_{k-1}]_j \tag{7.31}$$

将式（7.31）写成矩阵形式为

$$\frac{\partial \mathcal{L}_t}{\partial \boldsymbol{U}}=\sum_{k=1}^{t}\delta_{t,k}\boldsymbol{h}_{k-1}^{\mathrm{T}} \tag{7.32}$$

将式（7.32）代入$\frac{\partial \mathcal{L}}{\partial \boldsymbol{U}}=\sum_{t=1}^{T}\frac{\partial \mathcal{L}_t}{\partial \boldsymbol{U}}$，得到整个序列的损失函数$\mathcal{L}$关于参数$\boldsymbol{U}$的梯度为

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{U}}=\sum_{t=1}^{T}\sum_{k=1}^{t}\delta_{t,k}\boldsymbol{h}_{k-1}^{\mathrm{T}} \tag{7.33}$$

同理可得，$\mathcal{L}$关于权重$\boldsymbol{W}$和偏置b的梯度为

$$\frac{\partial \mathcal{L}}{\partial W}=\sum_{t=1}^{T}\sum_{k=1}^{t}\delta_{t,k}\boldsymbol{x}_{k}^{\mathrm{T}} \tag{7.34}$$

$$\frac{\partial \mathcal{L}}{\partial b}=\sum_{t=1}^{T}\sum_{k=1}^{t}\delta_{t,k} \tag{7.35}$$

在 BPTT 算法中，将式（7.30）展开得

$$\delta_{t,k}=\prod_{\tau=k}^{t-1}(\mathrm{diag}(f'(Z_\tau))\boldsymbol{U}^{\mathrm{T}})\delta_{t,t} \tag{7.36}$$

如果定义$\gamma\cong\|\mathrm{diag}(f'(Z_\tau))U^{\mathrm{T}}\|$，则有

$$\delta_{t,k}\cong\gamma^{t-k}\delta_{t,t} \tag{7.37}$$

若$\gamma>1$，当$t-k\to\infty$时，$\gamma^{t-k}\to\infty$。当间隔$t-k$比较大时，梯度也变得很大，会造成系统不稳定，称为梯度爆炸问题（gradient exploding problem）。

相反，若$\gamma<1$，当$t-k\to\infty$时，$\gamma^{t-k}\to 0$。当间隔$t-k$比较大时，梯度变得非常小，会出现与深层前馈神经网络类似的梯度消失问题（vanishing gradient problem）。

7.5.4 长短时记忆网络

由于存在梯度消失和梯度爆炸的问题，导致 RNN 的训练很难获取长时依赖信息。为了解决长时依赖问题，一种解决方案是引入门控机制来控制新的累积速度，包括有选择地加入新的信息，并有选择地遗忘之前累积的信息，这类网络可以称为基于门控的循环神经网络（gated RNN）。长短时记忆网络（long short-term memory networks，LSTM）就是一种基于门控的循环神经网络。

LSTM 是循环神经网络的一个变体，可以有效地解决简单循环神经网络的梯度爆炸或消失问题。LSTM 的两大改进如下。

1）新的内部状态。LSTM 网络中引入一个新的内部状态$\boldsymbol{c}_t\in\mathbf{R}^D$专门进行线性循环信息传递，同时非线性地输出信息给隐藏层的外部状态$\boldsymbol{h}_t\in\mathbf{R}^D$。内部状态$\boldsymbol{c}_t$通过下式计算：

$$\boldsymbol{c}_t=f_t\odot c_{t-1}+i_t\odot\tilde{c}_t \tag{7.38}$$

$$\boldsymbol{h}_t=o_t\odot\tanh(c_t) \tag{7.39}$$

式中，$f_t\in[0,1]^D$、$i_t\in[0,1]^D$和$o_t\in[0,1]^D$为三个门，用于控制信息传递的路径；$\odot$为向量元素乘积；$\boldsymbol{c}_{t-1}$为上一时刻的记忆单元；$\boldsymbol{c}_t\in\mathbf{R}^D$是通过非线性函数得到的候选状态：

$$\tilde{\boldsymbol{c}}_t=\tanh(\boldsymbol{W}_c\boldsymbol{x}_t+\boldsymbol{U}_c\boldsymbol{h}_{t-1}+b_c) \tag{7.40}$$

在每个时刻 t，LSTM 网络的内部状态 $\boldsymbol{c}_t$ 记录了到当前时刻为止的历史信息。

2）门控机制。LSTM 网络引入门控机制来控制信息传递的路径。式（7.38）和式（7.39）中的三个门分别为输入门 i_t、遗忘门 f_t 和输出门 o_t，其作用如下。

① 遗忘门 f_t 控制上一个时刻的内部状态 $\boldsymbol{c}_{t-1}$ 需要遗忘多少信息。

② 输入门 i_t 控制当前时刻的候选状态 $\tilde{\boldsymbol{c}}$ 有多少信息需要保存。

③ 输出门 o_t 控制当前时刻的内部状态 $\boldsymbol{c}_t$ 有多少信息需要输出给外部状态 $\boldsymbol{h}_t$。

当 $f_t=0$，$i_t=1$ 时，记忆单元将历史信息清空，并将候选状态 $\tilde{\boldsymbol{c}}$ 写入。但此时记忆单元 $\boldsymbol{c}_t$ 依然与上一时刻的历史信息相关。当 $f_t=1$，$i_t=0$ 时，记忆单元将复制上一时刻的内容，不写入新的信息。

LSTM 网络中的“门”是一种“软”门，取值在（0,1）之间，表示以一定的比例允许信息通过。三个门的计算方式如下：

$$i_t=\sigma(\boldsymbol{W}_i\boldsymbol{x}_t+\boldsymbol{U}_i\boldsymbol{h}_{t-1}+b_i) \tag{7.41}$$

$$f_t=\sigma(\boldsymbol{W}_f\boldsymbol{x}_t+\boldsymbol{U}_f\boldsymbol{h}_{t-1}+b_f) \tag{7.42}$$

$$o_t=\sigma(\boldsymbol{W}_o\boldsymbol{x}_t+\boldsymbol{U}_o\boldsymbol{h}_{t-1}+b_o) \tag{7.43}$$

式中，$\sigma(\cdot)$ 为 logistic 函数，其输出区间为（0,1），$\boldsymbol{x}_t$ 为当前时刻的输入，$\boldsymbol{h}_{t-1}$ 为上一时刻的外部状态。

图 7.13 所示是 LSTM 网络的循环单元结构，其计算过程如下。

1）首先利用上一时刻的外部状态 $\boldsymbol{h}_{t-1}$ 和当前时刻的输入 $\boldsymbol{x}_t$，计算三个门，以及候选状态 $\tilde{\boldsymbol{c}}$。

2）结合遗忘门 f_t 和输入门 i_t 来更新记忆单元 $\boldsymbol{c}_t$。

3）结合输出门 o_t，将内部状态的信息传递给外部状态 $\boldsymbol{h}_t$。

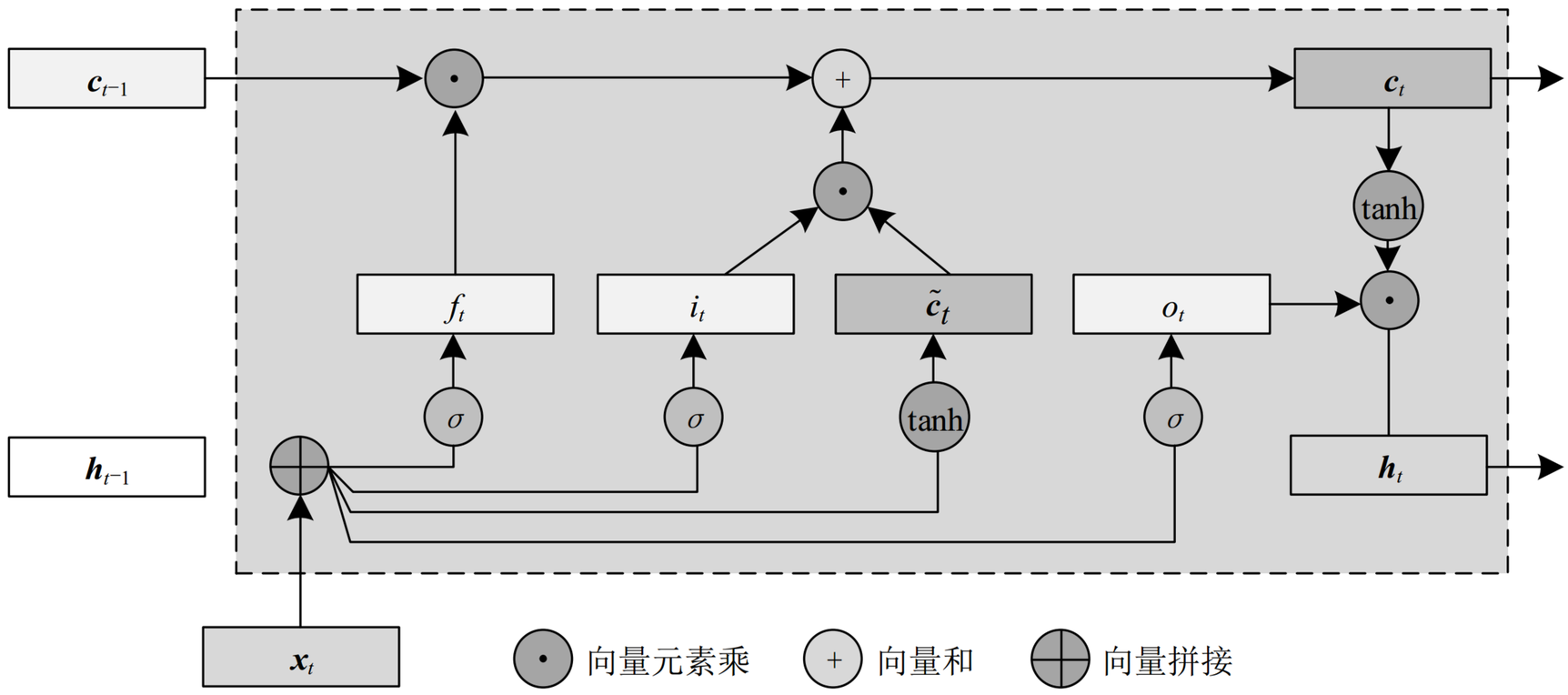


图 7.13　LSTM 网络的循环单元结构

7.5.5　循环神经网络的应用

循环神经网络是目前深度学习最有前景的工具之一，它解决了传统神经网络不能从数据中共享位置特征的问题，大部分涉及序列数据的处理问题，都可以使用 RNN。典型的 RNN 应用场景有文本生成、机器翻译、语音识别、生成图像描述、视频标记等。

7.6 生成对抗网络

生成对抗网络（generative adversarial networks，GAN）是一个具有开创意义的深度生成模型，突破了以往的概率模型必须通过最大似然估计来学习参数的限制。然而，生成对抗网络的训练通常比较困难。本节介绍生成对抗网络的基本原理和结构。

7.6.1 生成对抗网络的基本原理

生成对抗网络由生成模型和判别模型构成。生成模型捕捉样本数据的分布，用服从某一分布（均匀分布、高斯分布等）的噪声 z 生成一个类似真实训练数据的样本，效果越像真实样本越好。判别模型是一个二分类器，估计一个样本来自训练数据（而非生成数据）的概率，如果样本来自真实的训练数据，则判别模型输出大概率；否则，判别模型输出小概率。

可以做如下类比：生成网络如同假币制造团伙，专门制造假币，判别网络如同警察，专门检测使用的货币是真币还是假币，生成模型的目标是想方设法生成与真币一样的货币，使判别模型判别不出来，判别模型的目标是想方设法检测出生成模型所生成的假币。

假设输入是 x，G 表示生成模型，D 表示判别模型，模型 D 的输出是[0,1]之间的数，表示输入 x 是真实样本的概率。令 p_r 和 p_g 代表真实样本和生成样本的分布，那么对于判别模型，显然有目标函数

$$\max_D E_{x\sim p_r}[\lg D(x)] + E_{x\sim p_g}[\lg(1-D(x))] \tag{7.44}$$

也就是使判别模型对真实样本尽量输入 **1**，对生成样本尽量输出 **0**，类似的生成模型就是生成让判别模型无法分辨的样本，那么对于生成模型 G 的目标函数就是

$$\min_G \max_D E_{x\sim p_r}[\lg D(x)] + E_{z\sim \text{uniform}}[\lg(1-D(G(z)))] \tag{7.45}$$

生成模型 G 的输入是符合某一均匀分布的噪声数据 z，这个最大、最小化目标函数如何进行优化呢？直观的处理办法就是分别对 D 和 G 进行交互迭代。

1）在开始阶段，生成样本与真实样本的分布差别比较大，判别模型的输出没有规律。

2）固定生成模型，学习判别模型，判别模型学习后，可以对生成样本和真实样本进行判别。

3）固定判别模型，学习生成模型，生成模型学习后，生成样本的分布向真实样本逼近。

反复迭代更新判别模型和生成模型，生成样本的分布与真实样本一致，判别模型对真实样本和生成样本已经无法分辨，说明生成的样本足可以以假乱真，最后的生成模型就是我们希望得到的模型。所以，通过不断对抗得到的生成神经网络就是生成对抗网络。

7.6.2　生成对抗网络的结构

在 GAN 中，有两个网络进行对抗训练。一个是判别网络 D，目标是尽量准确地判断一个样本是来自真实数据还是由生成网络产生；另一个是生成网络 G，目标是尽量生成判别网络无法区分来源的样本，其结构如图 7.14 所示。

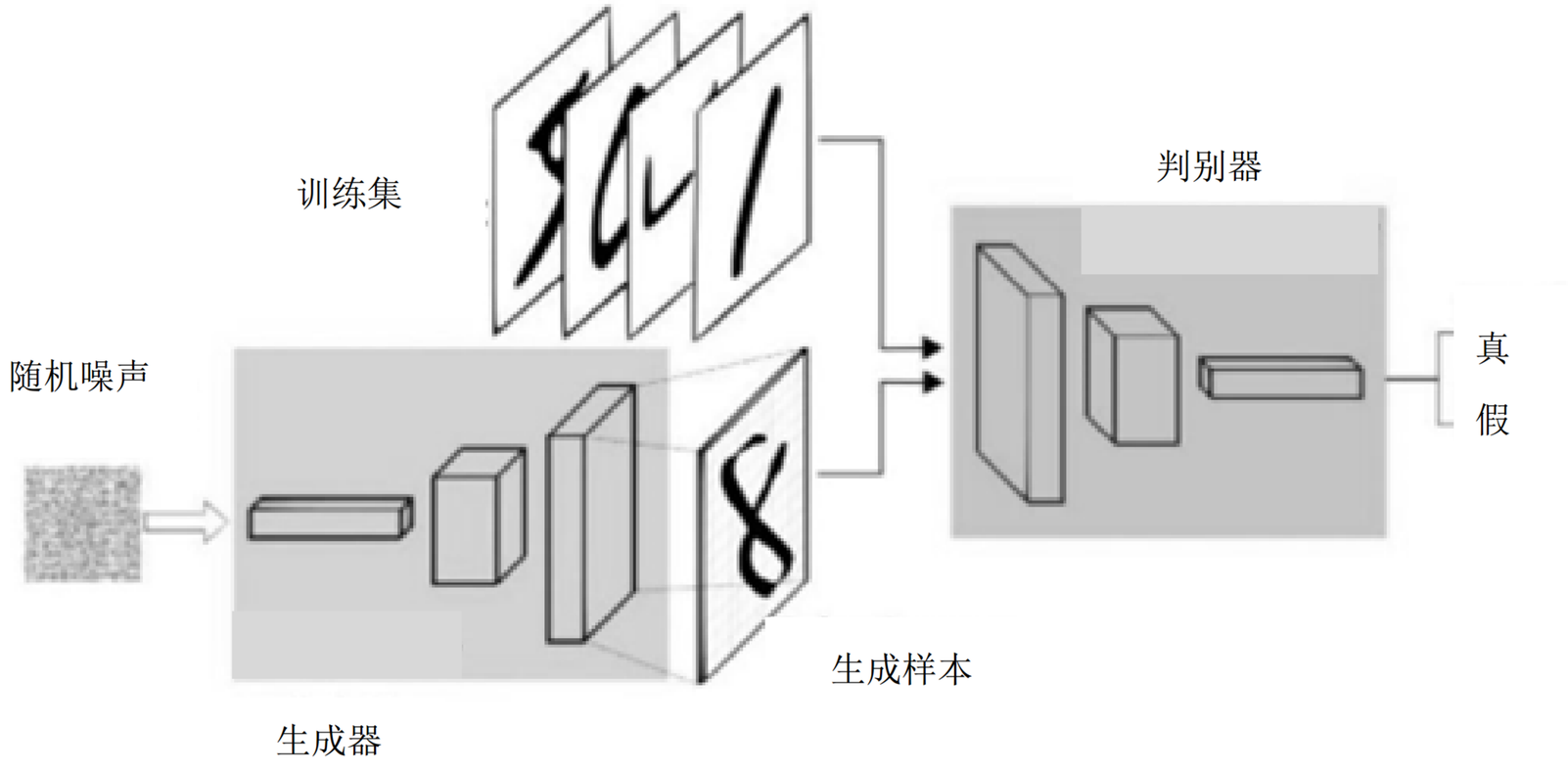


图 7.14　GAN 网络结构

7.7　深度学习的应用

近年来，随着人工智能的发展，深度学习俨然已经成为一个热词。谷歌将深度学习应用于语音识别和图像识别，奈飞（Netflix）和亚马逊则利用深度学习来了解客户的行为习惯。麻省理工学院的研究学者甚至已经在尝试用深度学习来预测未来。深度学习作为机器学习的一部分，利用人工神经元概念，模仿人类大脑的生物神经元，通过深度神经网络，可以自动提取对算法分类较为重要的特征，从而代替传统机器学习手动特征提取的任务。

7.7.1　深度学习在博弈中的应用

2016 年，由 Alphabet 旗下的 DeepMind 公司研发的计算机围棋程序阿尔法围棋（AlphaGo）打败了近 15 年来一直被认为是世界顶尖棋手的李世石九段。这距 IBM 的“深蓝”程序击败国际象棋棋王卡斯帕罗夫正好 20 年，也再一次在学术界和民间掀起了人工智能的热潮。与“深蓝”不同的是，AlphaGo 的成果极大程度上归功于其采用深度学习的算法。本节介绍深度学习在博弈中的应用。

博弈或称玩游戏，是人工智能的经典问题。根据不同的分类方法，游戏可以被分为单人游戏、双人（或多人）游戏，以及棋盘游戏、视频游戏、协作游戏和对战游戏等。双人或更多人的游戏常被称为博弈，博弈可以被分为完全信息博弈（如围棋和象棋）和

非完全信息博弈（如扑克和军棋）。

围棋是现存的极度复杂的完全信息博弈之一。相比其他的棋盘游戏（如国际象棋和西洋跳棋），围棋的搜索空间极大，局面的描述和评估都极难。随着卷积神经网络在其他游戏上取得的巨大成功，人们期望可以用同样的方法来解决计算机围棋问题。然而，由于围棋的策略和形势判断太过复杂，用相同方法对围棋的策略或值函数做近似很难。由于每个棋盘状态真正的值是未知的，如果没有足够好的初始策略作为强化学习的基础，利用不准确的动作和奖励信息训练出来的神经网络未必能做可靠的预测。为了解决这个难题，研究者们想到了用人类的棋谱做深度神经网络的训练集，因为在人类棋谱中每步棋的选择和每盘棋的胜负都是已知的。

AlphaGo 于是用自我对局的棋盘状态和胜负来训练一个价值网络，以预测任意棋盘状态的值。这个价值网络进一步提高了 AlphaGo 的棋力，为了让训练数据尽可能独立，AlphaGo 从每盘自战中只选取一个局面来做训练。另外，AlphaGo 推出了新的蒙特卡罗树搜索算法。该算法结合蒙特卡罗模拟以及价值和策略网络来减少计算开销。价值网络被用来评估当前棋盘的状态（即预测一个给定局面的值），从而有效地降低了搜索规划的深度。策略网络则被用来选择下一步（即预测一个给定局面的下一步棋的概率分布），从而有效地降低了搜索的宽度。这些网络训练是人类专家动作的监督学习和自我游戏的强化学习的精密结合。AlphaGo 力挫欧洲围棋冠军和世界顶尖棋手李世石。

扑克游戏的挑战来源于信息不完全，即玩家对历史事件只能进行部分观察，而看不到对手的信息。阿尔伯塔大学的迈克尔·鲍林（Michael Bowling）等为德州扑克游戏提出了一个近似纳什均衡的解决方案。此方案的基本思想是用两个遗憾（regret）最小化算法反复自我对局。雅克文科（Yakovenko）等与哥伦比亚大学的研究者们用类似的方法训练了针对扑克的卷积神经网络，并在三种常见的扑克游戏中能与人类专家抗衡。DeepMind 的约翰内斯·海因里希（Johannes Heinrich）等和大卫·西尔韦（David Silver）等提出了更适用于非完全信息博弈的深度强化学习算法，称为神经虚拟自我对局（neural fictitious self-play，NFSP）。NFSP 的主要思想是近似博弈论中经典的虚拟对局（fictitious play）模型，其不借助先验知识，并能在二人零和游戏（zero-sum games）或者多人博弈（potential games）中通过自我对局收敛到纳什平衡。

围棋和扑克只是博弈中的冰山一角。最近的研究热点逐渐从棋牌游戏转向视频游戏，如微软把游戏《我的世界》作为测试人工智能的研究平台，DeepMind 声称他们的下一个挑战是《星际争霸》。相比棋盘游戏，这些即时战略游戏的状态更复杂，信息更不完全，动作的选择空间更大。

7.7.2 深度学习在医学影像识别中的应用

近年来，由于深度学习可以学习图像中的复杂特征，因此其在医学图像分析中得到了广泛的应用，其中包括医学图像病变部位的识别和分析。在医学图像病变部位的识别和分析方面，传统机器学习的方法针对疑似病变组织图像需要人工分割图像进行特征提取，然后通过训练好的分类器进行病变图像的识别。人工分割图像是一个烦琐的过程，使用深度学习技术，可以直接对疑似病变图像进行训练，可以“自动”地将病变组织进

行识别、分类。

1．肺部图像识别和分析

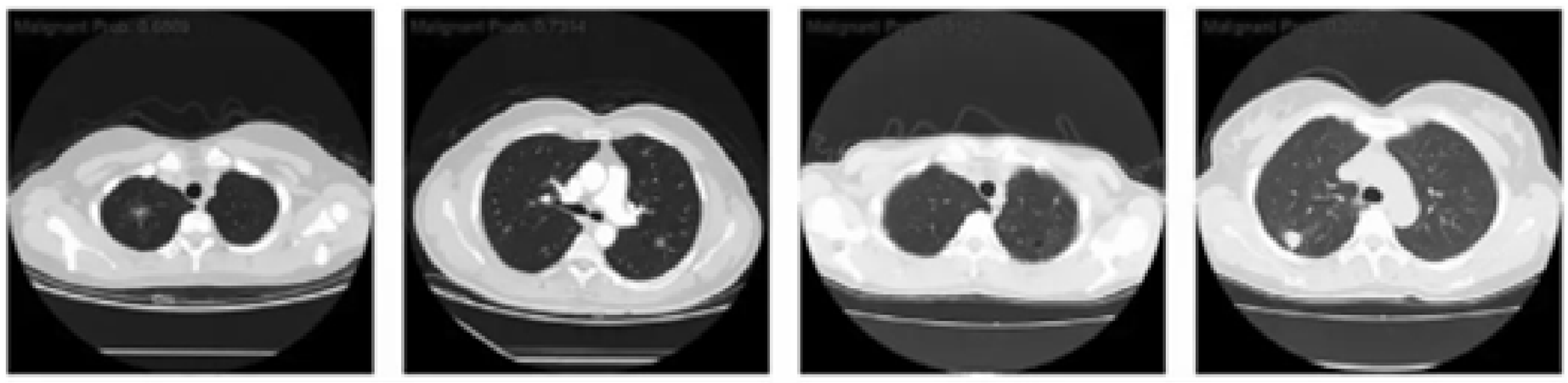
腾讯优图实验室联合广东省肺癌研究所吴一龙教授/钟文昭教授团队，与清华大学以及国内多家呼吸科/放射科团队等共同开发了基于卷积神经网络算法的肺结节诊断模型。其中包含以下几个模块。

1）预处理模块：从 CT 切片中分离出包含肺组织的图像区域，阈值 0HU 去除无关组织（骨与软组织），使用自适应阈值图像分割方法，重建三维图像为标准化的切片厚度和分辨率，以避免不同 CT 机器的误差。

2）结节诊断模块：建立 3D 肺结节检测网络获得重建图像的 3D 特征；在 Pytorch 平台上分两步训练卷积神经网络模型、结节检测网络，输入信息包括图像和相应的位置附加信息；根据第一步的检测网络参数初始化结节诊断网络，并对输出图像和相关诊断结果进行微调。

3）输出模块：在结节诊断网络计算出所有结节的恶性评分后，对其进行融合以得出最后的图像及恶性评分。

基于 CNN 的肺结节诊断如图 7.15 所示。



病理证实的恶性疾病，恶性概率高于50%　　病理证实的恶性疾病，恶性概率低于50%

图 7.15　基于 CNN 的肺结节诊断

这项研究使用基于深度学习算法的模型显著提高了早期肺癌检出和诊断的敏感性及特异性，且其诊断效力较经验丰富的专科医师团队更优，显示出今后应用这一类模型算法辅助临床医师进行日常肺部影像诊断的可行性，同时提高了早期结节检出率及诊断率，使更多患者能够得到早期治疗干预，达到早期临床治愈的效果。

2．乳腺图像识别和分析

卡内罗（Carneiro）等利用卷积神经网络对乳房 X 射线照片进行分类研究，说明了两个问题。其一，对有显著差异的医学图像使用卷积神经网络训练时非常有效；其二，卷积神经网络对医学图像可以进行多角度分析，并不需要对原始图像进行标准化处理。在应用方面，在 ImageNet 数据集上预训练过的神经网络在检测良、恶性乳腺癌方面 AUC［AUC（area under curve）被定义为 ROC 曲线下与坐标轴围成的面积］超过 0.9。

7.7.3 生成对抗网络在图像处理中的应用

生成对抗网络作为一个具有“无限”生成能力的模型，它的直接应用就是建模，生成与真实数据分布一致的数据样本，如可以生成图像、视频等。GAN 可以用于解决一些传统的机器学习中所面临的标注数据不足时的学习问题，如无监督学习，半监督学习，多视角、多任务学习等。从目前的文献来看，GAN 在图像上的应用主要用于图像修改，包括单图像超分辨率、交互式图像生成、图像编辑、图像到图像的翻译等。

单图像超分辨率 GAN 能够生成与真实数据分布一致的图像。一个典型应用来自推特（Twitter）公司，克里斯坦（Christian）等提出利用 GAN 将一个低分辨率的模糊图像变换为具有丰富细节的高清图像。其团队用 VGG 网络作为判别器，用参数化的残差网络表示生成器，其算法思路如图 7.16 所示，实验结果如图 7.17 所示，可以看到 GAN 生成了细节丰富的图像。

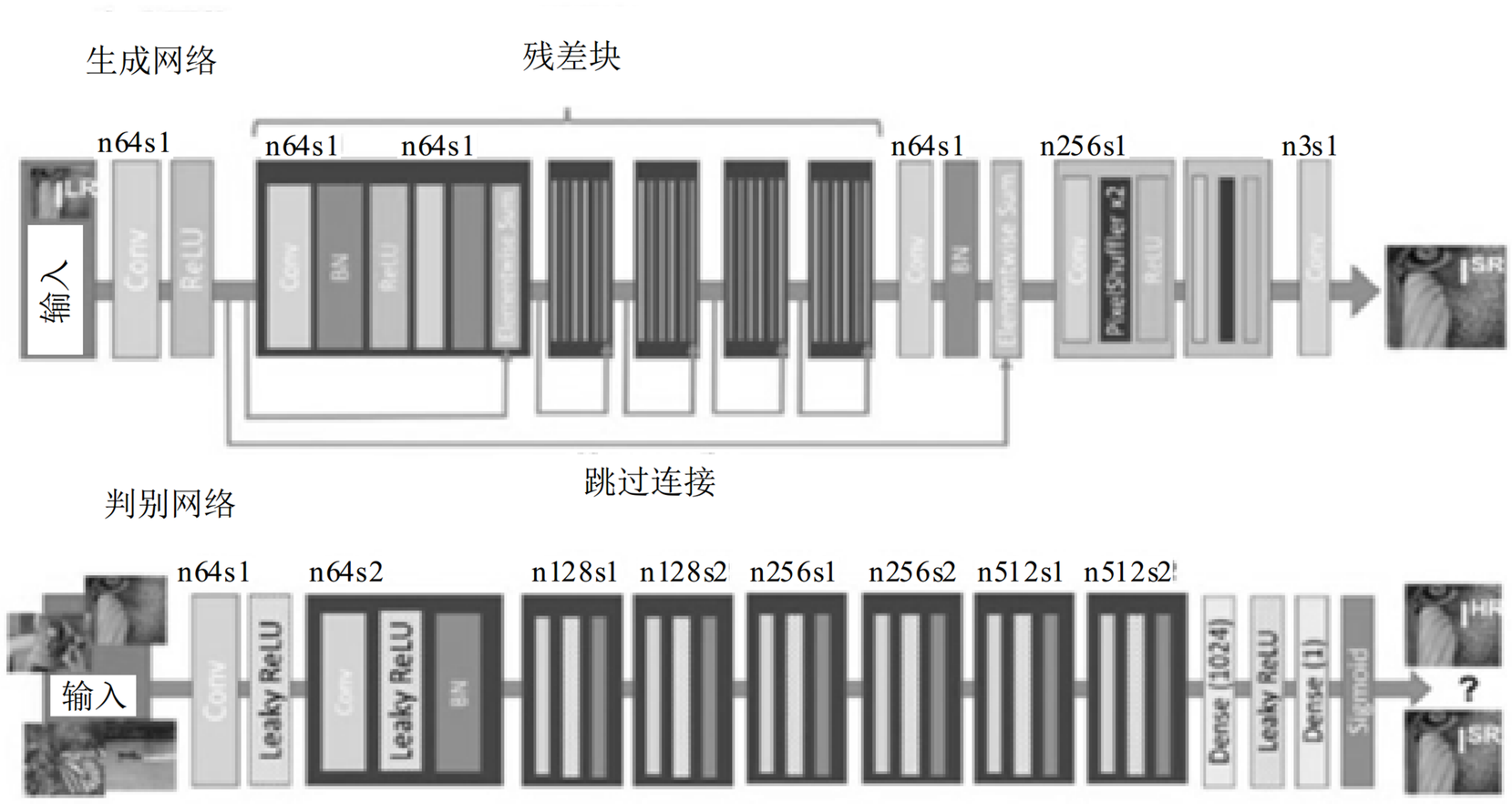


图 7.16　Twitter 图像超分辨率算法示意图

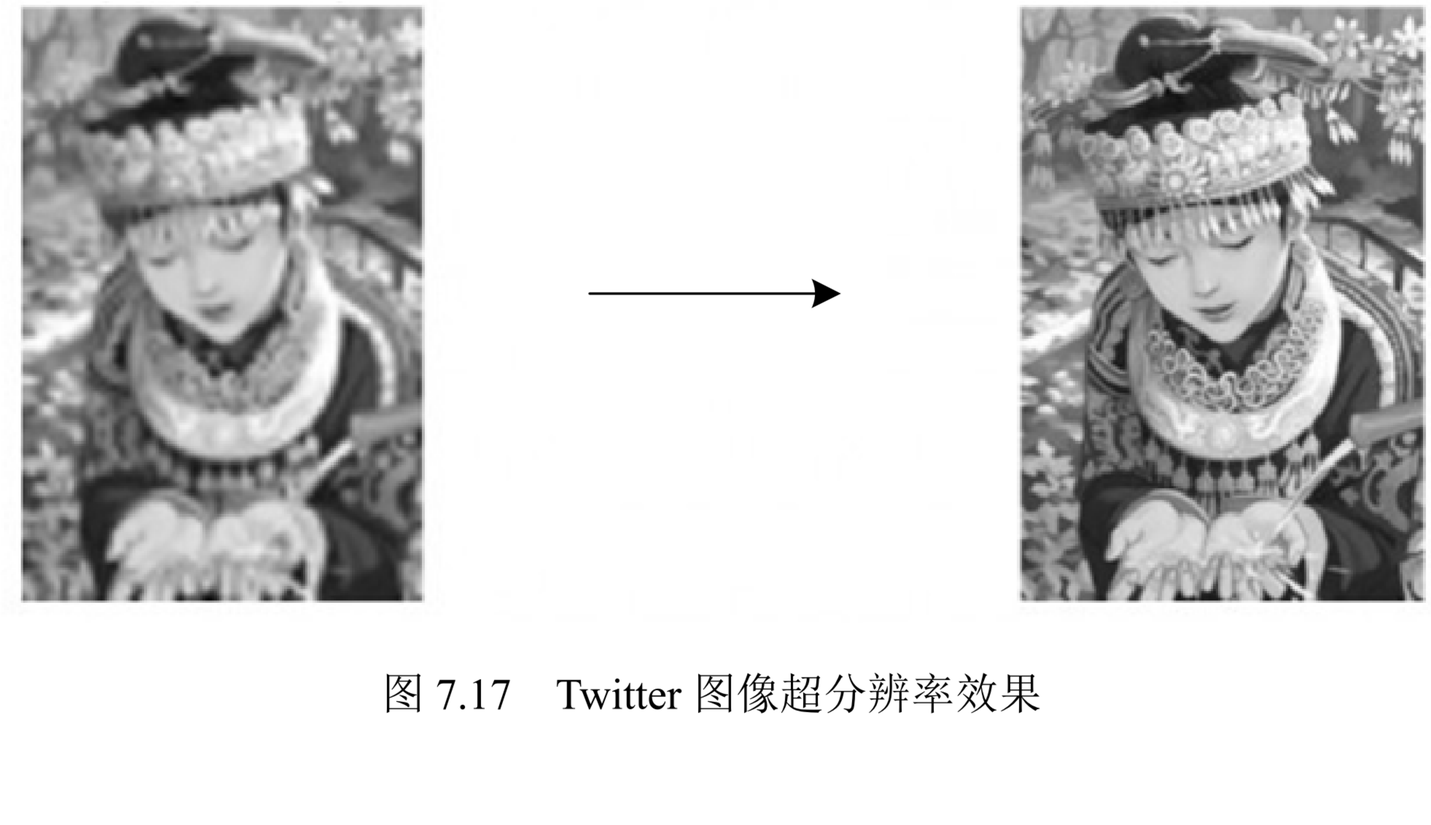


图 7.17　Twitter 图像超分辨率效果

Adobe 公司使用交互式图像生成技术构建了一套图像编辑操作，如图 7.18 所示。经过这些操作以后，图像依旧在“真实图像流形”上，因此编辑后的图像更接近真实图像。

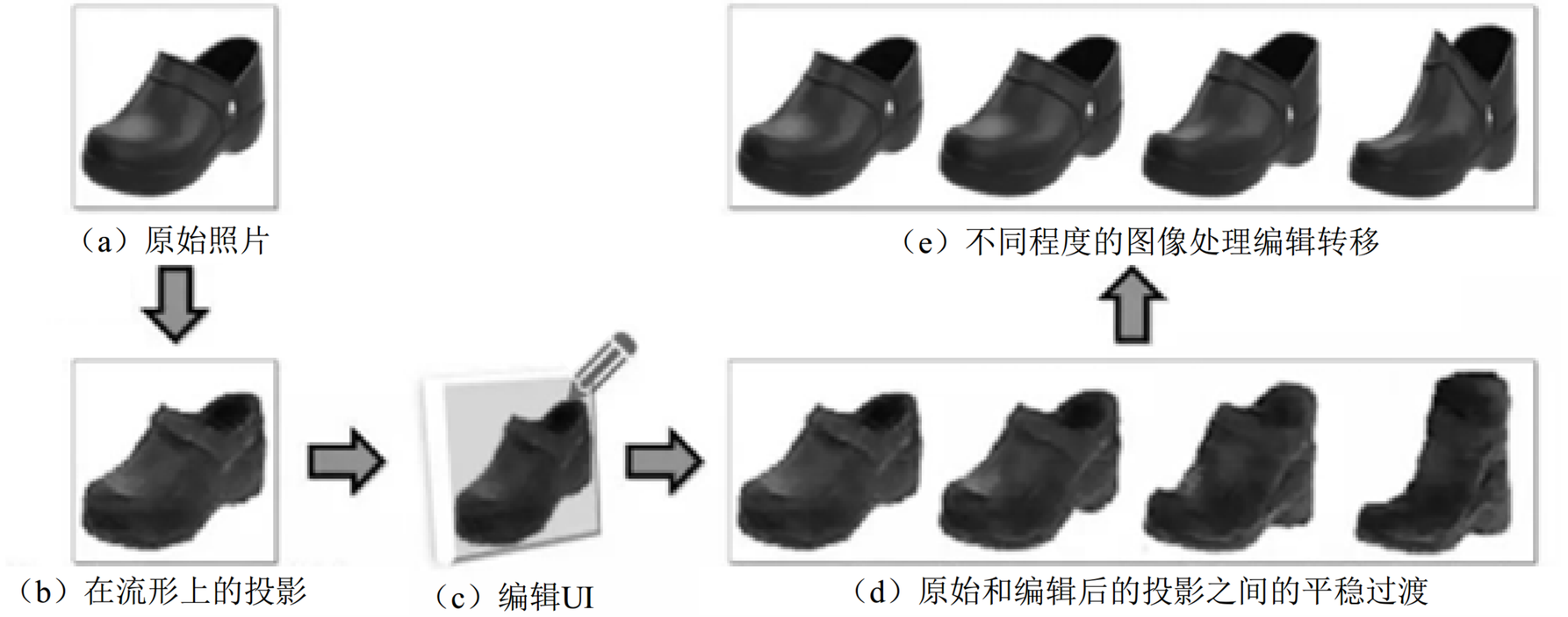


图 7.18　Adobe 图像编辑

具体来说，图像编辑的流程如下。

1）将原始图像投影到低维的隐向量空间。

2）将隐向量空间的值作为输入，利用 GAN 重构图像。

3）利用画笔工具对重构的图像进行修改（颜色、形状等）。

4）将等量的结构、色彩等修改应用到原始图像上。

值得一提的是，交互式图像生成技术保距映射的限制，使整个过程的大部分操作可以转换为求解优化问题，整个修改过程近乎实时。

图像到图像的翻译是指将一种类型的图像转换为另一种类型的图像，如图 7.19 所示。例如，将草图具象化，根据卫星图生成地图，等等。

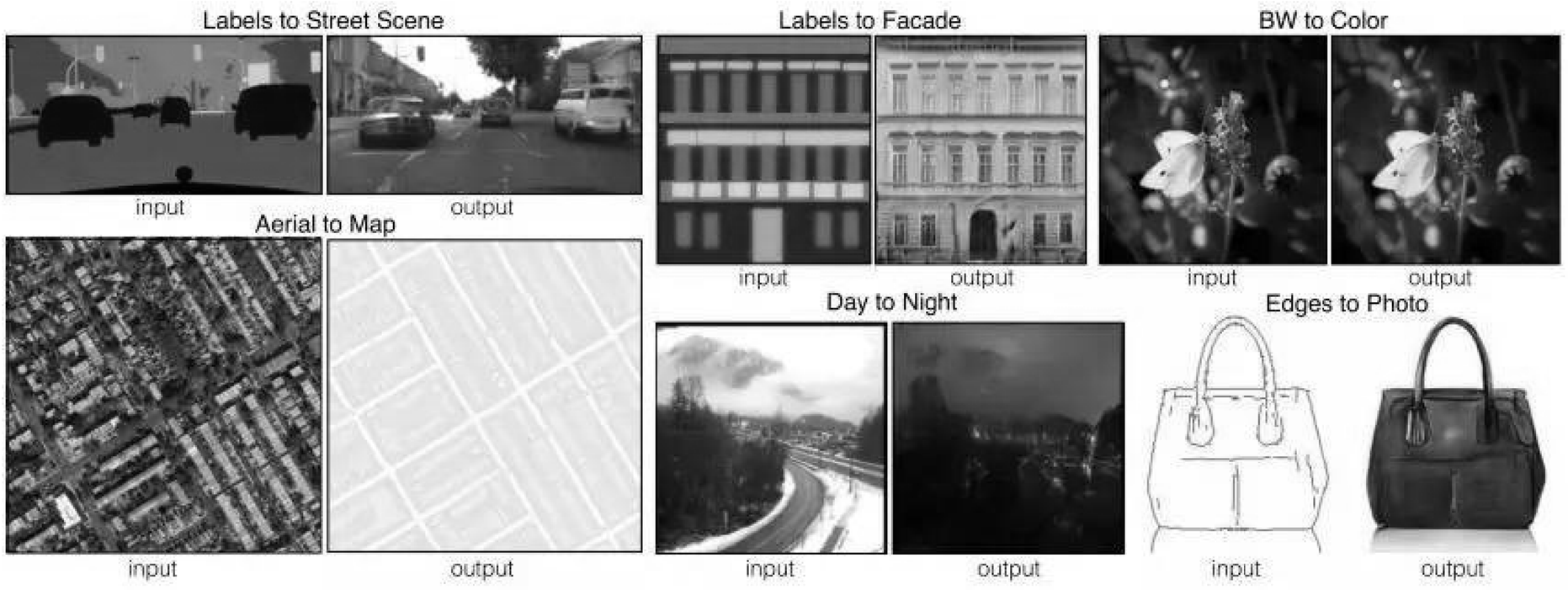


图 7.19　图像到图像的翻译

7.7.4　生成对抗网络在语言处理中的应用

GAN 在自然语言处理上的应用可以分为两类：生成文本；根据文本生成图像。其中，生成文本包括两种：根据隐向量（噪声）生成一段文本；对话生成。

1．生成对话

李纪为等于 2017 年发表的《基于对抗训练的对话生成》（*Adversarial Learning for Neural Dialogue Generation*）显示了 GAN 在对话生成领域的应用。实验效果如图 7.20 所示。可以看出，生成的对话具有一定的相关性，但是效果并不是很好，而且只能做单轮对话。

Input	tell me ... how long have you had this falling sickness ?
Vanilla-MLE	i 'm not a doctor .
Vanilla-Sample	well everything you did was totally untrue .
REINFORCE	i don 't know how long it 's been .
REGS Monte Carlo	A few months, I guess .
Input	so i had the doctors test sammy 's response to conditioning .
Vanilla-MLE	sammy wrote the test sammy wrote the test .
Vanilla-Sample	objects pick objects up objects objects objects objects objects
REINFORCE	i 'm not sure that 's relevant .
REGS Monte Carlo	so he took the pills .
Input	what are you going to do ?
Vanilla-MLE	i 'm going to the movies .
Vanilla-Sample	get him outta here first !
REINFORCE	i 'm going to get you .
REGS Monte Carlo	i 'm going to get you a new car .
Input	they fear your power your intellect .
Vanilla-MLE	you 're the only one who knows what 's going on .
Vanilla-Sample	when they are conquered and you surrender they will control all of us .
REINFORCE	i 'm afraid i 'm not ready yet .
REGS Monte Carlo	i 'm not afraid of your power .

图 7.20　用于神经对话生成的对抗式学习效果图

2．文本到图像的翻译

GAN 也能用于文本到图像的翻译。文本到图像的翻译指 GAN 的输入是描述图像内容的一句话，如“一只有着粉色的胸和冠的小鸟”，那么所生成的图像内容要与这句话所描述的内容相匹配。

在 2016 年的国际机器学习大会上，史考特·瑞德（Scott Reed）等提出了基于条件生成对抗网络（conditional genertive adversarial network，CGAN）的一种解决方案：将文本编码作为发生器（generator）的条件（condition）输入；对于鉴别器（discriminator），文本编码在特定层作为条件信息引入，以辅助判断输入图像是否满足文本描述。其团队提出了两种基于 GAN 的算法：GAN-CLS 和 GAN-INT。

图 7.21 所示为文中所用的 GAN 架构，实验结果如图 7.22 所示，实验发现生成的图像相关性很高。

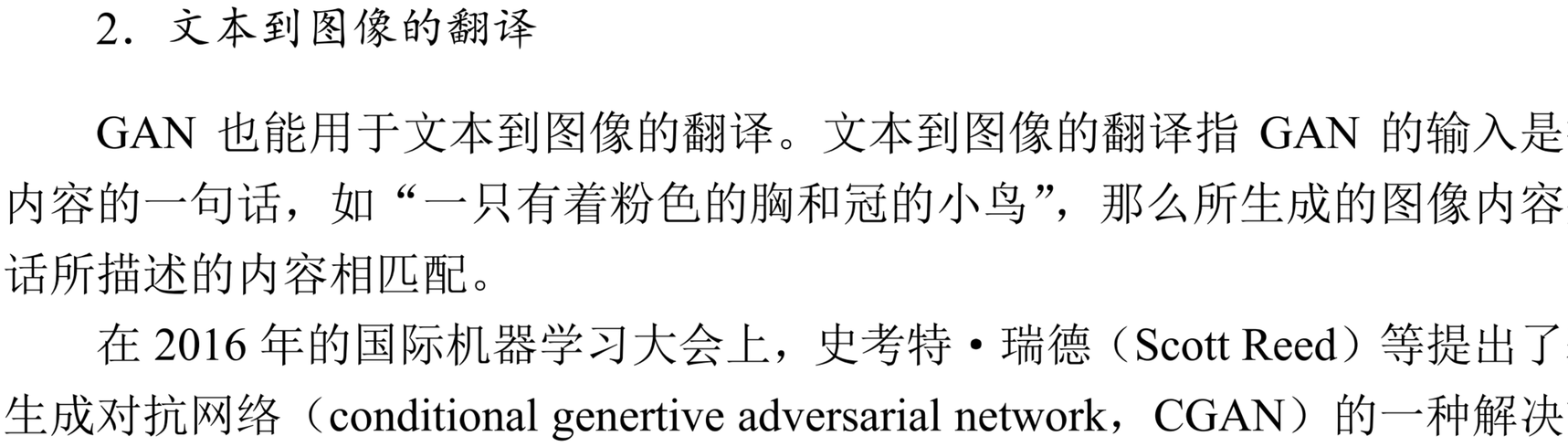


图 7.21　Scott Reed 论文中用到的 GAN 架构

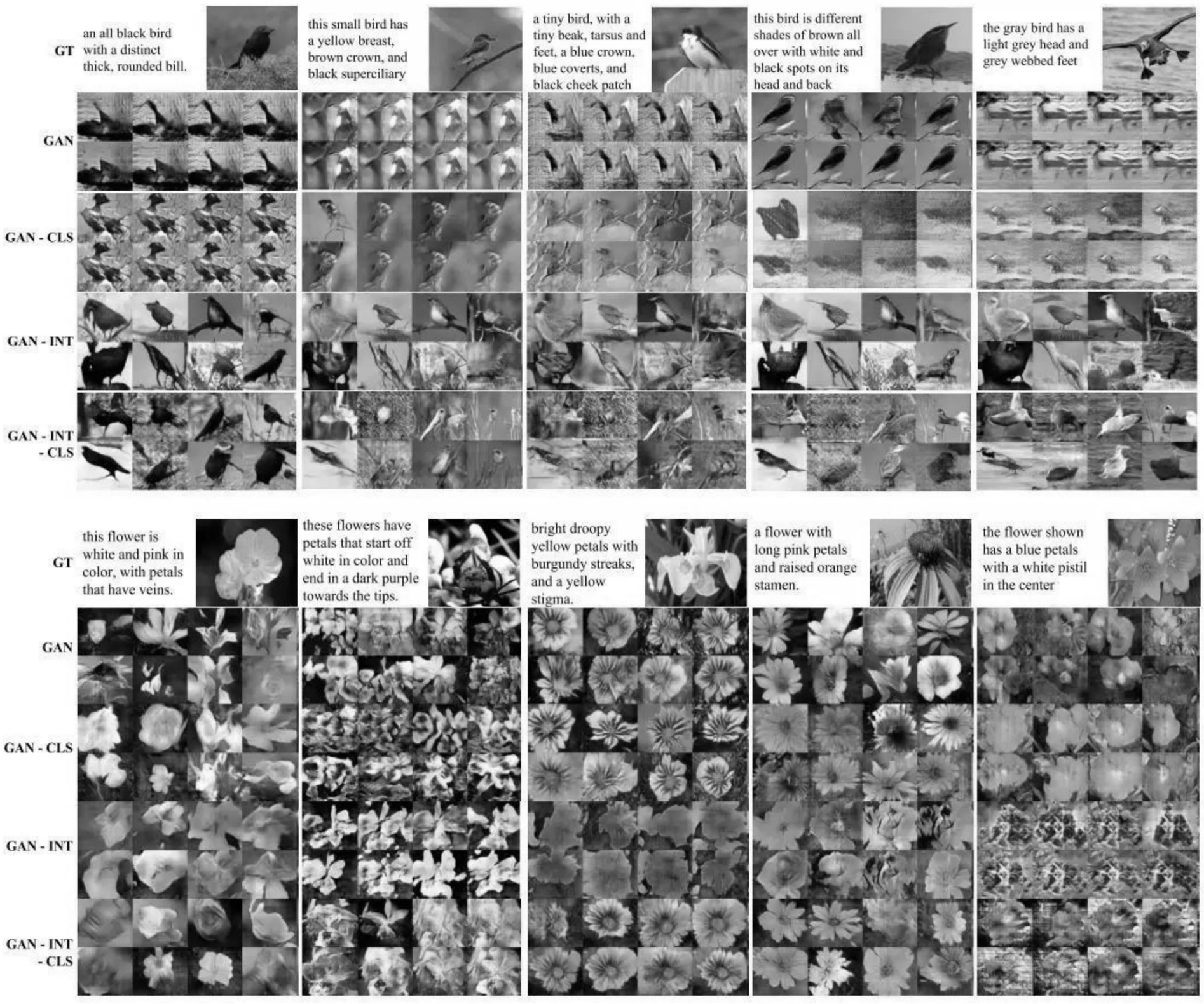


图 7.22　文本到图像的翻译效果图

本 章 小 结

本章首先介绍了神经网络和神经元的相关概念，并介绍了神经元数学模型、神经网络的结构和学习过程，然后介绍了反向传播算法，从概念、结构、应用等方面介绍了卷积神经网络和循环神经网络，还介绍了深度生成模型中的生成对抗网络，最后介绍了深度学习在不同领域的应用。通过本章的学习，读者能够对神经网络和深度学习所涉及的知识有较全面的认识。

本 章 习 题

1．简述人工神经元的特点。

2．简述 LSTM 的结构，并说明它相比传统 RNN 最大的优势是什么。

3．简述梯度消失的原因。

4．分析卷积神经网络中 1×1 的卷积核的作用。

第8章　强 化 学 习

8.1　任务与奖赏

考虑一下如何种西瓜。种瓜有许多步骤，从开始的选种，到定期浇水、施肥、除草、杀虫，经过一段时间后才能收获西瓜。通常要等到收获后，才知道种出的瓜好不好。若将得到好瓜作为辛勤种瓜劳动的奖赏，则在种瓜过程中执行某个操作时，并不能立即获得这个最终奖赏，甚至难以判断当前操作对最终奖赏的影响，仅能得到一个当前反馈（如瓜苗看起来更加健壮了）。因此需要多次种瓜，在种瓜过程中不断摸索，然后才能总结出较好的种瓜策略。这个过程抽象出来，就是强化学习（reinforcement learning）。

图 8.1 给出了强化学习的一个简单图示。强化学习任务通常用马尔可夫决策过程来描述：机器处于环境 E 中，状态空间为 X，其中每个状态 $x \in X$ 是机器感知到的环境的描述，如在种瓜任务中，这就是当前瓜苗长势的描述；机器能够采取的动作构成了动作空间 A，如种瓜过程中有浇水、施不同的肥、使用不同的农药等多种可供选择的动作；若某个动作 $a \in A$ 作用在当前状态 x 上，则潜在的转移函数 P 将使环境从当前状态按某种概率转移到另一个状态，如瓜苗状态为缺水，若选择动作浇水，则瓜苗长势发生变化，瓜苗有一定的概率恢复健康，也有一定的概率无法恢复；在转移到另一个状态的同时，环境会根据潜在的“奖励”函数 R 反馈给机器一个奖赏，如保持瓜苗健康对应奖赏为+1，瓜苗凋零对应奖赏为 -10，最终种出了好瓜对应奖赏为+100。综合起来，强化学习任务对应了四元组 $E=(X,A,P,R)$，其中 $P: X \times A \times X \mapsto R$ 指定了状态转移概率，$R: X \times A \times X \mapsto R$ 指定了奖赏；在有的应用中，奖赏函数可能仅与状态转移有关，即 $R: X \times X \mapsto R$。

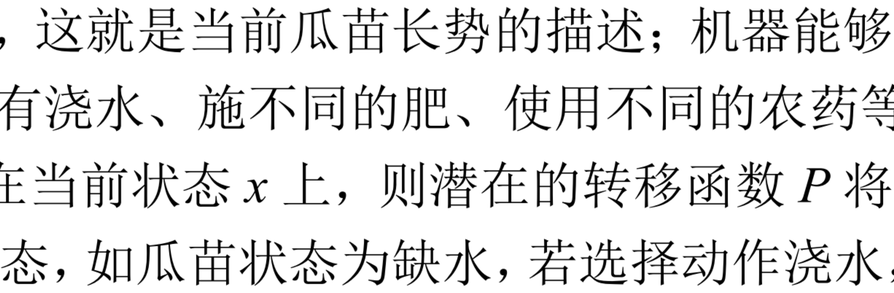


图 8.1　强化学习图示

图 8.2 给出了一个简单的例子：给西瓜浇水问题的马尔可夫决策过程。该任务只有四个状态（健康、缺水、溢水、凋亡）和两个动作（浇水、不浇水），在每一步转移后，若状态是保持瓜苗健康则获得奖赏 1，瓜苗缺水或溢水则奖赏为-1，这时通过浇水或不浇水可以恢复健康状态，当瓜苗凋亡时奖赏是最小值-100 且无法恢复。图中箭头表示

状态转移，箭头旁的 a、p、r 分别表示导致状态转移的动作、转移概率以及返回的奖赏。容易看出，最优策略在“健康”状态选择动作“浇水”，在“溢水”状态选择动作“不浇水”，在“缺水”状态选择动作“浇水”，在“凋亡”状态可选择任意动作。

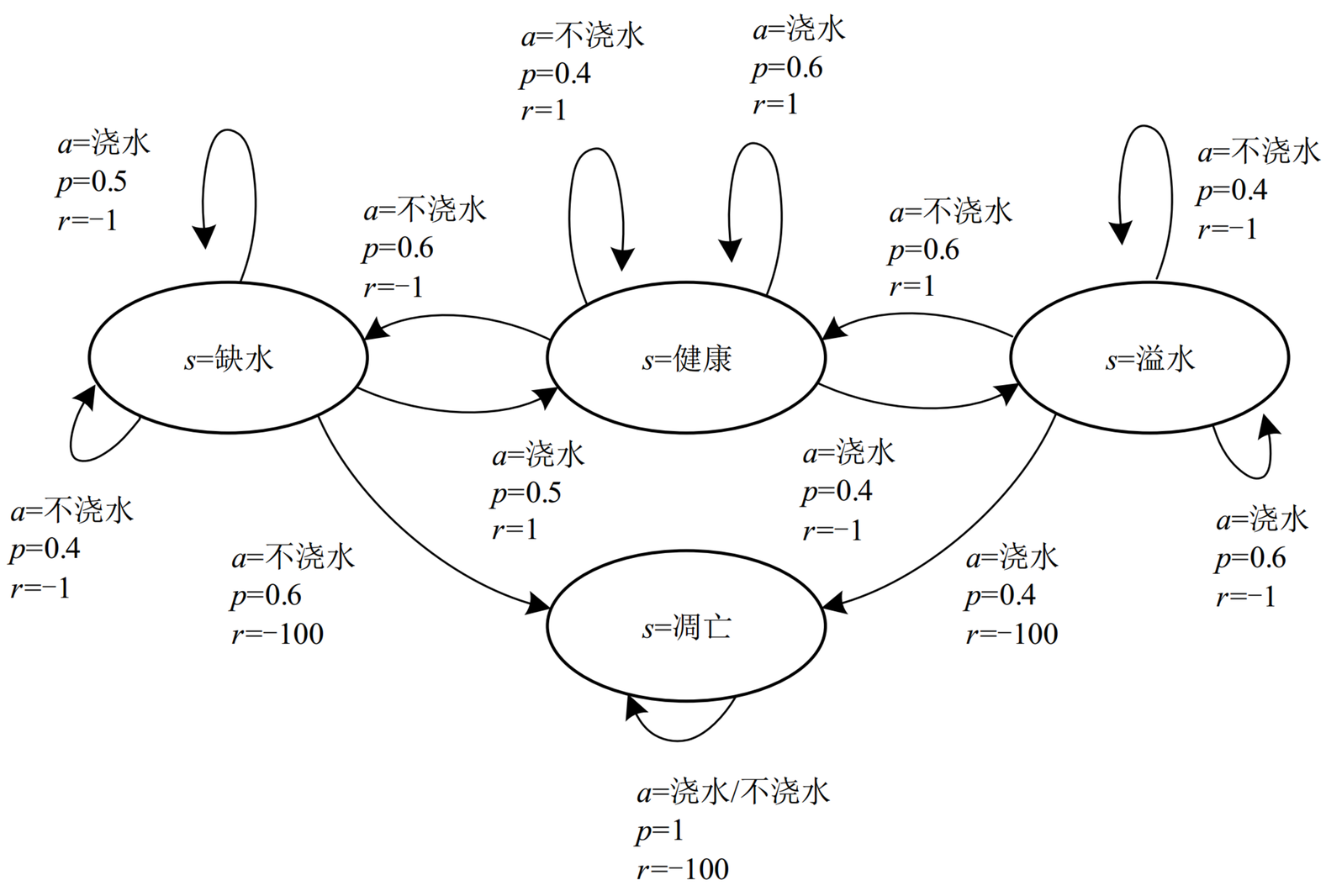


图 8.2　给西瓜浇水问题的马尔可夫决策过程

需要注意“机器”与“环境”的界限。例如，在种西瓜任务中，环境是西瓜生长的自然世界；在下棋博弈中，环境是棋盘与对手；在机器人控制中，环境是机器人的躯体与物理世界。总之，在环境中状态的转移、奖赏的返回是不受机器控制的，机器只能通过选择要执行的动作来影响环境，也只能通过观察转移后的状态和返回的奖赏来感知环境。

机器要做的是通过在环境中不断地尝试来学习一个策略 π，根据这个策略，在状态 x 下就能得知要执行的动作 $a=\pi(x)$。例如，看到瓜苗状态是缺水时，能返回动作“浇水”，策略有两种表示方法：一种是将策略表示为函数 $\pi: X \to A$，确定性策略常用这种表示；另一种是概率表示 $\pi: XxA \to R$，随机性策略常用这种表示。$\pi(x,a)$ 为状态 x 下选择动作 a 的概率，这里必须有 $\sum_{a}\pi(x,a)=1$。

策略的优劣取决于长期执行这一策略后得到的累积奖赏。例如，某个策略使瓜苗枯死，它的累积奖赏会很小，另一个策略种出了好瓜，它的累积奖赏就会很大。在强化学习任务中，学习的目的就是要找到能使长期累积奖赏最大化的策略长期累积奖赏有多种计算方式，常用的有 T 步累积奖赏 $E\left[\frac{1}{T}\sum_{t=1}^{T} r_t\right]$ 和 γ 折扣累积奖赏 $E\left[\sum_{t=0}^{+\infty}\gamma^t r_{t+1}\right]$。其中，$r_t$ 表示第 t 步获得的奖赏值，E 表示对所有随机变量求期望值。

读者也许已经感觉到强化学习与监督学习的差别，若将这里的“状态”对应为监督

学习中的“示例”，“动作”对应为“标记”，则可看出强化学习中的“策略”实际上就相当于监督学习中的“分类器”（当动作是离散的）或“回归器”（当动作是连续的），模型的形式并无差别。不同的是，在强化学习中并没有监督学习中的有标记样本（即“示例-标记”对）。换言之，没有人直接告诉机器在什么状态下应该做什么动作，只有等到最终结果揭晓，才能通过“反思”之前的动作是否正确来进行学习。因此，强化学习在某种意义上可看作具有“延迟标记信息”的监督学习问题。

8.2　k-摇臂赌博机

8.2.1　探索与利用

与一般监督学习不同，强化学习任务的最终奖赏是在多步动作之后才能观察到，这里不妨先考虑比较简单的情形——最大化单步奖赏，即仅考虑一步操作。需要注意的是，即便在这样的简化情形下，强化学习仍与监督学习有显著不同，因为机器需要通过尝试来发现各个动作产生的结果，而没有训练数据告诉机器应当做哪个动作。

欲最大化单步奖赏需要考虑两个方面：一是需要知道每个动作带来的奖赏，二是执行奖赏最大的动作。若每个动作对应的奖赏是一个确定值，那么尝试一遍所有的动作便能找出奖赏最大的动作。然而，更一般的情形是，一个动作的奖赏值是来自一个概率分布，仅通过一次尝试并不能确切地获得平均奖赏值。

实际上，单步强化学习任务对应一个理论模型，即 k-摇臂赌博机（k-armed bandit），如图 8.3 所示。k-摇臂赌博机有 k 个摇臂，赌徒在投入一个硬币后可选择按下其中一个摇臂，每个摇臂以一定的概率吐出硬币，但是赌徒并不知道这个概率，赌徒的目标是通过一定的策略最大化自己的奖赏，即获得最多的硬币。

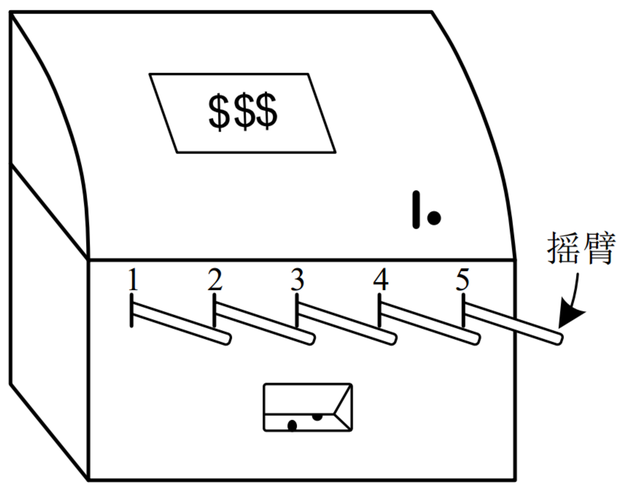


图 8.3　k-摇臂赌博机图示

若仅为获知每个摇臂的期望奖赏，则可采用“仅探索（exploration-only）”法：将所有的尝试机会平均分配给每个摇臂（即轮流按下每个摇臂），最后以每个摇臂各自的平均吐币概率作为其奖赏期望的近似估计；若仅为执行奖赏最大的动作，则可采用“仅利用（exploitation-only）”法：按下目前最优的（即到目前为止平均奖赏最大的）摇臂，若有多个摇臂同为最优，则从中随机选取一个。显然，“仅探索”法能够很好地估计每个摇臂的奖赏，却会失去很多选择最优摇臂的机会；“仅利用”法则相反，它没有很好

地估计摇臂期望奖赏，很可能经常选不到最优摇臂。因此，这两种方法都难以使最终的累积奖赏最大化。

事实上，探索（即估计摇臂的优劣）和利用（即选择当前最优摇臂）这两者是矛盾的，因为尝试次数（即总投币数）有限，加强了一方自然会削弱另一方，这就是强化学习所面临的探索-利用窘境（exploration-exploitation dilemma）。显然，欲累积奖赏最大，则必须在探索与利用之间达成较好的折中。

8.2.2 ε-贪心

ε-贪心法基于一个概率来对探索和利用进行折中：每次尝试时，以 ε 的概率进行探索，即以均匀概率随机选取一个摇臂；以 $1-\varepsilon$ 的概率进行利用，即选择当前平均奖赏最高的摇臂（若有多个，则随机选取一个）。

令 $Q(k)$ 记录摇臂 k 的平均奖赏，若摇臂 k 被尝试了 n 次，得到的奖赏为 $v_1,v_2,\cdots,v_n$，则平均奖赏为

$$Q(k)=\frac{1}{n}\sum_{i=1}^{n}v_i \tag{8.1}$$

若直接根据式（8.1）计算平均奖赏，则需要记录 n 个奖赏值。显然，更高效的做法是对均值进行增量式计算，即每尝试一次就立即更新 $Q(k)$，可以用下标来表示尝试的次数。初始时 $Q_0(k)=0$，对于任意的 $n>1$，若第 $n-1$ 次尝试后的平均奖赏为 $Q_{n-1}(k)$，则在经过第 n 次尝试获得奖赏 v_n 后，平均奖赏应更新为

$$\begin{aligned}Q_n(k)&=\frac{1}{n}\left[(n-1)\times Q_{n-1}(k)+v_n\right]\\&=Q_{n-1}(k)+\frac{1}{n}\left[v_n-Q_{n-1}(k)\right]\end{aligned} \tag{8.2}$$

这样，无论摇臂被尝试多少次，都仅需要记录这两个值：已尝试次数 $n-1$ 和最近平均奖赏 $Q_{n-1}(k)$。

若摇臂奖赏的不确定性较大，如概率分布较宽时，则需更多的探索，此时，需要较大的 ε 值；若摇臂的不确定性较小，如概率分布较集中时，则少量的尝试就能很好地近似真实奖赏，此时需要的 ε 较小。通常令 ε 取一个较小的常数，如 0.1 或 0.01。然而，若尝试次数非常大，那么一段时间后，奖赏都能很好地近似出来，不再需要探索，这种情况下可让 ε 随着尝试次数的增加而逐渐减小，如令 $\varepsilon=\frac{1}{\sqrt{t}}$。

8.2.3 Softmax

Softmax 算法基于当前已知的摇臂平均奖赏来对探索和利用进行折中。若各摇臂的平均奖赏相当，则选取各摇臂的概率也相当；若某些摇臂的平均奖赏明显高于其他摇臂，则它们被选取的概率也明显更高。

Softmax 算法中摇臂的分配是居于玻尔兹曼分布的，用公式表示如下。

$$P(k)=\frac{\mathrm{e}^{\frac{Q(k)}{r}}}{\sum_{i=1}^{K}\mathrm{e}^{\frac{Q(i)}{r}}} \tag{8.3}$$

式中，$Q(i)$记录当前摇臂的平均奖赏。$r>0$ 称为“温度”，r 越小则平均奖赏高的摇臂被选取的概率越高。r 趋于 0 时 Softmax 趋于“仅利用”，r 趋于无穷大时 Softmax 趋于“仅探索”。

ε-贪婪算法与 Softmax 算法孰优孰劣，主要取决于具体应用。为了更直观地观察它们的差别，考虑一个简单的例子：假定 2-摇臂赌博机的摇臂 1 以 0.4 的概率返回奖赏 1，以 0.6 的概率返回奖赏 0；摇臂 2 以 0.2 的概率返回奖赏 1，以 0.8 的概率返回奖赏 0。图 8.4 显示了不同算法在不同参数下的平均累积奖赏，其中每条曲线对应于重复 1000 次实验的平均结果。可以看出，Softmax（r=0.01）的曲线与“仅利用”的曲线几乎重合。

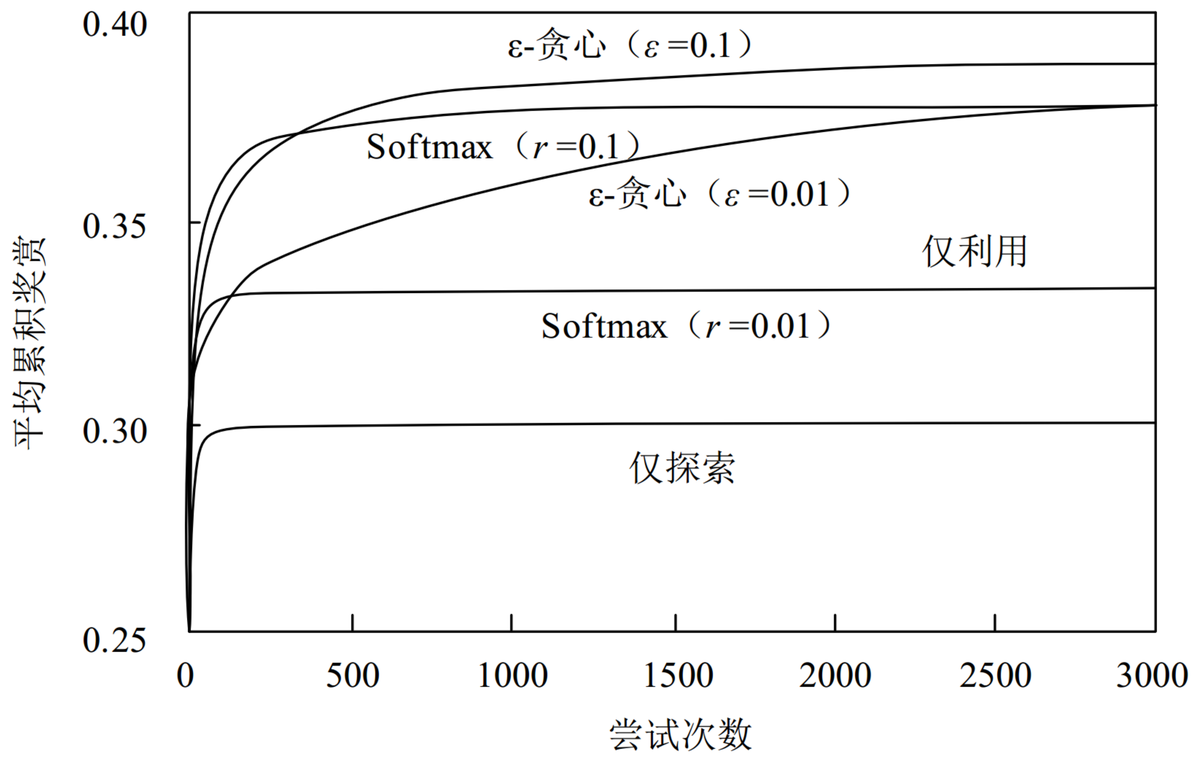


图 8.4　不同算法在 2-摇臂机上的性能比较

对于离散状态空间、离散动作空间上的多步强化学习任务，一种直接的办法是将每个状态上动作的选择看作一个 k-摇臂赌博机问题，用强化学习任务的累积奖赏来代替 k-摇臂赌博机算法中的奖赏函数，即可将赌博机算法用于每个状态：对每个状态分别记录各动作的尝试次数、当前平均累积奖赏等信息，基于赌博机算法选择要尝试的动作。然而，这样的做法有很多局限，因为它没有考虑强化学习任务马尔可夫决策过程的结构。

8.3　有模型学习

考虑多步强化学习任务，暂且先假定任务对应的马尔可夫决策过程四元组 $E=(X,A,P,R)$ 均为已知，这样的情形称为模型已知，即机器已对环境进行了建模，能在机器内部模拟出与环境相同或近似的状况。在已知模型的环境中学习称为有模型学习（model-based learning）。此时，对于任意状态 x、x' 和动作 a，在 x 状态下执行动作 a 转移到 x' 状态的概率 $P^{a}_{x\to x'}$ 是已知的，该转移所带来的奖赏 $R^{a}_{x\to x'}$ 也是已知的。为便于讨

论，假设状态空间 X 和动作空间 A 均为有限。

8.3.1 策略评估

在模型已知时，对任意策略 π 能估计出该策略带来的期望累积奖赏。令函数 $V^{\pi}(x)$ 表示从状态 x 出发，使用策略 π 所带来的累积奖赏；函数 $Q^{\pi}(x,a)$ 表示从状态 x 出发，执行动作 a 后再使用策略 π 带来的累积奖赏。这里的 $V(\cdot)$ 称为状态值函数（state value function），$Q(\cdot)$ 称为状态-动作值函数（state-action value function），分别表示指定状态上以及指定状态-动作上的累积奖赏。

由累积奖赏的定义，有如下状态值函数

$$\begin{cases} V_T^{\pi}(x) = E_{\pi}\left[\dfrac{1}{T}\sum\limits_{t=1}^{T} r_t \mid x_0 = x\right], & T\text{步累积奖赏} \\ V_{\gamma}^{\pi}(x) = E_{\pi}\left[\sum\limits_{t=0}^{+\infty} \gamma^t r_{t+1} \mid x_0 = x\right], & \gamma\text{折扣累积奖赏} \end{cases} \tag{8.4}$$

为叙述简洁，后面在涉及上述两种累积奖赏时，不再说明奖赏类别，从上下文便可轻易地判知。令 x_0 表示起始状态，a_0 表示起始状态上采取的第一个动作；对于 T 步累积奖赏，用下标 t 表示后续执行的步数。有如下状态-动作值函数

$$\begin{cases} Q_T^{\pi}(x,a) = E_{\pi}\left[\dfrac{1}{T}\sum\limits_{t=1}^{T} r_t \mid x_0 = x, a_0 = a\right] \\ Q_{\gamma}^{\pi}(x,a) = E_{\pi}\left[\sum\limits_{t=0}^{+\infty} \gamma^t r_{t+1} \mid x_0 = x, a_0 = a\right] \end{cases} \tag{8.5}$$

由于 MDP 具有马尔可夫性质，即系统下一时刻的状态仅由当前时刻的状态决定，不依赖以往任何状态，于是值函数有很简单的递归形式。对于 T 步累积奖赏有

$$\begin{aligned} V_T^{\pi}(x) &= E_{\pi}\left[\frac{1}{T}\sum_{t=1}^{T} r_t \mid x_0 = x\right] \\ &= E_{\pi}\left[\frac{1}{T} r_1 + \frac{T-1}{T}\frac{1}{T-1}\sum_{t=2}^{T} r_t \mid x_0 = x\right] \\ &= \sum_{a\in A}\pi(x,a)\sum_{x'\in X} P_{x\to x'}^{a}\left(\frac{1}{T}R_{x\to x'}^{a} + \frac{T-1}{T}\right) E_{\pi}\left[\frac{T}{T-1}\sum_{t=1}^{T-1} r_t \mid x_0 = x'\right] \\ &= \sum_{a\in A}\pi(x,a)\sum_{x'\in X} P_{x\to x'}^{a}\left(\frac{1}{T}R_{x\to x'}^{a} + \frac{T-1}{T}V_{T-1}^{\pi}(x')\right) \end{aligned} \tag{8.6}$$

类似地，对于 γ 折扣累积奖赏有

$$V_{\gamma}^{\pi}(x) = \sum_{a\in A}\pi(x,a)\sum_{x'\in X} P_{x\to x'}^{a}\left(R_{x\to x'}^{a} + \gamma V_{\gamma}^{\pi}(x')\right) \tag{8.7}$$

需要注意的是，正是由于 P 和 R 已知，才可以进行全概率展开。

用上面的递归等式来计算值函数，实际上是一种动态规划算法。对于 V_T^{π}，可设想

递归已知进行下去，直到最初的起点；换言之，从值函数的初始值 V_0^π 出发，通过一次迭代能够计算出每个状态的单步奖赏 V_1^π，进而从单步奖赏出发，通过一次迭代计算出两步累积奖赏 V_2^π，以此类推。对于 T 步累积奖赏，只需迭代 T 轮就能精确地求出值函数。

对于 $V_\gamma^\pi(x)$，由于 γ^t 在 t 很大时趋于 0，因此也能使用类似的算法，只需根据式(8.7)进行替换。此外，由于算法可能会迭代很多次，因此需要设计一个停止准则。常见的是设置一个阈值 θ，若在执行一次迭代后值函数的改变小于 θ 则算法停止；相应地，可使用式（8.8）作为推出条件的判断：

$$\max_{x\in X}\left|V(x)-V'(x)\right|<\theta \tag{8.8}$$

有了状态值函数 V，就能直接计算出状态-动作值函数：

$$\begin{cases} Q_T^\pi(x,a)=\sum\limits_{x'\in X}P_{x\to x'}^a\left(\dfrac{1}{T}R_{x\to x'}^a+\dfrac{T-1}{T}V_{T-1}^\pi(x')\right) \\ Q_\gamma^\pi(x,a)=\sum\limits_{x'\in X}P_{x\to x'}^a\left(R_{x\to x'}^a+\gamma V_\gamma^\pi(x')\right) \end{cases} \tag{8.9}$$

8.3.2　策略改进

对某个策略的累积奖赏进行评估后，若发现它并非最优策略，则对其进行改进。理想的策略应能最大化累积奖赏：

$$\pi^*=\arg\max_\pi\sum_{x\in X}V^\pi(x) \tag{8.10}$$

一个强化学习任务可能有多个最优策略，最优策略对应的值函数 V 称为最优值函数，即

$$\forall x\in X:V^*(x)=V^{\pi^*}(x) \tag{8.11}$$

注意，当策略空间无约束时式（8.11）的 V^* 才是最优策略对应的值函数。例如，对离散状态空间和离散动作空间，策略空间是所有状态上所有动作的组合，共有 $|A|^{|X|}$ 种不同的策略。若策略空间有约束，则违背约束的策略是“不合法”的，即便其值函数所取得的累积奖赏值最大，也不能作为最优值函数。

由于最优值函数的累积奖赏值已达最大，因此可对前面的 Bellman 等式[即式（8.6）和式（8.7）]做一个改动，即将对动作的求和改为取最优：

$$\begin{cases} V_T^*(x)=\max\limits_{a\in A}\sum\limits_{x'\in X}P_{x\to x'}^a\left(\dfrac{1}{T}R_{x\to x'}^a+\dfrac{T-1}{T}V_{T-1}^*(x')\right) \\ V_\gamma^*(x)=\max\limits_{a\in A}\sum\limits_{x'\in X}P_{x\to x'}^a\left(R_{x\to x'}^a+\gamma V_\gamma^*(x')\right) \end{cases} \tag{8.12}$$

换言之，将

$$V^*(x)=\max_{a\in A}Q^{\pi^*}(x,a) \tag{8.13}$$

代入式（8.9）可得最优状态-动作值函数：

$$\begin{cases} Q_T^*(x,a)=\sum\limits_{x'\in X}P_{x\to x'}^a\left(\dfrac{1}{T}R_{x\to x'}^a+\dfrac{T-1}{T}\max\limits_{a'\in A}Q_{T-1}^*(x',a')\right) \\ Q_\gamma^*(x,a)=\sum\limits_{x'\in X}P_{x\to x'}^a\left(R_{x\to x'}^a+\gamma\max\limits_{a'\in A}Q_\gamma^*(x',a')\right) \end{cases} \tag{8.14}$$

上述关于最优值函数的等式，称为最优贝尔曼等式，其唯一解是最优值函数。

最优 Bellman 等式揭示了非最优策略的改进方式：将策略选择的动作改变为当前最优的动作。显然，这样的改变能够使策略更好。令动作改变后对应的策略为 π'，改变动作的条件为 $Q^\pi(x,\pi'(x))\geqslant V^\pi(x)$，以 γ 折扣累积奖赏为例，由式（8.9）可计算出递推不等式如下：

$$\begin{aligned} V^\pi(x)&\leqslant Q^\pi\left(x,\pi'(x)\right) \\ &=\sum_{x'\in X}P_{x\to x'}^{\pi'(x)}\left(R_{x\to x'}^{\pi'(x)}+\gamma V^\pi(x')\right) \\ &\leqslant\sum_{x'\in X}P_{x\to x'}^{\pi'(x)}\left(R_{x\to x'}^{\pi'(x)}+\gamma Q^\pi\left(x',\pi'(x')\right)\right) \\ &=\cdots \\ &=V^{\pi'}(x) \end{aligned} \tag{8.15}$$

值函数对于策略的每一点改进都是单调递增的，因此对于当前策略 π，可放心地将其改进为

$$\pi'(x)=\arg\max_{a\in A}Q^\pi(x,a) \tag{8.16}$$

直到 π' 与 π 一致且不再发生变化，此时就满足了最优 Bellman 等式，即找到了最优策略。

8.3.3 策略迭代与值迭代

由前两小节可知，如何评估一个策略的值函数，以及在策略评估后如何改进至最优策略。显然，将这两者结合起来即可得到求解最优解的方法：从一个初始策略（通常是随机策略）出发，先进行策略评估，然后改进策略，评估改进的策略，再进一步改进策略……不断迭代进行策略评估和改进，直到策略收敛、不再改变为止。这样的做法称为策略迭代（policy iteration）。

T 步累积奖赏的策略迭代就是在基于 T 步累积奖赏策略评估的基础上，加入策略改进形成的策略迭代算法。类似地，可得到基于 γ 折扣累积奖赏的策略迭代算法。策略迭代算法在每次改进策略后都需重新进行策略评估，这通常比较耗时。

由式（8.15）可知，策略改进与值函数的改进是一致的，因此可将策略改进视为值函数的改善，即由式（8.12）可得

$$\begin{cases} V_T(x)=\max\limits_{a\in A}\sum\limits_{x'\in X}P_{x\to x'}^a\left(\dfrac{1}{T}R_{x\to x'}^a+\dfrac{T-1}{T}V_{T-1}(x')\right) \\ V_\gamma(x)=\max\limits_{a\in A}\sum\limits_{x'\in X}P_{x\to x'}^a\left(R_{x\to x'}^a+\gamma V_\gamma(x')\right) \end{cases} \tag{8.17}$$

于是可得到值迭代（value iteration）算法。

8.4　免模型学习

在现实的强化学习任务中，环境的转移概率、奖赏函数往往很难得知，甚至很难知道环境中一共有多少状态。若学习算法不依赖环境建模，则称为免模型学习（model-free learning），这比有模型学习要困难得多。

8.4.1　蒙特卡罗强化学习

在免模型情形下，策略迭代算法首先遇到的问题是策略无法评估，这是由于模型未知而导致无法做全概率展开。此时，只能通过在环境中执行选择的动作来观察转移的状态和得到的奖赏。受 k-摇臂赌博机的启发，一种直接的策略评估替代方法是多次“采样”，然后求取平均累积奖赏来作为期望累积奖赏的近似，称为蒙特卡罗强化学习。由于采样必须为有限次数，因此该方法更适合使用 T 步累积奖赏的强化学习任务。

另一方面，策略迭代算法估计的是状态值函数 V，而最终的策略是通过状态-动作值函数 Q 来获得。当模型已知时，从 V 到 Q 有简单的转换方法，而当模型未知时，也会出现困难。于是，将估计对象从 V 转变为 Q，即估计每一对状态-动作的值函数。

此外，在模型未知的情形下，机器只能从一个起始状态（或起始状态集合）开始探索环境，而策略迭代算法由于需要对每个状态分别进行估计，因此在这种情形下无法实现。例如，探索种瓜的过程只能从播下种子开始，而不能任意选择种植过程中的一个状态开始。因此，只能在探索的过程中逐渐发现各个状态并估计各状态-动作对的值函数。

综合起来，在模型未知的情形下，从起始状态出发，使用某种策略进行采样，执行该策略 T 步并获得轨迹

$$\langle x_0, a_0, r_1, x_1, a_1, r_2, \cdots, x_{T-1}, a_{T-1}, r_T, x_T \rangle$$

然后，对轨迹中出现的每一对状态-动作，记录其后的奖赏之和，作为该状态-动作对的一次累积奖赏采样值。多次采样得到多条轨迹后，将每个状态-动作对的累积奖赏采样值进行平均，即得到状态-动作值函数的估计。

可以看出，欲较好地获得值函数的估计，就需要多条不同的采样轨迹。然而，策略有可能是确定的，即对于某个状态只会输出一个动作，若使用这样的策略进行采样，则只能得到多条相同的轨迹。这与 k-摇臂赌博机的“仅利用”法面临相同的问题。因此可借鉴探索与利用折中的办法，如使用 ε-贪心法，以 ε 的概率从所有动作中均匀随机选取一个，以 $1-\varepsilon$ 的概率选取当前最优动作。将确定性的策略 π 称为原始策略，在原始策略上使用 ε-贪心法的策略记为

$$\pi^{\varepsilon}(x)=\begin{cases}\pi(x), & \text{以概率} 1-\varepsilon \\ A\text{中以均匀概率选取的动作}, & \text{以概率} \varepsilon\end{cases} \tag{8.18}$$

对于最大化值函数的原始策略 $\pi=\arg\max_a Q(x,a)$，其 ε-贪心策略 π^{ε} 中，当前最优

动作被选中的概率是$1-\varepsilon+\frac{\varepsilon}{|A|}$，而每个非最优动作被选中的概率是$\frac{\varepsilon}{|A|}$，于是，每个动作都有可能被选取，多次采样将会产生不同的采样轨迹。

与策略迭代算法类似，使用蒙特卡罗方法进行策略评估后，同样要对策略进行改进。前面在讨论策略改进时利用式（8.15）揭示的单调性，通过换入当前最优动作来改进策略。对于任意原始策略π，其 ε-贪心策略π^{ε}仅是将ε的概率均匀分配给所有动作，因此对于最大化值函数的原始策略π'，同样有$Q^{\pi}\left(x,\pi'(x)\right)\geqslant V^{\pi}(x)$，于是式（8.15）仍成立，即可以使用同样的方法进行策略改进。

同策略蒙特卡罗强化学习算法最终产生的是 ε-贪心策略。然而，引入 ε-贪心是为了便于策略评估，而不是为了最终使用，实际上希望改进的是原始（非 ε-贪心）策略。那么，能否仅在策略评估时引入 ε-贪心，而在策略改进时却改进原始策略呢？

这其实是可行的。不妨用两个不同的策略π和π'来产生采样轨迹，两者的区别在于每个状态-动作对被采样的概率不同。一般地，函数f在概率分布p下的期望可表示为

$$E[f]=\int_x p(x)f(x)\mathrm{d}x \tag{8.19}$$

可通过从概率分布p上的采样$\{x_1,x_2,\cdots,x_m\}$来估计f的期望，即

$$\hat{E}[f]=\frac{1}{m}\sum_{i=1}^{m}f(x) \tag{8.20}$$

若引入另一个分布q，则函数在概率分布p下的期望也可等价地写为

$$E[f]=\int_x q(x)\frac{p(x)}{q(x)}f(x)\mathrm{d}x \tag{8.21}$$

式(8.21)可看作$\frac{p(x)}{q(x)}f(x)$在分布q下的期望，因此通过在q上的采样$\{x_1',x_2',\cdots,x_m'\}$可估计为

$$\hat{E}[f]=\frac{1}{m}\sum_{i=1}^{m}\frac{p(x_i')}{q(x_i')}f(x') \tag{8.22}$$

使用策略π的采样轨迹来评估策略π，实际上就是对累积奖赏估计期望

$$Q(x,a)=\frac{1}{m}\sum_{i=1}^{m}R_i \tag{8.23}$$

式中，R_i表示第i条轨迹上自状态x至结束的累积奖赏。若改用策略π'的采样轨迹来评估策略π，则仅需要对累积奖赏加权，即

$$Q(x,a)=\frac{1}{m}\sum_{i=1}^{m}\frac{P_i^{\pi}}{P_i^{\pi'}}R_i \tag{8.24}$$

式中，P_i^{π}和$P_i^{\pi'}$分别表示两个策略产生第i条轨迹的概率。对于给定的一条轨迹$\langle x_0,a_0,r_1,\cdots,x_{T-1},a_{T-1},r_T,x_T\rangle$，策略$\pi$产生该轨迹的概率为

$$P^{\pi}=\prod_{i=0}^{T-1}\pi(x_i,a_i)P_{x_i\to x_{i+1}}^{a_i} \tag{8.25}$$

虽然这里用到了环境的转移概率 $P_{x_i \to x_{i+1}}^{a_i}$，但是式（8.22）中实际只需两个策略概率的比值

$$\frac{P^{\pi}}{P^{\pi'}} = \prod_{i=0}^{T-1} \frac{\pi(x_i, a_i)}{\pi'(x_i, a_i)} \tag{8.26}$$

若 π 为确定性策略，而 π'是 π 的 ε-贪心策略，则 $\pi(x_i, a_i)$ 对于 $a_i = \pi(x_i)$ 始终为 1，$\pi'(x_i, a_i)$ 为 $\frac{\varepsilon}{|A|}$ 或 $1-\varepsilon+\frac{\varepsilon}{|A|}$，于是就能对策略 π 进行评估了。

8.4.2　时序差分学习

蒙特卡罗强化学习算法通过考虑采样轨迹，克服了模型未知给策略估计造成的困难。此类算法需要完成一个采样轨迹后再更新策略的值估计，而前面介绍的基于动态规划的策略迭代和值迭代算法在每执行一步策略后就进行值函数更新。两者相比，蒙特卡罗强化学习算法的效率低得多，这里的主要问题是蒙特卡罗强化学习算法没有充分利用强化学习任务的 MDP 结构。时序差分（temporal difference，TD）学习结合动态规划与蒙特卡罗方法的思想，能够做到更高效的免模型学习。

蒙特卡罗强化学习算法的本质是通过多次尝试后求平均，以此作为期望累积奖赏的近似，但是它在求平均时是“批处理式”进行的，即在一个完整的采样轨迹完成后再对所有的状态-动作对进行更新，实际上这个更新过程能够增量式进行。对于状态-动作对 (x,a)，假定基于 t 个采样已估计出值函数 $Q_t^{\pi}(x,a) = \frac{1}{t}\sum_{i=1}^{t} r_i$，则在得到第 t+1 个采样 r_{t+1} 时，有

$$Q_{t+1}^{\pi}(x,a) = Q_t^{\pi}(x,a) + \frac{1}{t+1}\left(r_{t+1} - Q_t^{\pi}(x,a)\right) \tag{8.27}$$

显然，只需给 $Q_t^{\pi}(x,a)$ 加上增量项 $\frac{1}{t+1}\left(r_{t+1} - Q_t^{\pi}(x,a)\right)$ 即可。一般地，将 $\frac{1}{t+1}$ 替换为系数 α_{t+1}，则可将增量项写作 $\alpha_{t+1}\left(r_{t+1} - Q_t^{\pi}(x,a)\right)$。在实践中通常令 α_t 为一个较小的正数值 α，若将 $Q_t^{\pi}(x,a)$ 展开为每步累积奖赏之和，则可看出系数之和为 1，即令 $\alpha_t = \alpha$ 不会影响 Q_t 是累积奖赏之和这一性质。更新步长 α 越大，则越靠后的累积奖赏越重要。

以 γ 折扣累积奖赏为例，利用动态规划方法且考虑模型未知时使用状态-动作值函数更方便，有

$$\begin{aligned} Q^{\pi}(x,a) &= \sum_{x' \in X} P_{x \to x'}^{a}\left(R_{x \to x'}^{a} + \gamma V^{\pi}(x')\right) \\ &= \sum_{x' \in X} P_{x \to x'}^{a}\left(R_{x \to x'}^{a} + \gamma \sum_{a' \in A} \pi(x', a') Q^{\pi}(x', a')\right) \end{aligned} \tag{8.28}$$

通过增量求和可得

$$Q_{t+1}^{\pi}(x,a) = Q_t^{\pi}(x,a) + \alpha\left(R_{x \to x'}^{a} + \gamma Q_t^{\pi}(x', a') - Q_t^{\pi}(x,a)\right) \tag{8.29}$$

式中，x'是前一次在状态 x 执行动作 a 后所转移的状态；a'是策略 π 在 x'上选择的动作。

使用式（8.28），每执行一步策略就更新一次值函数估计。该算法由于每次更新值函数前需要知道前一步的状态（state）、前一步的动作（action）、奖赏值（reward）、当前状态（state）、将要执行的动作（action），因此得名 Sarsa 算法。

将 Sarsa 修改为异策略算法，则得到 Q-学习算法，该算法评估的是原始策略，而执行的是 ε-贪心策略。

8.5 值函数近似

假定强化学习任务是在有限状态空间上进行的，每个状态可用一个编号来指代；值函数则是关于有限状态的表格值函数（tabular value function），即值函数能表示为一个数组，输入 i 对应的函数值就是数组元素 i 的值，且更改一个状态上的值不会影响其他状态上的值。然而，现实强化学习任务所面临的状态空间往往是连续的，有无穷多个状态。这该怎么办呢？

一个直接的想法是对状态空间进行离散化，将连续状态空间转化为有限离散状态空间，然后就能使用前面介绍的方法求解。遗憾的是，如何有效地对状态空间进行离散化是一个难题，尤其是在对状态空间进行探索之前。

实际上，不妨直接对连续状态空间的值函数进行学习。假定状态空间为 n 维实数空间 $X=R^n$，此时显然无法用表格值函数来记录状态值。先考虑简单情形，即值函数能够表示为状态的线性函数：

$$V_\theta(\boldsymbol{x})=\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x} \tag{8.30}$$

式中，$\boldsymbol{x}$ 为状态向量，$\boldsymbol{\theta}$ 为参数向量。由于此时的值函数难以像有限状态那样精确记录每个状态的值，因此这种值函数的求解被称为值函数近似（value function approximation）。

希望通过式（8.30）学得的值函数尽可能近似真实值函数 V^π，近似程度常用最小二乘误差来度量：

$$E_\theta=E_{x\sim\pi}\left[\left(V^\pi(x)-V_\theta(x)\right)^2\right] \tag{8.31}$$

式中，$E_{x\sim\pi}$ 表示由策略 π 采样得到的状态上的期望。

为了使误差最小化，采用梯度下降法，对误差求负导数。

$$\begin{aligned}-\frac{\partial E_\theta}{\partial\theta}&=E_{x\sim\pi}\left[2\left(V^\pi(x)-V_\theta(x)\right)\frac{\partial V_\theta(x)}{\partial\theta}\right]\\&=E_{x\sim\pi}\left[2\left(V^\pi(x)-V_\theta(x)\right)x\right]\end{aligned} \tag{8.32}$$

于是可得到对于单个样本的更新规则

$$\theta=\theta+\alpha\left(V^\pi(x)-V_\theta(x)\right)x \tag{8.33}$$

虽然并不知道策略的真实值函数 V^π，但可借助时序差分学习，基于 $V^\pi(x)=r+\gamma V^\gamma(x')$ 用当前估计的值函数代替真实值函数，即

$$\begin{aligned}\theta &= \theta + \alpha\left(\gamma V^{\pi}(x) - V_{\theta}(x)\right)x \\ &= \theta + \alpha\left(r + \gamma V^{\pi}(x') - \theta^{\mathrm{T}}x\right)x \end{aligned} \tag{8.34}$$

式中，x'是下一时刻的状态。

需要注意的是，在时序差分学习中需要状态-动作值函数以便获取策略。这里一种简单的做法是令 θ 作用于表示状态和动作的联合向量上，如给状态向量增加一维用于存放动作编号，即将式（8.30）中的 $\boldsymbol{x}$ 替换为 $(\boldsymbol{x}:\boldsymbol{a})$；另一种做法是用 **0**、**1** 对动作选择进行编码，得到向量 $\boldsymbol{a}=(\mathbf{0};\cdots;\mathbf{1};\cdots;\mathbf{0})$，其中 **1** 表示该动作被选择，再将状态向量与其合并得到 $(\boldsymbol{x}:\boldsymbol{a})$，用于替换式（8.30）中的 $\boldsymbol{x}$。这样就使得线性近似的对象为状态-动作值函数。

基于线性值函数近似来替代 Sarsa 算法中的值函数，即可得到线性值函数近似 Sarsa 算法。类似地可得到线性值函数近似 Q-学习算法。显然，可以容易地用其他学习方法来代替式（8.30）中的线性学习器，如通过引入核方法实现非线性值函数近似。

8.6　模 仿 学 习

在强化学习的经典任务设置中，机器所能获得的反馈信息仅有多步决策后的累积奖赏，但是在现实任务中，往往能够得到人类专家的决策过程范例。例如，在种瓜任务上能够得到农业专家的种植过程范例。从这样的范例中学习，称为模仿学习（imitation learning）。

8.6.1　直接模仿学习

强化学习任务中多步决策的搜索空间巨大，基于累积奖赏来学习很多步之前的合适决策非常困难，而直接模仿人类专家的状态-动作对可显著缓解这一困难，称其为直接模仿学习。

假定获得了一批人类专家的决策轨迹数据 $\{\tau_1,\tau_1,\cdots,\tau_m\}$，每条轨迹包含状态和动作序列

$$\tau_i = \langle s_1^i, a_1^i, s_2^i, a_2^i, \cdots, s_{n_i+1}^i \rangle$$

式中，n_i 为第 i 条轨迹中的转移次数。

有了这样的数据，就相当于告诉机器在什么状态下应选择什么动作，于是可利用监督学习来学得符合人类专家决策轨迹数据的策略。

可将所有轨迹上的所有状态-动作抽取出来，构造出一个新的数据集合：

$$D = \left\{(s_1,a_1),(s_2,a_2),\cdots,\left(s_{\sum_{i=1}^{m} n_i}, a_{\sum_{i=1}^{m} n_i}\right)\right\}$$

即把状态作为特征，动作作为标记；然后，对这个新构造出的数据集合 D 使用分类（对于离散动作）或回归（对于连续动作）算法即可学得策略模型。学得的这个策略模型可

作为机器进行强化学习的初始策略，再通过强化学习方法基于环境反馈进行改进，从而获得更好的策略。

8.6.2 逆强化学习

在很多任务中，设计奖赏函数往往相当困难，从人类专家提供的范例数据中反推出奖赏函数有助于解决该问题，这就是逆强化学习（inverse reinforcement learning）。

在逆强化学习中可知，状态空间 X、动作空间 A 与直接模仿学习类似，有一个决策轨迹数据集 $\{\tau_1,\tau_1,\cdots,\tau_m\}$。逆强化学习的基本思想如下：欲使机器做出与范例一致的行为，等价于在某个奖赏函数的环境中求解最优策略，该最优策略所产生的轨迹与范例数据一致。换言之，寻找某种奖赏函数使范例数据是最优的，然后使用这个奖赏函数来训练强化学习策略。

假设奖赏函数能够表示为状态特征的线性函数，即 $R(x)=\boldsymbol{w}^{\mathrm{T}}x$（$x$ 是将状态映射到低维空间，$\boldsymbol{w}$ 为系数向量），于是，策略 π 的累积奖赏可写为

$$\begin{aligned}\rho^{\pi}&=E\left[\sum_{t=0}^{+\infty}\gamma^{t}R\left(x_t\right)\middle|\pi\right]\\&=E\left[\sum_{t=0}^{+\infty}\gamma^{t}\boldsymbol{w}^{\mathrm{T}}x_t\middle|\pi\right]\\&=\boldsymbol{w}^{\mathrm{T}}E\left[\sum_{t=0}^{+\infty}\gamma^{t}x_t\middle|\pi\right]\end{aligned}\tag{8.35}$$

即状态向量加权和的期望与系数向量 $\boldsymbol{w}$ 的内积。

将状态向量的期望 $E\left[\sum_{t=0}^{+\infty}\gamma^{t}x_t\middle|\pi\right]$ 简写为 $\overline{x}^{\pi}$。注意，要获得 $\overline{x}^{\pi}$ 需求的期望，可使用蒙特卡罗方法通过采样来近似期望，而范例轨迹数据集恰可看作最优策略的一个采样，于是，可将每条范例轨迹上的状态加权求和再平均，记为 $\overline{x}^{*}$。对于最优奖赏函数 $R(x)=\boldsymbol{w}^{*\mathrm{T}}x$ 和任意其他策略产生的 $\overline{x}^{\pi}$，有

$$\boldsymbol{w}^{*\mathrm{T}}\overline{x}^{*}-\boldsymbol{w}^{*\mathrm{T}}\overline{x}^{\pi}=\boldsymbol{w}^{*\mathrm{T}}(\overline{x}^{*}-\overline{x}^{\pi})\geqslant 0\tag{8.36}$$

若能对所有策略计算出 $(\overline{x}^{*}-\overline{x}^{\pi})$，即可解得

$$\begin{aligned}w^{*}&=\arg\max_{\boldsymbol{w}}\min_{\pi}\boldsymbol{w}^{\mathrm{T}}(\overline{x}^{*}-\overline{x}^{\pi})\\&\|\boldsymbol{w}\|\leqslant 1\end{aligned}\tag{8.37}$$

显然，我们难以获得所有策略，一个较好的办法是从随机策略开始，迭代地求解更好的奖赏函数，基于奖赏函数获得更好的策略，直至最终获得符合范例轨迹数据集的奖赏函数和策略。注意在求解更好的奖赏函数时，需要将式（8.37）中对所有策略求最小改为对之前学得的策略求最小。

本 章 小 结

强化学习是机器学习的范式和方法论之一，用于描述和解决智能体在与环境的交互过程中通过学习策略以达成回报最大化或实现特定目标的问题。强化学习的核心优势在于可以主动学习，从环境当中获取所需要的反馈，这也是强化学习相对于传统机器学习的重要优势之一；另一个优势是强化学习算法学习的是在动态环境中可执行的策略，这一点比监督和非监督学习更接近于人们通常所理解的人工智能。

k-摇臂赌博机是单步强化学习任务对应的一个理论模型，目的是通过一定的策略最大化自己的奖赏。“仅探索”法能够很好地估计每个摇臂的奖赏，却会失去很多选择最优摇臂的机会；“仅利用”法没有很好地估计摇臂期望奖赏，很可能经常选不到最优摇臂。所以，欲积累奖赏最大，必须在“探索”和“利用”之间达成较好的折中。相应的策略有 ε-贪心和 Softmax 算法。

在已知模型的环境中学习称为有模型学习。在模型已知时，对任意策略都能估计出该策略带来的期望累积奖赏，通过状态值函数可直接计算出状态-动作值函数。对某个策略的累积奖赏进行评估后，若发现它并非最优策略，可通过其进行改进，以期望达到最大化累积奖赏。

若学习算法不依赖于环境建模，则称为免模型学习。在免模型情形下，策略迭代算法首先遇到的问题是策略无法评估。此时，智能体通过在环境中执行选择的动作来观察转移的状态和得到的奖赏。常见的免模型学习有蒙特卡罗强化学习和时序差分学习。

现实强化学习任务所面临的状态空间往往是连续的，有无穷多个状态，如何有效地对状态空间进行离散化是一个难题，尤其是在对状态空间进行探索之前。值函数近似法通过对连续状态空间的值函数进行学习、用最小二乘误差来度量近似程度等方法缩小误差，可有效解决无穷多个状态无法有效映射的问题。

在强化学习的经典人物设置中，所能获得的反馈信息仅有多步决策后的累积奖赏，但是在现实任务中，往往能够得到人类专家的决策过程范例，从这样的范例中学习，称为模仿学习，常见的模仿学习包括直接模仿学习和逆强化学习。

本 章 习 题

1．用于 k-摇臂赌博机的上置信界（upper confidence bound，UCB）方法每次选择 $Q(k)+UC(k)$ 最大的摇臂，其中 $Q(k)$ 为摇臂 k 当前的平均奖赏，$UC(k)$为置信区间。例如，$Q(k)+\sqrt{\dfrac{2\ln n}{n_k}}$，其中 n 为已执行所有摇臂的总次数，n_k 为已执行摇臂 k 的次数。试比较 UCB 方法与 ε-贪心法和 Softmax 方法的异同。

2．试写出基于 γ 折扣奖赏函数的策略评估算法。

3．试写出基于 γ 折扣奖赏函数的策略迭代算法。

4．在没有 MDP 模型时，可以先学习 MDP 模型（如使用随机策略进行采样，从样本中估计出转移函数和奖赏函数），然后再使用有模型强化学习方法，试述该方法与免模型强化学习方法的优缺点。

5．试推导出 Sarsa 算法的更新公式（8.29）。

6．线性值函数近似在实践中往往有较大误差。试结合 BP 神经网络，将线性值函数近似 Sarsa 算法推广为使用神经网络近似的 Sarsa 算法。

7．试结合核方法，将线性值函数近似 Sarsa 算法推广为使用核函数的非线性值函数近似 Sarsa 算法。

8．对于目标驱动（goal-directed）的强化学习任务，目标是达到某一状态，如将汽车驾驶到预定位置。试为这样的任务设置奖赏函数，并讨论不同奖赏函数的作用（如每一步未达目标的奖赏为 0、−1 或 1）。

9．与传统监督学习不同，直接模仿学习在不同时刻所面临的数据分布可能不同。试设计一个考虑不同时刻数据分布变化的模仿学习算法。

参 考 文 献

蔡自兴，贺汉根，2002．智能科学发展的若干问题[J]．自动化学报（增刊）：142-150．

蔡自兴，徐光祐，2010．人工智能及其应用[M]．4 版．北京：清华大学出版社．

陈国良，王熙法，庄镇泉，等，1999．遗传算法及其应用[M]．北京：人民邮电出版社．

段海滨，2005．蚁群算法原理及其应用[M]．北京：科学出版社．

段晓东，王存睿，刘向东，2007．粒子群算法及其应用[M]．沈阳：辽宁大学出版社．

高济，朱淼良，何钦铭，2002．人工智能基础[M]．北京：高等教育出版社．

高尚，杨静宇，2006．群智能算法及其应用[M]．北京：中国水利水电出版社．

高阳，陈世福，陆鑫，2004．强化学习研究综述[J]．自动化学报，30（1）：15．

惠军华，2021．知识表示与处理[M]．北京：电子工业出版社．

纪震，廖惠连，吴青华，2009．粒子群算法及应用[M]．北京：科学出版社．

李德毅，于剑，2018．人工智能导论[M]．北京：中国科学技术出版社．

李航，2019．统计学习方法[M]．2 版．北京：清华大学出版社．

李敏强，寇纪淞，林丹，等，2002．遗传算法的基本理论与应用[M]．北京：科学出版社．

林尧瑞，郭木河，2001．人类智慧与人工智能[M]．北京：清华大学出版社．

刘全，翟建伟，章宗长，等，2018．深度强化学习综述[J]．计算机学报，41（1）：1-27．

邱锡鹏，2020．神经网络与深度学习[M]．北京：机械工业出版社．

孙俊，方伟，吴小俊，等，2011．量子行为粒子群优化：原理及其应用[M]．北京：清华大学出版社．

王万良，2005．人工智能及其应用[M]．北京：高等教育出版社．

王万良，2020．人工智能导论[M]．5 版．北京：高等教育出版社．

王智明，杨旭，平海涛，2006．知识工程及专家系统[M]．北京：化学工业出版社．

尹朝庆，尹皓，2002．人工智能与专家系统[M]．北京：中国水利水电出版社．

周志华，2016．机器学习[M]．北京：清华大学出版社．

朱福喜，汤怡群，傅建明，2002．人工智能原理[M]．武汉：武汉大学出版社．

（TP-9011.0101）

本书特点

○ 轻松入门，浅显易懂。
○ 内容系统，应用性强。

普通高等教育信息技术类系列教材

- 数据库应用与实践（MySQL数据库）
- 人工智能基础
- Python程序设计基础
- 数据结构（C语言版）
- 现代教育技术
- 计算机网络实验教程
- 计算机维护与管理实例教程
- 离散数学
- 机器人项目实践案例库
- Linux操作系统基础理论与实践（基于CentOS 8）
- 数字逻辑
- Java Web程序设计实用教程
- Office办公软件高级修炼
- 大学计算机基础（Windows 10+Office 2016）

科学出版社 技术分社
http://www.abook.cn

www.sciencep.com

ISBN 978-7-03-072396-3

定价：42.00元