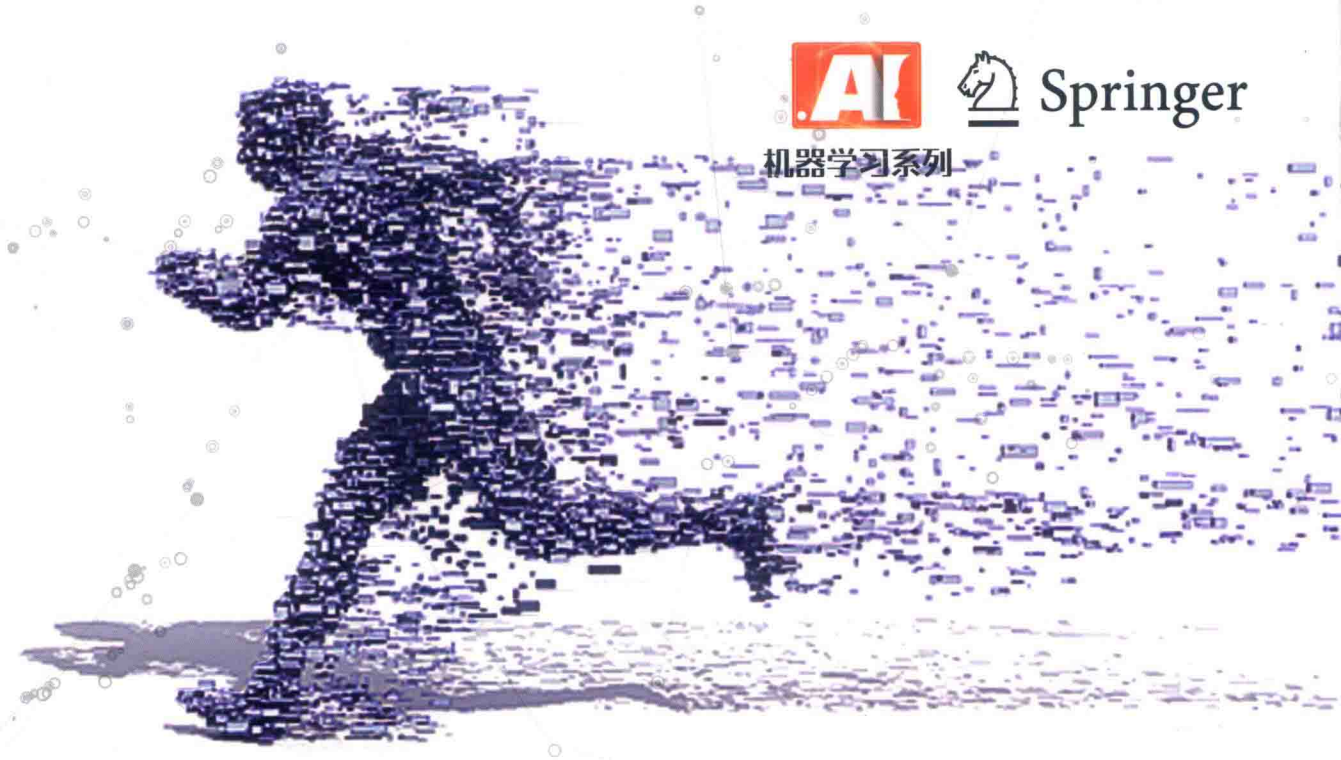


机器学习系列

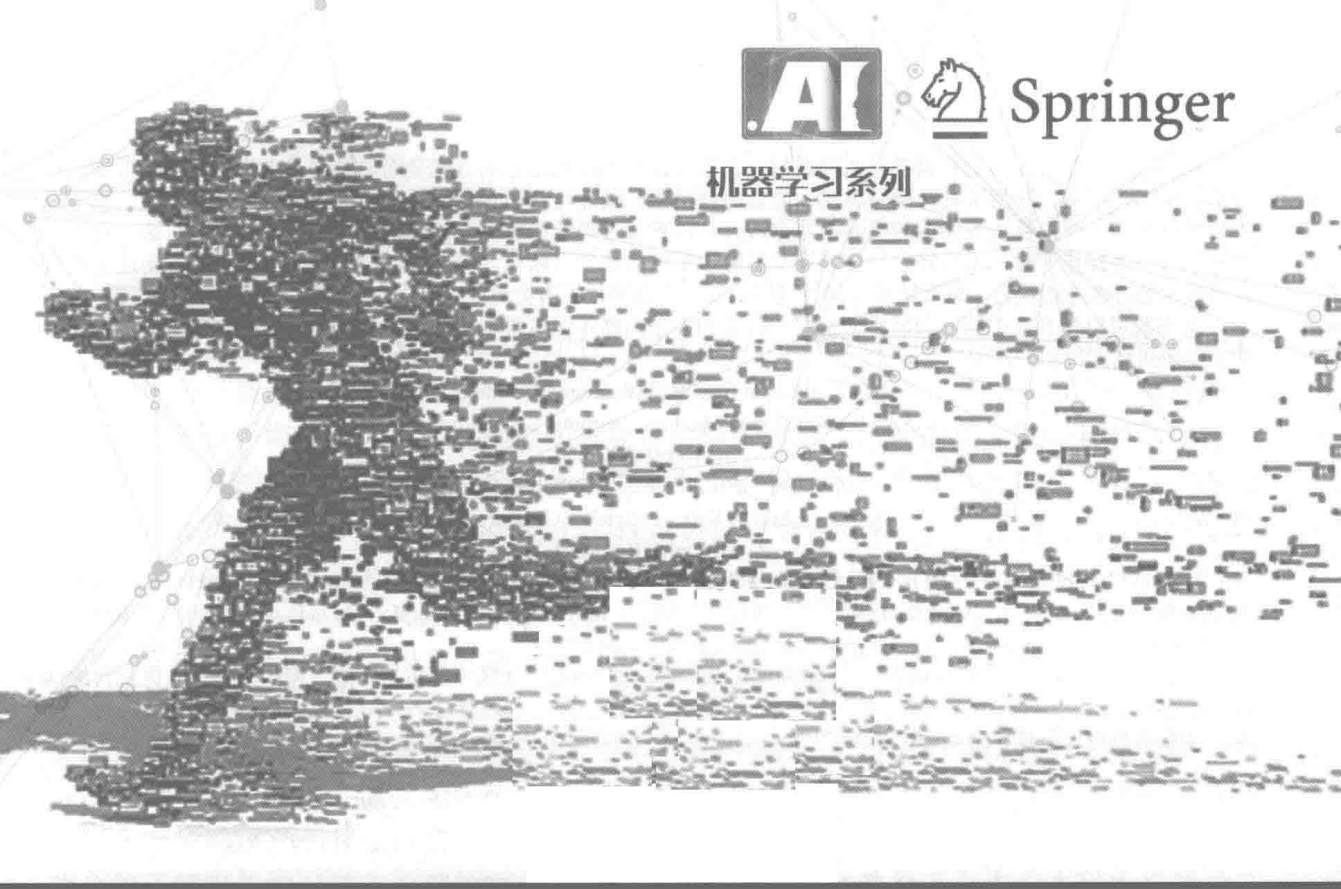
An Introduction to Machine Learning Second Edition

机器学习导论

（原书第2版）

[美] 米罗斯拉夫·库巴特（Miroslav Kubat）著
王勇　仲国强　孙鑫　等译

机械工业出版社
CHINA MACHINE PRESS



机器学习系列

An Introduction to Machine Learning Second Edition

机器学习导论

（原书第2版）

[美] 米罗斯拉夫•库巴特（Miroslav Kubat）著

王 勇　仲国强　孙 鑫　等译

机械工业出版社
CHINA MACHINE PRESS

本书是一本浅显易懂的机器学习入门教材，它以理论与实际相结合的方式全面地涵盖了主流的机器学习理论与技术。全书共17章，介绍了贝叶斯分类器、最近邻分类器、线性与多项式分类器、人工神经网络、决策树、基于规则集的分类器、遗传算法等经典的机器学习方法，对计算学习理论、性能评估、统计显著性等进行了讨论。讲解了集成学习、多标签学习、无监督学习和强化学习等重要的机器学习领域。本书还通过大量的应用实例，阐述了机器学习技术的许多应用技巧。每章结尾对相关机器学习工作都进行了历史简评，并附有练习、思考题和上机实验。

本书既可以作为人工智能领域机器学习方向及相关方向高年级本科生或低年级研究生的教材，也可供机器学习相关专业研究人员和工程师参考阅读。

图书在版编目（CIP）数据

机器学习导论：原书第2版/（美）米罗斯拉夫·库巴特（Miroslav Kubat）著；王勇等译.
—北京：机械工业出版社，2018.8
（机器学习系列）
书名原文：An Introduction to Machine Learning Second Edition
ISBN 978-7-111-60581-2

Ⅰ.①机…　Ⅱ.①米…②王…　Ⅲ.①机器学习－高等学校－教材　Ⅳ.①TP181

中国版本图书馆CIP数据核字（2018）第171513号

机械工业出版社（北京市百万庄大街22号　邮政编码100037）
策划编辑：顾　谦　　责任编辑：顾　谦
责任校对：张　薇　　责任印制：李　昂
北京宝昌彩色印刷有限公司印刷
2018年9月第1版第1次印刷
184mm×240mm · 16.25印张 · 368千字
0001—4000册
标准书号：ISBN 978-7-111-60581-2
定价：79.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务		网络服务	
服务咨询热线：	010-88361066	机工官网：	www.cmpbook.com
读者购书热线：	010-68326294	机工官博：	weibo.com/cmp1952
	010-88379203	金书网：	www.golden-book.com
封面无防伪标均为盗版		教育服务网：	www.cmpedu.com

译者序

目前，人类已经进入了智能信息时代。以机器学习为核心的人工智能技术正在推动着各个领域的发展，比如机器人、智能手机、物联网、数据挖掘等。机器学习这一学科融合了计算机科学、统计学、最优化理论和神经计算学等多个学科的精华，以学习数据内在结构和其中蕴藏的规律性知识为目标，如果说数据是矿藏资源，那无疑机器学习就是发掘资源价值的生产工具。而现在，几乎每个学科，包括社会科学领域，都在不停地产生数据，这也就是机器学习得以快速发展和普及的缘由。

本书翻译自 Miroslav Kubat 所撰写的一本入门级的机器学习教材。Kubat 现在是美国迈阿密大学电子与计算机工程系的副教授，也是机器学习和人工智能方面的专家。本书几乎覆盖了机器学习所有的基本议题，比如贝叶斯分类器、决策树、神经网络、计算学习理论、集成学习、遗传算法和强化学习等，而且本书特别强调学以致用，不仅希望学生能够掌握基本的机器学习技术，而且希望他们作为未来的工程师了解各种方法的优势与劣势，以适应工业界的需求。和其他机器学习图书相比，本书最大的特点是浅显易懂，可很好地满足公众想了解机器学习基本原理的需求。另外，本书的组织和选材适合作为本科生机器学习课程的教材。当然，本书也可以作为一本参考书来使用。

本书基于原书第 2 版翻译，相较原书第 1 版有大量修订，并增加了 3 章内容。全书共 17 章，主要由王勇、仲国强和孙鑫三位老师共同翻译完成。王勇负责第 1 章、第 10~12 章、第 14 章、第 16 章、第 17 章，仲国强负责第 2~6 章、第 15 章，孙鑫负责第 7~9 章、第 13 章。本书得到了高宏老师的大力支持，以及张伊丹、谭碧君同学的帮助，王天之、王丽丽、李逸、刘雯、凌霄、王海珍、魏洪旭、张康、王丽娜、于黛薇、李涛、高威、刘文雪、岳国华、林鑫、矫文聪、马雪莹、史均宇、李振华和刘利朋等同学参与了部分内容的翻译整理工作。

在本书翻译过程中，得到了华东政法大学高宏老师的大力支持，在此表示衷心感谢。译者也感谢机械工业出版社的编辑们，正是他们的努力，本书才得以与读者见面。

本书涉及的内容广泛，对一些没有标准译法的术语，尽量采用当前流行的译法，对可能产生歧义和重要的术语均附上了英文原文。翻译时力求忠于原著，并符合中文的表述习惯，翻译过程中进行了大量讨论。尽管如此，限于水平，不当之处和错误仍在所难免，恳请读者批评指正。

译　者

2018 年 6 月 10 日

原书前言

机器学习已走向成熟。如果读者觉得只是说说而已，请允许我做些解释。

人们希望机器某一天能够自己学习，这个梦想几乎在计算机出现时就有了，也许更早。不过，长久以来，这仅仅是一个想象而已。Rosenblatt 感知机的提出曾经掀起过一股热潮，但是现在回想起来，这股热潮没能持续很长的时间。至于接下来的尝试，使情况发展得更糟，这个领域甚至没有再引起人们的注意，长期被忽视，无法取得重大突破，也没有这一类的软件公司，后续研究寥寥无几并且得到的资金支持也不多。这个阶段，机器学习一直不被看好，像进入休眠一样，在其他成功学科的阴影里生存。

接下来一切都改变了。

一群有识之士指出，在 20 世纪 70 年代的人工智能领域，基于知识的系统曾经风靡一时，但它们有一个弱点："知识"从哪里来？当时主流的观点认为，应该让工程师和领域专家合作，用 if-then 的形式表示出来。但是实际情况差强人意，专家们发现很难把掌握的知识表达给工程师。反过来，工程师也不知道该问什么问题以及如何表示答案。尽管有几个广为人知的成功案例，但是其他大多数研究都试图建立知识库，并且成千上万的规则令人沮丧。

这些有识之士主张简单和直接的操作。如果难以准确地告诉机器如何处理某个问题，为什么不间接地给出指令，让计算机通过例子来学习——对，就是学习——所需要的技能？

当然，这必须要有能够进行学习的算法才有意义，这是主要的困难。结果发现无论是 Rosenblatt 感知机还是后来出现的技术都不太管用。然而机器学习技术的缺乏不是障碍，相反是一个挑战，并激发出了很多绝妙的点子。其中使计算机有学习能力这个想法开创了一个激动人心的新领域，并引起了世人的关注。

这一想法在 1983 年爆发了。一卷很厚的论文集——《机器学习：人工智能的方法》[⊖] 提出了各种方法来解决这个巨大的问题。在它的影响下，几乎一夜之间一个新的学科诞生了。3 年后，后续著作一本接一本地出现。相关学术刊物也很快被创立，有着巨大影响力的年度学术会议相继召开。几十或许是几百篇博士论文完成并通过答辩。

早期阶段，问题不仅是如何学习，而是学什么和为什么学。这段充满创造力的岁月让人难以忘怀。唯一有些遗憾的是很多非常好的想法后来被放弃了。实用主义占了上风，资源都被投向那些最有希望的方向。经过一段时间的发展，具体研究基本成形：知识系统 if-then 规则的归纳、分类归纳、程序基于经验来提高技能、Prolog 程序自动调优以及其他方面。相关的研究方向非常多，一些知名学者希望通过写书引领未来的发展，这其中有些人做得很成功。

机器学习发展的一个重要的转折点是 Tom Mitchell 的传奇教科书[⊖]。该书向博士生和科

⊖ R. Michalski、J. Carbonell、T. Mitchell 编辑。

⊖ T. Mitchell，机器学习，McGraw-Hill（1997）。

学家们总结了该领域的发展现状，慢慢地大学也用这本书作为研究生的教材。同时，研究方法也变得更加系统化。大量机器学习测试库被建立起来，用于比较性能或者学习算法的优劣。统计评估方法也被广泛地使用在评估过程中。相关流行程序的公开版本很容易获得，从事这个学科的人数增至数千甚至更多。

现在，到了很多大学都为本科生开设机器学习课程的阶段，通常这些课程需要不同类型的教材。除了掌握基本技术以外，学生还需要了解不同方法的优点和缺点，以及不同情况下每种方法的独特之处。最重要的是，他们需要理解在特定情况下，哪些技术是可行的，哪些是不可行的。只有这样才能在解决具体问题时做出正确的选择。一本教材除了满足以上各项要求外，还应该少讲一些数学概念，多包括一些实用的建议。

关于教材，还要考虑材料的多少、结构以及风格，以便能够支持一个学期的导论课程。

第一个问题是材料的选择。当高科技公司准备成立机器学习研究团队时，大学就要向学生传授相关的知识和技能，以及对行业当前需求的理解。为此，本书重点介绍了贝叶斯分类器、最近邻分类器、线性和多项式分类器、决策树、神经网络的基础以及提升（Boosting）算法的原理。本书很大篇幅用来描述具体应用的典型特征。在现实中，当把基本技术用于真正有难度的任务上时，它们的表现可能和老师在课上的简单演示不完全一样。学生应对此有所了解。

本书共包括 17 章，每章覆盖一个专题。各章分成很多节，每节介绍一个关键问题。建议学生在做完每一节后面的 2~4 个问题后再学习下一节。这些问题用来帮助检查对学习材料的掌握情况。如果不会做这些题，有必要重新阅读相关内容。

俗话说熟能生巧。每章结尾安排了必要的练习用于实际操作。如果接下来的思考、实验能够全部完成，将有助于更深入理解所学内容的各个方面。不过这些实验难度较大，只有付出很大努力才能获得正确的理解。所学的知识在上机实验中可被进一步巩固。编程对于学习同样重要。现在，人们都习惯从网上下载所需的程序，这是捷径，但本书不建议这样做。因为只有强迫自己实现了程序的全部细节，才能领会本书机器学习技术的精妙之处。

目 录

第 1 章
一个简单的机器学习任务

人们会发现，很难通过精确描述自己母亲的长相，让朋友在超市里认出她。但如果给朋友看几张自己母亲的照片，朋友会立刻把握住所需要的特征。这就是所说的，一幅图（样例）胜过千言万语。

这是人们希望技术去实现的。不能足够精确地定义事物或概念，则想把它们以样例的形式传送给计算机。为了能这么做，计算机需要有能力把样例转换成知识。所以，人们的兴趣在于机器学习（Machine Learning）的算法和技术，这也是本书的主题。

本章把任务表示为搜索问题，介绍了爬山搜索算法。它不仅是解决机器学习任务的初步尝试，也是解决后面各章附加问题的便利工具。有了这些基础，将继续学习诸如性能准则、实验方法以及某些学起来很难但很有意思的内容。

1.1 训练集和分类器

首先介绍一些贯穿全书的问题和基本概念。

预分类训练样例集 图 1.1 展示了 Johnny 喜欢的和不喜欢的 6 种派（Pie）。这些正例（Positive Examples）和负例（Negative Examples）构成了训练集（Training Set），机器可以从中归纳出分类器（Classifier）——一种将来能把任何派归到正例或负例中去的算法。

毫无疑问，分类的个数可以更多。例如，一个分类器要有春、夏、秋、冬 4 个分类才能区分出一张风景照拍摄的季节。iPad 上识别手写字迹的软件至少需要 36 个分类：26 个字母和 10 个数字，文件分类系统则要能够识别上百种甚至上千种不同的主题。这里选二分类问题是因为简单。

属性向量 为了能把训练样例发送给计算机，必须用适当的方法描述它们，最常见的方法是使用所谓的属性（Attributes）机制。在“派”问题中，有 5 种属性：形状（shape），取值包括圆形（circle）、三角形（triangle）、正方形（square）；外壳尺寸（crust-size），取值为厚（thick）、薄（thin）；外壳色度（crust-shade），取值为白色（white）、灰色（gray）、深色（dark）；馅料尺寸（filling-size），取值为厚（thick）、薄（thick）；馅料色度（filling-shade），取值为白色（white）、灰色（gray）、深色（dark）。表 1.1 给出了图 1.1 中 12 个训练样例的属性值。例如，图 1.1 中左上角的派（表 1.1 中的 ex1）可用下列组合属性描述：

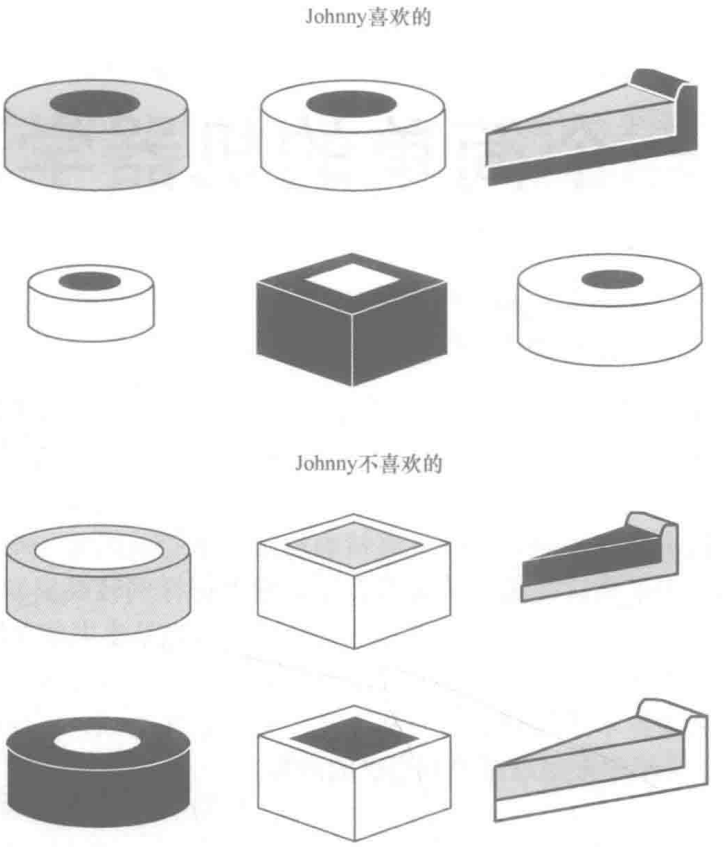


图 1.1　一个简单的机器学习任务：归纳学习出能将今后任意派归到正例或负例的分类器。例如，归纳学习一个“Johnny 喜欢的派”分类器

表 1.1　以矩阵表示的 12 个训练样例

样例	形状	外壳		馅料		类别
		尺寸	色度	尺寸	色度	
ex_1	圆形	厚	灰色	厚	深色	正例
ex_2	圆形	厚	白色	厚	深色	正例
ex_3	三角形	厚	深色	厚	灰色	正例
ex_4	圆形	薄	白色	薄	深色	正例
ex_5	正方形	厚	深色	薄	白色	正例
ex_6	圆形	厚	白色	薄	深色	正例
ex_7	圆形	厚	灰色	厚	白色	负例
ex_8	正方形	厚	白色	厚	灰色	负例
ex_9	三角形	薄	灰色	薄	深色	负例
ex_{10}	圆形	厚	深色	厚	白色	负例
ex_{11}	正方形	厚	白色	厚	深色	负例
ex_{12}	三角形	厚	白色	厚	灰色	负例

(shape=circle) AND (crust–size=thick) AND (crust–shade=gray)
AND (filling–size=thick) AND (filling–shade=dark)

归纳分类器 训练集构成了输入，从中可归纳出分类器。不过，是什么样的分类器？

假设希望它是布尔函数的形式，正例为真（true），负例为假（false）。对训练集检查表达式 [（shape=circle）AND（filling–shade=dark），会发现所有负例的值都是假：虽然也能找到圆形的负例，但它们没有灰色的馅。对正例来说，有 4 个表达式是真，其余的是假。分类器错了两次，这不能接受，应该有更好的解法。事实上，很容易验证下面的表达式在整个训练集上都不出错：

[(shape=circle) AND (filling–shade=dark)] OR
[NOT(shape=circle) AND (crust–shade=dark)]

强力计算的问题 计算机如何去找这种分类器？强力计算（计算机最擅长的）不能用在这里。只需要想一下派问题中那些属性可以标识出多少不同的样例。对 3 种不同的 shape，每种对应两个不同的 crust-size，组合的数量为 $3 \times 2=6$。对里面的每一个，下一个属性，即 crust-shade，有 3 种不同的值，组合数变为 $3 \times 2 \times 3=18$。以此类推到全部属性，将发现样例空间包含 $3 \times 2 \times 3 \times 2 \times 3=108$ 个不同的样例。

这些样例的每个子集（一共有 2^{108} 个子集！）构成了“好派”的可能正例清单，且每个子集至少可以用一个布尔表达式描述。在训练集中运行每个分类器是不可能的。

手工方法及搜索 不确定怎样找到归纳分类器的算法，可以从用纸笔的“手算”中寻找启发。这时算法的初级版本从一些假定开始，比如说 shape=circular。用训练集检验该表达式，发现 4 个正例为真，但是也有 2 个负例为真。显然，分类器需要“缩小”（特殊化）以便排除那两个负例。特殊化的一种方法是增加连接词，例如把 shape=circular 变成 [（shape=circular）AND（filling–shade=dark）]。新的表达式虽然使所有负例都为假，但也有问题，它只包含了 6 个正例中的 4 个（ex1、ex2、ex4、ex6）。因此接下来应尝试一般化，比如增加分离词：{[（shape=circular）AND（filling–shade=dark）] OR（crust–size=thick）}。这样继续下去，直到找到 100% 正确的分类器为止（如果存在）。

从这种对照检查的方法中可以学到，分类器可以通过一系列特殊化以及一般化的步骤来创建，逐步修改，直到符合预定义的要求。这令人振奋。有人工智能知识基础的读者知道这个过程是在布尔表达式空间中的一种搜索。可以知道人工智能开发出了很多这样的搜索算法，至少了解其中一种算法是有必要的。

学到了什么？

为确保读者已经理解了本节的内容，请试着回答以下问题。如果在解答过程中有任何困难，请回顾本节的相关内容。

- 什么是学习问题的输入和输出？
- 如何描述训练样例？什么是实例空间？可以计算实例空间的大小吗？
- 在“派”问题中，找出能将表 1.1 中的所有训练样例正确分类的布尔表达式。

1.2 题外话：爬山搜索

现在把讨论过的搜索形式化，并介绍一种流行的算法，即所谓的爬山法（Hill Climbing）。人工智能这样定义搜索：从初态（Initial State）开始，找到一系列步骤，通过一组中间的搜索状态（Search States），到达预定义的终态（Final State）。从一种搜索状态转移到另一种搜索状态，每一步都由程序员预设的搜索算符（Search Operator）完成。应用搜索算符的次序遵循特定的搜索策略（Search Strategy）（见图 1.2）。

图 1.2 由初态、终态、搜索算符和搜索策略描述的搜索问题

爬山法：示例 爬山法是一种流行的搜索策略。用熟知的数字滑块游戏来演示爬山法的原理。这个游戏最简单的版本是在一块板上画出 3 行共 9 个方格，其中 8 个方格上放着印有数字的滑块（整数 1~8），最下一行左边的方格空着。通过把邻近滑块移到空方格中来转变搜索状态。目标是达到一个预定的滑块布局。

在图 1.3 所示的流程中，从一个具体的初态开始，在这里可从两个搜索算符中选择："上移滑块 6" 和 "左移滑块 2"。选择的依据是评估函数（Evaluation Function），它估算每个状态到目标的距离。一个简单的算法可能是数一下滑块到达最终目的地所要经过的方格数。在初态中，滑块 2、4、5 已经在正确的位置；滑块 3 必须滑过 4 个方格；滑块 1、6、7、8 都要滑过 2 个方格。因此，距离总和为 $d = 4 + 4 \times 2 = 12$。

在图 1.3 中，两种搜索算符都能使初态到达距离终态 $d = 13$ 的状态。因为没有其他提示，随机选择左移，使空方格在第一行的中间。这时有 3 种可能的移动。其中一种只能返回初态，可以忽略掉。其余的 2 种：一种能达到 $d = 14$ 的状态；另一种能达到 $d = 12$ 的状态，这是本书要走的。下一步很简单，因为只有一种走法是没走过的。之后，又有两种选择……搜索就这样一直进行下去直至达到了终态。

可选的终止准则和评估函数 也可以考虑其他的终止准则（Termination Criteria）。当超过了允许的最长运行时间（不想让计算机永远运行）时、当经过的状态数量超过了某个界限时、当发现已经十分接近终态时、当所有的状态都经过了等，具体公式反映了特定应用的关键特征，有时可以把两个或多个准则合成一个。

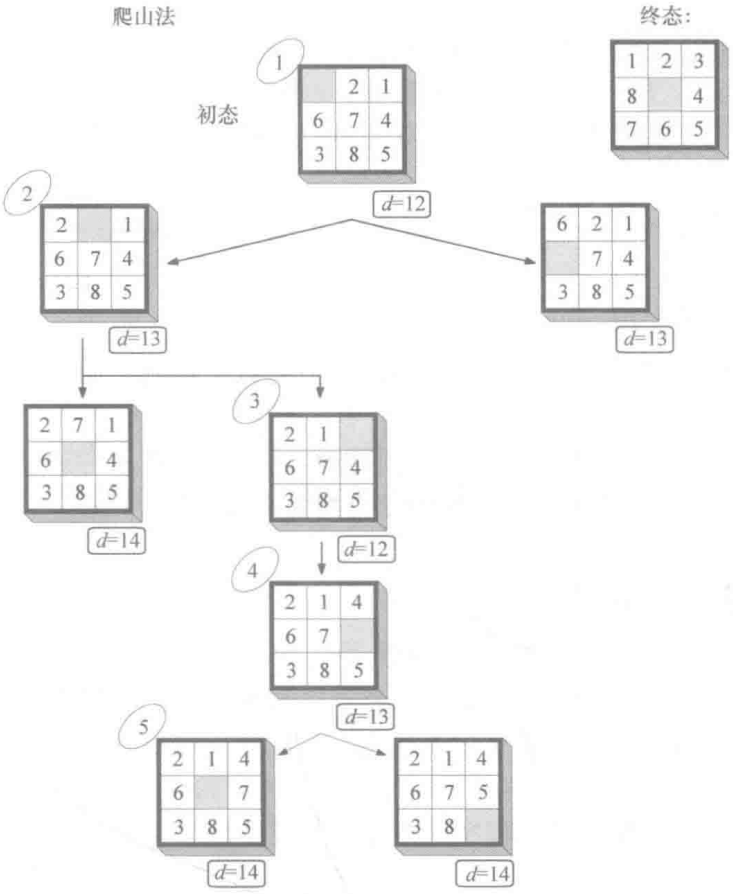


图 1.3　爬山法。带圈的整数指示搜索状态经过的顺序，d 是用评估函数算出的到终态的距离。距离一样时随机选择

另外，数字滑块例子中用的评估函数十分简单，刚够完成任务：使用者把对问题的理解，提示给可能喜欢这么走的人。为了在现实中好用，还要找出更精巧的函数。经常的情况是，有很多不同的变化，每种都产生一串不同的走法。有些很快就到达终点，其他的则绕了更远的路。程序的性能取决于程序员选出正确方案的能力。

爬山算法　算法伪代码见表 1.2。虽然细节取决于程序员的风格，但总有几个典型函数。一个是比较状态，如果相同则返回真；程序根据它判断是不是到达终态。另一个函数根据给定的状态，把所有搜索算符应用一遍，产生一个“子状态”全集。为避免死循环，第 3 个函数检查一个状态是否被经历过了。第 4 个函数计算给定状态到终态的距离，第 5 个函数根据距离对“子状态”排序，并置于列表 L 顶端，最后一个函数检验状态是否满足终止条件[⊖]。

最后看一下图 1.3 中的有些状态，没有“子状态”可以改进“父状态”，只有暂时妥协才能得到较低的 d。登山者可能经历过：为了继续向上，有时不得不先向下过一个山谷。可用爬山来比喻，也是这个方法名字的由来。

⊖　为简单起见，伪代码忽略了达到或不能达到终态以外的终止准则。

表 1.2　爬山搜索算法

1. 创建两个列表，L 和 L_{seen}。最初 L 只包含初态，L_{seen} 是空的。
2. 令 n 是 L 中的第一个元素，并与终态进行比较。如果相同表示成功，停止执行。
3. 对 n 使用所有搜索算符，得到了一组新状态。去掉 L_{seen} 中已经有的状态。其余状态用评估函数排序，置于 L 的顶端。
4. 将 n 从 L 转移到 L_{seen}，L_{seen} 中都是经历过的状态。
5. 如果 $L=\phi$，停止运算并报告失败。否则转到第 2 步

学到了什么?

为确保读者已经理解了本节的内容，请试着回答以下问题。如果在解答过程中有任何困难，请回顾本节的相关内容。

- 人工智能是如何定义搜索问题的？如何理解“搜索空间”和“搜索算符”？
- 评估函数承担什么角色？它如何影响爬山搜索算法？

1.3　机器学习中的爬山法

下面将研究具体的做法，把爬山法用于机器学习。

爬山法和 Johnny 的派　从确定 Johnny 喜欢哪种派开始。一组训练样例作为输出，它们都由属性描述。输出—终态—是布尔表达式，对训练集中所有正例为真、负例为假。这个表达式包括属性 - 值对、逻辑运算符（连接词、分离词和否定）以及所需的括号。评估函数度量给定表达式在训练集上的错误率。对初态，任意随机生成的表达式都可以使用。图 1.4 中选择（shape=circle）作为初态，因为超过半数的训练样例是圆形的。

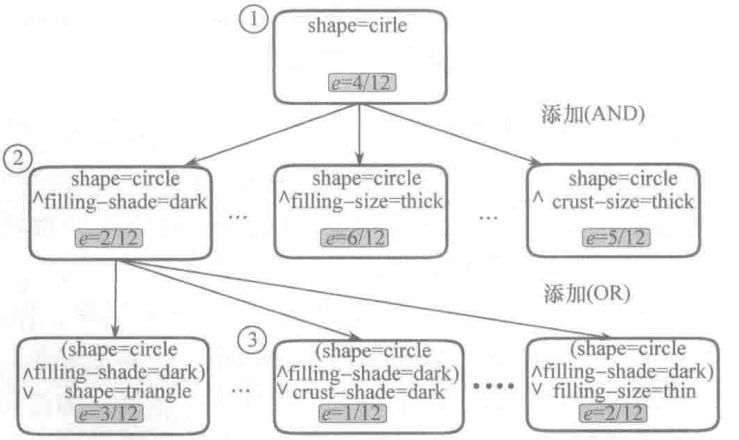


图 1.4　“派”问题中的爬山搜索法

至于搜索算符（Search Operator），一种可能是添加连接词，如图 1.4 上部所示：例如，根最左边的子节点用 [(shape=circle) AND (filling–shade=dark)] 来替换 (shape=circle)（图中逻辑算符 AND 用符号“∧”表示）。可以发现即便这么简单的问题，连接算符还是生成了如此多不同的表达式。对（shape=circle），其他的属性 - 值对都可以 AND。因为余下的 4 个属性（除了 shape）分别取 2、3、2、3 个不同的值，所以可以连接到 (shape=circle) 上的表

达式总数为 $2 \times 3 \times 2 \times 3 = 36$。[⊖]

或者可以添加分离词，如图 1.4 中最左子节点的 3 个扩展所示。其他算符还可以是“去掉一个连接词”“去掉一个分离词”“加上一个否定词”“否定一个条件”以及各种括号的操作等。总之，几百种搜索算符可以用到一个状态及其后续的结果状态上。即使在非常简单的问题中也是难以管理的。

数值属性 “派”问题中，每种属性需要取多个离散值中的一个，但实际应用中，属性可能是数值。例如，每个派都有价格，它的值是连续的。这时搜索会是什么样的？

简单起见，假设只有两个属性：重量和价格。这样就可以用平面上的点来表示每个训练样例，如图 1.5 所示。可以发现，同类的样例倾向于处在特定的区域，可以用数学方法定义的直线、圆、多项式曲线把这些区域分离开来。例如，图 1.5 右侧有 3 个圆圈，每个都相当于一个分类器：圈里面的样例认为是正例；圈以外是负例。再次强调，这些分类器中，有一些比其他的要好。爬山法如何找出最好的那些？下面是一种可能的情况。

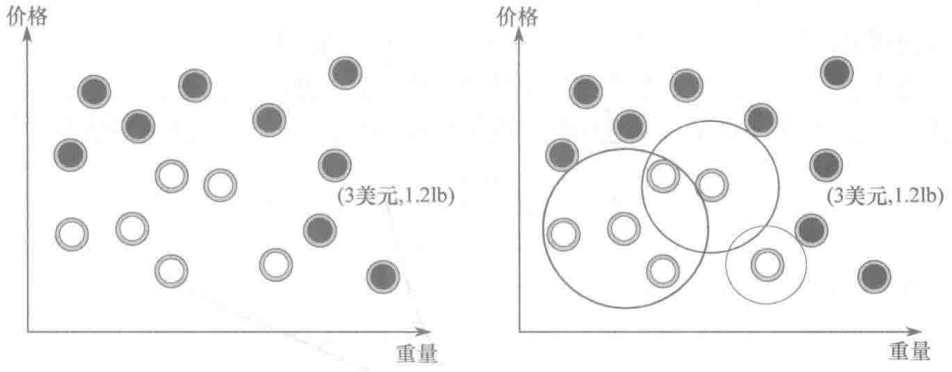


图 1.5　左：具有连续属性的问题；右：一些“圆形”分类器

注：1lb=0.45359237kg。——译者注

数值属性问题中的爬山法

初态（Initial State） 圆心和半径决定一个圆。可以随机挑选一个正例作为初态的圆心，并使最初的半径小到只能包含这一个样例。

搜索算符（Search Operators） 可用两种搜索算符：一种增大圆的半径；另一种把圆心从一个训练样例换成另一个。在前者中，还需要决定半径能变化多少。一种做法是，增大的半径仅够多包括进来一个训练样例。起初，圆内只有 1 个训练样例。第一步过后，将会有 2 个，然后 3 个、4 个等。

终态（Final State） 圆并不是代表正例区域的理想形状。在这个例子中，100% 精确是达不到的，更愿意把终态定义为“能对 95% 训练样例正确分类的分类器”。

评估函数（Evaluation Function） 和前面一样，选最小的错误率。

⊖ 图 1.4 只展示了 36 个新建状态中的 3 个。

学到了什么？

为确保读者已经理解了本节的内容，请试着回答以下问题。如果在解答过程中有任何困难，请回顾本节的相关内容。

- 在机器学习中，使用爬山法前必须指定搜索的哪些特征？
- 哪些搜索算符可以用在“派”问题和“圆”问题中？如何定义评估函数、初态和终态？

1.4 分类器的性能

到目前为止，通过比较训练样例的已知类别和分类器给出的类别，来衡量错误率。从实用来讲，这里的目标不是把类别已知的对象再分类一次，真正想做的是标记未来的样例，它们的类别是不知道的。分类器在这上面的性能表现可通过实验来估计，知道如何做很重要。

独立的测试样例 最简单的做法是把已标记类别的样例分成两部分：训练集（Training Set）从它们中归纳分类器；测试集（Testing Set）在上面做分类器评估（见图1.6）。因此在“派”问题中，对12个预分类样例，可能在随机选出的8个上进行归纳，在其余4个上测试。如果分类器“猜”对了3个测试样例的类别（错了1个），它的性能估计为75%。

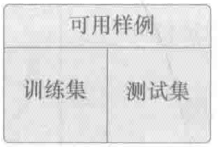


图1.6 预分类样例被分为训练集和测试集

虽然这种方法看起来合理，但有个大的缺陷：随机选出的8个训练样例可能不足以代表所属的概念——对测试集（样例数更少）也是如此。如果用蓝鲸、海豚和鸭嘴兽构成的训练集来归纳哺乳动物的概念，学习器可能认为哺乳动物生活在海洋里（蓝鲸、海豚），有时会产卵（鸭嘴兽），生物学家显然难以认同。不过，另外选择的一组训练样例可能产生符合最高标准的分类器。重点在于不同的训练集/测试集划分产生不同的分类器，以及未来不同的性能估计。预分类样例很少时，这种情况会更严重。

假设想比较两种机器学习算法归纳出的分类器的质量。训练集代表性不足的问题可以用所谓的随机子抽样（Random Subsampling）来减小[⊖]。思想是把随机划分训练集和测试集的过程重复很多次，从第i组训练集中归纳分类器，在第i组测试集中衡量错误率E_i。得到的E_i平均值低的算法更好——分类的性能也要考虑。

做解释的需要 在某些应用中，只给出样例的分类是不够的，还要知道分类的理由。同样，如果手术的依据只是“这是计算机说的”，病人是不会同意做截肢手术的。但如何找到一个更好的解释？

在“派”问题中，从布尔表达式本身可收集到很多信息。例如，可以注意到形状是正

⊖ 后面将介绍其他的方法。